Low Power JPEG2000 5/3 Discrete Wavelet Transform Algorithm and Architecture

Kay-Chuan Benny Tan

A thesis submitted for the degree of Doctor of Philosophy.

The University of Edinburgh.

November 2004

Abstract

With the advances in VLSI digital technology, many high throughput and performance imaging and video applications have emerged and increased in usage. At the core of these imaging and video applications is the image and video compression technology. Image and video compression processes are by nature very computational and power consuming. Such high power consumption will shorten the operating time of a portable imaging and video device and can also cause overheating. As such, ways of making image and video compression processes inherently low power is needed.

The lifting based Discrete Wavelet Transform (DWT) is increasingly used for compression digital image data and is the basis for the JPEG2000 standard (ISO/IEC 15444). Even though the lifting based DWT has resulted in several implementations of this algorithm, there is no work on the low power realisation of such an algorithm. Recent JPEG2000 DWT implementations are pipelined data-path centric designs and do not consider the issue of power. This thesis therefore sets out to realise a low power JPEG2000 5/3 lifting based DWT hardware architecture and investigates whether optimising at both algorithmic and architectural level will yield a lower power hardware. Besides these, this research also ascertains whether the accumulating Arithmetic Logic Unit (ALU) centric processor architecture is more low power than the feed-through pipelined data-path centric processor architecture.

A number of novel implementation schemes of the realisation of a low power JPEG2000 5/3 lifting based DWT hardware are proposed and presented in this thesis. These schemes aim to reduce the switched capacitance by reducing the number of computational steps and data-path/arithmetic hardware through the manipulation of the lifting-based 5/3 DWT algorithm, operation scheduling and alteration to the traditional processor architecture. These resulted in a novel SA-ALU centric JPEG2000 5/3 lifting based DWT hardware architecture that saves about 25% of hardware with respect to the two presented existing 5/3 DWT lifting-based architectures.

The proposed SA-ALU centric architecture also allows innovative scheduling and a novel embedded extension algorithm to be incorporated into its main operation. The embedded extension algorithm saves more than 50% of memory and power by doing away with explicit data extension operations entirely. Optimisation at architectural level also results in a novel addressing architecture called the dyadic multi-segments multi-level one-hot addressing architecture (MSML-OHA). The MSML-OHA reduces power by 20% through reducing switched capacitance by isolating and confining switching activities into a small local area by segmentation. A further reduction of 59% is achieved in the dyadically structured MSML-OHA (DMSML-OHA) when this is combined with the shared address lines architecture bought about by innovative scheduling.

Publication

Refereed Journal Papers

K. C. B. Tan and T. Arslan, *Low Power Embedded Extension Algorithm for the Lifting based Discrete Wavelet Transform in JPEG2000*, IEE Electronic Letters, Vol. 37, No.22, pp. 1328-1330, Oct. 2001.

Refereed Conference Papers

K. C. B. Tan and T. Arslan, *An Embedded Extension Algorithm for the Lifting based Discrete Wavelet Transform in JPEG2000*, 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2002), Vol: 4, pp. 3513-3516, May 2002.

K. C. B. Tan and T. Arslan, *Shift-accumulator ALU Centric JPEG2000 5/3 Lifting based Discrete Wavelet Transform Architecture*, 2003 IEEE International Symposium on Circuits and Systems (ISCAS 2003),Vol:5, pp. 161-164, May 2003.

Submitted Papers

K. C. B. Tan and T. Arslan, *Low Power Multi-segment Sequential One-hot Addressing Architecture*, accepted to be published in the IEE Proceedings of Circuits, Devices and Systems.

K. C. B. Tan and T. Arslan, *Low Power JPEG2000 5/3 Lifting Based Discrete Wavelet Transform Architecture*, submitted to IEEE transactions on Integrated Circuits and Systems I.

Declaration of originality

I hereby declare that the research recorded in this thesis and the thesis itself was composed and originated entirely by myself in the Institute of Integrated Micro and Nano Systems, School of Engineering and Electronics at The University of Edinburgh.

Kay-Chuan Benny Tan

Acknowledgements

Firstly, I would like thank my Supervisor Dr. Tughrul Arslan for giving me his guidance and this opportunity to do this PhD.

I would also like to especially thank my second supervisors Dr. John Hannah for his kind and invaluable advices, guidance and support for the last 3 years.

Thanks also go to Dr. Ahmet Erdogan for his expert advice on digital hardware design and tool.

I would like to thank Prof. Steve McLaughlin for giving me the occasional but necessary sanity checks. Thanks also to Dr. David Renshaw for making my 3 years interesting by kindly letting me demonstrate in the 2nd year project laboratory.

Many thanks to all those in the Department of Electronics and Electrical Engineering at the University of Edinburgh who made the last 3 years both rewarding and enjoyable.

And most important of all, I like to thank GOD for all He had done.

This thesis and the research project have been kindly supported by EPSRC under grant number N08322.

Contents

List of figures

List of tables

Acronyms and abbreviations

1-D	One-Dimensional
2-D	Two-Dimensional
ABITS	Advance Bus-invert/Transition Signalling Coding
ADCVSL	Adiabatic Differential Cascode Voltage Switch Logic
ALU	Arithmetic Logic Unit
AVC	Advance Video Coding
BITS	Bus-invert/Transition Signalling Coding
CAD	Computer Aided Design
CCD	Conventional Counter-decoder
CGC	Control Generated Clocks
CPL	Complementary Pass-gate Logic
CMOS	Complementary Metal Oxide Semiconductor
COM	(Electrical) Common
CWT	Continuous Wavelet Transform
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DG-MOSFET	Double Gate Metal Oxide Semiconductor Field-Effect Transistor
DG-SOI	Double Gate Silicon On Insulator
DPCM	Differential Pulse Code Modulation
DSDL-OHA	Double-Segment Double-Level One-Hot Addressing Architecture
DSP	Digital Signal Processor
DTMOS	Dynamic-threshold MOSFET
DMSML-OHA	Dyadic Structured Multi-Segments Multi-Level One-Hot Addressing Architecture
DVD	Digital Versatile Discs
DWT	Discrete Wavelet Transform
EMC	Electromagnetic Compatibility
ES	Elementary Stream
FDCT	Fast Discrete Cosine Transform

FET	Field-Effect Transistor
FIFO	First In First Out
FinFET	Fin Field-Effect Transistor
FIR	Finite Impulse Response
FSM	Finite State Machine
GALS	Globally Asynchronous Locally Synchronous
GCD	Greatest Common Divisor
GND	(Electrical) Ground
HDL	Hardware Description Language
HFET	Heterostructure Field-Effect Transistor
HH	2-D DWT High-High sub-band output
HL	2-D DWT High-Low sub-band output
HP	High Pass
HPF	High Pass Filter
HS	Half Sample
IEC	The International Electrotechnical Commission
INMEM	Intermediate Memory
IP	Intellectual Property
ISO	International Organisation for Standardization
ITU	The International Telecommunications Union
ITU-T	The International Telecommunication Union, Telecommunication Standardization Sector
ITRS03	International Technology Roadmap for Semiconductors, 2003 Edition
KLT	Karhunen-Loeve Transform
JPEG	Joint Photographic Experts Group
LH	2-D DWT Low-High sub-band output
LL	2-D DWT Low-Low sub-band output
LP	Low Pass
LOP	Low Operating Power
LPF	Low Pass Filter
LSTP	Low Standby Power
NMOS	N-channel Metal Oxide Semiconductor Field-Effect Transistor
NUDC	Source-to-Drain Non-uniformly Doped Channel

MJ2	Motion-JPEG2000
MMC	Multi-Module Chip
MPEG	Moving Picture Expert Group
MOBILE	Monostable-Bistable Transition Logic Element
MOP	Million Operation
MOSFET	Metal Oxide Semiconductor Field-Effect Transistor
MRA	Multi-Resolution Analysis
MSML-OHA	Multi-Segments Multi-Level One-Hot Addressing Architecture
MSB	Most Significant Bits
OD	Object Description
OHA	One-Hot Addressing Architecture
PC	Personal Computer
PMOS	P-channel Metal Oxide Semiconductor Field-Effect Transistor
PDA	Personal Digital Assistant
RF	Radio Frequency
RPMEM	Re-process Memory
RTD	Resonant Tunnel Diodes
RTL	Register Transfer Level
SA-ALU	Shift-Accumulator Arithmetic Logic Unit
SAIF	Switching Activity Interchange Format
SCCMOS	Super Cut-off Complementary Metal Oxide Semiconductor
SDF	Standard Delay Format
SiGe	Silicon Germanium
SOC	System-On-a-Chip
SOI	Silicon On Insulator
TSDL-OHA	Triple-Segment Double-Level One-Hot Addressing Architecture
TSTL-OHA	Triple-Segment Triple-Level One-Hot Addressing Architecture
UTB-SOI	Ultra-Thin Body Silicon On Insulator
VCD	Video Compact Disc
VTCMOS	Variable Threshold Voltage Complementary Metal Oxide Semiconductor
VQ	Vector Quantisation
WS	Whole Sample

Nomenclature

A	Electric current unit (Ampere)
C	Electric charge unit (Coulomb)
C_l	Load capacitance
$C(n)$	1-D DCT boundary condition correcting factor
$C(u)$	2-D DCT row boundary condition correcting factor
$C(v)$	2-D DCT column boundary condition correcting factor
$c_{j-1}(i)$	Present DWT level low pass output coefficients
$c_j(m)$	Previous DWT level low pass output coefficients
D_n	Electron diffusion coefficient
$DCTcl_{256}$	Computational load of 2-D DCT processing a 256 × 256 image
$DCTclb_{256}$	Computational load of 2-D DCT processing a 256 × 256 image with 16 × 16 blocking
$DWT5/3_{256}$	Computational load of 2-D 5/3 DWT processing a 256 × 256 image
$d_{j-1}(i)$	Present DWT level high pass output coefficients
E_{0^o}	Band-gap energy of a semiconductor at 0^o Kevin
E_1	Energy consumed by Andra's processor for processing a 128 × 128 image at clock period of 10ns with process completion time of 84650ns
E_2	Energy consumed by Andra's processor in idle period of 244350ns
E_3	Average energy consumed by Andra's processor for processing a 128 × 128 image at clock period of 10ns with process completion time of 329000ns
E_{cg}	Energy consumed by Andra's processor with clock gating
E_{psa}	Energy consumed by the proposed SA-ALU processor for processing a 128 × 128 at clock period of 10ns with process completion time of 329000ns
E_{sck}	Energy consumed by Andra's processor with its clock period stretched from 10ns to 38.8457ns
e	Natural number (Value = 2.718281828)
$F(n)$	Output coefficients of 1-D DCT
$F(u,v)$	Output coefficients of 2-D DCT
f_0	The maxima of the square of the elementary signal in fourier domain

f_{clk}	Clock frequency
$f(j)$	Input data samples to 1-D DCT process
$f(t)$	Time domain signal function
$f(x,y)$	Input data samples to 2-D DCT process
$\widetilde{H_1}(z)$	Sub-sampled output sample of analysis high pass filter
$\widetilde{HP}(z)$	Output sample from analysis high pass filter
$\widetilde{HPF}(z)$	Analysis high pass filter
$\widetilde{HPF}_e(z)$	Even coefficients of analysis high pass filter
$\widetilde{HPF}_o(z)$	Odd coefficients of analysis high pass filter
g	Spatial index
$g(i)$	Multi-resolution analysis wavelet function coefficients
$h(i)$	Multi-resolution analysis scaling function coefficients
I	Unit matrix
$I_{C_l sw}$	Load-capacitance C_l switching current
I_{dr}	Drain leakage current
I_{dsb}	Drain-to-substrate leakage current
I_l	Leakage current
I_{rs}	Reverse saturation current
I_{rvf}	Reduced voltage feed leakage current
I_{sbth}	Sub-threshold leakage current
I_{sc}	Short-circuit current
I_{sca}	Maximum instantiate short-circuit current
I_{sco}	Overall average CMOS system short-circuit current
i	Translation and time shift index
i_o	Index of the first sample
i_l	Index of the last sample
j	Scaling and frequency index
K	Scaling step
K^o	Temperature unit (Kelvin)
k	Boltzmann constant (Value = $1.3807 \times 10^{-23} J/K^o$)
Lc	Effective channel length
$\widetilde{L_1}(z)$	Sub-sampled output sample of analysis low pass filter
$\widetilde{LP}(z)$	Output sample from analysis low pass filter

$\widetilde{LPF}(z)$	Analysis low pass filter
$\widetilde{LPF}_e(z)$	Even coefficients of analysis low pass filter
$\widetilde{LPF}_o(z)$	Odd coefficients of analysis low pass filter
N	Total number of data sample in a row or column
n	Frequency index
n_{po}	Electron concentration in p-type silicon at thermal equilibrium
P_1	Power consumed by Andra's processor for processing a 128 $\times$ 128 image at clock period of 10ns with process completion time of 84650ns
P_2	Power consumed by Andra's processor in idle period of 244350ns
P_3	Average power consumed by Andra's processor for processing a 128 $\times$ 128 image at clock period of 10ns with process completion time of 329000ns
P_{ave}	Average CMOS power
P_{cg}	Power consumed by Andra's processor for processing a 128 $\times$ 128 image at clock period of 10ns with process completion time of 5.5555ms with clock-gating
P_{psa}	Power consumed by the proposed SA-ALU processor for processing a 128 $\times$ 128 at clock period of 10ns with process completion time of 329000ns
P_l	Static-leakage power
P_{sc}	Short circuit power
P_{sck}	Power consumed by Andra's processor with its clock period stretched from 10ns to 38.8457ns
P_{sco}	Overall average CMOS system short-circuit power
P_{sw}	Load-capacitance switching power
$P(z)$	Synthesis polyphase matrix
$\widetilde{P}(z)$	Analysis polyphase matrix
p	Size of block size in number of pixel
q	Elementary charge (value = $1.6022 \times 10^{-19}C$)
S	Transition/switching activity factor
s	Scale of the basic wavelets
$\tilde{s}(z)$	Analysis update (Primal lifting) step
T	Temperature
T_1	Active time period of 84650ns for Andra's processor to processing a 128 $\times$ 128 image at clock period of 10ns

T_2	Idle time period of 244350ns
T_3	Mixed activities time period of 329000ns consisting of 84650ns of active period and 244350ns of idle period
$\tilde{t}(z)$	Analysis predict (Dual lifting) step
t_0	The maxima of the square of the elementary signal in time domain
u	Row frequency domain index
V_{dd}	Rail supply voltage
V_G	Gate voltage
V_{gs}	Gate-source voltage
V_{rs}	Drain-to-substrate voltage
V_T	Threshold voltage
v	Column frequency domain index
W	Width of the MOSFET
$X(2n+1)$	Odd indexed input data sample
$X(2n)$	Even indexed input data sample
$X_{ext}(2n+1)$	Odd indexed extended input data sample
$X_{ext}(2n)$	Even indexed extended input data sample
X_{i_0}	First input data sample
X_{i_l}	Last input data sample
x	Row spatial index
x_{ac}	Channel thickness
$x(z)$	Input data sample
$x_e(z)$	Even input data sample
$x_o(z)$	Odd input data sample
$Y(2n+1)$	Odd indexed output processed coefficient
$Y(2n)$	Even indexed output processed coefficient
Y_{i_0}	First output processed coefficient term
Y_{i_l}	Last output processed coefficient term
$\widetilde{Y}_{H1}(z)$	Remainder factor after the first analysis high pass filter factoring
$\widetilde{Y}_{H2}(z)$	Remainder factor after the second analysis high pass filter factoring
$\widetilde{Y}_{L1}(z)$	Remainder factor after the first analysis low pass filter factoring
$\widetilde{Y}_{L2}(z)$	Remainder factor after the second analysis low pass filter factoring
y	Column spatial index

α	DCT processing asymptotics term
η	Time domain pulse width of elementary signal
$\gamma(s,\tau)$	Continuous wavelet transform output coefficients
$\Phi(2^j t)$	Multi-resolution analysis scaling function
$\Phi_{j,i}(t)$	Discrete scaling function
$\Psi'(2^j t)$	Multi-resolution analysis wavelet function
$\Psi(t)$	Elementary signal
$\Psi'(t)$	Mother wavelet
$\Psi_{s,\tau}(t)$	Basic wavelet function
$\Psi_{j,i}(t)$	Discrete wavelet
π	Pi, the ratio of the circumference of a circle to its diameter (Value = 3.141592654)
τ	Translation of the basic wavelets
θ	Phase angle of the sinusoid function of elementary signal

Chapter 1

Introduction

1.1 Introduction

The advance in Very Large Scale Integration (VLSI) digital technology during the last decades has led to the development of systems with both high throughput and performance capabilities. These capabilities have given systems, devices and applications the abilities to perform complex and computationally intensive tasks in real-time that were never possible before. The impact of this revolution has been most effective in the area of digital signal processing (DSP) where high throughput is most desirable. However, one major limitation of these systems is its operating time, which is dependent on the characteristics of the battery and the power consumption of such system. Battery technology has peaked in recent years whereas the number of computationally intensive applications requiring portability is increasing, placing an increasing demand on the power budget. Another issue, which applies to both portable and non-portable systems, is the heat dissipation. Overheating can reduce throughput and the operation life of the systems. Significant cost will be added if heat sinks and cooling fans are used in such systems. Therefore, in order for these applications to be viable, low-power design techniques and methodologies are needed.

Image/video processing is one of the power hungry applications that is increasingly in demand for portable systems. With the portable video telephony, internet and entertainment applications in increasing demand, embedded image and video landline becomes an essential feature. Thus, making image and video compression one of the essential enabling technologies for this bandwidth limited world. There are many different image and video compression techniques in existence; one of these techniques that has been increasingly used for image coding is the Discrete Wavelet Transform (DWT). As reasonable operating time due to the high power consumption of image video applications cannot be met by current battery technology, it is important that such an image/video system is low power, so that productive applications and services are viable.

The JPEG2000 lifting based DWT is one such image compression algorithm which is increasingly being used as a core for image and intra-frame coding based video compression applications. Only with a low power JPEG2000 system would many useful applications be possible. Therefore, this thesis sets out to realise a low power JPEG2000 5/3 lifting based DWT hardware architecture. This thesis also investigates whether power reduction techniques applied at both the algorithmic and architectural level would yield a lower power hardware. Besides these, this research also tries to ascertain whether the accumulating Arithmetic Logic Unit (ALU) centric processor architecture is more low power than the feed-through pipelined data-path centric processor. The following sections of this chapter gives details of the motivation, contribution and structure of this thesis. The remainder of this chapter is structured as follows: Section 1.2 presents motivations for this work and Section 1.3 outlines the main contributions of the thesis. Section 1.4 describes the structure of the thesis and the chapter contents. A summary of this chapter is given in Section 1.5.

1.2 Motivations

With the advance in VLSI digital technology, many new high performance and throughput imaging and video applications have emerged and have been increased in use. Demand for image/video applications in portable form has especially increased in recent years. At the core of these productive and useful applications is image/video compression technology. The DWT is one of these algorithms and technologies that has been developed for compression of digital image and (intra-frame coding based) video data.

The DWT is becoming popular because it has features such as progressive image transmission by quality/resolution, and ease of compressed image manipulation. Such features led to significant interest in efficient algorithms for hardware realisation [4] of the DWT. However, conventional convolution based DWT is computationally intensive and both area and power hungry. Fortunately, some of these drawbacks were overcome by using the lifting based scheme for the DWT [5, 6]. As such, the lifting based DWT has been picked for the core DWT algorithm in the recently released JPEG2000 [1] standard. These developments had aroused considerable interest in the implementation [2, 7] of this algorithm. Two types of DWT filters are specified by the JPEG2000 standard, these are the reversible integer 5/3 filter [8] and the irreversible floating point Daubechies 9/7 filters [9]. The reversible integer 5/3 filter is more suitable for low power implementation than the 9/7 filter as the 5/3 filter is inherently low

power and less complex than the 9/7 filter. Therefore, the 5/3 filter is chosen as the candidate for low power implementation. The details of how the 5/3 filter is low power and less complex compare to the 9/7 filter is briefly described in section 4.2.1 in Chapter 4.

Even with the lifting based scheme, there is no low power implementation of the JPEG2000's 5/3 lifting based DWT algorithm. Especially through a reduction of switched capacitance [10] by reducing the number of data-path/arithmetic hardware and computational steps through generic algorithmic techniques. Recent implementation [2, 7] of the JPEG2000's DWT are pipelined data-path centric designs and did not consider the issue of power consumption. Hence, this thesis sets out to realise a low power JPEG2000 5/3 DWT lifting based hardware architecture. The JPEG2000 5/3 DWT lifting based hardware architecture is realised with power reduction techniques at both algorithmic and architectural level. As both algorithmic and architectural level power reduction are inter-related with each other, this thesis also investigates whether a lower power JPEG2000 5/3 DWT lifting based hardware architecture can be realised with both of these levels of power reduction technique.

Traditionally, the feed-through pipelined data-path centric processors are perceived as more power consuming than the ALU centric processor. It is perceived as more power consuming because of the substantial amount of pipeline registers it possesses. Therefore, the aim of this thesis is also to determine whether feed-through pipelined data-path centric processor design is more power consuming than accumulating ALU centric processor design. As such, a novel low power JPEG2000 5/3 lifted based DWT hardware architecture centred around Shifter-Accumulator (SA)-ALU processors was developed and built. The first known power analysis and comparison of the proposed SA-ALU centric JPEG2000 5/3 DWT architecture and existing feed-through pipelined data-path centric architectures was made.

1.3 Contribution

The aim of this thesis is to realise a low power JPEG2000 5/3 lifting based DWT architecture using power reduction technique at the algorithmic and architectural level. As such, this work has led to the development of the novel SA-ALU centric JPEG2000 5/3 lifting based DWT architecture. This architecture is built around Shift-Accumulator Arithmetic Logic Units (ALUs) which allows flexible incorporation of an innovative algorithm and scheduling.

This work also led to the development of the novel embedded data extension algorithm that

can be incorporated into the SA-ALU centric JPEG2000 5/3 lifting based DWT architecture. This novel algorithm eliminates the need of explicitly extending the data as required by the JPEG2000 specification. Thus, saving both hardware resource and power consumption.

Besides this, a novel low power multi-segment sequential addressing one-hot architecture at various address depths with 5-levels of dyadic structure has been developed. This architecture together with the shared address lines structure resulted in significant reduction of power consumption.

Finally, the proposed SA-ALU centric architectures together with the modified Lian et. al's [7] and the modified Andra et. al's [2] architecture were implemented. The power consumption of these architectures were evaluated and compared with one another. This comparison, of the power consumption of these architectures, is the first known comparison.

1.4 Thesis Structure

This thesis is divided into eight chapters with this chapter as the first chapter, the remaining structure of this thesis are organised as follows:

- **Chapter 2** gives an overview of low power engineering research. The mechanics of how and where power is dissipated in digital CMOS circuits are described. Power reduction strategies and techniques researched and developed in the past few decades are presented along with the concept of power reduction at the different levels of design abstraction. The implementation and power evaluation flow are also presented.

- **Chapter 3** gives a brief overview of common image and video compression technique with attention focussed on the discrete wavelet transform (DWT) and its lifting based scheme. The international standards for image and video compression are presented. The need to have low power image compression techniques to make image and video applications viable is also shown.

- **Chapter 4** details the overall structure of the proposed novel SA-ALU centric JPEG2000's 5/3 lifting based DWT hardware architecture and the architecture of each SA-ALU processor. Along with it, two of the existing feed-through pipelined data-path centric JPEG2000 5/3 lifting based DWT architectures (Andra's and Lian's architecture) are presented. In addition, the operation schedule and memory access scheme of the

proposed SA-ALU centric architecture are presented. Finally, a hardware architecture comparison in terms of the number of data-path units of the three JPEG2000 5/3 lifting based DWT architectures is made and presented.

- **Chapter 5** presents the proposed novel embedded data extension algorithm for the JPEG2000 5/3 lifted based DWT. It also introduces the topic of distortion or artefacts at the edge of a multi-resolution sub-band coded image and briefly discusses the need of using data extension to reduce and eliminate such artefacts. Different types of symmetric data extension along with the data extension requirement of the JPEG2000 specification are presented. Finally, the power evaluations of a hardware performing explicit data extension together with the computational steps saving resulted by using the embedded data extension are also presented.

- **Chapter 6** gives the detailed architecture of the modified version of Andra's and Lian's feed-through pipelined data-path centric processor architecture. Power analyses and comparisons of Andra's, Lian's and the proposed SA-ALU column processors along with the discrepancy of different evaluation results are presented and discussed. A new set of criteria for power consumption evaluation are proposed as a consequence of the analysis of the power evaluation results.

- **Chapter 7** presents the novel scalable low power sequential multi-segments multi-level one-hot addressing architecture (MSML-OHA) and its application extension Dyadic MSML-OHA (DMSML-OHA). The description of both comparing architectures; the conventional counter-decoder architecture (CCD) and the one-hot addressing architecture (OHA) are also presented. Following these, the power evaluation results of MSML-OHA and DMSML-OHA are presented, analysed and compared with those of OHA. Finally, a discussion on why not all of the configurations of the MSML-OHA are power efficient is made.

- **Chapter 8** presents the summaries of all the findings of the contributing chapters, reiterates the main contributions and gives a summary of this thesis. And finally, give suggestions of topics for future research and development.

1.5 Summary

This thesis sets out to investigate whether a low power JPEG2000 5/3 Discrete Wavelet Transform (DWT) lifting based hardware architecture can be realised with low power techniques at both the algorithmic and the architectural level. Besides that, the aim is also to determine whether a feed-through pipelined data-path centric processor design is more power consuming than an accumulating ALU centric processor. The SA-ALU centric JPEG2000 5/3 DWT architecture with power reduction at both algorithmic and architectural level was designed and implemented for this purpose. The power consumption figures of the proposed SA-ALU centric JPEG2000 5/3 DWT architecture were compared with existing conventional architectures.

This thesis makes a number of contributions to the Low Power Design and Engineering research fields. Empirical results are presented to confirm whether the utilization of both inter-related algorithmic and architectural level power reduction techniques will result in a lower power hardware. It also establishes whether an accumulating ALU centric processor architecture is inherently less power consuming.

Chapter 2

Low Power Digital CMOS System and Circuit Design

2.1 Introduction

The ever-increasing computational demands of modern day applications has driven up the power dissipation of modern digital electronics systems. The increase in power consumption of such systems has created a few concerns. One of the major concerns is the operating time of these systems, as reasonable operating time cannot be met by current battery technology. The other concern is the heat dissipation issue of such systems. Heat generated by such systems can cause the system to overheat. Overheating can reduce throughput, reliability and the operational life of the system. Therefore, heat generated by these systems has to be removed by heat sink and cooling fans. However, cost will be added if heat sinks and cooling fans are used in such system. Thus, design engineers are forced to look into other ways of overcoming this power issue. One broad approach which engineers take, is to minimize power consumption in a specific digital electronics system without compromising its high performance and throughput. This way of tackling the escalating power consumption issue has been quite effective in lowering the power consumption of these systems. As such, many have looked into this area in the attempt to make computationally intensive systems (especially portable ones) viable.

This chapter presents an overview of the power reduction strategies and techniques researched and developed general and in the field of Discrete Wavelet Transform (DWT) implementation in the past few decades. This chapter also introduces the mechanics of how and where power is dissipated in digital CMOS circuit. The concept of power reduction at the different levels of design abstractions along with the limits and shortcomings of existing techniques are also presented. Finally, the workings and limitations of existing power evaluation tools are presented.

This chapter is organised as follows. In Section 2.2, the power consumption sources of a Complementary MOSFET (CMOS) circuit are presented. Following that, the power reduction strategies and power reduction techniques at the various design abstraction and the existing low power research work on the DWT are presented in Section 2.3. The limits and shortcomings of existing techniques is also briefly discussed in Section 2.3. Next, the design implementation and power evaluation flow used in this research and the limitation of existing evaluation tool are described in Section 2.4. Finally, the summary of this chapter is given in Section 2.5.

2.2 Power Consumption Sources of Complementary MOSFET (CMOS) Circuit

The Complementary Metal Oxide Semiconductor Field-Effect Transistor (MOSFET) logic circuits are widely used to build a digital system because it is power-efficient. It also has lesser transistors per logic gate and is easier to fabricate compare to bipolar logic (like Transistor-Transistor Logic (TTL) and Emitter-Coupled Logic (ECL)). A typical Complementary MOSFET (CMOS) circuit would normally comprised of two groups of elementary logic switching MOSFET transistor as shown in Figure 2.1.

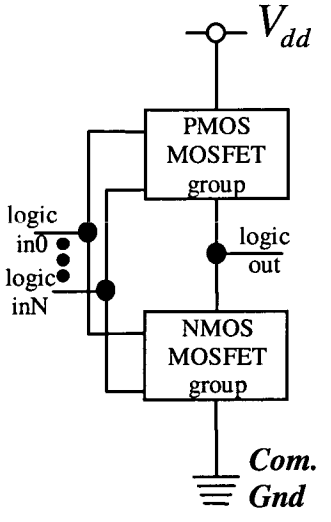
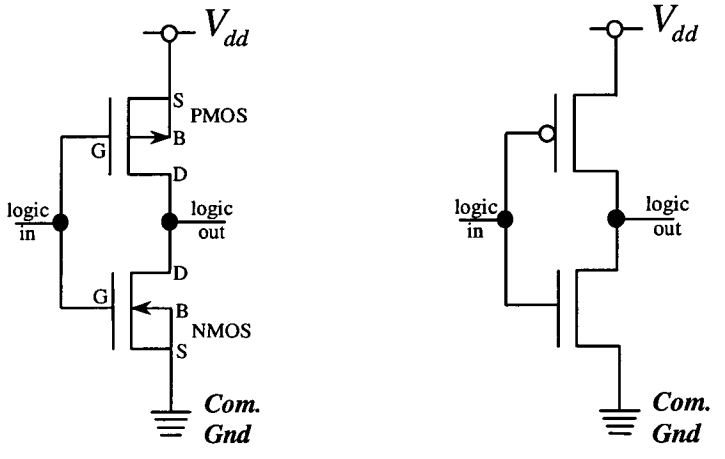


Figure 2.1: *Generic make-up of a CMOS logic gate circuit*

The group of the MOSFETs at the north part of the circuit connecting to the positive rail are P-channel Metal Oxide Semiconductor Field-Effect Transistors (PMOSs). At the south part of the circuit connecting to the common ground or negative rail are N-channel Metal Oxide

Semiconductor Field-Effect Transistors (NMOSs). The PMOSs are turned on with a voltage value close to the negative rail or logic '0', whereas the NMOSs are turned on with a voltage value close to the positive rail or logic '1'. An actual CMOS inverter circuit are shown in Figure 2.2.



(a) CMOS inverter circuit notation

(b) Simplified CMOS inverter circuit notation

Figure 2.2: *Diagrams of CMOS inverter circuit*

As the only one (or group of) MOSFET is turned on at one time when a stable logic level is maintained at the input, there is no direct current path from the positive rail to ground (or negative rail). Therefore, the CMOS digital circuits are inherently low power. Even though the CMOS circuits are inherently low power, digital system using CMOS can be engineered to consume even lower power. In order to reduce the power consumption of the CMOS-based system, understanding where and how a typical CMOS circuit dissipates its power is important. The power consumption sources are generally categorised into two type of power dissipation; the static power dissipation and dynamic power dissipation. The power dissipation of a typical digital CMOS circuit is described by the following equations:

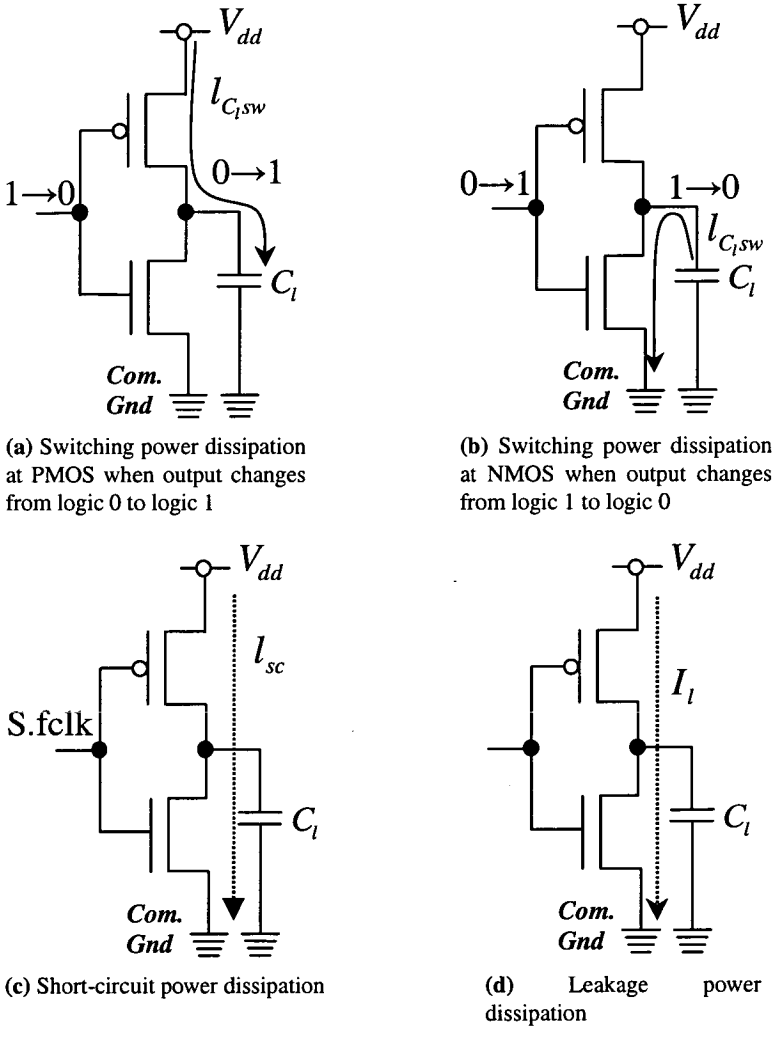
$$P_{ave} = P_{sw} + P_{sc} + P_l \tag{2.1}$$

$$= I_{C_l sw}.V_{dd} + I_{sc}.V_{dd} + I_l.V_{dd} \tag{2.2}$$

P_{ave} of Equation 2.1 denotes the average power dissipated by a CMOS circuit. The three major

power dissipation components of a CMOS circuit are the load-capacitance switching power P_{sw}, the short-circuit power P_{sc} and the static-leakage power P_l.

All these component of power dissipation are silicon fabrication technology dependent. However, both switching power P_{sw} and short-circuit power P_{sc} are especially dependent on switching activities which is subjected to the design of the digital system. The following figures of a CMOS inverter circuit serve to further illustrate these three main power dissipation components of a CMOS circuit.



(a) Switching power dissipation at PMOS when output changes from logic 0 to logic 1

(b) Switching power dissipation at NMOS when output changes from logic 1 to logic 0

(c) Short-circuit power dissipation

(d) Leakage power dissipation

Figure 2.3: *Diagrams of CMOS inverter circuit power dissipation*

$I_{C_l sw}$ in Figure 2.3(a) and 2.3(b) is the current needed to charge or discharge the load capacitance when the output logic changes from logic '0' to logic '1' or from logic '1' to logic

'0'. The I_{sc} in Figure 2.3(c) denotes the short-circuit current and the I_l in Figure 2.3(d) denotes the leakage current. Majority of the power is dissipated across both the PMOS and NMOS when these currents flow through them. These above-mentioned two main type of power dissipation components will be described in greater detail in the following two sub-sections.

2.2.1 Static Power Dissipation

Ideally, there should not be any static power dissipation in a CMOS circuit as there is no direct path from the power rail to ground. However, there are no perfect MOSFETs in real life. The imperfection of the transistors is the main cause of the static power dissipation. As the characteristics of the transistors depends on the device and fabrication technology, the static-leakage power P_l is therefore greatly effected by these.

$$P_l = I_l.V_{dd} \tag{2.3}$$

The static-leakage power shown in Equation 2.3 is calculated by multiplying the total leakage current I_l by the rail supply voltage V_{dd}. The total leakage current I_l consists of the lump sum of all the leakage currents in the complementary circuits as shown in Equation 2.4.

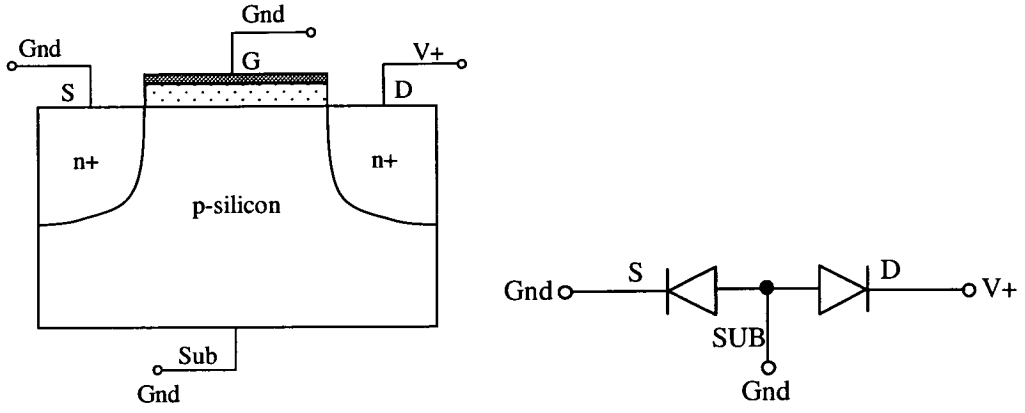
$$I_l = I_{dsb} + I_{dr} \tag{2.4}$$

These leakage currents are namely drain-to-substrate leakage current I_{dsb} and drain leakage current I_{dr}.

2.2.1.1 Drain-To-Substrate Leakage

The drain to substrate leakage occurs when the transistor is turned off. As the substrate node is always connected to the source node, it is always connected to the supply voltage for PMOS and to ground for NMOS in a CMOS circuit. When the transistor is turned off, it sets up a reverse biased diode in the transistor between the drain and the substrate. In the case of the NMOS (as shown in Figure 2.4), the voltage at the N-type drain is equal to V_{dd} (or $V+$) and

the P-type substrate voltage is $0V$.



(a) Voltage values at the terminals of a turned-off NMOS

(b) Electronics schematic equivalent representation

Figure 2.4: *Voltage values at the terminals of a turned-off NMOS*

Equation 2.5 describes the reverse bias leakage current (or drain-to-substrate leakage current). I_{rs} in Equation 2.5 denotes the reverse saturation current which flows from the drain to the substrate. V_{rs} represents the voltage across the drain and the substrate.

$$I_{dsb} = I_{rs}(1 - e^{\frac{-qV_{rs}}{kT}}) \tag{2.5}$$

The reverse saturation current I_{rs} is dependent on temperature T as shown in the following equation.

$$I_{rs} = P.T^3(e^{\frac{-E_{0^o}}{kT}}) \tag{2.6}$$

The P term in Equation 2.6 is a constant associated with devices and fabrication design and setting. E_{0^o} in the equation denotes the band-gap energy of the semiconductor at 0^o Kevin.

2.2.1.2 Drain Leakage

The drain leakage occurs when the MOSFET in CMOS circuits is weakly turned on or not completely turned off. There are two main current leakage contributions (see Equation 2.7) to

the drain leakage current I_{dr}. They are the reduced voltage feed leakage current I_{rvf} and the sub-threshold leakage current I_{sbth}.

$$I_{dr} = I_{rvf} + I_{sbth} \tag{2.7}$$

Reduced Voltage Feed Leakage

Reduced voltage feed leakage occurs when the driving logic signal at the input of the gate is short of the transistor's turn-off voltage. This arises when the original driving signal is connected via path that (for example a transmission gate) causes a reduction in voltage level and presenting a voltage close to the threshold voltage of the PMOS. This reduced voltage not only turns on the NMOS, but it also faintly turns on the PMOS, thus, conducting leakage current I_{rvf} down to ground.

Sub-Threshold Leakage

The other drain leakage comes from the sub-threshold leakage current I_{sbth}. The transistor continues to conduct even though it is supposed to turned off when the voltage at the gate is below its threshold voltage. The sub-threshold current is one of the inherent characteristic of a MOSFET. This sub-threshold current is exponentially proportional to the gate-source voltage V_{gs}. This current consists of mainly diffusion current rather than drift current and is temperature dependent.

$$\begin{aligned} I_{sbth} &= \frac{q.W.x_{ac}.D_n.n_{po}}{Lc}.(e^{\frac{q\psi_s}{kT}}).(1 - e^{\frac{-qV_{ds}}{kT}}) \\ &\approx \frac{q.W.x_{ac}.D_n.n_{po}}{Lc}.(e^{\frac{q\psi_s}{kT}}) \\ &= \frac{q.W.x_{ac}.D_n.n_{po}}{Lc}.(e^{\frac{q(V_G - V_T)}{kT}}) \end{aligned} \tag{2.8}$$

The W, x_{ac}, D_n, n_{po}, V_G, V_T and Lc in Equation 2.8 denotes the width of the MOSFET, the channel thickness, the diffusion coefficient for the electrons, the electron concentration in p-type silicon at thermal equilibrium, the gate voltage, the threshold voltage and the effective

channel length respectively. As can be observed, the sub-threshold leakage current can be minimized by optimising the dimension and the amount of doping for the device.

2.2.2 Dynamic Power Dissipation

Even though static power dissipation is one of the main power dissipation of CMOS circuit, from Equation 2.2 it can be seen that two out of three main contributors of power dissipation comes from dynamic power dissipation. Therefore, understanding and minimizing the dynamic power dissipation will help to reduce the overall power consumption of a CMOS system. The following sub-sections describe the two dynamic power dissipation component; the Short-Circuit Power Dissipation and the Load-Capacitance Switching Power Dissipation.

2.2.2.1 Short-Circuit Power

The following Equation 2.9 describes the short-circuit power dissipation. The short-circuit power dissipation P_{sc} is due to the conducting direct-path short circuit current I_{sc}, when the CMOS gates output switches logic state.

$$P_{sc} = I_{sc}.V_{dd} \tag{2.9}$$

As the input of the CMOS gate switches from one logic state to another, the input voltage raises or falls passing the turn-on threshold voltage of both group of PMOS and NMOS in the CMOS gate circuit. Hence, turning both the NMOS and PMOS transistors simultaneously active and briefly conducting current directly from supply to ground.

Although the instantiate short-circuit current may be substantially large, the flow of current only last for an extremely short moment. As such, a single instance of short-circuit power dissipation is small. However, with frequent switching, this short-circuit power can add up to become very significant. The short-circuit power dissipation can be minimized by designing the threshold voltages of both PMOS and NMOS to be suitably high. This is so that both PMOS and NMOS would not turn-on simultaneously or the period of the turn-on would be even shorter. However, by doing these, the switching speed of the CMOS circuit will be reduced. Another way of reducing the short-circuit power dissipation is by reducing the switching activities at the input

CMOS logic gate circuit. As the short-circuit power dissipation is dependent on switching activities, Equation 2.9 can be re-expressed to account for this relationship in Equation 2.10.

$$P_{sco} = I_{sco}.V_{dd} \tag{2.10}$$

The P_{sco} represents the overall average system short-circuit power dissipation for a computation process over a period of time. The overall average system short-circuit current I_{sco} can be expressed to reveal the direct relationship with switching activities as shown in the following equation.

$$I_{sco} = S.f_{clk}.I_{sca} \tag{2.11}$$

The f_{clk} term in the equation is the clock frequency and the S term, is the transition or switching activity factor. The switching activity factor S is the average number of times the node makes a power consuming ($0 \rightarrow 1$ or $1 \rightarrow 0$) transition in one clock period. The switching activity factor S can also be appreciate as the probability of a switching event taking place in a clock period. This probability can be calculated by the number of switching activities of a node observed over a period of time. I_{sca} is the maximum instantiate short-circuit current possible in a CMOS system.

2.2.2.2 Load-Capacitance Switching Power

Load-capacitance switching power is the other dynamic power that accounts for a substantial amount of a CMOS system's power consumption. The switching power is the power associated with the load current charging and discharging the load capacitance C_l. The following equation describes the switching power.

$$P_{sw} = S.f_{clk}.I_{C_l sw}.V_{dd} \tag{2.12}$$

$$= S.f_{clk}.C_l.V_{dd}^2 \tag{2.13}$$

P_{sw} represents the load-capacitance switching power, where I_{Clsw} is the load-capacitance switching current, C_l is the load capacitance, f_{clk} is the clock frequency and S, is the transition or switching activity factor which is described in Section 2.2.2.1. V_{dd} is the voltage swing which is (in most cases) equal to the supply voltage. The load capacitance C_l which makes up mainly of the MOSFET gate capacitance and wire capacitance. These load capacitances are mostly parasitic capacitance with some exceptional cases of intended capacitance.

2.3 Low Power Engineering

Low-power electronics engineering is not a radically new area of research. In recent years, demands for high performance, small geometries, large chip sizes and yet low-power devices for battery-operated portable applications has increased. Digital signal processing (DSP) is one of the essential core enabling technologies of many modern digital applications. Current DSPs require a substantial amount of power to perform complex and computationally intensive signal processing tasks at speeds suitable for real-time operation. Such high power consumption limits their use in portable devices. Therefore, research into low-power techniques in this important area is especially necessary for the portable applications to be viable.

Although it can be seen from Section 2.2 that there are ways where a CMOS circuit can dissipate its power, there are also many ways and approaches to reduce such power dissipations. The following sub-sections reviews all of these power reduction techniques and approaches. Along with it, the limitation of such techniques will also be presented in the later part of this section.

2.3.1 Low Power Design Strategies and Techniques

There are many different existing low power design approaches and techniques. These techniques tackles power consumption issues on different fronts by minimizing the key power consuming factors discussed in Section 2.2. However, only some of these factors are effective in reducing the overall power consumption of the CMOS system.

The static power dissipation is historically very small compare to dynamic power dissipation. Therefore, minimizing the load-capacitance switching power P_{sw} is one of main focuses of most low power research in the past years. However, in the 2003 edition of International

Technology Roadmap for Semiconductors [11] (ITRS03) the static power dissipation had been projected to increase to the same order as the dynamic power dissipation in next next years times as the geometric size of the MOSFET in a CMOS circuit continues to scale down.

In the ITRS03, two low power CMOS logic technology categories, Low Operating Power (LOP) and Low Standby Power (LSTP) logic, were defined. The LOP logic is for systems such as a notebook computer which is relatively high-performance application and its battery is likely to be high capacity and the design focus is on reducing its operating power. The LSTP logic is for systems with relatively lower performance like mobile phones and personal digital assistants (PDA) where their battery has lower capacity and focus is get the static power dissipation to be as low as possible. By year 2018, the LOP logic's static power dissipation will surpass the dynamic power dissipation by about 5% [11]. Therefore, substantial work had gone into reducing the static power dissipation.

Although, the LOP logic's static power dissipation will surpass the dynamic power dissipation in ten years time, the LSTP logic's static power dissipation will remain low (about 1.8% of the total power consumed in year 2018) [11] compared to dynamic power dissipation. As such, the dynamic power dissipation is still consider as the dominating power contributor to the overall power consumption of a low power CMOS circuits for now and some years to come, and reducing the amount of switching power dissipation is still one of the most effective ways of reducing the overall power consumption of the CMOS system.

Although the switching power dissipation is the main contributing power consumption in a CMOS system, there are many ways to optimise and reduce the switching power dissipation. From Equation 2.13 in Section 2.2.2.2, it can be seen that there are about 4 degrees of freedom to reduce the load-capacitance switching power P_{sw}. The 4 terms in Equation 2.13 which can be minimize are the supply voltage V_{dd}, the switching activity S, the clock frequency f_{clk} and the load capacitance C_l.

One of the key to reduce the load-capacitance switching power P_{sw} significantly is to reduce the supply voltage V_{dd}. This is due to the fact that the switching power dissipation of a CMOS circuit is quadratically proportional to its supply voltage V_{dd}. The another important factor in power reduction, is the reduction of the switching activities S and the clock frequency f_{clk}. By targeting and reducing the switching activity S and the clock frequency f_{clk}, the power dissipation by short-circuit power P_{sc} will also be reduced. Finally, reducing the load

capacitance C_l has direct effect on reducing the switching power consumption.

The power reduction strategies of the above-mentioned four key factors of switching power and the static-leakage power (described in Section 2.2) manifests itself in many different forms and at all levels of the design abstraction hierarchy.

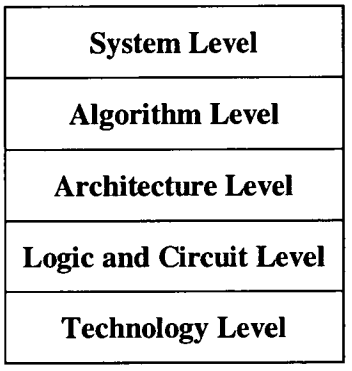


Figure 2.5: *Power reduction design abstraction hierarchy*

There are five different yet somewhat interdependent hierarchical design abstraction levels. These five design abstraction levels (from the top) are the System level, the Algorithm level, the Architecture level, the Logic and Circuit level and finally the Technology level (as shown in Figure 2.5). Work on these different power reduction strategies which are categorised in these design levels are reviewed in the following.

2.3.1.1 System Level

Work at the system level focuses on reducing the power consumption at a larger scale of manipulation which affects major part of the whole system. Techniques developed by such works can be categorised by the following areas;

- **Power Management**
- **Low Power Resource Assignment and Task Scheduling**
- **Code Traffic Reduction**
- **Software Instruction Design and Selection**
- **Low Power Hardware Synthesis**

Works in areas are reviewed in the following: -

Power Management

Power management reduces power consumption of a digital CMOS system by turning off or scaling down the supply voltage or clock frequency when the system is idling or the system computation needs are reduced. The two main schemes/strategies [12] which are used in power management are:-

- Activity-based system **shutdown** for event-driven applications
- **Variable supply voltage and clock frequency** for continuous data-flow, real-time applications such as a DSP applications.

Shutdown

Within the shutdown scheme itself, it can be further categorised into two main approaches [13]. These two main approaches are called the *Static and the Dynamic Power Management.*

Static Power Management

A static power management system [14–16] defines one or several "sleep" modes with various levels of power saving and delay overhead costing. These "sleep" modes can be controlled externally by the user via software. The system will enter into and exit from such modes by user activation.

Dynamic Power Management

A dynamic power management system uses schemes which automatically detect idle periods so that it can disable its clocks or powering down (by means of clock gating (see Section 2.3.1.4)) portions of the systems. There are 3 shutdown approaches to dynamic power management.

The first approach is called the **conventional straight-forward** shutdown [15, 17, 18]. This approach switches the whole system into one of the power saving modes or shuts it down whenever a predefined period of idle time or an end of a computation event is detected. A partial system shutdown technique has also been developed and used in a DSP system. Ludwig et al. [19] developed a technique that shutdowns part of a DSP (FIR and cascaded IIR) system

according to the input signal spectrum, characteristics and statistics. Another example of a similar technique is presented in [20].

The second approach is the **predictive** shutdown [12, 13, 21–27] which tries to predict the length of idle time to decide a shutdown and when to exit from a shutdown. The length of idle time is predicted using the system's computation process time period history and its previously predicted idle time. Besides qualifying a shutdown, the shutdown period is also predicted with the consideration of cost (in terms of both power & system responsiveness). Srivastava et al. [12] developed two predictive algorithms. The first algorithm is based on regression analysis. It uses a quadratic model obtained from the results of correlating the length of the upcoming off (idle) period with the previous on and off (idle) periods. The second algorithm uses a simple filter to filter out the idle periods that fulfil the conditions for a shutdown. Hwang et al. [13] proposed an exponential-average algorithm predictive shutdown system with "pre-wake up" state for fast wake up response. The exponential-average algorithm is based on the statistics of the system. This algorithm uses a number of formulae to determine the threshold (the shortest) idle period for a shutdown, to predict the length of the idle period and to qualify a corrective shutdown if the subsequent predicted idle period is shorter than the threshold idle period and actual idle period. Ramanathan et al. [21] proposed an algorithm similar to the exponential-average but which is relatively simpler to implement than the sets of Hwang's algorithms. The proposed algorithm is called the competitive adaptive algorithm.

The third approach [22–25] is to use some form of **stochastic/statistics models** to model the system and then makes the decisions (or calculate the Optimal Policy (PO)) to switch between different power modes of the system. Paleologo et al. [22] use a discrete-time Markov decision process model, whereas Qiu et al. [23, 24] use a continuous Markov decision process model. In [25], Qiu et al. use Generalised Stochastic Petri Nets (GSPN) first to model the system then convert it to a continuous Markov decision process model to solve for the optimal policy as a linear programming problem.

Variable Supply Voltage and Clock Frequency

A technique that has been developed which is particularly useful for a DSP system is the variable supply voltage and clock frequency technique. In a continuous real-time systems/application there are usually no periods of inactivity. However, these systems do have some periods when computation requirement are lighter. The lightening of workload is due to

a changes in its input signal data characteristics/statistics.

Gutnik et al. [28] proposed a system that varies both the supply voltage and clock frequency with an embedded power supply and clock generator respectively according to the requirements of the incoming data. Embedded variable voltage supply design for such techniques were proposed by Goodman et al. [29] and Kuroda et al. [30]. Hong et al. [31] developed a variable voltage supply scheduling algorithm to schedule the voltage level according to its workload and its task completion time. Systems with dual variable voltage supplies were also proposed by Kuroda et al. [32, 33] and Usami et al. [34, 35]. Lee et al. [36] proposed a multiple variable voltage supplies system of interacting modules (e.g. Intellectual Properties (IPs) on a SOC platform) and voltage scheduling for such a system. Qu et al. [37] presented a similar system. Chang et al. [38] proposed a scheme to schedule and assign the multiple voltage supplies to different parts of the system according to the speed required.

Low Power Resource Assignment and Task Scheduling

Lu et al. [26] looked at facilitating system level power management in the system's task scheduler or kernel. Lu et al. proposed a greedy on-line scheduling algorithm for a kernel that is used in a system with multiple devices. This kernel schedule tasks in such a way that enables a device to shutdown for a longer period and minimize the number of shutdowns by clustering idle periods together so that an idle period is longer.

Benini et al. [27] proposed a kernel-based power optimised finite-state machine (FSM) based sequential or control logic. This technique reduces power by extracting high probability states of the logic block and implementing it efficiently in the computational kernel. The computational kernel is a FSM of a sequential circuit that mimics the steady-state behaviour of the original network of logic blocks.

Code Traffic Reduction

System level code traffic reduction techniques have been developed to reduce the overall system power consumption in [39]. Lekatsas et al. proposed a random access code compression together with a modified computer architecture to reduce the code transfer overhead from the main memory to the system processor via the cache.

Software Instruction Design and Selection

Su et al. [40] and Tiwari et al. [41] look into generating low power software assembly code through a specially designed software compiler for a general purpose DSP or processor. Gebotys et al. [42] uses a similar technique for generating low power assembly code for Very Long Instruction Word (VLIW) DSPs. All this work involves the scheduling or ordering of the assembly instructions to minimize the switching activities of adjacent software codes through the compilation of high-level software code by the compiler specialised in power optimisation.

Lee et al. [43] and Mehta et al. [44] also proposed a similar technique. Lee et al. [43] took into account the allocation variables in different blocks of memories of different access buses, whereas Mehta et al. [44] took into consideration the ordering of the instructions using CPU registers. Lee et al. [45] also investigated into reducing power consumption through the design of packed software instructions. Snyder et al. [46] proposal a compact software kernel that allows low power compact software code to be written, compiled and run on it.

Low Power Hardware Synthesis

Low power hardware synthesis is another area where low power DSP hardware can be synthesized with the requirements and constraints given by the designer. Such a synthesis engine usually utilizes Artificial Intelligence (AI) to generate low power hardware. Works presented in [47, 48] uses Genetic Algorithms (GA) to do just that.

2.3.1.2 Algorithm Level

At the algorithm level, work usually involves optimising a power and computationally dominating algorithm to reduce its computational steps and switching activities so that the power consumed by the algorithm can be reduced.

Transforming or rearranging the original algorithm to another form is one of the ways to reduce the system's power consumption. Many techniques have been developed based on such concepts and reordering is one of them. **Reordering** of the data and coefficients of a Discrete Cosine Transform (DCT) in [10, 49] has shown significant reduction in computation switching resulting in reduction of power consumption.

Block processing of a filtering operation described in [50, 51] reduces memory accesses

and switching activities in multiplier by grouping a set of data and coefficient together for processing. **Coefficient segmentation** [51, 52] proposed by Erdogan et al. reduces the number of significant bits of the coefficients that go through the multiplying process by utilising a shifter.

Lidsky et al. [53] derived a **tree search algorithm** for Vector Quantisation (VQ) which reduces the computational complexity of N times (of a full search) to $Log_2 N$ times. In another development on VQ coding, Masselos et al. [54] developed a **transformation**, which involved luminance shift and isometries. This transformation makes the resulting code-book smaller which in turn reduces the extent of the code search and thus saves power. Chan et al. [55] developed an algorithm that implements the 2-D DCT directly reducing computation steps and therefore power.

Look-ahead [56, 57] is a transformation which has been developed to allow easier implementation of a recurrence algorithm in a parallel structure. **Unfolding** [57] is another transformation that allows an easier extraction of concurrency of an algorithm so that it can be implemented in parallel. Other algorithm transformations like **associativity, commutativity** and **inverse element law** were described in [58]. With these, Potkonjak et al. went on to propose the **algorithm selection** [59] which evaluates different types of low power algorithm (or a combination of them) for a certain application and chooses the best one (or combination) for the application.

2.3.1.3 Architecture Level

One effective way of reducing the power consumption of a system is to tackle it at the architecture level. Most work at the architecture level reduces power consumption by reducing unnecessary switching activities and load capacitance brought about by the structure of the architecture.

One effective way of reducing the load capacitance is integrating a large system into a system chip like System-On-a-Chip (SOC) [60, 61] or Multi-Module Chip (MMC) designs. Therefore, numerous work has been focused on designing **low power bus architectures** that would allow seamless and low power integration of Intellectual Property (IP) modules together. A bus splitting architecture [61] for SOC platform proposed by Hsieh et al. is one such example.

Other similar work was done involving developing a coding technique and bus architecture for

reducing switching activity on buses. Such techniques are called **Bus Coding**. There are two main branches of bus coding, one of these is for data bus and the other one is for bus addressing. Bus-invert [62] coding proposed by Stan et al is one of the first bus-activity reduction codings for **data bus**. The data on the bus is inverted if more than half of the data bits on the bus are inverted with respect to the previous data. An extra line is added to indicate to the decoder whether or not a data inversion had taken place. Extending this technique, Shin et al. [63] applied the bus-invert coding only on the part of the bus that exhibits a high probability of transition. Shin et al. went on to develop two more coding techniques [64] which are called the Bus-invert/Transition Signalling (BITS) coding and Advance Bus-invert/Transition Signalling coding (ABITS). Besides the bus-invert coding, Stan et al. also proposed a low-power bus coding [65] for terminated buses. A set of algorithms and a general architecture framework were proposed by Sotiriadis et al. [66] which generalises spatial redundant low power bus coding system. This spatial redundant bus coding system in a special case converges into the bus-invert coding. Another type of coding which is not based on bus data inversion was proposed by Benini et al. [67]. This coding is based on a-priori knowledge of the data on the bus. A theoretic lower bound of bus-transition has been shown by Ramprasad et al. [68]. With this theoretic model and calculation, an encoder-decoder framework for bus coding was developed.

Bus coding targeted for **address bus** were also developed. Gray coding [40, 69, 70] was proposed to reduce the Hamming distance of adjacent sequential address on the address bus. 'The Bench Coding' [71] and Cold scheduling [40] using a heuristic approach to reduce the Hamming distance of the address data on the address bus to reduce switching activities. Other address bus codings based on bus data inversion and difference coding were developed. Extended transition activity measure (ETAM++) [72] extends the bus-invert to include transition activity of adjacent address buses. Self-organizing list encoding [73] and Address Level Bus Power Optimisation (ALBORZ) [74] is based on difference coding where a reduced Hamming distance of the difference in the address data is sent across the bus.

Bus coding is also extended to the memory access buses. Work in [75] extended bus-invert coding into two memory data bus coding which takes into account the end-termination network. Chang et al. [76] continued to develop inverted data for low power storage and access in pre-charging DRAM. In another development, Panda et al. [70] developed power optimised 2-dimensional (2-D) row-major, column-major and tile-based address mapping with Gray

coding. Coumeri et al. [77] proposed a method to minimize the power of memory architecture by finding the best combination of the following two configuration; Wide configuration and Segmented configuration. Kapoor et al. [78] investigated the relationship of the cache sizes of different bytewidths with the power consumption. Similar work was done by Brockmeyer et al. [79] which partitioned memories into structure similar to a cache and powering down part of them when it is not accessed.

Asynchronous architecture [80] is one of the ways to reduce the switching activities of a CMOS system by inherently self clock-gating. Asynchronous systems get rid of the need of a reference clock and only computes when the present state is different from the previous state. Jou et al. [81] worked on a similar architecture but by proposing an architecture that is Globally Asynchronous Locally Synchronous (GALS). And this is done with self-timed circuits. Hemani et al. [82] extend the GALS idea to a higher (at the system) level by including an automated design methodology for such system. Self-timed [83] circuits were also used in memory circuits to reduce glitches on high capacitance buses.

Numerical data representation is one of the factors that will influence the both the architecture and the power consumption of DSP system. By optimising the numerical data representation power consumption of the DSP system can be reduced. The following work optimised the data representations to avoid redundant computation and switching in the system caused by signed extension. Khoo et al. [84] proposed a biased two's (2's) complement representation by shifting the whole numerical range of the data up to the positive range by a fixed amount and subtracting the same amount back again after processing. Truncation of the most significant bits (MSB) of the sign extension of the negative 2's complement number in subtraction operation was proposed by Moshnyaga [85]. The number of bits for truncation depends on the numerical need of both the two data operands. Slicing and tagging each of the 2's complement numbers according to their actual arithmetic needs was proposed by Nielsen et al. in [80]. Signed magnitude representation number was proposed in [86–88] to eliminate the switching activities of sign extension altogether during addition and subtraction. However, there is a trade-off of area for power reduction, as two additional adders are needed to perform a normal addition/subtraction operation. In [89, 90] an arithmetic technique, termed as the distributed arithmetic, was presented. This arithmetic technique allows a result to be computed to variable accuracy and resolution according to the actual need.

Pipelining and parallel implementation increases the throughput of a computing system

as the supply voltage decreases. Pipelining [91] is a technique that increases throughput of a system by inserting registers in between combination logic blocks. *Retiming* [91] is the technique of reordering registers of a pipelined system so that it will result in higher speed performance and reduced power consumption.

Parallel architecture is another architecture that is closely related to its implemented algorithm. Such architectures which duplicate a single basic processing unit into multiple units to operate together in parallel are presented in [16, 17, 28, 55, 88, 90, 92–95]. Parallelism not only reduces clock frequency but at the same time increases the throughput and increases hardware area as well.

Low power **Arithmetic Logic Unit** (ALU) is one of the key portions of a computing system that affects the speed and power consumption of its operations. By optimising the ALU architecture, substantial amount of power can be reduced. As such, Sakuta et al. [96] proposed a delay-balanced multiplier that reduces power by reducing glitches within the multiplier. A low power single bit adder with bypass function was also proposed in [17] by Garrett et al.

Optimisation of a cache-like codebook search architecture of a Variable Length Decoder (VLD) was proposed by Cho et al. [97] and Lee et al. [98].

2.3.1.4 Logic and Circuit Level

On the logic and circuit physical design level, most work has been done on designing logic circuit configurations that have not only lower power consumption at both normal supply voltage level and very low supply voltage level, but have also improved the speed performance and reduced area.

Most work on **logic designs** focuses research on reducing power consumption by reducing both CMOS's (short-circuit) switching activities and leakage current. The Complementary Pass-gate Logic (CPL) presented in [92, 99] is one such example. The CPL reduces power consumption by reducing capacitance due to the PMOS network and reduces short-circuit current paths. Besides that, another logic configuration design which is built on conventional CMOS logic was also proposed. The F-Gate [100, 101] and the Super cut-off CMOS (SCCMOS) [102] are one such example. F-Gate is a glitch power minimization technique that uses gate freezing to eliminate glitches in a CMOS system. SCCMOS incorporated a virtual supply voltage system for low voltage supply CMOS circuit that will fully cut off leakage current in static standby

condition. **Adiabatic** Differential Cascode Voltage Switch Logic (ADCVSL) [103] reduces power consumption by reusing the supplied energy within itself. The ADCVSL is member of the dynamic CMOS logic configuration family.

Another **non-CMOS** based low power digital logic circuit family has also been researched and developed. One such logic family is the Monostable-bistable transition logic element (MOBILE) [104]. MOBILE is a logic element made up of Resonant Tunnel Diodes (RTD) connected serially with a Heterostructure FET (HFET), which can operate at a very low voltage of 0.8v with very good noise margin of 0.6v. Besides this, **customised** design of low-power logic circuits for a specific application like embedded 3 transistor DRAM for video application [105] was also be developed. Other customised low power logic circuit designs can be found in [89].

Clock gating and **gated clock** [15, 18, 29, 49, 83, 86, 95, 105–110] is another technique used at the logic circuit level, to reduce power by gating the clock to part of the circuit when it is needed. It is closely controlled by system-level power management signal for powering down of unnecessary portions of the circuit when it is not needed. Another similar technique to clock gating is the Control Generated Clocks (CGC) [111] which generates a control clock centrally.

2.3.1.5 Technology Level

One effective way to reduce power is to reduce the supply voltage as stated in Section 2.3.1. To maintain the speed performance while reducing voltage, the device must geometrically scale down in size. However, by scaling down the voltage supply and size of the transistor, many other problems emerge. Problems like reduced noise immunity, increased leakage current, decreased output current driving capability, increased interconnect delays and device capacitance become apparent [112]. Therefore, there is a trade-off of performance for reduced power consumption by scaling down in size. Two general approaches have been taken by researchers to get round these problems.

The **first approach** is to look for ways to reduce the adverse effect of scaling both transistor size and voltage supply down while continuing trend of reducing the size of the transistor. Down scaling of the transistor size is seen in the research of novel low-power deep sub-micron transistor technology like FinFET [113–115] (or double gate MOSFET (DG-MOSFET) [112, 114–116]), double gate silicon-on-insulator transistor (DG-SOI) [114] and ultra-thin body

silicon-on-insulator transistor (UTB-SOI) [112, 114, 117].

In order to achieve the power-delay characteristic needed for some particular application as conventional MOSFET size continues to scale down, a optimal transistor size [83, 93] had to be found. Even when the optimal scale down size is found, the threshold voltage of a MOSFET still needs to be adjusted so that the MOSFET can work properly. As the MOSFET channel length gets shorter, the threshold voltage of a MOSFET needs to be increased [118] so that no punch-through will occur. However, by increasing the threshold voltage the MOSFET will slow down. Therefore, work in [119, 120] set out to look for the optimal threshold voltage for the MOSFET to have a satisfactory speed at low power.

The **second approach** is to look for new material and design to boost the transistor performance at the present attainable dimension or at a slightly scaled down dimension. The Silicon Germanium (SiGe) MOSFET [112], the Source-to-Drain Non-uniformly Doped Channel (NUDC) MOSFET [121], the Dynamic-threshold MOSFET (DTMOS) [112] and the Variable Threshold Voltage CMOS (VTCMOS) [30, 32, 34, 122] are examples of such endeavour.

2.3.2 Multi-level Power Reduction

Most low-power systems usually use more than one type of power reduction technique. This is because by using multiple power reduction techniques, the power consumption of a system can be further reduced. Furthermore, some of these power reduction techniques are interdependent. By using one of these techniques, it is inevitable that another technique had to be used. One example of such interdependent power reduction techniques is the shut-down power management technique which can only work when there is clock gating. The systems described in [15, 17, 18, 29, 31, 34, 49, 53–55, 80, 83, 85–87, 92, 93, 95, 107–109] are examples of such systems, which use more than one type of power reduction technique.

Multi-level power reduction is so effective that many have developed automated design software to try out different combinations of low power techniques (or software instruction) and accurately evaluate them to find the optimal solution. Work in these areas can be found in [45, 123–126]. Each of these describes the power estimation and modelling for different design levels and applications. These estimates and models are then used for finding the most optimal combination. A specific example of such is the re-targetable Instruction Level Parallelism

(ILP) Compiler [127] which combines the compiler with an architectural component selection algorithm. This compiler has the ability to arrange architectural parameters to yield an optimal low power system combination of software and hardware known as an application specific programmable processor.

2.3.3 Low Power Works on DWT

The DWT had been around for quite a few decades. As conventional convolution based DWT is computationally intensive making it not very viable for implementation, few researcher work on it. However, in the late 1980s, a good number of DWT implementation emerged. Out of these work, none were target for low power. It was only in the late 1900s that low power DWT implementation starts to emerged. However, these works are still not viable for implementation as it is based on the traditional convolution based DWT. Examples of these works can be found in [105, 128–132].

In [128], Namgoong et. al. proposed a low power conventional convolution 5/3 DWT implementation. The low power design of this system was approached at mainly the algorithmic, architectural and the circuit level. Kuntzel et. al. [129] proposed a lossless lifting-based 6/10 DWT implementation. This implementation reduces power by designing the DWT filter pair as a integer fixed-point filter pair. Another implementation by Zhang et. al. [130] was based on the conventional convolution DWT. This 3-D DWT implementation reduced power by using low power ALU building block cells, minimize the the number of processing element by using central control design and maximizing the usage of sub-chip components by eliminating any redundant modules. A similar 3-D system was also proposed by Das et. al. [132]. Simon et. al. proposed a Embedded Zero-tree Wavelet (EZW) DWT video system that reduces power by changing both voltage and clock frequency of the system according to the requirement of both the data and the condition of the transmission media. Another work by Marino et. al. [131] reduced the power of a conventional convolution 2-D DWT system by using pipelining to increase throughput and decrease clock frequency.

After the released of the JPEG2000 standard, there were only two low power works on the JPEG2000's core DWT filter pairs the (5/3 and 9/7) [133, 134]. Dang et. al. came up with a scheme that express all the wavelet filter coefficients (for both 5/3 and 9/7) as scaled power-of-two integer numbers (with common denominator) so that shift-add operation will be sufficient to compute the filter output values. To reduce the number of adders and shifters for

the implementation, Dang et. al. use sub-expression elimination method to reduce the common sub-expression between the two channel filter banks. Dang et. al. also use canonic-sign digits to reduce the number of adders in the filter's tap numerator. The work by Dang et. al. on the JPEG2000's 5/3 and 9/7 filters were not lifting-based but convolution based.

2.3.4 Low Power Design Approaches

There are various approaches to go about designing a low power system in term of design abstraction levels. While power reduction can be employed at only one design level, it is more advantageous to consideration power reduction at more than one level of design abstraction because of the interdependent nature of power reduction techniques. There are two main approaches for embarking on a low power system design. The two approaches are: -

- Bottom-up design approach
- Top-down design approach

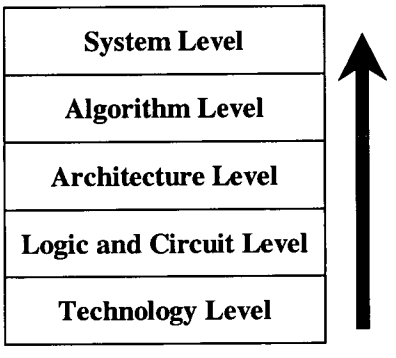


Figure 2.6: *Bottom-up design flow approach*

The **bottom-up design approach** (as shown in Figure 2.6) starts implementing power reduction techniques from the bottom of the design hierarchy and works up the design abstraction levels. The power reduction can start at the bottom of the design hierarchy either with the technology level or the logic and circuit level. In the similar way, power reduction can stop at the top of the design hierarchy with either the algorithm level or the system level. Voltage scaling is an example of a low power strategy that takes the bottom-up power reduction design approach starting from the logic and circuit level.

The **top-down design approach** (as shown in Figure 2.7) starts implementing power reduction techniques from the top of the design hierarchy at either with system level or algorithm level.

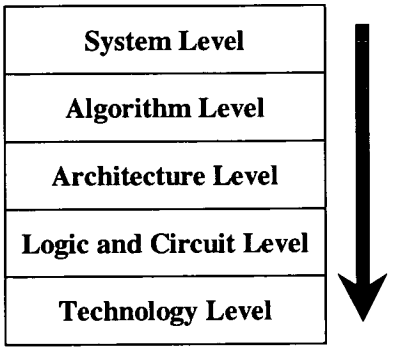


Figure 2.7: *Top-down design flow approach*

The power reduction work will continue down the design abstraction hierarchy and stop at either of the two bottom design abstraction levels. The top-down power reduction design approach can be seen in use when employing the switched capacitance reduction strategy.

2.3.5 Short-comings and Limits of Existing Power Reduction Strategies and Techniques

Although there are many benefits in employing power reduction strategies and techniques in designing a low power system, there also short-comings and limits when utilising such techniques. Most power reduction techniques reduce the power consumption of a system at the expense of design time, performance of the system and the semiconductor area consumption. Every power reduction technique has a limit to which the technique can reduce a CMOS system power. The following are examples of the limits and short-comings of two power reduction techniques.

The first of these two examples is the limit of bus coding. Bus coding is very effective in terms of reducing switching power on buses. However, there is a limit to which bus coding is able to reduce power on data/address buses. This theoretical lower bound limit for bus coding was derived from calculation based on the entropy rate are presented in [68]. All bus coding are based on the principle of coding as many state changes as possible into a single sample transition. According to this theoretical lower bound, to further 'squeeze' anymore state changes into a single bit transition will result in an increase in power. This is due to the fact that the overhead incurred by implementing such coding will consume more power than it saves.

The second example is the limit to which the supply voltage of a system can be reduced to. Supply voltage scaling is an extremely effective way to reduce the CMOS power consumption. However, there is a limit below which the supply voltage cannot go down any further. One problem would be finding a suitable device that would operate at such a low voltage with a reasonable speed performance. Decreasing supply voltage slows down a transistor as the transistor cannot fully turn on and thus conducts less current. As such it makes a CMOS system unsuitable for high performance computing.

To enable the system to perform at a higher speed, additional power reduction techniques like architecture parallelism and pipelining are needed. However, using such techniques will further increased the system's semiconductor area overhead and switched capacitance. To get round this problem, the size of the transistor and its threshold voltage has to also be scaled down to increase its speed at low voltage. However, it is getting more and more difficult to reduce the size of a transistor as the physical limit which a transistor can scale down to is approaching. In addition, the threshold voltage of a deep sub-micron transistor is not scaling down fast enough to keep up with the reducing supply voltage [118]. To make things worse, the lowering of the threshold voltage creates other problems. One pronounce problem is the problem of an increase in leakage current due lowering of the transistor threshold voltages which is countering the benefit of its gain in speed and power.

Ever decreasing supply voltage could also pose yet another potential problem in Electromagnetic Compatibility (EMC), with the digital circuit itself being the 'victim'. In particular, when a low supply voltage digital circuit is in a system with radiating Radio Frequency (RF) circuit like a portable communication system. Reliability of the digital circuit will be affected and damage to parts of the digital circuit can even be possible. Shielding will increase the cost of such system. Finally, as supply voltage scales further down, a variable voltage low power system is forced to have a power-supply with voltage specification of very fine resolution to ensure proper operation of the system as well as power saving. Having a power-supply with voltage specification of very fine resolution would further adds to the cost and the design demand of the system.

2.4 Design Flow and Tools

Most low power CMOS system were designed and power evaluated with Computer Aided Design (CAD) tools and this research is no exception. The design in this research were captured using commercially available CAD tools. The design implementation cum power evaluation flow and the tools used in this research are briefly presented in the following sub-section. As existing power evaluation tools are just software programmes that estimates the power consumption of a CMOS circuit when it is operation, the workings and limitations of such software are also briefly discussed in this section.

2.4.1 Design Implementation and Power Evaluation Flow

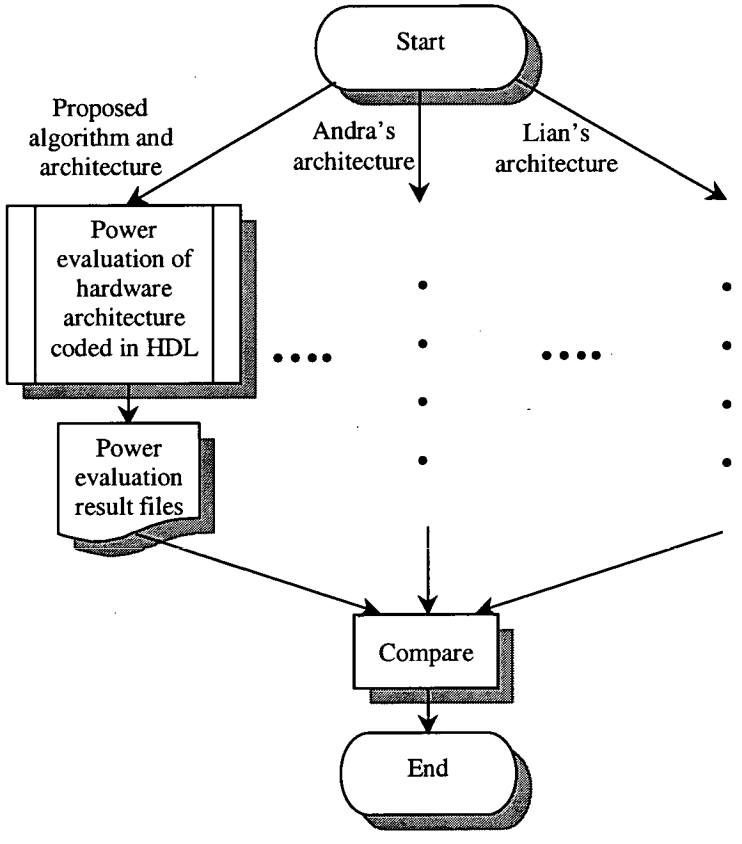


Figure 2.8: *Flowchart for power evaluation of different algorithm and architecture*

Often in designing a low power system, the power consumptions of the realised hardware system needs to be evaluated and compared with other similar implementations. These evaluations and comparison are needed so as to determine the effectiveness of the low

power techniques applied to the system. In this research, the power consumption of the lifting-based DWT hardware architectures in question are power evaluated and compare with one another. The power evaluation comparison flow of the architectures in question adopted in this research is as shown in Figure 2.8. The core design and power evaluation process shown in Figure 2.8 is depicted in the following Figure 2.9 with greater detail. The flow shown in Figure 2.9 is the design implementation and power evaluation flow used in this research for the implementation and the development of the algorithms and architectures.

The following are the major steps used in the design implementation and power evaluation in this research:-

1. Algorithm development, verification, test vectors/stimuli generation with Matlab.
2. Manual algorithm mapping to hardware system and component design and testbenches development in Verilog HDL.
3. Hardware components and system functional testing and verification using Cadence's Verilog-XL™ and Cadence's SimWave for waveform analysis.
4. Hardware synthesis to UMC 0.18μm technology using Synopsys's Design Compiler. Netlist and Standard Delay Format (SDF) files were generated as a result.
5. Hardware components' and system's netlist functional testing and verification using Cadence's Verilog-XL™ and Cadence's SimWave for waveform analysis.
6. Hardware switching (toggle) activity information extraction and Switching Activity Interchange Format (SAIF) files generation using Synopsys's Design Power via Cadence's Verilog-XL™ through Verilog PLI interface.
7. Power figure conversion from SAIF files using Synopsys's Design Power.
8. Manual comparison of the hardware power figures.

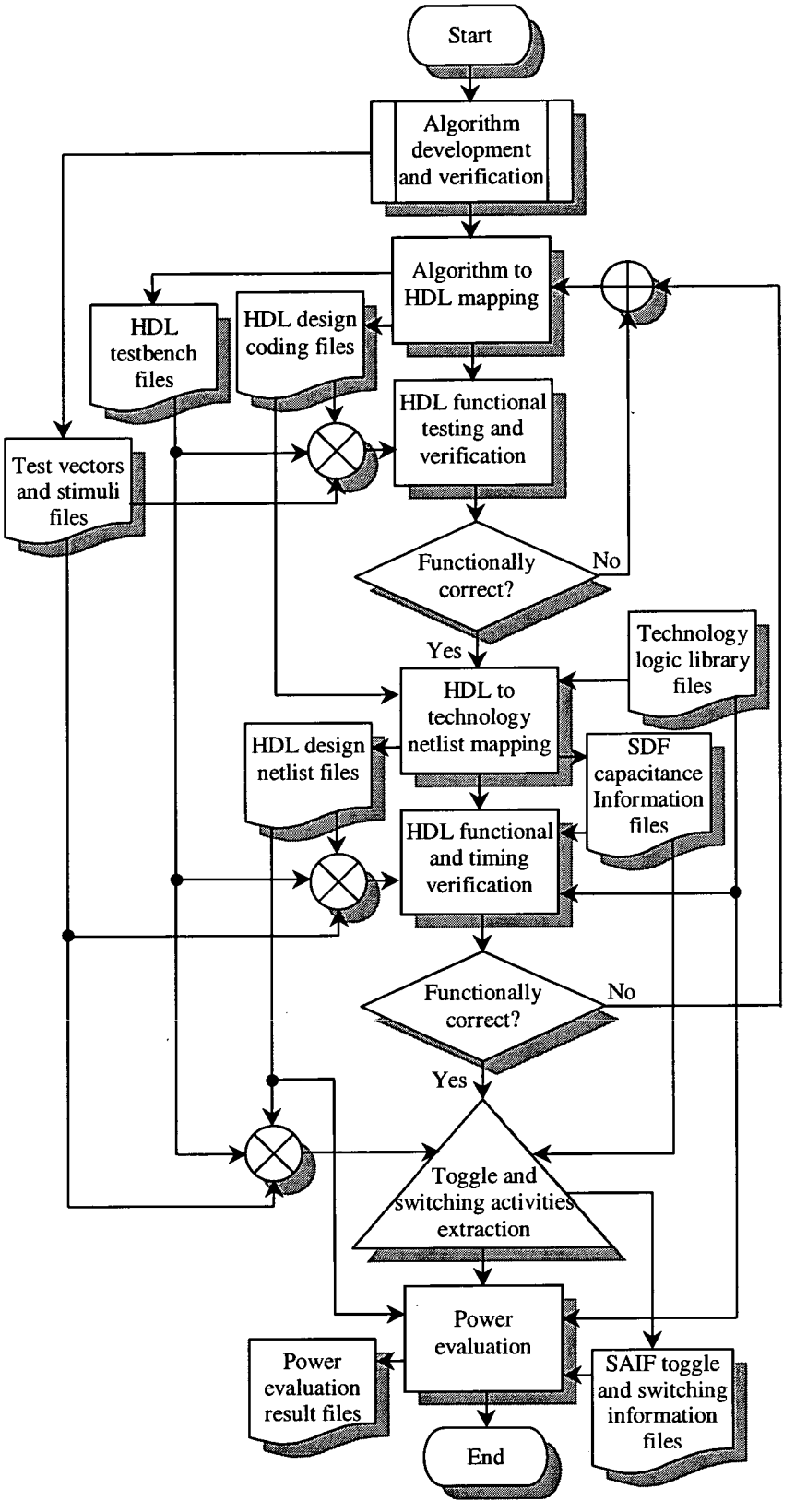


Figure 2.9: *Flowchart of the system hardware implementation and coding*

The Matlab programme was used to development the algorithm, verify the developed algorithm and generate the test vectors and stimuli because as a high-level language, the Matlab provides a quick way of authoring and verifying an algorithm. Cadence's Verilog-XL™ was to simulate and verify the functionality of the designs as is the only Verilog simulator available at in the institute at the time of development. Synopsys's Design Power was used to evaluate the power consumption of the hardware design as it is the only known stable commercially available tool for power evaluation at the time of evaluation.

2.4.2 Workings of Existing Power Evaluation Tool

Existing power evaluation tools calculate and estimate a CMOS circuit's power consumption value based on the relationship given by Equation 2.3 for static-leakage power, 2.10 for short-circuit power and 2.13 for load-capacitance switching power. The power evaluation/estimation process can generally be divided into two main processes; these are the static power estimation process and the dynamic power estimation process. These power estimation processes are described in the following sub-subsections.

2.4.2.1 Static Power Estimation Process

The power evaluation tool calculates the static power of a CMOS circuit by the following steps:

-

1. Extract the type and number of gate cells from the target design netlist which had been mapped to a particular CMOS technology by the synthesis programme.
2. Power evaluation tool retrieve the static-leakage power figures of those particular gate cells mapped in the netlist from the technology library file.
3. The retrieved gate cell static-leakage power figures is multiplied by the number of gate cells in the netlist to accumulate to the static power grand-total.
4. The static power grand-total figure is scaled according to the length of time which the power evaluation take places.

2.4.2.2 Dynamic Power Estimation Process

The power evaluation tool calculates the dynamic power of a CMOS circuit by the following steps: -

1. Extract the type and number of gate cells from the target design netlist which had been mapped to a particular CMOS technology by the synthesis programme.
2. Extract of the gate cell load-capacitance from the target design netlist and the technology library file via the synthesis programme.
3. Power evaluation tool retrieve the dynamic power figures (which includes both short-circuit power and load-capacitance switching)in term of energy (for both positive clock edge and negative clock edge) of those particular gate cells mapped in the target design netlist from the technology library file.
4. Gather the switching activities information of the mapped target design netlist from the target design testing simulation for each type of the gate cells used in the target design netlist.
5. The retrieved gate cell dynamic energy figures and load-capacitance value of respective gate cells are multiplied by the number of switching activities of respective gate cells in the target design circuit netlist to accumulate to the dynamic energy grand-total.
6. The dynamic energy grand-total figure is converted into power according to the length of time which the power evaluation take places.

2.4.3 Limitation of Existing Power Evaluation Tool

The processes described in the above section are generally accepted way of estimating power consumption figures of a CMOS circuit. However, such way of estimating the power consumption has its limitations and short-comings. The following will discuss some of these limitations and short-comings.

As it can be seen from the above section that the existing power evaluation tool rely heavily based on the information provided in technology library file to calculate/estimate the power consumption figures of a CMOS circuit. One of the short-comings is that estimated power consumption figure can only be as accurate as the figures given in the technology library.

Although, the foundry vendors will always try to provide the most accurate figures as possible, the provision of 3 classes (Best Case, Typical Case and Worst Case) of the technology libraries shows that it is extreme hard to get accurate gates power consumption figures in reality. These 3 classes of technology libraries are provided by the foundry vendors to cover as wide as possible the condition of which the CMOS circuit would turn out after it is fabricated. Besides that, power evaluation tool do not take into account of the operating environment variation as the power evaluation tool are based on the following assumptions:-

1. Consistent fabrication processes
2. Constant operating temperature
3. Constant voltage supply
4. Electrically noiseless environment
5. Infinite and continuous time process

These assumptions would help to simplify the estimation of a CMOS circuit's power consumption. However, it will not give a very accurate view of a CMOS circuit in real life operating environment. Such assumptions, especially the assumption on the infinite and continuous time process, can also impedes a fair power comparison between two CMOS circuits which perform the same task but complete the task at different time. The details of this short-coming is presented in Chapter 6. Despite of these short-comings, the power evaluation results from existing power evaluation are still very valuable and useful in the design of a low power design. This is because these power figures will still give circuit and system designers a very good gauge of how much power a design will consume when the whole design is finished.

2.5 Summary

This chapter introduced the sources of power dissipation of a CMOS circuit. These power dissipation sources can generally be categorised into two main types; these are the statics power and the dynamic power dissipation. The main contributor to the static power is device technology dependent transistor static-leakage power P_l component. The short-circuit power P_{sc} dissipation and the load-capacitance switching power P_{sw} dissipation are the two main contributors of the dynamic power dissipation. Although, the static-leakage power P_l is

projected to increase in the next ten years, low power CMOS's (LSTP logic [11]) static-leakage power dissipation will remain low. Consequently, the dynamic power is still the dominant factor in the power consumption value of low power CMOS digital systems for now and some years to come. Therefore, by reducing the switching dependent power like short-circuit power P_{sc} and load-capacitance switching power P_{sw}, significant power can be reduced. The supply voltage V_{dd}, the switching activities S, the clock frequency f_{clk} and the load capacitance C_l are the 4 key terms of the load-capacitance switching power P_{sw} which can be minimized to reduce the power consumption of a digital CMOS system were presented. By targeting and reducing the switching activity S and the clock frequency f_{clk}, the power dissipation by short-circuit power P_{sc} will also be reduced. Power reduction techniques developed to reduce both the load-capacitance switching power and the static-leakage power can be categorised into 5 interdependence hierarchical design abstraction levels. These 5 hierarchical design abstraction levels (from the top) are the System level, the Algorithm level, the Architecture level, the Logic and Circuit Level and the Technology level. Various existing power reduction techniques at all 5 design abstraction levels were reviewed. Along with these, examples of the limits and short-comings of some of these power reduction strategies and techniques were discussed. The design implementation and power evaluation flow of the hardware architecture used in this research were also presented. Finally, the workings and short-comings of existing power evaluation tools were briefly discussed.

Chapter 3
Image Compression

3.1 Introduction

With the advances in VLSI digital technology, many high throughput and performance imaging and video applications have emerged and increased in usage. Demand for image/video applications in portable form has especially increased in recent years. At the core of these productive and useful applications is image and video compression technology. With portable video telephony, internet and entertainment applications in increasing demand, embedded image and video has become an essential feature in most modern day portable applications.

There are many different image and video compression techniques in existence. In order for image and video application to be viable, image and video compression technology has to be efficient. Beside efficiency, there must be a standardized way of how the image or video is encoded or decoded, so that an encoded image or video data can be used by all. Therefore, a significant amount of work has being put into the development of image and video compression techniques and standards.

This chapter presents an overview and the principles behind common image compression techniques especially the discrete wavelet transform (DWT). The evidence that low power image compression techniques are needed for image and video application to be viable is presented. The standardization work of the international standard organisation on image and video compression is also presented.

This chapter is organised as follows. Section 3.2, gives a brief overview of the two commonly used image compression techniques. In Section 3.3, the development, algorithm and implementation of the discrete wavelet transform and its lifting based scheme are presented. Following that, the international standards for image and video compression are presented in Section 3.4. Next, in Section 3.5 the computational intensiveness of image compression and the justification for low power image compression are shown. Finally, the summary of this chapter is given in Section 3.6.

3.2 Image Compression Techniques

The use of digital image and video are on the increase and are becoming widespread in recent years. The usage of image and video can be found in a wide range of applications like videophone, digital still and video camera, webcam, internet imagery and video, digital video compact discs (VCD) system, digital versatile discs (DVD) system etc. These applications have virtually become one of the modern day essentials. At the core of all these image and video applications is image compression technology. A typical image compression technology usually consists of image pre-processing, transform coding, coefficient data quantization and entropy coding as seen in Figure 3.1.

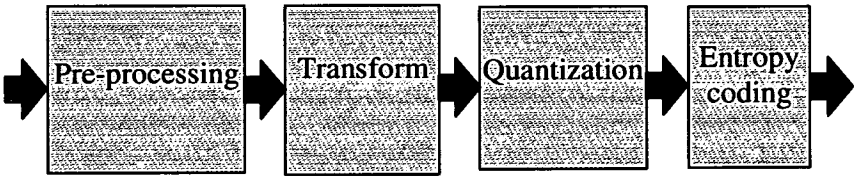


Figure 3.1: *Generic image compression core process blocks (Modified from [1])*

When the raw data image samples go into the image compression system the data samples are passed through a range of image pre-processing so that these image samples can be more efficiently transformed later on. After the transformation, the processed data is then quantised with steps that were optimised to preserve as much as possible the energy profile of the transformed values. Finally, these quantised values are further code-compressed with entropy coding.

At the heart of these image compression technologies is the transform coding which is an essential part that makes image compression possible. Most of these image compression technologies transform the image data samples from one (spatial) domain to another (spatial frequency) domain to alter the distribution of the luminance level values so that most of them can be eliminated or be quantised with very few bits.

As most of the energy of an image concentrates at the low frequency end of the spatial spectrum, another type of transform coding called sub-band coding was developed. Sub-band coding is a technique that filters the image data samples into different frequency bands and a more efficient technique is then apply to the individual sub-bands. This is so that the luminance level values can be removed totally or be quantised with very few bits.

There are many different transform codings in existence, however there are two most commonly

used transform codings. These two most commonly used transform codings are the discrete cosine transform (DCT) [135] based coding and the sub-band (and wavelet [136, 137]) based coding. As the topic and theory of transform coding is well documented it will not be covered comprehensively in this chapter. However, a brief overview of these commonly used and relevant transform codings will be given in the following.

3.2.1 Discrete Cosine Transform (DCT) Coding

The Discrete Cosine Transform (DCT) is the most commonly used transformation technique in most modern image and video compression technology. It was proposed by Ahmed et al. [135] in 1974. Since then it has been widely used for image compression as it possesses a high energy compaction property with most of the image signal information concentrated in a few low-frequency components of the DCT. The image coding performance of DCT is approaching that of the optimum Karhunen-Loeve transform (KLT). The DCT is a frequency transform similar to Discrete Fourier Transform (DFT) except that DCT has only real and even symmetry (cosine) terms. The following Equation 3.1 describes the one-dimensional (1-D) DCT.

$$F(n) = \frac{2}{N} C(n) \sum_{x=0}^{N-1} f(x) \cos \frac{(2x+1)n\pi}{2N} \tag{3.1}$$

$F(n)$ in Equation 3.1 represents the output coefficients and the $f(x)$ represents the input data samples. $C(n)$ in the equation is equal to $\frac{1}{\sqrt{2}}$ when n is 0 and is 1 when the value of n not 0. The term N is the total number of data samples. The $N \times N$ two-dimensional (2-D) DCT seen in Equation 3.2 is the form commonly use in image compression transform coding.

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos \frac{(2x+1)u\pi}{2N} \cos \frac{(2y+1)v\pi}{2N} \tag{3.2}$$

$F(u,v)$ is the output coefficients and the $f(x,y)$ is the 2-D input image data samples where x, y are spatial coordinates in the sample domain and u, v are coordinates in the transform domain. The coordinates u, v, x and y range from 0 to N-1. The $C(u)$ and $C(v)$ in the equation are equal

to $\frac{1}{\sqrt{2}}$ when u,v is 0 and is 1 when u,v are otherwise.

The number of frequency bands and the computation complexity is fixed and is determined by the number of pixels in the 2-D image data. Although, the DCT is a very efficient transform in term of energy compaction, the transform is a computationally involved process because it requires a substantial amount of multiplication operations (as can be seen in Equation 3.2) to process a single frame of image data. As such, most transforms divide-up the image frame into pixel blocks of 8×8 or 16×16 to reduced computational complexity and memory requirements. This process of dividing-up the image frames into pixel blocks is called blocking.

The reduction of the computational load can be calculated as follows. For an $N \times M$ image, using a block-size of $p \times p$ and with a fast algorithm, the computational load per block is $\alpha\ p^2 log_2 p^2$. α is the DCT processing asymptotics term which is the infinite asymptotics that takes into account of the small and non-contributing terms in the equation which differs from one fast algorithm to another. These terms can be neglected as it diminishes when the dominating term (in this case $p^2 log_2 p^2$) come to dominate when it (p) grows. The computational load per image is $\alpha\ N^2 log_2 N^2$ and the computational load per image with blocking is $\alpha\ N^2 log_2 p^2$. Therefore, the computation reduction due to the blocking is $\frac{log_2 N^2}{log_2 p^2}$. The computation reduction for using blocking of block size of 16×16 on a 512×512 image is 2.25. The reduction for the memory requirement is calculated as follows. For an $N \times N$ image, using a block-size of $p \times p$, the total number of blocks in the $N \times N$ image is therefore $\frac{N^2}{p^2}$. The main memory required for the transform is $p \times p$ which is an reduction of $\frac{N^2}{p^2}$.

Although blocking reduces both memory and computation requirement, blocking creates blocky artefacts at low rate compression. Besides blocking, a fast DCT [138] (FDCT) was also proposed to decrease the computation redundancy and complexity of the DCT by algorithmic means.

3.2.2 Sub-Band Coding

As noted in Section 3.2.1, most of the energy of an image is concentrated at the low frequency end of the spatial spectrum. Therefore, the sub-band coding is developed to capitalize on this image spatial frequency characteristic. Sub-band coding usually involves filtering the image data into different spatial frequency bands. After that an efficient compression scheme (like differential pulse code modulation (DPCM) and vector quantisation (VQ)) can be applied to

the individual sub-bands with attention focussed on high energy bands. This is so that the luminance level values in these bands can be eliminated totally or be more efficiently quantised with very few bits.

Sub-band coding filtering is implemented with digital filters and most of these digital filters are finite impulse response (FIR) filters. Most of these filters are FIR filters because of its linear phase response that will not cause phase error which our human visual system is very sensitive to. Using filters for the sub-bands however poses another problem. The amount of output data will now increase to the amount of input data multiplied with the number of frequency bands. However, as it turns out, signal processing principles allows data samples of each sub-band to be down-sampled according to the reduction of signal detail due to the sub-band filtering [139]. As such, the total number of output samples neither increases nor decreases. With down-sampling, interpolation is needed to reconstruct the transformed sample to it original image data. And in order to have a perfect reconstruction from the interpolated data, an ideal initial filtering of the input signal is needed. Ideal filtering in reality is not possible as all implementable filters have transition bands of a finite width [139] (i.e. no real life filter has a perfect vertical cut-off frequency response characteristic). As such, perfect reconstruction is not possible as errors in the image are inevitable due to the imperfect filter. Therefore, in order to reduce these errors in the image, the number of sub-bands are kept low and work was focused on designing a filter that will produce output close to the ideal requirement. The two bands (low and high band) scheme is one of the popular sub-band coding schemes as shown in Figure 3.2.

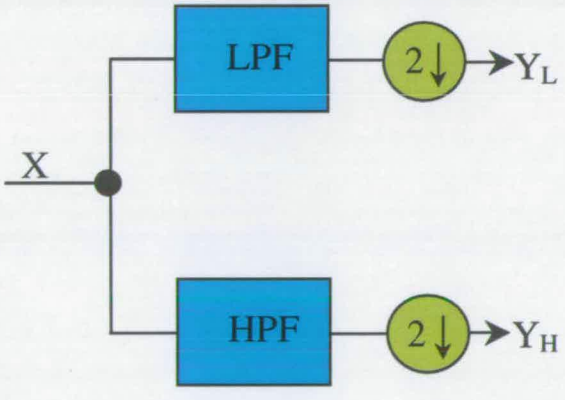


Figure 3.2: *Two bands (low and high band) sub-band coding structure*

One of the more popular filters which are used for sub-band coding is those filters used in the Discrete Wavelet Transform (DWT). The wavelet technology started off from quantum mechanics as a mathematical work and not from sub-band coding. The DWT is very similar

to the traditional sub-coding except that the frequency response of the wavelet function at the bandpass region might not be flat. The bandpass frequency response might not be flat, as a wavelet waveform in the time domain can be any arbitrary complex waveform which might not be a simple sinusoid sinc waveform. The DWT is also very similar to Fourier transforms except that DWT is localised in both time and frequency domain whereas the Fourier transform is only localised in the frequency domain. A further review of the DWT and the development of its lifting based algorithm are given in Section 3.3.1 and 3.3.3 respectively.

3.3 Discrete Wavelet Transform (DWT) Algorithm and Implementation

Most DWT video compression is implemented as a dyadic sub-band coding process. Large parts of these processes are implemented as FIR filters for both the high and low band processing. The large amount of resource seems to be inevitable and necessary to realise the DWT, when considering both the dyadic sub-band coding and FIR filtering needed for DWT. However, on a closer look, there are many redundancies in the DWT system which can be eliminated or minimized. The following sub-sections describe an overview of the DWT, the conventional DWT implementation and area where such implementation can be improved (in term of power and area consumption). With the redundant area identified, an alternative low resource realisation scheme, the lifting based scheme is introduced. As the theory of DWT and the lifting based scheme are well documented [5, 6, 137, 140], it will not be covered comprehensively in this chapter. However, the relevant part of the algorithm and implementation of the transforms will be presented in the following sub-sections. As most DWT are implemented with FIR filters, the effect of using a FIR filter for its implementation will be considered briefly. However, the theory on FIR filters will not be covered in this thesis as it is well established and documented [141, 142].

The mathematical terms in this thesis with the tilde ($\tilde{}$) on top of it, indicates that the term or function is related to the analysis/decomposition process. The term without the tilde on top of it, indicates that the term or function is related to the synthesis/recomposition process (except for input samples).

3.3.1 Discrete Wavelet Transform (DWT)

The concept of wavelets arose from quantum mechanics, however the use of wavelets in communication was introduced by Gabor [143] with the elementary signal $\Psi(t)$. The elementary signal is used to generalise all periodic waveform for any frequency and length of time. This elementary signal basically describes a sinusoid waveform enveloped by a Gaussian function. The following Equation 3.3 describes this elementary signal.

$$\Psi(t) = e^{-\eta^2(t-t_o)^2} e^{j(2\pi f_0 t+\theta)} \tag{3.3}$$

The t_0 and f_0 represent the maxima of $|\Psi(t)|^2$ in time domain and the Fourier transform of $|\Psi(t)|^2$ in frequency domain respectively. The θ in the equation represents the phase angle of the sinusoid and the η is the factor that determines the time domain pulse width. This elementary signal can be used to describe on one extreme, an impulse with $\eta \rightarrow \infty$, and on the other extreme, the Fourier analysis with $\eta = 0$.

It is with the elementary signal that the continuous wavelet transform was developed. The following Equation 3.4 describes the continuous wavelet transform (CWT).

$$\gamma(s,\tau) = \int f(t)\Psi^*_{s,\tau}(t)dt \tag{3.4}$$

The * in Equation 3.4 denotes the complex conjugation of the wavelet basis function $\Psi^*_{s,\tau}(t)$. The $f(t)$ is the function that is to be decomposed into sets of wavelets. The variables s and τ represent the scale and translation of the basic wavelets. The wavelets are generated from a single basic wavelet $\Psi'(t)$ called the mother wavelet by scaling and translation as described in the following Equation 3.5.

$$\Psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\Psi'(\frac{t-\tau}{s}) \tag{3.5}$$

Note that the wavelet basis function (mother wavelet) $\Psi'(t)$ is conceptually similar to the

elementary signal $\Psi(t)$ except that the mother wavelet is not specified. This is because the mother wavelet is now allowed to take any arbitrary waveform.

However, there are 4 problems with the CWT which make it impractical to be used widely. First of all, as a continuous frequency transform, the CWT requires a continuous shifting of a scalable wavelet function over the signal of interest to compute the correlation between them.

Secondly, even if the transform is discrete, there are an infinite number of wavelets to be translated and dilated for the transform of a signal of interest as there are no stopping boundaries for the translation and dilation.

Thirdly, the frequency of a wavelet can never reach zero frequency in reality. This is due to the fact that the wavelet is a wave and therefore has a band-pass type frequency response. As such, for the transform to cover the spectrum down to zero, infinite dilation of the wavelet is needed. Thus, making it impractical and unsuitable for a full spectrum transformation. Finally, as the wavelet can be any arbitrary signal, most of the wavelet functions do not have analytical solutions and can only be calculated numerically. In order for the wavelet transform to be practical these problems need to be solved.

The solution to the first problem is to discretize the wavelet by introducing discrete integer variables j and i to the scale and the translation respectively as can be seen in Equation 3.6.

$$\Psi_{j,i}(t) = \frac{1}{\sqrt{s_0^j}} \Psi'\left(\frac{t_0 - i\tau s_0^j}{s_0^j}\right) \tag{3.6}$$

By setting s_0 to 2 and τ to 1, the wavelet function then has dyadic sampling in both time and frequency axes giving it a convenient dyadic structure. As for solving the second problem, the upper boundary for translation of the wavelets can be set with the duration of the signal of interest. As for dilating, the scale can be capped to a finite number by defining a scaling function $\Phi(t)$ [144] as it covers the rest of frequency spectrum that a finite number of dilations of the wavelets cannot cover. The introduction of the scaling function $\Phi(t)$ also solves the third problem. This is because the scaling function $\Phi(t)$ has a frequency response likened to a low pass filter response which will adequately cover the spectrum down to zero frequency. A family of functions can be generated from the basic scaling function $\Phi(t)$ from the following general scaling function equation.

$$\Phi_{j,i}(t) = \frac{1}{\sqrt{s_0^j}}\Phi(\frac{t_0 - i\tau s_0^j}{s_0^j}) \tag{3.7}$$

Finally, to get round the fourth problem, the wavelet transform is implemented as an iterated digital filter bank like that of the sub-band coding. In this way, the wavelets do not have to be explicitly analytically solved. By taking all the above into account both the scaling function and wavelet function can be expressed in the following multi-resolution analysis (MRA) formulae.

$$\Phi(2^j t) = \sum_i h_{j+1}(i)\Phi(2^{j+1}t - i) \tag{3.8}$$

$$\Psi'(2^j t) = \sum_i g_{j+1}(i)\Psi(2^{j+1}t - i) \tag{3.9}$$

The term $h(i)$ in Equation 3.8 represents the scaling function coefficients and the term $g(i)$ in Equation 3.9 represents the wavelet function coefficients. With these formulaes, a signal $f(t)$ can be describes by the following expressions.

$$f(t) = \sum_i c_{j-1}(i)\Phi(2^{j-1}t - i) + \sum_i d_{j-1}(i)\Psi'(2^{j-1}t - i) \tag{3.10}$$

If the scaling function $\Phi_{j,i}(t)$ and the wavelet $\Psi_{j,i}(t)$ are orthonormal or a tight frame, the coefficients $c_{j-1}(i)$ and $d_{j-1}(i)$ are found by taking the inner product and can be expressed as follows.

$$c_{j-1}(i) = \sum_m h(m - 2i)c_j(m) \tag{3.11}$$

$$d_{j-1}(i) = \sum_{m} g(m - 2i)c_j(m) \tag{3.12}$$

From the above two equations, it can be seen that the present coefficients can be computed with a weighted sum of the previous scaling coefficients. Thus, reflecting the iterated filter bank structure of the sub-band coding as shown in Figure 3.3. At this point, the transform becomes a discrete wavelet transform (DWT). Note that Equation 3.11 and 3.12 are in fact describing a convolution process. As such, most DWT filters are normally implemented as a FIR filter.

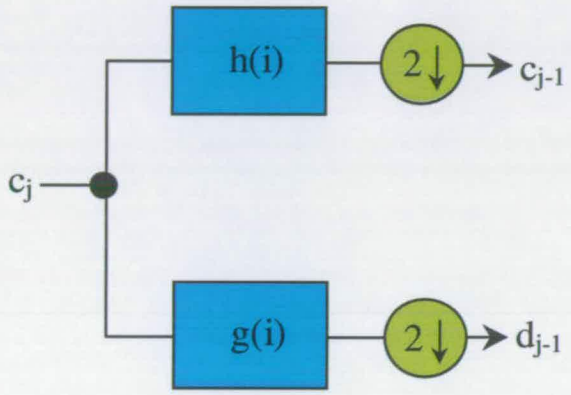


Figure 3.3: *One stage of the iterated filter bank structure of wavelet transform*

As the DWT is implemented as sub-band coding with iterated filter bank, it allows a selectable number of dyadic set of frequency bands through number of levels of transformation. However, as these filters are implemented as FIR filters, the DWT operation can be very resource and power consuming.

3.3.2 Conventional Convolution Based DWT Implementation

Conventional DWT implementations are convolution based. As established before, that the core of the wavelet transformation is FIR filters. To get an image to decompose at a single level we have to go through six filtering processes (see Figure 3.4). This process is a 2-D process. There are two main filters in the whole DWT processes; the Low Pass Filter (LPF) and the High Pass Filter (HPF). These two filters are used recursively to produce the final decomposed image coefficients. The output of the LPF ($\widetilde{LP}$) and HPF ($\widetilde{HP}$) are related to the input by the following equations:-

$$\widetilde{LP}(z) = \widetilde{LPF}(z)x(z) \tag{3.13}$$

$$\widetilde{HP}(z) = \widetilde{HPF}(z)x(z) \tag{3.14}$$

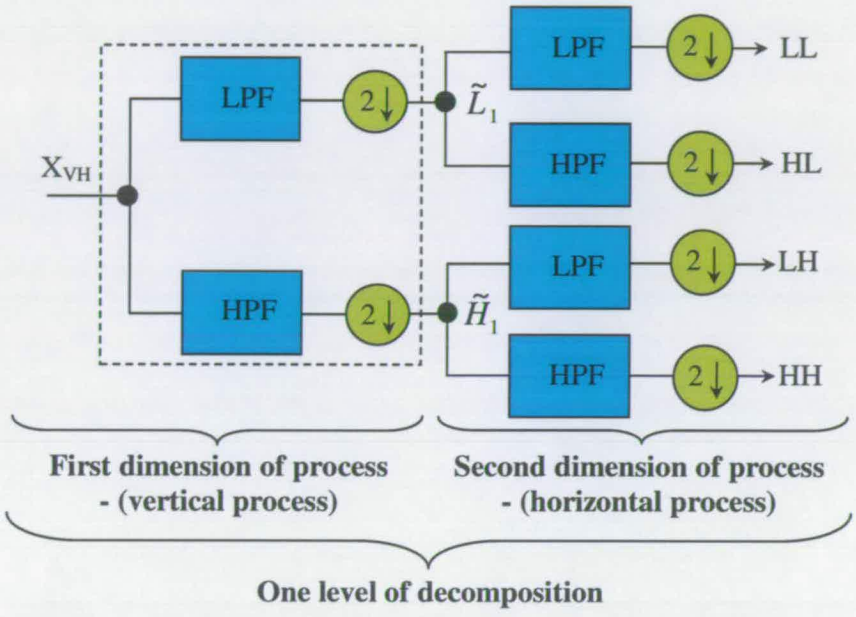


Figure 3.4: *Conventional convolution-based two-dimensional DWT process flow*

The LL, HL, LH and HH terms in Figure 3.4 are the 2-D single level of decomposition sub-band output that has gone through low pass both vertically and horizontally, high pass vertically and low pass horizontally, low pass vertically and high pass horizontally, and high pass both vertically and horizontally respectively. Conventional realisations of the DWT are direct implementations. The conventional convolution-based DWT is computationally intensive and area wasting as all the image pixels are put through all of the 6 filtering processes just for a 2-D single level of decomposition or recomposition. This makes the tradition implementation of DWT not viable as it is power hungry. Conventional DWT wastes redundant computing power and memory space on processing and storing data vectors, since half of these will be dropped during the down-sampling or sub-sampling later on in the process as can be seen from Figure 3.4. By analytically removing these redundancies by an algorithmic method, great savings can be achieved.

3.3.3 Lifting Based DWT Algorithm and Implementation

The drawbacks of the conventional convolution-based implementation are overcome by the lifting-based DWT (Figure 3.5) algorithm. The lifting scheme was devised not only to reduce the computational requirement of DWT, it can also be used to obtain a DWT filter pair that is invertible or reversible. However, the details of how to find the reversible filter pair will not be covered here. The lifting-based DWT reduces computational and area redundancy by:-

- Firstly, taking into account the redundancy of down-sampling and avoiding computation of the output vectors that will eventually be dropped.
- Secondly, exploiting the similarities between the low-pass filter (LPF) and high-pass filter (HPF) to further reduce redundancies.

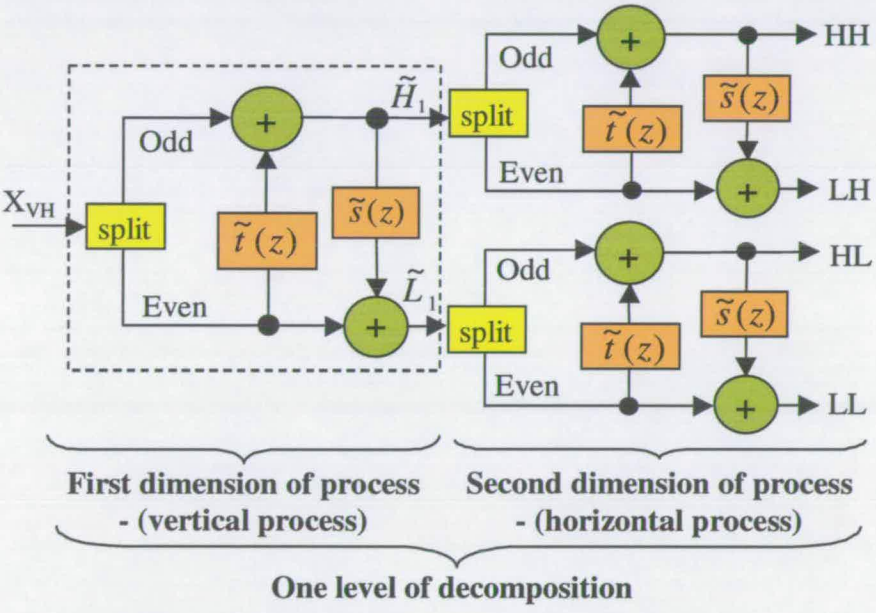


Figure 3.5: *Lifting-based two-dimensional DWT process flow*

By considering these computational redundancies, the lifting-based scheme can reduce more than 50% of the computational requirement compared to a conventional DWT as it does not have to compute data that will be down-sampled later on. The lifting scheme reduces redundancy and utilises similarities within the transform by factorising the original LPF and HPF equations into common factors. These commons factors are then efficiently combined again later on to obtain the final sub-sampled outputs. The common factors are incorporated into two sets of equations (predict and update) that are responsible for producing a direct

down-sampled low pass (odd terms) outputs and high pass (even terms) outputs. As this scheme utilises similarity of both the LPF and HPF, the values of previously calculated terms are reused further down the process as can be seen in Figure 3.5. The definition of the sub-band outputs is the same as described in Section 3.3.2

3.3.3.1 Sub-Sampling Effects

It has been observed that the effect of sub-sampling results in only part of the wavelet filter tap coefficients being used by only part of the input samples. In order to understand and to exploit this effect to reduce the DWT computation requirement, the following equations have to be considered first.

$$\widetilde{LPF}(z) = \widetilde{LPF}_e(z) + \widetilde{LPF}_o(z) \tag{3.15}$$

$$\widetilde{HPF}(z) = \widetilde{HPF}_e(z) + \widetilde{HPF}_o(z) \tag{3.16}$$

Equations 3.15 and 3.16 shows the coefficient make-up of both the analysis LPF and HPF respectively. Both the LPF and HPF filter coefficients ($\widetilde{LPF}$ and $\widetilde{HPF}$) can be split into two parts (polyphase), the even coefficients (terms with sub-script$_e$) the odd coefficients (terms with sub-script$_o$). In the same way, the original input samples x can be also split into even-termed input samples x_e and odd-termed input samples x_o as shown in Equation 3.17. The delay term z^{-1} points to the fact that there is a delay between even and odd samples.

$$x(z) = x_e(z) + z^{-1}x_o(z) \tag{3.17}$$

The effect of sub-sampling causes the even-termed wavelet filter coefficients ($\widetilde{HPF}_e$ and $\widetilde{LPF}_e$) to be only used with even-termed input sample x_e and the odd-termed wavelet filter coefficients ($\widetilde{HPF}_o$ and $\widetilde{LPF}_o$) only with odd-termed input sample x_o. As such, the sub-sampled output of the LPF ($\widetilde{L_1}$) and HPF ($\widetilde{H_1}$) can be written down as shown in Equation 3.18 and 3.19 to reflect this effect.

$$\widetilde{L_1}(z) = \widetilde{LPF}_e(z)x_e(z) + z^{-1}\widetilde{LPF}x_o(z) \tag{3.18}$$

$$\widetilde{H_1}(z) = \widetilde{HPF}_e(z)x_e(z) + z^{-1}\widetilde{HPF}x_o(z) \tag{3.19}$$

To quantify both the sub-sampled output of the LPF ($\widetilde{L_1}$) and HPF ($\widetilde{H_1}$) together, the function $\widetilde{LPF}$ (Equation 3.15) and $\widetilde{HPF}$ (Equation 3.16) are combined together to get a polyphase matrix $\widetilde{P}(z)$ (Equation 3.20).

$$\widetilde{P}(z) = \begin{pmatrix} \widetilde{LPF}_e(z) & \widetilde{LPF}_o(z) \\ \widetilde{HPF}_e(z) & \widetilde{HPF}_o(z) \end{pmatrix} \tag{3.20}$$

With Equation 3.20, the down-sampled output of the LPF ($\widetilde{L_1}$) and HPF ($\widetilde{H_1}$) is related to the input as shown in Equation 3.21.

$$\begin{pmatrix} \widetilde{L}_1(z) \\ \widetilde{H}_1(z) \end{pmatrix} = \widetilde{P}(z) \begin{pmatrix} x_e(z) \\ z^{-1}x_o(z) \end{pmatrix} \tag{3.21}$$

The relationship of the down-sampled output of the LPF ($\widetilde{L_1}$) and HPF ($\widetilde{H_1}$) to the input samples, can be pictured as shown in Figure 3.6.

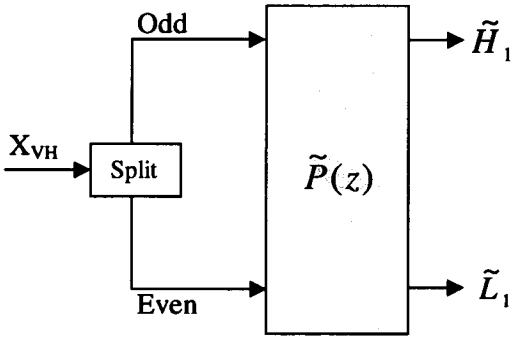


Figure 3.6: *Discrete Wavelet Transform using polyphase matrix*

3.3.3.2 Factoring Filter into Lifting Steps

Using the polyphase matrix $\widetilde{P}(z)$, the transform can be made invertible or reversible, meaning that perfect reconstruction is possible. This is done by making the inverse synthesis polyphase matrix $P(z)^{-1}$ equal to the analysis polyphase matrix $\widetilde{P}(z)$ satisfying the condition given in Equation 3.22. The $P(z)$ and I in Equation 3.22 represents the synthesis polyphase matrix and the unit matrix respectively.

$$\widetilde{P}(z^{-1})P(z) = I \tag{3.22}$$

The detailed steps to make the transform invertible or reversible by lifting will not be covered in this thesis as it is not the aim of this research to design a set of reversible DWT filters. The filters are therefore assumed to be reversible to start with. To exploit the similarities between the filters, the common factors of both filters (LPF and HPF) will have to be extracted into lifting steps [5, 6].

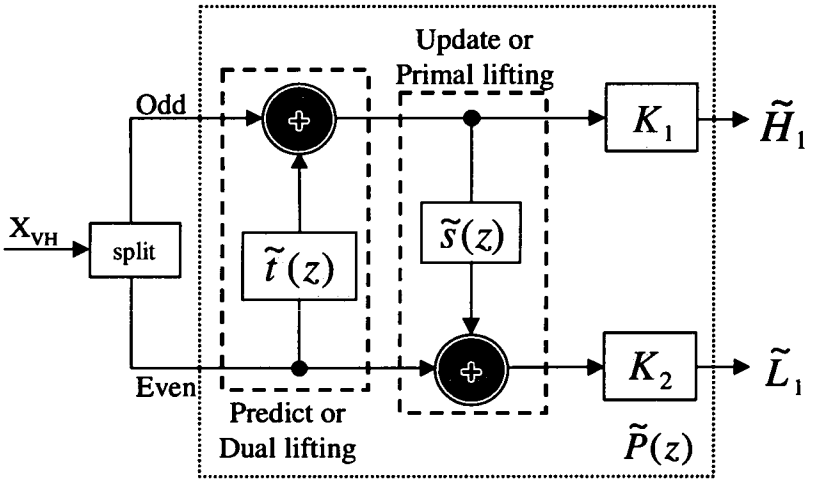


Figure 3.7: *Lifting-based DWT diagram showing the 3 distinct lifting steps; Predict (Dual lifting) step, Update (Primal lifting) step and Scaling step*

The lifting steps consist of three distinct steps are as shown in Figure 3.7. The three steps are: -

1. Predict (Dual lifting) step ($\tilde{t}(z)$)
2. Update (Primal lifting) step ($\tilde{s}(z)$)

3. Scaling step (K)

Predict or dual lifting lifts the high-pass sub-band with the help of the low-pass sub-band and the update or primal lifting lifts the low-pass sub-band with the help of the high-pass sub-band. The polyphase matrix that has been factored into lifting steps will take the form characterized by the following Equation 3.23.

$$\widetilde{P}(z) = \begin{pmatrix} K_1 & 0 \\ 0 & K_2 \end{pmatrix} \prod_{i=1}^{m} \begin{pmatrix} 1 & \widetilde{s}_i(z) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \widetilde{t}_i(z) & 1 \end{pmatrix} \tag{3.23}$$

The scaling step can be made to be unity (i.e. 1) by making the determinant of the synthesis polyphase matrix $P(z)$ equal to 1. When this happens, the filter pair (LPF and HPF) is set to be a complementary filter pair doing away with the scaling process. Furthermore, it will make the factors of the filters relative prime. To extract the factors of the filter into lifting steps, the lifting step matrix is laid out as shown in Equation 3.24. Either the predict step or the update step can be factor out first. However, it would more logical to start with the longer filter. In this case (Equation 3.24), extraction starts with the predict step $\widetilde{t}_i(z)$.

$$\widetilde{P}(z) = \begin{pmatrix} \widetilde{Y}_{L1}(z) & \widetilde{LPF}_o(z) \\ \widetilde{Y}_{H1}(z) & \widetilde{HPF}_o(z) \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \widetilde{t}_i(z) & 1(z) \end{pmatrix} = \begin{pmatrix} \widetilde{LPF}_e(z) & \widetilde{LPF}_o(z) \\ \widetilde{HPF}_e(z) & \widetilde{HPF}_o(z) \end{pmatrix} \tag{3.24}$$

With the above equation, the following two equations (3.25 and 3.26) are extracted. As $\widetilde{LPF}_o(z)$ and $\widetilde{HPF}_o(z)$ are known, $\widetilde{Y}_{L1}(z)$, $\widetilde{Y}_{H1}(z)$ and predict step $\widetilde{t}_i(z)$ needs to be solved for. $\widetilde{Y}_{L1}(z)$ is the remainder factor of from analysis low pass filter after the first factoring and $\widetilde{Y}_{H1}(z)$ is the remainder factor from analysis high pass filter after the first factoring. The Euclidean algorithm[1] is used to factorise $\widetilde{HPF}_e(z)$ first with $\widetilde{LPF}_e(z)$ as a_0 and $\widetilde{LPF}_o(z)$ as b_0. Only one step of the Euclidean algorithm needed be executed to find one lifting step each.

$$\widetilde{LPF}_e(z) = \widetilde{t}_i(z)\widetilde{LPF}_o(z) + \widetilde{Y}_{L1}(z) \tag{3.25}$$

[1] see Appendix A for the Euclidean algorithm

$$\widetilde{HPF}_e(z) = \widetilde{t}_i(z)\widetilde{HPF}_o(z) + \widetilde{Y}_{H1}(z) \tag{3.26}$$

Depending on the number of terms in $\widetilde{HPF}_e(z)$, there will be more than one possible factorization. Once a suitable factorization is chosen, we substitute the new found value of $\widetilde{Y}_{L1}(z)$ and $\widetilde{t}_i(z)$ into Equation 3.26 to find $\widetilde{Y}_{H1}(z)$. When all the three unknown are found, they are substituted into Equation 3.27 to solve for $\widetilde{Y}_{L2}(z)$, $\widetilde{Y}_{H2}(z)$ and update step $\widetilde{s}_i(z)$. $\widetilde{Y}_{L2}(z)$ is the remainder factor of from analysis low pass filter after the second factoring and $\widetilde{Y}_{H2}(z)$ is the remainder factor from analysis high pass filter after the second factoring.

$$\begin{pmatrix} \widetilde{Y}_{L1}(z) & \widetilde{Y}_{L2}(z) \\ \widetilde{Y}_{H1}(z) & \widetilde{Y}_{H2}(z) \end{pmatrix} \begin{pmatrix} 1 & \widetilde{s}_i(z) \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \widetilde{Y}_{L1}(z) & \widetilde{LPF}_o(z) \\ \widetilde{Y}_{H1}(z) & \widetilde{HPF}_o(z) \end{pmatrix} \tag{3.27}$$

From Equation 3.27, two equations are again extracted as shown in Equation 3.28 and 3.29. Once again, the Euclidean algorithm is used to factorize $\widetilde{LPF}_o(z)$ and $\widetilde{HPF}_o(z)$ to solve for $\widetilde{Y}_{L2}(z)$, $\widetilde{Y}_{H2}(z)$ and update step $\widetilde{s}_i(z)$.

$$\widetilde{LPF}_o(z) = \widetilde{s}_i(z)\widetilde{Y}_{L1}(z) + \widetilde{Y}_{L2}(z) \tag{3.28}$$

$$\widetilde{HPF}_o(z) = \widetilde{s}_i(z)\widetilde{Y}_{H1}(z) + \widetilde{Y}_{H2}(z) \tag{3.29}$$

When $\widetilde{Y}_{L2}(z)$, $\widetilde{Y}_{H2}(z)$ and update step $\widetilde{s}_i(z)$ are found and the leftmost matrix shown in Equation 3.30 is a unity matrix or a diagonal matrix with only integers (like the constant K_1 and K_2 in Equation 3.31), the factoring process is completed. If the remaining terms in the matrix are polynomials, factoring has to continue until the above-mentioned condition is met.

$$\widetilde{P}(z) = \begin{pmatrix} \widetilde{Y}_{L1}(z) & \widetilde{Y}_{L2}(z) \\ \widetilde{Y}_{H1}(z) & \widetilde{Y}_{H2}(z) \end{pmatrix} \begin{pmatrix} 1 & \widetilde{s}_i(z) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \widetilde{t}_i(z) & 1 \end{pmatrix} \tag{3.30}$$

With these steps, the sub-sampled outputs of the LPF ($\widetilde{L_1}$) and HPF ($\widetilde{H_1}$) in relation to the input can be directly written as shown in Equation 3.31. From this equation, two sets of lifting-based equations can be drawn out, one for the LPF (odd) terms ($\widetilde{L_1}$) and the other for the HPF (even) terms ($\widetilde{H_1}$).

$$\begin{pmatrix} \widetilde{L}_1(z) \\ \widetilde{H}_1(z) \end{pmatrix} = \begin{pmatrix} K_1 & 0 \\ 0 & K_2 \end{pmatrix} \prod_{i=1}^{m} \begin{pmatrix} 1 & \widetilde{s}_i(z) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \widetilde{t}_i(z) & 1 \end{pmatrix} \begin{pmatrix} x_e(z) \\ z^{-1}x_o(z) \end{pmatrix} \tag{3.31}$$

Using Equation 3.31, starting from the right side matrix to the left side matrix, the wavelet transform can be implemented as shown in Figure 3.8

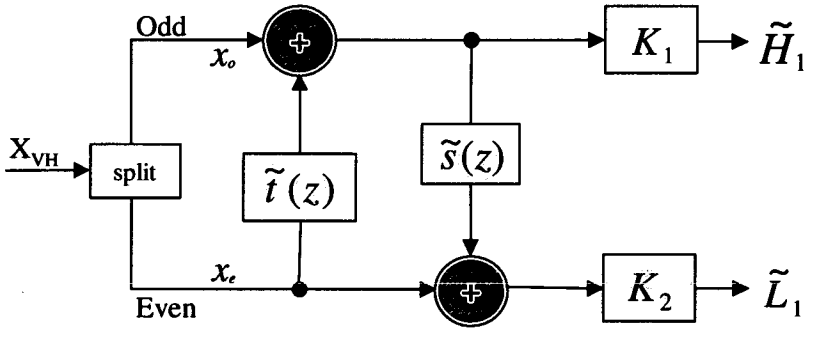


Figure 3.8: *Lifting-based DWT implementation diagram*

3.4 Image Compression Standards

Having a good image compression technology is not good enough for it to be widely utilised. For image compression to be widely used an image and video standard is needed, so that systems anywhere are able to know the compression convention and protocol to decode and decompress the compressed image and video data. An image/video standard with a standardized method of encoding and decoding image/video data will ensure that a digital

image or video system built anywhere in the world is able process video data. The following sub-sections are summary of current widely used international standards for image and video coding.

There are many image and video standards in existence. These standards provide generic specifications on a range of image and video compression coding and systems. These standards range from digital still image compression coding systems to stand alone video compression coding to a whole image/video compression coding system architecture. There are two main international standardization organisations responsible for standardizing image and video compression coding and systems. These organisations are the International Organisation for Standardization and International Electrotechnical Commission (ISO/IEC) and the International Telecommunication Union (ITU). Standards from ITU are normally termed as recommendations. ISO/IEC standards are usually design to address the needs of image/video storage, broadcasting video, image propagation and video streaming over the internet. Whereas, the ITU-T recommendations address the requirement for real-time video communication applications such as video conferencing and video telephony. Image and video standards and recommendations by ISO/IEC and ITU are usually denoted as JPEG/JPEG2000 or MPEG-x (MPEG-1, MPEG-2 and MPEG-4) and ITU-T H26x (e.g. H.261, H.262, H.263 and H.264) respectively. JPEG stands for Joint Photographic Experts Group and MPEG stands for Moving Picture Expert Group. There are so far two main types of compression coding technology adopted by these standard bodies. The two compression technologies are the discrete cosine transform (DCT) and the discrete wavelet transform (DWT) based coding technology.

3.4.1 Discrete Cosine Transform (DCT) Based Standards

JPEG (ISO/IEC 10928-1) [145] is an international standard for compressing continuous tone greyscale and colour digital still image data. It is an DCT based image compression coding system standards developed by the ISO/IEC. JPEG standards allow progressive image transmission and hierarchical encoding with the lossless coding option (JPEG-LS (ISO/IEC 14495-1) [146]).

Besides the DCT based image coding standard, there is also DCT based video compression coding for real-time video communication. ITU-T H.261 [147] is one such recommendation put forward by ITU for video compression coding. ITU-T H.263 [148] is a more recent

ITU recommendation which largely replaced H.261. The technology described in H.263 can deliver reasonable high quality video streaming at a very low bit rate and bandwidth. There are two further development works on the enhancing of the performance of H.263. The H.263+ and H.263++ are results of this enhancement development work. Both of these two recommendations are considered as a short-term improvement to video compression coding. A long-term solution for a significantly better low-bit rate video compression coding is drafted into the ITU-T H.264 [149]. ITU-T H.264 is more well-known by its old working title H.26L and is identical to ISO/IEC standard MPEG-4 Part 10 Advance Video Coding (AVC) (ISO/IEC 14496-10) [150] as it is jointly developed by the two standard organisations.

A series of video (with audio) compression coding systems for storage and transmission were specified by ISO/IEC in MPEG-1 (ISO/IEC 11172) [151], MPEG-2 (ISO/IEC 13818) [152], and MPEG-4 (ISO/IEC 14496) [153] standards. The MPEG-1 standard was drafted for the coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s. The MPEG-2 standard generally can be regarded as an improvement built upon the MPEG-1 specification. The MPEG-2 was also jointly developed with ITU which resulted in the ITU recommendation ITU-T H.262 [154]. The purpose of MPEG-2 was to standardize the generic coding of moving pictures and associated audio information. MPEG-4 is similar to MPEG-2 (from part 1 to 5) except that it has other added features which are for the coding of audio-visual objects. An unique feature of MPEG-4 is that it can code individual video information and objects that might not be rectangular in shape.

All the DCT based video standards cited in the above utilise inter-frame prediction, motion compensation technology which yields very good compression efficiency. However, as motion compensation technique depends heavily on the previously sent reference image frame, prominent errors are very like to propagate down several prediction frames if the reference frame is corrupted during transmission. Hence, such video technology and standards are not robust enough in error and noise prone environments, making them unsuitable for mission critical applications. Beside this, such video technology and standards are not suitable for professional high-quality film/video mastering and editing as such application needs editing of each and every real frame of video images. Thus, these reduce the application area of such video technology and standards.

3.4.2 Discrete Wavelet Transform (DWT) Based Standards

Although many of the standards are DCT based, there is a standard that uses DWT as the core or part of the coding systems. JPEG2000 (ISO/IEC 15444) [1] is one such standard defined by ISO/IEC that uses DWT as the core of it compression coding. JPEG2000 is a standard for compressing both digital still image and motion pictures (Motion-JPEG2000 ISO/IEC 15444-3 [155]). Both lossy and lossless compression are provided for in the JPEG2000 standard. Besides being a DWT based standard, JPEG2000 is also fundamentally different in the way it codes motion pictures. JPEG2000 does not employ motion compensation technology for motion picture coding and is essentially an intra-frame coding system. As such its compression efficiency for motion pictures is much lower than the DCT based coding. However, as it does not employ inter-frame motion compensation technique it is suitable for mission critical applications in error-prone environments. As such, the usage of such coding systems had been widens.

JPEG2000 provides a number of mechanisms for locating and extracting data for the purpose of retransmission, storage, display, or editing, it is very suitable for the following application examples: digital still cameras, error-prone environments such as wireless and the internet, PC-based video capturing, high quality digital video recording for professional broadcasting and motion picture production (and editing) from film-based system, remote surveillance, high-resolution medical and satellite imaging/archiving.

Besides the JPEG2000, MPEG-4 (ISO/IEC 14496) also uses the DWT to compress still image (texture) for various applications from simple still digital photos to photo realistic 3D models to animated meshes. MPEG-4 uses a Daubechies (9,3) tap biorthogonal filter bank and does not provide for lossless compression. The coding efficiency of JPEG2000 is better than MPEG-4 still texture coding in a number of ways [156]. JPEG2000 is superior to the MPEG-4 still texture coding in the area of compression efficiency, progressive image transmission, image quality and processing time.

3.5 Computational Load of Image Compression

Image compression process is particularly computational intensive because of the sheer amount of image data in a single image and the amount of computational operations needed to perform on each of them. Consider the following numerical estimation of the computational load (in

term of number of operations) of processing a 256 × 256 greyscale image with both DCT and DWT (5/3) transformation process (without fast or optimised algorithm). The 5/3 DWT is the reversible integer filter pair defined in the JPEG2000 standard. The 5/3 filter pair has a 5 tap LPF analysis filter and a 3 tap HPF analysis filter. The detail description of the JPEG2000 standard is presented in Section 4.2.1 of Chapter 4. The following calculation is for the computational load for 2-D DCT without blocking.

$$\begin{aligned} DCTcl_{256} &= (\text{no. of freq. coefficients}) \times (\text{no. of ops} \times \text{no. of pixels}) \quad (3.32) \\ &= (256 \times 256) \times (2 \times 256 \times 256) \\ &= 8589934592 \\ &\approx 8590 \text{ MOP} \quad (3.33) \end{aligned}$$

The following calculation is for the computational load for 2-D DCT with blocking. Assuming that the blocking size is 16 × 16, then the total amount of computational load required for the 2-D DCT process to compute a 256 × 256 image with blocking is as follows.

$$\begin{aligned} DCTclb_{256} &= (\text{no. of freq. coefficients}) \times (\text{no. of ops} \times \text{no. of pixels in block}) \\ &\quad \times (\text{no. of blocks in a frame}) \quad (3.34) \\ &= (16 \times 16) \times (2 \times 16 \times 16) \times \frac{(256 \times 256)}{(16 \times 16)} \\ &= (2 \times 16 \times 16) \times (256 \times 256) \\ &= 33554432 \\ &\approx 34 \text{ MOP} \quad (3.35) \end{aligned}$$

Consider also the computational load of the 5/3 DWT process. Assuming a 3 levels DWT process. The calculation of the computational load of a 5 levels 5/3 DWT process is shown as follows.

$$
\begin{aligned}
DWT5/3_{256} = \; & \{[2\text{D} \times \text{no. of ops} \times \text{no of HPF taps} \times \text{no. of pixels in frame(lvl1)}]+ \\
& [2\text{D} \times \text{no. of ops} \times \text{no of HPF taps} \times \text{no. of pixels in frame(lvl2)}]+ \\
& [2\text{D} \times \text{no. of ops} \times \text{no of HPF taps} \times \text{no. of pixels in frame(lvl3)}]+ \\
& [2\text{D} \times \text{no. of ops} \times \text{no of HPF taps} \times \text{no. of pixels in frame(lvl4)}]+ \\
& [2\text{D} \times \text{no. of ops} \times \text{no of HPF taps} \times \text{no. of pixels in frame(lvl5)}]+ \\
& [2\text{D} \times \text{no. of ops} \times \text{no of LPF taps} \times \text{no. of pixels in frame(lvl1)}]+ \\
& [2\text{D} \times \text{no. of ops} \times \text{no of LPF taps} \times \text{no. of pixels in frame(lvl2)}]+ \\
& [2\text{D} \times \text{no. of ops} \times \text{no of LPF taps} \times \text{no. of pixels in frame(lvl3)}]+ \\
& [2\text{D} \times \text{no. of ops} \times \text{no of LPF taps} \times \text{no. of pixels in frame(lvl4)}]+ \\
& [2\text{D} \times \text{no. of ops} \times \text{no of LPF taps} \times \text{no. of pixels in frame(lvl5)}]\} \qquad (3.36) \\
= \; & \{[2 \times 2 \times 3 \times (256 \times 256)] + [2 \times 2 \times 3 \times (128 \times 128)] + \\
& [2 \times 2 \times 3 \times (64 \times 64)] + [2 \times 2 \times 3 \times (32 \times 32)] + \\
& [2 \times 2 \times 3 \times (16 \times 16)]\} + \{[2 \times 2 \times 5 \times (256 \times 256)] + \\
& [2 \times 2 \times 5 \times (128 \times 128)] + [2 \times 2 \times 5 \times (64 \times 64)] + \\
& [2 \times 2 \times 5 \times (32 \times 32)] + [2 \times 2 \times 5 \times (16 \times 16)]\} \\
= \; & (786432 + 196608 + 49152 + 12288 + 3072) + \\
& (1310720 + 327680 + 81920 + 20480 + 5120) \\
= \; & 1047552 + 1745920 \\
= \; & 2793472 \\
\approx \; & 3 \text{ MOP} \qquad (3.37)
\end{aligned}
$$

It can be seen from Equations 3.33, 3.35 and 3.37 that image transform processes in direct

form are computationally very intensive and hence very power consuming. For useful portable image and video applications to be viable, it is important to have a low power image and video compression system. As such, techniques are needed to reduce the power consumed by image compression coding. From the two DCT computational load values shown in Equation 3.33 and 3.35, it can be seen that a simple algorithmic management like blocking can save great amount of computational steps (thus power). Therefore, with effective power techniques significant amount of power can be saved.

3.6 Summary

This chapter has introduced briefly the principles of image compression techniques along with two of the commonly used image compression techniques. The discrete cosine transform (DCT) and sub-band coding based compression are the two most commonly used compression transform techniques. The DCT converts image data samples from one domain to another domain in order to alter the distribution of the luminance level values so that most of them can be eliminated or be quantised with very few bits. Whereas, the sub-coding splits the image data sample into different bands of frequency and concentrates its compression technique at the low frequency end in which most of the energy of an image resides. Then, the development of the wavelet and discrete wavelet transform (DWT) were briefly presented. The conventional implementation of the DWT which is manifested as a sub-band coding is also described. However, a conventional convolution-based DWT is computationally intensive and area wasting. Therefore, the lifting based DWT algorithm along with its implementation were introduced and described. Having a good image compression technology is not good enough for such technology to be widely utilised. For an image compression to be widely used, a image or video standard is needed so that a digital image or video system built anywhere in the world is able to store and process the same stream of image or video data. As such, a review of existing popular image and video compression standards by two main groups of international standardization organisation (ITU and ISO/IEC) was given. Finally, the case to reduce the power of a digital image and video compression system was made.

Chapter 4

JPEG2000 5/3 Lifting Based DWT Implementation

4.1 Introduction

There has been significant interest in producing efficient Discrete Wavelet Transform (DWT) algorithm for the realisation of DWT hardware. This is because the conventional convolution based DWT implementation [4] is computationally intensive and both area and power hungry. The introduction of the lifting based scheme DWT [5, 6] overcomes these traditional problems. As such the lifting-based scheme has been chosen in the JPEG2000 [1] standard (ISO/IEC15444-1). Such developments aroused considerable interest in the implementation [2, 3, 157] of this algorithm. Recent implementations [2, 7] of the JPEG2000's lifting-based DWT are pipelined data-path centric designs and no ALU-centric architecture was proposed. These pipelined data-path centric designs are inflexibly hardwired and require more arithmetic logic hardware. Furthermore, they assume a horizontal (row) feed and process first which does not fully comply with JPEG2000's specification.

Therefore, this chapter presents a novel JPEG2000's 5/3 lifting based DWT hardware architecture that has significantly lower hardware count, in terms of arithmetic logic units and memory compared to the architecture presented in [2, 3]. This proposed architecture, built around parallel Shift-Accumulator Arithmetic Logic Units (ALUs) (with implicit embedded extension [158] and Chapter 5), has an efficient memory organisation that uses a smaller amount of memory for processing and buffering. It also can encode up to five levels of transformation (at one level at a time). This architecture can start its computation in either the row or column direction depending on the orientation of image vectors that are fed into its input. However, in this thesis the computation is assumed to commence in the column-row direction to adhere to the JPEG2000 specification.

This chapter is organised as follows. In Section 4.2, the JPEG2000's 5/3 lifting based DWT

algorithms, the existing JPEG2000 lifting based architectures and the organisation of its components and their workings are introduced and described. In Section 4.3, the proposed architecture and its main components and workings are presented. Comparisons of the proposed architecture to the two existing lifting based architecture are presented in Section 4.4. Finally, the summary of this chapter is given in Section 4.5.

4.2 JPEG2000 5/3 Lifting-Based DWT and Implementation

Conventional DWT video compression is usually employed as a dyadic sub-band coding process. Many parts of these conventional processes are implemented as convolution based Finite Impulse Response (FIR) filters for both the high and low band processing. These processes waste computing power and memory space on processing and storing data vectors, half of which will be dropped during the down-sampling or sub-sampling later on in the process. Such drawbacks make the conventional convolution based DWT very resource intensive and unattractive for hardware implementation. However, most of the redundancy in the DWT system can be eliminated or minimized by the lifting based DWT as established in Chapter 3.

The lifting based DWT reduces computational requirement by taking into account the redundancy of down-sampling and avoiding computation of the output vectors that will eventually be dropped. The lifting based algorithm also exploits the similarities between the low-pass filter (LPF) and high-pass filter (HPF) to further reduce redundancies. With the lifting based DWT, more than 50% of the computational requirement can be reduced. Therefore, the lifting based DWT has been specified by the JPEG2000 standard as its core compression algorithm. The following is a brief description of the JPEG2000 5/3 lifting-based DWT and its existing implementations. Refer to Section 3.3 of Chapter 3 for the mathematical notation used in the following sub-sections.

4.2.1 JPEG2000 5/3 Lifting-Based DWT

JPEG2000 standard [1] specified two types of DWT filters; the reversible integer 5/3 filter [8] and the irreversible floating point Daubechies 9/7 filters [9]. In this research, the reversible integer 5/3 filter was found to be inherently low power and less complex than the 9/7 filter. This is because the 5/3 filter has a shorter filter length for both the low pass and the high pass filter compared to the 9/7 filter. Furthermore, the 5/3 filter coefficient tap values are all fixed point

integer coefficients with a common factor of $\frac{1}{8}$ (3 right shift), whereas 9/7 coefficients are all floating point numbers. Even though with the Fast Integer 9/7 DWT [134] algorithm, 9/7 is still far more complex than 5/3 DWT. In addition, 5/3 implemented with a lifting scheme has one set of lifting steps compare to 9/7, which has two. These characteristics make 5/3 filters very suitable to be implemented for applications where resource and power consumption is a major consideration. Therefore, the 5/3 filter is chosen as the candidate for this low power implementation research.

$$\widetilde{HP} : Y(n) = -\frac{1}{2}X_{ext}(n-1) + X_{ext}(n) - \frac{1}{2}X_{ext}(n+1) \tag{4.1}$$

$$\widetilde{LP} : Y(n) = -\frac{1}{8}X_{ext}(n-2)+\frac{2}{8}X_{ext}(n-1)+\frac{6}{8}X_{ext}(n)+\frac{2}{8}X_{ext}(n+1)-\frac{1}{8}X_{ext}(n+2) \tag{4.2}$$

The transfer functions of 5/3 DWT analysis filters of HPF and LPF are shown in Equations 4.1 and 4.2 respectively. The output of the LPF is denoted as $\widetilde{LP}$ and HPF as $\widetilde{HP}$ (as defined in Section 3.3.2 of Chapter 3). The term $Y(n)$ is the discrete output of the transfer function with n as the sample index. The term $X_{ext}(n)$ represent the extended input data. There are 5 taps in LPF analysis filter and 3 taps in the HPF analysis filter. The tap values of the 5/3 analysis filter pair are shown in Table 4.1.

Tap number	Analysis Filter Coefficients		Synthesis Filter Coefficients	
	LPF	HPF	LPF	HPF
0	$\frac{6}{8}$	1	1	$\frac{6}{8}$
±1	$\frac{2}{8}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{2}{8}$
±2	$-\frac{1}{8}$			$\frac{1}{8}$

Table 4.1: *5/3 DWT tap coefficient value*

These original 5/3 DWT transfer functions (Equation 4.1 and 4.2) can be factored to obtain Equation 4.3 and 4.4 for odd terms ($\widetilde{HP}$) and even terms ($\widetilde{LP}$) respectively. It is noteworthy that the outputs of the lifting-based 5/3 DWT are now indexed by either odd or even terms.

$$\widetilde{HP} : Y(2n+1) = X_{ext}(2n+1) - \left\lfloor \frac{X_{ext}(2n) + X_{ext}(2n+2)}{2} \right\rfloor \tag{4.3}$$

$$\widetilde{LP} : Y(2n) = X_{ext}(2n) + \left\lfloor \frac{Y_{ext}(2n-1) + Y_{ext}(2n+1) + 2}{4} \right\rfloor^{1} \tag{4.4}$$

Equations 4.5 and 4.6 show the lifting-based 5/3 inverse (synthesis) DWT equation for odd terms ($X(2n+1)$) and even terms ($X(2n)$) respectively.

$$X(2n+1) = Y_{ext}(2n+1) + \left\lfloor \frac{X(2n) + X(2n+2)}{2} \right\rfloor \tag{4.5}$$

$$X(2n) = Y_{ext}(2n) - \left\lfloor \frac{Y_{ext}(2n-1) + Y_{ext}(2n+1) + 2}{4} \right\rfloor \tag{4.6}$$

Even though the lifting-based 5/3 DWT significantly reduces the resource and power consumption, there is still room for further reduction in area and computation (hence the power) in the implementation of this transformation. By careful design, redundancies can be further reduced.

4.2.2 Existing JPEG2000 Lifting-Based DWT Implementation

As the JPEG2000 lifting-based DWT is very suitable for implementation, there are two hardware realisations of this algorithm to date. Both of the implementations are pipelined data-path centric architectures. Andra et. al. proposed a conventional pipelined ALU architecture [2] and Lian et. al. [7] proposed a folded architecture for their processor design. These two architectures were chosen as the candidate for comparison as these are the only two known JPEG2000's lifting-based 5/3 DWT architecture to date. Overview of these architectures are briefly described in the following subsections.

[1] JPEG2000's 5/3 even terms have an additional constant term of $\frac{2}{4}$ or (0.5) added to it

4.2.2.1 Andra's Lifting-Based DWT Architecture

Andra's lifting based DWT architecture [2] (shown in Figure 4.2) is centred on four finely pipelined feed-through data-path ALU processors as shown in Figure 4.1. The processors shown in Figure 4.1 are simplified and tuned for the 5/3 transform[2]. As the lifted-based 5/3 coefficients are $\frac{1}{2}$ and $\frac{1}{4}$, the multiplier can be replaced with a right shifter. The processors are grouped into modules with two processors in each module. The module that is responsible for processing the input data sample in the horizontal direction is known as the *Row module*. And the *Column module* is responsible for processing the input samples in the vertical directions. Alongside with the two processing modules are two memory modules. The memory modules are for buffering the incoming input sample data (MEM1), the processed intermediate data (MEM2) and the processed LL sub-band data for transformation at the next level (MEM1). The size[3] of MEM1 is $N \times \frac{N}{2}$ and the size of MEM2 is 4N. The total number of memory required to process a whole image frame with 5/3 DWT is $[(N \times \frac{N}{2}) + 4N]$.

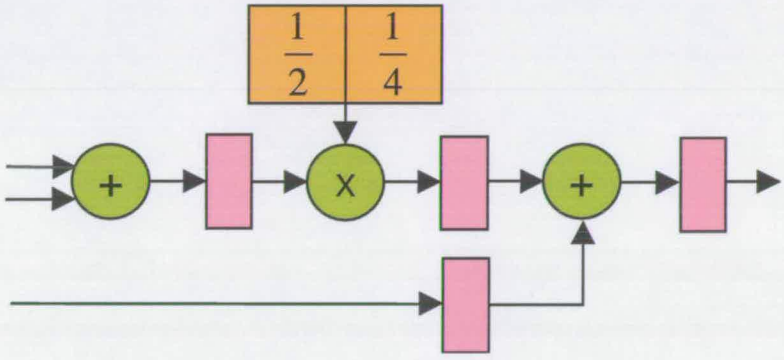


Figure 4.1: *Simplified Andra's lifting based 5/3 DWT processor architecture modified from [2]*

Due to these memory buffers and two processing modules, the architecture is a self-contained module which is capable of multi-level processing of the image sample in 2-D without the need of accessing external memory. The dataflow of this architecture is shown in Figure 4.3. Latency for the architecture to process a frame in 2-D for one level is $[(2 \times N) + 1]$ clock cycles.

[2]To see the original processor architecture refer to [2]
[3]N is the number of pixels in a row or column of image

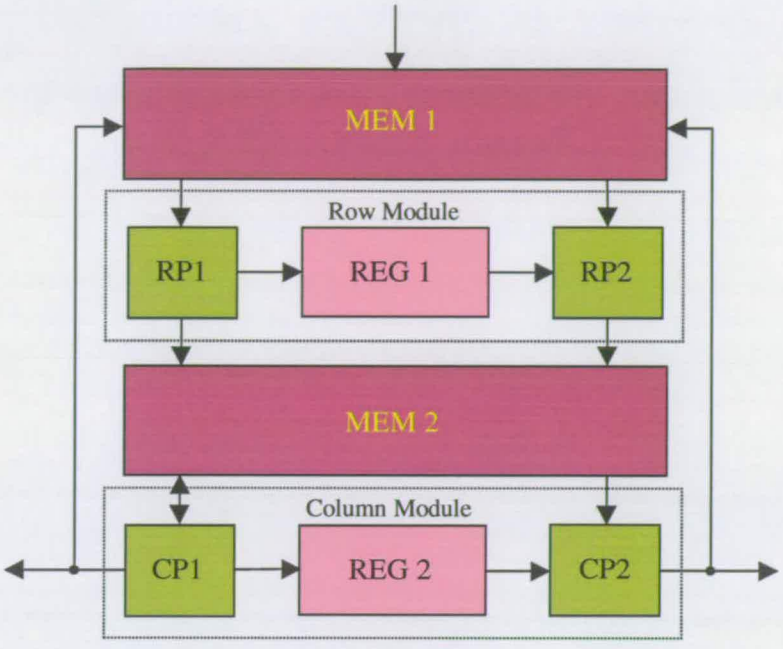


Figure 4.2: *Andra's lifting based DWT overall architecture block diagram [2]*

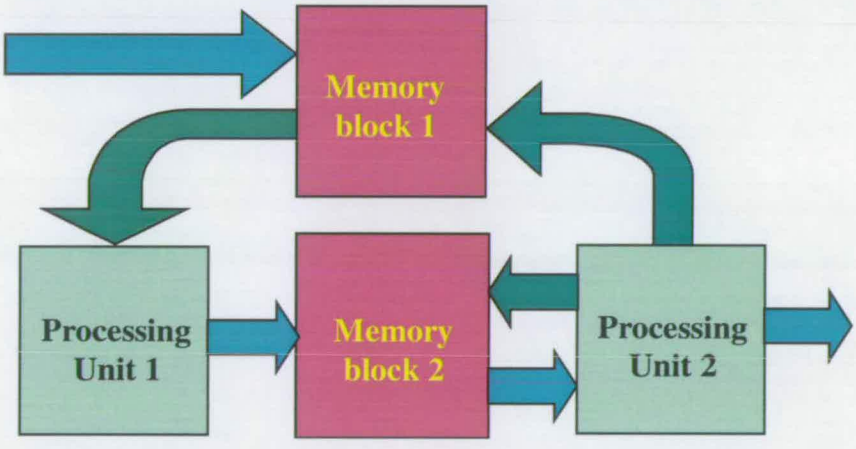


Figure 4.3: *Andra's JPEG2000 lifting based DWT overall dataflow diagram modified from [2]*

4.2.2.2 Lian's Lifting-Based DWT Architecture

Lian's lifting based DWT architecture (shown in Figure 4.5) centres on a single pipelined feed-through folded data-path ALU processor architecture as shown in Figure 4.4. The folded architecture shown in 4.4 is simplified to suit only the 5/3 transform[4]. As the coefficients of the 5/3 are in the order of 2, the multiplier in the folded processor architecture can be replaced by

[4]To see the original processor and system architecture refer to [3]

a right shifter.

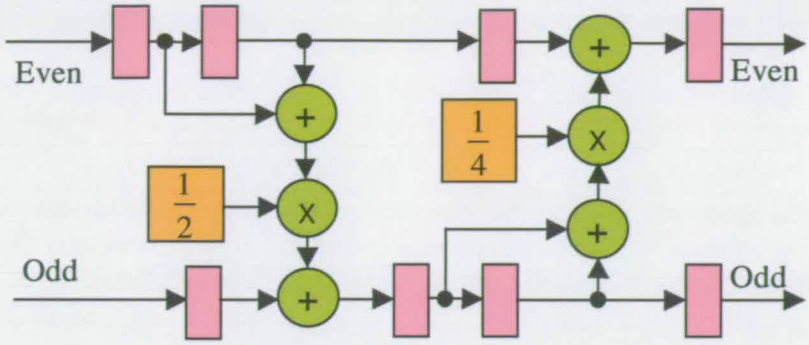


Figure 4.4: *Simplified Lian's lifting based 5/3 DWT processor architecture modified from [3]*

As the whole lifting based DWT system is built around a system bus as shown in Figure 4.5, this architecture only needs one processor and one set of memory modules (Frame Memory) to process in 2-D. This is done by storing the processed intermediate data back into the Frame memory. Therefore, the total number of memory needed to process a whole frame is $N \times N$. A single frame of image represents a single full $N \times N$ image.

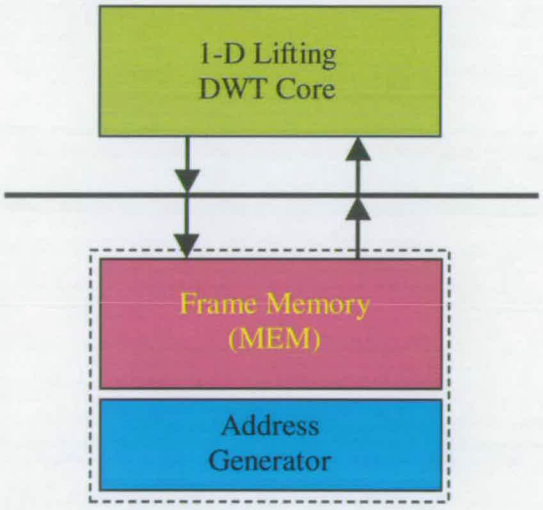


Figure 4.5: *Simplified Lian's JPEG2000 lifting based DWT overall architecture block diagram modified from [3]*

Although Lian's architecture uses only a single set of processor and memory modules, it can still perform multi-level of analysis. This is again by storing the 2-D processed data back into the frame memory to be reprocessed at the next level after the present level of processing is completed. As the Lian's DWT architecture was built around a system bus, the dataflow of the

overall system is simple as shown in Figure 4.6. Latency for a 2-D process is $2(N \times N)$ clock cycle. This is because this architecture needs to buffer the whole frame of image $(N \times N)$ before the vertical processing and the horizontal processing (after the vertical processing) can start.

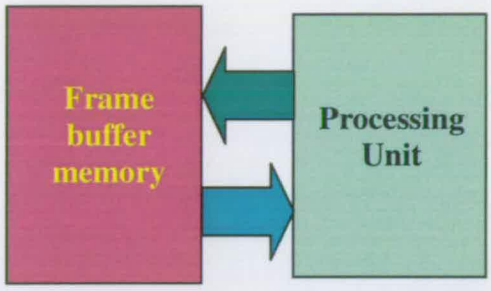


Figure 4.6: *Lian's JPEG2000 lifting based DWT overall dataflow diagram*

4.3 Proposed Shifter-Accumulator ALU Centric Lifting Based DWT Architecture

Both the existing architectures presented in the above sections are pipelined data-path centric lifting based DWT architectures. These pipelined data-path centric architectures are relatively simple to build, especially in term of the control unit as it works merely by clocking the pipeline registers. However, it needs more arithmetic logic hardware to realise the same amount of arithmetic functions as compare to an ALU centric architecture. A pipelined data-path centric architecture requires one arithmetic hardware unit for each arithmetic operation. This extra arithmetic hardware might contribute to the power consumption of the system. The only way of reducing this hardware requirement is to minimize the computation steps at the algorithm level. In addition, this type of architecture is hardwired and less programmable as compare to ALU centric architecture. Thus, such architectures are inflexible and cannot be be easily made to accommodate algorithm changes or add on any algorithm (e.g. incorporating an innovative low power accessing scheme) if the need arises.

With such restrictions, the pipelined data-path centric architecture may not be suitable for low power implementation. Therefore, an alternative architecture is needed to overcome these limitations. An ALU centric architecture is one such architecture that does not have these restrictions. Hence, the novel Shifter-Accumulator ALU (SA-ALU) centric lifting based

DWT architecture is proposed. The overview of proposed SA-ALU architecture as shown in Figure 4.7. Note that there are 2 SA-ALUs in ALU2.2 as indicated by a dotted line in the middle of ALU2.2 in Figure 4.7. The details of the processor architecture is presented in Section 4.3.2. The overall dataflow of the proposed architecture is as shown in Figure 4.8. This lifting based DWT architecture processes an image in 2-D in one iteration.

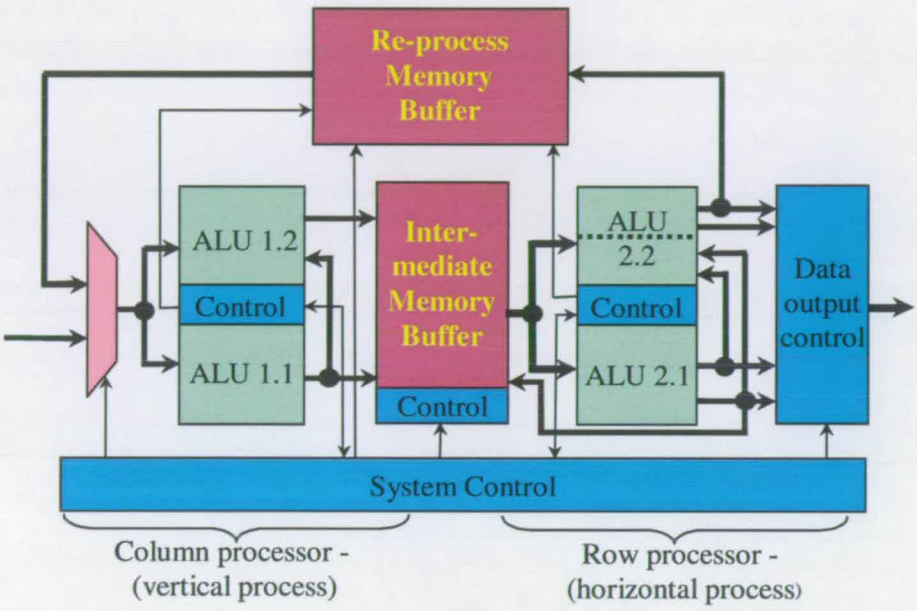


Figure 4.7: *Proposed SA-ALU centric JPEG2000 5/3 DWT overall architecture block diagram*

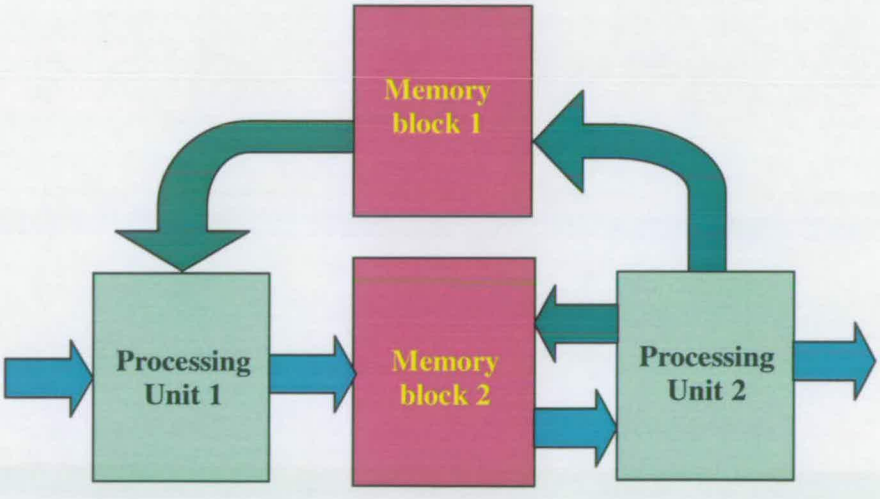


Figure 4.8: *Proposed SA-ALU centric JPEG2000 5/3 DWT overall dataflow diagram*

The architecture consists of the following 3 main building blocks: -

1. Control Unit (CON)
2. SA-ALU processor
3. Buffering Memory (MEM)

The following sub-sections describe the structure and the workings of the 3 main building blocks of the proposed SA-ALU centric lifting based DWT architecture.

4.3.1 Control Unit

There are two levels of control in this architecture, namely:-

1. System control
2. Local control

The system control is responsible for regulating the whole process flow of the 2-D transform which includes the number of analysis levels, coordinating the input data samples and processed data output. The system control is also responsible for the dataflow shown in Figure 4.8. To be in compliance with the specification laid out in the JPEG2000 standard, the system control was designed to process the data in the column-row direction. Besides the system control there are numerous local controls distributed in the architecture. Each of the main components depicted in Figure 4.7 has its own local control. All of these individual controls coordinate and synchronise with one another via the system control. Each processor has its own control which is programmed to know when and how many right shifts are to be applied to the chosen vectors. Implicit embedded data extension [158] of the input samples is also incorporated into the local control of the processors. The details of the embedded data extension will be presented in Chapter 5.

4.3.2 Shift-Accumulator ALU (SA-ALU) Processors

The proposed architecture computes the input image vectors in 2-dimensions (2-D) in the column-row fashion. The 2-D transformation of the input image vectors is carried out in the following two main processors: -

1. Column Processor

2. Row Processor

The column processor is responsible for processing the data sample in the vertical direction and the row processor is responsible for processing the data sample in the horizontal direction. At the heart of these two processors is the shift-accumulator ALUs (SA-ALUs). Accumulators were used as part of the core of the processors to reduce the movement of data sample from register to register by storing calculated values in its register. As the lifting-based 5/3 DWT coefficients are factors of base two ($\frac{1}{2}$ and $\frac{1}{4}$ - as seen from Equations 4.3 to 4.6), processors consisting of shifter and adder are sufficient to realise the transform.

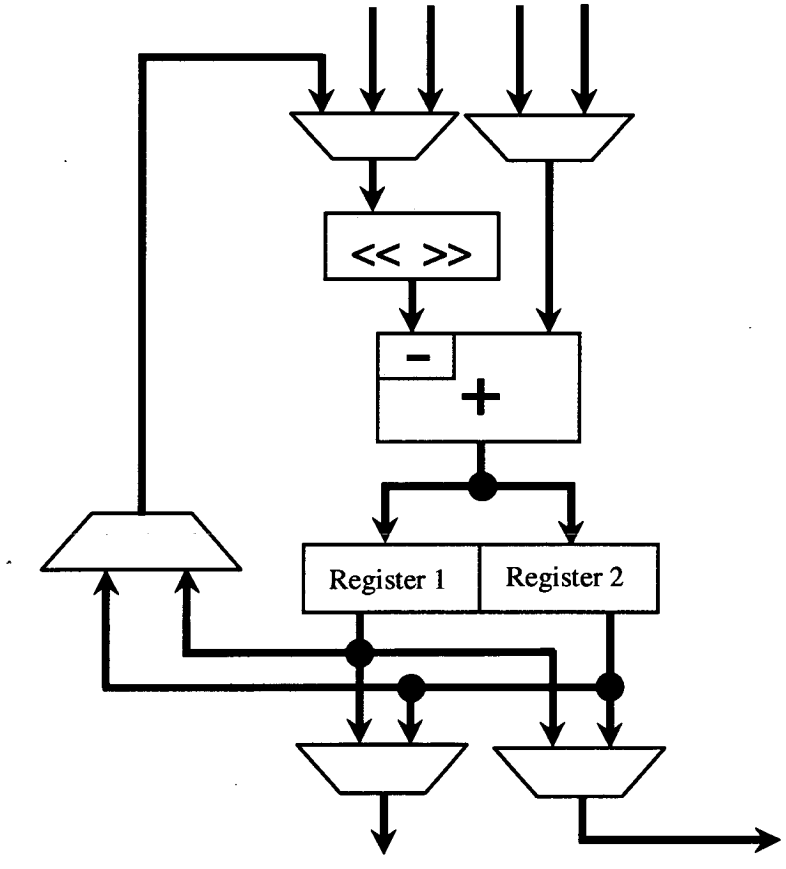


Figure 4.9: *Simplified SA-ALU architecture*

All SA-ALUs in the architecture are based on (or a slight variation of) the ALU architecture shown in Figure 4.9. There are a total of five SA-ALUs in the architecture; two in the column processor and three in the row processor. The proposed SA-ALU centric processor architectures are shown in Figure 4.10 and 4.11, these are the vertical (column) processor and the horizontal (row) processor respectively. There are a total of five SA-ALUs in the architecture; two in the column processor and three in the row processor. The SA-ALU is made-up of the following: -

1. Input multiplexer
2. Shifter unit
3. Adder or Subtractor-adder
4. Accumulator registers
5. Output and feedback multiplexer

The input multiplexers are used to switch the appropriate data (by the processor's control) from three data buffer registers to the ALU. The shifter is used to administer the lifting-based 5/3 DWT coefficients to the input data sample. The coefficient of $\frac{1}{2}$ is realised by applying one right shift and the coefficient of $\frac{1}{4}$ with two right shifts. ALU1.1 and ALU2.1 are odd term ALUs, whereas ALU1.2 and ALU2.2 are even term ALUs. Odd term ALUs have a subtractor-adder each, whereas even term ALUs only utilise an adder. Except for ALUB2, which has two SA-ALUs, all the rest of the ALU units have only one unit of SA-ALU internally. All the adders in the architecture are two data input adders. The row processor is slightly different from the column processor as it has one extra set of SA-ALU. The extra SA-ALU are necessary to make sure that the row processor is fast enough to be in synchronisation with the column processor. These extra SA-ALUs enable the horizontal processor to compute 4 terms simultaneously in one interweaving process stage. Accumulator registers in ALU1.2 and ALU2.2 reset with a value of $\frac{1}{2}$ so that no extra hardware and step is needed to add the extra $\frac{1}{2}$ term.

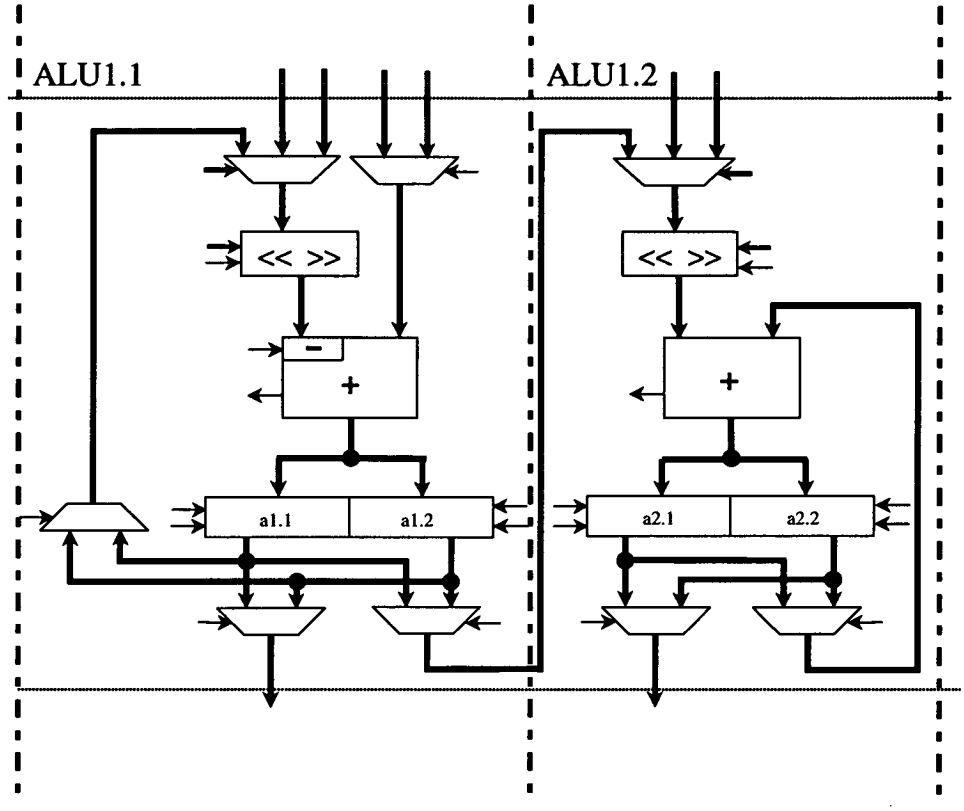


Figure 4.10: *Architecture of SA-ALU vertical (column) processor - ALU1.1 and ALU1.2*

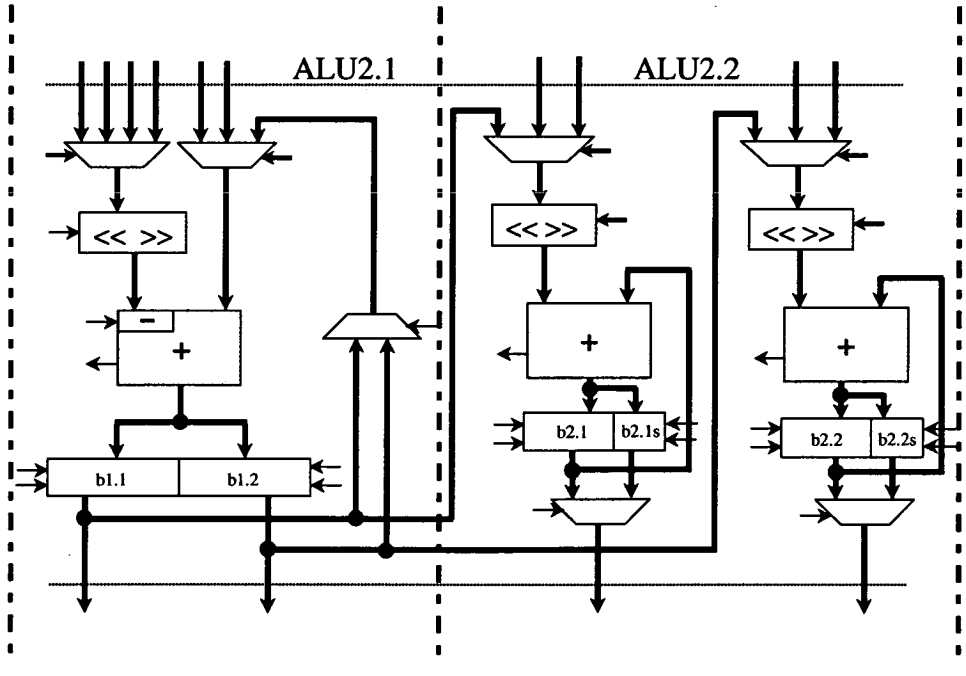


Figure 4.11: *Architecture of SA-ALU horizontal (row) processor - ALU2.1 and ALU2.2*

The feedback multiplexer is used to select the data in the accumulators for further processing or accumulation. Only the odd term ALUs have feedback multiplexer and there are no feedback multiplexer in both ALU1.2 and ALU2.2. Output multiplexers are used to select final accumulated results in either of the accumulator registers for storage in the memories or to the next stage of computation.

4.3.3 Memory Buffer Structure

As the proposed architecture is a line-based DWT [159] architecture, it does not require frame buffers. However, there are two major blocks of memory components in the architecture. These are the Intermediate Memory Buffer (INMEM) and the Re-process Memory Buffer (RPMEM). Intermediate data vectors from both the column processor and the odd outputs of the row processor are stored in the INMEM. RPMEM is used to store LL (refer to Section 3.3.2 in Chapter 3) sub-band outputs data from the row processor for the next level of analysis process. Both of the memory buffers are dual port memory modules and they can be dyadically addressed.

Intermediate Memory

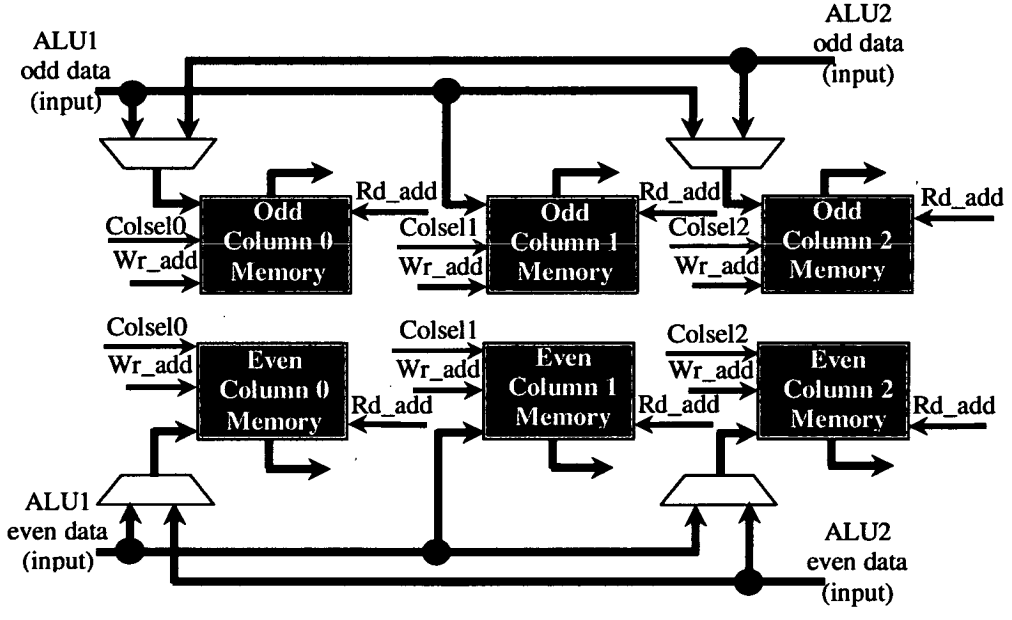


Figure 4.12: *SA-ALU architecture's intermediate memory (INMEM) structure*

The intermediate memory buffers consist of six individual (one port write, one port read)

dual-port memory blocks (see Figure 4.12). The six dual-port memory blocks are grouped into two main blocks, an odd block and an even block, so as to save addressing hardware by sharing its address lines. These memory buffers are designed to store up to three columns of processed data. The write ports of column 0 and 2 memory buffers are connected to the data multiplexer, which switches data between ALU1 and ALU2. Whereas, write ports of column 1 are connected only to ALU1. The write address lines of all the memory blocks are connected to the same write address generator, likewise the read address lines are connected to the same read address generator. Write select signals of each of the memory blocks are generated separately and fed into their respective memory blocks. Read ports of the memory blocks are individually connected to the input multiplexer of ALU2. There is no read control signal for the read ports as these are always read enabled and switched by ALU2 control unit via the input multiplexer of ALU2. The total size of the intermediate memory is $3 \times N$.

Re-process Memory

The re-process memory buffer is implemented with a First-In-First-Out (FIFO) memory. The data input port is connected to the LL band output port of the row processor and the data output port is connected to the multiplexer input of the column processor. ALU2 control unit supplies the write signals and ALU1 control unit supplies the read signals. The size of this re-process memory is $\frac{N}{2} \times \frac{N}{2}$.

4.3.4 Operation Schedule

Coordination between Row and Column Processor

In order to ensure that correct outputs are obtained at all times, all computation steps of the processors have to be determined first. The detailed schedule for the operation of the whole proposed architecture was generated by hand by first listing an efficient base schedule for the first processor. Then, using the first processor's schedule for its timing reference, the schedule of the second processor was drawn out. All the code for scheduling is in pseudo-instruction code as shown in Figure 4.13. Further details of the pseudo-assemble code for Start process, Normal process, End process and the mapping process of pseudo-assemble code to Finite State Machine (FSM) states are shown in Appendix B and C.

Time	Vertical (Column) Processor				Horizontal (Row) Processor					
	ALU1.1 (Odd)		ALU1.2 (Even)		ALU2.1 (Odd)		ALU2.2 (Even)			
	a1.1	a1.2	a2.1	a2.2	b1.1	b1.2	b2.1	b2.1s	b2.2	b2.2s
1	$X_{24}+X_{44} \rightarrow$ a1.2		½+$X_{24} \rightarrow$ a2.2		$Y_{33}-Y_{32}/2 \rightarrow$ b1.1		½+$Y_{32} \rightarrow$ b2.1		b2.2+b1.2/4 →b2.2s ($=Z_{02}$)	
2	X_{34}–a1.2/2 →a1.2 ($=Y_{34}$)		a2.2+a1.1/4 →a2.2		$Y_{23}-Y_{22}/2 \rightarrow$ b1.2		b2.1+Z_{31}/4 → b2.1		½+$Y_{22} \rightarrow$ b2.2	
3	-		a2.2+a1.2/4 →a2.2 ($=Y_{24}$)		b1.1–Y_{34}/2 →b1.1 ($=Z_{33}$)		-		b2.2+Z_{21}/4 → b2.2	
4	-		-		b1.2–Y_{24}/2 →b1.2 ($=Z_{23}$)		b2.1+b1.1/4 →b2.1s ($=Z_{32}$)		-	
5	$X_{44}+X_{64} \rightarrow$ a1.1		½+$X_{44} \rightarrow$ a2.1		$Y_{53}-Y_{52}/2 \rightarrow$ b1.1		½+$Y_{52} \rightarrow$ b2.1		b2.2+b1.2/4 → b2.2s ($=Z_{22}$)	

Figure 4.13: *Pseudo-instruction code of the column and row processor (both in Normal process) for system scheduling - The Red letter terms in Green background are codes that belongs to the next set of Normal process cycle. Accumulator register in Yellow term in Dark Green background indicates that the values in these accumulator registers are the results from the previous clock cycle. In the Vertical Processor column, the accumulator register highlighted in Blue contains the odd term value calculated in the previous process cycle. The Yellow term in Dark Green background indicates the Column processor's present result outputs. In the Horizontal Processor column, the Dark Green terms are codes that belong to the previous process cycle. The terms in Red are previously calculated terms from the Row processor that were stored in the INMEM. The Yellow terms in Blue background are recently calculated terms from the column processor. The present outputs of the Row processor for this process cycle are in Blank letter in Red background.*

The pseudo-instruction code in Figure 4.13 are codes for one cycle of Normal process for both processors. Note that the odd terms are are computed and output first. The red terms in green background in Figure 4.13 are codes that belongs to the next set of Normal process cycle. Accumulator register in yellow terms in dark green background indicates that the values in these accumulator registers are the results from the previous clock cycle. In the 'Vertical (Column) Processor' column, the accumulator registers highlighted in blue contains value that was calculated in the previous process cycle. The yellow terms in dark green background indicates the column processor's present result outputs. In the 'Horizontal (Row) Processor' column, the dark green terms are codes that belong to the previous set of Normal process cycle. The term that is in red are previously calculated terms from the row processor that were stored in the INMEM. The yellow terms in blue background are recently calculated terms from the column processor. The present outputs of the row processor for this process cycle are in black

letters in red background. From the Figure 4.13, it can be seen that the column processor produced the odd and even outputs in the space of one clock cycle apart (in tandem) and there is a large gap between the present set and the next set of odd and even outputs. Hence, a single address bus lines and two (for odd and even lines) separate write enable lines are sufficient to address both odd and even locations of the INMEM at any one set of odd and even outputs.

The following figures show the FSM states transition diagrams of both processors that implements the operation schedule. These states were mapped from the pseudo-assemble codes shown in Appendix B. Figure 4.14 is the FSM states transition diagram of Vertical (Column) processor.

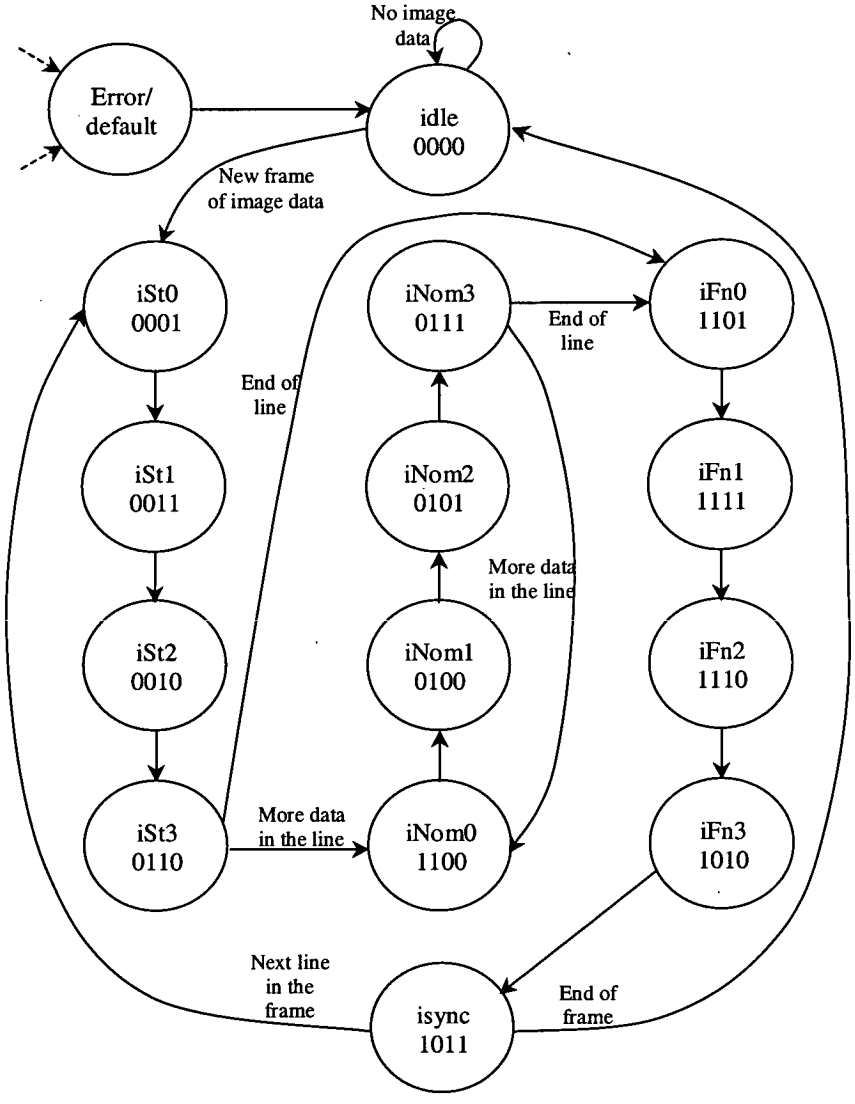


Figure 4.14: *FSM states transition diagram of Vertical (Column) processor*

There are in total of 14 states in the Vertical (Column) processor FSM. There are 4 states each for each of the main (Start, Normal and End) processes. The detail of these processes are deliberated in Chapter 5. The '*iSt*', '*iNom*' and '*iFn*' in the states transition diagram represent the states in the Start, the Normal and End process respectively. The '*idle*' is the idle state which the FSM resets and recovers to. The '*isync*' is the synchronise state which was inserted for the recovery from the main processes. Note that most part of the FSM states (especially within the main processes) proceed from one state to another without any decision or condition. The following Figure 4.15 is the FSM states transition diagram of Horizontal (Row) processor.

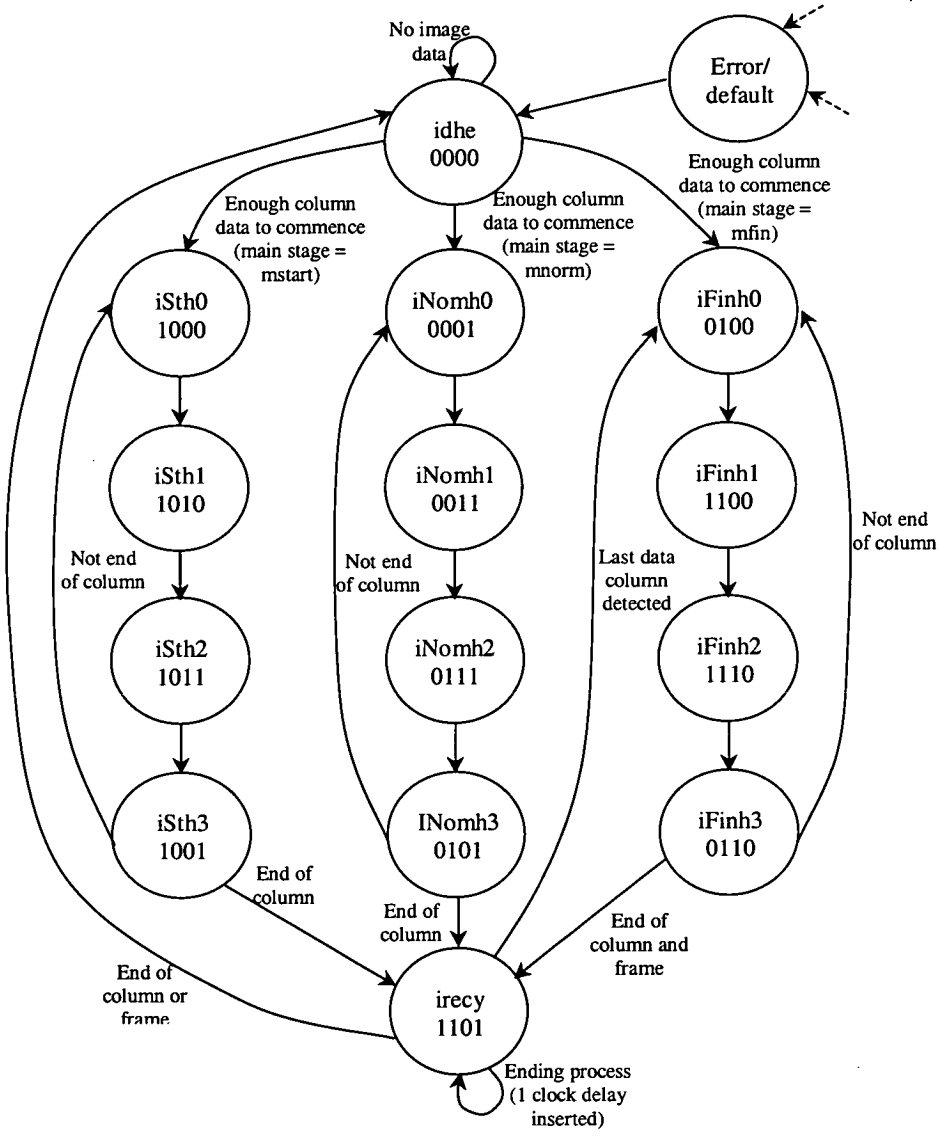


Figure 4.15: *FSM states transition diagram of Horizontal (Row) processor*

Like the Vertical (Column) processor FSM, there are in total of 14 states in the Horizontal (Row) processor FSM. There are 4 states each for each of the main (Start, Normal and End) processes. The '*iSth*', '*iNomh*' and '*iFnh*' in the states transition diagram represent the states in the Start, the Normal and End process respectively. The '*idhe*' is the idle state which the FSM resets and recovers to. The '*irecy*' is the recovery state so that the FSM can recover from the main processes before going back to the '*idhe*' state. Like the Vertical (Column) processor FSM, most part of the FSM states (especially within the main processes) proceed from one state to another without any decision or condition. However, all the main processes are launched from the '*idhe*' state and end with the '*irecy*'. As all the main processes are launched from the '*idhe*' state and do not follows the flow of the states (as seen in the Vertical (Column) processor FSM), the Horizontal (Row) processor FSM needs to know which main process to launch into. The Horizontal (Row) processor FSM knows which process to launch into by referring to another (main) state register which the Horizontal (Row) processor FSM updates every time it enter or exit a main process. The states transition diagram of the main state register is as shown in Figure 4.16.

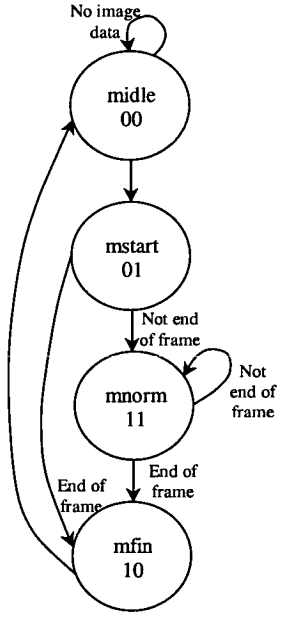


Figure 4.16: *Main states transition diagram of Horizontal (Row) processor state register*

Coordination between The Processors and The Intermediate Memory (Inmem) Buffers

At the beginning of each frame, the process first starts with the column processor. The output data from the column processor are stored into the INMEM in a column by column fashion.

When a column of memory block is filled, the next adjacent column of memory block will be used to store the subsequent streams of data. Starting from the column on the left, the writing of the intermediate data will move right to the centre column when the left column is filled. When the centre column is filled, the writing of the intermediate data will then start on the top of extreme right column memory block (see Figure 4.17). When the extreme right column, is filled the writing of the data will move left to the centre column again. The writing process will move to the left extreme column when it finishes with the centre column as illustrated in Figure 4.18.

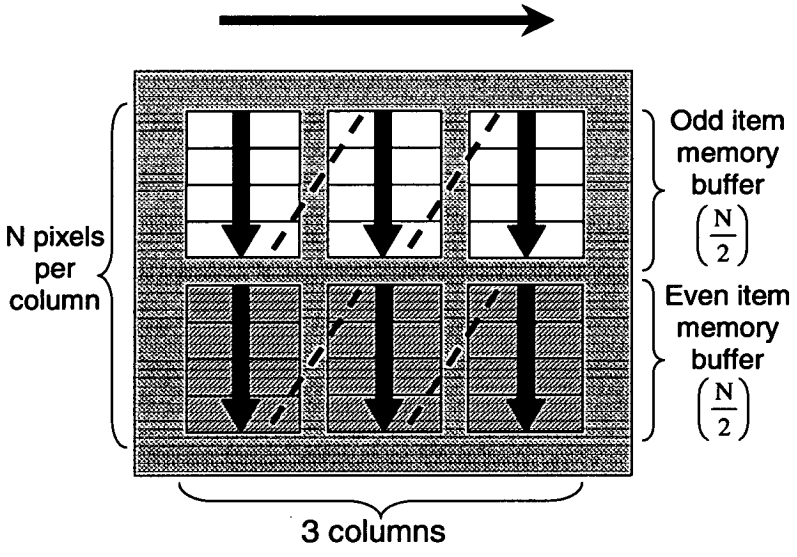


Figure 4.17: *INMEM write access from left to right*

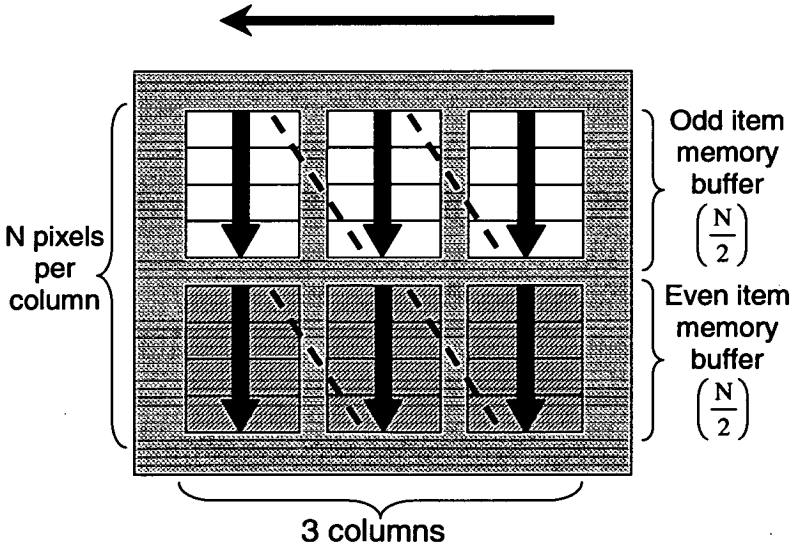


Figure 4.18: *INMEM write access from right to left*

The row processor will only commence its (column-wise) operation when the column processor starts processing the third column data. This is because it needs at least three input vector terms to calculate an odd term (see Equation 4.3). The row processor will stop processing at the end of the third column. It will be idle and wait while the column processor processes the next column. The row processor will resume processing when the column processor starts computation on the next subsequent even numbered column (of input sample). Henceforth, this process will repeat itself with the row processor only starting its column-wise processing when the column processor starts computation on even numbered columns. The row processor is designed to process data samples of two rows at one process cycle. Therefore, during processing the row processor outputs four sets of data over a period of three clock cycles (see Figure 4.13).

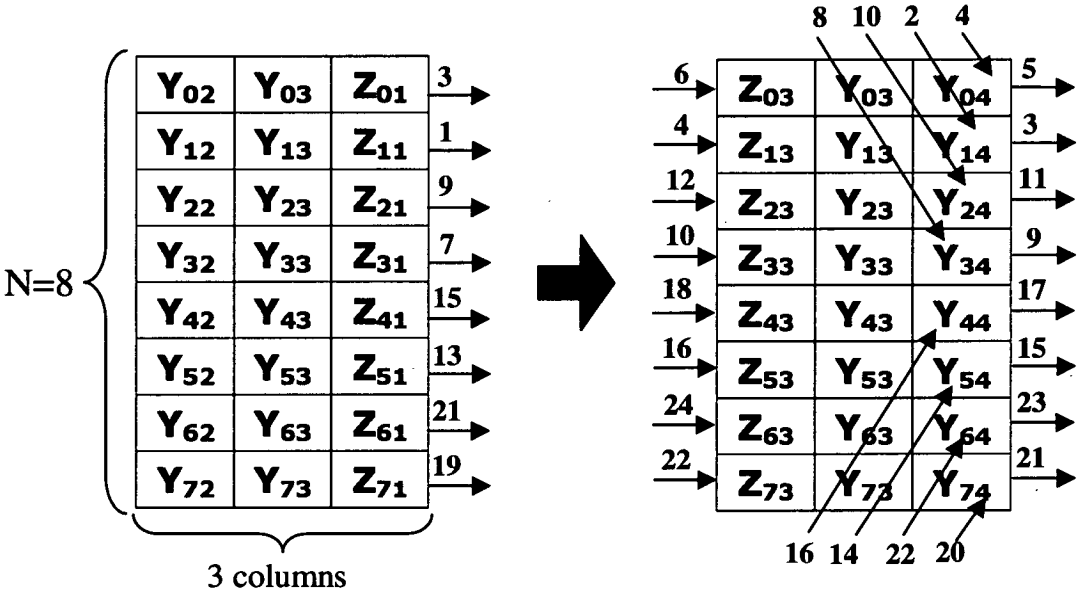


Figure 4.19: *Tandem write-read-write access sequence of INMEM - Column of data on the left shows the data value before the process cycle starts. Column of data on the left shows the data samples at the end of the process cycle. The Red terms indicate that the terms are from the Row processor and the Blue terms are from the Column processor. The Arrow-in indicates a write access and Arrow-out indicates a read access. The numbers next to the arrow are sequence indicators. The sequence number in Blue represents access by the Column processor and the Red by the Row processor.*

These outputs of the row processor are fed into the data output control, the RPMEM (LL sub-band data output) and the INMEM (HH and HL sub-band data output). The HH and HL sub-band data output are stored only in the extreme left and right column of the INMEM. The writing and reading of the data to and from the INMEM are done in tandem as illustrated in

Figure 4.13 and 4.19. The 3 columns of data shown on the left of Figure 4.19 show the data samples before the process cycle starts. The 3 columns on the right show the data samples at the end of the process cycle. The red terms indicate that the terms are from the row processor and the blue terms are from the column processor. The arrow in the figures show the type of access; arrow in indicates a write access and arrow out indicates a read access. The numbers next to the arrow are sequence indicators. The sequence number in Blue represents access by the Column processor and the Red by the Row processor. The data from the row processor previously stored in the intermediate memory will be used in subsequent column processing.

Coordination between The Processors and The Re-process Memory (RPMEM) Buffers

Writing the LL sub-band output data from the row processor to the RPMEM is straightforward, as the RPMEM is a FIFO memory. Writing the processed output data from the row processor happens every time the LL sub-band data is valid. Data from RPMEM are only read by the column processor if multi-level analysis is set. The memory location of the RPMEM for reading and writing are advanced simply by clocking it.

4.4 Architecture Comparison

The three architectures presented in the previous two sections are 3 distinctive architectures. Even though both Andra's and Lian's design are pipelined data-path centric based architectures, there are still differences between them. Therefore, a data-path arithmetic logic hardware count comparison of all 3 architectures was made. The hardware count was calculated on the basis of the minimum number of data-path arithmetic logic hardware unit (for each architecture) to complete a 2-D transformation in one iteration. Table 4.2 provides a comparison of the proposed architecture with the two existing architectures presented in this chapter. The term N in the table represents the maximum number of elements in a row/column of the input data. All the multipliers in all the architectures were replaced with shifters. The value in Table 4.2 for Lian's architecture is based on the assumption that Lian's architecture would use two of the same processors to complete a 2-D DWT operation in one iteration. The number of registers under consideration does not include the register in the register files.

	K. Andra et. al's architecture	C-J. Lian et. al's architecture	The proposed architecture	Worst case % saving
Adders	8	8	5	37.5%
Shifters	4	4	5	-25%
Registers	16*	16	10	37.5%
INMEM	4N	NXN	3N	25%
RPMEM	$N \times \frac{N}{2}$	$\frac{N}{2} \times \frac{N}{2}$	$\frac{N}{2} \times \frac{N}{2}$	0%

Table 4.2: *Data-path arithmetic hardware comparison table. The asterisk * indicates that the comparison exclude the register files.*

It can be seen from Table 4.2 that our proposed architecture has a significantly lower number of hardware components in comparison to the other two referenced architectures. Hardware savings in all of the listed components (except for shifter and reprocess memory) are achieved. The worst case percentage saving of components shown in the table are as follows; 37.5% for the adder, -25% for the shifter, 37.5% for the register, 25% for the intermediate memory buffer and 0% for the re-process memory buffer. As the increase in area (due to the extra shifter) is small, the impact on the saving in terms of area and power of the overall system will not be affected.

The ALU centric architecture has less data-path arithmetic logic hardware as most of the arithmetic logic hardware is reused (one arithmetic hardware for more than one arithmetic operation). The arithmetic logic hardware is reused by the feedback path which forms part of the accumulator structure. Although an ALU centric architecture is relatively more programmable than pipelined centric architecture (by changing the microcodes), it is slightly more complex to build. It is more complex to build because it needs more effort to schedule its operation and thus designing its control unit.

4.5 Summary

This chapter briefly introduces the lifting based DWT image compression standard JPEG2000 along with its 5/3 lifting based DWT algorithm. Following that, two of the existing JPEG2000 5/3 2-D DWT architectures were introduced and described. These two existing lifting based

architectures (Andra's and Lian's) were pipelined data-path centric architectures which are inflexibly hardwired and require more arithmetic logic hardware design. Therefore, a shift-accumulator ALU (SA-ALU) centric architecture was proposed. The proposed architecture's construction and workings were elaborated. The SA-ALU centric architecture which has been optimised at the algorithmic and architectural level requires significantly less hardware than the existing lifting based architectures. Hence, the close relationship between the algorithm and architecture can be exploited and optimised to yield a better hardware. It has been shown that this architecture has a significant reduction in area as compare to Andra's and Lian's architecture. Such reduction in area can in turn reduce power consumption. Besides the reduction in area, combining the ALU centric architecture with an efficient system schedule can also save the number of data accesses.

Chapter 5

Embedded Extension for JPEG2000's 5/3 Lifting Based Discrete Wavelet Transformation

5.1 Introduction

Despite having good features such as progressive image transmission by quality/resolution, Discrete Wavelet Transform (DWT) based applications are not very widespread. This is due to the fact that conventional convolution based implementation of a DWT is not very viable for implementation. These problems were overcome by using the lifting based scheme for the DWT [5, 6], which has been selected as the scheme used in the JPEG2000 [1] standard. However, even with the lifting based scheme, few have targeted their work on low power implementation of this algorithm, especially by the reduction of computation steps and switched capacitance through generic algorithmic techniques [10]. Such implementation approaches are desirable since they yield generic circuits with no added design effort.

This chapter presents a novel algorithmic technique for reducing the power dissipation and memory requirements for the data-extension process prior to one-dimensional (1-D) DWT. This embedded data-extension algorithm can be apply to any lifting based scheme DWT. As DWT is fundamentally implemented as a FIR filter, data extension is important to ensure that the FIR filter, at the edge/boundary of an image has enough data in order to produce the same number of data samples as the originating input data and as well as to reduce artefacts. This chapter will demonstrate that the embedded algorithm can lead to more than 50% reduction in the amount of memory required resulting in significant reduction in area and power. Power reduction is achieved by combining and embedding data-extension into the main DWT algorithm in both the start and the end of the transformation process. This reduction is achieved as a result of the following;

- Reduction in the amount of storage memory since explicit extensions are not required.
- Reduction in the number of read/write accesses since extension is only performed when the main data term is read or written.
- Reduction in the number of arithmetic operations.

This chapter is organised as follows: Section 5.2 presents the background as to why an image would need to undergo data extension before going into the compression process. Section 5.3 presents the proposed embedded data extension which reduces both the power consumption and hardware area. Then, in Section 5.4 the power consumption of conventional explicit data extension hardware is evaluated and presented. These power consumption values are compared with the analytical results of the proposed embedded extension algorithm. Section 5.5 gives the summary of this chapter.

5.2 Data Extension for Image Compression

Sub-band coding operations are basically filtering operations. From the principle of signal processing, a perfect linear filter should have infinite number of taps and the data which the filter operates on should also be infinite in number. However, in real life, neither the number of pixels in an image nor the number of taps in a digital filter is infinite. As such, practical filters will not only produce data output that have number of data/pixels not equal to the original data/image but also inevitably produces data output that contains distortions or artefacts, especially at the edge of the image. These artefacts are made worse at low-bit rate compression as aggressive quantisation that follows the transformation adds on to the distortion. If a system has no resource constraints, the wavelet transform can be performed on the whole image. In this way, the artefacts are made less obvious to the observer since they appear at the edge of the image. However, when there are limited memories available, the image has to be tiled and processed independently, the artefacts are especially disturbing along the boundaries of the tiles due to this discontinuity. These problems can be reduced or eliminated by using several different techniques. One of the effective ways of reducing edge artefacts is by symmetric extension [160]. Such extension also ensures that a DWT, at the edge/boundary of an image, has enough data in order to produce a single output.

5.2.1 Symmetric Extension

The work on symmetric extension was first presented by Smith et. al. [160]. Symmetric extension, not only keeps the number of analyzed or decomposed wavelet transform coefficients the same as the number of pixels in the original input image, it also results in symmetrically decomposed coefficients. There are two types of symmetric extension; they are half sample (HS) extension and whole sample (WS) extension [161]. Examples of a HS and WS extension are shown in Figure 5.2 and 5.3 respectively. The original input data that were used for the extension is shown in Figure 5.1.

A B C D E F G H

Figure 5.1: *Original input data samples*

G F E D C B A **A B C D E F G H** H G F E D C B

Figure 5.2: *Half sample (HS) extension*

G F E D C B **A B C D E F G H** G F E D C B

Figure 5.3: *Whole sample (WS) extension*

The HS extension repeats the first/last sample (see Figure 5.2) of the original input and extended data is symmetric about the half-way (half-sample) between the repeated first/last samples. Whereas, the first/last sample of the original input data is shared by the WS extension. Therefore, the WS extended data is symmetric about the first/last sample.

5.2.2 JPEG2000 5/3 Data Extension

The data extension scheme that was adopted by the JPEG2000 standard is the WS symmetric as shown in Figure 5.4. The i_o indicates the first indexed input data sample and the i_l the index for the last input data sample. The 1-D data extension had to be perform on the input data samples prior to 1-D DWT process.

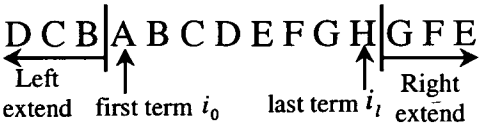


Figure 5.4: *JPEG2000's symmetric data extension - with i_0 as the index for the first data sample and i_l as the index for the last data sample*

The standard also specifies the number of samples to extend to the left/right depending on whether the first/last is an odd or even indexed sample term. The number of term to extend is as stipulated in Table 5.1.

	$i_0(i_{left})$ - Starting term	$i_l(i_{right})$ - Ending term
Odd	1	1
Even	2	2

Table 5.1: *Number of extension terms both even and odd starting and ending term specified by JPEG2000 standard*

As can be seen from Table 5.1, if the data starts or ends with even number indexed term, then two data terms are extended to the left or to the right of the data set. If the data starts or ends with an odd number indexed term, then a single data extension is added to both the left and the right of the data set. The extension of the input data and its relationship to the main algorithm is illustrated in the later Section 5.3. This data extension is required to be applied to both the horizontal and vertical process for all levels of analysis/decomposition and synthesis/recomposition.

5.2.3 Conventional Implementation of JPEG2000's Data Extension

There are several ways of implementing the JPEG2000's data extension. A common way is to first assume that a line of the image data samples is stored and available for processing.

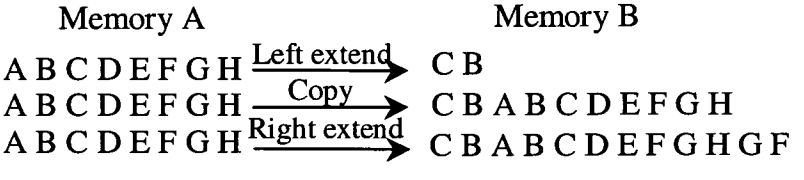


Figure 5.5: *Conventional data extension process*

Conventional straightforward data extension first detects whether the starting term is odd or even termed. Then, the process involves extending the data to the left, copying all the original data and lastly performing a right extension on the data as seen in Figure 5.5. This extension operation is both a memory and computationally intensive process.

5.3 Embedded Data Extension Algorithm

It has been mentioned in the previous section that conventional straightforward data extension is both area and operation intensive. In order for the DWT compression to be viable for wide usage especially in the low power mobile application area, the power consumption of the DWT process needs to be reduced. An embedded data extension algorithm is designed and developed to reduce both the redundant power and area required by the data extension process. Both area and power reduction are achieved by combining and embedding data extension into the main DWT algorithm in both the start and the end of the transformation process. This method of combining and embedding data extension into the main DWT algorithm to reduce power and area can be used by any lifting based DWTs. An embedded data extension algorithm can be applied both to the analysis/decomposition and synthesis/recompositon process. However, in this thesis the deliberations of the embedded data extension are made only with the analysis/decomposition equations of 5/3 DWT.

The embedded data extension algorithm embeds the data extension into the main DWT by simplifying the main lifting based on the consideration of the repetition of some terms in the extended data. Equations 4.3 and 4.4 in Chapter 4 describe the implementation of the 5/3 lifting based forward (analysis) DWT. Equation 4.3 is used for the computation of the odd coefficients, whereas Equation 4.4 is used for the computation of the even coefficients. The computations of the odd coefficients have to be performed first followed by the computations of the even coefficients. Note that both of the equations operate on the extended input data sample X_{ext}. The relationship between the output Y and its extended input data sample X_{ext} is depicted in Figure 5.6 and 5.7.

Repetition of some terms at the left side of the extended data samples can be observed from both Figure 5.7 and 5.6. From Figure 5.7, it can be seen that X_{i_0-1} is equal to X_{i_0+1} $(X(2))$ in the odd indexed starting term data samples. And from Figure 5.6, Y_{i_0-1} (or $Y(1')$) is equal to Y_{i_0+1} $(Y(1))$ in the even indexed starting term data samples, as $X(2)$ and $X(1)$ is duplicated as

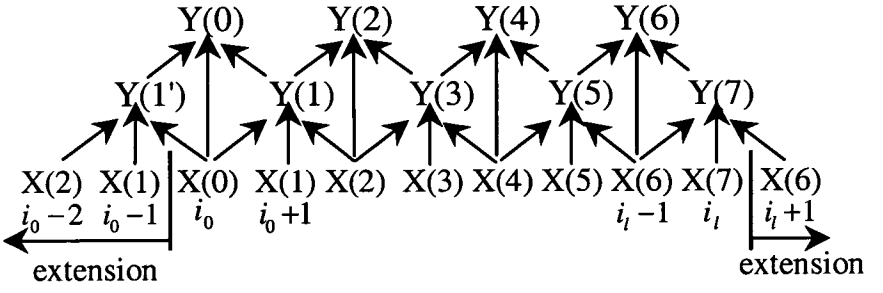


Figure 5.6: *Data dependence graph of JPEG2000's 5/3 DWT data extension with even numbered start term and odd numbered ending term*

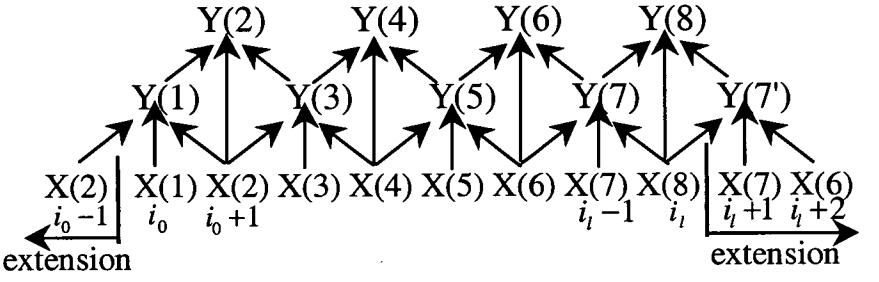


Figure 5.7: *Data dependence graph of JPEG2000's 5/3 DWT data extension with odd numbered start term and even numbered ending term*

X_{i_0-2} and X_{i_0-1} respectively. Similar repetition at the right side of the extended data sample can also be observed. In the odd indexed ending extend data sample, X_{i_l+1} is equal to X_{i_l-1} ($X(6)$) as seen in Figure 5.6. And in the even indexed ending extend data sample in Figure 5.7, Y_{i_l+1} ($Y(7')$) is equal to Y_{i_l-1} ($Y(7)$) as $X(6)$ and $X(7)$ is duplicated as X_{i_l+2} and X_{i_l+1} respectively.

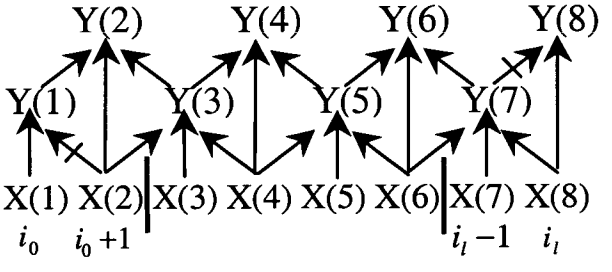


Figure 5.8: *Data dependence graph of JPEG2000's 5/3 DWT embedded data extension with odd numbered start term and even numbered ending term - crossed arrow means that the originating terms have a double contribution to the resulting coefficient terms*

As such, both the data flow-graphs, Figure 5.7 and 5.6, can be collapsed and be redrawn as shown in Figure 5.8 and 5.9 to reflect these relationships. The arrow with a line across it, in

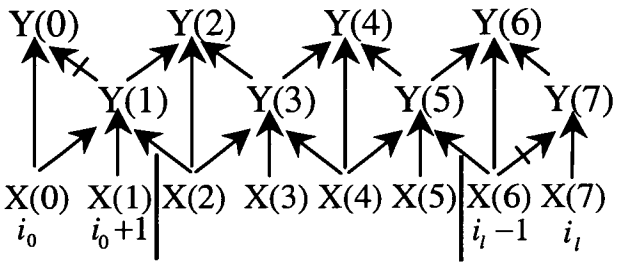


Figure 5.9: *Data dependence graph of JPEG2000's 5/3 DWT embedded data extension with even numbered start term and odd numbered ending term - crossed arrow means that the originating terms have a double contribution to the resulting coefficient terms*

both of the Figure 5.8 and 5.9, represents doubling the proportion of the originating terms in the contribution to the resulting coefficient terms. From Figure 5.8 and 5.9, it can be seen that the need for explicit extension has been eliminated. However, the original homogenous lifting based 5/3 DWT process is broken up. The embedded extension algorithm splits the homogenous conventional computation process into three parts. The first part is termed as the Start Process, the second part the Normal Process (which is the same as the conventional process), and the third part the End Process. The data-extensions are only embedded in the Start Process and the End Process. The Start Process embeds the left extension and the End Process the right extension. Using the above-mentioned relationships, four new equations (Equation 5.1 to 5.4) were derived from Equation 4.3 and 4.4. Two of the equations (Equation 5.1 and 5.2) are for Start Process and the other two equations (Equation 5.3 and 5.4) are for the End Process. These four new equations will be described and elaborated in the following two processes.

Start Process

There are two equations for computing the coefficients in the Start Process. The first, Equation 5.1, is the embedded extension equation for computing the first odd coefficient when the input data starts with an odd number indexed term. The second, Equation 5.2, is used for the computation of the first even coefficient when the input data starts with an even number indexed term. Note that i_0 in Y_{i_0} indexes the first term of the data.

$$Y_{i_0} = X_{i_0} - X_{i_0+1} \tag{5.1}$$

$$Y_{i_0} = X_{i_0} + \left\lfloor \frac{Y_{i_0} + 1}{2} \right\rfloor \tag{5.2}$$

As X_{i_0-1} is equal to X_{i_0+1}, Equation 4.3 is reduced to Equation 5.1. Also, using the fact that Y_{i_0-1} is equal to Y_{i_0+1}, Equation 4.4 reduces to Equation 5.2.

End Process

Similarly in the End Process, Equation 4.3 and 4.4 were reduced to become Equation 5.3 and 5.4 respectively.

$$Y_{i_l} = X_{i_l} - X_{i_l+1} \tag{5.3}$$

$$Y_{i_l} = X_{i_l} + \left\lfloor \frac{Y_{i_l} + 1}{2} \right\rfloor \tag{5.4}$$

Equations 5.3 and 5.4 are the embedded extension equations for computing the last odd coefficient, when the input data ends with an odd number indexed term, and the last even coefficient when the input data ends with even number indexed term respectively. The i_l in Y_{i_l} indexes the last term of the data.

5.4 Power Analysis of Embedded Extension Algorithm

It is evident from the new set of equations 5.1, 5.2, 5.3 and 5.4, computation steps are reduced as the whole explicit data extension process is done away with. Besides that, the entire computation of the odd coefficient is omitted when there is either an even number indexed start or end term. In the same way, a read access and two arithmetic operations are omitted in computing the first/last odd coefficient when there is either an odd number indexed start or end term. Thus, achieving even more saving in computation steps. The effect of these savings are analysed in the following sub-sections.

5.4.1 Power Consumption of Explicit Data Extension's Hardware

In order to establish the extent of the area and power saving of this embedded extension scheme, the area overhead and power consumed by the explicit data extension must first be evaluated. Hence, the power consumption of the explicit data extension process using dedicated hardware is evaluated and analysed. The dedicated hardware that is used for data extension is as shown in Figure 5.10.

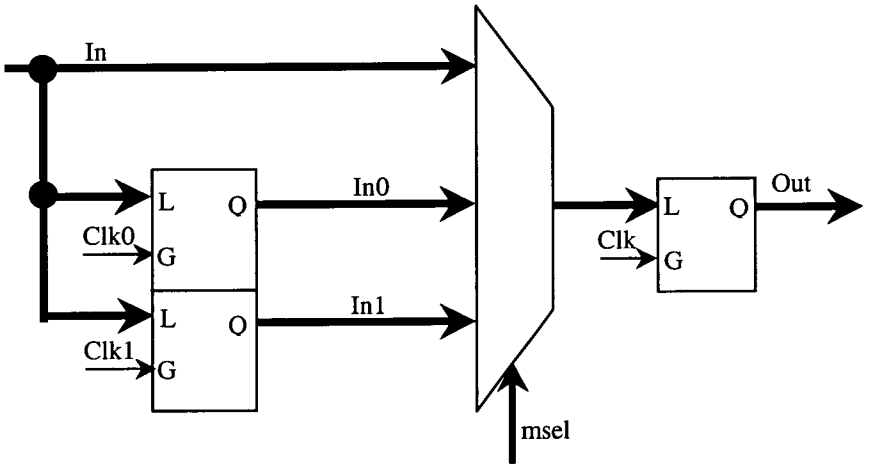


Figure 5.10: *Explicit data extension hardware*

The data extension hardware was coded and evaluated with the flow and softwares described in Section 2.4.1 of Chapter 2. The power evaluations results of the explicit data extension process on the dedicated hardware are shown in Table 5.2. The table shows the power consumption values of the data-path portion of the extension hardware in extending a frame (in the horizontal direction) of (greyscale) test images Lenna, Baboon and Pepper [1] at image size of 64 × 64, 128 × 128 and 256 × 256.

Conventional Explicit Data Extension Hardware'S Data-Path Hardware Power Consumption

It can be seen from Table 5.2 that the power consumption of explicitly extending the data for any size and type of image is significant when compared to the power consumption of the actual 5/3 transformation process (see Chapter 6) itself. It can also be observed from Table 5.2 and Figure 5.11 that the actual and average power consumption is getting lower as the size of the image increases.

[1] Test images are shown in Appendix D

Image Size	Type of Image	Power Consumption (μW)	Average Power Consumption (μW)
64 X 64	Lenna	306.46	313.00
	Baboon	320.32	
	Pepper	312.22	
128 X 128	Lenna	293.24	300.55
	Baboon	310.88	
	Pepper	297.52	
256 X 256	Lenna	278.64	289.67
	Baboon	303.82	
	Pepper	286.54	

Table 5.2: *Power consumption table of the explicit data extension's data-path hardware. These values are from evaluating the data extension process (in the horizontal direction) of a single frame of test image Lenna, Baboon and Pepper at image size of* 64×64, 128×128 *and* 256×256

The power (as opposed to energy) consumption is becoming lower as the number of pixel/operation increases as the amount of energy needed for the extension is getting smaller in proportion to the image size. It is in effect averaging and distributing this extension energy among each individual pixel operation. Note also that power is energy per unit time, which is an average of energy over time. This also does not mean that when the power is low, the total energy used in the operation is low. In this case, the total energy consumed increases exponentially as image size per row/column increases dyadically, but the power consumption is about constant or is slightly decreased.

Conventional Explicit Data Extension Hardware'S Control Logic Power Consumption

Most power evaluation only focuses on the data-path power consumption. However, the power consumption of the control logic of any processing is too great to be disregarded when taking into account the power consumption of a process. As such the explicit data extension control logic's power consumptions were evaluated. Table 5.3 shows the power consumption of the explicit data extension hardware's control logic.

From Table 5.3 and Table 5.2 it can be seen that the power consumption of the control logic is greater than the power consumption of the data-path hardware. It can also be observed from the table that the power of the control logic decreased slightly as the image size increased from 64×64 to 128×128. This is again due to the same reason that as the image size increases the extra amount of power used for the extension is getting smaller in proportion to the image size.

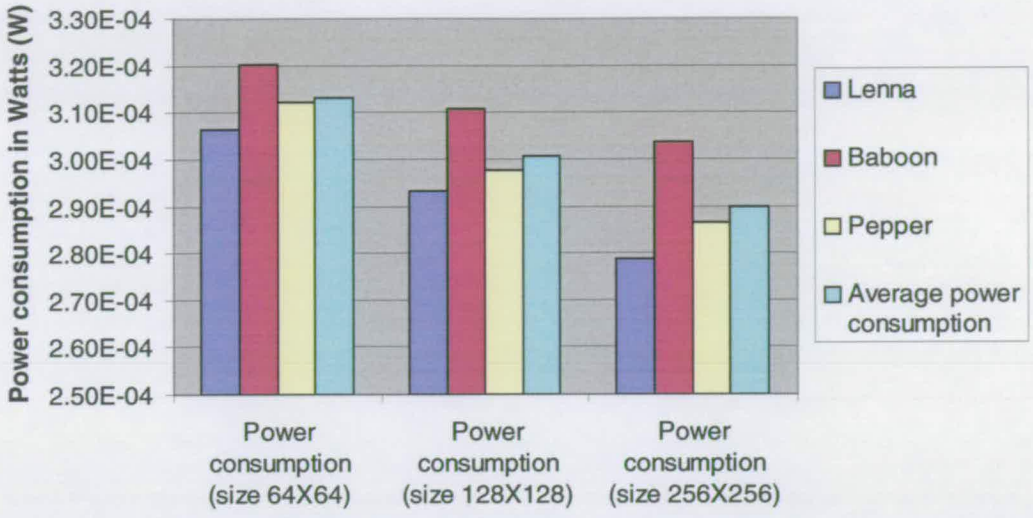


Figure 5.11: *Graph of explicit data extension power consumption of test image Lenna, Baboon, Pepper at image size of* 64×64, 128×128 *and* 256×256

Image size	Power consumption (μW)	Average Power Consumption (μW)
64 X 64 (8 bits control counter)	462.78	468.84
128 X 128 (8 bits control counter)	461.11	
256 X 256 (9 bits control counter)	482.64	

Table 5.3: *Power consumption table of the control logic of explicit data extension hardware at image size of* 64×64, 128×128 *and* 256×256

However, the power consumption of the control logic increases as the image size increased to 256×256. This is due to fact that a 9 bit counter is needed to index a image with column or row of 256 pixels. The power increase is mainly due to the increased control counter's width.

Area Requirement of Conventional Explicit Data Extension Hardware

Besides the power penalty that explicit data extension hardware has to incur, the hardware also incurs an area overhead. The area requirements of explicit data extension hardware are stipulated in Table 5.4.

	Data-path area in square microns (μm^2)	8 bits control area in square microns (μm^2)	9 bits control area in square microns (μm^2)	Total area in square microns (μm^2) for both 8 bits control (and 9 bits control)
Combinational area	1174.96	5313.60	5679.55	6488.56 (6854.51)
Non-combinational area	2536.80	3764.65	4016.71	6301.45 (6553.51)
Total cell area	3711.76	9078.25	9696.26	12790.01 (13408.02)

Table 5.4: *Area requirement table of explicit data extension hardware at image size of* 64×64, 128×128 *and* 256×256

The values from Table 5.4 indicates that a massive area is used by data extension hardware. It can also be noted that the area occupied by the control logic is 3 times more than the data-path hardware. The 3 times larger area occupied by control logic is the reason why the power consumption of the control logic is higher than the power consumption of data-path hardware.

5.4.2 Analytical Results

With the four new equations (equations 5.1, 5.2, 5.3 and 5.4) and the new data flow-graphs (Figure 5.8 and 5.9), hardware and computation saving can be calculated analytically. Three techniques were evaluated analytically in a 1-D access with a 5/3 DWT filter. These are; the conventional straight-forward extension, extension by scheduled access and the new embedded extension algorithm. The extension by scheduled access is a scheme where the data (read) access controller is scheduled to read the original data in a manner that the data are extended systematically, so that the DWT processor can calculate all the coefficients as if it is in a homogenous process. Results of the analytical evaluation of the memory, computation usage and saving are stipulated in Table 5.5. The output of the embedded data extension algorithm is verified and compared to that of the conventional straight-forward extension using MATLAB. The reductions in memory requirement are shown in the fourth column and the reductions in operation are shown in the last (extreme right) column of Table 5.5. Note that 'N' represents the number of data items in a row or column and 'Number of Operations' refers to the number of read/write access operations used in the respective data extension operation. Three cases of extension are considered in the calculation of computational and memory savings. In the first

case (case A), we consider the extension with input data that starts with an odd number indexed term and ends with an odd number indexed term. The second case (case B) considers input data that either starts with an even number indexed term and ends with an odd number indexed term or starts with odd number indexed term and ends with even number indexed term. The third case (case C) considers extensions with input data that starts with an even number indexed terms and ends with even number indexed term.

Technique	Case	Memory Used	Memory Reduction	Number of Operations	Operations Reduction
Conventional Straightforward Extension	A	N + 2	0	2N + 4	0
	B	N + 3	0	2N + 6	0
	C	N + 4	0	2N + 8	0
Extension by Scheduled Accessing	A	-	N + 2	2	2N + 2
	B	-	N + 3	3	2N + 3
	C	-	N + 4	4	2N + 4
Embedded Extension	A	-	N + 2	0	2N + 6
	B	-	N + 3	0	2N + 11
	C	-	N + 4	0	2N + 16

Table 5.5: *Analytical result of memory and operation usage and saving at per line extension process basis*

From the analysis of the above equations and figures, it is evident that two operation overheads of a read access and a write access are incurred for every data term processed in the conventional straight-forward extension. As for the extension by scheduled access, each data in the extension only incurs one operation. The embedded extension algorithm not only does not incur any extension overhead, but also saves up to an additional of eight arithmetic operations (for data that starts and ends with an even number indexed term) per row/column. The savings are due to the fact that several computation steps are eliminated. Although the savings with the embedded extension algorithm in 1-D process are only a few operations more than those of extension by scheduled access, savings will be considerable when it comes to the 2-D processing of a whole image frame.

5.4.3 Discussion

From Table 5.5 and Section 5.4.1, it has been shown that significant savings in memory and number of operations has been achieved. Firstly, the whole of the explicit data extension is totally done away with. Secondly, with the embedded extension algorithm less operations are needed during the implicit extension compared to normal processes. Such effects lead to significant savings in both area and power. However, the implementation of the embedded extension algorithm has a small area overhead. This small increase in area is mainly due to the fact of incorporating Start/End extension processes into the controller. Although there is an increase in area, the overall reduction in power consumption will not be affected. This is because the increase is insignificant compare to the savings achieved, due to the reduction in memory size and the number of arithmetic operations. Furthermore, these additional states used for data extension can be effectively manipulated by appropriate power-down mechanisms. Additionally, the embedded extension control logic is relatively smaller than the existing pipeline centric architecture.

5.5 Summary

This chapter introduced the topic of distortion or artefacts at the edge of multi-resolution sub-band coded image which is worsened by aggressive quantisation that normally proceeds transformation in low-bit rate compression. Thus, the concept of using data extension was looked into to reduce and eliminate artefacts as well as to keep the output data samples equal to the originating input data samples. Two types of symmetric extensions, the HS and WS extension, were presented along with the data extension requirement of JPEG2000. Extension of the data unavoidably increased the consumption of power and memory resources. As such a low power algorithm for the data extension of the DWT on CMOS based DSPs was proposed. This embedded extension algorithm reduces power by combining the data-extension into the main lifting based DWT such that the computation steps are reduced and hence resulting in a reduction in the switched capacitance sector of the dynamic power (see Equation 2.13). Implicitly extending the data in the DWT processing save power and memory requirements. It has been shown that the embedded extension algorithm can be used to obtain more than 90% memory and power reduction as explicit data extension process and hardware are done away all together. This method of combining and embedding data extension into the main DWT algorithm to reduce power and area is suitable to be use in any lifting based DWT.

Although there is an increase in area due to the extra control logic, the overall reduction in power consumption is not affected.

Chapter 6

Power Analysis of Shifter-Accumulator Arithmetic Logic Unit Centric Processor

6.1 Introduction

Dedicated hardwired processors have been used extensively in DSP. Many of these dedicated hardwired processors are direct implemented pipelined data-path centric processors. These pipelined data-path centric processors are popular because these processors are high speed and easy to implement. Besides the pipelined data-path centric processor, the accumulator arithmetic logic unit (ALU) centric processor is another type of processor which is widely used. The accumulator ALU centric processor differs from the pipelined data-path centric processor in the fact that the accumulator ALU centric processor centres its design on an accumulator feedback architecture. Whereas, the pipelined data-path centric processor is built around a non-accumulating, feed-through pipeline architecture. Traditionally, the pipelined data-path centric processor is perceived as more power consuming than the ALU centric processor. It is perceived as more power consuming because of the substantial amount of pipeline registers it possesses. However, to date there are no power evaluation figures on both the pipelined data-path centric processor and the ALU centric processor to show whether a pipelined data-path centric processor consumes more power than an ALU centric processor.

Therefore, this chapter presents the power evaluation and comparison of the modified version of Andra's [2] and Lian's [7] feed-through pipelined data-path centric processor architecture and the proposed SA-ALU centric processor architecture. This chapter will also show that the proposed SA-ALU centric processor architecture consumes 25.1% lesser power than the feed-through pipelined data-path centric processor architecture at the same clock period and power evaluated over the same process time duration when processing 128×128 test images.

In addition, cases where the two feed-through pipelined data-path centric processors can be made to consume lesser power than the proposed SA-ALU centric processor architecture are also shown.

This chapter is organised as follows: In Section 6.2, the details of two existing feed-through pipelined data-path centric processors architectures are presented. Following that, the power analysis of the architectures evaluated at different time durations are presented together with the analysis of the results in Section 6.3. Next, insights from the analysis are discussed in Section 6.4. Finally, the summary of the findings and contribution made by this study are given in Section 6.5.

6.2 Processor Architecture

In Chapter 4, the overall architecture of both the existing pipelined data-path centric and the proposed SA-ALU lifting-based DWT architecture were described. However, the processor architectures presented in Chapter 4 are the general make up of the processor. In order to make a fair comparison of their power consumption, these processors have to be slightly modified.

In order to ensure that process image output results from these lifting-based DWT architectures are reversible or invertible for up to 5 levels of analysis, the bit-width of all processors data-path units is 16 bits. The bit-width of 16 was derived from Matlab simulations. All three processors were built using the same type of data-path components. The description of the modified architecture of the two pipelined data-path centric processor are described in greater detail in the following. The structure of both the vertical (column) and horizontal (row) processor of the proposed SA-ALU processor were as described in Section 4.3.2 of Chapter 4.

6.2.1 Pipelined Data-path Centric Processor Architecture

As presented in Chapter 4, there are two existing pipelined data-path centric architectures. The two architectures are Andra's architecture [2] and Lian's architecture [7]. Although, both of these designs are based on the feed-though pipelined data-path centric architecture, they are very different at both the processor and overall architecture level. To ensure a fair power consumption comparison, these two distinct processor designs were specifically modified to perform only the lifting based 5/3 DWT operation.

6.2.1.1 Andra's Pipelined Data-path Centric Processor Architecture

Figure 6.1 shows the modified Andra's lifting based 5/3 DWT processor architecture. Both the vertical (column) and horizontal (row) processors have the same processor architecture. This architecture is very different from the one presented in Chapter 4 as the one in the previous chapter is only capable of processing one set of (odd) coefficient terms. The complete architecture in Figure 6.1 shows both the odd term and even term processor.

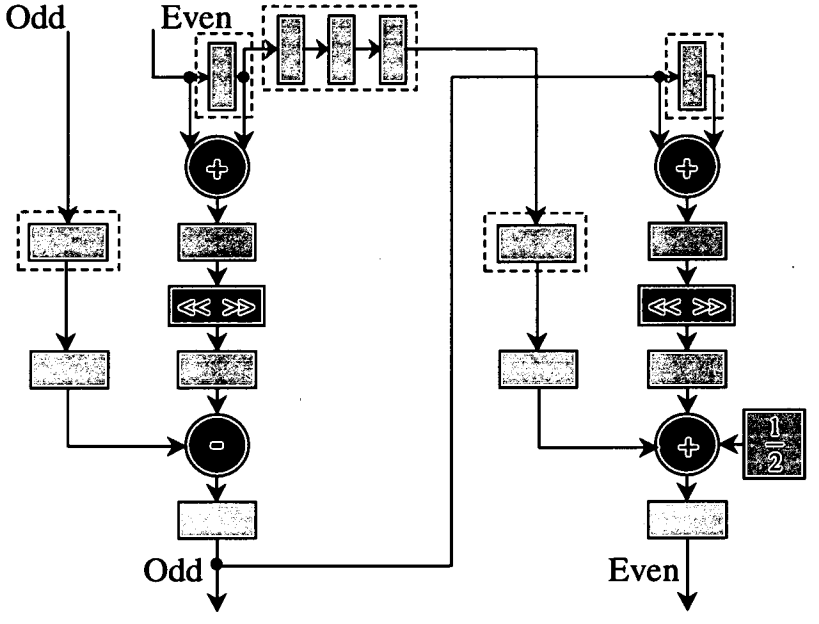


Figure 6.1: *Simplified and modified 1-D Andra's lifting based 5/3 DWT processor architecture for both odd and even term. The registers with dotted rectangle box indicate the part of the architecture not considered in the data-path power evaluation.*

Furthermore, the multipliers of the modified processors architecture were replaced with shifters. The shifter used in the computation of the odd terms can only perform one right shift (for coefficient of $\frac{1}{2}$) whereas the shifter for the even terms can shift up to two bits position to the right (for coefficient $\frac{1}{4}$). Note also that the last adder in the processor is a three input adder with one of the input set as a constant value of $\frac{1}{2}$. This factor of $\frac{1}{2}$ is for computing the even terms of the JPEG2000's 5/3 DWT where a constant of $\frac{1}{2}$ is added to it.

As Andra's processor is built around the finely pipelined data-path hardware, the large number of data-path registers are necessary to keep the samples in synchronisation with each pipeline stage. The registers with dotted rectangle box are the registers used for synchronising the data samples to ensure that the processor processes the correct data samples and outputs the correct

processed data samples. The registers with dotted rectangle box in between the odd and even processor is termed the register file in [2]. As the design of the register file is not published and therefore is unknown, to ensure a fair comparison, the power consumption of the registers used for synchronising the data sample are not included in the data-path power evaluation.

This modified processor has a data latency of 5 clock cycles for the odd terms and 9 clock cycles for the even terms. This processor outputs one valid data result every clock cycle 5 clock cycles after the first data is fed into the inputs. And it outputs two valid data results every clock cycle 9 clock cycles after the first data is fed into the inputs.

6.2.1.2 Lian's Pipelined Data-path Centric Processor Architecture

Lian's processor architecture was modified in the same way, so that a fair comparison of its power consumption can be made. The folded architecture is simplified and modified to specifically perform only the 5/3 transform. The modified Lian's processor architecture is as shown on Figure 6.2. Both vertical (column) and horizontal (row) processor utilise the same processor architecture.

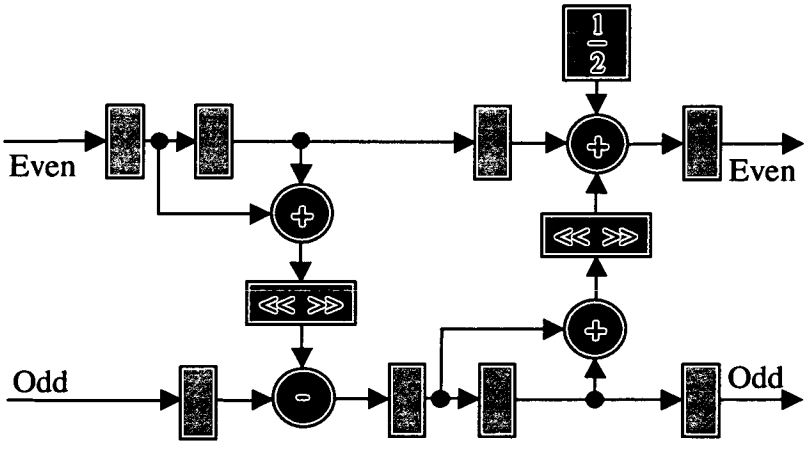


Figure 6.2: *Simplified and modified 1-D Lian's lifting based 5/3 DWT processor architecture for both odd and even term*

Lian's processor was modified to a lesser degree as compared to Andra's processor as Lian's processor is (by itself) sufficient to compute both the odd and even term of the 5/3 DWT. Lian's processor consists of two pipeline threads running parallel with each other. Each of the pipeline thread is responsible for computing either the odd or the even terms as can be seen from Figure 6.2. The Lian's processor was modified in only two places; the multiplier units

have been replaced by shifter units and the last adder on the even term computation path have been replaced with a 3-input adder. The shifter which is used for the computation of the odd terms can only performance one right shift (for coefficient of $\frac{1}{2}$) whereas the shifter for the even terms can shift up to two bits position to the right (for coefficient $\frac{1}{4}$). The last adder on the even term computation path that had been replaced with a three input adder has one of the input set as a constant value of $\frac{1}{2}$. This factor of $\frac{1}{2}$ is for computing the JPEG2000's 5/3 DWT even terms. The modified Lian's processor has a data latency of 5 clock cycles for the odd and even terms. This processor outputs 2 valid processed results every clock cycle after 5 clock cycles.

6.2.2 Area Consumption of Processors

The area consumption of hardware unit is important as it affects not only the cost of the hardware but it also influences the amount of power consumed by the hardware. Power consumption of hardware is somewhat proportional to the areas where there are frequent switching activities. By knowing the amount of switching area, it is possible to approximate the relative power consumptions of similar types of hardware. The rule of thumb is that the larger the switching area the higher the power consumption. However, in real life, it is very difficult to define and isolate switching area. Therefore, the total area of a hardware is used as a rough gauge for the relative power consumption. The following is the comparison of the area consumption of the data-path units of Andra's, Lian's and the proposed SA-ALU column processors.

	Andra's processor area in square microns (μm^2)* and in %	Lian's processor area in square microns (μm^2) and in %	Proposed SA-ALU (column) processor area in square microns (μm^2) and in %	Area saving in square microns (μm^2) and in % w.r.t. Andra's processor	Area saving in square microns (μm^2) and in % w.r.t. Lian's processor
Combinational area	7033.45 35.09%	7049.71 40.38%	7890.65 58.06%	857.20 12.19%	840.94 11.93%
Non-combinational area	13009.58 64.91%	10407.67 59.62%	5699.78 41.94%	-7309.80 -56.19%	-4707.89 -45.23%
Total cell area	20043.02 100%	17457.38 100%	13590.43 100%	-6452.59 -32.19%	-3866.95 -22.15%

Table 6.1: *Area consumption table of Andra's, Lian's and proposed SA-ALU column processor*

The area consumed by the three processors are shown in Table 6.1. From the table, it can be seen that the total area of the proposed SA-ALU processor is smaller than both Andra's and Lian's processor. Note that there is a slight increase in the combinational logic area. This increase is mainly due to the number of multiplexers in the proposed column processor.

6.3 Power Analysis

Traditionally, the pipelined data-path centric processor is usually perceived as more power consuming than the ALU centric processor. However, to date there is no power evaluation figures on both pipelined data-path centric processor and ALU centric processor to show whether pipelined data-path centric processor consumes more power than ALU centric processor.

In order to establish whether the proposed SA-ALU centric processor consumes lower power than the pipelined data-path centric processor, the power consumption of all these three different processor architectures are evaluated and analysed.

6.3.1 Power Evaluation

Power evaluations were done on the data-path units of Andra's, Lian's and the proposed SA-ALU processor. The power evaluations were done only on the column processors of these three processors. All the three processors was coded and evaluated with the flow and softwares described in Section 2.4.1 of Chapter 2.

In order to compare the power consumption of different hardware fairly, one has to evaluate each of the hardware module when they are performing the same task with the same sets (and size) of data. The performance of each hardware module and the time that each hardware module finishes its task had to be also taken into account. The time which the hardware module finishes its task will greatly influence the power consumption figure the hardware module in question as power is a measurement with respect of the inverse of time. Therefore, in this thesis, all the column processors of the three architecture were put through a series of power evaluations with different images with different image sizes and with different power evaluation time. Each of the processors was only fed with the same set of test image vectors with the same image size for one set of power evaluation time in one power evaluation item. The power

evaluation figures are then compare with one another in each power evaluation item.

There is a total of three sets of (greyscale) test images used for testing in this thesis. They are test image Lenna, Baboon and Pepper[1]. The sizes of the three images are 64×64 and 128×128. The proposed SA-ALU column processor was fed with raw test images whereas the rest of the processors were fed with test images which had been extended in the vertical direction. Andra's and Lian's processor were fed extended data images as it must only process extended data to get the correct output. Therefore, the actual number of data items fed into Andra's and Lian's processor is $(N + 3)$ per column.

The three processors first went through two main sets of power evaluation experiments. The first set of power evaluation runs all three processors for a time duration equal to the proposed SA-ALU processor's process completion time. The second set runs for a duration of time equal to each processor's process completion time. A negative value in *Power Consumption Difference* column of the following tables indicates a reduction in power consumption while a positive value indicates an increase in power consumption.

6.3.1.1 Evaluation Duration at Proposed Processor'S Process Completion Time

The following tables and graphs show the power consumption value evaluated at the proposed SA-ALU processor's process completion time for both 64 64 and 128 128 test images. The proposed SA-ALU processor's process completion time for image size of 64 64 and 128 128 are 82640ns at a clock period of 10ns and 329000ns at a clock period of 10ns respectively. As the process completion time of the feed-through pipelined data-path centric processors is shorter than the process completion time of the proposed SA-ALU processor, the input data samples fed into the feed-through pipelined data-path centric processors after its process completion time is set to '0000H'.

Image Size of 64×64

Table 6.2 shows the power consumption and comparison of Andra's, Lian's and the proposed SA-ALU processor for a single frame of test image Lenna, Baboon and Pepper at image size of 64×64 for the duration of 82640ns at clock period of 10ns. As JPEG2000 core is also used as the core of intra-frame video standard Motion-JPEG2000 (MJ2 [155]), a single image is also considered as a single frame of a video. Therefore, throughout the whole of this thesis

[1]Refer to Appendix D

Image size	Type of image	Type of architecture	Power consumption (mW)	Power consumption difference - w.r.t Andra's (mW)	Power consumption difference - w.r.t Andra's (%)	Power consumption difference - w.r.t Lian's (mW)	Power consumption difference - w.r.t Lian's (%)
64 × 64	Lenna	Andra's	2.40	-	-	0.35	17.28
		Lian's	2.05	-0.35	-14.73	-	-
		SA-ALU	1.77	-0.63	-26.31	-0.28	-13.58
	Baboon	Andra's	2.43	-	-	0.35	16.87
		Lian's	2.08	-0.35	-14.44	-	-
		SA-ALU	1.82	-0.61	-24.94	-0.25	-12.27
	Pepper	Andra's	2.45	-	-	0.36	17.10
		Lian's	2.10	-0.36	-14.60	-	-
		SA-ALU	1.79	-0.66	-26.92	-0.30	-14.43

Table 6.2: *Power consumption and comparison of Andra's, Lian's and the proposed SA-ALU processor for processing a single frame of test image Lenna, Baboon and Pepper at image size of* 64×64. *Test conducted for 82640ns at clock period of 10ns.*

a single image is also refer as a frame of video. The break-down of the power consumption of the processors into *Net Switching* and *Cell Internal Switching* are graphically presented in Figure 6.3.

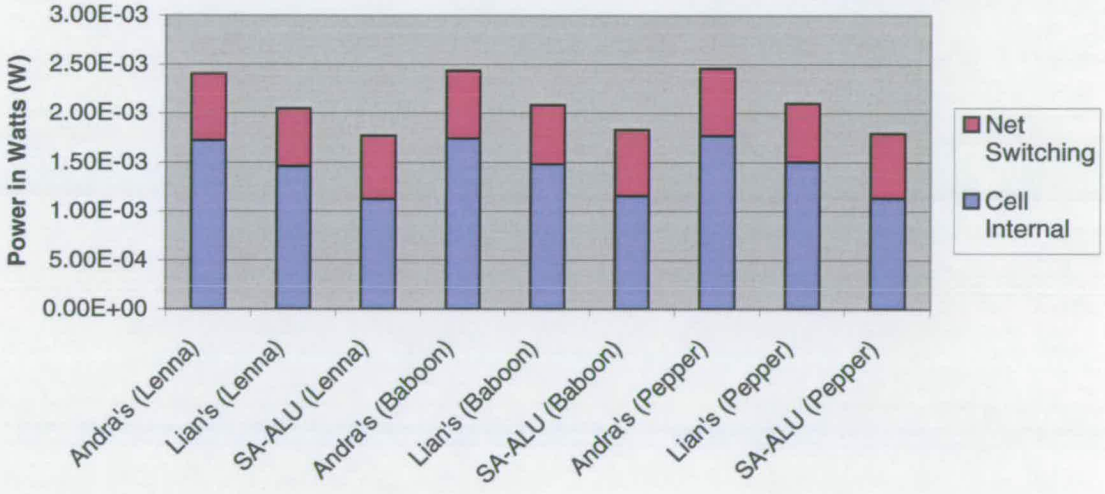


Figure 6.3: *Power consumption graph of Andra's, Lian's and the proposed SA-ALU processor for processing a single frame of test image Lenna, Baboon and Pepper at image size of* 64×64 *for duration of 82640ns at clock period of 10ns*

Despite having three very different test images, there is only a slight difference in the power

consumed by the processors for each of the test images. From Table 6.2 and Figure 6.3, it can be seen that the proposed SA-ALU consumes on average 26.06% and 13.42% lesser power for all three test images compare to Andra's and Lian's processor respectively.

Image Size of 128 × 128

The values in Table 6.3 show the power consumption and comparison of Andra's, Lian's and the proposed SA-ALU processor for a single frame of test image Lenna, Baboon and Pepper at image size of 128 × 128 for the duration of 329000ns at clock period of 10ns. The graphical break-down of the power consumption of the processors into *Net Switching* and *Cell Internal Switching* is shown in Figure 6.4.

Image size	Type of image	Type of architecture	Power consumption (mW)	Power consumption difference - w.r.t Andra's (mW)	Power consumption difference - w.r.t Andra's (%)	Power consumption difference - w.r.t Lian's (mW)	Power consumption difference - w.r.t Lian's (%)
128 × 128	Lenna	Andra's	2.38	-	-	0.36	17.52
		Lian's	2.03	-0.36	-14.90	-	-
		SA-ALU	1.76	-0.62	-25.92	-0.26	-12.94
	Baboon	Andra's	2.40	-	-	0.35	17.15
		Lian's	2.05	-0.35	-14.64	-	-
		SA-ALU	1.83	-0.57	-23.74	-0.22	-10.66
	Pepper	Andra's	2.40	-	-	0.35	17.11
		Lian's	2.05	-0.35	-14.61	-	-
		SA-ALU	1.78	-0.61	-25.58	-0.26	-12.85

Table 6.3: *Power consumption and comparison of Andra's, Lian's and the proposed SA-ALU processor for processing a single frame of test image Lenna, Baboon and Pepper at image size of* 128 × 128. *Test conducted for 329000ns at clock period of 10ns.*

The power consumption values of the three processors shown in Table 6.3 at image size 128 × 128 are very close to the value obtained from power evaluation with test images of size 64 × 64. The actual power consumption values for all three processors had in fact decreased for all test images. The average power reduction is 25.08% and 12.15% compared to Andra's and Lian's processor respectively. This is because as the image size increases, the power overhead for setting up and initialising the process has been averaged out. Note also that the power is energy per unit time, which is an average of the total amount of energy over time. This does not mean that when the power is low, the total energy used in the operation is low. In this case, the total energy consumed actually increased by about 3 times and the power consumption is

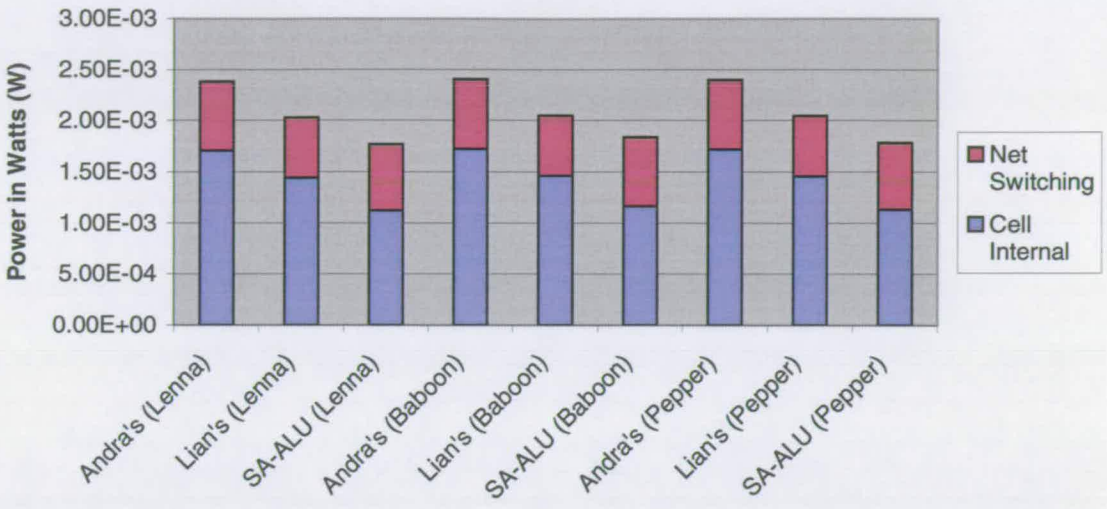


Figure 6.4: *Power consumption graph of Andra's, Lian's and the proposed SA-ALU processor for processing a single frame of test image Lenna, Baboon and Pepper at image size of* 128×128 *for duration of 329000ns at clock period of 10ns.*

about constant or is slightly decreased, while the number of pixels per row/column increased by two times. Note also that power consumed by three processors had increased slightly when processing test image Baboon. This shows that the power needed to process the test image Baboon is slightly greater than the power consumed by the process setup. This increased in power is mainly due to the fact that there are a significantly higher number of high frequency components in the test image Baboon. As the number of pixel increase, the number of pixel with big adjacent humming distance (due to the high frequency component) also increases.

6.3.1.2 Evaluation Duration at Individual Processor's Process Completion Time

Although, all three processors process about the same number of data samples, the time duration for each processor to complete one level DWT process is different. The proposed SA-ALU processor has a process completion time which is longer than those of both Andra's and Lian's processor. Both feed-through processors share the same process completion time for all image sizes. The process completion time for the three processors are given in Table 6.4.

The two feed-through pipelined data-path centric processors are about 3 times faster than the proposed SA-ALU processor when clocked at the same frequency. As power is a function of time, the different in process completion time has an impact on the power consumption of the processor. Consequently, power evaluations at each individual processors process completion

time were carried out.

Image Size	Process Completion Time		
	Andra's	Lian's	Proposed SA-ALU
64 × 64	21920ns	21920ns	82640ns
128 × 128	84650ns	84650ns	329000ns

Table 6.4: *Process completion time Andra's, Lian's and the proposed SA-ALU processor for a single frame of test image at image size of* 64 × 64 *and* 128 × 128 *at clock period of 10ns.*

Image Size of 64 × 64

Table 6.5 shows the power consumption and comparison of Andra's, Lian's and the proposed SA-ALU processor for processing a single frame of test image Lenna, Baboon and Pepper at image size of 64 × 64 for each individual processor's actual process completion time. The evaluation duration for both Andra's and Lian's processor is 21920ns at clock period of 10ns. And the evaluation duration for the proposed SA-ALU remains at 82640ns at clock period of 10ns. The break-down of the power consumption of the processors into *Net Switching* and *Cell Internal Switching* were graphically presented in Figure 6.5.

Image size	Type of image	Type of architecture	Power consumption (mW)	Power consumption difference - w.r.t Andra's (mW)	Power consumption difference - w.r.t Andra's (%)	Power consumption difference - w.r.t Lian's (mW)	Power consumption difference - w.r.t Lian's (%)
64 × 64	Lenna	Andra's	3.50	-	-	0.23	6.95
		Lian's	3.28	-0.23	-6.50	-	-
		SA-ALU	1.77	-1.73	-49.52	-1.51	-46.01
	Baboon	Andra's	3.58	-	-	0.22	6.39
		Lian's	3.37	-0.22	-6.00	-	-
		SA-ALU	1.82	-1.76	-49.14	-1.55	-45.89
	Pepper	Andra's	3.57	-	-	0.22	6.51
		Lian's	3.35	-0.22	-6.11	-	-
		SA-ALU	1.79	-1.77	-49.75	-1.56	-46.48

Table 6.5: *Power consumption and comparison of Andra's, Lian's and proposed SA-ALU processor for processing a single frame of test image Lenna, Baboon and Pepper at image size of* 64 × 64. *Test conducted for actual process completion time of 21920ns at clock period of 10ns for both Andra's and Lian's processor. And 82640ns at clock period of 10ns for the proposed SA-ALU processor.*

The power consumption of both Andra's and Lian's processor had actually increased as shown in Table 6.5 and graph 6.5. The Andra's and Lian's power increased by about 45% and 60% respectively when comparing to the power consumption results evaluated at the proposed SA-ALU processor's process completion time. The increased power also increased the power saved by the proposed SA-ALU processor to an average of 49.47% over Andra's processor and 46.13% over Lian's processor.

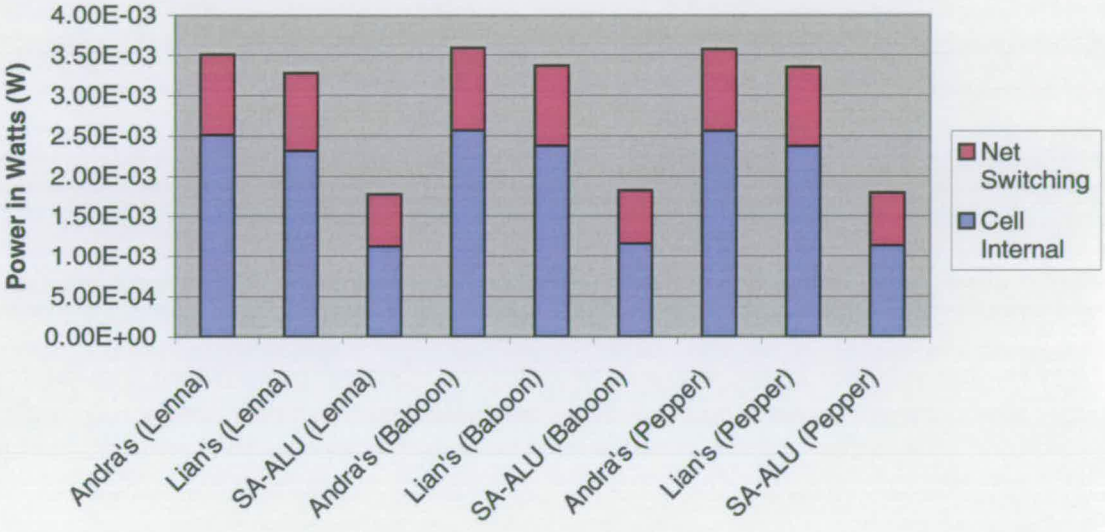


Figure 6.5: *Power consumption graph of Andra's, Lian's and the proposed SA-ALU processor for processing a single frame of test image Lenna, Baboon and Pepper at image size of* 64×64. *Test conducted for exact process completion time of 21920ns at clock period of 10ns for both Andra's and Lian's processor. And 82640ns at clock period of 10ns for the proposed SA-ALU processor.*

Image Size of 128×128

The results in Table 6.6 show the power consumption and comparison of Andra's, Lian's and the proposed SA-ALU processor for processing a single frame of test image Lenna, Baboon and Pepper at image size of 128×128. Power evaluations were conducted on the Andra's and Lian's processor for time duration of 84650ns at clock period of 10ns. And the power evaluation time duration for the proposed SA-ALU processor is 329000ns at clock period of 10ns. The graphical break-down of the power consumption of the processors into *Net Switching* and *Cell Internal Switching* are shown in Figure 6.6.

Image size	Type of image	Type of architecture	Power consumption (mW)	Power consumption difference - w.r.t Andra's (mW)	Power consumption difference - w.r.t Andra's (%)	Power consumption difference - w.r.t Lian's (mW)	Power consumption difference - w.r.t Lian's (%)
128 × 128	Lenna	Andra's	3.50	-	-	0.23	6.97
		Lian's	3.27	-0.23	-6.51	-	-
		SA-ALU	1.76	-1.74	-49.62	-1.51	-46.11
	Baboon	Andra's	3.59	-	-	0.22	6.47
		Lian's	3.37	-0.22	-6.08	-	-
		SA-ALU	1.83	-1.76	-48.91	-1.54	-45.60
	Pepper	Andra's	3.52	-	-	0.21	6.37
		Lian's	3.31	-0.21	-5.99	-	-
		SA-ALU	1.78	-1.74	-49.39	-1.53	-46.17

Table 6.6: *Power consumption and comparison of Andra's, Lian's and the proposed SA-ALU processor for processing a single frame of test image Lenna, Baboon and Pepper at image size of* 128 × 128. *Test conducted for actual process completion time of 84650ns at clock period of 10ns for both Andra's and Lian's processor. And 329000ns at clock period of 10ns for the proposed SA-ALU processor.*

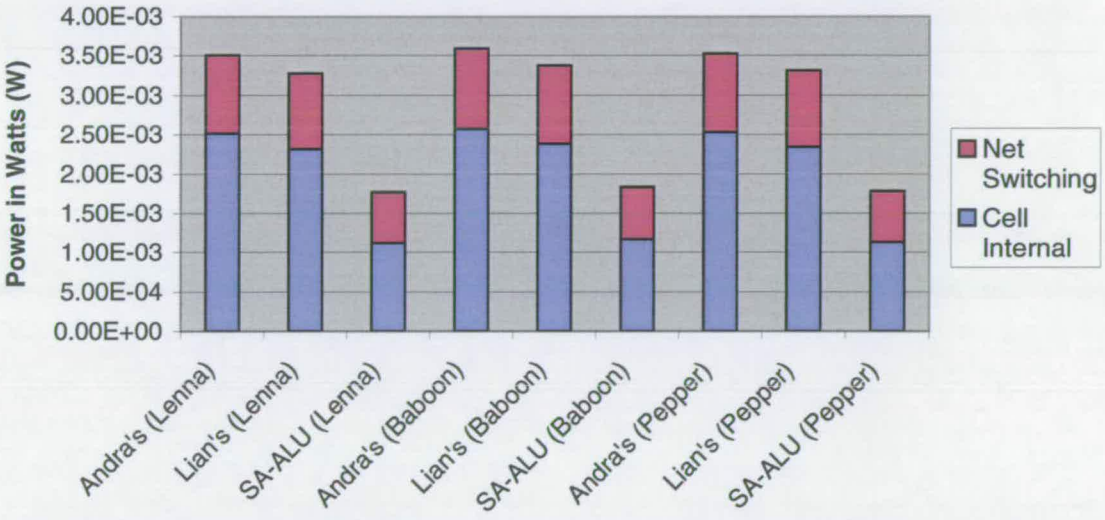


Figure 6.6: *Power consumption graph of Andra's, Lian's and the proposed SA-ALU processor for a single frame of test image Lenna, Baboon and Pepper at image size of* 128 × 128. *Test conducted for actual process completion time of 84650ns at clock period of 10ns for both Andra's and Lian's processor. And 329000ns at clock period of 10ns for the proposed SA-ALU processor.*

The results from Table 6.6 and graph 6.6 shows that there is a similar increase in power consumption of about 45% and 60%, for Andra's and Lian's processor respectively, with respect

to the power consumption results evaluated at the proposed SA-ALU processor's process completion time. The increase in the power consumption results are due to the difference in power consumption during the actual image processing period and during the processor idle period. The effect of the difference of the power between these two distinct periods in power analysis will be discussed in Section 6.3.2. Besides the increase in power, the results from all the processors are fairly consistent except for a slight power decrease with respect to the results obtained from the power evaluation with an image size 64×64. The slight decrease in power is due to averaging effect of power consumed by process set up. It is also noteworthy that all the three processors recorded an increased of power consumption when processing the test image Baboon. The increased in power is due to the fact that the power needed to process test image Baboon is slightly higher than the power consumed in the set-up process. This is due to the fact that there are a significantly higher number of high frequency components in the test image Baboon. As the number of pixel increase, the number of pixel with big adjacent Hamming distance (due to the high frequency component) also increases.

6.3.2 Analysis

In Section 6.3.1, the two sets of power evaluation results showed that the proposed SA-ALU centric processor consumes much less power than the pipelined data-path centric processor. However, there are discrepancies between the power consumption results obtained with an evaluation duration equal to the proposed SA-ALU processor process completion time and duration equal to individual process completion time for both Andra's and Lian's processor. The reason for these discrepancies is addressed in this section. All the power consumption profiles used in this section are Andra's processor power consumption profiles when it processes a 128×128 Pepper test image. The power consumption values and their time duration are as presented in the Section 6.3.1. The power profile of Andra's processor for performing one level of DWT on an 128×128 image is shown in Figure 6.7 at its process completion time of 84650ns at clock period of 10ns.

6.3.2.1 Power Consumption Profile Analysis

The power P_1 on the graph shown in Figure 6.7 is 3.52mW (which is the same as the value in Table 6.6). T_1 is the time duration for Andra's processor to complete one level of DWT on an 128×128 image. On normal power evaluation assumption, this power consumption value

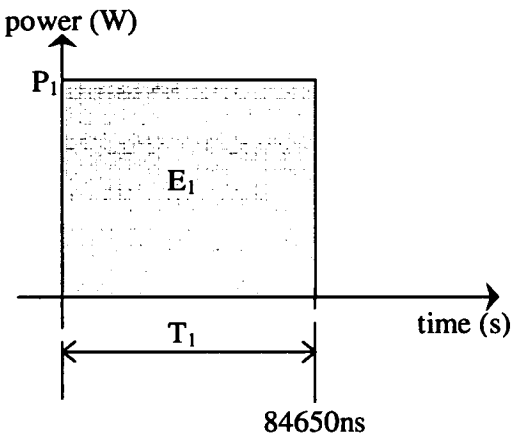


Figure 6.7: *Graphical representation of the average power consumed by Andra's processor for processing an* 128 × 128 *image at its process completion time of 84650ns at clock period of 10ns.* P_1 *is 3.52mW.* T_1 *is the Andra's processor process completion time of 84650ns.* E_1 *is energy consumed*

should remain the same for the remaining time after 84650ns. This assumption does not hold as the power consumption of Andra's processor evaluated at the proposed SA-ALU processor process completion time (T_3) of 329000ns at a clock period of 10ns shows a reduction in power consumption as depicted in Figure 6.8.

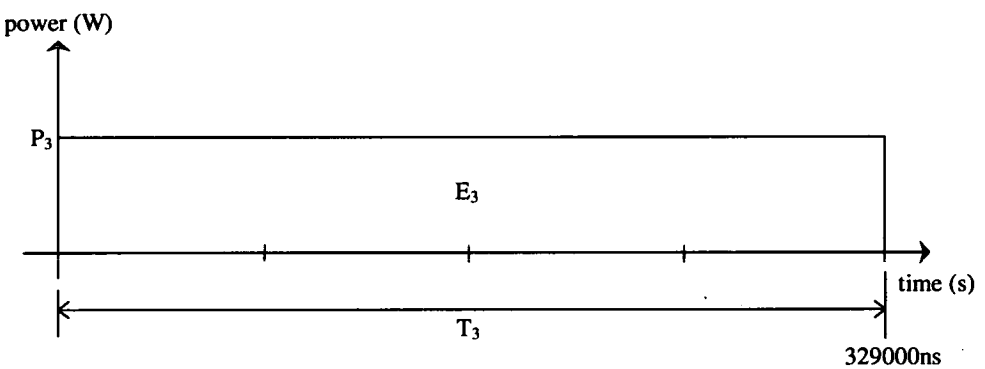


Figure 6.8: *Graphical representation of the average power consumed by Andra's processor for processing an* 128 × 128 *image after (*T_3*) 329000ns at clock period of 10ns.* P_1 *is 2.40mW.* E_3 *is energy consumed.*

The power consumption value fell from 3.52mW to 2.40mW (P_3). The reduction in power is due to averaging of the difference in power consumption during the actual image processing period and during the processor idle period. Therefore, a more accurate representation of the power consumption profile should be a representation that has two distinct power consumption

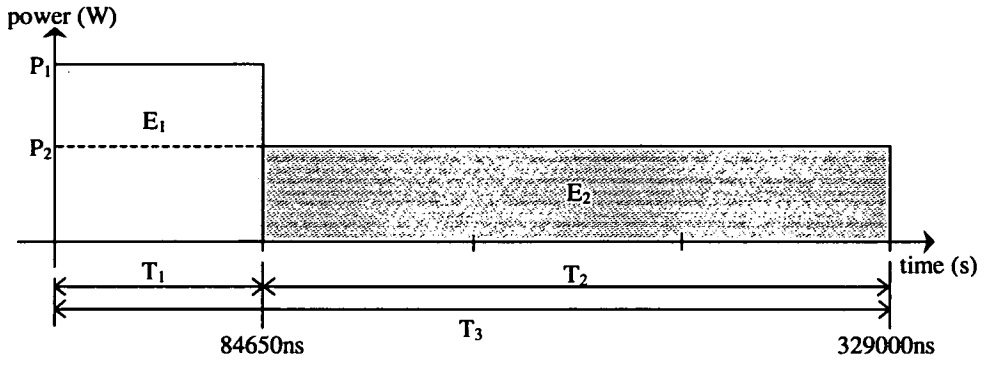


Figure 6.9: *Graphical representation of the power consumption profile of Andra's processor for processing an* 128 × 128 *image after 329000ns at clock period of 10ns.* $E_1 + E_2$ *is total energy consumed.* P_1 *is 3.52mW.* T_1 *is the Andra's processor process completion time of 84650ns.* E_1 *is the energy associated with* P_1. P_2 *is the power consumed for the remaining time* T_2 *after the process completion time.* E_2 *is the energy associated with* P_2. *The value for* P_2 *is 1.85mW.*

time regions as shown in Figure 6.9. The power consumption profile depicted in Figure 6.9 shows the power consumption of the two time periods T_1 and T_2. The power P_2 is the power consumed in the remaining idle period T_2 after it's process completion time. E_2 is the energy associated with P_2. The value for P_2 of 1.85mW can be obtained either by calculation or by empirical method. The idle period power consumption (P_2) of Andra's, Lian's and the proposed SA-ALU processor shown in Table 6.7 were obtained from power evaluation with null input value over a time duration of 329000ns at a clock period of 10ns.

Type of processor architecture	Type of power	Power consumption	Power consumption (%)	Total power consumption difference - w.r.t Andra's (mW)	Total power consumption difference - w.r.t Andra's (%)	Total power consumption difference - w.r.t Lian's (mW)	Total power consumption difference - w.r.t Lian's (%)
Andra's	Cell Internal Net switching Total	1.29mW 0.57mW 1.85mW	69.46% 30.54% 100%	-	-	0.37	24.92
Lian's	Cell Internal Net switching Total	1.03mW 453.13μW 1.48mW	69.48% 30.52% 100%	-0.37	-19.95	-	-
Proposed SA-ALU	Cell Internal Net switching Total	330.43pW 2.3nW 2.68nW	12.34% 87.66% 100%	-1.85	-99.99	-1.48	-99.99

Table 6.7: *The idle period power consumption (*P_2*) of Andra's, Lian's and the proposed SA-ALU processor with null input value for an evaluation duration of 329000ns at clock period of 10ns*

To prove that the reduction in the power consumption from 3.52mW to 2.40mW is due to the averaging effect of the two regions, the following calculations will use the two energy

values (E_1 and E_2) in the two regions to get the power consumption by Andra's processor when computing test image Pepper (P_3). In the following calculations, the power consumption figures are converted into energy figures. The idle period power consumption value of Andra's processor (shown in Figure 6.7) is also used. The area under the power curve/line is the energy consumed by the processor hardware. To get the energy consumed, the power function is integrated with respect to time as shown in Equation 6.1.

$$E = \int_0^{t_{pc}} P(t)dt \tag{6.1}$$

Therefore, energy E_1 consumed by Andra's processor when its power consumption is 3.52mW (P_1) (from Figure 6.7) for a period of 84650ns (T_1) is calculated as shown in Equation 6.2.

$$\begin{aligned} E_1 &= P_1 \times T_1 \\ &= 3.52mW \times 84650ns \\ &= 297.97nJ \end{aligned} \tag{6.2}$$

The idle time period T_2 is calculated as follows.

$$\begin{aligned} T_2 &= T_3 - T_1 \\ &= 329000ns - 84650ns \\ &= 244350ns. \end{aligned} \tag{6.3}$$

With the idle time period T_2 and the idle period power consumption, the idle time period energy idle E_2 can be calculated.

$$\begin{aligned} E_2 &= P_2 \times T_2 \\ &= 1.85mW \times 244350ns \\ &= 452.05nJ \end{aligned} \tag{6.4}$$

Hence, the both E_1 and E_2 can be added together to get total energy consumption for the whole 329000ns period.

$$\begin{aligned} E_3 &= E_1 + E_2 \\ &= 297.97nJ + 452.05nJ \\ &= 750.02nJ \end{aligned} \tag{6.5}$$

Finally, the average power P_3 consumed by Andra's processor when computing test image Pepper is calculated by the following.

$$\begin{aligned} P_3 &= \frac{E_3}{T_3} \\ &= \frac{750.02nJ}{329000ns} \\ &= 2.28mW \end{aligned} \tag{6.6}$$

The calculated power consumption value P_3 of 2.28mW in the above calculation is close to the value of 2.40mW shown in Table 6.3. This proves that the power consumption profile shown 6.9 is correct and the reduced in power is due to the averaging effect of the energy of the two distinct power consumption time regions.

Image size	Type of image	Type of architecture	Power consumption (mW)	Power consumption difference - w.r.t Andra's (mW)	Power consumption difference - w.r.t Andra's (%)	Power consumption difference - w.r.t Lian's (mW)	Power consumption difference - w.r.t Lian's (%)
128 × 128	Lenna	Andra's	2.01	-	-	0.40	24.65
		Lian's	1.61	-0.40	-19.78	-	-
		SA-ALU	0.11	-1.90	-94.54	-1.50	-93.19
	Baboon	Andra's	2.01	-	-	0.40	24.62
		Lian's	1.61	-0.40	-19.76	-	-
		SA-ALU	0.11	-1.89	-94.36	-1.50	-92.97
	Pepper	Andra's	2.02	-	-	0.40	24.47
		Lian's	1.62	-0.40	-19.66	-	-
		SA-ALU	0.11	-1.90	-94.56	-1.51	-93.23

Table 6.8: *Power consumption and comparison of Andra's, Lian's and the proposed SA-ALU processor for a single frame of test image Lenna, Baboon and Pepper at image size of* 128 × 128. *Test conducted for 5.5555ms at clock period of 10ns.*

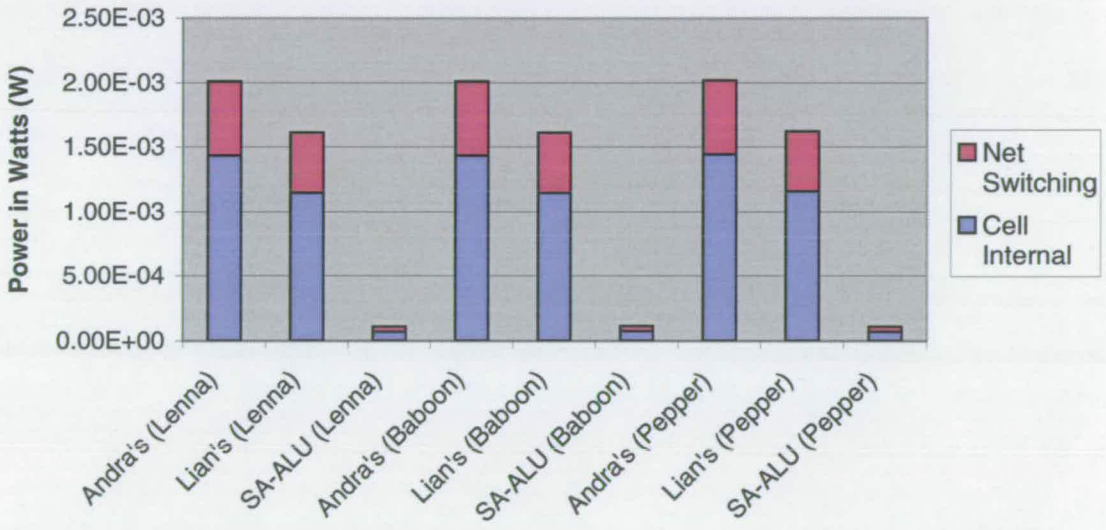


Figure 6.10: *Power consumption graph of Andra's, Lian's and the proposed SA-ALU processor for a single frame of test image Lenna, Baboon and Pepper at image size of* 128 × 128. *Test conducted for 5.5555ms at clock period of 10ns.*

To further demonstrate the averaging effect of the energy of the two distinct power consumption time regions, the power consumption of the three processors is evaluated for a time duration of 5.5555ms. This will show that the averaging effect will further reduce the overall power consumption value toward the idle period power consumption value as the idle period becomes proportionally much longer than active computation period. The 5.5555ms time duration is

chosen on assumption of a practical specification in mind. The 5.5555ms is derived from the assumption that there are 3 image components with a frame rate of 60 frames per second. Table 6.8 and Figure 6.10 shows the power consumption of Andra's, Lian's and the proposed SA-ALU processor computing one level of a DWT on test image Lenna, Baboon and Pepper at image size of 128×128 evaluated for a duration of 5.5555ms at clock period of 10ns.

That the power consumption value approaches the idle period power consumption value is evident from the value in the Table 6.8, where the idle period is purposely lengthened by lengthening the power evaluation duration to 5.5555ms. From Table 6.8, the power consumption of all three processors were significantly reduced as compared to the values given in Table 6.3 and 6.6. It can be seen that the proposed SA-ALU processor power consumption had dropped very significantly by about 93% compared to the values presented in Table 6.3. This significant reduction is due to the extremely low idle period power characteristic the proposed SA-ALU processor possessed. Note also that the Andra's processor power consumption value is approaching the idle period power consumption due to the averaging effect. The reduction in power again does not mean that the amount of energy consumed had decreased. On the contrary, the total energy consumed had increased.

6.3.2.2 Energy Efficiency of the Proposed SA-ALU Processor

Although the proposed SA-ALU processor power consumptions are lower than the two feed-through pipelined data-path centric processors in most of the cases presented in this thesis, the proposed SA-ALU processor is not considered as energy efficient as any of the two feed-through pipelined data-path centric processors. Consider the following energy consumption calculation. The power consumption value used in the calculation from the Pepper test image process shown in Table 6.6 (in Section 6.3.1.1).

$$\begin{aligned} E_{psa} &= P_{psa} \times 329000ns \\ &= 1.78mW \times 329000ns \\ &= 585.62nJ \end{aligned} \tag{6.7}$$

P_{psa} in Equation 6.7 is the power consumed by the proposed SA-ALU processor for processing

a 128 × 128 at clock period of 10ns with process completion time of 329000ns. And E_{psa} is the amount of energy consumed corresponding to P_{psa}. The energy consumption of 585.62nJ is about twice the amount of energy consumed by either one of the pipelined data-path centric processors (of 297.97nJ for Andra's processor as shown in the above Section 6.3.2.1) at their individual process completion time.

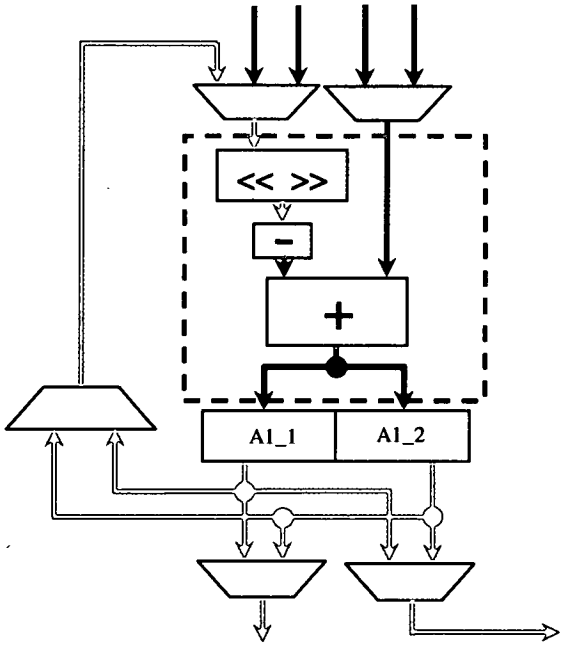


Figure 6.11: *Switching activities hot-zone of SA-ALU vertical (column) processor - ALUA1. Yellow connection nets are nets that have data with values of large Hamming distance with respect to the input data. Blue connection nets are nets that have only positive values with small Hamming distance. Red connection nets are nets that have both positive and negative values and that the values changes from positive to negative (and vice versa) frequently and data with values of large Hamming distance with respect to the input data. The area inside the red dashed line box indicates the switching activities hot-zone.*

The proposed SA-ALU processor consumes more energy per process function because of the frequent switching of 2's complements positive and negative numbers at the input of the subtractor-adder. The negative values and values that has very big Hamming distance with respect to the input data come from the accumulator ALUA1.1 and ALUA1.2 which is fed back into the input of the subtractor-adder via the multiplexer. As a result, a switching activities hot-zone is created. See Figure 6.11 for the switching activities hot-zone of the SA-ALU vertical (column) ALUA1 processor. The increased switching activities due to the feedback and multiplexing is much more higher than the other nets and components in the architecture as seen

in the Table 6.12. The two feed-through pipelined data-path centric processors are relatively more energy efficient because its feed-through architecture does not have to feedback data with different 2's complements polarity. However, the two feed-through pipelined data-path centric processors are more energy efficient per process function only when it is operating for the time duration of its process completion time or with clock manipulation.

Components	Power consumption (nW)	Power consumption (%)
Input Multiplexer (3 to 1)	52.6	6.77
Input Multiplexer (2 to 1)	25.9	3.33
Shifter	65.7	8.45
Number inverter	106	13.64
Adder	215	27.66
Accumulator A1_1	132	16.98
Accumulator A1_2	131	16.85
Feedback Multiplexer	25.5	3.28
Output Multiplexer (left)	11.3	1.45
Output Multiplexer (right)	12.3	1.58
Total	777.3	100

Figure 6.12: *Power consumption break-down of SA-ALU vertical (column) ALUA1 processor*

Without clock manipulation, the two feed-through pipelined data-path centric processors are very energy wasting. The pipelined data-path centric processors are energy wasting because the pipeline registers consume power even though its outputs are not changing when it is clocked in the idle period. The details of this effect is presented in Chapter 7. The power evaluation results and analysis of the two feed-through pipelined data-path centric processors with clock-manipulation are presented and discussed in Section 6.4.

6.4 Discussion

From the above analysis, it can be seen that for 1-D processing, the proposed SA-ALU processor is much more low power than the two existing feed-through pipelined data-path centric processors at the same clock speed without clock manipulation. However, there are discrepancies between the power consumption results of Andra's and Lian's processor obtained from the evaluation duration at the proposed SA-ALU processor process completion time and the individual process completion time.

This is due to fact that Andra's and Lian's processors have a faster process completion

time, hence an idle period which consumes less power then the active processing period. The averaging effect of both the active and idle period power consumption is the cause of the significant drop in the power consumption. It also shows that both Andra's and Lian's processors are more energy efficiency as regards to completing the 1-D DWT processing.

As such, the question about whether conventional power evaluation techniques are adequate arises. Comparing the power consumption evaluated with each individual hardware process completion time is not good enough. The reason is that conventional power evaluation assumes that the computational process takes place indefinitely at the same clock rate, which is not always true. Such a power evaluation method also does not give consideration to short process completion time. In image and video processing, computational activities take place on a frame by frame basis. Once a image or a frame of video had been processed, useful computational activities will cease. Therefore, the conventional power evaluation assumption of continual processing does not hold here.

6.4.1 New Criteria for Power Evaluation

Conventional power evaluation techniques are no longer adequate to give a fair and accurate power consumption view of hardware. Therefore, new criteria and ways of evaluating and comparing power consumption of hardware are needed. The criteria and considerations that are needed are as follows: -

- It must be task or process based
- It must be able to take into account of the total time which the hardware takes to complete its task
- It must be able to take into account the presence and the number of distinct power consumption time periods
- It must be able to log power consumption profile

From the analysis above, power evaluation alone is inadequate to fulfil all the criteria mentioned. As demonstrated in the previous two sub-sections, power consumption reading alone does not show adequately the power requirement differences between processing a small image and a big image. This means if the evaluation time duration is not given, there is no way

to tell whether one hardware is energy efficiency in relation to another or not, if both hardware units had different process completion time. The relationship between the power consumption of the active period and idle period impacts profoundly on the way power consumption of hardware is measured and evaluated. This shows that the power consumption figure alone do not have a convenient way of presenting an account of the power consumption per process. This is because power by definition is a function per unit time. Therefore, in order to establish a more wholesome picture of the power consumption figure of a hardware, it would be more appropriate to convert power consumption figures into the energy consumption figures. In this way, the energy consumption figures will be constant regardless of the clock rate and process completion time.

6.4.2 Better Low Power Design with Better Power Evaluation

Besides giving a better understanding of the power consumption of hardware, another advantage of using both the two type of consumption figures, is that it sheds light on ways of optimising and reducing the power consumption of hardware. For example, by knowing that hardware consumed different amounts of power in different process stages and periods, ways of reducing power consumption in a redundant and unproductive stage and period can be developed. One of the ways that have been developed to reduce the power consumption in an unproductive stage and period is by clock manipulation. Example of such a method and its effectiveness are described in the following.

6.4.2.1 Clock Manipulation

Clock-Gating

One way of manipulating the clock is clock-gating [15] (or clock shut-down of) the hardware when it is deem to go into an idle period or stage. The clock-gating technique is very efficient in reducing the power consumption for both Andra's and Lian's processor. This is because both of them have high power consumption values at idle periods. By gating the clock shut when the processors are in the idle period, significant power reductions were observed for both Andra's and Lian's processor as shown in Table 6.9 and Figure 6.13.

The power reduction is so significant that these now consumed only about half of the power consumed by the proposed SA-ALU processor. Note that the power consumption value of the

Image size	Type of image	Type of architecture	Power consumption (μW)	Power consumption difference - w.r.t Andra's (μW)	Power consumption difference - w.r.t Andra's (%)	Power consumption difference - w.r.t Lian's (μW)	Power consumption difference - w.r.t Lian's (%)
128 × 128	Lenna	Andra's	53.91	-	-	3.21	6.33
		Lian's	50.70	-3.21	-5.95	-	-
		SA-ALU	109.60	55.70	103.33	58.91	116.20
	Baboon	Andra's	55.11	-	-	2.97	5.70
		Lian's	52.14	-2.97	-5.39	-	-
		SA-ALU	113.33	58.22	105.65	61.19	117.37
	Pepper	Andra's	54.42	-	-	2.86	5.55
		Lian's	51.56	-2.86	-5.26	-	-
		SA-ALU	109.62	55.19	101.41	58.06	112.60

Table 6.9: *Power consumption and comparison of Andra's, Lian's and the proposed SA-ALU processor for a single frame of test image Lenna, Baboon and Pepper at image size of* 128 × 128. *Test conducted for 5.5555ms at clock period of 10ns with macro-function clock-gating.*

proposed SA-ALU processor remains the same as those shown in Table 6.8. The proposed SA-ALU processor does not benefit from the clock-gating as it is inherently clock-gated and the power it consumes in its idle period is already very low.

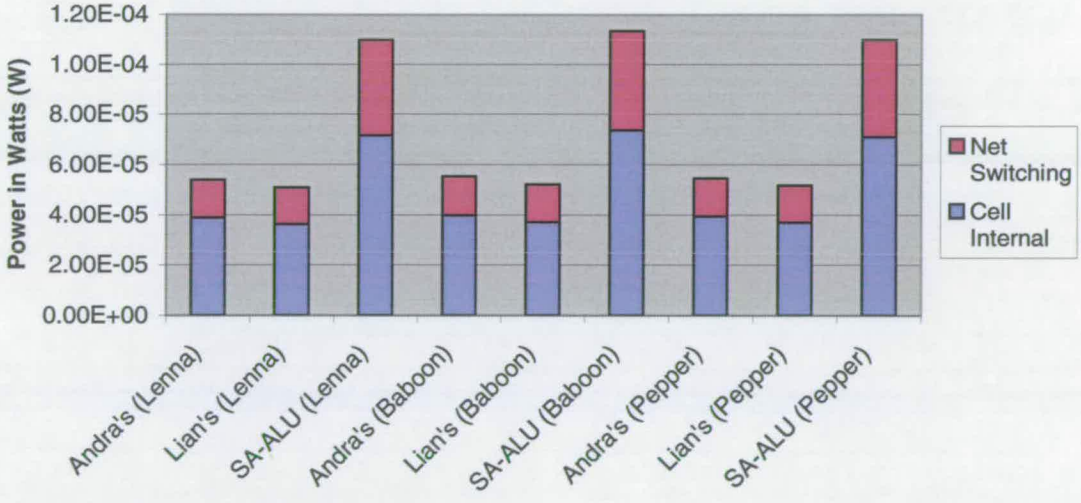


Figure 6.13: *Power consumption graph of Andra's, Lian's and the proposed SA-ALU processor for a single frame of test image Lenna, Baboon and Pepper at image size of* 128 × 128. *Test conducted for 5.5555ms at clock period of 10ns with macro-function clock-gating.*

Reduced Clock Frequency

However, there is a downside of using clock-gating. To utilise clock-gating, extra hardware area and extra effort are needed to design and to implement it. Fortunately, there are other techniques to manipulate the clock without incurring too much hardware overhead. One such technique is to reduce the frequency of the clock so that the useful active processes will be stretch out and therefore removing any idle period in between active processes. Table 6.10 shows the power consumption and comparison of Andra's, Lian's and the proposed SA-ALU processor for a single frame of test image Lenna, Baboon and Pepper at image size of 128×128. The power evaluation was conducted for 329000ns at clock period of 38.8457ns (25.74Mhz) for both Andra's and Lian's processor and at clock period of 10ns (100Mhz) for the proposed SA-ALU processor.

Image size	Type of image	Type of architecture	Power consumption (mW)	Power consumption difference - w.r.t Andra's (mW)	Power consumption difference - w.r.t Andra's (%)	Power consumption difference - w.r.t Lian's (mW)	Power consumption difference - w.r.t Lian's (%)
128 × 128	Lenna	Andra's	0.90	-	-	0.06	6.99
		Lian's	0.84	-0.06	-6.53	-	-
		SA-ALU	1.76	0.86	95.70	0.92	109.37
	Baboon	Andra's	0.92	-	-	0.06	6.48
		Lian's	0.87	-0.06	-6.08	-	-
		SA-ALU	1.83	0.91	98.47	0.97	111.33
	Pepper	Andra's	0.91	-	-	0.05	6.39
		Lian's	0.85	-0.05	-6.01	-	-
		SA-ALU	1.78	0.88	96.58	0.93	109.15

Table 6.10: *Power consumption and comparison of Andra's, Lian's and the proposed SA-ALU processor for a single frame of test image Lenna, Baboon and Pepper at image size of* 128×128. *Test conducted for 329000ns at clock period of 38.8457ns for both Andra's and Lian's processor and at clock period of 10ns for the proposed SA-ALU processor.*

Reducing the clock frequency is also very successful in reducing the power consumption of Andra's and Lian's processor as can be seen from Table 6.10 and Figure 6.14. There are very significant power consumption reductions in both Andra's and Lian's processors. The two processors now consume about only half of the proposed SA-ALU processor's power. These power consumption values of both Andra's and Lian's processor are very close to the values obtained by clock gating. The power consumption of Andra's processor when processing a

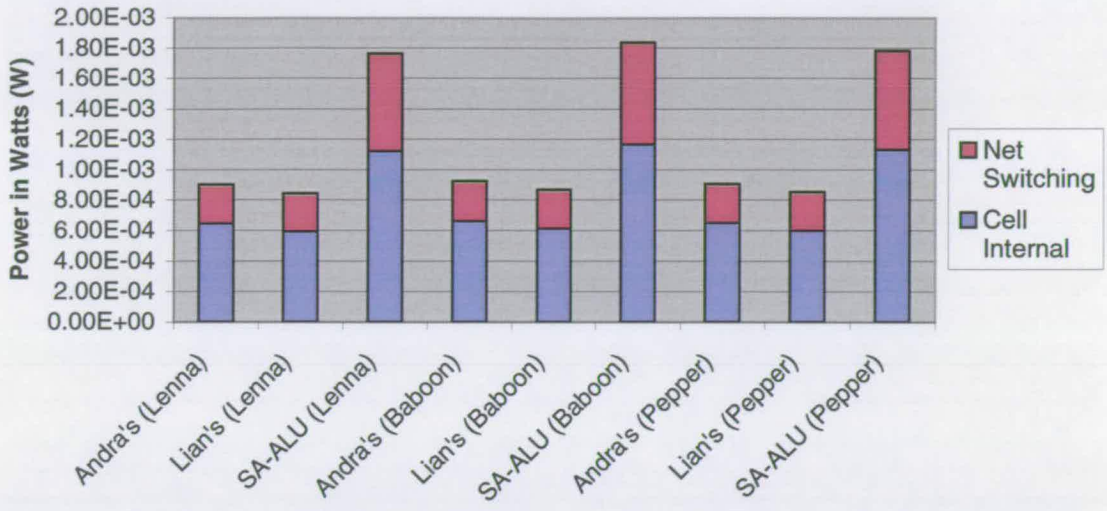


Figure 6.14: *Power consumption graph of Andra's, Lian's and proposed SA-ALU processor for a single frame of test image Lenna, Baboon and Pepper at image size of* 128×128. *Test conducted for 329000ns at clock period of 38.8457ns for both Andra's and Lian's processor and at clock period of 10ns for proposed SA-ALU processor.*

128×128 Pepper test image was reduced by 1.49mW which is a 62.14% reduction compared to the values presented in Table 6.3. These techniques had effectively spreading the energy consumed by the hardware, therefore reducing the power consumption value of these hardware. Thus, from Table 6.3 hardware with a high power consumption reading does not always necessarily mean that it is not energy efficient.

6.4.2.2 Constant Energy Consumption Value

The power consumption values of Andra's processor for processing 128×128 Pepper test image in Tables 6.6, 6.9 and 6.10 appeared to be very different from one another. However, the amount of energy consumed is about the same. The energy consumed by Andra's processor with process completion time of 84650ns at a clock period of 10ns for is 298.1551612nJ as shown in Equation 6.2. The other two cases of energy consumption resulted from clock manipulation of Andra's processor are calculated as follows.

Energy consumption of Andra's processor with clock-gating for the duration of 5.5555ms is calculated as shown.

$$\begin{aligned} E_{cg} &= P_{cg} \times 5.5555ms \\ &= 54.42\mu W \times 5.5555ms \\ &= 302.33nJ \end{aligned} \tag{6.8}$$

P_{cg} in Equation 6.7 is the power consumed by Andra's processor for processing a 128 × 128 image at clock period of 10ns with process completion time of 5.5555ms with clock-gating (from Table 6.9). And E_{cg} is the amount of energy consumed corresponding to P_{cg}. Energy consumption of Andra's processor with clock period of 38.8457ns for the duration of 329000ns is calculated as shown.

$$\begin{aligned} E_{sck} &= P_{sck} \times 329000ns \\ &= 0.91mW \times 329000ns \\ &= 299.39nJ \end{aligned} \tag{6.9}$$

P_{sck} in Equation 6.7 is the power consumed by Andra's processor with its clock period stretched from 10ns to 38.8457ns. And E_{sck} is the amount of energy consumed corresponding to P_{sck} (from Table 6.10). Both the above calculated energy consumption values for Andra's processor are very close to the value of 297.97nJ (E_1) calculated in Equation 6.2 in the above Section 6.3.2.1. The energy consumption value of 299.39nJ for Andra's processor operated with a clock period of 38.8457 ns is especially close to the value 297.97nJ. All these show that the energy consumption values are rather constant regardless of the evaluation time and the clock frequency. Hence, the energy consumption reading has added accuracy, consistency and convenience over the conventional power evaluation, analysis and comparison.

6.5 Summary

This chapter introduces the modified version of Andra's and Lian's feed-through pipelined data-path centric processor architecture. These two feed-through pipelined data-path centric

processor were modified for the lifting based 5/3 DWT and build with similar component as the proposed SA-ALU processor. Besides the feed-through pipelined centric processor architecture, the architecture of both the proposed SA-ALU column and row processors were also described in greater detail. Power analyses were conducted on Andra's, Lian's and the proposed SA-ALU column processors. Two set of main power evaluations were carried out on the three processors fed with test image Lenna, Baboon and Pepper. These test images come in two image sizes, these are 64×64 and 128×128. The two power evaluations that were carried out were based on two sets of evaluation times. The first set of power evaluations was conducted over a duration time which is equal to the proposed SA-ALU processor process completion time for all the mentioned processors. The second set of power evaluations was conducted for the duration of each individual processor's process completion time. There were discrepancies between the power consumption value obtained from an evaluation duration equal to the proposed SA-ALU processor process completion time and results of evaluation from each individual processor's process completion time. These discrepancies are due to the presence and the averaging of two distinct process time periods with two distinct power consumptions. As the evaluation duration gets longer, the idle period gets longer while the active period remains the same. When the duration of the idle period is long enough, its power consumption value starts to dominate. These discrepancies exposed the inadequacy of conventional power evaluation assumptions, techniques and tools. Therefore, having an energy consumption evaluation results and power consumption profile log alongside the existing power consumption evaluation result were proposed to give the conventional power consumption evaluation an added clarity. It has also been shown in this chapter, that for 1-D processing, the proposed SA-ALU processor is relatively lower power than the two existing feed-through pipelined centric processors without clock-gating and at clock frequency of 100Mhz (clock period of 10ns). However, the proposed SA-ALU processor is not power or energy efficiency as compared to the feed-through pipelined centric processors if the pipelined centric processors had their clock manipulated to reduce its idle time period in between active time periods.

Chapter 7

Low Power Multi-Segment One Hot Addressing

7.1 Introduction

High-speed, low-level, essential and repetitive sequential tasks in high throughput DSP systems are often the parts of the system that consume a considerable amount of power. One such essential power-consuming component of a DSP system is the address decoder or local addressing logic, which is mainly used for sequential memory selection. So far, no-one has targeted their research into reducing internal power consumption of local addressing or address decoder of a VLSI system, through switching area reduction and architecture optimisation. Most low power research work to date only considered reducing switching activities [40, 69–74] on the address bus by some means of bus coding. These techniques seek to reduce power consumption in transmitting addressing information from one part of a system to another part of a system, which assume that there are considerable amounts of capacitance on these buses. For example the work in [73, 74] was targeted in reducing power on an off-chip address bus by modifying and utilising transition coding. Most of these do not take into consideration the internal switching of the addressing logic and the additional hardware overhead that contribute to the overall power consumption of the system.

In this chapter, a novel scalable sequential multi-segment multi-level one-hot addressing architecture (MSML-OHA) is proposed to reduce the internal local sequential addressing power consumption. The architecture is made up of chains of one-hot shift registers segments and a combinational logic network for the final output selection. This architecture saves power by more than 20% over conventional counter-decoder architecture and it can be used as part of any low power system. A further power reduction of more than 59% was achieved in the dyadic MSML-OHA by combining the low power characteristic of MSML-OHA with a shared address line scheme.

This chapter is organised as follows: In Section 7.2, various sequential addressing architectures are presented. Next in Section 7.3, the proposed MSML-OHA are introduced and described. Following that, the power consumption of the presented architectures were evaluated and compared at different depths in Section 7.4. Discussions on why the proposed architecture is power efficient and how it can be further optimised are presented in Section 7.5. Finally, summary of the findings and contribution made by this study are given in Section 7.6.

7.2 Sequentially Addressed Memory

A computing and digital processing system utilizes sequentially addressed memory or sequential addressing in one form or another. The use of sequential addressing can be seen in First-In-First-Out (FIFO) memory, ring buffer memory and coefficients pointer. Sequential addressing is not only widespread, but also particularly important in digital signal processing applications where most of the computation proceeds in a sequential manner. Sequential addressing can be found in a wide variety of signal processing applications. One such application is video compression which heavily utilizes sequential addressing for major parts of its processes. It has been noted (in Section 4.3.3) that a large portion of the DWT process involves considerable amounts of sequential memory accesses. As the majority of the DWT processing involves computation in a sequential manner, the overall power consumption of the system can effectively be reduced if both the power of the memory addressing hardware and accesses within the DWT system are reduced.

7.2.1 Conventional Sequential Addressing Hardware

Conventionally, sequential addressing logic is built from two main components; the up-counter and the decoder (see Figure 7.1). The up-counter provides the sequentially incremental address whereas the decoder decodes the address into individual select-enable signals. This architecture is commonly used in digital systems for sequential addressing as well as to some extent for random addressing. The up-counter and the decoder (page 298-300 and 227-229 of [162]) can be broadly categorised into two logic group types. The up-counter is categorised as register-memory logic and the decoder as combinational logic. By simply clocking the up-counter, this design would be able to address sequentially. This simple conventional counter-decoder architecture (CCD) is relatively power efficient and easy to build when the

depth size (size of address space) is small.

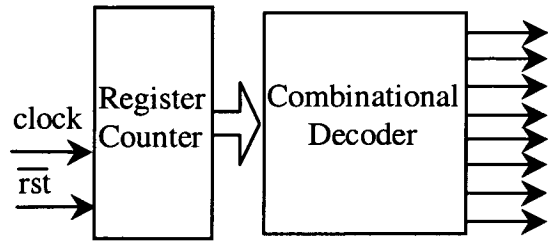


Figure 7.1: *Conventional counter-decoder (CCD) architecture*

However, when this architecture is scaled up, it is no longer power efficient. This is because when the counter size and the depth size increases, the number of gates in the combinational logic portion of the CCD increases exponentially. As a result of this increase in the number of combinational gates, the total amount of switching activities of the system is also increased. This is due to the fact that switching signals from counter would have to ripple through the increased number of gates before reaching the outputs. Such increase in switching paths not only increases the signal propagation delay, it also increases the power consumption of the system significantly. Therefore, alternative architectures were examined to try to minimize the power consumed by the sequential addressing hardware. One such architecture that has been examined is the one-hot one-bit shift-register architecture.

7.2.2 One-Hot Addressing Architecture

Besides shedding redundant address generators by sharing address lines (in Section 4.3.3), more can be done on the architecture of the addressing hardware to reduce the power consumption. Another way which the addressing hardware can reduce the power consumption is by isolating and localising the switching activities to a small area. One-hot addressing architecture (OHA) is one such hardware architecture which reduces power consumption by isolating switching activities to a small area. The OHA was first reported in Chapter 7 of [163] as a scheme for addressing. The one-hot addressing architecture is made up of a chain of one-bit shift-registers as shown in Figure 7.2. This architecture consists purely of register-memory logic (i.e. without combinational gates). The number N in the figure represents the total number of shift-registers and outputs in the chain.

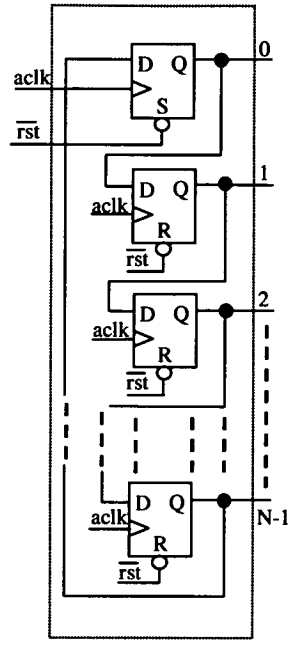


Figure 7.2: *One-hot shift-register*

Clock event	Output port select '7'	Output port select '6'	Output port select '5'	Output port select '4'	Output port select '3'	Output port select '2'	Output port select '1'	Output port select '0'
Tclk = 0	0	0	0	0	0	0	0	1
Tclk = 1	0	0	0	0	0	0	1	0
Tclk = 2	0	0	0	0	0	1	0	0
Tclk = 3	0	0	0	0	1	0	0	0
Tclk = 4	0	0	0	1	0	0	0	0
Tclk = 5	0	0	1	0	0	0	0	0
Tclk = 6	0	1	0	0	0	0	0	0
Tclk = 7	1	0	0	0	0	0	0	0
Tclk = 8	0	0	0	0	0	0	0	1
Tclk = 9	0	0	0	0	0	0	1	0
Tclk = 10	0	0	0	0	0	1	0	0
Tclk = 11	0	0	0	0	1	0	0	0
Tclk = 12	0	0	0	1	0	0	0	0
Tclk = 13	0	0	1	0	0	0	0	0
Tclk = 14	0	1	0	0	0	0	0	0
Tclk = 15	1	0	0	0	0	0	0	0

Table 7.1: *Output of OHA versus number of clock tick*

In order for the one-bit shift-register OHA to address sequentially, it must first be initialised by setting (or resetting) the first register (at the start of the chain) and resetting (or setting) the rest

of the registers. Then, by clocking the clock inputs of the registers, the active bit is shifted down the chain. Hence, with the active bit acting like a address pointer enables the OHA to advance sequentially. The output of each of the registers will serve as an address pointer/selector (see Table 7.1). As it is 'one-hot', only one register is active at one time. Therefore, this architecture has a constant and low number of switching activity (of 2 transitions per address advance). As a result, it is relatively faster than a CCD architecture. The Verilog code of an equivalent OHA is shown in Figure 7.3.

```
module ohaddos ( rst,
                 clkon,
                 stin,
                 decsel
                );

   parameter            noele = 4;

   input                rst;

   input                clkon;

   input                stin;

   output [noele-1:0]   decsel;

   reg [noele-1:0]      decsel;

   integer              i;


   always @(posedge clkon or negedge rst)
      begin
       if(!rst)
          begin
             decsel = ~0;
             decsel[0] = 1'b0;
          end
       else
          begin
             for(i=noele-1; i>0; i=i-1)
              begin
                 decsel[i] = decsel[i-1];
              end
             decsel[0] = stin;
          end
      end


endmodule
```

Figure 7.3: *Verilog code of an equivalent OHA*

An architecture similar to the OHA was used by Tsern et. al. [109] as a FIFO address pointer. Unfortunately, it is not low power since its depth size is 16 (which is greater than 8). The proof and reasons why this architecture is not low power will be presented in Section 7.5. Another

variation of the OHA is also proposed by Wu et. al. [164]. The *one-hot-zero* utilises all-zeros state as a valid state to reduce the total of flip-flops in the chain. Although, the all-zeros state results in an OHA architecture that is inherently clock gated, it still needs extra logic to produce a single select-enable signal line for the all-zeros state.

7.3 Multi-Segment Multi-Level OHA (MSML-OHA)

The short-comings of the traditional OHA can be overcome with the proposed novel multi-segment multi-level one-hot addressing architecture. The novel multi-segment multi-level OHA (MSML-OHA) consists of smaller segments of OHA which are designed to break-up the original long single one-hot chain. The structure of each segment of the one-hot chain is the same as the original one-hot chain (see Figure 7.2). The architecture is termed as multi-level as its originating clock signal has to propagate through levels of logic gates before reaching its final output. By segregating the OHA's one-hot chain into multi-segment and combining its output by layers of AND gate array clusters, switching activities can effectively be contained in a small area. As this architecture is made up of segments of one-hot chain and clusters of AND gate arrays, it is scalable and has many different possible configurations of multi-segment OHA. The following subsections describe some of the possible configurations that have been explored in this research.

7.3.1 Double-Segment Double-Level OHA (DSDL-OHA)

As mentioned in Section 7.2.2, the conventional OHA is not low-power (compared to CCD) when the depth size is greater 8. In order to achieve low power consumption at depth sizes of more than 8, a Double-Segment Double-Level (DSDL) OHA is proposed. The DSDL-OHA consists of two segments of one-hot shift-registers which are individually chained as shown in Figure 7.4. It is termed as double-level because the originating clock signal has to propagate through two levels (or layers) of logic gates before reaching its final output as seen in Figure 7.4. In this case, the originating clock has to be first gated through the one-hot shift-registers and then through the AND gate array. The Verilog code of an equivalent DSDL-OHA in 7.5.

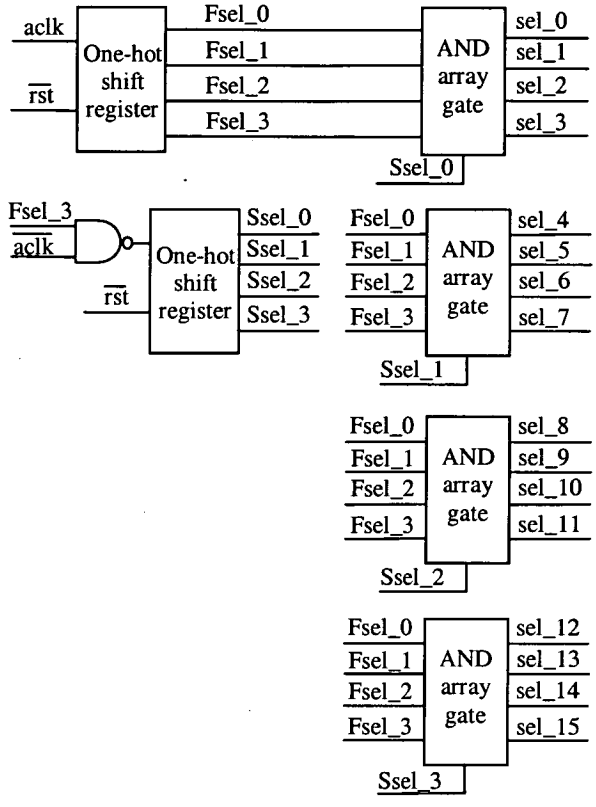


Figure 7.4: *DSDL-OHA 4x4 architecture*

```
`include "ohaddos.v"

`include "gandary.v"

module secclkdec2gl ( rst,
                      clk,
                      decsel
                    );

   parameter              depth = 32;

   parameter              sectsize = 4;

   parameter              sectmax = (depth/sectsize);

   input                  rst;

   input                  clk;

   output [depth-1:0]     decsel;

   wire [sectsize-1:0]    inpsel;

   wire [sectmax-1:0]     inssel;

   wire [depth-1:0]       decsel;

   wire                   clksec;
```

```
    ohaddos #(sectsize) flvl ( .rst(rst),
                               .clkon(!clk),
                               .stin(inpsel[sectsize-1]),
                               .decsel(inpsel[sectsize-1:0])
                               );

    ohaddos #(sectmax) slvl ( .rst(rst),
                              .clkon(clksec),
                              .stin(inssel[sectmax-1]),
                              .decsel(inssel[sectmax-1:0])
                              );

    gandary #(sectsize) gate[sectmax-1:0] ( .inp(inpsel),
                                            .ins(inssel[sectmax-1:0]),
                                            .gatesig(decsel)
                                            );

    assign clksec = ~(clk & ~inpsel[sectsize-1]);

endmodule
```

Figure 7.5: *Verilog code of an equivalent DSDL-OHA*

The output of OHA is combined and gated through one level of AND gate array (see Figure 7.6 and 7.7). Each AND gate array consists of two sets of inputs. The two sets of input are the main inputs (inN) and the enable input (enb_in). The input data signals destined for the outputs are connected to the main inputs. An active signal (of logic 'high') on the enable input will switch the input signal to the outputs otherwise the outputs will remain inactive (logic 'low').

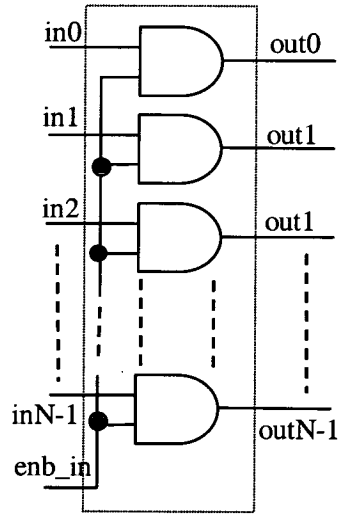


Figure 7.6: *AND gate array*

```
module gandary (  inp,
                  ins,
                  gatesig
                );

   parameter            depth = 128;

   input [depth-1:0]    inp;

   input                ins;

   output [depth-1:0]   gatesig;

   reg [depth-1:0]      gatesig;

   integer              i;

always @(inp or ins)
   begin
      gatesig = ~0;
      for(i=0; i<depth; i=i+1)
       begin
          gatesig[i] = inp[i] | ins;
       end
   end

endmodule
```

Figure 7.7: *Verilog code of an equivalent 2-input AND gate array*

By dividing the one-hot shift-registers into two segments and combining these with an AND gate array, the number of outputs of the first segment is effectively multiplied by the second segment. For instance, in the DSDL-OHA (seen in Figure 7.4) the effective depth size of the addressing architecture is 16 (which is 4×4). It can be seen from the figure that there are two sets of OHAs. All the outputs of the first OHA are connected directly into the main inputs of all the AND gate array. Whereas, each of the second OHA's output are connected to the enable input of each AND gate array. When the DSDL-OHA is clocked, the first OHA will advance sequentially. However, the second OHA will only advance when it is clocked if the fourth output of the first OHA (Fsel_3) is active. Therefore, only one AND gate array cluster is enabled every 4^{th} clock cycle. The enabled AND gate array will then switch the output signal of the first OHA through to its output.

As the signal from the registers only passes through one level of AND gate array, this architecture is (one combinational gate delay) slower than OHA. However, as a single gate delay is small, it does not have much of an impact on its performance. The propagation delay of the switching is fairly constant except in instances when a signal roll over from segment to segment occurs (i.e. at the moment when the AND gate array is enabled).

7.3.2 Triple-Segment Double-Level OHA (TSDL-OHA)

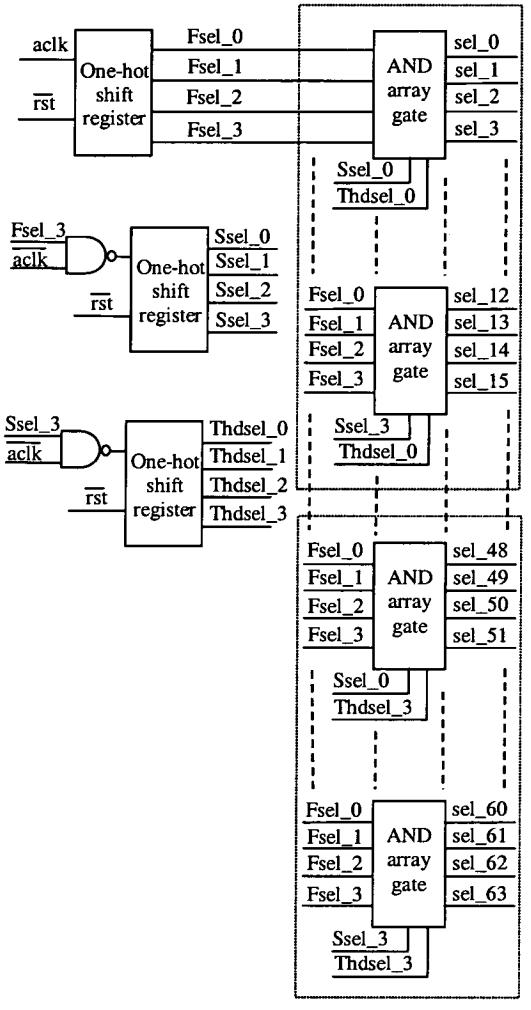


Figure 7.8: *TSDL-OHA 4x4x4 architecture*

This concept of segmentation is further extended to get two variations of three segments OHA. One of these is the Triple-Segment Double-Level (TSDL) OHA. This TSDL-OHA (Figure 7.8) has one OHA segment more than DSDL-OHA. The Verilog code for the TSDL-OHA is shown in Figure 7.9. However, instead of a 2-input AND gates array, TSDL-OHA uses a 3-input AND gate array. With this third segment, the number of outputs is multiplied yet again.

```
`include "/u43/kct/JPEG2000/design/HDL/addec/ohaddos.v"

`include "/u43/kct/JPEG2000/design/HDL/addec/gatri.v"

module trilvlsecgl ( rst,
                     addadv,
                     decsel
                     );

   parameter             depth = 256;

   parameter             numlvl = 4;

   parameter             sectsize = 16;

   parameter             sectmax = (depth/sectsize);

   parameter             newsect = (sectmax/numlvl);

   input                 rst;

   input                 addadv;

   output [depth-1:0]    decsel;

   wire [sectsize-1:0]   inpsel;

   wire [newsect-1:0]    inssel;

   wire [numlvl-1:0]     inlsel;

   reg [sectmax-1:0]     insselp;

   reg [sectmax-1:0]     inlselp;

   wire [depth-1:0]      decsel;

   wire                  clksec;

   wire                  clkthd;

   integer               i, j, x, y;

   ohaddos #(sectsize) flvl ( .rst(rst),
                              .clkon(!addadv),
                              .stin(inpsel[sectsize-1]),
                              .decsel(inpsel[sectsize-1:0])
                              );


   ohaddos #(newsect) slvl ( .rst(rst),
                             .clkon(clksec),
                             .stin(inssel[newsect-1]),
                             .decsel(inssel[newsect-1:0])
                             );

   ohaddos #(numlvl) thrdlvl ( .rst(rst),
                               .clkon(clkthd),
                               .stin(inlsel[numlvl-1]),
                               .decsel(inlsel[numlvl-1:0])
                               );
```

```
    gatri #(sectsize) gate[sectmax-1:0] ( .inp(inpsel),
                                          .ins(insselp[sectmax-1:0]),
                                          .inl(inlselp[sectmax-1:0]),
                                          .gatesig(decsel)
                                          );

    assign clksec = ~(addadv & (~inpsel[sectsize-1]));

    assign clkthd = ~(~clksec & (~inssel[newsect-1]));

    always @(inssel)
       begin
        for(i=0; i<numlvl; i=i+1)
           begin
              for(j=0; j<newsect; j=j+1)
               begin
                  insselp[j+(newsect*i)] = inssel[j];
               end
           end
       end

    always @(inlsel)
       begin
        for(x=0; x<numlvl; x=x+1)
           begin
              for(y=0; y<newsect; y=y+1)
               begin
                  inlselp[y+(newsect*x)] = inlsel[x];
               end
           end
       end

endmodule
```

Figure 7.9: *Verilog code of an equivalent TSDL-OHA*

Like the DSDL-OHA, the TSDL-OHA combines the output of the segments of OHA with one level of AND gate array. As such, the nominal delay for TSDL-OHA is one combinational gate delay slower than OHA. However, in cases where there is a signal roll over from one segment to another segment, the worst case propagation delay is two combinational gate delays longer than OHA. This is due to the fact that the signal needs to ripple from the first OHA segment through to the third segment. Hence, the propagation delay is not constant.

7.3.3 Triple-Segment Triple-Level OHA (TSTL-OHA)

The Triple-Segment Triple-level (TSTL) OHA (see Figure 7.10) is similar to TSDL-OHA except that it has another level of AND gate array added to it. The Verilog code of an equivalent TSTL-OHA is shown in Figure 7.11. This architecture is the slowest architecture presented in this study because of the third level of logic gate the signals have to propagate through. However, the addition of another level of AND gate array has a profound impact on its power consumption. The power consumption figure of the TSTL-OHA will be presented

Section 7.4.2. Like the DSDL-OHA, this architecture uses its second and third segment output to individually control signal gating at the first and second AND gate array.

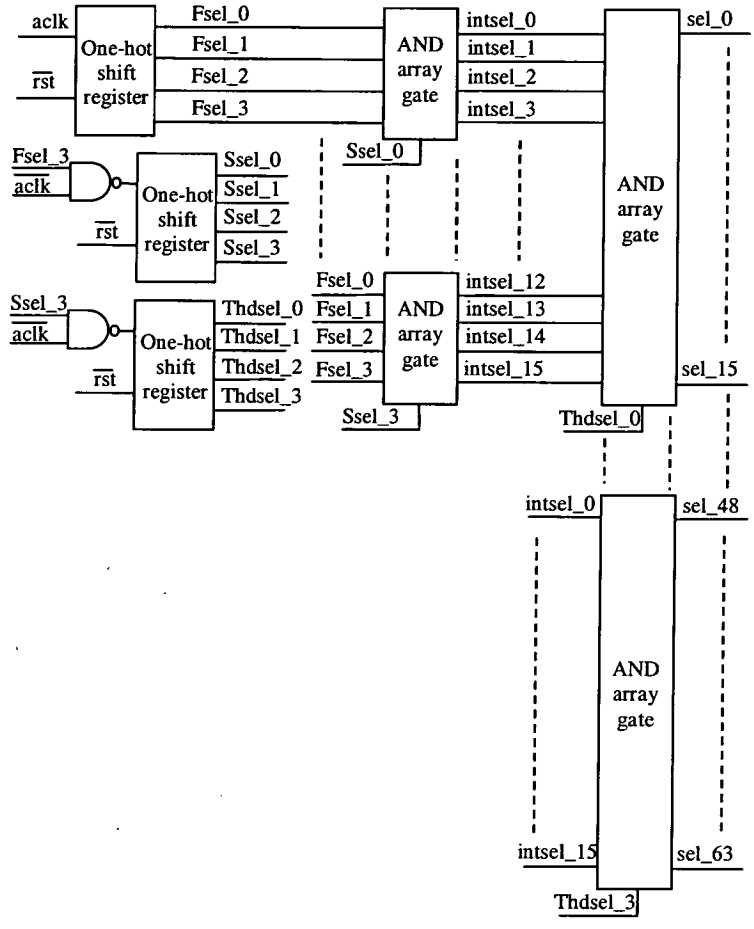


Figure 7.10: *TSTL-OHA 4x4x4 architecture*

```
`include "/u43/kct/JPEG2000/design/HDL/addec/ohaddos.v"

`include "/u43/kct/JPEG2000/design/HDL/addec/gandary.v"

module trilvlsecIIgl ( rst,
                       addadv,
                       decsel
                     );

   parameter             depth = 128;

   parameter             sectsize = 4;

   parameter             thdstg = 8;

   parameter             grpsectsize = (depth/thdstg);

   parameter             numsect = (grpsectsize/sectsize);
```

```
    input                  rst;

    input                  addadv;

    output [depth-1:0]     decsel;

    wire [sectsize-1:0]    inpsel;

    wire [numsect-1:0]     inssel;

    wire [thdstg-1:0]      inlsel;

    wire [grpsectsize-1:0] stcon;

    wire                   clksec;

    wire                   clkthd;


    ohaddos #(sectsize) flvl ( .rst(rst),
                               .clkon(!addadv),
                               .stin(inpsel[sectsize-1]),
                               .decsel(inpsel[sectsize-1:0])
                               );


    ohaddos #(numsect) slvl ( .rst(rst),
                              .clkon(clksec),
                              .stin(inssel[numsect-1]),
                              .decsel(inssel[numsect-1:0])
                              );

    ohaddos #(thdstg) thrdlvl ( .rst(rst),
                                .clkon(clkthd),
                                .stin(inlsel[thdstg-1]),
                                .decsel(inlsel[thdstg-1:0])
                                );

    gandary #(sectsize) secgate[numsect-1:0] ( .inp(inpsel),
                                               .ins(inssel),
                                               .gatesig(stcon)
                                               );

    gandary #(grpsectsize) trigate[thdstg-1:0] ( .inp(stcon),
                                                 .ins(inlsel),
                                                 .gatesig(decsel)
                                                 );

    assign clksec = ~(addadv & (~inpsel[sectsize-1]));

    assign clkthd = ~(~clksec & (~inssel[numsect-1]));

endmodule
```

Figure 7.11: *Verilog code of an equivalent TSTL-OHA*

7.3.4 Dyadic Multi-Segment Multi-Level OHA (DMSML-OHA)

To suit practical applications like the DWT better, dyadic multi-segment multi-level OHA (DMSML-OHA) is needed. Dyadic addressing hardware is designed to address memory locations in multiple of twofold. The DMSML-OHA are used in both the intermediate memory and re-process memory of the SA-ALU centric lifting-based 5/3 DWT hardware architecture. The dyadic feature is needed to be incorporated into the addressing hardware because it is one of the fundamental attributes of sub-coding and DWT process as presented in Chapter 3. An example of a dyadically addressed output is shown in Table 7.2. The table shows the output of a 3 level dyadic OHA at depth size of 16. Conventional dyadic addressing architecture is similar to CCD. The DMSML-OHA is also implemented using an up-counter and a decoder, except that there is additional logic to detect and reset the counter when a dyadic value is reach. The dyadic value is held in another loadable up-down counter which can be incremented or decremented by multiple of two when desired.

Clock Event	Decoder output when dyadic control = '001' (level 1)	Decoder output when dyadic control = '010' (level 2)	Decoder output when dyadic control = '100' (level 3)
Tclk = 0	0000000000000001	0000000000000001	0000000000000001
Tclk = 1	0000000000000010	0000000000000010	0000000000000010
Tclk = 2	0000000000000100	0000000000000100	0000000000000100
Tclk = 3	0000000000001000	0000000000001000	0000000000001000
Tclk = 4	0000000000010000	0000000000010000	0000000000000001
Tclk = 5	0000000000100000	0000000000100000	0000000000000010
Tclk = 6	0000000001000000	0000000001000000	0000000000000100
Tclk = 7	0000000010000000	0000000010000000	0000000000001000
Tclk = 8	0000000100000000	0000000000000001	0000000000000001
Tclk = 9	0000001000000000	0000000000000010	0000000000000010
Tclk = 10	0000010000000000	0000000000000100	0000000000000100
Tclk = 11	0000100000000000	0000000000001000	0000000000001000
Tclk = 12	0001000000000000	0000000000010000	0000000000000001
Tclk = 13	0010000000000000	0000000000100000	0000000000000010
Tclk = 14	0100000000000000	0000000001000000	0000000000000100
Tclk = 15	1000000000000000	0000000010000000	0000000000001000

Table 7.2: *Dyadic decoder's output with respect to the dyadic level and clock signal - logic '1' in the output indicates that particular output is selected. 'Tclk' stands for the clock tick.*

The dyadic multi-segment multi-level (DMSML-OHA) is essentially the same as the MSML-OHA. Like conventional dyadic addressing hardware, the DMSML-OHA also has additional logic circuit which enables it to address dyadically. However, this additional logic is comparatively smaller than that of the conventional dyadic addressing hardware. This

additional logic allows the addressing hardware to address dyadically is called the 'turn around' logic. The 'turn around' logic that is inserted into the register-memory logic portion of the MSML-OHA is as shown on Figure 7.12.

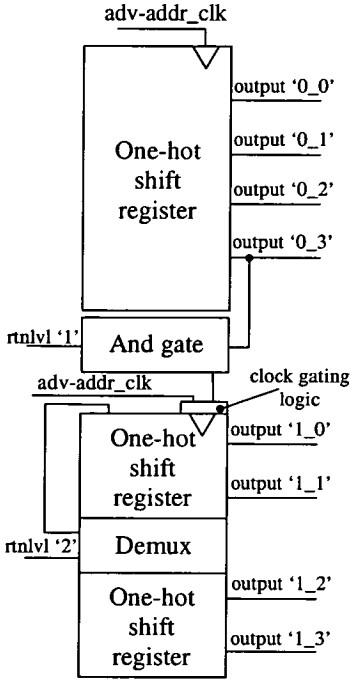


Figure 7.12: *Dyadic MSML-OHA (DMSML-OHA)*

The configuration shown in Figure 7.12 is a 3 levels depth-size of 16 DMSML-OHA. The 'turn around' logic is inserted either between two segments or within a single segment. The 'turn around' logic enables the OHA to cut short the route for the active output bit to return, at the appropriate dyadic position, to the top of the one-hot chain. The 'turn around' logic consists of a multiplexer and an OR gate (see Figure 7.13) for turn around within a segment and a simple AND gate for turn around in between two segments (see Figure 7.12).

7.4 Power Evaluation Results

The power analysis is divided into three main parts. The first part of the power evaluation focuses on OHA. Power comparison with CCD architecture will be made. The second part of the power evaluation focuses on the DSDL-OHA, TSDL-OHA, TSTL-OHA and CCD circuits. The final part is the power evaluation of DMSML-OHA circuits. Results of these power evaluations will be compared against the power consumption figures of CCD. All designs in

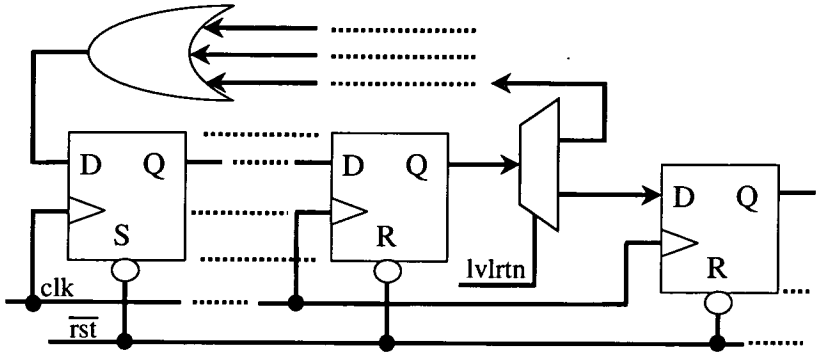


Figure 7.13: *Dyadic 'turn around' logic*

this chapter were coded and evaluated with the flow and softwares described in Section 2.4.1 of Chapter 2.

7.4.1 OHA

Results from the power evaluations on the OHA at depth 4, 8 and 16 are as shown in Table 7.3. The negative value in Table 7.3 indicates a reduction in power consumption while a positive value indicates an increase in power consumption.

Depth size	Type of Architecture	Power (μW)	Power difference (μW)	Power difference %
4	CCD OHA	1.055733 0.7301997	-3.255329	-30.83
8	CCD OHA	1.536677 1.423499	-0.1131779	-7.37
16	CCD OHA	2.027821 2.928043	0.900222	44.39

Table 7.3: *Power evaluation results of OHA*

It can be seen from Table 7.3 that there is a significant power reduction with OHA at the depth size of 4. The power saving is mainly due to the absent of the combinational gate decoder. However, at the depth size of 8, only a small power reduction was achieved as the power consumed internally by shift register flip-flops is slowly dominating the power consumption figures. At a depth of 16, the power consumption actually increases by 44%. This is due to the fact that the power saved by the absence of the combinational gate decoder has been overwhelmed by the amount of power consumed internally by shift register flip-flops.

7.4.2 Multi-Segment Multi-Level OHA

Power evaluations were performed on the multi-segment OHA. As the meaningful minimum number of register in an OHA is 4, entries for DSDL-OHA commence from depth size of 8. In the same way, entries TSDL-OHA and TSTL-OHA commence from depth size of 16. The evaluations are divided into 3 portions as follows: -

7.4.2.1 Depth Size of 8, 16 and 32

The Table 7.4 stipulates the results of the power evaluations (with 100 clock events) on DSDL-OHA at depth sizes of 8, 16 and 32. This table also shows the results of TSDL-OHA and TSTL-OHA at depth sizes of 16 and 32. As the depth size of the architecture is a product of the number of registers in the first segment and second segment, different combinations of the number of registers can be used to achieve the same depth size. As such, there are two different DSDL-OHA entries each at depth size of 16 and 32. Likewise, there is more than one combination for TSDL-OHA and TSTL-OHA at depth sizes greater than 64. As seen from Table 7.4, different combinations of register numbers can affect the power consumption of the addressing architecture.

Depth size	Type of Architecture	Power (μW)	Power difference (μW)	Power difference %
8	CCD	1.536677	-	-
	DSDL-OHA-4x2	1.536784	0.000107	0.01
16	CCD	2.027821	-	-
	DSDL-OHA-8x2	2.129009	0.101188	4.99
	DSDL-OHA-4x4	1.771555	-0.256266	-12.64
	TSDL-OHA-4x2x2	1.856205	-0.171616	-8.46
	TSTL-OHA-4x2x2	1.891720	-0.136101	-6.71
32	CCD	2.297859	-	-
	DSDL-OHA-8x4	2.323781	0.025922	1.13
	DSDL-OHA-4x8	1.709058	-0.588801	-25.62
	TSDL-OHA-4x4x2	2.312827	0.014968	0.65
	TSTL-OHA-4x4x2	2.091590	-0.206269	-8.98

Table 7.4: *Power evaluation results of multi-segment OHA at depth size of 8, 16 and 32*

Results from Table 7.4 shows that there is no significant power reduction shown for both TSDL-OHA and TSTL-OHA at all depth sizes and for DSDL-OHA at depth of 8. This is because the power saved by the absence of a combinational gate decoder has been over taken by the additional power consumption needed for gating the signal from the OHA segments

to the output. In order for any MSML-OHA to be low power, the power saving due to the absence of the combinational gate decoder must be significantly greater than the additional power consumed by the OHA segments and the additional logic needed for gating the signal from the OHA segments to the output.

At the depth sizes of 16 and 32, significant power reduction can be observed from DSDL-OHA 4×4 and 4×8 as the combinational gate decoder (in the CCD) starts to consume more power than the DSDL-OHA. However, the power consumption of DSDL-OHA remains high if the number of shift-registers in the first OHA segment is greater than 4 (i.e. DSDL-OHA 8×2 and 8×4).

7.4.2.2 Depth Size of 64 and 128

Table 7.5 shows the power evaluation results of DSDL-OHA, TSDL-OHA and TSTL-OHA (which were evaluated over 400 clock events) at depth size of 64 and 128. Note that at depth size of 128, there are three different entries each for TSDL-OHA and TSTL-OHA. From the table, it can be seen that all the addressing architectures except for the TSTL-OHA have noticeable power consumption increases.

Depth size	Type of Architecture	Power (μW)	Power difference (μW)	Power difference %
64	CCD	11.5954	-	-
	DSDL-OHA-16x4	15.0941	3.4987	30.17
	DSDL-OHA-8x8	11.1341	-0.4613	-3.98
	DSDL-OHA-4x16	15.6215	4.0261	34.72
	TSDL-OHA-4x4x4	14.0312	2.4358	21.01
	TSTL-OHA-4x4x4	8.8594	-2.736	-23.6
128	CCD	13.7395	-	-
	DSDL-OHA-16x8	16.957	3.2175	23.42
	DSDL-OHA-8x16	16.6128	2.8733	20.91
	DSDL-OHA-4x32	26.1091	12.3696	90.03
	TSDL-OHA-4x4x8	22.4866	8.7471	63.66
	TSDL-OHA-8x4x4	15.8352	2.0957	15.25
	TSDL-OHA-4x8x4	22.6275	8.888	64.69
	TSTL-OHA-4x4x8	10.7559	-2.9836	-21.72
	TSTL-OHA-8x4x4	10.8386	-2.9009	-21.11
	TSTL-OHA-4x8x4	11.5757	-2.1638	-15.75

Table 7.5: *Power evaluation results of multi-segment OHA at depth size of 64 and 128*

At depth size of 64, most of the MSML-OHA's power consumption has increased because

all the MSML-OHA (except for TSDL-OHA) have more than 4 shift-register flip-flops in one of its OHA segments. However, TSTL-OHA is able to gain a significant amount of power reduction, as switching activities within its architecture are restricted to only a small area and the switched capacitances were kept low with the 3 levels configuration. At depth size of 128, TSDL-OHA-4 $\times$ 4 $\times$ 4's power consumption has increased even though none of its OHA is greater than 4. This is due to a significant increased in switched capacitance caused by the substantial increases in the number of connections from the OHA segments to the AND gate array inputs.

7.4.2.3 Depth Size of 256

The results of DSDL-OHA's power consumption were not included in Table 7.6 as their values were too high. However, three more entries were added for both TSDL-OHA and TSTL-OHA. These bring the number of configuration entries to a total of 6 each for each of the two type of architecture.The power consumption of these architectures was evaluated for the duration of 800 clock events.

Depth size	Type of Architecture	Power (μW)	Power difference (μW)	Power difference %
256	CCD	28.6038	-	-
	TSDL-OHA-4x8x8	90.9674	62.3636	218.03
	TSDL-OHA-8x4x8	50.1409	21.5371	75.29
	TSDL-OHA-8x8x4	50.3627	21.7589	76.07
	TSDL-OHA-4x4x16	91.7088	63.105	220.62
	TSDL-OHA-4x16x4	92.97	64.3662	225.03
	TSDL-OHA-16x4x4	42.9211	14.3173	50.05
	TSTL-OHA-4x8x8	26.0657	-2.5381	-8.87
	TSTL-OHA-8x4x8	25.4304	-3.1734	-11.09
	TSTL-OHA-8x8x4	25.7758	-2.828	-9.89
	TSTL-OHA-4x4x16	34.3538	5.75	20.10
	TSTL-OHA-4x16x4	35.6379	7.0341	24.59
	TSTL-OHA-16x4x4	33.1718	4.568	15.97

Table 7.6: *Power evaluation results of multi-segment OHA at depth size of 256*

From the results, all but three entries of TSTL-OHA had an increase in power. TSTL-OHA-4 $\times$ 8 $\times$ 8, 8 $\times$ 4 $\times$ 8 and 8 $\times$ 8 $\times$ 4 were the three that possess a low power characteristic. These TSTL-OHAs were able to have a significantly lower power consumption figures compare to CCD because these TSTL-OHAs keep the switching activities to small designated area and keep the switched capacitance low with the 3 levels configuration. The power saved was significant

enough to overwhelm the power dissipated by the combinational gate decoder and the 8 bit counter in the CCD. The rest of the architectures shown on the table recorded a huge increase in power consumption. This is because the power consumed by the large number of shift-register flip-flops and the additional gating logic in the MSML-OHA were exceeding that of the CCD.

7.4.3 Dyadic Multi-Segment Multi-level OHA

The power evaluation results of DMSML-OHA at equivalent depth size of 64, 128 and 256 are shown in Table 7.7. Comparison with the power consumption of CCD at actual depth size is made. The power of the addressing architecture in question evaluated with the number of clock events as indicated on Table 7.7 at the clock period of 10ns. Note that DMSML-OHA has a effective depth size which is double its actual depth size. This is due to the saving in the overall architecture design of the intermediate memory (see Chapter 4) which exploited the sharing of the address select/enable lines. The sharing of the address select/enable lines for two blocks of memory effectively multiples the number of address locations by two, therefore the effective depth size is double the actual depth size. The combination of multi-segment OHA and the sharing of address lines significantly reduced the power consumption of the addressing architecture as can be seen in Table 7.7.

Depth size	Type of Architecture	Number of Events	Power (μW)	Power difference (μW)	Power difference %
64	CCD-64	400	185.9169	-	-
64* (32)	DSDL-OHA32-4x8	200	73.54890	-112.3680	-60.44
64* (32)	DSDL-OHA32-4x8 (Dyadic)	200	75.65330	-110.2636	-59.31
128	CCD-128	800	218.3935	-	-
128* (64)	TSTL-OHA64-4x4x4	400	69.66590	-148.7276	-68.10
128* (64)	TSTL-OHA64-4x4x4 (Dyadic)	400	74.00650	-144.3870	-66.11
256	CCD-256	1600	227.4792	-	-
256* (128)	TSTL-OHA128-4x4x8	800	84.14220	-143.3370	-63.01
256* (128)	TSTL-OHA128-4x4x8 (Dyadic)	800	84.57400	-142.9052	-62.82

Table 7.7: *Power evaluation results of dyadic multi-Segment OHA at depth size of 64, 128 and 256. The notation * indicates the effective depth size where the value in brackets ( ) are the actual depth size*

The power consumption figures of both DMSML-OHA and MSML-OHA stipulated in Table 7.7 show that both the addressing architectures had a considerably large power reduction compared to CCD at all depth sizes. From the table, it can be seen that there is more than 59%

of saving across the board for both the DMSML-OHA and MSML-OHA.

7.4.4 Summary

The graph in Figure 7.14 shows the best case power savings of the different OHA and MSML-OHA architectures at different depth sizes. The positive value in the figure represents the percentage of power saving compared to CCD architecture. A negative value in the graph indicates that the addressing architecture is consuming more power than CCD architecture. From the results presented in the above section and Figure 7.14, it can be seen that there is a significant power reduction at depth size of 4 for OHA. The graph also shows that multi-segment OHAs have significant power reduction at all depth sizes (except at 4). It can also be seen that only one type of the multi-segment OHA architecture is power efficient at certain range of depth sizes. OHA is very power efficient at depth size of 4 to 8. DSDL-OHA is power efficient at depth size of 16 to 64. TSTL-OHA is power efficient from depth size of 16, all the way to depth size of 256. And lastly, the TSDL-OHA is power efficient only at depth size 16.

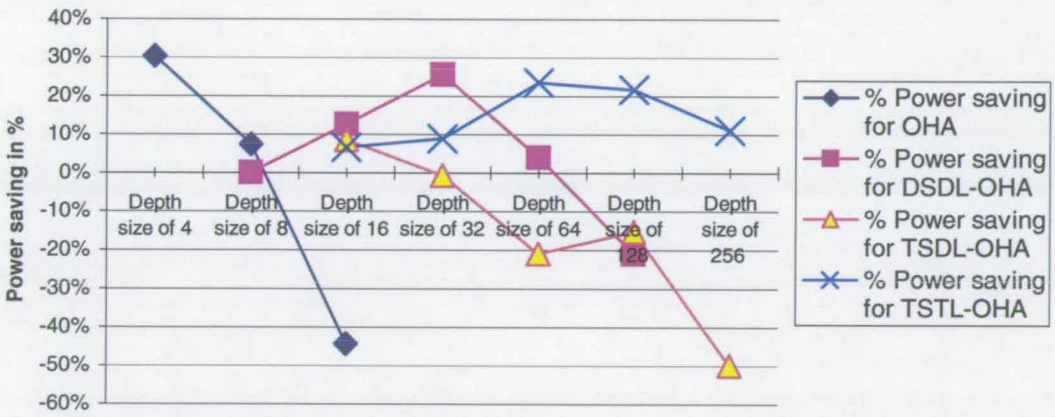


Figure 7.14: *Best case power saving (with respect to CCD) at different depth size*

It can be seen from Figure 7.14 that the multi-segment OHA reduces about 20% of the power compared to CCD. A further reduction of at least 59% (see Figure 7.15) compared to CCD was observed in the DMSML-OHA when the combining the power reduction effect of multi-segment OHA with the shared address line scheme which is an power optimisation at a higher architectural level. From Figure 7.15, it can be seen that greater power saving can be achieved by combining two levels of architectural power reduction techniques.

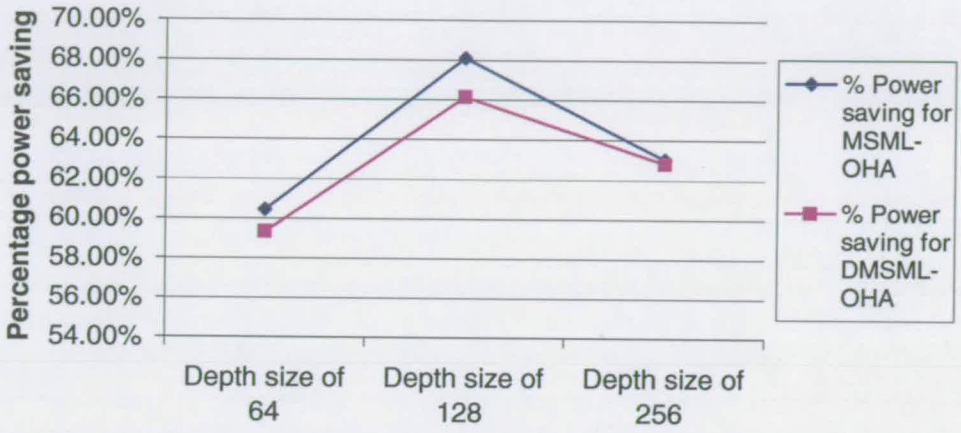


Figure 7.15: *Power saving of shared address lines DMSML-OHA and MSML-OHA compared to CCD at address depth size of 64, 128 and 256*

7.5 Discussion

From Section 7.4.2, it is apparent that not all the MSML-OHA are low power compared to the CCD. Whether or not a MSML-OHA is low power, depends a lot on the combinational gates' switching activities and the number of clocked registers in the architecture. In Section 7.2.1, it was noted that a considerable amount of power is consumed when switching signal ripples through layers of the combinational gates in the addressing architecture. However, these combinational gates are not the sole contributor to the redundant power. The total number of flip-flop registers in the whole address, the number of flip-flop registers in each segment and the sequence of how the segments are arranged greatly affects the addressing hardware's power consumption. The amount of power which the segment sequence can impinge on can be seen in the DSDL-OHA at depth size of 32 when the sequence of the number of flip-flops is changed from 8×4 to 4×8. This is due to the fact that the edge-triggered flip-flop consumes about 3 times more power than a combinational gate, when it is clocked, even without any switching at its output (see Table 7.8). The power consumption values in Table 7.8 were obtained from power evaluating a D-type flip-flop and a 2 input AND gate over 300 clock events at clock period of 10ns.

The above-mentioned attribute is also the reason why OHA are not low power (see Section 7.4.1) when its depth size is greater than 8. The power saving achieved by eliminating the combinational gate switched capacitance was soon overtaken by the internal power consumption of clocked flip-flops. As such, by reducing the total number of registers in the addressing architecture and the number of registers in a single one-hot chain where

	Power (μW)	Power difference (μW)	Power difference %
D-type flip-flop	6.970100	-	-
AND gate	1.940772	-5.029328	-72.16

Table 7.8: *Power consumption of AND gate compared with D-type flip-flop - power consumption values were obtained from 300 clock cycles of power evaluation*

frequent switching takes place, the overall power consumption will be reduced. Therefore, by segregating the OHA's one-hot chain into multi-segment and combining its outputs with a group of AND gate arrays to form MSML-OHA, the power consumption in the register-memory logic are minimized. However, by combining its outputs with a group of AND gate arrays creates additional combinational logic switching capacitance in the addressing architecture. Therefore, in order to achieve power reduction, an optimal combination of the number of segments and registers in the addressing architecture needs to be determined. The optimal combination can be obtained by an empirical method which is the approach adopted in this thesis. The MSML-OHA has an area overhead of about 25% to 45% larger than CCD. It can also be scaled up to higher segments and levels. However, by scaling it up, signal propagation delay will also increase as the level and number of segment increases. This increase in signal delay nevertheless will be fairly constant, except when a signal roll over from segment to segment occurs.

7.6 Summary

This chapter introduces the critical and essential role of sequentially addressed memory and logic in modern computer and signal processing systems. The widespread and heavily used sequential addressing logic circuit contributes a significant proportion of the system's power. Therefore, reducing the power consumption of sequential addressing logic will greatly impact the overall power consumption. To understand and to reduce the power of sequential addressing logic, the make-up of the CCD and its higher power consumption at depth size of greater than 4 were presented. It was found that by reducing the amount of switching activities in the combinational gate area, the addressing circuit's overall power can be reduced. The OHA was proposed as a possible solution to eliminate the combinational logic and its switching

power. However, OHA has it's limitation as it consumes more power than CCD when its depth size is greater than 8. Hence, the novel MSML-OHA and its application extension DMSML-OHA were proposed. Power evaluation results on MSML-OHA shows that these architectures reduce power on average by more than 20% compared to CCD. However, not all of the configurations of the MSML-OHA are power efficient. Some of these configurations are not power efficient because the configurations in question uses more clocked register than a CCD and a clocked register or flip-flop consumes 3 times more power than a clocked combinational gate. Finding the optimal number of registers and register segments in the MSML-OHA will help to ensure that the MSML-OHA is power efficient. A further reduction of at least 59% (compared to CCD) in the DMSML-OHA was achieved when combining the power reduction effect of multi-segment OHA with the shared address line scheme. Such is an example of power reduction optimisation with the combination of two architectural levels of optimisation. The MSML-OHA architecture has an area overhead of about 25% to 45% greater than CCD. It is scalable and can be scaled up to higher segments and levels. However, its signal propagation delay will also increase as the level and number of segment increases. The increase in signal delay will be fairly constant except when a segment signal roll over occurs. In all of these aspects, this address architecture can be used to trade off area and speed for power reduction. This architecture is suitable to be used as part of any low power system.

Chapter 8
Summary and Conclusion

8.1 Introduction

The purpose of this thesis has been to investigate whether a low power JPEG2000 5/3 Discrete Wavelet Transform (DWT) lifting based hardware architecture can be realised with low power techniques at both the algorithmic and the architectural level. Besides that, the aim is also to determine whether a pipelined data-path centric processor design is more power consuming than an Arithmetic Logic Unit (ALU) centric processor. The novel SA-ALU centric JPEG2000 5/3 DWT architecture has been designed and implemented for this purpose using power reduction optimisation at both the algorithmic and architectural level. The power consumption figures of the proposed SA-ALU centric JPEG2000 5/3 DWT architecture were compared with existing conventional architectures.

This final chapter summarises the work presented in the main body of this thesis, evaluates the extent to which the original goal has been accomplished and shows clearly the contribution to the research on low power realisation of JPEG2000 5/3 DWT architecture, low power design and power evaluation.

The remainder of this chapter is organised as follows: Section 8.2 draws conclusions from the work presented in this thesis. Section 8.3 highlights what has been achieved as a result of this research. Section 8.4 draws up the future work which would add on to the knowledge already gained from this research and discusses any possible ways to achieve a even lower power JPEG2000 5/3 DWT hardware architecture. Finally, the final comments regarding the thesis are made in Section 8.5.

8.2 Conclusions

This thesis set out to realise a low power JPEG2000 5/3 lifting based DWT hardware architecture. It can be seen from the information in the Chapter 2 that many power reduction

strategies and techniques have been developed to counter the effect of ever-increasing power consumption of high performance systems. These power reduction strategies and techniques have been very effective in reducing the power of these systems especially when power reduction techniques are applied at more than one level of design abstraction. As such this thesis concludes that it is feasible for more than one design abstraction levels be applied to an image compression hardware system.

In Chapter 3, it can be seen that image and video compression is essential to many applications. The gap and need for lossless image compression coding in a wide range of applications were met by the release of the JPEG2000 standard by ISO/IEC which provides lossless DWT coding. With the lifting based DWT algorithm most of the computation redundancy is removed from the traditional DWT. Due to the computation intensive nature of image compression systems, it is concluded in Chapter 3 that any low power realisation of any standard image and video compression system is always needed and welcomed.

From Chapter 4, it can be seen that there is no low power realisation of the JPEG2000 5/3 lifting based DWT. Therefore, a low power JPEG2000 5/3 lifting based DWT VLSI hardware is needed. As existing implementations are feed-through pipelined data-path based architecture, this thesis sought to ascertain whether feed-through pipelined data-path based processor architecture is more power consuming compared to a feedback accumulating ALU based processor architecture. As such, a novel JPEG2000 5/3 lifting based DWT VLSI hardware was built around Shifter-Accumulator Arithmetic Logic Unit (SA-ALU) processors. From the information presented in Chapter 4, it is evident that a viable SA-ALU centric lifting based architecture has been realised. From Chapter 4 and 5 it was shown that a architecture with a lower hardware overhead is obtained when the design is optimised at both the algorithmic and architectural level.

Innovative scheduling algorithm presented in Chapter 4 has shown to reduce the number of memory buffers needed for the transformation as well as reducing the address generating hardware for memory access by sharing of addressing lines. This is evidence that the optimisation at the algorithmic level gives room and spurs on architectural design optimisation. The reverse is also true as a good hardware architecture design gives room for algorithmic optimisation and exploration.

Incorporating the embedded data-extension into the main operations of the proposed SA-ALU

centric architecture is an instance where the architecture design allowed for an optimised algorithm. This novel embedded data-extension algorithm can be used by any lifting based DWTs. From Chapter 5, the embedded data-extension algorithm eliminated both the need for a explicit data-extension operation and its associating extension hardware, reducing both the power and area consumption of the overall hardware architecture. Thus, it can be concluded that there are more low power optimisation possibilities when both algorithmic and architectural power optimisation issues are considered together. By exploiting the close relationship between the algorithm and the architecture a more optimised hardware can be realised.

Chapter 6 shows the analyses and values from the power evaluation of the Andra's, Lian's and the proposed SA-ALU centric processor. It can be seen from the values of the two main power evaluations, that there are obvious discrepancies between the value obtained from evaluation duration equal to the proposed SA-ALU processor process completion time and value from evaluation duration of each individual processor's process completion time. It can be concluded from these discrepancies that there are inadequacies in the conventional power evaluation assumption and method. It shows that power consumption value alone is no longer enough, especially for evaluating the power consumption of hardware running complex processes with idle intervals. Therefore, the total energy consumption value of hardware processing a complete task or function along with the power consumption profile log are needed.

From the power evaluation value of the Andra's, Lian's and the proposed SA-ALU centric processor, it can be seen that the SA-ALU centric processor is much more low power than the two presented feed-through pipelined data-path centric processors when clocked at the same frequency. However, when the clock manipulation technique is applied to both the two feed-through pipelined data-path centric processors, both Andra's and Lian's processor are twice as power and energy efficient as the proposed SA-ALU centric processor. Therefore, it is concluded that the proposed SA-ALU centric processor is consider as power efficient when clocked with the same frequency as the feed-through pipelined data-path centric processor, but the proposed SA-ALU centric processor is not as energy efficient per task/function as the two feed-through pipelined data-path centric processors.

Chapter 7 presents the novel multi-segments multi-level one-hot addressing architecture (MSML-OHA). The MSML-OHA is made up of individual chains of one-hot shift registers segments and clusters of combinational AND gate array. The MSML-OHA is the outcome of optimisation done at the architecture level by containing and isolating switching activities to a

small defined local area.

From the power evaluation values of the MSML-OHA it can be seen that its power consumption are significantly lower than that of the conventional counter-decoder (CCD) addressing architecture. Hence, isolating switching activities into a small local area by segmenting the overall architecture is effective in reducing switching activities area and thus the power consumption of the architecture. A further significant reduction in power was observed when the MSML-OHA is utilized together with the shared address line architecture described in Chapter 4. It is concluded that local architectural level optimisation (of containing and isolating the switching activities a small local area) together with the optimisation at a system architecture level reduced the power consumption of addressing hardware significantly.

8.3 Achievements

The research present in this thesis required to design and developed a number of algorithm, hardware architecture and their power analysis. The following highlights these developments:

- Designed and implemented a novel SA-ALU Centric JPEG2000 5/3 lifting based DWT system architecture.
- Designed novel SA-ALU centric processor system which formed majority parts of the lifting based DWT system architecture.
- Implemented the explicit data-extension hardware.
- Power evaluated an explicit data-extension hardware.
- Designed the novel low power embedded extension algorithm which can be used by any lifting based DWTs.
- Incorporated the novel low power embedded extension algorithm into the SA-ALU Centric JPEG2000 5/3 lifting based DWT architecture.
- Implemented Lian et. al's and Andra et. al's pipelined centric JPEG2000 5/3 lifting based DWT architecture.
- Designed and implemented the novel low power multi-segment one-hot sequential addressing architecture at various address depths.

- Designed and implemented the novel low power multi-segment one-hot sequential addressing architecture with 5-levels dyadic structure.
- Power evaluated Lian et. al's design and Andra et. al's design.
- Power evaluated the SA-ALU centric processor (with embedded extension) and compares the results to pipelined data-path centric architecture (Lian et. al's design and Andra et.al's design).
- Power evaluated low power multi-segment sequential addressing one-hot architecture (with and without dyadic structure) at various address depths and compares the results to conventional counter and decoder sequential addressing architecture. More than 50% of power saving were achieved.
- First known ALU centric architecture versus pipelined data-path centric architecture power evaluations and comparisons were done.

8.4 Future Work

Although this thesis has strived to give a rigorous investigation of the research objective as outlined in Section 8.1, there are still areas which can potentially add to the knowledge already gained from the research presented.

The embedded data-extension was proven to reduce significantly the amount of processing steps and power when incorporated into the main operation of a processor. Currently, only the proposed SA-ALU centric processor because of its flexible architecture can be incorporated with the embedded data-extension. Investigation to ascertain whether Andra's or Lian's processor can be modified to incorporate the embedded data-extension together with the clock-manipulation technique presented in Chapter 6 to yield the low power benefits of both technique would be worthwhile.

Most of the switching activities in the SA-ALU processor occurs in the adder as a result of the large Hamming distance of the different data from the input registers and the accumulator of which some of them are negative 2's complement data. Further research and development can be made on the SA-ALU processor architecture to ensure that negative value results would not be fedback and switched together with a positive value. Alternatively, a sign magnitude number

system and operation can also be explored to determine whether the switching activities of whole JPEG2000 5/3 lifting based DWT can be further reduced.

In Chapter 6 the inadequacy of the current and conventional power evaluation and modelling tools were highlighted. Research into better power evaluation methods and development of new tools that will provide an energy consumption figure and a power profile log are needed.

8.5 Final Comments

Low power design and engineering is an important area of research as it is one of the main points that decides the viability of application for widespread usage. Image/video compression has been seen to be increasingly being used in many applications especially in portable form. As a result, there is much research work on the low power implementation of these image/video applications.

Therefore, the motivation of this research was to realised a low power JPEG2000 5/3 Discrete Wavelet Transform (DWT) lifting based hardware architecture. And to establish whether with power optimisation techniques at the algorithmic and the architectural level, a better hardware architecture can be realised. Besides that, the aim was also to ascertain whether a feed-through pipelined data-path centric processor design consumed more power than an accumulating ALU centric processor.

With the embedded data-extension algorithm, low data-path and memory area consumption design, efficient memory access and the low power MSML-OHA, it is concluded that utilising both (closely related) algorithmic and architectural level power optimisation design can significantly reduces power consumption of the JPEG2000 5/3 lifting based DWT hardware. It is also concluded that the proposed SA-ALU centric processor is power efficient as compared to the two presented feed-through pipelined data-path centric processors when all of them were clocked with the same frequency. Therefore from the information and results presented this thesis, it is concluded that a low power JPEG2000 5/3 Discrete Wavelet Transform (DWT) lifting based hardware architecture has been realised.

Appendix A
Euclidean Algorithm

The Euclidean algorithm is an algorithm for finding the greatest common divisor (GCD) of two integer numbers a and b. However, it can also be use to factor polynomials into lifting steps. The filter polynomial $a(z)$ is to be factored into the form of $b(z)q(z) + r(z)$. They are related as $a(z) = b(z)q(z) + r(z)$, where $a(z)$ is the dividend, $b(z)$ the divisor, $q(z)$ the quotient and $r(z)$ the remainder. All of these factors can be obtained by long division of $a(z) \div b(z)$. The following are the steps to obtain the value of $q(z)$ and $r(z)$ needed for factoring filter into lifting steps.

To find the GCD, the algorithm will normally have to run iteratively until the remainder ($r_n(z)$) is zero. However, as factoring of the filter into lifting steps occurs only one step at a time, only one iteration is needed at one time.

Step 1: Initialise $a_0(z)$ ***and*** $b_0(z)$***:-***

To start the process of value of $a_0(z)$ and $b_0(z)$ need to be first initialised. For example for the equation $\widetilde{LPF}_e(z) = \widetilde{t}_i(z)\widetilde{LPF}_o(z) + \widetilde{Y}_{L1}(z)$, $\widetilde{LPF}_e(z)$ is $a_0(z)$ and $\widetilde{LPF}_o(z)$ is $b_0(z)$.

Step 2: Compute the following:-

Starts with n=0.

$$q_n(z) = a_n(z)/b_n(z) \tag{A.1}$$

$$r_n(z) = a_n(z)\%b_n(z) \tag{A.2}$$

$$a_{n+1}(z) = q_n(z) \tag{A.3}$$

$$b_{n+1}(z) = r_n(z) \tag{A.4}$$

The symbol '/' in A.1 represents the division operation which divides $a_n(z)$ with $b_n(z)$ keeping the quotient and discarding the remainder. The symbol '%' in A.2 represents the modulus

operation which divides $a_n(z)$ with $b_n(z)$ keeping the remainder and discarding the quotient. $a_{n+1}(z)$ and $b_{n+1}(z)$ of A.3 and A.4 are the value of $a(z)$ and $b(z)$ of the next iteration respectively. In the case of lifting steps, finding $a_{n+1}(z)$ and $b_{n+1}(z)$ may not be necessary, it is covered here for completeness.

Depending on the number of term in $a_n(z)$, there will be more than one possible factorization. In this case, all factor must be considered.

Step 3: Then the results are written as:-

$$a_n(z) = b_n(z)q_n(z) + r_n(z) \tag{A.5}$$

In this case, it should be: -

$$a_0(z) = b_0(z)q_0(z) + r_0(z) \tag{A.6}$$

Appendix B

Pseudo-Assemble Code for Row and Column Processor

Integrated processes and steps of Start, Normal and End process of the Vertical Processor when Horizontal Processor is at Start process

Computation sequences at the beginning of the 3^{rd} column of the vertical processing with $N = 8$

Vertical Processor	Horizontal Processor	
Start process - Computing Y_{12} and Y_{02}	Start process – Computing Z_{X1} and Z_{X0}	
step0:- transfer X_{02}, X_{12}, X_{22} to treg0, treg1, treg2		
step1:- acc_a.1.1 = X_{22}+ X_{02}	step 0:- acc_b.1.1 = $Y_{11} - Y_{10}/2$	
step2:- acc_a.2.1 = ½ + X_{02} acc_a.1.1 = X_{12} - acc_a.1.1 /2 = $\mathbf{Y_{12}}$	step 1:- acc_b.2.1 = ½ + Y_{10}	step 0:- acc_b.1.2 = $Y_{01} - Y_{00}/2$
step3:- acc_a.2.1 = acc_a.2.1 + acc_a.1.1 /2 = $\mathbf{Y_{02}}$ mov acc_a1.1, MEM1_0x1	step 2:- acc_b.1.1 = acc_b.1.1 – $\mathbf{Y_{12}}/2 = Z_{11}$	step 1:- acc_b.2.2 = ½ + Y_{00}
Normal process - Computing Y_{32} and Y_{22}		
step0:- mov acc_a2.1, MEM1_0x0 % transfer X_{42}, X_{32} to treg0, treg1 %	step 3:- acc_b.2.1 = acc_b.2.1 + acc_b.1.1/2 = Z_{10}	step 2:- acc_b.1.2 = acc_b.1.2 – $\mathbf{Y_{02}}/2 = Z_{01}$
step1:- acc_a.1.2 = X_{22}+ X_{42} acc_a.2.2 = ½ + X_{22}	step 0:- acc_b.1.1 = $Y_{31} - Y_{30}/2$	step 3:- acc_b.2.2 = acc_b.2.2 + acc_b.1.2/2 = Z_{00}
step2:- acc_a.1.2 = X_{32} - acc_a.1.2 /2 = $\mathbf{Y_{32}}$ acc_a.2.2 = acc_a.2.2 + acc_a.1.1 /4 *% ** assuming Y_{12} is in acc_a1.1*	step 1:- acc_b.2.1 = ½ + Y_{30}	step 0:- acc_b.1.2 = $Y_{21} - Y_{20}/2$
step3:- acc_a.2.2 = acc_a.2.2 + acc_a.1.2 /4 = $\mathbf{Y_{22}}$ mov acc_a1.2, MEM1_0x3	step 2:- acc_b.1.1 = acc_b.1.1 – $\mathbf{Y_{32}}/2 = Z_{31}$	step 1:- acc_b.2.2 = ½ + Y_{20}
- Computing Y_{52} and Y_{42}		
step4:- transfer X_{52}, X_{62} to treg1, treg2 % for next set of computation mov acc_a2.2, MEM1_0x2	step 3:- acc_b.2.1 = acc_b.2.1 + acc_b.1.1/2 = Z_{30}	step 2:- acc_b.1.2 = acc_b.1.2 – $\mathbf{Y_{22}}/2 = Z_{21}$
step5:- acc_a.1.1 = X_{42}+ X_{62} acc_a.2.1 = ½ + X_{42}	step 0:- acc_b.1.1 = $Y_{51} - Y_{50}/2$	step 3:- acc_b.2.2 = acc_b.2.2 + acc_b.1.2/2 = Z_{20}

step6:- acc_a.1.1 = X_{52} - acc_a.1.1 /2 = $\mathbf{Y_{52}}$ acc_a.2.1 = acc_a.2.1 + acc_a.1.2 /4 *% ** assuming Y_{32} is in acc_a1.2*	step 1:- acc_b.2.1 = ½ + Y_{50}	step 0:- acc_b.1.2 = $Y_{41} - Y_{40}/2$
step7:- acc_a.2.1 = acc_a.2.1 + acc_a.1.1 /4 = $\mathbf{Y_{42}}$ mov acc_a1.1, MEM1_0x5	step 2:- acc_b.1.1 = acc_b.1.1 – $\mathbf{Y_{52}}/2 = Z_{51}$	step 1:- acc_b.2.2 = ½ + Y_{40}
Ending process - Computing Y_{72} and Y_{62}		
step0:- transfer X_{72} to treg1 % for next set of computation mov acc_a2.2, MEM1_0x4 % from previous process	step 3:- acc_b.2.1 = acc_b.2.1 + acc_b.1.1/2 = Z_{50}	step 2:- acc_b.1.2 = acc_b.1.2 – $\mathbf{Y_{42}}/2 = Z_{41}$
step1:- acc_a.2.2 = ½ + X_{62}	step 0:- acc_b.1.1 = $Y_{71} - Y_{70}/2$	step 3:- acc_b.2.2 = acc_b.2.2 + acc_b.1.2/2 = Z_{40}
step2:- acc_a.1.2 = X_{72} - X_{62} = $\mathbf{Y_{72}}$ acc_a.2.2 = acc_a.2.2 + acc_a.1.1 /4 *% ** assuming Y_{52} is in acc_a1.1*	step 1:- acc_b.2.1 = ½ + Y_{70}	step 0:- acc_b.1.2 = $Y_{61} - Y_{60}/2$
step3:- acc_a.2.2 = acc_a.2.2 + acc_a.1.2 /4 = $\mathbf{Y_{62}}$ mov acc_a1.2, MEM1_0x7	step 2:- acc_b.1.1 = acc_b.1.1 – $\mathbf{Y_{72}}/2 = Z_{71}$	step 1:- acc_b.2.2 = ½ + Y_{60}
step4:- mov acc_a2.2, MEM1_0x6	step 3:- acc_b.2.1 = acc_b.2.1 + acc_b.1.1/2 = Z_{70}	step 2:- acc_b.1.2 = acc_b.1.2 – $\mathbf{Y_{62}}/2 = Z_{61}$
		step 3:- acc_b.2.2 = acc_b.2.2 + acc_b.1.2/2 = Z_{60}

Table B.1: *Pseudo-Assemble code of Start, Normal and End process of the Vertical Processor when Horizontal Processor is at Start process*

Integrated processes and steps of Start, Normal and End process of the Vertical Processor when Horizontal Processor is at Normal process

Computation sequences at the beginning of the 5^{th} column of the vertical processing with $N = 8$

Vertical Processor	**Horizontal Processor**	
Start process-Computing Y_{14} and Y_{04}	Normal process - Computing Z_{X3} and Z_{X2}	
step0:- transfer X_{04}, X_{14}, X_{24} to treg0, treg1, treg2		
step1:- acc_a.1.1 = X_{24}+ X_{04}	step 0:- acc_b.2.1 = ½ + Y_{12} acc_b.1.1 = Y_{13} – Y_{12}/2	
step2:- acc_a.2.1 = ½ + X_{04} acc_a.1.1 = X_{14} - acc_a.1.1 /2 = $\mathbf{Y_{14}}$	step 1:- acc_b.2.1 = acc_b.2.1 + Z_{11}/4	step 0:- acc_b.2.2 = ½ + Y_{02} acc_b.1.2 = Y_{03} – Y_{02}/2
step3:- acc_a.2.1 = acc_a.2.1 + acc_a.1.1 /2 = $\mathbf{Y_{04}}$ mov acc_a1.1, MEM1_0x1	step 2:- acc_b.1.1 = acc_b.1.1 – $\mathbf{Y_{14}}$/2 = Z_{13}	step 1:- acc_b.2.2 = acc_b.2.2 + Z_{01}/4
Normal process - Computing Y_{34} and Y_{24}		
step0:- mov acc_a2.1, MEM1_0x0 transfer X_{44}, X_{34} to treg0, treg1	step 3:- acc_b.2.1 = acc_b.2.1 + acc_b.1.1/4 = Z_{12}	step 2:- acc_b.1.2 = acc_b.1.2 – $\mathbf{Y_{04}}$/2 = Z_{03}
step1:- acc_a.1.2 = X_{24}+ X_{44} acc_a.2.2 = ½ + X_{24}	step 0:- acc_b.2.1 = ½ + Y_{32} acc_b.1.1 = Y_{33} – Y_{32}/2	step 3:- acc_b.2.2 = acc_b.2.2 + acc_b.1.2/4 = Z_{02}
step2:- acc_a.1.2 = X_{34} - acc_a.1.2 /2 = $\mathbf{Y_{34}}$ acc_a.2.2 = acc_a.2.2 + acc_a.1.1 /4 *% ** assuming Y_{14} is in acc_a1.1*	step 1:- acc_b.2.1 = acc_b.2.1 + Z_{31}/4	step 0:- acc_b.2.2 = ½ + Y_{22} acc_b.1.2 = Y_{23} – Y_{22}/2
step3:- acc_a.2.2 = acc_a.2.2 + acc_a.1.2 /4 = $\mathbf{Y_{24}}$ mov acc_a1.2, MEM1_0x3 %	step 2:- acc_b.1.1 = acc_b.1.1 – $\mathbf{Y_{34}}$/2 = Z_{33}	step 1:- acc_b.2.2 = acc_b.2.2 + Z_{21}/4
- Computing Y_{54} and Y_{64}		
step4:- transfer X_{54}, X_{64} to treg1, treg2 % for next set of computation mov acc_a2.2, MEM1_0x2	step 3:- acc_b.2.1 = acc_b.2.1 + acc_b.1.1/4 = Z_{32}	step 2:- acc_b.1.2 = acc_b.1.2 – $\mathbf{Y_{24}}$/2 = Z_{23}
step5:- acc_a.1.1 = X_{44}+ X_{64} acc_a.2.1 = ½ + X_{44}	step 0:- acc_b.2.1 = ½ + Y_{52} acc_b.1.1 = Y_{53} – Y_{52}/2	step 3:- acc_b.2.2 = acc_b.2.2 + acc_b.1.2/4 = Z_{22}
step6:- acc_a.1.1 = X_{54} - acc_a.1.1 /2 = $\mathbf{Y_{54}}$ acc_a.2.1 = acc_a.2.1 + acc_a.1.2 /4 *% ** assuming Y_{34} is in acc_a1.2*	step 1:- acc_b.2.1 = acc_b.2.1 + Z_{51}/4	step 0:- acc_b.2.2 = ½ + Y_{42} acc_b.1.2 = Y_{43} – Y_{42}/2
step7:- acc_a.2.1 = acc_a.2.1 + acc_a.1.1 /4 = $\mathbf{Y_{44}}$ mov acc_a1.1, MEM1_0x5	step 2:- acc_b.1.1 = acc_b.1.1 – $\mathbf{Y_{54}}$/2 = Z_{53}	step 1:- acc_b.2.2 = acc_b.2.2 + Z_{41}/4
Ending process - Computing Y_{74} and Y_{64}		
step0:- transfer X_{74} to treg1 % for next set of computation mov acc_a2.2, MEM1_0x4 % from previous process	step 3:- acc_b.2.1 = acc_b.2.1 + acc_b.1.1/4 = Z_{52}	step 2:- acc_b.1.2 = acc_b.1.2 – $\mathbf{Y_{44}}$/2 = Z_{43}
step1:- acc_a.2.2 = ½ + X_{64}	step 0:- acc_b.2.1 = ½ + Y_{72} acc_b.1.1 = Y_{73} – Y_{72}/2	step 3:- acc_b.2.2 = acc_b.2.2 + acc_b.1.2/4 = Z_{42}
step2:- acc_a.1.2 = X_{74} - X_{64} = $\mathbf{Y_{74}}$ acc_a.2.2 = acc_a.2.2 + acc_a.1.1 /4 *% ** assuming Y_{54} is in acc_a1.1*	step 1:- acc_b.2.1 = acc_b.2.1 + Z_{71}/4	step 0:- acc_b.2.2 = ½ + Y_{62} acc_b.1.2 = Y_{63} – Y_{62}/2

step3:- acc_a.2.2 = acc_a.2.2 + acc_a.1.2 /4 = Y_{64} mov acc_a1.2, MEM1_0x7	step 2:- acc_b.1.1 = acc_b.1.1 – $Y_{74}/2 = Z_{73}$	step 1:- acc_b.2.2 = acc_b.2.2 + $Z_{61}/4$
step4:- mov acc_a2.2, MEM1_0x6	step 3:- acc_b.2.1 = acc_b.2.1 + acc_b.1.1/4 = Z_{72}	step 2:- acc_b.1.2 = acc_b.1.2 – $Y_{64}/2 = Z_{63}$
		step 3:- acc_b.2.2 = acc_b.2.2 + acc_b.1.2/4 = Z_{62}

Table B.2: *Pseudo-Assemble code of Start, Normal and End process of the Vertical Processor when Horizontal Processor is at Normal process*

Integrated processes and steps of Start, Normal and End process of the Vertical Processor when Horizontal Processor is at End process

Computation sequences at the beginning of the last (7^{th}) column of the vertical processing with $N = 8$

Vertical Processor	Horizontal Processor	
Start process-Computing Y_{17} and Y_{07}	Ending process - Computing Z_{X7} and Z_{X6}	
step0:- transfer X_{07}, X_{17}, X_{27} to treg0, treg1, treg2		
step1:- acc_a.1.1 = X_{27}+ X_{07}		
step2:- acc_a.2.1 = 1/2 + X_{07} acc_a.1.1 = X_{17} - acc_a.1.1 /2 = $\mathbf{Y_{17}}$	step 0: - acc_b.2.1 = ½ +Y_{16}	
step3:- acc_a.2.1 = acc_a.2.1 + acc_a.1.1 /2 = $\mathbf{Y_{07}}$ mov acc_a1.1, MEM1_0x1	step 1: - acc_b.1.1 = $\mathbf{Y_{17}}$ – Y_{16} = Z_{17} acc_b.2.1 = acc_b.2.1 + Z_{15}/4	step 0: - acc_b.2.2 = ½ +Y_{06}
Normal process - Computing Y_{37} and Y_{27}		
step0:- mov acc_a2.1, MEM1_0x0 transfer X_{47}, X_{37} to treg0, treg1	step 2: - acc_b.2.1 = acc_b.2.1 + acc_b.1.1/4 = Z_{16}	step 1: - acc_b.1.2 = $\mathbf{Y_{07}}$ – Y_{06} = Z_{07} acc_b.2.2 = acc_b.2.2 + Z_{05}/4
step1:- acc_a.1.2 = X_{27}+ X_{47} acc_a.2.2 = 1/2 + X_{27}	step 3: -	step 2: - acc_b.2.2 = acc_b.2.2 + acc_b.1.2/4 = Z_{06}
step2:- acc_a.1.2 = X_{37} - acc_a.1.2 /2 = $\mathbf{Y_{37}}$ acc_a.2.2 = acc_a.2.2 + acc_a.1.1 /4 *% ** assuming Y_{17} is in acc_a1.1*	step 0: - acc_b.2.1 = ½ +Y_{36}	step 3: -
step3:- acc_a.2.2 = acc_a.2.2 + acc_a.1.2 /4 = $\mathbf{Y_{27}}$ mov acc_a1.2, MEM1_0x3	step 1: - acc_b.1.1 = $\mathbf{Y_{37}}$ – Y_{36} = Z_{37} acc_b.2.1 = acc_b.2.1 + Z_{35}/4	step 0: - acc_b.2.2 = ½ +Y_{26}
- Computing Y_{57} and Y_{47}		
step4:- transfer X_{57}, X_{67} to treg1, treg2 % for next set of computation mov acc_a2.2, MEM1_0x2	step 2: - acc_b.2.1 = acc_b.2.1 + acc_b.1.1/4 = Z_{36}	step 1: - acc_b.1.2 = $\mathbf{Y_{27}}$ – Y_{26} = Z_{27} acc_b.2.2 = acc_b.2.2 + Z_{25}/4
step5:- acc_a.1.1 = X_{47}+ X_{67} acc_a.2.1 = 1/2 + X_{47}	step 3: -	step 2: - acc_b.2.2 = acc_b.2.2 + acc_b.1.2/4 = Z_{26}
step6:- acc_a.1.1 = X_{57} - acc_a.1.1 /2 = $\mathbf{Y_{57}}$ acc_a.2.1 = acc_a.2.1 + acc_a.1.2 /4 *% ** assuming Y_{37} is in acc_a1.2*	step 0: - acc_b.2.1 = ½ +Y_{56}	step 3: -
step7:- acc_a.2.1 = acc_a.2.1 + acc_a.1.1 /4 = $\mathbf{Y_{47}}$ mov acc_a1.1, MEM1_0x5	step 1: - acc_b.1.1 = $\mathbf{Y_{57}}$ – Y_{56} = Z_{57} acc_b.2.1 = acc_b.2.1 + Z_{55}/4	step 0: - acc_b.2.2 = ½ +Y_{46}
Ending process - Computing Y_{77} and Y_{67}		
step0:- transfer X_{77} to treg1 % for next set of computation mov acc_a2.2, MEM1_0x4 % from previous process	step 2: - acc_b.2.1 = acc_b.2.1 + acc_b.1.1/4 = Z_{56}	step 1: - acc_b.1.2 = $\mathbf{Y_{47}}$ – Y_{46} = Z_{47} acc_b.2.2 = acc_b.2.2 + Z_{45}/4
step1:- acc_a.2.2 = 1/2 + X_{67}	step 3: -	step 2: - acc_b.2.2 = acc_b.2.2 + acc_b.1.2/4 = Z_{46}
step2:- acc_a.1.2 = X_{77} - X_{67} = $\mathbf{Y_{77}}$ acc_a.2.2 = acc_a.2.2 + acc_a.1.1 /4 *% ** assuming Y_{57} is in acc_a1.1*	step 0: - acc_b.2.1 = ½ +Y_{76}	step 3: -

step3:- acc_a.2.2 = acc_a.2.2 + acc_a.1.2 /4 = Y_{67} mov acc_a1.2, MEM1_0x7	step 1: - acc_b.1.1 = $Y_{77} - Y_{76} = Z_{77}$ acc_b.2.1 = acc_b.2.1 + Z_{75}/4	step 0: - acc_b.2.2 = ½ + Y_{66}
step4:- mov acc_a2.2, MEM1_0x6	step 2: - acc_b.2.1 = acc_b.2.1 + acc_b.1.1/4 = Z_{76}	step 1: - acc_b.1.2 = $Y_{67} - Y_{66} = Z_{67}$ acc_b.2.2 = acc_b.2.2 + Z_{65}/4
	step 3: -	step 2: - acc_b.2.2 = acc_b.2.2 + acc_b.1.2/4 = Z_{66}

Table B.3: *Pseudo-Assemble code of Start, Normal and End process of the Vertical Processor when Horizontal Processor is at End process*

Appendix C

FSM Mapping Process

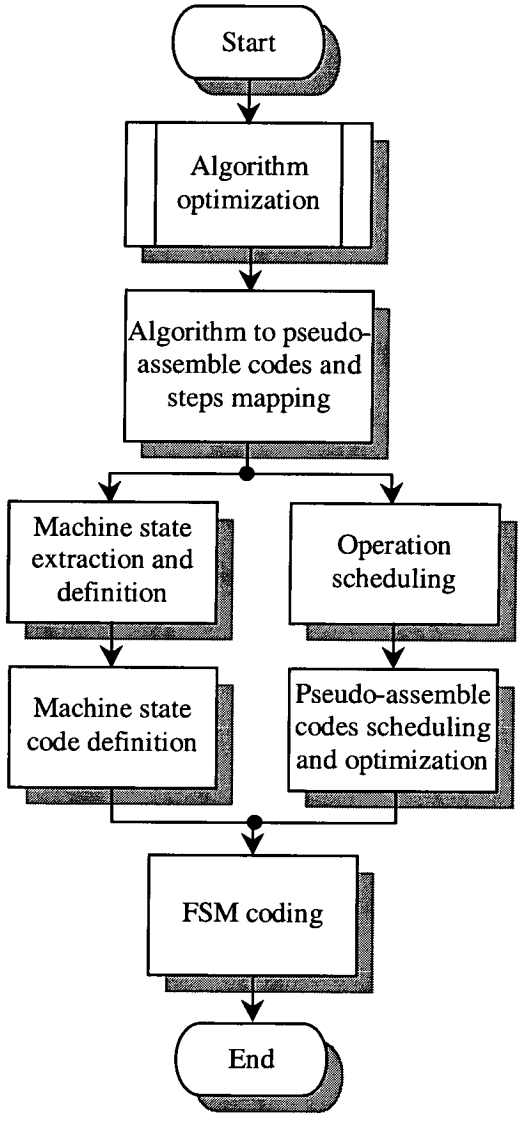


Figure C.1: *FSM mapping flowchart*

Appendix D

Test Images

Figure D.1: *Test image 'Lenna'*

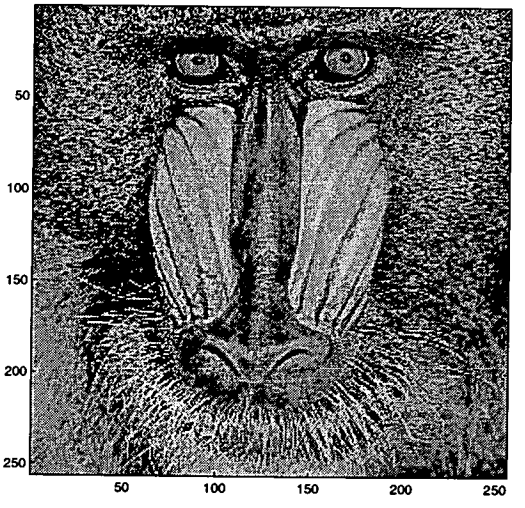


Figure D.2: *Test image 'Baboon'*

Figure D.3: *Test image 'Pepper'*

References

[1] ISO/IEC, "ISO/IEC15444-1, Information technology - JPEG 2000 image coding system - Part 1: Core coding system," http://www.jpeg.org/JPEG2000.html, 2000.

[2] K. Andra, C. Chakrabarti and T. Acharya, "A VLSI architecture for lifting-based forward and inverse wavelet transform," in *IEEE transactions on signal processing*, vol. 50, no. 3, pp. 966–977, April 2002.

[3] C-J. Lian, K-F. Chen, H-H.Chen and L-G. Chen, "Analysis and architecture design of lifting based DWT and EBCOT for JPEG2000," in *Proceedings of Technical Papers of 2001 International Symposium on VLSI Technology, Systems and Applications*, pp. 180–183, 2001.

[4] C. Yu and S.-J. Chen, "VLSI implementation of 2D discrete wavelet transform for real-time video signal processing," in *IEEE Transactions on Consumer Electronics*, vol. 43, no. 4, pp. 1270–1279, November 1997.

[5] W. Sweldens, "The lifting scheme: A new philosophy in biorthogonal wavelet constructions," in *Proceedings of SPIE*, pp. 68–79, 1995.

[6] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," in *Journal of Fourier Analysis and Applications*, vol. 4, pp. 247–269, 1998.

[7] C-J. Lian, K-F. Chen, H-H. Chen and L-G. Chen, "Lifting based discrete wavelet transform architecture for JPEG2000," in *Proceedings of the 2001 IEEE International Symposium on Circuits and Systems, ISCAS 2001*, vol. 2, pp. 445–448, 2001.

[8] D. Le Gall and A. Tabatabai, "Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 761–765, 1988.

[9] M. Antonini, M. Barlaud, P. Mathieu and I. Daubechies, "Image coding using wavelet transform," in *IEEE Transactions on Image Processing*, vol. 1, issue no.2, pp. 205–220, April 1992.

[10] S. Masupe and T. Arslan, "Low power DCT implementation approach for VLSI DSP processors," in *IEEE International Symposium on Circuits & Systems*, vol. 1, pp. 149–152, July 1999.

[11] International Technology Roadmap for Semiconductors, "ITRS, 2003Edition," http://public.itrs.net, 2003.

[12] Mani B. Srivastava, Anantha P. Chandrakasan and Robert W. Brodersen, "Predictive system shutdown and other architectural techniques for energy efficient programmable computation," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 4, issue no.1, pp. 42–55, March 1996.

[13] Chi-Hong Hwang and Allen C.-H. Wu, "A predictive system shutdown method for energy saving of event-driven computation," in *Proceedings of the International Conference on Computer Aided Design*, pp. 28–32, November 1997.

[14] H. H. Hellmich, A. T. Erdrogan and T. Arslan, "Re-usable low power DSP IP embedded in an ARM based SOC architecture," in *IEE Coloquim on Intellectual Property, Edinburgh, U.K.*, pp. 10/1–10/5, July 2000.

[15] Sonya Gary, Pete Ippolito, Gianfranco Gerosa, Carl Dietz, Jim Eno and Hector Sanchez, "PowerPC 603, a microprocessor for portable computers," in *IEEE Design & Test of Computers*, vol. 11, issue no.4, pp. 14–23, 1994.

[16] Klaus Herrmann, Klaus Gaedke, Hartwig Jeschke and Peter Pirsch, "A monolithic low power video signal processor for multimedia applications," in *IEEE International Conference on Consumer Electronics, Digest of Technical Papers*, pp. 176–177, June 1996.

[17] D. Garret and M. Stan", "Low power parallel spread-spectrum correlator," in *IEE Proceedings of Circuits Devices and System*, vol. 146, issue no.4, pp. 191–196, August 1999.

[18] T. Shikata, S. Koudou, M. Nose, Y. Kuniyasu, M. Naitoh and H. Suzuki, "A single-chip low power DSP/RISC CPU with 0.25μm CMOS technology," in *IEEE 1998 Custom Integrated Circuits Conference*, pp. 123–126, May 1998.

[19] Jeffrey T. Ludwig, S. Hamid Namab and Anantha P. Chandrakasan, "Low-power digital filtering using approximate processing," in *IEEE Journal of Solid-State Circuits*, vol. 31, issue no.3, pp. 395–400, March 1996.

[20] S. Hong and W. Stark, "Low power VLSI DS/FH hybrid CDMA spread spectrum IF/baseband transceiver design," in *48th IEEE Vehicular Technology Conference*, vol. 2, pp. 1573–1577, May 1998.

[21] D. Ramanathan and R. Gupta, "System level online power management algorithms," in *Proceedings of Design, Automation & Test in Europe Conference & Exhibition*, pp. 606–611, March 2000.

[22] G. A. Paleologo, L. Benini, A. Boglido and G. De Micheli, "Policy optimization for dynamic power management," in *Proceedings of the Design Automation Conference*, pp. 182–187, June 1998.

[23] Q. Qiu and M. Pedram, "Dynamic power management based on continuous-time markov decision processes," in *Proceedings of the Design Automation Conference*, pp. 555–561, June 1999.

[24] Q. Qiu, Q. Wu and M. Pedram, "Dynamic power management of complex systems using generalised stochastic petri nets," in *Proceedings of the Design Automation Conference*, pp. 352–356, June 2000.

[25] Q. Qiu, Q. Wu and M. Pedram, "Stochastic modeling of a power-managed system: Construction and optimization," in *Proceedings of International Symposium on Low Power Electronics and Design*, pp. 194–199, June 1999.

[26] Yung-Hsian Lu, Luca Benini and Giovanni De Micheli, "Low-power task scheduling for multiple devices," in *International Workshop on Hardware/Software Codesign*, pp. 39–43, 2000.

[27] L. Benini, G. De Micheli, E. Macii, G. Odasso and M.Poncino, "Kernel-based power optimization of RTL components: Exact & approximate extraction algorithms," in *Proceedings of Design Automation Conference*, pp. 247–252, June 1999.

[28] V. Gutnik and A. P. Chandrakasan, "Embedded power supply for low-power DSP," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 5, issue no.4, pp. 425–435, December 1997.

[29] James Goodman, Anantha P. Chandrakasan and Abram P. Dancy, "Design and implementation of a scalable encryption processor with embedded variable DC/DC converter," in *Proceedings of the Design Automation Conference*, pp. 855–860, June 1999.

[30] T. Kuroda, K. Suzuki, S. Mita, T.Fujita, F. Yamane, F. Sano, A. Chiba, Y. Watanabe, K. Matsuda, T. Maeda, T. Sakurai and T. Furuyama, "Variable supply-voltage scheme for low-power high-speed CMOS digital design," in *IEEE Journal of Solid-State Circuits*, vol. 33, issue no.3, pp. 452–462, March 1998.

[31] Inki Hong, Darko Kirouski, Gang Qu, Miodrag Potkonjak and Mani B. Srivastava, "Power optimisation of variable voltage core-based systems," in *Proceedings of Design Automation Conference*, pp. 176–181, June 1998.

[32] T. Kuroda, "Low power CMOS digital design for multimedia processors," in 6^{th} *International Conference on VLSI & CAD*, pp. 359–367, 1999.

[33] T. Kuroda and M. Hamda, "Low-power CMOS digital design with dual-embedded adaptive power supplies," in *IEEE Journal of Solid-State Circuits*, vol. 35, issue no.4, pp. 652–655, April 2000.

[34] K. Usami, M. Igarushi, T. Ishikawa, Masahiro Kanazara, M. Takahashi, M. Hamada, H. Arakida, T. Terazawa and T. Kuroda, "Design methodology of ultra low-power MPEG4 codec core exploiting voltage scaling techniques," in *Proceedings of Design Automation Conference*, pp. 483–488, 1998.

[35] K.Usami, M. Igarashi, F. Minami, T. Ishikawa, M. Kanazawa, M. Ichida and K. Nogami, "Automated low-power technique exploiting multiple supply voltages applied to a media processor," in *IEEE Journal of Solid-State Circuits*, vol. 33, issue no.3, pp. 463–472, March 1998.

[36] S. Lee and T. Sakurai, "Run-time voltage hopping for low-power real-time systems," in *Proceedings of the Design Automation Conference*, pp. 806–809, June 2000.

[37] Gang Qu, Darko Kirovski, Miodrag Potkonjak and Mani B. Srivastava, "Energy minimisation of system pipelines using multiple voltages," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 362–365, July 1999.

[38] J.-M. Chang and M. Pedram, "Energy minimisation using multiple supply voltages," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 5, issue no. 4, pp. 436–443, December 1997.

[39] Haris Lekatsas, Jórg Henkel and Wayne Wolf, "Code compression for low power embedded system design," in *Proceedings of Design Automation Conference*, pp. 294–299, 2000.

[40] Ching-Long Su, Chi-Ying Tsui and Alvin M. Despain, "Low power architecture design and compilation techniques for high-performance processors," in *Digest of Papers of Compcon Spring*, pp. 489–498, March 1994.

[41] Vivek Tiwari, Sharad Malik and Andrew Wolfe, "Compilation techniques for low energy: An overview," in *Digest of Technical Paper of IEEE Symposium on Low Power Electronics*, pp. 38–39, October 1994.

[42] Gebotys, R. Gebotys and S. Winotunge, "Power minimisation derived from architectural-usage of VLIW processors," in *Proceedings of Design Automation Conference*, pp. 308–311, 2000.

[43] Mike Tien-Chien Lee and Vivek Tiwari, "A memory allocation technique for low-energy embedded DSP software," in *IEEE Symposium on Low Power Electronic*, pp. 24–25, October 1995.

[44] H. Mehta, R. Micheal Owens, M. J. Irwin, R. Chen and D. Ghosh, "Techniques for low energy software," in *Proceedings of International Symposium on Low Power Electronics and Design*, pp. 72–75, 1997.

[45] Mike Tien-Chien Lee, Vivek Tiwari, Sharad Malik and Masahiro Fujita, "Power analysis and low-power scheduling techniques for embedded DSP software," in *Proceedings of the 8th International Symposium on System Synthesis*, pp. 110–115, October 1995.

[46] J. H. Snyder, J. B. Mckie and B. N. Locanthi, "Low-power software for low-power people," in *IEEE Symposium on Low Power Electronics*, pp. 33–35, October 1994.

[47] M. S. Bright and T. Arslan, "A genetic framework for the high-level optimisation of low power VLSI DSP systems," in *IEE Electronics Letters*, vol. 32, issue no.13, pp. 1150–1151, June 1996.

[48] M. S. Bright and T. Arslan, "Transformational-based synthesis of VLSI based DSP systems for low power using a genetic algorithm," in *IEEE International Symposium on Circuits & Systems*, vol. 6, pp. 45–48, June 1998.

[49] T. H. Meng, "A wireless portable video-on-demand system," in *Proceedings of Eleventh International Conference on VLSI Design*, pp. 4–9, 1997.

[50] T. Arslan and A. T. Erdogan, "Data block processing for low power implementation of direct form FIR filters on single multiplier CMOS DSPs," in *IEEE International Symposium on Circuits & Systems*, vol. 5, pp. 441–444, June 1998.

[51] A. T. Erdogan and T. Arslan, "A hybrid segmentation and block processing algorithm for low power implementation of digital filters," in *IEEE International Symposium on Circuits & Systems*, vol. 5, pp. 349–352, May 2000.

[52] A. T. Erdogan and T. Arslan, "A coefficient segmentation algorithm for low power implementation of FIR filters," in *IEEE International Symposium on Circuits & Systems*, vol. 3, pp. 359–362, July 1999.

[53] D. B. Lidsky and J. M. Rabaey, "Low power design of memory intensive functions case study: Vector quantization," in *Workshop on VLSI Signal Processing*, pp. 378–387, 1994.

[54] K. Masselos, P. Merakos, T. Stouraitis and C. E. Goutis, "A novel algorithm for low power image and video coding," in *IEEE transactions on circuits and systems for video technology*, vol. 8, issue no.3, pp. 258–263, June 1998.

[55] Liang-Gee Chan, Juing-Ying Jin and Hao-Chieh Chang, "Design and implementation of low-power DCT chip for portable multimedia terminals," in *IEEE Workshop on Signal Processing Systems*, pp. 85–93, October 1998.

[56] G. P. Fettweis and L. Thiele, "Algebraic recurrence transformations for massive parallelism," in *Workshop on VLSI Signal Processing*, pp. 332–341, October 1992.

[57] K. K. Parhi, "Algorithm transformation techniques for concurrent processors," in *Proceedings of the IEEE*, vol. 77, issue no.12, pp. 1879–1895, December 1989.

[58] M. Potkonjak and J. Rabaey, "Optimizing resource uitilization using transformations," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, issue no.3, pp. 277–292, March 1994.

[59] M. Potkonjak and J. M. Rabaey, "Power minimization in DSP application specific system using algorithm selection," in *International Conference on Acoustic Speech & Signal Processing*, vol. 4, pp. 2639–2642, May 1995.

[60] S. Okada, S. Okada, Y. Matsuda, T. Yamada and A. Kobayashi, "System on a chip for digital still camera," in *IEEE Transactions on Consumer Electronics*, vol. 45, no. 3, pp. 584–590, August 1999.

[61] C.-T. Hsieh and M. Pedram, "Architectural power optimisation by bus splitting," in *Proceedings of Design, Automation & Test in Europe Conference & Exhibition*, pp. 612–616, 2000.

[62] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power I/O," in *IEEE Transactions on Very Large Scale Integration (VLSI) System*, vol. 3, issue no.1, pp. 48–58, March 1995.

[63] Youngsoo Shin, Soo-Ik Chae and Kiyoung Choi, "Partial bus-invert coding for power optimisation of system level bus," in *Proceedings of International Symposium on Low Power Electronics and Design*, pp. 127–129, August 1998.

[64] Y. Shin and K. Choi, "Narrow bus encoding for low power systems," in *Proceedings of Asia & South Pacific Design Automation Conference*, pp. 217–220, 2000.

[65] M. R. Stan and W. P. Burleson, "Coding a terminated bus for low power," in *Proceedings of Greats Lakes Symposium on VLSI*, pp. 70–73, March 1995.

[66] P. P. Sotiriadis and A. P. Chandrakasan, "Low power bus coding techniques considering inter-wire capacitances," in *IEEE 2000 Custom Integrated Circuits Conference*, pp. 507–510, 2000.

[67] L. Benini, A. Macii, E. Macii, M. Poncino and Scarsi, "Synthesis of low-overhead interfaces for power-efficient communication over wide buses," in *Proceedings of Design Automation Conference*, pp. 128–123, 1999.

[68] Sumant Ramprasad, Naresh R. Shanbhag and Ibrahim N. Hajj, "Signal coding for low power: Fundamental circuits and practical realisations," in *IEEE Transactions on Circuits and Systems- II: Analog & Digital Signal Processing*, vol. 46, issue no.7, pp. 923–929, July 1999.

[69] P. R. Panda and N. D. Dutt, "Reducing address bus transitions for low power memory mapping," in *Proceedings of European Design & Test Conference*, pp. 63–68, March 1996.

[70] P. R. Panda and N. D. Dutt, "Low power memory mapping through reducing address bus activity," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, issue no.3, pp. 309–320, September 1999.

[71] L. Benini, G. De Michel, E. Macii, M. Poncino and S. Quer, "System-level power optimisation of special purpose applications: The Beach solution," in *Proceedings of International Symposium on Low Power Electronics and Design*, pp. 24–29, August 1997.

[72] H. Lekatsas and J. Henkel, "ETAM++: extended transition activity measure for low power address bus designs," in *Proceedings of the International Conference on VLSI Design*, pp. 113–120, 2002.

[73] M. Mamidipaka, D. Hirschberg and N. Dutt, "Low power address encoding using self-organizing list," in *International Symposium on Low Power Electronics and Design*, pp. 188–193, 2001.

[74] Y. Aghaghiri, F. Fallah and M. Pedram, "ALBORZ:address level bus power optimization," in *Proceedings of the International Symposium on Quality Electronic Design*, pp. 470–475, 2002.

[75] Naehynck Chang, Kwanho Kim and Jimsung Cho, "Bus encoding for low-power high-performance memory systems," in *Proceedings of Design Automation Conference*, pp. 800–805, 2000.

[76] You-Sung Chang, Bong-Il Park and Chong-Min Kyang, "Conforming inverted data store for low power memory," in *Proceedings of International Symposiums on Low Power Electronics and Design*, pp. 91–93, 1999.

[77] S. L. Cournei and D. E. Thomas, "An environment for exploring low power memory configuration in system level design," in *International Conference on Computer Design*, pp. 348–353, 1999.

[78] B. Kapoor, "Low power memory architectures for video applications," in *Proceedings of the* 8^{th} *Great Lakes Symposiums on VLSI*, pp. 2–7, February 1998.

[79] E. Brockmeyer, L. Nachtergaele, F. V. M. Catthoor, J. Bormans and H. J. De Man, "Low power memory storage and transfer organisation for the MPEG-4 full pel motion estimation on a multimedia processor," in *IEEE Transactions on Multimedia*, vol. 1, issue no.2, pp. 202–215, June 1999.

[80] L. S. Nielen and J. Sparso, "Designing asynchronous circuits for low power: An IFIR filter bank for a digital hearing aid," in *Proceedings of the IEEE*, vol. 87, issue no.2, pp. 268–281, February 1999.

[81] S.-J. Jou and I.-Y. Chung, "Low power globally asynchronous locally synchronous design using self-timed circuit technology," in *IEEE International Symposium on Circuits And Systems*, vol. 3, pp. 1808–1811, June 1997.

[82] H. Mehta, R. M. Owens, M. J. Irwin, R. Chen and D. Ghosh, "Lowering power consumption in clock by using globally asynchronous locally synchronous design style," in *Proceedings of Design Automation Conference*, pp. 873–878, 1999.

[83] Anantha P. Chandrakasan, Andrew Burstein and Robert W. Brodersen, "A low-power chipset for a portable multimedia I/O terminal," in *IEEE Journal of Solid-State Circuits*, vol. 29, issue no.12, pp. 1413–1428, December 1994.

[84] Kei-Yong Khoo, Chao-Liang Chen and Alan N. Willson, Jr, "Biased two's complement representation for low-power DSP system," in *IEEE Proceedings of* 40^{th} *Midwest Symposium on Circuits & Systems*, vol. 2, pp. 786–789, August 1997.

[85] V. G. Moshnyaga, "A MSB truncation scheme for low-power video processors," in *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 291–294, July 1999.

[86] K. Hasegana, K. Ohara, A. Oka, T. Kamada, Y. Nagaoka, K. Yano, E. Yamauchi, T. Kashiro and T. Nakagawa, "Low-power video encoder/decoder chip set for digital VCRs," in *IEEE Journal of Solid-State Circuits*, vol. 31, issue no. 11, pp. 1780–1787, November 1996.

[87] A. P. Chandrakasan, "Ultra low power digital signal processing," in *Proceedings of the* 9^{th} *International Conference on VLSI Design*, pp. 352–356, June 1996.

[88] W. Namgoong and T. Meng, "Power consumption of parallel spread spectrum correlator architectures," in *International Symposium on Low Power Electronics & Design*, pp. 133–135, August 1998.

[89] T. Xanthopoulos and A. P. Chandrakasan, "A low-power DCT core using adaptive bitwidth and arithmetic activity exploiting signal correlations and quantization," in *IEEE Journal of Solid-State Circuits*, vol. 35, issue no.5, pp. 740–750, May 2000.

[90] R.Amirtharajah, T. Xanthopoulos and A. P. Chandrakasan, "Power scalable processing using distributed arithmetic," in *International Symposium on Low Power Electronics design*, pp. 170–175, 1999.

[91] Jose Monteiro, Srinivas Devadas and Abhijit Ghosh, "Retiming sequential circuit for low power," in *Proceedings of International Conference of CAD*, pp. 398–402, November 1993.

[92] Anantha P. Chandrakasan, Samuel Sheng and Robert W. Brodersen, "Low power CMOS digital design," in *IEEE Journal of Solid-state circuit*, vol. 27, issue no.4, pp. 473–484, December 1992.

[93] A. P. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," in *Proceedings of the IEEE*, vol. 83, issue no.4, pp. 498–523, April 1995.

[94] Neungsoo Park, Jongwoo Bae and Viktor K. Prasanna, "Synthesis of VLSI architectures for tree-structured image coding," in *Proceedings of International Conference on Image Processing*, vol. 2, pp. 999–1006, September 1996.

[95] Huan-Chang Liu, Jonathan Min and Henry Samueli, "A low-power baseband receiver IC for frequency-hopped spread spectrum applications," in *IEEE 1995 Custom Integrated Circuits Conference*, pp. 311–314, May 1995.

[96] Toshiyuki Sakuta, Wai Lee and Poras T. Balsara", "Delay balanced multipliers for low power/low voltage DSP core," in *IEEE Symposium on Low Power Electronics*, pp. 36–37, October 1995.

[97] S. H. Cho, T. Xanthopoulos and A. P. Chandrakasan, "A low power variable length decoder for MPEG-2 based on non-uniform fine grain table partitioning," in *IEEE*

Transactions on Very Large Scale Integration (VLSI) Systems, vol. 7, issue no.2, pp. 249–257, June 1999.

[98] S.-W. Lee and I.-C. Park, "Low-power variable length decoder considering successive codewords," in *IEE Electronics Letters*, vol. 36, issue no.5, pp. 440–442, 2^{nd} March 2000.

[99] K. Yano, T. Yamanaka, T. Nishida, M. Saito, K. Shimohigashi and A. Shimizu, "A 3.8-ns CMOS 16 x 16-b multiplier using complementary pass-transistor logic," in *IEEE Journal of Solid-State Circuits*, vol. 25, issue no.2, pp. 388–395, April 1990.

[100] L. Benini, A. Macii, E. Macii, M. Poncino and R. Scarsi, "F-Gate: A device for glitch power minimization," in *Conference Record of the Thirty-Second Asilomar Conference on Signals, Systems & Computers*, vol. 2, pp. 1047–1051, November 1998.

[101] L. Benini, G. De Micheli, A. Macii, E. Macii, M. Poncino and R. Scarsi, "Glitch power minimization by selective gate freezing," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, issue no.3, pp. 287–298, June 2000.

[102] Hiroshi Kawaguchi, Ko-ichi Nose and Takayasu Sakurai, "A CMOS scheme for 0.5v supply voltage with pico-ampere standby current," in *Digest of Technical Papers of IEEE International Solid-State Circuits Conference*, pp. 192–193, February 1998.

[103] D. Suvakovic and C. Salama, "Two phase non-overlapping clock adiabatic differential cascode voltage switch logic (ADCVSL)," in *Digest of Technical Papers of IEEE International Solid-State Circuits Conference*, pp. 364–365, 2000.

[104] W. Prost, U. Auer, C. Pacha, A. Brennemann, K.F. Goser and F.-J. Tegude, "Novel mobile gates with low-power, relaxed parameter sensitivity, and increased driving capability," in *11th International Conference on Indium Phosphide and Related Materials*, pp. 411–414, 1999.

[105] T. Simon and A. P. Chandrakasan, "An ultra low power adaptive wavelet video encoder with integrated memory," in *IEEE Journal of Solid-state Circuits*, vol. 35, issue no.4, pp. 572–582, April 2000.

[106] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel and F. Baez, "Reducing power in high-performance microprocessors," in *Proceedings of the Design Automation Conference*, pp. 732–737, June 1998.

[107] T. Xanthopoulos and A. P. Chandrakasan, "A low-power IDCT marcocell for MPEG-2 MP@ML exploiting data distribution properties for minimal activity," in *IEEE Journal of Solid-State Circuits*, vol. 34, issue no.5, pp. 693–703, May 1999.

[108] F. Mombers, M. Gumm, D. Stephanie, P. Garino and D. Mlynek, "IMAGE: a low cost, low power video processor for high quality motion estimation in MPEG-2 encoding," in *IEEE Transactions on Consumer Electronics*, vol. 44, issue no.3, pp. 774–783, August 1998.

[109] E. K. Tsern and T. H. Meng, "A low power video-rate pyramid VQ decoder," in *IEEE Journal of Solid-State Circuits*, vol. 31, issue no.11, pp. 1789–1794, November 1996.

[110] V. Tiwari, R.Donnelly, S.Malik and R. Gonzalez, "Dynamic power management for microprocessors: A case study," in 10^{th} *International Conference on VLSI Design*, pp. 185–192, January 1997.

[111] S. M. Rao and S. Nandy, "Power minimisation using control generated clocks,"in *Proceedings of* 37^{th} *Design Automation Conference*, pp. 794–799, 2000.

[112] Y. Taur, D. A. Buchanan, W. Chen, D. J. Frank, K. E. Ismail, S-H. Lo, G. A. Sai-Halasz, R. G. Viswanathan, H-J. C. Wann, S. J. Wind and H-S. Wong, "CMOS scaling into the nanometer regime," in *Proceedings of the IEEE*, vol. 85, issue no.4, pp. 486–504, April 1997.

[113] B. Yu, L. Chang, S. Ahmed, H. Wang, S. Bell, C-Y. Yan, C. Tabery, C. Ho, Qi Xiang, T-J. King, J. Bokor, C. Hu, M-R. Lin and D. Kyser, "FinFET scaling to 10nm gate length," in *Digest of International Electron Devices Meeting 2002*, pp. 251–254, 2002.

[114] L. Chang, Y-K. Choi, J. Kedzierski, N. Lindert, P. Xuan, J. Bokor, C. Hu and T-J. King, "Moore's law lives on," in *IEEE Circuits and Devices Magazine*, vol. 19, no. 1, pp. 35–42, January 2003.

[115] S. H. Tang, L. Chang, N. Lindert, Y-K. Choi, W-C. Lee, X. Huang, V. Subramanian, J. Bokor, T-J. King and C. Hu, "FinFET - A quasi-planar double-gate MOSFET," in *Digest of Technical Papers of 2001 IEEE International Solid-State Circuits Conference*, pp. 118–119,437, 2001.

[116] Q. Chen, K. A. Bowman, E. M. Harrell and J. D. Meindl, "Double jeopardy in the nanoscale court," in *IEEE Circuits and Devices Magazine*, vol. 19, no. 1, pp. 28–34, January 2003.

[117] Ran-Hong Yan, Abbas Ourmazd and Kwing F. Lee, "Scaling the Si MOSFET: From bulk to SOI to bulk," in *IEEE Transactions on Electron Devices*, vol. 39, issue no.7, pp. 1704–1710, July 1992.

[118] I. C. Kizilyalli, M. M. Rambaud, A. Duncan, S. A. Lytle and M. J. Thoma, "Threshold voltage-minimum gate length trade-off in buried channel PMOS devices for scaled supply voltage CMOS technologies," in *IEEE Electron Device Letters*, vol. 16, no. 10, pp. 457–459, October 1995.

[119] Alice Wang, Anantha P. Chandrakasan and Stephen V. Kosonocky, "Optimal supply and threshold scaling for subthreshold CMOS circuits," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI 2002*, pp. 5–9, 2002.

[120] Ricardo Gonzalez, Benjamin M. Gordon and Mark A. Horowitz, "Supply and threshold voltage scaling for low power CMOS," in *IEEE Journal of Solid-State Circuits*, vol. 32, no. 8, pp. 1210–1216, August 1997.

[121] Y. Okumura, M. Shirahata, A. Hachisuka, T. Okudaira, H. Arima and T. Matsukawa, "Source-to-drain nonuniformly doped channel (NUDC) MOSFET structures for high current drivability and threshold voltage controllability," in *IEEE Transactions on Electron Devices*, vol. 39, issue no.11, pp. 2541–2552, June 1992.

[122] Takashi Inukai, Toshiro Hiramoto and Takayasu Sakurai, "Variable threshold voltage CMOS (VTCMOS) in series connected circuits," in *International Symposium on Low Power Electronics and Design 2001*, pp. 201–206, 2001.

[123] M. Lundberg, K. Muhammad, K. Roy and S. K. Wilson, "High-level modelling of switching activity with application to low-power DSP system synthesis," in *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, vol. 4, pp. 1877–1880, March 1999.

[124] C. Brandolese, W. Fornaciari, F. Salice and D. Sciuto, "A instruction-level functionality-based energy estimation model for 32-bits microprocessors," in *Proceeding of Design Automation Conference*, pp. 346–350, 2000.

[125] W. Ye, N. Vijaykrishnan, M. Kandemir and M. J. Irwin, "The design and use of simplepower: A cycle-accurate energy estimation tool," in *Proceeding of Design Automation Conference*, pp. 340–345, 2000.

[126] R. P. Dick, G. Lakshminarayan, A. Raghunathan and N. K. Jha, "Power analysis of embedded operating systems," in *Proceeding of Design Automation Conference*, pp. 312–315, 2000.

[127] J. Kin, C. Lee, W. H. Mangione-Smith and M. Potkonjah, "Power efficient mediaprocessors: Design space exploration," in *Proceedings of Design Automation Conference*, pp. 321–326, 1999.

[128] Won Namgoong, Navin Chaddha and Teresa H. Y. Meng, "Low-power video encoder/decoder using Wavelet/TSVQ with conditional replenishment," in *International Conference on Acoustic, Speech & Signal Processing*, vol. 6, pp. 3240–3243, May 1996.

[129] E. Kuntzel and T. Q. Nguyen, "Lossless image coder with low power implementation," in *Thirty-First Asilomar Conference on Signal, Systems & Computers*, vol. 2, pp. 1460–1464, November 1997.

[130] Guoqing Zhang, Mike Talley, Wael Badawy, Micheal Weeks and Mafay Bayoumi, "A low power prototype for a 3-D discrete wavelet transform processor," in *IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 145–148, June 1999.

[131] F. Marino, "Efficient high-speed/low-power pipelined architecture for the 2-D discrete wavelet transform," in *IEEE Tranactions on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 47, no.12, pp. 1476–1491, December 2000.

[132] B. Das and S. Banerjee, "Low power architecture of running 3-D wavelet transform for medical imaging application," in *Proceedings of the second joint EMBS/BMES Conference*, vol. 2, pp. 1062–1063, October 2002.

[133] P. P. Dang and P. M. Chau, "A high performance, low power VLSI design of discrete wavelet transform for lossless compression in JPEG2000 standard,"in *2001 IEEE International Conference on Consumer Electronics*, pp. 126–127, June 2001.

[134] P. P. Dang and P. M. Chau, "Integer fast wavelet transform and its VLSI implementation for low power applications," in *IEEE Workshop on Signal Proceesing Systems*, pp. 93–98, October 2002.

[135] N. Ahmed, T. Natarjan and K. R. Rao, "Discrete cosine transform," in *IEEE Transactions on Computer*, pp. 90–93, January 1974.

[136] C. S. Burrus, R. A Gopinath and H-T. Guo, *Introduction to wavelets and wavelet transforms: A primer*. Prentice-Hall, 1998.

[137] C. Valens, "A really friendly guide to wavelets," http://perso.wanadoo.fr/polyvalens/clemens/wavelets/wavelets.html.

[138] Wen-Hsiung Chen, C. Harrispn Smith and S. C. Fralick, "A fast computional algorithm for the discrete cosine transform," in *IEEE Transactions on Communications*, vol. 25, no. 9, pp. 1004–1009, September 1977.

[139] R. J. Clarke, *Digital compression of still images and video*. Academic Press, 1995.

[140] C. Sidney Burrus, Ramesh A. Gopinath and Haitao Guo, *Introduction to Wavelets and Wavelet Transforms - A Primer*. Prentice-Hall, 1998.

[141] A. V. Oppenheim and R. W. Schafer, *Digital signal processing*. Prentice-Hall, 1997.

[142] J. G. Proakis and D. G. Manolakis, *Digital signal processing: Principles, Algorithms and Applications, 3 ed.* Prentice-Hall, 1996.

[143] D. Gabor, "Theory of communication," in *Journal of IEE*, vol. 93, Part III, no.26, pp. 429–457, November 1946.

[144] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelets representation," in *IEEE Transactions on Pattern Analysis and Machine Intelligent*, vol. 11, no. 7, pp. 674–693, July 1989.

[145] ISO/IEC, "ISO/IEC10918-1:1994, Information technology - Digital compression and coding of continuous-tone still images. requirements and guidelines.," 1994.

[146] ISO/IEC, "ISO/IEC10918-1:1999, Information technology - Lossless and near-lossless compression and coding of continuous-tone still images: Baseline," 1999.

[147] International Telecommunication Union, "ITU-T H.261, Video codec for audiovisual services at p x 64 kbit/s," 1993.

[148] International Telecommunication Union, "ITU-T H.263, Video coding for low bit rate communication," 1998.

[149] International Telecommunication Union, "ITU-T H.264, Advance video coding for generic audiovisual services," 2003.

[150] ISO/IEC, "ISO/IEC14496-10:2003, Information technology - Coding of audio-visual object - Part 10: Advanced video coding," 2003.

[151] ISO/IEC, "ISO/IEC11172:1993, Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5Mbit/s," 1993.

[152] ISO/IEC, "ISO/IEC13818:2000, Information technology - Generic coding of moving pictures and associated audio information," 2000.

[153] ISO/IEC, "ISO/IEC14496:1999, Information technology - Coding of audio-visual objects," 1999.

[154] International Telecommunication Union, "ITU-T H.262, Information technology - Generic coding of moving pictures and associated audio information: Video," 2000.

[155] ISO/IEC, "ISO/IEC15444-3, Part 3: Motion JPEG2000," http://www.jpeg.org/JPEG2000.html, 2000.

[156] D. Santa-Cruz, T. Ebrahimi, J. Askelof, M. Larsson and C. A. Christopoulos, "JPEG2000 still image coding versus other standards," in *SPIE's 45th annaual meeting, Applications of Digital Image Processing XXIII*, vol. 4115 of *Proceedings of SPIE 2000*, pp. 446–454, August 2000.

[157] W-H. Chang, Y-S. Lee, W-S. Peng and C-Y. Lee, "A line-based, memory efficient and programmable architecture for 2D DWT using lifting scheme," in *Proceedings of the 2001 IEEE International Symposium on Circuits and Systems, ISCAS 2001*, vol. 4, pp. 330–333, 2001.

[158] K. C. B. Tan and T. Arslan, "Low power embedded extension algorithm for the lifting-based discrete wavelet transform in JPEG2000," in *IEE Electronics Letters*, vol. 37, no. 22, pp. 1328–1330, October 2001.

[159] C. Chrysafis and A. Ortega, "Line-based, reduced memory, wavelet image compression," in *IEEE Transactions on Image Processing*, vol. 9, no. 3, pp. 378–389, March 2000.

[160] M. J. T. Smith and S. L. Eddins, "Analysis/synthesis techniques for subband image coding," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 38, no. 8, pp. 1446–1456, August 1990.

[161] S. A. Martucci and R. M. Mersereau, "The symmetric convolution approach to the nonexpansive implementations of FIR filter banks for images," in *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP93*, vol. 5, pp. 65–68, April 1993.

[162] Charles H. Roth Jr., *Fundamental of logic design, 4th Edition.* West Publishing Company, 1992.

[163] A. P. Chandrakasan and R. W. Brodersen, eds., *Low Power Digital* CMOS *Design.* Kluwer Academic Publishers, 1995.

[164] X. W. Wu and M. Pedram, "Low power sequential circuit design by using priority encoding and clock gating," in *Proceedings of the 2000 International Symposium on Low Power Electronics and Design*, pp. 143–148, 2000.