Efficient Reconfigurable Architectures for 3-D Medical Image Compression

A thesis submitted for the degree of
Doctor of Philosophy

by

Afandi Ahmad

Supervisor

Dr Abbes Amira

Department Electronic and Computer Engineering
School of Engineering and Design
Brunel University, West London

July 2010

Abstract

Recently, the more widespread use of three-dimensional (3-D) imaging modalities, such as magnetic resonance imaging (MRI), computed tomography (CT), positron emission tomography (PET), and ultrasound (US) have generated a massive amount of volumetric data. These have provided an impetus to the development of other applications, in particular telemedicine and teleradiology. In these fields, medical image compression is important since both efficient storage and transmission of data through high-bandwidth digital communication lines are of crucial importance.

Despite their advantages, most 3-D medical imaging algorithms are computationally intensive with matrix transformation as the most fundamental operation involved in the transform-based methods. Therefore, there is a real need for high-performance systems, whilst keeping architectures flexible to allow for quick upgradeability with real-time applications. Moreover, in order to obtain efficient solutions for large medical volumes data, an efficient implementation of these operations is of significant importance. Reconfigurable hardware, in the form of field programmable gate arrays (FPGAs) has been proposed as viable system building block in the construction of high-performance systems at an economical price. Consequently, FPGAs seem an ideal candidate to harness and exploit their inherent advantages such as massive parallelism capabilities, multimillion gate counts, and special low-power packages.

The key achievements of the work presented in this thesis are summarised as follows. Two architectures for 3-D Haar wavelet transform (HWT) have been proposed based on transpose-based computation and partial reconfiguration suitable for 3-D medical imaging applications. These applications require continuous hardware servicing, and as a result dynamic partial reconfiguration (DPR) has been introduced. Comparative study for both non-partial and partial reconfiguration implementation has shown that DPR offers many advantages and leads to a compelling solution for implementing computationally intensive applications such as 3-D medical image compression. Using DPR, several large systems are mapped to small hardware

resources, and the area, power consumption as well as maximum frequency are optimised and improved.

Moreover, an FPGA-based architecture of the finite Radon transform (FRAT) with three design strategies has been proposed: direct implementation of pseudo-code with a sequential or pipelined description, and block random access memory (BRAM)-based method. An analysis with various medical imaging modalities has been carried out. Results obtained for image de-noising implementation using FRAT exhibits promising results in reducing Gaussian white noise in medical images. In terms of hardware implementation, promising trade-offs on maximum frequency, throughput and area are also achieved.

Furthermore, a novel hardware implementation of 3-D medical image compression system with context-based adaptive variable length coding (CAVLC) has been proposed. An evaluation of the 3-D integer transform (IT) and the discrete wavelet transform (DWT) with lifting scheme (LS) for transform blocks reveal that 3-D IT demonstrates better computational complexity than the 3-D DWT, whilst the 3-D DWT with LS exhibits a lossless compression that is significantly useful for medical image compression. Additionally, an architecture of CAVLC that is capable of compressing high-definition (HD) images in real-time without any buffer between the quantiser and the entropy coder is proposed. Through a judicious parallelisation, promising results have been obtained with limited resources.

In summary, this research is tackling the issues of massive 3-D medical volumes data that requires compression as well as hardware implementation to accelerate the slowest operations in the system. Results obtained also reveal a significant achievement in terms of the architecture efficiency and applications performance.

Certificate of Originality

"I hereby certify that the work presented in this thesis is my original research and has not been presented for a higher degree at any other university or institute".

.....

(Afandi Ahmad – 0729168)

Afandi.Ahmad@brunel.ac.uk

23 July 2010

To everyone who supports me, it just begins...

Acknowledgements

I gratefully acknowledge who has supported me throughout my PhD work and finally the preparation of this thesis. In particular, I would like to thank my supervisor, Dr Abbes Amira, for accepting me as a student, whilst I am in a dark tunnel. Abbes, many thanks for your relentless commitment you have shown along this journey. You could have taken the short cut, but you stood by me. I will always remember and look to you as an example of how to work hard! I would also like to thank Dr Hassan Rabah, Dr Yves Berviller and Dr David Smith for insightful guidance of my research studies. All your effort shows me the real meaning of sincerity!

I would like to thank all the great friends I have had at Brunel, Ulster and Nancy over the years: Ben, Abdallah, Michael, Hairol, Thian, Linda, Khalid, Shafinar and Aida. Thank you all for providing many happy memories and truly friendship colours. I wish to thank my sponsors, Ministry of Higher Education Malaysia (MOHE), Universiti Tun Hussein Onn Malaysia (UTHM) and the British Council. I must also thank to Faculty of Electrical and Electronic Engineering, especially to colleagues in Department of Computer Engineering for their support.

I would achieve nothing without the encouragement and compassion I received from my understanding wife, parents and all of my families. This thesis is dedicated to them. Their love and support kept me going. I owe you!

Author's Publications

Journal Papers – Accepted

1. **A. Ahmad**, B. Krill, A. Amira, and H. Rabah, "Efficient Architectures for 3-D HWT using Dynamic Partial Reconfiguration", *Elsevier Journal of System Architecture - Special Issue on Hardware/Software Co-Design*, ISSN 1383-7621, Volume 56, Issue 8, pp. 305–316, August 2010.

2. B. Krill, **A. Ahmad**, A. Amira, and H. Rabah, "An Efficient FPGA-based Dynamic Partial Reconfiguration Framework for Image and Signal Processing IP Cores", *Elsevier Journal of Signal Processing: Image Communication - Special Issue on Breakthrough Architectures for Image and Video Systems*, ISSN 0923-5965, Volume 25, Issue 5, pp. 377–387, May 2010.

3. P. Nicholl, **A. Ahmad**, and A. Amira, "A Novel Feature Vectors Construction Approach for Face Recognition", *Springer Transactions on Computational Science (TCS) - Special Issue on "Security in Computing".*

Journal Papers – In Revision

1. M. Guarisco, **A. Ahmad**, H. Rabah, A. Amira, and Y. Berviller, "3-D Medical Image Compression System using CAVLC", *IEEE Transactions on Medical Imaging.*

Journal Papers – Under Review

1. M. Guarisco, **A. Ahmad**, H. Rabah, A. Amira, and Y. Berviller, "FPGA-based Implementation of a CAVLC for 3-D Medical Compression", *IEEE Transactions on Consumer Electronics".*

2. **A. Ahmad**, A. Amira, and M. Jiang , "Reconfigurable Architectures for 3-D Medical Image Processing: Design Issue and Challenges", *ACM Computing Surveys.*

3. **A. Ahmad**, A. Amira, H. Rabah, and Y. Berviller, "An Efficient FPGA-based Architecture of Finite Radon Transform for Medical Imaging Application", *IEEE Transactions on Medical Imaging.*

Conference Papers – Accepted

1. **A. Ahmad**, A. Amira, M. Guarisco, H. Rabah and Y. Berviller, "Efficient Implementation of a 3-D Medical Imaging Compression System using CAVLC", *The 2010 International Conference on Image Processing (ICIP)*, September 26^{th} - 29^{th} 2010, Hong Kong.

2. **A. Ahmad**, B. Krill, A. Amira, and H. Rabah, "3-D Haar Wavelet Transform with Dynamic Partial Reconfiguration for 3-D Medical Image Compression", *The IEEE Biomedical Circuits and Systems Conferences (BIOCAS)*, November 26^{th} - 28^{th} 2009, Beijing, China, pp. 137–140.

3. **A. Ahmad** and A. Amira, "Efficient Reconfigurable Architectures for 3-D Medical Image Compression", *The 2009 International Conference on Field-Programmable Technology (FPT)*, December 9^{th} - 11^{th} 2009, Sydney, Australia, pp. 472–474.

4. H. Taha, A. Sazish, **A. Ahmad**, M. Sharif, and A. Amira, "Efficient FPGA Implementation of a Wireless Communication System using Bluetooth Connectivity", *The 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 30^{th} - June 2^{nd} 2010, Paris, France, pp. 1767–1770.

5. **A. Ahmad**, A. Amira, Y. Berviller, and H. Rabah, "Rapid Prototyping of Finite Radon Transform (FRAT) for Medical Imaging Applications", *The* 2^{nd} *European Workshop On Visual Information Processing (EUVIP)*, July 5^{th} - 7^{th} 2010, Paris, France.

6. B. Krill, **A. Ahmad**, A. Amira and H. Rabah, "New FPGA-Based Dynamic Partial Reconfiguration Design Flow and Environment For Image Processing Applications", *The* 2^{nd} *European Workshop On Visual Information Processing (EUVIP)*, July 5^{th} - 7^{th} 2010, Paris, France.

7. **A. Ahmad**, B. Krill, A. Amira, and H. Rabah, "Dynamic Partial Reconfigurable 3-D Haar Wavelet Transform IP Cores Design", *The* 2^{nd} *UK-Malaysia Engineering Conference (UK-MEC)*, April 8^{th} - 9^{th} 2010, London, United Kingdom, pp. 25–35. [**Awarded as a best paper**]

8. **A. Ahmad** and A. Amira, "FPGA-based Architectures for 3-D Medical Image Compression", *The* 1^{st} *Malaysia Glasgow Doctoral Colloquium (MGDC)*, January 20^{th} - 21^{st} 2010, Glasgow, Scotland, pp. EA49–EA50.

9. **A. Ahmad**, A. Amira, Y. Berviller and H. Rabah, "FPGA-based Architectures of FRAT for Medical Image Processing", *United Kingdom - Malaysia - Ireland Engineering Science Conference (UMIES)*, June 23^{rd} - 25^{th} 2010, Belfast, Northern Ireland.

Conference Papers – Under Review

1. **A. Ahmad**, A. Amira, Hassan Rabah, and Yves Berviller, "FPGA-based Architectures of Finite Radon Transform for Medical Image De-noising", *The 2010 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS 2010)*, December 6^{th} - 9^{th} 2010, Kuala Lumpur Malaysia.

2. A. Gupta, **A. Ahmad**, and A. Amira,, "Rapid Prototyping of a Wireless Communication System using FPGA", *The 2010 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS 2010)*, December 6^{th} - 9^{th} 2010, Kuala Lumpur Malaysia.

3. P. Nicholl, **A. Ahmad**, and A. Amira,, "Optimal Discrete Wavelet Transform (DWT) Features for Face Recognition", *The 2010 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS 2010)*, December 6^{th} - 9^{th} 2010, Kuala Lumpur Malaysia.

List of Abbreviations

μblaze	Micro blaze
1-D	One-dimensional
2-D	Two-dimensional
3-D	Three-dimensional
AG	Address generator
AGWN	Additive Gaussian white noise
ASIC	Application specific integrated circuit
ALU	Arithmetic logic unit
BLV	Brent. Luk, Van
BPV	Bit per voxel
BRAM	Block random access memory
CABAC	Context-based adaptive binary arithmetic coding
CAVLC	Context-based adaptive variable length coding
CDF	Cohen-Daubechies-Favreau
CIF	Common intermediate format
CORDIC	Coordinate rotation digital computer
CPU	Central processing units
CR	Compression ratio
CSD	Canonical sign digit
CT	Computed tomography
CUDA	Compute unified device architecture
DA	Distributed arithmetic

DCM	Digital clock management
DCT	Discrete cosine transform
DDR-2	Double data rate
DFF	D flip-flop
DFT	Discrete Fourier transform
DHT	Discrete Hartley transform
DMA	Distortion minimisation algorithm
DPR	Dynamic partial reconfiguration
DSP	Digital signal processor
DWT	Discrete wavelet transform
EAPR	Early access partial reconfiguration
EDA	Electronic design automation
ESCOT	Embedded sub-band coding with optimal truncation
ESM	Erlangen slot machine
EVD	Eigen value decomposition
FIR	Finite impulse response
FFT	Fast Fourier transform
FIFO	First in first out
FMRI	Functional magnetic resonance imaging
FPGA	Field programmable gate array
fps	Frames per second
FRAT	Finite Radon transform
FRIT	Finite ridgelet transform
FWT	Fast wavelet transform
GOP	Group of pictures
GPGPU	General-purpose computation on graphics processing units
GPP	General purpose processor
GPU	Graphics processing unit
HBWD	Hierarchical block wavelet decomposition
HD	High-definition

HDMI	High-definition medical imaging
HDTV	High-definition TV
HLL	High-level language
HW	Hardware
HWT	Haar wavelet transform
HVS	Human visual system
I/O	Input/output
IOB	Input/output block
ICAP	Internal configuration access port
ILA	Integrated logic analyzer
IT	Integer transform
IRT	Inverse Radon transform
JPEG	Joint photographic experts group
LC	Logic cell
LUT	Look-up tables
MAV	Median absolute value
MPGA	Mask programmable gate array
MRI	Magnetic resonance imaging
MSE	Mean square error
NCD	Native circuit description
NFS	Networking file system
NMC	Native macro circuit
NSWD	Non-standard wavelet decomposition
NTSC	National television system committee
OT	Objective test
PAL	Programmable arrays logic
PAL	Phase alternate line
PAR	Place and route
PC	Personal computer
PCI	Peripheral component interconnect

PET	Positron emission tomography
PLL	Phase-locked-loop
PR	Partial reconfiguration
PSNR	Peak signal to noise ratio
QCIF	Quarter common intermediate format
RAM	Random access memory
RH	Reconfigurable hardware
ROM	Read only memory
ROI	Regions of interest
RPM	Reconfigurable processing modules
RT	Radon transform
RTL	Register-transfer level
RTR	Run-time reconfiguration
SoPC	Systems on a programmable chip
SPIHT	Set partitioning in hierarchical trees
SRAM	Static RAM
ST	Subjective test
STFT	Short time Fourier transform
SVD	Singular value decomposition
SW	Software
SWD	Standard wavelet decomposition
UCF	User constraint file
UK	United Kingdom
US	Ultrasound
VGA	Video graphic array
VHDL	Very-high-speed integrated circuit hardware description language
VLC	Variable length coding
VLSI	Very large scale integration
XE	Xilinx edition

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Medical imaging as an indispensable part of medical management of diseases appears as one of the most challenges areas and its full potential seems to be boundary-less. Doubtless, that medical imaging applications deal with massive amounts of data and Lee *et al.* [1] disclose an interesting fact on this issue:

"*The University of Washington Medical Centre, a medium-sized hospital with about 400 beds, performs approximately 80,000 studies per year. At 30 Mbytes per study, the amount of digital images generated is 2.4 Tera* (10^{12}) *bytes of data per year or approximately 10 Gbytes per day*".

To further highlight the issues and challenges ahead in these areas, in 2007, there were more than 155,000 cancer deaths in the United Kingdom (UK), and one in four (27%) of all deaths in the UK were due to cancer. Moreover, with more than 200 different types of cancer, empirical data shown in Figure 1.1 exposes 289,000 new cases of cancer diagnosed each year in the UK [2].

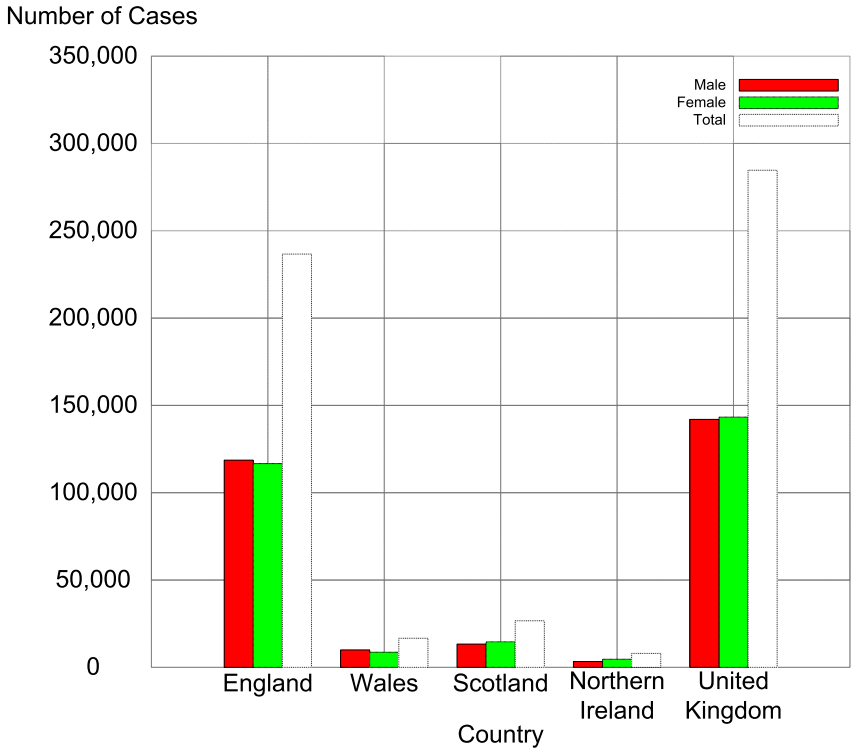

Figure 1.1: Number of new cases of all malignant neoplasms in UK 2007 (Excluding non-melanoma skin cancer) [2].

From medical technology perspective, there are various medical imaging modalities, such as magnetic resonance imaging (MRI), ultrasound (US), computed tomography (CT) and positron emission tomography (PET), which have been widely used for cancer diagnosis. However, MRI in particular offers tremendous potential for facilitating cancer screening and diagnosis, as well as for monitoring treatment, especially for some types of brain and primary bone tumours, soft tissue sarcomas and for tumours affecting the spinal cord [2]. On the other hand, a general shift from two-dimensional (2-D) slices to three-dimensional (3-D) models of organs has been observed [3]. Thus, it contributes for vast challenges in medical data management operations.

As a result of increasing number of people to be diagnosed and of considerable increase in the volume of medical image data generated in hospitals, medical image compression is imperative [4]. Additionally, in numerous medical applications both efficient storage and transmission of data through high-bandwidth digital

communication lines are of crucial. Moreover, it is well known also that noise on medical image resulting in low image quality, and yet limits the diagnostic effectiveness. Therefore, the field of medical imaging introduces a complex problem [5]. In the case of medical image compression for instance, it is mainly involves matrix transforms, repeatedly on a large set of image data, often under real-time requirements. As a result, there is a need for high-performance systems whilst keeping architectures flexible to allow for quick upgradeability. A lot of effort in research and development has been dedicated to computer and processor architectures suitable for such applications [6–10].

Spectrum of possible hardware solution has grown enormously. At one end of the spectrum are processors such as general purpose processors (GPPs) or digital signal processors (DSPs), which have an instruction-set architecture. They provide the possibility of processing arbitrary computations due to their architectural concept. Pursuant to the overhead paid for the flexibility, processors are rather inefficient regarding performance and power consumption [11]. At the other end of the spectrum is application specific integrated circuits (ASICs), which contain dedicated circuits specialised to a particular set of functions. Thus, the architecture is optimally suited for the functions at hand which is the reason of ASICs are efficient regarding performance and power consumption, but they lack flexibility, as no programmable resources are provided [11].

Due to the high demand of graphics processing of the video game industry, graphics processing units (GPUs) have evolved into massively parallel computing engines [12]. Moreover, the introduction of compute unified device architecture (CUDA) by NVIDIA is a significant step to derive more research and development in this area [13]. GPUs have become of choice for many computationally intensive applications as it contains with many processing elements, high-memory bandwidth, and programmability [6]. However, major obstacle of GPUs is concerned with less efficient mapping parallel application in the GPU's pixel processing data paths [12].

On the other hand, reconfigurable hardware (RH) and specifically field programmable gate array (FPGA) is a solution that can offer high-throughput to

numerous data-intensive applications with critical time constraints [11], [13], [14]. There are two basic categories of FPGAs in the market today: static random access memory (SRAM)-based FPGAs and antifuse-based FPGAs [15]. In the first category, Xilinx customers dominate over the half of the entire market at 51%, whilst the strongest competitor is Altera with 34% [16]. For antifuse-based product, Actel, Quicklogic and Cypress offer another available products [15]. To illustrate the advantages offered by SRAM over antifuse-based FPGAs, Table 1.1 briefly summarises the key features.

Table 1.1: Summary of programming technologies [17].

Feature	SRAM	Antifuse
Technology node	State-of-the-art	One or more generation behind
Reprogrammable	Yes	No
Volatility	Yes	No
Good for prototyping	Yes	No
Power consumption	Medium	Low

In this study, Xilinx FPGA devices have been selected to prototype the developed architectures due to the promising results that have been achieved by previous research group members in [18–20], in which some results can be further exploited. In addition, the nature of the implemented algorithms and applications in this research investigation require some flexibility, parallelism and performance in which the three features are offered by reconfigurable hardware using FPGAs.

It is worth mentioning that modern FPGA devices also offer a large number of look-up tables (LUTs), DSP blocks and a hierarchy of different memory sizes, providing high-level of design flexibility. Furthermore, FPGA run-time reconfigurability allows an excellent option for the design to be scalable and adaptive to different types of input data.

The trade-offs of different implementation approaches are shown in Table 1.2, and it can be evaluated using various metrics such as performance, cost, programmability,

power and development time.

Table 1.2: Comparison of different implementation approaches.

Platform	Performance	Cost	Power	Flexibility	Design effort
ASIC	High	High	Low	Low	High
DSP	Medium	Medium	Medium	Medium	Medium
GPP	Low	Low	Medium	High	Low
GPU	High	Medium	High	Medium	Medium
RH	Medium/High*	Medium	High/Low#	High	Medium

Note:
*Depends on technology and available embedded resources
#With Xilinx Spartan's FPGA

1.2 Three-dimensional (3-D) Medical Image Processing

Medical image processing is a niche area concerned with the operations and processes to generate images of a human body for clinical purposes and covering potential areas in medical image processing analysis such as image acquisition, image formation, image enhancement, image compression and storage, and image-based visualisation [21].

In contrast to general image processing analysis that converts an image signal into a physical image, various medical imaging modalities have been shown to be useful for patient diagnosis [5]. An overview of MRI, CT and PET image features is given in Figure 1.2, whilst some examples of MRI, CT and PET images are depicted in Figure 1.3(a) – (c).

To date, modern medical imaging technologies are capable of generating high-resolution 3-D images, and consequently, make medical image analysis tasks at least one-dimension more compute-intensive than standard planar 2-D images [6]. In brief, the higher computational cost appears in medical imaging analysis, introduces

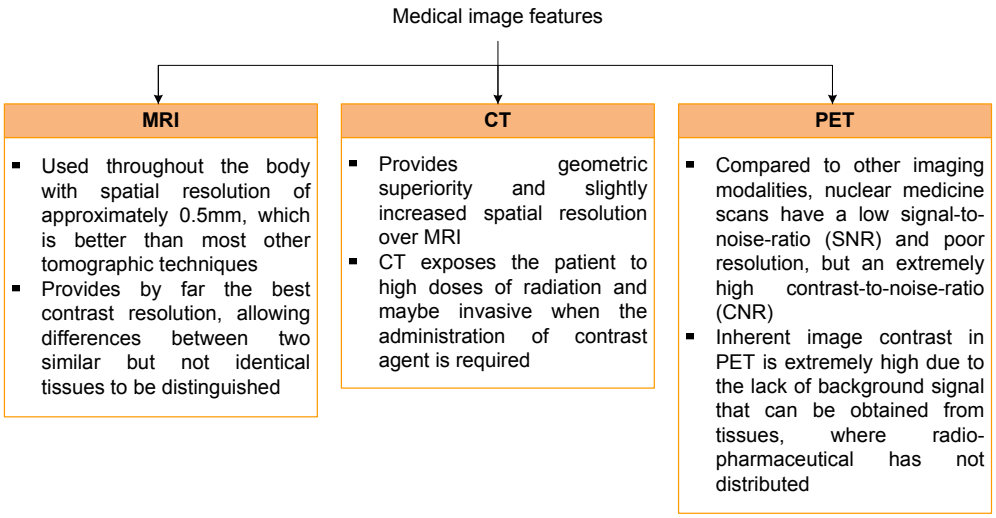

Figure 1.2: Medical image features.

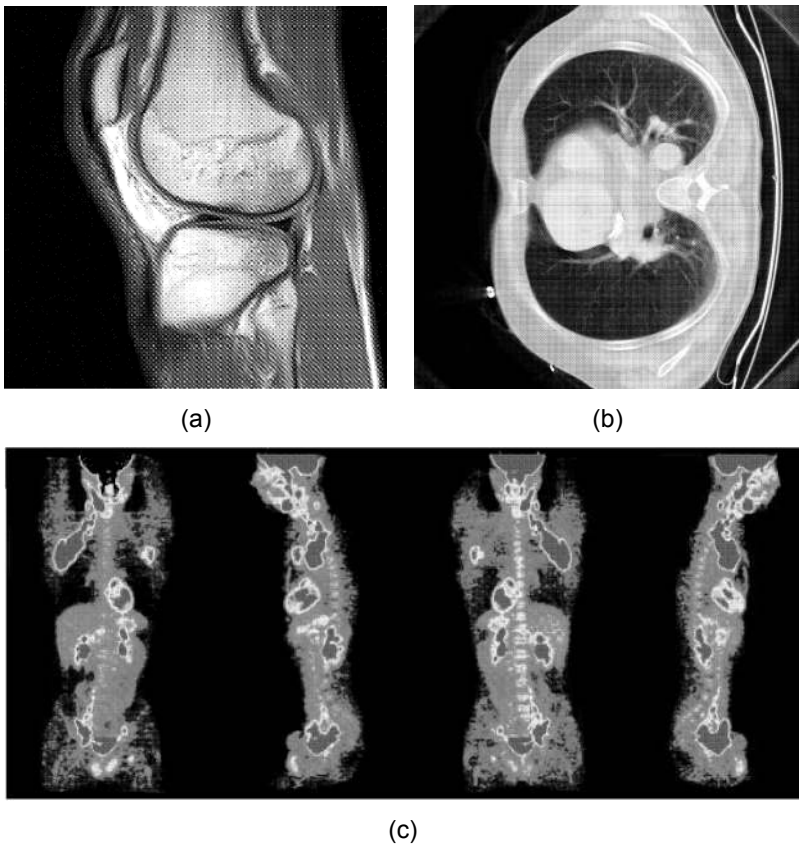

Figure 1.3: Examples of medical images (a) Sagittal MRI knee image (b) Transaxial CT lung slice (c) PET scan for lymphoma [22].

new technologies to be developed in many other areas, including computer graphics, computer vision as well as biomedical signal processing [23].

On top of that, a general shift from 2-D slices to 3-D models of organs has been observed [3]. As a result of this trend, medical imaging procedures are increasingly being used for guiding intervention, controlling therapy and monitoring the cause of illnesses [3]. The uniqueness of 3-D medical images in various modalities including CT, MRI, PET, US, and magnetic resonance angiography (MRA) have been addressed in [24–27], and these features can be simplified and shown in Figure 1.4.

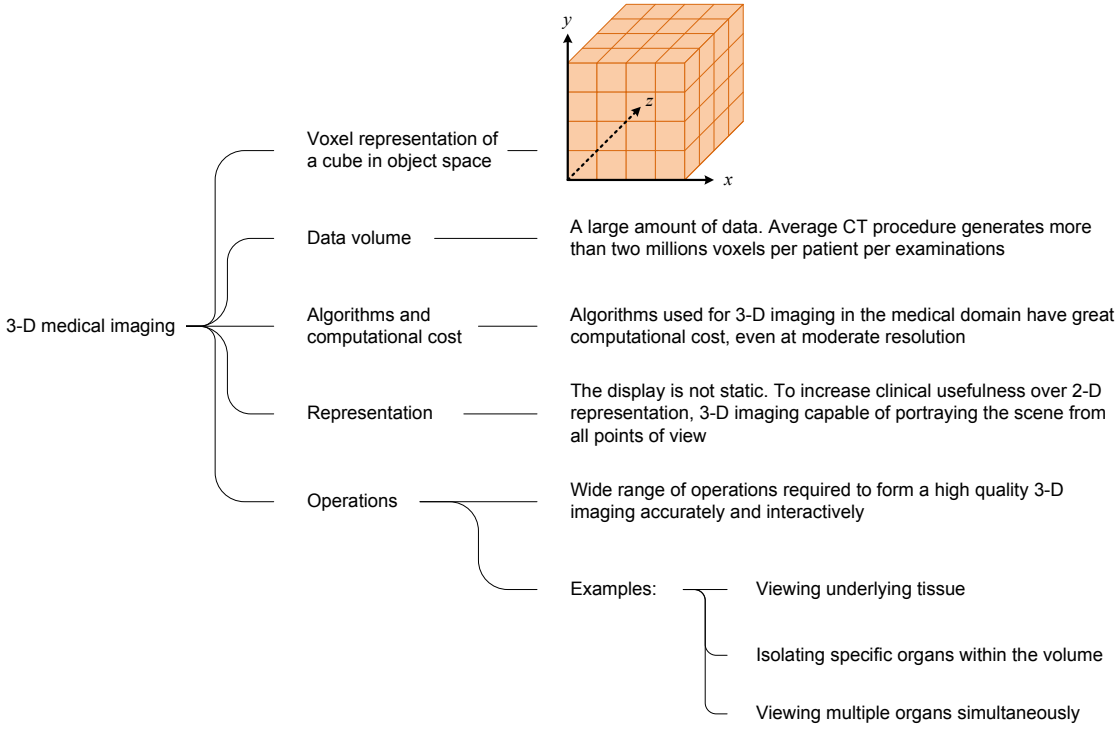

Figure 1.4: 3-D medical image features.

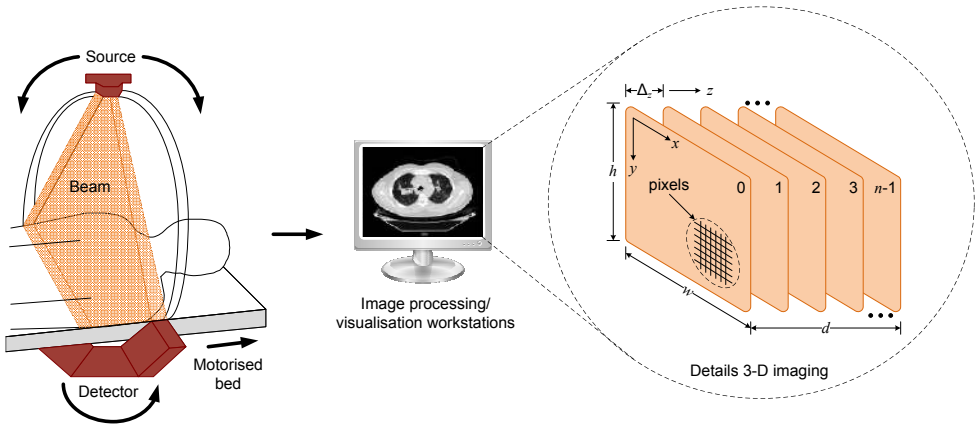

Figure 1.5: 3-D medical image data processing.

In 3-D medical imaging modalities, the data produced usually consists of a number of parallel slices for the body. As illustrated in Figure 1.5, most generated medical volumes acquire one slice at a time, with the patient moved along on a motorised bed between each slice. The resulting data set comprises n-slices and each containing $w \times h$ pixels. The slices are separated by a distance Δ_z pixels, where Δ_z is usually greater than one. The data is therefore, anisotropic, with inferior resolution perpendicular to the slices than within them. The depth d of the data set is $(n-1)\Delta_z$.

To paint a comprehensive picture of the central issues in 3-D medical image processing, several survey papers have been collected and analysed, then illustrated as a time line in Figure 1.6.

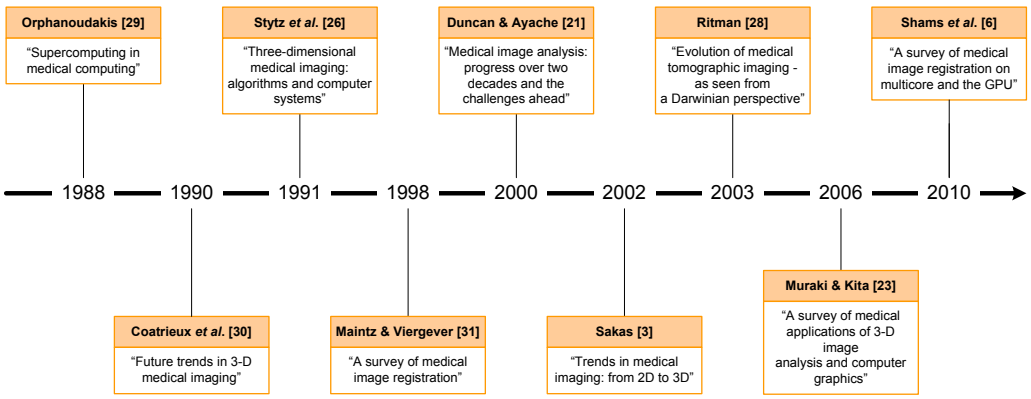

Figure 1.6: Survey on medical image processing.

Consequently, Table 1.3 illustrates the classification of all these works based on the following points:

1. Medical image processing applications – compression, segmentation, registration, enhancement and de-noising, quantification;
2. System implementation – hardware design and development, software simulation or algorithm development and optimisation; and
3. Types of images – 2-D or 3-D.

Table 1.3: Survey on medical image processing.

Refs.	Applications						Image type		Implementations		
	1	2	3	4	5	6	2-D	3-D	HW	SW	General
[3]							✓	✓			✓
[6]		✓	✓					✓	✓		
[21]		✓	✓				✓	✓			✓
[23]		✓	✓					✓		✓	
[26]								✓		✓	
[28]							✓	✓			✓
[29]							✓				✓
[30]		✓				✓		✓			✓
[31]			✓				✓	✓			✓

Note:
HW: Hardware, SW: Software, 1: Compression, 2: Segmentation, 3: Registration
4: Enhancement and de-noising, 5: Quantification, 6: Others

Based on the comprehensive survey that has been carried out in medical image processing trend, the following key conclusions can be made:

1. 3-D medical images demonstrate a significant shift as a result of remarkable advantages offered not only for diagnostic setting, but prominently in the aspects of planning and surgical radiotherapeutical procedures [31];

2. As diverse as the important contribution in segmentation and registration aspects, these applications have dominated most of the reported works [6], [21], [23], [30], [31]; and

3. The advancement for both algorithms development and optimisation as well as hardware implementation aspects lies as a result of intra-disciplinary advancement that involves medical specialities, industrial development, physics, engineering, computer science and mathematics [26], [28].

A close examination of the algorithms used in real-time medical image processing applications reveals that many of the fundamental actions involve matrix or vector operations [5]. Most of these operations are matrix transforms including fast Fourier transform (FFT), discrete wavelet transform (DWT) and some recently developed transforms such as finite Radon, curvelet and ridgelet transforms which are used in 2-D or 3-D medical imaging [32].

Unfortunately, computational complexity for the matrix transform algorithms is in the order from $O(N \times logN)$ for FFT to $O(N^2 \times J)$ for the curvelet transform (where N is the transform size and J is the maximum transform resolution level) are computationally intensive for large size problems. For that reason, efficient implementation for these operations are of interest not only because matrix transforms are important in their own right, but because they automatically lead to efficient solutions to deal with massive medical volumes [19].

As diverse as the spectrum that has been explained, hardware acceleration for medical image processing has attracted much attention in research and development. In the following section, discussions on the potential hardware platforms for consideration in this research study are given.

1.3 High-Performance Solutions for Medical Image Processing Applications

One of the primary methods in conventional computing for the execution of image and signal processing algorithms is the use of GPPs. Processors execute a set of instructions to perform a computation. By changing the software instructions, the functionality of the system is altered without the hardware modification.

However, this flexibility does not contribute for significant overall performance. The processor must read each instruction from memory, decode its meaning and only then execute it. This result in a high execution overhead for each individual operation.

Additionally, the set of instructions that may be used by a program is determined at the fabrication time of the processor. Any other operations that are to be implemented must be built out of existing instructions.

To achieve high-performance, image and signal processing applications implementation have moved away from the traditional approach of general-purpose computing towards systems containing specialist architectural support. A lot of research has been carried out on architectural support including DSPs and special purpose hardware [11]. An overview of possible platforms is given in the following subsections.

1.3.1 Digital Signal Processor (DSP)

One method of increasing the performance of GPP is to attach a specialised processing unit in the form of DSP. As illustrated in Figure 1.7, DSP has features that accelerate its capability for high-performance, repetitive and numerically intensive task applications.

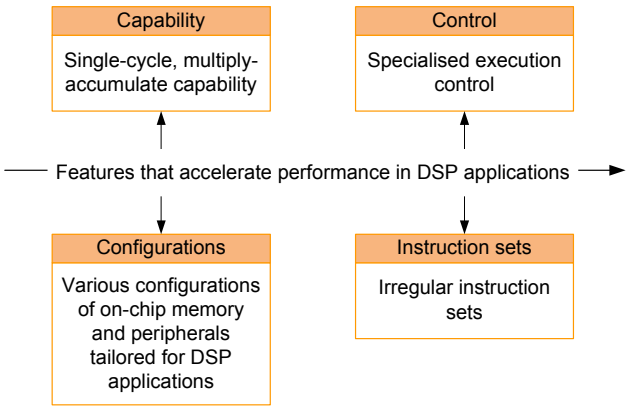

Figure 1.7: DSPs features for performance accelerations.

High performance DSPs often have two multipliers that enable two multiply-accumulate operations per instruction cycle. Moreover, DSPs generally feature multiple-access memory architectures that enable DSPs to complete several accesses to memory in a single instruction cycle. Furthermore, DSPs usually provide a loop instruction that allows tight loops to be repeated without spending any instruction

cycles for updating and testing the loop counter or for jumping back to the top of the loop.

DSPs generally allow several operations to be encoded in a single instruction. For example, a processor that uses 32-bits instructions may encode two additions and multiplications, and four 16-bits data moves into a single instruction. Besides, DSP instruction sets allow a data move to be performed in parallel with an arithmetic operation. GPPs, in contrast, usually specify a single operation per instruction.

It is worth mentioning that the DSPs are also equipped with embedded fused multiply/add which can be used for orthogonal transforms implementations such as discrete cosine transform (DCT), discrete Hartley transform (DHT) as well as others computation-intensive DSP functions like convolution, interpolation and adaptive filtering [33]. As a result, DSPs have been successfully used in a wide range of image processing applications [34–39].

1.3.2 Special Purpose Application Specific Integrated Circuit (ASIC) Hardware

ASICs give better performance for particular applications, and they are designed specifically to perform a specific computation. Owing to this feature, they efficiently perform the given task according to the application's design specification which may be to optimise for one or more of design flexibility, performance, power consumption and area [40–42]. However, after fabrication the circuit is unable to be altered. This forces a redesign and a refabrication of any part of the chip which requires modification. This is an expensive process, especially when one considers the difficulties in replacing ASICs in a large deployed system [11]. The main disadvantages of this approach can be summarised as shown in Figure 1.8.

A new breed of ASIC products, called "structured ASIC", can reduce the expenses by more than 90% for derivative chips, and speed up time-to-market [43]. The underlying concept behind structured ASICs is fairly simple. Although there

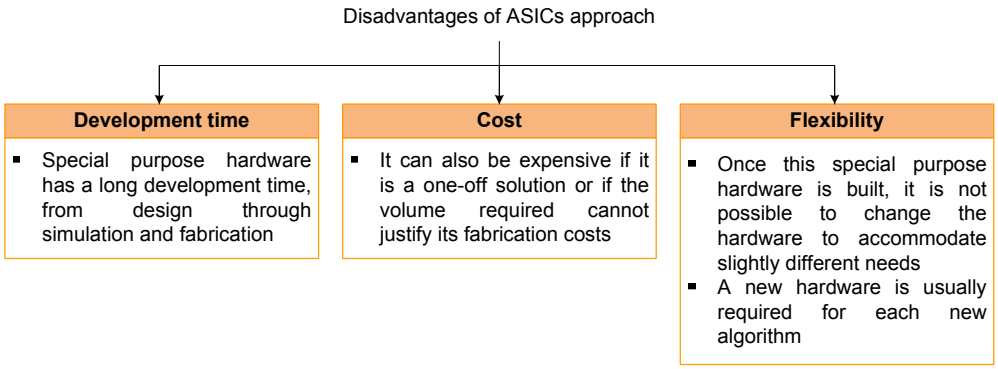

Figure 1.8: Main disadvantages of ASICs.

are a wide variety of alternative architectures, they are all based on a fundamental element called a "tile" by some or a "module" by others. This tile contains a small amount of generic logic implemented either as gates and/or multiplexers and/or a LUT. Depending on the particular architecture, the tile may contain one or more registers and possibly a very small amount of local random access memory (RAM). An array of these tiles is then pre-fabricated across the face of the chip [43], [44].

Structured ASICs also typically contain additional pre-fabricated elements, which may include configurable general-purpose input/output (I/O), microprocessor cores, gigabit transceivers and embedded block RAM. When compared with standard cell-based ASICs, structured ASICs offer shorter turnaround time, and require less cost for future functional changes. Structured ASIC technology is especially suitable for platform ASIC designs that have integrated most of the intellectual property (IP) blocks and leave some space for custom changes [45].

1.3.3 Graphical Processing Unit (GPU)

In these days, GPU computing has gained significant momentum and has evolved into an established research area. Hardware vendors have recognised the benefits of GPU computing and have provided high-level programming environments to express parallelism more efficiently [46]. In comparison with central processing units (CPUs) as shown in Figure 1.9(a) and (b), the GPUs architecture is to dedicate as much silicon

area as possible to arithmetic logic units (ALUs). By eliminating all the scheduling logic and caches, GPUs can exploit instruction-level parallelism, and hence reduce memory latency in CPUs [47].

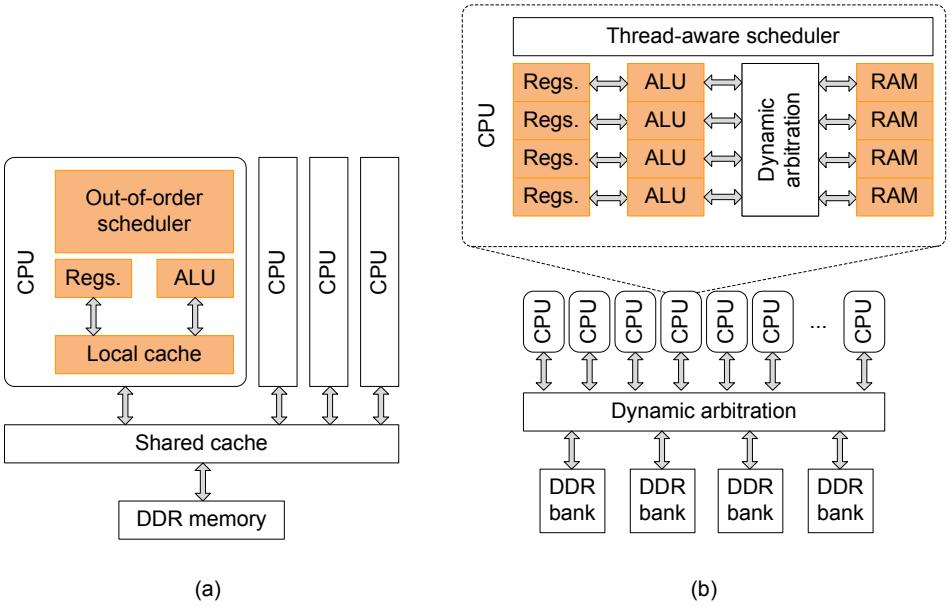

Figure 1.9: Architecture comparison (a) CPU (b) GPU [47].

The popular association of GPUs is with accelerating graphics, but the new architectures from manufactures such as NVIDIA corporation and ATI are capable of performing general-purpose computing. There are two approaches [13] for general-purpose computing using GPU: to pose the problem as a graphic problem and solve it using a graphic language such as OpenGL or DirectX GPU programming, or to program the GPU directly.

Even GPUs as commodity computer graphics chips are probably todays most powerful computational hardware with cost, the main limitations and difficulties [48] of this platform can be simplified as follows:

1. Applications:

   The increasing flexibility of GPUs, coupled with some ingenious uses of that flexibility by general-purpose computation on graphics processing units (GPGPU) developers, has enabled many applications outside the original narrow

tasks for which GPUs were originally designed, but many applications still exist for which GPUs are not well suited;

2. Computing constructs:

   The lack of integers and associated operations such as bit-shifts and bit-wise logical operations (`AND`, `OR`, `XOR`, `NOT`) makes GPUs unsuitable for many computationally intense tasks. Moreover, the lack of double precision prevents GPUs from being applicable to many very large-scale computational science problems; and

3. Non-graphics tasks:

   The GPU uses an unusual programming model, so effective programming is not simply a matter of learning a new language. Indeed, the computation must be recasting into graphics terms by a programmer familiar with the design, limitations, and evolution of the underlying hardware.

1.3.4 Reconfigurable Hardware (RH): A Review of Field Programmable Gate Array (FPGA)

The recent advances in RH are for the most part derived from the technologies developed for FPGAs in the mid 1980s [13]. FPGAs were originally created to serve as a hybrid device between programmable arrays logics (PALs) and mask programmable gate arrays (MPGAs). Like PALs, FPGAs are fully electrically programmable, meaning that the physical design costs are amortised over multiple application circuit implementations, and the hardware can be customised nearly instantaneously. Like MPGAs, they can implement very complex computations on a single chip, since it consists of an array of pre-fabricated transistors that can be customised during chip fabrication [15]. MPGAs allow for user's customisation by connecting the transistors with custom wires.

Because of these features, FPGAs have been viewed primarily as glue logic replacement and a rapid prototyping vehicle. However, the flexibility, capacity and performance of these devices have opened up completely new avenues in high-performance computation, forming the basis of reconfigurable computing [11], [49].

The early FPGA devices from Xilinx, Altera and others provided relatively little logic, but later generations provided enough logic for researchers to consider FPGAs for direct implementation of computational algorithms in reconfigurable logic devices. The densities of todays FPGAs have exceeded 150,000 6-input LUTs per device and some have developed into devices that can be used to build complete systems on a programmable chip (SoPC), providing such specialised features as DSP blocks, multi-gigabit serial I/O, embedded microprocessors and embedded static RAM (SRAM) blocks of various sizes.

Field Programmable Gate Array (FPGA) Structure

The basic architecture of FPGAs consists of three components: logic blocks, routing and I/O blocks. Generally, FPGAs consist of an array of programmable logic blocks that can be interconnected to each other as well as to the programmable I/O blocks through some sort of programmable routing architecture. To be more specific, Figure 1.10 provides an overview diagram of Xilinx's FPGA architecture.

A Basic Logic Block

As shown in Figure 1.10, a typical FPGA has a logic block with one or more 4-input LUT, optional D flip-flop (DFF) and some form of fast carry logic. The LUTs allow any function to be implemented, providing generic logic. The DFF can be used for pipelining, registers, state holding functions for finite state machines, or any other situation where clocking is required. The fast carry logic is a special resource provided in the cell to speed up carry-based computations, such as addition, parity, wide logical `AND` operations and other functions.

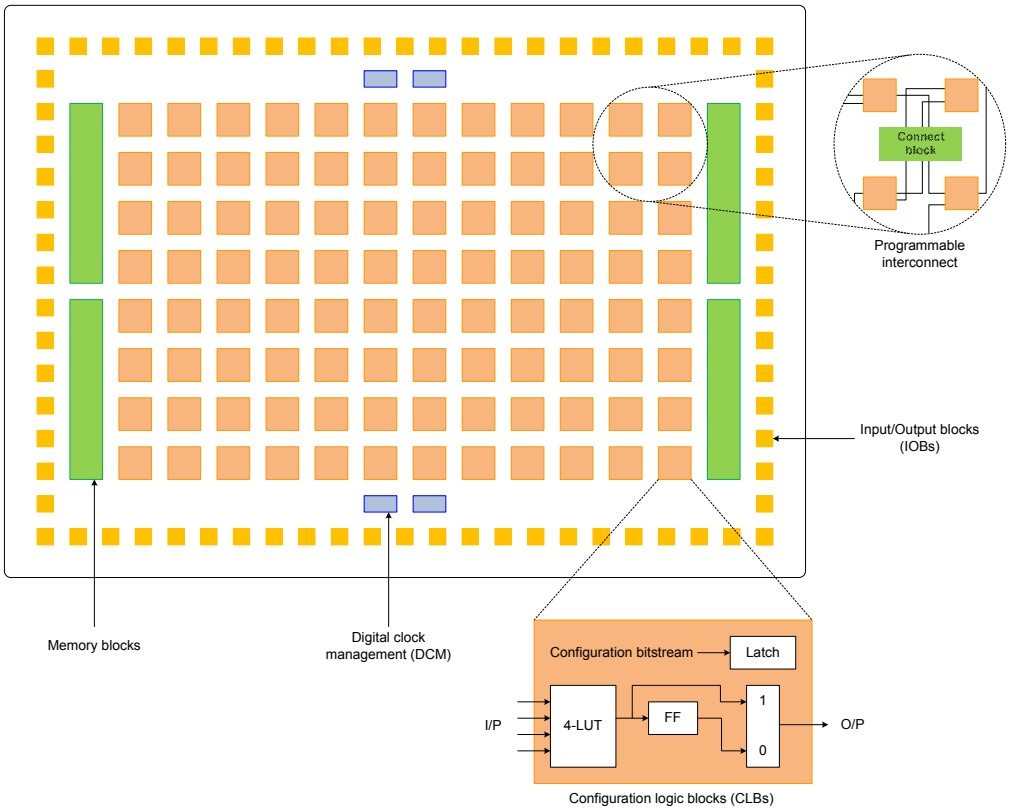

Figure 1.10: Xilinx's FPGA structure with internal blocks.

Routing

Most FPGA architectures organise their routing structures as a relatively smooth sea of routing resources, allowing fast and efficient communication along the rows and columns of logic blocks [49]. The logic blocks are embedded in a general routing structure, with input and output signals attaching to the routing fabric through connection blocks as shown in Figure 1.10.

Connection Blocks

The connection blocks provide programmable multiplexers, selecting which of the signals in the given routing channel will be connected to the logic block's terminals. These blocks also connect shorter local wires to longer distance routing resources. Signals flow from the logic block into the connection block and then along longer wires within the routing channels [49].

Switch Boxes

At the switch boxes, there are connections between the horizontal and vertical routing resources to allow signals to change their routing direction. Once the signal has traversed through routing resources and intervening switch boxes, it arrives at the destination logic block through one of its local connection blocks.

In this manner, relatively arbitrary interconnections can be achieved between the logic blocks in the system. Whilst the routing architecture of an FPGA is typically quite complex, the connection blocks and switch boxes surrounding a single logic block typically have thousands of programming points. They are designed to be able to support fairly arbitrary interconnection patterns [49]. A detailed descriptions of the FPGA devices that have been used in this research are presented in **Appendix A**.

1.4 Design and Implementation Strategies

In this research study, three design and implementation strategies have been used as illustrated in Figure 1.11. The design flows for these strategies are presented in Figure 1.12.

In Chapter 3, very-high-speed integrated circuit hardware description language (VHDL) and partial reconfiguration tools have been used to implement 3-D Haar wavelet transform (HWT). Four main stages involved: design entry, synthesis, implementation and programming. In case of partial reconfiguration, design partitioning, floor planning and budgeting are the main processes involved.

To deal with medical image de-noising as well as to evaluate the performance of finite Radon transform (FRAT), Xilinx AccelDSP tool has been utilised in Chapter 4. The design and implementation begin with an examination of floating point model followed with fixed point and register-transfer level (RTL) generation as well as synthesise and implementation processes.

Finally, VHDL has been fully used again to execute the design and implementation of 3-D compression system in Chapter 5. A detailed explanation for each tool used in this study are presented in **Appendix B**, **C** and **D**.

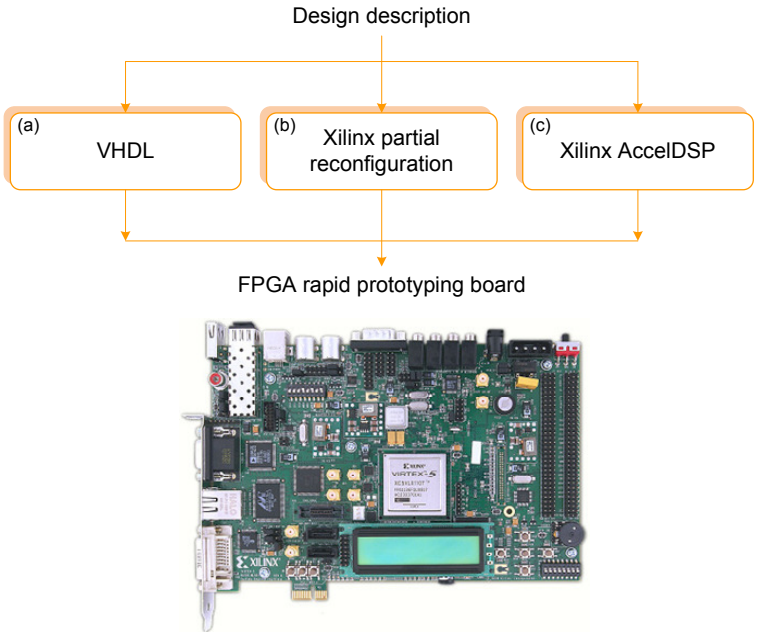

Figure 1.11: Generic design and implementation strategies.

1.5 Motivation and Research Objectives

FPGAs is an extremely powerful tool for several reasons. First and foremost, it allows for truly parallel computations to take place in a circuit. Many modern GPPs and operating systems can emulate parallelism by switching tasks very rapidly. Having operations occur in a parallel fashion results in a much faster overall processing time. This is the case even though the clock speed of the FPGA is lower than the GPPs.

With the availability of advances embedded resources on recent FPGAs devices such as soft cores, dedicated logic and block multipliers, FPGAs are being increasingly deployed in computationally intensive application areas. Moreover, prototyping is also a compelling reason to use FPGAs in the initial design phase. The description of a system can be written and actual hardware can be created to test, instead of simply

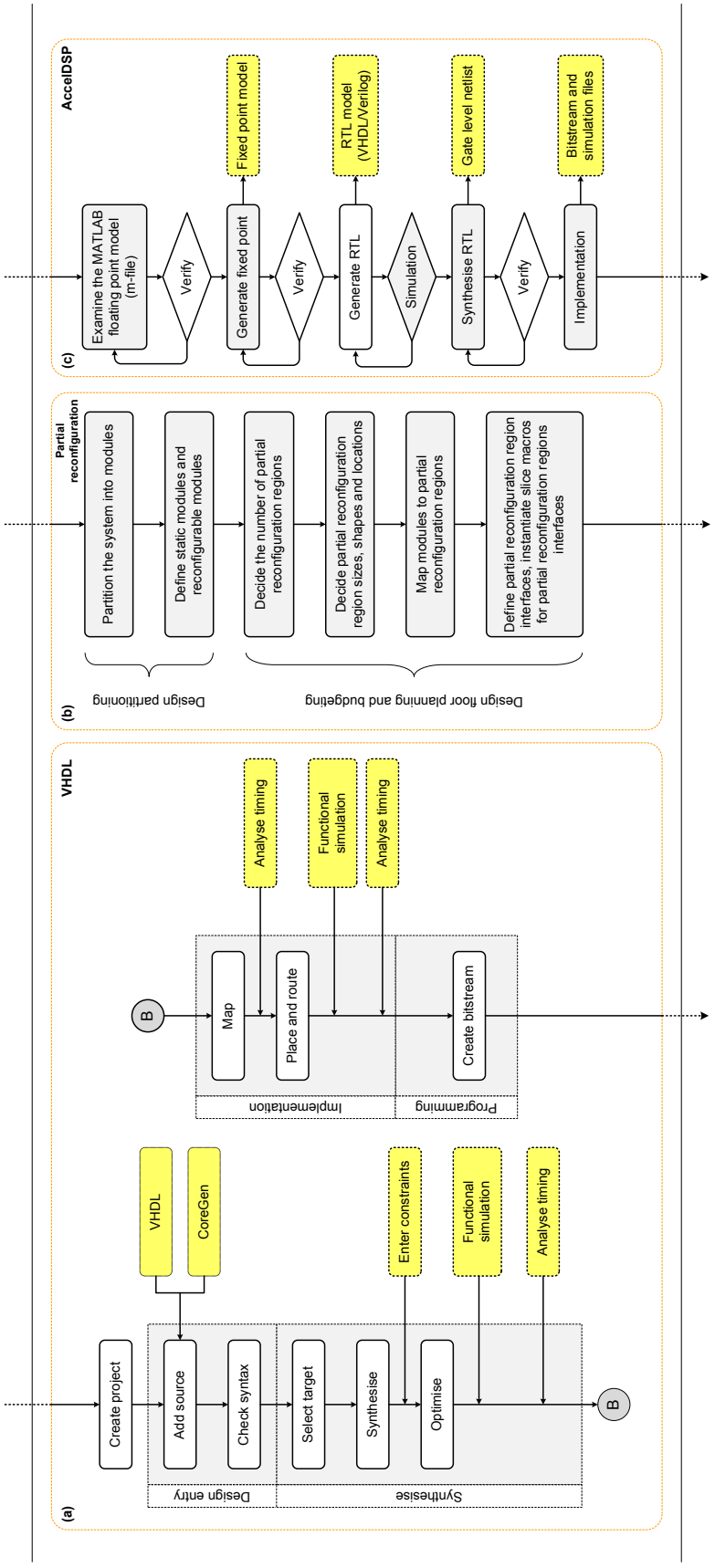

Figure 1.12: Overall design flow.

relying on simulators inside of design. Moreover, the design flexibility available on FPGAs also allows a design to be thoroughly tested and debugged before an ASIC is created, saving on production costs.

FPGAs are everywhere. Companies use them on development boards to help refine new chip designs. Students use them in the laboratory to run experiments. Companies and universities are using them in cutting-edge research on topics ranging from programming technology to real-time systems. The parts themselves are getting so inexpensive that some companies do not even fabricate an ASIC, they simply include the FPGA in their final product.

With the emergence of such reconfigurable hardware, it is not surprising that there has been a considerable amount of research into the use of FPGAs to increase the performance of a wide range of computationally intensive applications. One such application that could greatly benefit from the advantages offered by FPGAs is medical image processing. The regular nature of the complex computations performed repeatedly within medical image processing operations are well suited to a hardware-based implementation using FPGAs.

The application of 3-D medical image processing such as compression and de-noising uses several building blocks for its computationally intensive algorithms to perform matrix transformation operations. Moreover, complexity in addressing and accessing large medical volumes data to be processed have resulted in vast challenges from a hardware implementation point of view.

In order to cope with these issues, FPGAs with efficient reconfigurability techniques should be employed to meet the requirements of these applications in terms of speed, size (area), power consumption and throughput. Dynamic partial reconfiguration (DPR) is a promising technique for reducing the hardware required for implementing an efficient design for 3-D medical image processing application as well as improving the performance of the system. With this technique, the design can be divided into sub-designs that fit into the available hardware resources and can be

uploaded into the reconfigurable hardware when needed [50].

The general goal of this research is concerned with the design and implementation of efficient reconfigurable architectures for 3-D medical image processing, with more emphasis on compression systems and image de-noising. Based on the potential significant contributions in this area, the main objectives of the work presented in this research can be broadly summarised as follows:

1. To design and implement efficiently 3-D HWT architecture using DPR – efficiently can be used as a transform block in the proposed compression system;
2. To design and implement efficiently the finite Radon transform (FRAT) – to be applied for medical image de-noising in pre-processing stage; and
3. To design and implement the 3-D medical image compression system using context-based adaptive variable length coding (CAVLC) – to experimentally demonstrate the whole compression system functionality.

1.6 Overall Contribution

To support the research objectives that have been listed in Section 1.5, Figure 1.13 shows the overall research strategies with potential contributions to be achieved in this research. For the 3-D compression system, analysis of the transform block as well as utilisation of CAVLC are expecting to generate promising outcomes. In terms of transform block, an examination of different transform filters is anticipated to demonstrate a significant contribution. Moreover, by implementing DPR technique, better performance in terms of area, power consumption and maximum frequency is predicted. Furthermore, an evaluation of the FRAT's capability to deal with image de-noising is presumed to exhibit another noteworthy analysis and discussion.

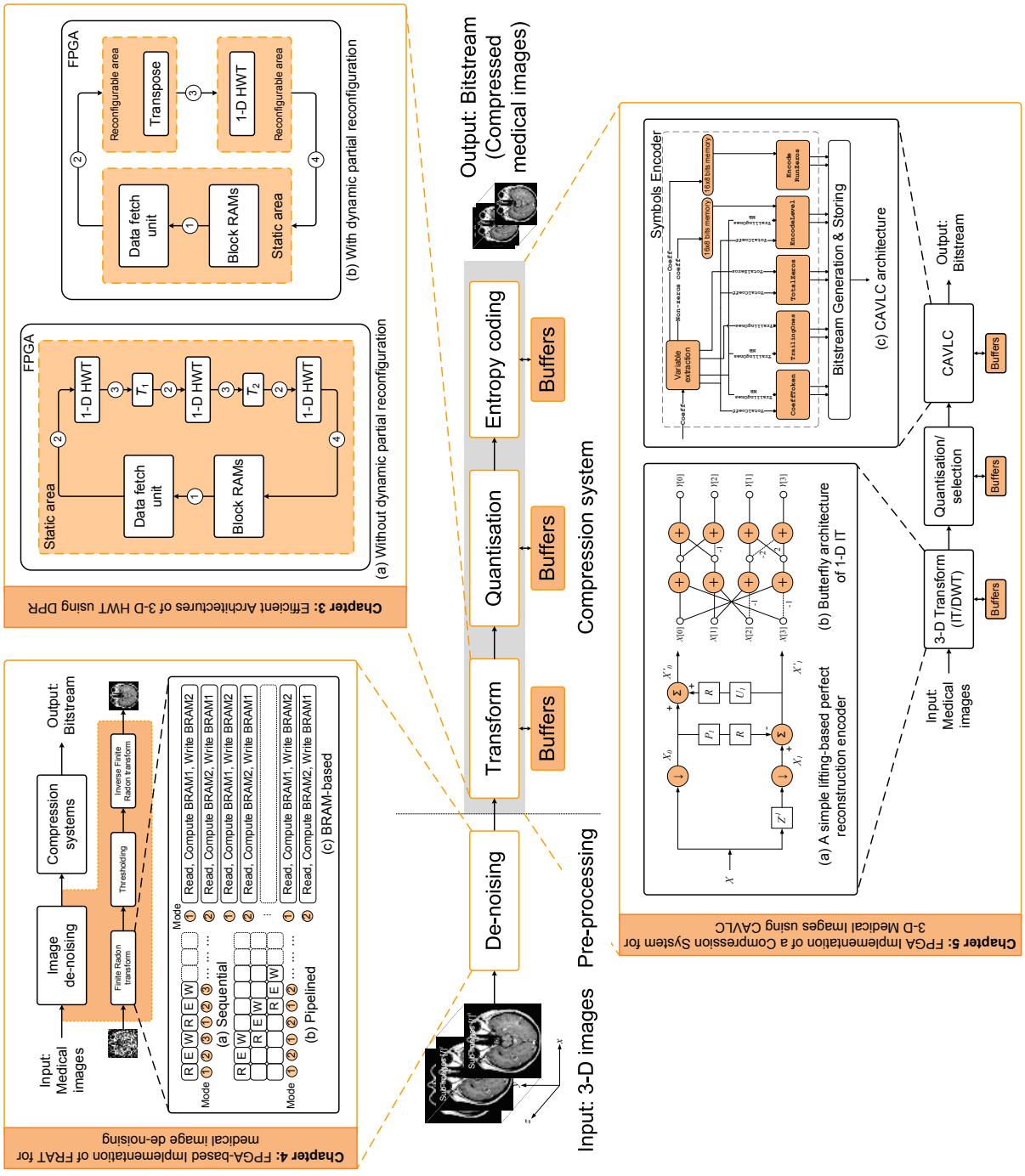

Figure 1.13: Overall research approaches and contributions.

1.7 Thesis Organisation

The structure of the remaining thesis is as follows. Chapter 2 takes a closer look at the most recent architectures and systems for 3-D medical image compression, reconfigurable architectures for DWT, FRAT, CAVLC as well as the DPR method.

Design and implementation of an efficient pipelined 3-D HWT architecture using DPR are presented in Chapter 3. A comparative study for the impact of transform sizes of architectures performance is also addressed.

In Chapter 4, medical image de-noising using the FRAT is given. Three design strategies and analysis of FRAT's performance for noise reduction in medical images is also discussed.

To give a complete overview of this research study, Chapter 5 describes the implementation of 3-D medical image compression system using CAVLC. In this chapter, an evaluation of 3-D integer transform (IT) and DWT have been carried out and discussion on the CAVLC architecture is also reported.

In Chapter 6, concluding remarks and possible refinement of the current research are highlighted. Finally, possible future research directions in the field of design and implementation of 3-D medical image compression systems is presented.

Chapter 2

Related Work

2.1 Overview

In this research study, medical image compression and reconfigurable architectures are two major concerns. The ultimate aim of this chapter is to provide a comprehensive summary of related work on efficient reconfigurable architectures for three-dimensional (3-D) medical image compression. Figure 2.1 illustrates the structure of related research issues presented in this chapter. In brief, the related work covered in this chapter has been reviewed based on three main issues as follows:

1. Medical image compression:

   As one of the rapidly growth areas in these days, medical image processing has received impressive attention, and emerges as an interesting domain of research. In particular, deeper exploration on medical image compression has been carried out as a result of its significant contributions to overcome issues of massive medical data generated and limited storage and bandwidth availability. In Section 2.2, an ample examination of 3-D medical image compression is described, for hardware implementation and algorithms development or optimisation;

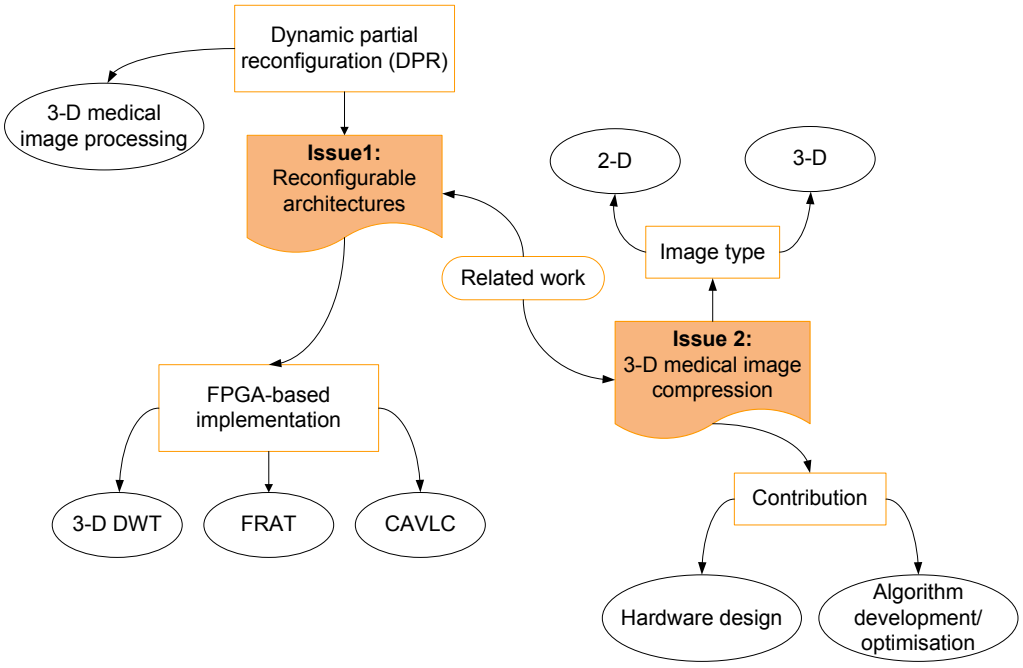

Figure 2.1: Structure of related research issues.

2. Reconfigurable architectures:

   Most of the effort towards the design and implementation in the form of field programmable gate array (FPGA) architectures of discrete wavelet transform (DWT) and finite Radon transform (FRAT) are explained in Section 2.3. Moreover, further discussion of the hardware implementation for image denoising is also given. Furthermore, to accomplish a compression system implementation, a critical analysis of FPGA-based architectures of context-based adaptive variable length coding (CAVLC) is also reported; and

3. Dynamic partial reconfiguration (DPR):

   The applications of medical image compression require several blocks for its computationally intensive algorithms. Dynamic partial reconfiguration (DPR) appears as a promising solution for reducing the hardware used, likewise, improving the performance. To justify the advantages of DPR, related discussions are also covered in Section 2.4.

The rest of the chapter is organised as follows. Section 2.2 gives an overview of the medical image compression, especially for 3-D modalities. Section 2.3 compiles the related work for 3-D DWT, FRAT as well as CAVLC. An explanation for DPR is given in Section 2.4. Discussion on limitation of existing work and research opportunities are explained in Section 2.5. Finally, a brief summary is given in Section 2.6.

2.2 Medical Image Compression

Compression has three steps as shown in Figure 2.2. The first step is transform, which represents the data in a different form, with no information lost. The second step, quantisation, which maps data values to a finite set, where some information is lost. The third step encodes the data in a more compact way.

Figure 2.2: Compression system.

In general, compression for non-medical image requires less critical performance measures compared with medical images [51]. As an example, non-medical image compression is normally based on compression ratio efficiency as well as taking the advantage of the human visual system (HVS) model to produce desired effects. However, for medical image compression specific performance measures are required such as: algorithm complexity, lossless and lossy compression and reconstructed image analysis [5]. Although the objective test plays a significant role, subjective test from a radiologist is also needed [52]. This is important to ensure the generated medical images not only contributes for better compression system, but also retains the medical interest of the images.

A close examination of the existing 3-D medical image compression system [4] reveals a huge gap, particularly for the hardware implementation, since most of the existing works contribute to algorithms development and optimisation [53–59]. In the following, an overview of these works is described, and the first two descriptions [7], [8]

will address the contributions on the hardware implementation of 3-D medical image compression, whilst the others focus on software simulation or algorithms development and optimisation [53–59].

In [7], an implementation of 3-D medical image compression using wavelet transform with parallel computing is presented. A medical image compression system including the 3-D wavelet transformation, scalar quantisation and entropy coding is developed, and yields a good reconstruction quality at a high compression ratio. In this work, a parallel 3-D compression algorithm that uses multiple workstations on a network to speed up the process as shown in Figure 2.3 is used.

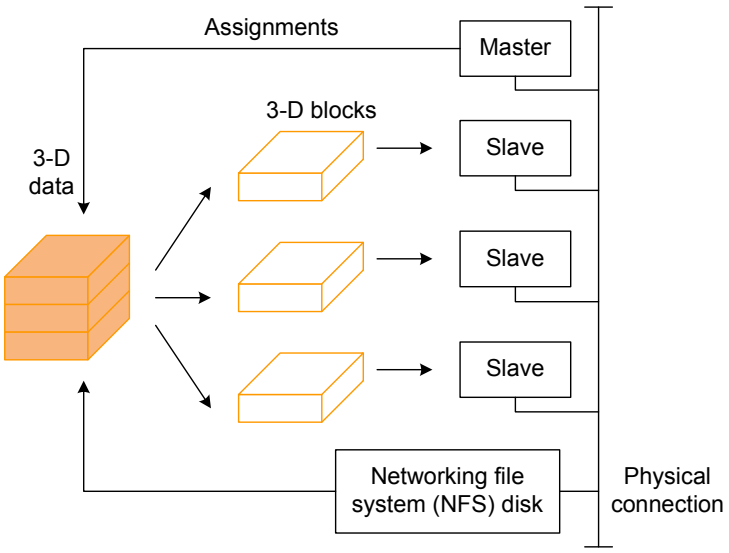

Figure 2.3: Implementation based on parallel computing [7].

In this implementation, the data is assigned using pre-determined-based scheduling and one computer in the network is designated as the master server to distribute the work load, managing the job arrangement as well as combining the results. In particular, master computer evenly divides 3-D image data into a number of blocks according to the number of slaves on the network and assigns jobs to the slaves. The slaves work to receive commands, executing jobs and sending the results back to the master. Furthermore, a networking file system (NFS) disk mounted on the board is used for common storage. To evaluate the compression performance, magnetic resonance imaging (MRI) data with dimensions of $256 \times 256 \times 124$ and a pixel

depth of 12-bits is used, whilst the peak signal to noise ratio (PSNR) and compression ratio (CR) are deployed for objective assessment. Results shown that the 3-D wavelet compression achieves good results with 40% to 90% higher than the two-dimensional (2-D) wavelet compression and 80% to 90% efficiency for the parallel implementation. After all, parallel computing clearly draw a significant idea to increase the compression and decompression speed, as well as maintaining good achievements.

An implementation of the image compression technique of set partitioning in hierarchical trees (SPIHT) on programmable hardware is presented in [8]. The proposed design is implemented on Xilinx Virtex-4 (XC4VLX25) device and to illustrate the area consumed, results obtained are listed in Table 2.1. To exploit the correlation among the image pixels, lifting-based DWT architectures are employed. In addition, an analysis on the storage elements required for the wavelet coefficients is also discussed. Instead of the hardware implementation, this work also proposed a modified SPIHT for encoding the wavelet coefficients. The modifications include the simplification of coefficient scanning process, optimisation of a one-dimensional (1-D) addressing method, and a fixed memory allocation. The proposed algorithm is validated with both the 2-D Lena image and a 3-D MRI data set and results have demonstrated a convincing compression performance with a high PSNR. However, no particular innovation is reported for the hardware implementation.

Table 2.1: Device utilisation [8].

Parameters	**Utilisation**	**%**
Number of slices	7,021 out of 10,752	65
Number of slice flip-flops	1,439 out of 21,504	6
Number of 4-input LUTs	13,356 out of 21,504	62
Number of bonded IOBs	79 out of 242	32
Number of GCLKs	1 out of 32	3

Another issue on the compression methodology is presented in [53]. 3-D medical image compression method for computed tomography (CT) and MRI that uses a separable non-uniform 3-D wavelet transform is proposed. A separable 3-D wavelet

transform with two wavelet filter banks are applied to an image set. The same filter bank is used in each slice for all image sets, but the wavelet filter bank used in slice direction is selected for each image set at various slice distances according to the best compression results. The 9/7 tap filter bank, which is recognised as one of the best filters for the purpose of image compression is used for the x and y directions. On the other hand, the 9/7, Daubechies 4 and the Haar were selected for the $z-$direction. To select an optimum wavelet filter for image sets at various slice thicknesses, different wavelet kernels for different directions are used. Although there is no hardware implementation discussed, interestingly, this work provides significant information towards the influence of various filter banks and slice distances on the compression performance.

An original 3-D coding scheme based on 3-D wavelet transform associated with 3-D lattice vector and uniform scalar quantisation is presented in [54] by Cattin *et al.* This work is mainly on the algorithm development and it is fully adaptive to different image modalities. This is due to the reason of a distortion minimisation algorithm (DMA) utilisation that is capable to select the best set of quantisers. 3-D mini-pig left ventricle angiography images are deployed for system evaluation using PSNR and bit rate. Moreover, subjective test is also conducted with two experienced senior radiologist to evaluate the quality of image for diagnosis purposes. In brief, both tests provide a significant results that urge for future evaluation using large databases and involving more radiologist for the subjective test.

An analysis of lossy-to-lossless compression medical volumetric data using 3-D integer wavelet transforms is discussed by Xiong *et al.* in [60]. In the proposed system, the front end adopts a memory-constrained integer wavelet transform implementation. In addition, two entropy coding techniques: 3-D SPIHT and 3-D embedded sub-band coding with optimal truncation (ESCOT) are applied and modified to suit with volumetric medical data. Results obtained demonstrate the following findings: better lossy compression performance, the memory-constrained integer wavelet transform implementation eliminates PSNR drops at a group of picture (GOP) boundaries, and

the 3-D ESCOT entropy coder achieves the best for lossy and lossless compression of medical volumetric data. In a nutshell, the proposed coding system for lossy-to-lossless compression is significant for transform block optimisation, hence contributes for better outcomes of medical volumetric data analysis.

A progressive transmission lossless compression method for 3-D medical image sets is reported in [55]. An automated filter-and-threshold based pre-processing technique is used to remove noise outside the diagnostic region. To identify the reference image for the entire 3-D medical image set, vector-based approach wavelet decomposition is applied. Moreover, run-length and arithmetic coding are further used to remove coding redundancy. Even the test data only for MRI images, the proposed method is extensible to any class of medical imaging modalities. The main problem of the work presented is on the transform block, specifically the advantages of integer wavelet transformations in the compression system. Experimentally, the compression results of using the proposed predicted wavelet compression method with and without progressive transmission achieve better compression ratio than the predicted wavelet compression method.

To extend the discussion on the transform block contribution, an approach for medical image compression using 3-D discrete Hartley transform (DHT) is discussed by Sunder *et al.* [56]. Two medical modalities including the MRI and X-ray angiogram are used as a test data. To evaluate the compression performance, the objective test is carried out. The performance of this transform is compared with 3-D discrete cosine transform (DCT) and 3-D discrete Fourier transform (DFT). Experimental results exhibit the 3-D DHT yields better compression efficiency in terms of PSNR and bit rate compared with the other two transforms for MRI images at higher bit rates, whereas at lower bit rates the 3-D DCT performs better. For the X-ray angiogram, the 3-D DCT is found to be superior to the other two transforms.

A novel approach using sequential 3-D DCT in medical image compression is presented in [57]. The basic idea of this work is to de-correlate similar pixel blocks through 3-D DCT transformation. A number of adjacent pixel blocks are grouped

together to form a 3-D data cube, and it then quantised and Huffman encoded. The proposed sequential 3-D DCT is benchmarked against the baseline joint photographic experts group (JPEG) compression algorithm, and it is proven with a promising better performance.

In [58], a new compression scheme based on 3-D fast wavelet transform (FWT) is presented and evaluated by Bernabe *et al.* To exploit the scarce presence of movement as well as the spatial and temporal redundancies, the wavelet transform is performed first in the time dimension and later in the space dimensions. The best trade-off between quality and compression ratio based on different wavelet mother functions, the percentile policy and the discarding of the less significant bits for thresholding are also addressed. Moreover, the implementation of the proposed quantiser and an entropy encoder based on a binary run-length code and a Huffman code reveal the compression ratio improvement without affecting the video quality. Therefore, it leads to a good compromise between compression ratio and quality of the reconstructed video.

In [59], a new lossy coding scheme based on 3-D wavelet transform and lattice vector quantisation for volumetric medical images is addressed by Gaudeau and Moureaux. A new code book enclosing a multi-dimensional dead zone during the quantisation step, which enables for better correlation between neighbour voxels is also explained. Additionally, an efficient rate-distortion model to simplify the bit allocation is presented. To evaluate the system feasibility, CT and MRI volumetric medical image data are used. Results gained exhibit a better overall quality of the reconstructed images.

Table 2.2 presents all the existing systems discussed, and they are focusing on software simulation or algorithms development and optimisation. Therefore, it justifies the significant work to be carried out on the hardware development of 3-D medical image compression.

Table 2.2: Summary of 3-D medical image compression systems.

Refs.	Data	Compression		Implementations		Evaluations/Tests		
		Lossless	**Lossy**	**Hardware**	**Software**	**Objective**	**Subjective**	**Others**
[7]	MRI		✓	✓	✓	✓		
[8]	2-D Lena and MRI		✓	✓	✓	✓		
[53]	CT and MRI		✓		✓	✓		
[54]	Angiography		✓		✓	✓	✓	
[55]	MRI	✓			✓			✓
[56]	MRI and angiogram		✓		✓	✓		
[57]	Medical images		✓		✓	✓		
[58]	Medical videos		✓		✓	✓		
[59]	CT and MRI		✓		✓	✓		
[60]	CT and MRI	✓	✓		✓	✓		

2.3 Reconfigurable Architectures

FPGA-based architectures for 3-D DWT, FRAT and CAVLC are discussed in the following subsections.

2.3.1 FPGA-based Architectures for 3-D Discrete Wavelet Transform (DWT)

The 3-D DWT must be separable, which means that the 3-D transform is accomplished by a 1-D DWT in each dimension. The 3-D DWT is the application of a 1-D DWT in three directions [7], [61]. As shown in Figure 2.4, first, the process transforms the data in the x-direction. Next, the low and high pass outputs both feed to other filter pairs, which transform the data in the y-direction.

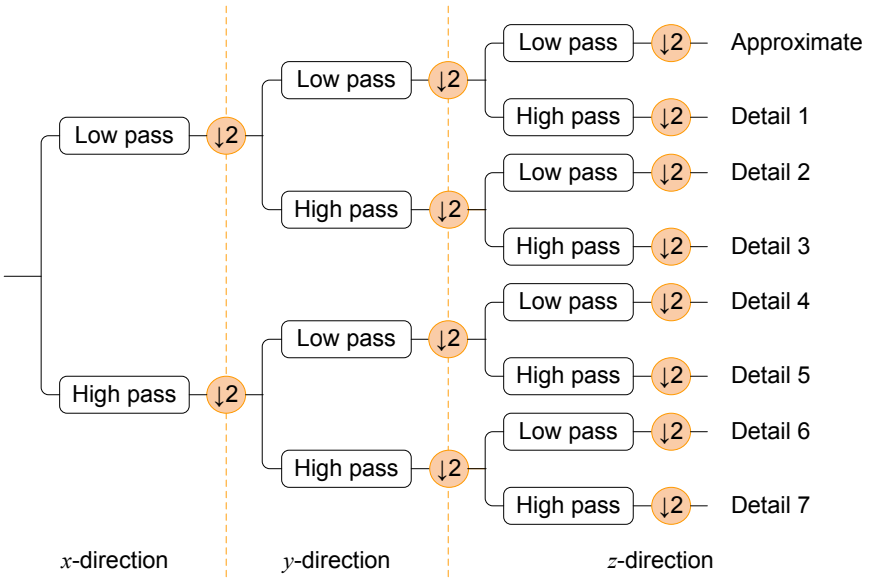

Figure 2.4: The 3-D DWT process.

These four output streams go to four more filter pairs, performing the final transform in the z-direction. The process results in eight data streams. The approximate signal, resulting from scaling operations only, goes to the next octave of the 3-D transform. It has roughly 90% of the total energy [7]. Meanwhile, the seven other streams contain the detail signals.

According to Lee *et al.* [1], the 3-D DCT is more efficient than the 2-D DCT for X-ray CT. Likewise, one would expect the 3-D DWT to outperform the 2-D DWT for MRI. Wang and Huang [7] show the 3-D DWT to outperform the 2-D DWT by 40% to 90% in terms of compression ratio. The advantages of DWT over the DCT can be summarised as follows:

1. The DCT difference coding is computationally expensive;
2. The wavelets do not generate blocking artefacts, which are unacceptable in medical images; and
3. The DCT is not optimal for medical image compression, even if it performs well on video [1], [4], [5], [62].

Currently, 3-D transforms have been implemented with other methods, such as a network of computers [7], but a chip dedicated to this transform will give tremendous results. Despite its complexity, there has been an interest in 3-D DWT implementation on various platforms. Existing survey exhibits that the research can be classified into three categories: architecture development [63–65] , architecture with FPGA implementation [9], [10], [66–68], and finally architecture that has been implemented on other silicon platforms [69]. Since the aim and contribution of this work are on the reconfigurable architecture, the reviews and discussions will be focused on FPGA-based implementation only.

In [66], a memory-efficient real-time architecture with an optimised memory requirement to the order $O(KN^2 + (K-2) \times N)$ is proposed. The architecture as illustrated in Figure 2.5 is considered as lower-power and works at a lower frequency which fully utilises the advantages of multiplierless structure of filter design, parallelism and pipelined structures of filter design. Canonical signed digit (CSD) multiplier is used to realise the transform. In terms of operation, in one clock cycle, K number of data points, each from one memory unit T_0 to T_{K-1} are fetch for temporal decomposition. The temporal decomposition unit, PE_T computes DWT of the corresponding data

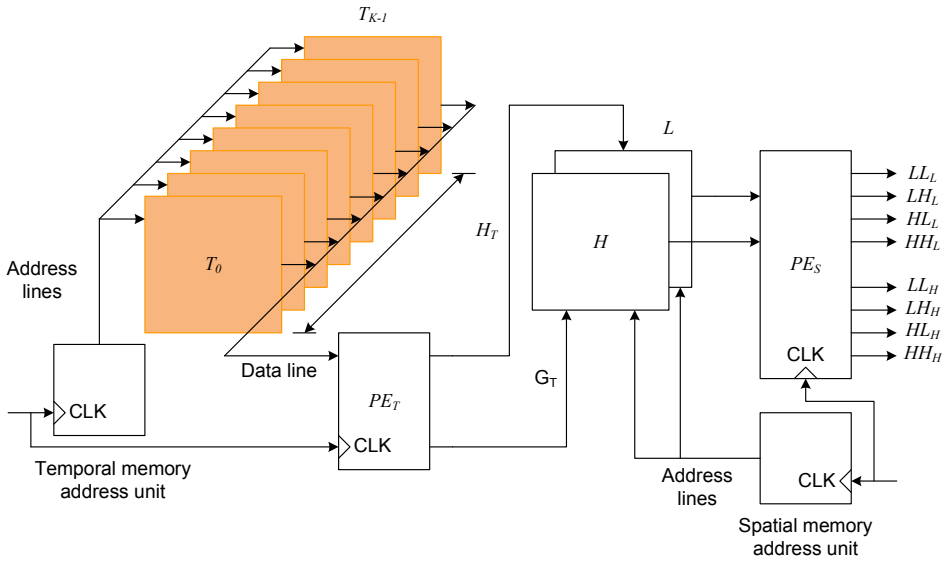

Figure 2.5: Block architecture for the 3-D DWT [66].

points, and stores back into the memory unit L and H. The proposed architecture is implemented on Xilinx FPGA devices, and operating at 75 MHz. However, the drawback of this architecture is the reduction in compression of video sequences and modifications are required for better performance optimisation.

Two architectures are presented in [9] and [10] for bit-parallel and bit-serial implementations, respectively. The architecture for bit-parallel as depicted in Figure 2.6 is based on distributed arithmetic (DA), and suitable for real-time medical imaging applications. DA concept requires the following operations: read only memory (ROM) accesses, addition, subtraction and shift operations of the input data sequence [70].

Utilisation of top-merged DA methodology in the design contributes to area reduction in polyphase sub-filters, thus make it possible to map three pipelined sub-filters into one FPGA for x, y and z-direction analysis. To reduce the required memory size for intermediate results, the $128 \times 128 \times 128$ volume is divided into eight blocks of $68 \times 68 \times 68$ pixels with fours-pixels overlap between adjacent blocks. In this case, the size of intermediate random access memories (RAMs) is ten lines of 68 pixels and ten blocks of 68×68 pixels. It is worth noting that the block arrow implies a parallel input of nine-pixels for y and z-direction sub-filters. In terms of outputs,

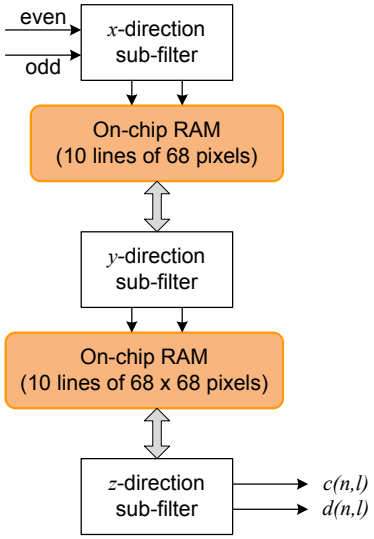

Figure 2.6: 3-D DWT processor architecture [9].

all polyphase sub-filters generate two outputs in every cycles for $c(n,l)$ and $d(n,l)$. The proposed architecture is implemented using very-high-speed integrated circuit hardware description language (VHDL) and synthesised on Xilinx Virtex-E FPGAs. With the advantages offered by the DA design technique, the architecture was proven as area-efficient high-throughput with the processor running at a frequency up to 85 MHz and capable of process five levels DWT analysis of a $128 \times 128 \times 128$ functional magnetic resonance imaging (FMRI) volume image in 20 ms.

Another hardware design and implementation of a new efficient 3-D DWT for video compression application is discussed in [67] by Ismail *et al.* The top level of system implementation is shown in Figure 2.7 with the "3D_V system" and a memory block. The "3D-V system" consists of a control unit, some counters and a block of "Conv_1d". The block of "Conv_1d" is simply the 1-D DWT filter bank with direct- and transposed-form finite impulse response (FIR) filter structures.

In this work, comparison for both filter structures is carried out and after the FPGA implementation, transposed-form FIR filter generates better achievement of latency and chip area. Thus, it is selected as filter structure in the "Conv_1d" block. The design is implemented on a device from Altera. It exhibits low memory

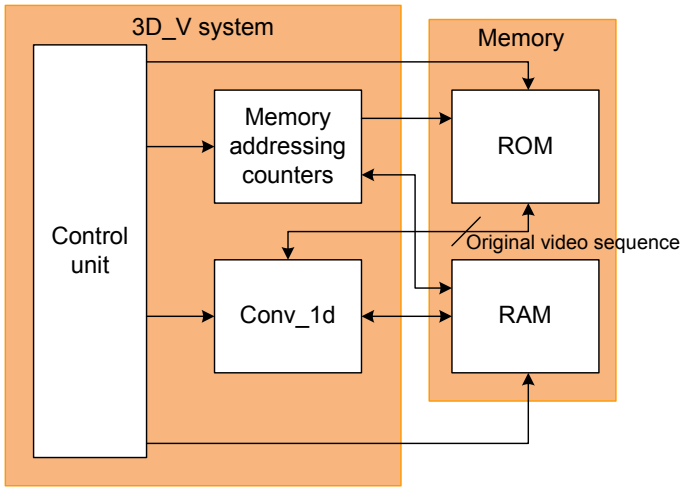

Figure 2.7: Design of 3D-V temporal decomposition system [67].

demands and shorter latencies for the compression and decompression processes. The advantages of the design are as follows: it is generic to any wavelet filter coefficients and scalable to fit for any frame size of the video sequence.

Haar wavelet transform (HWT) implementation on FPGA platform as illustrated in Figure 2.8 is proposed for video segmentation in traffic monitoring by Salem *et al.* [68].

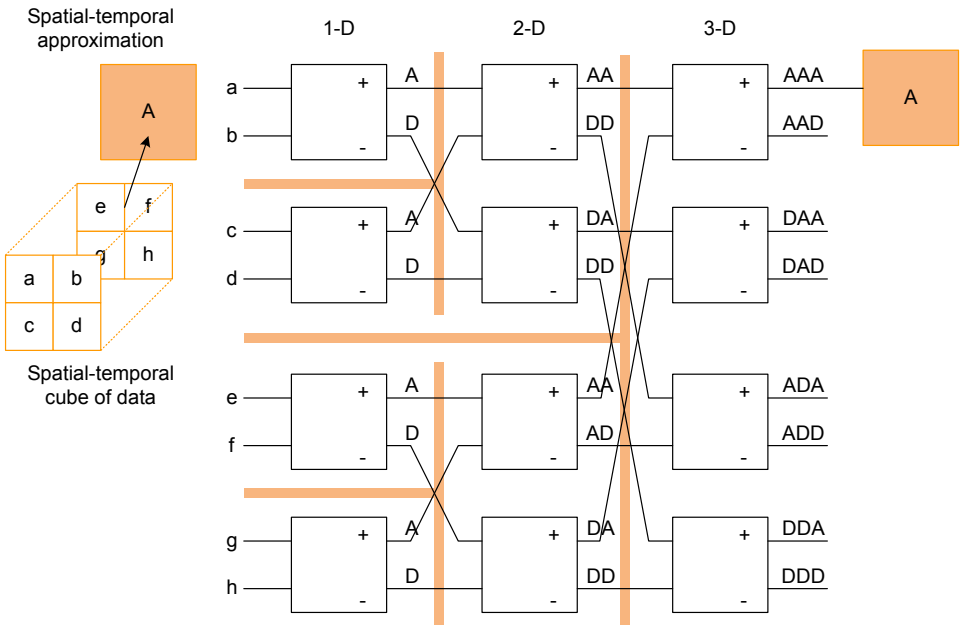

Figure 2.8: Hardware design for the 3-D Haar wavelet transform [68].

In this work the 3-D wavelet transform is used to detect moving objects or object groups as regions of interest (ROI). For the implementation, 12 basic components are used in three layers. The inputs are eight data points for a spatial-temporal cube, whilst the outputs are the eight coefficients, the conventional eight sub-bands of the 3-D transform. Notation "A" refers to the approximation operation, which is addition and right shifting. On the other hand, "D" represents the detailed operation (subtraction and right shifting).

The implementation demonstrates the system capability to deliver through 100 Mbit ethernet the acquired images. Moreover, the achievement of the primary segmentation is in rate of 25 frame per second (fps) phase alternate line (PAL), or 30 fps national television system committee (NTSC), which is the acquisition rate achieved by the digital camera. Furthermore, the design consumes 63% of slice utilisation, and the maximum operating frequency is 100 MHz.

Table 2.3 summarises the existing FPGA implementations of 3-D DWT [9], [66–68], in terms of FPGA devices, filter that has been used, as well as the design parameters of area and maximum frequency.

Table 2.3: Comparative study of the 3-D DWT architectures and the FPGA implementations.

Parameters	**Architectures and FPGA Implementations**			
	[9]	[66]	[67]	[68]
FPGA	V300EFG256	XCV50TQ144	EPF10K200	XC2VP30
Filter	Daubechies	Daubechies	Daubechies	Haar
Arith. tech.	DA	CSD	N/A	N/A
Area (slices)	1,271	738/768	LCs = 7.94%	8,663/13,696
Max. freq. (MHz)	85.00	75.00	N/A	101.25

Note:
LCs: Logic cells, Arith. tech.: Arithmetic techniques
Max. freq.: Maximum frequency, N/A: Not applicable

Due to the fact that the implementation of 3-D DWT on FPGA is still an active research area, for any comparative study the following justifications need to be taken into consideration:

1. Proposed architectures in [9], [66–68] were targeted for different FPGA platforms and applications, and using various design techniques. Moreover, with different FPGA resources, process technology and circuit topology available on each device, it draws inconsistent comparison and performance evaluation; and

2. With different design techniques and target applications, each architecture performs with different performance trade-offs.

2.3.2 FPGA-based Architectures for Finite Radon Transform (FRAT)

The contributions of transform domains in various applications including image de-noising, enhancement and compression are undebatable facts. As an example, the wavelet transform has been extensively used as a solution to the problem of the short time Fourier transform (STFT), and excels in isolation discontinuities and spikes [71]. However, the wavelet suffers from inflexible directionality, as it does not isolate the smoothness along edges. This demerit of wavelet is well addressed by the ridgelet and curvelet transforms, as they extend the functionality of wavelets to higher dimensional singularities, and it is proven as an effective tool to perform sparse directional analysis [71]. The basic building block of these transforms is the FRAT.

An orthonormal, digital and fully invertible form of ridgelets, called the finite ridgelet transform (FRIT) [71], [72] and the FRIT allows superior compaction over wavelets as a result of its directional nature. Therefore, better performance can be achieved in image compression and de-noising applications [71].

Since medical images contain several objects and curves, doubtless the curvelet and ridgelet as well as the basic building block of the FRAT play a major role for

better image analysis. By offloading the intensive processing procedures of these transforms into a properly hardware platform, computational acceleration can be achieved, meanwhile maintaining the outcomes quality. In the following, several hardware implementations of FRAT are discussed and evaluated.

In [73], two architectures are proposed, a generic and standard FRAT based pseudo-code. Figure 2.9(a) shows a generic architecture that uses a combination of look-up tables (LUTs), matrix of accumulators, and multiplexers to perform the FRAT with time complexity $O(p^4)$, where p is the block size in pixels. The second FRAT architecture as depicted in Figure 2.9(b) is based on the standard FRAT pseudo-code presented in [74] has a core time complexity of $O(p^4 \cdot (p+1))$.

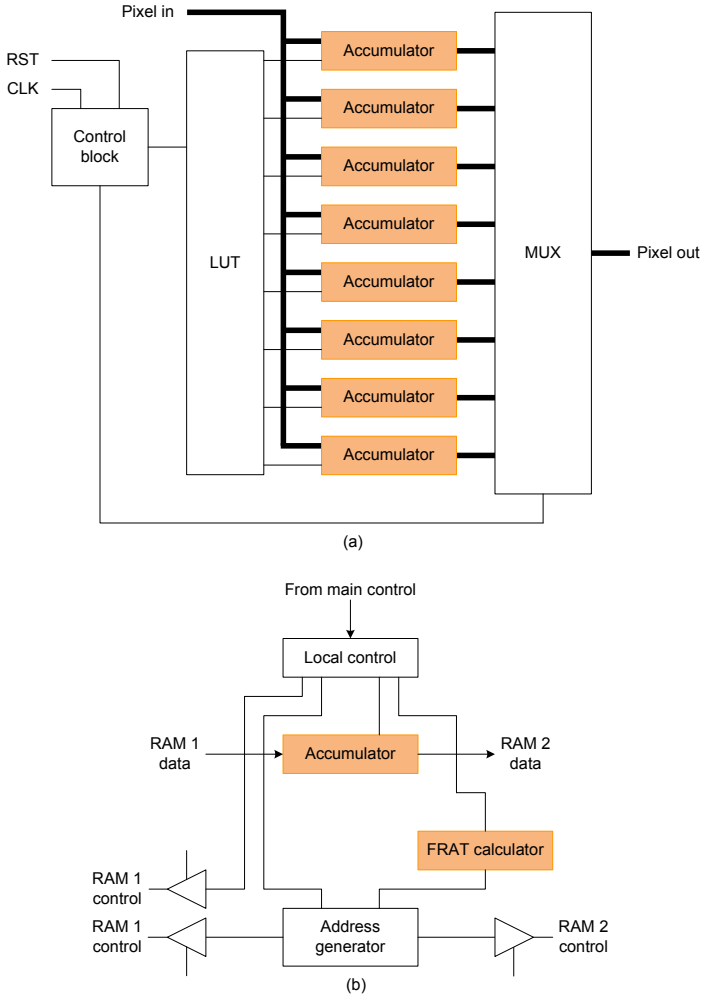

Figure 2.9: Proposed architectures (a) Generic transform architecture (b) Radon transform module [73].

In this architecture, the "Radon transform module" contains a "FRAT calculator" which uses address generators, accumulators, random access memories (RAMs) and local control logic to perform the FRAT iteratively. The work shows that by offloading some of the processing work into a properly configured FPGA, speeds can be achieved in excess of one hundred times faster than current high end servers.

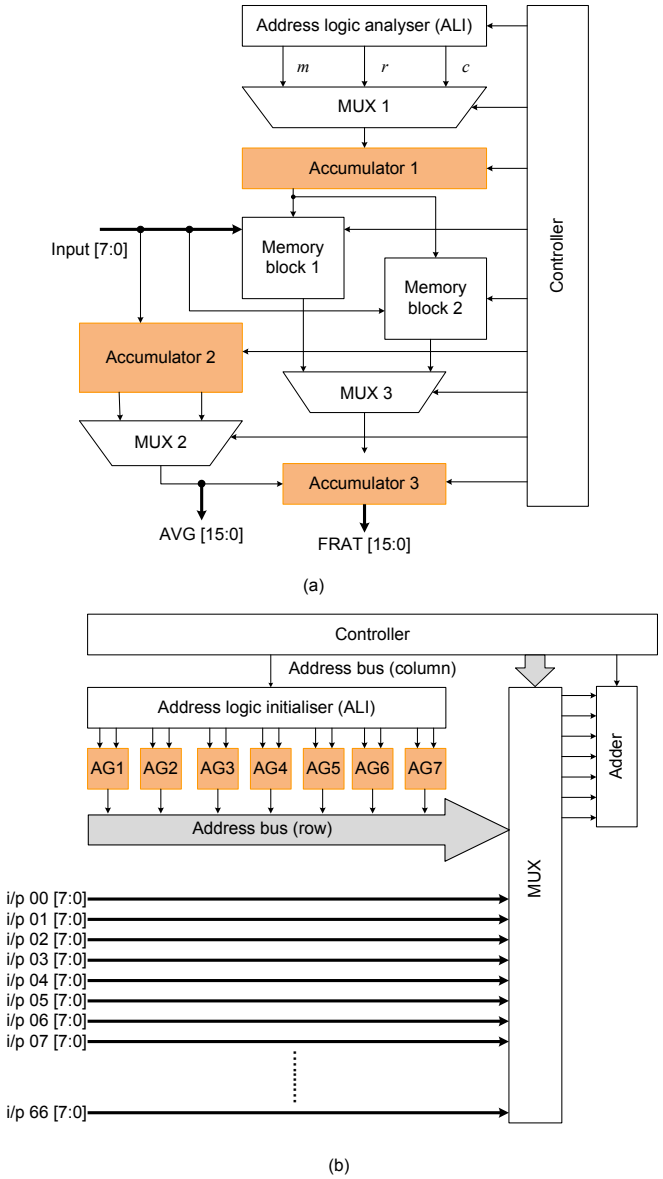

Figure 2.10: (a) Reference FRAT architecture (b) Memoryless FRAT architecture [75].

Two architectures of FRAT's FPGA-based implementation are described in [75]. The first architecture as shown in Figure 2.10(a) is a direct hardware implementation of

a suitable modified variant of the standard FRAT pseudo-code [74], is called a reference FRAT architecture. This reference architecture comprises an address logic initialiser (ALI), multiplexer, accumulators and two memory blocks for storing transform vectors. The second architecture as illustrated in Figure 2.10(b) is denoted as a memoryless FRAT architecture which operates in a parallel manner with p times the throughput of the first architecture. The ALI, wide multiplexer and adder blocks are used as sub-blocks in this architecture. The ALI drives the address generators (AGs) that in turn generate signals to control the address bus. Moreover, the multiplexer operates on all the p^2 pixels simultaneously, hence, scalability may not be inherently feasible as a result of its wiring complexities. The proposed architectures use 7×7 size image blocks and are prototyped for processing common intermediate format (CIF) image sequence. The simulation and synthesis results on Xilinx Virtex-II FPGA show that the core speeds of the two proposed architectures are around 100 and 82 MHz, respectively.

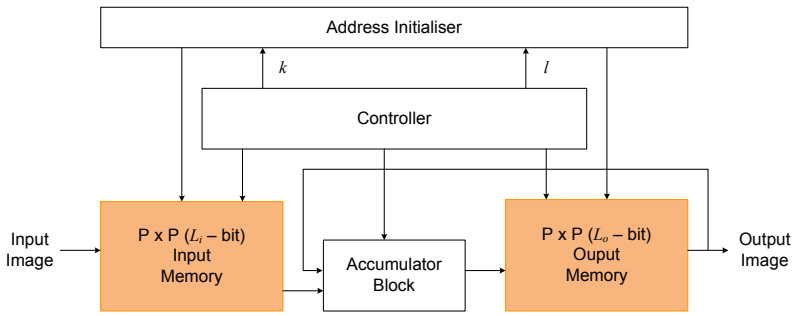

Figure 2.11: Block diagram of proposed FRAT implementation [72].

In [72], the proposed FRAT architecture as shown in Figure 2.11 is a direct hardware implementation of the pseudo-code that uses ALIs and a controller to control the accumulators and the memory blocks. The ALI along with controller block constitute the AG that generates addresses, i.e., $L_{k,l}$ for memory blocks. The accumulator is a L_O – bit accumulator that accumulates the l^{th} pixel value for the k^{th} Radon projection. The controller block organises the flow of this process with input and output data flow. To validate the performance, the design is ported to a Xilinx Virtex-II FPGA chip using Handel-C. Moreover, there is no specific optimisations to

the Radon block, since this work is mainly for FRIT implementation.

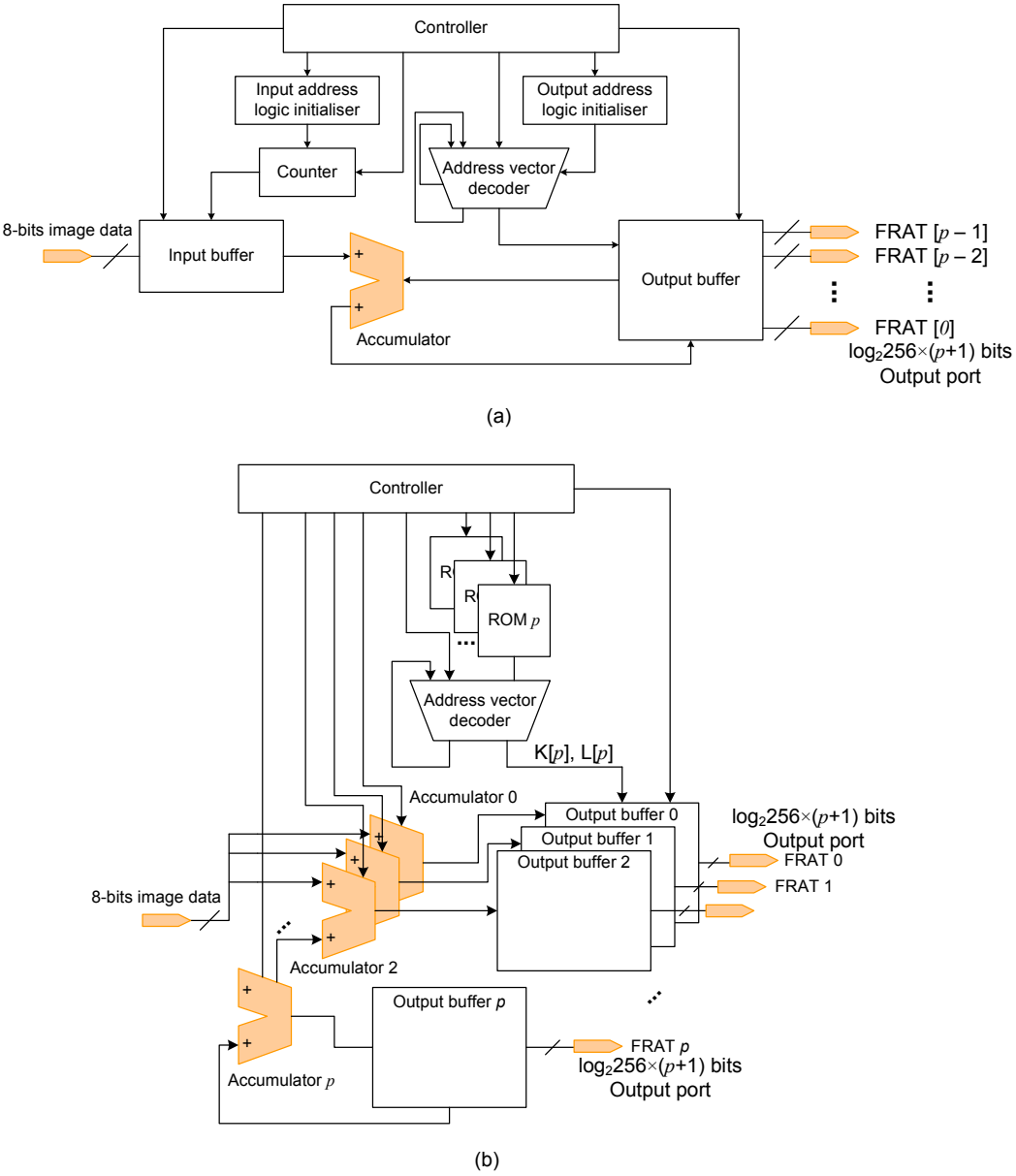

Figure 2.12: (a) Serial architecture (b) Parallel architecture [76].

In [76], two power efficient intellectual property (IP) core architectures for the FRAT with time complexities $O(p^2(p+1))$ and $O(p^2)$, respectively, are proposed. The first architecture is a serial architecture based on the FRAT pseudo-code and optimised for FPGA implementation. The second architecture is based on a parallelised version of the FRAT pseudo-code and consists of a combination of read only memory (ROM) based control logic and array based buffers for input/outputs (I/Os). Both are

illustrated in Figure 2.12(a) and (b), respectively.

In the first architecture, one input pixel is processed in each clock cycle. The merit of a serial input architecture is that the FRAT block can be easily included into a sequence of image processing/compression steps such as the ridgelet or curvelet, without imposing any restrictions on the nature of the inputs. No clock cycles are wasted in buffering the whole input block, and the input section can be pipelined. The controller has $p+1$ counters, which generate the address and the read/write status of the output buffer. Each accumulator reads the contents of the specified location from the output buffer, adds the data from the input port, and writes it back to the same location of the output buffer, all within the same clock cycle, with p^2 clock cycles. For the second architecture, the input buffer is a linear distributed RAM with p^2 address locations, with p as the block size. On the contrary, the output buffer is a linear array of shift registers with p locations. Both architectures are implemented on the Virtex-E FPGA series, and prototyped on the Celoxica RC1000 development board.

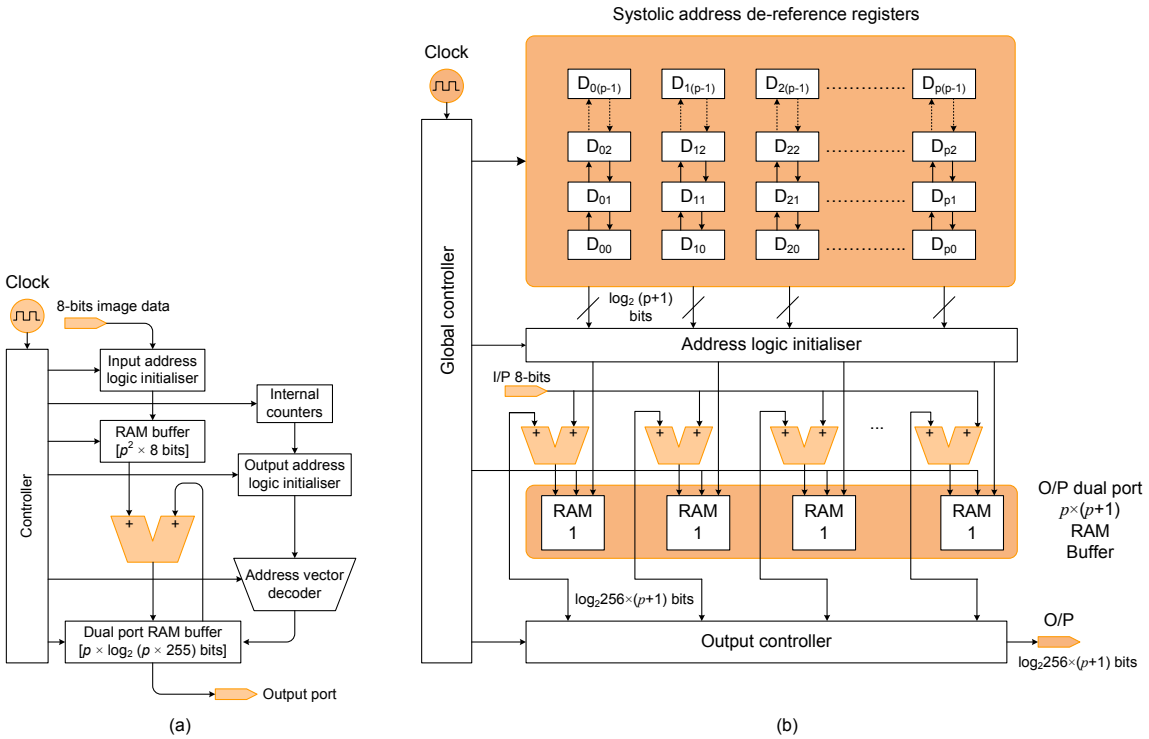

Figure 2.13: (a) Reference architecture (b) FRIT architecture with the FRAT [71].

The most recent implementation of the FRAT is described in [71]. A novel parameterisable, scalable and high-performance core of the FRAT sub-block is presented as shown in Figure 2.13(a) and (b). To provide with design reusability, the core of the FRAT is developed using Handel-C and an efficient VHDL core from Xilinx CoreGen. Two main strategies are imposed in this architecture: remapping the FRAT pseudo-code in the order of input signals and implementing a systolic array to store the address de-referencing values rather than using multiplexer or counter chains. The design is prototyped on the Celoxica RC1000 board containing the Xilinx XCV2000E FPGA, and results obtained reveal the design time and area complexity are $O(p^2)$ and $O(2p^2)$, respectively.

In addition to the summary of existing FRAT implementation on FPGA that have been discussed [71–73], [75], [76], Table 2.4 lists the important issues, such as the FPGA devices involved, programming approaches as well as the target applications.

Table 2.4: Summary of FPGA-based architectures of FRAT.

Refs.	FPGA Devices	Programming Approaches	Target Applications
[71]	Virtex-E	Handel-C and CoreGen	Image processing
[72]	Virtex-II	Handel-C	Image processing
[73]	Virtex-E	N/S	Image processing
[75]	Virtex-II	Verilog	Image processing
[76]	Virtex-E	Handel-C	Image processing

Note:
N/S: Not stated

In terms of programming approaches, Handel-C is used in [71], [72] and [76], whilst the target FPGA devices change inline with the FPGA technology advancement. For the target applications, previous works reported are mainly for general image processing. It can be seen from the test images used for the proposed architecture validation. To clearly show the innovation and improvement from the existing implementation of FRAT on the FPGA, each contributor with their important features and invention are depicted in the research time line as shown in Figure 2.14. In short, the research time line is also important to draw the research gap as well as to avoid

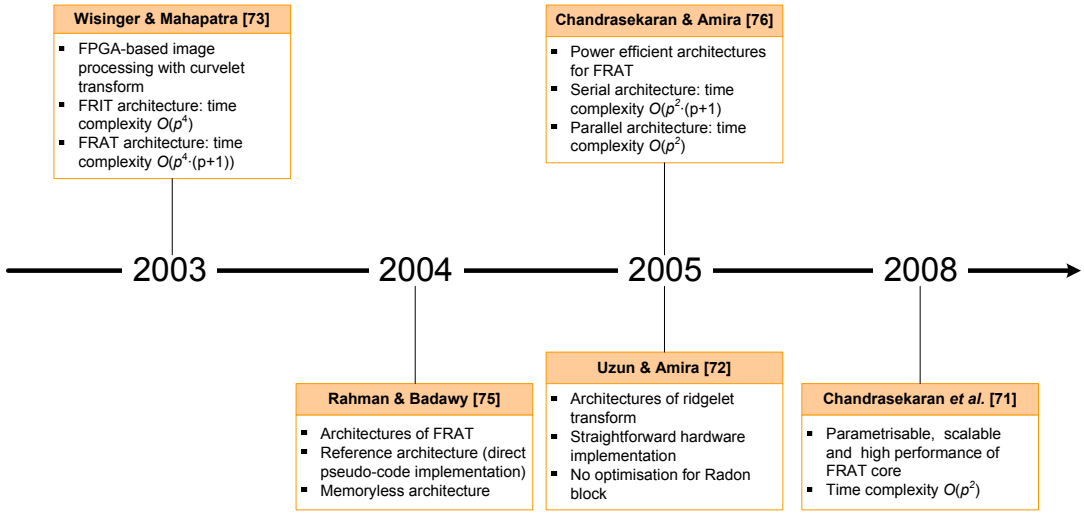

Figure 2.14: Review of FRAT's FPGA-based implementation.

any research duplication.

Hardware Implementation of Image De-noising

Despite the fact that the software experimentation of image de-noising [77–83] has gained much effort from the research community, it is worth noting that very few effort in hardware implementation.

In [84], the parallel implementation of an advanced wavelet-domain noise filtering algorithm, which uses a non-decimated wavelet transform and spatially adaptive Bayesian wavelet shrinkage is discussed. The proposed wavelet-domain video de-noising algorithm using FPGA is given in Figure 2.15.

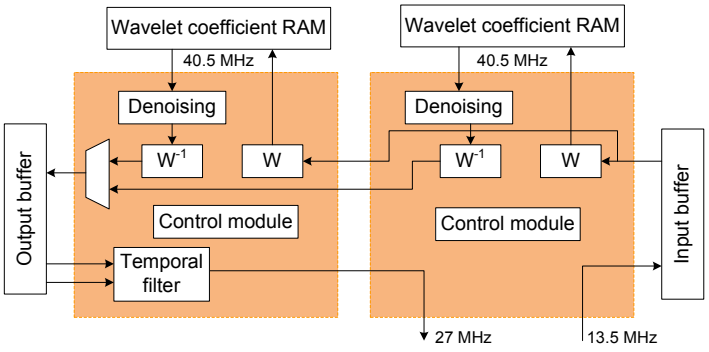

Figure 2.15: FPGA implementation of the proposed wavelet-domain video denoising algorithm [84].

In this implementation, two independent modules are working in parallel, and each module is implemented in a different FPGA. The idea of using two FPGAs is to execute the wavelet decomposition of an input TV frame as well as the inverse wavelet transform of the previous TV frame. These two modules can switch their roles in time. The wavelet-domain de-noising block is located in front of the inverse wavelet transform. The implemented arithmetic is de-centralised and distributed over two FPGAs. The standard composite television video stream is digitalised and used as a source for real-time video sequences. The results demonstrate the effectiveness of the developed scheme for real-time video processing. As the system implemented is computationally demanding with a de-noising filter algorithm, the solution based on two different FPGAs in this implementation is inefficient, hence, partial reconfiguration could be taken into account for better system implementation as well as maintaining the performance.

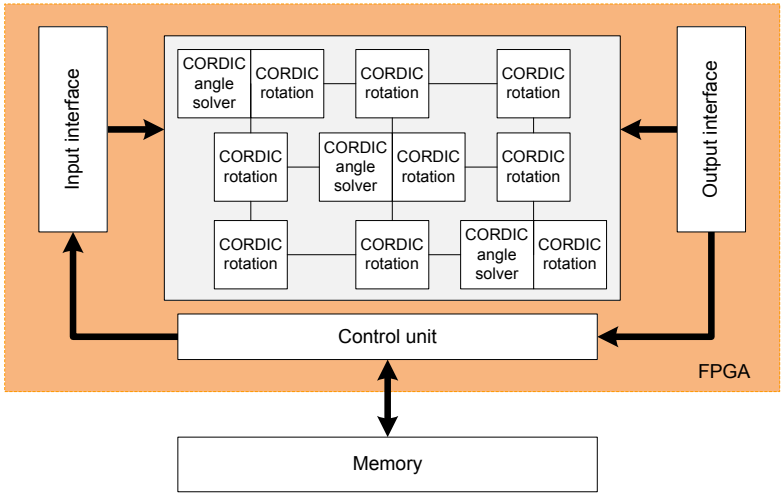

Figure 2.16: FPGA implementation of the SVD/EVD array [85].

An FPGA implementation of a block singular value decomposition (SVD) method for image de-noising is presented in [85]. This method exploits the fact that only the smallest SVD eigen values are affected by the noise and therefore, can be discarded leading to an efficient non-linear image filtering. An efficient architecture for SVD based on the Brent. Luk, Van (BLV) loan systolic array is proposed as depicted in Figure 2.16, and comprises a control unit, an input and output interface, and the

SVD/eigen value decomposition (EVD) array.

The control unit generates commands to the processors and controls all the I/O operations between the FPGA and the off-chip memory as well as the host personal computer (PC). The operation of the system implementation as follows. The data matrices are read/written from/to the off-chip memory element by element, thus the I/O requirement is independent from the matrix size. Shift buffers are used as the input and output interfaces, and it is used to transfer data and instruction in serial from the control unit to the systolic array, and the output data from the systolic array to the control unit. The coordinate rotation digital computer (CORDIC) algorithm [86] is used for the implementation of plane rotations. From the hardware implementation perspectives, the systolic array is promising approach for consideration. It is proven that the architecture exhibits three times more efficient and faster than the existing BLV structure. In this implementation, RC1000 rapid prototyping board and high-level language for hardware design, Handel-C are fully used.

Another implementation of a flexible hardware architecture for performing the DWT on a digital image is addressed in [87]. With lifting schemes, the proposed architecture provides advantages: small memory requirements, fixed-point arithmetic implementation, and a small number of arithmetic computations. In terms of image de-noising experimentation results, simple wavelet technique demonstrates in improved images up to 27 dB PSNR. The DWT core is modeled using MATLAB and VHDL, and implemented on Xilinx FPGA devices. Results for the hardware implementation show around 15,000 gates are utilised, at 2.185 MHz maximum clock speed and 24 mW power consumption. In brief, this work gives a significant idea of lifting schemes to be considered for the image de-noising design and implementation.

A hardware architecture for video noise estimation is discussed in [88] by Lapalme *et al.* Although the implementation is about noise estimation, the problem of noise estimation as well as noise elimination is remaining in the same boundary. Figure 2.17 shows the top-level architecture of the system on Virtex-II (XC2V4000) FPGA device. The aim of this architecture is to process two consecutive fields

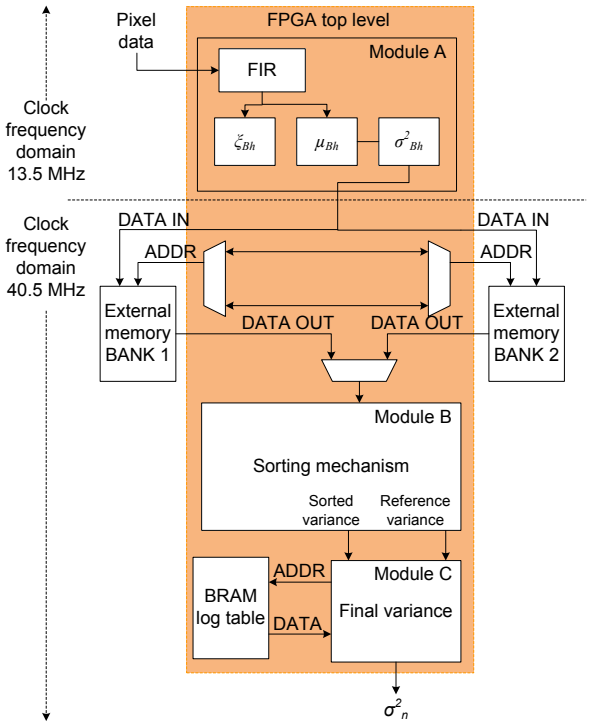

Figure 2.17: Block diagram of the proposed FPGA design [88].

simultaneously by acquiring data within "Module A" and processing data within "Module B" and "C". This concept implies a "ping-pong" structure and refers to the back and forth processing scheme to enable the parallel processing. To store and read data concurrently, simultaneous but separate accesses of two external memory components ("Bank 1" and "2") are being used. "Module A" generates the variance, $\sigma^2{}_{Bh}$, the sample mean, μ_{Bh}, and the homogeneity measures, ξ_{Bh}, of each block contained in a field. In case of "Module B" and "C", these two modules responsible for sorting the homogeneity values of the previous field, and works to find the logarithmic value of the variances by querying the logarithmic LUT, respectively. The logarithmic values are compared to an application-dependent threshold and the remaining valid variances are added and averaged to generate the final variance, $\sigma^2{}_n$. The advantages of this architecture are the design which is uniformly pipelined, and digital clock managers (DCMs) are used to accelerate the sorting mechanisms clock frequency, as well as to maximise the parallelism of arithmetic operations to achieve the real-time requirements. In terms of hardware implementation strategy, the use of VHDL generics

and packages contributes to scalable architecture that can accommodate different filter sizes. Moreover, the experimental results reveal that the noise variance generated was proven to be accurate in comparison to the original software simulation findings.

To conclude, Table 2.5 lists the summary for hardware implementations of image and video de-noising; classified in terms of algorithms or methods implemented, FPGA devices and target applications. Design strategies that have been manipulated such as systolic array, FPGA resources optimisation and generics design style provide a significant guidance for future work to be carried out on hardware implementation of image de-noising using the FRAT in Chapter 4.

Table 2.5: Hardware implementation of medical image de-noising.

Refs.	Algorithm/Method	Platform	Target Applications
[84]	Advanced wavelet-domain noise filtering	XC2V6000-5	Video de-noising
[85]	SVD	XCV2000E-6	Image processing
[87]	Cohen-Daubechies-Favreau (CDF) 5/3 and CDF 9/7 DWT	XCV300	Image processing
[88]	Structure-oriented noise estimation	XC2V4000	Interlaced phase alternate line (PAL) video sequence

2.3.3 FPGA-based Architectures for Context-based Adaptive Variable Length Coding (CAVLC)

CAVLC that has been adopted in H.264/AVC plays an important role in compression, especially for high resolution demand applications [89], [90]. CAVLC is used to encode residual quantised integer DCT coefficients and based on the previously encoded data, CAVLC can adaptively choose one of the several variable length coding (VLC), and then encode the current input signal efficiently by using various syntax elements [91], [92].

By assigning shorter codewords to infrequently occurring symbols, CAVLC can significantly reduce data redundancy and finally lead to higher coding efficiency even at low-bit rates with the cost of increased computational complexity. With the cost to

pay for high computational demand, it is essential to have a hardware implementation of the process required [89], [90].

In H.264 standard, the integer transform (IT) operates on residual blocks after motion-compensated prediction or intra-prediction. The residual block is obtained by subtracting the prediction block from the current blocks. Better compression can be achieved with successful prediction, that reduces the energy in the residual with fewer bits. Several research [93–95] propose hybrid schemes using both DWT and DCT for medical image compression. The DCT is applied to the DWT details, which generally have zero mean and small variance, thereby achieving better compression performance.

An existing implementation based on ESCOT and SPIHT are reported in [96–98]. A coding system for lossy-to-lossless compression of medical volumetric data is proposed in [96] by Xiong *et al.* Results obtained in [96] demonstrate that the intra-band coding of 3-D ESCOT [97] performs about 1 dB better than the inter-band coding of 3-D SPIHT [98] at 0.1 and 0.5 bit/voxel, with the expense of 50% more complexity. The commonly used implementation of 3-D SPIHT uses three linked lists and requires two passes [99]. However, these properties are not well adapted to a parallel low-power FPGA implementation, particularly for the utilisation of partial reconfiguration. To address these issues, the work presented in Chapter 5 describes a parallel implementation of the CAVLC coder that is applied to the coefficients resulting from the transform block. In the following, several existing FPGA-based implementation of CAVLC are examined.

FPGA- and application specific integrated circuit (ASIC)-based implementation of a high-performance and low-power hardware architecture for real-time implementation of CAVLC are proposed in [100]. Verilog is used to implement the architecture and it is targeted for a low-power H.264 video coding system in portable applications. Xilinx Virtex-II device is employed for FPGA prototyping at 76 MHz, and the design is verified to work at 233 MHz in 0.18μm ASIC implementation. As shown in Figure 2.18, the architecture performs CAVLC for a macroblock, in

the worst case, in 2,880 clock cycles. In this system, `Coeff_Token` "Table Selection Unit" calculates the parameter "nC" based on the number of non-zero coefficients in the left-hand and upper previously coded blocks, "nA" and "nB" respectively. The parameter "nC" is a function of the number of non-zero coefficients in neighbouring blocks and used for CAVLC when choosing an appropriate LUT. Experimentally, the FPGA and ASIC implementations are capable of coding 22 and 67 video graphics array (VGA) frames 640×480 per second, respectively. Although this work targeting a low-power application, there is no specific discussion on low-power design techniques.

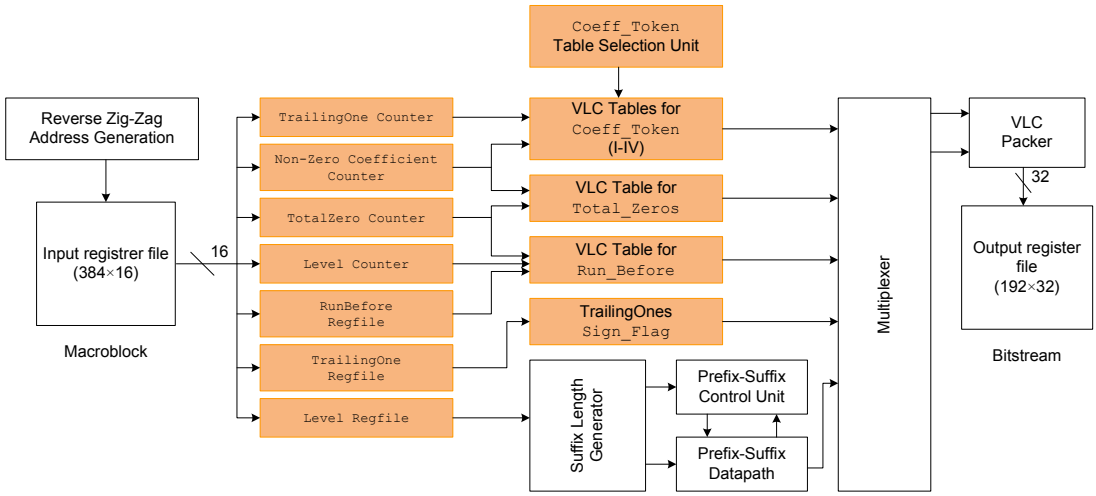

Figure 2.18: CAVLC hardware architecture [100].

A CAVLC encoder design for real-time mobile video applications is discussed in [101] by Rahman and Badawy. The block diagram of the proposed architecture is shown in Figure 2.19. In this architecture, the bit width of the transform coefficient is 13-bits. Luma/Chroma selector signal is set high for chroma block input or low otherwise, whilst N_U and N_L are the number of coefficients of the upper and left block of the input luma block, respectively. N_{U_valid} and N_{L_valid} are the input flags that indicate the availability of these neighbouring blocks. I_{P_valid} is set high for a single clock cycle when the input luma/chroma block is available. The O_{P_valid} is set high when valid data is available at the O_{P_data} output. When the processing of the current block is complete, the *Done_block* signal is set high and the coder waits for the next block input. The proposed architecture is realised as a very large scale

integration (VLSI) architecture of the CAVLC coder for the H.264/AVC baseline profile entropy coder and implemented on Xilinx Virtex-II (2V3000FG6764) FPGA. Results obtained show the design offers area saving by reducing the size of the statistic buffer. Moreover, the split VLC tables simplify the process of bitstream generation as well as reducing some area. This design is targeted for a real-time mobile video applications and simulation results demonstrate the architecture capability to process CIF/quarter common intermediate format (QCIF) frame sequences in real-time at 50 MHz with 6.85-K logic gates.

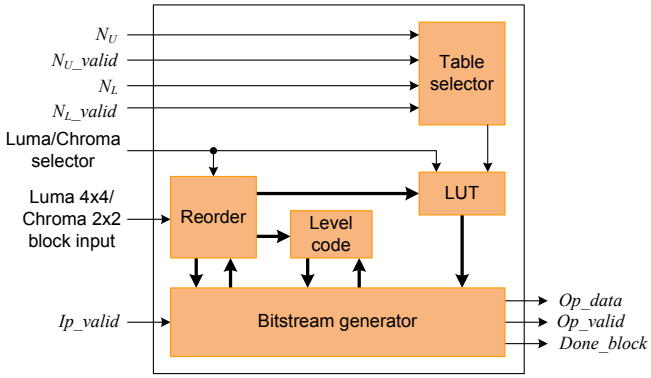

Figure 2.19: The proposed CAVLC architecture [101].

A high-performance parallel CAVLC encoder implemented on a fine-grained many-core system are shown in Figure 2.20(a) and (b) [102]. The system is composed of 164 simple 16-bits digital signal processors (DSPs), three hardware accelerators, and three 16-KB integrated big memories, all connected by a reconfigurable mesh network. Processors can directly communicate with their four nearest-neighbour processors, or distant processors using long-distance-capable configurable links. In terms of data flow of the proposed CAVLC encoder, the input residual coefficients are sent to the zig-zag and CAVLC scanning block for the first phase processing. After that, corresponding data elements are distributed to five different encoding units in parallel, and the final codes are assembled and packed by the packing unit. It is worth noting in this implementation, arithmetic table elimination and compression techniques are employed. Therefore, the data flow of the CAVLC encoder is partitioned and mapped to an array of 15 small processors. As this design is targeted for high-definition television (HDTV)

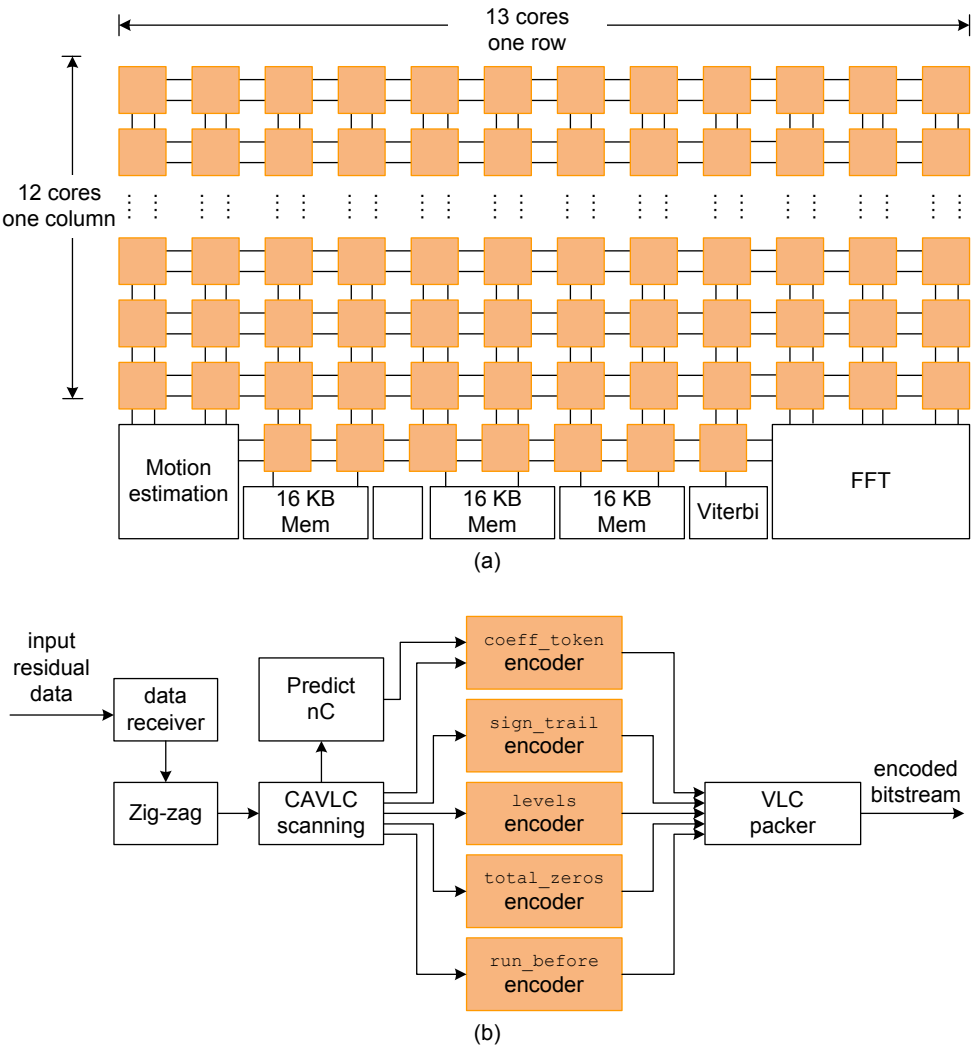

Figure 2.20: (a) Architecture of targeted many-core system (b) Data flow diagram of the CAVLC encoder [102].

application, the parallel workload of each processor is characterised and balanced to achieve throughput optimisation. Results achieved show that the proposed parallel CAVLC encoder reaches the real-time processing requirement of 30 frames per second (fps) for 720p HDTV. In terms of throughput and area utilisation, the proposed architecture has 4.86 to 6.83 times a higher throughput and requires far smaller chip area than the existing implementation on the general-purpose processors. Moreover, comparison with common DSP exhibits that the proposed design has approximately 1.0 to 6.15 times higher throughput as well as consuming less than six times a smaller area.

A full-hardware implementation of CAVLC targeted for real-time processing of 1920×1080 frame is addressed in [103] by Kim *et al.* The proposed architecture works to encode one block per 16 clock cycles, and it is implemented on Xilinx FPGA using Verilog. Equivalent gate counts for the implementation is shown in Table 2.6. Experimentally, this architecture enables real-time processing for 1920×1080 high-definition (HD) video frame with 30 fps, using 100 MHz clock.

Table 2.6: Equivalent gate for CAVLC items [103].

CAVLC items	Equivalent Gate
`Scanning`	1,362
`nCGen`	16,294
`Coefftoken`	298
`Levels`	2,249
`TotalZeros`	564
`RunBefore`	1,361
`PackData`	9,265

Another FPGA-based of CAVLC implementation is described in [104]. A parallel architecture of CAVLC encoder as shown in Figure 2.21 is implemented on FPGA of Cyclone II EP2C20F484, and the speed of the coding module is up to 165 MHz. The proposed architecture is focusing on the algorithm optimisation and there is no experimental validation with a real-time system. As an example, for the sake of enhancement of efficiency for VLC code table choosing, "nc" module is optimised and addressed in this study.

In this implementation, an inverse scan of zig operates for every 4×4 block. "T1 Counter" calculates and examines the number of `Tail_one coefficients` and `Level coefficients` included by 4×4 block. Next, the detected `Tail_one` symbol is written into `Trailones Reg` and "Level" symbol is written into "SIPO Level" buffer, which makes preparations for the implementation of "T1 Encoder" and "Level Encoder" parallel coding subsequently. After that, the number of every zero coefficients after the first `Total_coeffs` in a 4×4 data block scan-order is calculated by "TZ Counter". Then, "Run Counter" calculates the number of zero after every `Total_coeffs` in a 4×4

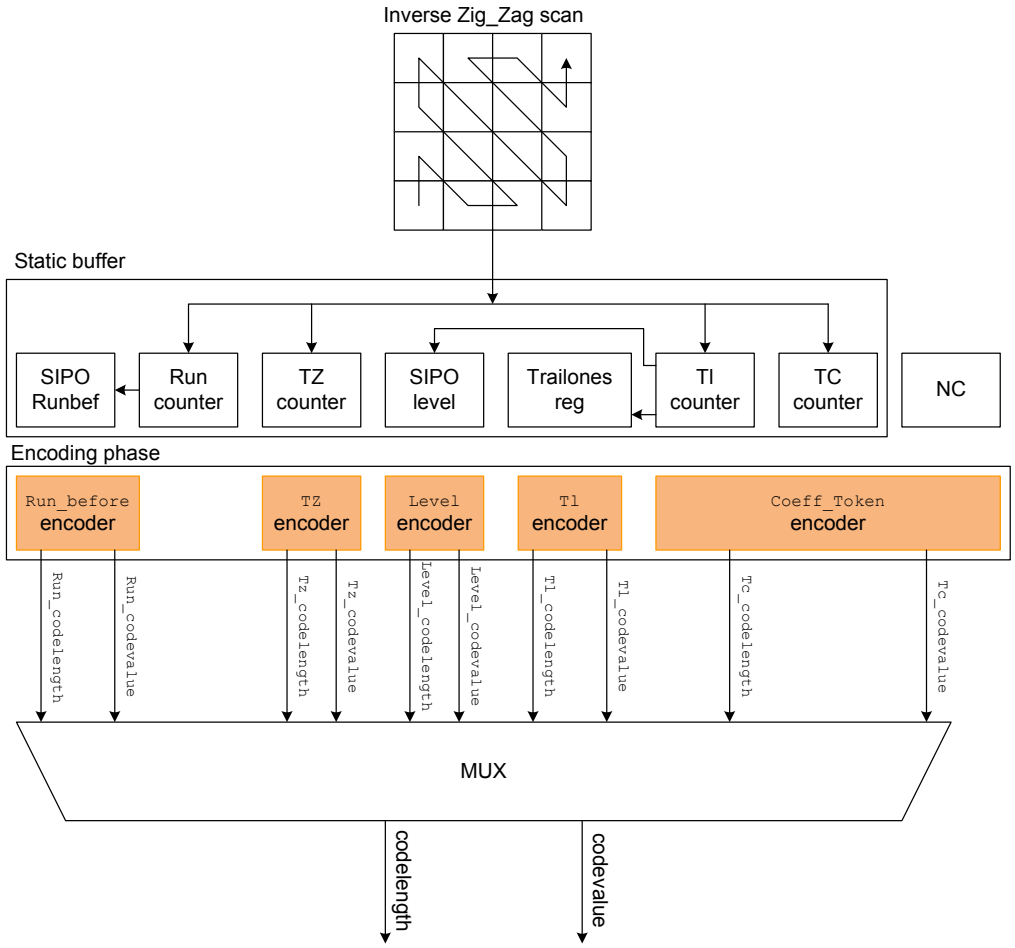

Figure 2.21: Framework of CAVLC encoder [104].

data block, and then sends the outcome to the "SIPO Runbef" buffer, which makes preparations for "Run-before" character coding afterwards. Finally, the code stream is disseminated through "MUX" module.

In brief, the examination of existing FPGA-based implementations for CAVLC is summarised in Table 2.7. Since the software implementation of CAVLC does not fulfill the real-time processing requirement as well as the growing demand in high-quality and low-bit rates in various applications, hardware implementation is gaining popularity. However, most of the work reported only focuses on the CAVLC itself. Thus, there is a huge potential to further research on the CAVLC implementation for compression system, especially for 3-D medical image compression system.

Table 2.7: Summary of hardware implementation of CAVLC.

Refs.	Platform	Target Applications
[100]	FPGA and ASIC	VGA frames
[101]	FPGA	CIF/QCIF video sequences
[102]	DSP	720p HDTV
[103]	FPGA	HD1080i
[104]	FPGA	N/A

2.4 Dynamic Partial Reconfiguration (DPR)

In SRAM-based FPGAs, full-device reconfiguration is required upon power-up [105]. The process of initialisation involves the FPGAs to be programmed with a configuration bitstream file. Partial reconfiguration concept appears after intialisation and works to modify a fraction of the resources by programming the FPGA with a partial bitstream file. Obviously, a full bitstream size is very massive whereas a partial bitstream may represents only 2% of the full bitstream [50], [105–107]. With smaller bitstreams, several advantages can be achieved: reduced reconfiguration time, reduced storage requirements, and dynamic allocation of functionality.

DPR is a technique that offers in changing the configuration of a part of a circuit while the rest of it executes its task [108]. During the last decade, DPR has been widely studied in various fields: the development of Erlangen slot machine (ESM), in FPGA-based reconfigurable computer [109], in automotive applications for video-based driver assistance applications [110] and a waveform-like reconfiguration for DPR in [111].

In partial reconfiguration modules, the number and sizes of the tiles and the arrangement within a partially reconfigurable region (PR region) can be defined as follows: 1-D, multi-1-D and 2-D partitioning [112] and an overview of the partitioning schemes is illustrated in Figure 2.22(a) – (c).

From designer and FPGA technology perspective, partial reconfiguration enables designers to realise the implementation of multiple modules in complex system on one

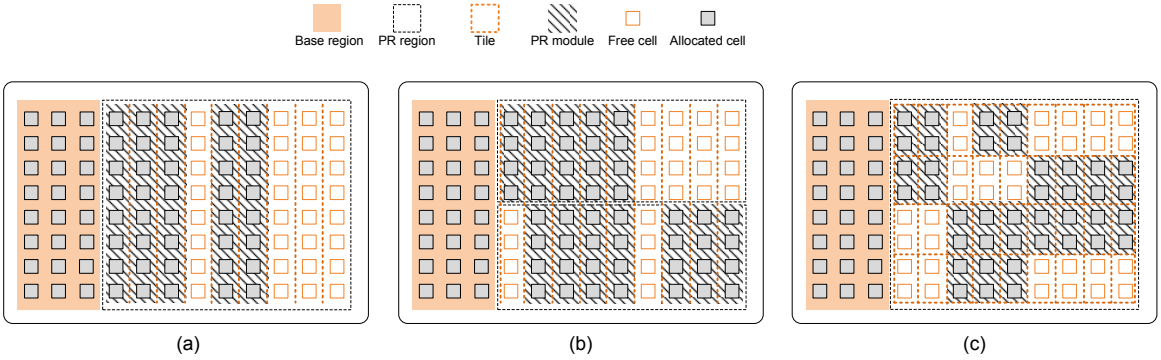

Figure 2.22: Overview of the partitioning scheme approaches (a) 1-D (b) Multi-1-D (c) 2-D [112].

FPGA device where a static area must be defined for components communication, while full reconfiguration implementation treat the entire FPGA as one module [106]. Xilinx provides many FPGA devices that support active partial reconfiguration, ranging from the Spartan-3 device to the latest Virtex-5 device. The active or dynamic reconfiguration allows the modification of a part of the FPGA while the remaining part of the device is active [113]. Thanks to the internal configuration access port (ICAP) the reconfiguration management can be implemented in the FPGA so that the final system is self-reconfigurable [114], [115].

Basically, a dynamically and partially reconfigurable system based on commercial FPGAs is composed of a static part and different dynamic parts. The communication between these parts requires the use of bus macros. In Virtex-II family this bus macros are implemented using tri-state buffers which is not very effective. A new alternative based on slice macro in form of predefined routed macro has been introduced in more recent devices (Virtex-4, 5 and 6) [116]. These macros are placed at the edge of boundaries separating dynamic and/or static region modules and allowing the communication between different regions of the FPGAs.

A novel FPGA-based scalable architecture for DCT using DPR is presented in [106]. The proposed scalable architecture is based on DA technique, and it works in two different modes to reduce computational complexity and power consumption. It has been reported that with DPR mechanism, the processing elements of the DCT

architecture can be changed easily and the scalable architecture can be adjusted to perform different types of DCT zonal coding.

Moreover, the scalable architecture can also reconfigure the unused DCT computation with the motion estimation module to fully utilise DPR advantages. The proposed architecture provides significant results for partial reconfiguration process with better saving of power consumption by 3.03%, reduce the processing clock cycles and the reconfiguration overhead by 39.5% and 50% respectively. These achievements motivate a strong justification to further explore the 3-D HWT implementation with both partial and non-partial reconfiguration processes and to evaluate their performance in terms of area, power consumption and maximum speed.

In addition to the previously reviewed work on DPR, there exist two other significant references discussing the advantages and challenges in DPR [117], [118]. Shoa and Shirani [117] thoroughly explain different issues in run-time reconfiguration (RTR) systems, and list the implemented systems which support RTR reconfiguration as well as discussing different applications of, and the improvements achieved by applying RTR. An evaluation of DPR for signal and image processing is presented in [118]. The authors present the advantages and limitation of DPR in professional electronics applications and guidelines to improve its applicability. The advantages addressed are as follows: task speed, power reduction, survivability, mission, environment and adaptive algorithm change, online system as well as hardware virtualisation. Furthermore, the missing elements of the design flow to use in DPR are identified and explained.

As the demand of resources utilisation as well as maintaining the outcomes analysis in any highly-demanding intensive application, DPR has been taken into consideration as a promising method in 3-D medical image compression, and it will be described in detail in Chapter 3.

2.5 Limitation of Existing Work and Research Opportunities

As can be seen from the preceding sections, there still remains a huge gap for further research in exploiting reconfigurable computing for 3-D medical image compression.

Three major limitations of the existing work can be identified as follows:

1. Medical image compression has not been intensively addressed in the existing 3-D DWT implementation. The Daubechies filter has been extensively used in several implementations [9], [10], [66], [67], whilst Haar filter remains open for further experimentation. Since the 3-D medical image compression introduces high computational demands, DPR that offers better hardware configuration is the best option to be considered. To the best of knowledge, there are no research publications available on DPR utilisation for medical imaging applications;

2. Although impressive image processing performance has been achieved with ridgelet, curvelet and Radon transforms, the complexity of their implementation still remains as a heavy burden on standard microprocessors, where large amounts of data needs to be processed. Surveying the literature, very limited FPGA-based implementation has been found, and interestingly, there is no discussion reported on medical image de-noising hardware implementation using the FRAT. Therefore, the design and implementation of the FRAT as a basic building block of ridgelet and curvelet transform, are strategic and very promising opportunity; and

3. Image compression is one of the well establish research area. However, medical image compression especially dealing with 3-D modalities is considered as a pre-mature research area. Moreover, even there is more and more new compression method have been proposed as well as the algorithm optimisation, very limited hardware implementation of 3-D medical image compression is discovered.

Furthermore, utilisation of CAVLC in 3-D medical image compression is also a convincing area.

Based on the existing work limitations, the objectives of the work presented in this thesis can be summarised as follows:

1. Designing and implementing of efficient and novel architectures for 3-D HWT using different design methodology for 3-D medical imaging application;
    - Evaluating the performance achievement of DPR methods for 3-D medical image processing applications;
    - Comparing the performance of partial and non-partial implementation of 3-D HWT for 3-D medical volumes; and
    - Investigating the influence of 3-D medical volumes transform size on the hardware performance, such as area, maximum frequency and power consumption.
2. Designing and implementing of efficient and novel architectures for the FRAT implementation, specifically for medical image de-noising;
    - Analysing the design trade-off for three implementation strategies on the hardware performance in terms of throughput, area and maximum frequency;
    - Evaluating the capability of the FRAT for medical image de-noising in pre-processing stage of the complete medical image compression system;
    - Evaluating the impact of block sizes on the PSNR values in medical images; and
    - Comparing the performance of software simulation and hardware implementation for various medical image modalities, and examining the capability of rapid prototyping using the Xilinx AccelDSP tool.

3. Designing and implementing of an efficient 3-D medical image compression system using CAVLC;

    - Evaluating the performance of IT and DWT for the transform block in the 3-D compression system;
    - Evaluating different processed medical image modalities after software simulation and hardware implementation; and
    - Comparing the performance of the transform block and the proposed CAVLC architecture implementation in terms of area, power consumption and maximum frequency.

2.6 Summary

This chapter reviews a number of significant architectures and systems for two main research problems: reconfigurable architectures and 3-D medical image compression. In addition, the advantages and drawbacks of the existing architectures have been also highlighted as well as limitations of the existing work were addressed. It is the ultimate aim of the research work presented in this thesis, to address the limitations presented in the previous section with an efficient means of providing efficient reconfigurable architectures for 3-D medical image compression.

Chapter 3

Efficient Architectures for 3-D HWT using DPR

3.1 Overview

The application of three-dimensional (3-D) real-time medical image processing uses several building blocks for its computationally intensive algorithms to perform matrix transformation operations. Moreover, complexity in addressing and accessing large medical volumes data and massive amount of data to be processed have resulted in vast challenges from a hardware implementation point of view. In order to cope with these issues, field programmable gate arrays (FPGAs) with efficient reconfigurability mechanisms should be employed to meet the requirements of these applications in terms of maximum speed, size, power consumption and throughput.

Dynamic partial reconfiguration (DPR) is a promising mechanism for reducing the hardware required for implementing an efficient design for 3-D medical image processing application as well as improving the performance, speed and power consumption of the system. With this technique, the design can be divided into sub-designs that fit into the available hardware resources and can be uploaded into the reconfigurable hardware when needed. As the first configuration finishes its operations,

the next configuration can be uploaded to the hardware [106]. In short, by timesharing, large designs can be implemented on limited hardware resources.

The work presented in this chapter is concerned with an efficient architecture for 3-D Haar wavelet transform (HWT) with transpose-based computation using DPR technique. The proposed architectures will be deployed in a reconfigurable environment for adaptive medical image compression. An in depth evaluation of these architectures in terms of area, power consumption and maximum frequency is also carried out. Influence of the transform size on the hardware performance for both proposed architectures is addressed [50].

This chapter begins with an overview of the mathematical background for HWT, matrix transposition and pipelined direct mapping implementation in Section 3.2. Section 3.3 exposes the proposed architecture of 3-D HWT with DPR mechanism. Experimental results, comparison and analysis are presented in Section 3.4. Finally, a brief summary is given in Section 3.5.

3.2 Mathematical Background and Design Methodology

Mathematical background and design methodology for the implementation of 3-D HWT are presented in the following subsections.

3.2.1 3-D Haar Wavelet Transform (HWT) and Matrix Transposition

The wavelet transform is a mathematical tool with a great variety of possible applications. As the simplest wavelet transform, the Haar wavelet ψ_{haar}, is discontinuous, symmetric wavelet in the Daubechies family, and the only one has an explicit expression. The scale function ϕ_{haar}, is a simple average function. The

wavelet and the scale functions can be expressed as follows:

$$\psi_{haar}(t) = \begin{cases} 1 & \text{if} \quad 0 \leq t < \frac{1}{2} \\ -1 & \text{if} \quad \frac{1}{2} \leq t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

$$\phi_{haar}(t) = \begin{cases} 1 & \forall \quad 0 \leq t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

In addition, the HWT wavelet is simple and computationally cheap because it can be implemented by a few integer additions, subtractions, and shift operations [119]. This wavelet was selected because of its simplistic nature, and mathematical features. The mathematical features of the basis are as follows: the most simplistic wavelet basis, can be implemented using pairwise averaging and differencing, both unitary and orthogonal, and also it has compact support. Therefore, this wavelet basis is by no means the most suitable to achieve close to optimal compression performance in 3-D medical image compression application.

$$\begin{aligned}
c_{LLL} &= (c_{000} + c_{001} + c_{010} + c_{011} + c_{100} + c_{101} + c_{110} + c_{111})/8 \\
c_{LLH} &= (c_{000} + c_{001} + c_{010} + c_{011} - c_{100} - c_{101} - c_{110} - c_{111})/8 \\
c_{LHL} &= (c_{000} + c_{001} - c_{010} - c_{011} + c_{100} + c_{101} - c_{110} - c_{111})/8 \\
c_{LHH} &= (c_{000} + c_{001} - c_{010} - c_{011} - c_{100} - c_{101} + c_{110} + c_{111})/8 \\
c_{HLL} &= (c_{000} - c_{001} + c_{010} - c_{011} + c_{100} - c_{101} + c_{110} - c_{111})/8 \\
c_{HLH} &= (c_{000} - c_{001} + c_{010} - c_{011} - c_{100} + c_{101} - c_{110} + c_{111})/8 \\
c_{HHL} &= (c_{000} - c_{001} - c_{010} + c_{011} + c_{100} - c_{101} - c_{110} + c_{111})/8 \\
c_{HHH} &= (c_{000} - c_{001} - c_{010} + c_{011} - c_{100} + c_{101} + c_{110} - c_{111})/8
\end{aligned}$$

Figure 3.1: 3-D HWT expression.

The one-dimensional (1-D) HWT can be extended naturally into higher dimensions simply by taking tensor products of 1-D filters [120]. This approach is fundamentally based on the partitioning concept of the image into small blocks containing eight voxels. Consider a block with dimensions of $2 \times 2 \times 2$, and are labeled as $c_{i,j,k}$, $0 \leq i, j, k \leq 1$. In this case, "0" represents low-pass filtering and "1" for high-pass filtering. The Haar transform in 3-D is expressed as in Figure 3.1 and the

decomposition of this small block is expressed with c_{LLL} represents the overall average or successive low-pass filtering coefficient, and the remaining seven values are detail, or wavelet coefficients, determined by the order of filtering.

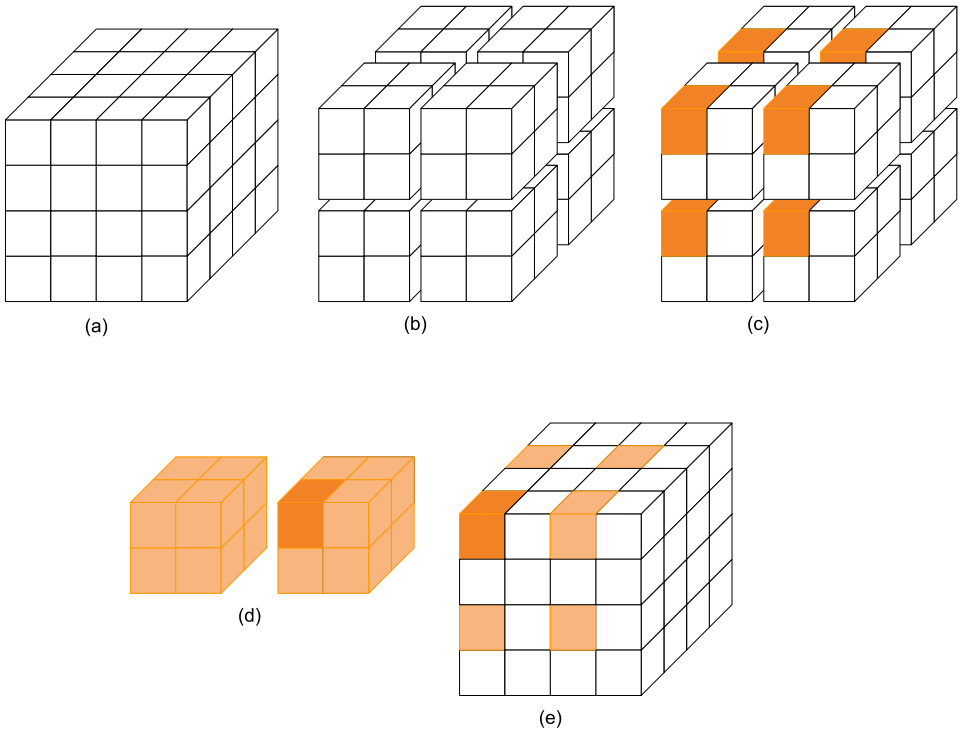

Figure 3.2: Decomposition based on tensor product of 1-D filters (a) Original image volume (b) Image volume partitioned into $2 \times 2 \times 2$ sub-blocks (c) One overall low-pass coefficient is obtained from each sub-block after the first decomposition stage (d) All sub-block averaging coefficients are clustered to form new sub-blocks, which are then decomposed further to obtain one overall low-pass coefficient (e) Image after two stage decomposition on a $4 \times 4 \times 4$ image volume.

As an example, the coefficient of c_{LHL} is obtained by applying the low-pass filter L, high-pass filter H and then the low-pass filter L, along the three principle axes, respectively [119]. In a more realistic example, taking an image with dimension of $8 \times 8 \times 8$, the image volume is partitioned into a series of sub-blocks as defined previously. After the decomposition of each of these sub-blocks the 64 overall averaging coefficients are combined to form eight new $2 \times 2 \times 2$ sub-blocks and the next level in the decomposition hierarchy. The complete decomposition end when only one overall averaging coefficient and five hundred and eleven detail coefficients remain. For an $n \times n \times n$ image volume, $\log_2 n$ iterations are required to perform a complete

decomposition and an illustration of the method is presented in Figure 3.2(a) – (e).

The implementation of the proposed 3-D HWT architecture requires three 1-D HWT and two transpose modules in between. Matrix transposition can be viewed as changing the storage of matrix elements from the row-major order to the column-major order, or vice versa. Mathematically, it can be expressed by Equation 3.3 for all $i, j = 0...N-1$, where a_{ij} is the coefficient of the matrix. An example is shown in Figure 3.3.

$$a_{ij} = a_{ji} \tag{3.3}$$

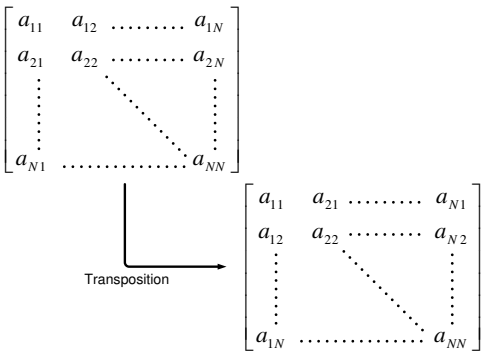

Figure 3.3: Transposition of a matrix.

3.2.2 Pipelined Direct Mapping Implementation

The 1-D HWT flow diagram with N-inputs sample for pipelined direct mapping implementation is shown in Figure 3.4 with "Avg." and "Diff." notation refer for average and differencing, respectively. The registers, adders and difference blocks are triggered at the rising edge of the clock. At the first clock edge, the data are loaded and on the second edge they will be retrieved from the registers to the adder and the difference blocks. At the third clock edge, the adder and the difference blocks perform their respective operations as described in Equation 3.4 and 3.5, where $i = 0...\left(\frac{N}{2} - 1\right)$.

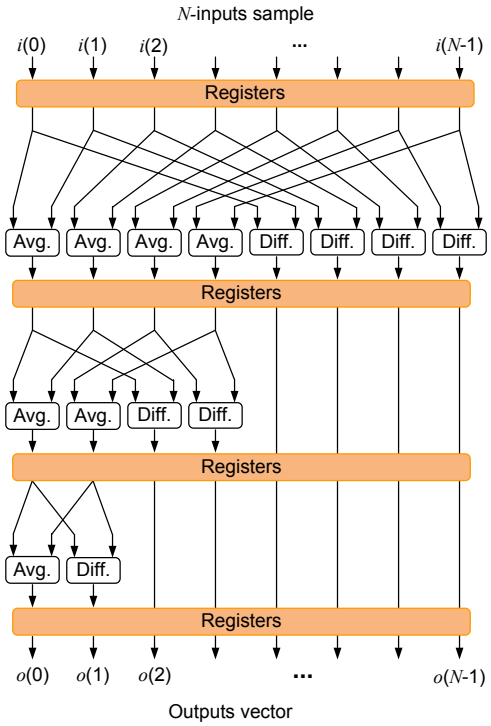

Figure 3.4: 1-D HWT flow diagram with N-inputs sample for direct mapped architecture.

$$H_i = \left(\frac{a_{2*i} + a_{2*i+1}}{2} \right) \tag{3.4}$$

$$H_{\left(\frac{N}{2}+i\right)} = (a_{2*i} - a_{2*i+1}) \tag{3.5}$$

3.3 Proposed Architectures

An overview of the proposed system applications and an explanation for both 3-D HWT architectures are discussed in the following subsections.

3.3.1 Proposed System Applications

Figure 3.5(a) illustrates an application overview of the proposed medical image compression system including the transform, quantisation and entropy coding blocks.

In each block, buffers have been used for storing intermediate results to be processed. Since the application targeted for 3-D medical images, the transform block in this system including the 3-D HWT. The DPR is used for two reasons. The first, is to have the possibility to exchange the type of algorithm (Haar, Daubechies, biorthogonal 9/7 ... etc.) thus making the architecture flexible. The second is to find the best trade-offs between area, power consumption and hardware performance.

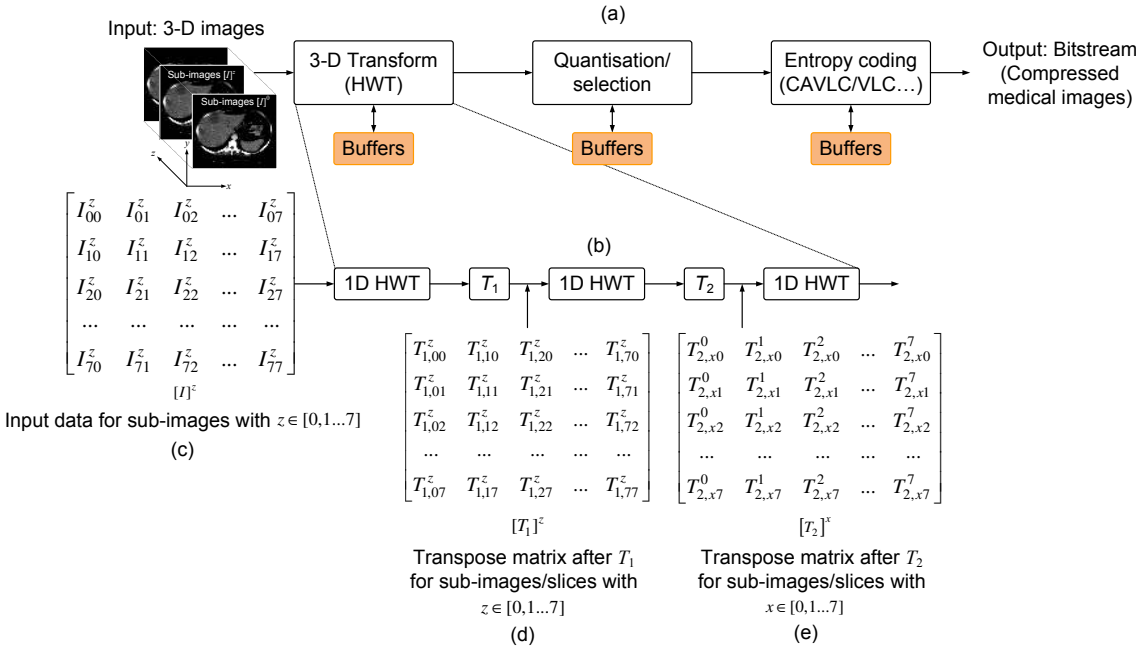

Figure 3.5: Proposed system architectures (a) Compression system overview (b) Architecture for 3-D HWT with transpose-based computation (c) Input data for sub-images for $[I]^z$ (d) Transpose matrix after T_1 (e) Transpose matrix after T_2.

3.3.2 3-D Haar Wavelet Transform (HWT) with Transpose-based Computation

System Architecture

The proposed system for 3-D HWT with transpose-based computation is illustrated in Figure 3.5(b). The whole chain to calculate the 3-D HWT gets an input as a 3-D image with $N \times N \times N$ point and outputs the coefficients of the $N \times N \times N$ point. To

simplify the hardware design, the 3-D HWT is split into three 1-D HWT calculation cascaded together with transpose modules in between. This is achieved by performing the first 1-D HWT along the rows (columns) of the array followed by 1-D HWT along the columns (rows) of the transformed array. The third 1-D HWT is performed on corresponding pixels in each of the N sub-images that constitute the third dimension. All transpositions modules with two memory banks is used to store the transposed coefficients into memory with a fetch unit module that reads back the coefficients for the next 1-D HWT calculation.

3-D Haar Wavelet Transform (HWT) and Transpose

The proposed 3-D HWT implementation works as follows. The input to the first 1-D HWT is read row by row, the 1-D HWT is performed on each input vector as they are provided and the calculated values are sent to the transpose module T_1 which calculated the memory addresses for the transposition and stores the data into memory. The transpose T_1 acts as a memory forwarder and performs matrix transpose, since row vectors are provided by the 1-D HWT. After transposition of the resultant matrix, another 1-D HWT is performed on the coefficients which are stored in memory to yield the two-dimensional (2-D) HWT coefficients.

This is the conventional row-column 2-D HWT computation. The 2-D HWT computation is performed on each sub-image $[I]^z$ for $z \in [0, 1...7]$, where $[I]^0$ is the first sub-image and $[I]^7$ is the eighth sub-image of the input volume as shown in Figure 3.5(c) with notation of I^z_{xy}. The output coefficients of the 2-D DWT are send to the second transpose, T_2. As described before, all coefficients are stored into memory and the transpositions of T_2 are stored after transformation into memory. As shown in Figure 3.5(d) and (e), $T^z_{1,xy}$ and $T^z_{2,xy}$ represents the coefficients after performing transposition T_1 and T_2 respectively. As an example, the HWT computations for the first transpose are $T^z_{1,00} = \frac{I^z_{00}+I^z_{01}}{2}$ for averaging and $T^z_{1,04} = I^z_{00} - I^z_{01}$ for differencing, and these two operations are based on Equation 3.4 and 3.5. The Algorithm 3.1 gives the complete description of the 3-D HWT process.

Algorithm 3.1 The 3-D HWT pseudo-code

```
1: for slice = 1 to noslices do
2:     for row = 1 to norows do
3:         Apply a 1-D HWT column-wise
4:     end for
5: end for
6: for slice = 1 to noslices do
7:     for col = 1 to nocols do
8:         Apply a 1-D HWT row-wise
9:     end for
10: end for
11: for col = 1 to nocols do
12:     for row = 1 to norows do
13:         Apply a 1-D HWT slice-wise
14:     end for
15: end for
```

The Haar function is able to calculate a given vector with the cycle rate of $\frac{N}{2}+4$ cycles (eight cycles for $N = 8$) as shown in Equation 3.6. Each transposition clock cycle as in Equation 3.7 with memory writes is able to operate with N cycles (eight cycles for $N = 8$). A sub-image is calculated in one clock cycle as shown in Equation 3.8 and the required 3-D image calculation clock cycle rate is given by Equation 3.9.

$$F_{c_h} = \frac{N}{2} + 4 \tag{3.6}$$

$$F_{c_t} = N \tag{3.7}$$

$$F_{c_{subimg}} = N * [6 * (F_{c_h} + F_{c_t})] \tag{3.8}$$

$$F_{c_{img}} = N * F_{c_{subimg}} \tag{3.9}$$

3.3.3 3-D Haar Wavelet Transform (HWT) with Dynamic Partial Reconfiguration (DPR)

Proposed Architecture

Figure 3.6 illustrates the proposed idea of reconfigurable and adaptive system architectures using DPR. It composes of several reconfigurable processing modules (RPM), a reconfigurable interface, an off-chip memory and micro blaze (μblaze) and the system connected to the host personal computer (PC) via peripheral component interconnect (PCI) express. μblaze is a soft processor core designed for Xilinx FPGAs [116]. The reconfigurable processing modules allow hardware acceleration and can be reconfigured based on the system demand, whilst the communication interface is used to build the interconnection between RPM and the other components.

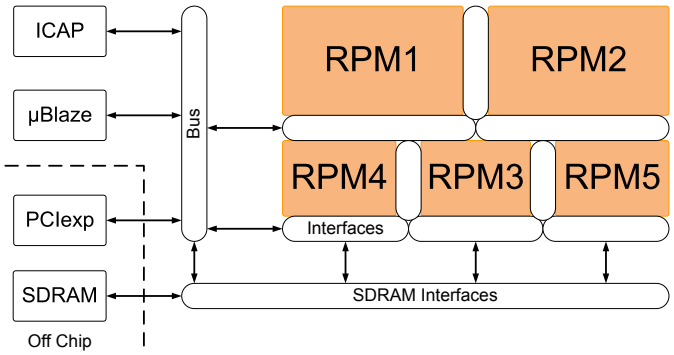

Figure 3.6: Proposed reconfigurable and adaptive system architectures.

Both proposed architecture implementation on the FPGA are given in Figure 3.7(a) and (b). The DPR framework consists of two reconfigurable areas and a static area. One DPR area is used for the 1-D HWT and the second is used for the different transposition modules. The 1-D HWT DPR area is directly connected to the transpose DPR area. The static area consists of the data fetch unit and the memory controller (Wishbone compliant). On the other hand, the implementation of 3-D HWT without DPR defined the entire FPGA devices as one module as oppose to the implementation with DPR mechanism.

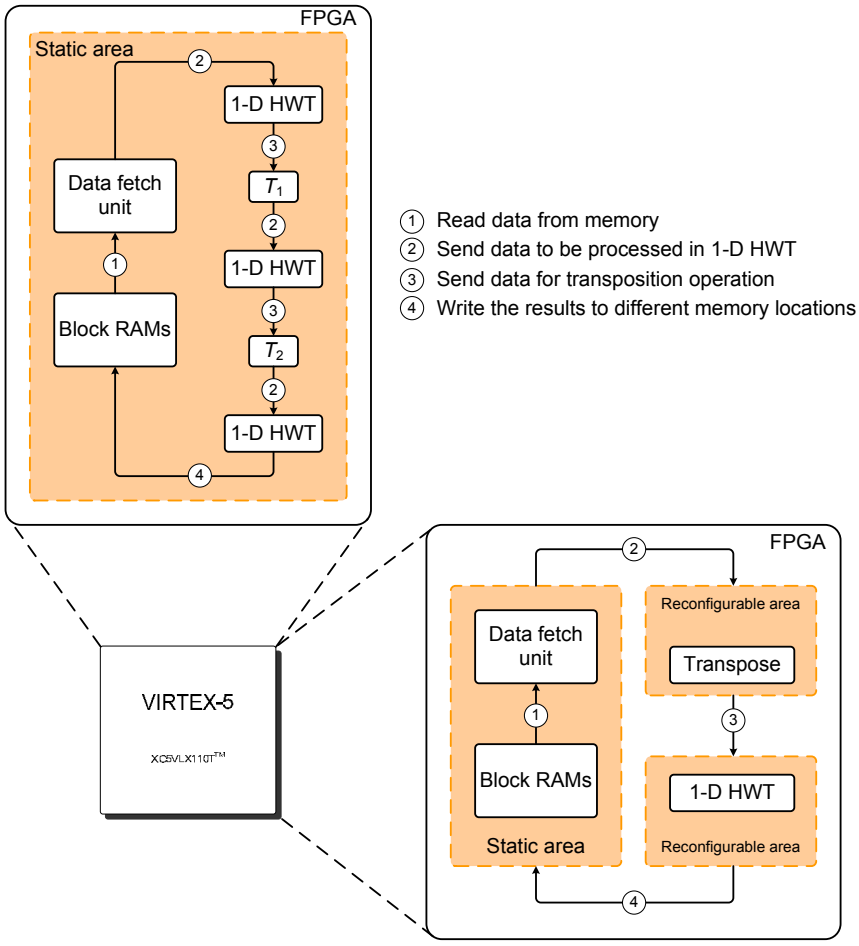

Figure 3.7: Proposed top architecture of 3-D HWT (a) Without DPR (b) With DPR.

Reconfigurable and Static Area

The proposed system is implemented with the partial reconfiguration suite from Xilinx (ISE 9.2PR and PlanAhead 10.1) [116]. It uses the module-based DPR where configuration frames are reconfigured and bus macros are used to connect the DPR areas with the static area [121]. The partial reconfiguration design flow is illustrated in Figure 3.8(a) and (b) for all the steps required and modules definition illustration.

This methodology has the restriction that all design files and reconfigurable modules must be available to the build environment to build partial modules. The main advantage of DPR is that an implementation of a given design can be integrated into a smaller FPGA. This reduces cost, package size and power. Also, power consumption

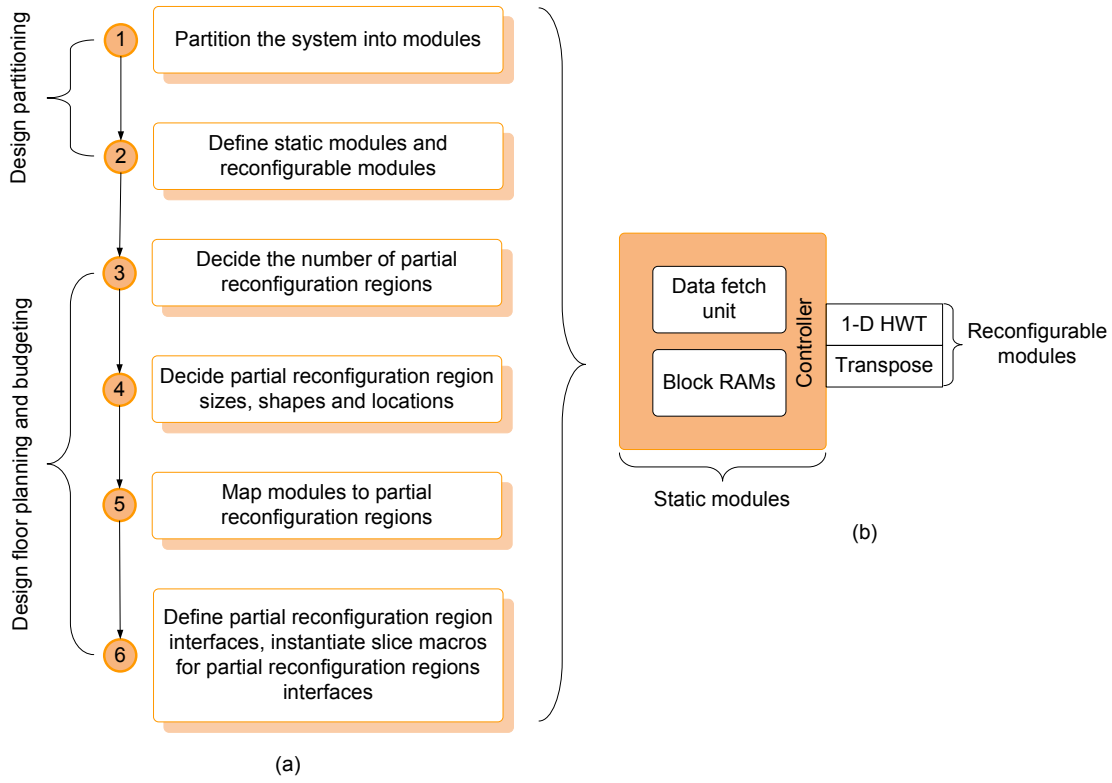

Figure 3.8: Partial reconfiguration design flow (a) Steps for partial design flow (b) Define static and reconfigurable modules.

and logic size can be reduced by cascading operation/calculation modules.

In the 3-D HWT case, the transposition module and the 1-D HWT module can be changed. The transposition module will be changed during image calculation three times for each sub-image. First, transposition T_1 performs the row to column transposition which are active till a sub-image is transposed. After the T_1 sub-image transposition, the DPR area is reconfigured with the T_2 transposition which saves the sub-images. This described operation will be repeated for all sub-images. After all sub-images are computed and transposed with T_2, the transposition DPR is reconfigured with the straight transposition and the last 1-D HWT is performed on all T_2 sub-images. The HWT DPR area can be reconfigured to switch between different pixel sizes. The pixel size N dependency is propagated from the HWT module to all connected modules, this gives the advantage that no other logic changes are necessary.

The DPR module connections are performed with simple bus interfaces. Data fetch unit and HWT DPR area are connected with a defined data bit width bus, a request line and back signal free. The fetch unit sends data and the request to the HWT core as long the free signal is active. HWT and transposition module are connected with the defined data bit width bus and an enable signal. Each cycle where the enable is active data will be transposed and written into the memory.

3.4 Experimental Results and Analysis

FPGA implementation and further discussion of the results are given in the following subsections.

3.4.1 Field Programmable Gate Array (FPGA) Implementation

Xilinx early access partial reconfiguration (EAPR) design flow [121] has been used as a design flow reference and the proposed two architectures have been implemented on the Xilinx Virtex-5 (XC5VLX110T-3FF1136). Table 3.1 lists the overall performance results for both proposed architectures for $N = 8$ and 128. The implementation of 3-D HWT with DPR mechanism provides significant results with better saving of area and reduce the power consumption by 47.07% and 9.77%, respectively. In terms of maximum frequency, DPR mechanism yielding 51.10% better maximum frequency than without DPR.

Table 3.2 lists a comparison between non-partial and partial reconfigurations scenarios. In the case of non-partial reconfiguration, a full bitstream need to be generated and stored for each 3-D HWT configuration. A full bitstream of 3,889,941 bytes is required for 3-D HWT configuration and the shortest configuration time needed is also the worst at 4.8 ms. On the contrary, a full partial bitstreams generated are significantly smaller and hence reducing the storage space required to

Table 3.1: Resources utilisation and overall proposed architectures performance on XC5VLX110T-3FF113.

Parameters	**Proposed 3-D HWT**			
	Without DPR		**With DPR**	
	$N = 8$	$N = 128$	$N = 8$	$N = 128$
Area (slices)	3,180 (4.60%)	39,261 (56.80%)	1,882 (2.72%)	20,779 (30.06%)
Power consumption (mW)	1102.64	1872.83	1094.48	1689.84
Maximum frequency (MHz)	206.00	170.13	371.15	347.92

Table 3.2: Comparison of bitstream generated and configuration times towards transform sizes.

N **(transform size)**	**Without DPR**		**With DPR**	
	BS (bytes)	**CT (ms)**	**BS (bytes)**	**CT (ms)**
8	3,889,941	4.8	191,024	0.23
16	3,889,941	4.8	287,573	0.36
32	3,889,941	4.8	313,101	0.39
64	3,889,941	4.8	507,677	0.63

Note:
BS: Bitstream, CT: Configuration time

store the various bitstreams. The results show that the file size of transform size ($N = 64$) for full partial bitstreams is reduced about 86.95% of a full bitstream and the configuration time also reduced by 86.88%. In summary, by comparing the file sizes of the bitstreams, it is obvious that partial reconfiguration has more efficient bitstream and as proven, smaller bitstream decreases the configuration time.

3.4.2 Discussions

Result and Analysis

For the FPGA implementation purposes, there are different transform sizes (N = 8, 16, 32, 64 and 128) have been used to evaluate the relationship of the transform sizes on the area (slices), power consumption (mW) and maximum speed (MHz). The ideas of various transform sizes used also act as a good mechanism based on the large medical volumes data involves in real-time 3-D medical image processing applications.

Influence of transform size on area, power consumption and maximum frequency is depicted in Figure 3.9 – 3.11, respectively. Results indicate that the proposed 3-D HWT with transpose-based computation requires more area to be implemented while by using DPR mechanism, the area saving can be achieved between 40.82% to 47.18% and the relationship of the area is increasingly proportional to the transform sizes. The power consumptions can be estimated by the large areas required by non-partial reconfiguration consumes up to 1872.83 mW for N = 128 and by performing partial reconfiguration it saves power consumption by 0.74% to 9.77%.

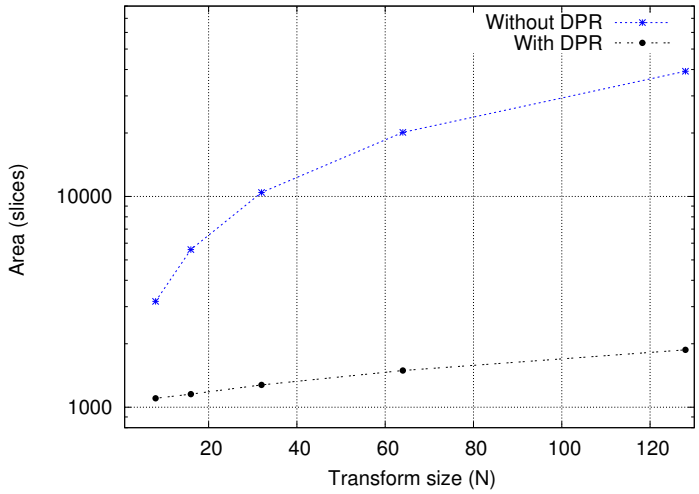

Figure 3.9: Influence of transform size on area.

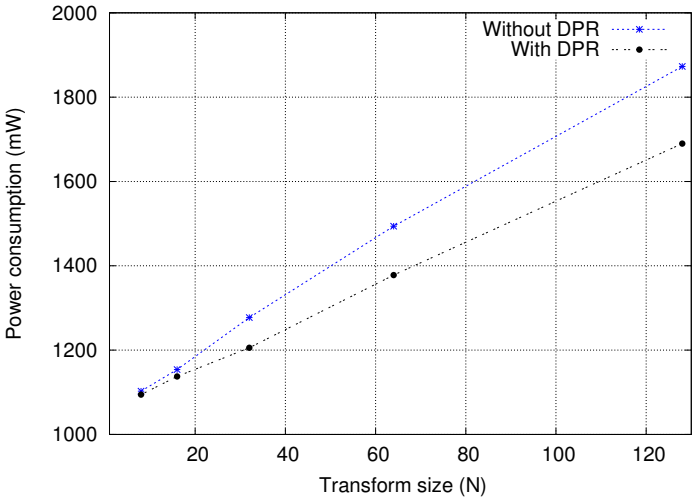

Figure 3.10: Influence of transform size on power consumption.

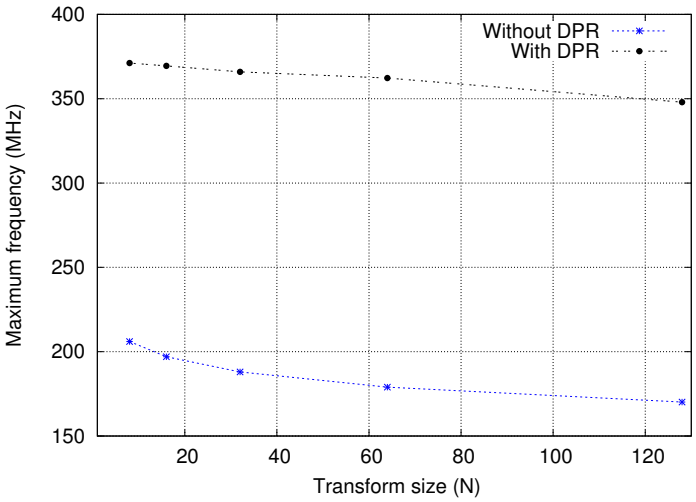

Figure 3.11: Influence of transform size on maximum frequency for 1-D HWT modules.

To evaluate the performance of the proposed architectures in terms of maximum frequency, Figure 3.11 depicted the idea of transform sizes relationship with its maximum achievable operating frequencies and resulting that with DPR, better maximum frequency can be achieved. Since the static build consists of all transpositions and the 1-D HWT function, the design requires more space on the whole FPGA. Moreover, the signals for the different transpositions have to be multiplexed and increases the routing and signal number. This will end in a slower frequency as the DPR design. Furthermore, the partial design can only performed with the slowest

clock speed of the slowest module. In the proposed system the maximum frequency is 356.90 MHz, since the transposition T_2 has the slowest frequency as shown in Figure 3.12.

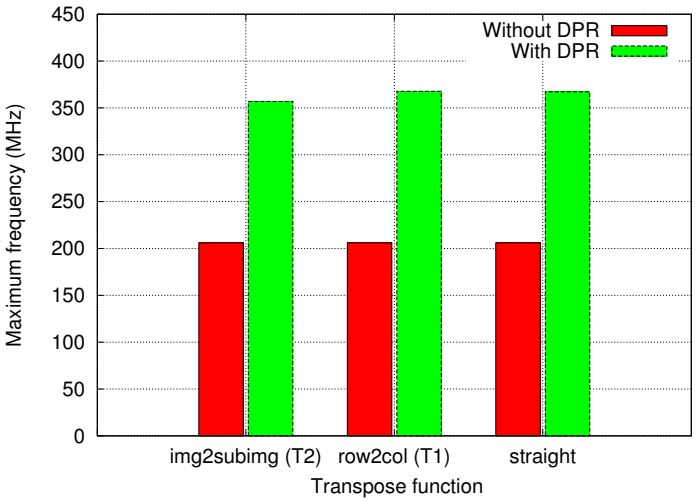

Figure 3.12: Comparison on maximum frequency achievement for transpose function.

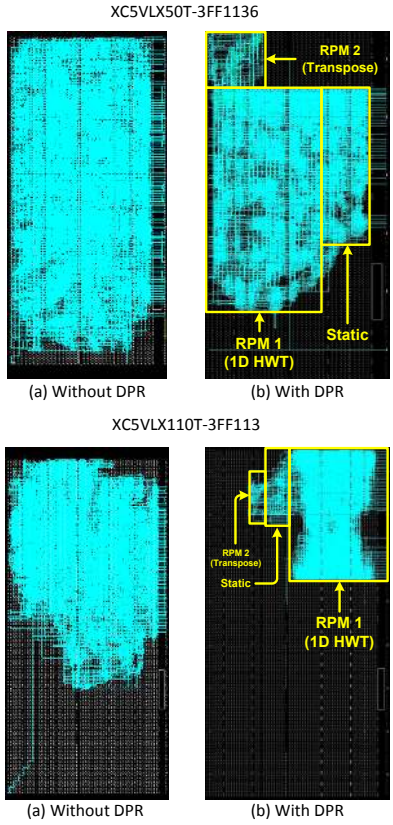

Figure 3.13: Comparison of chip layout for different Virtex-5 devices for $N = 64$.

Moreover, in order to visualise the impact of non-partial and partial reconfiguration for the proposed architecture, chip layouts on different FPGA devices of Virtex-5 are shown in Figure 3.13. With DPR mechanism, the area for static and reconfigurable area can be specified and can be clearly seen in the layouts generated. Comparative study for both non-partial and partial reconfiguration processes shows a significant conclusion concerning the advantages offered by DPR especially in processing large medical volumes. Analysis for the performance achieved for different parameters such as area utilised, power consumes and maximum frequency achieved clearly reveals that with DPR, complex designs can be implemented on limited hardware resources and hence lead to better performance achievements.

Advantages of Partial Reconfiguration

Non-partial reconfiguration can be defined as an arrangement process of all utilised resources on the FPGA and it offers full-device reconfiguration. With this approach, any design errors can be resolved by refining the bitstream of the design. An FPGA with full-device configuration allows the chip to be configured by specific design at one time and at another time the chip is configured with another design.

From hardware resources point of view, reconfiguration mechanism offers utilisation of the total number of resources on the FPGA by loading each design separately, while without reconfiguration both designs are loaded together. Therefore, the total of resources usage cannot exceed the available number of resources available on the FPGA.

Indeed non-partial reconfiguration suffers for limited hardware resources issue, partial reconfiguration provides all the advantages that have been offered by full-device reconfiguration with two additional advantages concerning its execution process and bitstreams size generated. Since the unchanged portion of the FPGA is not affected, smaller bitstream file size generated than a full bitstream, application of 3-D real-time medical imaging with requirement of its modules to continuously operates

can be implemented on the same chip as modules. With the size of the bitstream is directly proportional to the number of resources that have been configured, partial reconfiguration contributes for less space required in operation configurations.

In terms of configuration time, there are four major phases in the reconfiguration process: clearing configuration memory, initialisation, bitstream loading and device startup. As complex as the phases in configuration processes may seem, a smaller bitstream sizes resulting a shorter configuration time.

In the proposed 3-D HWT architecture with transpose-based computation, the implementation of 1-D HWT and transpose treat the entire Virtex-5 as one module as oppose to the second architecture with DPR mechanism. As shown in Table 3.3, results obtained show that the proposed architecture can be implemented on smaller Xilinx FPGA devices. With DPR, number of slices registers used is significantly reduced. As the implementation without DPR used the whole devices as one module, the proposed design and implementation suffers from the overmapped problem.

Table 3.3: Device summary report of the proposed architecture on XC5VLX30T-3FF323.

Parameters	**Proposed 3-D HWT**	
	Without DPR	**With DPR**
Number of slices registers	27,886 (145%) (Overmapped)	Haar static = 178 (1%) Haar partial = 14,469 (75%) Tr. img2subimg (T_2) = 267 (1%) Tr. row2col (T_1) = 235 (1%)

3.5 Summary

Two architectures for 3-D HWT have been proposed in this chapter based on transpose computation and partial reconfiguration. Comparative study for both non-partial and

partial reconfiguration processes has shown that DPR offers many advantages and lead to a promising solution for implementing computationally intensive applications such as 3-D medical image compression. Using DPR, several large systems are mapped to small hardware resources and the area, power and maximum frequency are optimised and improved.

Chapter 4

FPGA-based Architectures of FRAT for Medical Image De-noising

4.1 Overview

In general, an image may be degraded by various types of noise that arise during image acquisition or transmission and the most common type of noise is the additive one [122]. Noise reduction is very important, as various types of noise is generated by medical imaging equipment, consequently, limits the effectiveness of medical image diagnosis [80], [82].

Objects and curves are the main components in medical images. To produce better analysis of medical images, curvelet as well as ridgelet are the two main transforms that have been widely used. Moreover, the basic building block of these two transforms is finite Radon transform (FRAT) as shown in Figure 4.1. As an example for the curvelet transform, Figure 4.1(b) illustrates the decomposition of the original image into sub-bands, followed by the spatial partitioning of each sub-band.

The ridgelet transform is then applied to each block [19].

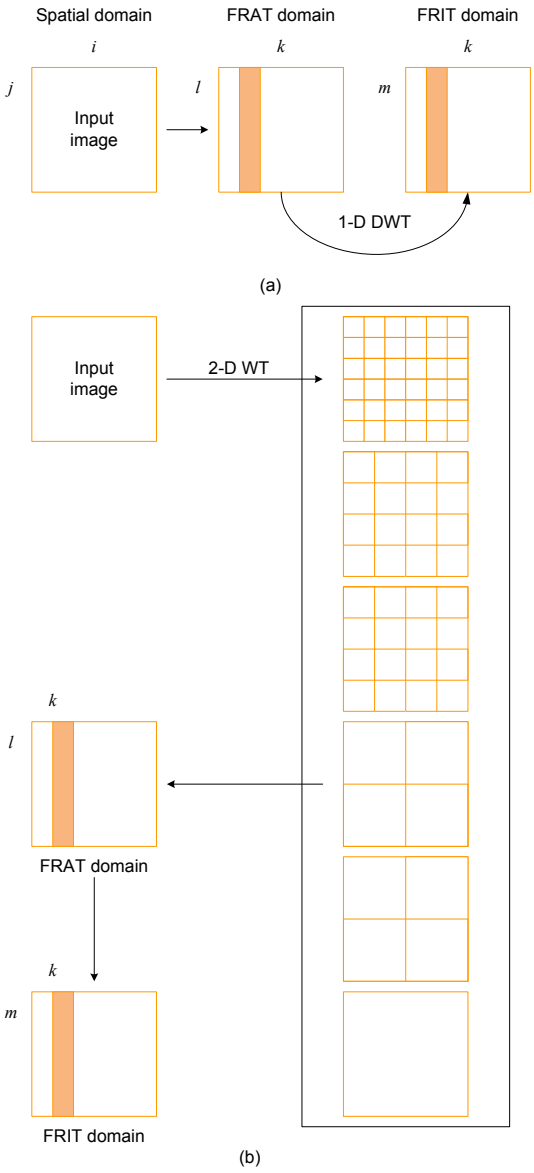

Figure 4.1: Transform flow graph (a) Ridgelet transform (b) Curvelet transform.

Curvelet, ridgelet and the FRAT emerge as a compelling aid to defeat the wavelet's limitation such as inflexible directionality. However, the FRAT algorithm is demanding, as it is inherently serial, iterative and has long latency. To overcome these restrictions, hardware implementation of FRAT is an interesting domain of study, especially for medical imaging application. Existing limited hardware implementation of the FRAT [71–73], [75], [76] in medical imaging application opens a huge gap to be filled.

This chapter presents the design and implementation of FRAT on field programmable gate array (FPGA) for medical image de-noising using Xilinx AccelDSP tool. The design and implementation of the FRAT is carried out in the pre-processing stage form image de-noising, before the complete implementation of the three-dimensional (3-D) compression systems.

Three design strategies have been proposed: direct implementation of pseudo-code with a sequential and pipelined description, and block random access memory (BRAM)-based method. Analysis for both software simulation and hardware implementation with different medical image modalities has been carried out and discussed. An evaluation of FRAT's capability on medical image de-noising is also addressed.

The organisation of this chapter is as follows. An overview of the algorithms used and the design flow for the methodology are presented in Section 4.2. Section 4.3 explains the proposed system implementation in two aspects: system applications and architectures. Experimental results, comparison and analysis for medical image de-noising, software simulation and hardware implementation are explained in Section 4.4. Finally, summary is given in Section 4.5.

4.2 Mathematical Background and Design Methodology

Mathematical background and design methodology for the implementation of FRAT using Xilinx AccelDSP are explained in the following subsections.

4.2.1 Radon Transform (RT)

Mathematically, the Radon transform (RT) in two dimensions is the integral transform comprising of a function over straight lines. If a straight line is represented

parametrically by Equation 4.1,

$$s = x\cos\theta + y\sin\theta \tag{4.1}$$

where s is the shortest distance from the straight line to the origin, and θ is the angle which the line makes with the y-axis. Pictorially, it can be also illustrated as shown in Figure 4.2.

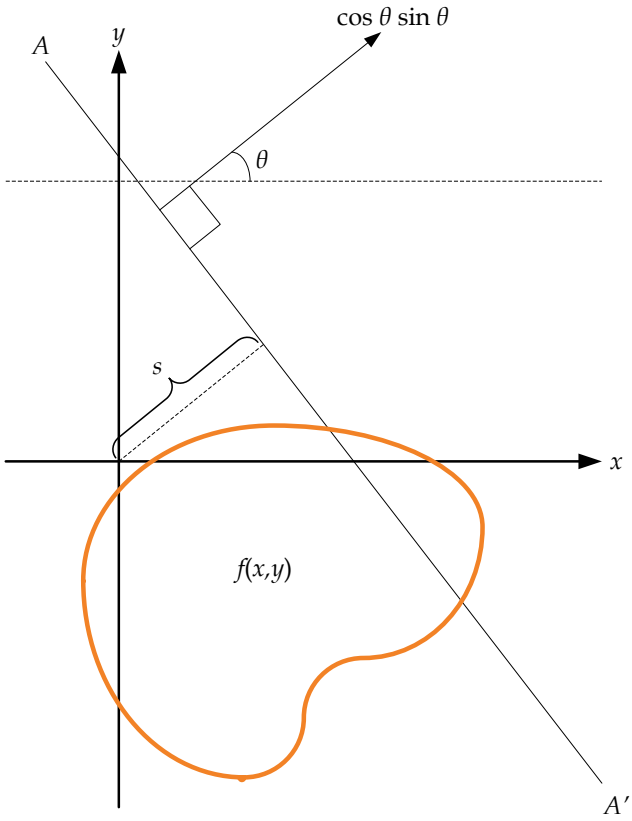

Figure 4.2: Radon transform representation.

The Radon transform function $R[f](\theta, s)$ can be given as follows:

$$R[f](\theta, s) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(x\cos\theta + y\sin\theta - s)dxdy \tag{4.2}$$

where δ is the Dirac impulse function with the following properties.

$$\delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases} \tag{4.3}$$

Based upon Equation 4.1 and 4.3, then the inverse Radon transform (IRT) is given in Equation 4.4, which is used for image reconstruction.

$$f(x,y) = \int_0^{2\pi} R[f](\alpha, x\cos\alpha + y\sin\alpha)d\alpha \tag{4.4}$$

4.2.2 Finite Radon Transform (FRAT)

The FRAT is defined as summations of image pixels over a certain set of lines. Those lines are defined in a finite geometry in a similar way to the lines for the continuous RT in the Euclidean geometry.

Consider a cyclic group Z_p denoted by $Z_p = (0, 1, ..., p-1)$ such that p is a prime number. Let the finite grid Z_p^2 be defined as the Cartesian product of $Z_p \times Z_p$. This finite grid has $(p+1)$ non-trivial sub-groups, given by:

$$L_{k,l} = \{(i,j) : j = (ki + l)(\mathrm{mod} p), i \in Z_p\}, \quad k < p \tag{4.5a}$$

$$L_{p,l} = \{(l,j) : j \in Z_p\} \tag{4.5b}$$

where each sub-group $L_{k,l}$, is the set of points that define a line on the lattice Z_p. The Radon projection of the function f on the finite grid Z_p^2 is given by:

$$r_k[l] = FRAT_f(k,l) = \frac{1}{\sqrt{p}} \left(\sum_{(i,j)\in L_{k,l}} f(i,j) \right) \tag{4.6}$$

From Equations 4.5 and 4.6, it can be seen that the function f is treated as a periodic function. Therefore, the digital representation of the line displays a “wrap

around" effect. Analogous to the continuous case, as in Euclidian geometry, any two lines intersect at only one point in the finite grid Z_p^2. Hence, the inverse transform, the finite back projection (FBP) is given by:

$$FBP_r(i,j) = \frac{1}{\sqrt{p}} \left(\sum_{(k,l) \in P_{i,j}} r_k[l], (i,j) \in Z_p^2 \right) \tag{4.7}$$

where $P_{i,j} = (k,l) : l = (j - ki)(\mathrm{mod} p), k| \in Z_p \cup (p,i)$.

Substituting Equation 4.6 in 4.7, Equation 4.9 proves that the FBP provides a perfect inversion for the FRAT [74]. Also, the algorithm for the FBP and FRAT are synonymous. As a result, the same architecture can be used to implement both the forward and inverse transforms. The computation of FRAT is listed by the pseudo-code given in Algorithm 4.1.

$$FBP_r(i,j) = \frac{1}{p} \left(\sum_{(k,l) \in P_{i,j}} \sum_{(i',j') \in L_{k,l}} f[i',j'] \right) \tag{4.8}$$

$$FBP_r(i,j) = \frac{1}{p} \left(\sum_{(i',j') \in Z^p} f[i',j'] + pf[i,j] \right) = f[i,j] \tag{4.9}$$

4.2.3 Xilinx AccelDSP Design Flow

To ease the process of transforming a MATLAB [123] floating point design into a hardware module, Xilinx introduced the Xilinx AccelDSP software for rapid prototyping of an algorithm in MATLAB into hardware [124]. The main feature of the Xilinx AccelDSP can be summarised as follows:

1. A synthesisable register transfer level (RTL) design can be obtained from the floating point M-code;

2. A set of test-bench can be automatically generated; and

Algorithm 4.1 Pseudo-code for the FRAT

```
1: for k = 0 : (p − 1) do
2:     n = k;
3:     for j = 0 : (p − 1) do
4:         if n < 0 then
5:             n = n + p;
6:         end if
7:         l = n − 1;
8:         for i = 0 : (p − 1) do
9:             l = l + 1;
10:            if l > p then
11:                l = l − p;
12:            end if
13:            FRAT(k, l) = FRAT(k, l) + f(i, j);
14:        end for
15:    end for
16: end for
17: for j = 0 : (p − 1) do
18:    for i = 0 : (p − 1) do
19:        FRAT(p, j) = FRAT(p, j) + f(i, j);
20:    end for
21: end for
```

3. Capability to invoke high description language (HDL) simulation, synthesis and implementation tools.

Verification in each stage is very significant. The Xilinx AccelDSP verifies the generated module in each step to be as true as the previous one, or to be subjectively acceptable with a minor difference during the conversion from floating point design to fixed point [124], [125].

There are two main parts in M-code: a script and function file [124]. In addition, there are three functions of script files. It creates stimuli, feeds the stimuli to the function in a streaming loop and verifies the output from the function. On top of that, the script file also serves as a source file for future test-bench auto generation. Furthermore, the function file comprises the actual function to be translated into HDL, and it is written as an ordinary MATLAB function with an interface of input and output variables.

In this study, the Xilinx AccelDSP has been selected, since it can automatically converted from high-level languages (HLLs) to RTL HDL, and even directly to FPGA

configuration bitstream [125]. This feature is important to reduce the design cycle as well as to allow more optimisation to be carried out on the algorithmic and architectural levels. More details about Xilinx AccelDSP can be found in **Appendix D**.

4.3 Proposed System Implementations

Proposed system applications and an overview of the architecture with the design strategies are described in the following subsections.

4.3.1 Systems Applications

Figure 4.3 illustrates an application overview of our proposed three-dimensional (3-D) compression system including the transform, quantisation and entropy coding blocks with the pre-processing block.

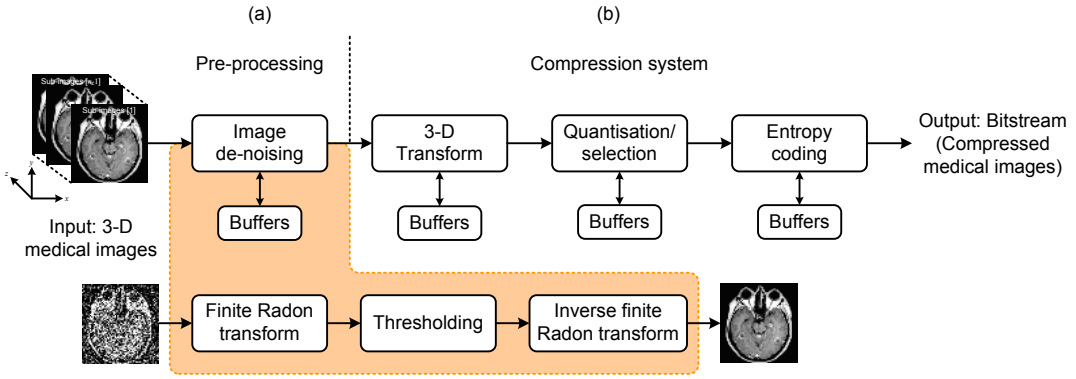

Figure 4.3: Proposed system applications (a) Image de-noising (b) Compression system.

It is well known that noise on medical image resulting in low image quality, and yet limits the diagnostic effectiveness. Therefore, noise reduction for medical images is significantly vital for the pre-processing stage before compression systems as shown as shown in Figure 4.3(a). In the real implementation, the proposed method can be modeled as follows:

$$Y = IRT(RT(X + N)) \tag{4.10}$$

and X, N, and Y are refer to the raw medical image, the noise introduced by equipment and the output image, respectively. From Equation 4.10 and assuming that the RT is linear, it implies that:

$$Y \approx X + IRT(N) \tag{4.11}$$

and the noise in medical image is approximately:

$$IRT(N) \approx Y - X \tag{4.12}$$

Medical acquisition technologies and systems introduce noise and artefacts in the images that should be attenuated by de-noising algorithms [81], likewise, the de-noising process must ensure the anatomical details retains for critical computer-aided analysis of the images. In the medical imaging system, noise can be classified as additive or multiplicative [80], and in this study, additive Gaussian white noise (AGWN) is used to describe noisy signals. Speckle and Poisson noise are two types of multiplicative noise and their variance is not constant but depends on the parameters to be estimated [81].

In this study, three steps are involved in image de-noising as follows:

Step 1

Adding a Gaussian white noise to the image, then applying the FRAT to the noisy image;

Step 2

Calculating the threshold and thresholding the FRAT coefficients with the universal thresholding; and

Step 3

Inverse transform of the thresholded coefficients.

To compute the k^{th} Radon projection for noise removal in FRAT domain, the k^{th} row of the array and all pixels of the original image need to be passed once and use p histogrammers with one for every pixel in the row. At the end, all p histogrammed values are divided by p to get the average values for thresholding process. This averaging process removes the noise in the images.

It is worth noting that the thresholding plays a significant role in the de-noising process [126] and finding an optimum threshold is a tedious process. Noise in medical images can be removed using thresholding approaches by partitioning image intensities. This concept attempts to identify an intensity value that can separate the signal into a desired number of classes. By clustering all pixels with intensities larger than the threshold value into one class and all others into another, image de-noising can be achieved [5]. Universal threshold as proposed by Donoho and Johnstone in [127] is defined as follows:

$$\lambda = \sigma_n \sqrt{\log 2N} \tag{4.13}$$

where N refers to the image size and σ_n is the noise standard deviation. To estimate the standard deviation of AGWN, the median absolute value (MAV) of the coefficients at the finest scale is first calculated and the standard deviation of noise is then estimated [83].

4.3.2 Proposed Architecture and Design Strategies

Figure 4.4 shows the reference architecture for the FRAT based on the FRAT's pseudo-code. To exploit the hardware resources available, the operations of the various counters used to track the addresses of the output vectors are parallelised and pipelined. This has been carried out by changing rollover conditions and count limits suitably.

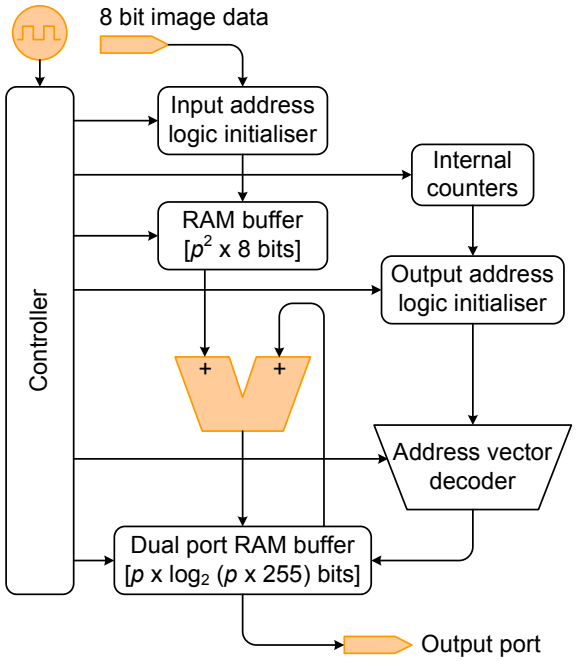

Figure 4.4: Proposed reference architecture for the FRAT.

The number of counters required remains the same, and only the triggering conditions, order and reset logic are modified suitably. It must be highlighted that whilst the algorithm is still serial and cycles through $p \cdot (p+1)$ iterations, the number of steps in the algorithm have been reduced, thereby improving latency. The architecture has serial inputs/outputs (I/Os) and a serial core. The total latency of the core is $O(p^2(p+1))$. The input section consists of a one-dimensional (1-D) random access memory (RAM) of width eight bits and a depth of p^2.

Although each input image block is a square tile of side p, buffering it in a 1-D RAM reduces the computational complexity of the control logic associated with data access. This is because, a two-dimensional (2-D) RAM is implemented on FPGA as a number of 1-D RAMs and uses additional multiplexing logic to de-reference the address locations. The FRAT operation requires reading and writing from the same memory location within a single clock pulse. This is compactly and effectively implemented using a dual ported RAM at the output section instead of an array based buffer. The output buffer is a 1-D dual port RAM of width $log_2(p \cdot 255)$ and depth p. Only a single FRAT vector is buffered and the final values are written to the output

port in serial fashion at the end of each iteration. At the end of $(p+1)$ iterations, the entire image block is transformed to the FRAT domain.

Based on the FRAT architecture, three design strategies have been proposed as shown in Figures 4.5(a) – (c), with "R", "E" and "W" refer to "Read", "Enable" and "Write" processes, respectively.

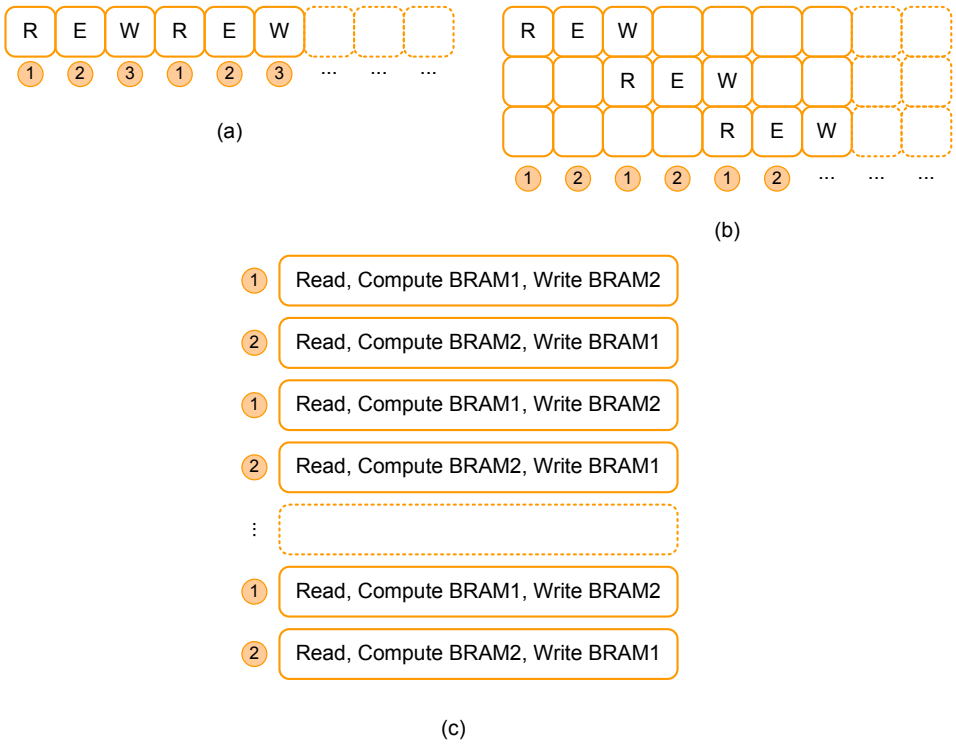

Figure 4.5: Implementation strategies (a) Sequential (b) Pipelined (c) BRAM-based method.

For the sequential fashion, the following modes involve:

Mode 1

Transfer data, a block of $p \times p$ pixels, pixel by pixel;

Mode 2

Compute FRAT; and

Mode 3

Transfer results, a block of $p \times p$ pixels, pixel by pixel.

On the other hand, the pipelined implementation modes are as follows:

Mode 1

Transfer data in pipelined fashion, read a block of $p \times p$ pixels, seven column, column by column (seven pixels at a time) simultaneously write a block of $p + p \times p$ pixels, seven column, column by column (seven pixels at a time); and

Mode 2

Compute FRAT.

In case of BRAM-based method, processes involved can be described as:

Mode 1

Compute the FRAT block in BRAM 1, and write the computed FRAT block in BRAM 2; and

Mode 2

Compute the FRAT block in BRAM 2, and write the computed FRAT block in BRAM 1.

To illustrate the processes involved, Figure 4.6 shows a part of script and function files for the sequential implementation. Detail function operations as well as the generated fixed point report are presented in Figure 4.7. Moreover, to illustrate the Xilinx AccelDSP capability to convert automatically from HLL to RTL HDL, Figure 4.8 presents the project explorer with VHDL files generated.

4.4 Results and Analysis

Three types of images [22] have been used for software simulation and hardware implementation: medical resonance imaging (MRI) scan of human brain 940×940, chest body CT 128×128, and positron emission tomography (PET) scan of normal human brain 109×109. To evaluate the quality of processed images and to demonstrate

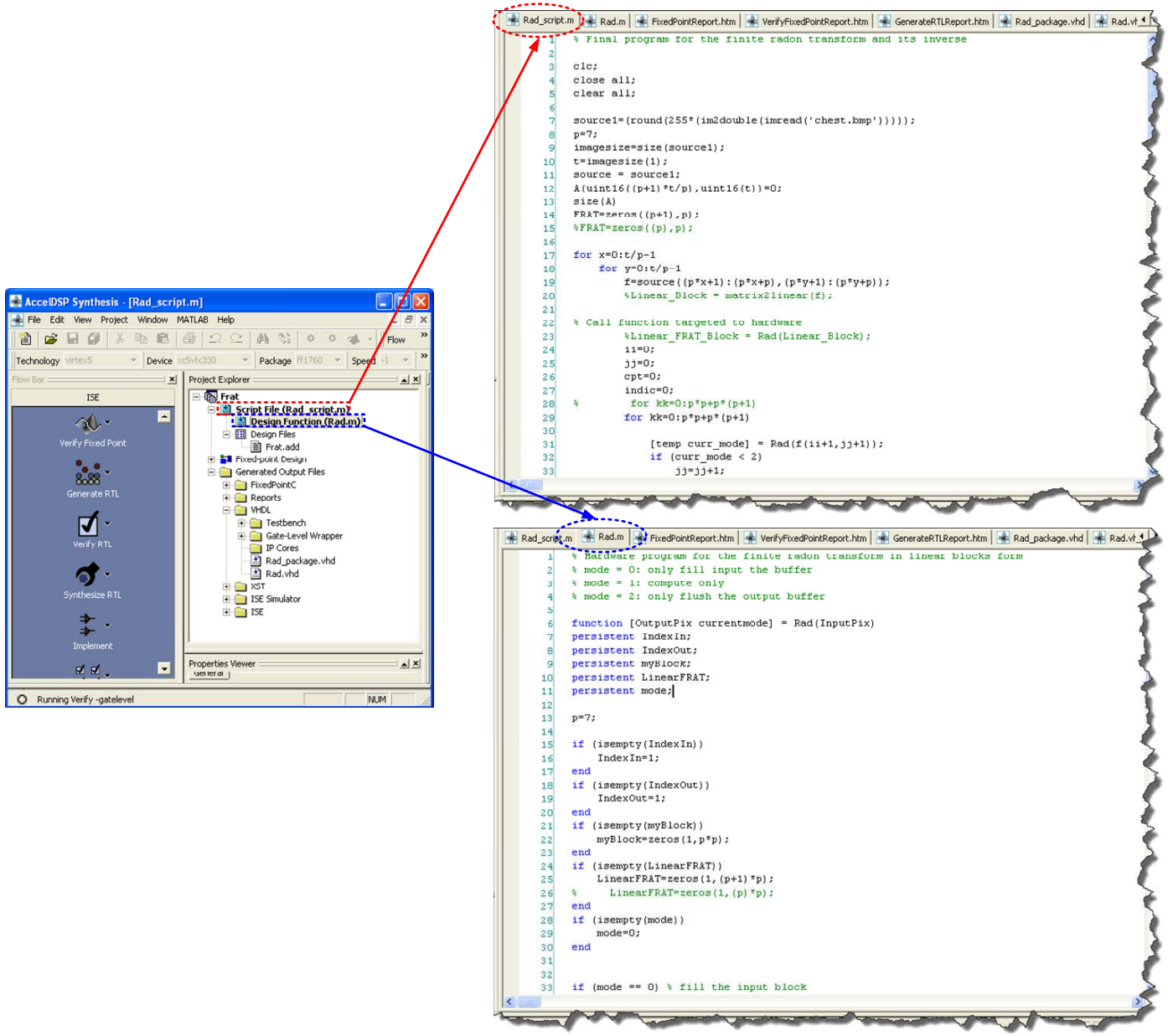

Figure 4.6: Script and function files for the sequential implementation.

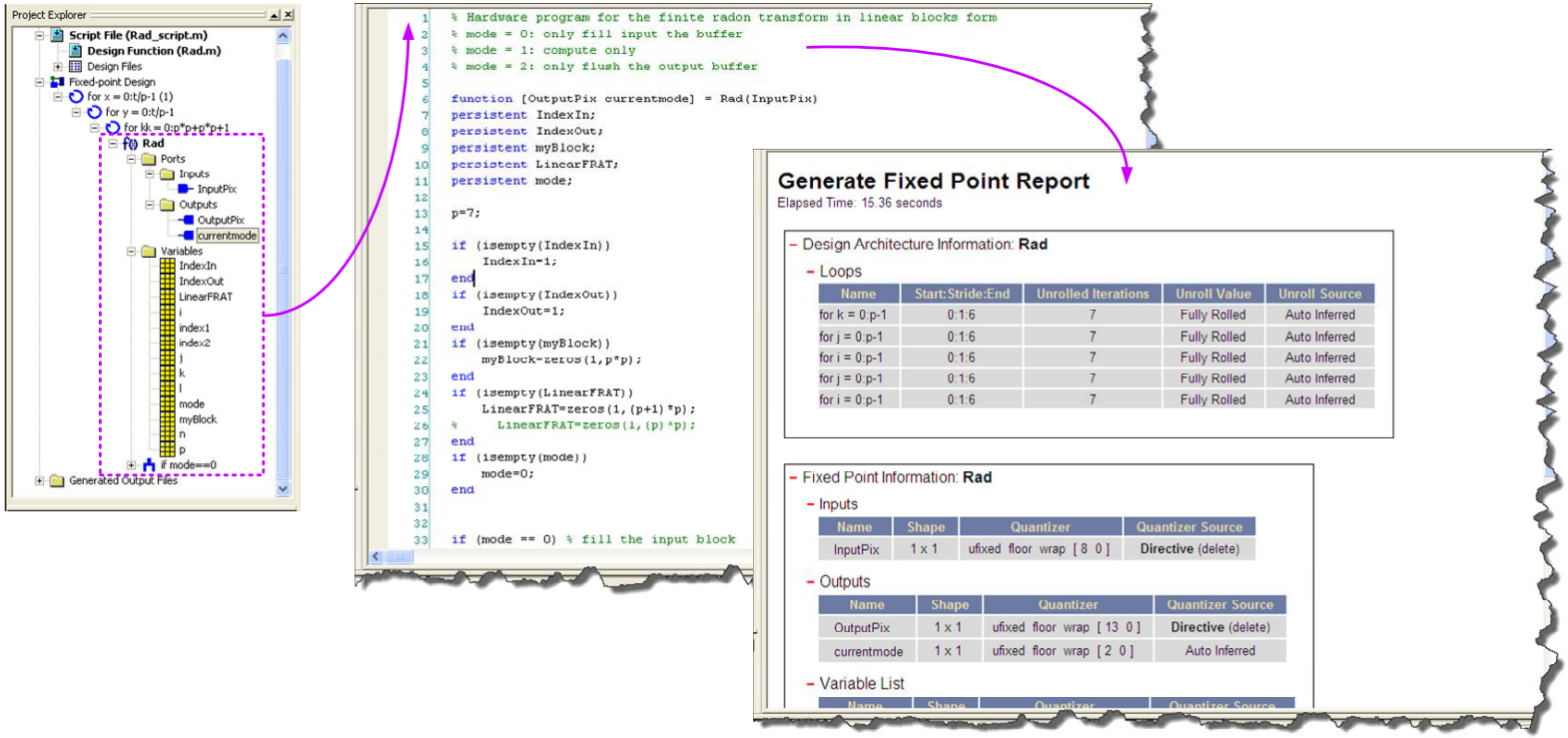

Figure 4.7: Function operations with generated fixed point report.

Figure 4.8: Project explorer with VHDL files generated.

the effectiveness of FRAT, peak signal to noise ratio (PSNR) has been calculated to quantitatively estimate the noise suppression.

4.4.1 Medical Image De-noising

To analyse the effectiveness of the FRAT in medical image de-noising, a Gaussian white noise with mean (μ) zero and various standard deviation (σ) has been added to the experimental images. By utilising FRAT in medical image noise reduction, results obtained have shown promising achievement. The de-noising results obtained reveal that the FRAT implementation is effective to reduce Gaussian noise.

Table 4.1 shows quantitative results for MRI images, while Figure 4.9(a) – (c) presents a significant achievement of 13.25% Gaussian de-noising for the MRI image using the FRAT. Better PSNR values have been achieved with smaller block size show a relationship of the block sizes which illustrates the relationship between the block size and the image blockiness problem. The smaller block size reduced the results of image blockiness in the Radon domain. Therefore, the FRAT representation of additive Gaussian noise reveals a better achievement with smaller block size as shown with $p = 7$.

Table 4.1: PSNR quantitative results of noisy image with a Gaussian white noise and MRI image.

σ	**Noisy (dB)**	**De-noising (dB)**		
		Block sizes (p)		
		7	**17**	**31**
0.01	31.15	43.60	39.51	36.48
0.02	30.63	41.68	37.75	34.67
0.04	30.30	39.34	35.47	32.38
0.08	30.06	36.98	32.95	31.14
0.16	29.91	34.48	30.44	30.29

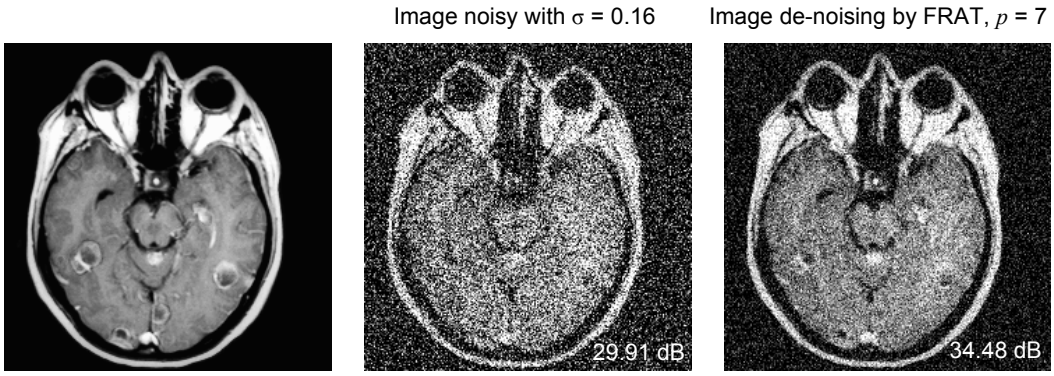

Figure 4.9: Gaussian noise reduction experimental results on MRI image (a) Original (b) Noisy (c) De-noising.

4.4.2 Software Simulation

Figures 4.10(a) – (h) show the FRAT domain visualisation of MRI and PET slices. The averaging impact of the FRAT on the image blockiness in the transform domain can be observed as p increases.

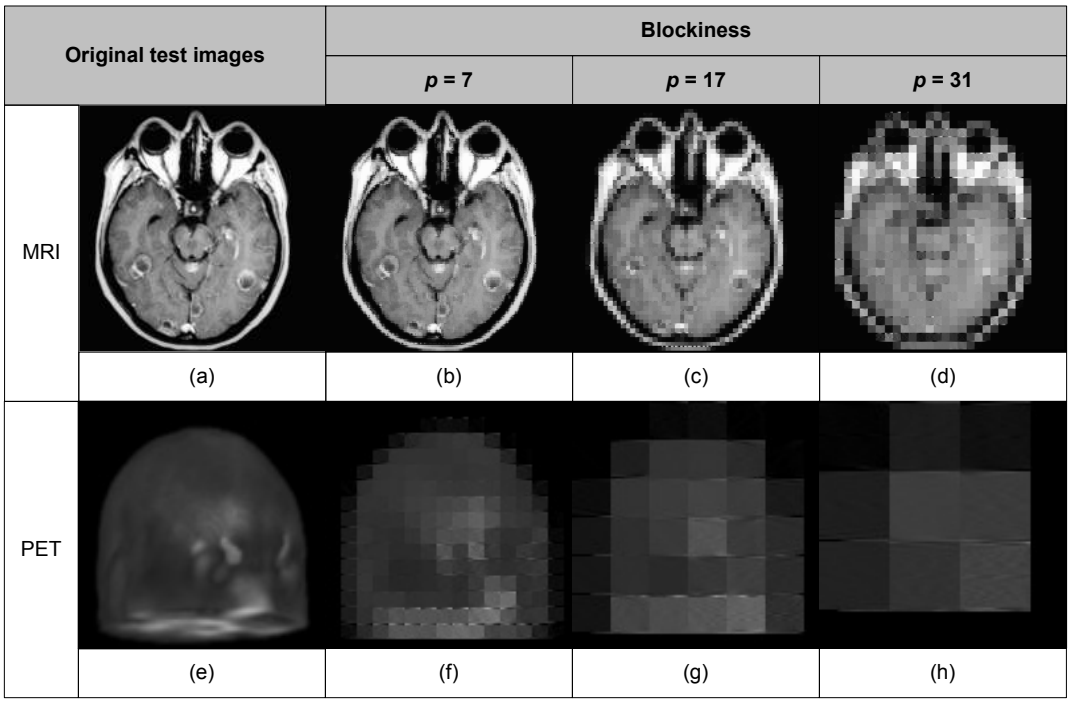

Figure 4.10: Original and blockiness images.

It is worth noting that the FBP is a mathematically perfect inversion for the FRAT, and the PSNR depends only on the accuracy required. The truncation or rounding step that follows the FRAT, determines the PSNR values. As it is usually used as a sub-block in other transforms such as finite ridgelet transform (FRIT) and curvelets, and it is followed by a wavelet stage in these transforms, the rounding or truncation process can easily be incorporated along with the wavelet block with no extra computational effort by suitably modifying the wavelet coefficients.

However, to illustrate the effect of bit-width limitations on the PSNR, Figure 4.11 shows the relationship of the PSNR values for the reconstructed medical images with various block sizes (p). Results obtained exhibit that the PSNR of the reconstructed image drops by 7.93, 21.70 and 21.80 dB for MRI, CT and PET, respectively when the block size increases from $p = 7$ to 31. This is because as p increases, the rounding error becomes more significant. Using a divider with greater precision can reduce the rounding error.

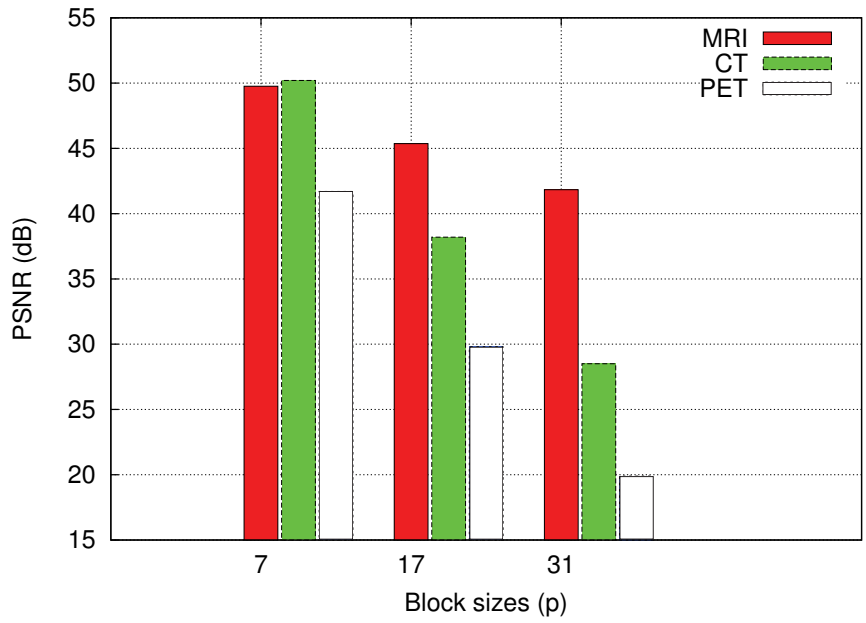

Figure 4.11: Analysis of PSNR with different block sizes (p).

4.4.3 Hardware Implementation

For all three cases of hardware implementation: sequential, pipelined and BRAM-based method, pseudo-codes have been implemented in MATLAB and the Xilinx AccelDSP has been used for architecture and synthesis exploration. The designs have been implemented on Virtex-5 (XC5VLX110T) FPGA devices. As the prime aim of this chapter is to examine the best hardware implementation applied for medical image de-noising, results for both medical image de-noising as well as the software simulation justify the hardware implementation with $p = 7$. Comparison of performance metrics for the proposed FRAT architectures with existing work is presented in Table 4.2.

Table 4.2: Comparison of performance with existing architectures for the case $p = 7$.

Type	Platform	Design	F	T	A
Sequential	Virtex-E	[71]	94.46	45.01	245
		[76]	69.00	6.90	345
	Virtex-II	[71]	79.973	37.32	215
		[75]	100.1	9.87	159
		[72]: A1	112.87	11.13	198
		[72]: A2	67.3	6.64	131
	Virtex-5	Proposed (1)	174.30	0.12	669
		Proposed (2)	127.80	8.52	2,704
Pipelined	Virtex-5	Proposed (1)	161.40	13.31	2,044
		Proposed (2)	103.50	48.30	5,286
BRAM-based	Virtex-5	Proposed (1)	188.90	6.30	637

Note:
F: Maximum frequency (MHz),T: Maximum throughput (MSPS), A: Area (slices)
Proposed (1): Loops rolled, Proposed (2): Loops unrolled
A1: Architecture 1, A2: Architecture 2

Results achieved for the hardware implementation demonstrate various trade-offs with sequential and pipelined descriptions yielding better achievement for maximum frequency and throughput, respectively. Moreover, BRAM-based method also reveals less area occupied and better maximum frequency. To visualise the design and implementation of the FRAT, Figure 4.12 illustrates the chip layout for sequential

implementation on XC5VLX110T FPGA device.

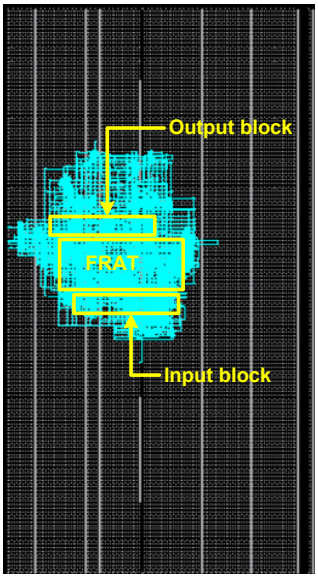

Figure 4.12: Chip layout for the sequential implementation.

A detail comparison for both hardware implementation and software simulation with test medical images has been carried out. As shown in Table 4.3, software simulation achieved better PSNR over hardware implementation with the percentage different 12.92%, 21.47% and 33.09% for $p = 7$, 17 and 31, respectively. This is due to the use of floating point in MATLAB, which yields better PSNR values compared with fixed point model in the hardware implementation.

Table 4.3: Comparison of PSNR values for CT images.

Experiments	**PSNR (dB)**		
	Block sizes (p)		
	7	**17**	**31**
Hardware implementation	46.30	38.13	30.30
Software simulation	53.71	48.56	45.29

4.5 Summary

In conclusion, an FPGA-based architecture with three different design strategies has been proposed and an analysis with various medical imaging modalities has been conducted. Image de-noising implementation using the FRAT exhibits a significant achievement to reduce a Gaussian white noise in medical images. An evaluation of the implementation results indicates promising trade-offs achievement in terms of maximum frequency, throughput and area. Chapter 5 will look into the design and implementation of medical imaging compression system on FPGA, where the FRAT can be used as a pre-processing block to improve the compression performance.

Chapter 5

FPGA-based Implementation of a 3-D Medical Image Compression System using CAVLC

5.1 Overview

Previous Chapter 4 discusses the capability of finite Radon transform (FRAT) for medical image de-noising in pre-processing stage before the compression. In this chapter, an implementation of three-dimensional (3-D) medical image compression on field programmable gate array (FPGA) is discussed.

The field of medical image compression introduces a complex problem. For certain image modalities, it is undesirable to lose information and the disregarding of certain image details through the use of lossy compression can consequently affect the outcome of patient diagnosis, hence lead to both critical human and legal consequences [5]. Higher compression ratios can be achieved using a lossy compressor

such as wavelet techniques [128], thus it allows a function to be described in terms of a coarse overall shape with details that range from broad to narrow.

H.264/AVC is one of the new emerging video compression standard that manipulates a combination of novel advanced coding technologies based on the mature hybrid block-based coding framework [129]. In the last step of H.264/AVC, either of the entropy coding techniques namely context-adaptive variable-length coding (CAVLC) or context-based adaptive binary arithmetic coding (CABAC) is used [130]. CAVLC utilisation demonstrates higher coding efficiency even at low-bit rates as well as reduce the data redundancy, whilst CABAC that is used specifically for lossless compression of images suffers with higher computational demand. To achieve better compression efficiency, CAVLC is fully utilised in this study and FPGA platform is used to accelerate the computational complexity of the 3-D compression processes [89].

This research aims at developing a novel implementation of three-dimensional (3-D) medical image compression system using CAVLC. An efficient hardware implementation for the transform block by optimising integer transform (IT) and discrete wavelet transform (DWT) has been proposed and evaluated. Software simulation as well as hardware implementation have been deployed using different medical image modalities. An in depth evaluation of the transform and CAVLC implementation in terms of area, power consumption and maximum frequency is also addressed.

The composition of this chapter as follows. Section 5.2 gives an overview of the algorithms and methodology. Section 5.3 explains the proposed system architecture including transform, quantisation and CAVLC blocks. Software simulation and hardware implementation results with analysis and discussion are discussed in Section 5.4. Finally, a brief summary is given in Section 5.5.

5.2 Algorithms and Methodology

Algorithms and design methodology for 3-D IT, DWT and the decomposition strategies are presented in the following subsections.

5.2.1 3-D Integer Transform (IT)

The 3-D discrete cosine transform (DCT) is used in video compression if there is a strong correlation between the adjacent pixels in a frame (spatial correlation), and the correlation between the pixels of the same position in adjacent frames (temporal correlation).

The principle is based on the idea that a video sequence could be seen as a tri-dimensional block, where the third dimension is the time. 3-D DCT is also applied for volume compression in medical imaging, and in this case the principle is the spatial correlation between the adjacent voxels in a volume.

Mathematically, the 3-D type-II DCT of $X(i,j,k)$, of size $N \times N \times N$, can be defined as:

$$Y(u,v,w) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} X(i,j,k) C(i,u) C(j,v) C(k,w) \tag{5.1}$$

where,

$$C(p,q) = \begin{cases} \frac{1}{\sqrt{N}} & , q = 0 \\ \sqrt{\frac{2}{N}} \cos\left(\frac{(2p+1)q\pi}{2N}\right) & , q \neq 0 \end{cases} \tag{5.2}$$

The two-dimensional (2-D) IT is a derivative of the 2-D DCT and has been adopted in the latest H.264/AVC standard for coding image blocks in residual data (texture). The associated integer arithmetic guarantees fast and accurate coding/decoding.

Though IT is a derivative of DCTs, in H.264, to maintain integer arithmetic capability, the post and pre-scaling factors of transform process are integrated into the forward and inverse quantiser stages, respectively, for reducing the total number of multiplications and avoiding the loss of accuracy.

Since the 3-D DCT is a separable transform, it can be implemented as a series of one-dimensional (1-D) transforms (or as a 2-D transform followed by a 1-D transform). The 3-D IT as shown in the Algorithm 5.1 can be derived from the 3-D DCT and implemented in the same manner.

Algorithm 5.1 The Forward 3-D IT

```
1: for slice = 1 to noslices do
2:    for row = 1 to norows do
3:       Apply a 1-D integer transform column-wise
4:    end for
5: end for
6: for slice = 1 to noslices do
7:    for col = 1 to nocols do
8:       Apply a 1-D integer transform row-wise
9:    end for
10: end for
11: for col = 1 to nocols do
12:    for row = 1 to norows do
13:       Apply a 1-D integer transform slice-wise
14:    end for
15: end for
```

The simplified representation of the 2-D IT for $N = 4$ and the output Y of the IT core can be defined in matrix representation as follows:

$$Y = (C\,X\,C^T) \odot E_f \tag{5.3}$$

where C and E_f are given by:

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \tag{5.4}$$

$$E_f = \begin{bmatrix} a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \\ a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \end{bmatrix} \tag{5.5}$$

and,

$$a = \frac{1}{2} b = \sqrt{\frac{1}{2}} \cos\left(\frac{\pi}{2}\right) \tag{5.6}$$

$$b = \sqrt{\frac{1}{2}} \cos(\frac{\pi}{2}) \tag{5.7}$$

The symbol $\odot$ indicates that each element of $(C\,X\,C^T)$ is multiplied by the scaling factor in the same position in matrix E_f, whilst C^T is the transpose of C.

Many video coding standards use a transform coding of the prediction residual. In previous standards, a DCT was used, but in the recent coding standard H.264, a new scheme of transform is used and called IT because of a separable IT with similar properties as a 4×4 DCT is used. As for DWT or DCT, IT performs a transformation between spatial and frequency domain in order to grab the most significant information of the picture in a particular place of each 4×4 block. This process improves drastically the entropy coding efficiency. The fundamentals difference between IT and DCT are as follows:

1. IT is based on integer coefficients, all operations can be performed using integer arithmetic and by the way, loss of accuracy is avoided;

2. The implementation can be made only with adders and shifters; and

3. A multiplication for scaling is implemented in the quantiser directly reducing the number of operation in the IT.

5.2.2 3-D Discrete Wavelet Transform (DWT)

The Haar wavelet mathematical background as has been explained in Chapter 3, is simple and computationally cheap, and can be implemented by a few integer additions, subtractions, and shift operations. It yields a multi-resolution representation for discrete data. The potentiality of the 3-D Haar wavelet in approximation of 3-D volumes was earlier discussed by Bajaj *et al.* in [119]. This wavelet filter has been selected as a consequence of its simplistic nature and mathematical features.

The factorisation of the Haar wavelet coefficients into lifting steps is expressed as follows:

$$d_{1,l} = s_{0,2l+1} - s_{0,2l} \tag{5.8}$$

$$s_{1,l} = \frac{s_{0,2l} + s_{0,2l+1}}{2} \tag{5.9}$$

where the original signal of interest is denoted by $(s_{0,j})_j$. $(s_{1,j})_j$ and $(d_{1,j})_j$ represent the low and high-pass coefficients after the wavelet transform. To obtain lossless compression, the Haar transform must support integer-to-integer mappings. This can be achieved by flooring the right hand side of Equation 5.9.

Decomposition strategy based on integer lifting offering the following advantages over classical wavelet construction utilising convolution:

1. Lifting allows for an in-place implementation of the wavelet transform, similar to that of the fast Fourier transform (FFT). The wavelet transform is calculated without the allocation of auxiliary memory;

2. Lifting allows for the construction of non-linear wavelet transforms. For example, wavelet transforms that map integers to integers, important for lossless image coding and hardware implementations;

3. Lifting allows for the construction of wavelets without making use of the

Fourier transform. This means wavelets can be built, which are not necessarily translations and dilations of one function, so called second generation wavelets;

4. Every transform constructed using lifting is immediately invertible, where the inverse transform has the same computational complexity as the forward transform. This is performed by reversing the operation order and inverting all signs; and

5. Lifting exposes the inherent parallelism in the wavelet transform. All operations in one lifting step can be computed in parallel, whilst the only sequential part is the order of the lifting operations.

Utilising the lifting scheme (LS), the order of the decomposed coefficients differs from the one that obtained via convolution. The decomposed stream is then reordered to regain the sub-band structure required for encoding. The coefficient orderings for both techniques are shown in Figure 5.1.

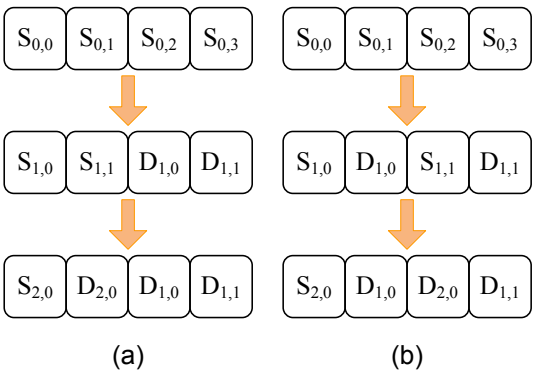

Figure 5.1: Coefficient orderings (a) Convolution-based (b) Lifting-based.

5.2.3 Decomposition Strategies

Both the non-standard wavelet decomposition (NSWD) and hierarchical block wavelet decomposition (HBWD) are frequently used for the compression of 3-D image volumes [119], [131]. The standard wavelet decomposition (SWD), however, has never been used for this purpose due to its unsuitability to generate appropriate

coefficients for entropy coding. This strategy, which is similar to the NSWD reflects the interactions between all pairs of scales, whereas the NSWD uncouples the interactions between the scales [132], [133].

The 3-D SWD is a novel extension of the 2-D SWD developed by Beylkin *et al.* [132] and this algorithm is described as follows:

1. A one level 1-D DWT is applied to each row of voxel values within the volume. This process splits the original image volume into two, one half representing a low-pass filtering coefficients and the other high-pass coefficients. The row length is then halved and this transform reapplied, the process being repeated until the required number of decomposition levels for the image is reached;

2. A one level DWT is applied to each column of the volume obtained as a result of Step 1. By repeatedly halving the column size and reapplying the 1-D DWT, the desired level of decomposition is once more achieved; and

3. A one level DWT is applied to all $z-$axis projections within the volume obtained as a result of Step 2. This is applied iteratively, halving the breadth as the number of decomposition levels increase, until the complete x-level 3-D SWD is obtained.

Figure 5.2 shows the sub-band structure achieved using a three level SWD, whilst Algorithm 5.2 gives the pseudo-code for the 3-D forward SWD. This decomposition can be inverted by initially calculating the dimensions of the finest sub-band and processing each axis in reverse order, using the same inherent looping structure. Each axis is processed independently, where recomposition is achieved fully in one orientation before the next is processed. The total recomposition along each orientation is obtained by iteratively applying the one step inverse 1-D DWT and doubling the individual orientation length. It is noted that the decomposition strategy that has been applied for 3-D IT is the same as the decomposition strategy that has been applied for the 3-D DWT.

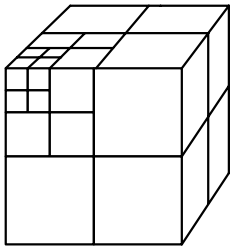

Figure 5.2: Sub-band structure obtained via a three level SWD.

Algorithm 5.2 The Forward 3-D SWD

```
for level = 1 to nolevels do
    for slice = 1 to noslices do
        for row = 1 to norows do
            Apply a one level 1-D DWT column-wise
        end for
        nocols = nocols/2
    end for
end for
for level = 1 to nolevels do
    for slice = 1 to noslices do
        for col = 1 to nocols do
            Apply a one level 1-D DWT row-wise
        end for
        norows = norows/2
    end for
end for
for level = 1 to nolevels do
    for col = 1 to nocols do
        for row = 1 to norows do
            Apply a one level 1-D DWT slice-wise
        end for
        noslices = noslices/2
    end for
end for
```

5.3 Proposed System Architectures

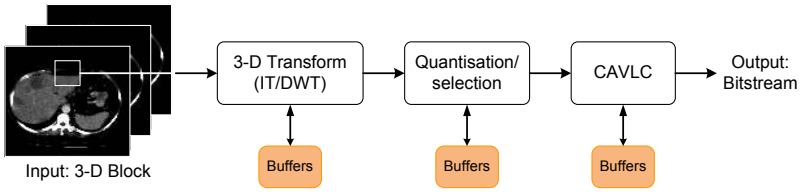

Figure 5.3: Proposed system overview.

Figure 5.3 illustrates an overview of the proposed medical image compression system using CAVLC including the transform and quantisation blocks. In each block,

buffers have been used for storing intermediate results to be processed. Since the application targeted for 3-D medical images, the transform block in this system including the 3-D IT and DWT (with Haar filter). The novelty in this proposed architecture highlighted by the utilisation of CAVLC hardware implementation and the possibility to reconfigure the system to choose between 3-D IT or the HWT.

5.3.1 Transform Block

Integer Transform (IT)

The architecture of 3-D IT can be realised as the architecture of 1-D IT and the butterfly expression of the four pixels 1-D IT as depicted in Figure 5.4.

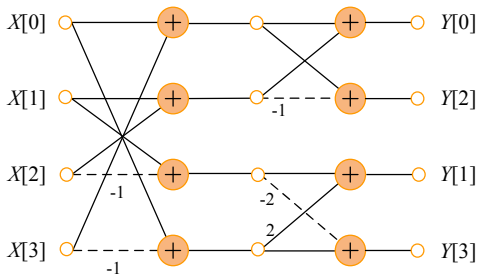

Figure 5.4: Butterfly architecture of 1-D IT.

It is composed of two stages regular butterfly architecture. The dashed lines refer to the subtractions, and the numbers represent the coefficients to be multiplied. To reduce the hardware complexity as well as to improve the area utilisation, the multiplication can be replaced by shifters and adders. The computation of IT can be divided into two stages: horizontal and vertical transform, and each process can be regarded as four 1-D vector transforms. Therefore, one full 4×4 IT contains eight 1-D transforms. At the same time, the butterfly computation process of 1-D transform needs eight "add" operations and two "shift" operations. Therefore, the computation of one full 4×4 IT requires 64 "add" operations and 16 "shift" operations. In this work, with a volume of $4 \times 4 \times 4$ voxels, the transformation can be performed by applying a 2-D IT over 4 slices of 4×4 voxels, followed by 1-D IT. The total number of

operations is $4 \times 64 + 32 = 288$ "add" operations and $4 \times 16 + 8 = 72$ "shift" operations.

Discrete Wavelet Transform (DWT) with Lifting Scheme (LS)

With an efficient integer wavelet decomposition to implement sampled filter banks that have an integer output, the LS has gained much interest in wavelet-based image representation especially for both high-throughput and low-power applications [134]. Moreover, it is also a better approach and more flexible than the convolution methodology, and capable of define a wavelet basis on an interval without using the Fourier transform concept [135], [136]. The LS that has been proposed by Herley and Swelden [137], [138] is a fast and efficient method to construct two-channel filter banks. The LS as depicted in Figure 5.5 with P as the prediction filter, U for update filter and R the rounding system.

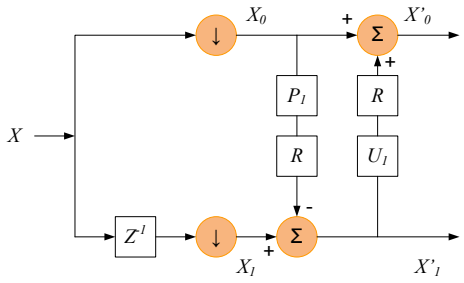

Figure 5.5: A simple lifting-based perfect reconstruction encoder.

Generally, this scheme consists of splitting the image in separate components, estimating components from others and further adding two components filtered version of other components [135]. The term "prediction", refers to the step with estimating the intensity of a coefficient, while the step related with smoothing the coefficients is called as "update".

5.3.2 Quantisation and Reordering Block

Quantisation is a very simple function that related with quantisation parameter that affects the coefficient value at the output module. Since it allows for hardware

simplification without multiplication and division operations, proposed quantisation architecture in [139] has been fully utilised as a design reference in this work.

5.3.3 Context-based Adaptive Variable Length Coding (CAVLC) Block

Architecture

Variable length coding (VLC) plays an important role in video and image compression applications. By assigning shorter codewords, VLC can removes redundant data efficiently. The recently developed video coding standard H.264/AVC significantly outperforms previous standards such as H.263 and MPEG-4 in terms of coding efficiency.

In H.264/AVC, a special VLC method named as CAVLC is used to encode residual, zig-zag ordered 4×4 (and 2×2) blocks of transform coefficients. According to the previously encoded data, CAVLC can adaptively choose one of the several VLC tables, and then encode the current input symbol efficiently by using various syntax elements. For quantised IT coefficients, most non-zero coefficients are centered in low-frequency. In high-frequency, most coefficients are zero and other non-zero coefficients are -1 or $+1$. CAVLC uses these characteristics to compress the data and increases the coding efficiency. The coding processes of CAVLC are as follows:

1. Scanning the quantised IT coefficients in zig-zag order;
2. Encoding the number of non-zero coefficients and the number of trailing coefficients (which value is -1 or $+1$), adaptively selecting the coding table according to the number of non-zero coefficients of left and upper blocks;
3. Encoding the sign of trailing coefficients;
4. Encoding the level information for other non-zero coefficients; and

5. Encoding the run information before each non-zero coefficient. It shows that the number of non-zero coefficients of current block will affect the selection of the coding table in other block; any change of it can affects the other block. Also, any change of trailing ones will cause the change of the non-zero coefficients. It will eventually affects the video quality and the bit-rate. So, the best method is not to change the number of non-zero coefficients of current block and the trailing ones.

The modules that compute and generate the bitstream, receive their coefficients from a transformation (IT or DWT), followed by a quantiser and reorganised through a zig-zag scanning. In general, CAVLC encodes each block in five independent steps. In the first step, it generates `CoeffToken` that encodes both the total number of non-zero coefficients (`TotalCoeffs`) and the number of trailing -1 or $+1$ values (`TrailingOnes`) in a block. `CoeffToken` also depends on the number of non-zero coefficients in the left-hand and upper of the previously coded blocks.

The second step allows the encoding of the `TrailingOnes` and represents the sign of each coefficient by a single bit. The `TotalZeros` parameter encodes the amount of all zeros preceding the highest non-zero coefficient, followed by the actual encoding of the non-zero coefficients contained in the block. It is worth noting that, at this step the proposed architecture contributes by removing the look-up tables (LUTs) and generating the code from simple values, the coefficient sign and its absolute value.

In the next step, the encoding number of zeros in the block that related to the non-zero coefficients is carried out. Finally, it is necessary to indicate in the final bitstream the location of these zeros among the non-zeros coefficients. This step is called `Run_Before`, because it gives the number of zeros preceding each non-zero coefficient within the zig-zag reorganised block. The corresponding syntactic element is calculated in the compression sub-module as `Encode RunZeros`. Each of these steps is represented by a syntactic element and by concatenating the values, the final bitstream is generated.

Most approaches encode symbols sequentially. In these approaches, the execution time of the VLC encoding for a given block depends on the quantisation parameter and the type of video [101], [140]. The corresponding architectures introduce a random execution time that requires an intermediate buffer in upstream of the entropy encoder between the encoder and the quantiser. Indeed, variable execution time implies a variable input rate, whereas the output rate of the quantiser is fixed. Therefore, it is required to have a buffer and a controller to synchronise the buffer with the CAVLC encoder.

CAVLC is the entropic coder of H.264 in the base profile. It takes advantage of several statistic events that occur the most frequently in the coefficient blocks to encode. For an efficient implementation of this function as well as to optimise the execution time, a parallel architecture has been selected. The majority of the CAVLC hardware implementations encode the coefficient values in an iterative way in order to save the silicon area. The number of iterations, and consequently the time of global calculation, is hence dependent on the number of non-zeros coefficients that belong to the block to be encoded. During the implementation, it is noted on a multi-processor system that this particular task requires numerous operations and the execution time is significantly modified according to the number of non-zero coefficients.

The originality of the proposed architecture as shown in Figure 5.6 is based on a well-balanced use of massive parallelism that contributes for fast calculation as well as fixed execution time. The architecture is also fully combinational and does not need any controller. Only the loading time of the block is performed sequentially with 16 clock transitions needed to load the 16 coefficients of the block in the memory, the encoding has been executed in one clock transition, at the cost of a reduced maximal frequency due to an important critical path. However, this is negligible as the proposed architecture is capable even at a very low frequency (4 MHz).

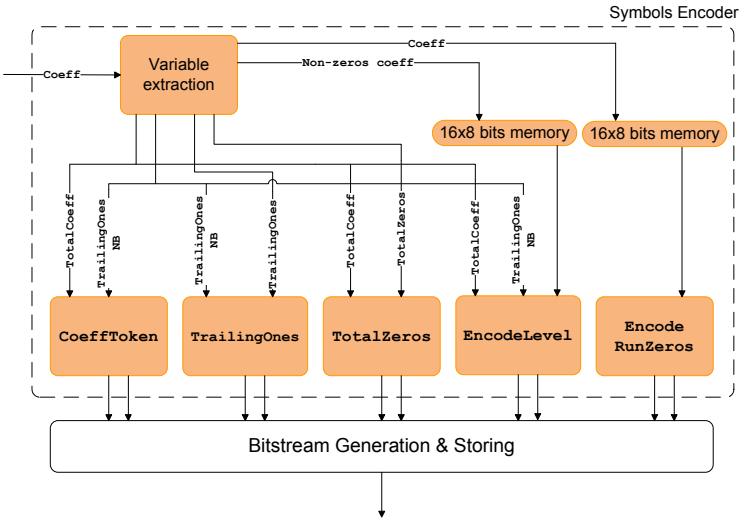

Figure 5.6: Block diagram of CAVLC architecture.

Operations

In Figure 5.6, a first phase of pre-treatment `Variable Extraction` can be distinguished, which introduces a latency of a few cycles. At this stage, there are data selections for the repartition to different encoding blocks. These data could be coefficients or intermediate results of non-zero coefficients (`Total_Coeff`) or the number of zeros (`Total_Zeros`). This pre-treatment is performed continuously by updating the output values for each new coefficient input. The expected values are obtained after 16 coefficients are processed and only then, the architecture delivers a valid bitstream for the block. It is then necessary to make a selection at the output of the encoder. Each of the tasks delivering a syntactic element is carried out in parallel, as they are independent of each other. The last module that generates the bitstream concatenates the syntactic elements issued from the preceding modules.

The CAVLC bitstream is encoded by coefficients blocks, in addition to encoding parameters that allow a reduction of the bitstream size. Thus, variables that have been defined in the algorithmic part of this chapter have to be encoded. Each of the sub-modules uses pre-determined tables that accelerate the values encoding. Only the `EncodeLevel` module, that processes coefficient values and `Encode RunZeros` require more complex calculations.

The number of non-zeros coefficients can be varied from 0 to 16 per block and most of the common architectures use an iterative computation on these numbers. This necessitates a controller and leads to a variable execution time that is unpredictable. For each coefficient, the code-word depends on both of its value and the preceding coefficients value. Indexes are used and will be incremented or not for the following coefficient encoding depending on the coefficient value. These indexes are calculated beforehand and then distributed to the 16 processing modules. Each module uses these indexes and the corresponding coefficient value, to find out the code-word to be used.

The sub-module `Table generation` contains the algorithm of the code-word generation. The code-words are calculated by two parameters: size and value. Using these two parameters, the bitstream generator can reconstitute the global bitstream. The same principle of parallelism has been used, in order to calculate syntactic elements of `RunBefore`, with the noteworthy difference that predefined tables is used to identify the value and size of each element. These tables are duplicated in each module in order to overcome the overloaded issue, and in contrast with the case where a single table is available for all 16 modules. As the number of the value is relatively restricted for these tables, the extra cost of memory is negligible.

A different approach by systematically performing in parallel the calculations of level encoding as shown in Figure 5.7 has been proposed. It obtained the totality of the codes in one clock transition regardless the number of the coefficients needed to be processed, and processes the iteration in 16 identical modules. These modules are purely combinational and allow encoding in a fixed time, which some of the modules are not being used and continue to function, whilst their results are not going to be taken into account. As an example, for seven non-zeros coefficients in the block imply that nine modules are not going to be used, but they will execute the calculation on the values which most likely, are coming from the preceding blocks.

Due to the treatment for each coefficient in a block is mainly parallelised, all module that has been placed after the variable extractor process all the 16 coefficients

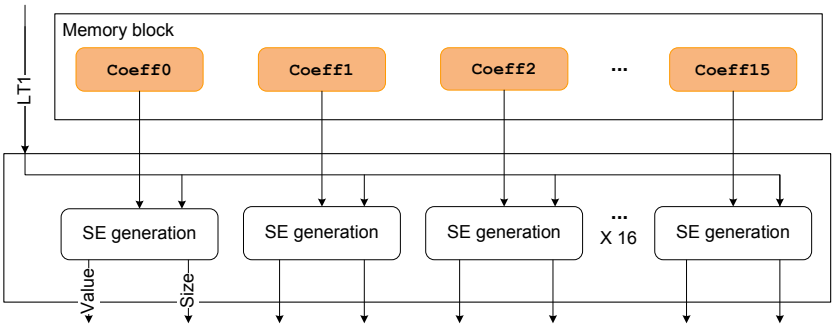

Figure 5.7: `Encode level` detail of the CAVLC architecture.

(as a whole block) in only one clock cycle. To encode of a 1920×1080 at 30 frames per second (fps), a throughput of 63 Mpixels is needed. Hence, this is the reason of the variable extractor runs at 63 MHz as one pixel (or coefficient) is treated per cycle. As the majority of CAVLC modules can process 16 coefficients per cycle, clock can be reduced by 16 and majority of the architecture runs at only 4 MHz.

5.4 Results and Analysis

Software simulation and hardware implementation with the results analysis are presented in the following subsections.

5.4.1 Computational Complexity

This section describes the computational complexity of different decomposition strategy for the proposed transform block. For this task, the Haar wavelet basis is employed, enabling a simple calculation of total arithmetic operations to be performed.

In all decomposition strategy testing using the Haar basis, only addition, subtraction and division operations are required. For more complex filters, multiplications can be converted to "add" and "shift" operations using Booth's algorithm [141], as on certain hardware and software architectures, multiplication operations are more costly than additions and shifts.

Using a two level decomposition for a $4 \times 4 \times 4$ image volume, the number of addition and shift operations required for each transform is given in Table 5.1 with the main functional blocks and the following conclusions can be made:

1. Decompositions based on the LS, even with the required reordering for this application, are far superior to their convolution based equivalent decompositions. The use of lifting over the convolution enables a reduction in shift operations by 50%; and

2. 3-D IT performs with better computational complexity by 22.2% for total arithmetic complexity operations.

Table 5.1: Computational complexity of the main functional blocks with various decomposition approaches.

Decomposition Type	Adds	Shifts	Comparator	Multiplier	Total
3-D SWD (Convolution)	288	288	N/A	N/A	576
3-D SWD (LS)	288	144	N/A	N/A	432
3-D IT	96	32	N/A	N/A	128
Intra-prediction	59	10	6	N/A	75
Quantisation	15	N/A	N/A	9	24
CAVLC	111	4	72	16	203

5.4.2 Objective Evaluation

Objective evaluation performance for a range of bit-rates assessed using peak signal to noise ratio (PSNR) and mean squared error (MSE) [53], [56–58]. Since more than one definition of PSNR exists, the one that has been employed in this study is:

$$\text{PSNR(dB)} = 20\log_{10}\left(\frac{\text{Maximum voxel}}{\sqrt{\text{MSE}}}\right) \tag{5.10}$$

and,

$$\text{MSE} = \frac{1}{N} \times \sum_{i} \sum_{j} \sum_{k} [f(i,j,k) - F(i,j,k)]^2 \tag{5.11}$$

where N is the total number of voxels, $F(i,j,k)$ is the voxel value at point (i,j,k) in the reconstructed image and $f(i,j,k)$ is the voxel value at point (i,j,k) in the original image. For the bits per voxel (BPV), the bit rate (BR) is defined as:

$$\text{BR(BPV)} = \frac{\text{Size of compressed } 3-\text{D image (bits)}}{\text{Total no. of voxels}} \tag{5.12}$$

Simulation has been performed using MATLAB and details of the experimental test data [22] that have been used are described in Table 5.2.

Table 5.2: Images used for testing.

Image	Bit-Depth	Dimensions	Description
CT	8	$128 \times 128 \times 239$	CT volume with active lung tumor
MRI	12	$256 \times 256 \times 60$	MRI scan of human brain with active cerebral lesion
PET	16	$91 \times 91 \times 109$	PET scan of normal human brain

Each of the test images are compressed in a lossy manner with a range of bits-rates using various decomposition strategies for both IT and DWT. The quality of the reconstructed images is then measured using PSNR metric and BPV, as a result of easy quantification for compression efficiency. Since the gray-scale images have been used for simulation, BPV is equal to eight for an eight-bits grey-scale image for a non-compressed frame. The results for each image from 0 to 6 BPV are given graphically in Figure 5.8 – 5.10 for DWT using convolution or LS and IT.

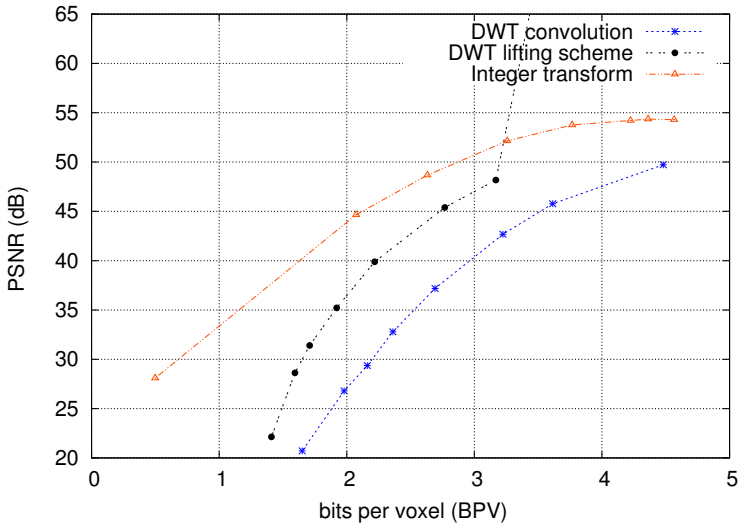

Figure 5.8: PSNR vs. BPV for CT.

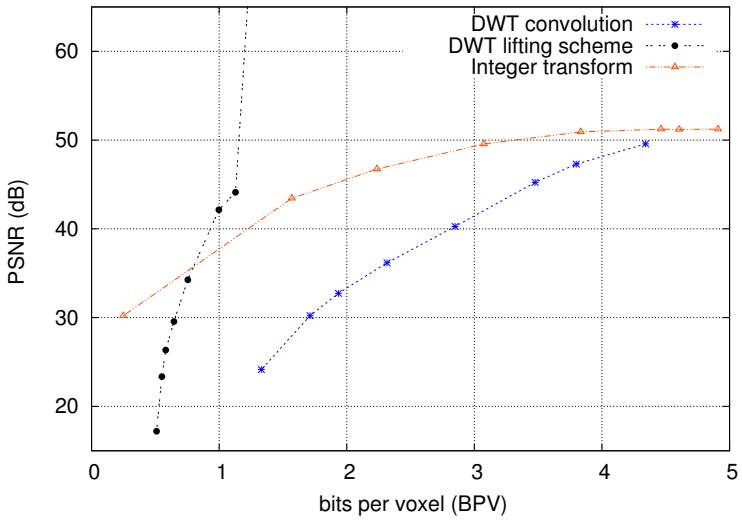

Figure 5.9: PSNR vs. BPV for MRI.

From the simulation results obtained, the following conclusions can be drawn:

1. In all cases, the DWT with LS performs a lossless compression on all images. However, the other two transforms yield some data loss;

2. In the case of CT images, it can be observed that the IT with CAVLC is the best solution to perform a good compression, but even without quantisation, compression is not lossless;

3. In the case of MRI images, the DWT with LS performs better results than the

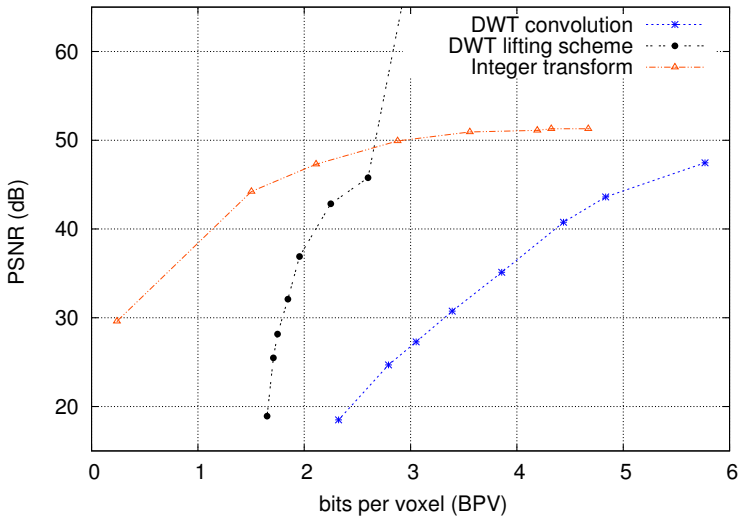

Figure 5.10: PSNR vs. BPV for PET.

IT and generates an efficient compression rate; and

4. In the case of PET images, the scheme is sensibly the same as for the CT images.

Figure 5.11(a) – (l) illustrates the comparison for the first medical volumes slices of the original and all the reconstructed slices for CT, MRI and PET images using 3-D DWT with LS, convolution and 3-D IT. It is noted that 3-D IT provides better results for CT and PET images, whilst LS exhibits better results in MRI image. Some factors such as sharpness and global frequencies that present in the frame, influence the compression ratio and quality.

It is worth mentioning that 3-D IT is always better than other transform due to the combination of intra-prediction, DCT-like transform and CAVLC that capable of reduce the spatial redundancy. However, DWT performs significant results for a lossless compression with high quality and significantly useful for medical image compression application. In simulation results, lossless compression can be achieved clearly by increasing the BR.

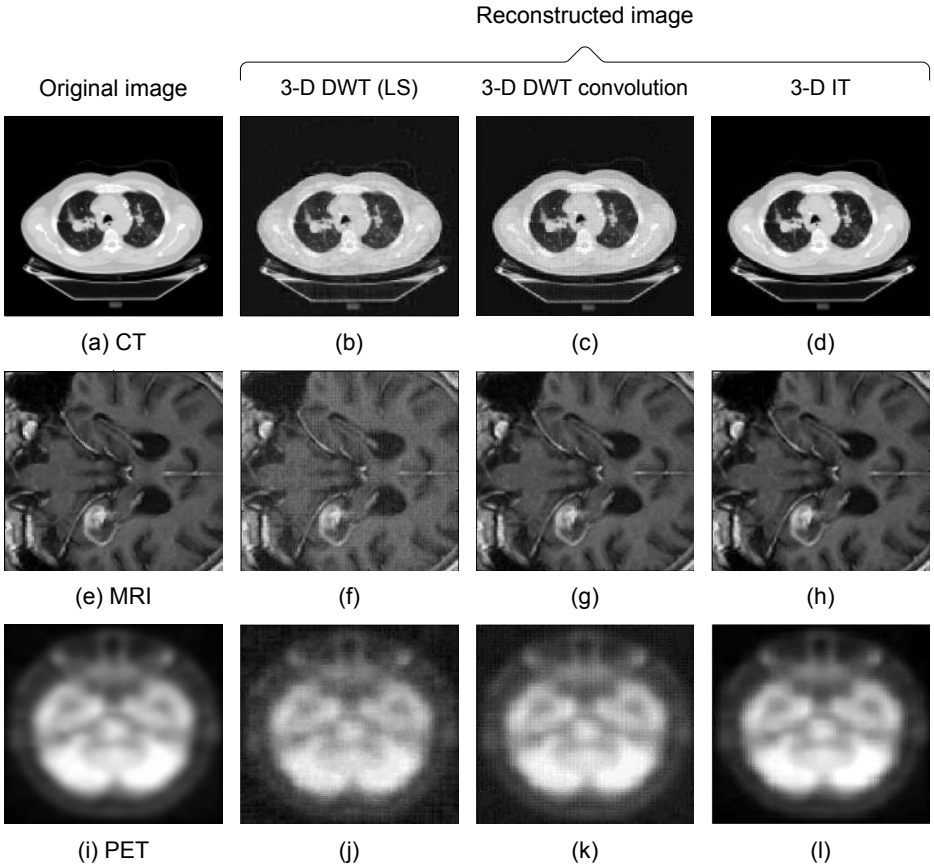

Figure 5.11: Comparison of original and reconstructed CT, MRI and PET images for the first slices.

5.4.3 Field Programmable Gate Array (FPGA) Implementation

The proposed FPGA-based architectures of a compression system for 3-D medical images using CAVLC have been synthesised using very-high-speed integrated circuit hardware description language (VHDL) and implemented on Xilinx University Program XUPV5-LX110T Development System with Virtex-5 XC5VLX110T device. Table 5.3 provides the details of results for each block and resources utilisation in terms of register, look-up table (LUT), digital signal processor (DSP) 48E and buffers. Results obtained reveals that intra-prediction and transform are the most consuming blocks in terms of registers and LUTs, respectively. Moreover, the frequency selection module is very negligible contrary to the other modules.

Table 5.3: Hardware resources utilisation for each block.

Blocks	Hardware Resources Utilisation			
	Registers	**LUTs**	**DSP 48Es**	**Buffers**
Transform	797	3,359	N/A	1
Quantisation	N/A	175	3	N/A
CAVLC	37	2,555	N/A	4
Intra-prediction	4,018	225	N/A	N/A

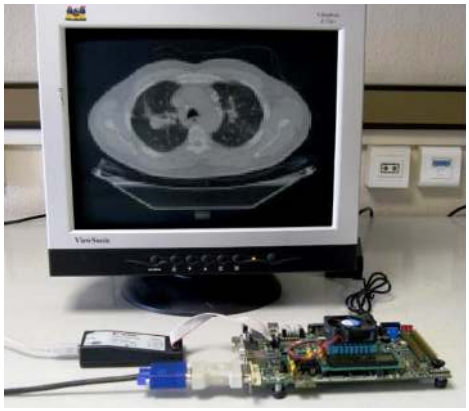

Figure 5.12: Compression system.

A complete functional 3-D medical image compression system as shown in Figure 5.12 has been implemented and medical images to be processed were stored in external double data rate (DDR-2) memory. The implemented system has been successfully demonstrated its functionality to compress and decompress with 60 fps for 640 × 480p format. Indeed the video graphic array (VGA) frames have been used for demonstration, the proposed system can be performed for a high-definition (HD) 1080p format at 63 MHz frequency. With parallelisation that has been applied, it is noted that the CAVLC block runs at 16 MHz.

Transforms Block

Comparison of the results obtained for both 3-D IT and 3-D HWT of the transform block implementation is listed in Table 5.4. In this case, N implies the transform size that reflecting the size of volume data in 3-D medical imaging modality. Results shown that the implementation of 3-D IT as a transform block requires more 70.51% and 3.52% for area and power, respectively, whilst 3-D HWT exhibits a better maximum frequency performance.

Table 5.4: Resources utilisation and overall transform architectures performance for $N = 4$.

Parameters	**Transform Architectures**	
	3-D IT	**3-D HWT**
Area (slices)	5,307 (7.68%)	1,562 (2.26%)
Power consumption (mW)	459.09	442.91
Maximum frequency (MHz)	128.00	223.56

Context-based Adaptive Variable Length Coding (CAVLC) Block

Table 5.5 presents the implementation results of the CAVLC encoder different modules with the pre-treatment module as a master in the architecture. It runs at a maximum frequency of 152 MHz and generates a clock signal for the sub-modules that operates at frequency 16 times lower. As a result of parallelism, in one clock cycle these sub-modules provide the value of each syntactic element for a block of 16 coefficients.

Comparative study with the existing work in [142] demonstrates that the proposed architecture can save either the internal random access memory (RAM) or registers in the interface, between the quantiser and the entropy coding block. Moreover, thanks to the parallelisation of most computations, only the first part

Table 5.5: FPGA implementation results of CAVLC.

CAVLC/Processes	Hardware Resources Utilisation		
	Registers	**LUTs**	**BUFG/BUFGCTRLs**
`Pre-processing`	32	28	1
`Memory480`	5	75	45
`CoeffToken`	N/A	1	N/A
`TotalZeros`	N/A	52	N/A
`MemoryEL`	N/A	16	1
`MemoryERZ`	N/A	16	1
`EncodeLevel`	N/A	1,906	N/A
`EncodeRunZero`	N/A	417	N/A

(variable extraction) of the architecture requires a frequency of 63 MHz as shown in Table 5.6.

In the CAVLC architecture, there are two levels: the first level computes some information for the current block, whilst the second level computes the syntactic element according to the variables in the first level. The proposed architecture is divided in half to detail the frequencies used with one or the other stages of the pipeline, in order to highlight the fact that most of the elements of the architecture work at a very low frequency.

Table 5.6: Comparison of CAVLC architectures performance on FPGA platforms.

Parameters	Comparison Study	
	[101]	**Proposed (stage1/stage2)**
FPGA	Virtex-II	Virtex-5
Gate count	6,855	28,152 (351/27,801)
Clock frequency (MHz)	50	63/4

Results obtained for the power consumes as shown in Figure 5.13, exhibit that the proposed architecture generates an efficient power usage. In brief, the total consumption is only decreased of 13%, whilst the dynamic power is also decreased by 85% for the same performance.

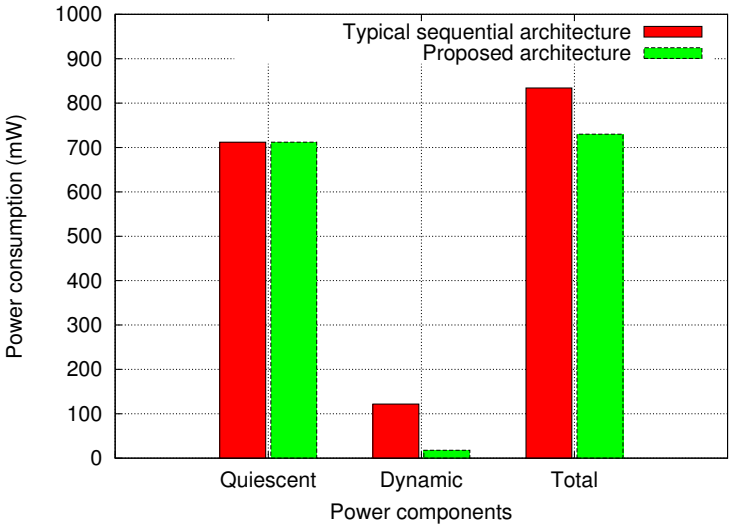

Figure 5.13: Power consumption comparison for the CAVLC architecture.

5.5 Summary

In this chapter, a novel hardware implementation of 3-D medical image compression system with CAVLC has been proposed. Analysis and performance evaluation of the 3-D images have been carried out for both aspect, computational complexity and quality. The evaluation of different transform filters has shown that 3-D IT generates better performance with regards to the computational complexity, whilst the DWT with LS provides a lossless compression that is significantly useful for medical image compression.

An architecture that is capable of compress high-definition images in real-time has been proposed. Through a judicious parallelisation, promising results have been obtained with limited resources. Furthermore, the architecture needs a relatively low

working frequency of 63 MHz for some pre-treatment and of only 4 MHz for the rest of the encoder.

Finally, the proposed architecture consumes 1.8 times less energy per processed block with the 3-D HWT than with the 3-D IT. This could be useful to make some trade-offs between compression ratio as well as energy consumption and even a good case for the use of run time reconfiguration (RTR).

Chapter 6

Conclusions and Future Work

6.1 Overview

This thesis has investigated issues and challenges of efficient reconfigurable architectures for three-dimensional (3-D) medical image compression. To date, the more widespread used of 3-D modalities in medical diagnosis have generated a massive amount of volumetric data. Moreover, other applications such as telemedicine and teleradiology require medical volumes to be transmitted from one station to another. Therefore, an efficient volumetric data transmission with limited bandwidth and storage is important. Although tremendous advantages offered by 3-D modalities, it is worth mentioning that the algorithms are computationally intensive and require hardware implementation to accelerate the process. In this research study, medical image compression for 3-D modalities is the main focus as well as the design and implementation of the algorithms using field programmable gate array (FPGA). In the rest of this chapter, results obtained throughout this research study are summarised and evaluated. Moreover, some possible routes to be explored for a future extension of this work are also provided.

6.2 Achievements

The main objectives of this work are discussed in Chapter 1 and 2. They are derived from limitations of some of the current issues in design and implementation of 3-D medical image compression system. Each of these objectives is now revisited in order to determine whether it has been met by this research.

1. **Has this research investigated the efficient implementation of reconfigurable architectures for 3-D Haar wavelet transform (HWT) in medical image processing applications?**

   This research has thoroughly investigated the efficient implementation of 3-D HWT for medical image processing. It has examined the contribution of dynamic partial reconfiguration (DPR) technique to deal with computational intensive 3-D medical image processing applications. As the transform size imply the complexity of medical volumes, the impact of transform size on architecture performance has also been examined. By implementing DPR technique in this research study, better area utilisation and minimum power consumption can be achieved as well as better maximum frequency [50], [107], [143].

2. **Has the research developed a novel implementation of the finite Radon transform (FRAT) for medical image de-noising?**

   The design and implementation of FRAT's field programmable gate array (FPGA)-based architecture for medical image de-noising has been developed. Since the image de-noising is crucial in pre-processing stage of a compression system, three design strategies to accelerate the computational process as well as maintaining the outcomes have been investigated. Moreover, to reduce the design cycle and to allow more effort to be carried out for architecture optimisation, Xilinx AccelDSP tool has been employed. Research findings of this work exhibit the design trade-offs for the hardware implementation. Furthermore, relationship of the block sizes with the image analysis has also been discovered. In terms of application, the proposed design and implementation of FRAT's on FPGA

demonstrates its capability to reduce a Gaussian white noise in medical images.

3. **Has the compression system using context-based adaptive variable length coding (CAVLC) been developed for 3-D medical images?**
   To evaluate the complete compression system, design and implementation of 3-D medical image compression system using CAVLC have been proposed. An evaluation for both software simulation and hardware implementation has been carried out for integer transform (IT) and discrete wavelet transform (DWT). On top of that, efficient architecture of CAVLC with parallelism optimisation has also been proposed. Through a judicious parallelisation, promising results have been obtained with limited resources [144].

6.3 Limitations

The objectives stated in Chapter 1 and 2 have been successfully fulfilled. However, a number of restrictions and limitations have been identified during this research.

1. Magnetic resonance imaging (MRI), computed tomography (CT) and positron emission tomography (PET) have been used as test images for the system implementation evaluation. Other medical modalities such as ultrasound (US) and angiogram should be taken into account to further examine the proposed design and implementation. To validate the implementation outcomes, clinical judgement from the experts will be very useful.

2. The maximum transform size that implies the medical volumes data is limited for $128 \times 128 \times 128$. To further increase the transform size, FPGA board configuration as well as considering an external memory are required. Moreover, matrix partitioning technique can be taken into consideration as a part of solution to partition the matrix into smaller sizes.

3. Real hardware implementation of the proposed architectures have been deployed on the Xilinx University Program XUPV5-LX110T Development System with Virtex-5 (XC5VLX110T-3FF1136) FPGA. It is notably that proposed designs and implementations are platform independent, and can be implemented easily on the most recent FPGAs. Moreover, by using the resources available on the recent platforms, better performance can be achieved. But, due to time and funding limitations, real implementations on these platforms were unable to be carried out.

6.4 Future Work

The following are suggestions for future work, which build upon the ideas and concepts presented in this thesis.

1. Tackling the issues of power dissipation and energy efficiency:

   The migration from application specific integrated circuits (ASICs) to FPGAs for a variety of applications brings along with it a number of issues to be resolved, including techniques for minimising power dissipation. Since FPGAs are now being increasingly employed in various applications, higher power dissipation results can make the chip run slower. If power dissipation exceeds the specification for a pin-package, the chip may get permanently damaged. Power dissipation is not only interesting from a packaging perspective, but also in determining the battery life of portable devices.

2. Exploring the implementation of floating-point cores:

   In the most previous implementations of architectures on FPGAs, floating-point arithmetic has not been used due to the considerable additional complexity and area needed. With recent and future FPGAs, the floating-point arithmetic operation can be efficiently used and implemented for architectures presented in this thesis.

3. Investigating other partitioning strategies on multi-FPGA platforms:

   Very large medical volumes can be accommodated by developing architectures based on partitioning strategies and algorithms. The partitioning can be performed at the algorithm level as well as the hardware level on multi-FPGA based hyper-computer platforms [145]. The reconfigurable hyper-computer allows high memory bandwidth, fast processing element inter-communication speeds, and fast external input/output (I/O) capabilities.

4. Exploring the wireless transmission of medical volumes:

   Although high-definition medical imaging (HDMI) wireless communication modules is now under development, it is worth mentioning that further investigation in the domain of 3-D volume analysis is of significant importance. Video transmission over wireless networks can be very useful, but also very challenging. Hardware implementation using FPGA with a capability to send and receive data in the wireless domain is significant, especially for telemedicine and teleradiology applications. In these applications, limited storage and bandwidth availability are of crucial. Therefore, medical image compression with efficient design and implementation for wireless domain can be further investigated.

5. Investigating heterogeneous system using graphical processing unit (GPU)-FPGA:

   An investigation into the used of heterogeneous system using GPU-FPGA is essential, since both platforms offer great mutual advantages for high computational demanding applications, such as medical image processing applications. Design trade-offs such as flexibility, performance and power consumption can be further analysed.

6. Implementing FRAT in the proposed compression system:

   Using FRAT as a basic building block for ridgelet transform to be deployed in the proposed medical image compression system is another interesting examination.

Since ridgelet transform is excellent for objects and curves representation, an examination of its capability is a very promising research area.

Appendix A

Rapid Prototyping Board and FPGA Devices

A.1 Overview

This appendix provides an overview of Xilinx University Program XUPV5-LX110T Development System and a brief discussion about the Virtex-5 field programmable gate array (FPGA) device. Comparative study of the FPGA resources for Virtex-5 with other FPGA devices such as Virtex-4, Virtex-E and Spartan-3L is also presented. Review and comparison cover the following features: configurable logic blocks (CLBs), digital signal processor (DSP) element, and the processor. Finally, a comparison table describing different resources available in different FPGA platforms is presented. This comparison enables the hardware designer to select the appropriate resources for a better hardware optimisation.

A.2 XUPV5-LX110T Prototyping Board

There are various rapid prototyping boards available in the market. Xilinx University Program XUPV5-LX110T Development System can be considered as a good option

due to the following features: cost, easy to operate and performance. Throughout this project, XUPV5-LX110T platform has been used to experimentally prototype the architectures as well as carried out the evaluation of the proposed system's applications. Figure A.1 shows a block diagram of the XUPV5-LX110T platform board, whilst Figure A.2 shows real top views of the platform with label of each component/peripheral.

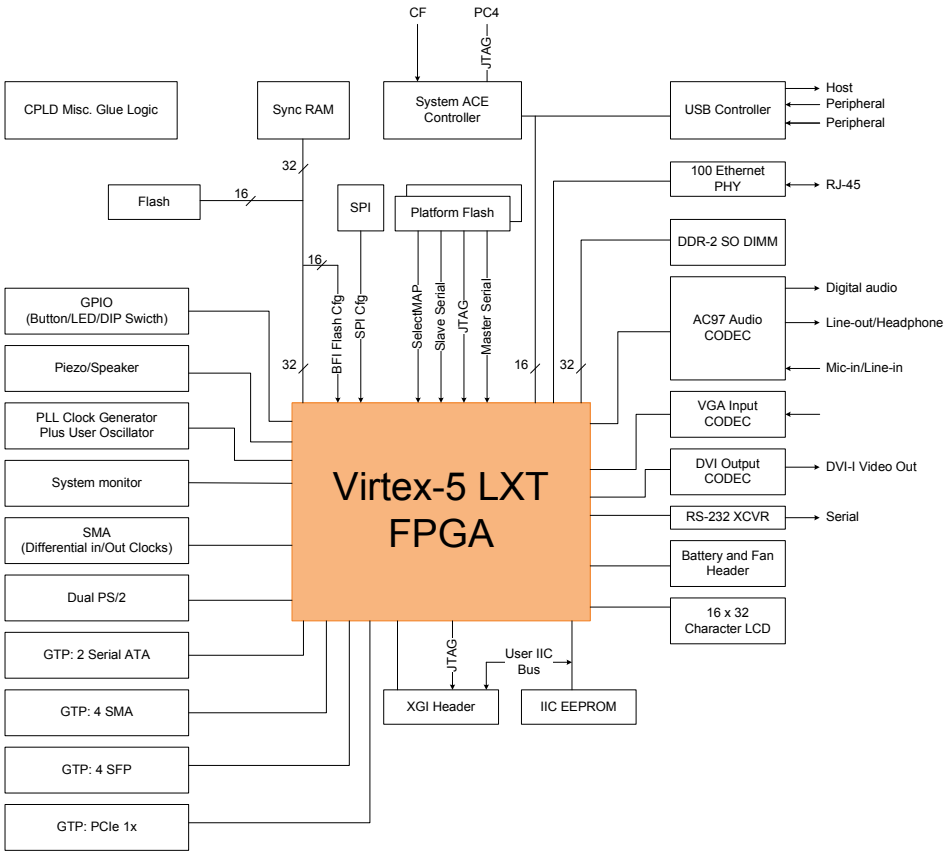

Figure A.1: Virtex-5 FPGA and XUPV5-LX110T platform block diagram [146].

A.3 Virtex-5 Field Programmable Gate Array (FPGA)

The Virtex-5 family provides powerful features in the FPGA market, and it has been used for synthesis and design prototyping throughout this research. With

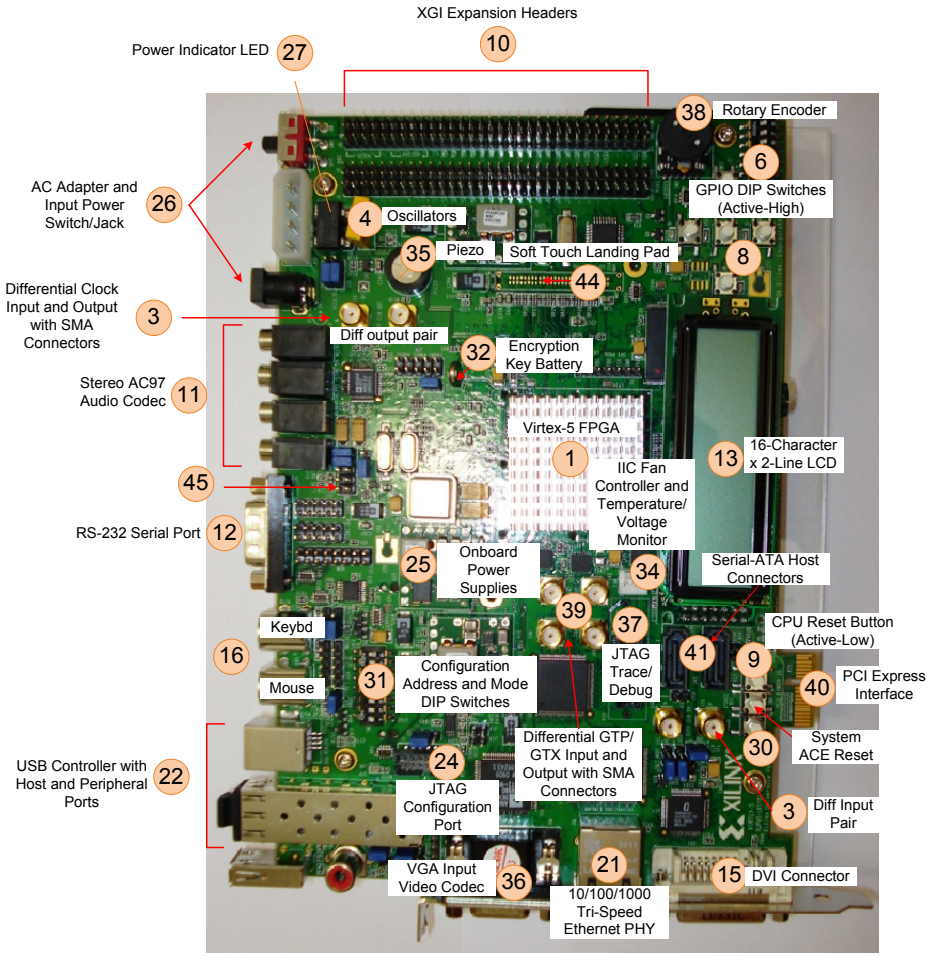

Figure A.2: Detailed description of XUPV5-LX110T platform components (front view).

65-nm state-of-the-art copper process technology, Virtex-5 FPGAs contain many hard-intellectual property (IP) system level blocks, including powerful 36-Kbits block random access memory (BRAM)/first in first out (FIFO), second generation 25×18 digital signal processor (DSP) slices, select I/O technology with built-in digitally controlled impedance, ChipSync source-synchronous interface blocks, system monitor functionality, enhanced clock management tiles with integrated digital clock managers (DCMs) and phase-locked-loop (PLL) clock generators, and advanced configuration options [146].

A.3.1 Configurable Logic Block (CLB)

A CLB element contains a pair of slices and for each CLB element is connected to a switch matrix for access to the general routing matrix as depicted in Figure A.3. Every slice contains four logic-function generators (or look-up tables), four storage elements, wide-function multiplexers, and carry logic. These elements are used by all slices to provide logic, arithmetic, and read only memory (ROM) functions. Additionally, some slices support two additional functions: storing data using distributed RAM and shifting data with 32-bits registers. As illustrated in Figure A.4, the Virtex-5 family is the first FPGA platform to offer a real 6-input look-up table (LUT) with fully independent (not shared) inputs, and it leads to significant advantages for hardware optimisation [146].

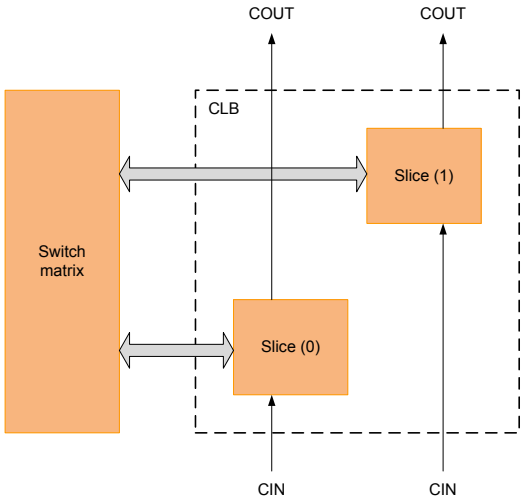

Figure A.3: Arrangement of slices within the CLB for Virtex-5 [146].

A.3.2 Block Random Access Memory (BRAM)

The BRAM base size in the Virtex-5 family has increased to 36-Kbits, from 18-Kbits in the Virtex-4 family. This makes it easier to build larger memory arrays in Virtex-5 devices. Moreover, the 36-Kbits block RAM can be configured as either two independent 18-Kbits RAMs, or one 36-Kbits RAM, hence, there is essentially no

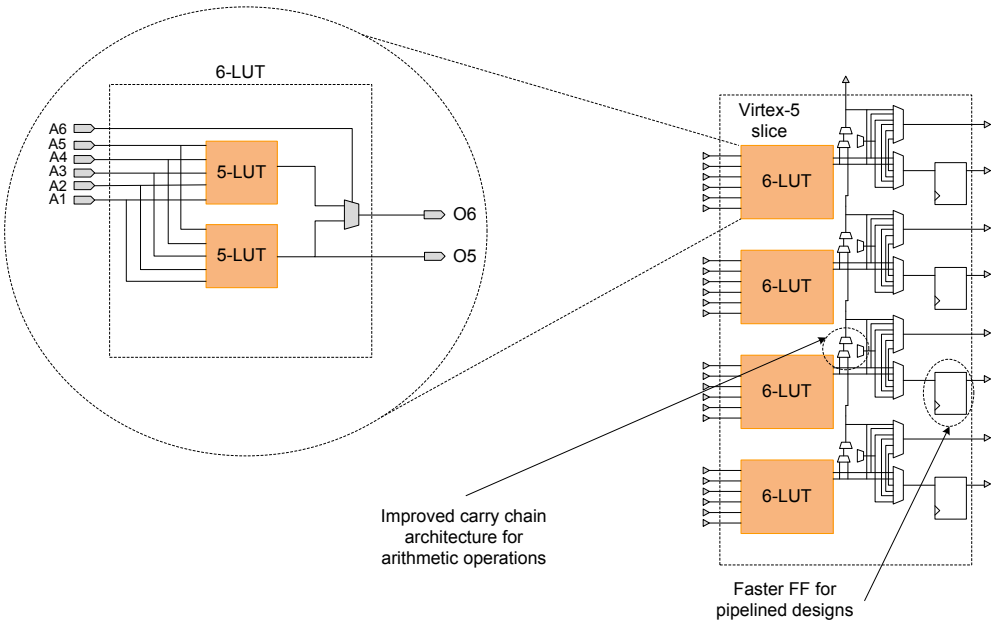

Figure A.4: Details of CLBs and slices for Virtex-5 [146].

penalty for building many 18-Kbits or smaller RAM arrays on-chip. The Virtex-5 family BRAM can be operated in simple dual port mode to effectively double the BRAM bandwidth. Simple dual port mode allows the Virtex-5 family BRAM width to be expanded beyond 32-bits up to as much as 72-bits per BRAM.

A.3.3 Digital Signal Processor (DSP) Element

The Virtex-5 DSP48E slice includes all Virtex-4 DSP48 features with a variety of new feature. With new cascade paths, the Virtex-5 DSP48E slice features with a wider 25×18 multiplier and an add/subtract function that has been extended to function as a logic unit. This logic unit can perform a host of bitwise logical operations when the multiplier is not used. In addition, the DSP48E slice includes a pattern detector and a pattern bar detector that can be utilised for convergent rounding, overflow/underflow detection for saturation arithmetic, and auto-resetting counters/accumulators. Moreover, the single instruction multiple data (SIMD) mode of the adder/subtractor/logic unit is also new feature to the DSP48E slice and this mode is available when the multiplier is not used [147].

A.4 Comparison

Based on the survey that has been carried for Virtex-E, Virtex-4, Virtex-5 and Spartan-3L FPGA, Table A.1 gives a brief summary and comparison between each device in terms of the CLB, BRAM, DSP and reduced instruction set computing (RISC) processor. In conclusion, the following notes can be made:

1. The BRAM is usually portioned in large arrays separate from the logic regions of the chip. It greatly increases FPGA processing capability by allowing for fast data storage. The increasing of RAM base size contributes to build larger memory arrays;

2. The multipliers on FPGA are highly optimised speciality blocks that greatly increase the speed and space efficiency of multiplication operations. The increasing of multiplier width leads to fewer cascade stages and yielding higher overall performance and utilisation; and

3. The 6-input LUT in Virtex-5 leads to several benefits: with wider functions directly in the LUT, the number of logic levels between registers is reduced and lead to better performance. Moreover, power consumption is also reduced as the larger LUT reduces the amount of required interconnect (routing resources).

Table A.1: Comparison of selected Xilinx FPGA devices resources.

	Parameter	Virtex-E	Virtex-4	Virtex-5	Spartan-3L
CLB	Slices	4	4	2	4
	LUTs	4	8	8	8
	Flip-flops	8	8	8	8
	Clocks, Clock-enables, Reset	N/A	four each	two each	four each
	Distributed RAM	64-bits	64-bits	256-bits	64-bits
	Shift Register Length	N/A	64-bits	128-bits	16-bits
	Multiplexers	N/A	16 - 1	2 × (16 - 1)	1 × (16 - 1)
BRAM	Base size	N/A	18-Kbits	36-Kbits	N/A
DSP	Modules	N/A	DSP48	DSP48E	N/A
	Multiplier width	N/A	18 × 18-bits	25 × 18-bit	18 × 18-bits
	Operating frequency	N/A	500 MHz	550 MHz	N/A
RISC processor		N/A	PowerPC	Power 440	N/A

Note:
N/A: Not applicable, PowerPC: IBM PowerPC RISC processor Core (FX only), Power440: Power440 microprocessors

Appendix B

Xilinx ISE and FPGA Programming

B.1 Overview

Design procedures involved in field programmable gate array (FPGA)-based logic design can be summarised as shown in Figure B.1. Generally, five steps involved:

1. Design description:

   Two main strategies can be used: a hardware description language (HDL), such as very-high-speed integrated circuit hardware description language (VHDL) or Verilog, and a schematic editor. In the case of Xilinx tools particularly, CoreGen can be also instantiated for both design descriptions, either with an HDL or a schematic editor.

2. Netlist generation:

   Netlist can be defined as a description of the various logic gates in the design and represents their interconnection. Logic synthesis program is used to transform the design description into a netlist.

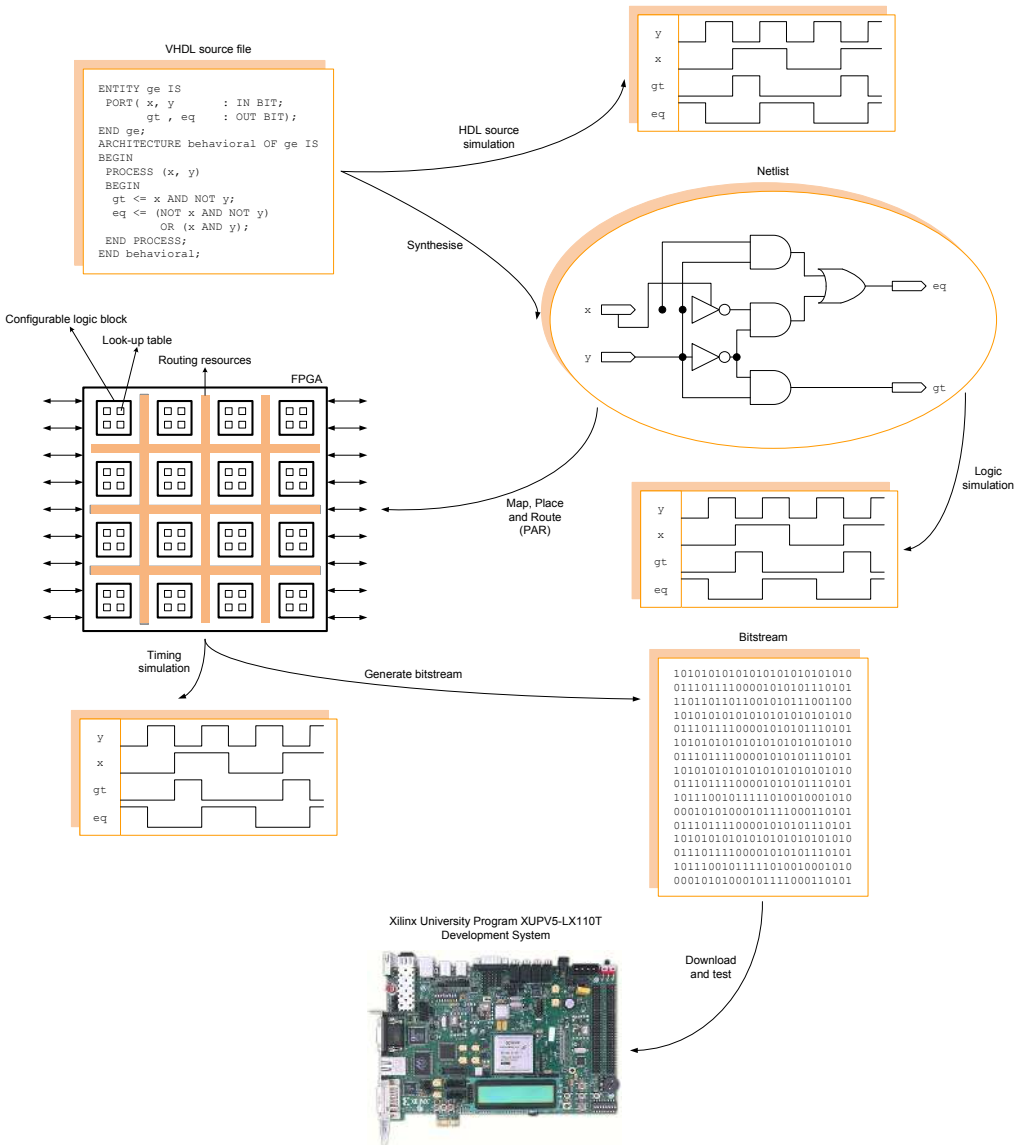

Figure B.1: General design route from VHDL to prototyping board.

3. Mapping and routing:

   An FPGA topology is structured by configurable logic blocks (CLBs), which can be further decomposed into look-up tables (LUTs) that perform logic operations. The CLBs and LUTs are interwoven with various routing resources. The mapping tool acts to collect the netlist gates into groups that fit into the LUTs, and then place and route (PAR) tool assigns the groups to specific CLBs while opening or closing the switches in the routing matrices to connect them together.

4. Bitstreams generation:

   Once the implementation phase is complete, a program extracts the state of the switches in the routing matrices and generates a bitstream, with the ones and zeroes correspond to open or close switches.

5. Rapid prototyping:

   The bitstream is downloaded into a physical FPGA chip and the electronic switches in the FPGA open or close in response to the binary bits in the bitstream. Upon completion of the downloading, the FPGA will perform the operations based on the design description specified by an HDL code or a schematic.

B.2 Implementing VHDL Design

This section gives an overview of the VHDL design implementation in FPGA chips using Xilinx ISE Design Suite 10.1 and ModelSim Xilinx Edition (XE) III 6.4b.

B.2.1 Xilinx ISE

ISE design suite controls all aspects of the design flow for Xilinx FPGAs. Through the `Project Navigator` as shown in Figure B.2, all the design entry and design implementation tools can be accessed. To illustrate the simulation results, Modelsim as shown in Figure B.3 has been used. In the following, different important steps are described.

Specifying Design Options

The `Design Goals and Strategies Editor` as shown in Figure B.4 allows the design goal and strategies to be defined. A series of categories, each containing property for a different aspect of design implementation can be seen.

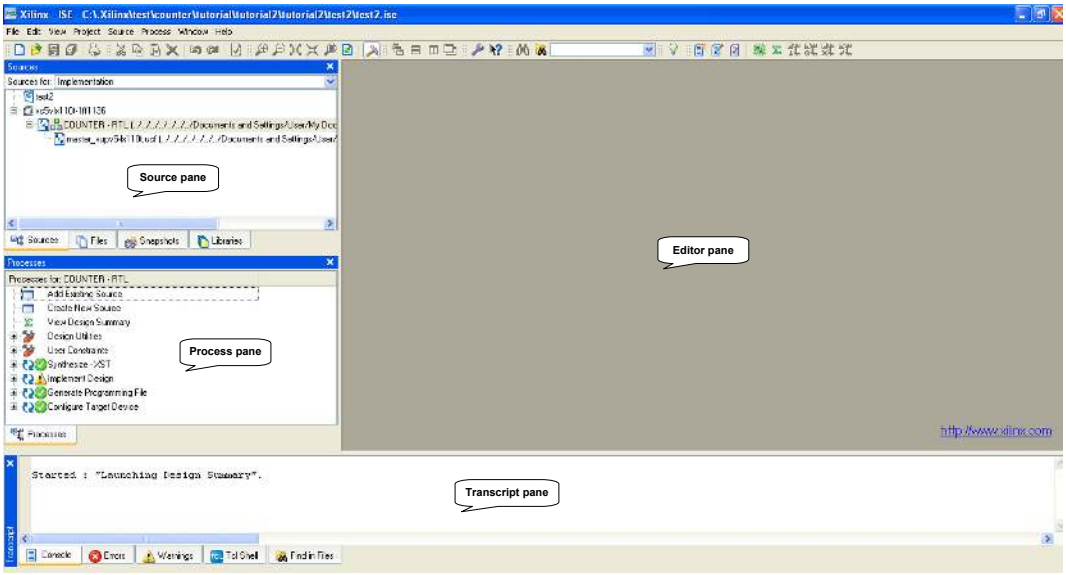

Figure B.2: Sample window displaying ISE `project navigator`.

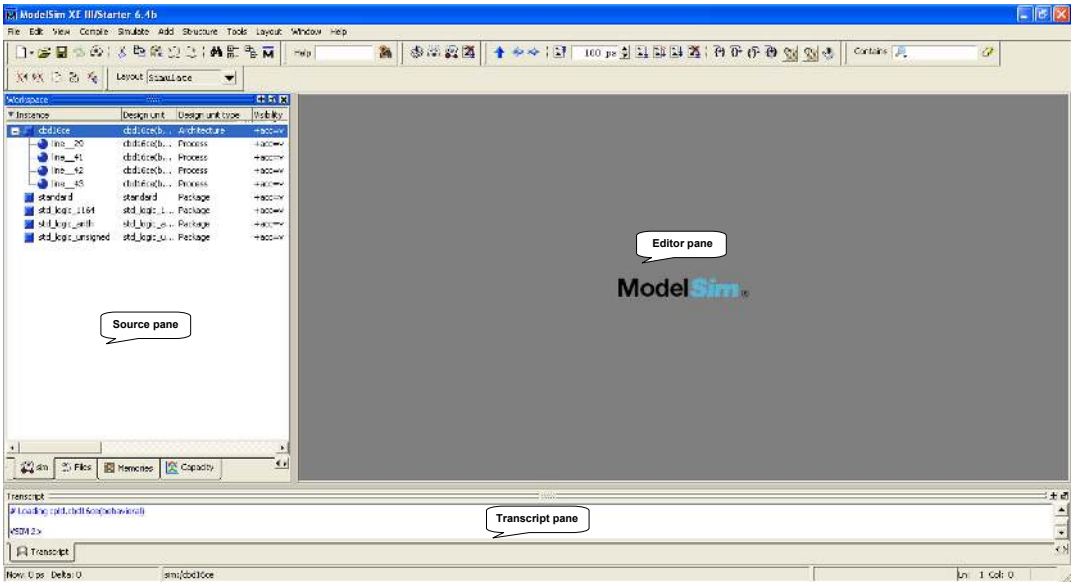

Figure B.3: ModelSim simulator window.

Design Translation

During translation, the NGDBuild program performs the following functions:

1. Converts input design netlists and writes results to a single merged NGD netlist. The merged netlist describes the logic in the design as well as any location and timing constraints;

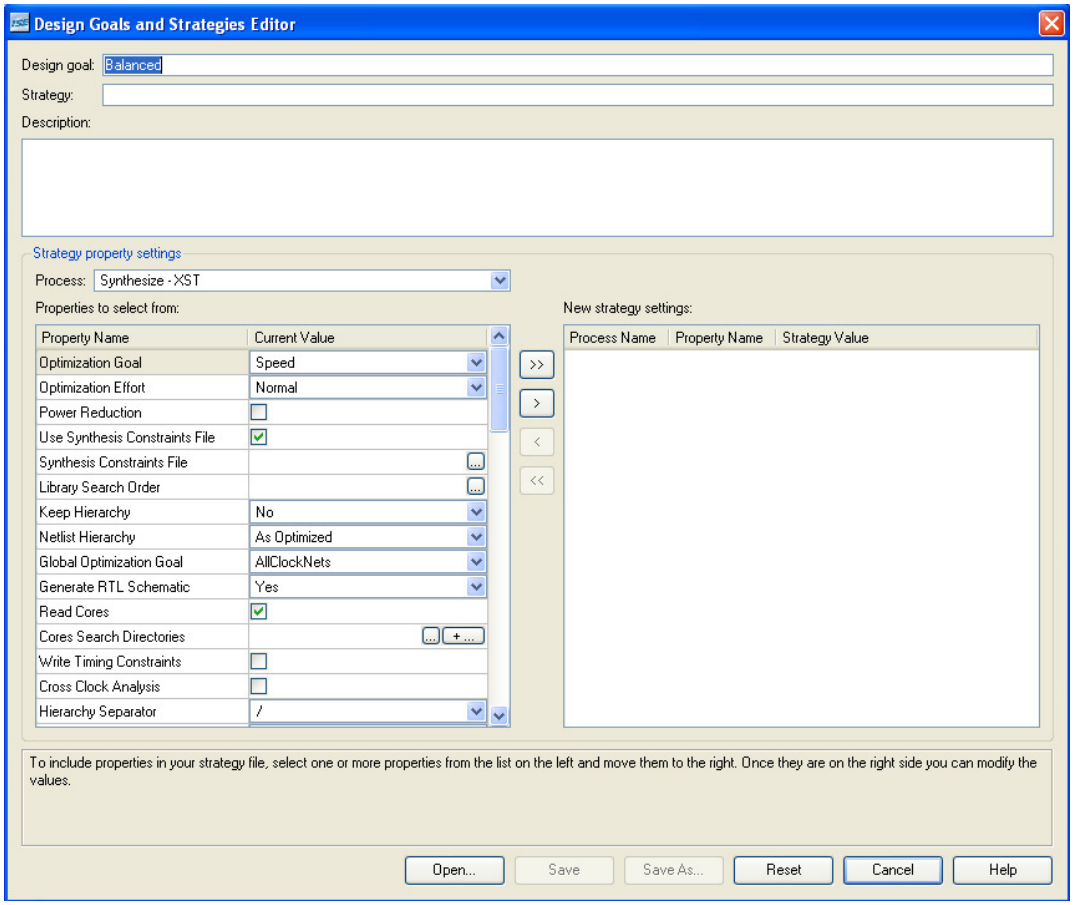

Figure B.4: Setting the design options in ISE.

2. Performs timing specification and logical design rule checks; and

3. Adds the user constraints file (UCF) to the merged netlist.

Timing Constraints

The UCF provides a mechanism for constraining a logical design without returning to the design entry tools. The `Constraints Editor` and `Floorplan` are tools that enable entry of timing and pin location constraints as shown in Figure B.5 and B.6, respectively.

```
1  NET  CLK                LOC="AG18";  # Bank 4, Vcco=3.3V, No DCI
2  NET  counter<0>         LOC="H18";   # Bank 3, Vcco=2.5V, No DCI
3  NET  counter<1>         LOC="L18";   # Bank 3, Vcco=2.5V, No DCI
4  NET  counter<2>         LOC="G15";   # Bank 3, Vcco=2.5V, No DCI
5  NET  counter<3>         LOC="AD26";  # Bank 21, Vcco=1.8V, DCI using 49.9 ohm resistors
6  NET  rst                LOC="AJ6";   # Bank 18, Vcco=3.3V, No DCI GPIO_SW_C
7  NET  count              LOC="AK7";   # Bank 18, Vcco=3.3V, No DCI GPIO_SW_E
```

Figure B.5: Setting for UCF.

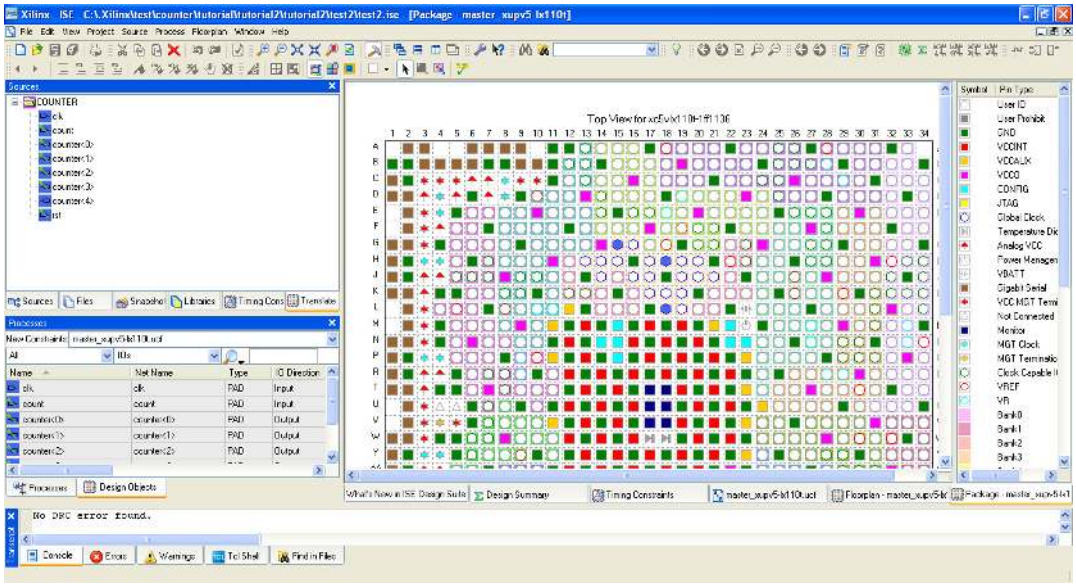

Figure B.6: Floorplan for pin location constraints.

Mapping, Place and Route (PAR)

Since all implementation strategies have been defined (properties and constraints), the next design step of mapping is performed. After the mapped design is evaluated, the design can be placed and routed. One of two PAR algorithms is performed during the PAR process:

1. Timing driven PAR is run with the timing constraints specified in the input netlist and/or in the constraints file; and

2. Non-timing driven PAR is run, ignoring all timing constraints.

Verification of Place and Route (PAR)

The FPGA editor as shown in Figure B.7 reads and writes native circuit description (NCD) files, native macro circuit (NMC) files, and physical constraints files (PCF). It performs the following tasks:

1. PAR critical components before running the automatic PAR tools;
2. Finish placement and routing, if the routing program does not completely route your design;
3. Add probes to the design to examine the signal states of the targeted device;
4. View and change the nets connected to the capture units of an integrated logic analyser (ILA) core in the design;
5. Run the BitGen program and download the resulting bitstream file to the targeted device; and
6. View and change the nets connected to the capture units of an ILA core in your design.

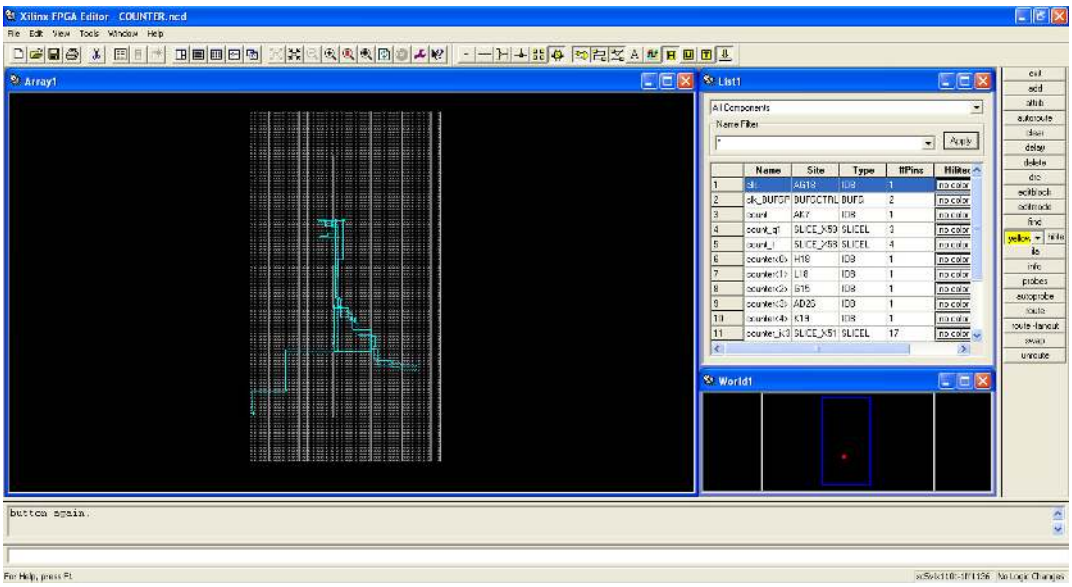

Figure B.7: FPGA editor window.

B.2.2 Field Programmable Gate Array (FPGA) Configuration

To properly synthesise the design, pins definition on the chip for all the inputs and outputs are required. In general, pin assignment could be assigned without restriction, but for any specific prototyping board it is required to make sure that the pin assignments match the switches, buttons, light emitting diodes (LEDs) and so on.

To download the bitstream generated into the FPGA, configure a device (iMPACT) underneath the `Generate Programming File` selection needs to be selected and results should be illustrated as shown in Figure B.8 and B.9, respectively. After all, test and validation of the design implementation can be carried out on the prototyping board using LEDs or other peripheral options as shown in Figure B.10.

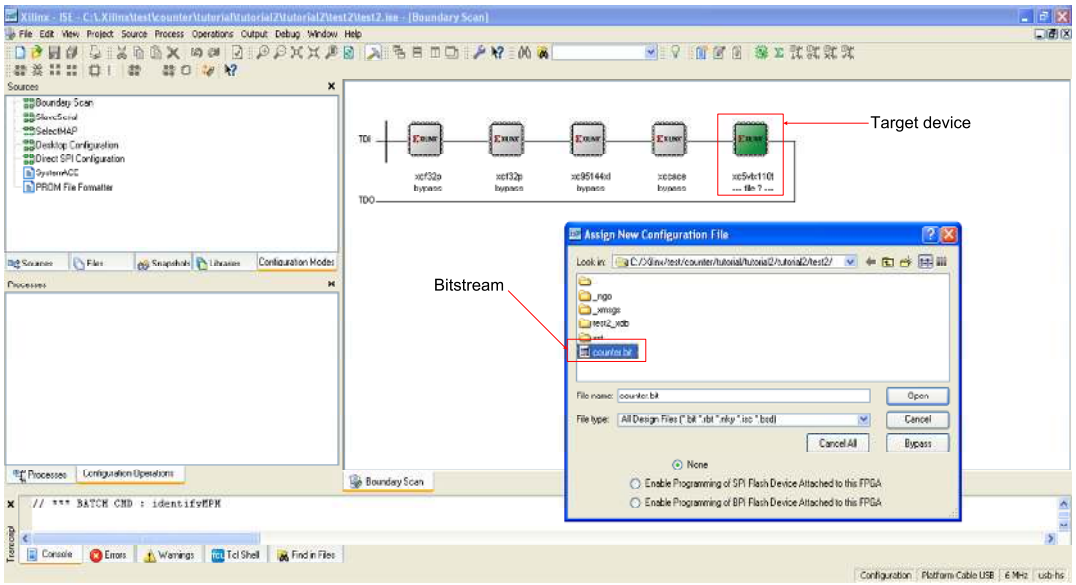

Figure B.8: Device configuration using iMPACT.

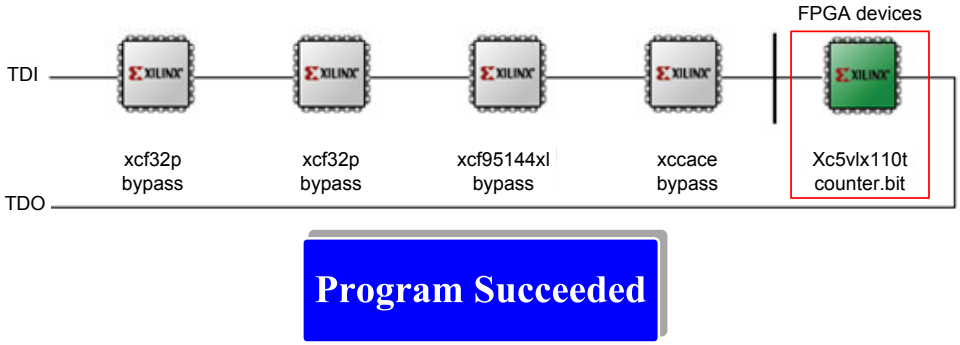

Figure B.9: Program succeeded to be downloaded.

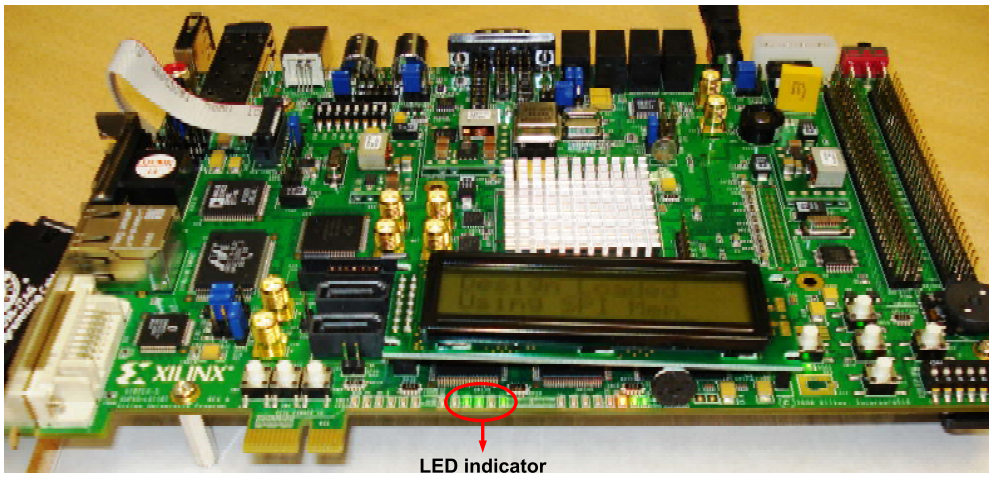

Figure B.10: Results verification using LEDs indicator.

Appendix C

Partial Reconfiguration (PR) in Xilinx FPGA Devices

C.1 Overview

To date, a technology called partial reconfiguration (PR) has been widely applied with several Xilinx field programmable gate arrays (FPGAs). PR is a concept that manipulates the FPGA technology flexibility as it allows modification to be carried out of an operating FPGA design by loading a partial bit file [148]. As illustrated in Figure C.1, the logic in the FPGA is defined into two different types: reconfigurable and static logic.

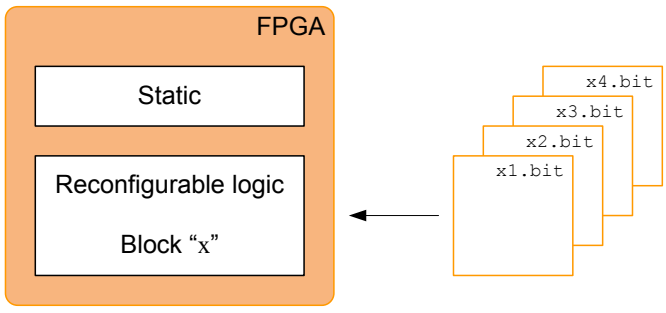

Figure C.1: Basic concept of partial reconfiguration.

With PR concept, the static logic remains functioning, and it is completely unaffected by the loading of a partial bit file, while the reconfigurable logic is replaced by the contents of the partial bit file. The function to be implemented in reconfigurable logic block "x" is modified by downloading one of the several partial bit files, `x1.bit`, `x2.bit`, `x3.bit` and `x4.bit`. There are many reasons of PR to be fully deployed on a single FPGA device is advantageous. These include:

1. Area:

   Reducing the size of an FPGA device required for implementing a given function with consequent reductions in cost and power consumption; and

2. Flexibility:

   Providing flexibility in the choices of algorithms or protocols available to an application.

C.2 Design Requirements

Xilinx design suite accommodates all design tools to execute any PR project. In this study, Xilinx service pack 9.2i for PR tools has been used. Summary of the tools required for PR project execution is depicted in Figure C.2.

C.3 Implementation Design Flow

A general modular design flow for PR project implementation is given in Figure C.3. It is worth noting that the process of implementing a partially reconfigurable FPGA design is similar to implementing multiple non-PR designs that share common logic [148]. However, design partitions are to ensure that the common logic between the multiple designs is identical. Figure C.4 illustrates this concept and gives an overview of design structure and file formats. Moreover, Table C.1 provides a list of the useful file that needs to be handled throughout the PR execution.

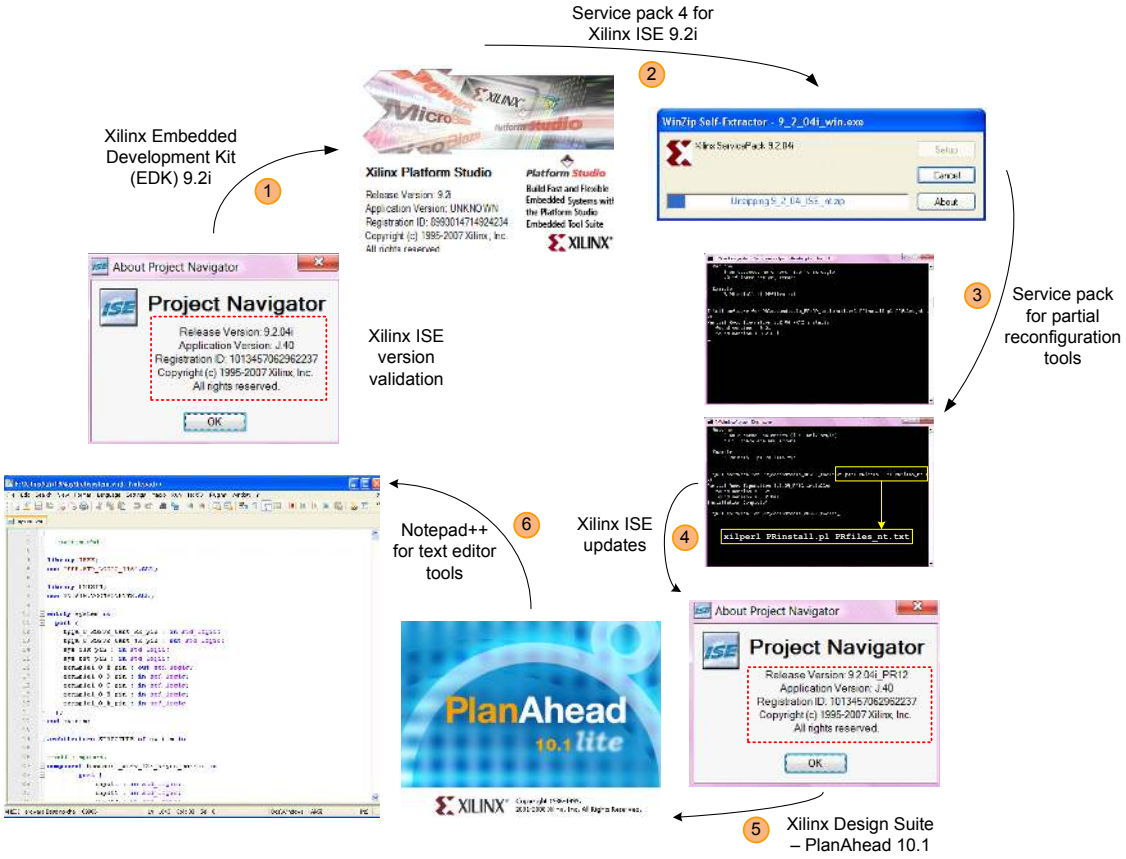

Figure C.2: Design tools requirement in PR.

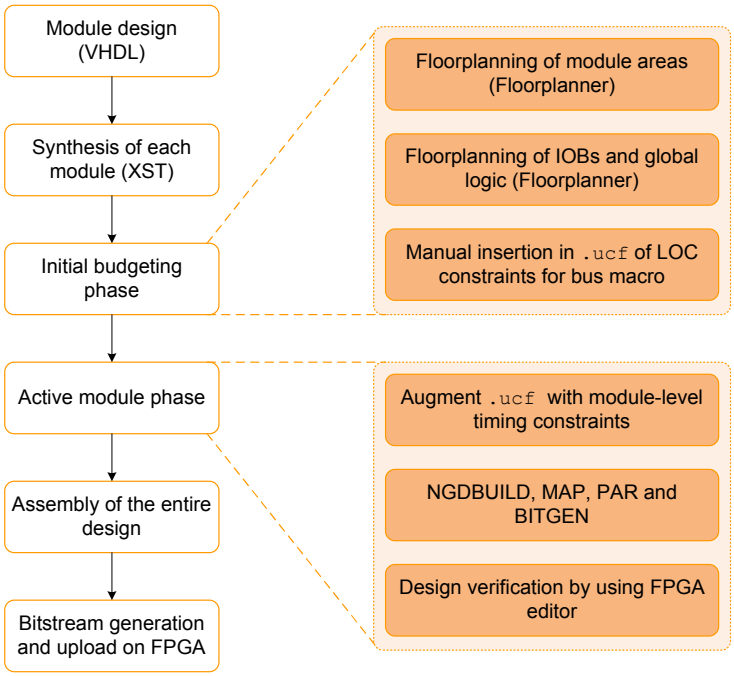

Figure C.3: General PR design flow.

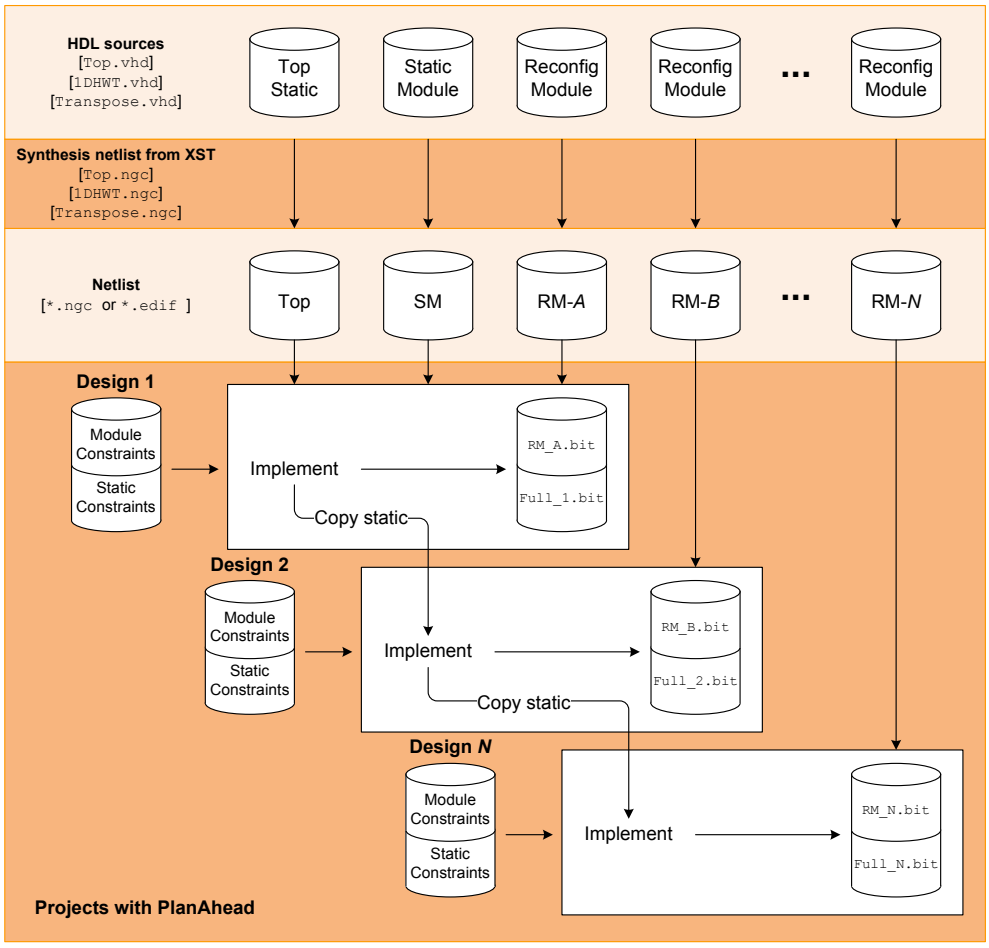

Figure C.4: Overview of PR software design flow.

Table C.1: Description of files format for PR process.

Extension	Description
*.vhd	VHDL design file and serve as input for a synthesis tool like Xilinx XST.
*.ngc	XST netlist and this file is the result of a synthesis using XST.
*.ngd	Design file. This binary file contains a logical description of the design in terms of both its original components and hierarchy and the NGD primitives to which the design is reduced.
*.pcf	Physical constraints file. An ASCII text file containing the constraints specified during design entry expressed in terms of physical elements.
*.ncd	Native circuit description. A physical description of a of the design terms of the components in the target Xilinx device.
*.par	PAR report including summary information of all placement and routing iterations. original components and hierarchy and the NGD primitives to which the design is reduced.
*.pad	A file containing input/output (I/O) pin assignments in a parsable database format.
*.bit	A binary file that contains proprietary header information as well as configuration data.
*.msk	A binary file that contains the same configuration commands as a *.bit file, but has mask data and routing iterations. The configuration data is used for verification purpose.

Appendix D

Xilinx AccelDSP Synthesis Tool

D.1 Overview

AccelDSP is one of the digital signal processing (DSP) synthesis tool by Xilinx with an easy-to-use graphical user interface (GUI) that provides a promising solution to transform a MATLAB floating-point design into a hardware module that can be implemented on a Xilinx field programmable gate array (FPGA). The AccelDSP tool is also integrated with other design tools such as MATLAB, Xilinx ISE tools and other industry-standard hardware description language (HDL) simulators and logic synthesisers [124].

D.2 Design Flow and Operations

An overview of features and advantages offered by AccelDSP synthesis tool is given in Figure D.1.

To implement the project, two "`.m`" files. required: script and function m-file. The script m-file is used to apply stimulus and plot results, whilst the function m-file is to realise the function of the project.

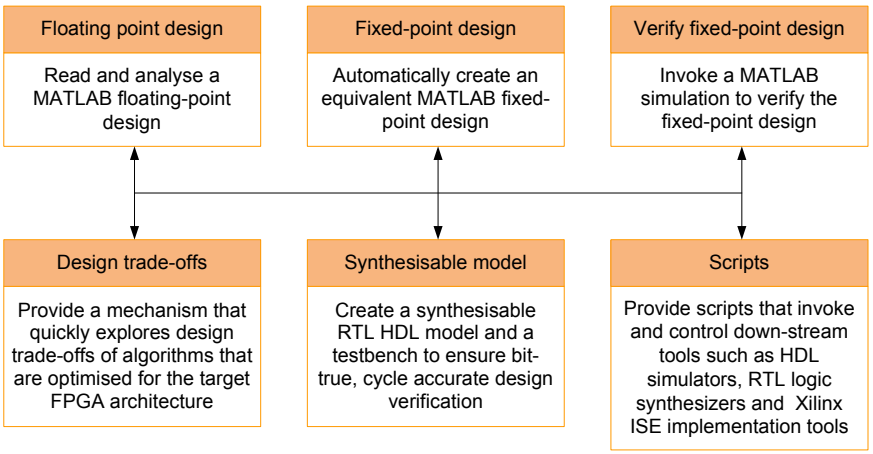

Figure D.1: Advantages of AccelDSP synthesis tool.

In terms of design flow, verification is made in several steps in the AccelDSP work flow and an overview of the work flow is given in Figure D.2.

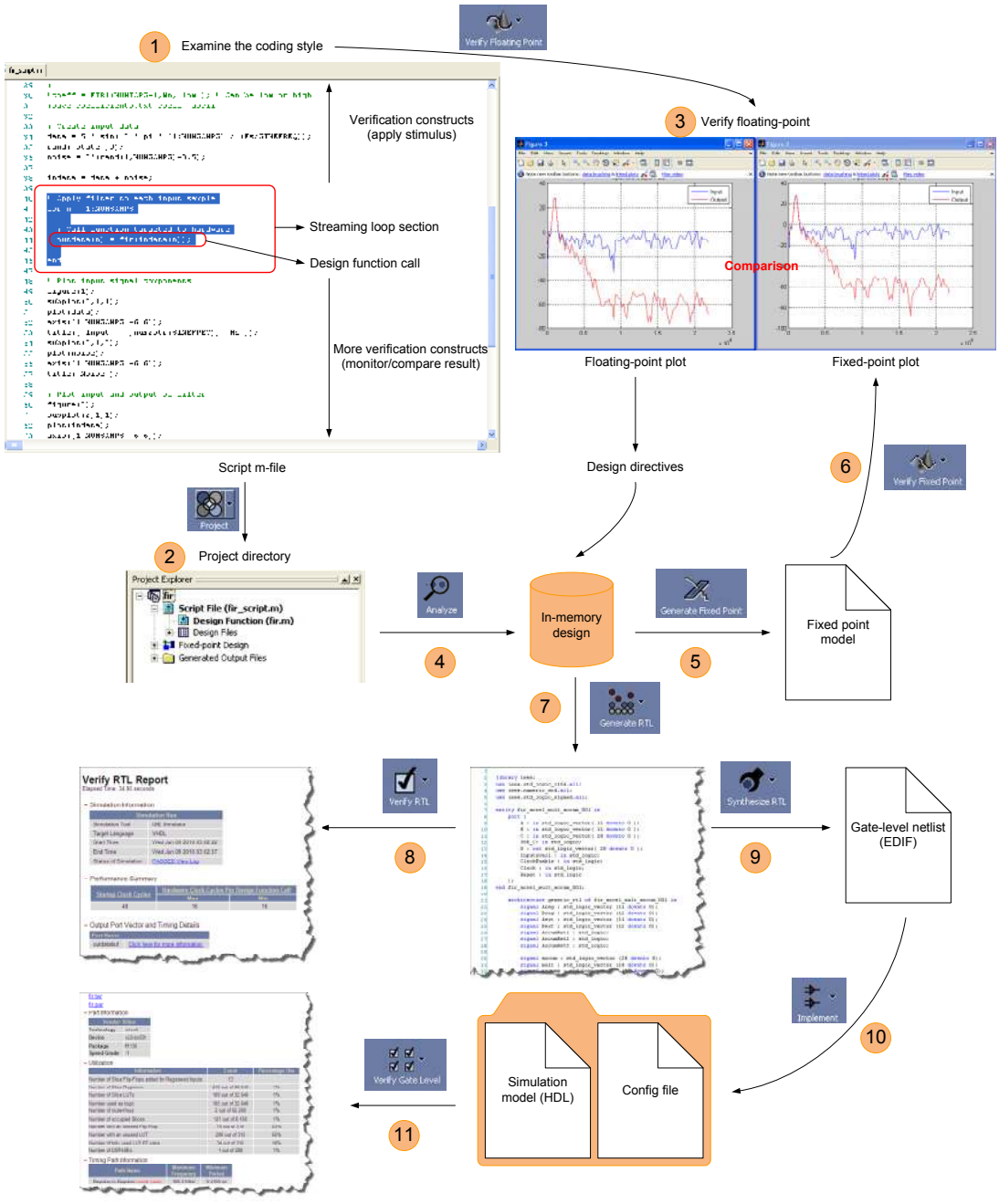

Figure D.2: The AccelDSP ISE synthesis work flow.

Bibliography

[1] H. Lee, Y. Kim, A. H. Rowberg, and E. A. Riskin, "Statistical distributions of DCT coefficients and their application to an interframe compression algorithm for 3-D medical images," *Medical Imaging, IEEE Transactions on*, vol. 12, no. 3, pp. 478–485, Sept. 1993.

[2] [Online], "Accessed 20 June 2009," http://info.cancerresearchuk.org/cancerstats/incidence/index.htm.

[3] G. Sakas, "Trends in medical imaging: from 2D to 3D," *Computers & Graphics*, vol. 26, no. 4, pp. 577–587, 2002.

[4] M. J. Zukoski, T. Boult, and T. Iyriboz, "A novel approach to medical image compression," *Int. J. Bioinformatics Research and Applications*, vol. 2, no. 1, pp. 89–103, 2006.

[5] D. W. G. Montgomery, "Multiscale compression and segmentation of volumetric oncological PET imagery," PhD Thesis, School of Computer Science, The Queen's University of Belfast, 2006.

[6] R. Shams, P. Sadeghi, R. Kennedy, and R. Hartley, "A survey of medical image registration on multicore and the GPU," *Signal Processing Magazine, IEEE*, vol. 27, no. 2, pp. 50–60, Mar. 2010.

[7] J. Wang and H. K. Huang, "Three-dimensional medical image compression using a wavelet transform with parallel computing," *Medical Imaging 1995: Image Display*, vol. 2431, no. 1, pp. 162–172, 1995.

[8] J. Jyotheswar and S. Mahapatra, "Efficient FPGA implementation of DWT and modified SPIHT for lossless image compression," *J. Syst. Archit.*, vol. 53, no. 7, pp. 369–378, 2007.

[9] M. Jiang and D. Crookes, "Area-efficient high-speed 3D DWT processor architecture," *IEE Electronics Letter*, vol. 43, pp. 502–503, 2007.

[10] ——, "FPGA implementation of 3D discrete wavelet transform for real-time medical imaging," in *Circuit Theory and Design (ECCTD 2007), Proc. 18th European Conf. on*, Seville, Spain, 2007, pp. 519–522.

[11] P. Dang, "VLSI architecture for real-time image and video processing systems," *Journal of Real-Time Image Processing*, vol. 1, pp. 57–62, Oct. 2006.

[12] M. Papadonikolakis, C. S. Bouganis, and G. Constantinides, "Performance comparison of GPU and FPGA architectures for the SVM training problem," in *Field-Programmable Technology, 2009 (FPT 2009), Proceedings of International Conference on*, Dec. 2009, pp. 388–391.

[13] M. L. Stokes, "A brief look at FPGAs, GPUs and cell processors," *Journal of The International Test and Evaluation Association (ITEA)*, pp. 9–11, June/July 2007.

[14] T. Todman, G. Constantinides, S. Wilton, O. Mencer, W. Luk, and P. Cheung, "Reconfigurable computing: architectures and design methods," *Computers and Digital Techniques, IEE Proceedings*, vol. 152, no. 2, pp. 193–207, Mar. 2005.

[15] S. Brown and J. Rose, "Architecture of FPGAs and CPLDs: A tutorial," *IEEE Design and Test of Computers*, vol. 13, no. 2, pp. 42–57, 1996.

[16] [Online], "Accessed 20 Jan 2008," http://www.xilinx.com/publications/archives/xcell/Xcell-customer-innovation-2010.pdf.

[17] C. Maxfield, *The Design Warrior's Guide to* FPGA*s: Devices, Tools and Flows*. Elsevier, 2004.

[18] F. Bensaali, "Accelerating matrix product on reconfigurable hardware for image processing applications," PhD Thesis, School of Computer Science, The Queen's University of Belfast, 2005.

[19] I. S. Uzun, "Design and FPGA implementation of matrix transforms for image and video processing," PhD Thesis, School of Computer Science, The Queen's University of Belfast, 2006.

[20] S. Chandrasekaran, "Efficient FPGA implementation and power modelling of image and signal processing IP cores," PhD Thesis, School of Engineering and Design, Brunel University, West London, 2007.

[21] J. S. Duncan and N. Ayache, "Medical image analysis: progress over two decades and the challenges ahead," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 1, pp. 85–106, Jan. 2000.

[22] [Online], "Accessed 10 Feb. 2009," http://www.rsna.org.

[23] S. Muraki and Y. Kita, "A survey of medical applications of 3D image analysis and computer graphics," *Syst. Comput. Japan*, vol. 37, no. 1, pp. 13–46, 2006.

[24] R. S. Lazebnik and T. S. Desser, "Clinical 3D ultrasound imaging: beyond obstetrical applications," *Diagnostic Imaging: Continuing Medical Education*, pp. 1–6, 2009.

[25] A. Kurjak, B. Miskovic, W. Andonotopo, M. Stanojevic, G. Azumendi, and H. Vrcic, "How useful is 3D and 4D ultrasound in perinatal medicine?" *J. Perinat. Med.*, vol. 35, pp. 10–27, 2007.

[26] M. R. Stytz, G. Frieder, and O. Frieder, "Three-dimensional medical imaging: algorithms and computer systems," *ACM Comput. Surv.*, vol. 23, no. 4, pp. 421–499, 1991.

[27] G. D. Michailidis, P. Papageorgiou, and D. L. Economides, "Assessment of fetal anatomy in the first trimester using two- and three-dimensional ultrasound," *The British Journal of Radiology*, vol. 75, no. 891, pp. 215–219, 2002.

[28] E. L. Ritman, "Evolution of medical tomographic imaging - as seen from a Darwinian perspective," *Proceedings of the IEEE*, vol. 91, no. 10, pp. 1483–1491, Oct. 2003.

[29] S. C. Orphanoudakis, "Supercomputing in medical imaging," *Engineering in Medicine and Biology Magazine, IEEE*, vol. 7, no. 4, pp. 16–20, Dec. 1988.

[30] J. Coatrieux, C. Toumoulin, C. Hamon, and L. Luo, "Future trends in 3D medical imaging," *Engineering in Medicine and Biology Magazine, IEEE*, vol. 9, no. 4, pp. 33–39, Dec. 1990.

[31] J. Maintz and M. A. Viergever, "A survey of medical image registration," *Medical Image Analysis*, vol. 2, no. 1, pp. 1–36, 1998.

[32] D. Z. Tian and M. H. Ha, "Applications of wavelet transform in medical image processing," in *Machine Learning and Cybernetics, Proceedings of International Conference on*, Baoding, China, 2004, pp. 1816–1821.

[33] A. K. Rath and P. K. Meher, "Design of a merged DSP microcontroller for embedded systems using discrete orthogonal transform," *Journal of Computer Science*, vol. 2, no. 5, pp. 388–394, 2006.

[34] A. Bensrhair, N. Chafiqui, and P. Miche, "Implementation of a 3D vision system on DSPs TMS320C31," *Journal of Real Time Image Processing*, vol. 6, no. 3, pp. 213–221, June 2000.

[35] C. Basoglu, W. Lee, and J. O'Donnell, "The equator MAP-CA(TM)DSP: An end-to-end broadband signal processor (TM) VLIW," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 12, no. 8, pp. 646–659, Aug. 2002.

[36] J. Furtler, K. J. Mayer, W. Krattenthaler, and I. Bajla, "SPOT: Development tool for software pipeline optimization for VLIW-DSPs used in real-time image processing," *Journal of Real Time Image Processing*, vol. 9, no. 6, pp. 387–399, Dec. 2003.

[37] D. Jinghong, D. Yaling, and L. Kun, "Development of image processing system based on DSP and FPGA," in *Electronic Measurement and Instruments, 2007. ICEMI '07. 8th International Conference on*, Aug. 2007, pp. 791–794.

[38] Y. Zou, G. Shi, Y. Jin, and Y. Zheng, "Extraocular image processing for retinal prosthesis based on DSP," in *Nano/Micro Engineered and Molecular Systems, 2009. NEMS 2009. 4th IEEE International Conference on*, 2009, pp. 563–566.

[39] K. Hang, "Real-time image acquisition and processing system design based on DSP," in *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on*, vol. 4, 2010, pp. 492–496.

[40] R. Kordasiewicz and S. Shirani, "ASIC and FPGA implementations of H.264 DCT and quantization blocks," in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. 3, 2005, pp. III–1020–3.

[41] L. Liu, H. Meng, and M. Zhang, "An ASIC implementation of lifting-based 2-D discrete wavelet transform," in *Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on*, 2006, pp. 271–274.

[42] S. Mittal, Z. Khan, and M. Srinivas, "Area efficient high speed architecture of Bruun's FFT for software defined radio," in *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, 2007, pp. 3118–3122.

[43] Y. W. Tsai, K. C. Wu, H. H. Tung, and R. B. Lin, "Using structured ASIC to improve design productivity," in *Integrated Circuits, ISIC '09. Proceedings of the 2009 12th International Symposium on*, 2009, pp. 25–28.

[44] H. K. Phoon, M. Yap, and C. K. Chai, "A highly compatible architecture design for optimum FPGA to structured-ASIC migration," in *Semiconductor Electronics, 2006. ICSE '06. IEEE International Conference on*, Oct. 2006, pp. 506–510.

[45] B. Zahiri, "Structured ASICs: Opportunities and challenges," in *Computer Design, Proceedings of the International Conference on*, Oct. 2003, pp. 404–409.

[46] D. Gddeke, "Fast and accurate finite-element multigrid solvers for PDE simulations on GPU clusters," PhD Thesis, Institute of Applied Mathematics, Technical University Dortmund, 2010.

[47] D. B. Thomas, L. Howes, and W. Luk, "A comparison of CPUs, GPUs, FPGAs, and massively parallel processor arrays for random number generation," in *Field programmable gate arrays FPGA '09, Proceeding of the ACM/SIGDA international symposium on*, 2009, pp. 63–72.

[48] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krger, A. E. Lefohn, and T. J. Purcell, "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum*, vol. 26, no. 1, pp. 80–113, 2007.

[49] K. Compton and S. Hauck, "Reconfigurable computing: A survey of systems and software," *ACM Comput. Surv.*, vol. 34, no. 2, pp. 171–210, June 2002.

[50] A. Ahmad, B. Krill, A. Amira, and H. Rabah, "Efficient architectures for 3D HWT using dynamic partial reconfiguration," *Journal of Systems Architecture*, vol. 56, no. 8, pp. 305–316, Aug. 2010.

[51] P. C. Cosman, R. M. Gray, and R. A. Olshen, "Evaluating quality of compressed medical images: SNR, subjective rating, and diagnostic accuracy," *Proceedings of the IEEE*, vol. 82, no. 6, pp. 919–932, Jun. 1994.

[52] S. Ghrare, M. Ali, M. Ismail, and K. Jumari, "Diagnostic quality of compressed medical images: Objective and subjective evaluation," in *Modeling Simulation,*

2008. AICMS 08. Second Asia International Conference on, May 2008, pp. 923–927.

[53] J. Wang and H. K. Huang, "Medical image compression by using three-dimensional wavelet transformation," *Medical Imaging, IEEE Transactions on*, vol. 15, no. 4, pp. 547–554, Aug. 1996.

[54] H. B. Cattin, A. Baskurt, F. Turjman, and R. Prost, "3D medical image coding using separable 3D wavelet decomposition and lattice vector quantization," *Signal Processing*, vol. 59, no. 2, pp. 139–153, 1997.

[55] X. Qi and J. M. Tyler, "A progressive transmission capable diagnostically lossless compression scheme for 3D medical image sets," *Information Sciences*, vol. 175, no. 3, pp. 217–243, 2005.

[56] R. S. Sunder, C. Eswaran, and N. Sriraam, "Medical image compression using 3-D Hartley transform," *Computers in Biology and Medicine*, vol. 36, no. 9, pp. 958–973, 2006.

[57] N. P. Raj and T. Venkateswarlu, "A novel approach to medical image compression using sequential 3D DCT," in *Computational Intelligence and Multimedia Applications, 2007. International Conference on*, vol. 3, Dec. 2007, pp. 146–152.

[58] G. Bernabe, J. M. Garcia, and J. Gonzalez, "A loosy 3D wavelet transform for high-quality compression of medical video," *The Journal of Systems and Software*, vol. 82, pp. 526–534, 2009.

[59] Y. Gaudeau and J. M. Moureaux, "Lossy compression of volumetric medical images with 3D dead zone lattice vector quantization," *Annals of Telecommunications*, vol. 64, no. 5-6, May 2009.

[60] Z. Xiong, X. Wu, S. Cheng, and J. Hua, "Lossy-to-lossless compression of medical volumetric data using three-dimensional integer wavelet transforms," *Medical Imaging, IEEE Transactions on*, vol. 22, no. 3, pp. 459–470, Mar. 2003.

[61] M. C. Weeks, "Architectures for 3-D discrete wavelet transform," PhD Thesis, Department of Computer Engineering, University of Southwestern Louisiana, Spring 1998.

[62] S. Saha, "Image compression—from DCT to wavelets: a review," *Crossroads*, vol. 6, no. 3, pp. 12–21, 2000.

[63] M. Weeks and M. Bayoumi, "3D discrete wavelet transform architectures," in *Circuits and Systems (ISCAS '98), Proceedings of the IEEE International Symposium on*, 1998, pp. 57–60.

[64] ——, "3-D discrete wavelet transform architectures," *Signal Process. IEEE Trans. on*, vol. 50, pp. 2050–2063, Aug. 2002.

[65] ——, "Discrete wavelet transform: Architectures, design and performance issues," *The Journal of* VLSI *Sig. Process.*, vol. 35, pp. 155–178, 2003.

[66] B. Das and S. Banerjee, "A memory efficient 3-D DWT architecture," in *Acoustics, Speech, and Signal Processing, Proc. Conference IEEE International Conference on*, Florida, USA, May 2002, pp. 3224–3227.

[67] S. Ismail, A. Salama, and M. Abu-ElYazee, "FPGA implementation of an efficient 3D-WT temporal decomposition algorithm for video compression," in *Signal Processing and Information Technology, Proc. Conference IEEE International Symposium on*, Cairo, Egypt, Dec. 2007, pp. 154–159.

[68] M. A. M. Salem, M. Appel, F. Winkler, and B. Meffert, "FPGA-based smart camera for 3D wavelet-based image segmentation," in *Distributed Smart Cameras (ICDSC 2008), Proc. Second ACM/IEEE Int. Conf. on*, California, USA, 2008, pp. 1–8.

[69] G. Zhang, M. Talley, W. Badawy, M. Weeks, and M. Bayoumi, "A low power prototype for a 3-D discrete wavelet transform processor," in *Circuits and*

Systems (ISCAS '99), Proc. Conference IEEE International Symposium on, Florida, USA, May 1999, pp. 145–148.

[70] S. Chandrasekaran and A. Amira, "Novel sparse OBC based distributed arithmetic architecture for matrix transforms," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, 2007, pp. 3207–3210.

[71] S. Chandrasekaran, A. Amira, S. Minghua, and A. Bermak, "An efficient VLSI architecture and FPGA implementation of the finite ridgelet transform," *Journal of Real-Time Image Processing*, vol. 3, no. 3, pp. 183–193, Sept. 2008.

[72] I. S. Uzun and A. Amira, "Design and FPGA implementation of finite ridgelet transform," in *Circuits and Systems (ISCAS), 2005. Proceedings of the IEEE International Symposium on*, vol. 6, May 2005, pp. 5826–5829.

[73] J. Wisinger and R. Mahapatra, "FPGA based image processing with the curvelet transform," Texas A & M University, TX, *Technical Report* TR-CS-2003-01-0, 2003.

[74] F. Matus and J. Flusser, "Image representation via a finite Radon transform," *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, vol. 15, pp. 996–1006, Oct. 1993.

[75] C. A. Rahman and W. Badawy, "Architectures the finite Radon transform," *IEE Electronic Letters*, vol. 40, no. 15, pp. 931–932, July 2004.

[76] S. Chandrasekaran and A. Amira, "High speed/low power architectures for the finite Radon transform," in *Field Programmable Logic and Applications, 2005. Proceedings of International Conference on*, Aug. 2005, pp. 450–455.

[77] D. Zha and T. Qiu, "A new algorithm for shot noise removal in medical ultrasound images based on alpha-stable model," *Int. J. Adapt. Control Signal Process.*, vol. 20, pp. 251–263, 2006.

[78] Y. X. Liu, Y. H. Peng, and W. C. Siu, "Energy-based adaptive transform scheme in the DPRT domain and its application to image denoising," *Signal Process.*, vol. 89, no. 1, pp. 31–44, 2009.

[79] L. Guo, O. C. Au, M. Ma, and Z. Liang, "Fast multi-hypothesis motion compensated filter for video denoising," *Journal of Signal Processing Systems*, vol. 60, no. 3, pp. 273–290, 2010.

[80] G. Y. Chen and B. Kgl, "Image denoising with complex ridgelets," *Pattern Recognition*, vol. 40, no. 2, pp. 578–585, 2007.

[81] J. Sanches, J. Nascimento, and J. Marques, "Medical image noise reduction using the Sylvester Lyapunov equation," *Image Processing, IEEE Transactions on*, vol. 17, no. 9, pp. 1522–1539, Sept. 2008.

[82] D. Zhang and T. Nishimura, "Medical image noise reduction using Radon transform and Walsh list in Laplacian pyramid domain," in *Consumer Electronics, 2009. ISCE '09. IEEE 13th International Symposium on*, 2009, pp. 756–760.

[83] P. Bao and L. Zhang, "Noise reduction for magnetic resonance images via adaptive multiscale products thresholding," *Medical Imaging, IEEE Transactions on*, vol. 22, no. 9, pp. 1089–1099, Sept. 2003.

[84] M. Katona, A. Pižurica, N. Teslić, V. Kovačević, and W. Philips, "A real-time wavelet-domain video denoising implementation in FPGA," *EURASIP J. Embedded Syst.*, vol. 2006, no. 1, pp. 1–12, 2006.

[85] A. Ahmedsaid and A. Amira, "Accelerating SVD on reconfigurable hardware for image denoising," in *Image Processing, 2004. ICIP '04. 2004 International Conference on*, vol. 1, 2004, pp. 259–262.

[86] J. A. Lee and T. Lang, "Constant-factor redundant CORDIC for angle calculation and rotation," *Computers, IEEE Transactions on*, vol. 41, no. 8, pp. 1016–1025, Aug. 1992.

[87] A. E. Savakis and R. Carbone, "Discrete wavelet transform core for image processing applications," *Real-Time Imaging IX*, vol. 5671, no. 1, pp. 142–151, 2005.

[88] F. X. Lapalme, A. Amer, and C. Wang, "FPGA architecture for real-time video noise estimation," in *Image Processing, 2006 IEEE International Conference on*, 2006, pp. 3257–3260.

[89] M. Alle, J. Biswas, and S. K. Nandy, "High performance VLSI architecture design for H.264 CAVLC decoder," in *Application-specific Systems, Architectures and Processors, 2006. ASAP '06. International Conference on*, Sept. 2006, pp. 317–322.

[90] Y. Qu, Y. He, and S. Mei, "A novel cost-effective and programmable VLSI architecture of CAVLC decoder for H.264/AVC," *Journal of Signal Processing Systems*, vol. 50, pp. 183–193, Jan. 2008.

[91] P. Y. Chen and Y. M. Lin, "A Low-Cost CAVLC Encoder," *IEICE Trans. Electron.*, vol. E89-C, no. 12, pp. 1950–1953, 2006.

[92] H. Hu, J. Sun, and J. Xu, "High performance architecture design of CAVLC encoder in H.264/AVC," in *Image and Signal Processing, 2008. CISP '08. Congress on*, vol. 1, May 2008, pp. 613–616.

[93] H. Wei, B. Zhao, and P. He, "Hyperspectral image compression using SPIHT based on DCT and DWT," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 6787, 2007, pp. 67 870H–1–67 870H–7.

[94] X. L. Wen and Y. Xiao, "The 2-D directional DCT-DWT hybrid transform and its application in denoising ultrasound image," in *Signal Processing, 2008. ICSP 2008. 9th International Conference on*, Oct. 2008, pp. 946–949.

[95] Y. Xiao, T. Xiao, S. H. Hu, and M. H. Lee, "Two-dimensional hybrid transform (DCT-DWT) for 2-D signal processing," in *Signal Processing, 2006 8th International Conference on*, vol. 1, 2006, pp. 16–20.

[96] Z. Xiong, X. Wu, S. Cheng, and J. Hua, "Lossy-to-lossless compression of medical volumetric data using three-dimensional integer wavelet transforms," *Medical Imaging, IEEE Transactions on*, vol. 22, no. 3, pp. 459–470, Mar. 2003.

[97] J. Xu, Z. Xiong, S. Li, and Y. Q. Zhang, "Three-dimensional embedded subband coding with optimized truncation 3-D ESCOT," *Applied and Computational Harmonic Analysis*, vol. 10, no. 3, pp. 290–315, 2001.

[98] B. J. Kim and W. A. Pearlman, "An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees SPIHT," in *In Proc. Data Compression Conference*. IEEE Computer Society, 1997, pp. 251–260.

[99] A. Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 6, pp. 243–250, 1996.

[100] E. Sahin and I. Hamzaoglu, "A high performance and low power hardware architecture for H.264 CAVLC algorithm," in *Signal Processing, 2005. EUSIPCO 2005. European Conference on*, Sept. 2005.

[101] C. Rahman and W. Badawy, "CAVLC encoder design for real-time mobile video applications," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 54, no. 10, pp. 873–877, Oct. 2007.

[102] Z. Xiao and B. Baas, "A high-performance parallel CAVLC encoder on a fine-grained many-core system," in *Computer Design, 2008. ICCD 2008. IEEE International Conference on*, 2008, pp. 248–254.

[103] D. Kim, E. Jung, H. Park, H. Shin, and D. Har, "Implementation of high performance CAVLC for H.264/AVC video codec," in *System-on-Chip for Real-Time Applications, The 6th International Workshop on*, Dec. 2006, pp. 20–23.

[104] X. Meihua, L. Ke, X. Xiangguang, and F. Yule, "Optimization of CAVLC algorithm and its FPGA implementation," in *Electronic Packaging Technology High Density Packaging, 2008. ICEPT-HDP 2008. International Conference on*, 2008, pp. 1–4.

[105] M. G. Parris, "Optimizing dynamic location realizations of partial reconfiguration of field programmable gate arrays," Master Thesis, School of Electrical Engineering and Computer Science, University of Central Florida Orlando, 2009.

[106] J. Huang, M. Parris, J. Lee, and R. F. Demara, "Scalable FPGA-based architecture for DCT computation using dynamic partial reconfiguration," *ACM Trans. Embed. Comput. Syst.*, vol. 9, no. 1, pp. 9–18, 2009.

[107] B. Krill, A. Ahmad, A. Amira, and H. Rabah, "An efficient FPGA-based dynamic partial reconfiguration design flow and environment for image and signal processing IP cores," *Signal Processing: Image Communication*, vol. 25, no. 5, pp. 377–387, May 2010.

[108] B. Rousseau, P. Manet, D. Galerin, D. Merkenbreack, J.-D. Legat, F. Dedeken, and Y. Gabriel, "Enabling certification for dynamic partial reconfiguration using a minimal flow," in *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, 2007, pp. 1–6.

[109] M. Majer, J. Teich, A. Ahmadinia, and C. Bobda, "The Erlangen slot machine: A dynamically reconfigurable FPGA-based computer," *The Journal of VLSI Signal Processing.*, vol. 47, pp. 15–31, Apr. 2007.

[110] C. Claus, J. Zeppenfeld, F. Muller, and W. Stechele, "Using partial-run-time reconfigurable hardware to accelerate video processing in driver assistance system," in *Proc. Conference Design, Automation, Test and Exhibition in Europe (DATE '07)*, Nice, France, 2007, pp. 1–6.

[111] L. Braun, K. Paulsson, H. Kromer, M. Hubner, and J. Becker, "Data path driven waveform-like reconfiguration," in *Proc. International Conference on Field Programmable Logic and Applications (FPL 2008)*, Heidelberg, Germany, 2008, pp. 607–610.

[112] J. Hagemeyer, B. Kettelhoit, M. Koester, and M. Porrmann, "Design of homogeneous communication infrastructures for partially reconfigurable FPGAs," in *Engineering of Reconfigurable Systems and Algorithms. ERSA07. The 2007 International Conference on*, June 2007, pp. 1–10.

[113] C. Kao, "Benefits of partial reconfiguration," in *Xcell Journal Fourth Quarter*, 2005, pp. 66–67.

[114] R. Fong, S. Harper, and P. Athanas, "A versatile framework for FPGA field updates: An application of partial self-reconfiguration," in *Rapid Systems Prototyping, 2003. Proceedings. 14th IEEE International Workshop on*, June 2003, pp. 117–123.

[115] X. Zhang, H. Rabah, and S. Weber, "Dynamic slowdown and partial reconfiguration to optimize energy in FPGA based auto-adaptive SoPC," in *Electronic Design, Test and Applications, 2008. DELTA 2008. 4th IEEE International Symposium on*, Jan. 2008, pp. 153–157.

[116] [Online], "Accessed 10 Jan. 2008," http://www.xilinx.com.

[117] A. Shoa and S. Shirani, "Run-time reconfigurable systems for digital signal processing applications: A survey," *The Journal of VLSI Signal Processing.*, vol. 39, pp. 213–235, 2005.

[118] P. Manet, D. Maufroid, L. Tosi, G. Gailliard, O. Mulertt, M. Di Ciano, J.-D. Legat, D. Aulagnier, C. Gamrat, R. Liberati, V. La Barba, P. Cuvelier, B. Rousseau, and P. Gelineau, "An evaluation of dynamic partial reconfiguration for signal and image processing in professional electronics applications," *EURASIP J. Embedded Syst.*, vol. 2008, pp. 1–11, 2008.

[119] C. Bajaj, I. Ihm, and S. Park, "3D RGB image compression for interactive applications," *ACM Transactions on Graphics*, vol. 20, no. 1, pp. 10–38, Jan. 2001.

[120] D. Montgomery, F. Murtagh, and A. Amira, "A wavelet based 3D image compression system," in *Signal Process. and Its Applications, Proc. Seventh Int. Symp. on*, vol. 1, July 2003, pp. 65–68.

[121] P. Lysaght, B. Blodget, J. Mason, J. Young, and B. Bridgford, "Invited paper: Enhanced architectures, design methodologies and CAD tools for dynamic reconfiguration of Xilinx FPGAs," in *Field Programmable Logic and Applications, 2006. FPL '06. International Conference on*, Aug. 2006, pp. 1–6.

[122] M. Ghazel, "Adaptive fractal and wavelet image denoising," PhD Thesis, School of Electrical and Computer Engineering, University of Waterloo, 2004.

[123] [Online], "Accessed 1 Dec. 2009," http://www.mathworks.com.

[124] "AccelDSP Synthesis Tool User Guide," Xilinx Inc., Tech. Rep. UG634 (v11.4), Dec. 2009.

[125] G. Yu, T. Vladimirova, X. Wu, and M. Sweeting, "A new high-level reconfigurable lossless image compression system for space applications," in *Adaptive Hardware*

and Systems, 2008. AHS '08. NASA/ESA Conference on, June 2008, pp. 183–190.

[126] S. K. Mohideen, A. Perumal, and M. M. Sathik, "Image de-noising using discrete wavelet transform," *International Journal of Computer Science and Network Security*, vol. 8, no. 1, pp. 213–216, Jan. 2008.

[127] D. L. Donoho and I. M. Johnstone, "Adapting to unknown smoothness via wavelet shrinkage," *Journal of the American Statistical Association*, vol. 90, no. 432, pp. 1200–1224, 1995.

[128] Y. F. Low and R. Besar, "Optimal wavelet filters for medical image compression," *Int. Journal of Wavelets, Multiresoultion and Info. Process.*, vol. 1, pp. 179–197, June 2003.

[129] J. Zheng, D. Wu, D. Xie, and W. Gao, "A novel pipeline design for H.264 CABAC decoding," *Lecture Notes in Computer Science – Advances in Multimedia Information Processing*, vol. 4810, pp. 559–568, 2007.

[130] V. Sanchez, P. Nasiopoulos, and R. Abugharbieh, "Efficient 4D motion compensated lossless compression of dynamic volumetric medical image data," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, Mar. 2008, pp. 549–552.

[131] M. D. Adams and F. Kossentni, "Reversible integer-to-integer wavelet transforms for image compression: performance evaluation and analysis," *Image Processing, IEEE Transactions on*, vol. 9, no. 6, pp. 1010–1024, June 2000.

[132] G. Beylkin, R. Coifman, and V. Rokhlin, "Fast wavelet transforms and numerical algorithms I," *Communications on Pure and Applied Mathematics*, vol. 44, no. 2, pp. 141–183, 1991.

[133] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin, "Wavelets for computer graphics: A primer, part 1," *IEEE Comput. Graph. Appl.*, vol. 15, no. 3, pp. 76–84, 1995.

[134] X. Lan, N. Zheng, and Y. 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, May 2005.

[135] N. V. Boulgouris, A. Leontaris, and M. G. Strintzis, "Wavelet compression of 3D medical images using conditional arithmetic coding," in *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, vol. 4, 2000, pp. 557–560.

[136] D. Montgomery, A. Amira, and F. Murtagh, "A non-separable lifting approach for 3D image compression," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, vol. 3, May 2004, pp. 137–140.

[137] W. Sweldens, "Lifting scheme: a new philosophy in biorthogonal wavelet constructions," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, ser. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 2569, Sept. 1995, pp. 68–79.

[138] A. R. Calderbank, I. Daubechies, W. Sweldens, and Y. Boon-Lock, "Lossless image compression using integer to integer wavelet transforms," in *Image Processing, 1997. Proceedings., International Conference on*, vol. 1, Oct. 1997, pp. 596–599.

[139] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity transform and quantization in H.264/AVC," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 598–603, July 2003.

[140] T. C. Chen, Y. W. Huang, C. Y. Tsai, B. Y. Hsieh, and L. G. Chen, "Architecture design of context-based adaptive variable-length coding for H.264/AVC," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 53, no. 9, pp. 832–836, Sept. 2006.

[141] A. D. Booth, "A signed binary multiplicaton technique," *Quart. Journ. Mech. and Applied Math.*, vol. 4, no. 2, pp. 236–240, 1951.

[142] C. D. Chien, K. P. Lu, Y. H. Shih, and J. I. Guo, "A high performance CAVLC encoder design for MPEG-4 AVC/H.264 video coding applications," in *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, May 2006, pp. 3838–3841.

[143] A. Ahmad, B. Krill, A. Amira, and H. Rabah, "3D Haar wavelet transform with dynamic partial reconfiguration for 3D medical image compression," in *Biomedical Circuits and Systems Conference, 2009. BioCAS 2009. IEEE*, 2009, pp. 137–140.

[144] A. Ahmad and A. Amira, "Efficient reconfigurable architectures for 3D medical image compression," in *Field-Programmable Technology, 2009. FPT 2009. International Conference on*, 2009, pp. 472–474.

[145] "Starbridge HC-36 and HC-62 hypercomputers system hardware manual," Starbridge Systems., *Technical Report* 00001D-000A, 2006.

[146] "Virtex-5 FPGA User Guide," Xilinx Inc., Tech. Rep. UG190 (v4.3), Sept. 2008.

[147] "Virtex-5 FPGA XtremeDSP Design Considerations," Xilinx Inc., Tech. Rep. UG193 (v3.2), Sept. 2008.

[148] "Partial Reconfiguration User Guide," Xilinx Inc., Tech. Rep. UG702 (v 12.1), May 2010.