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.
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
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCM</td>
<td>Digital clock management</td>
</tr>
<tr>
<td>DCT</td>
<td>Discrete cosine transform</td>
</tr>
<tr>
<td>DDR-2</td>
<td>Double data rate</td>
</tr>
<tr>
<td>DFF</td>
<td>D flip-flop</td>
</tr>
<tr>
<td>DFT</td>
<td>Discrete Fourier transform</td>
</tr>
<tr>
<td>DHT</td>
<td>Discrete Hartley transform</td>
</tr>
<tr>
<td>DMA</td>
<td>Distortion minimisation algorithm</td>
</tr>
<tr>
<td>DPR</td>
<td>Dynamic partial reconfiguration</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital signal processor</td>
</tr>
<tr>
<td>DWT</td>
<td>Discrete wavelet transform</td>
</tr>
<tr>
<td>EAPR</td>
<td>Early access partial reconfiguration</td>
</tr>
<tr>
<td>EDA</td>
<td>Electronic design automation</td>
</tr>
<tr>
<td>ESCOT</td>
<td>Embedded sub-band coding with optimal truncation</td>
</tr>
<tr>
<td>ESM</td>
<td>Erlangen slot machine</td>
</tr>
<tr>
<td>EVD</td>
<td>Eigen value decomposition</td>
</tr>
<tr>
<td>FIR</td>
<td>Finite impulse response</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier transform</td>
</tr>
<tr>
<td>FIFO</td>
<td>First in first out</td>
</tr>
<tr>
<td>FMRI</td>
<td>Functional magnetic resonance imaging</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field programmable gate array</td>
</tr>
<tr>
<td>fps</td>
<td>Frames per second</td>
</tr>
<tr>
<td>FRAT</td>
<td>Finite Radon transform</td>
</tr>
<tr>
<td>FRIT</td>
<td>Finite ridgelet transform</td>
</tr>
<tr>
<td>FWT</td>
<td>Fast wavelet transform</td>
</tr>
<tr>
<td>GOP</td>
<td>Group of pictures</td>
</tr>
<tr>
<td>GPGPU</td>
<td>General-purpose computation on graphics processing units</td>
</tr>
<tr>
<td>GPP</td>
<td>General purpose processor</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics processing unit</td>
</tr>
<tr>
<td>HBWD</td>
<td>Hierarchical block wavelet decomposition</td>
</tr>
<tr>
<td>HD</td>
<td>High-definition</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>HDMI</td>
<td>High-definition medical imaging</td>
</tr>
<tr>
<td>HDTV</td>
<td>High-definition TV</td>
</tr>
<tr>
<td>HLL</td>
<td>High-level language</td>
</tr>
<tr>
<td>HW</td>
<td>Hardware</td>
</tr>
<tr>
<td>HWT</td>
<td>Haar wavelet transform</td>
</tr>
<tr>
<td>HVS</td>
<td>Human visual system</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/output</td>
</tr>
<tr>
<td>IOB</td>
<td>Input/output block</td>
</tr>
<tr>
<td>ICAP</td>
<td>Internal configuration access port</td>
</tr>
<tr>
<td>ILA</td>
<td>Integrated logic analyzer</td>
</tr>
<tr>
<td>IT</td>
<td>Integer transform</td>
</tr>
<tr>
<td>IRT</td>
<td>Inverse Radon transform</td>
</tr>
<tr>
<td>JPEG</td>
<td>Joint photographic experts group</td>
</tr>
<tr>
<td>LC</td>
<td>Logic cell</td>
</tr>
<tr>
<td>LUT</td>
<td>Look-up tables</td>
</tr>
<tr>
<td>MAV</td>
<td>Median absolute value</td>
</tr>
<tr>
<td>MPGA</td>
<td>Mask programmable gate array</td>
</tr>
<tr>
<td>MRI</td>
<td>Magnetic resonance imaging</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean square error</td>
</tr>
<tr>
<td>NCD</td>
<td>Native circuit description</td>
</tr>
<tr>
<td>NFS</td>
<td>Networking file system</td>
</tr>
<tr>
<td>NMC</td>
<td>Native macro circuit</td>
</tr>
<tr>
<td>NSWD</td>
<td>Non-standard wavelet decomposition</td>
</tr>
<tr>
<td>NTSC</td>
<td>National television system committee</td>
</tr>
<tr>
<td>OT</td>
<td>Objective test</td>
</tr>
<tr>
<td>PAL</td>
<td>Programmable arrays logic</td>
</tr>
<tr>
<td>PAR</td>
<td>Phase alternate line</td>
</tr>
<tr>
<td>PC</td>
<td>Personal computer</td>
</tr>
<tr>
<td>PCI</td>
<td>Peripheral component interconnect</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>PET</td>
<td>Positron emission tomography</td>
</tr>
<tr>
<td>PLL</td>
<td>Phase-locked-loop</td>
</tr>
<tr>
<td>PR</td>
<td>Partial reconfiguration</td>
</tr>
<tr>
<td>PSNR</td>
<td>Peak signal to noise ratio</td>
</tr>
<tr>
<td>QCIF</td>
<td>Quarter common intermediate format</td>
</tr>
<tr>
<td>RAM</td>
<td>Random access memory</td>
</tr>
<tr>
<td>RH</td>
<td>Reconfigurable hardware</td>
</tr>
<tr>
<td>ROM</td>
<td>Read only memory</td>
</tr>
<tr>
<td>ROI</td>
<td>Regions of interest</td>
</tr>
<tr>
<td>RPM</td>
<td>Reconfigurable processing modules</td>
</tr>
<tr>
<td>RT</td>
<td>Radon transform</td>
</tr>
<tr>
<td>RTL</td>
<td>Register-transfer level</td>
</tr>
<tr>
<td>RTR</td>
<td>Run-time reconfiguration</td>
</tr>
<tr>
<td>SoPC</td>
<td>Systems on a programmable chip</td>
</tr>
<tr>
<td>SPIHT</td>
<td>Set partitioning in hierarchical trees</td>
</tr>
<tr>
<td>SRAM</td>
<td>Static RAM</td>
</tr>
<tr>
<td>ST</td>
<td>Subjective test</td>
</tr>
<tr>
<td>STFT</td>
<td>Short time Fourier transform</td>
</tr>
<tr>
<td>SVD</td>
<td>Singular value decomposition</td>
</tr>
<tr>
<td>SW</td>
<td>Software</td>
</tr>
<tr>
<td>SWD</td>
<td>Standard wavelet decomposition</td>
</tr>
<tr>
<td>UCF</td>
<td>User constraint file</td>
</tr>
<tr>
<td>UK</td>
<td>United Kingdom</td>
</tr>
<tr>
<td>US</td>
<td>Ultrasound</td>
</tr>
<tr>
<td>VGA</td>
<td>Video graphic array</td>
</tr>
<tr>
<td>VHDL</td>
<td>Very-high-speed integrated circuit hardware description language</td>
</tr>
<tr>
<td>VLC</td>
<td>Variable length coding</td>
</tr>
<tr>
<td>VLSI</td>
<td>Very large scale integration</td>
</tr>
<tr>
<td>XE</td>
<td>Xilinx edition</td>
</tr>
</tbody>
</table>
Table of Contents

Abstract iii

Declaration v

Acknowledgements vii

Author’s Publications viii

1 Introduction 1

1.1 Overview ........................................... 1

1.2 Three-dimensional (3-D) Medical Image Processing ............... 5

1.3 High-Performance Solutions for Medical Image Processing Applications 10

1.3.1 Digital Signal Processor (DSP) .......................... 11

1.3.2 Special Purpose Application Specific Integrated Circuit (ASIC) Hardware ........................................... 12

1.3.3 Graphical Processing Unit (GPU) ............................ 13

1.3.4 Reconfigurable Hardware (RH): A Review of Field Programmable Gate Array (FPGA) .......................... 15

1.4 Design and Implementation Strategies .......................... 18

1.5 Motivation and Research Objectives .......................... 19
# Table of Contents

1.6 Overall Contribution ....................................... 22  
1.7 Thesis Organisation ........................................ 24  

2 Related Work .................................................. 25  
2.1 Overview .................................................... 25  
2.2 Medical Image Compression ................................. 27  
2.3 Reconfigurable Architectures ............................... 34  
   2.3.1 FPGA-based Architectures for 3-D Discrete Wavelet Transform (DWT) .............................................. 34  
   2.3.2 FPGA-based Architectures for Finite Radon Transform (FRAT) .................................................. 40  
   2.3.3 FPGA-based Architectures for Context-based Adaptive Variable Length Coding (CAVLC) ............................... 51  
2.4 Dynamic Partial Reconfiguration (DPR) ..................... 58  
2.5 Limitation of Existing Work and Research Opportunities .... 61  
2.6 Summary ...................................................... 63  

3 Efficient Architectures for 3-D HWT using DPR ................. 64  
3.1 Overview ...................................................... 64  
3.2 Mathematical Background and Design Methodology ............ 65  
   3.2.1 3-D Haar Wavelet Transform (HWT) and Matrix Transposition .................................................. 65  
   3.2.2 Pipelined Direct Mapping Implementation ................. 68  
3.3 Proposed Architectures ....................................... 69  
   3.3.1 Proposed System Applications ........................... 69  
   3.3.2 3-D Haar Wavelet Transform (HWT) with Transpose-based Computation .................................................. 70
Table of Contents

3.3.3 3-D Haar Wavelet Transform (HWT) with Dynamic Partial Reconfiguration (DPR) 73

3.4 Experimental Results and Analysis 76
   3.4.1 Field Programmable Gate Array (FPGA) Implementation 76
   3.4.2 Discussions 78

3.5 Summary 82

4 FPGA-based Architectures of FRAT for Medical Image De-noising 84
   4.1 Overview 84
   4.2 Mathematical Background and Design Methodology 86
      4.2.1 Radon Transform (RT) 86
      4.2.2 Finite Radon Transform (FRAT) 88
      4.2.3 Xilinx AccelDSP Design Flow 89
   4.3 Proposed System Implementations 91
      4.3.1 Systems Applications 91
      4.3.2 Proposed Architecture and Design Strategies 93
   4.4 Results and Analysis 96
      4.4.1 Medical Image De-noising 100
      4.4.2 Software Simulation 101
      4.4.3 Hardware Implementation 103
   4.5 Summary 105

5 FPGA-based Implementation of a 3-D Medical Image Compression System using CAVLC 106
   5.1 Overview 106
   5.2 Algorithms and Methodology 108
# Table of Contents

5.2.1 3-D Integer Transform (IT) .................................. 108  
5.2.2 3-D Discrete Wavelet Transform (DWT) ...................... 111  
5.2.3 Decomposition Strategies .................................... 112  

5.3 Proposed System Architectures ................................. 114  
5.3.1 Transform Block ........................................... 115  
5.3.2 Quantisation and Reordering Block ......................... 116  
5.3.3 Context-based Adaptive Variable Length Coding (CAVLC) Block 117  

5.4 Results and Analysis ........................................... 122  
5.4.1 Computational Complexity ................................... 122  
5.4.2 Objective Evaluation ....................................... 123  
5.4.3 Field Programmable Gate Array (FPGA) Implementation ... 127  

5.5 Summary .................................................... 131  

6 Conclusions and Future Work .................................. 133  
6.1 Overview .................................................... 133  
6.2 Achievements ............................................... 134  
6.3 Limitations .................................................. 135  
6.4 Future Work ................................................ 136  

Appendices ...................................................... 139  
A Rapid Prototyping Board and FPGA Devices .................. 139  
A.1 Overview .................................................... 139  
A.2 XUPV5-LX110T Prototyping Board .............................. 139  
A.3 Virtex-5 Field Programmable Gate Array (FPGA) ............ 140  
A.3.1 Configurable Logic Block (CLB) ............................ 142
List of Figures

1.1 Number of new cases of all malignant neoplasms in UK 2007 (Excluding non-melanoma skin cancer) [2]. ........................................ 2
1.2 Medical image features. ............................................. 6
1.3 Examples of medical images (a) Sagittal MRI knee image (b) Transaxial CT lung slice (c) PET scan for lymphoma [22]. .............. 6
1.4 3-D medical image features. ........................................ 7
1.5 3-D medical image data processing. ............................... 7
1.6 Survey on medical image processing. ............................ 8
1.7 DSPs features for performance accelerations. .................... 11
1.8 Main disadvantages of ASICs. ..................................... 13
1.9 Architecture comparison (a) CPU (b) GPU [47]. ................. 14
1.10 Xilinx’s FPGA structure with internal blocks. ................... 17
1.11 Generic design and implementation strategies. .................. 19
1.12 Overall design flow. ............................................. 20
1.13 Overall research approaches and contributions. ................. 23
2.1 Structure of related research issues. ............................. 26
2.2 Compression system. ............................................. 27
2.3 Implementation based on parallel computing [7]. ................. 28
2.4 The 3-D DWT process. ........................................ 34
2.5 Block architecture for the 3-D DWT [66]. ........................ 36
2.6 3-D DWT processor architecture [9]. .......................... 37
2.7 Design of 3D-V temporal decomposition system [67]. ............ 38
2.8 Hardware design for the 3-D Haar wavelet transform [68]. ......... 38
2.9 Proposed architectures (a) Generic transform architecture (b) Radon transform module [73]. ................................. 41
2.10 (a) Reference FRAT architecture (b) Memoryless FRAT architecture [75]. 42
2.11 Block diagram of proposed FRAT implementation [72]. ......... 43
2.12 (a) Serial architecture (b) Parallel architecture [76]. ........... 44
2.13 (a) Reference architecture (b) FRIT architecture with the FRAT [71]. 45
2.14 Review of FRAT’s FPGA-based implementation. ................. 47
2.15 FPGA implementation of the proposed wavelet-domain video denoising algorithm [84]. ........................................ 47
2.16 FPGA implementation of the SVD/EVD array [85]. ............... 48
2.17 Block diagram of the proposed FPGA design [88]. ............... 50
2.18 CAVLC hardware architecture [100]. ............................ 53
2.19 The proposed CAVLC architecture [101]. ......................... 54
2.20 (a) Architecture of targeted many-core system (b) Data flow diagram of the CAVLC encoder [102]. .......................... 55
2.21 Framework of CAVLC encoder [104]. ............................ 57
2.22 Overview of the partitioning scheme approaches (a) 1-D (b) Multi-1-D (c) 2-D [112]. ................................................. 59
3.1 3-D HWT expression. ........................................... 66
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. 67

3.3 Transposition of a matrix. 68

3.4 1-D HWT flow diagram with $N$-inputs sample for direct mapped architecture. 69

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 $[I]^2$ (d) Transpose matrix after $T_1$ (e) Transpose matrix after $T_2$. 70

3.6 Proposed reconfigurable and adaptive system architectures. 73

3.7 Proposed top architecture of 3-D HWT (a) Without DPR (b) With DPR. 74

3.8 Partial reconfiguration design flow (a) Steps for partial design flow (b) Define static and reconfigurable modules. 75

3.9 Influence of transform size on area. 78

3.10 Influence of transform size on power consumption. 79

3.11 Influence of transform size on maximum frequency for 1-D HWT modules. 79

3.12 Comparison on maximum frequency achievement for transpose function. 80

3.13 Comparison of chip layout for different Virtex-5 devices for $N = 64$. 80

4.1 Transform flow graph (a) Ridgelet transform (b) Curvelet transform. 85

4.2 Radon transform representation. 87
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3</td>
<td>Proposed system applications (a) Image de-noising (b) Compression system.</td>
<td>91</td>
</tr>
<tr>
<td>4.4</td>
<td>Proposed reference architecture for the FRAT.</td>
<td>94</td>
</tr>
<tr>
<td>4.5</td>
<td>Implementation strategies (a) Sequential (b) Pipelined (c) BRAM-based method.</td>
<td>95</td>
</tr>
<tr>
<td>4.6</td>
<td>Script and function files for the sequential implementation.</td>
<td>97</td>
</tr>
<tr>
<td>4.7</td>
<td>Function operations with generated fixed point report.</td>
<td>98</td>
</tr>
<tr>
<td>4.8</td>
<td>Project explorer with VHDL files generated.</td>
<td>99</td>
</tr>
<tr>
<td>4.9</td>
<td>Gaussian noise reduction experimental results on MRI image (a) Original (b) Noisy (c) De-noising.</td>
<td>101</td>
</tr>
<tr>
<td>4.10</td>
<td>Original and blockiness images.</td>
<td>101</td>
</tr>
<tr>
<td>4.11</td>
<td>Analysis of PSNR with different block sizes ($p$).</td>
<td>102</td>
</tr>
<tr>
<td>4.12</td>
<td>Chip layout for the sequential implementation.</td>
<td>104</td>
</tr>
<tr>
<td>5.1</td>
<td>Coefficient orderings (a) Convolution-based (b) Lifting-based.</td>
<td>112</td>
</tr>
<tr>
<td>5.2</td>
<td>Sub-band structure obtained via a three level SWD.</td>
<td>114</td>
</tr>
<tr>
<td>5.3</td>
<td>Proposed system overview.</td>
<td>114</td>
</tr>
<tr>
<td>5.4</td>
<td>Butterfly architecture of 1-D IT.</td>
<td>115</td>
</tr>
<tr>
<td>5.5</td>
<td>A simple lifting-based perfect reconstruction encoder.</td>
<td>116</td>
</tr>
<tr>
<td>5.6</td>
<td>Block diagram of CAVLC architecture.</td>
<td>120</td>
</tr>
<tr>
<td>5.7</td>
<td><strong>Encode level</strong> detail of the CAVLC architecture.</td>
<td>122</td>
</tr>
<tr>
<td>5.8</td>
<td>PSNR vs. BPV for CT.</td>
<td>125</td>
</tr>
<tr>
<td>5.9</td>
<td>PSNR vs. BPV for MRI.</td>
<td>125</td>
</tr>
<tr>
<td>5.10</td>
<td>PSNR vs. BPV for PET.</td>
<td>126</td>
</tr>
<tr>
<td>5.11</td>
<td>Comparison of original and reconstructed CT, MRI and PET images for the first slices.</td>
<td>127</td>
</tr>
</tbody>
</table>
5.12 Compression system. ........................................... 128
5.13 Power consumption comparison for the CAVLC architecture. ..... 131

A.1 Virtex-5 FPGA and XUPV5-LX110T platform block diagram [146]. .... 140
A.2 Detailed description of XUPV5-LX110T platform components (front view). ......................................................... 141
A.3 Arrangement of slices within the CLB for Virtex-5 [146]. .............. 142
A.4 Details of CLBs and slices for Virtex-5 [146]. ............................ 143

B.1 General design route from VHDL to prototyping board. ............... 147
B.2 Sample window displaying ISE project navigator. .................... 149
B.3 ModelSim simulator window. ...................................... 149
B.4 Setting the design options in ISE. ................................... 150
B.5 Setting for UCF. .................................................. 151
B.6 Floorplan for pin location constraints. ................................ 151
B.7 FPGA editor window. ............................................. 152
B.8 Device configuration using iMPACT. .................................. 153
B.9 Program succeeded to be downloaded. .................................. 154
B.10 Results verification using LEDs indicator. .............................. 154

C.1 Basic concept of partial reconfiguration. ............................... 155
C.2 Design tools requirement in PR. .................................... 157
C.3 General PR design flow. ........................................... 157
C.4 Overview of PR software design flow. ................................ 158

D.1 Advantages of AccelDSP synthesis tool. ............................. 161
D.2 The AccelDSP ISE synthesis work flow. .............................. 162
# List of Tables

1.1 Summary of programming technologies [17]...
1.2 Comparison of different implementation approaches...
1.3 Survey on medical image processing.

2.1 Device utilisation [8]...
2.2 Summary of 3-D medical image compression systems.
2.3 Comparative study of the 3-D DWT architectures and the FPGA implementations.
2.4 Summary of FPGA-based architectures of FRAT.
2.5 Hardware implementation of medical image de-noising.
2.6 Equivalent gate for CAVLC items [103].
2.7 Summary of hardware implementation of CAVLC.

3.1 Resources utilisation and overall proposed architectures performance on XC5VLX110T-3FF113.
3.2 Comparison of bitstream generated and configuration times towards transform sizes.
3.3 Device summary report of the proposed architecture on XC5VLX30T-3FF323.
List of Tables

4.1 PSNR quantitative results of noisy image with a Gaussian white noise and MRI image. ........................................ 100

4.2 Comparison of performance with existing architectures for the case $p = 7$. ..................................................... 103

4.3 Comparison of PSNR values for CT images. .......................... 104

5.1 Computational complexity of the main functional blocks with various decomposition approaches. ............................. 123

5.2 Images used for testing. .............................................. 124

5.3 Hardware resources utilisation for each block. ....................... 128

5.4 Resources utilisation and overall transform architectures performance for $N = 4$. .................................................. 129

5.5 FPGA implementation results of CAVLC. .......................... 130

5.6 Comparison of CAVLC architectures performance on FPGA platforms. 130

A.1 Comparison of selected Xilinx FPGA devices resources. ........ 145

C.1 Description of files format for PR process. ......................... 159
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].
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 \cite{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 \cite{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 \cite{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 \cite{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 \cite{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 \cite{13}. GPUs have become of choice for many computationally intensive applications as it contains with many processing elements, high-memory bandwidth, and programmability \cite{6}. However, major obstacle of GPUs is concerned with less efficient mapping parallel application in the GPU’s pixel processing data paths \cite{12}.

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

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 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].

<table>
<thead>
<tr>
<th>Feature</th>
<th>SRAM</th>
<th>Antifuse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology node</td>
<td>State-of-the-art</td>
<td>One or more generation behind</td>
</tr>
<tr>
<td>Reprogrammable</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Volatility</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Good for prototyping</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Power consumption</td>
<td>Medium</td>
<td>Low</td>
</tr>
</tbody>
</table>

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,
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

<table>
<thead>
<tr>
<th>Platform</th>
<th>Performance</th>
<th>Cost</th>
<th>Power</th>
<th>Flexibility</th>
<th>Design effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASIC</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>DSP</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>GPP</td>
<td>Low</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>GPU</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>RH</td>
<td>Medium/High*</td>
<td>Medium</td>
<td>High/Low#</td>
<td>High</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Note:
* Depends on technology and available embedded resources
# With Xilinx Spartan’s FPGA
1.2. Three-dimensional (3-D) Medical Image Processing

Medical image features

<table>
<thead>
<tr>
<th>Modality</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRI</td>
<td>Used throughout the body with spatial resolution of approximately 0.5mm, which is better than most other tomographic techniques. Provides by far the best contrast resolution, allowing differences between two similar but not identical tissues to be distinguished.</td>
</tr>
<tr>
<td>CT</td>
<td>Provides geometric superiority and slightly increased spatial resolution over MRI. CT exposes the patient to high doses of radiation and maybe invasive when the administration of contrast agent is required.</td>
</tr>
<tr>
<td>PET</td>
<td>Compared to other imaging modalities, nuclear medicine scans have a low signal-to-noise-ratio (SNR) and poor resolution, but an extremely high contrast-to-noise-ratio (CNR). Inherent image contrast in PET is extremely high due to the lack of background signal that can be obtained from tissues, where radio-pharmaceutical has not distributed.</td>
</tr>
</tbody>
</table>

Figure 1.2: Medical image features.

![MRI image](a) ![CT image](b) ![PET image](c)

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.
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 \( \Delta_z \) pixels, where \( \Delta_z \) 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.](image)

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

Table 1.3: Survey on medical image processing.

<table>
<thead>
<tr>
<th>Refs.</th>
<th>Applications</th>
<th>Image type</th>
<th>Implementations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 4 5 6</td>
<td>2-D 3-D HW</td>
<td>SW General</td>
</tr>
<tr>
<td>[3]</td>
<td>✓ ✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>[6]</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓</td>
</tr>
<tr>
<td>[21]</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓</td>
</tr>
<tr>
<td>[23]</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓</td>
</tr>
<tr>
<td>[26]</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓</td>
</tr>
<tr>
<td>[28]</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓</td>
</tr>
<tr>
<td>[29]</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓</td>
</tr>
<tr>
<td>[30]</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓</td>
</tr>
<tr>
<td>[31]</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

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 $O(N \times \log N)$ 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.](image)

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
1.3. High-Performance Solutions for Medical Image Processing Applications

Figure 1.8: Main disadvantages of ASICs.

Disadvantages of ASICs approach

Development time
- Special purpose hardware has a long development time, from design through simulation and fabrication

Cost
- It can also be expensive if it is a one-off solution or if the volume required cannot justify its fabrication costs

Flexibility
- Once this special purpose hardware is built, it is not possible to change the hardware to accommodate slightly different needs
- A new hardware is usually required for each new algorithm

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].

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 today’s 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

---

**Figure 1.9:** Architecture comparison (a) CPU (b) GPU [47].
1.3. High-Performance Solutions for Medical Image Processing Applications

For the most part, GPUs were originally designed for tasks for which GPUs were not well suited; 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 bitwise 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 today's 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.
1.3. High-Performance Solutions for Medical Image Processing Applications

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].
1.4 Design and Implementation Strategies

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.](image)

### 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...
1.5. Motivation and Research Objectives

Examine the MATLAB floating point model (m-file)
Verify
Simulation
Verify
Fixed point model
RTL model (VHDL/Verilog)
Gate level netlist
Bitstream and simulation files

Generate fixed point
Synthesise RTL
Implementation

Partition the system into modules
Define static modules and reconfigurable modules
Decide the number of partial reconfiguration regions
Decide partial reconfiguration region sizes, shapes and locations
Map modules to partial reconfiguration regions
Define partial reconfiguration region interfaces, instantiate slice macros for partial reconfiguration regions interfaces

Design partitioning
Design floor planning and budgeting

Create project
Add source
Check syntax
Select target
Synthesise
Optimise
Analyse timing

VHDL
Generate
CoreGen

Figure 1.12: Overall design flow.
1.5. Motivation and Research Objectives

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
1.6. Overall Contribution

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.
Bibliography


[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*


