VLSI

VLSI

Edited by
Zhongfeng Wang

In-Tech
intechweb.org

Published by In-Teh

In-Teh
Olajnica 19/2, 32000 Vukovar, Croatia

Abstracting and non-profit use of the material is permitted with credit to the source. Statements and opinions expressed in the chapters are these of the individual contributors and not necessarily those of the editors or publisher. No responsibility is accepted for the accuracy of information contained in the published articles. Publisher assumes no responsibility liability for any damage or injury to persons or property arising out of the use of any materials, instructions, methods or ideas contained inside. After this work has been published by the In-Teh, authors have the right to republish it, in whole or part, in any publication of which they are an author or editor, and the make other personal use of the work.

© 2010 In-teh
www.intechweb.org
Additional copies can be obtained from:
publication@intechweb.org

First published February 2010
Printed in India

Technical Editor: Melita Horvat
Cover designed by Dino Smrekar

VLSI,
Edited by Zhongfeng Wang

p. cm.
ISBN 978-953-307-049-0

Preface

The process of integrated circuits (IC) started its era of very-large-scale integration (VLSI) in 1970's when thousands of transistors were integrated into a single chip. Since then, the transistors counts and clock frequencies of state-of-art chips have grown by orders of magnitude. Nowadays we are able to integrate more than a billion transistors into a single device. However, the term "VLSI" remains being commonly used, despite of some effort to coin a new term ultralarge- scale integration (ULSI) for finer distinctions many years ago. In the past two decades, advances of VLSI technology have led to the explosion of computer and electronics world. VLSI integrated circuits are used everywhere in our everyday life, including microprocessors in personal computers, image sensors in digital cameras, network processors in the Internet switches, communication devices in smartphones, embedded controllers in automobiles, et al.

VLSI covers many phases of design and fabrication of integrated circuits. In a complete VLSI design process, it often involves system definition, architecture design, register transfer language (RTL) coding, pre- and post-synthesis design verification, timing analysis, and chip layout for fabrication. As the process technology scales down, it becomes a trend to integrate many complicated systems into a single chip, which is called system-on-chip (SoC) design. In addition, advanced VLSI systems often require high-speed circuits for the ever increasing demand of data processing. For instance, Ethernet standard has evolved from 10 Mbps to 10 Gbps, and the specification for 100 Gbps Ethernet is underway. On the other hand, with the growing popularity of smartphones and mobile computing devices, low-power VLSI systems have become critically important. Therefore, engineers are facing new challenges to design highly integrated VLSI systems that can meet both high performance requirement and stringent low power consumption.

The goal of this book is to elaborate the state-of-art VLSI design techniques at multiple levels. At device level, researchers have studied the properties of nano-scale devices and explored possible new material for future very high speed, low-power chips. At circuit level, interconnect has become a contemporary design issue for nano-scale integrated circuits. At system level, hardware-software co-design methodologies have been investigated to coherently improve the overall system performance. At architectural level, researchers have proposed novel architectures that have been optimized for specific applications as well as efficient reconfigurable architectures that can be adapted for a class of applications.

As VLSI systems become more and more complex, it is a great challenge but a significant task for all experts to keep up with latest signal processing algorithms and associated architecture designs. This book is to meet this challenge by providing a collection of advanced algorithms

in conjunction with their optimized VLSI architectures, such as Turbo codes, Low Density Parity Check (LDPC) codes, and advanced video coding standards MPEG4/H.264, et al. Each of the selected algorithms is presented with a thorough description together with research studies towards efficient VLSI implementations. No book is expected to cover every possible aspect of VLSI exhaustively. Our goal is to provide the design concepts through those selected studies, and the techniques that can be adopted into many other current and future applications.

This book is intended to cover a wide range of VLSI design topics – both general design techniques and state-of-art applications. It is organized into four major parts:

- Part I focuses on VLSI design for image and video signal processing systems, at both algorithmic and architectural levels.
- Part II addresses VLSI architectures and designs for cryptography and error correction coding.
- Part III discusses general SoC design techniques as well as system-level design optimization for application-specific algorithms.
- Part IV is devoted to circuit-level design techniques for nano-scale devices.

It should be noted that the book is not a tutorial for beginners to learn general VLSI design methodology. Instead, it should serve as a reference book for engineers to gain the knowledge of advanced VLSI architecture and system design techniques. Moreover, this book also includes many in-depth and optimized designs for advanced applications in signal processing and communications. Therefore, it is also intended to be a reference text for graduate students or researchers for pursuing in-depth study on specific topics.

The editors are most grateful to all coauthors for contributions of each chapter in their respective area of expertise. We would also like to acknowledge all the technical editors for their support and great help.

Zhongfeng Wang, Ph.D.
Broadcom Corp., CA, USA

Xinming Huang, Ph.D.
Worcester Polytechnic Institute, MA, USA

Contents

1

Discrete Wavelet Transform Structures for VLSI Architecture Design

Hannu Olkkonen and Juuso T. Olkkonen
Department of Physics, University of Kuopio, 70211 Kuopio, Finland
VTT Technical Research Centre of Finland, 02044 VTT, Finland

1. Introduction

Wireless data transmission and high-speed image processing devices have generated a need for efficient transform methods, which can be implemented in VLSI environment. After the discovery of the compactly supported discrete wavelet transform (DWT) (Daubechies, 1988; Smith & Barnwell, 1986) many DWT-based data and image processing tools have outperformed the conventional discrete cosine transform (DCT) -based approaches. For example, in JPEG2000 Standard (ITU-T, 2000), the DCT has been replaced by the biorthogonal discrete wavelet transform. In this book chapter we review the DWT structures intended for VLSI architecture design. Especially we describe methods for constructing shift invariant analytic DWTs.

2. Biorthogonal discrete wavelet transform

The first DWT structures were based on the compactly supported conjugate quadrature filters (CQFs) (Smith & Barnwell, 1986), which had nonlinear phase effects such as image blurring and spatial dislocations in multi-resolution analyses. On the contrary, in biorthogonal discrete wavelet transform (BDWT) the scaling and wavelet filters are symmetric and linear phase. The two-channel analysis filters $H_0(z)$ and $H_1(z)$ (Fig. 1) are of the general form

$$\begin{aligned} H_0(z) &= (1+z^{-1})^K P(z) \\ H_1(z) &= (1-z^{-1})^K Q(z) \end{aligned} \tag{1}$$

where the scaling filter $H_0(z)$ has the Kth order zero at $\omega = \pi$. The wavelet filter $H_1(z)$ has the Kth order zero at $\omega = 0$, correspondingly. $P(z)$ and $Q(z)$ are polynomials in z^{-1}. The reconstruction filters $G_0(z)$ and $G_1(z)$ (Fig. 1) obey the well-known perfect reconstruction condition

$$\begin{aligned} H_0(z)G_0(z) + H_1(z)G_1(z) &= 2\,z^{-k} \\ H_0(-z)G_0(z) + H_1(-z)G_1(z) &= 0 \end{aligned} \tag{2}$$

The last condition in (2) is satisfied if we select the reconstruction filters as $G_0(z) = H_1(-z)$ and $G_1(z) = -H_0(-z)$.

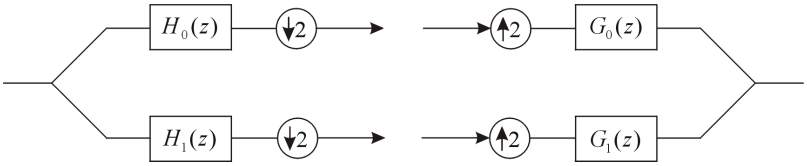


Fig. 1. Analysis and synthesis BDWT filters.

3. Lifting BDWT

The BDWT is most commonly realized by the ladder-type network called lifting scheme (Sweldens, 1988). The procedure consists of sequential down and uplifting steps and the reconstruction of the signal is made by running the lifting network in reverse order (Fig. 2). Efficient lifting BDWT structures have been developed for VLSI design (Olkkonen et al. 2005). The analysis and synthesis filters can be implemented by integer arithmetics using only register shifts and summations. However, the lifting DWT runs sequentially and this may be a speed-limiting factor in some applications (Huang et al., 2005). Another drawback considering the VLSI architecture is related to the reconstruction filters, which run in reverse order and two different VLSI realizations are required. In the following we show that the lifting structure can be replaced by more effective VLSI architectures. We describe two different approaches: the discrete lattice wavelet transform and the sign modulated BDWT.

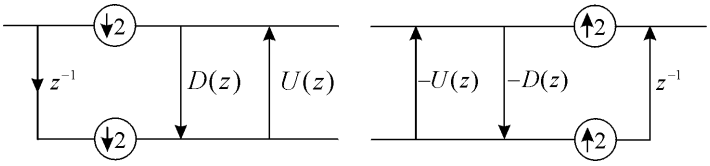


Fig. 2. The lifting BDWT structure.

4. Discrete lattice wavelet transform

In the analysis part the discrete lattice wavelet transform (DLWT) consists of the scaling $H_0(z)$ and wavelet $H_1(z)$ filters and the lattice network (Fig. 3). The lattice structure contains two parallel transmission filters $T_0(z)$ and $T_1(z)$, which exchange information via two crossed lattice filters $L_0(z)$ and $L_1(z)$. In the synthesis part the lattice structure consists of the transmission filters $R_0(z)$ and $R_1(z)$ and crossed filters $W_0(z)$ and $W_1(z)$, and finally the reconstruction filters $G_0(z)$ and $G_1(z)$. Supposing that the scaling and wavelet filters obey (1), for perfect reconstruction the lattice structure should follow the condition

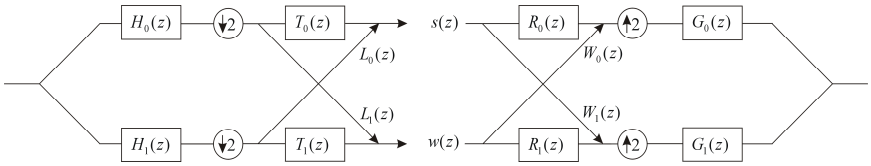


Fig. 3. The general DLWT structure.

$$\begin{bmatrix} T_0R_0 + L_1W_0 & L_0R_0 + T_1W_0 \\ T_0W_1 + L_1R_1 & T_1R_1 + L_0W_1 \end{bmatrix} = \begin{bmatrix} z^{-k} & 0 \\ 0 & z^{-k} \end{bmatrix} \tag{3}$$

This is satisfied if we state $W_0 = -L_0$, $W_1 = -L_1$, $R_0 = T_1$ and $R_1 = T_0$. The perfect reconstruction condition follows then from the diagonal elements (3) as

$$T_0(z)T_1(z) - L_0(z)L_1(z) = z^{-k} \tag{4}$$

There exists many approaches in the design of the DLWT structures obeying (4), for example via the Parks-McChellan-type algorithm. Especially the DLWT network is efficient in designing half-band transmission and lattice filters (see details in Olkkonen & Olkkonen, 2007a). For VLSI design it is essential to note that in the lattice structure all computations are carried out parallel. Also all the BDWT structures designed via the lifting scheme can be transferred to the lattice network (Fig. 3). For example, Fig. 4 shows the DLWT equivalent of the lifting DBWT structure consisting of down and uplifting steps (Fig. 2). The VLSI implementation is flexible due to parallel filter blocks in analysis and synthesis parts.

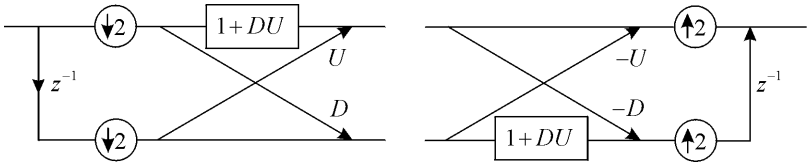


Fig. 4. The DLWT equivalence of the lifting BDWT structure described in Fig. 2.

5. Sign modulated BDWT

In VLSI architectures, where the analysis and synthesis filters are directly implemented (Fig. 1), the VLSI design simplifies considerably using a spesific sign modulator defined as (Olkkonen & Olkkonen 2008)

$$S_n = (-1)^n = \begin{cases} 1 \text{ for n even} \\ \text{-1 for n odd} \end{cases} \tag{5}$$

A key idea is to replace the reconstruction filters by scaling and wavelet filters using the sign modulator (5). Fig. 5 describes the rules how $H(-z)$ can be replaced by $H(z)$ and the sign modulator in connection with the decimation and interpolation operators. Fig. 6

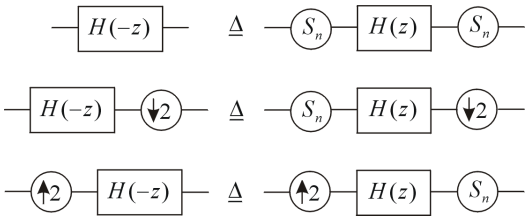


Fig. 5. The equivalence rules applying the sign modulator.

describes the general BDWT structure using the sign modulator. The VLSI design simplifies to the construction of two parallel biorthogonal filters and the sign modulator. It should be pointed out that the scaling and wavelet filters can be still efficiently implemented using the lifting scheme or the lattice structure. The same biorthogonal DWT/IDWT filter module can be used in decomposition and reconstruction of the signal e.g. in video compression unit. Especially in bidirectional data transmission the DWT/IDWT transceiver has many advantages compared with two separate transmitter and receiver units. The same VLSI module can also be used to construct multiplexer-demultiplexer units. Due to symmetry of the scaling and wavelet filter coefficents a fast convolution algorithm can be used for implementation of the filter modules (see details Olkkonen & Olkkonen, 2008).

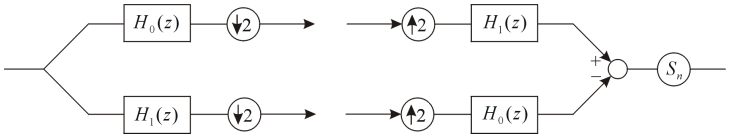


Fig. 6. The BDWT structure using the scaling and wavelet filters and the sign modulator.

6. Design example: Symmetric half-band wavelet filter for compression coder

The general structure for the symmetric half-band filter (HBF) is, for k odd

$$H(z) = z^{-k} + B(z^2) \tag{6}$$

where $B(z^2)$ is a symmetric polynomial in z^{-2}. The impulse response of the HBF contains only one odd point. For example, we may parameterize the eleven point HBF impulse response as $h[n] = [c\ 0\ b\ 0\ a\ 1\ a\ 0\ b\ 0\ c]$, which has three adjustable parameters. The compression efficiency improves when the high-pass wavelet filter approaches the frequency response of the sinc-function, which has the HBF structure. However, the impulse response of the sinc-function is infinite, which prolongs the computation time. In this work we select the seven point compactly supported HBF prototype as a wavelet filter, which has the impulse response

$$h_1[n] = [b\ 0\ a\ 1\ a\ 0\ b] \tag{7}$$

containing two adjustable parameters a and b. In our previous work we have introduced a modified regulatory condition for computation of the parameters of the wavelet filter (Olkkonen et al. 2005)

$$\sum_{n=0}^{N} n^m h_1[n] = 0; m = 0,1,...,M-1 \tag{8}$$

This relation implies that $H_1(z)$ contains Mth-order zero at $z=1$ ($\omega=0$), where M is the number of vanishing moments. Writing (8) for the prototype filter (7) we obtain two equations $2a+2b+1=0$ and $20a+36b+9=0$, which give the solution $a=-9/16$ and $b=1/16$. The wavelet filter has the z-transform

$$H_1(z)=(1-z^{-1})^4(1+4z^{-1}+z^{-2})/16 \tag{9}$$

having fourth order root at z=1. The wavelet filter can be realized in the HBF form

$$H_1(z)=z^{-3}-A(z^2);\ A(z^2)=(-1+9z^{-2}+9z^{-4}-z^{-6})/16 \tag{10}$$

Using the equivalence

$$\left[H(z^2)\right]_{\downarrow 2}\equiv(\downarrow 2)H(z) \tag{11}$$

the HBF structure can be implemented using the lifting scheme (Fig. 7). The functioning of the compression coder can be explained by writing the input signal via the polyphase components

$$X(z)=X_e(z^2)+z^{-1}X_0(z^2) \tag{12}$$

where $X_e(z)$ and $X_o(z)$ denote the even and odd sequences. We may present the wavelet coefficents as

$$W(z)=[X(z)H_1(z)]_{\downarrow 2}=z^{-2}X_o(z)-A(z)X_e(z) \tag{13}$$

$A(z)$ works as an approximating filter yielding an estimate of the odd data points based on the even sequence. The wavelet sequence $W(z)$ can be interpreted as the difference between the odd points and their estimate. In tree structured compression coder the scaling sequence $S(z)$ is fed to the next stage. In many VLSI applications, for example image compression, the input signal consists of an integer-valued sequences. By rounding or truncating the output of the $A(z)$ filter to integers, the compressed wavelet sequence $W(z)$ is integer-valued and can be efficiently coded e.g. using Huffman algorithm. It is essential to note that this integer-to-integer transform has still the perfect reconstruction property (2).

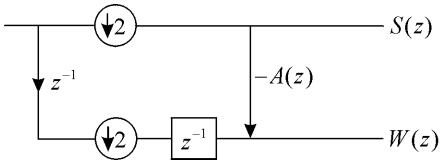


Fig. 7. The lifting structure for the HBF wavelet filter designed for the VLSI compression coder.

7. Shift invariant BDWT

The drawback in multi-scale BWDT analysis of signals and images is the dependence of the total energy of the wavelet coefficients on the fractional shifts of the analysed signal. If we have a discrete signal $x[n]$ and the corresponding time shifted signal $x[n-\tau]$, where $\tau\in[0,1]$, there may exist a significant difference in the energy of the wavelet coefficients as a function of the time shift. Kingsbury (2001) proposed a nearly shift invariant complex wavelet transform, where the real and imaginary wavelet coefficients are approximately Hilbert transform pairs. The energy (absolute value) of the wavelet coefficients equals the envelope,

which warrants smoothness and shift invariance. Selesnick (2002) observed that using two parallel CQF banks, which are constructed so that the impulse responses of the scaling filters are half-sample delayed versions of each other: $h_0[n]$ and $h_0[n-0.5]$, the corresponding wavelets are Hilbert transform pairs. In z-transform domain we should be able to construct the scaling filters $H_0(z)$ and $z^{-0.5}H_0(z)$. However, the constructed scaling filters do not possess coefficient symmetry and in multi-scale analysis the nonlinearity disturbs spatial timing and prevents accurate statistical correlations between different scales. In the following we describe the shift invariant BDWT structures especially designed for VLSI applications.

7.1 Half-delay filters for shift invariant BDWT

The classical approach for design of the half-sample delay filter $D(z)$ is based on the Thiran all-pass interpolator

$$D(z) = z^{-0.5} = \prod_{k=1}^{p} \frac{c_k + z^{-1}}{1 + c_k z^{-1}} \tag{14}$$

where the c_k coefficients are designed so that the frequency response follows approximately

$$D(\omega) = e^{-j\omega/2} \tag{15}$$

Recently, half-delay B-spline filters have been introduced, which have an ideal phase response. The method yields linear phase and shift invariant transform coefficients and can be adapted to any of the existing BDWT (Olkkonen & Olkkonen, 2007b). The half-sample delayed scaling and wavelet filters and the corresponding reconstruction filters are

$$\begin{aligned} \bar{H}_0(z) &= D(z)H_0(z) \\ \bar{H}_1(z) &= D^{-1}(-z)H_1(z) \\ \bar{G}_0(z) &= D^{-1}(z)G_0(z) \\ \bar{G}_1(z) &= D(-z)G_1(z) \end{aligned} \tag{16}$$

The half-delayed BDWT filter bank obeys the perfect reconstruction condition (2). The B-spline half-delay filters have the IIR structure

$$D(z) = \frac{A(z)}{B(z)} \tag{17}$$

which can be implemented by the inverse filtering procedure (see details Olkkonen & Olkkonen 2007b).

7.2 Hilbert transform-based shift invariant DWT

The tree-structured complex DWT is based on the FFT-based computation of the Hilbert transform (H_a operator in Fig. 8). The scaling and wavelet filters both obey the HBF structure (Olkkonen et al. 2007c)

$$\begin{aligned} H_0(z) &= \frac{1}{2} + z^{-1}B(z^2) \\ H_1(z) &= \frac{1}{2} - z^{-1}B(z^2) \end{aligned} \tag{18}$$

For example, the impulse response $h_0[n]=[-1\ 0\ 9\ 16\ 9\ 0\ -1]/32$ has the fourth order zero at $\omega=\pi$ and $h_1[n]=[1\ 0\ -9\ 16\ -9\ 0\ 1]/32$ has the fourth order zero at $\omega=0$. In the tree structured HBF DWT the wavelet sequences $w_a[n]$. A key feature is that the odd coefficients of the analytic signal $w_a[2n+1]$ can be reconstructed from the even coefficient values $w_a[2n]$. This avoids the need to use any reconstruction filters. The HBFs (18) are symmetric with respect to $\omega=\pi/2$. Hence, the energy in the frequency range $0\to\pi$ is equally divided by the scaling and wavelet filters and the energy (absolute value) of the scaling and wavelet

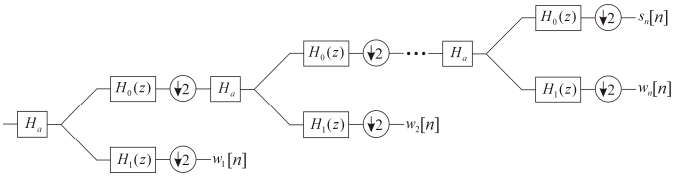


Fig. 8. Hilbert transform-based shift invariant DWT.

coefficients are statistically comparable. The computation of the analytic signal via the Hilbert transform requires the FFT-based signal processing. However, efficient FFT chips are available for VLSI implementation. In many respects the advanced method outperforms the previous nearly shift invariant DWT structures.

7.3 Hilbert transform filter for construction of shift invariant BDWT

The FFT-based implementation of the shift invariant DWT can be avoided if we define the Hilbert transform filter $\mathcal{H}(z)$, which has the frequency response

$$\mathcal{H}(\omega)=e^{-j\pi/2}\,\mathrm{sgn}(\omega) \tag{19}$$

where $\mathrm{sgn}(\omega)=1$ for $\omega\ge 0$ and $\mathrm{sgn}(\omega)=0$ for $\omega<0$. In the following we describe a novel method for constructing the Hilbert transform filter based on the half-sample delay filter $D(z)$ (17), whose frequency response follows (15). The quadrature mirror filter $D(-z)$ has the frequency response

$$D(\omega-\pi)=e^{-j(\omega-\pi)/2} \tag{20}$$

The frequency response of the filter $D(z)D^{-1}(-z)$ is, correspondingly

$$\frac{D(\omega)}{D(\omega-\pi)}=e^{-j\omega/2}e^{j(\omega-\pi)/2}=e^{-j\pi/2} \tag{21}$$

Comparing (19) and using notation (17) we obtain the Hilbert transform filter as

$$\mathcal{H}(z)=\frac{A(z)B(-z)}{A(-z)B(z)} \tag{22}$$

The corresponding parallel BDWT filter bank is

$$\begin{aligned}
\bar{H}_0(z)&=\mathcal{H}(z)H_0(z)\\
\bar{H}_1(z)&=\mathcal{H}^{-1}(-z)H_1(z)\\
\bar{G}_0(z)&=\mathcal{H}^{-1}(z)G_0(z)\\
\bar{G}_1(z)&=\mathcal{H}(-z)G_1(z)
\end{aligned} \tag{23}$$

By filtering the real-valued signal $X(z)$ by the Hilbert transform filter results in an analytic signal $[1+j\mathcal{H}(z)]X(z)$, whose magnitude response is zero at negative side of the frequency spectrum. For example, an integer-valued half-delay filter $D(z)$ for this purpose is obtained by the B-spline transform (Olkkonen & Olkkonen, 2007b). The frequency response of the Hilbert transform filter designed by the fourth order B-spline (Fig. 9) shows a maximally flat magnitude spectrum. The phase spectrum corresponds to the ideal Hilbert transformer (19).

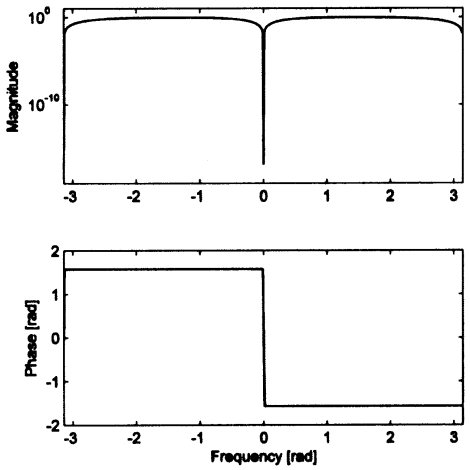


Fig. 9. Magnitude and phase spectra of the Hilbert transform filter yielded by the fourth order B-spline transform.

8. Conclusion

In this book chapter we have described the BDWT constructions especially tailored for VLSI environment. Most of the VLSI designs in the literature are focused on the biorthogonal 9/7 filters, which have decimal coefficients and usually implemented using the lifting scheme (Sweldens, 1988). However, the lifting BDWT needs two different filter banks for analysis and synthesis parts. The speed of the lifting BDWT is also limited due to the sequential lifting steps. In this work we showed that the lifting BDWT can be replaced by the lattice structure (Olkkonen & Olkkonen, 2007a). The two-channel DLWT filter bank (Fig. 3) runs parallel, which significantly increases the channel throughout. A significant advantage compared with the previous maximally decimated filter banks is that the DLWT structure allows the construction of the half-band lattice and transmission filters. In tree structured wavelet transform half-band filtered scaling coefficients introduce no aliasing when they are fed to the next scale. This is an essential feature when the frequency components in each scale are considered, for example in electroencephalography analysis.

The VLSI design of the BDWT filter bank simplifies essentially by implementing the sign modulator unit (Fig. 5), which eliminates the need for constructing separate reconstruction filters. The biorthogonal DWT/IDWT transceiver module uses only two parallel filter structures. Especially in bidirectional data transmission the DWT/IDWT module offers several advantages compared with the separate transmit and receive modules, such as the

reduced size, low power consumption, easier synchronization and timing requirements. For the VLSI designer the DWT/IDWT module appears as a "black box", which readily fits to the data under processing. This may override the relatively big barrier from the wavelet theory to the practical VLSI and microprocessor applications. As a design example we described the construction of the compression coder (Fig. 7), which can be used to compress integer-valued data sequences, e.g. produced by the analog-to-digital converters.

It is well documented that the real-valued DWTs are not shift invariant, but small fractional time-shifts may introduce significant differences in the energy of the wavelet coefficients. Kingsbury (2001) showed that the shift invariance is improved by using two parallel filter banks, which are designed so that the wavelet sequences constitute real and imaginary parts of the complex analytic wavelet transform. The dual-tree discrete wavelet transform (DT-DWT) has been shown to outperform the real-valued DWT in a variety of applications such as denoising, texture analysis, speech recognition, processing of seismic signals and neuroelectric signal analysis (Olkkonen et al. 2006). Selesnick (2002) made an observation that a half-sample time-shift between the scaling filters in parallel CQF banks is enough to produce the analytic wavelet transform, which is nearly shift invariant. In this work we described the shift invariant DT-BDWT bank (16) based on the half-sample delay filter. It should be pointed out that the half-delay filter approach yields wavelet bases which are Hilbert transform pairs, but the wavelet sequences are only approximately shift invariant. In multi-scale analysis the complex wavelet sequences should be shift invariant. This requirement is satisfied in the Hilbert transform-based approach (Fig. 8), where the signal in every scale is Hilbert transformed yielding strictly analytic and shift invariant transform coefficients. The procedure needs FFT-based computation (Olkkonen et al. 2007c), which may be an obstacle in many VLSI realizations. To avoid this we described a Hilbert transform filter for constructing the shift invariant DT-BDWT bank (23). Instead of the half-delay filter bank approach (16) the perfect reconstruction condition (2) is attained using the IIR-type Hilbert transform filters, which yield analytic wavelet sequences.

9. References

Daubechies, I. (1988). Orthonormal bases of compactly supported wavelets. *Commmun. Pure Appl. Math.,* Vol. 41, 909-996.

Huang, C.T., Tseng, O.O. & Chen, L.G. (2005). Analysis and VLSI architecture for 1-D and 2-D discrete wavelet transform. *IEEE Trans. Signal Process.* Vol. 53, No. 4, 1575-1586.

ITU-T (2000) Recommend. T.800-ISO DCD15444-1: *JPEG2000 Image Coding System. International Organization for Standardization,* ISO/IEC JTC! SC29/WG1.

Kingsbury, N.G. (2001). Complex wavelets for shift invariant analysis and filtering of signals. *J. Appl. Comput. Harmonic Analysis.* Vol. 10, 234-253.

Olkkonen, H., Pesola, P. & Olkkonen, J.T. (2005). Efficient lifting wavelet transform for microprocessor and VLSI applications. *IEEE Signal Process. Lett.* Vol. 12, No. 2, 120-122.

Olkkonen, H., Pesola, P., Olkkonen, J.T. & Zhou, H. (2006). Hilbert transform assisted complex wavelet transform for neuroelectric signal analysis. *J. Neuroscience Meth.* Vol. 151, 106-113.

Olkkonen, J.T. & Olkkonen, H. (2007a). Discrete lattice wavelet transform. *IEEE Trans. Circuits and Systems II.* Vol. 54, No. 1, 71-75.

Olkkonen, H. & Olkkonen, J.T. (2007b). Half-delay B-spline filter for construction of shift-invariant wavelet transform. *IEEE Trans. Circuits and Systems II.* Vol. 54, No. 7, 611-615.

Olkkonen, H., Olkkonen, J.T. & Pesola, P. (2007c). FFT-based computation of shift invariant analytic wavelet transform. *IEEE Signal Process. Lett.* Vol. 14, No. 3, 177-180.

Olkkonen, H. & Olkkonen, J.T. (2008). Simplified biorthogonal discrete wavelet transform for VLSI architecture design. *Signal, Image and Video Process.* Vol. 2, 101-105.

Selesnick, I.W. (2002). The design of approximate Hilbert transform pairs of wavelet bases. *IEEE Trans. Signal Process.* Vol. 50, No. 5, 1144-1152.

Smith, M.J.T. & Barnwell, T.P. (1986). Exaxt reconstruction for tree-structured subband coders. *IEEE Trans. Acoust. Speech Signal Process.* Vol. 34, 434-441.

Sweldens, W. (1988). The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.* Vol. 29, 511-546.

2

High Performance Parallel Pipelined Lifting-based VLSI Architectures for Two-Dimensional Inverse Discrete Wavelet Transform

Ibrahim Saeed Koko and Herman Agustiawan
Electrical & Electronic Engineering Department
Universiti Teknologi PETRONAS, Tronoh
Malaysia

1. Introduction

Two-dimensional discrete wavelet transform (2-D DWT) has evolved as an effective and powerful tool in many applications especially in image processing and compression. This is mainly due to its better computational efficiency achieved by factoring wavelet transforms into lifting steps. Lifting scheme facilitates high speed and efficient implementation of wavelet transform and it attractive for both high throughput and low-power applications (Lan & Zheng, 2005).

DWT considered in this work is part of a compression system based on wavelet such as JPEG2000. Fig.1 shows a simplified compression system. In this system, the function of the 2-D FDWT is to decompose an *NxM* image into subbands as shown in Fig. 2 for 3-level decomposition. This process decorrelates the highly correlated pixels of the original image. That is the decorrelation process reduces the spatial correlation among the adjacent pixels of the original image such that they can be amenable to compression.

After transmitting to a remote site, the original image must be reconstructed from the decorrelated image. The task of the reconstruction and completely recovering the original image from the decorrelated image is performed by inverse discrete wavelet transform (IDWT).

The decorrelated image shown in Fig. 2 can be reconstructed by 2-D IDWT as follows. First, it reconstructs in the column direction subbands LL3 and LH3 column-by-column to recover L3 decomposition. Similarly, subbands HL3 and HH3 are reconstructed to obtain H3 decomposition. Then L3 and H3 decompositions are combined row-wise to reconstruct subband LL2. This process is repeated in each level until the whole image is reconstructed (Ibrahim & Herman, 2008).

The reconstruction process described above implies that the task of the reconstruction can be achieved by using 2 processors (Ibrahim & Herman, 2008). The first processor (the column-processor) computes column-wise to combine subbands LL and LH into L and subbands HL and HH into H, while the second processor (the row-processor) computes row-wise to

combine L and H into the next level subband. The decorrelated image shown in Fig. 2 is assumed to be residing in an external memory with the same format.
In this chapter, parallelism is explored to best meet real-time applications of 2-D DWT with demanding requirements. The single pipelined architecture developed by (Ibrahim & Herman, 2008) is extended to 2- and 4-parallel pipelined architectures for both 5/3 and 9/7 inverse algorithms to achieve speedup factors of 2 and 4, respectively. The advantage of the proposed architectures is that the total temporary line buffer (TLB) size does not increase from that of the single pipelined architecture when degree of parallelism is increased.

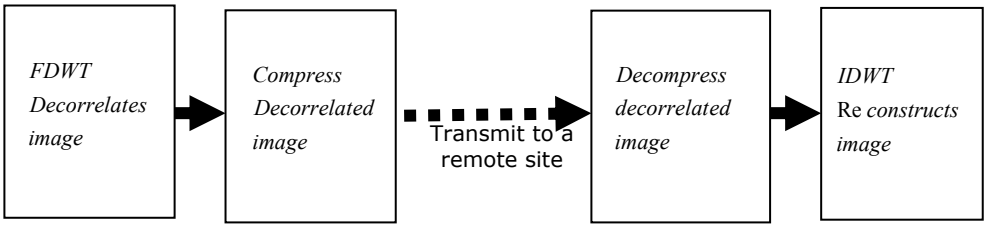


Fig. 1. A simplified Compression System

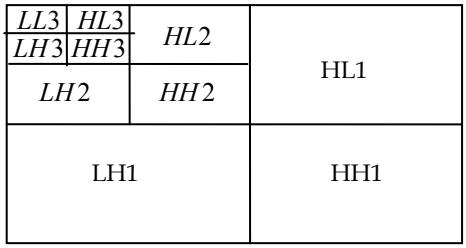


Fig. 2. Subband decomposition of an *NxM* image into 3 levels.

2. Lifting-based 5/3 and 9/7 synthesis algorithms and data dependency graphs

The 5/3 and the 9/7 inverse discrete wavelet transforms algorithms are defined by the JPEG2000 image compression standard as follow (Lan & Zheng, 2005):

5/3 synthesis algorithm

$$step1: X(2n) = Y(2n) - \left\lfloor \frac{Y(2n-1) + Y(2n+1) + 2}{4} \right\rfloor$$
$$step2: X(2n+1) = Y(2n+1) + \left\lfloor \frac{X(2n) + X(2n+2)}{2} \right\rfloor \quad (1)$$

9/7 synthesis algorithm

$$\text{Step 1: } Y'(2n) = 1/k \cdot Y(2n)$$
$$\text{Step 2: } Y'(2n+1) = k \cdot Y(2n+1)$$
$$\text{Step 3: } Y''(2n) = Y'(2n) - \delta(Y'(2n-1) + Y'(2n+1))$$
$$\text{Step 4: } Y''(2n+1) = Y'(2n+1) - \gamma(Y''(2n) + Y''(2n+2)) \quad (2)$$

Step 5: $X(2n) = Y''(2n) - \beta(Y''(2n-1) + Y''(2n+1))$

Step 6: $X(2n+1) = Y''(2n+1) - \alpha(X(2n) + X(2n+2))$

The data dependency graphs (DDGs) for 5/3 and 9/7 derived from the synthesis algorithms are shown in Figs. 3 and 4, respectively. The DDGs are very useful tools in architecture development and provide the information necessary for designer to develop more accurate architectures. The symmetric extension algorithm recommended by JPEG2000 is incorporated into the DDGs to handle the boundaries problems. The boundary treatment is necessary to keep number of wavelet coefficients the same as that of the original input Pixels and as a result prevent distortion from appearing at image boundaries. The boundary treatment is only applied at the beginning and ending of each row or column in an *NxM* image (Dillin et al., 2003). In DDGs, nodes circled with the same numbers are considered redundant computations, which will be computed once and used thereafter. Coefficients of the nodes circled with even numbers in the DDGs are low coefficients and that circled with odd numbers are high coefficients.

3. Scan methods

The hardware complexity and hence the memory required for 2-D DWT architecture in general depends on the scan method adopted for scanning external memory. Therefore, the scan method shown in Fig. 5 is proposed for both 5/3 and 9/7 CPs. Fig. 5 (A) is formed for illustration purposes by merging together subbands LL and LH, where subband LL coefficients occupy even row and subband LH coefficients occupy odd rows, while Fig. 5 (B) is formed by merging subband HL and HH together.

According to the scan method shown in Fig. 5, CPs of both 5/3 and 9/7 should scan external memory column-by-column. However, to allow the RP, which operates on data generated by the CP, to work in parallel with the CP as earlier as possible, the A's (LL+LH) first two columns coefficients are interleaved in execution with the B's (HL+HH) first two columns coefficients, in the first run. In all subsequent runs, two columns are interleaved, one from A with another from B.

Interleaving of 4 columns in the first run takes place as follows. First, coefficients LL0,0, LH0,0 from the first column of (*A*) are scanned. Second, coefficients HL0,0 and HH0,0 from the first column of *B* are scanned, then LL0,1 and LH0,1 from the second column of *A* followed by HL0,1 and HH0,1 from the second column of *B* are scanned. The scanning process then returns to the first Fig. 3. 5/3 synthesis algorithm's DDGs for (a) odd and (b) even length signals returns to the first column of *A* to repeat the process and so on.

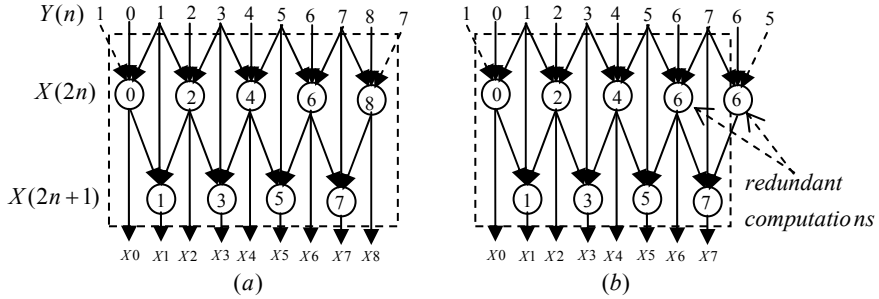


Fig. 3. 5/3 synthesis algorithm's DDGs for (a) odd and (b) even length signals

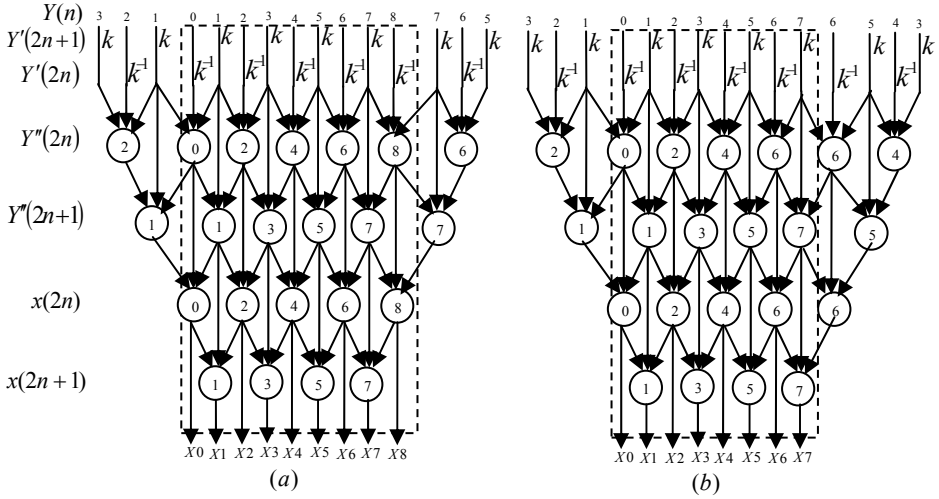


Fig. 4. 9/7 synthesis algorithm's DDGs for (a) odd and (b) even length signals

The advantage of interleaving process not only it speedups the computations by allowing the two processors to work in parallel earlier during the computations, but also reduces the internal memory requirement between CP and RP to a few registers.

The scan method illustrated in Fig. 5 for 5/3 and 9/7 CP along with the DDGs suggest that the RP should scan coefficients generated by CP according to the scan method illustrated in Fig. 6. This figure is formed for illustration purposes by merging L and H decompositions even though they are actually separate. In Fig. 6, L's coefficients occupy even columns, while H's coefficients occupy odd columns. In the first run, the RP's scan method shown in Fig. 6 requires considering the first four columns for scanning as follows. First, coefficients L0,0 and H0,0 from row 0 followed by L1,0 and H1,0 from row 1 are scanned. Then the scan returns to row 0 and scans coefficients L0,1 and H0,1 followed by L1,1 and H1,1. This process is repeated as shown in Fig. 6 until the first run completes.

In the second run, coefficients of columns 4 and 5 are considered for scanning by RP as shown in Fig. 6, whereas in the third run, coefficients of columns 6 and 7 are considered for scanning and so on.

According to the 9/7 DDGs, the RP's scan method shown in Fig. 6 will generate only one low output coefficient each time it processes 4 coefficients in the first run, whereas 5/3 RP will generate two output coefficients. However, in all subsequent runs, both 9/7 and 5/3 RPs generate two output coefficients.

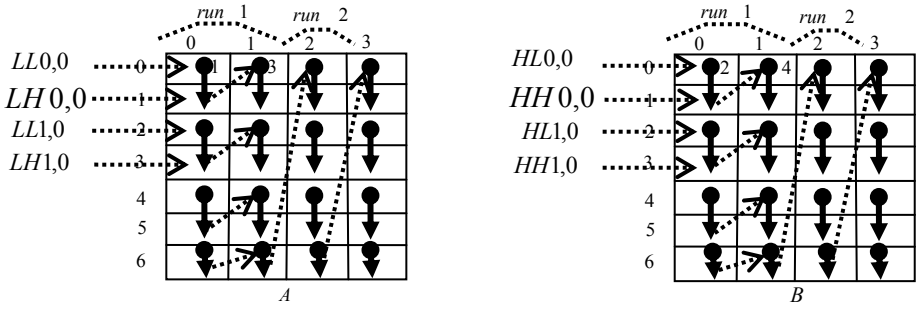


Fig. 5. 5/3 and 9/7 CPs scan method

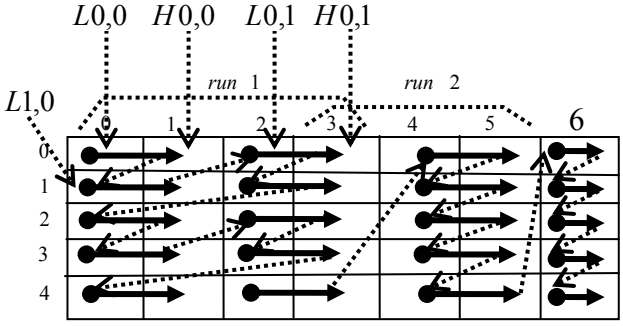


Fig. 6. 5/3 and 9/7 RPs scan method

4. Approach

In (Ibrahim & Herman, 2008), to ease the architecture development the strategy adopted was to divide the details of the development into two steps each having less information to handle. In the first step, the DDGs were looked at from the outside, which is specified by the dotted boxes in the DDGs, in terms of the input and output requirements. It can be observed that the DDGs for 5/3 and 9/7 are identical when they are looked at from outside, taking into consideration only the input and output requirements; but differ in the internal details. Based on this observation, the first level, called external architecture, which is identical for both 5/3 and 9/7, and consists of a column-processor (CP) and a row-processor (RP), was developed. In the second step, the internal details of the DDGs for 5/3 and 9/7 were considered separately for the development of processors' datapath architectures, since DDGs internally define and specify the internal structure of the processors

In this chapter, the first level, the external architectures for 2-parallel and 4-parallel are developed for both 5/3 and 9/7 inverse algorithms. Then the processors' datapath

architectures developed in (Ibrahim & Herman, 2008) are modified to fit into the two proposed parallel architectures' processors.

4.1 Proposed 2-parallel external architecture

Based on the scan methods shown in Figs. 5 and 6 and the DDGs for 5/3 and 9/7, the 2-parallel external architecture shown in Fig. 7 (a) is proposed for 5/3 and 9/7 and combined 5/3 and 9/7 for 2-D IDWT. The architecture consists of two *k*-stage pipelined column-processors (CPs) labeled CP1 and CP2 and two *k*-stage pipelined row-processors (RPs) labeled RP1 and RP2. The waveforms of the two clocks f_2 and $f_2/2$ used in the architecture are shown in Fig. 7 (b).

In general, the scan frequency f_l and hence the period $\tau_l = 1/f_l$ of the parallel architectures can be determined by the following algorithm when the required pixels I of an operation are scanned simultaneously in parallel. Suppose t_p and t_m are the processor and the external memory critical path delays, respectively.

$$\begin{aligned} &If\ t_p/l \cdot k \geq t_m \quad then \\ &\qquad \tau_l = t_p/l \cdot k \\ &else \quad \tau_l = t_m \end{aligned} \tag{3}$$

Where l = 2, 3, 4 ... denote 2, 3, and 4-parallel and t_p/k is the stage critical path delay of a k-stage pipelined processor. The clock frequency f_2 is determined from Eq(3) as

$$f_2 = 2k/t_p \tag{4}$$

The architecture scans the external memory with frequency f_2 and it operates with frequency $f_2/2$. Each time two coefficients are scanned through the two buses labeled *bus0* and *bus1*. The two new coefficients are loaded into CP1 or CP2 latches Rt0 and Rt1 every time clock $f_2/2$ makes a negative or a positive transition, respectively.

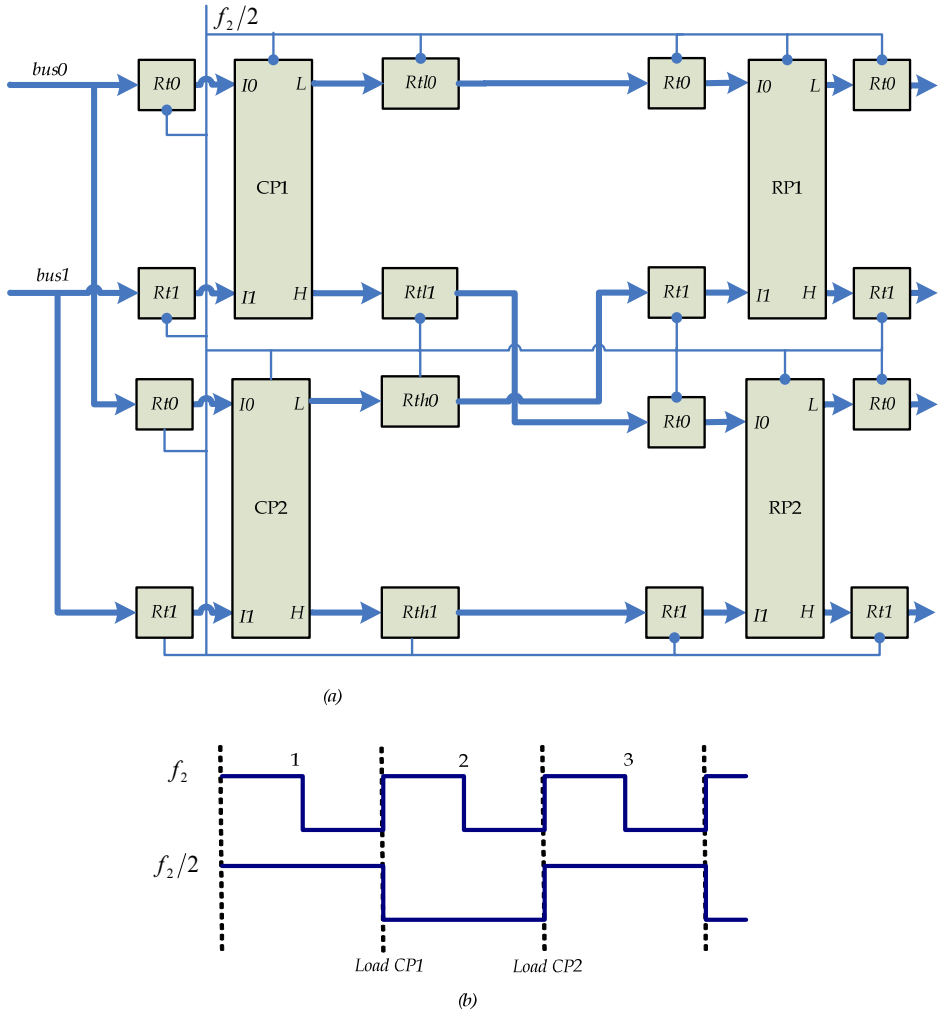


Fig. 7. (a) Proposed 2-parallel pipelined external architecture for 5/3 and 9/7 and combined (b) Waveform of the clocks.

On the other hand, both RP1 and RP2 latches Rt0 and Rt1 load simultaneously new data from CP1 and CP2 output latches each time clock $f_2/2$ makes a negative transition.

The dataflow for 5/3 2-parallel architecture is shown in Table 1, where CPs and RPs are assumed to be 4-stage pipelined processors. The dataflow for 9/7 2-parallel architecture is similar, in all runs, to the 5/3 dataflow except in the first, where RP1 and RP2 of the 9/7 architecture each would generate one output coefficient every other clock cycle, reference to clock $f_2/2$. The reason is that each 4 coefficients of a row processed in the first run by RP1 or RP2 of the 9/7 would require, according to the DDGs, two successive low coefficients

from the first level of the DDGs labeled $Y''(2n)$ in order to carry out node 1 computations in the second level labeled $Y''(2n+1)$. In Table 1, the output coefficients in Rt0 of both RP1 and RP2 represent the output coefficients of the 9/7 in the first run.

The strategy adopted for scheduling memory columns for CP1 and CP2 of the 5/3 and 9/7 2-parallel architectures, which are scanned according to the scan method shown in Fig. 5, is as follow. In the first run, both 5/3 and 9/7 2-parallel architectures are scheduled for executing 4 columns of memory, two from each (A) and (B) of Fig. 5. The first two columns of Fig. 5 (A) are executed in an interleaved manner by CP1, while the first two columns of Fig. 5 (B) are executed by CP2 also in an interleaved fashion as shown in the dataflow Table 1 In all other runs, 2 columns are scheduled for execution at a time. One column from (A) of Fig. 5 will be scheduled for execution by CP1, while another from (B) of Fig. 5 will be scheduled for CP2. However, if number of columns in (A) and (B) of Fig. 5 is not equal, then the last run will consist of only one column of (A). In that case, schedule the last column in CP1 only, but its output coefficients will be executed by both RP1 and RP2. The reason is that if the last column is scheduled for execution by both CP1 and CP2, they will yield more coefficients than that can be handled by both RP1 and RP2.

On the other hand, scheduling RP1 and RP2 of 5/3 and 9/7 2-parallel architectures occur according to scan method shown in Fig. 6. In this scheduling strategy, all rows of even and odd numbers in Fig. 6 will be scheduled for execution by RP1 and RP2, respectively. In the first run, 4 coefficients from each 2 successive rows will be scheduled for RP1 and RP2, whereas in all subsequent runs, two coefficients of each 2 successive rows will be scheduled for RP1 and RP2, as shown in Fig. 6. However, if number of columns in Fig. 6 is odd which occur when number of columns in (A) and (B) of Fig. 5 is not equal, then the last run would require scheduling one coefficient from each 2 successive rows to RP1 and RP2 as shown in column 6 of Fig. 6.

In general, all coefficients that belong to columns of even numbers in Fig. 6 will be generated by CP1 and that belong to columns of odd numbers will be generated by CP2. For example, in the first run, CP1will first generate two coefficients labeled L0,0 and L1,0 that belong to locations 0,0 and 1,0 in Fig. 6, while CP2 will generate coefficient H0,0 and H1,0 that belong to locations 0,1 and 1,1. Then coefficients in locations 0,0 and 0,1 are executed by RP1, while coefficients of locations 1,0 and 1,1 are executed by RP2. Second, CP1 will generate two coefficients for locations 0,2 and 1,2, while CP2 generates two coefficients for locations 0,3 and 1,3. Then coefficients in locations 0,2 and 0,3 are executed by RP1, while coefficients in locations 1,2 and 1,3 are executed by RP2. The same process is repeated in the next two rows and so on.

In the second run, first, CP1 generates coefficients of locations 0,4 and 1,4, whereas CP2 generates coefficients of locations 0,5 and 1,5 in Fig. 6. Then coefficients in locations 0,4 and 0,5 are executed by RP1, while coefficients in locations 1,4 and 1,5 are executed by RP2. This process is repeated until the run completes. However, in the even that the last run processes only one column of (A), CP1 would generate first coefficients of locations 0,m and 1,m where m refers to the last column. Then coefficients of location 0,m is passed to RP1, while coefficient of location 1,m is passed to RP2. In the second time, CP1 would generate coefficients of locations 2,m and 3,m. Then 2,m is passed to RP1 and 3,m to RP2 and so on.

In the following, the dataflow shown in Table 1 for 2-parallel pipelined architecture will be explained. The first run, which ends at cycle 16 in the table, requires scheduling four columns as follows. In the first clock cycle, reference to clock f_2, coefficients LL0,0 and

LH0,0 from the first column of LL3 and LH3 in the external memory, respectively, are scanned and are loaded into CP1 latches Rt0 and Rt1 by the negative transition of clock $f_2/2$. The second clock cycle scans coefficients HL0,0 and HH0,0 from the first column of HL3 and HH3, respectively, through the buses labeled *bus0* and *bus*1 and loads them into CP2 latches Rt0 and Rt1 by the positive transition of clock $f_2/2$. In the third clock cycle, the scanning process returns to the second column of subbands LL3 and LH3 in the external memory and scans coefficients LL0,1 and LH0,1, respectively, and loads them into CP1 latches Rt0 and Rt1 by the negative transition of the clock $f_2/2$. The fourth clock cycle scans coefficients HL0,1 and HH0,1 from the second column of HL3 and HH3, respectively, and loads them into CP2 latches Rt0 and Rt1. The scanning process then returns to the first column in subbands LL3 and LH3 to repeat the process until the first run is completed.

In cycle 9, CP1 generates its first two output coefficients L0,0 and L1,0, which belong to L3 decomposition and loads them into its output latches Rtl0 and Rtl1, respectively, by the negative transition of clock $f_2/2$. In cycle 10, CP2 generates its first two output coefficients H0,0 and H1,0, which belong to H3 decomposition and loads them into its output latches Rth0 and Rth1, respectively, by the positive transition of clock $f_2/2$.

In cycle 11, contents of Rtl0 and Rth0 are transferred to RP1 input latches Rt0 and Rt1, respectively. The same clock cycle also transfers contents of Rtl1 and Rth1 to RP2 input latches Rt0 and Rt1, respectively, while the two coefficients L0,1 and L1,1 generated by CP1 during the cycle are loaded into Rtl0 and Rtl1, respectively, by the negative transition of clock $f_2/2$.

In cycle 21, both RP1 and RP2 each yield its first two output coefficients, which are loaded into their respective output latches by the negative transition of clock $f_2/2$. Contents of these output latches are then transferred to external memory where they are stored in the first 2 memory locations of each rows 0 and 1. The dataflow of the first run then proceeds as shown in Table 1. The second run begins at cycle 19 and yields its first 4 output coefficients at cycle 37.

	Ck f_2	CP	CP1 & CP2 input latches Rt0 Rt1	CP1 output latches Rtl0 Rtl1	CP2 output latches Rth0 Rth1	RP1 input latches Rt0 Rt1	RP2 input latches Rt0 Rt1	Output latches of RP1 RP2 Rt0 Rt1 Rt0 Rt1
RUN 1	1	1	LL0,0 LH0,0					
	2	2	HL0,0 HH0,0					
	3	1	LL0,1 LH0,1					
	4	2	HL0,1 HH0,1					
	5	1	LL1,0 LH1,0					
	6	2	HL1,0 HH1,0					
	7	1	LL1,1 LH1,1					
	8	2	HL1,1 HH1,1					
	9	1	LL2,0 LH2,0	L0,0 L1,0				
	10	2	HL2,0 HH2,0		H0,0 H1,0			
	11	1	LL2,1 LH2,1	L0,1 L1,1		L0,0 H0,0	L1,0 H1,0	
	12	2	HL2,1 HH2,1		H0,1 H1,1			
	13	1	LL3,0 LH3,0	L2,0 L3,0		L0,1 H0,1	L1,1 H1,1	
	14	2	HL3,0 HH3,0		H2,0 H3,0			
	15	1	LL3,1 LH3,1	L2,1 L3,1		L2,0 H2,0	L3,0 H3,0	
	16	2	HL3,1 HH3,1		H2,1 H3,1			
RUN 2	17	1	------ ------- -	L4,0 L5,0		L2,1 H2,1	L3,1 H3,1	
	18	2	------ -------		H4,0 H5,0			
	19	1	LL0,2 LH0,2	L4,1 L5,1		L4,0 H4,0	L5,0 H5,0	
	20	2	HL0,2 HH0,2		H4,1 H5,1			
	21	1	LL1,2 LH1,2	L6,0 L7,0		L4,1 H4,1	L5,1 H5,1	X0,0 X0,1 X1,0 X1,1
	22	2	HL1,2 HH1,2		H6,0 H7,0			
	23	1	LL2,2 LH2,2	L6,1 L7,1		L6,0 H6,0	L7,0 H7,0	X0,2 ---- X1,2 -----

	24	2	HL2,2 HH2,2		H6,1 H7,1			
	25	1	LL3,2 LH3,2	----- ------		L6,1 H6,1	L7,1 H7,1	X2,0 X2,1 X3,0 X3,1
	26	2	HL3,2 HH3,2		----- -----			
	27	1		L0,2 L1,2		----- ---- -	----- -- ----	X2,2 ---- X3,2 -----
	28	2			H0,2 H1,2			
	29	1		L2,2 L3,2		L0,2 H0,2	L1,2 H1,2	X4,0 X4,1 X5,0 X5,1
	30	2			H2,2 H3,2			
	31	1		L4,2 L5,2		L2,2 H2,2	L3,2 H3,2	X4,2 ---- X5,2 -----
	32	2			H4,2 H5,2			
	33	1		L6,2 L7,2		L4,2 H4,2	L5,2 H5,2	X6,0 X6,1 X7,0 X7,1
	34	2			H6,2 H7,2			
	35	1				L6,2 H6,2	L7,2 H7,2	X6,2 ---- X7,2 -----
	36	2						
	37	1						X0,3 X0,4 X1,3 X1,4
	38	2						
	39	1						X2,3 X2,4 X3,3 X3,4

Table 1. Dataflow for 2-parallel 5/3 architecture

4.2 Modified CPs and RPs for 5/3 and 9/7 2-parallel external architecture

Each CP of the 2-parallel external architecture is required to execute two columns in an interleave fashion in the first run and one column in all other runs. Therefore, the 5/3 processor datapath developed in (Ibrahim & Herman, 2008) should be modified as shown in Fig. 8 by adding one more stage between stages 2 and 3 for 5/3 2- parallel external architecture to allow interleaving of two columns as described in the dataflow Table 1. Through the two multiplexers labeled *mux* the processor controls between executing 2 columns and one column. Thus, in the first run, the two multiplexers' control signal labeled s is set 1 to allow interleaving in execution and 0 in all other runs. The modified 9-stage CP for 9/7 2-parallel external architecture can be obtained by cascading two copies of Fig. 8.

On the other hand, RP1 and RP2 of the proposed 2-parallel architecture for 5/3 and 9/7 are required to scan coefficients of H and L decompositions generated by CP1 and CP2 according to the scan method shown in Fig. 6. In this scan method, all rows of even numbers are executed by RP1 and all rows of odds numbers are executed by RP2. That is, while RP1 is executing row0 coefficients, RP2 will be executing row1 coefficients and so on. In addition, looking at the DDGs for 5/3 and 9/7 show that applying the scan methods in Fig.

6 would require inclusion of temporary line buffers (TLBs) in RP1 and RP2 of the proposed 2-parallel external architecture as follows. In the first run, the fourth input coefficient of each row in the DDGs and the output coefficients labeled X(2) in the 5/3 DDGs and that labeled Y"(2), Y"(1), and X(0) in the 9/7 DDGs, generated by considering 4 inputs coefficients in each row, should be stored in TLBs, since they are required in the next run's computations. Similarly, in the second run, the sixth input coefficient of each row and the output coefficients labeled X(4) in the 5/3 DDGs and that labeled Y"(4), Y"(3), and X(2) in the 9/7 DDGs generated by considering 2 inputs coefficients in each row, should be stored in TLBs. Accordingly, 5/3 would require addition of 2 TLBs each of size N, whereas 9/7 would require addition of 4 TLBs each of size N. However, since 2-parallel architecture consists of two RPs, each 5/3 RP will have 2 TLBs each of size N/2 and each 9/7 RP will have 4 TLBs each of size N/2 as shown in Fig. 9. Fig. 9 (a) represents the 5/3 modified RP, while both (a) and (b) represent the 9/7 modified RP for 2- parallel architecture.

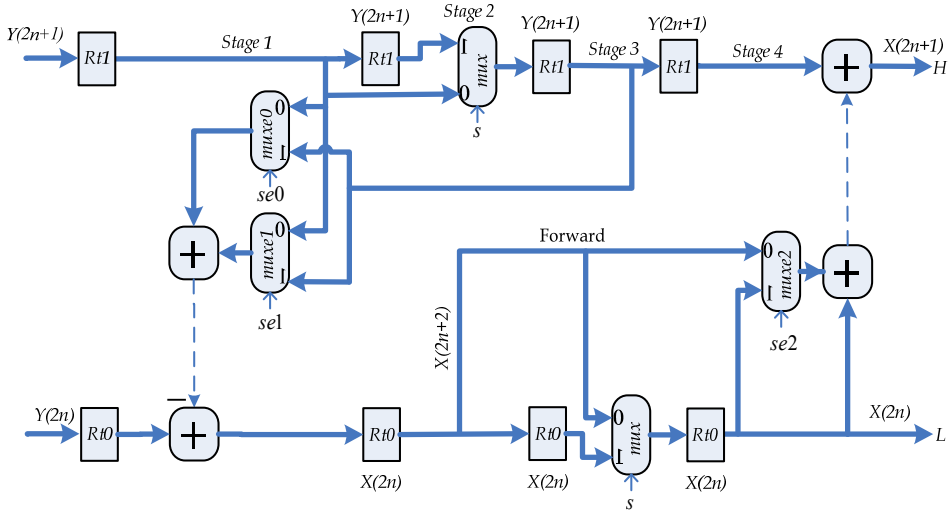


Fig. 8. Modified inverse 5/3 CP for 2-parallel External architecture

To have more insight into the two RPs operations, the dataflow for 5/3 RP1 is given in Table 2 for first and second runs. Note that stage 1 input coefficients in Table 2 are exactly the same input coefficients of RP1 in Table 1. In the first run, TLBs are only written, but in the second run and in all subsequent runs, TLBs are read in the first half cycle and written in the second half cycle. In the cycle 15, Table 2 shows that coefficients H0,1 is stored in the first location of TLB1, while coefficient H2,1 is stored in the second location in cycle 19 and so on. Run 2 starts at cycle 29. In cycle 30, the first location of TLB1, which contains coefficients H0,1 is read during the first half cycle of clock $f_2/2$ and is loaded into Rd1 by the positive transition of the clock, whereas coefficient H0,2 is written into the same location in the second half cycle. Then, the negative transition of clock $f_2/2$ transfers contents of Rd1 to Rt2 in stage 2.

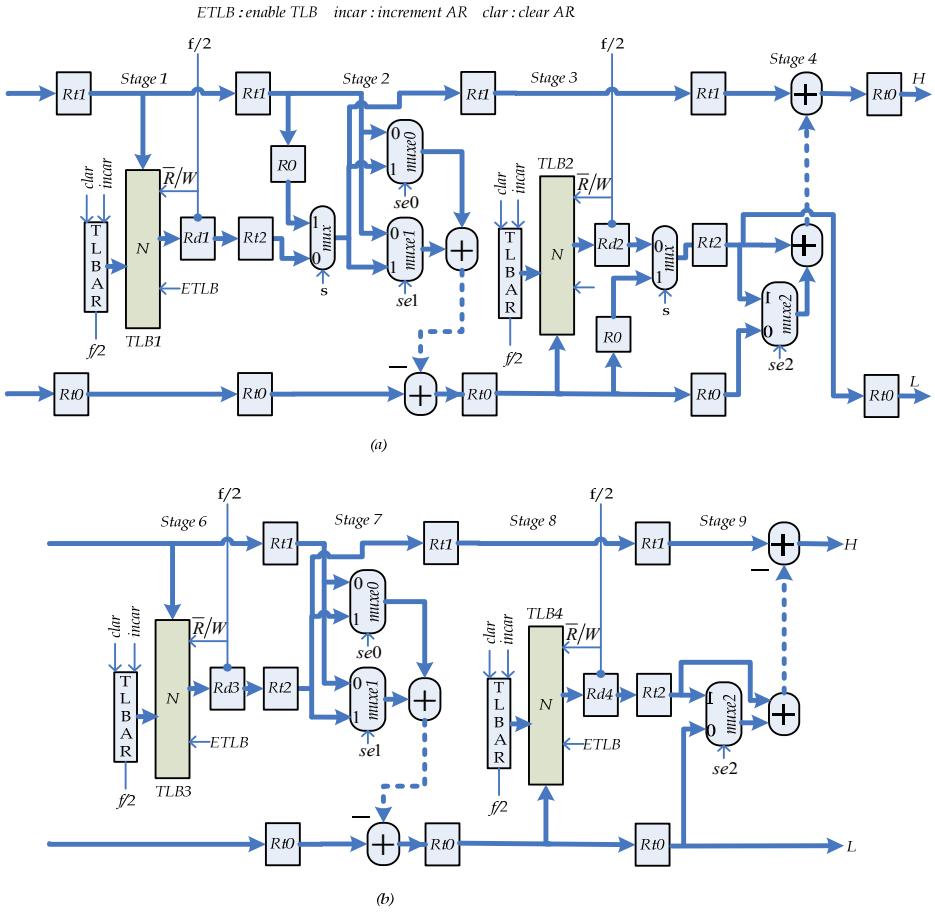


Fig. 9. Modified RP for 2-parallel architecture (a) 5/3 and (a) & (b) 9/7

	CK f_2	RP1 input latches STAGE 1 Rt0 Rt1 TLB1	STAGE 2 Rt0 Rt2 Rt1 R0	STAGE 3 Rt0 Rt1 R0 TLB2	STAGE 4 Rt0 Rt1 Rt2	RP1output latches Rt0 Rt1
RUN 1	11	L0,0 H0,0				----- -----
	13	L0,1 H0,1	L0,0 ---- H0,0			----- -----
	15	L2,0 H2,0 H0,1	L0,1 ---- H0,1 H0,0	X0,0 --- ----		----- -----
	17	L2,1 H2,1	L2,0 ---- H2,0 -----	X0,2 H0,0 X0,0	X0,0 ---- ----	----- -----
	19	L4,0 H4,0 H2,1	L2,1 ---- H2,1 H2,0	X2,0 ----- X0,2 X0,2	X0,2 H0,0 X0,0	----- -----

	21	L4,1 H4,1	L4,0 ---- H4,0 -----	X2,2 H2,0 X2,0	X2,0 ----- X0,2	X0,0 X0,1
	23	L6,0 H6,0 H4,1	L4,1 ---- H4,1 H4,0	X4,0 ----- X2,2 X2,2	X2,2 H2,0 X2,0	X0,2 -----
	25	L6,1 H6,1	L6,0 ---- H6,0 -----	X4,2 H4,0 X4,0	X4,0 ----- X2,2	X2,0 X2,1
RUN 2	27	----- ----- H6,1	L6,1 ---- H6,1 H6,0	X6,0 ----- X4,2 X4,2	X4,2 H4,0 X4,0	X2,2 -----
	29	L0,2 H0,2 ---- -	----- ----- -----	X6,2 H6,0 X6,0	X6,0 ----- X4,2	X4,0 X4,1
	31	L2,2 H2,2 H0,2	L0,2 H0,1 H0,2 -----	---- ----- X6,2 X6,2	X6,2 H6,0 X6,0	X4,2 -----
	33	L4,2 H4,2 H2,2	L2,2 H2,1 H2,2 -----	X0,4 H0,1 ----- X6,2	---- ---- X6,2	X6,0 X6,1
	35	L6,2 H6,2 H4,2	L4,2 H4,1 H4,2 -----	X2,4 H2,1 X0,4 X0,4	X0,4 H0,1 X0,2	X6,2 -----
	37	---- ----- H6,2	L6,2 H6,1 H6,2 -----	X4,4 H4,1 ----- X2,4	X2,4 H2,1 X2,2	X0,3 X0,4
	39	---- -----	----- ----- ------ - -----	X6,4 H6,1 ----- X4,4	X4,4 H4,1 X4,2	X2,3 X2,4
	41	---- -----	----- ----- ------ - -----	------ ----- ----- X6,4	X6,4 H6,1 X6,2	X4,3 X4,4
	43	---- -----	----- ----- ------ - -----	----- ----- ------ - -----	----- ----- --- ---	X6,3 X6,4

Table 2. Dataflow of the 5/3 RP1 (Fig. 9a)

In Fig. 9 (a), the control signal, s, of the two multiplexers' labeled mux is set 1 during run 1 to pass R0 of both stages 2 and 3, whereas in all other runs, it is set 0 to pass coefficients stored in TLB1 and TLB2.

4.3 Proposed 4-parallel external architecture

To further increase speed of computations twice as that of the 2-parallel architecture, the 2-parallel architecture is extended to 4-parallel architecture as shown in Fig. 10 (a). This architecture is valid for 5/3, 9/7, and combined 5/3 and 9/7. It consists of 4 *k*-stage pipelined CPs and 4 *k*-stage pipelined RPs. The waveforms of the 3 clocks f_4, f_{4a}, and f_{4b} used in the architecture are shown in Fig. 10 (b). The frequency of clock f_4 is determined from Eq(3) as

$$f_4 = 4k/t_p \tag{5}$$

The architecture scans the external memory with frequency f_4 and it operates with frequency f_{4a} and f_{4b}. Every time clock f_{4a} makes a negative transition CP1 loads into its input latches Rt0 and Rt1 two new coefficients scanned from external memory through the buses labeled *bus0* and *bus1*, whereas CP3 loads every time clock f_{4a} makes a positive transition. CP2 and CP4 loads every time clock f_{4b} makes a negative and a positive transition, respectively, as indicated in Fig. 10 (b). On the other hand, both RP1 and RP2 load simultaneously new data into their input latches Rt0 and Rt1 each time clock f_{4a} makes a negative transition, whereas RP3 and RP4 loads each time clock f_{4b} makes a negative transition.

The dataflow for 4-parallel 5/3 external architecture is given in Table 3, where CPs and RPs are assumed to be 3- and 4-stage pipelined processors, respectively. The dataflow table for

4-parallel 9/7 external architecture is similar in all runs to the 5/3 dataflow except in the first run, where RPs of the 9/7 architecture, specifically RP3 and RP4 generate a pattern of output coefficients different from that of the 5/3. RP3 and RP4 of the 9/7 architecture would generate every clock cycle, reference to clock f_{4b}, two output coefficients as follows. Suppose, at cycle number n the first two coefficients X(0,0) and X(1,0) generated by RP3 and RP4, respectively, are loaded into output latch Rt0 of both processors. Then, in cycle n+1, RP3 and RP4 generate coefficients X(2,0) and X(3,0) followed by coefficients X(4,0) and X(5,0) in cycle n+1 and so on. Note that these output coefficients are the coefficients generated by RP1 and RP2 in Table 3.

The strategy used for scheduling memory columns for CPs of the 5/3 and 9/7 4-parallel architecture, which resemble the one adopted for 2-parallel architecture, is as follow. In the first run, both 5/3 and 9/7 4-parallel architecture will be scheduled to execute 4 columns of memory, two from (A) and the other two from (B), both of Fig. 5. Each CP will be assigned to execute one column of memory coefficients as illustrated in the first run of the dataflow shown in Table 3, whereas in all subsequent runs, 2 columns at a time will scheduled for execution by 4 CPs. One column from Fig. 5 (A) will be assigned to both CP1 and CP3, while the other from Fig. 5 (B) will be assigned to both CP2 and CP4 as shown in the second run of Table 3. However, if number of columns in (A) and (B) of Fig. 5 is not equal, then the last run will consist of only one column of (A). In that case, schedule the last column's coefficients in both CP1 and CP3 as shown in the third run of Table 3, since an attempt to execute the last column using 4 CPs would result in more coefficients been generated than that can be handled by the 4 RPs.

On the other hand, scheduling rows coefficients for RPs, which take place according to scan method shown in Fig. 6, can be understood by examining the dataflow shown in Table 3. At cycle 13, CP1 generates its first two output coefficients labeled L0,0 and L1,0, which correspond to locations 0,0 and 1,0 in Fig. 6, respectively. In cycle 14, CP2 generates its first two output coefficients H0,0 and H1,0, which correspond to locations 0,1 and 1,1 in Fig. 6, respectively. In cycle 15, CP3 generate its first two coefficients L0,1 and L1,1, which correspond to locations 0,2 and 1,2 in Fig. 6, respectively. In cycle 16, CP4 generates its first two output coefficients H0,1 and H1,0 which correspond to locations 0,3 and 1,3 in Fig. 6. Note that L0,0, H0,0, L0,1, and H0,1 represents the first 4 coefficients of row 0 in Fig. 6, whereas L1,0, H1,0, LL1,1 and H1,1, represent the first 4 coefficients of row1.

In cycle 17 and 18, the first two rows coefficients are scheduled for RPs as shown in Table 3, while CPs generating coefficients of the next two rows, row2 and row3. Table 3 shows that the first 4 coefficients of row 0 are scheduled for execution by RP1 and RP3, while the first 4 coefficients of row 1 are scheduled for RP2 and RP4. In addition, note that all coefficients generated by CP4, which belong to column 3 in Fig. 6, are required in the second run's computations, according to the DDGs. Therefore, this would require inclusion of a TLB of size $N/4$ in each of the 4 RPs to store these coefficients. The second run, however, requires these coefficients to be stored in the 4 TLBs in a certain way as follows. Coefficients H0,1 and H1,1 generated by CP4 in cycle 16 should be stored in the first location of TLB of RP1 and RP2, respectively. These two coefficients would be passed to their respective TLB through the input latches of RP1 and RP2 labeled Rt2, as shown in cycle 17 of Table 3, whereas, coefficients H2,1 and H3,1 generated by CP4 at cycle 20 should be stored in the first location of TLB of RP3 and RP4, respectively. These two coefficients are passed to their respective TLB for storage through the input latches of RP3 and RP4 labeled Rt1, as shown in cycle 22

of Table 3. Similarly, coefficients H4,1 and H5,1 generated by CP4 at cycle 24 should be stored in the second location of TLB of RP1 and RP2, respectively, and so on. Note that these TLBs are labeled TLB1 in Fig. 12.

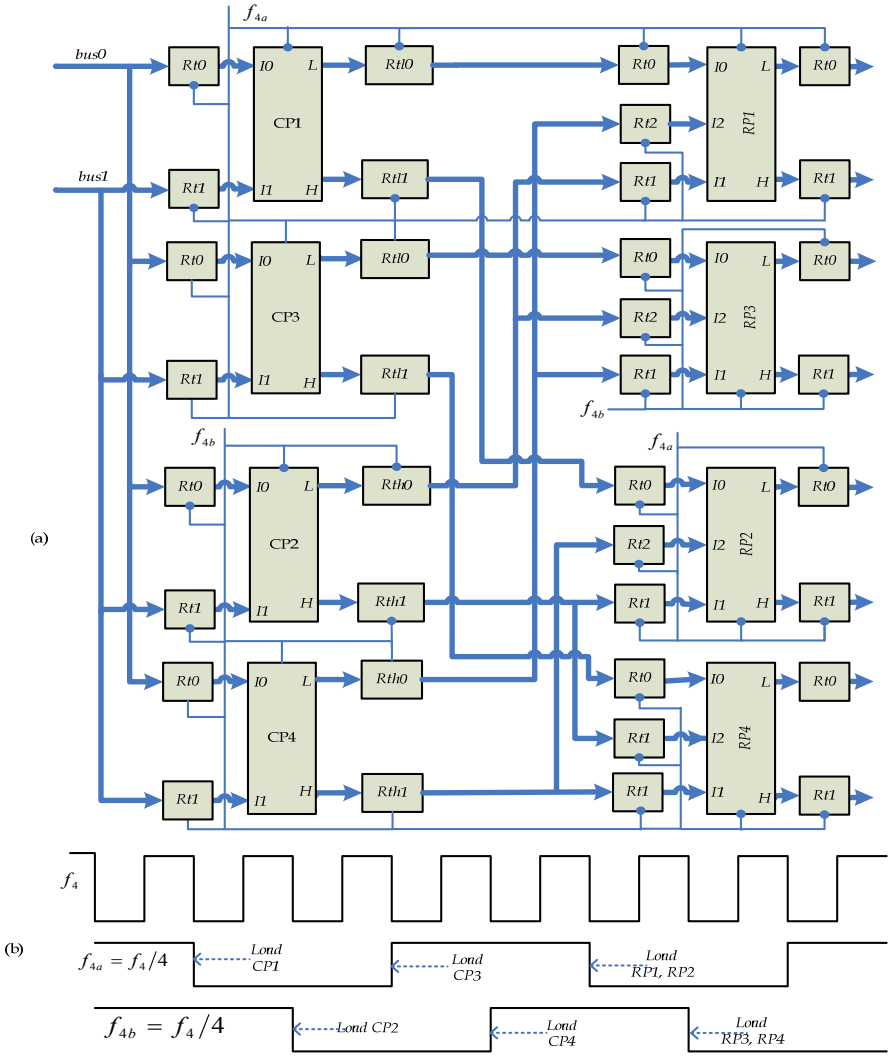


Fig. 10. (a) Proposed 2-D IDWT 4-parallel pipelined external architecture for 5/3 and 9/7 and combined (b) Waveforms of the clocks

	CK f_4	CP	CPs input Latches Rt0 Rt1	CPs 1 &3 Out latches Rtl0 Rtl1	CPs 2 & 4 Out latches Rth0 Rth1	RPs 1 & 3 input latches RP Rt0 Rt1 Rt2	RPs 2 & 4 input latches RP Rt0 Rt1 Rt2	RPs 1 & 3 Out latches Rt0 Rt1	RPs 2 & 4 Out latches Rt0 Rt1
RUN 1	1	1	LL0,0 LH0,0						
	2	2	HL0,0 HH0,0						
	3	3	LL0,1 LH0,1						
	4	4	HL0,1 HH0,1						
	5	1	LL1,0 LH1,0						
	6	2	HL1,0 HH1,0						
	7	3	LL1,1 LH1,1						
	8	4	HL1,1 HH1,1						
	9	1	LL2,0 LH2,0						
	10	2	HL2,0 HH2,0						
	11	3	LL2,1 LH2,1						
	12	4	HL2,1 HH2,1						
	13	1	LL3,0 LH3,0	L0,0 L1,0					
	14	2	HL3,0 HH3,0		H0,0 H1,0				
	15	3	LL3,1 LH3,1	L0,1 L1,1					
	16	4	HL3,1 HH3,1		H0,1 H1,1				
	17	1	LL4,0 --- ---	L2,0 L3,0		1 L0,0 H0,0 H0,1	2 L1,0 H1,0 H1,1		
	18	2	HL4,0 -- ----		H2,0 H3,0	3 L0,1 H0,1 H0,0	4 L1,1 H1,1 H1,0		
	19	3	LL4,1 --- ---	L2,1 L3,1					
	20	4	HL4,1 -- ----		H2,1 H3,1				
RUN 2	21	1	LL0,2 LH0,2	L4,0 L5,0		1 L2,0 H2,0 -----	2 L3,0 H3,0 -----		
	22	2	HL0,2 HH0,2		H4,0 H5,0	3 L2,1 H2,1 H2,0	4 L3,1 H3,1 H3,0		
	23	3	LL1,2	L4,1					

			LH1,2	L5,1					
	24	4	HL1,2 HH1,2		H4,1 H5,1				
	25	1	LL2,2 LH2,2	L6,0 L7,0		1 L4,0 H4,0 H4,1	2 L5,0 H5,0 H5,1		
	26	2	HL2,2 HH2,2		H6,0 H7,0	3 L4,1 H4,1 H4,0	4 L5,1 H5,1 H5,0		
	27	3	LL3,2 LH3,2	L6,1 L7,1					
	28	4	HL3,2 HH3,2		H6,1 H7,1				
	29	1	LL4,2 ------	L8,0 ------		1 L6,0 H6,0 -----	2 L7,0 H7,0 -----		
	30	2	HL4,2 ------		H8,0 ------	3 L6,1 H6,1 H6,0	4 L7,1 H7,1 H7,0		
	31	3	------- ------	L8,1 ------					
	32	4	------- ------		H8,1 ------				
RUN 3	33	1	LL0,3 LH0,3	L0,2 L1,2		1 L8,0 H8,0 H8,1	2 ----- ------	X0,0 ------	X1,0 ------
	34	2	----- ------		H0,2 H1,2	3 L8,1 H8,1 H8,0	4 ----- ------	X0,1 X0,2	X1,1 X1,2
	35	3	LL1,3 LH1,3	L2,2 L3,2					
	36	4	------ ------		H2,2 H3,2				
	37	1	LL2,3 LH2,3	L4,2 L5,2		1 L0,2 H0,2 -----	2 L1,2 H1,2 -----	X2,0 ------	X3,0 ------
	38	2	------- ------		H4,2 H5,2	3 L2,2 H2,2 -----	4 L3,2 H3,2 -----	X2,1 X2,2	X3,1 X3,2
	39	3	LL3,3 LH3,3	L6,2 L7,2					
	40	4	------- ------		H6,2 H7,2				
	41	1	LL4,3 ------	L8,2 ------		1 L4,2 H4,2 -----	2 L5,2 H5,2 -----	X4,0 ------	X5,0 ------
	42	2	------- ------		H8,2 ------	3 L6,2 H6,2 -----	4 L7,2 H7,2 -----	X4,1 X4,2	X5,1 X5,2
	43	3	------- ------	----- ------					
	44	4	------- ------		------ ------				
	45	1		L0,3 L1,3		1 L8,2 H8,2 -----	2 ----- ------	X6,0 ------	X7,0 ------
	46	2			------ ------	3 ----- ------	4 ----- ------	X6,1 X6,2	X7,1 X7,2
	47	3		L2,3 L3,3					
	48	4			------ ------				

	49	1		L4,3 L5,3		1 L0,3 ---- - -----	2 L1,3 --- -- -----	X8,0 --- -	----- --- --
	50	2			------ -- ----	3 L2,3 --- -- -----	4 L3,3 --- -- -----	X8,1 X8,2	----- --- --
	51	3		L6,3 L7,3					
	52	4			------ -- ----				
	53	1		L8,3 ---- -		1 L4,3 --- -- -----	2 L5,3 --- -- -----	X0,3 X0,4	X1,3 X1,4
	54	2			------ -- ----	3 L6,3 --- -- -----	4 L7,3 --- -- -----	X2,3 X2,4	X3,3 X3,4
	55	3		------ --- ---					
	56	4			----- --- ---				

Table 3. Dataflow for 4-parallel 5/3 architecture

At cycle 33, RP1 and RP2 yield their first output coefficients X0,0 and X1,0, respectively, which must be stored in the external memory locations 0,0 and 1,0, respectively. Note that indexes of each output coefficient indicate external memory location where the coefficient should be stored.

The second run, which requires scheduling two columns for execution by CPs, starts at cycle 21. In cycle 33, it generates its first two output coefficients L0,2 and L1,2, which belong to locations 0,4 and 1,4 in Fig. 6, respectively. In cycle 34, CP2 generates coefficients H0,2 and H1,2 which belong to locations 0,5 and 1,5 in Fig. 6. In cycle 35, CP3 generates coefficients L2,2 and L3,2, which belong to locations 2,4 and 3,4 in Fig. 6, whereas in cycle 36, CP4 generates coefficients H2,2 and H3,2 that belong to locations 2,5 and 3,5 in Fig. 6. From the above description it is clear that these 8 coefficients are distributed along 4 rows, 0 to 3 in Fig. 6 with each row having 2 coefficients. Table 3 shows that in cycle 37, the two coefficients of row 0, L0,2 and H0,2, and the two coefficients of row 1, L1,2 and H1,2 are scheduled for RP1 and RP2, respectively, while coefficients L4,2 and L5,2 generated by CP1 during the cycle are loaded into Rtl0 and Rtl1, respectively. In cycle 38, the two coefficients of row 2, L2,2 and H2,2, and that of row 3 L3,2 and H3,2 are scheduled for RP3 and RP4, respectively, while coefficients H4,2 and H5,2 generated by CP2 during the cycle are loaded into Rth0 and Rth1, respectively. In cycle 53, RP1 and RP2 generate the first output coefficients of the second run.

4.4 Column and row processors for 5/3 and 9/7 4-parallel external architecture

The 5/3 and the 9/7 processors datapath architectures proposed in (Ibrahim & Herman, 2008) were developed assuming the processors scan external memory either row by row or column by column. However, CPs and RPs of the 4-parallel architecture are required to scan external memory according to scan methods shown in Figs. 5 and 6, respectively. The 4-parallel architecture, in addition, introduces the requirement for interactions among the processors in order to accomplish their task. Therefore, the processors datapath architectures proposed in (Ibrahim & Herman, 2008) should be modified based on the scan methods and taking into consideration also the requirement for interactions so that they fit

into the 4-parallel's processors. Thus, in the following, the modified CPs will be developed first followed by RPs.

4.5 Modified CPs for 4-parallel architecture

The 4 CPs of the 4-parallel architecture each is required in the first run to execute one column at a time. That means the first run requires no modifications of the 5/3 and 9/7 datapath architectures proposed in (Ibrahim & Herman, 2008). However, in all subsequent runs, each two processors (CP1 and CP3 or CP2 and CP4) are assigned to execute one column together, which requires interactions between the two processors to accomplish the required task. Therefore, both CPs 1 and 3, similarly, CPs 2 and 4 should be modified as shown in Fig. 11 to allow interactions to take place. The two processors communicate or interact through the paths (buses) labeled *P1, P2, P3,* and *P4*. Fig. 11 shows modified 5/3 CPs 1 and 3 which is identical to CPs 2 and 4. Fig. 11 also represents the first 3 stages of both 9/7 CPs 1 and 3 and CPs 2 and 4 and the remaining stages are identical to stages 1 to 3. Note that since the first 3 stages of 5/3 and 9/3 are similar in structure, the 5/3 processor can be easily incorporated into 9/7 processor to obtain the combined 5/3 and 9/7 processor for 4-parallel architecture.

The control signal, *s* of the 4 multiplexers, labeled *mux* is set 0 in the first run to allow each processor to execute one column and 1 in all other runs to allow execution of one column by two processors.

In the following, the second run's dataflow of Fig. 11, which starts at cycle 21 in Table 3 will be described. In cycle 21, two coefficients LL0,2 and LH0,2, which correspond to the first two coefficients of the third column in Fig. 5 (A), are read from external memory and are loaded into CP1 first stage latches Rt0 and Rt1, respectively, by negative transition of clock f_{4a} to initiate the first operation. In cycle 23, the third and the fourth coefficients LL1,2 and LH1,2 of the third column are scanned from external memory and are loaded along with Rt1 in stage1 of CP1 into CP3 first stage latches Rt0, Rt1 and Rd3, respectively, by the positive transition of clock f_{4a}. Note that content of Rt1 in stage 1 of CP1 takes the path labeled *P*1 to Rd3.

In cycle 25, coefficients LL2,2 and LH2,2 are scanned from external memory and are loaded along with Rt1 in stage 1 of CP3 into CP1 first stage latches Rt0, Rt1, and Rd1, respectively, by the negative transition of clock f_{4a}, while coefficient X(0) calculated in stage 1 of CP1 and content of Rt1 are loaded into Rt0 and Rt1 of stage 2, respectively.

In cycle 27, coefficients LL3,2 and LH3,2 are scanned from external memory and are loaded, along with Rt1 in stage 1 of CP1, into CP3 first stage latches Rt0, Rt1, and Rd3, respectively, by the positive transition of clock f_{4a}, while coefficients X(2) calculated in stage 1 and content of Rt1 are transferred to Rt0 and Rt1 of the next stage, respectively.

In cycle 29, only one coefficient LL4,2 is scanned from external memory, which implies the third column of Fig. 5 (A) contains odd number of coefficients, and is loaded along with Rt1 in stage1 of CP3 into CP1 first stage latches Rt0 and Rd1, respectively, by the negative transition of clock f_{4a}, while coefficient X(4) calculated in stage 1 and content of Rt1 are transferred to Rt0 and Rt1 of stage 2 and that of Rt0 and Rt1 including Rt0 in stage 2 of CP3 to Rt0, Rt1, and Rt2 in stage 3 of CP1, respectively, to compute the first high coefficient labeled X(1) in 5/3 DDGs. Note that CP1 latches Rt0 and Rt2 of stage 3 now contain coefficients X(0) and X(2), while Rt1 contains coefficient LH0,2 (or Y(1) as labeled in the 5/3

DDGs). These 3 coefficients are required according to the DDGs to compute the high coefficient, X(1).

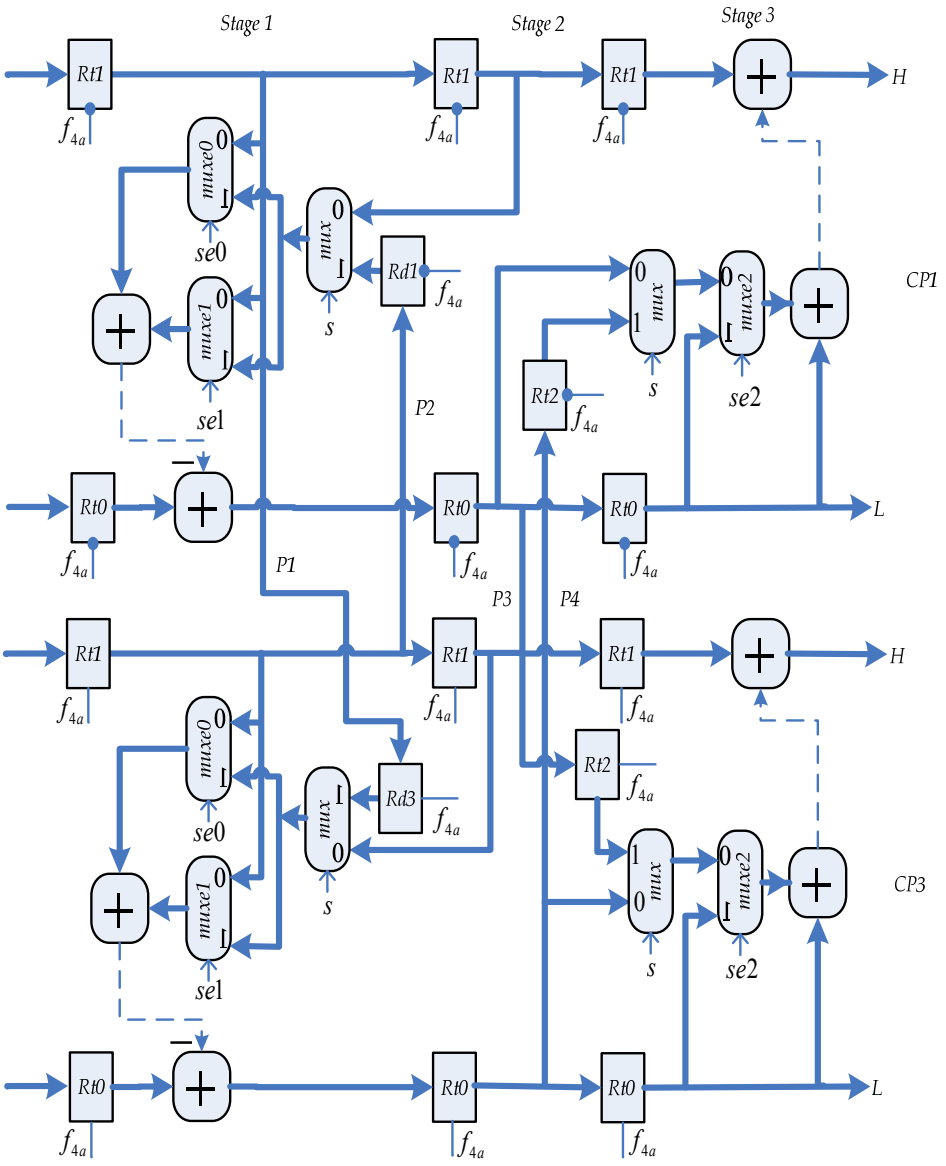


Fig. 11. Modified 5/3 CPs 1 & 3 for 4-parallel architecture

In cycle 31, the positive transition of clock f_{4a} transfers Rt0 and Rt1 in stage 2 of CP3 and Rt0 in stage 2 of CP1 to stage 3 latches Rt0, Rt1, and Rt2 of CP3, respectively, to compute the second high coefficient, X(3). While coefficient X(6) calculated in stage 1 of CP3 and coefficient in Rt1 are loaded into Rt0 and Rt1 of stage 2. As indicated in cycle 31 of Table 3, no coefficients are loaded into CP3 first stage latches.
In cycle 33, the negative transition of clock f_{4a} loads the first high coefficient, X(1) calculated in stage 3 of CP1 and Rt0, which contains X(0), into CP1 output latches Rt0 and Rt1, respectively. Coefficients X(0) and X(1) are labeled L0,2 and L1,2 in Table 3. The same negative transition of clock f_{4a} transfers contents of Rt0 and Rt1 in stage 2 of CP1 and Rt0 in stage 2 of CP3 to Rt0, Rt1, and Rt2 in stage 3 of CP1, respectively, to compute the third coefficient, X(3) and so on.

4.6 Modified RPs for 4-parallell architecture

In section 5.2, it has been pointed out the reasons for including TLBs in the two RPs of the 2-parallel architecture. For the same reasons, it is also necessary to include TLBs in the 4 RPs of the 4-parallel architecture, as shown in Figs. 12 (a) and (a,b) for 5/3 and 9/7, respectively. The processor datapath for both RP1 and RP3, which is also identical to the processor datapath of both RP2 and RP4, are drawn together in Figs.12 (a) and (a,b) for 5/3 and 9/7, respectively, because in the first run, both processors are required to execute together the 4 coefficients of each row. Which implies interactions between the two processors during the computations and that take place through the paths (buses) labeled *P1, P2, P3,* and *P4*. However, in all subsequent runs, according to the scan method shown in Fig. 6, each RP will be scheduled to execute each clock cycle two coefficients of a row as shown in cycles 37 and 38 of Table 3. The advantage of this organization is that the total size of the TLBs does not increase from that of the single pipelined architecture, when it is extended to 2- and 4-parallel architecture.
In the first run, all TLBs in Fig. 12 will be written only, whereas in all other runs, the same location of a TLB will be read in the first half cycle and written in the second half cycle with respect to clock f_{4a} or f_{4b}.
The control signal, *s* of the six multiplexers, labeled *mux* in Fig. 12, is set 0 in the first run to allow in the RP1, coefficient coming through path 0 of each multiplexer to be stored in its respective TLB, whereas in the RP3, it allows contents of Rt2 and Rd1 in stages 1 and 3, respectively, to be passed to the next stage. In all subsequent runs, *s* is set 1 to allow in the RP1, coefficient coming through path 1 of each multiplexer to be stored in the TLB, whereas in the RP3, it passes coefficients read from TLB1 and TLB2 to next stage.

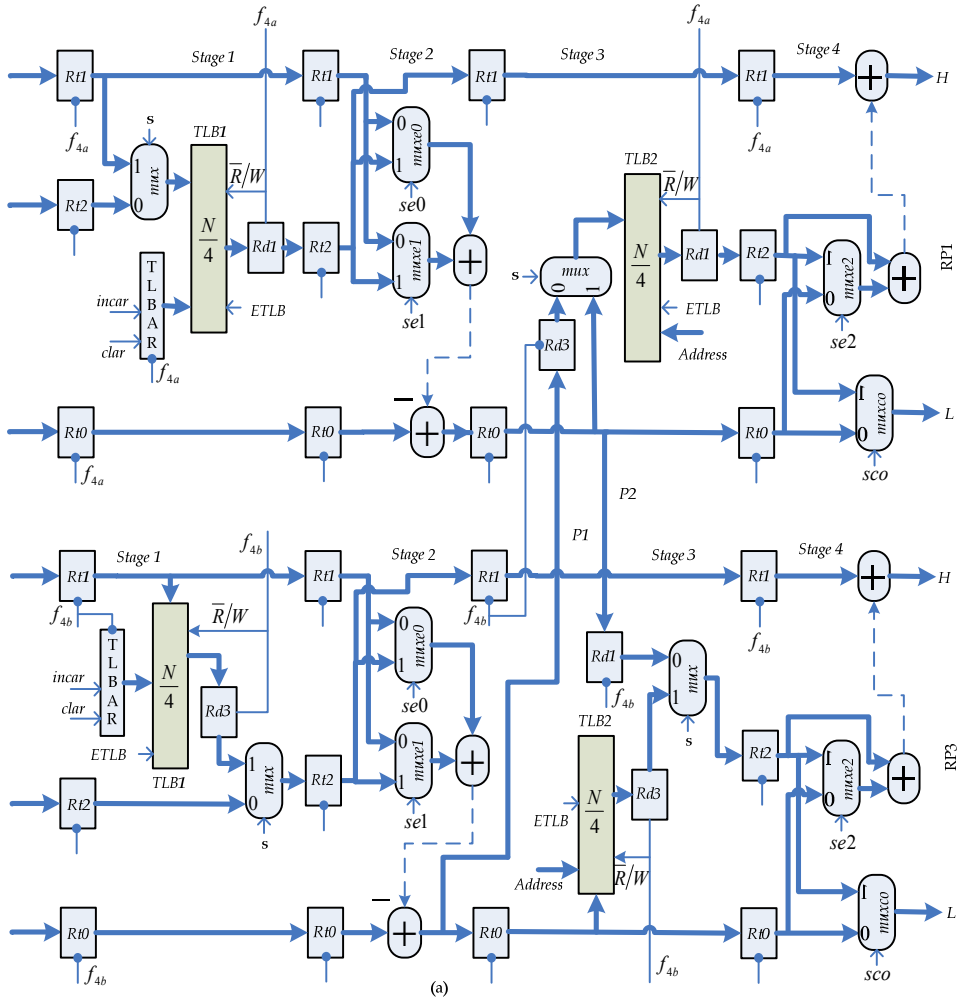


Fig. 12. (a) Modified 5/3 RPs 1 and 3 for 4-parallel External Architecture

In the following, the dataflow of the processor datapath architecture shown in Fig. 12 (a) will described in details starting from cycle 17 in Table 3. The detailed descriptions will enable us to fully understand the behavior of the processor. In cycle 17, the negative transition of clock f_{4a} (consider it the first cycle of f_{4a}) loads coefficients L0,0 and H0,0, which represent the first two coefficients of row 0 in Fig. 6, and H0,1 into RP1 first stage latches Rt0, Rt1, and Rt2, respectively. During the positive (high) pulse of clock f_{4a}, coefficient in Rt2 is stored in the first location of TLB1.

In cycle 18, Table 3 shows that the negative transition of clock f_{4b} (consider it the first cycle of f_{4b}) loads coefficients L0,1, H0,1, and H0,0 of row 0 into RP3 first stage latches Rt0, Rt1, and Rt2,

respectively. In cycle 21, the negative transition of the second cycle of clock f_{4a} transfers contents of RP1 latches Rt0 and Rt1 of the first stage to stage 2 latches Rt0 and Rt1, respectively, to compute the first low coefficient, X0,0, while loading two new coefficients L2,0 and H2,0, which are the first two coefficients of row 2 in Fig. 6, into RP1 first stage latches Rt0 and Rt1, respectively.

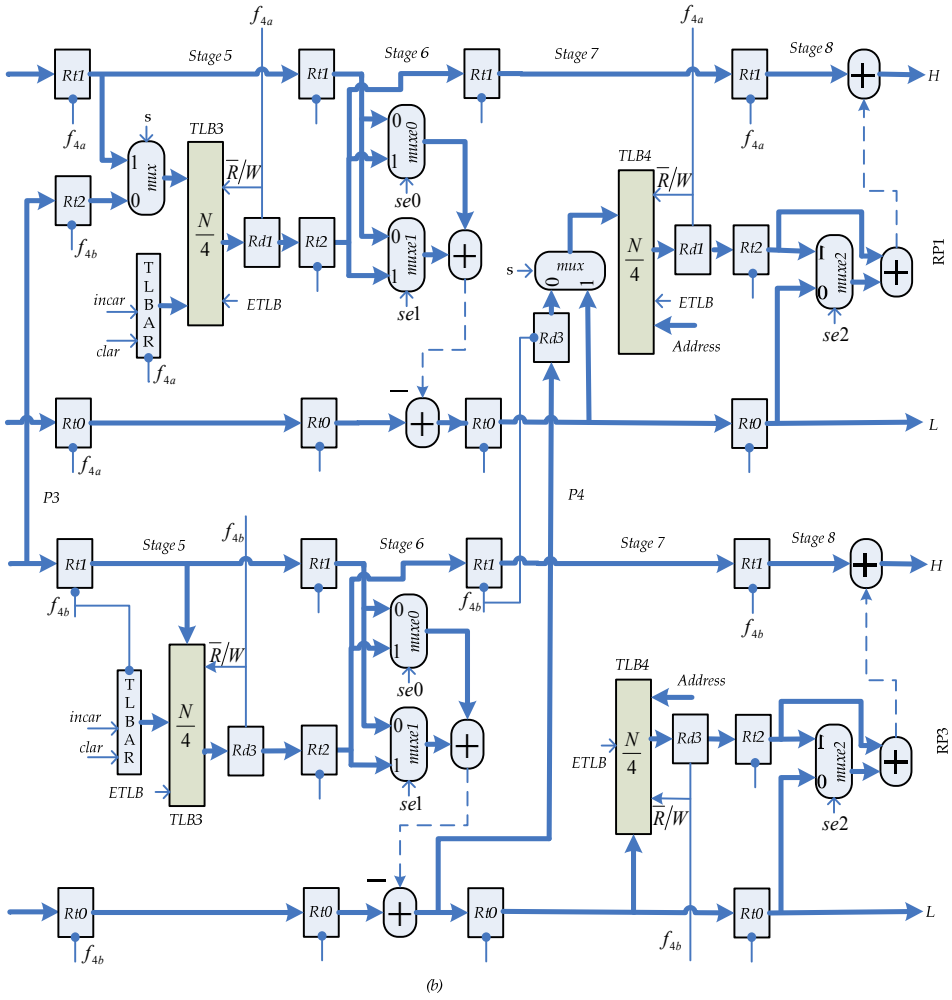


Fig. 12. (b) Modified 9/7 RPs 1 and 3 for 4-parallel External Architecture

During the second cycle of clock f_{4a} no data are stored in TLB1 of RP1.
In cycle 22, the negative transition of the second cycle of clock f_{4b} transfers contents of RP3 latches Rt0, Rt1 and Rt2 of the first stage to stage 2 latches Rt0, Rt1 and Rt2 to compute the second low coefficient of row 0, X0,2, while loading two new coefficients L2,1, H2,1, and

H2,0 into RP3 first stage latches Rt0, Rt1 and Rt2, respectively. During the second cycle of clock f_{4b}, the positive pulse stores content of Rt1 of the first stage in the first location of TLB1 of RP3.

In cycle 25, the negative transition of the third cycle of clock f_{4a} loads coefficient X0,0 computed in stage 2 of RP1, into Rt0 of stage 3 and transfers contents of Rt1 and Rt0 of stage 1 into Rt1 and Rt0 of stage 2 in order to compute the first low coefficient, X2,0 of row 2 labeled X(0) in the 5/3 DDGs, while loading new coefficients L4,0, H4,0, and H4,1 of row 4 in Fig. 6, into RP1 first stage latches Rt0, Rt1 and Rt2, respectively. During the third cycle of clock f_{4a}, the positive pulse stores Rt2 of the first stage in the second location of TLB1 of RP1.

In cycle 26, the negative transition of the third cycle of clock f_{4b} loads coefficient X0,2 calculated in stage 2 of RP3 into both Rt0 in stage 3 of RP3 and Rd3 in stage 3 of RP1, while content of Rt2 in stage 2 of RP3 is transferred to Rt1 of stage 3 and that of Rt0 in stage 3 of RP1 to Rd1 in stage 3 of RP3. The same negative transition of clock f_{4b} also transfers Rt0, Rt1, and Rt2 of stage 1 into Rt0, Rt1, and Rt2 of stage 2, respectively, to compute the second low coefficient, X2,2 of row 2, while loading new coefficients L4,1, H4,1, and H4,0 of row 4 into RP3 first stage latches Rt0, Rt1, and Rt2, respectively. During the third cycle of clock f_{4b}, coefficient in Rt1 is not stored in TLB1 of RP3. It is important to note that Rd3 in stage 3 of RP1, which holds coefficient X0,2, will be stored in the first location of TLB2 by the positive pulse of the third cycle of clock f_{4a}.

In cycle 29, the fourth cycle's negative transition of clock f_{4a} transfers Rt0 in stage 3 of RP1, which contains X0,0, to Rt0 of stage 4, while X2,0 calculated in stage 2 is loaded into Rt0 of stage 3. The same negative transition of clock f_{4a} transfer Rt1 and Rt0 of stage 1 to Rt1 and Rt0 of stage 2, respectively, to compute the first low coefficient of row 4, X4,0, and loads new coefficients L6,0 and H6,0 into Rt0 and Rt1 of stage 1, respectively. During the fourth cycle of clock f_{4a}, coefficient in Rt2 of stage 1 is not stored in TLB1 of RP1.

In cycle 30,the negative transition of the fourth cycle of clock f_{4b} transfers contents of Rt0, Rt1, and Rd1 in stage 3 of RP3 to Rt0, Rt1, and Rt2 of the next stage, respectively, to compute the first high coefficient, X0,1 of row 0. While coefficient X2,2 calculated in stage 2 of RP3 is transferred to both Rt0 in stage3 of RP3 and Rd3 in stage 3 of RP1 and coefficient X2,0 in Rt0, in stage 3 of RP1, is loaded into Rd1 in stage 3 of RP3, whereas Rt2 of stage 2 is transferred to Rt1 in stage 3 of RP3. The same negative transition of clock f_{4b} transfers contents of Rt0, Rt1, and Rt2 of stage 1 to Rt0, Rt1, and Rt2 of stage 2 to compute the second low coefficient, X4,0 of row 4, while loading new coefficients L6,1, H6,1, and H6,0 of row 6 into RP3 first stage latches Rt0, Rt1, and Rt2, respectively. During the fourth cycle's positive pulse of clock f_{4b}, content of Rt1 in stage 1 of RP3 will be stored in the second location of TLB1 and content of Rt0, X2,2, in stage 3 of RP3 will be stored in the first location of TLB2, while content of Rd3, X2,2, in stage 3 of RP1 will not be stored in TLB2 of RP1.

In cycle 33, the negative transition of the fifth cycle of clock f_{4a} transfers content of Rt0, X0,0, in stage 4 of RP1 to RP1 output latch Rt0, as first output coefficient and Rt0 of stage 3 holding coefficient X2,0 to Rt0 of stage 4, while coefficient X4,0 calculated in stage 2 is loaded into Rt0 of stage 3. The same negative transition of clock f_{4a} transfers contents of Rt0 and Rt1 of stage 1 to Rt0 and Rt1 of stage 2 to compute the first low coefficient, X6,0 of row 6, while loading new coefficients L8,0, H8,0, and H8,1 of row 8 into RP1 first stage latches Rt0, Rt1, and Rt2, respectively. During the fifth cycle of clock f_{4a}, the positive pulse of the clock stores content of Rt2 in stage 1 of RP1 into the third location of TLB1.

In cycle 34, the negative transition of the fifth cycle of clock f_{4b} transfers content of Rt0, X0,2, and the high coefficient, X0,1 computed in stage 4 of RP3 to RP3 output latches Rt0 and Rt1, respectively, while loading contents of Rt0, Rt1, and Rd1 of stage 3 into Rt0, Rt1, and Rt2 of stage 4 to compute the first high coefficient of row 2, X2,1. Furthermore, the same negative transition of clock f_{4b} also transfers coefficient X4,2 calculated in stage 2 of RP3 to both Rt0 in stage 3 of RP3 and Rd3 in stage 3 of RP1. It also transfers coefficient X4,0 in Rt0, in stage 3 of RP1, and content of Rt2 in stage 2 of RP3 to stage 3 of RP3 latches Rd1 and Rt1, respectively, and contents of Rt0, Rt1, and Rt2 of stage 1 to Rt0, Rt1, and Rt2 in stage 3 of RP3, while loading new coefficients L8,1, H8,1, and H8,0 into Rt0, Rt1, and Rt2 of stage 1. Content of Rd3, X4,2 in stage 3 of RP1 will be stored in the second location of TLB2 by the positive pulse of the fifth cycle of clock f_{4a}.

In cycle 37, the second run of the RPs begins when the negative transition of the sixth cycle of clock f_{4a} loads two new coefficients L0,2 and H0,2, which are the fifth and sixth coefficients of row 0, into first stage latches Rt0 and Rt1 of RP1, respectively. During the first half (low pulse) of cycle 6, the first location of TLB1 will be read and loaded into Rd1 by the positive transition of clock f_{4a}, whereas during the second half cycle, content of Rt1 will be written in the first location of TLB1.

In cycle 38, the negative transition of the sixth cycle of clock f_{4b} loads two new coefficients L2,2 and H2,2, the fifth and sixth coefficients of row 2, into RP3 first stage latches Rt0 and Rt1, respectively. The first half cycle of clock f_{4b} reads the first location of TLB1 and loads it into Rd3 by the positive transition of the clock, whereas the second half cycle writes content of Rt1 in the same location of TLB1.

In cycle 41, the negative transition of the seventh cycle of clock f_{4a} transfers contents of Rt0, Rt1, and Rd1 of stage 1 to stage 2 of RP1 latches Rt0, Rt1, and Rt2, respectively, to compute the third low coefficient, X0,4 of row 0, while loading two new coefficients L4,2 and H4,2 of row 4 into RP1 first stage latches Rt0 and Rt1 respectively.

Note that during run 2 all RPs execute independently with no interactions among them. In addition, in the first run, if the first coefficient generated by stage 2 of RP3 is stored in TLB2 of RP1, then the second coefficient should be stored in TLB2 of RP3 and so on. Similarly, TLB1, TLB3, and TLB4 of both RP1 and RP3 are handled. Furthermore, during the whole period of run 1, the control signals of the three extension multiplexers labeled *muxe0, muxe1,* and *muxe2* in RP1 should be set 0, according to Table 4 (Ibrahim & Herman, 2008), whereas those in RP3 should be set normal as shown in the second line of Table 4, since RP3 will execute normal computations during the period. However, in the second run and in all subsequent runs except the last run, the extension multiplexers control signals in all RPs are set normal. Moreover, the multiplexers labeled *muxco* in stage 4 is only needed in the combined 5/3 and 9/7 architecture, otherwise, it can be eliminated and Rt2 output can be connected directly to Rt0 input of the next stage in case of 9/7, whereas in 5/3, Rt0 is connected directly to output latch Rt0. In the combined architecture, signal *sco* of *muxco* is set 0 if the architecture is to perform 5/3; otherwise, it is set 1 if the architecture is to perform 9/7.

It is very important to note that when the RP executes its last set of input coefficients, according to 9/7 DDGs for odd and even signals shown in Fig. 4 it will not yield all required output coefficients as expected by the last run. For example, in the DDGs for odd length signals shown in Fig. 4 (a), when the last input coefficient labeled Y8 is applied to RP it will yield output coefficients X5 and X6. To get the last remaining two coefficients X7 and X8, the

RP must execute another run, which will be the last run in order to compute the remaining two output coefficients. Similarly, when the last two input coefficients labeled Y6 and Y7 in the DDG for even length signals shown in Fig. 4 (b) are applied to 9/7 RP it will yield output coefficients X3 and X4. To obtain the remaining output coefficients X5, X6, and X7, two more runs should be executed by RP according to the DDG. The first run will yield X5 and X6, whereas the last run will yield X7. The details of the computations that take place during each of these runs can be determined by examining the specific area of the DDGs.

	se0	*se1*	*se2*
First	0	0	0
Normal	0	1	0
Last	1	1	0

a) Odd length signals

	se0	*se1*	*se2*
First	0	0	0
Normal	0	1	0
Last	0	1	0

b) Even length signals

Table 4. Extension's control signals

5. Performance Evaluation

In order to evaluate performance of the two proposed parallel pipelined architectures in terms of speedup, throughput, and power as compared with single pipelined architecture proposed in (Ibrahim & Herman, 2008) consider the following. Assume subbands HH, HL, LH, and LL of each level are equal in size. The dataflow table for single pipelined architecture (Ibrahim & Herman, 2008) shows that $\rho_1 = 20$ clock cycles are needed to yield the first output coefficient. Then, the total number of output coefficients in the first run of the J[th] level reconstruction can be estimated as

$$N/2^{J-1} \tag{6}$$

and the total number of cycles in the first run is given by

$$2N/2^{J-1} \tag{7}$$

The total time, $T1$, required to yield n pairs of output coefficients for the J[th] level reconstruction on the single pipelined architecture can be estimated as

$$\begin{aligned} T1 &= \left(\rho_1 + 2N/2^{J-1} + 2\left(n - 1/2\, N/2^{J-1}\right)\right)\tau_1 \\ &= \left(\rho_1 + N/2^{J-1} + 2n\right)t_p/2k \end{aligned} \tag{8}$$

On the other hand, the dataflow for 2-parallel pipelined architecture shows that $\rho_2 = 21$ clock cycles are required to yield the first 2 pairs of output coefficients. The total number of paired output coefficients in the first run of the J[th] level reconstruction on the 2-parallel architecture can be estimated as

$$3/2\,N/2^{J-1} \tag{9}$$

and the total number of 2-paired output coefficients is given by

$$3/4\,N/2^{J-1} \tag{10}$$

While, the total number of cycles in the first run is

$$2\,N/2^{J-1} \tag{11}$$

Note that the total number of paired output coefficients of the first run in each level of reconstruction starting from the first level can be written as

$$3/2\,N,\ 3/2\,N/2,\ 3/2\,N/4,\ldots\ldots\ldots\ldots,3/2\,N/2^{J-1}$$

where the last term is Eq (9).

The total time, *T2*, required to yield *n* pairs of output coefficients for the J[th] level reconstruction of an *NxM* image on the 2-parallel architecture can be estimated as

$$T2 = \left(\rho_2 + 2N/2^{J-1} + 2(n/2 - 3/4\,N/2^{J-1})\right)t_2 \tag{12}$$

$$T2 = \left(\rho_2 + 1/2\,N/2^{J-1} + n\right)t_p/2k \tag{13}$$

The term $2\left(n/2 - 3/4\,N/2^{J-1}\right)$ in (12) represents the total number of cycles of run 2 and all subsequent runs.

The speedup factor, *S2,* is then given by

$$S2 = \frac{T1}{T2} = \frac{\left(\rho_1 + N/2^{J-1} + 2n\right)t_p/2k}{\left(\rho_2 + 1/2\,N/2^{J-1} + n\right)t_p/2k}$$

For large *n*, the above equation reduces to

$$S2 = \frac{2(1/2\,N/2^{J-1} + n)}{(1/2\,N/2^{J-1} + n)} = 2 \tag{14}$$

Eq(14) implies that the 2-parallel architecture is 2 times faster than the single pipelined architecture.

Similarly, the dataflow for the 4-parallel pipelined architecture shows that $\rho_4 = 33$ clock cycles are needed to yield the first two output coefficients. From the dataflow table of the 4-paralell architecture it can be estimated that both RP1 and RP2, in the first run of the J[th] level reconstruction, yield $(N/2^{J-1})/2$ pairs of output coefficients, whereas both RP3 and RP4 yield $N/2^{J-1}$ pairs of output coefficients, a total of $3/2\,N/2^{J-1}$ pairs of output coefficients in the first run. The total number of cycles in run 1 is then given by

$$4(N/2^{J-1})/2 \tag{15}$$

Thus, the total time, *T4*, required to yield *n* pairs of output coefficients for the J[th] level reconstruction of an *NxM* image on the 4-parallel architecture can be estimated as

$$T4 = \left(\rho_4 + 2N/2^{J-1} + 2\left(n - 3/2\,N/2^{J-1}\right)/2\right)t_4$$

$$T4 = \left(\rho_4 + 2N/2^{J-1} + \left(n - 3/2\,N/2^{J-1}\right)\right)t_p/4k \tag{16}$$

$$T4 = \left(\rho_4 + 1/2\,N/2^{J-1} + n\right)t_p/4k \tag{17}$$

The term $(n - 3/2\,N/2^{J-1})$ in (16) represents the total cycles of run 2 and all subsequent runs.

The speedup factor, *S4,* is then given by

$$S4 = \frac{T1}{T4} = \frac{\left(\rho_1 + N/2^{J-1} + 2n\right)t_p/2k}{\left(\rho_4 + 1/2\,N/2^{J-1} + n\right)t_p/4k}$$

For large *n* it reduces to

$$S4 = \frac{4(1/2\,N/2^{J-1} + n)}{(1/2\,N/2^{J-1} + n)} = 4 \tag{18}$$

Thus, the 4-parallel architecture is 4 times faster than the single pipelined architecture.
The throughput, *H,* which can be defined as number of output coefficients generated per unit time, can be written for each architecture as

$$H(\text{sin}\,gle) = n/(\rho_1 + N/2^{J-1} + 2n)\,t_p/2k$$

The maximum throughput, H^{max}, occurs when *n* is very large ($n\to\infty$), thus,

$$\begin{aligned} H^{\max}(\text{sin}\,gle) &= H(\text{sin}\,gle)_{n\to\infty} \\ &= 2nkf_p/2n = kf_p \end{aligned} \tag{19}$$

$$H(2-parallel) = n/(\rho_2 + 1/2\,N/2^{J-1} + n)t_p/2k$$

$$\begin{aligned} H^{\max}(2-parallel) &= H(2-parallel)_{n\to\infty} \\ &= 2knf_p/n = 2kf_p \end{aligned} \tag{20}$$

$$H(4-parallel) = n/(\rho_4 + 1/2\,N/2^{J-1} + n)\,t_p/4k$$

$$\begin{aligned} H^{\max}(4-parallel) &= H(4-parallel)_{n\to\infty} \\ &= 4knf_p/n = 4kf_p \end{aligned} \tag{21}$$

Thus, the throughputs of the 2-parallel and the 4-parallel pipelined architectures have increased by factors of 2 and 4, respectively, as compared with the single pipelined architecture.
On the other hand, the power consumption of *l*-parallel pipelined architecture as compared with the single pipelined architecture can be obtained as follows. Let P_1 and P_l denote the power consumption of the single and *l*-parallel architectures without the external memory, and P_{m1} and P_{ml} denote the power consumption of the external memory for the single and *l*-parallel architectures, respectively. The power consumption of VLSI architecture can be estimated as

$$P = C_{total} \cdot V_0^2 \cdot f$$

where C_{total} denotes the total capacitance of the architecture, V_0 is the supply voltage, and *f* is the clock frequency. Then,

$$P_1 = C_{total} \cdot V_0^2 \cdot f_1/2\,,\quad P_l = l \cdot C_{total} \cdot V_0^2 \cdot f_l/l \qquad \text{and}$$

$$\begin{aligned} \frac{P_l}{P_1} &= \frac{l \cdot C_{total} \cdot V_0^2 \cdot f_l/l}{C_{total} \cdot V_0^2 \cdot f_1/2} = \frac{2f_l}{f_1} \\ &= 2\left(\frac{l \cdot k}{t_p}\right)\Big/2k/t_p = l \end{aligned} \tag{21}$$

While, P_{m1} and P_{ml} can be estimated as

$$P_{m1} = C^{m}_{total} \cdot V_0^2 \cdot f_1 \quad , \quad P_{ml} = 2 \cdot C^{m}_{total} \cdot V_0^2 \cdot f_l \, , \quad \text{and} \quad \frac{P_{ml}}{P_{m1}} = \frac{2 \cdot f_l}{f_1} = \frac{2 \cdot l \cdot k/t_p}{2k/t_p} = l \tag{22}$$

C^{m}_{total}, is the total capacitance of the external memory.

In summary, the above equations indicates that as degree of parallelism increases the speedup and the power consumption of the proposed architectures, without external memory, and the power consumption of the external memory increase by a factor of *l*, as compared with single pipelined architecture.

6. Comparisons

Table 5 provides comparison results of the proposed architectures with most recent architectures in the literature. The architecture proposed in (Lan & Zheng, 2005) achieves a critical path of one multiplier delay using very large number of pipelined registers, 52 registers. In addition, it requires a total line buffer of size 6N, which is a very expensive memory component, while the proposed architectures require only 4N. In (Rahul & Preeti, 2007), a critical path delay of Tm + Ta is achieved through optimal dataflow graph, but requires a total line buffer of size 10N.

In (wang et at., 2007), by rewriting the lifting-based DWT of the 9/7, the critical path delay of the pipelined architectures have been reduced to one multiplier delay but it requires a total line buffer of size 5.5N. In addition, it requires real floating-point multipliers with long delay that can not be implemented by using arithmetic shift method (Qing & Sheng, 2005). (Qing & Sheng, 2005) has illustrated that the multipliers used for scale factor *k* and coefficients $\alpha, \beta, \gamma,$ and δ of the 9/7 filter can be implemented in hardware using only two adders. Moreover, the fold architecture which uses one module to perform both the predictor and update steps in fact increases the hardware complexity, e.g., use of several multiplexers, and the control complexity. In addition, use of one module to perform both predictor and update steps implies both steps have to be sequenced and that would slow down the computation process.

In the 2-parallel architecture proposed in (Bao & Yong, 2007), writing results generated by CPs into MM (main memory) and then switching them out to external memory for next level decomposition is really a drawback, since external memory in real-time applications, e.g., in digital camera, is actually consist of charge-coupled devices which can only be scanned. In addition, it requires a total line buffer of size 5N for 5/3 and 7N for 9/7 while the proposed architectures require 2N and 4N for 5/3 and 9/7 respectively. The architecture requires also used of several FIFO buffers in the datapath, which are complex and very expensive memory components, while the proposed architectures require no such memory components.

The 2-parallel architecture proposed in (Cheng et al., 2006) requires a total line buffer of size 5.5N and use of two-port RAM to implement FIFOs, whereas, the proposed architectures require only use of single port RAM.

Architecture	Multi	Adders	Line Buff.	Computing time	Critical Path
Lan	12	12	6N	$2(1-4^{-j})NM$	Tm
Rahul	9	16	10N	$2(1-4^{-j})NM$	Tm + Ta
Wang	6	8	5.5N	$2(1-4^{-j})NM$	Tm
Ibrahim	10	16	4N	$2(1-4^{-j})NM$	Tm +2Ta
Cheng (2-parallel)	18	32	5.5N	$(1-4^{-j})NM$	Tm +2Ta
Bao (2-parallel)	24	32	7N	$(1-4^{-j})NM$	Tm +2Ta
Proposed (2-parallel)	18	32	4N	$(1-4^{-j})NM$	Tm +2Ta
Proposed (4-parallel)	36	64	4N	$1/2\ (1-4^{-j})NM$	Tm +2Ta

Tm: multiplier delay Ta: adder delay

Table 5. Comparison results of several 2-D 9/7 architectures

7. Conclusions

In this chapter, two high performance parallel VLSI architectures for 2-D IDWT are proposed that meet high-speed, low-power, and low memory requirements for real-time applications. The two parallel architectures achieve speedup factors of 2 and 4 as compared with single pipelined architecture.

- The advantages of the proposed parallel architectures :

i. Only require a total temporary line buffer (TLB) of size 2N and 4N in 5/3 and 9/7, respectively.
ii. The scan method adopted not only reduces the internal memory between CPs and RPs to a few registers, but also reduces the internal memory or TLB size in the CPs to minimum and allows RPs to work in parallel with CPs earlier during the computation.
iii. The proposed architectures are simple to control and their control algorithms can be immediately developed.

8. References

Bao-Feng, L. & Yong, D. (2007). "FIDP A novel architecture for lifting-based 2D DWT in JPEG2000," MMM (2), lecture note in computer science, vol. 4352, PP. 373-382, Springer, 2007.

Cheng-Yi, X.; Jin-Wen, T. & Jian, L. (2006). "Efficient high-speed/low-power line-based architecture for two-dimensional discrete wavelet transforms using lifting scheme," IEEE Trans. on Circuits & sys. For Video Tech. Vol.16, No. 2, February 2006, PP 309-316.

Dillin, G.; Georis B.; Legant J-D. & Cantineau, O. (2003). "Combined Line-based Architecture for the 5-3 and 9-7 Wavelet Transform of JPEG2000," IEEE Trans. on circuits and systems for video tech., Vol. 13, No. 9, Sep. 2003, PP. 944-950.

Ibrahim saeed koko & Herman Agustiawan, (2008). "Pipelined lifting-based VLSI architecture for two-dimensional inverse discrete wavelet transform," proceedings of the IEEE International Conference on Computer and Electrical Engineering, ICCEE 2008, Phuket Island, Thailand.

Lan, X. & Zheng N. (2005). "Low-Power and High-Speed VLSI Architecturefor Lifting-Based Forward and Inverse Wavelet Transform," IEEE tran. on consumer electronics, Vol. 51, No. 2, May 2005, PP. 379 –385.

Qing-ming Yi & Sheng-Li Xie, (2005). "Arithmetic shift method suitable for VLSI implementation to CDF 9/7 discrete wavelet transform based on lifting scheme," Proceedings of the Fourth Int. Conf. on Machine Learning and `Cybernetics, Guangzhou, August 2005, PP. 5241-5244.

Rahul. J. & Preeti R. (2007). "An efficient pipelined VLSI architecture for Lifting-based 2D-discrete wavelet transform," ISCAS, 2007 IEEE, PP. 1377-1380.

Wang, C.; Wu, Z.; Cao, P. & Li, J. (2007). "An efficient VLSI Architecture for lifting-based discrete wavelet transform," Mulltimedia and Epo, 2007 IEEE International conference, PP. 1575-1578.

3

Contour-Based Binary Motion Estimation Algorithm and VLSI Design for MPEG-4 Shape Coding

Tsung-Han Tsai, Chia-Pin Chen, and Yu-Nan Pan
Department of Electronic Engineering
National Central University, Chung-Li, Taiwan, R.O.C

1. Introduction

MPEG-4 is a new international standard for multimedia communication [1]. It provides a set of tools for object-based coding of natural and synthetic videos/audios. MPEG-4 also enables content-based functionalities by introducing the concept of video object plane (VOP), and such a content-based representation is a key to enable interactivity with objects for a variety of multimedia applications. The VOP is composed of texture components (YUV) and an alpha component [2]-0. The texture component contains the colorific information of video object, and the alpha component contains the information to identify the pixels. The pixels which are inside an object are opaque and the pixels which are outside the object are transparent. MPEG-4 supports a content-based representation by allowing the coding of the alpha component along with the object texture and motion information. Therefore, MPEG-4 shape coding becomes the key technology for supporting the content-based video coding.
MPEG-4 shape coding mainly comprises the following coding algorithms: binary-shaped motion estimation/motion compensation (BME/BMC), context-based arithmetic coding (CAE), size conversion, mode decision, and so on. As full search (FS) algorithm is adopted for MPEG-4 shape coding, most of the computational complexity is due to binary motion estimation (BME). From the profiling on shape coding in Fig. 1, it can be seen that BME contributes to 90% of total computational complexity of MPEG-4 shape encoder. It is well known that an effective and popular technique to reduce the temporal redundancy of BME, called block-matching motion estimation, has been widely adopted in various video coding standard, such as MPEG-2 0, H.263 [5] and MPEG-4 shape coding [1]. In block-matching motion estimation, the most accurate strategy is the full search algorithm which exhaustively evaluates all possible candidate motion vectors over a predetermined neighborhood search window to find the global minimum block distortion position.

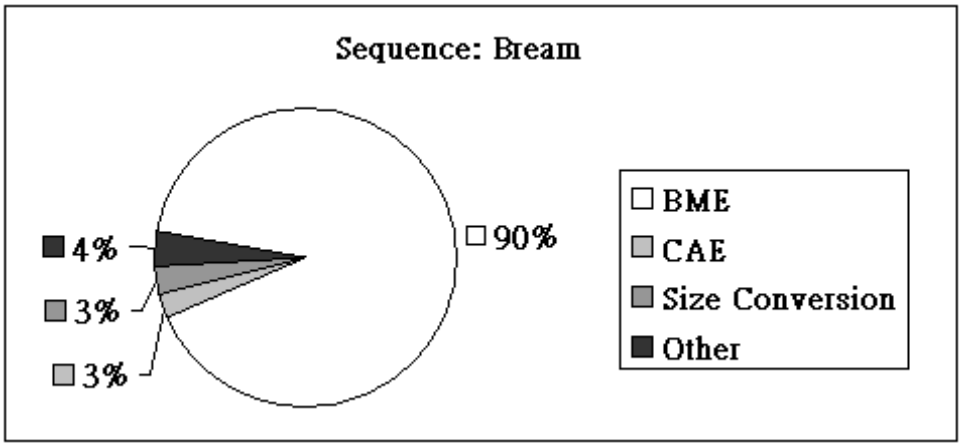


Fig. 1. Computational complexity of MPEG-4 shape encoder.

Fast BME algorithms for MPEG-4 shape coding were presented in several previous papers [6]-[8]. Among these techniques, our previous work, contour-based binary motion estimation (CBBME), largely reduced the computational complexity of shape coding [9]. It is applied with the properties of boundary search for block-matching motion estimation, and the diamond search pattern for further improvement. This algorithm can largely reduce the number of search points to 0.6% compared with that of full search method, which is described in MPEG-4 verification model (VM) [2].

In contrast with algorithm-level developments, architecture-level designs for shape coding are relatively less. Generally, a completed shape coding method should include different types of algorithms. In CAE part, it needs some bit-level operation. However, in binary motion estimation part, a high speed search method is needed. With these algorithm combinations on the shape coding, implementation should not be as straightforward as expected and it offers some challenges especially on architecture design. Since MPEG-4 shape coding has features of high-computing and high-data-traffic properties, it is suitable with the consideration of efficient VLSI architecture design. Most literatures have also been presented to focus on the main computation-expensive part, BME, to improve its performance [10]. Additionally, CAE is also an important part for architecture design and discussed in [11]-[12]. They utilized the multi-symbol technique to accelerate the arithmetic coding performance. As regards the complete MPEG-4 shape coding, some of these designs utilized array processor to perform the shaping coding algorithm [13]-[15], while others used pipelined architecture [16]. They can reach the relative high performance at the expense of these high cost and high complexity architectures. All of them intuitively apply the full search algorithm for easy realization on architecture design. However, the algorithm-level achievement on the large reduction of computation complexity is attractive and not negligible. This demonstrates that, without the supporting on an efficient algorithm, the straightforward implementation based on full search algorithm is hard to reach a cost-effective design.

In this paper, we proposed a fast BME algorithm, diamond boundary search (DBS), for MPEG-4 shape coding to reduce the number of search points. By using the properties of

block-matching motion estimation in shape coding and diamond search pattern, we can skip a large number of search points in BME. Simulation results show that the proposed algorithm can marvelously reduce the number of search points to 0.6% compared with that of full search method, which is described in MPEG-4 verification model (VM)[2]. Compared with other fast BME in [6]-[7], the proposed BME algorithm uses less search points especially in high motion video sequences, such as 'Bream' and 'Forman'. We also present an efficient architecture design for MPEG-4 shape coding. This architecture is elaborated based on our fast shape coding algorithm with the binary motion estimation. Since this block-matching motion estimation can achieve the high performance based on the information of boundary mask, the dedicated architecture needs some optimization and consideration to reduce the memory access and processing cycles. Experimental results also demonstrate the equal performance on full-search based approach. This paper contributes a comprehensive exploration of the cost-effective architecture design of shaping coding, and is organized as follows.

In Section 2 the binary motion estimation in shape coding is described. We describe the highlights of the proposed fast BME algorithm for MPEG-4 shape coding in Section 3. The design exploration on CBBME is described in Section 4. In Section 5, the architecture design based on this BME algorithm is proposed. In Section 6, we present the implementation results and give some comparisons. Finally we summarize the conclusions in Section 7.

2. BME for MPEG-4 Shape Coding

The MPEG-4 VM [2] describes the coding method for binary shape information. It uses block-matching motion estimation to find the minimum block distortion position and sets the position to be motion vector for shape (MVS). The procedure of BME consists of two steps: first to determine motion vector predictor for shape (MVPS) and then to compute MVS accordingly.

MVPS is taken from a list of candidate motion vectors. As indicated in Fig. 2, the list of candidate motion vectors includes the shape motion vectors (MVS) from the three binary alpha blocks (BABs) which are adjacent to the current BAB and the texture motion vectors (MV) associated with the three adjacent texture blocks. By scanning the locations of MVS1, MVS2, MVS3, MV1, MV2 and MV3 in this order, MVPS is determined by taking the first encountered MV which is valid. Note that if the procedure fails to find a defined motion vector, the MVPS is set to (0, 0).

Based on MVPS determined above, the motion compensated (MC) error is computed by comparing the BAB indicated by the MVPS and current BAB. If the computed MC error is less or equal to 16xAlphaTH for any 4x4 sub-blocks, the MVPS is directly employed as MVS and the procedure terminates. Otherwise, MV is searched around the MVPS while computing sum of absolute difference (SAD) by comparing the BAB indicated by the MV and current BAB. The search range is ±16 pixels around MVPS along both horizontal and vertical directions. The MV that minimizes the SAD is taken as MVS and this is further interpreted as MV Difference for shape (MVDS), i.e. MVDS=MVS-MVPS.

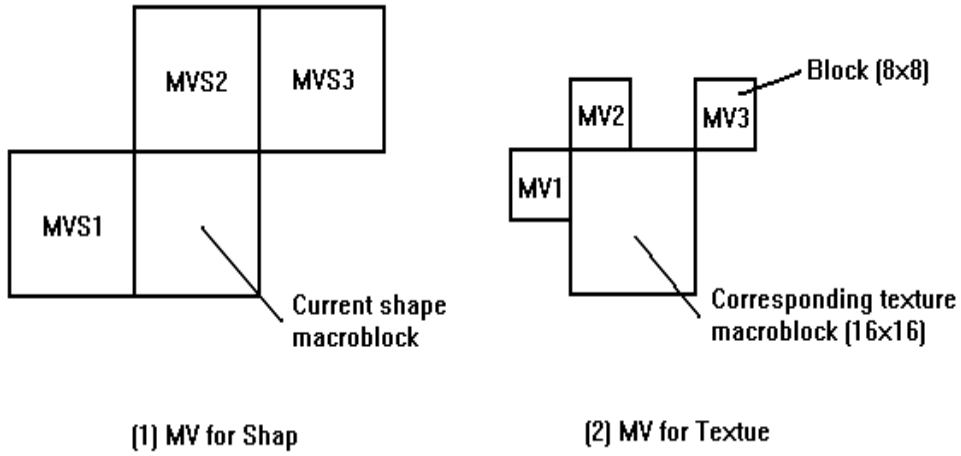


Fig. 2. Candidates for MVPS in shape VOP and texture VOP.

If more than one MVS minimize SAD by an identical value, the MVDS that minimizes the code length of MVDS is selected. If more than one MVS minimize SAD by an identical value with an identical code length of MVDS, MVDS with smaller vertical element is selected. If the vertical elements are also the same, MVDS with smaller horizontal element is selected.
After binary motion estimation, motion compensated block is constructed from the 16x16 BAB with a border of width 1 around the 16x16 BAB (bordered MC BAB). Then, context-based arithmetic encoding (CAE) is adopted for shape coding.

3. Proposed Contour-Based Binary Motion Estimation (CBBME) Method

The basic concept of the proposed BME method is that the contour of video objects in current BAB should overlap that in the motion compensated BAB, which is determined by binary motion estimation [9]. Therefore, those search positions, which contour lays apart from the contour of video objects in current BAB, can be skipped and the reduced number will be enormous. Moreover, based on the property that most real-world sequences have a central biased motion vector distribution [23], we use weighted SAD and diamond search pattern for furthermost improvement.

3.1 Definition of Boundary Pixel

In order to decide whether the pixel is on the contour of VOP, the boundary pixel is determined by the following procedure:

- If current pixel is opaque and one of its four adjacent pixels is transparent, the current pixel directly employed as boundary pixel.

Fig. 3 shows the correlation between current pixel and its four adjacent pixels. In the figure, the light grey area corresponds to the pixels outside the binary alpha map. It can be seen that the pixels outside the binary alpha map will not be taken into consideration, and the number of adjacent pixels will change into two or three.

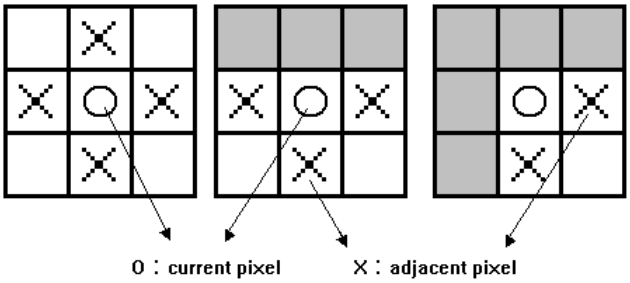


Fig. 3. The correlation between current pixel and adjacent pixel. Where gray part denotes pixels outside the VOP.

3.2 Boundary Search (BS)

According to the boundary pixels in VOP, we build a mask for BME process in MPEG-4 shape coding. Fig. 4 shows an example of boundary mask from the 'Foreman' sequence. In this figure, the white area denotes the efficient search position for fast BME, and it is much more efficient than the fast algorithm 0 illustrated in Fig. 5.

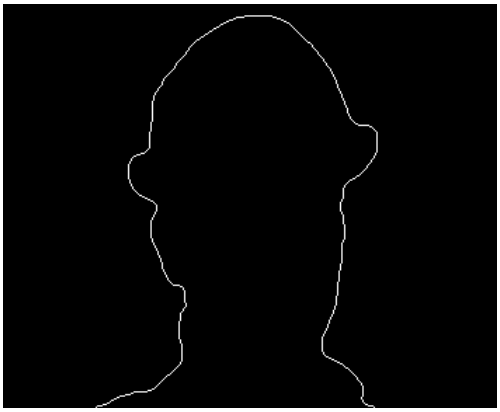


Fig. 4. Example of a boundary mask for shape coding. White area denotes boundary pixel.

A suggested implementation of the proposed Boundary search (BS) algorithm for shape coding is processed as follows:

Step 1. Perform a pixel loop over the entire reference VOP. If pixel (x,y) is an boundary pixel, set the mask at (x,y) to '1'. Otherwise set the mask at (x,y) to '0'.

Step 2. Perform a pixel loop over the entire current BAB. If pixel (i,j) is a boundary pixel, set (i,j) to be "reference point", and terminate the pixel loop. This step is illustrated in Fig. 6(b). Therefore, there is only one reference point in current BAB.

Step 3. For each search point within ±16 search range, check the pixel (x+i, y+j) which is fully aligned with the "reference point" from the current BAB. If the mask value at (x+i, y+j) is '1', which means that the reference point is on the

boundary of reference VOP, the procedure will compute SAD of the search point (x, y). Otherwise, SAD of the search point (x, y) will not be computed, and the processing continues at the next position. Fig. 6(a) shows an example of this step. The search points in (x1, y1) and (x2, y2) will be skipped by this procedure, while the SAD will be computed in (x3, y3).

Step 4. When all the search points within ±16 search range is done, the MV that minimizes the SAD will be taken as MVS. Fig. 7 illustrates the overall scheme of proposed BS algorithm for MPEG-4 shape coding.

In the worst case, the proposed BS algorithm needs (256+ $(16+1)^2$) determinations to check whether the pixel is a boundary pixel. For each non-skipped search point, the SAD obtained by 256 exclusive-OR operations and 255 addition operations was taken as the distortion measure. However, based on BS algorithm, the number of non-skipped search points was reduced significantly, and the additional computational load due to BS algorithm was negligible.

Fig. 5. Example of an effective search area, which has been proposed in reference 0.

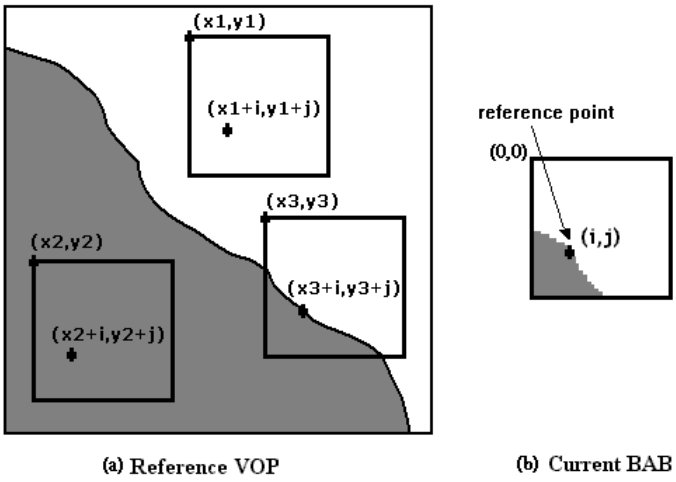


Fig. 6. The illustration of the proposed BS algorithm.

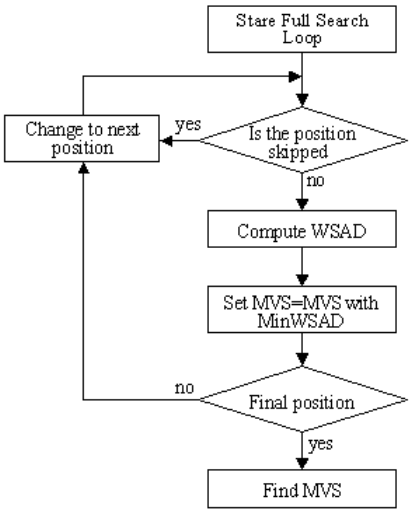


Fig. 7. Flow chart for proposed BS algorithm.

3.3 Diamond Boundary Search (DBS)

A better solution for block-matching motion estimation is to perform the search using a diamond pattern because of center-biased motion vector distribution characteristic. This is achieved by dividing the search area into diamond shaped zones and using half-stop criterion [17]. Fig. 8 shows an example of diamond shaped zones in a ±5 search window, and each number denotes the search zone in the search procedure.

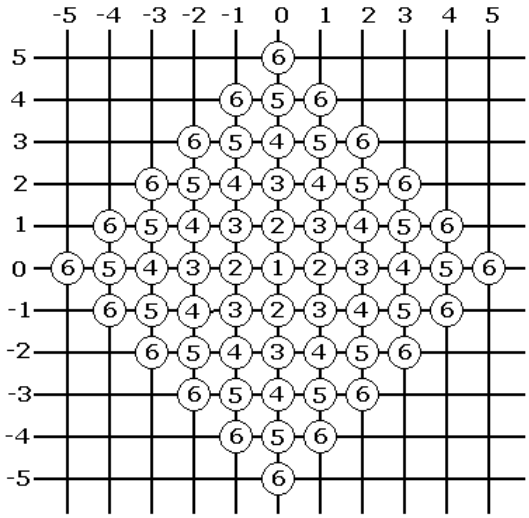


Fig. 8. A ±5 search window using diamond-shaped zones.

We combine the proposed BS algorithm with diamond-shaped zones, called DBS, and give different thresholds (Th_n) for each search zone for furthermost improvement. The procedure is explained in below.

Step 1. Construct diamond-shaped zones around MVPS within ±16 search window. Se t n=1.

Step 2. Calculate SAD for each search point in zone n. Let MinSAD be the smallest SAD up to now.

Step 3. If MinSAD $\leqq Th_n$, goto Step 4. Otherwise, set n=n+1 and goto Step 2.

Step 4. The motion vector is chosen according to the block corresponding to MinSAD.

3.4 Weight SAD (WSAD)

In MPEG-4 shape coding, the reference BAB with minimum SAD will be selected as motion compensated BAB. However, the BAB with minimum SAD may be far away from the original. It means that encoder should waste much more bits to code MVDS. Some previous motion estimation algorithms for color space used the concept of the weighted SAD (WSAD) to compensate the distortion [24]. In this paper, we proposed the similar concept of WSAD which takes both SAD and MVDS into consideration as the distortion measure. The WSAD is given by

$$WSAD=W_1*SAD+W_2*(|mvds\_x|+|mvds\_y|) \quad (1)$$

$$SAD=\sum\sum|p_{i-1}(i+u,j+v)-p_i(i,j)| \quad \ldots \quad (2)$$

where *mvds_x* is MVDS in the horizontal direction and *mvds_y* is MVDS in the vertical direction. *W1* and *W2* denote the weighting values for SAD and MVDS, respectively. The WSAD is evaluated in every search points and the BAB with minimum WSAD is selected as

motion compensated BAB. Based on the experimentation, *W1* and *W2* are determined as 10 and 7, respectively. The improvement of WSAD on bit-rate can compensate for the drawback when fast BME algorithm was adopted. Since the number of search points has been reduced significantly, the computational power due to calculate the WSAD will increase negligibly.

4. Design Exploration on CBBME

In this section, two approaches are developed based on CBBME. One is the center-biased motion vector distribution. The other is the search range shrinking. Both of them can further reduce the computation complexity in BME.

4.1 Center-biased motion vector distribution

The property is obvious that real-world sequences have a central biased motion vector distribution. This can be achieved by dividing the search area into diamond shaped zones and using half-stop criterion [17]. Diamond shaped zone has the higher center-biased distribution. The closer the search area to the center position, the less the number of the search zone in the search procedure.

4.2 Search Range Shrinking

With the property on Section 4.1, it is possible to reduce the default search range to a less size of search range. Default search range is ±16 in standard and it is straightforward used in conventional works. With the aid of WSAD in CBBME and the diamond shaped zones on search range, we make a search range shrinking technique from ±16 to ±13 pixels in our experimentation. Table 1 shows the simulation result of the proposed binary motion estimator using the above two techniques. This result supports the usage of the ±13 shrinking search range. An important contribution is that the WSAD makes some improvement on bit-rate. Therefore it takes similar even less bits to represent the shape per VOP in the same quality.

Sequence	Full Search (SR=±16)		Proposed Architecture with limited SSB (SR=±13)	
	Bits/VOP	%	Bits/VOP	%
Bream	1599.80	100	1599.27	99.97
News	890.22	100	890.18	100.00
Foreman	1186.94	100	1183.12	99.68
Children	2056.03	100	2055.43	99.97

Table 1. Performance comparison of FS and proposed architecture.

Fig. 9 shows the algorithm flow of CBBME. First, we construct diamond-shaped zones around motion vector predictor for shape (MVPS) within shrinking search range. Second, we calculate WSAD for each search point. Let MinWSAD be the smallest SAD up to now. If MinWSAD is smaller or equal to a threshold (*Th*) for each search zone, then the motion vector is chosen according to the block corresponding to MinWSAD; otherwise, it changes to next position and calculates the WSAD again.

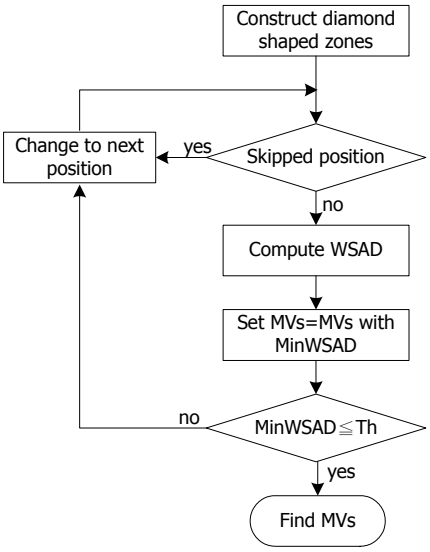


Fig. 9. Flow chart for the CBBME algorithm.

5. Architecture Design for MPEG-4 Shape Coding

The proposed MPEG-4 shape coding system, as illustrated in Fig. 10 consists of five major modules: BAB type decision, size conversion, BME, CAE and variable length coding (VLC). Based on the computational complexity analysis in Fig. 1, it can be seen that BME, size conversion and CAE take a great part of the processing time. In this section we propose an efficient architecture for these main modules with some design optimization and consideration.

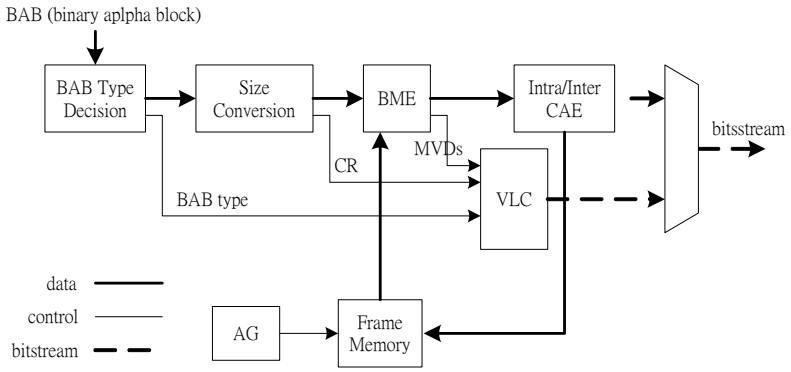


Fig. 10. Block diagram of MPEG-4 shape encoding.

5.1 Binary Motion Estimation

Fast motion estimation architectures were presented in several previous papers [12]-[15]. The performance could be high but high cost and high complexity architectures are always needed. Therefore, a novel and efficient architecture for binary motion estimation using proposed CBBME algorithm is proposed. Fig. 11 shows the block diagram of the proposed BME architecture. It mainly consists of a Boundary Pixel Detector, a processing element (PE) Array, a Compare and Selection (CAS) module and two main memories for search range (SR) and BAB buffer. Boundary Pixel Detector finds the "reference point" in current BAB and checks whether the candidate search positions are non-skipped positions. PE Array performs the binary motion estimation. CAS finds the minimum WSAD and its MVDS for shape coding.

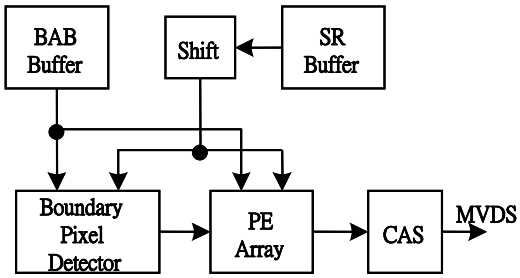


Fig. 11. Block diagram of the proposed BME architecture.

A. Boundary Pixel Detector

Boundary Pixel Detector is a design dedicated to our CCBME algorithm. From the analysis in Section 4, since the search range has been reduced to ±13, the search points with one dimension is 13+13+1=27 and induce a detected region with 27×27 search points. Including the BAB size of 16×16, totally 42×42 pixels are used as the total search area.

According to the effect on 27×27 search points, a multiple of 3 is considered on architecture design. In our binary motion estimator, a 3×3 PE array is used. Therefore, we separate the search positions into a 9×9 sub-search-block (SSB) array structure, and each SSB contains 3×3 search positions for the calculation of PE array. Fig.12 illustrates the detected region and its related array structure with totally 9x9= 81 SSBs.

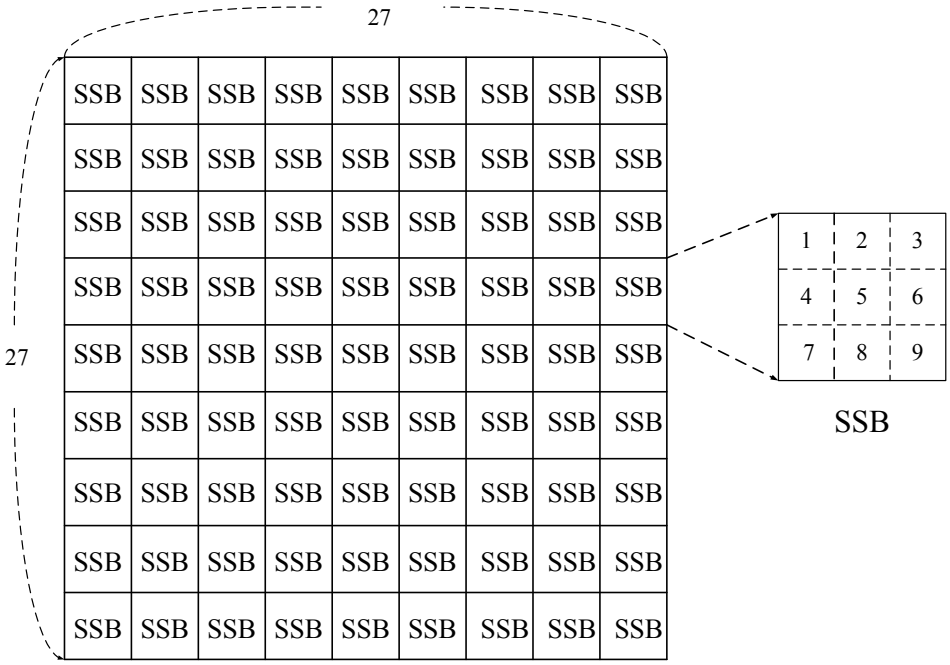


Fig. 12. The 27x27 detected region composes of 9x9=81 SSBs; each SSB contains 9 checking pixel.

In each SSB, 9 checking pixels are included. In addition, in order to detect boundary pixels and obtain the bordered MC-BAB, a border with the width of one pixel around the 42×42 search area, called bordered search area, is applied. As depicted in Fig. 13, the pixels in the grey area are the border pixels. This range indicates the total needed pixels in memory.
Boundary Pixel Detector first selects a boundary pixel *(i,j)* as "reference point" in 16×16 current BAB. Based on the reference point, a 27×27 detected region is generated as shown in Fig. 14. Each pixel in detected region, called checking pixel, denotes whether the correlative search position is non-skipped search position or not. If the checking pixel belongs to a boundary pixel, the correlative search position is denoted as non-skipped search position. Hereafter, Boundary Pixel Detector detects 9 checking pixels, which are relative to the coding SSB in detected region. As shown in Fig. 14, the pixels in the dark area are used for detecting boundary pixel. If all of the checking pixels in light grey area are not boundary pixels, it means that all search positions in the SSB are skipped search positions. Therefore, the coding SSB will not be processed in PE Array. Otherwise, the PE Array calculates WSAD of these 9 search positions in coding SSB.

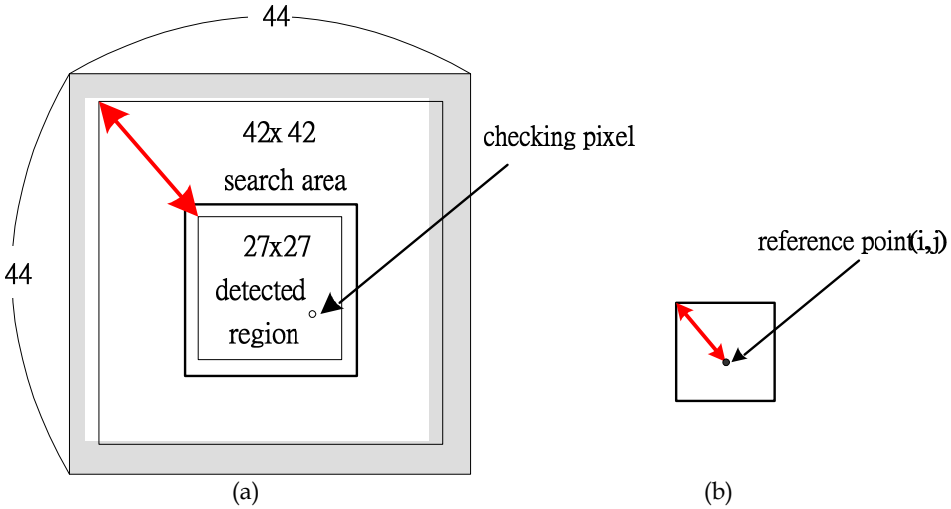


Fig. 13. (a) 44×44 bordered search area, (b) 16×16 current BAB.

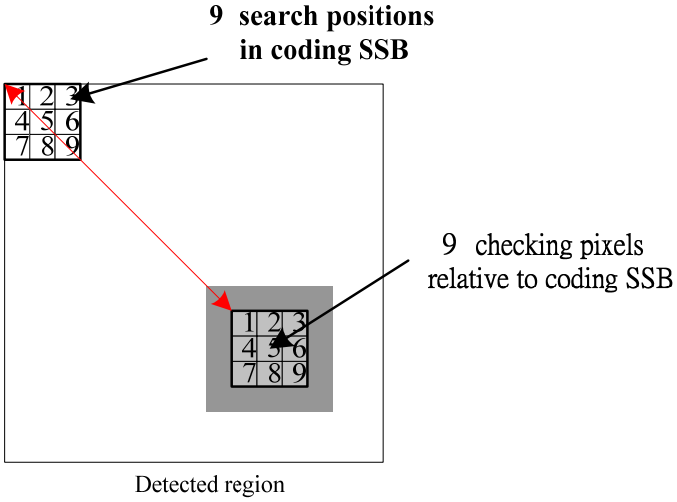


Fig. 14. The relation between search positions and detected region.

B. PE Array and CAS

PE array architecture which performs the binary motion estimation algorithm is shown in Fig. 15(a). The array is the 3×3 architecture and totally consists of 9 PEs. In this architecture, 18-bits reference data (denoted as *Ref[17:0]*) are read from SR buffer, and 16-bits current data are read from BAB buffer. The reference data are broadcasted to all PEs (*Ref[17:2]* for PE1, PE4 and PE7; *Ref[16:1]* for PE2, PE5 and PE8; *Ref[15:0]* for PE3, PE6 and PE9 respectively), while the current data is delayed and fed to the corresponding PE. Each PE calculates the WSAD of a search position in coding SSB. It is noted that only the non-skipped SSB, which is

detected from Boundary Pixel Detector, is processed in PE Array.

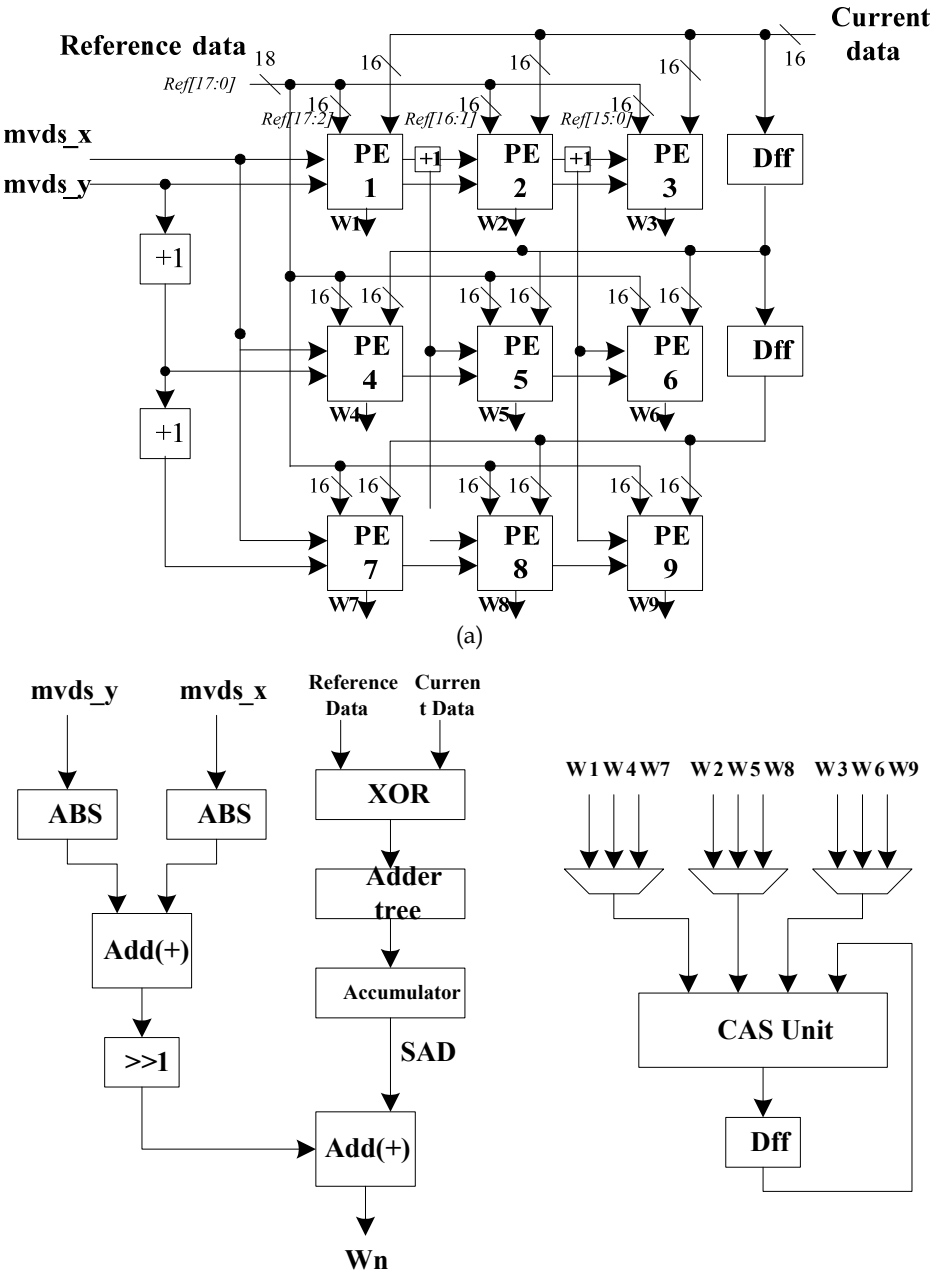


Fig. 15. Architecture of (a) PE Array (b) PE element (c) CAS unit.

The weighted data is calculated by adding absolute MVDS in both horizontal and vertical directions and shifting right one bit. From the analysis in our previous paper [9], the ratio for *V2/V1* will not make large difference from 0.5 to 0.8. And in that range our WSAD is indeed better than the result for SAD. For reducing the computational complexity, *V1* and *V2* in (1) are determined as 1 and 0.5, respectively. The architecture of PE element is shown in Fig. 15(b). It produces WSAD with the sum of weighted data and SAD, where *Wn* means the WSAD value for each PE element from *n*=1 to 9.
The architecture of Compare and Selection (CAS) module is shown in Fig. 15(c). It finds the smallest WSAD and its MVS in coding SSB and feedbacks the smallest WSAD as the input for the next SSB.

5.2 Size Conversion

In MPEG-4 shape coding, rate control and rate reduction are realized through size conversion. Fig. 16 shows the block diagram of size conversion module.

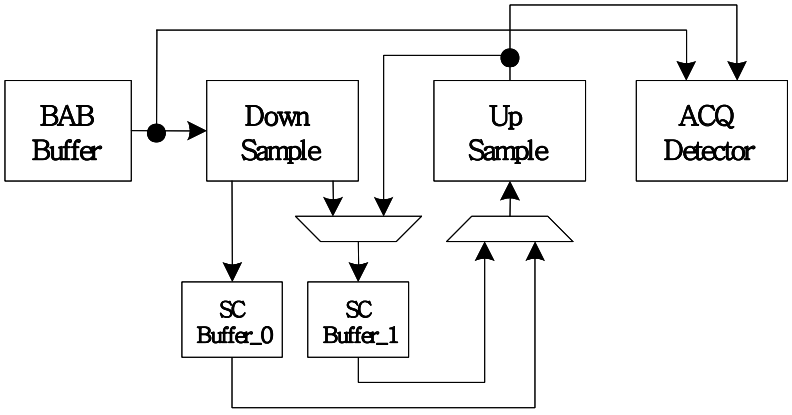


Fig. 16. Block diagram of size conversion module.

It consists of three major units: down-sample, up-sample and accepted quality (ACQ) detector. The bordered BAB is read from BAB buffer and down-sampled to 4×4 and 8×8 BAB in "SC Buffer_0" and "SC Buffer_1" respectively. Then, the 4×4 BAB is up-sampled to SC Buffer_1, and 8×8 BAB is up-sampled to ACQ detector by up-sample unit. ACQ detector calculates the conversion error between the original BAB and the BAB which is down-sampled and reconstructed by up-sample unit. ACQ also needs to determine the conversion ratio. In down-sample procedure, several pixels are down-sampled to one pixel, while interpolated pixels are produced between original pixels in up-sample procedure. To compute the value of the interpolated pixel, a border with the width of two around the current BAB is used to obtain the neighboring pixels (A~L), and the unknown pixels are extended from the outermost pixels inside the BAB. The template and pixel relationship used for up-sampling operation can be referred as in MPEG-4 standard and [15].
Since the implementation of the down-sample and ACQ detector is relatively simple, we only address the design of up-sample unit here. The block diagram of up-sample unit is

shown in Fig. 17. Due to the window-like slicing operations, the up-sample can be easily mapped into a delay line model. A delay line model is used to obtain the pixels in A~L, which determine interpolated pixels (P1~P4). Based on the pixels in A~L, four 4-bits threshold values are obtained from "Table CF". "UP_PE" generates corresponding values for comparison. After comparison between threshold values and the values from "UP_PE", four interpolated pixels (P1~P4) are stored in "Shift Register" and outputted later.

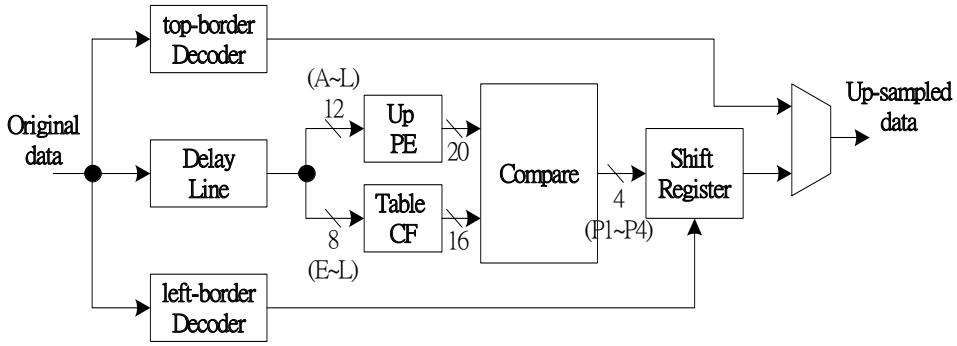


Fig. 17. Block diagram of up-sample unit.

5.3 Context Based Arithmetic Encoder (CAE)

CAE architecture mainly comprises the context generation unit and the binary arithmetic coder [18]-[19]. Fig. 18(a) shows the block diagram of CAE module for our design.

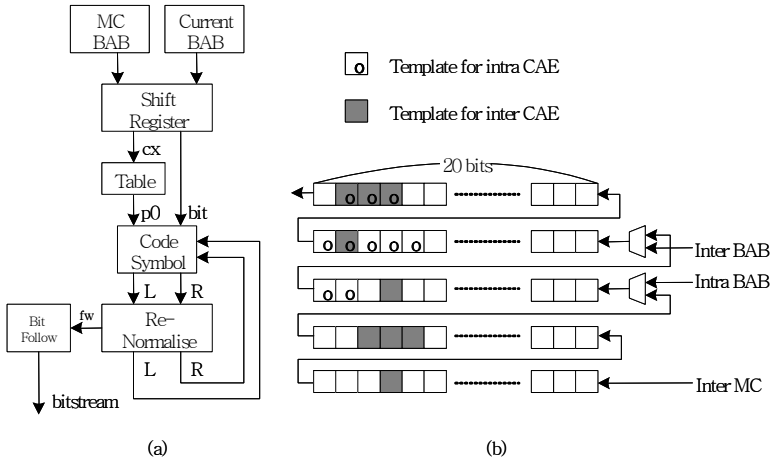


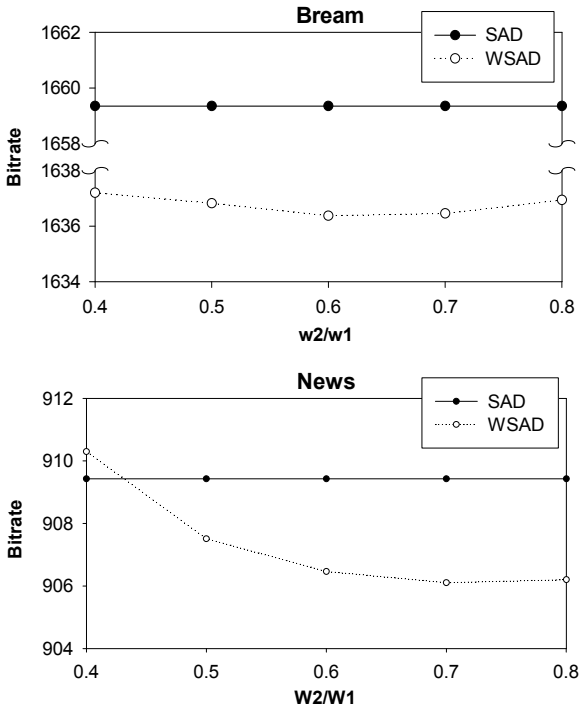
Fig. 18. (a) Block diagram of CAE module. (b). Illustration of the Shift Register.

As mentioned before, for pixel-by-pixel processing in CAE, it basically uses the raster scan order. Since most of the execution time is spent on the context generation in the CAE, the "Shift Register" is used to obtain context and the related operation is illustrated in Fig. 18(b) [20]. Data in the shift registers can be effectively reused and thus this redundant data

accesses can be removed. In Fig. 18(b) all the rectangles are represented as registers. Pixels in current BAB and MC-BAB are first loaded into the Shift Register, and then shifted left one bit at every clock cycle. Registers in context box are arranged such that various contexts can be achieved. For intra-CAE mode, the first three rows of Shift Register are used to store current BAB. The first two rows of Shift Register are used to store current BAB and the last three rows are used to store MC-BAB in inter-CAE mode. Therefore, the context (*cx*) and coded bit (*bit*) are obtained from Shift Register per cycle.

6. Implementation Results and Comparisons

We use three MPEG-4 test video sequences of CIF (352×288) format for experiment: Bream, News and Foreman. The three sequences characterize a variety of spatial and motion activities. Each sequence consists of 300 VOP's of arbitrary shape. A search range of ±16 pixels is used and frame-based lossless shape coding is performed for all the test sequence.
The comparisons of SAD and WSAD using full search algorithm are shown in Fig. 19. In this figure, the correlations between bit-rate and the ratio of *W2* to *W1* (*W2*/*W1*) are also shown in Fig. 19.



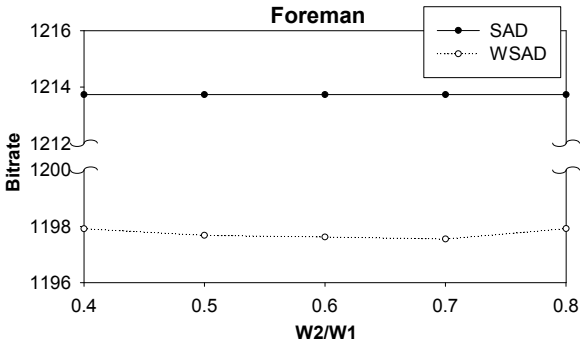


Fig. 19. Performance comparisons of SAD and WSAD using full search algorithm.

It can be seen that the result of using WSAD as the distortion measure takes less bits than that of using SAD except "News" in $W2/W1$= 0.4. This is because the "News" sequence is a low motion video sequence, and WSAD is of no benefit in such sequence. As the result of Fig. 19, $W1$ and $W2$, which is derived from (1), are determined as 10 and 7, respectively. Based on the weighting values, the average number of bits to represent the shape per VOP is shown in Table 2. The percentages compared to the results of full search, which uses SAD as the distortion measure, are also shown in Table 2. An important contribution is that the WSAD makes some improvement on bit-rate, and it can compensate for the inaccuracy of motion vector when the fast search algorithm is used.

Sequence	Full Search with SAD		Full Search with WSAD	
	Bits/VOP	%	Bits/VOP	%
Bream	1659.35	100	1636.46	98.62
News	909.43	100	906.10	99.63
Foreman	1213.73	100	1197.55	98.67

Table 2. Performance comparison of SAD and WSAD based on full search algorithm in bit-rate ($W1$=10, $W2$=7).

Fig. 20 shows the comparison of various search algorithms. It can be seen that the proposed BS and DBS algorithm take less search points than FS algorithm and the algorithms described in 0-0.

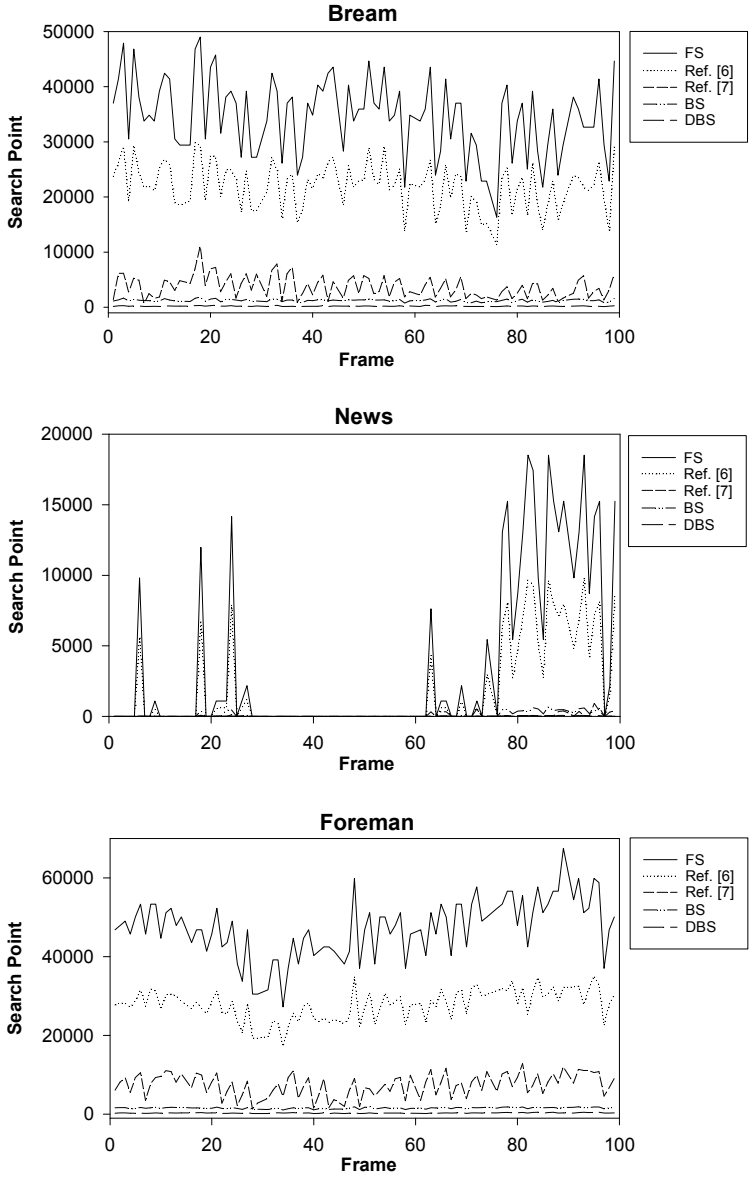


Fig. 20. Performance comparisons of various search algorithms.

Table 3 shows the number of search points (SP), and Table 4 shows the average number of bits to represent the shape per VOP. The percentages compared to the results corresponding to the full search in MPEG-4 VM are also shown in these tables. "BS" denotes the proposed algorithm without using diamond search pattern and "DBS" denotes the proposed algorithm

using diamond search pattern. Table 5 shows the runtime simulation results of various BME algorithms. It is noted that the BME algorithm 0 takes much more computational complexity because of the generation of the mask for effective search area.

Sequence	Full Search	Ref. 0		Ref. 0		Proposed (BS)		Proposed (DBS)	
	SP	SP	%	SP	%	SP	%	SP	%
Bream	10,504,494	6,495,838	61.84	1,560,221	14.85	367,115	3.49	67,232	0.64
News	747,054	403,131	53.96	6,568	0.88	24,923	3.36	2,048	0.27
Foreman	9,085,527	5,093,409	56.06	1,564,551	17.22	287,272	3.16	57,214	0.63

Table 3. Total search points for various search algorithms.

Sequence	Full Search	Ref. 0		Ref. 0		Proposed (BS)		Proposed (DBS)	
	Bits/VOP	Bits/VOP	%	Bits/VOP	%	Bits/VOP	%	Bits/VOP	%
Bream	1659.35	1659.35	100.00	1683.28	101.44	1655.78	99.78	1669.58	100.62
News	909.43	909.43	100.00	902.68	99.26	910.82	100.15	908.39	99.89
Foreman	1213.73	1213.73	100.00	1219.47	100.47	1209.35	99.64	1219.23	100.45

Table 4. Average bit-rate for various search algorithms.

Compared with the full search method, the proposed fast BME algorithm (BS) needs 3.5% search points and takes equal bit rate in the same quality. By using the diamond-shaped zones, the proposed algorithm (DBS) needs only 0.6% search points. Compared with other fast BME 0-0, our algorithm uses less search points, especially in high motion video sequences, such as 'Bream' and 'Foreman'.

Sequence	Full Search	Ref. 0		Ref. 0		Proposed (BS)		Proposed (DBS)	
	ms	ms	%	ms	%	ms	%	ms	%
Bream	57718.67	62776.36	108.76	7782.83	13.48	8678.02	15.04	1738.24	3.01
News	3830.13	4202.36	109.72	35.05	0.92	574.04	14.99	22.16	0.58
Foreman	44877.19	52574.24	117.15	7507.74	16.73	6487.52	14.46	1703.88	3.80

Table 5. Runtime simulation results of various search algorithms.

Table 6 shows the number of non-skipped SSB per PE. Due to the contribution of the proposed CBBME algorithm, the number of non-skipped SSB is reduced largely. It can be seen that the average number of non-skipped SSB is much less than the total number of SSB, which is denoted as 81 per PE. In the worst case, the additional non-skipped SSB is usually in the positions with large motion vector, which does not tend to be the adoptive MV.

Sequence	Average non-skipped SSB per PE	Maximum non-skipped SSB per PE
Bream	12.78	33
News	10.73	16
Foreman	10.94	24
Children	13.40	46

Table 6. Average and maximum number of non-skipped SSB per PE.

Therefore, in the binary motion estimator, the number of non-skipped SSB is limited to 32 from experimentation in maximum situation. For average situation, we set the number as 12.In our architecture, the processing cycles for various BAB types are shown in Fig. 21.

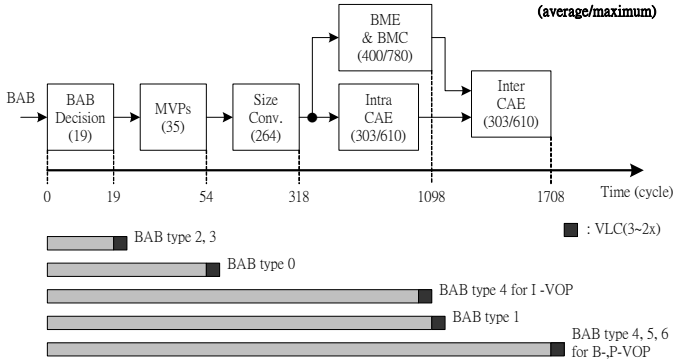


Fig. 21. Processing cycles for various BAB types.

Totally seven types of mode are described for BAB. The number in parentheses indicates the latency of that module with average and maximum number of cycles. Notice that the processing cycles of BME and CAE depend on the content of BAB. To complete one BAB processing in the worst case scenario, our architecture requires 1708 clock cycles, including: 19 clock cycles for mode decision, 35 clock cycles for identifying MVPS, 264 clock cycles for size conversion, 780 clock cycles for BME, and 610 clock cycles for inter CAE.
Actually, few literatures explored the architecture design of shape coding. In [10], they only designed the BME. We can extract our data for BME part as comparison in Table 7. In terms of the whole shape coding,

	E. A. Al_Qaralleh's [10]	Proposed
Gate count	11582	10523
One BAB processing (cycles)	563	780
Comment	Partial design (Only BME implemented)	Completed design

Table 7. Comparison with BME only.

Table 8 illustrates some results with different architectures. For our design the size of BAB

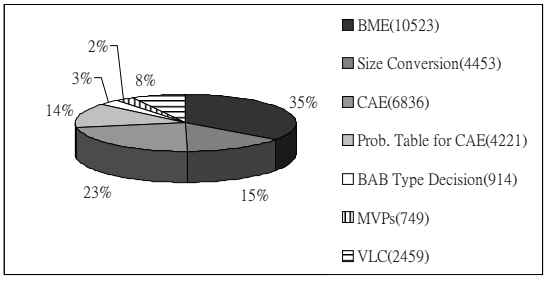
buffer and SR buffer are 16×16 and 44×44 respectively. The average and maximum numbers of non-skipped SSB are determined as 12 and 32 from experiments.

	DDBME [16]	Natarajan's [21]	Proposed
Current BAB size	16×16 bits RAM	16×16 bits SRAM	16×16 bits SRAM
SR buffer size	16×32 bits RAM	47×32 bits SRAM	44×44 bits SRAM
Access from SR buff to obtain one MV (Bytes)	4096	5828	Average: 1348 Maximum: 3328
Latency to obtain one MV (cycles)	1039	1039	Average: 360 Maximum: 740
One BAB processing (cycles)	3034 (without pipelinig)	N/A	1708 (without pipelining)

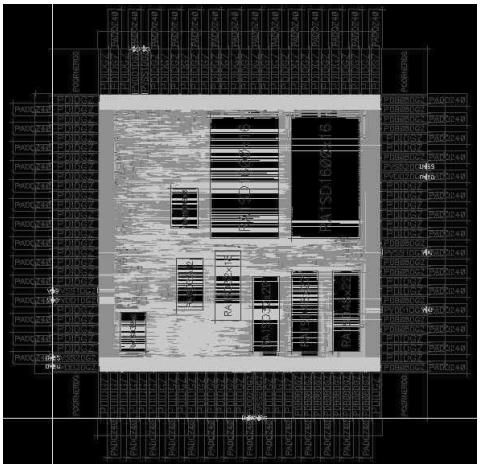
Table 8. Architecture analysis and comparison for various binary motion estimations.

Table 8 also lists the architecture comparisons between the proposed and some previous works in [16] and [21]. In [16] it adopts the data-dispatch technique and is named as data-dispatch based BME (DDBME). [21] is Natarajan's architecture which is modified from BME-based Yang's 1-D systolic array [22]. In their design, they use extra memory, SAP module, to process the bit shifting and bit packing for the alignment of BAB. It also results in a computation overhead. In our design, we have used the Boundary Pixel Detector for the alignment of boundary of BAB. Accordingly, no SAP memory is needed. Furthermore, the proposed CBBME design needs less data transfer and latency to obtain one motion vector compared with [16] and [21], because we consider the skipping on redundant searches. Compared with the implementation for one BAB processing in the worst case, our design also requires less cycles than [16] with the same base of non-pipelining work. Only 56% cycles of [16] is needed in our approach.

Fig. 22(a) shows the synthesized gate count of each module and Fig. 22(b) shows the chip layout using synthesizable Verilog HDL. There are 7 synchronous RAMs in the chip. Two 1600×16 bits RAMs are used for frame buffer. Two 48×22 bits RAMs are used for SR buffer. One 32×20 bits RAM is used for SC buffer, one 32×18 bits RAM is used for MC buffer and one 32×20 bits RAM is used for BAB buffer, respectively. The chip feature is summarized in Table 9. Total gate count is 40765. The chip size is 2.4×2.4 mm^2 with TSMC 0.18μm CMOS technology and the maximum operation frequency is 53 MHz



(a)



(b)

Fig. 22. (a) Synthesized gate count of each module. (b) Chip layout of shape coding encoder.

Technology	**TSMC 0.18μm CMOS (1P6M)**
Package	**128 CQFP**
Die size	**2.4×2.4 mm^2**
Core size	**1.4×1.4 mm^2**
Clock rate	**53 MHz**
Power dissipation	**35mW**
Gate count	**40765**
Memory size (bits)	**Frame buffer: 2×1600×16** **SR buffer: 2×48×22** **SC buffer: 32×20** **MC buffer: 32×18** **BAB buffer: 32×20**

Table 9. Chip Features.

7. Conclusion

MPEG-4 has provided a well-adopted object-based coding technique. When people migrate from compressed coding domain to object coding domain, the complexity issue on shape coding is converged. In this paper we propose a fast binary motion estimation algorithm using diamond search pattern for shape coding and an efficient architecture for MPEG-4 shape coding. By using the properties of shape information and diamond shaped zones, we can reduce the number of search points significantly, resulting in a proportional reduction

of computational complexity. The experimental results show that the proposed method can reduce the number of search points of BME for shape coding to only 0.6% compared with that of the full search method described in MPEG-4 verification model. Specifically, the fast algorithm takes equal bit rate in the same quality compared with full search algorithm. The proposed algorithm is simple, efficient and suitable for real-time software and hardware applications. This architecture is based on the boundary search fast algorithm which accomplishes the large reduction on computation complexity. We also apply the approaches on center-biased motion vector distribution and search range shrinking for further improvement. In this paper we report a comprehensive exploration on each module of shape coding encoder. Our architecture completely elaborates the advantages of the proposed fast algorithm with a high performance and regular architecture. The result shows that our design can reduce the memory access and processing cycles largely. The average number of clock cycles for one binary alpha block processing is only 1708, which is far less than other designs. The system architecture is implemented by synthesizable Verilog HDL with TSMC 0.18μm CMOS technology. The chip size is 2.4 × 2.4 mm^2 and the maximum operation frequency is 53 MHz.

8. Acknowledgements

This work was supported by the CIC and the National Science Council of Taiwan, R.O.C. under Grant *NSC97-2220-E-008-001*.

9. References

B. Natarajan, V. Bhaskaran, and K. Konstantinides, "Low-complexity block-based motion estimation via one-bit trasforms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 702 -707, Aug. 1997.

D. Yu, S. K. Jang, and J. B. Ra, "A fast motion estimation algorithm for MPEG-4 shape coding," *IEEE Int. Conf. Image Processing*, vol. 1, pp. 876-879, 2000.

E. A. Al_Qaralleh, T. S. Chang, and K. B. Lee, "An efficient binary motion estimation algorithm and its architecture for MPEG-4 shape encoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 17, pp. 859-868, Jul. 2006.

G. Feygin, P. Glenn and P. Chow, "Architectural advances in the VLSI implementation of arithmetic coding for binary image compression," *Proc. Data Compression Conf. (DCC'94)*, pp. 254 -263, 1994.

G. Sullivan and T. Wiegand, "Rate-Distortion optimization for video compression", *IEEE Signal Processing Magazine*, pp. 74-90, Nov. 1998.

H. C. Chang, Y. C. Chang, Y. C. Wang, W. M. Chao, and L. G. Chen, "VLSI architecture design for MPEG-4 shape coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, pp. 741-751, Sep. 2002.

ISO/IEC 13818-2, "Information technology-generic coding of moving pictures and associated audio information-Video," 1994.

ISO/IEC JTC1/SC29/WG11 N 2502a, "Generic coding of audio-visual objects: Visual 14492-2," Atlantic City Final Draft IS, Dec. 1998.

ISO/IEC JTC1/SC29/WG11 N 3908, "MPEG-4 video verification model version 18.0," Jan. 2001.

J. L. Mitchell and W. B. Pennebaker, "Optimal hardware and software arithmetic coding procedures for the Q-Coder," *IBM J. Res. Devel.*, vol. 32, pp. 717-726, Nov. 1998.

Jo Yew Tham, Surendra Ranganath, Maitreya Ranganath, and Ashraf Ali Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 169-177, Aug. 1998.

K. B. Lee, J. Y. Lin and C. W. Jen, "A Multisymbol Context-Based Arithmetic Coding Architecture for MPEG-4 Shape Coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, pp. 283-295, Feb. 2005.

K. B. Lee, J. Y. Lin, and C. W. Jen, "A fast dual symbol context-based arithmetic coding for MPEG-4 shape coding," *IEEE International Symposium on Circuits and Systems (ISCAS2004)*, pp. 317-320, 2004.

K. M. Yang, M. T. Sun, and L. Wu, "A family of VLSI design for the motion compensation block-matching algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 36, pp. 1317 -1325, Oct. 1989.

K. Panusopone, and X. Chen, "A fast motion estimation method for MPEG-4 arbitrarily shaped objects," *IEEE Int. Conf. Image Processing*, vol. 3, pp. 624-627, 2000.

L. K. Liu, and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 419-422, Aug. 1996.

M. Tourapis, O. C. Au, and M. L. Liou, "Highly efficient predictive zonal algorithms for fast block-matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, pp. 934-947, Oct. 2002.

N. Brady, "MPEG-4 standardized methods for the compression of arbitrarily shaped video objects," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 1170-1189, Dec. 1999.

Recommendation H.263: Video coding for low bit-rate communication, ITU-T H.263, 1998.

S. Dutta and W. Wolf, "A flexible parallel architecture adapted to block-matching motion estimation algorithms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no 1, pp. 74-86, Feb. 1996.

S. H. Han, S. W. Kwon, T. Y. Lee, and M. K. Lee, "Low power motion estimation algorithm based on temporal correlation and its architecture," *IEEE Internal Symposium on Signal Processing and its Applications (ISSPA)*, vol. 2, pp.647-650, Aug. 2001.

T. H. Tsai and C. P. Chen, "A fast binary motion estimation algorithm for MPEG-4 shape coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, pp. 908-913, Jun. 2004.

T. Komarek and P. Pirsch, "Array architectures for block-matching algorithms," *IEEE Trans. Circuits Syst.*, vol. 36, pp.1301-1308, Oct. 1989.

T. M. Liu, B. J. Shieh and C. Y. Lee, "An efficient modeling codec architecture for binary shape coding," *IEEE International Symposium on Circuits and Systems (ISCAS2002)*, vol. 2, pp. 316 -319, 2002.

4

Memory-Efficient Hardware Architecture of 2-D Dual-Mode Lifting-Based Discrete Wavelet Transform for JPEG2000

Chih-Hsien Hsia and Jen-Shiun Chiang
Department of Electrical Engineering, Tamkang University
Taipei, Taiwan

1. Introduction

Discrete wavelet transform (DWT) has been adopted in a wide range of applications, including speech analysis, numerical analysis, signal analysis, image coding, pattern recognition, computer vision, and biometrics (Mallat, 1989). It can be considered as a multi-resolution decomposition of a signal into several components with different frequency bands. Moreover, DWT is a powerful tool for signal processing applications, such as JPEG2000 still image compression, denoising, region of interest, and watermarking. For real-time processing it needs small memory access and low computational complexity. Implementations of two-dimensional (2-D) DWT can be classified as convolution-based operation (Mallat, 1989) (Marino, 2000) (Vishwanath et al., 1995) (Wu. & Chen, 2001) and lifting-based operation (Sweldens, 1996). Since the convolution-based implementations of DWT have high computational complexity and large memory requirements, lifting-based DWT has been presented to overcome these drawbacks (Sweldens, 1996) (Daubechies & Sweldens, 1998). The lifting-based scheme can provide low-complexity solutions for image/video compression applications, such as JPEG2000 (Lian et al., 2001), Motion-JPEG2000 (Seo & Kim, 2007), MPEG-4 still image coding, and MC-EZBC (Ohm, 2005) (Chen & Woods, 2004). However, the real-time 2-D DWT for multimedia application is still difficult to achieve. Hereafter, efficient transformation schemes for real-time application are highly demanded. Performing 2-D (or multi-dimensional) DWT requires many computations and a large block of transpose memory for storing intermediate signals with long latency time. This work presents new algorithms and hardware architectures to improve the critical issues in 2-D dual-mode (supporting 5/3 lossless and 9/7 lossy coding) lifting-based discrete wavelet transform (LDWT). The proposed 2-D dual-mode LDWT architecture has the merits of low transpose memory, low latency, and regular signal flow, making it suitable for VLSI implementation. The transpose memory requirement of the $N \times N$ 2-D 5/3 mode LDWT is $2N$ and that of 2-D 9/7 mode LDWT is $4N$.

Low transpose memory requirement is of a priority concern in spatial-frequency domain implementation. Generally, raster scan signal flow operations are popular in $N \times N$ 2-D DWT, and under this approach the memory requirement ranges from $2N$ to N^2 (Diou et al., 2001)

(Andra et al., 2002) (Chen & Wu, 2002) (Chen, 2002) (Chiang & Hsia, 2005) (Jung & Park, 2005) (Vishwanath et al., 1995) (Huang et al., 2005) (Mei et al., 2006) (Huang et al, 2005) (Wu & Lin, 2005) (Lan ., 2005) (Wu. & Chen, 2001) in 2-D 5/3 and 9/7 modes LDWT. In order to reduce the amount of the transpose memory, the memory access must be redirected. In our approach, the signal flow is revised from row-wise only to mixed row- and column-wise, and a new approach, called interlaced read scan algorithm (IRSA), is used to reduce the amount of the transpose memory. By the IRSA approach, a transpose memory size is of 2*N* or 4*N* (5/3 or 9/7 mode) for an *N*×*N* DWT. The proposed 2-D LDWT architecture is based on parallel and pipelined schemes to increase the operation speed. For hardware implementation we replace multipliers with shifters and adders to accomplish high hardware utilization. This 2-D LDWT has the characteristics of high hardware utilization, low memory requirement, and regular signal flow. A 256×256 2-D dual-mode LDWT was designed and simulated by VerilogHDL, and further synthesized by the Synopsys design compiler with TSMC 0.18μm 1P6M CMOS process technology.

2. Survey of 2-D LDWT Architecture

Among the variety of DWT algorithms, LDWT provides a new approach for constructing biorthogonal wavelet transforms and also provides an efficient scheme for calculating classical wavelet transforms (Sweldens, 1996) (Chen & Wu, 2002) (Andra et al., 2000) (Andra et al., 2002) (Diou et al., 2001) (Chen, 2002) (Chiang & Hsia, 2005) (Tan & Arslan, 2001) (Huang et al., 2005) (Huang et al., 2002) (Mei et al., 2006) (Weeks & Bayoumi, 2002) (Varshney et al., 2007) (Huang et al., 2004) (Tan & Arslan, 2003) (Jiang & Ortega, 2001) (Jung & Park, 2005) (Chen, 2004) (Lian et al, 2001) (Seo & Kim, 2007) (Huang et al., 2005) (Wu & Lin, 2005) (Lan et al., 2005) (Wu. & Chen, 2001). Factoring the classical wavelet filter into lifting steps can reduce the computational complexity of the corresponding DWT by up to 50% (Daubechies & Sweldens, 1998). The lifting steps can be implemented easily, which is different from the direct finite impulse response (FIR) implementations of Mallat's algorithm (Daubechies & Sweldens, 1998). Andra *et al.* (Andra et al., 2000) (Andra et al., 2002) proposed a block-based simple four-processor architecture that computes several stages of the DWT at a time. Diou *et al.* (Diou et al., 2001) presented an architecture that performs LDWT with a 5/3 filter by interleaving technique. Chen *et al.* (Chen & Wu, 2002) proposed a folded and pipelined architecture for a 2-D LDWT implementation, with memory size of 2.5*N* for an *N*×*N* 2-D DWT. This lifting architecture for vertical filtering is divided into two parts, each consisting of one adder and one multiplier. Since both parts are activated in different cycles, they can share the same adder and multiplier to increase the hardware utilization and reduce the latency. However, this architecture also has high complexity due to the characteristics of the signal flow. Chen *et al.* (Chen, 2002) proposed a flexible folded architecture for 3-level 1-D LDWT to increase the hardware utilization. Chiang *et al.* (Chiang & Hsia, 2005) proposed a 2-D DWT folded architecture to improve the hardware utilization. Jiang *et al.* (Jiang & Ortega, 2001) presented a parallel processing architecture that models the DWT computation as a finite state machine and efficiently computes the wavelet coefficients near the boundary of each segment of the input signal. Lian *et al.* (Lian et al, 2001) and Chen *et al.* (Chen, 2004) used a 1-D folded architecture to improve the hardware utilization of 5/3 and 9/7 filters. The recursive architecture is a general scheme to implement any wavelet filter that is decomposed into lifting steps in

smaller hardware complexity. Jung *et al.* (Jung & Park, 2005) presented an efficient VLSI architecture of dual-mode LDWT that is used by lossy or lossless compression of JPEG2000. Marino (Marino, 2000) proposed a high-speed/low-power pipelined architecture for the direct 2-D DWT by four-subband transforms performed in parallel. The architecture of (Huang et al., 2002) implements 2-D DWT with only transpose memory by using recursive pyramid algorithm (PRA). In (Vishwanath et al., 1995) it has the average of N^2 computing time for all DWT levels. However, they use many multipliers and adders. Varshney *et al.* (Varshney et al., 2007) presented energy efficient single-processor and fully pipelined architectures for 2-D 5/3 lifting-based JPEG2000. The single processor performs both row-wise and column-wise processing simultaneously to achieve the 2-D transform with 100% hardware utilization. Tan *et al.* (Tan & Arslan, 2003) presented a shift-accumulator arithmetic logic unit architecture for 2-D lifting-based JPEG2000 5/3 DWT. This architecture has an efficient memory organization, which uses a small amount of embedded memory for processing and buffering. Those architectures achieve multi-level decomposition using an interleaving scheme that reduces the size of memory and the number of memory accesses, but have slow throughput rates and inefficient hardware utilization. Seo *et al.* (Seo & Kim, 2007) proposed a processor that can handle any tile size, and supports both 5/3 and 9/7 filters for Motion-JPEG2000. Huang *et al.* (Huang et al, 2005) proposed a generic RAM-based architecture with high efficiency and feasibility for 2-D DWT. Wu *et al.* (Wu & Lin, 2005) presented a high-performance and low-memory architecture to implement a 2-D dual-mode LDWT. The pipelined signal path of their architecture is regular and practical. Lan *et al.* (Lan et al., 2005) proposed a scheme that can process two lines simultaneously by processing two pixels in a clock period. Wu *et al.* (Wu. & Chen, 2001) proposed an efficient VLSI architecture for direct 2-D LDWT, in which the poly-phase decomposition and coefficient folding are adopted to increase the hardware utilization. Despite these efficient improvements to existed architectures, further improvements in the algorithm and architecture are still needed. Some VLSI architectures of 2-D LDWT try to reduce the transpose memory requirements and communication between the processors (Chen & Wu, 2002) (Andra et al., 2000) (Andra et al., 2002) (Diou et al., 2001) (Chen, 2002) (Chiang & Hsia, 2005) (Tan & Aslan, 2002) (Jiang & Ortega, 2001) (Lian et al., 2001) (Jung & Park, 2005) (Chen, 2004) (Huang et al., 2005) (Daubechies & Sweldens, 1998) (Marino, 2000) (Vishwanath et al., 1995) (Taubman & Marcellin, 2001) (Marcellin et al., 2000) (Mei et al., 2006) (Varshney et al., 2007) (Huang et al., 2004) (Tan & Arslan, 2003) (Seo & Kim, 2007) (Huang et al., 2005) (Wu & Lin, 2005) (Lan et al., 2005) (Wu. & Chen, 2001), however these hardware architectures still need large transpose memory.

3. Discrete wavelet transform and lifting-based method

This section briefly reviews the use of DWT in the coding engine of JPEG2000 (Taubman & Marcellin, 2001). The classical DWT employs filtering and convolution to achieve signal decomposition (Mallat, 1989) (Marino, 2000) (Vishwanath et al., 1995) (Wu. & Chen, 2001). Meyer and Mallat found that the orthonormal wavelet decomposition and reconstruction can be implemented in the multi-resolution signal analysis framework (Mallat, 1989). The multi-resolution analysis is now a standard method for constructing the orthonormal wavelet-bases. JPEG2000 adopts this characteristic to transform an image in the spatial domain into the frequency domain.

3.1 Classical DWT

DWT performs multi-resolution decomposition of the input signals (Mallat, 1989). The original signals are first decomposed into two subspaces, called the low- (low-pass) and high-frequency (high-pass) subbands. The classical DWT implements the decomposition (analysis) of a signal by a low-pass digital filter H and a high-pass digital filter G. Both digital filters are derived using the scaling function and the corresponding wavelets. The system downsamples the signal to decimate half of the filtered results in decomposition processing. The Z-transfer functions of $H(z)$ and $G(z)$ based on four-tap and non-recursive FIR filters with length L are represented as follows:

$$H(z)= h_0 + h_1z^{-1} + h_2z^{-2}+ h_3z^{-3}, \tag{1}$$

$$G(z)= g_0 + g_1z^{-1} + g_2z^{-2}+ g_3z^{-3}. \tag{2}$$

The reconstruction (synthesis) process is implemented using an up-sampling process. Mallat's tree algorithm or pyramid algorithm (Mallat, 1989) can be used to find the multi-resolution decomposition DWT. The decomposition DWT coefficients at each resolution level can be calculated as follows:

for (j=1 to J)
for (i=0 to $N/2^i$-1)
{

$$X_H^j(n)=\sum_{i=0}^{k-1} G(z)X^{J-1}H(2n-i) \tag{3}$$

$$X_L^j(n)=\sum_{i=0}^{k-1} H(z)X^{J-1}G(2n-i) \tag{4}$$

}

where j denotes the current resolution level, k the number of the filter tap, $X_H^j(n)$ the nth high-pass DWT coefficient at the jth level, $X_L^j(n)$ the nth low pass DWT coefficient at the jth level, and N the length of the original input sequence. Fig. 1 shows a 3-level 1-D DWT decomposition using Mallat's algorithm.

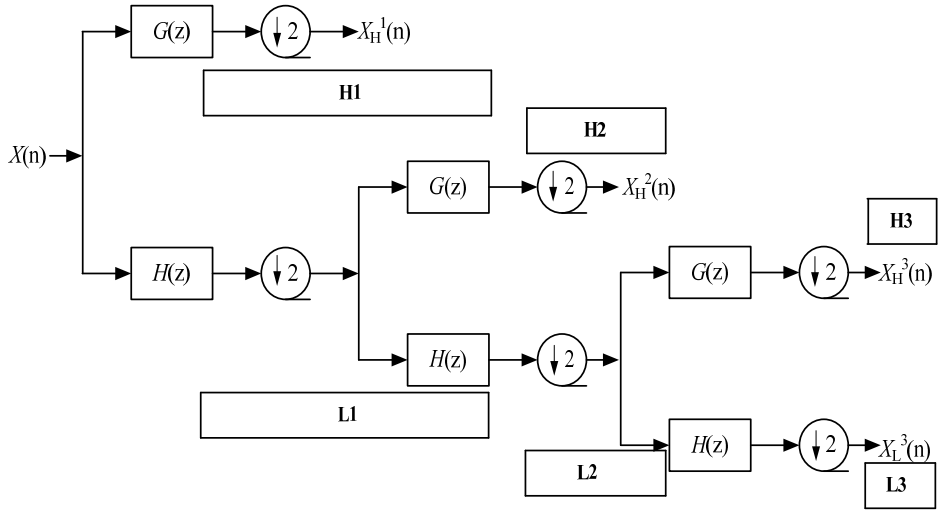


Fig. 1. 3-level 1-D DWT decomposition using Mallat's algorithm.

The downsampling operation is then applied to the filtered results. A pair of filters are applied to the signal to decompose the image into the low-low (LL), low-high (LH), high-low (HL), and high-high (HH) wavelet frequency bands. Fig. 2 illustrates the basic 2-D DWT operation and the transformed result which is composed of two cascading 1-D DWTs. The image is first analyzed horizontally to generate two subimages. The information is then sent into the second 1-D DWT to perform the vertical analysis to generate four subbands, and each with a quarter of the size of the original image. Considering an image of size $N \times N$, each band is subsampled by a factor of two, so that each wavelet frequency band contains $N/2 \times N/2$ samples. The four subbands can be integrated to generate an output image with the same number of samples as the original one.

Most image compression applications can reapply the above 2-D wavelet decomposition repeatedly to the LL subimage, each time forming four new subband images, to minimize the energy in the lower frequency bands.

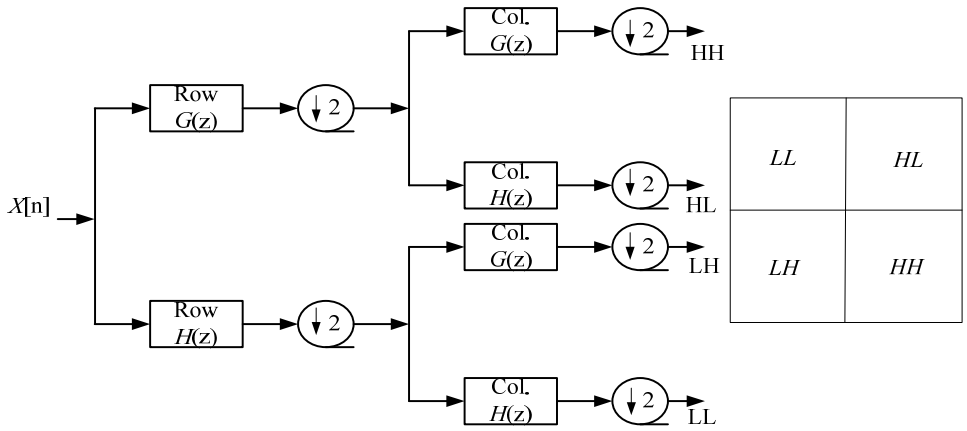


Fig. 2. The 2-D analysis DWT image decomposition process.

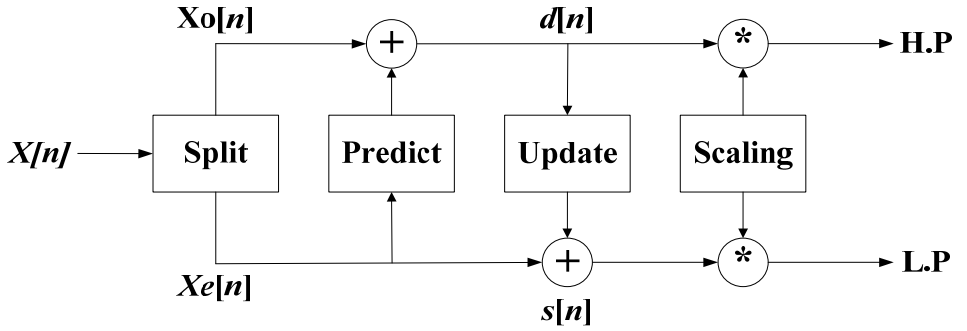


Fig. 3. Block diagram of the LDWT.

3.2 LDWT algorithm

The lifting-based scheme proposed by Daubechies and Sweldens requires fewer computations than the traditional convolution-based approach (Sweldens, 1996) (Daubechies & Sweldens, 1998). The lifting-based scheme is an efficient implementation for DWT; it can easily use integer operations and avoid the problems caused by the finite precision or rounding. The Euclidean algorithm can be used to factorize the poly-phase matrix of a DWT filter into a sequence of alternating upper and lower triangular matrices and a diagonal matrix. The variables $h(z)$ and $g(z)$ in (5) respectively denote the low- and high-pass analysis filters, which can be divided into even and odd parts to generate a poly-phase matrix $P(z)$ as in (6).

$$g(z)=g_e(z^2)+ z^{-1}g_o(z^2),$$

$$h(z)=h_e(z^2)+z^{-1}h_o(z^2). \quad (5)$$

$$P(z) = \begin{bmatrix} h_e(z) & g_e(z) \\ h_o(z) & g_o(z) \end{bmatrix}. \tag{6}$$

The Euclidean algorithm recursively finds the greatest common divisors of the even and odd parts of the original filters. Since h(z) and g(z) form a complementary filter pair, P(z) can be factorized into (7):

$$P(z) = \prod_{i=1}^{m} \begin{pmatrix} 1 & s_i(z) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ t_i(z) & 1 \end{pmatrix} \begin{pmatrix} k & 0 \\ 0 & 1/k \end{pmatrix}. \tag{7}$$

where s_i(z) and t_i(z) are Laurent polynomials corresponding to the prediction and update steps, respectively, and k is a nonzero constant. Therefore, the filter bank can be factorized into three lifting steps.

As illustrated in Fig. 3, a lifting-based scheme has the following four stages:

1) Split phase: The original signal is divided into two disjoint subsets. Significantly, the variable *Xe* denotes the set of even samples and *Xo* denotes the set of odd samples. This phase is also called lazy wavelet transform because it does not decorrelate the data but only subsamples the signal into even and odd samples.

2) Predict phase: The predicting operator P is applied to the subset *Xo* to obtain the wavelet coefficients d[n] as in (8).

$$d[n]=Xo[n]+P\times(Xe[n]). \tag{8}$$

3) Update phase: *Xe*[n] and d[n] are combined to obtain the scaling coefficients s[n] after an update operator U as in (9).

$$s[n]=Xe[n]+U\times(d[n]). \tag{9}$$

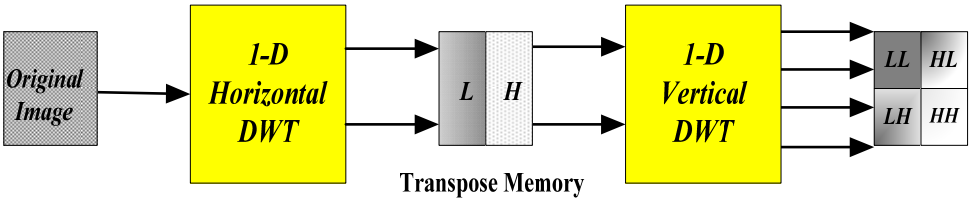
4) Scaling: In the final step, the normalization factor is applied on s[n] and d[n] to obtain the wavelet coefficients. For example, (10) and (11) describe the implementation of the 5/3 integer lifting analysis DWT and are used to calculate the odd (high-pass) and even coefficients (low-pass), respectively.

$$d^*[n] = X(2n+1) - \lfloor X(2n) + X(2n+2)/2 \rfloor. \tag{10}$$

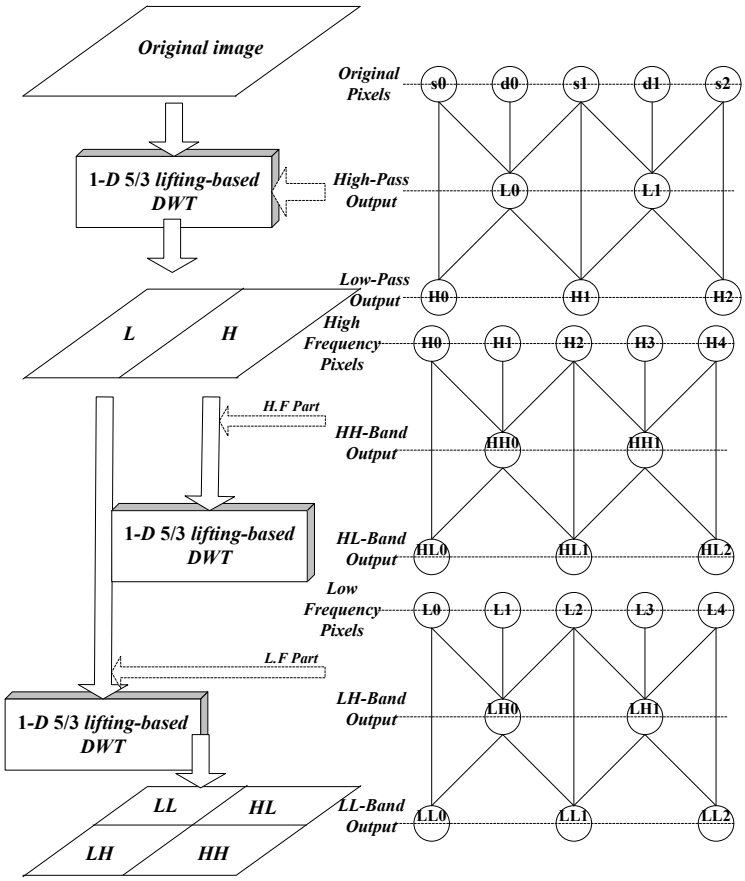
$$s^*[n] = X(2n) - \lfloor d(2n-1) + d(2n+1) + 2/4 \rfloor. \tag{11}$$

Although the lifting-based scheme has low complexity, its long and irregular signal paths cause the major limitation for efficient hardware implementations. Additionally, the increasing number of pipelined registers increases the internal memory size of the 2-D DWT architecture. The 2-D LDWT uses a vertical 1-D LDWT subband decomposition and a horizontal 1-D LDWT subband decomposition to find the 2-D LDWT coefficients. Therefore, the memory requirement dominates the hardware cost and architectural complexity of 2-D LDWT. Fig. 4 shows the 5/3 mode 2-D LDWT operation. The default wavelet filters in JPEG2000 are dual-mode (5/3 and 9/7 modes) LDWT (Taubman & Marcellin, 2001). The lifting-based steps associated with the dual-mode wavelets are shown in Figs. 5 and 6,

respectively. Assuming that the original signals are infinite in length, the first lifting stage is first applied to perform the DWT.



(a)



(b)

Fig. 4. 5/3 mode 2-D LDWT operation. (a) The block diagram flow of a traditional 2-D DWT. (b) Detailed processing flow.

Fig. 5 shows the lifting-based step associated with the wavelet algorithm. The original signals including $s0$, $d0$, $s1$, $d1$, $s2$, $d2$, … are the input pixel sequences. If the original signals are infinite in length, then the first-stage lifting is applied to update the odd index data $s0$, $s1$, …. In (12), the parameters –1/2 and H_i denote the first stage lifting parameter and outcome, respectively. Equation (12) shows the operation of the 5/3 integer LDWT (Wu & Lin, 2005) (Martina & Masera, 2007) (Hsia & Chiang, 2008).

$$H_i = [(S_i+S_{i+1})\times\alpha+d_i]\times K_0,$$

$$L_i = [(H_i+H_{i-1})\times\beta+S_i]\times K_1, \tag{12}$$

where α= –1/2, β=1/4, and K_0= K_1= 1.

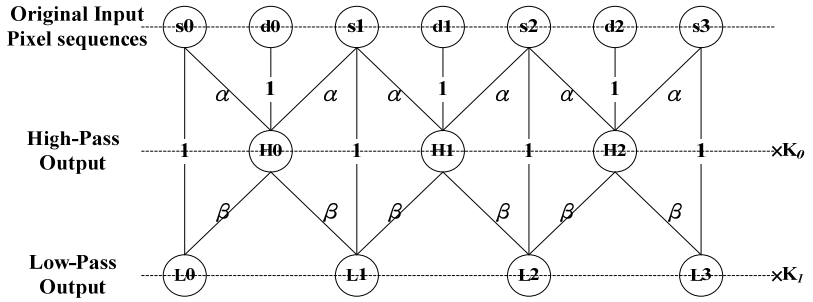


Fig. 5. 5/3 LDWT algorithm.

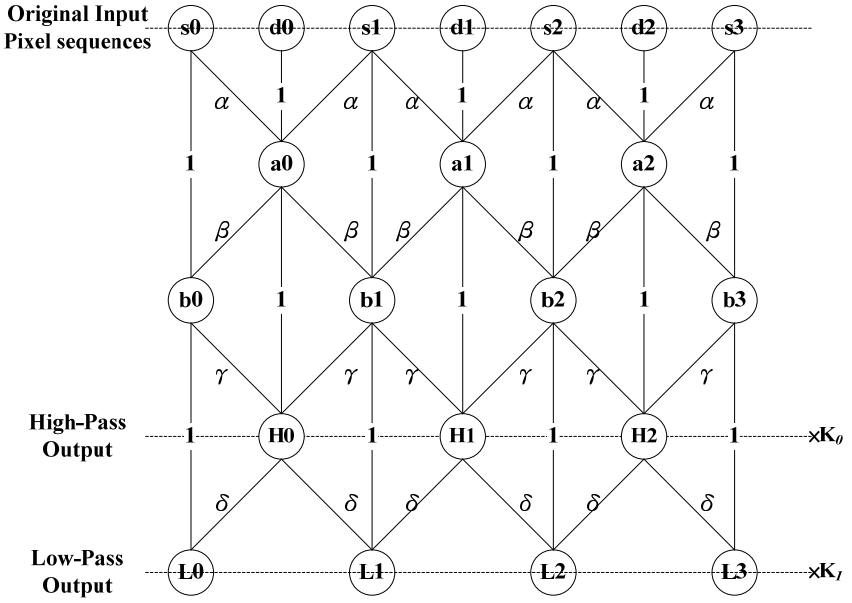


Fig. 6. 9/7 LDWT algorithm.

Together with the high-frequency lifting parameter, α, and the input signal we can find the first stage high-frequency wavelet coefficient, H_i. After H_i is found, H_i together with the low-frequency parameter, β, and the input signals of the second stage low-frequency wavelet coefficients, L_i, can be found. The third and fourth stages lifting can be found in a similar manner.
Similar to the 1-level 1-D 5/3 mode LDWT, the calculation of a 1-level 1-D 9/7 mode LDWT is shown in (13).

$$\begin{aligned} a_i &= [(S_i+S_{i+1})\times\alpha+d_i], \\ b_i &= [(a_i+a_{i-1})\times\beta+S_i], \\ H_i &= [(b_i+b_{i+1})\times\gamma+a_i]\times K_{0,} \\ L_i &= [(H_i+H_{i-1})\times\delta+b_i]\times K_{1.} \end{aligned} \tag{13}$$

where α= −1.586134142, β= −0.052980118, γ= +0.882911075, δ= +0.443506852, K_0= 1.230174104, and K_1= 1/K_0.
The calculation comprises four lifting steps and two scaling steps.

3.3 Boundary extension treatment for LDWT

The finite-length signal processed by DWT leads to the boundary effect. The JPEG2000 standard enhances the symmetric extension pixel at the edge, as shown in Table 1. An appropriate signal extension is required to maintain the same number of the wavelet coefficients as in the original signal. The embedded signal extension algorithm (Tan & Arslan, 2001) can be used to compute the boundary of the image.
Since both the extended signal and the lifting structure are symmetrical, all the intermediate and final results of the lifting-based DWT are also symmetrical with regard to the boundary points, and the boundary extension can be performed without additional computational complexity. The inverse lifting structure is easily derived from Table 1 (Tan & Arslan, 2001). The boundary extension signal reflects the signal to i_{left} samples on the left and i_{right} samples on right. Table 1 shows the extension parameters i_{left} and i_{right} for the reversible transform 5/3 mode and the irreversible transform 9/7 mode.

i_0	i_{left}(5/3)	i_1	i_{right}(5/3)	i_0	i_{left}(9/7)	i_1	i_{right}(9/7)
even	2	odd	1	even	4	odd	3

* i_0: first sample index; i_1: last sample index.
Table 1. Boundary extension to the left and to the right for JPEG2000.

4. Interlaced read scan algorithm (IRSA)

In recent years, many 2-D LDWT architectures have been proposed to meet the requirements of on-chip memory for real-time processing. However, the hardware utilization of these architectures needs to be further improved. In DWT implementation, a 1-D DWT needs very massive computation and therefore the computation unit takes most of the hardware cost (Chen & Wu, 2002) (Andra et al., 2000) (Andra et al., 2002) (Diou et al., 2001) (Chen, 2002) (Chiang & Hsia, 2005) (Chen, 2004) (Huang et al., 2005) (Daubechies & Sweldens, 1998) (Marino, 2000) (Vishwanath et al., 1995) (Taubman & Marcellin, 2001)

(Varshney et al., 2007) (Huang et al., 2004) (Tan & Arslan, 2003) (Seo & Kim, 2007) (Huang et al., 2005) (Wu & Lin, 2005) (Lan et al., 2005) (Wu. & Chen, 2001). A 2-D DWT is composed of two 1-D DWTs and a block of transpose memory. In the conventional approach, the size of the transpose memory is equal to the size of the processed image signal. Fig. 7(a) shows the concept of the proposed dual-mode LDWT architecture, which consists of signal arrangement unit, processing element, memory unit, and control unit, as shown in Fig. 7(b). The outputs are fed to the 2-D LDWT four-subband coefficients, HH, HL, LH, and LL. The proposed architecture is described in detail in this section, and we focus on the 2-D dual-mode LDWT.

Compared to the computation unit, the transpose memory becomes the main overhead in the 2-D DWT. The block diagram of a conventional 2-D DWT is shown in Fig. 4. Without loss of generality, the 2-D 5/3 mode LDWT is considered for the description of the 2-D LDWT. If the image dimension is $N \times N$, during the transformation we need a large block of transpose memory (order of N^2) to store the DWT coefficients after the computation of the first stage 1-D DWT decomposition. The second stage 1-D DWT then uses the stored data to compute the 2-D DWT coefficients of the four subbands (Chen & Wu, 2002) (Andra et al., 2000) (Andra et al., 2002) (Diou et al., 2001) (Chen, 2002) (Varshney et al., 2007) (Huang et al., 2004) (Tan & Arslan, 2003) (Seo & Kim, 2007) (Huang et al., 2005) (Wu & Lin, 2005) (Lan et al., 2005) (Wu. & Chen, 2001). The computation and the access of the memory may take time and therefore the latency is long. Since the memory size of N^2 is a large quantity, here we try to use the approach, interlaced read scan algorithm (IRSA), to reduce the required transpose memory to an order of $2N$ or $4N$ (5/3 or 9/7 mode).

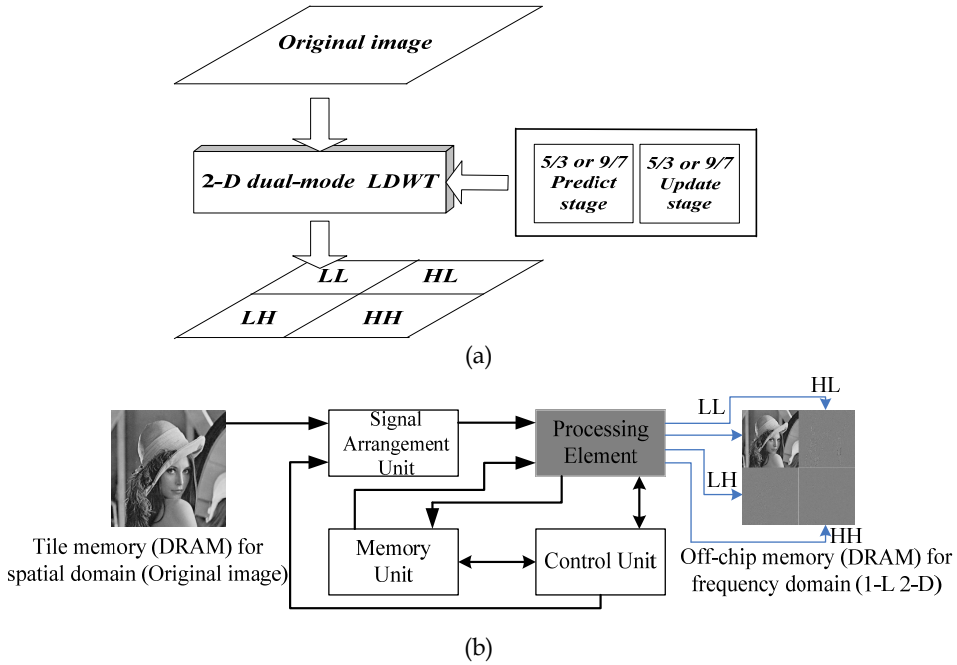
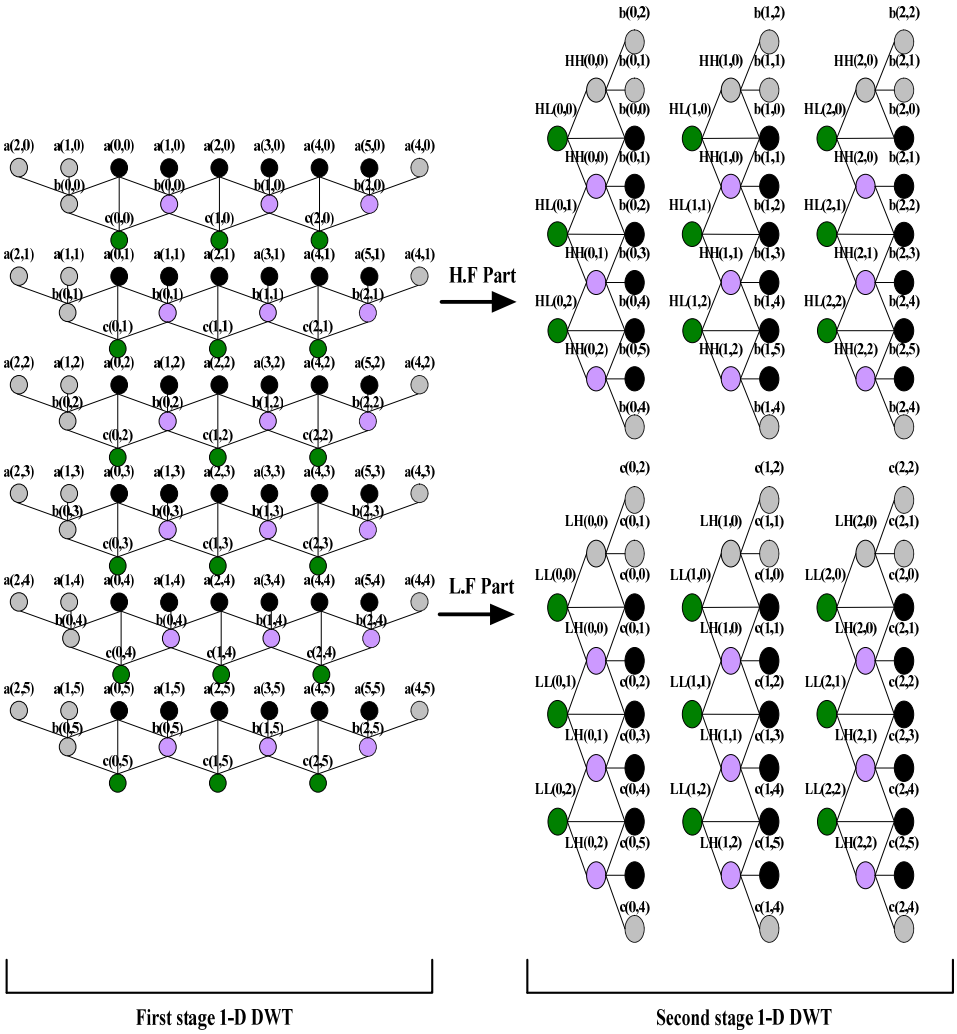


Fig. 7. The system block diagram of the proposed 2-D DWT. (a) 2-D dual-mode LDWT. (b) Block diagram of the proposed system architecture.



$x(i,j)$: original image, i = 0~5 and j = 0~5
$b(i,j)$: high frequency wavelet coefficient of 1-D LDWT
$c(i,j)$: low frequency wavelet coefficient of 1-D LDWT
HH: high-high frequency wavelet coefficient of 2-D LDWT
HL: high-low frequency wavelet coefficient of 2-D LDWT
LH: low-high frequency wavelet coefficient of 2-D LDWT
LL: low-low frequency wavelet coefficient of 2-D LDWT
Fig. 8. Example of 2-D 5/3 mode LDWT operations.

Without loss of generality, let us take a 6×6-pixel image to describe the 2-D 5/3 mode LDWT operation and IRSA. Fig. 8 shows the operation diagram of the 2-D 5/3 mode LDWT operations of a 6×6 image. In Fig. 8, $x(i,j)$, i = 0 to 5 and j = 0 to 5, represents the original

image signal. The left most two columns are the left boundary extension columns, and the right most column is the right boundary extension column. The details of the boundary extension were described in the previous section. The left half of Fig. 8 shows the first stage 1-D DWT operations. The right half of Fig. 8 shows the second stage 1-D DWT operations for finding the four subband coefficients, HH, HL, LH, and LL. In the first stage 1-D DWT, three pixels are used to find a 1-D high-frequency coefficient. For example, $x(0,0)$, $x(0,1)$, and $x(0,2)$ are used to find the high-frequency coefficient $b(0,0)$, $b(0,0) = -[x(0,0) + x(0,2)]/2 + x(0,1)$. To calculate the next high-frequency coefficient $b(0,1)$, we need pixels $x(0,2)$, $x(0,3)$, and $x(0,4)$. Here $x(0,2)$ is used to calculate both $b(0,0)$ and $b(0,1)$ and is called the overlapped pixel. The low-frequency coefficient is calculated using two consecutive high-frequency coefficients and the overlapped pixel. For example, $b(0,0)$ and $b(0,1)$ are together with $x(0,2)$ to find the low-frequency coefficient $c(0,1)$, $c(0,1) = [b(0,0) + b(0,1)]/4 + x(0,2)$. The calculated high-frequency coefficients, $b(i,j)$, and low-frequency coefficients, $c(i,j)$, are then used in the second stage 1-D DWT to calculate the four subband coefficients, HH, HL, LH, and LL.

In the second stage 1-D DWT of Fig. 8, the first HH coefficient, HH(0,0), is calculated by using $b(0,2)$, $b(0,1)$, and $b(0,0)$, $HH(0,0) = -[b(0,0) + b(0,2)]/2 + b(0,1)$. The other HH coefficients can be computed in the same manner using three column consecutive $b(i,j)$ signals. For two column consecutive HH coefficients it has an overlapped $b(i,j)$ signal. For example $b(0,3)$ is the overlapped signal for computing HH(0,0) and HH(0,1). To compute HL coefficients, it needs two column consecutive HH coefficients and an overlapped $b(i,j)$ signal. For example, HL(0,1) is computed from HH(0,0), HH(0,1), and $b(0,3)$, $HL(0,1) = [HH(0,0) + HH(0,1)]/4 + b(0,3)$. The LH coefficients are computed from the $c(i,j)$ signal, and each LH coefficient needs the calculation of three $c(i,j)$ signals. For example, LH(0,1) is computed from $c(0,2)$, $c(0,3)$, and $c(0,4)$, $LH(0,1) = -[c(0,2) + c(0,4)]/2 + c(0,3)$. For two column consecutive LH coefficients it has an overlapped $c(i,j)$ signal. For example, $c(0,3)$ is the overlapped signal for computing LH(0,0) and LH(0,1). To compute LL coefficients, it needs two column consecutive LH coefficients and an overlapped $c(i,j)$ signal. For example, LL(0,1) is computed from LH(0,0), LH(0,1), and $c(0,2)$, $LL(0,1) = [LH(0,0) + LH(0,1)]/4 + c(0,2)$. The detail calculation equations for the four subband coefficients are summarized in the following equations:

$$HH(h,v) = x(2h+1,2v+1)+(1/4)\sum_{s=0}^{1}\sum_{t=0}^{1} x(2h+2s,2v+2t)+(-1/2)\sum_{s=-1}^{2} x(2h+|s|,2v+|-1+s|). \quad (14)$$

$$\begin{aligned} HL(h,v) &= (1/4)[HH(h,v-1)+HH(h,v)]+b(h,2v) \\ &= (1/4)[HH(h,v-1)+HH(h,v)]+(-1/2)[x(2h,2v)+x(2h+2,2v)]+x(2h+1,2v). \end{aligned} \quad (15)$$

$$LH(h,v) = (1/4)[HH(h-1,v)+HH(h,v)]+(-1/2)[x(2h,2v)+x(2h,2v+2)]+x(2h,2v+1). \quad (16)$$

$$\begin{aligned} LL(h,v) &= (1/4)[LH(h,v-1)+LH(h,v)]+c(h,2v) \\ &= (1/4)[LH(h,v-1)+LH(h,v)]+(1/4)[b(h-1,2v)+b(h,2v)]+x(2h,2v) \\ &= (1/4)[LH(h,v-1)+LH(h,v)]+(1/4)[(-1/2)x(2h-2,2v)+x(2h-1,2v)+(-1)x(2h,2v)+x(2h+1,2v)+(-1/2)x(2h+2,2v)]+x(2h,2v). \end{aligned} \quad (17)$$

The parameters in the above equations are defined as follows:

h: horizontal row, *v*: vertical column, *x*: original image, b: high-frequency wavelet coefficient of 1-D DWT, c: low-frequency wavelet coefficient of 1-D DWT, –1/2: Prediction parameter, and 1/4: Updating parameter.

From the description of the operations of the 2-D 5/3 mode LDWT we find that each 1-D high-frequency coefficient, $b(i,j)$, is calculated from three image signals, and one of the image signal is overlapped with the previous $b(i,j)$. The 1-D low-frequency coefficient, $c(i,j)$, is calculated from two row consecutive $b(i,j)$'s and an overlapped pixel. The HH, HL, LH, and LL coefficients are computed from $b(i,j)$'s and $c(i,j)$'s. If we can change the scanning order of the first stage 1-D LDWT and the output order of the second stage 1-D LDWT, during the 2-D LDWT operation we need only to store the $b(i,j)$'s to the transpose memory (First-In First-Out, FIFO size of N) and the overlapped pixels to the internal memory (R4+R9 size of N). For an $N \times N$ image, the transpose memory block can be reduced to only a size of $2N$ as shown in Fig. 9. IRSA is based on this idea, and it can reduce the requirement of the transpose memory significantly. The block diagram of IRSA with several pixels of an image is shown in Fig. 9. In Fig. 9, the numbers on top and left represent the coordinate indexes of a 2-D image. In order to increase the operation speed, IRSA scans two pixels in the consecutive rows a time. IN1 (Initial in $x(0,0)$) and IN2 (Initial in $x(1,0)$) are the scanning inputs at beginning. At the first clock, the system scans two pixels, $x(0,0)$ and $x(1,0)$, from IN1 and IN2, respectively. At the second clock, IN1 and IN2 read pixels $x(0,1)$ and $x(1,1)$, respectively. At clock 3, IN1 and IN2 read pixels $x(0,2)$ and $x(1,2)$, respectively. After IN1 and IN2 have read three pixels, the DWT processor tries to compute two 1-D high-frequency coefficients, $b(0,0)$ and $b(0,1)$, and these two high-frequency coefficients are stored in the transpose memory for the subsequent computation of the low-frequency coefficients. Pixels $x(0,2)$ and $x(1,2)$ are stored in the internal memory for the subsequent computation of the 1-D high-frequency coefficients.

At clock 4, the DWT processor scans pixels on row 2 and row 3, and IN1 and IN2 read pixels $x(2,0)$ and $x(3,0)$, respectively. At clock 5, IN1 and IN2 read pixels $x(2,1)$ and $x(3,1)$, respectively. At clock 6, IN1 and IN2 read pixels $x(2,2)$ and $x(3,2)$, respectively. At this moment the DWT processor tries to compute the two high-frequency coefficients, $b(2,0)$ and $b(3,0)$, upon pixels $x(2,0)$ to $x(2,2)$ and $x(3,0)$ to $x(3,2)$ respectively and these two high-frequency coefficients are stored in the transpose memory for the subsequent computation of the low-frequency coefficients. Pixels $x(2,2)$ and $x(3,2)$ are stored in the internal memory for the subsequent computation of the high-frequency coefficients. Then (at clock 7) the DWT processor scans the subsequent two rows to read three consecutive pixels in each row and compute the high-frequency coefficients. The coefficients are stored in the transpose memory and pixels $x(4,2)$ and $x(5,2)$ are stored in the internal memory. This procedure will continue to read three pixels and compute the high-frequency coefficients and store the coefficients to the transpose memory and store pixels $x(2,j)$ and $x(2,j+1)$ to the internal memory in each row until the last row.

Then the DWT processor scans row 0 and row 1 and makes N1 and N2 read pixels $x(0,3)$ and $x(1,3)$, respectively. At the next clock, IN1 and IN2 read pixels $x(0,4)$ and $x(1,4)$, respectively. The DWT processor uses pixels $x(0,3)$, $x(0,4)$, and $x(0,2)$, which was stored previously, to compute the high-frequency coefficient $b(0,1)$. Simultaneously the DWT processor uses pixels $x(1,3)$, $x(1,4)$, and $x(1,2)$, which was stored previously, to compute the high-frequency coefficients $b(1,1)$. As soon as $b(0,1)$ and $b(1,1)$ are found, $b(0,0)$, $b(0,1)$, and $x(0,2)$ are used to generate the low-frequency coefficient $c(0,1)$; $b(1,0)$, $b(1,1)$, and $x(1,2)$ are

used to generate the low-frequency coefficient $c(1,1)$. The computed high-frequency coefficients are then stored in the transpose memory, and pixels $x(0,4)$ and $x(1,4)$ replace pixels $x(0,2)$ and $x(1,2)$ to be stored in the internal memory. IN1 and IN2 then read the data in rows 2 and 3 to process the same operations until the end of the last pixel. The detail operations are shown in Fig. 10.

The second stage 1-D DWT works in the similar manner as the first stage 1-D DWT. In the HH and HL operations, when three column consecutive $b(i,j)$'s are found in the first stage 1-D DWT, an HH coefficient can be computed. As soon as two column consecutive HH coefficients are found, the two HH coefficients and the overlapped $b(i,j)$'s can be combined to compute an HL coefficient. Similarly, when three column consecutive $c(i,j)$'s are found in the first stage 1-D DWT, an LH coefficient can be computed. As soon as two LH coefficients are found, the two LH coefficients and the overlapped $c(i,j)$ are used to compute an LL coefficient. The detailed operations for the second stage 1-D DWT are shown in Fig. 11.

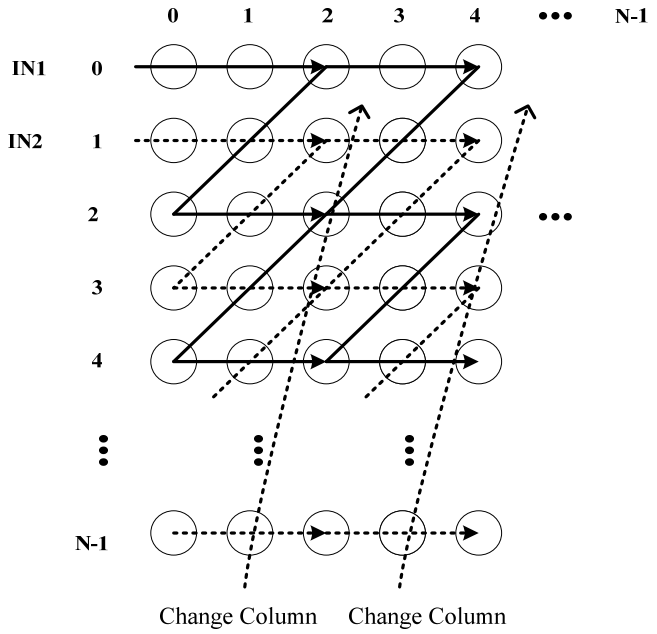


Fig. 9. IRSA of the 2-D LDWT.

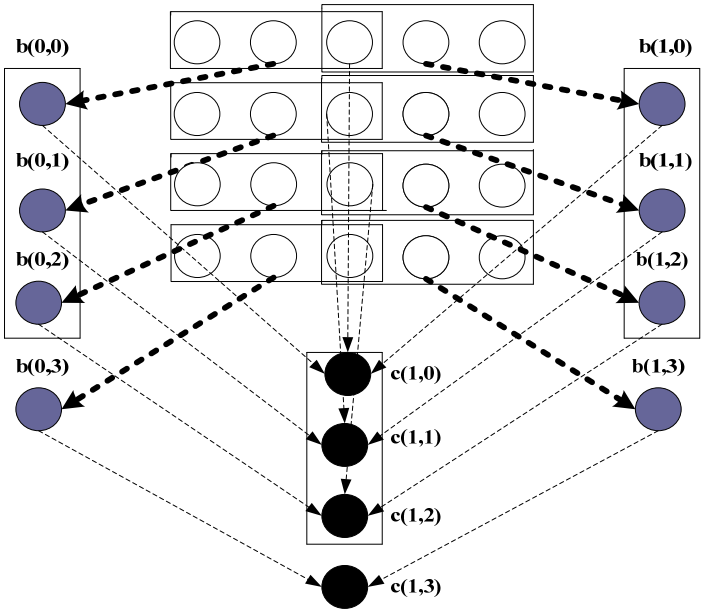
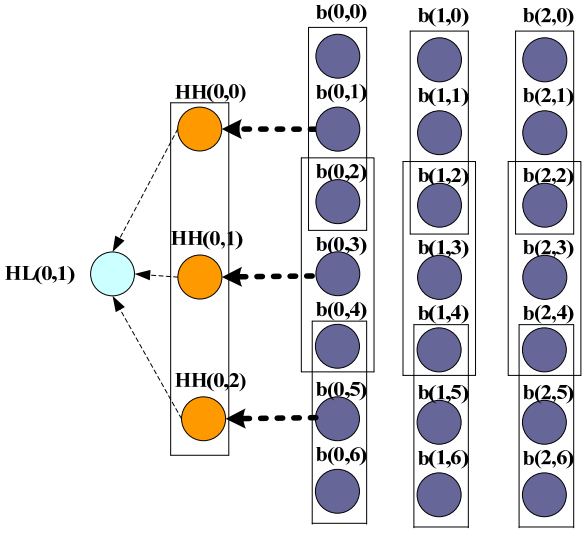


Fig. 10. The detail operations of the first stage 1-D DWT.



(a)

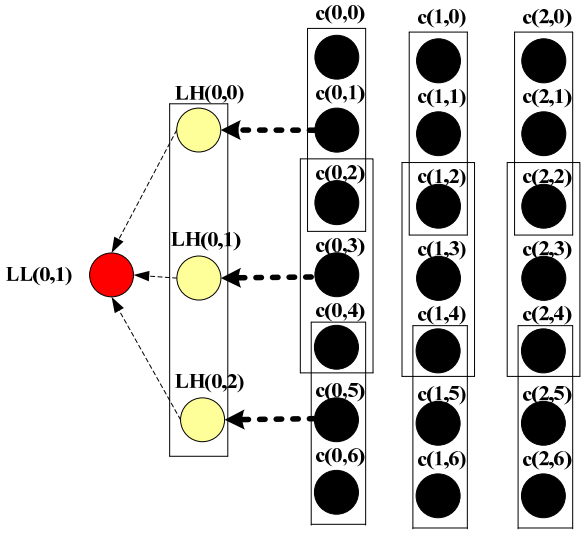


Fig. 11. The detailed operations of the second stage 1-D DWT. (a) The HF (HH and HL) part operations. (b) The LF (LH and LL) part operations.

5. VLSI architecture and implementation for the 2-D dual-mode LDWT

The IRSA approach has been discussed in the previous section, and the architecture of IRSA is described in this section. We can manipulate the control unit to read off-chip memory. In IRSA, two pixels are scanned concurrently, and the system needs two processing units. For the 2-D LDWT processing, the pixels are processed by the first stage 1-D DWT first. The outputs are then fed to the second stage 1-D DWT to find the four subband coefficients, HH, HL, LH, and LL. There are two parts in the architecture, the first stage 1-D DWT and the second stage 1-D DWT. Here we concentrate on the 2-D 5/3 mode LDWT.

5.1 The first stage 1-D LDWT

The first stage 1-D LDWT architecture consists of the following units: signal arrangement unit, multiplication and accumulation cell (MAC), multiplexer (MUX), and FIFO register. The block diagram is shown in Fig. 12.

The signal arrangement unit consists of three registers, R1, R2, and R3. The pixels are input to R1 first, and subsequently the content of R1 is transferred to R2 and then R3, and R1 keeps reading the following pixels. The operation is like a shift register. As soon as R1, R2, and R3 get signal data, MAC starts operating. The signal arrangement unit is shown in Fig. 13. In Fig. 13 MAC operates at the clock with gray circles.

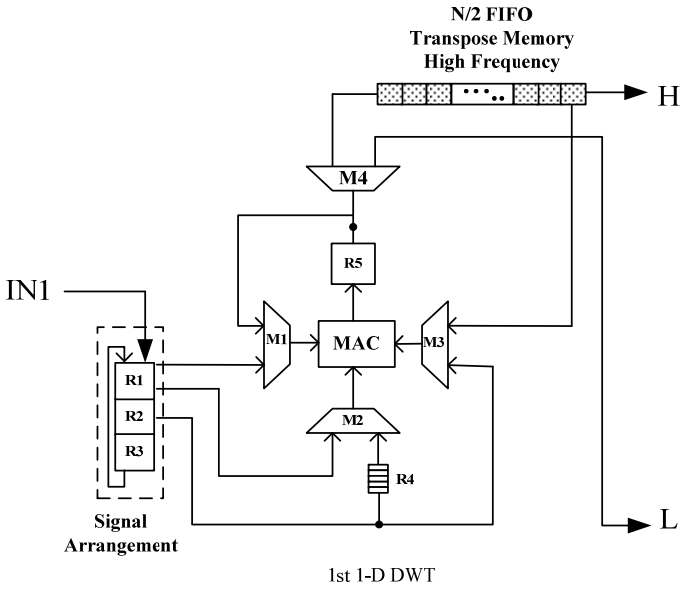


Fig. 12. The architecture of the first stage 1-D DWT.

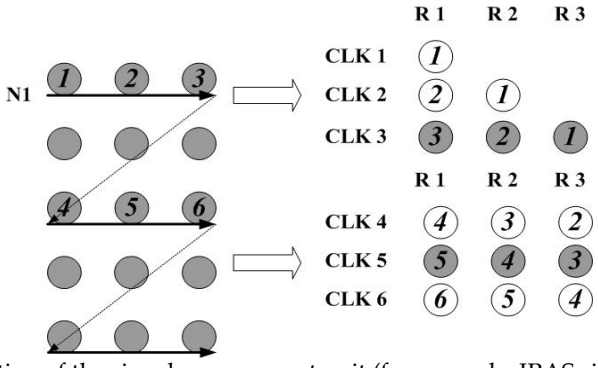


Fig. 13. The operation of the signal arrangement unit (for example, IRAS signal in N1).

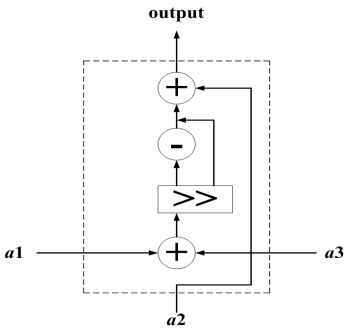


Fig. 14. The block diagram of MAC.

For the low-frequency coefficients calculation we need two high-frequency coefficients and an original pixel. Internal register R4 is used to store the original even pixel (N1) and internal register R9 is used to store the original odd pixel (N2). We can simply shift the content of R3 to R4 after the MAC operation. The FIFO is used to store the high-frequency coefficients to calculate the low-frequency coefficients. Register R5 has two functions: 1) to store the high-frequency coefficients for the low-frequency coefficient calculation, 2) to be used as a signal buffer for MAC. MAC needs time to compute the signal, and the output of MAC cannot directly feed the result to the output or the following operation may be incorrect due to the synchronization problems. R5 acts as an output buffer for MAC to prevent the error in the following operations. In the 5/3 integer lifting-based operations, MAC is used to find the results of the high-frequency output, $-(a1+a3)/2 + a2$, and the low frequency output, $(a1+a3)/4+a2$. There are two multiplication coefficients, $-1/2$ and $1/4$. To save hardware, we can use shifters to implement the $-1/2$ and $1/4$ multiplications. Therefore the MAC needs adders, complementer, and shifters. The MAC block diagram is shown in Fig. 14, where $a1$, $a2$, and $a3$ are the inputs, ''–'' the 2's complement converter, and ''>>'' the right shifter.

5.2 The second stage 1-D LDWT

Similar to the first stage 1-D DWT, the second stage 1-D DWT consists of the following units: signal arrangement unit, MAC, and MUX, as shown in Fig. 15. Due to the parallel architecture, two outputs are generated concurrently from the first stage 1-D DWT, and these two outputs must be merged in the second stage 1-D DWT. The signal arrangement unit processes the signal merging; Fig. 16 shows the processing diagram of the signal arrangement unit. At beginning, signals H0 and H1 are from IN1 and IN2 and these two signals are stored in R3 and R4, respectively. At the next clock, H0 and H1 are moved to R1 and R2 respectively, and concurrently new signals H3 and H4 from IN1 and IN2 are stored to R3 and R4 respectively. The signal arrangement unit operates repeatedly to input signals for the second stage 1-D DWT.

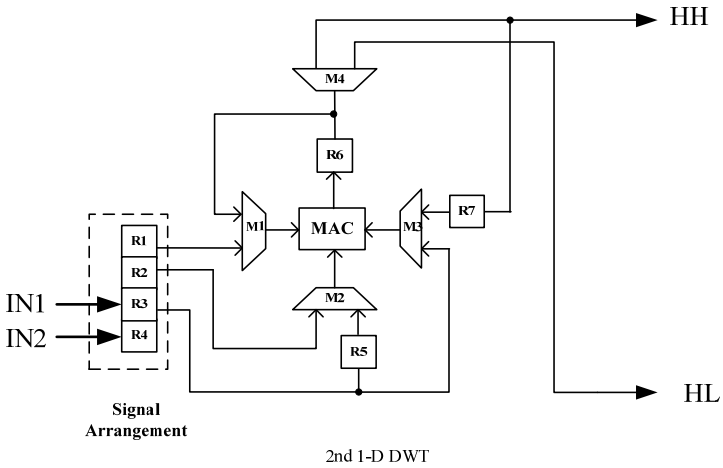


Fig. 15. The block diagram of the second stage 1-D LDWT.

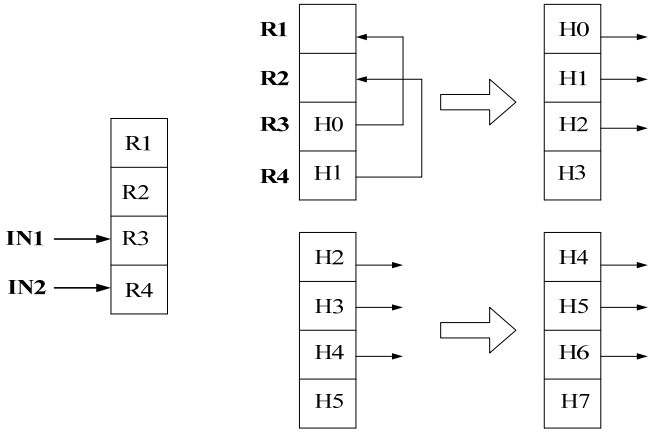


Fig. 16. Signal merging process for the signal arrangement unit.

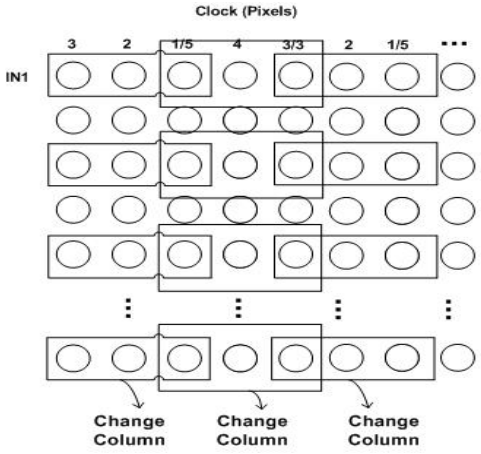
5.3 2-D LDWT architecture

In our IRSA operation, IN1 and IN2 read signals of even row and odd row in a zig-zag order, respectively. The detail is shown in Fig. 17. The block diagram of the proposed 2-D LDWT is shown in Fig. 19(b). It consists of two stages, the first stage 1-D DWT and the second stage 1-D DWT. This architecture needs only a small amount of transpose memory.

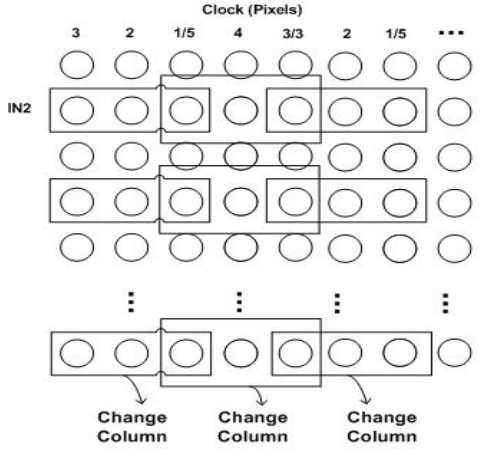
Let us consider a 4×4 image. The signal processing of the first stage 1-D DWT is shown in Fig. 18. The pixels from the even rows are processed by the upper 1-D DWT, and the pixels from the odd rows are processed by the lower 1-D DWT. Each 1-D DWT generates one set of 2×2 high-frequency coefficients and one set of 2×2 low-frequency coefficients, respectively. The generated coefficients are fed to the second stage 1-D DWT under the direction of the arrow head. The input for each second stage 1-D DWT becomes a set of 2×4 signals. The

signal processing of the second stage 1-D DWT is shown in Fig. 19. The 2×4 signals in each second stage 1-D DWT are then processed, and then HH, HL, LH, and LL are generated and each has 2×2 signal data.

The complete architecture of the 2-D LDWT is shown in Fig. 19. The complete 2-D LDWT consists of four parts, two sets of the first stage 1-D DWT, two sets of the second stage 1-D DWT, control unit, and MAC unit.



(a)



(b)

Fig. 17. The input signal sequences. (a) IN1 read signal of even row in zig-zag orders. (b) IN2 read signal of odd row in zig-zag orders.

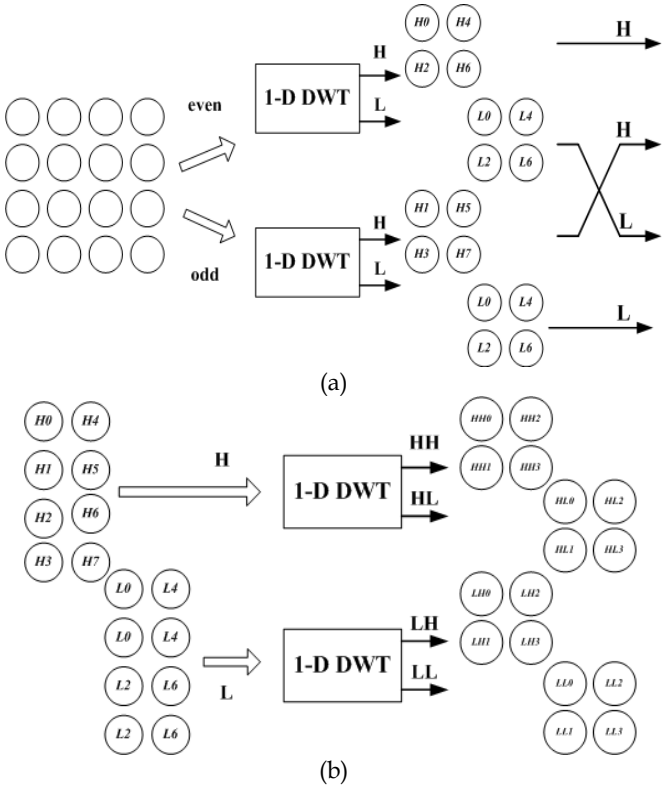


Fig. 18. The signal process of the two stage LDWT. (a) First stage 1-D LDWT. (b) Second stage 1-D LDWT.

According to (12) and (13), the proposed IRSA architecture can also be applied to the 9/7 mode LDWT. Fig. 20 illustrates the approach. From Figs. 10 and 11 in Section 3, the original signals (denoted as black circles) for both 5/3 and 9/7 modes LDWT can be processed by the same IRSA for the first stage 1-D DWT operation. The high-frequency signals (denoted as grey circles) and the correlated low-frequency signals together with the results of the first stage are used to compute the second stage 1-D DWT coefficients. Compared to the 9/7 mode LDWT computation, the 5/3 mode LDWT is much easier for computation, and the registers arrangement in Figs. 12 and 15 is simple. For 9/7 mode LDWT implementation with the same system architecture of 5/3 mode LDWT, we have to do the following modifications: 1) The control signals of the MUX in Figs. 12 and 15 must be modified. We have to rearrange the registers for the MAC block to process the 9/7 parameters. 2) The wavelet coefficients of the dual-mode LDWT are different. The coefficients are $\alpha = -1/2$ and $\beta = 1/4$ for 5/3 mode LDWT, but the coefficients are $\alpha = -1.586134142$, $\beta = -0.052980118$, $\gamma = +0.882911075$, and $\delta = +0.443506852$ for 9/7 mode LDWT. For calculation simplicity and good precision, we can use the integer approach proposed by Huang *et al*. (Huang et al., 2004) and Martina *et al*. (Martina & Masera, 2007) for 9/7 mode LDWT calculation. Similar to the multiplication implementation by shifters and adders in the 5/3 mode LDWT, we can adopt the shifters approach proposed

in (Huang et al., 2005) further to implement the 9/7 mode LDWT. 3) According to the characteristics of the 9/7 mode LDWT, the control unit in Fig. 19(b) must be modified accordingly.

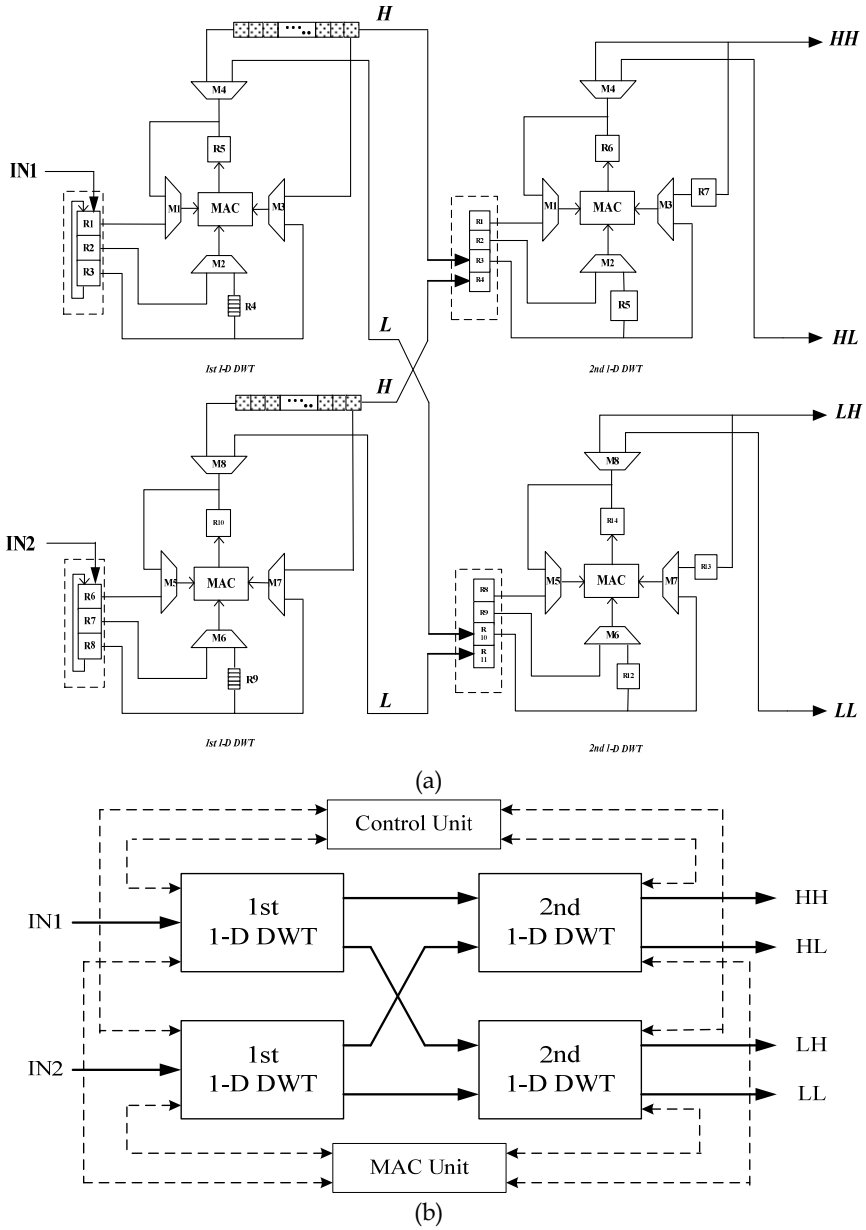


Fig. 19. The complete 2-D DWT block diagram. (a) DSP diagram of the 2-D LDWT. (b) System diagram of the 2-D LDWT.

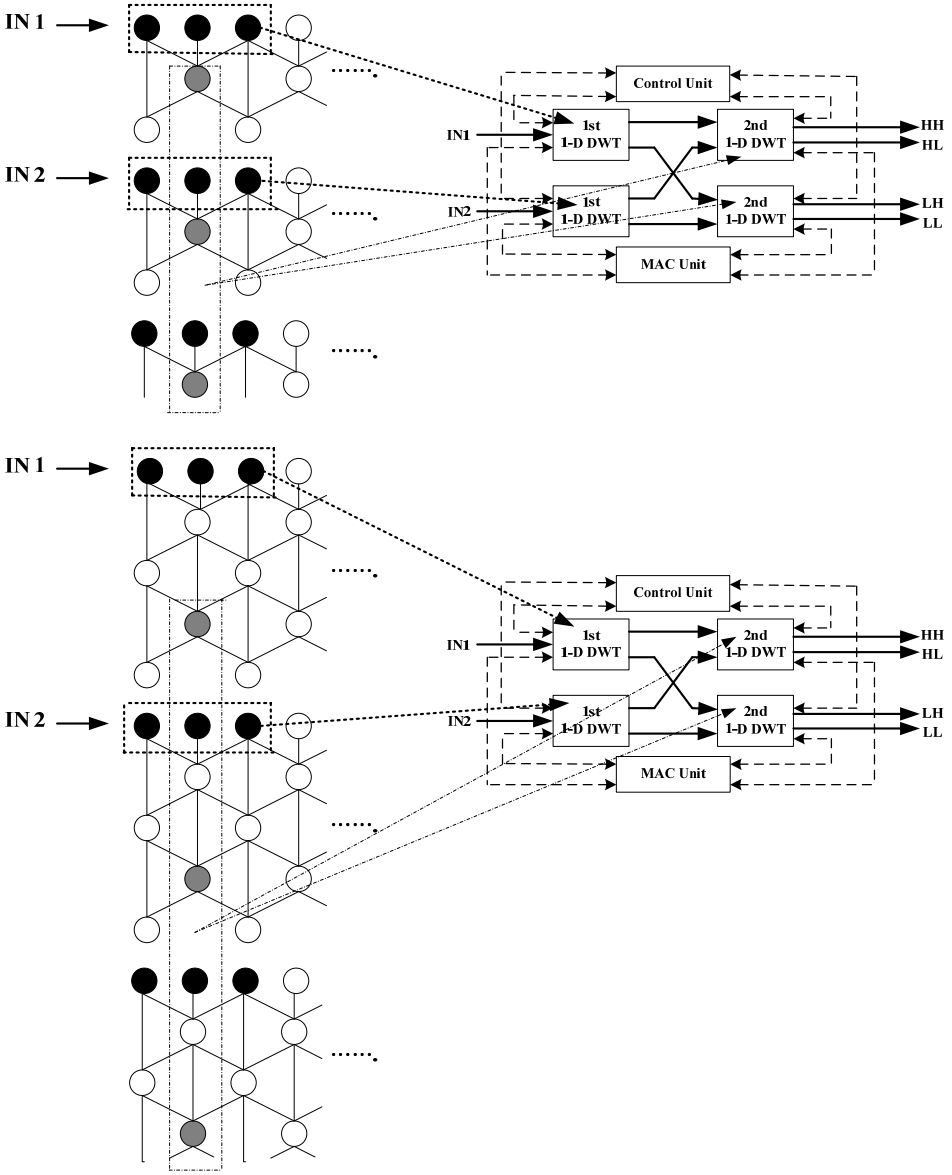


Fig. 20. The processing procedures of 2-D dual-mode LDWTs under the same IRSA architecture.

The multi-level DWT computation can be implemented in a similar manner by the high performance 1-level 2-D LDWT. For the multi-level computation, this architecture needs $N^2/4$ off-chip memory. As illustrated in Fig. 21, the off-chip memory is used to temporarily store the LL subband coefficients for the next iteration computations. The second level

computation requires $N/2$ counters and $N/2$ FIFO's for the control unit. The third level computation requires $N/4$ counters and $N/4$ FIFO's for the control unit. Generally in the jth level computation, we need $N/2^{j-1}$ counters and $N/2^{j-1}$ FIFO's.

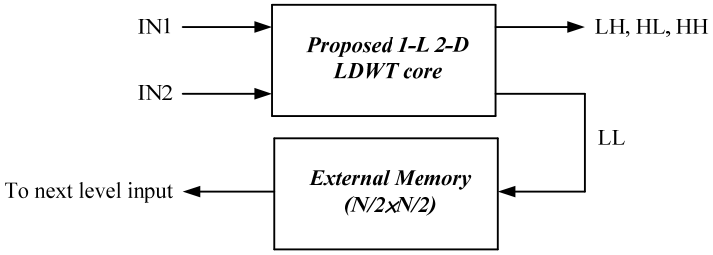


Fig. 21. The multilevel 2-D DWT architecture.

6. Experimental results and comparisons

The 2-D dual-mode LDWT considers a trade-off between low transpose memory and low complexity in the design of VLSI architecture. Tables 2 and 3 show the performance comparisons of the proposed architecture and other similar architectures. Compression results indicate that the proposed VLSI architecture outperforms previous works in terms of transpose memory size, requiring about 50% less memory than the JPEG2000 standard (Chen, 2004) architecture. Moreover, the 2-D LDWT is frame-based, and its implementation bottleneck is the huge transpose memory. Less memory units are needed in our architecture and the latency is fixed on $(3/2)N+3$ clock cycles. Our architecture can also provide an embedded symmetrical extension function. The proposed IRSA approach has the advantages of memory-efficient and high-speed. The proposed 2-D dual-mode LDWT adopts parallel and pipelined schemes to reduce the transpose memory and increase the operation speed. The shifters and adders replace multipliers in the computation to reduce the hardware cost. Chen *et al*. (Chen & Wu, 2002) proposed a folded and pipelined architecture to compute the 2-D 5/3 lifting-based DWT, and they used transpose memory size of $2.5N$ for an $N \times N$ 2-D DWT. This lifting architecture for vertical filtering with two adders and one multiplier is divided into two parts, and each part has one adder and one multiplier. Because both parts are activated in different cycles, they can share the same adder and multiplier. It can increase the hardware utilization and reduce the latency. However, according to the characteristics of the signal flow, it will increase the complexity at the same time.

A 256×256 2-D LDWT was designed and simulated with VerilogHDL and further synthesized by the Synopsys design compiler with TSMC 0.18μm 1P6M CMOS standard process technology. The detailed specs of the 256×256 2-D LDWT are listed in Table 4.

5/3 LDWT architecture	Ours	Diou et al., 2001	Andra et al., 2002	Chen & Wu, 2002	Chen, 2002	Chiang & Hsia, 2005	Mei et al., 2006	Huang et al, 2005	Wu & Lin, 2005
Transpose memory[1] (bytes)	2N	3.5N	3.5N	2.5N	3N	N^2/4+5N	2N	3.5N	3.5N
Computation time[2]	(3/4)N^2+(3/2)N +7	---	(N^2/2)+N+5	N^2	(N^2/2)+N +5	N^2	(N^2/2)+N	---	10+(4/3)N^2[1-(1/4)]+2N[1-(1/2)]
Adders	8	12	8	6	5	4	8	---	---
Multipliers	0	6	4	4	0	0	0	---	6

[1] Transpose memory size is used to store frequency coefficients in the 1-L 2-D DWT.
[2] In a system, computing time represents the time used to compute an image of size N×N.
[3] Suppose the image is of size N×N.
Table 2. Comparisons of 2-D architectures for 5/3 LDWT.

9/7 LDWT architecture	Ours	Andra et al., 2002	Jung & Park, 2005	Chen, 2004[1]	Vishwanath et al., 1995	Huang et al., 2005	Huang et al, 2005	Wu & Lin, 2005	Lan et al., 2005	Wu. & Chen, 2001
Transpose memory (bytes)	4N	N^2	12N	N^2/4+L N+L	22N	14N	5.5N	5.5N	---	N^2+4N+4
Computation time	(3/4)N^2 +(3/2) N +7	4N^2/3+2	N^2	N^2/2~(2/3)N	N^2	---	---	22+(4/3)N^2[1-(1/4)]+6N[1-(1/2)]	---	2N^2/3
Adders	16	8	12	4 L	36	16	16	8	32	16
Multipliers	0	4	9	4 L	36	12	10	6	20	16

[1] L: the filter length.
Table 3. Comparisons of the 2-D architectures for 9/7 LDWT.

Chip specification	N = 256, Tile size = 256×256
Gate count	29,196 gates
Power supply	1.8V
Technology	TSMC 0.18μm 1P6M (CMOS)
On-Chip memory size (Transpose + Internal)	2-D 5/3 DWT: 512 bytes 2-D 9/7 DWT: 1,024 bytes
Latency	(3/2)N+3 = 387 clock cycles
Computing time	(3/4)N^2+(3/2)N+7 = 49,543 clock cycles
Maximum clock rate	83 MHz

Table 4. Design specification of the proposed 2-D DWT.

7. Conclusions

This work presents a new architecture to reduce the transpose memory requirement in 2-D LDWT. The proposed architecture has a mixed row- and column-wise signal flow, rather than purely row-wise as in traditional 2-D LDWT. Further we propose a new approach, interlaced read scan algorithm (IRSA), to reduce the transpose memory for a 2-D dual-mode LDWT. The proposed 2-D architectures are more efficient than previous architectures in trading off low transpose memory, output latency, control complexity, and regular memory access sequence. The proposed architecture reduces the transpose memory significantly to a memory size of only 2*N* or 4*N* (5/3 or 9/7 mode) and reduces the latency to (3/2)*N*+3 clock cycles. Due to the regularity and simplicity of the IRSA LDWT architecture, a dual mode (5/3 and 9/7) 256×256 2-D LDWT prototyping chip was designed by TSMC 0.18μm 1P6M standard CMOS technology. The 5/3 and 9/7 filters with different lifting steps are realized by cascading the four modules (split, predict, update, and scaling phases). The prototyping chip takes 29,196 gate counts and can operate at 83 MHz. The method is applicable to any DWT-based signal compression standard, such as JPEG2000, Motion-JPEG2000, MPEG-4 still texture object decoding, and wavelet-based scalable video coding (SVC).

8. References

Andra, K.; Chakrabarti, C. & Acharya, T. (2000). A VLSI architecture for lifting-based wavelet transform, *IEEE Workshop on Signal Processing Systems*, (October 2000) pp. 70-79.

Andra, K.; Chakrabarti, C. & Acharya, T. (2002). A VLSI architecture for lifting-based forward and inverse wavelet transform, *IEEE Transactions on Signal Processing*, Vol. 50, No.4, (April 2002) pp. 966-977.

Chen, P.-Y. (2002). VLSI implementation of discrete wavelet transform using the 5/3 filter, *IEICE Transactions on Information and Systems*, Vol. E85-D, No.12, (December 2002) pp. 1893-1897.

Chen, P.-Y. (2004). VLSI implementation for one-dimensional multilevel lifting-based wavelet transform, *IEEE Transactions on Computer*, Vol. 53, No. 4, (April 2004) pp. 386-398.

Chen, P. & Woods, J. W. (2004). Bidirectional MC-EZBC with lifting implementation, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 14, No. 10, (October 2004) pp. 1183-1194.

Chen, S.-C. & Wu, C.-C. (2002). An architecture of 2-D 3-level lifting-based discrete wavelet transform, *VLSI Design/ CAD Symposium,* (August 2002) pp. 351-354.

Chiang, J.-S. & Hsia, C.-H. (2005). An efficient VLSI architecture for 2-D DWT using lifting scheme, *IEEE International Conference on Systems and Signals*, (April 2005) pp. 528-531.

Christopoulos, C.; Skodras, A. N. & Ebrahimi, T. (2000). The JPEG2000 still image coding system: An overview, *IEEE Trans. on Consumer Electronics*, Vol. 46, No. 4, (November 2000) pp. 1103-1127.

Daubechies, I. & Sweldens, W. (1998). Factoring wavelet transforms into lifting steps, *The Journal of Fourier Analysis and Applications*, Vol. 4, No.3, (1998) pp. 247-269.

Diou, C.; Torres, L. & Robert, M. (2001). An embedded core for the 2-D wavelet transform, *IEEE on Emerging Technologies and Factory Automation Proceedings*, Vol. 2, (October 2001) pp. 179-186.

Habibi, A. & Hershel, R. S. (1974). A unified representation of differential pulse code modulation (DPCM) and transform coding systems, *IEEE Transactions on Communications*, Vol. 22, No. 5, (May 1974) pp. 692-696.

Hsia, C.-H. & Chiang, J.-S. (2008). New memory-efficient hardware architecture of 2-D dual-mode lifting-based discrete wavelet transform for JPEG2000, *IEEE International Conference on Communication Systems*, (November 2008) pp. 766-772.

Huang, C.-T.; Tseng, P.-C. & Chen, L.-G. (2002). Efficient VLSI architecture of lifting-based discrete wavelet transform by systematic design method, *IEEE International Symposium Circuits and Systems*, Vol. 5, (May 2002) pp. 26-29.

Huang, C.-T.; Tseng, P.-C. & Chen, L.-G. (2004). Flipping structure: An efficient VLSI architecture for lifting-based discrete wavelet transform, *IEEE Transactions on Signal Processing*, Vol. 52, No. 4, (April 2004) pp. 1080-1089.

Huang, C.-T.; Tseng, P.-C. & Chen, L.-G. (2005). VLSI architecture for lifting-based shape-adaptive discrete wavelet transform with odd-symmetric filters, *Journal of VLSI Signal Processing Systems*, Vol. 40, No. 2, (June 2005) pp.175-188.

Huang, C.-T.; Tseng, P.-C. & Chen, L.-G. (2005). Analysis and VLSI architecture for 1-D and 2-D discrete wavelet transform, *IEEE Transactions on Signal Processing*, Vol. 53, No. 4, (April 2005) pp. 1575-1586.

Huang, C.-T.; Tseng, P.-C. & Chen, L.-G. (2005). Generic RAM-based architecture for two-dimensional discrete wavelet transform with line-based method, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 15, No. 7, (July 2005) pp. 910-919.

ISO/IEC 15444-1 JTC1/SC29 WG1. (2000). *JPEG 2000 Part 1 Final Committee Draft Version 1.0*, Information Technology.

ISO/IEC JTC1/SC29/WG1 Wgln 1684 (2000). *JPEG 2000 Verification Model 9.0*.

ISO/IEC 15444-1 JTC1/SC29 WG1. (2000). *Motion JPEG2000, ISO/IEC ISO/IEC 15444-3*, Information Technology.

ISO/IEC JTC1/SC29 WG11. (2001), *Coding of Moving Pictures and Audio*, Information Technology.

Jiang, W. & Ortega, A. (2001). Lifting factorization-based discrete wavelet transform based architecture design, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 11, No. 5, (May 2001) pp. 651-657.

Jung, G.-C. & Park, S.-M. (2005). VLSI implement of lifting wavelet transform of JPEG2000 with efficient RPA (recursive pyramid algorithm) realization, *IEICE Transactions on Fundamentals*, Vol. E88-A, No. 12, (December 2005) pp. 3508-3515.

Kondo, H. & Oishi, Y. (2000). Digital image compression using directional sub-block DCT, *International Conference on Communications Technology*, Vol. 1, (August 2000) p p. 985 -992.

Lan, X.; Zheng, N. & Liu, Y. (2005). Low-power and high-speed VLSI architecture for lifting-based forward and inverse wavelet transform, *IEEE Transactions on Consumer Electronics*, Vol. 51, No. 2, (May 2005) pp. 379-385.

Li, W.-M.; Hsia, C.-H. & Chiang, J.-S. (2009). Memory-efficient architecture of 2-D dual-mode lifting scheme discrete wavelet transform for Moion-JPEG2000, *IEEE International Symposium on Circuits and Systems*, (May 2009) pp. 750-753.

Lian, C.-J.; Chen, K.-F.; Chen, H.-H. & Chen, L.-G. (2001). Lifting based discrete wavelet transform architecture for JPEG2000, *IEEE International Symposium on Circuits and Systems*, Vol. 2, (May 2001) pp. 445-448.

Mallat, S. G. (1989). A theory for multi-resolution signal decomposition: The wavelet representation, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 7, (July 1989) pp. 674-693.

Mallat, S. G. (1989). Multi-frequency channel decompositions of images and wavelet models, IEEE *Transactions on* Acoustics, Speech and Signal Processing, Vol. ASSP-37, No. 12, (December 1989) pp. 2091-2110.

Marcellin, M. W.; Gormish, M. J. & Skodras, A. N. (2000). JPEG2000: The new still picture compression standard, *ACM Multimedia Workshops*, (September 2000) pp. 45-49.

Marino, F. (2000). Efficient high-speed/low-power pipelined architecture for the direct 2-D discrete wavelet transform, *IEEE Transactions on Circuits and Systems II*, Vol. 47, No. 12, (December 2000) pp. 1476-1491.

Martina, M. & Masera, G. (2007). Folded multiplierless lifting-based wavelet pipeline, *IET Electronics Letters*, Vol. 43, No. 5, (March 2007) pp. 27-28.

Mei, K.; Zheng, N. & van de Wetering, H. (2006). High-speed and memory-efficient VLSI design of 2-D DWT for JPEG2000, *IET Electronics Letter*, Vol. 42, No. 16, (August 2006) pp. 907-908.

Ohm, J.-R. (2005). Advances in scalable video coding, *Proceedings of The IEEE,* Invited Paper, Vol. 93, No.1, pp. 42-56, (January 2005) pp. 42-56.

Richardson, I. (2003). *H.264 and MPEG-4 Video Compression*, John Wiley & Sons Ltd.

Seo, Y.-H. & Kim, D.-W. (2007). VLSI architecture of line-based lifting wavelet transform for Motion JPEG2000, *IEEE Journal of Solid-State Circuits*, Vol. 42, No. 2, (February 2007) pp. 431-440.

Sweldens, W. (1996). The lifting scheme: A custom-design construction of biorthogonal wavelets, *Applied and Computation Harmonic Analysis*, Vol. 3, No. 15, (1996) pp.186-200.

Tan, K.C.B. & Arslan, T. (2001). Low power embedded extension algorithm for the lifting based discrete wavelet transform in JPEG2000, *IET Electronics Letters*, Vol. 37, No. 22, (October 2001) pp.1328-1330.

Tan, K.C.B. & Arslan, T. (2003). Shift-accumulator ALU centric JPEG 2000 5/3 lifting based discrete wavelet transform architecture, *IEEE International Symposium on Circuits and Systems,* Vol. 5, (May 2003) pp. V161-V164.

Taubman, D. & Marcellin, M. W. (2001). *JPEG2000 image compression fundamentals, standards, and practice*, Kluwer Academic Publisher.

Varshney, H.; Hasan, M. & Jain, S. (2007). Energy efficient novel architecture for the lifting-based discrete wavelet transform, *IET Image Process*, Vol. 1, No. 3, (September 2007) pp.305-310.

Vishwanath, M.; Owens, R. M. & Irwin, M. J. (1995). VLSI architecture for the discrete wavelet transform, *IEEE Transactions on Circuits and Systems II*, Vol. 42, No. 5, (May 1995) pp. 305-316.

Weeks, M. & Bayoumi, M. A. (2002). Three-dimensional discrete wavelet transform architectures, *IEEE Transactions on Signal Processing*, Vol. 50, Vo.8, (August 2002) pp. 2050-2063.

Wu, B.-F. & Lin, C.-F. (2005). A high-performance and memory-efficient pipeline architecture for the 5/3 and 9/7 discrete wavelet transform of JPEG2000 codec, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 15, No. 12, (December 2005) pp. 1615-1628.

Wu, P.-C. & Chen, L.-G. (2001). An efficient architecture for two-dimensional discrete wavelet transform, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 11, No. 4, (April 2001) pp. 536-545.

5

Full HD JPEG XR Encoder Design for Digital Photography Applications

Ching-Yen Chien*, Sheng-Chieh Huang*, Chia-Ho Pan, and Liang-Gee Chen
DSP/IC Design Lab, Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University
**Sense/TCM SOC Lab, Department of Electrical and Control Engineering, National Chiao-Tung University*
Taiwan, R.O.C.

1. Introduction

Multimedia applications, such as radio, audio, camera phone, digital still camera, camcoder, and mobile broadcasting TV, are more and more popular in our life as the progress of image sensor, communication, VLSI manufacture, and image/video coding standards. With rapid progress of image sensor, display devices, and computing engines, image coding standards are used in the digital photography application everywhere. It has been merged together with our life such as camera phone, digital still camera, blog and many other applications.

Many advanced multimedia applications require image compression technology with higher compression ratio and better visual quality. High quality, high compression rates of digital image and low computational cost are important factors in many areas of consumer electronics, ranging from digital photography to the consumer display equipments such as digital still camera and digital frame. These requirements usually involve computationally intensive algorithms imposing trade-offs between quality, computational resources, and throughput.

For high quality of digital image applications, the extension of color range has becoming more important in the consumer product. In the past, the digital cameras and the display equipments in the consumer market typically had 8 bits information per channel. Today the condition is quite different. In the consumer market, digital cameras and the desktop display panels also have at least 12 bits of information per channel. If the information per channel of digital image is still compressed into 8 bits, 4 or more bits of information per channel are lost and the quality of the digital image is limited. Due to the improvement of the display equipments, the JPEG XR is designed for the high dynamic range (HDR) and the high definition (HD) photo size. JPEG XR which is already under organized by the ISO/IEC Joint Photographic Experts Group (JPEG) Standard Committee is a new still image coding standard and derived from the window media photo (Srinivasan et al., 2007; Srinivasan et al., 2008; Schonberg et al., 2008). The goal of JPEG XR is to support the greatest possible level

of image dynamic range and color precision, and keep the device implementations of the encoder and decoder as simple as possible.

For the compression of digital image, the Joint Photographic Experts Group, the first international image coding standard for continuous-tone natural images, was defined in 1992 (ITU, 1992). JPEG is a well-known image compression format today because of the population of digital still camera and Internet. But JPEG has its limitation to satisfy the rapid progress of consumer electronics. Another image coding standard, JPEG2000 (ISO/IEC, 2000), was finalized in 2001. Differed from JPEG standard, a Discrete Cosine Transform (DCT) based coder, the JPEG2000 uses a Discrete Wavelet Transform (DWT) based coder. The JPEG2000 not only enhances enhances the compression, but also includes many new features, such as quality scalability, resolution scalability, region of interest, and lossy/lossless coding in a unified framework. However, the design of JPEG2000 is much complicated than the JPEG standard. The core techniques and computation complexity comparisons of these two image coding standard are shown in (Huang et al., 2005).

For satisfaction of the high quality image compression and lower computation complexity, the new JPEG XR compression algorithm is discussed and implemented with the VLSI architecture. JPEG XR has high encoding efficiency and versatile functions. The XR of JPEG XR means the extended range. It means that JPEG XR supports the extended range of information per channel. The image quality of JPEG XR is nearly equal to JPEG 2000 with the same bit-rate. The computation complexity is much lower than JPEG2000 as shown in Table 1.

The efficient system-level architecture design is more important than the module design since system-level improvements make more impacts on performance, power, and memory bandwidth than the module-level improvements. In this chapter, the new JPEG XR compression standard is introduced and the analysis and architecture design of JEPG XR encoder are also proposed. Comparison was made to analyze the compression performance among JPEG2000 and JPEG XR. Fig. 1 is the peak signal-to-noise ratio (PSNR) results under several different bitrates. The test color image is 512x512 Baboon. The image quality of JPEG XR is very close to that of JPEG2000. The PSNR difference between JPEG XR and JPEG2000 is under 0.5dB. Fig. 2 shows the subjective views at 80 times compression ratio. The block artifact of JPEG image in Fig. 2 is easily observed, while the JPEG XR demonstrates acceptable qualities by implementing the pre-filter function. For the architecture design, a 4:4:4 1920x1080 JPEG XR encoder is proposed. From the simulation and analysis, entropy coding is the most computationally intensive part in JPEG XR encoder. We first proposed a timing schedule of pipeline architecture to speed up the entropy encoding module. To optimize memory bandwidth problem and maximize the silicon area efficiency, we also proposed a data reuse skill to solve this problem. The data reuse skill can reduce 33% memory bandwidth form the memory access. The hardware design of JPEG XR encoder has been implemented by cell-based IC design flow. The encoder design is also verified by FPGA platform. This JPEG XR chip design can be used for digital photography applications to achieve low computation, low storage, and high dynamical range features.

Technologies	Operations(GOPs)
JPEG2000	4.26
JPEG XR	1.2

Table 1. Machine cycles comparison between JPEG XR and JPEG 2000.

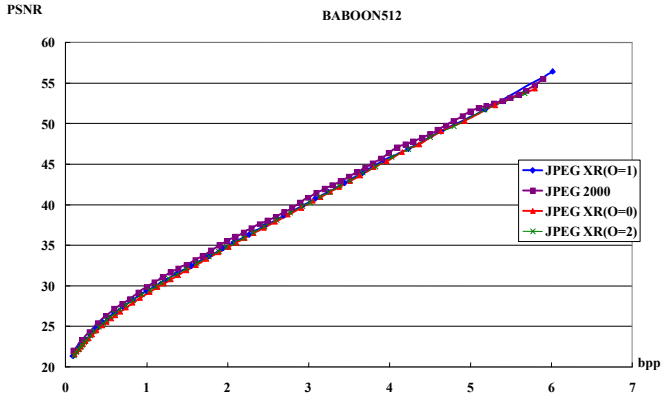


Fig. 1. PSNR Comparison between different image standard.

(a) (b)

Fig. 2. Image Quality Comparison with (a) JPEG XR and (b) JPEG.

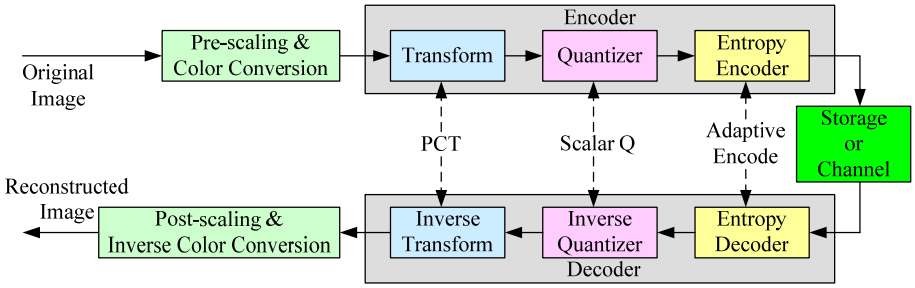


Fig. 3. JPEG XR Coding Flow.

The coding flow of JPEG XR is shown in the Fig. 3. The JPEG XR has lower computation cost in each module and simple coding flow while maintaining similar PSNR quality at the same bitrate as compared with the coding flow of JPEG2000. Hence, the JPEG XR is very suitable to be implemented with the dedicated hardware to deal with HD photo size image for the HDR display requirement. The JPEG XR encoder architecture design is presented in the following section.
This paper is organized as follows. In Section 2, we present the fundamentals of JPEG XR. Section 3 describes the characteristics of our proposed architecture design of JPEG XR encoder. Section 4 shows the implementation results. Finally, a conclusion is given in Section 5.

2. JPEG-XR

The JPEG XR image compression standard has many options for different purposes. In the following section, the fundamentals of JPEG XR are introduced.

2.1 Image Partition and Windowing

Figure 4 shows the partition and windowing example of JPEG XR standard. As the different tile size makes the different compression result, the compression results under the same quantization with different tile size for four test 512x512 benchmark images are discussed in (Pan et al., 2008). When the tile function is used, the compressed image size is increased with the tiles number when the small tiles size has been chosen.
When the image has been divided into the different tiles, each tiles are processed independently which are more suitable for the design of hardware implementation and memory buffer size. The pixels acrossing the boundary of the tiles have to be processed without any data dependency. The data dependancy must be changed accordingly, and the scan order has to be rebuilt as well. This characteristic overcome the error impact of data independency when the error occurs. Although the division of tiles is helpful for the hardware implementation and data reserving in error environment, but it decreases the data compression efficiency. However, it depends on the tradeoff between hardware design considerations and the data compression efficiency.

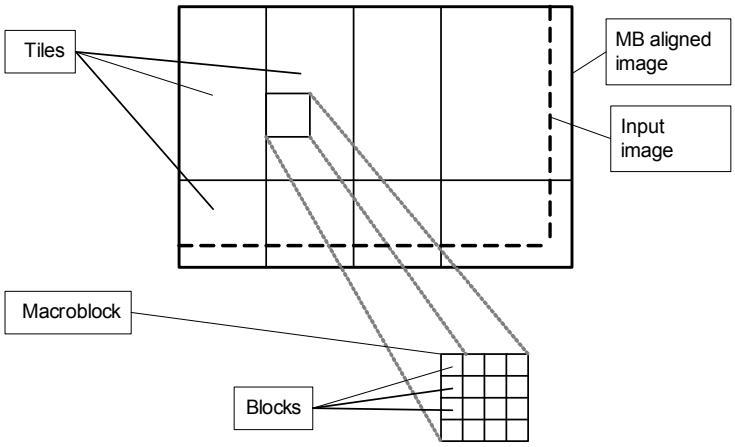


Fig. 4. Example of Image Partitions and Windowing.

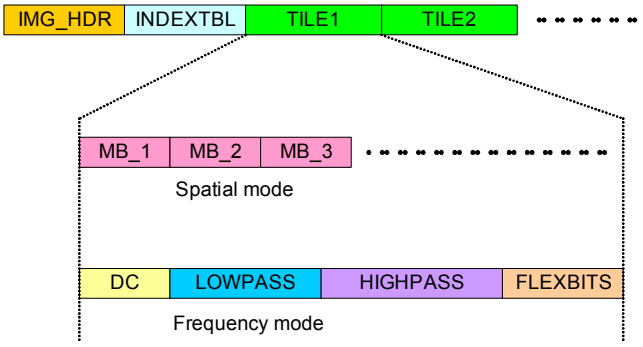


Fig. 5. Layout of Bitstream.

2.2 Operation Modes

In the Fig. 5, there are two modes of bitstream in JPEG XR which are named as spatial mode and frequency mode. In the spatial mode, a single tile packet carries the bitstream in macroblock raster order scanning from left to right and top to bottom. In the frequency mode, the bitstream of each tile is carried in multiple tile packets, where each tile packet carries transform coefficients of each frequency band in the tile. The freqency mode transmits the DC coefficients at first. When the user receive the bitstreams, images are decoded progressively. The frequency mode is very suitable for limited bandwidth applications such as web browsing.

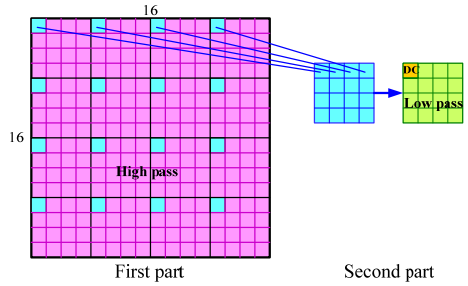


Fig. 6. The 1st part and 2nd part process of PCT.

2.3 Color Conversion

The purpose of color conversion is to reflect the color sensivity according to the characteristics of human eyes. The color space is converted from RGB to YUV. Then, the important energies are concentrate on Y channel. The color conversion is reversible, in other words, this color conversion is a lossless conversion. The color conversion equation is

$$\begin{aligned} V &= B - R \\ U &= -\left\lfloor R - G + \left\lfloor \frac{V}{2} \right\rfloor \right\rfloor \\ Y &= G + \left\lfloor \frac{U}{2} \right\rfloor - offset \end{aligned} \tag{1}$$

where offset is 128.

2.4 Pre-filter

There are three overlapping choices of the pre-filter function: non-overlapping, one-level overlapping and two-level overlapping. (Pan et al., 2008) discuss different trade-offs of three overlapping choices. The non-overlapping condition is used for fastest encoding and decoding. It is efficient in low compression ratio mode or lossless mode. However, this mode potentially introduces blocking effect in low bitrate images. The one-level overlapping function, compared to the non-overlapping, has higher compression ratio but it needs additional time for the overlapping operation. The two-level overlapping has the highest computation complexity. The PSNR and objective image quality of two-level overlapping are better than the other two at low bitrate area. The pre-filter function is recommended for both high image quality and further compression ratio considerations at high quantization levels. For high image quality issue, the pre-filter function eliminate the block effect which is sensitive of visual quality.

2.5 Photo Core Transform

Photo Core Transform (PCT) function transforms the pixel values after pre-filter computing to the lowest frequency coefficient DC, low frequency coefficients AD, and high frequency coefficients AC. There are two parts process for the PCT as shown in Fig. 6. The macroblock (MB) is partitioned into 16 4x4 blocks. In each part process, each 4x4 block is pre-filtered and then transformed by 4x4 PCT. A 2x2 transform is applied to four coefficients by four times

for each 4x4 block in first part process. The low frequency coefficient of these four coefficients is processed to the top-left coefficient. After first part process, the DC coefficients of 16 4x4 blocks can be collected as a 4x4 DC block. The second part process is for the 4x4 DC block from the first part process. The second part 2D 4x4 PCT is built by using the three operators: 2x2 T_h, T_odd and T_odd_odd. After second part process, the 16 DC coefficients are processed as DC and AD coefficients.

2.6 Quantization

The quantization is the process of rescaling the coefficients after applying the transform process. The quantization uses the quantized value to divide and round the coefficients after PCT transformed into an integer value. For the lossless coding mode, the quantized value = 1. For the lossy coding mode, the quantized value > 1. The quantization of JPEG XR use integer operations. The advantage of integer operation keeps the precision after scaling operations and only uses shift operation to perform the division operations. The quantization parameter is allowed to differ across high pass band, low pass band, and DC band. It varies to different values according to the sensitivity of human vision in different coefficient bands. Fig. 7 shows a example that pixels transformed by 2 stage PCT and quantized by the quantization process.

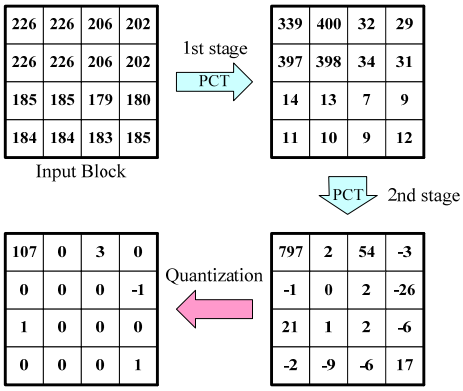


Fig. 7. Example of PCT.

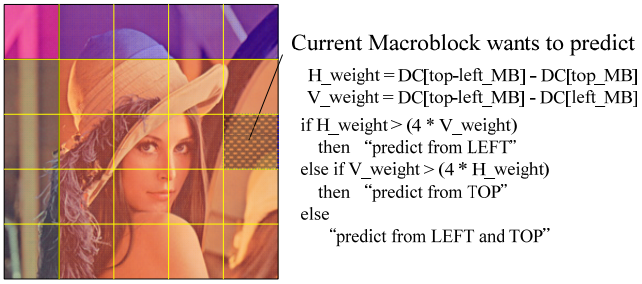


Fig. 8. DC prediction model.

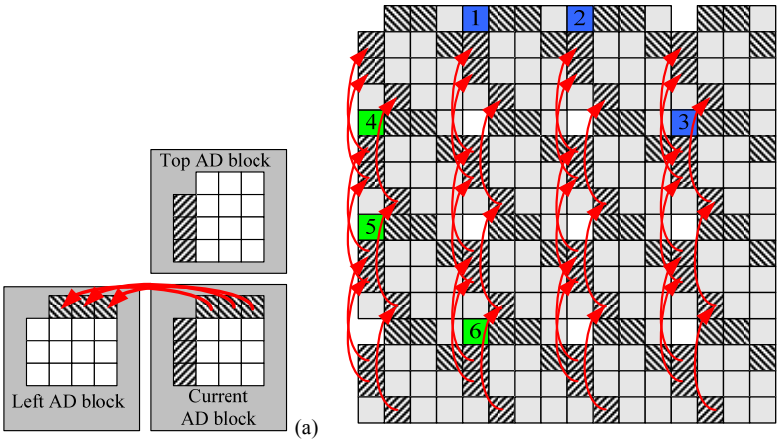


Fig. 9. Example of (a) Prediction of AD coefficients. (b) Prediction model and AC coefficients.

2.7 Prediction

There are three directions of DC prediction: LEFT, TOP, and LEFT and TOP. As shown in Fig. 8, the JPEG XR prediction rules of DC coefficient use the DC coefficient of left MB and top MB to decide the direction of DC prediction. The DC predicted direction can be decided by the pseudo code described in Fig. 8. After comparing the H_weight and the V_weight, the predicted direction can be decided. At the boundary MBs of image, the purple MBs only predict from left MB and the gray MBs only predict from top MB.

The AD/AC block can be predicted from its TOP or LEFT block as Fig. 9. If the predicted direction is LEFT, the prediction relationship example of AD coefficients is shown as Fig. 9(a). The AD predicted direction follows DC prediction model as described in Fig. 8. The computation after prediction judgment of AC is a little similar to AD that can reduce the coefficient value of the block. The Fig. 9(b) shows the prediction relationship example of AC coefficients when the prediction direction is TOP.

2.8 Entropy Encoding

The adaptive scan is used for entropy coding which is based on the latest probability of non-zero coefficients. Fig. 10 shows an example that the previous scan order is changed to update scan order based on the probability. The most probable non-zero coefficients are scanned firstly and the probability is counted by numbers of the non-zero coefficients. If the non-zero probability is larger than the previous scanned coefficient, the scan order is exchanged. By doing so, the non-zero coefficients are collected together and processed in an orderly fashion.

After the coefficients of current block are scanned by adaptive scan order, JPEG XR uses two entropy coding schemes. Fig. 11 demonstrates an example of coding a 4x4 block when ModelBits is 4 bits. The coefficients of 4x4 block are scanned by adaptive scan order first, then the ModelBits is updated by the total overhead coefficients in each band per channel. The coefficients which can be represented under the ModelBits are encoded by the FlexBits table. For the extra bit such as the 17 can not be represented in four bits, the encoding block

will increase extra bit for the new Run-Level Encode (RLE) coding. The RLE function is added into the bitstream length for the overhead coefficient. The Levels, the Runs before nonzero coefficients, and the number of overhead coefficients are counted for RLE. Then the RLE block is encoded firstly while different size of Huffman table is used to make the bit allocation in optimization. After processing the RLE algorithm, the RLE results and FlexBits are packetized.

3. Architecture Design of JPEG XR

In the following section, a JPEG XR encoder architecture design is presented. For the system requirement, the consideration of functional block pipelining, and the architecture design of the pre-filter, transform, quantization, prediction, and the entropy coding modules are all discussed in the following sections.

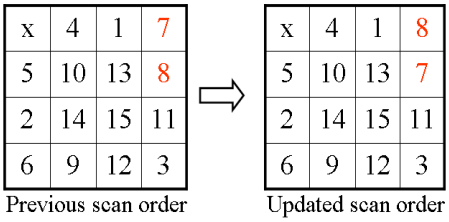


Fig. 10. Adaptive scan order updating example.

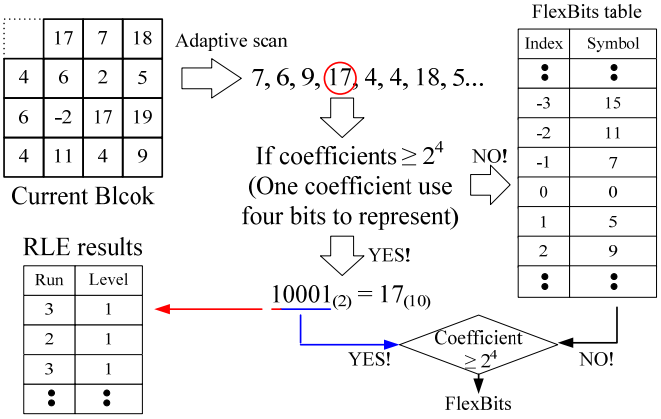


Fig. 11. Adaptive scan and Run-Level encode example.

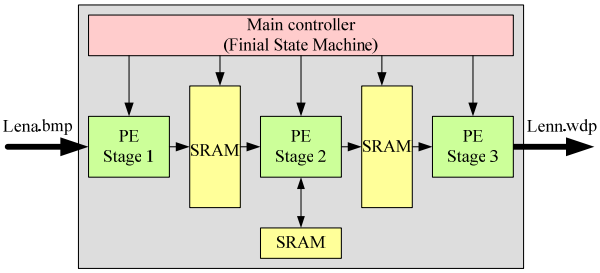


Fig. 12. Pipeline stage consideration of JPEG XR.

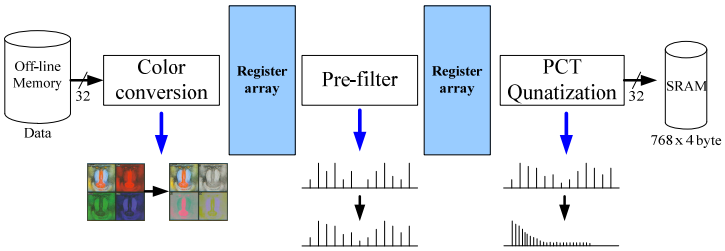


Fig. 13. Data flow of JPEG XR stage 1.

3.1 System Architecture and Functional Block Pipelining

The pipeline stages of this JPEG XR encoder design can be divided into three steps, as shown in Fig. 12. Since the color conversion, pre-filter and the PCT modules are computed with the 4x4 block matrix style with no feedback information, they are arranged into the same stage at the beginning. The prediction unit is used as second stage for different direction comparisons with the feedback processed pixels for the DC/AD/AC region. Then, the entropy encoding module with high data dependency are divided as the third stage.

3.2 Color Conversion, Pre-filter, PCT, and Quantization

Fig. 13 shows the data flow of stage 1. A 32 bits external bus can transmit 2 R/G/B coefficients in one clock cycle. The pre-filter and PCT/quantization modules process the coefficients after color conversion. This paper uses a memory reused method to reduce the memory access bandwidth for the color conversion, pre-filter, and PCT module. The black line in Fig. 14 is the boundary of MB. When the blue range is to be processed after the yellow range, the pre-filter function only have to deal with the amount of new data instead of the entire block data into the register. This is due to the presence of previously stored column coefficients of blocks in the registers and the capability to be utilized as the coefficients in the yellow range. Therefore, the memory buffer size and the memory access for pre-filter are reduced. Otherwise, the additional SRAM will be needed to store the data of the entire block from off-chip memory and the execution time will be increased. The detail simulations is discussed in (Pan et al., 2008).

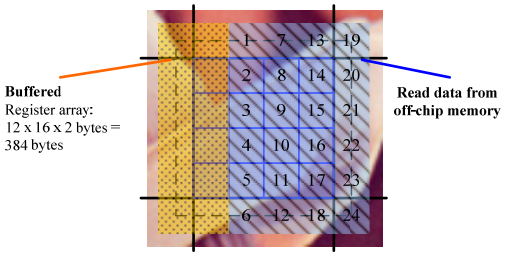


Fig. 14. Memory reused method.

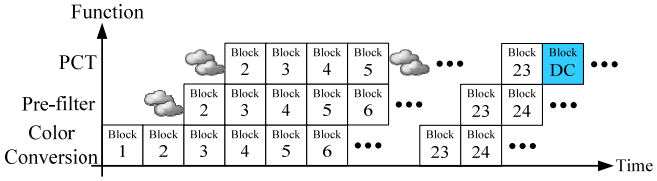


Fig. 15. DC block insertion on the block pipeline.

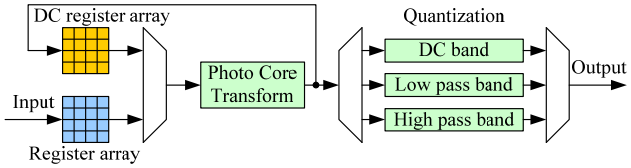


Fig. 16. Architecture of PCT, and Quantization.

Because the functions of stage 1 are computing with the 4x4 block style with no feedback information, the three pipeline architecture for stage1 is used: color conversion (CC), pre-filter, PCT (include quantization). For the memory allocation, the color conversion requires 6 blocks per column, the pre-filter requires 5 blocks, and the PCT including quantization uses 4 blocks to execute the function. At the end of PCT in Fig. 15, DC block can be processed to reduce one pipeline bubble when the next new pipeline stage starts. Hence, the well arranged timing schedule eliminated the access of additional clock cycle to process the DC block.

The architecture for PCT including Quantization is shown in Fig. 16. Additional registers are implemented to buffer the related coefficients for the next pipeline processing. The left multiplex selects the inputs for two parts PCT process. Initially, the input pre-filtered coefficients are selected to process the PCT algorithm. Then, the yellow block (DC) will be processed after the 16 blocks have been computed. The quantization stage de-multiplex the DC, low pass band AD and high pass band AC coefficients to suitable process element. The processed data are arranged into the quantized coefficients of Y, U, V SRAM blocks for prediction operation.

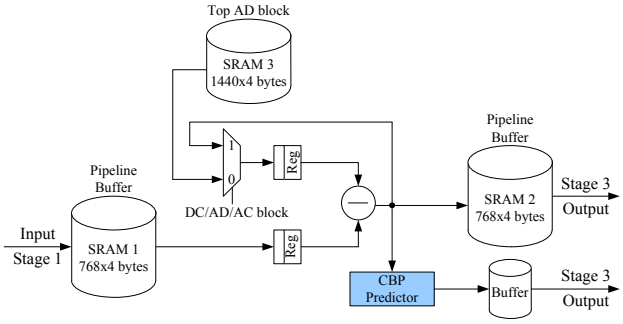


Fig. 17. Prediction architecture.

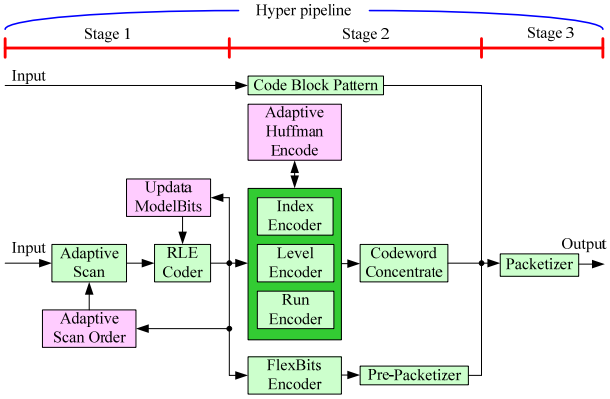


Fig. 18. Adaptive encode architecture.

3.3 Prediction

The quanitzed data stored in Y, U, V SRAM blocks are processed with the subtract operation as the prediction algorithm. Three SRAM blocks are used in the this design. One 1440x4 byte SRAM is used to buffer the 1 DC and 3 AD coefficients for the TOP AD prediction coefficients in the prediction judgement, so that the regeneration of these data are unnecessary when they are selected in the prediction mode. In Fig. 17, two 768x4 bytes SRAMs are used to save the quantized coefficients of current block and the predicted coefficients for current block.

3.4 Entropy Encoding

There are three complex data dependency loops in the entropy encoding module as shown in Fig. 18. The first one is the adaptive-scan-order block used to refresh the scan order. The second is the updated ModelBits block, which decides how many bits are necessary to represent a coefficient. Third, the adaptive Huffman encode block to choose the most efficient Huffman table.

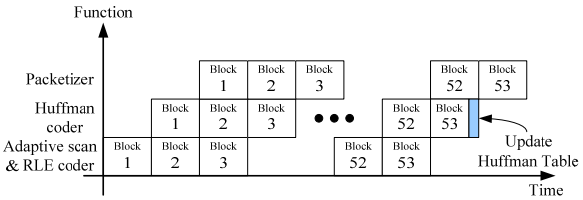


Fig. 19. Timing scheduling of entropy encoding.

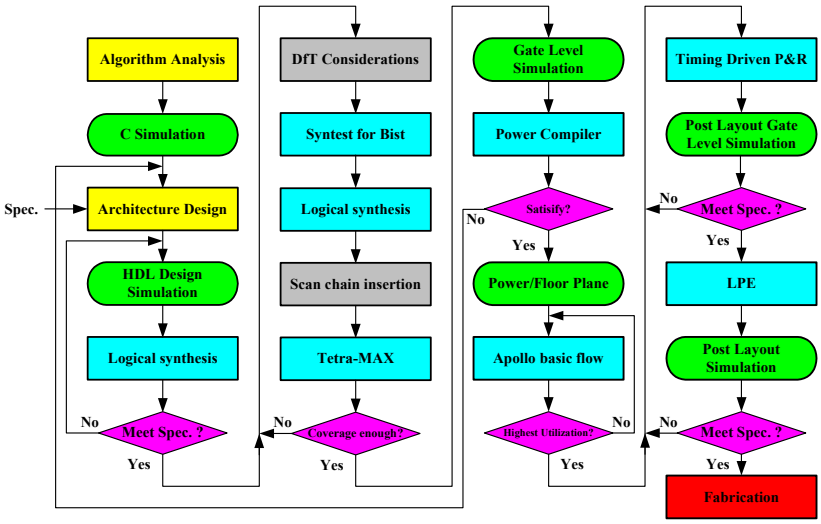


Fig. 20. Chip design flow.

In order to increase the throughput and decrease the timing of critical patch, we use three sub-pipeline stages architecture to implement the entropy encoding module as described in (Chien et al., 2009). Because of the feedback patch, stage 1 includes the modules of feedback path and the feedback information generator. Stage 2 includes the modules that encode the RLE information to symbol-based data. The bit-wise packetizer module of stage 3 processes the symbol-based data bit by bit. By doing this, we can increase the throughput about 3 times by well arranged pipeline timing schedule as shown in Fig. 19.

The design of adaptive scan block counts the numbers of the non-zero coefficients to decide whether the scan order should be exchanged or not. After the processing of the adaptive scan block, the coefficients which can be represented under ModelBits are coded by the FlexBits table. The other coefficients change to generate the Flexbits and Level. After the processing of the RLE algorithm, the Level and Run choose the suitable Huffman table to generate the RLE codeword. The RLE results are translated into the codewords by different Huffman tables. The Huffman encoder in JPEG XR is different from the other standards. It is composed of many small Huffman tables and can adaptively choose the best option in these tables for the smaller codesize for Run and Level. Many codewords are produced after the Huffman encoding. In order to increase the throughput, the codeword concentrating

architecture is proposed. Linked to the output of the above operation, the whole RLE codeword and RLE codesize will be produced to the packetizer. The packetizer architecture is modified from the (Agostini et al., 2002) architecture by combining the RLE codeword and the FlexBits for generating the JPEG XR compressed file. More detail design is described in (Pan et al., 2008).

4. Implementations

This design is implemented to verify the proposed VLSI architecture for JPEG XR encoder. And it is also verified by FPGA platform. The detail information about implementation result of each module by the FPGA prototype system is shown in (Pan et al., 2008). It is used to test the conformance bitstreams for the certification.

A three-stage MB pipelining of 4:4:4 lossless JPEG XR encoder was proposed to process the capacity and hardware utilization. In our design, the extra registers are used to increase the pipeline stages for achieving the specification, such as the color conversion, PCT/ quantization and the adaptive encode block. And the on-chip SRAM blocks are used to store the reused data processed with the prediction module to eliminate the memory access. For the entropy encoding module, the timing schedule and pipelining is well designed. The proposed architecture of entropy encoding module increases the total throughput about three times. We use 0.18um TSMC CMOS 1P6M process to implement the JPEG XR encoder. Our design flow is standard cell based chip design flow. The design flow of our design is shown as Fig 20. Test consideration is also an important issue in chip design. Therefore, the scan chain and the built-in self-test (BIST) are considered in our chip. The chip synthesis layout is shown as Fig. 21. The implementation results are shown as the Table 2. The power dissipation distribution in shown as Fig. 22.

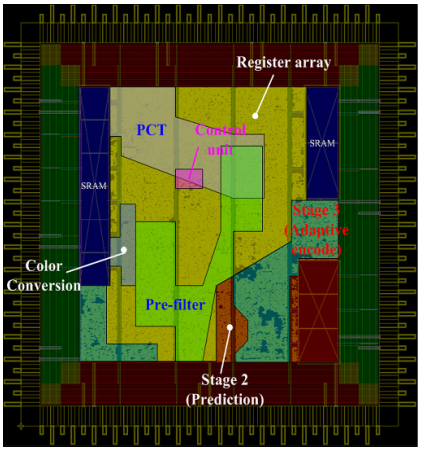


Fig. 21. Chip synthesis layout.

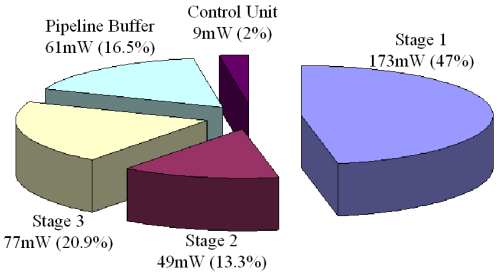


Fig. 22. Chip power dissipation distribution.

Technology	TSMC 0.18um CMOS 1P6M
Core Size	3.18 mm x 3.18 mm (9.3025 mm^2)
Die Size	4.64 mm x 4.64 mm (20.47 mm^2)
Gate Count	651.7 K
Work Clock Rate	62.5 MHz
Power Consumption	368.7 mW@62.5MHz，1.8V
On-Chip Memory	256x32 SRAM x 6 (single port) 768x32 SRAM x 2 (single port) 480x32 SRAM x 3 (single port) Total: 144,384 Bits = 18,047 Bytes
Processing Capability	34.1 Mega pixels within one second 5.45 fps for 4:4:4 HDTV(1920x1080) @62.5MHz 32.9 fps for 4:4:4 VGA(640x480) @62.5MHz 112 fps for 4:4:4 CIF(352x288) @62.5MHz
Input Pad	37
Output Pad	34

Table 2. Chip specification.

5. Conclusion

Compared with the JPEG2000, the coding flow of the JPEG XR is simple and has lower complexity in the similar PSNR quality at the same bit rate. Hence, the JPEG XR is very suitable for implementation with the dedicated hardware used to manage HD photo size images for the HDR display requirement. In this paper, we initially analyzed the comparison of JPEG XR with other image standard, and then a three-stage MB pipelining was proposed to process the capacity and hardware utilization. We also made a lot of efforts on module designs. The timing schedule and pipelining of color conversion, pre-filter, PCT & quantization modules are well designed. In order to prevent accessing the coefficients from off-chip memory, an on-chip SRAM is designed to buffer the coefficients for the prediction module with only some area overhead. The pre-filter and PCT function was designed to reduce 33.3% memory access from off-chip memory.For the entropy coding, we designed a codeword concentrating architecture for the throughput increasing of RLE algorithm. And the adaptive encode and packetizer modules efficiently provide the coding information required for packing the bitstream. Based on this research result, we contribute

a VLSI architecture for 1920x1080 HD photo size JPEG XR encoder design. Our proposed design can be used in those devices which need powerful and advanced still image compression chip, such as the next generation HDR display, the digital still camera, the digital frame, the digital surveillance, the mobile phone, the camera and other digital photography applications.

6. References

B. Crow, Windows Media Photo: A new format for end-to-end digitalimaging, *Windows Hardware Engineering Conference*, 2006.

C.-H. Pan; C.-Y. Chien; W.-M. Chao; S.-C. Huang & L.-G. Chen, Architecture design of full HD JPEG XR encoder for digital photography applications, *IEEE Trans. Consu. Elec.*, Vol. 54, Issue 3, pp. 963-971, Aug. 2008.

C.-Y. Chien; S.-C. Huang; C.-H. Pan; C.-M. Fang & L.-G. Chen, Pipelined Arithmetic Encoder Design for Lossless JPEG XR Encoder, *IEEE Intl. Sympo. on Consu. Elec.*, Kyoto, Japan, May 2009.

D. D. Giusto & T. Onali. Data Compression for Digital Photography: Performance comparison between proprietary solutions and standards, *IEEE Conf. Consu. Elec.*, pp. 1-2, 2007.

D. Schonberg; S. Sun; G. J. Sullivan; S. Regunathan; Z. Zhou & S. Srinivasan, Techniques for enhancing JPEG XR / HD Photo rate-distortion performance for particular fidelity metrics, *Applications of Digital Image Processing XXXI, Proceedings of SPIE*, vol. 7073, Aug. 2008.

ISO/IEC JTC1/SC29/WG1. JPEG 2000 Part I Final Committee Draft, Rev. 1.0, Mar. 2000.

ITU. T.81 : Information technology - Digital compression and coding of continuous-tone still images. 1992.

L.V. Agostini; I.S. Silva & S. Bampi, Pipelined Entropy Coders for JPEG compression, *Integrated Circuits and System Design*, 2002.

S. Groder, Modeling and Synthesis of the HD Photo Compression Algorithm, Master Thesis, 2008.

S. Srinivasan; C. Tu; S. L. Regunathan & G. J. Sullivan, HD Photo: a new image coding technology for digital photography, *Applications of Digital Image Processing XXX, Proceedings of SPIE*, vol. 6696, Aug. 2007.

S. Srinivasan; Z. Zhou; G. J. Sullivan; R. Rossi; S. Regunathan; C. Tu & A. Roy, Coding of high dynamic range images in JPEG XR / HD Photo, *Applications of Digital Image Processing XXXI, Proceedings of SPIE*, vol. 7073, Aug. 2008.

Y.-W. Huang; B.-Y. Hsieh; T.-C. Chen & L.-G. Chen, Analysis, Fast Algorithm, and VLSI Architecture Design for H.264/AVC Intra Frame Coder, IEEE Trans. Circuits Syst. Video Technol., vol. 15, no. 3, pp. 378-401, Mar. 2005.

6

The Design of IP Cores in Finite Field for Error Correction

Ming-Haw Jing, Jian-Hong Chen, Yan-Haw Chen,
Zih-Heng Chen and Yaotsu Chang
I-Shou University
Taiwan, R.O.C.

1. Introduction

In recent studies, the bandwidth of communication channel, the reliability of information transferring, and the performance of data storing devices become the major design factors in digital transmission /storage systems. In consideration of those factors, there are many algorithms to detect or remove the noisefrom the communication channel and storage media, such as cyclic redundancy check (CRC) and errorcorrecting code (Peterson & Weldon, 1972; Wicker, 1995). The former, a hush function proposed by Peterson and Brown (Peterson & Brown, 1961), is utilized applied in the hard disk and network for error detection; the later is a type of channel coding algorithms recover the original data from the corrupted data against various failures. Normally, the scheme adds redundant code(s) to the original data to provide reliability functions such as error detection or error correction. The background of this chapter involves the mathematics of algebra, coding theory, and so on.

In terms of the design of reliable components by hardware and / or software implementations, a large proportion of finite filed operations is used in most related applications. Moreover, the frequently used finite field operations are usually simplified and reconstructed into the hardware modules for high-speed and efficient features to replace the slow software modules or huge look-up tables (a fast software computation). Therefore, we will introduce those common operations and some techniques for circuit simplification in this chapter. Those finite field operations are additions, multiplications, inversions, and constant multiplications, and the techniques include circuit simplification, resource-sharing methods, etc. Furthermore, the designers may use mathematical techniques such as group isomorphism and basis transformation to yield the minimum hardware complexities of those operations. And, it takes a great deal of time and effort to search the optimal designs. To solve this problem, we propose the computer-aided functions which can be used to analyze the hardware speed/complexity and then provide the optimal parameters for the IP design.

This chapter is organized as follows: In Section 2, the mathematical background of finite field operations is presented. The VLSI implementation of those operations is described in Section 3. Section 4 provides some techniques for simplification of VLSI design. The use of

computer-aided functions in choosing the suitable parameters is introduced in Section 5. Finally, the result and conclusion are given.

2. The mathematic background of finite field

Elements of a finite field are often expressed as a polynomial form over $GF(q)$, the characteristic of the field. In most computer related applications, the Galois field with characteristic 2 is wildly used because its ground field, $GF(2)$, can be mapped into bit-0 and bit-1 for digital computing. For convenience, the value within two parenthesises indicates that the coefficients for a polynomial in descending order. For example, the polynomial, $x^6+x^5+x^3+1$, is represented by {1101001} in binary form or {69} in hexadecimal form. So does an element $\alpha \in GF(2^m)$ is presented as symbol based polynomial.

2.1 The common base representations

2.1.1 The standard basis

If an element $\alpha \in GF(2^m)$ is the root of a degree m irreducible polynomial $f(x)$, i.e., $f(\alpha)=0$, then the set $\{1,\alpha^1,\alpha^2,\ldots,\alpha^{m-1}\}$ forms a basis, is called a standard basis, a polynomial basis or a canonical basis (Lidl & Niederreiter, 1986). For example, construct $E=GF(2^4)$ with the degree 4 irreducible polynomial $f(x)=x^4+x+1$, suppose $f(\alpha)=0$, that is, $\alpha^4=\alpha+1$ and $E=<\alpha>\cup\{0\}$ as Table 1.

element	α^3	α^2	α^1	α^0	element	α^3	α^2	α^1	α^0
0	0	0	0	0	α^7	0	1	1	1
α^0	0	0	0	1	α^8	1	1	1	0
α^1	0	0	1	0	α^9	0	1	0	1
α^2	0	1	0	0	α^{10}	1	0	1	0
α^3	1	0	0	0	α^{11}	1	1	0	1
α^4	1	0	0	1	α^{12}	0	0	1	1
α^5	1	0	1	1	α^{13}	0	1	1	0
α^6	1	1	1	1	α^{14}	1	1	0	0

Table 1. The standard basis expression for all elements of $E=GF(2^4)$

2.1.2 The normal basis

For a given $GF(2^m)$, there exists a normal basis $\{\alpha,\alpha^2,\alpha^{2^2},\ldots,\alpha^{2^{m-1}}\}$. Let $\beta=\sum_{i=0}^{m-1}b_i\alpha^{2^i}$ be represented in a normal basis, and the binary vector $(b_0,b_1,\ldots b_{m-1})$ is used to represent the coefficients of β, denoted by $\beta=(b_0,b_1,\ldots b_{m-1})$. Since $\alpha^{2^m}=1=\alpha^{2^0}$ by Fermat's little theorem (Wang et al., 1985), $\beta^2=b_{m-1}\alpha^{2^0}+b_0\alpha^{2^1}+\cdots+b_{m-2}\alpha^{2^{m-1}}=(b_{m-1},b_0,\ldots b_{m-2})$ or $\beta^{2^i}=(b_{m-i},b_{m-i+1},\ldots,b_{m-1},b_0,b_1,\ldots,b_{m-i-1})$. That is, the squaring operations (2^ith power operations) can be constructed by cyclic rotations in software or by changing lines in

hardware, which is with low complexity for practical applications (Fenn et al., 1996).

2.1.3 Composite field

For circuit design, using a composite field to execute some specific operations is an effective method, for example, the circuit of finite field inversion obtained in composite filed has the minimum complexity. The famous example is found in most hardware designs of AES VLSI (Hsiao et al., 2006; Jing et al., 2007), in which the S-box is a non-linear substitution for all elements in $GF(2^8)$ can be designed with a less area complexity by several isomorphism composite fields such as $GF((2^2)^4)$, $GF((2^4)^2)$, and $GF(((2^2)^2))$ (Morioka & Satoh, 2003). In this section, we introduce the process to construct a composite field and the basis transformation between a standard basis and a basis in composite field.

Let $GF(2^8)$ be represented in a standard basis with relation polynomial $f(x)=x^8+x^4+x^3+x^2+1$ ($f(x)$ is primitive) and $f(\alpha)=0$ such that $\langle\alpha\rangle=GF(2^8)$ and $\gamma=\alpha^r$ is a primitive element in the ground field $GF(2^4)$, where $r=(2^8-1)/(2^4-1)=17$. We construct the composite field $GF((2^4)^2)$ over the field $GF(2^4)$ using the irreducible polynomial $q(x)$ with degree 2 over $GF(2^4)$, which is given as follows

$$q(x)=(x+\alpha)(x+\alpha^{2^4})=x^2+(\alpha+\alpha^{2^4})x+\alpha^{17}=x^2+\alpha^{34}x+\alpha^{17}. \tag{1}$$

Such that $\gamma=\alpha^{17}$ is an element of $GF(2^2)$. In order to represent the elements of the ground field $GF(2^2)$, we use the term in $q(x)$ as the basis element, which is $\gamma=\alpha^{17}$. An element A is expressed in $GF((2^4)^2)$ as

$$A=a_0'+a_1'\alpha\ . \tag{2}$$

where $a_j'\in GF(2^4)$. We can express a_j' in $GF(2^4)$ using $\gamma=\alpha^{17}$ as the basis element

$$a_j'=\overline{a}_{j0}+\overline{a}_{j0}\gamma+\overline{a}_{j0}\gamma^2+\overline{a}_{j0}\gamma^3=\overline{a}_{j0}+\overline{a}_{j1}\alpha^{17}+\overline{a}_{j1}\alpha^{34}+\overline{a}_{j1}\alpha^{51}\ . \tag{3}$$

where $\overline{a}_{ji}\in GF(2)$ for $j=0,1$ and $i=0,1,2,3$. Therefore, the representation of A in the composite field is obtained as

$$A=a_0'+a_1'\alpha=(\overline{a}_{00}+\overline{a}_{01}\alpha^{17}+\overline{a}_{02}\alpha^{34}+\overline{a}_{03}\alpha^{51})+(\overline{a}_{10}\alpha+\overline{a}_{11}\alpha^{18}+\overline{a}_{12}\alpha^{35}+\overline{a}_{13}\alpha^{52})\ . \tag{4}$$

Next, substitute the terms α^{17i+j} for $j=0,1$ and $i=0,1,2,3$ by the relation polynomial $f(x)=x^8+x^4+x^3+x^2+1$ as follows:

$$\begin{aligned}&\alpha^{17}=\alpha^7+\alpha^4+\alpha^3,\ \ \alpha^{34}=\alpha^6+\alpha^3+\alpha^2+\alpha^1,\ \ \alpha^{51}=\alpha^3+\alpha^1,\\ &\alpha^{18}=\alpha^5+\alpha^3+\alpha^2+1,\ \ \alpha^{35}=\alpha^7+\alpha^4+\alpha^3+\alpha^2,\ \ \alpha^{52}=\alpha^4+\alpha^2\ .\end{aligned} \tag{5}$$

By substituting the above terms in expression Equation (4), we obtain the representation of

A in the standard basis $(1, \alpha, \alpha^1, \ldots \alpha^7)$ as

$$A = \alpha_0 + a_1\alpha_1 + a_2\alpha^2 + a_3\alpha^3 + a_4\alpha^4 + a_5\alpha^5 + a_6\alpha^6 + a_7\alpha^7 . \quad (6)$$

The relationship between the terms a_h for $h = 0,1,\ldots,7$ and $\overline{a}_{ji}$ for $j = 0,1$ and $i = 0,1,2,3$ determines a 8 by 8 conversion matrix T (Sunar et al., 2003). The first row of the matrix T is obtained by gathering the constant terms in the right hand side of Equation (4) after the substitution, which gives the constant coefficients in the left hand side, i.e., the term a_0. A simple inspection shows that $\alpha_0 = \overline{a}_{00} + \overline{a}_{11}$. Therefore, we obtain the 8×8 matrix T and this matrix gives the representation of an element in the binary field $GF(2^8)$ given its representation in the composite field $GF((2^4)^2)$ as follows:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \overline{a}_{00} \\ \overline{a}_{01} \\ \overline{a}_{02} \\ \overline{a}_{03} \\ \overline{a}_{10} \\ \overline{a}_{11} \\ \overline{a}_{12} \\ \overline{a}_{13} \end{bmatrix} . \quad (7)$$

The inverse transformation, i.e., the conversion from $GF(2^8)$ to $GF((2^4)^2)$, requires computing the T^{-1} matrix. We can use Gauss-Jordan Elimination to derive the T^{-1} matrix as follows:

$$\begin{bmatrix} \overline{a}_{00} \\ \overline{a}_{01} \\ \overline{a}_{02} \\ \overline{a}_{03} \\ \overline{a}_{10} \\ \overline{a}_{11} \\ \overline{a}_{12} \\ \overline{a}_{13} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} . \quad (8)$$

2.1.4 The basis transformation between standard basis and normal basis

The normal basis is with some good features in hardware, but the standard basis is used in popular designs. Finding the transformation between them is an important topic (Lu, 1997), we use $GF(2^4)$ as an example to illustrate that. Suppose $GF(2^4)$ is with the relation $p(x) = x^4 + x^3 + 1$ which is a primitive polynomial. Let $p(\alpha) = 0$ such that $B_1 = (\alpha^0, \alpha^1, \alpha^2, \alpha^3)$ form a standard basis. Let $\gamma = \alpha^3$ and the set $\{\gamma^1, \gamma^2, \gamma^4, \gamma^8\}$ is linear

independent such that $B_2 = (\gamma^1, \gamma^2, \gamma^4, \gamma^8)$ forms a normal basis. There exists a matrix T such that $B_2^T = T \times B_1^T$ and $B_1^T = T^{-1} \times B_2^T$. The matrixes T and T^{-1} are listed as follows.

$$T = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \quad T^{-1} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \gamma^8 \\ \gamma^4 \\ \gamma^2 \\ \gamma^1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha^3 \\ \alpha^2 \\ \alpha^1 \\ \alpha^0 \end{bmatrix}, \quad \begin{bmatrix} \alpha^3 \\ \alpha^2 \\ \alpha^1 \\ \alpha^0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \gamma^8 \\ \gamma^4 \\ \gamma^2 \\ \gamma^1 \end{bmatrix}. \tag{9}$$

2.2 The basic operation in finite field

2.2.1 Addition and subtraction

For a finite field with characteristic 2, addition and subtraction are performed by the bitwise XOR operator. For example, let $a(x) = x^4 + x^2 + x^1 + 1$, $b(x) = x^4 + x^3 + x^1 + 1$, and $c(x)$ be the summation of two polynomials, thus, $c(x) = a(x) + b(x) = 2x^4 + x^3 + x^2 + 2x^1 + 2 = x^3 + x^2$ or perform in binary form {10111} + {11011} = {01100}.

2.2.2 Multiplication and inversion

The multiplication in a finite field is performed by multiply two polynomials modulo a specific irreducible polynomial. For example, consider the finite field $E = GF(2^4)$ which is with the relation $p(x) = x^4 + x + 1$ and let $p(\alpha) = 0$ thus $(\alpha^0, \alpha^1, \alpha^2, \alpha^3)$ forms a standard basis. Suppose $a, b, c \in E$ and $a = \alpha^3 + 1$, $b = \alpha^2 + \alpha + 1$, and c is the product of them. Thus $c = a \times b = (\alpha^3 + 1) \times (\alpha^2 + \alpha + 1) = \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$, refer to Table 1, we have the product result as $c = (\alpha^2 + \alpha) + (\alpha + 1) + \alpha^3 + \alpha^2 + \alpha + 1 = \alpha^3 + \alpha$. For every nonzero element $\gamma \in E = GF(2^m)$, one has $\gamma^{2^n} = \gamma$ or $\gamma^{-1} = \gamma^{2^n - 2}$ equivalently (Dinh et al., 2001). Therefore, the division for finite field can be performed by the multiplicative inversion. For example, consider the inversion in $GF(2^8)$, $\gamma^{-1} = \gamma^{2^n - 2}$, and one can obtain this as Fig. 1.

2.2.3 Square operation

Consider an element $A = a_0 + a_1 x^1 + \cdots + a_{m-1} x^{m-1} \in E$ where $a_i \in GF(2)$ for $0 \le i < m$, the square operation for the characteristic 2 finite field is: $A^2 = (a_0 + a_1 x^1 + \cdots + a_{m-1} x^{m-1})^2$. For $a_i \in GF(2)$, we have $a_i^2 = a_i$ and thus $A^2 = a_0 + a_1 x^2 + \cdots + a_{m-1} x^{2(m-1)}$. Besides, those items with power not less m can be expressed by standard basis. Thus, we can perform the square operation by some finite field additions, i.e., XOR gates. For instance, let $E = GF(2^4)$ constructed by $f(x) = x^4 + x + 1$, an element $A = a_0 + a_1 x^1 + a_2 x^2 + a_3 x^3 \in E$, $A^2 = a_0 + a_1 x^2 + a_2 x^4 + a_3 x^6$. Two terms x^4 and x^6 can be substituted by $x + 1$ and $x^3 + x$ according to Table 1. We have $A^2 = a_0 x^0 + a_1 x^2 + a_2 (x + 1) + a_3 (x^3 + x)$ or $A^2 = (a_0 + a_2) + (a_2 + a_3) x + a_1 x^2 + a_3 x^3$. The same

property is also suitable for the power 2^i operation, such as $A^{2^0}, A^{2^1}, \ldots, A^{2^{m-1}}$.

3. The hardware designs for finite field operations

3.1 Multiplier

Finite field multiplier is the basic component for most applications. Many designers choose the one with standard basis for their applications, because the standard basis is easier to show the value by the bit-vector in digital computing. As follows, we introduce two most used types of finite field multipliers, one is the conventional multiplier and another is the bit-serial one.

3.1.1 Conventional multiplier

As the statement in Section 2.2.2, let $A, B, C \in GF(2^m)$ are represented with standard basis and $C = A \times B$, where $A = \sum_{i=0}^{m-1} a_i \alpha^i$, $B = \sum_{i=0}^{m-1} b_i \alpha^i$, and the product $P = \left(\sum_{i=0}^{m-1} a_i \alpha^i\right) \times \left(\sum_{i=0}^{m-1} b_i \alpha^i\right) = \sum_{i=0}^{2m-1} p_i \alpha^i$. Note that every element in $GF(2^m)$ is with the relation $f(x)$ described in Section 2.1.1, such that the terms with order greater than m, $\alpha^m, \alpha^{m+1}, \ldots, \alpha^{2m-1}$, can be substituted by the linear combination of standard basis $\{1, \alpha^1, \ldots, \alpha^{m-1}\}$. Thus, we can observe that there are m^2 and gate and about $m \times O(m)$ XOR gates in the substitution for high-order terms.

3.1.2 Massey-Omura multiplier

Here, we introduce the popular version named the bit-serial type of Massey-Omura multiplier. It is based on the normal basis, and the transformation between standard basis and normal basis is introduced in Section 2.1.4. Let $A, B, C \in GF(2^m)$ are represented with normal basis and $C = A \times B$, where $A = \sum_{i=0}^{m-1} a_i \alpha^{2^i}$, $B = \sum_{i=0}^{m-1} b_i \alpha^{2^i}$, and $C = \sum_{i=0}^{m-1} c_i \alpha^{2^i}$. Denote the coefficient-vector of A, B, and C by a, b, and c, and the notation $a^{(i)}$ means A^{2^i}, we have:

$$C = A \times B = \left[a_0, a_1, \ldots, a_{m-1}\right] \times \begin{bmatrix} \alpha^{2^0+2^0} & \alpha^{2^0+2^1} & \cdots & \alpha^{2^0+2^{m-1}} \\ \alpha^{2^1+2^0} & \alpha^{2^1+2^1} & \cdots & \alpha^{2^1+2^{m-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{2^{m-1}+2^0} & \alpha^{2^{m-1}+2^1} & \cdots & \alpha^{2^{m-1}+2^{m-1}} \end{bmatrix} \times \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{m-1} \end{bmatrix} = a \times M \times b^T, \tag{10}$$

where $M = M_0 \alpha + M_1 \alpha^2 + \cdots + M_{m-1} \alpha^{2^{m-1}}$, such that

$$c_{m-1-i} = a \times M_{m-1-i} \times b^T = a^{(i)} \times M_{m-1} \times \left(b^{(i)}\right)^T. \tag{11}$$

Using Equation (11), the bit-serial Massey-Omura multiplier can be designed as following:

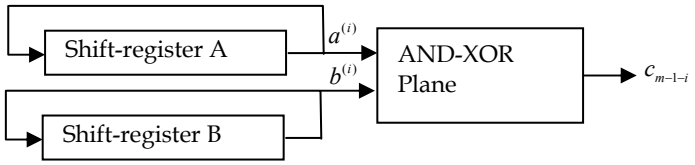


Fig. 1. The Massey-Omura bit-serial multiplier

In Fig. 1, the two shift-register perform the square operation in normal basis, and the complexity of and-xor plane is about $O(m)$ and relative to the number of nonzero element in M_{m-1-i}. Therefore, Massey-Omura multiplier is suitable to the design of area-limited circuits.

3.2 Inverse

In general the inverse circuit is usually with the biggest area and time complexity among other operations. There are two main methods to implement the finite field inverse, that is, multiplicative inversion and inversion based on composed field. The first method decomposes inversion by multiplier and squaring, and the optimal way for decomposing is proposed by Itoh and Tsujii (Itoh & Tsujii, 1988). The later one is based on the composed field and suited for area-limited circuits, which has been widely used in many applications.

3.2.1 Multiplicative inversion

From Fermat's theorem, for any nonzero element $\alpha \in GF(2^m)$ holds $\alpha^{2^m-1}=1$. Therefore, multiplicative inversion is equal to α^{2^m-2}. Based on this fact $\alpha^{-1}=\alpha^{2^m-2}=\prod_{i=1}^{m-1}\alpha^{2^i}$, Itoh and Tsujii reduced the number of required multiplications to $O(\log m)$, which is based on the decomposition of integer. Suppose $m-1=\sum_{n=0}^{b-1}a_n\times 2^n$, where $a_n \in GF(2)$ and $a_{b-1}=1$ denoted the decimal number $[1a_{b-2}\ldots a_1a_0]_2$, we have the following facts:

$$\begin{aligned}
2^{m-1}-1 &= (2^{2^{b-1}}-1)\cdot 2^{[a_{b-2}\ldots a_1a_0]_2}+2^{[a_{b-2}\ldots a_1a_0]_2}-1\\
&= (2^{2^{b-1}}-1)(2^{2^{b-2}}+1)\cdot 2^{[a_{b-2}\ldots a_1a_0]_2}+2^{[a_{b-2}\ldots a_1a_0]_2}-1\\
&= (2^{2^{b-2}}+1)\cdots(2^{2^2}+1)(2^{2^1}+1)\cdot 2^{[a_{b-2}\ldots a_1a_0]_2}+2^{[a_{b-2}\ldots a_1a_0]_2}-1
\end{aligned} \quad (12)$$

$$\begin{aligned}
2^{[a_{b-2}\ldots a_1a_0]}-1 &= a_{b-2}(2^{2^{b-2}}-1)2^{[a_{b-3}\ldots a_1a_0]_2}+2^{[a_{b-3}\ldots a_1a_0]_2}-1\\
&= a_{b-2}(2^{2^{b-2}}-1)\cdots(2^{2^2}+1)(2^{2^1}+1)2^{[a_{b-3}\ldots a_1a_0]_2}+2^{[a_{b-3}\ldots a_1a_0]_2}-1
\end{aligned} \quad (13)$$

$$\begin{aligned}
2^{m-1}-1 &= (2^{2^{b-2}}+1)\cdots(2^{2^2}+1)(2^{2^1}+1)\cdot 2^{[a_{b-2}\ldots a_1a_0]_2}+2^{[a_{b-2}\ldots a_1a_0]_2}-1\\
&= ((2^{2^{b-2}}+1)2^{2^{b-2}})(2^{2^{b-3}}+1)\cdots(2^{2^2}+1)(2^{2^0}+1)\cdot 2^{[a_{b-3}\ldots a_1a_0]_2}+\\
&\quad a_{b-2}(2^{2^{b-3}}+1)\cdots(2^{2^2}+1)(2^{2^0}+1)2^{[a_{b-3}\ldots a_1a_0]_2}+2^{[a_{b-3}\ldots a_1a_0]_2}-1\\
&= ((2^{2^{b-2}}+1)2^{2^{b-2}}+a_{b-2})(2^{2^{b-3}}+1)\cdots(2^{2^2}+1)(2^{2^0}+1)+2^{[a_{b-3}\ldots a_1a_0]_2}-1\\
&= (((\cdots((2^{2^{b-2}}+1)2^{2^{b-2}}+a_{b-2})(2^{2^{b-3}}+1)2^{2^{b-3}}+a_{b-3})\cdots)(2^{2^2}+1)2^{2^2}+a_2)\\
&\quad (2^{2^1}+1)2^{2^1}+a_1)(2^{2^0}+1)2^{2^0}+a_0
\end{aligned} \quad (14)$$

This algorithm requires $N_M = len.(m-1) + wt.(m-1) - 2$ multipliers, and $N_P = len.(m-1) + wt.(m-1) - 1$ square circuits, where $len.(m-1)$ the length of binary representation of $m-1$ and $wt.(m-1)$ is the number of nonzero bit in the representation. For instance, if $m=8$ then $m-1=7$, $N_M = len.(7) + wt.(7) - 2 = 3+3-2 = 4$ and $N_P = len.(7) + wt.(7) - 1 = 3+3-1 = 5$. For the latency of circuit, it takes $(\lceil \log_2(m-1) \rceil) T_M + (\lceil \log_2(m-1) \rceil + 1) T_S$, where T_M (resp. T_S) is the latency of multiplier (resp. squaring circuit). We list some results of this algorithm as Table 2.

m	area	latency	m	area	latency
5	2 NM +3 NP	2 TM +3 TP	11	4 NM +5 NP	4 TM +5 TP
6	3 NM +4 NP	3 TM +4 TP	12	5 NM +6 NP	4 TM +5 TP
7	3 NM +4 NP	3 TM +4 TP	13	4 NM +5 NP	4 TM +5 TP
8	4 NM +5 NP	3 TM +4 TP	14	5 NM +6 NP	4 TM +5 TP
9	3 NM +4 NP	3 TM +4 TP	15	5 NM +6 NP	4 TM +5 TP
10	4 NM +5 NP	4 TM +5 TP	16	6 NM +7 NP	4 TM +5 TP

Table 2. The list of Itoh and Tsujii algorithm

3.2.2 Composite field inversion

The use of composite field provides an isomorphism for $GF(2^m)$, while m is not prime. Especially, if m is even, then inverse using composite field is with very low complexity. Consider the inverse in $GF((2^{m/2})^2)$ where m is even. Suppose $A, B \in GF((2^{m/2})^2)$ constructed by an irreducible polynomial $P(x) = p_1 x + p_0$, where $p_0, p_1 \in GF(2^{m/2})$. Let $A = a_1 x + a_0$ and $B = b_1 x + b_0$, where $a_1, a_0, b_1, b_0, p_1, p_0 \in GF(2^{m/2})$. Assume that B is the inverse of A, thus $A \times B = 1$ or $(a_1 x + a_0) \times (b_1 x + b_0) \equiv 1$ modulo $P(x)$. After the distribution, one has $A \times B = (a_1 b_1 p_1 + a_1 b_0 + a_0 b_1) x + (a_1 b_1 p_0 + a_0 b_0) = 1$. Therefore, $a_1 b_1 p_1 + a_1 b_0 + a_0 b_1 = 0$ and $a_1 b_1 p_0 + a_0 b_0 = 1$. Let $\Delta = (a_0^2 + a_0 a_1 p_1 + p_0 a_1^2)$, one has $b_1 = a_1 \Delta^{-1}$ and $b_0 = (a_0 + a_1 p_1)\Delta^{-1}$, which is design as Fig. 2. Obviously, one can observe the inversion in $GF(2^m)$ is executed by several operations which are all in $GF((2^{m/2})^2)$, thus the total gated count used can be reduced.

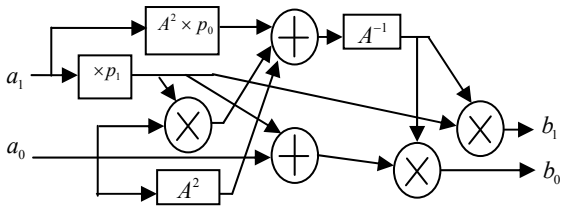


Fig. 2. The circuit for composite field inversion

4. Some techniques for simplification of VLSI

4.1 Finding common sharing resource in various design levels

Sharing resource is a common method to reduce the area cost. This skill can be used in different design stages. For example, consider the basis transformation in Section 2.1.4, the element of normal basis is obtained by the linear combination of standard basis as follows:

$$\gamma^8 = \alpha^1 + \alpha^0,\ \gamma^4 = \alpha^2 + \alpha^0,\ \gamma^2 = \alpha^2 + \alpha^1 + \alpha^0,\ \gamma^1 = \alpha^3 + \alpha^2 + \alpha^1 + \alpha^0. \quad (15)$$

It takes 7 XOR gates for the straightforward implementation. However, if one calculate the summation $t = \alpha^2 + \alpha^1 + \alpha^0$ firstly, then $\gamma^2 = t$ and $\gamma^1 = \alpha^3 + t$. Therefore, the number of XOR gates is reduced to 5. Although it is effective in the bit-level, this idea is also effective in other design stages. Consider another example in previous section, when we form those components $\Delta = (a_0^2 + a_0 a_1 p_1 + p_0 a_1^2)$ and $b_0 = (a_0 + a_1 p_1)\Delta^{-1}$, it takes 3 2-input adders in two expressions. Suppose we form the component $a_0 + a_1 p_1$ firstly, thus the number of 2-input adder is reduced from 3 to 2 ($\Delta = (a_0(a_0 + a_1 p_1) + p_0 a_1^2)$). Therefore, the resource-sharing idea is suitable to different design stages.

4.2 Finding the optimal parameters of components

Another technique used to simplify circuits for finite field operations is change the original field to another isomorphism. Although these methods are equal in mathematics, it provides different outcomes in VLSI designs. There are two main methods to be realized.

4.2.1 Change the relation polynomial

Consider the implementations of hardware multiplier/inverse in $GF(2^8)$ using FPGA, we gather area statistics of multiplier/inverse by using different irreducible polynomials ($f(x)$) and draw the line chart as Fig. 3 and Fig. 4, where the X axis indicates various irreducible polynomials in decimal representation and the Y axis is the number of needed XOR gates. In Fig. 3, one can observe the lowest complexity of area and delay is with $f(x)$ is 45. The maximum difference of XOR number (resp. delay) between two polynomials is 50 (resp. 2). Therefore, choosing the optimal parameters has great influence in complexity in VLSI. The same phenomenon is also been observed in Fig. 4, the maximum difference is 196 XOR gates.

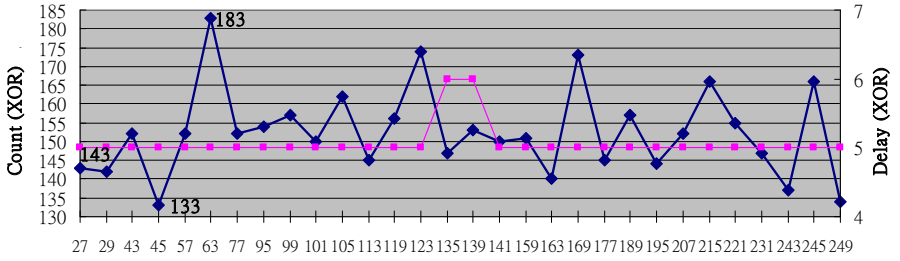


Fig. 3. The statistic of area for multiplier v.s. $f(x)$

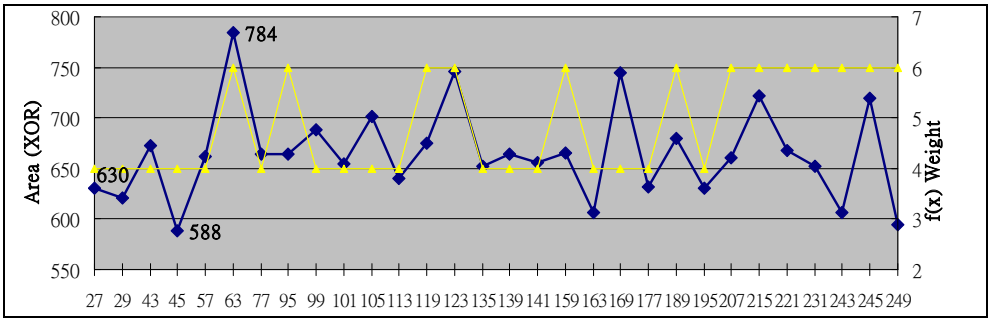


Fig. 4. The statistic of area for inverse v.s. $f(x)$

4.2.2 Using composite field

In Section 2.1.3, we illustrate the transformation between a finite field represented by standard basis and a composite field. The most applications for composite field are to design the inverse, for instance, the S-box in AES algorithm (Morioka & Satoh, 2003). As we know, the main component in S-box is the finite field inverse of $GF(2^8)$. Here, we implement the S-box by the multiplicative inversion described in Section 3.2.1 and by using composite field $GF((2^4)^2)$ described in 3.2.2 as Table 3 by using the Altera FPGA Stratix 2S1020C4 device. Obviously, the later method is with more advantages for both area and time complexity than that of previous one.

Method	LE/ALUT	Delay (ns)	CLK (MHz)	Throughput (MHz)
mult. inverse	210	23.240	43.029	344.232
composite field	82	20.219	49.458	395.664

Table 3. The results for S-box using multiplicative inversion and using composite field

5. Using computer-aided functions to choose suitable parameters

According to the explanations in Section 4, we can realize the related VLSI IPs using various parameters to bring the benefits for lower area or time complexity. However, there exist so many isomorphisms in using finite filed, it seems that there are so many procedures and variations to choose the parameters and hard to find a better ones. As a result, our group developed a software tools which is the computer-aided design (CAD) to help engineers to do the tedious analysis and search. This section will introduce the methods to apply the isomorphism transformations between $GF(2^8)$ and $GF((2^4)^2)$ illustrated in Section 4.1 and 4.2 step by step.

Firstly, list all irreducible and primitive polynomials in two fields as shown in Table 4 and 5, respectively. In this table, all irreducible and primitive polynomials are represented in hexadecimal form and we omit the most significant bit. For example, in Table 4, one chooses 1B that means $(00011011)_2$ or $x^8+x^4+x^3+x+1$; in Table 5, suppose the primitive element $\omega \in GF(2^4)$, one chooses 18 that means $(00011000)_2$ or $x^2+1x+\omega^3$.

$GF(2^8)$	irreducible polynomials															
#=30	1B	1D	2B	2D	39	3F	4D	5F	63	65	69	71	77	7B	87	8B
	8D	9F	A3	A9	B1	BD	C3	CF	D7	DD	E7	F3	F5	F9		
	primitive polynomials															
#=16	1D	2B	2D	4D	5F	63	65	69	71	87	8D	A9	C3	CF	E7	F5

Table 4. The irreducible and primitive polynomials in $GF(2^8)$

$GF((2^4)^2)$	irreducible polynomials															
#=120	18	19	1A	1B	1C	1D	1E	1F	21	22	25	26	29	2A	2D	2E
	31	33	34	36	39	3B	3C	3E	41	42	44	47	48	4B	4D	4E
	51	53	55	57	59	5B	5D	5F	62	63	64	65	6A	6B	6C	6D
	72	73	74	75	78	79	7E	7F	81	83	84	86	88	8A	8D	8F
	92	93	96	97	9A	9B	9E	9F	A1	A2	A4	A7	A9	AA	AC	AF
	B4	B5	B6	B7	BC	BD	BE	BF	C1	C3	C5	C7	C8	CA	CC	CE
	D4	D5	D6	D7	D8	D9	DA	DB	E2	E3	E6	E7	E8	E9	EC	ED
	F1	F2	F5	F6	F8	FB	FC	FF								
	primitive polynomials															
#=60	19	1B	1D	1E	22	25	29	2D	2E	33	34	39	3B	3E	42	44
	55	59	5B	5D	62	63	64	65	6B	6D	72	73	74	75	79	7E
	83	84	8D	92	93	9B	9E	A2	A4	A9	B4	B5	BD	BE	C3	C5
	CE	D4	D5	D9	DB	E2	E3	E9	ED	F2	F5	FB				

Table 5. The irreducible and primitive polynomials in $GF((2^4)^2)$

Secondly, the CAD searches for all possible combinations by the proposed algorithm as shown in Table 6. This algorithm regards as a function used to find transformation matrices as shown in Table 7. After we gather all results, we can choose the better parameters from the list of analyzed results for hardware design of new IP.

Chose relation polynomial 1D for $GF(2^8)$ ➔ $p(x)=x^8+x^4+x^3+x^2+1$. Let $p(\alpha)=0$, such that $\forall\beta\in GF(2^8)$ can be expressed by binary form as $(\alpha_7,\alpha_6,\alpha_5,\alpha_4,\alpha_3,\alpha_2,\alpha_1,\alpha_0)$	
Step 1:	Find a $GF((2^4)^2)$ irreducible polynomial. Select an irreducible polynomial in ground field $GF(2^4)$ is $f_1(x)=x^4+x+1$. Let ω be the root of $f_1(x)$, thus $f_1(\omega)=\omega^4+\omega+1=0$. Select an irreducible polynomial in $GF((2^4)^2)$ is 18➔ $f_2(x)=x^2+x+\omega^3$ and let γ be the root of $f_2(x)$, $f_2(\gamma)=\gamma^2+\gamma+\omega^3=0$.
Step 2:	Assume a generator σ in $GF((2^4)^2)$ and generate all none-zero elements of $GF((2^4)^2)$. For any element in $GF((2^4)^2)$ can be expressed in binary form as $(\gamma_{13},\gamma_{12},\gamma_{11},\gamma_{10},\gamma_{03},\gamma_{01},\gamma_{02},\gamma_{03})$. Assume the T matrix = $\left[(\sigma^7)^T\ (\sigma^6)^T\ \ldots\ (\sigma^0)^T\right]_{8\times8}$, we have

	$\begin{bmatrix}\gamma_{13}\\ \gamma_{12}\\ \gamma_{11}\\ \gamma_{10}\\ \gamma_{03}\\ \gamma_{02}\\ \gamma_{01}\\ \gamma_{00}\end{bmatrix}_{8\times1} = \begin{bmatrix}(\sigma^7)^T & (\sigma^6)^T & \dots & (\sigma^0)^T\end{bmatrix}_{8\times8} \begin{bmatrix}\alpha_7\\ \alpha_6\\ \alpha_5\\ \alpha_4\\ \alpha_3\\ \alpha_2\\ \alpha_1\\ \alpha_0\end{bmatrix}_{8\times1}$
Step 3:	Compute the T^{-1} matrix= $\begin{bmatrix}(\sigma^7)^T & (\sigma^6)^T & \dots & (\sigma^0)^T\end{bmatrix}^{-1}_{8\times8}$. Put all none-zero elements $GF((2^4)^2)$ in the following equation and check if they all hold, i.e., $[\alpha^i]^T = T^{-1}[\sigma^i]^T$, *where* $0 \le i \le 254$.
Step 4:	If Step 3 does not hold, return to Step 2 and choose another generator; If Step 3 holds, then the T and T^{-1} matrices are found.

Table 6. The proposed algorithm for searching transformation matrices

input	$p(x) = x^8 + x^4 + x^3 + x^2 + 1 \qquad p(x) \Rightarrow GF(2^8)$ $f_1(x) = x^4 + x + 1 \qquad f_1(\omega) = \omega^4 + \omega + 1 = 0$ $f_2(x) = x^2 + x + \omega^3 \qquad f_1(x), f_2(x) \Rightarrow GF((2^4)^2)$
output	$T = \begin{bmatrix}0&0&1&0&0&0&0&0\\ 0&1&1&0&0&1&0&0\\ 0&0&0&1&1&0&1&0\\ 1&0&0&1&0&0&0&0\\ 0&1&1&1&1&0&0&0\\ 0&1&0&1&0&1&0&0\\ 1&1&0&0&1&1&0&0\\ 1&1&0&1&1&1&1&1\end{bmatrix}_{8\times8}, T^{-1} = \begin{bmatrix}1&1&0&1&0&1&0&0\\ 0&1&0&1&1&0&1&0\\ 1&0&0&0&0&0&0&0\\ 1&1&0&0&0&1&0&0\\ 0&0&0&1&0&1&1&0\\ 1&0&0&1&1&0&1&0\\ 1&1&1&1&0&0&1&0\\ 0&0&1&1&0&1&0&1\end{bmatrix}_{8\times8}$

Table 7. The result of transformation matrices between $GF(2^8)$ and $GF((2^4)^2)$.

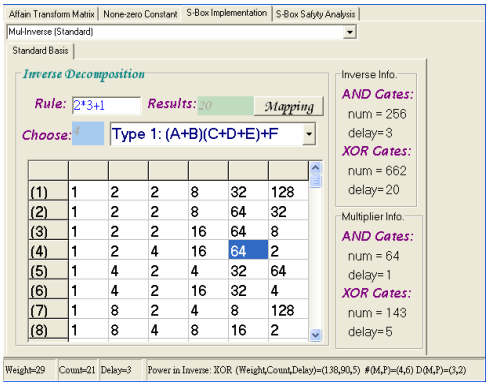


Fig. 5. The interface of CAD tool

Because various parameters provide VLSI's outputs with huge variation and it's seem impossible to run all parameters, we should provide engineers a CAD tool to obtain the analyzed algorithm and results. In Fig. 5, a designer can use a CAD with Windows interface to find better parameters of S-box. In this CAD, it provides the complexity information of the multipliers or the inverse in $GF(2^8)$. In this figure, designer chooses the fourth result, and the estimative complexity of inverse is shown in the top right of the figure; that the choice of multiplier is shown under the inverse information. Therefore, the CAD tool helps designer to choose the better parameters efficiently.

6. Summary

In this chapter, we introduce the common concepts of finite field regarding its applications in error correcting coding, cryptography, and others, including the mathematical background, some important designs for multiplier and inversion, and the idea to utilize the computer-aided functions to find better parameters for IP or system design. The summary of this chapter is as follows:

1. Introducing the basic finite field operations and their hardware designs: Those common operations include addition, multiplication, squaring, inversion, basis transformation, and so on. The VLSI designs of those operations may be understood through the mathematical background provided in Section 2. From the mathematical background, one should realize the benefits and the processes of the transformation between two isomorphic finite fields.
2. Using some techniques to simplify the circuits: We have introduced some useful techniques to reduce the area cost in VLSI design, such as the resource-sharing method, utilization of different parameters, or use some isomorphic field to substitute the used field. The first technique is widely used in various design stages. The later two techniques depend on the parameters used. Different parameters lead to different hardware implementation results. However, it seems infeasible to analyze all possible parameters manually.
3. Using the composite field inversion: Composite field inversion is used in the finite field inversion due to its superiority in hardware implementation. The main idea is to consider the use of intermediate fields and decompose the inversion on the original field into several operations on smaller fields. This method has been used in the AES S-box design to minimize the area cost.
4. Calculating the transformation matrices between isomorphic finite fields. It is well known that finite fields of the same order are isomorphic, and this implies the existence of transformation matrices. Finding the optimal one is important in the investigation of the VLSI designs. Two methods are presented; one is to change the relation polynomial, and the other is to use the composite field. An algorithm to calculate the transformation matrices is provided in Section 5, and it can be used to find the optimal one.
5. Using the computer-aided design to search for better parameters: A good hardware CAD tool provides fast search and enough information for designer, because it brings fast and accurate designs. In Section 5, the computer-aided function based on the proposed algorithms is one of the examples. When the order of the finite field gets large, the number of isomorphic field increases rapidly. This makes it almost impossible to do the exhausting search, and the proposed CAD can be used to support engineers to get the best choices.

7. Conclusion

In this chapter, we use the concept of composite fields for the CAD designs, which can support the VLSI designer to calculate the optimal parameters for finite field inversion.

8. Acknowledgments

This work is supported in part by the Nation Science Council, Taiwan, under grant NSC 96-2623-7-214-001.

9. References

Dinh, A.V.; Palmer, R.J.; Bolton, R.J. & Mason, R. (2001). A low latency architecture for computing multiplicative inverses and divisions in GF(2^m). *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 48, No. 8, pp. 789-793, ISSN: 1057-7130

Fenn, S.T.J.; Benaissa, M. & Taylor, D. (1996). Fast normal basis inversion in GF(2^m). *Electronics Letters*, Vol. 32, No. 17, pp. 1566-1567, ISSN: 0013-5194

Hsiao, S.-F.; Chen, M.-C. Chen & Tu, C.-S. (2006). Memory-free low-cost designs of advanced encryption standard using common subexpression elimination for subfunctions in transformations. *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 53, No. 3, pp. 615–626, ISSN: 1549-8328

Itoh, T. & Tsujii, S. (1988). A fast algorithm for computing multiplicative inverses in GF(2^m) using normal basis. *Information and Computing*, Vol. 78, No. 3, pp. 171-177, ISSN: 0890-5401

Jing, M.-H.; Chen, Z.-H.; Chen, J.-H. & Chen, Y.-H. (2007). Reconfigurable system for high-speed and diversified AES using FPGA. *Microprocessors and Microsystems*, Vol. 31, No. 2, pp. 94-102, ISSN: 0141-9331

Lidl, R. & Niederreiter, H. (1986). *Introduction to finite fields and their applications*, Cambridge University Press, ISBN: 9780521460941

Lu, C.-C. (1997). A search of minimal key functions for normal basis multipliers. *IEEE Transactions on Computers*, Vol. 46, No. 5, pp.588–592, ISSN: 0018-9340

Morioka, S. & Satoh, A. (2003). An optimized S-box circuit architecture for low power AES design. *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, *Lecture Notes in Computer Science*, Vol. 2523, pp. 172–186, ISBN: 3-540-00409-2, August, 2002, Redwood Shores, California, USA

Peterson, W.W. & Brown, D.T. (1961). Cyclic Codes for Error Detection, *Proceedings of the IRE,* Vol. 49, No. 1, pp. 228-235, ISSN: 0096-8390

Peterson, W.W. & Weldon, E.J. (1972). *Error-Correcting Codes*, The MIT Press, 2 edition, Cambridge, MA, ISBN: 3540004092

Sunar, B.; Savas, E. & Koc, C.K., (2003). Constructing composite field representations for efficient conversion. *IEEE Transactions on Computer*, Vol. 52, No. 11, pp. 1391-1398, ISSN: 0018-9340

Wang, C.C.; Truong, T.K.; Shao, H.M.; Deutsch, L.J.; Omura, J.K.; & Reed, I.S. (1985). VLSI architecture for computing multiplications and inverses in GF(2^m). *IEEE Transactions on Computers*, Vol. 34, No. 8, pp. 709-716, ISSN: 0018-9340

Wicker, S.B. (1995). *Error Control Systems for Digital Communication and Storage*, Prentice Hall, ISBN: 0-13-308941-X, US

7

Scalable and Systolic Gaussian Normal Basis Multipliers over GF(2^m) Using Hankel Matrix-Vector Representation

Chiou-Yng Lee
Lunghwa University of Science and Technology
Taoyuan County, Taiwan

1. Introduction

Efficient design and implementation of finite field multipliers have received high attention in recent years because of their applications in elliptic curve cryptography (ECC) and error control coding (Denning, 1983; Rhee, 1994; Menezes, Oorschot & Vanstone, 1997). Although channel codes and cryptographic algorithms both make use of the finite field GF(2^m), the field orders needed differ dramatically: channel codes are typically restricted to arithmetic with field elements which are represented by up to eight bits, whereas ECC rely on field sizes of several hundred bits. The majority of publications concentrate on finite field architectures for relatively small fields suitable for implementation of channel codes. In finite field GF(2^m), multiplication is one of the most important and time-consuming computations. Since cryptographic applications (Menezes, Oorschot & Vanstone, 1997) are the Diffie-Hellman key exchange algorithm based on the discrete exponentiation over GF(2^m), the methods of computing exponentiation over GF(2^m) based on Fermat's theorem are performed by the repeated multiply-square algorithm. Therefore, to provide the high performance of the security function, the efficient design of high-speed algorithms and hardware architectures for computing multiplication is required and considered.
There are three popular basis representations, termed polynomial basis (PB), normal basis (NB), and dual basis (DB). Each basis representation has its own advantages. The normal basis multiplication is generally selected for cryptography applications, because the squaring of the element in GF(2^m) is simply the right cyclic shift of its coordinates. NB multiplication depended the selection of key function is discovered by Massey and Omura (1986). For the elliptic curve digital signature algorithm (ECDSA) in IEEE Standard P1363 (2000) and National Institute of Standards and Technology (NIST) (2000), Gaussian normal basis (GNB) is defined to implement the field arithmetic operation. The GNB is a special class of normal basis, which exists for every positive integer m not divisible by eight. The GNB for GF(2^m) is determined by an integer t, and is called the type-t Gaussian normal basis. However, the complexity of a type-t GNB multiplier is proportional to t (Reyhani-Masoleh, 2006), small values of t are generally chosen to ensure that the field multiplication is implemented efficiently.

Among various finite field multipliers are classified either as a parallel or serial architectures. Bit-serial multipliers (Reyhani-Masoleh & Hasan, 2005; Lee & Chang, 2004) require less area, but are slow that is taken by m clock cycles to carry out the multiplication of two elements. Conversely, bit-parallel multipliers (Lee, Lu & Lee, 2001; Hasan, Wang & Bhargava, 1993; Kwon, 2003; Lee & Chiou, 2005) tend to be faster, but have higher hardware costs. Recently, various multipliers (Lee[1], 2003; Lee, Horng & Jou, 2005; Lee, 2005; Lee[2], 2003) focus on bit-parallel architectures with optimal gate count. However, previously mentioned bit-parallel multipliers show a computational complexity of $O(m^2)$ operations in GF(2); canonical and dual basis architectures are lower bounded by m^2 multiplications and $k^2 - 1$ additions, normal basis ones by k^2 multiplications and $2k^2 - k$ additions. A multiplication in GF(2) can be realized by a two-input AND gate and an adder by a two-input XOR gate. For this reason, it is attractive to provide architectures with low computational complexity for efficient hardware implementations.

There have digit-serial/scalable multiplier architectures to enhance the trade-off between throughput performance and hardware complexity. The scalable architecture needs both the element A and B are separated into $n=[m/d]$ sub-word data, while the digit-serial architecture only requires one of the element to separate sub-word data. Both architectures are used by the feature of the scalability to handle growing amounts of work in a graceful manner, or to be readily enlarged. In (Tenca & Koc, 1999), a unit is considered scalable defined that the unit can be reused or replicated in order to generate long-precision results independently of the data path precision for which the unit was originally designed. Various digit-serial multipliers are recently developed in (Paar, Fleischmann & Soria-Rodriguez, 1999; Kim & Yoo, 2005; Kim, Hong and Kwon, 2005; Guo & Wang, 1998; Song & Parhi, 1998; Reyhani-Masoleh & Hasan, 2002). Song and Parhi (1998) proposed MSD-first and LSD-first digit-serial PB multipliers using Horner's rule scheme. For partitioning the structure of two-dimension arrays, efficient digit-serial PB multipliers are found in (Kim & Yoo, 2005; Kim, Hong & Kwon, 2005; Guo & Wang, 1998). The major feature of these architectures is combined with both serial and parallel algorithms.

For large word lengths commonly found in cryptography, the bit-serial approach is rather slow, while bit-parallel realization requires large circuit area and power consumption. In elliptic curve cryptosystems, it strongly depends on the implementation of finite field arithmetic. By employing the Hankel matrix-vector representation, the new GNB multiplication algorithm over $GF(2^m)$ is presented. Utilizing the basic characteristics of MSD-first and LSD-first schemes (Song & Parhi, 1998), it is shown that the proposed GNB multiplication can be decomposed into $n(n+1)$ Hankel matrix-vector multiplications. The proposed scalable GNB multipliers are including one $d \times d$ Hankel multiplier, two registers and one final reduction polynomial circuit. The results reveal that, if the selected digital size is $d \geq 4$ bits, the proposed architecture has less time-space complexity than traditional digit-serial systolic multipliers, and can thus obtain an optimum architecture for GNB multiplier over $GF(2^m)$. To further saving both time and space complexities, the proposed scalar multiplication algorithm with Hankel matrix-vector representation can also be realized by a scalable and systolic architecture for polynomial basis and dual basis of $GF(2^m)$.

The rest of this paper is structured as follows. Section 2 briefly reviews a conventional NB multiplication algorithm and a Hankel matrix-vector multiplication. Section 3 proposes the two GNB multipliers, based on Hankel matrix-vector representation, to yield a scalable and systolic architecture. Section 4 introduces the modified GNB multiplier. Section 5 analyzes

our proposed GNB multiplier in the term of the time-area complexity. Finally, conclusions are drawn in Section 6.

2. Preliminaries

2.1 Gaussian normal basis multiplication

The finite field GF(2^m) is well known to be viewable as a vector space of dimension m over GF(2). A set $N = \{\alpha, \alpha^2, \cdots, \alpha^{2^{m-1}}\}$ is called the normal basis of GF(2^m), and α is called the normal element of GF(2^m). Let any element $A \in$GF(2^m) can be represented as

$$A = \sum_{i=0}^{m-1} a_i \alpha^{2^i} = (a_0, a_1, \cdots, a_{m-1}) \tag{1}$$

where $a_i \in$GF(2), $0 \le i \le m-1$, denotes the ith coordinate of A. In hardware implementation, the squaring element is performed by a cyclic shift of its binary representation. The multiplication of elements in GF(2^m) is uniquely determined by the m cross products $\alpha\alpha^{2^i} = \sum_{j=0}^{m-1} \mu_{ij}\alpha^{2^j}$, $\mu_{ij} \in$GF(2). $\mathbf{M}=\{\mu_{ij}\}$ is called a multiplication matrix. Let $A=(a_0, a_1, \ldots, a_{m-1})$ and $B=(b_0, b_1, \ldots, b_{m-1})$ indicate two normal basis elements in GF(2^m), and $C=(c_0, c_1, \ldots, c_{m-1}) \in$GF($2^m$) represent their product, i.e., $C=AB$. Coordinate ci of C can then be represented by

$$c_i = A^{(i)}\mathbf{M}(B^{(i)})^T \tag{2}$$

where $A^{(i)}$ denotes a right cyclic shift of the element A by i positions. To compute the multiplication matrix **M**, one can see in (IEEE Standard P1363, 2000; Reyhani-Masoleh & Hasan, 2003). When the multiplication matrix M is found, the NB multiplication algorithm is described as follows:

Algorithm 1: (NB multiplication) (IEEE Standard P1363, 2000)
Input: $A=(a_0, a_1, \ldots, a_{m-1})$ and $B=(b_0, b_1, \ldots, b_{m-1}) \in$GF($2^m$)
Output: $C=(c_0, c_1, \ldots, c_{m-1})=AB$
1. initial: $C=0$
2. for $i = 0$ to $m - 1$ {
3. $c_i = A\mathbf{M}B^T$
4. $A=A^{(1)}$ and $B=B^{(1)}$
5. }
6. output $C=(c_0, c_1, \ldots, c_{m-1})$

Applying Algorithm 1, Massey and Omura (1986) first proposed bit-serial NB multiplier. The complexity of the normal basis N, represented by C_N, is the number of nonzero μ_{ij} values in **M**, and determines the gate count and time delay of the NB multiplier. It is shown in (Mullin, Onyszchuk, Vanstone & Wilson, 1988/1989) that C_N for any normal basis of GF(2^m) is greater or equal to $2m-1$. In order to an efficient and simple implementation, a normal basis is chosen that C_N is as small as possible. Two types of an optimal normal basis (ONB), type-1 and type-2, exist in GF(2^m) if $C_N = 2m-1$. However, such ONBs do not exist for all m.

Definition 1. Let $p=mt+1$ represent a prime number and $gcd(mt/k,m)=1$, where k denotes the multiplicative order of 2 module p. Let γ be a primitive p root of unity. The type-t Gaussian

normal basis (GNB) is employed by $\alpha=\gamma+\gamma^{2^m}+\gamma^{2^{2m}}+...+\gamma^{2^{(t-1)m}}$ to generate a normal basis N for GF(2^m) over GF(2).

Significantly, GNBs exist for GF(2^m) whenever m is not divisible by 8. By adopting Definition 1, each element A of GF(2^m) can also be given as

$$\begin{aligned} A &= a_0\alpha + a_1\alpha^2 + \cdots + a_{m-1}\alpha^{2^{m-1}} = a_0(\gamma + \gamma^{2^m} + \cdots + \gamma^{2^{m(t-1)}}) \\ &+ a_1(\gamma^2 + \gamma^{2^{m+1}} + \cdots + \gamma^{2^{m(t-1)+1}}) + \cdots + a_{m-1}(\gamma^{2^{m-1}} + \gamma^{2^{2m-1}} + \cdots + \gamma^{2^{mt-1}}) \end{aligned} \tag{3}$$

From Equation (3), the type-t GNB can be represented by the set $\{\gamma,\gamma^2,\cdots,\gamma^{2^{m-1}},\gamma^{2^m},\gamma^{2^{m+1}},\cdots,\gamma^{2^{2m-1}},\cdots,\gamma^{2^{m(t-1)}},\gamma^{2^{m(t-1)+1}},\cdots,\gamma^{2^{mt-1}}\}$. Since γ is a primitive p root of unity, we have

$$\gamma\gamma^i = \begin{cases} \gamma^{i+1} & if \quad i \neq p-1 \\ 1 & if \quad i = p-1 \end{cases}. \tag{4}$$

Thus, the GNB can then alternate to represent the redundant basis $R=\{\gamma, \gamma^2,\ldots, \gamma^{p-1}\}$. The field element A can be alternated to define the following formula:

$$A = a_{F(1)}\gamma + a_{F(2)}\gamma^2 + \cdots + a_{F(p-1)}\gamma^{p-1} \tag{5}$$

where

$F(2^i 2^{mj} \bmod p)=i$ for $0 \le i \le m-1$ and $0 \le j \le t-1$.

Example 1. For example, Let $A=(a_0,a_1,a_2,a_3,a_4)$ be the NB element of GF(2^5), and let $\alpha=\gamma+\gamma^{10}$ be used to generate the NB. Applying the redundant representation, the field element A can be represented by $A=a_0\gamma+a_1\gamma^2+a_3\gamma^3+a_2\gamma^4+a_4\gamma^5+a_4\gamma^6+a_2\gamma^7+a_3\gamma^8+a_1\gamma^9+a_0\gamma^{10}$.

Observing this representation, the coefficients of A are duplicated by t-term coefficients of the original normal basis element $A=(a_0, a_1,\ldots, a_{m-1})$ if the field element A presents a type-t normal basis of GF(2^m). Thus, by using the function $F(2^i 2^{mj} \bmod p)=i$, the field element $A=(a_{F(1)}, a_{F(2)},\cdots, a_{F(p-1)})$ with the redundant representation can be translated into the following representation, $A=(\underbrace{a_0,\cdots,a_0}_{t},a_1,\cdots a_1,a_2,\cdots,a_2,\cdots\cdots a_{m-1},\cdots,a_{m-1})$. Therefore, the redundant basis is easy converted into the normal basis element, and is without extra hardware implementations.

Let $A=(a_0, a_1,\ldots, a_{m-1})$ and $B=(b_0, b_1,\ldots, b_{m-1})$ indicate two normal basis elements in GF(2^m), and $C=(c_0, c_1,\ldots, c_{m-1})$ represent their product, i.e., $C=AB$. Coordinate ci of C can then be calculated as in the following formula: (IEEE Standard P1363, 2000)

$$c_0 = G(A,B) = \sum_{j=1}^{p-2} a_{F(j+1)} b_{F(p-j)} \tag{6}$$

According to Algorithm 1, the GNB multiplication algorithm for even t is modified as follows.

Algorithm 2: (GNB multiplication for t even) (IEEE Standard P1363, 2000)

Input: $A=(a_0,a_1,\ldots,a_{m-1})$ and $B=(b_0,b_1,\ldots,b_{m-1})\in$GF($2^m$)

Output: $C=(c_0,c_1,\ldots,c_{m-1})=AB$

1. initial : $C=0$
2. for $k=0$ to $m-1$ {
3. $c_k= G(A,B)$
4. $A=A^{(1)}$ and $B=B^{(1)}$

5. }
6. Output $C=(c_0,c_1,\ldots,c_{m-1})$

2.2 Bit-parallel systolic Hankel multiplier

This section briefly introduces a bit-parallel systolic Hankel multiplier in (Lee & Chiou, 2005).

Definition 2. An $m\times m$ matrix H is known as the Hankel matrix if it satisfies the relation $H(i,j)=H(i+1,j-1)$, for $0 \le i \le m-2$ and $1 \le j < m-1$, where $H(i,j)$ represents the element in the intersection of row i and column j, such that

$$\mathbf{H}=\begin{bmatrix} h_0 & h_1 & \cdots & h_{m-3} & h_{m-2} & h_{m-1} \\ h_1 & h_2 & \iddots & \iddots & h_{m-1} & h_m \\ h_2 & h_3 & \iddots & \iddots & \iddots & h_{m+1} \\ \vdots & \vdots & \iddots & \iddots & \iddots & \vdots \\ h_{m-2} & h_{m-1} & \iddots & \iddots & h_{2m-4} & h_{2m-3} \\ h_{m-1} & h_m & \cdots & h_{2m-4} & h_{2m-3} & h_{2m-2} \end{bmatrix}.$$

A Hankel matrix **H** is entirely determined by its last column and first row, and thus depends on having $2m-1$ parameters, i.e., $h=(h_0, h_1,\ldots, h_{m-2}, h_{m-1}, h_m,\ldots, h_{2m-3}, h_{2m-2})$. The entries of **H** are constant down the diagonals parallel to the main anti-diagonal. Let $C=hB$ be the product of h and B, where $B=(b_0,b_1,\ldots,b_{m-1})$. The coordinate c_i of C is given by

$$c_i=\sum_{j=0}^{m-1} h_{j+i}b_j \tag{7}$$

Definition 3. Let i,j,m be these positive integers with $0 \le i,j \le m-1$, one can define the following function $\sigma(i,j)$:

$$\sigma(i,j)=\begin{cases} <\dfrac{j-i}{2}> & for \quad i+j=even \\ <-\dfrac{j+i+1}{2}> & for \quad i+j=odd \end{cases}$$

where $<q>$ denotes q mod m.

Let i denote a fixed integer in the complete set $\{1,2,\ldots,m-1\}$, one verifies that the map $k=\sigma(i,j)$. For instance, Table 1 presents the relationship between j and k with $k=\sigma(i,j)$, where $m=7$, $i=2$, and $0 \le j \le 6$. Therefore, by substituting $j=\sigma(i,j)$ into Equation (6), the product C can be denoted as

$$C=\sum_{i=0}^{m-1}\left(\sum_{j=0}^{m-1} h_{\sigma(i,j)+i}b_{\sigma(i,j)}\right)\gamma^i \tag{8}$$

j	0	1	2	3	4	5	6
k=σ(i,j)	6	5	0	4	1	3	2

Table 1. The relationship between j and k for i=2

Example 2: Let $A=(a_0, a_1, a_2, a_3, a_4)$ and $h=(h_0, h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8)$ represent two vectors and **H** represents the Hankel matrix defined by h; and let $C=(c_0, c_1, c_2, c_3, c_4)$ be the product of hA. By applying Equation (8), the product can be derived as follows.

$$\begin{array}{cccccc} & 1 & x & x^2 & x^3 & x^4 \\ & a_0h_0 & a_4h_5 & a_4h_6 & a_3h_6 & a_3h_7 \\ & a_4h_4 & a_0h_1 & a_3h_5 & a_4h_7 & a_2h_6 \\ & a_1h_1 & a_3h_4 & a_0h_2 & a_2h_5 & a_4h_8 \\ & a_3h_3 & a_1h_2 & a_2h_4 & a_0h_3 & a_1h_5 \\ + & a_2h_2 & a_2h_3 & a_1h_3 & a_1h_4 & a_0h_4 \\ \hline & c_0 & c_1 & c_2 & c_3 & c_4 \end{array}$$

Figure 1 depicts the bit-parallel systolic Hankel multiplier given by Example 1. The multiplier comprises 25 cells, including 14 U-cells and 11 V-cells. Every $U_{i,j}$ cell (Figure 2(a)) contains one 2-input AND gate, one 2-input XOR gate and three 1-bit latches to realize $c_i=c_i+a_{\sigma(i,j)}h_{\sigma(i,j)+i}$. Each $V_{i,j}$ cell (Figure 2(b)) is formed by one 2-input AND gate, one 2-input XOR gate and four 1-bit latches to realize the operation of $c_i=c_i+a_{\sigma(i,j)}h_{\sigma(i,j)+i}$, or $c_i=c_i+a_{\sigma(i,j)}h_{\sigma(i,j)+i+m}$. As stated in the above cell operations, the latency needs m clock cycles, and the computation time per cell is needed by one 2-input AND gate and one 2-input XOR gate.

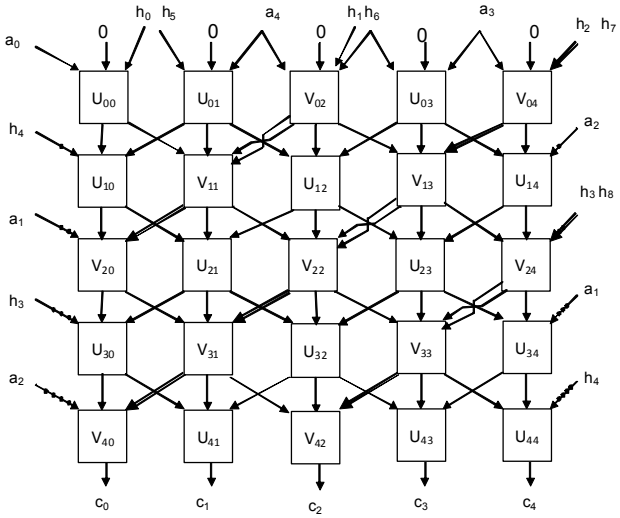


Fig. 1. The bit-parallel systolic Hankel multiplier [10]

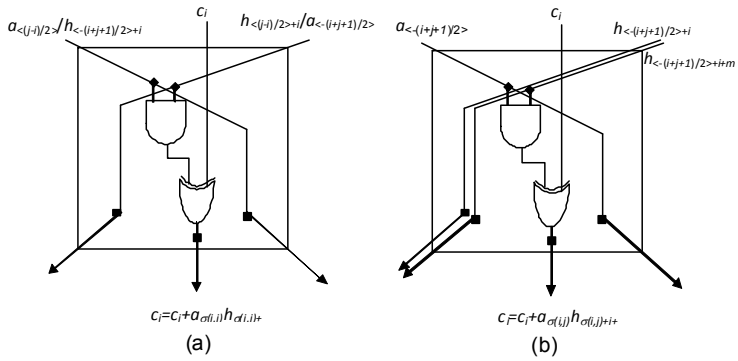


Fig. 2. (a) The detailed circuit of U-cell; (b) The detailed circuit of V-cell

3. Proposed scalable and systolic GNB multiplier architectures

Let $A = \sum_{i=0}^{p-1} a_{F(i)}\gamma^i$ and $B = \sum_{i=0}^{p-1} b_{F(i)}\gamma^i$ with $a_{F(0)}=b_{F(0)}=0$ and $a_{F(i)}$, $b_{F(i)}\in$GF(2) for $1\le i \le p-1$ denote two type-t GNB elements in GF(2^m), where γ represents the root of x^p+1. Assume that the chosen digital size is a d-bit, and $n=\lceil p/d \rceil$, both elements A and B can also be expressed as follows.

$$A = \sum_{i=0}^{n-1} A_i\gamma^{di} \ , \ B = \sum_{i=0}^{n-1} B_i\gamma^{di}$$

where

$$A_i = \sum_{j=0}^{d-1} a_{F(di+j)}\gamma^j \ , B_i = \sum_{j=0}^{d-1} b_{F(di+j)}\gamma^j \ .$$

Based on the partial multiplication for determining AB_0, the partial product can be denoted by

$$AB_0= A_0B_0 + A_1B_0\gamma^d+\ldots+ A_{n-1}B_0\gamma^{d(n-1)} \quad (9)$$

Each term A_iB_0 of degree $2d-2$ is the core computation of Equation (9). In a general multiplication, let us define that AiB0 is formed by

$$A_iB_0=S_i+D_i\gamma^d, \text{ for } 0 \le i \le n-1. \quad (10)$$

where

$$S_i=s_{i,0}+s_{i,1}\gamma+\ldots+s_{i,d-1}\gamma^{d-1}$$
$$D_i=d_{i,0}+d_{i,1}\gamma+\ldots+d_{i,d-1}\gamma^{d-2}$$
$$s_{i,j}= \sum_{k=0}^{j} a_{F(id+k)}b_{F(j-k)} \text{, for } 0 \le j \le d-1$$
$$d_{i,j}= \sum_{k=j+1}^{d-1} a_{F(id+k)}b_{F(d+j-k)} \text{, for } 0 \le j \le d-2$$

Therefore, Equation (9) can be re-expressed as

$AB_0=(S_0+D_0\gamma^d)+(S_1+D_1\gamma^d)\gamma^d+\ldots+(S_{n-1}+D_{n-1}\gamma^d)\gamma^{d(n-1)}$

$$= S_0+(S_1+D_0)\,\gamma^d +...+ (S_{n-1}+D_{n-2})\,\gamma^{d(n-1)}+ D_{n-1}\gamma^{dn}$$
$$= C_0+C_1\gamma^d +...+ C_{n-1}\gamma^{d(n-1)} + C_n\gamma^{dn} \quad (11)$$

where
$C_0= S_0$,
$C_i = S_i+D_{i-1}$, for $1 \le i \le n-1$
$C_n= D_{n-1}$
$S_i=(s_{i,0}, s_{i,1} ,..., s_{i,d-1})$ in Equation (10) can be translated with the following matrix-vector

$$\begin{bmatrix} s_{i,0} \\ s_{i,1} \\ \vdots \\ s_{i,d-2} \\ s_{i,d-1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cdots & 0 & a_{F(id)} \\ 0 & 0 & \iddots & a_{F(id)} & a_{F(id+1)} \\ \vdots & \iddots & \iddots & \vdots & \vdots \\ 0 & a_{F(id)} & \cdots & a_{F(id+d-3)} & a_{F(id+d-2)} \\ a_{F(id)} & a_{F(id+1)} & \cdots & a_{F(id+d-2)} & a_{F(id+d-1)} \end{bmatrix} \begin{bmatrix} b_{F(d-1)} \\ b_{F(d-2)} \\ \vdots \\ b_{F(1)} \\ b_{F(0)} \end{bmatrix} \quad (12)$$

Similarly, $D_{i-1}=(d_{i-1,0}, d_{i-1,1} ,..., d_{i-1,d-2}, 0)$ can also be translated with the following matrix-vector

$$\begin{bmatrix} d_{i-1,0} \\ d_{i-1,1} \\ \vdots \\ d_{i-1,d-2} \\ 0 \end{bmatrix} = \begin{bmatrix} a_{F(d(i-1)+1)} & \cdots & a_{F(id-2)} & a_{F(id-1)} & 0 \\ a_{F(d(i-1)+2)} & \cdots & a_{F(id-1)} & 0 & 0 \\ \vdots & \iddots & \iddots & \iddots & \vdots \\ a_{F(id-1)} & \iddots & \iddots & \iddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 \end{bmatrix} \begin{bmatrix} b_{F(d-1)} \\ b_{F(d-2)} \\ \vdots \\ b_{F(1)} \\ b_{F(0)} \end{bmatrix} \quad (13)$$

According to Equations (12) and (13), $C_i=S_i+D_{i-1}$ in Equation (11) can obtain

$$\begin{bmatrix} c_{F(di)} \\ c_{F(di+1)} \\ \vdots \\ c_{F(d(i+1)-1)} \end{bmatrix} = \begin{bmatrix} a_{F(d(i-1)+1)} & a_{F(d(i-1)+2)} & \cdots & a_{F(id)} \\ a_{F(d(i-1)+2)} & a_{F(d(i-1)+3)} & \cdots & a_{F(id+1)} \\ \vdots & \vdots & \iddots & \vdots \\ a_{F(di)} & a_{F(di+1)} & \cdots & a_{F(id+d-1)} \end{bmatrix} \begin{bmatrix} b_{F(d-1)} \\ b_{F(d-2)} \\ \vdots \\ b_{F(0)} \end{bmatrix} \quad (14)$$
$$= \mathbf{H}_{n-i}B_0$$

From the above matrix-vector multiplication, the Hankel vector $h_{n-i}=(a_{F(d(i-1)+1)}, a_{F(d(i-1)+2)},\ldots, a_{F(id)}, a_{F(id+1)},\ldots, a_{F(d(i+1)-1)})$ is defined by the $d\times d$ Hankel matrix H_{n-i}. Hence, the computation AB_0 using the Hankel matrix-vector representation can be computed as follows:

$$AB_0=h_nB_0+ h_{n-1}B_0\gamma^d+\ldots+h_0B_0\gamma^{dn} \quad (15)$$

Therefore, AB_0 can be dismembered into $(n+1)$ Hankel multiplications, and can be performed by the following algorithm:

Algorithm 3: $PM(A,B_0)$

Input: $A=\sum_{i=0}^{p-1} a_{F(i)}\gamma^j$ and $B_0 = \sum_{i=0}^{d-1} b_{F(i)}\gamma^i$

Output: $C = \sum_{i=0}^{p-1} c_{F(i)}\gamma^i = AB_0 \bmod (\gamma^p+1)$

1. convert the Hankel vector $h_{n-i}=(a_{F(d(i-1)+1)}, a_{F(d(i-1)+2)},\ldots, a_{F(d(i+1)-1)})$, for $0\le i \le n$, from A.
2. $C=(c_{F(0)}, c_{F(1)},\ldots, c_{F(p-1)})=(0,\ldots,0)$
3. for i=0 to n {
4. $X_i=h_{n-i}B_0$
5. }

6. $C= X \bmod (\gamma^p+1)$
7. return $C=(c_{F(0)}, c_{F(1)},\ldots, c_{F(p-1)})$

Algorithm 3 for determining AB_0 includes two core operations, namely the Hankel multiplication and the reduction polynomial γ^p+1, as illustrated in Figure 3. The proposed partial multiplier architecture in Figure 3 can be calculated using the following procedure.

Step 1: From Equation (14), we can see that Hankel matrix H_{n-i} is defined by all coefficients in A. Here $A=(a_{F(0)}, a_{F(1)},\ldots, a_{F(p-1)})$ is firstly converted to the Hankel vector $h_{n-i}=(a_{F(d(i-1)+1)}, a_{F(d(i-1)+2)},\ldots, a_{F(d(i+1)-1)})$ and its result is stored in the register H_{n-i}.

Step 2: Applying the bit-parallel systolic Hankel multiplier as shown in Figure1, Figure 3 shows AB_0 computation. Each Hankel multiplications in Step 4 of Algorithm 3, the result of AB_0 is stored in the register X.

Step 3: After (n+1) Hankel multiplications, the result needs to perform the reduction polynomial γ^p+1.

Generally, the computation of AB_i for $0 \le i \le n-1$ can be obtained by the following formula:

$$AB_i=h_nB_i+ h_{n-1}B_i\gamma^d+\ldots+ h_0B_i\gamma^{dn} \quad (16)$$

The above equation indicates that each AB_i computation can be dismembered into (n+1) Hankel multiplications. As mentioned above, two GNB multipliers are described in the following subsections.

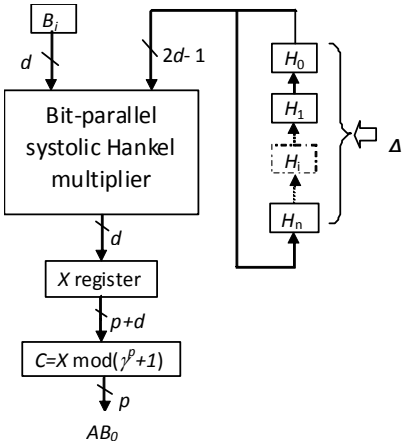


Fig. 3. The proposed scalable and systolic architecture for computing AB0

3.1 LSD-first scalable systolic multiplier

The product $C= AB \bmod (\gamma^p+1)$ using LSD-first multiplication algorithm can be represented as

$$\begin{aligned} C=AB_0 \bmod (\gamma^p+1) +AB_1\gamma^d \bmod (\gamma^p+1)+\ldots+ AB_{n-1}\gamma^{d(n-1)} \bmod (\gamma^p+1) \\ = A^{(0)}B_0 +A^{(d)}B_1+\ldots+ A^{(d(n-1))}B_{n-1} \end{aligned} \quad (17)$$

where $A^{(id)} = A\gamma^{id} \bmod(\gamma^p+1) = A^{(d(i-1))}\gamma^d \bmod(\gamma^p+1) = \sum_{j=0}^{p-1} a_{F(j-id)}\gamma^j$.

Applying Equations (16) and (17), the proposed LSD-first scalable systolic GNB multiplication is addressed as follows:

Algorithm 4. (LSDGNB scalable multiplication)
Input: $A=(a_0,a_1,\cdots,a_{m-1})$ and $B=(b_0,b_1,\cdots,b_{m-1})$ are two normal basis elements in GF(2^m)
Output: $C=(c_0,c_1,\cdots,c_{m-1})=AB$
1. Initial step:
1.1 $A=(a_{F(0)},a_{F(1)},\cdots,a_{F(p-1)})\leftarrow(a_0,a_1,\cdots,a_{m-1})$
1.2 $B=(b_{F(0)},b_{F(1)},\cdots,b_{F(p-1)})\leftarrow(b_0,b_1,\cdots,b_{m-1})$
1.3 $B=\sum_{i=0}^{n-1}B_i\gamma^{di}$, where $B_i=\sum_{j=0}^{d-1}b_{F(id+j)}\gamma^j$ and $n=\lceil p/d\rceil$
1.4 $C=0$
2. Multiplication step:
2.1 for $i=0$ to $n-1$ do
2.2 $C=C+PM(A,B_i)$ (where $PM(A,B_i)$ as referred to Algorithm 3)
2.3 $A=A\gamma^d \bmod (\gamma^p+1)$
2.4 endfor
3. Basis conversion step:
3.1 $C=(\underbrace{c_0,\cdots,c_0}_{t},\cdots,c_{m-1},\cdots,c_{m-1})\leftarrow(c_{F(0)},c_{F(1)},\cdots,c_{F(p-1)})$
4. Return $(c_0,c_1,\cdots,c_{m-1})$

The proposed LSDGNB scalable multiplication algorithm is split into n-loop partial multiplications. Figure 4 depicts the LSDGNB multiplier based on the proposed partial multiplier in Fig.3. Both NB elements A and B are initially transformed into the redundant basis given by Equation (4), and are stored in both registers A and B, respectively. In round 0 (see Figure 4), the systolic array in Figure 3 is adopted to compute $C=A^{(0)}B_0$, and the result is stored in register C. In round 1, the element A must be cyclically shifted to the right by d digits. The result produced in the systolic array is added to register C in the round 0. The first round, which estimates the latency, requires $d+n$ clock cycles. Each subsequent round computation requires a latency of $n+1$ clock cycles. Finally, the entire multiplication requires a latency of $d+n(n+1)$ clock cycles. The critical propagation delay of every cell is the total delay of one 2-input AND gate, one 2-input XOR gate and one 1-bit latch.

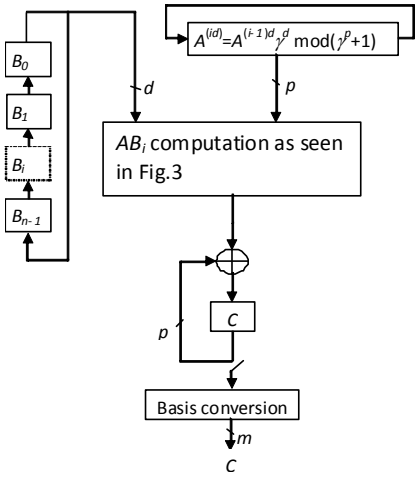


Fig. 4. The proposed LSD-first scalable systolic GNB multiplier over GF(2^m)

3.2 MSD-first scalable multiplier

From Equation (17), the product C can also be re-written by

$$C=(\ldots(AB_{n-1} \bmod (\gamma^p+1))\ \gamma^d+AB_{n-2} \bmod (\gamma^p+1))\ \gamma^d+\ldots)\ \gamma^d+AB_0 \bmod (\gamma^p+1) \quad (18)$$

Therefore, the MSD-first scalable systolic multiplication is addressed using the following algorithm.

Algorithm 5. (MSDGNB scalable multiplication)

Input: A and B are two normal basis elements in GF(2^m)

Output: $C=AB$

1. initial step:

1.1 $A=(a_{F(0)},a_{F(1)},\cdots,a_{F(p-1)}) \leftarrow (a_0,a_1,\cdots,a_{m-1})$

1.2 $B=(b_{F(0)},b_{F(1)},\cdots,b_{F(p-1)}) \leftarrow (b_0,b_1,\cdots,b_{m-1})$

1.3 $B=\sum_{i=0}^{n-1} B_i\gamma^{di}$, where $n=\lceil p/d \rceil$ and $B_i=\sum_{j=0}^{d-1} b_{F(id+j)}\gamma^j$

1.4 $C=0$

2. multiplication step:

2.1 for i=1 to n do

2.2 $C=C\gamma^d \bmod (\gamma^p+1)+PM(A,B_{n-i})$, where $PM(A,B_{n-i})$ as referred to Algorithm 3

2.3 endfor

3. basis conversion step:

3.1 $C=(\underbrace{c_0,\cdots,c_0}_{t},\cdots,c_{m-1},\cdots,c_{m-1}) \leftarrow C=(c_{F(0)},c_{F(1)},\cdots,c_{F(p-1)})$

4. return $(c_0,c_1,\cdots,c_{m-1})$

Algorithm 5 presents the MSD-first scalable multiplication, and Figure 5 presents the entire GNB multiplier architecture. As compared to both GNB multiplier architectures, the LSDGNB multiplier before each round computation, the element A must be performed by $A^{(id)}=A^{(i-1)d}\gamma^d \bmod(\gamma^p+1)$. The MSDGNB multiplier after each round computation, the result C

must be performed by $C^{(id)}=C^{(i-1)d}\gamma^d \bmod(\gamma^p+1)$. Notably, both operations $A^{(id)}$ and $C^{(id)}$ represent a right cyclic shift to id positions. Hence, the two proposed architectures have the same time and space complexity.

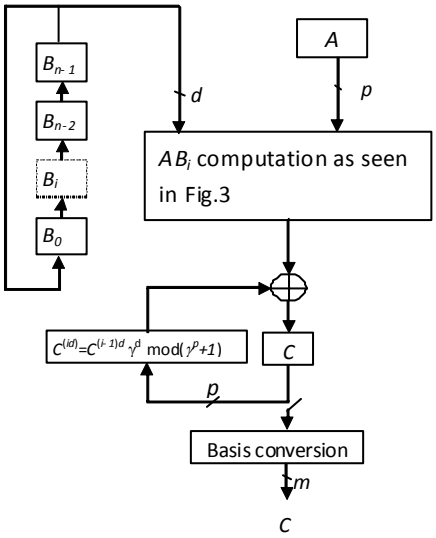


Fig. 5. The proposed MSD-first scalable systolic GNB multiplier over GF(2^m)

4. Modified Scalable and systolic GNB multiplier over GF(2^m)

In the previous section, two scalable GNB multipliers are including one $d\times d$ Hankel multiplier, four registers and one final reduction polynomial circuit. For the whole multiplication scheme, both circuits demand $n(n+1)$ Hankel multiplications. To reduce time- and space-complexity, this section will develop another version of the GNB scalable multiplication scheme.

Let A and B in GF(2^m) be represented by the redundant basis representation. If the selected digital size is d digits, then element B can be represented as $B=\sum_{i=0}^{n-1} B_i\gamma^{di}$, where $B_i=\sum_{j=0}^{d-1} b_{F(id+j)}\gamma^j$ and $n=\lceil (mt+1)/d \rceil$. From Equation (14), using LSD-first multiplication algorithm, the partial product can be modified by

$$\begin{aligned} C_0 &= AB_0 \bmod(\gamma^p+1) \\ &= Ab_{F(0)}+Ab_{F(1)}\gamma+\ldots+Ab_{F(d-1)}\gamma^{d-1} \bmod(\gamma^p+1) \\ &= A^{(0)}b_{F(0)}+A^{(1)}b_{F(1)}+\ldots+A^{(d-1)}b_{F(d-1)} \\ &= c_{0,F(0)}+c_{0,F(1)}\gamma+\ldots+c_{0,F(p-1)}\gamma^{p-1} \end{aligned} \tag{19}$$

where, $A^{(j)} = A\gamma^j \bmod(\gamma^p+1) = \sum_{i=0}^{p-1} a_{F(i-j)}\gamma^j$, for $0 \le j \le d-1$ and $c_{0,F(i)} = \sum_{j=0}^{d-1} b_{F(j)}a_{F(i-j)}$, for $0 \le i \le p-1$.

In (Wu, Hasan, Blake & Gao, 2002), it is shown that, from the GNB representation in (Reyhani-Masoleh & Hasan, 2005) to convert the normal basis, the minimum representation of A has a Hamming weight equal to or less than $mt/2$ if m is even, and $(mt-t+2)/2$ if m is odd. Assume that coordinate numbers of the partial product C_0 in Equation (19) are selected by $q=dk$ consecutive coordinates to satisfy the corresponding normal basis representation, where $q \ge mt/2$ if m is even, and $q \ge (mt-t+2)/2$ if m is odd. Then, the partial product AB_0 can be calculated by

$AB_0 = h_{k-1}B_0+h_{k-2}B_0\gamma^d +\ldots+h_0B_0\gamma^{d(k-1)}$.

Similarly,

$AB_1\gamma^d = h_kB_1+h_{k-1}B_1\gamma^d +\ldots+ h_1B_1\gamma^{d(k-1)}$

$AB_2\gamma 2^d = h_{k+1}B_2+h_kB_2\gamma^d +\ldots+ h_2B_2\gamma^{d(k-1)}$

$\vdots$

$AB_{n-1}\gamma^{(n-1)d} = h_{k+n-2}B_{n-1}+h_{k+n-3}B_{n-1}\gamma^d +\ldots+ h_{n-1}B_{n-1}\gamma^{d(k-1)}$

Thus, the modified multiplication requires only nk Hankel multiplication. As stated above, the modified LSD-first scalable multiplication is addressed as follows:

Algorithm 6. (modified LSDGNB scalable multiplication)

Input: $A=(a_0,a_1,\cdots,a_{m-1})$ and $B=(b_0,b_1,\cdots,b_{m-1})$ are two normal basis elements in GF(2^m)

Output: C= $(c_0,c_1,\cdots,c_{m-1})$ =AB

1. initial step:

1.1 $A=(a_{F(0)},a_{F(1)},\cdots,a_{F(p-1)}) \leftarrow (a_0,a_1,\cdots,a_{m-1})$

1.2 $B=(b_{F(0)},b_{F(1)},\cdots,b_{F(p-1)}) \leftarrow (b_0,b_1,\cdots,b_{m-1})$

1.3 $B=\sum_{i=0}^{n-1} B_i\gamma^{di}$, where $n=p/d$ and $B_i=\sum_{j=0}^{d-1} b_{F(id+j)}\gamma^j$

1.4 $C=\sum_{i=0}^{k-1} C_i\gamma^{di}=0$, where $k=q/d$ and $C_i=\sum_{j=0}^{d-1} c_{F(id+j)}\gamma^j$

1.5 All Hankel vector hi's for $0 \le i \le n+k-1$ are converted from the redundant basis representation of A.

2. multiplication step:

2.1 for i=0 to $k-1$ do

2.2 for j=0 to $n-1$ do

2.3 $C_{k-1-i}=C_{k-1-i}+H_{i+j}B_j$

2.4 endfor

2.5 endfor

3. basis conversion step:

3.1 $C=(c_0,\cdots,c_0,\cdots,c_{m-1},\cdots,c_{m-1}) \leftarrow (c_{F(0)},c_{F(1)},\cdots,c_{F(q-1)})$

4. return $(c_0,c_1,\cdots,c_{m-1})$

Applying Algorithm 6, Figure 6 shows a LSDGNB multiplier using the redundant representation. The circuit includes two registers, one $d \times d$ Hankel multiplier, and one

summation circuit. In the initial step, two registers H and B are converted by Steps 1.5 and 1.3, respectively. As for the register Hi represent $(2d–1)$-bit latches; and the register B_i is d-bit latches. The operation of a d × d Hankel matrix-vector multiplier has been described in the previous section. In Figure 6, the MUX is responsible for shifting register H. The SW is in charge of shifting the outcome of the GNB multiplication. As mentioned above, the total Hankel multiplications can be reduced from $n(n+1)$ to nk, where $k=\lceil mt/2d \rceil$ if m is even and $k=\lceil (mt-t+2)/2d \rceil$ if m is odd. By the configuration of Figure 6, the modified multiplier is without the final reduction polynomial circuit. It is revealed that the modified multiplier has lower time- and space-complexity as compared to Figures 4 and 5.

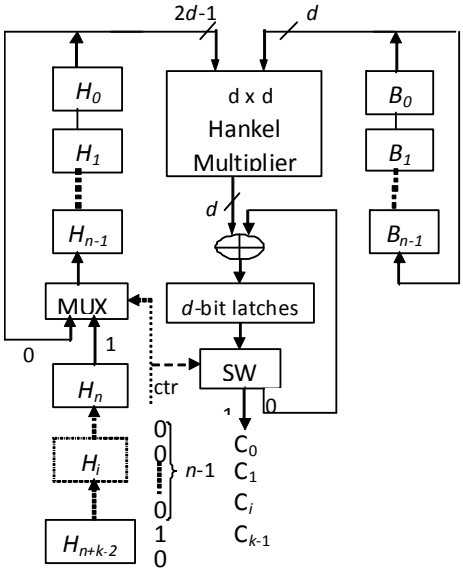


Fig. 6. The modified LSD-first scalable systolic GNB multiplier over GF(2^m)

5. Time and Area Complexity

Various bit-parallel systolic NB multipliers are only discussed on type-1 and 2 ONBs of GF(2^m), as in (Lee & Chiou, 2005 ; Kwon, 2005 ; Lee, Lu & Lee, 2001). As is well known, a type-1 ONB is built from an irreducible AOP, while a type-2 ONB can be constructed from a palindromic representation of polynomials of length $2m$. However, both ONB types exist about 24.5% for m< 1000, as depicted in IEEE Standard P1363 (2000). For the ECDSA (Elliptic Curve Digital Signature Algorithm) applications, NIST (2000) has recommended five binary fields. They are GF(2^{163}), GF(2^{233}), GF(2^{283}), GF(2^{409}) and GF(2^{571}). Their ONB multipliers are implemented with unscalable architectures, and the latency needs $m+1$ clock cycles. The space complexity of the previous architectures is proportional to m^2. As m becomes large, their hardware implements need very large area. Hence, the NIST architectures have limited very applications in cryptography. However, the proposed LSDGNB and MSDGNB multipliers do not have this problem.

Table 2 compares the circuits of the proposed scalable multipliers with those of the other un-scalable (bit-parallel) multipliers. Table 3 lists the proposed multipliers with the total latency. According to this table, the proposed multipliers for a type-2 GNB save about 40% latency as compared to Kwon's (2003) and Lee & Chiou's (2005) multipliers, and those for a type-1 GNB save about 60% latency as compared to Lee-Lu-Lee's multipliers (2001). Since the selected digital size d must minimize the total latency, the proposed multipliers then have low hardware complexity and low latency.

Multipliers	Kwon (2003)	Lee-Lu-Lee (2001)	Lee-Chiou (2005)	LSD-GNBM in Figure 4	MSD-GNBM in Figure 5
Basis	Type-II ONB	Type-I ONB	Type-II ONB	Gaussian NB	Gaussian NB
Architecture	bit-parallel	bit-parallel	bit-parallel	scalable	scalable
Total Complexity 2-input XOR 2-input AND 1-bit latch 1x2 SW	 $2m^2+m$ $2m^2+m$ $5m^2$ 0	 m^2+2m+1 m^2+2m+1 $3m^2+6m+1$ 0	 $2m^2+m$ $2m^2$ $7m^2$	 d^2+d+p d^2 $3.5d^2+5p+3nd$ p	 d^2+d+p d^2 $3.5d^2+5p+3nd$ p
Computation time per cell	T_A+2T_X	T_A+T_X	T_A+T_X	T_A+T_X	T_A+T_X
Latency	m+1	m+1	m+1	d+n(n+1)	d+n(n+1)

Note: the value d is the selected digital size; p=mt+1 is a prime number; $n=\lceil p/d \rceil$; T_X denotes 2-input XOR gate delay; T_A denotes 2-input AND gate delay

Table 2. Comparison of various systolic normal basis multipliers of GF(2^m)

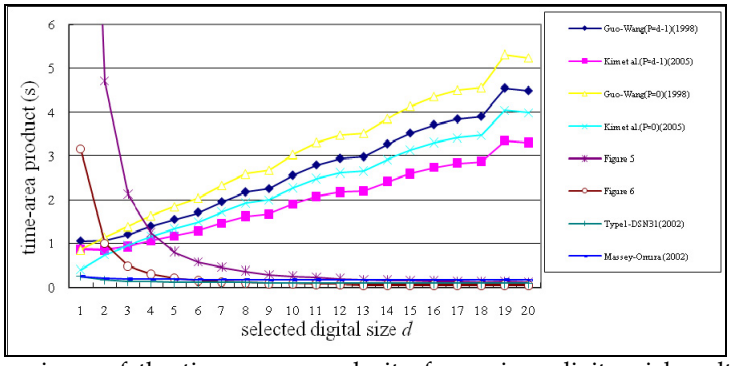


Fig. 7. Comparisons of the time-area complexity for various digit-serial multipliers over GF(2^{233})

Type-2 GNB					Type-1 GNB				
	the proposed architecture		Kwon (2003)	reduced latency		the proposed architecture		Lee-Lu-Lee (2001)	reduced latency
m	d	Minimum latency	latency	Compared to Kwon (2003)	m	d	Minimum latency	latency	Compared to Lee-Lu-Lee (2001)
146	46	88	147	40%	100	29	41	101	59.40%
155	57	87	156	44%	106	31	43	107	59.80%
158	58	88	159	45%	130	30	50	131	61.80%
173	64	94	174	46%	138	31	51	139	63.80%
174	64	94	175	46%	148	34	54	149	63.80%
179	66	96	180	46.60%	162	37	57	163	65%
183	67	97	184	47%	172	39	59	173	65.90%
186	68	98	187	47.60%	178	40	60	179	66.50%
189	69	99	190	47.90%	180	41	61	181	66.30%
191	59	101	192	47.40%	196	44	64	197	67.50%
194	60	102	195	47.70%	210	48	68	211	67.80%
209	65	107	210	49%	226	51	71	227	68.70%

Table 3. Lists the total latency for type-1 and type-2 GNB multipliers over GF(2^m)

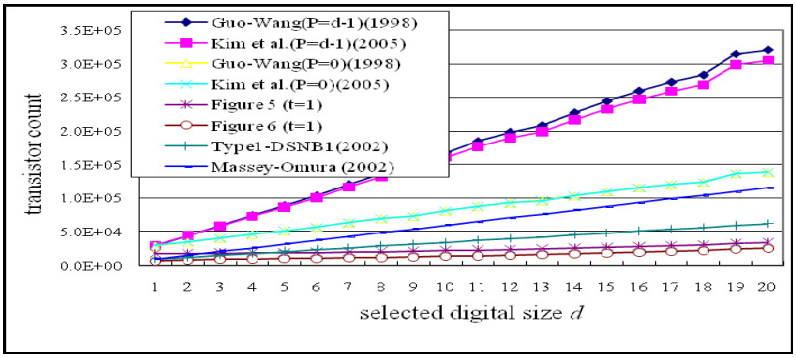


Fig. 8. Comparisons of transistor count for various digit-serial multipliers over GF(2^{233})

By applying the cut-set systolization techniques (Kung, 1988), various digit-serial systolic multipliers are recently reported in (Kim, Hong & Kwon, 2005; Guo & Wang, 1998), which are identical of n processing elements (PE) to enhance the trade-off between throughput performance and hardware complexity. Each PE requires a maximum propagation delay of $T_{max}=(d-1)(T_A+T_{iX}+T_M)+T_A+T_{iX}$, where T_A, T_{iX} and T_M denote the propagation delays through a 2-input AND gate, an i-input XOR gate and a 2-to-1 multiplexer, respectively. The maximum propagation delay in each PE is large if the selected digital size d is large. However, the proposed scalable systolic architectures do not have such problems, since the propagation delay of each PE is independent of the selected digital size d. Applying Horner's rule, Song and Parhi (1998) suggested MSD-first and LSD-first digit-serial multipliers. Various digit-serial multipliers use only one of the input signals A and B to separate $n=\lceil m/d \rceil$ sub-word data. Our

proposed architectures separate both input signals into n sub-word data, in which one of the input element is translated into Hankel vector representation. The proposed LSD-first and MSD-first scalable multiplication algorithms require $n(n+1)$ Hankel multiplications, and the modified multiplication algorithm only demands nk Hankel multiplications, where $k=\lceil mt/2d \rceil$ if m is even and $k=\lceil (mt-t+2)/2d \rceil$ if m is odd. Using a single Hankel multiplier to implement our proposed scalable multipliers, we have $O(d^2)$ space complexity, while other digit-serial multipliers require O(md) space complexity, as seen in Tables 4 and 5.

For comparing the time-area complexity, the transistor count based on the standard CMOS VLSI realization is employed for comparison. Therefore, some basic logic gates: 2-input XOR, 2-input AND, 1×2 SW, MUX and 1-bit latch are assumed to be composed of 6, 6, 6, 6 and 8 transistors, respectively (Kang & Leblebici, 1999). Some real circuits (STMicroelectronics, http://www.st.com) such as M74HC86 (STMicroelectronics, XOR gate, $T_X=12ns$ (TYP.)), M74HC08 (STMicroelectronics, AND gate, $T_A=7ns$ (TYP.)), M74HC279 (STMicroelectronics, SR Latch, $T_L=13ns$ (TYP.)), M74H257 (STMicroelectronics, Mux, $T_M=11ns$ (TYP.)) are employed for comparing time complexity in this paper. In the finite field GF(2^{233}), Figures 7 and 8 show that our proposed scalable multipliers compare to the corresponding digit-serial multipliers (Kim, Hong & Kwon, 2005; Guo & Wang, 1998; Reyhani-Masoleh & Hasan, 2002). As the selected digital size $d \geq 4$, the proposed scalable multipliers have lower time-area complexity than two reported digit-serial PB multipliers (Kim, Hong & Kwon, 2005; Guo& Wang, 1998) (as shown in Figure 7). When the selected digital size $d \geq 8$, the modified scalable multiplier has lower time-area complexity than the corresponding digit-serial NB multiplier (Reyhani-Masoleh & Hasan, 2002). For comparing a transistor count, Figure 8 reveals that our scalable multipliers have low space complexity as compared to the reported digit-serial multipliers (Kim, Hong & Kwon, 2005; Guo & Wang, 1998; Reyhani-Masoleh & Hasan, 2002).

Multipliers	Guo & Wang (1998)	Kim, Hong & Kwon (2005)	Figure 5	Figure 6
Basis	polynomial	polynomial	Gaussian normal	Gaussian normal
Architecture	digit-serial	digit-serial	scalable	scalable
Total Complexity				
2-input XOR	$2ed^2$	$2ed^2$	d^2+d+p	d^2+d
2-input AND	$e(2d^2+d)$	$e(2d^2+d)$	d^2	d^2
1-bit latch	$\lceil 10d+5Pd \rceil e$	$\lceil 10d+1+4.5Pd+P \rceil e$	$3.5d^2+5p+3nd$	Q
1x2 SW	0	0	p	d
MUX	2ed	2ed	0	1
Critical Path	Pipelined: $d(T_A+2T_X+2T_M)/(P+1)$ Non-Pipelined: $T_A+3T_X+(d-1)(T_A+2T_X+2T_M)$	Pipelined: $d(T_A+T_X+2T_M)/(P+1)$ Non-Pipelined: $T_A+T_X+(d-1)(T_A+T_X+2T_M)$	T_A+T_X	T_A+T_X
Latency	Pipelined: 3+P)e Non-Pipelined: 3e	Pipelined: (3+P)e Non-Pipelined: 3e	d+n(n+1)	d+nk

Note: $k=\lceil h/d \rceil$, $e=\lceil m/d \rceil$; $Q=3.5d^2+nd+(2d-1)(n+k-1)$; TM: denotes 2-by-1 MUX gate delay; P+1 number of pipelining stages inside each basic cell (Kim, Hong & Kwon, 2005; Guo & Wang, 1998)

Table 4. Comparison of various digit-serial systolic multipliers over GF(2m)

Multipliers	Type1-DSNB1 (Reyhani-Masoleh & Hasan , 2002)	Massey-Omura (Reyhani-Masoleh & Hasan , 2002)	Figure 5	Figure 6
Architecture	digit-serial nonsystolic	digit-serial nonsystolic	scalable systolic	scalable systolic
Total Complexity				
2-input XOR	(d+1)(m–1)	d(2m–2)	d^2+d+p	d^2+d
2-input AND	d(m–1)+m	d(2m–1)	d^2	d^2
1-bit latch	0	0	$3.5d^2+5p+3nd$	Q
1x2 SW	0	0	p	d
MUX	0	0	0	1
Critical Path	TA+(1+$\lceil \log_2 m \rceil$)TX	TA+(1+$\lceil \log_2 m \rceil$)TX	TA+TX	TA+TX
Latency	1	1	d+n(n+1)	d+nk

Table 5. Comparison of various digit-serial multipliers for optimal normal basis of GF(2^m)

6. Conclusions

This work presents new multiplication algorithms for the GNB of GF(2^m) to realize LSD-first and MSD-first scalable multipliers. The fundamental difference of our designs from other digit-serial multipliers described in the literature is based on a Hankel matrix-vector representation to achieve scalable multiplication architectures. In the generic field, the GNB multiplication can be decomposed into $n(n+1)$ Hankel multiplications. To use the relationship from the GNB to NB, we can modify the LSD-first scalable multiplication algorithm to decrease the number of Hankel multiplication from $n(n+1)$ into nk, where $k=mt/2d$ if m is even and $k=(mt-t-2)/2d$ if m is odd. Our analysis shows that, in finite field GF(2^{233}), if the selected digital size $d \geq 8$, the proposed scalable multipliers then have lower time-area complexity as compared to existing digit-serial multipliers for polynomial basis and normal basis of GF(2^m). Since our proposed scalable multiplication algorithms have highly flexible and are suitable for implementing all-type GNB multiplications. Finally, the proposed architectures have good trade-offs between area and speed for implementing cryptographic schemes in embedded systems.

7. References

Denning, D.E.R.(1983). *Cryptography and Data Security*, Reading, MA: Addison-Wesley.

Rhee, M.Y. (1994). *Cryptography and Secure Communications*, McGraw-Hill, Singapore.

Menezes, A. Oorschot, P. V. & Vanstone, S. (1997). *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL.

Massey, J.L. & Omura, J.K. (1986). Computational method and apparatus for finite field arithmetic," U.S. Patent Number 4,587,627.

Reyhani-Masoleh, A. & Hasan, M.A. (2005). Low complexity word-level sequential normal basis multipliers. *IEEE Transactions on Computers*, Vol. 54, No.2.

Lee, C.Y. & Chang, C.J. (2004). Low-complexity linear array multiplier for normal basis of type-II, *IEEE International Conference on Multimedia and Expo*, Vol. 3, pp. 1515-1518.

Lee, C.Y., Lu, E.H., & Lee, J.Y. (2001). Bit-Parallel Systolic Multipliers for GF(2^m) Fields Defined by All-One and Equally-Spaced Polynomials," *IEEE Transactions on Computers*, Vol. 50, No. 5, pp. 385-393.

Hasan, M.A., Wang, M.Z. & Bhargava, V.K. (1993). A modified Massey-Omura parallel multiplier for a class of finite fields," *IEEE Transactions on Computers*, Vol. 42, No.10, pp. 1278-1280.

Kwon, S. (2003). A low complexity and a low latency bit parallel systolic multiplier over GF(2^m) using an optimal normal basis of type II. *Proceedings of 16th IEEE Symp. Computer Arithmetic*, pp. 196-202.

Lee, C.Y. & Chiou, C.W. (2005). Design of low-complexity bit-parallel systolic Hankel multipliers to implement multiplication in normal and dual bases of GF(2^m). *IEICE Transactions on Fundamentals*, vol. E88-A, no.11, pp. 3169-3179.

IEEE Standard P1363 (2000). *IEEE Standard Specifications for Public-Key Cryptography*.

National Inst. of Standards and Technology,(2000). Digital Signature Standard, FIPS Publication 186-2.

Reyhani-Masoleh, A. (2006). Efficient algorithms and architectures for field multiplication using Gaussian normal bases. *IEEE Transactions on Computers*, Vol. 55, No. 1, pp.34-47.

Lee[1], C.Y. (2003). Low-Latency Bit-Parallel Systolic Multiplier for Irreducible x^m+x^n+1 with gcd(m,n)=1. *IEICE Transactions on Fundamentals*, Vol. E86-A, No.11, pp. 2844-2852.

Lee, C.Y., Horng, J.S. & Jou, I.C. (2005). Low-complexity bit-parallel systolic Montgomery multipliers for special classes of GF(2^m). *IEEE Transactions on Computers*, vol. 54, no.9, pp. 1061-1070.

Lee, C.Y. (2005). Systolic architectures for computing exponentiation and multiplication over GF(2^m) using polynomial ring basis. *Journal of LungHwa University*, vol. 19, pp.87-98.

Lee[2], C.Y. (2003). Low complexity bit-parallel systolic multiplier over GF(2^m) using irreducible trinomials. *IEE Proceeding Computer, and Digital Technical*, Vol. 150, pp. 39-42.

Paar, C., Fleischmann, P. & Soria-Rodriguez, P. (1999). Fast arithmetic for public-key algorithms in Galois fields with composite exponents. *IEEE Transactions on Computers*, vol. 48, no.10, pp. 1025-1034.

Kim, N.Y. & Yoo, K.Y. (2005). Digit-serial AB2 systolic architecture in GF(2^m). *IEE Proceeding Circuits Devices Systems*, Vol. 152, No. 6, pp. 608-614.

Kang, S.M. & Leblebici, Y. (1999). *CMOS Digital Integrated Circuits Analysis and Design*, McGrawHill.

Logic selection guide: *STMicroelectronics*. <http://www.st.com > .

Kim, C.H., Hong, C.P. & Kwon, S. (2005). A digit-serial multiplier for finite field GF(2^m). *IEEE Transactions on VLSI*, Vol. 13, No. 4, pp. 476-483.

Guo, J.H. & Wang, C.L. (1998). Digit-serial systolic multiplier for finite fields GF(2^m). IEE Proc.-Comput. Digit. Tech., Vol. 145, No. 2, pp. 143-148, March.

Kung, S.Y. (1988). *VLSI array processors*, Englewood Cliffs, NJ: Prentice-Hall.

H. Wu, M.A. Hasan, I.F. Blake and S. Gao, "Finite field multiplier using redundant representation. *IEEE Transactions on Computers*, Vol. 51, No. 11, pp.1306-1316, Nov. 2002.

Mullin, R.C., Onyszchuk, I.M., Vanstone, S.A. & Wilson, R. M. (1988/1989). Optimal Normal Bases in GF(p^n). *Discrete Applied Math.*, vol. 22, pp.149-161.

Reyhani-Masoleh, A. & Hasan, M.A. (2003). Fast normal basis multiplication using general purpose processors. *IEEE Transactions on Computers*, Vol. 52, No. 11, pp. 1379-1390.

Song, L. & Parhi, K.K. (1998). Low-energy digit-serial/parallel finite field multipliers. *Journal of VLSI Signal Processing* , Vol.19, pp.149-166.

Tenca, A.F. & Koc, C.K. (1999). A scalable architecture for Montgomery multiplication. *Proceedings of Cryptographic Hardware and Embedded System* (CHES 1999), No. 1717 in Lecture Notes in Computer Science, pp. 94-108.

Reyhani-Masoleh, A. & Hasan, M.A. (2002). Efficient digit-serial normal basis multipliers over GF(2^M). *IEEE International Conference on Circuits and Systems*.

8

High-Speed VLSI Architectures for Turbo Decoders

Zhongfeng Wang[1] and Xinming Huang[2]
[1]*Broadcom Corporation, 5300 California Avenue, Irvine, CA 92617, USA,*
[2]*Department of ECE, Worcester Polytechnic Institute, Worcester, MA 01609, USA.*

Turbo code, being one of the most attractive near-Shannon limit error correction codes, has attracted tremendous attention in both academia and industry since its invention in early 1990's. In this chapter, we will discuss high-speed VLSI architectures for Turbo decoders. First of all, we will explore joint algorithmic and architectural level optimization techniques to break the high speed bottleneck in recursive computation of state metrics for soft-input soft-output decoders. Then we will present area-efficient parallel decoding schemes and associated architectures that aim to linearly increase the overall decoding throughput with sub-linearly increased hardware overhead.

Keywords: Turbo code, MAP algorithm, parallel decoding, high speed, VLSI.

1. Introduction

Error correction codes are an essential component in digital communication and data storage systems to ensure robust operation of digital applications, wherein, Turbo code, invented by Berrou (1993), is among the two most attractive near-optimal (i.e., near-Shannon limit) error correction codes. As a matter of fact, Turbo codes have been considered in several new industrial standards, such as 3rd and post-3rd generation cellular wireless systems (3GPP, 3GPP2, and 3GPP LTE), Wireless LAN (802.11a), WiMAX (broadband wireless, IEEE 802.16e) and European DAB and DVB (digital audio broadcasting and digital video broadcasting) systems.

One key feature associated with Turbo code is the iterative decoding process, which enables Turbo code to achieve outstanding performance with moderate complexity. However, the iterative process directly leads to low throughput and long decoding latency. To obtain a high decoding throughput, a large amount of computation units have to be instantiated for each decoder, and this results in a large chip area and high power consumption. In contrast, the growing market of wireless and portable computing devices as well as the increasing desire to reduce packaging costs have directed industry to focus on compact low-power circuit implementations. This tug-of-war highlights the challenge and calls for innovations on Very Large Scale Integration (VLSI) design of high-data rate Turbo decoders that are both area and power efficient.

For general ASIC design, there are two typical ways to increase the system throughput: 1) raise the clock speed, and 2) increase the parallelism. In this chapter, we will tackle the high speed Turbo decoder design in these two aspects. Turbo code decoders can be based on either *maximum-a-posterior* probability (MAP) algorithm proposed in Bahl (1974) (or any variants of approximation) or soft-output Viterbi algorithm (SOVA) proposed in Hagenauer (1989) (or any modified version). However, either algorithm involves recursive computation of state metrics, which forms the bottleneck in high speed integrated circuit design since conventional pipelining techniques cannot be simply applied for raising the effective clock speed. Look-ahead pipelining in Parhi (1999) may be applicable. But the introduced hardware overhead can be intolerable. On the other hand, parallel processing can be effective in increasing the system throughput. Unfortunately, direct application of this technique will cause hardware and power consumption to increase linearly, which is against the requirement of modern portable computing devices.

In this chapter, we will focus on MAP-based Turbo decoder design since MAP-based Turbo decoder significantly outperforms SOVA-based Turbo decoders in terms of Bit-Error-Rate (BER). In addition, MAP decoders are more challenging than SOVA decoders in high speed design (Wang 2007). Interested readers are referred to Yeo (2003) and Wang (2003c) for high data-rate SOVA or Turbo/SOVA decoder design. The rest of the chapter is organized as follows. In Section 2, we give background information about Turbo codes and discuss simple serial decoder structure. In Section 3, we will address high speed recursion architectures for MAP decoders. Both Radix-2 and Radix-4 recursion architectures are investigated. In Section 4, we present area-efficient parallel processing schemes and associated parallel decoding architectures. We conclude the chapter in Section 5.

2. Background of Turbo Codes

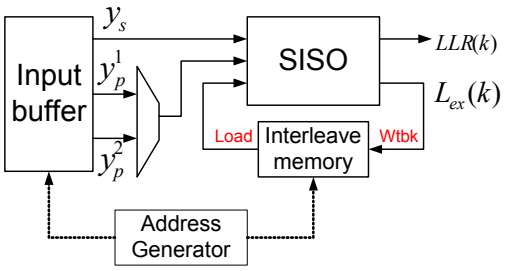


Fig. 1. A serial Turbo decoder architecture.

A typical Turbo encoder consists of two recursive systematic convolutional (RSC) encoders and an interleaver between them (Wang 1999). The source data are encoded by the first RSC encoder in sequential order while its interleaved sequence is encoded by the second RSC encoder. The original source bits and parity bits generated by two RSC encoders are sent out in a time-multiplexed way. Interested reader are referred to Berrou (1993) for details. Turbo code usually works with large block sizes for the reason that the larger the block size, the better the performance in general. In order to facilitate iterative decoding, the received data

of a whole decoding block have to be stored in a memory, whose size is proportional to the Turbo block size. Hence, Turbo decoders usually require large memory storage. Therefore serial decoding architectures are widely used in practice.

A typical serial Turbo decoder architecture is shown in Fig. 1. It has only one soft-input soft-output (SISO) decoder, which works in a time-multiplexed way as proposed in Suzuki (2000). Each iteration is decomposed into two decoding phases, i.e., *the sequential decoding phase*, in which the data are processed in sequential order, and *the interleaved decoding phase*, in which the source data are processed in an interleaved order. Both probability MAP algorithm proposed in Berrou (1993) and SOVA proposed in Hagenauer (1989) can be employed for the SISO decoding.

The serial Turbo decoder includes two memories: one is used for received soft symbols, called the input buffer or the receiver buffer, the other is used to store the extrinsic information, denoted as the interleaver memory. The extrinsic information is feedback as the *a priori* information for next decoding. The input buffer is normally indispensable. With regard to the interleaver memory, either two ping-pong buffers can be used to complete one Load and one Write operations required to process each information bit within one cycle or one single-port memory can be employed to fulfil the two required operations within two clock cycles. A memory-efficient architecture was presented by Wang (2003a) and Parhi, which can process both Read and Write operations at the same cycle using single-port memories with the aid of small buffers.

Turbo decoder works as follows. The SISO decoder takes soft inputs (including the received systematic bit y_s and the received parity bit y_p^1 or y_p^2) from the input buffer and the *a priori* information from the interleaver memory. It outputs the log likelihood ratio $LLR(k)$, and the extrinsic information, $L_{ex}(k)$, for the k-th information bit in the decoding sequence. The extrinsic information is sent back as the new *a priori* information for next decoding. The interleaving and de-interleaving processes are completed in an efficient way. Basically the data are loaded according to the current decoding sequence. For instance, the extrinsic information is loaded in sequential order at sequential decoding phase while being loaded in the interleaved order at the interleaved decoding phase. After processing, the new extrinsic information is written back to the original places. In this way, no de-interleave pattern is required for Turbo decoding.

3. High Speed Log-MAP Decoder Design

3.1 Maximum A Posterior (MAP) algorithm

As discussed in Section 2, practical Turbo decoders usually employ serial decoding architectures, such as Suzuki (2000), for area-efficiency. Thus, the throughput of a Turbo decoder is highly limited by the clock speed and the maximum number of iterations to be performed. To facilitate iterative decoding, Turbo decoders require soft-input soft-output decoding algorithms, among which the probability MAP algorithm proposed in Bahl (1974) is widely adopted for its excellent performance.

The MAP algorithm is commonly implemented in log domain, thus called Log-MAP (Wang 1999). The Log-MAP algorithm involves recursive computation of forward state metrics (simply called α metrics) and backward state metrics (simple called β metrics). The log-likelihood-ratio is computed based on the two types of state metrics and associated branch metrics (denoted as γ metrics). Due to different recursion directions in computing α and β metrics, a straightforward implementation of Log-MAP algorithm will not only consume large memory but also introduce large decoding latency. The sliding window approach was proposed by Viterbi (1998) to mitigate this issue. In this case, pre-backward recursion operations are introduced for the warm-up process of real backward recursion. For clarity, we denote the pre-backward recursion computing unit as $\beta 0$ unit and denote the real (effective or valid) backward recursion unit as $\beta 1$ unit. The details of Log-MAP algorithm will not be given in the chapter. Interested readers are referred to Wang (1999).

The timing diagram for typical sliding-window-based Log-MAP decoding is shown in Fig. 2, where SB1, SB2, etc, stand for consecutive sub-blocks (*i.e.*, sliding windows), the branch metrics computation was computed together with pre-backward recursion, though not shown in the figure for simplicity and clarity.

The structure of a typical serial Turbo decoder based on log-MAP algorithm is shown in Fig. 3, where the soft-output unit is used to compute LLR and extrinsic information (denoted as Lex), the interleaver memory is used to store the extrinsic information for next (phase of) decoding. It can be seen from the figure that both the branch metric unit (BMU) and soft-output unit (SOU) can be pipelined for high speed applications. However, due to recursive computation, three state metrics computation units form the high-speed bottleneck. The reason is that the conventional pipelining technique is not applicable for raising the effective processing speed unless one MAP decoder is used to process more than one Turbo code blocks or sub-blocks as discussed in Lee (2005). Among various high-speed recursion architectures in the literature such as Lee (2005), Urard (2004), Boutillon (2003), Miyouchi (2001) and Bickerstaff (2003), the designs presented in Urard (2004) and Bickerstaff (2003) are most attractive. In Urard (2004), an offset-add-compare-select (OACS) architecture is proposed to replace the traditional add-compare-select-offset (ACSO) architecture. In addition, the look-up table (LUT) is simplified with only 1-bit output, and the computation of absolute value is avoided through introduction of the reverse difference of two competing path (or state) metrics. An approximate 17% speedup over the traditional Radix-2 ASCO architecture was reported. With one-step look-ahead operation, a Radix-4 ACSO architecture can be derived. Practical Radix-4 architectures such as Miyouchi (2001) and Bickerstaff (2003) always involve approximations in order to achieve higher effective speedup. For instance, the following approximation is adopted in Bickerstaff (2003):

$$\max^*(\max^*(A, B), \max^*(C, D)) = \max^*(\max(A, B), \max(C, D)), \quad (1)$$

where

$$\max^*(A, B) = \max(A,B) + \log(1 + e^{-|A-B|}). \quad (2)$$

This Radix-4 architecture can generally improve the processing speed (equivalent to twice of its clock speed) by over 40% over the traditional Radix-2 architecture, and it has *de facto* the highest processing speed among all existing (MAP decoder) designs found in the literature.

However, the hardware will be nearly doubled compared to the traditional ACSO architecture presented in Urard (2004).

In this section, we will first present an advanced Radix-2 recursion architecture based on algorithmic transformation, approximation and architectural level optimization, which can achieve comparable processing speed as the state-of-the-art Radix-4 design while having significantly lower hardware complexity. Then we discuss an improved Radix-4 architecture that is 32% faster than the best existing approach.

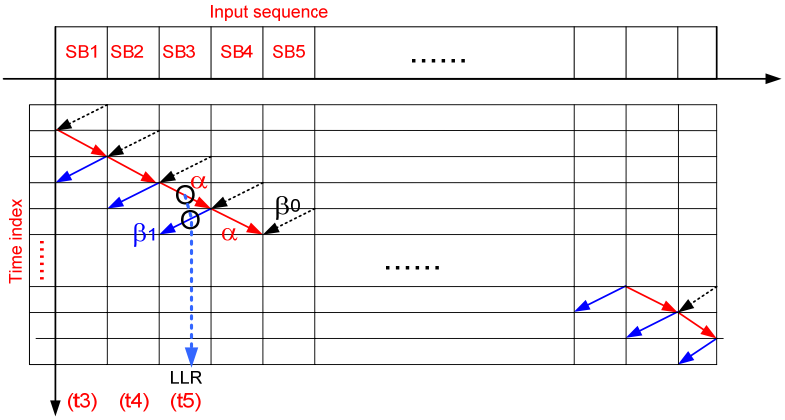


Fig. 2. The timing diagram for Log-MAP decoding.

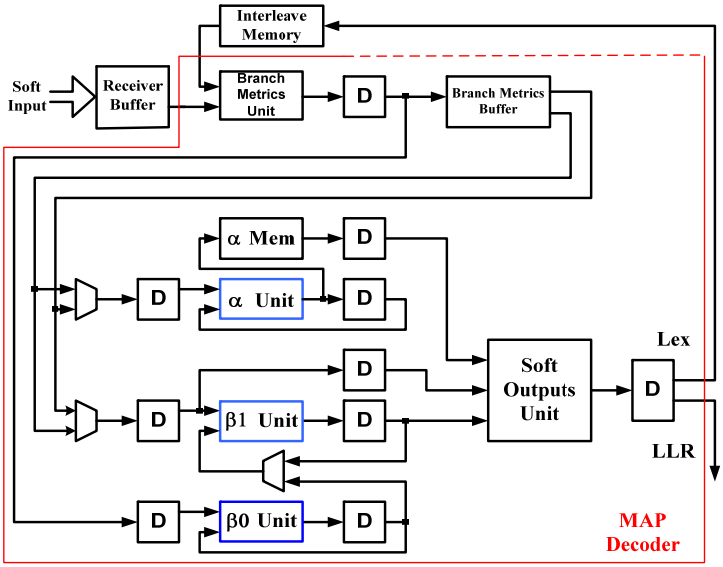


Fig. 3. A Log-MAP Turbo decoder structure.

3.2 An Advanced High-Speed Radix-2 Recursion Architecture for MAP Decoders

For convenience in later discussion, we first give a brief introduction to MAP-based Turbo decoder structure. As shown in Fig. 3, the branch metrics unit (**BMU**) takes inputs from the receiver buffer and the interleaver memory. The outputs of BMU are directly sent to the pre-backward recursion unit (*i.e.*, **β0** unit). The previously stored branch metrics for consecutive sliding windows are input to the forward recursion unit (*i.e.*, **α** unit) and the effective backward recursion unit (*i.e.*, **β1** unit), respectively. The soft output unit (**SOU**) that is used to compute the log likelihood ratio (**LLR**) and the extrinsic information (Lex) takes inputs from the previously stored **α** metrics, the currently computed **β** metrics and the previously stored branch metrics (**γ**). The **SOU** starts to generate soft outputs after the branch metrics have been computed for the first two sliding windows. It can be observed that the high-speed bottleneck of a Log-MAP decoder lies in the three recursive computation units since both BMU and SOU can be simply pipelined for high-speed applications. Thus, this section is dedicated to the design of high-speed recursive computation units, *i.e.*, **α**, **β0** and **β1** units shown in Fig. 3.

It is known from Log-MAP algorithm that all the three recursion units have similar architectures. So we will focus our discussion on design of **α** units. The traditional design for **α** computation is illustrated in Fig. 4, where the **ABS** block is used to compute the absolute value of the input and the **LUT** (*i.e.*, look-up table) block is used to implement a nonlinear function $\log(1+e^{-x})$, where x > 0. For simplicity, only one branch (*i.e.*, one state) is drawn. The overflow approach (Wu 2001) is assumed for normalization of state metrics as used in conventional Viterbi decoders.

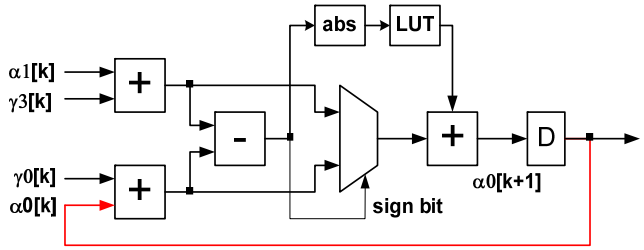


Fig. 4. The original recursion architecture: Arch-O.

It can be seen that the computation of the recursive loop consists of three multi-bit additions, the computation of absolute value and a random logic to implement the LUT. As there is only one delay element in each recursive loop, the traditional retiming technique in Denk (1998) can not be used to reduce the critical path.

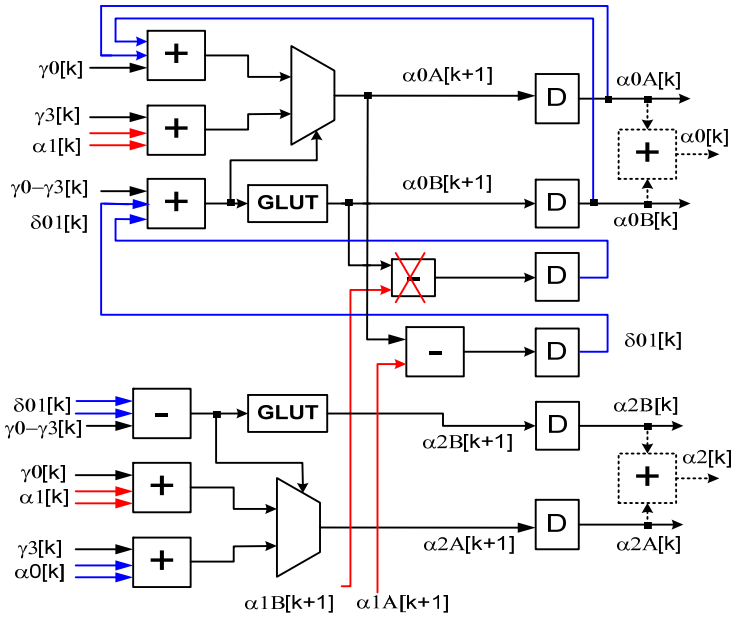


Fig. 5. An advanced Radix-2 fast recursion architecture.: Arch-A

In this work, we propose an advanced Radix-2 recursion architecture shown in Fig. 5. Here we first introduce a difference metric for each competing pair of states metrics (*e.g.*, **α0** and **α1** in Fig. 4) so that we can perform the front end addition and the subtraction operations simultaneously in order to reduce the computation delay of the loop. Secondly, we employ a generalized LUT (see GLUT in Fig. 5) that can efficiently avoid the computation of absolute value instead of introducing another subtraction operation as in Urard (2004). Thirdly, we move the final addition to the input side as with the OACS architecture in Boutillon (2003) and then utilize one stage carry-save structure to convert a 3-number addition to a 2-number addition. Finally, we make an intelligent approximation in order to further reduce the critical path.

The following equations are assumed for the considered recursive computation shown in Fig. 5:

$$\alpha 0[k+1] = \max^*(\alpha 0[k] + \gamma 0[k],\ \ \alpha 1[k] + \gamma 3[k]), \tag{3}$$

$$\alpha 2[k+1] = \max^*(\alpha 0k] + \gamma 3[k],\ \ \alpha 1[k] + \gamma 0[k]),$$

where max* function is defined in (2).

In addition, we split each state metrics into two terms as follows:

$$\begin{aligned} \alpha 0[k] &= \alpha 0A[k] + \alpha 0B[k], \\ \alpha 1[k] &= \alpha 1A[k] + \alpha 1B[k], \\ \alpha 2[k] &= \alpha 2A[k] + \alpha 2B[k]. \end{aligned} \tag{4}$$

Similarly, the corresponding difference metric is also split into two terms:

$$\delta_{01A}[k] = \alpha_{0A}[k] - \alpha_{1A}[k],$$
$$\delta_{01B}[k] = \alpha_{0B}[k] - \alpha_{1B}[k]. \tag{5}$$

In this way, the original add-and-compare operation is converted as an addition of three numbers, *i.e.*,

$$(\alpha_0 + \gamma_0) - (\alpha_1 + \gamma_3) = (\gamma_0 - \gamma_3) + \delta_{01A} + \delta_{01B} \tag{6}$$

where $\gamma 0 - \gamma 3$ is computed by BMU, the time index [k] is omitted for simplicity. In addition, the difference between the two outputs from two GLUTs, *i.e.*, δ_{01B}, can be neglected. From extensive simulations, we found that this small approximation doesn't cause any performance loss in Turbo decoding with either AWGN channels or Raleigh fading channels. This fact can be simply explained in the following. If one competing path metrics (*e.g.*, $p0 = \alpha 0 + \gamma 0$) is significantly larger than the other one (*e.g.*, $p1 = \alpha 1 + \gamma 3$), the GLUT output will not change the decision anyway due to their small magnitudes. On the other hand, if the two competing path metrics are so close that adding or removing a value from one GLUT may change the decision (*e.g.*, from *p0>p1* to *p1>p0*), picking any survivor (*p0 or p1*) should not make big difference.

At the input side, a small circuitry shown in Fig. 6 is employed to convert an addition of 3 numbers to an addition of 2 numbers, where FA and HA represents full-adder and half-adder respectively, XOR stands for exclusive OR gate, d0 and d1 correspond to the 2-bit output of GLUT. The state metrics and branch metrics are represented with 9 and 6 bits, respectively in this example. The sign extension is only applied to the branch metrics. It should be noted that an extra addition operation (see dashed adder boxes) is required to integrate each state metric before storing it into the α memory.

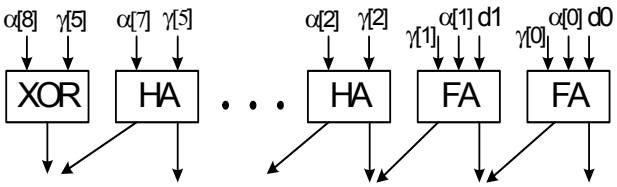


Fig. 6. A carry-save structure in the front end of Arch-A.

The generalized LUT (GLUT) structure is shown in Fig. 7, where the computation of absolute value is eliminated by including the sign bit into 2 logic blocks, *i.e.*, **Ls2** and **ELUT**, where the Ls2 function block is used to detect if the absolute value of the input is less than 2.0, and the ELUT block is a small LUT with 3-bit inputs and 2-bit outputs. It can be derived that $Z = \overline{S} b_7 \ldots + b_4 + \overline{b}_3 + S(b_7 \ldots b_4 b_3)$. It was reported in Gross (1998) that using two output values for the LUT only caused a performance loss of 0.03 dB from the floating point simulation for a 4-state Turbo code. The approximation is described as follows:

$$\text{If } |x| < 2,\ f(x) = 3/8;\ \text{else } f(x) = 0; \tag{7}$$

where x and f(x) stands for the input and the output of the LUT, respectively. In this approach, we only need to check if the absolute value of the input is less than 2 or not, which can be performed by the **Ls2** block in Fig. 7. A drawback of this method is that its performance would be significantly degraded if only two bits are kept for the fractional part of the state metrics, which is generally the case.

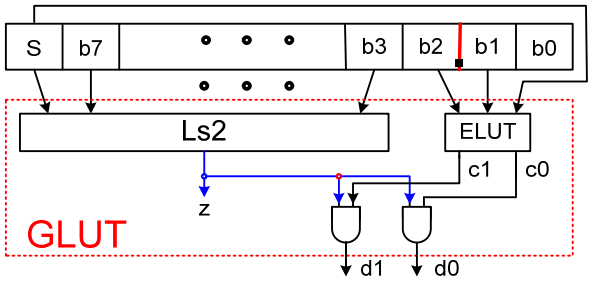


Fig. 7. The structure of GLUT used in Arch-A.

|x|	0.0	0.50	1.0	1.50
f(x)	¾	2/4	¼	¼

Table 1. The proposed LUT approximation

In our design, both the inputs and outputs of the LUT are quantized with 4 levels. The details are shown in Table 1. The inputs to ELUT are treated as a 3-bit signed binary number. The outputs of ELUT are *AND*ed with the output of **Ls2** block. This means, if the absolute value of the input is greater than 2.0, the output from the GLUT is set as 0. Otherwise the output from ELUT will be the final output.

The ELUT can be implemented with combinational logic for high speed applications. Its computation latency is smaller than the latency of **Ls2** block. Therefore, the overall latency of the GLUT is almost the same as the above-discussed simplified method, where the total delay consists of 1 MUX delay and the computation delay of **Ls2**.

After all the above optimization, the critical path of the recursive architecture is reduced to 2 multi-bit additions, one 2:1 MUX operation and 1-bit addition operation, which saves nearly 2 multi-bit adder delay compared to the traditional ACSO architecture. We will show detailed comparisons in subsection 3.4.

3.3 An Improved Radix-4 Architecture for MAP Decoders

In the following, we discuss an improved Radix-4 recursion architecture. The computation for $\alpha 0[k+2]$ is expressed as follows:

$$\begin{aligned}\alpha 0[k+2] &= \max^*(\alpha 0[k+1]+\gamma 0[k+1],\ \alpha 1[k+1]+\gamma 3[k+1]) \\ &= \max^*(\max^*(\alpha 0[k]+\gamma 0[k],\ \alpha 1[k]+\gamma 3[k])+\gamma 0[k+1], \\ &\quad \max^*(\alpha 2k]+\gamma 2[k],\ \alpha 3[k]+\gamma 1[k])+\gamma 3[k+1]),\end{aligned} \tag{8}$$

where

$$\alpha 1[k+1] = \max^*(\alpha 2[k]+\gamma 2[k],\ \alpha 3[k]+\gamma 1[k]). \tag{9}$$

In Bickerstaff (2003), Lucent Bell Labs proposed the following approximation:

$$\begin{aligned}\alpha 0[k+2] \approx \max^*(&\max(\alpha 0[k]+\gamma 0[k],\ \alpha 1[k]+\gamma 3[k])+\gamma 0[k+1], \\ &\max(\alpha 2k]+\gamma 2[k],\ \alpha 3[k]+\gamma 1[k])+\gamma 3[k+1]).\end{aligned} \tag{10}$$

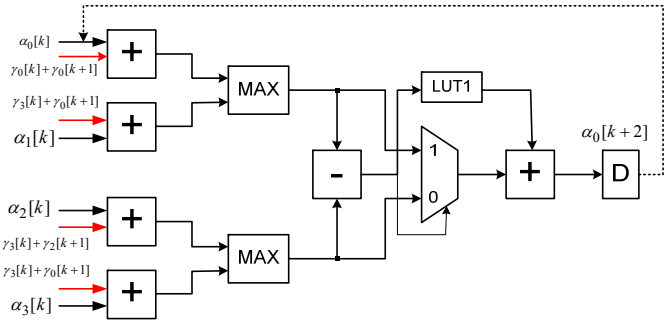


Fig. 8. The Radix-4 architecture proposed by Lucent Bell Labs: Arch-L.

This approximation is reported to have a 0.04 dB performance loss compared to the original Log-MAP algorithm. The architecture to implement the above computation is shown in Fig. 8 for convenience in later discussion. As it can be seen, the critical path consists of 4 multi-bit adder delay, one generalized LUT delay (Note: the LUT1 block includes absolute value computation and a normal LUT operation) and one 2:1 MUX delay.

Similarly, we can take an alternative approximation as follows:

$$\begin{aligned}\alpha 0[k+2] \approx \max(&\max^*(\alpha 0[k]+\gamma 0[k],\ \alpha 1[k]+\gamma 3[k])+\gamma 0[k+1], \\ &\max^*(\alpha 2k]+\gamma 2[k],\ \alpha 3[k]+\gamma 1[k])+\gamma 3[k+1]).\end{aligned} \tag{11}$$

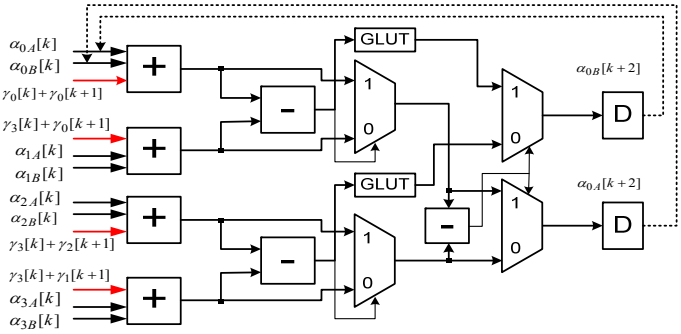


Fig. 9. The improved Radix-4 recursion arch.: Arch-B.

Intuitively, Turbo decoder employing this new approximation should have the same decoding performance as using equation (7). While directly implementing (11) does not bring any advantage to the critical path, we intend to take advantages of the techniques that we developed in subsection 3.2. The new architecture is shown in Fig. 9. Here we split each state metric into two terms and we adopt the same GLUT structure as we did before. In addition, a similar approximation is incorporated as with Arch-A. In this case, the outputs from GLUT are not involved in the final stage comparison operation. It can be observed that the critical path of the new architecture is close to 3 multi-bit adder delay. To compensate for all the approximation introduced, the extrinsic information generated by the MAP decoder based on this new Radix-4 architecture should be scaled by a factor around 0.75.

3.4 Performance Comparisons

For quantitative comparison in hardware complexity and processing speed, we used TSMC 0.18 um standard cells to synthesize one α unit for 5 different recursion architectures, *i.e.*, 1) the traditional ACSO architecture: Arch-O, 2) the reduced-precision Radix-2 recursion architecture presented in Urard (2004): Arch-U, 3) the Radix-4 architecture proposed by Lucent: Arch-L, 4) the advanced Radix-2 recursion architecture presented in this work: Arch-A, and 5) the improved Radix-4 recursion architecture: Arch-B. All the state metrics are quantized as 9 bits while the branch metrics are represented using 6 bits. The detailed synthesis results are listed in Table 2.

It can be observed that the proposed Radix-2 architecture has comparable processing speed as the Radix-4 architecture proposed by Bell Labs with significantly lower complexity, while the improved Radix-4 architecture is 32% faster with only 9% extra hardware. This amount of hardware overhead should be negligible compared to an entire Turbo decoder. It can also be seen that the new Radix-4 architecture achieves twice speedup over the traditional Radix-2 recursion architecture.

	Max clock Frequency (Mhz)	Relative Area	Relative Processing Speed
Arch-O	241	1.0	1.0
Arch-U	355	1.14	1.39
Arch-L	182	1.82	1.51
Arch-A	370	1.03	1.54
Arch-B	241	1.99	2.0

Table 2. Comparison for various recursion architectures

We have performed extensive simulations for Turbo codes using the original MAP and using various approximations. Fig. 10 shows the BER (bit-error-rate) performance of a rate-1/3, 8-state, block size of 512 bits, Turbo code using different MAP architectures. The simulations were undertaken under the assumption of AWGN channel and BPSK signaling. A maximum of 8 iterations was performed. More than 40 million random information bits were simulated for both Eb/No=1.6 dB and Eb/No=1.8dB cases. It can be noted from Fig. 10 that there is no observable performance difference between the true MAP algorithm and two approximation methods associated with the proposed recursion architectures while the approximation employed in Urard (2004) caused approximately 0.2 dB performance degradation in general.

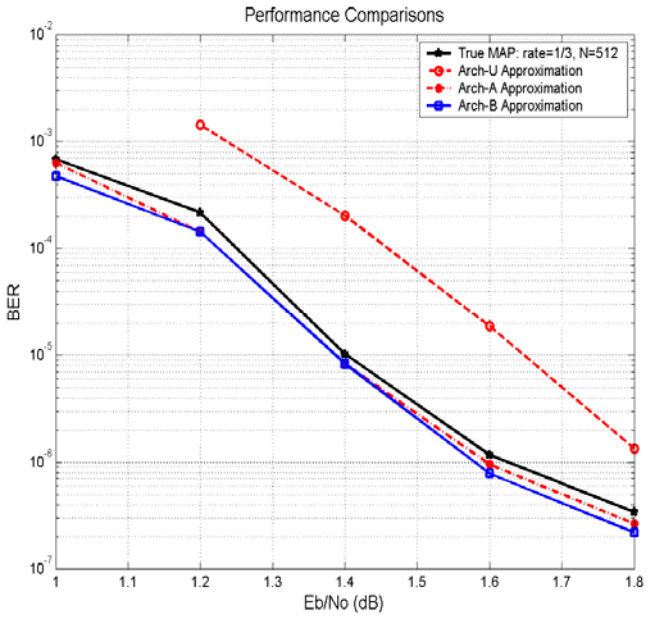


Fig. 10. Performance comparisons between the original MAP and some approximations.

We argue that the proposed Radix-4 recursion architecture is optimal for high-speed MAP decoders. Any (significantly) faster recursion architecture (*e.g.*, a possible Radix-8 architecture) will be at the expense of significantly increased hardware. On the other hand, when the target throughput is moderate, these fast recursion architectures can be used to reduce power consumption because of their dramatically reduced critical paths.

In general, the throughput of a serial Turbo decoder is significantly limited by recursive computation and VLSI technology used to implement it. On the other hand, increasing the clock frequency would cause the dynamic power dissipation to increase linearly. Therefore, it is not a good option to increase the clock frequency for high throughput applications.

4. Efficient Parallel Turbo Decoding Architectures

In this section, we first introduce an area-efficient parallel Turbo decoding scheme, where a data frame is partitioned into several equivalent segments with proper overlap between adjacent segments, and sliding-window-based Turbo decoding is then applied to each segment. The goal is to maintain the same memory requirement as serial decoding case while increasing the complexity of the logic components only, thus linearly increase the system throughput with sub-linearly increased hardware.

The major challenge lies in real implementation. As each memory (receiver memory or extrinsic information memory) needs to support multiple data (for multiple component soft-input soft-output decoders) at the same cycle, memory access conflicts are inevitable unless M-port (M = parallelism level) memory is used, which contradicts the original low-complexity design objective. Two different solutions are proposed in this work. First of all, if interleave patterns are free to design, we can adopt dividable interlavers, which inherently ensure no memory access conflict after proper memory partition. For practical applications wherein Turbo code interleave patterns are fixed, e.g., 3GPP and 3GPP2, a more generic solution is introduced in this chapter. By introducing some small buffers for data and addresses as well, we are able to avoid memory access conflict under any practical random interleavers. Combining all the proposed techniques, it is estimated that 200 Mb/s Turbo decoder is feasible with current CMOS technology with moderate complexity.

4.1 Area-efficient parallel decoding schemes

Parallel processing has been widely used in low power and high throughput circuit design. Multiple serial Turbo decoders can be concatenated in a parallel way (see Fig. 11 A) or in a serial way. In either case, both the area and power are increased linearly as the throughput increases. The area-efficient parallel Turbo decoding schemes were first proposed in Wang (2001). The idea is to let multiple SISO decoders work on the same data frame simultaneously. In this case, only the hardware of the SISO decoder part needs to be increased while the total hardware in the memory part remains unchanged on a large extent (see Fig. 11 B). As the memory part usually dominates the overall hardware of a Turbo decoder, the area-efficient parallel Turbo decoding architectures are expected to achieve multiple times the throughput of a serial decoding decoder with a small fraction of hardware overhead. Unfortunately, the real implementation issues were not covered in the original work.

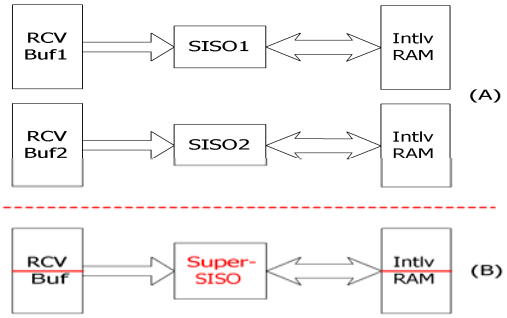
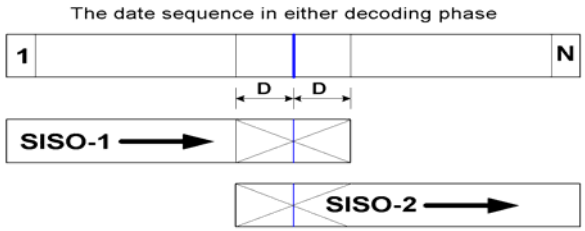


Fig. 11. Traditional vs. area-efficient parallel Turbo decoding.



a) A simple area-efficient 2-level parallel Turbo decoding scheme.

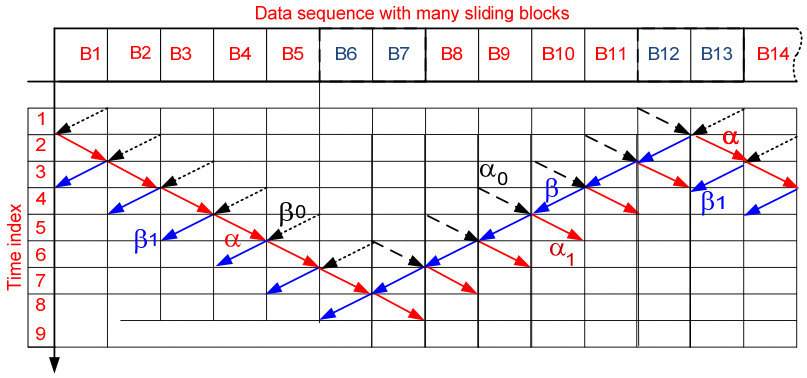


Fig. 12. b) Multi-level parallel Turbo decoding scheme based on sliding-window approach.

A simple area-efficient 2-level parallel Turbo decoding scheme is shown in Fig. 12 a), where a sequence of data is divided into two segments with equivalent length. There is an overlap with length of 2D between two segments, where D equals the sliding window size if Log-MAP (or MAX-Log-MAP) algorithm is employed in the SISO decoder or the survivor length if SOVA is employed, as proposed in Wang (2003c). The two SISO decoders work on two different data segments (with overlap) at the same time. Fig. 12 b) shows (a portion of) a sliding-window-based area-efficient multi-level parallel Turbo decoding data flow, where

SISO decoders responsible for two adjacent data segments are processing data in reverse directions in order to reuse some of the previously computed branch metrics and state metrics. For other parallel decoding schemes with various trade-offs, interested reader are referred to Wang (2001) and Zhang (2004).

In this section, we will focus on area-efficient two-level parallel decoding schemes. It can be observed from Fig. 12.a) that two data accesses per cycle are required for both the receiver buffer and the interleaver memory (assuming two ping-pong buffers are used for the interleaver memory). Using a dual-port memory is definitely not an efficient solution since a normal dual-port memory consumes as much area as two single-port memories with the same memory size.

4.2 Area-efficient parallel decoding architectures

In order to maintain the total hardware of these memories, we choose to partition each memory into multiple segments to support multiple data accesses per cycle.

Memory partitioning can be done in various ways. An easy and yet effective way is to partition the memory according to a number of least significant bits (lsb's) of the memory address. To partition a memory into two segments, it can be done according to the least significant bit (lsb). For the resultant two memory segments, one contains data with even addresses and the other contains data with odd addresses.

The memory partitioning can also be done in a fancy way, *e.g.*, to partition the memory into 2 segments, let the 1st memory segment contain data with addresses $b_2b_1b_0$ ={0, 2, 5, 7}, and the 2nd segment contain data with addresses $b_2b_1b_0$ ={1, 3, 4, or 6}, where $b_2b_1b_0$ denotes the 3 lsb's. Depending on the applications, different partitioning schemes may lead to different hardware requirements.

For the two memory segments, it is possible, in principle, to support two data accesses within one cycle. However, it is generally impossible to find an efficient partitioning scheme to ensure the target addresses (for both Load and Write operations) are always located in different segments at each cycle because the Turbo decoder processes the data in different orders during different decoding phases. Consider a simple example, given a sequential data sequence {0, 1, 2, 3, 4, 5, 6, 7}, the interleaved data sequence is assumed as {2, 5, 7, 3, 1, 0, 6, 4}. If we partition the memory into two segments according to the sequential decoding phase, *i.e.*, one segment contains data sequence {0, 1, 2, 3} and the other has {4, 5, 6 7}. During the sequential phase, SISO-1 works on data set {0, 1, 2, 3} and SISO-2 works on {4, 5, 6, 7}. So there is no data conflict (note: the overlap between two segments for parallel decoding is ignored in this simple example). However, during the interlaved decoding phase, SISO-1 will process data in set {2, 5, 7, 3} and SISO-2 will process data in set {1, 0, 6, 4}, both in sequential order. It is easy to see that, at the first cycle, both SISO decoders require data located in the first segment (1st and 2nd data in the input data sequence). Thus a memory access conflict occurs. More detailed analysis can be found in Wang (2003b). The memory access issue in parallel decoding can be much worse when multiple rates and

multiple block sizes of Turbo codes are supported in a specific application, *e.g.*, in 3GPP CDMA systems.

In principle, the data access conflict problem can be avoided if the Turbo interleaver is specifically designed. A generalized even-odd interleave is defined below:

- All even indexed data are interleaved to odd addresses,
- All odd indexed data are interleaved to even addresses.

Back to the previous example, an even-odd interleaver may have an interleaved data sequence as {3, 6, 7, 4, 1, 0, 5, 2}.

With an even-odd interleaver, we can partition each memory into two segments: one contains even-addressed data and the other with odd-addressed data.

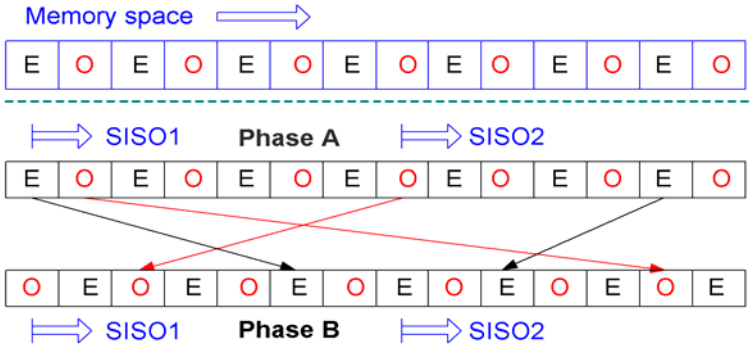


Fig. 13. Data processing in either decoding phase with an even-odd interleaver.

As shown in Fig. 13, the data are processed in an order of {even address, odd address, even , odd, …} in Phase A *(i.e.*, the sequential phase) and an order of {odd address, even address, odd, even, ..} in Phase B (*i.e.*, the interleaved phase). If we let SISO-1 and SISO-2 start processing from different memory segments, then there would be no data access conflict during either decoding phase. In other words, it is guaranteed that both SISO decoders always access data located in different memory segments.

More complicated interleavers can be designed to support multiple (M>=2) data accesses per cycle. The interested readers are referred to Wang (2003c), He (2005), Giulietti (2002), Bougard (2003), and Kwak (2003) for details.

In real applications, the Turbo interleaver is usually not free to design. For example, they are fixed in WCDMA and CDMA 2000 systems. Thus, a generic parallel decoding architecture is desired to accommodate all possible applications. Here we propose an efficient memory arbitration scheme to resolve the problem of data access conflict in parallel Turbo decoding.

The fundamental concept is to partition one single-port memory into S (S>=2) segments and use B (B >=2) small buffers to assist reading from or writing data back to the memory, where S does not have to be the same as B. For design simplicity, both S and B are normally chosen to be a power of 2. S is better chosen to be a multiple of B. In this chapter, we will consider

only one simple case, *i.e.*, S=2, B=2. For all other cases of B=2, they can be easily extended from the illustrated case. However, for cases with B>2, the control circuitry will be much more complicated.

For the receiver buffer, only the Load operation is involved while bother Load and Write operations are required for the interleaver memory. So we will focus our discussion on the interleaver memory.

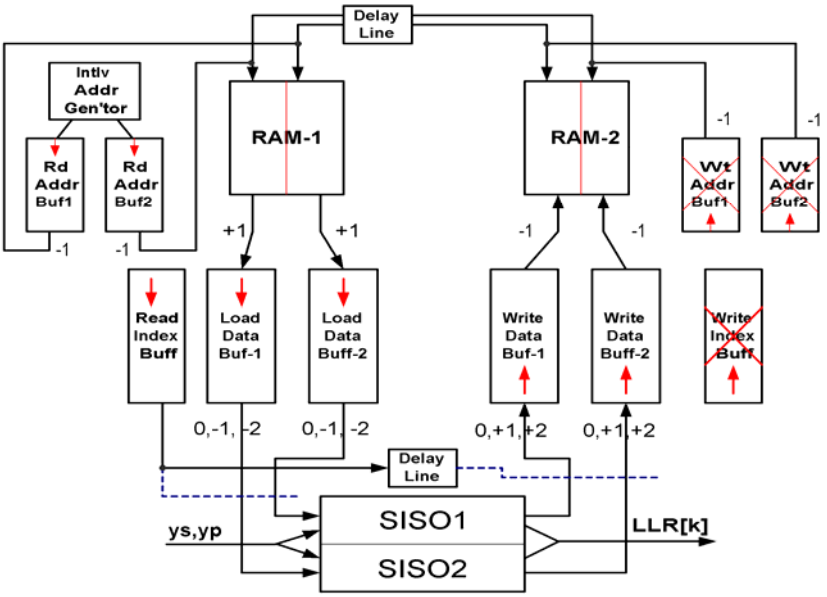


Fig. 14. The area-efficient parallel decoding architecture.

With regard to the interleaver memory, we assume two ping-pong buffers (i.e., RAM1 and RAM2 shown in Fig.14) are used to ensure that one Load and one Write operations can be completed within one cycle. Each buffer is partitioned into two segments: Seg-A contains even addressed data and Seg-B contains odd-addressed data. For simplicity, we use Seg-A1 to represent the even-addressed part of RAM1, Seg-B1 to represent the odd-addressed part of RAM1. The similar representations are used for RAM2 as well.

An area-efficient 2-parall Turbo decoding architecture is shown in Fig. 14. The interleaver address generator (IAG) generates two addresses for two SISO decoders respectively at each cycle. They must belong to one of the following cases: (1) two even addresses, (2) two odd addresses and (3) one even and one odd address. In Case 1, two addresses are put into Read Address Buffer 1 (RAB1), In Cases 2, two addresses are put in Read Address buffer 2 (RAB2). In Cases 3, the even address goes to RAB1 and the odd address goes to RAB2.

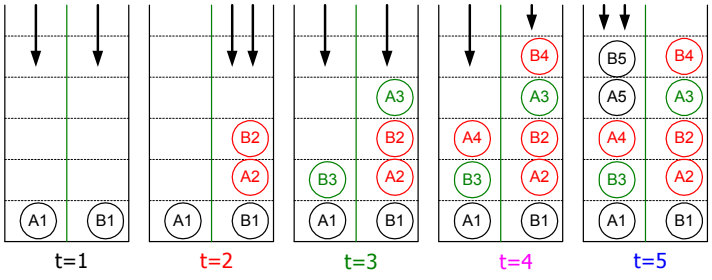
A small buffer called Read Index Buffer (RIB) is introduced in this architecture. This buffer is basically a FIFO (first-in first-out). Two bits are stored at each entry. The distinct four values represented by the two bits have following meanings:

- 00: 1st and 2nd even addresses for SISO-1 and SISO-2 respectively,
- 11: 1st and 2nd odd addresses for SISO-1 and SISO-2 respectively,
- 01: the even address is used for SISO-1 and the odd address for SISO-2,
- 10: the even address is used for SISO-1 and the odd address for SISO-2.

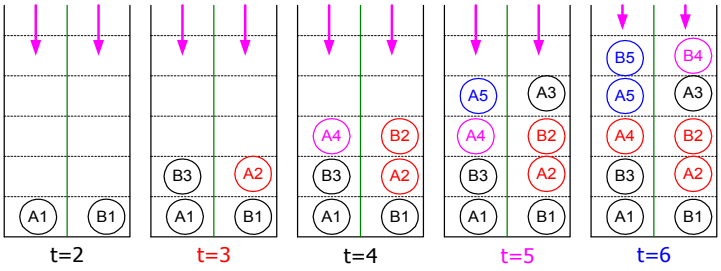
After decoding for a number of cycles (*e.g.*, 4 ~ 8 cycles), both RAB1 and RAB2 will have a small amount of data. Then the data from the top of RAB1 and RAB2 are used respectively as the addresses to Seg-A1 and Seg-B1 respectively. After one cycle, two previously stored extrinsic information symbols are loaded to Load Data Buffer 1 (LAB1) and Load Data Buffer 2 (LDB2). Then the data stored in RIB will be used to direct the outputs from both load buffers to feed both SISO decoders. For example, if the data at the top of RIB is 00, then LDB1 output two data to SISO-1 and SISO-2 respectively. The detailed actions of address buffers and data buffers controlled by RIB for loading data are summarized in Table 3.

Value of RIB	Action of address buffers	Action of data buffers
0	RAB1 load nothing, RAB2 loads 2 addresses, 1st for Seg-1 and 2nd for Seg-2	LDB1 output nothing, LDB2 out 2 data, 1st for SISO1 and 2nd for SISO2
1	RAB1 load 1 address for Seg-1, RAB2 load 1 address for Seg-2	LDB1 output 1 for SISO1 and LDB2 out 1 for SISO2
2	RAB2 load nothing, RAB1 load 2 addresses, 1st for Seg-1 and 2nd for Seg-2	LDB1 output 2 data, 1st for SISO1 and 2nd for SISO2
3	RAB1 load 1 address for Seg-1, RAB2 load 1 for Seg-2	LDB1 output 1 for SISO2 and LDB2 out 1 for SISO1

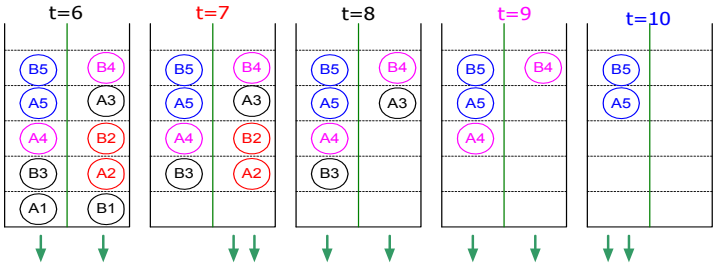
Table 3. Summary of loading data operations.



(a) load addresses to RAB1 and RAB2



(b) load data to RDB1 and RDB2



(c) output data to SISO1 and SISO2

Fig. 15. Procedure of loading data from memory to SISO decoders.

A simple example is shown in Fig. 15 to illustrate the details of loading data from memory to SISO decoders, where, for instance, A3 (B5) denotes the address of data to be loaded or the data itself corresponding to memory segment A (segment B) for the 3rd (5th) data in the processing sequence. Fig. 15 (a) shows the details of feeding two dada addresses to two address buffers at every cycle. Fig. 15 (b) shows the details of feeding one data to each data buffer at every cycle. The details of outputting the required two data to both SISO decoders are shown in Fig. 15 (c). As can be seen, both SISO1 and SISO2 can get their required data that may be located in the same memory segment at the same cycle (*e.g.*, t=7 in this example). A tiny drawback of this approach is that a small fixed latency is introduced. Fortunately this latency is negligible compared with the overall decoding cycles for a whole Turbo code block in most applications.

In general (*e.g.*, when sliding-window-based MAP algorithm is employed for SISO decoders), the write sequence for each SISO decoder is the delayed version of the read sequence. So the write index buffer and write address buffers can be avoided by using delay lines as shown in Fig. 14.

At each cycle, two extrinsic information symbols are generated from the two SISO decoders. They may both feed Write Data Buffer 1 (WDF1), or both feed Write Data Buffer 2 (WDF2), or one feeds WDF1 and the other feeds WDF2. The actual action is controlled by the delayed output from RIB.

Similar to loading data, after the same delay (*e.g.*, 4 ~ 8 cycles) from the first output of either SISO decoder, the data from WDB1 and WDB2 will be written back to Seg-B1 and Seg-B2 respectively. In the next decoding phase, RAM-1 and RAM-2 will exchange roles. So some extra MUXs must be employed to facilitate this functional switch.

5.3 Implementation issues and simulation results

The integer numbers shown in Fig. 14 indicated the data flow associated with each buffer. For example, {0, -1, -2} is shown along the output line of LDB1, which means LDB1 may output 0, 1, or 2 data at each cycle. It can be observed that all data and address buffers work in a similar way. One common feature of these buffers is that they have a constant data flow (+1 or –1) at one end (Load or Write) while having an irregular data flow ({0, -1, -2} or {0, +1, +2}) at the other end (Write or Load). On the average, incoming and outgoing data are largely balanced. So the buffer sizes can be very small.

The RIB can be implemented with a shift register as it has a regular data flow in both ends while a 3-port memory suffices to fulfill the functions of the rest buffers.

It has been found from our cycle-accurate simulations that it is sufficient to choose a buffer length of 25 for all buffers if the proposed 2-parallel architecture is applied in either WCDMA or CDMA2000 systems. The maximum Turbo block size for WCDMA system is approximately 5K bits. The lowest code rate is 1/3. Assume both the received soft inputs and the extrinsic information symbols are expressed as 6 bits per symbol.

- The overall memory requirement is 5K*(2+3)*6=150K bits.
- The total overhead of small buffers is approximately 25*3*2*(3*6+2*6) + 25*3*13*2~= 6K bits, where the factor 3 accounts for 3 ports of memory, 13 represents each address contains 13 binary bits.

It can be seen that the overhead is about 4% of total memory. With CDMA2000 systems, the overhead occupies an even smaller percentage in the total hardware because the maximum Turbo block size is several times larger.

For WCDMA systems, it is reasonable to assume the total area of a Log-MAP decoder consumes less than 15% of total hardware of a serial Turbo decoder. Thus, we can achieve twice the throughput with the proposed 2-parallel decoding architecture while spending less than 20% hardware overhead. If SOVA is employed in the SISO decoder, the overhead could be less than 15%. In either case, the percentage of the overhead will be even smaller in CDMA2000 systems.

Assume 6 iterations are performed at the most, the proposed 2-level parallel architecture, if implemented with TSMC 0.18 um technology, can achieve a minimum decoding throughput of 370 M*2/(6*2) > 60 Mbps. If state-of-the-art CMOS technology (*e.g.*, 65nm CMOS) is used, we can easily achieve 100Mbps data-rate with the proposed 2-parallel decoding architecture.

If an 4-parallel architecture is employed, over 200Mbps data rate can be obtained, though a direct extension of the proposed 2-paralell architecture will significantly complicate the control circuitry.

It is worthwhile to note, when the target throughput is moderately low, the proposed area-efficient parallel Turbo decoding architecture can be used for low power design. The reason is that the former architecture can work at a much lower clock frequency and the supply voltage can thus be reduced significantly.

6. Conclusions

Novel fast recursion architecture for Log-Map decoders have been presented in this chapter. Experimental results have shown that the proposed fast recursion architectures can increase the process speed significantly over traditional designs while maintaining the decoding performance. As a more power-efficient approach to increase the throughput of Turbo decoder, area-efficient parallel Turbo decoding schemes have been addressed. A hardware-efficient 2-parallel decoding architecture for generic applications is presented in detail. It has been shown that twice the throughput of a serial decoding architecture can be obtained with an overhead of less than 20% of an entire Turbo decoder. The proposed memory partitioning techniques together with the efficient memory arbitration schemes can be extended to multi-level parallel Turbo decoding architectures as well.

7. References

3rd Generation Partnership Project (3GPP), Technical specification group radio access network, multiplexing and channel coding (TS 25.212 version 3.0.0), http://www.3gpp.org.

3rd Generation Partnership Project 2 (3GPP2), http://www.3gpp2.org.

A. Giulietti *et al.* (2002), Parallel Turbo code interleavers: Avoiding collisions in accesses to storage elements, *Electron. Lett.*, vol. 38, no. 5, pp. 232–234, Feb. 2002.

A. J. Viterbi. (1998). An intuitive justification of the MAP decoder for convolutional codes, *IEEE J. Select. Areas Commun.*, vol.16, pp. 260-264, February 1998.

A. Raghupathy. (1998). Low power and high speed algorithms and VLSI architectures for error control coding and adaptive video scaling, Ph.D. dissertation, *Univ. of Maryland, College Park*, 1998.

B. Bougard *et al.* (2003). A scalable 8.7 nJ/bit 75.6 Mb/s parallel concatenated convolutional (Turbo-) codec, in *IEEE ISSCC Dig. Tech. Papers*, 2003, pp. 152–153.

C. Berrou, A. Clavieux & P. Thitimajshia. (1993). Near Shannon limit error correcting coding and decoding: Turbo codes, *ICC'93*, pp 1064-70.

E. Boutillon, W. Gross & P. Gulak. (2003). VLSI architectures for the MAP algorithm, *IEEE Trans. Commun.*, Volume 51, Issue 2, Feb. 2003, pp. 175 – 185.

E. Yeo, S. Augsburger, W. Davis & B. Nikolic. (2003). A 500-Mb/s soft-output Viterbi decoder, IEEE Journal of Solid-State Circuits, Volume 38, Issue 7, July 2003, pp:1234 – 1241.

H. Suzuki, Z. Wang & K. K. Parhi. (2000). A K=3, 2Mbps Low Power Turbo Decoder for 3rd Generation W-CDMA Systems, in *Proc. IEEE 1999 Custom Integrated Circuits Conf. (CICC)*, 2000, pp 39-42.

J. Hagenauer & P. Hoher. (1989). A Viterbi algorithm with soft decision outputs and its applications, *IEEE GLOBECOM*, Dallas, TX, USA, Nov. 1989, pp 47.1.1-7.

J. Kwak & K Lee (2002). Design of dividable interleaver for parallel decoding in turbo codes. *Electronics Letters*, vol. 38, issue 22, pp. 1362-64, Oct. 2002.

J. Kwak, S. M. Park, S. S. Yoon & K Lee (2003). Implementation of a parallel Turbo decoder with dividable interleaver, *ISCAS'03*, vol. 2, pp. II-65- II68, May 2003.

K. K. Parhi. (1999). VLSI Digital signal Processing Systems, *John Wiley & Sons*, 1999.

L.Bahl, J.Jelinek, J.Raviv & F.Raviv. (1974). Optimal Decoding of Linear Codes for minimizing symbol error rate, *IEEE Trans.s Inf. Theory*, vol. IT-20, pp.284-287, March 1974.

M. Bickerstaff, L. Davis, C. Thomas, D. Garret & C. Nicol. (2003). A 24 Mb/s radix-4 LogMAP Turbo decoder for 3 GPP-HSDPA mobile wireless, in *Proc. IEEE ISSCC Dig. Tech. Papers*, 2003, pp. 150–151.

P. Urard et al. (2004). A generic 350 Mb/s Turbo codec based on a 16-state Turbo decoder, in *Proc. IEEE ISSCC Dig. Tech. Papers*, 2004, pp. 424–433.

S. Lee, N. Shanbhag & A. Singer. (2005). A 285-MHz pipelined MAP decoder in 0.18 um CMOS, *IEEE J. Solid-State Circuits*, vol. 40, no. 8, Aug. 2005, pp. 1718 – 1725.

T. Miyauchi, K. Yamamoto & T. Yokokawa. (2001). High-performance programmable SISO decoder VLSI implementation for decoding Turbo codes, in *Proc. IEEE Global Telecommunications Conf.*, vol. 1, 2001, pp. 305–309.

T.C. Denk & K.K. Parhi. (1998). Exhaustive Scheduling and Retiming of Digital Signal Processing Systems, *IEEE Trans. Circuits and Syst. II*, vol. 45, no.7, pp. 821-838, July 1998

W. Gross & P. G. Gulak. (1998). Simplified MAP algorithm suitable for implementation of Turbo decoders, *Electronics Letters*, vol. 34, no. 16, pp. 1577-78, Aug. 1998.

Y. Wang, C. Pai & X. Song. (2002). The design of hybrid carry-lookahead/carry-select adders, *IEEE Trans. Circuits and Syst. II*, vol.49, no.1, Jan. 2002, pp. 16 -24

Y. Wu, B. D. Woerner & T. K. Blankenship, Data width requirement in SISO decoding with module normalization, *IEEE Trans. Commun.*, vol. 49, no. 11, pp. 1861–1868, Nov. 2001.

Y. Zhang & K. Parhi (2004). Parallel Turbo decoding, Proceedings of the 2004 International Symposium on Circuits and Systems (ISCAS04), Volume 2, 23-26 May 2004, pp: II - 509-12 Vol.2.

Z Wang, Y. Tan & Y. Wang. (2003b). Low Hardware Complexity Parallel Turbo Decoder Architecture, *ISCAS'2003*, vol. II, pp. 53-56, May 2003.

Z. He, S. Roy & Fortier (2005). High-Speed and Low-Power Design of Parallel Turbo Decoder Circuits and Systems, *IEEE International Symposium on Circuits and Systems (ISCAS'05)*, 23-26 May 2005, pp. 6018 – 6021.

Z. Wang & K. Parhi. (2003a). Efficient Interleaver Memory Architectures for Serial Turbo Decoding, *ICASSP'2003*, vol. II, pp. 629-32, May 2003.

Z. Wang & K. Parhi. (2003c). High Performance, High Throughput Turbo/SOVA Decoder Design, *IEEE Trans. on Commun.*, vol. 51, no 4, April 2003, pp. 570-79.

Z. Wang, H. Suzuki & K. K. Parhi. (1999). VLSI implementation issues of Turbo decoder design for wireless applications, in *Proc. IEEE Workshop on Signal Process. Syst. (SiPS)*, 1999, pp. 503-512.

Z. Wang, Z. Chi & K. K. Parhi. (2001). Area-Efficient High Speed Decoding Schemes for Turbo/MAP Decoders, in *Proc. IEEE ICASSP'2001*, pp 2633-36, vol. 4, Salt Lake City, Utah, 2001.

Z. Wang. (2000). Low complexity, high performance Turbo decoder design, Ph.D. dissertation, *University of Minnesota*, Aug. 2000.

Z. Wang. (2007). High-Speed Recursion Architectures for MAP-Based Turbo Decoders, in *IEEE Trans. on VLSI Syst.*, vol. 15, issue 4, pp: 470-74, Apr. 2007.

9

Ultra-High Speed LDPC Code Design and Implementation

Jin Sha[1], Zhongfeng Wang[2] and Minglun Gao[1]
[1]*Nanjing University,* [2]*Broadcom Corporation*
[1]*China,* [2]*U.S.A.*

1. Introduction

The digital communications are ubiquitous and have provided tremendous benefits to everyday life. Error Correction Codes (ECC) are widely applied in modern digital communication systems. Low-Density Parity-Check (LDPC) code, invented by Gallager (1962) and rediscovered by MacKay (1996), is one of the two most promising near-optimal error correction codes in practice. Since its rediscovery, significant improvements have been achieved on the design and analysis of LDPC codes to further enhance the communication system performance. Due to its outstanding error-correcting performance (Chung 2001), LDPC code has been widely considered in next generation communication standards such as IEEE 802.16e, IEEE 802.3an, IEEE 802.11n, and DVB-S2. LDPC code is characterized by sparse parity check matrix. One key feature associated with LDPC code is the iterative decoding process, which enables LDPC decoder to achieve outstanding performance with moderate complexity. However, the iterative process directly leads to large hardware consumption and low throughput. Thus efficient Very Large Scale Integration (VLSI) implementation of high data rate LDPC decoder is very challenging and critical in practical applications.

A satisfying LDPC decoder usually means: good error correction performance, low hardware complexity and high throughput. There are various existing design methods which usually encounter the limitations of high routing overhead, large message memory requirement or long decoding latency. To implement the decoder directly in its inherent parallel manner may get the highest decoding throughput. But for large codeword lengths (e.g., larger than 1000 bits), to avoid routing conflict, the complex interconnection may take up more than half of the chip area. Both serial and partly parallel VLSI architectures are well studied nowadays. However, none of these approaches are good for very high throughput (i.e., multi-Gb/s) applications.

In this chapter, we present the construction of a new class of implementation-oriented LDPC codes, namely shift-LDPC codes to solve these issues integrally. Shift-LDPC codes have been shown to perform as well as computer generated random codes following some optimization rules. A specific high-speed decoder architecture targeting for multi-Gb/s applications namely shift decoder architecture, is developed for this class of codes. In

contrast with conventional decoder architectures, the shift decoder architecture has three major merits:

1) Memory efficient. By exploring the special features of the min-sum decoding algorithm, the proposed architecture stores the message in a compact way which normally leads to approximately 50% savings on message memory over the conventional design for high rate codes.

2) Low routing complexity. Through introducing some novel check node information transfer mechanisms, the complex message passing between variable nodes and check nodes can be alleviated by the regular communication between check nodes and thus the complex global interconnection networks can be replaced by local wires. One important fact is that in the new architecture, check node processing units have few and fixed connections with variable node processing units.

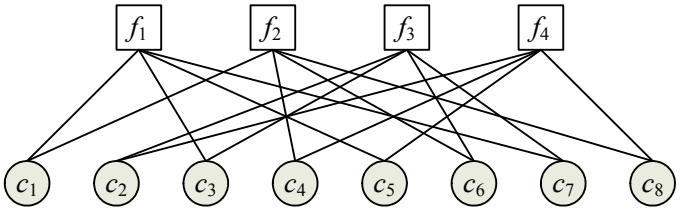
3) High parallel level of decoding. The architecture can normally exploit more levels of parallelism in the decoding algorithm than conventional partially parallel decoder architectures. In addition, the decoding parallel level can be further linearly increased and the critical path can be significantly reduced by proper pipelining.

The chapter is organized as follows. Section 2 gives an overview of LDPC codes, with the main attention being paid to the decoding algorithm and decoder architecture design. Section 3 introduces the code construction of shift-LDPC codes and Section 4 presents the decoder design with shift decoder architecture and demonstrates the benefits of proposed techniques. In Section 5, we consider the application of shift architecture to some well known LDPC codes such as RS-based LDPC codes and QC-LDPC codes. Section 6 concludes the chapter.

2. Review of LDPC codes

2.1 LDPC decoding algorithms

LDPC codes are a set of linear block codes corresponding to the $M \times N$ parity check matrix H, which has very low density of 1's. M is the parity check number and N is the codeword length. With the aid of the bipartite graph called Tanner graph (Tanner 1981), LDPC codes can be effectively represented. There are two kinds of nodes in Tanner graph, variable nodes and check nodes, corresponding to the columns and rows of parity check matrix respectively. Check node f_i is connected to variable node c_j only when the element h_{ij} of H is a 1. Fig.1 shows the parity check matrix H of an (8, 4) regular LDPC code and its Tanner graph. A LDPC code is called (c, t) regular if the number of 1's in every column is a constant c and the number of 1's in every row is also a constant t.



(a)

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

(b)

Fig. 1. An LDPC code example. (a) Tanner graph. (b) Parity check matrix.

The typical LDPC decoding algorithm is the Sum-Product (or belief propagation) algorithm. After variable nodes are initialized with the channel information, the decoding messages are iteratively computed by all the variable nodes and check nodes and exchanged through the edges between the neighbouring nodes (Kschischang 2004).

The modified min-sum decoding algorithm studied in Guilloud (2003), Chen (2005) and Zhao (2005) is similar to the Sum-Product algorithm, with a simplification of check node process. It has some advantages in implementation over the Sum-Product algorithm, such as less computation complexity and no requirement of knowledge of SNR (signal-to-noise ratio) for AWGN channels. In the modified min-sum decoding algorithm, the check node computes the check-to-variable messages R_{cv} as follows:

$$R_{cv} = \alpha \times \prod_{n \in N(c) \setminus v} sign\ (L_{vc}) \times \min_{n \in N(c) \setminus v} |L_{vc}| \tag{1}$$

where a is a scaling factor around 0.75, L_{vc} is the variable-to-check messages, $N(c)$ denotes the set of variable nodes that participate in c-th check node. The variable node computes the variable-to-check messages L_{vc} as the following:

$$L_{vc} = \sum_{m \in M(v) \setminus c} R_{mv} + I_v \tag{2}$$

where $M(v) \setminus c$ denotes the set of check nodes connected to the variable node v excluding the variable node c, I_v denotes the intrinsic message of variable node v.

2.2 Implementation options

From the decoding algorithm described above, it can be seen that the calculations between different variable nodes or check nodes are independent at each iteration. Therefore, LDPC codes are especially suitable for parallel implementation. In addition, the computation in the decoder is fairly simple, which means low logic consumption. The main obstacle in the implementation of fully parallel LDPC decoders is the complicated interconnection between variable node processing units (VNU) and check node processing units (CNU). This interconnection complexity is due to the random-like locations of ones in the code's parity-check matrix and it is a crucial problem for practical fully parallel decoders when the code block length is large. The routing congestion or interconnection problem will arise and thus causes high area consumption and low logic utilization in the decoder. For instance, with 4-bit precision of probability messages, the 52.5 mm² die size of the (1024, 512) decoder in Blanksby (2002) has a logic utilization of 50% in its core and the rest of the core area is occupied by wires. There are some approaches developed to alleviate this routing

congestion problem. Darabiha (2008) uses bit-serial architectures and message broadcasting technique to reduce the number of wires in fully parallel decoders. Sharifi Tehrani (2008) presents a stochastic decoder design where the messages or probabilities are converted to streams of stochastic bits and complex probability operations can be performed on stochastic bits using simple bit-serial structures.

The more general methods to alleviate the routing problem is through reducing the parallelism (by using partially parallel or serial processing) and using storage elements to store the intermediate messages passed along the edges of the graph (e.g., see Mansour 2002, Yeo 2003, Cocco 2004). Various approaches are investigated in the literature at both code design and hardware implementation levels. One approach is to design "implementation-aware" codes (e.g., see Boutillon 2000, Zhang 2002, Mansour 2003a, Liao 2004, Zhang 2004, Sha 2009). In this approach, instead of randomly choosing the locations of ones in the parity-check matrix at the code design stage, the parity-check matrix of an LDPC code is decided with constraints allowing a suitable structure for decoder implementation and providing acceptable decoding performance. In these cases, the problem becomes how to reduce the complexity incurred by the message storage elements and how to speedup the decoding process. For example, one important subclass of LDPC codes with regular structure is quasi-cyclic (QC) LDPC codes which has received the most attentions and has been selected by many industry standards. Due to their regular structure, QC-LDPC codes lend themselves conveniently to efficient hardware implementations. Not only the partly parallel decoder architectures for QC-LDPC codes require simpler logic control (e.g., see Chen 2004b, Wang 2007) but also the encoders can be efficiently built with shift registers (e.g., see Li 2006). Construction of QC-LDPC codes with good error performances is therefore of both theoretical and practical interest. Various construction methods of QC-LDPC codes have been proposed and the satisfying error correcting performances are reported (e.g., see Fossorier 2004, Chen 2004a).

On the other hand, LDPC decoders can be implemented with a programmable architecture or processor, which lend themselves to Software Defined Radio (SDR). SDR offers flexibility to support codes with different block lengths and rates, however, Seo (2007) shows that the throughput of SDR-based LDPC decoders is usually low. In addition to digital decoders, continuous time analog implementations have also been considered for LDPC codes. Compared to their digital counterparts, analog decoders offer improvements in speed or power. However, because of the complex and technology-dependent design process, the analog approach has been only considered for very short error-correcting codes, for example, the (32, 8) LDPC code in Hemati (2006).

In the following, a new ensemble of LDPC codes, called shift-LDPC codes, will be introduced to mitigate the above decoder implementation problems.

3. Shift-LDPC Code Construction

3.1 Construction of Shift-LDPC Code

Shift-LDPC code is constructed in a well-regulated way. A regular *(N, M) (c, t)* shift LDPC code example is shown in Fig. 1, with $N=t\times q$ and $M=c\times q$, where each submatrix has a dimension of $q\times q$. As displayed, the parity check matrix consists of $c\times t$ submatrices. The structures of the leftmost c submatrices H_{i1} ($1 \le i \le c$) are random column permutations of the identity matrix. The other submatrices are decided by these H_{i1} submatrices.

$$H = \begin{bmatrix} H_{11} & H_{12} & \cdots & H_{1t} \\ H_{21} & H_{22} & \cdots & H_{2t} \\ \cdots & \cdots & \ddots & \cdots \\ H_{c1} & H_{c2} & \cdots & H_{ct} \end{bmatrix}$$

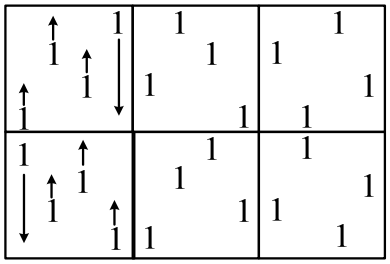


Fig. 2. The construction of shift-LDPC parity check matrix and an example of the (2,3) shift-LDPC code

The 1's in the shift LDPC code parity check matrix are arranged as the example in Fig. 1. At first, the 1's in the leftmost submatrices are arranged randomly under the constraint of keeping one "1" in each row and one "1" in each column. Thus the submatrix H_{i1} is a permutation matrix of identity matrix. Next, for each submatrix on the right hand side, the 1's are cyclic-shifted up by 1 space. The "1" at the top is moved down to the bottom. Finally we can get a (c, t) regular shift-LDPC code.

It should be noted that shift-LDPC code is a kind of code specially designed for hardware implementation. In fact, the shift decoder architecture is designed first and then the code construction is found. This follows the "decoder-first code design" methodology proposed in Boutillon (2000).

3.2 Code examples and performance

Given a fixed block length, column weight, and row weight, the ensemble of shift-LDPC codes can have considerable variations in performance. An effective method to find good LDPC codes is to find the codes with large girth. Girth is referred to the length of the shortest cycle in a Tanner graph. A large girth generally improves the bit error performance of the codes. Hence, LDPC codes with large girth are particularly desired. By exploiting the structured property of shift-LDPC codes, an efficient girth optimization algorithm is developed. Normally, by using this girth optimization soft program, the cycle-4 can be eliminated, and the cycle-6 can be avoided for moderate code rate shift-LDPC codes.

Through extensive simulation, we found that the performance of the optimized shift LDPC codes can be comparable to the computer generated random codes. Fig. 2 shows the performance comparison between a girth optimized (1008, 504) (3, 6) shift-LDPC code, the same size rate-1/2 code downloaded from Mackey's website and a (960,480) irregular WiMAX code as well as comparison between a (8192, 7168) (4, 32)-regular shift-LDPC code and a girth optimized QC-LDPC code. It can be seen that the WiMAX code performs better due to its irregular property, and in other cases the shift-LDPC codes have comparable performance. The performance of shift-LDPC codes could be further improved by introducing limited irregularity, e.g., to zero out some of the submatrices, which will cause tiny hardware overhead.

Shift-LDPC codes are a kind of implementation oriented codes and are constructed after the decoder design. Thus in this chapter we will focus on the efficient decoder design instead of the code performance. On the other hand, the shift decoder architecture can be applied to a large number of known structured LDPC codes such as quasi cyclic LDPC codes and RS

based LDPC codes. For example, QC-LDPC codes can be converted to shift architecture compliant code with proper matrix permutation. It will be explained later in the chapter.

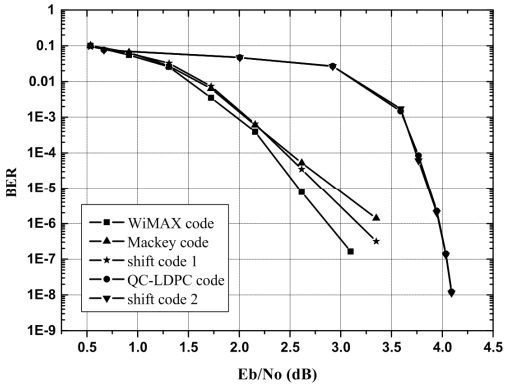


Fig. 3. Performance of (1008, 504) (3, 6) shift-LDPC code and (8192, 7168) (4, 32) shift-LDPC code compared with the same length WiMAX standard code and computer generated random codes

4. Shift LPDC Decoder Architecture

In this section, we present the shift decoder architecture specifically designed for shift-LDPC codes.

4.1 Decoding schedule

Firstly, let us discuss about the decoding schedule. For a regular (N, M) (c, t) shift-LDPC code, we assume q variable node processing units and M check node processing units are instantiated in the decoder where q is the size of submatrix (q=N/t=M/c). Fig. 3 illustrates the decoding flow for a simple shift-LDPC code example with q=4. The schedule can be applied to other cases with different q, c, and t parameters.

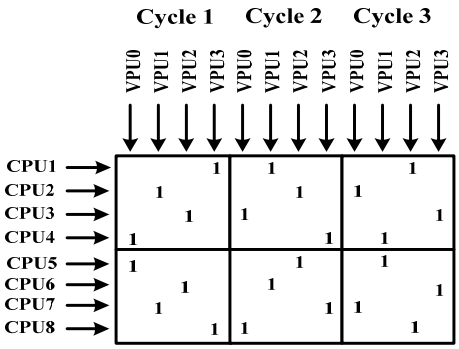


Fig. 4. Decoding schedule for a sample shift-LDPC code

The leftmost q columns are processed first, then the second left-most q columns, and so on. So q columns are processed concurrently in one clock cycle. The column process and the row process are interleaved. The whole check node process is divided into t steps. In each clock cycle, q VNUs get M check-to-variable messages and compute the M variable-to-check messages, so that M CNUs get one message each, so each CNU can deal with one step of the check node process. With this decoding schedule, we can finish one iteration in t clock cycles. It is normally much faster than the traditional partly parallel decoder architectures.
By combining the decoding schedule and the min-sum algorithm, the decoding process can be expressed as below:

Min-Sum Algorithm and Decoding Schedule

1: Initialization: $L_{vc} = I_v,\ i = 0,1,\ldots,N-1;$

2: repeat

3: for $k = 0$ **to** $k = d_c$ -1 **do**

4: { process of kq th $\sim (k+1)q$ th columns }

4: *compute* R_{cv} *from row process result of last iteration*

5: *magnitude of* R_{cv} *= minimum or 2nd-minimum*

6: *sign of* $R_{cv} = \prod_{n \in N(c)} sign(L_{nc}) \times sign(L_{vc})$

7: *for q columns process* :

8: $L_{vc} = \sum_{m \in M(v)} R_{mv} + I_v - R_{cv}$

9: *for all check nodes* (*receive one* L_{vc} *per row*) :

10: *update minimum*; *2nd-minimum*; *location of minimum*;

11: *record signs*

12: endfor

13: until *max iteration times reached or convergence to a codeword*

14: Output: *decoded bit*

Table 1. Decoding schedule designed for shift decoder architecture

4.2 Overall decoder architecture

The overall decoder block diagram is shown in Fig. 4. q variable node processing units and M check node processing units are instantiated in the decoder. The critical part of this implementation is the network connecting VNUs and CNUs. The shuffle network A transmits variable-to-check messages; the shuffle network B transmits check-to-variable messages. Shuffle network B is the same as shuffle network A except that the data flow is in the reverse direction.
In LDPC decoding algorithms, CNUs only communicate with VNUs. The row processes of different check nodes are independent of each other and so do the variable nodes processes. In shift-LDPC decoder architecture, a novel block called *CNU communication network* is added into the decoder. This new block brings some communications between CNUs. By introducing these originally unwanted communications, the complexity of connections

between CNUs and VNUs and thus the complexity of shuffle networks can be significantly reduced. This will be shown next.

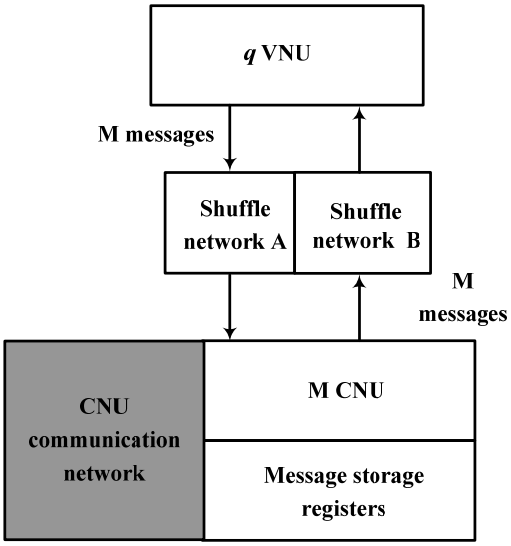


Fig. 5. Overall shift decoder block diagram. Massages are iteratively exchanged between VNUs and CNUs through the shuffle network

For random codes, the shuffle network routing complexity is normally intolerable, while for shift-LDPC codes we introduced above, the shuffle networks become really simple. Through the introduction of the *CNU communication network*, we can ensure that each CNU processes the variable-to-check messages from and transmit the check-to-variable messages to a fixed VNU during the entire decoding process. Therefore, the shuffle network connecting CNUs and VNUs only consists of $M*b$ wires, where we assume each message is quantized as b bits. (Normally the quantization bits b is chosen as 6.) In contrast, the number of wires required by original LDPC decoding algorithm is $M*t*b$. For high rate codes, for example, t can be as large as 32.

Fig. 5 explains why each CNU always computes with the messages from a fixed VNU during the entire decoding process. With the simple shuffle network, CNU i is connected with VNU x. The row process of each check node is separated into t steps with one variable-to-check message being processed in each step. At the first clock cycle of an iteration, having received the message from VNU x, CNU i performs the first step of i-th row process. After that, CNU i passes the i-th row process intermediate result to CNU i+1 through the *CNU communication network*. Meanwhile, CNU i receives the (i-1)-th row process intermediate result from CNU i-1. In the second clock cycle of the iteration, CNU i still receives the variable-to-check message from VNU x. This message and the row process intermediate result in CNU i are arranged both corresponding to row i-1. Therefore, CNU i can perform the second step of (i-1)-th row process. At the same time, CNU i+1 is performing the second step of i-th row process. For variable node processing, VNU x receives the check-to-variable

message in sequential from row i to i-1, i-2, …, etc. In the same way, the *CNU communication network* also can ensure that VNU x only need to receive its message from CNU i.

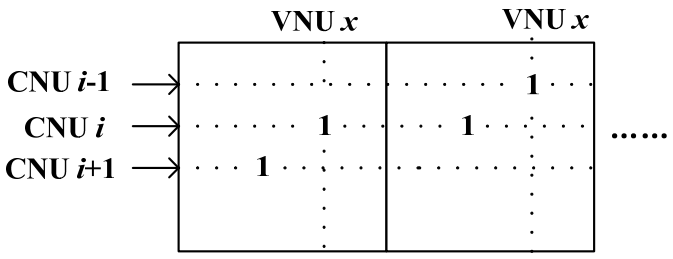


Fig. 6. Data flow of row process intermediate result

To show the decoding procedure more clearly, we illustrate the whole process with the simple (12, 4) (2, 3) shift LDPC code presented in Fig. 1 as an example. The procedure for one entire iteration is shown in Fig. 6. It can be seen that the *CNU communication network* is separated into two independent networks: *CNU communication network (intra iteration)* and *CNU communication network (inter iterations)* which transfer the row processing results during iteration and between iteration respectively.

The connections between CNUs and VNUs are fixed and simplified. CNU 1 connects with VNU 4; CNU 2 connects with VNU 2…These connections are based on the 1's positions in the leftmost submatrices.

One time iteration consists of three steps or three cycles (the row weight t equals 3). In each step/clock cycle, each CNU generates a check-to-variable message, passes the message to its connected VNU and processes a variable-to-check message which is received from its connected VNU. At the starting of iteration, in the first clock cycle, CNU 1 process the data of row 1; CNU 2 process the data of row 2. After this one step row process, the row process intermediate results are shifted between CNUs through the *CNU communication network (intra iteration)*. The result according to row 1 is passed to CNU 2; the result according to row 2 is passed to CNU 3, etc.

In the second clock cycle, CNU 2 still transmits to and gets message from VNU 2. This time the variable-to-check message it gets is corresponding to row 1, thus it can deal with one step of row 1 process. Likely CNU 3 performs one step of row 2 process; CNU 4 performs one step of row 3 process; etc. The third step is performed in the same way.

Once the iteration is finished, the row process results will be transferred back to appropriate CNUs for the check-to-variable messages generation in next iteration. The results are transferred through the *CNU communication network (inter iterations)*. This network is a one-to-one communication network as shown in Fig. 6.

As a result, the communication between CNUs and VNUs required by the original LDPC decoding algorithm is decomposed into three kinds of connections: the simplified connection between CNUs and VNUs; the *CNU communication network (intra iteration)* and the *CNU communication network (inter iterations)*.

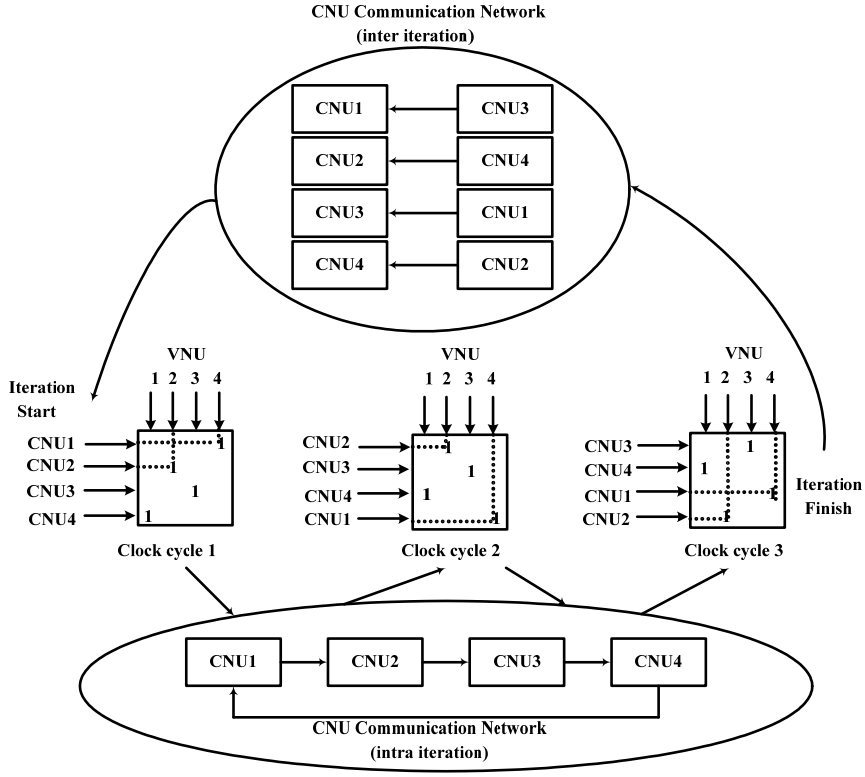


Fig. 7. Decoder scheduling of the simple shift-LDPC code in Fig. 1

In the following, we will introduce the architecture of CNU applying the min-sum decoding algorithm. The (8192, 7168) (4, 32) shift LDPC code is chosen as the design example.

4.3 Architecture of check node processing unit

The check node processing unit executes the check-to-variable message computation and the magnitude comparison between the variable-to-check message and the intermediate result of row process. Fig. 7 shows the architecture of CNU. The old register, new register, and sign register store the row process results of last iteration, the intermediate results of row process, and the sign bits of the check-to-variable messages respectively. To manage the message storage more efficiently, only row process results are stored in a compressed way that only the minimum magnitude, 2nd-minimum magnitude, the location of the minimum and the sign bits are saved.

The data in new registers and old registers are always being passed between CNUs through the *CNU communication network (intra iteration and inter iterations)*. The signs in the sign register are from the same VNU, so they do not need to be passed. In each clock cycle, sign register shifts in a sign bit of the variable-to-check message and shifts out a sign bit of a check-to-variable message.

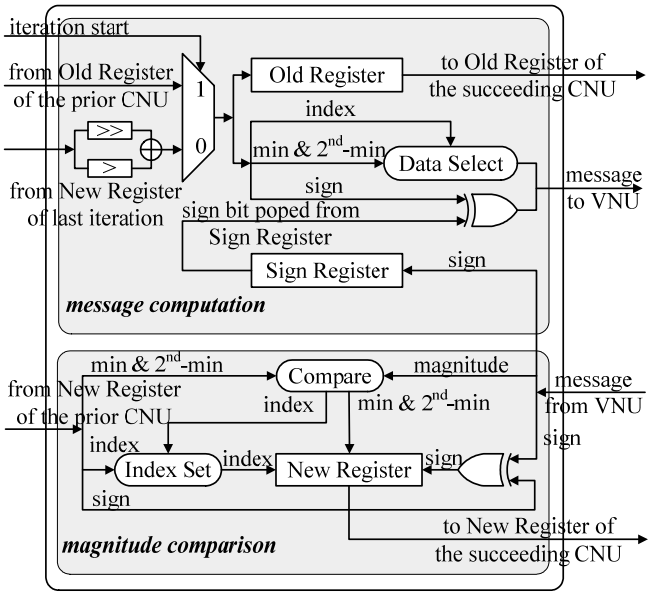


Fig. 8. Architecture of check node processing unit

Because only one variable-to-check message is processed in one time, the computation in CNU has very low complexity. The magnitude comparison part compares the magnitude of input message with the current row process intermediate result to update the magnitudes, sign and index. Then the updated row process intermediate result is registered and passed to its next CNU neighbour through the *CNU communication network (intra iteration)*. The message computation part selects the proper message magnitude according to the index value, and computes the sign of the message. At the beginning of each iteration, it performs the message scale calculation (a = 0.75).

4.4 Architecture of variable node processing unit

The architecture of variable node processing unit is the same with other LDPC decoder designs. It can be designed with only combinational logic. Fig. 8 shows the VNU architecture. The inputted check-to-variable messages are firstly converted from sign-magnitude format to two's complement format. Then the adder tree computes the variable-to-check messages according to equation (2). Finally the messages are converted back to sign-magnitude format and transmitted to CNUs.

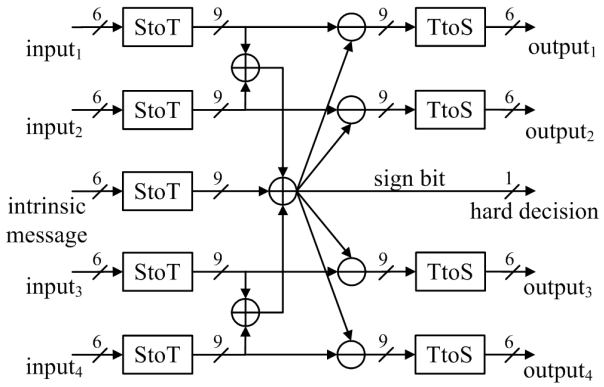


Fig. 9. Architecture of the variable node processing unit. StoT module converts from the sign-magnitude format to two's complement format. TtoS is for reverse conversion

4.5 Implementation results

To further increase the decoding speed, more parallel levels can be added to the decoder architecture. For example, to double the parallel level, 2**q* VNUs will be instantiated. The architecture of CNU should be changed to process two messages per clock cycle. Then each iteration can be finished in *t*/2 clock cycles. A double parallel level decoder example is designed for the (8192, 7168) (4, 32) shift LDPC code to investigate the tradeoff between hardware cost and decoding speed.

In addition, to increase the clock frequency, some pipeline levels can be added to the data path. The critical path is from old register, through CNU message computation part, to VNU computation, to CNU row processing part, and finally ends at the new register. Pipeline will increase the number of clock cycles needed for each iteration. For example, a three stage pipeline will add two clock cycles to each iteration.

Gate count	CNU combination logic	VNU combination logic	total combination logic	frequency (MHz)	Throughput per iteration (Gbps)	Message area (mm^2)	Total area (mm^2)
One parallel level design	148	953	395K	160	40.96	6.1	10
One parallel + pipelining	148	1410	512K	317	76.4 ★	6.1	11.3
Double parallel level	342	953	838K	151	77.3	6.1	14
Double parallel + pipelining	342	1410	1072K	317	153	6.1	16.7

Table 2. Synthesis result for (8192, 7168) (4, 32) shift LDPC decoder under 0.18μm technology

Table 2 shows the synthesis result of four design examples: one-level parallel design, one-level parallel with pipelining, two-level parallel design, and two-level parallel with

pipelining. Due to the large number of CNUs and VNUs (submatrix size q = 256), the bottom-up synthesis strategy is applied. These results are achieved with 0.18μm technology. It can be seen that, by applying pipelining, the clock frequency can be almost doubled, and the overhead is 81 registers per CNU. The one-level parallel design with pipelining can achieve the best trade-off between speed and hardware complexity.

For the message storage, it is implemented with registers. The transferring messages take (16*2+32)*1024 = 65536 bits. In comparison, the traditional partially parallel architecture (e.g., Wang (2007)) needs to store 8192*4*6 = 196224 bits in total. It saves 67% of the message memory needed. The initial intrinsic information takes 8192*6 = 49152 bits. It is the same for both cases.

To further examine the efficiency of the shift decoder architecture, backend design of the sample (8192, 7168) (4, 32) shift LDPC decoder is completed. The technology used for implementation is a 0.18μm CMOS process with 6 metal layers. Fig. 9 shows the floor-plan and layout of the decoder chip with a die size of 4.1 mm ×4.1mm. The logic density is 70%. The placement of CNUs is critical to reducing the routing congestion of CNU communication networks. The CNU array located in the centre of the chip is specially arranged: the 1024 CNUs are aligned with 31 CNUs per row, so that the communications between CNUs can be locally routed.

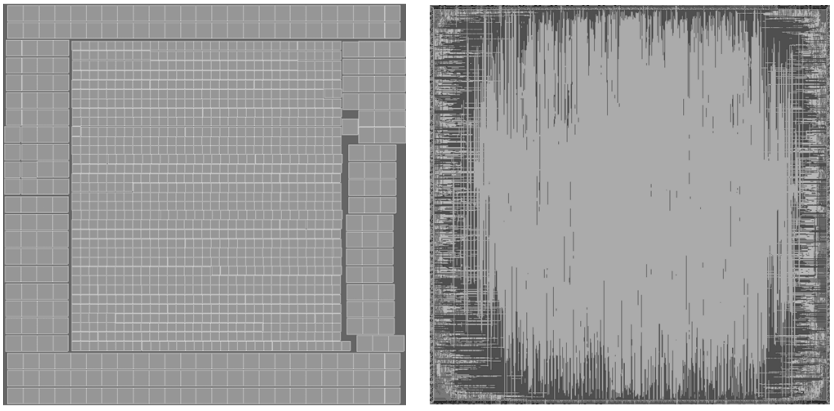


Fig. 10. The floor-plan and layout of the (8192, 7168) (4, 32) shift LDPC decoder

	Partly parallel Mansor (2006)	Ultra sparse Brack (2007)	Proposed
Code length	2048	9600	8192
Code rate	1/2	3/4	7/8
Edges	6144	26400	32768
Quantization	4b	6b	6b
Algorithm	Min-Sum	Min-Sum	Min-Sum
technology	180nm	65nm	180nm
frequency	125 MHz	500MHz	317 MHz
Iterations	10	10	15
Throughput	640Mb/s	1.45Gb/s	5.1 Gb/s
Area (mm2)	14.3	0.504	11.3/16.8
Area [scaled to 90nm]	3.57	1.008	2.82 /4.2
Hardware Efficiency	1778	14384	25500

Table 3. Comparisons with other decoder architectures

Table 3 shows the decoder implementation results compared with some other LDPC decoder architectures. The throughput achieved here is 5.1 Giga bits per second at a maximum iteration times of 15. One parameter "Hardware Efficiency" as (Throughput * Iterations / Area) is defined to evaluate the efficiency of each architecture. The area metrics are scaled to resemble 90nm CMOS results (65nm by a factor of 2 and 180nm by a factor of 1/4). From this table, we can see that the proposed design can achieve more than 70% improvement in hardware efficiency compared with an advanced existing design while the higher clock speed benefited from much more advanced CMOS technology is not considered for this comparison. Otherwise the improvement would be even more significant. So shift decoder architecture is very efficient for high-speed LDPC decoder implementation.

5. Apply the shift architecture to known LDPC codes

5.1 RS-based LDPC code

In Djurdjevic (2004), the authors constructed a kind of LDPC codes based on Reed-Solomon codes and called them RS-based LDPC codes. These codes have good minimum distances, girth properties, and perform very well with iterative decoding. One code constructed in this way was chosen as the forward error correction coding scheme in IEEE 802.3an 10GBase-T Ethernet standard (LAN/MAN CSMA/CD Access Method). RS-based LDPC codes are constructed in an algebraic way and by exploiting the shift-structured properties hidden in the parity check matrices, the shift decoder architectures presented above can be applied to this kind of code. In the following, we will explain the shift properties in these codes by going through the code generation procedure.

We start from the extended (q, 2, q -1) RS code C_b over GF(q) which has a length of q and two information symbols. By using the generation matrix

$$G = \begin{bmatrix} 1 & 1 & \cdots & 1 & 1 & 0 \\ 1 & \alpha^{q-2} & \cdots & \alpha^2 & \alpha & 1 \end{bmatrix}, \tag{3}$$

we can get the extended RS code with q^2 codewords in total. Let $\mathbf{v}$ be a nonzero codeword in C_b with weight q, for example $\mathbf{v}$ = (1, a^{q-2}, ⋯, a^2, a, 1). Then, the set $C_b^{(0)} = \{c\mathbf{v} : c \in GF(q)\}$ of q codewords forms the subcode of C_b. Set $C_b^{(0)}$ always contains the all zero codeword and q-1 weight q codewords, no matter which weight q codeword $\mathbf{v}$ is chosen.

Partition C_b into q additive cosets, $C_b^{(0)}$, $C_b^{(1)}$,…, $C_b^{(q-1)}$ based on the subcode $C_b^{(0)}$. Each coset $C_b^{(i)}$ is composed of q codewords, $W_0^{(i)}$, $W_1^{(i)}$, … , $W_{q-1}^{(i)}$ and each codeword has a length of q, as below:

$$C_b^{(i)} = \begin{bmatrix} W_0^{(i)} \\ W_1^{(i)} \\ \vdots \\ W_{q-1}^{(i)} \end{bmatrix} = \begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,q-1} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,q-1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{q-1,0} & w_{q-1,1} & \cdots & w_{q-1,q-1} \end{bmatrix}$$

$$= \begin{bmatrix} 1+\alpha^i & \alpha^{q-2}+\alpha^i & \alpha^{q-3}+\alpha^i & \cdots & \alpha+\alpha^i & 1 \\ \alpha+\alpha^i & 1+\alpha^i & \alpha^{q-2}+\alpha^i & \cdots & \alpha^2+\alpha^i & \alpha \\ \alpha^2+\alpha^i & \alpha+\alpha^i & 1+\alpha^i & \cdots & \alpha^3+\alpha^i & \alpha^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \alpha^{q-2}+\alpha^i & \alpha^{q-3}+\alpha^i & \alpha^{q-4}+\alpha^i & \cdots & 1+\alpha^i & \alpha^{q-2} \\ 0+\alpha^i & 0+\alpha^i & 0+\alpha^i & \cdots & 0+\alpha^i & 0 \end{bmatrix}. \tag{4}$$

Then the cosets are arranged together to get the $q^2 \times q$ matrix H_{rs}

$$H_{rs} = \begin{bmatrix} C_b^{(0)} \\ C_b^{(1)} \\ \vdots \\ C_b^{(q-1)} \end{bmatrix}. \tag{5}$$

Replace each symbol in H_{rs} with a location vector:

$$z(\alpha^i) = (z_{-\infty}, z_0, z_1, z_2, \cdots, z_{q-2}), \tag{6}$$

where the i-th componet z_i = 1 and all the other components equal zero. The exponential value can be denoted as the position of "1" in the location vector replacement. Finally, choose a $d_v \times d_c$ subarray from H_{rs} to get a (d_v, d_c) regular LDPC code. This can also be stated as that select d_v cosets randomly and then select d_c columns in them, finally replace each symbol with a location vector to get a sparse parity check matrix $\boldsymbol{H}$.

A coset example is given here to show the matrix properties clearly. It is generated in GF(8):

$$\begin{bmatrix} 4 & 6 & 7 & 3 & 1 & 5 & 0 & 1 \\ 0 & 4 & 6 & 7 & 3 & 1 & 5 & 2 \\ 5 & 0 & 4 & 6 & 7 & 3 & 1 & 3 \\ 1 & 5 & 0 & 4 & 6 & 7 & 3 & 4 \\ 3 & 1 & 5 & 0 & 4 & 6 & 7 & 5 \\ 7 & 3 & 1 & 5 & 0 & 4 & 6 & 6 \\ 6 & 7 & 3 & 1 & 5 & 0 & 4 & 7 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 0 \end{bmatrix}. \tag{7}$$

The examination of the matrix and equation (4) reveals the four following properties:
1) Each column is composed of the q elements of Galois Field GF(q).
2) qth row is a special row. It is composed with only one value a^i except the last one "0".
3) qth column is a special column. It is always "1, a, a^2, … a^{q-2}, 0".
4) Except the special row and special column, the remaining q-1 × q-1 matrix is a special type of circulant in which each row is the right cyclic-shift of the row above it while the first row is the right cyclic-shift of the last row.
Careful readers will find that property 4) is exactly the shift property which is introduced earlier in the chapter. Thus the shift decoder architecture can be applied to this kind of RS-based LDPC codes. Fig. 10 shows the location vector replacement in RS-based LDPC code construction and the mapping of the parity check matrix to decoder hardware. The chosen code sample is a (32, 24) (1, 4) RS-based LDPC code generated in GF(8).

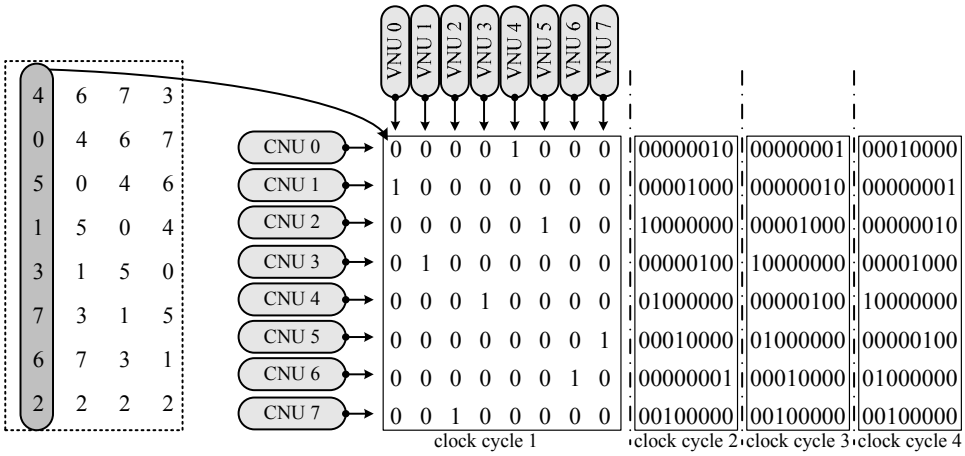


Fig. 11. The location vector replacement in RS-based LDPC code construction and the mapping of the parity check matrix to decoder hardware

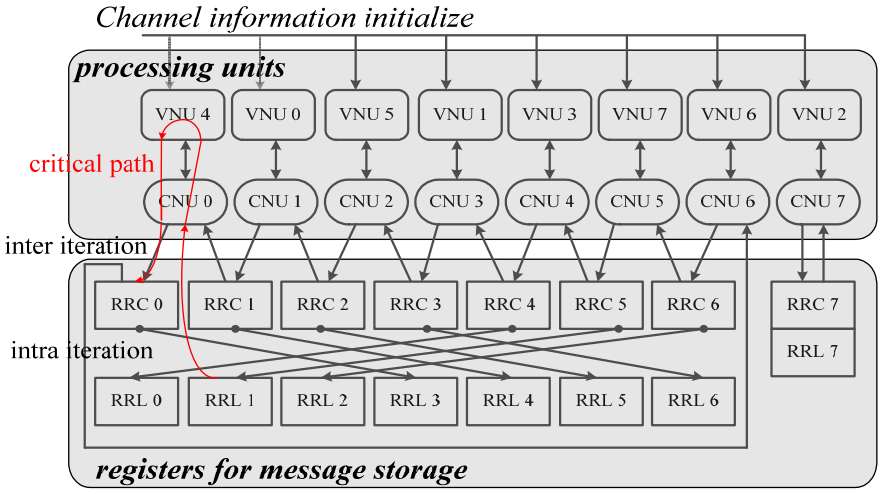


Fig. 12. Block diagram of the shift decoder design for the matrix defined in Fig.10

Fig. 11 shows the block diagram of decoder designed for matrix defined in Fig. 10. It can be decomposed into two parts: the processing units and the message storage registers. There is one row result registers (RR) corresponding to each CNU. One RR is composed of current iteration row result registers (RRC) and last iteration row result registers (RRL). Both contain minimum value, 2nd-minimum value, minimum value index and sign bit.

Since the offset value in qth row is fixed at "2", qth row's message always comes from VNU 2. CNU 7 is only in charge of the qth row's processing and RR 7 stores processing result of the qth row, which corresponds to the special row property 2).

With the shift decoder architecture, two decoder examples are designed. The target code is the (2048, 1723) (6, 32) LDPC code generated from (64, 32, 2) RS code. A 6 × 32 sub-array at the upper left corner of H_{rs} is selected.

Table 4 shows the synthesis results under 90nm CMOS technology of two design examples: one basic design and a four times parallel level design. Compared to the basic design, the four times parallel level design can achieve an approximately 3x decoding speed with 2x hardware consumption. Table 3 also compares the shift architecture decoders with state-of-the-art bit serial fully parallel LDPC decoder (Darabiha (2008)) and partially parallel LDPC decoder (Chen (2003)). Comparing the throughput to area ratio metric, it can be clearly seen that the shift architecture-based designs are very much hardware efficient.

Design	Module	Gate count × number	Total gate	parameters	Clock speed	throughput
basic 34 cycles	CNU	148 × 384	420K	6 bit quan 8 iterations	500MHz	3.8 Gbps
	VNU	1450 × 64				
	storage	2048 × 6 bits initial 64 × 384 bits message				
4parallel 10 cycles	CNU	515 × 384	820K	6 bit quan 8 iterations	400MHz	10 Gbps
	VNU	1450 × 256				
	storage	2048 × 6 bits initial 64 × 384 bits message				
Darabiha (2008)	(2048, 1723) (6,32) RS-based LDPC code fully parallel architecture		2230K	4 bit quan 8 iterations	250MHz	16 Gbps
Chen (2003)	8088 codeword length, 1/2 rate partially parallel architecture		742K	6 bit quan 25 iterations	212MHz	188 Mbps

Table 4. Synthesis results of (2048, 1723) RS-LDPC code decoder under shift architecture and comparisons with other designs

5.2 Quasi-Cyclic LDPC codes

Quasi-cyclic LDPC (QC-LDPC) codes are well known that they can achieve comparable performance with equivalent random-like LDPC codes. In addition, more importantly, they are well suited for hardware implementation because of the regularity in their parity check matrices. In particular the encoder of a QC-LDPC code can be easily built with shift registers while usually it is hardware complex to encode an LDPC code.

In general, a regular (c, t) quasi-cyclic LDPC code is defined as its parity check matrix H (M×N) consisting of a c×t array of submatrices. The dimension of each submatrix is q×q (q = M/c = N/t). Every submatrix H_{ij} is a circulant matrix of identity matrix I. Fig.12 shows an example of (24, 8) (2, 3) regular QC-LDPC code.

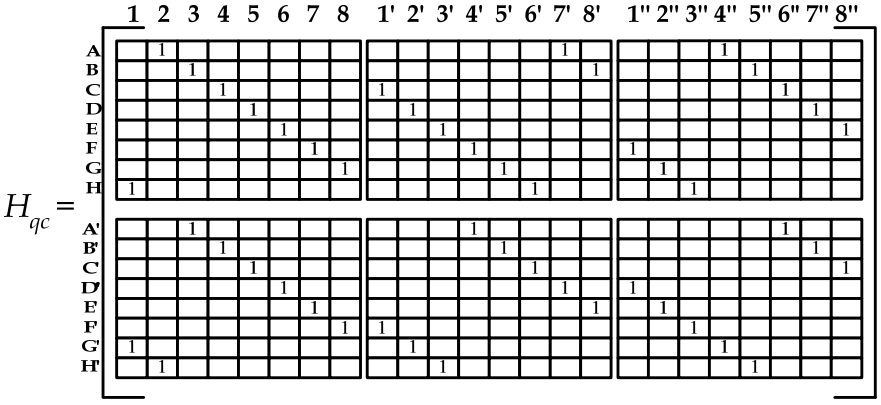


Fig. 13. The construction of QC-LDPC parity check matrix H_{qc}: an array of circulant submatrices

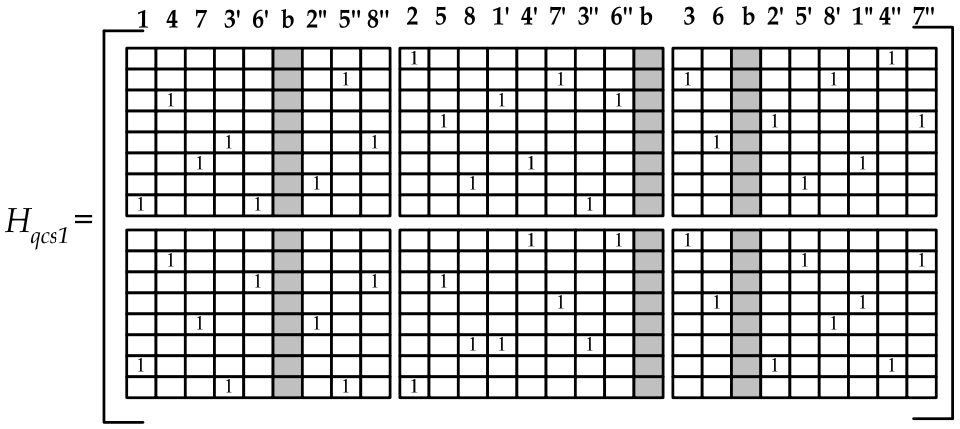


Fig. 14. The transformation from the sample QC-LDPC parity check matrix into a shift like LDPC code through column permutation

Fig. 13 illustrates the way to transform the QC-LDPC code parity check matrix into a shift kind matrix whose decoder can be implemented with the shift decoder architecture. Firstly the q columns are distributed to t block columns of H_{qcs1} in a round-robin fashion. Then the second q columns are permutated in the same way and so on until all columns are distributed into new matrix $H_{qcs1.}$ Careful readers will find that a new property is inducted in the new matrix that in some submatrices there are multiple "1"s in one row. This newly inducted property requires a special CNU design which processes multiple messages instead of one at each step. It will cause some additional logic delay in CNU. In Cui (2008a), the authors proposed an efficient matrix permutation optimization method to minimize the maximum row weight. Normally, it can be no bigger than 2. In addition, in Cui (2008a) the authors optimized the column layered decoding algorithm and applied it to shift decoder architecture. By this way, the iteration number to converge can be reduced and the decoding throughput can be further increased.

There are other matrix transformation methods to apply the extended shift decoder architecture on QC-LDPC codes. For example, the quasi cyclic matrix can be transformed into a row shift construction by row permutations as shown in Fig. 14. Firstly the q rows are distributed to 4 block rows of H_{qcs2} in a round-robin fashion (i.e., rows A-H of H_{qc} are distributed to row 1, 5, 9, 13, 2, 6, 10 and 14 of H_{qcs2}). Then the second q rows are permutated in the same way and so on until all rows are distributed into new matrix $H_{qcs2.}$ The quasi cyclic matrix is converted to a form as:

$$H_{qcrs} = \begin{bmatrix} A_1 & A_2 & A_3 & \cdots & A_t \\ A_1\alpha & A_2\alpha & A_3\alpha & \cdots & A_t\alpha \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_1\alpha^{q/m} & A_2\alpha^{q/m} & A_3\alpha^{q/m} & \cdots & A_t\alpha^{q/m} \end{bmatrix} \tag{8}$$

where a is a $q \times$ q permutation matrix representing a single right cyclic shift. m is an integer such that q can be divided by m.

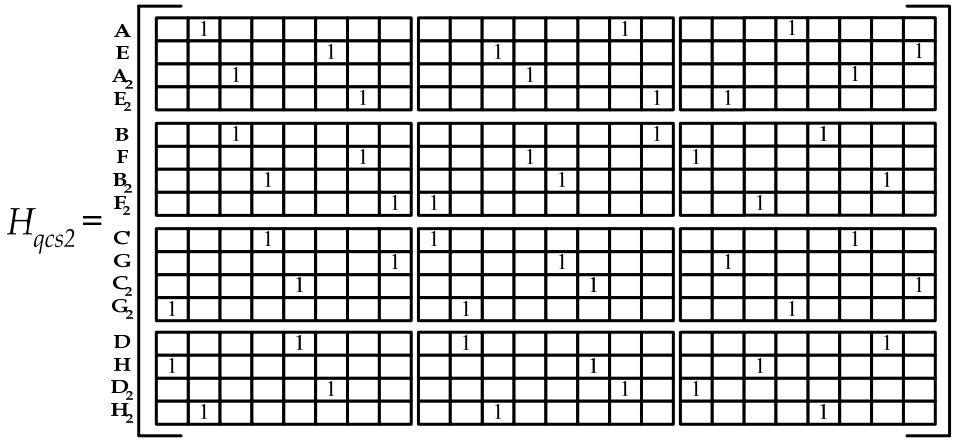


Fig. 15. The transformation from the sample QC-LDPC parity check matrix into a shift like LDPC code through row permutation

After the matrix conversion, the row shift property can be exploited to apply a row shift decoder architecture just like the presented column shift architecture. By this means, the row layered decoding algorithm should be applied. Fortunately the row layer algorithm can increase the convergence speed significantly (Mansour (2003b)). Interested readers can be referred to Cui (2008b) for more information.

5.3 Array LDPC code

Array LDPC code is a more structured QC-LDPC code which can achieve comparable performance to random codes. Efficient encoding, minimum distance properties, and performance of array-code based LDPC codes for binary as well as multilevel modulation are addressed in Fan (2000). The parity check matrix based on array codes can be represented as follows:

$$H_{array} = \begin{bmatrix} I & I & I & \cdots & I \\ I & \alpha & \alpha^2 & \cdots & \alpha^{t-1} \\ I & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(t-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I & \alpha^{c-1} & \alpha^{2(c-1)} & \cdots & \alpha^{(c-1)(t-1)} \end{bmatrix} \tag{9}$$

Where I is a $q \times q$ identity matrix with a prime number q. a is a permutation matrix with a single cyclic right shift or left shift. For example, a (2209, 2024) (4, 47) array LDPC code can be constructed with $q = 47$. We use this code in the decoder explanation.

For array LDPC codes, they have the similar shift property as shift-LDPC codes. Some modifications in the *CNU communication network* can be enough for the decoder implementation. Fig. 15 shows the modified CNU connections. They are grouped into four

CNU groups. Each group has 47 members. Group 1 contains from CNU-1 to CNU-47, in charge of the row operation of row 1~47. Group 2 contains from CNU-48 to CNU-94, in charge of the row operation of row 48~94... There is no communication between groups. The difference between array LDPC codes and shift LDPC codes is that the shift values are changing from 0 to 3 instead of keeping at 1. In addition, for this special kind of LDPC codes, the *CNU communication network (intra iteration)* is exactly the same with *CNU communication network (inter iteration)*, thus these two networks can be incorporated into one simple network and the routing complexity can be further simplified. As a result, except the CNUs in Group 1, every 47 CNUs in a group form a chain and each CNU only communicates with its two neighbors. For example, in CNU group 3, CNU-*i* only gets data from CNU-(*i*-2) and delivers date to CNU-(*i*+2).
A sample decoder is designed based on an ALTERA FPGA EP2C35. 50MHz clock speed and 120Mbps throughput (20 iterations) are achieved with 23k logic elements and 26k memory bits. Interested readers are referred to Sha (2006) for more information.

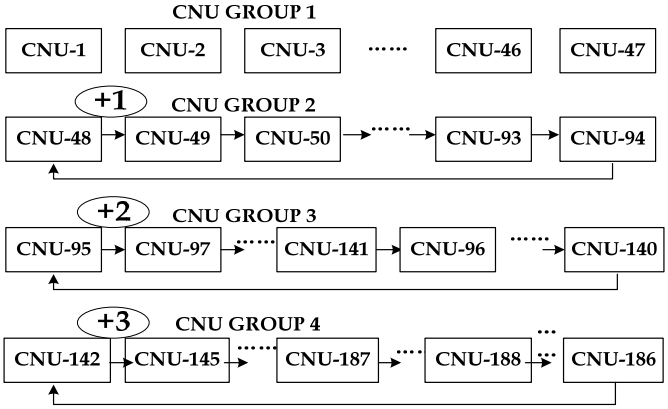


Fig. 16. The CNU communication network designed for (2209, 2024) (4, 47) array LDPC code

6. Concluding Remarks

Since MacKay's (1997) rediscovery of LDPC codes, the LDPC decoder design has experienced considerable development and enhancement over the last ten years. Various optimization techniques on both decoding algorithms and decoder designs have been developed. This chapter provided a brief overview of the problems in existing LDPC decoder designs and presented a novel shift-structured decoder design approach to tackle these problems. The codes suited for this kind of decoder architecture are called shift-LDPC codes. Several decoder samples are discussed to illustrate the effectiveness of the proposed architecture. In addition, it was shown in the chapter that some popular classes of LDPC codes such as RS-based LDPC codes and QC-LDPC codes can be implemented with the shift decoder architecture through simple matrix permutations or architecture extensions. As a conclusion, the presented shift decoder architecture will be a competent candidate in future high-speed communication system designs.

7. References

A. Darabiha, A. C. Carusone, & F. R. Kschischang. (2008). Power Reduction Techniques for LDPC Decoders, *IEEE Journal of Solid-State Circuits*, vol. 43, no. 8, pp. 1835-1845

A. J. Blanksby & C .J .Howland. (2002). A 690-mW 1-Gbps 1024-b, rate-1/2 Low-Density Parity-Check code decoder, *IEEE J. Solid-State Circuits*, vol. 37, no. 3, pp. 404-412

D. J. C. MacKay & R. M.Neal. (1996). Near Shannon limit performance of low density parity check codes, *Electron. Lett*., vol. 32, pp. 1645-1646

E. Boutillon, J. Castura, & F. R. Kschischang, (2000). Decoder-First Code Design, *Proceedings of the 2nd International Symposium on Turbo Codes and Related Topics*, Brest, France, pp. 459-462

F. Guilloud, E. Boutillon & J.L. Danger. (2003). λ-Min Decoding Algorithm of Regular and Irregular LDPC Codes, *Proc. 3nd International Symposium on Turbo Codes and Related Topics*, pp. 451-454

E. Liao, E. Yeo & B. Nikolic. (2004). Low-density parity-check code constructions for hardware implementation, *Proc. IEEE Int. Conf. on Commun.* vol. 5, pp. 2573-2577

E. Yeo, B. Nikolic & V. Anantharam. (2003) Iterative decoder architectures, *IEEE Commun. Mag.*, vol. 41, pp. 132-140

F. R. Kschischang, B. J. Frey & H. A. Loeliger. (2001). Factor graphs and the sum-product algorithm, *IEEE Trans. Inf. Theory*, vol. 47, pp. 498-519

G. Liva, S. Song, L. Lan, Y. Zhang, S. Lin, & W. E. Ryan. (2006). Design of LDPC Codes: A Survey and New Results. *J. Comm. Software and Systems,* vol. 2, pp. 191

I. Djurdjevic, J. Xu, K. Abdel-Ghaffar, & S. Lin. (2004). Construction of low-density parity-check codes based on Reed-Solomon codes with two information symbols, *IEEE Commu. Lett.*, vol. 8, no. 7, pp. 317-319

J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, & X. Hu. (2005). Reduced-complexity decoding of LDPC codes, *IEEE Trans. Commun.*, vol. 53, pp. 1288-1299

J. L. Fan. (2000). Array codes as low-density parity-check codes, *Proc. 2nd Int. Symp. Turbo Codes and Related Topics Brest*, France, pp. 543

J. Sha, M. Gao, Z. Zhang, L. Li, Z. Wang. (2006). An FPGA Implementation of array LDPC decoder, *IEEE Asia Pacific Conference on Circuits and Systems*, pp. 1675-1678

J. Sha, Z. Wang, M. Gao, & Li. (2009). Multi-Gb/s LDPC Code Design and Implementation, *IEEE Trans. on VLSI Systems*, vol. 17, no. 2, pp. 262-268

LAN/MAN CSMA/CD Access Method, *IEEE 802.3 Standard* Online available: http://standards.ieee.org/getieee802/802.3.html

J. Zhao, F. Zarkeshvari, & A. H. Banihashemi. (2005). On implementation of min-sum algorithm and its modifications for decoding Low-Density Parity-Check (LDPC) codes, *IEEE Trans. Commun.*, vol. 53, no. 4, pp. 549-554

L. Chen, J. Xu, I. Djurdjevic & S. Lin. (2004a) Near-Shannon limit quasi-cyclic low-density parity-check codes, *IEEE Trans. Commun*, vol. 52, pp. 1038

M. Cocco, J. Dielissen, M. Heijligers, A. Hekstra, & J. Huisken. (2004). A scalable architecture for LDPC decoding, *Proc. Design, Automation and Test in Europe*, vol. 3, pp. 88-93

M. M. Mansour & N. R. Shanbhag. (2002). Low power VLSI decoder architecture for LDPC codes, *Proc. IEEE Int. Symp. on Low Power Electron. Design*, pp. 284-289

M. M. Mansour & N. R. Shanbhag, (2003a). Architecture-Aware Low-Density Parity-Check Codes, *Proc. IEEE ISCAS*, pp. 57-60

M. M. Mansour & N. R. Shanbhag. (2003b). High throughput LDPC decoders, *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, pp. 976-996

M.M. Mansour & N. R. Shanbhag. (2006). A 640-Mb/s 2048-bit programmable LDPC decoder chip, *IEEE J. Solid-State Circuits*, vol. 41, no. 3, pp. 684- 698

M. P. C. Fossorier. (2004). Quasi-cyclic low-density parity-check codes from circulant permutation matrices, *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1788–1793

R. G. Gallager. (1962). Low-density parity-check codes, *IRE Transactions on Information Theory*, vol. IT-8, pp. 21-28

R. M. Tanner. (1981). A recursive approach to low complexity codes, *IEEE Trans. Inf. Theory*, vol. IT-27, pp. 533-547

S. Hemati, A. Banihashemi, & C. Plett. (2006). A 0.18 μm analog min-sum iterative decoder for a (32,8) low-density parity-check (LDPC) code, *IEEE J. Solid-State Circuits*, vol. 41, pp. 2531–2540

S. Seo, T. Mudge, Y. Zhu & C. Chakrabarti. (2007). Design and analysis of LDPC decoders for software defined radio, *Proc. IEEE Workshop on Signal Processing Systems*, Shanghai, China, pp.210–215

S. Sharifi Tehrani, S. Mannor & W. J. Gross. (2008). Fully Parallel Stochastic LDPC Decoders *IEEE Trans. Signal Processing*, Vol. 56, no. 11, pp. 5692 - 5703

S. Y. Chung, G. D. Forney, T. J. Richardson & R. Urbanke. (2001). On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit, *IEEE Commun. Lett.*, vol. 5, pp. 58-60

T. Brack, M. Alles, T. Lehnigk-Emden, F. Kienle, N. Wehn, & L. Fanucci. (2007). Low Complexity LDPC Code Decoders for Next Generation Standards, *Proc. Design, Automation and Test in Europe,* pp. 1-6

T. Zhang & K. Parhi. (2002). A 54Mbps (3,6)-regular FPGA LDPC decoder, *Proc. IEEE Sips*', pp. 127-132

T. Zhang & K. K. Parhi. (2004). Joint (3,k)-regular LDPC code and decoder/encoder design, *IEEE Trans. Signal Process.*, vol. 52, no. 4, pp.1065–1079

Y. Chen & D. Hocevar. (2003). A FPGA and ASIC implementation of rate 1/2, 8088-b irregular low density parity check decoder, *Proc. IEEE GLOBECOM*, San Francisco, CA, pp. 113–117

Y. Chen & K. K. Parhi. (2004b). Overlapped message passing for quasi-cyclic low density parity check codes, *IEEE Trans. Circuits Syst. I,* vol. 51, pp. 1106-1113

Z.–W. Li, L. Chen, L.-Q. Zeng, S. Lin & W.H. Fong. (2006). Efficient encoding of quasi-cyclic low-density parity-check codes, *IEEE Transactions on Communications*, Vol. 54 , no. 1, pp. 71 – 81

Z. Cui, Z. Wang, X. Zhang & Q. Jia. (2008a). Efficient decoder design for high-throughput LDPC decoding, *APCCAS*, pp. 1640-1643

Z. Cui, Z. Wang & Y. Liu. (2008b). High-throughput layered LDPC decoding architecture, *IEEE Trans. VLSI Systems*, vol. 17, no. 4, pp. 582-587

Z. Wang & Z. Cui. (2007). Low-complexity high-speed decoder design for quasi-cyclic LDPC codes, *IEEE Trans. on VLSI Systems*, vol. 15, no. 1, pp. 104-114

10

A Methodology for Parabolic Synthesis

Erik Hertz and Peter Nilsson
Department of Electrical and Information Technology, Lund University
Sweden

1. Introduction

In relatively recent research of the history of science interpolation theory, in particular of mathematical astronomy, revealed rudimentary solutions of interpolation problems date back to early antiquity (Meijering, 2002). Examples of interpolation techniques originally conceived by ancient Babylonian as well as early-medieval Chinese, Indian, and Arabic astronomers and mathematicians can be linked to the classical interpolation techniques developed in Western countries from the 17th until the 19th century. The available historical material has not yet given a reason to suspect that the earliest known contributors to classical interpolation theory were influenced in any way by mentioned ancient and medieval Eastern works. For the classical interpolation theory it is justified to say that there is no single person who did so much for this field as Newton. Therefore, Newton deserves the credit for having put classical interpolation theory on a foundation. In the course of the 18th and 19th century Newton's theories were further studied by many others, including Stirling, Gauss, Waring, Euler, Lagrange, Bessel, Laplace, and Everett. Whereas the developments until the end of 19th century had been impressive, the developments in the past century have been explosive. Another important development from the late 1800s is the rise of approximation theory. In 1885, Weierstrass justified the use of approximations by establishing the so-called approximation theorem, which states that every continuous function on a closed interval can be approximated uniformly to any prescribed accuracy by a polynomial. In the 20th century two major extensions of classical interpolation theory is introduced: firstly the concept of the cardinal function, mainly due to E. T. Whittaker, but also studied before him by Borel and others, and eventually leading to the sampling theorem for band limited functions as found in the works of J. M. Whittaker, Kotel'nikov, Shannon, and several others, and secondly the concept of oscillatory interpolation, researched by many and eventually resulting in Schoenberg's theory of mathematical splines.

The parabolic synthesis methodology
Unary functions, such as trigonometric functions, logarithms as well as square root and division functions are extensively used in computer graphics, digital signal processing, communication systems, robotics, astrophysics, fluid physics, etc. For these high-speed applications, software solutions are in many cases not sufficient and a hardware implementation is therefore needed. Implementing a numerical function $f(x)$, by a single

look-up table (Tang, 1991) is simple and fast which is strait forward for low-precision computations of *f(x)*, i.e., when *x* only has a few bits. However, when performing high-precision computations a single look-up table implementation is impractical due to the huge table size and the long execution time.
Approximations only using polynomials have the advantage of being ROM-less, but they can impose large computational complexities and delays (Muller, 2006). By introducing table based methods to the polynomials methods the computational complexity can be reduced and the delays can also be decreased to some extent (Muller, 2006).
The CORDIC (COordinate Rotation DIgital Computer) algorithm (Volder, 1959) (Andrata, 1998) has been used for these applications since it is faster than a software approach. CORDIC is an iterative method and therefore slow which makes the method insufficient for this kind of applications.
The proposed methodology of parabolic synthesis (Hertz & Nilsson, 2008) develops functions that perform an approximation of original functions in hardware. The architecture of the processing part of the methodology is using parallelism to reduce the execution time. For the development of approximations of functions a parabolic synthesis methodology has been applied. Only low complexity operations that are simple to implement in hardware are used

2. Methodology

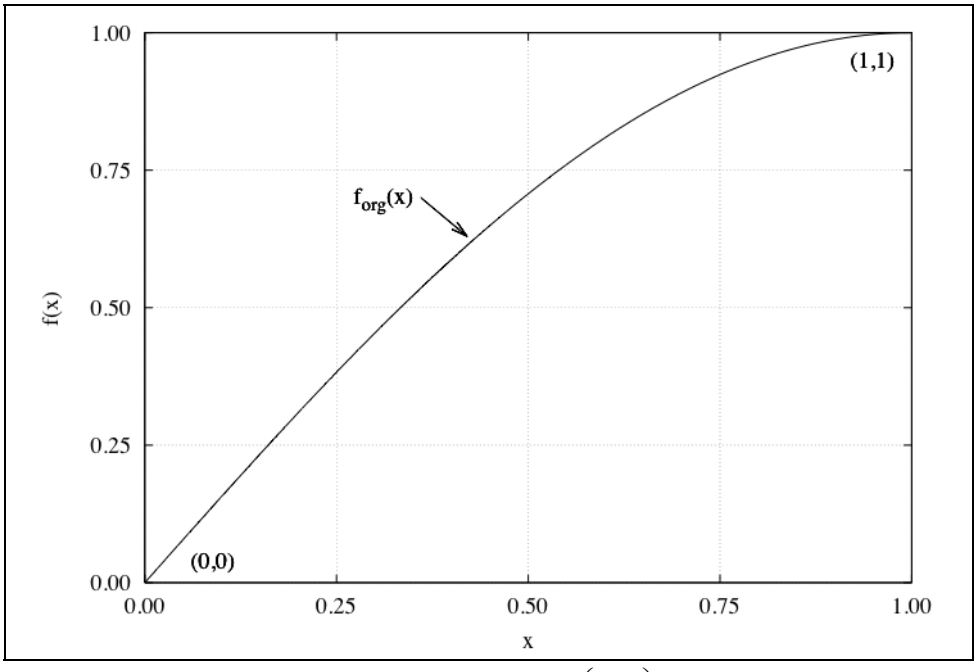


Fig. 1. Example of normalized function, in this case $\sin\left(\frac{\pi \cdot x}{2}\right)$.

The methodology is developed for implementing approximations of unary functions in hardware. The approximation part is of course the important part of this work but there are sometimes two other steps that are necessary, a preprocessing normalization and postprocessing transformation as described by (P.T.P. Tang, 1991) (Muller, 2006). The computation is therefore divided into three steps, normalizing, approximation and transforming.

2.1 Normalizing

The purpose with the normalization is to facilitate the hardware implementation by limiting the numerical range.

The normalization has to satisfy that the values are in the interval $0 \leq x < 1$ on the x-axis and $0 \leq y < 1$ on the y-axis. The coordinates of the starting point shall be (0,0). Furthermore, the ending point shall have coordinates smaller than (1,1) and the function must be strictly concave or strictly convex through the interval. An example of such a function, called an original function $f_{org}(x)$, is shown in Fig. 1.

2.2 Developing the Hardware Architecture

When developing a hardware architecture that approximates an original function, only low complexity operations are used. Operations such as shifts, additions and multiplications are efficient to implement in hardware and therefore searched for. The downscaling of the semiconductor technologies and the development of efficient multiplier architectures has made the multiplication operation efficient in both size and execution, time when implemented in hardware. The multiplier is therefore commonly used in this methodology when developing the hardware.

As in Fourier analysis (Fourier, 1822) the proposed methodology is based on decomposition of basic functions. The proposed methodology is not, as in Fourier analysis, a decomposition method in terms of sinusoidal functions but in second order parabolic functions. Second order parabolic functions are used since they can be implemented using low complexity operations. The proposed methodology also differs from the Fourier synthesis process since the proposed methodology is using multiplications in the recombination process and not additions as in the Fourier case.

The proposed methodology is founded on terms of second ordered parabolic functions called sub-functions $s_n(x)$, that when recombined, as shown in (1), obtains to the original function $f_{org}(x)$. When developing the approximate function, the accuracy depends on the number of sub-functions used.

$$f_{org}(x) = s_1(x) \cdot s_2(x) \cdot \ldots \cdot s_\infty(x) \tag{1}$$

The procedure when developing sub-functions is to divide the original function $f_{org}(x)$, with the first sub-function $s_1(x)$. This division generates the first function $f_1(x)$, as shown in (2).

$$f_1(x) = \frac{f_{org}(x)}{s_1(x)} \tag{2}$$

The first sub-function $s_1(x)$, will be chosen to be feasible for hardware, according to the methodology described in (4). In the same manner the following functions $f_n(x)$, are generated, as shown in (3).

$$f_{n+1}(x) = \frac{f_n(x)}{s_{n+1}(x)} \tag{3}$$

The purpose with the normalization is to facilitate the hardware implementation by limiting the numerical range.

2.3 Methodology for developing sub-functions

The methodology for developing sub-functions is founded on decomposition of the original function $f_{org}(x)$, in terms of second order parabolic functions for the interval $0 \leq x < 1.0$ and the sub intervals within the interval. The second order parabolic function is chosen as decomposition function since the structure is reasonable simple to implement in hardware i.e. only low complexity operations such as additions and multiplications are used.

First sub-function

The first sub-function $s_1(x)$, is developed by dividing the original function $f_{org}(x)$, with x as an approximation.

As shown in Fig. 2 there are two possible results after dividing the original function with x, one where $f(x)>1$ and one where $f(x)<1$.

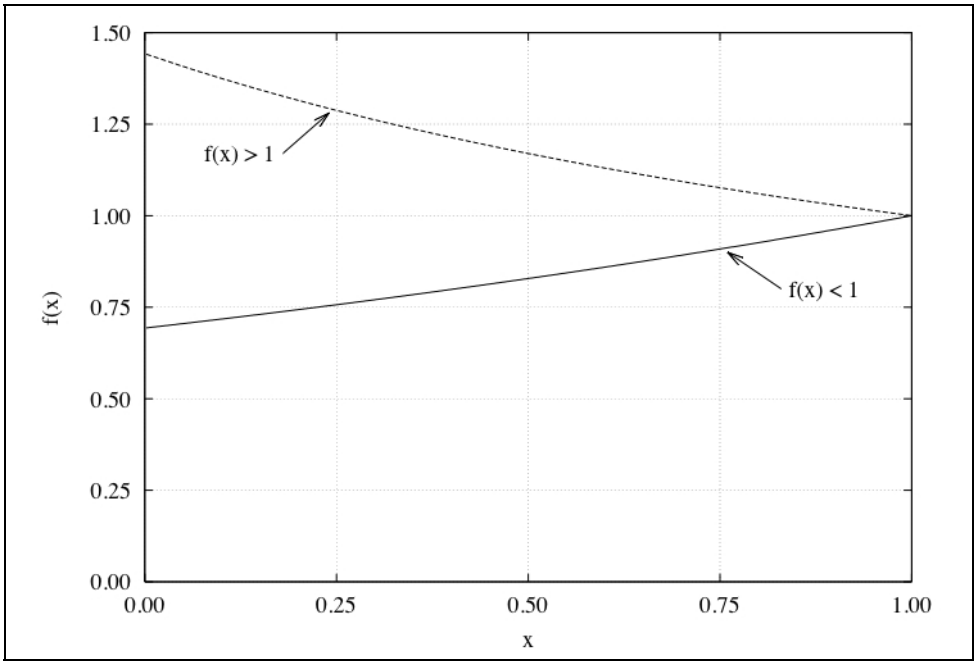


Fig. 2. Two possible results after dividing an original function with x.

The first sub-function $s_1(x)$, is according to (4). To approximate these functions $1+(c_1 \cdot (1-x))$ is used. The first sub-function $s_1(x)$, is given by a multiplication of x and $1+(c_1 \cdot (1-x))$ which results is a second order parabolic function according to (4).

$$s_1(x) = x \cdot (1 + (c_1 \cdot (1 - x))) = x + (c_1 \cdot (x - x^2)) \tag{4}$$

In (4) the coefficient c_1 is determined as the limit from the division of the original function with x and subtracted with 1, according to (5).

$$c_1 = \lim_{x \to 0} \frac{f_{org}(x)}{x} - 1 \tag{5}$$

Second sub-function

The first function $f_1(x)$, is calculated according to (2) and the result of this operation is a function which appearance is similar to a parabolic function, as shown in Fig. 3.

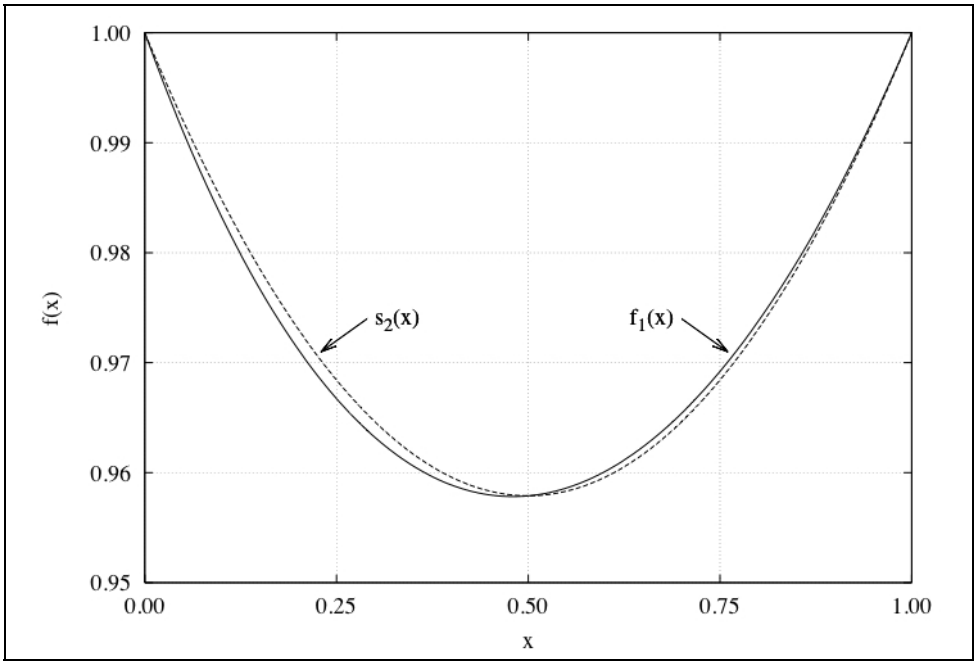


Fig. 3. Example of the first function $f_1(x)$ compared with sub-function $s_2(x)$.

The second sub-function $s_2(x)$, is chosen according to the methodology as a second order parabolic function, see (6).

$$s_2(x) = 1 + (c_2 \cdot (x - x^2)) \tag{6}$$

In (6) the coefficient c_2, is chosen to satisfy that the quotient between the function $f_1(x)$, and the second sub-function $s_2(x)$, is equal to 1 when x is equal to 0.5, see (7).

$$c_2 = 4 \cdot \left(f_1\left(\frac{1}{2}\right) - 1 \right) \tag{7}$$

Thereby the second function $f_2(x)$, will get a shape of a lying **S**, as shown in Fig. 4.

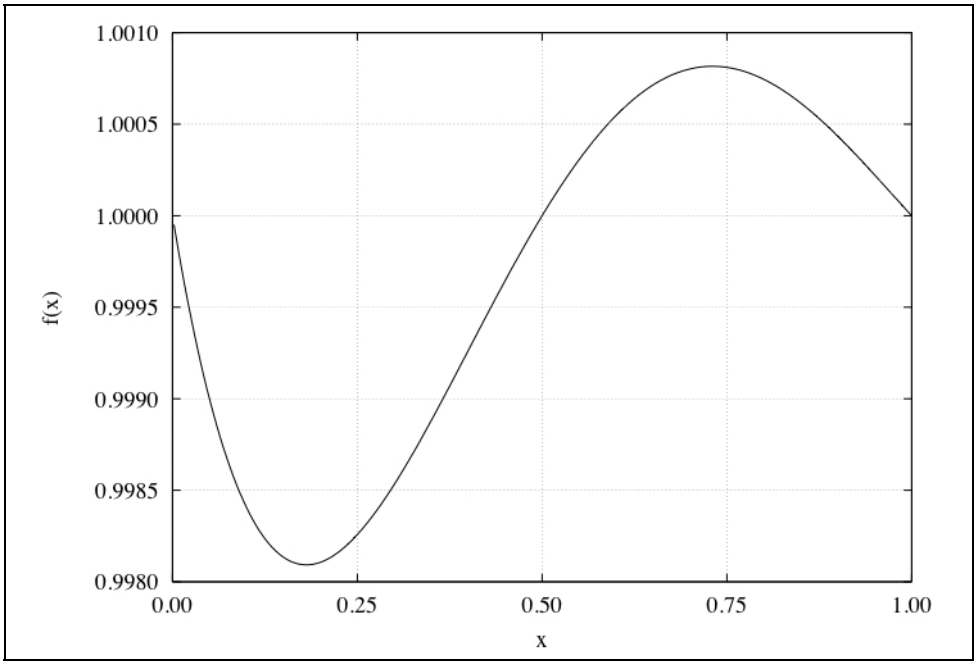


Fig. 4. Example of the second function $f_2(x)$, shaped like a lying **S**.

When developing the third sub-function $s_3(x)$, the function is to be split into two parabolic functions where the first function is restricted by function $f_2(x)$ to be in the interval $0 \leq x < 0.5$ and the second function is thus restricted to the interval $0.5 \leq x < 1.0$. By splitting the function we get strictly convex and concave functions in each interval. The intervals can be chosen differently but that will lead to a more complex hardware, as shown in section 3.

Sub-functions when n > 2

For functions $f_n(x)$ when $n > 2$, the function is characterized by the form of one or more **S** shaped functions. When developing the higher order sub-functions, each **S** shaped function is divided into two parabolic functions. For each sub interval, a parabolic sub-function is developed as an approximation of the function $f_n(x)$ in the sub interval. To show which sub

interval the partial functions is valid for, the subscript index is increased with the index m, which gives the following appearance of the partial function $f_{n,m}(x)$.

$$f_n(x) = \begin{cases} f_{n,0}(x), & 0 \le x < \dfrac{1}{2^{n-1}} \\ f_{n,1}(x), & \dfrac{1}{2^{n-1}} \le x < \dfrac{2}{2^{n-1}} \\ \cdots & \\ f_{n,2^{n-1}-1}(x), & \dfrac{2^{n-1}-1}{2^{n-1}} \le x < 1 \end{cases} \tag{8}$$

In equation (8) it is shown how the function $f_n(x)$, is divided into partial functions $f_{n,m}(x)$, when $n > 2$.

As shown in (8), the number of partial functions is doubled for each order of $n > 1$ i.e. the number of partial functions is 2^{n-1}. From these partial functions, the corresponding sub-functions are developed. Analogous to the function $f_n(x)$, also the sub-function $s_{n+1}(x)$, will have partial sub-functions $s_{n+1,m}(x)$. In equation (9) it is shown how the sub-function $s_n(x)$, is divided into partial functions when $n > 2$.

$$s_n(x) = \begin{cases} s_{n,0}(x), & 0 \le x < \dfrac{1}{2^{n-2}} \\ s_{n,1}(x), & \dfrac{1}{2^{n-2}} \le x < \dfrac{2}{2^{n-2}} \\ \cdots & \\ s_{n,2^{n-2}-1}(x), & \dfrac{2^{n-2}-1}{2^{n-2}} \le x < 1 \end{cases} \tag{9}$$

Note that in (9), the partial functions to the sub-functions; x has been changed to x_n. The change to x_n is normalization to the corresponding interval, which simplifies the hardware implementation of the parabolic function. To simplify the normalization of the interval of x_n it is selected as an exponentiation by 2 of x where the integer part is removed. The normalization of x is therefore done by multiplying x with 2^{n-2}, which in hardware is $n-2$ left shifts and the integer part is dropped, which gives x_n as a fractional part ($frac(\ )$) of x, as shown in (10).

$$x_n = frac\left(2^{n-2} \cdot x\right) \tag{10}$$

As in the second sub-function $s_2(x)$, the second order parabolic function is used as an approximation of the interval of the function $f_{n-1}(x)$, as shown in (11).

$$s_{n,m}(x_n) = 1 + \left(c_{n,m} \cdot \left(x_n - x_n^2\right)\right) \tag{11}$$

Where the coefficients $c_{n,m}$ is computed according to (12).

$$c_{n,m} = 4 \cdot \left(f_{n-1,m} \left(\frac{2 \cdot (m+1) - 1}{2^{n-1}} \right) - 1 \right) \tag{12}$$

After the approximation part the result is transformed into its desired form.

3. Hardware Implementation

For the hardware implementation two's complement representation (Parhami, 2000) is used. The implementation is divided into three hardware parts, preprocessing, processing, and postprocessing as shown in Fig. 5, which was introduced by (P.T.P. Tang, 1991), (Muller, 2006).

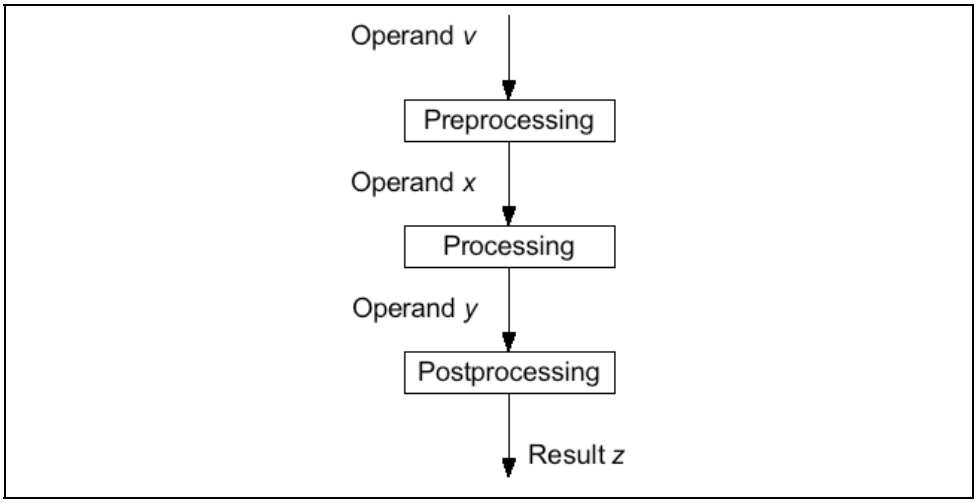


Fig. 5. The hardware architecture of the methodology.

3.1 Preprocessing

In this part the incoming operand v is normalized to prepare the input to the processing part, according to section 2.1.

If the approximation is implemented as a block in a system the preprocessing part can be taken into consideration in the previous blocks, which implies that the preprocessing part can be excluded.

3.2 Processing

In the processing part the approximation of the original function is directly computed in either iterative or parallel hardware architecture.

The three equations (4), (6) and (11) has the same structure which gives that the approximation can be implemented as an iterative architecture as shown in Fig. 6.

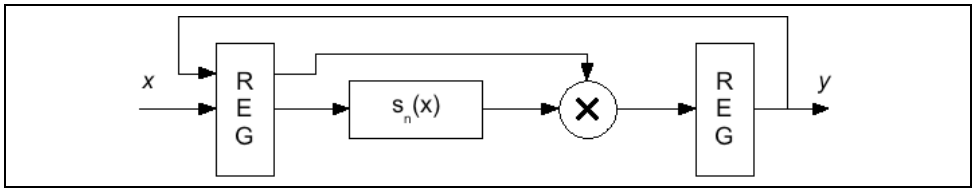


Fig. 6. The principle of an iterative hardware architecture.

The benefit of the iterative architecture is the small chip area whereas the disadvantage is longer computation time.
The advantages with parallel hardware architecture are that it gives a short critical path and fast computation to the prize of a larger chip area. The principle of the parallel hardware architecture for four sub-functions is shown in Fig. 7.

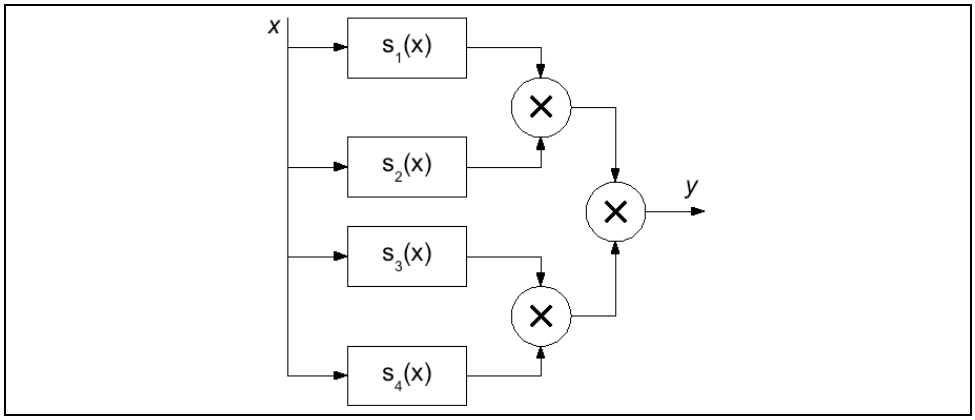


Fig. 7. The architecture principle for four sub-functions.

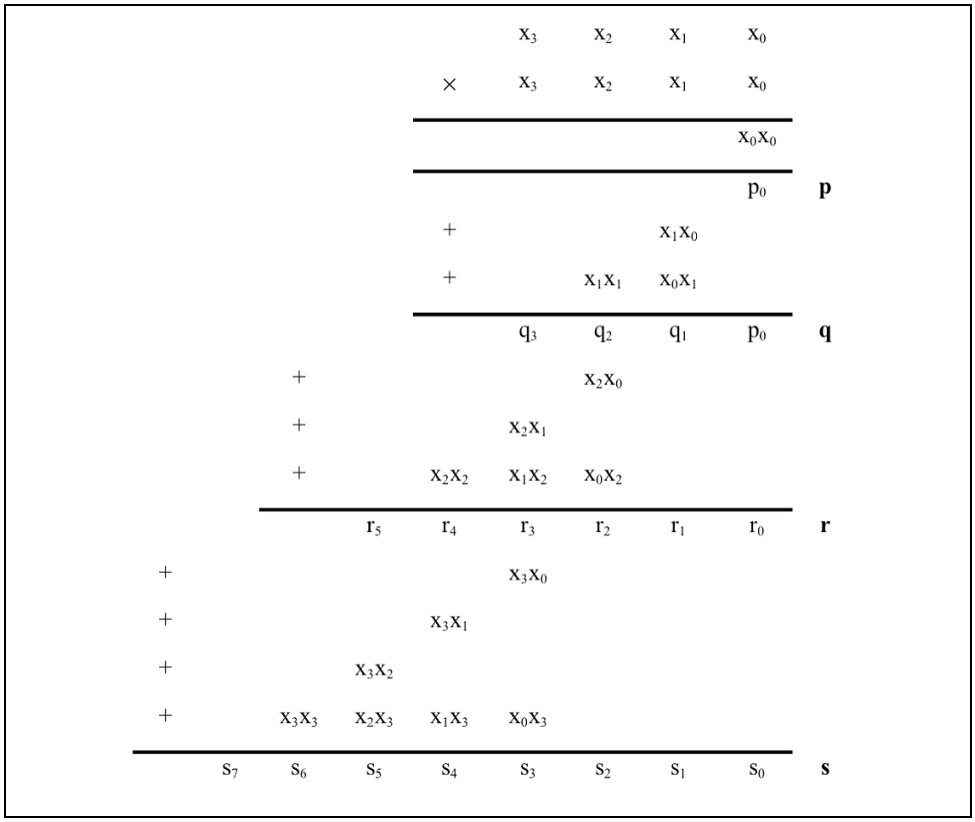


Fig. 8. Squaring algorithm for the partial products x_n^2.

To increase the throughput even more, pipeline stages can be implemented in the parallel hardware architecture.

In the sub-functions (4), (6) and (11) x^2 and x_n^2 are reoccurring operations. Since the square operation x_n^2, in the parallel hardware architecture is a partial result of x^2 a unique squarer has been developed. In Fig. 8 the algorithm that performs the squaring and delivers partial product of x_n^2 is described.

The squaring algorithm for the partial products x_n^2 can be simplified as shown in Fig. 9.

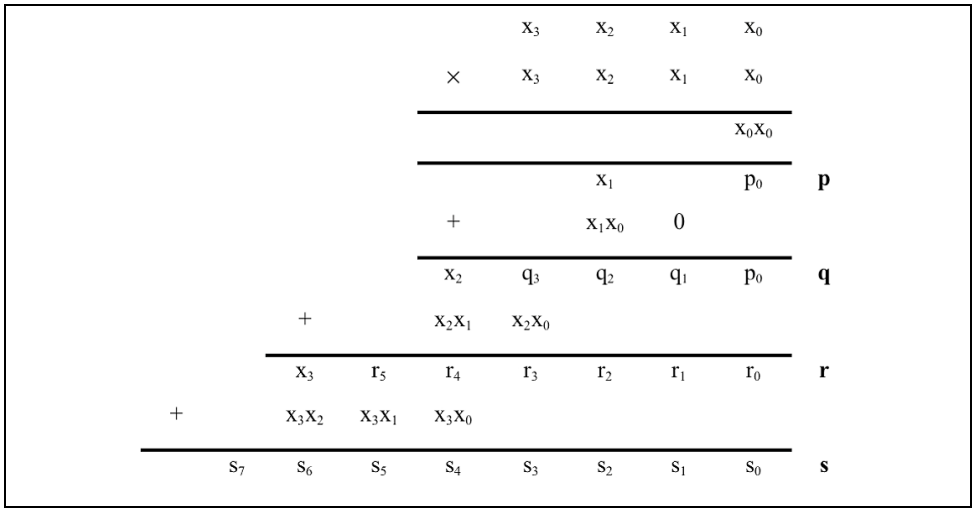


Fig. 9. Simplified squaring algorithm for the partial products x_n^2.

In Fig. 8 and Fig. 9, the squaring algorithm that performs the partial products x_n^2, shown. The first partial product p, is the squaring of the least significant bit in x. The second partial product q, is the squaring of the two least significant bits in x. The partial product r, is the result of the squaring of the three least significant bits in x and s is the result of the squaring of x. The squaring operation is performed with unsigned numbers. When analyzing the squarer in Fig. 8 and Fig. 9, it was found that the resemblance to a bit-serial squarer (Ienne & Viredaz, 1994) (Pekmestz et al., 2001) is large. By introducing registers in the design of the bit-serial squarer the partial results of x_n^2 is easily extracted. The squaring algorithm can thus be simplified to one addition only when computing each partial product.

From (4), (6) and (11) it is found that only the coefficients values differentiate when implementing different unary functions. This implies that different unary functions can be realized in the same hardware in the processing part, just by using different sets of coefficients.

Since the methodology is calculating an approximation of the original function the error to the desired precision can be both positive and negative. Especially, if the value of the approximation is less than the desired precision, the word length can have to be increased compared with the word length needed to accomplish the desired precision. If the order of the last used sub-function is $n > 1$, an improvement of the precision can be done by optimizing one or more coefficients c_2 in (7) or $c_{n,m}$ in (12). The optimization of the coefficients will minimize the error in the last used sub-function and thereby it can reduce the word length needed to accomplish the desired accuracy. Computer simulations perform such coefficient optimization numerically.

3.3 Postprocessing

The postprocessing part transforms the value to the output result z. If the approximation is implemented as a block in a system the postprocessing part can be taken into consideration in the following blocks, which implies that the postprocessing part can be excluded.

4. Implementation of the sine function

An implementation of the function *sin(v)*, using the proposed methodology (Hertz & Nilsson, 2009) is described in this section as an example.

4.1 Preprocessing

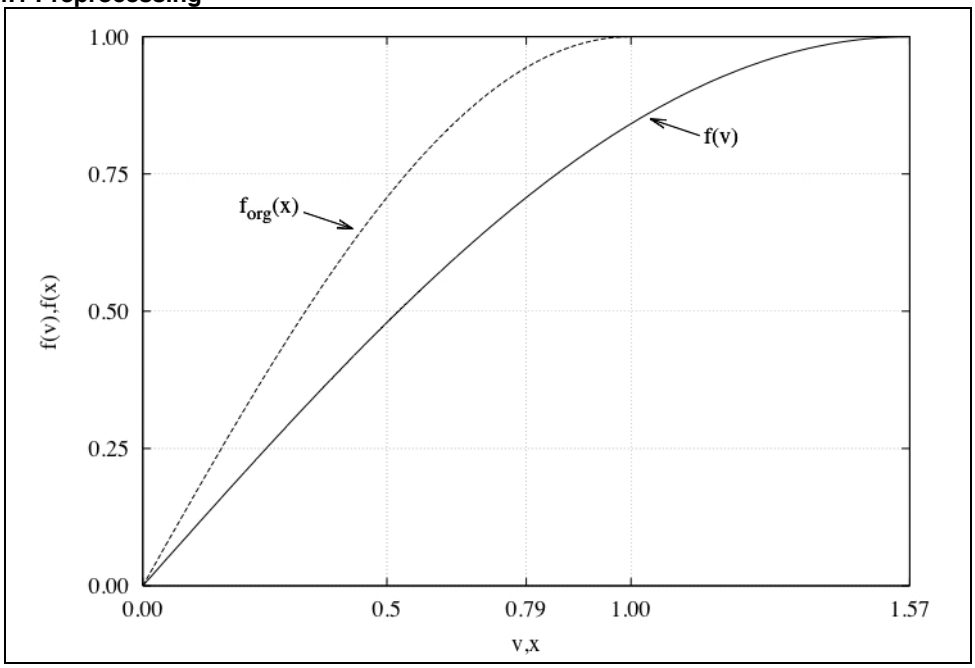


Fig. 10. The function *f(v)* before normalization and the original function $f_{org}(x)$.

To satisfy that the values of the incoming operand x is in the interval $0 \leq x < 1$ a $\pi/2$ is multiplied with the operand as shown in (13).

$$v = \frac{\pi}{2} \cdot x \tag{13}$$

To normalize the *f(v)=sin(v)* function v is substituted with x which gives the original function $f_{org}(x)$ (14).

$$f_{org}(x) = \sin\left(\frac{\pi}{2} \cdot x\right) \tag{14}$$

In Fig. 10 the *f(v)* function is shown together with the original function $f_{org}(x)$.

4.2 Processing

For the processing part, sub-functions are developed according to the proposed methodology. For the first sub-function $s_1(x)$, the coefficient c_1 is defined according to (5). The determined value of the coefficient is shown in (15).

$$s_1(x) = x + \left(\left(\frac{\pi}{2} - 1 \right) \cdot \left(x - x^2 \right) \right) \tag{15}$$

The first function $f_1(x)$, is computed as shown in (16).

$$f_1(x) = \frac{f_{org}(x)}{s_1(x)} \tag{16}$$

To develop the second sub-function $s_2(x)$, the coefficient c_2 is defined according to (7). The determined value of the coefficient is shown in (17).

$$s_2(x) = 1 + \left(\left(0.400858 \right) \cdot \left(x - x^2 \right) \right) \tag{17}$$

The second function $f_2(x)$, is computed as shown in (18).

$$f_2(x) = \frac{f_1(x)}{s_2(x)} \tag{18}$$

To develop the third sub-functions $s_3(x)$, the second function $f_2(x)$, is divided into its two partial functions as shown in (8). The third order of sub-functions is thereby divided into two sub-functions, where $s_{3,0}(x_3)$ is restricted to the interval $0 \le x < 0.5$ and $s_{3,1}(x_3)$ is restricted to the interval $0.5 \le x < 1.0$ according to (9). A normalization of x to x_3 is done to simplify in the implementation in hardware, which is described in (10).
For each sub-function, the corresponding coefficients $c_{3,0}$ and $c_{3,1}$ is determined. These coefficients are determined according to (12) where higher order sub-functions can be developed. The determined values of the coefficients are shown in (19).

$$\begin{aligned} s_{3,0}(x_3) &= 1 + \left(-0.0122452 \cdot \left(x_3 - x_3^2 \right) \right) \quad 0 \le x < 0.5 \\ s_{3,1}(x_3) &= 1 + \left(0.0105947 \cdot \left(x_3 - x_3^2 \right) \right) \quad 0.5 \le x < 1 \end{aligned} \tag{19}$$

The third function $f_3(x)$, is computed as shown in (20).

$$f_3(x) = \frac{f_2(x)}{s_3(x)} \tag{20}$$

To develop the fourth sub-functions $s_4(x)$, the third function $f_3(x)$, is divided into its four partial functions as shown in (8). The fourth order of sub-functions is thereby divided into four sub-functions, where $s_{4,0}(x_4)$ is restricted to the interval $0 \le x < 0.25$, $s_{4,1}(x_4)$ is restricted

to the interval $0.25 \le x < 0.5$, $s_{4,2}(x_4)$ is restricted to the interval $0.5 \le x < 0.75$ and $s_{4,3}(x_4)$ is restricted to the interval $0.75 \le x < 1.0$ according to (9). A normalization of x to x_4 is done to simplify the hardware implementation, which is described in (10).
For each sub-function, the corresponding coefficients $c_{4,0}$, $c_{4,1}$, $c_{4,2}$ and $c_{4,3}$ is determined. These coefficients are determined according to (12) which accomplish that higher order of sub-functions can be developed. The determined values of the coefficients are shown in (21).

$$\begin{aligned} s_{4,0}(x_4) &= 1 + \left(-0.00223363 \cdot \left(x_4 - x_4^2\right)\right) \quad 0 \le x < 0.25 \\ s_{4,1}(x_4) &= 1 + \left(0.00192558 \cdot \left(x_4 - x_4^2\right)\right) \quad 0.25 \le x < 0.5 \\ s_{4,2}(x_4) &= 1 + \left(-0.00119216 \cdot \left(x_4 - x_4^2\right)\right) \quad 0.5 \le x < 0.75 \\ s_{4,3}(x_4) &= 1 + \left(0.00126488 \cdot \left(x_4 - x_4^2\right)\right) \quad 0.75 \le x < 1 \end{aligned} \tag{21}$$

No postprocessing is needed since the result out from the processing part has the right size.

4.3 Optimization

If no more sub-functions are to be developed the precision of the approximation can be further improved by optimization of coefficients $c_{4,0}$, $c_{4,1}$, $c_{4,2}$ and $c_{4,3}$. As shown in Fig. 12 sub-function $s_{4,3}(x)$ in the interval $0.75 \le x < 1.0$ has the largest relative error. When performing an optimization of sub-function $s_{4,3}(x)$ in the interval $0.75 \le x < 1.0$ it was found that the word length in the computations could be reduced from 17 bits to 16 bits.

4.4 Architecture

In Fig. 11, architecture of the approximation of the sine function using the proposed methodology is shown.
The x^2 block in Fig. 11 is the special designed multiplier described in Fig. 8 and Fig. 9 that delivers the partial results q, q_3 and q_4 used in the following blocks. In the x-q block, x is subtracted with the partial result q, from the x^2 block. The result r from the x-q block is then used in the two following blocks as shown in Fig. 11. In the $x+(c_1 \cdot r)$ block is $s_1(x)$ performed, in $1+(c_2 \cdot r)$ is $s_2(x)$ performed, in $1+(c_3 \cdot (x_3 - q_3))$ is $s_3(x)$ performed and in $1+(c_4 \cdot (x_4 - q_4))$ is $s_4(x)$ performed. Note, that in the blocks for sub-function $s_3(x)$ and $s_4(x)$, the individual index m is addressing the MUX that selects the coefficients in the block.

4.5 Optimization of Word Length

As shown in (19) and (21) the absolute value of the coefficients decreases in size with increasing index number of the coefficient. In similarity to the word length of the coefficients the word length of the $(x_n - q_n)$ part, shown in Fig. 11, will decrease in size with increasing index number. The decreased word length will course that the size of the multiplier used in a sub-function to decreas accordingly to the highest value bit in the coefficients and of the $(x_n - q_n)$ part. In resemblance to above the size of the multipliers computing the multiplication of the sub-functions can be analyzed. This analysis will also result in that some of the following multipliers accordingly can be decrease in size.

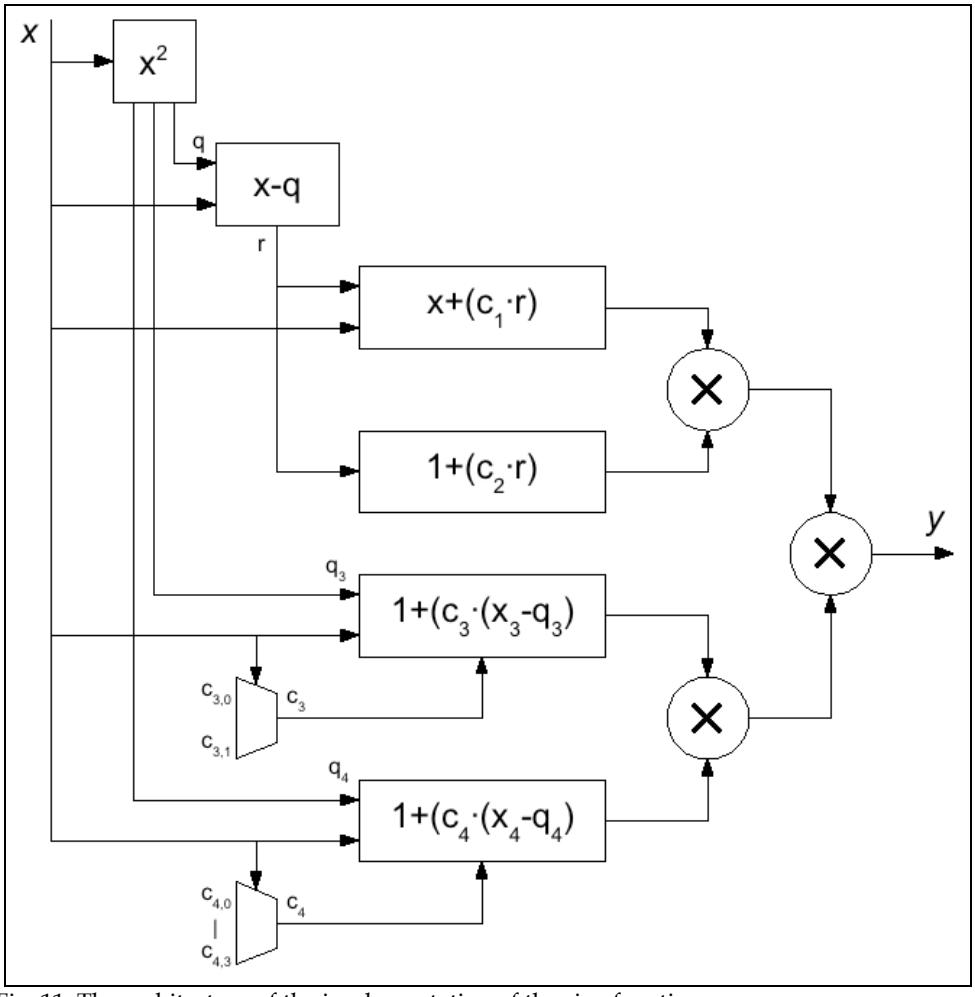


Fig. 11. The architecture of the implementation of the sine function.

4.6 Precision

In Fig. 12 the resulting precision when using one to four sub-functions is shown. Decibel scale is used to visualize the precision since the combination of binary numbers and dB works very well together. In dB scale 2 is equal to $20\log_{10}(2) = 20 \cdot (0.3) = 6$ dB and since 6 dB corresponds to 1 bit, this will make it simpler to understand the result. As shown in Fig. 12, the relative error decreases with the number of used sub-functions. With 4 sub-functions we can see that we have accuracy better that 14 bits that will result in at least a latency of 14 adders in the CORDIC algorithm is used.

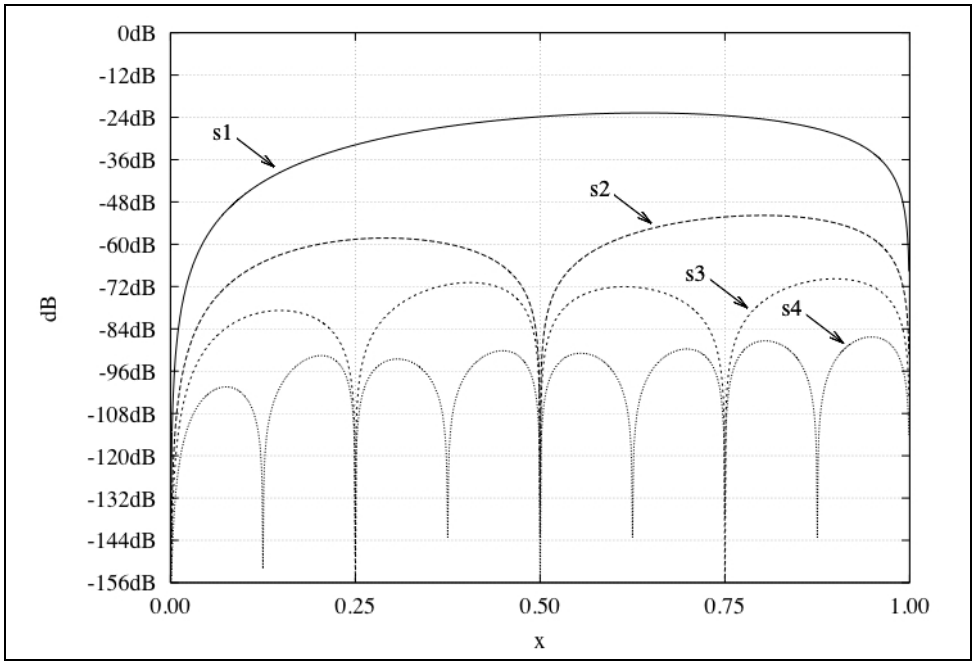


Fig. 12. Estimation of the relative error between the original function and different numbers of sub-functions.

As shown in Fig. 12, the relative error decreases with the number of sub-functions used. However, increases the delay with the number of sub-function as shown in Table 1.

Number of sub-functions	Delay
1	2 mult + 2 add
2	3 mult + 2 add
3 to 4	4 mult + 2 add
5 to 8	5 mult + 2 add

Table 1. Delay relative the number of sub-functions.

The delay of one multiplier is about two adders.

5. Comparison

The most common methods used when implementing approximation of a unary functions in hardware are look-up tables, polynomials, table-based methods with polynomials and CORDIC. Computation by table look-up is attractive since memory is much denser than random logic in VLSI realizations. However, since the size of the look-up table grows exponentially with increasing word lengths, both the table size and execution time becomes totally intolerable. Computation by polynomials is attractive since it is ROM-less. The disadvantages are that it can impose large computational complexities and delays.

Computation by table-based methods combined with polynomials is attractive since it reduces the computational complexity and decreases the delays. Since the size of the look-up tables grows with the accuracy the execution time also increases with the needed accuracy. Computation by using CORDIC is attractive since it is using an angular rotation algorithm that can be implemented with small look-up tables and hardware, which is limited to simple shifts and additions. The CORDIC algorithm is an iterative method with high latency and long delays. This makes the method insufficient for applications where short execution time is essential.
In all methods including the proposed method, it is a trade-off between complexity and memory storage. By using parallelism in the computation and parabolic synthesis in the recombination process, the proposed methodology thereby gets a short critical path, which assures fast computation.

6. Using the Methodology

It has been shown that the methodology of parabolic synthesis can directly compute the sine function but the methodology is also able to compute other trigonometric functions, logarithms as well as square root and division. In the following parts algorithms for elementary functions will be shown.
When describing the implementation of each function the different parts are shown in a table. The first row in the table shows the function to be implemented and in which interval the function is implemented. In the second row it is described how to perform the normalization of the function. The third row shows the original function to be used when developing the approximation. The last row describes how to perform transformation of the approximation into desired interval.

6.1 The Sine Function

When developing the algorithm that performs the approximation of the sine function, the normalization in the preprocessing part is performed as a substitution according to Table 2. Since the outcome of the approximation has the desired form no postprocessing is needed.

	Algorithm	Range
Function	$f(v) = \sin(v)$	$0 \le v < \frac{\pi}{2}$
Preprocessing	$x = \frac{2}{\pi} \cdot v$	$0 \le x < 1$
Processing	$y = \sin\left(\frac{\pi}{2} \cdot x\right)$	$0 \le y < 1$
Postprocessing	$z = y$	$0 \le z < 1$

Table 2. The algorithms for the sine function.

6.2 The Cosine Function

The algorithm that performs the approximation of the cosine function is founded on the algorithm that performs the approximation of the sine function. To perform the

approximation of the cosine function x is substituted with $1-x$ in the preprocessing part of the approximation for the sine function.

	Algorithm	Range
Function	$f(v)=\cos(v)$	$0 < v \le \frac{\pi}{2}$
Preprocessing	$x = 1 - \frac{2}{\pi} \cdot v$	$0 \le x < 1$
Processing	$y = \sin\left(\frac{\pi}{2} \cdot x\right)$	$0 \le y < 1$
Postprocessing	$z = y$	$0 \le z < 1$

Table 3. The algorithm for the cosine function.

6.3 The Arcsine Function

When developing the algorithm that performs the approximation of the arcsine function, the methodology has problems to perform an approximation for angels larger than $\pi/4$. Therefore, the range of the approximation has been limited according to the range of the function in Table 4. To satisfy the requirements of the methodology in the preprocessing part a substitution according to Table 4 has to be performed. To get the desired outcome the approximation is multiplied with a factor according to Table 4.

	Algorithm	Range
Function	$f(v)=\arcsin(v)$	$0 \le v < \frac{1}{\sqrt{2}}$
Preprocessing	$x = \sqrt{2} \cdot v$	$0 \le x < 1$
Processing	$y = \arcsin\left(\frac{x}{\sqrt{2}}\right)$	$0 \le y < 1$
Postprocessing	$z = \frac{\pi}{4} \cdot y$	$0 \le z < \frac{\pi}{4}$

Table 4. The algorithm for the arcsine function.

6.4 The Arccosine Function

The algorithm that performs the approximation of the arccosine function is founded on the algorithm performing the approximation of the arcsine function. The difference between the two approximations is in the transformation in the postprocessing part, as shown in Table 5.

	Algorithm	Range
Function	$f(v) = \arccos(v)$	$0 \le v < \frac{1}{\sqrt{2}}$
Preprocessing	$x = \sqrt{2} \cdot v$	$0 \le x < 1$
Processing	$y = \arcsin\left(\frac{x}{\sqrt{2}}\right)$	$0 \le y < 1$
Postprocessing	$z = \frac{\pi}{4} + \frac{\pi}{4} \cdot (1 - y)$	$\frac{\pi}{4} < z \le \frac{\pi}{2}$

Table 5. The algorithm for the arccosine function.

6.5 The Tangent Function

When developing the algorithm that performs the approximation of the tangent function the angle range is from 0 to $\pi/4$, since the tangent function is not strictly concave or convex for higher angles. To perform the normalization the preprocessing part is performed as a substitution according to Table 6. Since the outcome of the approximation has the desired form no postprocessing is needed.

	Algorithm	Range
Function	$f(v) = \tan(v)$	$0 \le v < \frac{\pi}{4}$
Preprocessing	$x = \frac{4 \cdot v}{\pi}$	$0 \le x < 1$
Processing	$y = \tan\left(\frac{\pi}{4} \cdot x\right)$	$0 \le y < 1$
Postprocessing	$z = y$	$0 \le z < 1$

Table 6. The algorithm for the tangent function.

6.6 The Arctangent Function

When developing the algorithm that performs the approximation of the arctangent function it can only be performed in the range from 0 to 1 where the function is strictly concave or convex. To get the desired outcome the approximation is in the postprocessing part multiplied with a factor according to Table 7.

	Algorithm	Range
Function	$f(v) = \arctan(v)$	$0 \le v < 1$
Preprocessing	$x = v$	$0 \le x < 1$
Processing	$y = \arctan(x) \cdot \frac{4}{\pi}$	$0 \le y < 1$
Postprocessing	$z = \frac{\pi}{4} \cdot y$	$0 \le z < \frac{\pi}{4}$

Table 7. The algorithm for the arctangent function.

6.7 The Logarithmic Function

When developing the algorithm that performs the approximation of the logarithm function with the base two, it is only perform on the mantissa of the floating-point number, since the exponent part is scaling the mantissa. For the preprocessing part a substitution according to Table 8 has to be performed to satisfy the normalization criteria's for the methodology. Since the outcome of the approximation has the desired form no postprocessing is needed.

	Algorithm	Range
Function	$f(v) = \log_2(v)$	$1 \le v < 2$
Preprocessing	$x = v - 1$	$0 \le x < 1$
Processing	$y = \log_2(1 + x)$	$0 \le y < 1$
Postprocessing	$z = y$	$0 \le z < 1$

Table 8. The algorithm for the logarithm function.

6.8 The Exponential Function

When developing the algorithm that performs the approximation of the exponential function with the base two, it is only performed on the fractional part of the logarithm since the integer part is scaling the fractional part of the logarithm. As shown in Table 9 only a one needs to be added in the postprocessing part to get the desired outcome.

	Algorithm	Range
Function	$f(v) = 2^v$	$0 \le v < 1$
Preprocessing	$x = v$	$0 \le x < 1$
Processing	$y = 2^x - 1$	$0 \le y < 1$
Postprocessing	$z = 1 + y$	$1 \le z < 2$

Table 9. The algorithm for the exponential function.

6.9 The Division Function

When developing the algorithm that performs the approximation of the division it is limited to the range according to Table 10, since the division is not strictly concave or convex outside this range. The pre- and post-processing part both needs computation when performing the approximation of the division.

	Algorithm	Range
Function	$f(v) = \frac{1}{1+v}$	$0.5 < v \le 1$
Preprocessing	$x = 2 \cdot (1 - v)$	$0 \le x < 1$
Processing	$y = \frac{6}{1 + \left(1 - \frac{x}{2}\right)} - 3$	$0 \le y < 1$
Postprocessing	$z = \frac{3+y}{6}$	$\frac{1}{2} \le z < \frac{2}{3}$

Table 10. The algorithm for the division function.

6.10 The Square Root Function

When developing the algorithm that performs the approximation of the square root function the range is limited according to Table 11. The pre- and post-processing part both needs computation when performing the approximation of the square root function.

	Algorithm	Range
Function	$f(v)=\sqrt{1+v}$	$1 \le v < 2$
Preprocessing	$x = v - 1$	$0 \le x < 1$
Processing	$y = \frac{\sqrt{2+x}-\sqrt{2}}{\sqrt{3}-\sqrt{2}}$	$0 \le y < 1$
Postprocessing	$z = \sqrt{2} + y \cdot \left(\sqrt{3}-\sqrt{2}\right)$	$\sqrt{2} \le z < \sqrt{3}$

Table 11. The algorithm for the square root function.

7. Conclusions

A novel methodology for implementing approximations of unary functions such as trigonometric functions, logarithmic functions, as well as square root and division functions etc. in hardware is introduced. The architecture of the processing part automatically gives a high degree of parallelism. The methodology to develop the approximation algorithm is founded on parabolic synthesis. This combined with that the methodology is founded on operations that are simple to implement in hardware such as addition, shifts, multiplication, contributes to that the implementation in hardware is simple to perform. By using the parallelism and parabolic synthesis, one of the most important characteristics with the out coming hardware is the parallelism that gives a short critical path and fast computation. The structure of the methodology will also assure an area efficient hardware implementation. The methodology is also suitable for automatic synthesis.

8. References

B. Parhami (2000), *Computer Arithmetic*, Oxford University Press Inc., ISBN: 0-19-512583-5, 198 Madison Avenue, New York, New York 10016, USA.

E. Hertz, P. Nilsson (2008), A Methodology for Parabolic Synthesis of Unary Functions for Hardware Implementation, *Proc. of the 2nd International Conference on Signals, Circuits and Systems*, SCS08_9.pdf, pp 1-6, ISBN-13: 978-1-4244-2628-7, Hammamet, Tunisia, Nov. 2008, Inst. of Elec. and Elec. Eng. Computer Society, 445 Hoes Lane - P.O.Box 1331, Piscataway, NJ 08855-1331, United States.

E. Hertz, P. Nilsson (2009), Parabolic Synthesis Methodology Implemented on the Sine Function, *Proc. of the 2009 IEEE International Symposium on Circuits and Systems*, pp 253-256, ISBN: 978-1-4244-3828-0, Taipei, Taiwan, May. 2009.

E. Meijering (2002), A Chronology of Interpolation From Ancient Astronomy to Modern Signal and Image Processing, *Proceedings of the IEEE*, vol. 90, no. 3, March 2002, pp. 319-342, ISSN: 00189219, Institute of Electrical and Electronics Engineers Inc.

J. E. Volder (1959), The CORDIC Trigonometric Computing Technique, *IRE Transactions on Electronic Computers*, vol. EC-8, no. 3, 1959, pp. 330–334.

J. Fourier (1822), *Théorie Analytique de la Chaleur*, Paris, France.

J.-M. Muller (2006), *Elementary Functions: Algorithm Implementation, second ed*. Birkhauser, ISBN 0-8176-4372-9, Birkhauser Boston, c/o Springer Science+Business Media Inc., 233 Spring Street, New York, NY 10013, USA.

K. Z. Pekmestzi, P. Kalivas, and N. Moshopoulos (2001), Long unsigned number systolic serial multipliers and squarers, *IEEE Circuits and Systems II:*, vol. 48, no 3, March 2001, pp. 316 -321, ISSN 1057-7130.

P. Ienne, M. A. Viredaz (1994), Bit-serial multipliers and squarers, *IEEE Transactions on Computer*, vol. 43, no12, Dec. 1994, pp. 1445 -1450, ISSN 0018-9340.

P.T.P. Tang (1991), Table-lookup algorithms for elementary functions and their error analysis, *Proc. of the 10th IEEE Symposium on Computer Arithmetic*, pp. 232 – 236, ISBN: 0-8186-9151-4, Grenoble, France, June 1991.

R. Andrata (1998), A survey of CORDIC algorithms for FPGA based computers, *Proc. of the 1998 ACM/SIGDA Sixth Inter. Symp. on Field Programmable Gate Array* (FPGA'98), pp. 191-200, ISBN: 0-89791-978-5 , Monterey, CA, Feb. 1998, ACM Inc.

11

Fully Systolic FFT Architectures for Giga-sample Applications

D. Reisis
Department of Physics, Electronics Laboratory,
National Kapodistrian University of Athens,
Greece

1. Introduction

This chapter presents a technique for designing architectures, which execute Fast Fourier Transform (FFT) algorithms involving a large number of complex points and they target real-time applications (Ersoy, 1997). Hitherto published techniques and FFT architectures include mainly ASIC designs (Thompson, 1983; Wold and Despain, 1984; He and Torkelson, 1996; Choi et al., 2003; Uzun et al., 2005; Bouguezel et al., 2004, 2006; Jo & Sunwoo, 2005; Chang & Nguyen, 2006; Yang et al., 2006; Lin et al., 2005; Takala & Punkka, 2006; Wang & Li, 2007; Reisis & Vlassopoulos, 2006), which vary with respect to the level of parallelism, the throughput rate, the latency, the hardware cost and the power consumption. The most common ASIC architectures are the fully unfolded FFT realizations (Rabiner & Gold) utilizing large memory arrays between their successive stages and also the latency and memory efficient cascade FFT topologies (Thomson, 1983; He & Torkelson, 1996, 1998). The cascade solutions though, as well as the high radix techniques lead to complicated designs in the case of architectures parameterized with respect to the pipelining of computations, the FFT size and the data length.

This chapter describes a technique to design efficient, very high speed, deeply-pipelined FFT architectures maximizing throughput and keeping control and memory organizations simple compared to the cascade and the fully unfolded FFT architectures. Moreover, the design is proven more efficient comparing to the previously mentioned architectures in terms of scalability, maximum operating frequency and consequently, in terms of power consumption, pipeline depth and data and/or twiddle bit-widths. The technique improves the latency and the memory requirements -particularly for large input data sets- of systolic FFT architectures by combining three (3) Radix-4 circuits to result in a 64-point FFT engine.

The efficiency of organizing 64-point FFT engines based on Radix-4 FFT engines is shown by a 4096-complex point design. This architecture requires only two dual memory banks of 4096 words each and on a Xilinx Virtex II FPGA performs at 200 MHz to sustain a throughput of 4096 points/20.48 us. The design implemented on a high performance 0.13 um, 1P8M CMOS (standard cell) process from UMC achieved a worst-case (0.9V, 125 C) post-route frequency of 604.5 MHz, while consuming 4.4 Watts. It is interesting to point out

that the design exceeded the 1 GHz frequency (rate of 1 GSample/sec) for typical conditions (1.0V, 25C).
Towards designing architectures FFTs with large input data sets we consider the 4096 complex point FFT architecture as a core. The core constitutes the basis of FFT architectures computing transforms of 16K, 64K and 256K complex points. These architectures implemented in the 0.13 CMOS process perform at 352, 256 and 188 Mhz worst-case (0.9V, 125 C) post-route frequencies respectively. The 16K and the 64K architectures have a four point parallel input/output achieving throughputs of 1.4 and 1 GSample/sec respectively. Further, this chapter will present a technique, which allows the parallelization of the memory accesses in hardware implementations of the FFT algorithm. This technique enables each processor to perform a radix-b butterfly by loading the b-tuple data from b memory banks in parallel, then by operating on the b data and finally, by storing the resulting b-tuple in b memory banks in parallel. Hence, the speedup and the throughput increase by b. Techniques parallelizing the FFT accesses are reported in (Johnson, 1992; Ma, 1999; Reisis & Vlassopoulos, 2006). We describe the technique in (Reisis & Vlassopoulos, 2006), which is developed for arbitrary radix and straightforward to implement.
The chapter is organized in three technical sections. Section 2 shows how to organize radix-4 computations to result in a radix-4^3–equivalent to radix-64–computation and describes the fully systolic 4096-point architecture. Section 3 will describe the 16K, 64K and 256K point architectures. Section 4 presents the access parallelization technique in the FFT architectures and section 5 concludes the chapter.

2. A Fully Systolic 4096 complex point FFT Architecture

This section presents an attractive solution combining the advantages of both the cascaded (Thompson, 1983; He & Torkelson, 1998; OH & Lim, 2005) and the unfolded (Rabiner & Gold; Thompson, 1983) structures. The organization uses the unfolded structure with radix-64 (R64) stages. We note that 64 is a power of either 2 or 4 and it is sufficiently large to result in small number of stages. Realizing the 64 point FFT at high speed with radices greater than 2 and 4 becomes inefficient, because the use of higher radices imposes complicated structures and leads to lower the architecture's performance. R4 computations are equally straightforward to implement as the R2, but R4 designs reduce the number of complex multipliers to the half of those required by the R2 designs. Also, R4 process 4 points in a 2-step computation *accumulating-multiplying* with the *multiplying* reduced to *sign-inversion* and *swap* operations. This result minimizes the number of possible bottlenecks in the data flow and allows the maximal pipelining and/or parallelization of the four computing steps. Consequently, the choice of R4 leads to a more efficient design with respect to the multipliers, the pipeline depth and the scalability of the operating frequency and/or of the word length.

2.1 Radix-4^3 Algorithm

To produce the $R4^3$ structure we start with the Discrete Fourier Transform (DFT) of a signal x[n] of length N,

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \tag{1}$$

where, $W_N = e^{-j\frac{2\pi}{N}}$ are the twiddle factors and denote the N-th primitive root of unity (Oppenheim, 1975). The architecture presented in this section is based on a four-dimensional index map and on a R4 decomposition of the DFT series. The derivation of the Radix − 4^3 algorithm lies on three (3) steps in the cascade decomposition. In the frame work of these 3 steps, the linear mapping transforms into a four-dimentional index map [10,4] as follows:

$$\begin{aligned} n &= n_1 + \frac{N}{64}n_2 + \frac{N}{16}n_3 + \frac{N}{4}n_4 \\ k &= 64k_1 + 16k_2 + 4k_3 + k_4 \end{aligned} \tag{2}$$

Applying equation 2 to the DFT equation yields

$$X(64k_1 + 16k_2 + 4k_3 + k_4) = \sum_{n_1=0}^{\frac{N}{64}-1}\sum_{n_2=0}^{3}\sum_{n_3=0}^{3}\sum_{n_4=0}^{3} x(n_1 + \frac{N}{64}n_2 + \frac{N}{16}n_3 + \frac{N}{4}n_4)W_N^{nk} \tag{3}$$

The composite twiddle factors can be expressed as follows

$$\begin{aligned} W_N^{kn} &= W_N^{[n_1+\frac{N}{64}n_2+\frac{N}{16}n_3+\frac{N}{4}n_4][64k_1+16k_2+4k_3+k_4]} \Rightarrow \\ W_N^{kn} &= (-j)^{n_2k_2+n_3k_3+n_4k_4} W_{16}^{n_3k_4} W_{64}^{n_2(4k_3+k_4)} \times W_N^{n_1(16k_2+4k_3+k_4]} W_{\frac{N}{64}}^{n_1k_1} \end{aligned} \tag{4}$$

Applying equation 4 into equation 3 and expanding the summation with index n_4 yield

$$\begin{aligned} &X(64k_1 + 16k_2 + 4k_3 + k_4) = \\ &\sum_{n_1=0}^{\frac{N}{64}-1}\sum_{n_2=0}^{3}\sum_{n_3=0}^{3}\left[B_{\frac{N}{4}}^{k_4}\left(n_1 + \frac{N}{64}n_2 + \frac{N}{16}n_3\right)\right]\times \\ &\times(-j)^{n_2k_2+n_3k_3} W_{16}^{n_3k_4} W_{64}^{n_2(4k_3+k_4)} \times W_N^{n_1(16k_2+4k_3+k_4)} W_{\frac{N}{64}}^{n_1k_1} \end{aligned} \tag{5}$$

Where $B_{\frac{N}{4}}^{k_4}\left(n_1 + \frac{N}{64}n_2 + \frac{N}{16}n_3\right)$ denotes the first butterfly unit and can be written as

$$\begin{aligned} B_{\frac{N}{4}}^{k_4}\left(n_1 + \frac{N}{64}n_2 + \frac{N}{16}n_3\right) &= x\left(n_1 + \frac{N}{64}n_2 + \frac{N}{16}n_3\right) + (-j)^{k_4} x\left(n_1 + \frac{N}{64}n_2 + \frac{N}{16}n_3 + \frac{N}{4}\right) \\ &+ (-1)^{k_4} x\left(n_1 + \frac{N}{64}n_2 + \frac{N}{16}n_3 + \frac{N}{2}\right) + j^{k_4} x\left(n_1 + \frac{N}{64}n_2 + \frac{N}{16}n_3 + \frac{3N}{4}\right) \end{aligned} \tag{6}$$

Expanding equation 5 with respect to the next summation with index n_3 yields

$$X(64k_1+16k_2+4k_3+k_4)= \sum_{n_1=0}^{\frac{N}{64}-1}\sum_{n_2=0}^{3} H_{\frac{N}{16}}^{k_3k_4}\left(n_1+\frac{N}{64}n_2\right)(-j)^{n_2k_2}\times W_{64}^{n_2(4k_3+k_4)}W_N^{n_1(16k_2+4k_3+k_4)}W_{\frac{N}{64}}^{n_1k_1} \tag{7}$$

where $H_{\frac{N}{16}}^{k_3k_4}\left(n_1+\frac{N}{64}n_2\right)$ is the secondary butterfly structure and can be expressed as

$$\begin{aligned} H_{\frac{N}{16}}^{k_3k_4}\left(n_1+\frac{N}{64}n_2\right) &= B_{\frac{N}{4}}^{k_4}\left(n_1+\frac{N}{64}n_2\right)+(-j)^{k_3}W_{16}^{k_4}B_{\frac{N}{4}}^{k_4}\left(n_1+\frac{N}{64}n_2+\frac{N}{16}\right)+ \\ &+(-1)^{k_3}W_8^{k_4}B_{\frac{N}{4}}^{k_4}\left(n_1+\frac{N}{64}n_2+\frac{N}{8}\right)+j^{k_3}W_8^{k_4}W_{16}^{k_4}B_{\frac{N}{4}}^{k_4}\left(n_1+\frac{N}{64}n_2+\frac{3N}{16}\right) \end{aligned} \tag{8}$$

Finally, expanding the summation of equation 7 with regard to index n_2 provides a set of 64 DFTs of length N/64.

$$X(64k_1+16k_2+4k_3+k_4)=\sum_{n_1=0}^{(N/64)-1}\left[T_{N/64}^{k_2k_3k_4}(n_1)W_N^{n_1(16k_2+4k_3+k_4)}\right]W_{N/64}^{n_1k_1} \tag{9}$$

where $T_{N/64}^{k_2k_3k_4}(n_1)$ represents the third butterfly and is expressed according to equation 10

$$\begin{aligned} T_{\frac{N}{64}}^{k_2k_3k_4}(n_1) &= H_{\frac{N}{16}}^{k_3k_4}(n_1)+(-j)^{k_2}W_{16}^{k_3}W_{64}^{k_4}H_{\frac{N}{16}}^{k_3k_4}\left(n_1+\frac{N}{64}\right)+ \\ &+(-1)^{k_2}W_8^{k_3}W_{32}^{k_4}H_{\frac{N}{16}}^{k_3k_4}\left(n_1+\frac{N}{32}\right)+ \\ &+j^{k_2}W_{64}^{3(4k_3+k_4)}H_{\frac{N}{16}}^{k_3k_4}\left(n_1+\frac{3N}{64}\right) \end{aligned} \tag{10}$$

Equations 5 to 10 describe a radix-64 based FFT. Further, equations 6, 8 and 10 describe the internal structure of the radix-64 butterfly based on three radix-4 butterflies and therefore called Radix - 4^3 ($R4^3$).

2.2 $R4^3$ Features

Figure 1 shows the flow diagram of the $R4^3$ computations where Δ, $\circ$ and $\triangleright$ present multiplications by $(-j)^{k_3}W_{16}^{k_3}$, $(-1)^{k_3}W_8^{k_4}$ and $j^{k_3}W_8^{k_4}W_{16}^{k_4}$ respectively, while the $\Diamond$, ∇, $\triangleleft$ present multiplications by $(-j)^{k_3}W_{16}^{k_3}W_{64}^{k_4}$, $(-1)^{k_2}W_8^{k_3}W_{32}^{k_4}$ and $j^{k_2}W_{64}^{3(4k_3+k_4)}$. Note that the order of the twiddle factors in the $R4^3$ differs from the respective order of the R64.

The FFT architecture with $R4^3$ stages and input N complex points uses $\log_4 N-1$ complex multipliers. With each R4 using 3 complex adders to produce 1 result/cycle the architecture has $3\log_4 N$ complex adders. The memory size is $(N/3)\log_4 N$.
Comparing to the cascaded $R2^2$ (He & Torkelson, 1998) and $R2^4$ (OH & Lim, 2005) the proposed unfolded $R4^3$ has equal number of multipliers (Table 1). The $R4^3$ has less complex adders ($3\log_4 N$) than the cascaded ($4\log_4 N$) but uses larger memory size. The $R4^3$ FFT design achieves the least number of multipliers and adders in the literature, equal to that in (Bidet et al., 1995), and although it requires larger memory, it uses simple control and it can achieve high frequency performance as it is shown in the following subsection.

2.3 Architecture

Figure 2 depicts the architecture of the 4096-point FFT. The implementation of the 4096-point FFT using $R4^3$ butterflies includes: first the use of a 2-dimensional index map based on R64, second the decomposition of the 4096-point series into two sums using the R64

	Complex Multipliers	Complex Adders	Memory	Control
R2MDC *	$2\log_4 N - 1$	$4\log_4 N$	3N/2-2	Simple
R2SDF *	$2\log_4 N - 1$	$4\log_4 N$	N − 1	Simple
R4SDF *	$\log_4 N - 1$	$8\log_4 N$	N − 1	Medium
R4MDC *	$3(\log_4 N - 1)$	$8\log_4 N$	5N/2 − 4	Simple
R4SDC *	$\log_4 N - 1$	$3\log_4 N$	2N − 2	Complex
$R2^2$SDF **	$\log_4 N - 1$	$4\log_4 N$	N − 1	Simple
$R4^3$	$\log_4 N - 1$	$3\log_4 N$	(N/3)log4N	Simple

*(He & Torkelson, 1998), **(OH & Lim, 2005)
Table 1. FFT architecture hardware complexity comparison

butterfly and finally the replacement of each R64 with a $R4^3$. Consequently, the architecture consists of two $R4^3$ engines, two 4096-word dual bank memory modules, one 4096 point read-only memory storing the W_{4096} twiddles and one complex multiplier. The control is local to each module and a global sync signal synchronizes the modules. The throughput is 1 complex-point/clock-cycle.
The overall design allows the optimization of the architecture, either globally, or locally: The architecture can be improved with respect to the operating frequency, or the area, or both without alterations in the control, the scheduling of the 4K buffers, the registers and the memories within the $R4^3$ engines. Hence, we can modify the pipelined computations and exploit the specifics of each technology to maximize the operating frequency.
The following subsections describe the details of the R4 processor used as the basis of the $R4^3$, the $R4^3$ architecture, the addressing scheme used for each $R4^3$ engine and discuss the performance of the entire FFT architecture.

2.3.1 The Radix-4 basis of the $R4^3$

Figure 3 depicts the Radix-4 engine architecture used in the $R4^3$. Each Radix-4 engine consists of one complex Accumulator and one ”Swap” module. The task of the ”Swap” module is to switch the real part of each input value with the corresponding imaginary and

the opposite in order to perform the correct butterfly operations. Figure 4 depicts the architecture of the Accumulator module. The accumulator consists of 8 registers and 3 add/sub units. The four ″A″ registers are used as input registers storing every 4-tuple of data, on which the R4 butterfly operation will be applied. The four ″B″ resisters are used as an intermediate stage holding the 4-tuple (while the next 4-tuple of input data is shifted into the ″A″ registers) and loading the data in parallel to the add/sub units. The add-sub units form an adder-tree architecture, thus avoiding the feedback loops that common accumulators use.

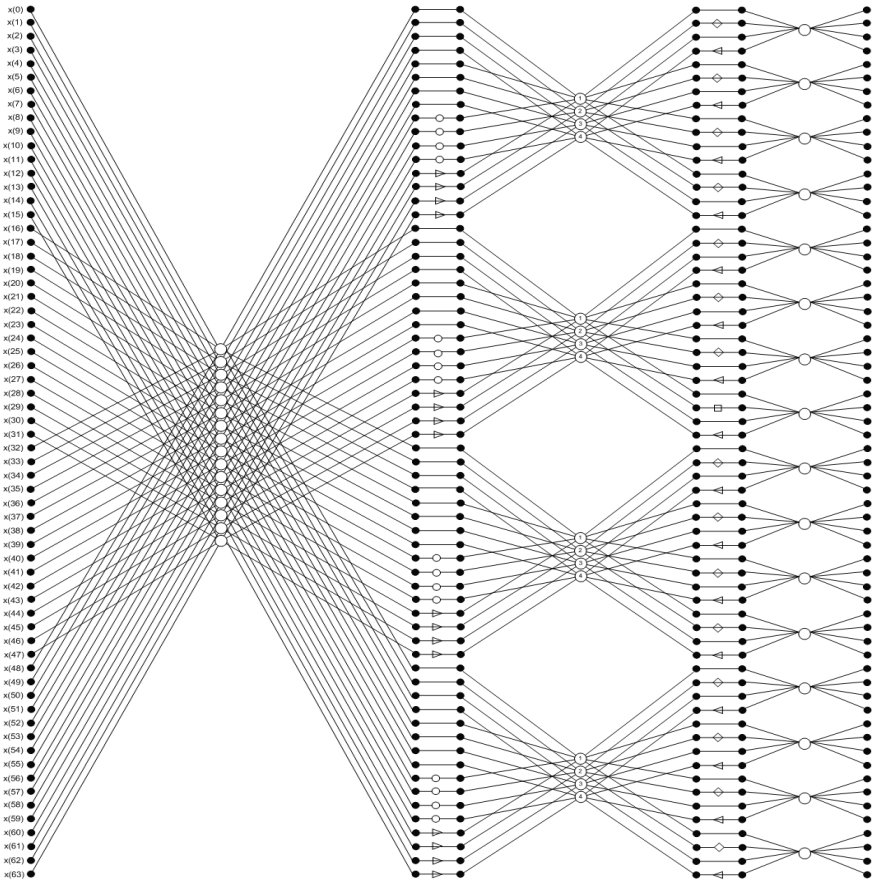


Fig. 1. Radix − 4^3 Butterfly SFG

The data flow of the Radix-4 engine starts with the data entering the Radix-4 engine as a word serial input stream at a rate of one complex-point/cycle. During each period of 4 cycles four consecutive input data are shifted in the 4 input registers (Reg A). In the fifth cycle this 4-tuple is latched in the 4 ″storage″ registers (Reg B), while the next input 4-tuple is starting its input to the ″A″ registers. During the sixth cycle the 4-tuple enters in parallel the adder-tree to produce the first of the four R4 results. During the following 3 cycles, the

adder-tree uses as input the 4-tuple stored in the "B" registers and it produces the remaining 3 of the four R4 results, one result/cycle, by following the Radix-4 computational flow. Therefore, the total latency of each accumulator is 5+2k cycles, where k is the latency of each add/sub unit, k = 5 in the implementation.
A control unit synchronizes the operations of the 'Swap' module and the accumulator. A 2-bit counter generates the necessary signals, controlling the add/sub units in order to perform the correct additions and/or subtractions according to the Radix-4 schema.

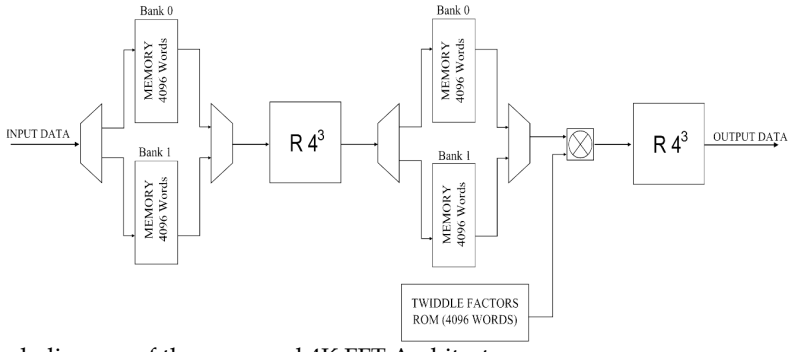


Fig. 2. Block diagram of the proposed 4K FFT Architecture

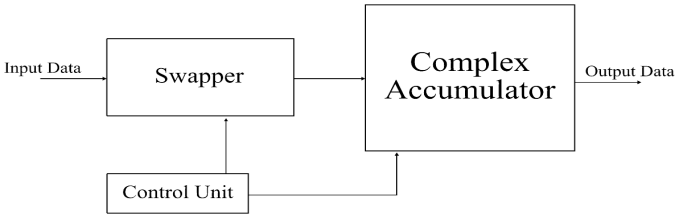


Fig. 3. Overall Engine of the Radix − 4

2.3.2 The $R4^3$ Architecture

Figure 5 shows the $R4^3$ engine consisting of 3 R4 butterflies, 2 complex multipliers, 2 Dual Bank memory elements (1 consisting of 2x64 words and 1 of 2x16 words) and 2 Read Only Memories storing the W_{64} and W_{16} twiddles. The complex twiddle multipliers are 4 integer-multipliers of input 14 × 14 and are pipelined with depth 5. The control of each pipeline-stage is local and all stages are synchronized through a global sync signal. Realizing the R64 stage by the $R4^3$ engine offers a combination of advantages: the simplicity of the circuits as a result of using as basic butterfly small radix (4) and the regularity of the architecture, which is built hierarchically with contiguous $R4^3$ engines at the higher level and with each $R4^3$ formed by three R4 pipelined computations at the basic level. Furthermore, the $R4^3$ design is VSLI area efficient because it utilizes the complex multipliers 75% of the cycles and 100% of the other resources at each cycle. The FFT achieves the dynamic range of 84 dB (amplitude) by using the width of 14 bits for each real and imaginary part of each sample. To maintain the maximum precision (14 bits) for each part, the $R4^3$ uses increased precision for internal

calculations. The bit width of each part is 16 and 20 bits at the output of the first and second R4 stages respectively. At the output of the final (third) R4 stage, the real and imaginary parts are truncated to provide an output of 14 bits each. The twiddle factors are 18 bits real

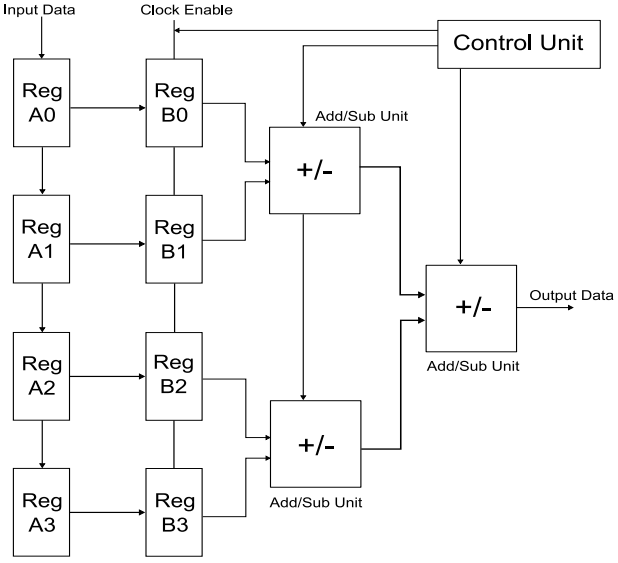


Fig. 4. The Adder Tree of the Complex Accumulator

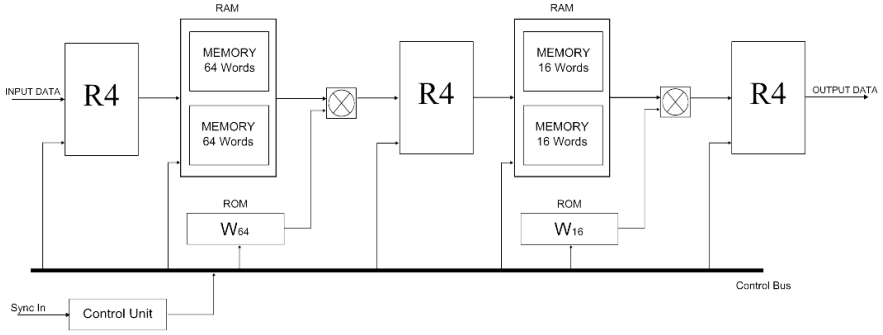


Fig. 5. Block diagram of the Radix − 4^3 Butterfly Architecture

and 18 bits imaginary. By applying the above mentioned data format, the implementation of the 4096 point FFT with fixed point calculations is almost as accurate as a floating point implementation given the dynamic range of 84 dB. The fixed point FFT's outputs are within a margin of +/- (1) compared to the output of a floating point FFT whose values are normalized (divided by N = 4096) and truncated to 14 bit integers.

2.3.3 Data Scheduling and Addressing in R4³

The first stage of the 4K point FFT algorithm is realized by the first R4³ processor, which computes $\frac{4096}{64}$ FFTs with each transform consisting of 64 points. Each transformation 64-tuple contains the elements whose address is of the form 64i + k, where i = 0, . . . , 63 and k is the tuple index, ranging from 0 to $\frac{4096}{64} - 1$.

In the first R4³ processor the data within each 64 -tuple are processed in terms of quadruples, so that the first R4 processor computes the DFT of the elements with address $16i_1 + k_1$, $i_1 = 0, \ldots, 3$ and $k_1 = 0, \ldots, \frac{64}{4} - 1$ is again the tuple index. We can use the notation $[a_{11}, \ldots, a_0]$ for each point's input address (position within the 4096 points) and $[d_5d_4d_3d_2d_1d_0]$ for the address of each bank of the first intermediate memory. The resulting 64 elements are written in one bank of the first intermediate memory in consecutive addresses, i.e. $[a_{11}a_{10}a_9a_8a_7a_6] \rightarrow [d_5d_4d_3d_2d_1d_0]$.

The second R4 reads the data from the first intermediate bank of size 64 by using the address permutation $[a_1a_0a_3a_2a_5a_4] \rightarrow [d_5d_4d_3d_2d_1d_0]$. Every 4 consecutive results of 4-point DFTs produced by the second R4 are stored in one of the two banks of the second intermediate memories of size 16. When the third R4 starts reading these data the second R4 will store the next 4 DFT results into the second intermediate memory of size 16.

Note that the second R4 produces results so that the third R4 can use all the data stored in the intermediate memory to complete its 4-point DFT fully pipelined and with 100% resource utilization. With $[d_3d_2d_1d_0]$ denoting the address within the second intermediate memory, the third R4 reads the data with address permutation $[a_1a_0a_3a_2] \rightarrow [d_3d_2d_1d_0]$.

During the second stage of the 4K FFT algorithm, the second R4³ processor computes $\frac{4096}{64}$ FFTs with each transform consisting of 64 points, so that each transformation 64-tuple contains the elements whose address is of the form 64k + i, where i = 0, . . . , 63 and k is the tuple index, ranging from 0 to $\frac{4096}{64} - 1$. The first R4 produces 64 elements, which are written in one bank of the first intermediate memory in consecutive addresses, i.e. $[a_5a_4a_3a_2a_1a_0] \rightarrow [d_5d_4d_3d_2d_1d_0]$. Apart from this difference the addressing and the data scheduling within the second R4³ is exactly the same as in the first R4³ described above.

2.3.4 FPGA and VLSI Implementation

The proposed architecture has been implemented in RTL VHDL using fixed point arithmetic. The project aims at both high capacity and high speed FPGA devices such as the XILINX Virtex II 6000, as well as a high performance 0.13um standard cell process. For the Xilinx implementation we have used the Xilinx tool and the automatic floor planning option. The Xilinx implementation on the 6000 part results in 20% of logic area utilization and 25% Blocks RAM utilization. The use of optimized Xilinx components (CoreLib multipliers) reduces the area to 13% of the total resources of the 6000 part.

The TSMC 0.13 library implementation achieved 604 MHz worst case. It includes 96K standard cells and 64 RAMs (1361 standard cell rows), (×64 configuration in Table 2) occupying a silicon area of 2630x5129 um square (1.42*10E7 um sq) at 84.2% utilization and achieves a worst-case (0.9V, 125C) post-route performance of 604.5 MHz and a 4.4 Watts

power consumption. The use of typical process parameters (1V, 25C) results in exceeding the 1GHz post-route frequency mark (data rate 1 GSample/sec), making the proposed architecture the fastest standard-cell 4096 complex point FFT implementation reported in the literature. In addition, a second 4094 complex point engine has been implemented in the same standard cell library, this time using deeper RAMs (×16 configuration in Table 2). Power consumption in this case was substantially reduced to 722.8 mW from the 4.4 W of the ×64 configuration. Figure 6 depicts the VLSI layout of the Radix − 4^3 engine, while figure 7 depicts the final layouts of the 4K FFT, of both the ×16 and the ×64 implemented.
The 4K FFT has been designed for an experiment involving a frequency analyzer for a bandwidth of 200 MHz. The band has been divided into four sub-bands and each sub-band has been accommodated by a 4K FFT architecture. The FPGAs perform at 102.5 MHz on a 18-layer board which has a compact-PCI interface performing at 51.25 MHz. The task is to perform FFT and use a ”Threshold” filter to identify the frequencies of high power within each sub-band. The expected output set includes at most 10 frequencies per sub-band, per FFT. After the prototype completion the 4K FFT has been delivered as an IP core with specifications achieving 5 times the performance of the FPGA prototype. The 16K, 64K and 256K have been realized as IP cores for research purposes.

2.4 Architecture’s Performance and Advantages

The proposed architecture demonstrates improved latency compared to other unfolded architectures because it requires data buffering only between the two $R4^3$ stages instead of the buffering required at all the 6 stages of the unfolded FFT using R4. The existing cascade FFT architectures require 6 $R2^2$ (He & Torkelson, 1996, 1998) and although they use 1/4 of the memory of the proposed $R4^3$ architecture for 4096 points, they are not scalable and they must be designed for specific performance due to their organization involving “closed computing loops”.
The achieved performance establishes the proposed architecture as a real-time high performance FFT realization, the importance of which can be further pointed out by

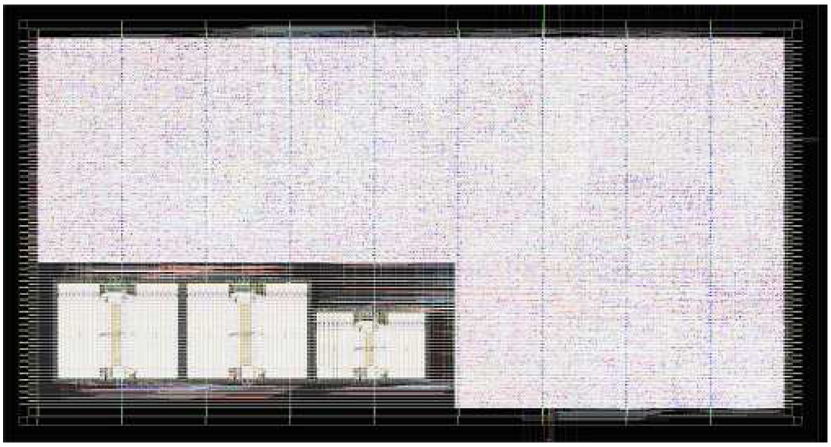


Fig. 6. VLSI implementation layout of the Radix − 4^3 engine

comparing its characteristics to other relevant results. Tables 3 and 4 present a comparison of the proposed FFT architecture with related results. We distinguish related published results into two categories.

	FFT 4096 Ramsx16	FFT 4096 Ramsx64
Clock (period)	1.6 ns	1.2 ns
Fmax in MHz	386.8	604.5
Std Cells (RAMs)	100347(16)	95897(64)
Std Cells (rows)	805	1361
Chip Size (um sq)	5.34E+06	1.42E+07
Util (%)	80.30%	84.20%
X(um)	1681	2630
Y(um)	3165	5129
Power (mW)	722.8	4414.1

Table 2. 4K FFT VLSI Implementation Routing Results

In the first category we compare the hitherto published architectures executing FFT algorithms up to 128 points to the features of the R−4^3 playing the role of a complete 64 complex point FFT architecture. This comparison is shown in Table 3. The second category includes the architectures solving FFTs of size 1024 to 4096 points presented in Table 4. The comparison includes FFT size, word length, algorithm, FFT architecture, technology process, voltage, area, power, maximum operating frequency and sustained throughput in both MSamples/s and Gbits/s. Since the FFT designs vary with respect to FFT size, algorithm and architecture we have also included the Normalized Area (Bidet et al., 1995), in order to evaluate the silicon cost. Moreover, we compare the efficiency (performance/cost) of the proposed design to the related results by using the fraction Sustained Throughput/Normalized Area.

The proposed FFT architecture achieves the highest sustained throughput compared to all the other designs. Furthermore, the efficiency expressed as the fraction Sustained Throughput/Normalized Area of the proposed design is the highest considering both small input size (Table 3) and large input size FFT architectures (Table 4). Note that, in both categories the architectures R − 4^3 and the proposed 4K occupy more area than their competitors respectively. This is a penalty though in achieving the highest throughput possible.

Also note that (Lin et al., 2005) performs transformations of only 128 points, and there is no provision taken so that it will constitute a core for scalable architectures with respect to FFT size. The 64-point Fourier transform chip, presented in (Maharatna et al, 2004) operates at 20 MHz with latency 3.85 us, comparing to the $R4^3$ processor performing a 64 complex point FFT while operating at a 200 MHz clock frequency with latency 0.32 us.

The architecture described in (Lenart & Owall, 2003) is a 2K complex point FFT processor which achieves maximum operating frequency of 76MHz and sustains a throughput of 2048 points/26us. The design presented in (Cortes et al., 2006) implements a 2K/4K/8K multimode FFT and achieves 9 MHz clock frequency, at a computation time of up to 450us.

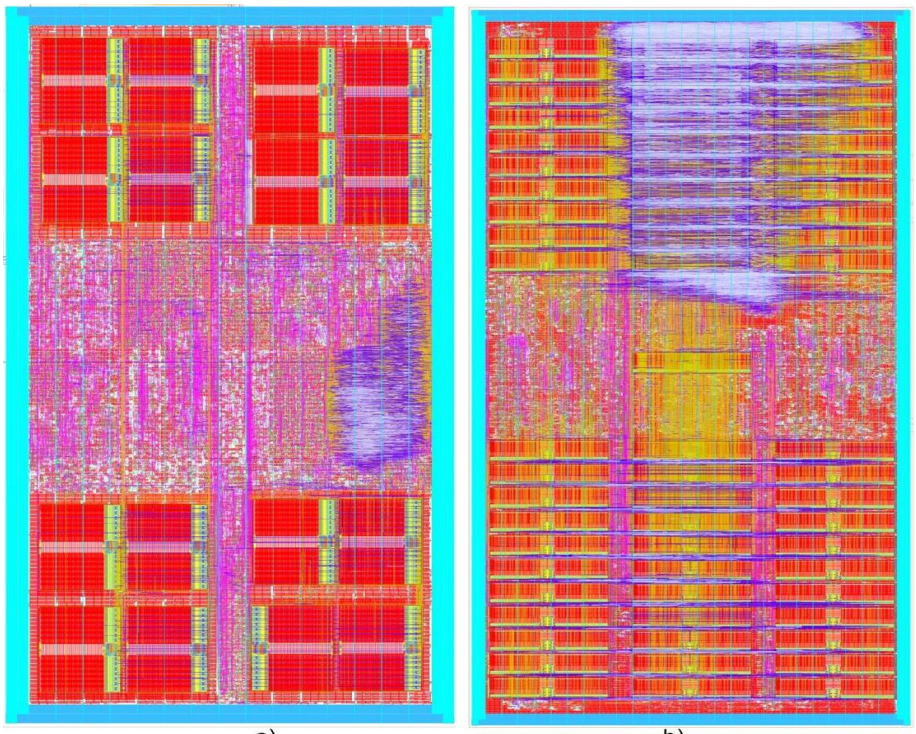


Fig. 7. VLSI implementation layout of the 4K FFT processor: a) 16 × 4096 Configuration, b) 64 × 4096 Configuration

Characteristics	(Lin et al., 2005)	(Maharatna et al., 2004)	Proposed Design (R−4^3)
FFT size	128p	64p	64p
Word Length, bit	10	16	14
Algorithm	R-2, R-8	R-2	R-4^3
FFT Architecture	MRMDF	Unrolled	Unrolled
Process (um)	0.18	0.25	0.13
Voltage (V)	3.3	1.8	0.9
Area (mm^2)	3.52	6.8	5.09
Normalized Area (mm^2)	3.52	3.5	9.7
Power (mW)	77.6	41	78.2
Fmax (MHz)	110	20	604.5
Sustained Throughput in MSamples/s	480	16.6	1052
Sustained Throughput in Gb/s	9.6	0.53	29.5
$\frac{\text{Throughput(Gb/s)}}{\text{Norm.Area}}$	2.72	0.15	3.04

Table 3. Comparison Table with FFT designs from 64-128 points

Characteristics	(Lenart & Owall, 2003)	(Cortes et al., 2006)	(Swartzlander, 2007)	Proposed Design
FFT size	2K	2K/4K/8K	4K	4K
Word Length, bit	10	16	16	14
Algorithm	$R-2^2$	$R-2^2$	$R-2$	$R-4^3$
FFT Architecture	SDF	SDF	Split Systolic	Unrolled
Process (um)	0.35	0.35	0.25	0.13
Voltage (V)	N/A	3.3	N/A	0.9
Area (mm^2)	6	18.7	N/A	13.48
Normalized Area (mm^2)	1.58	4.9	N/A	25.84
Power (mW)	N/A	114.65	260	4414.1
Fmax (MHz)	76	9.1	100	604.5
Sustained Throughput in MSamples/s	75.8	18.2	200	1052
Sustained Throughput in Gb/s	1.5	0.582	6.4	29.5
$\frac{\text{Throughput(Gb/s)}}{\text{Norm.Area}}$	0.94	0.11	N/A	1.14

Table 4. Comparison table with FFT designs from 1K-4K

Finally, a single ASIC chip, systolic FFT processor, developed by the Mayo Foundation computes 4096-point FFTs sustaining a throughput of 200 Ms/s (Swartzlander, 2007).
Considering FPGA implementations, the corresponding XILINX designs (www.xilinx.com) achieve equal maximum operating frequency of 200MHz, but occupy considerably larger chip area than the $R4^3$ approach. Also note that, ALTERA designs (www.altera.com) utilize FFT cores with FFT length varying from 64 points up to 4K points. They demonstrate a maximum operating frequency of 300 MHz. Among the ALTERA's FFT designs we compare the 64 point FFT at 300 MHz to the $R4^3$ performance, which realized on the same ALTERA FPGA (ALTERA STRATIX II EP2S30F484C3) achieves operating frequency of 350 MHz.

3. 16K, 64K and 256K complex point FFT Architectures

The high utility prospects of the efficient 4K point FFT design as presented above becomes tangible through the utilization of this architecture to develop larger size FFT architectures to compute 16K, 64K and 256K point transforms. This section describes the exploitation of the 4K point FFT as a "radix-4096" processor resulting in architectures meeting different requirements with respect to parallelism, silicon area and throughput. These architectures can be tailored to a broad area of applications.

3.1 Deriving the 16K point Architecture

The DFT equation for a 16K point FFT takes the form

$$X[k] = \sum_{n=0}^{16383} x[n] W_{16384}^{nk} = \sum_{n=0}^{4\cdot4096-1} x[n] W_{4\cdot4096}^{nk} \tag{11}$$

Setting $n = n_1 + \frac{N}{4096} n_2$ and $k = k_1 4096 + k_2$:

$$k \cdot n = \left(n_1 + \frac{N}{4096} n_2 \right)(4096 k_1 + k_2) \Rightarrow k \cdot n = n_1 k_1 + n_1 k_2 + N n_2 k_1 + \frac{N}{4096} k_2 \Rightarrow$$

$$W_{4\cdot4096}^{kn} = W_4^{n_1 k_1} W_{4\cdot4096}^{n_1 k_2} W_{4096}^{n_2 k_2}$$

Therefore, the transform becomes

$$X[k] = \sum_{n_1=0}^{3} W_4^{n_1 k_1} \left(\sum_{n_2=0}^{4095} x[n] W_{4096}^{n_2 k_2} \right) W_{4\cdot4096}^{n_1 k_2} \tag{12}$$

According to equation 12, the 4K points FFT can be extended to 16K points. This is accomplished by first performing four 4K point FFT transforms. Next, the data is multiplied by the twiddle factors that correspond to a 16K points FFT and finally, a radix-4 stage completes the 16K point FFT computation.

The architecture is presented in figure 8. There are four 4K FFT blocks operating in parallel. The architecture has a four complex point input per cycle. The 16K FFT architecture was implemented in a high performance, 0.13um, 1Poly-8Copper layer standard cell technology from TSMC. A flat back-end flow was used in which the design was first synthesized to gates using Synopsys Design Compiler and optimized for a frequency of 300 MHz. The uniquified netlists was then read into Cadence SoC Encounter where floorplanning, power planning, clock-tree-synthesis, placement, routing and IPO routing took place. Finally, the design was brought back into the top-level for final top-level placement, routing and timing analysis. Table 5 shows the Implementation Routing Results and figure 8 depicts the VLSI Cells for the 16K FFT. The throughput is 1.4 Gs/sec (39.2 Gbits/sec).

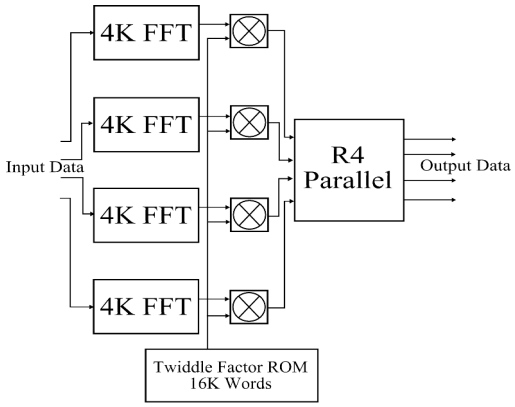


Fig. 8. Block diagram of the 16K FFT - Parallel/Parallel Architecture

Parameter	FFT 16K
Core Size	1.6561e + 07um²
Std Cell Rows	1102
Number of Cells	885456
Number of RAMs	64
Statistical Power	3.45 W
Fmax	352 MHz

Table 5. VLSI Implementation Routing Results of the 16K FFT architecture

The 16K architecture implemented on the Xilinx Virtex 5 (-2) achieves operating frequency of 250MHz, occupies 12264 slices and sustains a throughput of 1Gs/s (28 Gbits/sec).

3.2 64K and 256K complex points FFT Architectures

The DFT equation for a 64K point FFT takes the form

$$X[k] = \sum_{n=0}^{65535} x[n] W_{65536}^{nk} = \sum_{n=0}^{4\cdot 16384-1} x[n] W_{4\cdot 16384}^{nk} \tag{13}$$

Setting $n = n_1 + \frac{N}{16384} n_2$ and $k = k_1 16384 + k_2$:

$$k \cdot n = 16384 n_1 k_1 + n_1 k_2 + N n_2 k_1 + \frac{N}{16384} k_2 \Rightarrow W_{4\cdot 16384}^{kn} = W_4^{n_1 k_1} W_{4\cdot 16384}^{n_1 k_2} W_{16384}^{n_2 k_2}$$

Therefore, the transform becomes

$$X[k] = \sum_{n_1=0}^{3} W_4^{n_1 k_1} \left(\sum_{n_2=0}^{16383} x[n] W_{16384}^{n_2 k_2} \right) W_{4\cdot 16384}^{n_1 k_2} \tag{14}$$

According to equation 14, the 16K points FFT can be extended to 64K points. This is accomplished by first performing a 16K point FFT transform. Next, the data is multiplied by the twiddle factors that correspond to a 64K points FFT and finally, a R4 stage completes the 64K point FFT computation, as shown in figure 10. The 64K FFT has been VLSI (and FPGA) implemented by using the 16K parallel/parallel computation with a R4 stage with four parallel inputs and outputs. The architecture has a post routing frequency of 256 Mhz with a throughput of 1 Gs/sec (28 Gbits/sec). Figure 11 depicts the VLSI Cell for the 64K FFT design. The 64K has been implemented on the Xilinx Virtex 5 (-2) achieving an operating frequency of 125MHz and using 13461 slices. The architecture has a four data parallel input and output and sustains a throughput of 500Ms/s (14 Gbits/sec).

3.3 256K complex point FFT Architecture

Figure 12 depicts the 256K FFT architecture, which is a straightforward application of the R4³ algorithm. Three consecutive R4³ engines are used to accomplish the task.

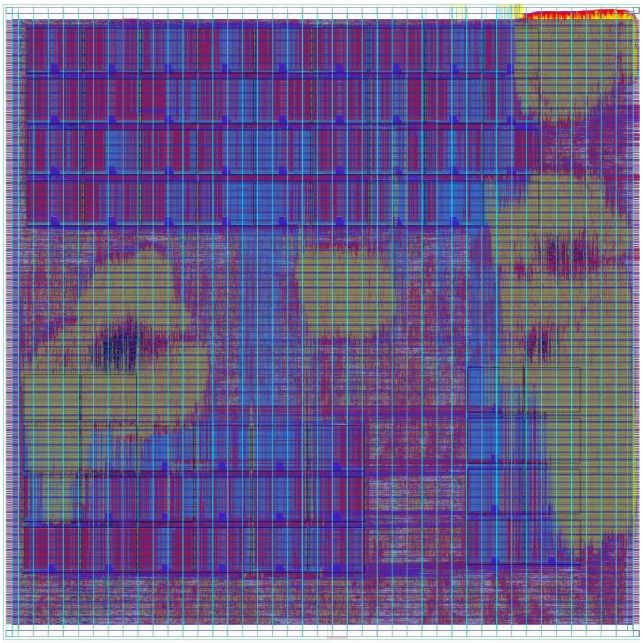


Fig. 9. VLSI implementation layout of the 16K FFT Parallel/Parallel architecture

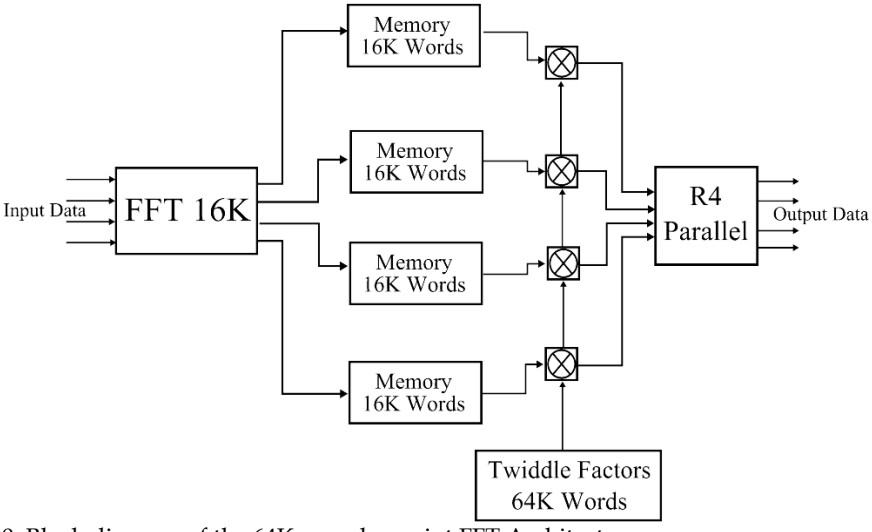


Fig. 10. Block diagram of the 64K complex point FFT Architecture

Parameter	FFT 64K	FFT 256K
Core Size	3.4894e + 08um²	2.7647e + 08um2
Std Cell Rows	1600	4500
Number of Cells	896148	735945
Number of RAMs	192	384
Statistical Power	9.8 W	35.75 W
Fmax	256.5 MHz	188 MHz

Table 6. VLSI Implementation Routing Results of the 64K and 256K FFT designs

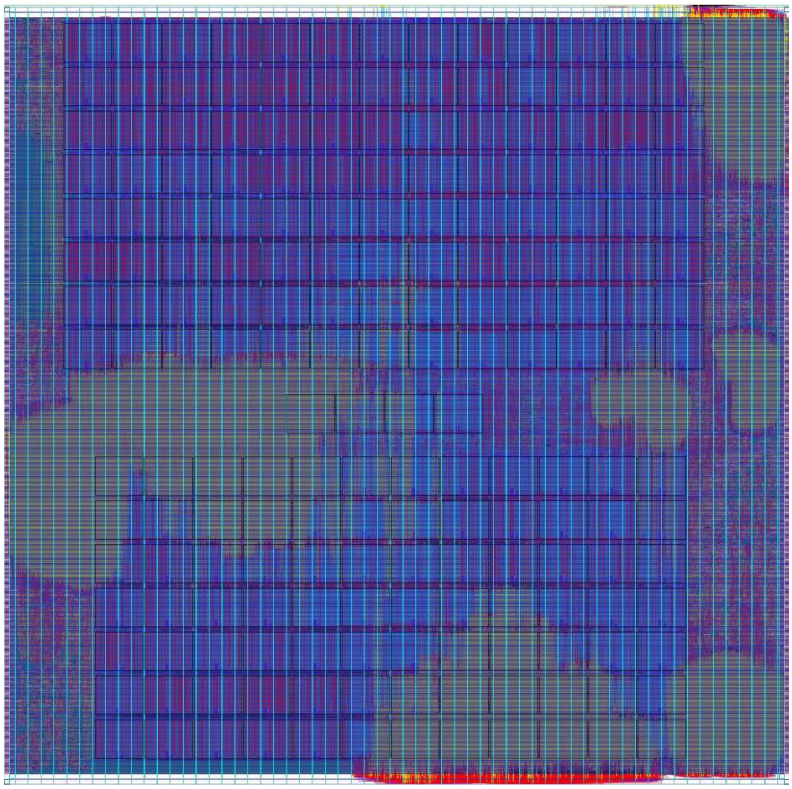


Fig. 11. VLSI implementation layout of the 64K FFT

In the case of the very large 256K FFT, tool capacity mandated the use of a hierarchical flow. Following the same front-end synthesis process (again optimized for 300 MHz), the optimized netlist was read into Cadence SoC encounter where partitioning was first performed. This process created six instances of the 256K memory block (for a total of 1.5 MB of on-chip SRAM) which were individually placed and routed. The same process was performed for the R4³ engines and the twiddle ROM block. Note that to complete the entire FFT computation it requires a 3-frame latency (786792 cycles) and the computation latency within the three R4³ (360 cycles), which is 4.1 msec. The 256K FFT architecture has a post routing frequency of 188 Mhz. Finally, the design was brought back into the top-level for

final top-level placement, routing and timing analysis. Figure 13 depicts the VLSI Cell for the 256K FFT design and Table 6 shows the VLSI Implementation Routing Results of the 64K and 256K FFT designs.

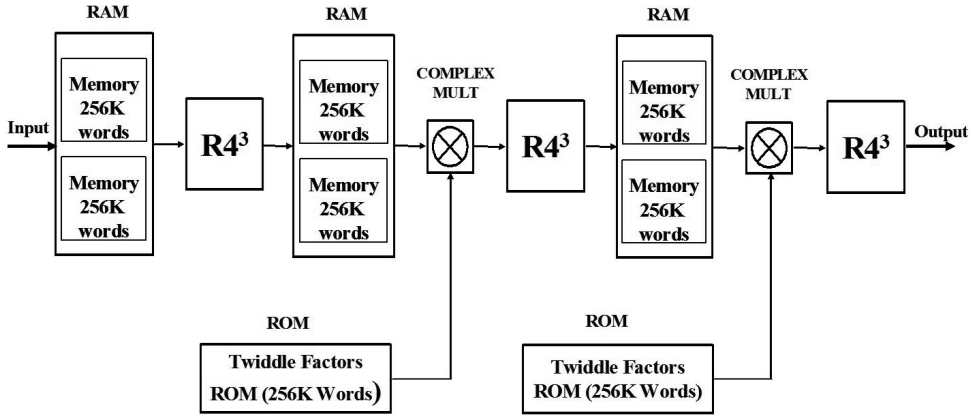


Fig. 12. Block diagram of the 256K FFT Architecture

4. Parallel Accessing in FFT Architectures

This section presents a technique, which allows the parallelization of the memory accesses in hardware implementations of the FFT algorithm by enabling a processor during a FFT stage to access in parallel b memory banks without conflicts. Consequently the processor performs a radix-b butterfly by loading the b-tuple data from b memory banks in parallel, then by operating on the b data and finally by storing the resulting b-tuple in b memory banks in parallel. Hence, the speedup and the throughput increase by b.

4.1 Problem Definition and Related Work

Let $N = b^p$ denote the number of points of the Fast Fourier Transform, where b is an arbitrary base such that $b \geq 2$ and $p \geq 1$. Using b as a base, the input address 0,…,N-1 (or index) of each element is represented as $[a_{p-1}...a_0]$, where $0 \leq a_i \leq b-1$. We will denote each element only by its address $[a_{p-1},...,a_0]$, which represents the element in the input stream with index $a_{p-1}b^{p-1} + ... + a_1 b + a_0$. This notation holds for each of the p stages of the FFT.

We will use $P_{b[n \to m]}$ to denote a processor performing all radix-b computations, which can also read (load) *n* data in parallel and write (store) *m* data in parallel. We assume a generic architecture using ψ, $1 \leq \psi \leq \log_b N$, $P_{b[b \to b]}$ processors and $2 \cdot b \cdot (\psi + 1)$ memory banks, so that each bank can store N/b elements. Each processor realizes N/b consecutive stages of the FFT computation.

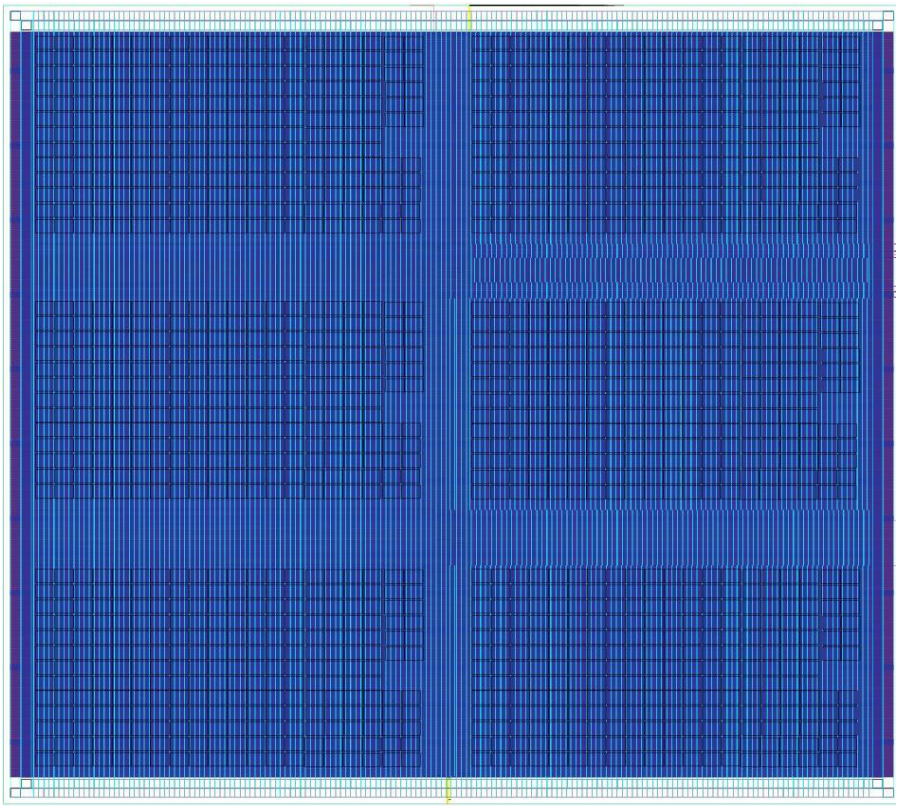


Fig. 13. VLSI implementation layout of the 256K FFT

The memory banks are arranged according to the scheme described in figure 14. Since, we are using a radix-b processor, the DFT decomposition consists of $\log_b N$ stages. If we assume $\psi = \log_b N$ then we consider a $P_{b[b\rightarrow b]}$ processor realizing each stage of the algorithm. The stages are indexed from 0 to $\log_b N - 1$. Between two (2) consecutive stages i and i+1 there is a memory of size $2 \cdot N$ divided into two (2) sets of size N each. We will denote the memory between stage i and stage i+1 as memory with index i. The memories among the FFT stages are indexed as 0 to $\log_b N - 2$. There are memories indexed as -1 and $\log_b N - 1$ serving as input and output memories respectively. The memory of each set i is divided into b memory banks with indices 0 to b-1.

At each clock cycle, the $P_{b[b\rightarrow b]}$ processor of stage i loads from its ''read'' memory (memory i-1), b elements that form a transform b-tuple. These are the elements whose address is of the form $[a_{p-1},...,a_{i+1}la_{i-1},...,a_0]$, where the indices $[a_0,...,a_{i-1},a_{i+1},...,a_{p-1}]$ are the same for all elements in the b-tuple and l ranges from 0 to b-1. These elements will be loaded in parallel if they are stored in distinct memory banks. During the next $((i + 1)^{th})$ step of the algorithm, these b elements have the same a_{i+1} address digit and hence they belong to

different transformation b-tuples. These elements must be stored in the processor's write memory so that they can be read in parallel from the processor at stage i + 1.

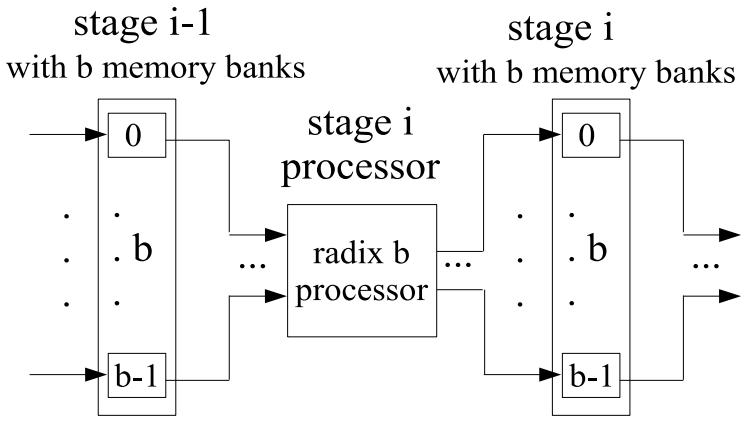


Fig.14. Generic arrangement of a $P_{b[b\rightarrow b]}$ processor and its write memories

The straightforward approach is to distribute the N elements to the b memory banks using their i^{th} address digit. Then the processor at stage i can load them in parallel. However, if we try to store the same b elements of each b-tuple to the memory bank at stage (i+1) according to their $(i+1)^{th}$ digit of their address, we will notice that all the elements in each b-tuple must be stored in the same memory bank. Since we cannot store more than one element in a memory bank at each clock cycle this situation constitutes a "conflict".

To illustrate the situation where a "conflict" occurs, consider a 16 point transform that is based on $P_{4[4\rightarrow 4]}$ processors. The address of each element in the input stream is $[a_1 a_0]$, where $a_i = 0,1,2,3$. Figure 15 shows the first stage of the example. The first 4-tuple to be transformed consists of the elements [(0, 0) (0, 1) (0, 2) (0, 3)]. Note that, these elements can be read in parallel from the set of memory banks of stage i − 1, but must be stored in the same memory bank of memory i.

The first technique for parallelizing the memory accesses in FFT architectures (Johnson, 1992) describes a hardware architecture designed for in-place computation of the FFT algorithm. This technique uses r (r being the radix) banks and permutes the output data of each processor, to be written in the same memory locations within each bank as the input data have been read. (Ma, 1999) describes a technique for radix-2 based FFT by using queues at the input of each memory to rearrange the data before they are stored. Johnson's technique can be extended to the more general case of a pipelined FFT implementation at the cost of complex addressing hardware implementation. (Thomas & Yelick, 1999) present a technique that is used in vector processors. This technique permutes the input data prior to the radix calculations in order to maximize the efficiency of the algorithm. These permutations are performed within the vector registers (In-register transpose). This functionality has been implemented by extending the instruction set of the vector processor with two instructions that shift the data within a register, so that each permutation is performed by using a register to register copy and a shift instruction.

All the FFT techniques (Johnson, 1992; Ma, 1999), with b parallel memory accesses, involve at each stage b banks for reading and b banks for writing the N FFT data with bank size N/b. The in-place case uses the same b banks (each bank of size N/b) for both read and write. To compare the hardware complexity of the previously published results, we first consider designs that are based on queues, such as the one described in (Ma, 1999).

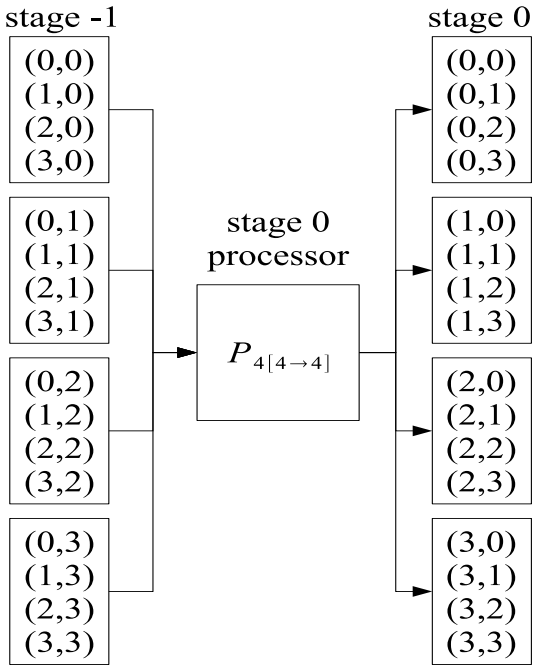


Fig. 15. First stage of a 16-point transform using $P_{4[4\rightarrow 4]}$ processors

Such designs require $O(b^2)2w$ -bits wide registers, where w is the word length, arranged in b queues with each queue having $b - 1$ registers. b is the processor radix, or in the general case, the number of parallel read/write accesses. Each set of $b-1$ registers is arranged so that $b-1$ elements are loaded in parallel while one element is written into the memory bank. The other $b-1$ elements are shifted -one element per memory access cycle- into each memory bank. The technique in (Johnson, 1992) uses fixed hardware to implement first a permutation on the input data and then this specific hardware can apply a permutation at each FFT stage. More specifically, (Johnson, 1992)'s approach gives only a subset of the possible solutions. This subset of solutions considers only in-place implementations ($\psi = 1$ and memory size N). The circuit in (Johnson, 1992) is optimized for implementing these solutions with respect to the VLSI area. The address generation is based on a circuit involving a tree of modulo-b adders with $O(b^2)$ exclusive-OR gates.

In this section we describe a technique (Reisis & Vlassopoulos, 2006) that can be used both for pipelined architectures and in place implementations (memory size equals to N). We follow a different approach than (Johnson,1992) leading to a different proof and providing a

general solution, which includes that in (Johnson,1992). Moreover, the presented technique results in a simple and improved hardware implementation compared to (Johnson,1992). The presented solution shows improved performance with respect to the latency and the hardware cost compared to other solutions (Ma, 1999; Suter & Stevens,1998; He & Torkelson; Thomas & Yelick, 1999) while it provides the same throughput as these. Our technique is based on memory address permutations and can be realized using look up tables with each stage table occupying $O(b^2)$ bits. The presented approach results in a set of permutations, which can be applied in the design of pipeline FFT architectures with parallel memory accesses per stage. A subset of these permutations accommodates the in-place implementations.

4.2 Parallelizing the Memory Accesses

The following Lemmas 1 and 2 prove the existence of the required permutations and Theorem 1 proves that these permutations result in a correct FFT algorithm.

Lemma 1: Assume that the $N = b^p$ elements with indices $[a_{p-1}...a_0]$, $0 \le a_i \le b-1$, are distributed in b sets S_l, $l = 0, \ldots, b-1$, such that: $S_l = \{[a_{p-1}...a_{i+1} l a_{i-1}...a_0]\}$

Let $f_j : \{0, \ldots, b-1\} \rightarrow \{0, \ldots, b-1\}$, where $0 \le j \le b-1$, be a set of functions such that

- $\exists f_j^{-1} \Rightarrow f$ is 1-1 and onto, and
- $\forall m, \forall n$, where $m, n \in \{0,..., b-1\}$,

  If $m \neq n$, then $\forall x$, $f_m(x) \neq f_n(x)$

The sets $S'_{l'}$, where $l' = f_{a_{i+1}}(l)$ are defined as follows $S'_{l'} = \{[a_{p-1}...a_{i+2} f_{a_{i+1}}(l)\ l\ a_{i-1}...a_0]\}$

Where l is fixed and $a_{i+1} = 0,..., b-1$

Under the above conditions, the following hold:

1) Each $S'_{l'}$ contains at most N/b items

2) Every two elements of two sets S_{l_1}, S_{l_2} whose indices differ only in the i^{th} digit, i.e. q and r will be distributed in different S′ sets, namely the $S'_{q'}$ and $S'_{r'}$.

Proof of Lemma 1: Each set $S'_{l'}$ contains the data with indices $[a_{p-1}...a_{i+2} f_{(a_{i+1})}(l)\ l\ a_{i-1}...a_0]$, where $a_{i+1} = 0,..., b-1$ and l is fixed. To prove (1) assume that there exists a set that contains more than N/b elements. Then, there is at least one element with index $[a_{p-1}..a_{i+2} f_{(a_{i+1})}(l')\ l''\ a_{i-1}..a_0]$ where l′ is different than l″. Since such an element does not belong to the set $S'_{l'}$, we conclude that each set cannot contain more than N/b elements.

Similarly, to prove (2), assume that two elements of the sets S_q and S_r, respectively, whose indices differ in the i^{th} digit are distributed in the same set, $S'_{s'}$. Then, for these elements, $[a_{p-1}...a_{i+2} f_{a_{i+1}}(q)\ l\ a_{i-1}...a_0] = [a_{p-1}...a_{i+2} f_{a_{i+1}}(r)\ l\ a_{i-1}...a_0] \Rightarrow f_{a_{i+1}}(q) = f_{a_{i+1}}(r)$. Since the f_j are invertible, $f_{a_{1+1}}^{-1}(f_{a_{i+1}}(q)) = f_{a_{1+1}}^{-1}(f_{a_{i+1}}(r)) \Rightarrow q = r$, which contradicts our initial assumption that $l_1 \neq l_2$. The following Lemma shows that these functions exist.

Lemma 2: There exists a set of functions $f_j : \{0,..., b-1\} \rightarrow \{0,..., b-1\}$, $j = 0,..., b-1$ and $b \ge 2$ such that the conditions of Lemma 1 are satisfied.

Proof of Lemma 2: Let M = {0, . . . , b − 1}. Further, let C(M) denote the set of cyclic permutations of length b on the set M. These permutations are of the form

$$\begin{pmatrix} 0 & 1 & \dots & (b-1) \\ (p) \bmod b & (1+p) \bmod b & \dots & (b-1+p) \bmod b \end{pmatrix}$$

for p = 0, . . . , b − 1. Now, these permutations are invertible and there are exactly b such permutations. To prove that

$$f_i(x) \neq f_j(x) \qquad (15)$$

for $i \neq j$ and i, j = 0, . . . , b − 1 it suffices to show that $(x+i) \bmod b \neq (x+j) \bmod b$, which is true since i and j could only be in the same equivalence class, if i = k · j, where k is an integer, but this can be true only for k = 1, since i, j ≤ b − 1. Therefore equation 15 holds.

Theorem 1: Let $N = b^p$. Assume that we have a $P_{b[b \to b]}$ processor. Then, a radix-b based FFT having $\log_b N$ stages can use b memory banks at each stage, such that all the read and write operations are performed in parallel.

Proof: Let $[a_{p-1} \dots a_0]$, $0 \le a_i \le b-1$, be the address of each element according to its initial position within the input data set of the algorithm. The i^{th} stage of the algorithm has arranged the N elements in b memory banks according to their i^{th} digit of their address:

$$[a_{p-1} \dots a_{i+1} a_{i-1} \dots a_0][a_i] \qquad (16)$$

where a_i is the index of the memory bank.
To write the outputs of a radix-b processor into b distinct memory banks in parallel, we use the results of Lemma 1. First, we assign each memory bank to a set, S_l, so that each transform b-tuple consists of elements that differ in the i^{th} element of their address. Next, we select a set of b functions $f_0, f_1, \dots, f_{(b-1)}$ using Lemma 2.
Since the requirements of Lemma 1 hold, we can distribute the elements according to the equation:

$$[a_{p-1} \dots a_{i+2} a_i \dots a_0][f_{a_{i+1}}(a_i)] \qquad (17)$$

These elements will be distributed in distinct memory banks and can be stored in parallel.
Executing the following stage, we will use the inverse permutation to read each b-tuple in parallel: During the $(i+1)^{th}$ stage of the FFT algorithm, the $(i+1)^{th}$ processor should read the data whose address is

$$[a_{p-1} \dots a_{i+2} a_i \dots a_0][a_{i+1}] \qquad (18)$$

where $a_{i+1} = 0, \dots, b-1$ and a_j = const, for $j \neq i+1$. The inverse permutation is given by

$$[a_{p-1} \dots a_{i+2} a_i \dots a_0][f^{-1}_{a_{i+1}}(a_i)] \qquad (19)$$

where a_i = const and $a_{i+1} = 0, \dots, b-1$, since this permutation yields a b-tuple whose elements differ only in the $(i+1)^{th}$ digit. To prove that these elements are stored in b distinct memory banks assume that this is not the case, i.e. there exist two distinct elements, $q_1 = [a_{p-1} \dots a_{i+2} a_i \dots a_0][f^{-1}_{a_{i+1}}(a_i)]$ and $q_2 = [a'_{p-1} \dots a'_{i+2}\, a'_i \dots a'_0][f^{-1}_{a'_{i+1}}(a'_i)]$ that are

in the same b-tuple and at the same time are stored in the same memory bank. Since q_1 and q_2 are on the same b-tuple, therefore $a_j = a'_j$ for $j \neq i+1$. Further, $f^{-1}_{a'_{i+1}}(a_i) = f^{-1}_{a_{i+1}}(a_i)$, which is true only when $a'_{i+1} = a'_{i+1}$. Therefore, the elements q_1 and q_2 coincide.

To conclude the proof we show that the forward and inverse permutations at all stages result in a correct FFT algorithm. We proceed by induction on the number of stages. During the first stage of the FFT the data are written in the input memory banks (memory −1) according to $[a_{p-1}...a_1][a_0]$. The outputs of the butterfly are written to the first intermediate memory (memory 0) using the permutation $[a_{p-1}...a_2a_0][f_{a_1}(a_0)], a_0 = 0,...,b-1$. We assume that the functions f_j are identical for all stages, although this may not be the case. The only actual restriction is to use a permutation P_i for writing the data at stage i and its inverse permutation P_i^{-1} for retrieving the data at stage i+1. Applying the inverse permutation we obtain $[a_{p-1}...a_2a_0][f^{-1}{}_{a_1}(a_0)]$, with a_0 = constant and $a_1 = 0, \ldots, b-1$. The set $\{f^{-1}{}_{a_1}(a_0)\}$ consists of b distinct elements, such that their address differs only in the digit a_1 and therefore, they constitute a valid transform b-tuple. Further, assume that in the i^{th} stage the elements are stored according to equation 17. Using the same reasoning as above, we can see that the inverse transform yields a valid b-tuple and the proof is complete. Note that the write operation of the elements in the first set of banks (memory 0) does not require any permutation on the input data set. The elements are written to the bank corresponding to their 0^{th} (zero) address digit, in radix b notation. Similarly, during the final step of the algorithm, the data can be written in the same addresses and banks as those that have been read from, since no computation follows

The current section has shown the technique and the properties of the permutations required to the parallel access of each b-tuple at each FFT stage. An engineer can realize the technique by choosing among the straightforward solutions, e.g. the cyclic permutations. A ninteresting research topic is to identify those permutations, which can be realized with minimal interconnection and address generation circuits and thus lower the VLSI cost.

5. Conclusion

The present chapter has shown a technique to design FFT architectures for real-time applications, which involve input with large number of complex points. The technique bases on combining three consecutive R4 stages to realize a R64 computation. The resulting $R4^3$ as well as the systolic architectures, which utilize a $R4^3$ as a stage for executing 4K, 16K, 64K and 256K point FFTs, have been shown to provide higher throughput compared to hitherto published architectures solving the corresponding transformations. Moreover, the $R4^3$ and 4K FFT architectures achieve the highest ratio of throughput to normalized area. Furthermore, the chapter has proven a technique for parallelizing the memory access for each butterfly radix-b computation so that the throughput can be improved further by a factor of b.

6. References

A. Cortes, I. Velez, I. Zalbide, A. Irizar, J.F. Sevillano "An FFT Core for DVB-T/DVB-H Receivers ICECS'06, page(s):102-105, December 2006.

A. Oppenheim, R. Schafer Digital Signal Processing. Prentice Hall, 1975.

B. G. Jo and M. H. Sunwoo, "New Continuous-Flow Mixed-Radix (CFMR) FFT Processor Using Novel In-Place Strategy," IEEE Trans. Circuits Syst. I, vol.52, no.5, May 2005.

B. Suter and K. S. Stevens "A Low Power, High Performance approach for Time-Frequency / Time-Scale Computations," Proceedings SPIE98 Conference on Advanced Signal Processing Algorithms, Architectures and Implementations VIII. Vol. 3461, pp. 86-90, July 1998.

C. D. Thompson, "Fourier Transforms in VLSI", IEEE Transactions on Computers, Vol. 32, 1047 - 1057, 1983

D. Harper, "Block , Multistride Vector, and FFT Accesses in Parallel Memory Systems", IEEE Trans. on Parallel and Distributed Systems, Vol. 2, No. 1, pp. 43 - 51, January 1991

D. Reisis, N.Vlassopoulos, "Address Generation Techniques for Conflict Free Parallel Memory Accessing in FFT Architectures" ICECS, pp.1188-1191, December 2006.

E. Bidet, D. Castelain, C. Joanblanq and P. Stenn "A fast single-chip implementation of 8192 complex point FFT" IEEE Journ. of SSC, 30(3):300-305, Mar. 1995.

E. H. Wold and A. M. Despain "Pipeline and Parallel FFT Processors for VLSI Implementations," IEEE Transactions on Computers, vol. C-33, 1984.

Earl E. Swartzlander, Jr "Systolic FFT Processors: A Personal Perspective Journal of VLSI Signal Processing, June 2007.

I. S. Uzun, A. Amira and A. Bouridane, "FPGA implementations of Fast Fourier Transforms for real-time signal and image processing," IEEE Vision, Image and Signal Processing, 2005.

J. Lee, J. Lee, M. H. Sunwoo, S. Moh and S. Oh "A DSP Architecture for High-Speed FFT in OFDM Systems," ETRI Journal, 2002.

J. Takala and K. Punkka Scalable FFT Processors and Pipelined Butterfly Units Journal of VLSI Signal Processing 43, 113-123, 2006.

J. Y. OH and M. S. Lim, "New Radix-2 to the 4th Power Pipeline FFT Processor," IEICE Trans. Electron., VOL. E88-C, NO. 8, August 2005.

J.W. Cooley and J.W. Tukey, "An algorithm for the machine computation of complex Fourier series", Mathematics of Computation, 1965

K. Babionitakis, K. Manolopoulos, K. Nakos, N. Vlassopoulos, D. Reisis, V. Chouliaras, "A High Performance VLSI FFT Architecture", 13th IEEE International Conference on Electronics, Circuits and Systems, pp. 810-813, December 2006

K. Maharatna, E. Grass, and U. Jagdhold, "A 64-Point Fourier Transform Chip for High-Speed Wireless LAN Applications Using OFDM," IEEE Journal of Solid State Circuits, VOL. 39, NO. 3, March 2004.

K. Manolopoulos, K. Nakos, D. Reisis, N.Vlassopoulos, V.A. Chouliaras "High Performance 16K, 64K, 256K complex points VLSI Systolic FFT Architecture" ICECS, pp. 146-149, December 2007.

L. G. Johnson, "Conflict Free Memory Addressing for Dedicated FFT Hardware", IEEE Transactions on Circuits and Systems - II: Analog and Digital Signal Processing, Vol. 39, No. 5, May 1992

L. R. Rabiner and B. Gold ”Theory and Application of Digital Signal Processing,” Prentice-Hall.

L. Yang, K. Zhang, H. Liu, J. Huang and S. Huang ”An Efficient Locally Pipelined FFT Processor,” IEEE Trans. Circuits Syst. II, vol. 53, no. 7, July 2006.

N. Hu, O. Ersoy, “Fast Computation of Real Discrete Fourier Transform for Any Number of Data Points”, IEEE Transactions on Circuits and Systems, Vol. 38, No. 11, pp. 1280 - 1292, November 1991

O.K . Ersoy, Fourier-Related Transforms, Fast Algorithms and Applications. Englewood Cliffs, NJ:Prentice Hall, 1997.

R. Thomas and K. Yelick, “Efficient FFTs on IRAM”, Proceedings of the 1st Workshop on Media Processors and DSPs, 1999

S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, ”A New Radix-2/8 FFT Algorithm for Length−q × 2m DFTs,” IEEE Trans. Circuits Syst. I, vol. 51, no. 9, September 2004.

S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, ”New Radix-(2×2×2)/(4×4×4) and Radix-(2×2×2)/(8×8×8) DIF FFT Algorithms for 3-D DFT,” IEEE Trans. Circuits Syst. I, vol. 53, no. 2, February 2006.

S. Choi, G. Govindu, J. W. Jang, V. K. Prasanna ”Energy-Efficient and Parameterized Designs of Fast Fourier Transforms on FPGAs,” The 28th International Conference on Acoustics, Speech, and Signal Processing (ICASSP), April 2003.

S. He and M. Torkelson ”A New Approach to Pipeline FFT Processor,” Proceedings of the IPPS, 1996.

S. He and M. Torkelson ”Design and Implementation of a 1024-point Pipeline FFT Processor,” IEEE 1998 Custom Integrated Circuits.

S.S. Wang and C.S. Li An Area-Efficient Design of Variable-Length Fast Fourier Transform Processor Journal of VLSI Signal Processing, March 2007.

T. Lenart and V. Owall, ”A 2048 Complex Point FFT Processor Using a Novel Data Scaling Approach,” IEEE ISCAS 2003.

W. H. Chang, T. Nguyen, ”An OFDM-Specified Lossless FFT Architecture,” IEEE Trans. Circuits Syst. I, vol. 53, no. 6, June 2006.

www.altera.com

www.xilinx.com

Y. Ma, “An Effective Memory Addressing Scheme for FFT Processors”, IEEE Transactions on Signal Processing, Vol. 47, No. 3, 907 - 911, May 1999

Y. Ma, L.Wanhammar, “A Hardware Efficient Control ofMemory Addressing for High-Performance FFT Processors”, IEEE Transactions on Signal Processing, Vol. 48, No. 3, 917 - 921, March 2000.

Y.N. Lin, H.Y. Liu, and C.Y.Lee “A 1-GS/s FFT/IFFT Processor for UWB Applications,” IEEE Journ. of SSC, vol. 40, Issue 8, Aug. 2005.

12

Radio-Frequency (RF) Beamforming Using Systolic FPGA-based Two Dimensional (2D) IIR Space-time Filters

Arjuna Madanayake, *Member, IEEE,* and Leonard T. Bruton, *Fellow, IEEE*
Multidimensional Signal Processing Group, University of Calgary
Calgary, Alberta, Canada
(hmadanay, bruton@ucalgary.ca)

1. Introduction

Plane-waves are far-field solutions to (1) the vector wave equation, for the case of electromagnetic waves, (2) the scalar wave equation, for the case of longitudinal pressure waves in seismic, acoustic, and ultrasonic systems, as well to (3) linear surface waves, such as those created by dropping a pebble into still the waters of a pond. Far-field beamforming refers to the highly-selective directional enhancement of propagating spatio-temporal plane-waves based on their directions-of-arrival (DOA).

The directional enhancement (beamforming) of electromagnetic plane-waves is of importance in many areas of electrical engineering, such as in wireless communications, radar and radio-frequency (RF) imaging (multi-GHz range). Of particular importance are applications in radio-astronomy and space physics (Van Ardenne,2000), where far-field beamforming is increasingly employed in aperture arrays, and in wireless mobile voice and data communication systems. In the case of data communications, beamforming is used for mitigating the fading effects of multipath propagation (Litva and Lo,1996; Liberti Jr. and Rappaport,1999; Huseyin Arslan, Zhi-Ning Chen and Maria-Gabriella Di Benedetto,2006) in satellite-borne remote sensing applications involving synthetic aperture and Doppler radar, in navigation and location devices based on GPS (Silva, Worrel and Brown), as well as in various ultra-wideband location technologies (Ghavami, Michael and Kohno).

Traditionally, receiver-side beamforming has been achieved using highly-directional receiving antennas, typically employing parabolic reflectors and horns, antenna array configurations with passive phasing networks (such as delay-and-sum networks and phased-array feeds) and reflect-arrays (Hum, Okoniewski and Davies). Digital beamforming algorithms are often based on fractional-delay steering algorithms and/or finite impulse response (FIR) digital filters (Ghavami, Michael et al.; Johnson and Dudgeon,1993; Liberti Jr. and Rappaport,1999; Staderini,2002; Huseyin Arslan, Zhi-Ning Chen et al.,2006; J. Roderick, H. Krishnaswamy, K.Newton et al.,2006). The use of digital signal processing (DSP) in far-

field broadband beamforming for smart antenna array applications is currently receiving much attention, mainly due to the continuously increasing availability of digital programmable logic and custom silicon fabrication technologies that are gradually enabling the typically high levels of real-time computational throughputs necessitated by such DSP-based broadband smart antenna arrays.

In this contribution, we describe a particular type of recently proposed far-field beamformer that is based on two-dimensional (2D) space-time digital filters having infinite impulse responses (IIRs) (Ramamoorthy and Bruton; Agathoklis and Bruton,1983; Bruton and Bartley,1985). Unlike the more widely-used DSP-based 2D *FIR* beamformers, the described 2D *IIR* beamformers have 2D **z**-domain transfer-functions $H(z_1, z_2) \equiv N(z_1, z_2) / D(z_1, z_2)$ havng pole-manifolds, as well as zero-manifolds, in the 2D complex plane $\mathbb{C}^2$. Further, for beamforming applications, $D(z_1, z_2)$ *must be non-separable*, implying non-trivial design challenges to avoid multidimensional instability and computability constraints (such challenges are not encountered in 1D design or if $D(z_1, z_2)$ is separable in the 2D case). However, suitable closed-form *algebraic* expressions for beamforming discrete-domain 2D IIR transfer functions are available in the literature that are computable, stable and realizable using hardware with lower complexity than FIR beamformers having similar directional selectivity, angular half-power bandwidth, etc. Furthermore, the existence of such closed-form algebraic transfer functions facilitates the real-time continuous steering of the direction of the beam and adjustment of its bandwidth, making these filters attractive for applications in emerging software-defined radio (SDR), microwave imaging and cognitive radio systems.

This paper is organized as follows: in section 2, we provide a brief review of space-time plane-waves and their properties in the 2D space-time frequency domain, followed by section 3, where we provide a comprehensive review of broadband plane-wave digital filter design. Thereafter, in section 4, we discuss practical hardware implementations starting from 2D difference equations that lead to signal flow graphs and massively-parallel systolic-array hardware realizations. Section 5 describes some recent progress we have achieved in prototyping these new systolic-array circuits using field programmable gate array (FPGA) technology. Finally, in section 6 we describe recently available technology and electronic design automation (EDA) tools that may eventually lead to 2D IIR beamformers that operate in real-time for various broadband microwave beamforming applications.

2. Electromagnetic Plane-waves in Space-time

We consider here either the transverse electric field $E_y(x, y, z, ct)$ or magnetic field $H_x(x, y, z, ct)$ of a propagating electromagnetic plane wave, where $(x, y, z, ct) \in \mathbb{R}^4$ is the 4D space-time continuous-domain, $(x, y, z) \in \mathbb{R}^3$ is 3D space, $t \in \mathbb{R}$ is time and $c \approx 3 \times 10^8 \ ms^{-1}$ is the speed of light in air/vacuum. By analogy with 3D planes, the equation

$$\alpha_1 x + \alpha_2 y + \alpha_3 z + ct = \pm\lambda,\ \ \alpha_{1,2,3} \in \mathbb{R},\ \text{and},\ \sqrt{\sum_{k=1}^{3} \alpha_k^2} \equiv 1 \tag{1}$$

is a 4D hyper-plane in the 4D continuous-domain $(x, y, z, ct) \in \mathbb{R}^4$. Propagating electromagnetic plane waves are 4D hyper-plane waves in (x, y, z, ct) given by

$$w(x, y, z, ct) = w_{PW}\left(\underbrace{\alpha_1 x + \alpha_2 y + \alpha_3 z + ct}_{\pm\lambda} \right), \qquad \sqrt{\sum_{k=1}^{3} \alpha_k^2} \equiv 1 \quad \forall \lambda \in \mathbb{R}^1 \tag{2}$$

and therefore have the property that they are constant-valued in each of the hyper-planes (1): that is, for each $\lambda \in \mathbb{R}^1$. Equivalently, for each value of λ, $w_{PW}(\pm\lambda)$ is a corresponding 4D iso-surface in (x, y, z, ct). In Fig. 1, we show the 4D plane wave $w_{PW}\left(\alpha_1 x + \alpha_2 y + \alpha_3 z + ct\right)$ in the 3D spatial domain $(x, y, z) \in \mathbb{R}^3$ in an iso-plane which, by simply 3D geometry, is perpendicular distance ct from the origin, as shown. In 3D, we may therefore visualize the 4D space-time plane wave of equation (2) as an infinite set of such iso-planes $w(\pm\lambda)$, each of which is *propagating in* (x, y, z) *over time t* with speed *c* in a direction normal to the iso-planes. Depending on the *1D* spectral properties of the *c*-scaled temporal signal $w_{PW}(ct)$, the plane wave might be temporally-narrowband or temporally-broadband.

Note that, for the case of the *ideal* plane wave, the region of support (ROS) of equation (2) in $(x, y, z, ct) \in \mathbb{R}^4$ extends, in general, to infinity in at least some directions in $\mathbb{R}^4$. Equation (2) represents either the electric or magnetic field of the plane wave in 4D space-time. In this chapter, we are only concerned with the values of the 4D plane wave signal as received on a straight line in (x, y, z). Therefore, we consider only the special case of the 2D space-time representation for which equation (2) reduces to the form $w(x, 0, 0, ct) = w_{PW}\left(\underbrace{\alpha_1 x + ct}_{\lambda} \right)$ for signals on the x-axis. With the DOA in 3D space defined by the angles θ_o(measured on x - z plane) and ψ_o as shown in Fig. 1, it is easily shown that (2) may be written in the form

$$w(x, y, z, ct) = w_{PW}\left(\underbrace{\left(-\sin\theta_o \cos\psi_o\right) x + \left(\cos\theta_o \cos\psi_o\right) z + \left(\sin\psi_o\right) y + ct}_{\lambda} \right) \tag{3}$$

with $-\pi/2 < \theta_o, \psi_o \le \pi/2,$ from which it follows that the corresponding 2D space-time plane wave signal *received on the x-axis* is given by

$$w(x,ct) = w_{PW}\left(\underbrace{\left(-\sin\theta_o \cos\psi_o\right)x + ct}_{\lambda}\right) \tag{4}$$

The 4D hyper-planar iso-*surfaces* of constant λ in (3) become 2D iso-lines of (4) in $(x,ct) \in \mathbb{R}^2$ given by

$$\left(-\sin\theta_o \cos\psi_o\right)x + ct = \lambda \tag{5}$$

As shown in Fig. 2, the space-time direction of the 2D space-time plane wave is defined by the normal to these contours and is given by (Gunaratne and Bruton; Khademi)

$$\theta = \tan^{-1}(\sin\theta_o \cos\psi_o) \tag{6}$$

with respect to the *ct*-axis in the 2D Cartesian space-time domain (x,ct), where $\sin\phi = \sin\theta_o \cos\psi_o$. Note from (6) that the 2D spatio-temporal direction $\theta = \tan^{-1}(\sin\phi)$ is constrained by $-\pi/4 \le \theta < \pi/4$ where the extreme values $\pm\pi/4$ occur where (($\theta_o = \pm\pi/2$) and/or $\psi_o = 0$. These extreme directions are known as the 'end-fire' angles, corresponding to plane waves having DOAs in 3D space that are in the direction of the x-axis. The so-called 'broadside' DOAs, with respect to the x-axis, are those directions for which the 4D space-time signal in (3) is constant everywhere on the x-axis at all instants of time *t*, which corresponds to the 2D space-time direction $\theta = 0$ and is equivalent to the DOAs in 3D space given by ($\theta_o = 0, \pi$ and/or $\psi_o = \pm\pi/2$).

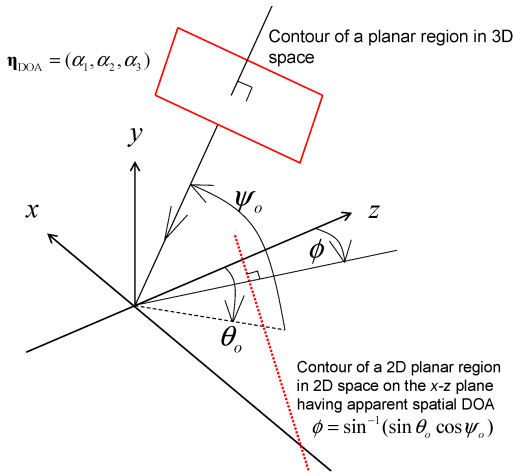


Fig. 1. The direction of arrival (DOA) of a plane-wave in 3D space, (θ_o, ψ_o), and the appararent spatial DOA seen by a linear array along x-axis, ϕ.

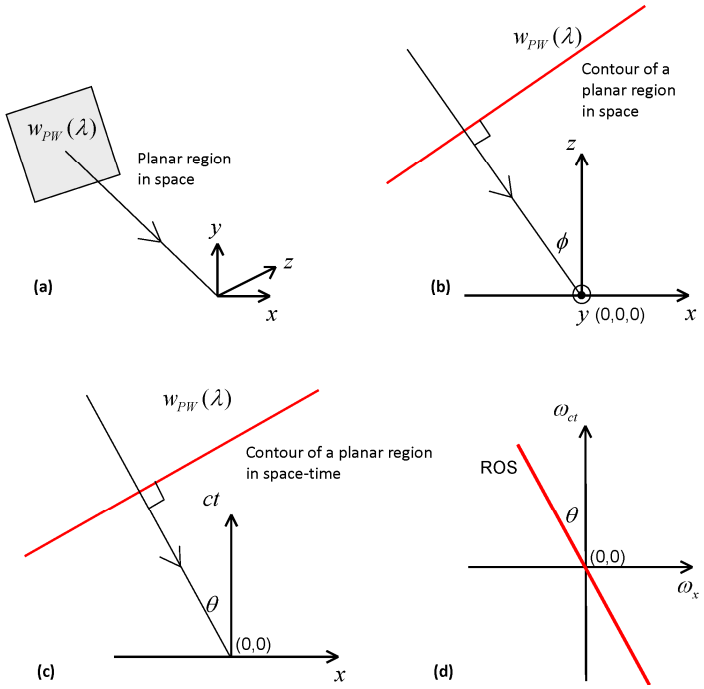


Fig. 2. Propagating plane-wave in 3D space (a), 2D spatial view on the $y = 0$ plane (b), 2D spatio-temporal DOA (c), and region of support (ROS) on 2D frequency-domain, aligned along the spatio-temporal DOA (d).

2.3 On The Region of Support (ROS) of the 2D Fourier Transform of 2D space-time Plane Waves

Given the 2D Fourier transform pair for 2D space-time plane waves $w_{PW}\left(-x\sin\theta + ct\right) \Leftrightarrow W_{PW}(e^{j\omega_x}, e^{j\omega_{ct}})$, it may be shown (Bruton and Bartley,1985) that the Region of Support (ROS) of the spectrum $\left|W_{PW}(e^{j\omega_x}, e^{j\omega_{ct}})\right|$ in the 2D Cartesian frequency domain (ω_x, ω_{ct}) is confined to the straight line

$$\omega_x + \omega_{ct}\sin\theta = 0 \tag{7}$$

which passes through the origin and subtends angle θ to the ω_{ct} axis. It lies on the ω_{ct} axis for broadside DOAs and on $\omega_{ct} = \pm\omega_x$ for end-fire DOAs. Importantly therefore, the ROS of all 2D space-time electromagnetic plane wave signals, propagating at speed *c*, cannot lie outside the 90-degree wide 2D fan-shaped region $\left|\omega_x\right| \le \left|\omega_{ct}\right|$ in (ω_x, ω_{ct}).

2.4 Spectral-Filtering a Desired 2D Space-time Plane-wave in the Presence of other Plane Waves and Noise

Typically, the signal received on the x-axis may be represented by M multiple broadband plane-waves, each having a different orientation $\theta_{o,k}, \psi_{o,k}$, $k = 0,1,2,\ldots(M-1)$, in 2D space-time, and additive 2D noise. We assume the first plane wave, given by $k = 0$ is the desired plane wave to be recovered by filtering. Then the received signal may be written in the form

$$w(x,ct) = \sum_{k=0}^{M-1} w_{PW,k}(-x\sin\phi_k + ct) + n_v(x,ct) \tag{8}$$

Where $\sin\phi_k = \sin\theta_{o,k}\cos\psi_{o,k}$, and where $n_v(x,ct)$ represents 2D space-time noise. The Fourier transform of (8) is therefore given by

$$W(e^{j\omega_x}, e^{j\omega_{ct}}) = \sum_{k=0}^{M-1} W_{PW,k}(e^{j\omega_x}, e^{j\omega_{ct}}) + N_v(e^{j\omega_x}, e^{j\omega_{ct}}) \tag{9}$$

where $N_v(\omega_x, \omega_{ct}) \overset{2D}{\Leftrightarrow} n_v(x,ct)$. Typically, $n_v(x,ct)$ corresponds to non-plane-wave electromagnetic propagating interference or other sources of 2D broadband noise, modelled as additive white Gaussian noise (AWGN). Therefore, the 2D ROS of the noise spectrum $|N_v(\omega_x, \omega_{ct})|$ is typically uniform throughout the 2D frequency-domain $(\omega_x, \omega_{ct}) \in \mathbb{R}^2$. The ROS of $W(e^{j\omega_x}, e^{j\omega_{ct}})$ therefore consists of the uniform ROS of $|N_v(\omega_x, \omega_{ct})|$ and M lines through the origin, where the orientation of each line is given by the M different angles $\theta_k = \tan^{-1}(\sin\phi_k)$.

For notational convenience in the rest of the chapter, we will use $\omega_1 \equiv \omega_x$ as the spatial frequency variable, and $\omega_2 \equiv \omega_{ct}$ as the time-frequency variable corresponding to spacetime ct.

2.5 Beamforming of a Broadband Plane Wave using Space-time Filters

The simplest of 2D space-time plane-wave filter is known as a 'frequency-planar beam' filter (Bruton and Bartley,1985), because it's passband lies on a line-through the origin and ideally has a 'beam' shaped 2D passband of uniform width. The beam-shaped 2D passband is oriented to enclose the ROS of the 2D spectrum of the desired plane-wave over its full temporal bandwidth while attenuating all spectra away from this narrow passband. Such beam filters can be realized using several methods involving both FIR or IIR digital filters (Gunaratne and Bruton; Khademi; Bruton and Bartley,1985). FIR filters are inherently stable but are of high arithmetic complexity due to the relatively higher order of the filter that is required for a given selectivity, relative to approximately-equivalent IIR filters. However, the latter are not as straightforward to design and to implement.

The focus of this chapter is on the design and real-time hardware implementation of a first-order 2D IIR beam digital plane-wave filter.

2.6 Effects of 2D Space-time Sampling using a Uniform Linear Array (ULA)

The 2D continuous-domain space-time signal $w(x,ct)$ is sampled in space using a number of equi-spaced antennas along the x-axis and sampled in time to yield the corresponding 2D sampled space-time signal $w(n_1\Delta x, n_2 c\Delta T_{CLK})$, $n_{1,2} = 0,1,2,3,\ldots$, and where. In order to prevent undesirable spatial aliasing, such uniform linear arrays (ULAs) of antennas require that the uniform distance between antennas satisfy the Nyquist condition (the distance between antennas is $\Delta x \le c\Delta T$, where $f_U \equiv 1/\Delta T$ is the upper temporal frequency of the input signal beyond which its spectrum lacks support). For N antennas, the 2D spatially-sampled continuous-time antenna array signals are given by $w(n_1\Delta x, ct), n_1 = 0,1,...,N-1$. The 2D spectrum of $w(n_1\Delta x, ct)$ replicates on the spatial-frequency ω_1 axis with periodicity 2π. The N continuous-time signals $w(n_1\Delta x, ct)$ are amplified, using a low noise amplifier having the required temporal bandwidth and noise performance, then low-pass filtered prior to A/D conversion, resulting in the 2D discrete-domain antenna signal $w(n_1\Delta x, n_2 c\Delta T_{CLK})$. The ADC clock frequency $F_{CLK} = 1/\Delta T_{CLK}$ where is chosen by selecting the inter-sample time $\Delta T_{CLK} \le \Delta T$ such that Nyquist sampling theorem is satisfied in both the spatial and temporal frequency domains. Usually, $\Delta T_{CLK} = K_U \Delta T$, $K_U > 1$, where K_U is the so-called temporal oversampling factor and is sometimes necessary for minimizing the frequency-warping effects that are introduced in the design of the digital filter transfer function.

Methods have been proposed and implemented for significantly reducing the required number of antennas without significantly reducing performance, for wireless communications and other applications. These methods also lead to much reduced arithmetic complexity of the filter and are based on allowing a controlled amount of multidimensional spatial aliasing and thereby spatial under-sampling, as reported in (Khademi and Bruton; Madanayake, Hum and Bruton).

3. First-order 2D IIR Frequency-Beam Plane-wave Filter

In the above, it is established that the directional enhancement of an ideal desired space-time plane-wave may be achieved using a 2D space-time filter having a 2D passband that encompasses the line-shaped ROS of the desired plane-wave signal. Further, undesired signals, such asother plane-waves and/or noise may be attenuated by ensuring that the ROS of the 2D stopband correspondas to the ROS of the spectrum of the undesired signals. Although this approach has been extended to 3D and 4D space-time signals (Kuenzle and Bruton; Bolle,1994; Bruton,2003; Dansereau,2003; Kuenzle and Bruton,2005; Dansereau and Bruton,2007), here we focus on the simplest 2D case by describing the design and implementation of a suitable 2D filter.

3.1 The Prototype Resistively-terminated 2D Passive Frequency-Beam Network

Consider a 2D first-order continuous-domain inductance-resistance network (Bruton and Bartley,1985), where s_1 and s_2 are spatial and temporal Laplace variables (Dudgeon and Mersereau,1990; Johnson and Dudgeon,1993; Schroeder and Blume,2000), respectively. The input-output 2D Laplace voltage transfer function of this network is given by

$$T(s_1, s_2) = \frac{R}{R + L_1 s_1 + L_2 s_2} \equiv \frac{Y(s_1, s_2)}{W(s_1, s_2)} \tag{10}$$

where the parameters $L_1 \geq 0, L_2 \geq 0,$ and $R > 0$ correspond to a passive *spatial* inductor, passive *temporal* inductor and passive resistance, respectively, with transform inputs and outputs $W(s_1, s_2)$ and $Y(s_1, s_2)$, respectively. We denote the respective transform pairs by $w(x, ct) \overset{2D}{\Leftrightarrow} W(s_1, s_2)$ and $y(x, ct) \overset{2D}{\Leftrightarrow} Y(s_1, s_2),$ respectively. The steady-state input-output frequency-response of (10) is found by setting $s_1 = j\omega_1$ and $s_2 = j\omega_2,$ leading to the 2D frequency response transfer function

$$T(j\omega_1, j\omega_2) = \frac{R}{R + j\left(L_1\omega_1 + L_2\omega_2\right)} \equiv \frac{Y(j\omega_1, j\omega_2)}{W(j\omega_1, j\omega_2)} \tag{11}$$

From (5), the network under consideration is *2D resonant on the 2D line-shaped region*

$$\omega_1 L_1 + \omega_2 L_2 = 0 \tag{12}$$

passing through the frequency-origin (Note: *In 2D, capacitors are not required to induce resonance*). At all finite frequencies where (12) is satisfied (i.e. throughout the 2D passband) , network energy resonates between the two inductance elements and $T(j\omega_1, j\omega_2)$ is unity. By choosing $L_1 = \cos\theta$ and $L_2 = \sin\theta$, $0 \leq \theta \leq 90^o$, we can orient the axis of the 2D passband to the angle θ. A typical response is shown in Fig. 3. The shape of the 2D gain $|T(j\omega_1, j\omega_2)|$ of the filter may be envisaged in 2D frequency space by noting that $L_1\omega_1 + L_2\omega_2 = \gamma$ describes, for constant λ, a line that is parallel to the 2D passband and along which $|T(j\omega_1, j\omega_2)|$ is constant and less than unity. Importantly, $|T(j\omega_1, j\omega_2)|$ decreases monotonically with increasing values of $|\lambda|$ with the two –3dB lines having gain 0.707 given by $\lambda = \pm R$. We make the following summary observations (see (Bruton and Bartley,1985) for details):

1. At 2D resonance, that is on the frequency-line in (ω_1, ω_2) where $\gamma = 0,$ the transfer function $T(j\omega_1, j\omega_2) = 1$. This defines the 2D passband axis and unity gain on the centre of the beam-shaped passband.
2. Along all directions orthogonal to the passband axis in (ω_1, ω_2), the magnitude and phase frequency response of the filter correspond to that of a first-order low-

pass transfer function, monotonically-decreasing with distance from the passband axis.

3. The uniform –3dB bandwidth of the beam is given by $\omega_{-3dB} = R / \sqrt{L_1^{\,2} + L_2^{\,2}}$.

3.2 The Transfer-functions of the First-order Beam Filter in the 2D s- and z- Domains

Although the inverse 2D Laplace transform of equation (10) yields a continuous-domain partial differential equation for the input-output transfer-function, practical implementations have so far been in the discrete-domain of 2D finite-difference equations, implemented in the form of digital circuits. Transformation to the discrete-domain is achieved by applying the normalized 2D bilinear transform (2D BLT) $s_k = \dfrac{z_k - 1}{z_k + 1}, k = 1, 2,$ to equation (10) leads, after considerable algebraic manipulation (Bruton and Bartley,1985), to the 2D **z**-transform transfer function

$$H(z_1, z_2)\Big|_{T\left(\frac{z_1-1}{z_1+1}, \frac{z_2-1}{z_2+1}\right)} = \frac{\left(1 + z_1^{-1}\right)\left(1 + z_2^{-1}\right)}{1 + \left(b_{10} + b_{11} z_2^{-1}\right) z_1^{-1} + b_{01} z_2^{-1}} \equiv \frac{Y(z_1, z_2)}{W(z_1, z_2)} \tag{13}$$

where $W(z_1, z_2) \overset{2D}{\Leftrightarrow} w(n_1 \Delta x, n_2 c \Delta T_{CLK})$ and $Y(z_1, z_2) \overset{2D}{\Leftrightarrow} y(n_1 \Delta x, n_2 c \Delta T_{CLK})$, respectively, and where $b_{ij} = \left(R + (-1)^i L_1 + (-1)^j L_2\right) / \left(R + L_1 + L_2\right)$. The above application of the 2D BLT, which is a conformal mapping between the 2D Laplace and 2D **z**-domain, results in a distortion ofn the high frequency part of the 2D passband, known an bilinear warping, that leads to a practical limitation of the upper frequency $\left(0.5\pi < |\, \omega_2 \,| < \pi\right)$ of the beam-shaped passband. This effects of this limitation may be avoided by suitable temporal and/or spatial over-sampling of the input signal.

For example, here we shall employ a temporal over-sampling factor of 2 for which we show in Fig. 4 the correpsonding 'weakly-warped' magnitude response of the discrete-domain frequency-response transfer-function over the useful range $|\, \omega_2 \,| \leq 0.5\pi$.

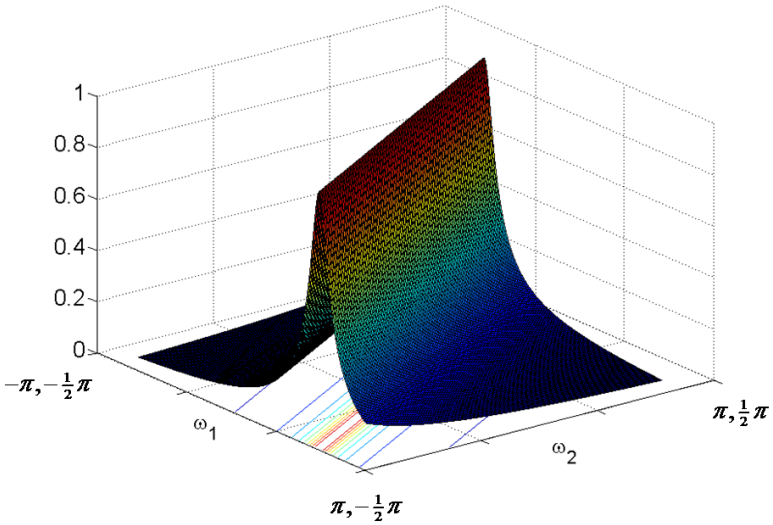


Fig. 3. The 2D continuous-domain steady-state Magnitude Frequency Response $|T(j\omega_1, j\omega_2)|$ of the filter in (10) for $|\omega_k| \le \pi, k = 1,2,$ and $R = 0.1,\ L_1 = \cos(30^o),\quad L_2 = \sin(30^o)$.

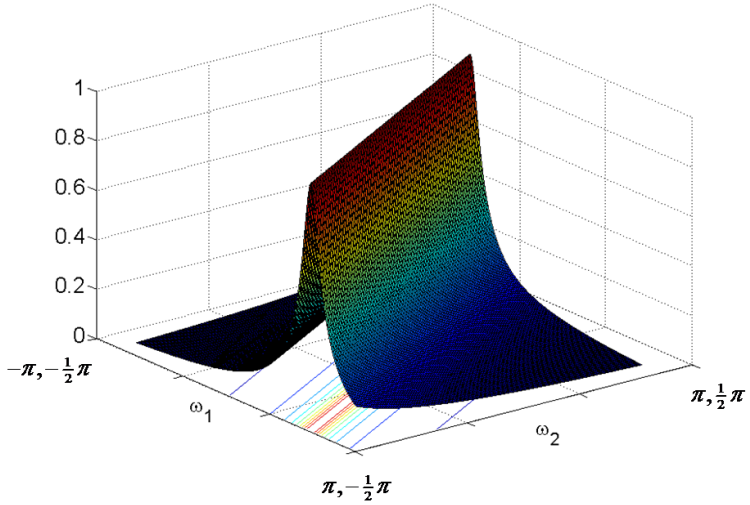


Fig. 4. A Beam-shaped Response that is warped by the 2D BLT, shown in the usable range $-\pi \le \omega_1 \le \pi$ and $-0.5\pi \le \omega_2 \le 0.5\pi$. The beam shape at frequencies $(0.5\pi < |\omega_2| < \pi)$ are not used in our application because the beam-shape is significantly off-axis, due to binear warping. The interested reader is referred to (Madanayake and Bruton; Bruton,2003) for details.

3.3 On Implementing the 2D Difference-Equations Using Differential Operators

Taking the inverse 2D **z**-transform of (10) leads to the 2D space-time input-output direct-form difference-equation, which we have shown can be implemented in massively-parallel systolic-array hardware for real-time filtering applications. However, the recently proposed hybrid-form signal flow graph (Madanayake, Hum et al.; Madanayake,2008), although relatively complicated to design, offers lower complexity and high-speeds of operation than direct-form methods. In this paper, we pursue the hybrid-form signal flow graph method. The 2D **z**-domain transfer-function that leads to the so-called hybrid-form structure (an architecture that enjoys the high-speeds of operation of direct-form structures while being as low in complexity as so-called differential-form structures) can be obtained by methods available in the literature (Madanayake, Hum et al.; Madanayake and Bruton,2007; Madanayake,2008). For example, we may write a suitable transfer function in terms of the spatial differential operator $z_{1D}^{-1} \equiv z_1^{-1}/(1+z_1^{-1})$ as

$$H(z_1,z_2) = \frac{1+z_2^{-1}}{1-\alpha \underbrace{\frac{z_1^{-1}}{1+z_1^{-1}}}_{z_{1D}^{-1}}\left(1+z_2^{-1}\right)+\beta z_2^{-1}} \equiv \frac{Y(z_1,z_2)}{W(z_1,z_2)} \tag{14}$$

where we require

$$\alpha = \frac{2\cos\theta}{R+\cos\theta+\sin\theta}, \quad \beta = 1-\frac{2\sin\theta}{R+\cos\theta+\sin\theta} \tag{15}$$

Note that the passband gain in (14) is scaled by the constant $\frac{R}{R+L_1}$, relative to the direct-form case (Bertschmann, Bartley and Bruton; Madanayake, Hum et al.; Liu and Bruton,1989; Madanayake and Bruton,2007; Madanayake,2008), and is ignored in the following because it is not of practical significance. Re-writing the direct-form transfer-function using the spatial-differential operator results in just two filter coefficients in the denominator of (14) instead of three, implying a 33% reduction in the number of parallel hardware multipliers required in circuit realizations, relative to direct-form realizations.

The difference-equation realizations described here lead to practical-bounded-input-bounded-output (practical-BIBO) stable (Agathoklis and Bruton,1983) performance under finite precision arithmetic, assuming zero initial conditions (ZICs) for both spatial and temporal iterations. Methods that guarantee ZICs are considered later in the following.

4. A Real-time High Throughput Implementation using the Hybrid-form Systolic-Array Processor Architecture

Systolic-array processors are massively-parallel computers having identical, synchronously clocked, fully-pipelined, high throughput identical processing elements, which are connected in a linear- or meshed-array configuration

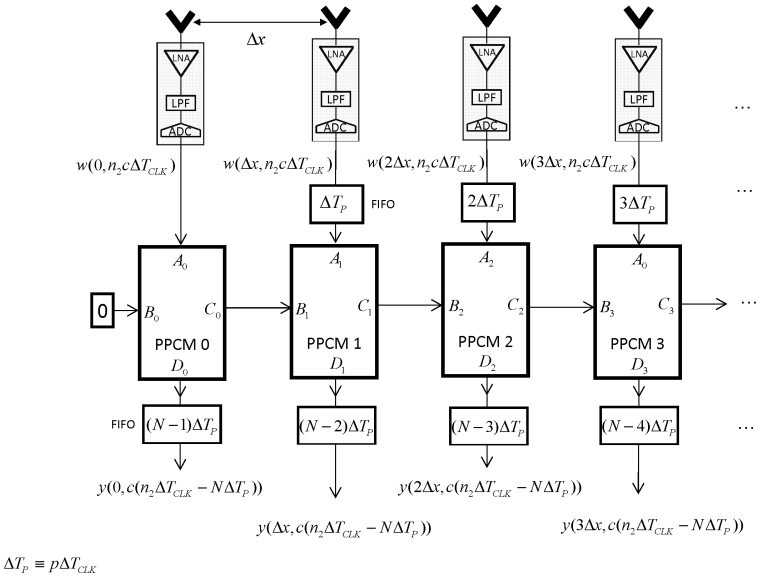


Fig. 5. Overview of the plane-wave filter implementation consisting of N parallel processing core modules (PPCMs), which have an internal signal flow graph based on the hybrid-form transfer-function in equation (14).

(Sid-Ahmed; Kung,1988a; Kung,1988b; Shanbhag,1991; Rader,1996; Zajc, Sernec and Tasic,2000). Such systolic-array processors are modular, regular, and locally interconnected, making them well-suited for real-time signal processing using application-specific VLSI hardware for digital signal processing applications at radio-frequency.

Research on novel systolic-array architectures for 2D/3D IIR frequency-planar digital plane-wave filters for beamforming applications has lead to field-programmable gate-array (FPGA) based single-chip multiprocessor implementations capable of real-time operation at a sustained arithmetic throughput of one-frame-per-clock-cycle (OFPCC), a requirement for real-time plane-wave filtering at RF using linear- or rectangular-arrays of antenna elements (Hum, Madanayake and Bruton; Madanayake, Bruton and Comis; Madanayake and Bruton; Madanayake and Bruton; Madanayake, Hum et al.; Madanayake, Hum and Bruton; Madanayake,2004; Madanayake,2008; Madanayake and Bruton,2008). The required OFPCC throughput rate, required for multi-GHz implementations, arises due to the fact that the signals of interest are of ultra-wide RF bandwidth, which leads to Nyquist sample rates that are at least twice the full RF bandwidth of the signal.

The beamformers therefore directly sample RF signals from the antennas without down-conversion (or bandpass sampling), and leads to frame sample rates in the GHz. Such excessively-high frame sample rates (multiple GHz) make software-based realizations infeasible using traditional DSP technologies. Our research indicates (Madanayake and Bruton; Madanayake,2008) that massively-parallel synchronously-clocked, speed-optimized,

fully-pipelined systolic-array processors are currently the best available solution for the broadband real-time DSP-based radio-frequency (RF) beamforming applications using sampled antenna arrays (Arnold Van Ardenne; Ellingson,1999; Liberti Jr. and Rappaport,1999; Weem, Noratos and Popovic,1999; Frederick, Wang and Itoh,2002; Do-Hong and Russer,2004; Rodenbeck, Sang-Gyu, Wen-Hua et al.,2005; Madanayake,2008; Devlin,Spring 2003).

4.1 Overview of the Architecture

The massively-parallel systolic-array architecture consists of an array of identical parallel-processing core-modules (PPCMs), sometimes called "processing elements" in the literature (Kung,1988a). Each PPCM is dedicated to processing an antenna element, and signals from all N elements are amplified using a low-noise amplifier (LNA), low-pass filtered (LPF), and time-synchronously sampled using N identical analog signal processing chains. The PPCMs and analog-to-digital converters (ADCs) are clocked using a single-phase master clock signal, of frequency $F_{CLK} = 1/\Delta T_{CLK}$. The PPCMs are derived using the recently-proposed hybrid-form signal flow graph having the required **z**-domain transfer-function (14) (Hum, Madanayake et al.; Madanayake, Hum et al.).

The PPCMs that comprise the systolic-array processor are fully-parallel, speed-maximized, fully-pipelined, multi-input-multi-output (MIMO) processors, each consisting of 2 input ports and 2 output ports. A PPCM at spatial location n_1 has its input port A connected to the ADC at location n_1 and input port B connected to the output port C of the PPCM at location $n_1 - 1$. Port D provides the computed output signal $y(n_1\Delta x, n_2 c\Delta T_{CLK})$ for spatial location n_1.

4.2 Inter-PPCM and Intra-PPCM Pipelines

The PPCMs are pipelined such that signals entering through the input ports A_{n_1} and B_{n_1} undergo p additional clocked delays, as a result of internal pipelining. These additional delays can be compensated by delaying the input signal A_{n_1+1} by $(n_1+1)p$ clock cycles, leading to a delay of $(n_1+1)\Delta T_x$ seconds, where $\Delta T_x = p\Delta T_{CLK}$ is the pipelining latency of a PPCM. For N PPCMs, the final output signal at the output of the N^{th} PPCM therefore undergoes a pipelining delay of Np clock cycles. When 2D space-time output signals are required, the output signals from each PPCM must be fed through additional clocked FIFOs, having depth $(N-1-n_1)p$, so that the signals at all spatial output locations are uniformly delayed by Np clock cycles. The implemented transfer-function is therefore modified, in the presence of pipelining, to the linear phase-delayed form $H(z_1, z_2)z_2^{-Np}$ which has no effect on the magnitude frequency-response function (because $\left|e^{-j\omega_2 Np\Delta_{CLK}}\right| = 1.$)

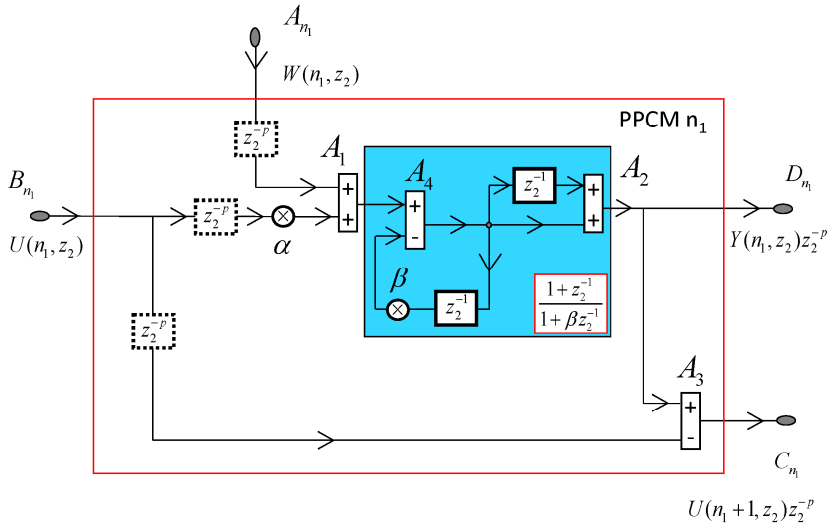


Fig. 6. Hybrid-form PPCM circuit having 2 inputs, 2 outputs, 4 two-input parallel adder/subtractors, and 2 parallel hardware multipliers.

4.3 Design of the Hybrid-form PPCMs

Having obtained a basic overview of the systolic-array architecture, we now derive the internal signal flow and internal components of each PPCM. Recall the 2D hybrid-form transfer-function (14):

$$H(z_1, z_2) = \frac{1+z_2^{-1}}{1-\alpha\dfrac{z_1^{-1}}{1+z_1^{-1}}\left(1+z_2^{-1}\right)+\beta z_2^{-1}} \equiv \frac{Y(z_1, z_2)}{W(z_1, z_2)} \tag{16}$$

Cross-multiplying terms in (16), we get the 2D **z**-domain input-output form, given by

$$\left(1+z_2^{-1}\right)W(z_1, z_2) = \left(1-\alpha\frac{z_1^{-1}}{1+z_1^{-1}}1+z_2^{-1}+\beta z_2^{-1}\right)Y(z_1, z_2), \tag{17}$$

leading to

$$Y(z_1, z_2) = \frac{\left(W(z_1, z_2)+\alpha\dfrac{z_1^{-1}}{1+z_1^{-1}}Y(z_1, z_2)\right)}{1+\beta z_2^{-1}}\left(1+z_2^{-1}\right) \tag{18}$$

Multiplying both sides by z_2^{-p} yields the required form

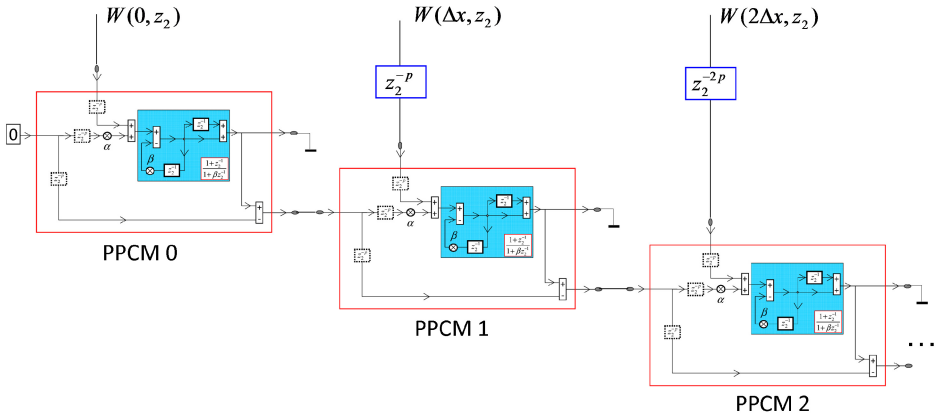


Fig. 7. Interconnections between PPCMs, shown here in the mixed domain $(n_1,z_2)\in\mathbb{ZC}$, leads to the massively-parallel systolic-array processor implementation of the beam plane-wave filter.

$$Y(z_1,z_2)z_2^{-p}=\frac{\left(W(z_1,z_2)z_2^{-p}+\alpha z_2^{-p}\dfrac{z_1^{-1}}{1+z_1^{-1}}Y(z_1,z_2)\right)}{1+\beta z_2^{-1}}\left(1+z_2^{-1}\right) \tag{19}$$

Computing the inverse z1-transform of (19) under spatial ZICs, we obtain the 2D mixed-domain $(n_1,z_2)\in\mathbb{ZC}$ form, given by

$$Y(n_1,z_2)z_2^{-p}=\frac{\left(W(n_1,z_2)z_2^{-p}+\alpha z_2^{-p}U(n_1,z_2)\right)}{1+\beta z_2^{-1}}\left(1+z_2^{-1}\right) \tag{20}$$

where

$$U(n_1,z_2)=Y(n_1-1,z_2)z_2^{-p}-U(n_1-1,z_2)z_2^{-p} \tag{21}$$

and, where z_2^{-p} is the z_2-transform of the internal pipelining delays at each PPCM. Because the depth of pipelining is arbitrary, the numerator of (20) can be pipelined at will using straightforward 1D FIR filter pipelining methods, noting that this 1D FIR section has two terms $W(n_1,z_2)$ and $U(n_1,z_2)$ which are obtained using digital ports A_{n_1} and B_{n_1}, respectively. Equations (20) and (21) describe the 2-input-2-output z_2-domain transfer-functions of a PPCM at location $0\le n_1<N-1$. The hybrid-form signal flow-graph is thereby obtained, and is shown in Fig. 6. The first 3 PPCMs in the systolic-array are shown in Fig. 7 as an interconnection of processors.

4.4 A Pipelining Example

In order to familiarize the reader with pipelining concepts, we now provide a simple example where it is assumed that $p = 12$ internal pipelining stages are sufficient for achieving the required throughput. The 12 stage pipeline will be distributed as follows: the multiplier α will consist of 3 level pipelining; the three 2-input adders/subtractors denoted A1, A2, and A3, are to have 3 levels of pipelining. It is important to ensure that all signal components that connect to a particular 2-input adder/subtractor undergo equal delays. This is essential for correct operation, and must be satisfied for all pipelined designs.

The hybrid-form signal-flow graph does not allow pipelining of A4, because only one unit-delay buffer is available in the first-order feedback loop, which is usually absorbed inside the parallel logic of multiplier β. Provided all feed-forward paths are fully pipelined, the critical path delay of the hybrid-form PPCM cannot be reduced beyond $T_{CPD} \approx T_{Mul} + T_{A/S}$ where T_{Mul} and $T_{A/S}$ are the propagation delays of a parallel multiplier and adder/subtractor circuit, respectively. The maximum speed of operation for a hybrid-form PPCM is therefore less than $F_{CLK} \leq 1/\Delta T_{CPD}$ unless additional speed-optimization methods, based on look-ahead optimization, are employed. This method is discussed in the next section.

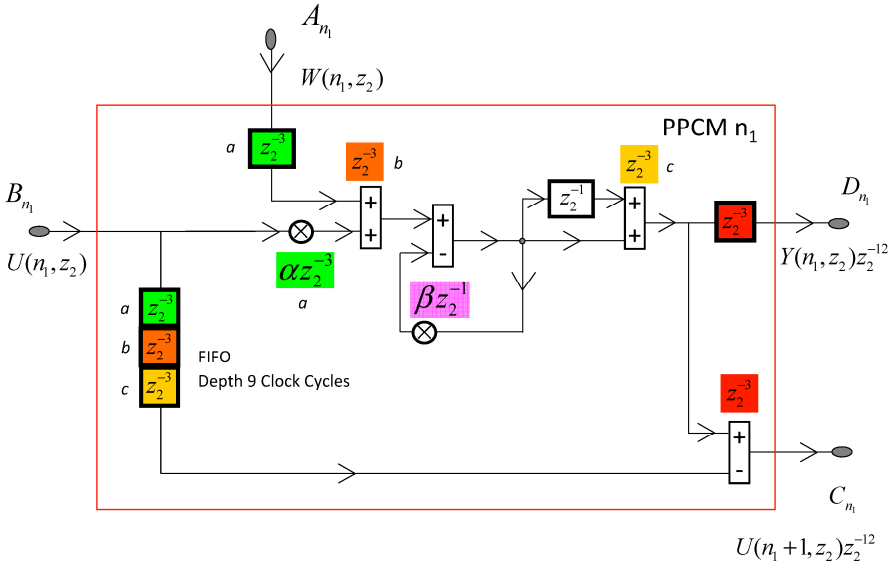


Fig. 8. Signal flow graph of a hybrid-form PPCM having 12 cycles of pipeline latency (arbitrarily chosen for the purpose of demonstration). The 12 clock-cycles of additional pipelining can be used as required to reduce the critical-path delay (CPD) of the systolic-array.

The pipelined version, having $p = 12$ for the hybrid form PPCM, is shown in Fig. 8. We now describe look-ahead speed-optimization of the internal 1D temporal IIR digital filter section having transfer-function $\frac{1+z_2^{-1}}{1+\beta z_2^{-1}}$ that enables much greater levels of real-time throughput at the cost of additonal circuit complexity.

4.5 Additional Look-Ahead Speed-Maximization

We extend well-known 1D IIR pipelining using "look-ahead" optimization, a method pioneered by Parhi et al (Parhi; Parhi,1991; Parhi,1999). Look-ahead is a method for reducing additional delays into a critical feedback loop and is based on pole-zero cancellation of 1D z-domain transfer-functions.

In section 4.4, we described an example for which 10 additonal delays are distributed in the forward (that is, FIR, also known as feed-forward) signal paths of the PPCM, such that the critical path delay of the PPCM (and therefore, of the systolic-array) is reduced to the latency for a multiply-add-operation, denoted ΔT_{CPD}. The speed-bottleneck for this example lies within the first-order feed-back IIR filter, which has a simple real-pole at $z_2 = -\beta$ where it may be shown that $|\beta| \le 1$ for passive filter network prototypes. Because this pole is within (or on) the unit circle $|z_2| = 1$ the 1D IIR filter section is unconditionally stable (ignoring effects due to finite precision).

Let us further assume that our objective is to halve the critical path delay using look-ahead optimization of the IIR section. This can be achieved by increasing the number of internal delays in the first-order feedback loop to 2 (causing the feedback loop to increas in order): this may be easilty achieved by multiplying both numerator and denominator of $\frac{1+z_2^{-1}}{1+\beta z_2^{-1}}$ by $1-\beta z_2^{-1}$ leading to the 2nd order section, given by $\frac{\left(1+z_2^{-1}\right)\left(1-\beta z_2^{-1}\right)}{1-\beta^2 z_2^{-2}}$ leading to a new critical path delay in the feedback loop $T_{CPD,LA} \approx T_{CPD}/2$, implying an almost 100% increase in the maximum speed of operation (Parhi; Parhi; Parhi and Messerschmitt; Parhi and Messerschmitt; Sundarajan and Parhi; Parhi and Messerschmitt,1989; Parhi,1991; Parhi,1999). This "look-ahead" speed-maximization process may be repeated: for example, by multiplying the numerator and denominator of $\frac{\left(1+z_2^{-1}\right)\left(1-\beta z_2^{-1}\right)}{1-\beta^2 z_2^{-2}}$ by $1+\beta^2 z_2^{-2}$ leads to a 4th-order feedback loop having transfer-function $\frac{\left(1+z_2^{-1}\right)\left(1-\beta z_2^{-1}\right)\left(1+\beta^2 z_2^{-2}\right)}{1-\beta^4 z_2^{-4}}$ which allows the multiplier β^4 to consist of 3 levels of internal pipelining, while the fourth delay can be used in the 2-input adder that completes the feedback loop (Madanayake and Bruton; Madanayake,2008). The additional terms in the numerator that appear due to the application of look-ahead speed-maximization lead to additional circuit complexity – this is

the price for the extensive gain in real-time throughput, which is 300% for 4th order feedback loops. The additional arithnetic circuits that appear in the feed-forward sections can be easily pipelined by increasing the depth of pipelining p as required. In our example, we have increase the depth of pipelining up to $p = 22$ which allows 3-level pipelining to all additional adders/subtractors and multipliers in the PPCM.

5. FPGA Circuit Prototypes

In this section, we provide a proof-of-concept circuit design using a field programmable gate array (FPGA) device. An example implementation of a hybrid-form systolic-array processor containing 21 fully pipelined PPCMs is provided. The target FPGA is a Xilinx Virtex-4 Sx35-10ff668 device installed on a Nallatech BenADDA daughter card, which in turn is installed on a Nallatech BenONE mainboard. This particular combination is widely known as the Xilinx XtremeDSP Kit-4.

The logic design flow starts with the Xilinx System Generator (XSG) design tool, which is a plug-in for Matlab/Simulink. We chose XSG as our FPGA design tool, although conventional design methods based on hardware description languages such as VHDL or Verilog may also be attempted. The modular regular nature of the systolic-array, together with the complicated pipelines and dataflow structure, makes the use of a graphical FPGA design method such as XSG, easier, compared to text-based design tools. We however note that XSG, in the end, leades to synthesizable VHDL (or Verilog), which is subsequently processed by conventional FPGA logic synthesis tools such as the Xilinx Synthesis Tool (XST) or Synplify Pro.

5.1 Finite Precision Arithmetic and the FPGA Circuit

The arithmetic circuits on the FPGA are obviously based on finite precision hardware blocks for the multipliers, adders/subtractors, and memory devices. The designation of precisions (word sizes) is an important design step that requires extensive further research. Our example is based on experience with many similar circuits, and is largely a result of experiential learning accumulated over several years of research on similar systolic-arrays. At this time, a comprehensive design method that can lead to optimal finite precision levels (in terms of hardware resource consumption, quantization noise statistics, power consumption, and throughput) is not available, and is an interesting subject for research activities.

The following example assumes input signals obtaining from 4-bit A/D converters. Preliminary studies show that 3 bit A/D converters are quite sufficient for ultra-wideband wireless communications applications. Our choice of 4-bits in our A/D converters results from 1-bit overdesign, mainly as a margin of safety, in order to ensure good performance

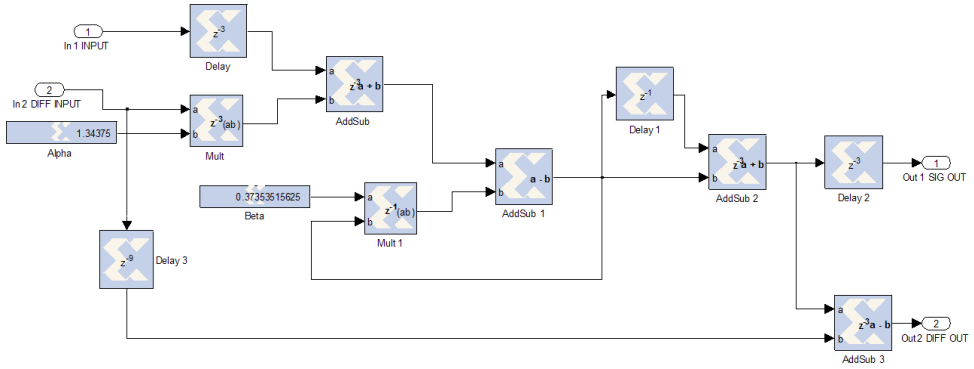


Fig. 9. Xilinx FPGA circuit for a hybrid-form PPCM having 12 cycles of pipeline latency (corresponding to the signal-flow graph in Fig. 8).

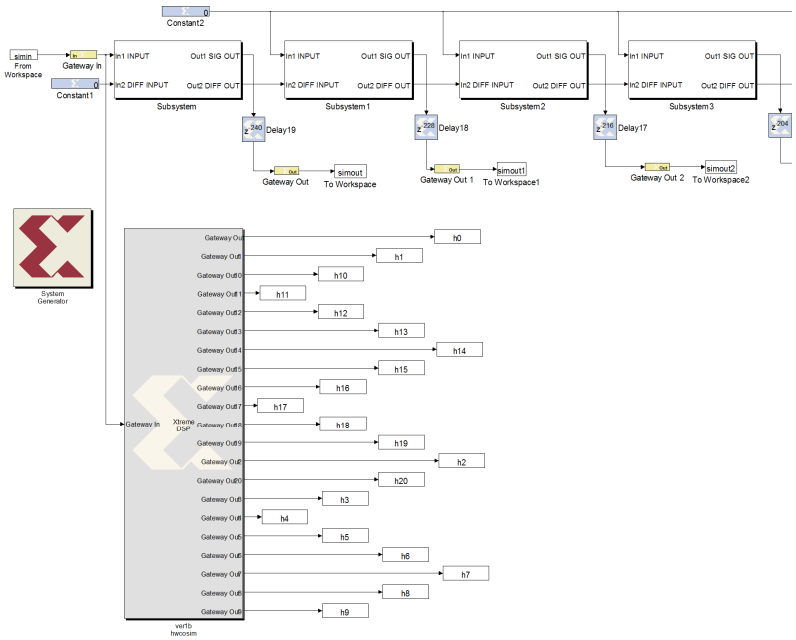


Fig. 10. First 4 PPCMs of a hybrid-form systolic-array FPGA circuit showing inter-PPCM interconnections. The FPGA circuit is tested on-chip using stepped hardware co-simulation using a 2D unit impulse input at PPCM #1, with inputs of PPCM #2, #3, …, #21, set to zero, leading to the 2D measured impulse response $h(n_1\Delta x, n_2 c\Delta T_{CLK})$. A bit-true cycle-accurate FPGA circuit simulation of the 2D impulse response is available in the Matlab variables *simout,simout1,…,simout20,* and the measured on-chip FPGA circuit response are available in Matlab variables *h0,h1,…,h20.*

from a real-world application. The multiplier coefficients are assumed to be 12 bits, with the binary point at position 10. All other registers, including quantized outputs from multipliers and adder/subtractor blocks, are fixed at 14-bits, with binary point assumed at position 10. The design of a PPCM is shown in Fig. 9, followed by the systolic-array, in Fig. 10. The finite precision values at various locations on the PPCM signal flow graph can be widely optimized against various requirements, but is not attempted here, because we are only interested in giving our readers a basic design overview of the hybrid-form systolic-array processor.

The FPGA circuit was tested, using on-chip hardware-in-the-loop co-simulation, using Matlab/Simulink, XSG, and FUSE, using the XtremeDSP Kit-4 device, which was installed on the 5V 32-bit PCI slot of the host PC. Figure 11 shows the measured 2D magnitude frequency response of an example beam filter having spatial DOA $\phi_o = 25^o$, and bandwidth parameter $R = 0.02,$ computed for 21 spatial samples, and 256 time samples, of the impulse response. The "uneven" nature of the measured response is attributed to quantization effects, and magnitude sensitivity, for which a comprehensive study remains as useful future research.

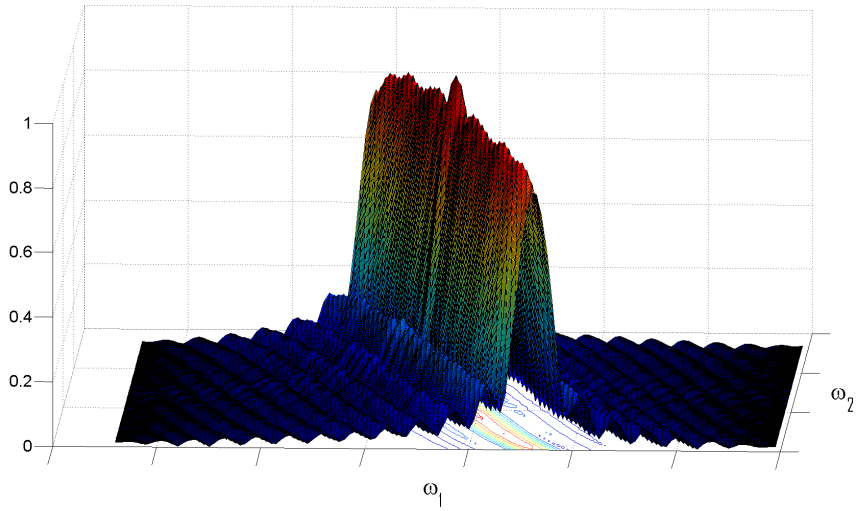


Fig. 11. Measured 2D magnitude response, $-\pi \le \omega_1 \le$ and $-0.5\pi \le \omega_2 \le 0.5\pi,$ obtained from a 21 PPCM FPGA implementation using on-chip hardware co-simulation using the Xilinx XtremeDSP Kit-4. Quantization effects cause the implementation to have extra ripples in both pass- and stop-bands, and lead to addition of AGWN. A detailed study of quantization effects remain for future work.

5.2 High-speed Implementation Technologies

At present, systolic-array implementations of the proposed 2D IIR beam plane-wave filters have been limited to proof-of-concept realizations on FPGA circuits that operate at clock rates of up to 100 MHz. However, real-world electromagnetic applications requires frame rates in excess of 1 GHz and can be high as 21 GHz for full-band UWB radio systems. FPGA circuit implementations are often impractical for product applications and are mostly used as prototypes for eventual implementation using application-specific integrated circuits (ASICs) using high-speed VLSI platforms such as the state-of-the-art 40nm digital CMOS process. Porting the available FPGA designs to 40nm CMOS (or similar) VLSI technology remains an exciting field for future research.

5.3 Field-Programmable Object Arrays (FPOAs) and Asynchronous FPGA Circuits

In general, although FPGAs from vendors such as Xilinx and Altera are limited to speeds less than ≈ 300 MHz for most *recursive filter designs*, future developments may facilitate the use of conventional FPGA technology to implement the proposed systolic arrays at 1 GHz or higher. Furthermore, it should be noted that fab-less semiconductor technologies, such as MathStar's (*http://www.mathstar.com/*) Arrix field programmable object array (FPOA) devices (Anonymous,2007) and high-speed "picoPIPE" FPGAs capable of 1.5 GHz operation (Anonymous,2008) from *Achronix Semiconductor (http://www.achronix.com/)*, are emerging as an alternative to conventional ASIC solutions, and forms a basis for future research.

FPOAs, from MathStar, consists of an array of hard IP blocks such as arithmetic-and-logic units (ALUs) and multiply-accumulate (MACs) blocks, within a reconfigurable switching fabric, which are ideal for systolic-array realizations due to their modularity, regularity, and local interconnectivity. These "objects" are pre-fabricated onto the FPOA structure, and meet stringent timing standards, enabling deterministic design at 1 GHz which are independent of the logic being implemented. On the other hand, *Speedster* FPGAs from *Achronix Semiconductor*, employ asynchronous handshaking protocols between combinational logic blocks, which they describe as the merging of clock and data tokens into one signal, which in turn, according to *Achronix* documentation, enables faster operation compared to conventional synchronous FPGA architectures. The Speedster family boasts 1.5 GHz, and is potential candidate for real-time implementation of the systolic-array architectures described herein, following future research.

6. Conclusions

The above new systolic implementation of a 2D IIR frequency-beam filter transfer function has promising engineering applications for the directional enhancement of a propagating broadband space-time plane-wave received on an array of sensors. A particularly important case is the use of an array of broadband antennas for the directional enhancement (that is, beamforming) of ultra-wideband electromagnetic plane-waves.

A massively-parallel systolic-array custom architecture, that is capable of processing one linear frame per clock cycle (OFPCC) with detailed design and optimization information, has been described. The architecture is based on the recently proposed hybrid-form 2D signal flow graph, which has been shown to be optimal in terms of critical path delay (hence

maximum throughput, because at OFPCC, the clock rate is equal to the frame rate in these architectures) and low computational complexity.

A design example for the proposed systolic-array processor architecture has been described using a Xilinx Virtex-4 Sx35 FPGA device, and the Matlab/Simulink based FPGA design tool called Xilinx System Generator. The example FPGA implementation of the 2D IIR frequency- beam filter was tested on-chip using the hardware-in-the-loop verification method called 'hardware co-simulation', and the on-chip 2D unit-impulse response was *measured*, which in turn led to *measured* 2D frequency response results that confirm correct implementation of the hardware.

Although the FPGA-based example is generally too slow for microwave imaging applications, it serves as a validation of the proposed OFPCC systolic-array processor and can be used in its current form for slower applications in audio, ultra-sound, and lower radio frequencies (of up to approximately 100 MHz). Finally, promising new VLSI implementation platforms are described here , which may eventually enable the proposed architecture to operate at the required multi-GHz clock frequency to enable real-time ultra-wideband digital smart antenna array applications.

7. References

Agathoklis, P. and L. T. Bruton (1983). "Practical-BIBO stability of N-dimensional discrete systems." Proc. Inst. Elec. Eng. **130, Pt. G**(6): 236-242.

Anonymous (2007). Arrix FPOA Overview. Available online at http://www.mathstar.com.

Anonymous (2008). Using High-Performance FPGAs for Advanced Radio Signal Processing. . Available online at http://www.achronix.com.

Arnold Van Ardenne. The Technology Challenges for the Next Generation Radio Telescopes. Perspectives on Radio Astronomy - Technologies for Large Antenna Arrays, Netherlands Foundation for Research in Astronomy.

Bertschmann, R. K., N. R. Bartley and L. T. Bruton A 3-D integrator-differentiator double-loop (IDD) filter for raster-scan video processing. IEEE Intl. Symp. on Circuits and Systems, ISCAS'95.

Bolle, M. (1994). A Closed-form Design Method for 3-D Recursive Cone Filters IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP.

Bruton, L. T. (2003). "Three-dimensional cone filter banks." IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications **50**(2): 208-216.

Bruton, L. T. and N. R. Bartley (1985). "Three-dimensional image processing using the concept of network resonance." IEEE Trans. on Circuits and Systems **32**(7): 664-672.

Dansereau, D. (2003). 4D Light Field Processing and its Application to Computer Vision. Electrical and Computer Engineering. Calgary, University of Calgary. **MSc**.

Dansereau, D. and L. T. Bruton (2007). "A 4-D Dual-Fan Filter Bank for Depth Filtering in Light Fields." Signal Processing, IEEE Transactions on **55**(2): 542-549.

Devlin, M. (Spring 2003) "How to Make Smart Antenna Arrays." Xcell Journal Online

Do-Hong, T. and P. Russer (2004). Signal Processing for Wideband Smart Antenna Array Applications. IEEE Microwave Magazine. **5**.

Dudgeon, D. E. and R. M. Mersereau (1990). Multidimensional Digital Signal Processing. Englewood Cliffs, N.J. 07632, Prentice-Hall.

Ellingson, S. W. (1999). A DSP Engine for a 64-Element Array. Proceedings on Perspectives for Radio Astronomy-- Technologies for Large Antenna Arrays, Netherlands.

Frederick, J. D., Y. Wang and T. Itoh (2002). "A smart antenna receiver array using a aingle RF channel and digital beamforming." IEEE Trans. on Microwave Theory and Techniques **50**(12): 3052-3058.

Ghavami, M., L. B. Michael and R. Kohno Ultra wideband signals and systems in communication engineering, John Wiley and Sons., Inc.

Gunaratne, T. K. and L. T. Bruton "Beamforming of Broadband-bandpass Plane Waves using Polyphase 2D FIR Trapezoidal Filters." IEEE Trans. on. Circuits and Systems: Regular Papers **55**(3): 838-850.

Hum, S. V., H. L. P. A. Madanayake and L. T. Bruton "UWB Beamforming using 2D Beam Digital Filters." IEEE Trans. on Antennas and Propagation **57**(3): 804-807.

Hum, S. V., M. Okoniewski and R. J. Davies "Modeling and Design of Electronically Tunable Reflectarrays." IEEE Trans. on Antennas and Propagation **55**(8): 2200-2210.

Huseyin Arslan, Zhi-Ning Chen and Maria-Gabriella Di Benedetto (2006). Ultra-wideband Wireless Communication, Wiley Interscience.

J. Roderick, H. Krishnaswamy, K.Newton, et al. (2006). "Silicon-based ultra-wideband beam-forming." IEEE Journal of Solid-State Circuits **41**(8): 1726-1739.

Johnson, D. H. and D. E. Dudgeon (1993). Array Signal Processing-Concepts and Techniques. Englewood Cliffs, N.J. 07632, Prentice-Hall.

Khademi, L. Reducing the computational complexity of FIR 2D fan and 3D cone filters [MSc Thesis]. Electrical and Computer Engineering, University of Calgary.

Khademi, L. and L. T. Bruton On the limitations of narrow 2D fan filters speech processing. IEEE 2003 Pacific Rim Conference on Communications, Computers, and Signal Processing (PACRIM'03).

Kuenzle, B. and L. T. Bruton "3-D IIR filtering using decimated DFT-polyphase filter bank structures." IEEE Trans. on Circuits and Systems I: Regular Papers **53**(2): 394-408.

Kuenzle, B. and L. T. Bruton (2005). A novel low-complexity spatio-temporal ultra wide-angle polyphase cone filter bank applied to sub-pixel motion discrimination. IEEE Intl. Symp. on Circuits and Systems, ISCAS'05, Kobe, Japan.

Kung, S. Y. (1988a). VLSI Array Processors, Prentice-Hall, Englewood Cliffs, N.J.

Kung, S. Y. (1988b). VLSI Array Processors: Designs and Applications. 1988 IEEE International Symp. on Circuits and Systems, ISCAS'88.

Liberti Jr., J. C. and T. S. Rappaport (1999). Smart Antennas for Wireless Communications-IS-95 and Third Generation CDMA Applications. Upper Saddle River, N.J. 07632, Prentice-Hall.

Litva, J. and T. K.-Y. Lo (1996). Digital Beamforming in Wireless Communications, Artech House.

Liu, Q. and L. T. Bruton (1989). "Design of 3-D planar and beam recursive digital filters using spectral transformation." IEEE Trans. Circuits and Systems **36**(3): 365-374.

Madanayake, A. (2004). FPGA Architectures for 2D/3D Digital Filters. Electrical and Computer Engineering. Calgary, University of Calgary. **MSc:** 205.

Madanayake, A. (2008). Real-time FPGA Architectures for Space-time Frequency-planar MDSP. Electrical and Computer Engineering. Calgary, University of Calgary. **PhD:** 371.

Madanayake, A. and L. Bruton A Review of 2D/3D IIR Plane-wave Real-time Digital Filter Circuits. IEEE Canadian Conference on Electrical and Computer Engineering, CCECE'05, Saskatoon, Sasketchawan, Canada.

Madanayake, A., L. Bruton and C. Comis FPGA architectures for real-time 2D/3D FIR/IIR plane wave filters. IEEE Intl. Symp. on Circuits and Systems, ISCAS'04.

Madanayake, A. and L. T. Bruton "A Speed-optimized systolic-array processor architecture for spatio-temporal 2D IIR broadband beam filters." IEEE Trans. on. Circuits and Systems: Regular Papers **55**(7): 1953 - 1966.

Madanayake, H. L. P. A. and L. T. Bruton "A Systolic-array Architecture for First-Order 3D IIR Frequency-planar Filters." IEEE Trans. Circuits and Systems: Regular Papers **55**(6): 1546-1559.

Madanayake, H. L. P. A. and L. T. Bruton (2007). "Low-complexity distributed-parallel-processor for 2D IIR broadband beam plane-wave filters." Canadian Journal of Electrical and Computer Engineering (CJECE) **32**(3): 123-131.

Madanayake, H. L. P. A. and L. T. Bruton (2008). A Real-time Systolic Array Processor Implementation of Two-dimensional IIR Filters for Radio-frequency Smart Antenna Applications. IEEE Intl. Symp. on Circuits and Systems (ISCAS'08), Seattle.

Madanayake, H. L. P. A., S. V. Hum and L. T. Bruton "A Systolic Array 2D IIR Broadband RF Beamformer." IEEE Trans. on Circuits and Systems-II: Express Briefs **55**(12): 1244-1248.

Madanayake, H. L. P. A., S. V. Hum and L. T. Bruton UWB Beamforming Using Digital 2D Frequency-planar Filters. IEEE 2008 Antenna and Propagation Society Symposium/URSI Symposium, San Diego.

Parhi, K. K. "Finite word effects in pipelined recursive filters." IEEE Trans. on Signal Processing, IEEE Trans. on Acoustics, Speech, and Signal Processing. **39**(6): 1450-1454.

Parhi, K. K. "Pipelining in algorithms with quantizer loops." IEEE Trans. on Circuits and Systems **38**(7): 745-754.

Parhi, K. K. (1991). "Finite word effects in pipelined recursive filters." IEEE Trans. on Signal Processing [see also IEEE Trans. on Acoustics, Speech, and Signal Processing] **39**(6): 1450-1454.

Parhi, K. K. (1999). VLSI Digital Signal Processing Systems: Design and Implementation, John Wiley and Sons.

Parhi, K. K. and D. Messerschmitt Look-ahead computation: Improving iteration bound in linear recursions. IEEE Intl. Conf. on Acoustics, Speech and Signal Processing, ICASSP'87.

Parhi, K. K. and D. G. Messerschmitt Pipelined VLSI Recursive Filter Architectures using Scattered Look-Ahead and Decomposition. 1988 IEEE Intl. Conf. on Acoustics, Speech and Signal Processing, ICASSP88, New York, N.Y., USA.

Parhi, K. K. and D. G. Messerschmitt (1989). "Concurrent architectures for two-dimensional recursive digital filtering." IEEE Trans. on Circuits and Systems **36**(6): 813-829.

Rader, C. M. (1996). VLSI Systolic Arrays for Adaptive Nulling. IEEE Signal Processing Magazine. **13:** 29-49.

Ramamoorthy, P. A. and L. T. Bruton "Design of stable two-dimensional analog and digital filters with applications in image processing." Int. J. Circuit Theory Appl. **7**: 229-245.

Rodenbeck, C. T., K. Sang-Gyu, T. Wen-Hua, et al. (2005). "Ultra-wideband low-cost phased-array radars." Microwave Theory and Techniques, IEEE Transactions on **53**(12): 3697-3703.

Schroeder, H. and H. Blume (2000). One- and Multidimensional Signal Processing-Algorithms and Applications in Image Processing, John Wiley and Sons, Ltd.

Shanbhag, N. R. (1991). "An improved systolic architecture for 2-D digital filters." Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on] **39**(5): 1195-1202.

Sid-Ahmed, M. A. "A systolic realization for 2-D digital filters." IEEE Trans. on Acoustics, Speech, and Signal Processing **37**(4): 560-565.

Silva, R., R. Worrel and A. Brown Reprogrammable, Digital Beam Steering GPS Receiver Technology for Enhanced Space Vehicle Operations. Core Technologies for Space Systems Conference, Colorado Springs, CO.

Staderini, E. M. (2002). "UWB radars in medicine." Aerospace and Electronic Systems Magazine, IEEE **17**(1): 13-18.

Sundarajan, V. and K. K. Parhi Synthesis of folded multidimensional DSP systems. IEEE Intl. Symp. on Circuits and Systems (ISCAS'98).

Van Ardenne, A. (2000). Concepts of the Square Kilometre Array; toward the new generation radio telescopes. IEEE 2000 Intl. Symp. on Antennas and Propagation.

Weem, J. P., B. M. Noratos and Z. Popovic (1999). Broadband Array Considerations for SKA. Proceedings on Perspectives for Radio Astronomy-- Technologies for Large Antenna Arrays.

Zajc, M., R. Sernec and J. Tasic (2000). Array processors for DSP: implementation considerations. 10th Mediterranean Electrotechnical Conference, 2000, MELECON 2000.

13

A VLSI Architecture for Output Probability Computations of HMM-based Recognition Systems

Kazuhiro Nakamura, Masatoshi Yamamoto,
Kazuyoshi Takagi and Naofumi Takagi
Nagoya University
Japan

1. Introduction

Mobile embedded systems with natural human interfaces, such as speech recognition, lip reading, and gesture recognition, are required for the realization of future ubiquitous computing. Recognition tasks can be implemented either on processors (CPUs and DSPs) or dedicated hardware (ASICs). Although processor-based approaches offer flexibility, real-time recognition tasks using state-of-the-art recognition algorithms exceed the performance level of current embedded processors, and require modern high-performance processors that consume far more power than dedicated hardware. Dedicated hardware, which is optimized for low-power, real-time recognition tasks, is more suitable for implementing natural human interfaces in low power mobile embedded systems. VLSI architectures optimized for recognition tasks with low power dissipation have been developed. Yoshizawa et al. investigated a block-wise parallel processing method for output probability computations of continuous hidden Markof models (HMMs), and proposed a low power, high-speed VLSI architecture. Output probability computations are the most time-consuming part of HMM-based recognition systems. Mathew et al. developed low-power accelerators for the SPHINX 3 speech recognition system, and also developed perception accelerators for embedded systems. In this chapter, we present a fast and memory efficient VLSI architecture for output probability computations of continuous HMMs using a new block-wise parallel processing method. We show block-wise frame parallel processing (BFPP) for output probability computations and present an appropriate VLSI architecture for its implementation. Compared with a conventional block-wise state parallel processing (BSPP) architecture, when there are a sufficient number of HMM states for accurate recognition, the BFPP architecture requires fewer registers and processing elements (*PEs*), and less processing time. The *PEs* used in the BFPP architecture are identical to those used in the BSPP architecture. From a VLSI architectural viewpoint, a comparison shows the efficiency of the BFPP architecture through efficient use of registers for storing input feature vectors and intermediate results during computation. The remainder of this chapter is organized as follows: the structure of HMM based recognition systems is described in

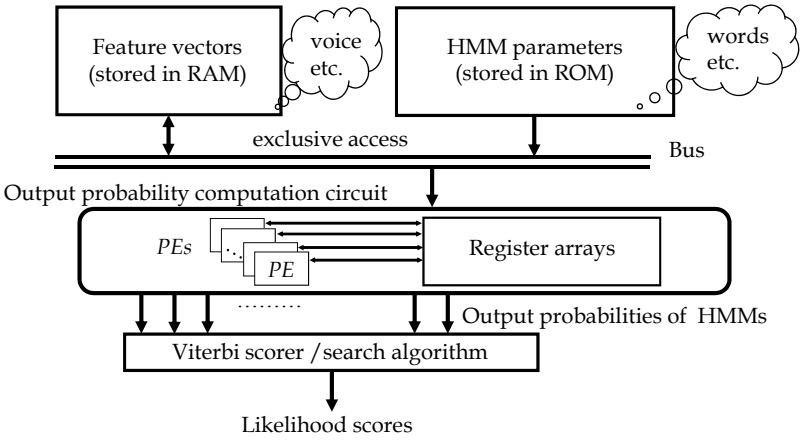


Fig. 1. Basic structure of HMM-based recognition hardware

Section 2, BFPP and BFPP-based VLSI architecture are introduced in Section 3, the evaluation of the BFPP architecture is described in Section 4, and conclusions are presented in Section 5.

2. HMM-based Recognition Systems

2.1 HMM-based Recognition Hardware

Due to their effectiveness and efficiency for user-independent recognition, HMMs are widely used in applications such as speech recognition, lip-reading, and gesture recognition. Figure 1 shows the basic structure of HMM-based recognition hardware (Yoshizawa et al., 2006, Yoshizawa et al., 2004, Yoshizawa et al., 2002, Mathew et al., 2003a). The output probability computation circuit and Viterbi scorer work together as a recognition engine. The inputs to the output probability computation circuit are feature vectors of several dimensions and model parameters of HMMs. These values are stored in RAM and ROM respectively. The RAM, ROM and output probability computation circuit interconnect via a single bus, and memory accesses are exclusive. The output probability computation circuit outputs the results of the output probability computation of HMMs. The Viterbi scorer outputs likelihood score using the Viterbi algorithm. In HMM-based recognition systems, the most time-consuming task is output probability computations, and the output probability computation circuit accelerates these computations. The output probability computation circuit has several register arrays and processing elements (*PEs*) for efficient high-speed parallel processing.

2.2 Output Probability Computation of HMMs

Ret $\mathbf{O}_1$, $\mathbf{O}_2$, ..., $\mathbf{O}_T$ be a sequence of P-dimensional input feature vectors (input frames) to HMMs, where $\mathbf{O}_t = (o_{t1}, o_{t2}, ..., o_{tP})$, $1 \le t \le T$. T is the number of input feature vectors, and P is the dimension of the input feature vector. For an input frame $\mathbf{O}_t$, the output probability of N-state left-to-right continuous HMM at the j-th state is given by

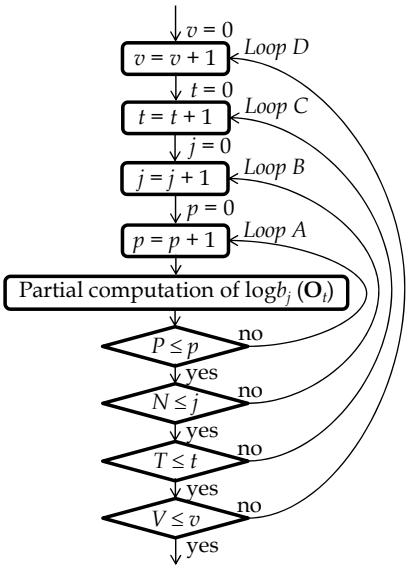


Fig. 2. Flowchart of output probability computation

$$\log b_j(\mathbf{O}_t) = \omega_j + \sum_{p=1}^{P} \sigma_{jp}\left(o_{tp} - \mu_{jp}\right)^2, \ 1 \le j \le N, \ 1 \le t \le T, \tag{1}$$

where ω_j, σ_{jp} and μ_{jp} are the factors of the Gaussian probability density function (Yoshizawa et al., 2006).

The output probability computation circuit computes $\log b_j(\mathbf{O}_t)$ based on Eq. (1), where all HMM parameters ω_j, σ_{jp} and μ_{jp} are stored in ROM, and the input frames are stored in RAM. The values of T, N, P and the number of HMMs V differ for each recognition system. For a recent isolated word recognition system (Yoshizawa et al., 2006, Yoshizawa et al., 2004), T, N, P and V are 86, 32, 38, and 800, respectively, and for another word recognition system (Yoshizawa et al., 2002), T, N, P and V are 89, 12, 16 and 100 respectively. For a continuous speech recognition system (Mathew et al., 2003a), T, N, P and V are approximately 20, 10, 40, and 50, respectively. Different applications require different output probability computation circuit architectures. A flowchart of output probability computations for V HMMs is shown in Fig. 2. Output probabilities are obtained by $T \times N \times P \times V$ times the partial computation of $\log b_j(\mathbf{O}_t)$. Partial computation of $\log b_j(\mathbf{O}_t)$ performs four arithmetic operations, a subtraction ($a = o_{tp} - \mu_{jp}$), an addition ($acc = acc + b$, where the initial value of acc is ω_j), and two multiplications ($b = a \times a \times \sigma_{jp}$) for Eq. (1), and computes $\log b_j(\mathbf{O}_t)$.

3. Fast and memory efficient VLSI architecture

3.1 Block-wise frame parallel processing

Block parallel processing (BPP) for output probability computations was proposed as an efficient parallel processing method for word HMM-based speech recognition by Yoshizawa

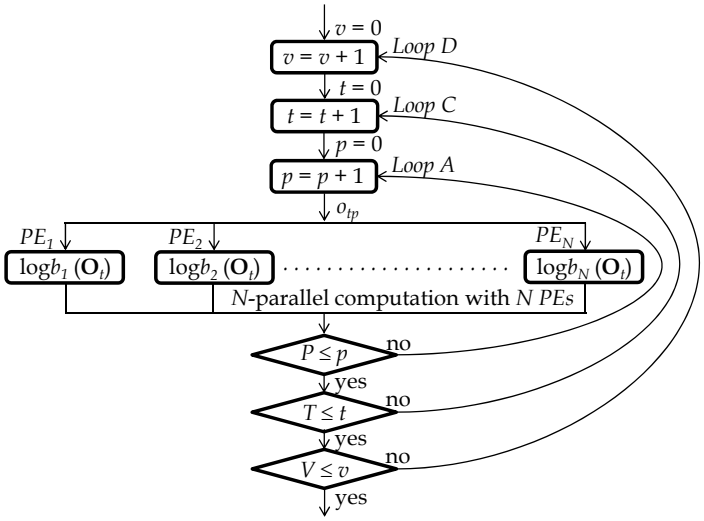


Fig. 3. Flowchart of output probability computation using BSPP

et al. (Yoshizawa et al., 2006, Yoshizawa et al., 2004, Yoshizawa et al., 2002). In this method, the set of input frames is called a *block*, and HMM parameters are effectively shared between different input frames in the computation. *N*-parallel computation is performed by their BPP. In this chapter, we classify two types of BPP according to data flow of output probability computations: block-wise frame parallel processing (BFPP) and block-wise state parallel processing (BSPP). A block can be seen as a set of M ($\leq T$) input frames, whose elements are $\mathbf{O}_{t'}$'s, $1 \leq t' \leq M$. M frames in T input frames are processed in block. *BFPP* performs arithmetic operations to locally stored input frames, which are $\mathbf{O}_1$, $\mathbf{O}_2$, ..., $\mathbf{O}_M$, and output probability computations for multiple frames are carried out simultaneously. On the other hand, a block can also be seen as a $M \times P$ matrix whose elements are $o_{t'p}$, $1 \leq t' \leq M$, $1 \leq p \leq P$. *BSPP* performs arithmetic operations to an input sequence, which is o_{11}, ..., o_{1P}, o_{21}, ..., o_{2P}, ..., o_{M1}, ..., o_{MP}, and output probability computations for multiple states are carried out simultaneously.

The BPP proposed by Yoshizawa et al. (Yoshizawa et al., 2006, Yoshizawa et al., 2004, Yoshizawa et al., 2002) is classified as a BSPP. In this chapter, we present BFPP for output probability computations. $M/2$-parallel computations are performed by our BFPP.

A flowchart of the output probability computations with the conventional BSPP (Yoshizawa et al., 2006, Yoshizawa et al., 2004, Yoshizawa et al., 2002) is shown in Fig. 3.

PE_i represents the i-th processing element, which computes $\log b_i(\mathbf{O}_t)$ by a subtraction, an addition, and two multiplications for Eq. (1). *Loop B* (Fig. 2) is expanded as shown in Fig. 3, and $\log b_1(\mathbf{O}_t)$, $\log b_2(\mathbf{O}_t)$, ..., and $\log b_N(\mathbf{O}_t)$ are computed simultaneously with *N PEs*, where o_{tp} is fed to the *N PEs* in *Loop A*. In addition to the N-state parallel computation, the same HMM parameters μ_{jp}'s and σ_{jp}'s, and ω_j's, $1 \leq j \leq N$, $1 \leq p \leq P$, are used repeatedly during *Loop C* in Fig. 3.

A flowchart of the output probability computation with BFPP is shown in Fig. 4. The *PEs* in

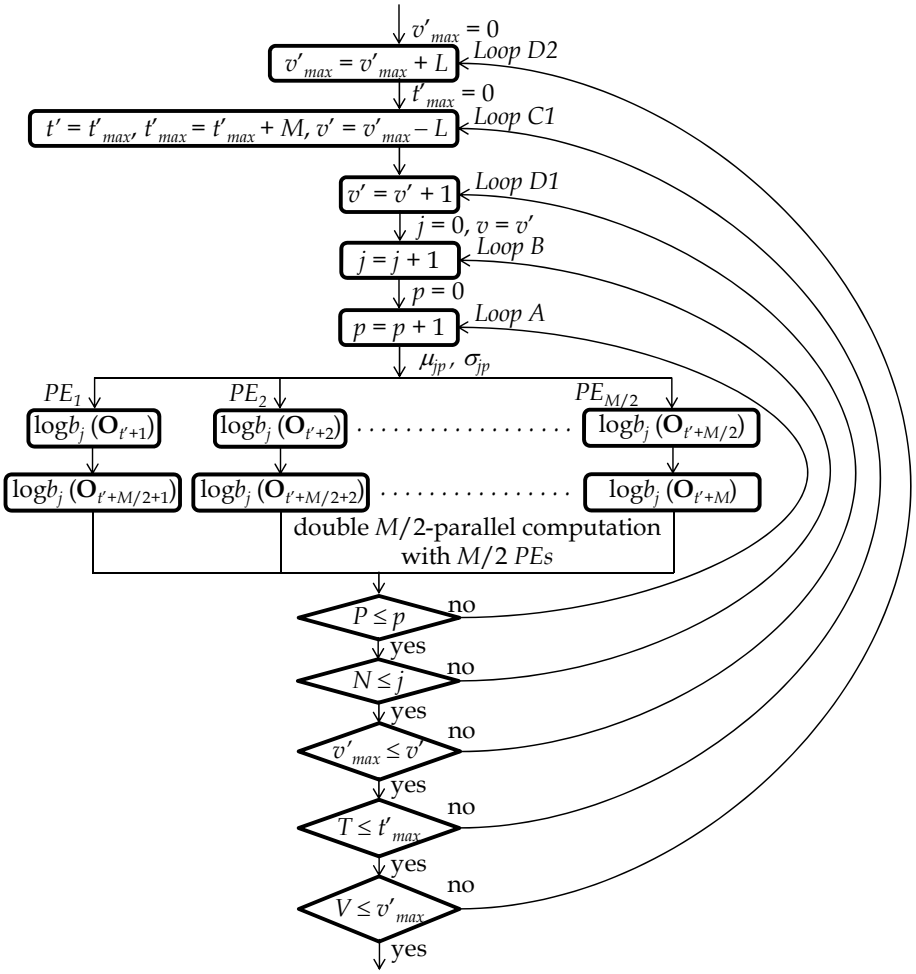


Fig. 4. Flowchart of output probability computation using BFPP

Figs. 4 and 3 are identical, but in a different number. *Loop C* in Fig. 2 is partially expanded in Fig. 4, and $\log b_j(\mathbf{O}_{t'+1})$, $\log b_j(\mathbf{O}_{t'+2})$, ..., and $\log b_j(\mathbf{O}_{t'+M/2})$ are computed simultaneously with $M/2$ *PEs* in *Loop C1*, where μ_{jp} and σ_{jp} are fed to the $M/2$ *PEs* in *Loop A*. In addition to the $M/2$-frame parallel computations, $\log b_j(\mathbf{O}_{t'+M/2+1})$, $\log b_j(\mathbf{O}_{t'+M/2+2})$, ..., and $\log b_j(\mathbf{O}_{t'+M})$ are also computed with the same $M/2$ *PEs*. In this double $M/2$-parallel computation, the same HMM parameters μ_{jp} and σ_{jp} are used twice, because the parameters are independent of t. In addition to the $M/2$-parallel computations, *Loop D* (Fig. 2) is divided into *Loops D1* and *D2* (Fig. 4). The same input frames $\mathbf{O}_{t'+1}$, $\mathbf{O}_{t'+2}$, ..., and $\mathbf{O}_{t'+M}$ are used repeatedly during *Loop D1*, because the input frames are independent of v.

3.2 A VLSI architecture for output probability computation

Our BFPP VLSI architecture for output probability computations is shown in Fig. 5. The architecture consists of five register arrays and $M/2$ *PEs*. *Reg***O** stores M input frames $\mathbf{O}_{t'+1}$, $\mathbf{O}_{t'+2}$, ..., $\mathbf{O}_{t'+M}$. *Reg*μ and *Reg*σ store HMM parameters $-\mu_{jp}$, and σ_{jp}, respectively. *Reg*ω stores

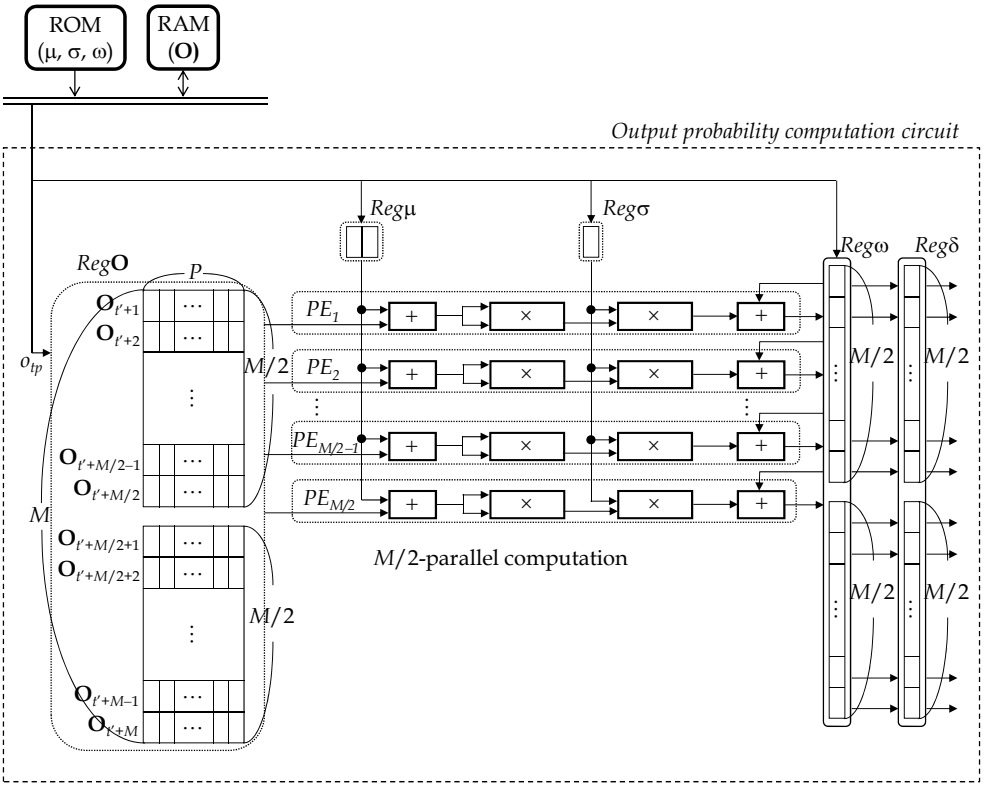


Fig. 5. BFPP VLSI architecture

HMM parameter ω_j and intermediate results. *Reg*δ stores computed output probabilities for a Viterbi scorer. Each PE_i consists of two adders and two multipliers, which are used for computing $\omega_j + \sum_{p=1}^{P} \sigma_{jp}\left(o_{t'p} - \mu_{jp}\right)^2$.

Figure 6 shows the flowchart of output probability computations using the BFPP architecture. The computation starts by reading M input frames from RAM and storing them to *Reg***O** in *Loop C1*, which are $\mathbf{O}_{t'+1}$, $\mathbf{O}_{t'+2}$, ..., $\mathbf{O}_{t'+M/2}$, $\mathbf{O}_{t'+M/2+1}$, $\mathbf{O}_{t'+M/2+2}$, ..., $\mathbf{O}_{t'+M}$. The HMM parameters of v-th HMM are read from ROM and stored in *Reg*μ, *Reg*σ and *Reg*ω, which are μ_{11}, σ_{11}, and ω_1. The value of all registers in *Reg*ω is set to ω_1. For the first half of the stored input frames $\mathbf{O}_{t'+1}$, $\mathbf{O}_{t'+2}$, ..., and $\mathbf{O}_{t'+M/2}$, $M/2$ intermediate results are simultaneously computed with the stored μ_{11}, σ_{11}, and ω_1 by $M/2$ *PE*s, where the HMM parameters are shared by all *PE*s. At the same time, an HMM parameter $\mu_{j\,p+1}$ of v-th HMM

is read from ROM and stored in *Reg*μ. Then, for the other half of the stored input frames $\mathbf{O}_{t'+M/2+1}$, $\mathbf{O}_{t'+M/2+2}$, ..., and $\mathbf{O}_{t'+M}$, $M/2$ intermediate results are simultaneously computed with the same μ_{11}, σ_{11}, and ω_1 by $M/2$ *PEs*. At the same time, an HMM parameter $\sigma_{j\,p+1}$ of v-th HMM is read from ROM and stored in *Reg*σ. In this double $M/2$-parallel computation, the

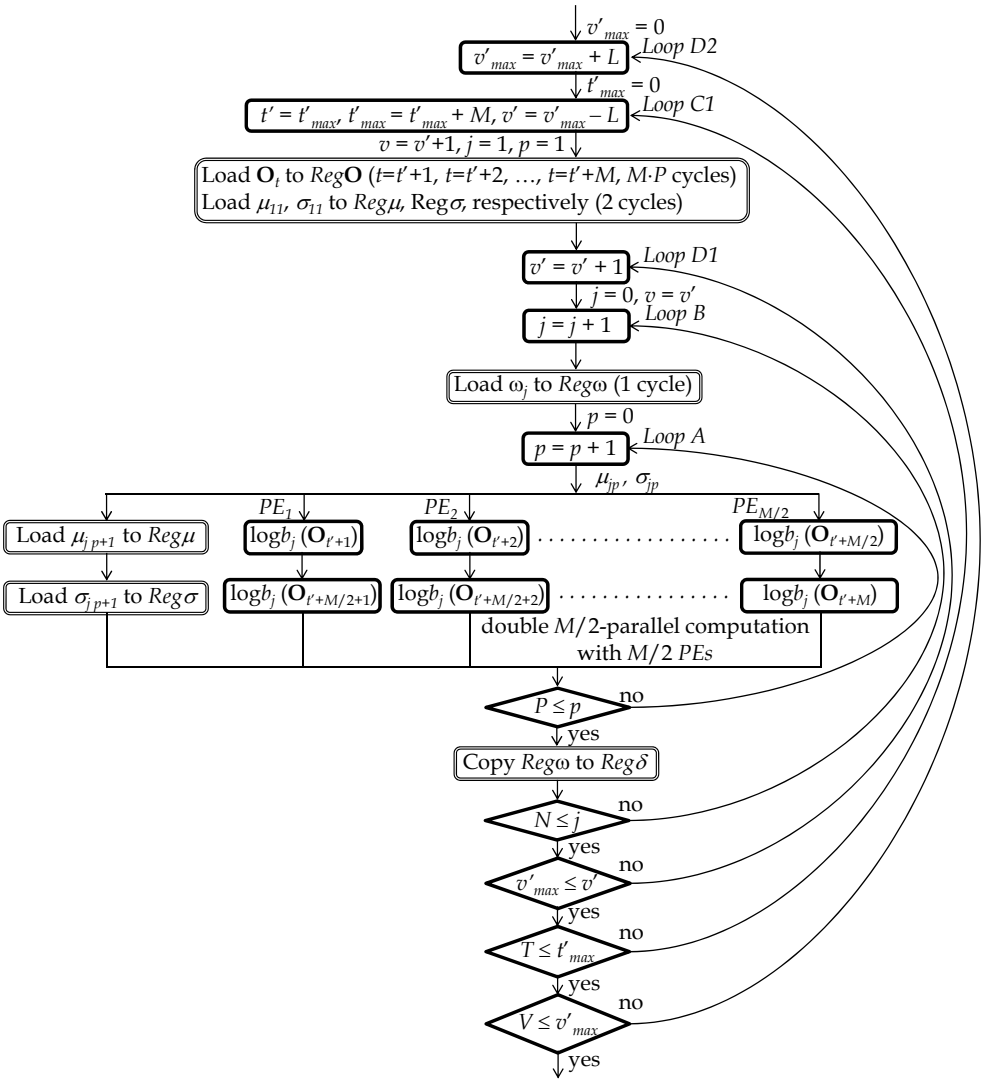


Fig. 6. Flowchart of computations using the BFPP architecture

same HMM parameters μ_{11}, σ_{11}, and ω_1 are used twice. In the next double $M/2$-parallel computation, the stored HMM parameters $\mu_{j\,p+1}$ and $\sigma_{j\,p+1}$ are used twice. M output probabilities $\log b_j(\mathbf{O}_{t'+1})$, $\log b_j(\mathbf{O}_{t'+2})$, ..., and $\log b_j(\mathbf{O}_{t'+M})$ of v-th HMM are obtained by *Loop*

A. The obtained results are transfered from *Reg*ω to *Reg*δ for starting the next output probability computation, $\log b_{j+1}(\mathbf{O}_{t'+1})$, $\log b_{j+1}(\mathbf{O}_{t'+2})$, ..., $\log b_{j+1}(\mathbf{O}_{t'+M})$ of v-th HMM. The stored results are fed to the Viterbi scorer. The $M{\cdot}N$ output probabilities of v-th HMM are obtained by *Loop B*. $M{\cdot}N{\cdot}L$ output probabilities of HMM $v' + 1$, $v' + 2$, ..., $v' + \mathrm{L}$ are obtained by *Loop D1* with the same M input frames $\mathbf{O}_{t'+1}$, $\mathbf{O}_{t'+2}$, ..., and frames $\mathbf{O}_{t'+1}$, $\mathbf{O}_{t'+2}$, ..., and $\mathbf{O}_{t'+M.}$

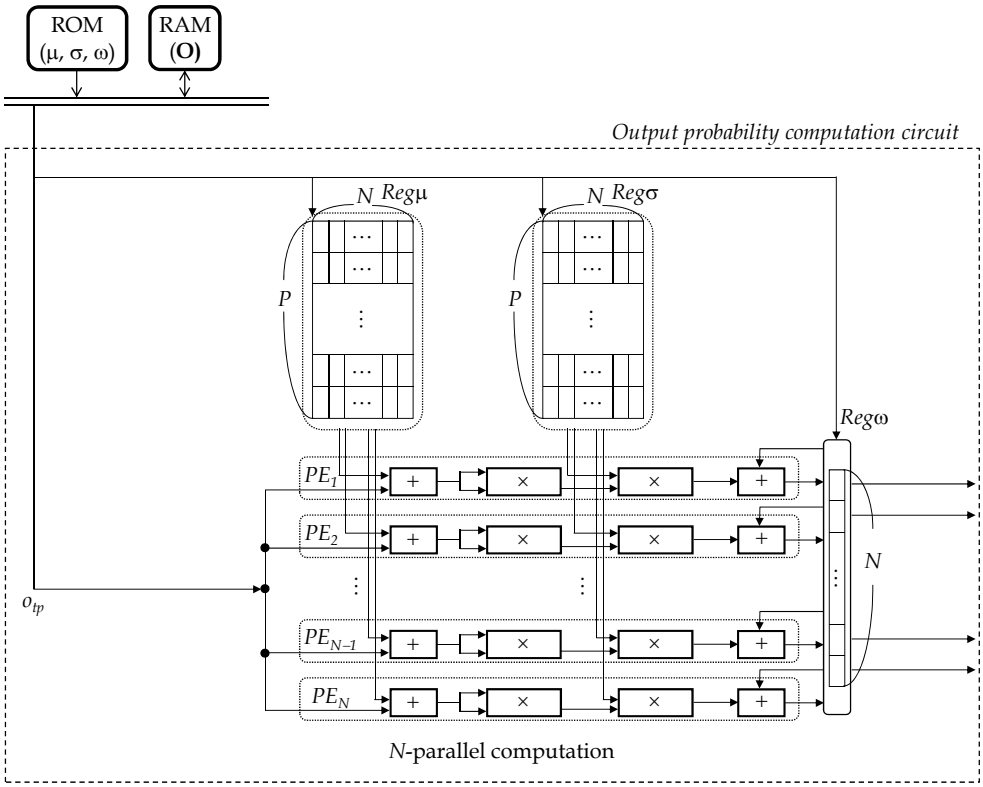


Fig. 7. BSPP VLSI architecture

frames $\mathbf{O}_{t'+1}$, $\mathbf{O}_{t'+2}$, ..., and $\mathbf{O}_{t'+M.}$ The $M{\cdot}N{\cdot}L{\cdot}(T/M)$ output probabilities of HMM $v' + 1$, $v' + 2$, ..., $v' + L$ are obtained by *Loop C1*, and finally the $M{\cdot}N{\cdot}L{\cdot}(T/M){\cdot}(V/L)$ output probabilities of all HMMs are obtained by *Loop D2*.

4. Evaluation

We compared the proposed BFPP with BSPP (Fig. 7) VLSI architecture (Yoshizawa et al., 2006, Yoshizawa et al., 2004, Yoshizawa et al., 2002). The BSPP architecture consists of three register arrays and *N PEs*. *Reg*μ and *Reg*σ store HMM parameters $-\mu_{jp}$ and σ_{jp}, respectively, and *Reg*ω stores HMM parameter ω_j and intermediate results. The *PEs* in Figs. 7 and 5 are identical.

Figure 8 shows the flowchart of the computations of BSPP architecture. The computation starts by reading all $2{\cdot}N{\cdot}P + N$ HMM parameters of v-th HMM from ROM and storing them to *Reg*μ, *Reg*σ, and *Reg*ω in *Loop D*. For input o_{tp}, the intermediate results are computed with stored HMM parameters by N *PEs*. N output probabilities $\log b_1(\mathbf{O}_t)$, $\log b_2(\mathbf{O}_t)$, ..., $\log b_N(\mathbf{O}_t)$ of the HMM are obtained by *Loop A*. The obtained results are fed to a Viterbi scorer. $N{\cdot}T$

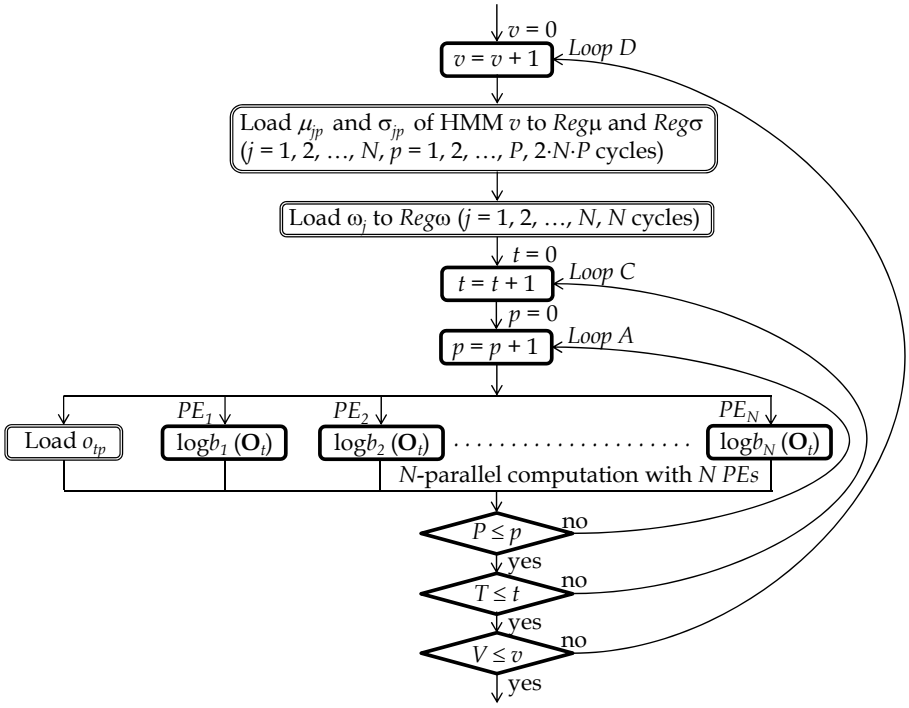


Fig. 8. Flowchart of computations using the BSPP architecture

	Register size (bit)
BFPP (ours)	$P{\cdot}M{\cdot}x_o + 2{\cdot}x_\mu + x_\sigma + 2{\cdot}M{\cdot}x_f$
BSPP	$N{\cdot}P{\cdot}x_\mu + N{\cdot}P{\cdot}x_\sigma + N{\cdot}x_f$

Table 1. Register size

	Processing time (cycles)
BFPP (ours)	$\lceil V/L \rceil{\cdot}\{P{\cdot}M + (1 + 2{\cdot}P){\cdot}L{\cdot}N\}{\cdot}\lceil T/M \rceil$
BSPP	$V{\cdot}(2{\cdot}N{\cdot}P + N + P{\cdot}T)$

Table 2. Processing time

output probabilities of v-th HMM are obtained by *Loop C* with the same HMM parameters. The $N{\cdot}T{\cdot}$V output probabilities of all HMMs are obtained by *Loop D*.

Table 1 shows the register size of the BSPP and BFPP architectures, where x_μ, x_σ, x_o, and x_f represent the bit length of μ_{jp}, σ_{jp}, o_{tp}, and the output of *PE*, respectively. N, P, and M are the

number of HMM states, the dimension of input feature vector (frame), and the number of input frames in a block, respectively.
Table 2 shows the processing time for computing output probabilities of V HMMs with the BFPP and BSPP architectures, where T and L are the number of input frames and the number of HMMs whose output probabilities are computed with the same input frames during *Loop D1* of Fig. 6, respectively.

	Register size (bit)	Processing time (cycles)	#*PEs*
BFPP (ours)	15,512	4,477,440	22
BSPP	20,224	4,585,600	32

Table 3. Evaluation of the BSPP and BFPP performance

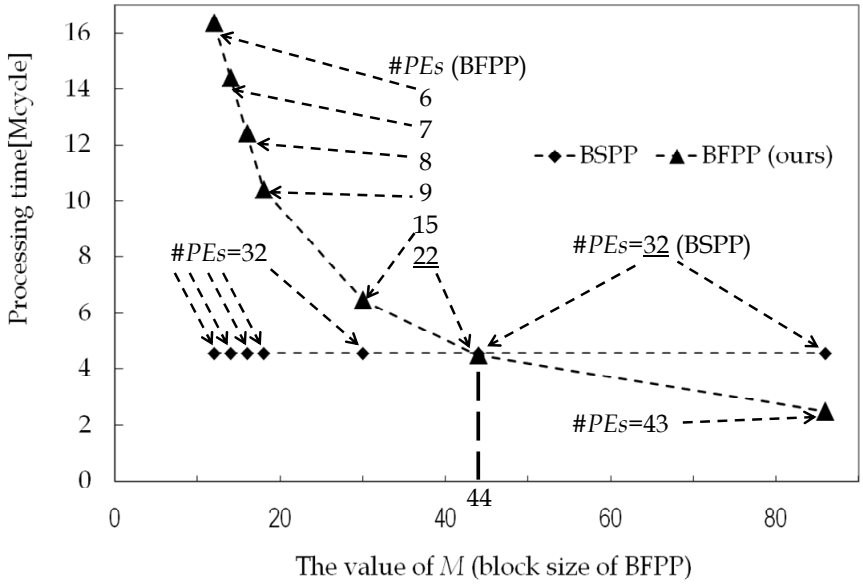


Fig. 9. Evaluation of the BSPP and BFPP performance, and the value of M of the BFPP (N = 32, P = 38, T = 86, V = 800)

Table 3 shows the register size, the processing time, and the number of *PEs* for computing output probabilities of 800 HMMs, where we assume that N = 32, P = 38, T = 86, x_μ = 8, x_σ = 8, x_o = 8, x_f = 24, and V = 800, the same values used in a recent circuit design for isolated word recognition (Yoshizawa et al., 2006, Yoshizawa et al., 2004). We also assume that M = 44 and L = 5 for the BFPP architecture. The *PEs* used in the BSPP and BFPP architectures are identical. Compared with the BSPP architecture, the BFPP architecture has fewer registers, requires less processing time, and has fewer *PEs*. From the VLSI architecture viewpoint, this is because the register size of the BFPP architecture is independent of N, and its *PEs* can repeatedly use the same input frames. The BFPP architecture has fewer wait cycles for

reading data from ROM before parallel computations, 586,240 ($\lceil V/L \rceil \cdot (P \cdot M + L \cdot N) \cdot \lceil T/M \rceil$), than the BSPP architecture, which has 1,971,200 ($V \cdot (2 \cdot N \cdot P + N)$).
Fig. 9 shows the processing time and the number of *PEs* of the BFPP and BSPP architectures, and the value of M of the BFPP architecture. The processing time and the number of *PEs* of the BFPP architecture are less than those of the BSPP architecture when $M = 44$ (Fig. 9).
From a logic design viewpoint, the register arrays of the BSPP and BFPP architectures are designed with Flip-Flops or on-chip multi-port memories of different sizes. Data paths are designed with identical *PEs*, but in a different number. The control paths of these architectures are designed, as shown in the flowcharts Figs. 8 and 6. The data path delay is the same for both the BSPP and BFPP designs, equal to the delay time of one *PE*. The delay times of control paths differ between the two, but the control path delay is small compared with the data path delay.

5. Conclusions

We presented BFPP for output probability computations and presented an appropriate VLSI architecture for its implementation. BFPP performs arithmetic operations to locally stored input frames, and output probability computations for multiple frames are carried out simultaneously. Compared with the conventional BSPP architecture, when the number of HMM states is large enough for accurate recognition, the BFPP architecture requires fewer registers and *PEs*, and less processing time. In terms of the VLSI architecture, a fast and memory efficient VLSI architecture for output probability computations of HMM-based recognition systems has been presented. A logic design, a Viterbi scorer for the BFPP architecture, and a reconfigurable architecture for both the BSPP and BFPP architectures are our future works.

6. References

B. Mathew, A. Davis & Z. Fang (2003a). Perception Coprocessors for Embedded Systems, *Proc. of Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia),* pp. 109-116, 2003.

B. Mathew, A. Davis & Z. Fang (2003b). A Low-Power Accelerator for the SPHINX 3 Speech Recognition System, *Proc. of Int'l Conf. on Compilers, Architecture and Synthesis for Embedded Systems,* pp. 210-219, 2003.

S. Yoshizawa, Y. Miyanaga & N. Yoshida (2002). On a High-Speed HMM VLSI Module with Block Parallel Processing, *IEICE Trans. Fundamentals (Japanese Edition),* Vol. J85-A, No. 12, pp. 1440-1450, 2002.

S. Yoshizawa, N. Wada, N. Hayasaka & Y. Miyanaga (2004). Scalable Architecture for Word HMM-Based Speech Recognition, *Proc. of 2004 IEEE Int'l Symposium on Circuits and Systems (ISCAS'04),* pp. 417-420, 2004.

S. Yoshizawa, N. Wada, N. Hayasaka & Y. Miyanaga (2006). Scalable Architecture for Word HMM-Based Speech Recognition and VLSI Implementation in Complete System, *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS,* Vol. 53, No. 1, pp. 70-77, 2006.

X. Huang, F. Alleva, H. W. Hon, M. Y. Hwang, K. f. Lee & R. Rosenfeld (1992). The SPHINX-II speech recognition system: an overview, *Computer Speech and Language,* Vol. 7(2), pp. 137-148, 1992.

14

Efficient Built-in Self-Test for Video Coding Cores: A Case Study on Motion Estimation Computing Array

Chun-Lung Hsu, Yu-Sheng Huang and Chen-Kai Chen
Department of Electrical Engineering, National Dong Hwa University
Taiwan, R.O.C

1. Introduction

In more recent years, multimedia technology applications have been becoming more flexible and powerful with the development of semiconductors, digital signal processing (DSP), and communication technology. The latest video standard, H.264/AVC/MPEG-4 Part 10 (Advance Video Coding) (Wiegand, 2003), is regarded as the next generation video compression standard (VCS). For video compression standards, the motion estimation computing array (MECA) is the most computationally demanding component in a video encoder/decoder (Kuhn, 1999; Komarek and Pirsch 1989). It is known that about 60-90% of the total of computation time is consumed in motion estimation. Additionally, the motion estimation algorithms used also profoundly influences the visual quality of reconstructed images. More accurate predictions increase the compression ratio and improve peak signal-to-noise ratio (PSNR) at a given bit-rate. Since the motion estimation algorithm is not specified in the video coding standards that many algorithms are applied to with different hardware platform, system frequencies, operating voltage and power dissipation.

On the other hand, due to the rapid advance in semiconductor fabrication technology, a large number of transistors can be integrated on a single chip. However, integrating large number of processor on a single chip results in the increase in the logic-per-pin ratio, which drastically reduces the controllability and observability of the logic on the chip. Consequently, testing such highly complex and dense circuits become very difficult and expensive (Lu et al., 2005).

For a commercial chip, the VCS must introduce design-for-testability (DFT), especially in an MECA. The objective of DFT is to increase the ease with which a device can be tested to guarantee high system reliability. Many DFT approaches have been developed such as *ad hoc*, *structured*, and *built-in self-test* (BIST) (Wu et al., 2007; Nagle et al., 1989; McCluskey, 1985; Kung et al., 1995; Touba and McCluskey, 1997). Among these DFT approaches, BIST has an obvious advantage in that reduces the need for testing of expensive test equipment, since the circuit/chip and its tester are implemented in the same circuit/chip. In a word, the BIST can generate test simulations and analyze test responses without outside support, making tests and diagnoses of digital systems quick and effective.

Thus, this chapter proposed a minimal performance penalty BIST design with significantly smaller area overhead. In normal mode, the BIST circuit does not be performed and do not deliver the test pattern to AD-PEs for testing. Thus, each AD-PE in MECA performs its normal operation to determine the SAD values. In testing mode, the BIST circuits, comprise test pattern generator (TPG) and output response analyzer (ORA), are performed to testing itself. In terms of results, after employing the presents BIST design, the circuit guarantee 100% fault coverage with low test application time at low area overhead. Moreover, the experimental results prove the effectiveness and value of this work.

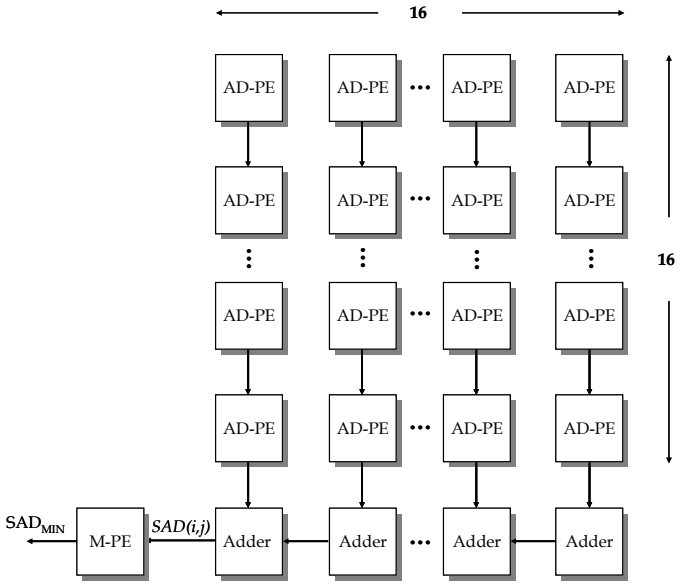
2. Motion estimation computing array

The motion estimation process in any VCS comprises a search strategy to determine the motion vectors (MVs). The motion vector is to describe the transformation from adjacent frames in the video sequences. On the other hand, the motion estimation also comprises a process to determine a matching cost metric computation such as the sum of absolute differences (SAD). In a word, the search strategy in motion estimation aims at selecting a set of candidates and selects the one with the minimum cost metric. After selecting minimizes the cost metric, it encodes the prediction residual between the original and motion compensated blocks. Each residual block is transformed, quantized, and entropy coded. Additionally, the main objective of motion estimation algorithms is to exploit the temporal redundancy of a segmented video sequences. For example, the full-search block-matching algorithm has been developed that introduced the best results in terms of finding motion vectors. Such algorithms are implemented in two stages, namely the calculation of the sum of absolute differences (SAD) for each displacement vector, followed by method for finding the smallest SAD values. This is summarized by Eq. (1) and (2).

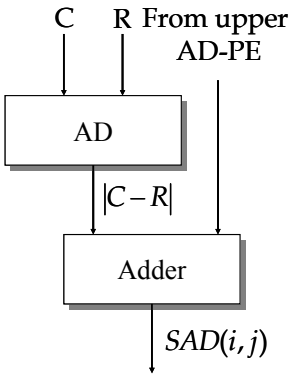
$$\mathrm{SAD}\,(i,j) = \sum_{k=0}^{N-1}\sum_{l=0}^{N-1}\left|C(k,l) \;-\; R(i+k,j+l)\right| \tag{1}$$

$$\mathrm{SAD}_{\mathrm{MIN}} = \min(\mathrm{SAD}(i,j)) \tag{2}$$

where $C(k,l)$ and $R(i+k,j+l)$ represent the current frame and search region's macroblock, respectively (Pirsch et al., 1995). In the other hand, block-matching in MECA is performed by a sequential exploration of the search region when the computations are performed in parallel (see Fig. 1). In Fig.1 , the MECA is the parallel architecture for computing the SAD value and its corresponding AD-PE structure are shown in Fig.1 (a) and (b), respectively. The purpose of AD-PEs is to store the value of $C(k,l)$ and $R(i+k,j+l)$ and receive the value of corresponding to the current position of the reference in search region. In other word, the AD-PEs perform the processing of the subtraction and the absolute value computation. After that, AD-PE adds the results with the partial result coming from the upper AD-PE (see Fig. 1 (b)). The partial results are added on columns and a linear array of adders performs the horizontal summation of the row sums, and compute $SAD(i,j)$. For each position (i,j) of the reference block, the M-PE checks if the matching cost metric computation, $SAD(i,j)$, is smaller the previous smaller SAD value that updates the smaller SAD value to the register which stores the previous smaller SAD value.



(a)



(b)

Fig. 1. The generic architecture of motion estimation and structure of AD-PE

3. The BIST design and its test strategy

Figure 2 shows the proposed BIST design, which consists of test pattern generator and output response analyzer. The main objective of the proposed BIST design is to effectively test the MECA by itself. Moreover, using the BIST design can drastically reduce the cost of

expensive test equipment and testing time. The targeted fault, test pattern generator and output response analyzer are explicitly described as follows.

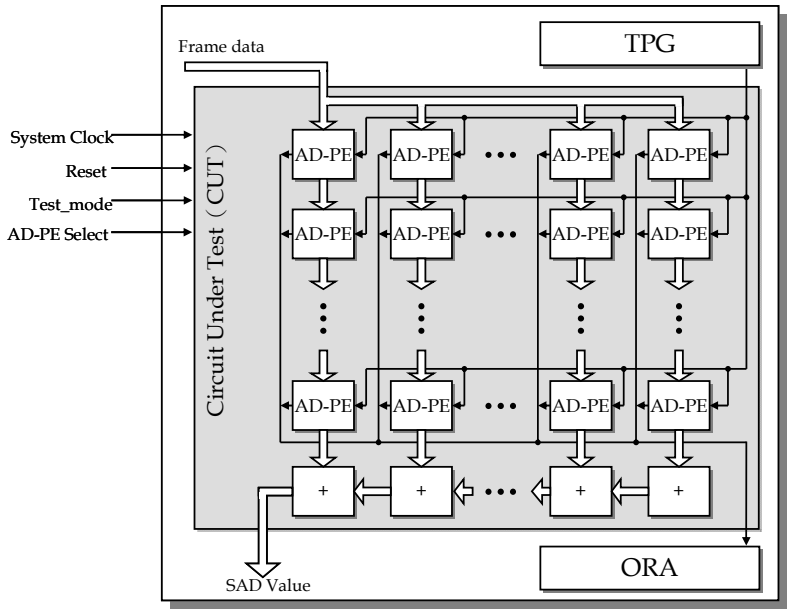


Fig. 2. The proposed BIST design for MECA architecture

3.1 Fault model, AD-PE design and test strategy

In this chapter, the proposed BIST design aimed at the single stuck-at fault (SSAF). In general, considerable physical faults in a circuit/chip are be discussed individually, such as stuck-at fault, bridge fault, open fault, delay fault, crosstalk, etc.. However, the stuck-at fault is classical and most widely fault, which covers 80-90% of possible manufacturing defect in CMOS circuits. Thus, the targeted faults of this work are the single stuck-at faults. On the other hand, the proposed BIST design consists of two major objectives: 1) the targeted fault coverage is first achieved by using efficient BIST design. Meanwhile, the test pattern generators can be implemented with linear-feedback shift register (LFSR). 2) only a few test patterns are required to test the entire MECA architecture.

In addition, to achieve higher controllability and observability for single AD-PE, each AD-PE uses the ripple-carry adders (RCAs) to produce the processing of the absolute difference value computation and addition units, as shown in Fig.3. In Fig. 3, each multiplexers (mux) are designed for testing requirement and are performed normal and test mode by using the signal *sel*. The functions of the absolute difference value computation and addition are designed by using ripple-carry adder. Besides, the frame data delivered the pixel value to RCAs to produce the partial SAD value when the normal mode is performed. Then, the test patterns from test pattern generator are delivered to testing the RCAs while the test mode is selected.

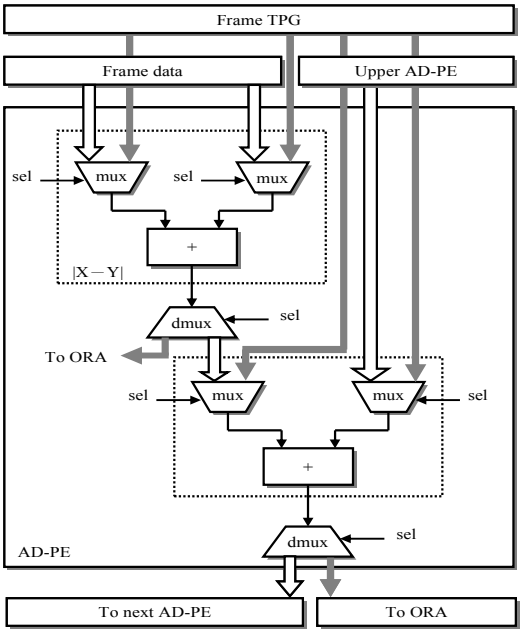


Fig. 3. The internal architecture of AD-PE

In this chapter, the proposed BIST design includes three modes: 1) *Normal mode*: In this mode, each AD-PE performs its normal function, which is to determine the minimum SAD value. 2) *Test mode*: The test pattern generator delivers the test patterns to each AD-PE for testing. Then, the testing output results are compressed by the output response analyzer for testing analysis. 3) *Analyze mode*: In analyze mode, output response analyzer used the compressed signature to determine each AD-PE in MECA architecture that are fault-free or not.

3.2 Test pattern generator

According to the RCA structure, it was comprised by many one bit full-adder (FA). in other word, each AD-PE consists of many one bit full-adder. Consequently, according to the characteristic of one bit full-adder, the FA has three primary inputs. Thus, the test patterns of FA1 has 2^3 different inputs that can input to one bit full-adder for testing requirement. In a word, one bit full-adder can be obtained 100% fault coverage for single stuck-at faults by using test patterns (000, 001, 010, 011, 100, 101, 110, 111). Based on the above-mentioned concept, the Table 1 shows the proposed test patterns for each one bit full-adder in AD-PE which can achieve 100% single stuck-at fault coverage. On the other hand, in order to realize a simple and low-cost test pattern generator, this chapter exploits the LFSR to realize the test pattern generator of BIST design. Figure 4 shows the proposed test pattern generator which can deliver the test patterns to each RCA by using LFSRs. In order to effectively realize, these test patterns can be simplified as segment patterns and generated by a LFSR. The connection between the proposed test patterns generators (8 bits and 12 bits) and RCAs (8

bits and 12 bits) are shown in Fig. 4 and 5, respectively. For example, the test patterns for 8 bits RCA are shown in Table I, which can achieve 100% single stuck-at fault coverage while the LFSRs deliver the test patterns to RCA for testing. Furthermore, the test pattern generator of 8 bits RCA comprises four LFSR to generate the test pattern for testing requirement. Firstly, LFSR0 generated and delivered the test patterns to three inputs of FA0 in 8 bits RCA that the sequences of test pattern listed as follow: 111→011→001→100→010→101→110. Then, the next stage carry is generated by itself. Secondly, the inputs of FA1 in 8 bits RCA are inputted the test pattern sequences, 00→10→01→10→11→11→01, by LFRS1. Similarly, the LFSR2 and LFSR3 deliver the test pattern sequences, 10→11→11→01→00→10→01 and 11→01→00→10→01→10→11, to the inputs of FA2 and FA3 in 8 bits RCA, respectively. Besides, these inputs, a_n and b_n, need to match up different carry as shown in Table 1. in other words, the propagations of test pattern are generated by LFSRs as shown in Fig. 6.

TEST	Test patterns																	
TP_1	1	1	1	0	0	1	0	1	1	0	0	1	0	1	1	0	0
TP_2	0	1	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0
TP_3	0	0	1	0	1	1	1	0	0	0	1	1	1	0	0	0	1
TP_4	1	0	0	1	0	0	1	1	0	1	0	0	1	1	0	1	0
TP_5	0	1	0	1	1	0	0	0	1	1	1	0	0	0	1	1	1
TP_6	1	0	1	1	1	1	0	1	0	1	1	1	0	1	0	1	1
TP_7	1	1	0	0	1	0	1	1	1	0	1	0	1	1	1	0	1

Table 1. 17-bit test pattern for 8-bit RCA

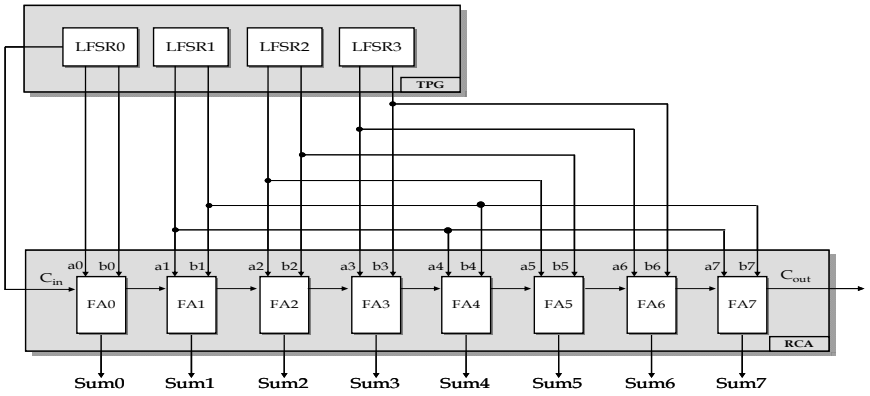


Fig. 4. The connection between the proposed test pattern generator and 8 bits RCA

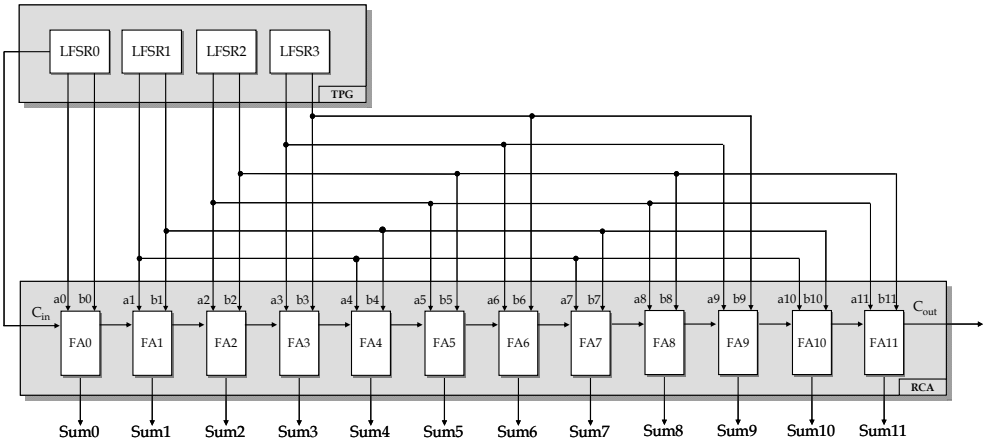
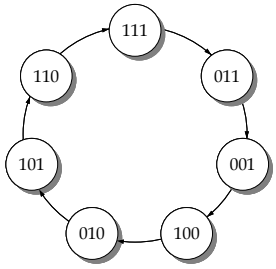
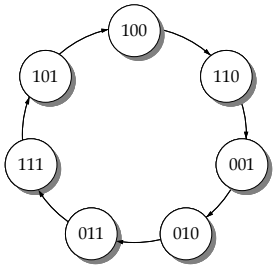


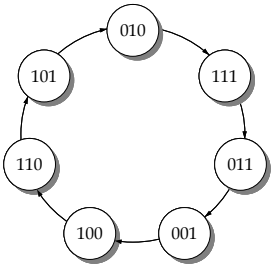
Fig. 5. The proposed test pattern generator for 12 bits RCA



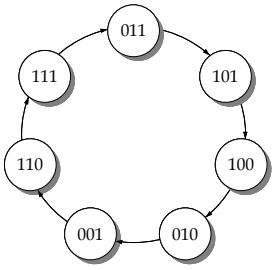
(a) LFSR 0



(b) LFSR 1



(c) LFSR 2



(d) LFSR 3

Fig. 6. The test patterns propagation of 8 bits RCA

3.3 Output response analyzer

Multi-Input Shift Register (MISR) is widely used as the signature analyzer to compact the output response of the circuit under test (CUT). Thus, this chapter exploits the MISR to analyze the testing result. Figure 7 shows the output response analyzer (ORA) that the ORA of the two adders can be implemented by MISR and adding some extra logics. In test mode, the MISR compresses the output responses and the final result of the MISR to serve as the

signature which is used to check and determine whether the CUT is fault-free or not. In a word, testing with signature analyzer, MISR, has the merits of simplicity and low cost hardware because MIRS does not need to store the entire responses of test patterns. The MISR of ORA has two kinds of input data which are from AD-PE output data through some XOR gates and the test patterns (see Fig. 7). On the other hand, the results are checked by MISR and propagated the final result to the last model.

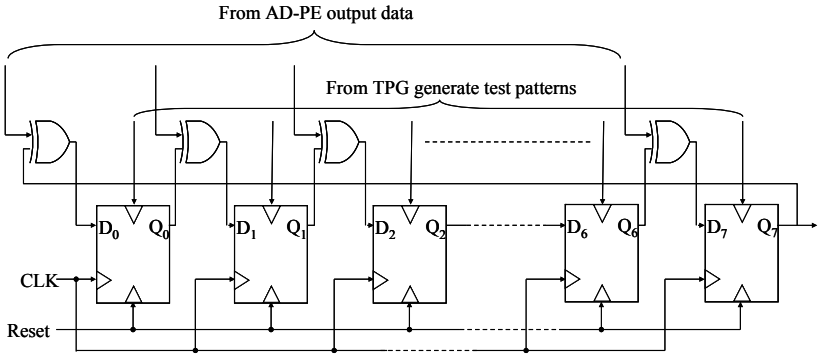


Fig. 7. The output response analyzer for 8 bits RCA

3.4 Test strategy

Based on the discussion of each sub-circuit (AD-PE, TPG and ORA), the overall working flow of the proposed BIST design is summarized in Fig. 8, and is briefly described as follow.

Step 1: Selecting the operation modes

The operation mode, normal and test mode, are determined by test controller. In normal mode, the CUT circuit performs its normal operation to determine the SAD values, whereas the MECA architecture is tested by itself.

Step 2: Initial setting of BIST design

In test mode, the test patterns of BIST design are generated by TPG which was made by four LFSRs. Then, the TPG delivered the test patterns to each AD-PE for testing.

Step 3: Analyzing the test result of AD-PE

The testing results of each AD-PE analyzed by the ORA until all AD-PE are tested and analyzed completely.

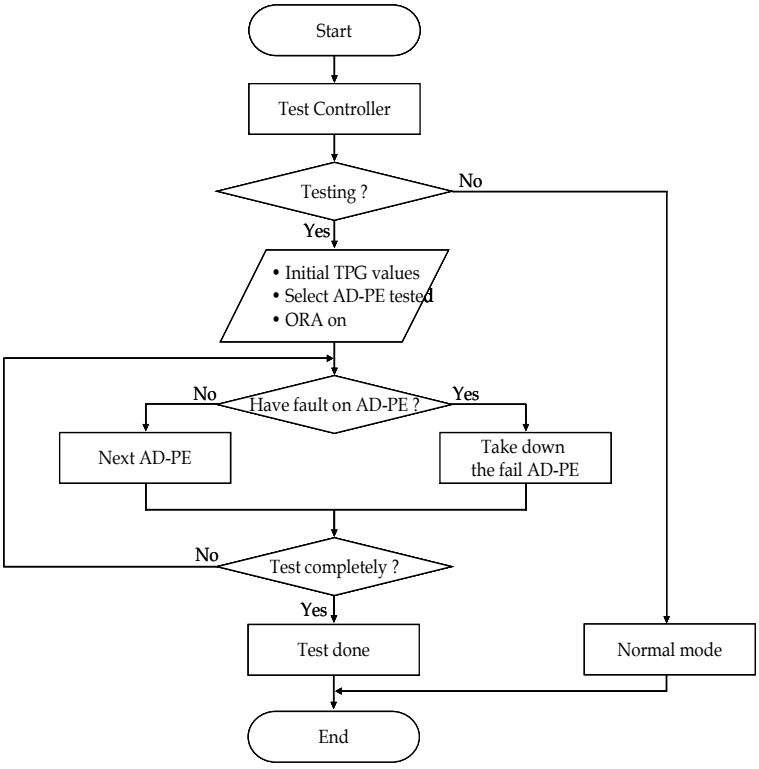


Fig. 8. The overall working flow of the proposed BIST design

4. Results discussion

The proposed BIST design was realized using Verilog HDL and synthesized Design Compiler of Synopsys. The performance comparisons aim at area overhead, fault coverage and test patterns discussion which are also presented here to verify the good performance of the proposed BIST design.

The design is carried out top-down at the gate-level in the system of Quartus II by means of waveform and the design finally passes both the unit test and the integrated test. Figure 9 shows a functionality of the waveforms for AD-PE (as you can see in section II). Figure 10 shows the functionality of test pattern generator. For test mode, initial test patterns are scanned in, an AD-PE is tested, and test responses are scanned out. All related control signals are generated from controller. And, the fault coverage of AD-PE reaches 100% with 7 test patterns.

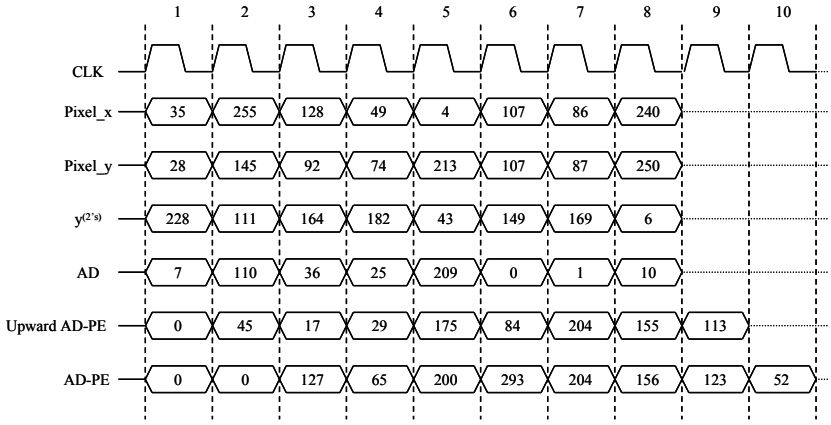


Fig. 9. The logic simulation of AD-PE

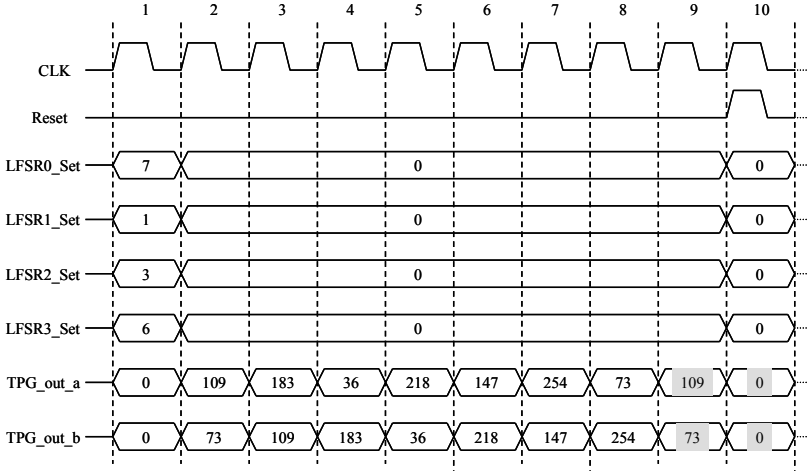


Fig. 10. The waveform for test pattern generator

Comparing with previous work (Li et al., 2004), numbers of test patterns, the pin overheads, test time and fault coverage are listed in Table 2.

By using the proposed BIST design and fault models, the single stuck-at fault coverage of each AD-PE can achieve 100%. This perfectly proves the validity of the test patterns. And, the area overhead of the MECA architecture including BIST is 0.2%, which is tolerable in industry

	Proposed	(Li et al., 2004)
No. of test pattern	8	30
Total test time	35us	13us
Pins overhead	25	1904
Fault coverage	100%	96%

Table 2. Comparisons

5. Conclusions

This chapter describes a BIST design for MECA architecture in video coding systems. In test mode, test patterns are generated by TPG and scanned into each AD-PE of MECA architecture to testing, and test responses are scanned out. All of control signals are generated by the controller. And, experimental results show that the area overhead of the BIST architecture for motion estimation architecture is less than 1%. The fault coverage of each AD-PE can achieve 100%, and it perfectly proves the validity of the test patterns. Moreover, BIST structure can easily be designed and applied to the MECA architecture. That means the simplification in the BIST design of AD-PE in MECA architecture is reasonable.

6. References

Abramovici, M.; Breuer, M. A. & Friedman, A. D. (1990). Digital Systems Testing and Testable Design. Boston, MA: Computer Science Press, 1990.

Gallagher, P.; Chickermane, V.; Gregor, S. & Pierre, T. S. (2001). A Building Block BIST Methodology for SOC Designs: A Case Study, *Proceedings of International Test Conference*, PP. 111-120, Oct. 2001.

Komarek, T. & Pirsch, P. (1989). Array Architectures for Block Matching Algorithms, *IEEE Transactions on Circuits and Systems*, Vol. 36, No. 2, PP. 1301–1308, Oct. 1989.

Kuhn, P. (1999). Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation. New York, NY: Kluwer Academic Publishers, 1999.

Kung, C. P.; Huang, C. J. & Lin, C. S. (1995). Fast Fault Simulation for Bist Applications," *Proceedings of the Fourth Asian Test Symposium*, PP. 93–99, 1995.

Li, D.; HU, M. & Mohamed, O. (2004). Built-In Self-Test Design of Motion Estimation Computing Array, *Proceedings of IEEE Northeast Workshop on Circuits and Systems (NEWCAS'04)*, June 2004.

Lu, S. K.; Shih, J. S. & Huang, S. C. (2005). Design-for-Testability and Fault-Tolerant Techniques for FFT Processors," IEEE Trans. VLSI Systems, vol. 13, no. 6, pp. 732-741, June 2005.

McCluskey, E. J. (1985). Built-In Self-Test Technique, *IEEE Design and Test of Computers*, Vol. 2, No. 2, PP. 29–36, Apr. 1985.

Nagle, H. T.; Roy, S. C.; Hawkins, C. F.; Macnamer, M. G. & Fritzemeier, R. R. (1989). Design for Testability and Built-in Self Test : A review, *IEEE Transactions on Industrial Electronics*, Vol. 36, No. 2, PP. 129–140, May. 1989.

Pirsch, P.; Demassieux, N. & Gehrke, W. (1995). Vlsi architecture for video compression-a survey, *Proceedings of the IEEE*, No. 2, P. 220, Feb. 1995.

Touba, N. A. & McCluskey, E. J. (1997). Pseudo-Random Pattern Testing of Bridging Faults, *Proceeding sof International Conference on Computer Design* , PP. 54–60, 1997.

Wiegand, T. (2003) Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264/ ISO/ IEC 14496-10 AVC), Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Mar. 2003.

Wu, T. H.; Tsai, Y. L.; & Chang, S. J. (2007). An Efficient Design-for-Testability Scheme for Motion Estimation in H.264/AVC, *Proceedings of International Symposium VLSI Design Automation and Test*, PP.25-27, Apr. 2007.

15

SOC Design for Speech-to-Speech Translation

Shun-Chieh Lin[1], Jia-Ching Wang[2], Jhing-Fa Wang[3],
Fan-Min Li[4] and Jer-Hao Hsu[5]
[1]*Industrial Technology Research Institute*
[2]*National Central University*
[3, 4, 5]*National Cheng Kung University*

1. Introduction

In today's globalised world, information exchange between different languages is indispensable. Accordingly, speech-to-speech translation researches [1]-[6] grew to a leading-edge technology enabling multilingual human-to-human and human-to- machine interaction. In previous work, two different architectures are adopted by many researchers for speech-to-speech translation research [4],[15] – a conventional sequential architecture and a fully integrated architecture. The sequential architecture is composed of a speech recognition system followed by a linguistic (or non-linguistic) text-to-text translation system and a text-to-speech system [1],[2],[5],[16-20]. The integrated architecture combines speech feature models and translation models in a manner similar to that used for speech recognition. This integration brings about efficient translation by searching for an optimal word sequence of target language through the integrated network such as finite-state transducer [4],[21] and the others [22],[23].

According to these two architectures, there are two system implementations – client-server-based systems and stand-alone handheld systems. Nevertheless, a critical shortcoming of client-server based speech translation systems is that they should be built on the server computer. In other words, it is not available anytime or anywhere. An obvious solution would be to build portable stand-alone speech-to-speech translation handheld devices. To mention just a few: Isotani *et al.* [7] used a Pocket PC PDA with 206 MHz StrongARM/64 MB RAM to build a speech-to-speech translation system for the use in various situations while traveling. *Waibel et al.* [8] adopted a Pocket PC PDA with 400 MHz StrongARM/64 MB RAM to construct a speech-to-speech translation system for medical interviews. Watanabe *et al.* [9] developed a mobile device running on 400 MHz Pentium II-class processor/192 MB RAM that helps speech-to-speech translation in various situations during their travel abroad. However, these works show that real-time speech-to-speech translation with a resource-limited device is still a problem.

Table 1 shows a comparison among related speech-to-speech translation systems, including JANUS [1], Verbmobil [2], EUTRANS [4] etc. Clearly, the client-server based architecture can work in real time, but is not portable; while the stand-alone system is portable, but lacks the real-time performance. Therefore, a VLSI solution is presented by realizing the entire

speech-to-speech translation algorithm within a single chip. This SOC chip only requires a few peripheral components for complete operation, and is characterized by small size, low cost, real-time operation, and high reliability. The construction of this chip is accomplished in two main phases: the software simulation phase and the SOC design phase.

In the software simulation phase, the simulation is based on the multiple-translation spotting (MTS) method, a kind of integration of speech analysis and language translation [23]. The proposed multiple-translation spotting approach is directly from speech to speech without language models like other automatic speech recognition (ASR) approaches. With identifying speech features, translation primarily stays in the speech modality and does not go through a textual modality. The proposed MTS method not only retrieves the optimal multiple-translation spotting template, but also extracts the appropriate target patterns. With the extracted patterns, the target speech can be generated by a concatenation-based waveform segment synthesis method.

In the SOC design phase, besides a cost efficient programmable core used for system control and non-computation-intensive tasks, three specific hardware cores were designed to perform cepstrum extraction, template retrieval, target pattern extraction. Moreover, the A/D converter (ADC) and D/A converter (DAC) are also designed.

The rest of this paper is organised as follows. Section 2 gives the software simulation of the proposed speech-to-speech translation system. Section 3 discusses the SOC architecture for the MTS-based speech-to-speech translation system. Finally, a short conclusion is provided in Section 4.

System	Vocabulary Size	Response time	Specification
JANUS [1]	3,000~5,000	≤ 2 times real-time	PC-based platform (server-client)
Verbmobil [2]	≥10,000	4 times real-time	PC-based platform (server-client)
EUTRANS [4]	1,701(English) 2,459(Italian)	≤ 3 times real-time	PC-based platform (server-client)
ATR-MATRIX [5]	13,000	0.1 sec for TDMT	PC-based platform (server-client)
Isotani *et al*. [7]	20,000(English) 50,000(Japan)	Slower than V_R5500 processor	Pocket PC PDA (206 MHz StrongARM/64 MB RAM)
Speechalator [8]	–	≥2~3 sec	Pocket PC PDA (400 MHz StrongARM/64 MB RAM)
Watanabe *et al.* [9]	10000 (English) 50000 (Japan)	≥2~3 sec	Mobile PC (400 MHz Pentium II-class processor/192 MB RAM)

Table 1. A comparison among related speech-to-speech translation systems

		Source-language (English)	Target-language (Chinese)
TS	SL Input		
	I want a double room.	I want a single room with shower for tomorrow.	我 明天 要 一 間 有 淋浴 設備 的 單人 房
		⟨**I, want, a, room**⟩	⟨**我,要,一,間,房**⟩
	Is there a single room for tomorrow?	Is there a double room for tonight?	請 問 你們 今晚 有 一 間 雙人 房 嗎
		⟨**Is there, a, room, for**⟩	⟨**有,一,間,房,嗎**⟩
MTS	SL Input		
	I want a double room.	I want a ⟨single, double⟩ room with shower for ⟨tomorrow, tonight⟩.	我 ⟨明天,今晚⟩ 要 一 間 有 淋浴 設備 的 ⟨單人,雙人⟩ 房
		⟨**I, want, a, double, room**⟩	⟨**我,要,一,間,雙人,房**⟩
	Is there a single room for tomorrow?	Is there a ⟨single, double⟩ room for ⟨tomorrow, tonight⟩?	請 問 你們 ⟨明天,今晚⟩ 有 一 間 ⟨單人,雙人⟩ 房 嗎
		⟨**Is there, a, single, room, for, tomorrow**⟩	⟨**明天,有,一,間,單人,房,嗎**⟩

Table 2. Examples for TS and MTS.[1]

2. The Proposed Speech-to-Speech Translation System

2.1 System Overview

Multiple-translation spotting (MTS) is proposed as an improvement on the traditional translation spotting (TS) method [11], [12] and uses multiple-translation spotting templates derived from multiple pairs of translations for spotting all hypothesised target-language patterns. Table 2 lists some examples of TS and MTS. The source-language input "*I want a double room*" can only provide five target-language patterns <我,要,一,間,房> in traditional TS method but the proposed MTS provides all target-language patterns <我,要,一,間,雙人,房>. All target-language patterns <明天,有,一,間,單人,房,嗎> are also extracted by MTS while inputting the sentence "*Is there a single room for tomorrow.*" For each multiple-translation spotting template, hypothesised target patterns are generated in the MTS process.

Figure 1 shows an example of the MTS process for the direction of English-to-Chinese while inputting a speech. The multiple-translation spotting template of this example possesses eleven English patterns in a feature template. Based on the one-stage algorithm [13], when a speaker inputs an English speech "*Is a single room still available for tonight*", the proposed system can obtain the identified results – "*is*", "*a*", "*single room*", "*still*", "*available*", "*for*", and "*tonight*." Following the identified results, the translations of "*is*↔嗎", "*a*↔一間," "*single room*↔單人房", and so on are extracted. A Mandarin Chinese speech pattern set that includes seven hypothesised speech patterns, "嗎", "一間", "單人房", "還", "有", and "今晚" is generated by the constructed waveform translations in the multiple-translation spotting template and represented as darker waveforms in Fig. 1.

For speech generation, after determining the optimal target speech sequence, these waveforms are rearranged with adequate overlapping portions to generate speech with the

[1] The gray tablets list the translation spotting results.

waveform similarity overlap and add (WSOLA) algorithm [24]. WSOLA introduces a tolerance on the desired time-warping function to ensure signal continuity at waveform segment joins. With a proper windows length and a timing tolerance, WSOLA usually produces high quality time-scaled speech [25],[26]. Therefore, the system can generate high phonetic/prosodic quality in the translated speech output. The advantages of this method are the small computational cost during the generation process and the high intelligibility of the generated speech. The following subsections further discuss the details of the kernel spotting algorithm within the speech translation.

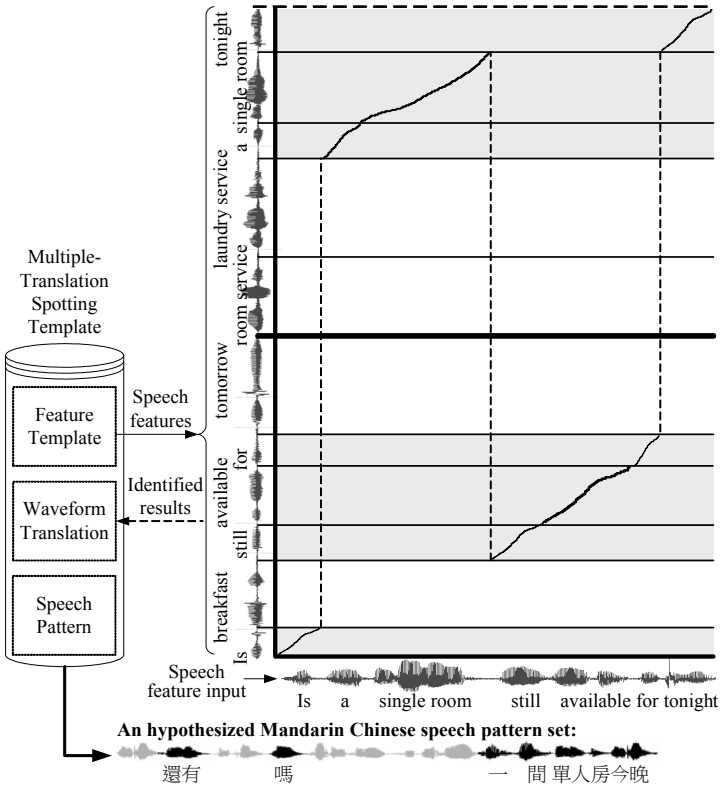


Fig. 1. An example of MTS process for the direction of English-to-Mandarin Chinese.

2.2 Multiple-Translation Spotting

To formulate the concept of MTS between input speech (X_1^L) and the *v*-th multiple-translation spotting template (r_v), we use the following notations: l represents the frame index within X_1^L, $1 \le l \le L$, j represents the translation pair $\langle s_j^v, t_j^v \rangle$ index within r_v, $1 \le j \le J$, and k represents the frame index within the j-th source pattern s_j^v and its mapped target pattern denoted by t_j^v, $1 \le k \le K_j$. Then for each input frame, the accumulated distortion $d_A(l,k,j)$ is defined by:

$$d_A(l,k,j) = d(l,k,j) + \min_{k-2\le m\le k}\left(d_A(l-1,m,j)\right), \tag{1}$$

for $2 \le k \le K_j$, $1 \le j \le J$, where $d(l,k,j)$ is the local distortion between the *l*-th frame of X_1^L and the *k*-th frame of the source pattern s_j^v. The recursion in (1) is carried out for the internal frames (i.e., $k \ge 2$) of each source pattern. At the pattern boundary, i.e., when $k = 1$, the recursion can be calculated as:

$$d_A(l,1,j) = d(l,1,j) + \min\left[\min_{1\le m\le J}\left(d_A(l-1,K_m,m)\right), d_A(l-1,1,j)\right]. \tag{2}$$

And the best path is determined by

$$d_G^v = \min_{1\le j\le J}\left[d_A\left(L,K_j,j\right)\right]. \tag{3}$$

After ranking all the templates, the hypothesized spotting template is decided from the Top *N* candidates with minimum distortion by

$$\hat{v} = \arg\max_{1\le v\le N} \sum_{j=1}^{J} \tau_j^v \tag{4}$$

According to the decided $\hat{v}$*-th* template from (4), the target patterns $\left\{t_j^{\hat{v}}\right\}_{j=1}^{J}$ can be obtained by $\left\{\tau_j^{\hat{v}}\right\}_{j=1}^{J}$. With the determining target speech patterns, these waveforms are rearranged with adequate overlapping portions to generate speech with the waveform similarity overlap and add (WSOLA) algorithm.

2.3 Software Simulation

The translation experiments were performed on both a PC-based platform and iPAQ PDAs. On the PC-based platform, the software simulations were done using Windows CE 3.0 on a Pentium® IV 1.8 GHz, 1 GB RAM, Windows® XP PC. On the COMPAQ iPAQ PDAs, the system was implemented on a 400 MHz Intel® XScale processor with 128 MB RAM. This work built a collection of English sentences and their Chinese translations that frequently appear in phrasebooks for foreign tourists. The phrasebooks are referred to [27] and [28]. Six sub-domains are used for collection including accommodation, restaurant, traffic, shopping, tourism, and asking for directions. Each sub-domain contains 174~251 translations.

Experiments were conducted between Mandarin Chinese and English. To evaluate the system performance, the 1,260 utterances of the training set used for constructing multiple-translation spotting templates were collected from one speaker (Sp1) and the 105 utterances of the test set were also collected from the same speaker (Sp1) for closed testing and two bilingual male speakers (Sp2 and Sp3) for open testing. First speaker's feature models (spotting templates) were used to proceed tests on the remaining two speakers. All speeches had an 8 kHz sampling rate with a precision of 16 bits on both the PCs and the PDAs. The speech feature analysis was performed using 10th-order linear prediction cepstral coefficients (LPCCs) using 32 ms frames with 8 ms overlap. The total memory requirement of the whole system at the startup was about 22.9 MB. The required work memory is about 1 MB, depending on the length of a speech input.

While hesitations or insertions are occurred, the proposed MTS approach can spot critical patterns as keyword spotting or partial matching and a target output can be properly generated by the spotted patterns with the decided spotting template. However, the proposed approach would be sensitive to disturbances including speaking rate, noise, speaker properties, and so on. For the effect of speaker properties for the proposed system,

For retrieving Top 5 templates, Table 3 shows that the spotting accuracy of Sp2 and Sp3 drops by 10 to 15 percent. A given spotting template is called a *match* when it obtains the same intention of the input speech. In addition, from the research presented in [31], the speaking rate had a significant effect on recognition accuracy and further adaptation methods of duration models for spotting templates are needed. Based on the proposed approach, when template or vocabulary size increases, the increasing spotting templates will lead to more speech feature vectors and hence more similarities will occur in speech spotting measurement, thus causing false spotting results and lowering spotting accuracy. By collecting more speech databases, the system can apply speaker-dependent or speaker-independent HMM to MTS for more robust speech translation. Speech translation performance also is degraded by noise. Related works to minimize the effects of the noise on the system performance are presented in [29],[30] and would be applied to the proposed system in the future.

To judge the generated translations from the matched templates, a subjective sentence error rate (SSER) in [3] is used to evaluate and classify the target generation results into three categories by three bilingual evaluators. Referring to the SSER evaluation method, *good* (*G*), *understandable* (*U*), *bad* (*B*) levels of translation quality are scaled to 1.0, 0.5, and 0.0, respectively. The understandable translation rate is calculated by

$$\text{rate} = \frac{1.0 \times (\text{no. of } G \text{ level}) + 0.5 \times (\text{no. of } U \text{ level})}{\text{no. of tests}} \times 100\% \tag{5}$$

The results revealed that our proposed approach roughly achieves 89% and 92% understandable translation rate from (5) for the Mandarin Chinese-to-English and the English-to-Mandarin Chinese translations, respectively.

			Template Size (spotting templates of Sp1)									
		Top5	360	460	560	660	760	860	960	1060	1160	1260
All	Sp1	E2C	0.96	0.93	0.93	0.9	0.86	0.83	0.83	0.8	0.76	0.73
		C2E	0.93	0.9	0.86	0.83	0.8	0.8	0.76	0.76	0.73	0.7
	Sp2	E2C	0.83	0.8	0.76	0.73	0.73	0.7	0.66	0.66	0.66	0.63
		C2E	0.8	0.76	0.73	0.73	0.7	0.66	0.63	0.63	0.63	0.6
	Sp3	E2C	0.76	0.76	0.73	0.7	0.7	0.66	0.66	0.63	0.6	0.6
		C2E	0.73	0.73	0.7	0.66	0.66	0.63	0.6	0.6	0.6	0.6

Table 3. Average spotting accuracy in multiple speaker testing.

3. SOC Design for the MTS-based Speech-to-Speech Translation System

The VLSI architecture for the speech-to-speech translation SOC was developed using the programmable application-specific technique. Besides a cost efficient programmable core [14], which is used for system control and other non computation-intensive tasks, three specific hardware cores are also designed for feature extraction, template retrieval, pattern extraction. Moreover, the A/D converter and D/A converter are also designed.

Figure 2 shows the overall block diagram of the VLSI architecture for the speech-to-speech translation SOC. This architecture mainly consists of a cepstrum extraction core, a template retrieval core, a pattern extraction core, a programmable core, and ADC/DAC. In the

following paragraph, we will discuss the three specific processing hardware cores and the ADC/DAC for our speech-to-speech translation system.

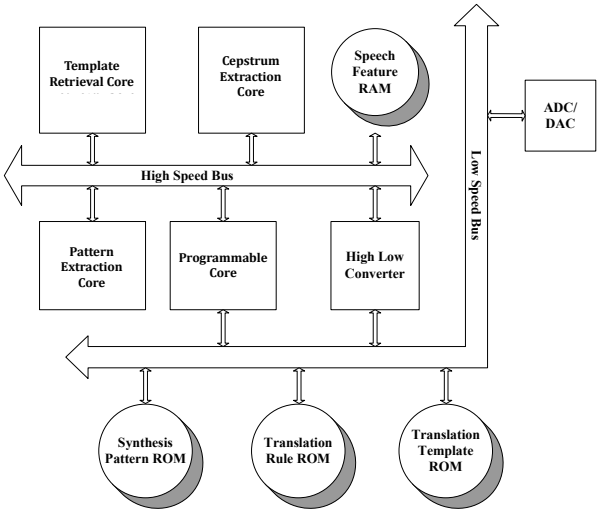


Fig. 2. Block diagram of the SOC architecture for the proposed speech-to-speech translation system.

3.1 The Design of the Cepstrum Extraction Core

Feature extraction is one of the most important issues in the field of speech recognition. In many speech recognition systems, the cepstral coefficients are treated as the feature parameters, derived by the Fourier Transform or from the LPC coefficients. We selecte the latter way as it yields high recognition rate and lower computational load.

3.1.1 Autocorrelation Analysis

The sampling rate of the input speech for the cepstrum extraction core is 8 KHz. The pre-emphasis filter for the digitized speech is defined by

$$H(z) = 1 - hz^{-1}, \quad 0.9 \le h \le 1.0 \cdot \tag{6}$$

Because we adopt fixed-point implementation, h is chosen as 15/16. The pre-emphasized speech is then blocked into N samples, with adjacent frames being separated by M samples. In other words, the adjacent frame begins M samples later than the previous frame and overlaps with it by N-M samples. In our design, M=192, the frame size N=256, and therefore the frame rate equals 31.25 frames per second. The next step is to window each frame so that the signal discontinuities at the beginning and the end of each frame are minimized. The Hamming window is used and has the form of

$$w(i) = \begin{cases} 0.54 - 0.46\cos(\frac{2i\pi}{N-1}), & 0 \le i \le N-1 \\ 0 \quad , & otherwise \end{cases} \tag{7}$$

For each frame of the windowed speech signal, autocorrelation analysis is performed by the following formula:

$$R(k) = \sum_{i=k}^{N-1} x(i-k)x(i), 0 \le k \le P' \tag{8}$$

where P is the order of the LPC analysis.

The detailed architecture for the autocorrelation analysis of overlapping frames from the digitized input speech can be seen in [32]. The architecture is based on a calculation procedure which is depicted in Fig. 3.

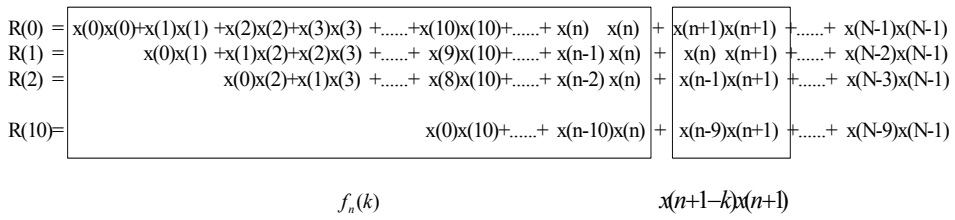


Fig. 3. An example illustrating the autocorrelation calculation procedure.

3.1.2 Linear Predictive Analysis

Of the various linear predictive analysis algorithms, we adopt the autocorrelation method because it leads to more stable results than the covariance method and has lower computational load than the lattice method [33]. For the autocorrelation method, the matrix equation for solving the LPC coefficients is in the form of

$$R(i) = \sum_{k=1}^{P} a_k R(|i-k|), \qquad 1 \le i \le P' \tag{9}$$

where $R(i)$ is the autocorrelation coefficient, P is the order of the LPC analysis, chosen as 10 in this paper.

The most efficient method known for solving this particular system of equations is the Levinson-Durbin recursion, which can be formulated as follows [33]:

$$E^{(0)} = R(0) \tag{10}$$

for $1 \le i \le P$

$$temp = R(i) - \sum_{j=1}^{i-1} a_j^{(i-1)} R(i-j) \tag{11}$$

$$K_i = temp / E^{(i-1)} \tag{12}$$

end for

$$a_i^{(i)} = K_i \tag{13}$$

$$a_j^{(i)} = a_j^{(i-1)} - K_i a_{i-j}^{(i-1)}, \qquad 1 \le j \le i-1 \cdot \tag{14}$$

$$E^{(i)} = (1 - K_i^2) E^{(i-1)}, \tag{15}$$

The above equations are solved recursively for $i = 1, 2, \cdots, P$ and the final solution is given as

$$a_j = a_j^{(P)}, \qquad 1 \le j \le P \cdot \tag{16}$$

Substituting (12) into (15), we obtain that

$$E^{(i)} = E^{(i-1)} - temp \cdot K^{(i)} \cdot \tag{17}$$

The architecture based on pipeline fashion for the Levinson-Durbin recursion is described in

Fig. 4. There are a small number of divisions needed for the LPC computation. However, it is not economical to prepare extra hardware for them. Instead, the division operations can be performed by prune-and-search [34] on the hardware without the use of an individual divider.

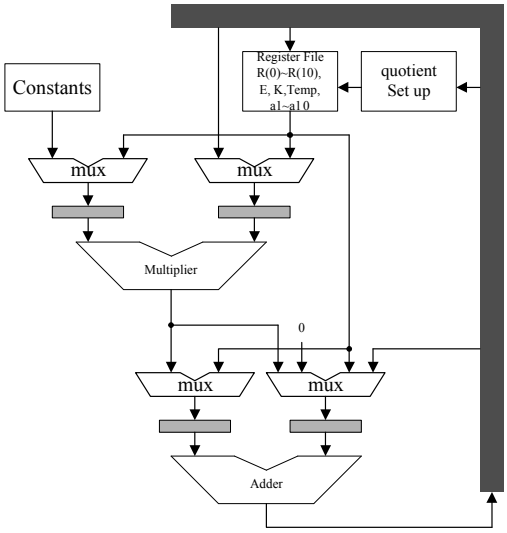


Fig. 4. The architecture for the LPC analysis.

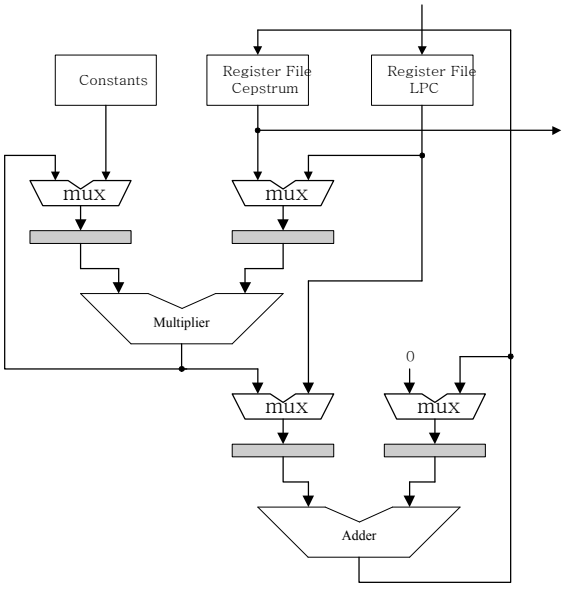


Fig. 5. The architecture for the conversion from LPC to cepstrum.

3.1.3 Conversion from LPC to Cepstrum

The cepstrum is computed from the linear prediction model. The recursive equations of the LPC based cepstrum are as follows [35]:

$$C_1 = -a_1, \tag{18a}$$

$$C_n = -a_n - \sum_{m=1}^{n-1}(1-m/n)a_m C_{n-m}, \qquad 1 \le n \le P, \tag{18b}$$

where C_n is the nth cepstral coefficient, a_n is the nth LPC coefficient, and P is the cepstrum order, which is set equal to the LPC order in this paper.

The memory requirement for the constant $(1-m/n)$ can be reduced as follows. There are n-1 constants for any n, so the total number of the various constants is 1+2+3+…+P-1=P(P-1)/2. By taking into consideration the pattern of the constant array, let $k_{m,n}$=(1-m/n), then we can show that

$k_{n\text{-}1\text{-}i,n}$=1- $k_{i,n}$, (19)

for i=1, 2 ,…,$\lceil n-1/2 \rceil$, and if n is even then $k_{n/2,n}$ =1/2.

Equation (19) implies that $k_{i,n}$ is equal to the complement of $k_{n-1-i,n}$, therefore, the size of the constant array can be reduced to half. This algorithm is implemented by a store-and-accumulate technique and the architecture is described in Fig. 5. We need two storage elements to hold the LPC coefficients and the previous order cepstral coefficients. In addition, an accumulator is necessary for computing and totaling the main expression, (1-m/n)$a_n C_{n-m}$ to yield the cepstral coefficients defined in (18b).

The dedicated architecture for the autocorrelation analysis, the LPC analysis, and the conversion from LPC to LSP are individually designed. However, since the three procedures are performed sequentially, a resource-sharing technique,is performed so that the cepstrum extraction core will need only one multiplier and one adder.

3.2 The Design of the Template Retrieval Core

To exemplify our design concept, Figs. 6(a) and 6(b) display two dependence graphs (DGs) of the template retrieval. These dependence graphs are constructed according to (1) and (2), and can be regarded as the template retrieval space. The DG shown in Fig. 6(a) describes the case with 12 frames in the input speech (X_1^{12}). There are three source patterns ($s_1^{v1}, s_2^{v1}, s_3^{v1}$) in this template ($r_{v1}$), and the frame number of them is 7, 8, and 6, respectively. The DG shown in Fig. 6(b) has 8 frames in the input speech (X_1^8) and 2 patterns in the template (r_{v2}). The frame number of the two patterns (s_1^{v2}, s_2^{v2}) in this template is 6 and 7, respectively.

The different frame numbers within different input speeches cause the variation along the horizontal axis in the template retrieval space. The different pattern numbers within different templates and the different frame numbers within different patterns are responsible for the variation along the vertical axis in the template retrieval space. The architecture design for coping with the variation of the structure of the template retrieval space is described as follows. First, horizontal projection is performed in the original DGs (Fig. 6(a)) and the result is shown in. Fig. 6(c). This projection reduces the two-dimensional (2-D) template retrieval space into a one-dimensional (1-D) one and eliminates the variation resulting from different frame numbers within different input speeches. Each node in Fig. 6(c) requires a register to store the computational results for the distortion accumulation of the next frame. The nodes within one

column are divided into different blocks according to the patterns they belong to. Thus the DG in Fig. 6(c) consists of three blocks corresponding to three different patterns.

To further reduce the size of the DG, a vertical projection is performed in Fig. 6(c). This projection transforms the three-block DG into a one-block DG (see Fig. 6(e)). Because the frame numbers for the blocks are different, the largest frame number is taken as the frame number for the new one-block DG. To deal with the discrepancy of having different frame numbers within different patterns, we added some multiplexers. In addition, two extra registers are added in front of each node to replace the two eliminated blocks. Figure 6(e) is then modified as Fig. 6(g) to have a regular wire connection.

Figures 6(b), 6(d), 6(f) and 6(h) provide another example of the use of horizontal and vertical projections. One can find that the frame number and the register number for a certain node between the DGs in Figs. 6(g) and 6(h) are different. To combine the two DGs into a single one, the largest frame number and the largest register number between them are chosen, see Fig. 6(i). The added multiplexers in front of each node provide the selectivity among the one-block DGs.

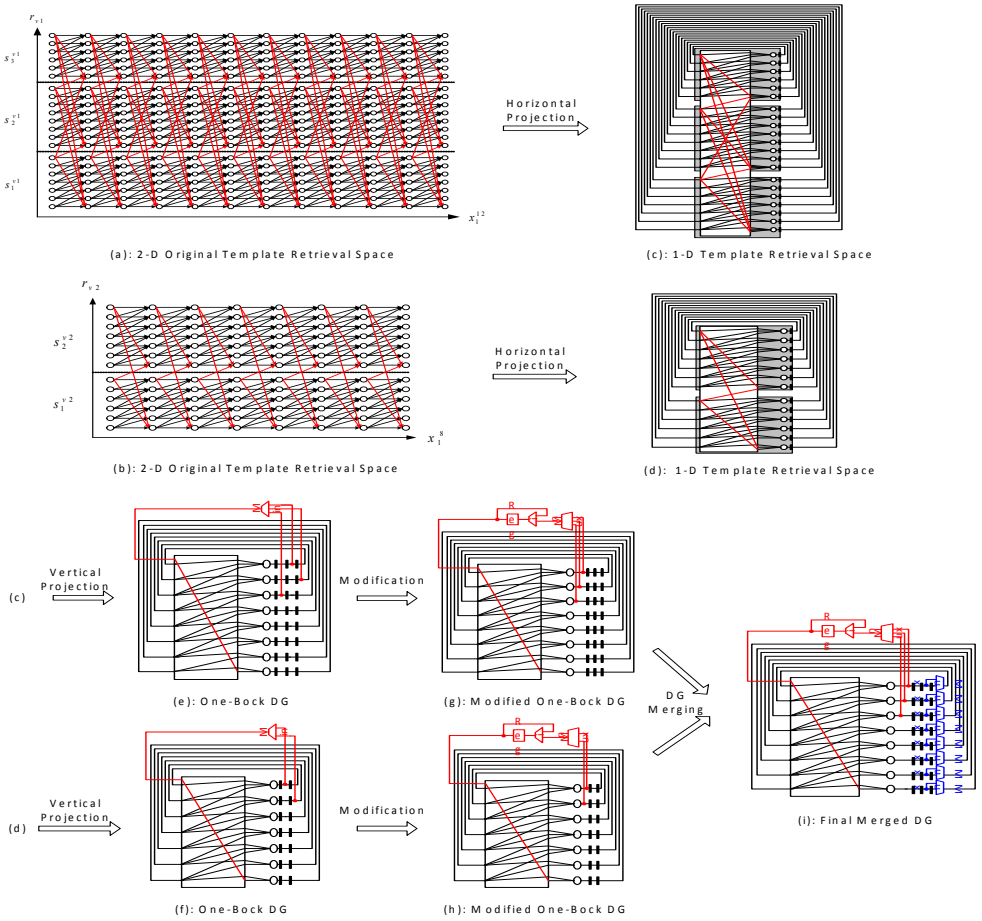


Fig. 6. Architecture inference of the template retrieval based on DG.

3.2.1 The Distortion Unit

Each node in the template retrieval DG associates with a local distortion. The local distortion between a frame of the template and a frame of the input speech is defined by

$$\text{Local distortion} = \sum_{i=0}^{P} |R_i - U_i|' \tag{20}$$

where R_i denotes the *i*-th cepstral coefficient in the *R*-th frame of the template, U_i represents the *i*-th cepstral coefficient in the *U*-th frame of the input speech, and *P* is the cepstrum order. The cepstral coefficients of the input speech and the templates are stored in RAM0 and ROM1, respectively. The distortion unit accesses the cepstral coefficients from the two memories, and then accumulates the distortion in register *R*.

3.2.2 The Processing Element

Each node in the DGs shown in Fig. 6(i) is implemented as a processing element. The architecture design of the PE is displayed in Fig. 7. After receiving all the accumulated distortions from the registers, the PE selects the minimum accumulated distortion, and then adds it to the local distortion $d(l,k,j)$. The result is the new accumulated distortion $d_A(l,k,j)$ associated with the current node. In addition to the accumulation process, the PE also generates the decision information that indicates the source node in a path transition. The use of the decision information is discussed together with the pattern extraction core later.

Based on the data path of the template retrieval core, a corresponding template retrieval controller is also developed. In addition to producing control signals, this controller also functions as a memory address generator for accessing the template speech features. The template retrieval core not only accumulates the minimum local distortion but also writes the decision information into the memory. The decision information for a node indicates its source node in a path transition. The best path can be obtained from the decision information, tracing it backward for extracting the hypothesised speech patterns. The design of the pattern extraction core is given in the next subsection.

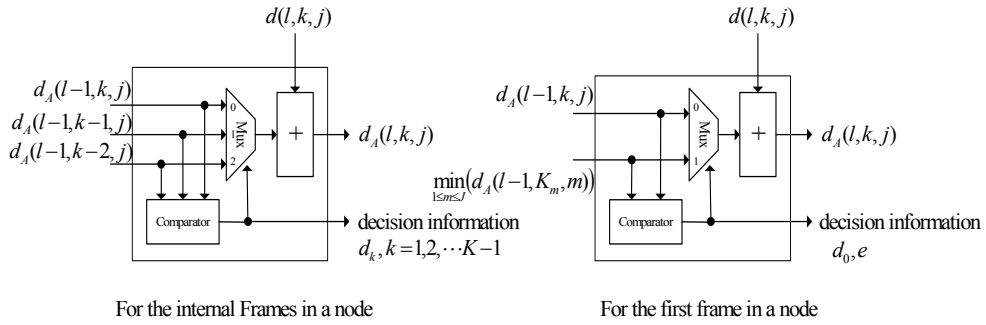


Fig. 7. Architecture of the processing element.

3.3 The Design of the Pattern Extraction Core

To extract the best pattern sequence, this work presents a block-node pattern extraction method. Figure 8 exemplifies the relationship between nodes and blocks. The block boundaries are the same as the pattern boundaries. This template retrieval space contains 24

blocks, numbered from 0 to 23. Blocks in the same row belong to the same pattern, and have the same number of nodes (frames). For example, blocks 2, 5, 8, …, 23 belong to the third pattern within this template. The pattern extraction method used in this work can be regarded as a two-level address decoding process. At the first level, the blocks are the addressing units. Each block consists of frame nodes, which become addressing units at the second level. If b denotes the block index and k denotes the local node index in a block, each node can also be referred to by the pair (b, k). For example, *node* 64 can be referred to as *node* (21, 1).

The decision word is generated for each block, and is denoted by

$$D \equiv \{d_0, d_1, d_2, ..., d_{K-1}, e\}, \tag{21}$$

where K is the largest number among all the block node ones, e is the pattern decision information from the first node of a block, and d_k, $0 \le k \le K-1$, is the decision information from the k-th node of a block. We use e to indicate the source pattern for an external (between-patterns) transition, and d_k to indicate the source node for an internal (within-patterns) transition. Assume that K_j denotes the number of nodes in the j-th pattern.

Pattern extraction starts from the end node, and recursively update the local node index and the block index to construct a best path in the template retrieval space. The block index b is updated by

$$b = \begin{cases} b - J, & \text{for internal } transition, \\ b - J + new\_e - old\_e, & \text{for external } transition, \end{cases} \tag{22}$$

where J is the number of patterns in this template, *new_e* is the current value of the decision information e, and *old_e* is the previous value of the decision information e.

As for the local node index k, it is updated by

$$k = \begin{cases} k, & \text{for internal transition with } d_k = 0, \\ k-1, & \text{for internal transition with } d_k = 1, \\ k-2, & \text{for internal transition with } d_k = 2, \\ K_e - 1, & \text{for external transition.} \end{cases} \tag{23}$$

Let us denote the best path end in the block by *end_block*, and its pattern by *end_pattern*. The flow chart of the algorithm is depicted in Fig. 9.

The block-node pattern extraction method is implemented as a finite state machine (FSM). Because template retrieval and pattern extraction are required for all templates, this design is based on a two-memory scheme. While the decision word is being written into RAM1, the back tracing is being performed in RAM2; and vice versa. This reduces the overall computation time of template retrieval and pattern extraction for all templates.

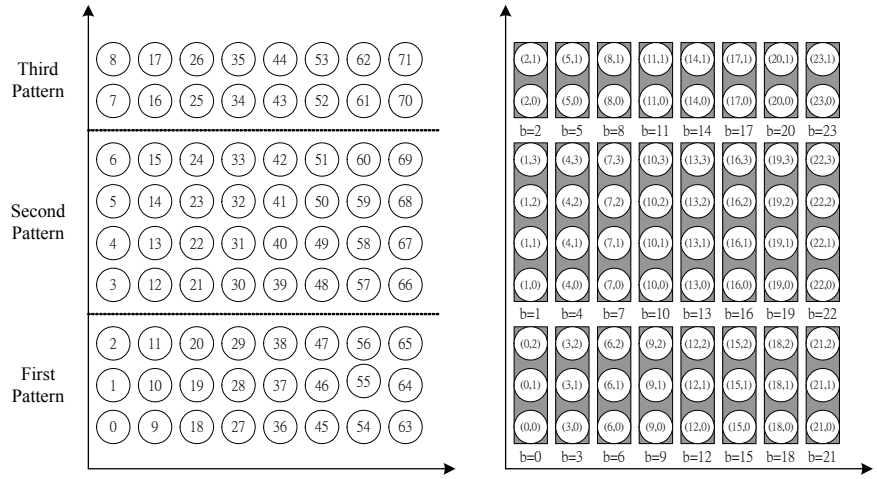


Fig. 8. Example illustrating the block-node addressing method.

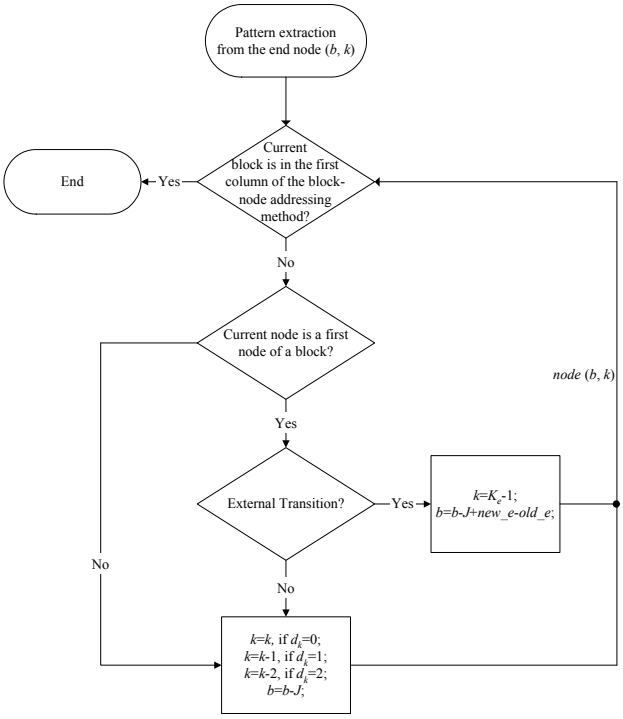


Fig. 9. Flow chart of the block-node pattern extraction.

3.4 The Design of the ADC/DAC

The last specific hardware core is ADC/DAC. In this part, we mainly present an efficient architecture to improve the conventional fully differential successive approximation ADC. The proposed architecture includes two optimal design solutions. First, in order to avoid the mismatch between positive and negative reference voltages, a single reference voltage (SRV) method that removes the negative reference voltage is developed by the distinct switched capacitor controlling algorithm. Figure 10 shows the implementation of the SRV algorithm in circuit. Second, a small area scheme for successive approximation register (SAR) is introduced. Comparing to the traditional successive approximation ADC, our proposed architecture is contributive to improve the area saving in SAR and the accuracy of ADC.

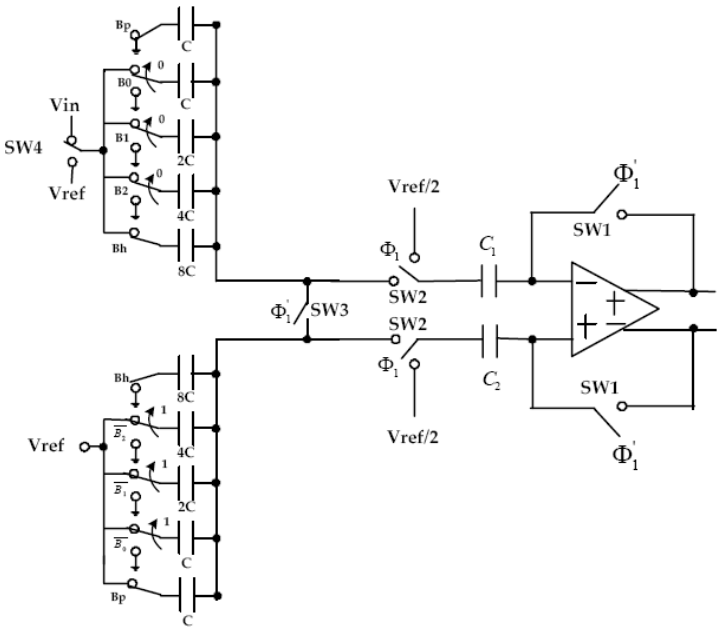


Fig. 10. The architecture of fully differential successive approximation ADC using single reference voltage.

To reduce the area for successive approximation register, this work presents a simplified non-redundant successive approximation register (SSAR). The basic architecture of the SSAR is a multiple input n-bit shift register, shown in Fig. 11. This multiple input register consists of a general D flip-flop and a multiplexer with two inputs. The function of SSAR is shown in Fig. 11. Suppose the "1" is the input token in Fig. 12. Whenever the SSAR is triggered, each register of SSAR must go through three modes: (1) when the token has not passed yet, the values of the registers are "0"; (1) when the token is staying at a certain register, its register value is changed to "1", and receives the result of comparator; (3) when the token has passed through, the value determined by the result of comparator is held until the whole conversion is done. The function can be implemented as below.

Therefore, when the value of k-th register of the SSAR is still "0", the Q_{k+1} selected by the multiplexer is connected to D_k. If the k-th register receives the token, the Q_k is changed to "1",

and the CMP would be selected. The result of comparator Z_k would be held in the k-th register until the whole conversion is done. The system can be implemented by the blocked CLK_k. The combination of all above implementations makes it possible to improve the accuracy and save the chip area.

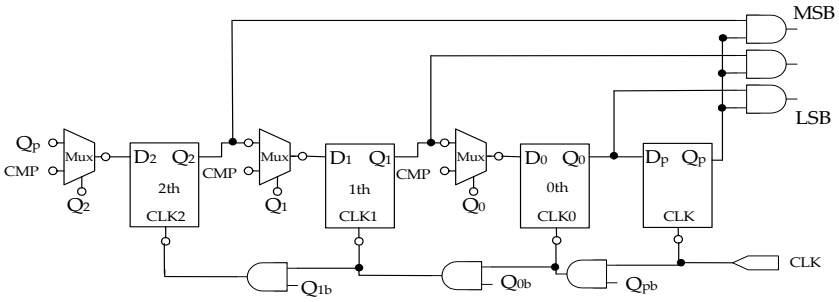


Fig. 11. The architecture of the simplified non-redundant SAR.

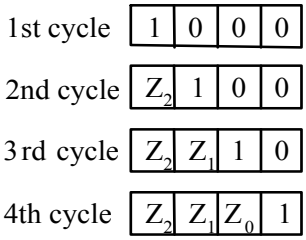


Fig. 12. The function of simplified non-redundant SAR.

4. Summary

Speech-to-speech machine translation is a prospective application of speech and language technology. This work presents an MTS based speech-to-speech translation system between Mandarin Chinese and English. The proposed MTS approach achieves about a 90% understandable translation rate on average. For a portable and real-time speech-to-speech translation system, this work also proposes the SOC realization. The architecture design is based on the semi-ASIC technique, which incorporates a cost efficient programmable core along with specific hardware accelerators: an LPC extraction core, a template retrieval core, and a pattern extraction core. Besides, the ADC/DAC are also included. A SRV-based fully differential successive approximation ADC with reduced SSAR is designed. These hardware cores construct a complete speech-to-speech translation SOC. The proposed SOC chip is the first one dedicated for speech-to-speech translation.

5. References

[1]A. Lavie *et al*., "JANUS III: speech-to-speech translation in multiple languages," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, Apr. 1997, pp. 99–102.

[2]W. Wahlster, *Verbmobil: Foundations of Speech-to-Speech Translation*. Berlin Heidelberg, New York: Springer-Verlag, 2000.

[3]H. Ney, S. Nießen, F. J. Och, H. Sawaf, C. Tillmann, and S. Vogel, "Algorithms for statistical translation of spoken language," *IEEE Trans. Speech and Audio Processing*, vol. 8, pp. 24-36, Jan. 2000.

[4]F. Casacuberta *et al*., "Speech-to-speech translation based on finite-state transducers," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, May 2001, pp. 613–616.

[5]F. Sugaya, T. Takezawa, A. Yokoo, and S. Yamamoto, "End-to-end evaluation in ATR-MATRIX: speech translation system between English and Japanese," in *Proc. 6th Eur. Conf. Speech Communication and Technology*, Sep. 1999, pp. 2431–2434.

[6]P. C. Ching and H. H. Chi, "ISIS: a trilingual conversational system with learning capabilities and combined interaction and delegation dialogs," in *Proc. National Conf. Man-Machine Speech Communications*, Nov. 2001, pp. 119–124.

[7]R. Isotani, K. Yamabana, S. Ando, K. Hanazawa, S. Ishikawa, T. Emori, H. Hattori, A. Okumura, and T. Watanabe, "An automatic speech translation system on PDAs for travel conversation," in *Proc. IEEE Int. Conf. Multimodal Interfaces*, Oct. 2002, pp. 211–216.

[8]A. Waibel, A. Badran, A. W. Black, R. Frederking, D. Gates, A. Lavie, L. Levin, K. Lenzo, L. M. Tomokiyo, J. Reichert, T. Schultz, D. Wallace, M. Woszczyna, and J. Zhang, "Speechalator: two-way speech-to-speech translation on a consumer PDA", in *Proc. European Conf. Speech Communication and Technology*, Sep. 2003, pp. 369–372.

[9]T. Watanabe, A. Okumura, S. Sakai, K. Yamabana, S. Doi, and K. Hanazawa, "An automatic interpretation system for travel conversation," in *Proc. Int. Conf. Spoken Language Processing*, Sep. 2000, pp. IV-444–IV-447.

[10]J. F. Wang, B. Z. Houg, and S. C. Lin, "A study for Chinese text to Taiwanese speech system," in *Proc. Int. Conf. Research on Computational Linguistics*, Aug. 1999, pp. 37–53.

[11]M. Simard, "Translation spotting for translation memories," in *Proc. HLT-NAACL Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, May 2003, pp. 65–72.

[12]J. Véronis and P. Langlais, "Evaluation of parallel text alignment systems – the ARCADE project," in *Parallel Text Processing*, Dordrecht: Kluwer Academic, 2000, pp. 369–388.

[13]L. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., 1993.

[14]J. F. Wang, A. N. Suen, and C. K. Chieh, "A programmable application specific architecture for real-time speech recognition," in *Proc. of VLSI Design/CAD Symposium*, Aug. 1995, pp. 261–264.

[15]Y. Zhang, "Survey of current speech translation research," presented at Multilingual Speech-to-Speech Translation Seminar, Carnegie Mellon University, Pittsburgh, PA, 2003.

[16]Y. S. Lee and S. Roukos, "IBM Spoken Language Translation System Evaluation," in *Proc. INTERSPEECH2004 Workshop on Spoken Language Translation: Evaluation Campaign on Spoken Language Translation*, Oct. 2004, pp.39–46.

[17]L. Gu and Y. Q. Gao, "On Feature Selection in Maximum Entropy Approach to Statistical Concept-based Speech-to-Speech Translation," in *Proc. INTERSPEECH2004 Workshop on Spoken Language Translation: Evaluation Campaign on Spoken Language Translation*, Oct. 2004, pp.115–121.

[18]S. Nakamura, K. Markov, T. Jitsuhiro, J. S. Zhang, H. Yamamoto and G. Kikui, "Multi-Lingual Speech Recognition System for Speech-To-Speech Translation," in *Proc. INTERSPEECH2004 Workshop on Spoken Language Translation: Evaluation Campaign on Spoken Language Translation*, Oct. 2004, pp.146–154.

[19]K. Matsui, Y. Wakita, T. Konuma, K. Mizutani, M. Endo, and M. Murata, "An experimental multilingual speech translation system," in *Proc. ICMI-PUI*, 2001, pp. 1–4.

[20]S. Rossato, H. Blanchon, and L. Besacier, "Speech-to-speech translation system evaluation: Results for French for the Nespole! project first showcase," in *Proc. ICSLP*, 2002, pp. 1905–1908.

[21]F. Casacuberta, E. Vidal, and J. M. Vilar, "Architectures for speech-to-speech translation using finite-state models," in *Proc. ACL Workshop on Speech-to-Speech Translation: Algorithms and Systems*, 2002, pp. 39–44.

[22]H. Ney, "Speech translation: Coupling of recognition and translation," in *Proc. ICASSP*, 1999, pp. 517–520.

[23]J. F. Wang and S. C. Lin, and H.W. Yang, "Multiple-Translation Spotting for Mandarin-Taiwanese Speech-to-Speech Translation," *Int. Journal of Computational Linguistics and Chinese Language Processing*, vol.9, no.2, 2004, pp. 13-28.

[24]W. Verhelst and M. Roelands, "An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech," in *Proc. ICASSP*, 1993, pp. 554–557.

[25]M. Demol, K. Struyve, W. Verhelst, H. Paulussen, P. Desmet, and P. Verhoeve, "Efficient non-uniform time-scaling of speech with WSOLA for CALL applications," presented at InSTIL/ICALL Symp. Computer Assisted Learning, Venice, Italy, 2004.

[26]H. G. Ilk and S. Tugac, "Channel and source considerations of a bit-rate reduction technique for a possible wireless communications system's performance enhancement," *IEEE Trans. Wireless Communications*, vol. 4, no. 1, pp. 93–99, Jan. 2005.

[27]E. T. Cornelius, *English 900*. Pace Group International Inc., 1999.

[28]B. E. Bagnell and M. Lee, *New Globe English Course on Travel*. New Globe Publishing Co., LTD, 1990.

[29]J. F. Wang and S. H. Chen, "Speech Enhancement Using Perception Wavelet Packet Decomposition and Teager Energy Operator", Journal of VLSI Signal Processing, Vol. 36 I: 2-3, pp. 125–139, Feb. 2004..

[30]J. F. Wang, C. H. Yang, and K. H. Chang, "Design of a Subspace Tracking Based Speech Enhancement System", in *Proc. IEEE TENCON 2004*, vol. 1, pp. 147–150..

[31]J. T. Chien and C. H. Huang, "Bayesian Learning of Speech Duration Models," *IEEE Trans. Speech and Audio Processing*, vol. 11 I:6, pp. 558–567, Nov. 2003.

[32]Jhing-Fa Wang, Jia-Ching Wang, Han-Chiang Chen, Tai-Lung Chen, Chin-Chan Chang, and Ming-Chi Shih, "Chip Design of Portable Speech Memopad Suitable for

Persons with Visual Disabilities," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 8, pp. 644-658, November 2002.

[33]J. Makhoul, "Linear prediction: a tutorial review," *Speech Analysis*, IEEE Press, New York, 1979.

[34]L. Y. Liu, J. F. Wang, J. Y. Lee, M. H. Sheu, and Y. L. Jeang, "An ASIC design for linear predictive coding of speech signals," in *Proc. Euro ASIC' 92*, 1992, pp.288-291.

[35]S. Furui, "Cepstral analysis technique for automatic speaker verification," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-29, no. 2, pp. 254-272, 1981.

16

A Novel De Bruijn Based Mesh Topology for Networks-on-Chip

Reza Sabbaghi-Nadooshan[1], Mehdi Modarressi[2,3] and Hamid Sarbazi-Azad[2,3]
[1]*Islamic Azad University Central Tehran Branch, Tehran, Iran*
[2]*Sharif University of Technology, Tehran, Iran*
[3]*IPM School of computer science, Tehran, Iran*

1. Introduction

The mesh topology is the most dominant topology for today's regular tile-based NoCs. It is well known that mesh topology is very simple. It has low cost and consumes low power. During the past few years, much effort has been made toward understanding the relationship between power consumption and performance for mesh based topologies (Srivasan et al., 2004). Despite the advantages of meshes for on-chip communication, some packets may suffer from long latencies due to lack of short paths between remotely located nodes. A number of previous works try to tackle this shortcoming by adding some application-specific links between distant nodes in the mesh (Ogras & Marculescu, 2005) and bypassing some intermediate nodes by inserting express channels (Dally, 1991), or using some other topologies with lower diameter (sabbaghi et al., 2008).

The fact that the de Bruijn network has a logarithmic diameter and a cost equal to the linear array topology motivated us to evaluate it as an underlying topology for on-chip networks. De Bruijn topology is a well-known network structure which was initially proposed by de Bruijn (de Bruijn, 1946) as an efficient topology for parallel processing. Samathan (Samathan & Pradhan, 1989) showed that de Bruijn networks are suitable for VLSI implementation, and several other researchers have studied topological properties, routing algorithms, efficient VLSI layout and other important aspects of the de Bruijn networks (Park & Agrawal, 1995; Ganesan & Pradhan, 2003).

In this chapter, we propose a two-dimensional de Bruijn based mesh topology (2D DBM for short) for NoCs. We will compare equivalent mesh and 2D DBM architectures using the two most important factors, network latency and power consumption. A routing scheme for the 2D DBM network has been developed and the performance and power consumption of the two networks under similar working conditions have been evaluated using simulation experiments. Simulation results show that the proposed network can outperform its equivalent popular mesh topology in terms of network performance and energy dissipation.

2. The 2D DBM Topology

2.1 The Structure

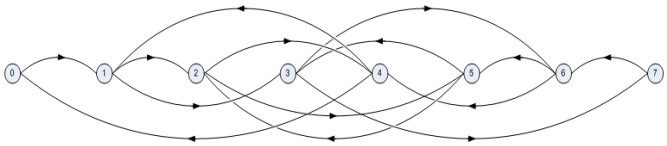
The basic idea about our work is based on digraphs, and in this section, we present the information compiled from studies conducted on the de Bruijn digraph (Liu & Lee, 1993 ; Mao & Yang , 2000).

The de Bruijn topology has many applications in communication networks and parallel processing (Samanathan & Pradhan, 1989). A de Bruijn graph has k^n nodes. Each node $u=(u_{n-1},...,u_0)$ has an edge to node $v=(v_{n-1},...,v_0)$ if and only if $v_i = u_{i-1}$ $1 \le i \le n-1$. Node v has a unidirectional direct link to node u if and only if

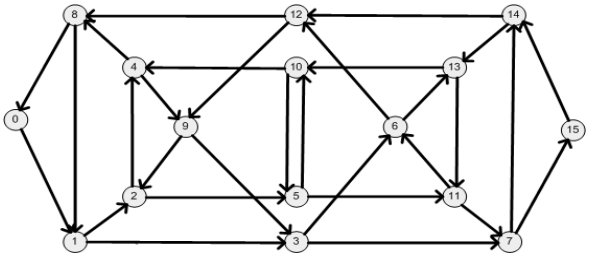
$$u = v \times k + r (\text{mod}\, k^n), \quad 0 \le r \le k-1 \tag{1}$$

The in-degree and out-degree of a node is equal to k. Therefore, the degree of each node is equal to $2k$. The diameter of a de Bruijin graph is equal to n which is optimal. The de Bruijn also has a simple routing algorithm. Case k=2 is the most popular de Brujin network which is also used in this study. Due to the logarithmic (optimal) diameter and a simple routing algorithm, it can be expected that the traffic on channels in the network will be less than other networks, resulting in a better performance (Ganesan & Pradhan, 2003).

Examples of de Bruijn networks are illustrated in Fig. 1. Several researchers have studied the topological properties (Liu & Lee, 1993 ; Mao & Yang , 2000) and efficient VLSI layout (Samanathan & Pradhan, 1989; Chen et al., 1993) of the de Bruijn networks. Moreover, the scalability problem of de Bruijn networks is addressed in (Liu & Lee, 1993). In de Brujin network data is circulated from node to node until it reaches its destination. Each node has two outgoing (incoming) connections to (from) other nodes via shuffle (rotate left by one bit) and shuffle-exchange (rotate left by one bit and complement LSB) operations to neighboring nodes (Louri & Sung, 1995). Owing to the fact that these connections are unidirectional, the degree of the network is the same as the one-dimensional mesh networks (or linear array network). The diameter of a de Bruijn network with size N, that is, the distance between nodes 0 and N-1, is equal to log (N).



(a)



(b)

Fig. 1. The de Bruijn network with (a) 8 nodes and (b) 16 nodes

2.2 The 2D DBM Topology

In a 2D DBM network, the nodes in each row and each column form a de Bruijn network. Each node has two outgoing edges along which data packets can be sent to other nodes, and two incoming links receiving data packets from other nodes in each dimension. Thus, node (u, v), $u = (u_{n-1},...,u_0)$, $v = (v_{n-1},...,v_0)$, has edges to node (u', v) if and only if $u'_i = u_{i-1}$ for $1 \le i \le n-1$, and node (u, v'), if and only if $v'_i = v_{i-1}$ for $1 \le i \le n-1$. Node (u', v') has a direct link to node (u, v') if and only if

$$u = 2u' + r_{\bmod 2^n}, \quad r = 0,1 \tag{2}$$

and to node (u', v) if and only if

$$v = 2v' + r_{\bmod 2^n}, \quad r = 0,1 \tag{3}$$

The 8×8 2D DBM is shown in Fig. 2.

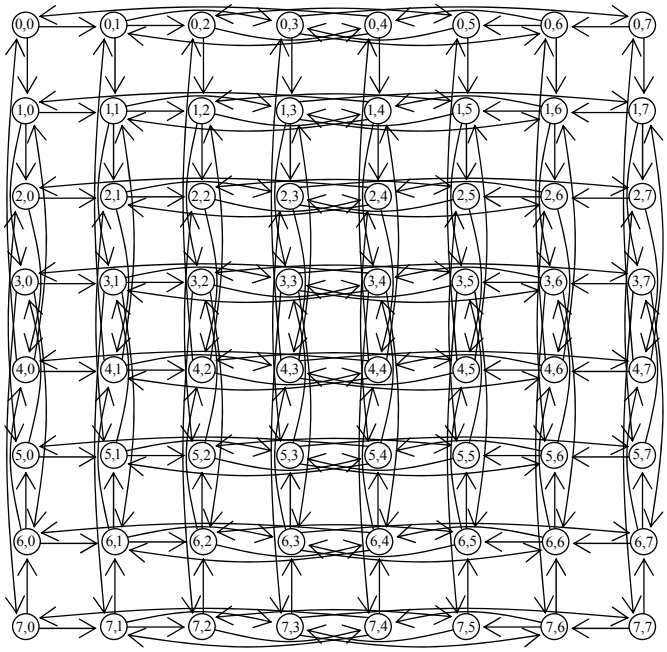


Fig. 2. A 2D DBM with 64 nodes composed from eight 8-node de Bruijn networks (as shown in Fig.1. a) along each dimension

The 2D DBM networks have some interesting topological properties that motivate us to consider them as a suitable candidate for on-chip network architectures. The most important property of 2D DBM networks is that while the number of links in a 2D DBM and an equal-sized mesh are exactly the same, the network diameter of this network is less than that of the

mesh. More precisely, the diameter of a 2D DBM and a mesh are $2log\ N^{0.5}$ and $2(N^{0.5}-1)$, respectively, where N represents the network size.
Although, establishing the new links removes the link between some adjacent nodes (for example 1 to 0, 2 to 1 and 3 to 4 connections in Fig. 1) and increases their distance by one hop. In this network, however, the distance between many nodes is decreased by one or multiple hops, compared to a mesh, and this can lead to a considerable reduction in the average inter-node distance in the network.
The 2D DBM links are unidirectional and at most 8 unidirectional links are used per node. This is equal to the number of links connected to a node in a mesh (which is 4 bidirectional links). Since the node degree of a topology has an important contribution in (and usually acts as the dominant factor of) the network cost, the proposed topology can achieve lower average distance than a 2D mesh while it has almost the same cost. However, we will discuss the area overhead due to longer links in 2D DBM in next sections.

2.3 Routing Algorithm

During past years, a number of routing algorithms have been developed for the original de Bruijn network. Ganesan (Ganesan & Pradhan, 2003) proposed a routing algorithm which routes the packets toward the destination by changing one bit at a time, starting from the most significant bit of an n-bit address in a network of size 2^n. At the i^{th} step of this algorithm, the n-i^{th} bit of the destination address is compared to the MSB of the current address. If they are equal, the message is routed over the shuffle channel, to keep the bit unchanged and rotate the address. Otherwise, the message is routed over the shuffle-exchange channel to make the two bits identical and then rotate the address (self-loops are avoided). This algorithm involves a maximum of n steps. In order to be deadlock-free, this algorithm requires n virtual channels and the message uses the i^{th} channel at the n-i^{th} step. Since in this virtual channel selection scenario routing is performed in a descending order regarding channel numbers, the dependency graph of virtual channels is acyclic and the routing is deadlock-free (Dally & Seitz, 1987).
Ganesan (Ganesan & Pradhan, 2003) splits the networks into two trees: T1 and T2. A message is routed between T1 to T2, and then in T2, and then the message is routed between T2 to T1 and then in T1. Therefore, this routing algorithm has four different steps, which may also decrease, depending on the source and destination nodes. The algorithm has two phases and initiates with phase 0 (using virtual channel 0). When the message goes through T2 to T1, the phase number increases (virtual channel 1 is used). Ganesan (Ganesan & Pradhan, 2003) proved that this method is deadlock-free. The two trees, T1 and T2, are depicted in Fig. 3 for N=8.

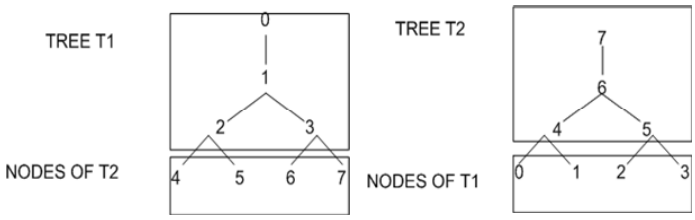


Fig. 3. Trees T1 and T2 for N=8

Park (Park & Agrawal, 1995) has deformed the de Bruijn as two graphs, increasing and decreasing. In the i^{th} stages, the MSB of the current node is compared with the n-i^{th} bit of the destination node, and if they are the same, shuffle cycle is used; otherwise, shuffle-exchange is used. If the path switches from increasing graph to the decreasing graph, the virtual channel number increments by 1. It is proved that the mentioned algorithm for the de Brujin network is deadlock-free (Park & Agrawal, 1995). Park (Park & Agrawal, 1995) has shown that for a de Brujin network with N nodes (N=2^n) this kind of routing requires

$$\text{Number of V.C.} = n - \left\lfloor \frac{n-1}{2} \right\rfloor \tag{4}$$

virtual channels to ensure deadlock freedom.
So, for N=8, it requires 2 virtual channels. For N=16 nodes, (n=4), based on equation (4), the number of virtual channels needed for a deadlock-free routing is equal to 3. Virtual channel 0 is more crowded than the other virtual channels. The routing algorithm uses virtual channels unevenly and very few packets use all the virtual channels.
We revised the routing algorithm to balance the use of virtual channels. In our proposed solution, at each node, the header flit has a degree of flexibility in selecting virtual channels, as used in (Kiasari et al., 2005). For example, for N=16, some of the source nodes may use only 1 virtual channel. Some nodes use 2 virtual channels and a few source nodes use 3 virtual channels. For the last group (those use 3 virtual channels), virtual channel 0 is selected in the source node. For the second group, we start with virtual channel 1 and for the first group we start from virtual channel 2. If virtual channel 2 is occupied, then we can use virtual channel 1 and finally virtual channel 0 is chosen. Using this method, virtual channels are used uniformly.
In this chapter, for routing in the 2D DBM, we use the revised method of Park (Park & Agrawal, 1995) in each dimension and we choose the paths and nodes in a way that the routing is minimal. Like XY routing in mesh networks, the deterministic routing first applies the routing mechanism in rows in order to deliver the packet to the column at which the destination is located. Afterwards, the message is routed to the destination by applying the same routing algorithm in the columns. Obviously, adding the second dimension in this routing scheme does not generate a cycle and the whole routing in the 2D network is deadlock-free provided that the routing in each dimension is deadlock-free (Duato et al., 2005).

3. Simulation Results

3.1 Simulator

To simulate the proposed NoC topology, we have used an interconnection network simulator that is developed based on POPNET simulator (Popnet, 2007) with Orion power library embedded in it. Orion is a library which models the power consumption of the interconnection networks (Wang et al., 2002). Providing detailed power characteristics of the network elements, Orion enables the designers to make rapid power performance tradeoffs at the architecture level (Wang et al., 2002). As mentioned in (Wang et al., 2002), the total energy each flit consumes at a specified node and its outgoing link is given by

$$E_{flit} = E_{wrt} + E_{arb} + E_{read} + E_{xb} + E_{link} \quad (5)$$

It consists of five components: 1) the power that is consumed for writing into buffers, 2) the power of arbiter, 3) the power that is consumed in reading from buffers, 4) the power of the internal crossbar, and 5) the power that is consumed in the links.

The POPNET simulator is only introduced for a two-dimensional mesh topology (Popnet, 2007). We have customized the simulator to support other topologies and other routing algorithms such as shuffle-exchange (Sabbaghi et al., 2008), and de Bruijn topologies. The power is obtained and reported for each layer and each component of the network.

We set the networks link width to 32 bits. Each link has the same bandwidth and one flit transmission is allowed on a link. The power is calculated based on a NoC with 90 nm technology whose routers operate at 250 MHz. Based on the core size information presented in (Mullins et al., 2006), we set the width of the IP cores to 2 mm, and the length of each wire is set based on the number of cores it passes. The simulation results are obtained for 8×8 and 16×16 mesh NoCs with XY routing algorithm, and 8×8 and 16×16 de Bruijn NoCs using the routing algorithms described in the previous section. The message length is assumed to be 32 and 64 flits and 2 and 3 virtual channels per physical channel are used. Messages are generated according to a Poisson distribution with rate λ. The traffic pattern can be *Uniform, Matrix-transpose*, and *Hotspot* (Duato et al., 2005).

3.2 Comparison Results

In this section, we evaluate the 2D DBM and compare it with the mesh, the most common topology for NoCs. Based on the necklace properties in de Bruijn layout (Chen et al., 1993), we have considered a more efficient layout for each row and column of the 2D DBM as shown in Fig. 4. With this new layout the total wire length used in the network is decreased. For example, for an 8×8 2D DBM about 25% reduction in total wire length is obtained and this can be increased to a 50% reduction for a 16×16 2D DBM.

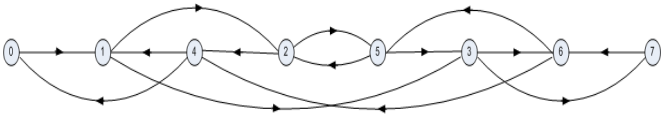


Fig. 4. A better node placement of de Bruijn network of 8 nodes

As mentioned before, we have also revised the routing algorithm in (Park & Agrawal, 1995) to have balanced use of virtual channels. Fig. 5 compares the performance of original routing algorithm and the new routing algorithm in the 8×8 2D DBM using 2 virtual channels with messages of 32 flits. As can be seen in the figure, the new algorithm exhibits better performance in terms of average message latency.

Figures 6-9 compare power consumption and the performance of simple 2D mesh and 2D DBM NoCs under various traffic patterns, network sizes and message lengths. In Fig. 6 and Fig. 7, the average message latency is displayed as a function of message generation rate at each node for the 8×8 and 16×16 networks under deterministic routing. As can be seen in the figures, the 2D DBM NoC achieves a reduction in message latency with respect to the popular 2D mesh network for the full range of network load under various traffic patterns (especially in uniform traffic). Note that for matrix-transpose traffic load, it is assumed that 30% of messages generated at a node are of matrix-transpose type (i.e. node (x,y) sends the message to

node (y,x)) and the rest of 70% messages are sent to other nodes uniformly. For hotspot traffic load a hotspot rate of 16% is assumed (i.e. each node sends 16% of its messages to the hotspot node (node (4,4) in 8×8 network and node (8,8) for 16×16 network) and the rest of 84% of messages are sent to other nodes uniformly). Note that increasing the network size causes earlier saturation in a simple 2D mesh.

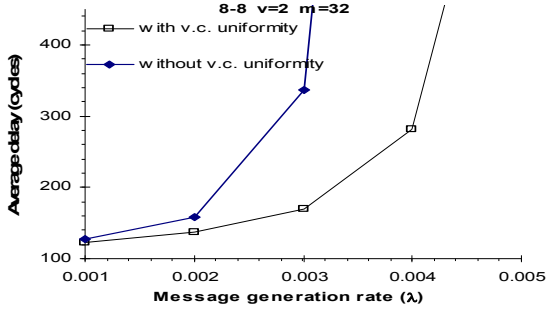
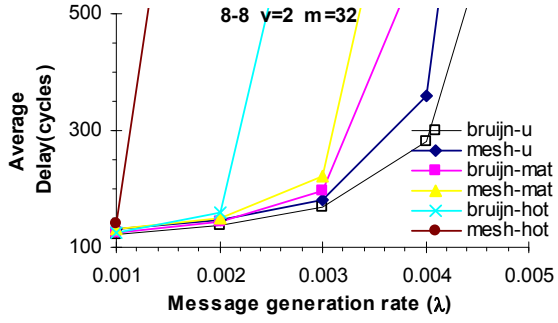
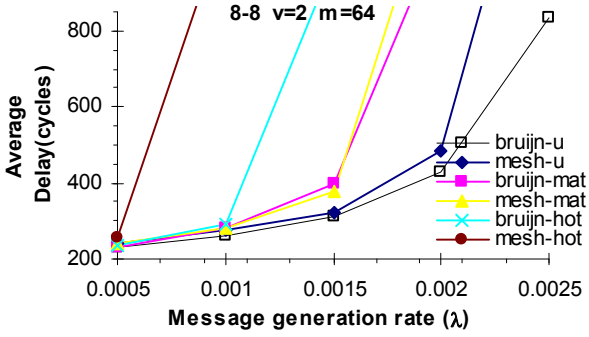


Fig. 5. The effect of balanced use of virtual channels

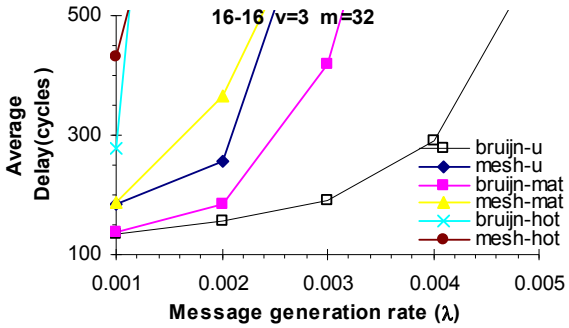


a)

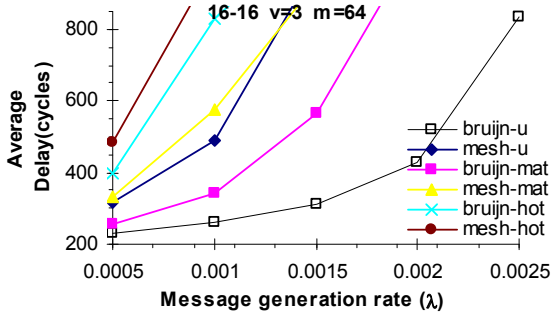


b)

Fig. 6. The average message latency in the 8×8 simple 2D mesh and 2D DBM for different traffics patterns with message size of (a) 32 flits and (b) 64 flits



a)

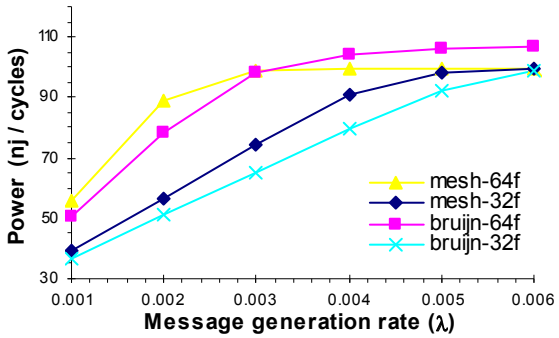


b)

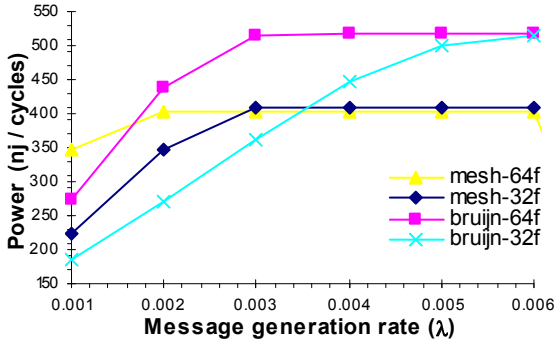
Fig. 7. The average message latency in the 16×16 simple 2D mesh and 16×16 network of 2D DBM for different traffics patterns with message size of (a) 32 flits and (b) 64 flits

According to the simulation results reported above, the 2D DBM has a better performance compared to the equivalent simple 2D mesh NoC. The reason is that the average distance a message travels in the network in a 2D DBM network is lower than that of a simple 2D mesh. The node degree of the 2D DBM and simple 2D mesh networks (hence the structure and area of the routers) are the same. However, unlike the simple 2D mesh topology, the 2D DBM links do not always connect the adjacent nodes and therefore, some links may be longer than the links in an equivalent mesh. This can lead to an increase in the network area and also create problems in link placement. The latter can be alleviated by using efficient VLSI layouts (Samanathan & Pradhan, 1989; Chen et al., 1993) proposed for de Bruijn networks, as we used.

Fig. 8 demonstrates power consumption of the simple 2D mesh and 2D DBM under deterministic routing scheme with uniform traffic. It is again the 2D DBM that shows a better behavior before reaching to the saturation point. Fig. 9 reports similar results for hotspot and matrix-transpose traffic patterns in the two networks.

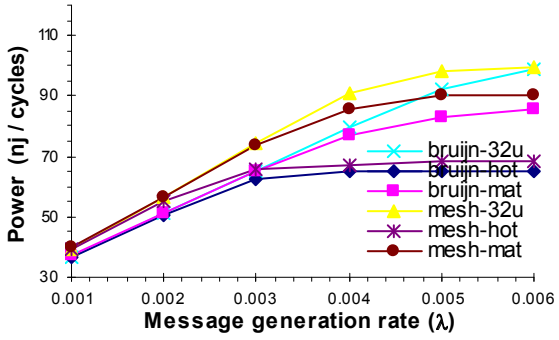


a)

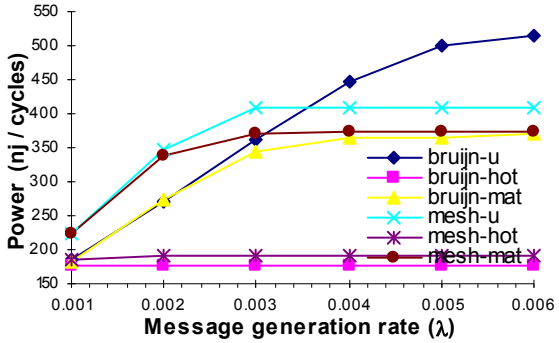


b)

Fig. 8. Power consumption of the simple 2D mesh and 2D DBM with uniform traffic pattern and message size of 32 and 64 flits for (a) 8×8 network and (b) 16×16 network



a)



b)

Fig. 9. Power consumption of simple 2D mesh and 2D DBM for different traffic patterns and message size 32 flits for (a) 8×8 and (b) 16×16 networks

The results indicate that the power of 2D DBM network is less for light to medium traffic loads. The main source of this reduction is the long wires which bypass some nodes and hence, save the power which is consumed in intermediate routers in an equivalent mesh topology.

Although for low traffic loads the 2D DBM network provides a better power consumption compared to the simple 2D mesh network, it begins to behave differently near heavy traffic regions.

It is notable that a usual advice on using any networked system is *not to take the network working near saturation region* (Duato et al., 2005). Having considered this and also the fact that most of the networks rarely enter such traffic regions, we can conclude that the 2D DBM network can outperform its equivalent mesh network when power consumption is considered.

The area estimation is done based on the hybrid synthesis-analytical area models presented in (Mullins et al. , 2006; Kim et al., 2006; Kim et al. 2008). In these papers, the area of the router building blocks is calculated in 90nm standard cell ASIC technology and then analytically combined to estimate the router total area. Table 1 outlines the parameters. The analytical area models for NoC and its components are displayed in Table 2. The area of a router is estimated based on the area of the input buffers, network interface queues, and crossbar switch, since the router area is dominated by these components.

The area overhead due to the additional inter-router wires is analyzed by calculating the number of channels in a mesh-based NoC. An $n \times n$ mesh has $2 \times n \times (n\text{-}1)$ channels. The 2D DBM has the same number of channels as mesh but with longer wires. In the analysis, the lengths of packetization and depacketization queues are considered as large as 64 flits.

In Table 3, the area overhead of 2D DBM NoC is calculated for 8×8 and 16×16 network sizes in a 32-bit wide system. The results show that, in an 8×8 mesh, the total area of the 2mm links and the routers are 0.0633 mm^2 and 0.1089 mm^2, respectively. Based on these area estimations, the area of the network part of the 2D DBM network shows a 44% increase compared to a simple 2D mesh with equal size. Considering 2mm×2mm processing elements, the increase in the entire chip area is less than 3.5%. Obviously, by increasing the

buffer sizes, the network node/configuration switch area increases, leading to much reduction in the area overhead of the proposed architecture.

Parameter	**Symbol**
Flit Size	F
Buffer Depth	B
No. of Virtual channels	V
Buffer area (0.00002 *mm²/bit* (Kim et al., 2008))	B_{area}
Wire pitch (0.00024 *mm* (ITRS, 2007)	W_{pitch}
No. of Ports	P
Network Size	N (= n×n)
Packetization queue capacity	PQ
Depacketization queue capacity	DQ
Channel Area (0.00099 mm²/bit/mm (Mullins et al. , 2006)	W_{area}
Channel Length (2mm)	L
No. Of Channels	$N_{channel}$

Table 1. Parameters

	Symbol	**Model**
Crossbar	RCX_{area}	$W^2_{pitch}\times P\times P\times F^2$
Buffer (per port)	RBF_{area}	$B_{area}\times F\times V\times B$
Router	R_{area}	$RCX_{area}+P\times RBF_{area}$
Network Adaptor	NA_{area}	$PQ\times B_{area} + DQ \times B_{area}$
Channel	CH_{area}	$F\times W_{area}\times L\times N_{channel}$
NoC Area	NoC_{area}	$n^2\times (R_{area}+ NA_{area})+ CH_{area}$

Table 2. Area analytical model

Network	Link Area	Router Area	Increase percent to mesh	increase percent in the entire chip
8×8 mesh	.06338	.1089	0	0
8×8 2D DBM	.1086	.1089	44.38	3.46
16×16 mesh	.06338	.1217	0	0
16×16 2D DBM	.1626	.1217	103.58	9.57

Table 3. 2D DBM area overhead

4. Conclusion

The simple 2D mesh topology has been widely used in a variety of applications especially for NoC design due to its simplicity and efficiency. However, the de Bruijn network has not been studied yet as the underlying topology for 2D tiled NoCs. In this chapter, we introduced the two-dimensional de Bruijn Mesh (2D DBM) network which has the same cost as the popular mesh, but has a logarithmic diameter. We then conducted a comparative simulation study to assess the network latency and power consumption of the two

networks. Results showed that the 2D DBM topology improves on the network latency especially for heavy traffic loads. The power consumption in the 2D DBM network was also less than that of the equivalent simple 2D mesh NoC.
Finding a VLSI layout for the 2D and 3D DBM networks based on the design considerations in deep sub-micron technology, especially in three dimensional design, can be a challenging future research in this line.

5. References

http://www.princeton.edu/~lshang/popnet.html, August 2007.

Chen, C.; Agrawal, P. & Burke, JR. (1993). dBcube : A New class of Hierarchical Multiprocessor Interconnection Networks with Area Efficient Layout, *IEEE Transaction on Parallel and Distributed Systems*, Vol. 4, No. 12, pp. 1332-1344.

Dally, WJ. & Seitz, C. (1987). Deadlock-free Message Routing in Multiprocessor Interconnection Networks, *IEEE Trans. on Computers*, Vol. 36, No. 5, pp. 547-553.

Dally, WJ. (1991). Express Cubes: Improving the Performance of K-ary N-cube Interconnection Networks, *IEEE Trans. on Computers*, Vol. 40, No. 9, pp. 1016-1023.

De Bruijn, NG. (1946). A Combinatorial Problem," *Koninklijke Nederlands Akademie van Wetenschappen Proceedings,* 49-2, pp.758–764.

Duato, J. (1995). A Necessary and Sufficient Condition for Deadlock-free Adaptive Routing in Wormhole Networks, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 6, No. 10, pp. 1055–1067.

Duato, J.; Yalamanchili, S. & Ni, L. (2005). *Interconnection Networks: An Engineering Approach*, Morgan Kaufmann Publishers.

Ganesan, E. & Pradhan, DK. (2003). Wormhole Routing in de Bruijn Networks and Hyper-de Bruijn Networks, *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 870-873.

ITRS. (2007). International technology roadmap for semiconductors. *Tech. rep.*, International Technology Roadmap for Semiconductors.

Kiasari, AE.; Sarbazi-Azad, H. & Rezazad, M. (2005). Performance Comparison of Adaptive Routing Algorithms in the Star Interconnection Network, *Proceedings of the 8th International Conference on High Performance Computing in Asia-Pacific Region (HPCAsia)*, pp. 257-264.

Kim, M.; Kim, D. & Sobelman, E. (2006). NoC link analysis under power and performance constraints, *IEEE International Symposium on Circuits and Systems (ISCAS)*, Greece.

Kim, MM.; Davis, JD.; Oskin, M & Austin, T. (2008). Polymorphic on-Chip Networks, *International Symposium on Computer Architecture(ISCA)*, pp. 101 -112.

Liu, GP. & Lee, KY. (1993). Optimal Routing Algorithms for Generalized de Bruijn Digraph, *International Conference on Parallel Processing*, pp. 167-174.

Louri, A. & Sung, H. (1995). An Efficient 3D Optical Implementation of Binary de Bruijn Networks with Applications to Massively Parallel Computing, *Second Workshop on Massively Parallel Processing Using Optical Interconnections*, pp.152-159.

Mao, J. & Yang, C. (2000). Shortest Path Routing and Fault-tolerant Routing on de Bruijn Networks, *Networks*, vol.35, pp.207-215.

Mullins, R.; West, A. & Moore, S. (2006). The Design and Implementation of a Low-Latency On-Chip Network, *Asia and South Pacific Design Automation Conference(ASP-DAC)*, pp. 164-169.

Ogras, UY. & Marculescu, R. (2005). Application-Specific Network-on-Chip Architecture Customization via Long-Range Link Insertion, *IEEE/ACM Intl. Conf. on Computer Aided Design*, San Jose, pp. 246-253.

Park, H.; Agrawal, DP. (1995). A Novel Deadlock-free Routing Technique for a class of de Bruijn based Networks, *IPPS*, pp. 524-531.

Sabbaghi-Nadooshan, R.; Modarressi, M. & Sarbazi-Azad, H. (2008). A Novel high Performance low power Based Mesh Topology for NoCs, *PMEO-2008, 7th International Workshop on Performance Modeling, Evaluation, and Optimization*, pp. 1-7.

Samanathan, MR.; Pradhan, DK. (1989). The de Bruijn Multiprocessor Network: a Versatile Parallel Processing and Sorting Network for VLSI, *IEEE Trans. On Computers*, vol. 38, pp.567-581.

Srivasan, K.; Chata, KS. & Konjevad, G. (2004). Linear Programming Based Techniques for Synthesis of Networks-on-chip Architectures, *IEEE International conference on Computer Design*, pp. 422-429.

Wang, H.; Zhu, X.; Peh, L. & Malik, S. (2002). Orion: A Power-Performance Simulator for Interconnection Networks, *35th International Symposium on Microarchitecture (MICRO)* , Turkey, pp. 294-305.

17

On the Efficient Design & Synthesis of Differential Clock Distribution Networks

Houman Zarrabi[1], Zeljko Zilic[2], Yvon Savaria[3] and A. J. Al-Khalili[1]
[1] *Department of Electrical and Computer Engineering, Concordia University*
[2] *Department of Electrical and Computer Engineering, McGill University*
[3] *Department of Electrical Engineering, École Polytechnique de Montréal*
Canada

1. Introduction

Almost all high-performance VLSI systems in today technologies are synchronous. These systems use a *clock* signal to control the flow of data throughout the chip. This greatly facilitates the design process of systems because it provides a global framework that allows many different components to operate simultaneously while sharing data. The only price for using synchronous type of systems is the additional overhead required to generate and distribute the clock signal.
Nearly all on-chip Clock Distributions Networks (CDNs) contain a series of buffers and interconnects that repeatedly power-up the clock signal from the clock source to the clock sinks. Conventionally, CDNs consisted of only a single stage buffer driving wires to the clock loads. This is still the case for clock distribution in very small scale systems; yet contemporary complex systems use multiple buffer stages. A typical clock tree distribution network in modern complex systems is shown in Figure 1. This design is based on the reported CDNs in (O'Mahony et al, 2003; Restle et al, 1998; Vasseghi et al, 1996).

1.1 Hierarchy in CDNs

The clock signal is generated with a Phase Lock Loop (PLL). A PLL is a control system that generates a signal having a fixed relation to the phase of its reference signal. A PLL circuit responds to both the frequency and the phase of its input signal and automatically raises/lowers the frequency of the controlled oscillator until it matches the reference (Wikipedia, 2009). The core clock signal is then amplified through the global buffer and distributed through a hierarchical network and buffers. The system CDN is generally defined to span from the PLL to the clock *pins*. The pin is the input to a buffer that locally amplifies and distributes the clock signal to clocked storage elements within a macro, the small blocks that make up a system. There can be any number of buffer levels between the PLL and the clock pin. In modern VLSI systems, there are up to four buffer levels. The last buffer level before the clock pin is generally called a sector buffer. This stage drives the interconnect leading to the macros and the local buffers at the pins. A synchronous VLSI

system has thousands of loads to be driven by clock signal. In CDNs, the loads are grouped together creating a (sub-) block. This trend results in a hierarchy in the design of CDNs including three different levels/categories of clock distribution namely as *global, regional* and *local* as shown in Figure 1. At each level of hierarchy there are buffers associated with that level to regenerate and to improve the clock signal at that level.

The global clock distribution connects the global clock buffer to the inputs of the sector buffers. This level of the distribution has usually the *longest path* in CDN because it relays the clock signal from the central point on the die to the sector buffers located throughout the die. The issues in designing the global tree is *mostly related to signal integrity* which is meant to maintain a fast edge rate over long wires while not introducing a large amount of timing uncertainty. Skew and jitter accumulate as the clock signal propagates through the clock network and both tend to accumulate proportional to the latency of the path. Because most of the latency occurs in the global clock distribution, this is also a primary source of skew and jitter (Restle et al, 2001). From a design point of view, achieving low timing uncertainty is the most critical challenge at this level.

The regional clock level is defined to be the distribution of clock signals from the sector buffers to the clock pins. This level is the middle ground between global and local clock distribution; it does not span as much area as the global level and it does not drive as much load or consume nearly as much power as the local level.

The local level is the part of the CDN that delivers the clock pin to the load of the system to be synchronized. This network drives the final loads and hence consumes the most power. As a design challenge, the power at the local level is about one order of magnitude larger than the power in the global and regional levels combined (Restle et al, 2001).

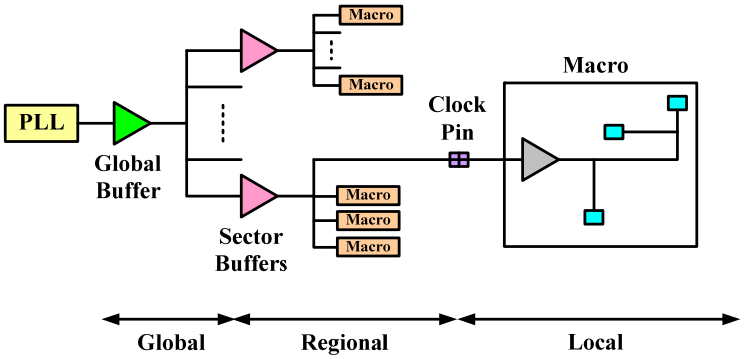


Fig. 1. A typical hierarchical CDN for a high-performance synchronous VLSI system

1.2 CDNs figures of merit

The main figures of merit for a CDN are the components of timing uncertainty, as well as, power consumption. All of these performance metrics have significant impacts on the design, evaluation and verification of synchronous system performance and reliability.

As mentioned previously, the advantage of a synchronous system is to regulate the flow of data throughout the system. However, this synchronizing approach depends on the ability to accurately relay a clock signal to millions of individual clocked loads. Any timing error introduced by the clock distribution has the potential of causing a functional error leading to

system malfunctioning. Therefore, the timing uncertainty of the clock signal must be estimated and taken into account in the first design stages. The two categories of timing uncertainties in a clock distribution are *skew* and *jitter*.

Clock skew refers to the absolute time difference in clock signal's arrival time between two points in a CDN. Clock skew is generally caused by mismatches in either device or interconnect within the clock distribution or by temperature or voltage variations around the chip. There are two components for clock skew: the skew caused due to the static noise (such as imbalanced routing) which is *deterministic* and the one caused by the system device and environmental variations which is *random*. An ideal clock distribution would have zero skew, which is usually unachievable.

Jitter is another source of dynamic timing uncertainties at a single clock load. The key measure of jitter for a synchronous system is the period or cycle-to-cycle jitter, which is the difference between the nominal cycle time and the actual cycle time. The first cycle, the period is the same as the clock signal period and the second cycle, the clock period becomes longer/shorter. The total clock jitter is the sum of the jitter from the clock source and from the clock distribution. Power supply noise may cause jitter in both the clock source and the distribution (Herzel et al, 1999).

Clock network also involves long interconnects which implies having lots of parasitics associated with the network contributing to the power consumption of the clock signal. Having the highest switching activity of the circuit in a chip is another fact of consuming a large amount of power of the system. This power consumption can be as high as 50% of the total power consumption of the chip according to (Zhang et al, 2000). The components of power consumption of CDN are: static, dynamic and leakage power. The power consumption due to the leakage current, in CDNs, is relatively small. In the same way, keeping the proper rise/fall times, minimizes the static power consumption. Thus the main portion of the power consumption is due to the dynamic power consumption. This is estimated as:

$$P=f\,C_L\,V_{dd}\,V_{swing}$$

in which f, C_L, V_{dd} and V_{swing} respectively represent frequency of the clock network, total load capacitances, supply-voltage and voltage-swing of clock signal. For the case of full swing (in which the clock signal swing reaches the voltage-supply level) V_{swing} is the same as V_{dd}. Accordingly, methods to reduce the power consumption are:

a. Reduce total load capacitances (C_L)
b. Reduce voltage-supply (V_{DD})
c. Reduce clock signal swing (V_{swing})

The intrinsic load capacitance relies on the process technology and there is no handy way to improve it. Yet, from the design aspects by breaking down interconnects by repeater insertion the total interconnect load is reduced. Worth mentioning that in coupled lines, the total load is greater than that of single-node lines, thus compensating design methods should be taken into consideration for power-saving improvement. Typically, power reduction is achieved by means of supply and/or swing voltage scaling in CDNs.

2. Differential Clock Distribution Networks (DCDNs)

In this section, based on the general overview given on CDNs, we will introduce the concepts and motivations toward the design of Differential CDNs (DCDNs). For this, we initially address the preliminaries needed for the design of DCDNs. These theories include differential signaling and differential signal integrity.

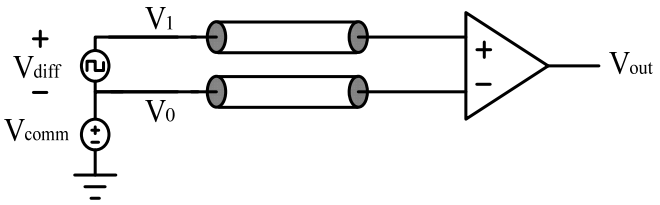


Fig. 2. Voltage-mode differential signaling

2.1 Preliminaries

2.1.1 Differential signaling

A digital signal can be transmitted *differentially* over the medium by utilizing two conductors. One of which is used for transmitting the signal and the other is used for the complement of the signal. Figure 2 shows a differential voltage-mode signaling system. To transmit logic '1', the upper voltage source drives V_1 and the lower voltage source drives V_0. For logic '0' transmission, the voltages are reversed.

As is shown in Figure 2, the following voltages are defined in a differential system: V_1 is the signal on the first line with respect to common return path, V_0 is the signal on the second line with respect to common return path, V_{diff} is the differential signal which is the voltage difference of the two signal pair, and, V_{comm} is the common voltage signal which is in common between both of signal pair. Differential signal V_{diff} carries the information and at the receiver the information is extracted from this voltage difference. In addition to the differential voltage there is a common-mode signal. This signal is used to give an initial biasing to the differential signal pair. In ideal conditions, the common-mode signal is constant and it does not carry any information. In this case:

$$V_{diff}=V_1-V_0$$

Differential signaling requires more routing and wires and pins than its single-ended counterpart system. In return for this increase, differential signaling offers the following advantages over single-ended signaling:

a. A differential system, serves its own reference. The receiver at the far end of the system compares the two signal pair to detect the value of the transmitted information. Transmitters are less critical in terms of noise issues, since the receiver is comparing two pair of signals together rather than comparing to a fixed reference. This results in canceling any noises in common to the signals.
b. The voltage difference for the two signal pair between logic'1' and '0' is:

$$\Delta V=2(V_1-V_0)$$

which is twice as much as is defined for a single-ended signaling system. This shows that *the noise margin of the differential system is twice as much as the single-ended signaling system*. This doubling effect of signal swing improves the speed of the signaling system. It affects the transition times (rise/fall time) which is done in half of the transition time of single-ended signaling system.

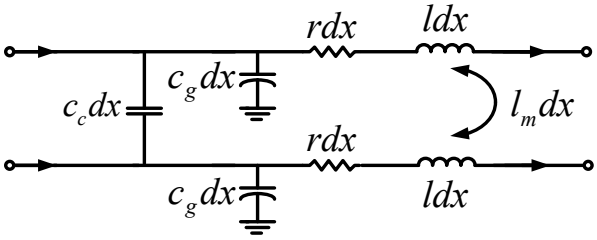


Fig. 3. A segment of a coupled interconnect

2.1.2 Differential signal integrity

In order to employ differential signaling, the coupled interconnects model is utilized and applied to the system. This type of interconnects not only have the intrinsic signal integrity issues, but also, they are involved with their mutual signal integrity aspects. In Figure 3, a segment of a coupled interconnect is shown.

The mutual parasitic elements are due to the adjacent line. These are mutual capacitance *Cc* and mutual inductance l_m in addition to the intrinsic parasitic elements *r*, *Cg* and *l* which indicate intrinsic resistance, capacitance and inductance of each line. The effective capacitance *Ceff* associated with each line, depending on the direction/mode of the signaling (in-phase or out-of-phase usually called *even* and *odd* mode respectively) can be calculated from the following equations (Hall et al, 2000):

$$C_{eff}(odd) = \eta Cc + Cg$$
$$C_{eff}(even) = Cg$$

And for effective inductance we have:

$$l_{eff}(odd) = l - l_m$$
$$l_{eff}(even) = l + l_m$$

As the above equations indicate, for the case of differential signaling (or out-of-phase signaling), the effective capacitance is increased by the factor of η due to coupling capacitances and the effective inductance is decreased due to the effect of mutual inductance. In (Kahng et al, 2000) it was shown that η has the value of {0, 2 and 3} depending on the mode of signaling and slew rates of the coupled signals. The typical value for η, for typical sharp input signals designs, is taken as 2.

2.1.3 Differential Buffers

The configuration of differential buffers is based on current steering devices, in which the output logic can be set by steering the current in the circuit. These devices are also considered as Current Mode Logic (CML) circuits. CML circuits are known to outperform the conventional CMOS circuits in Giga Hertz (GHz) operation frequency. A basic differential buffer is given in Figure 4. The current source in differential buffer is the tail current I_{ss}. When the common-mode voltage V_{comm} is applied to the differential buffer, due to the symmetry of the differential buffer, the current is split equally between the two wings ($I_{ss}/2$). Increasing one of the input voltages which implies the decrease in the other one, will result in increase in current of one branch and decrease in current of the other branch. Note that the total possible current to steer is I_{ss} and when one input voltage rises, the other one decreases by the same amount. When the input differential voltage $\Delta V=V_{in}-V'_{in}$ has passed a specific threshold, in other words when one of the transistors derives all the possible current from one branch the other transistors turn off, hence the output voltage reaches V_{dd} whereas the first branch drops to $V_{dd}-RI_{ss}$. Several differential loads also have been introduced in the literature (Dally et al, 1998). These loads may use resistor, current mirror and cross-coupled transistors. The differential load is characterized by its differential and common-mode impedances, known as r_Δ and r_c respectively. The differential impedance determines the change in the differential current I_Δ when the voltages on the two inputs of the terminal are varied in opposite directions. The common-mode impedance implies the average current changes when both input voltages are varied in the same direction. Depending on the type of application, the design may chose from these design options. Table I demonstrate the r_Δ and r_c for each load.

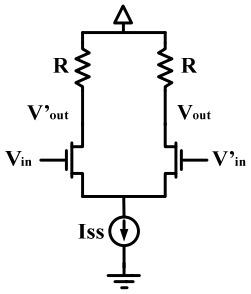


Fig. 4. A basic differential buffer

Load	r_c	r_Δ
Resistor	R	R
Current-mirror	$1/g_m$	$-1/\lambda I$
Cross-coupled	$1/g_m$	$1/g_m$

Table 1. Impedance of differential loads

2.2 Differential Clock Distribution Networks (DCDNs)

As discussed previously, differential signaling offers higher immunity against external perturbations. Due to the complexity increase and the need for error-free operation in contemporary systems, the idea of integrating differential signaling and clock distribution is seemingly becoming a viable solution for modern and for future IC designs.

Historically the idea of DCDN was to be utilized for off-chip clock distribution and for PCB-level synchronization. This technique was utilized to reduce and suppress the Electro-Magnetic Interference (EMI) of the neighboring circuits and systems waves. Due to the superiority of DCDN, recently there has been a couple of works on on-chip DCDN as well, such as (Sekar, 2002; Anderson et al, 2002). The idea of utilizing on-chip DCDN has not been widely used in the literature. In (Anderson et al, 2002) a DCDN is used in global level of the hierarchical CDN for Itanium Microprocessor. They reported that the use of DCDN has given the advantage of 10% less skew variation. In (Sekar, 2002) it is reported that DCDN has 25%-42% less sensitivity to power supply noises and 6% less sensitivity to manufacturing variations when they utilized H-Tree DCDN.

A general model of a DCDN is given in Figure 5. The DCDN is composed of a differential signal pair shown in two different patterns. The clock tree generally is a binary tree. The differential signal is dispersed along the clock network. Throughout the clock network at branching points the differential clock signals are regenerated by differential buffers to improve the signal integrity of the clock network. Finally at the last stage, they are all converted to single-ended signals for compatibility with the rest of the system functionality, which normally use single-ended signals. For the regenerative buffers a simple differential buffer introduced in the previous part can be utilized. The only design issue related to the buffer is the choice of differential loads. Based on the process technology, or design criteria, this item can be chosen from the design library. For final stage converters, usually the choice of current mirror load is the superior choice. As Table 1 demonstrates, current mirror loads have high differential output impedance which results in fast change in the output that is used to drive the output of the clock network.

Differential clocking eliminates the induced crosstalk due to aggression of clock signals. Clock signal is spread all over the chip area. It also has full switching activity. Also device sizes tend to shrink as technology advances. These facts show that as technology advances the clock signal aggression can be quite harmful for all system components all over the chip area. Distribution of clock with differential signals eliminates this problem to a certain extent, as both positive and negative signal values are applied and the noise would be cancelled. Furthermore, as given in (Anderson et al, 2002), DCDN offers less skew variations in the presence of external noises; it has less sensitivity in presence of supply and process variations (Sekar 2005).

The aforementioned points are of the most important criteria/solutions for reliable system design. Due to technology advances and increase in system complexity, the design with low or no parameter variation in ideal case, has become the most concerning issue. Timing error results directly in system malfunctioning. Thus designing a reliable and noise tolerant, clock distribution may help significantly for a reliable system design. As introduced in the literature, DCDN has these potentials; thus this design methodology can be a solution for future robust system design.

Plus the pros and cons of DCDN, there are some design/synthesis challenges associated with the efficient design of DCDNs. Some of most challenges may be summarized as:

- Differential signaling is involved with higher parasitic, due the existence of coupled lines. In this case the total power consumption is commonly increased.
- Coupled lines are commonly routed and synthesized using symmetrical path models (which is not the general case).
- Routing complex DCDNs may take too much computation time. Using existing routing methods is not time efficient.

Proposing solutions to address the above challenges can efficiently help the design and synthesis of DCDNs, needed for modern complex VLSI technologies. These solutions are given in the following sections.

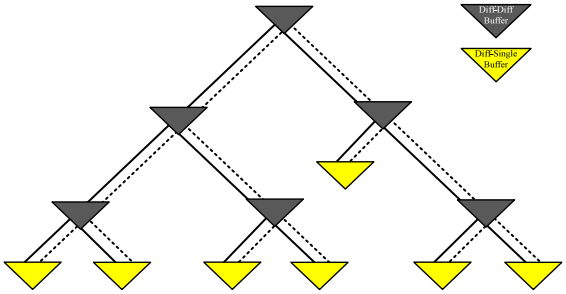


Fig. 5. A general structure for DCDN

3. Efficient design of DCDNs

3.1 Dynamic Threshold (DT) MOS for low-voltage DCDNs

For the design and synthesis of CDNs, buffers are inserted to improve the performance of CDNs in order to reduce the overhead capacitances. In this part, Dynamic Threshold (DT) (sometimes referred to as Variable Threshold) transistors (Assaderaghi et al, 1997) are utilized in conventional differential buffer structures. These transistors outperform the conventional transistors in low voltage applications which are suitable for advanced low voltage technologies. The use of DT transistors helps improve the buffer performance. DT transistors switch faster since their threshold voltage decrease dynamically when the input is applied to their gate terminal due to body effect. Such buffers are depicted in Figure 6 (Zarrabi et al, 2006). Figure 6(a) presents a Low-Swing:Low-Swing differential buffer. DT transistors help improve the speed of these buffers when low swing inputs are applied to the buffer. The use of cross-coupled differential load with high differential impedance helps to have a fast transition of the inputs to the outputs. Figure 6(b) represents a Low-Swing: Full-Swing level converter. This buffer is used at the sinks to restore clock signals to their single-node full amplitude. The current-source pull-ups help to have asymmetrical fast transformation of differential to single-ended signals. The structure is based on Chappell amplifier which offers good common-mode noise rejection (Chappell et al, 1998).

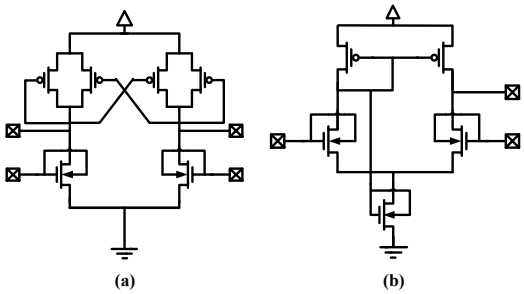


Fig. 6. (a) Low-Swing: Low-Swing (b) Low-Swing: Full-Swing DT differential buffers

3.2 Differential low-power buffers

Recalling from Section 2.1.1, two voltage components are associated with differential signaling: common-mode and differential. Common-mode also refers to DC voltage biasing and is the voltage used for initial biasing of the differential buffer.

In order for receivers (differential to single-ended converters) to operate efficiently and have a full/proper output swing, the common-mode voltage or DC biasing of the differential buffer should be low enough to turn the input transistors off. In the literature, the method used in order to overcome this issue is to increase the voltage swing as much as possible to be able to decrease the common-mode voltage to the sufficient supply level (usually used differential voltage of 50% of V_{dd}) (Anderson et al, 2002; Sekar, 2005). This method results in high power consumption in DCDN. Recalling from Section 2.1:

$$V_{diff\_low}=V_{dd}-RI_{ss}$$

The above equation implies that in order to increase the differential voltage swing, the tail current need to be increased. This technique largely affects the power consumption in DCDN. Note that, it is not possible to touch the load (R) as it directly affects the bandwidth of the clock network. Therefore, in previous works, in order to reach sufficient output swing, the differential voltage swing is increased to reduce common-mode voltage. Correspondingly, a circuit technique is proposed to address this design problem.

The proposed technique for differential receiver is given in Figure 7 (Zarrabi, 2006). The buffer configuration is based on Chappell amplifier as introduced in the previous section. Attached to the buffer are the level-shifting circuits. The buffer functionality is as follows:

The dashed parts in Figure 7 are the level shifters (also referred to as source followers) (Razavi, 2001). When the input is applied to the gate terminals of the level shifters, the outputs are dropped and follow their inputs. In other words, the voltage gain equals one (no voltage amplification), and the following relations are applicable (Broderson, 2005):

$$I=(\beta/2)(V_{IN}-V_{OUT}-V_T)^2$$
$$V_{OUT}-=IR_s$$
$$V_{OUT}=(R_s\beta/2)(V_{IN}-V_{OUT}-V_T)^2$$
$$V_{IN}=V_{OUT}+V_T+[(V_{IN}-2)/(R_s\beta)]^{0.5}$$

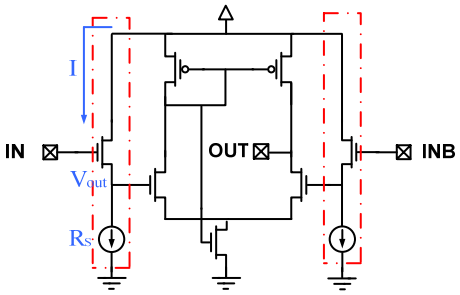


Fig. 7. Differential receiver with level shifter

The last result shows that V_{OUT} can be derived by solving the final equation iteratively. However, by making the first order approximation that R_S is large enough (especially in current sources) to make the third term equal to zero, we can conclude:

$$V_{OUT} = V_{IN} - V_T$$

This shows that the output of the source follower circuits copies the input of the gates with a shift of a transistor threshold which is a technology dependent factor. The transistor ratios for buffers sizing are the same as the ones given in (Chappell et al, 1998). However, the total size of the buffer is scaled to minimize the skew.

The above configuration for the differential receivers helps lower the common-mode (DC bias) of the internal input transistors of the receiver. Utilizing this design technique, it is possible to further reduce the differential voltage swing while maintaining a sufficient output swing at the final nodes.

In order to perform differential voltage scaling in DCDN, previously a new design for level converter was given. For the case of intermediate buffers, in order to be able to vary the differential voltage while maintaining the linearity of the buffer, the differential load should be reconfigured in a way to establish this design goal. In this part, a new configuration for differential load is proposed which enables us to have linearity in the buffer. Figure 8 shows the proposed buffer configuration.

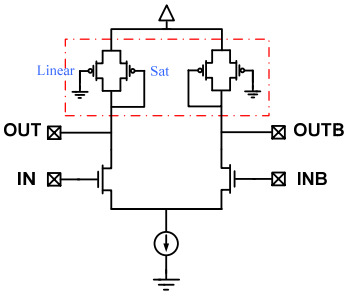


Fig. 8. Differential buffer with composite load

The dashed part demonstrates the proposed composite configuration of the differential load. Such composition enables the circuit to combine both the characteristics of the diode connected device and triode transistor together to have a linear operating load in various

voltage ranges (Dally et al, 1998). The proposed buffer based on composite differential load is a technology portable design and can be used in any available design process whereas the use of resistance is limited to current and future advanced technologies. This portable design method comes at the price of increase in area and parasitic elements. The transistor ratios (for buffers sizing) are 1 to 3 which refer to the ratio of pull up to pull down transistors ($L=2L_{min}$ to reduce the channel length modulation effect). The total size of the buffer is scaled to reach the objective frequency of operation.

4. Efficient synthesis of DCDNs

4.1 Zero skew DCDN routing

As seen in the overview section, DCDNs are commonly routed assuming symmetrical CDN path models. This however is not the general case. Here we will propose a method for zero skew routing of DCDNs applicable to general (asymmetric) path models. In the literature, especially in (Cong et al, 1996) a comprehensive study on efficient clock routing is studied. Tsay's method (Tsay, 1991) is one of the methods introduced for zero skew routing of clock trees. In order to route differential clock trees with zero skew characteristic, the existing methods are modified to satisfy design objectives. To achieve this aim here in this part, a line equivalent delay model is utilized. The model is applied to Tsay's method (Tsay, 1991) for zero skew routing of differential clock trees (DCDNs).

4.1.1 Utilizing Tsay's method

In this method, zero skew is achieved by locating tapping points throughout the clock tree. Tapping points are the branching points at which sub-trees are chosen to maintain equal delay as shown in Figure 9.

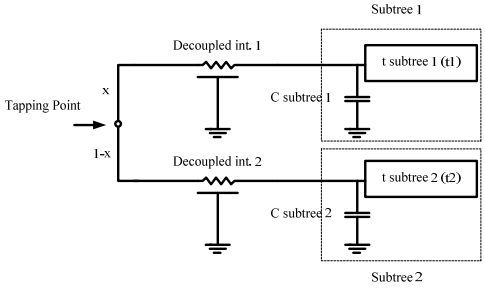


Fig. 9. Tapping point extraction through merging decoupled sub-tree(s)

As was seen in Section 2.1.2, the effective capacitance associated with each segment of a coupled line, considering both intrinsic and mutual effects is:

$$C_{eff}(odd) = \eta Cc + Cg$$

The effective capacitance is applied to both signal lines independent of each other; in this way, we call these lines as decoupled lines and we name this model as *decoupled line* model. This model is employed for the purpose of clock routing. In this way, the decoupled *RC-Π* delay model is used to model interconnects. The methodology of tapping extraction is as

follows. Figure 9 shows a schematic of a decoupled clock tree branch in which each line of the branch is a decoupled distributed *RC* model connected to its sub-tree child, for which the distributed line propagation delay is given by $t_{int}=0.37R_{int}C_{eff}$. Each sub-tree is modeled by a total capacitance $C_{subtree}$ and total propagation delay $t_{subtree}$ as shown in Figure 9. Considering tapping location x, to satisfy the equality of the two branch delays, the following equation is realized:

$$t_{int1}+0.74R_{int1}C_{effsubtree1}+t_1= t_{int2}+0.74R_{int2}C_{effsubtree2}+t_2 \quad *$$

In the second part of the equality, since the interconnect resistance combined with sub-tree capacitance creates a *Lumped* loop, it has the lumped propagation delay of $0.74R_{int}C_{subtree}$. Rewriting interconnect parasitics by per unit length parameters, we have:

$$R_{int1}=r_0xl,\ C_{int1}=c_0xl$$
$$R_{int2}=r_0(1- x)l,\ C_{int1}=c_0(1- x)l$$

in which r_0 and c_0 are the resistance and capacitance per unit length of the wire, l is total interconnection length between the two sub-trees and tapping location x. Solving Equation * with respect to x results into:

$$x=[1.35(t_{int2}-t_{int1})+r_0l(C_{effsubtree2}+0.5c_0l)]/[r_0l(C_{effsubtree1}+c_0l+C_{effsubtree2})]$$

In case of ($x \leq 0$ or $x \geq 1$), elongation would be needed. Elongation is the process of adding extra wire length to the sub-tree which has less effective capacitance, in order to equalize the delay of both sub-trees. The length of elongation to maintain zero skew is given by:

$$L'=[-20r_0C_{effsubtree2}+2(100r_0^2C_{effsubtree2}^2+270r_0c_0(t_{int2}-t_{int1}))^{0.5}]/[20r_0c_0]$$

This methodology is applied for zero skew routing in DCDNs. The results are given in Section 5.1 will validate the efficiency of this methodology.

4.2 Parallel synthesis of DCDNs

CDN synthesis is one of the primary time-consuming steps, performed in the synthesis flow of VLSI systems. Especially with the growth of complex SoCs in current advanced technologies, this part has become more complicated and less computational cost-effective. Many efforts have been put into parallel computer aided design, all with the goal of reducing the computation time. In literature, methodologies have been proposed for parallel synthesis of CDN such as the ones proposed in (Banerjee et al, 1992; Banerjee, 1994). These methods however, focus mainly on the single-ended clock tree structure. In this section, the goal is to leverage distinctive features of parallel computation to reduce computational time required to synthesize DCDNs (Zarrabi et al, 2007). The methodology utilizes and extends the technique proposed in Section 4.1, to synthesize zero skew DCDNs in parallel. This is a flexible methodology, applicable to symmetric/asymmetric and hybrid (differential and/or single-ended) clock tree structures.

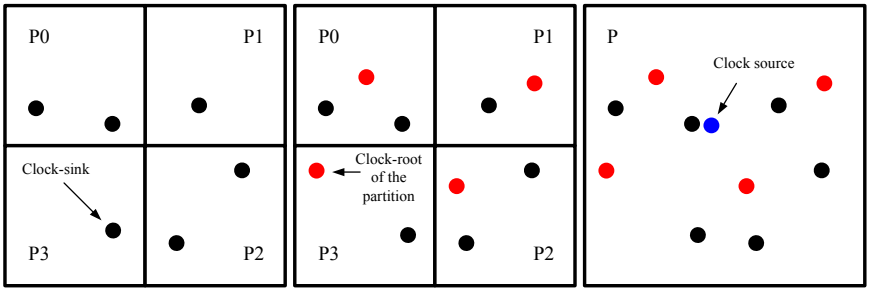


Fig. 10. Parallel DCDN distribution: a) partitioning the die area into sub-regions, b) locating the clock-root of each region, c) finding the source of the clock network

The methodology for parallel synthesis of zero skew DCDNs is as follows. Initially the total chip area is partitioned into sub-regions (partitioning phase). Later, synthesis of zero skew differential clock distribution networks is performed on each of the partitioned regions (local clock distribution phase). In the final stage, the global differential clock network is routed for each of the previously-extracted clock-roots of the sub-regions (global clock distribution phase). The obtained source of the clock network can end up anywhere in the whole chip area (Manhattan surface), regardless of the initial partitioning. The proposed scenario is illustrated in Figure 10. The proposed method may be implemented using C++ language and the Message Passing Interface (MPI) platform (MPI). A pseudo-code describing the method is given in Figure 11.

A possible negative side effect of parallel synthesis is the increase in the total wire-length in the clock network. This could be interpreted as the impact of multi-stage distribution of the clock network which results in initial local zero-skew clock networks and a final global clock network routed on top of regional clock networks. In general, this parallel processing approach results in a clock-tree different from the one routed in a single step, due to die area partitioning; thus, the characteristics of the new clock tree such as total wire-length and skew may be slightly different. This proposed methodology is flexible, as it allows having a hybrid (differential and/or single-ended) distribution of the clock network. The global CDN could be differential, while the local (lower levels) CDNs could be single-ended to alleviate routing complexity. It is possible to enhance the global/local distribution algorithm with refined interconnect models. This methodology is also applicable to all symmetric/asymmetric clock-trees.

Parallel Zero Skew Differential Clock Distribution
(Clock-sinks, Number of Processing-nodes)
{
1. Partition chip area according to the number of **processing nodes**.
2. Apply '**local**' zero skew (differential) clock distribution to the partitioned areas and **send** the clock-tree **root(s)** to the root processing node.
3. Receive the processed clock-tree root(s) from processing nodes, and, apply '**global**' zero skew (differential) clock distribution.
4. Return the obtained final clock-tree root as the **source** of the (differential) clock distribution network.
}

Fig. 11. Pseudo-code for parallel synthesis of zero skew differential clock distribution

5. Results

In this section, the quantitative results related to the given design and synthesis methods for DCDNs are given.

5.1 Zero skew routing

The zero skew routing method, inspired by Tsay's algorithm and given in Section 4.1, was applied to IBM benchmarks r1…r5 (Tsay, 1991). Modified Tsay's method (for differential signal integrity) was implemented using C++ for clock routing, and PERL language for netlist manipulation was utilized for design and simulations. HSPICE simulation results for the proposed method are tabulated in Table 2. The delay and skew results presented throughout this work represent the average and absolute difference of clock signal phase delay at sink nodes respectively.

Two methodologies were used for routing differential lines: Single-Spaced (SS) and Double-Spaced (DS) routing. In single-spaced routing scheme, the mutual coupling effects are stronger, therefore differential characteristics of the pair is more dominant. DS offers smaller mutual coupling, consequently this reduces delay while degrading noise immunity. This is due to the fact that the stronger the coupling effect is, the stronger common-mode noise rejection becomes.

Table 2 demonstrates that, on average, clock trees generated with the proposed model show 97% skew reduction compared to those obtained using Elmore model, i.e. neglecting coupling effects. This improvement is achieved because coupling effects of differential lines are more accurately considered in the algorithm leading to tapping point selection. As technology advances the coupling effects increase and we are no longer able to neglect these effects in system modeling. In this case, neglecting the coupling effects, results in the misplacement of tapping points and reduces the effectiveness of the considered zero skew DCDNs. Simulation results also show smaller delay and skew for the DS scheme due to reduced coupling, however this design strategy as we will see degrades robustness in presence of external noise.

Bench Mark	Single Spaced (SS)		Single Spaced (SS)		Double Spaced (DS)	
	Elmore	Proposed	Proposed	Elmore	Proposed	Proposed
	Skew (ps)	Delay (ns)	Skew (ps)	Delay (ns)	Skew (ps)	Delay (ns)
r1	115	1.1	5.9	1.1	2.4	0.9
r2	199	3.8	15	3.7	8.8	3.3
r3	341	5.1	14	5.1	8.0	4.6
r4	759	17.5	36	17.1	20	14.5
r5	1825	34	51	34	39	28

Table 2. Skew and delay of DCDN for r1-r5 benchmarks in 180nm technology

5.2 Applying DT buffers

Buffers were inserted to improve the performance of benchmark clock networks. The buffer insertion procedure is as follows. Low-swing buffers were inserted at branching points and level converting buffers were inserted at sinks. Low-swing buffers are sized to reduce propagation delay throughout the clock network. To accomplish this, a base size for buffer according to its delay-power characteristic diagram was chosen to drive a unit length interconnect segment. Further, the buffers were uniformly sized according to the longest interconnect. Full-swing level converters (differential to single-ended) were composed of minimum size transistors to reduce the power consumption and skew reduction; their sizes were scaled up relative to their load capacitance.

Bench Mark	Low-Swing Conventional		Low-Swing Proposed		Full-Swing Conventional	
	Skew (ps)	Delay (ns)	Skew (ps)	Delay (ns)	Skew (ps)	Delay (ns)
r1	695	9.6	545	7.0	71	1.1
r2	844	10	679	7.7	163	1.3
r3	808	10	667	7.8	127	1.3
r4	1388	11.9	981	8.8	379	1.6
r5	1566	13.1	1135	9.6	532	1.9

Table 3. Buffered DCDNs (Full-Swing Vdd=1.8V, Low-Swing Vdd=0.5V)

Table 3 shows the skew and delay difference for similarly sized, buffered DCDN based on the conventional (Dally et al, 1998) and the proposed buffers. It shows 25% delay and skew improvement on average compared to conventional buffers in low-swing differential clocking scheme. Results show that the delays are reduced significantly while skews are degraded as compared with un-buffered DCDNs. It is believed that skews in buffered clock networks can be reduced significantly by enhancing the process of buffer insertion. For instance, differential buffers delay model should be considered when tapping points are selected in the zero skew DCDN design algorithm.

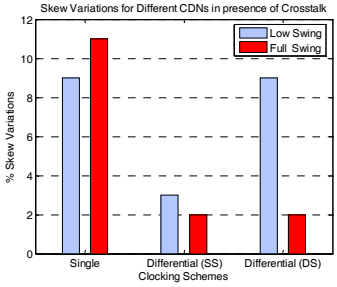


Fig. 12. Skew variations due to crosstalk

With regards to the skew sensitivity of the proposed DT DCDNs, two types of external aggressors, resulting into random skew are investigated: power-supply variations and crosstalk. Comparisons were made between similarly designed single-node CDN, single-

spaced DCDNs and double-spaced DCDNs (Figure 12). Benchmark r3 is used for simulations due to its average characteristic in terms of size and simulation time. For the low-swing scheme, the power supplies were V_{ddH}=1.8V & V_{ddL}=0.5V, whereas for full-swing scheme a single supply voltage (V_{ddH}=1.8V) was used. In those experiments, supply-voltages were varied by ±10%.

Simulation results show that for both clocking schemes, the single-spaced DCDN is the most robust design method in the presence of power-supply variations when compared to other CDNs. Skew variations increase when low-swing clocking is used. Double-spaced DCDN has less robustness to supply variations. DCDN is seen to have up to 25% less skew variations in low-swing clocking scheme and up to 9% less skew variations in full-swing clocking scheme than single-node CDN, in presence of power-supply variations.

Another source of perturbation that causes delay uncertainty in CDNs is crosstalk. For experiments, a full-swing aggressor is applied to one of the two big-child of the clock tree. The same low-swing and full-swing clocking schemes were considered. Simulation results show that the single-spaced DCDN shows 6% less skew variations when combined with low swing clocking scheme and 9% when combined with full swing clocking scheme as compared to single-node CDN subject to crosstalk.

5.3 Applying low-power buffers

In this part, the effect of employing low-power buffers introduced in Section 3.2 is studied. For comparison and performance evaluation of clock networks, a set of 400 MHz CDNs were designed for benchmark r3 due to its average size suitable for simulations. All reported designs were minimally sized to meet the target operating frequency (400MHz) based on 180nm technology parameters. In DCDNs, the differential signal swing was scaled by adjusting the tail current source of intermediate differential buffers. The lowest potential reached by either part of the differential signal is V_{dd}-RI_{ss} where V_{dd} is the supply voltage, I_{ss} is the tail current and R is the equivalent resistance of the transistor loads. Note that, the load resistance determines the bandwidth of the clock network; hence the only possible variable to tune is the tail current to scale the differential voltage swing. In the following, the effect of differential voltage scaling on power consumption and clock skew variations in the presence of power supply variations is explored.

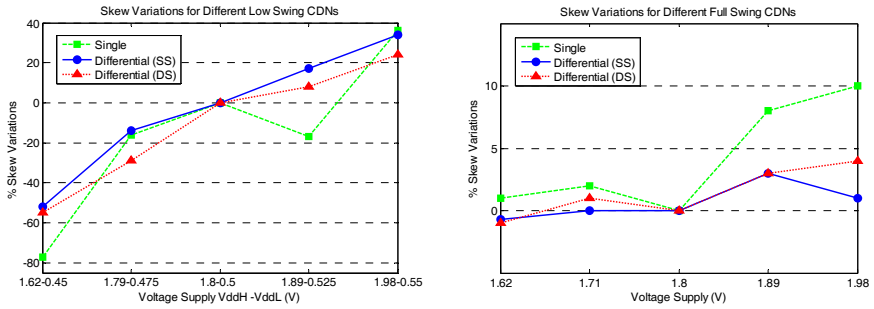


Fig. 13. Skew due to supply-voltage variations in low/full -swing schemes

5.3.1 Effect on power consumption

The tail current affects both the short circuit and dynamic power dissipation. Figure 14 demonstrates the effect of differential voltage scaling on the power consumption of DCDN and its single-node CDN counterpart both optimized for the same benchmark; which is obtained by scaling the tail current source of the differential buffer.

Figure 14 shows the power consumption obtained with three different clocking schemes: differential CDNs with composite-load buffers, differential CDNs with only grounded-gate load transistor buffers and the conventional single-node clocking scheme. The reason for exploring the network with only grounded-gate load transistor buffers is to demonstrate the effectiveness of composite-load buffers in terms of reduced sensitivity when large signal swings are considered. In this case, the size of the load transistors is increased (by approximately 20%) to get the same amplitude for a given tail current. The CDN used as reference is a conventional balanced single-node clocking tree, where transistors have the minimal size necessary to give full output swing while operating at the target frequency (400MHz).

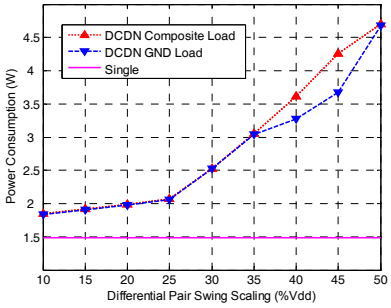


Fig. 14. Power consumption for voltage scaled DCDNs vs. a single-node CDN (r3).

Figure 14 shows that as the differential swing increases beyond 25% of supply-voltage (450mV, V_{dd}=1.8V), the power consumption increases drastically. This emphasizes the significant impact of large differential voltage swings on the power consumption of the clock network. For differential voltage swings below 450mV, the power consumption is not reduced much if the differential swing scaling is further reduced. A lower bound of 10% of V_{dd} was imposed to the differential swing to ensure a sufficient noise margin. Another consideration is that even with a differential swing as low as 10% of V_{dd} (180mV), the power consumption of the differential clock network remains almost 30% higher than that of single-node clock distribution network. Thus, trying to match the power dissipation of a single-node network by decreasing the swing of differential networks does not appear to be a viable option. A final observation from Figure 14 is that the DCDN with grounded-gate loads (GND) consumes less power over a limited region where the differential swing is large. However, as we will see in the following, this slight reduction in dissipated power comes at a large price in clock skew variability.

5.3.2 Supply-voltage scaling

In the previous section, we observed that the reduction of the differential swing has a strong impact on the power consumption. For differential swings smaller than 25% of V_{dd}, the power consumption becomes less than half of that observed when the clock network operates with the voltage swing of 50% of V_{dd}. Second, as Figure 15 suggests, due to the fluctuations of common-mode voltage in the clock network, the voltage swing of 25% of V_{dd} (450mV) shows the least skew when the clock network is subject to supply variations. We also observe that DCDN with composite load is more robust. This is due to the linear characteristic of composite load buffers, as was seen in Section 3.2.

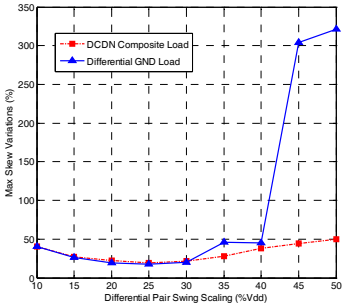


Fig. 15. Peak to peak skew variations in differential voltage scaled DCDNs.

Taking these considerations into account, we consider a design point for which the differential swing is 25% of V_{dd} (450mV) and we reduce the supply voltage to the point where we reach the same power consumption as that observed for the single-node and differential clock networks. HSPICE simulations demonstrate that for a supply voltage of 1.4V and differential swing 450mV, we obtain the same power consumption for the differential clock network. Yet, as can be observed from Figure 16, the variation of clock skew is still less than that of the comparable single-node CDN. Another interesting point that was observed during supply voltage scaling in DCDN is a *negligible signal latency difference.* This can be justified since as the tail current is lowered to achieve lower differential swings; the necessary differential voltage needed for the differential buffer to switch is also decreased. This enables the differential buffer to operate/switch faster than in the case where greater supply voltage with greater differential swing is used. Also as observed from Figure 16 and as discussed previously, the DCDN based on only grounded-gate loads is less resilient.

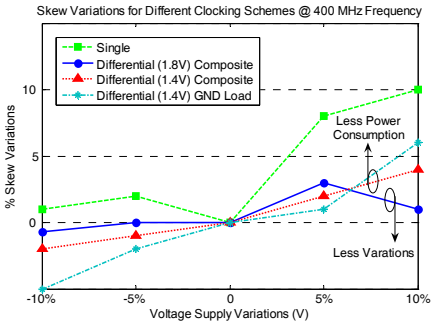


Fig. 16. HSPICE simulation show less variations in DCDN compared to single-node CDN for equal nominal power consumption.

5.4 Parallel zero skew routing

Benchmarks from (Tsay, 1991) with as many as 3101 clock sinks and total area as large as 1.4 cm * 1.4 cm were chosen. The reported results are based on 180nm CMOS technology file, for which interconnect parameters are: C_g= 8e-17(F/um), C_c= 8e-17(F/um) and R_{int}=0.022(Ω/um). The processing time, speed up and resulting simulated skews when synthesizing zero skew DCDNs using 1 (sequential), 2 and 4 processing nodes are reported in Table 4. These results demonstrate a *nearly-linear speed-up*. It is expected that for very large benchmarks, the speed-up grows *linearly* when the number of sinks is sufficiently large compared to the number of processing nodes. Thus, the processing time overhead due to performing that synthesis in parallel was almost negligible. Figure 17 confirms these observations for 2 and 4 processing nodes. As the number of clock sink nodes increases, the speed-up (dashed lines) converges to the maximum expected speed-up (rigid lines).

Benchmark (# of sinks)	Skew (ps)			Computation Time (s)			Speed-Up		
	1 PN	2 PN	4 PN	1 PN	2 PN	4 PN	1 PN	2 PN	4 PN
r3 (862)	14	12	14	0.70	0.48	0.21	1.0	1.45	3.24
r4 (1903)	36	35	36	1.60	0.90	0.46	1.0	1.76	3.46
r5 (3101)	51	49	52	2.74	1.45	0.73	1.0	1.89	3.74

Table 4. Run-time and speed-up results of benchmarks

In general, the parallel processing approach results in a clock-tree different from the one routed in a single step, due to die area partitioning; thus, the characteristics of the new clock tree such as total wire-length and skew may be slightly different. This proposed methodology is flexible, as it allows having a hybrid (differential and/or single-ended) distribution of the clock network. The global CDN could be differential, while the local (lower levels) CDNs could be single-ended to alleviate routing complexity. It is possible to enhance the global/local distribution algorithm with refined interconnect models. This methodology is also applicable to all symmetric/asymmetric clock-trees.

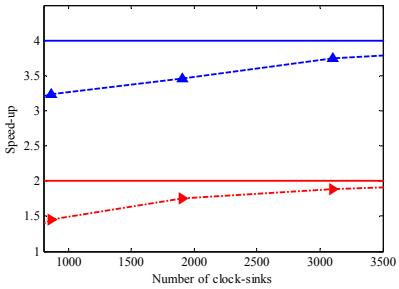


Fig. 17. Speed-up approaches its maximum, as the size of clock network increases, for 2 and 4 processing node synthesis cases.

6. Conclusions

In this chapter, some techniques for efficient design and synthesis of on-chip Differential Clock Distribution Networks (DCDNs) were given.

Initially design techniques were proposed that improve the performance of differential buffers which result into the performance improvement of DCDNs. This was achieved by means of introducing configurations for differential buffers based on Dynamic Threshold (DT) transistors. It was shown that for low supply-voltages, they outperform the conventional buffers with 25% delay reduction. Also, in order to overcome the high power consumption of DCDNs, a circuit configuration was proposed by which it is possible to reduce the differential voltage swings (down to 10% of V_{dd}) which reduces the power consumption significantly (30% more than single-node CDN). Furthermore, by scaling the supply voltage of the system from 1.8V to 1.4V, we reach a design point where the DCDN consumes the same power as its single-node CDN counterpart but has less variation (in terms of skew). This however comes at the expense of delay and reduced voltage swing.

Various synthesis techniques were introduced that improve the DCDNs routing to achieve low (and possibly zero) skew. For this, a line equivalent delay model was suggested by which it is possible to route DCDNs with low (zero) skew. On average, 97% skew reduction was obtained utilizing this model compared to the classic Elmore delay model. A methodology for parallel distribution (routing) of zero skew DCDNs was also proposed. The method is applicable to all symmetric/asymmetric clock networks with ability for hybrid implementation (differential and/or single-ended). The proposed method alleviates the problem of high computational cost of such CDNs in complex VLSI systems. Utilizing this method, nearly-linear speed-up is achieved for zero skew DCDNs.

In the hierarchy of CDNs in modern high-performance complex systems, DCDNs can be effectively fit in the global level of CDNs; yet they can be used as the sole solution to the clock distribution of the system, when noise is the main design issue.

7. References

Anderson, F. E.; Wells, J. S. & Berta, E. Z. (2002). The core clock system on the next generation Itanium microprocessor, in *ISSCC Digest of Technical Papers*, pp. 146-7.

Assaderaghi, F.; Sinitsky, D.; Parke, S.A.; Bokor, J.; Ko, P.K. & Hu, Chenming. (1997). Dynamic threshold-voltage MOSFET (DTMOS) for ultra-low voltage VLSI", IEEE Transactions on Electron Devices, Volume 44, Issue 3, pp.414 – 422.

Banerjee, Prithviraj. & Xing, Zhaoyun. (1992). A parallel algorithm for zero skew clock tree routing, International Symposium on Physical Design, pp. 118 – 123.

Banerjee, Prithviraj. (1994). Parallel Algorithms for VLSI Computer-Aided Design, PTR Prentice Hall, Englewood Cliffs, New Jersey 07632.

Broderson, Bob. (2005). Analog Integrated Circuits, online material, available: http://bwrc.eecs.berkeley.edu/People/Faculty/rb/.

Chappell, B.A.; Chappell, T.I.; Schuster, S.E.; Segmuller, H.M.; Allan, J.W.; Franch, R.L. & Restle, P.J. (1988). Fast CMOS ECL receivers with 100-mV worst-case sensitivity, IEEE JSSC, Volume 23, Issue 1, pp:59 – 67.

Cong, J.; He, L.; Koh, C. K. & Madden, P. (1996). Performance Optimization of VLSI Interconnect Layout, *Integration, the VLSI Journal*, vol. 21, pp. 1-94.

Dally, William J. & Poulton, John. (1998). Digital Systems Engineering, Cambridge University Press.

Hall, S.H.; Hall G.W. & McCall, J.A. (2000). High-Speed Digital system Design, A Handbook of Interconnect theory and Design Practices. John Wiley & Sons INC.

Herzel, F.; Razavi, B. (1999). *A study of oscillator jitter due to supply and substrate noise*, IEEE J. Circuits and Systems, Volume 46, pp. 56 - 62.

Kahng, A.B.; Muddu, S.; Sarto, E. (2000). On switch factor based analysis of coupled RC interconnects, Design Automation Conference, pp. 79 – 84.

MPI, Message Passing Interface, online: http://www.mpi-forum.org/.

O'Mahony, Frank P. (2003). 10GHz Global Clock Distribution Using Coupled Standing-wave Oscillators, PhD Dissertation, Stanford University.

Razavi, Behzad. (2001). Design of Analog CMOS Integrated Circuits. Mc Graw Hill.

Restle, P. J. & A. Deutsch (1998). Designing the best clock distribution network, in *Symposium VLSI Circuits Digest of Technical Papers*.

Restle, P.J.; McNamara, T.G.; Webber, D.A.; Camporese, P.J.; Eng, K.F.; Jenkins, K.A.; Allen, D.H.; Rohn, M.J.; Quaranta, M.P.; Boerstler, D.W.; Alpert, C.J.; Carter, C.A.; Bailey, R.N.; Petrovick, J.G.; Krauter, B.L. & McCredie, B.D (2001). A clock distribution network for microprocessors, *IEEE J. Solid-State Circuits*, vol. 36, no.5, pp. 792-799.

Sekar, D.C. (2005). Clock trees: differential or single ended? , International Symposium on Quality of Electronic Design, pp.548 – 553.

Tsay, R. S. (1991). Exact zero skew, in Proc. IEEE Int. Conf. Computer-Aided Design, pp. 336–339, Nov.

Vasseghi, N.; Yeager, K.; Sarto, E. & Seddighnezhad, M. (1996). 200-MHz superscalar RISC microprocessor, *IEEE J. Solid-State Circuits*, vol. 31, no. 11, pp. 1675-1685.

Wikipedia, online: http://en.wikipedia.org/wiki/Phase-locked_loop

Zarrabi, Houman. (2006). On the design and synthesis of differential clock distribution network, MASc Dissertation, Concordia University.

Zarrabi, Houman; Saaied, Haydar; Al-Khalili, A. J. & Savaria, Yvon. (2006). Zero Skew Differential Clock Distribution Network, International Symposium on Circuit And Systems (ISCAS), Greece, Island of Kos.

Zarrabi, Houman; Zilic, Zeljko; Al-Khalili, A. J. & Savaria, Yvon. (2007). A methodology for parallel synthesis of zero skew differential clock distribution networks, Joint Conference of MWSCAS/NEWCAS, Montreal, Canada.

Zhang, H.; Varghese George & Rabaey, J. M. (2000). Low-swing on-chip signaling techniques: effectiveness and robustness, IEEE Trans. on VLSI Syst., Volume 8, Issue 3, pp. 264 – 272.

18

Robust Design and Test of Analog/Mixed-Signal Circuits in Deeply Scaled CMOS Technologies

Guo Yu and Peng Li
Texas A&M University
College Station, TX, USA

1. Introduction

The proliferation of communication and consumer electronic systems leads to the large demands of high-performance & robust analog/mixed-signal circuits. On the other hand, although deeply scaled CMOS technologies enable greater degrees of semiconductor integration and lower manufacturing cost, the advancements of technologies also introduce several new challenges for VLSI circuit design. The increasing parametric variations and their impacts on circuit performances are becoming key issues which make already complex circuits even more sophisticated in design practice (Nassif, 2001). Give these barriers, designing robust mixed-signal circuits in deeply scaled CMOS technologies become a real challenge for circuit designers.

In this chapter, we propose to solve the problems by first introducing efficient and accurate modeling techniques for large analog/mixed-signal circuit designs with consideration of process variations. Powerful statistical dimension reduction techniques are utilized to make performance modeling of large circuits possible. Then these novel circuit models are used to achieve efficient parametric system performance analysis. In such ways the designers can have the full knowledge of real circuit performances under process variations, which makes it feasible to do robust system topology design and circuit optimization for large mixed-signal systems. The efficient modeling framework is also extended for circuit test purposes, which leads to robust Built-in Self-Test (BIST) circuit design and optimization.

We demonstrate the effectiveness of the proposed ideas with two popular types of mixed-signal circuit examples, Sigma-Delta A/D converters and Phase Looked Loops (PLLs).

For Sigma-Delta ADCs, we present a novel parameterized lookup table (LUT) technique for capturing performances of building blocks in the systems, and use these LUTs to perform topology trade-off analysis and system optimization. Modeling of circuit level nonlinearities, adaptive LUT generation and robust system design are explained in detail with comprehensive experimental results (Yu & Li, 2006 & 2007a).

For PLLs, we discuss building parametric Verilog-A models for charge-pump PLLs and use these models for high-level performance analysis. In order to handle large number of parametric variables, dimension reduction technique is applied to reduce simulation complexities. We apply the obtained system simulation framework to evaluate the efficiencies of parametric failure detecting of different BIST circuits and perform optimization based on the experimental results (Yu & Li, 2007b).

2. Robust Sigma-Delta ADCs Design

Sigma-Delta ADCs have been widely used in data conversion applications due to the good resolution. However, oversampling and complex circuit behaviors render the transistor-level analysis of these designs prohibitively time consuming (Norsworthy & et al., 1997). The inefficiency of the standard simulation approach also rules out the possibility of analyzing the impacts of a multitude of environmental and process variations critical in modern VLSI technologies. We propose a lookup table (LUT) based modelling technique to facilitate much more efficient performance analysis of Sigma-Delta ADCs. Various transistor-level circuit nonidealities are systematically characterized at the building block level and the whole system is simulated much more efficiently using these building block models. Our approach can provide up to four orders of magnitude runtime speedup over SPICE-like simulators, hence significantly shortening the CPU time required for evaluating system performances such as SNDR (signal-to-noise-and-distortion-ratio). The proposed modeling technique is further extended to enable scalable performance variation analysis of complex Sigma-Delta ADC designs. Such approach allows us to perform trade-off analysis of various topologies considering not only nominal performances but also their variabilities.

2.1 Background of Sigma-Delta ADC design

We briefly discuss the background and difficulties for Sigma-Delta ADC design in this section. Various important circuit nonidealities, which are difficult to model accurately using analytical equations, are also discussed. The two basic components of Sigma-Delta ADCs are modulators and digital filters as shown in Fig. 1. The analog input of the ADC is sampled by a very high frequency clock in the modulator, then the sampled signal is passed through the loop filter to perform noise-shaping. The output of the loop-filter is quantized by an internal A/D converter, producing a bit-stream at the same speed as the sampling clock. The low-pass digital filter in the decimator then removes the out-of-band noise, and the down-sampler converts the high speed bit-stream to high resolution digital codes.

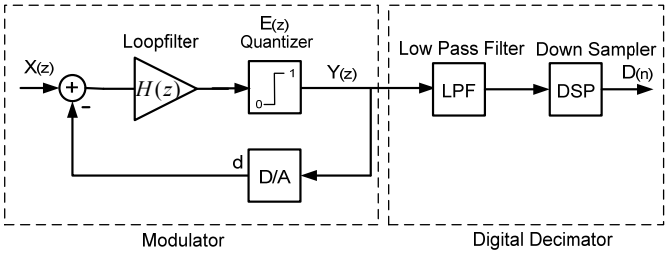


Fig. 1. Block diagram of a Sigma-Delta ADC

The difference between the ideal digital output of the quantizer and the actual analog signal is called quantization noise. The goal of Sigma-Delta technique is to eliminate this unwanted quantization noise as much as possible. By oversampling the input signal, the modulator moves the majority of the quantization noise out of the signal bandwidth. The principle of noise-shaping can be analyzed using transfer functions, which can be obtained using a linear model in the frequency domain. The quantization noise E(z) is modelled as additive noise to the quantizer, the output of the quantizer Y(z) can be written as

$$Y(z) = \frac{H(z)}{1 + d \cdot H(z)} \cdot X(z) + \frac{1}{1 + d \cdot H(z)} \cdot E(z) \tag{1}$$

where d is the feedback gain of the DAC, X(z) is the input signal, H(z) is the transfer function of the loop filter. By configuring H(z) and d, we can have different noise shaping functions so that the signal-to-noise ratio in the output can be optimized.

Sigma-Delta ADC performances are greatly impacted by various circuit-level nonidealities, such as the finite DC gain, slew of the operational amplifies, charge injections of the switches, mismatch of the internal quantizers and D/A converters. The effects are difficult to analyze accurately by hand calculation or simple analytical models, neither are their impacts on system performances. There exist several high-level simulators for Sigma-Delta ADC design, like MIDAS (Babii et al., 1997) and SWITCAP (Fang & Suyama, 1992). These techniques are only suitable for architecture-level exploration and determination of building block specifications in the early design phase, but no suitable for the consideration of circuit-level non-idealities. Often the time, transistor-level simulation is the only choice for current performance evaluation, which may take a few weeks for a single transient simulation. Analyzing the impact of process variations is an even greater challenge because a large number of long transient simulations are needed to derive the performance statistics under process variations. In the following sections, we address these issues by adopting the lookup table based models, which can handle the circuit-level details and process variation induced performance statistics accurately and efficiently.

2.2 Performance modeling with Lookup table (LUT) technique

The need of efficient simulation techniques capable of including transistor-level details is particularly pressing for assessing the impact of process variations, where the analysis complexity tends to explode in a large parameter space. We start with modelling of switched-capacitor type Sigma-Delta ADCs, which are clocked by some global sampling signals. The major components in the converters are integrators, quantizers and feedback DACs.

Since the switched-capacitor integrators in the discrete-time Sigma-Delta modulators are clocked by the sampling clock as shown in Fig. 2, it is possible to use lookup tables to represent the nonlinear state transfer function of the system (Bishop et al., 1990, Brauns et al., 1990). The output of integrator can be expressed as a function of input signals and their previous states as in (2)

$$y[k+1] = F(y[k], x[k+1], d[k+1]) \tag{2}$$

where y[k+1] is the current output of the integrator, y[k] is the integrator output in the previous clock cycle, x[k+1] and d[k+1] are the current input signal and the digital feedback

output of the DAC, respectively. This property of Sigma-Delta ADCs makes it possible to predict the new integrator output using the previous state and the new input. The previous state of the integrator, the digital feedback and the new analog input are discretized at a set of discrete voltage levels that are used as the indices to the lookup table models.

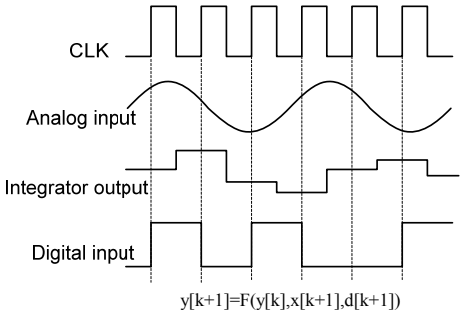


Fig. 2. Integrator behaviours under clocking

As illustrated in equation (2) and Fig. 2, the output of an integrator is a function of the input signals and the initial state of the integrator, which are discretized to generate the lookup table entries. The number of discretization levels depends on the accuracy requirement of the simulation. The internal circuit node voltage swings can be estimated by the system architecture. For low-voltage Sigma-Delta ADC designs, the internal voltages can change from 0 to supply voltage V_{dd}. To cover the whole range of voltage swing, we discretize the inputs and outputs of the integrators uniformly at N levels from 0 to V_{dd}, where N is in the range of 10. The extraction setup for an integrator with a multi-bit DAC implemented in thermometer code is shown in Fig.3. A large inductor L together with a voltage source Vs is used to set the initial value of the integrator output. The input of the integrator is also set by a voltage source Vi. The digital output of the quantizer controls the amount of charge to be fed back. An m-bit DAC implemented in thermometer code has $2^m - 1$ threshold voltages. The digital codes from 0 to $2^m - 1$ can be represented by counting the number of voltage sources that are connected to the integrator inputs from a set of voltages sources V_{d1}, V_{d2}, …, $V_{d(2^m-1)}$,the voltages of which are set to be either digital "1" or digital "0".

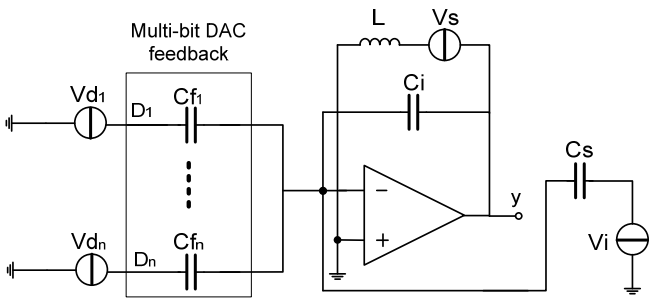


Fig. 3. LUT generation setup for integrators

The nonlinearities of quantizers can be captured using lookup tables as well. The quantizer acts as a comparator, the input threshold voltage varies depending on the direction in which the input voltage changes. To capture the hysteresis effect accurately, we use the transistor-level simulation to find the input threshold voltages at which the digital output switches from 0 to 1 (V_{off+}) and from 1 to 0 (V_{off-}), respectively. The quantizer is then modeled as

$$d[k+1] = \begin{cases} 1 & (V_{in}[k+1] > V_{off+}) \\ d[k] & (V_{off-} < V_{in}[k+1] < V_{off+}) \\ 0 & (V_{in}[k+1] < V_{off-}) \end{cases} \quad (3)$$

where d[k+1] is the new output of the quantizer, d[k] is the output in the previous clock cycle. Multi-bit quantizers can be modeled in a similar way since they are built from several 1-bit quantizers.

Sigma-Delta ADCs with continous-time modulators can also be modeled using the proposed technique with minor modificiation. Continuous-time Sigma-Delta ADCs are different from discrete-time counterparts since the integrators are not clocked by the sampling clock, and the input and the output of integrator changes throughout a clock period. In order to make the lookup-based modeling possible, we discretize each clock cycle into M time intervals with a step size dT=T/M. If dT is small enough, then in each small time period the behaviours of continous-time modulators can be approximated using the presented technique, detailed implementation for continous-time Sigma-Delta ADCs can be found in (Yu & Li, 2007a).

2.3 Parametric LUT based modeling

Process variations in the fabrication stage can cause significant performance shift for analog and mixed-signal circuits. So handling of process variations in the early design stage is critical for robust analog/mixed-signal design. In order capture the influence of process variations, we extract parameterized LUT models and perform fast statistical simulation to evaluate the performance distributions of complex Sigma-Delta ADCs. Using this efficient modeling technique, we have the ability to find the most suitable system topologies and design parameters for ADC designs.

Since the number of process variables is large, it is not possible to exhaust all the possible performances under process variations. Here we use parameterized LUT-based models to capture the impacts of circuit parametric variations that include both environmental and process variations. In this case, a nonlinear regression model (macromodel) is extracted for each table entry. In general, a macromodel correlating the input variables and their responses can be stated as follows:
given n sets observed responses $\{y_1,y_2,\ldots,y_n\}$ and n sets of m input variables $[x_1,x_2,\ldots,x_m]$, we can determine a function to relate input x and response y as (Low & Director, 1989)

$$\begin{aligned} y_1 &= h(x_{11}, x_{12}, \cdots, x_{1m}) \\ y_2 &= h(x_{21}, x_{22}, \cdots, x_{2m}) \\ &\vdots \\ y_n &= h(x_{n1}, x_{n2}, \cdots, x_{nm}) \end{aligned} \tag{4}$$

where

y_i	i-th response
h	function relating y and x
x_i	i-th set of process variables
m	number of process variables
n	number of experimental runs

The task of constructing each macromodel is achieved by applying the response surface modeling (RSM) technique where empirical polynomial regression models relating the inputs and their outputs are extracted by performing nonlinear least square fitting over a chosen set of input and output data (Box et al., 2005). To systematically control the model accuracy and cost, design of experiment (DOE) technique is applied to properly choose a smallest set of data points while satisfying a given modeling regulation. For our circuit modeling task, the input parameters are the parametric circuit variations and the output is an entry in the lookup tables. Then, a nonlinear function such as a quadratic function relating each entry in the tables with the circuit parametric variations can be determined via regression

$$\hat{y} = \hat{\beta}_0 + \sum_{i=1}^{m} \hat{\beta}_i x_i + \sum_{i=1}^{m} \sum_{j=1}^{m} \hat{\beta}_{ij} x_i x_j \tag{5}$$

where

x_i	i-th process variable set
Y	approximated response
β	estimated model fitting coefficient
m	number of process variables

Equation (5) can be rewritten in a more compact matrix form as

$$X \cdot \beta = Y \tag{6}$$

The fitting coefficient vector β can be calculated using the least-square fitting of experimental data as

$$\beta = (X^T X)^{-1} X^T Y \tag{7}$$

A major problem in solving equations (5 – 7) is the number of experimental data, so a second-order central composite plan consisting of a cube design sub-plan and a star design sub-plan is employed (Box & et al., 2005). The cube design plan is a two-level fractional factorial plan that can be used to estimate first-order effects (e.g., x_i) and interaction effects (e.g., x_i_x_j), but it is not possible to estimate pure quadratic terms (e.g., x_i^2). The star design plan is used as a supplementary training set to provide pure quadratic terms in equation (5).

In our implementation, the cube design plan is selected in order to estimate all the first-order and cross-factor second-order coefficient of the input variables.

The ranges of all parametric variations are usually obtained from the process characterization. This information is used to setup the model extraction procedure. In the cube design plan, each factor takes on two values -1 and +1 to represent the minimum and the maximum values of the parametric variation. Each factor in the star plan takes on three levels -a, 0, a, where 0 represents the nominal condition and the level range $|a| < 1$. As illustrated in Fig. 4, for each point (i,j) in the lookup table, n simulations runs are conducted using fractional factorial plan to provide the required data to generate the regression model in equation (5). As long as the lookup tables for specified process variation distributions are generated, we can perform fast system-level simulation to evaluate the performances under process variations, and in turn to achieve optimization as to be discussed in the following section.

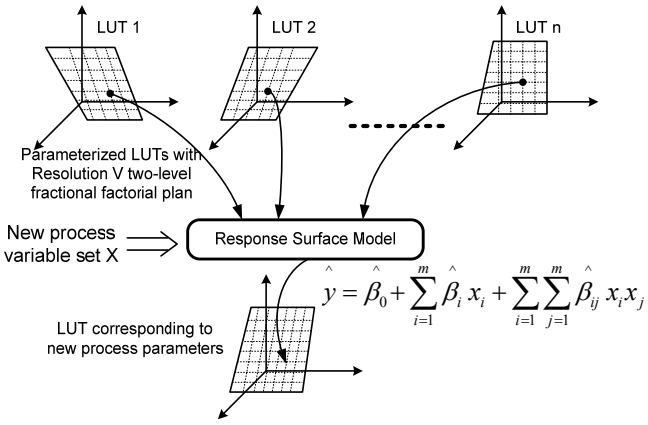


Fig. 4. Response surface modelling of parameterized LUTs

2.4 System optimization using parametric LUTs

The application of the proposed modeling techniques are demonstrated with three discrete-time Sigma-Delta ADC designs with different topologies including 2nd-order with 1-bit quantizer (SDM 1), 2nd-order with 2-bit quantizer (SDM 2) and 3rd-order with 1-bit quantizer (SDM 3). All these ADCs are implemented in 130nm CMOS technology with a single 1.5V power supply. The sampling clock and oversampling ratio are chosen to be 1MHz and 128, respectively.

Using our parameterized LUT-based infrastructure, we are able to not only predict the nominal case design performances but also their sensitivities to parametric variations. Hence, our technique provides an efficient way for statistical circuit simulation as well as performance-robustness trade-off analysis. For statistical analysis, a Resolution V $2^{(8-2)}$ fractional factorial design plan that includes 64 runs for the cube design plan and 17 runs for the star design plan is employed by SDM 1. For SDM 2 and SDM 3, a Resolution VI $2^{(6-1)}$

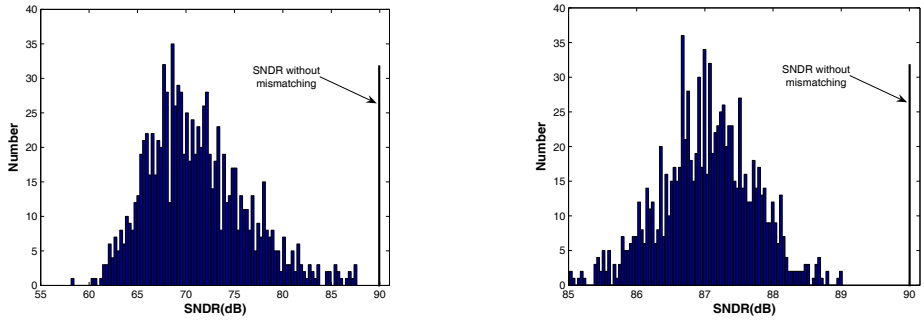
fractional factorial design plan with 45 runs is employed, resulting in 32 runs for the cube design plan and 13 runs for the star design plan.

In Table 1, the proposed LUT-based simulator is compared with the transistor-level simulator (Spectre) in terms of model extraction time, simulation time, and predicted nominal SNDR and THD values. Once the LUT models are extracted, the LUT-based simulator can be efficiently employed to perform statistical performance analysis, which is infeasible for the transistor-level simulator. For the 2nd-order Sigma-Delta ADC with 1-bit quantizer, it only takes 20 minutes to conduct 1,000 LUT-based transient simulations each including 64k clock cycles. For the same analysis, transistor-level simulation with conventional simulators is expected to take 4,500 hours to complete. In terms of accuracy, the SNDRs and THDs predicted by Spectre and the LUT simulator are also presented in Table 1. The error of SNDR of our LUT-based simulator is within 1dB, which demonstrates the accuracy of the proposed technique.

	LUT-based simulation					Spectre simulation		
Design	Sin. gen.	Par. gen.	Runtime	SNDR	THD	Runtime	SNDR	THD
SDM 1	7 min	9.5 hr	2 s	73.8dB	-63.1dB	4.5 hr	74.1dB	-62.6dB
SDM 2	20 min	15 hr	4 s	90.2dB	-77.4dB	9.5 hr	90.0dB	-76.8dB
SDM 3	8 min	11 hr	2 s	83.3dB	-67.5dB	5.5 hr	83.5dB	-66.9dB

Table 1. Runtime and accuracy of the proposed LUT simulation (© [2007] IEEE, from Yu & Li, 2007a)

With the powerful LUT-based simulator, we can perform system evaluation very efficiently so the optimization of system topologies and detailed design become possible. First we use the optimization of 2nd-order Sigma-Delta ADC with multi-bit quantizer as an example by investigating the impacts of DAC capacitance mismatch. The capacitor mismatch level decreases as the capacitance increases, so it is of interest to investigate the trade-offs of system noise performances and area (Pelgrom & et al., 1989). Statistical simulations are performed to analyze the influence of the mismatch of the two internal DACs by sweeping the values of the three charging capacitors in each DAC. The variation of capacitances is modeled using a Gaussian distribution with $3\sigma = 1\%$. The distributions of SNDR due to the capacitance mismatch in the two DACs are shown in Fig. 5, respectively.

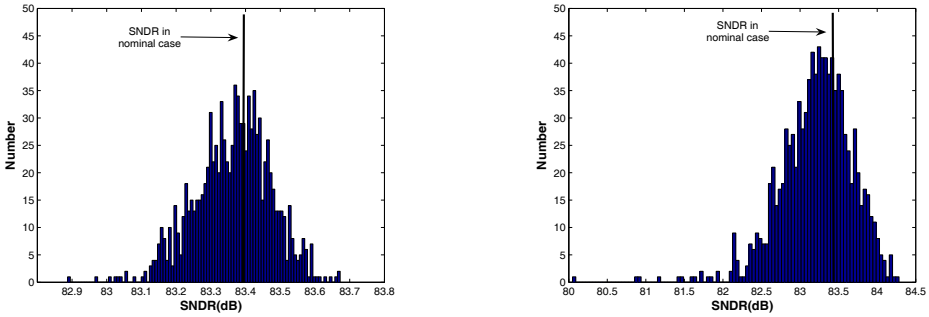


(a) DAC connected to first-stage (b) DAC connected to second-stage

Fig. 5. SDNR distributions for DACs connected to different stages in SDM 2 (© [2007] IEEE, from Yu & Li, 2007a)

We can see from the two figures that the mismatch of the DAC connected to the first stage integrator (left figure) has much more influence to the system performance than that of the other DAC (right figure). This can be explained by the fact that the first stage DAC is connected directly to the system input, so the feedback error because of the DAC mismatch will be magnified by the second stage integrator. The result of this analysis indicates that more attention should be paid to the first stage DAC in the design process.

Another optimization example is to evaluate the charging capacitor mismatches in SDM 3. The mismatch of each capacitor is modelled using Gaussian distribution. We evaluate the system performance distributions with variation of capacitances set to $\sigma = 1\%$ and $\sigma = 5\%$, as illustrated in Fig. 6.



(a) $\sigma = 1\%$ (b) $\sigma = 5\%$

Fig. 6. SNDR distributions for mismatches of charging and sampling capacitors in SDM 3 (© [2007] IEEE, from Yu & Li, 2007a)

We can observe from Fig. 6 that performance distribution deviations increase from 1dB to 4.5dB for capacitor variation $\sigma = 5\%$, and the impact of the mismatch of charging and sampling capacitors is not as critical as that of the multi-bit DAC even with $\sigma = 5\%$. It is also possible to perform more complete system analysis and optimization using the proposed parametric LUT-based simulation method depending on the target of the design (Yu & Li, 2007a).

3. Robust PLL Design

As an essential building block, PLLs are widely used in today's communication and digital systems for purposes such as frequency synthesis, low-jitter clock generation, data recovery and so on. Although the input and output signals of PLLs are in the digital domain, most PLLs implementations consist of both digital and analog components, which make them prone to process variation influences. In this section we propose an efficient parameter-reduction modelling technique to capture process variations and further achieve low-cost system performance evaluation using hierarchical system simulation. The proposed method not only can be used for robust PLL design under process variation, but also paves the road for effective built-in self-test circuit design as to be discussed in the next section.

3.1 Background of PLL design

As illustrated in Fig. 7, a typical charge-pump PLL system consists of a frequency detector, a charge pump, a loop filter, a voltage-controlled oscillator (VCO) and a frequency divider. The frequency of the output clock signal Fout is N times of that of reference clock signal Fref, where N can be an integer number or fractional number. The PLL design options include VCO topologies and component sizes, filter characterizations, charge current in the charge pump and so on. The metrics of PLL systems usually include acquisition/lock-in time, output jitter, system power, total area, etc. The goal of PLL design and optimization is to find the best overall system performances by searching in the design variable spaces.

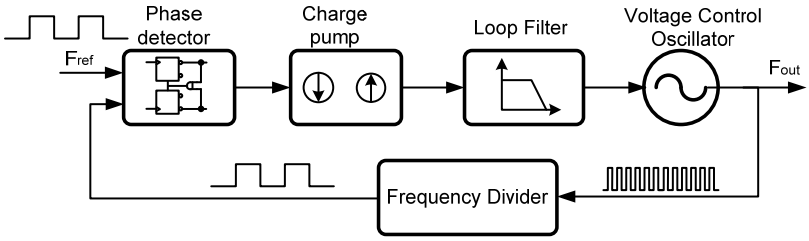


Fig. 7. Block diagram of charge-pump PLL

Due to the mixed-signal nature, the design and optimization of PLL system is quite complex and costly. For example, a long transient simulation (in the order of hours or days) is needed to obtain the lock-in time behavior of PLL, which is one of the most important performance metrics for a PLL. So the brute-force optimization by searching in the design space with transistor-level simulation is infeasible for PLL systems.

The difficulties of system performance analysis can be addressed by adopting a bottom-up modelling and simulation strategy. The performances of analog building blocks can be evaluated and optimized without too much cost. When the behaviours of analog building blocks are extracted, these building blocks can be mapped to Verilog-A models for fast system level evaluation (Zou & et al., 2006). By using this approach we can avoid the scalability issue associated with time consuming transistor-level simulations.

When process variations are considered, the situation becomes more sophisticated. The large number of process variables and the correlations between different building blocks introduce more uncertainties for PLL performance under process variations. In order to utilize the hierarchical simulation method while taking into consideration of statistical performance distributions, we propose an efficient macromodeling method to handle this difficulty. The key aspect of our macromodeling techniques is the extraction of parameterized behavioral models that can truthfully map the device-level variabilities to variabilities at the system level, so that the influence of fabrication stage variations can be propagated to the PLL system performances.

Parameterization can be done for each building block model as follows. First, multiple behavioural model extractions are conducted at multiple parameter corners, possibly following a particular design-of-experiments (DOE) plan (Box & et al., 2005). Then, a

parameterized behavioral model is constructed by performing nonlinear regression over the models extracted at different corners. This detailed parametric modeling step is advantageous since it systematically maps the device-level parametric variations to each of the behavioral models. However, difficulties arise when the number of parametric variations is large, which leads to a prohibitively high parametric model extraction cost. We address this challenge by applying design-specific parameter dimension reduction techniques as described in the following section.

3.2 Hierarchical modeling for PLLs

In this section we first describe the nominal behavioral model extraction for each PLL building block, then we discuss how a parameterized model can be constructed in the next section.

The voltage controlled oscillator is the core component of a PLL. The two mainstream types of VCOs are LC-tank oscillators and ring oscillators. In a typical VCO model, the dynamic (response to input change) and static (V-Freq relation) characteristics of the voltage to frequency transfer are modeled separately first and then combined to form the complete model. The static VCO characteristic can be written as Fout=f(V'con), where Fout is the output signal frequency, V'con is the delayed control voltage, and f(.) is a nonlinear mapping relating the voltage with the frequency. To generate the analytical model, the mapping function f(.) can be further represented by an n-th order polynomial function.

$$F_{out} = a_0 + a_1 V_{con}^{'} + a_2 V_{con}^{'\,2} + \cdots + a_n V_{con}^{'\,n} \tag{8}$$

where a_0, a_1, …, a_n are coefficients of the polynomial. To generate the above polynomial, multiple VCO steady-state simulations are conducted at different control voltage levels and a nonlinear regression is performed using the collected simulation data.

Suppose the control voltage is Vcon, the dynamic behavior of the VCO is modeled by adding a delay element that produces a delayed version of the control voltage (V'con). The delay element can be expressed using a linear transfer function H(s) (e.g. a second-order RC network consisting of two R's and two C's). H(s) can be determined via transistor-level simulation as follows: a step control voltage is applied to the VCO and the time it takes for the VCO to reach the steady-state output frequency, or the step-input delay of the VCO, is recorded. H(s) is then synthesized that gives the same step-input delay. The dynamic effect is usually notable in LC VCOs due to the high-Q LC tank while in ring oscillators this effect may be neglected.

The charge pump is mainly built with switching current sources. As illustrated in Fig. 8, the control signals of the two switches M_1 and M_2 come from the outputs of the phase and frequency detector. The currents through M_1 and M_2 can be turned on-and-off to provide desired charge-up or charge-down currents. The existing charge pump macromodels are very simplistic. Usually, both the charge-up and charge-down currents are modeled as constant values. A constant mismatch between the two currents may also be considered (Zou & et al., 2006). However, this simple approach is not sufficient to model the behavior of charge pump accurately. In real implementation, the current sources are implemented using transistors so that the actual output currents will vary according to the voltages across these

MOSFETs. Therefore, the dependency of charge-up and charge-down currents on Vcon must be considered.

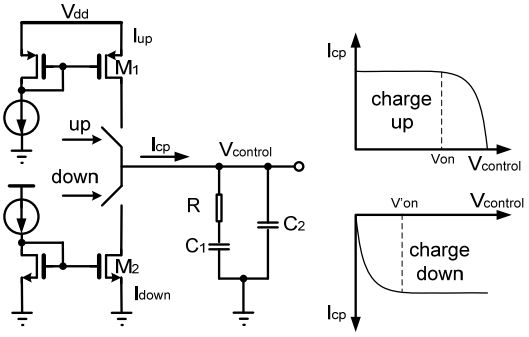


Fig. 8. Modeling of charge pump (© [2007] IEEE, from Yu & Li, 2007b)

In our charge pump model, for each output current, the current vs. Vcon characteristics is divided into two regions. When the output voltage Vcon is close to the supply voltage, then switch M_1 will be biased in triode region. The charge-up current Iup in the triode region can be written as

$$I_{up} = \mu_p C_{ox} \frac{W}{L}[(V_{gs} - V_{thp})V_{ds} - 0.5V_{ds}^{\ 2}] \tag{9}$$
$$V_{ds} = V_{dd} - V_{on} - V_{con}$$

where Vdd is the supply voltage, Von is the on-voltage across the switch, Vgs is the gate-source voltage, μ_p is the mobility, Cox is the oxide capacitance, W is width and L is the length of M_1. We can see from Equation (9) that the charge-up current is dependent on the output voltage Vcon. We use a polynomial to explicitly model such voltage dependency

$$I_{up} = b_0 + b_1 V_{con} + b_2 V_{con}^2 + b_3 V_{con}^3 \tag{10}$$

where bi are the polynomial coefficients. Similarly, the charge-down current has a strong Vcon dependency when Vcon is low. This voltage dependency is modeled in a similar fashion. When M_1 and M_2 operate in saturation region, they act as part of the current mirrors. In this case, constant output current values are assumed while the possible mismatches between the two are considered in our Verilog-A models.

The phase detector and the frequency divider are digital circuits so that they are more amenable to behavioral modeling. The two key parameters of the phase detector and the frequency divider are the output signal delay and the transition time, which are easy to extract from transistor-level simulation. The loop filters are usually comprised of passive RC elements, which can be directly modeled in Verilog-A simulation.

3.3 Efficient parameter-reduction modeling for PLLs

The key parametric variations for transistors may include variations in mobility, gate oxide, threshold voltage, effective length and so on (Nassif, 2001). The consideration of all possible sources of variations in transistors and interconnects can easily lead to explosion of the

parameter space, rendering the parametric modeling infeasible. Although the widely used principle component analysis (Reinsel & Velu, 1998) can be adopted to perform parameter dimension reduction, its effectiveness may be rather limited since the parameter reduction is achieved by only considering the statistics of controlling parameters while neglecting the important correspondence between these parameters and the circuit performances of interest. As such, the extent to which the parameter reduction can be achieved is not sufficient for our analog macromodeling problems. To address this difficulty, a more powerful design-specific dimension reduction technique, which is based on reduced rank regression (RRR), is developed. This new technique considers the crucial structural information imposed by the design and has been shown to be quite effective for parametric interconnecting modeling problems (Feng & et al., 2006).

Suppose we have a set of n process variations, X, and a set of N performances, Y. The objective is to identify a smaller set of new variables Z, based on X, which are statistically significant to the performances of interest, Y. Without loss of generality, let us assume Y nonlinearly depends on X through a quadratic model

$$Y = f(X) \approx \begin{bmatrix} C_1 & C_2 \end{bmatrix} \begin{bmatrix} X \\ X \otimes X \end{bmatrix} \tag{11}$$

where C_1 and C_2 are the first and second order coefficients, $X \otimes X$ represents the quadratic terms of X. The combination of the linear and quadratic terms of X are then defined as a new predictor vector $\widetilde{X} = \left[X^T \left(X \otimes X \right)^T \right]^T$. Now the quadratic model in equation (11) can be cast into a linear model as $Y = A\widetilde{X} + \varepsilon$. To identify the redundancy in X to facilitate parameter reduction, we seek a reduced rank regression model in the form

$$Y = A_R B_R \widetilde{X} + \varepsilon \tag{12}$$

where A_R and B_R have a reduced rank of R (R < n), and B_R has only R columns. We denote the covariance matrix of $\widetilde{X}$ as $Cov(X) = \Sigma_{\widetilde{X}\widetilde{X}}$, and covariance matrix between $\widetilde{X}$ and Y as $Cov(Y, \widetilde{X}) = \Sigma_{Y\widetilde{X}}$. It can be shown that an optimal reduced rank model (in the sense of mean square error) is given as (Reinsel & Velu, 1998)

$$A_R = U, B_R = U^T \Sigma_{Y\widetilde{X}} \Sigma_{\widetilde{X}\widetilde{X}}^{-1} \tag{13}$$

where U contains R normalized eigenvectors corresponding to the R largest eigenvalues of the matrix: $D = \Sigma_{Y\widetilde{X}} \Sigma_{\widetilde{X}\widetilde{X}}^{-1} \Sigma_{\widetilde{X}Y}$. It is important to note that a successful construction of the above reduced rank model indicates that only a smaller set of R new parameters $Z = B_R \widetilde{X}$ are critical to Y in a statistical sense, hence facilitating the desired parameter reduction.

It should be noted that the reduced rank regression is only employed as a means for parameter reduction so as to reduce the complexity of the subsequent parameterized macromodeling step. Hence, Y in the above equations does not have to be the true performances of interest and can be just some circuit responses that are highly correlated to the performances. This flexibility can be exploited to more efficiently collect $\Sigma_{Y\widetilde{X}}$ though Monte-Carlo sampling if Y are easier to obtain than the true performances in simulation.

The complete parameterized PLL macromodel extraction flow is shown in Fig. 9. Every building block is modeled using Verilog-A for efficient system-level simulation. Each model parameter for building blocks is expressed as a polynomial in the underlying device-level variations, such as

$$P = f(V_{th1}, L_{eff1}, T_{ox1}, \ldots, V_{thn}, L_{effn}, T_{oxn}) \tag{14}$$

where V_{thi}, L_{effi}, T_{oxi}, etc represent the parameters of i-th transistor, f(.) is the nonlinear polynomial function to connect process variations to system performances. f(.) is very difficult to obtain if the number of parameters is large. Hence, RRR-based parameter reduction is applied, which leads to a set of R new parameters Z that are the most important variations for the given circuit performances of interest. If R is small, then a new parameterized model in terms of Z can be easily obtained through conventional nonlinear regression for coefficients of equation (11)

$$C = f(Z_1, Z_2, \cdots Z_R) \tag{15}$$

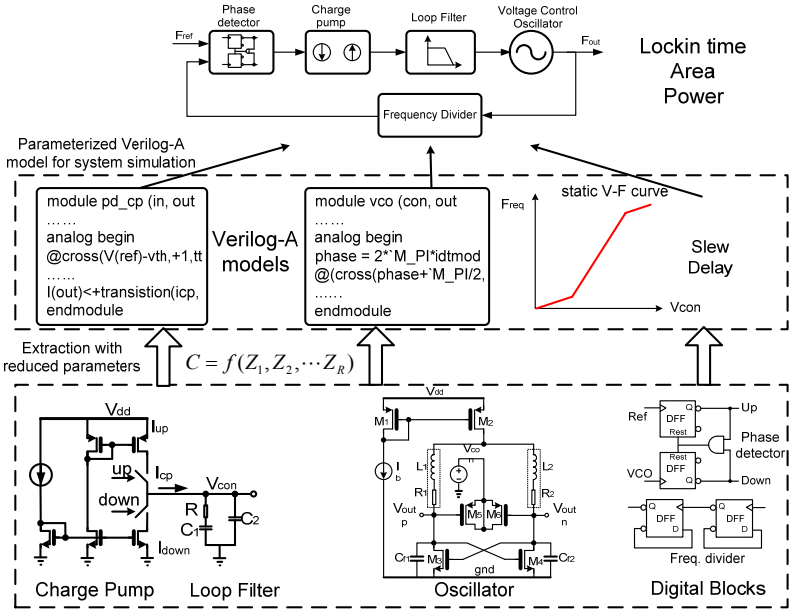


Fig. 9. Flow of PLL macromodeling

In addition to reducing the cost of parameterized macromodeling, parameter dimension reduction will also lead to more efficient statistical simulation of the complete PLL. This is because instead of analyzing the design performance variations over the original high-dimensional parameter space, statistical simulation can now be performed more efficiently in a much lower dimensional space that carries the essential information of the design variability, which is a very good property to achieve PLL system optimization. Detailed optimization method for PLL using the hierarchical macromodels can be found in (Yu & Li, 2008).

With the hierarchical models and parameter reduction technique, we can also perform built-in self-test circuit design and optimization since lengthy transient simulations can be relieved by the proposed method. We will discuss this part in the next section.

4. Built-in Self-test Scheme Design for PLLs

Testing of PLLs is very challenging and of great interest since: a) Usually only simple functional tests such as phase lock test is feasible in production test. However, it may not be sufficient for guaranteeing all the specifications; b) The operation of PLLs is intrinsically complex, e.g. simple frequency domain analysis is not applicable for PLL circuits due to the digitalized input/output signals and the closed-loop dynamics; c) Internal analog signals are difficult to access outside of the chip and system specifications such as jitter, frequency range and lock-time are very complex and costly to measure with digital testers. In this section, we discuss the implementations and optimization of built-in self-test circuits to capture PLL performance failures using the efficient modelling and simulation framework in Section 3.

4.1 BIST circuit design

Built-in self-test (BIST) has emerged as a very promising test methodology for integrated circuits although its application to mixed-signal ICs is more challenging compared to the digital counterparts. Sunter and Roy propose a BIST scheme to measure key analog specifications including jitter, open loop gain, lock range and lock time in PLL systems (Sunter & Roy, 1999), while most of other proposed BIST schemes mainly focus on catastrophic faults. Kim and Soma present an all-digital BIST scheme using charge pump as stimulus to charge VCO up and down to detect catastrophic faults (Kim & Soma, 2001). In (Hsu & et al., 2005) Hsu et *al* propose a different BIST scheme for catastrophic fault detection in PLLs by introducing the phase errors to the inputs to the phase frequency detector. Other BIST schemes proposed like (Azais & et al., 2003) are also targeted at catastrophic faults.

While detection of catastrophic (hard) faults remains as a key consideration in test, parametric faults caused by the growing process variations in modern nanometer VLSI technologies are receiving significant concerns. In most cases, chips with parametric faults may be still functional but cannot achieve the desired performance specifications, and hence should be screened out. These failing chips are free of catastrophic faults so that specific BIST schemes targeting at parametric failures must be developed.

Given that process variations and resultant parametric failures will continue to rise in sub-100-nm technologies, a design-phase PLL BIST development methodology is strongly desired. Such methodology should facilitate systematic evaluation of parametric variations of complex PLL system specifications and their relations to specific BIST measurements so as to enable optimal BIST scheme development. To this end, however, three major challenges must be addressed: a) Suitable modeling techniques must be developed in order to enable feasible whole system PLL analysis while considering realistic device-level process variations and mismatch; b) Device-level parametric variations and mismatch that contribute to parametric failures form a high-dimensional parameter space and the resulting

issue of *curse of dimensionality* brings significant modelling and analysis difficulty; and c) Effective optimization strategies are desired in order to develop optimal BIST schemes.

With the techniques presented in section 3, the first two difficulties have been addressed, and now we put more efforts on the optimization of BIST circuits. The most widely used approach in BIST design is to utilize the existing digital blocks, such as to use the frequency divider as counter and read out its state in order to detect chip failures (Sunter & Roy, 1999), (Kim & Soma, 2001), (Hsu & et al., 2005), (Azais & et al., 2003). It is expected that the frequency divider/counter output will change significantly if there exists a catastrophic fault. However, parametric failures may produce smaller variations in the readout values. Hence, they are more difficult to detect and deserve more careful treatments.

We consider three BIST schemes shown in Fig. 10 as potential candidates. Similar in spirit to the existing BIST schemes, the main idea of the proposed BIST schemes is to control the charge pump in a way such that the output frequency of the PLL will be altered and the state of the frequency divider is read out at certain time instance for failure detection. Device-level variations and mismatch will perturb the operation of the PLL and can push the system performances out of the specification window. The same parametric variations may be reflected in the variations in the readout values of the frequency divider. Parametric failures can be detected if the states of the frequency divider are strongly correlated with the design performances.

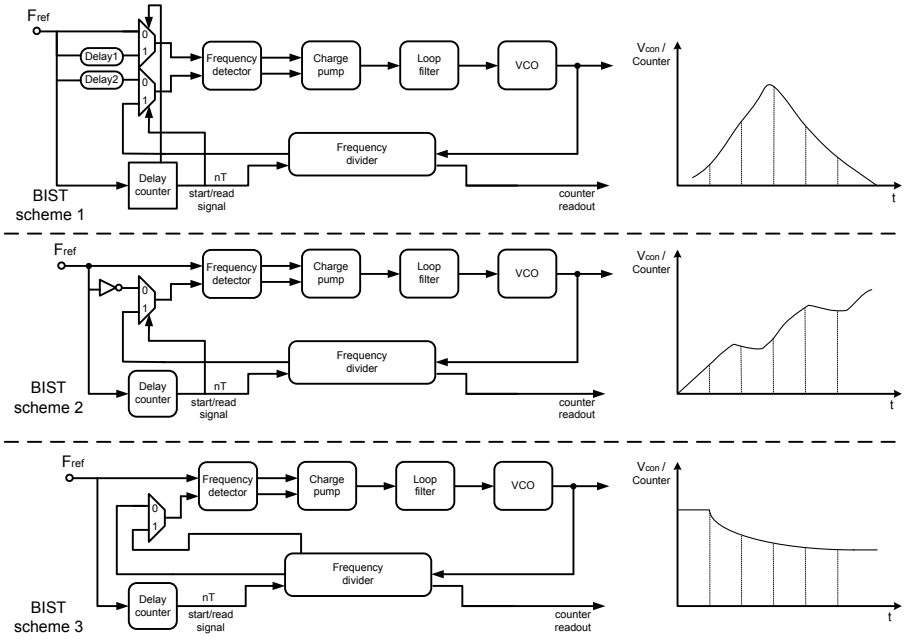


Fig. 10. Three BIST scheme candidates (© [2007] IEEE, from Yu & Li, 2007b)

Test schemes are described as follows:

a) BIST scheme 1
The first BIST scheme is similar to the one adopted in (Hsu & et al., 2005). In the normal operation mode, the reference input and the output of the frequency divider are applied to the frequency detector to form the closed loop configuration. In the test mode, the output of the frequency divider is disconnected from the input of the frequency detector. The reference input and or its delayed versions are fed through the muxes to the frequency divider forming a open loop configuration. The first delay element has a larger delay value than the second one. To charge up the VCO, the reference input and its delayed version through delay 2 are applied to the frequency detector. To charge down the VCO, the delayed versions of the reference input by both delay 1 and delay 2 are selected. The delay values of the two delay elements determine the phase error introduced at the frequency detector inputs. Hence, they also dictate the coverage of the VCO tuning range in this BIST setup. Under typical design values, delay values in the order of ten's of the reference clock signal period are required, which may cost significant silicon area to implement.

For all these three schemes, the counter read-out signals, which control the start and end points for a single test run, are generated by passing the reference clock signal F_{ref} through a series of D flip-flops. As such, the contents of the frequency divider within a defined time interval are read out.

b) BIST scheme 2
To solve the silicon overhead problem of scheme 1, we propose the second BIST scheme which employs an inverter to introduce the phase difference. Since this configuration introduces a constant phase delay of dT at the inputs of frequency divider, the charge pump experiences the following sequence of operation: charge up → stop → charge up → stop until the control voltage of the VCO reaches the fully voltage swing.

c) BIST scheme 3
The third BIST scheme is configured as follows: first the PLL is put in a standard closed-loop configuration and then a standard phase lock test is performed. Once the PLL is locked, the feedback signal frequency is changed from F_{out} to $2F_{out}$ by using the mux to select the output of the second last D flip-flop in the divider.

A brief summary of the three BIST schemes: for scheme 1, the area cost is high while the test time is short, for scheme 2, the area cost is low and the test time is also low, for scheme 3, the area cost is low, and the test time is medium. The most important aspect for BIST schemes, however, is the test accuracy. We use the macromodels developed in section 3 to perform test accuracy evaluation and optimization for these BIST scheme candidates.

4.2 Test scheme evaluation and optimization

The most straightforward way to evaluate a BIST scheme is to perform Monte-Carlo simulations and examine the ability of the specified BIST measurements to predict the pass/fail status of the design. Since a set of few hundred Monte-Carlo PLL simulations may

take tens of hours to complete, more runtime efficient approaches are needed, especially for the optimization purpose.

The optimization of a given BIST scheme with n digital outputs is illustrated in the second half of Fig. 11. Since a BIST scheme may be evaluated many times under different setups (e.g., the time interval within which the states of the frequency divider are read out) within the optimization loop, the correlation between the BIST schemes and the PLL performance must be efficiently conducted. This goal is achieved by identifying the critical sources of variations Z_v as in equation (15).

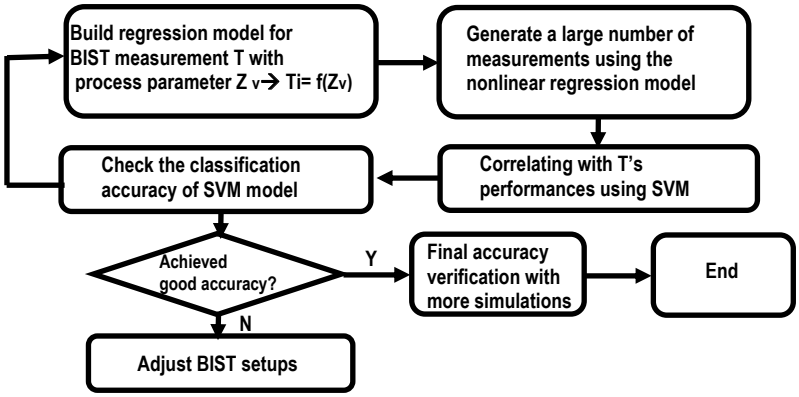


Fig. 11. BIST scheme optimization (© [2007] IEEE, from Yu & Li, 2007b)

Noticing that Z_v only contains a reduced set of variations, a nonlinear empirical model relating each measurement T_i of the given BIST scheme and Z_v can be rather efficiently generated. This is achieved by conducting a few Verilog-A based PLL simulations at different Z_v samples and performing nonlinear regression: $T_i = f(Z_v)$. Note that this step does not incur a high simulation cost since regression models are only built over a low-dimension parameter space represented by Z_v. Using these easily obtained regression models, a large set of samples for each T_i can be efficiently generated.

To capture the potential nonlinear correspondence between the design performances and the measurements, Support Vector Machine (SVM) is adopted as an accurate classifier (Vapnik, 1998). Support Vector Machine is a powerful method to build highly nonlinear multivariate regression/classification models. In SVM regression, we consider a set of training data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where x_i is the input vector and y_i is the corresponding output. The input X is mapped into a high dimensional feature space using nonlinear transformation, then a best fitting function is constructed in this feature space as

$$y = f(x) = \omega \cdot \phi(x) + b \tag{16}$$

where ϕ is the nonlinear transformation, b is the bias term, and ω represents the model parameters to be decided. Based on this nonlinear function f(.), we can classify the chip as faulty or not with the BIST circuit outputs.

We demonstrate the effectiveness of the proposed method using a PLL design implemented in 90nm CMOS technology. The frequency versus control voltage curve of the VCO is extracted using the model of equation (8). First we simulate the VCO for a few clock cycles and gather the time-domain output response as Y. Then RRR is applied to get a reduced parameter set Z to represent the important device-level parameters. A parametric model of the VCO in terms of Z is then built. To model the statistical characteristics of the VCO accurately, a 6-th order polynomial fitting is used to fit the output frequency vs. control voltage curve.

The Verilog-A models for other building blocks are extracted in a similar fashion. Specifically, the charge pump model is generated using a 3-rd order polynomial in the output voltage for each charge-up/down current. There are a total of 17 Verilog-A model parameters extracted for the complete PLL design.

We evaluate the effectiveness of three BIST schemes. SVM models are generated as in Fig. 11 to predict the pass/fail status of the chips based on the corresponding BIST outputs. 400 Monte-Carlo simulation samples are generated by conducting PLL system simulation using Verilog-A macromodels. These data are used to generate the SVM model. To evaluate the effectiveness of the each scheme more reliably, another 100 Monte-Carlo simulations are carried out and used as the test data for checking the accuracy of the SVM model. The pass/fail predictions achieved through the three SVM models are compared against the simulated chip performances, as shown in Fig. 12. Here, the predictions made through the simulation are labeled as "direct measurement", and +1 indicates a chip being classified as "fail", while -1 indicates the opposite.

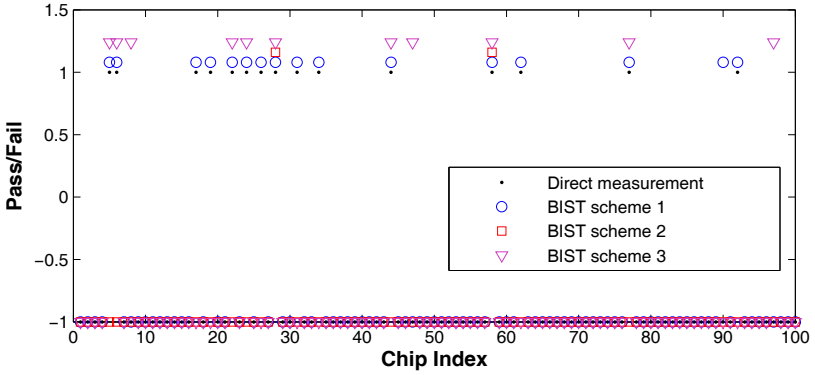


Fig. 12. Pass/fail predictions of three BIST schemes (© [2007] IEEE, from Yu & Li, 2007b)

From Fig. 12 we can see that BIST scheme 1 only has only 1 misclassification. The performance of BIST scheme 2 is verified to be poor as it can only detect two faulty chips which may have largest variations. BIST scheme 3 can detect more failures than scheme 2 but is still not as good as scheme 1.

We further look into the trade-offs between the accuracy and the number of digital outputs for optimization. This is important since fewer test codes will correspond to a shorter test time, if a similar accuracy can be achieved. This trade-off analysis is conducted for every scheme in Fig. 12. It can be observed that for a small number of digital outputs, the accuracy of scheme 2 is actually higher than that of scheme 3. It can be also seen that the accuracy of scheme 3 becomes quickly saturated as the number of outputs increases. Under all the cases, scheme 1 is always the optimal choice.

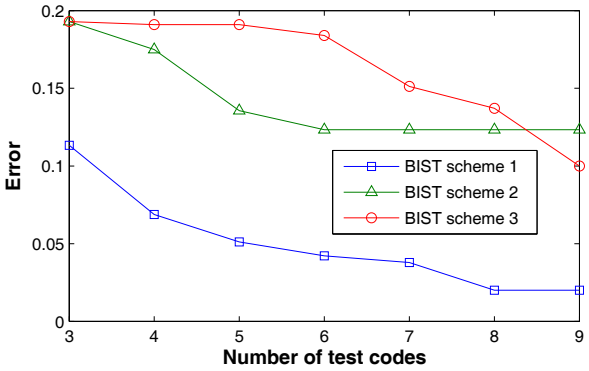


Fig. 13. Accuracy vs. test structures for three BIST schemes (© [2007] IEEE, from Yu & Li, 2007b)

There are other optimizations can be done to improve BIST schemes using the efficient macromodeling technique in Section 3 and Section 4 (Yu & Li, 2007b). It is of great benefit to do such kind of analysis and optimization by taking consideration of statistical system performances in the early design stage so that we can avoid the costly design iterations due to the influence of process and environmental variations.

5. Conclusion

In this chapter we have discussed the influences of process variations to analog/mixed-signal circuits in deeply scaled CMOS technologies. The performances of two types of popular mixed-signal systems, i.e. Sigma-Delta ADCs and Phase-locked Loops are evaluated under process variations. Parameterized lookup table technique and reduced-rank regression with hierarchical macromodeling method were proposed to fulfil the optimization of these two systems, respectively. We also extended the obtained fast system performance evaluation framework to compare the efficiencies of parametric failure detecting of different BIST circuits and perform test circuit optimization.

6. Acknowledgement

This work was funded in part by the FCRP Focus Center for Circuit & System Solutions (C2S2), under contract 2003-CT-888.

7. References

Azais, F. & et al. (2003). An all-digital DFT scheme for testing catastrophic faults in PLLs. *IEEE Design & Test of Computers*, Vol. 20, No. 1, pp. 60 - 67, Jan. 2003

Babii, S. & et al. (1997). *MIDAS User Manual,* Stanford University, Stanford, CA

Bishop, R. & et al. (1990). Table-based modeling of delta-sigma modulators. *IEEE Trans. On Circuits and Systems*, Vol. 37, No. 3, pp. 447-451, Mar. 1990

Box, G.; Hunter, D. & Hunter, W. (2005). *Statistics for Experiments: Design, Innovation, and Discovery,* John Wiley & Son, 978-0471718130, Hoboken, NJ

Brauns, G. & et al. (1990). Table-based modeling of delta-sigma modulators using ZSIM. *IEEE Trans. On Computer-aided Design*, Vol. 9, No. 2, pp. 142-150, Feb. 1990

Fang, S. & Suyama, K. (1992). *User's Manual for SWITCAP2,* Columbia University, New York, NY

Feng, Z.; Yu, G. & Li, P. (2007). Reducing the Complexity of VLSI Performance Variation Modeling Via Parameter Dimension Reduction. *Proceedings of International Symposium on Quality Electronic Design, pp. 737-742*, 978-0769527957, Mar. 2007, IEEE press, San Jose, CA

Hsu, C. ; Lai, Y. & Wang, S. (2005). Built-in self-test for phase-locked loops. *IEEE Trans. On Instrument and Measurement*, Vol. 54, No. 3, pp. 996-1002, Jun. 2005

Kim, S. & Soma, M. (2001). An all-digital built-in self-test for high-speed phase-locked loops. *IEEE Trans. On Circuits and Systems II*, Vol. 48, No. 2, pp. 141-150, Feb. 2001

Low, K. & Director, S. (1989). An efficient methodology for building macromodels of IC fabrication processes. *IEEE Trans. On Computer-aided Design*, Vol. 8, No. 12, pp. 1299-1313, Dec. 1989

Nassif, S. (2001). Modeling and Analysis of Manufacturing Variations. *Proceedings of Custom Integrated Circuits Conference, pp. 223-228*, 978-0780365917, May 2001, IEEE press, San Diego, CA

Norsworthy, S.; Schreier, R. & Temes G. (1997). *Delta-Sigma Data Converters: Theory, Design, and Simulation*. IEEE Press, 978-0780310452, Piscataway, NJ

Pelgrom, M.; Duinmaijer, A. & Welbers, A. (1989). Matching properties of MOS transistors. *IEEE Journal of Solid-State Circuits*, Vol. 24, No. 5, pp. 1433 - 1440, Oct. 1989

Reinsel, G. & Velu, R. (1998). *Multivariate Reduced-Rank Regression, Theory and Applications*. Springer, 978-0387986012, New York, NY

Sunter, S. & Roy A. (1999). BIST for phase-locked loops in digital applications. *Proceedings of International Test Conference*, pp. 532-540, 978-0780357531, Sep. 1999, IEEE Press, Atlantic City, NJ

Vapnik, V. (1998). *Statistical Learning Theory*. Wiley-Interscience, 978-0471030034, New York, NY

Yu, G. & Li, P. (2006). Lookup Table Based Simulation and Statistical Modeling of Sigma-Delta ADCs. *Proceedings of Design Automation Conference*, pp. 1035-1040, 978-1595933816, San Francisco, CA, July 2006, IEEE Press

Yu, G. & Li, P. (2007a). Efficient Lookup Table Based Modeling for Robust Design of ΣΔ ADCs. *IEEE Trans. on Circuits and Systems – I*, Vol.54, No.7, Sep. 2007, pp.1513-1528,

Yu, G. & Li, P. (2007b). A Methodology for Systematic Built-in Self-Test of Phase-locked Loops Targeting at Parametric Failures. *Proceedings of International Test Conference*, pp. 1-10, 978-1424411276, Oct. 2007, IEEE Press, Santa Clara, CA

Yu, G. & Li, P. (2008). Yield-aware hierarchical optimization of large analog integrated circuits. *Proceedings of International Conference on Computer-Aided Design*, pp. 79-84, 978-1424428199, Nov. 2008, IEEE Press, San Jose, CA

Zou, J.; Mueller, D.; Graeb, H. & Schlichtmann, U. (2006). A CPPLL hierarchical optimization methodology considering jitter, power and locking time. *Proceedings of Design Automation Conference*, pp. 19-24, 978-1595933816, San Francisco, CA, July 2006, IEEE Press

19

Nanoelectronic Design Based on a CNT Nano-Architecture

Bao Liu
Electrical and Computer Engineering Department
The University of Texas at San Antonio
San Antonio, TX, 78249-0669
Email: bliu@utsa.edu

Abstract — Carbon nanotubes (CNTs) and carbon nanotube field effect transistors (CNFETs) have demonstrated extraordinary properties and are widely expected to be the building blocks of next generation VLSI circuits. This chapter presents (1) the first purely CNT and CNFET based nano-architecture, (2) an adaptive configuration methodology for nanoelectronic design based on the CNT nano-architecture, and (3) robust differential asynchronous circuits as a promising nano-circuit paradigm.

1. Introduction

Silicon based CMOS technology scaling has driven the semiconductor industry towards cost minimization and performance improvement in the past five decades, and is rapidly approaching its end (30). On the other hand, nanotechnology has achieved significant progress in recent years, fabricating a variety of nanometer scale devices, e.g., molecular diodes (44) and carbon nanotube field effect transistors (CNFETs) (46). This provides new opportunities for VLSI circuits to achieve continuing cost minimization and performance improvement in a post-silicon-based-CMOS-technology era.
However, we must overcome a number of significant challenges for practical nanoelectronic systems, including achieving some of the most critical nanoelectronic design metrics as follow.

1. Manufacturability. As minimum layout feature size becomes smaller than lithography light wavelength, traditional lithography based manufacturing process can no longer achieve satisfiable resolution, and leads to significant process variations. Resolution enhancement and other design for manufacturability techniques become less applicable as scaling continues. Alternatively, nanoelectronic systems are expected to be based on bottom-up self-assembly based manufacturing processes, e.g., molecular beam epitaxy (MBE). Such bottom-up self-assembly manufacturing processes provide regular structures, e.g., perfectly aligned carbon nanotubes (23). Consequently, nanoelectronic systems need to rely on reconfigurability to achieve functionality and reliability (51).
2. Reliability. Technology scaling has led to increasingly significant process and system runtime variations, including critical dimension variation, dopant fluctuation, electromagnetic emission, alpha particle radiation and cosmos ray strikes. Such variations cannot be avoided by manufacturing process improvement, and is inherent at nanometer

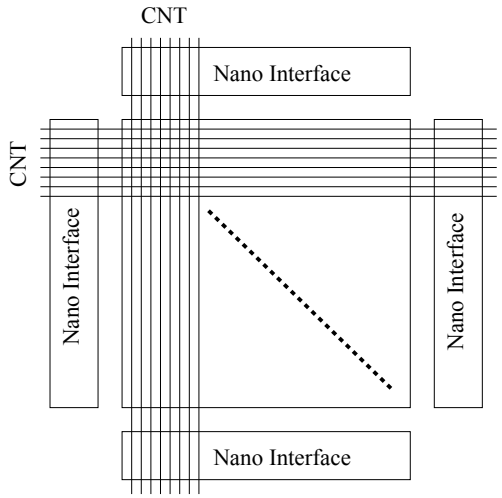


Fig. 1. The proposed CNT crossbar nano-architecture: layers of orthogonal carbon nanotubes form a dense array of RDG-CNFETs and programmable interconnects with voltage-controlled nano-addressing circuits on the boundaries.

scale due to the uncertainty principle of quantum physics. Robust design techniques, including redundant, adaptive, and resilient design techniques at multiple (architecture, circuit, layout) levels, are needed to achieve a reliable nanoelectronic system (5).

3. Performance. Nanoscale devices have achieved ultra-high performance in the absence of load, however, nanoelectronic system performance bottleneck lies in global interconnects. Rent's rule states that the maximum interconnect length scales with the circuit size in a power law (24), while signal propagation delay across unit length interconnect increases as technology scales (30). As a result, interconnect design will be critical to nanoelectronic system performance.
4. Power consumption. As technology scaling leads to increased device density and design performance, power consumption is also expected to be critical in nanoelectronic design.

This chapter presents several recent technical advancements towards manufacturable, reliable, high performance and low power nanoelectronic systems.

1. The first purely CNT and CNFET based nano-architecture, which is constructed by layers of orthogonal CNTs with via-forming and gate-forming molecules sandwiched in between, forming a dense array of reconfigurable double gate carbon nanotube field effect transistors (RDG-CNFETs) and programmable interconnects. Such a CNT array is addressed by novel voltage-controlled nano-addressing circuits on the boundaries, which do not require precise layout design and achieve yield in aggressive scaling and adaptivity to process variations. Simulation based on CNFET and molecular device compact models demonstrates superior logic density, reliability, performance and power consumption for nano-circuits implemented in this CNT crossbar based nano-architecture compared with the existing, e.g., molecular diode and MOSFET based nano-architectures.

2. A complete set of linear complexity methods for adaptive configuration of nanoelectronic systems based on a CNT crossbar based nano-architecture (Fig. 1) (26), including (1) adaptive nano-addressing, (2) RDG-CNFET gate matching, and (3) catastrophic defect mapping methods. Compared with the previous nano-architecture defect mapping and adaptive configuration proposals, these methods are complete, specific, deterministic, of low runtime complexity. These methods demonstrate the promising prospect of achieving nanoelectronic systems of correct functionality, performance, and reliability based on the CNT crossbar nano-architecture.
3. Robust Differential Asynchronous (RDA) circuits as a promising paradigm for reliable (noise immune and delay insensitive) high performance and low power nano-circuits based on the CNT crossbar nano-architecture. Theoretical analysis and SPICE simulation based on $22nm$ CMOS Predictive Technology Models show that RDA circuits achieve much enhanced reliability in logic correctness in the presence of a single bit soft error or common multiple bit soft errors, and timing correctness in the presence of parametric variations given the physical proximity of the circuit components.

The rest of this chapter is organized as follows. Section 2 reviews the existing nanoelectronic devices, nano-architectures and nano-addressing circuits. Section 3 presents the proposed CNT crossbar based nano-architecture including a novel RDG-CNFET device, a multi-layer CNT crossbar structure, and a voltage-controlled nano-addressing circuit. Section 4 presents adaptive configuration methods for nanoelectronic systems based on the CNT crossbar nano-architecture. Section 5 presents robust differential asynchronous circuits as a promising nano-circuit paradigm. Section 6 presents simulation results which evaluate the CNT crossbar nano-architecture and robust differential asynchronous nano-circuits. Section 7 concludes this paper with a list of nanotechnologies which enable and improve the proposed CNT crossbar based nano-architecture.

2. Background

2.1 Existing Nanoscale Devices

Carbon nanotube is one of the most promising candidates for interconnect technology at nanometer scale, due to its extraordinary properties in electrical current carrying capability, thermal conductivity, and mechanical strength. A carbon nanotube is a one-atom-thick graphene sheet rolled up in a cylinder of a nanometer-order diameter, which is semiconductive or metallic depending on its chirality. The cylinder form eliminates boundaries and boundary-induced scattering, yielding electron mean free path on the order of micrometers compared with few tens of nanometers in copper interconnects (32). This gives extraordinary current carrying capacity, achieving a current density on the order of $10^9 A/cm^2$ (56). However, large resistance exists at CNT-metal contacts, reducing the performance advantage of CNTs over copper interconnects (38).

Among various nanotechnology devices, carbon nanotube field effect transistors are the most promising candidates to replace the current CMOS field effect transistors as the building blocks of nanoelectronic systems. Three kinds of carbon nanotube based field effect transistors (CNFETs) have been manufactured: (1) A Schottky barrier based carbon nanotube field effect transistor (SB-CNFET) consists of a metal-nanotube-metal junction, and works on the principle of direct tunneling through the Schottky barrier formed by direct contact of metal and semiconducting nanotube. The barrier width is modulated by the gate voltage. This device has the most mature manufacturing technique up to today, while two problems limit its

future: (a) The metal-nanotube contact severely limits current. (b) The ambipolar conduction makes this devices cannot be applied to conventional circuit design methods. (2) A MOSFET-like CNFET is made by doping a continuous nanotube on both sides of the gate, thus forming the source/drain regions. This is a unipolar device of high on-current. (3) A band-to-band tunneling carbon nanotube field effect transistor (T-CNFET) is made by doping the source and the drain regions into p^+ and n^+ respectively. This device has low on-current and ultra low off current, making it potential for ultra low power applications. It also has the potential to achieve ultra fast signal switching with $< 60mV/decade$ subthreshold slope (46).
Molecular electronic devices are based on two families of molecules: the catenanes which consist of two or more interlocked rings, and the rotaxanes which consist of one or more rings encircling a dumbbell-shaped component. These molecules can be switched between states of different conductivities in a redox (reduction/oxidation) process by applying currents through them, providing reconfigurability for nanoscale devices (44).
A variety of reconfigurable nanoscale devices have been proposed. Resonant tunneling diodes based on redox active molecules are configurable on/off (44). Nanowire field effect transistors with redox active molecules at gates are of high/low conductance (17). Spin-RAM devices are of high/low conductivity based on the parallel/anti-parallel magnetization configuration of the device which is configured by the polarity of the source voltage (40). A double gate Schottky barrier CNFET is configurable to be a p-type FET, an n-type FET, or off, by the electrical potential of the back gate (25). A double gate field effect transistor with the back gate driven by a three state RTD memory cell is configurable to be a transistor or an interconnect, reducing reconfiguration cost of a gate array (4).

2.2 Existing Nanoelectronic Architectures

At least three categories of nanoelectronic architectures have been proposed. An early nanoelectronic architecture NanoFabrics was based on molecular resonant tunneling diodes (RTDs) and negative differential resistors (NDRs) (20). The insightful authors have observed that passive device (diode/resistor) based circuits lack signal gain to recover from signal attenuation, while combining with CMOS circuits compromises scaling advantages. They proposed latches based on negative differential resistors (NDRs), which, unfortunately, have become obsolete since the publication.
The majority of the existing nanoelectronic architectures are based on a hybrid nano-CMOS technology, with CMOS circuits complementing nano-circuits. In FPNI (50) (CMOL (54)), a nanowire crossbar is placed on top of CMOS logic gates (inverters). The nanowires provide programmable interconnects (and wired-OR logic), while the CMOS gates(inverters) provide logic implementation (signal inversion and gain). Such architectures achieve compromised scaling advantage in term of device density. DeHon (11; 13) proposed to combine programmable nanoscale diode logic arrays with fixed simple CMOS circuitry, e.g., of precharge and evaluation transistors as in domino logic for signal gain. Sequential elements need also to be implemented as CMOS circuits. However, the optimal size of a combinational logic block is typically small (e.g., of 30-50 gates), which results in significant CMOS circuitry overhead in such architectures. An exception is memory design, where CMOS technology provides peripheral circuitry such as address decoders and read sensors with moderate overhead, while nanotechnology provides scaling advantage in memory cells (17; 47; 63).
The third category of existing nanoelectronic architectures rely on DNA-guided self-assembly to form 2-D scuffles for nanotubes (42; 43) or 3-D DNA-rods (14). Such technologies target application in the far future.

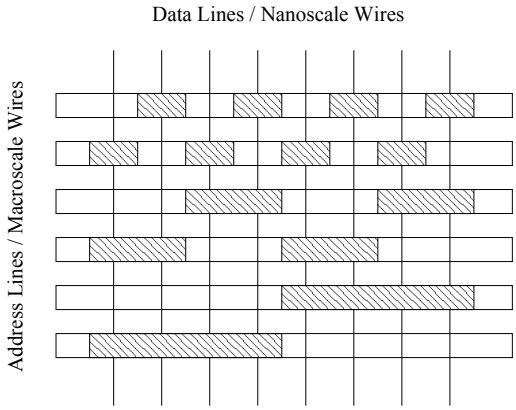


Fig. 2. Layout of undifferentiated nanoscale wires (data lines) addressed by microscale wires (address lines). Lithography defines high- and low-k dielectric regions, which gives field effect transistors and direct conduction, respectively.

2.3 Existing Nano-Addressing Circuits

A nano-addressing circuit selectively addresses a nanoscale wire in an array, and enables data communication between a nano-system and the outside world. The existing nano-addressing circuits are based on binary decoders, with an array of (microscale) address lines running across the (nanoscale) data lines, forming transistors at each crossing (e.g., Fig. 2). Each data line is selected by a unique binary address, given each data line has a unique gate configuration. However, such precise layout design is highly unlikely to achieve at a sublithographic nanometer scale (without significantly compromised yield).
In details, the existing nano-addressing circuits are in four categories as follow.

1. Randomized contact decoder (59) includes gold particles which are deposited at random as contacts between nanoscale and microscale wires. Testing and feedback provide a one-to-one mapping between a nanoscale wire and an address.
2. Undifferentiated nanoscale wires are addressable by microscale wires with (e.g., lithography defined) different gate configurations (which requires nanoscale wire spacing in the same order of lithography resolution) (22) (Fig. 2).
3. Alternatively, different gate configurations are realized in the nanoscale wires, by growing lightly-doped and heavily-doped carbon nanotubes of different length alternatively, while the microscale wires are undifferentiated. A microscale wire crossing a lightly-doped nanotube segment forms a gate, while a heavily-doped nanotube segment is always conductive for all possible signals in the microscale wire. In such a case, precise control of the lengths of the lightly- and heavily-doped nanotube segments would be critical (12; 21).
4. In radial addressing, multi-walled carbon nanotubes are grown with lightly- and heavily-doped shells, an etching process removes the heavily-doped outer shells at precise locations, and defines the gate configurations at each crossing of nanoscale and microscale wires (48).

Because process variations are inevitably significant at nanometer scale, these existing nano-addressing structures achieve limited yield, e.g., there is certain probability that two nanoscale wires have identical or similar gate configuration due to process variation. Furthermore, nanoscale wires are mostly partially selected, e.g., they may not achieve the ideal conductivity upon selected, due to process variations such as misalignment, dopant variation, etc.

2.4 Existing Nano-Architecture Defect-Mapping and Adaptive Configuration Methods

Existing nano-architecture defect mapping techniques are as follow. (1) On a Teramac reconfigurable computing platform, signals are propagated along each row or each column in a crossbar structure, a defect is located at the intersection of a defective row and a defective column, based on the assumption that a single defect is present (10). (2) In the NanoFabrics nano-architecture, the roughly estimated number of defects for a subset of computing resources are collected by counter or none-some-many circuits, a simple graph based algorithm or a Bayes' rule based probabilistic computation procedure gives defect occurrence probability estimates. E.g., highly likely defects are detected in the *probability assignment* phase, which accumulates defect probability in different test configurations, while less likely defects are located in the *defect location* phase, which incrementally clears certain spots as non-defects during test of different configurations (33). (3) A Build-In Self-Test (BIST) method in the NanoFabrics nano-architecture brings much increased complexity with limited applicability (in finding available defect-free neighboring nanoBlocks to implement test circuitry) (8; 58).
After a defect map is achieved presumably, logic circutis can be constructed avoiding or utilizing the defects. For example, a nanoPLA block can be synthesized in the presence of defective crosspoints (36), a CNT nano-circuit layout can be synthesized in the presence of misaligned and mispositioned CNTs (41), metallic CNTs (61), and CNTs of variational density (62).

2.5 Existing CNT Nano-Circuit Design

A very limited number of primitive combinational logic circuits have been fabricated based on CNFETs, including an inverter and two NOR gates in NMOS logic based on SB-CNFETs (3), and a five-inverter ring oscillator based on MOSFET-like CNFETs (9). While nano-circuits based on ambipolar SB-CNFETs need different topologies (3; 46; 53), nano-circuits based on unipolar MOSFET-like CNFETs can be identical to CMOS circuits (9).

3. CNT Crossbar based Nano-Architecture

As we have seen, most existing nanoelectronic architectures are based on diode/resistor logic and CMOS/nano-technologies (11; 13; 20; 50; 54; 63), which only achieve limited manufacturability, reliability, and performance. Carbon nanotubes (CNTs) and carbon nanotube field effect transistors (CNFETs) are the most promising candidates as the the building blocks of nanoelectronic systems due to their extraordinary properties. CNTs possess excellent electrical current carrying capability, thermal conductivity, and mechanical strength. CNFETs are potential to achieve high on-current, ultra-low off-current, and ultra-fast switching ($< 60mV/decade$ sub-threshold slope). CNT crossbar structure (Fig. 1) is one of the most promising candidates for nanoelectronic design platform. Recently, UIUC researchers have achieved fabrication of dense perfectly aligned CNT arrays (23). Such a CNT crossbar structure forms the basis of nanoscale memories (17; 47; 63).
However, no nanoelectronic architecture has been proposed which is solely based on CNTs and CNFETs. The reasons include lack of (1) a reconfigurable CNT based device which could provide functionality and reliability, (2) a self-assembly process which forms complex CNT

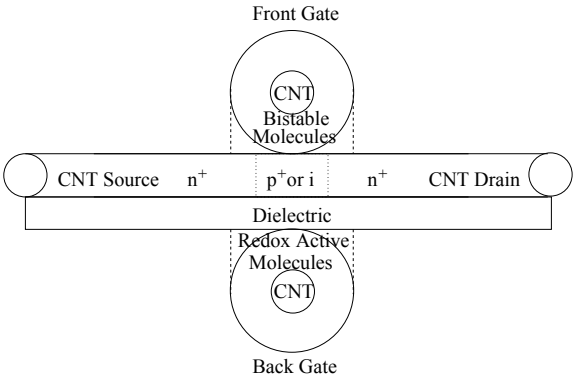


Fig. 3. A n-type MOSFET-like reconfigurable double gate carbon nanotube field effect transistor (RDG-CNFET).

structures, and (3) an achievable mechanism which precisely addresses an individual CNT in an array.

In this section, we investigate the first purely CNT and CNFET based nano-architecture, which is based on a novel RDG-CNFET device, includes a CNT crossbar structure on multiple layers, and a novel voltage-controlled nano-addressing circuit.

3.1 RDG-CNFET Device Structure

As the building block of a purely CNT and CNFET based nano-architecture, a reconfigurable double-gate CNFET (RDG-CNFET) is constructed by sandwiching electrically bistable molecules in a double gate CNFET. The double gate CNFET is constructed by three overlapping orthogonal carbon nanotubes. The top and the bottom carbon nanotubes form the front gate and the back gate, while doping the carbon nanotube in the middle layer forms the source and the drain of a n- or p-type MOSFET-like CNFET (46). Electrically bistable molecules are coated around the front gate and sandwiched between the front gate and the source/drain regions. Dielectric and redox active molecules are coated around the back gate and sandwiched between the back gate and the source/drain regions (Fig. 3).

The redox active molecules at the back gate are electrically reconfigurable to hold/release charge in a redox process, which controls the CNFET threshold voltage and conductance, or, turns the CNFET on or off. An example of such configuration is reported in (17), wherein a $\pm 10V$ voltage applied to cobalt phthalocyanine (CoPc) molecules triggers a redox process, and results in a NW-FET conductance change of nearly 10^4 times. Such reconfiguration of CoPc molecules is repeatable for more than 100 times.

The bistable molecules sandwiched between the front gate and the source/drain regions are electrically reconfigurable to be conductive or insular, making the device a via or a FET. An example of such electrically bistable molecules is reported in (44), wherein oxidative degradation reduces resonant tunneling current of the V-shaped amphiphilic [2]-rotaxane 5^{4+} molecules by nearly a factor of 100. Alternatively, the anti-fuse technologies in the existing reconfigurable architectures provide one-time configurability. For example, the QuickLogic ViaLink technology include a layer of amorphous silicon sandwiched between two layers of metal. A $10V$ programming voltage provides a resistance difference between $G\Omega$ and 80Ω (6).

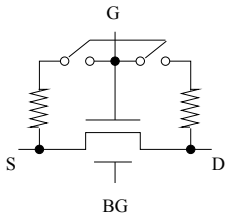


Fig. 4. Compact model of a n-type MOSFET-like reconfigurable double gate carbon nanotube field effect transistor (RDG-CNFET).

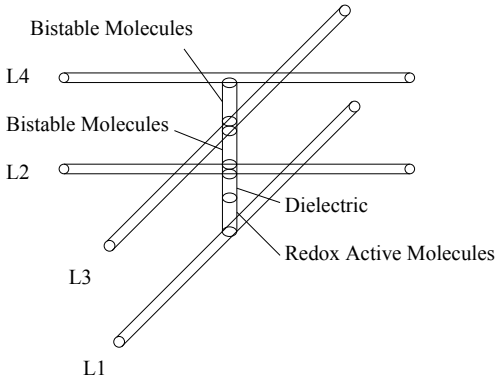


Fig. 5. Carbon nanotube (CNT) layers in the proposed nanoelectronic architecture.

3.2 RDG-CNFET Device Behavior

Such a RDG-CNFET device is described in a compact model as is shown in Fig. 4, and is reconfigurable to the following components, making it an ideal nanoelectronic architecture building block.

1. Via, when the front gate bistable molecules are configured to be conductive. The overlapping of the front gate and the source/drain regions form conductive contacts. As a result, the front gate, the source, and the drain are short circuited. The device is configured as a via between the carbon nanotubes on the top and in the middle.
2. Short, when the front gate bistable molecules are configured to be insular, and the back gate redox active molecules are configured to hold positive(negative) charge in a n-type(p-type) CNFET. The CNFET is on for any front gate voltage.
3. MOSFET-like CNFET, when the front gate bistable molecules are configured to be insular, and the back gate redox active molecules are configured to hold negative(positive) charge in a n-type(p-type) MOSFET-like CNFET. The CNFET threshold voltage is adjustable by the doping concentration in the channel (p or n doping for a n- or p-type CNFET), such that when the back gate redox active molecules are configured to hold negative(positive) charge in a n-type(p-type) MOSFET-like CNFET, the CNFET achieves both performance and leakage control.
4. Open, when the MOSFET-like CNFET is turned off. This is achieved at the architecture level as follows.

3.3 CNT Crossbar Structure

At a larger scale, a nanoelectronic architecture is constructed by growing layers of orthogonal carbon nanotubes, with via-forming (electrically bistable) and gate-forming (dielectric and redox active) molecules sandwiched at each crossing (Fig. 1). The carbon nanotubes are either (1) semiconductive CNTs which are doped to have low resistivity and are reconfigurable to opens by gate isolation, or (2) metallic CNTs which upon identification can be utilized as global interconnects if not avoided or removed (1; 64). The (1) via-forming (electrically bistable) and (2) gate-forming (dielectric and redox active) molecules can be first coated around a carbon nanotube (e.g., as in (17)), then undergo an etching process with the top layer of carbon nanotubes as masks (e.g., as in (49)). The remaining molecules are sandwiched between two orthogonal carbon nanotubes on adjacent layers. A top-down (e.g., lithography) process defines the areas for each type of molecules to assemble on each layer, as well as the p-wells and n-wells. P-type and n-type of MOSFET-like CNFETs are formed by (e.g., potassium or electrostatic (46)) doping of the carbon nanotubes selectively. E.g., a p-well or n-well of dimensions in the order of $22nm$ include about 10 rows of CNFETs.

Configuration of such a CNT crossbar based nanoelectronic architecture gives a nanoscale VLSI implementation including MOSFET-like CNFETs and interconnects with opens, shorts and vias (Fig. 6), which can be 2-D (compatible to traditional VLSI systems) or 3-D.

In a 2-D VLSI implementation, MOSFET-like CNFETs are formed on the bottom three layers of carbon nanotubes, with the first layer (L1) from bottom of carbon nanotubes provides the back gates, the second layer (L2) provides the source and the drain regions, and the third layer (L3) provides the front gates of the MOSFET-like CNFETs. Dielectric and redox active (back gate) molecules are sandwiched between the L1 and L2 layer carbon nanotubes, and electrically bistable (front gate) molecules are sandwiched between the L2 and L3 layer carbon nanotubes. A multi-layer reconfigurable interconnect structure with programmable vias and opens is achieved with via-forming and gate-forming molecules sandwiched between interconnects which are formed above the first (L1) layer (Fig. 5).

3-D VLSI circuits are under active research in recent years due to their potential of achieving reduced wirelength, reduced power consumption and improved performance. However, silicon based VLSI circuits are essentially 2-D, because MOSFETs are surface devices on the bulk of silicon, 3-D MOSFET circuits can only be achieved by bonding chips. It is therefore critical to achieve (1) bonding technology which provides acceptable mechanical strength, (2) via technology which provides low resistive interconnects between chips, and (3) heat dissipation in a multiple chip system for silicon based 3-D circuits. On the contrary, CNFET and CNFET based nano-architectures provide excellent platforms for 3-D VLSI circuits, because (1) CNTs and CNFETs are not confined to certain surface and can be manufactured in 3-D space, (2) CNTs possess excellent current carrying, mechanical and heat dissipation properties which are critical to 3-D VLSI circuits.

In a 3-D VLSI implementation, the RDG-CNFETs do not need to be confined on the bottom layers, with the upper layers dedicated to interconnects. Instead, transistors and interconnects are free to be located on each layer of carbon nanotubes. Gate forming (dielectric and redox active) molecules and via-forming (electrically bistable) molecules are distributed between adjacent CNT layers. Combination of the types of molecules surrounding a CNT segment gives three components.

1. Gate-forming molecules both on top and on bottom of a CNT segment give a device which is reconfigurable to either open or short,

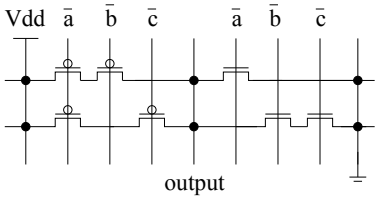


Fig. 6. An RDG-CNFET based Boolean logic $a(b+c)$ implementation.

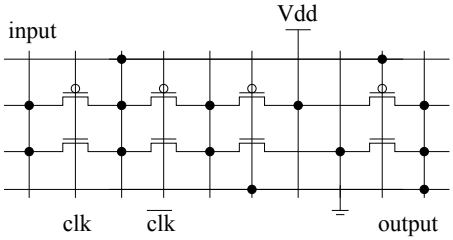


Fig. 7. An RDG-CNFET based latch implementation.

2. Gate-forming and via-forming molecules on top and on bottom of a CNT segment give the RDG-CNFET, which is reconfigurable to via, short, MOSFET-like CNFET, and open,
3. Via-forming molecules both on top and on bottom of a CNT segment give a device which is reconfigurable to be stacked via, simple via, or double gate FET.

We have the following observations.

Observation 1. *Via-forming (electrically bistable) molecules must be present between any two adjacent layers.*

Observation 2. *Gate-forming (redox active) molecules must be present next to each layer for gate isolation.*

Observation 3. *Gate-forming (redox active) and via-forming (electrically bistable) molecules need to be evenly distributed on each layer for performance.*

3.4 Circuit Paradigms and Analysis

This CNT crossbar based nano-architecture provides regularity and manufacturability for high logic density implementations of all CMOS logics, including the standard CMOS logic (e.g., in Fig. 6), domino logic, pass-transistor logic, etc., for combinational circuits, as well as latches (e.g., in Fig. 7), flip-flops, memory input address decoder and output sensing circuits. Such high logic density is achieved via direct connection of CNFETs through their source/drain regions (e.g., as in an latest Intel microprocessor implementation (15)), without going through additional (e.g., metal) interconnects. CNT-metal contacts are known to bring the most significant resistivity in CNT technology (38). Avoiding such CNT-metal contacts contributes to performance and reliability improvements. Furthermore, reduced interconnect length also leads to reduced interconnect capacitance, and improved circuit performance.
This CNT crossbar based nano-architecture also provides a high reconfigurability by allowing an arbitrary ratio of logic gates and interconnect switches (a RDG-CNFET device can be

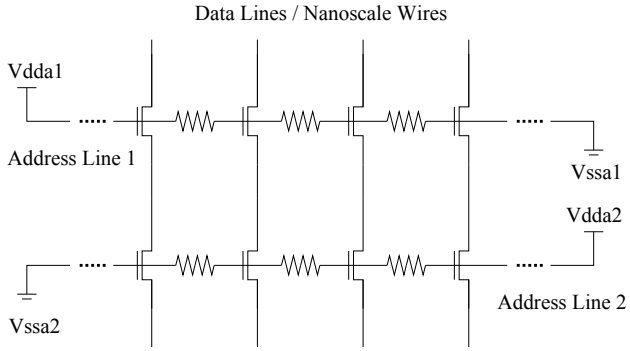


Fig. 8. Schematic of the proposed voltage-controlled nano-addressing circuit.

configured as either a logic gate or an interconnect switch). A pre-determined ratio of logic devices and interconnect switches (e.g., in standard cell designs and FPGA architectures where cells and routing channels are separated) constrains design optimization and may lead to inefficient device or interconnect utilization. Allowing an arbitrary ratio of logic gates and interconnect switches (e.g., as in sea-of-gate designs) provides increased degree of freedom for design optimization (4).

The CNT crossbar based nano-architecture is also the first to include multiple routing layers. Multiple routing layers (as in the current technologies) are necessary for VLSI designs, as Rent's rule suggests that the I/O number of a circuit module follows a power law with the gate number in the module (24). A small routing layer number could lead to infeasible physical design or significant interconnect detouring, resulting in degraded performance and device utilization.

3.5 Voltage Controlled Nano-Addressing Structure

The final piece of the CNT crossbar nanoelectronic architecture is the nano-addressing circuits on the boundary of the carbon nanotube crossbar structure.

Designing a nano-addressing circuit is a challenging task, because (1) the nanoscale layout cannot be manufactured precisely unless it is of a regular structure, and (2) the nano-addressing circuit cannot be based on reconfigurability since it provides reconfigurability to the rest of the nanoelectronic system.

A novel voltage-controlled nano-addressing circuit (Fig. 8) is constructed by running two address lines (of either microscale or nanoscale wires) on top of the data lines (of nanoscale wires in an array which are to be addressed). The address lines and the data lines are orthogonal. At each crossing of an address line and a data line, a field effect transistor is formed by doping the data line into the source and the drain regions while the address line provides the gate of the transistor, with a thin layer of dielectric sandwiched between the gate and the transistor channel. Such field effect transistors have been successfully fabricated based on either nanowires or carbon nanotubes (17; 37; 46).

3.6 Voltage-Controlled Nano-Addressing Principle

The address line provides the gate voltage for the transistors. Each address line is connected to two external voltages at the ends (V_{dda1} and V_{ssa1} for address line 1, V_{dda2} and V_{ssa2} for address line 2). The position of a nanoscale wire in the array gives the gate voltage for the transistor

on the nanoscale wire alone the address line. For example, a i-th nanoscale wire (starting from V_{ss}) in an array of n equally spaced nanoscale wires has a transistor gate voltage

$$V_g(i,n) = \frac{i}{n}V_{dd} + \frac{n-i}{n}V_{ss} \tag{1}$$

in an address line connecting to two external voltage sources V_{dd} and V_{ss}. Here we assume uniform address lines of negligible external resistance (from the first or the last nanoscale wire to the nearest external voltage source).

A transistor is on if its gate voltage exceeds the threshold voltage $V_g > V_{th}$. A nanoscale wire is conductive if both transistors on it are on. Because the two address lines provide an increasing series and a decreasing series of gate voltages respectively, only nanoscale wires at specific positions in the array are conductive. For example, for $V_{dda1} = V_{dda2}$ and $V_{ssa1} = V_{ssa2}$, the nanoscale wire in the middle of the array gets conductive.

In general, to select the i-th data line from the left in an array of n nanoscale wires, the external voltages need to be such that all the transistors on the right hand side of the i-th data line in the first address line are off, and all the transistors on the left hand side of the i-th data line in the second address line are off:

$$\begin{aligned} V_{ga1}(i+1,n) &= (1-\frac{i+1}{n})V_{dda1} + \frac{i+1}{n}V_{ssa1} < V_{th} \\ V_{ga2}(i-1,n) &= \frac{i-1}{n}V_{dda2} + (1-\frac{i-1}{n})V_{ssa2} < V_{th} \end{aligned} \tag{2}$$

3.7 Voltage Controlled Nano-Addressing Analysis

Compared with the existing nano-addressing circuits, the proposed voltage-controlled nano-addressing circuit leads to significant manufacturing yield improvement due to the following reasons.

The existing nano-addressing circuits are based on binary decoders and require every nanoscale wire have a *unique physical structure* to differentiate itself, which is highly unlikely in a nanotechnology manufacturing process - lithography cannot achieve nanoscale resolution, while bottom-up self-assembly based nanotechnology manufacturing processes provide only regular structures. Even at microscale, such a structure is subject to prevalent catastrophic defects and significant parametric variations, which result in low yield.

On the contrary, the proposed circuit consists of only uniform components in a regular structure. Every nanoscale wire has a *uniform physical structure* and is *differentiated by their electrical parameters, e.g., the node voltages*. This scheme avoids any precise layout design and significantly improves yield and enables aggressive scaling of the addressing circuit with the rest of the nanoelectronic system.

Furthermore, let us compare voltage-controlled nano-addressing with the existing binary decoder based nano-addressing mechanisms in terms of addressing accuracy and resolution. These two key quantitative metrics for nano-addressing circuits are defined as follow since such definition is not available in previous publications to the best of the author's knowledge.

Definition 1. Addressing inaccuracy *of a nano-addressing circuit is the offset between the target data line i and the data line j of maximum current.*

$$AI = |i-j| \tag{3}$$

In voltage-controlled nano-addressing, addressing inaccuracy is given by inaccurate addressing voltages from the voltage dividers. Such addressing inaccuracy can be further minimized by adjusting the external voltages to adapt to manufacturing process and system runtime parametric variations. As a result, a mis-addressing is localized, i.e., the data line j of maximum current is not far from the target data line i.
In traditional binary decoder based nano-addressing, an n-bit binary address has n neighboring binary addresses of Hamming distance 1. A 1-bit error could lead to n different mis-addressings. This leads to non-localized mis-addressing and a more significant addressing inaccuracy.

Definition 2. Addressing resolution *of a nano-addressing circuit is the minimum ratio between the on current $I_{on}(i)$ of a target data line i and the off current $I_{off}(j)$ of a non-target data line j (under all conditions, e.g., different inputs and parametric variations).*

$$AR = Min\{\frac{I_{on}(i)}{I_{off}(j)}\} \tag{4}$$

In traditional binary decoder based nano-addressing, the achievable addressing resolution depends on the conductance difference between the target data line and other non-target data lines. There are n non-target data lines with Hamming distance 1 for a n-bit target address, which have similar if not identical conductances. The presence of parametric variations further reduces addressing resolution.
In voltage-controlled nano-addressing, addressing resolution is largely given by the addressing voltage difference between two adjacent data lines. Applying high voltages leads to a number of reliability issues, such as electromigration and gate dioxide breakdown. Carbon nanotubes are highly resistive to electromigration, while new material is needed to enhance reliability for gate dioxide breakdown.
Alternatively, for given gate voltage difference, transistor current difference can be improved by improving the inverse subthreshold slope. However, MOSFETs and MOSFET-like CNFETs are limited to an inverse subthreshold slope S (which is the minimum gate voltage variation needed to bring a $10\times$ source-drain current increase) of $2.3\frac{kT}{q} \approx 60mV/decade$ at $300K$ (46). This requires development of novel devices for larger inverse subthreshold slopes.

4. Adaptive Configuration of Nanoelectronic Systems Based on the CNT Crossbar Nano-Architecture

In this section, we examine a list of nanoelectronic design adaptive configuration methods which cancel the effects of catastrophic defects and parametric variations in the proposed CNT crossbar nano-architecture.

4.1 Adaptive Nano-Addressing

A variety of parametric variations are expected to be prevalent and significant in nanoelectronic systems. Their effects on the voltage-controlled nano-addressing circuit (Fig. 8) are as follow.

1. Global address line resistance variations, e.g., due to uniform width, height, and/or resistivity variations of the address lines, have no effect on the voltage divider hence the addressing scheme.
2. Address line misalignment (shifting) has no effect on the conductances of the data lines.

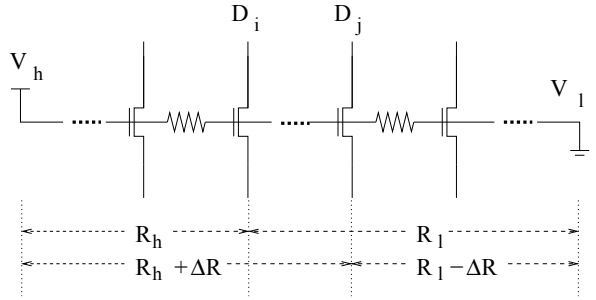


Fig. 9. Addressing two CNTs D_i and D_j with a resistance of ΔR in between.

3. Global data line misalignment (i.e., shifting of all data lines), variations of external voltage sources, and variations of external wire/contact resistance (between the resistive voltage divider and the external voltage sources) lead to potential addressing inaccuracy (CNT target offset).
4. Individual data line misalignment (shifting) could decrease the difference between the gate voltages of two adjacent transistors, leading to degraded addressing resolution (on/off CNT current ratio between two adjacent CNTs).
5. Process variations of the transistors, including width, length, dopant concentration, and oxide thickness variations, lead to transistor conductivity uncertainty and degraded addressing resolution.

The nano-addressing scheme needs to achieve a higher enough addressing resolution which endures the above-mentioned parametric variation effects (e.g., by applying high external addressing voltages, and/or novel CNFETs of $< 60mV/decade$ subthreshold slope).
After achieving satisfiable addressing resolution, we need to minimize any addressing inaccuracy and address the correct CNT data line (Problem 1).

Problem 1 (Adaptive Nano-Addressing)**.** *Given a voltage-controlled nano-addressing circuit, address the i-th CNT data line in the presence of parametric variations.*

Let us first derive the external voltage offset needed for a data address offset. Suppose for an address line, the external voltages V_h and V_l address the i-th CNT data line D_i. The resistance between CNT data line D_i and the high (low) external address voltage V_h (V_l) is R_h (R_l) (Fig. 9).[1] We have

$$\frac{R_l}{R_h + R_l} V_h + \frac{R_h}{R_h + R_l} V_l = V_{on} \tag{5}$$

where V_{on} is the voltage needed to address a CNT data line of peak current. Shifting the external voltages to $V_h + \Delta V$ and $V_l + \Delta V$ addresses another CNT data line D_j. The resistance between CNT data line D_j and the high (low) external address voltage is $R_h + \Delta R$ ($R_l - \Delta R$). We have

$$\frac{R_l - \Delta R}{R_h + R_l} (V_h + \Delta V) + \frac{R_h + \Delta R}{R_h + R_l} (V_l + \Delta V) = V_{on} \tag{6}$$

[1] For the first address line, the high external voltage $V_h = V_{l1}$ is on the left, the low external voltage $V_l = V_{r1}$ is on the right. For the second address line, the high external voltage $V_h = V_{r2}$ is on the right, the low external voltage $V_l = V_{l2}$ is on the left.

As a result,

$$\Delta V = \frac{\Delta R}{R_h + R_l}(V_h - V_l) \tag{7}$$

Observation 4. *The external voltage offset ΔV is proportional to the resistance offset ΔR between two CNT data lines, and is proportional to the physical offset ΔL between the two CNT data lines, if the resistive voltage dividers are uniform (e.g., the CNT data lines are equally spaced and the address lines have uniform resistivity).*

Based on Observation 1, Method 1 gives an adaptive nanoelectronic addressing method, which finds the external voltage shifts needed to address the left most and the right most CNT data lines first. Any other external voltage shift needed to address a specific CNT data line is then computed based on a linear interpolation. To address the left most or the right most CNT data line, we apply a gradually increasing/decreasing external voltage offset ΔV at an address line, keep all the transistors at the other address line on, and measure the conductance of the array of CNT data lines. The maximum and the minimum ΔV's (e.g., ΔV_{mink} and ΔV_{maxk}, $k = 1$ or 2) with non-zero CNT data line conductances address the left most and the right most CNT data lines, respectively.

Algorithm 1: Adaptive Voltage Controlled Nano Addressing

Input: An array of n CNT data lines, address i
Output: Addressing i-th data line

1. Turn on all transistors at address line 2 ($V_{l2} = V_{r2} > V_{th}$)
2. Find ΔV_{l1} which addresses first data line (binary search)
3. Find ΔV_{r1} which addresses n-th data line (binary search)
4. Turn on all transistors at address line 1 ($V_{l1} = V_{r1} > V_{th}$)
5. Find ΔV_{l2} which addresses first data line (binary search)
6. Find ΔV_{r2} which addresses n-th data line (binary search)
7. Shift V_{l1} and V_{r1} by $\frac{n-i}{n}\Delta V_{l1} + \frac{i}{n}\Delta V_{r1}$
8. Shift V_{l2} and V_{r2} by $\frac{n-i}{n}\Delta V_{l2} + \frac{i}{n}\Delta V_{r2}$

Observation 5. *The addressing accuracy given by Method 1 depends only on the uniformity of the resistive voltage divider, and the time domain variations of the external voltage differences $V_{l1} - V_{r1}$ and $V_{l2} - V_{r2}$. Any time-invariant (e.g., manufacturing process) variations of the external voltages (V_{l1}, V_{r1}, V_{l2}, and V_{r2}) or the external address line resistances (from the outer most data lines to the external voltage sources) do not affect the achievable addressing accuracy.*

4.2 RDG-CNFET Gate Matching

Another process variation is the misalignment of the front gate CNT and the back gate CNT of a reconfigurable double-gate CNFET (RDG-CNFET). This is because that the front gate CNT and the back gate CNT are on different ($i - 1$ and $i + 1$) layers, while CNT arrays on different layers do not have and are not expected to have a *precise* alignment mechanism.

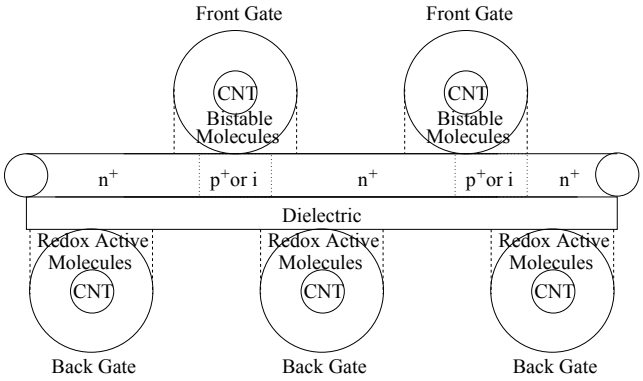


Fig. 10. CNT misalignment in dense CNT arrays. The closest CNT pair forms the front gate and the back gate of a RDG-CNFET. The neighboring CNTs have cross-coupling effect which needs to be simulated/tested or avoided by shielding.

Fortunately, we observe that precise alignment between a front gate CNT and a back gate CNT is not necessarily required as long as the CNT arrays are dense, e.g., with the spacing between CNTs close to the CNT diameters. In such a case, a double gate field effect transistor is formed even in the presence of CNT misalignment (Fig. 10). A CNFET channel is formed by doping the source/drain regions with the front gate CNT on the upper layer as mask. The resultant CNFET channel aligns with the front gate CNT. A misaligned back gate injects a weaker electrical field in the CNFET channel from a longer distance. A neighboring back gate may also injects a weak electrical field in the channel. This is either tolerated (which needs to be verified by simulation or testing) or avoided (by reserving the neighboring back gates for shielding).
The question is then how to find the closest CNT pair on different layers which form the front gate and the back gate of a RDG-CNFET (such that we can address them and configure the RDG-CNFET).

Problem 2 (RDG-CNFET Gate Matching)**.** *Given a CNT i on layer l, locate the closest CNT j on layer* $l + 2$ *(or* $l - 2$*) where CNTs i and j form the front gate and the back gate of a RDG-CNFET.*

Method 2 solves Problem 2 and finds the closest CNT pairs which form the front gate and the back gate of a RDG-CNFET.

Algorithm 2: RDG-CNFET Gate Matching

Input: CNT i on layer l which is a gate of CNFET T
Output: Closest CNT j to CNT i on layer $l+2$ ($l-2$) which is the other gate of CNFET T

1. Apply a turn-off gate voltage to CNT i
2. For each CNT j on layer $l+2$ ($l-2$),
3. Apply a turn-off gate voltage to CNT j
4. Measure the conductance of CNFET T
5. Find CNT j for the smallest CNFET conductance

Once a matching gate is identified, the CNFET can be characterized (by achieving its I-V curves). A parasitic CNFET can also be identified by finding the second closest CNT (with the second smallest CNFET conductance in the algorithm), which is either tolerated or avoided in a nanoelectronic design.

4.3 Catastrophic Defects and Mapping Techniques

In this subsection, we examine catastrophic defects for CNTs, programmable vias and CNFETs, and their corresponding detection and location methods.

4.3.1 Metallic, Open and Crossover CNTs

CNTs are metallic or semiconductive depending on their chirality. One third of CNTs are metallic if they are grown isotropically. Metallic CNTs can be removed by either chemical etching (64) or electrical breakdown (1). However, such techniques bring large process variation effects (34). Mitra et al. propose use of CNT bundles for each nanoelectronic signal to reduce metallic CNT effect (34). We observe that metallic CNTs need not necessarily to be removed and CNT bundles are not needed for each nanoelectronic signal as long as metallic CNTs can be detected and located. Upon detection and location, metallic CNTs can be configured to form global interconnects if not avoided. Their low resistivity helps to reduce signal propagation delay in global interconnects which are critical to nanoelectronic system performance.

Open CNTs are expected to be prevalent in a CNT array, as open CNT occurrence is proportional to the length of the CNT. A CNT with a single open can be largely included in a correct nanoelectronic design, upon detection and location of the single defect. A CNT with two (or more) opens is not fully utilizable. The segment between the two (extreme) opens are not accessible by any nano-addressing circuit, and components attached to that segment are not configurable. Upon detection and location of the extreme opens, the end segments of an open CNT can be included in a nano-circuit. Or, we can simply avoid open CNTs.

CNTs which are supposedly-parallel may cross over each other, resulting in different addresses for a CNT on two sides of a crossbar, and unexpected resistive contacts between CNTs. If not corrected by etching (34), such crossover CNTs can be taken as multi-thread cables and included in a correct nano-circuit. It is necessary to solve the following problem for nanoelectronic system configuration on a CNT crossbar nano-architecture.

Problem 3. *Detect and locate metallic, open and crossover CNTs in an CNT array, which are addressed on both ends by nano-addressing circuits.*

Such metallic, open, and crossover CNTs can be captured in a $n \times n$ resistance matrix $\mathbf{R}_{CNT}$, where each entry $R_{CNT}(i,j)$ gives the resistance of CNT between the i-th CNT end and the j-th CNT end on the opposite sides of an array of n CNTs (if $i \neq j$, $R_{CNT}(i,j)$ gives the resistance of a crossover CNT, otherwise, $R_{CNT}(i,i)$ gives the i-th CNT's resistance).
Method 3 solves Problem 3 by giving such a $n \times n$ resistance matrix $\mathbf{R}_{CNT}$. With this CNT resistance matrix $\mathbf{R}_{CNT}$, we avoid open CNTs, and consider only semiconductive CNTs, metallic CNTs, and crossover CNT bundles (as multi-thread cables) for the rest of the calibration (Methods 4 and 5 and 2).

Algorithm 3: Metallic, Open, Crossover CNT Detection and Location

Input: Array of n CNTs with nano-addressing circuits on both ends (Fig. 1)
Output: Resistance map $\mathbf{R}_{CNT}$ for metallic, open, crossover CNTs

1. Configure all CNFETs as shorts
2. For each i
3. For each j
4. Address the i-th CNT on one end of CNT
5. Address the j-th CNT on the other end of CNT
6. Measure resistance $R_{CNT}(i,j)$
7. If $i = j$ and $R_{CNT}(i,j) \approx \infty$
8. Open CNT (i,j)
9. If $i \neq j$ and $R_{CNT}(i,j) \ll \infty$
10. Crossover CNT (i,j)
11. If $R_{CNT}(i,j) \approx R_{metallic}$
12. Metallic CNT (i,j)
13. If $R_{CNT}(i,j) \approx R_{semiconductive}$
14. Semiconductive CNT (i,j)

4.3.2 Opens and Shorts in Programmable Vias

A CNT junction with electrically bistable molecules is a programmable via, which is supposedly reconfigured as a conductive via or open. A catastrophic defect at such a junction can be either (1) permanent open, or (2) permanent short. It is necessary to solve the following problem for nanoelectronic system configuration on a CNT crossbar nano-architecture.

Problem 4. *Detect and locate permanently open or short vias in a CNT crossbar nano-architecture.*

Method 4 solves Problem 4 by giving two $m \times n$ resistance maps $\mathbf{R}_{Pmin}$ and $\mathbf{R}_{Pmax}$, where each entry $R_{Pmin}(i,j)$ or $R_{Pmax}(i,j)$ gives the resistance of a L-shaped path which includes the i-th CNT segment on the top(bottom) of the CNT crossbar, the j-th CNT segment on the left(right) of the CNT crossbar, and a programmable via which is configured as conductive or open, respectively. Given non-open CNTs, these resistance matrices give a defect map for permanently open or short vias.

Algorithm 4: Permanently Open or Short Via Detection and Location

Input: Two layers of $m \times n$ CNT crossbar with nano-addressing interface on four sides
Output: Resistance maps $\mathbf{R}_{Pmin}$ and $\mathbf{R}_{Pmax}$ for permanently open or short vias

1. For each non-open CNT i
2. For each non-open CNT j
3. Address i-th CNT from top(down) of crossbar
4. Address j-th CNT from left(right) of crossbar
5. Program via $V(i,j)$ to conductive
6. Measure path resistance $R_{Pmin}(i,j)$
7. Program via $V(i,j)$ to insular
8. Measure path resistance $R_{Pmax}(i,j)$
9. If $R_{Pmin}(i,j) = R_{Pmax}(i,j) \approx \infty$
10. Permanently open via $V(i,j)$
11. If $R_{Pmin}(i,j) = R_{Pmax}(i,j) \approx R_{CNT}(i,i)$ or $R_{CNT}(j,j)$
12. Permanently short via $V(i,j)$

4.3.3 Opens and Shorts in CNFETs

A CNT junction with dielectric and redox active molecules is supposedly reconfigured as a FET. A catastrophic defect could lead to (1) short between source and drain (e.g., due to channel punchthrough, no intrinsic channel area, redox active molecules cannot release charge), (2) short between gate and source or drain (e.g., due to dielectric breakthrough), or (3) constant open gate (e.g., redox active molecules cannot hold charge). It is necessary to solve the following problem for nanoelectronic system configuration on a CNT crossbar nano-architecture.

Problem 5. *Detect and locate permanently open or short CNFETs in a CNT crossbar nano-architecture.*

Shorts between CNFET gate and source or drain can be detected in a method which is similar to Method 4 but without via programming. Method 5 finds permanent opens or shorts between the source and the drain of a CNFET by giving a $m \times n$ resistance matrix $\mathbf{R}_{CNFET}$. Upon detection and location, these catastrophic defects (metallic, open and crossover CNTs, permanently open or short vias and CNFETs) can be included in a correct nano-circuit. Nano-circuit physical design needs to be adaptive to the presence of these catastrophic defects, and will be different from die to die, based on the catastrophic defect maps ($\mathbf{R}_{CNT}$, $\mathbf{R}_{Pmin}$, $\mathbf{R}_{Pmax}$, and $\mathbf{R}_{CNFET}$) for each die.

Algorithm 5: Permanently Open or Short CNFET Detection and Location

Input: CNFETs in crossbar with nano-addressing interface, CNT resistance matrix $\mathbf{R}_{CNT}$
Output: Resistance map $\mathbf{R}_{CNFET}$ for permanently open or short CNFETs

1. Configure all CNFETs as shorts
2. For each non-open CNT i
3. For each non-open CNT j
4. Address the i-th CNT on both ends
5. Configure CNFET (i, j) as open
6. Measure resistance $R_{CNFET}(i, j)$
7. If $R_{CNFET}(i, j) \approx R_{CNT}(i, i) \ll \infty$
8. Short between CNFET (i, j) source-drain
9. If $R_{CNFET}(i, j) \approx R_{CNT}(i, i) \approx \infty$
10. Open between CNFET (i, j) source-drain
11. Configure CNFET (i, j) as short

4.4 Parametric Variation and Adaptive Design

Other than catastrophic defects, process variations are also critical to nanoelectronic system performance and reliability. Compared with catastrophic defects, process variations are more prevalent, and they are more difficult to detect since their effects are accumulated in affecting the underlying circuit. Adaptive or resilient nano-circuit design techniques are expected to achieve functionality and reliability in the presence of such process variations, besides online calibration and adaptive configuration as follows.

In adaptive configuration, each module of the circuit is configured with its test circuit. The test circuit can be as simple as additional interconnects which connect the inputs and the outputs of the module to some of the primary inputs and the primary outputs, respectively. In such cases, function and performance calibration is performed externally. Alternatively, self-test can be performed given the complexity of the test circuit. If the current configuration passes online function and performance verification, the auxiliary test circuit will be removed, and the current configuration of the module is committed. Otherwise, the same circuit module needs to be realized using other hardware resources on the reconfigurable platform.

5. Reliable, High Performance and Low Power Nano-Circuits

5.1 Nano-Circuit Design Challenges and Promising Techniques

As we have seen, the CNT crossbar nano-architecture provides regularity and manufacturability for high logic density implementations of all CMOS combinational logic families, including static logic, domino logic, pass-transistor logic, as well as latches, flip-flops, memory input address decoder and output sensing circuits such as differential sense amplifiers (Fig. 6 and Fig. 7).

However, nano-circuits in a CNT crossbar nano-architecture face a number of unique challenges. Nano-circuits must achieve reliability in the presence of prevalent defects and signif-

icant parametric variations, must achieve performance with highly resistive CNT interconnects, etc. We discuss nano-circuit design in a CNT crossbar nano-architecture in this section.
Nanoscale computing systems are expected to be subject to prevalent defects and significant process and environmental variations inevitably as a result of the uncertainty principle of quantum physics. E.g., the conductance of a CNT or a CNFET is very sensitive to chirality, diameter, etc. (38). Besides adaptive configuration, nanoscale computing systems need new computing models or circuit paradigms for reliability enhancement, performance improvement and power consumption reduction.
As technology scales, nanoelectronic computing systems are expected to be based on single electron devices (the average number of electrons in a transistor channel is approaching one for the current technologies). In quantum mechanics, the occurrence probability of an electron is the wavefunction given by the Schrödinger equation. How to extract a deterministic computation result from stochastic events such as electron occurrences is one of the fundamental problems that we are facing in designing nanometer scale computing systems. Traditional computation based on large devices can be modeled as redundancy and threshold based logic (which includes majority logic). In redundancy and threshold based logic, the error rate is given by a binomial distribution (the probability of observing m events in an environment of expecting an average of n independent events). As a result, a minimum signal-to-noise ratio is required with performance and power consumption implications. Finding a more efficient reliable computing model is of essential interest in nanoelectronic design.
Besides stochastic signal occurrence, signal propagation delay variability is another category of uncertainty in stochastic nanoscale systems. Nanoelectronic design needs to be adaptive to or resilient in the presence of signal propagation delay variations. Existing techniques (e.g., the Razor technology (2; 18) wherein a shadow flip-flop captures a delayed data signal for timing verification and correction) achieves only limited performance adaptivity, e.g., the circuity is adaptive to performance variations only within a given range. Asynchronous circuits have unlimited adaptivity to performance variations, and are ideal for high performance (enabling performance scaling in the presence of significant performance variations) and low power (being event-driven and clockless) nanoelectronic design (e.g., multi-core chips are expected to be increasingly self-timed, global-asynchronous-locally-synchronous, or totally asynchronous). However, existing asynchronous design techniques suffer in reliability in the presence of soft errors (e.g., glitches, coupling noises, radiation or cosmos ray strike induced random noises), which has limited their applications for decades.
A number of robust design techniques at multiple levels help to enhance reliability and reduce error rate of a nanoelectronic computing system. At the circuit level, differential signaling and complementary logic reduces parametric variation effects by exploiting spatial and temporal correlations (e.g., by correlating m and n for reduced error rate in a binomial distribution) (15; 31; 57). At a higher level, we believe that Error Detection/Correction Code (35) is the key to the stochastic signal occurrence problem in nanoscale systems (e.g., for lower required signal-to-noise ratios, which lead to high performance and low power), and needs to be applied more extensively at a variety of design hierarchy levels. Error correction coding has been applied widely in today's memories and wireless communication systems. Proposals also exist for applying (AN or residue) error detection/correction coding in arithmetic circuits (7).

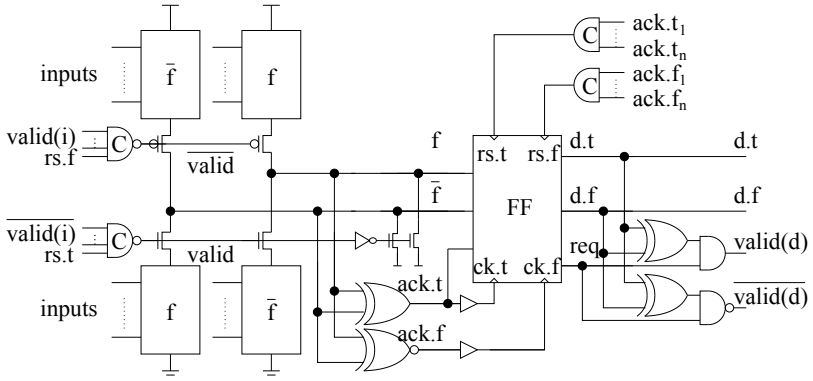


Fig. 11. A static logic based RDA (robust differential asynchronous) circuit.

5.2 Robust Asynchronous Circuits

A promising CNT nano-circuit paradigm is robust asynchronous circuits, which are formed by applying Error Detection/Correction Code to asynchronous circuit design for enhanced reliability in the presence of soft errors, while achieving delay insensitive nano-circuits (28).

For a 1-bit data signal, a robust differential asynchronous (RDA) communication channel includes three rails, two differential data signals $d.t$ and $d.f$, and a request signal req. The data and request signals are valid only if $(d.t, d.f, req) = (1,0,1)$ or $(0,1,1)$, while $(d.t, d.f, req) = (0,0,0)$ when the data and request signals return to zero. A validity check circuit detects the arrival of valid data and triggers the receiver flip-flop. The Hamming distance between any two legal codes for the data and request signals is at least two, instead of one in the dual-rail asynchronous signaling $(d.t, d.f) = (0,1), (1,0)$ or $(0,0)$. Differential acknowledgment signals $ack.t$ and $ack.f$ are also included.[2]

Fig. 11 gives a static logic based RDA circuit. [3] Two Muller C elements take the input validity signals $valid(i)$ and $\overline{valid(i)}$ (as well as the $rs.t$ and $rs.f$ signals coming from the downstream acknowledgment signals) and generate two differential validity signals $valid$ and $\overline{valid}$, which trigger the combinational logic computation. The Muller C elements have the maximum signal propagation delay for a rising(falling) output for all combinational logic circuits with the same number of inputs, due to the presence of the longest path of transistors to the power supply or the ground. Also the $valid$ and $\overline{valid}$ signals derive from and arrive later than the data signals. As a result, the $valid$ and $\overline{valid}$ signals always arrive later than the differential combinational logic outputs f and $\bar{f}$.

The later arriving validity signals filter out the internal glitches which come from combinational logic computation before the differential outputs f and $\bar{f}$ settle to their final values. Two NMOS transistors clamp the differential outputs f and the $\bar{f}$ to the ground until the $valid$ signal rises and the $\overline{valid}$ signal falls for noise immunity. The differential acknowledgment

[2] In general, multiple-bit data can be encoded in a variety of error detection codes (35). A single parity bit for n-bit data provides a Hamming distance of two which is immune to any single bit error, while an error detection code of a Hamming distance larger than k is immune to any k-bit error.

[3] Alternative implementations (in dynamic circuits, e.g., dual-rail domino, DCVSL, etc.) achieve different cost and reliability tradeoffs, and are potentially preferrable depending on the manufacturing technology and the environment, e.g., parametric variabilities, soft error rates, etc.

signals $ack.t$ and $ack.f$ derive from and always appear later than the differential data signals f and $\bar{f}$.
At the sender's end of the interconnect, for sequential elements, flip-flops are preferred over latches for reliability. A flip-flop is only vulnerable to noise when capturing the signal, while a latch is vulnerable to noise whenever it is transparent. The flip-flops send out differential data signals $d.t$ and $d.f$ (which come from the differential combinational logic outputs f and $\bar{f}$) as well as a request req signal (which comes from the acknowledgment signal $ack.t$). At the receiver end of the interconnect, a group of XOR and AND(NAND) gates verify the differential data and request signals, and generate two differential validity signals $valid(d)$ and $\overline{valid(d)}$. Any single bit soft error or common multiple bit soft errors injected to the interconnects or at the validity signals will halt the circuit.
Each sender flip-flop is triggered by two differential acknowledgment signals $ack.t$ and $ack.f$ as the differential clock signals $ck.t = 1$ and $ck.f = 0$, and is reset by two differential reset signals when $rs.t = 1$ and $rs.f = 0$. The differential reset signals come from the downstream differential acknowledgment signals $ack.t_i$ and $ack.f_i$ via the Muller C elements. They also generate the $valid$ and $\overline{valid}$ signals which trigger the combinational logic computation.
In the presence of multiple fanouts, multiple sets of differential acknowledgment signals will be sent back to the upstream stage. With the Muller C elements holding the input validity signals, the early arriving acknowledgment signals hold until the latest acknowledgment signal arrive from the fanouts. At that time the Muller C elements close the inputs to the combinational logic block and reset the flip-flop at the upstream stage, which brings all differential data and request signals $d.t$, $d.f$ and req as well as the acknowledgment signals $ack.t$ and $ack.f$ back to the ground, completing an asynchronous communication cycle.

5.3 Logic and Timing Correctness of RDA Circuits

An RDA circuit achieves logic and timing correctness in the presence of a single bit soft error or common multiple bit soft errors given the physical proximity of the circuit components.

Definition 3 (Single Bit Soft Error)**.** A single bit soft error *is a glitch or toggling caused by a single event upset as a result of an alpha particle or neutron strike from radioactive material or cosmos rays.*

Definition 4 (Common Multiple Bit Soft Error)**.** A common multiple bit soft error *is glitches or togglings of the same magnitude and polarity caused by common noises such as capacitive or inductive interconnect coupling, or spatially correlated transient parametric (e.g., supply voltage, temperature) variations (19; 39; 60), which have near identical effects on components at close physical proximity.*

Theorem 1 (Logic Correctness)**.** *An robust differential asynchronous circuit achieves logic correctness at the event of a single bit soft error or common multiple bit soft errors.*

Proof. An RDA circuit achieves logic correctness in the following cases.

1. A single bit soft error or a common multiple bit soft error at the input data signals leads to invalid data. The $valid(i)(\overline{valid(i)})$ signal will not rise(fall).
2. A single bit soft error or a common multiple bit soft error at the $valid(i)$ and $\overline{valid(i)}$ signals, at the Muller C elements computing the $valid$ and $\overline{valid}$ signals, or at the $valid$ and $\overline{valid}$ signals, leads to an early false or a late $valid(\overline{valid})$ signal. In this case, the differential structure for the $valid/\overline{valid}$ signals prevents any logic error. A false $valid$ signal turns off the keeper circuit, but can rise neither f nor $\bar{f}$, because the $\overline{valid}$ signal is still high. A false $\overline{valid}$ signal rises either f or $\bar{f}$, while the keeper circuit still clamps

both f and $\bar{f}$ to the ground. Only when both the $valid$ and $\overline{valid}$ signals arrive, the differential combinational logic computation is enabled.

3. A single bit soft error or a common multiple bit soft error at the differential combinational logic block or the differential data signals f and $\bar{f}$ will not raise the $ack.t$ signal nor lower the $ack.f$ signal.[4]
4. A single bit soft error or a common multiple bit soft error at the differential acknowledgment signals $ack.t$ and $ack.f$ does not trigger the flip-flop.
5. A single bit soft error or a common multiple bit soft error at the differential reset signals RS and $\overline{RS}$ does not reset the flip-flop.
6. A single bit soft error or a common multiple bit soft error at the differential data signals $d.t$ and $d.f$ and the request req signal leads to invalid data and does not generate a validity signal.

In summary, in order to make an RDA circuit to fail, the glitches must follow certain specific patterns, e.g., to reverse a "01" to a "10", which is highly unlikely to take place.

□

Theorem 2 (Timing Correctness)**.** *An robust differential asynchronous circuit achieves timing correctness for any delay variation given the physical proximity of the circuit components.*

Proof. Prevalent parametric (process, temperature, supply voltage) variations in nanoelectronic circuits lead to significant delay variations for the components in the circuit. Because such delay variations are spatially correlated (19; 39; 60), given the physical proximity of the circuit components, their delay variations are tightly correlated. Consequently, an RDA circuit achieves timing correctness in the following cases.

1. The input data signals $d.t$ and $d.f$ always arrive earlier than the differential validity signals $valid(d)$ and $\overline{valid(d)}$, which derive from the differential data and the request signals.
2. The $valid(\overline{valid})$ signal derives from the input validity signals $valid(i)$ and $\overline{valid(i)}$ through the Muller C elements.

   The rising(falling) delay of an Muller C element is the maximum of any combinational logic block of the same number of inputs with the longest serial transistor path to the power supply and the ground. Given the tight correlation of parametric variations for the transistors and the interconnects in the circuit, the $valid$ and $\overline{valid}$ signals arrive no early than the final combinational logic computation results f and $\bar{f}$. The $valid$ and $\overline{valid}$ signals enable the differential outputs f and $\bar{f}$ via the tri-state output structure and the keeper circuit, thus filtering out the glitches in combinational logic computation.
3. The differential acknowledgment signals $ack.t$ and $ack.f$ derive from and arrive no early than the differential data signals f and $\bar{f}$. They are further delayed (e.g., via buffers) such that at the rising edge of the flip-flop clock signal, the input data signals have settled to their final values, and the flip-flop captures the correct data f and $\bar{f}$. Consequently, no setup time constraint is required. The differential data signals f and $\bar{f}$ hold

[4] A glitch does not appear before valid data given the clamp NMOS transistors and fast data rise time in a static RDA circuit. A dynamic RDA circuit needs to enhance robustness for a soft error strike at a combinational logic output, e.g., by including weak keeper PMOS transistors, or delaying precharge PMOS transistors.

until the differential acknowledgment signals $ack.t$ and $ack.f$ reach the upstream stage, reset the upstream flip-flop, and lower the $valid(i)$ and $valid$ signals, which take much longer time than the hold time of the flip-flop. Consequently, no hold time constraint is required.

4. The differential acknowledgment signals $ack.t$ and $ack.f$ arrive after the combinational logic computation completes and the differential data signals f and $\bar{f}$ settle to their final values.
5. After all downstream stages send back acknowledgment signals, the flip-flop is reset, bringing the differential data $d.t$ and $d.f$ and the request req signals to the ground. The downstream stage acknowledgment signals are also brought back to the ground as a result. This completes a four-phase asynchronous communication cycle.

As a result, the proposed robust differential asynchronous circuit is delay insensitive, i.e., achieves correct timing (signal arrival time sequence) in the presence of delay variations, which is critical for nanoelectronic circuits. □

6. Experiments

6.1 Voltage-Controlled Nano-Addressing

In this section, we first verify the effectiveness of the proposed voltage-controlled nano-addressing circuit (Fig. 8) by running SPICE simulation based on the Stanford CNFET compact model (52).

In the proposed voltage-controlled nano-addressing circuit, each nanotube is gated by two N-type MOSFET-like CNFETs. These CNFETs are of $6.4nm$ gate width and $32nm$ channel length, as are described in the Stanford CNFET compact model. The two CNFETs in each nanotube are given a voltage drop of $V_{dd} = 1V$. The external address voltages are $V_{dda1} = V_{dda2} = 1V$, $V_{ssa1} = V_{ssa2} = 0$. As a result, the CNFETs have complementary gate voltages $V_{g1} + V_{g2} = 1V$. Fig. 12 gives the nanotube currents in the array with different gate voltage at the first address line. The nanotubes carry a significant current only with specific gate voltages, e.g., reaching $I_{out} = 5.064mA$ at gate voltage $V_{g1} = 0.495V$.

With 0 and $1V$ external voltages, Fig. 12 gives the currents for all the nanotubes in the array. With larger external voltages, Fig. 12 is extended to give the nanotube currents: any nanotube with a $V_{ga1} > 1V$ or $V_{ga2} < 0V$ gate voltage at the first address line carries zero current. Addressing resolution is given by the difference of addressing voltages between two adjacent nanotubes (since MOSFETs and MOSFET-like CNFETs are limited to a $< 60mV/decade$ inverse subthreshold slope). Adjusting the external address voltages minimizes any addressing inaccuracy due to manufacturing process and system runtime parametric variations.

6.2 Comparison of CNT Crossbar based and the Existing Nano-Architecture

Let us now compare nano-circuits implemented in the proposed CNT crossbar based nano-architecture and the existing nano-architectures. Considering DNA-guide self-assembly based nanoelectronic architectures such as NANA (42) and SOSA (43) target the far future, and FPNI (50) is very similar to CMOS technology by employing CMOS transistors and nanowires, I will compare RDG-CNFET based logic implementation with molecular diode and MOS transistor based logic implementation which is the mainstream nanoelectronic architecture in literature.[5]

[5] Comparing CNFET and CMOS-FET circuits gives approximately 5× performance improvement (16).

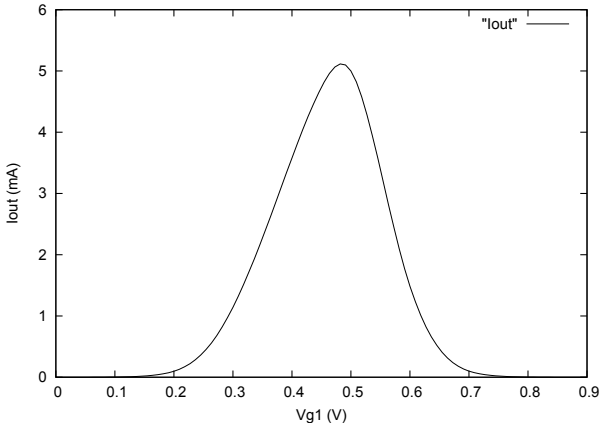


Fig. 12. Nanotube current I_{out} in mA for CNFET gate voltage V_{g1} in the first address line.

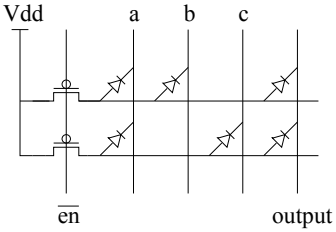


Fig. 13. A molecular diode/MOSFET based Boolean logic $a(b+c)$ implementation.

As an example of a combinational logic block, a Boolean logic function $a(b+c)$ is implemented based on RDG-CNFETs (Fig. 6) and by molecular diodes and peripheral CMOS transistors (Fig. 13). In the following experiments, SPICE simulation is conducted based on the latest Stanford compact CNFET model (52), a molecular device model from a latest publication (17), and the latest Predictive CMOS Technology Model (45).

The RDG-CNFETs are constructed based on an enhancement mode CNFET of $6.4nm$ gate width and $32nm$ channel length, as is described in the Stanford compact model (52). The bistable molecules at the front gate provide a resistance difference between $G\Omega$ and about 80Ω (6).[6] The redox active molecules at the back gate are cobalt phthalocyanine (CoPc) (17), which have been the basis of a NW-FET device with $1000\times$ conductance difference (17).

The molecular diodes are based on V-shaped amphiphilic [2]rotaxane 5^{4+} molecules (44), with saturation current $I_s = 36pA$, emission coefficient $N = 14.66$, and an on/off current ratio of 194.9. The CMOS transistors are modeled by $22nm$ Predictive Technology Models (45). To balance the current difference between molecular diodes and PMOS transistors, the PMOS transistors have a channel width/length ratio $W/L = 1/10$, while each molecular diode consists of $10,000$ V-shaped amphiphilic [2]rotaxane 5^{4+} molecules. As a result, the circuit has a current on the order of nA.

[6] The amorphous silicon based anti-fuse technology works with silicon based nanowires (6). Similar technologies are expected and assumed here for carbon nanotubes.

	Mo. Diode		RDG-CNFET	
abc	V_{out} (V)	P_{static} (W)	V_{out} (V)	P_{static} (W)
111	0.999	1.49n	1.000	0.25n
110	0.807	0.83μ	1.000	0.33n
101	0.807	0.83μ	1.000	0.32n
011	0.497	1.45μ	0.000	15.34p
000	0.265	1.47μ	0.000	41.32p

Table 1. Output voltage and static power consumption with different inputs of RDG-CNFET and molecular diode based Boolean logic $a(b+c)$ implementations.

Comparing the CNFET based and the molecular diode/CMOS based logic implementations, we have the following observations.

1. Area: The CNFET based logic implementation takes an area of $2 \times 6 = 12$ CNFETs and $2 \times 3 = 6$ vias, while molecular diodes and MOSFET based implementation takes an area of $2 \times 4 = 8$ molecular diodes and 2 MOSFETs (and two more MOSFETs if an inverter is included at each output to restore signal voltage swing). Considering CNFET based implementation is in a complementary logic, and the MOS transistors do not scale well, CNFET based implementation is expected to achieve superior logic density at a nanometer technology node.
2. Signal reliability: The CNFET based logic implementation achieves full voltage swing at the outputs, while in the diode logic circuit, the output swing depends on the inputs, and varies between $0.503V$ to $0.735V$ in the experiment (Table 1). Additional CMOS circuitry (e.g., an inverter) can be included at each output to restore full voltage swing, however, the reduced signal voltage swing in the diode logic circuit still implies compromised signal reliability.
3. Static power: The CNFET based logic implementation in CMOS logic achieves orders of magnitudes of less power consumption compared with molecular diodes and MOSFET based implementation for most input vectors (Table 1).
4. Performance: The CNFET based logic implementation achieves orders of magnitude of timing performance improvement compared with molecular diodes and MOSFET based implementation (Table 2).

In summary, CNFET based logic implementation achieves superior logic density, reliability, performance, and power consumption compared with molecular diodes and CMOS-FET based Boolean logic implementation.

6.3 Verification of RDA Circuits

In this section, we verify logic and timing correctness of robust differential asynchronous circuits by running HSPICE simulation based on $22nm$ Predictive Technology Models (45).
Fig. 14 gives signal waveforms for a perfect RDA circuit implementing Boolean function $f = \overline{ab}$ in CMOS static logic. Input signal a falls from logic 1 at $100ps$ to logic 0 at $200ps$, input signal b rises from logic 0 at $0ps$ to logic 1 at $50ps$. The validity signal $valid(\overline{valid})$ rises(falls) from logic 0(1) at $100ps$ to logic 1(0) at $200ps$. As a result, we observe that the late arrival of the $valid(\overline{valid})$ signals enable the combinational logic computation for f and $\bar{f}$, and filter out

	Mo. Diode		RDG-CNFET	
C_L (fF)	D_r (ns)	D_f (ns)	D_r (ns)	D_f (ns)
1	0.37	0.63	0.01	0.01
10	3.31	6.61	0.08	0.08
100	32.77	62.29	0.77	0.78

Table 2. Rising/falling signal propagation delays D_r/D_f (ns) (from a to $output$) for various load capacitance C_L (fF) of RDG-CNFET and molecular diode based Boolean logic $a(b+c)$ implementations. Input signal transition time varying from $1ps$ to $100ps$ leads to no considerable delay difference.

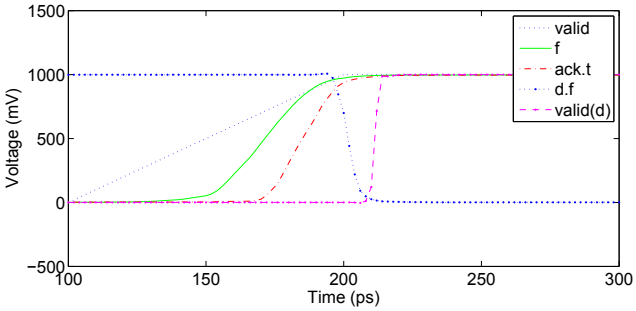


Fig. 14. Signal waveforms in a robust differential asynchronous circuit with no single bit soft error.

the internal glitches that would be observed at the f and $\bar{f}$ signals if the validity signals arrive early.

Fig. 15 gives signal waveforms for the same RDA circuit with the same input signals a and b, while the validity signal $valid$ is delayed by $50ps$ compared to the complementary validity signal $\overline{valid}$, representing an early false validity signal ($\overline{valid}$) or a late arriving validity signal ($valid$) due to an injected negative glitch at either the $\overline{valid}$ or the $valid$ signal. We observe that the differential data signals f and $\bar{f}$ are clamped to the ground until both validity signals settle to their final values $valid = 1$ and $\overline{valid} = 0$. As a result, no logic malfunction is present while all signals are delayed by $50ps$.

Fig. 16 gives signal waveforms for the RDA circuit with a triangle current (of $0.1mA$ peak current starting at $160ps$ ending at $180ps$) injected to the f signal. Comparing with Fig. 14, we observe that such a negative glitch does not lead to any logic error, instead, the arrivals of the $ack.t$ and $ack.f$ signals are postponed for about $40ps$, as well as all the downstream signals $d.t$, $d.f$, and $valid(d)$.

Fig. 17 gives signal waveforms for the RDA circuit with a triangle current (of $0.1mA$ peak current starting at $120ps$ ending at $140ps$) injected to the $ack.t$ signal. The glitch at the $ack.t$ signal does not trigger the flip-flop, and the subsequent signals $d.t$, $d.f$, $valid(d)$ and $\overline{valid(d)}$ are not affected.

Fig. 18 gives signal waveforms for the RDA circuit with two identical triangle currents (of $0.1mA$ peak current starting at $120ps$ ending at $140ps$) injected to both differential acknowl-

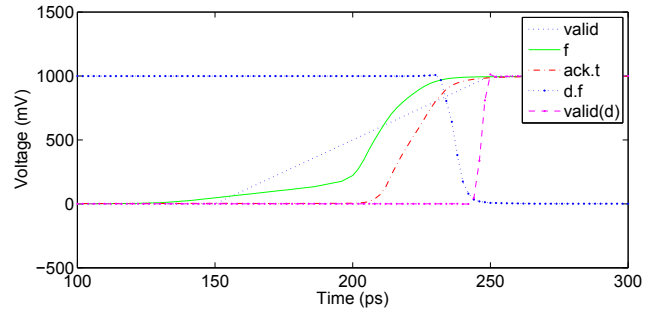


Fig. 15. Signal waveforms in a robust differential asynchronous circuit with an early false $\overline{valid}$ or a late arriving $valid$ signal.

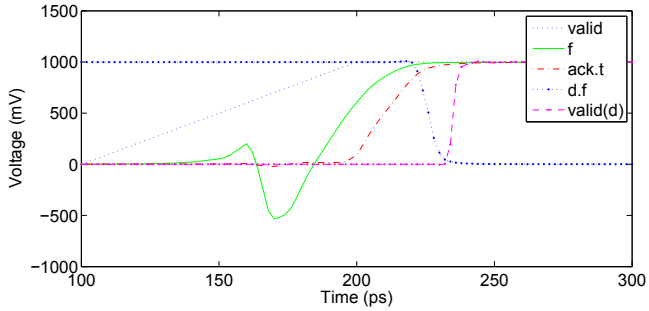


Fig. 16. Signal waveforms in a robust differential asynchronous circuit with a negative glitch injected at the f signal.

edgment signals $ack.t$ and $ack.f$. We observe that the double glitches do not trigger the flip-flop either, the subsequent signals $d.t$, $d.f$, $valid(d)$ and $\overline{valid(d)}$ are not affected.
From these experiments, we observe that the RDA circuit achieves correct logic and correct timing (signal arrival time sequence) at the event of a single bit soft error or common multiple bit soft errors, by temporarily halting the circuit operation until the valid data re-appear.

7. Conclusion

In this chapter, we have studied the first purely CNT and CNFET based nano-architecture, which is based on (1) a novel reconfigurable double gate carbon nanotube field effect transistor (RDG-CNFET) device, (2) a multi-layer CNT crossbar structure with sandwiched via-forming and gate-forming molecules, and (3) a novel voltage-controlled nano-addressing circuit not requiring precise layout design, enabling manufacture of nanoelectronic systems in all existing CMOS circuit design styles.
A complete methodology of adaptive configuration of nanoelectronic systems based on the CNT crossbar nano-architecture is also presented, including (1) an adaptive nano-addressing method for the voltage-controlled nano-addressing circuit, (2) an adaptive RDG-CNFET gate matching method, and (3) a set of catastrophic defect mapping methods, which are specific (to

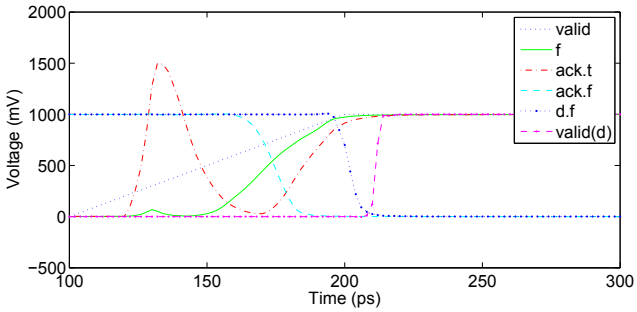


Fig. 17. Signal waveforms in a robust differential asynchronous circuit with a positive glitch injected at the *ack.t* signal.

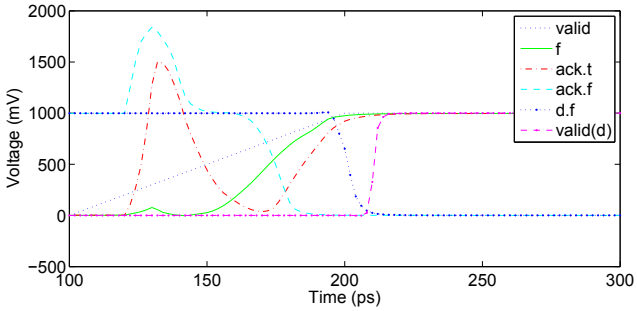


Fig. 18. Signal waveforms in a robust differential asynchronous circuit with positive glitches injected at both differential acknowledgment signals *ack.t* and *ack.f*.

the CNT crossbar nano-architecture), complete (in detecting and locating all possible catastrophic defects in the CNT crossbar nano-architecture), deterministic (with no probabilistic computation), and efficient (test paths are rows or columns of CNT, or L-shaped CNT paths, CNT open/short defect detection is separated with via/CNFET open/short defect detection, runtime is linear to the number of defect sites). This is significant improvement compared with the previous techniques (which either detects only a single defect (10), or is generic, abstract, probabilistic, and highly complex (8; 33; 58)).

We have also examined some of the design challenges and promising techniques for CNT and CNFET based nano-circuits. We identify significant parametric variation effects on logic correctness and timing correctness, and propose robust (differential) asynchronous circuits by applying Error Detection Code to asynchronous circuit design for noise immune and delay insensitive nano-circuits.

SPICE simulation based on compact CNFET and molecular device models demonstrates superior logic density, reliability, performance and power consumption of the proposed RDG-CNFET based nanoelectronic architecture compared with previously published nanoelectronic architectures, e.g., of a hybrid nano-CMOS technology including molecular diodes and MOSFETs. Furthermore, theoretical analysis and SPICE simulation based on 22nm Predictive Technology Models show that RDA circuits achieve much enhanced reliability in logic correct-

ness in the presence of a single bit soft error or common multiple bit soft errors, and timing correctness in the presence of parametric variations given the physical proximity of the circuit components.

While nanotechnology development has not enabled fabrication of such a system, this chapter has demonstrated the prospected manufacturability, reliability, and performance of a purely carbon nanotube and carbon nanotube transistor based nanoelectronic system. These nanoelectronic system design techniques are expected to be further developed along with nanoscale device fabrication and integration techniques which are critical to achieve and to improve the proposed nanoelectronic architecture in several aspects, including: (1) search of electrically bistable molecules of repeated reconfigurability and low contact resistance with carbon nanotubes, (2) development of etching processes for electrically bistable and redox active molecules with carbon nanotubes as masks, and (3) manufacture of nanoscale devices of superior subthreshold slope for enhanced nano-addressing resolution, and ultra high performance low power nanoelectronic systems.

8. References

[1] I. Amlani, N. Pimparkar, K. Nordquist, D. Lim, S. Clavijo, Z. Qian and R. Emrick, "Automated Removal of Metallic Carbon Nanotubes in a Nanotube Ensemble by Electrical Breakdown," *Proc. IEEE Conference on Nanotechnology*, 2008, pp. 239-242.

[2] T. Austin, V. Bertacco, D. Blaauw and T. Mudge, "Opportunities and Challenges for Better Than Worst-Case Design," *Asian South Pacific Design Automation Conference*, 2005.

[3] A. Bachtold, P. Hadley, T. Nakanishi and C. Dekker, "Logic Circuits with Carbon Nanotube Transistors," *Science*, 2001, 294(5545), pp. 1317-1320.

[4] P. Beckett, "A Fine-Grained Reconfigurable Logic Array Based on Double Gate Transistors," *International Conference on Field-Programmable Technology*, 2002, pp. 260-267.

[5] L. Benini and G. De Micheli, "Networks on chips: A new paradigm for component-based MPSoC design," *Proc. MPSoC.*, 2004.

[6] J. Birkner, A. Chan, H. T. Chua, A. Chao, K. Gordon, B. Kleinman, P. Kolze and R. Wong, "A Very High-Speed Field Programmable Gate Array Using Metal-To-Metal Anti-Fuse Programmable Elements," *New Hardware Product Introduction at Custom Integrated Circuits Conference,* 1991.

[7] T. J. Brosnan and N. R. Strader II, "Modular Error Detection for Bit-Serial Multiplication," *IEEE Trans. on Computers*, 37(9), 1988, pp. 1043-1052.

[8] J. G. Brown and R. D. Blanton, "CAEN-BIST: Testing the NanoFabric," *Proc. International Test Conference*, 2004, pp. 462-471.

[9] Z. Chen, J. Appenzeller, Y.-M. Lin, J. Sippel-Oakley, A. G. Rinzler, J. Tang, S. J. Wind, P. M. Solomon and P. Avouris, "An Integrated Logic Circuit Assembled on a Single Carbon Nanotube," *Science*, 2006, 311(5768), pp. 1735.

[10] B. Culbertson, R. Amerson, R. Carter, P. Kuekes and G. Snider, "Defect Tolerance on the Teramac Custom Computer," *Proc. Symposium on FPGA's for Custom Computing Machines (FCCM)*, 2000, pp. 185-192.

[11] A.DeHon, "Array-Based Architecture for FET-Based, Nanoscale Electronics," *IEEE Trans. on Nanotechnology*, 2(1), pp. 23-32, 2003.

[12] A. DeHon, P. Lincoln and J. E. Savage, "Stochastic Assembly of Sublithographic Nanoscale Interface," *IEEE Trans. Nanotechnology*, 2(3), pp. 165-174, 2003.

[13] A. DeHon and M. J. Wilson, "Nanowire-Based Sublithographic Programmable Logic Arrays," *Proc. FPGA*, 2004, pp. 123-132.

[14] C. Dwyer, L. Vicci, J. Poulton, D. Erie, R. Superfine, S. Washburn and R. M. Taylor, "The design of DNA self-assembled computing circuitry," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 12(11), pp. 1214-1220, Nov. 2004.

[15] D. J. Deleganes, M. Barany, G. Geannopoulos, K. Kreitzer, M. Morrise, D. Milliron, A. P. Singh and S. Wijeratne, "Low-Voltage Swing Logic Circuits for a Pentium 4 Processor Integer Core," *IEEE J. of Solid-State Circuits*, 40(1), pp. 36-43, 2005.

[16] J. Deng, and H.-S. P. Wong, "A Compact SPICE Model for Carbon Nanotube Field Effect Transistors Including Non-Idealities and Its Application âĂŤ Part II: Full Device Model and Circuits Performance Benchmarking," *IEEE Trans. Electron Devices*, 2007.

[17] X. Duan, Y. Huang and C. M. Lieber, "Nonvolatile Memory and Programmable Logic from Molecule-Gated Nanowires," *Nano Letters*, 2(5), pp. 487-490, 2002.

[18] D. Ernst, N. S. Kim, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge and K. Flautner, "Razor: Circuit-Level Correction of Timing Errors for Low-Power Operation," *IEEE MICRO special issue on Top Picks From Microarchitecture Conferences of 2004*, 2005.

[19] P. Friedberg, Y. Cao, J. Cain, R. Wang, J. Rabaey, and C. Spanos, "Modeling Within-Die Spatial Correlation Effects for Process-Design Co-Optimization," *Proc. International Symposium on Quality Electronic Design*, pp. 516-521, 2005.

[20] S. C. Goldstein and M. Budiu, "NanoFabrics: Spatial Computing Using Molecular Electronics," *Proc. International Symposium on Computer Architecture*, 2001, pp. 178-191.

[21] B. Gojman, E. Rachlin, and J. E. Savage, "Evaluation of Design Strategies for Stochastically Assembled Nanoarray Memories," *Journal of Emerging Technologies,* 1(2), 2005, pp. 73-108.

[22] J. R. Heath and M. A. Ratner, "Molecular Electronics," *Physics Today*, 56(5), pp. 43-49, 2003.

[23] S. J. Kang, C. Kocabas, T. Ozel, M. Shim, N. Pimparkar, M. A. Alam, S. V. Rotkin and J. A. Rogers, "High-Performance Electronics Using Dense, Perfectly Aligned Arrays of Single-Walled Carbon Nanotubes," *Nature Nanotechnology*, Vol. 2, pp. 230-236, April 2007.

[24] B. S. Landman and R. L. Russo, "On a Pin Versus Block Relationship for Partitions of Logic Graphs," *IEEE Trans. on Computers*, C-20, pp. 1469-1479, 1971.

[25] J. Liu, I. O'Connor, D. Navarro and F. Gaffiot, "Design of a Novel CNTFET-Based Reconfigurable Logic Gate," *Proc. ISVLSI*, 2007, pp. 285-290.

[26] B. Liu, "Reconfigurable Double Gate Carbon Nanotube Field Effect Transistor Based Nanoelectronic Architecture," *Proc. Asia and South Pacific Design Automation Conference*, 2009.

[27] B. Liu, "Adaptive Voltage Controlled Nanoelectronic Addressing for Yield, Accuracy and Resolution," *Proc. International Symposium on Quality Electronic Design*, 2009.

[28] B. Liu, "Robust Differential Asynchronous Nanoelectronic Circuits," *Proc. International Symposium on Quality Electronic Design*, 2009.

[29] B. Liu, "Defect Mapping and Adaptive Configuration of Nanoelectronic Circuits Based on a CNT Crossbar Nano-Architecture," *Workshop on Nano, Molecular, and Quantum Communications (NanoCom)*, 2009.

[30] International Technology Roadmap for Semiconductors, `http://www.itrs.net/`.

[31] A. Maheshwari and W. Burleson, "Differential Current-Sensing for On-Chip Interconnects," *IEEE Trans. on VLSI Systems*, 12(12), 2004, pp. 1321-1329.

[32] P. L. McEuen, M. S. Fuhrer and P. Hongkun, "Single-walled Carbon Nanotube Electronics," *IEEE Trans. Nanotechnology*, 1(1), pp. 78-85, 2002.

[33] M. Mishra and S. C. Goldstein, "Defect Tolerance at the End of the Roadmap," *Proc. International Test Conference*, 2003, pp. 1201-1211.

[34] S. Mitra, N. Patil and J. Zhang, "Imperfection-Immune Carbon Nanotube VLSI Logic Circuits," *Foundations of NANO*, 2008.

[35] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*, Wiley-Interscience, 2005.

[36] H. Naeimi and A. DeHon, "A Greedy Algorithm for Tolerating Defective Crosspoints in NanoPLA Design," *Proc. Intl. Conf. on Field-Programmable Technology*, 2004, pp. 49-56.

[37] P. Nguyen, H. T. Ng, T. Yamada, M. K. Smith, J. Li, J. Han and M. Meyyappan, "Direct Integration of Metal Oxide Nanowire in Vertical Field-Effect Transistor," *Nano Letters*, 2004, 4(4), pp. 651-657.

[38] A. Nieuwoudt and Y. Massoud, "Assessing the Implications of Process Variations on Future Carbon Nanotube Bundle Interconnect Solutions," *Proc. Intl. Symp. on Quality Electronic Design*, 2007, pp. 119-126.

[39] M. Orshansky, L. Milor, P. Chen, K. Keutzer, C. Hu, "Impact of spatial intrachip gate length variability on the performance of high-speed digital circuits," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2002, pp. 544-553.

[40] S. S. P. Parkin, "Spintronics Materials and Devices: Past, Present and Future," *IEEE International Electron Devices Meeting (IEDM) Technical Digest*, pp. 903-906, 2004.

[41] N. Patil, J. Deng, A. Lin, H.-S. Philip Wong and S. Mitra, "Design Methods for Misaligned and Mispositioned Carbon-Nanotube Immune Circuits," *IEEE Tran. on CAD*, 2008, 27(10), pp. 1725-1736.

[42] J. P. Patwardhan, C. Dwyer, A. R. Lebeck and D. J. Sorin, "NANA: A Nano-Scale Active Network Architecture," *ACM Journal on Emerging Technologies in Computing Systems,* 2(1), pp. 1-30, 2006.

[43] J. P. Patwardhan, V. Johri, C. Dwyer and A. R. Lebeck, "A Defect Tolerant Self-Organizing Nanoscale SIMD Architecture," *International Conference on Architecture Support for Programming Languages and Operating Systems,* 2006, pp. 241-251.

[44] A. R. Pease, J. O. Jeppesen, J. F. Stoddart, Y. Luo, C. P. Collier and J. R. Heath, "Switching Devices Based on Interlocked Molecules," *Acc. Chem. Res.*, 34, pp. 433-444, 2001.

[45] Predictive Technology Model, `http://www.eas.asu.edu/~ptm/`.

[46] A. Raychowdhury and K. Roy, "Carbon Nanotube Electronics: Design of High Performance and Low Power Digital Circuits," *IEEE Trans. on Circuits and Systems - I: Fundamental Theory and Applications,* 54(11), pp. 2391-1401, 2007.

[47] G. S. Rose, A. C. Cabe, N. Gergel-Hackett, N. Majumdar, M. R. Stan, J. C. Bean, L. R. Harriott, Y. Yao and J. M. Tour, "Design Approaches for Hybrid CMOS/Molecular Memory Based on Experimental Device Data," *Proc. Great Lakes Symposium on VLSI*, 2006, pp. 2-7.

[48] J. E. Savage, E. Rachlin, A. DeHon, C. M. Lieber and Y. Wu, "Radial Addressing of Nanowires," *ACM Journal of Emerging Technologies in Computing Systems,* 2(2), pp. 129-154. 2006.

[49] M. S. Schmidt, T. Nielsen, D. N. Madsen, A. Kristensen and P. BÃÿggild, "Nano-Scale Silicon structures by Using Carbon Nanotubes as Reactive Ion Masks," *Nanotechnology*, 16, pp. 750-753, 2005.

[50] G. S. Snider and R. S. Williams, "Nano/CMOS Architectures Using a Field-Programmable Nanowire Interconnect," *Nanotechnology*, 18(3), 2007.

[51] M. R. Stan, P. D. Franzon, S. C. Goldstein, J. C. Lach and M. M. Ziegler, "Molecular Electronics: From Devices and Interconnect to Circuits and Architecture," *Proc. of the IEEE*, 91(11), pp. 1940-1957, 2003.

[52] Stanford CNFET Model, `http://nano.stanford.edu/models.php`.

[53] R. Sordan, K. Balasubramanian, M. Burghard and K. Kern, "Exclusive-OR Gate with a Single Carbon Nanotube," *Appl. Phys. Lett.*, 2006, 88.

[54] D. B. Strukov and K. K. Likharev, "CMOL FPGA: A Reconfigurable Architecture for Hybrid Digital Circuits with Two-Terminal Nanodevices," *Nanotechnology*, 16(6), pp. 888-900, 2005.

[55] J. P. Sun, G. I. Haddad, P. Mazumder and J. N. Schulman, "Resonant Tunneling Diodes: Models and Properties," *Proc. of the IEEE*, 86(4), 1998, pp. 641-660.

[56] Y.-C. Tseng, P. Xuan, A. Javey, R. Malloy, Q. Wang, J. Bokor and H. Dai, "Monolithic Integration of Carbon Nanotube Devices with Silicon MOS Technology," *Nano Letters*, 4(1), pp. 123-127, 2004.

[57] N. Tzartzanis and W. W. Walker, "Differential Current-Mode Sensing for Efficient On-Chip Global Signaling," *IEEE J. of Solid-State Circuits*, 40(11), 2005, pp. 2141-2147.

[58] Z. Wang and K. Chakrabarty, "Using Built-In Self-Test and Adaptive Recovery for Defect Tolerance in Molecular Electronics-Based NanoFabrics," *Proc. International Test Conference*, 2005.

[59] R. S. Williams and P. J. Kuekes, "Demultiplexer for a Molecular Wire Crossbar Network," *US Patent Number 6,256,767*, 2001.

[60] J. Xiong, V. Zolotov, and L. He, "Robust Extraction of Spatial Correlation," *Proc. International Symposium on Physical Design*, 2006, pp. 2-9.

[61] J. Zhang, N. P. Patil, A. Hazeghi and S. Mitra, "Carbon Nanotube Circuits in the Presence of Carbon Nanotube Density Variations," *Proc. Design Automation Conference*, 2009.

[62] J. Zhang, N. P. Patil and S. Mitra, "Design Guidelines for Metallic-Carbon-Nanotube-Tolerant Digital Logic Circuits, *Proc. Design Automation and Test in Europe*, 2008, pp. 1009-1014.

[63] W. Zhang, N. K. Jha and L. Shang, "NATURE: A Hybrid Nanotube/CMOS Dynamically Reconfigurable Architecture," *Proc. Design Automation Conference*, 2006, pp. 711-716.

[64] G. Zhang, P. Qi, X. Wang, Y. Lu, X. Li, R. Tu, S. Bangsaruntip, D. Mann, L. Zhang and H. Dai, "Selective Etching of Metallic Carbon Nanotubes by Gas-Phase Reaction," *Science*, 314, 2006, pp. 974-977.

20

A New Technique of Interconnect Effects Equalization by using Negative Group Delay Active Circuits

Blaise Ravelo, André Pérennec and Marc Le Roy
UEB, University of Brest
Lab-STICC, UMR CNRS 3192,
France

1. Introduction

During the last two decades, technological progresses in VLSI process have brought an outstanding development of information technology equipments and thus a great increase in the use of communication services all over the world. As reported by both the International Technology Roadmap for Semiconductors (ITRS) and the Overall Roadmap Technology Characteristics (ORTC), the exponential reduction of the feature size of electronic chips according to Moore's law (Moore, 1965) still occurs together with the exponential increase in time of the number of transistor per unit area. Combined to this shrinking of feature sizes, the on-chip clock frequency increases continually and should exceed 10 GHz in 2010. These ceaseless trends in VLSI circuits have led to more and more complex interconnect systems, and thus, the implementation of metal multi-layers for intra-chip interconnects has become a must. In the mid-1980's, the devices were, thus, composed of one or two layers of aluminium; in 2011, according to ITRS prediction, chips will consist of more than ten layers of copper.
Under these conditions, owing to the higher operation speeds, the interconnect propagation delay becomes more and more significant and dominates considerably the logic propagation delay (Deutsch, 1990; Rabay, 1996). Because of the sensitivities of parametric variations, clock and data flow may not be synchronized (Friedman, 1995). This explains why interconnections are so important in the determination of VLSI system performances. Besides, simplified models are worth being considered in order to reduce the complexity of any study on interconnects. Since the beginning of the 1980s the modelling of propagation delay in order to estimate the delay of an interconnect line driven by a CMOS gate has been the subject of numerous papers (Sakurai, 1983 and 1993; Deng & Shiau, 1990). The simplest and most used model of this delay was proposed by Elmore in 1948; it relies on the use of only an RC-line model. Nevertheless, due to the elevation of system data rates, this model tends to be insufficiently accurate. Therefore, more accurate models that sometimes take into account the inductive effect (Wyatt, 1987; Ismail et al., 2000) have been proposed.

To solve the problem of clock skew and propagation delays, a technique of signal integrity enhancement based on repeater insertion was proposed by different authors (Adler and Friedman, 1998; Ismail & Friedman, 2000). But, when the signals are significantly attenuated, such a solution may be unable to conserve the data duration and thus, inefficient. These considerations drove us to recently propose a new technique for interconnect-effect equalization (Ravelo, Perennec & Le Roy, 2007a, 2008a, 2009 and 2009) through use of negative group delay (NGD) active circuits. As shown in Fig. 1, it consists merely in cascading these NGD circuits at the end of the interconnect line. The possibility of signal recovery with a reduction of signal rise/fall-, settling- and propagation-delays was theoretically demonstrated and evidenced through simulations in (Ravelo et al., 2007a, 2008a) and confirmed by experiments in (Ravelo et al., 2009).

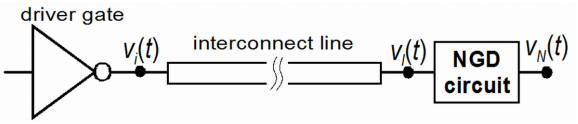


Fig. 1. Interconnect line driven by a logic gate ended by an NGD circuit.

In fact, evidences of the NGD phenomenon have been provided through theoretical demonstrations and experiments with passive-electronic devices (Lucyszyn et al., 1993; Eleftheriades et al., 2003; Siddiqui et al., 2004 and 2005) and active ones (Solli & Chiao, 2002; Kitano et al. 2003; Nakanishi et al., 2002; Munday & Henderson, 2004). As described in several physics domains (Wang et al., 2000; Dogariu et al., 2001; Solli & Chiao, 2002), in the case of a smoothed signal propagating in a device/material that generates NGD, the, the peak of the output signal and its front edge are both in time advance compared to the input ones. Then, confirmations that this counterintuitive phenomenon is not physically at odds with the causality principle have been provided (Wang et al., 2000; Nakanishi et al., 2002). A literature review shows that the first circuits that exhibited NGD at microwave wavelengths displayed also significant losses, whereas the baseband-operating ones were intrinsically limited to low frequencies. To cope with these issues, we, recently, reported on the design, test and validation through simulations and experiments of a new and totally integrable topology of NGD active circuit (Ravelo et al., 2007b, 2007c and 2008b); this topology relies on the use of a FET and showed its ability to compensate for losses at microwave frequencies over broad bandwidth. Transposition of this NGD topology to baseband frequencies allowed us to develop new structures that demonstrated their ability to simultaneously generate an NGD and gain for broad and baseband signals (Ravelo et al, 2008). Then, the idea put forward by Solly and Chiao (Solly, 2002) to compensate degradations introduced by passive systems such as interconnect lines by using NGD devices became possible.

As a continuation of these investigations, this chapter deals with further developments of this technique. Section 2 gives insight into the way this technique works, and briefly explains the role of the NGD circuit. The theory of interconnect modelling and the definition of the propagation delay are both recalled in Section 3. Analytical approach and experimental validations of RC-model equalization are presented in Section 4. The feasibility of the proposed technique, when the inductive effects are taken into account, is dealt in Section 5. In Section 6, a completely original and fully-integrable topology is proposed by

getting rid off inductance to cope with their implementation issue. Thus, the results of simulations, which provided a very good validation of the performances expected from theory, are analysed and discussed. A summary of this chapter is given in the last Section together with proposals about possible future developments.

2. The basic principle of the proposed compensation technique

Figure 2 illustrates the general case of the distortion undergone by a numerical signal degraded by the interconnect circuitry as introduced in Fig. 1. Compared to the input signal, $v_i(t)$, the degradation of the output one, $v_l(t)$, can be assessed from the signal attenuation as well as the rise-, fall- and settling-times and the 50% propagation delay. It is worth recalling that this last parameter, denoted here by $T_{p50\%}$ or merely T_p, is defined as the time needed by the output signal, v_l, to reach 50% of the unit-step input of amplitude, V_M, here. The delay required to reach 50% of the logic swing is traditionally referred as the delay time.

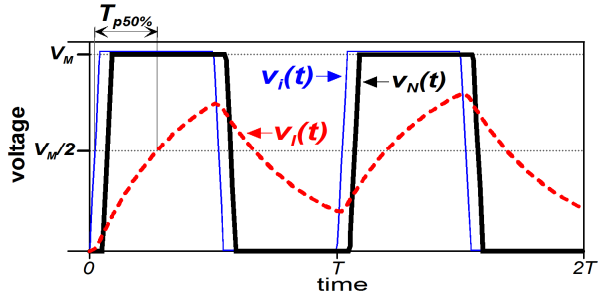


Fig. 2. Time-domain responses of the ideal system shown in Fig. 1 for a periodical input voltage, $v_i(t)$.

In frequency domain, the degradation between the input, v_i and the output, v_l corresponds to a transfer function denoted, $G_l(s)$ whose gain magnitude and group delay usually verify the following inequalities:

$$\left|G_l(j\omega)\right| < 1 \text{ and } \tau_l(\omega) = -\partial \angle G_l(j\omega) / \partial\omega > 0\,. \tag{1}$$

The output Laplace transform of this interconnect line can be written as:

$$v_l(s) = G_l(s)\,.\,v_i(s). \tag{2}$$

As shown by Figs. 1 and 2, this study was aimed at finding a relevant configuration or circuit able to provide a compensated output, v_N, (black thick curve) as close as possible to the input signal, v_i, (red dashed curve). It means that the following mathematical approximation can be made:

$$v_N(t) \approx v_i(t). \tag{3}$$

In theory, for well-matched circuits, the transfer system to be found, $G_x(s)$, must be associated to $G_l(s)$ so that equation (4) is verified:

$$v_N(s) = G_l(s).G_x(s).v_i(s)\,. \tag{4}$$

According to the circuit and system theory, through use of equations (3) and (4), one gets the adequate transfer function:

$$G_l(s).G_x(s).v_i(s) \approx v_i(s) \quad \Rightarrow \quad G_x(s) \approx 1/G_p(s)\,. \tag{5}$$

Consequently, in the frequency domain, the system gain and group delay must be such that:

$$G_x(j\omega)\big|_{dB} = -G_l(j\omega)\big|_{dB}\,, \tag{6}$$

$$\tau_x(\omega) = -\tau_l(\omega)\,. \tag{7}$$

So, on condition to take into account the condition expressed in equation (1), the gain and the group delay must be respectively such that $|G_x(j\omega)|_{dB} > 0$ and $\tau_x(\omega) < 0$. Technically, these conditions require the cascade of a system able to simultaneously exhibit Gain and an NGD in baseband. As described by the block diagram of Fig. 3, the whole cascaded system is characterized by its transfer function G(s) where G_l is the interconnect transfer function and $G_x = G_{NGD}$; τ is the whole group delay.

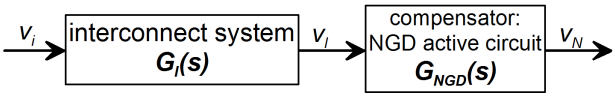


Fig. 3. Block diagram: interconnect passive system, $G_l(s)$ cascaded by the active NGD circuit, $G_{NGD}(s)$: $G(s) = G_l(s).G_{NGD}(s)$.

This process constitutes, then, a technological solution of equalization technique. In this case, the gain and group delay generated by the compensation system should be respectively the reverse and the opposite of those of the interconnect circuitry as depicted in Fig. 4. The principle of interconnect loss compensation and group delay reduction is illustrated in Fig. 4. At this stage, it is worth noting that the compensator must contain a system able to exhibit a group delay negative at base band frequencies not only with an opposite value of the interconnect one but also over the corresponding frequency band. A similar remark can be made about the gain compensation. The need for active circuits drove us to propose, here, a topology based on the use of a FET, chosen because of its biasing simplicity, the different models available (simple for analytical theory and more accurate ones for final simulations), its faculty to operate at tens of GHz as well as its easy integration.

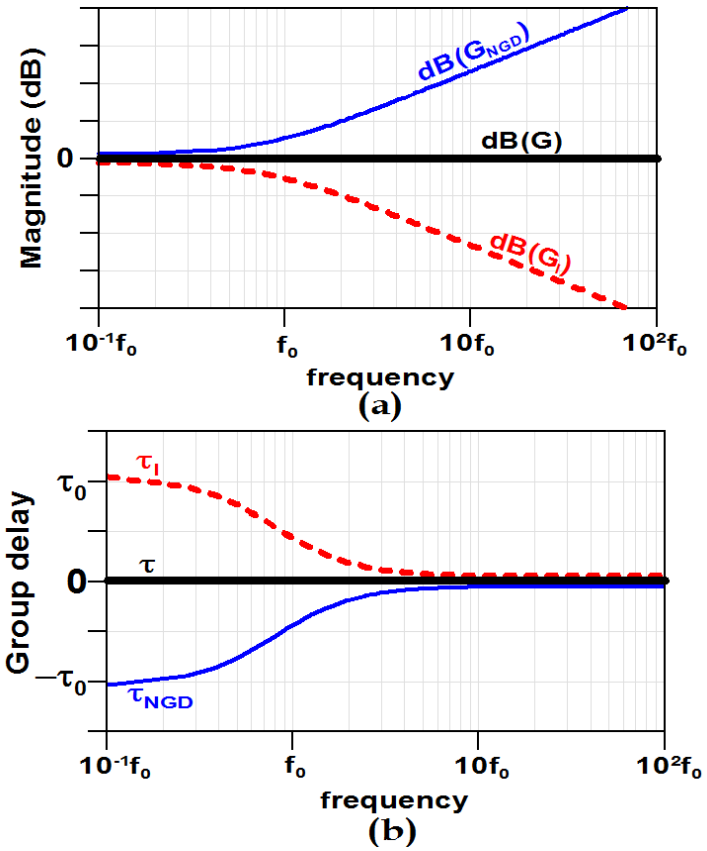


Fig. 4. Illustration of the compensation principle: frequency responses of the magnitude in dB (a) and group delay (b).

In conclusion, the NGD circuit consists in the achievement of a group delay and attenuation compensatory function. Fig. 4 depicts the compensation principle by considering the general frequency behaviour of interconnects or transmission lines. So, prior to conducting a feasibility study of this technique with concrete systems, let us briefly recall the theory on commonly used models of interconnects, i.e. RC- and RLC-circuits, and on propagation delay assessments.

3. Theory on the interconnect modelling and the propagation delay approximation

In most of microelectronic device interconnect models, the conductance effect can be neglected compared to the per-unit length resistance, R_l, the inductance, L_l, and the capacitance, C_l. Therefore, let us consider the interconnect structure presented in Fig. 5: it consists of an RLC-line model of length, d, driven by a gate with an output resistance, R_s. As previously mentioned, v_l is the circuit output voltage and v_i, the input one.

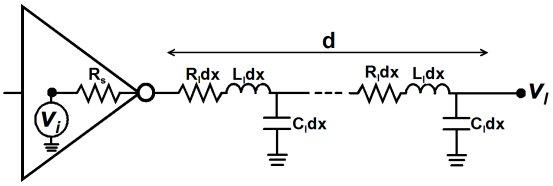


Fig. 5. A gate with output resistance, R_s, driving a RLC-model interconnect.

For the time-domain analysis of the structure under study, the input voltage, v_i, is assigned as a Heaviside unit step function, $\Gamma(t)$, of amplitude, V_M:

$$v_i(t)=V_M\Gamma(t). \tag{8}$$

As defined previously, the 50% propagation delay, T_p , is defined as the root of equation (9):

$$v_l(T_p) = V_M/2. \tag{9}$$

Before the calculation of this propagation delay, it is worth focusing on the system transfer function, which is defined as $G_l(s) = V_l(s)/V_i(s)$. It was established (Ajoy et al., 2004) that, according to the configuration of Fig. 5, this quantity is expressed as:

$$G_l(s) = \frac{1}{(1+sR_s)\cosh(\gamma d)+(R_s/Z_c)\sinh(\gamma d)}, \tag{10}$$

where

$$Z_c = \sqrt{(R_l + L_l s)/C_l s}, \tag{11}$$

and

$$\gamma = \sqrt{(R_l + L_l s)C_l s}, \tag{12}$$

are, respectively, the characteristic impedance and the propagation constant of the line. This transfer function is analysed through use of polynomial expansion as done for classical linear systems. For simplification, let us deal with the normalized transfer function, $g_l(s)$ which is determined by $G_l(s)/G_l(0)$, and that can be expressed by the m-order linear expression:

$$g_l(s) = \frac{1+a_1 s + a_2 s^2 + \ldots + a_n s^n}{1+b_1 s + b_2 s^2 + \ldots + b_m s^m}, \tag{13}$$

where the coefficients, a_i (i = {1,...,n}) and b_j (j = {1,...,m}) are real numbers, and m and n are integers. According to the literature (Elmore, 1948; Wyatt, 1987; Ismail et al., 2000), use of this expression allows one to estimate the 50% propagation delay expressed in equation (9). Among the existing approximation, it is worth recalling that the simplest and the most used in the industrial context is the one proposed by Elmore in 1948. It is based on the first-order consideration of equation (13). Indeed, this estimation of the propagation delay is merely defined by:

$$T_{pElmore} = b_1 - a_1. \tag{14}$$

One should note that this propagation delay is exactly equal to the group delay of the system under consideration at very low frequencies (Vlach et al. 1991). Nevertheless, this

formula has proven to become less and less accurate towards the elevation of the operating frequency. A new formula was proposed by Wyatt (Wyatt 1987): it differs from the previous approach by only the values of the coefficients, a_i and b_i, which are defined by the reciprocal of the dominant pole of the system transfer function:

$$a_1 = \sum_{i=1}^{n} z_i^{-1} , \tag{15}$$

$$b_1 = \sum_{i=1}^{m} p_i^{-1} , \tag{16}$$

where the real numbers, z_i and p_i, are, respectively, the zeros and the poles of $g_l(s)$. It is worth noting that this approach provides an exact expression of T_p in the case of the RC-model ($L_l = 0$).

3.1 Recall on RC-line model

It was established (Sakurai , 1983; Deng & Shiau, 1990) that the first-order approximation of the transfer function in equation (10) leads to the following expression:

$$G_l(s) = 1/\{1 + [(R_s + R_l d) + C_l d]s\} = G_{rc}(s) . \tag{17}$$

This allows the modelling of the interconnect circuitry presented in Fig. 5 as a simple equivalent circuit (Fig. 6).

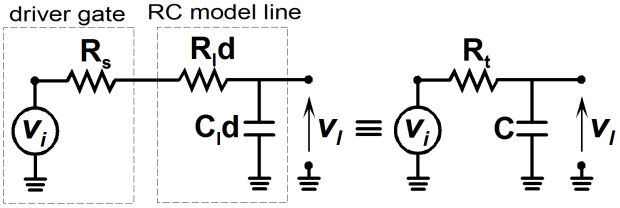


Fig. 6. Simplified representation of the structure shown in Fig. 5 by considering the first-order approximation of the transfer function.

Therefore, the driver gate loaded by the distributed transmission line can be equivalent to a lumped RC-circuit. To make easier the analytical calculation, let us consider the equivalent parameters of the system under study, $R_t = R_s + R_l d$ and $C = C_l d$. So, the Elmore propagation delay of this well-known circuit is given by:

$$T_{pRC} = R_t C . \tag{18}$$

But, it can be shown by calculation through the unit step response that the exact value of this quantity is expressed as:

$$T_{pRC} = R_t C \ln(2) . \tag{19}$$

Then, the rise time, denoted t_{rRC}, which is defined as the time needed by the output signal to pass from 10% to 90% of the final output value, is written as:

$$t_{rRC} = R_t C \ln(9) . \tag{20}$$

3.2 Summary on RLC-line theory

As previously mentioned, nowadays, interconnect line modelling requires, in most cases, to thoroughly consider the inductance effect. This parameter is usually taken into account through use of a second-order system, such as an RLC-network with the following canonical transfer function:

$$g_1(s) = \omega_n^2 / (s^2 + 2\zeta\omega_n s + \omega_n^2) . \tag{21}$$

where

$$\omega_n = 1/\sqrt{L_t C} , \tag{22}$$

and

$$\zeta = (R_t/2)\sqrt{C/L_t} , \tag{23}$$

are, respectively, the undamped natural angular frequency and the damping ratio. From this second-order transfer function, an accurate propagation delay, T_{pd}, was established by Ismail & Friedman (2000):

$$T_p = (e^{-2.9\zeta^{1.35}} + 1.48\zeta)/\omega_n . \tag{24}$$

To reduce this propagation delay, and as first envisaged by Solli and Chiao (Solli et al, 2002), it could be worth cascading the interconnect line with an NGD active circuit. But a preliminary to this proposal is a detailed study of the resulting device in order to get a confirmation of the compensation principle efficacy. This will be the focus of the next Section.

4. Compensation of a first-order interconnect model (RC-circuit)

To compensate for interconnect spurious effects, especially losses and time delay, we recently developed and tested a new equalization technique based on the use of an NGD active circuit (Ravelo et al., 2007a, 2008a and 2009). In this section, we will briefly recall the theoretical fundamentals and validate the technique through experiments carried out in frequency- and time-domains in the case where a first-order circuit, i.e. an RC-circuit, is used to model the interconnect effects.

4.1 Theory

As explained above, the equalization principle consists in ending the circuit to be compensated, here an RC one, with an NGD active cell as depicted in Fig. 7. To simplify the analytical approaches, the FET is modelled by a controlled voltage current source with a transconductance, g_m, and the drain source resistance, R_{ds}.

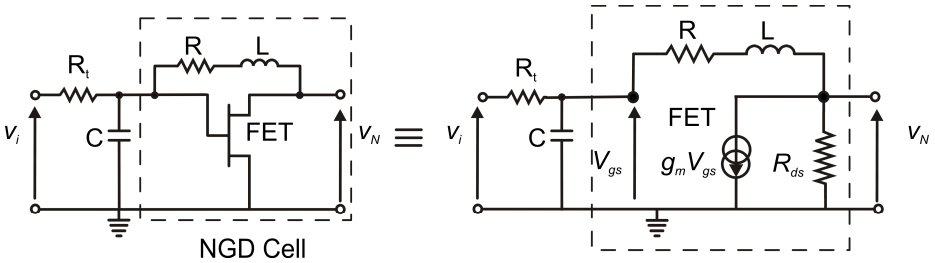


Fig. 7. RC-NGD circuit: RC-circuit cascaded with a basic cell of the NGD active circuit (FET in feedback with an RL series network) and the corresponding equivalent circuit.

In a first step, let us recall (Ravelo et al., 2007a, 2008a, 2008b, 2009) the transfer function and the group delay expressions of the NGD cell alone (in the dash box in Fig. 7) at low frequencies:

$$G_{NGD}(0) = \frac{(1-g_m R)R_{ds}}{R+R_{ds}}, \tag{25}$$

$$\tau_{NGD}(0) = \frac{L(1+g_m R_{ds})}{(R+R_{ds})(1-g_m R)}. \tag{26}$$

On condition that:

$$1-g_m R < 0, \tag{27}$$

the NGD cell exhibits a negative group delay ($\tau_{NGD}(\omega \to 0) < 0$) at very low frequencies.

Then, the transfer function and the gain at low frequencies of the whole RC-NGD circuit presented in Fig. 7 are, respectively, expressed as:

$$G(s) = \frac{R_{ds}(1-g_m R) - g_m R_{ds} L s}{R+R_{ds}+R_t(1+g_m R_{ds}) + \left[R_t C(R+R_{ds})+L\right]s + R_t C L s^2}, \tag{28}$$

$$G(0) = \frac{R_{ds}(1-g_m R)}{R+R_{ds}+R_t(1+g_m R_{ds})}. \tag{29}$$

One should note that, because of the unmatched effect between the RC- and NGD-circuits, the transfer function, $G(s)$, is not equal to $G_{rc}(s).G_{NGD}(s)$ (Ravelo et al., 2009). The same remark applies to the group delays:

$$\left|G(j\omega)\right| \neq \left|G_{rc}(j\omega)\right|.\left|G_{NGD}(j\omega)\right|, \tag{30}$$

$$\tau(\omega) \neq \tau_{rc}(\omega) + \tau_{NGD}(\omega). \tag{31}$$

(i) Demonstration of the 50% propagation delay reduction:
From equations (18) and (28), the Elmore propagation delay (or the group delay at low frequencies) of the compensated RC-NGD circuit can be expressed as follows (Ravelo et al., 2008a):

$$T_p = \frac{[1 + g_m(R_t + R_{ds})g_m^2 R_t R_{ds}]L + T_{pRC}(R + R_{ds})(1 - g_m R)}{(R + R_t + R_{ds} + g_m R_t R_{ds})(1 - g_m R_{ds})}. \quad (32)$$

So, the delay is reduced ($T_p < T_{pRC}$,) on condition that:

$$L > (1 - g_m R)R_t^2 C / (1 + g_m R). \quad (33)$$

But this inequality is automatically verified if the condition expressed in equation (27) is true, i.e. if the active circuit generates NGD. According to this study, the optimum values of the NGD circuit can be synthesised as a function of the RC-values.

(ii) Synthesis of an NGD cell according to the RC-model values

As the aim of the equalization principle is the generation of an output signal equal or close to the input one, the transfer function magnitude and group delay should be, respectively almost equal to unity and null. For simplification purpose, let us apply these conditions at very low frequencies:

$$|G(0)| \approx 1 \text{ and } \tau(0) \approx 0. \quad (34)$$

From equations (31) and inversion of (29) and (32), one gets the synthesis relations expressed in equations (35) and (36):

$$R = [2R_{ds} + (g_m R_{ds} + 1)R_t]/(g_m R_{ds} - 1), \quad (35)$$

$$L = \frac{(g_m R - 1)(R + R_{ds})R_t C}{g_m^2 R_{ds} R_t + g_m(R_{ds} + R_t) + 1}. \quad (36)$$

These expressions are meaningful when the losses displayed by the interconnect are less than the maximal gain magnitude of an NGD cell, G_{max} :

$$G_{\max} = g_m R_{ds}. \quad (37)$$

Otherwise, several stages of NGD cells are needed to generate a whole gain of about unity; then, as reported in (Ravelo et al., 2008a), it is advised to use equation (38) for the calculation of the optimal number of cells, n:

$$n \approx 1 + \text{int}[-\ln(1 - e^{-T/(R_t C)})/\ln(G_{\max})]. \quad (38)$$

T is the time duration of the considered input square pulse. For a real x, the value of the greatest integer given by the function, $int(x)$, is equal to or lower than x. It is worth underlining that the implementation of, at least, two or an even number of transistors is preferable when the application under study requires avoiding signal inversion.

To validate these theoretical predictions, proof-of-concept (POC) circuits were designed, fabricated and measured in frequency- and time-domains as explained hereafter.

4.2 Experimental validations

The starting values of the NGD circuit come from the synthesis relations knowing the characteristics of the chosen FET and the RC-model values. Then, the most accurate available models were used to run simulations of the POC circuits on *ADS™* simulator software from *Agilent™* ; a sensitivity analysis was also performed. The different POC circuits presented in this section were implemented in hybrid planar technology (association of microstrip and surface mount chip/package components).

(i) Design Process

This demonstrator was aimed at illustrating visually the recovery of the degraded signal and the delay reduction. The circuits were designed and implemented separately; they consisted of an RC-circuit, an NGD-one and the combination of both into an RC-NGD circuit (Fig. 8(b)). To perform a broadband biasing and to avoid the eventual disruptions caused by the bias network at low frequencies, the FET was biased through an active-load technique.

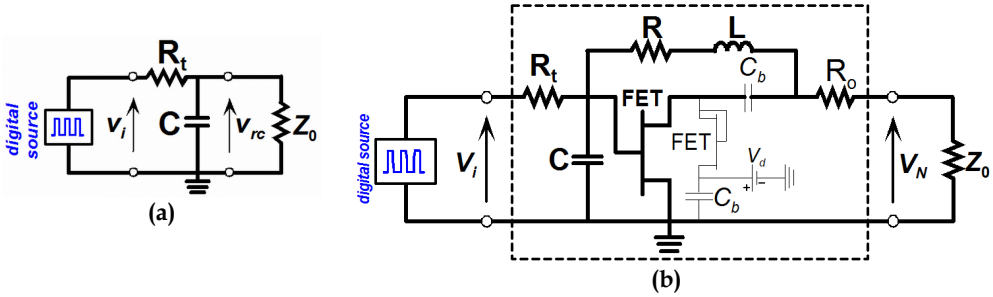


Fig. 8. Schematics of the (a) RC- and (b) RC-NGD-circuits (including the biasing network in thin lines) using a PHEMT FET (ATF-34143, V_{gs} = 0 V, V_d = 3 V, I_d = 110 mA), for R_t = 33 Ω, C = 680 pF, R = 56 Ω, R_o = 10 Ω, L = 220 nH, C_b = 100 nF and Z_0 is the output reference load.

The FET parameters required for the synthesis equations were extracted from the non-linear model and found to be g_m = 226 mS and R_{ds} = 27 Ω. For the RC circuit, the RL values of the NGD circuit were synthesised from equations (33) and (34). Then, accurate frequency responses were obtained through combination of circuit simulations (lumped components and non linear FET model) with electromagnetic co-simulations of the distributed parts by Momentum Software (from *Agilent™*). The values of the available lumped components were used in a final slight optimisation procedure. The layout of the hybrid planar circuit shown in Fig. 9 was printed on an 800-µm-thick FR4 substrate of relative permittivity, ε_r = 4.3; then the surface-mount chip passive components were set onto the substrate.

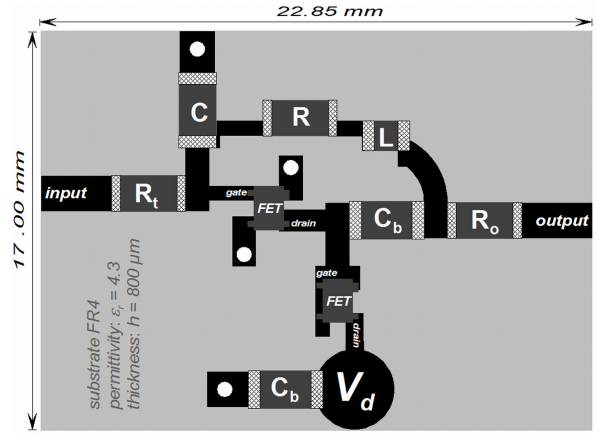


Fig. 9. Layout of the implemented RC-NGD circuit.

(ii) Simulation Results and Sensitivity Analysis

The simulations of the circuit pictured in Fig. 9 and of the RC and NGD ones were independently run in frequency- and time-domains under the specifications and conditions mentioned above. A sensitivity Monte-Carlo analysis over 10 trials was carried out on using tolerance values of ± 5% around their nominal values for the NGD elements (R = 56 Ω and L = 220 nH). All the results are presented in Figs. 10 and 11.

- ***Frequency-domain analysis***: Fig. 10(a) shows that the magnitude $|G(f)|$ of the whole RC-NGD circuit is kept at about 0 dB up to 40 MHz, and the corresponding group delay $|\tau(f)|$ is, as expected, strongly decreased from DC to 15 MHz compared to that of the RC circuit. Moreover, the frequency responses are only slightly sensitive to lumped-component tolerance.

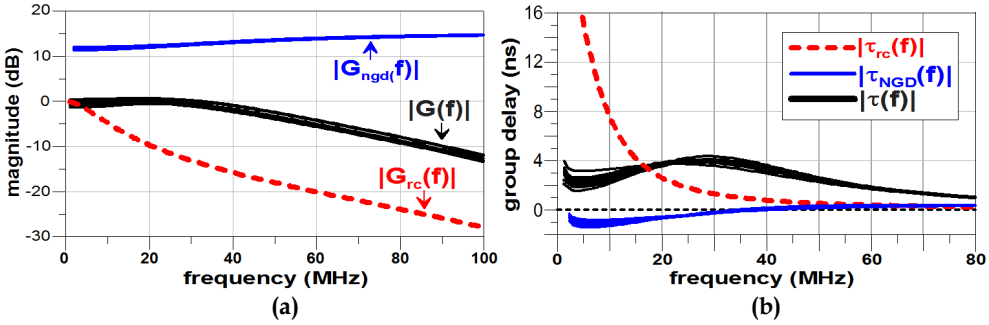


Fig. 10. (a) Magnitude and (b) group delay frequency responses issued from simulations with a Monte-Carlo analysis of the NGD circuit elements R and L (± 5%).

- ***Time-domain analysis***: It is worth underlining that, at first, the signal was measured at the output of a *Rhode & Schwarz Signal Generator SMJ 100A* at the highest available rate, i.e. 25 Msym/s, and then further used in simulations as the input signal. Then, excitation of the

simulated RC- and RC-NGD-circuits with this input square wave pulse (magnitude, V_M = 1 V) led to the time-domain simulation results presented in Fig. 11, where the dotted curve indicates the degradation induced by the RC-circuit alone; moreover, the signal recovery is evidenced by the thick black curve. It clearly appears that the compensation is significant, but incomplete. Because of the drain-source current inversion, the output signal is reversed compared to the input voltage. This is why the plot presented in Fig. 11 is that of $-V_N$.

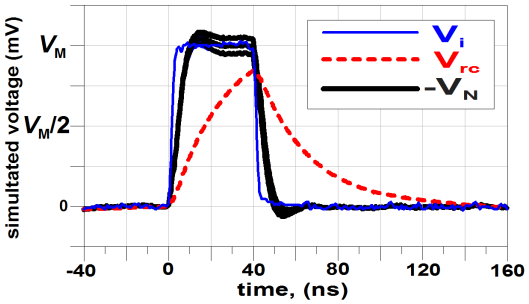


Fig. 11. Results of time-domain simulations (with a Monte Carlo analysis of the NGD circuit elements, R and L (± 5%)) for an input square wave pulse (V_M = 1 V with a 40 ns data duration).

It is worth noting that the 50% propagation delay is shortened from 16.5 ns (for the RC circuit) to about 3 ns (for the RC-NGD circuit), i.e. a relative reduction of, at least, 81%. A Monte-Carlo sensitivity analysis with a ± 5%-variation around the *R* and *L* nominal values showed nearly no change of the RC-NGD circuit output signal (including the front and leading edges), but the final value is somewhat slightly affected.

(iii) Measured results and discussions:
To check for the validity of the results of theoretical analysis and simulations, three circuits were fabricated and tested in both frequency- and time-domains.

- Frequency-domain results: These measurements were made with a vector network analyzer (*Rhode & Schwarz ZVRE 9kHz - 4GHz*), which provides the scattering matrix (S-parameters) of the devices under test. Then, the transfer function is calculated through use of the classical passage relationships in order to get the magnitude and the group delay (calculated from the transfer function phase response). The fabricated devices consisted of the single RC-circuit, the NGD one and a third one, denoted RC-NGD circuit, which combined both functions in order to avoid any connector mismatch liable to occur if the first two ones were simply cascaded. Figure 12(a) shows that the magnitude and group delay responses of the three circuits are in good agreement with the simulation results presented in Figs. 10. As expected from equations (30) and (31), the total magnitude and group delay values are different from those issued from the addition of the individual ones because of the existence of mismatch between the RC and NGD parts ($G_{RCNGD} \neq G_{RC}.G_{NGD}$). The measured magnitude, $|G(f)|_{dB}$, is close to 0 up to 40 MHz and gets down to -10 dB at 80 MHz.

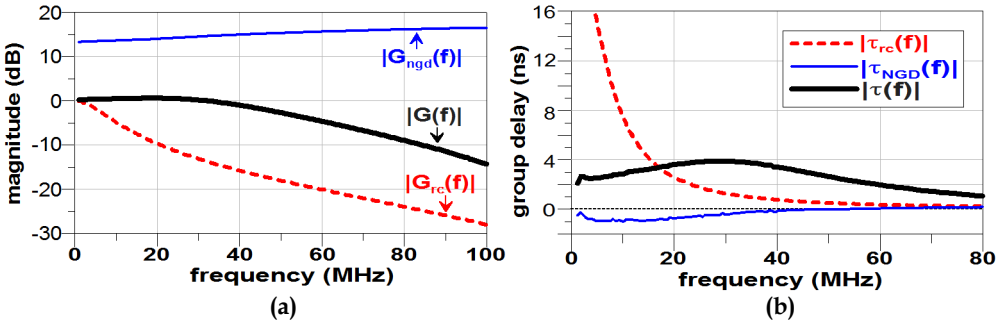


Fig. 12. Measurements of (a) magnitude and (b) group delay produced by the RC-, NGD- and RC-NGD-circuits.

One should note that, though the RC-NGD circuit group delay, $\tau(f)$, is not fully cancelled (Fig. 12(b)), it is strongly reduced below 20 MHz thanks to the NGD circuit; moreover, it is kept below 4 ns up to 80 MHz. In theory, a higher absolute value of NGD could be obtained, but a compromise between NGD value, NGD bandwidth and gain flatness has to be found to minimize the overshoot or the ripple in time-domain

- ***Time-domain experimental results***: Throughout the time-domain simulations and measurements, the circuit load, Z_0, is set at a high impedance value (Z_0 = 1 MΩ).

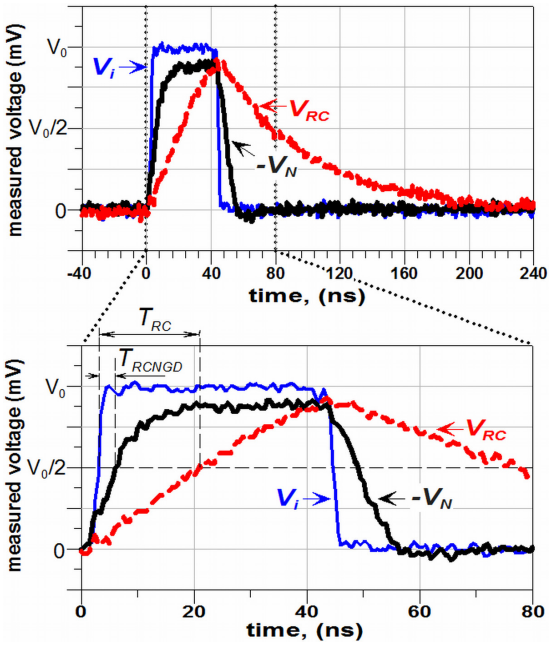


Fig. 13. Time-domain responses with an input square pulse (25-Msym/s rate, 2 ns rise- and fall-times) and zoom on twice the symbol duration.

A square wave input pulse of amplitude $V_M = 1$ V was delivered at a rate of 25-Msym/s (corresponding to a 40-ns data duration) by the baseband data output of an *R&S SMJ 100A* vector signal generator and recorded on a *2 Gs/s* LeCroy digital oscilloscope. The input pulse, V_i, was thus monitored as well as the output ones of the RC- and RC-NGD-circuits, V_{RC} and $-V_N$, prior to their resynchronization through use of the same reference signal. Figure 13 shows that the output leading edge provided by the RC-circuit is degraded and characterised by $t_{rRC} \approx 35$ ns as rise time and a 50% propagation delay of $T_{pRC} \approx 18.50$ ns. Compared to V_{RC}, the output waveform, $-V_N$, is reshaped and less distorted. Hence, the NGD circuit compensation allowed a reduction of both parameters down to $t_r \approx 10$ ns and $T_p \approx 2.50$ ns, that is relative reductions of 71.4% ($1-t_r/t_{rRC}$) and 86.5% ($1-T_p/T_{pRC}$), respectively. As shown in Fig. 13, the trailing edge is also strongly improved. This point is worth being noted in the case of an enhancement of the data rate.
The proposed interconnect equalization technique was validated through modelling of the interconnect line degradation by an RC-circuit, which is the most current model in use. However, in order to supplement this study, the inductive effects of interconnect lines will be taken into account in the next Section.

5. Equalization under conditions of interconnect line inductive effects

As underlined at the beginning of this chapter, the ceaseless increase of operating frequencies, the inductive effects can no longer be neglected in the modelling of interconnect systems. In this study, the line is modelled by the RLC network shown in Fig. 14(a) and driven by a logic gate with R_s as output resistance. In order to check whether the proposed compensation technique still works, let us consider the whole circuit presented in Fig. 14(b): it consists of the interconnect model cascaded by the compensation circuit, which is composed of a FET fedback by an RL-series network. To simplify the theoretical study, the FET is replaced by its low frequency equivalent-circuit model as introduced in Fig. 5. The terms, $R_t = R_s + R_l d$, $L_t = L_l d$ and $C = C_l d$, introduced in the previous study of the RC-network are still used.

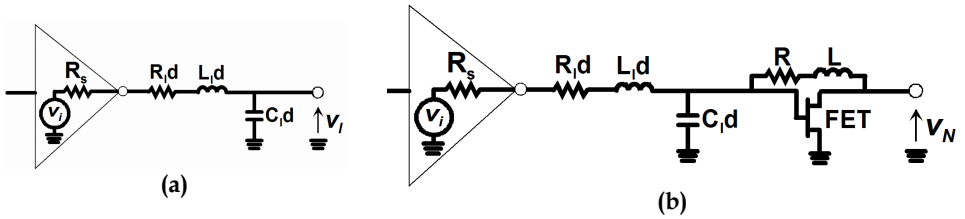


Fig. 14. (a) line model: RLC network driven by a logic gate with output resistance, R_s; (b) the whole circuit composed of the line model compensated by an NGD cell.

5.1 Theory

According to the procedure used in the previous section, the transfer function of the whole system is:

$$G(s) = R_{ds}[1 - g_m(R + Ls)]/(\alpha_0 + \alpha_1 s + \alpha_2 s^2 + \alpha_3 s^3), \quad (39)$$

where

$$\alpha_0 = R + R_t + R_{ds}(1 + g_m R_t), \quad (40)$$

$$\alpha_1 = R_t C(R + R_{ds}) + g_m R_{ds} L_t + L + L_t, \quad (41)$$

$$\alpha_2 = [LR_t + L_t(R + R_{ds})]C, \quad (42)$$

$$\alpha_3 = LL_t C. \quad (43)$$

Then, at very low frequencies ($\omega \rightarrow 0$), the corresponding gain, G(0), and the Elmore propagation delay, T_p, are respectively expressed as:

$$G(0) = [R_{ds}(1 - g_m R)]/[R + R_t + R_{ds}(1 + g_m R_t)], \quad (42)$$

$$T_p = \frac{(g_m R - 1)[R_t C(R + R_{ds}) + L_t(1 + g_m R)] - (1 + g_m R_{ds})(1 + g_m R_t)}{[R + R_t(1 + g_m R_{ds}) + R_{ds}](g_m R - 1)}. \quad (43)$$

Application of the equalization objectives expressed in (32) to equations (42) and (43) allows one to extract the following synthesis relations:

$$R = [2R_{ds} + (g_m R_{ds} + 1)R_t]/(g_m R_{ds} - 1), \quad (44)$$

$$L = (g_m R - 1)\left[(R + R_{ds})CR_t + (1 + g_m R_{ds})L_t\right]/[g_m^2 R_{ds} R_t + g_m(R_{ds} + R_t) + 1]. \quad (45)$$

To evidence the relevance of this RLC effect equalization, simulations under realistic conditions were run.

5.2 Validations by simulations

In order to solve the problem of the output voltage sign and to take into account inductive effects, a two-stage NGD circuit was simulated through use of *ADS™* for two types of interconnect lines models: i) a model with lumped RLC elements and an input signal of 1 ns in data duration, and ii) an RLC distributed line and an input signal of 5-ns data duration. Moreover, using two NGD cells allows widening the NGD frequency bandwidth (Ravelo et al, 2007c) and thus, input signals with higher data rates and smaller rise-/fall-times can be considered.

(i) Compensation of a lumped RLC-circuit for an input signal with 1 ns-data duration

Fig. 15 depicts the whole compensated RLC-NGD circuit under study. It consists in a lumped RLC-circuit ended with a two-stage NGD active circuit. The FETs used by the latter (EC-2612 with g_m = 98.14 mS and R_{ds} = 116.8 Ω) are from *Mimix Broadband™*.

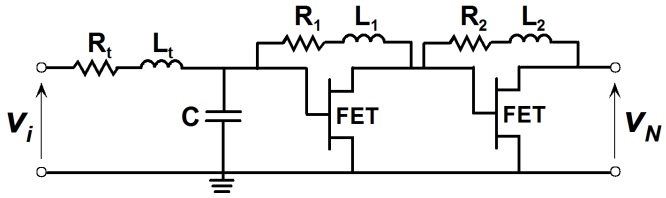


Fig. 15. Two-stage NGD active circuit compensating the RLC network (R_t = 100 Ω, L_t = 6 nH, C = 4.3 pF, R_1 = 86 Ω, R_2 = 89 Ω, L_1 = 5.2 nH, L_2 = 2.6 nH and FET/EC-2612).

One should note that the time- and frequency-domain results presented here were obtained by using the FET linear model provided by this manufacturer and recorded at the biasing point V_{ds} = 3 V and I_{ds} = 25 mA.

- Frequency-domain results: Figs. 16 (a) and (b) present the magnitudes and the group delays of the RLC- and NGD-circuits separately, and then, both cascaded (noted RLC-NGD), as depicted in Fig. 15. Fig. 16(a) shows that the transfer-function magnitude of the overall circuit (black thick curve) is kept within -4 and 0 dB up to 3 GHz.

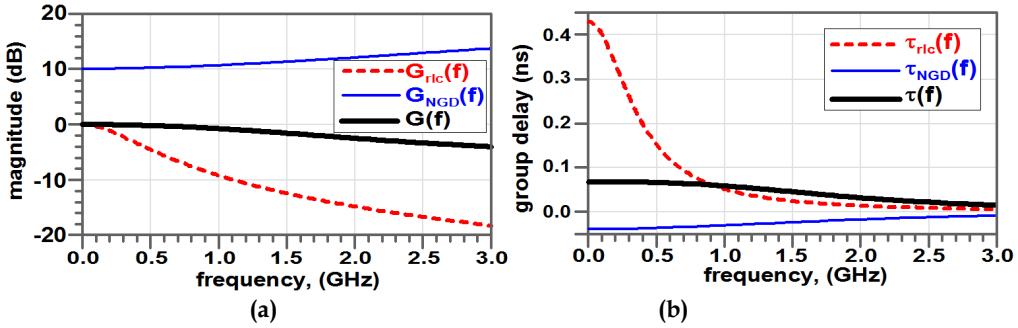


Fig. 16. Simulated frequency responses of the RLC, NGD and RLC-NGD circuits: (a) magnitude and (b) group delay.

Once again, one should note that the value of this whole transfer-function magnitude is different from the sum (in dB) of the individual magnitudes ($G_{RLCNGD} \neq G_{RLC}.G_{NGD}$) because of mismatch between both parts. Figure 16(b) evidences that, thanks to the NGD effect (thin blue curve), the total group delay of the whole circuit (black thick curve) is less than 68 ps. A comparison of the group delays respectively produced by the RLC and the RLC-NGD circuits shows a significant reduction up to about 0.8 GHz with the latter.

- Time-domain results: Figure 17 presents the results of transient simulations run in the case of a 2-ns periodic input signal, V_i, whose rise-/fall-times are about 92 ps (thin red curve). The response at the output of the RLC circuit alone, V_{RLC}, is depicted by the degraded dashed curve, and the corresponding 50% propagation delay, T_{pRLC}, is equal to 304 ps. Further to the insertion of the NGD circuit, the black thick curve representative of V_N, i.e. at the output of the RLC-NGD circuit, shows improvement with a relative reduction of more than 85% of propagation delay since T_p is about 44 ps. Moreover, by comparison to V_i, V_N presents neither attenuation, nor overshoot, and the leading and trailing edges are both improved.

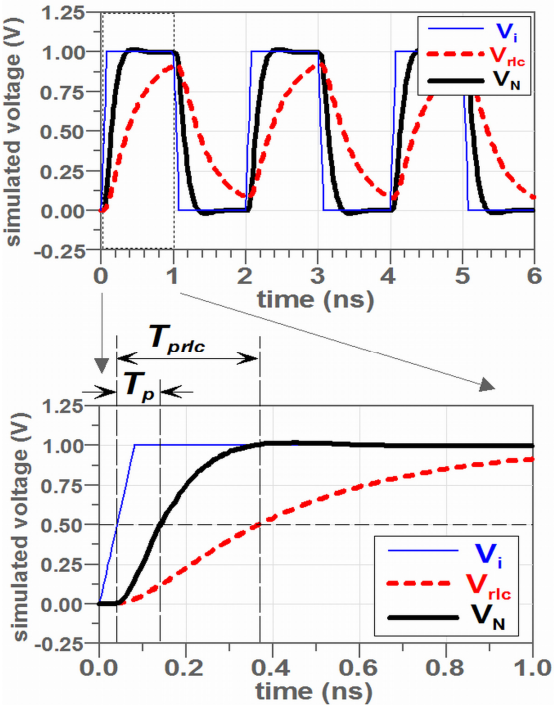


Fig. 17. Time-domain responses produced by simulations of the circuit in Fig. 15 for a 2-ns periodic trapezoidal input, rise-/fall-times, t_r = 92 ps and 50%-duty ratio with zoom on a half-period (bottom).

As the accuracy of the lumped RLC model of interconnect line used in these simulations might be insufficient for certain VLSI configurations, the simulation described in the next paragraph dealt with a distributed RLC-model (Ravelo et al, 2007a).

(ii) Compensation of a distributed RLC-line in the case of an input signal of 5-ns data-duration

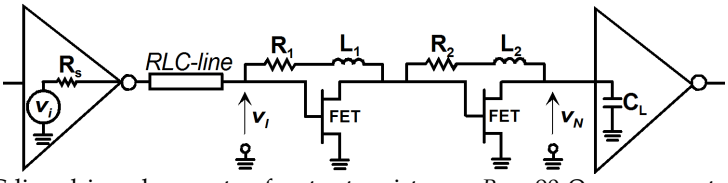


Fig. 18. RLC-line driven by a gate of output resistance, R_s = 90 Ω, compensated by a two-stage NGD circuit (FET/EC-2612, R_1 = 73 Ω, L_1 = 99 nH, R_2 = 102 Ω and L_2 = 17 nH) loaded by another gate of input capacitor C_L = 30 pF.

Fig. 18 shows the circuit under study; the interconnect line is modelled by an RLC distributed circuit. The classical RLC-line is driven by a gate of output resistance, R_s,

compensated by a two-stage NGD circuit loaded with another gate of input capacitor, C_L. The transmission line parameters were taken from ITRS roadmap so that R_l = 76 Ω/cm, L_l = 5.3 nH/cm and C_l = 2.6 pF/cm for a 0.8-cm-long line. For these values, the synthesis relations were used together with an optimization process to get an output, V_N, close to the input, V_i. Finally, the component values for the two NGD cells were: R_1 = 73 Ω, L_1 = 99 nH, R_2 = 102 Ω and L_2 = 17 nH.

- Frequency-domain results: Figs. 19(a) and (b) respectively show the transfer function magnitude and the group delays produced by simulations of the three circuits (RLC-line, NGD- and RLC-NGD-circuits). In Fig. 19(a), the important attenuation displayed by the RLC-line is compensated by the NGD cells so that the total gain is kept at about 0 dB up to about 500 MHz. With respect to the group delay produced by the RLC-line, the one by the RLC-NGD circuit (thick black curve) is strongly reduced as expected thanks to the NGD contribution (thin blue curve). Indeed, at baseband frequencies, this latter provides a minimal NGD value around -0.5 ns.

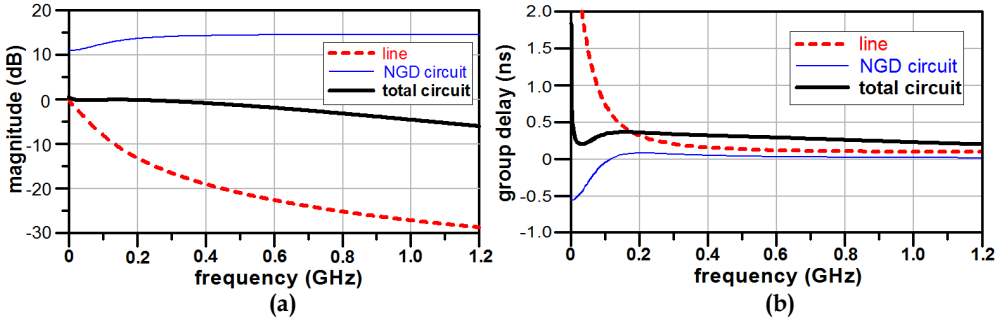


Fig. 19. Frequency responses produced through simulations of the RLC-, NGD-, and RLC-NGD circuits shown in Fig. 18: magnitude (a) and group delay (b).

- Time-domain results: Time-domain simulations were run on using at the input a 10-ns-period signal with 0.60 ns as rise-/fall-times.

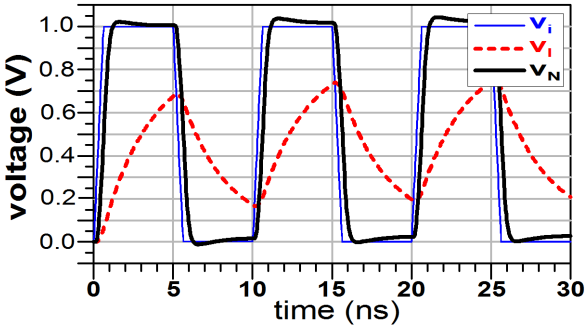


Fig. 20. Time domain output responses of the RLC-line and of the RLC-NGD compensated structure for a trapezoidal input voltage at a 200 Mbit/s rate.

Fig. 20 evidences a marked degradation of the output waveform, V_l, of the RLC-line compared to the input waveform. Once again, the insertion of the NGD active circuit allows great enhancement illustrated by a notable signal regeneration, a reduction of the propagation delay of about 1.60 ns (from 1.86 ns to 0.26 ns) accompanied with an enhancement of the signal leading- and trailing-edges. These results confirm the efficacy and the reliability of this technique to equalize the degradation induced either by lumped or distributed RLC-model of interconnect lines.

6. Improvement of the NGD topology toward an integration process

Application of these innovative cells to the compensation of VLSI interconnect-induced degradation requires compatibility with VLSI integration process. So, here, the main issue is the integration and manufacturing of inductive elements such as the inductance of the FET feedback and the biasing Inductance (choke self with a high value). The latter can be replaced with, for example, an active bias which, moreover, provides a possibility of operating bandwidth enhancement. To get rid of the feedback inductance, let us consider a new topology of NGD active circuit with no inductance. It simply consists of a FET cascaded with a shunt RC-network. To match the cell output to the following stage input, an R_2 resistor is connected in shunt at its end. As described in Fig. 21, this new topology is cascaded at the end of an RC circuit, which models the interconnect line to be compensated.

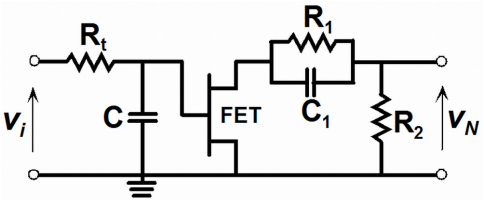


Fig. 21. RC-circuit compensated by a NGD cell without inductance.

Let us focus, at first, on the analytical study of the proposed cell according to the RC circuit values prior providing evidence, through simulations, of the compensation of the RC effects, i.e. signal recovery and delay reduction, by using the proposed NGD topology

6.1 Theory

According to (Ravelo et al, 2008), the transfer function of the RC-NGD device described in Fig. 21 is:

$$G(s) = \frac{-G_{max}R_2(1+R_1C_1s)}{(1+R_tCs)[R_1+R_2+R_{ds}+R_1C_1s(R_2+R_{ds})]}, \tag{48}$$

where G_{max} is the maximal gain value expressed in equation (37). The minus sign is explained by the intrinsic FET model, which entails naturally the inversion of the output voltage direction compared to the input one. Otherwise, with the same approach as in Section 4.1, from this transfer function it can be established that the gain at very low frequencies and the Elmore propagation delay are given by:

$$G(0) = \frac{-G_{max}R_2}{(R_1 + R_2 + R_{ds})}, \tag{49}$$

$$T_p = \frac{[(R_2 + R_{ds})T_{pRC} - R_1C_1(R_t + R_1)]}{(R_1 + R_2 + R_{ds})}. \tag{50}$$

From expression (49), loss compensation ($|G(0)| > 1$) is effective on condition that the following condition between G_{max} and the resistance values be met:

$$G_{max} > 1 + \frac{R_1 + R_{ds}}{R_2}. \tag{51}$$

In addition, from expression (50), it can be found that, whatever the values of the RC- and the NGD-circuit parameters, T_P is always lower than T_{pRC}. In other words, the RC-propagation delay is always reduced in the configuration of Fig. 21. At this stage, it is worth pointing out that equation (50), despite its usefulness, is an approximation of the 50% propagation delay as proposed by Elmore. Indeed, according to Ismail and Friedman, a relative inaccuracy of about 30% is possible. Another limitation appears when the following condition is satisfied:

$$R_tC < \frac{R_1C_1(R_t + R_1)}{R_2 + R_{ds}}. \tag{52}$$

In this case, expression (50) provides a negative value for T_p, which leads to an unrealistic behaviour that would contradict the causality principle. Indeed, calculation of the exact expression for T_p from the root of the equation, $v_N(T_p) = V_M/2$, gives always a positive value for the total propagation delay.
Despite these restrictions, equations (49) and (50) are particularly useful for a first analytical approximation and permit the extraction of the synthesis relations needed to determine the values of the NGD cell components:

$$R_1 = R_2(G_{\max} - 1) - R_{ds}, \tag{53}$$

$$R_2 = (R_1 + R_{ds})/(G_{\max} - 1), \tag{54}$$

and by using the equation, $\tau(0) \approx 0$:

$$C_1 = CR(R_1 + R_2 + R_{ds})/R_1^2. \tag{55}$$

The synthesis relations (53) and (54) are physically realistic under the following conditions:

$$G_{\max} > 1, \tag{56}$$

$$R_2 > R_{ds}/(G_{\max} - 1). \tag{57}$$

6.2 Discussion on the simulation results

The previous synthesis relations were used to design and simulate, with *ADS* software, the circuit presented in Fig. 22(a). The thin lines indicate the DC bias network. The RC-circuit is compensated by a two-stage NGD circuit, and the whole circuit is excited at the rate of 1 Gbit/s by an input signal, V_i, with rise and fall times of about 0.2 ns.

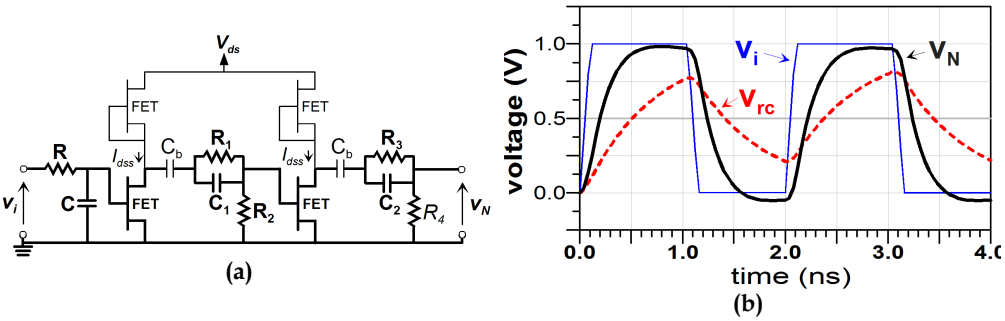


Fig. 22. (a): The whole circuit composed of an RC-model compensated by a two-stage NGD circuit in active biasing; FET (EC-2612, V_d = 3 V, I_{ds} = 30 mA), R = 68 Ω, C = 10 pF, R_1 = 142 Ω, R_2 = 32 Ω, C_1 = C_2 = 10 pF, R_3 = 100 Ω, R_4 = 51 Ω and (b) the corresponding simulation results.

An active load bias with the same FETs (EC-2612) was applied to the circuit. No inductance element is needed in this circuit. Transient simulations run with ideal components gave the results displayed on Fig. 22(b). Once again, the comparison of the RC- and RC-NGD-outputs highlights a signal recovery, characterised by a gain of about 1.85 dB at $t = T/2$ and a reduction of propagation from $T_{pRC} \approx 47$ ps to $T_p \approx 12$ ps or $\Delta T/T_{pRC} \approx 74.4\%$ in relative value. Time-domain measurements are scheduled and will indicate if further improvements are required prior to the integration in a final VLSI device.

7. Conclusion and future works

A novel and innovative technique of interconnect effect equalization in electronic systems was developed through theoretical studies and experimentally validated. It relies on the use of active circuits able to simultaneously generate gain and negative group delay in baseband over broad bandwidth.

The properties of these circuits were used to propose a new compensation approach consisting in an equalization of both the positive group delay and attenuation induced by interconnects by an equivalent negative group delay and gain.

The theory on commonly used circuits to model interconnect effects were briefly recalled. Then, a circuit composed of a first-order interconnect model (i.e. an RC-circuit) cascaded with an NGD cell was theoretically studied in order to determine the conditions required to compensate for both the degraded propagation delay and the attenuation and to express the synthesis relations to be used in the determination of the values of the NGD cell components. This NGD cell simply consists in a FET fedback by an RL series network. Then, for this first modelling of interconnect line, a proof-of-concept circuit implemented in hybrid planar technology was fabricated to demonstrate the efficiency of this technique. The

experimental results in frequency- and time-domains were in very good agreement with simulations and validated the compensation technique in the case of an input signal with a 25 Mbit/s data rate. Indeed, the reductions of the rise time and the 50% propagation delay were 71 and 86%, respectively.
In many VLSI systems and particularly in long wires and/or for high data rates or clocks, the inductive spurious effects can no longer be neglected. So, a more elaborated system composed of an RLC interconnect model compensated with NGD cells was also studied analytically in order to check for the validity and efficacy of the equalization technique and to determine the synthesis relations to be used in further applications. To validate the approach, a first series of simulations was run with an RLC lumped model for an input signal at 1 Gbits/s-rate; then , the model used in the second set of simulations was an RLC distributed line for an input signal at a rate of 200 Mbit/s. These simulations under realistic conditions confirmed the compensation approach with reduction of the propagation delay of the same order as previously. Moreover, as observed with the RC-model, the front and trailing edges both showed great enhancements indicative of a good recovery of the signal integrity.
Finally, to be able to compensate for interconnect effects in VLSI systems, the proposed circuits must be compatible with a VLSI integration process. This requirement drove us to propose improvements of the proposed topology in order to cope with inductance integration and manufacturing prerequisites. So, a topology with no inductance, but with the same behaviour and performances as previously was proposed. A theoretical study provided evidence of its ability to exhibit a negative group delay in baseband together with gain. Then, time-domain simulations of a two-stage NGD device excited by a 1 Gbits/s-rate input signal were run to validate the expected compensation approach and check for the signal recovery.

The implementation of this equalization technique in the case of a VLSI integration process is expected to allow compensation for spurious effects such as delay and attenuation introduced by long inter-chip interconnects in SiP and SoC equipments or by long wires and buses. A preliminary step would be the design and implementation of such a circuit in MMIC technology and especially by using distributed elements. At this stage, even if experimentally the NGD cells were not particularly sensitive to noise contribution, it would be worth comparing this approach and repeater insertion under rough conditions, i.e. long wires with a significant attenuation, in order to gain key information on their respective behaviour under conditions of significant noise. As identified in ITRS roadmap, the power consumption is now one of the major constraints in chip design and has been identified as one of the top three overall challenges over the last 5 years. Faced to these constraints, further investigations are needed to accurately evaluate the consumption of the presented NGD active circuits.

8. References

Adler, V. & Friedman, E. G. (1998). Repeater Design to Reduce Delay and Power in Resistive Interconnect, *IEEE Trans. Circuits Syst. II, Analog and Digital Signal Processing*, Vol. 54, No. 5, pp. 607-616.

Bakoglu H. B. & Meindl J. D. (1985). Optimal Interconnection Circuits for VLSI, *IEEE Trans. On Electron. Devices*, Vol. 32, No. 5, pp. 903-909.

Barke E. (1988), Line-to-ground capacitance calculation for VLSI: a comparison, *IEEE Trans. of Computer Added Design*, Vol. 7, No. 2, pp. 295-298.

Deng, A. C. & Shiau, Y. C. (1990). Generic linear RC delay modeling for digital CMOS circuits, *IEEE Tran. on Computer-Aided Design*, Vol. 9, No. 4, pp. 367-376.

Deutsch., A. (1990). High-speed signal propagation on lossy transmission lines, *IBM J. Res. Develop.*, Vol. 34, No. 4, pp. 601-615.

Deutsch, A.; Kopcsay, G. V.; Restle, P. J.; Smith, H. H.; Katopis, G.; Becker, W. D.; Coteus, P. W.; Surovic, C. W.; Rubin, B. J.; Dune, R. P.; Gallo, T. A.; Jankis, K. A.; Terman, L. M.; Dennard, R. H.; Asai-Halsz, G.; Krauter, B. L. & Knebel, D. R. (1997). When are the transmission line effects important for on-chip interconnections, *IEEE Trans. on MTT*, Vol. 45, No. 10, pp. 1836-1846.

Deutsch, A.; Kopcsay, G. V.; Surovic, C. W.; Rubin, B. J.; Terman, L.M.; Dunne, Jr. R. P.; Gallo, T. A. & Dennard, R. H. (1995). Modeling and characterization of long on-chip interconnections for high speed-performance microprocessors, *IBM J. Res. Develop.*, Vol. 39, No. 5, pp. 547-667.

Dogariu, A.; Kuzmich, A. & Cao, H. & Wang, L. J. (2001). Superluminal light pulse propagation via rephasing in a transparent anomalously dispersive medium, *Optics Express*, Vol. 8, No. 6, pp. 344-350.

Eleftheriades, G. V.; Siddiqui, O. & Iyer, A. K. (2003). Transmission line for negative refractive index media and associated implementations without excess resonators, *IEEE MWC Lett.*, Vol. 13, No. 2, 51, pp. 51-53.

Elmore, W. C. (1948). The transient response of damped linear networks, *J. Appl. Phys.*, Vol. 19, pp. 55-63.

Friedman, E. (1995). Clock distribution networks in VLSI circuits and systems, *New York IEEE Press*.

Grover, F. (1945). Inductance Calculations Working Formulas and Tables, *Instrum. Soc. of America*.

Ismail, Y. I. & Friedman, E. G. (2000). Effects of inductance on the propagation, delay and repeater insertion in VLSI circuits, *IEEE Tran. VLSI Sys.*, Vol. 8, No. 2, pp. 195-206.

Ismail, Y. I.; Friedman, E. G. & Neves, J. L. (2000). Equivalent Elmore delay for RLC trees, *IEEE Tran. Computed-Aided Design*, Vol. 19, No. 1, pp. 83-97.

Kitano, M.; Nakanishi, T. & Sugiyama, K. (2003). Negative group delay and superluminal propagation: an electronic circuit approach, *IEEE Journal of Selected Topics in Quantum Electronics*, Vol. 9, No. 1, pp. 43-51.

Krauter, B. & Mehrotra S. (1998). Layout based frequency dependent inductance and resistance extraction for on-chip interconnect timing analysis, IEEE Design Automation Conference, *Proceedings of the ACM*, pp. 303–308.

Lucyszyn, S.; Robertson, I. D. & Aghvami, A. H. (1993). Negative group delay synthesiser, *Electron. Lett.*, Vol. 29, pp. 798-800.

Moore, G. E. (1965). Cramming more components into integrated circuits, *in Electronics*, Vol. 38, No. 8, pp. 114-117.

Munday, J. N. & Henderson, R. H. (2004). Superluminal time advance of a complex audio signal, *Appl. Phys. Lett.*, Vol. 85, pp. 503-504.

Nakanishi, T.; Sugiyama, K. & Kitano, M. (2002). Demonstration of negative group delays in a simple electronic circuit, *American Journal of Physics*, Issue 11, Vol. 70, pp. 1117-1121.

Palit, A. K.; Meyer, V.; Duganapalli, K. K.; Anheier, W. & Schloeffel, J. (2004). Test pattern generation based on predicted signal integrity loss through reduced order interconnect model, *16th Workshop Test Methods and Reliability of Circuits and Systems*.

Rabay, J. M. (1996). Digital integrated circuits, a design perspective, *Englewood Cliffs*, NJ: Prentice-Hall.

Ravelo, B. (Dec. 2008). Negative group delay active devices: theory, experimental validations and applications, *Ph.D. thesis* (*in French*), Lab-STICC, UMR CNRS 3192, University of Brest, France.

Ravelo, B.; Pérennec, A. & Le Roy, M. (2007a). Equalization of interconnect propagation delay with negative group delay active circuits, *11th IEEE Workshop on SPI*, Genova, Italy, pp. 15-18.

Ravelo, B.; Perennec, A.; Le Roy, M. & Boucher Y. (2007b). Active microwave circuit with negative group delay, *IEEE MWC Lett.*, Vol. 17, Issue 12, pp. 861-863.

Ravelo, B.; Perennec, A. & Le Roy, M. (2007c). Synthesis of broadband negative group delay active circuits, *IEEE MTT-S Symp. Dig.*, *Honolulu (Hawaii)*, pp. 2177-2180.

Ravelo, B.; Pérennec, A. & Le Roy, M. (2008a). Application of negative group delay active circuits to reduce the 50% propagation delay of RC-line model, *12th IEEE Workshop on SPI*, Avignon, France.

Ravelo, B.; Pérennec, A. & Le Roy, M. (2008b). Negative group delay active topologies respectively dedicated to microwave frequencies and baseband signals, *Journal of the EuMA*, Vol. 4, pp. 124-130.

Ravelo, B.; Pérennec, A. & Le Roy, M. (2009). Experimental validation of the RC-interconnect effect equalization with negative group delay active circuit in planar hybrid technology, *13th IEEE Workshop on SPI*, Strasbourg, pp. 1-4.

Ruehli, A. & Brennan, P. (1975). Capacitance models for integrated circuit metallization wires, *J. of Solid-State Integrated Circuits*, Vol. 10, No. 6, pp. 530-536.

Sakurai, T. (1983). Approximation of wiring delay in MOSFET LSI, *IEEE J. of Solid State Circuits*, Vol. 18, No. 4, pp. 418-425.

Sakurai, T. (1993). Closed-form expressions of interconnection delay, coupling and crosstalk in VLSI's, *IEEE Tran. on Electron. Devices*, Vol. 40, No. 1, pp. 118-124.

Siddiqui, O. F.; Erickson, S. J.; Eleftheriades, G. V. & Mojahedi, M. (2004). Time-domain measurement of negative-index transmission-line metamaterials, *IEEE Trans. MTT*, Vol. 52, No. 5, pp. 1449-1453.

Siddiqui, O.; Mojahedi, M.; Erickson, S. & Eleftheriades, G. V. (2003). Periodically loaded transmission line with effective negative refractive index and negative group velocity, *IEEE Tran. on Antennas and Propagation*, Vol. 51, No. 10.

Solli, D. & Chiao, R. Y. (2002). Superluminal effects and negative delays in electronics and their applications, *Physical Review E*, Issue 5.

Standley D. & Wyatt, J. L. Jr., (1986). Improved signal delay bounds for RC tree networks, *VLSI Memo*, No. 86-317, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Vlach, J.; Barby, J. A.; Vannelli, A.; Talkhan, T. & Shi. C. J. (1991). Group delay as an estimate of delay in logic, *IEEE Tran. Computed-Aided Design*, Vol. 10, No. 7, pp. 949-953.

Wang, L. J.; Kuzmich, A. & Dogariu, A. (2000). Gain-assisted superluminal light propagation, *Nature 406*, pp. 277-279.

Wyatt, J. L. Jr., & Yu, Q. (1984). Signal delay in RC meshes, trees and lines, *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 15-17.

Wyatt, J. L. Jr. (1985). Signal delay in RC mesh networks, *IEEE Tran. Circuits and Systems*, CAS-32(5), pp. 507-510.

Wyatt, J. L. Jr. (1987). Signal propagation delay in RC models for interconnect, *Circuit Analysis*, *Simulation and Design*, *Part II: VLSI Circuit Analysis and Simulation*, A. Ruehli, ed., Vol. 3 in the series Advances in CAD for VLSI, North-Holland.

Yu, Q.; Wyatt, J. L. Jr.; Zukowski, C.; Tan, H-N. & O'Brien, P. (1985). Improved bounds on signal delay in linear RC models for MOS interconnect, *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 903-906.

21

Book Embeddings

Saïd Bettayeb
University of Houston Clear Lake
Houston, TX 77058, USA

1. Introduction

Graph embeddings play an important role in interconnection network and VLSI (Very Large Scale Integration) design. Simulation of one interconnection network by another can be represented as a graph embedding problem. Determining the number of layers required to build a VLSI chip, also called book-embedding, is another application of graph embeddings. In this chapter, we explore the latter problem. After an overview of the results on book embedding of the hypercube, we present results on book embedding of the k-ary hyperube, a variant of the hypercube. We also present recently obtained results on the book embedding of the torus graph.

Graph embeddings have been studied in the literature extensively for the important role it plays in interconnection network and VLSI (Very Large Scale Integration) design (Bernhart & Kainen, 1979; Bettayeb et al., 1989; Bettayeb & Sudborough., 1992; Bettayeb & Sudborough, 1989; Chung et., 1987; Yannakakis, 1989). Simulating an interconnection network, say A, by another, say B, is represented as a graph embedding problem where the nodes and edges of A are mapped to nodes and paths of B. A book embedding of a graph G is the mapping of the nodes of G onto the spine of a book and the edges of G onto pages so that the edges assigned to the same page do not cross. Determining the minimum number of pages required for such an embedding is the focus of this chapter. The minimum number of pages in which a graph G can be embedded is called the pagenumber of G, pg(G). Determining the pagenumber of an arbitrary graph has been shown to be NP-complete (Chung et al, 1987; Yannakakis, 1989). It remains NP-complete to determine if an arbitrary graph can be embedded in two pages. In a 1980 article, Garey et al (Garey et al., 1980) proved that determining the pagenumber of an arbitrary graph remains NP-complete even if we assume that the node embedding part is fixed, i.e. the layout of the nodes is given. An equally challenging task is the problem of determining the pagerwidth or geometric thickness of an arbitrary graph G which defined to be the minimum number of layers in a planar drawing of an arbitrary G.

Researchers have been drawn to study this problem and variations of this problem (Chung et al., 1987; Galil et al., 1989; Yannakakis, 1989) because of its many and diverse applications such as fault tolerant computing (Chung et al., 1987), graph drawing, and graph separators (Galil et al., 1989). Other problems remain to be solved such as the relationship of the

pagenumber of a graph and other invariants. Enomoto and Miyauchi(Enomoto & Miyauchi, 1999) considered the case where edges may use more than one page.
The pagenumber of a graph has strong implications in VLSI design. The pagenumber is the minimum number of layers required to produce a VLSI chip. Another area of VLSI design that can be described in terms of book embedding is the configuring of processors in the presence of faults. Given an array of processors, some of which may be faulty, we lay them in a line. This could be either physical or logical. Running parallel to the line of processors are bundles of wires. As we scan the line of processors, we activate switches connecting the good processors to a bundle of wires and bypassing the bad processors. The bundle of wires act like a stack in that, when a processor u requests a connection to another processor, u is connected to a particular bundle and pushes the other processor connections down to a wire. When our scan reaches the processor to which processor u connects, it is popped off the bundle since the wire is no longer needed, and the other connections are returned to their original positions. The desired property in this case, is the minimization of the number of bundles required to interconnect all of the good processors in the desired layout. This is used in the Diogenes method of fault tolerant design as described by Chung, Leighton, and Rosenberg (Chung et al., 1987). If we take each bundle of wires and represent it as a page, we have a book embedding. Chung, Leighton, and Rosenberg (Chung et al., 1987) have studied the book embedding problem for a variety of graphs including trees, grids, X-trees, Benes networks, complete graphs, and the binary hypercube.

2. Definitions and Terminology

Let $G = (V, E)$ be an undirected graph. A linear layout L of the vertices of G is a mapping of V to the set $\{1, 2, 3, \dots, n\}$, where $n = |V|$. Let (x, y) and (x', y') be edges in G such that $L(x) < L(y)$ and $L(x') < L(y')$. Then (x, y) and (x', y') intersect with respect to L if $L(x) < L(x') < L(y) < L(y')$. If, in the other hand, $L(x) < L(x') < L(y') < L(y)$ or $L(x') < L(x) < L(y) < L(y')$ the edges (x, y) and (x', y') are said to nest with respect to L.
A *book embedding*, also referred to as stack embedding, is a linear layout of the vertices and an assignment of the edges to pages such that no page contains a pair of edges that intersect. The pagenumber of a graph, also referred to as book thickness, is the minimum number of pages achieved among all possible book embeddings.
When each edge has to be drawn as a straight line segment, the problem is referred to as the *geometric thickness* or *real linear thickness* (Kainen, 1990; Dillencourt et al, 2000)
Another variation studied in the literature is the so-called *queue embedding*. A queue embedding is a linear layout of the vertices of a graph and an assignment of the edges to queues such that no queue contains a pair of edges that nests (Bettayeb et al. 2010).

A d-dimensional torus is the d-dimensional mesh with wraparound edges. The wraparound edges connect the first and last vertex in each dimension. The notation $[n_1 \times n_2 \times \dots n_d]$ denotes the d-dimensional torus where $n_i, 1 \leq i \leq d,$ is the size of the i^{th} dimension.
The binary hypercube of dimension n, denoted by $Q_2(n)$ has 2^n vertices labeled by the binary representation of integers between 0 and $2^n - 1$. Two vertices are connected by an edge if and only if their labels differ in exactly one bit position. The k-ary hypercube of dimension n, denoted by $Q_k(n)$ has k^n vertices labeled by the k-ary representation of

integers between 0 and $k^n - 1$. Two vertices are connected by an edge if and only if their labels differ in exactly one position by one (modulo k).

3. Book Thickness of Graphs

In a 1979 article, Bernhart and Kainen (Bernhart & Kainen, 1979) introduced the problem of book embedding. They proved that the pagenumber of a graph G, $pg(G) \leq 2$ if and only G is a subgraph of some planar graph. They also conjectured that planar graphs have unbounded pagenumber. Heath and Istrail (Heath & Istrail, 1992) disproved this conjecture and proved that the pagenumber of graphs with genus g is O($\sqrt{g}$). The genus of a graph is defined as follows. A graph G can be embedded with no edge crossing on a compact orientable two-manifold surface on which a number of handles have been placed. A handle is used by an edge that would otherwise cross other edges. The genus of a surface is the number of handles on that surface. The genus of a graph is the minimum genus of all possible surfaces on which G can be embedded. So planar graphs have genus 0 since no handle is needed.

In (Kainen & Overbay, 2003, the authors studied the pagenuumber in terms of block-cutpoint tree. A block-cutpoint graph of a graph $G = (V, E)$ is a bipartite graph $G' = (V', E')$ where $V' = X \cup Y$. There is a vertex $x\ in\ X$ for each block of G and a vertex $y\ in\ Y$ for each cutpoint in G. The vertices x and y are connected by an edge if and only if the block represented by x contains the cutpoint y. A graph G is connected if and only if its block-cutpoint is tree. They showed that the pagenumber of a graph is the maximum of the pagenumber of its blocks. They also showed that a graph is planar if and only if it homeomorphic to a graph with pagenumber at most two.

4. Book Embedding of Planar Graphs

A graph $G = (V, E)$ has pagenumber 0 if and only if $|E| = 0$, $i.e.\ G$ consists of isolated vertices. Trees are shown to have pagenumber 1. A graph is said to be outerplanar if it can be drawn in the plane so that all of its vertices lie on the same face. Outerplanar graphs have pagenumber one. The following theorem is due to Kainen and Overbay (Kainen & Overbay, 2003):

Theorem: A graph G has pagenumber k with vertex ordering $v1, v2, \ldots, vn$ if and only $G = G_1\ U\ G_2\ U\ \ldots\ G_k$, where each G_i is an outerplanar graph embedded with vertex ordering $v_1, v_2, \ldots, v_n$.

In (Bernhart & Kainen, 1979), the authors conjectured that planar graphs have unbounded pagenumber but this was disproved by Heath and Istrail (Heath & Istrail, 1992). Buss and Shor (Buss & Shor, 1984) gave an algorithm that embeds all planar graphs in nine pages. Heath and Istrail (Heath & Istrail, 1992)gave an improvement that achieves seven pages. Yannakakis (Yannakakis, 1989) brought the number to four and showed that indeed four pages are sufficient and necessary.

4.1 Outline of Yannakakis Algorithm

Let C be a cycle C of the graph G that contains no external chords. Denote the vertices of C by $1, 2, \ldots, p$. The edge $(1, p)$ is called the long edge, and the rest of edges of C are called

short edges. Let G_c be the graph obtained from C by adding its chords. If F and F′ are two inner faces of G_c where the long edge of F is a short edge of F′ then all vertices of F must appear between two consecutive vertices of F′. First, Let K be the cycle bounding the inner face of G_c that contains the long edge of C. Its short edges must be the long edges of the other inner cycles. Layout the interior of K. Then, expand recursively the inner cycles.
As pointed out in (Yannakakis, 1989), the vertices of a planar graph can be partitioned into levels. All edges connect either vertices of the same level or vertices of adjacent levels. The former are called *level-edges* and the latter *binding edges*. Level 0 consists of the vertices forming the cycle K. Laying out the interior of K is accomplished by first laying level 1 vertices and coloring the short edges of K and the binding edges between levels 0 and 1. Expand recursively the cycles formed by level 1 vertices.

5. Book Embedding of the Torus and the k-ary hypercube

In (Bettayeb & Hoelzeman, 2009), we established the upper bound and lower bound on the pagenumber of the k-ary hypercube. The *k*-ary hypercube is a generalization of the binary hypercube. It is defined as follows. The *k*-ary hypercube of dimension *d* is an undirected graph of k^d vertices labeled by the integers 0 through k^{d-1}. Two nodes x and y are connected if and only the k-ary representations of their labels differ in exactly one position by one modulo k. We showed that the pagenumber of the n-dimensional k-ary hypercube is at least 2^{n-1} and the upper bound is $(\frac{a^n-1}{a-1})$ where a $=\lceil\frac{k}{2}\rceil$. In (Chung et al., 1987), Chung et al. showed that the binary hypercube of dimension n, $Q_2(n)$, admits an $n-1$ page embedding which is within a factor of 2 of optimal because the lower bound can easily be shown to be $\frac{n}{2}$. However, Heath, Leighton and Rosenberg (Heath et al., 1992) have shown that the pagenumber of the ternary hypercube of dimension n has a lower bound $\Omega(3^{n/9})$. It seems that when k is even the pagenumber grows linearly with the number of dimensions while it grows exponentially when k is odd. In (Bettayeb et al., 1020) we show that the pagenumber the k-ary hypercube depends on the parity of k. When k is even the pagenumber of $Q_k(n)$ grows linearly with the dimension n but when k is odd, the pagenumber grows exponentially with n. With the d-dimensional torus $[n_1 \times n_2 \times \ldots n_d]$, if all n_i, $1 \le i \le d$, are even the pagenumber is grows linearly with the number of dimensions d. In that paper, we also describe a layout technique with sequential corrections of the order of vertices that mitigates the problem for the case when dimensions are odd. Basically, the technique modifies the standard layout of alternating left-to-right and right-to-left segments with an amortizations of corrections that allow the reverse order to be realized without paying the penalty of all edges in the first-to-last wraparound connections to be simultaneouslu crossing. This would reduce the number of pages.
Recently, we showed that for any d-dimensional torus the pagenumber is bounded by $2^{d+1} - 3$ (Bettayeb et al., 2010). It had been shown (Bettayeb & Hoelzeman, 2009; Chung et al., 1987; Yannakakis, 1989) that the pagenumber of a graph is at least as large as the minimum number of outerplanar graphs into which it can decomposed.
For the queue embedding problem, the queue number for the torus grows linearly in the number of dimensions, regardless of the parity of the sizes of its dimensions. The queue number of the k-ary hypercube $Q_k(n)$ also grows linearly with the dimension n and does not depend on the parity of k. It is indeed $2n-1$.

6. Conclusion

In this chapter, we presented results on book embedding and queue embedding of graphs. The upper bound for the book-embedding of the torus we achieved is $2^{d+1} - 3$. It is interesting to know if this could be improved. Heath, Leighton and Rosenberg (Heath et al. 1992) showed that the ternary hypercube has a lower bound $\Omega(3^{n/9})$. It follows that the lower bound for torus is exponential in the number of dimensions when the sizes of its dimensions are odd. The authors in (Heath et al., 1992) conjectured that family of graphs with large queue number and small page (or stack) number do not exist. In (Bettayeb et al., 2010), we describe a class of graphs, namely the n-dimensional k-ary modified hypercubes which have pagenumber $O(n)$. We conjectured that the queue number for such graphs grow more rapidly than anu linear function of the dimension n.

7. References

Bernhart, F.; Kanien P.C., (1979). The Book Thickness of a Graph, *Journal of Combinatorial Theory*, Series B (27), 1979, pp. 320-331.

Bettayeb, S.; (1995). On the K-ary Hypercube. *Journal of Theoretical Computer Science* 140, 1995, pp. 333-339.

Bettayeb, S.; Heydari, H.; Morales, L.; Sudborough, I.H., (2010). Stack and Queue Layouts for Toruses and Extended Hypercubes. To appear

Bettayeb, S.; Hoelzeman, D., (2009). Upper and Lower Bounds on the Pagenumber of the Book Embedding of the k-ary Hypercube. *Journal of Digital Information Management*, 7 (1), 2009, pp. 31-35.

Bettayeb, S.; Miller, Z.; Sudborough, I.H., (1992). Embedding Grids into Hypercubes. *Journal of Computer and System Sciences*, 45 (3), 1992, pp. 340-366.

Bettayeb, S.; Sudborough, I.H., (1989). Grid Embedding into Ternary Hypercubes. *Proc. Of the ACM South Central Regional Conference*, 1989, pp. 62-64.

Buss, J.F.; Shor, P.W., (1984). On the Pagenumber of Planar Graphs. Proc. Of the 16th Annual Symposium on Theory of Computing, 1984, pp. 98-100.

Chung, F.R.K; Leightonm F.T; Rosenberg, A.L. (1983). DIOGENES: A Methodology for Designing Fault Tolerant Processor Arrays. *13th Conf. on Fault Tolerant Computing, 1983, pp. 26-32.*

Chung, F.R.K; Leightonm F.T; Rosenberg, A.L. (1987). Embedding Graphs in Books: A Layout Problem with Application to VLSI Design. *SIAM Journal of Algebraic Discrete Methods*, 8 (1), 1987, pp. 33-58.

Dean, A.M.; Hutchinson, J.P. (1991). Relations among Embedding Parameters for Graphs. *Graph Theory, Combinatorics, and Applications*, vol. 1, Wiley Interscience Publ., New York, 1991, pp. 287-296.

Dean, A.M.; Hutchinson, J.P. ; Sheinerman, E.R. (1991). On the Thickness and Arboricity of a Graph. *Journal of Combinatorial Theory Series B,* vol. 52, 1991, pp. 147-151.

Dillencourt, M.B.; Epstein, D.; Hirschberg, D.S. (2000). Geometric Thicknes of Complete Graphs. *Journal of Graph Algorithms and Applications.* vol. 4, 2000, pp. 5-17.

Enomoto, H.; Miyauchi, M.S., (1999). Embedding Graphs into a Three Page Book with O(M log N) crossings of edges over the spine. *SIAM Journal of Discrete Math*. 12, 1999, pp. 337-341.

Heath, L.S; Istrail, S. (1992). Comparing Queues and Stacks As Mechanisms for Laying out Graphs. *Journal of the Association for computing Machinery*, 39, 1992, pp. 479-501.

Heath, L.S; Leighton, F.T.; Rosenberg, A.L. (1992). Comparing Queues and Stacks As Mechanisms for Laying out Graphs. *SIAM Journal of Discrete Mathematics*, 5 (3), 1992, pp. 398-412.

Kainen, P.C.; Overbay, S. (2003). Book Embeddings of Graphs and a Theorem of Whitney. *Technical Report GUGU-2/25/03, http:// www.georgetown.edu/faculty/kainen/pbip3.pdf* .

Kainen, P.C. 1990. The Book Thickness of a Graph. *Congr. Numer*., 71, 1990, pp. 127-132.

Yannakakis, M. (1989) Embedding Planar Graphs in four Pages., Journal of Computer and System Sciences, 38 (1), 1989, pp. 36-67.

22

VLSI Thermal Analysis and Monitoring

Ahmed Lakhssassi and Mohammed Bougataya
Université du Québec en Outaouais
Canada

1. Introduction

The rapid evolution in the industry of the integrated circuits during the last decade was so quick that currently it is possible to integrate complex systems on a single SoC (System on a Chip). Due to aggressive technology scaling, VLSI integration density as well as power density increases drastically. As thermal phenomena research activities on micro-scale level are gaining popularity due to an abundance of SoC and MEMS-based applications, various measurement techniques are needed to understand the thermal behaviour of VLSI chip. In particular, measurement techniques for surface temperature distributions of large VLSI systems are a highly challenging research topic. Hence, surface peaks thermal detection is necessary in modern VLSI circuits; their internal stress due to packaging combined with local self heating becomes serious and may result in large performance variation, circuit malfunction and even chip cracking.

One of the important questions in the field of thermal issues of VLSI systems and micro-systems is how to perform the thermal monitoring, in order to indicate the overheating situations, without complicated control circuits. Traditional approach consists of placement of many sensors everywhere on the chip, and then their output can be read simultaneously and compared with the reference voltage recognized as the overheating level.

The idea of the proposed method is to predict the local temperature and gradient along the given distance in a few places only on the monitored surface, and evaluate obtained information in order to predict the temperature of the heat source. Therefore, in the case of an SoC devices, there is no place on the layout for the complicated unit performing computations, but there is also no need for it, as we only want to detect the overheating situations. These peaks are essentials during thermal die monitoring to avoid a critical induced thermo-mechanical stress. Moreover, in most cases, the overheating occurs only in one place.

Due to aggressive technology scaling, VLSI integration density as well as power density increases drastically. For example, the power density of high performance microprocessors has already reached 50W/cm^2 at 100nm technology and it will reach 100W/cm^2 at 50nm technology (ITRS, 2003). This evolution towards higher integration levels is motivated by the needs of advanced high performance, lighter and more compact systems with less

power consumption. Meanwhile, to mitigate the overall power consumption, many low power techniques such as dynamic power management (Wu et al., 2000), clock gating (Oh & Pedram, 2001), voltage islands (Puri et al., 2003), dual V_{dd}/V_{th} (Srivastava et al., 2004) and power gating (Kao et al., 1997), (Long & He, 2003) are proposed recently. These techniques, though helpful to reduce the overall power consumption, may cause significant on-chip thermal gradients and local hot spots due to different clock/power gating activities and varying voltage scaling. It has been reported in (Gronowski et al., 1998) that temperature variations of 30°C can occur in a high performance microprocessor design. The magnitude of thermal gradients and associated thermo-mechanical stress is expected to increase further as VLSI designs move into nanometer processes and multi-GHz frequencies.

Nevertheless, the growth of power density dissipated brought a number of critical thermo-mechanical problems. Thus the heat produced in the structure of the VLSI devices is directed towards its edges where it is dissipated by radiation, conduction, or convection. The principal effect of the absence of a good dynamic thermal management is the gradual and continuous degradation of the quality of performance as well as some other direct effects on the life cycle of the electronic systems (Buedo et al., 2002). Thus, an algorithm for the detection and the localization of the thermal peaks is extremely important in order to manage the thermal stress on high density semiconductor devices. These peaks are essentials during thermal die monitoring to avoid a critical induced thermo-mechanical stress.

This study presents a VLSI thermal peak monitoring approach, using GDS, for development of SPTDA algorithm. The proposed algorithm using only two sensor cells will be formulated in a manner to facilitate the development of modular architectures using minimum silicon space in regards of their implementation in VLSI. The architecture selected will be modelled in high level languages, simulated in order to evaluate its performances, and then implemented on a FPGA (Field-Programmable Gate Array). A closed loop of simulation is used in order to evaluate the performances of the architectures proposed at each stage. The proposed architecture of the algorithm will be designed in a modular perspective after the separation of the different elementary functions of the algorithm. Hence the design of SPTDA with flexible modular-based architecture will be presented. The architecture is designed in high level languages such as Matlab™ – Simulink®, simulated, tested using VHDL and synthesized using Xilinx™ ISE (Xilinx, 2009) and Altera™ DSP (Altera, 2009) tools. The simulation and hardware implementation results obtained will be compared to a finite element method (FEM) temperature prediction of the entire GDS method configuration cells.

This chapter will present a packaged VLSI thermal analysis by FEM, thermal monitoring approach using GDS (Gradient Direction Sensors) method. Also the design of surface peaks thermal detector algorithm (SPTDA) with flexible modular-based architecture will be presented.

2. Thermal monitoring by Gradient Direction Sensors method

This part presents the first approach to the method of predicting the temperature of a single heat source on the VLSI device surface. The second approach used concern prediction of the temperature gradient along the given distance in a few places only on the monitored chip-surface. The information obtained will be used to evaluate thermal peak position in order to achieve the temperature of the heat source. This means that we know temperature values in some places on the chip and we try to find the temperature values of the heat sources and eventually predict thermo-mechanical stress distribution in the whole structure.

The proposed algorithm is based on the GDS method for evaluating a single heat source on the chip surface. This method principle of work is explained in details in (Wójciak & Napieralski, 1997). For two sensors, A and C placed in the distance *a* (Fig. 1), the difference between their output voltages is proportional to the changes of the temperature value along the distance *a* (Wójciak & Napieralski, 1997). This is true only when the heat source is directly on the line AC for any other cases the values of the angle α has to be taken into account for the proper calculation of ΔT (1).

$$\frac{\Delta T}{\Delta r} = \frac{T_C - T_A}{a.\cos\alpha} \leftrightarrow \frac{V_C - V_A}{a.\cos\alpha} \tag{1}$$

where r is the distance from the heat source. In figure 1, we have:

$$b = \overline{AD}, \overline{AD} \leftrightarrow T_B - T_A \leftrightarrow V_B - V_A \tag{2}$$

$$b + c = \overline{AE}, \overline{AE} \leftrightarrow T_C - T_A \leftrightarrow V_C - V_A \tag{3}$$

In order to obtain the information about angle α we should apply the third sensor. In the simplest case the GDS contains only three temperature sensors placed in distance *a* (fig. 1)

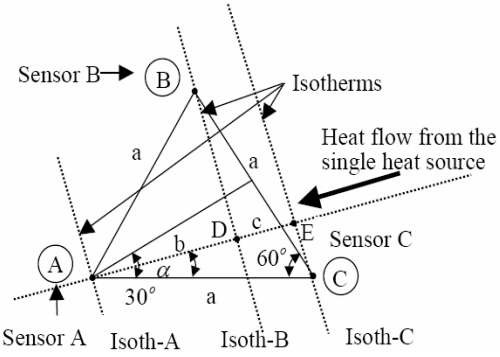


Fig. 1. The 3 sensors cell α ∈ (0°, 30°) (Wójciak & Napieralski, 1997)

On the basis of eqs. (2) and (3) and the geometrical dependencies from (fig. 1), eq. (4) is obtained:

$$\tan\alpha \leftrightarrow \frac{2}{\sqrt{3}} \cdot \left(\frac{V_B - V_A}{V_C - V_A} - \frac{1}{2} \right) \tag{4}$$

Using the cell we can obtain information on the temperature distribution and partly on the position of the heat source. In order to obtain the temperature value of a single punctual heat source we have to calculate the distance between the sensor and the heat source.

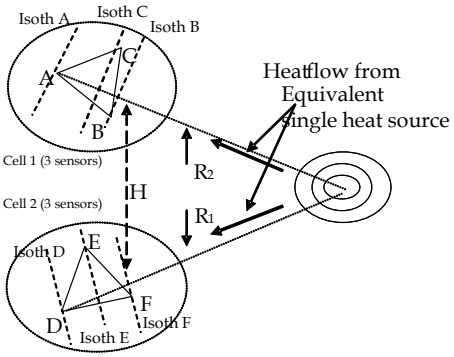


Fig. 2. Problem description and the distribution of the sensors cells

Two sensor cells are required for this purpose (fig2). The cells are placed in a given distance (H) and each of them gives information about the angle α (α1 and α2) in the direction of the heat source. Hence, the heat source and cells from a triangle in which the length of one side and values of the angles adjacent to this side are known. This means that we can calculate the distances between the heat source and sensors. Now we can calculate the temperature gradient along the known distance. By adding it to the temperature of the sensor we obtain the temperature of the heat source. Two sensor cells A,B,C and D,E,F are placed in two corners of a monitored layout in the distance H. Hence, the temperature of the heat source can be obtained by equation (5).

$$T_S \leftrightarrow V_S = \frac{H}{a} \times (V_C - V_A) \times \frac{(\tan^2 \alpha_1 + 1) \times (\sqrt{3} + \tan \alpha_2)}{\sqrt{3} \times (1 - \tan \alpha_1 \times \tan \alpha_2) - (\tan \alpha_1 + \tan \alpha_2)} + V_A \quad (5)$$

Figure 2 shows the proposed distribution of the 6 sensors divided into 2 cells located whether on or outside the chip.

3. Algorithmic Design Methodology

3.1 Presentation of algorithmic division

The general flowchart of the proposed SPTDA algorithm is shown in figure 3. It shows some parallelism in computation between the internal variables represented by tanα1 and tanα2. This parallelism will be used later in the architectural design in order to optimize the speed of the running implementation. Since successive evaluation of heat source temperature is needed for fast heat-source changes. The algorithm will create 2 triplets of values coming form the detectors. These 2 triplets will be used to calculate the tangents values and estimate R_1, R_2 and the temperature of the heat source. By dividing the algorithm into its basic arithmetic elementary operations (AEO) we will be able to formulate its data flow graph in order to model the algorithm.

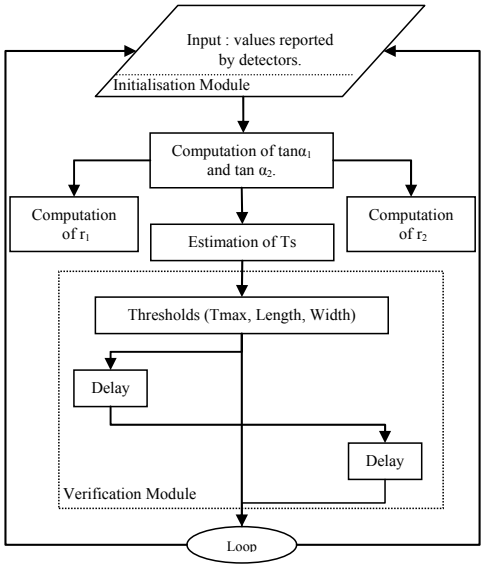


Fig. 3. Flowchart of the SPTDA algorithm

3.2 Architecture modeling using Simulink®

Before modelling the system using Xilinx ISE® or Altera DSP® tools we have to perform a quantification analysis in order to determine the combination of fixed bits that will lead to the minimum quantification error. Due to the fixed number of bits in the fixed point representation, overflow and underflow problems are serious and common. Thus, a set of simulations is required to determine the best fixed point representation. Figure 4 show that a quantification combination of a signed 11.4 and 12.4 bits had great differences with the main function; 13.4 was the turning point, and increasing the number of bits presented no great enhancement to the result. However, it is important to note that the quantification process induce a certain loss of precision. The fixed point specification has to be applied to every operation and even every constant block in a model.

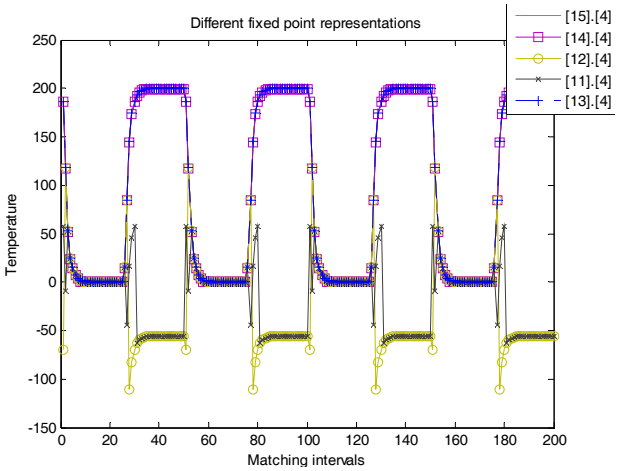


Fig. 4. Results generated using different fixed-point representations

4. Thermal analysis and computational results

4.2 WSI physical structure

In this part we will present steady state and transient thermal analysis of WSI (Wafer Scale Integration) chip junction by FEM approach. It will be used to build models to validate thermal peaks prediction by GDS method. Hence, the geometrical coordinates of the investigated source can be obtained by applying the gradient direction sensors. This way, the possibilities to minimize the thermal peaks in the critical surface areas for BGA (Ball Grid Array) packaged WSI devices can be explored.

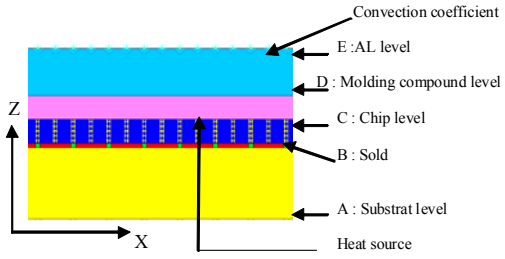


Fig. 5. Cross section view of the simulated WSI structure

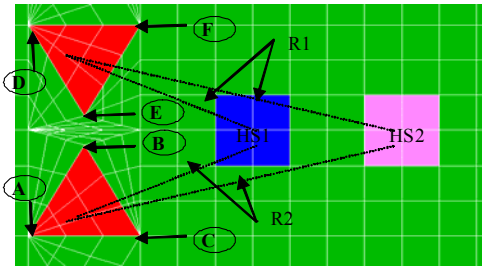


Fig. 6. Physical position of cell sensors and heat sources on the FEM model

As illustrated in figure 5 (dimensions not respected), the WSI device studied is multilevel structure with simple Si (silicon) substrate covered with different layers, Al (Aluminum), solder ball, and molding compound. The packaging assembly was a ceramic BGA. Once the geometry of the device had been determined and the heat transfer mechanisms quantified, it was possible to model the system using finite element analysis. Using the computer code NISA (Numerical Integrated Elements for System Analysis), a 3-D model was created with more than 100 000 isoparametric thermal shell elements. For the heating computations, this element models the 3-D state of heat flow. For the thermal part, the element has the temperature (T) as the only degree of freedom at each node. Figure 6 shows heat sources and sensors emplacement in the surface of processor. That will enable us to establish in-situ sensor network to achieve the most homogeneous thermo-mechanical cartography. In this study we present a case of WSI structure with internal heat generation. This configuration will enable us to simulate device intense activity and to construct thermal control unit that will ensure a suitable cooling. Moreover, we have to make sure that the temperature device structure variation remains suitable for appropriate induced thermo-mechanical stress. Figure 7 shows position of cell sensors on WSI device for temperature and stress results.

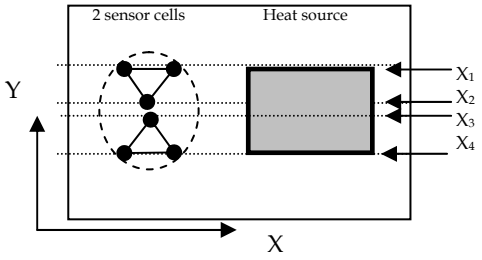


Fig. 7. Schematic position of sensors cell and heat source on WSI device

4.3 WSI Thermal monitoring

In (Lakhsasi et al., 2006) a method has been presented in details, which can be used for mixed fluid-heat transfer approach for VLSI steady state thermal analysis used to analyze IC package problem. However, during thermal analysis the temperature of the chip is determined for typical packages and power levels using fluid boundary conditions to

evaluate equivalent convection coefficient to be applied as a thermal junction BC's. Therefore, the knowledge of the complete temperature field implies the knowledge of the temperature gradients at all times, which is of significance for reliability issues. For the large VLSI device dissipating multiple localized heat sources the single heat source can be considered. As shown in figure 8, fast variations of temperatures in space and in time influence the local peak temperature. Hence, in the case of excessive heating or large amount of power dissipated during short time, the complete transient finite element computations are strongly recommended.

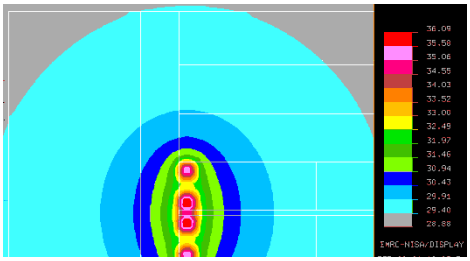


Fig. 8. VLSI device transient thermal analysis dissipating multiple localized heat sources

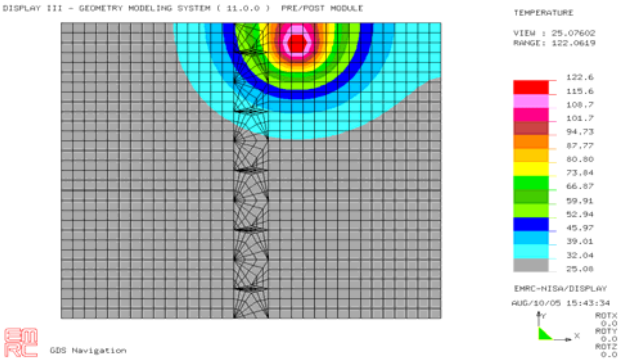


Fig. 9. VLSI device steady state thermal analysis dissipating single localized heat source

In this study, investigations are done for the simplest case, only six temperature sensors (A,B,C,D,E and F, Figure 2) in the form of two sensor cells and one single power heating source, in order to validate prediction with the 3-D FEM model. The simulations have been carried out for a one source placed at the junction surface level. As expected, the peak temperature profile is located at the centre of the heat source (figure 9). There is subsequent relaxation on temperature gradients through the structure leading to an essentially uniform temperature variation ΔT. The sensor cells can be placed in any way out of the monitored area (different distance H between cell 1 and cell 2), but in some cases adequate placement can simplify the thermal control unit design. In this part the results of thermal peaks can be very useful for indicating overheating situations and critical thermo-mechanical stress occurring in the device structure. Hence, table 1 display a comparison between the temperatures peaks on surface with different implementation under the same conditions.

Thus, the FEM results obtained (Figure 10 and 11) are in full agreement with the GDS predictions.

In this study we use NISA program to construct 3D thermo-mechanical model of WSI device. Furthermore, the application of the finite element method to determine the gradient of temperature that arises in WSI structure is not always simple especially for multi interconnection components such as ball grid array and flip-chip packages. This thermal investigation is based on power loss density distribution (thermal cartography) combined with steady state finite element analysis to predict spatial thermo-mechanical stress at different location in the WSI structure. Therefore, analytical GDS thermal prediction will be compared with FEM method computations.

Detectors Values (°C)			
Detector	**Set 1**	**Set 2**	**Set 3**
V_A	68.655	98.655	75.324
V_B	70.248	100.248	76.324
V_C	71.325	101.325	77.055
V_E	69.365	99.365	75.603
V_F	70.325	100.325	76.540
V_G	72.318	102.318	78.649
Temperature peaks estimated on surface (°C)			
Estimated by	**Set 1**	**Set 2**	**Set 3**
FEM	82.115	115.213	84.632
SPTDA Float	85.320	115.300	85.730
SPTDA Fix	84.060	114.100	84.190
SPTDA Altera	85.630	115.600	85.310

Table 1. Temperature peaks comparisons by FEM and SPTDA

During implementation the fixed-point representation of the SPTDA algorithm was used. After implementation, the same input is directed to the algorithm simultaneously in Simulink® and on the FPGA board. The results are routed back to Simulink®, multiplexed, and projected on the same 2-D graph in order to compare the output. As many simulations and co-simulations have preceded the implementation, the result was expected to be the same as the comparison between the floating and the fixed point comparison. An optimal frequency close to 100 MHZ has been achieved. Furthermore, the VHDL TB (Test Bench) was constructed and the force-file was used to stimulate inputs and to compare with the algorithm predictions. Hence, the set of simulations revealed that the estimations generated by the SPTDA algorithm presented a great concordance with the predictions generated by the Finite Element Method (FEM) presented in (Lakhsasi et al., 2006; Bougataya et al., 2006).

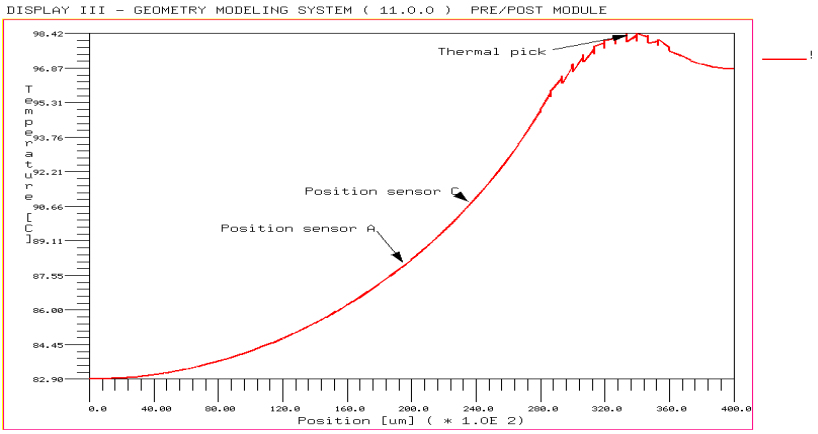


Fig. 10. Temperature distribution according to A-C-axis [C°]

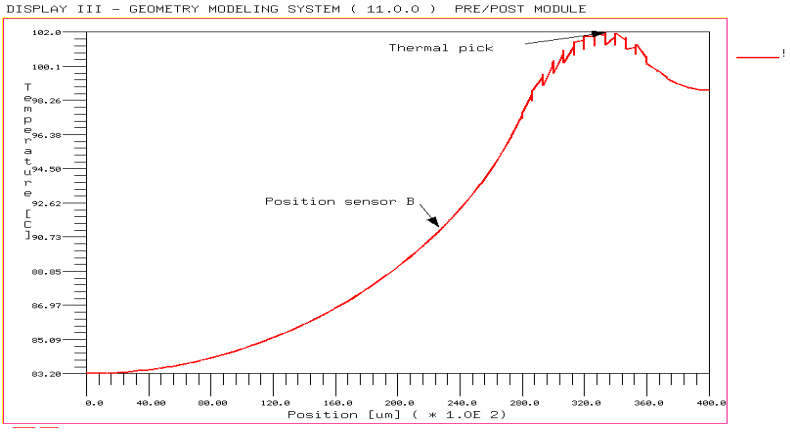


Fig. 11. Temperature distribution according to B-axis [C°]

4.4 Thermal boundary conditions issue

One of the most challenging issues in creating compact thermal models is to use an appropriate set of boundary conditions for generating 'data' with a detailed finite element model representing the thermal envelope of the application. The thermal analysis depends on:

- The cooling option (radiators top and/or bottom) applied,
- The location/vicinity and power of its heat-dissipating neighbours,
- The thermal conductivity of the VLSI materials: PCB, heat sink, package, substrate and heat spreader.

In this study we use the NISA finite element program to predict thermal behaviour of the VLSI device structure. A wide variety of boundary conditions can be applied using the FEM software. However, the boundary condition on the vertical sides of the simulation region is somewhat problematic. Placing a fixed boundary condition on these surfaces produces an

incorrect result, unless a very large simulation region is used at the expense of very long simulation run times. A more natural boundary condition is a zero flow condition across these tiny surfaces (adiabatic boundary conditions) as shown in figure 12. The remaining boundary conditions to be defined are on the bottom and top surfaces of the VLSI device, representing the heat sink interfaces. Because the VLSI devices is relatively thin and silicon and solder are good thermal conductors, heat flows happens mainly in the vertical direction, so the boundary conditions in both horizontal directions can be considered adiabatic. The uniform heat removal at the bottom and top is modelled by heat flux exchange coefficient *h* [W/m^2*^{o}K]. The power dissipated by the device is modelled by heat flux produced into the components. The problem formulation is presented graphically in figure 12.

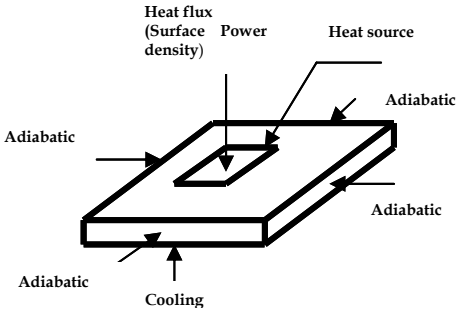


Fig. 12. VLSI device: inside thermal boundary conditions (TBC's) (BC).

5. Thermal stress prediction

If we know the peak temperature along with materials property and boundary conditions we can evaluate the thermal stress using the proposed approach. The algorithm presented herein might prove to be practical comparatively with thermal stress sensors methods especially for applications where the temperature has to be known only in a limited number of points, e.g. the determination of hot spot temperature or on-line temperature monitoring. The thermal stress (excluding the intrinsic one induced by packaging process) in thin films is given by the following expression:

$$\sigma_{th} = (E / (1 - \nu)) \Delta \alpha.\Delta T$$

Where E / (1 – ν) is the composite elastic constant for the different layers and Δα the difference in the coefficients of thermal expansion (CTE) between different level of packaging (Kobeda et al., 1989). In (Lakhsasi & Skorek, 2002) a method has been presented in details for calculation of the compressive stress in the silicon level given by:

$$\sigma_{xx,\,max} \approx -9.63 \, (E\alpha)Si \, .\Delta T_{in} \quad \text{(Kobeda et al., 1989)}$$
$$\sigma_{xx,\,max} \approx -9.63 \, (0.15 \times 2.8)Si \, .(102.3\text{-}81.1)$$
$$\sigma_{xx,\,max} \approx -8.574 \text{ MPa}$$

However, the induced compressive thermal stress will be combined to the intrinsic one due to the fabrication processes, and the stress due to the mechanical clamping mechanism.

Table 2 gives the thermo mechanical parameters for different materials used in the computation.

Materials	Chip (die)	M	BT Substrate	Solder Ball	Al	Die attach
Young's Modulus (GPa)	131	16	26(xy), 11(z)	17	70	100
Poison's Ratio	0.3	0.25	0.39(xz.yz), 0.11(xy)	0.4	0.33	0.25
CTE ppm/^{0}C	2.8	15	15(xy), 52(z)	21	22.4	0.8
Thermal Conductivity W/m.^{0}C	150	65	0.3	50	160	33
Density Kg/m^3	2330	1660	1660	8460	2700	3300
Specific heat J/kg. ^{0}C	712	1672	1672	957	960	1100

Table 2. Device Materials thermo-mechanical Properties

Method	Distance Cell 1–cell 2, H (μm)	Source Temp Tmax (^{o}C)	Distance Cell 1-source 1 (μm)	Distance Cell 2-source 1 (μm)
Analytical results by GDS	8000	100.2	10376	10809
3-D FEM model results.	8000	102.3	10148	10984

Table 3. Results comparison between GDS method and 3-D FEM model

The nodal temperatures stored in the thermal part were used to perform a thermal stress analysis of the device structure. It is assumed that the structure is stress-free at 25 ^{o}C. The presence of a temperature variation throughout the device structure causes deformation due to thermal contraction and expansion. A thermal stress analysis is performed to calculate these deflections and associated stresses due to the thermal loading. The computation is extended to the whole volume of the device structure.

The present approach may be suitable for large VLSI devices packaging, because it potentially allows the combination and integration of both thermal and mechanical control, in a single unit. As an example, the maximum level of stress generated by intense heating of WSI devices have been evaluated and examined by GDS method. During design and finale packaging of large VLSI devices, in-situ thermo-mechanical control unite must be implemented to ensure the safe fulfillments of their operating conditions. The nature of such interface materials must therefore be considered very carefully during WSI design and packaging. Further research should be focused on elaborating software tools for optimization of temperature sensor positions within the allowed area for an IC designer.

Figure 13 and 14 shows thermal stress profile according to X_1-axis. The peak compressive stress of σ_{xx}= -8.5 MPa is reported around the region of the heat source.

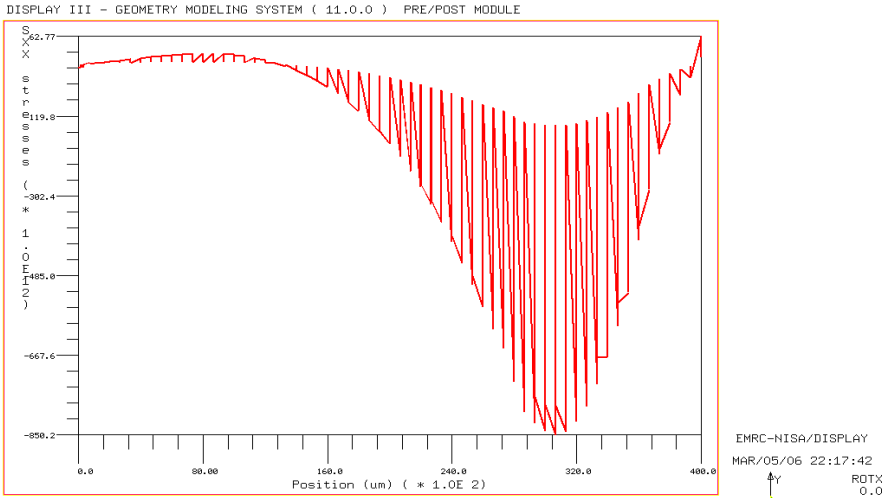


Fig. 13. VLSI thermal stress profile according to X_1-axis, σ_{xx}= -8.5 MPa Maximum

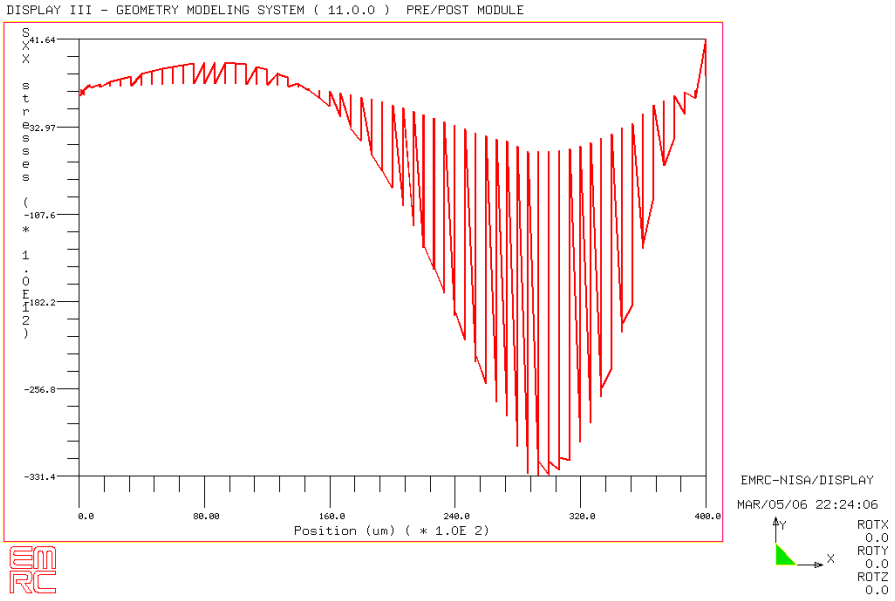


Fig. 14. VL VLSI thermal stress profile according to X_2-axis, σ_{xx}= -3.3 MPa Maximum

VLSI layers are very thin and any imperfection in structure may lead to cracking and subsequent shear-initiated delamination at the silicon level. The nature of such interface materials must therefore be considered very carefully in VLSI devices design for intense applications. Depending on the technology requirements and definition of failure, the mechanism of failure may take several forms.

6. Discussion

One of the important questions in the field of thermal issues of VLSI systems and microsystems is how to perform the thermal monitoring, in order to indicate the overheating situations, without complicated control circuits. Traditional approach consists of placement of many sensors everywhere on the chip, and then their output can be read simultaneously (Szekely, 1994) and compared with the reference voltage recognized as the overheating level.

The idea of the proposed algorithm is to predict the local temperature and gradient along the given distance in a few places only on the monitored surface, and evaluate obtained information in order to predict the temperature of the heat source. Therefore, in the case of an SoC devices, there is no place on the layout for the complicated unit performing computations, but there is also no need for it, as we only want to detect the overheating situations. These peaks are essentials during thermal die monitoring to avoid a critical induced thermo-mechanical stress. Moreover, in most cases, the overheating occurs only in one place.

In this chapter, a methodology to evaluate and predict a thermal peak of large VLSI circuit was presented. The important factors contributing to the device's thermal heating were characterized. The monitoring approach reported in this paper can be applied to predict the thermal stress peak of multilevel structures. Surface peaks thermal detection is necessary in modern VLSI circuits; their internal stress due to packaging combined with local self heating becomes serious and may result in large performance variation, circuit malfunction and even chip cracking. Also, in this paper GDS technique was used to develop SPTDA algorithm.

7. Conclusion

This study presented an approach to thermal and thermo-mechanical stress monitoring of VLSI chip. The possibility of evaluating thermal peaks and associated thermal stress distribution all over the monitored area by using GDS and FEM has been shown. Furthermore, this study presented an approach to the application of inverse problem for thermo-mechanical analysis. The thermal peaks of investigated source can be obtained by applying the gradient direction sensors method. The adequate placement of sensors can accurately evaluate the thermal peaks and simplify the thermo-mechanical control unit design. That will enable the chip designer to establish the most homogeneous thermo-mechanical cartography during operation. The cost of the thermal management of WSI device depends heavily upon the efficiency of the chip design. However, heat sources spatial distribution has a significant effect on the WSI device operation. Hence, in-situ thermo-mechanical control unite must be implemented to prevent unexpected pitfalls.

Another aspect presented in this study is related to surface peaks thermal detection in modern VLSI circuits; because their internal stress due to packaging combined with local self heating becomes serious and may result in large performance variation, circuit malfunction and even chip cracking. As an example, in this study GDS technique was used to develop SPTDA algorithm. Several approaches were implemented to achieve a better performance for the SPTDA operation. In (Wójciak & Napieralski, 1997), physical circuit architecture was proposed. However, complicated control components had to be deployed in the circuit which is not suitable for today's constraints regarding area, especially if the control algorithm needs to be placed on-chip, for example, on a sensor network node. Thereby, deploying the SPTDA in sensor network can be made feasible in the future.

8. References

A. Lakhsasi, A. Skorek," Dynamic Finite Element Approach for Analyzing Stress and Distortion in Multilevel Devices ", SOLID-STATE ELECTRONICS, PERGAMON, Elsevier Science Ltd., Volume 46/6 pp. 925-932, May 2002.

A. Lakhsasi, M. Bougataya, D. Massicotte: Practical approach to gradient direction sensor method in very large scale integration thermomechanical stress analysis, J. Vac. Sci. Technol. A 24(3), pp. 758-763, May 2006.

A. Srivastava, D. Sylvester, and D. Blaauw, "Concurrent Sizing, Vdd and Vth Assignment for Low-Power Design," in Proc. Design, Automation and Test in Eurpoe, vol. 1, Feb 2004, pp. 718–719.

Altera corp (http://www.altera.com), 2009.

C. Long and L. He, "Distributed sleep transistor network for power reduction," in Proc. Design Automation Conf., 2003.

E Kobeda and al '' in situ measurements during thermal oxidation of silicon '' J.vac.sci Technol B7 (2), Mar/aprl 1989

ITRS, International Technology Roadmap for Semiconductors (ITRS), 2003.

J. Kao, A. Chandrakasan, and D. Antoniadis, "Transistor sizing issues and tool for multi-threshold CMOS technology," in Proc. Design Automation Conf., 1997.

J. Oh and M. Pedram, "Gated clock routing for low-power microprocessor design," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 20, pp. 715–722, Jun 2001.

M. Bougataya, A. Lakhsasi and D. Massicotte: Steady State Thermo-mechanical Stress Prediction for Large VLSI circuits using GDS Method. IEEE CCECE06 Proceeding SBN:1-4244-0038-4, pp.917-921

P. Gronowski et al., "High performance microprocessor design," IEEE J. Solid-State Circuits, vol. 33, pp. 676–686, May 1998.

Q. Wu, Q. Qiu, and M. Pedram, "Dynamic power management of complex systems using generalized stochastic Petri nets," in Proc. Design Automation Conf., Jun 2000.

R. Puri, L. Stok, J. Cohn, D. Kung, D. Pan, D. Sylvester, A. Srivastava, and S. H. Kulkarni, "Pushing ASIC Performance in a Power Envelope," in Proc. Design Automation Conf., 2003.

Sergio Lopez-Buedo, Javier Garrido, and Eduardo I. Boemo: 'Dynamically Inserting, Operating, and Eliminating Thermal Sensors of FPGA-Based Systems', IEEE TRAN

ON COM AND PACK TECHNOLOGIES, VOL. 25, NO. 4, DECEMBER 2002, pp. 561-566.

V. Szekely, Thermal monitoring of microelectronic structures, Microelectronic J., 25 (1994) 157-170.

W. Wójciak and A. Napieralski ''Thermal monitoring of a single heat source in semiconductor devices – the first approach'' Microelectronics Journal 28 (1997) p 313-316.

Xilinx corp (http://www.xilinx.com), 2009.