Memory- and energy-efficient VLSI architectures for 2-D discrete wavelet transformation

Hu, Yusong

2015

Hu, Y. (2015). Memory- and energy-efficient VLSI architectures for 2-D discrete wavelet transformation. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/65234>

<https://doi.org/10.32657/10356/65234>

**MEMORY- AND ENERGY-EFFICIENT VLSI
ARCHITECTURES FOR 2-D DISCRETE WAVELET
TRANSFORMATION**

Yusong Hu

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

May 2015

Acknowledgement

The completion of my Ph. D. dissertation has been the most challenging mission I have ever faced. Without the support, guidance and encouragement of many people, there is no way I could have completed it. First and foremost, I would like to thank my supervisor, Associate Professor Jong Ching Chuen, who guided me all the way through my Ph. D. journey. I appreciate his time and effort spent on the research meetings, manuscript and thesis reviewing. His liberal thinking always inspires me and guides me to define problems and to discover the unknowns. I couldn't have proposed the new ideas and solutions without his constructive suggestions and technical advices. I cannot imagine a better supervisor. I am also indebted to my senior fellow Joshua Low Yung Lih for his valued suggestions and technical support.

I would like to thank Professor Viktor K. Prasanna. Without his support, I wouldn't have the chance to work in his group as a research scholar for one year in the University of Southern California. He has broadened my view, encouraged me to think from a higher level and guided me to develop a research methodology. His passion and dedication to research has greatly motivated me. I feel privileged to work with one of the best research teams in the field. I would like to acknowledge Sanmukh Rao Kuppannagari, Ren Chen, Geoffrey Tran, Andrea Sanny, Da Tong, Shijie Zhou, Shreyas Girish Singapura and Yun Qu. I have learnt a lot while working with them and I am grateful for their help.

I would like to thank my friends who have spent so many memorable days together with me and helped me a lot, including Xiang Li, Yi Yu, Jie Chen, Peng Hu, Zhen Zuo, Yan Wang, Zhichen Li, Yuan Yuan, Shuanglong Xie, Jiye Zhang, Zheyu Li, Zhixiang Bai, Wenxin Li, Yueming Zhou, Zhen Chen, Yujia Ma, Ye Du, etc.

I must thank my girlfriend, Yunyun Huang, whose support and encourage are crucial for me throughout the journey.

Lastly, I would like to thank my family for their love and encouragement. I am grateful to my parents, Aiping Li and Hongtao Hu, who have raised me and supported me in all my pursuits. I could not have accomplished the Ph. D. study without them.

Table of Contents

Acknowledgement	i
Table of Contents	iii
Abstract	vii
List of Figures	ix
List of Tables	xiv
Abbreviations	xvi
Chapter 1. Introduction	1
1.1 Background	1
1.2 Motivation	4
1.2.1 Reducing the Silicon Area	4
1.2.2 Improving the Energy Efficiency	5
1.3 Contributions	5
1.4 Organization	8
Chapter 2. Review of DWT Architectures	9
2.1 Fundamentals of Discrete Wavelet Transform	9
2.2 Mohanty's Convolution-Based Multi-Level DWT Architecture	17
2.3 Mohanty's Memory-Efficient Architecture for Multi-Level 2-D DWT	19
2.4 Huang's Flipping Method	20
2.5 Huang's Overlapped Stripe-Based Scanning Method	22
2.6 Mohanty's Area- and Power-Efficient Lifting-Based 2-D DWT Architecture	25
2.7 Tian's Efficient MIMO Architecture for 2-D Lifting-Based DWT	28
2.8 Xiong's Efficient Architecture for 2-D Multi-Level DWT	29
2.9 Lai's High-Performance and Memory-Efficient 2-D DWT Architecture	30
2.10 Lee's Precision-Aware Self-Quantizing Lifting-Based 2-D DWT Architecture	31

Chapter 3. Memory-Efficient Single-Level 2-D DWT Architecture	34
3.1 Overview	34
3.2 Lifting Scheme and Flipping Structure	34
3.3 Proposed Architecture.....	35
3.3.1 Data Scanning Method	35
3.3.2 Data Processing Pipe (DPP)	37
3.3.3 DWT Architecture	39
3.4 Hardware Estimation and Performance Comparison	43
3.4.1 Hardware Estimation	43
3.4.2 Comparison	43
3.4.3 Implementation.....	44
3.5 Memory-Efficient 2-D DWT Architecture Based on Precision-Aware Bit Width Optimization.....	46
3.5.1 Flipping Scheme with Variable Coefficients	46
3.5.2 Memory-Efficient 2-D DWT Architecture.....	47
3.5.3 Integer Bit Optimization Based on Dynamic Range Analysis.....	50
3.5.4 Fractional Bit Optimization Based on Noise Modeling	53
3.6 Hardware Estimation and Performance Comparison	57
3.7 Summary	58
Chapter 4. Energy- and Area-Efficient Parameterized Lifting-Based 2-D DWT Architecture	60
4.1 Overview	60
4.2 Energy-Efficient 2-D DWT Architecture.....	61
4.2.1 Design Assumptions	61
4.2.2 Overlapped Block-Based Scanning Method	63
4.2.3 Parameterized 2-D DWT Processor	65
4.2.4 DRAM Power Modeling and Activation Schedule	69

4.3	Experiments and Evaluation	70
4.3.1	Experimental Setup	70
4.3.2	Algorithm-Mapping Parameter Optimization	70
4.3.3	Peak Energy Efficiency	77
4.3.4	Performance Comparison	77
4.4	Discussions on Energy Performance.....	82
4.5	Summary	85
Chapter 5.	Memory-Efficient Multi-Level 2-D DWT Architecture	86
5.1	Overview	86
5.2	Review of Lifting Scheme and Flipping Method.....	88
5.3	Proposed Input Data Scanning Method.....	90
5.3.1	Overlapped Stripe-Based Scanning Method	90
5.3.2	Data Sequencing for Higher Level Decomposition	94
5.4	Proposed Architecture.....	99
5.4.1	Data Processing Pipe (DPP)	99
5.4.2	DWT Architecture for Level 1 Decomposition (Arch I)	101
5.4.3	DWT Architecture for Level j Decomposition (Arch j)	105
5.4.4	Proposed Pipelined Multi-Level DWT Architecture.....	108
5.5	Hardware Estimation and Performance Analysis.....	108
5.5.1	Theoretical Estimation	108
5.5.2	Performance Comparison	111
5.5.3	Synthesize Results	113
5.6	Summary	115
Chapter 6.	Conclusions and Future Works	116
6.1	Conclusions.....	116
6.2	Future Works.....	118

6.2.1	Evaluation of Memory and Energy Efficiency Based on Fully Layout Designs	118
6.2.2	Energy-Efficient Architecture for Multi-Level 2-D DWT Architecture	119
6.2.3	Memory- and Energy-Efficient 3-D DWT Architecture	119
6.2.4	Model-Based Design Space Exploration Methodology for Energy Efficiency..	120
6.2.5	Memory- and Energy-Efficient Architecture for Machine Learning Applications..	120
Bibliography		122

Abstract

The introduction of Discrete Wavelet Transform (DWT) has opened a new arena for digital signal processing. DWT has been applied to various applications such as image analysis, video processing and computer graphics. DWT has also been adopted in the JPEG 2000 standard due to its favorable characteristics, such as multi-resolution representation and the ability to decorrelate large image. While being ubiquitous, DWT, especially 2-D (2-Dimensional) DWT, is computationally intensive. It requires not only a large amount of arithmetic resources but also sizable memories. Many modern applications for real-time processing and mobile computation are time critical and energy sensitive. The conventional general purpose processors are unable to satisfy the hunger for computational speed and the demand for energy efficiency at the same time. Therefore, specifically designed VLSI DWT architectures are required to tackle the issue. In this thesis, the architecture for 2-D DWT is studied and novel 2-D DWT architectures are proposed which achieve high memory and energy efficiency.

The existing 2-D DWT architectures are investigated and it is observed that the data scanning method has a significant impact on the memory efficiency of DWT architecture. Based on the observation, a memory-efficient DWT architecture is proposed by employing a novel stripe-based data scanning method, which is able to reduce the on-chip memory size of the DWT architecture with only a minor increase in external bandwidth. Besides, by employing the flipped lifting algorithm, the critical path length is reduced without increasing the arithmetic resource. The proposed architecture also has a scalable throughput and constant latency. The synthesis results show that the proposed architecture achieves better area-delay product (ADP) by up to 32.3% compared to the best existing design. In order to further reduce the on-chip memory size of the proposed DWT architecture, a wordlength optimization is proposed. Instead of reducing the number of words of the on-chip memory, the proposed method reduces the wordlength of the on-chip memory without sacrificing the precision of output. The synthesis results show a better ADP of 25.1% over the best existing design.

As energy efficiency has become one of the most important metrics in computing, an energy-efficient 2-D DWT architecture is proposed. With the observation that it is the external memory and the on-chip memory that dominates the energy consumption of DWT architecture, an

overlapped block-based scanning method is proposed, which optimizes the number of external memory accesses and the on-chip memory size. By evaluating the energy efficiency of various design points, the energy-optimized algorithm-mapping parameters are determined. As a result, the proposed architecture reduces the number of external accesses by up to 33% compared to the best existing design. Furthermore, the on-chip memory size is also reduced to a constant. Consequently, the energy consumption is significantly reduced. The experimental results show the proposed design achieves better energy efficiency by 269% and 181% over the best existing design for 9/7 and 5/3 filter, respectively.

A memory-efficient high-throughput architecture for multi-level 2-D DWT is also proposed and presented. The overlapped striped-based scanning method is adopted to the architecture for the first level DWT to eliminate the area-dominating temporal memory. The design is based on the pipelined architecture, which does not require frame memories. The data flow and computations of higher level DWT are re-scheduled in accordance to the proposed data scanning method to achieve 100% hardware utilization. The proposed architecture has a constant latency, short critical path delay and scalable throughput. Compared with the best existing design, the proposed architecture achieves a better ADP by 60% and higher throughput by 97%.

The thesis details the development of the proposed data scanning methods, the computation scheduling, the evaluation of energy efficiency, the proposed hardware architectures, the performance evaluation of the architectures and the comparison with the state-of-the-art designs.

List of Figures

Fig. 1.1 Convolution-based and lifting-based architecture (single level)	2
Fig. 1.2 (a) 2-D separable DWT architecture, (b) 2-D non-separable DWT architecture (single-level)	3
Fig. 1.3 (a) Folded 2-D DWT architecture, (b) Pipelined 2-D DWT architecture	3
Fig. 2.1 Quadrature mirror filter structure	13
Fig. 2.2 Filter bank structure	13
Fig. 2.3 Lifting structure	15
Fig. 2.4 Construct filter in lifting steps [70].....	16
Fig. 2.5 Three-level 2-D DWT [24]	17
Fig. 2.6 Structure of PU-1 and delay unit [24].....	18
Fig. 2.7 Structure of Subcell [24]	18
Fig. 2.8 The architecture for 1-level lifting 2-D DWT [34]	20
Fig. 2.9 The cascaded architecture for J -level lifting 2-D DWT [34]	20
Fig. 2.10 Signal flow graph of lifting-scheme [30]	21
Fig. 2.11 Flipping method [30]	22
Fig. 2.12 Block-based scanning method [31].....	24
Fig. 2.13 Stripe-based scanning method [31].....	24
Fig. 2.14 1-D and 2-D systolic array [29]	26

Fig. 2.15 Proposed data access scheme and overall architecture [29]	27
Fig. 2.16 MIMO architecture for 2-D lifting-based DWT [37]	28
Fig. 2.17 FA and HA for 2-D DWT [35]	29
Fig. 2.18 Block diagram of the 2-D DWT architecture [32].....	30
Fig. 2.19 Parallel scanning method for 2-D DWT architecture [32].....	30
Fig. 2.20 High-level architecture of the 2-D DWT architecture [84]	31
Fig. 2.21 Data path of the 9/7 1-D DWT module [84].....	31
Fig. 3.1 Stripe-based scanning method	36
Fig. 3.2 Structure of the input buffer	37
Fig. 3.3 Data flow graph of flipped lifting scheme	38
Fig. 3.4 A data processing pipe comprising 4 cells.....	38
Fig. 3.5 The structure of Row-PU ($S = 4$)	39
Fig. 3.6 The transposition register	41
Fig. 3.7 The structure of Col-PU ($S = 4$).....	41
Fig. 3.8 Scaling unit	42
Fig. 3.9 Overall architecture	42
Fig. 3.10 Hardware resource tree	43
Fig. 3.11 Structure of Row-PU, temporal memory and transpose memory	48
Fig. 3.12 Structure of Col-PU and SU	49
Fig. 3.13 Comparison of ADP for image size 512×512	59
Fig. 3.14 Comparison of ADP for image size 1024×1024	59

Fig. 4.1 High level abstraction of DWT architecture.....	61
Fig. 4.2 Overlapped block-based image scanning method.....	64
Fig. 4.3 Number of image memory accesses (5/3 filter, input size of 2160×3840, $R=543$)	64
Fig. 4.4 Proposed DWT processor	65
Fig. 4.5 Structure of Cell.....	67
Fig. 4.6 Row-PU and transposition memory for 9/7 filter	67
Fig. 4.7 Col-PU and scaling unit for 9/7 filter	68
Fig. 4.8 Timing diagram of DRAM activation schedule.....	70
Fig. 4.9 Energy efficiency for 9/7 filter with small size input and various R	73
Fig. 4.10 Energy efficiency for 9/7 filter with medium size input and various R	73
Fig. 4.11 Energy efficiency for 9/7 filter with large size input and various R	73
Fig. 4.12 Energy efficiency for 5/3 filter with small size input and various R	74
Fig. 4.13 Energy efficiency for 5/3 filter with medium size input and various R	74
Fig. 4.14 Energy efficiency for 5/3 filter with large size input and various R	74
Fig. 4.15 Energy efficiency for 9/7 filter with small size input and various P ($R = 67$) ...	75
Fig. 4.16 Energy efficiency for 9/7 filter with medium size input and various P ($R = 75$)	75
Fig. 4.17 Energy efficiency for 9/7 filter with large size input and various P ($R = 75$)....	75
Fig. 4.18 Energy efficiency for 5/3 filter with small size input and various P ($R = 67$) ...	76
Fig. 4.19 Energy efficiency for 5/3 filter with medium size input and various P ($R = 75$)	76

Fig. 4.20 Energy efficiency for 5/3 filter with large size input and various P ($R = 75$)....	76
Fig. 4.21 Energy efficiency and energy consumption per image for large size input (9/7 filter)	79
Fig. 4.22 Energy efficiency and EAT comparison (9/7 filter, small size)	80
Fig. 4.23 Energy efficiency and EAT comparison (9/7 filter, medium size)	80
Fig. 4.24 Energy efficiency and EAT comparison (9/7 filter, large size)	80
Fig. 4.25 Energy efficiency and EAT comparison (5/3 filter, small size)	81
Fig. 4.26 Energy efficiency and EAT comparison (5/3 filter, medium size)	81
Fig. 4.27 Energy efficiency and EAT comparison (5/3 filter, large size)	81
Fig. 4.28 Energy efficiency for 5/3 filter with small size input and various R ($P = 20$)...	83
Fig. 4.29 Energy efficiency for 5/3 filter with medium size input and various R ($P = 20$)	83
Fig. 4.30 Energy efficiency for 5/3 filter with large size input and various R ($P = 20$)....	83
Fig. 4.31 Energy efficiency for 9/7 filter with small size input and various R ($P = 45$)...	84
Fig. 4.32 Energy efficiency for 9/7 filter with medium size input and various R ($P = 45$)	84
Fig. 4.33 Energy efficiency for 9/7 filter with large size input and various R ($P = 45$)....	84
Fig. 5.1 Overlapped stripe-based scanning method for Level 1.....	91
Fig. 5.2 Dependency graph of 2-D lifting DWT	92
Fig. 5.3 Output and input sequences of Levels 1, 2 and 3 DWTs ($r = 0$).....	96
Fig. 5.4 Output sequence of Level $j-1$ and Input sequence of Level j ($j > 1$, $r = 0$).....	97

Fig. 5.5 (a) Transposition order of Level 2 decomposition (b) Transposition order of Level j decomposition	98
Fig. 5.6 Flipped DFG of lifting scheme	100
Fig. 5.7 Data processing pipe (DPP)	100
Fig. 5.8 Structure of rDWT in Arch I: (a) Auxiliary-PU (b) Row-PU	102
Fig. 5.9 Structure of transposition register	103
Fig. 5.10 Structure of cDWT	103
Fig. 5.11 Scaling unit	104
Fig. 5.12 Structure of Arch I	104
Fig. 5.13 The structure of rDWT of Arch j	107
Fig. 5.14 Structure of Transposition Register for DPP s of Arch j	107
Fig. 5.15 The pipelined multi-level DWT architecture	109
Fig. 5.16 Structure of Splitter j	109

List of Tables

TABLE 2.1 MEMORY READ/WRITE AND SIZE OF VARIOUS SCANNING METHODS [31].	25
TABLE 2.2 COMPARISON OF MEMORY SIZE BETWEEN VARIOUS 2-D DWT ARCHITECTURES [31].	25
TABLE 2.3 INTEGER BIT AND FACTIONAL BIT FOR TWO-LEVEL DWT [84].	32
TABLE 3.1 COMPARISON BETWEEN PROPOSED ARCHITECTURE AND EXISTING ARCHITECTURES IN HARDWARE RESOURCES AND TIME COMPLEXITY (FOR 1 LEVEL 9/7 LIFTING-BASED DWT)	45
TABLE 3.2 COMPARISON OF SYNTHESIS RESULTS FOR IMAGE SIZE 512×512	45
TABLE 3.3 INTEGER BIT WIDTH OF SIGNALS (BITS)	52
TABLE 3.4 SYNTHESIZED AREA FOR ARITHMETIC RESOURCES AND TEMPORAL MEMORY	56
TABLE 3.5 FRACTIONAL BIT WIDTH OF SIGNALS (BITS)	56
TABLE 3.6 SYNTHESIS RESULTS FOR PROPOSED ARCHITECTURE	59
TABLE 4.1 OPTIMIZED VALUE OF R FOR VARIOUS IMAGE SIZES	71
TABLE 4.2 ENERGY EFFICIENCY (GOPS/W) COMPARISON BETWEEN BASELINE AND OPTIMIZED ARCHITECTURES	79
TABLE 5.1 INPUTS AND OUTPUTS OF DPPs AT DIFFERENT DWTs	101
TABLE 5.2 COMPARISON OF THEORETICAL ESTIMATION BETWEEN PROPOSED ARCHITECTURE AND EXISTING ARCHITECTURES IN HARDWARE RESOURCES AND TIME COMPLEXITY (FOR 3 LEVEL 9/7 2-D DWT)	112
TABLE 5.3 COMPARISON OF ARITHMETIC RESOURCES AND MEMORY RESOURCES (FOR $N = 512$ AND 3 LEVEL 9/7 DWT).	114

TABLE 5.4 COMPARISON OF SYNTHESIZED RESULTS (FOR $N = 512$ AND 3 LEVEL 9/7 DWT)	
.....	114

Abbreviations

DWT	Discrete Wavelet Transform
JPEG	Joint Photographic Experts Group
1-D	1-Dimensional
2-D	2-Dimensional
3-D	3-Dimensional
VLSI	Very-Large-Scale-Integration
ADP	Area-Delay Product
FIR	Finite Impulse Response
rDWT	Row-wise Discrete Wavelet Transform
cDWT	Column-wise Discrete Wavelet Transform
tDWT	Temporal-wise Discrete Wavelet Transform
HH	High-High
HL	High-Low
LH	Low-High
LL	Low-Low
PU	Processing Unit
CMOS	Complementary Metal-Oxide-Semiconductor
IB	Integer Bit
FB	Factional Bit
STM	STMicroelectronics
CDF	Cohen-Daubechies-Favreau
FPGA	Field-Programmable Gate Array
RPA	Recursive Pyramid Algorithm
PE	Processing Element
MU	Multiplier Unit
DU	Delay Unit
EPI	Energy Consumption Per Image
SFG	Signal Flow Graph
MIMO	Multiple-Input and Multiple-Output
SISO	Single-Input and Single-Output

TITO	Two-Input and Two-Output
FA	Fast Architecture
HA	High-Speed Architecture
IDBU	Input Data Buffer
WT	Wavelet Transform Module
BP	Bit Parallel
DS	Data Serial
PSNR	Peak Signal-to-Noise-Ratio
CPD	Critical Path Delay
RAM	Random-Access Memory
DPP	Data Processing Pipe
PU	Processing Unit
SR	Shift Register
HDL	Hardware Description Language
TP	Throughput
NAND	Not And
DRAM	Dynamic Random-Access Memory
SRAM	Static Random-Access Memory
SU	Scaling Unit
SA	Simulated Annealing
EAT	Energy×Area×Time
MUX	Multiplexer
GOPS/W	Giga-Operations Per Second Per Watt
DAT	Data Arrival Time
CNN	Convolutional Neural Network
RTL	Register-Transfer Level

Chapter 1. Introduction

1.1 Background

The introduction of wavelet representation has opened a new realm for signal processing. Instead of characterizing the signal over the whole time scale as the Fourier bases do, the wavelet bases provide both time (or space) and frequency localization [1]. As a result, the wavelet bases can represent piecewise regular signals with a few coefficients. Up to now, the wavelet representation has been widely applied to various fields such as audio signal processing, image analysis, signal compression, video processing and computer graphics, etc [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16]. Despite its desirable characteristics, the wavelet transform is computationally intensive. Besides, owing to the widespread use of mobile devices, the energy efficiency has become one of the most important performance metrics for computing today [17]. For many time critical and energy sensitive applications, the conventional microprocessors are unable to satisfy the intensive requirement for throughput and energy efficiency at the same time. In order to mitigate the problem, specifically designed hardware architecture for discrete wavelet transform (DWT) is necessary. Over the past decades, numerous DWT architectures have been proposed [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69].

The existing DWT architectures can be classified into two categories, namely, convolution-based [19], [20], [24], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66] and lifting-based [21], [22], [23], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [39], [40], [41], [42], [43], [44], [45], [46], [47]. The single-level one-dimension (1-D) convolution-based DWT architecture is shown in Fig. 1.1(a). The convolution-based architecture is composed of two FIR filter banks ($h(n)$ and $g(n)$) and a down-sampling step. The input signal x is processed by the $h(n)$ and $g(n)$ filter banks concurrently. Then, two set of outputs, namely low-pass (L) and high-pass (H) subbands are generated by down-sampling the outputs of the $h(n)$ and $g(n)$ filter banks, respectively. The L and H subbands are the wavelet representation of the input signal x . The idea of lifting scheme

Fig. 1.1 Convolution-based and lifting-based architecture (single level)

is to decompose the filter banks $h(n)$ and $g(n)$ into a finite sequence of simple filtering steps (lifting steps) [70]. The single-level 1-D lifting-based DWT architecture is composed of one scaling step, one splitting step and several lifting steps (number of lifting steps depends on the employed wavelet bases) as shown in Fig. 1.1(b). The input signal x is split into odd samples x_o and even samples x_e by the split step. The odd and even samples are processed by several predict and update steps before being scaled and output. While the lifting-based architectures have the advantage of less arithmetic resources, memory-efficient in-place computation and inherent parallelism, the convolution-based architectures have the advantage of short critical path [30].

In two dimensions, the wavelet representation can be computed with a method similar to the 1-D DWT. Both the convolution-based and lifting-based architectures perform the 2-D DWT of input signal in two stages, the row-wise DWT (rDWT) followed by the column-wise DWT (cDWT), or vice versa. Both types of the architectures are composed of arithmetic resources such as multipliers, adders, multiplexers and storage resources. The storage resources include transposition memory, temporal memory and frame memory. Transposition memory is used in the 2-D DWT to transpose the intermediate results produced by the rDWT for the input to the subsequent cDWT. Temporal memory is required for storing the partial results produced in both

the rDWT and the cDWT. Frame memory is required in multi-level DWT, which transforms successively the outputs of more than one level, to store the subband coefficients produced at each level for the succeeding level.

As shown in Fig. 1.2(a), the 2-D separable DWT architecture is composed of two 1-D DWT processors and memories. The first 1-D DWT processor performs the rDWT(cDWT) to the input signal x and generates two subbands H and L . The second 1-D DWT processor performs the cDWT(rDWT) to the H and L subbands respectively. The four output subbands, namely high-high (HH), high-low (HL), low-high (LH) and low-low (LL), are the wavelet representation of the input image x . The 2-D non-separable DWT architecture is shown in Fig. 1.2(b), which

Fig. 1.2 (a) 2-D separable DWT architecture, (b) 2-D non-separable DWT architecture (single-level)

Fig. 1.3 (a) Folded 2-D DWT architecture, (b) Pipelined 2-D DWT architecture

computes the rDWT and cDWT in one step. For 1-D DWT architectures, the most important design goal is to improve the utilization of arithmetic resources, while for 2-D DWT architectures, the most important design goal is to improve memory efficiency as the memories dominate the overall area and energy consumption.

In order to compute the multi-resolution wavelet representation of the signal, the multi-level DWT architecture is introduced. The multi-level 2-D DWT architectures can be classified into the folded architecture and the pipelined architecture. For the folded architecture as shown in Fig. 1.3(a), one 2-D DWT processor is in charge of performing the 2-D DWT for all levels. The *LL* subband of each level is stored in a frame memory and sent to the DWT processor as the input of the next level. Different from the folded architecture, the pipelined architecture (Fig. 1.3(b)) contains multiple 2-D DWT processors, each in charge of performing one level of 2-D DWT. As a result, no frame memory is required for storing the *LL* subband. On the other hand, the pipelined architecture consumes more arithmetic resources than the folded architecture does.

Though much work has been done on various aspects of the design of DWT architecture, the performance of DWT architecture can be further improved. There are several problems still yet to be tackled as discussed in the following section.

1.2 Motivation

Broadly, the research is motivated by improving two most important performance metrics, namely silicon area and energy efficiency.

1.2.1 Reducing the Silicon Area

In both the convolution-based and lifting-based architectures, the area is dominated by the memory. Despite the memory-efficiency advantage of the lifting-based DWT over its convolution-based counterpart, reducing the memory size is still a major challenge in 2-D lifting-based DWT architecture design [38]. Furthermore, the data scanning method employed by a DWT architecture has a significant impact on its memory size as it decides how the data flows and how the computations are scheduled. Therefore, the research work focuses on improving the memory efficiency by developing a novel data scanning method that enables the trade-off between the external memory bandwidth and internal buffer size.

1.2.2 Improving the Energy Efficiency

Energy efficiency has emerged as one of the most important metrics for computing [17]. Recently, some works have been done on investigating the energy efficiency of various computation kernels and hardware architectures [71], [72], [73], [74], [75], [76]. Although the design space of DWT architecture has been explored to certain extent by prior work, most of the existing works focus on improving the hardware utilization and reducing the memory size. Designers still face challenge when choosing an appropriate architecture for high energy efficiency. To address this problem, it is necessary to evaluate the energy efficiency of design strategies with various buffering schemes and levels of parallelism. The research work focuses on rethinking the design of DWT architecture and optimize the design for energy efficiency.

1.3 Contributions

The main contributions of the research work performed are summarized as follows.

- A new parallel lifting-based 2-D DWT architecture with high memory efficiency and short critical path is proposed. The memory efficiency is achieved with a novel scanning method that enables trade-off of external memory bandwidth and on-chip memory size. Based on the data flow graph of the flipped lifting algorithm, processing units (PUs) are developed for maximally utilizing the inherent parallelism. With S number of PUs, the throughput can be scaled while keeping the latency constant. Compared with the best existing architecture, the proposed architecture requires less memory. For an $N \times N$ image, the proposed architecture consumes a total of only $3N + 24S$ words of transposition memory, temporal memory, and pipeline registers. The synthesized results in 90-nm CMOS process show that it achieves better area-delay products than the best existing design by 32.3%, 31.5%, and 27.0% when $S = 2, 4$, and 8 , respectively, and by 26%, 26%, and 22% when the overhead for buffering the required overlapped pixels is taken into account.

The proposed architecture is described in details in Chapter 3 and in the published paper.

[77] **Y. Hu** and C. C. Jong, "A Memory-Efficient Scalable Architecture for Lifting-Based Discrete Wavelet Transform," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 60, no. 8, pp. 502-506, 2013.

- A memory-efficient 2-D Discrete Wavelet Transform (DWT) architecture based on wordlength optimization is introduced. Unlike the existing designs that improve the memory efficiency by reducing the on-chip memory words, the proposed architecture improves the memory efficiency by reducing the wordlength of the words. Based on the above proposed memory-efficient 2-D DWT architecture [77], which achieves the highest memory efficiency among the existing designs, the dynamic range and the required integer bit (IB) width of every internal signal are analyzed for further memory reduction. An architecture-specific accuracy model is constructed and the fractional bit (FB) width of every internal signal is optimized without sacrificing the precision of the subband results. Theoretical estimation shows a 17.5% reduction of the temporal memory regardless of input image size and throughput. In addition to the memory reduction, the bit width optimization also reduces the arithmetic resource. The synthesis result of a complete 2-D DWT in the STM 90-nm CMOS process shows a better area-delay product (ADP) of 25.1% over the best existing design.

The proposed architecture is described in details in Chapter 3 and in the manuscript.

[78] **Y. Hu** and C. C. Jong, "Memory-Efficient Discrete Wavelet Transform Architecture Based on Wordlength Optimization," in *IEEE International Symposium on Circuits and Systems (ISCAS '2015)*, Lisbon, May 2015, accepted for lecture presentation.

- An energy- and area- efficient parameterized lifting-based 2-D DWT architecture is proposed. Different from the existing designs which focus on improving hardware utilization and memory efficiency, energy efficiency is considered the key performance metric in this paper. Based on the observation that it is the external memory and on-chip memory that dominate the total energy consumption, a DWT architecture is proposed with an overlapped block-based image scanning method that optimizes the number of external memory reads and the on-chip memory size. By employing the overlapped block-based scanning method, the required number of external memory reads of the proposed architecture is reduced by up to 30% when compared with state-of-the-art design. Consequently, the energy efficiency of the proposed architecture is significantly improved. Additionally, compared with the best existing designs, the proposed design also consumes less on-chip memory. The proposed architecture is synthesized with STM 90-nm library for various image sizes. The design sustains up to 86.6% of the peak energy efficiency of the

device. Compared with the state-of-the-art design, the proposed architecture achieves a better energy efficiency by up to 269% and 181%, respectively for 9/7 and 5/3 filter. The proposed architecture is described in details in Chapter 4 and in the published paper.

[79] **Y. Hu** and Viktor K. Prasanna, "Energy- and Area-Efficient Parameterized Lifting-Based 2-D DWT Architecture on FPGA," *IEEE 18th International Conference on High Performance Extreme Computing (HPEC '14)*, September 2014.

[80] **Y. Hu** and Viktor K. Prasanna, "Energy-Efficient Parameterized 2-D Separable Convolution on FPGA," *IEEE 5th International Green Computing Conference (IGCC '14)*, November 2014.

[81] Sanmukh R. Kuppannagari, **Y. Hu** and Viktor K. Prasanna, "High Level Performance Model Based Design Space Exploration for Energy-Efficient Designs on FPGAs," *IEEE 5th International Green Computing Conference (IGCC '14)*, November 2014.

- A novel memory-efficient high-throughput scalable architecture for multi-level 2-D DWT is developed. By studying the existing DWT architectures, the data scanning method is observed to have a significant impact on the memory efficiency of DWT architecture. A novel parallel stripe-based scanning method based on the analysis of the dependency graph of the lifting scheme. With the new scanning method for multi-level 2-D DWT, a high memory efficient scalable parallel pipelined architecture is developed. The proposed architecture requires no frame memory and a temporal memory of size only $3N + 682$ for the 3-level DWT decomposition with an image of size $N \times N$ pixels with 32 pixels processed concurrently. The elimination of frame memory and the small temporal memory lead to significant reduction in overall size. The proposed architecture has a regular structure and achieves 100% hardware utilization. The synthesis results in 90 nm CMOS process show that the proposed architecture achieves a better area-delay product by 60% and higher throughput by 97% when compared to the best existing design for the CDF (Cohen-Daubechies-Favreau) 9/7 2-D DWT.

The proposed architecture is described in details in Chapter 5 and in the published paper.

[82] **Y. Hu** and C. C. Jong, "A Memory-Efficient High-Throughput Architecture for Lifting-Based Multi-Level 2-D DWT," *Signal Processing, IEEE Transactions on* , vol. 61, no. 30, pp. 4975-4987, 2013.

1.4 Organization

This thesis is organized into six chapters. In Chapter 1, the background of DWT architecture, the motivation of the research and the contribution are introduced.

In Chapter 2, the related existing works on improving the memory and energy efficiency are discussed.

In Chapter 3, a novel parallel lifting-based 2-D DWT architecture with high memory efficiency and short critical path is presented. The wordlength optimization is employed to optimize the on-chip memory size.

In Chapter 4, an energy-efficient lifting-based 2-D DWT architecture is presented.

In Chapter 5, a novel memory-efficient high-throughput scalable architecture for multi-level 2-D DWT is presented.

Finally, Chapter 6 summarizes the works and discusses the areas for future study.

Chapter 2. Review of DWT Architectures

2.1 Fundamentals of Discrete Wavelet Transform

The goal of multiresolution signal decomposition [1] is to decompose the square-integrable function space into subspaces, which are represented by a series of orthonormal wavelet basis. The signal with finite energy is denoted by $f(x) \in L^2(\mathbf{R})$, where $L^2(\mathbf{R})$ denotes the vector space of square-integrable function $f(x)$. Let V_j and W_j denote the scaling subspace and the wavelet subspace of $L^2(\mathbf{R})$ at resolution level j , respectively. Each scaling subspace V_j represents an approximation of $L^2(\mathbf{R})$ at resolution level j . Correspondingly, the approximation of $f(x)$ on V_j is denoted by $f_j(x)$. The scaling subspace increases with the index j . $\forall j \in \mathbb{Z}$, $V_j \subset V_{j+1}$. Each wavelet subspace W_j represents the detail information of $L^2(\mathbf{R})$ at resolution level j . The detail information of $f(x)$ on W_j is denoted by $\Delta f_j(x)$. The detail information is the difference between two successive approximation subspaces V_j and V_{j+1} . $\forall j \in \mathbb{Z}$, $V_{j+1} = W_j \oplus V_j$ and V_j is orthogonal to W_j . Both $f_j(x)$ and $\Delta f_j(x)$ can be regarded as a projection on subspace V_j and W_j , respectively.

Theorem 1 [1]: Let V_j be a multiresolution representation of $L^2(\mathbf{R})$, there exists a unique scaling function $\phi(x) \in L^2(\mathbf{R})$ such that $\phi_{j,k}(x) = \sqrt{2^j} \phi(2^j x - k)$, $k \in \mathbb{Z}$ is an orthonormal basis of V_j .

Based on theorem 1, the orthogonal projection of $L^2(\mathbf{R})$ on scaling subspace V_j can be built by $\phi_{j,k}(x)$, which is derived from scaling function $\phi(x)$ by dilation and translation:

$$\forall f(x) \in L^2(\mathbf{R}), f_j(x) = \sum_{k=-\infty}^{+\infty} a_{j,k} \phi_{j,k}(x), k \in \mathbb{Z} \quad (2.1)$$

$$a_{j,k} = \int_{-\infty}^{+\infty} f(x) \phi_{j,k}(x) dx \quad (2.2)$$

In order to build a multiresolution representation based on detail information of $L^2(\mathbf{R})$, Theorem 2 [1] was introduced.

Theorem 2: Let $\phi(x)$ be the scaling function with corresponding conjugate filter H . Let $\psi(x)$ be a function whose Fourier transform is $\hat{\psi}(\omega) = G(\frac{\omega}{2})\hat{\phi}(\frac{\omega}{2})$, where $G(\omega) = e^{-j\omega}\overline{H(\omega + \pi)}$.

Then, $\psi_{j,k}(x) = \sqrt{2^j}\psi(2^j x - k)$, $k \in \mathbb{Z}$ is an orthonormal basis of W_j , and $\psi_{j,k}(x) = \sqrt{2^j}\psi(2^j x - k)$, $(j,k) \in \mathbb{Z}^2$ is an orthonormal basis of $L^2(\mathbf{R})$.

Based on Theorem 2, the detail information of $L^2(\mathbf{R})$ on wavelet subspace W_j can be built by $\psi_{j,k}(x)$, which is derived from wavelet function $\psi(x)$ by dilation and translation:

$$\forall f(x) \in L^2(\mathbf{R}), \Delta f_j(x) = \sum_{k=-\infty}^{+\infty} b_{j,k} \psi_{j,k}(x), k \in \mathbb{Z} \quad (2.3)$$

$$b_{j,k} = \int_{-\infty}^{+\infty} f(x) \psi_{j,k}(x) dx \quad (2.4)$$

With Theorem 1 and 2, $L^2(\mathbf{R})$ can be reconstructed with scaling subspace V_j and wavelet subspace W_j . As W_j is a orthogonal complement of V_j , $\phi_{j,k}(x) = \sqrt{2^j}\phi(2^j x - k)$, $k \in \mathbb{Z}$ and $\psi_{j,k}(x) = \sqrt{2^j}\psi(2^j x - k)$, $k \in \mathbb{Z}$ together compose a orthogonal basis of V_{j+1} . Let $f_{j+1}(x) \in V_{j+1}$ be the corresponding approximation of $f(x)$, (2.5) is obtained.

$$\forall f(x) \in L^2(\mathbf{R}), f_{j+1}(x) = f_j(x) + \Delta f_j(x) \quad (2.5)$$

In practice, digital system can only process discrete signal and the original signal has limited samples. Therefore, the measured samples are assumed to be the approximation of $f(x)$ on resolution level 0. Suppose V_0 is to be composed by J ($J > 0$) levels, and $f_0(x)$ is the approximation of $f(x)$ on resolution level 0. After multiresolution signal decomposition, those samples can be further represented by detail on the lower resolution levels and its approximation on $-J$ level:

$$\forall f(x) \in L^2(\mathbf{R}), f_0(x) = \sum_{k=-\infty}^{+\infty} a_{-J,k} \sqrt{2^{-J}} \phi_{-J,k}(x) + \sum_{j=-J}^{-1} \sum_{k=-\infty}^{+\infty} b_{j,k} \psi_{j,k}(x), k \in \mathbb{Z} \quad (2.6)$$

$$a_{-J,k} = \int_{-\infty}^{+\infty} f(x) \phi_{-J,k}(x) dx \quad (2.7)$$

$$b_{j,k} = \int_{-\infty}^{+\infty} f(x) \psi_{j,k}(x) dx \quad (2.8)$$

The above representation is called orthogonal wavelet representation. Based on orthogonal wavelet representation and given the coefficients $a_{j+1,k}$ on the V_{j+1} basis, it is an important concern to compute the coefficients $a_{j,n}$ and $b_{j,n}$ on the W_j and V_j basis, respectively.

Because $V_j \subset V_{j+1}$, $\phi_{j+1,k}(x) = \sqrt{2^{j+1}} \phi(2^{j+1}x - k)$, $k \in \mathbb{Z}$ is a basis for both V_j and V_{j+1} . $\phi_{j,n}(x)$ is defined by:

$$\phi_{j,n}(x) = \sum_{k=-\infty}^{+\infty} h(n,k) \phi_{j+1,k}(x) \quad (2.9)$$

$$h(n,k) = \int_{-\infty}^{+\infty} \phi_{j,n}(x) \phi_{j+1,k}(x) dx = \int_{-\infty}^{+\infty} \sqrt{2} \phi(2x - k) \phi(x - n) dx = \int_{-\infty}^{+\infty} \phi_{1,k}(x) \phi(x - n) dx \quad (2.10)$$

Substitute x with $x + n$:

$$h(k - 2n) = \int_{-\infty}^{+\infty} \sqrt{2} \phi(2x + 2n - k) \phi(x) dx = \int_{-\infty}^{+\infty} \phi_{1,k-2n}(x) \phi(x) dx \quad (2.11)$$

Computing the inner products of $f(x)$ with both sides of the equation (2.9):

$$\int_{-\infty}^{+\infty} f(x) \phi_{j,n}(x) dx = \sum_{k=-\infty}^{+\infty} h(k - 2n) \int_{-\infty}^{+\infty} f(x) \phi_{j+1,k}(x) dx \quad (2.12)$$

Substitute the integrals with $a_{j,n}$ in (2.2):

$$a_{j,n} = \sum_{k=-\infty}^{+\infty} h(k - 2n) a_{j+1,k} \quad (2.13)$$

The computation of the wavelet coefficients $b_{j,n}$ is similar to the process of computing the approximation coefficients $a_{j,n}$. Because $f_{j+1,k}(x) = \sqrt{2^{j+1}} f(2^{j+1}x - k)$, $k \in \mathbb{Z}$ is also a basis for both W_j , $\psi_{j,n}(x)$ is defined by:

$$\psi_{j,n}(x) = \sum_{k=-\infty}^{+\infty} g(n,k) \phi_{j+1,n}(x) \quad (2.14)$$

$$g(n,k) = \int_{-\infty}^{+\infty} \phi_{j,n}(x) \psi_{j+1,k}(x) dx = \int_{-\infty}^{+\infty} \sqrt{2} \phi(2x-k) \psi(x-n) dx = \int_{-\infty}^{+\infty} \phi_{1,k}(x) \psi(x-n) dx \quad (2.15)$$

Substitute x with $x+n$:

$$g(k-2n) = \int_{-\infty}^{+\infty} \sqrt{2} \phi(2x+2n-k) \psi(x) dx = \int_{-\infty}^{+\infty} \phi_{1,k-2n}(x) \psi(x) dx \quad (2.16)$$

Computing the inner products of $f(x)$ with both sides of the equation (2.14):

$$\int_{-\infty}^{+\infty} f(x) \psi_{j,n}(x) dx = \sum_{k=-\infty}^{+\infty} g(k-2n) \int_{-\infty}^{+\infty} f(x) \phi_{j+1,k}(x) dx \quad (2.17)$$

Substitute the integrals with $a_{j,n}$ (2.2) and $b_{j,n}$ (2.4) and get:

$$b_{j,n} = \sum_{k=-\infty}^{+\infty} g(k-2n) a_{j+1,k} \quad (2.18)$$

Let filter H and G have the impulse response $h(k-2n)$ and $g(k-2n)$ for $(k-2n) \in \mathbb{Z}$, respectively. Let $\hat{H}$ and $\hat{G}$ be the mirror filters of H and G , with the impulse response of $h(2n-k)$ and $g(2n-k)$. With the equations (2.13) and (2.18), the approximation coefficients $a_{j,n}$ and wavelet coefficients $b_{j,n}$ from the original approximation coefficient $a_{0,k}$ of $f(x)$ can be recursively derived. The computation process is convolving the approximation coefficients of a resolution level with $\hat{H}$ and $\hat{G}$ and down sampling by 2.

Fig. 2.1 Quadrature mirror filter structure

Fig. 2.2 Filter bank structure

Quadrature mirror filters was introduced in [1] to implement the wavelet transform. Fig. 2.1 shows the structure of the Quadrature mirror filter.

For each input sample set of N sample points, the H and G filters each outputs $N/2$ points. Then, the approximated coefficients generated by the H filter are recursively processed by the H and G filters for another level of resolution. The H and G filters can be implemented with two FIR filters followed by down sampling with a factor of 2. The structure is common known as the convolution-based architecture.

The lifting scheme, proposed in [83], is an alternative way of constructing the wavelet filters. The idea of lifting scheme is to make use of the spatial correlation inherent in practical signal to build a sparse approximation of the signal [70]. Since wavelet is built based on the perfect reconstruction filter banks, the condition for perfect reconstruction [84] is first discussed.

The filter bank structure is illustrated in

Fig. 2.2, where $x(n)$ is input signal, $\hat{x}(n)$ is the reconstructed signal, $\uparrow 2$ and $\downarrow 2$ denote respectively up and down sampling by 2. H_0 and G_0 are analysis filters and H_1 and G_1 are synthesis filters.

After the transform of $H_1(\uparrow 2)(\downarrow 2)H_0$ and $G_1(\uparrow 2)(\downarrow 2)G_0$, the z-transform of reconstructed signal $\hat{x}(n)$ is obtained.

$$\hat{X}(z) = \frac{1}{2}(H_1(z)H_0(z) + H_1(z)H_0(-z))X(z) + \frac{1}{2}(G_1(z)G_0(z) + G_1(z)G_0(-z))X(z) \quad (2.19)$$

Where $H_1(z)H_0(-z)$ and $G_1(z)G_0(-z)$ are called the aliasing terms.

The perfectly reconstructed signal $\hat{X}(z)$ should have neither aliasing nor distortion, but only delay of the filter bank. Supposing the delay of filter bank is l , the condition for perfect reconstruction is given by:

$$H_1(z)H_0(-z) + G_1(z)G_0(-z) = 0 \quad (2.20)$$

$$H_1(z)H_0(z) + G_1(z)G_0(z) = 2z^{-l} \quad (2.21)$$

The conditions can be represented by matrix:

$$\begin{bmatrix} H_1(z) & G_1(z) \end{bmatrix} \begin{bmatrix} H_0(z) & H_0(-z) \\ G_0(z) & G_0(-z) \end{bmatrix} = \begin{bmatrix} 2z^{-l} & 0 \end{bmatrix} \quad (2.22)$$

where $H_m = \begin{bmatrix} H_0(z) & H_0(-z) \\ G_0(z) & G_0(-z) \end{bmatrix}$ is known as the modulation matrix.

The operation $(\downarrow 2)H_0$ is not efficient because half of the components of H_0x are not required [85]. For the sake of efficiency, the polyphase matrix was introduced in [70], which enables the down sampling the signal before computing it. The FIR filters H_i and G_i are represented as:

$$H_i(z) = H_{i0}(z^2) + z^{-1}H_{i1}(z^2) \quad (2.23)$$

$$G_i(z) = G_{i0}(z^2) + z^{-1}G_{i1}(z^2) \quad (2.24)$$

where $H_{i0}(z^2)$ and $H_{i1}(z^2)$, and $G_{i0}(z^2)$ and $G_{i1}(z^2)$, represent the even-indexed and odd-indexed terms of $H_i(z)$ and $G_i(z)$, respectively.

Fig. 2.3 Lifting structure

The polyphase matrix $H_p(z^2)$ and $\hat{H}_p(z^2)$ of the analysis and synthesis filter bank are defined as:

$$H_p(z^2) = \begin{bmatrix} H_{00}(z^2) & H_{01}(z^2) \\ G_{00}(z^2) & G_{01}(z^2) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} H_0(z) & H_0(-z) \\ G_0(z) & G_0(-z) \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & z \end{bmatrix} \quad (2.25)$$

$$\hat{H}_p(z^2) = \begin{bmatrix} H_{10}(z^2) & H_{11}(z^2) \\ G_{10}(z^2) & G_{11}(z^2) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} H_1(z) & H_1(-z) \\ G_1(z) & G_1(-z) \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & z \end{bmatrix} \quad (2.26)$$

Combined with (2.19), $\hat{X}(z)$ can be represented as:

$$\hat{X}(z) = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \begin{bmatrix} H_{10}(z^2) & G_{10}(z^2) \\ H_{11}(z^2) & G_{11}(z^2) \end{bmatrix} \begin{bmatrix} H_{00}(z^2) & H_{01}(z^2) \\ G_{00}(z^2) & G_{01}(z^2) \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} X(z) \\ X(-z) \end{bmatrix} \quad (2.27)$$

From this equation, the structure of lifting structure can be constructed as shown in Fig. 2.3.

The lifting scheme is actually a relationship between the perfect reconstruction filter pairs, $[H_0, G_0]$ and $[H_1, G_1]$. With lifting scheme, the analysis and synthesis wavelet filters can be gradually constructed with different properties from the simple Lazy wavelet as discussed below.

Theorem 1 (Lifting) [70]:

Let filter pair H and G be complementary, i.e. the determinant of their corresponding polyphase matrix is a monomial. Any other finite filter G' complementary to H can be derived from:

$$G'(z) = G(z) + H(z)S(z^2) \quad (2.28)$$

where $S(z^2)$ is a Laurent polynomial [70].

Fig. 2.4 Construct filter in lifting steps [70]

Theorem 2 (Dual Lifting) [70]:

Let filter pairs H and G be complementary, any other finite filter H' complementary to G can be derived from:

$$H'(z) = H(z) + G(z)T(z^2) \quad (2.29)$$

where $T(z^2)$ is a Laurent polynomial.

From Theorem 1, a new polymatrix below can be obtained.

$$P'(z) = P(z) \begin{bmatrix} 1 & 0 \\ T(z) & 1 \end{bmatrix} \quad (2.30)$$

From Theorem 2, a new polymatrix below can be obtained.

$$P'(z) = P(z) \begin{bmatrix} 1 & S(z) \\ 0 & 1 \end{bmatrix} \quad (2.31)$$

Fig. 2.4 shows the wavelet analysis filter constructed by lifting steps.

Conversely, a pair of complementary filter H and G can be decomposed into lifting steps. By factorizing the polyphase matrix into several upper and lower triangular matrices and one diagonal matrix as given in (2.32), computation speed can be gained when compared to convolution-based implementation.

$$P(z) = \prod_{i=1}^m \begin{bmatrix} 1 & S_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ T_i(z) & 1 \end{bmatrix} \begin{bmatrix} 1/K & 0 \\ 0 & K \end{bmatrix} \quad (2.32)$$

where K is the scaling coefficient. The fundamental structure of the forward lifting scheme is

composed of four steps, namely, split, predict, update and scaling, which corresponding respectively to the down sampling block the $S(z)$ blocks, the $T(z)$ blocks and the K and $1/K$ blocks in Fig. 2.4 [70].

2.2 Mohanty's Convolution-Based Multi-Level DWT Architecture

A memory-efficient architecture for multi-level 2-D DWT is introduced by Mohanty and Meher [24]. Their architecture is convolution-based pipelined design, which improves the memory-efficiency and reduces power dissipation by trading-off memory and arithmetic resources. The following design strategies are adopted in their proposed architecture.

- Pipelined architecture is chosen to avoid frame buffer.
- Convolution scheme is used to take the maximum advantage of overlapped stripe-based data scanning method.
- Overlapped stripe-based data scanning method is employed at each DWT level to reduce the size of on-chip memory.
- Appropriate pixel block size is selected for each DWT level to achieve 100% hardware utilization
- RPA (Recursive Pyramid Algorithm) [64] alike computation approach and line-based scanning are adopted to compute the rest of the higher level DWT if block size is insufficient.

Fig. 2.5 [24] shows their proposed three-level 2-D DWT architecture. It has three processing units (PUs), each of which corresponds to one level of decomposition. As shown in Fig. 2.6(a),

Fig. 2.5 Three-level 2-D DWT [24]

the image is partitioned into the blocks of 16 pixels and two consecutive input blocks are overlapped by $K - 2$ pixel samples, where K is the filter order. As a result of down sampling in both row-wise and column-wise processing, the computational complexity at each level of the 2-D DWT is reduced to a quarter of its preceding level. Thus, for a parallel architecture, in order to achieve 100% hardware utilization, the computational resources at each PU is also a quarter of its preceding PU. Let the input image be $M \times N$. Within every M clock cycles, the pixel blocks are fed into PU-1 in a top-down direction from the first row to the last row. Then, the next set of M clock cycle begins.

The PU-1 is composed of 8 processing elements (PE), and each PE is composed of Subcell-1, Subcell-2, Delay-unit and 2DMUXes. The structure of PU-1, PE and Delay-unit are shown in Fig. 2.6(a), (b) and (c), respectively.

Fig. 2.6 Structure of PU-1 and delay unit [24]

Fig. 2.7 Structure of Subcell [24]

The structure of Subcell is shown in Fig. 2.7, where Fig. 2.7(a) is the Subcell of the orthogonal wavelet filter, Fig. 2.7(b) the structure of Multiplier Unit (MU) and Fig. 2.7(c) the Subcell of the biorthogonal filter. Within each PE, the pixels are multiplied with the high-pass and low-pass coefficients $g(k)$ and $h(k)$ respectively in the MU. The partial results generated by the MUs are then added together in two adder trees. Every clock cycle, the Subcell-1 generates a pair of high-pass and low-pass intermediate result u_l^1 and u_h^1 , which are transposed in the Delay Unit (DU) and output alternately for column-wise processing in Subcell-2. PU-1 outputs a pair of subband $[v_{ll}^1, v_{lh}^1]$ and $[v_{hl}^1, v_{hh}^1]$ alternately in two clock cycles. The PU-2 and PU-3 have the similar structure to PU-1 except that some internal buffers are used to cache the intermediate results.

The authors of [24] applied a memory centric strategy for their design. By proposing a convolution-based parallel architecture, the transposition memory and temporal memory are completely avoided, leading to a significant saving in memory. The memory save leads to 2.6 times less area-delay product (DAP) and 1.7 times less energy consumption per image (EPI), compared with the design proposed in [34]. However, this architecture consumes a lot more arithmetic resources than its lifting-based counterpart.

2.3 Mohanty's Memory-Efficient Architecture for Multi-Level 2-D DWT

A modular pipelined architecture for lifting-based multi-level 2-D DWT architecture is proposed in [34]. In order to achieve high memory-efficiency, regular structure and 100% hardware utilization, the following design approaches are proposed:

- A parallel line-based data scanning method is proposed, which reads multiple pixels from one line every clock cycle. The number of pixels are determined so that sufficient number of intermediate results along the row direction can be generated. With enough number of intermediate results, the processing along column direction can be initiated as early as possible. As a result, the size of the transposition number is reduced.
- The pipelined architecture is employed to process each level of DWT in different computing blocks in a cascaded structure. As a result, the output of each level is input directly to the next level without caching and the frame memory is eliminated.

Fig. 2.8 The architecture for 1-level lifting 2-D DWT [34]

Fig. 2.9 The cascaded architecture for J -level lifting 2-D DWT [34]

- In order to guarantee a 100% hardware utilization, the input rows are down-sampled at higher level DWT.

The architectures for 1-level and J -level 2-D lifting DWT are shown in Fig. 2.8 and Fig. 2.9, respectively. By employing the abovementioned methods, the architecture proposed in [34] achieves small on-chip memory size, simple control circuitry and 100% hardware utilization.

2.4 Huang's Flipping Method

The lifting-based architecture has various advantages compared to the convolution-based architecture, except longer critical path length. In [30], a flipping structure is proposed to reduce the critical path length of the lifting-based architecture. For the Daubechies 9/7 wavelet filter [70], the lifting steps can be represented by (2.33):

Fig. 2.10 Signal flow graph of lifting-scheme [30]

$$P(z) = \prod_{i=1}^m \begin{bmatrix} 1 & a(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ b(1+z) & 1 \end{bmatrix} \begin{bmatrix} 1 & c(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ d(1+z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix} \quad (2.33)$$

where a , b , c and d are the lifting coefficients. Based on (2.33), the signal flow graph (SFG) can be derived as shown in Fig. 2.10. The arrows with solid tail represent multiplications. The arrow with dotted tail represent delay by one block cycle while the circles are partial results generated between lifting steps. The critical path length is $4T_m + 8T_a$ (Fig. 2.10(a)), where T_m and T_a are the delay of a multiplier and an adder respectively. Although by introducing pipeline registers, the critical path length can be reduced to $T_m + 2T_a$ as shown in Fig. 2.10(b), it requires more registers and has longer latency.

Because the long critical path length is introduced by the multiplications on each computational node, the multiplications on the accumulative data path can be removed by flipping the computation nodes with the inverse of the lifting coefficients. Fig. 2.11 illustrates the flipping process.

Fig. 2.11(a) shows the critical path length of $2T_m + 3T_a$ before flipping. The flipped SFG is shown in Fig. 2.11(b). The computation node in Fig. 2.11(b) is split into two adders and two multipliers in Fig. 2.11(c). Because the multiplications are removed from the accumulative path, the two multiplications can be performed in parallel. The critical path length is therefore reduced

Fig. 2.11 Flipping method [30]

to $T_m + 3T_a$. Fig. 2.11(d) illustrates a modified flipping method. Instead of flipping the computation nodes with the inverse of the coefficients, it flips the nodes with the inverse of the coefficients multiplied by 2^k , where $k \in \mathbb{Z}$. After flipping, the accumulative path needs to be shifted by k bits. As the hardwired shifting operation will not increase the length of the critical path, the critical path remains $T_m + 3T_a$. This method provides greater flexibility when choosing the flipping coefficients.

2.5 Huang's Overlapped Stripe-Based Scanning Method

In [31], the VLSI DWT architectures are classified into three categories, namely, convolution-based, lifting-based and B-spline-based. All these three categories are discussed in details, including hardware complexity, memory consumption and critical path length. The paper also reviews five previous scanning methods and presents an overlapped stripe-based scanning method. The scanning method enables reasonable trade-off between external memory

bandwidth and on-chip memory size, which is a dominant factor of the design area. The five scanning methods are summarized below.

- Direct scanning method:

Assume that an $N \times N$ image is stored in the external frame buffer. The direct scanning method scans the image frame buffer in a raster order and directly stores the low-pass and high-pass intermediate results produced by rDWT in two external intermediate frame buffers. Each intermediate frame buffer has a size of $N \times N/2$. The intermediate results are then read from the intermediate frame buffer for cDWT. The frame buffer is both read and written $2N^2$ times.

- Line-based scanning method:

The line-based scanning method introduces an internal line buffer to store the partial results. It scans the image frame buffer in a raster order and caches the partial results in a line buffer. The intermediate results produced by rDWT are directly fed to cDWT. The size of the line buffer is LN where L is the number of registers used for the 1-D DWT module [31]. The frame buffer is read and written for both N^2 times

- Non-overlapped and overlapped block-based scanning method:

The block-based scanning method scans the frame buffer block by block and generates DWT coefficients block by block. Fig. 2.12 shows the scanning method, where Fig. 2.12 (a) shows the non-overlapped scanning and Fig. 2.12(b) and Fig. 2.12(c) show the overlapped scanning. B_x and B_y are the width and height of the blocks, respectively.

For both the non-overlapped and overlapped methods, an internal memory is required to store the partial results, which are shown as the grey area in the figures. Suppose the blocks are scanned in row-wise order, the size of the internal buffer is $LN + LB_y$ for the non-overlapped scanning method. For the overlapped method, there is a $2K$ -pixel overlap between the blocks of consecutive rows. K is a coefficient related to filter length. The overlapped method reduces the internal buffer size to LB_y but increases the frame buffer reads to $N^2(B_y / (B_y - 2K))$ words.

- Non-overlapped stripe-based scanning method:

The optimal Z-scan method proposed in [86] can be regarded as a non-overlapped block-based scanning method with block width $B_x = N$. In this scenario, the blocks are actually stripes. The size of the internal buffer required to store the partial results is

$LN + LS$, where S is the width of the stripe. The non-overlapped stripe-based scanning method is shown in Fig. 2.13(a).

- Overlapped stripe-based scanning method:

The proposed overlapped stripe-based scanning method in [31] is shown in Fig. 2.13(b). The difference between the overlapped and non-overlapped method is that there is a $2K$ -pixel overlap between consecutive stripes for the overlapped scanning. As a result, the internal buffer size can be reduced to LS at the expense of increased external bandwidth.

Fig. 2.12 Block-based scanning method [31]

Fig. 2.13 Stripe-based scanning method [31]

TABLE 2.1 MEMORY READ/WRITE AND SIZE OF VARIOUS SCANNING METHODS [31]

	Direct	Line-based	Non-overlapped Block-based	Overlapped Block-based	Non-overlapped Stripe-based	Proposed Overlapped Stripe-based
Frame Memory Read (words/image)	$2N^2$	N^2	N^2	$N^2 B_y / (B_y - 2K)$	N^2	$N^2 S / (S - 2K)$
Frame Memory Write (words/image)	$2N^2$	N^2	N^2	N^2	N^2	N^2
Internal Buffer Size (words)	0	LN	$L(N + B_y)$	LB_y	$L(N + S)$	LS
Control Complexity	Low	Medium	High	High	Medium	Medium

TABLE 2.2 COMPARISON OF MEMORY SIZE BETWEEN VARIOUS 2-D DWT ARCHITECTURES [31]

		Direct	Line-based		Overlapped Block-/Stripe-based		
			Recursive 1-level	Multi-level	B _y =S=16	B _y =S=64	B _y =S=256
Frame Memory Read (Mwords/sec)		198.9	99.4	74.6	198.9	113.7	102.6
Frame Memory Write (Mwords/sec)		198.9	99.4	74.6	99.4	99.4	99.4
Internal Buffer Size (words)	No Tile	0	4320	8370	64	256	1024
	Tile Size=256		1024	1984			-

TABLE 2.1 [31] compares the scanning methods listed above in terms of memory reads and writes, internal buffer size and control complexity. TABLE 2.2 [31] compares the different scanning methods for 5-level DWT decomposition of 256×256 image. As shown in TABLE 2.2, the overlapped stripe-based scanning method achieves a large memory saving at the cost of a slightly larger external memory bandwidth compared with the line-based methods. Compared with the direct method, the bandwidth is much smaller at the expense of adding an internal buffer. However, considering that the direct method needs two more frame buffers of size $N \times N / 2$, the proposed scanning method of [31] is still advantageous.

2.6 Mohanty's Area- and Power-Efficient Lifting-Based 2-D DWT Architecture

In [29], a modified line-based scanning method for 2-D lifting-based DWT is proposed. Two rows of the input image are accessed concurrently. Based on the scanning method, the data transposition between row-wise and column-wise transform is avoided, leading to a 2-D systolic

architecture with only $4N + 8P$ internal memory size, where $N \times N$ is the image size and P is the number of pixel samples in each input block.

Mohanty *et al.* observe that the data flow graph (DFG) of the lifting scheme is actually composed of several identical nodes as shown in Fig. 2.14(a). The corresponding systolic array and processing element (PE) can be derived as shown in Fig. 2.14(b) and Fig. 2.14(c), respectively. Assuming that a block of P samples is input to the architecture every clock cycle, $P/2$ systolic arrays are required to consume the pixels for a 2-D systolic array, as shown in Fig. 2.14(d).

Fig. 2.14 1-D and 2-D systolic array [29]

The data scanning method is shown in Fig. 2.15(a), (b) and (c), and the overall architecture is shown in Fig. 2.15(d), assuming that there is an 8×8 data block matrix and the block size $P = 4$. During the first set of $N/2$ clock cycles, the top $P/2$ consecutive rows are read from right to left into the row-processor. Then, during the following set of $N/2$ clock cycles, the next $P/2$ consecutive rows are read. The scanning continues until the whole image is processed. Two intermediate result matrices u_h and u_l are generated by the row-processor and directly input to the column-processor without transposition, as shown in Fig. 2.15(d). The column processor outputs one of the four subbands $[v_{LL}, v_{LH}]$ and $[v_{HH}, v_{HL}]$ every clock cycle.

Compared with the previous designs, the architecture in [29] achieves up to 62% area-delay product (ADP) reduction and 74% energy saving per image (EPI).

Fig. 2.15 Proposed data access scheme and overall architecture [29]

2.7 Tian's Efficient MIMO Architecture for 2-D Lifting-Based DWT

In [37], a multi-input/multi-output (MIMO) VLSI architecture for lifting-based DWT is proposed, which achieves high processing speed with marginal increase of hardware resources and simple control circuitry. As shown in Fig. 2.16, the architecture is composed of multiple single-input/single-output (SISO) modules, two-input/two-output (TITO) modules and delay registers (D). The SISO modules are in charge of performing row-wise DWT for multiple lines of the input. The intermediate results generated by the SISO modules are stored in the delay registers before they are input to the TITO modules for column-wise DWT. Compared with existing 2-D DWT architectures, the architecture proposed in [37] has the advantages of scalable throughput and simple control. On the other hand, the size of its on-chip memory is large.

Fig. 2.16 MIMO architecture for 2-D lifting-based DWT [37]

2.8 Xiong's Efficient Architecture for 2-D Multi-Level DWT

In [35], a novel lifting-based architecture for 2-D DWT is presented. An embedded decimation technique is exploited to optimize the architecture for 1-D DWT, which is designed to receive an input and generate an output with the low- and high-frequency components of the original data being available alternately. Based on the 1-D DWT architecture, two 2-D DWT architectures, namely a fast architecture (FA) and a high-speed architecture (HA) (Fig. 2.17), are further proposed by employing a line-based data scanning sequence. Both the FA and HA are composed of input data buffer (IDBU) and wavelet transform module (WT). The FA and HA can perform J levels of decomposition in about $2N^2(1-4^{-J})/2$ and $N^2(1-4^{-J})/3$ clock cycles, respectively. Compared with existing designs, this work improves the memory-efficiency. However, the size of the temporal memory is still large.

Fig. 2.17 FA and HA for 2-D DWT [35]

2.9 Lai's High-Performance and Memory-Efficient 2-D DWT Architecture

In [32], a high-performance and memory-efficient pipelined architecture with parallel scanning method is presented. The block diagram of the architecture in [32] is shown in Fig. 2.18. The architecture consists of two 1-D DWT processors (one column-processor and one row-processor), a temporal memory and a transposition buffer. Instead of line-based data scanning method, a parallel scanning method is proposed to reduce the internal buffer size, as shown in Fig. 2.19. Because two lines of the input image are scanned alternately into the DWT architecture, the output of the row-processor can be input to the column-processor immediately after they are generated, significantly reducing the size of the transposition buffer. On the other hand, a temporal buffer of size $4N$ is required to store the partial results. Besides, the lifting algorithm is modified to reduce the critical path length of the proposed architecture to one multiplier.

Fig. 2.18 Block diagram of the 2-D DWT architecture [32]

Fig. 2.19 Parallel scanning method for 2-D DWT architecture [32]

2.10 Lee's Precision-Aware Self-Quantizing Lifting-Based 2-D DWT Architecture

In [87], both the bit-parallel (BP) and digital-serial (DS) precision-optimized 2-D DWT architectures are presented. The precision of a multi-level DWT is customized to a given error tolerance requirement and to ensure an energy-minimal implementation. Both the coefficient precision and the internal data paths are simultaneously optimized, allowing the design to have any desired precision.

The proposed architecture is shown in Fig. 2.20. A 1-D DWT module is in charge of performing both the rDWT and cDWT. The dual-port buffer stores the original raw data, intermediate data and the final results. The data path of the 1-D DWT module is shown in Fig. 2.21. For every internal signal x , the number of integer bits (IB), fractional bits (FB) and the total number of bits are denoted by IB_x , FB_x and B_x , respectively, where $IB_x + FB_x = B_x$. The IBs and FBs of all the signals of the 1-D DWT module are all optimized to reduce the arithmetic resources and improve the energy efficiency.

Fig. 2.20 High-level architecture of the 2-D DWT architecture [84]

Fig. 2.21 Data path of the 9/7 1-D DWT module [84]

For IB optimization, Lee *et al.* employed the approached proposed in [88], which is based on computing the roots of the derivatives of every signal.

The fractional bit optimization is composed of two steps.

- Static optimization: An analytical noise model is built and the worst case error expression of each level of DWT output is derived. The worst case error of the DWT output is bounded by the precision requirement of each level. Then, by applying the adaptive simulated annealing method [89], the fractional bit of every internal signal can be determined.
- Dynamic optimization: Because the worst case error is employed in static optimization, the result is conservative. By employing binary search, the fractional bits are uniformly reduced until the peak signal-to-noise ratio (PSNR) loss of a set of test images is above

TABLE 2.3 INTEGER BIT AND FACTIONAL BIT FOR TWO-LEVEL DWT [84]

Signal	<i>IB</i>	<i>IB</i>	<i>IB</i>	<i>IB</i>	<i>FB</i>
	Level 1 Row	Level 1 Column	Level 2 Row	Level 2 Column	
C_0	1	1	1	1	15
C_1	1	1	1	1	18
C_2	1	1	1	1	14
C_3	1	1	1	1	19
C_4	3	3	3	3	12
C_5	3	3	3	3	12
$x_i / \hat{x}_i$	0 (0)	1 (1)	2 (1)	3 (2)	8 or 19
D_0	2 (0)	3 (0)	4 (1)	5 (1)	16
D_1	0 (0)	1 (1)	2 (1)	3 (2)	18
D_2	2 (1)	3 (2)	4 (2)	5 (3)	16
D_3	2 (2)	3 (2)	4 (3)	5 (3)	15
D_4	0 (-1)	1 (-1)	2 (0)	3 (0)	18
D_5	1 (1)	2 (1)	3 (2)	4 (2)	18
D_6	1 (1)	2 (2)	3 (2)	4 (3)	18
D_7	1 (1)	2 (1)	3 (2)	4 (2)	18
D_8	0 (0)	1 (0)	2 (1)	3 (1)	16
D_9	0 (0)	1 (0)	2 (1)	3 (1)	15
D_{10}	0 (0)	1 (0)	2 (1)	3 (1)	16
D_{11}	0 (0)	1 (0)	2 (1)	3 (1)	16
$s_i / \hat{s}_i$	1 (1)	2 (1)	3 (2)	4 (2)	19
$d_i / \hat{d}_i$	1 (1)	2 (1)	3 (2)	4 (2)	19

0.1 dB or the unit in the last place (ulp) error of any of the DWT output exceeds 2 ulp.

The dynamic optimization can further reduce the fractional bits by 20% approximately.

The optimized integer bits and fractional bits of the 1-D DWT module are listed in the TABLE 2.3. The integer bits in the brackets exclude the extra bits due to binary point alignment.

Chapter 3. Memory-Efficient Single-Level 2-D DWT Architecture

3.1 Overview

In this chapter, a novel parallel stripe-based data scanning method is presented, which enables the trade-off between the external memory bandwidth and the internal buffer size. With the newly developed data scanning method, a novel memory-efficient parallel 2-D DWT architecture is proposed, which only requires an on-chip memory of size $3N + 2S$ (transposition memory and temporal memory). Based on the flipped data flow graph (DFG) proposed by Huang *et al.* [30], a regular operation unit (Cell) is developed as the building block of the architecture. As a result, the critical path delay (CPD) is reduced to $T_m + T_a$. The proposed architecture also has constant latency, and its hardware resource requirement and computation time are comparable with those of the best existing designs.

With the observation that the temporal memory for storing the three partial results in DWT is large, area-dominating and not optimized, wordlength optimization is performed on the proposed architecture with priority given to memory size reduction over data path reduction. While keeping the proposed architecture unchanged, a new objective function with larger weight on the fractional bit (FB) of the three partial results is proposed to optimize their fractional bit and integer bit (IB) before they are stored. As a result, not only is the memory wordlength reduced, resulting in smaller overall memory size, the arithmetic resource and the critical path delay are also reduced due to the optimized data path. Consequently, the total design area is reduced and the area-delay product (ADP) improved without compromising the precision of the final outputs.

3.2 Lifting Scheme and Flipping Structure

The lifting method is based on a series of matrix decomposition [70]. Huang *et al.* introduced a flipping method to reduce the critical path delay [30]. By flipping the DFG of the lifting scheme, the multipliers in the longest delay path are eliminated, resulting in a shorter critical path. The flipping 1-D DWT formulation is as follows [30]:

$$d_h(n) = \alpha^{-1}x(2n+1) + x(2n) + x(2n+2) \quad (3.1)$$

$$d_l(n) = \beta^{-1}x(2n) + d_h(n-1) + d_h(n) \quad (3.2)$$

$$H(n) = \gamma^{-1}d_h(n) + d_l(n-1) + d_l(n) \quad (3.3)$$

$$L(n) = \delta^{-1}d_l(n) + H(n-1) + H(n) \quad (3.4)$$

where α^{-1} , β^{-1} , γ^{-1} and δ^{-1} are the flipped lifting coefficients and $x(n)$ is the n^{th} input. 2-D DWT can be achieved with separate rDWT and cDWT.

3.3 Proposed Architecture

3.3.1 Data Scanning Method

In 2-D DWT architecture design, the size of the line buffer can be reduced by trading-off the external memory bandwidth and the internal buffer size. In [31], Huang *et al.* introduced a stripe-based scanning method for efficient bandwidth-memory trade-off. However, the memory reduction is achieved at the expense of longer computation time. In this chapter, the stripe-based scanning approach is modified and a new parallel lifting-based 2-D DWT architecture is proposed. The modified data scanning approach has two major differences from the existing one as described below, whereas the newly proposed architecture is presented in the following sections.

Let matrix A be the image of size $N \times (N+1)$ with one column of zeros padded as the first column. Fig. 3.1(a) shows the image matrix with each small square representing a pixel and the black squares representing the overlapped pixels between two stripes. For S parallel processing units, a stripe width of $2S+1$ is employed, with only one column overlapped between two adjacent stripes, instead of 8 overlapped columns for the lifting 9/7 filter in the approach by Huang *et al* [31]. As such, the memory size is reduced without increasing the computation time. Secondly, a new data scanning sequence is proposed. In the proposed approach, the rDWT is scheduled before the cDWT. During the first cycle of the rDWT, all the pixels from $A(0,0)$ to $A(0,2S)$ of the first row are fed in concurrently. The scanning continues in the top-down direction until $A(N,0)$ to $A(N,2S)$ are fed in. Subsequently, it continues from the first row of the next stripe, $A(0,2S)$ to $A(0,4S)$. For $N = R \times 2S$, there are R stripes in total. With the proposed scanning sequence, the intermediate matrices H and L are produced by the rDWT in the sequence as shown in the upper part of Fig. 3.1(b). The pixels in the overlapped columns, i.e., $A(n,2rS)$ for

Fig. 3.1 Stripe-based scanning method

$1 \leq r \leq R-1$ and $0 \leq n \leq N-1$, are needed twice. By reading them twice, a temporal buffer of size N is saved at the expense of reading only one additional pixel each clock cycle without increasing the computation time. The column processor requires two successive intermediate data from matrix H or L in column order every clock cycle. However, the row processor produces only one pair of intermediate data of H and L per cycle. Thus, the intermediate matrices H and L are cached and transposed before being alternately input to the cDWT, as shown in the lower part of Fig. 3.1(b). The proposed scanning method offers one additional advantage. It allows the column processing to start as soon as the intermediate results of the row processor are generated, resulting in a constant latency that is independent of N and S . On the other hand, an additional input buffer is required to store the overlapped column. The adopted input buffer is similar to the one proposed by Mohanty and Meher in Fig. 8 of [24]. But instead of fourteen, only two N -word single-port RAMs are required as shown in Fig. 3.2 by the dotted box. Every clock cycle, one pixel, $A(n, 2rS + 2S)$, of the overlapped column is written into the input buffer while one pixel is read from the input buffer. With such a buffer, there is no overhead of memory read/write cycles. There is an area overhead incurred by the additional buffer but it is insignificant compared to the saving of the temporal memory as discussed in Section 3.4.3.

Fig. 3.2 Structure of the input buffer

3.3.2 Data Processing Pipe (DPP)

The DFG of flipping lifting-scheme can be derived from (3.1) – (3.4) as shown in Fig. 3.3, where z^{-1} denotes a unit delay. The basic operation nodes are identical and consist of only addition and multiplication operations. Each node is implemented as a 3-input 2-output Cell (dotted box), consisting of 2 adders and a multiplier. The Cell has a critical path delay of $T_m + T_a$.

The data path (Fig. 3.3) is implemented with the Data Processing Pipe (DPP) that comprises four Cells, each with a flipped coefficient α^{-1} , β^{-1} , γ^{-1} and δ^{-1} respectively, as shown in Fig. 3.4. The DPP has three inputs x_0 , x_1 and x_2 , three partial result inputs p_α , p_β and p_γ , three partial result outputs p'_α , p'_β and p'_γ and a pair of low-pass and high-pass outputs L and H . By scheduling the correct pixel inputs, the partial results and intermediate results, several DPPs can be collocated to establish a 1-D parallel DWT processor as discussed in the next section.

Fig. 3.3 Data flow graph of flipped lifting scheme

Fig. 3.4 A data processing pipe comprising 4 cells

3.3.3 DWT Architecture

With the pixel inputs and the intermediate result inputs scheduled according to the scanning sequence proposed in Section 3.3.1, the Row-PU (Processing Unit) that implements the rDWT is constructed as shown in Fig. 3.5. As the stripe width is $2S+1$, there are $2S+1$ pixels fed into S DPPs ($S = 4$ in Fig. 3.5) every clock cycle. For the s^{th} DPP, the three input pixels x_0, x_1 and x_2 are $A(n, 2rS + 2s)$, $A(n, 2rS + 2s + 1)$ and $A(n, 2rS + 2s + 2)$ respectively, the three corresponding partial result outputs p_α, p_β and p_γ are $d_h(n, rS + s)$, $d_l(n, rS + s)$ and $H(n, rS + s)$, where $0 \leq s \leq S-1$, $0 \leq r \leq R-1$ and $0 \leq n \leq N-1$. The partial results generated by the s^{th} DPP ($0 \leq s \leq S-2$) are the partial result inputs to the $(s+1)^{th}$ DPP. Thus, the partial inputs p_α, p_β and p_γ for the s^{th} DPP are $d_h(n, rS + s - 1)$, $d_l(n, rS + s - 1)$ and $H(n, rS + s - 1)$. The partial result outputs $d_h(n, (r+1)S - 1)$, $d_l(n, (r+1)S - 1)$ and $H(n, (r+1)S - 1)$ of the last DPP ($s = S - 1$) are cached for N cycles to be used by the first DPP for the next stripe. As such, a temporal memory of size $3N$ is required to cache 3 partial results generated by the last DPP in each of the N cycles. The temporal memory is implemented by 3

Fig. 3.5 The structure of Row-PU ($S = 4$)

shift registers (SR in Fig. 3.5) of length N . Within the Row-PU, each single DPP gets the partial result inputs from their preceding DPPs or the shift registers (for $s = 0$ DPP). The intermediate result outputs of the Row-PU are fed into the transposition registers. One high-pass output (H) and one low-pass output (L) are generated in each pipe of the Row-PU every clock cycle. These interleaved outputs are fed into the transposition registers before the Col-PU. The transposition register alternately outputs two high-pass and low-pass data as shown in Fig. 3.6, where the dotted inputs have appeared at the output. Compared to those in the existing architectures with the same throughput, this transposition register is the smallest and depends on only S but not N (see TABLE 3.2).

As the intermediate outputs H and L are fed in an alternative order, a pair of H and L is processed in one DPP of the Col-PU in an alternative manner. The partial results produced are consumed by the following Cell two clock cycles later. As such, shift registers of length two are needed within the Col-PU between each lifting step for caching the partial results. At the output of the Col-PU, the four subbands are generated in an interleaved pattern, i.e., (HL, HH) , (LL, LH) , (HL, HH) , (LL, LH) and so on. The structure of the Col-PU is shown in Fig. 3.7. As the Col-PU is able to process the intermediate outputs of the Row-PU as soon as they are produced, the proposed architecture has a constant latency of 9 clock cycles, regardless of image size N and core size S . Each subband is scaled in the scaling unit shown in Fig. 3.8, where K_1 , K_2 and K_3 are the scaling coefficients. Fig. 3.9 shows the overall architecture of the proposed single level 2-D DWT processor. For multi-level decomposition, the same DWT core can be used in a folded architecture with an external frame buffer for the LL subband coefficients.

Fig. 3.6 The transposition register

Fig. 3.7 The structure of Col-PU ($S = 4$)

Fig. 3.8 Scaling unit

Fig. 3.9 Overall architecture

3.4 Hardware Estimation and Performance Comparison

3.4.1 Hardware Estimation

The hardware hierarchical tree of the proposed DWT architecture is shown in Fig. 3.10, with the number of units in the brackets. In total, the architecture has $3N$ temporal buffer, $14S$ pipeline registers, $8S$ registers for partial results, $2S$ transposition registers, $10S$ multipliers and $16S$ adders.

3.4.2 Comparison

TABLE 3.2 shows the comparison of the existing architectures and the proposed architecture for 2-D DWT in terms of multipliers, adders, on-chip memories, critical path delay, computation time and throughput. Compared with the design in [36], which achieves the least memory consumption among the existing architectures, the proposed design uses the same number of multipliers and adders with N less temporal buffers and 14 less registers for $S = 1$, and has the same computation time. Among the existing parallel architectures [29], [34] and [37], the architecture in [29] is the most memory efficient design. Compared with [29], the proposed architecture has $2S$ instead of N transposition buffers, where $S \ll N$ as N is usually large (>256) and S small (<64) in practical applications. Thus, it is evident that the proposed architecture consumes the least total memory. The architectures in [32] and [36] have the shortest critical path delay of only T_m . However, the short critical path is achieved at the expense of large number of pipeline registers. The proposed structure has a critical path delay of $T_m + T_a$, which is shorter than the FA (Fast Architecture) and HA (High-speed Architecture) of [35] and all the

Fig. 3.10 Hardware resource tree

parallel architectures [29], [34], [37]. On the other hand, when compared with [29] and [34], the proposed design consumes S more multipliers, which is the cost for a shorter critical path.

3.4.3 Implementation

The best existing parallel 2-D DWT architecture [29] and the proposed design with different S are model and verified in structural Verilog HDL with the input pixel width of 8 bits, datapath width of 16 bits for image size of 512×512 . One design each for $S = 2$, $S = 4$ and $S = 8$ was synthesized in the STM 90 nm CMOS process with Synopsys Design Compiler. The synthesized results are tabulated in TABLE 3.2, where the improvements ($\Delta\%$) over the counterpart designs in [29] are also indicated. The throughput (TP) is calculated from the image size, total computation cycles and critical path delay (CPD), and the equivalent gate count from the ratio of the total area and the area of one 2-input NAND gate of $6.59 \mu\text{m}^2$. From TABLE 3.2, the proposed designs have shorter critical path delay, incur less core area cost, achieve better area-delay product (ADP) and higher throughput and consume less power than the three designs in [29], respectively.

It should be noted that the area and power consumption in TABLE 3.1 do not account for the input buffer (Fig. 3.2) of size 8K bits (2×512 8-bit words) required for the overlapped pixels in the proposed designs. The buffer is conventionally implemented with off-chip DRAMs and hence, incurs a memory bandwidth overhead of 8 bits (one pixel). With the estimation approach suggested in [24] for DRAM size at $138671.9 \mu\text{m}^2$ for 1 Megabits in 90 nm CMOS process, the size of the 8K-bit buffer is negligible compared to the core area of the designs. If the additional buffer is implemented with two 512-word single-port SRAMs instead, it will incur an area cost of $58482 \mu\text{m}^2$ and consume 1.12 mW of power according to the IBM-ARM SRAM generator for 90 nm CMOS process. Even with this overhead included, the proposed designs still have ADP advantage of 26%, 26% and 22% as well as power advantage of 14%, 12% and 10% when compared to the designs in [29] for $S = 2, 4$ and 8 respectively.

It is observed from TABLE 3.2 that the ADP advantage of the proposed design is lesser for larger S and vanishes when $S > N/4$. However, N is usually much larger than S and for large S (≥ 16), the external memory bandwidth and the overall size of the design become impractically large. Note that no low power design technique was employed in any of the

designs and the total power is obtained at a supply voltage of 1.2 V and an operating frequency of 20 MHz.

TABLE 3.1 COMPARISON BETWEEN PROPOSED ARCHITECTURE AND EXISTING ARCHITECTURES IN HARDWARE RESOURCES AND TIME COMPLEXITY (FOR 1 LEVEL 9/7 LIFTING-BASED DWT)

Architecture	Multiplier	Adder	On-chip memory			Critical path delay	Computation Cycles	Throughput
			Register	Trans. buffer	Temporal buffer			
Lai <i>et al.</i> [32]	10	16	44	4	$4N$	T_m	$N^2/2$	$2/T_m$
Xiong <i>et al.</i> [35] (HA)	18	32	52	4	$5.5N$	T_m+2T_a	$N^2/4$	$4/(T_m+2T_a)$
Xiong <i>et al.</i> [35] (FA)	10	16	38	4	$5.5N$	T_m+2T_a	$N^2/2$	$2/(T_m+2T_a)$
Zhang <i>et al.</i> [36]	10	16	34	3	$4N$	T_m	$N^2/2$	$2/T_m$
Tian <i>et al.</i> [37]	10S	16S	28S	$NS+N$	$3N$	T_m+2T_a	$N^2/2S$	$2S/(T_m+2T_a)$
Mohanty <i>et al.</i> [34]	9S	16S	15S	$2.5N$	$3N$	T_m+2T_a	$N^2/2S$	$2S/(T_m+2T_a)$
Mohanty <i>et al.</i> [29]	9S	16S	20S	N	$3N$	T_m+2T_a	$N^2/2S$	$2S/(T_m+2T_a)$
This work	10S	16S	22S	2S	$3N$	T_m+T_a	$N^2/2S$	$2S/(T_m+T_a)$

TABLE 3.2 COMPARISON OF SYNTHESIS RESULTS FOR IMAGE SIZE 512×512

Designs	S	Area (μm^2) ($\Delta\%$)	CPD (ns) ($\Delta\%$)	ADP ($\mu\text{m}^2\text{s}$) ($\Delta\%$)	Power (mW) ($\Delta\%$)	TP (pix./ns) ($\Delta\%$)	Gate Count ($\Delta\%$)
Mohanty <i>et al.</i> [29]	2	824913	3.66	198	12.2	1.09	125177
	4	899841	3.67	108	13.1	2.18	136546
	8	1049596	3.67	63	15.0	4.36	159271
This work	2	644900 (21.8)	3.17 (13.4)	134 (32.3)	9.4 (23.0)	1.26 (15.6)	97860 (21.8)
	4	729047 (19.0)	3.10 (15.5)	74 (31.5)	10.4 (20.6)	2.58 (18.3)	110629 (19.0)
	8	897365 (14.5)	3.10 (15.5)	46 (27.0)	12.4 (17.3)	5.16 (18.3)	136171 (14.5)

3.5 Memory-Efficient 2-D DWT Architecture Based on Precision-Aware Bit Width Optimization

Although the existing methods [29], [77], including the above proposed DWT architecture, achieve memory reduction to some extent, they reduce only the number of memory words (i.e. memory length) and assume a fixed data path width. Moreover, it becomes harder to further reduce the memory length without compromising the external bandwidth and increasing on-chip processing units. Optimizing the memory wordlength can be an alternative as the total memory size is the product of the number of words and their wordlength. By carefully tuning the wordlength of the partial results that need to be stored on-chip, the total memory size could be reduced. In this section, the memory-efficient 2-D DWT architecture presented in the earlier sections is optimized for bit width to further reduce its memory size without compromising the precision of the results.

3.5.1 Flipping Scheme with Variable Coefficients

Instead of flipping with the inverse of the lifting coefficients [30], the flipping coefficients could be the inverse of the lifting coefficients multiplied by 2^k , with an integer k . With variable flipping coefficients, the architecture could achieve better precision without increasing the datapath length. The flipping rDWT formulation is as below [30].

$$d_h(m, n) = C_0 x(m, 2n+1) + x(m, 2n) + x(m, 2n+2) \quad (3.5)$$

$$d_l(m, n) = C_1 x(m, 2n) + (d_h(m, n-1) + d_h(m, n)) / 2^4 \quad (3.6)$$

$$H(m, n) = C_2 d_h(m, n) + (d_l(m, n-1) + d_l(m, n)) / 2 \quad (3.7)$$

$$L(m, n) = C_3 d_l(m, n) + (H(m, n-1) + H(m, n)) / 2 \quad (3.8)$$

where C_0 , C_1 , C_2 and C_3 are the flipped lifting coefficients and $x(m, n)$ is the n^{th} pixel of the m^{th} row of the input image. For 9/7 DWT, $C_0 = -0.6304636206$, $C_1 = 0.7437502472$, $C_2 = -0.6680671710$ and $C_3 = 0.6384438531$. The cDWT has the similar formulation. The four resultant subbands HH , HL , LH and LL are scaled before being output. The scaling coefficients $C_4 = 4.2652337873$ and $C_5 = 5.8613434184$ scale HH and LL , respectively, while

LH and HL are scaled by $C_6 = 4.9999999987$. The integer bit width and the fractional bit width of the coefficients are optimized as described in the following sections.

3.5.2 Memory-Efficient 2-D DWT Architecture

The wordlength optimization is applied on the proposed memory-efficient 2-D DWT architecture presented in Sections 3.2 – 3.4. The row-wise and the column-wise processing unit (Row-PU and Col-PU) of the architecture are shown in details in Fig. 3.11 and Fig. 3.12 respectively. The input image x is first processed by the Row-PU where the intermediate results H and L are produced and then transposed in the transpose memory. During the row-wise processing, three partial results are produced and stored into the temporal memory while three partial results stored earlier in the temporal memory are read as indicated by the dotted arrows in Fig. 3.11. The temporal memory is large and dominates the chip area. After the transposition, H and L are fed into the Col-PU as shown in Fig. 3.12 to generate four subband outputs, namely HH , HL , LH and LL , which are then scaled in the scaling unit (SU) to produce the final results. The architecture has a scalable throughput of $2S$ when employing S parallel data processing pipes (DPPs), each consisting of 4 connected Cells (dotted boxes in Fig. 3.11 and Fig. 3.12) in the Row-PU and Col-PU.

While keeping the high-level structure unchanged, the wordlength of every signal, each consisting of an integer bit and a fractional bit, in the DPP is optimized for memory efficiency. The integer bit is determined by dynamic range analysis while the fractional bit is determined by fractional bit optimization as described below.

Fig. 3.11 Structure of Row-PU, temporal memory and transpose memory

Fig. 3.12 Structure of Col-PU and SU

3.5.3 Integer Bit Optimization Based on Dynamic Range Analysis

In [87], only the dynamic range of the internal signals in the 1-D DWT architecture is analyzed. In the proposed optimization, the dynamic range of every internal signal in the proposed parallel 2-D DWT architecture is obtained and the integer bit width of these signals is derived as described next. In the analysis, each internal signal d_i ($0 \leq i \leq 11$) of the Row-PU is first expressed by the linear combinations of the input pixels $x(m, n)$ in the form of $d_i(m, n) = \sum h_k x(m_k, n_k)$, where h_k is a constant. For example, in the Row-PU $d_0(m, n) = C_0 x(m, 2n+1)$ and $d_1(m, n) = x(m, 2n) + x(m, 2n+2)$ are expressed in terms of the input pixels. Since $d_2(m, n) = d_0(m, n) + d_1(m, n)$, its linear combinations can be represented as $d_2(m, n) = C_0 x(m, 2n+1) + x(m, 2n) + x(m, 2n+2)$. i.e. $d_2(m, n) = \sum_{k=0}^2 h_k x(m_k, n_k)$, where $h_0 = C_0$, $h_1 = h_2 = 1$, $m_0 = m_1 = m_2 = m$, $n_0 = 2n+1$, $n_1 = 2n$ and $n_2 = 2n+2$. As another example, $d_4(m, n) = (d_2(m, n-1) + d_2(m, n)) / 16 = (x(m, 2n-2) + C_0 x(m, 2n-1) + 2x(m, 2n) + C_0 x(m, 2n+1) + x(m, 2n+2)) / 16$. After the linear combinations of all the other internal signals are derived in a similar way, the maximal and minimal value of the signal d_s , $\max(d_s)$ and $\min(d_s)$, can be obtained. Assume the input pixels $x(m, n) \in [-0.5, 0.5)$ are represented with 8-bit 2's complement numbers. $\max(d_s)$ is obtained when $d_s(m, n) = \sum h_k x(m_k, n_k)$ is computed with:

$$x(m_k, n_k) = \begin{cases} -128 / 256, & h_k < 0 \\ +127 / 256, & h_k \geq 0 \end{cases}$$

and $\min(d_s)$ is obtained with

$$x(m_k, n_k) = \begin{cases} -128 / 256, & h_k \geq 0 \\ +127 / 256, & h_k < 0 \end{cases}.$$

Hence, the integer bit width of the signal d_s can be calculated with $IB_s = \lceil \log_2(\max(d_s) - \min(d_s)) \rceil$. For the Row-PU, the internal signals d_i are expressed in their linear combinations of the input pixels. But for the internal signals sh_i and sl_i ($0 \leq i \leq 11$) in the Col-PU, their linear combinations are in terms of its inputs ($L_{\text{even}} / H_{\text{even}}$ and $L_{\text{odd}} / H_{\text{odd}}$).

$\max(sh_i)$ (or $\max(sl_i)$) and $\min(sh_i)$ (or $\min(sl_i)$) are computed from the maximal and minimal values of L_{even} / H_{even} and L_{odd} / H_{odd} . As sh_i and sl_i share the same data path, the integer bit of the signals is $IB_{s_i} = \max(IB_{sh_i}, IB_{sl_i})$. With the above dynamic range analysis, the integer bits of all the internal signals, the coefficients, the inputs and outputs are obtained as shown in TABLE 3.3, where the ranges of the signals are also listed. As examples, the integer bits of d_0 and s_2 are 0 and 2 bits respectively as $d_0 \in [0.302, -0.312]$ and $s_2 \in [0.120, -1.168]$ resulted from the analysis.

TABLE 3.3 INTEGER BIT WIDTH OF SIGNALS (BITS)

	Signals	LL	LH	HL	HH	C_6	C_5	C_4	C_3	C_2	C_1	C_0	x
Constants, inputs & outputs	Min	-1.905	-1.791	-1.625	-1.528	5	5.861	4.265	0.638	-0.668	0.743	-0.630	-0.5
	Max	1.826	1.716	1.557	1.464	5	5.861	4.265	0.638	-0.668	0.743	-0.630	0.479
	IBs	2	2	2	2	4	4	4	1	1	1	1	0
	Signals	L_{odd}/H_{odd}	L_{even}/H_{even}	s_{11}	s_{10}	s_9	s_8	s_7	s_6	s_5	s_4	s_3	s_2
Col-PU	Max	-0.444	-0.444	-0.358	-0.349	-0.304	-0.394	-0.476	-0.780	-0.476	-0.146	-0.330	-1.168
	Min	0.425	0.425	0.343	0.335	0.291	0.378	0.456	0.748	0.456	0.14	0.316	1.120
	IBs	0	0	0	0	0	0	0	1	0	-1	0	2
	Signals	H	L	d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2
Row-PU	Min	-0.444	-0.403	-0.403	-0.393	-0.342	-0.444	-0.536	-0.878	-0.536	-0.164	-0.371	-1.312
	Max	0.425	0.386	0.386	0.377	0.328	0.425	0.514	0.842	0.514	0.157	0.356	1.260
	IBs	0	0	0	0	0	0	1	1	1	-1	0	2
	Signals												

3.5.4 Fractional Bit Optimization Based on Noise Modeling

In order to maintain the precision of the DWT outputs, the error must be less than 2-ulp (unit in the last place) error [87], [90]. For an input signal of 8-bit precision, the fractional bit of the four output subbands must be 8 bits, i.e. the ulp is 2^{-8} . Consequently, the maximum error for the subbands must be less than 2^{-7} . In this design, the fractional bits of the outputs are round off by truncating the bits that are smaller than 2^{-T} when T -bit precision is to be maintained ($T = 8$ in this case). The round-off error can be modeled by the following functions f_{add} and f_{mul} , respectively for the addition and multiplication operations [87].

$$E_c = f_{add}(a, b, c) = E_a + E_b + \max(0, 2^{-FB_c} - 2^{-\max(FB_a, FB_b)}) \quad (3.9)$$

$$E_c = f_{mul}(a, b, c) = \max(b)E_a + \max(a)E_b + E_a E_b + \max(0, 2^{-FB_c} - 2^{-\max(FB_a, FB_b)}) \quad (3.10)$$

where a and b are the inputs and c the output of the adder / multiplier. E_s and FB_s are respectively the maximum error and the fractional bit of a signal s . The error of an input image pixel $x(m, n)$ is 0, and the error of a coefficient C_i is $2^{-FB_{C_i}}$.

Based on the accuracy models of (3.9) and (3.10) and the DWT architecture in Fig. 3.11 and Fig. 3.12, the maximum error models of the signals can be constructed as given below. For Row-PU:

$$E_{d_0} = f_{mul}(x_{odd}, C_0, d_0) \quad (3.11)$$

$$E_{d_1} = f_{add}(x_{even}, x_{even}, d_1) \quad (3.12)$$

$$E_{d_2} = f_{add}(d_0, d_1, d_2) \quad (3.13)$$

$$E_{d_3} = f_{mul}(x_{even}, C_1, d_3) \quad (3.14)$$

$$E_{d_4} = f_{mul}(f_{add}(d_2, d_2, d'_4), 2^{-4}, d_4) \quad (3.15)$$

$$E_{d_5} = f_{add}(d_3, d_4, d_5) \quad (3.16)$$

$$E_{d_6} = f_{mul}(d_2, C_2, d_6) \quad (3.17)$$

$$E_{d_7} = f_{mul}(f_{add}(d_5, d_5, d'_7), 2^{-1}, d_7) \quad (3.18)$$

$$E_{d_8} = f_{add}(d_6, d_7, d_8) \quad (3.19)$$

$$E_{d_9} = f_{mul}(d_5, C_3, d_9) \quad (3.20)$$

$$E_{d_{10}} = f_{mul}(f_{add}(d_8, d_8, d'_{10}), 2^{-1}, d_{10}) \quad (3.21)$$

$$E_{d_{11}} = f_{add}(d_9, d_{10}, d_{11}) \quad (3.22)$$

d_8 and d_{11} are generated by the Row-PU and fed into the Col-PU as the intermediate results H and L , respectively. As H and L share the same data path in the Col-PU, their fractional bit widths must satisfy both the signal sets sh_i and sl_i . The maximum error models for the signal sets H and L in the Col-PU are given below.

$$E_{sh_0} = f_{mul}(d_8, C_0, sh_0) \quad (3.23)$$

$$E_{sh_1} = f_{add}(d_8, d_8, sh_1) \quad (3.24)$$

$$E_{sh_2} = f_{add}(sh_0, sh_1, sh_2) \quad (3.25)$$

$$E_{sh_3} = f_{mul}(d_8, C_1, sh_3) \quad (3.26)$$

$$E_{sh_4} = f_{mul}(f_{add}(sh_2, sh_2, sh'_4), 2^{-4}, sh_4) \quad (3.27)$$

$$E_{sh_5} = f_{add}(sh_3, sh_4, sh_5) \quad (3.28)$$

$$E_{sh_6} = f_{mul}(sh_5, C_2, sh_6) \quad (3.29)$$

$$E_{sh_7} = f_{mul}(f_{add}(sh_5, sh_5, sh'_7), 2^{-1}, sh_7) \quad (3.30)$$

$$E_{sh_8} = f_{add}(sh_6, sh_7, sh_8) \quad (3.31)$$

$$E_{sh_9} = f_{mul}(sh_8, C_3, sh_9) \quad (3.32)$$

$$E_{sh_{10}} = f_{mul}(f_{add}(sh_8, sh_8, sh'_{10}), 2^{-1}, sh_{10}) \quad (3.33)$$

$$E_{sh_{11}} = f_{add}(sh_9, sh_{10}, sh_{11}) \quad (3.34)$$

$$E_{sl_0} = f_{mul}(d_{11}, C_0, sl_0) \quad (3.35)$$

$$E_{sl_1} = f_{add}(d_{11}, d_{11}, sl_1) \quad (3.36)$$

$$E_{sl_2} = f_{add}(sl_0, sl_1, sl_2) \quad (3.37)$$

$$E_{sl_3} = f_{mul}(d_{11}, C_1, sl_3) \quad (3.38)$$

$$E_{sl_4} = f_{mul}(f_{add}(sl_2, sl_2, sl'_4), 2^{-4}, sl_4) \quad (3.39)$$

$$E_{sl_5} = f_{add}(sl_3, sl_4, sl_5) \quad (3.40)$$

$$E_{sl_6} = f_{mul}(sl_5, C_2, sl_6) \quad (3.41)$$

$$E_{sl_7} = f_{mul}(f_{add}(sl_5, sl_5, sl'_7), 2^{-1}, sl_7) \quad (3.42)$$

$$E_{sl_8} = f_{add}(sl_6, sl_7, sl_8) \quad (3.43)$$

$$E_{sl_9} = f_{mul}(sl_5, C_3, sl_9) \quad (3.44)$$

$$E_{sl_{10}} = f_{mul}(f_{add}(sl_8, sl_8, sl'_{10}), 2^{-1}, sl_{10}) \quad (3.45)$$

$$E_{sl_{11}} = f_{add}(sl_9, sl_{10}, sl_{11}) \quad (3.46)$$

The sh_8 , sh_{11} , sl_8 and sl_{11} are scaled before output as HH , HL , LH and LL , respectively. Thus, the error model of SU can be constructed as follows.

$$E_{HH} = f_{mul}(sh_8, C_4, HH) \quad (3.47)$$

$$E_{HL} = f_{mul}(sh_{11}, C_6, HL) \quad (3.48)$$

$$E_{LH} = f_{mul}(sl_8, C_6, LH) \quad (3.49)$$

$$E_{LL} = f_{mul}(sl_{11}, C_5, LL) \quad (3.50)$$

In order to satisfy the precision requirement of the output subbands, the constraint (3.51) must be satisfied.

$$\max(E_{HH}, E_{HL}, E_{LH}, E_{LL}) < 2^{-7} \quad (3.51)$$

In addition to (3.51), an object function is required to optimize the fractional bits of the internal signals. As the goal is to optimize the fractional bits for achieving the smallest area, the object function is the area estimated based on the fractional bits. Accurate estimation of the area directly from the fractional bits is not straightforward as there are many other factors, such as HDL coding style, cell library used, synthesis constraints, etc., that will impact the resultant area. Thus, instead of trying to establish an accurate area model, a coarse-grain method is proposed to establish the object function by differentiating the most area-dominating signals from the other signals as elaborated below.

The size of the area-dominating temporal memory is determined by the wordlength of the signals d_2 , d_5 and d_8 (dotted arrows in Fig. 3.11). Thus, compared to the fractional bit width of the other signals, their fractional bit widths have the most significant impact on the area. Based on this observation, the object function to be optimized is formulated as (3.52).

$$f_{obj} = p(FB_{d_2} + FB_{d_5} + FB_{d_8}) + q(\sum_{i=0}^{11} FB_{d_i} + \sum_{i=0}^{11} FB_{s_i} + \sum_{i=0}^6 FB_{C_i}) \quad (3.52)$$

where s_i is either sh_i or sl_i , p is the weight for d_2 , d_5 , d_8 and q for the other signals. p and q are adjustable according to different throughput S , image size, technology process, etc. To determine the appropriate weight values of p and q for practical designs, the proposed DWT architecture is synthesized with full precision (4-bit integer bit width for coefficients, 2-bit integer bit width for internal signals and 17-bit fractional bit width for both coefficients and internal signals) for various image sizes and different degrees of parallelism in the STM 90-nm CMOS process with Synopsys Design Compiler. The results are tabulated in TABLE 3.4. The memory-arithmetic resource area ratio (R) varies with throughputs and image sizes. Accordingly, the ratio is employed as the weight ratio, i.e. $p/q = R$ to estimate the influence of the fractional bit width on the area. With the constraint (3.51) and the object function (3.52), the fractional bit

TABLE 3.4 SYNTHESIZED AREA FOR ARITHMETIC RESOURCES AND TEMPORAL MEMORY

Image size	S	DAT (ns)	Area (μm^2)		Memory to Arithmetic Area Ratio (R)
			Arithmetic resources	Temporal memory	
512×512	2	4.67	111498	664488	5.96
	4	4.87	223516	664488	2.97
	8	4.90	447419	664488	1.49
1024×1024	2	4.76	111499	1328969	11.92
	4	4.92	223499	1328969	5.95
	8	5.01	447412	1328969	2.97

TABLE 3.5 FRACTIONAL BIT WIDTH OF SIGNALS (BITS)

Row-PU	d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}	L	H
	14	8	14	19	19	15	17	17	15	18	18	18	18	15
Col-PU	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	$L_{\text{even}}/H_{\text{even}}$	$L_{\text{odd}}/H_{\text{odd}}$
	16	17	17	17	17	16	17	16	17	17	17	15	18	18
Constants,	x	C_0	C_1	C_2	C_3	C_4	C_5	C_6	HH	HL	LH	LL		
Inputs & Outputs	8	18	19	19	18	8	12	9	8	8	8	8		

Algorithm 1 Simulated Annealing (SA)

```
Select an initial state  $i \in S$  ;
Select an initial temperature  $T > 0$  ;
Set temperature change counter  $t = 0$  ;
Repeat
    Set repetition counter  $n = 0$  ;
    Repeat
        Generate state  $j$ , a neighbor of  $i$ ;
        Calculate  $\delta = f(j) - f(i)$  ;
        if  $\delta < 0$  then  $i = j$ 
        else if  $\text{random}(0,1) < \exp(-\delta/T)$  then  $i = j$ ;
         $n = n + 1$  ;
    until  $n = N(t)$  ;
     $t = t + 1$  ;
     $T = T(t)$  ;
until stopping criterion true
```

widths of the internal signals can be optimized by global optimization algorithms such the genetic algorithm and the simulated annealing (SA) [89], [91]. In this work, the simulated annealing is chosen because of its ability of locating a good approximation of the global optimum in a large discrete search space [92]. The pseudo code of the SA algorithm is shown in Algorithm 1 [93]. The initial state and initial temperature are determined empirically. The search terminates when the temperature drops below the minimum temperature or 500 new solutions are rejected successively. TABLE 3.5 shows one set of optimized fractional bit widths of the internal signals with $p/q = 11.92$ for an image size of 1024×1024 and a throughput with $S = 2$.

3.6 Hardware Estimation and Performance Comparison

From TABLE 3.4 and TABLE 3.5, the overall bit width of d_2 , d_5 and d_8 can be calculated as $\sum_{i=2,5,8} (IB_{d_i} + FB_{d_i}) = 47$. On the other hand, should constant wordlength be applied to the architecture, the fractional bit of all C_i , d_i and s_i would have been 17, the integer bit of C_i 4 and the integer bit of d_i and s_i 2, in order to meet the precision constraint (3.51). Consequently,

the overall bit width of d_2 , d_5 and d_8 would have been $(2+17) \times 3 = 57$. As the length of the temporal memory remains the same, its size is reduced by $((57-47)/57) \times 100\% = 17.54\%$. It is worth noting that after optimization, the total wordlength of the internal signals other than d_2 , d_5 and d_8 is also reduced. This reduction leads to a smaller area and shorter critical path length incurred by the arithmetic resources. The optimized designs are modeled in structural Verilog HDL and verified. The synthesis results of both the original (baseline) designs and the wordlength optimized designs for the input image sizes of 512×512 and 1024×1024 and $S = 2, 4$ and 8 are tabulated in TABLE 3.6. Fig. 3.13 and Fig. 3.14 show the comparison of the ADPs. Compared with the proposed architecture without wordlength optimization, the optimized architecture achieves the best ADP improvement of more than 25% for $S = 8$ and image size 1024×1024 .

3.7 Summary

In this chapter, a memory efficient 2-D parallel DWT architecture with regular structure and short critical path is proposed. A new stripe-based scanning method is introduced to achieve the trade-off between the external input bandwidth and the internal buffer size. Based on the stripe-based scanning method, a new row processing unit and a new column processing unit are developed. The proposed architecture achieves the smallest memory size of $3N + 24S$ compared to the existing architectures. It has constant latency and its throughput is scalable by scaling the number (S) of data processing pipes, in which a block of $2S$ pixels is processed every clock cycle. The synthesis results show that the designs based on the proposed architecture have the advantages of ADP, throughput and power consumption over the designs based on the best existing architecture [29] by 27.0 ~ 32.3%, 15.6 ~ 18.3 and 17.3 ~ 23.0% respectively for an image size of 512×512 the degree of parallelism of $S = 2, 4$ and 8 .

Based on the proposed architecture, a wordlength optimization approach is proposed to further improve the memory efficiency. While keeping the high-level architecture unchanged, the bit width of every internal signal is optimized, with priority given to the signals that impact the area-dominating on-chip memory size. Dynamic range analysis and architecture-specific noise modeling are applied to optimize the bit width. Compare with the designs without optimization, the optimized designs exhibit an ADP improvement of 19.2% ~ 25.1% for the input image sizes of 512×512 and 1024×1024 and the degree of parallelism of $S = 2, 4$ and 8 .

TABLE 3.6 SYNTHESIS RESULTS FOR PROPOSED ARCHITECTURE

Image size	S	DAT (ns)	Area (μm^2)		Total area (μm^2)
			Arithmetic resources	Temporal memory	
512×512	2	4.51	89285	559583	648868
	4	4.50	165254	571258	736512
	8	4.45	348816	571258	920074
1024×1024	2	4.58	90828	1102982	1193810
	4	4.64	180878	1119163	1300041
	8	4.50	361243	1119163	1480406

Fig. 3.13 Comparison of ADP for image size 512x512

Fig. 3.14 Comparison of ADP for image size 1024x1024

Chapter 4. Energy- and Area-Efficient Parameterized Lifting-Based 2-D DWT Architecture

4.1 Overview

The energy efficiency has emerged as one of the most important performance metrics in computing [17]. Although there are many existing DWT architectures, most of them focus on improving the hardware utilization and reducing the on-chip memory size and critical path length, while energy efficiency has not been considered as a key performance metric. In this chapter, an energy- and area-efficient parameterized lifting-based 2-D DWT architecture is proposed. Based on the observation that the external memory and on-chip memory dominates the total energy consumption, an overlapped block-based image scanning method is employed. As a result, the required number of external memory reads are significantly reduced without compromising the on-chip memory size. With the reduced external memory accesses, the energy consumption of the proposed architecture is significantly reduced. The proposed architecture is parameterized and achieves high memory efficiency. While the on-chip memory size of the existing designs depends on the input image width or height, the proposed architecture consumes only a constant number of on-chip memory.

In this chapter, the following contributions are made:

- An overlapped block-based image scanning method is proposed to improve the energy efficiency by optimizing the number of external memory reads and the on-chip memory size (Section 4.2.2).
- An energy-efficient parameterized DWT architecture is proposed for the lifting $5/3$ and $9/7$ filter (Section 4.2).
- A DRAM activation schedule is developed to reduce the energy consumption of the external memory (Section 4.2.4).
- The proposed designs achieve up to 269% and 181% improvement in energy efficiency compared with the state-of-the-art design for the $9/7$ and $5/3$ filters, respectively (Section 4.3).

4.2 Energy-Efficient 2-D DWT Architecture

4.2.1 Design Assumptions

DWT architecture: As shown in Fig. 4.1, the DWT architecture is composed of an image memory and a DWT processor. The input image of size $M \times N$ is stored in the external image memory, which is implemented by external DRAMs. The image memory outputs the image pixels to the DWT processor, which performs DWT to the input image. The DWT processor is composed of a functional unit and an on-chip memory.

Operations: There are two types of operations, namely computation operations and memory-access operations. A computation operation in this investigation is defined as just one multiplication in the DWT architecture and the additions are ignored because their energy consumption is insignificant compared to that of the multiplication. A memory operation is defined as a read/write operation to the image memory or the on-chip memory. For image of size $M \times N$, a minimum of $2MN$ and $4.5MN$ computation operations are required for the 5/3 and 9/7 filters, respectively. A minimum of MN image memory read operations are required for both the 5/3 and 9/7 filters.

Algorithm-mapping parameters: Two algorithm-mapping parameters are introduced to characterize the DWT architecture, namely the parallelism and the data scanning method.

- *Data scanning method:* There are many existing data scanning methods employed by various DWT architectures. With different data scanning method results, different on-chip memory sizes and external bandwidths are required. In this chapter, an overlapped

Fig. 4.1 High level abstraction of DWT architecture

block-based data scanning method is proposed for improving the energy efficiency. The height of the image blocks (R) is used as a parameter to explore the design space and find the energy-optimized design point.

- *Parallelism*: The parallelism is defined as the number of computation operations performed by the DWT architecture in one clock cycle.

Energy efficiency: Energy efficiency is defined as the number of operations that can be performed by a unit of energy (giga-operations per second per watt (GOPS/W)). It is a widely accepted performance metric for computing [17]. In this work, only the dynamic power is considered. In order to make a fair comparison among various architectures, when computing energy efficiency, it is assumed that for all designs, $2MN$ and $4.5MN$ computation operations are performed for each frame respectively for the $5/3$ and $9/7$ filters, which is the smallest number of operations required for transforming a frame. Let the minimum required computation operations per frame be C , the energy consumed by processing one frame be E , the average power of the DWT architecture be $\bar{P}$ and the processing time per frame be t . The energy efficiency η is obtained by:

$$\eta = C / E = C / \bar{P}t \quad (4.1)$$

Throughput: The number of image frames processed per second.

Energy×Area×Time (EAT): EAT is defined as the product of the energy consumption per frame, the area usage of the architecture and the computation time per frame. The EAT is introduced as a composite metric to evaluate the energy efficiency, area and throughput of various architectures.

Peak energy efficiency: The peak energy efficiency is introduced to evaluate if an architecture is well optimized with respect to energy efficiency. The energy efficiency of a specific target device is upper bounded by the peak energy efficiency no matter how the computations are performed. To compute the peak energy efficiency of an algorithm on a given target device, the energy consumed to perform the minimum number of required operations (e.g. computation and memory accesses) is considered while all the overheads such as I/O, control logic, routing and on-chip buffers that may be employed by an implementation are ignored. For 2-D DWT, assume the average power dissipation of the required computation and external memory on a given target device are P_c and P_i , respectively. The peak energy efficiency η_{peak} is obtained by:

$$\eta_{peak} = C / (P_c + P_i) t \quad (4.2)$$

The peak energy efficiency of the target device considered in this work is evaluated and presented in Section 4.3.3.

4.2.2 Overlapped Block-Based Scanning Method

The stripe-based overlapped scanning method is employed by the state-of-the-art architecture proposed and presented in Section 3.3.1. With the observation that the external memory and the on-chip memory dominate the energy consumption of DWT architecture, a new overlapped block-based scanning method is proposed here, which enables the trade-off between external memory access and on-chip memory size and is more efficient in memory reads. Hence, more energy efficient than the stripe-based overlapped scanning method is. The overlapped block-based scanning method is illustrated in Fig. 4.2. The input image is partitioned into image blocks of width $2S$ and height R as represented by the rectangular boxes with thick border. There are overlapped rows (grey stripe in Fig. 4.2) between vertically adjacent blocks. Let F denotes the number of overlapped rows. $F = 3$ and 7 for the $5/3$ and $9/7$ DWT filter, respectively. The blocks are processed one-by-one in a line-based order from left to right. After a line (row) of the blocks is processed, the process continues from the next line of blocks. Within each block, the pixels are scanned in a top-down order. Every clock cycle, $2S$ pixels are concurrently output to the DWT processor. Therefore, the width of the image block (S) determines the parallelism (P) of the architecture. For the $5/3$ and $9/7$ filters, $P = 4S$ and $9S$, respectively, due to the number of cells in their DPPs (data processing pipes). For the $5/3$ filter, there are two cells in each DPP and there is at least one DPP in the row processing unit and one in the column processing unit. For the $9/7$ filters, there are four cells in each DPP and one in the scaling unit (Fig. 3.4). In order to process an image of size $M \times N$, $MN + MF \frac{N-R}{R-F}$ image memory reads are required.

Consequently, for each input frame, $2(MN + MF \frac{N-R}{R-F})$ and $4.5(MN + MF \frac{N-R}{R-F})$ computations have to be performed for the $5/3$ and $9/7$ filters, respectively. For the best existing DWT architecture proposed and presented in Section 3.3, $MN + N \frac{M}{2S}$ image memory reads are required for each frame. For the proposed data scanning method, an energy-optimized trade-off can be achieved by tuning the height of the image block (R) and the parallelism (P) so that the size of the on-chip memory is significantly reduced with only a little increase in external

bandwidth. The number of image memory reads of the proposed scanning method and the stripe-based scanning method (Section 3.3) are compared in Fig. 4.3 with various parallelisms for an image size of 2160×3840 and block height of 543. The number of image memory accesses is reduced by 33% ~ 14% for $P = 4 \sim 16$, resulting in higher energy efficiency and lower EAT. In Section 4.3.2, the energy efficiency of various design points is compared by varying the R and P . Note that in practical applications, a design with small S can offer sufficiently high throughput ($S \leq 12$).

Fig. 4.2 Overlapped block-based image scanning method

Fig. 4.3 Number of image memory accesses (5/3 filter, input size of 2160×3840 , $R=543$)

4.2.3 Parameterized 2-D DWT Processor

Based on the overlapped block-based scanning method, the proposed parameterized DWT architecture for the 9/7 DWT filter is shown in Fig. 4.4. The architecture is composed of an image memory and a DWT processor. The image memory is a DRAM bank composed of two DRAM chips to offer enough external bandwidth for high throughput. Each DRAM has a 16-bit output. The DWT processor is composed of a data buffer, a row processing unit (Row-PU), a column processing unit (Col-PU), a transposition memory, a temporal memory and a scaling unit. The data buffer is used to balance the throughput of the image memory and the DWT processor. The Row-PU and Col-PU are in charge of performing rDWT and cDWT, respectively. The temporal memory stores the partial results generated by the Row-PU. The transposition memory transposes the intermediate results H and L generated by the Row-PU and outputs these results to the Col-PU. The H and L are processed by the Col-PU in an interleaved order. The four subbands HH , HL , LH , LL generated by the Col-PU are then scaled by the scaling unit.

The detailed structure of the proposed design is shown in Fig. 4.5, Fig. 4.6 and Fig. 4.7. The lifting steps are performed by the Cell (Fig. 4.5), which is composed of one multiplier and two adders. Four Cells with different lifting coefficients are connected to constitute the data

Fig. 4.4 Proposed DWT processor

processing pipe (DPP) to perform the 1-D DWT as shown in Fig. 4.6 and Fig. 4.7. There are S DPPs in the Row-PU and the Col-PU. The Row-PU consumes $2S$ pixels from the image memory and 4 partial results from the temporal memory every clock cycle. Meanwhile, 4 partial results are generated by the $S-1^{\text{th}}$ DPP of the Row-PU and shifted into the temporal memory. The temporal buffer is composed of four shift registers each of length R . While the DPPs in the Row-PU receive/send the partial results from/to their adjacent DPPs, the DPPs in the Col-PU output the four partial results to four shift registers and consume them 2 clock cycles later. The shift registers in the Col-PU have a length of 2. Compared with the existing designs, the proposed architecture achieves a constant on-chip memory size, resulting in small EAT as demonstrated in Section 4.3.4.

The DWT architecture for 5/3 filter has a similar structure as that of 9/7 filter, except that there are only two Cells in the DPP and no scaling unit. Consequently, there are only 2 shift registers of size R in the temporal memory. For the 5/3 and 9/7 DWT filter, the parallelism is $P = 4S$ and $9S$, respectively.

In order to obtain the algorithm-mapping parameters that achieve the most energy-efficient design, the energy efficiency of designs with various R and P are compared in two steps in the following way.

- For each image size, the energy efficiency of designs with various block heights (R) and fixed parallelism (P) are compared. For the 5/3 and 9/7 filter, $P = 4$ and 9 are the smallest (minimum architecture). The value of R is determined by choosing the design that achieves the highest energy efficiency.
- The optimized value of R in the previous step is employed in this step to determine the value of P . For each image size, the energy efficiency of designs with various parallelism (P) and fixed block height (R) are compared. The value of P is determined by choosing the design that achieves the highest energy efficiency.

The experimental results are presented in Section 4.3.2.

Fig. 4.5 Structure of Cell

Fig. 4.6 Row-PU and transposition memory for 9/7 filter

Fig. 4.7 Col-PU and scaling unit for 9/7 filter

4.2.4 DRAM Power Modeling and Activation Schedule

- DRAM power modeling

The DRAM power dissipation is composed of three parts, namely the active power, the read/write power and the background power [94]. While the active and read/write power is determined by the percentage of the clock cycles that DRAM outputs data, the background power is determined by the power mode in which the DRAM is operated. There are two status of DRAM, namely pre-charge and active. In order to perform read and write operations, the DRAM has to be switched from the pre-charge status to the active status. For each status, there are two power modes, namely standby and power-down. The DRAM is turned into power-down mode when the clock signal is disabled. The active_standby mode and the pre-charge_power-down mode have the highest and lowest background power dissipation, respectively. On the other hand, there is energy and latency penalty of powering up/down DRAM [95].

- DRAM activation schedule

As the input image is stored in the external DRAM, at least MN pixels have to be read in order to process an image. As shown in Section 4.3.4, the DRAM and the on-chip memory dominates the total energy consumption. Because the number of read operations is fixed and the read/write power cannot be reduced, the aim is to reduce the background power of the DRAM. It is observed that, for the proposed DWT architecture, the DRAM only outputs data for a small percentage of the clock cycles (less than 40%). Therefore, instead of keeping the DRAM in active_standby mode all the time, the DRAM of all the designs is scheduled to the pre-charge_power-down mode when not being read. The proposed DRAM activation schedule to reduce the DRAM energy consumption is shown in Fig. 4.8. Without the activation schedule, the DRAM is read whenever new pixels are required by the DWT processor and stays in active_standby mode all the time. The DRAM cannot be powered down frequently because of the energy and latency penalty. By introducing the activation schedule, multiple pixels are pre-fetched and stored in the data buffer (Fig. 4.4) in one time, and then the DRAM is powered down to the pre-charge_power-down mode. As there is only a small number of powering up and down cycles, the energy penalty is negligible. As a result, the energy consumption of DRAM is significantly reduced (see Section 4.3.4).

Fig. 4.8 Timing diagram of DRAM activation schedule

4.3 Experiments and Evaluation

4.3.1 Experimental Setup

In this section, the proposed architecture and the baseline architecture (i.e. the state-of-the-art architecture in Section 3.3) for the 5/3 and 9/7 DWT filter with various block and input image sizes are implemented and verified. The energy efficiency and EAT are obtained and compared. In the experiments, three image sizes are selected as the implementation specification for the comparison. The small size image (640×480) is used in regular definition video [96] while the medium (1920×1080) and large (3840×2160) size image are employed in recently proposed high efficiency video coding standard (HEVC) [97]. The input images and data paths all have 32-bit depth.

The designs were implemented and verified in Verilog and synthesized in the STM 90-nm CMOS process with Synopsis Design Compiler. The Micron DDR3 chip with 16-bit output and 1GB density was used to estimate the power of the image memory. The Micron DDR3 power calculator [98] is used to compute the power dissipation. The operating frequencies of the image memory and the DWT processor were set at 800MHz and 50MHz, respectively.

4.3.2 Algorithm-Mapping Parameter Optimization

In the experiments, the energy efficiency is first evaluated with various heights of image block R and the simplest parallelism $P = 4$ and 9, respectively for the 5/3 and 9/7 filter. The block height R is set to the image height initially and then halved in each subsequent evaluation each

TABLE 4.1 OPTIMIZED VALUE OF R FOR VARIOUS IMAGE SIZES

	Small	Medium	Large
9/7 filter	67	75	75
5/3 filter	67	75	75

with an extra F overlapped rows. The energy efficiency of 9/7 filter for the small, medium and large size input with the selected R is shown in Fig. 4.9, Fig. 4.10 and Fig. 4.11, respectively. For 5/3 filter, the energy efficiency of the small, medium and large size image with varying R is shown in Fig. 4.12, Fig. 4.13 and Fig. 4.14. TABLE 4.1 summarizes the optimized values of R for various image sizes. From the results in the table, for both 5/3 and 9/7 filters, the energy optimized designs for the small, medium and large size input are $R = 67, 75$ and 75 , respectively. As R decreases, the size of the on-chip memory, which dominates the energy consumption of the DWT processor, is significantly reduced. Therefore, the DWT processor energy reduces significantly as demonstrated by the experimental results as shown in the figures. On the other hand, for $R > 67$, the DRAM energy only increases slightly as the height of image block R decreases. As a result, the energy efficiency is higher for smaller R . As R continues to decrease ($R < 67$), the DRAM energy begin to increase rapidly, caused by the increased number of DRAM accesses. Consequently, the energy efficiency becomes lower for very small value of R . In conclusion, by employing an optimized value of height of block R , the overlapped block-based image scanning method can significantly reduce the size of on-chip memory with an overhead of very small increase of DRAM accesses and therefore improve the energy efficiency of the DWT architecture.

With the optimized values of R , the energy efficiency of designs with various parallelism P is evaluated. The energy efficiency of 9/7 filter for the small, medium and large size input are shown in Fig. 4.15, Fig. 4.16 and Fig. 4.17, respectively. For 5/3 filter, the energy efficiency of the small, medium and large size input are shown in Fig. 4.18, Fig. 4.19 and Fig. 4.20, respectively. For all the image sizes, the energy efficiency of both the 9/7 and 5/3 filters increases with P . As P increases, the DRAM bank outputs data at a higher rate. As a result, the energy wasted in idle state is reduced. The larger the P , the smaller the energy consumed on DRAM access per image. Besides, the increased data parallelism facilitates the reuse of the partial results, reducing the number of on-chip memory accesses per image. Therefore, the on-chip memory energy decreases as the data parallelism increases. In conclusion, the energy

efficiency of the DWT architecture can be increased by employing larger parallelism P . However, the rate of increment becomes lower with increasing P as can be observed from the Fig. 4.15 ~ Fig. 4.20.

Fig. 4.9 Energy efficiency for 9/7 filter with small size input and various R

Fig. 4.10 Energy efficiency for 9/7 filter with medium size input and various R

Fig. 4.11 Energy efficiency for 9/7 filter with large size input and various R

Fig. 4.12 Energy efficiency for 5/3 filter with small size input and various R

Fig. 4.13 Energy efficiency for 5/3 filter with medium size input and various R

Fig. 4.14 Energy efficiency for 5/3 filter with large size input and various R

Fig. 4.15 Energy efficiency for 9/7 filter with small size input and various P ($R = 67$)

Fig. 4.16 Energy efficiency for 9/7 filter with medium size input and various P ($R = 75$)

Fig. 4.17 Energy efficiency for 9/7 filter with large size input and various P ($R = 75$)

Fig. 4.18 Energy efficiency for 5/3 filter with small size input and various P ($R = 67$)

Fig. 4.19 Energy efficiency for 5/3 filter with medium size input and various P ($R = 75$)

Fig. 4.20 Energy efficiency for 5/3 filter with large size input and various P ($R = 75$)

4.3.3 Peak Energy Efficiency

In this subsection, the peak energy efficiency of the STM 90-nm CMOS process library is evaluated. The power dissipation of the 32-bit multiplier at 50 MHz is 6mW, as obtained by Synopsys Design Compiler. As the DRAM dissipates a significant amount of power even in the power-down mode, it is assumed that the image memory is accessed at its peak bandwidth (1600M 32-bit pixels/sec.) when computing the peak energy efficiency. In order to consume all the input pixels every second, $\frac{1600 \times 2}{50} = 64$ and $\frac{1600 \times 4.5}{50} = 144$ multipliers are required for the 5/3 and 9/7 DWT filters, respectively. Consequently, according to (4.2), the peak energy efficiency of 9/7 DWT filter in the STM 90-nm CMOS process can be obtained by:

$$\eta_{peak} = \frac{4.5MN}{(144 \times 0.446 + 2 \times 367.9) \frac{4.5MN}{4.5 \times 1600}} = 9.00 \text{ GOPS/W}$$

With the same method, the peak energy efficiency of 5/3 DWT filter in the STM 90-nm CMOS process is obtained by:

$$\eta_{peak} = \frac{2MN}{(64 \times 0.446 + 2 \times 367.9) \frac{2MN}{2 \times 1600}} = 4.19 \text{ GOPS/W}$$

4.3.4 Performance Comparison

The energy efficiency and the energy consumption of the DRAM and the DWT processor among the baseline architecture (without any optimization) and the optimized architectures for the 9/7 filter and large size input are compared as shown in Fig. 4.21. The baseline architecture is defined as the design point with $R = 2160$ and $P = 9$. The parameter optimized architecture is defined as the design point with $R = 67$ and $P = 45$. Compared with the baseline architecture, the architecture with activation schedule achieves 34% energy efficiency improvement. The architecture with optimized parameters and the architecture with both optimized parameter and activation schedule achieve 4.55 times and 6.28 times better energy efficiency respectively when compared with the baseline architecture. The comparison of energy efficiency among the baseline and optimized architectures for all image sizes is listed in TABLE 4.2, where the normalized values are shown in the brackets for the convenient comparisons with the respective baseline architectures.

The energy efficiency and EAT of the proposed architecture is also compared with that of the proposed DWT architecture (Section 3.3). The experimental results of the 9/7 filter for the small, medium and large size input are shown in Fig. 4.22, Fig. 4.23 and Fig. 4.24, respectively. For various parallelism, the proposed architecture achieves better energy efficiency and EAT compared to the state-of-the-art architecture. The energy efficiency is improved by up to 269% of the state-of-the-art architecture. The energy efficiency of 5/3 filter for the small, medium and large size input is shown in Fig. 4.25, Fig. 4.26 and Fig. 4.27. The proposed architecture achieves better energy efficiency for all parallelism and improves the energy efficiency by up to 181% when compared with the state-of-the-art architecture. The proposed architecture has higher energy efficiency due to the lower DRAM and on-chip memory energy consumption. For both the proposed and the state-of-the-art architecture, as parallelism increases, more partial results are used immediately after they are generated, reducing the number of on-chip memory access per frame and the on-chip memory energy consumption. As the image size increases, the on-chip memory size of the state-of-the-art architecture increases with image height whereas the on-chip memory size of the proposed design is decided by the block height. Consequently, the area of the proposed design is smaller than the state-of-the-art architecture and the EAT ratio decreases with the input size increasing. Compared with the upper bound on energy efficiency, the proposed architecture achieves up to 82.8% and 86.6% of the peak energy efficiency for the 5/3 and 9/7 DWT filter, respectively.

Fig. 4.21 Energy efficiency and energy consumption per image for large size input (9/7 filter)

TABLE 4.2 ENERGY EFFICIENCY (GOPS/W) COMPARISON BETWEEN BASELINE AND OPTIMIZED ARCHITECTURES

Architecture	5/3 filter			9/7 filter		
	small	medium	large	small	medium	large
Baseline	0.89 (1.00)	0.77 (1.00)	0.62 (1.00)	1.77 (1.00)	1.41 (1.00)	1.02 (1.00)
Activation schedule	1.78 (2.00)	1.38 (1.77)	0.97 (1.54)	3.17 (1.79)	2.17 (1.54)	1.36 (1.34)
Parameter optimization	2.73 (3.07)	2.74 (3.52)	2.73 (4.35)	5.66 (3.19)	5.68 (4.03)	5.66 (5.55)
Activation schedule & parameter optimization	3.62 (4.07)	3.62 (4.66)	3.62 (5.76)	7.42 (4.19)	7.45 (5.28)	7.43 (7.28)

Fig. 4.22 Energy efficiency and EAT comparison (9/7 filter, small size)

Fig. 4.23 Energy efficiency and EAT comparison (9/7 filter, medium size)

Fig. 4.24 Energy efficiency and EAT comparison (9/7 filter, large size)

Fig. 4.25 Energy efficiency and EAT comparison (5/3 filter, small size)

Fig. 4.26 Energy efficiency and EAT comparison (5/3 filter, medium size)

Fig. 4.27 Energy efficiency and EAT comparison (5/3 filter, large size)

4.4 Discussions on Energy Performance

In Section 4.3, it is demonstrated that with the parameter optimization, the energy efficiency can be significantly improved. The optimized height of image block (R) is obtained based on the minimum parallelism ($P = 4$ and 9 for the $5/3$ and $9/7$ filter). The optimized R is then used for design points with various P . The results obtained from the experiments show that, under the constraint of practical external bandwidth, the larger the value of P , the more energy-efficient is the design. On the other hand, it is worth to note that for the P values investigated in the experiments, the best R value may not necessarily be the same as the optimized R value obtained for the design with the minimum P . Therefore, the parameter optimization leads to a good design point instead of the optimal design point. In order to obtain the best design point, an exhaustive search needs to be performed. However, in many practical situations, the design space is very large, making it impossible to synthesize every design point and evaluate their energy efficiency. An alternative to obtain the optimal design is to explore the design space by using energy performance model, as part of the future works elaborated in Section 6.2.4. While building an energy performance model might be challenging, the design space exploration can be significantly accelerated without having to synthesize all the design points.

In order to better evaluate the results of the proposed parameter optimization, the height of image block (R) with the maximum parallelism ($P = 20$ and 45 for the $5/3$ and $9/7$ filter) is optimized. As shown in Fig. 4.28 ~ Fig. 4.33, the best energy efficiency is achieved with $R = 127$, 142 and 142 respectively for the small, medium and large size image for both the $5/3$ and $9/7$ filter, instead of $R = 67$, 75 and 75 (TABLE 4.1). Compared with the design points obtained with the minimum parallelism ($P = 4$ and 9) in Section 4.3, the best R values for the maximum parallelism ($P = 20$ and 45) are only one step size different from those for the minimum parallelism and the energy efficiency is almost the same. Therefore, the designs obtained in Section 4.3 are suitable if the external bandwidth is limited. For other values of P , the designs can be optimized in the same way.

Fig. 4.28 Energy efficiency for 5/3 filter with small size input and various R ($P = 20$)

Fig. 4.29 Energy efficiency for 5/3 filter with medium size input and various R ($P = 20$)

Fig. 4.30 Energy efficiency for 5/3 filter with large size input and various R ($P = 20$)

Fig. 4.31 Energy efficiency for 9/7 filter with small size input and various R ($P = 45$)

Fig. 4.32 Energy efficiency for 9/7 filter with medium size input and various R ($P = 45$)

Fig. 4.33 Energy efficiency for 9/7 filter with large size input and various R ($P = 45$)

4.5 Summary

In this chapter, a lifting-based parameterized DWT architecture for the 5/3 and 9/7 DWT filter is proposed. Compared with prior designs, the energy efficiency is used as the main performance metric. The proposed architecture achieves high energy and area efficiency by adopting an overlapped block-based image scanning method, which optimizes the number of external memory reads and the on-chip memory size. By reducing the external memory and on-chip memory energy, high energy efficiency and low EAT for various input sizes can be achieved with the proposed architecture. The experimental results show that the proposed design achieves up to 86.6% of the peak energy efficiency. Compared with the state-of-the-art design, the proposed architecture achieves up to 269% energy efficiency improvement.

Chapter 5. Memory-Efficient Multi-Level 2-D DWT Architecture

5.1 Overview

As discussed in Chapter 1, the memory efficiency is one of the most important performance metrics for 2-D DWT architecture. The existing DWT architectures and their memory efficiency are studied in this work. The following observations are resulted and a design strategy is developed based on the observations to improve the performance of 2-D DWT architecture.

- In both the convolution-based and the lifting-based architectures, the area is dominated by the memory.
- Data scanning method has a significant impact on memory size as it decides how the data flows and how the computations are scheduled.
- To achieve high memory efficiency, the temporal memory of the lifting-based architecture can be further reduced with reasonable trade-off between the external memory bandwidth and the internal arithmetic resources.
- The lifting-based architecture demands less arithmetic resources than the convolution-based architecture does.
- For multi-level DWT, the pipelined architecture leads to higher memory efficiency than the folded architecture does.

Based on these observations, a design strategy is formulated to further improve the performance of multi-level 2-D DWT architecture for memory efficiency and high throughput. First, as the convolution-based architecture could not facilitate further reduction of arithmetic resource, the lifting-based architecture is chosen in this work. The inherent parallelism of the lifting-based architecture makes it more suitable for large parallel design and the flipping method [30] can be applied for the lifting scheme to keep the critical path length short. The pipelined architecture is employed to eliminate the frame memory. The overlapped stripe-based scanning method proposed by Huang *et al.* [31] is adopted and modified to suit the parallel lifting-based architecture. Based on the above strategy, a pipelined lifting-based multi-level 2-D DWT

architecture is proposed, which is memory-efficient and has high throughput. The features of the proposed design and its major differences from the existing designs are summarized below.

- The overlapped stripe-based scanning method employed in the proposed design has scalable stripe width with 7 columns overlapped between two adjacent stripes. The data in the overlapped columns are processed by a newly proposed partial processor in the rDWT of the first level (Level 1) DWT. With this approach, the temporal memory of Level 1 DWT, which occupies a dominating chip area in other lifting-based architectures, is eliminated at the expenses of reading seven more pixels per cycle and a few extra arithmetic resources. The elimination of the temporal memory results in significant memory saving.
- A new stripe-based data sequencing scheme is proposed for feeding the output data of Level 1 DWT to the input of Level 2. The method is also extended and applied to the inputs of the higher level DWTs. With this data sequencing scheme, 100% hardware utilization is achieved.
- Compared with the most efficient existing stripe-based multi-level DWT architecture reported in [24], the proposed design consumes only half of the arithmetic resources and 60% of the memory due to its lifting-based scheme instead of the convolution-based scheme used in [24]. Moreover, the proposed design has a scalable throughput whereas the design in [24] has a fixed throughput of 16 pixels per cycle.
- The proposed design is based on the pipelined architecture, which does not require frame memories, unlike the existing folded multi-level DWT architectures [35], [32], [37], where a frame memory is needed between every two levels of DWT.
- The flipping method is applied in the proposed design for shortening the critical path delay.

With the above features, the proposed architecture is more memory-efficient and has higher throughput than the existing parallel multi-level DWT architectures. The synthesis results (Section 5.5) in 90 nm CMOS process show that the proposed design is 60.2% smaller in area-delay product (ADP) and 97.4% higher in throughput for an image size of 512×512 pixels when compared with the best existing design [24] for the CDF 9/7 2-D DWT. The proposed data

sequencing scheme, the multi-level DWT architecture, its performance and the comparisons with the existing designs are detailed in the following sections.

5.2 Review of Lifting Scheme and Flipping Method

The lifting scheme, proposed by Daubechies and Sweldens [70], is an alternative way of constructing the wavelet filters by lifting steps, namely, split, predict, update and scaling. The polyphase matrix of the low-pass and high-pass FIR filter bank can be factorized into the lifting steps as below:

$$P(z) = \prod_{i=1}^m \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \begin{bmatrix} 1/K & 0 \\ 0 & K \end{bmatrix} \quad (5.1)$$

where $P(z)$ is the polyphase matrix of the FIR filter banks, K is the scaling coefficient, $s_i(z)$ and $t_i(z)$ are the Laurent polynomial of update and prediction steps [70], respectively.

The 2-D DWT can be decomposed into two steps, namely rDWT and cDWT. The rDWT generates the high-pass (H) and low-pass (L) intermediate results from the input samples and sends them to the cDWT. The cDWT then decomposes the high-pass and low-pass intermediate results into four subbands, namely the high-low (HL), high-high (HH), low-low (LL) and low-high (LH) subbands. The rDWT for the 9/7 filter can be represented by (5.2) – (5.5) [39].

$$p(m, n) = x(m, 2n) + \alpha(x(m, 2n-1) + x(m, 2n+1)) \quad (5.2)$$

$$q(m, n) = x(m, 2n-1) + \beta(p(m, n-1) + p(m, n)) \quad (5.3)$$

$$H(m, n) = p(m, n-1) + \gamma(q(m, n-1) + q(m, n)) \quad (5.4)$$

$$L(m, n) = q(m, n-1) + \delta(H(m, n-1) + H(m, n)) \quad (5.5)$$

where $x(m, 2n)$, $x(m, 2n-1)$, and $x(m, 2n+1)$, for $0 \leq m \leq N-1$ and $0 \leq n \leq N/2-1$, are the pixels of the input image of size $N \times N$. α , β , γ and δ are the lifting coefficients. $p(m, n)$, $q(m, n)$ and $H(m, n)$ are the partial results generated by the lifting steps of the rDWT. Note that $H(m, n)$ in (5.4) is both the high-pass intermediate result and the partial result for (5.5) of the rDWT.

The outputs H and L of the rDWT are fed into the cDWT alternately. The formulation of the cDWT can be obtained by substituting either H or L into $x(m,n)$ of (5.2) and (5.3). When H is the input to the cDWT, its outputs are $HH(m,n)$ and $HL(m,n)$ as given in (5.8) and (5.9). When L is the input, the outputs $LH(m,n)$ and $LL(m,n)$ can be similarly obtained.

$$ph(m,n) = H(2m,n) + \alpha(H(2m-1,n) + H(2m+1,n)) \quad (5.6)$$

$$qh(m,n) = H(2m-1,n) + \beta(ph(m-1,n) + ph(m,n)) \quad (5.7)$$

$$HH(m,n) = ph(m-1,n) + \gamma(qh(m-1,n) + qh(m,n)) \quad (5.8)$$

$$HL(m,n) = qh(m-1,n) + \delta(HH(m-1,n) + HH(m,n)) \quad (5.9)$$

where $ph(m,n)$, $qh(m,n)$ and $HH(m,n)$, for $0 \leq m \leq N/2-1$ and $0 \leq n \leq N/2-1$, are the partial results generated by the cDWT for the high-pass band H . Note that $HH(m,n)$ is both the partial result of cDWT and the high-high subband coefficient. The cDWT for the low-pass subband L shares the same equations as those for H (5.6) – (5.9), except that $H(m,n)$, $ph(m,n)$, $qh(m,n)$, $HH(m,n)$ and $HL(m,n)$ are replaced by $L(m,n)$, $pl(m,n)$, $ql(m,n)$, $LH(m,n)$ and $LL(m,n)$, respectively.

For the higher level DWTs, the equations are the same as (5.2) – (5.9) but with the low-low subband generated by the preceding level as the input. As the low-low subband is down-sampled in both the column and row direction, its size is only one quarter of the input of its preceding level and consequently, the ranges of m and n are halved after each level. Here, the superscript j denotes the signals in Level j ($j > 1$) DWT. For instance, LL^j , p^j and L^j represent the low-low subband output, partial result and low-pass intermediate result of Level j , respectively. Note that the superscript j is omitted for Level 1 DWT.

Despite having several favorable characteristics, the lifting scheme suffers from a long critical path. The flipping method was proposed by Huang *et al.* [30] to shorten the critical path length by flipping the computation nodes with the reciprocals of the lifting coefficients. The 1-D flipping method can be seen from the following mathematical formulation that the multiplications with the coefficients no longer succeed the additions on the critical paths.

$$p(m,n) = \alpha^{-1}x(m,2n) + 2^{-k}(x(m,2n-1) + x(m,2n+1)) \quad (5.10)$$

$$q(m, n) = \beta^{-1} x(m, 2n-1) + 2^{-k} (p(m, n-1) + p(m, n)) \quad (5.11)$$

$$H(m, n) = \gamma^{-1} p(m, n-1) + 2^{-k} (q(m, n-1) + q(m, n)) \quad (5.12)$$

$$L(m, n) = \delta^{-1} q(m, n-1) + 2^{-k} (H(m, n-1) + H(m, n)) \quad (5.13)$$

where α^{-1} , β^{-1} , γ^{-1} and δ^{-1} are the flipped lifting coefficients, 2^{-k} is a constant factor for scaling the large coefficient reciprocals to circumvent large wordlength or overflow in practical designs and can be implemented with a k -bit right shift operation without incurring hardware or delay overhead.

5.3 Proposed Input Data Scanning Method

5.3.1 Overlapped Stripe-Based Scanning Method

As discussed in Section 5.1, the temporal memory needed for storing the partial results constitutes a large portion of the memory in the existing lifting-based 2-D DWT architectures. The temporal memory can be eliminated if the partial results are not stored but produced as and when they are needed. A new overlapped stripe-based scanning method is proposed here to eliminate the temporal memory at the expense of additional arithmetic resources by regenerating the partial results when they are needed. As a result, high memory efficiency is achieved. The overlapped stripe-based scanning method is first presented, followed by the descriptions of the data input sequencing for Level 2 and Level j DWT.

In the proposed overlapped stripe-based scanning method, an image of size $N \times N$ is divided into $R = N/2S$ stripes, each of width $2S$ columns and height N rows, with S being the number of parallel processing units in the rDWT that processes $2S$ pixels concurrently. Fig. 5.1 shows three stripes surrounded by thick borders, $r-1$, r , and $r+1$, ($0 \leq r \leq R-1$), of a partial image, where the grey and white squares representing respectively the overlapped and non-overlapped pixels. The image is scanned into the rDWT stripe by stripe, and row by row within each stripe in a top-down direction as indicated by the arrows. In each cycle, $2S$ pixels from the current stripe r with 7 pixels from the preceding stripe $r-1$ are scanned into the rDWT. For the first stripe, seven padding columns of zeros are used as the overlapped pixels.

This method is developed based on the analysis of the dependency graph of the lifting scheme

Fig. 5.1 Overlapped stripe-based scanning method for Level 1

derived from (5.2) – (5.9) and shown in Fig. 5.2, where the two rectangular boxes at the top contain the input pixels of the current stripe ($r = 0$) and 7 columns of input pixels of the preceding stripe ($r = -1$). The circle nodes marked with α , β , γ , and δ are the computation nodes, whereas the dotted oblique boxes contain the parallel processing units, each composing of four computation nodes. Each processing unit has three pixel inputs and three partial result inputs. Considering the computation of the first element of the low-pass subband $L_{0,0}$, it can be seen from the dependence graph of the rDWT that the shaded processing unit requires $x_{0,-1}$, $x_{0,0}$ and $x_{0,1}$ as well as three partial results, $p_{0,-1}$, $q_{0,-1}$ and $H_{0,-1}$, which are produced from 7 pixels of the preceding stripe. Hence, to avoid storing the partial results from the proceeding stripe, 7 pixels from the preceding stripe have to be available at the same time when the pixels in the current stripe are input for processing. In other words, the current stripe has to overlap with the preceding one with 7 columns of pixels. The shaded triangular box encloses the computation nodes that are needed for generating the three partial results. For clarity, the current stripe shown in Fig. 5.2 is the stripe $r = 0$ whereas $r = -1$ is its proceeding stripe. The negative stripe number and subscripts denote the paddings of zeros. In general, if the current stripe is r , the column indices (m) of the input pixels will be offset by $+2rS$ and the column indices of the intermediate results and the partial results will be offset by $+rS$. For instance, $x_{1,3}$ and $L_{2,1}$ would be $x_{1,3+2rS}$ and $L_{1,3+2rS}$ should the current stripe be r . Also note that the indices m and n of (5.2) – (5.9) are shown as subscripts. e.g., $H_{0,-1}$ denotes $H(0,-1)$.

Fig. 5.2 Dependency graph of 2-D lifting DWT

With $2S$ columns in each stripe, S parallel processing units are required in the rDWT. One pair of H and L are produced by each processing unit s ($0 \leq s \leq S-1$) in each cycle as shown in the dotted rectangular boxes immediately below the rDWT in Fig. 5.2, where the H and L outputs are drawn offset by one cycle to illustrate that the H outputs are generated by the computation nodes that precede those for producing the L outputs. The offset is also reflected in the actual implementation (Section 5.4). On the other hand, each processing unit of the cDWT needs to consume two H or two L results in every alternate cycle. Therefore, the intermediate results H and L produced by the rDWT have to be buffered, transposed and interleaved as shown in Fig. 5.2 between the rDWT and cDWT. Consequently, each processing unit of the cDWT will produce in two cycles one set of four subband coefficients, HH , HL , LH and LL , in an interleaved sequence with $[HH, HL]$ in one cycle and $[LH, LL]$ in another. Two sets of such interleaved outputs are shown in the boxes at the bottom of Fig. 5.2, where the outputs in the dotted boxes correspond to the inputs in the dotted boxes at the top. The proposed hardware structures for the processing unit, the rDWT and cDWT as well as the transposition memory are detailed in Section 5.4.

Unlike the 3 existing stripe-based scanning methods [86], [31] and [24], the proposed scanning method takes as input 7 additional (overlapped) pixels per row for the computation of the partial results so that no temporal memory is needed to store them in Level 1 DWT. The original stripe-based method proposed by Chiu *et al.* [86] does not have overlapped pixels but needs a large memory. Also, no hardware architecture for processing the input pixels is reported in [86]. The modified stripe-based method by Huang *et al.* [31] has 8 overlapped columns per stripe for the 9/7 filter, resulting in longer computation time. Their method can be used for both single-level convolution- or lifting-based DWT but not for high throughput architecture. Coincidentally, the scanning method proposed by Mohanty and Meher [24] also has 7 overlapped columns per stripe but the stripe width is fixed at 16 pixels. The corresponding multi-level convolution-based architecture reported in [24] is the most efficient existing design in terms of ADP. However, the scanning method is proposed for the convolution-based design and the resulting architecture consumes more arithmetic and memory resources compared to the proposed design. The proposed method is extended for multi-level decomposition and applied to a newly designed lifting-based multi-level 2-D DWT architecture as described below. The experimental results show the proposed architecture achieves a smaller ADP and higher throughput than the best existing design [24] (TABLE 5.4 in Section 5.5.3).

5.3.2 Data Sequencing for Higher Level Decomposition

The DWT produces one set of $[HL, HH]$ on one cycle and one set of $[LL, LH]$ in another as shown in Fig. 5.2. In the pipelined multi-level DWT architectures, only the LL coefficients are input to the higher level DWT for further decomposition. Due to the down-sampling in both the row and column direction, the size of LL^j is only one quarter of LL^{j-1} . In order to achieve 100% hardware utilization, the throughput of each level should also be one quarter of its preceding level. For example, the level-2 DWT should only consume $S/2$ LL samples, given that the level-1 structure consumes $2S$ pixel samples every cycle. For the level- j DWT, $S / 2^{2j-3}$ samples are to be consumed. As such, the LL^{j-1} output sequence has to be considered when deriving the input data sequencing for the level- j ($j > 1$) decomposition. The following data sequencing scheme is proposed for the higher level decomposition.

- a) Each row of the LL outputs of one level are split into two segments and fed into the next level in two cycles to balance the throughput and achieve 100% hardware utilization.
- b) The split segments are input into the succeeding level in an interleaved order. Consequently, the corresponding outputs of the succeeding level are also interleaved.

The sequencing scheme is first illustrated with the level-2 decomposition as an example and then generalized for the level- j ($j > 1$) decomposition. The outputs of the level-1 decomposition, i.e. LL and HL at the bottom of Fig. 5.2, corresponding to the input stripe $r = 0$ are shown in Fig. 5.3(a). During every even cycle, S HL coefficients are output from the level-1 structure, shown in the dotted boxes that shaded with light grey stripes. During every odd cycle, one row of S LL coefficients is output from the level-1 DWT with the first half row of $S/2$ coefficients shown in the white boxes and the other half in dark grey boxes. As the level-2 DWT consumes only $S/2$ samples every cycle for balancing the throughput, each row of the LL coefficients, generated during every odd cycle, is split in two halves and fed into the level-2 DWT in two consecutive cycles, as shown in Fig. 5.3(b). For example, the samples $LL_{0,0} \sim LL_{0,\Delta-1}$ and $LL_{0,\Delta} \sim LL_{0,2\Delta-1}$ generated in the cycle 1 ($t = 1$) are fed into the level-2 DWT in the cycles 1 and 2, respectively. In this way, the output LL coefficients of the level-1 DWT that correspond to one stripe ($r = 0$ in this case) of the input image are now organized into two segments for feeding into the level-2 DWT in an interleaved manner. Note that the segment width Δ is also the number of samples consumed in each cycle. Also note that the two segments in Fig. 5.3(b) are fed into the same

inputs of the next level DWT in alternate cycles. They are shown in two separated columns solely for clarity.

Fig. 5.3 (c) and (d) show the output and input sequences of Level 2 and Level 3 (for $r = 0$), respectively. Because the LL coefficients of the same row within a stripe are split into two segments and fed into Level 2 in interleaved order, the outputs of Level 2 are also interleaved. For example, in the cycles 0 and 1, $HL_{0,0}^2 \sim HL_{0,2\Delta-1}^2$ and $LL_{0,0}^2 \sim LL_{0,2\Delta-1}^2$, correspond to the first input segment, are generated at the output of Level 2, respectively. In the cycles 2 and 3, $HL_{0,2\Delta}^2 \sim HL_{0,4\Delta-1}^2$ and $LL_{0,2\Delta}^2 \sim LL_{0,4\Delta-1}^2$, correspond to the second input segment, are generated, respectively. While $HL_{0,0}^2 \sim HL_{0,2\Delta-1}^2$ and $HL_{0,2\Delta}^2 \sim HL_{0,4\Delta-1}^2$ are directly output, $LL_{0,0}^2 \sim LL_{0,2\Delta-1}^2$ and $LL_{0,2\Delta}^2 \sim LL_{0,4\Delta-1}^2$ are split into four segments and fed into Level 3 during the cycles 1 to 4 as shown in Fig. 5.3 (d).

The sequencing scheme can be generalized for Level j as shown in Fig. 5.4. At the output of Level $j-1$, the LL^{j-1} coefficients of one row corresponding to one stripe ($r=0$ in this case) of the input image are output in 2^{j-2} interleaved segments as in Fig. 5.4 (a). Each of these 2^{j-2} segments is further split into 2 halves for the input to Level j as in Fig. 5.4 (b). Consequently, 2^{j-1} segments are fed into Level j in 2^{j-1} consecutive cycles. For clarity, the indices in Fig. 5.3 and Fig. 5.4 are shown for the stripe $r=0$. In general, the sequencing scheme for the stripe r is the same except that the column indices are offset by $+rS/2^{j-2}$ for the level- j inputs. E.g. $LL_{0,2\Delta-1}^{j-1}$ for the stripe $r=0$ would be $LL_{0,2\Delta-1+rS/2^{j-2}}^{j-1}$ in general, with $\Delta = S/2^{2j-3}$.

Fig. 5.4 Output sequence of Level $j-1$ and Input sequence of Level j ($j > 1, r = 0$)

With the above sequencing scheme, the throughput is maintained throughout all the processing units at all the levels. i.e. It takes the same amount of time to input one stripe of the original image, to produce one stripe of LL , and to input one stripe of LL to the succeeding level, to produce the LL^2 subband, and so on. As such, no frame buffer is required and the subband outputs are consumed as and when they are produced, resulting in 100% hardware utilization. It is worth noting that should the LL coefficients not be split but fed into the succeeding level directly when they are produced, more processing units would be required, resulting in larger hardware size and lower utilization despite the transposition memory would be smaller.

Similar to the transposition needed in the level-1 decomposition (Fig. 5.2), the intermediate results need to be stored and transposed after the rDWT of Level j . In the level-2 decomposition, two interleaved LL input segments (Fig. 5.3 (b)) are scanned into the rDWT in alternate cycles. At a processing unit s in the rDWT, only one intermediate result of the intermediate matrices L^2 and H^2 is generated every two cycles as shown at the upper part of Fig. 5.5(a). However,

Fig. 5.5 (a) Transposition order of Level 2 decomposition (b) Transposition order of Level j decomposition

each processing unit in the cDWT consumes two intermediate results of L^2 or H^2 . Therefore, both the interleaved L^2 and H^2 intermediate results have to be stored and transposed. The input sequence of the cDWT for Level 2 is shown in the lower dotted box of Fig. 5.5(a), where $\Delta = S / 2$ is the input segment width of the level-2 DWT (Fig. 5.3 (b)).

The transposition order is generalized for Level j as shown in Fig. 5.5(b). At Level j , 2^{j-1} interleaved segments of LL^j are fed into the rDWT and 2^{j-1} interleaved intermediate results of H^j and L^j are produced. Consequently, all these 2^{j-1} interleaved intermediate results have to be stored and transposed as shown in Fig. 5.5(b), where $\Delta = S / 2^{2^{j-3}}$ is the input segment width of the level- j DWT (Fig. 5.4(b)).

5.4 Proposed Architecture

This section presents the modules for realizing the proposed multi-level 2-D DWT.

5.4.1 Data Processing Pipe (DPP)

Based on (5.10) – (5.13), the flipped data flow graph (DFG) of lifting scheme can be derived as shown in Fig. 5.6, where the computation nodes are flipped to eliminate the multiplier on the critical path. Instead of multiplications, only the overflow prevention factors 2^{-k} (see Section 5.2) are on the critical path. They are implemented as k -bit right shifts with hard-wired interconnection and do not incur delay on the critical path. In this design, $k = 1$ is used.

The basic operation nodes of the flipped DFG are implemented as Cells as shown in the dotted box of Fig. 5.6. All the Cells are functionally and structurally identical and each consists of one multiplier, two adders and three k -bit right shifters. Its critical path length of $T_m + T_a$ consists of the delay of one multiplier (T_m) and one adder (T_a).

The data path that consists of 4 operation nodes in Fig. 5.6 is implemented as a Data Processing Pipe (DPP) that comprises four Cells, each with a constant coefficient α^{-1} , β^{-1} , γ^{-1} and δ^{-1} as shown in Fig. 5.7. The DPP is used to realize the parallel processing units in Fig. 5.2. i_0 , i_1 and i_2 are the three inputs. a , b and c are the three partial result inputs whereas a' , b' and c' are the three partial result outputs. o_0 and o_1 are the high-pass and low-pass outputs, respectively and

the dotted grey output of the last Cell is ignored. All the DPPs in both the rDWT and cDWT at all the levels are identical but with different inputs and outputs as listed in TABLE 5.1.

Fig. 5.6 Flipped DFG of lifting scheme

Fig. 5.7 Data processing pipe (DPP)

TABLE 5.1 INPUTS AND OUTPUTS OF DPPs AT DIFFERENT DWTs

DWT	Level 1 rDWT	Level 1 cDWT	Level j rDWT	Level j cDWT
$i_0 \sim i_2$	x	H or L	LL^{j-1}	H^j or L^j
a	p	ph or pl	p^j	ph^j or pl^j
b	q	qh or ql	q^j	qh^j or ql^j
c	H	HH or LH	H^j	HH^j or LH^j
a'	p	ph or pl	p^j	ph^j or pl^j
b'	q	qh or ql	q^j	qh^j or ql^j
c'	H	HH or LH	H^j	HH^j or LH^j
o_0	H	HH or LH	H^j	HH^j or LH^j
o_1	L	HL or LL	L^j	HL^j or LL^j

5.4.2 DWT Architecture for Level 1 Decomposition (Arch I)

With the proposed data scanning method for Level 1 decomposition described in Section 5.3 and based on the DPP structure proposed above, the architecture for Level 1 decomposition, Arch I, is presented below.

Without loss of generality, let the stripe width of the input image be $2S$. The 7 overlapped columns are generated by an external input buffer (see Fig. 3.2) similar to the one proposed by Mohanty and Meher as shown in Fig. 8 of their work [24]. Hence, $2S + 7$ pixels are input into the rDWT concurrently. The rDWT is realized by a Row-PU (Processing Unit) and an Auxiliary-PU as shown in Fig. 5.8 for processing the $2S$ and 7 pixels respectively. The Row-PU in Fig. 5.8(b) consists of S parallel DPPs with each DPP consumes 2 pixels every clock cycle. In each clock cycle, $2S$ pixels are concurrently processed by the Row-PU and S pairs of the intermediate results H and L (one pair from each DPP) are produced. The partial result outputs a' , b' and c' of the s th DPP ($0 \leq s \leq S - 2$) are directly fed to the partial result inputs a , b and c of the

Fig. 5.8 Structure of rDWT in Arch I: (a) Auxiliary-PU (b) Row-PU

$(s + 1)$ th DPP, respectively. The 3 partial results generated by the last DPP ($s = S - 1$) are ignored. The partial result inputs of the first DPP ($s = 0$) are generated from the 7 overlapped pixels. The Auxiliary-PU in Fig. 5.8(a), derived from the nodes in the triangle of the dependency graph in Fig. 5.2, is used to process these 7 pixels. It does not contain any DPP but consists of only 6 Cells. It consumes 7 pixels and produces 3 partial results every clock cycle. These 3 partial results are not needed in the subsequent cycles and not stored. As a result, a temporal memory of size $3N$ is saved. Replacing the temporal memory with the Auxiliary-PU leads to significant area reduction as observed from the performance analysis presented in Section 5.5.

A transposition memory is needed to store and interleave the H and L intermediate results according to the scanning method described in Section 5.3.1 and illustrated in Fig. 5.2. Each DPP of the rDWT of Arch I needs one transposition register at its output. The transposition register corresponding to one DPP (s) is depicted in Fig. 5.9, where four pairs of H and L before (in the dotted boxes) and after passing through the transposition registers (z^{-1}) are shown. In every clock cycle, one H and one L produced by the respective DPP are fed into the transposition register. The transposition register stores these intermediate results, transposes them and outputs a pair of H or L in alternate clock cycles to the cDWT.

The cDWT is realized with S independent DPPs as shown in Fig. 5.10. In every clock cycle, it alternately consumes $2S$ intermediate results of either H or L and produces a subband pair $[HH, HL]$ or $[LH, LL]$. As H and L are processed in an interleaved order, one intermediate result and three partial results generated by each DPP will be consumed by the four Cells only two cycles later, i.e. one extra cycle for interleaving the partial results in addition to the cycle

Fig. 5.9 Structure of transposition register

Fig. 5.10 Structure of cDWT

needed in the original lifting scheme (z^{-1} in Fig. 5.6). Consequently, three partial results as well as one intermediate result need to be stored in four shift registers (SR in Fig. 5.10) each of length 2.

The generated coefficients of the four subbands HH , HL , LH , and LL are scaled by the Scaling Units (SUs), of which the structure is shown in Fig. 5.11, where K_1 , K_2 and K_3 are the scaling coefficients. The structure of Arch I is composed of one rDWT, one cDWT, S transposition registers and S SUs as shown in Fig. 5.12.

Fig. 5.11 Scaling unit

Fig. 5.12 Structure of Arch I

5.4.3 DWT Architecture for Level j Decomposition (Arch j)

Based on the DPP structure and the stripe-based sequencing scheme for j ($j > 1$) level decomposition described in Section 5.3.2, the rDWT, cDWT and transposition register of Arch j for Level j decomposition are built with careful scheduling of the inputs, outputs, partial result inputs and partial results outputs as described below. Fig. 5.13 shows the structure of the rDWT for Arch j . With the segment width of $S / 2^{2j-3}$ at Level j (Fig. 5.4), $S / 2^{2(j-1)}$ DPPs are needed in the rDWT to process $S / 2^{2j-3}$ LL^{j-1} coefficients concurrently. As described in Section 5.3.2, Arch j receives the LL^{j-1} coefficients from its preceding level and the overlapped scanning method used for the pixel inputs at Arch I is not applied to the inputs at Arch j to avoid the temporal memory. Instead of an Auxiliary-PU, four shift registers (SRs) are introduced to store one LL^{j-1} coefficient and three partial results produced by the last DPP every clock cycle. All the partial results input at the s th DPP ($1 \leq s \leq S / 2^{2(j-1)} - 1$) come from the partial result outputs of the $(s-1)$ th DPP directly as shown in Fig. 5.13, whereas the partial result inputs of the first DPP ($s=0$) come from two sources. In the first situation when the first segment of LL^{j-1} (e.g. $LL_{0,0}^{j-1} \sim LL_{0,\Delta-1}^{j-1}$ in Fig. 5.4(b)) of the 2^{j-1} interleaved segments is input for processing, the rDWT requires the partial results generated from the preceding stripe N clock cycles earlier by the last DPP ($s = S / 2^{2(j-1)} - 1$). In the other situation, the partial result inputs are those generated one clock cycle earlier by the last DPP. To resolve these two situations, four shift registers each of length $N / 2^{j-1}$ and four multiplexers (MUXes) are adopted. The shifter registers are used to store one coefficient and three partial results from the last DPP when processing the last segment. Once every 2^{j-1} clock cycles, each shift register is enabled once and a new coefficient/partial result is stored. When a shift register is enabled, it is also selected by the corresponding MUX. In all the other $2^{j-1} - 1$ clock cycles, the coefficient and partial results of the last DPP pass through the MUXes and are consumed by the first DPP as shown in Fig. 5.13. Note that the enable and selection signals are staggered when activated due to the pipelined operations.

Fig. 5.14 shows the structure of the transposition register for Arch j , which is designed according to the transposition sequence described in Section 5.3.2 and shown in Fig. 5.5(b). Each transposition register is composed of two 1-to- 2^{j-1} de-multiplexers, two 2^{j-1} -to-1 multiplexers and $2^{j+1} - 2$ registers. There are $S / 2^{2(j-1)}$ transposition registers in Arch j and they constitute

the transposition memory. Each transposition register receives the intermediate results generated by one DPP of the level- j rDWT. Different from the transposition register of Arch I, which stores and transposes one pair of interleaved H and L (Fig. 5.9), each transposition register in Arch j needs to store and transpose 2^{j-1} pairs of L^j and H^j (Fig. 5.5(b)), because each DPP of the level- j rDWT needs to process 2^{j-1} columns from the 2^{j-1} interleaved segments (Fig. 5.4(b)). For the first pair of L^j or H^j , one register is needed. For the rest of the pairs, two registers are needed for each pair as shown in Fig. 5.14. Therefore, there are $2^{j+1} - 2$ registers in each transposition register for transposing the 2^{j-1} columns of L^j and H^j at the output of the level- j rDWT (Fig. 5.5(b)). It takes 2^j clock cycles to transpose all the 2^{j-1} interleaved intermediate results L^j and H^j .

The cDWT of Arch j processes the 2^{j-1} L^j and H^j intermediate results also in an interleaved way. The structure of the level- j cDWT is the same as the cDWT of Arch I (Fig. 5.10) except that it contains $S / 2^{2(j-1)}$ DPPs and the length of the four shift registers in each DPP is 2^j instead of 2 in Arch I as it needs to store the partial results from processing 2^{j-1} pairs of interleaved intermediate results L^j and H^j (Fig. 5.5(b)) instead of just one pair. The level- j cDWT alternately produces a pair of the subbands $[HH^j, HL^j]$ and $[LH^j, LL^j]$. As described in Section 5.3.2, the complete subband ($[HH^j, HL^j]$ or $[LH^j, LL^j]$) consisting of 2^{j-1} segments each of $N / 2^j$ rows and $S / 2^{2(j-1)}$ columns are produced in N cycles, i.e. one row is produced in every 2^j clock cycles.

The scaling unit used in Arch j is the same as the one in Arch I (Fig. 5.11). At Level j , there are $S / 2^{2(j-1)}$ such units. They alternately scale the subbands $[HH^j, HL^j]$ and $[LH^j, LL^j]$ produced by the cDWT.

Fig. 5.13 The structure of rDWT of Arch j

Fig. 5.14 Structure of Transposition Register for DPP s of Arch j

5.4.4 Proposed Pipelined Multi-Level DWT Architecture

The proposed pipelined multi-level DWT architecture is shown in Fig. 5.15, where the decomposition at Level j is performed by Arch j . The subband LL^j is fed as the input to the succeeding level, while the other subbands LH^j , LH^j and HH^j are output directly. Between every pair of processors, there is a Splitter j , which realizes the sequencing scheme by splitting the output rows into halves as described in Section 5.3 and illustrated in Fig. 5.3 and Fig. 5.4. The structure of Splitter j is shown in Fig. 5.16. Splitter j receives $S / 2^{2(j-1)}$ coefficients of LL^j every two clock cycles. Assuming the coefficients arriving at Splitter j in the first clock cycle be $LL_{0,0}^j$ to $LL_{2^{\Delta}-1}^j$. The first half of the inputs, $LL_{0,0}^j$ to $LL_{2^{\Delta}-1}^j$, is immediately input to Arch $j+1$ through the MUX, while the second half, LL_{Δ}^j to $LL_{2^{\Delta}-1}^j$, is stored in the SegmentReg register for one cycle. In the second cycle, they are output through the MUX. The process continues until the whole LL^j subband is generated. In this way, the input pixels are processed in Arch I and subsequently the LL^j subbands are fed directly into and processed in Arch $j+1$. Therefore, the complete input image is decomposed seamlessly in the pipelined multi-level 2-D DWT architecture, achieving 100% utilization in all the hardware modules. It is worth mentioning that should the LL^j coefficients not be split but fed to Arch $j+1$ in the same cycle, more DPPs would be needed and the hardware utilization would be halved at each level.

5.5 Hardware Estimation and Performance Analysis

5.5.1 Theoretical Estimation

Let L ($L > 1$) be the total decomposition levels and $2S$ the input stripe width for Arch I. Both the Row-PU and cDWT of Arch I need S DPPs to process the pixels in parallel and S SUs are needed to scale the coefficients. Consequently, Arch j requires $S / 2^{2(j-1)}$ DPPs (the coefficients are down sampled to one quarter after each level) in each of its rDWT and cDWT, as well as $S / 2^{2(j-1)}$ SUs. Hence, the total numbers of DPPs η_{DPP} and SUs η_{SU} in the L -level pipelined architecture are:

$$\eta_{DPP} = \sum_{j=1}^L 2S / 2^{2(j-1)} = 8S(1 - 4^{-L}) / 3, \text{ and}$$

Fig. 5.15 The pipelined multi-level DWT architecture

Fig. 5.16 Structure of Splitter j

$$\eta_{SU} = \sum_{j=1}^L S / 2^{2^{j-1}} = 4S(1 - 4^{-L}) / 3.$$

One DPP is composed of 4 Cells, while one Cell has 2 adders and 1 multiplier. One SU is composed of 2 multipliers. In addition to the DPPs in the Row-PU and cDWT, Arch I has one Auxiliary-PU, which is composed of 6 Cells. Therefore, the total numbers of multipliers η_{Mul} and adders η_{Add} are:

$$\eta_{Mul} = 4\eta_{DPP} + 2\eta_{SU} + 6 = 40S(1 - 4^{-L}) / 3 + 6, \text{ and}$$

$$\eta_{Add} = 2 \times 4 \times \eta_{DPP} + 2 \times 6 = 64S(1 - 4^{-L}) / 3 + 12.$$

The on-chip memory is composed of the pipeline registers, temporal memory and transposition memory. There are η_{DPP} DPPs in all with 4 Cells in each DPP and 2 registers in each Cell (for the 2 outputs). In addition, there are η_{SU} SUs with 2 registers in each SU. Thus, the total number of pipeline registers is:

$$\eta_{PReg} = 4 \times 2 \times \eta_{DPP} + 2 \times \eta_{SU} = 24S(1 - 4^{-L}).$$

The rDWT (Fig. 5.8) of Arch I requires no temporal memory, but the rDWT of Arch j ($j > 1$) has a temporal memory of size $4 \times N / 2^{j-1} = N / 2^{j-3}$ as there are 4 shift registers (Fig. 5.13) and the length of the shift registers is reduced by half each level due to the down sampling of the row coefficients. The temporal memory in the cDWT at Level j has a size of $4 \times (S / 2^{2(j-1)}) \times 2 \times 2^{j-1} = S / 2^{j-4}$ as there are $S / 2^{2(j-1)}$ DPPs, with 4 shift registers in each DPP (Fig. 5.10) and the length of the shift registers is $2 \times 2^{j-1}$ due to the $2^{j-1} L^j$ and H^j interleaved intermediate results entering the cDWT at Level j (Fig. 5.5(b)). Therefore, the total temporal memory size is:

$$\eta_{TReg} = \sum_{j=2}^L 4 \times N / 2^{j-1} + \sum_{j=1}^L S / 2^{j-4} = 4(1 - 2^{1-L})N + 16(1 - 2^{-L})S.$$

For $L = 3$, $\eta_{TReg} = 3N + 14S$.

The transposition memory of each decomposition level is composed of $S / 2^{2(j-1)}$ sets of transposition registers with 1 set for each DPP. Each set of the transposition registers contains $2^{j+1} - 2$ registers at Level j ($j \geq 1$). Therefore, the transposition memory size of the L -level architecture is:

$$\eta_{XReg} = \sum_{j=1}^L (2^{j+1} - 2)S / 2^{2(j-1)} = 8S(2 - 3 \cdot 2^{-L} + 4^{-L}) / 3.$$

The proposed multi-level architecture does not require frame memory, but a splitter (Fig. 5.16) with a register of size $S / 2^{2j-1}$ between every two levels. Thus, the splitters have a total memory size of

$$\eta_{SReg} = \sum_{j=1}^{L-1} S / 2^{2j-1} = 2S(1 - 4^{1-L}) / 3.$$

The latency of the proposed architecture does not depend on the image size but on only the number of decomposition levels L . The total latency at each level is the sum of the latency of its rDWT (τ_r), transposition register (τ_t), cDWT (τ_c) and SU (τ_s). All the rDWTs and cDWTs at all the levels have a latency of 4 cycles due to their DPPs. The SU has a latency of 1 cycle at each level. The level-1 transposition register (Fig. 5.9) has a latency of 2 cycles. The latency for each higher level of transposition register is increased by 2^{j-1} cycles each level (Fig. 5.14). Hence, the total latency of the proposed L -level architecture τ is given below and $\tau = 39$ for $L = 3$.

$$\tau = L\tau_r + L\tau_t + L\tau_c + L\tau_s = 4L + (2L + \sum_{j=2}^L 2^{j-1}) + 4L + L = 11L + 2(2^{L-1} - 1).$$

5.5.2 Performance Comparison

The comparison in terms of multipliers, adders, on-chip memory, critical path delay (CPD), throughput, external bandwidth, latency and computation cycle among the proposed architecture and several well-known existing multi-level architectures [32], [34], [29], [24], [37], [35] are shown in TABLE 5.2. Among the four folded architectures [29], [37], [32] and [35], the architecture in [29] consumes the least multipliers and on-chip memory for $S = 1$. However, the critical path length is larger than that of [32]. When $S = 2$, the design in [29] offers the same throughput as that of [35] while consumes less on-chip memory and multipliers. All these four 3-level folded architectures suffer from a large frame memory of size $N^2 / 4$.

Among all the existing pipelined architectures, the convolution-based architecture by Mohanty and Meher [24] is the most efficient design due to its smallest on-chip memory. Their lifting-based architecture [34] is the second most efficient design among all the existing pipelined designs and the best among the lifting-based designs. The proposed design requires a smaller memory than both of them. The advantage in memory size is even greater when image size becomes larger as $N \gg S$ usually. The proposed architecture is compared with both of them for $S = 8$. For $S = 16$, the proposed architecture is only compared with the lifting-based architecture in [34] as the convolution-based architecture in [24] was reported with only $S = 8$ with a throughput fixed at $N^2 / 16$.

To compare with both the architectures for $S = 8$, the proposed design is modified slightly as the proposed pipelined 3-level architecture requires $S \geq 16$. This is because the processing units at each level of the proposed design are one quarter of its preceding level and at least one DPP

must be used in the rDWT and cDWT of the third level. For a fair comparison of the architectures with the same throughput, the number of DPPs in Arch I, II and III of the proposed design are modified to 8, 2 and 1, respectively, from 16, 4 and 1. In this case, Arch III is connected directly to Arch II without Splitter II and operates at half the clock frequency of Arch I and II so that the throughput is balanced throughout all 3 levels. The comparison on the requirements of arithmetic and memory resources for $N = 512$ is shown in TABLE 5.3, where this work with $S = 8$ is the modified design and this work with $S = 16$ is not modified. Compared to the convolution-based design with $S = 8$ [24], the proposed design requires 38.6%, 36.1% and

TABLE 5.2 COMPARISON OF THEORETICAL ESTIMATION BETWEEN PROPOSED ARCHITECTURE AND EXISTING ARCHITECTURES IN HARDWARE RESOURCES AND TIME COMPLEXITY (FOR 3 LEVEL 9/7 2-D DWT)

Architecture	Lai <i>et al.</i> [32]	Xiong <i>et al.</i> [35]	Tian <i>et al.</i> [37]	Mohanty <i>et al.</i> [29]	Mohanty <i>et al.</i> [34]	Mohanty <i>et al.</i> [24]	This work
Latency	$66 + \frac{7N}{2} + \frac{5N^2}{8}$	$39 + \frac{7N}{4} + \frac{5N^2}{16}$	$39 + \frac{7N}{2S} + \frac{5N^2}{8S}$	$27 + \frac{5N^2}{8S}$	$24 + \frac{67N}{2S}$	38	39
External bandwidth	2	4	$2S$	$2S$	$2S+1$	23	$2S+7$
Throughput	$\frac{2}{T_m}$	$\frac{4}{T_m + 2T_a}$	$\frac{2S}{T_m + 2T_a}$	$\frac{2S}{T_m + 2T_a}$	$\frac{2S}{T_m + 2T_a}$	$\frac{16}{T_m}$	$\frac{2S}{T_m + 2T_a}$
Multiplier	10	18	$10S$	$9S$	$\frac{189S}{16}$	189	$\frac{150S}{8} + 6$
Adder	16	32	$16S$	$16S$	$21S$	294	$21S+12$
On-chip memory	$4N+44$	$55N+6$	$5S+4N+SN$	$4N+20S$	$\frac{89N}{8} + \frac{315S}{16}$	$\frac{21N}{4} + 443$	$3N + \frac{341S}{8}$
Frame memory	$\frac{N^2}{4}$	$\frac{N^2}{4}$	$\frac{N^2}{4}$	$\frac{N^2}{4}$	0	0	0
CPD	T_m	$T_m + 2T_a$	$T_m + 2T_a$	$T_m + 2T_a$	$T_m + 2T_a$	$\approx T_m$	$T_m + T_a$
Computation cycle	$\frac{21N^2}{32}$	$\frac{21N^2}{64}$	$\frac{21N^2}{32}$	$\frac{21N^2}{32}$	$\frac{N^2}{2S}$	$\frac{N^2}{16}$	$\frac{N^2}{2S}$

39.9% less multipliers, adders and on-chip memory respectively. Compared to the lifting-based design with $S = 8$ [34], the proposed design requires marginally more multipliers (17.2%) and adders (6.8%), but significantly less memory (67.9%), resulting in an overall area saving of 55.7% (TABLE 5.4). Compared to the lifting-based design with $S = 16$, the proposed design requires marginally more multipliers (14.3%) and adders (3.6%), but saves about two thirds (63.1%) of the memory, resulting in an overall area saving of 46.2% (TABLE 5.4).

5.5.3 Synthesize Results

As the throughput, the critical path delay, the size of the multiplier, adder and memory are all technology dependent and vary with different implementation technologies, the performance merits listed in TABLE 5.2 and TABLE 5.3 cannot be combined into a single factor for comparison. As such, the proposed architecture and the best existing architectures by Mohanty *et al.* [24], [34] are implemented in the same technology for comparison. The comparisons with the other designs can be found in [24] and [34]. All the designs listed in TABLE 5.3 are modeled and verified with Verilog HDL and used 8-bit input pixel wordlength and 16-bit datapath width, leading to 18- and 20-bit datapath respectively in Arch II and Arch III, to avoid overflow and maintain precision. The designs were all synthesized in the STM 90nm CMOS process with Synopsys Design Compiler. The results in terms of DAT (Data Arrival Time), area, ADP, maximum clock frequency, power (at 50 MHz) and throughput (TP) are tabulated in TABLE 5.4. Compared to the designs by Mohanty and Meher [24], [34] for $S = 8$, the proposed design has a lower ADP by 42.5% and 62.7% respectively. The proposed design (lifting-based) has a comparable throughput with their convolution-based design [24] but a higher throughput than their lifting-based design [34] by 18.8%. Compared with the lifting-based design with $S = 16$ [34], the proposed design has a lower ADP by 54.6% and a higher throughput by 18.6%. Compared with the best current design by Mohanty and Meher [24], the proposed design is 21.5% smaller in area despite the proposed design has 16 processing units whereas their design has only 8. The area saving is the result of the smaller memory needed by the proposed design. The proposed design also achieves 60.2% reduction in ADP and 97.4% increase in throughput.

TABLE 5.3 COMPARISON OF ARITHMETIC RESOURCES AND MEMORY RESOURCES (FOR $N = 512$ AND 3 LEVEL 9/7 DWT)

S	Designs	Multipliers	Adders	On-chip memory
8	Mohanty <i>et al.</i> [24]	189	294	3131
	Mohanty <i>et al.</i> [34]	99	176	5854
	This work	116	188	1882
	Δ over [24]	-38.6%	-36.1%	-39.96%
	Δ over [34]	+17.2%	+6.8%	-67.9%
16	Mohanty <i>et al.</i> [34]	189	336	6011
	This work	216	348	2218
	Δ over [24]	+14.3%	+3.6%	-63.1%

TABLE 5.4 COMPARISON OF SYNTHESIZED RESULTS (FOR $N = 512$ AND 3 LEVEL 9/7 DWT)

S	Designs	DAT (ns)	Area (μm^2)	ADP ($\mu\text{m}^2\text{s}$)	TP (pix./ns)	Max Freq. (MHz)	Power (mW)
8	Mohanty <i>et al.</i> [24]	3.46	2109754	119.6	4.62	289	72.6
	Mohanty <i>et al.</i> [34]	4.17	2702024	184.6	3.84	240	86.4
	This work	3.51	1195925	68.8	4.56	285	35.2
	Δ over [24]	+1.45%	-43.3%	-42.5%	-1.30%	-1.38%	-51.5%
	Δ over [34]	-15.8%	-55.7%	-62.7%	+18.8%	18.8%	-59.3%
16	Mohanty <i>et al.</i> [34]	4.16	3075430	104.8	7.69	240	93.2
	This work	3.51	1655988	47.6	9.12	285	42.8
	Δ over [24] (S=8)	+1.45%	-21.5%	-60.2%	+97.4%	-1.38%	-41.0%
	Δ over [34]	-15.6%	-46.2%	-54.6%	+18.6%	18.8%	-54.1%

5.6 Summary

The existing multi-level DWT architectures are studied in this work. It is observed that the area is dominated by the memory size and the data scanning method has a significant influence on the memory size of the DWT architecture as it decides how the data flows and how the computation is scheduled. It is also observed that the lifting-based architectures generally consume less arithmetic resources than the convolution-based architectures and the pipelined architectures are more memory efficient than the folded architectures. Based on the observation, a novel overlapped stripe-based scanning method for the multi-level decomposition is proposed and a scalable pipelined lifting-based DWT architecture for high throughput is developed. With the newly proposed scanning method for multi-level decomposition, the basic Cell, the processing units (Data Processing Pipe), the row and column DWTs of each level, the transposition memory and the interface between every two levels (Splitter) are designed. Due to the newly proposed scanning method, the size of temporal memory is significantly reduced and the frame buffer is eliminated, resulting in significant saving of memory and consequently the overall size. The synthesis results in 90 nm CMOS process show that the proposed design achieves 60.2% smaller ADP and 97.4% higher throughput for an image size of 512×512 pixels when compared to the best existing design [24]. When compared with the best existing lifting-based design [34], the proposed design is 46.2% smaller in size, 54.6% smaller in ADP and 18.6% higher in throughput.

Chapter 6. Conclusions and Future Works

6.1 Conclusions

The DWT has become an important algorithm in the field of image compression, digital signal processing, computer vision, etc. The hardware implementation of DWT is necessary for many applications in order to fulfill the requirement for high speed, high throughput and low energy consumption. The thesis reviews existing architectures for 2-D DWT and presents several novel hardware architectures for single-level and multi-level 2-D DWT. Two performance metrics, namely memory efficiency and energy efficiency, are identified as the most important metrics in the design of 2-D DWT architecture. Several design strategies and optimization methods are proposed to facilitate the reduction of on-chip memory size and the energy consumption per image.

The design space of 2-D DWT has been explored by many existing works, most of which aim to reduce the on-chip memory size. However, due to the limitation of 2-D DWT, most existing designs have to employ a large size on-chip memory to store the partial results for multiple lines of the input image. As a result, the on-chip memory still dominates the area, especially for large size input. As the size of on-chip memory is unlikely to be further reduced by re-structuring the computation of DWT, it is necessary to develop novel strategies to trade other design properties, such as external bandwidth, critical path length, arithmetic resources, etc., for smaller on-chip memory size. An overlapped stripe-based data scanning method is proposed, the data-flow of 2-D DWT is re-scheduled accordingly and a memory-efficient scalable 2-D DWT architecture is derived. By employing the proposed data scanning method, the outputs of the Row-PU can be input to the Col-PU as soon as they are generated. Consequently, the size of the transposition memory is reduced. For an input image of size $N \times N$, the size of on-chip memory of the proposed architecture is reduced to $3N + 24S$ with an overhead of only one more pixel read every clock cycle. The proposed architecture is composed of S DPPs and has a scalable throughput by varying S . The flipping method is also applied to the proposed architecture to reduce the critical path length without introducing extra arithmetic resource and on-chip memory. Compared with the best existing design [29], the proposed architecture achieves a better ADP by 32.3%, 31.5% and 27.0% for $S = 2, 4$ and 8 , respectively.

Different from the existing architectures which improve the memory efficiency by reducing the number of words of the on-chip memory, a wordlength optimization is developed to reduce the wordlength of the on-chip memory. Instead of assuming a fixed wordlength of data path, the wordlength of every signal of the DWT architecture is optimized for a given precision of the output. Observing that the memory for storing the three partial results are large, area-dominating and not optimized, the wordlength optimization is performed with priority given to the three partial results over the other signals of the data path. While keeping the overall proposed architecture unchanged, a new objective function is introduced with larger weight on the fractional bit of the three partial results to optimize their fractional bit and integer bit before they are stored. As a result, not only is the memory wordlength reduced, resulting in smaller overall memory size, the arithmetic resource and the critical path length are also reduced due to the optimized data path. Consequently, both the total design area and the delay are reduced without compromising the precision of the outputs. The synthesized design in 90-nm CMOS process exhibits a 25.1% improvement in ADP when compared with the design without wordlength optimization.

Instead of focusing on improving hardware utilization and memory efficiency, an energy- and area-efficient parameterized 2-D DWT architecture is proposed and energy efficiency is considered the key performance metric. Because the external memory and the on-chip memory dominate the total energy consumption, an overlapped block-based image scanning method is proposed to optimize the number of external memory reads and the on-chip memory size. In the proposed method, the height of block and the parallelism are the two algorithm-mapping parameters that characterize the design space. The most energy-efficient design is determined by comparing the design points with various height of block and parallelism. A DARM activation schedule method is proposed to reduce the energy consumption of external DRAM. The DRAMs are scheduled to the pre-charge power-down mode when not being accessed, resulting a lower background power dissipation. By employing the overlapped block-based scanning method with the optimized algorithm-mapping parameters, the number of external memory reads of the proposed architecture is reduced by up to 30% when compared with the state-of-the-art design. Consequently, the energy efficiency of the proposed architecture is improved by up to 269%. Additionally, compared with the existing designs, the proposed design also consumes less on-chip memory and thus has a EAT ratio as small as 0.02.

Compared with the single-level 2-D DWT architecture, the on-chip memory of the multi-level DWT architecture accounts a even larger proportion of the total area. A memory-efficient high-throughput scalable architecture for multi-level 2-D DWT architecture is proposed. Because the data scanning method has a significant impact on the memory efficiency of the architecture, the overlapped stripe-based data scanning method is adopted to the multi-level 2-D DWT architecture. Unlike the single-level DWT, there is a 7-column overlap between the adjacent stripes for the multi-level DWT. As a result, the temporal memory of the first-level DWT is eliminated with an overhead of 7 pixels reads every clock cycle and an auxiliary-PU, which is composed of 6 multipliers and 12 adders. The scanning method is also extended to the higher-level DWTs and the computations are re-structured to achieve 100% hardware utilization. The proposed architecture has a scalable throughput, constant latency and simple control. Compared with the best existing design, the on-chip memory size is reduced by more than 39%. In addition, as lifting-based algorithm is chosen, the arithmetic resource is also reduced by approximately 36% compared with convolution-based design. The synthesis results in 90 nm CMOS process show that the proposed architecture achieves a better ADP by 60% and higher throughput by 97% when compared to the best existing design for 9/7 3-level 2-D DWT.

In summary, several energy- and memory-efficient 2-D DWT architectures, including a single level memory-efficient 2-D DWT with the wordlength of its data path optimized, an energy-efficient 2-D DWT architecture that can be optimized for both 5/3 and 9/7 filters and various image size, and a memory-efficient multi-level 2-D DWT architecture, were proposed and implemented in this work.

6.2 Future Works

Although there are numerous existing DWT architectures and the design space has already been widely explored, new problems emerge as new technologies evolve. There are still potential research areas to be explored. Some possible topics are discussed below.

6.2.1 Evaluation of Memory and Energy Efficiency Based on Fully Layout Designs

The experimental results of the proposed DWT architectures in Chapter 3, 4 and Chapter 5 are all based on synthesized designs and not post-layout results. However, wiring, manufacturing process and loading will influence on the area, delay and energy consumptions. Besides, the energy consumption of DRAM can be more accurate if the results are based on measurement

instead of evaluation. Therefore, in order to obtain more accurate results, the architectures can be fabricated to obtain measurement results and the energy consumption of DRAM can also be verified with actual measurement.

6.2.2 Energy-Efficient Architecture for Multi-Level 2-D DWT Architecture

As the size of CMOS transistor scales down, the transistor density continues to double every two years, while the transistor energy efficiency only improves by 1.4 times, causing large under-clocked silicon area – hence the term “dark silicon” [99], [100], [101], [102]. Therefore, in the era of dark silicon, the silicon area is considered as a much cheaper resource when compared with the energy efficiency. The rise of dark silicon is driving a novel architecture design technique that trades silicon area for energy efficiency. Most existing works of multi-level DWT architecture focus on improving the memory efficiency for smaller silicon area while little effort is spent on improving the energy efficiency. The design technique developed in Chapter 4 can be adopted to multi-level DWT architecture for higher energy efficiency. Different from the single-level DWT architecture, all levels of DWT processing have to be considered when developing the data scanning method. The throughput at each level of the DWT has to be balanced for high hardware utilization rate. The energy consumption of higher level DWT has to be included in the evaluation of the energy efficiency. To overcome the problem, the energy efficiency of various multi-level 2-D DWT architectures can be compared in order to determine an energy-optimized design strategy. A larger design space can be defined to determine the best design point by varying multiple algorithm-mapping parameters.

6.2.3 Memory- and Energy-Efficient 3-D DWT Architecture

3-D DWT has been applied on various applications such as video compression, video watermarking, volumetric image compression, etc. As 3-D DWT is even more computational and memory intensive than the 2-D DWT, it requires even larger silicon area and energy consumption. Several hardware architectures have been proposed in [67], [68], [69]. In addition to the rDWT and cDWT, a temporal-wise DWT (tDWT) needs to be performed based on the coefficients generated by the rDWT and cDWT. As a result, a large spatial memory is needed to store the coefficients. The design methodologies proposed in this work for improving the memory and energy efficiency can be extended to the design of 3-D DWT.

6.2.4 Model-Based Design Space Exploration Methodology for Energy Efficiency

Energy efficiency has become a key performance metric for computing. For a specific computation kernel, the design space is characterized by several algorithm-mapping parameters such as data sequencing, throughput, parallelism, external bandwidth, memory hierarchy, etc. In order to determine the most energy efficiency design for a computation kernel, a highly parameterized design is necessary to compare the energy efficiency among many design points with various design trade-offs. However, as the number of algorithm-mapping parameter increases, the design space expands exponentially and makes it impossible to evaluate the energy efficiency of every design points by synthesizing all the designs. One solution to this problem is to develop an energy performance model for the computation kernel. Instead of handwriting HDL code and synthesizing every design, the energy efficiency can be rapidly evaluated by the energy performance model. In order to build the energy performance model, the computation kernel needs to be decomposed into several algorithm-architecture pairs. The basic building blocks, such as multipliers, adder, multiplexer, etc., of the algorithm-architecture pair is implemented and the energy efficiency is first evaluated by the conventional RTL flow. Given the energy efficiency of basic building blocks and algorithm-mapping parameters, the energy performance model can be built for the exploration of the design space for energy-efficient DWT architecture.

6.2.5 Memory- and Energy-Efficient Architecture for Machine Learning Applications

The design strategies and optimization methods proposed in this work can be extended to other classes of widely applied algorithms such as machine learning algorithms. Recently, the deep learning algorithm [103], [104], [105] has attracted much attention in the research community, due to its state-of-the-art performance in many domains such as speech recognition, collaborative filtering, computer vision, etc. The convolutional neural network (CNN) is a well-established deep learning architecture, which is composed of multiple layers for feature maps and classification. For each layer, the 2-D convolution is performed to the output of its preceding layer. Due to its intensive demand on computation, hardware implementation is preferred to improve the memory and energy efficiency of such widely applied algorithms. The memory hierarchy between adjacent layers can be optimized to reduce the memory size and improve the energy efficiency. Within each layer, the design space of the 2-D convolution module can be

explored to find the best design point for CNN architecture. This research could lead to a high throughput, energy efficient CNN architecture suitable for various applications.

Bibliography

- [1] S. G. Mallat, "A wavelet tour of signal processing: the Sparse way," Elsevier /Academic Press, 2009.
- [2] M. Dyer, S. Nooshabadi and D. Taubman, "Design and Analysis of System on a Chip Encoder for JPEG2000," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 2, pp. 215 - 225, 2009.
- [3] K. A. Kotteri, A. E. Bell and J. E. Carletta, "Design of multiplierless, high-performance, wavelet filter banks with image compression applications," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 51, no. 2, pp. 483 - 494, 2004.
- [4] L. Liu, N. Chen, H. Meng, L. Zhang, Z. Wang and H. Chen, "A VLSI architecture of JPEG2000 encoder," *Solid-State Circuits, IEEE Journal of*, vol. 39, no. 11, pp. 2032-2040, 2004.
- [5] T. W. Fry and S. A. Hauck, "SPIHT image compression on FPGAs," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 15, no. 9, pp. 1138 - 1147, 2005.
- [6] M. Charrier, D. S. Cruz and M. Larsson, "JPEG2000, the next millennium compression standard for still images," in *Multimedia Computing and Systems, 1999. IEEE International Conference on*.
- [7] A. N. Akansu, W. A. Serdijn and I. W. Selesnick, "Emerging applications of wavelets: A review," *Physical Communication*, vol. 3, no. 1, pp. 1 - 18, 2010.

- [8] C. Van Bouwel, J. Potemans, S. Schepers, B. Nauwelaers and A. Van de Capelle, "Wavelet packet based multicarrier modulation," in *Communications and Vehicular Technology, 2000. SCVT-200. Symposium on*.
- [9] S. Galli, H. Koga and N. Kodama, "Advanced signal processing for PLCs: Wavelet-OFDM," in *Power Line Communications and Its Applications, 2008. ISPLC 2008. IEEE International Symposium on*, 2008.
- [10] A. Laine and J. Fan, "Texture classification by wavelet packet signatures," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 15, no. 11, pp. 1186 - 1191, 1993.
- [11] J. Chilo and T. Lindblad, "Hardware implementation of 1D wavelet transform on an FPGA for infrasound signal classification," *Nuclear Science, IEEE Transactions on*, vol. 55, no. 1, pp. 9-13, 2008.
- [12] E. Loupias, N. Sebe, S. Bres and J. M. Jolion, "Wavelet-based salient points for image retrieval," in *Image Processing, 2000. Proceedings. 2000 International Conference on*, 2000.
- [13] L. Zhang, W. Zhou and L. Jiao, "Wavelet support vector machine," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 1, pp. 34 - 39, 2004.
- [14] H. He and J. A. Starzyk, "A self-organizing learning array system for power quality classification based on wavelet transform," *Power Delivery, IEEE Transactions on*, vol. 21, no. 1, pp. 286 - 295, 2006.
- [15] G. Van de Wouwer, P. Scheunders and D. Van Dyck, "Statistical texture characterization from discrete wavelet representations," *Image Processing, IEEE Transactions on*, vol. 8, no. 4, pp. 592 - 598, 1999.
- [16] T. Acharya and C. Chakrabarti, "A survey on lifting-based discrete wavelet transform architectures," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 42, no. 3, pp. 321 - 339, 2006.

- [17] "The Green500 List – November 2013," [Online]. Available: <http://www.green500.org/news/green500-list-november-2013>.
- [18] J. Fridman and E. S. Manolakos, "On the scalability of 2-D discrete wavelet transform algorithms," *In Multidimensional Filter Banks and Wavelets*, vol. 8, pp. 185-218, 1997.
- [19] N. D. Zervas, G. P. Anagnostopoulos, V. Spiliotopoulos, Y. Andreopoulos and C. E. Goutis, "Evaluation of design alternatives for the 2-D-discrete wavelet transform," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 11, no. 12, pp. 1246-1262, 2001.
- [20] K. C. Hung, Y. S. Hung and Y. J. Huang, " A nonseparable VLSI architecture for two-dimensional discrete periodized wavelet transform," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 9, no. 5, pp. 565-576, 2001.
- [21] P. Y. Chen, "VLSI implementation for one-dimensional multilevel lifting-based wavelet transform," *Computers, IEEE Transactions on*, vol. 53, no. 4, pp. 386-398, 2004.
- [22] P. C. Tseng, C. T. Huang and L. G. Chen, " Reconfigurable discrete wavelet transform processor for heterogeneous reconfigurable multimedia systems," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 41, no. 1, pp. 35-47, 2005.
- [23] C. H. Hsia, J. M. Guo and J. S. Chiang, "Improved low-complexity algorithm for 2-D integer lifting-based discrete wavelet transform using symmetric mask-based scheme," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 8, pp. 1202-1208, 2009.
- [24] B. K. Mohanty and P. K. Meher, "Memory-Efficient High-Speed Convolution-Based Generic Structure for Multilevel 2-D DWT," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 23, no. 2, pp. 353- 363, 2013.

- [25] G. Shi, W. Liu, L. Zhang and F. Li, "An efficient folded architecture for lifting-based discrete wavelet transform," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 56, no. 4, pp. 290-294, 2009.
- [26] M. E. Angelopoulou, K. Masselos, P. Y. Cheung and Y. Andreopoulos, "Implementation and comparison of the 5/3 lifting 2D discrete wavelet transform computation schedules on FPGAs," *Journal of Signal Processing Systems*, vol. 51, no. 1, pp. 3-21, 2008.
- [27] C. Wang and W. S. Gan, "Efficient VLSI architecture for lifting-based discrete wavelet packet transform," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 54, no. 5, pp. 422-426, 2007.
- [28] H. Liao, M. K. Mandal and B. F. Cockburn, "Efficient architectures for 1-D and 2-D lifting-based wavelet transforms," *Signal Processing, IEEE Transactions on*, vol. 52, no. 5, pp. 1315-132, 2004.
- [29] B. K. Mohanty, A. Mahajan and P. K. Meher, "Area- and Power-Efficient Architecture for High-Throughput Implementation of Lifting 2-D DWT," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 59, no. 7, pp. 434-438, 2012.
- [30] C.-T. Huang, P.-C. Tseng and L.-G. Chen, "Flipping structure: an efficient VLSI architecture for lifting-based discrete wavelet transform," *Signal Processing, IEEE Transactions on*, vol. 52, no. 4, pp. 1080-1089, 2004.
- [31] C.-T. Huang, P.-C. Tseng and L.-G. Chen, "Analysis and VLSI architecture for 1-D and 2-D discrete wavelet transform," *Signal Processing, IEEE Transactions on*, vol. 53, no. 4, pp. 1575-1586, 2005.
- [32] Y.-K. Lai, L.-F. Lien and Y.-C. Shih, "A high-performance and memory-efficient VLSI architecture with parallel scanning method for 2-D lifting-based discrete wavelet transform," *Consumer Electronics, IEEE Transactions on*, vol. 55, no. 2, pp. 400-407, 2009.

- [33] C.-Y. Xiong, J. Tian and J. Liu, "Efficient high-speed/low-power line-based architecture for two-dimensional discrete wavelet transform using lifting sche," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 16, no. 2, pp. 309-316, 2006.
- [34] B. K. Mohanty and P. K. Meher, "Memory Efficient Modular VLSI Architecture for Hightthroughput and Low-Latency Implementation of Multilevel Lifting 2-D DWT," *Signal Processing, IEEE Transactions on*, vol. 59, no. 5, pp. 2072-2084, 2011.
- [35] C.-Y. Xiong, J. Tian and J. Liu, "Efficient Architectures for Two-Dimensional Discrete Wavelet Transform Using Lifting Scheme," *Image Processing, IEEE Transactions on*, vol. 16, no. 3, pp. 607-614, 2007.
- [36] W. Zhang, Z. Jiang, Z. Gao and Y. Liu, "An Efficient VLSI Architecture for Lifting-Based Discrete Wavelet Transform," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 59, no. 3, pp. 158-162, 2012.
- [37] X. Tian, L. Wu, Y.-H. Tan and J.-W. Tian, "Efficient Multi-Input/Multi-Output VLSI Architecture for Two-Dimensional Lifting-Based Discrete Wavelet Transform," *Computers, IEEE Transactions on*, vol. 60, no. 8, pp. 1207-1211, 2011.
- [38] M. Weeks and M. Bayoumi, "Discrete Wavelet Transform: Architectures, Design and Performance Issues," *The Journal of VLSI Signal Processing*, vol. 35, no. 2, pp. 155-178, 2003.
- [39] P. K. Meher, B. K. Mohanty and J. C. Patra, "Hardware-Efficient Systolic-Like Modular Design for Two-Dimensional Discrete Wavelet Transform," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 55, no. 2, pp. 151-155, 2008.
- [40] Y. H. Seo and D. W. Kim, "VLSI Architecture of Line-Based Lifting Wavelet Transform for Motion JPEG2000," *Solid-State Circuits, IEEE Journal of*, vol. 42, no. 2, pp. 431-440, 2007.
- [41] C.-C. Cheng, C.-T. Huang, C.-Y. Ching, C.-J. Chung and L.-G. Chen, "On-Chip Memory Optimization Scheme for VLSI Implementation of Line-Based Two-Dimentional

- Discrete Wavelet Transform," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17, no. 7, pp. 814-822, 2007.
- [42] B.-F. Wu and C.-F. Chung, "A high-performance and memory-efficient pipeline architecture for the 5/3 and 9/7 discrete wavelet transform of JPEG2000 codec," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 15, no. 12, pp. 1615-1628, 2005.
 - [43] C.-T. Huang, P.-C. Tseng and L.-G. Chen, "Generic RAM-based architectures for two-dimensional discrete wavelet transform with line-based method," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 15, no. 7, pp. 910-920, 2005.
 - [44] X.-G. Lan, N.-N. Zheng and Y.-H. Liu, "Low-power and high-speed VLSI architecture for lifting-based forward and inverse wavelet transform," *Consumer Electronics, IEEE Transactions on*, vol. 51, no. 2, pp. 379-385, 2005.
 - [45] G. Dillen, B. Georis, J. D. Legat and O. Cantineau, "Combined line-based architecture for the 5-3 and 9-7 wavelet transform of JPEG2000," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 9, pp. 944-950, 2003.
 - [46] K. Andra, C. Chakrabarti and T. Acharya, "A VLSI architecture for lifting-based forward and inverse wavelet transform," *Signal Processing, IEEE Transactions on*, vol. 50, no. 4, pp. 966-977, 2002.
 - [47] W. Jiang and A. Ortega, "Lifting factorization-based discrete wavelet transform architecture design," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 11, no. 5, pp. 651-657, 2001.
 - [48] M. Martina and G. Masera, "Multiplierless, Folded 9/7 - 5/3 Wavelet VLSI Architecture," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 54, no. 9, pp. 770-774, 2007.
 - [49] C. Cheng and K. K. Parhi, "High-Speed VLSI Implementation of 2-D Discrete Wavelet Transform," *Signal Processing, IEEE Transactions on*, vol. 56, no. 1, pp. 393-403, 2008.

- [50] C. Zhang, C. Wang and M. O. Ahmad, "A Pipeline VLSI Architecture for Fast Computation of the 2-D Discrete Wavelet Transform," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 59, no. 8, pp. 1775-1785, 2012.
- [51] C. Zhang, C. Wang and M. O. Ahmad, "A Pipeline VLSI Architecture for High-Speed Computation of the 1-D Discrete Wavelet Transform," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 57, no. 10, pp. 2729-2740, 2010.
- [52] S. Masud and J. V. McCanny, "Reusable silicon IP cores for discrete wavelet transform applications," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 51, no. 6, pp. 1114-1124, 2004.
- [53] P.-C. Wu and L.-G. Chen, "An efficient architecture for two-dimensional discrete wavelet transform," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 11, no. 4, pp. 536-545, 2001.
- [54] F. Marino, "Two fast architectures for the direct 2-D discrete wavelet transform," *Signal Processing, IEEE Transactions on*, vol. 49, no. 6, pp. 1248-1259, 2001.
- [55] F. Marino, "Efficient high-speed/low-power pipelined architecture for the direct 2-D discrete wavelet transform," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 47, no. 12, pp. 1476-1491, 2000.
- [56] F. Marino, D. Guevorkian and J. T. Astola, "Highly efficient high-speed/low-power architectures for the 1-D discrete wavelet transform," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 47, no. 12, pp. 1492-1502, 2000.
- [57] F. Marino, "A double-face bit-serial architecture for the 1D discrete wavelet transform," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 47, no. 1, pp. 65-71, 2000.

- [58] M. Ferretti and D. Rizzo, "Handling borders in systolic architectures for the 1-D discrete wavelet transform for perfect reconstruction," *Signal Processing, IEEE Transactions on*, vol. 48, no. 5, pp. 1365-1378, 2000.
- [59] J. T. Kim, Y. H. Lee, T. Isshiki and H. Kunieda, "Scalable VLSI architectures for lattice structure-based discrete wavelet transform," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 45, no. 8, pp. 1031-1043, 1998.
- [60] T. C. Denk and K. K. Parhi, "VLSI architectures for lattice structure based orthonormal discrete wavelet transforms," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 44, no. 2, pp. 129-132, 1997.
- [61] J. Fridman and E. S. Manolakos, "Discrete wavelet transform: data dependence analysis and synthesis of distributed memory and control array architectures," *Signal Processing, IEEE Transactions on*, vol. 45, no. 5, pp. 1291-1308, 1997.
- [62] A. Grzeszczak, M. K. Mandal and S. Panchanathan, "VLSI implementation of discrete wavelet transform," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 4, no. 4, pp. 421-433, 1996.
- [63] C. Chakrabarti and M. Vishwanath, "Efficient realizations of the discrete and continuous wavelet transforms: from single chip implementations to mappings on SIMD array computers," *Signal Processing, IEEE Transactions on*, vol. 43, no. 3, pp. 759-771, 1995.
- [64] M. Vishwanath, "The recursive pyramid algorithm for the discrete wavelet transform," *Signal Processing, IEEE Transactions on*, vol. 42, no. 3, pp. 673-676, 1994.
- [65] M. Vishwanath, R. M. Owens and M. J. Irwin, "VLSI architectures for the discrete wavelet transform," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 42, no. 5, pp. 305-316, 1995.
- [66] K. K. Parhi and T. Nishitani, "VLSI architectures for discrete wavelet transforms," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 1, no. 2, pp. 191-202, 1993.

- [67] A. Das, A. Hazra and S. Banerjee, "An Efficient Architecture for 3-D Discrete Wavelet Transform," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, no. 2, pp. 286-296, 2010.
- [68] B. K. Mohanty and P. K. Meher, "Parallel and Pipeline Architectures for High-Throughput Computation of Multilevel 3-D DWT," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, no. 9, pp. 1200-1209, 2010.
- [69] B. K. Mohanty and P. K. Meher, "Memory-Efficient Architecture for 3-D DWT Using Overlapped Grouping of Frames," *Signal Processing, IEEE Transactions on*, vol. 59, no. 11, pp. 5605-5616, 2011.
- [70] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *Journal of Fourier Analysis and Applications*, vol. 4, no. 3, pp. 247-269, 1998.
- [71] J. Fowers, G. Brown, J. Wernsing and G. Stitt, "A performance and energy comparison of convolution on GPUs, FPGAs, and multicore processors," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 9, no. 4, 2013.
- [72] B. Betkaoui, D. B. Thomas and W. Luk, "Comparing performance and energy efficiency of FPGAs and GPUs for high productivity computing," in *Field-Programmable Technology (FPT), 2010 International Conference on*, 2010.
- [73] J. W. Jang, S. B. Choi and V. K. Prasanna, "Energy- and time-efficient matrix multiplication on FPGAs," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 13, no. 11, pp. 1305-1319, 2005.
- [74] Y. S. Shao, B. Reagen, G. Y. Wei and D. Brooks, "Aladdin: A pre-RTL, power-performance accelerator simulator enabling large design space exploration of customized architectures," in *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*.

- [75] R. Chen, H. Le and V. K. Prasanna, "Energy Efficient Parameterized FFT Architecture," in *Field Programmable Logic and Applications (FPL)*, 2013 23rd International Conference on.
- [76] R. Chen and V. K. Prasanna, "Energy-efficient architecture for stride permutation on streaming data," in *Reconfigurable Computing and FPGAs (ReConFig)*, 2013 International Conference on.
- [77] Y. Hu and C. C. Jong, "A Memory-Efficient Scalable Architecture for Lifting-Based Discrete Wavelet Transform," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 60, no. 8, pp. 502-506, 2013.
- [78] Y. Hu and C. C. Jong, "Memory-Efficient Discrete Wavelet Transform Architecture Based on Wordlength Optimization," in *IEEE International Symposium on Circuits and Systems (ISCAS '2015)*, Lisbon, May 2015.
- [79] Y. Hu and V. K. Prasanna, "Energy- and Area-Efficient Parameterized Lifting-Based 2-D DWT Architecture on FPGA," in *IEEE 18th International Conference on High Performance Extreme Computing (HPEC'14)*, September 2014.
- [80] Y. Hu and V. K. Prasanna, "Energy-Efficient Parameterized 2-D Separable Convolution on FPGA," in *IEEE 5th International Green Computing Conference (IGCC'14)*, November 2014.
- [81] S. R. Kuppannagari, Y. Hu and V. K. Prasanna, "High Level Performance Model Based Design Space Exploration for Energy-Efficient Designs on FPGAs," in *IEEE 5th International Green Computing Conference (IGCC'14)*, November 2014.
- [82] Y. Hu and C. C. Jong, "A Memory-Efficient High-Throughput Architecture for Lifting-Based Multi-Level 2-D DWT," *Signal Processing, IEEE Transactions on*, vol. 61, no. 30, pp. 4975-4987, 2013.

- [83] W. Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets," *Applied and computational harmonic analysis*, vol. 3, no. 2, pp. 186-200, 1996.
- [84] M. Vetterli and H. Cormac, "Wavelets and filter banks: Theory and design," *Signal Processing, IEEE Transactions on*, vol. 40, no. 9, pp. 2207-2232, 1992.
- [85] G. Strang and N. Truong, Wavelets and filter banks, SIAM, 1996.
- [86] M.-Y. Chiu, K.-B. Lee and C.-W. Jen, "Optimal data transfer and buffering schemes for JPEG2000 encoder," in *Signal Processing Systems. IEEE Workshop on* , 2013.
- [87] D.-U. Lee, L.-W. Kim and J. D. Villasenor, "Precision-Aware Self-Quantizing Hardware Architectures for the Discrete Wavelet Transform," *Image Processing, IEEE Transactions on* , vol. 21, no. 2, pp. 768 - 777 , 2012.
- [88] D.-U. Lee and J. Villasenor, "A Bit-Width Optimization Methodology for Polynomial-Based Function Evaluation," *Computers, IEEE Transactions on* , vol. 56, no. 4, pp. 567 - 571, 2007.
- [89] L. Ingber, "Adaptive Simulated Annealing (ASA) 25.15," 2004. [Online]. Available: <http://www.ingber.com/#ASA>.
- [90] K. Varma and A. Bell, "JPEG2000 - Choices and tradeoffs for encoders," *Signal Processing Magazine, IEEE* , vol. 21, no. 6, pp. 70 - 75 , 2004.
- [91] L. Davis, Handbook of genetic algorithms, New York: Van Nostrand Reinhold., 1991.
- [92] "Simulated annealing," [Online]. Available: http://en.wikipedia.org/wiki/Simulated_annealing.
- [93] R. Eglese, "Simulated annealing: A tool for operational research," *European Journal of Operational Research*, vol. 46, no. 3, pp. 271 - 281, 1990.

- [94] "Micron DDR3 SDRAM System-Power Calculator," [Online]. Available: <http://www.micron.com/products/support/powercalc>.
- [95] K. T. Malladi, I. Shaeffer, L. Gopalakrishnan, D. Lo, B. C. Lee and M. Horowitz, "Rethinking DRAM Power Modes for Energy Proportionality," in *2012 45th Annual IEEE/ACM International Symposium on*, 2012.
- [96] G. A. Davidson, M. A. Isnardi, L. D. Fielder, M. S. Goldman and C. C. Todd, "ATSC Video and Audio Coding," *Proceedings of the IEEE*, vol. 94, no. 1, pp. 60- 76, 2006.
- [97] G. J. Sullivan, J. Ohm, W.-J. Han and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1649-1668, 2012.
- [98] "Micron DDR3 SDRAM System-Power Calculator," [Online]. Available: <http://www.micron.com/products/support/power-calc..>
- [99] H. Esmaeilzadeh, "Dark silicon and the end of multicore scaling," in *Computer Architecture (ISCA), 38th Annual International Symposium on*, 2011.
- [100] M. Taylor, "A Landscape of the New Dark Silicon Design Regime," *Micro, IEEE*, vol. 33, no. 5, pp. 8 - 19, 2013.
- [101] H. Esmaeilzadeh, E. Blem, R. St Amant, K. Sankaralingam and D. Burger, "Power limitations and dark silicon challenge the future of multicore," *ACM Transactions on Computer Systems (TOCS)*, vol. 30, no. 3, p. 11, 2012.
- [102] N. Hardavellas, M. Ferdman, B. Falsafi and A. Ailamaki, "Toward dark silicon in servers," *IEEE Micro*, vol. 31, pp. 6-15, 2013.
- [103] G. Hinton, S. Osindero and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527-1554, 2006.

- [104] Y. Bengio, "Learning deep architectures for AI," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1-127, 2009.
- [105] X.-W. Chen and X. Lin, "Big Data Deep Learning: Challenges and Perspectives," *Access, IEEE*, vol. 2, pp. 514 - 525, 2014.