

Efficient FPGA Implementation and Power Modelling of Image and Signal Processing IP Cores

A thesis submitted for the degree of
Doctor of Philosophy

by

Shrutisagar Chandrasekaran
B.E.

Brunel
UNIVERSITY
WEST LONDON

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

May 2007

Additional comments: _____

Part B, when complete, should be returned to the Assistant Registrar (Graduate Studies).

For the attention of candidates who have completed Part A

- i) Attention is drawn to the fact that the copyright of a thesis rests with its author.
- ii) A copy of a candidate's thesis is supplied to the University Library on condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without the prior written consent of the author or University, as appropriate.

Requests for such permission should be addressed in the first instance to the Head of Library Services.

Abstract

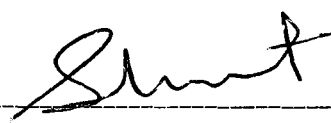
Field Programmable Gate Arrays (FPGAs) are the technology of choice in a number of image and signal processing application areas such as consumer electronics, instrumentation, medical data processing and avionics due to their reasonable energy consumption, high performance, security, low design-turnaround time and reconfigurability. Low power FPGA devices are also emerging as competitive solutions for mobile and thermally constrained platforms. Most computationally intensive image and signal processing algorithms also consume a lot of power leading to a number of issues including reduced mobility, reliability concerns and increased design cost among others. Power dissipation has become one of the most important challenges, particularly for FPGAs. Addressing this problem requires optimisation and awareness at all levels in the design flow. The key achievements of the work presented in this thesis are summarised here.

Behavioural level optimisation strategies have been used for implementing matrix product and inner product through the use of mathematical techniques such as Distributed Arithmetic (DA) and its variations including offset binary coding, sparse factorisation and novel vector level transformations. Applications to test the impact of these algorithmic and arithmetic transformations include the fast Hadamard/Walsh transforms and Gaussian mixture models. Complete design space exploration has been performed on these cores, and where appropriate, they have been shown to clearly outperform comparable existing implementations. At the architectural level, strategies such as parallelism, pipelining and systolisation have been successfully applied for the design and optimisation of a number of cores including colour space conversion, finite Radon transform, finite ridgelet transform and circular convolution. A pioneering study into the influence of supply voltage scaling for FPGA based designs, used in conjunction with performance enhancing strategies such as parallelism and pipelining has been performed. Initial results are very promising and indicated significant potential for future research in this area.

A key contribution of this work includes the development of a novel high level power macro-modelling technique for design space exploration and characterisation of custom IP cores for FPGAs, called Functional Level Power Analysis and Modelling (FLPAM). FLPAM is scalable, platform independent and compares favourably with existing approaches. A hybrid, top-down design flow paradigm integrating FLPAM with commercially available design tools for systematic optimisation of IP cores has also been developed.

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.

Signed:  Dated: 12 JULY 2007
(Shrutisagar Chandrasekaran)

To my parents

Acknowledgements

While one does not happen to notice time fly by during the course of a PhD program, the actual process of writing the dissertation can be an eye opening experience. As I put the finishing touches to my thesis, the personal and practical support of numerous people who have helped me through this process immediately springs to my mind.

First and foremost, I would like to thank my parents for being there (a phone call away, considering the distances involved) whenever the need arose. Special mention goes to my mother for her uncanny knack of tuning into my feelings and for her incredible ability to roll up love, patience, encouragement and realism into one package. Thank you dad for being my constant sounding board in all practical matters, small or great and for teaching me the value of objectivity in every action.

This PhD would not have been possible at all, without the constant support, encouragement, motivation and most importantly constructive criticism provided by my supervisor, Dr. Abbas Amira. Clichéd as the previous statement sounds, it is true in every sense. Although an extremely tough taskmaster, I greatly value the opportunity of working with him.

I would like to acknowledge the collaborative support provided by Prof. Amine Bermak of HKUST and Prof. P. K. Meher of NTU. Along the way, a number of colleagues, friends and acquaintances have enriched my life at Belfast, Hong Kong and London. Due to the impracticality of naming everyone and to avoid missing out someone inadvertently, I would like to just say - thank you all for the times we have spent together. Special mention would go to Ravi Kiran for listening to my rants patiently at all times of the day and night.

Author's Publication

Journal Papers

1. A. Amira and S. Chandrasekaran, "Power Modeling and Efficient FPGA Implementation of FHT for Signal Processing", *IEEE Transactions on VLSI Systems*, vol. 15, no. 3, pp. 286-295, March 2007
2. S. Chandrasekaran and A. Amira, "Power Modelling and Acceleration of Finite Radon Transform on Reconfigurable Hardware", *IEEE Transactions on Computers*, In Revision
3. A. Amira, S. Chandrasekaran, D. W. G. Montgomery and I S Uzun, "High Performance Implementation of PET Segmentation using a Multiresolution-Statistical Approach", *Elsevier Neurocomputing Special Issue on Vision Research*, Submitted
4. P. K. Meher, S. Chandrasekaran and A. Amira, "FPGA Realization of FIR Filters by Efficient and Flexible Systolization Using Distributed Arithmetic", *IEEE Transactions on Signal Processing*, Submitted

Conference Papers (selected)

1. S. Chandrasekaran, A. Amira, A. Bermak and M. Shi, "Performance Enhanced Voltage Scaling in FPGAs", *International Symposium on Integrated Circuits (ISIC) 2007*, September 26th - 28th, Singapore

2. M. Shi, S. Chandrasekaran, A. Bermak and A. Amira, "FPGA Based Run Time Reconfigurable Gas Discrimination System", *International Symposium on Integrated Circuits (ISIC) 2007*, September 26th - 28th, Singapore
3. S. Chandrasekaran and A. Amira, "A General Framework for Efficient FPGA Implementation and Power Modelling of Image Processing Cores", *IET Visual Information Engineering 2007 Conference*, July 25th - 27th, London, UK
4. A. Amira, S. Chandrasekaran and D. Skinner, "A Fast Hybrid LSI Approach for Intelligent Information Retrieval", *IET Visual Information Engineering 2007 Conference*, July 25th - 27th, London, UK
5. S. Chandrasekaran and A. Amira, "A New Behavioural Power Modelling Approach for FPGA based Custom Cores", *2007 NASA/ESA Conference on Adaptive Hardware and Systems*, August 5th - 8th, Edinburgh, Scotland, UK
6. S. Chandrasekaran and A. Amira, "Novel Sparse OBC Based Distributed Arithmetic Architecture for Matrix Transforms", *2007 IEEE International Symposium on Circuits and Systems*, May 27th - 30th, New Orleans, USA
7. F. Bensaali, S. Chandrasekaran, A. Amira, "Power Modeling and Efficient FPGA Implementation of Color Space Conversion", *13th IEEE International Conference on Electronics, Circuits and Systems 2006*, December 10th - 13th, Nice, France
8. M. Shi, A. Bermak, S. Chandrasekaran, A. Amira, "An Efficient FPGA Implementation of Gaussian Mixture Models-Based Classifier Using Distributed Arithmetic", *13th IEEE International Conference on Electronics, Circuits and Systems 2006*, December 10th - 13th, Nice, France
9. S. Chandrasekaran and A. Amira, "Multi-Level Parallelism for Power and Energy Aware Design Verified Using Novel Functional Level Power Analysis & Modelling (FLPAM)", *NASA Military and Aerospace Applications of Programmable Devices and Technologies Conference 2006*, September 26th - 28th, Washington DC, USA

-
10. S. Chandrasekaran and A. Amira, “FPGA Implementation and Power Modelling of FWT for Pattern Recognition”, *NASA Military and Aerospace Applications of Programmable Devices and Technologies Conference 2006*, September 26th – 28th, Washington DC, USA
 11. S. Chandrasekaran and A. Amira, “FPGA Implementation And Power Modelling of the Fast Walsh Transform”, *International Conference on Field Programmable Logic and Applications (FPL)*, August 28th – 30th 2006, Madrid, Spain
 12. S. Chandrasekaran and A. Amira, “Power Reduction For FPGA Implementations : Design Optimisation And High Level Modelling”, *International Conference on Field Programmable Logic and Applications (FPL)*, August 28th – 30th 2006, Madrid, Spain
 13. S. Chandrasekaran and A. Amira, “High Speed Energy Efficient Architectures for Finite Ridgelet Transform”, *NASA Military and Aerospace Applications of Programmable Devices and Technologies Conference 2005*, September 7th – 9th, Washington DC, USA
 14. S. Chandrasekaran, A. Amira and F. Bensaali, “FPGA Implementation of Reduced Bit Plane Motion Estimation”, *NASA Military and Aerospace Applications of Programmable Devices and Technologies Conference 2005*, September 7th – 9th, Washington DC, USA
 15. S. Chandrasekaran and A. Amira, “High Speed / Low Power Architectures for the Finite Radon Transform”, *International Conference on Field Programmable Logic and Applications (FPL)*, August 24th – 26th 2005, Tampere, Finland
 16. S. Chandrasekaran and A. Amira, “An Area Efficient Low Power Inner Product Computation for Discrete Orthogonal Transforms”, *IEEE International Conference on Image Processing (ICIP)*, September 11th – 14th 2005, Genoa, Italy

-
17. S. Chandrasekaran and A. Amira, "A Novel Methodology for Temporal Partitioning in Self Reconfigurable Driven Multicontext FPGA", *Postgraduate Research Conference in Electronics, Photonics, Communications and Networks, and Computing Science (PREP) 2005*, March 30th – April 1st 2005, Lancaster, UK
 18. S. Chandrasekaran and A. Amira, "An FPGA based Accelerator for the Finite Ridgelet Transform", *IEEE International Computer Systems and Information Technology Conference 05*, July 19th – 21st 2005, Algiers, Algeria
 19. S. Chandrasekaran and N. S. Krishnan, "Double Abstraction Level Heuristic Power Optimization for Digital Signal Processors", *46th IEEE International Midwest Symposium on Circuits & Systems*, December 27th – 30th 2003, Cairo, Egypt
 20. S. Chandrasekaran and H. Pattabhiraman, "Automatic Damage Detection for Railroad Tracks by Real Time DSP Based Dual Analysis System", *46th IEEE International Midwest Symposium on Circuits and Systems*, December 27th – 30th 2003, Cairo, Egypt

Contents

Abstract	iii
Declaration	iv
Acknowledgements	vi
Author's Publication	vii
List of Abbreviations	xxvi
1 Introduction	1
1.1 Hardware Acceleration	2
1.1.1 Digital Signal Processors	3
1.1.2 Special Purpose Application Specific Integrated Circuits Hard- ware	4
1.2 Field Programmable Gate Arrays: A Review	5
1.2.1 FPGA Structure	6
1.2.2 FPGA Design Entry and Synthesis	7
1.3 Opportunities in Deploying FPGA Based Solutions	8
1.3.1 The Importance of FPGAs in Digital Logic Implementation .	8
1.3.2 In Field Design Upgradation and Future Proofing	9
1.4 Design Challenges in Deploying FPGA Based Solutions	10
1.4.1 The Need for Core Based Design	10
1.4.2 The Importance of Power Aware Design	11
1.5 Research Objectives and Motivations	16
1.6 Overall Project Strategy	16

1.7	IP Core Selection Strategy	19
1.8	Organisation of the Thesis	20
2	Literature Review	22
2.1	Introduction	22
2.2	FPGA Implementations of Selected Matrix Algorithms and Related Architectures	23
2.2.1	Existing Architectures and FPGA Implementations of the Walsh / Hadamard Transform	23
2.2.2	Architectures for Higher Dimensional Algorithms for Multiresolution Methods: The Finite Radon & Ridgelet Transforms	29
2.2.3	Architectures for Colour Space Conversion	34
2.2.4	Application Specific Hardware Implementation of Gaussian Mixture Modelling	39
2.2.5	FPGA Implementation of Finite Digital Convolution	40
2.3	Power Modelling	43
2.3.1	Versatile Place and Route Based FPGA Power Models	43
2.3.2	Probabilistic Power Prediction for the Xilinx 4000-Series FPGA	44
2.3.3	Cycle Accurate Energy Measurement and Characterisation of FPGAs	45
2.3.4	High-Level Power Modelling of CPLDs and FPGAs	45
2.3.5	Macromodels for High Level Area and Power Estimation on FPGAs	46
2.3.6	Methodology for High Level Estimation of FPGA Power Consumption	48
2.3.7	Post Synthesis Level Power Modelling of FPGAs	49
2.3.8	Power Estimation and Power Measurement of Xilinx Virtex FPGAs: Trade-offs and Limitations	50
2.3.9	Power Estimation for Cycle-Accurate Functional Descriptions of Hardware	51
2.3.10	Dynamic Power Estimation Technique for FPGAs	52

2.3.11	Power Modelling and Characteristics of Field Programmable Gate Arrays	53
2.3.12	Brief Review of selected ASIC Power Modelling Techniques . .	54
2.4	Architectural Optimisations for Performance Power Tradeoffs	56
2.5	Shortcomings and Disadvantages of Existing Work	59
2.6	Conclusions	61
3	IP Core Optimisations at Algorithmic and Behavioural Level	63
3.1	Introduction	63
3.2	An Area Efficient Low Power Inner Product Computation For Discrete Orthogonal Transforms	64
3.2.1	Mathematical Basis	65
3.2.2	Architectural Details	67
3.2.3	Key Performance Measures	67
3.2.4	Application of Scheduling	68
3.2.5	FPGA Implementation	71
3.3	Novel Sparse OBC based Distributed Arithmetic Architecture for Matrix Transforms	73
3.3.1	Mathematical Background	74
3.3.2	Proposed Architecture and OBC-DA Operation	76
3.3.3	Implementation Details and Results Obtained	78
3.3.4	Power Analysis	79
3.4	Efficient FPGA Implementation of FHT for Signal Processing	80
3.4.1	Mathematical Background	81
3.4.2	Proposed Architecture for FHT - Design and Evaluation	85
3.4.3	FPGA Implementation	88
3.5	Efficient FPGA Implementation of GMM-Based Classifier Using DA	93
3.5.1	Algorithmic Review of GMM	94
3.5.2	Architecture Description	95
3.5.3	FPGA Implementation	99
3.6	Conclusions	100

4	Architectural Level Optimisation: Parallelism, Pipelining and Systolisation	102
4.1	Introduction	102
4.2	Acceleration of Finite Radon Transform on Reconfigurable Hardware	103
4.2.1	The Finite Radon Transform: A Brief Review	105
4.2.2	Proposed Architectures for FRAT - Design and Evaluation . .	106
4.2.3	FPGA Implementation	111
4.3	Efficient VLSI Architecture for the Finite Ridgelet Transform	117
4.3.1	Mathematical Background of the Finite Ridgelet Transform .	117
4.3.2	Proposed Architectures for FRIT - Design and Evaluation . .	119
4.3.3	Implementation Results and Metrics	122
4.4	FPGA Realisation of FIR Filters by Efficient and Flexible Systolisation using Distributed Arithmetic	126
4.4.1	Formulation of the Proposed Algorithm	128
4.4.2	The Proposed Structures	131
4.4.3	Results and Discussions	135
4.5	Efficient FPGA Implementation of Colour Space Conversion	142
4.5.1	Mathematical Background	142
4.5.2	Proposed Architecture for CSC	147
4.5.3	FPGA Implementation	149
4.6	Conclusions	151
5	Performance Enhanced Voltage Scaling in FPGAs	153
5.1	Power and Energy Dissipation: An FPGA perspective	154
5.1.1	Power Dissipation Sources in Digital Circuits	154
5.1.2	Power Dissipation: FPGA Specific Details	155
5.2	FPGA Implementation: Empirical Study	157
5.2.1	Description of Methodology	157
5.2.2	Description of the Benchmark Cores	158
5.2.3	FPGA Implementation Details	159
5.2.4	Power/Parallelism/Voltage Tradeoffs	160
5.3	Conclusions	162

6	Functional Level Power Analysis and Modelling	163
6.1	Introduction	163
6.2	FLPAM: A Brief Introduction to Underlying Concepts	164
6.3	Mathematical Basis behind FLPAM	165
6.4	FLPAM Applied to Various Cores	170
6.4.1	Case Study I: Modelling the CSC Core	171
6.4.2	Case Study II: Modelling the FHT	173
6.4.3	Case Study III: Modelling the FRAT	179
6.4.4	Case Study IV: Modelling the FRIT	186
6.5	FLPAM Evaluation	193
6.6	Conclusion	194
7	Conclusions and Future Work	195
7.1	Introduction	195
7.2	Evaluation of Results and Contributions	196
7.2.1	Measurement of Success	196
7.2.2	Important Novelty Claims	197
7.2.3	Results Achieved	197
7.2.4	Limitations	199
7.3	Future Work	200
	Bibliography	202
	Appendix	ii
A	Design Entry Tool Suite and Software Packages	ii
A.1	Handel-C	ii
A.1.1	Parallel Hardware Generation	v
A.1.2	Variables	v
A.1.3	Bit Level Operators	vi
A.1.4	Channel Communications	vi
A.1.5	Memory	vii
A.1.6	External Communication	ix

A.2	Xilinx-ISE	ix
A.2.1	Specifying Design Options	ix
A.2.2	Design Translation	ix
A.2.3	Timing Constraints	xi
A.2.4	Setting Pin and Location Constraints	xi
A.2.5	Mapping, Place & Route	xii
A.2.6	Verification of Place & Route	xii
A.2.7	Estimating and analysing Power using ISE XPower	xiv
A.3	NLREG	xiv
B	Hardware Platforms for Synthesis & Implementation	xvi
B.1	RC1000 Prototyping Platform	xvi
B.1.1	Host-FPGA Communication	xvii
B.2	FPGA Devices Used in This Research	xix
B.2.1	Virtex-E FPGA	xix
B.2.2	Virtex-II FPGA	xxii
B.2.3	Virtex-II Pro FPGA	xxiv
B.2.4	Spartan-3L FPGA	xxv
B.2.5	Virtex-4 FPGA	xxv
C	Algorithms and Arithmetic	xxvii
C.1	The Finite Radon Transform and its Extension to other Higher Di- mensional Generalisations of Wavelets	xxvii
C.1.1	The Continuous Radon Transform	xxvii
C.1.2	The Finite Radon Transform and its Inverse- A Primer	xxviii
C.1.3	Ridgelets and Curvelets	xxx
C.2	Colour Spaces and Conversion Formulae	xxx
C.3	Distributed Arithmetic	xxxiii
C.3.1	Suitability of DA for FPGAs	xxxiii
C.3.2	Mathematical Formulation of DA	xxxiii
C.4	ROM Size Reduction in DA	xxxiv

D Various Compound Energy Cost Functions	xxxvi
D.1 Energy per Operation	xxxvi
D.2 Energy Area	xxxvii
D.3 Energy Throughput	xxxvii
D.4 Energy Per Pixel and Energy Per Frame	xxxviii

List of Figures

1.1	Generic FPGA structure showing internal blocks	6
1.2	FPGA design cycle	8
1.3	Gordon Moore's original graph from 1965. <i>Courtesy Intel</i>	11
1.4	Battery capacity growth in comparison to other related technology trends	12
1.5	Electronic Nose Prototype	13
1.6	Dynamic power dissipation on FPGAs	14
1.7	Various abstraction levels at which power reduction strategies can be applied	18
2.1	BWM based systolic architecture [16]	25
2.2	DA based architecture for the W-H transform [16]	26
2.3	Butterfly architecture for 4 stage WHT for the case $N=16$ [27]	27
2.4	Basic LUT cascade [29]	28
2.5	Reference architecture for the FRAT [34]	30
2.6	Memoryless architecture for the FRAT [34]	31
2.7	Generic architecture for the FRAT [37]	32
2.8	Generic tree-based architecture for the DWT [37]	32
2.9	Standard pseudocode based architecture for the FRAT [37]	33
2.10	DWT sub-block for the standard pseudocode based FRIT architecture [37]	33
2.11	Standard pseudocode based architecture for the FRAT [21]	34
2.12	DBWT sub-block based on the àtrous algorithm [21]	34
2.13	The video decoding process	35

2.14	Simple arithmetic architecture for CSC [40]	36
2.15	LUT based CSC architecture [40]	36
2.16	CSC architecture based on embedded multiplier [40]	37
2.17	Four stage YCrCb to RGB FPGA Trimedia co-processor [41]	38
2.18	Functional units in the GMM classifier [47]	40
2.19	LUT-less OBC DA architecture for 4-tap FIR filter [54]	42
2.20	r -bit serial DA one-LUT design for a four-product MAC [56]	43
2.21	Modified VPR framework used for power modelling [61, 62]	44
2.22	Real-time cycle accurate measurement system for SRAM FPGAs [64, 65]	46
2.23	High level power macromodelling for reconfigurable hardware [66] . . .	47
2.24	Macro-model characterisation procedure [67]	48
2.25	High level FPGA power estimation methodology [68]	49
2.26	High level FPGA power estimation methodology [69]	49
2.27	Measurement cycle [71]	50
2.28	Overview of the CAFD power estimation methodology [72]	52
2.29	CAD flow for activity analysis [73]	53
2.30	Overall power calculation [74]	54
2.31	Parallel implementation of a simple datapath [102]	58
3.1	Architectural block diagram	67
3.2	Unscheduled dataflow	69
3.3	Latency optimised scheduling	70
3.4	Area optimised scheduling	70
3.5	Comparison of dynamic power consumption	73
3.6	Block diagram of the overall OBC-DA-sparse matrix based DOT im- plementation	77
3.7	Architecture for the OBC-DA block for the case $N = 4$ and $W = 8$.	78
3.8	Frequency-Area trends for both architectures for different values of N and $W = 8$. Frequency is in MHz and area is represented in FPGA slices.	79
3.9	Total dynamic power dissipation for different values of N and $W = 8$.	80

3.10	Novel DA based architecture for FHT	86
3.11	DA module structure and ROM contents for $N = 8, W = 8$	87
3.12	Maximum frequency obtained for all platforms : $N = 4, 8, 16$	91
3.13	Chip diagram showing manual placement, routing and pin assignment; Virtex-E XCV2000E FPGA	91
3.14	Dynamic power dissipation for the FHT core	92
3.15	Data flow block diagram of GMM classifier. Memory units are used to store various GMM coefficients such as K, G and μ	97
3.16	DA sub-block for the VM-multiplier	98
3.17	Chip diagram showing manual placement, routing and pin assignment; Virtex-E XCV2000E FPGA	100
4.1	Reference architecture for the FRAT	107
4.2	FRAT architecture with parallel core	109
4.3	Initial array register contents for the case $p = 7$	110
4.4	Spatial domain and transformed images (a) Spatial Domain Lena (b) FRAT domain, $p = 7$ (c) Reconstructed Image (d) Spatial Domain Peppers (e) FRAT domain, $p = 17$ (f) Reconstructed Image (g) Spatial Domain Baboon (h) FRAT domain, $p = 31$ (i) Reconstructed Image	112
4.5	Comparison of EPF for the “reference” and proposed parallel FRAT architectures	115
4.6	Chip diagrams for $p = 31$ (a) Parallel architecture (b) Reference architecture; (Virtex-E XCV2000E FPGA)	116
4.7	Finite Ridgelet transform obtained by performing DWT on the FRAT vectors	119
4.8	FRIT architecture with the FRAT and Haar DWT sub-blocks	120
4.9	Comparison of EOP across different platforms	123
4.10	Chip diagram for the case $p = 31$ implemented on the Virtex-E XCV2000E FPGA	125
4.11	Chip layout for ASIC implementation for the case $p = 31$	125

4.12	Spatial domain and transformed images (a) Spatial domain Lena (b) FRIT domain, $p = 7$ (c) Spatial domain Peppers (d) Reconstructed image (e) Spatial domain Baboon (f) FRIT domain, $p = 31$	127
4.13	The DG for DA-based implementation of FIR filter. (a) The DG. (b) Function of node A. (c) Function of node B.	131
4.14	The proposed 1-D array for DA-based implementation of FIR filter. (a) The linear systolic array. (b) Function of PE. (c) Function of output cell. Δ stands for a unit delay.	133
4.15	The proposed 2-D array for FIR filter. (a) The 2-D systolic array. (b) Function of PE. (c) Function of SA cell. Δ stands for unit delay.	134
4.16	FPGA chip diagram for FIR filter realisation for $N = 64$, $M = 8$ and $L = 8$ (Xilinx XCV2000E FPGA)	139
4.17	Plot of variation of dynamic on-chip power with filter order.	140
4.18	Energy per operation of FPGA implementations FIR filter for different values of N and M	141
4.19	Energy throughput of FPGA implementations for different values of N and M	141
4.20	RGB to YCrCb conversion	142
4.21	Proposed architecture based on DA principles	148
4.22	MB structure	149
4.23	Design layout for the proposed architecture (XCV2000E FPGA)	151
5.1	Delay vs supply voltage tradeoff trends and FPGA operational conditions	157
5.2	Chip diagrams of (a) Multiplier (b) DCT cores (Virtex II XC2V1000 FPGA)	160
5.3	Power/Parallelism/Voltage tradeoff for the adder core	160
5.4	Power/Parallelism/Voltage tradeoff for the multiplier core	161
5.5	Power/Parallelism/Voltage tradeoff for the DCT core	161
6.1	Proposed design flow for FLPAM based power and energy optimised design of FPGA cores	166

6.2	A generic Data Flow Graph indicating all parameters	168
6.3	CSC Power diagram	173
6.4	Area occupied for different transform lengths : $N = 4, 8, 16$	174
6.5	CP for $N = 4, 8, 16$ at different frequencies	175
6.6	SP for $N = 4, 8, 16$ at different frequencies	176
6.7	LP for $N = 4, 8, 16$ at different frequencies	177
6.8	IpP for $N = 4, 8, 16$ at different frequencies	177
6.9	OP for $N = 4, 8, 16$ at different frequencies	178
6.10	CP chart for the FRAT core	181
6.11	SP chart for the FRAT core	181
6.12	LP chart for the FRAT core	182
6.13	IpP chart for the FRAT core	183
6.14	OP chart for the FRAT core	184
6.15	CP chart for the FRIT core	188
6.16	SP chart for the FRIT core	189
6.17	LP chart for the FRIT core	189
6.18	IpP chart for the FRIT core	190
6.19	OP chart for the FRIT core	191
6.20	ASIC SP chart for the FRIT core	193
A.1	The DK design synthesis tool for entering Handel-C code	iii
A.2	Handel-C/ANSI-C comparison	iii
A.3	Handel-C design flow	iv
A.4	The <i>PAR</i> construct	vi
A.5	Variables declaration example	vi
A.6	Bit level operators in Handel-C	vii
A.7	Channel communication	vii
A.8	RAM/ROM declaration in Handel-C	viii
A.9	Sample window displaying ISE Project Navigator	x
A.10	Setting the design options in ISE	x
A.11	Setting constraints in the timing editor	xi
A.12	PACE	xii

A.13 FPGA Editor showing a section of the clock tree in a design	xiii
A.14 XPower	xiv
A.15 NLREG	xv
B.1 RC1000 functional block diagram	xvii
B.2 Host-FPGA communication	xviii
B.3 Virtex-E architecture overview	xx
B.4 Virtex-E local routing	xxi
B.5 2-Slice Virtex-E CLB	xxii
B.6 Virtex-II platform FPGA	xxiii
B.7 Virtex-II Pro FPGA platform	xxiv
B.8 Spartan-3L Pro FPGA platform	xxv
B.9 Virtex-4 complete feature set with the chip in background	xxvi
C.1 FRAT basis function (projection kernel) for blocksize $p = 7$ and $k =$ 4. The digital line superimposed over the kernel corresponds to $l = 4$	xxx
C.2 Finite Ridgelet transform obtained by performing DWT on the FRAT vectors	xxxii
C.3 RGB to YCrCb conversion	xxxiii

List of Tables

3.1	Implementation results	72
3.2	Power dissipation data	72
3.3	On-chip dynamic power at constant frequency	80
3.4	Comparison of design metrics with existing architectures	88
3.5	Implementation results for different FPGA platforms	90
3.6	Comparison of implementation results for the Virtex-E platform	90
3.7	<i>EOP</i> metrics obtained for $N = 4, 8, 16$	92
3.8	<i>EA</i> values obtained for $N = 4, 8, 16$	93
3.9	Implementation results	99
4.1	Comparison with existing architectures	110
4.2	PSNR of images reconstructed from 8 BPP Radon domain standard images	111
4.3	Performance metrics for the Virtex-E platform	113
4.4	FPGA implementation - comparison of area (number of slices) with existing architectures	113
4.5	Comparison of performance with existing architectures for the case $p = 7$	114
4.6	EPP (nJ) for the proposed parallel FRAT core on the Virtex-E FPGA platform	114
4.7	Entropy measures for Source, FRAT, Wavelet and FRIT domain im- ages using different wavelet filters	121
4.8	Comparison of design parameters of the Radon block with existing architectures	121

4.9	Performance metrics for the FRIT IP Core on the FPGA and ASIC platforms. Maximum frequency is in MHz and throughput is denoted in Million pel/sec	123
4.10	Comparison of performance metrics with existing architectures in place	126
4.11	Key FPGA performance metrics of the proposed DA-Based FIR Filter (for Word-Length $L = 8$)	136
4.12	Comparison of performance of the proposed architecture with existing work.	138
4.13	Power dissipation of the proposed FPGA implementation of FIR filter for different filter orders and address-lengths.	140
4.14	Content of the ROM i ($0 \leq i \leq 2$)	147
4.15	Performance comparison with existing CSC cores	150
5.1	Area (slice) metrics under various levels of parallelism	159
6.1	Final estimates of the scaling coefficients	172
6.2	Coefficient values for the area and power models (Virtex-E Platform)	179
6.3	Coefficient values for area and power models for the proposed FRAT reference architecture	184
6.4	Coefficient values for area and power models for the proposed FRAT parallel architecture	185
6.5	FLPAM model accuracy for the optimised FRAT core implemented on different platforms	185
6.6	Coefficient values for area and power models for the proposed FRIT architecture on different FPGA platforms	192
6.7	Coefficient values for area and power models for the ASIC implementation of the proposed FRIT architecture	193
6.8	Comparison modelling characteristics of different approaches with FLPAM	194

List of Abbreviations

3G : Third Generation

4G : Fourth Generation

AC : Area Complexity

AG : Address Generator

ALI : Address Logic Initialiser

ASIC : Application Specific Integrated Circuit

BPP : Bits Per Pixel

CAD : Computer Aided Design

CAFD : Cycle Accurate Functional Description

CDFG : Control-Data Flow Graph

CDMA : Code Division Multiple Access

CLB : Configurable Logic Block

CMOS : Complementary Metal Oxide Semiconductor

CORDIC : COordinate Rotation DIgital Computer

CP : Clock Power

CPLD : Complex Programmable Logic Device

CSC : Colour Space Conversion

DA : Distributed Arithmetic

DBWT : Discrete Bi-orthogonal Wavelet Transform

DCT : Discrete Cosine Transform

DDF : D Flip Flop

DFG : Data Flow Graph

DFT : Discrete Fourier Transform

DHT : Discrete Hartley Transform

DMT : Discrete Multi-Tone

DOT : Discrete Orthogonal Transform

DP : Dynamic Power

DSL : Digital Subscriber Line

DSP : Digital Signal Processing/Processor

DVS : Dynamic Voltage Scaling

EDIF : Electronic Design Interchange Format

EN : Electronic Nose

FDS : Force Directed Scheduling

FFT : Fast Fourier Transform

FHT : Fast Hadamard Transform

FIR : Finite Impulse Response

FLPAM : Functional Level Power Analysis and Modelling

FMAT : FPGA MATrix Algorithms

FPGA : Field Programmable Gate Array

FRAT : Finite Radon Transform

FRIT : Finite Ridgelet Transform

GMM : Gaussian Mixture Modelling

GPP : General Purpose Processor

HDL : Hardware Definition Language

HDTV : High Definition TeleVision

HKUST : Hong Kong University of Science and Technology

HT : Hadamard Transform

I/O : Input/Output

IC : Integrated Circuit

InP : Inner Product

IP : Intellectual Property

IpP : Input Power

JHDL : Just-Another Hardware Description Language

KLT : Karhunen Loeve Transform

KNN : K-Nearest Neighbour

LE : Logic Element

LFSR : Linear Feedback Shift Registers

LP : Logic Power

LPW : Linear PieceWise

LSB : Least Significant Bit

LUT : Look Up Table

MB : Memory Block

MIMO : Multi Input Multi Output

MLP : Multi-Layer Perceptron

MRI : Magnetic Resonance Imaging

MSB : Most Significant Bit

NRE : Non Recurring Expenditure

NTT : Number Theoretic Transform

NTU : Nanyang Technological University

OBC : Offset Binary Coding

OFDM : Orthogonal Frequency-Division Multiplexing

OP : Output Power

PACE : Pinout Area Constraints Editor

PAL : Programmable Array Logic

PAR : Place and Route

PDA : Personal Digital Assistant

PE : Processing Element

PLD : Programmable Logic Device

PPCA : Probabilistic Principal Component Analysis

PRCCV : Parallel and Reconfigurable Computing for Computer Vision

PSNR : Peak Signal to Noise Ratio

R&D : Research and Development

RAM : Random Access Memory

RBF : Radial Basis Functions

ROM : Read Only Memory

RT : Radon Transform

RTL : Register Transfer Level

S2IS : Smart Sensory Integrated Systems

SA : Shift Add

SDR : Software Defined Radio

SF : Self Force

SIPOSR : Serial-In Parallel-Out Shift Register

SoC : System on Chip

SoPC : System on a Programmable Chip

SP : Signal Power

SRAM : Static Random Access Memory

StP : Static Power

SVD : Singular Value Decomposition

TC : Time Complexity

TDM : Time Division Multiplexing

VPR : Versatile Place and Route

WCDMA : Wideband Code Division Multiple Access

W-H : Walsh-Hadamard

WHT : Walsh-Hadamard Transform

WiFi : Wireless Fidelity

WiMAX : Worldwide Interoperability for Microwave Access

WTA : Winner Take All

XCL : Xilinx Coregen Library

Chapter 1

Introduction

Digital image and signal processing is one of the fastest growing areas of the electronics industry. These technologies are having an increasing impact in a wide variety of application areas such, wireless communications, biometrics, biomedical imaging, multimedia indexing storage & retrieval, computer vision and remote sensing.

State-of-the-art signal processing has changed the way we characterise and solve many applications, especially those involving increasingly sophisticated algorithms that require real-time processing of high volumes of data. Increasing demands for faster and more sophisticated processing show absolutely no sign of abating for signal processing applications.

In the context of Digital Signal Processing (DSP), the wired and wireless communications markets are exploding with a multiplicity of hi-demand standards such as Wireless Fidelity (WiFi), Worldwide Interoperability for Microwave Access (WiMAX), Third Generation (3G), Fourth Generation (4G) and Wideband Code Division Multiple Access (WCDMA) system standards, application standards such as Orthogonal Frequency-Division Multiplexing (OFDM) and Multi Input Multi Output (MIMO), as well as technology standards such as Software Defined Radio (SDR). In other areas such as classification, a number of complex algorithms including non-linear functions, statistical techniques such as mixture modelling, intelligent approaches such as neural-networks are scaling up computational complexity towards newer records.

Additionally, a continuous stream of changes is also erupting from the audio, video

and broadcast markets as well as the defense electronics industry. Within the context of image processing, emerging ultra-high complexity application areas include face recognition, 3D medical volume segmentation, new generation and higher dimensional transforms for image representation and consumer driven technologies such as High Definition TeleVision (HDTV).

The following key examples richly illustrates the challenges signal processing system developers must be prepared to address. In a Discrete Multi-Tone (DMT) modulation based Digital Subscriber Line (DSL) transceiver, it is necessary to compute transform of order as high as 4096 at sampling rate up to 44.16 MHz [1]. Similarly, in a video processor it is necessary to compute $O(10)$ of 8-point transform samples for encoding/decoding of a single image of (512×512) pixels. Moreover, the computational demand has been increasing with time along with the widespread use of multimedia communication.

Application designers face many new and difficult challenges as they attempt to deploy technology that can execute high-performance computations, manipulate larger and larger data sets and better visualise increasingly complex data. These needs must be balanced with the constraints of rising costs, shrinking space, legacy systems, mobility and power constraints and a lack of sufficient human resources. To help address these needs, appropriate choices need to be made at every stage of the design process.

Hardware acceleration has become inevitable for providing the necessary performance that is demanded for image and signal processing applications. Hardware design paradigms are also undergoing a sea change in order to address the various issues involved. There is a perceptible shift towards top-down design flow and a preference for modular, integrated and flexible solutions.

1.1 Hardware Acceleration

A lot of research has been carried out into several areas of architectural support for complex applications using DSPs and special purpose hardware [2]. Conventional approaches for hardware acceleration are listed below.

1.1.1 Digital Signal Processors

One method of increasing the performance of a General Purpose Processor (GPP) is to attach a specialised processing unit in the form of DSP. DSP processors have features designed to support high-performance, repetitive and numerically intensive tasks. Features that accelerate performance in DSP applications include:

- Single-cycle multiply-accumulate capability. High-performance DSPs often have two multipliers that enable two multiply-accumulate operations per instruction cycle;
- Most DSPs provide various configurations of on-chip memory and peripherals tailored for DSP applications. DSPs generally feature multiple-access memory architectures that enable DSPs to complete several accesses to memory in a single instruction cycle;
- Specialised execution control. Usually, DSP processors 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; and
- DSP processors are known for their irregular instruction sets, which generally allow several operations to be encoded in a single instruction. For example, a processor that uses 32-bit instructions may encode two additions, two multiplications and four 16-bit data moves into a single instruction. In general, DSP processor 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.

As a result, DSPs have been successfully used in a wide range of image processing applications [3, 4]. They provide the computing power necessary to process large amounts of data in real-time [5].

1.1.2 Special Purpose Application Specific Integrated Circuits Hardware

The other method which gives better performance for particular applications is the use of Application Specific Integrated Circuits (ASICs). They are designed specifically to perform a given computation and consequently they efficiently perform the given computation according to the application's design objectives which may be to optimise for one or more of design flexibility, performance, power consumption and area. However, after fabrication the circuit can not be altered. This forces a re-design and a re-fabrication of any part of the chip which requires modification. This is an expensive process, especially when one consider the difficulties in replacing ASICs in a large deployed system [2, 6].

The main disadvantages of this approach can be summarised in the three following points:

- Special purpose hardware has a long development time, from design through simulation and fabrication;
- It can also be expensive if it is a one-off solution or if the volume required cannot justify its fabrication costs; and
- Once this special purpose hardware is built, it is not possible to change the hardware to accommodate slightly different needs. With such a solution a new piece of hardware is usually required for each new algorithm.

A new breed of ASIC products, called “Structured ASIC”, can cut Non-Recurring Expenditure (NRE) expenses by more than 90% for derivative chips and speedup time-to-market. The underlying concept behind structured ASICs is actually fairly simple. Although there 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 Look-Up Table (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 (sea) of these tiles is then

prefabricated across the face of the chip. Structured ASICs also typically contain additional prefabricated elements, which may include configurable general-purpose Input/Output (I/O), microprocessor cores, gigabit transceivers, embedded (block) RAM and so forth. When compared with standard cell-based ASICs, structured ASICs offer shorter turnaround time and require less NRE charges 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 [7, 8].

A more recent approach, which aims to benefit from the advantages of special purpose hardware by avoiding many of its disadvantages, is to use dynamically reprogrammable hardware in the form of Field Programmable Gate Arrays (FPGAs). An overview of this technology is provided in the next section.

1.2 Field Programmable Gate Arrays: A Review

Since their introduction in 1985, FPGAs have steadily established themselves as an alternative for implementing digital logic in systems. First generation FPGAs were used to provide a denser solution for glue logic, but now they have expanded their applications to the point that it is not uncommon to find FPGAs as the central processing devices within systems.

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 4-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.

1.2.1 FPGA Structure

The basic architecture of FPGAs consists of three kinds of 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. Figure 1.1 provides a very simplified diagram of a generic FPGA architecture.

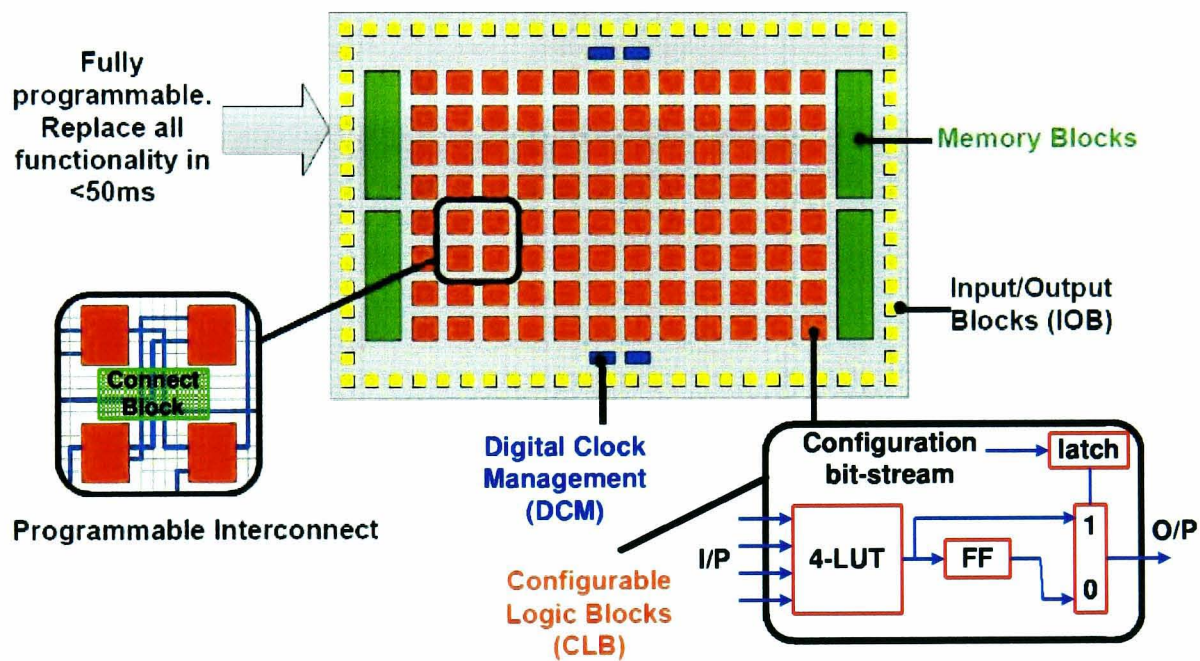


Figure 1.1: Generic FPGA structure showing internal blocks

A Basic Logic Block

As shown in Fig. 1.1, a typical FPGA has a logic block with one or more 4-input LUT, optional D Flip-Flops (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.

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. 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 Fig. 1.1 [9].

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 [9].

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. While 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 [9].

Detailed descriptions of the FPGA devices that have been used in this research are presented in Appendix B.

1.2.2 FPGA Design Entry and Synthesis

A typical design flow for FPGA design is given in Figure 1.2. It consists of a number of tools: high-level design languages, Hardware Description Languages (HDLs),

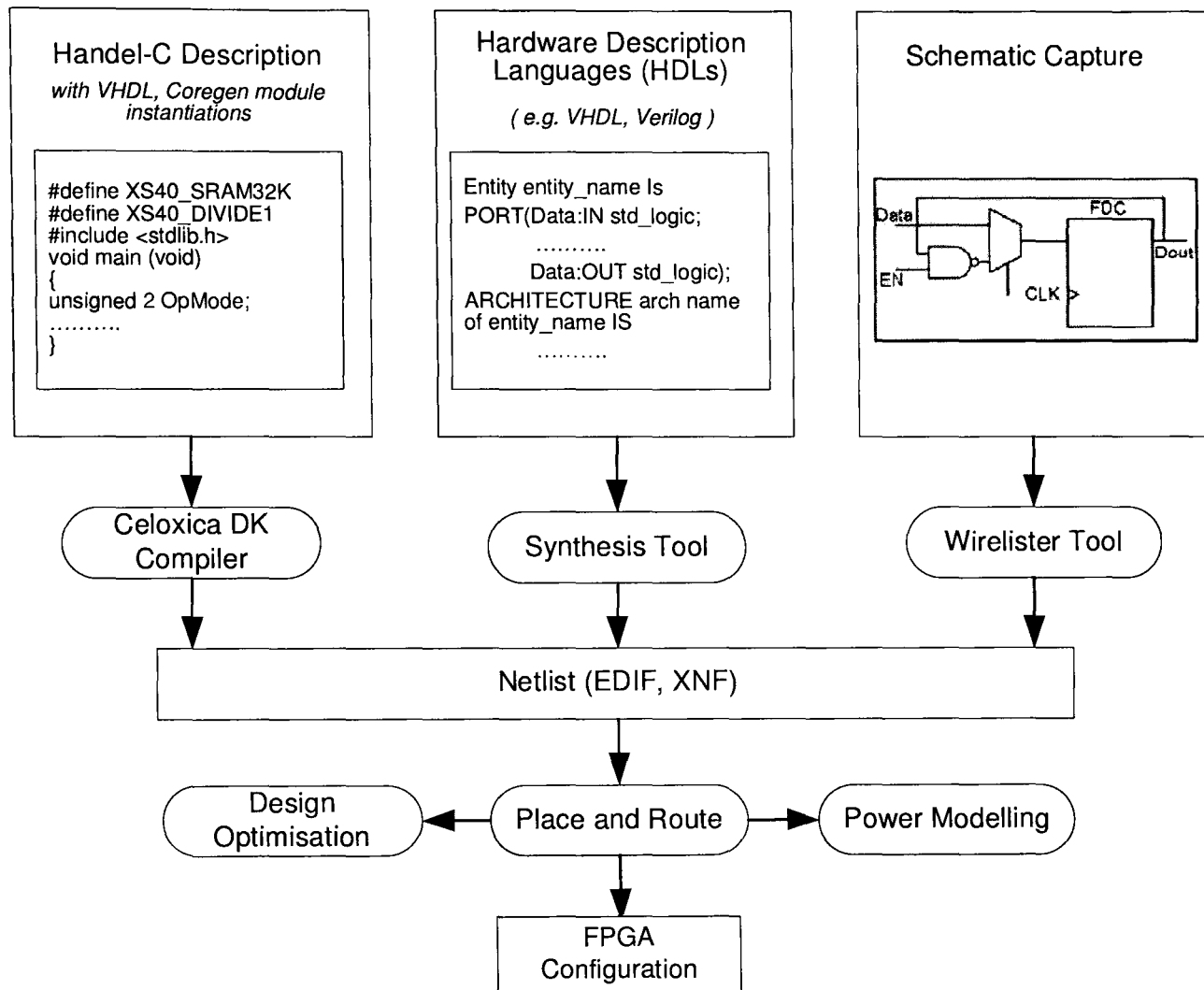


Figure 1.2: FPGA design cycle

schematic capture tools, netlist converters and Place And Route (PAR) tools.

Handel-C has been used for most implementations carried out in this research project. Full details about the various tools used to synthesise the designs presented in this research work are presented in Appendix A.

1.3 Opportunities in Deploying FPGA Based Solutions

1.3.1 The Importance of FPGAs in Digital Logic Implementation

Reconfigurable hardware in the form of FPGAs is an extremely powerful implementation approach 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 that of the GPPs. 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 relying on simulators or dead reckoning inside of design. This 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 classroom to run experiments. Companies and universities are using them in cutting-edge research on topics ranging from programming technology, cryptography to real-time systems. As a result of rapid advances in the semiconductor industry, FPGAs themselves are getting so inexpensive that some companies do not even fabricate an ASIC. They simply include the FPGA in their final product. The considerable interest in reconfigurable hardware has been highlighted by an increasing amount of research carried out in the area, coupled with the development of several commercial systems based on FPGAs. There is no doubt that this level of interest will certainly continue to grow over the next number of years. With the emergence of such reconfigurable hardware it is not surprising that there has been wide ranging research into the use of FPGAs to increase the performance of a wide range of computationally intensive applications.

With soft cores, dedicated logic, block multipliers and specialised versions available in modern FPGAs, they are being increasingly deployed in computationally intensive application areas involving image and signal processing. The regular nature of the complex computations performed repeatedly within such application areas are well suited to a hardware based implementation using FPGAs.

1.3.2 In Field Design Upgradation and Future Proofing

Future proofing in FPGA based designs is a function of two different issues:

- Field FPGA reconfiguration, driven by bug fixes, changing standards, equip-

ment upgrades and addition of services, continues to grow rapidly in importance.

- The ability to port designs from older devices to new generation FPGAs in order to take advantage of advances in device capabilities is an important advantage of FPGAs. This process allows the designer to make progressive improvements to the capabilities of a design, in step with improvements in FPGAs. For example, with the Virtex-4 [10] family, Xilinx introduced a new concept called the multi-platform FPGA which enables tuning the ratio of key features on the FPGA to match the requirements of an application domain [11].

1.4 Design Challenges in Deploying FPGA Based Solutions

1.4.1 The Need for Core Based Design

It is well known that FPGAs are now being used to build signal processing functions such as filters, matrix operations and transforms as either pre-processors or co-processors, or stand alone processing engines, thereby helping to bridge the performance gap for high-complexity and high-performance designs. In short, FPGAs are being used as full system replacements for DSPs and ASICs. However, this step-up in design complexity from their erstwhile function as simple glue logic devices brings along with it a number of design challenges.

As FPGA densities increase, in order to get complex products to market quickly, it is essential to use IP cores which are essentially pre-designed library components and are at a much higher level of complexity than the general purpose components normally supplied as part of a design environment.

The key design motivating factors for moving towards IP core based design approaches are:

- Design reuse to minimise the low-level design effort required for realising a system;

- Using pre-verified and optimised core to reduce design effort and improve overall design quality; and
- Achieve fast time to market.

1.4.2 The Importance of Power Aware Design

Power budgets are becoming increasingly stringent and need higher attention in the early stages of the design cycle. Poor design choices early on in the design cycle can result in expensive corrections and modifications. This fact, coupled with the advent of battery operated devices and increased deployment of processing in energy and thermal constrained environments such as satellites has accelerated interest in power awareness as a key requirement of the design process.

Motivations for Power Awareness

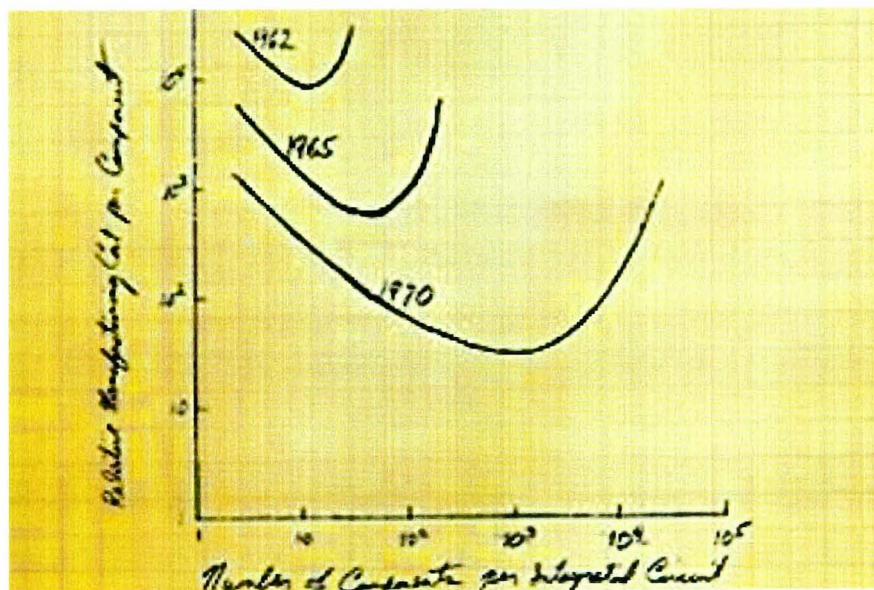


Figure 1.3: Gordon Moore's original graph from 1965. *Courtesy Intel*

The well known Moore's Law [12] has been the guiding beacon for the electronics industry. Gordon E. Moore penned down a graph (reproduced in Fig. 1.3) and postulated that number of transistors on a chip doubles about every two years. This is entirely true of FPGAs as well. FPGA vendors are embracing latest cutting edge fabrication technologies resulting in a quadrupling of FPGA capabilities every three years; and latest FPGAs have even surpassed the 1 billion transistor mark [11].

The increase in chip capacities and growth in the implementation of power hungry algorithms on FPGAs has necessitated the adoption of power aware design practices for FPGAs. Motivations for lowering power are listed below:

- Limitations of battery technology and the demands of extended mobility. It can be seen from Fig. 1.4 (reproduced from [13]) that battery energy is one of the most laggard trends in mobile computing;

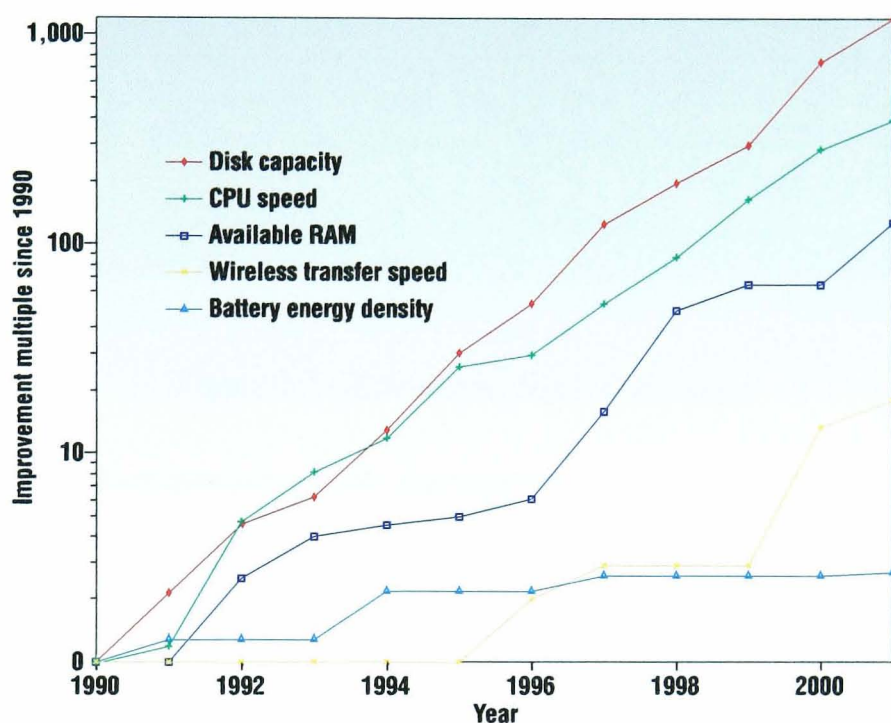


Figure 1.4: Battery capacity growth in comparison to other related technology trends

- Mobility: This is an important issue in a number of environments where un-tethered device uptime is critical. Examples of such application areas include Personal Digital Assistant (PDA) devices (e.g. Compaq H210), mobile entertainment solutions and custom purpose processing engines. An example of the latter is the FPGA based Electronic Nose (EN) that has been developed in a joint project led by Hong Kong University of Science and Technology [14]. A picture of the prototype is presented in Fig. 1.5. Further details can be obtained from Chapter 4.
- Thermal stability and noise immunity:
 - Drift: Passive components, such as resistors, capacitors and inductors typically change in value with temperature - unpredictable operation;

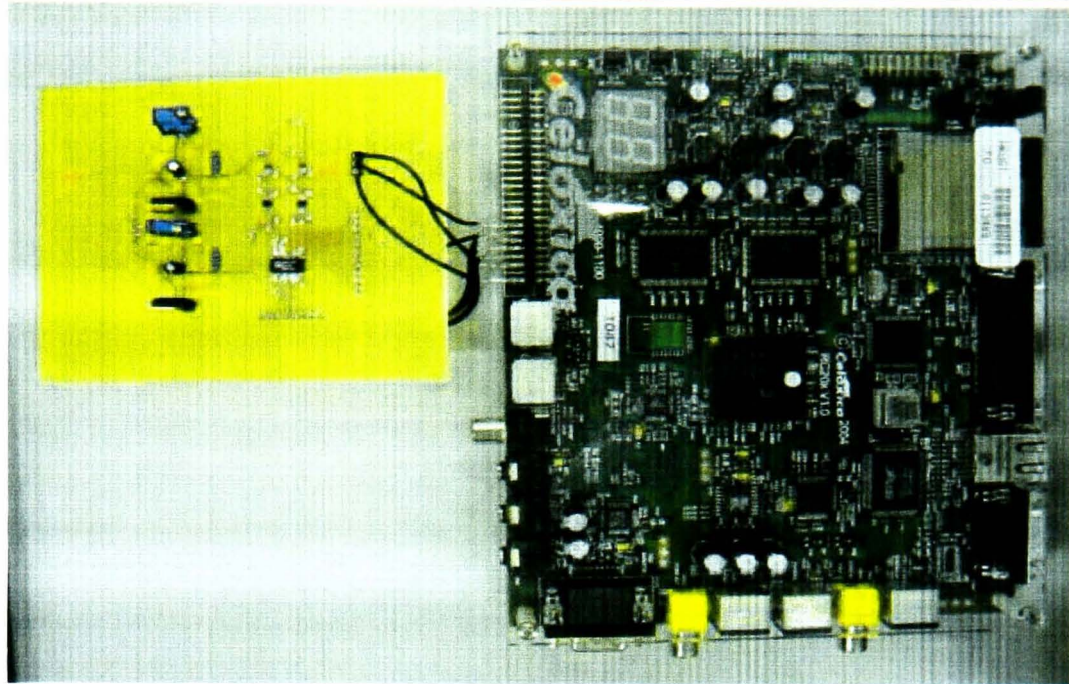


Figure 1.5: Electronic Nose Prototype

- Leakage currents typically increase;
- Applications: Satellites and network chips are very susceptible to such effects, due to lower tolerance and high sensitivity to the effects of drift;
- Expensive: Cooling technology required to handle excess heat generated; and
- Interference can be unacceptable in sensitive environments such as satellites and Magnetic Resonance Imaging (MRI) scanners where FPGAs are used heavily for processing large volumes of data.

- ● Environmental concerns:

- Energy star compliance; and
- Global warming.

Power Dissipation: FPGA Specific Details

Compared to ASICs and other custom chips, FPGAs contain long routing tracks with significant parasitic capacitance. During high-speed operations, the switching activity on these long routing tracks causes significant power dissipation. Power dissipation calculations for FPGAs are similar to other complementary metal-oxide

semiconductor application-specific integrated circuit (Complementary Metal Oxide Semiconductor (CMOS) ASIC) devices. The total power usage of an FPGA device (P_{Total}) can be broken down into total Static Power (StP) and total Dynamic Power (DP):

$$P_{Total} = StP + DP \quad (1.1)$$

Static Power Dissipation in FPGAs The StP of an FPGA is proportional to the static current I_{dd} - the current that flows regardless of gate switching (transistor is 'on' or 'off'). This is otherwise called the quiescent power. DC power dissipation can be estimated by the worst-case equivalent equation: $StP = V_{dd}I_{dd}$. StP is inherently dependant on the architectural layout of the FPGA itself and is technology dependant. As such, it cannot be controlled by the FPGA based designer and will not be addressed in this work.

Dynamic Power Dissipation in FPGAs

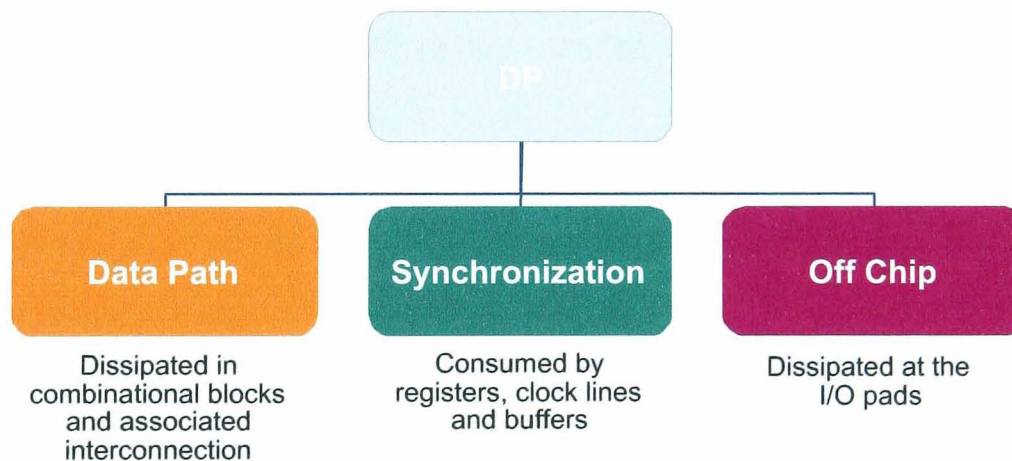


Figure 1.6: Dynamic power dissipation on FPGAs

The DP consumption of FPGAs can be separated into data-path, synchronisation and off-chip power (Fig. 1.6). Datapath power corresponds to combinational blocks and associated interconnection power. Synchronisation power is the power consumed by registers, clock lines and buffers. Datapath and synchronisation power are together termed as on-chip DP. Off-chip power is the fraction dissipated in the circuit output pads. Knowledge of the relationship between these components for a given FPGA technology is fundamental in calculating the power consumption of

an FPGA-based system. For the case of Xilinx SRAM FPGAs, DP is classified into clock, signal, logic, input and output power. Power consumption of the data-path interconnection (programmability) is the highest of the three parts and increases linearly with the input clocking frequency. Various techniques (including pipelining and partitioning, clock gating, bus multiplexing, asynchronous design and clock frequency reduction) can be applied to an FPGA design to reduce this power consumption. For reasons of clarity, it is worthwhile to mention that in this work henceforth, terms 'power' or 'dynamic power' are used to indicate dynamic on-chip power only.

Power Modelling as a Critical Step in Achieving Power Awareness

Hardware implementations of low power applications necessitate design space exploration of an optimal solution. Design improvement is typically an iterative process that must take into account optimisation strategies at all feasible levels of abstraction. The most effective strategies must then be selected by taking into account feedback on the impact of the different choices on a level-by-level basis, instead of just at the very end of the flow. This enables us to shorten the overall design time in the development of chips with one or more constraints. However, it requires the development of power modelling tools that provide reliable estimates of the power and energy metrics, to enable the designer to make the right design choices while optimising the IP core.

Models have always been important for electronic system design at all levels, whether at the component (Integrated Circuit (IC) design), board or system level. Lately, with deep sub-micron CMOS technology, the power levels of large FPGAs have become a strong function of application conditions such as supply voltage, pattern, clock frequency and output loading. Large FPGAs can easily exceed 5-10W power dissipation as a function of the foregoing variables.

Currently, most approaches to hardware power estimation and modelling operate at the Register-Transfer Level (RTL) or lower levels of design abstraction. Attempts at power estimation for functional descriptions have suffered from poor accuracy

because the design decisions performed during their synthesis lead to an unavoidable, large uncertainty in any power estimate that is based solely on the functional description.

1.5 Research Objectives and Motivations

The considerable interest in reconfigurable hardware has been highlighted by the increasing amount of research carried out in the area, coupled with the development of several commercial systems based on FPGAs . There is no doubt that this level of interest will certainly continue to grow over the next number of years.

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.

However, wider acceptance of FPGAs as a replacement to traditional hardware and software platforms for performance enhancement acceleration depends on the ability of FPGA based designers to learn from experiences in the ASIC design domain and to evolve solutions to the greatest FPGA design challenges for the next decade - adopting higher level design paradigms and simultaneously addressing the challenges of power consumption. Keeping these challenges in perspective, the key objectives of this research project can be broadly summarised as follows:

- To design and implement scalable, parameterisable, efficient and novel IP cores for a range of 1-D and 2-D transforms and matrix operations; suitable for use in both general and special purpose signal, image and video processing problems, through the application of optimisation strategies at various abstraction levels
- To develop a novel and accurate high level power modelling methodology suited for optimisation and power aware deployment of FPGA based IP cores.

1.6 Overall Project Strategy

In order to ease the development of efficient FPGA based solutions for these applications, the Parallel and Reconfigurable Computing for Computer Vision (PRCCV)

Research Group at Brunel University has developed a number of high performance cores that have been grouped together into a library called the FPGA MATrix Algorithms (FMAT) IP core library [15]. Existing cores in the FMAT library (prior to the commencement of the work in this thesis) can be broadly classified into image and signal processing transforms and other matrix operations including decompositions. The library includes transform based cores such as the Fast Hadamard Transform (FHT) [16, 17], Discrete Hartley Transform (DHT) [18], Fast Fourier Transform (FFT) [19], Discrete Wavelet Transform (DWT) [20], Finite RIDgelet Transform (FRIT) [21], etc. Matrix operations based cores in this library include matrix multiplication and Colour Space Conversion (CSC) [22]. Decomposition related cores that have been implemented include Singular Value Decomposition (SVD), QR and LU. Historically, computational efficiency and compact area footprint were the key objectives in the design of the cores in this library. While high performance hardware systems are essential for performing large scale number-crunching, a contrary design goal of reducing power consumption is rapidly assuming importance. In this work, we seek to balance both these issues. This research work is primarily concerned with the following tasks:

- Optimising selected image and signal processing cores from the FMAT library with particular emphasis on applying techniques for minimising power and energy consumption;
- Design and implementation of additional novel and optimised cores to be added to this library; and
- Development of a high level power modelling technique to accurately model the interplay of various design and performance metrics and their influence on power and energy dissipation.

Design optimisation through common-sub expression reduction, design space exploration for multiple transform lengths and power modelling has been performed on the FHT core. The Finite RAdon Transform (FRAT) and FRIT cores have been completely redeveloped by applying a number of architectural modifications. CSC

has been re-implemented for higher performance and has also been modelled. A novel and optimised architecture for Gaussian Mixture Model (GMM) based implementation for an EN application has been developed in collaboration with the Smart Sensory Integrated Systems (S2IS) Research Lab at the Hong Kong University of Science and Technology (HKUST). Another important addition to the library is the development and implementation of novel systolised architectures for Finite Impulse Response (FIR) filters and circular convolution in collaboration with Nanyang Technological University (NTU).

The relative placement of different abstraction levels at which design optimisations are possible are indicated pictorially in Fig. 1.7. The area above the dotted horizontal line indicates the abstraction levels at which the FPGA based designer can apply optimisation strategies. The area below the dotted line denotes abstraction levels which cannot be controlled at design time, due to the fact that the FPGA fabric itself cannot be modified.

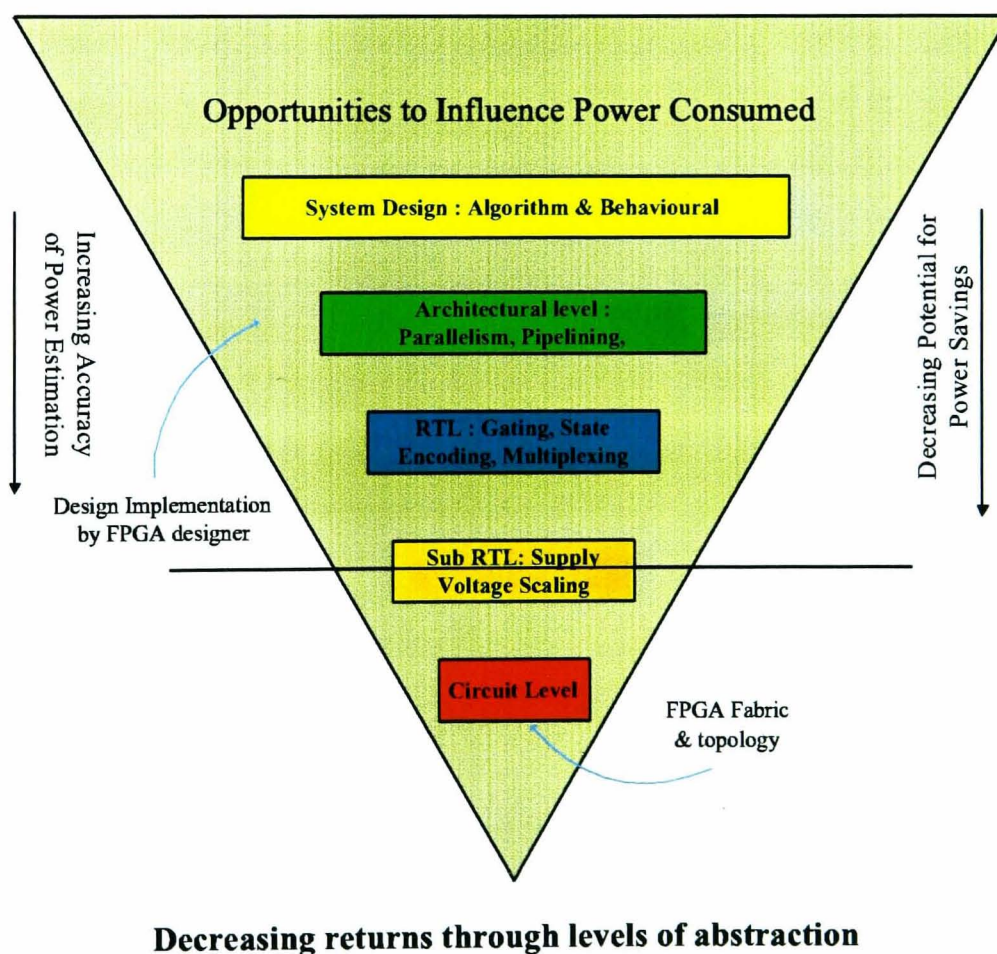


Figure 1.7: Various abstraction levels at which power reduction strategies can be applied

1.7 IP Core Selection Strategy

At this juncture, it is important to highlight that the choice of the above mentioned cores for optimisation and development in this research work is influenced by the following factors:

Application Driven IP core selection The PRCCV Research Group has worked primarily on developing multiresolution image processing frameworks for applications such as face recognition, medical image analysis, image and video compression and information retrieval. The choice of cores to be optimised, developed and modelled is influenced by the fact that they are used as important computational sub-blocks in these applications. For example, the FHT has a wide range of image and signal processing applications including compression, error correction, spectral analysis and pseudo noise sequence generation. The FRIT and its higher dimensional derivatives including curvelets and contourlets are highly suitable for a number of image processing applications such as detection of line singularities and contiguous edges (useful for compression and feature detection), image denoising and image segmentation (useful in medical imaging). CSC allows us to represent image data in a more compact and efficient manner and is used as an important sub-block in image compression. GMMs are a special class of statistical functions which are widely used for classification applications in image and signal processing. Fast convolution is a fundamental operation in numerous signal processing applications, particularly in the fields of spectral analysis and communications.

Group Strategy Driven IP core Development Some cores in the FMAT library are already highly optimised, whereas the implementations of others provided further opportunities for optimisation. New cores presented in this research work have been carefully chosen as a logical extension to available cores in the FMAT library in order to extend and improve the functionality, performance and range of applications that can be built using this library.

Client and Collaboration Driven IP core Development The FMAT cores and related applications have been developed in response to demand from the re-

search and industry community for efficient and high performance hardware implementations. New cores arising out of collaborative efforts have been carefully selected in line with joint R&D goals and synergistic requirements.

Mathematical modelling techniques for characterisation of power dissipation of VLSI designs implemented on various platforms have been gaining a lot of attention over the last decade. The development of a novel high level modelling technique particularly suited for IP cores implemented on SRAM FPGAs called Functional Level Power Analysis and Modelling (FLPAM) is an important contribution of this work. A number of existing power modelling approaches have been analysed in order to set the context for highlighting the key advantages of the proposed FLPAM methodology.

1.8 Organisation of the Thesis

The structure of this thesis is as follows. Chapter 2 takes a closer look at the most recent architectures and systems for the various IP cores that have been developed in this work including Dot Product, GMM, CSC, FHT/FWT, FRAT, FRIT and Circular Convolution. A brief overview of existing power modelling techniques for ASICs in general, and a detailed and thorough of various power modelling approaches for FPGAs are also provided in this chapter. Chapter 3 presents an analysis of the application of algorithm level techniques for IP core optimisation. The cores that have been targeted for optimisation in this Chapter are novel transformation techniques for Distributed Arithmetic (DA) implementations of dot product, FHT and GMM. Architectural level techniques for IP core optimisations for high performance and power aware implementations of FRAT, FRIT, circular convolution and CSC are discussed in Chapter 4. Chapter 5 is concerned with a discussion on performance enhanced voltage scaling for FPGAs. In this Chapter, standard arithmetic circuits such as adders, multipliers and Discrete Cosine Transform (DCT) are implemented on FPGAs using a combination of performance enhancement measures and voltage scaling. In Chapter 6, a novel functional level power analysis and modelling methodology is presented. The complete mathematical formulation of FLPAM is

described, and a number of benchmark circuits developed in the previous chapters are modelled. Concluding remarks and opportunities for future work are presented in Chapter 7.

Chapter 2

Literature Review

2.1 Introduction

Image and signal processing algorithms are commonly used in the application areas such as telecommunications, speech and audio-visual processing. These areas require enormous computing power. A close examination of the algorithms used in these and related applications shows that these operations are also data intensive. Hence a number of performance metrics including area occupied, maximum frequency, throughput, power dissipation, etc. need to be considered while validating the work presented in this thesis against comparable existing work. In the area of power modelling, the nature of comparison needs to include quantitative and qualitative measures such as model accuracy, parametrisation, scalability and model abstraction level among others. In order to benchmark our Ip cores and modelling methodology with the best in class, a thorough and extensive literature survey has been conducted and updated throughout the period of research in this work.

Consequently, this chapter is organised as follows. Existing architectures and implementations for a number of matrix algorithms, most of which have been mentioned above, are presented in Section 2.2. An overview of the existing power modelling methodologies for FPGAs is presented in Section 2.3. Related work on voltage scaling is reviewed in Section 2.4. A synopsis of the shortcomings of existing work and concluding remarks are provided in Sections 2.5 and 2.6 respectively.

2.2 FPGA Implementations of Selected Matrix Algorithms and Related Architectures

In this section, existing architectures and implementations for a number of matrix operation based algorithms including FHT, FRAT, FRIT, CSC and GMM have been collated.

2.2.1 Existing Architectures and FPGA Implementations of the Walsh / Hadamard Transform

Number Theoretic Transforms (NTTs) are classes of transforms that provide natural solutions for problems like rounding errors and complex computations in sinusoidal transforms such as the FFT etc. This is because NTT can be computed without general multiplication and do not depend on special basis functions. The FHT is a generalised class of the Fourier transform performed on the two-element additive group of $Z/2$ [23]. FHT is sometimes known as the “poor man’s FFT”, a sobriquet it gained because of the fact that it similar to NTTs in computational requirements, but like the FFT, has a number of applications. The FHT is faster than sinusoidal-like transforms as it can be decomposed into a combination of additions and subtractions only [23,24]. It can also be formulated as a matrix-vector multiplication similar to the Discrete Fourier Transform (DFT) and it has a fast algorithm which has $N\log_2 N$ additions and subtractions for N -point input samples [25]. A survey of literature shows that a number of existing architectures for FHT and its hardware implementation have been presented. The Walsh transform is similar to the Hadamard Transform (HT), with minor differences only in the order of transform coefficients. Hence they are grouped together and are also referred to in literature as the Walsh-Hadamard Transform (WHT). Consequently, existing architectures for all WHTs suitable for FPGA implementation have been listed out in this subsection.

Architectures for the Walsh-Hadamard Transform

Systolic array based design paradigms include multiple data counters and support data parallelism. Systolic array based designs are highly suitable for implementation on FPGAs because of their regular structure and massive parallelism capabilities.

A chip for a systolic array based implementation of the HT was designed by the University of South California and fabricated by MOSIS [25]. Its computation requires $(2N - 1)$ clock cycles while its latency is N cycles. However, the addition implemented in the array is at word level. Thus, the time required is proportional to $(W + \log_2 N)$, where W and N are the wordlength and transform length respectively.

A bit level systolic architecture is reported in [24]. This has a computation time of $(2N - 1)(W + \log_2 N)$ clock cycles with a latency of $N(W + \log_2 N)$ cycles.

Baugh Wooley Based Systolic Architecture for the Walsh-Hadamard Transform

In [16], a systolic approach for high throughput implementation of the FHT is presented. The multiplication of the Hadamard matrix (H) and input vector (X) is expressed in two's complement representation and written in a form which involves only positive bit products and the matrix-vector product is then computed using two's complement multiplication based on the Baugh-Wooley (BW) algorithm. This is performed using a serial-parallel multiplier as shown in Fig. 2.1 in the case of $n=4$, where n is the wordlength. The superscript m in the term h^m in Fig. 2.1 refers to the bit position of 2's complement notation of the Hadamard coefficient being operated upon by the B-W Multiplier (BWM). The logic unit is used to obtain the partial products and adding the extra one in the fifth and the eighth cycle through the OR and the AND gates respectively using the two control signals S1 and S2 for the BWM. The adder unit is used for the addition of the partial products and the carry propagation. The time and area complexity of the structure are $n(N + I)T$ and $2N$, respectively (where T is the clock cycle fixed by the total gate delay of the BWM). The systolic architecture presented in 2.1 is designed and targeted to the Xilinx XCV1000E FPGA of the Virtex-E family [26]. This architecture occupies

188 FPGA slices and operates at a maximum frequency of 31 MHz.

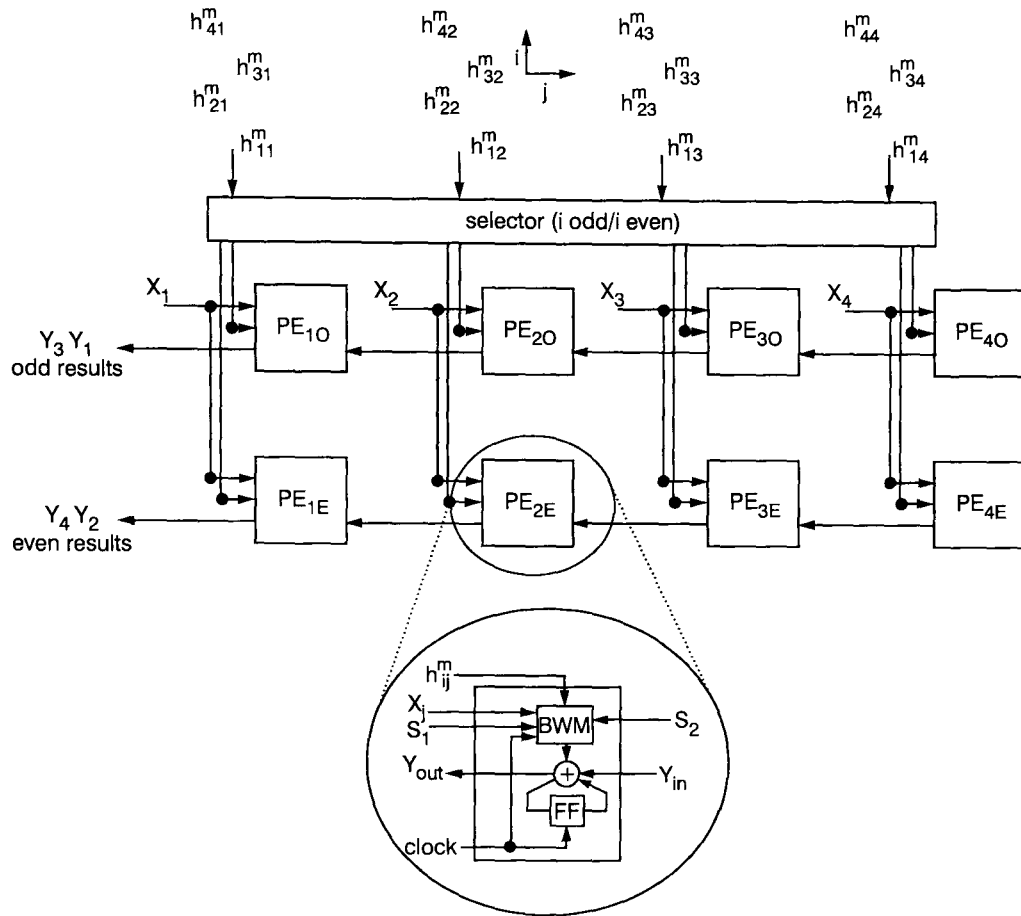


Figure 2.1: BWM based systolic architecture [16]

Sparse Matrix Distributed Arithmetic based Architecture for FHT

A second approach for implementing the FHT based on DA principles for inner product implementation has also been presented in [16]. To reduce the number of arithmetic operations and to speed up the process, the symmetry in the FHT matrix coefficients and a sparse matrix factorisation are exploited in this architecture. The architecture for transform length $N = 8$ is presented in in this work. Four separate ROM-Accumulate blocks calculate the eight transforms as follows: a butterfly structure of bit-serial adders and subtractors is used to generate the elements of the input matrix as shown in Fig. 2.2. This architecture has an effective computational complexity of $2n$ where n is the wordlength. This DA based architecture is designed and implemented on the Xilinx XCV1000E FPGA of the Virtex-E family [26]. For $N, n = 8$, it is reported that a maximum frequency of approximately 50 MHz is achieved and the design occupies around 75 FPGA slices.

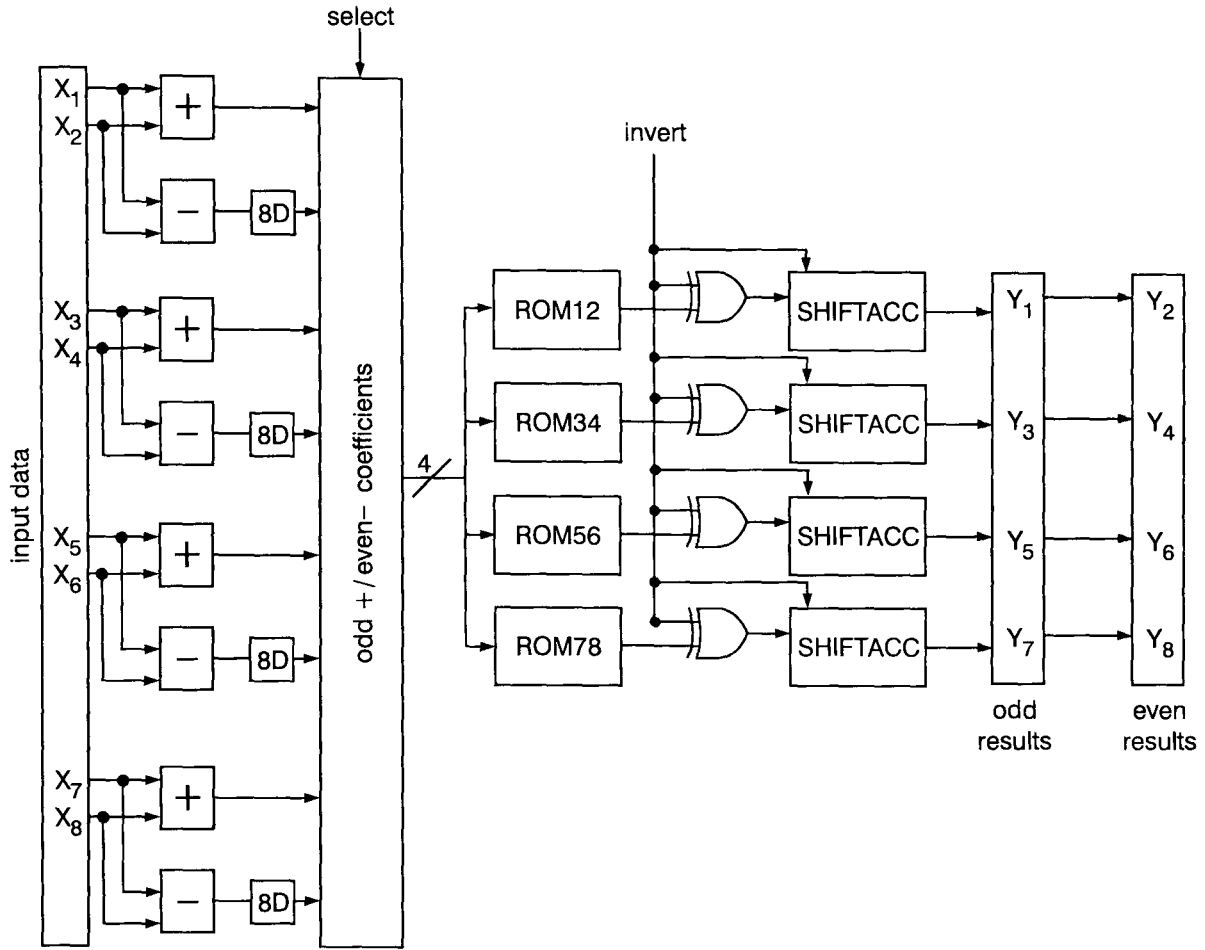


Figure 2.2: DA based architecture for the W-H transform [16]

Butterfly Architecture for the Fast Walsh-Hadamard Transform

An FHT based architecture for wideband Code Division Multiple Access (CDMA) applications is presented in [27] (Fig. 2.3). The architecture is based on a multi stage butterfly design that operates on single bit inputs. There are a number of stages in the FHT design depending on the length of the WHT sequence. Each stage has an upper and a lower input terminal. The output of each stage of the FHT is then given to the next stage. The number of bits in the counter depends on the number of stages which in turn depends on the length of Walsh-Hadamard (W-H) sequence to be used. If there are N W-H chips then the counter length must be $\log_2 N$ bits (where N denotes transform length). The length of the shift register in each of the stages s of the design is given by the following relation $(N/4)/2^s$. This design has a computation time of N clock cycles and a latency of $(\log_2 N + 1)$. In addition, the architecture is a straightforward implementation of the decomposition of the Hadamard matrix, without any algorithm or architecture level optimisations.

It is worth mentioning that the architecture accepts single bit input values, thereby reducing the throughput by a factor of W , where W is the wordlength. The 16 chip FHT structure is implemented on the Xilinx Virtex-E XCV1000E FPGA [26] and occupies 71 slices with a maximum operating frequency of about 36 MHz. It must be highlighted that the architecture is for single bit FHT only which must be considered while calculating effective throughput.

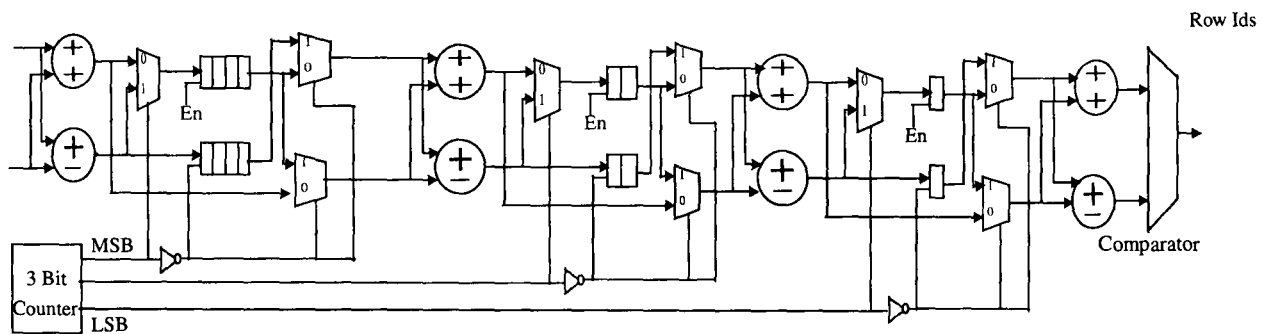


Figure 2.3: Butterfly architecture for 4 stage WHT for the case $N=16$ [27]

Hadamard Based Image Encoder

A low gate image encoder based on the WHT used as a replacement for the DCT is presented in [28]. The architecture is a straightforward implementation of the Hadamard matrix, with Read Only Memory (ROM) based control logic for the adder/subtractor block. The operation of the WHT block is divided into two consecutive passes (modes). In the first mode a row pixels of an image block are stored in eight registers in eight clock cycles. This stored data (pixels) is then operated upon, using the add/subtract units. The choice of add or subtract in each operation is instructed by the WHT coefficients stored in the ROMs. The resulted data, known as a row operation, is stored in the RAM. In the second mode the data stored in the RAM is sent to the registers and follow similar operations after which final transformed 8×8 block is produced. The time complexity of the design in [28] is $O(N^2)$. Individual blocks in the design have been constructed and tested using Xilinx XC4010E FPGA. The construction of the entire design is mapped into Xilinx XC4053XL-740 FPGA and yields a maximum clock speed of about 6 MHz.

Unified Algorithm and Architecture for the Walsh and Related Transforms

A Unified algorithm to generate Walsh functions in four different orderings and its programmable hardware implementation is presented in [29]. This architecture is designed using transformation tree based butterfly structure networks to implement the transform for lengths $N < 7$ and Time Division Multiplexing (TDM) for greater values of N , where N is the natural logarithm of the size of the transform. The authors have mapped the combinational part of the function generator into LUT cascades, where each cell has at most 6 inputs and 6 outputs. The basic structure of the LUT cascade that is used is shown in Fig. 2.4. The target device used by the authors to prototype their design is the Altera FLEX10K EPF10K10LC84-3 FPGA [30] which contains 576 Logic Elements (LEs) and 59 user IO pins. Design optimisation is performed using by Synplify Pro and mapped using Quartus II. The implementation for $N = 8$ requires 102 LEs and the minimum delay achieved is 25.7 ns. As in the case of [27], this architecture has also been used for single bit WHT implementation only.

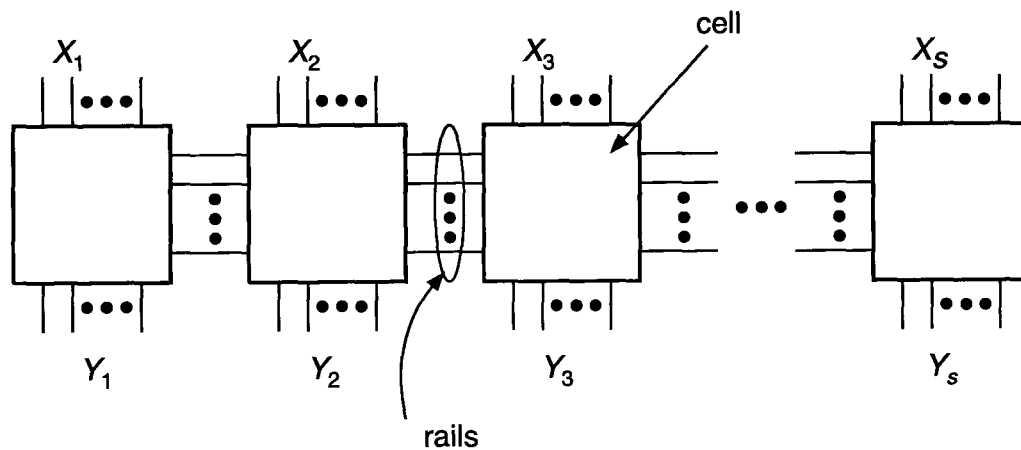


Figure 2.4: Basic LUT cascade [29]

2.2.2 Architectures for Higher Dimensional Algorithms for Multiresolution Methods: The Finite Radon & Ridgelet Transforms

The search for “true” multidimensional bases beyond wavelets that are directional in nature has led to the development of ridgelets and similar transforms. Recently, the curvelet and ridgelet transforms [31–33] have been generating a lot of interest due to their superior performance over wavelets. While wavelets have been very successful in applications such as denoising and compact approximations of images containing zero dimensional (point) singularities, they do not isolate the smoothness along edges that occurs in images because they lack flexible directionality. Wavelets are thus more appropriate for the reconstruction of sharp point-like singularities than lines or edges. These shortcomings of wavelets are well addressed by the ridgelet and curvelet transforms, as they extend the functionality of wavelets to higher dimensional singularities and are effective tools to perform sparse directional analysis. An orthonormal, digital and fully invertible form of Ridgelets, called the FRIT was first proposed in [31].

The FRIT has gained attention recently due to its directional nature, resulting in better performance in applications such as compression and denoising. The FRIT is generated by performing a FRAT on a non-dyadic sized block of the source image followed by a DWT operation. Existing FPGA based implementations of the FRAT and FRIT have been presented in the following subsections.

Reference and Memoryless Implementation of the FRAT

Two architectures for the FRAT and their FPGA implementation have been described in [34]. The first architecture shown in Fig. 2.5 is called a *reference FRAT architecture* and is a direct hardware implementation of a suitable modified variant of the standard FRAT pseudocode [35] [Appendix C]. The architecture comprises an address logic initialiser, multiplexer, accumulators and two memory blocks for storing transform vectors.

The second architecture in [34] (Fig. 2.6) is denoted as a Memoryless FRAT archi-

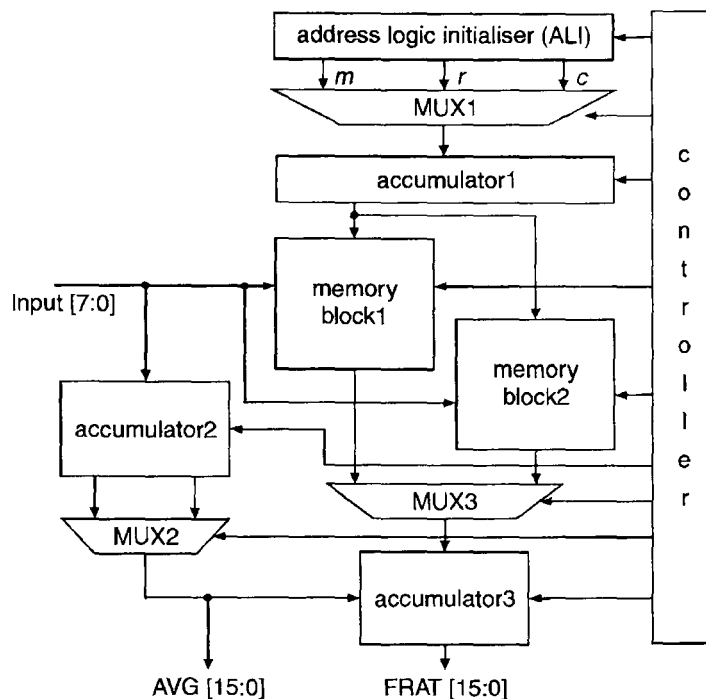


Figure 2.5: Reference architecture for the FRAT [34]

ture which operates in a parallel manner with p times the throughput of the first architecture, where p is the blocksize. Address Logic Initialiser (ALI), wide multiplexer and adder blocks are used as sub-blocks in this architecture. The ALI drives the Address Generators (AG) that in turn generate signals to control the address bus. It is worth mentioning that the multiplexer operates on all the p^2 pixels simultaneously in this architecture and hence scalability may not be inherently feasible as a result of wiring complexities.

Both architectures have been designed using Verilog HDL and synthesised by Xilinx ISE development tools using the Virtex-II device family [36]. The *reference architecture* occupies 159 slices and provides a maximum operating frequency of 100 MHz with a power consumption figure of 114.98 mW at 50 MHz. The memoryless architecture requires 558 FPGA slices and provides a maximum operating frequency of about 82 MHz. Power is consumed at the rate of 253 mW at 50 MHz. Xilinx XPower [11] is used for power estimation assuming a 1.5 V supply with a capacitive load of 10 pF.

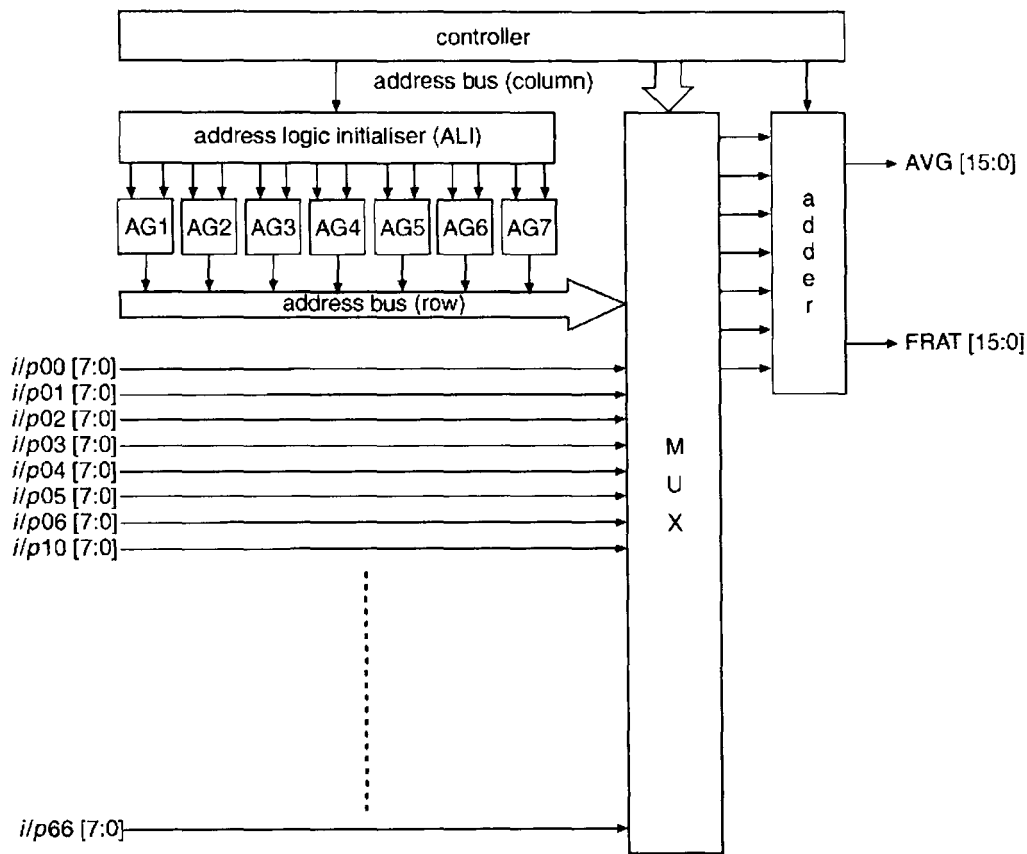


Figure 2.6: Memoryless architecture for the FRAT [34]

Generic and Pseudo-code Based Implementation of the FRIT for Curvelet

In [37] two architectures have been presented for the implementation of FRIT. The first architecture uses a *generic* block (Fig. 2.7) that uses a combination of LUTs, matrix of accumulators and multiplexers to perform the FRAT. This is followed by a tree structure based architecture for implementing the DWT (Fig. 2.8). The time complexity of this design is $O(p^4)$, where p is the block size. The maximum frequency for the forward transform using this architecture is reported to be 33MHz. The second FRIT architecture presented in [37] is based on the standard FRAT pseudocode provided in [35] and has a core time complexity of $O(p^4 \cdot (p + 1))$. This architecture is presented in Fig. 2.9. The “Radon Transform Module” in this architecture contains a “FRAT Calculator” which uses address generators, accumulators, RAMs and local control logic to perform the FRAT iteratively. Both the Haar and the à trous algorithm have been implemented for the wavelet sub-section of the FRIT. The block diagram of the wavelet module is reproduced in Fig. 2.10. Each ridgelet block in this architecture consisting 1 Radon and 1 Haar sub-block occupies a total area of 828 FPGA slices.

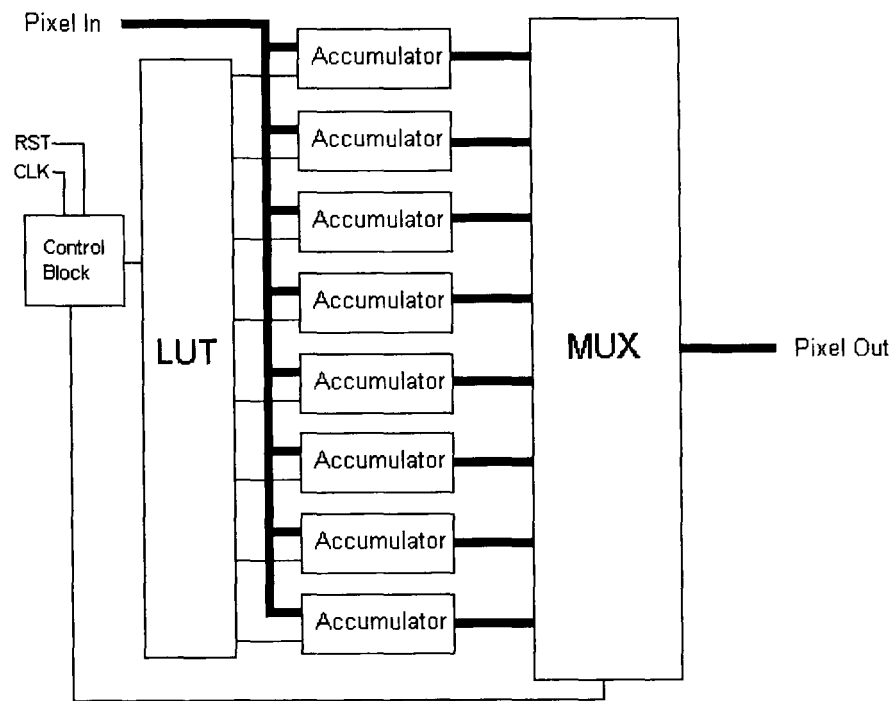


Figure 2.7: Generic architecture for the FRAT [37]

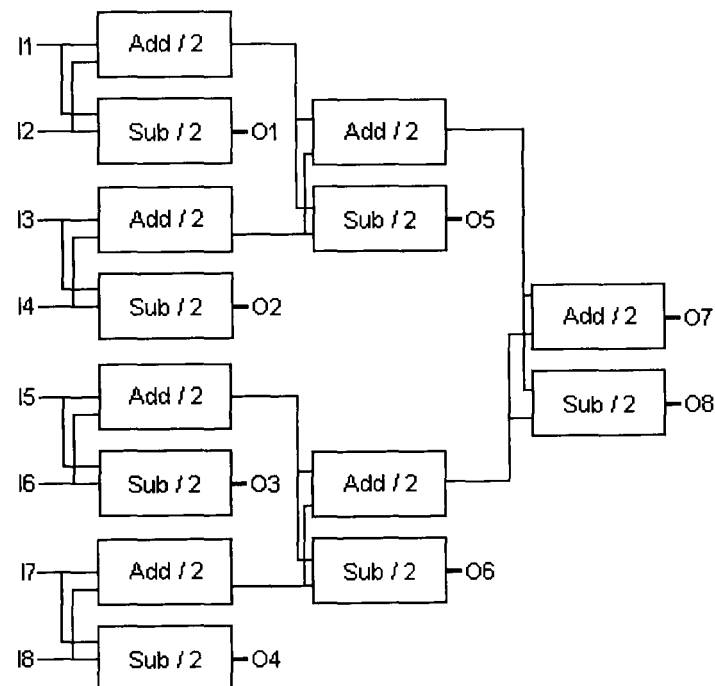


Figure 2.8: Generic tree-based architecture for the DWT [37]

FPGA Prototyping of the FRIT Based on Standard FRAT and Àtrous

The architecture for the FRIT that is proposed in [21] uses FRAT and 1-D Discrete Biorthogonal Wavelet Transform (DBWT) as building blocks. The architecture of FRAT that is implemented in [21] is a straightforward implementation of the FRAT pseudo-code [35] [Appendix C]. The address logic initialiser along with controller

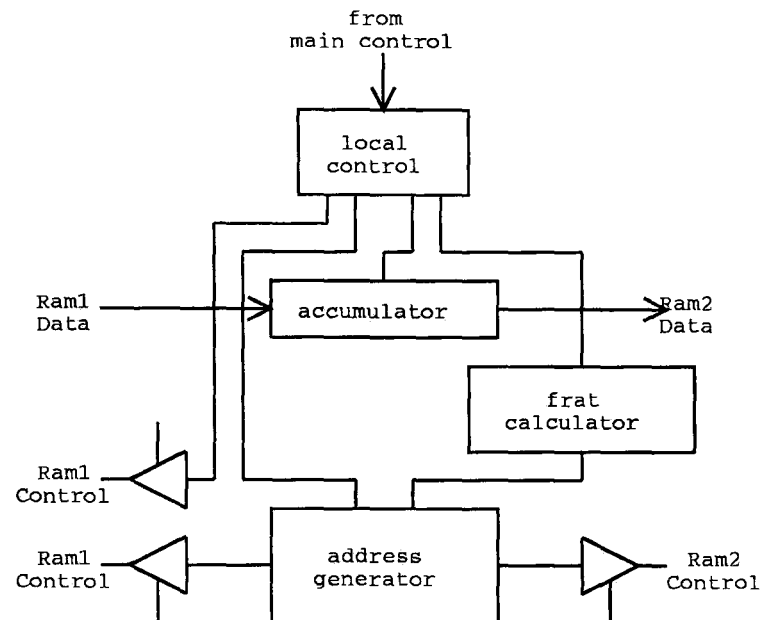


Figure 2.9: Standard pseudocode based architecture for the FRAT [37]

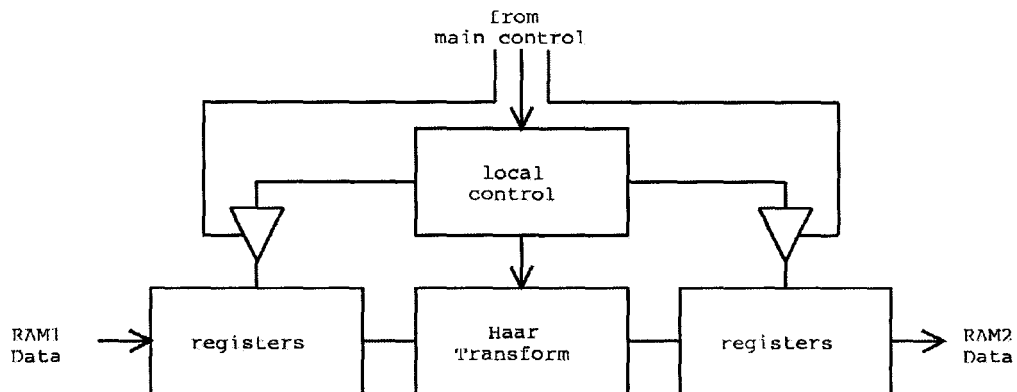


Figure 2.10: DWT sub-block for the standard pseudocode based FRIT architecture [37]

block constitute the address generator 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. Architectural details of the FRAT and the DBWT sub-blocks in [21] are pictorially presented in Figs. 2.11 and 2.12 respectively. In order to verify the performance of these architectures, the designs have been ported to a Xilinx Virtex-II FPGA [36] chip using Handel-C [38]. The implementation results show that the core speed for the FRIT architecture in [21] is around 100MHz and it occupies 491 Slices for an input image size of 7×7 with a distributed RAM based implementation.

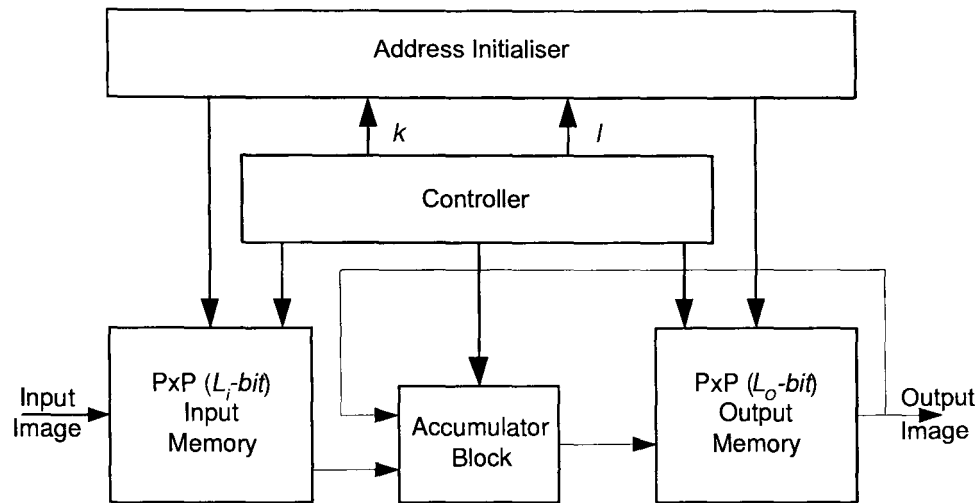


Figure 2.11: Standard pseudocode based architecture for the FRAT [21]

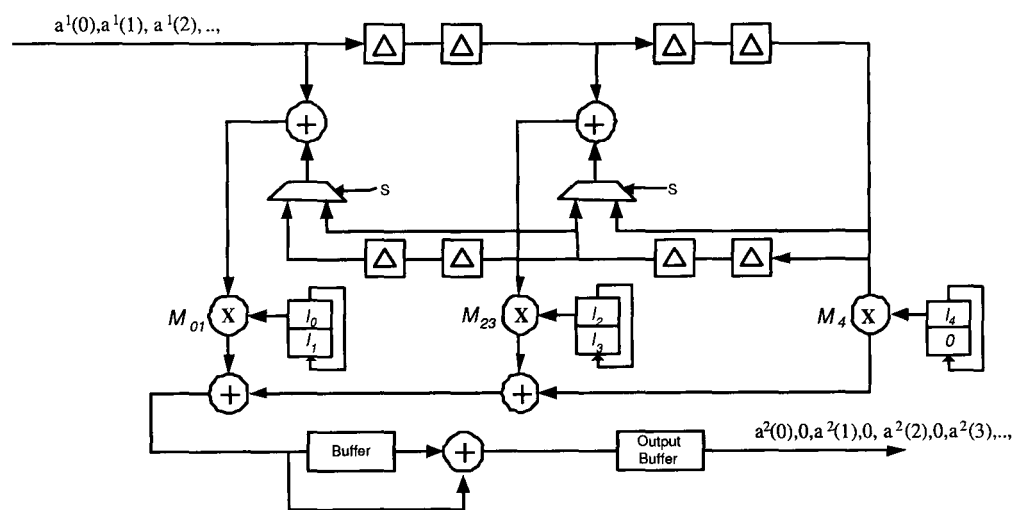


Figure 2.12: DBWT sub-block based on the à trous algorithm [21]

2.2.3 Architectures for Colour Space Conversion

Colour spaces (also called colour models or colour systems) is a method by which we can specify, create and visualise colour. There are many existing colour spaces and most of them represent each colour as a point in a three-dimensional coordinate system. Each colour space is optimised for a well-defined application area [39]. The three most popular colour models are RGB (used in computer graphics); YIQ, YUV and YCrCb (used in video systems); and CMYK (used in colour printing). All of the colour spaces can be derived from the RGB information supplied by devices such as cameras and scanners.

RGB colour space is a simple and robust colour definition used in computer systems and the Internet to help ensure that a colour is correctly mapped from one platform to another without significant loss of colour information. YCrCb is a scaled and

offset version of the YUV colour space where Y represents luminance (or brightness), U represents colour and V represents the saturation value. Conversion between these two colour representations systems is the cornerstone of integration between efficient compression and representation of images offered by the YCrCb required for efficient storage and transmission; and traditional RGB systems used in visual display units.

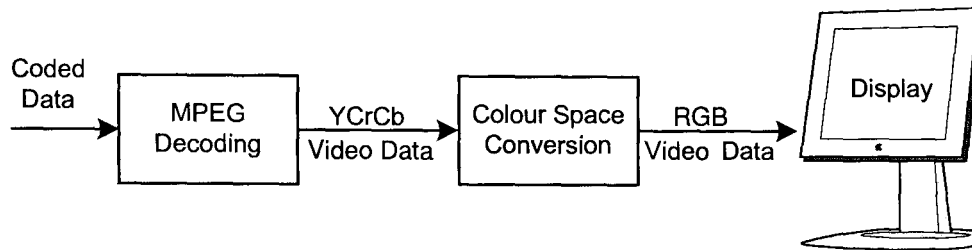


Figure 2.13: The video decoding process

Existing FPGA based systems for the implementation of CSC are documented in the following subsections.

FPGA Based Color Space Converter: YCrCb to RGB

In [40] three ways to implement YCrCb to RGB CSC which is essential in many video processing applications have been presented. All architectures have been synthesised and prototyped on the Xilinx XC2V500-5 FPGA series belonging to the Virtex-II family [36].

The first implementation, reproduced in Fig. 2.14 shows how to simply write behavioural Verilog to describe the conversion equations and then synthesise to a silicon target. This technique infers MULT_ANDs for the constant coefficient multiplier. The implementation summary for this architecture is as follows: 258 LUTs required, maximum frequency of 71 MHz.

The second implementation uses embedded RAMs available in recent Xilinx FPGAs as LUTs or ROMs, to store all possible intermediate results for the terms in the three equations. Since three of the seven total terms are identical, only five ROMs are needed. The depth of the ROM (1 Kilobit) is driven by the colour component bit width of 10 bits or studio quality video. The architecture that is developed for this implementation is presented in Fig. 2.15. This architecture operates at a maximum frequency of 103 MHz and consumed 60 LUTs.

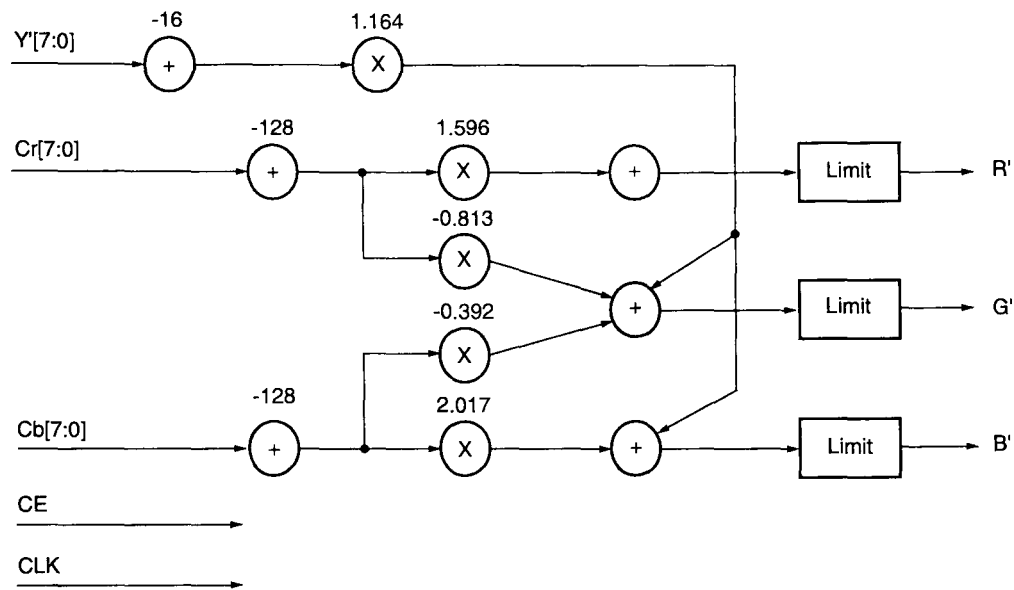


Figure 2.14: Simple arithmetic architecture for CSC [40]

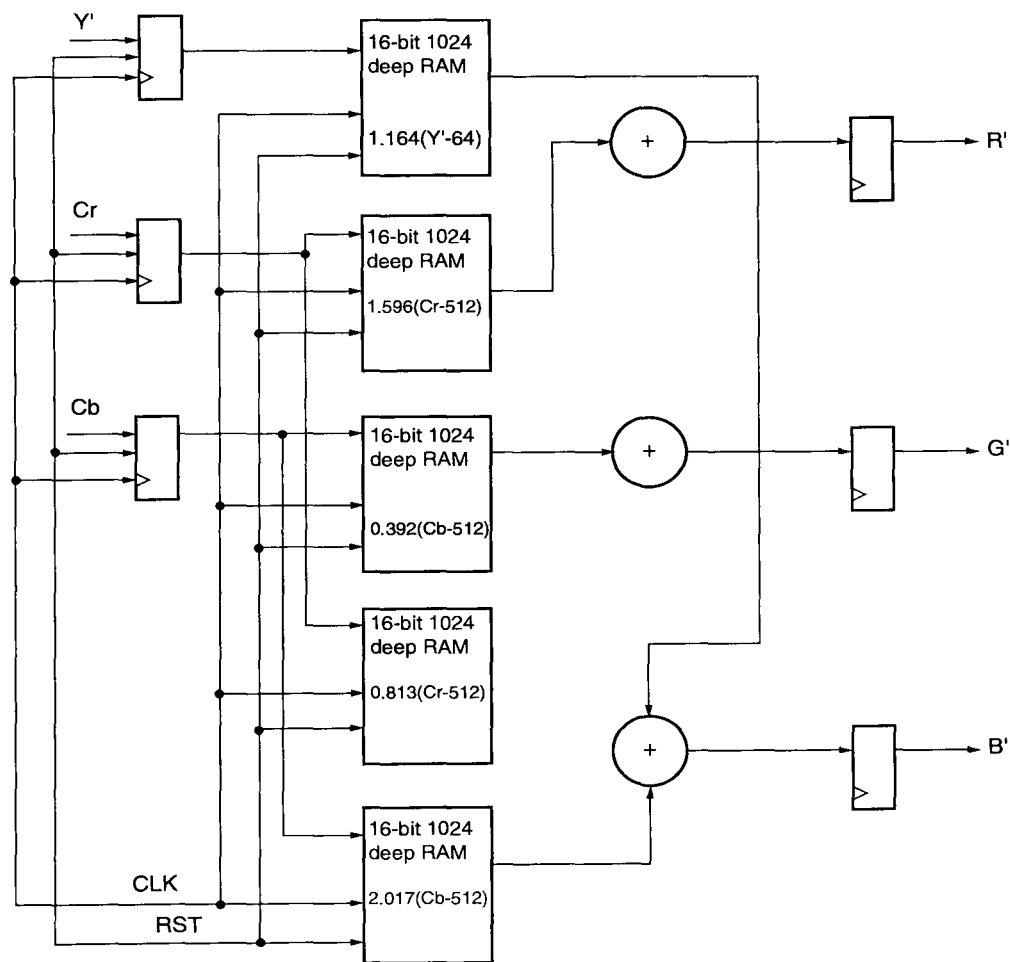


Figure 2.15: LUT based CSC architecture [40]

The third implementation makes use of the embedded multiplier in the Virtex-II series of devices to perform CSC. Again, only five multipliers are used. The architecture for this implementation is presented in Fig. 2.16. The design has a

clock performance of 185 MHz after place and route, using simple constraints. 131 LUTs are used up in the FPGA implementation of this architecture. A maximum operating frequency of 111 MHz is achieved.

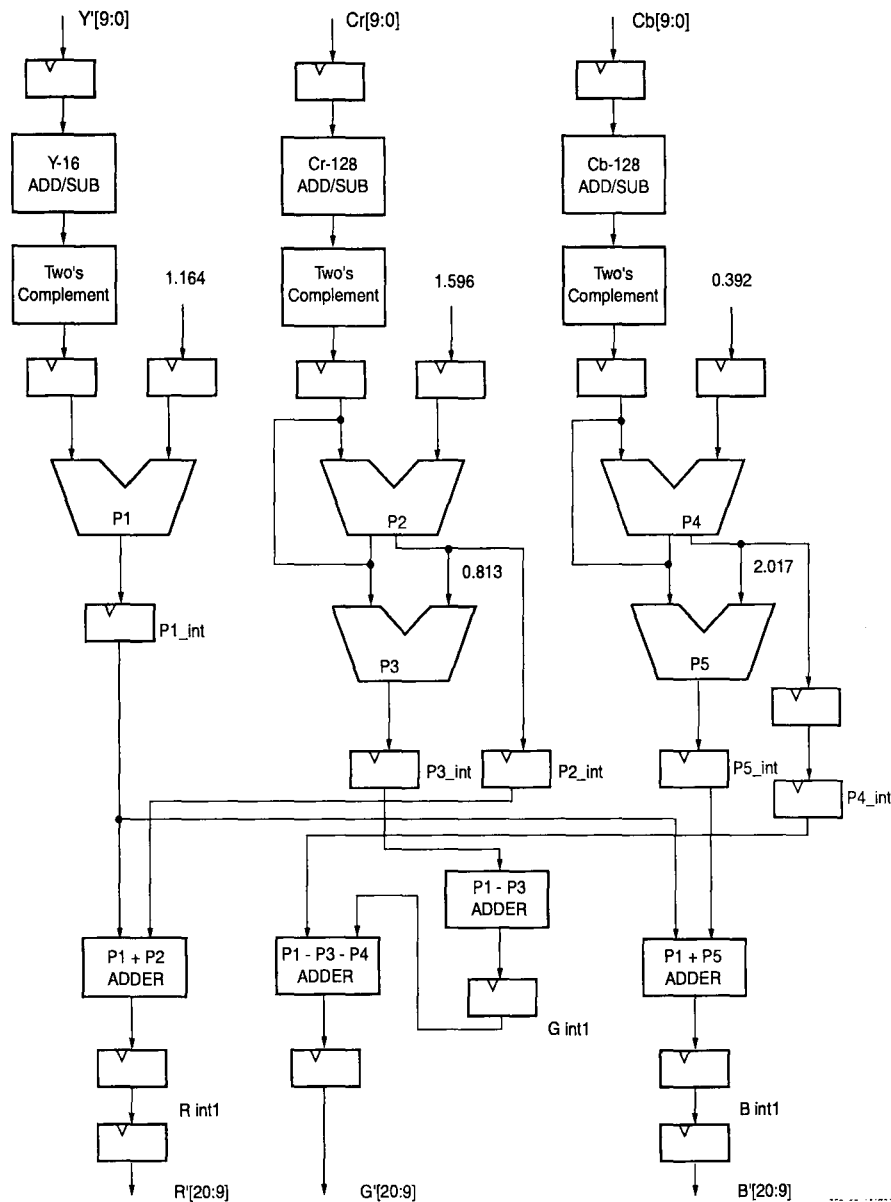


Figure 2.16: CSC architecture based on embedded multiplier [40]

FPGA based Tri-media co-processor for CSC

In [41], a case study on a transformer from YCrCb colour space to RGB colour space for MPEG decoding, carried out on FPGA augmented TriMedia processor is presented. The architecture is based on a 4 stage pipelined implementation of the standard conversion formulae and is presented pictorially in Fig. 2.17. The design is mapped on an ACEX EP1K100 FPGA device [42]. This CSC core exhibits a latency

of 10 cycles and a recovery of 2 cycles respectively, with a maximum operational frequency of 200 MHz; and occupies 57% of the device area available.

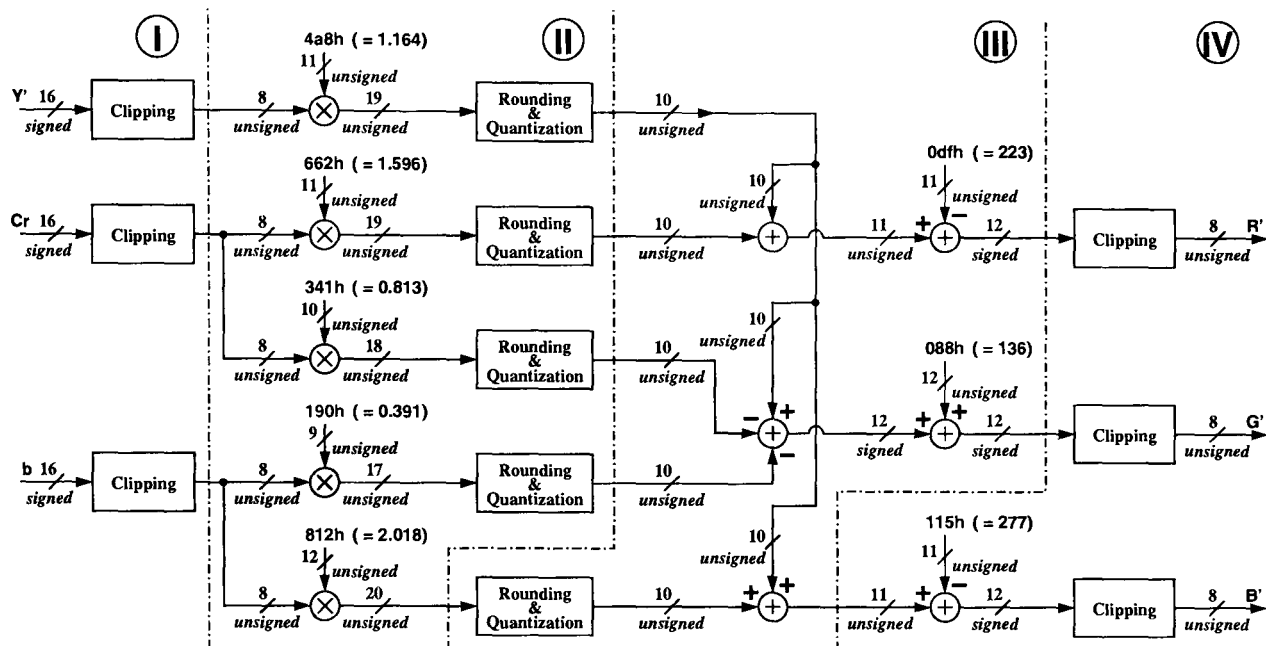


Figure 2.17: Four stage YCrCb to RGB FPGA Trimedia co-processor [41]

Commercially Available IP Cores suitable for FPGA implementation

The work presented in [43] is concerned with the implementation of CSC on the Cyclone-II using two approaches based on DA principles and multipliers using LUTs achieving 216 MHz and 175 MHz respectively. In addition, the results achieved in terms of speed and area consumed for FPGA based CSC implementations using Xilinx Virtex-E XCV50E-8 FPGA series have been presented in [44–46]. It is worth noting that in all three cases, the implementations have been carried out using 13-bits internal arithmetic. The on-chip areas occupied by these designs are 222, 222 and 204 slices; with a maximum operational frequency of 112, 105 and 90 MHz respectively. All the above mentioned cores are available in commercial IP libraries and supplied as black-boxes. Internal architectural details are not available in public domain.

2.2.4 Application Specific Hardware Implementation of Gaussian Mixture Modelling

GMM can be classified as a semi-parametric density estimation method since it defines a very general class of functional forms for the density model. In this mixture model, a probability density function is expressed as a linear combination of basis functions. The GMM algorithm is specific to a given classification problem and requires training before the system parameters can be hard-coded. Additionally, the size of training set, accuracy of internal representation, classification resolution and a number of additional design factors need to be taken into consideration while designing an GMM classifier. This makes it impossible to evoke meaningful comparisons with other GMM architectures in print. A key contribution in collaboration with HKUST is the optimisation and FPGA implementation of their existing GMM implementation using novel arithmetic techniques, algorithmic transformations and architectural optimisation techniques resulting in compact and efficient design. In the following sub-section, a pre-optimisation architecture for the GMM is presented as a study in comparison.

Gaussian Mixture Model Based Classifier

In the architecture for the GMM presented in [47], a number of design strategies are proposed in order to achieve the best possible tradeoffs between circuit complexity and real-time processing. First, a serial-parallel vector-matrix multiplier combined with an efficient pipelining technique is used. A novel exponential calculation circuit based on a Linear Piecewise Function (LPF) approximation is used to reduce hardware complexity. The precision requirement of the GMM parameters in the classifier in [47] has also been studied for various classification problems. The control unit is implemented using a finite-state machine. Information related to the GMM classifier such as the total number of Gaussian models and the number of Gaussian models for each class is stored in the two 20-bit registers within the control unit. The square calculation is performed using an array multiplier. The final classification decision is made using a Winner Take All (WTA) circuit. A block

diagram of all the functional blocks in this architecture is presented in Fig. 2.18. The latency and throughput of the system is reported to be $0.79 \mu s$ and $0.63 \mu s$, respectively, for an operating frequency of 80 MHz and for ten Gaussian models. For design validation, a prototype was implemented in 0.25μ CMOS technology and its operation was successfully tested for gas identification application. The key disadvantages of this implementation is the lack of architectural optimisation in the choice of multipliers used. Additionally, the architecture can be significantly improved by means of pipelining and effective use of parallelisation. These issues are addressed in Chapter 3, where a completely new high-speed and efficient architecture for the GMM is developed for the same application.

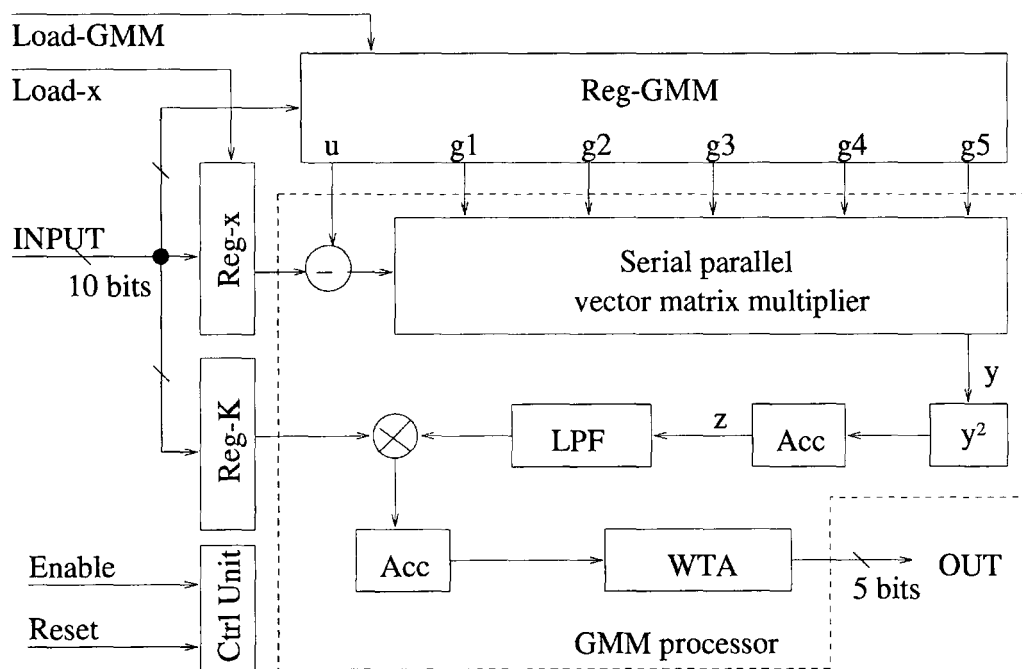


Figure 2.18: Functional units in the GMM classifier [47]

2.2.5 FPGA Implementation of Finite Digital Convolution

Calculation of finite digital convolution is frequently encountered in several DSP applications. Efficient VLSI implementation of the digital convolution for real-time DSP applications is, therefore, an important task. Our collaboration NTU has resulted in the design and efficient FPGA implementation of systolic architectures for finite digital convolution.

The memory requirement of DA-based implementation for FIR filters, however,

increases exponentially with the filter order. Attempts are, therefore, made to use Offset Binary Coding (OBC) [48] to reduce the ROM size by a factor of 2. Memory-partitioning and multiple memory-bank approach along with flexible multi-bit data-access mechanisms are suggested for FIR filtering and inner-product computation in order to reduce the memory-size of DA-based implementation [49–53].

All these structures, however, are not suitable for implementation of the FIR filters in systolic hardware since the partial products available from the partitioned memory modules are summed together by a network of output adders.

Comparable architectures available for review have been listed out in the following subsections.

Hardware-Efficient Distributed Arithmetic Architecture for High-Order Digital Filters

In [54], a memory-efficient DA based architecture (Fig. 2.19) for high-order FIR filters is presented. It is reported that the authors have used recursive iteration of the memory reduction technique significantly increases the maximum number of filter order that can be implemented on an FPGA platform. The presented DA architecture exploits symmetry property of the LUT of the original DA. LUT-less DA OBC based implementation presented in [54] is optimised by applying LUT reduction technique to the smallest 2-word LUT-based DA-OBC. Further optimisation is achieved by architectural reorganisation to enable the use of a single global adder/shifter unit which would be no longer controlled by either the input signal or initialisation trigger.

This architecture is prototyped on an Altera Stratix FPGA device [55]. However, it must be highlighted that, FPGA implementation details have been provided only for the case $B_c = 18$, filtersize $K = 16$ and decomposition level $k = 4$ (where B_c is defined in [54] as word length of the original LUT). The design occupies a total of 551 LEs and 1376 memory elements.

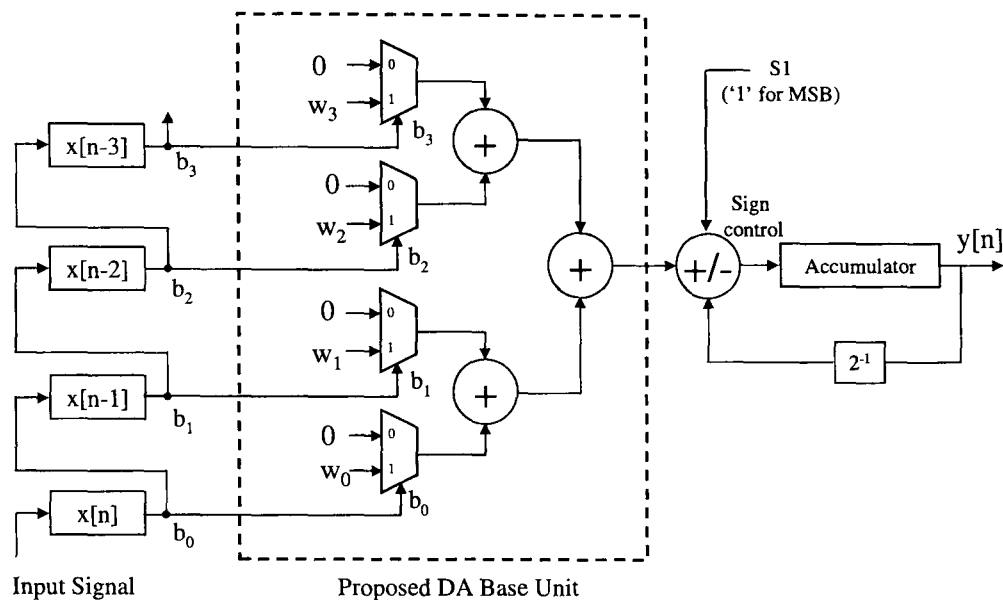


Figure 2.19: LUT-less OBC DA architecture for 4-tap FIR filter [54]

A Faster Distributed Arithmetic Architecture for FPGAs

In [56], an architecture for speeding up DA-LUT applications is presented. The key feature of this design is that the carry chain delay is reduced or eliminated from the critical path. In this design, individual bits of a word are not processed as a unit. Instead, the current iteration can start as soon as the Least Significant Bit (LSB) of the previous iteration is available, without waiting for the whole word from the previous computation. Designs in which an n -bit carry chain, where n is the word length, is broken into smaller r -bit chains, $1 \leq r < n$, are described in this work. Architectural details of the design proposed in [56] are presented graphically in Fig. 2.20.

The designs in [56] have been implemented on a Xilinx XC4028XL-3-BG256 FPGA device. The architecture requires 165 Configurable Logic Blocks (CLBs), and operates at a maximum frequency of approximately 62 MHz.

Non-DA, Systolic FIR Architectures

To utilise the advantages of systolic processing, several algorithms and architectures have been suggested for systolisation of FIR filters [57–60]. However, the multipliers in these structures require a large portion of the chip-area and consequently enforce limitation on the maximum possible number of Processing Elements (PEs) that can

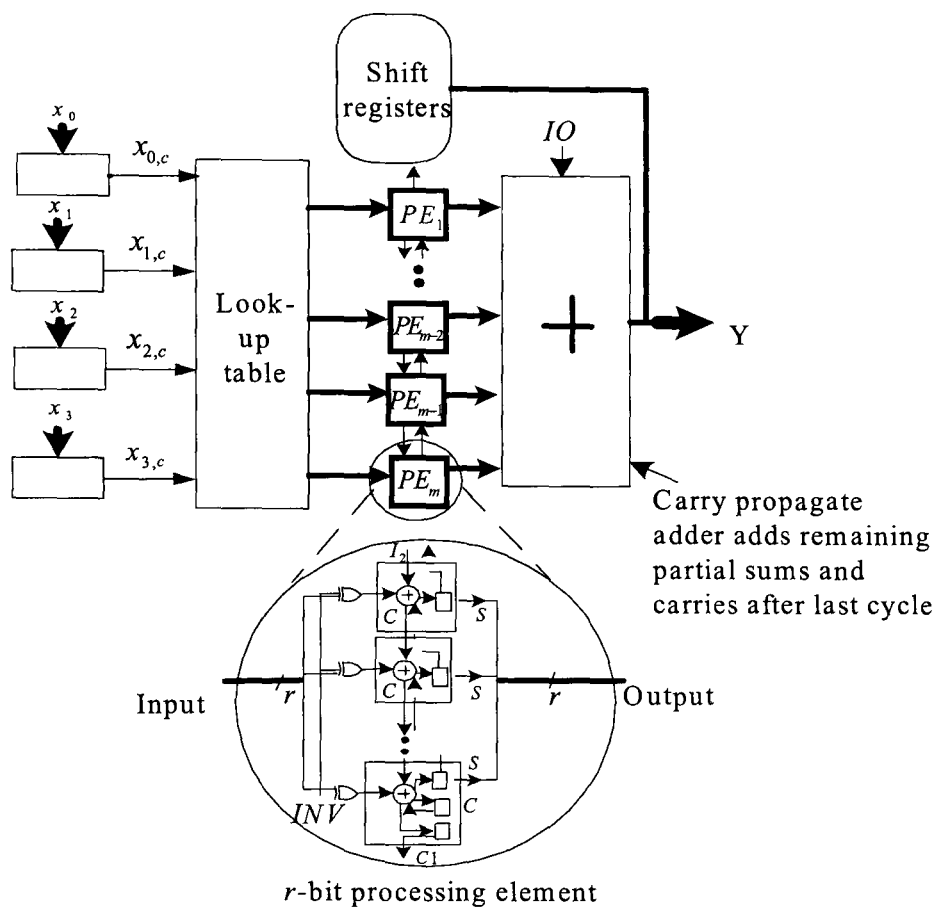


Figure 2.20: r -bit serial DA one-LUT design for a four-product MAC [56]

be accommodated and the highest order of the filter that can be realised.

2.3 Power Modelling

Power dissipation has become a key design issue in FPGA based architectures. Mobility, battery limitations, thermal constraints, reliability issues, cost of cooling subsystems and achieving time to market within these design constraints has necessitated the adoption of power aware design flows. In this section, a brief description of various existing FPGA power modelling techniques operating at different design abstraction levels have been documented.

2.3.1 Versatile Place and Route Based FPGA Power Models

In [61, 62] a detailed and flexible power model which is integrated within a modified Versatile Place and Route (VPR) Computer Aided Design (CAD) framework

(presented in Fig. 2.21 is described. This power model estimates the dynamic, short-circuit and leakage power consumed by FPGAs. The power model is aimed at island-style FPGA architectures with H-tree clock networks. The model has two modules: an activity generation module and a power estimation module. The first module employs the transition density model to determine the switching activities inside the circuit. The second module estimates the power consumption at the transistor level. The model was calibrated using HSPICE with the technology parameters from TSMC for a 1.8-V, 0.18- μ CMOS technology. Dynamic power estimation includes routing and logic block components. Total dynamic power dissipation is determined by the following expression:

$$\text{DynamicPower} = \sum_{\text{allnodes}} 0.5 \cdot C_y \cdot V_{\text{supply}} \cdot V_{\text{swing}} \cdot D(y) \cdot f_{\text{clk}} \quad (2.1)$$

where $D(y)$ and C_y are the transition density and capacitance at node y respectively. Short circuit power is assumed to be 10% of total dynamic power. HSPICE simulations for NMOS and PMOS transistors have been carried out to determine effective leakage power.

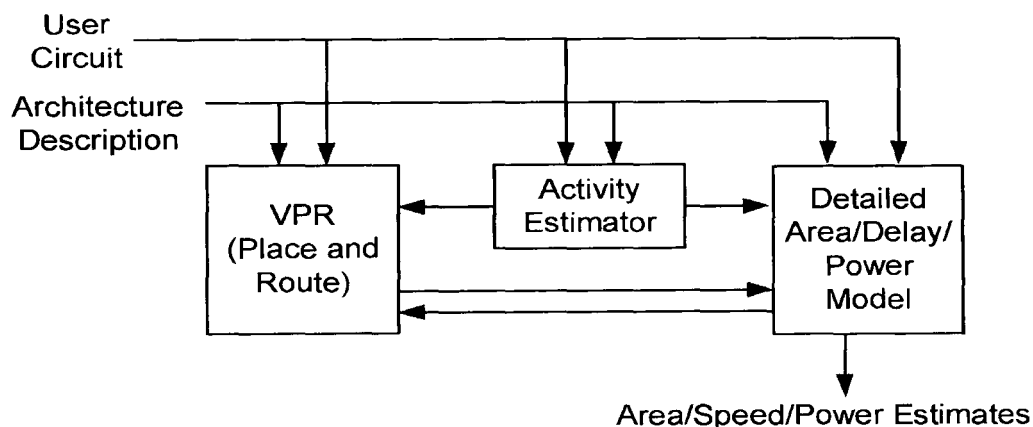


Figure 2.21: Modified VPR framework used for power modelling [61, 62]

2.3.2 Probabilistic Power Prediction for the Xilinx 4000-Series FPGA

In [63], a Java based probabilistic power prediction tool for the Xilinx 4000-series FPGA is represented. The average power dissipation due to a signal s is modelled

by $0.5 \cdot C_{d(s)} V^2 a_s f$, where $d(s)$ is the Manhattan distance spanned by s across the array of CLBs, $C_{d(s)}$ is the equivalent capacitance seen by s and V is the voltage level of the FPGA device, and f is the frequency. The tool is calibrated by partitioning a set of S signals into $2N + 1$ subsets based on the length associated with each signal. A mathematical representation is provided in Eq. 2.2:

$$P_{\text{avg}} = \frac{1}{2} V^2 f \left(C_0 \sum_{s \in S_0} a_s + C_1 \sum_{s \in S_1} a_s + \dots + C_{2N} \sum_{s \in S_{2N}} a_s \right) \quad (2.2)$$

In this work, a total of 70 power measurements have been performed using 5 different configuration files and 14 different data sets. It is reported that the the maximum error in predicted versus measured power is typically less than about 5%. Although the modelling methodology is high level, notable drawbacks are that it is not platform independent and not scalable in nature.

2.3.3 Cycle Accurate Energy Measurement and Characterisation of FPGAs

In [64, 65], a high level tool for operation-based energy characterisation of FPGAs is presented. The proposed cycle-accurate energy measurement and consequent low-level characterisation with several examples, followed by the introduction of a macro energy state machine have also been presented in [64, 65]. Data acquisition for calibration of the proposed tool is pictorially presented in Fig. 2.22.

This is followed by high level state machine based calibration for the design. Design strategies including pipelining, loop unrolling are taken into consideration for applications such as FIR filters for concept validation. This modelling methodology has approximately 95% accuracy. However, the accuracy level achieved is high, as it is an RTL level approach. Additionally, it is neither scalable nor platform independent.

2.3.4 High-Level Power Modelling of CPLDs and FPGAs

A high-level power modelling technique to estimate the power consumption of reconfigurable devices such as Complex Programmable Logic Devices (CPLDs) and FPGAs is presented in [66]. The development of models is based on input and output

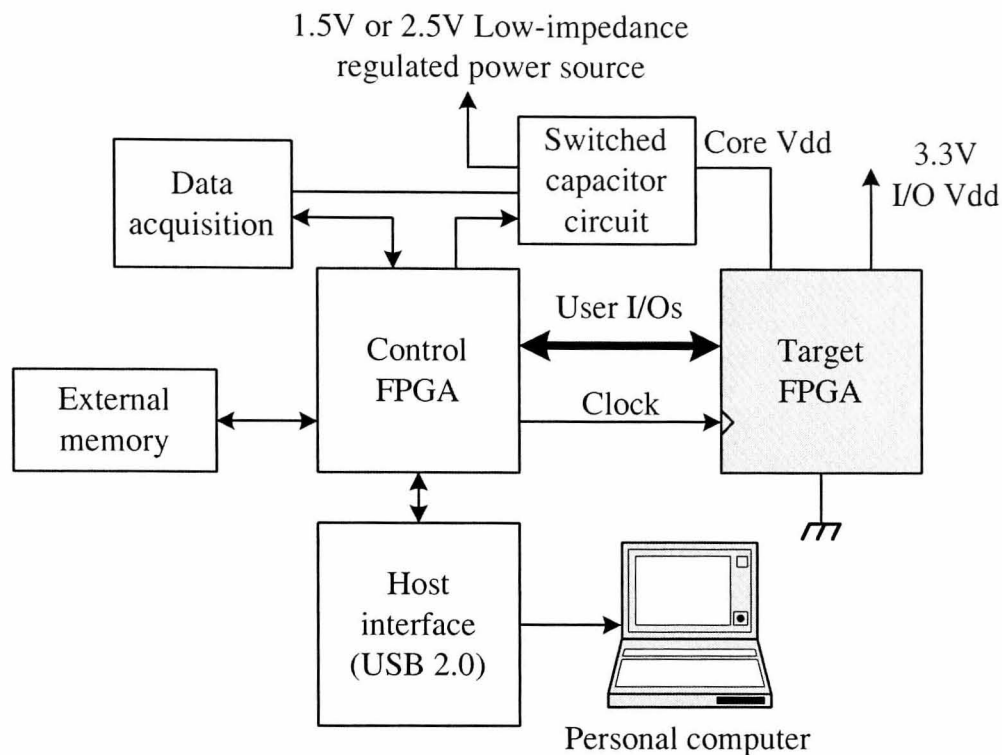


Figure 2.22: Real-time cycle accurate measurement system for SRAM FPGAs [64, 65]

signal statistics to estimate the internal power consumption of FPGAs. Input signal modelling is based on the determination of optimal signal partitioning, probability estimation, transition density, signal space correlation followed by output signal transition density. Models for differently-configured circuits are based on the same power macromodel template. To achieve tradeoff between accuracy and efficiency, an adaptive regression method is used to tackle the problem of biased training sequences. The overall process flow is presented in Fig. 2.23. This high level modelling methodology has a high accuracy of 97% and is platform independent. However, key disadvantages are that the modelling methodology requires training and is not scalable.

2.3.5 Macromodels for High Level Area and Power Estimation on FPGAs

A tool for estimation of area and power within the context of a high-level automated design space exploration pass which determines the effects of various compiler optimisations on the area and power of the synthesised hardware is presented in [67].

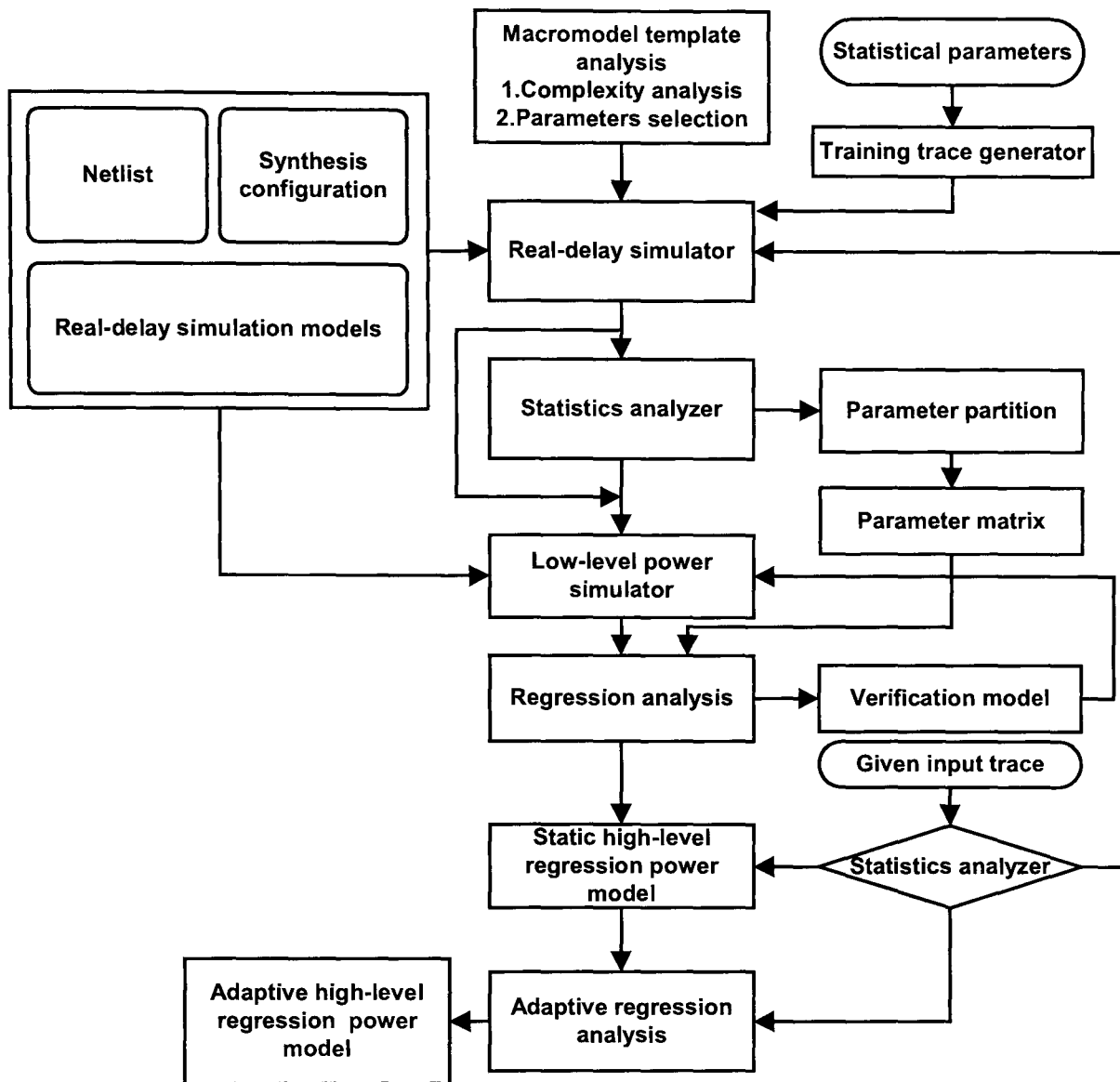


Figure 2.23: High level power macromodelling for reconfigurable hardware [66]

The presented area estimation technique is based on high-level compile time estimation of the areas of the Control-Data Flow Graph (CDFG) nodes. Each CDFG node represents an operator and is parameterised with the bit-widths of the inputs (such as N -bit adders and multipliers) and characterising the results obtained from post layout to take into account route-through. The proposed power macromodelling approach in [67] is based on average input signal probability P_{in} , the average input transition density D_{in} and the average input spatial correlation S_{in} as the candidates of input metrics. Input bit width N is also taken into account. The equation-based macro-model used to estimate the average dissipation power can be described as:

$$P_{out} = f(P_{in}, D_{in}, S_{in}) \quad (2.3)$$

where f is a mapping procedure to be determined during the characterisation, using sample input vector streams as shown in Fig 2.24. This platform independent modelling approach has varying levels of accuracy (depends on the core that is modelled) and is not scalable.

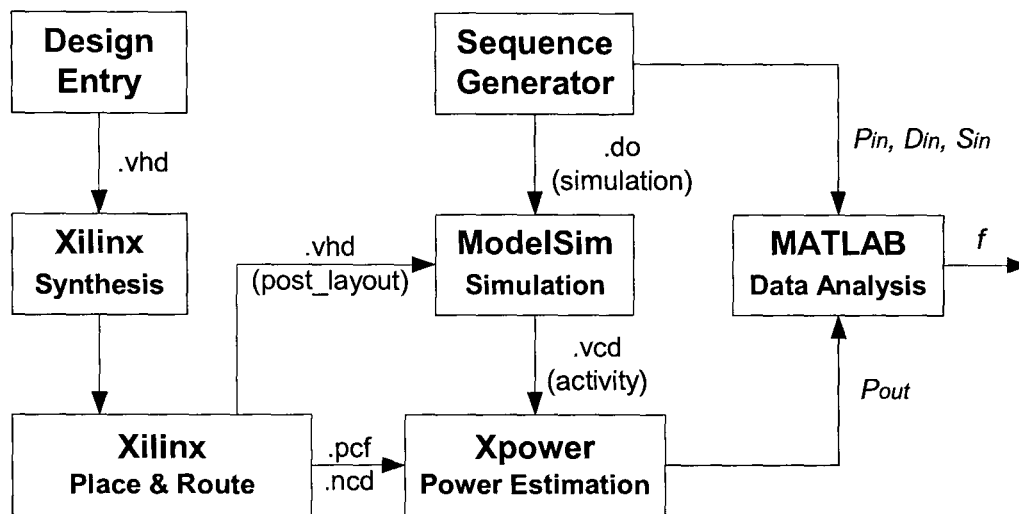


Figure 2.24: Macro-model characterisation procedure [67]

2.3.6 Methodology for High Level Estimation of FPGA Power Consumption

A circuit-level simulations to characterise a simple, coarse-grained FPGA architectural model is presented in [68]. The dynamic power estimation technique presented in this work involves two processes. First, each resource is characterised to find its effective capacitance. Characterisation is subdivided into global wire modelling and input dependency. Next, power of a given design is estimated by finding the utilisation of each resource and determining its switching activity. Power estimation and accuracy evaluation is performed for a set of 14 designs against silicon measurement. The measurements were taken on an internally developed test board hosting an XC3S1000 FPGA, a mid-sized device with 1,920 CLBs. The overall power estimation methodology in [68] is presented graphically in Fig. 2.25. Although this model is scalable, it has a low accuracy of around 82% only.

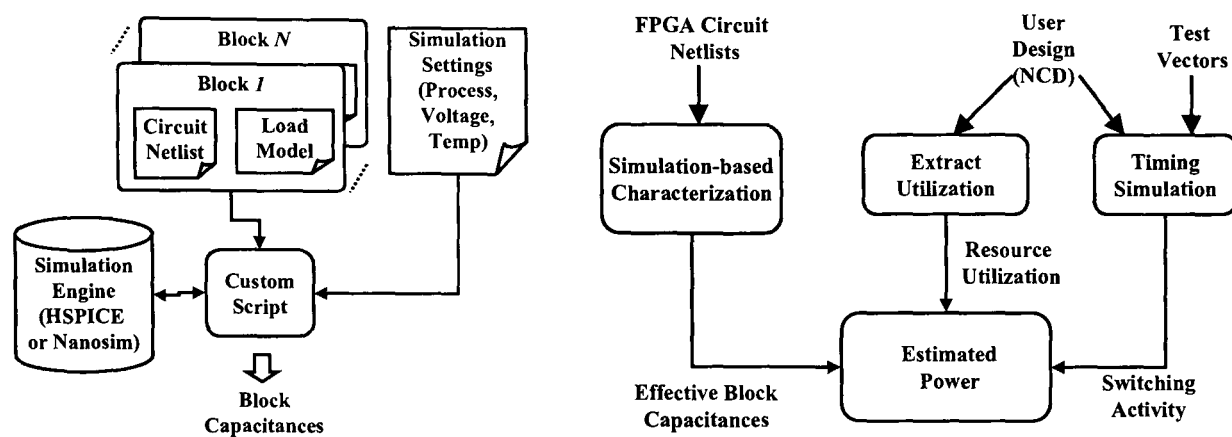


Figure 2.25: High level FPGA power estimation methodology [68]

2.3.7 Post Synthesis Level Power Modelling of FPGAs

In [69] a methodology and tool suite capable of modelling the power consumption of an FPGA design at the post synthesis, or Electronic Design Interchange Format (EDIF), level. It is suggested that modelling at this level has the following advantages: Firstly, early power feedback in the design flow. Secondly power results are displayed at a high level, closer to the logical design entry point. Finally, the elimination of bulky, low-level timing accurate simulation and stimulus files. These three aspects allow a designer to quickly and easily generate power estimates, relate the results back to their original logical level design entry and explore design trade-off scenarios.

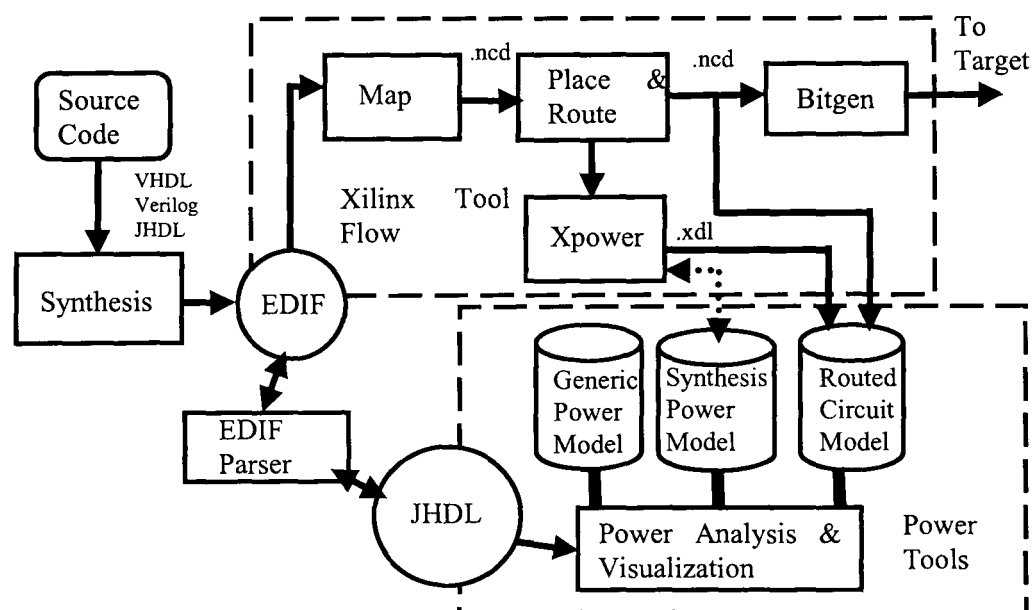


Figure 2.26: High level FPGA power estimation methodology [69]

The power modelling approach in [69] consists of: 1) developing a tool infrastructure to support synthesis level simulation and circuit queries and 2) developing a synthesis-level power model. Using Just-Another Hardware Description Language (JHDL) [70] power infrastructure, several designs on a Virtex-II 6000 FPGA varying in functionality and utilisation have been profiled and relationship between wire capacitance and fanout, wire length, number of switch boxes etc, is explored. The tool-chain process used is pictorially reproduced in Fig. 2.26. This modelling approach has a high accuracy of 97%, but is not scalable and not platform independent.

2.3.8 Power Estimation and Power Measurement of Xilinx Virtex FPGAs: Trade-offs and Limitations

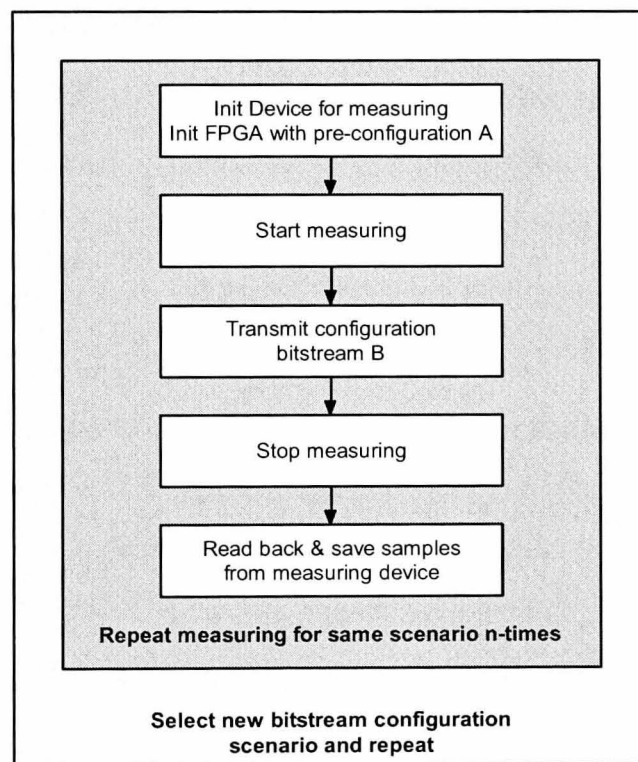


Figure 2.27: Measurement cycle [71]

In [71], power consumption trade-offs between the measured runtime consumption of a mapped application and the measured reconfiguration-time consumption of different dynamically (partially and completely) reconfigured applications is explored. The possibilities and limitations of today's available power estimation tools are discussed and compared to the exact measurements have also been studied in [71]. In this work, a Spyder Virtex Board equipped with a Xilinx XCV2000E BG560-6

FPGA [26] is used as the experimental platform. The measurement system consists of a PC with the control software, a Tektronix oscilloscope (Type TDS 220) connected to the PC's RS232 interface. The measurement cycle used is presented pictorially in Fig. 2.27. The modelling approach presented in this work is highly platform specific and is completely based on physical measurements. Theoretical extrapolation of model information for predicting power is not supported. Accuracy details have not been clearly described.

2.3.9 Power Estimation for Cycle-Accurate Functional Descriptions of Hardware

A methodology for Cycle-Accurate Functional Descriptions (CAFD) power estimation that combines the accuracy achieved by power estimation at the structural RTL with the efficiency of cycle-accurate functional simulation by viewing a CAFD as an abstraction of a specific, known RTL implementation that is synthesised from it is presented in [72]. This modelling methodology is described briefly as follows. For a given CAFD, corresponding simulation testbench and a power model library (generated once for each fabrication technology, using well-known characterisation techniques) for RTL components preprocessed is first carried out in order to enable easier back-annotation of RTL information. Virtual component instantiation and idle cycle analysis is then performed resulting in an RTL-aware CAFD which is co-simulated with the power model library to determine average power. An adaptive state-based sampling technique is used to optimise the allocation of sampling probabilities to different control states for improved representation of states with a higher time-variance of power. A diagrammatic representation of the modelling methodology in [72] is presented in Fig. 2.28. It is claimed that the accuracy of this architectural level modelling technique is high. Although it is scalable and parameterisable, the key disadvantage is the unsuitability of this modelling approach for IP core macromodelling.

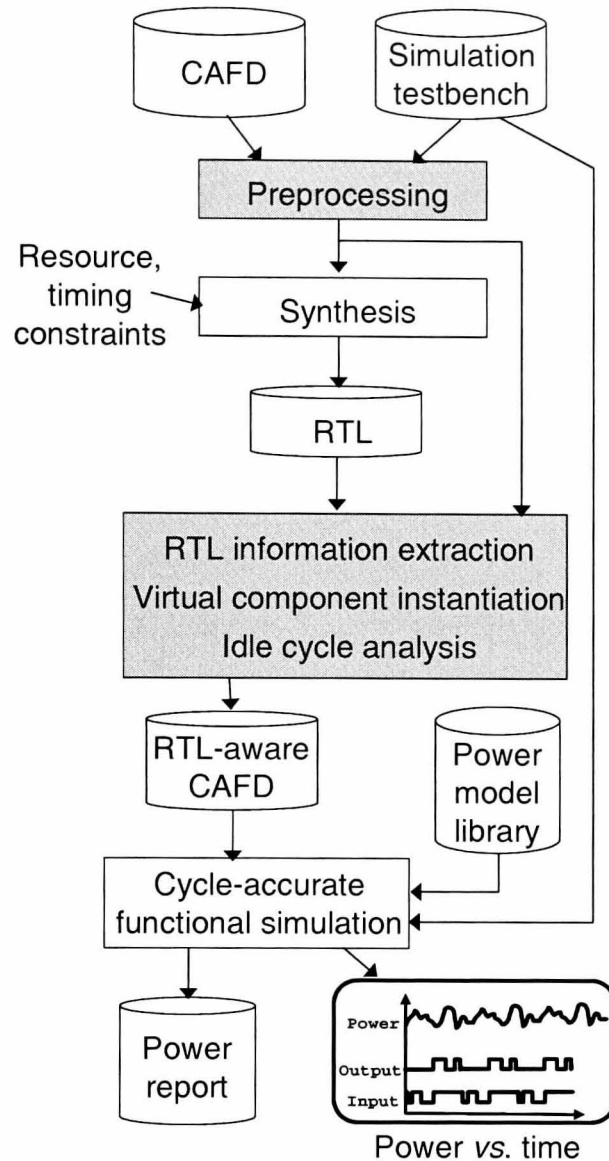


Figure 2.28: Overview of the CAFD power estimation methodology [72]

2.3.10 Dynamic Power Estimation Technique for FPGAs

Empirical early prediction models of net activity and interconnect capacitance in FPGA based designs suitable for use in power-aware layout synthesis and early power estimation/planning have been discussed in [73]. Delay based switching activity analysis and prelayout activity prediction has also been performed. The prediction methodology proposed in [73] consists of the following steps:

- Identifying suitable benchmark circuits to be implemented and modelled;
- Arbitrary division of circuits into characterisation and test sets respectively;
- Parameter extraction and analysis of characterisation circuits; and

- Statistical analysis of prediction and target parameters using multivariable regression analysis followed by verification.

The process flow is pictorially presented in Fig. 2.29.

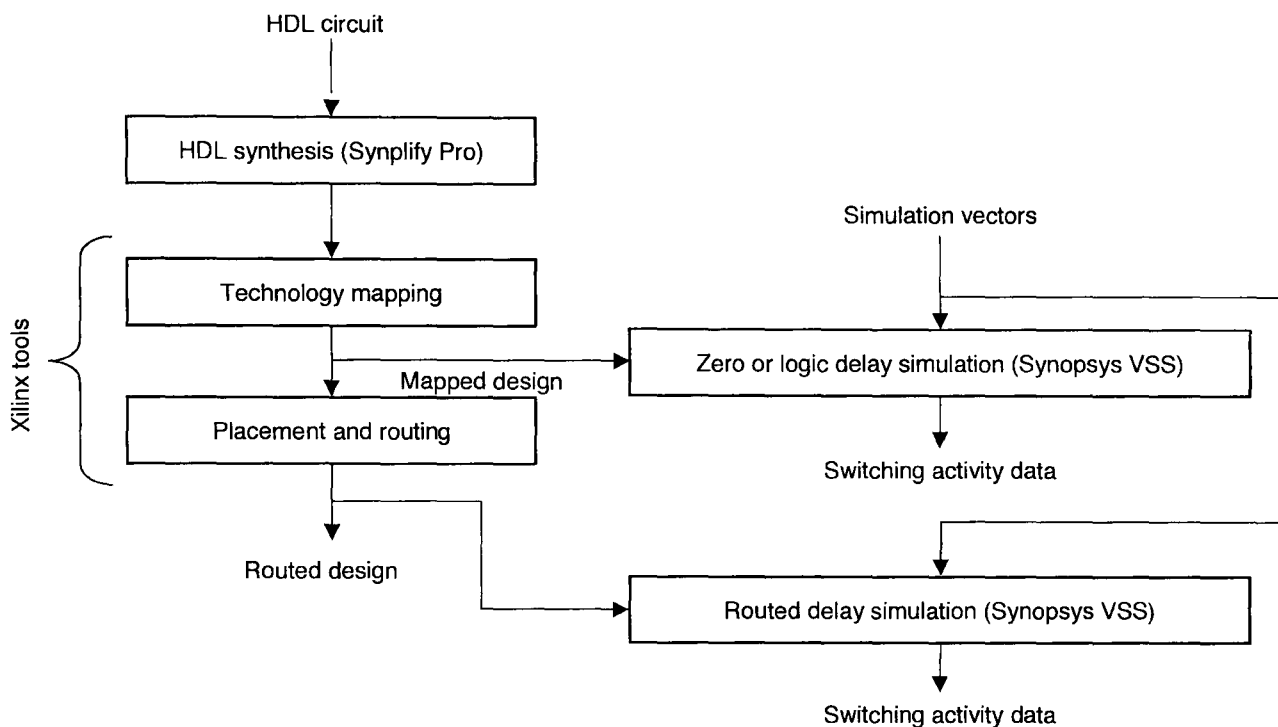


Figure 2.29: CAD flow for activity analysis [73]

The parameters used in the prediction model include fan out, half perimeter of nets, x-y span, area occupied, number of load pins on nets and average estimated routing congestion in net bounding box. The proposed prediction model is validated on the Xilinx Virtex-II PRO FPGA family.

2.3.11 Power Modelling and Characteristics of Field Programmable Gate Arrays

A mixed-level model called *fpgaEVA-LP2* for both dynamic and leakage power that combines switch-level models for interconnects and macromodels for LUTs is presented in [74]. The key objective is to examine the power impact of FPGA circuits, architectures and CAD algorithms and to study the power characteristics of existing FPGA architectures. The area model in *fpgaEVA-LP2* is based on the technology-scalable area model implemented in VPR. The delay model in *fpgaEVA-LP2* uses delay values obtained by SPICE simulations in the predictive 100 nm CMOS tech-

nology and various circuit paths inside a logic block are simulated and path delays are precharacterised. The dynamic power model is built using switch-level models for interconnects (as a function of signal transition), macromodels for LUTs for randomly generated input vectors and finally using transition density for glitch analysis. Static or leakage power as a result of various device level mechanisms is determined using SPICE simulations for LUT sizes ranging from three to seven and buffers of various sizes in global/local interconnects. The overall process flow is described pictorially in Fig. 2.30. This mixed level modelling approach has a medium accuracy level of 92% and is scalable. However, it is not platform independent.

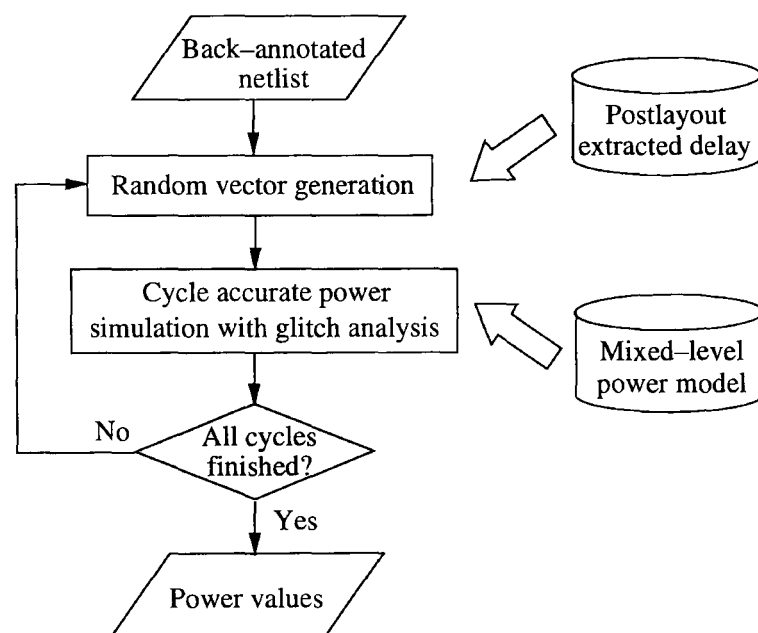


Figure 2.30: Overall power calculation [74]

2.3.12 Brief Review of selected ASIC Power Modelling Techniques

In this section, an overview of power estimation and modelling techniques for ASICs that are comparable to existing FPGA power modelling solutions is presented.

High/Algorithmic Level Power Estimation

High level power models for ASIC based on principles of information-theoretics have been explored in [75–78] for quick power estimation. Estimation and modelling in

these works is performed by measurement of the absolute entropy or difference in the entropies of input and output signals. Models that encapsulate circuit complexity as a key parameter in precharacterised high-level design libraries or limited sets of boolean functions are presented in [79–81].

Power Estimation and Modelling at RTL for ASIC

The dominant type of RTL power modelling is regression based approach and is also known as power macromodelling. The key process is pre-characterisation of individual components or blocks in a design from a library of IP cores under statistically modulated input conditions. This is usually followed by curve fitting using some measure of error estimation and reduction to identify accurate models. Simulation, probabilistic power estimation and statistical sampling techniques for test vector compaction are some of the techniques that are employed for low level power estimation and modelling are discussed in [82–90]. Stochastic and adaptive models for input data compaction for power estimation in finite state machines are presented in [91, 92]. A simple constant type power macro-model, known as the power factor approximation technique based on weighted estimation of module average power consumption as a function of input transitions is presented in [93]. Activity estimation based stochastic power modelling based on temporal data correlation is presented in [94]. In [95], a multidimensional, multivariate power macromodelling technique that trades additional computational complexity for higher accuracy is discussed. Parametric and cycle accurate sampling based power macromodelling approaches are discussed in [96] and [97] respectively.

Gate Level ASIC Modelling

At the bottom of the abstraction pyramid lies the gate and circuit level that allow to perform both static and dynamic modelling [75–77, 98–101] based on statistical information about the inputs applied to the circuit, such as toggling information, input correlation etc.

2.4 Architectural Optimisations for Performance Power Tradeoffs

Dynamic power consumption of designs implemented on FPGA can benefit from exploiting the massive parallelism capabilities of commercial FPGAs. In cases where the hardware resources available on the FPGA are not utilised completely, there is scope for trading off additional area for improved power and energy metrics with no penalty in terms of performance. At the architectural system level, this is possible by applying techniques such as parallelism or pipelining to the entire core or suitable subsections of the core implemented on FPGA. These techniques can be effectively coupled with proportionate reduction of clock frequency in order to maintain the same throughput. Additionally, quadratic improvement in power-delay can be achieved by reducing the supply voltage to take advantage of the reduced operational frequency.

Existing work in the domain of voltage scaling is predominantly ASIC centric [102–104]. The interesting concept of trading delay for V_{dd} and exploring parallelism and pipelining as strategies to maintain the required throughput at lower frequencies was first explored in [102]. In recent times, there is growing interest in applying voltage scaling for FPGA based designs as well.

A brief review of architectural optimisations and existing voltage scaling strategies applicable to FPGAs is presented in this section.

Power-Sensitive Design Techniques on FPGA Devices

In [105], a general overview of power saving design techniques, practices and rules covering various aspects of the design cycle at the system, architectural, RTL and low levels for both static and dynamic power have been summarised. The topics covered include switching activity reduction through clock gating, pipelining and re-timing for glitch reduction, effects of pipelining on design structure and power consumption, delay balancing and logic depth reduction, counter and state machine encoding, arithmetic optimisation techniques and design partitioning.

Low Power Digital Design in FPGAs: A study of Pipeline Architectures implemented in a FPGA using a low supply voltage to reduce power consumption

A circuit-architectural technique to reduce power consumption in FPGAs is presented in [106]. The authors analyse the influence of the supply voltage and the propagation delay using the Altera FLEX10K100-PGA-504 FPGA as a test platform. The Voltage/Frequency ratio is measured using a ring oscillator in order to obtain the maximum propagation delay with the lowest possible supply voltage; and show that power consumption decreases faster than frequency. Architectural optimisation is performed by increasing the pipeline granularity to increase the performance of the circuit and simultaneously reducing the power supply to save power consumption and maintaining the same level of performance. It is claimed that a reduction maximum global power consumption of of the FPGA by up to 75% is achieved.

The Impact of Pipelining on Energy per Operation in Field-Programmable Gate Arrays

An experimental investigation into the quantitative impact of pipelining on energy per operation for two representative FPGA devices: a 0.13μ CMOS high density/high speed FPGA (Altera Stratix EP1S40) and a 0.18μ CMOS low-cost FPGA (Xilinx XC2S200) both measurements and execution of vendor-supplied tools for power estimation is presented in [107]. The best possible gains that can be obtained by pipelining/retiming by considering circuits with registers after every logic element have been analysed. The interaction between pipelining (a system-level design optimisation) and clustering (a lower-level design optimisation) and the correlation between the degree of pipelining and the effectiveness of the lower-level CAD algorithms in reducing energy has also been studied in this work. Circuit analysis is performed using two test bench circuits: COordinate Rotation DIGital Computer (CORDIC) and 64-bit integer multiplier, since these two circuits have the largest number of variants. The authors show that the “maximally pipelined” variant of each benchmark circuit dissipates the least energy on both FPGA platforms. The

results obtained indicate that combining optimisation both at the system level as well as during low-level synthesis and physical design results in maximum power reduction.

Low-Power CMOS Digital Design

Seminal work on the impact of architecture-driven voltage scaling to achieve lower power-delay products (energy per computation) is presented in [102]. It is suggested that one way to maintain throughput while reducing the supply voltage is through utilisation of a parallel architecture (as shown in Fig. 2.31). It is pointed out that careful optimisation must be performed to minimise this overhead (for example, partitioning techniques for minimal overhead). It is shown that pipelining can also be used for power reduction by a factor of approximately 2.5, providing approximately the same power reduction as the parallel case with the advantage of lower area overhead. The author indicates that additionally, pipelining also has the effect of reducing logic depth and hence power contributed due to hazards and critical races. A combination of both pipelining and parallelism has also been analysed for achieving maximum power reduction.

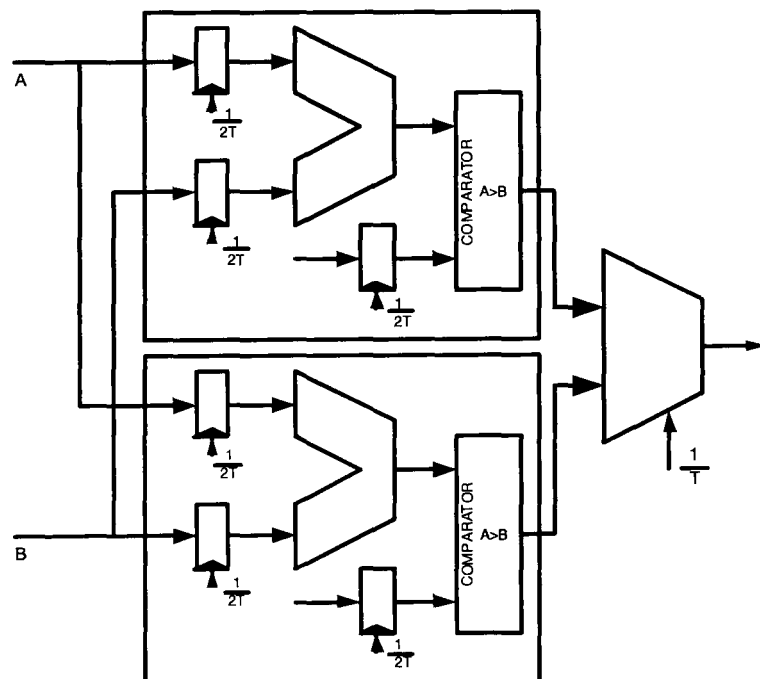


Figure 2.31: Parallel implementation of a simple datapath [102]

Voltage Scaling for the Virtex-II FPGA

The concept of voltage scaling and tolerance limits of commercially available FPGAs under varying operating conditions was explored in [108]. The authors explore design tradeoffs between throughput, power consumption and area using the Dynamic Voltage Scaling (DVS) technique on a few benchmark circuits. While the authors take cognisance of the fact that DVS can reduce both dynamic and leakage current, but at the expense of increasing circuit delay, measures for combining DVS with performance improvements through architectural strategies have not been analysed. Xilinx Virtex-II XC2V1000 FPGA [36] is chosen as the platform for which the testbench cores have been prototyped.

FPGA Power Reduction

In [109], the authors focus on power reduction in existing commercial FPGAs used in battery powered applications. It is clearly shown that voltage scaling can effectively provide a 35% and 25% reduction in active and standby power respectively of a Spartan FPGA.

Dual- V_{dd} FPGAs

Research has also been carried out in the design of dual/multiple/scaled V_{dd} FPGA fabrics [110,111]. However, most of these proposals have not been exploited on any commercially available platforms.

2.5 Shortcomings and Disadvantages of Existing Work

As can be seen from the preceding sections, there still remains plenty of scope for further research in exploiting reconfigurable computing for the various applications and algorithms that have been addressed. The major limitations of the existing work can be identified as follows:

Efficient IP core design

- In general, most of the IP cores that have been referred to are application specific. Modifications have to be made on the proposed cores to adapt them for a specific application, which require considerable FPGA knowledge for the application developers in order to re-implement the adapted cores;
- The design principles used in existing work is not strongly aligned current industry trend towards modularisation, top-down design approach and IP core reuse that are essential for the design of large and complex systems on FPGAs; and
- Implementation techniques are generally core specific and ad-hoc. Design optimisation has not been considered as a holistic challenge balancing the demands for power, performance, area etc.

Power awareness and modelling

- Power awareness has not been a major consideration in existing implementations of FPGA based design. Power is the most important challenge for FPGA based designs in the coming decade;
- While numerous power modelling tools are available for ASICs (VLSI, in general), only limited solutions are available for FPGAs; and
- Existing power modelling tools for FPGAs may have attributes such as high accuracy, low/mixed/high level, platform independence, scalability, etc. in varying measures; but generally not all together.

Optimisation through performance enhanced voltage scaling

- Most of the existing work is theoretical and have not been exploited on any commercially available platforms; and

- Measures for combining supply voltage scaling with performance improvements through architectural strategies have not been previously analysed.

Based on the limitations of existing work and the main objectives presented in Chapter 1, the work presented in this thesis can be summarised as follows:

- All the cores that have been developed in this work are parameterisable, scalable and present the end user with a number of configuration options;
- To design and implement scalable, parameterisable, efficient and novel IP cores for a range of 1-D and 2-D transforms and matrix operation; suitable for use in both general and special purpose signal, image and video processing problems and applications;
- To apply optimisation strategies at various abstraction levels and to analyse the effectiveness of these techniques for performance enhancement and power reduction;
- To investigate the best performance trade-offs such as area/speed for the FPGA implementations of these cores;
- To perform an exploratory study into the suitability of supply voltage scaling as a technique for controlling power dissipation in future generation FPGAs; and
- To develop a novel and accurate high level power modelling methodology suited for optimisation and power aware deployment of FPGA based IP cores.

2.6 Conclusions

This chapter summarises the state-of-art implementations and systems for a selected range of transform based and general architectures for image and signal processing applications on different reconfigurable platforms using various design methodologies and implementation approaches. Existing power modelling techniques at various

levels for FPGA based designs have also been reviewed. A brief introduction about trends in voltage scaling for FPGAs has been presented.

In addition, limitations of the existing work were stated. It is the aim of the research work presented in this thesis to address the limitations presented in the previous section through efficient implementations, power aware design, novel modelling methodology and a definitive study on the best approach towards future research voltage scaling for reducing power.

Chapter 3

IP Core Optimisations at Algorithmic and Behavioural Level

3.1 Introduction

Algorithmic and behavioural transformations are high level optimisations that can be performed to yield power and energy efficient designs. These transformations can help in reducing the area occupied, latency and clock count required and switching characteristics at the lowest level.

Algorithmic or behavioural transformations are changes to the computational structure in a manner such that the input/output behaviour is preserved. Alternative arithmetic operations (such as different numbering systems, distributed arithmetic, etc), fast algorithms, redundancy reduction, optimisation of computational sub-blocks, modifications to their interconnecting structures resulting in variations in dataflow, etc are some of the techniques exploited to yield behavioural transformations. This can be done through subexpression elimination, manifest expression elimination and distributivity [112]. The final goal of transformations is to reduce the number of operations involved in a certain computational process.

The relative placement of this abstraction level is indicated in the power triangle shown in Fig. 1.7.

System level choices have a number of cascading effects in overall design considerations at lower levels and thus on performance and power metrics that are ultimately

achieved. The use of transformations makes it possible to explore a number of alternative architectures and to choose those which result in the lowest power.

The development and subsequent performance of new and more sophisticated algorithms rely heavily on the architectures of integrated circuits [57]. In this chapter, a number of IP cores have been optimised using transformations such as fast algorithms, sparse matrix factorisation, distributed arithmetic principles and offset binary coding to yield high performance area efficient architectures. These architectures have been designed using a power aware design flow and complete design space exploration with power modelling has also been performed. Power modelling details are presented in Chapter 6.

The rest of this chapter is organised as follows. An area and power efficient architecture for Inner Product (InP) computation is presented in Section 3.2. A novel sparse OBC based DA architecture for matrix transforms is discussed in Section 3.3. An efficient architecture and FPGA implementation of FHT is presented in Section 3.4. In Section 3.5, the design and development of a GMM based classifier is discussed. Concluding remarks are presented in Section 3.6.

3.2 An Area Efficient Low Power Inner Product Computation For Discrete Orthogonal Transforms

DOTs, which are used frequently in many applications including image and speech processing have evolved quite rapidly over the last three decades. Typically, these applications require enormous computing power. However, a close examination of the algorithms used in such real world algorithms (e.g. the DCT, the DFT and SVD), reveal that many of the fundamental actions involve InP computation as the fundamental block [17, 113, 114].

For System on Chip (SoC) and ASIC implementations of DOTs, DA has been regarded as a very efficient way of calculating the InP. DA is used to design bit level architectures for vector-vector multiplications that covers many of the important

DSP filtering and frequency transforming functions [115]. Algorithmic and arithmetic details of DA principles are discussed in detail in Appendix C. The suitability of DA for FPGA based implementations is analysed. Techniques for reducing the hardware requirements of DA such as OBC and ROM decomposition [57] are also discussed in Appendix C.

In this section, a novel modified DA algorithm and its architecture is presented. It has also been shown that significant reduction in power consumption can be achieved under certain conditions achieved in comparison with conventional approach for implementing DA.

The novel architecture based on the proposed algorithm is prototyped on the Celoxica RC1000 PCI development board [116] [Appendix B].

3.2.1 Mathematical Basis

Consider an inner product of two vectors A and B of length N .

$$Y = \sum_{k=1}^N A_k B_k \quad (3.1)$$

where B_k is represented in 2's complement binary notation and is defined as $B_k : b_{k0}; b_{k1} \dots; b_{k(W-1)}$ such that B_k has a wordlength W and b_{k0} is the sign bit.

$$B_k = -b_{k0} + \sum_{n=1}^{W-1} b_{kn} 2^{-n} \quad (3.2)$$

Substituting Eq. 3.2 in Eq. 3.1, we get

$$Y = \sum_{k=1}^N A_k \left[-b_{k0} + \sum_{n=1}^{W-1} b_{kn} 2^{-n} \right] \quad (3.3)$$

$$Y = \sum_{k=1}^N A_k \sum_{n=1}^{W-1} b_{kn} 2^{-n} + \sum_{k=1}^N A_k (-b_{k0}) \quad (3.4)$$

The above expression leads to the development of the standard DA algorithm. In the proposed algorithm, changing the order of the vectors in Eq. 3.1, yields the following expression:

$$Y = \sum_{k=1}^N A_k B_k = \sum_{k=1}^N B_k A_k \quad (3.5)$$

The implication of Eq. 3.5 in architectural terms is that instead of the contents of the ROM being pre-calculated, priori knowledge of the locations of the ROM to be accessed is now available. Assuming B_k to be the constant coefficient vector, the N locations accessed in the ROM are known in advance. The matrix of addresses used to access the ROM is shown below:

$$\begin{bmatrix} b_{10} & b_{20} & \cdot & \cdot & b_{N0} \\ b_{11} & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ b_{1(W-1)} & \cdot & \cdot & \cdot & b_{N(W-1)} \end{bmatrix} \quad (3.6)$$

If $N \gg W$, a much smaller RAM or even array of registers consisting of an N to T decoder such that $T = \log_2 W$ can be used. Let the T addresses of the RAM be represented by $(R_0; R_1 \dots; R_{T-1})$. In general, the i^{th} RAM address R_i is the decoded value of the following expression:

$$\sum_{k=0}^{N-1} b_{k[-i+(W-1)]} 2^{(N-k)}; \forall 0 \leq i < W - 1 \quad (3.7)$$

It can be seen in Eq. 3.7 that k starts from “0” instead of “1” as seen in Eq. 3.1. This is accommodate the fact that the first address location in a regular RAM is denoted by the address value “0H”.

The contents of the RAM are calculated at runtime as shown in Eq. 3.8:

$$[R_i] = \sum_{k=1}^N A_k b_{k[-i+(W-1)]} \quad (3.8)$$

Since all the 2^T RAM encoded addresses are known beforehand, the number of 1’s in the term $b_{k[-i+(W-1)]}$ are already known for the 2^T addresses. Hence, the calculation of R_i can be done in parallel and optimised by applying scheduling.

3.2.2 Architectural Details

The architecture of the proposed algorithm is described in Fig. 3.1. The ROM that is usually present in the standard DA architecture is replaced by an array of registers or a RAM. Using a RAM may require the insertion of delays in the cascaded adder structure as only one RAM address can be accessed at a time. The contents of the array/RAM are then sequentially sent to a scaling accumulator through a data bus to generate the final result. The internal structure of the adder blocks used is data dependant and is allocated at compile-time. Each adder block in the cascaded structure is scheduled by due the availability of priori knowledge about the coefficients.

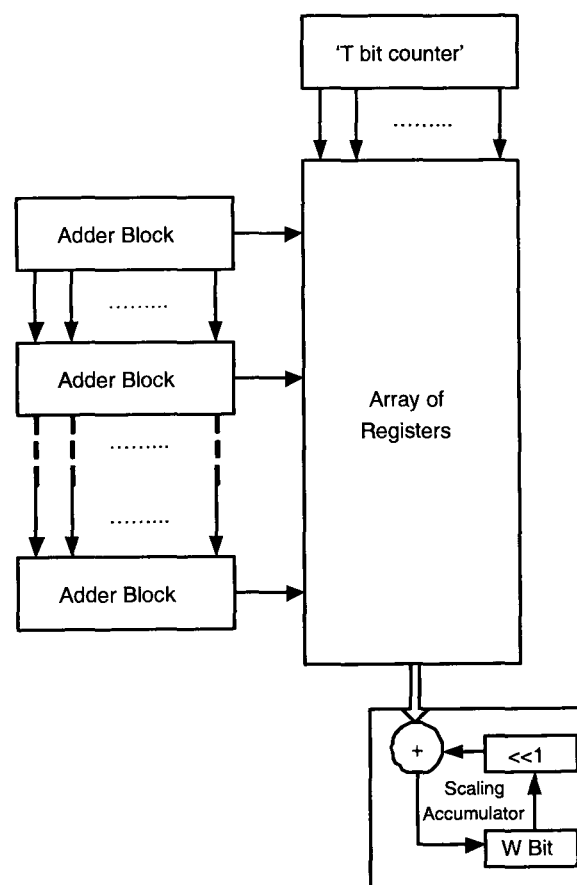


Figure 3.1: Architectural block diagram

3.2.3 Key Performance Measures

The two key performance measures that need to be considered for the architecture of the proposed algorithm are area and latency, both of which will have an effect on the

power consumption [57]. Design space exploration and analysis is important because larger area and longer latency both contribute to an increase in power consumption.

Additional Area Occupied

As the comparison is made with the standard DA approach, it is adequate to highlight the extra logic circuit area occupied, which is proportional to the total number of extra additions that need to be performed for populating the RAM.

$$Adds = \left(\sum_{k=1}^N \sum_{n=0}^{W-1} b_{kn} \right) \quad (3.9)$$

The additions are carried out in parallel branches of an adder tree, which can be scheduled since priori knowledge about the coefficients of the constant vector is available.

Additional Latency

Since the values needed for populating the RAM are calculated at runtime, there is a penalty in the latency of the proposed algorithm when compared with the standard DA. Since the maximum number of addition operations in each branch of the adder tree is $W - 1$ and the maximum height of the tree is given by $\log W + 1$, the total additional latency compared to standard DA, without application of scheduling is given by:

$$Latency = \log \left[\text{Max} \left[\sum_{n=0}^{W-1} b_{kn} \right] \right]; \quad k = 0, 1, 2, \dots, N - 1 \quad (3.10)$$

The total number of scaled accumulation operations needed is identical to the conventional DA.

3.2.4 Application of Scheduling

In the unscheduled case illustrated in Fig. 3.2, the design is sub optimal in terms of both area and latency, due to the presence of redundancy. Further optimisation by minimising the latency and reducing total number of adders can be performed

by means of using Force Directed Scheduling (FDS) [117], within the constraint of the total number of time-steps available for scheduling the entire adder tree. Thus, a force is associated with each feasible combination corresponding to its power cost.

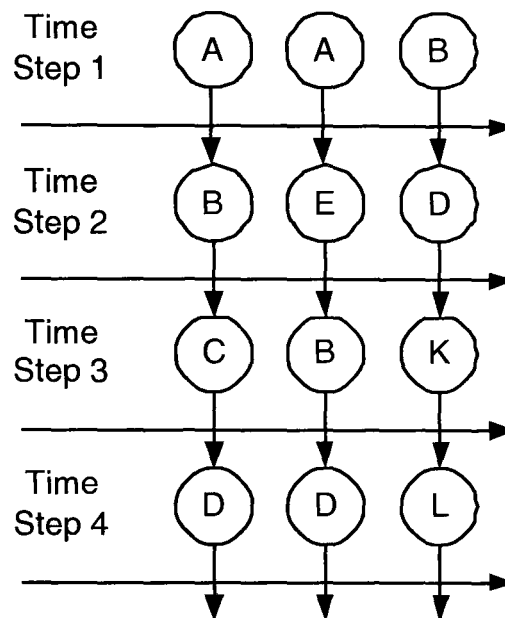


Figure 3.2: Unscheduled dataflow

The Self Force (SF) can be calculated as follows:

$$SF(j) = \sum_{i=S_j}^{L_j} DG(i)x(i) \quad (3.11)$$

where L_j represents the maximum latency. For each possible assignment of a node d_j belonging to any branch is assigned to a time-step TS_j , to be executed within the node's time period ($0 < t_j \leq TS_j$). $DG(i)$ is the distribution value at time-slot i , $x(i)$ is the change in probability associated with that time-slot. The optimisation tradeoff between area and time-slots is achieved by setting the value of the total time period for the entire tree. The FDS condition given by:

$$\sum_{\forall j} L_j d_j < \log \left[\text{Max} \left[\sum_{n=0}^{W-1} b_{kn} \right] \right] \quad (3.12)$$

The above expression provides us with a latency optimised design by splitting a single branch of the tree into parallel branches. This increases the number of registers required for intermediate steps in the computation but gives the benefit of reduced

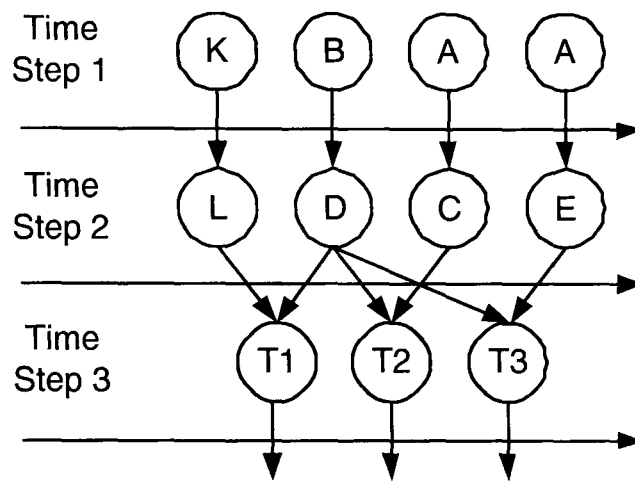


Figure 3.3: Latency optimised scheduling

latency (Fig. 3.3). On the other hand, the condition given by Eq. 3.13 yields an area optimised design by increasing the number of available time-slots and permitting reordering of the nodes in a branch (Fig. 3.4). For energy optimised design it is important to take into consideration the actual value of the coefficients to determine the most optimal design choice.

$$\sum_{\forall j} L_j d_j > \log[\text{Max}[\sum_{n=0}^{W-1} b_{kn}]] \tag{3.13}$$

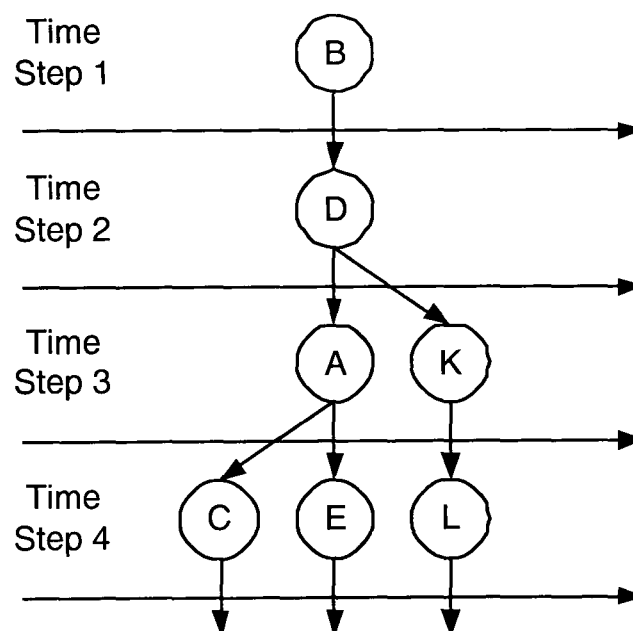


Figure 3.4: Area optimised scheduling

The overall scheduling strategy is decided on an ad-hoc basis because the actual values of the schedule obtained depends upon the actual values of the constant coef-

ficients involved in the DA process. A code snippet displaying the latency optimised schedule generated for a particular DA instance is presented in Algorithm 1.

Algorithm 1 Pseudocode snippet for latency optimised schedule for novel DA implementation, vector length = 12

```

1: par{ temp0 = 0; temp1 = 0@(vv1+vv2);
2: temp2 = 0@(vv1+vv3+vv4);
3: temp3 = 0@(vv1+vv5+vv4);
4: temp4 = 0@(vv1+vv3+vv4+vv6+vv7+vv12+vv11+vv10);
5: temp5 = 0@(vv1+vv8+vv3+vv9+vv8+vv4+vv2+vv7);
6: temp6 = 0@(vv1+vv8+vv5+vv6+vv9+vv11+vv10+vv2+vv7);
7: temp7 = 0@(vv1+vv8+vv5+vv6+vv3+vv11+vv10+vv10+vv7);
8: temp8 = 0;
9: temp9 = 0@(vv1+vv2+vv3+vv4+vv5+vv6+vv7+vv8+vv9);
10: temp10 = 0@(vv1+vv8+vv3+vv9+vv8+vv4+vv2+vv7);
11: temp11 = 0@(vv1+vv5+vv3+vv6+vv8+vv4+vv2+vv7);
12: temp12 = 0@(vv1+vv2+vv3+vv4+vv5+vv6+vv7+vv8+vv9); }
13: for i = 0 : 12 do
14:   par { t1 = t[0]@(t1[10:1]); t = 0@t[12:1]; }
15:   par { t= (0@(temp0)) + t; i++; temp0=temp1; temp1=temp2; }
16:   par { temp2=temp2; temp3=temp4; temp4=temp5; temp5=temp6;
temp6=temp7; }
17:   par { temp7=temp8; temp8=temp9; temp9=temp10; temp10=temp11; }
18:   temp11=temp12;
19: end for

```

3.2.5 FPGA Implementation

In order to compare the power/performance trade-offs of the proposed novel DA approach with conventional DA (based on the algorithm presented in [115]), both designs has been prototyped on the Celoxica RC1000 PCI development board [116] [Appendix B].

Performance Metrics

For the test case, vector lengths of 8, 10 and 12, each containing coefficients represented by 8 bits have been used. It can be seen from the results in Table 3.1 that the total number of 4-input LUT's required to implement the design in the case of vector length 12 is reduced by 77%. However, The maximum operating frequency decreases by 2% only. These results can be interpreted based on the observation

that the increase in complexity of control circuitry in the case of $W = 12$ is marginal for the modified DA approach compared to the increase in the ROM size for conventional DA. This difference is not very significant for cases with smaller vector lengths, as can be seen from the results. The comparison of performance metrics for all vector lengths for modified and conventional DA has been presented in Table 3.1. In this design, since the width of the coefficients in the vector used was 8 bits, in place of a RAM an array of registers has been used to store the outputs of the adder cascade.

Table 3.1: Implementation results

	Length (N)	Area (LUTs)	Max. Freq. (MHz)
Conventional DA	8	132	90.7
	10	344	86.5
	12	833	82.8
Modified DA	8	139	86.4
	10	160	82.2
	12	188	80.1

Power Analysis

Table 3.2: Power dissipation data

	Length (N)	Logic Power (mW)	Signal Power (mW)
Conventional DA	8	3.03	2.85
	10	6.23	5.22
	12	8.56	10.63
Modified DA	8	3.35	3.89
	10	3.44	3.98
	12	3.45	5.88

Power measurements were performed using Xilinx XPower [11]. The total dynamic power is computed as the sum of the logic and signal power. I/O power remains that same for both implementations, and is thus not considered for analysis. Power data is presented in Table 3.2. It can be seen from Fig. 3.5 that overall dynamic power consumption is reduced by 51%.

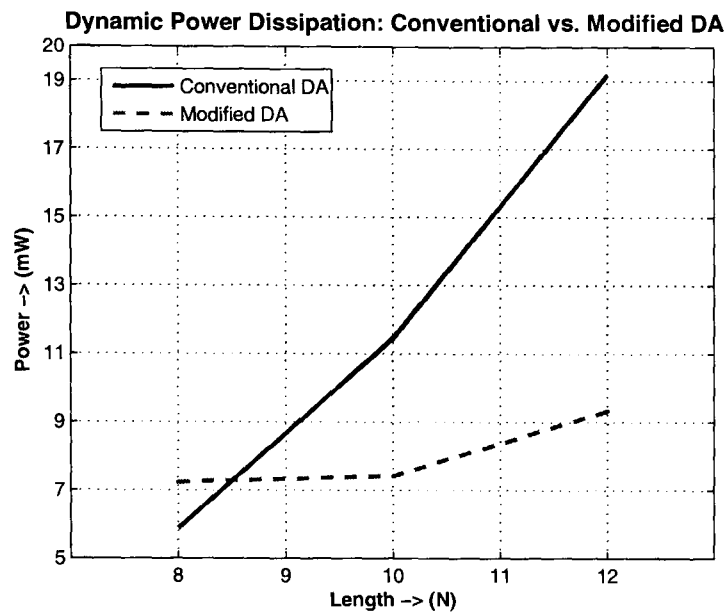


Figure 3.5: Comparison of dynamic power consumption

3.3 Novel Sparse OBC based Distributed Arithmetic Architecture for Matrix Transforms

Matrix multiplication forms the basic building block of a number of signal processing algorithms, including transformation kernels that are used in a number of image and signal processing applications. Key among these are DOTs that belong to the domain of linear transformations, which are mathematically well-founded [118]. Since DOTs are linear transformations on matrices, they are essentially composed of matrix-matrix or matrix-vector multiplications, which are computationally expensive. To process a N -length vector denoted by ϕ according to a transformation function given by $g = A \cdot \phi$ (where A is a square matrix) requires $O(N^2)$ operations. For large values of N , this is clearly a problem. There is a real need for dedicated processors for high speed computation of the transform to meet the requirements of real time signal processing. The choice of suitable arithmetic techniques for performing InP is also an important factor.

In this section, a mathematical framework for implementation of OBC based DA, in conjunction with matrix factorisation and sparse matrix techniques is discussed. Tradeoffs between various performance metrics and power are evaluated and a comparison is made with the standard DA approach in order to quantify the gains

achieved through the transformation processes.

3.3.1 Mathematical Background

Standard DA

Direct applications of the standard DA algorithm is mathematically presented as follows. Consider an InP of two vectors represented as a sum of products, and represented mathematically as seen in Eqs. 3.1 – 3.4.

The term $\sum_{k=1}^N A_k b_{kn}$ in Eq. 3.4 can have only 2^k possible values, it is possible to pre-compute and store these values in a ROM. By addressing this ROM through W cycles using the input data and performing simple shift-accumulate operations, the InP can be calculated. In order to accommodate both the terms in Eq. 3.4, a $2 \cdot 2^k$ word ROM is required to store the pre-computed data. By using an adder/subtractor block; this can be reduced to a size of 2^k . By using DA, we can clearly see that a typical InP has been reduced in complexity from $O(N)$ multipliers to just $O(N - 1)$ additions and shift operations. This represents an order of magnitude reduction in computational circuit complexity, which is traded off for memory of size 2^k , making DA ideal for resource constrained systems. However, the keen observer would notice that the reduction in computational complexity by offloading logic to memory does not necessarily result in a reduction of area complexity. This makes reduction of ROM size, without significant increase in control overhead a key requirement for efficient hardware implementation of DA.

OBC based DA

The ROM size of standard DA can be further reduced to 2^{k-1} by applying OBC technique. It is worth mentioning that OBC-DA does not necessitate recoding inputs or outputs, since it is only used to interpret and not convert the input data to be in OBC form [57]. An element of the input vector can be alternatively expressed as:

$$B_k = \frac{1}{2} [B_k - (-B_k)] \quad (3.14)$$

Substituting Eq. 3.3 in 3.14 we get:

$$B_k = \frac{1}{2} \left[-(b_{k0} - \bar{b}_{k0}) + \sum_{n=1}^{W-1} (b_{kn} - \bar{b}_{kn})2^{-n} - 2^{-(W-1)} \right] \quad (3.15)$$

Let $c_{kn} = b_{kn} - \bar{b}_{kn}$ $n \neq 0$ and $c_{k0} = -(b_{k0} - \bar{b}_{k0})$. Eq. 3.15 is now expressed as:

$$B_k = \frac{1}{2} \left[\sum_{n=0}^{W-1} c_{kn}2^{-n} - 2^{-(W-1)} \right] \quad (3.16)$$

Substituting Eq. 3.16 in Eq. 3.2 we get:

$$Y = \sum_{n=0}^{W-1} Q(b_n)2^{-n} + 2^{-(W-1)}Q(0) \quad (3.17)$$

where

$Q(b_n) = \sum_{k=1}^N \frac{A_k}{2} c_{kn}$ and $Q(0) = \sum_{k=1}^N \frac{A_k}{2}$. The lower half of the ROM table obtained from the pre-computation of the OBC DA combinations is a mirror image of the upper half, with sign reversed. By using the most significant address line (first element in the input vector) as an inverting signal, it is possible to reduce the ROM size by a factor of two, when compared to regular DA. Theoretically, it is possible to recursively apply OBC to reduce the ROM size, at the expense of additional logic. For the maximum case of recursion, the logic depth becomes extremely large and the purpose of trading computational complexity for memory is defeated.

DOT Manipulation for Complexity Reduction

For most unitary transforms (and for N an integral power of 2) with the exception of Karhunen Loeve Transform (KLT) (because of data dependency), fast algorithms exist. They are essentially based on the fact that the transformation kernel can be partitioned into some intermediate steps which can be subsequently reused in further iterations. The factorisation and partitioning techniques for each individual DOT is specific to the kernel structure and must be applied on a case to case basis. For example, in the case of the DFT, Cooley-Tukey factorisation is used as follows:

$$A = A_1 A_2 \dots A_m \quad (3.18)$$

where and A_1, A_2, \dots, A_{m-1} are 2-sparse matrices and A_m is $N/(2^{m-1})$ sparse. For a 1-D DFT the area complexity of the OBC-DA is now reduced to:

$$O\left(2\left[(m-1) + 2^{\left(\frac{N}{2^{m-1}} - 2\right)}\right]\right) \quad (3.19)$$

For a 2 dimensional case (matrix-vector), such as the N -point WHT or N -point HWT, sparse matrix factorisation can be effectively exploited for reducing the ROM size to:

$$O\left(4(m-1) + \frac{N}{2^{m-1}} \left(\frac{N}{2^{m-1}} - 1\right)\right) \quad (3.20)$$

as compared to $O(2^N)$ for the conventional DA approach. In the case of 8-point DCT, Chen's algorithm [119] is used to exploit symmetry of the coefficients effectively reducing an 8×8 matrix operation into two 4×4 operations. This is effectively the same as ROM decomposition which is also an effective technique for minimising ROM size in DA. Further reduction in ROM size when compared to conventional DA can be achieved by applying OBC technique. Quantitatively, for an N -point DCT, applying Chen's decomposition in conjunction with OBC reduces the area complexity of the algorithm to $O(N(2^{\frac{N}{2}} - 1))$.

It is clear that sparse matrix decomposition techniques when used in conjunction with OBC can yield highly efficient and compact structures with minimal overhead and are highly suitable for resource constrained systems [120]. A full discussion of all existing decomposition techniques is beyond the scope of this work.

3.3.2 Proposed Architecture and OBC-DA Operation

The architecture of the whole system is presented in Fig. 3.6. It can be seen that the output of each DA stage is sequentially passed on to the following stages. It is worthwhile to mention that complete sparse factorisation, or reducing all but the last matrix to 2-sparse matrices is not necessary and the exact level of factorisation and number of factors need to be evaluated on a case to case basis for each DOT by the designer and the best tradeoff between latency (pipeline length), complexity and DA size needs to be carefully evaluated. In the case of Fig. 3.6, a generic architecture

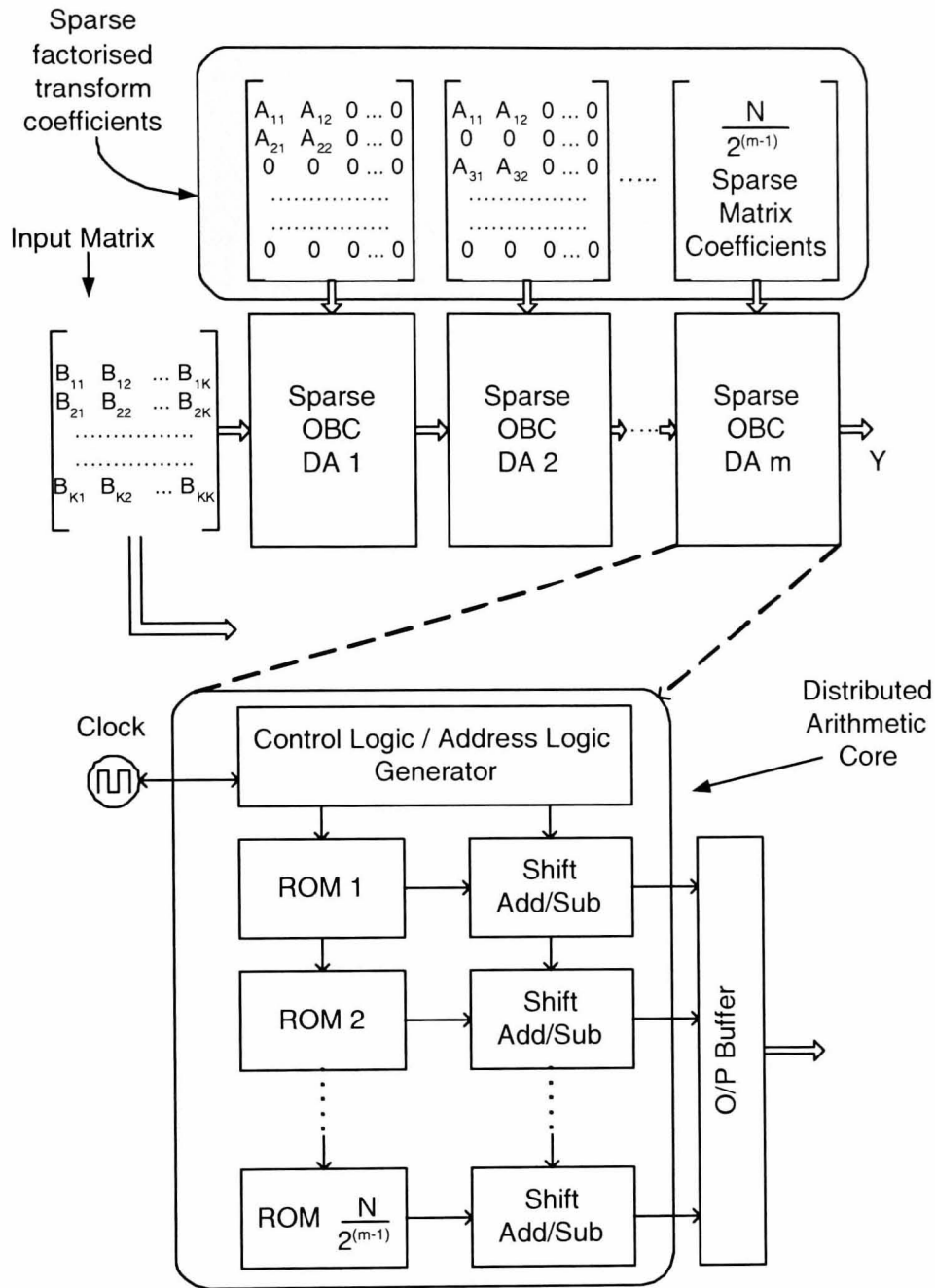


Figure 3.6: Block diagram of the overall OBC-DA-sparse matrix based DOT implementation

for $(m - 1)$ 2-sparse factors and one additional $N/(2^{(m-1)})$ sparse matrix has been presented.

Mapping the OBC-DA algorithm stated mathematically in Eq. 3.17 yields the architecture shown in Fig. 3.7.

The operation of the OBC-DA core for an example case $W = 8$ and $N = 4$ is described as follows. During the 1st clock cycle, the signal T_{s2} in Fig. 3.7 is asserted and the value $Q(0)$ is preloaded into the accumulator. For the first 7 clock cycles, the address locations of the OBC-DA ROM are selected using a bitwise XOR com-

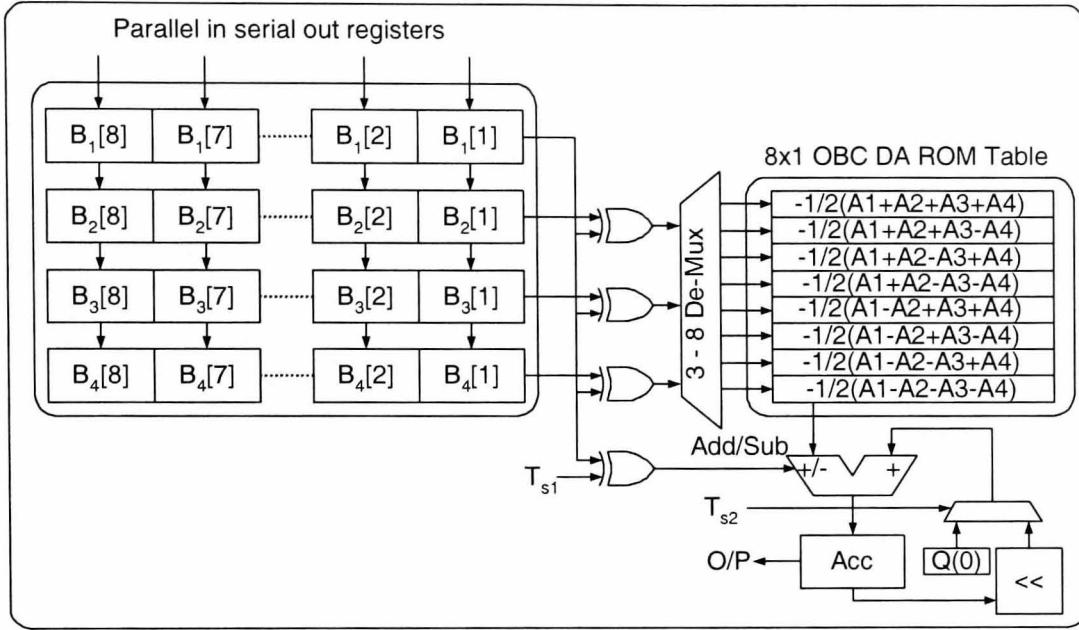


Figure 3.7: Architecture for the OBC-DA block for the case $N = 4$ and $W = 8$

combination of the first element of the variable vector in the InP, B_1 with all the other elements $B_2 \dots B_4$ starting with the LSB of each element. The mode of operation of the accumulator is controlled by the Add/Sub line. On the 8th clock cycle, the signal T_{s1} is also asserted and the final result is obtained at the end of the cycle. At the end of this process, the input vector of the InP is reloaded and the entire operation is repeated. This circuit can be easily generalised for all feasible values of W and N .

3.3.3 Implementation Details and Results Obtained

The IP cores developed for comparing various metrics of regular DA and OBC-DA have been implemented on the Xilinx Virtex-II-Pro XC2VP100 SRAM FPGA [36], [Appendix B] which is capable of handling large and complex designs. The choice of this high performance platform for prototyping the algorithms presented is influenced by the fact that InP is a highly computationally intensive operation.

Performance Metrics

Performance metrics obtained for the implementation of standard DA and OBC-DA are presented graphically in Fig. 3.8.

The graph yields some interesting insights into the tradeoffs and general trend for

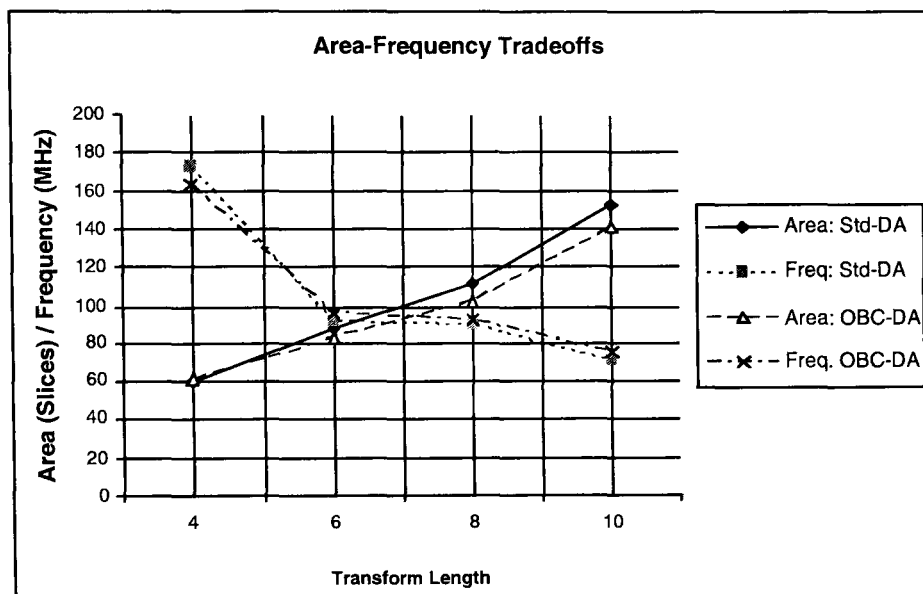


Figure 3.8: Frequency-Area trends for both architectures for different values of N and $W = 8$. Frequency is in MHz and area is represented in FPGA slices.

both cases. With respect to area, it can be clearly seen that for the case $N = 4$, the area of OBC-DA implementation is in fact marginally higher than standard DA. This can be attributed to the fact that the increase in circuit complexity slightly outweighs the influence of reduction of ROM size. However, for larger values of N , a divergent trend is clearly observed and we can see that OBC-DA implementation consumes less area when compared to the standard DA implementation. This gap becomes wider as N increases. In terms of frequency, for all cases of $N \geq 6$, it can be seen that OBC-DA outperforms standard DA. This indicates that the area reduction achieved in OBC-DA is not at the expense of performance. The overall trend of area increase and frequency reduction is similar for both architectures. This can be explained by the fact that maximum frequency is a function of logic depth of the critical path, which increases at the same rate for both architectures, unlike in the case of area.

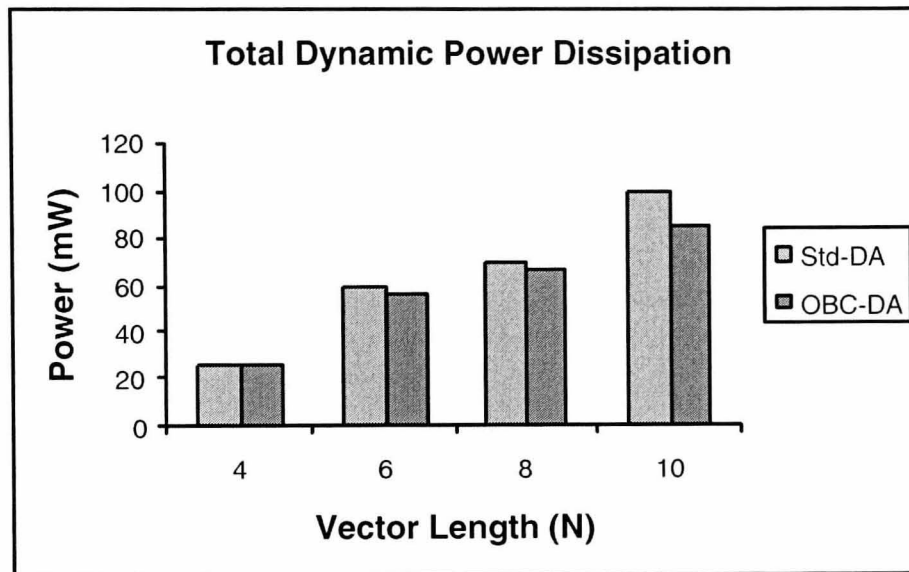
3.3.4 Power Analysis

Power metrics for both architectures at constant frequency of 50MHz are presented in Table 3.3.

Total dynamic power consumption of both DA implementations is presented in Fig.3.9. It can be seen that as N increases, power dissipation of both cores in-

Table 3.3: On-chip dynamic power at constant frequency

N	Standard DA			OBC-DA		
	Clock	Logic	Signal	Clock	Logic	Signal
4	4.68	9.46	11.9	3.93	9.5	12.68
6	12.29	13.37	33.85	13.61	12.86	30.28
8	13.42	16.56	39.67	13.99	15.75	36.93
10	14.55	23.46	61.61	13.47	22.24	49.23

Figure 3.9: Total dynamic power dissipation for different values of N and $W = 8$.

creases. However, OBC-DA is more energy efficient than standard DA for any given value of N . I/O power for both architectures are identical at 3.44 mW for input and 57.58 mW, 60.78 mW, 60.78 mW, 63.98 mW for output at $N = 4, 6, 8, 10$ respectively. This is completely in accordance with expectations as no change has been made to the I/O sections of the two DA implementations. It can be seen from Fig. 3.9 that for $N = 10$, a 12% reduction in total dynamic power has been achieved.

3.4 Efficient FPGA Implementation of FHT for Signal Processing

It is well known from the literature [121] that the FHT belongs to the class of rectangular real valued DOTs and can be efficiently used for the calculation of the DFT for implementing adaptive filters and spectrum filter realisations. The usual frequency domain FIR and IIR filtering problem can easily be converted into a Walsh

frequency domain-filtering problem. The advantages of the 2-D FHT, which is based on 1-D FHT also known as “*S*” or Sequential transform in lossless image compression, are well known [122]. The FHT is also an important tool in the fields of speech processing and error correction [122, 123]. The simple structure and orthogonality of the FHT has found significant use in the generation of pseudo noise sequences for spread spectrum methods of communication such as CDMA. Recent applications include quantum information processing and quantum cryptography, where FHT is used as a gate [124, 125]. Some other applications of Hadamard and other related transforms (Walsh, Paley) are described in [121].

In order to address the issues related to the efficient and power aware implementation of FHT on reconfigurable hardware, a number of solutions have been presented in this work which can be summarised as follows:

- The FHT algorithm has been modified using sparse matrix factorisation methodology which results in quadratic reduction in ROM size and greatly simplifies the DA implementation;
- Parallelism and pipelining have been explored at the architectural level to improve performance. Additionally, the parameterisable VHDL cores from Xilinx Coregen have been judiciously used to yield highly optimised designs; and
- Functional Level Power Analysis and Modelling (FLPAM) methodology (described in Chapter 6) has been used for early design space exploration and a priori estimation of power and energy metrics based on various system parameters included in the model.

3.4.1 Mathematical Background

Typically, a Hadamard matrix is defined iteratively as:

$$H_{2N} = \begin{pmatrix} H_N & H_N \\ H_N & -H_N \end{pmatrix} \quad (3.21)$$

where H_N is a Hadamard matrix of size $N \times N$ and

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (3.22)$$

Let \circ denote the Kronecker product between two matrices:

$$H_N = H_2 \circ H_{N/2} \quad (3.23)$$

Recursively expanding this expression, we get:

$$H_N = \prod_{i=1}^K (I_i \circ H_2 \circ I_{N/2^i}) \quad (3.24)$$

The above expression yields K sparse matrices on complete decomposition. If the transform length N is a power of two the decomposition described in Eq. 3.24 yields an algorithm which can be used to generate high speed and efficient architectures for the FHT.

Let the input data and the transformed data be represented by the two vectors X and Y of size N . Then Y can be written as follows:

$$Y = H_N X \quad (3.25)$$

such that:

$$Y_i = \sum_{k=0}^{N-1} H_{N,ik} X_k \quad (3.26)$$

where X_k 's are written in the fractional format as shown in equation 3.27:

$$X_k = -x_{k,W-1} + \sum_{m=1}^{W-1} x_{k,W-1-m} 2^{-m} \quad (3.27)$$

where $x_{k,m}$ is m^{th} bit of x_k (which are zero or one) and $x_{k,W-1}$ is the sign bit, where W is the word-length.

Substituting 3.27 in 3.26:

$$\begin{aligned}
 Y_i &= \sum_{k=0}^{N-1} H_{N,ik} \left(-x_{k,W-1} + \sum_{m=1}^{W-1} x_{k,W-1-m} 2^{-m} \right) \\
 &= - \sum_{k=0}^{N-1} H_{N,ik} x_{k,W-1} + \sum_{m=1}^{W-1} \left(\sum_{k=0}^{N-1} H_{N,ik} x_{k,W-1-m} \right) 2^{-m}
 \end{aligned} \tag{3.28}$$

Define:

$$H_{W-1} = - \sum_{k=0}^{N-1} H_{N,ik} x_{k,W-1} \quad (m = 0) \tag{3.29}$$

and

$$H_{W-1-m} = \sum_{k=0}^{N-1} H_{N,ik} x_{k,W-1-m} \quad (m \neq 0) \tag{3.30}$$

The output result is given by:

$$Y_i = - \sum_{m=0}^{W-1} H_{W-1-m} 2^{-m} \tag{3.31}$$

Since the term H_m depends on the $x_{k,m}$ values and has only 2^N possible values, it is possible to pre-compute and store these values in a ROM. An input set of N bits $(x_{1,m}, x_{2,m}, \dots, x_{N,m})$ is used as an address to retrieve the corresponding H_m values. A direct implementation of Eq. 3.25 would require (in the case of $N = 8$ and $W = 8$) 56 additions and subtractions. However, with the use of the FHT the number of the addition and subtraction operations is reduced to 24.

To reduce the number of arithmetic operations and to speed up the process, the symmetry in the FHT matrix coefficients and a sparse matrix factorisation are exploited as shown in Eq. 3.34, 3.35 and 3.36.

The matrix multiplication described in Eq. 3.36 is now performed using DA. Eq. 3.36 is essentially a matrix vector computation, where each element of the output vector Y is formed by multiplying each of the 8 elements of the input vector by one of 8 predefined coefficient values. By applying Eq. 3.24 for the case $N = 8$, we get:

$$H_8 = \prod_{i=1}^K (I_i \circ H_2 \circ I_{8/2^i}) \quad (3.32)$$

On recursive expansion of Eq. 3.32:

$$H_8 = (I_1 \circ H_2 \circ I_4) \times (I_2 \circ H_2 \circ I_2) \times (I_4 \circ H_2 \circ I_1) \quad (3.33)$$

Let $H_8^4 = (I_1 \circ H_2 \circ I_4) \times (I_2 \circ H_2 \circ I_2)$ and $H_8^2 = (I_4 \circ H_2 \circ I_1)$.

The transformed matrix can now be represented as:

$$Y = H_N X = H_8^4 (H_8^2 X) = H_8^4 T \quad (3.34)$$

where H_N^4 and H_N^2 are uniformly sparse with 4 and 2 non-zero coefficients respectively.

$$\begin{pmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \\ Y_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 & 0 & -1 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & -1 & 0 & 1 \end{pmatrix} \times$$

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{pmatrix} \quad (3.35)$$

$$\begin{pmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \\ Y_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 & 0 & -1 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} T_0 \\ T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \end{pmatrix} \quad (3.36)$$

3.4.2 Proposed Architecture for FHT - Design and Evaluation

In this section, the novel architecture designed for the implementation of FHT is described followed by a tabular comparison of the various parameters of the proposed architecture with other existing architectures in place.

Architecture Description

The architecture for the 1-D FHT is shown in Fig. 3.10. The $W = 8$ bit inputs to the circuit are fed in byte-serial fashion from the input port. The ADD/SUB block in the input module operates on every odd clock pulse and their output is appended to the input buffer two words at a time (one addition value and one subtraction value) on every even clock pulse in a systolic manner, until the full vector to be transformed has been read.

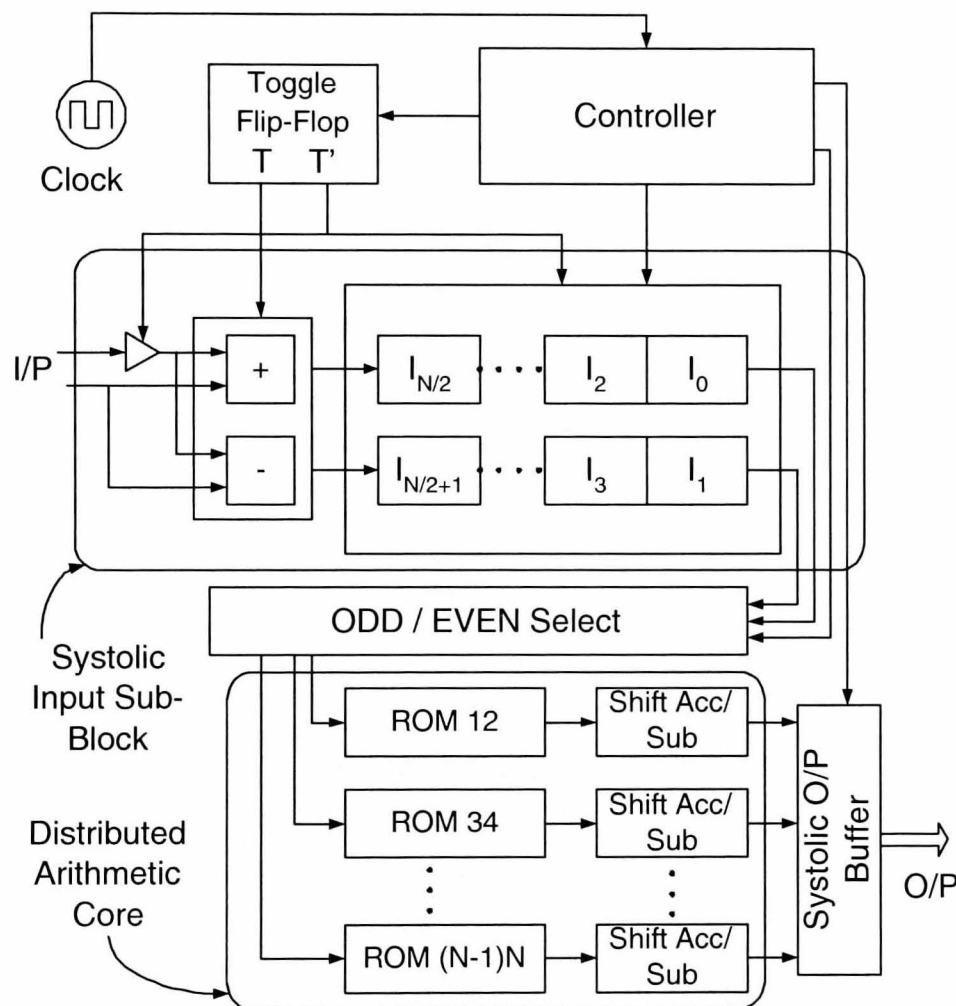


Figure 3.10: Novel DA based architecture for FHT

During the first 9 cycles, eight bit-serial outputs for each even vector are produced in parallel and during the second 9 cycles eight bit-serial outputs for each odd vector are produced in parallel. For the first 8 clock cycles the ADD/SUB module operates in addition mode. During the 9th clock pulse in each iteration, the invert flag is set to high and the ADD/SUB block in the DA module performs subtraction. The Odd + / Even - selector multiplexes the odd/even indexed values in the input buffer to the

DA module. From Eq. 3.36, it can be seen that each odd row vector is redundant as it is the shifted version of each preceding row vector. By splitting the column vector T into two sets of odd and even vectors, the same sets of 4 row vectors can be used in place of the 8×8 matrix H_N^4 . The odd and even vectors obtained have a vector-length of 4 words each thereby reducing the size of the ROM for each vector to 16 words instead of 256. In general, by applying this novel transformation to the Hadamard matrix following its decomposition, the ROM size for each vector used in the DA based implementation is reduced from 2^N to $2^{N/2}$. The novel architecture thus obtained has an area complexity that is reduced by orders of magnitude in terms of ROM area required. At the end of each iteration, the DA module generates output vectors of length $N/2$, for the even and odd vectors in alternation. These vectors are buffered at the output section and written to the output pads in a word serial manner. The contents of the ROMs in the DA module are shown in Fig. 3.11 where $A0 = H_{8,i(6+i \bmod 2)}^4$, $A1 = H_{8,i(4+i \bmod 2)}^4$, $A2 = H_{8,i(2+i \bmod 2)}^4$ and $A3 = H_{8,i(0+i \bmod 2)}^4$

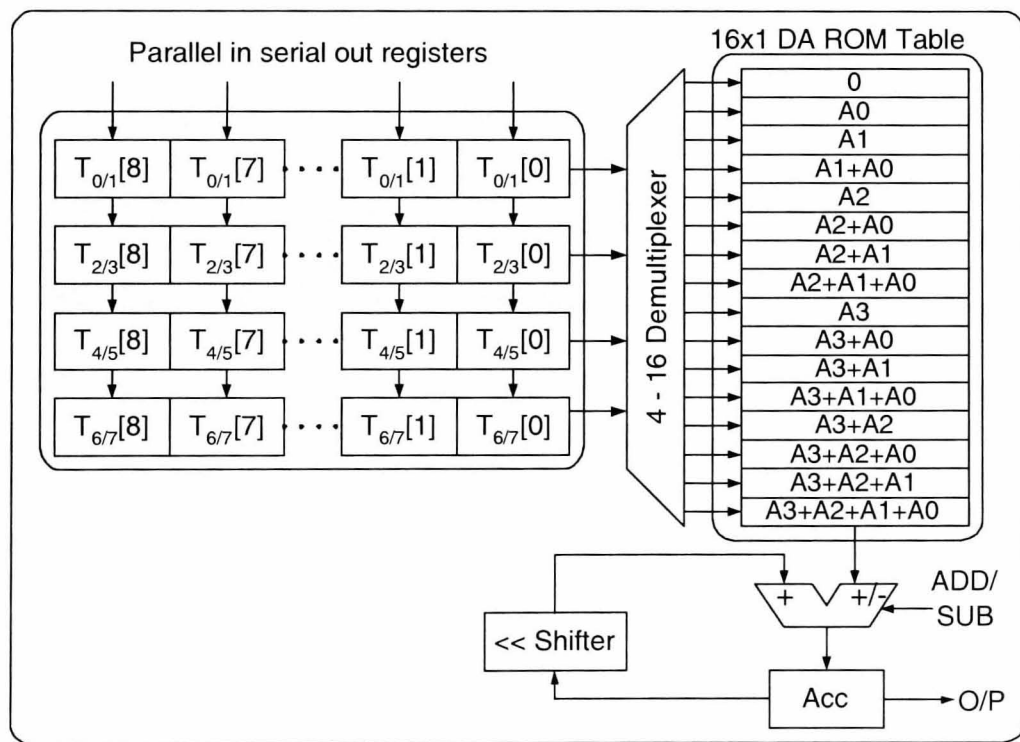


Figure 3.11: DA module structure and ROM contents for $N = 8$, $W = 8$

Comparison with Existing Architectures

Design parameters such as Time Complexity (TC), Area Complexity (AC) and I/O type of the proposed design with existing architectures are presented in Table 5.1.

Table 3.4: Comparison of design metrics with existing architectures

	TC	AC	I/O
Proposed	$O(2(W + 1))$	$O(2^{N/2})$	Serial
[24]	$(2N - 1)(W + \log_2 N)$	$O(n^2)$	Serial
[25]	$O(\log_2 N + W)$	$O(N^2)$	Serial
[27]	NA	$O(N \log_2 N)$	2 Words
[16]	$O(2(W + 1))$	$O(2^{N/2})$	Parallel
[16]	$O((N + 1)W)$	$O(2N)$	Serial
[126]	$O(2N)$	$O(N^2)$	Serial
[127]	NA	$O(N^2 \cdot 2^N)$	NA

It is worth noting that the time complexity of DA based FHT architectures in the proposed design and [16], is dependant only on the number of bits used to represent each word in the vector and not on vector length. Thus, the latency of the design and the number of clock cycles remains constant for a given input wordlength for all vector sizes. On the other hand, in all the other architectures referenced in Table 5.1, the computation time increases with the vector length. It is also worth mentioning that the type of I/O used also has an impact on the I/O power of the design. Serial architectures in general consume lesser I/O power when compared to parallel architectures. However, this cannot be readily assumed for energy measures as well, since the energy consumed is not a function of frequency, but rather depends on the throughput efficiency of the design.

3.4.3 FPGA Implementation

In order to verify the performance of the proposed architecture for FHT, the design has been prototyped on the Celoxica RC1000 PCI development board [116] [Appendix B].

Host-FPGA System

The host application developed accepts two forms of input: input from file and interactive input from the user. Once the input vector has been captured, it is sent to the SRAM Bank 0 by means of DMA transfer. On completion of data transfer, the host sends a signal to the FPGA and releases control over the specified memory bank. The FPGA assumes control over this memory bank and reads the input vector. After transforming the data to the Hadamard domain, the output vector is stored in SRAM Bank 1 and the control is relinquished to the host application, which reads the transformed vector and processes the output as required.

Implementation Results

Implementation results in terms of various performance metrics like area occupied, speed, number of I/O pins etc. for the proposed architecture with $N = 4, 8$ and 16 and on different FPGA platforms are presented in Table 3.5. It can be seen that the frequency variations for different vector lengths are similar across all platforms. This is because the same core has been reimplemented without any changes and the frequency gains obtained are clearly due to the improved fabrication technology of advanced FPGAs. On the other hand, some variations are observed in the area occupied. This is because each Configurable Logic Block (CLB) in the Virtex-4 FPGA series contains four slices and results in related logic being packed within the same CLB, thereby reducing routing complexity and the need to use slices for route through. Both the Virtex-E and Virtex-II Pro platforms have two slices per CLB and hence almost identical area metrics are obtained for these two platforms.

For a given platform, the maximum frequency of a design depends on a number of factors, including:

- Logic depth of the design: which depends on the complexity of the algorithm to be implemented, architectural choices - particularly specific resources used on the FPGA (embedded multipliers, BRAMs etc) and the device characteristics of the platform used (speed grade, circuit technology);
- For a fixed platform and a given parameterisable IP core, the factors that

Table 3.5: Implementation results for different FPGA platforms

	N	Area (Slices)	Freq (MHz)	Area/Freq (Slices/MHz)	I/O Pins
Virtex-E	4	82	127	0.65	19
	8	162	115	1.41	20
	16	335	67	5.00	21
Virtex-II Pro	4	83	204	0.41	19
	8	163	183	0.89	20
	16	377	100	3.79	21
Virtex-4	4	82	227	0.36	19
	8	163	212	0.77	20
	16	358	121	2.97	21

influence the maximum frequency obtained are the parameters of the core; which in the case of the FHT are the vector length. For larger vector lengths, naturally the IP core occupies more area after synthesis on account of larger DA ROMS and data buses; and

- The clock tree also spreads over a larger area and this results in reduction of the maximum frequency, on account of clock skew propagation delays.

The frequency chart for the FHT core implemented on all three FPGA platforms is shown in Fig. 3.12. Analysis of the data shows that maximum Frequency F displays a polynomial relationship with vector length N of the form:

$$F = f_1 \cdot N^{-1} + f_2 \cdot N^{-2} + f_3 \cdot N^{-3} \quad (3.37)$$

Comparison of the results for $N = 16$ with existing work is presented in Table 3.6.

Table 3.6: Comparison of implementation results for the Virtex-E platform

	Area (Slices)	Freq (MHz)	I/O Pins	Area/Freq (Slices/MHz)	Throughput (Mb/sec)
Prop. 8 *	162	115	20	1.41	818
Prop. 16 +	335	67	21	5.00	953
[16] *	124	90	116	1.38	640
[16] *	188	31	134	6.06	220
[29] +'	122	32	NA	3.81	512
[27] +'	71	36	NA	1.97	576

* corresponds to $N = 8$; + corresponds to $N = 16$ and ' corresponds to $W = 1$. It is

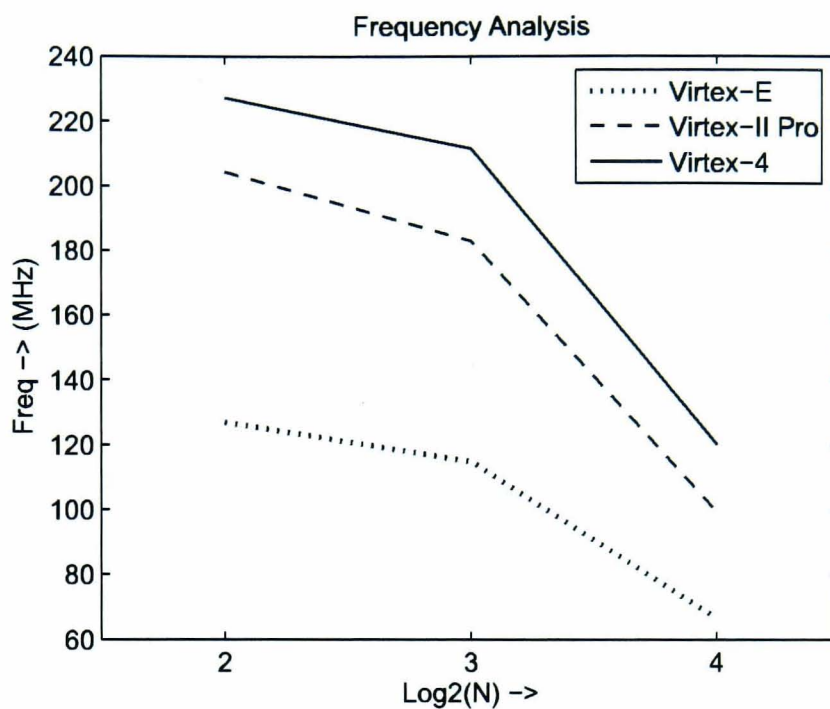


Figure 3.12: Maximum frequency obtained for all platforms : $N = 4, 8, 16$

worth mentioning that the architectures corresponding to '+' are designed for single bit inputs only. Effective bit-level throughput is therefore $1/8^{th}$ times that of the other architectures presented in Table 3.6. The throughput column clearly shows that the proposed architectures outperform other existing architectures in place.

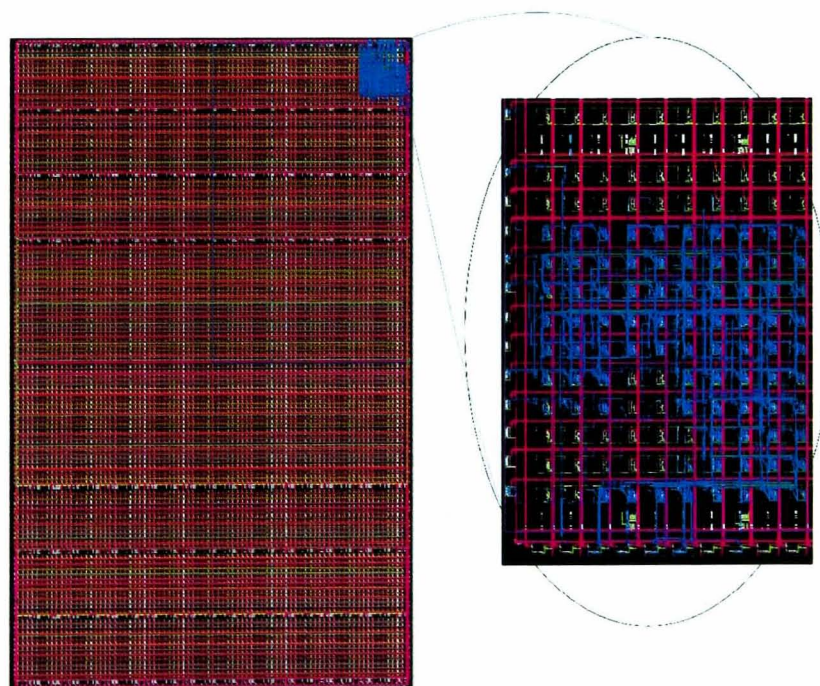


Figure 3.13: Chip diagram showing manual placement, routing and pin assignment; Virtex-E XCV2000E FPGA

Power and Energy Analysis

Power measurements were performed using Xilinx XPower [11]. The components that contribute to dynamic on-chip power are clock, signal and logic power. Dynamic power dissipation of the FHT IP core for different design parameters is presented graphically in Fig. 3.14.

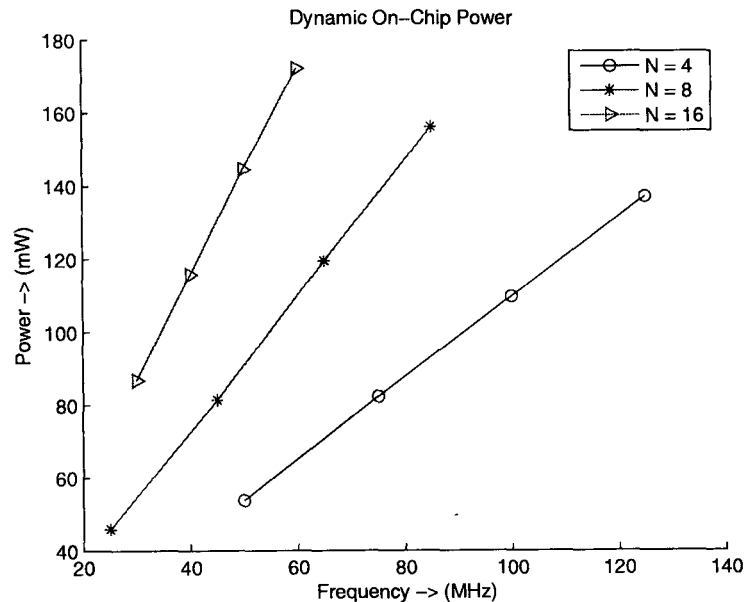


Figure 3.14: Dynamic power dissipation for the FHT core

Energy metrics calculated from the XPower power data are presented in Tables 3.7 and 3.8 respectively.

Table 3.7: *EOP* metrics obtained for $N = 4, 8, 16$

	<i>EOP</i> (nJ)		
	$N = 4$	$N = 8$	$N = 16$
Clock	4.89	12.088	28.981
Signal	8.082	19.635	57.581
Logic	17.792	34.480	62.686
Input	0.084	1.376	2.444
Output	6.724	9.974	15.253

The *EOP* and *EA*¹ values in Tables 3.7 and 3.8 are at $f = 125, 85, 60$ MHz for $N = 4, 8, 16$ respectively. The *EA* values for the output at other frequencies can be deduced by direct proportionality and are in conformance with the observed results.

¹Refer to Appendix D for an explanation about *EOP* and *EA*

Table 3.8: EA values obtained for $N = 4, 8, 16$

	EA (pJ/slice)		
	$N = 4$	$N = 8$	$N = 16$
Clock	58.915	74.157	76.874
Signal	97.373	120.459	152.736
Logic	214.365	211.531	166.276
Input	10.120	8.445	6.483
Output	81.018	61.191	40.460

Chip Level Details

Careful manual place and route of critical nets and manual pin assignment for the designs has been performed using Xilinx Pinout Area Constraints Editor (PACE) and Floorplanner [11]. This process yields compact and optimised design with short nets and serves two important purposes. Firstly, short nets have lesser propagation delay and upto 25% gains in maximum frequency have been achieved. Second, short nets have lesser parasitic capacitance and DC load and therefore dissipate lesser power than long nets. Manual pin assignment also enables us to locate the I/O pads close to the design area, further aiding the above two criteria. The chip diagram for $N = 16$ is shown in Fig. 3.13.

3.5 Efficient FPGA Implementation of GMM-Based Classifier Using DA

GMM based classifier has gained increasing attention in pattern recognition community. Improved classification performances have been demonstrated in many pattern recognition applications such as speaker recognition, handwriting recognition and gas identification, etc [128–135]. Performance figures of more than 95% have already been reported for applications such as EN [136] and gas identification [129]. The GMM can approximate any continuous density with an arbitrary accuracy provided the model has a sufficiently large number of components and provided the parameters of the model are chosen correctly. Another interesting property of GMM is that the training procedure is done independently for each class in turn by constructing

a gaussian mixture of a given class. Adding a new class to a given classification problem does not require retraining the whole system and does not affect the topology of the classifier making it attractive for pattern recognition applications. While GMM provides very good performances and interesting properties as a classifier, it presents some problems that may limit its practical use in real-time applications. It is worth mentioning that GMM can require large amounts of local memory to store various algorithmic coefficients. Hardware acceleration of the GMM is essential in real-time systems because it involves complex computations including matrix multiplications and exponential calculations. It is the aim of this work to develop a novel architecture for the GMM based classifiers using DA principles.

3.5.1 Algorithmic Review of GMM

The task of a pattern recognition algorithm is to set a decision rule, which optimally partitions the data space into c regions, one for each class C_k . The boundaries between regions are the separating surface or decision boundaries. A pattern classifier generates a class label for an unknown feature vector $\underline{x} \in R^d$ from a discrete set of previously learned classes. The most general classification approach is to use the posterior probability of class membership $\wp(C_k|\underline{x})$. To minimise the probability of miss-classification one should consider the maximum a posterior rule and assign \underline{x} to class $C_{\hat{k}}$ [137], such that:

$$[C_{\hat{k}} = \arg \max_{\{1, \dots, c\}} [\wp(C_k|x)] = \arg \max_{\{1, \dots, c\}} [\wp(x|C_k)\wp(C_k)] \quad (3.38)$$

where $\wp(x|C_k)$ is the class-conditional density and $\wp(C_k)$ is the prior probability.

One way to build a classifier is to estimate the class-conditional densities by using representation models for how each pattern class populates the feature space. In this approach, classifier systems are built by considering each of class in turn and estimating the corresponding class-conditional densities $\wp(x|C_k)$ from the data. An alternative method is to combine the advantages of both parametric and non-parametric methods, by allowing a very general class of functional forms in which the number of adaptive parameters can be increased to build more flexible models. This

leads us to a powerful technique for density estimation, called mixture models [138]. In a GMM, a classifier can be constructed by evaluating the posterior probability of an unknown input pattern x belonging to a given class C_k expressed as $\wp(C_k|x)$. This is achieved by using the probability density function $\wp(x|C_k)$, which is in the case of GMM expressed as a linear combination of basis functions $\wp(x|j)$. A model with M components is described as a mixture distribution [138]:

$$\wp(C_k|x) = \wp(C_k)\wp(x|C_k) = \wp(C_k) \sum_{j=1}^M \wp(j)\wp(x|j) \quad (3.39)$$

where $\wp(C_k)$ and $\wp(j)$ are the frequency of a given training sample in the data-set and the mixing coefficients of the component density functions $\wp(x|j)$, respectively. Each mixture component is defined by a Gaussian parametric distribution in d dimensional space

$$\wp(x|j) = \frac{\exp\{-\frac{1}{2}(x - \mu_j)^T \sum_j^{-1} (x - \mu_j)\}}{(2\pi)^{d/2} |\sum_j|^{1/2}} \quad (3.40)$$

The parameters to be estimated are the mixing coefficients $\wp(j)$, the covariance matrix \sum_j and the mean vector μ_j .

A detailed evaluation of the performance of GMM as a classifier using a number of discrimination experiments on various data sets has been presented in [47]. A detailed comparison of the classification performance of the GMM based approach with a wide range of classification algorithms including Multi-Layer Perceptron (MLP), K-Nearest Neighbour (KNN), Radial Basis Functions (RBF) and Probabilistic Principal Component Analysis (PPCA) has also been presented in [47].

3.5.2 Architecture Description

Mapping the GMM algorithm to hardware is a non-trivial task due to the high complexity of the algorithms and storage requirement of parameters. The simplification of GMM algorithm for efficient hardware implementation is discussed in this section. In order to reduce the memory size, new set of parameters (constant K_j and a triangular matrix G_j) are defined and used instead of $\wp(C_k)$, $\wp(j)$, $|\sum_j|^{1/2}$ and

\sum_j^{-1} (where \sum_j^{-1} is a full matrix). The new coefficients K_j and G_j are given by:

$$K_j = \frac{\wp(C_k)\wp(j)}{(2\pi)^{d/2} \left| \sum_j \right|^{1/2}} \quad (3.41)$$

$$G_j^T G_j = \frac{1}{2} \sum_j^{-1} \quad (3.42)$$

G_j is a triangular matrix introduced in order to reduce the complexity as compared to dealing with a full matrix as indicated by Eq. 3.40 calculation. If we assume that:

$$z = \left[(x - \mu_j)^T G_j \right] \left[(x - \mu_j)^T G_j \right]^T \quad (3.43)$$

Eq. 3.39 can be rewritten as:

$$\wp(C_k|x) = \sum_{j=1}^M K_j \exp\{-z\} \quad (3.44)$$

Thus the calculation of $\wp(C_k|x)$ can be divided into three steps: evaluation of parameter z , the exponential calculation and multiplication with constant K_j . It can be noted that the evaluation of parameter z is the most demanding operation. Using Eq. 3.43, we can reduce the complexity of GMM.

In order to obtain a clear understanding of computational requirements, GMM data flow block diagram has been presented in Fig. 3.15. First, $s = x - \mu_j$ is evaluated after d subtractions (d is the dimension of x vector). Next, $y_j = (x - \mu_j)^T G_j$ is calculated using a vector-matrix multiplier. G_j is a triangular matrix. Hence the calculation of y_j requires $\frac{d(d+1)}{2}$ multiplications and $\frac{(d-1)d}{2}$ additions. $z_j = y_j y_j^T$ requires d multiplications and $(d-1)$ additions in the square unit and accumulator. The above calculations are repeated M times to obtain $\wp(C_k|x)$. In addition, GMM also requires a number of memory units in order to store the coefficients, namely μ , G and K .

The subtractor sub-block has been implemented using efficient cores from Xilinx Core-generator. The Vector Matrix (VM) multiplication sub-block shown in Fig.

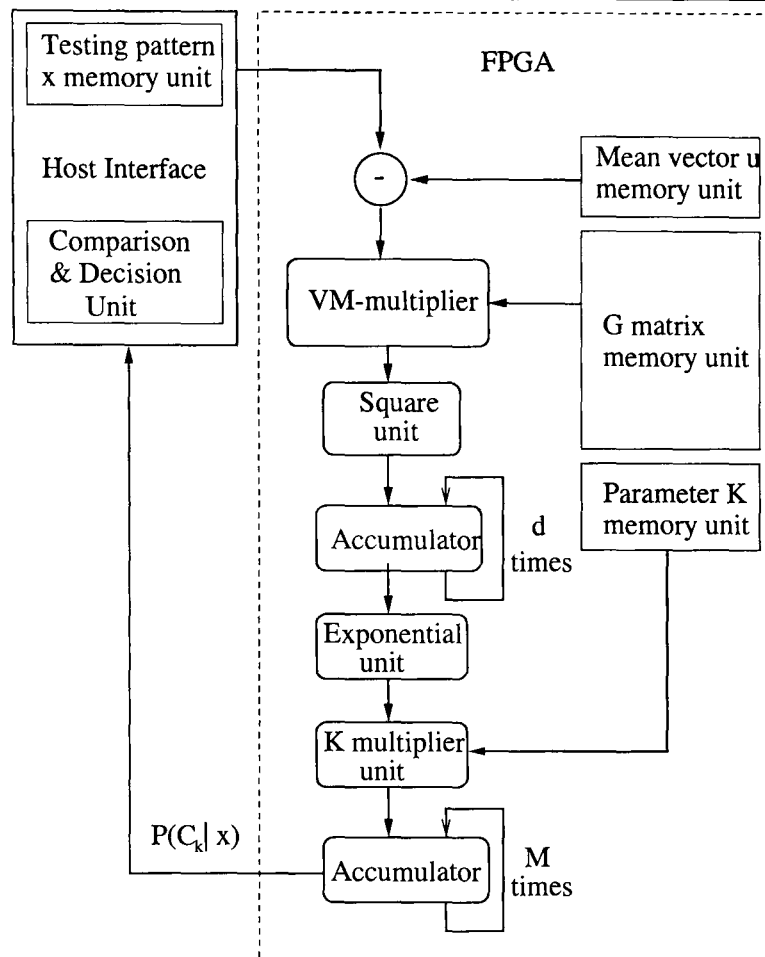


Figure 3.15: Data flow block diagram of GMM classifier. Memory units are used to store various GMM coefficients such as K , G and μ

3.15 has been implemented using DA. The coefficient matrix G is a square matrix with side d . The matrix represented by G is a lower triangular matrix and hence the vector length of each column decreases from d to 1. This means that the d number of ROMs used for the implementation of DA have reducing size from 2^d to 2. The VM multiplication is mathematically defined as follows:

$$VM[i] = \sum_{n=1}^{N-1} \left[\sum_{k=1}^{K_i} G_{ik} s_{kn} \right] 2^{-n} + \sum_{k=1}^{K_i} G_{ik} (-s_{k0}) \quad (3.45)$$

where $0 < i \leq d$ and $VM[i]$ is the output vector after performing the VM multiplication.

The general architectural framework for the VM block using DA has been described in Fig. 3.16

The squaring and accumulator sub-blocks have been efficiently implemented using cores from Xilinx Core-generator. The exponential unit is implemented using a

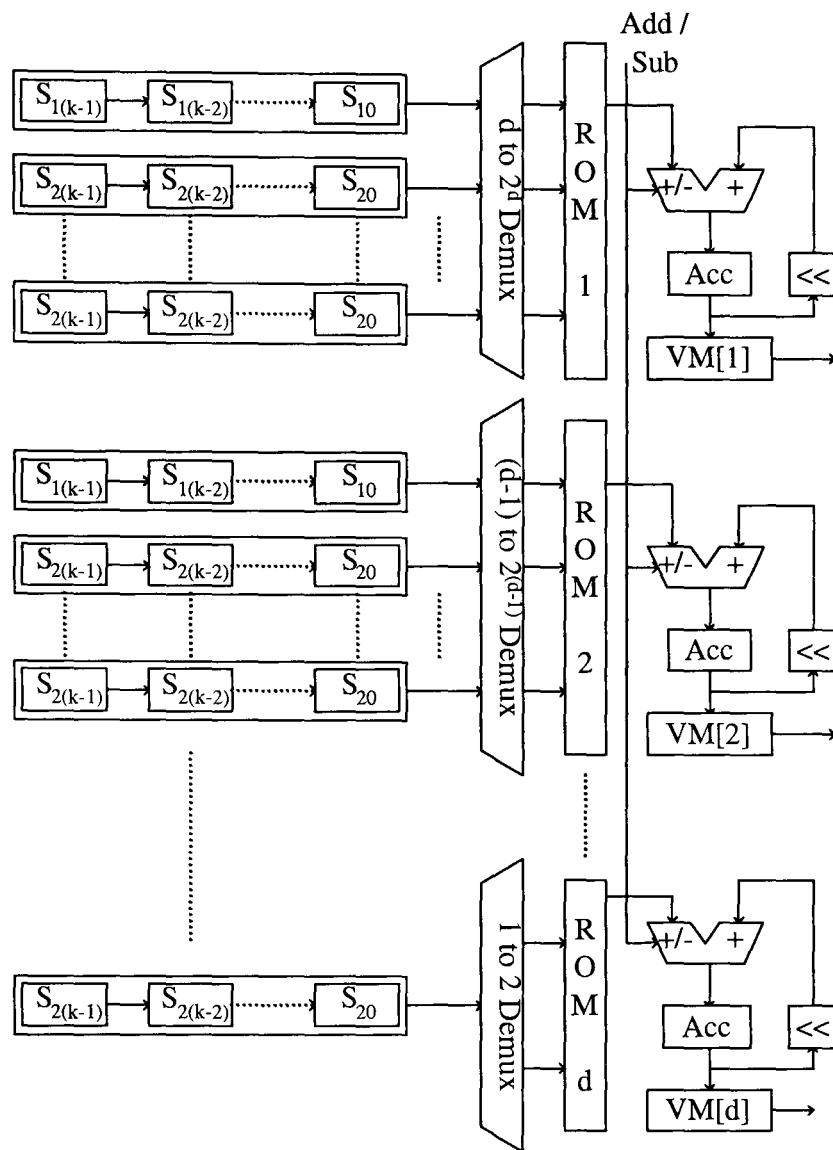


Figure 3.16: DA sub-block for the VM-multiplier

suitable Linear PieceWise function (LPW) [47]. In the gas-sensor application that has been targeted, the specific architectural details are as follows. The resolution of the input data from the gas sensor is 10 bits and the input vector length d is 5. The resolution of the μ vector and G matrix coefficients are 10 bits as well. The wordlengths of intermediate results in subsequent sub-blocks have been retained at full precision to eliminate truncation error which may adversely affect the accuracy of classification. The architecture is fully pipelined at the sub-block level. Each sub-block requires 5 clock cycles (same as the value of d) for processing the data. It is worth mentioning that since the resolution of the DA ROMs' addresses are 10 bits, a single DA sub-block will take 10 clock cycles for complete execution. To synchronise the VM unit with the rest of the sub-blocks in order to keep the pipeline operating

at full speed, two DA units are operated in parallel within the VM-multiplier sub block. It is also worth mentioning that the GMM-kernel that has been implemented on the FPGA can be implemented in different configurations - serial mode with 2 dimensional DA ROMs, fully parallel configuration, partial reconfiguration mode with changing coefficients and semi-parallel configuration. A comparative study of different configuration modes will be performed in future work.

3.5.3 FPGA Implementation

In order to verify the performance of the proposed architecture for the GMM based classifier, the design has been prototyped on the Celoxica RC1000 PCI development board [116] [Appendix B].

Implementation Results

Implementation results in terms of various performance metrics for the proposed DA based architecture for the GMM are presented in Table 3.9.

Table 3.9: Implementation results

Performance Metrics	
Gate Count	27366
Total LUTs	1845
Total Slices	1456
Max Freq	27.010
Bonded IOBs	42

Chip Level Details

Careful manual place and route of critical nets and manual pin assignment for the designs has been performed using Xilinx PACE and Xilinx Floorplanner [11]. This process yields compact and optimised design with short nets and serves two important purposes. Firstly, short nets have lesser propagation delay and upto 25% gains in maximum frequency have been achieved. Second, short nets have lesser parasitic

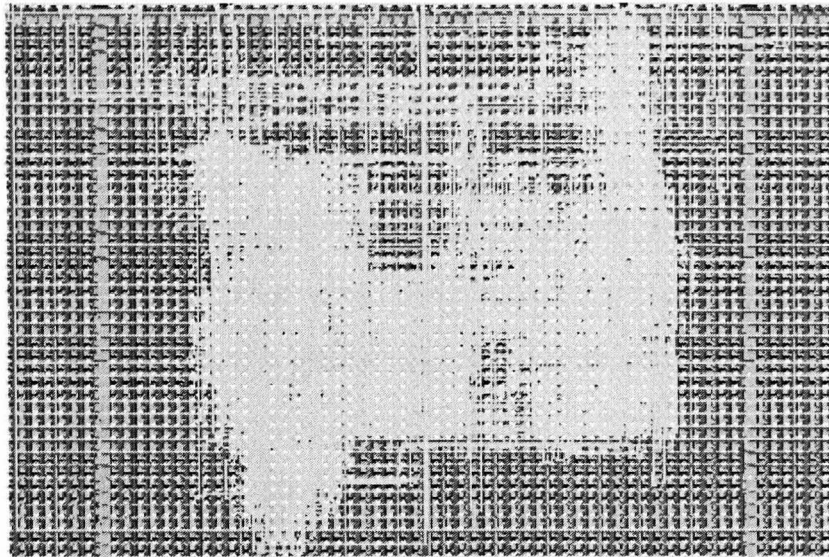


Figure 3.17: Chip diagram showing manual placement, routing and pin assignment; Virtex-E XCV2000E FPGA

capacitance and DC load and therefore dissipate lesser power than long nets. Manual pin assignment also enables us to locate the I/O pads close to the design area, further aiding the above two criteria. The chip diagram is shown in Fig. 4.23

3.6 Conclusions

The kernels of most image processing algorithms including DOTs are built upon inner product computations, which are power intensive due to the large number of multiply operations inherent.

In Section 3.2, a power efficient modification of the DA algorithm has been proposed and significant reduction in power consumption has been verified. The proposed DA algorithm will be suitable especially for vector products with large vector sizes and can be readily implemented for DCT, DHT, DST and many other image processing algorithms.

Implementations of IP, particularly for DOTs, based on sparse factorisation techniques combined with OBC-DA has been presented in Section 3.3. It has been mathematically shown that area complexity can be greatly reduced by using these techniques in conjunction with each other. Despite increased complexity of control circuitry, it has been shown that OBC-DA outperforms standard DA in all key performance metrics including area, frequency and power dissipation. It can be

concluded that OBC-DA is preferable for developing the IP core, particularly in resource (memory) constrained systems.

In Section 3.4, an efficient and optimised architecture for FHT has been presented. This architecture has been developed using a power aware design flow methodology. The proposed novel architecture is based on DA principles and sparse matrix factorisation technique, and is suitable for FPGA implementation. The evaluation of the implementation results has shown that this architecture outperforms existing implementations in all key performance metrics.

In Section 3.5 a pattern recognition system based on GMM classifier has been presented. A highly efficient FPGA implementation of GMM classifier that has been proposed offers a very good balance between hardware complexity and operating speed using DA technique to perform vector matrix multiplication. A prototype has been designed using highly optimised manual placement and routing. Successful FPGA implementation and resultant performance metrics have also been reported. To summarise, algorithmic transformations as a technique of design optimisation has been discussed for a number of IP cores in this chapter. In the next chapter, the application of architectural techniques such as parallelism, pipelining and systolisation for design optimisation will be discussed.

Chapter 4

Architectural Level Optimisation: Parallelism, Pipelining and Systolisation

4.1 Introduction

Parallelism can be used to assign different tasks to different concurrent objects in the design, or to speed up certain iterative operations by performing the same operation on different data-sets concurrently [139, 140]. Parallelism can be exploited in the implementation of matrix operations and transforms such as the FRAT where redundant subexpressions can be effectively parallelised. Although additional hardware resources are utilised, savings can be made in other areas such as minimising buffers and memory elements, reducing frequency to maintain throughput etc. FPGAs display massive parallelism capabilities because of their inherent structure and availability of on-chip resources.

Pipelining is a well known technique used in ASICs for reducing logic depth and improving throughput at the cost of additional latency. Pipelining is most effective for complex repetitive tasks where each task can be broken down into independent sub-tasks (or stages) which can be executed in a sequential manner. The key advantage of pipelining is the reduction of circuit glitching [141]. This is particularly significant in the case of FPGAs, because of limited availability of programmable interconnects.

This can cause designs with large logic depths to use long routing elements which can exacerbate glitching because of unequal delays in signal paths [142].

Systolisation special-purpose computing paradigm that supports the parallel implementation of iterative algorithms in a variety of areas, e.g., numerical analysis, signal or image processing and graph theory [25,143]. A systolic array is a regular network of similar PEs connected to immediate neighbours only. Essentially, systolisation combines the benefits of parallelism and pipelining to yield regular efficient architectures that are well suited for the fabric for FPGAs. The profits of logic depth reduction from a power perspective is reinforced on FPGAs by some peculiarities of these devices: a) neither synchronisation nor off-chip power fraction are high in comparison with the datapath component; b) the die size and clock trees are fixed for a given chip model; and c) the interconnection delay is dominant.

The relative placement of this abstraction level is indicated in the power triangle shown in Fig. 1.7.

In this chapter, a number of FPGA based IP cores including FRAT, FRIT, FIR filters & circular convolution and CSC have been developed by applying principles of parallelism, pipelining and systolisation as appropriate. These architectures have been designed using a power aware design flow and complete design space exploration with power modelling has also been performed. Power modelling details are presented in Chapter 6.

The rest of this chapter is organised as follows. High-speed, power efficient architectures for the FRAT and FRIT are presented in Sections 4.2 and 4.3 respectively. A high performance systolic architecture for FIR filtering is presented in Section 4.4. Efficient FPGA implementation DA based architecture for Colour Space Conversion is discussed in Section 4.5. Concluding remarks are presented in Section 4.6

4.2 Acceleration of Finite Radon Transform on Reconfigurable Hardware

Multiresolution techniques, particularly wavelets have been successfully exploited for devising algorithms for many of these applications. However, the inherent limita-

tions of wavelets in dealing with anisotropic features in images due to their geometric structure, particularly in higher dimensions have been addressed using newly developed transforms based on wavelets such as ridgelets, curvelets and contourlets which are defined by a class of directional, multidimensional basis functions [33, 144]. For example, while wavelets are good at denoising images by isolating point singularities, ridgelets have proven to be much more effective in isolating the smoothness along edges. The FRIT has been used for alternate image representation paradigms, denoising and compression [31, 145]. The ability of ridgelets in handling edge singularities stems from the fact that it is built upon the FRAT [35].

The algorithm for the implementation of FRAT that was proposed in [35] has a serial and iterative structure and hence has a high latency. A suitable translation of this pseudocode (reproduced in Appendix C) is implemented in [21, 34, 37, 146] for the hardware implementation of the FRAT. In previous works, it has become convention to label an architecture based on the straightforward implementation FRAT pseudocode [35][Appendix C] as the *reference architecture*. Direct implementation of this pseudocode by means of using equivalent architectures would lead to either wasted cycles or very deep logic. Suitable modifications to the architecture have been made by parallelising and pipelining certain steps, effectively reducing the depth of logic and also the number of clock cycles required in the reference architecture. It is the aim of this work to develop power efficient architectures for the FRAT, ideally suited for FPGA implementation. A reference serial architecture for the FRAT has been implemented by directly mapping the FRAT algorithm to hardware, in order to make a fair comparison with existing work and to set a benchmark to evaluate the performance gains that can be made of the second architecture by exploiting parallelism and pipelining. At the architecture level, a number of optimisations have also been introduced. These include exploiting specific resources on the FPGA such a block RAM, dual ported memories and register rich structure; reducing the depth of logic by judiciously introducing parallelism and pipelining in the address control logic for both FRAT architectures; and using highly optimised IP cores where possible to yield compact and power efficient design.

4.2.1 The Finite Radon Transform: A Brief Review

The Radon Transform (RT) is an integral transform used to represent an image as a collection of projections along various directions. Sparse representation of image data, especially in images that contain a number of line discontinuities can be effectively achieved using RT. It has enjoyed a position of fundamental importance to many applied problems in mathematics, physical and functional analysis. Applications of RT include seismology, radio astronomy, electron micrography and most famously in tomography [147]. While easy to implement in digital form by discretising the input image, the absence of a corresponding inverse was a key issue. The FRAT was first introduced in [148] as the finite analogue of integration in the continuous Radon transform, with origins in the field of combinatorics. The mathematical representation of an injective form of the FRAT to ensure invertibility when applied on finite Euclidian planes has been presented in [35]. It is worth mentioning that the FRAT is not a discretised version of the RT, but a discrete finite version.

Consider a cyclic group Z_p denoted by $Z_p = (0, 1, \dots, p - 1)$ 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 subgroups, given by:

$$L_{k,l} = \{(i, j) : j = (ki + l)(\text{mod } p), i \in Z_p\}, \quad k < p \quad (4.1)$$

and

$$L_{p,l} = \{(l, j) : j \in Z_p\} \quad (4.2)$$

where each subgroup $L_{k,l}$, 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 then given by:

$$r_k[l] = FRAT_f(k, l) = \frac{1}{\sqrt{p}} \left(\sum_{(i,j) \in L_{k,l}} f[i, j] \right) \quad (4.3)$$

The FRAT is the basic building block for a number of transforms, including the FRIT (known as ridgelets in short), curvelets, etc. A brief discussion of the theory, mathematics and applications of the FRAT and other discretised generalisations

of wavelets in higher dimensions has been presented in Appendix C. It has been also been shown in this appendix that the Filtered Back Projection (FBP) provides a perfect inversion for the FRAT. Also, the algorithm for the FBP and FRAT are synonymous. Hence the same architecture can be used to implement both the forward and inverse transforms.

4.2.2 Proposed Architectures for FRAT - Design and Evaluation

In this section, the architectures designed for the implementation of FRAT are described, followed by a tabular comparison of the various parameters with other existing architectures in place.

Reference FRAT Architecture

The reference architecture for the FRAT and is obtained by suitably applying the pseudocode for hardware implementation. In order 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 (by changing rollover conditions and count limits suitably). The number of counters required remains the same - only the triggering conditions, order and reset logic are modified suitably. It must be highlighted that while 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 I/Os and a serial core. The total latency of the core is $O(p^2(p + 1))$. The input section consists of a 1D RAM of width 8 bits and a depth of p^2 . Although each input image block is a square tile of side p , buffering it in a 1D RAM reduces the computational complexity of the control logic associated with data access. This is because, a 2D RAM is implemented on FPGA as a number of 1D RAMS and uses additional multiplexing logic to dereference the address locations. From the FRAT pseudocode, it is clear that 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 1D 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. The block diagrammatic description of the first FRAT architecture is shown in Fig. 4.1.

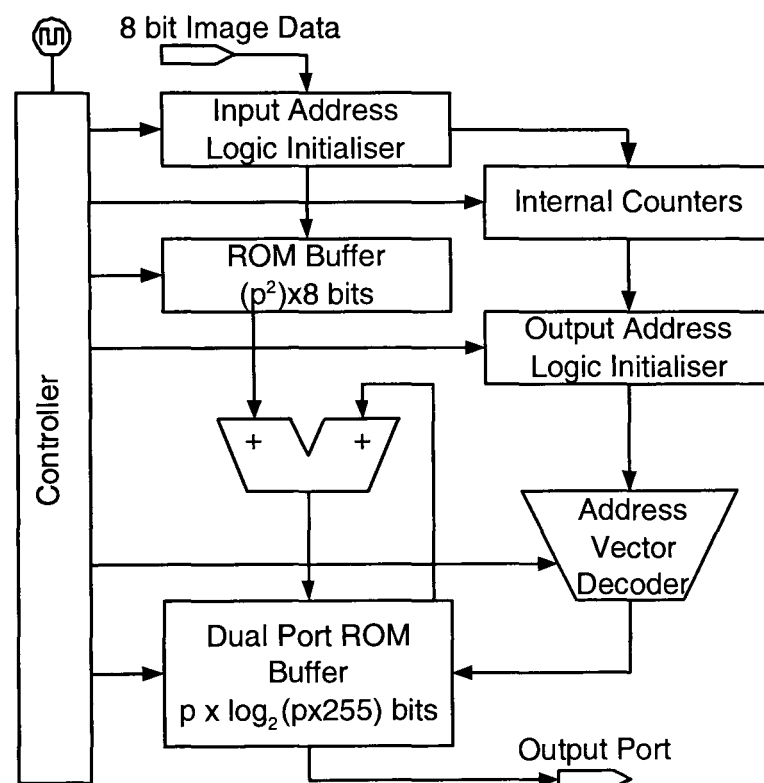


Figure 4.1: Reference architecture for the FRAT

Parallel Cyclic FRAT Architecture

The second FRAT architecture is a serial I/O architecture with a parallel core that computes all $(p + 1)$ FRAT vectors simultaneously. The standard pseudocode is not used as a basis for developing the design. Instead, a novel systolic based dereferencing technique is used to compute the FRAT coefficients. The input and core section of the design are completely pipelined. The output section cannot be pipelined because the FRAT is an over-complete transform that yields $p(p+1)$ output points for p^2 input points. For maintaining design simplicity and due to power considerations, a non-pipelined serial data flow in the output section is adopted. There is no input buffer and each input signal is processed in a single clock pulse.

Totally, there are $(p + 1)$ independent dual port RAM buffers at the output section, one for each FRAT vector.

The architecture uses an array of registers that store the address dereferencing values for each FRAT vector. By re-mapping the FRAT pseudocode in the order of input signals, it is observed that these register values exhibit a regular shifting sequence that can be easily implemented in a systolic-like fashion. A systolic array is used to store the address dereferencing values rather than multiplexer or counter chains that have been used in previous designs. Multiplexors and arrays utilise deep logic and also lie in the critical path of the design. This naturally reduces the maximum frequency obtained. The Radon transform uses modulo operations (logic intensive if used directly) and previous designs have utilised multiplexors and counter chains. Counter chains cannot be readily pipelined in the case of the Radon transform as it has a redundant form. To reduce logic depth, counter chains can be cascaded. This, however results in wastage of clock edges, where the rest of the logic is idle. The only other alternative is to parallelise the counter cascades, which comes at the expense of latency. The systolic array uses very shallow logic in comparison to counter cascades or multiplexors and hence results in greatly improved frequency metrics. The novelty of this architecture lies in the fact that a different approach has been taken to implement the FRAT efficiently, instead of just parallelising the standard FRAT pseudocode presented in [35]. Additionally, instead of using arrays to store the output coefficients, the design choice of using dual ported RAMs helps in reduction of area. Finally, RAMs are simpler to implement in hardware when compared to arrays; and take specific advantages of the hardware resources available on the FPGA. This fact also contributes to improved frequency performance when compared to previous designs.

The first register in each column is used as the address dereferencer for each output RAM buffer. At the end of p^2 clock cycles, the output buffer contains the image block in the transform domain. They are then ejected in a serial fashion from the output port. The block diagram description of the second FRAT architecture with parallel core is shown in Fig. 4.2. The systolic array used for RAM address generation for the case $p = 7$ is shown in Fig. 4.3. Downshift by one position is performed every

clock cycle and upshift by a predetermined, hard-coded position is performed every p clock cycles.

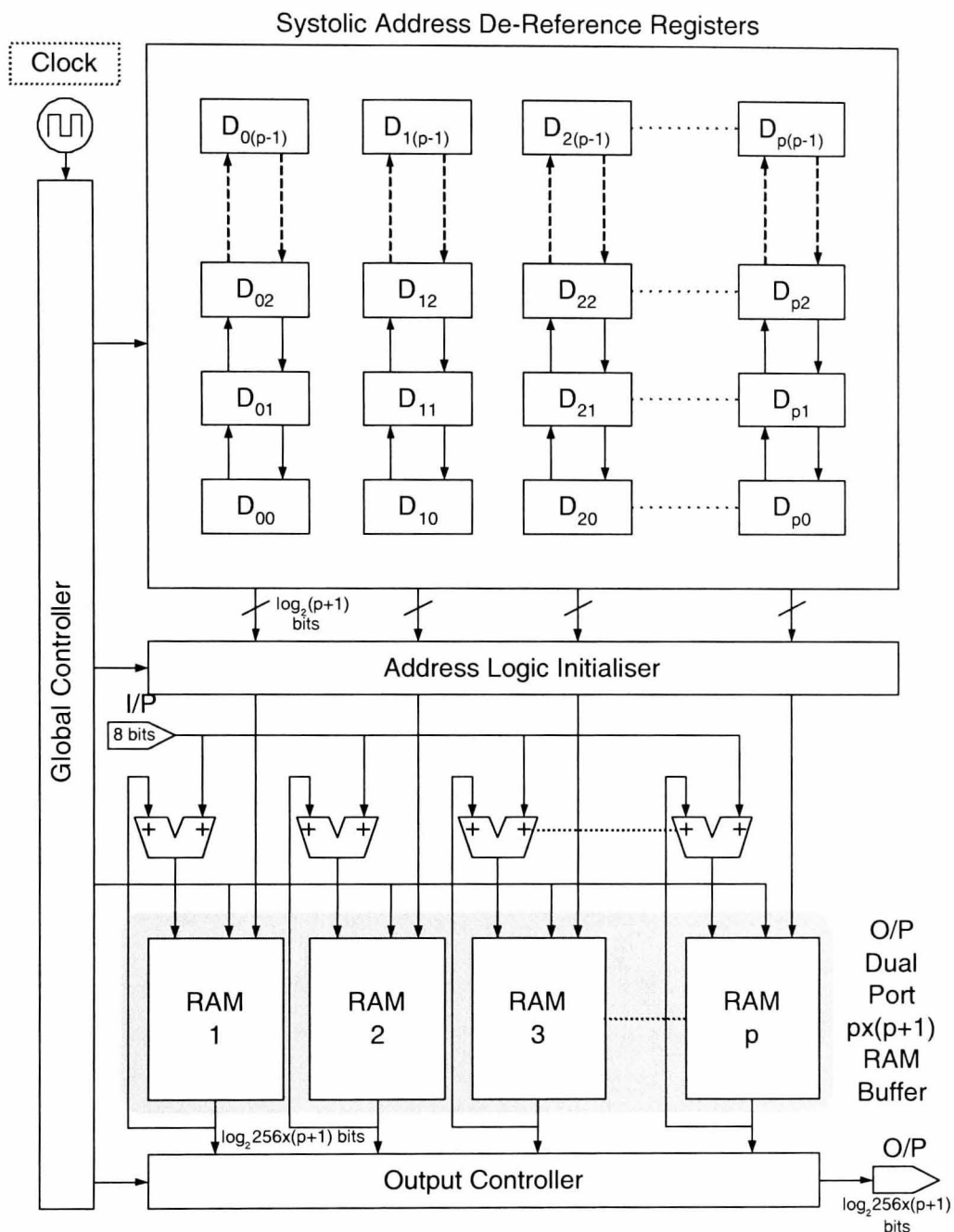


Figure 4.2: FRAT architecture with parallel core

Comparison with Existing Architectures

Design parameters such as TC, AC and I/O type of the proposed designs and other existing architectures are presented in Table 4.1.

It can be seen from Table 4.1 that the proposed reference architecture has a competitive area complexity figure while the proposed parallel architecture has the least

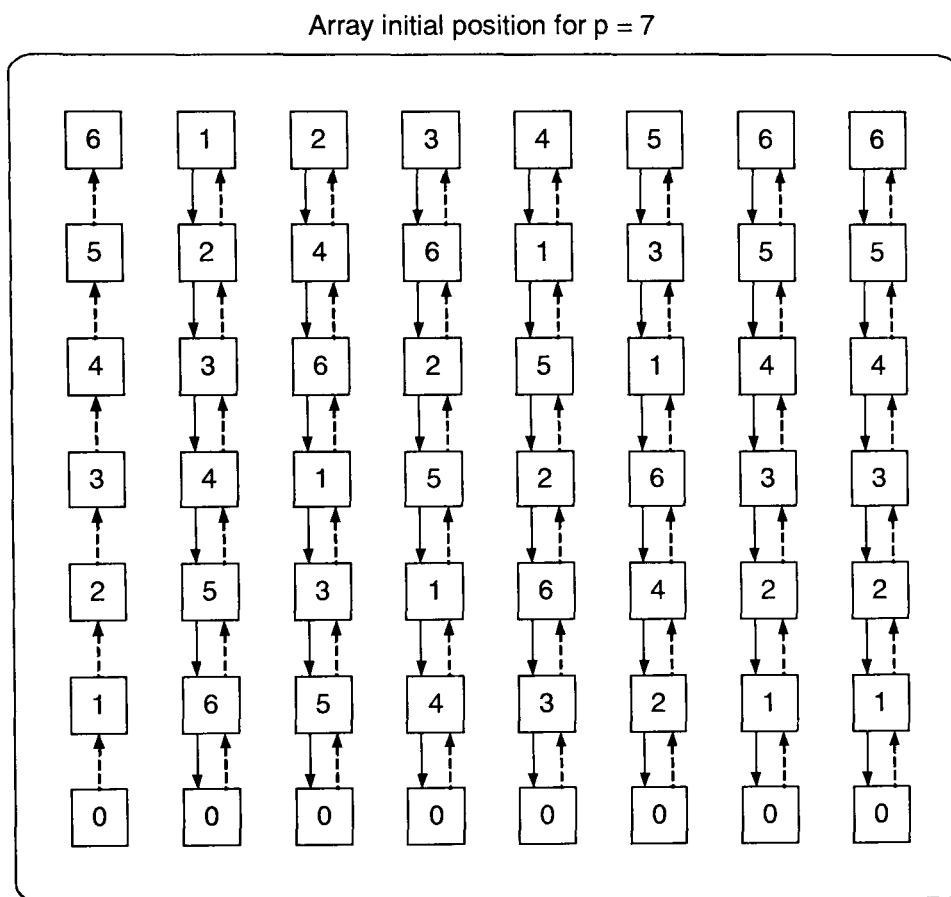


Figure 4.3: Initial array register contents for the case $p = 7$. Downshift is carried out every clock cycle and is indicated by solid arrow. Upshift is carried out every p clock cycles and is indicated by dashed arrow.

Table 4.1: Comparison with existing architectures

	TC	AC	I/O
Proposed Reference	$O(p^2(p + 1))$	$O(p(p + 1))$	Serial
Proposed Cyclic	$O(p^2)$	$O(2p^2)$	Parallel
[37] Generic	$O(p^2)$	NA	Parallel
[37] Serial	$O(p^2(p + 1))$	$O(2p^2)$	Serial
[34] Reference	$O(p^2(p + 1))$	$O(2p^2)$	Serial
[34] Memoryless	$O(p(p + 1))$	NA	Parallel
[21] Reference	$O(p^2(p + 1))$	$O(2p^2 + p)$	Serial

time complexity compared to other parallel cores. It is also worth mentioning that the type of I/O used also has an impact on the I/O power of the design. Serial architectures in general consume lesser I/O power when compared to parallel architectures. However, this cannot be readily assumed for energy measures as well, since the energy consumed is not a function of frequency, but rather depends on the throughput efficiency of the design.

4.2.3 FPGA Implementation

In order to verify the performance of the proposed architectures for FRAT, the design has been prototyped on the Celoxica RC1000 [116] [Appendix B]. In order to provide fair comparison with existing work and to demonstrate the fact that the proposed FLPAM methodology (described in Chapter 6) scales well with different FPGA platforms, the IP cores developed are resynthesised and implemented without any architectural modifications for the Xilinx XC2V8000 (Virtex-II) [36] and XC4VLX200 (Virtex-4) [10] platforms.

Image Data

The FRAT domain visualisation of commonly used images is shown in Fig. 4.4. It can be observed that the averaging effect of the FRAT and the block artifacts of the image in the transform domain become clearly visible as p increases. It is worth mentioning that the FBP is a mathematically perfect inversion for the FRAT and Peak Signal to Noise Ratio (PSNR) depends only on the accuracy required. The truncation or rounding step that follows the FRAT determines the PSNR figures. FRAT is usually used as a sub-block in other transforms such as FRIT and curvelets and 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. Also, the level of precision required is a choice best left to the end user. Hence the IP cores that have been developed operate at full precision. However, to illustrate the effect of bit-width limitations on PSNR, reconstruction has been carried out on standard images stored at 8 Bits Per Pixel (BPP) in the FRAT domain. The PSNR values of the reconstructed images have been presented in Table 4.2.

Table 4.2: PSNR of images reconstructed from 8 BPP Radon domain standard images

	p=7	p=17	p=31
Lena	47.82	43.56	40.49
Peppers	47.89	43.62	40.92
Baboon	47.86	43.83	41.09

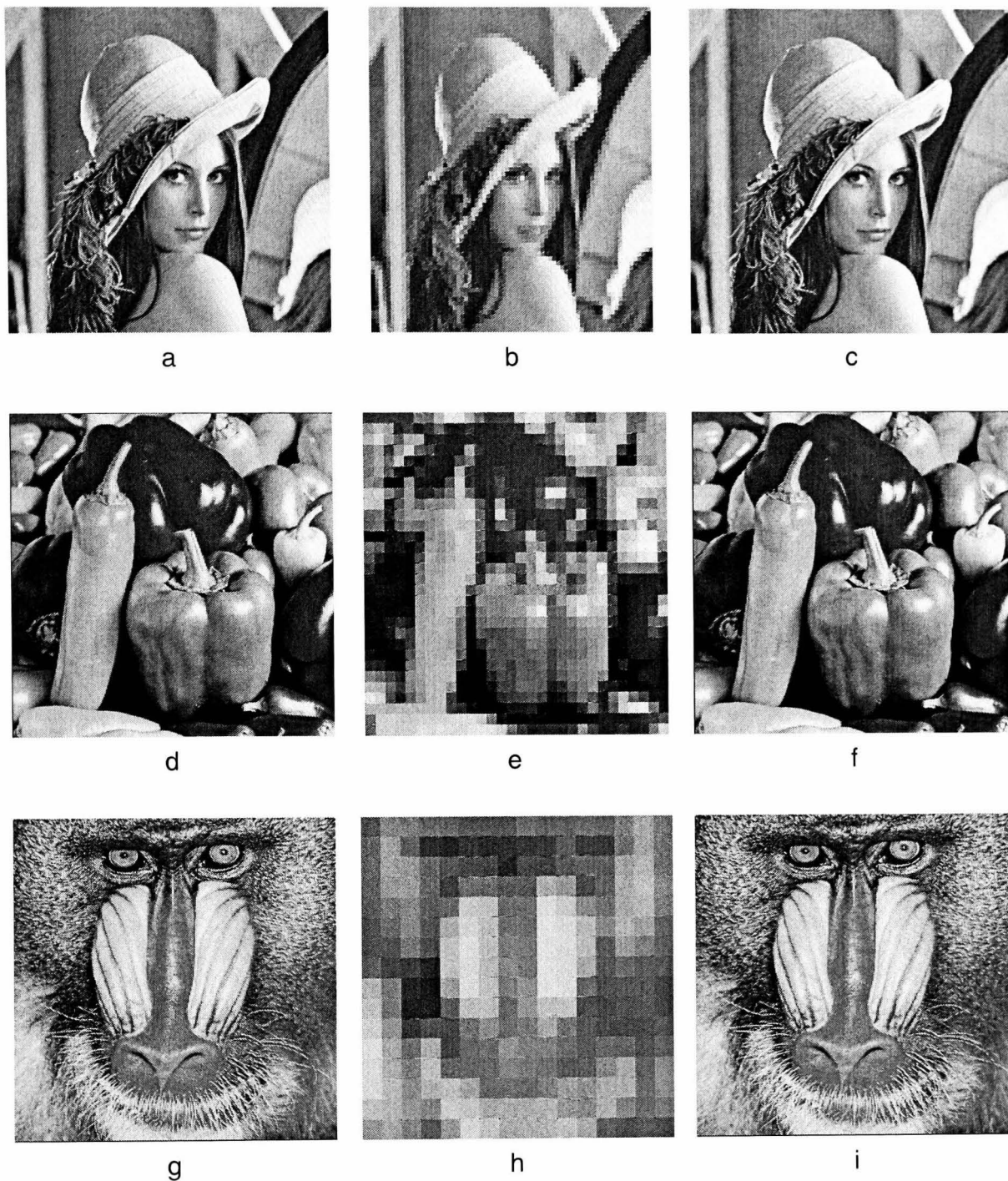


Figure 4.4: Spatial domain and transformed images (a) Spatial Domain Lena (b) FRAT domain, $p = 7$ (c) Reconstructed Image (d) Spatial Domain Peppers (e) FRAT domain, $p = 17$ (f) Reconstructed Image (g) Spatial Domain Baboon (h) FRAT domain, $p = 31$ (i) Reconstructed Image

Performance Metrics

Implementation results in terms of various performance metrics like area occupied, maximum frequency, effective latency and throughput rate for both FRAT architectures have been presented in Table 4.3.

Comparison of area metrics of the FPGA implementation of the proposed architec-

Table 4.3: Performance metrics for the Virtex-E platform

p	Reference Architecture			Parallel Architecture		
	7	17	31	7	17	31
Slices	95	241	498	215	1008	1793
Freq. (MHz)	62.58	42.21	26.91	79.97	40.32	35.06
Cycles	497	5797	32705	105	595	1953
Throughput (M Pix/sec)	6.17	2.10	0.79	37.32	19.56	17.25

Table 4.4: FPGA implementation - comparison of area (number of slices) with existing architectures

Type	Platform	Design	Block size (p)		
			7	17	31
Reference	Virtex-E	Proposed	95	241	498
		[37]	NA	828	NA
		[146]	345	969	NA
	Virtex-II	Proposed	97	229	427
		[34]	159	NA	NA
		[21] Arch 1	198	636	1118
		[21] Arch 2	131	300	500
Parallel	Virtex-E	Proposed	215	1008	1793
		[146]	1212	3398	NA
	Virtex-II	Proposed	245	824	1462
		[34]	558	NA	NA

NA: Data has not been provided

tures for the FRAT with existing architectures is presented in Table 4.4. It can be clearly seen that the proposed reference architecture as well as the parallel architecture occupy much lesser area than other comparable architectures. Comparison of performance metrics of the proposed FRAT architectures with existing work is presented in Table 4.5.

It is worth mentioning that although the proposed architectures have been implemented on the Virtex-E family of FPGA's, synthesis has also been carried out for the Virtex-II FPGA series to enable fair comparison with existing work. It can be seen from Table 4.5 that the parallel architecture implemented on Virtex-E outperforms other comparable architectures implemented on Virtex-II, signifying outstanding improvement in throughput for the same platform. The reference architecture

Table 4.5: Comparison of performance with existing architectures for the case $p = 7$

Type	Platform	Design	F	T	T/A
Reference	Virtex-E	Proposed	62.58	6.17	64.95
		[146]	69.00	6.90	19.71
	Virtex-II	Proposed	92.46	9.21	94.95
		[34]	100.13	9.87	62.08
		[21] Arch 1	112.87	11.13	56.21
		[21] Arch 2	67.3	6.64	50.96
Parallel	Virtex-E	Proposed	79.97	37.32	173.58
		[146]	59.76	27.89	23.01
	Virtex-II	Proposed	96.46	45.01	183.71
		[34]	81.92	38.23	68.51

where F is the maximum frequency in MHz, T is the effective throughput in Mega Pixels/second and A is the area occupied in slices in the term T/A

proposed has slightly lower maximum frequency than other existing comparable architectures. However, it must be pointed out that the corresponding area occupied by the proposed architecture is nearly half that of comparable architectures. The throughput/area column in Table 4.5 yields significant insight into the efficiency of the architectures that have compared. It can be seen that the proposed parallel architecture clearly outperforms all other architectures. The most efficient architectures are those that provide the best performance metrics for the least core footprint. This is an important consideration, as it has a direct relationship with the power efficiency of the core.

Energy Analysis and Observations

Table 4.6: EPP (nJ) for the proposed parallel FRAT core on the Virtex-E FPGA platform

p	Reference Architecture			Parallel Architecture		
	7	17	31	7	17	31
Clock	1.57	5.68	66	.49	1.94	3.34
Signal	6.84	36.32	146.82	2.13	10.02	19.26
Logic	12.32	61.67	150.89	6.36	30.54	54.44
Input	0.26	0.41	0.61	0.14	0.13	0.13
Output	6.14	6.72	6.49	5.76	6.54	6.46

The equation for EPP ¹ is obtained by rationalising energy per operation with respect to the number of pixels processed. Based on the power dissipation data available from XPower, the corresponding energy metrics obtained are presented in Table 4.6.

Since power dissipation directly proportional to frequency f for all components i.e. clock, signal, logic, input and output. Consequently, the corresponding EPP measures are independent of frequency and are constant for a given block size p . Based on the data in Table 4.6 the EPF ² data is graphically presented in Fig. 4.5

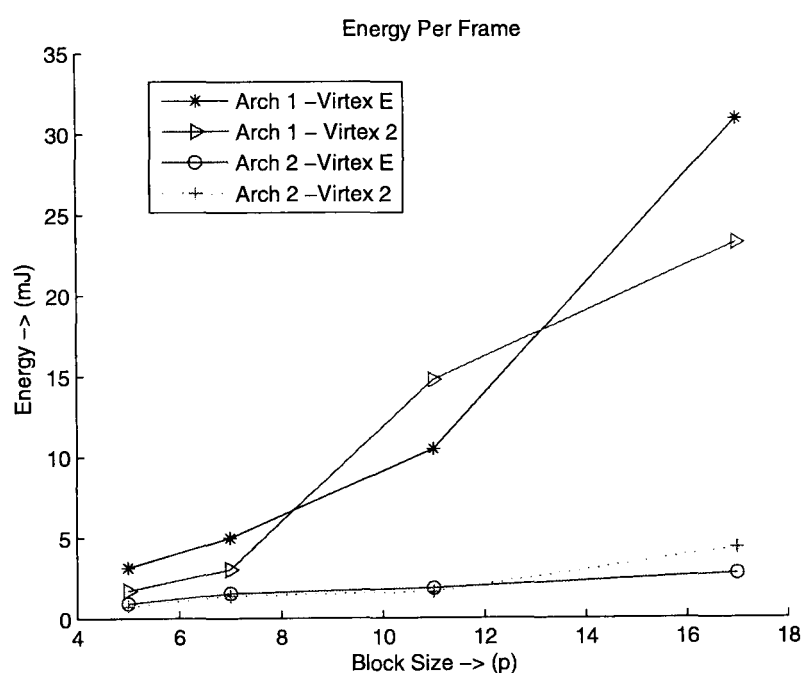


Figure 4.5: Comparison of EPF for the “reference” and proposed parallel FRAT architectures

Although the area occupied by the proposed parallel FRAT architecture is higher when compared to the reference architecture, the reduction of effective frequency by a factor of p for maintaining the same throughput yields significant reduction in energy dissipation. This can be clearly observed in Fig. 4.5. For typical value of $p = 7$, saving of upto 50% is achieved, and this increases to 80% for $p = 17$.

¹Refer to Appendix D for details

²Refer to Appendix D for details

Chip Level Details

Careful manual place and route of critical nets and manual pin assignment for the designs has been performed using Xilinx PACE and Xilinx Floorplanner [11]. This process yields compact and optimised design with short nets, and serves two important purposes. Firstly, short nets have lesser propagation delay and upto 25% gains in maximum frequency have been achieved. Second, short nets have lesser parasitic capacitance and DC load and therefore dissipate lesser power than long nets. Manual pin assignment also enables us to locate the I/O pads close to the design area, further aiding the above two criteria. Sample post place and route chip diagrams are shown in Fig. 4.6.

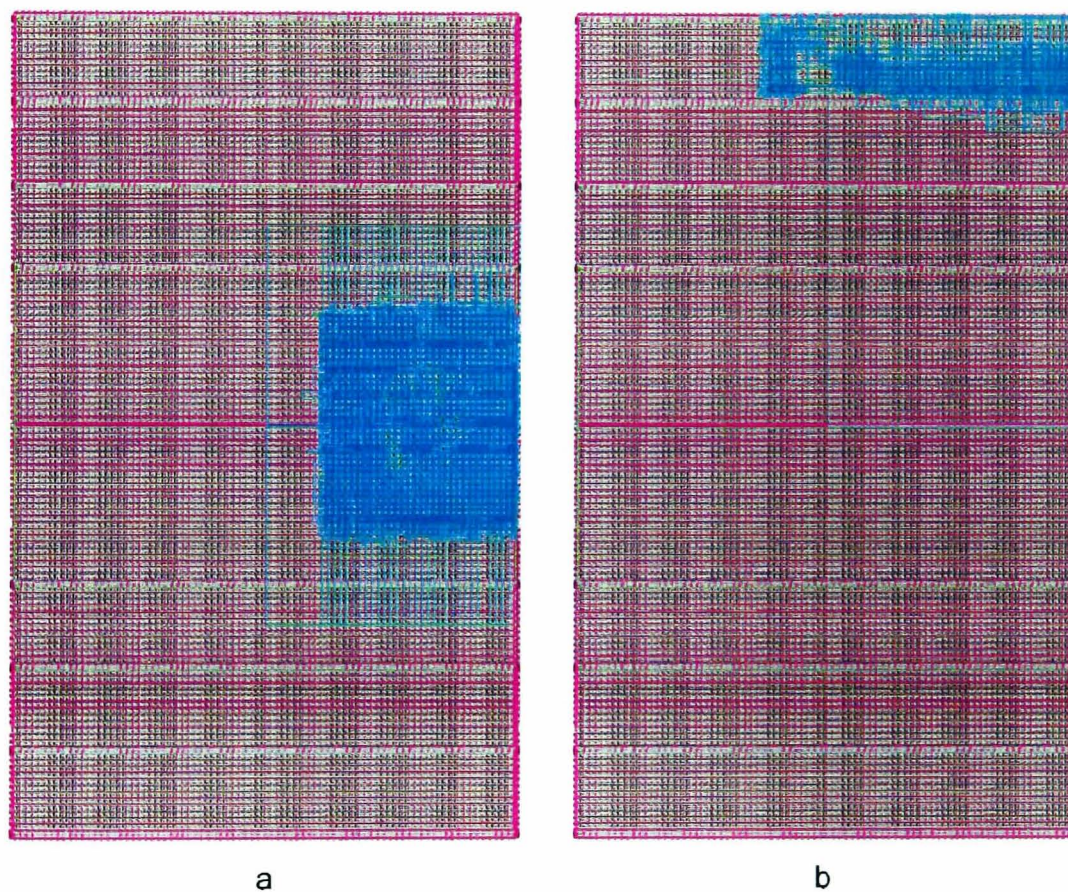


Figure 4.6: Chip diagrams for $p = 31$ (a) Parallel architecture (b) Reference architecture; (Virtex-E XCV2000E FPGA)

4.3 Efficient VLSI Architecture for the Finite Ridgelet Transform

Software implementation of the FRIT is usually based upon the standard pseudocode for the FRAT followed by a suitable DWT block. The computational complexity of the FRAT and DWT blocks in the FRIT make hardware acceleration essential for fast computing and achieving real time processing. A straightforward translation of this pseudocode for hardware implementation is sub-optimal, as it does not exploit the parallelism capabilities of dedicated processors. Non dyadic transform lengths and modulo operations in the FRAT sub-block necessitate algorithmic transformations to yield efficient hardware implementation. An elegant solution to this problem in the form of systolic address dereferencing and suitable re-ordering of the FRAT counters in the control circuitry is presented in this paper. The DWT block is based on the Haar Wavelet, which provides the best energy compaction (minimum entropy) when compared to other higher order wavelets. It is the aim of this work to present a novel and platform independent VLSI architecture for FRIT. Efficient FPGA and ASIC synthesis of the proposed architecture is also performed.

4.3.1 Mathematical Background of the Finite Ridgelet Transform

An introduction into the FRAT has been provided in the previous section. Additionally, a comprehensive mathematical review about the FRAT has also been provided in Appendix C. The final expression for the FRAT is reprinted here as a refresher: The Radon projection of the function f on the finite grid Z_p^2 is then given by:

$$r_k[l] = FRAT_f(k, l) = \frac{1}{\sqrt{p}} \left(\sum_{(i,j) \in L_{k,l}} f[i, j] \right) \quad (4.4)$$

Haar DWT

Popular discrete wavelets include Haar, Daubechies, Daubechies Biorthogonal, Coiflet, Symmlet etc. The Haar wavelet is one of the simplest wavelet transforms. Although

its simplicity makes it unsuitable for a number of applications, particularly due to the fact that it is not continuous and hence not differentiable, it is highly suitable for constructing the FRIT due to the following reasons. Firstly, the Haar wavelet is very simple in structure and is less expensive to compute compared to other wavelets. Secondly, due to the averaging characteristics of the FRIT, it has been observed that the Haar wavelets display superior performance in energy compaction when compared to other wavelets. Entropy measures for common test images are presented in Table 4.7 to substantiate this claim.

The mother scaling function for the Haar is given by:

$$\psi(x) = \begin{cases} 1 & 0 \leq x < 1/2 \\ -1 & 1/2 \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \quad (4.5)$$

Fast computation of the Haar transform can be performed by means of lifting using the Haar matrix. The smallest matrix of size 2x2 is given by:

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (4.6)$$

Building the FRIT from FRAT and DWT

The continuous ridgelet transform [33] of a bivariate function $f(x)$ is given by:

$$RT = \int_{R^2} \psi_{a,b,\theta}(x) f(x) dx \quad (4.7)$$

However, the use of digital images necessitates the development of suitable variations of the ridgelets to deal with images in the digital domain. An orthonormal, invertible and discrete form of the ridgelet, called finite ridgelet transform was first proposed in [31]. THE FRIT is obtained by performing the DWT on each FRAT projection sequence with fixed value of k . This process is pictorially represented in Fig. 4.7.

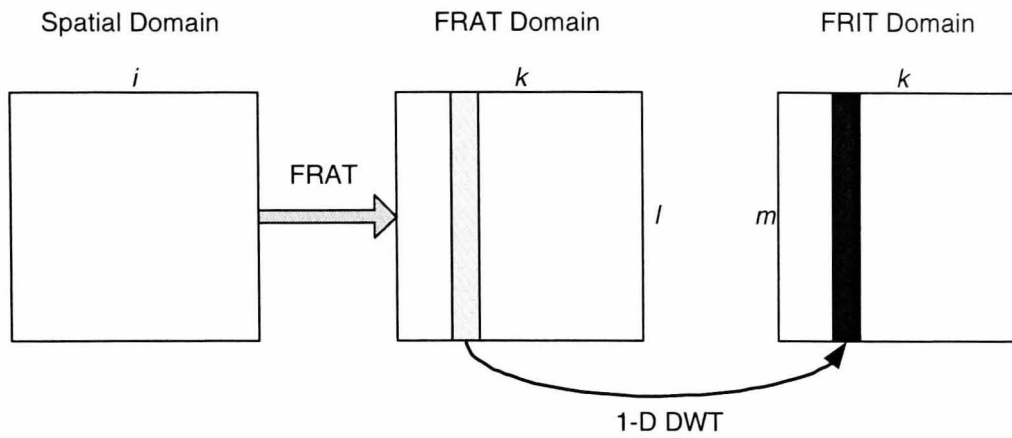


Figure 4.7: Finite Ridgelet transform obtained by performing DWT on the FRAT vectors

Mathematically, this is given by:

$$FRIT_{i,i+1}(k, l) = [FRAT_{i,i+1}(k, l) \times H_2]_2 \forall i = 1, 3, \dots, p \quad (4.8)$$

The finite and discretized variant of curvelets can be obtained by repeated application of the FRIT. The directional attributes of these higher dimensional generalisations of wavelets make them ideal for a number of applications such as alternate image representation, compression, denoising, etc.

4.3.2 Proposed Architectures for FRIT - Design and Evaluation

In the following subsection, the design flow used to generate the optimised architecture for FRIT is presented. Next, the architectures designed for the implementation of FRIT are described, followed by a tabular comparison of the various parameters with other existing architectures in place. The FRIT core consists of two basic sub-blocks: the novel FRAT block described in Subsection 4.2.2 and the Haar block. The first register in each column is used as the address dereferencer for each output RAM buffer. At the end of p^2 clock cycles, the output buffer contains the image block in the transform domain. They are then ejected in a serial fashion from the output port. The block diagram description of the FRIT architecture with parallel core is shown in Fig. 4.8.

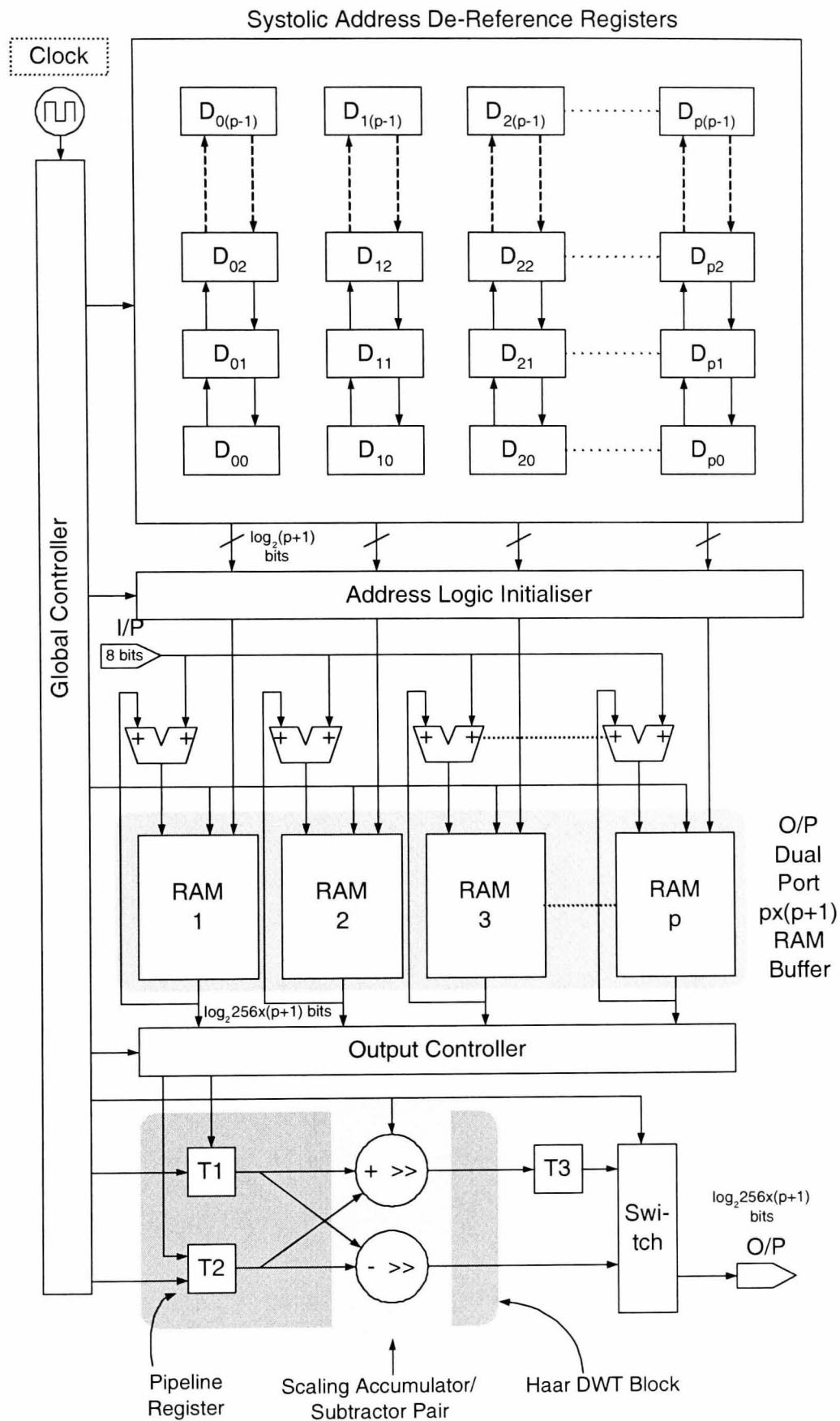


Figure 4.8: FRIT architecture with the FRAT and Haar DWT sub-blocks

Haar DWT Sub-Block

Although Haar is the simplest wavelet, entropy measures indicate that it yields much better performance than other complex wavelets for implementing the FRIT.

It can be observed from Table 4.7 that the FRIT domain images based on the Haar DWT yield the lowest entropy measures, indicating higher compressibility. This fact, coupled with the computationally inexpensive nature of the Haar DWT makes it an ideal choice for the FRIT IP core that has been designed. The DWT sub-block is fully pipelined and is efficiently implemented using accumulate shift and accumulate subtract sub-unit circuits that perform the combined task of computing the sum and differences, as well as scaling the image in the wavelet domain. The output is alternated in serial manner between these two sub-units by means of a switch.

Table 4.7: Entropy measures for Source, FRAT, Wavelet and FRIT domain images using different wavelet filters

	Baboon	Lena	Barbara	Peppers
Source	4.76	3.75	4.17	3.88
FRAT	3.88	2.44	3.00	2.67
Haar	3.73	2.86	3.25	3.07
FRIT	3.21	2.22	2.60	2.41
CDF(2,2)	3.79	2.99	3.42	3.27
FRIT	3.47	2.50	2.86	2.73
CDF(6.6)	4.03	3.58	3.83	3.75
FRIT	3.91	3.35	3.57	3.49
Sym4	3.88	3.34	3.58	3.46
FRIT	3.53	2.92	3.13	3.01

Comparison with Existing Architectures

Design parameters such as TC, AC and I/O type of the proposed designs and other existing architectures are presented in Table 4.8.

Table 4.8: Comparison of design parameters of the Radon block with existing architectures

	TC	AC
Proposed Architecture	$O(p^2)$	$O(2p^2)$
[34] Reference	$O(p^2(p+1))$	$O(2p^2)$
[34] Memoryless	$O(p(p+1))$	NA
[37] Generic	$O(p^2)$	NA
[37] Serial	$O(p^2(p+1))$	$O(2p^2)$
[21] Serial	$O(p^2(p+1))$	$O(2p^2+p)$

It can be seen from Table 4.8 that the proposed reference architecture has a competitive area complexity figure while the proposed parallel architecture has the least time complexity compared to other parallel cores. It is also worth mentioning that the type of I/O used also has an impact on the I/O power of the design. Serial architectures in general consume lesser I/O power when compared to parallel architectures. However, this cannot be readily assumed for energy measures as well, since the energy consumed is not a function of frequency, but rather depends on the throughput efficiency of the design.

4.3.3 Implementation Results and Metrics

Various details of the implementation of the FRIT IP core developed including performance metrics of the core on both the FPGA and ASIC platforms, evaluation of performance in comparison with existing work, chip diagrams and sample images are presented in this section.

In order to verify the performance of the proposed architectures for FRAT, the design has been prototyped on the Celoxica RC1000 [116] [Appendix B]. In order to provide a fair comparison with existing work the IP cores developed are re-synthesised and implemented without any architectural modifications for the Xilinx XC2V8000 (Virtex-II) platform. Synthesis is also carried out on the Spartan 3L [Appendix B] platform to perform a critical comparison of power dissipation on a low-power FPGA platform with an ASIC implementation. The mapping the proposed architecture to an ASIC platform has been performed in conjunction with our collaborators at the HKUST.

Performance Metrics

FPGA and ASIC implementation results in terms of various performance metrics like area occupied, maximum frequency, effective latency and throughput rate for the FRIT IP core have been presented in Table 4.9.

Table 4.9: Performance metrics for the FRIT IP Core on the FPGA and ASIC platforms. Maximum frequency is in MHz and throughput is denoted in Million pel/sec

	$p \rightarrow$	7	17	31
Virtex-E	Slices	312	1176	2064
	Max Freq.	70.6	40.6	35.1
	Throughput	32.9	19.7	17.3
Virtex-II	Slices	336	1235	2152
	Max Freq.	90.4	60.4	48.5
	Throughput	42.2	29.3	23.9
Spartan 3L	Slices	312	1124	2051
	Max Freq.	76.3	43.4	37.1
	Throughput	35.6	21.1	18.2
250 nm ASIC	Area (mm ²)	0.27	1.67	5.02
	Max Freq.	100.0	60.0	50.0
	Throughput	46.7	29.1	24.6

Dynamic Energy Consumption

In order to compare energy metrics of different platforms, the EOP³ values have been graphically presented in Fig. 4.9. It is worth mentioning that although different process technologies used for different platforms preclude straightforward one on one comparison of the numbers, the general trends yield interesting insights.

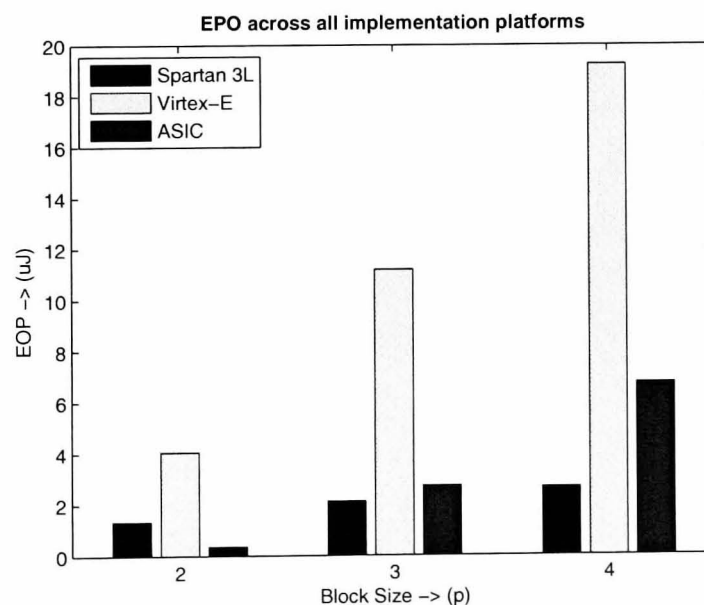


Figure 4.9: Comparison of EOP across different platforms

³Refer to Appendix D for details

The graph clearly indicates that the Spartan 3L platform outperforms the the ASIC implementation by a few percentage points for the case $p = 17$ and by almost a factor of 3 for the case $p = 31$. In all cases, the Virtex-E shows the least competitive power measures. A number of reasons can provide an insight into these results. Firstly, the Virtex-E is based on a 180nm process technology, which results in higher dissipation of static and dynamic power compared to other platforms. On the other hand, the Spartan 3L series is based on 90nm technology. Reduced V_{cc} levels, shrinking process size and a clear focus on making the FPGA fabric more power efficient has resulted in the excellent power and energy figures for this platform. The variation in energy metrics between the ASIC and Spartan 3L platforms for different values of block size p can be attributed to the fact that total power dissipation in FPGA depends on a number of factors in addition to total area and hence does not scale up with area as fast as seen in the case of ASIC. It must be highlighted that for the same process technology, ASIC would undoubtedly present the most power and energy efficient implementation. However, it is also worth mentioning that FPGA vendors are usually early adopters of new process technologies and typically stay one to two generations ahead of their ASIC counterparts. It must be highlighted that the graph indicates dynamic energy dissipation only (does not include quiescent energy dissipation in FPGAs, which is a constant).

FPGA Chip Level Details

For the FPGA implementation, careful manual place and route of critical nets and manual pin assignment for the designs has been performed using Xilinx PACE and Xilinx Floorplanner [11]. Post place and route chip diagram is shown in Fig. 4.10.

ASIC Chip Details

For ASIC implementation, the gate level circuit has been synthesised using SDA. Time constraints were set when performing the synthesis in order to decrease the worst case delay. The final circuit was designed using $0.25\mu m$ CMOS process with 5 metals. The automatic placement and routing were performed using Encounter. A sample chip layout using automatic placement and routing in $0.25\mu m$ process is

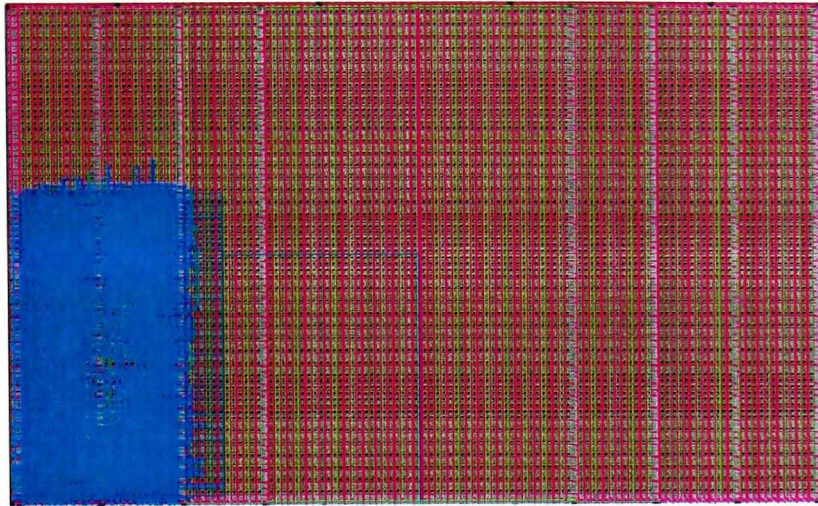


Figure 4.10: Chip diagram for the case $p = 31$ implemented on the Virtex-E XCV2000E FPGA

shown in Fig. 4.11.

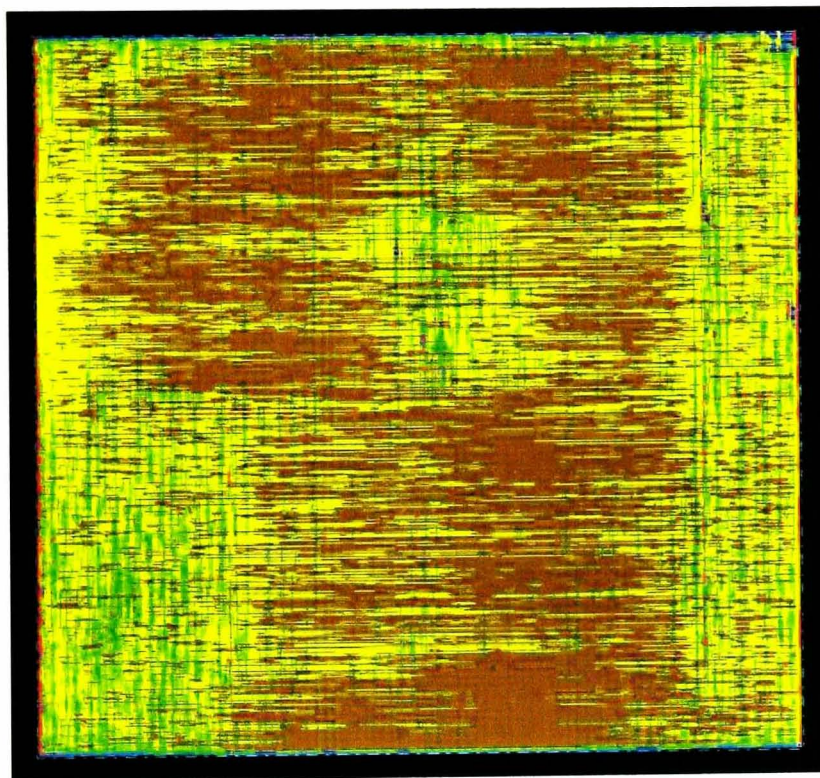


Figure 4.11: Chip layout for ASIC implementation for the case $p = 31$

Comparison with Existing Architectures

Comparison of performance metrics of the proposed architectures with those of existing architectures has been presented in Table 4.10.

It can be clearly seen that the proposed architectures outperform existing imple-

Table 4.10: Comparison of performance metrics with existing architectures in place

Platform	p	Design	A	F	T
Virtex-E	7	Proposed	312	70.59	32.94
	17	Proposed	1176	40.60	19.72
	17	[37] Parallel	NA	33.00	16.03
	17	[37] Reference	828	18.00	0.90
Virtex-II	7	Proposed	336	90.37	42.17
	7	[21] Reference	476	101.00	9.96
	17	Proposed	1235	60.40	29.34
	17	[21] Reference	911	32	1.60

mentations in terms of the maximum throughput achieved. This is because the proposed architecture has a parallel Radon core and thus has the least possible time complexity in its class, as can be seen from Table 4.8.

Image Data

The FRIT domain visualisation of commonly used images is shown in Fig. 4.12. Due to the FBP being a mathematically perfect inverse of the FRAT, PSNR depends only on the accuracy required. The truncation or rounding step that follows the DWT sub-block determines the PSNR figures. The level of precision required is a choice best left to the end user. Hence the IP cores that have been developed operate at full precision.

4.4 FPGA Realisation of FIR Filters by Efficient and Flexible Systolisation using Distributed Arithmetic

FIR digital filters are extensively used due to their key role in various DSP applications [149, 150]. Since the complexity of implementation grows with the filter order and the precision of computation, real-time realisation of these filters with desired level of accuracy is a challenging task. Systolic designs represent an attractive architectural paradigm for efficient hardware implementation of computation-intensive

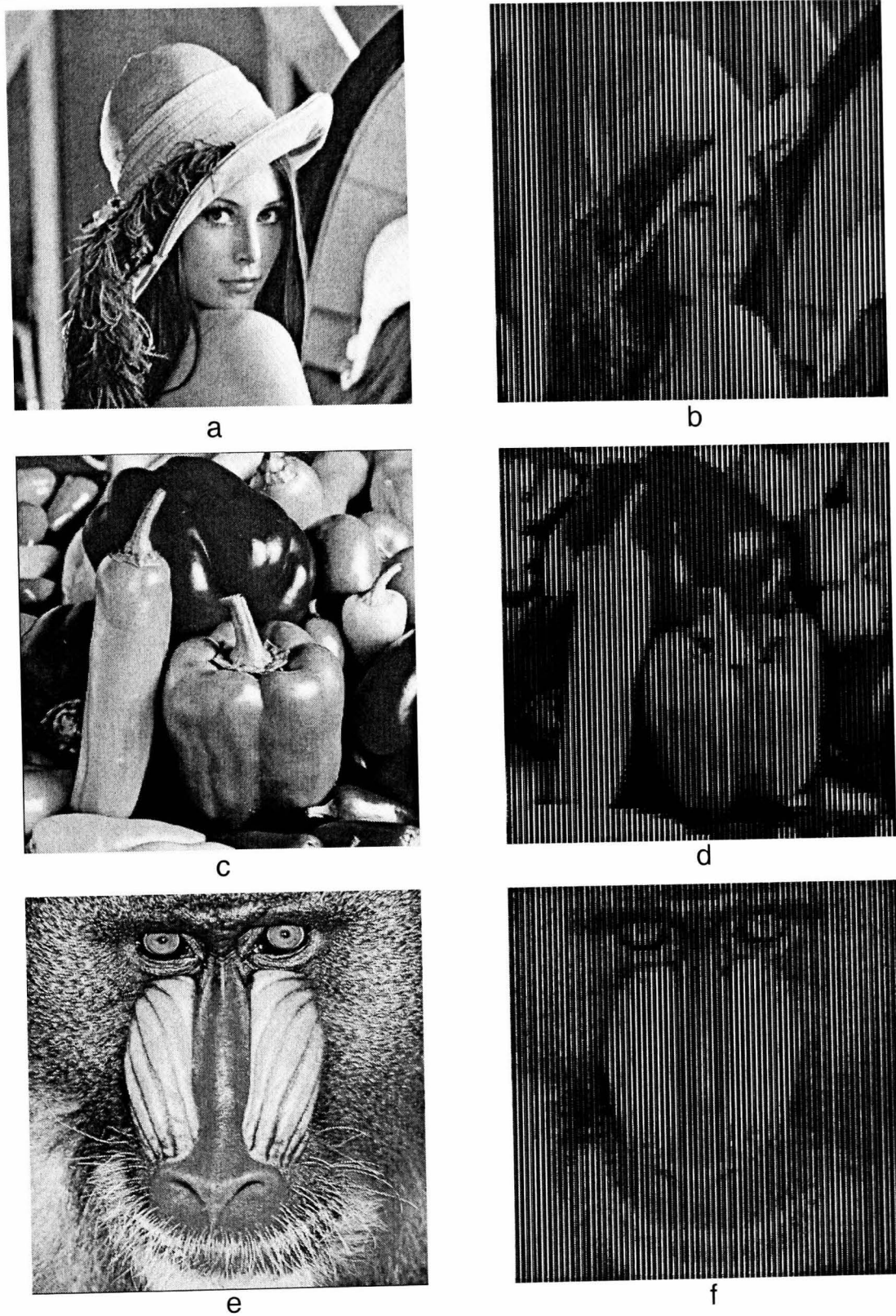


Figure 4.12: Spatial domain and transformed images (a) Spatial domain Lena (b) FRIT domain, $p = 7$ (c) Spatial domain Peppers (d) Reconstructed image (e) Spatial domain Baboon (f) FRIT domain, $p = 31$

DSP applications, being supported by the features like simplicity, regularity and modularity of structure. Additionally, they also possess significant potential to yield high-throughput rate by exploiting high-level of concurrency using pipelining

or parallel processing or both [151]. In this work-package, the work of [152] has been extended further to obtain an optimal area-delay-power-efficient implementation of FIR filter in FPGA platform.

4.4.1 Formulation of the Proposed Algorithm

A brief outline of conventional DA approach for inner product, followed thereafter by the proposed proposed decomposition scheme for DA-based systolisation of FIR filters has been presented in this subsection.

Conventional DA Approach for Inner-Product Computation

By applying DA principles [Appendix C], the inner-product of two N -point vectors A and B can be expanded as follows:

$$C = - \sum_{k=0}^{N-1} A_k \cdot b_{k0} + \sum_{k=0}^{N-1} A_k \cdot \left[\sum_{l=1}^{L-1} b_{kl} \cdot 2^{-l} \right] \quad (4.9)$$

where A is constant vector, while B may change from time to time and L is the word-length.

To convert the conventional sum-of-products form of inner-product of into a distributed form, the order of summations over the indices k and l in the second term of Eq. 4.9 can be interchanged to have:

$$C = - \sum_{k=0}^{N-1} A_k \cdot b_{k0} + \sum_{l=1}^{L-1} 2^{-l} \cdot \left[\sum_{k=0}^{N-1} A_k \cdot b_{kl} \right] \quad (4.10)$$

Without loss of generality, for simplicity of discussion, we may assume the signal samples to be unsigned words of size L , although the proposed algorithm can be used for 2's complement coding and offset binary coding also. The inner-product given by Eq. 4.10 then can be expressed in a simpler form:

$$C = \sum_{l=0}^{L-1} 2^{-l} \cdot C_l \quad (4.11)$$

where

$$C_l = \sum_{k=0}^{N-1} A_k \cdot b_{kl} \quad (4.12)$$

Since vector A is assumed to be constant and each element of the N -point bit-sequence $\{b_{kl} \text{ for } 0 \leq k \leq N - 1\}$ can either be zero or one, any of the partial sum C_l for $l = 0, 1, \dots, L - 1$, can have 2^N possible values. All the 2^N possible values of C_l can, therefore, be pre-computed and stored in a ROM, such that while computing the inner-product the partial sums C_l can be read out from the ROM using the bit-sequence $\{b_{kl} \text{ for } 0 \leq k \leq N - 1\}$ as address-bits. The inner-product can, therefore, be calculated according to Eq. 4.11, by L cycles of shift-accumulation followed by ROM-read operations corresponding L number of bit-sequences $\{b_{kl}\}$ for $0 \leq l \leq L - 1$.

Proposed Decomposition Scheme for DA-based Implementation of FIR Filter

The output of an FIR filter of order N can be computed as an inner-product of the impulse response vector $\{h(k), \text{ for } k = 0, 1, \dots, N - 1\}$ and an input vector $\{s_n(k), \text{ for } k = 0, 1, \dots, N - 1\}$, given by:

$$y(n) = \sum_{k=0}^{N-1} h(k) \cdot s_n(k) \quad (4.13)$$

where $s_n(k) = x(n - k)$ and $x(n)$ is the current input sample. $\{h(k)\}$ is a fixed sequence, while the input sequence $\{s_n(k)\}$ changes in every sampling instant. $\{s_n(k)\}$ is derived from serially-shifted input samples using a window of size N , such that it receives a fresh input sample and leaves its oldest sample. Comparing Eq. 4.13 with the original DA expression, the filter output can be computed according to Eq. 4.11 as follows:

$$cy(n) = \sum_{l=0}^{L-1} 2^{-l} \cdot C_l \quad (4.14)$$

where:

$$C_l = \sum_{k=0}^{N-1} h(k) \cdot (s_n(k))_l \quad (4.15)$$

$(s_n(k))_l$ for $l = 0, 1, \dots, L - 1$, being the l^{th} bit of $s_n(k)$.

Eq. 4.14 can be directly used for straight-forward DA-based implementation of FIR filter using a ROM containing of 2^N possible values of C_l . For large values of N , however, the ROM size becomes too large so also the ROM access time also becomes large. The straight-forward DA implementation is, therefore, not suitable for large filter orders.

When N is a composite number given by $N = PM$, (P and M may be any two positive integers) one can map the index k into $(m + pM)$ for $m = 0, 1, \dots, M - 1$ and $p = 0, 1, \dots, P - 1$ so as to express Eq. 4.14 in following form:

$$C_l y(n) = \sum_{l=0}^{L-1} 2^l \cdot \left(\sum_{p=0}^{P-1} (S_n)_{l,p} \right),$$

where

$$(S_n)_{l,p} = \sum_{i=0}^{M-1} h(m + pM) \cdot (s_n(m + pM))_l$$

for $l = 0, 1, \dots, L - 1$ and $p = 0, 1, \dots, P - 1$.

For any given sequence of impulse response $\{h(k)\}$, the 2^M possible values of $(S_n)_{l,p}$ corresponding to the 2^M permutations of M -point bit-sequence $\{(s_n(m + pM))_l, \text{ for } m = 0, 1, \dots, M - 1\}$ for $l = 0, 1, \dots, L - 1$ may be stored in an LUT of 2^M words. These values of $(S_n)_{l,p}$ can be read out when the bit-sequence is fed to the ROM as address. Eq. 4.16 may, thus, be written in term of memory-read operations as:

$$y(n) = \sum_{l=0}^{L-1} 2^l \left(\sum_{p=0}^{P-1} \mathcal{F}(b_n)_{l,p} \right), \quad (4.16)$$

where $\mathcal{F}((b_n)_{l,p}) = (S_n)_{l,p}$ and $(b_n)_{l,p} =$

$$[(s_n(pM))_l \ (s_n(1 + pM))_l \ \dots \ (s_n(M - 1 + pM))_l],$$

for $0 \leq l \leq L - 1$ and $0 \leq p \leq P - 1$.

The bit-vector $(b_n)_{l,p}$ is used as address word for the look-up-table and \mathcal{F} is the memory-read operation.

4.4.2 The Proposed Structures

The proposed DA-based 1- and 2-D systolic arrays for FIR filters presented in this subsection have been derived from from Dependence Graphs (DG).

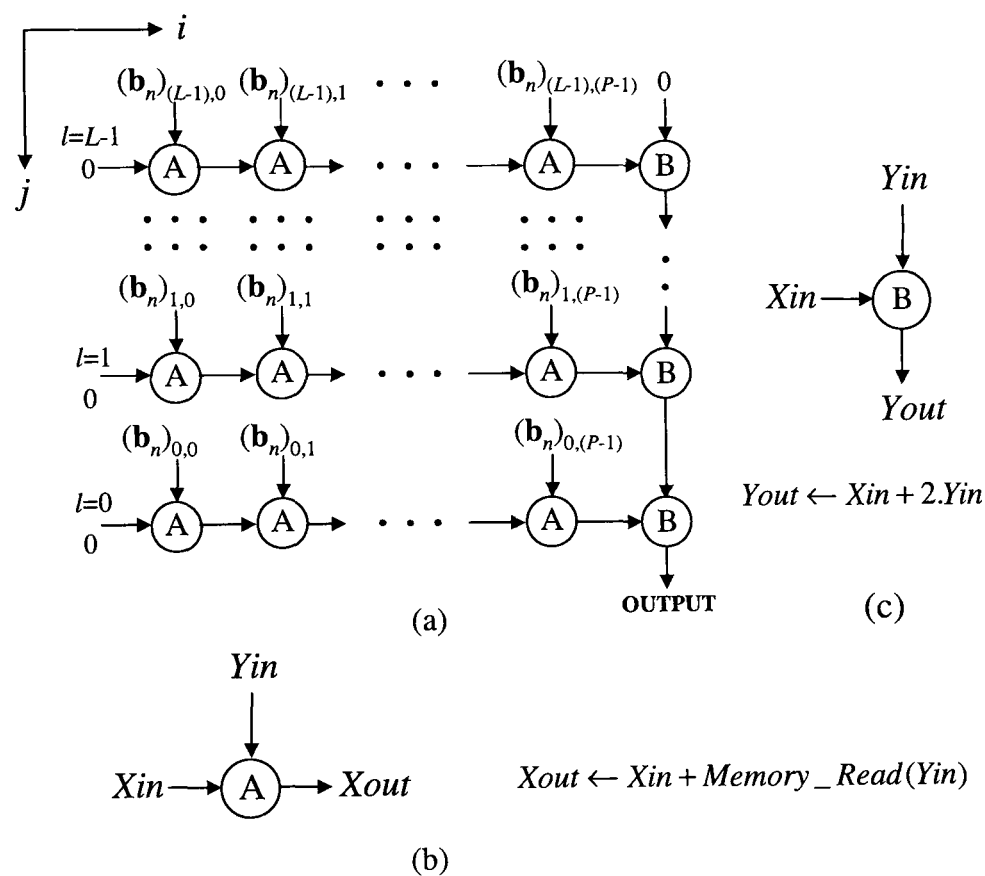


Figure 4.13: The DG for DA-based implementation of FIR filter. (a) The DG. (b) Function of node A. (c) Function of node B.

Proposed 1-D Systolic Array for FIR Filters

The DG for computation of FIR filter output according to Eq. 4.16 is shown in Fig. 4.13. It consists of L rows, where each row consists of P number of node-A and one boundary node-B. The functions of node-A and node-B are depicted in Figs. 1(b) and 1(c), respectively. A bit-vector $(b_n)_{l,p}$ consisting of a sequence of M bits [derived

from the l^{th} bit of the elements of the input sequence as given in Eq. 4.16] is fed to the node-A on $(l + 1)^{th}$ row and $(p + 1)^{th}$ column. The node uses the sequence of M input bits of the input bit-vector as address for an LUT and reads the content stored at the location specified by the address. The value read from the LUT is then added with the input available from its left and the sum is passed to the node on its right. Node-B performs a shift-add operation such that it makes a left-shift of the bits of the input available from the top, then adds the input available from the left to the left-shifted value and passes the result down to its adjacent node. The DG can be projected vertically along the projection direction $[0 \ 1]^T$ with default schedule [4] to derive a linear array consisting of P number of PEs and an output-cell as shown in Fig. 4.14.

The input sequence $\{x(n)\}$ is fed to a serial-in parallel-out input-register, where content of the register is serially-right-shifted by one position and transferred in parallel to the bit-serial word-parallel converter in every L cycles. The bits of vector $(b_n)_{l,p}$, are derived from the bit-serial word-parallel converter and fed to the $(p + 1)^{th}$ PE [for $p = 0, 1, \dots, P - 1$] in Most Significant Bits (MSBs) to LSBs order in each cycle period such that $(L - 1)^{th}$ bits of input values are fed to the PE at first and the zeroth bits are fed at the end. Besides, input to each PE is staggered by one cycle-period with respect to the preceding PE to meet the causality requirement. The function of the PEs is described in Fig. 4.14(b). Each PE consists of a ROM of 2^M words. During a cycle-period (time-step) each PE reads the content on its ROM at the location specified by the input bit-vector. The value read from the ROM is then added to the input available to the PE from its left. During every-cycle period, the sum is then transferred as output to its right. Function of the output-cell is shown in Fig. 4.14(c). Each output-cell contains a shift-register and an adder. During a cycle period it shifts the content of its register left by one position and then adds the available input to the recently shifted content in its register. After L cycles it delivers a desired filter output. The structure will yield its first filter output $(L + P)$ cycles after the first input is fed to the first PE, while the successive output becomes available in every L cycles. For high throughput applications one may, however, have a structure with N number of 1-D arrays which would yield N

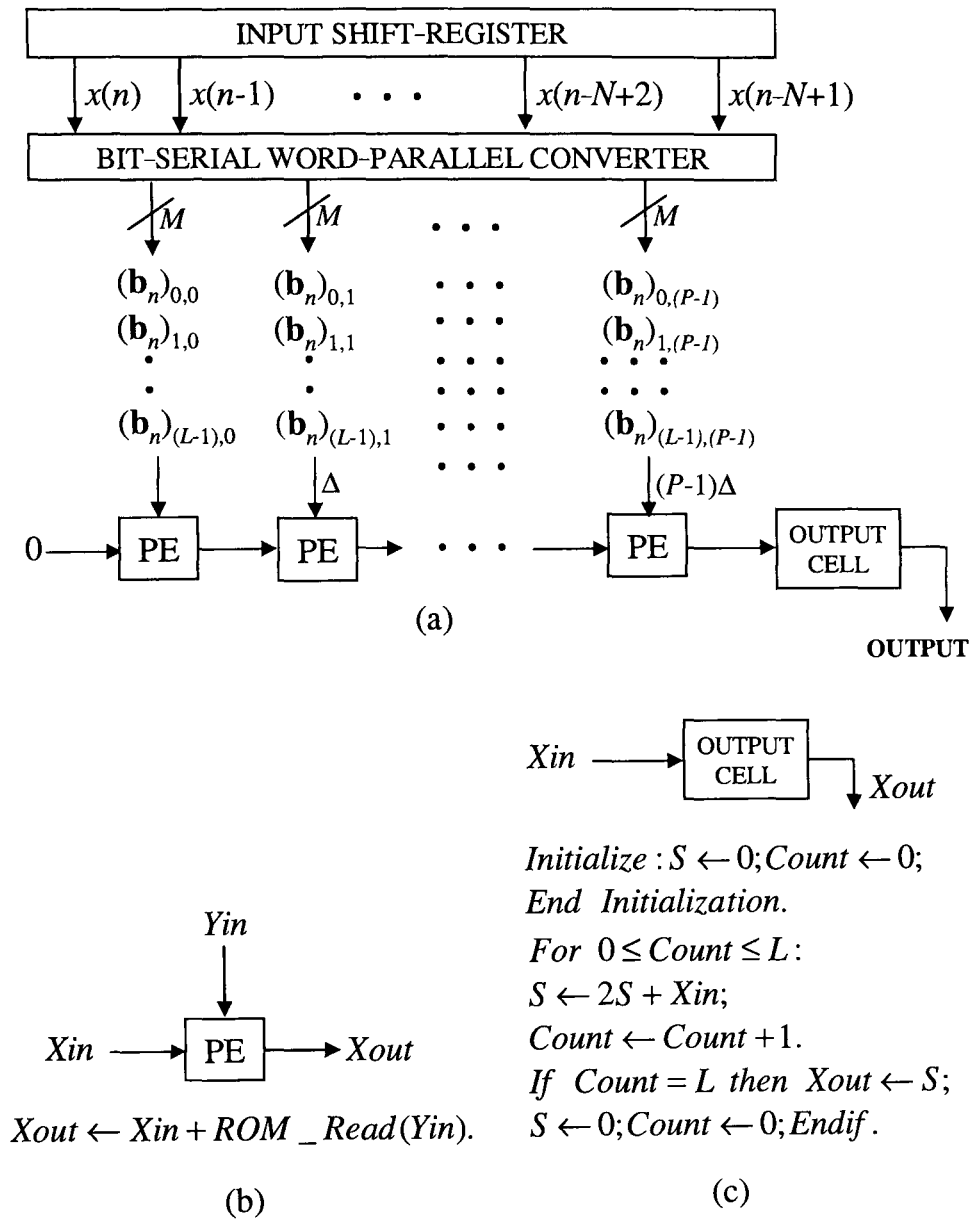
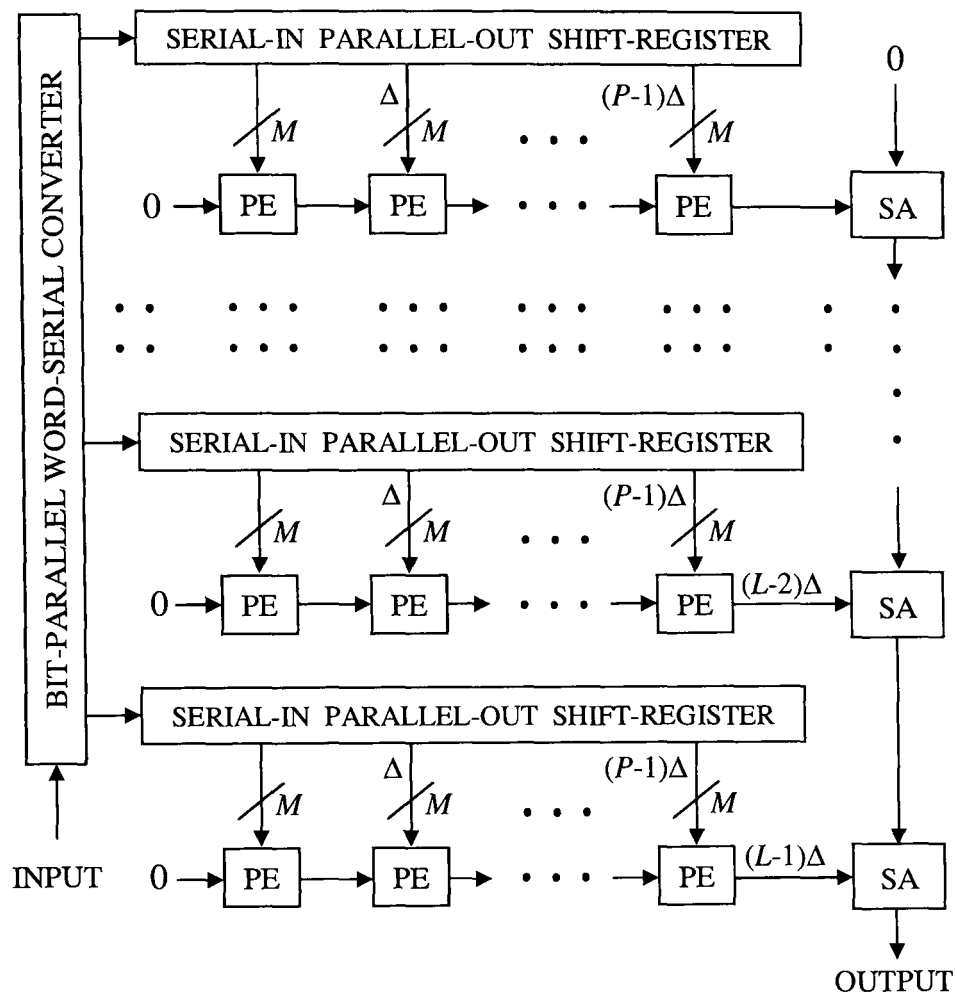


Figure 4.14: The proposed 1-D array for DA-based implementation of FIR filter. (a) The linear systolic array. (b) Function of PE. (c) Function of output cell. Δ stands for a unit delay.

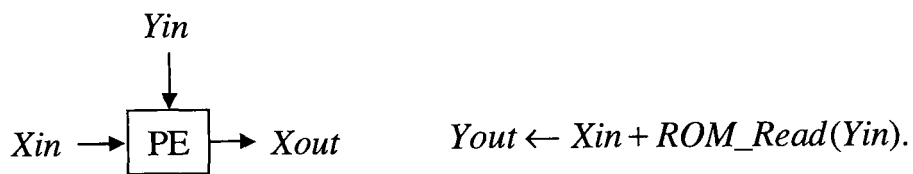
convolved output in every L cycles duration.

Proposed 2-D Systolic Structure for FIR Filters

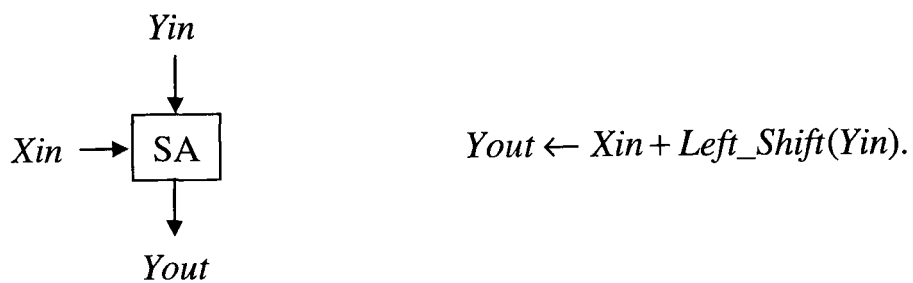
For high-throughput implementation of FIR filters, each node of the DG of Fig. 4.13 can be assigned to a PE exclusively to obtain a 2-D systolic array of L rows and $(P + 1)$ columns as shown in Fig. 4.15. Each row of the structure consists of P number of PEs and a Shift-Add cell (SA). The computation of all the subsequent values of filter output may also be given by similar DGs, and the computation of corresponding nodes of all such DGs may be folded to the same structure. The input



(a)



(b)



(c)

Figure 4.15: The proposed 2-D array for FIR filter. (a) The 2-D systolic array. (b) Function of PE. (c) Function of SA cell. Δ stands for unit delay.

samples are fed to a bit-parallel word-serial converter which receives a new input sample in every cycle period and generates L number of bits of the input sample and feeds one bit each to L number of bit-level Serial-In Parallel-Out Shift Register

(SIPOSR) associated with each row of PEs, as shown in Fig. 4.15(a). Each SIPOSR contains a bit-stream of the corresponding bits of all the input words, such that the SIPOSR on upper-rows contain the more significant bits compared to that of the lower rows. Each of the SIPOSRs of the structure shifts its content to right by one bit-location and receives a new bit in every cycle, upon arrival of a fresh sample to the bit-serial word-parallel converter. The bit-vector $(b_n)_{l,p}$ consisting of M number of bits from the $(l + 1)^{th}$ SIPOSR is loaded to the $(p + 1)^{th}$ PE of the $(l + 1)^{th}$ row (for $0 \leq l \leq L - 1$ and $0 \leq p \leq P - 1$). Each PE [shown in Fig. 4.15(b)] uses the bit vector $(b_n)_{l,p}$ as address for its LUT to read a partial result. The PE then adds the input available from the left with its recently read partial result and passes that out to its right. Each row of the structure is terminated with an SA cell. The function of SA is depicted in Fig. 4.14(c). Each SA during a cycle period makes a left-shift of its input available from the top and adds that input to its input available from the left. The sum is then passed downward to its adjacent SA. To meet the data-dependence requirement, the SA cell on every $(l)^{th}$ row is staggered by one cycle period with respect to the SA cell on the $(l + 1)^{th}$ row.

In the single-array structure of Fig. 4.14, the processing of different bit-streams are time-multiplexed to the same PE, while in the 2-D structure of Fig. 4.15 each bit-stream is processed by a separate row of PEs. We can also derive a structure with q number of such linear arrays (for $L = qu$, where q and u are positive integers) by projecting the nodes of u number of rows of the DG to a single array structure instead of projecting the nodes of all the rows to a single linear array. One may, therefore, opt to derive a structure with multiple linear arrays and similarly may also opt for a suitable value of P ($P =$ number of PEs on one row of the array) for flexible implementation to meet the hardware and time specification of constraint-driven systems.

4.4.3 Results and Discussions

The results of FPGA implementation, in terms of area and maximum usable frequency metrics with respect to the filter order N and address-length M are presented in Table 4.11.

Table 4.11: Key FPGA performance metrics of the proposed DA-Based FIR Filter (for Word-Length $L = 8$)

order (N)	address size (M)	area (slices)	frequency (MHz)
8	2	144	68.111
	4	133	70.413
	8	149	58.377
16	2	287	57.320
	4	260	55.066
	8	286	51.819
32	2	555	55.320
	4	524	53.392
	8	553	51.006
64	2	1057	51.722
	4	1061	50.584
	8	1094	50.360

A number of interesting observations can be made from the data presented in Table 4.11. It can be seen that for a given filter order N , the case for $M = 4$ yields the more area-efficient architecture when compared to the case for $M = 2$ and 8. This can be explained by the fact that the increase in control logic and number of delay elements outweighs the gains made by reduction of LUT size for $M = 2$, while for $M = 8$, the memory requirement of LUTs is too high. Also, it is worth mentioning that four input LUTs are the basic building block of the Virtex-E CLB structure and this accounts for the most efficient mapping of the DA-LUT to the available hardware resources for the case of $M = 4$.

For a given platform, the maximum usable frequency of a design depends on a number of factors:

- The logic depth of the design, which depends on the complexity of the algorithm to be implemented;
- The architectural choices, that demand specific FPGA resources (embedded multipliers, BRAMs etc);

- The device characteristics of the platform (e.g., speed grade and circuit technology);
- For a fixed platform and a given parameterisable IP core, the factors that influence the maximum usable frequency are the parameters of the core; which in this case are filter order N and the decomposition factor P ; and
- Larger the area of spread of the clock-tree lower is the maximum usable frequency because of the clock skew resulted by the propagation delay across the computing circuits.

The above points clearly account for the fact that the overall trend for maximum usable frequency shows a decreasing trend as the address-length increases. This is because of the fact that the depth of systolic logic is generally shallow and the critical-path is proportional to the ROM size. When the address-length increases, not only does the size of the ROM increases exponentially but also the number of address lines increase linearly. The size of the multiplexer logic for de-referencing the ROM locations depends on the above mentioned factors and contributes to the critical path. It must be highlighted that for the case $M = 16$ and above, it was not possible to synthesise or place and route the design due to exponential increase in ROM size to 65536 words.

Comparison with Existing Architectures

Details of the performance of the proposed architecture in terms of the basic design metrics are tabulated alongside with those of comparable existing architectures in Table 4.12. It must be highlighted that the architecture proposed in [153] has been implemented on an Altera Stratix FPGA device. Significant architectural differences in the FPGA fabric between Xilinx and Altera devices precludes the possibility of an objective and direct comparison between the design metrics of our architecture with those reported in [153]. Additionally, FPGA implementation details have been provided only for the case $B_c = 18$ (where B_c has been defined in [153] as word-length of the original LUT). To make a fair comparison; it is necessary to homogenise the FPGA platform and parameters of the DA-based FIR filter. Hence, we have faithfully

Table 4.12: Comparison of performance of the proposed architecture with existing work.

filter order	proposed		Yoo <i>et al</i> [153]	
	area (slices)	frequency (MHz)	area (slices)	frequency (MHz)
8	133	74.025	146	70.552
16	260	67.222	283	62.775
32	524	63.131	547	61.166
64	1061	64.049	1076	57.192

The address-length M is taken to be four for the proposed architecture.

re-implemented the architecture presented in [153] on the same platform that has been used in this work for word-length $L = 8$.

It can be seen from Table 4.12 that the proposed architecture clearly outperforms existing implementations in both key metrics of area occupied and maximum usable frequency for all the values of N . The superior performance of the proposed systolic design is due to the fact that the number of adders increases linearly with filter order N for the most optimum implementation as opposed to the case in [153] where the architecture presented uses a tree of adders to calculate the final values before shift-accumulation operation. Additionally, it is worth mentioning that 16 word ROMs are more efficiently mapped to the 4-input LUT structure (common to both Xilinx and Altera FPGAs) than the additional control logic in the architecture presented in [153]. Apart from that, the complexity of control logic is also minimal in the proposed systolic architectures. Moreover, in our design, only a single shift-accumulate operation is needed, irrespective of filter order N as a result of systolisation. All these factors yield the most efficient implementation in all key performance metrics in our proposed architecture.

Chip Level Details

Careful manual place and route of critical nets and manual pin assignment for the designs has been performed using Xilinx PACE and Floorplanner [11]. The chip diagram for one of the implementations is shown in Fig. 4.16.

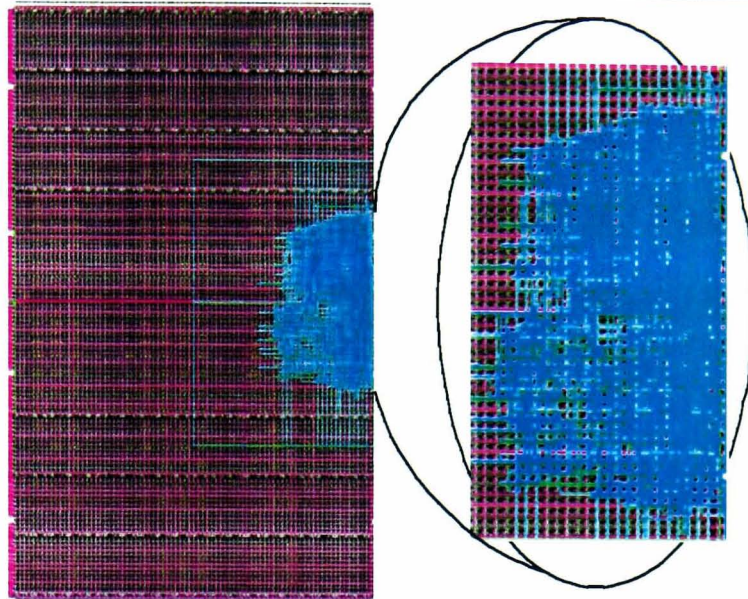


Figure 4.16: FPGA chip diagram for FIR filter realisation for $N = 64$, $M = 8$ and $L = 8$ (Xilinx XCV2000E FPGA)

Power Consumption

XPower estimates of power dissipation are presented in Table 4.13. It can be seen that I/O power remains constant for all architectures. This is explained by the fact that the architectures are fully parameterisable and pipelined and consequently process one I/O value per operational cycle (or L clock cycles). Hence, for a fixed word-length L for a given clock frequency, the input power remains constant. Output power increases linearly with the number of output pins, which is also a function of filter order N . However, for a fixed value of N , output power also remains constant across all values of address-length M . The total dynamic on-chip power is graphically presented for all cases in Fig. 4.17.

Energy Analysis

In case of high-throughput DSP circuits, energy is a more appropriate measure to quantify the efficiency of an operation. A suitable estimate of energy consumption will enable to decide on the design choice that can meet the throughput requirement while minimising power consumption as well. We have analysed two parameters of energy estimates: EOP and ET ⁴ of the proposed architecture for FIR filter.

⁴Refer to Appendix D for details

Table 4.13: Power dissipation of the proposed FPGA implementation of FIR filter for different filter orders and address-lengths.

N	M	power dissipation (mW)				
		clock	input	logic	output	signal
8	2	16.85	1.41	44.52	58.06	29.95
	4	13.90	1.41	42.16	58.06	33.63
	8	16.33	1.41	56.78	58.06	43.89
16	2	25.65	1.41	83.75	61.12	64.37
	4	28.23	1.41	75.13	61.12	65.42
	8	21.78	1.41	101.08	61.12	83.43
32	2	42.60	1.41	169.86	64.18	108.66
	4	43.93	1.41	144.90	64.18	108.03
	8	39.33	1.41	195.71	64.18	137.39
64	2	86.38	1.41	345.87	67.23	195.55
	4	71.32	1.41	296.81	67.23	179.11
	8	68.14	1.41	389.70	67.23	243.56

Power estimation has been carried out for 50 MHz frequency.

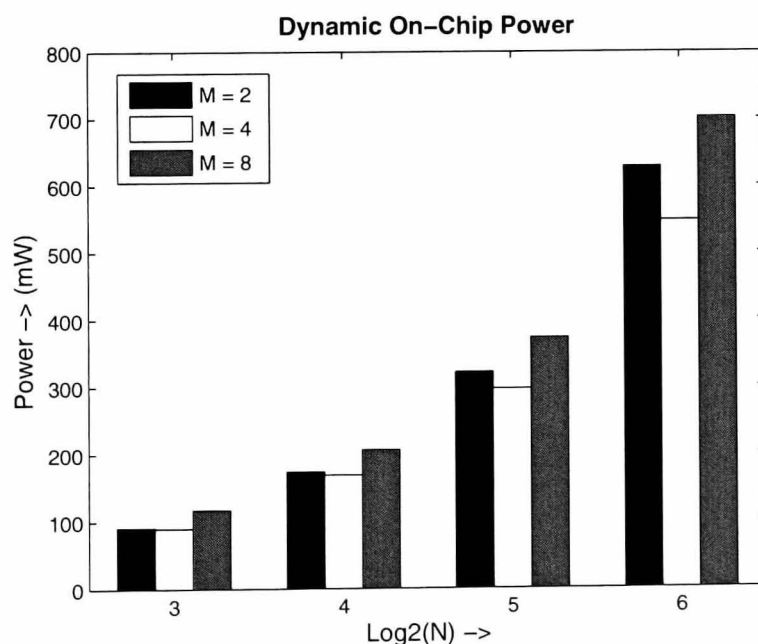


Figure 4.17: Plot of variation of dynamic on-chip power with filter order.

It can be seen from Fig. 4.18 that EOP steadily increases as N increases and is proportional to the increase in circuit size and complexity. The ET data is graphically presented in Fig. 4.19. It can be seen that the most energy efficient architecture is obtained for the case $M = 4$, in line with the power metrics obtained.

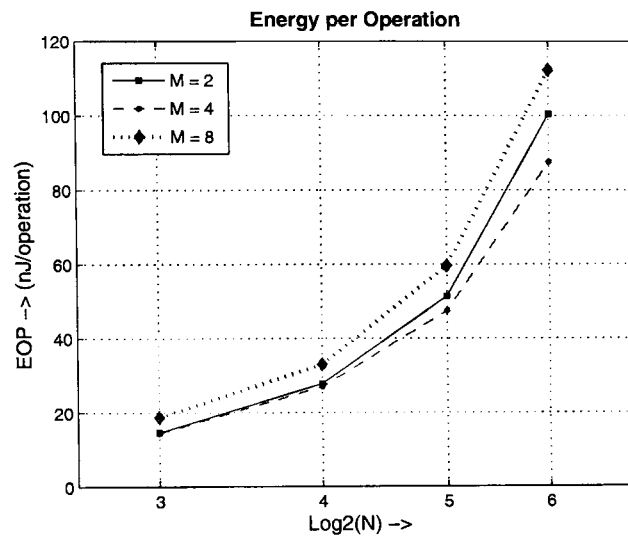


Figure 4.18: Energy per operation of FPGA implementations FIR filter for different values of N and M .

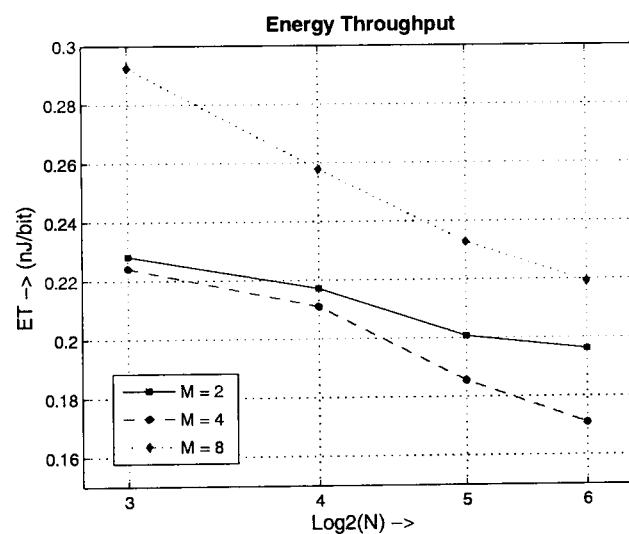


Figure 4.19: Energy throughput of FPGA implementations for different values of N and M .

4.5 Efficient FPGA Implementation of Colour Space Conversion

Processing an image in the RGB colour space, with a set of RGB values for each pixel is not the most efficient method. To speed up some processing steps many broadcast, video and imaging standards use luminance and colour difference video signals, such as YCrCb, making a mechanism for converting between formats necessary. Decomposing an RGB colour image into one luminance image and two chrominance images (YCrCb) is the method that has been used in most commercial applications such as face detection [154, 155], as well as the JPEG and MPEG imaging standards [156–158].

The calculation of RGB colour components from YCrCb components consumes up to 40% of the processing power in a highly optimised decoder [156]. Accelerating this operation would be useful for the acceleration of the whole process. Therefore, techniques which efficiently implement this conversion are desired.

It is the aim of this work-package to develop power efficient architecture based on DA, ideally suited for the implementation of an RGB to YCrCb CSC on FPGAs. The constant conversion matrices have been exploited to develop the mathematical model in order to reduce the ROM size, the area consumed by the design and to speed up the computation procedure by minimising the number of addition and subtraction operations required.

4.5.1 Mathematical Background

Consider an $N \times M$ image (N : image height, M : image width).

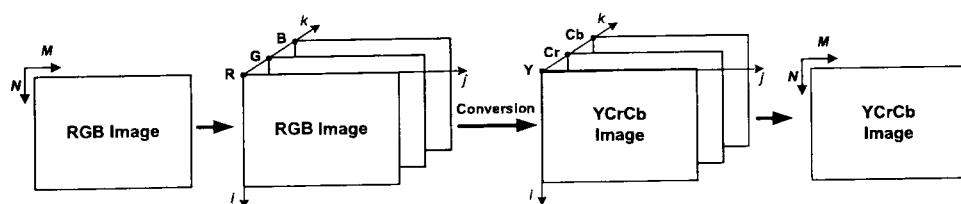


Figure 4.20: RGB to YCrCb conversion

Let us represent each image pixel by b_{ijk} , ($0 \leq i \leq N - 1, 0 \leq j \leq M - 1, 0 \leq k \leq 2$)

where:

$$\left\{ \begin{array}{l} b_{ij0} = R_{ij} \quad \text{the red component of the pixel in row } i \\ \quad \quad \quad \quad \quad \text{and column } j \\ \\ b_{ij1} = G_{ij} \quad \text{the green component of the pixel in row } i \\ \quad \quad \quad \quad \quad \text{and column } j \\ \\ b_{ij2} = B_{ij} \quad \text{the blue component of the pixel in row } i \\ \quad \quad \quad \quad \quad \text{and column } j \end{array} \right. \quad (4.17)$$

The image can be converted using the following mathematical formula:

$$\left(\begin{array}{c} \left(\begin{array}{c} c_{000} \\ c_{001} \\ c_{002} \\ c_{100} \\ c_{101} \\ c_{102} \\ \vdots \\ c_{(N-1)00} \\ c_{(N-1)01} \\ c_{(N-1)02} \end{array} \right) \quad \dots \quad \left(\begin{array}{c} c_{0(M-1)0} \\ c_{0(M-1)1} \\ c_{0(M-1)2} \\ c_{1(M-1)0} \\ c_{1(M-1)1} \\ c_{1(M-1)2} \\ \vdots \\ c_{(N-1)(M-1)0} \\ c_{(N-1)(M-1)1} \\ c_{(N-1)(M-1)2} \end{array} \right) \\ \\ \left(\begin{array}{c} b_{000} \\ b_{001} \\ b_{002} \\ 1 \\ b_{100} \\ b_{101} \\ b_{102} \\ 1 \\ \vdots \\ b_{(N-1)00} \\ b_{(N-1)01} \\ b_{(N-1)02} \\ 1 \end{array} \right) \quad \dots \quad \left(\begin{array}{c} b_{0(M-1)0} \\ b_{0(M-1)1} \\ b_{0(M-1)2} \\ 1 \\ b_{1(M-1)0} \\ b_{1(M-1)1} \\ b_{1(M-1)2} \\ 1 \\ \vdots \\ b_{(N-1)(M-1)0} \\ b_{(N-1)(M-1)1} \\ b_{(N-1)(M-1)2} \\ 1 \end{array} \right) \end{array} \right) = \left(\begin{array}{cccc} A_{00} & A_{01} & A_{02} & A_{03} \\ A_{10} & A_{11} & A_{12} & A_{13} \\ A_{20} & A_{21} & A_{22} & A_{23} \end{array} \right) \otimes \quad (4.18)$$

where the operation \otimes can be defined as follows:

Each vector $\begin{pmatrix} c_{ij0} \\ c_{ij1} \\ c_{ij2} \end{pmatrix}$ is the result of the product $\begin{pmatrix} A_{00} & A_{01} & A_{02} & A_{03} \\ A_{10} & A_{11} & A_{12} & A_{13} \\ A_{20} & A_{21} & A_{22} & A_{23} \end{pmatrix} \times \begin{pmatrix} b_{ij0} \\ b_{ij1} \\ b_{ij2} \\ 1 \end{pmatrix}$,

where c_{ijk} represent the output image colour space components and

$A = \begin{pmatrix} A_{00} & A_{01} & A_{02} & A_{03} \\ A_{10} & A_{11} & A_{12} & A_{13} \\ A_{20} & A_{21} & a_{22} & A_{23} \end{pmatrix}$ represents one of the constant matrices relating the the CSC coefficients.

The c_{ijk} elements (the output image colour space components) can be computed using the following equation:

$$c_{ijk} = \sum_{m=0}^3 A_{km} \times b_{ijm} \quad (4.19)$$

$$(0 \leq i \leq N - 1, 0 \leq j \leq M - 1, 0 \leq k \leq 2)$$

where $\{A_{km}\}$'s are L -bit constants and $\{b_{ijm}\}$'s are written in the unsigned binary representation as shown in equation 4.20:

$$b_{ijm} = \sum_{l=0}^{W-1} b_{ijm,l} \times 2^l \quad (4.20)$$

$$(0 \leq i \leq N - 1, 0 \leq j \leq M - 1, 0 \leq m \leq 2)$$

where $b_{ijm,l}$ is the l^{th} bit of b_{ijm} , which is zero or one, W is the word-length used which represents the resolution for each colour component of a pixel.

Substituting 4.20 in 4.19,

$$c_{ijk} = \sum_{m=0}^3 A_{km} \times \left(\sum_{l=0}^{W-1} b_{ijm,l} \times 2^l \right) =$$

$$\sum_{l=0}^{W-1} \left(\sum_{m=0}^3 A_{km} \times (b_{ijm,l} \times 2^l) \right) \quad (4.21)$$

Define:

$$Z_l = \sum_{m=0}^3 A_{km} \times b_{ijm,l} \quad (4.22)$$

Therefore, c_{ijk} can be computed as:

$$c_{ijk} = \sum_{l=0}^{W-1} Z_l \times 2^l \quad (4.23)$$

The idea is that since the term Z_l depends on the $b_{ijm,l}$ values and has only 2^4 possible values, it is possible to precompute and store them in ROMs. An input set of 4 bits ($b_{ij0,l}, b_{ij1,l}, b_{ij2,l}, b_{ij3,l}$) is used as an address to retrieve the corresponding Z_l values. The ROM's content is different and depends on the constant matrix A coefficients. These intermediate results are accumulated in W clock cycles to produce c_{ijk} coefficients.

Since all the components are in the range of 0 to 255, 8 bits ($W = 8$) are enough to represent them.

Equation 4.23 becomes:

$$c_{ijk} = \sum_{l=0}^7 Z_l \times 2^l \quad (4.24)$$

Three ROMs (one for each matrix A row) with the size of $2^N = 2^4 = 16$ are needed in order to store the precompute 2^4 possible partial products values.

Since the element b_{ij3} is equal to 1:

$$b_{ij3,l} = \begin{cases} 1 & \text{for } l = 0 \\ 0 & \text{for } l \neq 0 \end{cases} \quad (4.25)$$

Equation 4.24 can be rewritten as:

$$c_{ijk} = \sum_{l=0}^7 Z_l^* \times 2^l + A_{k3} \quad (4.26)$$

Where:

$$Z_i^* = \sum_{m=0}^2 A_{km} \times b_{ijm,l} \quad (4.27)$$

	$m = 0$	$m = 1$	$m = 2$		
$c_{ijk} =$	$(A_{k0} \times b_{ij0,0} + A_{k1} \times b_{ij1,0} + A_{k2} \times b_{ij2,0}) \times 2^0 +$			$l = 0$	
	$(A_{k0} \times b_{ij0,1} + A_{k1} \times b_{ij1,1} + A_{k2} \times b_{ij2,1}) \times 2^1 +$			$l = 1$	
	$(A_{k0} \times b_{ij0,2} + A_{k1} \times b_{ij1,2} + A_{k2} \times b_{ij2,2}) \times 2^2 +$			$l = 2$	(4.28)
\vdots	\vdots			\vdots	
	$(A_{k0} \times b_{ij0,7} + A_{k1} \times b_{ij1,7} + A_{k2} \times b_{ij2,7}) \times 2^7 +$			$l = 7$	
A_{k3}					

It is worth mentioning that the size of the ROMs has been reduced to 2^3 . Table 4.14 gives the content of each ROM.

Table 4.14: Content of the ROM i ($0 \leq i \leq 2$)

$b_{ij0,l}$	$b_{ij1,l}$	$b_{ij2,l}$	The Content of the ROM i
0	0	0	0
0	0	1	A_{i2}
0	1	0	A_{i1}
0	1	1	$A_{i1} + A_{i2}$
1	0	0	A_{i0}
1	0	1	$A_{i0} + A_{i2}$
1	1	0	$A_{i0} + A_{i1}$
1	1	1	$A_{i0} + A_{i1} + A_{i2}$

4.5.2 Proposed Architecture for CSC

Equation 4.26 can be mapped into the proposed architecture as shown in Fig. 4.21. The architecture consists of 8 identical PEs (PE_n s) ($0 \leq n \leq 7$). Each PE_n comprises three parallel signed integer adders, three n right shifters and one Memory Block (MB), which has the structure shown in Fig. 4.22.

It is worth noting that the architecture has a latency of W and a throughput rate equal to 1. The entire image conversion can be carried out in $(Latency + (N \times M) Throughput) = 8 + (N \times M)$ clock cycles, while using the standard algorithm ([159]), the conversion can be carried out in $(3 \times 4 \times N \times M)$ clock cycles, where (3×4) is the constant matrix A size.

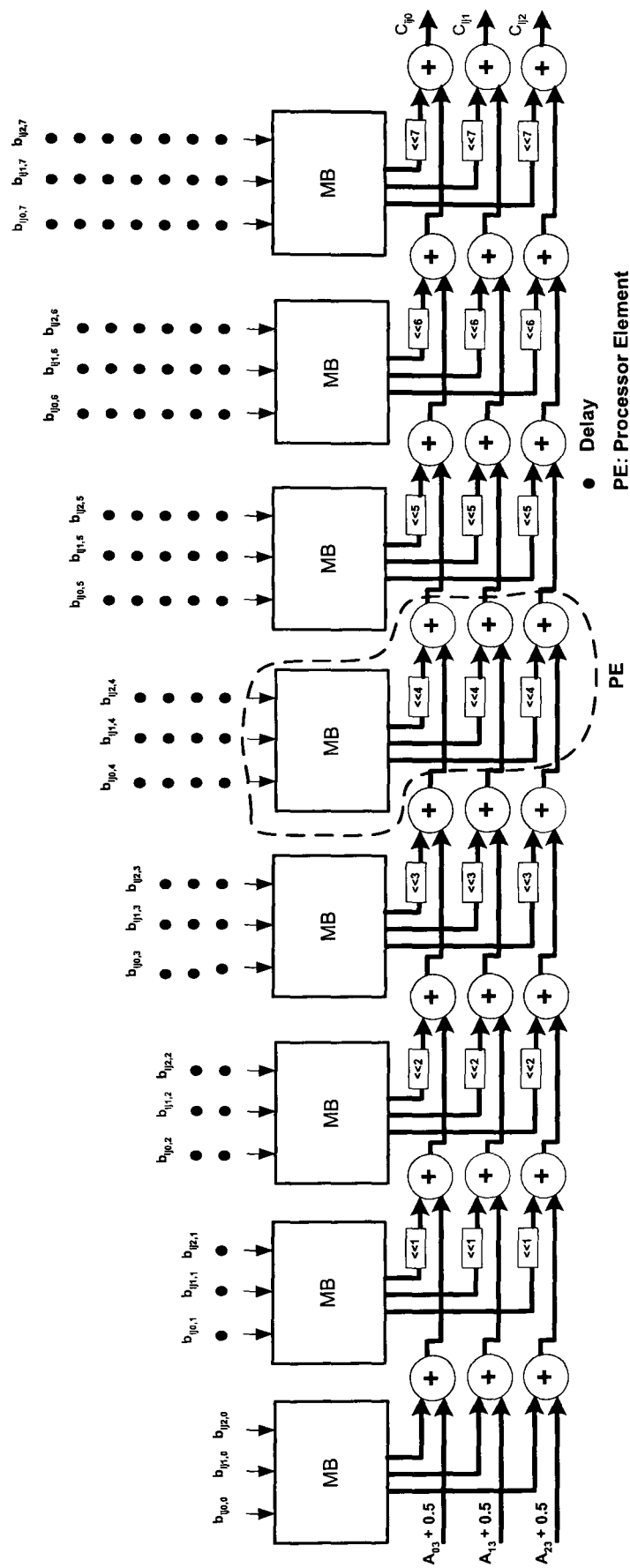


Figure 4.21: Proposed architecture based on DA principles

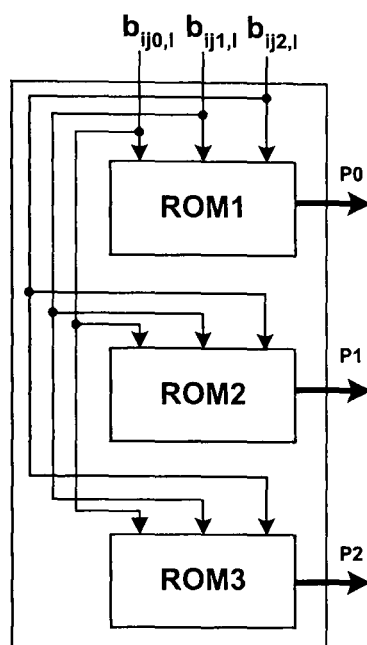


Figure 4.22: MB structure

4.5.3 FPGA Implementation

In order to verify the performance of the proposed CSC, the design has been prototyped on the Celoxica RC1000 board [116] [Appendix B].

The pre-computed partial products are stored in the ROMs using 13 bits fixed point representation (8 bits for integer part and 5 bits for fractional part). 13-bit arithmetic is used inside the architecture. The inputs and outputs of the architecture are presented using 8 bits and the outputs are rounded. Rounding usually looks at the decimal value and if it is greater than or equal to 0.5, then the result is increased by one. This implies a condition of verifying followed by another arithmetic operation. A more efficient way to round a number is to add 0.5 to the result and truncate the decimal value. This technique has been applied in our proposed architecture. The initial value for each PE's ACC (for the serial architecture) and for the first PEs adder (for the parallel architecture) is set in advance to $(A_{i3} + 0.5)$, where $(0 \leq i \leq 2)$. The MACs and parallel signed adders have been implemented using Xilinx CoreGen utility, which contains many efficient designs that can often save time for a programmer [160]. The shifters and ROMs initialisation have been implemented using VHDL. All design components have been connected together using Handel-C [161].

Performance Metrics and Results

The proposed CSC core has been prototyped on the RC1000 FPGA board, equipped with XCV2000E. The design 186 of the total available slices and runs with a maximum frequency of 277 MHz. In order to make a fairer and consistent comparison with some existing FPGA based CSCs using the same technology [44–46] the XCV50E-8 FPGA device has also been targeted. A comparison with other FPGA platforms utilising the new available features has been also performed [40, 162]. Table 4.15 illustrates the performances obtained for the proposed architecture in terms of area consumed and speed which can be achieved.

Table 4.15: Performance comparison with existing CSC cores

CSC Core	Platform	Resources	Speed (MHz)
Proposed	XCV50E	186 Slices	263
Xilinx [40]	XC2V500	131 LUTs + 5 [18x18] Mult	185
Architecture 1 [162]	Cyclone-II	292 LEs	216
Architecture 2 [162]	Cyclone-II	78 LEs	175
CAST Inc [45]	XCV50E	222 Slices	112
ALMA Tech [46]	XCV50E	222 Slices	105
Amphion Ltd. [44]	XCV50E	204 Slices	90

The proposed architecture shows significant improvements in comparison with the existing implementations [40, 44–46, 162], which perform the RGB to YCrCb conversion, in terms of the area consumed and the maximum running clock frequency.

Power and Energy Details

Since this is a relatively simple core with a minimal number of design parameters, power analysis and energy calculations are not required. Power modelling details for this core may be referred to in Chapter 6.

Chip Level Details

Careful manual place and route of critical nets and manual pin assignment for the designs has been performed using Xilinx PACE and Xilinx Floorplanner [11] on the

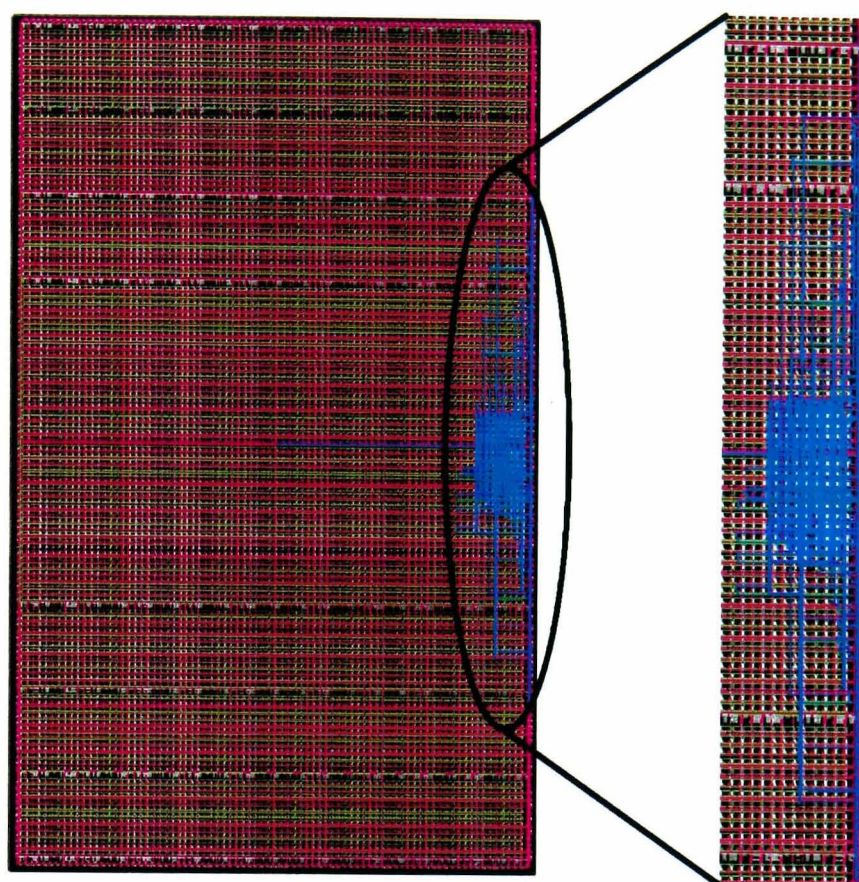


Figure 4.23: Design layout for the proposed architecture (XCV2000E FPGA)

XCV2000E and XCV50E chips. The optimised chip layout for the Xilinx XCV2000E FPGA platform is shown in 4.23.

4.6 Conclusions

In Sections 4.2 and 4.3, high-speed, power aware architectures for the FRAT and FRIT respectively have been presented. These architectures exploit various architectural strategies including parallelism, pipelining and sub-block systolisation to yield performance metrics that clearly outperform existing implementations.

A high performance systolic architecture for FIR filtering has been presented in Section 4.4. This architecture also outperforms comparable work. Design space exploration has also been carried out to determine tradeoffs in performance, factorisation level, power, and energy.

Efficient FPGA implementation of DA based architecture for CSC is discussed in Section 4.5. The proposed architecture clearly outperforms existing ones, including

commercially available IP cores. All the above mentioned cores have been developed using principles of power aware design.

To summarise, architectural level optimisation techniques have been discussed in this chapter. In the next chapter, performance enhanced voltage scaling as a power reduction technique for FPGA based designs is discussed.

Chapter 5

Performance Enhanced Voltage Scaling in FPGAs

Voltage scaling represents a sub-RTL level power and energy reduction technique for FPGA based designs. The relative placement of this abstraction level is indicated in the power triangle shown in Fig. 1.7.

While the fabric of FPGA chips and hence static power is technology dependant, dynamic power consumption of designs implemented on FPGA can benefit from exploiting the massive parallelism capabilities of commercial FPGAs. In cases where the hardware resources available on the FPGA are not utilised completely, there is scope for trading off additional area for improved power and energy metrics with no penalty in terms of performance. At the architectural system level, this is possible by parallelising the entire core or suitable subsections of the core implemented on FPGA and proportionately reducing the clock frequency in order to maintain the same throughput. Additionally, quadratic improvement in power-delay can be achieved by reducing the supply voltage to take advantage of the reduced operational frequency. It is the aim of this chapter to conduct the first systematic empirical study of the tradeoffs between degree of parallelism, threshold voltage and power consumption under constant throughput conditions commercially available FPGAs. Results indicate that there is excellent scope for reduction in dynamic voltage by suitably applying the tradeoffs in FPGA based designs in order to achieve energy efficient implementations. It must be highlighted that in previous chapters, algorithmic and

architectural optimisation techniques have been presented and applied to a number of cores that have been developed in-house. In contrast, the cores that are used to test the concepts presented in this paper are standard commercially available ones. Parallelism has been performed at the core level, and not within the cores themselves. The key focus is on the voltage scaling aspect rather than on core optimisation.

The rest of this chapter is organised as follows. A brief discussion on the motivation for power and energy reduction, theory of power dissipation in SRAM FPGAs and techniques for optimisation is presented in Section 5.1. Analysis of the overall methodology and results obtained are presented in Section 5.2. Concluding remarks are presented in Section 5.3.

5.1 Power and Energy Dissipation: An FPGA perspective

As opposed to Programmable Array Logics (PALs) and Programmable Logic Devices (PLDs), which have fixed power consumption, power dissipation of FPGAs depends on such factors as utilisation, operating frequency, operating voltage, capacitance and load conditions. So power is one of the two most serious concerns (along with design complexity) in FPGA design. The following subsections describe the various aspects of power and energy aware design on SRAM FPGAs. In this section, a brief analysis of power dissipation in digital circuits followed by FPGA specific details have been presented.

5.1.1 Power Dissipation Sources in Digital Circuits

There are various sources of power dissipation in digital circuits. They include capacitive switching, leakage and short circuit power [163]. Capacitive power occurs due to toggling of digital circuits which requires charge-discharge action. Due to the non-ideal subthreshold behaviour of transistors, there will be leakage currents I_{leak} flowing from the positive power supply to ground even in the static case resulting

in the leakage power P_{leak} . Short circuit power occurs due to the currents flowing from the positive power supply to ground when n and p channel transistors are conducting simultaneously for a short moment during node transitions.

5.1.2 Power Dissipation: FPGA Specific Details

Compared to ASICs and other custom chips, FPGAs contain long routing tracks with significant parasitic capacitance. During high-speed operations, the switching activity on these long routing tracks causes significant power dissipation. Power dissipation calculations for FPGAs are similar to other complementary metal-oxide semiconductor application-specific integrated circuit (CMOS ASIC) devices. The total power usage of an FPGA device (P_{Total}) can be broken down into total StP and total DP:

$$P_{Total} = StP + DP \quad (5.1)$$

Static and Dynamic Power Dissipation in FPGAs

The StP of an FPGA is proportional to the static current I_{dd} - the current that flows regardless of gate switching (transistor is 'on' or 'off'). This is otherwise called the quiescent power. DC power dissipation can be estimated by the worst-case equivalent equation: $StP = V_{dd}I_{dd}$. StP is inherently dependant on the architectural layout of the FPGA itself and is technology dependant. As such, it cannot be controlled by the FPGA *based* designer and will not be addressed in this work.

We can recollect from Chapter 1 that the DP consumption of FPGAs can be separated into data-path, synchronisation and off-chip power. In the following Sections, a mathematical quantification of the interplay of DP with Voltage Scaling and performance enhancement is presented.

Quantifying the Effect of Parallelism on Power Dissipation

We define the instantaneous dynamic power dissipation of an FPGA based design as:

$$P_t = V_{dd}^2 \cdot C \cdot f \cdot \delta_t \cdot A \quad (5.2)$$

where V_{dd} is the supply voltage, C is the effective capacitance per unit area, f is the frequency of operation, δ_t is the instantaneous activity rate and A is the area occupied in slices. The average power dissipation P_{av} is then given by:

$$P_{av} = \frac{\int P_t \cdot dt}{T} \quad (5.3)$$

where T is the number of clocks required to complete one computational cycle. Parallelism by a factor of k multiplies the resources occupied by the same amount. However, reducing the effective frequency can now be reduced by the same factor to maintain constant throughput. Theoretically, the FPGA should, under these circumstances consume the same amount of power.

Substituting Eq. 6.2 in Eq. 6.1 and quantitatively applying the effect of “ k ” level parallelism, we get:

$$P_{av} = \frac{\int V_{dd}^2 \cdot C \cdot (f/k) \cdot (A \cdot k) \cdot dt}{T} \quad (5.4)$$

which reduces to the same expression stated in Eq. 6.2.

The Effect of Voltage Scaling

Voltage scaling is an effective method of greatly reducing power consumption of the FPGA, at the expense of performance. The key benefit of supply voltage scaling is that it reduces both static and dynamic power. From Eq. 6.1 it is clear that reducing the supply voltage has a quadratic effect on power dissipation. However, there is a cost associated with this technique: an associated delay in the speed of the circuit, or effectively, a reduction in the frequency at which the circuit can operate reliably. However, when voltage scaling is combined with architectural modifications incorporating techniques such as parallelism and pipelining, the increased performance efficiency can be traded off for the reduction in effective operational frequency.

Circuit delay displays a first order relationship with supply voltage as shown in Fig. 5.1.

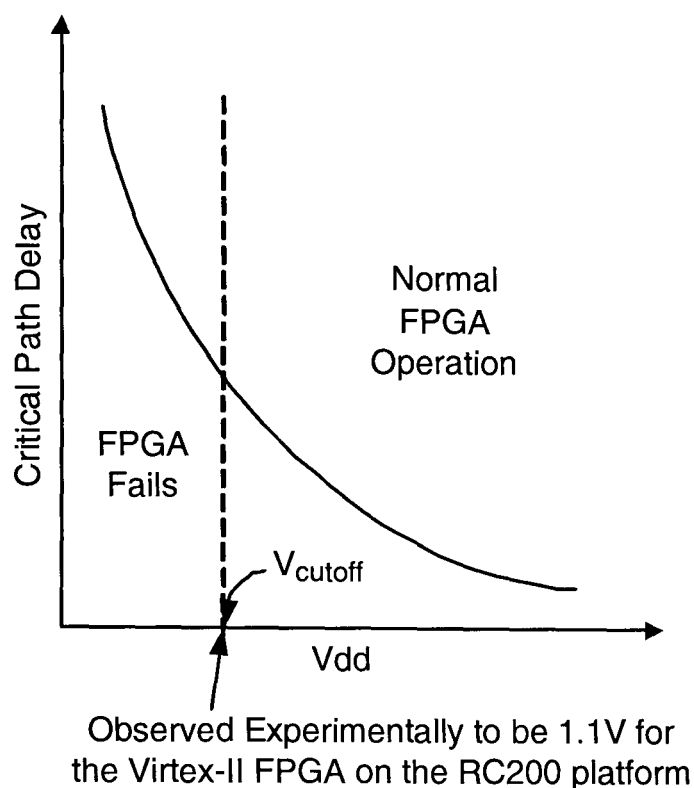


Figure 5.1: Delay vs supply voltage tradeoff trends and FPGA operational conditions

5.2 FPGA Implementation: Empirical Study

To study the effect of the various tradeoffs, a suitable test strategy to enable systematic tabulation of the results has been applied. The bitstream files for the testbench cores are then synthesised and the power dissipation details under varying parameters are tabulated.

5.2.1 Description of Methodology

- Identify suitable benchmarks for testing the proposed voltage scaling and parallelism strategies;
- Modify a suitable commercially available FPGA board to supply V_{dd} from an external bench power supply unit, to enable boundary condition testing and scaling of supply voltage;

- For each benchmark core to be tested, synthesise bitstreams for varying levels of parallelism;
- For each core at every level of parallelism, measure/simulate the power dissipation under the appropriate frequency required to maintain the same throughput using various supply voltage values. Supply voltages are stepped down from a peak recommended voltage by 0.1V until FPGA fails to generate the correct result;
- Simulate the power dissipation under XPower. This is done to eliminate board specific I/O overhead that is introduced to interface the core with external data stream; and
- Tabulate the results obtained and graphically plot the power dissipation metrics to visually analyze the tradeoff sample space.

It has been determined experimentally through bench measurements on a customised Celoxica RC200 [161] development board that the point of failure occurs at 1.1V for the Virtex-II FPGA for all three benchmark cores that have been analysed. This board has been primarily used for assessing the actual cut-off voltage and to understand scaling limits on a commercial platform. In order to ensure that power consumption due to board specific I/O's are not taken into account, the actual power estimation data that is graphically presented in the following Sections is obtained from XPower simulations. The lowest operational voltage for power estimation has been restricted to 1.2V.

5.2.2 Description of the Benchmark Cores

Typical mathematical operations that are invariably present in every digital signal processing algorithm are the ubiquitous adder and multiplier circuits. 8 bit non registered adder from the Xilinx Coregen Library (XCL) [11] has been chosen as the first benchmark circuit. 8 bit multiplier with 16 bit outputs have been chosen as the second benchmark circuit. Both the above mentioned cores are parallelised at factors of 2, 4, 8, 16, 32 and 64 respectively. Additionally, it is worth mentioning

that the multiplier is an order of magnitude more complex than the adder circuit. The Discrete Cosine Transform (DCT) core from the XCL has been chosen as the final benchmark circuit. This core represents an order of complexity markup over the multiplier (in terms of area occupied) and due to limitations of on-chip resources in terms of number of slices, the maximum level of parallelism possible for this core is 16. The above cores have been chosen on a representative basis only and enable us to make generalised analysis of the tradeoff outcomes.

5.2.3 FPGA Implementation Details

The Celoxica RC200 board [161] containing the Xilinx Virtex II XC2V1000 FPGA is suitable for evaluation and development of high performance applications. The development board contains two banks of SRAM providing a total of 4MBytes and a *SmartMedia* socket used to configure the FPGA. All the bit files for the FPGA re-configuration are stored in the *SmartMedia* card. The FPGA is reconfigured through CPLD by downloading bit files from the *SmartMedia* card. The FPGA has a total of 5120 slices. The maximum recommended voltage at which the FPGA can be operated is 1.6V.

The benchmark cores are imported into a Handel-C based design flow that acts as a top-level wrapper. Post place and route power estimation is performed using Xilinx XPower [11]. The area metrics obtained for the benchmark cores are presented in Table 5.1. The total available area on the Virtex II XC2V1000 FPGA limits the maximum level of parallelism implemented for the DCT core to a factor of 16, at which 83% of the available slices are occupied.

Table 5.1: Area (slice) metrics under various levels of parallelism

Parallelism	Adder	Multiplier	DCT
1	8	20	281
2	12	36	558
4	22	70	1114
8	37	133	2229
16	70	262	4454
32	137	521	NA
64	270	1038	NA

The chip diagrams for the multiplier and DCT cores at maximum levels of parallelism that have been implemented are presented in Fig. 5.2.

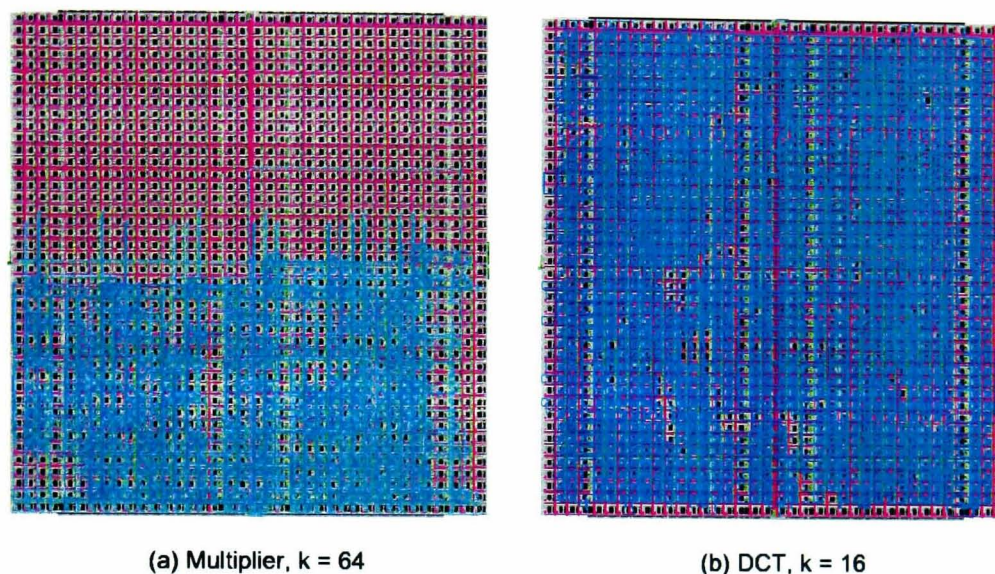


Figure 5.2: Chip diagrams of (a) Multiplier (b) DCT cores (Virtex II XC2V1000 FPGA)

5.2.4 Power/Parallelism/Voltage Tradeoffs

The power estimates under various conditions for the adder, multiplier and DCT cores have been presented graphically in Fig. 5.3, 5.4 and 5.5 respectively.

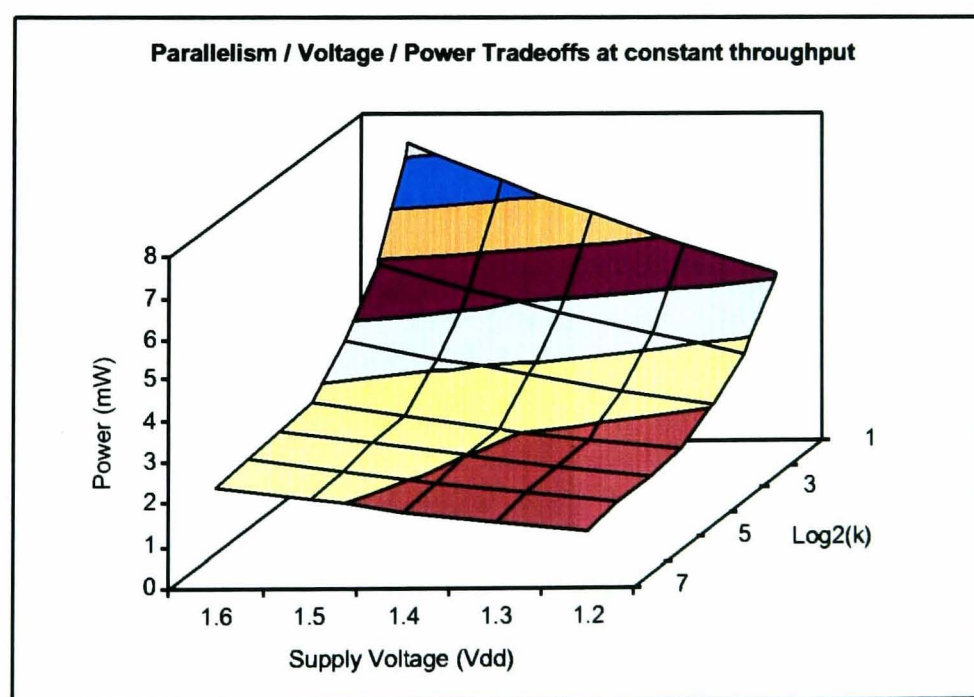


Figure 5.3: Power/Parallelism/Voltage tradeoff for the adder core

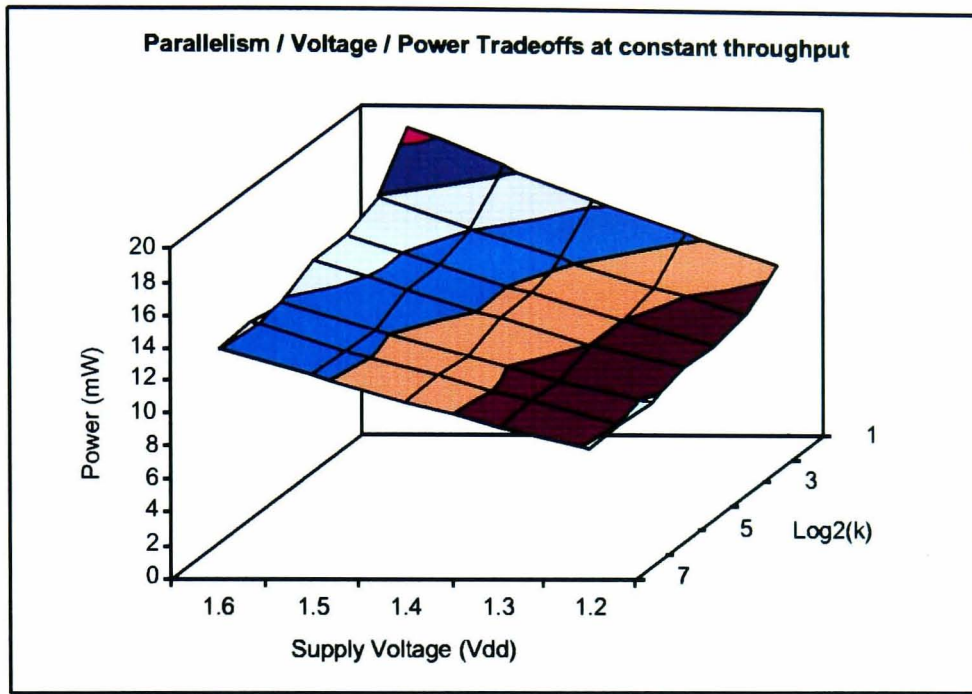


Figure 5.4: Power/Parallelism/Voltage tradeoff for the multiplier core

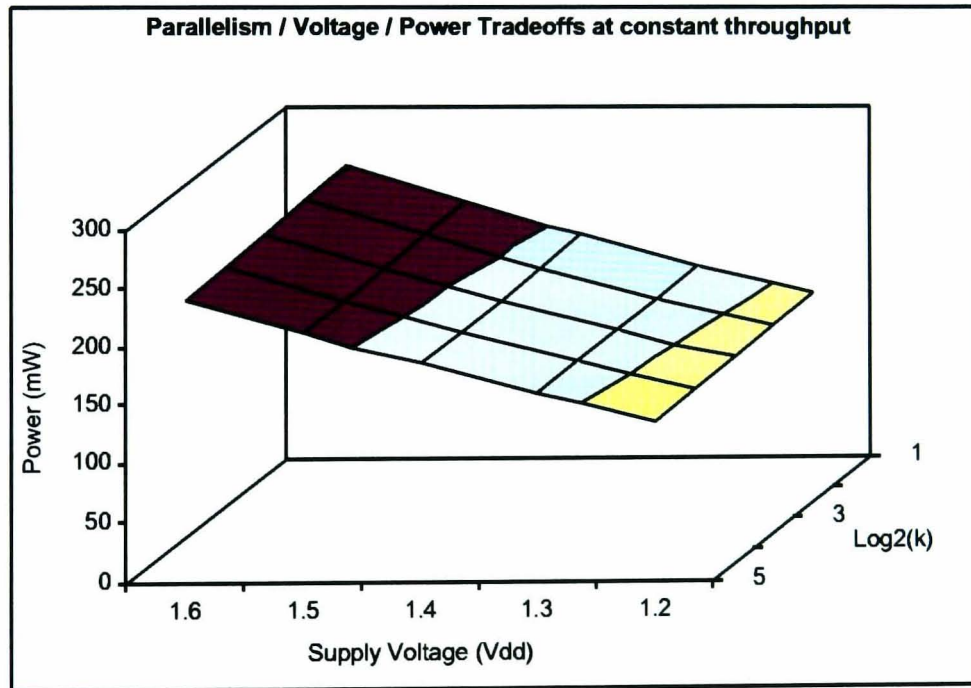


Figure 5.5: Power/Parallelism/Voltage tradeoff for the DCT core

It can be seen from the smaller cores that the I/O overhead results in higher power consumption for lower levels of parallelism. Since the extent of overhead is independent of the levels of parallelism, the effect is less pronounced as k approaches 7. On the other hand, the I/O overhead in the case of the DCT core is insignificant in comparison to the footprint of the core itself, resulting in a power curve largely in

line with theoretical expectations. In all cases, it can be clearly seen that the lowest power is achieved for the case of maximum parallelism and minimum voltage, thus validating our concept.

5.3 Conclusions

A systematic and empirical study of the tradeoffs between degree of parallelism, threshold voltage and power consumption under constant throughput conditions has been explored in this chapter. Proof of concept using suitable benchmarks of different orders of magnitude complexity: adder, multiplier and DCT has been presented. The preliminary results that have been obtained are very promising and this seems to be an increasingly important area of research that FPGA manufacturers themselves are studying seriously [109]. Significant scope for future work in this area exists.

Power reduction through performance enhanced voltage scaling has been discussed in this chapter. In the next chapter, high level power modelling of FPGA based designs is presented.

Chapter 6

Functional Level Power Analysis and Modelling

6.1 Introduction

Hardware implementations of power aware applications necessitate the design space exploration to realise an optimal solution. Design improvement is typically an iterative process that must take into account optimisation strategies at all feasible levels of abstraction. The most effective strategies must then be selected by analysing the impact of the different choices on a level-by-level basis, instead of just at the very end of the flow. This enables us to shorten the design flow. However, it requires the development of power modelling tools that provide reliable estimates of the power and energy metrics, to enable the designer to make the right design choices while optimising the core.

To overcome the limitations of existing work described in Chapter 2, a novel power modelling technique called FLPAM has been proposed. Each individual component of DP, i.e. Clock Power (CP), Signal Power (SP), Logic Power (LP), InPut Power (IpP) and Output Power (OP) is measured separately, and modelled individually. The power models are obtained by performing non linear regression analysis on system variables followed by multivariate parameter optimisation strategies.

The rest of this chapter is organised as follows. Brief introductions into the principles and mathematical basis behind FLPAM are presented in Sections 6.2 and 6.3

respectively. FLPAM results for various cores are discussed in Section 6.4. An objective comparison of various aspects of FLPAM with existing modelling techniques is presented in Section 6.5. Concluding remarks are provided in Section 6.6.

6.2 FLPAM: A Brief Introduction to Underlying Concepts

The underlying concept is to build a mathematical model that incorporates all the system variables, enabling the user to perform high level estimation of the power and energy metrics of the core for a given set of parameters early on in the design cycle itself. The steps involved in building the power model are as follows:

1. Create a power chart by measuring power for each individual component of DP, i.e. signal, logic, clock, I/O;
2. Identify all variables in the system, including user customisable ones, and internal system parameters, eg. frequency, vector length, area and voltage;
3. Deduce the order and number of terms in each equation from the logic resources used and the various parameters involved in the design. In the case of FPGAs, each individual component of power is modelled separately, and all the models are added together to yield a global power model. The order of the equation and the constant coefficients associated with each variable are unknown parameters. This step is essentially about choosing a model;
4. Derive the coefficients for each individual power component by performing non-linear regression analysis on the data in power chart for fitting the system variables;
5. Optimise the design by iterative analysis of the influence of design modifications on power and energy metrics until convergence has been achieved; and
6. Determine optimum operational parameters of the final model through constrained multivariate techniques.

The most significant difference between existing modelling methods and FLPAM is that our proposed solution is a high level modelling technique, and is used for modelling cores rather than the general fabric of the FPGA or the ASIC. This has a number of advantages. The FLPAM methodology scales very well with changes in platforms, and even prototyping technologies. Advances in the device technology of the prototyping platform naturally result in lower power and energy consumption metrics. Although this results in different constant coefficients for each platform, it is important to highlight that the model itself remains unchanged. The other key differentiator lies in the fact that the FLPAM tool is used by the core developer to optimise and model the core whereas it provides the system designer with the information that enables him to select the optimum set of core parameters to achieve the performance budgets; and even to analyse if the budgets are realistic in the first place. Low level power estimation techniques, on the other hand assume that core development and core deployment are integrated, and that the designer performs both the tasks. The FLPAM methodology has been successfully incorporated into a proposed design flow presented in Fig. A.3 for obtaining power and energy efficient implementations of FPGA based designs. A number of low power cores for various applications including different DOTs, CSC, FRAT, FRIT, etc have been successfully modelled and optimised using a combination of FLPAM and the proposed design flow.

6.3 Mathematical Basis behind FLPAM

In our proposed methodology, the mathematical model is derived as follows. We define the instantaneous DP dissipation of an FPGA based implementation as:

$$P_t = V_{cc} \cdot k \cdot f \cdot \delta_t \cdot A \quad (6.1)$$

where V_{cc} is the supply voltage, k is the constant of proportionality, f is the frequency of operation, δ_t is the instantaneous activity rate and A is the area occupied in slices. The average power dissipated in one computational cycle P_{cycle} is then given by:

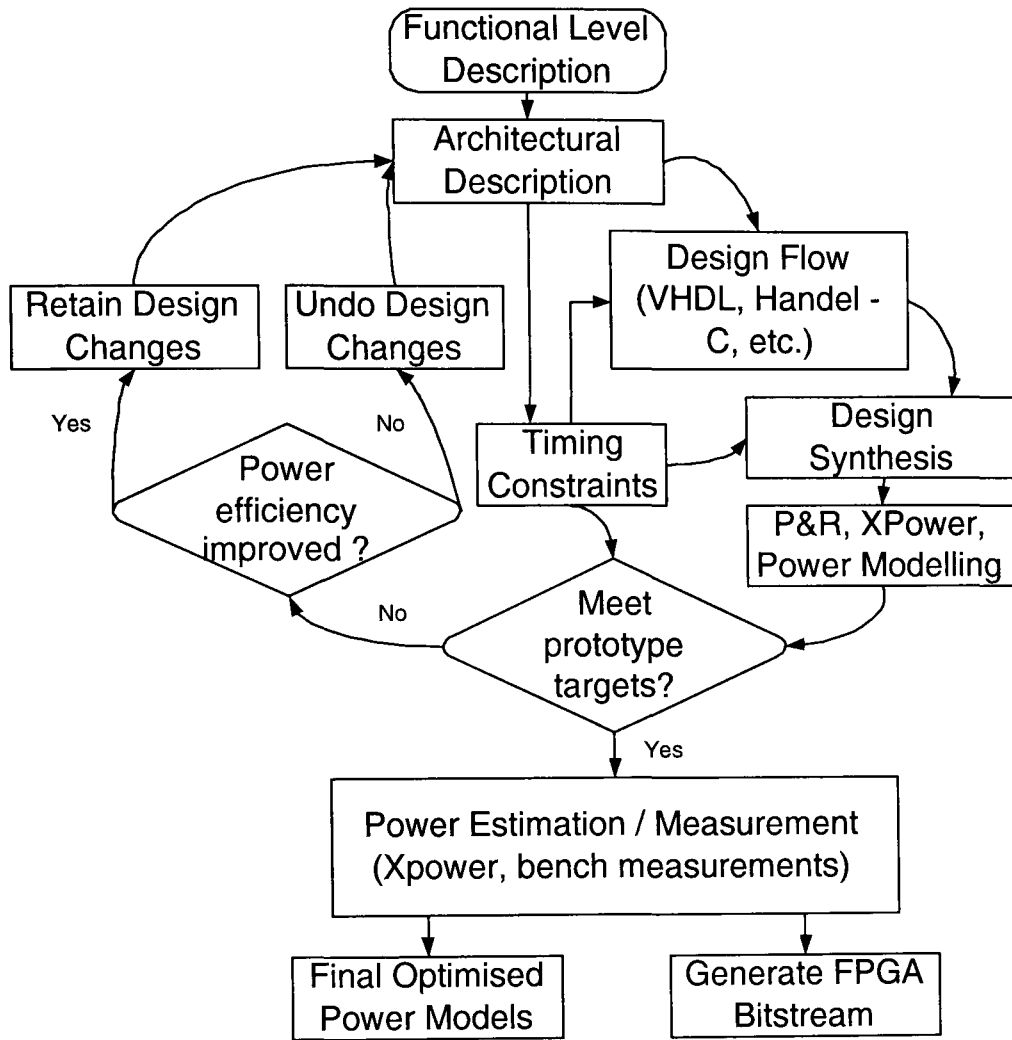


Figure 6.1: Proposed design flow for FLPAM based power and energy optimised design of FPGA cores

$$P_{cycle} = \frac{\int P_t \cdot dt}{T} \quad (6.2)$$

where T is the number of clocks required to complete one computational cycle. DP of SRAM based FPGAs comprises I/O, clock, logic and SP. Expanding Eq. 6.2, we get:

$$P_{cycle} = \frac{\int \sum_d P_{td} \cdot dt}{T} \quad (6.3)$$

where the sample space of d is given by $d = \{input, output, clock, signal, logic\}$. For a complex design with B sub-blocks, activity rates and area occupied must be computed individually for each block in order to derive accurate models. Taking into account each block separately, and substituting

Eq. 6.1 in Eq. 6.3, we get:

$$P_{cycle} = \frac{V_{cc} \cdot f \cdot A}{T} \sum_{b=1}^B \int_T \sum_d k_{db} \cdot \delta_{tb} \cdot dt \quad (6.4)$$

Each component of DP is modelled individually. By separating the models and rearranging the orders of integration and summation, we get:

$$P_{cycle.d} = \frac{V_{cc} \cdot f \cdot A}{T} \sum_{b=1}^B k_{db} \int_T \delta_{tb} \cdot dt \quad (6.5)$$

For an FPGA based architecture consisting of B sub-blocks, let us consider that the signal transition probability at the output of each block at instant t to be defined by a function of present and previous inputs as follows:

$$\delta_{tb} = \phi_b(i_b(t), i_b(t-1)) \quad (6.6)$$

where i_b is the *set* of inputs applied to block b that depends on the position and interconnection of block b in the Data Flow Graph (DFG) of the architecture in relation to other nodes. Since we are discussing synchronous FPGA design, in general, for node b , the term i_b is defined as:

$$i_b = \{o_{1-(b-1)}, o_{2-(b-1)}, \dots, o_{k-(b-1)}\} \quad (6.7)$$

where the input node representing the block under consideration is connected to the output of k nodes at a higher level of hierarchy in the DFG. An example illustrating this point pictorially is presented in Fig. 6.2. Substituting Eq. 6.6 in Eq. 6.5, we get:

$$P_{cycle.d} = \frac{V_{cc} \cdot f \cdot A}{T} \sum_{b=1}^B k_{db} \int_T \phi_b(i_b(t), i_b(t-1)) \cdot dt \quad (6.8)$$

Since we are interested in constructing a high level model, average power dissipation independent of signal statistics needs to be estimated. To simulate a random input distribution, out of phase maximum length Linear Feedback Shift Registers (LFSRs) are used to stimulate the input ports of the architecture being modelled. The average

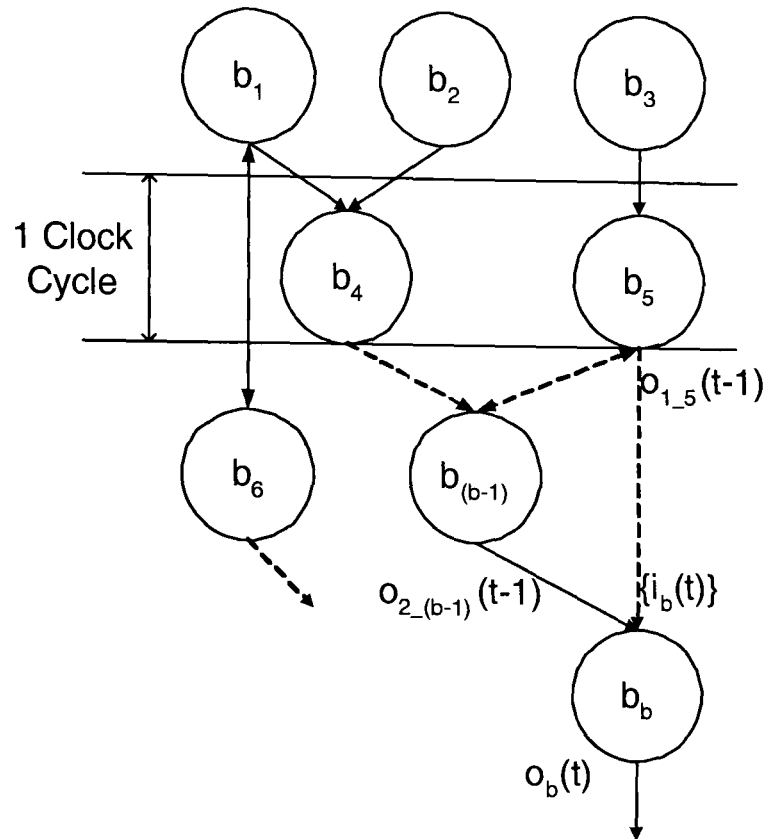


Figure 6.2: A generic Data Flow Graph indicating all parameters

power dissipated over a full cycle of LFSR operation is determined. If the model we construct consists of z parameters, where each parameter is represented by the symbol s , the model can be described in general as a function of s . Our model can be defined to be equivalent to the power equation as follows:

$$\Psi_d(s_1, s_2, \dots, s_z) = \frac{V_{cc} \cdot f \cdot A}{nT} \sum_n \sum_{b=1}^B k_{db} \cdot \Delta_b \quad (6.9)$$

where Δ_b is the average activity rate for each block and n is the length of the maximal LFSR sequence. Estimates of each individual component of DP are obtained using Xilinx Xpower [11], and the only unknown terms in Eq. 6.5 are the model coefficients and scaling coefficients k_{db} . These coefficients are determined by means of an adaptive choice of the model Hessian. The algorithm is essentially a combination of Gauss-Newton and Levenberg-Marquardt methods. The fundamental mathematical concept that is used for determining the coefficients is non-linear regression, and is implemented using NLREG [164]. This process yields an intermediate model which is used as a basis for iterative design optimisation.

Although the parameters of the function $\Psi_d(s_1, s_2, s_3, \dots, s_z)$ can be continuous or

discrete, let us assume for a moment that Ψ is continuous and fully differentiable in all z directions. Given $\Psi_d(\bar{s}) : R^z \rightarrow R$ is a smooth function we can now apply Newton's method of multivariate optimisation. An extrema or saddle point is obtained when:

$$\nabla \Psi_d(\bar{s}) = 0 \quad (6.10)$$

A point is said to converge to optimisation when the Eigenvalues of the Hessian of the function are positive definitive. The Hessian matrix contains the second derivatives in higher dimensions and is denoted as:

$$H = \begin{bmatrix} \frac{\partial^2 \Psi_d}{\partial s_1 \partial s_1} & \cdots & \frac{\partial^2 \Psi_d}{\partial s_1 \partial s_z} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 \Psi_d}{\partial s_z \partial s_1} & \cdots & \frac{\partial^2 \Psi_d}{\partial s_z \partial s_z} \end{bmatrix} \quad (6.11)$$

Let us denote:

$$\bar{g}_k = \nabla \Psi_d(\bar{s}_k) \quad (6.12)$$

$$H_k = H(\bar{s}_k) \quad (6.13)$$

Approximating $\Psi_d(s)$ around s_k by a quadratic function, we get:

$$Q(\bar{s}_k + \bar{m}) = \Psi_d(\bar{s}_k) + \bar{m} \cdot \bar{g}_k + \frac{1}{2} \bar{s}^T H_k \bar{s} \quad (6.14)$$

where m is the step to minimise the quadratic. By taking:

$$H'_k = H_k + \mu_k I \quad (6.15)$$

where μ_k is large enough to make \bar{H} positive definite, which is a requirement for locating a local minima. \bar{m}_k is determined by the following expression:

$$-\bar{H}_k \bar{m}_k = \bar{g}_k \quad (6.16)$$

The transformation presented in Eq. 6.16 guarantees that \bar{m}_k decreases, i.e. $\Psi_d(s_k + ts_k) < \Psi_d(s_k)$ if t is chosen to be small enough.

$$-\bar{g}_k \cdot \bar{m}_k = \bar{g}_k^T \bar{H}_k^{-1} \bar{g}_k > 0 \quad (6.17)$$

We need to find μ_k such that $\mu_k = 0$ for H_k to be positive definite, and $H_k + \mu_k I$ if not. This can be done by calculating the eigenvalues (λ) of H_k and setting $\mu_k = 0$ if $\lambda_{min} > 0$ and $\mu_k = -\lambda_{min} + \delta$ otherwise. Alternatively, μ_k can be determined by Cholesky decomposition. Convergence to optimum solutions of the model can be obtained by the following conditions:

$$\frac{\Psi_d(\bar{s}_k) - \Psi_d(\bar{s}_k + t_k \bar{m}_k)}{t} \geq -\alpha \bar{g}_k \cdot \bar{m}_k \quad (6.18)$$

$$-\bar{m}_k \cdot \nabla \Psi_d(\bar{s}_k + t_k \bar{m}_k) \leq -\beta \bar{g}_k \cdot \bar{m}_k \quad (6.19)$$

where $0 < \alpha < 1$ and for some $\beta \in (\alpha, 1)$.

It is important to highlight that the accuracy of the power models derived are relative to the accuracy of the power measurements / estimates obtained, and do not indicate the absolute accuracy by themselves. In the case of models built on XPower estimation data, we must take into account the accuracy of the XPower estimate as well, while calculating the absolute accuracy of the models. For models based on chip power measurements, the accuracy of the models is limited to the resolution of the power supply and measuring tools used.

6.4 FLPAM Applied to Various Cores

A number of cores that have been developed in-house have been successfully optimised and modelled using FLPAM and the proposed integrated design flow. The power macromodels obtained, and statistical properties of the corresponding models for each of the cores that have been modelled are described in the following subsections.

6.4.1 Case Study I: Modelling the CSC Core

The CSC core for FPGAs is one of the simpler, but most efficient cores in the FMAT library. Full details about the design and implementation of this core can be obtained from [22]. The original architecture already exploits a number of optimisation strategies including the use of DA, architectural techniques such as pipelining and parallelism. The only further opportunity for optimisation that has been exploited is the use of manual placement and routing of critical nets and manual pin assignment for the designs has been performed using Xilinx PACE and Xilinx Floorplanner [11]. This process yields a compact and optimised design with short nets, and serves two important purposes. Firstly, short nets have lesser propagation delay, and up to 25% gains in maximum frequency have been achieved. Second, short nets have lesser parasitic capacitance and DC load, and therefore dissipate lesser power than long nets. Manual pin assignment also enables us to locate the I/O pads close to the design area, further aiding the above two criteria. Details of implementation results are available in [165].

CSC Power Models

Power modelling is performed for implementations on the XCV50E and the XCV2000E FPGA chips. Differences in implementation results obtained are due to different chip topologies (area, pin distribution, etc.). However, since the design remains unaltered, the same model can be used to define the power consumption for both implementations. The corresponding models are derived by performing non-linear regression analysis on the power data obtained. On iterating the regression until convergence is achieved, the values of the scaling coefficients in the models can be determined. By back substitution of these values back into the model, a global equation that defines the power consumption of the system for any given combination of system parameters has been derived. The power measurements data on which the models are based are graphically represented in the following paragraph subsections.

Model Parameters, Coefficients and Accuracy

CP in FPGAs depends on the distribution of the clock nets (which depends on the chip area over which the design is spread out). Other factors influencing CP are chip frequency f , and voltage v . SP is proportional to the number and length of nets over which signal switching occurs. It also depends on the voltage levels between which the switching occurs, as well as the frequency of switching. LP consumption is a function of the number of slices occupied, chip frequency and voltage. The IpP depends on the number of input pins in the design, the vector length, wordlength, chip voltage, and input frequency. OP depends on the number of output pins in the design, the output vector length, wordlength, output voltage V_{cco} and frequency f . Since the architecture is completely pipelined and accepts once pixel per clock cycle, the wordlength W and vector length V do not influence I/O power. They are constants for the given design, and are automatically absorbed into the corresponding scaling coefficients. Based on an empirical analysis of the power measures obtained, it is observed that a simple power model is adequate to accurately represent each component. In general, the power model for each component, i.e. clock, signal, logic, input and output is defined as:

$$Power = c \times v^2 \times f \quad (6.20)$$

where c is the corresponding scaling coefficient for the power model of the component under consideration.

The final estimates of the scaling coefficients obtained are presented in Table 6.1.

Table 6.1: Final estimates of the scaling coefficients

	Constant Coefficient (c)	
	XCV50E	XCV2000E
Clock	0.0717223024	0.0819207953
Input	0.100675004	0.100675004
Logic	1.70921757	1.47180471
Output	0.857998163	0.857998163
Signal	0.41140388	0.458445086

Observations and Analysis

From the power model graphs, it can be seen that the power is directly proportional to frequency f for clock, signal, logic and input. It is worth mentioning that the input and OP measures for both FPGA chips is identical. This is because they depend only on frequency and the characteristics of the IBUFS and the OBUFS of the Virtex-E FPGA series. On the other hand, the actual placement and routing of the design is influenced by the topology of the FPGA, accounting for the small differences in clock, signal and LP for the two FPGA chips under consideration.

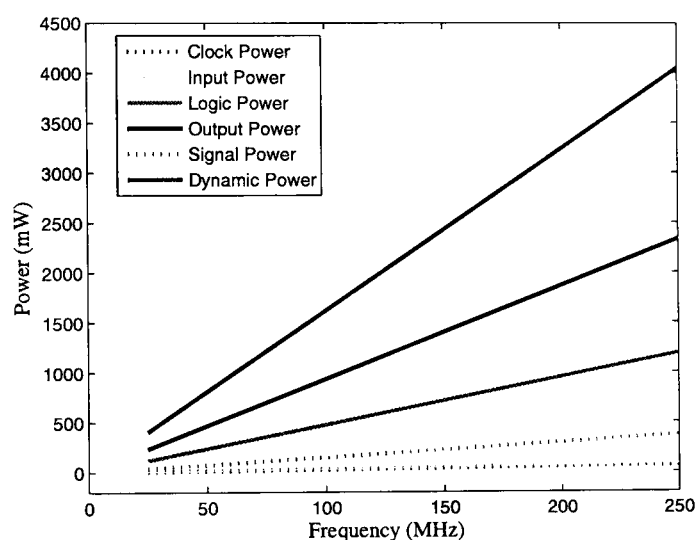


Figure 6.3: CSC Power diagram

6.4.2 Case Study II: Modelling the FHT

The FHT core in the FMAT core library presents a highly optimised and efficient one that takes advantages of a number of algorithmic and architectural techniques to obtain competitive implementation results. The design and implementation details of this core were first reported in [16]. Performance metrics were further improved through low level optimisation techniques including manual place and route of critical nets and manual pin assignment, details of which have been presented in [166] and Chapter 3. FLPAM model details for this core are presented in the following subsections.

Modelling Results

The corresponding models are derived by performing non-linear regression analysis on the power data obtained. On iterating the regression until convergence is achieved, the values of the scaling coefficients in the models can be determined. By back substitution of these values back into the model, a global equation that defines the power consumption of the system for any given combination of system parameters has been derived. The power measurements data on which the models are based are graphically represented in the following subsections:

Area Model

Area occupied by the design is one of the important system variables in the power model. Signal propagation takes place only along those nets that fall under the area of the placed and routed design. Hence, CP, LP and SP all depend on the area occupied. Area occupied by the FHT core in terms of number of slices depends on two components in the architecture: control logic area and the ROM area. The area occupied by ROM increases exponentially with vector length. It is also proportional to logarithm of vector length. The area occupied by the control logic is proportional to the vector length.

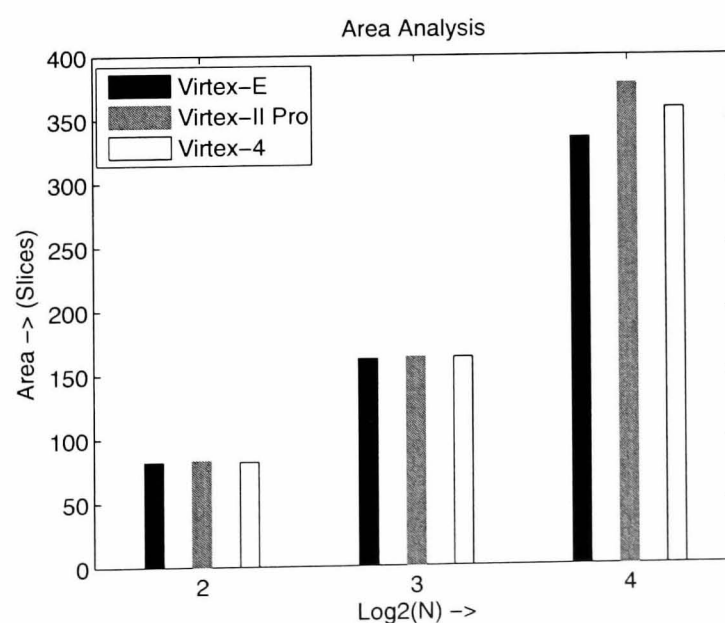


Figure 6.4: Area occupied for different transform lengths : $N = 4, 8, 16$

Based on data presented graphically in Fig. 6.4, the area model is described as follows:

$$TA = \alpha1' \cdot (ROMArea) + \alpha2' \cdot (ControlLogicArea) \quad (6.21)$$

$$TA = \alpha1 \cdot 2^{N/2} \cdot (\log_2 N + 1) + \alpha2 \cdot N + \alpha3 \quad (6.22)$$

where TA is the Total Area occupied, $\alpha1'$, $\alpha2'$ are constant coefficients, and $\alpha1$, $\alpha2$, $\alpha3$ are the scaling coefficients in the area model.

CP Model

CP in FPGAs depends on the distribution of the clock nets (which depends on the chip area over which the design is spread out). Other factors influencing CP are chip frequency f , and voltage v .

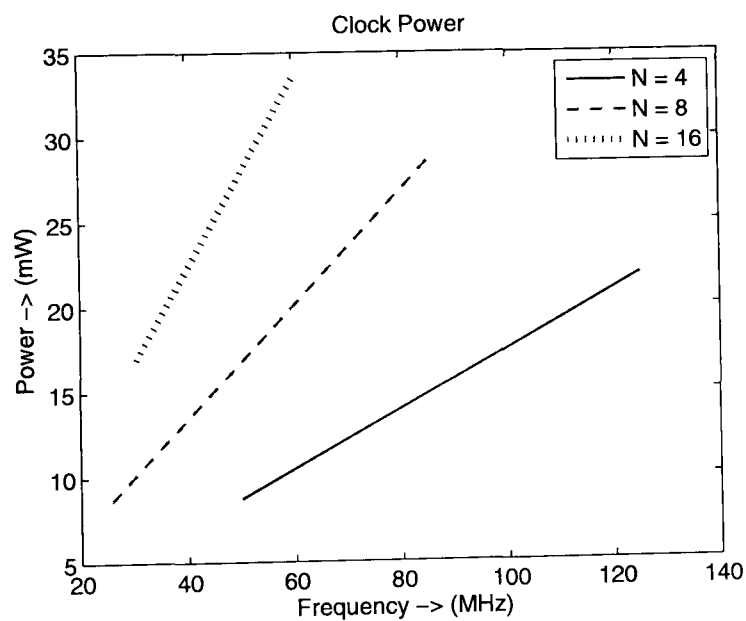


Figure 6.5: CP for $N = 4, 8, 16$ at different frequencies

Based on CP data presented graphically in Fig. 6.5, the corresponding model is defined as follows:

$$CP = c1 \cdot v^2 \cdot TA^{c2} \cdot f + c3 \cdot f + c4 \quad (6.23)$$

where, c_1, c_2, c_3 and c_4 are scaling coefficients in the CP model.

SP Model

SP is proportional to the number and length of nets over which signal switching occurs. It also depends on the voltage levels between which the switching occurs, as well as the frequency of switching. On observation and interpretation of the power data presented in Fig. 6.6, the best model describing the SP for FHT core is described in Eq. 6.24. s_1, s_2, s_3 and s_4 are scaling coefficients in the SP model.

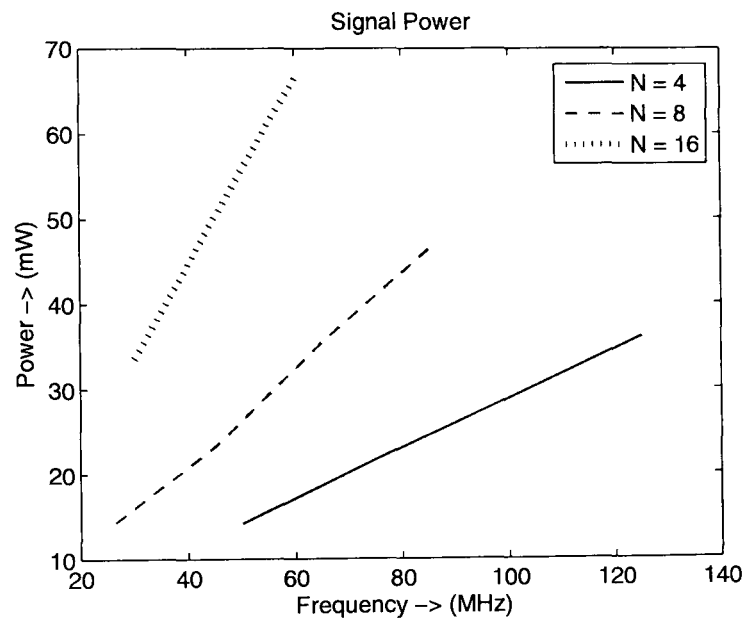


Figure 6.6: SP for $N = 4, 8, 16$ at different frequencies

$$SP = s_1 \cdot v^2 \cdot TA^{s_2} \cdot f + s_3 \cdot f + s_4 \quad (6.24)$$

LP Model

LP is a function of the number of slices occupied, chip frequency and voltage. Taking into consideration these parameters, and the LP measurement data shown in Fig. 6.7 SP model for both architectures is defined in Eq. 6.25 where l_1, l_2, l_3 and l_4 are scaling coefficients in the LP model.

$$LP = l_1 \cdot v^2 \cdot TA^{l_2} \cdot f + l_3 \cdot f + l_4 \quad (6.25)$$

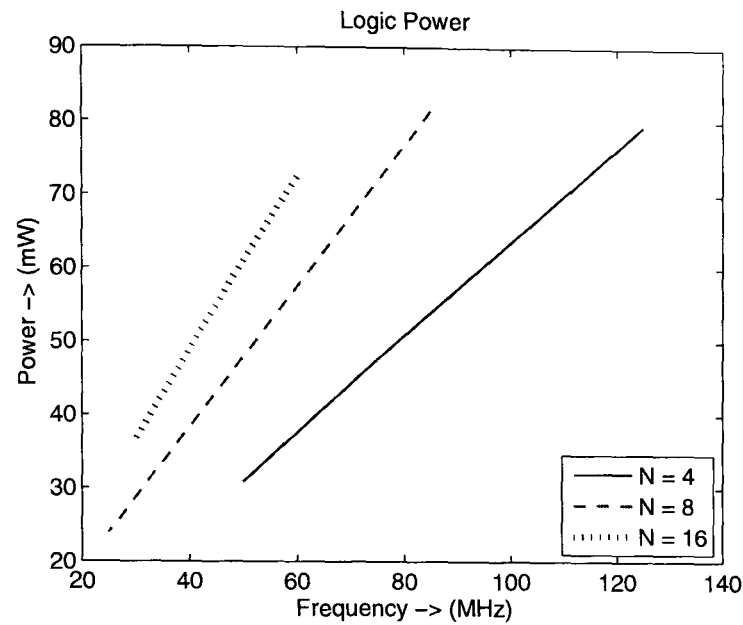


Figure 6.7: LP for $N = 4, 8, 16$ at different frequencies

IpP Model

IpP depends on the number of input pins in the design, the vector length, wordlength, chip voltage, and input frequency. The estimated values of IpP are graphically presented in Fig. 6.8, and the corresponding model is described in 6.26. i_1, i_2 and i_3 are scaling coefficients in the IpP model. It is worth mentioning that although the input wordlength W also affects IpP, it is a constant, and is automatically absorbed into the scaling coefficients.

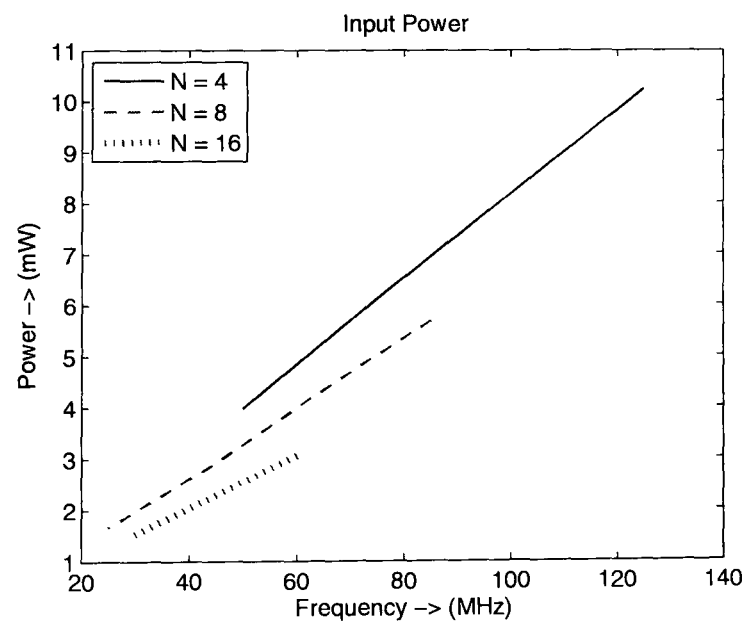


Figure 6.8: IpP for $N = 4, 8, 16$ at different frequencies

$$I_p P = i_1 \cdot v^2 \cdot f \cdot N^{i_2} + i_3 \quad (6.26)$$

OP Model

OP depends on the number of output pins in the design, the input vector length, wordlength, output voltage V_{cco} , and frequency f . The estimated values of OP are graphically presented in Fig. 6.9, and the corresponding model is described in 6.27 where U is the wordlength of the output vector, o_1, o_2, o_3 and o_4 are scaling coefficients in the OP model. It is worth mentioning that the output wordlength U is a function of input vector length N , and this is reflected in the model.

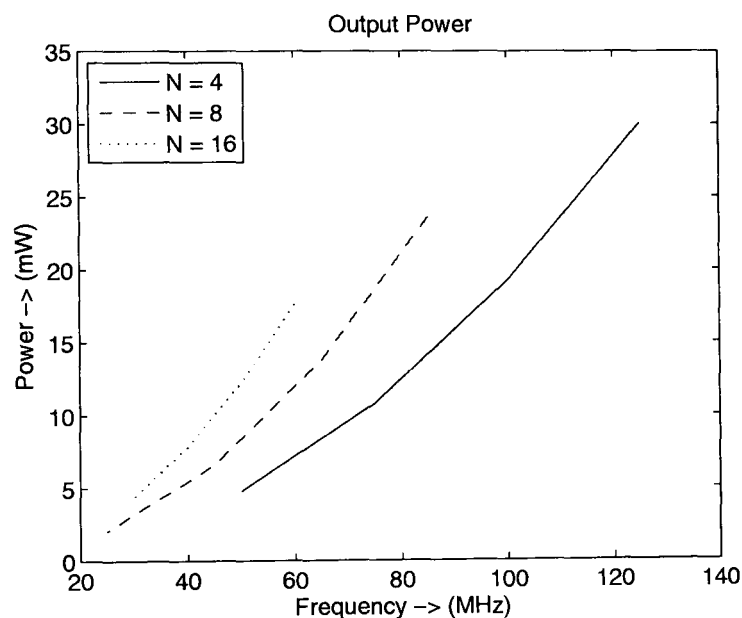


Figure 6.9: OP for $N = 4, 8, 16$ at different frequencies

$$OP = (v^2 \cdot N) \cdot f^2 \cdot (o_1 \cdot N + o_2 \cdot U + o_3) + o_4 \quad (6.27)$$

The coefficient values of the area and power models are obtained by performing non linear regression until convergence, and are presented in Table 6.2.

Observations and Analysis

From the power model graphs, it can be seen that the power is directly proportional to frequency f for clock, signal, logic and input. OP is proportional to f^2 .

Table 6.2: Coefficient values for the area and power models (Virtex-E Platform)

	Coeff. 1	Coeff. 2	Coeff. 3	Coeff. 4	
Area	1.53E+01	9.05E-02	1.65E+01	NA	
CP	2.89E-01	1.35E-01	-1.52E+00	-1.46E-03	
SP	3.77E-03	7.73E-01	-7.99E-02	-3.63E-01	NA: Not Applicable
LP	1.03E+02	1.18E-03	-3.35E+02	1.13E-01	
IpP	2.26E-02	9.81E-01	-3.76E-01	NA	
OP	-1.86E-08	-5.79E-06	1.54E-04	7.11E-03	

6.4.3 Case Study III: Modelling the FRAT

Modelling of a reference and novel parallel FRAT core whose architectural and implementation details have been published in [146] and explained in detail in Chapter 4 is presented in this section. The corresponding parameters that influence the choice of the model have been explained in the following subsections.

Area Model

The area occupied by the design in terms of number of slices depends on two components in the architecture: control logic area and the ROM area. The area occupied by ROM increases exponentially with vector length. It is also proportional to logarithm of vector length. This term is represented by the variables associated with coefficient α_1 in Eq. 6.28 and 6.29 respectively. The area occupied by the control logic and arithmetic blocks (shifters, adders etc.) is proportional to the vector length, and is represented by the variables associated with coefficient α_2 . Coefficient α_3 is introduced to represent unrepresented additional areas of the architectures; and to balance the models effectively. The term $p(p + 1)$ in Eq. 6.28 represents the input and output buffers. On the other hand, there is only an output buffer in the proposed parallel cyclic architecture. This, along with the systolic array are represented by the terms p^2 and p in Eq. 6.29. Based on these observations, the area models for the reference and parallel architectures are described in Eq. 6.28 and 6.29 respectively:

$$TA_{Ref} = (\alpha_1 \cdot p(p + 1) + \alpha_2) \cdot (\text{ceil}(\log_2(255 * p))) + \alpha_3 \quad (6.28)$$

$$TA_{Par} = \alpha1 \cdot p^2 + \alpha2 \cdot p \cdot \text{ceil}(\log_2(255 * p)) + \alpha3 \quad (6.29)$$

where TA_{Ref} and TA_{Par} are the Total Area occupied by the reference and parallel architectures respectively, $\alpha1, \alpha2$ and $\alpha3$ are the scaling coefficients in the area model. It has been observed that there are minor variations in the number of occupied slices. This is because of differences in the number of slices per CLB in different platforms. Correspondingly, the logic capacity of each CLB and interconnect structure between CLBs also differ. Placement and routing takes into account these variations in the FPGA fabric and results in differences in the area metrics for the same architecture on different platforms. Area occupied by the design is one of the important system variables in the power model. Signal propagation takes place only along those nets that fall under the area of the placed and routed design. Hence, CP, LP and system power all depend on the area occupied.

CP Model

CP in FPGAs depends on the distribution of the clock nets (which depends on the chip area over which the design is spread out). Other factors influencing clock power are chip frequency f , and voltage v . Based on these parameters, the same CP model holds true for both architectures. From the power data pictorially represented in Fig. 6.10, an appropriate model has been selected and is described in Eq. 6.30. $c1, c2$ and $c3$ are scaling coefficients in the CP model.

$$CP = c1 \cdot v^2 \cdot TA_{Ref/Par} \cdot f + c2 \cdot f + c3 \quad (6.30)$$

SP Model

SP is proportional to the number and length of nets over which signal switching occurs. It also depends on the voltage levels between which the switching occurs, as well as the frequency of switching. Taking into consideration these parameters and the SP measurement data shown in Fig 6.11, the SP model for both architectures is

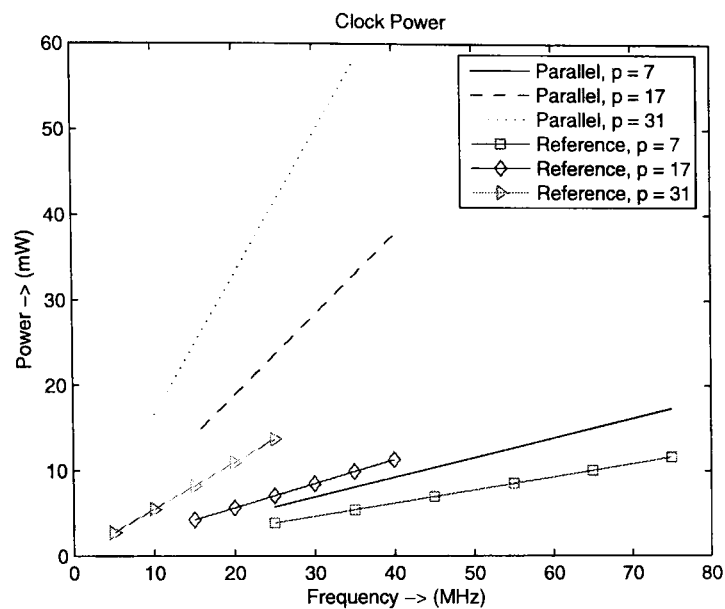


Figure 6.10: CP chart for the FRAT core

defined in Eq. 6.31 where s_1, s_2, s_3 and s_4 are scaling coefficients.

$$SP = s_1 \cdot v^2 \cdot TA_{R/P}^{s_2} \cdot f + s_3 \cdot f + s_4 \quad (6.31)$$

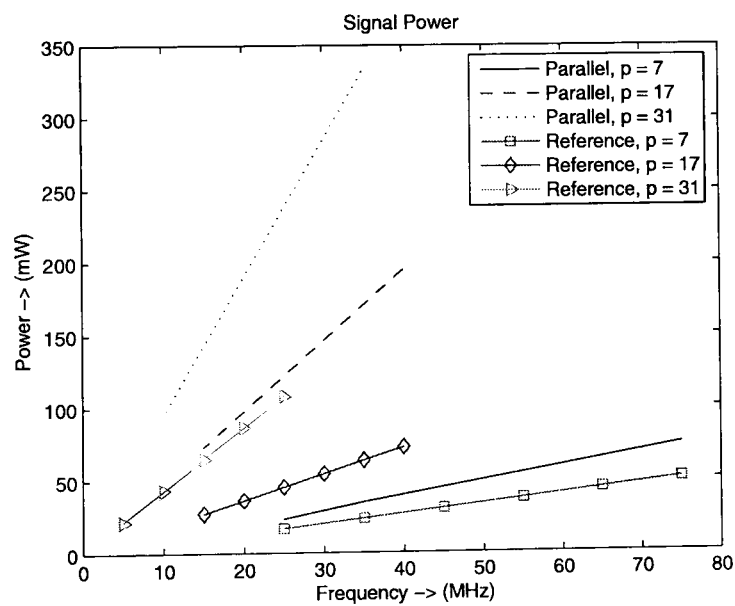


Figure 6.11: SP chart for the FRAT core

LP Model

LP dissipation is a function of the number of slices occupied, chip frequency and voltage. On observation and interpretation of the power data presented in 6.12, the

best models describing the LP for the reference and parallel FRAT architectures are as described in Eq. 6.32 and 6.33 respectively. $l1, l2, l3$ and $l4$ are scaling coefficients in the LP models.

$$LP_R = l1 \cdot v^2 \cdot TA_R^{l2} \cdot f + l3 \cdot f + l4 \quad (6.32)$$

$$LP_P = l1 \cdot v^2 \cdot TA_P \cdot f + l2 \cdot f + l3 \quad (6.33)$$

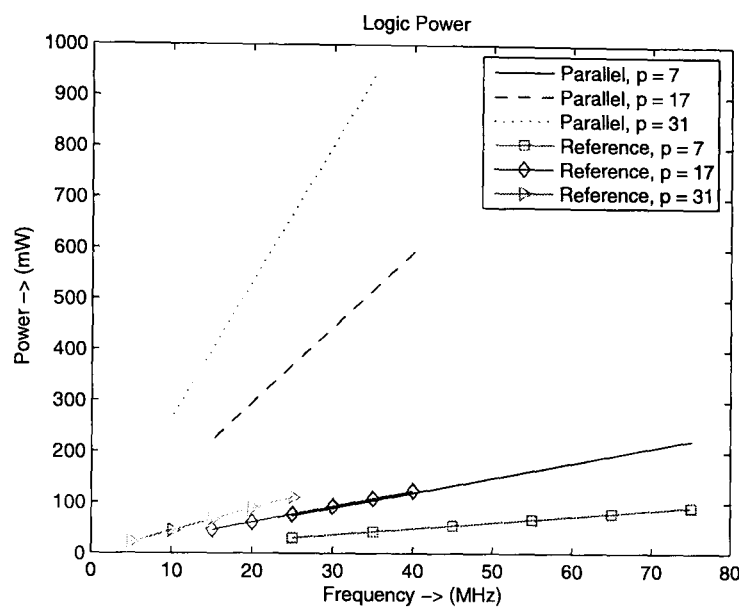


Figure 6.12: LP chart for the FRAT core

IpP Model

The IpP model depends on the number of input pins in the design, the block size, chip voltage, and input frequency. The corresponding model that satisfies the curve fitting requirements of both architectures for the data presented in Fig. 6.13 is defined in Eq. 6.34 where $i1, i2$ and $i3$ are scaling coefficients. It is interesting to note that the input buffers Virtex-4 platform dissipate no power, and hence the coefficients in the corresponding models for this platform are zero.

$$IpP = i1 \cdot v^2 \cdot f \cdot (p^2)^{i2} + i3 \quad (6.34)$$

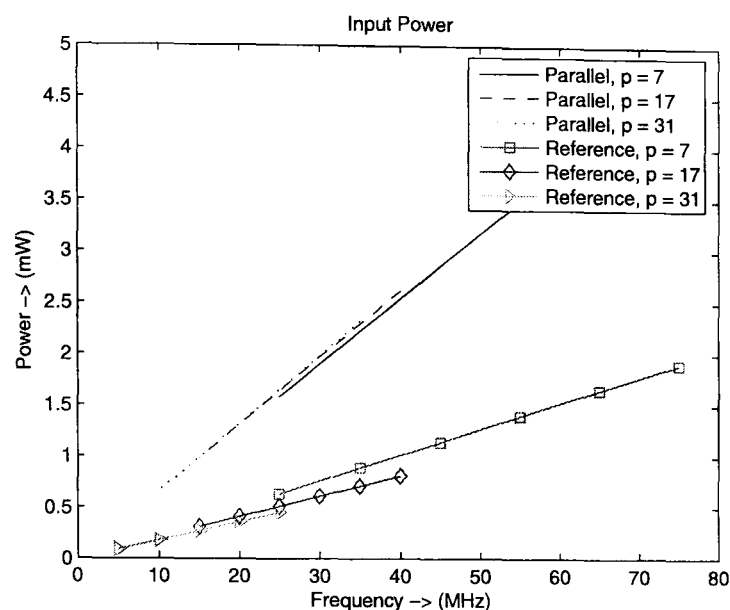


Figure 6.13: IpP chart for the FRAT core

OP Model

The OP model depends on the number of output pins in the design, the output block size, output voltage V_{CCO} , and frequency f . Based on measurement data presented in Fig 6.14 the corresponding model is described in Eq. 6.35 where p is the block size, of the output vector, $o1$, $o2$ and $o3$ are scaling coefficients in the output power model. It can be observed from the graph that the OP is nearly identical for parallel architectures $p = 17$ and $p = 31$. This is because they both have the same number of output pins and hence dissipate same amounts of power at the output pads for the same value of frequency.

$$OP = o1 \cdot (v^2 \cdot N) \cdot f^2 \cdot \text{ceil}(\log_2(255 \cdot p)) \cdot p \cdot (p + 1)^{o2} + o3 \quad (6.35)$$

The coefficient values of the area and power models are obtained by performing non linear regression until convergence, and are presented in Tables 6.3 and 6.4.

FRAT FLPAM Model Accuracy Across Different FPGA Platforms

It is interesting to note that the models are platform independent, and differences in the chip fabric are accounted for by the model coefficients. The proportion of variance of the models (R^2) is presented in Table 6.5.

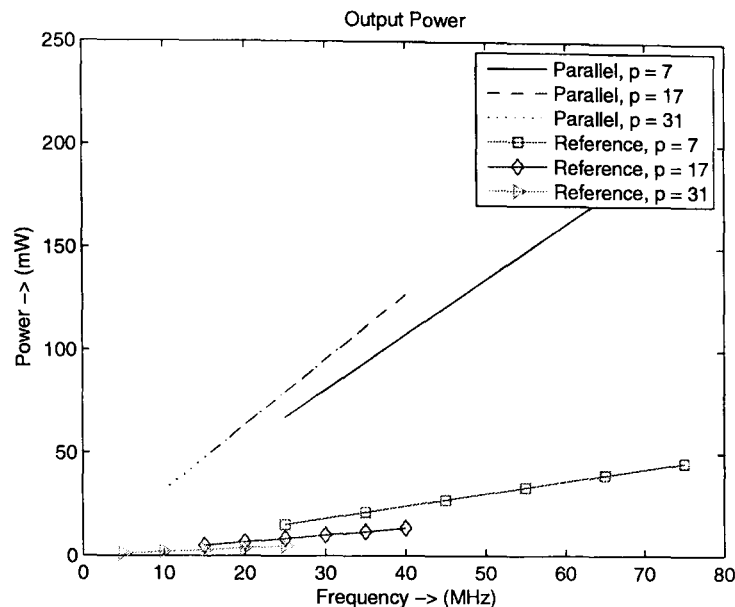


Figure 6.14: OP chart for the FRAT core

Table 6.3: Coefficient values for area and power models for the proposed FRAT reference architecture

Coeff.	Virtex-E	Virtex-II	Virtex-IV
$\alpha 1$	1.97E-01	2.22E-02	2.02E-02
$\alpha 2$	6.86E-01	2.87E+01	4.16E+01
$\alpha 3$	3.25E+01	-2.32E+02	-3.74E+02
$c 1$	2.95E-04	1.85E-04	5.79E-05
$c 2$	6.34E-02	6.45E-02	8.28E-03
$c 3$	-3.97E-02	1.93E+00	8.17E-03
$s 1$	4.45E-04	5.44E-05	3.90E-05
$s 2$	1.28E+00	1.59E+00	1.58E+00
$s 3$	1.80E-01	2.53E-01	6.73E-02
$s 4$	-5.27E-03	1.13E+01	3.39E-01
$l 1$	2.78E-03	1.85E-03	4.02E-04
$l 2$	4.04E-01	4.99E-01	3.25E-02
$l 3$	2.06E+00	1.68E+01	7.82E-01
$i 1$	-2.71E-07	-2.30E-07	0.00E+00
$i 2$	2.05E+00	2.04E+00	0.00E+00
$i 3$	2.47E+00	8.97E-01	0.00E+00
$o 1$	3.02E-02	-4.99E-07	-2.73E-07
$o 2$	-1.08E+00	1.60E+00	1.59E+00
$o 3$	-1.03E-01	2.08E+01	2.32E+01

Observations and Analysis

From the power models that have been derived, it can be clearly seen that the models are robust and can consistently provide good estimates of power dissipation within the design space for all platforms. This clearly shows the scalability

Table 6.4: Coefficient values for area and power models for the proposed FRAT parallel architecture

Coeff.	Virtex-E	Virtex-II	Virtex-IV
$\alpha 1$	8.80E-02	-2.54E-01	-7.48E-01
$\alpha 2$	5.31E-01	4.44E+00	7.77E+00
$\alpha 3$	2.57E+03	-8.48E+01	-3.39E+02
$c 1$	3.18E-05	2.76E-04	3.54E-05
$c 2$	7.75E-02	-1.60E-02	1.17E-02
$c 3$	3.09E+00	1.73E-01	1.79E-02
$s 1$	1.56E-04	1.20E-03	1.92E-04
$s 2$	1.12E+00	1.10E+00	1.18E+00
$s 3$	2.35E-01	1.61E-01	9.62E-02
$s 4$	1.33E+01	-2.15E-02	6.40E-04
$l 1$	1.61E+01	1.23E-03	1.13E-04
$l 2$	8.13E-03	1.11E+00	1.11E+00
$l 3$	-5.46E+01	1.85E-01	3.61E-02
$l 4$	6.83E+01	8.26E-01	9.11E-02
$i 1$	-8.80E-07	-9.40E-08	0.00E+00
$i 2$	1.66E+00	2.37E+00	0.00E+00
$i 3$	9.73E-01	1.96E+00	0.00E+00
$o 1$	-4.58E-06	-1.66E-09	2.35E-02
$o 2$	1.06E+00	3.49E+00	-1.08E+00
$o 3$	2.45E+01	1.42E+02	-6.41E-02

Table 6.5: FLPAM model accuracy for the optimised FRAT core implemented on different platforms

	Virtex-II		Virtex-E		Virtex-4	
	Arch 1	Arch 2	Arch 1	Arch 2	Arch 1	Arch 2
Clock	99.65%	96.82%	99.57%	97.65%	99.29%	98.96%
Logic	100.00%	93.00%	97.67%	88.92%	100.00%	94.96%
I/P	99.98%	98.72%	97.32%	99.14%	98.92%	99.03%
O/P	99.42%	99.96%	99.61%	99.22%	99.78%	99.26%
Signal	100.00%	99.93%	99.55%	100.00%	100.00%	99.68%

of FLPAM. The differences in the FPGA fabric are low level / circuit level characteristics, and should have no effect on the models since FLPAM is a high level modelling methodology. The architectural differences, and consequent variations in power consumption across different platforms are accounted by the coefficients in the models as shown in Tables 6.3 and 6.4.

6.4.4 Case Study IV: Modelling the FRIT

It has been seen from the previous section that the novel parallel, pipelined semi-systolic architecture for the FRAT core clearly outperforms the reference architecture in all key performance metrics, and particularly so in total energy per operation. Hence, this core has been used as the fundamental building block for the FRIT. As an exploratory step to judge the suitability of FLPAM across a different implementation platform, the ASIC implementation is modelled as well. Full details of the FRIT architecture and implementation can be obtained from Chapter 4.

Power Analysis

It is well known that the power consumption of any architecture implemented on FPGA is influenced by a number of factors such as clock frequency, design density (number of interconnects), activity rates, logic block and interconnect structure of the specific FPGA, power supply voltage levels and output loading. In the case of ASIC, power dissipation depends on a number of system variables that can also be modelled. The main difference between the two models is related to the fact in the case of FPGAs, the clock is routed to all LUTs irrespective of their activity. The details of both models have been presented in the following subsections.

Area Model

The area occupied by the design in terms of number of slices depends on two components in the architecture: control logic area and the ROM area. The area occupied by ROM increases exponentially with vector length. It is also proportional to logarithm of vector length. This term is represented by the variables associated with coefficient α_1 in Eq. 6.36. The area occupied by the control logic and arithmetic blocks (shifters, adders etc.) is proportional to the vector length, and is represented by the variables associated with coefficient α_2 . Coefficient α_3 is introduced to accommodate unrepresented additional areas of the architectures; and to balance the models effectively. The systolic array address dereferencing unit in the FRAT block is represented by the terms p^2 and p in Eq. 6.36. Based on these observations, the area model for the proposed FRIT architecture is described in Eq. 6.36:

$$TA = \alpha_1 \cdot p(p + 1) \cdot \lceil (\log_2(255 * p)) \rceil + \alpha_2 \cdot \lceil (\log_2(255 * p)) \rceil + \alpha_3 \quad (6.36)$$

where TA is the total area occupied by the proposed architecture, α_1, α_2 and α_3 are the scaling coefficients in the area model. It has been observed that there are minor variations in the number of occupied slices for different FPGA platforms. This is because of differences in the number of slices per CLB in different platforms. Correspondingly, the logic capacity of each CLB and interconnect structure between CLBs also differ. Placement and routing takes into account these variations in the FPGA fabric and results in differences in the area metrics for the same architecture on different platforms. Area occupied by the design is one of the important system variables in the power model. Signal propagation takes place only along those nets that fall under the area of the placed and routed design.

It is interesting to note that the very same area model holds true for both the FPGA platforms and the ASIC platform as well. The model equations are identical, and achieve 100% accuracy for all three platforms. The differences in the fabric of these different platforms are accounted for by the coefficients in the model, which naturally vary across different platforms. Additionally, another key difference is that the term TA in the above model corresponds to the area in terms of slices for the FPGA platforms, and μm^2 of silicon area for the ASIC platform. Clearly, the modelling methodology is high level as it is not influenced by low level details such as routing and circuit fabric. The coefficient details for these models are presented in Table 6.6.

Power Modeling for FPGA Platforms

The corresponding models for the FPGA platforms are derived by performing non-linear regression analysis on the power data obtained. On iterating the regression until convergence is achieved, the values of the scaling coefficients in the models can be determined. By back substitution of these values back into the model, a global equation that defines the power consumption of the system for any given combination of system parameters has been derived. The power measurements data on which the models are based are graphically represented in the following subsections.

CP Model

CP in FPGAs depends on the distribution of the clock nets (which depends on the chip area over which the design is spread out). Other factors influencing clock power are chip frequency f , and voltage v . Based on these parameters, the same CP model holds true for both FPGA platforms. From the power data pictorially represented in Fig. 6.15, an appropriate model has been selected and is described in Eq. 6.37. c_1, c_2 and c_3 are scaling coefficients in the CP model.

$$CP = c_1 \cdot v^2 \cdot TA \cdot f + c_2 \cdot f + c_3 \quad (6.37)$$

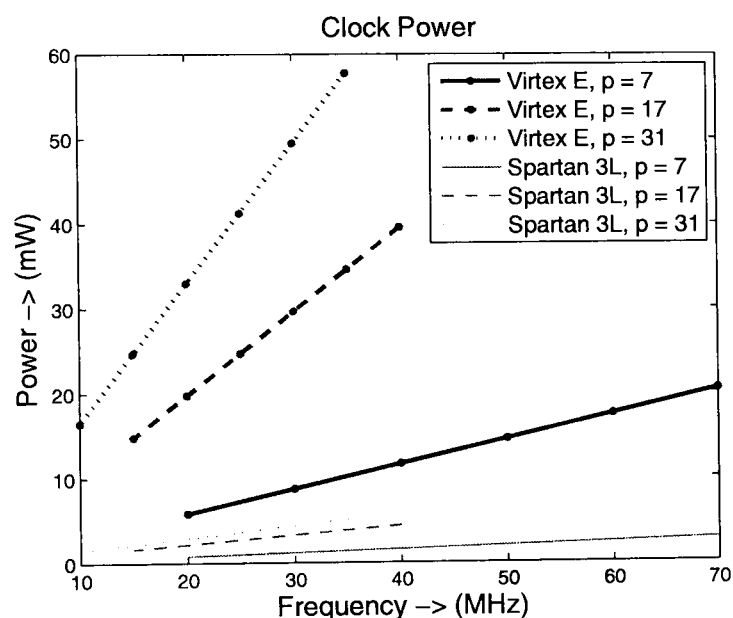


Figure 6.15: CP chart for the FRIT core

SP Model

SP is proportional to the number and length of nets over which signal switching occurs. It also depends on the voltage levels between which the switching occurs, as well as the frequency of switching. Taking into consideration these parameters, and the SP data presented in Fig 6.16, the SP model for the proposed FRIT core is defined in Eq. 6.38 where s_1, s_2, s_3 and s_4 are scaling coefficients.

$$SP = s_1 \cdot v^2 \cdot TA^{s_2} \cdot f + s_3 \cdot f + s_4 \quad (6.38)$$

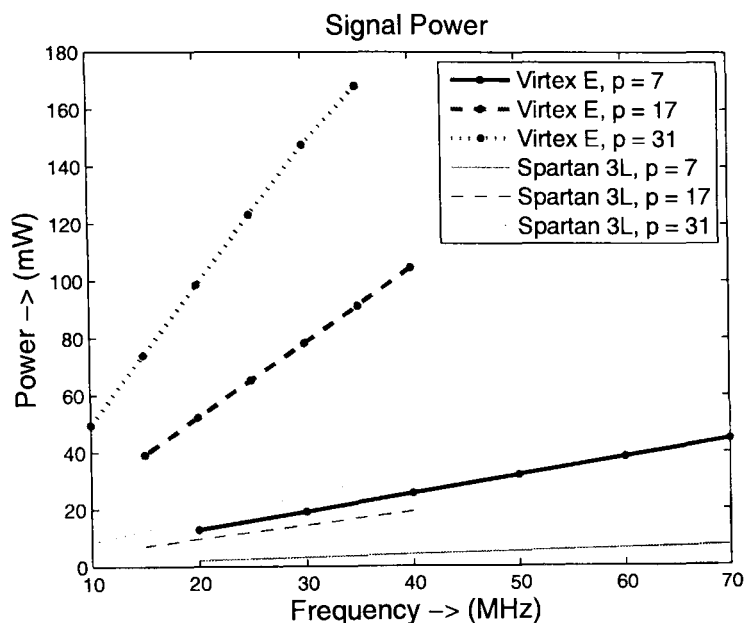


Figure 6.16: SP chart for the FRIT core

LP Model

LP consumption is a function of the number of slices occupied, chip frequency and voltage. By observing and interpreting the power data presented in Fig. 6.17, the best model describing the LP for the proposed FRIT architecture is described in Eq. 6.39. $l1$, $l2$, $l3$ and $l4$ are scaling coefficients in the LP model.

$$LP = l1 \cdot v^2 \cdot TA_P \cdot f + l2 \cdot f + l3 \quad (6.39)$$

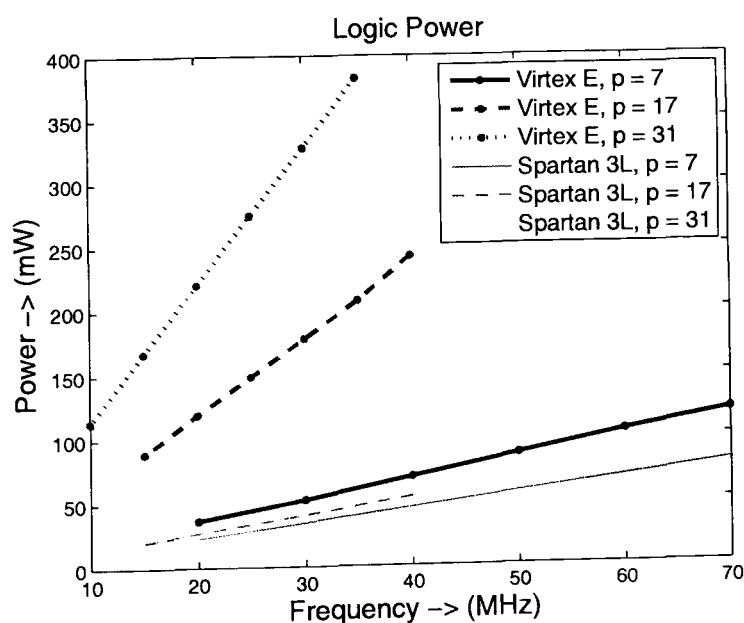


Figure 6.17: LP chart for the FRIT core

IpP Model

The IpP depends on the number of input pins in the design, the block size, chip voltage, and input frequency. The corresponding model that satisfies the curve fitting requirements of the FRIT architecture for the data presented in Fig. 6.18 is defined in Eq. 6.40 where $i1, i2$ and $i3$ are scaling coefficients. It is interesting to note that the input buffers of the Spartan 3L platform dissipate no power, and hence the coefficients in the corresponding models for this platform are zero.

$$IpP = i1 \cdot v^2 \cdot f \cdot (p^2)^{i2} + i3 \quad (6.40)$$

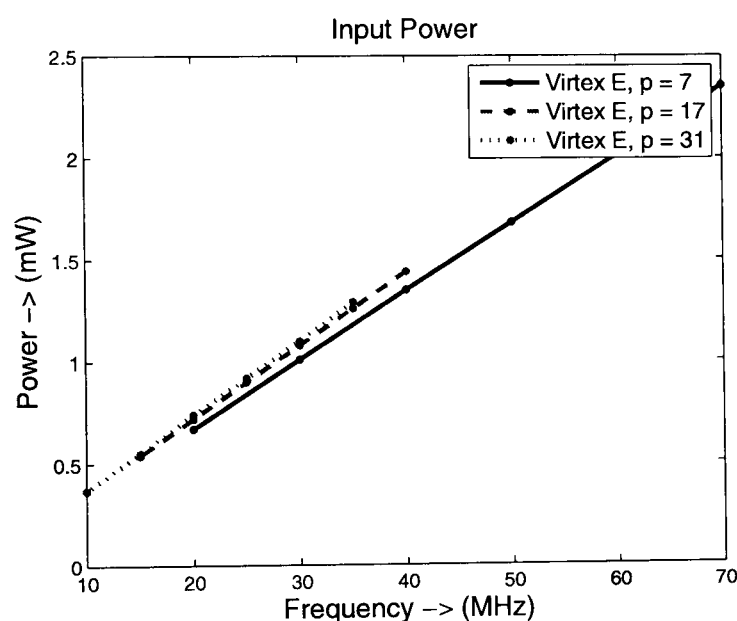


Figure 6.18: IpP chart for the FRIT core

OP Model

The OP depends on the number of output pins in the design, the output block size, output voltage V_{cco} , and frequency f . Based on OP data presented in Fig. 6.19 the corresponding model is described in Eq. 6.41 where p is the block size, of the output vector, $o1, o2$ and $o3$ are scaling coefficients in the output power model. It can be observed from the graph that the OP is nearly identical for all three block sizes $p = 17$ and $p = 31$. This is because they both have the same number of output pins, and nearly identical activity rates and hence dissipate almost same amounts

of power at the output pads for the same value of frequency.

$$OP = o1 \cdot (v^2 \cdot N) \cdot f^2 \cdot \lceil \log_2(255 \cdot p) \rceil \cdot p \cdot (p + 1)^{o2} + o3 \quad (6.41)$$

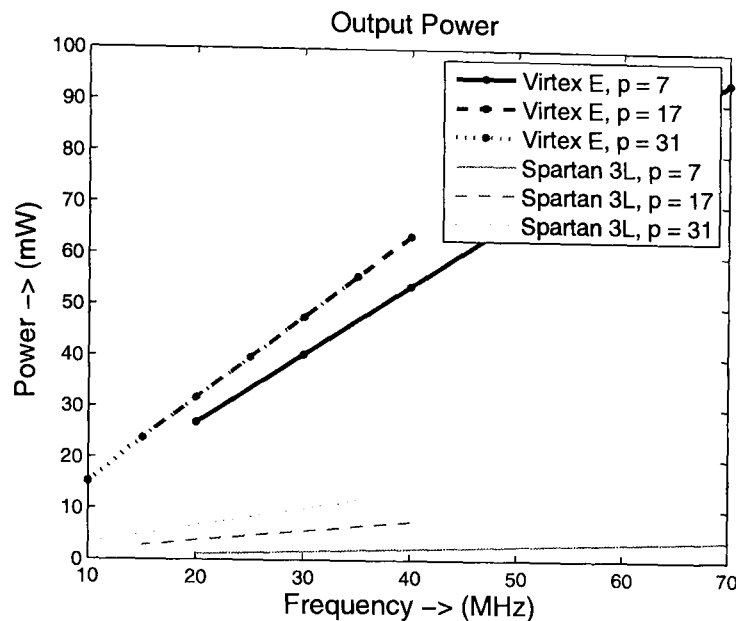


Figure 6.19: OP chart for the FRIT core

The coefficient values of the area and power models are obtained by performing non linear regression until convergence, and are presented in Table 6.6.

Power Modelling for the ASIC Platform

The ASIC platform is not usually characterised by individual components of power as in the case of FPGAs. Hence, a single power model describing the total on-chip power consumption has been constructed based on the power estimates for the FRIT chip developed.

Power estimation in ASIC is performed using System Design Automation (SDA). The power analysis is performed as per the following steps. First the Verilog models are analysed to check if they can be synthesised. Then the design is elaborated and built with generic and technology independent components like gates, flip flops, multiplexers, etc. This is followed by unquify, where multiple copies of the sub-design are instantiated whenever it is referred in the upper level of the hierarchy, and each copy is optimised in a unique way according to the conditions and constraints. The last step of synthesis is compilation, where the network generic components are

Table 6.6: Coefficient values for area and power models for the proposed FRIT architecture on different FPGA platforms

	Virtex-E	Spartan-3L
a1	9.957E-02	1.039E-01
a2	2.711E+02	2.378E+02
a3	-2.745E+03	-2.380E+03
r^2	1.000E+00	1.000E+00
c1	2.402E-04	4.372E-05
c2	4.867E-02	1.977E-02
c3	1.865E-01	1.600E-01
r^2	9.993E-01	9.413E-01
s1	2.393E-04	2.007E-03
s2	1.143E+00	7.610E-01
s3	1.143E+00	-1.242E-01
s4	8.498E-01	-2.311E-03
r^2	9.997E-01	1.000E+00
l1	1.607E-03	9.030E-05
l2	5.082E-02	1.095E+00
l3	3.810E-01	8.603E-01
r^2	9.975E-01	9.921E-01
i1	9.138E-03	0.000E+00
i2	3.217E-02	0.000E+00
i3	4.037E-03	0.000E+00
r^2	9.998E-01	NA
o1	1.508E-02	6.479E-05
o2	-1.078E+00	5.423E-02
o3	-1.978E-01	-6.745E-02
r^2	9.997E-01	9.949E-01

translated into a netlist of the target library. Finally, the power dissipation of the device can be determined from the power report.

A plot of the total internal power dissipation of the ASIC chip has been presented in figure 6.20. Based on this power data the overall power dissipation model for the ASIC based FRIT implementation is described in Eq. 6.42. The corresponding model coefficients are presented in Table 6.7.

$$TP = t1 \cdot v^2 \cdot TA \cdot f + t2 \cdot v^2 \cdot TA^{t3} \cdot f + t4 \cdot f + t5 \quad (6.42)$$

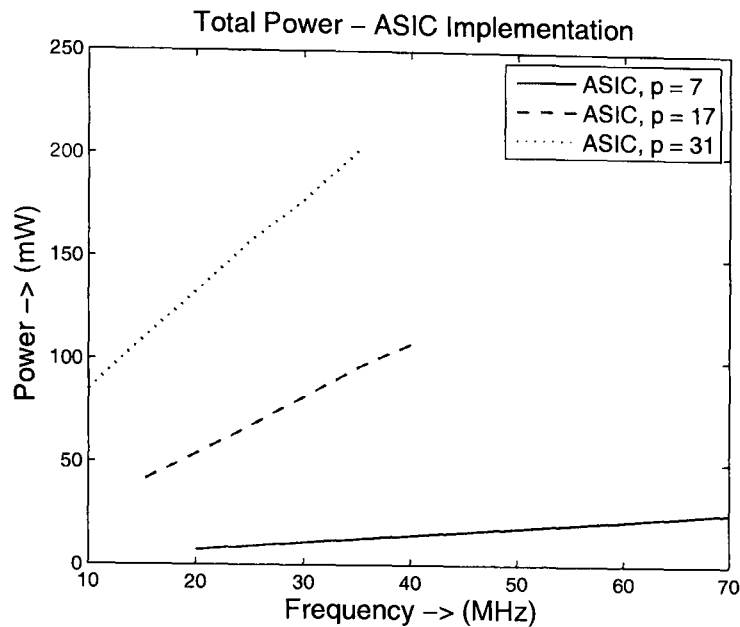


Figure 6.20: ASIC SP chart for the FRIT core

Table 6.7: Coefficient values for area and power models for the ASIC implementation of the proposed FRIT architecture

	Coefficient	r^2
a1	0.00	
a2	0.07	1.00
a3	-0.69	
t1	0.53	
t2	2.05	
t3	0.44	0.99
t4	-1.22	
t5	14.70	

6.5 FLPAM Evaluation

Unlike most high power modelling techniques available for FPGAs, the FLPAM methodology scales very well with changes in platforms, and even prototyping technologies. Advances in the device technology of the prototyping platform naturally result in lower power and energy consumption metrics. Although this results in different constant coefficients for each platform, it is important to highlight that the model itself remains unchanged. A comparison of FLPAM with existing FPGA power modelling methodologies is presented in Table 6.8.

Table 6.8: Comparison modelling characteristics of different approaches with FLPAM

	Avg r2	Level	Scaleable	Platform Independent
Proposed	.97	High	Yes	Yes
[66]	.97	High	No	Yes
[61]	Low	Architectural	Yes	No
[63]	.95	High	No	No
[65]	.95	RTL	No	No
[68]	.82	High	Yes	No
[67]	Various	High	No	Yes
[69]	.97	RTL	Yes	Yes
[72]	High	Architectural	Yes	Yes
[74]	.92	Mixed	Yes	No

6.6 Conclusion

In this chapter, a novel high level IP core macromodelling methodology called functional level power analysis and modelling that overcomes the lacunae in existing modelling methodologies for FPGA based designs has been presented. A power aware optimisation oriented design flow incorporating FLPAM has also been proposed. The mathematical techniques that form the basis of FLPAM have been validated by a range of custom IP cores.

In the next chapter, concluding remarks about the work presented in this thesis will be provided.

Chapter 7

Conclusions and Future Work

7.1 Introduction

Image processing and signal processing techniques requires enormous computing power. Applications such as transformations, image and signal analysis and classification, communication algorithms, etc are increasingly being used. All these applications are data and compute intensive, and the demand for such applications is only going to rise further in future with the advent of pervasive and ubiquitous computing. There is a real need of dedicated processors for high speed computation to meet the requirements of real time processing.

FPGAs can perform mathematical operations on an entire vector or matrix at the same time and the current generation of DSP-capable FPGAs yields ultra-high performance and highly flexible signal-processing systems [167]. This makes FPGAs an attractive platform for implementing many of these applications.

However, it is worth mentioning that the nature of the applications and algorithms that have been targeted in this work very often impose serious limitations on the amount of hardware involved and the rate of power consumption due to factors including mobility constraints, cost performance trade-offs, form factor, etc. Thus, there has been a continued effort to meet the conflicting challenges of ever-growing computational demand with minimal utilisation of hardware resources and power [168].

An important goal of the work reported in this thesis is to exploit techniques at

the algorithmic, architectural and sub-RTL levels to deliver highly optimised and efficient image and signal processing IP cores for FPGA based designs. Another key objective it to address the issues of dynamic power dissipation in FPGAs through the development and application of a novel high level power modelling technique encapsulated within a suitable design flow to optimise and model the aforementioned IP cores.

In the rest of this chapter results obtained throughout this research are summarised and evaluated. Some possible routes to be investigated for a future extension of this work are also provided.

7.2 Evaluation of Results and Contributions

The preceding chapters described different design methodologies used for the efficient design and implementation of various transform methods and signal processing algorithms on FPGA. A brief analysis of voltage scaling and a novel proposal for high level power modelling for FPGAs have also been discussed. This section is concerned with the evaluation of the work presented in these chapters.

7.2.1 Measurement of Success

The criteria undertaken in this project for the measurement of the proposed architectures performances was the comparison of the new architectures with existing implementations. The comparison was based on the computation time, area required, throughput rate and power dissipation; all of which depends on the the various design optimisation strategies pursued and design parameters such as word-length of the input data and transform size. In the case of the implementation of these architectures, the comparison was based on the number of CLBs used, the maximum running frequency of the design and the power and energy metrics. The proposed architectures were implemented and tested on the XCV2000E Virtex-E FPGA devices and in the meantime they were synthesised on other FPGA devices in order to make a fair and consistent comparison with existing cores using the same platform.

FLPAM has been developed in response to the lack of availability of a similar tool for FPGA based designs, which precludes the possibility of a direct comparison with existing methodologies. However, an objective comparison based on model accuracy and a number of other parameters such as modelling level, scalability and platform independence has been performed to evaluate the ability of FLPAM to perform stated objectives.

7.2.2 Important Novelty Claims

The key novel contributions of this work resulting in an advancement of the state of art in the field are listed below:

- The IP cores that have been developed in this work have a number of original features. The InP core that has been developed involves a novel mathematical transformation resulting in compact core footprint. The FRAT and FRIT architectures presented are based on a unique semi-systolic approach for address dereferencing. The GMM core is an application specific efficient architecture that involves a number of novel features including a special DA based triangular multiplication unit and LPF exponential unit. The circular convolution core presented is based on the first reported systolic implementation of DA;
- This work represents the first systematic study about the combined influence of voltage scaling and performance enhancement techniques on commercially available FPGA platforms; and
- FLPAM is a key contribution of this work. Novelty lies in various aspects including portability, accuracy and the modelling philosophy of creating a separation between the core designer and core user.

7.2.3 Results Achieved

In Chapter 2, a set of goals were specified which would determine the success of the work presented in this thesis, namely:

- To design and implement scalable, parameterisable, efficient and novel IP cores for a range of 1-D and 2-D transform and matrix operation based IP cores suitable for use in both general purpose signal processing and specific image and video processing problems and applications;
- To apply optimisation strategies at various abstraction levels and to analyse the effectiveness of these techniques for performance enhancement and power reduction;
- To investigate the best performance trade-offs such as area/speed for the FPGA implementations of these cores;
- To perform an exploratory study into the suitability of supply voltage scaling as a technique for controlling power dissipation in future generation FPGAs; and
- To develop a novel and accurate high level power modelling methodology suited for optimisation and power aware deployment of FPGA based IP cores.

Taking into account the initial objectives, the following points can be made about the achievements of the project:

- In Chapter 3, algorithmic level transformations for a number of FPGA based IP cores resulting in high performance and power efficient architectures has been studied for applications such as inner product using distributed arithmetic [169], sparse-OBC based DA for matrix transforms [170], fast Hadamard transform [166, 171] and Gaussian mixture modelling [14]. All these architectures outperform existing implementations in various key performance metrics.
- In Chapter 4, architectural techniques such as parallelism, pipelining and systolisation have been exploited to yield efficient and power aware architectures. The IP cores that have been developed include finite Radon transform [146, 172], finite ridgelet transform [173–175], systolised FIR filters [176] and colour space conversion [177]. All these architectures outperform existing implementations in various key performance metrics.

- In Chapter 5, it has been shown that supply voltage scaling is a promising technique for further reduction of power in FPGA based systems. The key objective of this chapter was to verify the suitability of voltage scaling as a key area of focus in future work and this has been achieved.
- In Chapter 6, a novel high level power analysis and modelling methodology [178,179] that is completely aligned with the objectives stated in Chapter 2 has been developed and successfully verified on the various FPGA IP Cores presented in preceding chapters.

It can therefore be claimed that the project has made significant progress in meeting the key stated objectives.

7.2.4 Limitations

The objectives stated in Chapter 2 have been met and fully achieved. In the meantime, a few of restrictions and limitations have been raised during the development of this research project:

- Architectures developed for general purpose use can be easily modified for efficient implementation on other platforms (e.g. using the on-chip memory allows us to manipulate just 32-bit arithmetic);
- Real hardware implementation of the proposed matrix transforms has been carried out on the RC1000 development board equipped with Virtex-2000E FPGA. It is worth mentioning that proposed designs are platform independent and can be adopted/implemented easily on the most recent FPGAs. We are fully aware by using the resources available on the recent platforms better performance can be achieved. But, due to time and funding limitations, we were unable to carry out real implementations on these platforms; and
- Scope for improving the efficiency of deploying FLPAM in iterative designs by using it to develop super-models incorporating a number of IP cores. This is a logical extension which is certain to yield interesting insights; but at present has not been carried out due to time limitations.

7.3 Future Work

The work undertaken in this Ph.D. project has concentrated on the development of a number of efficient high performance IP cores for DSP and DIP applications. A novel high level power modelling methodology has also been applied in conjunction to a power aware design flow in order to optimise these cores. A set of objectives for the future include:

- *Expanding the scope of IP core development and optimisation:*

A number of cores from the FMAT library that have not been addressed in this work can be optimised further. Additions to the core library in order to address newer applications is also an important objective for the future;

- *Extending FLPAM to the system level*

FLPAM is very successful in high level modelling of FPGA based IP cores. With the increasing tendency for IP core driven design, we believe this methodology holds a lot of promise for assisting in power aware design. However, a logical extension to FLPAM will be the development of techniques to model a complete system comprising a number of IP cores, using individual core data. This can be achieved by means of stochastic modelling. Stochastic modelling is based on the concept that results that are deduced from the best estimates of all parameters are not equal to the best estimates of the results taking into account all underlying variables. This can be done by formulating the problem of system-level power management as a stochastic optimisation problem based on the theories of continuous-time Markov decision processes and stochastic networks. Extensions to more complex systems, including non-Markovian models are also possible;

- *Extending the MCG system*

The Matrix Core Generator (MCG) system is a high level push-button design flow tool that has been developed by the PRCCV research group in order to easily access and deploy the FMAT cores [180].

FLPAM can be integrated into the MCG system to visualise the power and

energy characteristics of a particular core during the design phase. This will enable the designer to easily analyse the various tradeoffs involved;

- *Performance enhanced voltage scaling*

In this work, only an exploratory study has been carried out to analyse the influence of voltage scaling on power dissipation in FPGAs. The preliminary results that have been obtained are very promising and this seems to be an increasingly important area of research that FPGA manufacturers themselves are studying seriously [109]. The scope of this work package can be increased by analysing various other cores. Additionally, this analysis can be mathematically modelled and integrated into the existing FLPAM approach and

- *FLPAM validation on custom FPGA platforms*

The accuracy of FLPAM can be analysed on real FPGA development boards suitably modified for power measurements; in order to validate the mathematical concepts at a physical level. However, in this case, only a generalised power model can be derived, as it is not possible to measure individual components of power such as clock, signal, logic, etc. separately.

Bibliography

- [1] C.-L. Wang and C.-H. Chang, “A DHT-based FFT/IFFT processor for VDSL transceivers,” *Proceedings of the 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 1213–1216, May 2001.
- [2] B. L. Hutchings, “ASICs, processors and configurable computing,” in *Proceedings of the 30th Hawaii International Conference on System Sciences*, vol. 1, January 1997, p. 719.
- [3] K. Rajan, K. S. Sangunni, and J. Ramakrishna, “Dual-dsp systems for signal and image-processing,” *The EUROMICRO Journal on Microprocessing and Microsystems*, vol. 19, no. 9, pp. 556–560, 1993.
- [4] T. Akiyama, H. Aono, K. Aoki, K. W. Ler, B. Wilson, T. Araki, T. Morishige, H. T. A. Sato, S. Nakatani, and T. Senoh, “MPEG2 video codec using image compression DSP,” *IEEE Transactions on Consumer Electronics*, vol. 40, no. 3, pp. 466–472, August 1994.
- [5] P. Donachy, “Design and implementation of a high level image processing machine using reconfigurable hardware,” PhD Thesis, School of Computer Science, The Queen’s University of Belfast, 1996.
- [6] W. S. Carter, “The future of programmable logic and its impact on digital system design,” in *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors*, October 1990, pp. 10–16.

- [7] B. Zahiri, "Structured ASICs: Opportunities and challenges," in *Proceedings of the International Conference on Computer Design*, October 2003, pp. 404–409.
- [8] K. Wu and Y. Tsai, "Structured ASIC, evolution or revolution?" in *Proceedings of the International Symposium on Physical Design*, April 2004, pp. 103–106.
- [9] K. Compton and S. Hauck, "Reconfigurable computing: A survey of systems and software," *ACM Computing Surveys*, vol. 34, no. 2, pp. 171–210, June 2002.
- [10] "Virtex-4 family overview," Xilinx Inc., Tech. Rep. DS112 (v2.0), January 2007.
- [11] [Online]. Available: www.xilinx.com
- [12] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, April 1965.
- [13] T. E. Starner, "Powerful change part 1: Batteries and possible alternatives for the mobile market," *IEEE Pervasive Computing*, vol. 2, no. 4, pp. 86–88, October-December 2003.
- [14] S. Minghua, A. Bermak, S. Chandrasekaran, and A. Amira, "An efficient FPGA implementation of gaussian mixture models-based classifier using distributed arithmetic," in *Proceedings of the 13th IEEE International Conference on Electronics, Circuits and Systems*, December 2006.
- [15] F. Bensaali, A. Amira, A. AhmedSaid, and I. S. Uzun, "Efficient implementation of large parallel matrix product for DOTs," in *Proceedings of the International Conference on Computer, Communication and Control Technologies*, July 31 August 2 2003.
- [16] A. Amira, A. Bouridane, P. Milligan, and M. Roula, "Novel FPGA implementations of walsh-hadamard transforms for signal processing," *IEE Proceedings on Vision, Image and Signal Processing*, vol. 148, no. 6, pp. 377–383, 2001.

- [17] A. Amira, A. Bouridane, P. Milligan, and A. Belatreche, "Design of efficient architectures for discrete orthogonal transforms using bit level systolic structures," *IEE Proceedings - Computers and Digital Techniques*, vol. 149, no. 1, pp. 17–24, January 2002.
- [18] A. Amira, "An FPGA based parameterisable system for discrete hartley transforms implementation," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 2, 2003, pp. 567–570.
- [19] I. S. Uzun and A. Amira, "A FPGA-based parametrizable system for high-resolution frequency-domain image filtering," *Journal Of Circuits Systems And Computers*, vol. 14, no. 5, pp. 895–922, February 2005.
- [20] —, "Real-time 2-D wavelet transform implementation for HDTV compression," *Real-time imaging*, vol. 11, no. 2, pp. 151–165, April 2005.
- [21] —, "Design and FPGA implementation of finite ridgelet transform," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 6, May 2005, pp. 5826–5829.
- [22] F. Bensaali and A. Amira, "Accelerating colour space conversion on reconfigurable hardware," *Image and Vision Computing (Elsevier)*, vol. 23, no. 11, pp. 935–942, October 2005.
- [23] K. R. Castleman, *Digital Image Processing*. Prentice Hall, 1996.
- [24] L.-W. Chang and M.-C. Wu, "A bit level systolic array for walsh-hadamard transforms," *Signal Processing*, vol. 31, no. 3, pp. 341–347, April 1993.
- [25] S. Y. Kung, *VLSI Array Processors*. Prentice Hall, 1988.
- [26] "Virtex-E 1.8 V Field Programmable Gate Arrays," Xilinx Inc., Datasheet DS022-1 (v2.3), July 2002.
- [27] S. K. Bahl, "Design and prototyping a fast hadamard transformer for WCDMA," June 2003, pp. 134–140.

- [28] R. Hashemian, S. Vijayaraghavan, and J. Citta, "A new gate image encoder; algorithm, design and implementation," *IEEE Midwest Symposium Circuits and Systems*, vol. 1, pp. 418–421, 1999.
- [29] B. J. Falkowski and T. Sasao, "Unified algorithm to generate walsh functions in four different orderings and its programmable hardware implementations," *IEE Proceedings on Vision Image and Signal Processing*, vol. 152, no. 6, pp. 819–826, December 2005.
- [30] "FLEX 10K Embedded Programmable Logic Family Data Sheet ," Altera Corp., Datasheet DS-F10K-4.2, January 2003.
- [31] M. N. Do and M. Vetterli, "Orthonormal finite ridgelet transform for image compression," in *Proceedings of the International Conference on Image Processing*, vol. 2, September 2000, pp. 367–370.
- [32] E. J. Candes and D. L. Donoho, *Curves and Surfaces*. Vanderbilt University Press, 2000, ch. A Surprisingly Effective Nonadaptive Representation for Objects With Edges, pp. 105–120.
- [33] E. Candes, "Ridgelets: Theory and application," Ph.D. Thesis, Department of Statistics, Stanford University, September 1998.
- [34] C. A. Rahman and W. Badawy, "Architectures the finite radon transform," *IEE Electronic Letters*, vol. 40, no. 15, pp. 931–932, July 2004.
- [35] F. Matus and J. Flusser, "Image representation via a finite radon transform," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 996–1006, October 1993.
- [36] "Virtex-II platform FPGAs: complete data sheet," Xilinx Inc., Datasheet DS031 (v3.3), June 2004.
- [37] J. Wisinger and R. Mahapatra, "FPGA based image processing with the curvelet transform," Texas A & M University, TX, Tech Report TR-CS-2003-01-0, 2003.

- [38] "Handel-C language reference manual," Celoxica Ltd., Tech. Rep. RM-1003-4.2, 2004.
- [39] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Prentice Hall Inc., 2002.
- [40] L. Pillai, "Color space converter: YCrCb to RGB," Xilinx Inc., Application Note XAPP283 (v1.3.1), March 2005. [Online]. Available: www.xilinx.com
- [41] M. Sima, S. Vassiliadis, S. Cotofana, and J. T. J. van Eijndhoven, "Color space conversion for MPEG decoding on FPGA-augmented trimedia processor," in *Proceedings of the 14th IEEE International Conference on Application-specific Systems, Architectures and Processors*, June 2003.
- [42] "ACEX 1K Programmable Logic Device Family Data Sheet ," Altera Corp., Datasheet DS-ACEX-3.4, May 2003.
- [43] "Color space converter: Megacore function user guide," Altera Inc., Application Note (v.2.2.0), June 2004. [Online]. Available: www.altera.com
- [44] "Color space converters," Amphion Semiconductor Ltd., Datasheet DS6400 V1.1, May 2002. [Online]. Available: www.amphion.com
- [45] "Csc color space converter," CAST Inc., Application Note, April 2002. [Online]. Available: www.cast-inc.com
- [46] "High performance color space converter," ALMA Technologies, Datasheet, May 2002. [Online]. Available: www.alma-tech.com
- [47] M. Shi and A. Bermak, "An efficient digital VLSI implementation of gaussian mixture models-based classifier," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 9, pp. 962–974, September 2006.
- [48] J. P. Choi, S. C. Shin, and J. G. Chung, "Efficient ROM size reduction for distributed arithmetic," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 2, May 2000, pp. 61–64.

- [49] C. F. Chen, "Implementing FIR filters with distributed arithmetic," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 5, pp. 1318–1321, October 1985.
- [50] H. R. Lee, C. W. Jen, and C. M. Liu, "On the design automation of the memory-based VLSI architectures for FIR filters," *IEEE Transactions on Consumer Electronics*, vol. 39, no. 3, pp. 619–629, August 1993.
- [51] K. Nourji and N. Demassieux, "Optimal VLSI architecture for distributed arithmetic-based algorithms," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, April 1994, pp. 509–512.
- [52] M. Mehendale, S. D. Sherlekar, and G. Venkatesh, "Area-delay tradeoff in distributed arithmetic based implementation of FIR filters," in *Proceedings of the 10th International Conference on VLSI Design*, January 1997, pp. 124–129.
- [53] S. S. Jeng, H. C. Lin, and S. M. Chang, "FPGA implementation of FIR filter using M-bit parallel distributed arithmetic," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, May 2006, pp. 875–878.
- [54] H. Yoo and D. V. Anderson, "Hardware-efficient distributed arithmetic architecture for high-order digital filters," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, vol. 5, March 2005, pp. 125–128.
- [55] "Stratix Device Handbook," Altera Corp., Datasheet ver 3.4 vol.1, January 2006.
- [56] R. S. Grover, W. Shang, and Q. Li, "A faster distributed arithmetic architecture for FPGAs," in *Proceedings of the 10th ACM International Symposium on Field Programmable Gate Arrays*, February 2002, pp. 31–39.
- [57] K. K. Parhi, *VLSI Digital Signal Processing Systems Design and Implementation*. John Wiley and Sons, 1999.

- [58] R. Wyrzykowski and S. Ovrmenko, "Flexible systolic architecture for VLSI FIR filters," *IEE Proceedings Computers & Digital Techniques*, vol. 139, no. 2, pp. 170–172, March 1992.
- [59] B. K. Mohanty and P. K. Meher, "Cost-effective novel flexible cell-level systolic architecture for high throughput implementation of 2-D FIR filters," *IEE Proceedings-Computers and Digital Techniques*, vol. 143, no. 9, pp. 436–439, November 1996.
- [60] ———, "Novel flexible systolic mesh architecture for parallel VLSI implementation of finite digital convolution," *IETE Journal of Research*, vol. 44, no. 6, pp. 261–266, November 1988.
- [61] K. K. Poon, S. J. E. Wilton, and A. Yan, "A detailed power model for field-programmable gate arrays," *ACN Transactions on Design Automation of Electronic Systems*, vol. 10, no. 2, pp. 279–302, April 2005.
- [62] K. K. Poon, A. Yan, and S. J. E. Wilton, *Proceedings of the International Conference on Field Programmable Logic and Applications*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, September 2002, vol. 2438/2002, ch. A Flexible Power Model for FPGAs, pp. 312–321.
- [63] T. Osmulski, J. T. Muehring, B. Veale, J. M. West, H. Li, S. Vanichayobon, S.-H. Ko, J. K. Antonio, and S. K. Dhall, "A probabilistic power prediction tool for the xilinx 4000-series FPGA," in *Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing*, 2000, pp. 776–783.
- [64] H. G. Lee, K. Lee, Y. Choi, and N. Chang, "Cycle-accurate energy measurement and characterization of FPGAs," *Analog Integrated Circuits and Signal Processing*, vol. 42, no. 3, pp. 239–251, March 2005.
- [65] H. G. Lee, S. Nam, and N. Chang, "Cycle-accurate energy measurement and high-level energy characterization of FPGAs," in *Proceedings of the Fourth International Symposium on Quality Electronic Design*, March 2003, pp. 267–272.

- [66] L. Shang and N. K. Jha, "High-level power modeling of CPLDs and FPGAs," in *Proceedings of the International Conference on Computer Design*, September 2001, pp. 46–51.
- [67] T. Jiang, X. Tang, and P. Banerjee, "Macro-models for high level area and power estimation on FPGAs," in *Proceedings of the 14th ACM Great Lakes symposium on VLSI*, April, 2004, pp. 162–165.
- [68] V. Degalahal and T. Tuan, "Methodology for high level estimation of FPGA power consumption," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, vol. 1, Jan 2005, pp. 657–660.
- [69] M. French, L. Wang, T. Anderson, and M. Wirthlin, "Post synthesis level power modeling of FPGAs," in *Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, April 2005, pp. 281–282.
- [70] P. Bellows and B. Hutchings, "JHDL - An HDL for reconfigurable systems," in *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines*, April 1998, pp. 175–184.
- [71] J. Becker, M. Huebner, and M. Ullmann, "Power estimation and power measurement of xilinx virtex FPGAs: Trade-offs and limitations," in *Proceedings of the 16th Symposium on Integrated Circuits and Systems Design*, September 2003, pp. 283–288.
- [72] L. Zhong, S. Ravi, A. Raghunathan, and N. K. Jha, "Power estimation for cycle-accurate functional descriptions of hardware," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*, November 2004, pp. 668–675.
- [73] J. H. Anderson and F. N. Najm, "Power estimation techniques for FPGAs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 10, pp. 1015–1027, October 2004.

- [74] F. Li, Y. Lin, L. He, D. Chen, and J. Cong, "Power modeling and characteristics of field programmable gate arrays," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 11, pp. 1712–1724, November 2005.
- [75] D. Marculescu, R. Marculescu, and M. Petram, "Information theoretic measures for power analysis," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 15, no. 6, pp. 599–610, June 1996.
- [76] M. Nemani and F. N. Najm, "Toward a high-level power estimation capability," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 15, no. 6, pp. 588–598, June 1996.
- [77] K.-T. Cheng and V. D. Agrawal, "An entropy measure for the complexity of multi-output boolean functions," in *Proceedings of the 27th ACM/IEEE Design Automation Conference*, June 1990, pp. 302–305.
- [78] F. Ferrandi, F. Fummi, E. Macii, M. Poncino, and D. Sciuto, "Power estimation of behavioral descriptions," in *Proceedings of the Design, Automation and Test in Europe*, February 1998, pp. 762–766.
- [79] K. D. Muller-Glaser, K. Kirsch, and K. Neusinger, "Estimating essential design characteristics to support projectplanning for ASIC design management," in *Proceedings of the IEEE International Conference on Computer-Aided Design*, November 1991, pp. 148–151.
- [80] M. Nemani and F. N. Najm, "High-level area prediction for power estimation," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, May 1997, pp. 483–486.
- [81] —, "High-level area and power estimation for VLSI circuits," in *Proceedings of the IEEE/ACM international conference on Computer-aided design*, November 1997, pp. 114–119.

- [82] C. X. Huang, B. Zhang, A.-C. Deng, and B. Swirski, "The design and implementation of powermill," in *Proceedings of the 1995 International Symposium on Low Power Electronics and Design*, April 1995, pp. 105–110.
- [83] F. N. Najm, R. Burch, P. Yang, and I. N. Hajj, "Probabilistic simulation for reliability analysis of cmos vlsi circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 4, pp. 439–450, 1990.
- [84] C. Tsui, M. Pedram, and A. M. Despain, "Efficient estimation of dynamic power consumption under a real delay model," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, November 1993, pp. 224–228.
- [85] F. N. Najm, "Transition density: A new measure of activity in digital circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 2, pp. 310–323, February 1993.
- [86] R. Marculescu, D. Marculescu, and M. Pedram, "Efficient power estimation for highly correlated input streams," in *Proceedings of the 32nd ACM/IEEE Design Automation Conference*, June 1995, pp. 628–634.
- [87] C.-Y. Tsui, J. Monteiro, M. Pedram, S. Devadas, A. M. Despain, and B. Lin, "Power estimation methods for sequential logic circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 3, no. 3, pp. 404–416, September 1995.
- [88] R. Burch, F. N. Najm, P. Yang, and T. N. Trick, "A monte carlo approach for power estimation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 1, no. 1, pp. 63–71, March 1993.
- [89] L. Yuan, C. Teng, and S. Kang, "Statistical estimation of average power dissipation using nonparametric techniques," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 6, no. 1, pp. 65–73, March 1998.

- [90] T. Chou and K. Roy, "Statistical estimation of sequential circuit activity," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, November 1995, pp. 34–37.
- [91] D. Marculescu, R. Marculescu, and M. Pedram, "Stochastic sequential machine synthesis targeting constrained sequence generation," in *Proceedings of the 33^d Annual ACM IEEE Design Automation Conference*, June 1996, pp. 696–701.
- [92] R. Marculescu, D. Marculescu, and M. Pedram, "Adaptive models for input data compaction for power simulations," in *Proceedings of the Asia and South Pacific Design Automation Conference*, January 1997, pp. 391–396.
- [93] S. Powell and P. Chau, "Estimating power dissipation of VLSI signal processing chips: The FA techniques," in *Proceedings of the IEEE Workshop on VLSI Signal Processing*, vol. 4, 1990, pp. 250–259.
- [94] P. Landman and J. Rabaey, "Power estimation for high-level synthesis," in *Proceedings of the European Conference on Design Automation*, Paris, France, February 1993, pp. 361–366.
- [95] S. Gupta and F. N. Najm, "Power macromodeling for high-level power estimation," in *Proceedings of DAC-34: ACM/IEEE Design Automation Conference*, June 1997, pp. 365–370.
- [96] D. Liu and C. Svensson, "Power consumption estimation in CMOS VLSI chips," *IEEE Journal of Solid State Circuits*, vol. 29, no. 6, pp. 663–670, June 1994.
- [97] C.-T. H. Q Wu C-S Ding and M. Pedram, "Statistical design of macro-models for RT-level power evaluation," in *Proceedings of the ACM/IEEE Asia South Pacific Design Automation Conference*, 1997, pp. 523–528.
- [98] T. Sato, Y. Ootaguros, M. Nagamatsus, and H. Tago, "Evaluation of architectural-level power estimation for CMOS RISC processors," in *Proceed-*

- ings of the IEEE Symposium on Low Power Electronics*, October 1995, pp. 44–45.
- [99] C.-T. Hsieh, M. Pedram, G. Mehta, and F. Rastgar, “Profile-driven program synthesis for evaluation of system power dissipation,” in *Proceedings of the 34th Design Automation Conference*, June 1997, pp. 576–581.
- [100] C.-L. Su, C.-Y. Tsui, , and A. M. Despain, “Low power architecture design and compilation techniques for high-performance processors,” *Digest of Papers Comcon Spring '94*, pp. 489–498, February 1994.
- [101] V. Tiwari, S. Malik, and A. Wolfe, “Power analysis of embedded software: A first step toward software power minimization,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 437–445, December 1994.
- [102] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, “Low-power CMOS digital design,” *IEEE Journal of Solid State Circuits*, vol. 27, no. 4, pp. 473–484, 1992.
- [103] R. G. B. M. Gordon and M. A. Horowitz, “Supply and threshold voltage scaling for low power CMOS,” *IEEE Journal of Solid State Circuits*, vol. 32, no. 8, pp. 1210–1216, 1997.
- [104] E. Morifuji, T. Yoshida, T. Kanda, S. Matsuda, S. Yamada, and F. Matsuoka, “Supply and threshold-voltage trends for scaled logic and SRAM MOSFETs,” *IEEE Transactions on Electron Devices*, vol. 53, no. 6, pp. 1427–1432, June 2006.
- [105] H. Belhadj, B. Zahiri, and A. Tai, “Power-sensitive design techniques on FPGA devices,” in *Proceedings of the 46 International IC Conference*, 2001.
- [106] A. Garcia, W. Burleson, and J. L. Danger, “Low power digital design in FPGAs: a study of pipeline architectures implemented in a FPGA using a low

- supply voltage to reduce power consumption,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 5, May 2000, pp. 561–564.
- [107] S.-S. A. Steven J.E. Wilton¹ and W. Luk, *Proceedings of the International Conference on Field Programmable Logic and Application*. Springer, August–September 2004, vol. 3203, ch. The Impact of Pipelining on Energy per Operation in Field-Programmable Gate Arrays, pp. 719–728.
- [108] C. T. Chow, L. S. M. Tsui, P. H. W. Leong, W. Luk, and S. Wilton, “Dynamic voltage scaling for commercial FPGAs,” in *Proceedings of the 2005 IEEE International Conference on Field-Programmable Technology*, December 2005, pp. 173–180.
- [109] T. Tuan, A. Rahman, S. D. S. Trimberger, , and S. Kao, “A 90-nm low-power FPGA for battery-powered applications,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 296–300, February 2007.
- [110] Y. Lin and L. He, “Dual-V_{dd} interconnect with chip-level time slack allocation for FPGA power reduction,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 2023–2034, 2006.
- [111] D. Chen and J. Kong, “Delay optimal low-power circuit clustering for FPGAs with dual supply voltages,” in *Proceedings of the 2004 International Symposium on Low Power Electronics and Design*, August 2004, pp. 70–73.
- [112] A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Brodersen, “Optimizing power using transformations,” *IEEE Transactions On Computer-Aided Design Of Integrated Circuits And System*, vol. 14, no. 1, pp. 12–31, January 1995.
- [113] A. Amira, “A custom coprocessor for matrix algorithms,” Ph.D. dissertation. Queen’s University, Belfast, 2001.

- [114] A. Amira, A. Bouridane, and P. Milligan, "RCMAT: A reconfigurable co-processor for matrix algorithms," in *Proceedings of the Ninth ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, February 2001, pp. 228–231.
- [115] S. A. White, "Applications of distributed arithmetic to digital signal processing: a tutorial review," *IEEE ASSP Magazine*, vol. 6, no. 3, pp. 4–19, July 1989.
- [116] "RC1000 development platform product brief," Celoxica Ltd., Datasheet v1.1, August 2002.
- [117] P. G. Paulin and J. P. Knight, "Force-directed scheduling for the behavioral synthesis of ASICs," *IEEE Transactions in Computer Aided Design*, vol. 8, no. 6, pp. 661–679, June 1989.
- [118] N. U. Ahmed and K. R. Rao, *Orthogonal Transforms for Digital Signal Processing*. Springer-Verlag, 1975.
- [119] W. Chen, C. H. Smith, and S. Fralick, "A fast computation algorithm for the discrete cosine transform," *IEEE Transactions on Communications*, vol. 25, pp. 1004–1009, September 1977.
- [120] S. Chandrasekaran and A. Amira, "FPGA implementation and power modeling of the fast walsh transform," in *Proceedings of the 16th International Conference on Field Programmable Logic and Applications*, August 2006.
- [121] M. H. Lee and M. Kaveh, "Fast hadamard transform based on a simple matrix factorization," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 34, no. 6, pp. 1666–1667, December 1986.
- [122] K. G. Beauchamp, *Applications of Walsh and Related Functions*. Academic Press, 1984.
- [123] F. J. MacWilliams and N. J. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam: North-Holland, 1992.

- [124] M. L. M. Perkowski, "Evolving quantum circuits using genetic algorithm," in *Proceedings of the NASA/DoD Conference on Evolvable Hardware*, 2002, pp. 177–185.
- [125] P. Kitsos, N. Sklavos, K. Papadomanolakis, and O. Koufopavlou, "Hardware implementation of bluetooth security," *IEEE Pervasive Computing Magazine*, vol. 2, no. 1, pp. 21–29, 2003.
- [126] S. S. Nayak and P. K. Meher, "High throughput VLSI implementation of discrete orthogonal transforms using bit-level vector-matrix multiplier," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 46, no. 5, pp. 655–658, May 1999.
- [127] Y. Iguchi and T. Sasao, "Hardware to compute walsh coefficients," in *Proceedings of the International Symposium on Multiple-Valued Logic*, 2005, pp. 75–81.
- [128] Y. Huang, K. B. Englehart, B. Hudgins, and A. D. C. Chan, "Optimized gaussian mixture models for upper limb motion classification," in *proceedings of the Conference of the Engineering in Medicine and Biology Society*, vol. 1, 2004, pp. 72–75.
- [129] R. Gutierrez-Osuna, "Pattern analysis for machine olfaction: a review," *IEEE Sensors Journal*, vol. 2, no. 3, pp. 189–202, 2002.
- [130] H. J. Huang and C. N. Hsu, "Bayesian classification for data from the same unknown class," *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 32, no. 2, pp. 137–145, 2002.
- [131] F. Cardinaux, C. Sanderson, and S. Bengio, "User authentication via adapted statistical models of face image," *IEEE Transactions on Signal Processing*, vol. 54, no. 1, pp. 361–373, 2005.
- [132] C. P. Lim, K. Peh, and S. Quek, "Application of the gaussian mixture model to drug dissolution profiles prediction," *Neural Computing and Applications*, vol. 14, no. 4, pp. 345–352, 2005.

- [133] A. Lindemann, L. Dunis, and P. Lisboa, "Level estimation, classification and probability distribution architectures for trading the eur/usd exchange rate," *Neural Computing and Applications*, vol. 14, no. 3, pp. 256–271, 2005.
- [134] Q. Hong and S. Kwong, "A genetic classification method for speaker recognition," *Engineering Applications of Artificial Intelligence*, vol. 18, no. 1, pp. 13–19, 2005.
- [135] C. Sanderson, S. Bengio, and Y. Gao, "On transforming statistical models for non-frontal face verification," *Pattern Recognition*, vol. 39, no. 2, pp. 288–302, 2006.
- [136] H. T. Nagle, S. S. Schiffman, and R. Gutierrez-Osuna, "The how and why of electronic noses," *IEEE Spectrum*, vol. 35, no. 9, pp. 22–34, 1998.
- [137] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley Interscience, 2000.
- [138] D. M. Titterton, A. F. M. Smith, and U. E. Makov, *Statistical Analysis of Finite Mixture Distributions*. John Wiley & Sons, 1985.
- [139] R. J. Petersen and B. Hutchings, "An assessment of the suitability of FPGA-based systems for use in digital signal processing," in *Proceedings of the 5th International Workshop on Field-Programmable Logic and Applications*. Springer-Verlag, 1995, pp. 293–302.
- [140] T. J. Callahan and J. Wawrzynek, "Instruction-level parallelism for reconfigurable computing," in *Proceedings of the 8th International Workshop on Field-Programmable Logic and Applications, From FPGAs to Computing Paradigm*. Springer-Verlag, 1998, pp. 248–257.
- [141] N. Rollins and M. Wirthlin, "Reducing energy in FPGA multipliers through glitch reduction," in *Proceedings of the 7th Annual International Conference on Military Applications of Programmable Logic Devices*, September 2005.

- [142] S. J. E. Wilton, S.-S. Ang, and W. Luk, "The impact of pipelining on energy per operation in field-programmable gate arrays," *Lecture Notes in Computer Science*, pp. 719–728, 2004.
- [143] P. Gribomont and V. V. Dongen, "Generic systolic arrays: A methodology for systolic design," *Lecture Notes in Computer Science*, vol. 668, pp. 746–761, 1993.
- [144] M. N. Do and M. Vetterli, "Contourlets: A new directional multiresolution image representation," in *Proceedings of the International Conference on Image Processing*, vol. 1, September 2002, pp. I-357–I-360.
- [145] P. Zhang and L. Ni, "The curvelet transform based on finite ridgelet transform for image denoising," in *Proceedings of the 7th International Conference on Signal Processing*, vol. 2, Aug-Sep 2004, pp. 978–981.
- [146] S. Chandrasekaran and A. Amira, "High speed / low power architectures for the finite radon transform," in *Proceedings of the International Conference on Field Programmable Logic and Applications*, August 2005.
- [147] P. Toft, "The radon transform - theory and implementation," Ph.D. Thesis, Informatics and Mathematical Modelling, Technical University of Denmark, 1996.
- [148] E. D. Bolker, "The finite Radon transform," in *Integral Geometry (Contemporary Mathematics)*, S. Helgason, R. L. Bryant, V. Guillemin, , and R. O. Wells, Jr, Eds., 1987, vol. 63, pp. 27–50.
- [149] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms and Applications*. Upper Saddle River, NJ: Prentice-Hall, 1996.
- [150] A. Antoniou, *Digital filters : analysis, design, and applications*. New York: McGraw-Hill, 1993.
- [151] H. T. Kung, "Why systolic architectures?" *IEEE Computer*, vol. 15, pp. 37–45, January 1982.

- [152] P. K. Meher, "Hardware-efficient systolization of DA-based calculation of finite digital convolution," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, no. 8, pp. 707–711, August 2006.
- [153] H. Yoo and D. V. Anderson, "Hardware-efficient distributed arithmetic architecture for high-order digital filters," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, March 2005, pp. 125–128.
- [154] A. Albiol, L. Torres, and E. J. Delp, "An unsupervised color image segmentation algorithm for face detection applications," in *Proceedings of the International Conference on Image Processing*, vol. 2, October 2001, pp. 681–684.
- [155] P. Kuchi, P. Gabbur, P. S. Bhat, and S. David, "Human face detection and tracking using skin color modelling and connected component operators," *The IETE Journal of Research, Special issue on Visual Media Processing*, vol. 38, no. 3 & 4, pp. 289–293, May 2002.
- [156] M. Bartkowiak, "Optimizations of color transformation for real time video decoding," in *Proceedings of Digital Signal Processing for Multimedia Communications and Services, EURASIP ECMCS*, 2001.
- [157] J. L. Mitchell and W. B. Pennebaker, *MPEG Video Compression Standard*. Chapman & Hall, 1996.
- [158] J. Bracamonte, P. Standelmann, M. Ansorge, and F. Pellandini, "A multiplierless implementation scheme for the JPEG image coding algorithm," in *Proceedings of the IEEE Nordic Signal Processing Symposium*, June 2000, pp. 17–20.
- [159] F. Bensaali and A. Amira, "Accelerating color space conversion on reconfigurable hardware," *Image and Vision Computing (Elsevier)*, vol. 23, no. 11, pp. 935–942, October 2005.
- [160] "CORE generator guide," Xilinx Inc., Manual v 4.1i, July 2001.

- [161] [Online]. Available: www.celoxica.com
- [162] “Color space converter: MegaCore function user guide,” ALtera Inc., Application Note v.2.2.0, June 2004.
- [163] A. D. G. Garca, L. F. G. Prez, and R. F. Acua, “Power consumption management on FPGAs,” in *Proceedings of the 15th International Conference on Electronics, Communications and Computers*, February 2005, pp. 240–245.
- [164] [Online]. Available: www.nlreg.com
- [165] F. Bensaali, S. Chandrasekaran, and A. Amira, “Power modeling and efficient FPGA implementation of color space conversion,” *Proceedings of the 13th IEEE International Conference on Electronics, Circuits and Systems*, vol. 1, pp. 164–167, December 2006.
- [166] A. Amira and S. Chandrasekaran, “Power modeling and efficient FPGA implementation of FHT for signal processing,” *IEEE Transactions on VLSI Systems*, vol. 15, no. 3, pp. 286–295, March 2007.
- [167] “FPGA co-processing solutions for high-performance signal processing applications,” Altera Corporation, Altera Corporation, 101 Innovation Drive, San Jose, California 95134, USA, White Paper WP-041905-1.1, May, online: <http://www.altera.com/literature/>.
- [168] J. C. Chen, K. M. Sivalingam, P. Agrawal, and R. Acharya, “Scheduling multimedia services in a low-power mac for wireless and mobile atm networks,” *IEEE Transactions on Multimedia*, vol. 1, no. 2, pp. 187–201, June 1999.
- [169] S. Chandrasekaran and A. Amira, “An area efficient low power inner product computation for discrete orthogonal transforms,” in *Proceedings of the IEEE International Conference on Image Processing*, September 2005.
- [170] ———, “Novel sparse OBC based distributed arithmetic architecture for matrix transforms,” in *To Appear in the Proceedings of the 2007 IEEE International Symposium on Circuits and Systems*, 2007.

- [171] —, “FPGA implementation and power modelling of the fast walsh transform,” in *Proceedings of the International Conference on Field Programmable Logic and Applications*, August 2006.
- [172] —, “Power modelling and acceleration of finite radon transform on reconfigurable hardware,” *IEEE Transactions on Computers*, Under Review.
- [173] —, “High speed energy efficient architectures for finite ridgelet transform,” in *NASA Military and Aerospace Applications of Programmable Devices and Technologies Conference*, September 2005.
- [174] —, “Novel hardware / software co-design for finite ridgelet transform,” in *Proceedings of the IEEE International Computer Systems and Information Technology Conference*, July 2005.
- [175] S. Chandrasekaran, A. Amira, S. Minghua, and A. Bermak, “An efficient VLSI architecture and power modeling for the finite ridgelet transform,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Under Review.
- [176] P. K. Meher, S. Chandrasekaran, and A. Amira, “FPGA realization of FIR filters by efficient and flexible systolization using distributed arithmetic,” *IEEE Transactions on Signal Processing*, Under Review.
- [177] F. Bensaali, S. Chandrasekaran, and A. Amira, “Power modeling and efficient FPGA implementation of color space conversion,” in *Proceedings of the 13th IEEE International Conference on Electronics, Circuits and Systems*, December 2006.
- [178] S. Chandrasekaran and A. Amira, “Power reduction for FPGA implementations : Design optimisation and high level modelling,” in *Proceedings of the International Conference on Field Programmable Logic and Applications*, August 2006.
- [179] —, “A new behavioural power modelling approach for FPGA based custom cores,” in *Proceedings of the NASA/ESA Conference on Adaptive Hardware and Systems*, August 2007.

- [180] F. Bensaali, "Accelerating matrix product on reconfigurable hardware for image processing applications," Phd Thesis, School of Computer Science, The Queen's University of Belfast, May 2005.
- [181] I. Page and W. Luk, "Compiling occam into field-programmable gate arrays," in *FPGAs, Oxford Workshop on Field Programmable Logic and Applications*, W. Moore and W. Luk, Eds. 15 Harcourt Way, Abingdon OX14 1NV, UK: Abingdon EE&CS Books, 1991, pp. 271–283.
- [182] *DK Design Suit*, Version 1.1 ed., Celoxica Ltd., 2004.
- [183] S. M. Loo, B. E. Wells, N. Freije, and J. Kulick, "Handel-C for rapid prototyping of VLSI coprocessors for real time systems," in *Proceedings of the Thirty Fourth Southeastern Symposium on System Theory*, vol. 46, No.1, 2002, pp. 6–10.
- [184] D. Sulik, M. Vasilko, D. Durackova, and P. Fuchs, "Design of a RISC microcontroller core in 48 hours," May 2000, (presented at Embedded Systems Show 2000, London Olympia).
- [185] P. Voles, L. Holasek, and M. Vasilko, "ANSI C and Handel-C based rapid prototyping framework for real-time image processing algorithms," in *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms*. CSREA Press, 2002, pp. 153–159.
- [186] "Virtex-II Pro and Virtex-II Pro X platform FPGAs: Complete data sheet," Xilinx Inc., Datasheet DS083-2 (V4.2), March 2005.
- [187] A. Croisier, D. J. Esteban, and V. Riso, "Digital filter for PCM encoded signals," U.S. Patent 3777 130, December 1973.
- [188] A. Peled and B. Liu, "A new hardware realization of digital filters," *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 22, no. 6, pp. 456–462, December 1974.

-
- [189] C. S. Burrus, "Digital filter structures described by distributed arithmetic," *IEEE Transactions on Circuits and Systems*, vol. 24, no. 12, pp. 674–680, December 1977.
- [190] T. S. Chang, C. Chen, and C. W. Jen, "New distributed arithmetic algorithm and its application to IDCT," *IEE Proceedings on Circuits, Devices and Systems*, vol. 146, no. 4, pp. 4–19, August 1999.
- [191] K. P. Lim and A. B. Premkumar, "A modular approach to the computation of convolution sum using distributed arithmetic principles," *IEEE Transactions on Circuits and SystemsII: Analog and Digital Signal Processing*, vol. 46, no. 1, pp. 92–96, January 1999.

Appendix A

Design Entry Tool Suite and Software Packages

A.1 Handel-C

Prof. Ian Page, Oxford University Computing Laboratory: *“It really got its name because I think Handel was a rather good composer ...”*

Research into the field of hardware compilation for FPGAs started in 1991, when Ian Page and Wayne Luk (Hardware Compilation Group at the Programming Research Group (PRG) within the Oxford University Computing Laboratory (OUCL)), developed a compiler that transformed a subset of Occam into a netlist suitable for loading onto an FPGA [181]. Nearly ten years later we have seen the development of Handel-C, the first commercially available high-level language available for targeting FPGAs.

Established in 1996, Celoxica Ltd, was previously known as Embedded Solutions Ltd (ESL), is responsible of the marketing of Handel-C [161].

Handel-C aimed at compiling high level algorithms directly into gate level hardware. In order to support the use of the language the vendor also supplies a graphical design environment called DK (Design Kit) (Fig. A.1) that incorporates simulator, debugger, compiler and implementation generation, in EDIF, VHDL or Verilog [182]. Handel-C is essentially an extended subset of the standard ANSI-C Language. Be-

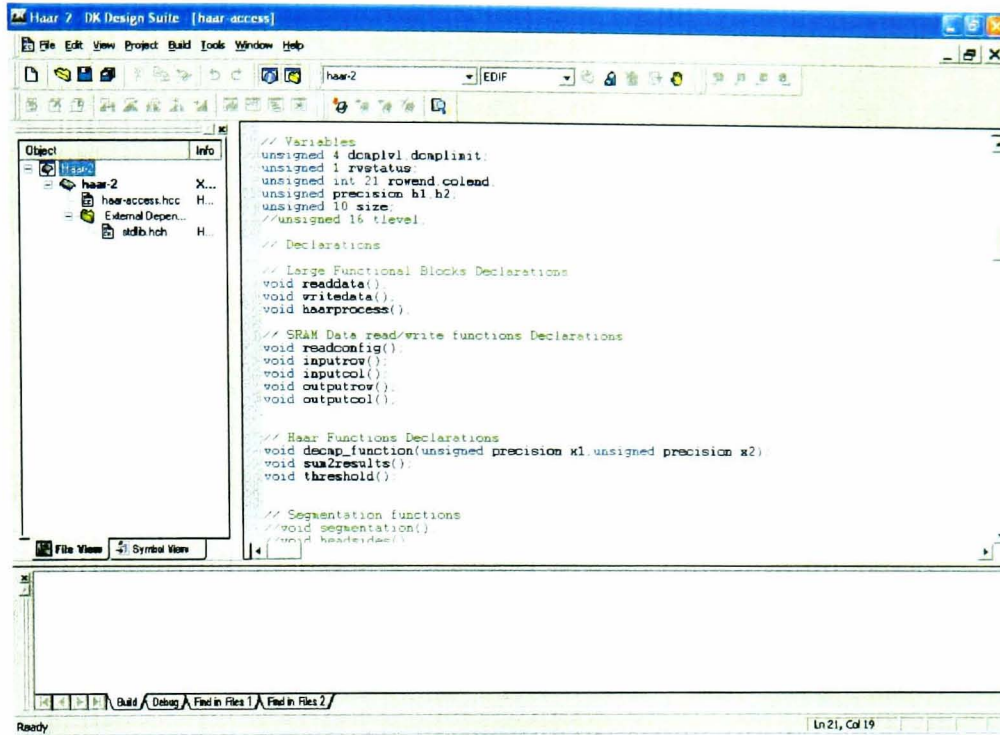


Figure A.1: The DK design synthesis tool for entering Handel-C code

cause standard C is a sequential language, Handel-C has additional constructs to support the parallelization of code and to allow fine control over what hardware is generated. Figure A.2 shows the most important additional features.

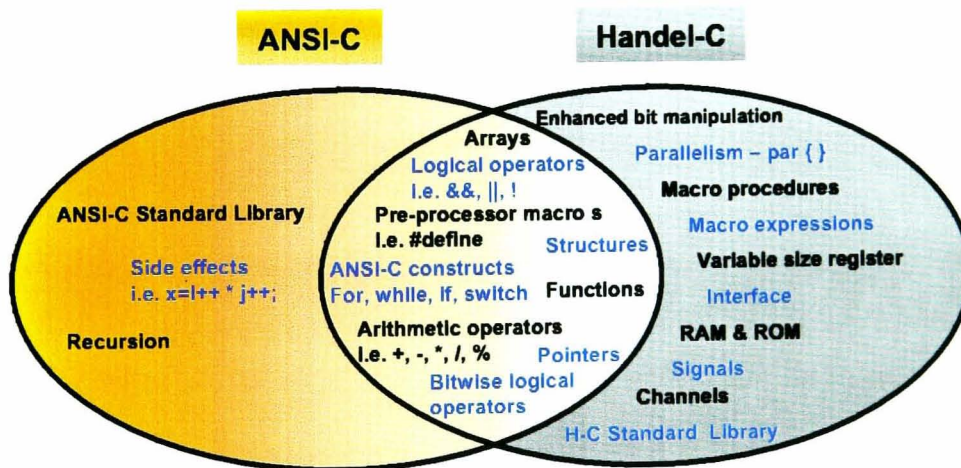


Figure A.2: Handel-C/ANSI-C comparison

DK produces a Netlist file, which is used during the place and route stage to generate the image or bitstream file as shown in Figure A.3.

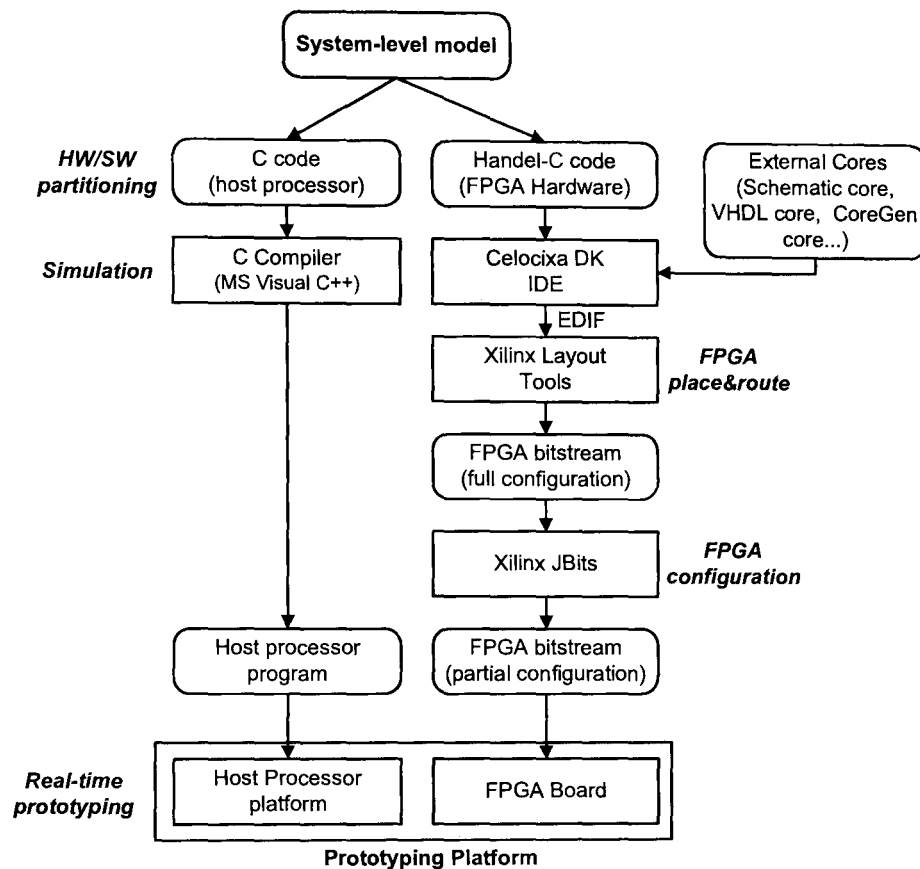


Figure A.3: Handel-C design flow

Unlike other C to FPGA tools which rely on going via several intermediate stages, Handel-C allows hardware to be directly targeted from software, allowing a more efficient implementation to be created. The language is designed around a simple timing model that makes it very accessible to system architects and software engineers. The advantages of Handel-C over the other HDLs is:

- The rapid prototyping: several works [183–185] have shown that Handel-C

shortens design time by a factor of 3-4 times with approximately the same operating speed compared to traditional HDLs.

- The simulator: the Handel-C simulator can display contents and the status of all variables (registers) in a program or design for every clock-cycle. The timing semantics of the simulator is straight forward, in that assignments require exactly one clock-cycle and expressions or control logic consume no cycles to evaluate. The simulator can visualise the actual timing behaviour as number of cycles over simulation steps. For convenience, programmers can also visualise the source code executed at each clock-cycle as well as the program state at any time.

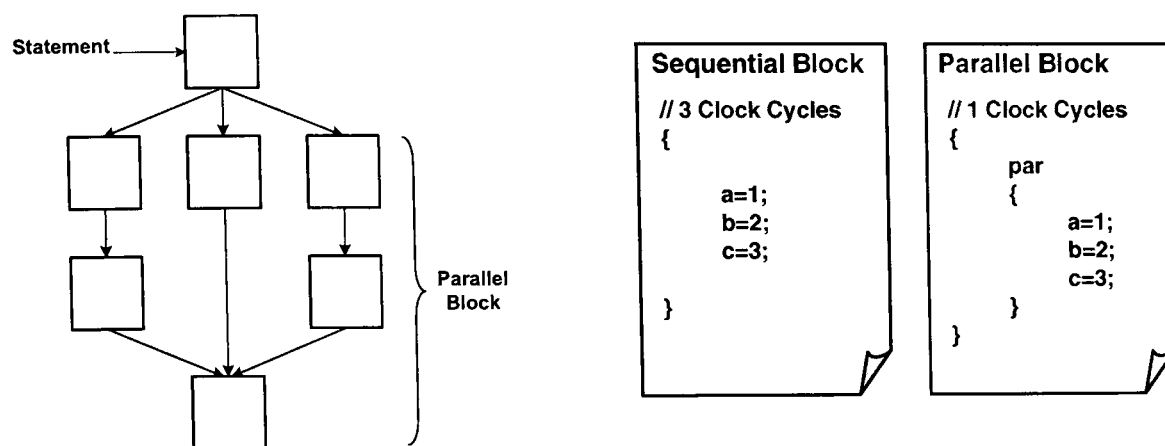
This appendix is not intended to be a full description of the tool and language, but it does describe the most important features, especially those that influence the design decisions described in this thesis. For full details of the language and development environment the reader is referred to the user guides and reference material from the manufactures.

A.1.1 Parallel Hardware Generation

One of the advantages of using hardware is the ability to exploit parallelism directly. Handel-C has additional constructs to support the parallelisation of code using the *par{ }* statement. When instructed to execute two instructions in parallel, those two instructions will be executed at exactly the same instant in time by two separate pieces of hardware. When a parallel block is encountered, execution flow splits at the start of the parallel block and each branch of the block executes simultaneously. Execution flow then re-joins at the end of the block when all branches have completed. Any branches that complete computation are forced to wait for the slowest branch before continuing as shown in Figure A.4 [38].

A.1.2 Variables

Handel-C has one basic variable type which is integer, it can be signed or unsigned with any width (Fig. A.5). Variables are mapped to hardware registers [38].



(a) Parallel branch execution flow

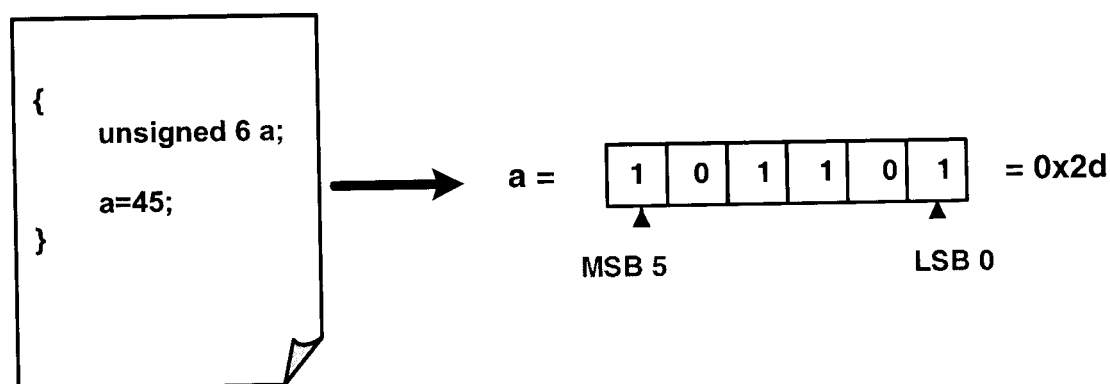
(b) *PAR* construct exampleFigure A.4: The *PAR* construct

Figure A.5: Variables declaration example

A.1.3 Bit Level Operators

Extra operators have been added in Handel-C to allow more "hardware" like bit manipulation.

A.1.4 Channel Communications

Channels provide a link between parallel branches. One parallel branch outputs data onto the channel and the other branch reads data from the channel. Channels also provide synchronisation between parallel branches because the data transfer can only complete when both parties are ready for it. If the transmitter is not ready for the communication then the receiver must wait for it to become ready and vice

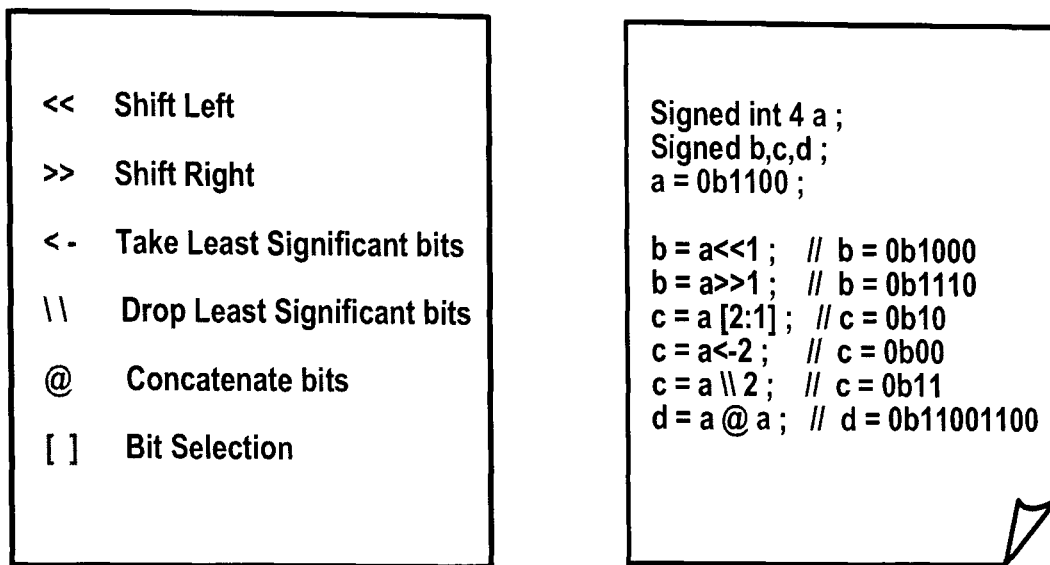


Figure A.6: Bit level operators in Handel-C

versa. In Figure A.7, the channel is shown transferring data from the left branch to the right branch. If the left branch reaches point *a* before the right branch reaches point *b*, the left branch waits at point *a* until the right branch reaches point *a* [38].

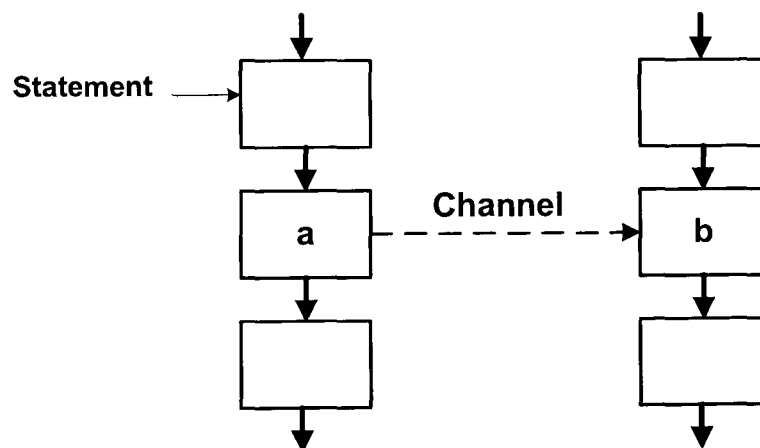


Figure A.7: Channel communication

A.1.5 Memory

RAMs and ROMs can be implemented directly using the keywords *ram* and *rom* respectively. Handel-C allows access to a number of different types of RAM:

- Distributed RAM, which is implemented in look-up tables in the logic blocks of the FPGA

- Block RAM, which is available on certain chips, can be identified by specifying the *block* parameter in conjunction with the *ram* keyword
- Off-chip RAM [38]

Normal variables are implemented as flip-flops. The Handel-C code for RAM declaration is shown in Figure A.8.

```
ram unsigned 8 DR[256];  
// Distributed RAM Declaration  
  
ram unsigned 8 BR[256] with {block=1};  
// Block RAM Declaration
```

Figure A.8: RAM/ROM declaration in Handel-C

A.1.6 External Communication

Communication between the hardware and the outside world is performed using interfaces. These may be specified as input or output, and, as with assignment, a write-to or a read-from an interface will take one clock cycle. The language allows the designer to target particular hardware, assign input and output pins, specify the timing of signals, and generally control the low level hardware interfacing details. Macros are available to help target particular devices [38].

There are three types of interfaces:

- Buses: used for connecting to external pins
- Ports: used for creating connection points for external logic e.g. Creating the ports for VHDL entity
- User Defined: used for including external logic blocks inside a Handel-C design

A.2 Xilinx-ISE

ISE controls all aspects of the design flow for Xilinx FPGAs. Through the Project Navigator interface, you can access all of the design entry and design implementation tools. ISE is a toolsuite that combines a number of distinct design implementation sub-tools under a common platform called “Project Navigator” Fig. A.9. The different design entry steps in ISE are described in the following subsections.

A.2.1 Specifying Design Options

The Process Properties dialog box shown in Fig. A.10 provides access to the Translate, Map, Place and Route, Simulation, and Timing Report properties. A series of categories, each containing properties for a different aspect of design implementation can be seen.

A.2.2 Design Translation

During translation, the NGDBuild program performs the following functions:

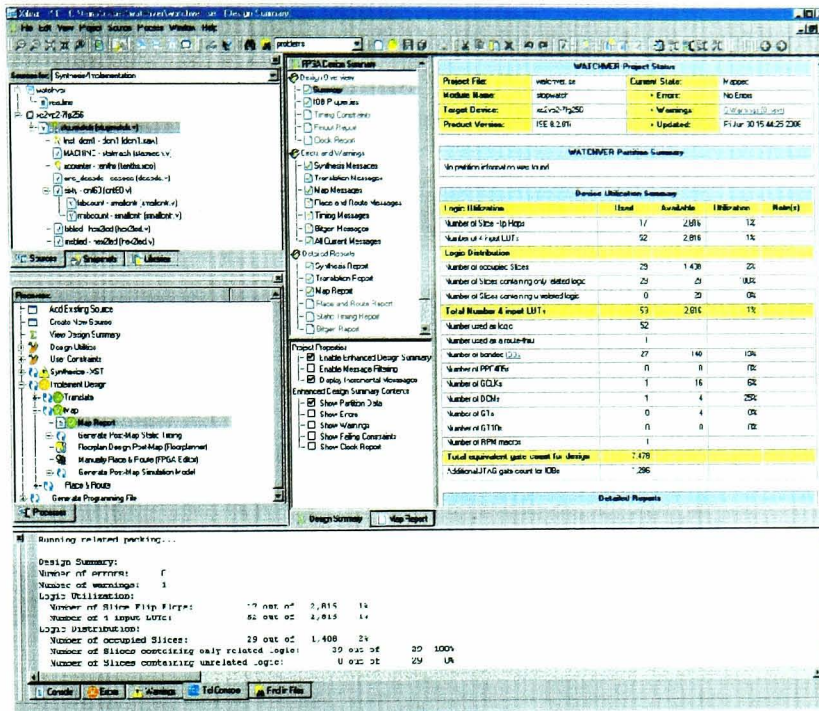


Figure A.9: Sample window displaying ISE Project Navigator

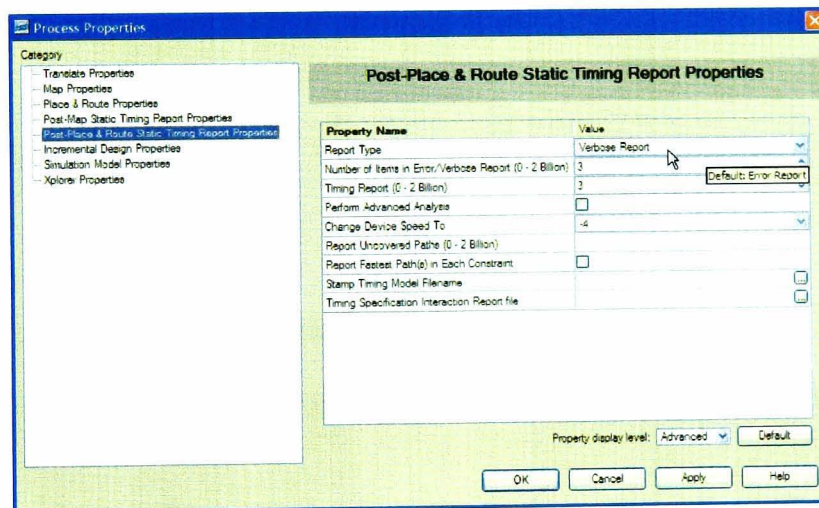


Figure A.10: Setting the design options in ISE

- 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;
- Performs timing specification and logical design rule checks; and
- Adds the User Constraints File (UCF) to the merged netlist.

A.2.3 Timing Constraints

The User Constraints File (UCF) provides a mechanism for constraining a logical design without returning to the design entry tools. The Constraints Editor and PACE are graphical tools that enable entry of timing and pin location constraints. The global tab of the timing constraints editor is presented in Fig. A.11. This window automatically displays all the clock nets in the design, and enables the definition of associated period, pad to setup, and clock to pad values. Many of the internal names will vary depending on the design flow and synthesis tool used.

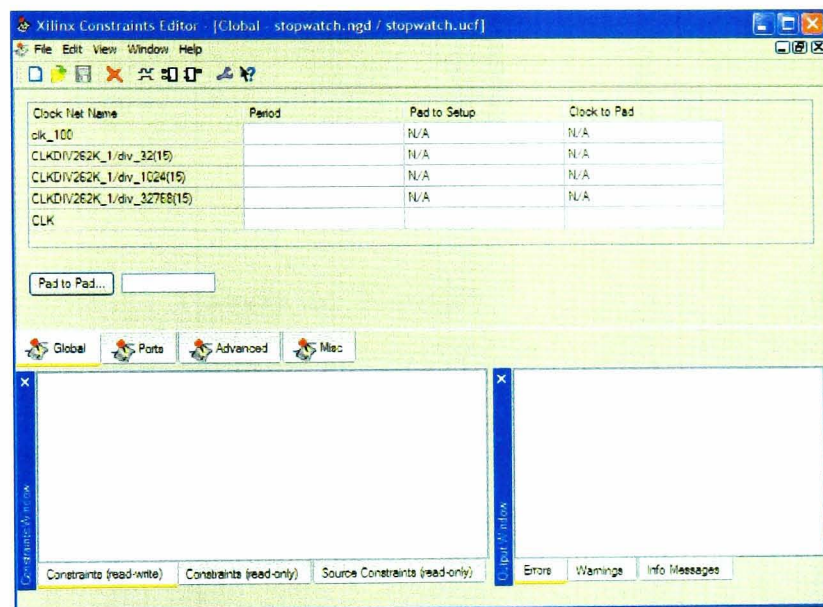


Figure A.11: Setting constraints in the timing editor

A.2.4 Setting Pin and Location Constraints

PACE (Fig. A.12) is used to add and edit the pin locations and area group constraints defined in the NGD file. PACE generates a valid UCF file. The Translate step uses this UCF file, along with the design source netlists, to produce a newer NGD file. The NGD file incorporates the changes made in the design and the UCF file from the previous section. PACE also places Global Logic at the Slice level with Block Ram, Digital Clock Managers (DCMs), Gigabit Transceivers (GTs), and BUFGs.

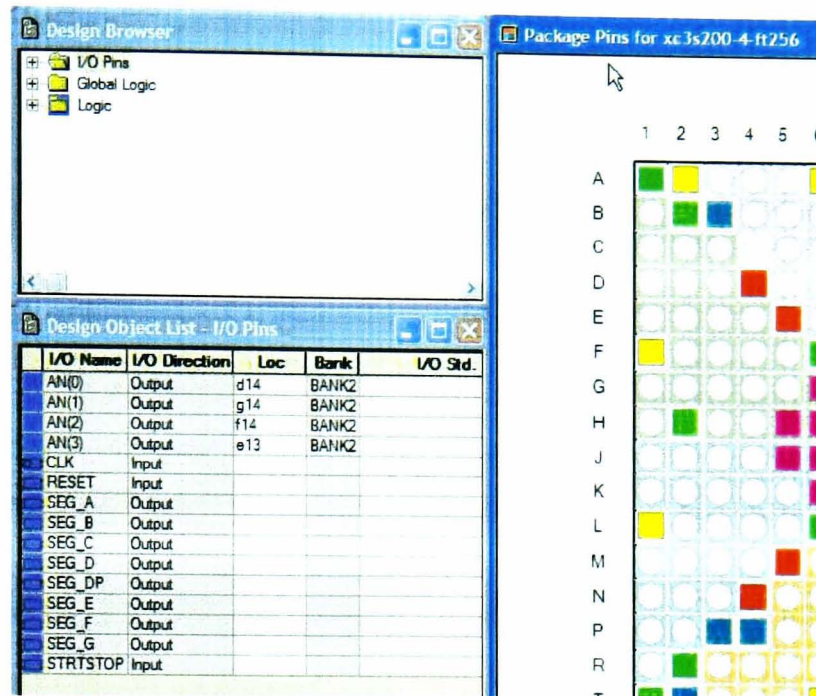


Figure A.12: PACE

A.2.5 Mapping, Place & Route

Now that 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 place-and-route algorithms is performed during the Place & Route (PAR) process:

- Timing Driven PAR PAR is run with the timing constraints specified in the input netlist and/or in the constraints file; and
- Non-Timing Driven PAR PAR is run, ignoring all timing constraints.

A.2.6 Verification of Place & Route

The FPGA Editor reads and writes Native Circuit Description (NCD) files, Native Macro Circuit (NMC) files and Physical Constraints Files (PCF). It performs the following tasks:

- Place and route critical components before running the automatic place-and-route tools;

- Finish placement and routing if the routing program does not completely route your design;
- Add probes to the design to examine the signal states of the targeted device;
- View and change the nets connected to the capture units of an Integrated Logic Analyser (ILA) core in the design;
- Run the BitGen program and download the resulting bitstream file to the targeted device; and
- View and change the nets connected to the capture units of an Integrated Logic Analyser (ILA) core in your design.

A snapshot of the FPGA editor showing a section of the placed and routed design is shown in Fig. A.13.

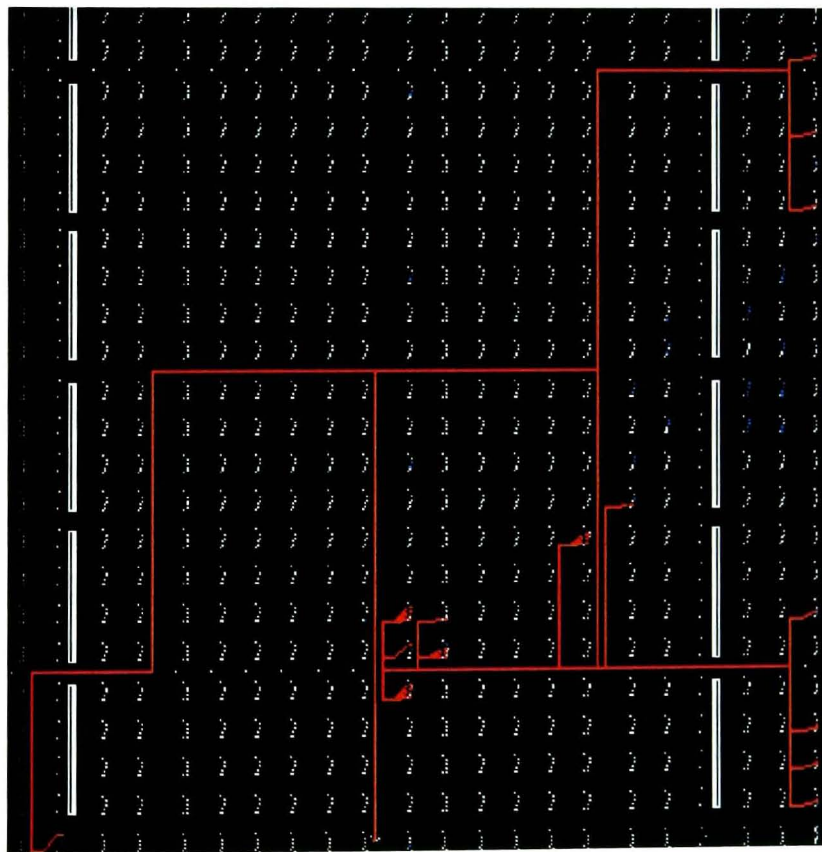


Figure A.13: FPGA Editor showing a section of the clock tree in a design

A.2.7 Estimating and analysing Power using ISE XPower

XPower (Fig. A.14) enables interactive analysis of total device power and power per-net for routed, partially routed or unrouted designs. XPower uses device knowledge and design data to estimate device power and by-net power utilisation. Information is presented in both text and HTML report formats. Circuit signal space information is supplied to XPower using either VCD files or by setting activity rates for the various nets using explorer/table views.

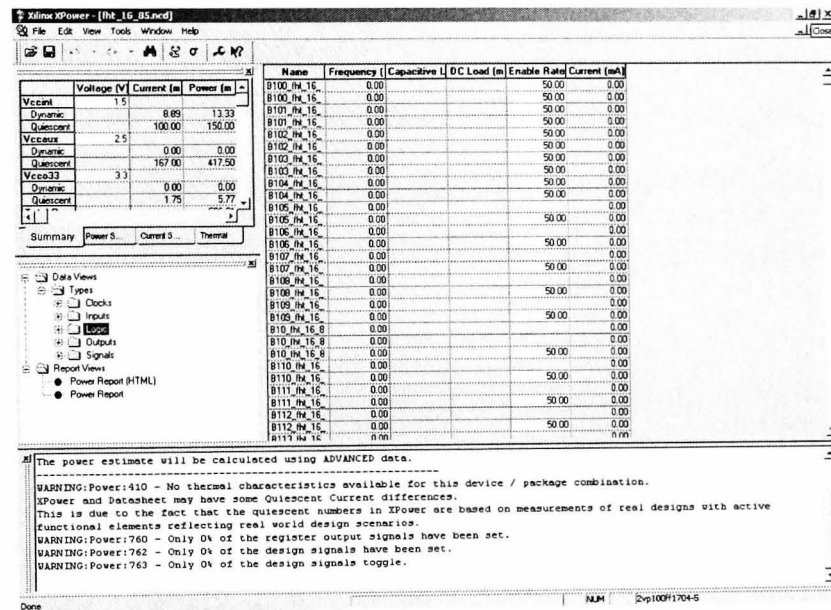


Figure A.14: XPower

A.3 NLREG

The goal of regression analysis is to determine the values of parameters for a function that cause the function to best fit a set of data observations that you provide. NLREG [164] is a very powerful regression analysis program which can be used to perform multivariate, linear, polynomial, exponential, logistic, and general nonlinear regression. For complex analyses, NLREG enabled specification of function models using conditional statements (IF, ELSE), looping (FOR, DO, WHILE), work variables and arrays.

The basis for the minimisation technique used by NLREG is to compute the sum of the squared residuals for one set of parameter values and then slightly alter

each parameter value and recompute the sum of squared residuals to see how the parameter value change affects the sum of the squared residuals. By dividing the difference between the original and new sum of squared residual values by the amount the parameter was altered, NLREG is able to determine the approximate partial derivative with respect to the parameter. This partial derivative is used by NLREG to decide how to alter the value of the parameter for the next iteration.

If the function being modelled is well behaved and the starting value for the parameter is not too far from the optimum value, the procedure will eventually converge to the best estimate for the parameter. This procedure is carried out simultaneously for all parameters and is, in fact, a minimisation problem in n-dimensional space, where 'n' is the number of parameters. A sample computation window of NLREG showing a section of the power data to be modelled, and regression results is shown in Fig. A.15.

The screenshot shows the NLREG software interface. The title bar reads 'C:\Pwr Parallel\NLR - NLREG - Nonlinear Regression Analysis'. The menu bar includes 'File', 'Edit', 'Show', 'Run', 'View', 'Evaluate', 'Save-plot', 'Colors', and 'Help'. The main window contains the following text:

```

Title "My Equation";
Variables CP,v,TA,t;
Parameters c1,c2,c3,c4;
Function CP = c1*(v^2)*(TA)*(t) + c3*(t) + c4;
data;
11.63  1.80 215.00  75.00
10.08  1.80 215.00  65.00
 8.53  1.80 215.00  55.00
 6.98  1.80 215.00  45.00
 5.43  1.80 215.00  35.00
 3.88  1.80 215.00  25.00
11.32  1.80 1008.00  40.00
 9.91  1.80 1008.00  35.00
 8.49  1.80 1008.00  30.00
 7.08  1.80 1008.00  25.00

```

Below the data, the following regression statistics are displayed:

```

Maximum allowed number of iterations = 500
Convergence tolerance factor = 1.000000E-010
Stopped due to: Both parameter and relative function convergence.
Number of iterations performed = 8
Final sum of squared deviations = 1.5681281E+002
Final sum of deviations = -1.3118280E-007
Standard error of estimate = 3.23329
Average deviation = 1.69207
Maximum deviation for any observation = 10.7369
Proportion of variance explained (R^2) = 0.2935 (29.35%)
Adjusted coefficient of multiple determination (Ra^2) = 0.1993 (19.93%)
Durbin-Watson test for autocorrelation = 1.612
Analysis completed 13-Mar-2007 00:32. Runtime = 0.09 seconds.

```

At the bottom, there is a section for 'Descriptive Statistics for Variables' with a table header:

Variable	Minimum value	Maximum value	Mean value	Standard deviation
----------	---------------	---------------	------------	--------------------

The table content is partially obscured by a scroll bar. At the bottom left, there is a button labeled 'Edit program source' and a 'NUM' field.

Figure A.15: NLREG

Appendix B

Hardware Platforms for Synthesis & Implementation

B.1 RC1000 Prototyping Platform

The RC1000 hardware platform, used in this thesis, is a standard PCI bus plug-in card for PCs equipped with a Xilinx XCV2000E-6 Virtex-E FPGA chip. It has 8 MBytes of SRAM directly connected to the FPGA in four 32-bit wide memory banks for data processing operations, a programmable clock and 50 auxiliary I/Os. All four memory banks are accessible by the FPGA and any device on the PCI bus. The FPGA has two of its pins connected to the clocks. One pin is connected to either a programmable clock or an external clock. The programmable clocks are programmed by the host PC, and have frequency range of 400 KHz to 100 MHz [116]. A schematic block diagram of the RC1000 board is shown in Figure B.1.

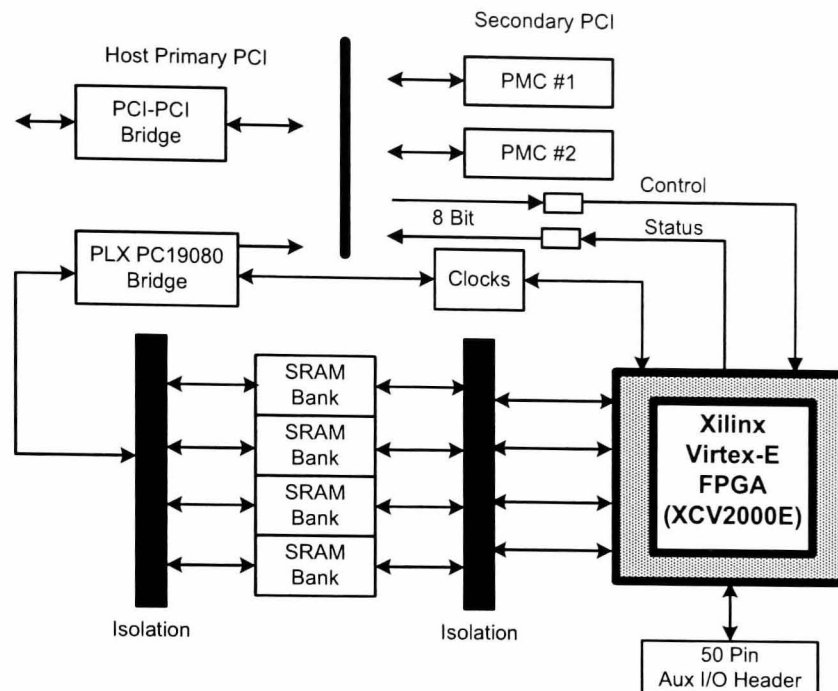


Figure B.1: RC1000 functional block diagram

The RC1000 board is supported with a macro library (the PP1000 library,) that simplifies the process of initialising and talking to the hardware. This library comprises driver functions with the following functionality:

- Initialisation and selection of a board
- Handling of FPGA configuration files
- Data transfer between PC and the RC1000 board
- Function to help with error checking and debugging these library functions can be included in a C or C++ program that runs on the host PC and performs data transfer via the PCI bus [38]

Direct Memory Access (DMA), data buffering and clock speed control make the RC1000 suitable for high-speed applications.

B.1.1 Host-FPGA Communication

Using the statically linked C library supplied and corresponding Handel-C macro functions, a programmer has three methods of communicating a host program with an FPGA across an RC1000 development board. These are:

- Single bit signalling using two pins on the FPGA;
- Single byte data transfers using the control/status ports on the RC1000; and
- Bulk data transfers using the DMA controller and four banks of SRAM.

The first method can be used to signal a state to the FPGA or to the host.

The second method can be used to send short control messages to the FPGA or short status messages from the FPGA.

The third method is recommended for large data transfers. Each bank of SRAM can be granted to either the host or FPGA at any one time. When a memory bank is granted to the host, the DMA controller can transfer data between the host memory and the SRAM memory bank. When a memory bank is granted to the FPGA, it can access the memory to read source data transferred by the host or fill in processed data to be read by the host. Functions that implement the second method block until the corresponding read/write has taken place at the other end and can therefore be used to synchronise the swapping of ownership of a memory bank. Under this model, the sequence shown in Figure B.2 could be used to transfer a 32-bit word of data from the host to the FPGA:

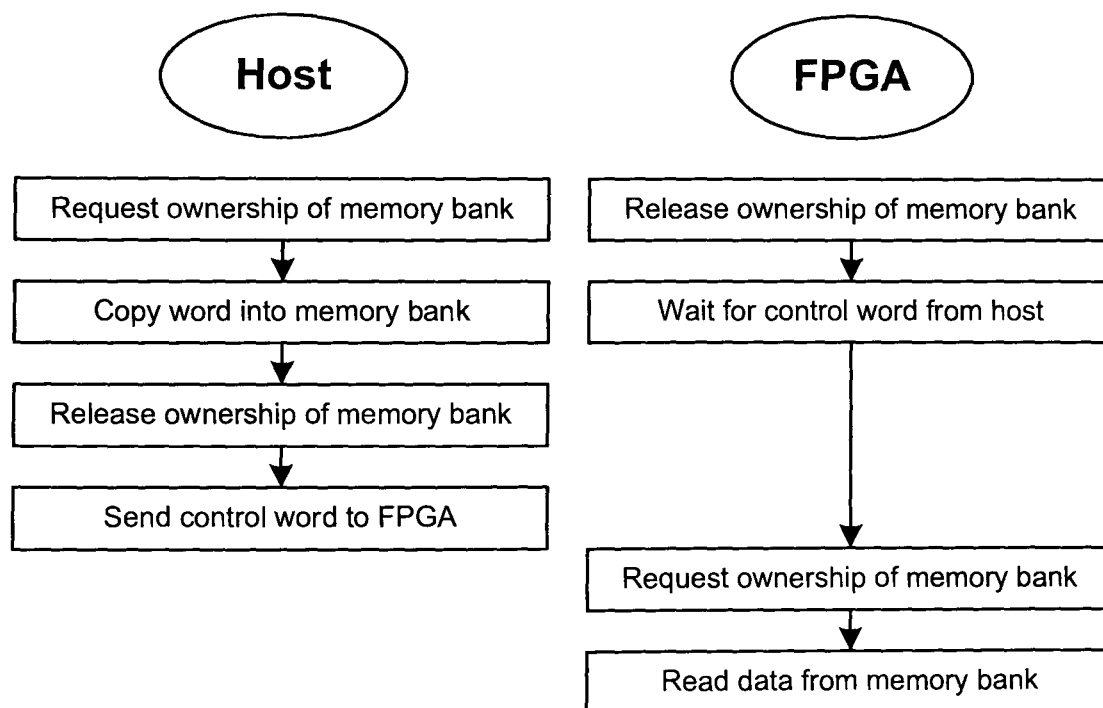


Figure B.2: Host-FPGA communication

B.2 FPGA Devices Used in This Research

A variety of FPGA devices have been used in the course of this research. The choice of the device used to prototype and synthesise results for specific work packages depends on a number of research and analysis factors:

Prototyping Platform The RC1000 and RC203 prototyping platforms that have been used to verify and implement the cores that have been proposed and modelled contain the Virtex-E and Virtex-II FPGA chips respectively;

Comparison with Existing Work Validation of performance measures for various IP cores that have been developed with respect to existing work requires a fair comparison approach. Hence, where necessary and possible, the developed designs have been resynthesised on platforms identical to those used in comparative work.

Power Modelling Validation FLPAM models are platform independent - confirmation of this key characteristic of FLPAM necessitated the use of multiple FPGA devices;

Power vs Process Technology To show relative power efficiencies of older FPGA platforms such as Virtex-E with latest and low power variants such as the Spartan-3L and Virtex-IV FPGAs.

A brief description of the hardware structures and resources available on these FPGAs is presented in the following subsections.

B.2.1 Virtex-E FPGA

Virtex-E FPGA produced by Xilinx, Inc. is used for most of our implementations. The Virtex-E FPGA architecture has two major configurable elements: Configurable Logic Blocks (CLBs) and Input/Output Blocks (IOBs). CLBs provide the functional elements for constructing logic. IOBs provide the interface between the package pins and the CLBs. The Virtex-E FPGA also has dedicated block memories called Block SelectRAM memories (BRAMs). The Virtex-E belongs to the Virtex family of

FPGAs which features regular arrays of CLBs arranged in columns surrounded on all sides by IOBs (Figure B.3) [26].

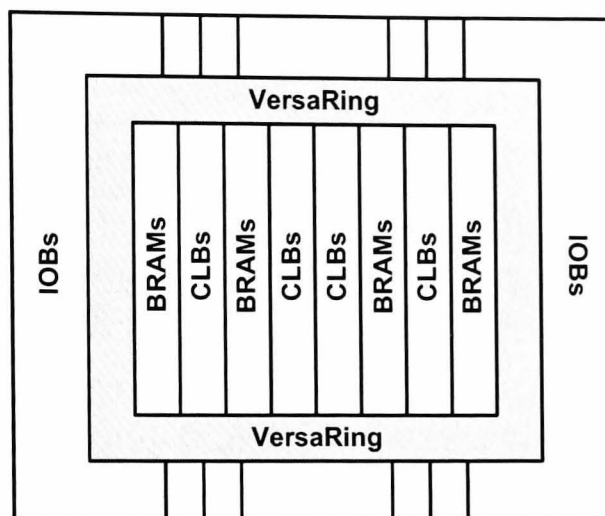


Figure B.3: Virtex-E architecture overview

The VersaRing I/O interface provides additional routing resources around the periphery of the device. This routing improves I/O routability and facilitates pin locking. The interconnection within them is very versatile as the wire segments are of varying lengths and the programmable switches are fast and placed in locations that allow them to efficiently connect these wire segments. Interconnection of CLBs is through a General Routing Matrix (GRM) as shown in Figure B.4. The GRM contains routing switches that connect the vertical and horizontal routing channels. Each CLB nests into a VersaBlock that connects the CLBs to the GRM. Virtex FPGAs are SRAM-based. A design is implemented by loading configuration data into their internal memory cells. The values stored in static memory cells control the configurable logic elements and the interconnect resources. These values load into the memory cells on power-up and can reload if necessary to change the function of the device [26].

Configurable Logic Block (CLB)

The basic building block of the Virtex-E CLB is the Logic Cell (LC). A Virtex-E CLB contains four LCs. An LC contains a four-input function generator, carry logic, and a storage element. The entire Virtex CLB is made of two CLB slices,

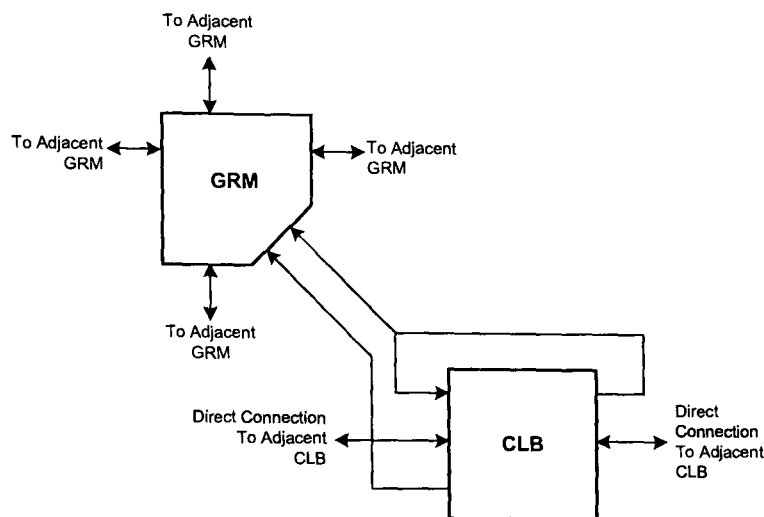


Figure B.4: Virtex-E local routing

each containing two LCs. Figure B.5 illustrates the various components of the Virtex-E CLB. The output from the function generator in each LC drives both the CLB output and the D Flip-Flop (FF). Four input Look-Up-Tables (LUTs) with 16 locations in each LUT implement function generators. A function is implemented in a LC by loading data into the LUT. The input into the LC is an address into the LUT. The value stored at that address is the output of the LC. Two LUT within a slice can be combined to create a 16×2 - bit or 32×1 - bit synchronous RAM, or a 16×1 dual-port synchronous RAM. LUT can also work as 16-bit shift register, which can be used to store data in high speed applications such as digital image processing [26].

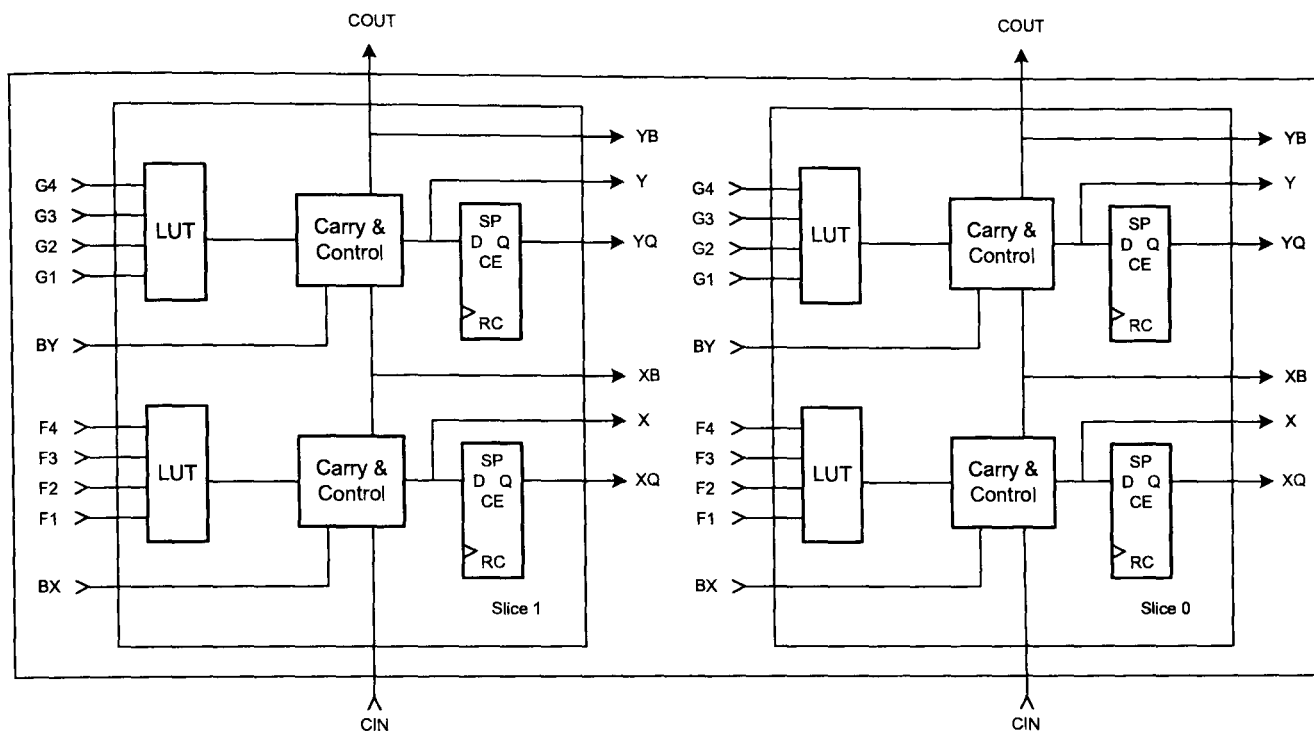


Figure B.5: 2-Slice Virtex-E CLB

Block SelectRAM (BRAM)

Virtex-E FPGAs incorporate large Block SelectRAM (BRAM) memories. These complement the Distributed SelectRAM memories that provide shallow RAM structures implemented in CLBs.

There is one such memory column between every twelve CLB columns. The BRAM also includes dedicated routing to provide an efficient interface with both CLBs and other BRAM. Each BRAM is a fully synchronous dual-ported and can store 4096 bits. Each port has independent control signals so that the two ports can be configured independently. The width of each addressable location can vary from 1 to 16 bits. For example, if each location is 16-bits wide, then there will be 2564 such locations within one BRAM memory [26].

B.2.2 Virtex-II FPGA

Virtex-II is was the successor of the Virtex-E FPGA series. it uses $0.15\mu\text{m}$ process with eight layers of metal at 1.5V power supply. In addition to advancements

in its process technology, Virtex-II is the first Xilinx FPGA with fully buffered interconnect, which may be considered as a turning point in its routing architecture. Figure B.6 shows the 2v40 FPGA, which is the smallest member of Virtex-II family.

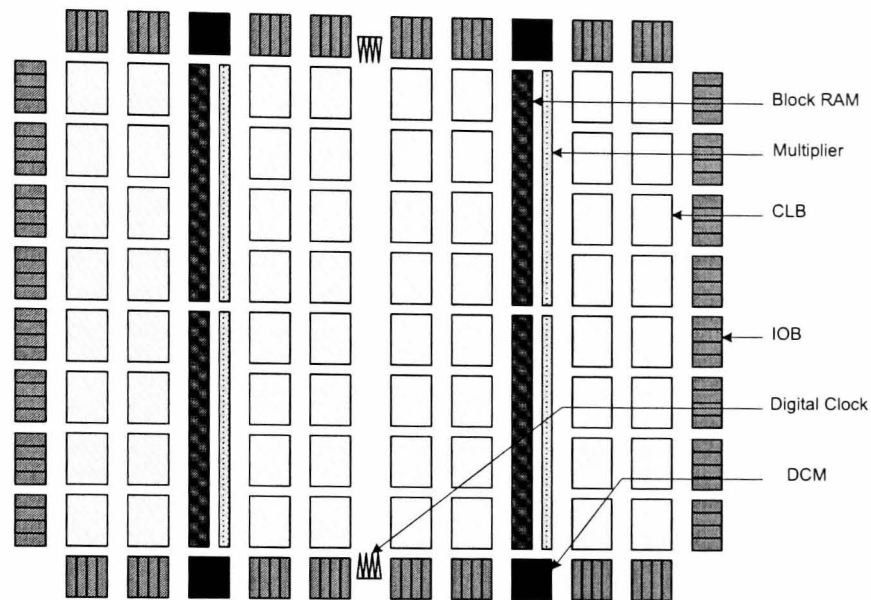


Figure B.6: Virtex-II platform FPGA

As shown in Figure B.6, Virtex-II includes four major elements organised in a regular array [36].

- CLBs provide functional elements for combinatorial and synchronous logic, including basic storage elements. Each Virtex-II CLB contains four slices, where each slice consists of two 4-input LUTs, two FFs, and a variety of dedicated circuitry to accommodate more efficient implementation of some specific logic.
- BRAM memory modules provide large $18Kbit$ storage elements of dual-port RAM.
- Multiplier blocks are 18×18 bits dedicated multipliers.
- DCM (Digital Clock Manager) blocks provide self-calibrating, fully digital solutions for clock distribution delay compensation and clock multiplication and division.

B.2.3 Virtex-II Pro FPGA

The Virtex-II Pro family is based on the Virtex-II technology and contains platform FPGAs for designs that are based on IP cores and customised modules. The family incorporates multi-gigabit transceivers and PowerPC CPU blocks (see Figure B.7). It empowers complete solutions for telecommunication, wireless, networking, video and DSP applications [186].

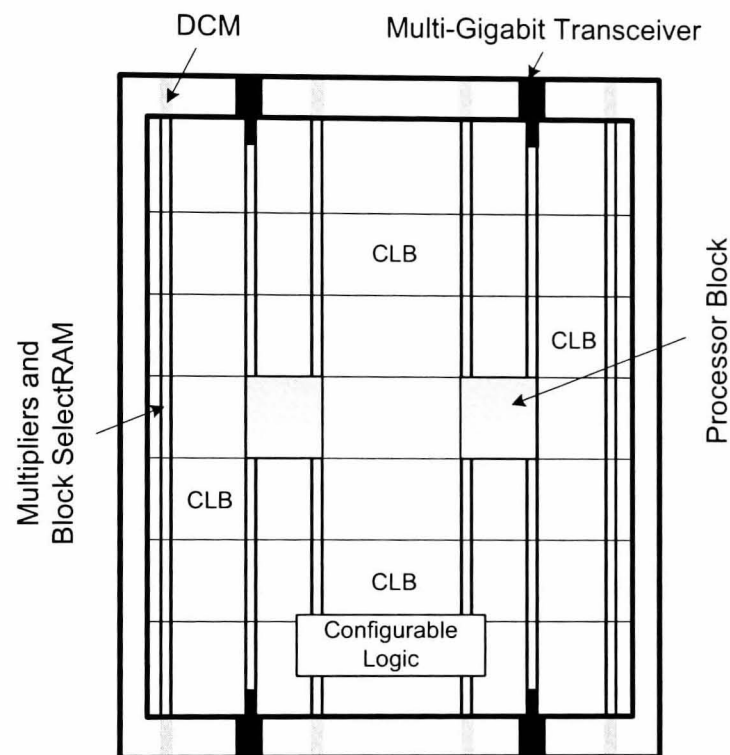


Figure B.7: Virtex-II Pro FPGA platform

Within the Virtex-II Pro Processor Block, which is the main additional feature, there are four components:

- Embedded IBM PowerPC 405-D5 RISC CPU core;
- On-Chip Memory (OCM) controllers and interfaces;
- Clock/control interface logic; and
- CPU-FPGA Interfaces

B.2.4 Spartan-3L FPGA

Spartan-3L FPGAs B.8 consume less static current than corresponding members of the standard Spartan-3 family. Spartan-3L devices provide the identical function, features, timing, and pinout of the original Spartan-3 family. Features include programmable I/Os, CLBs, RAM blocks, DCMs, and Multiplier blocks. Another power-saving benefit of the Spartan-3L family beyond static current reduction is the Hibernate mode, which lowers device power consumption. Additional features include MicroBlaze processor, PCI, and other cores.

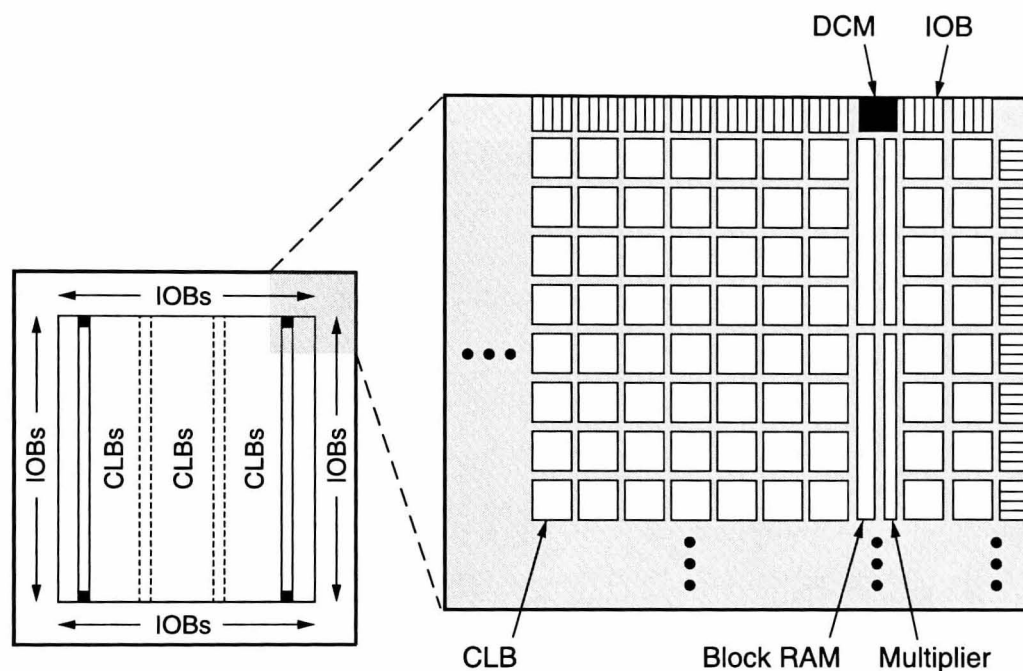


Figure B.8: Spartan-3L Pro FPGA platform

B.2.5 Virtex-4 FPGA

The Virtex-4 [10] is a recent high performance FPGA manufactured using 1.2v, 90nm triple-oxide technology. Additional features in comparison to older generation FPGAs include 80 independent clocks and 20 DCMs, Multi-Gigabit Serial I/O, Auxiliary Processor Unit (APU) controller providing a low-latency link from the embedded PowerPC core to custom hardware accelerator functions, and upto 500 MHz clocking performance. The Virtex-4 FPGA series consists of different package configurations with varying combinations of feature sets. Key device capabilities of the full featured platform are shown in Fig. B.9.

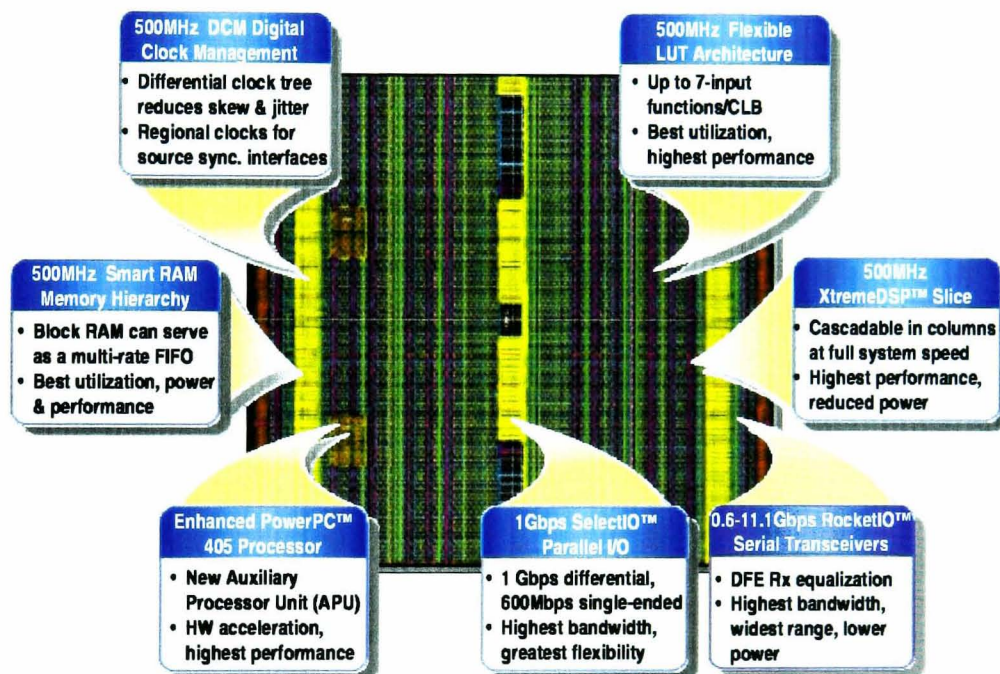


Figure B.9: Virtex-4 complete feature set with the chip in background

Appendix C

Algorithms and Arithmetic

C.1 The Finite Radon Transform and its Extension to other Higher Dimensional Generalisations of Wavelets

Recently, the curvelet and ridgelet transforms [31–33] have been generating a lot of interest due to their superior performance over wavelets. While wavelets have been very successful in applications such as denoising and compact approximations of images containing zero dimensional (point) singularities, they do not isolate the smoothness along edges that occurs in images because they lack flexible directionality. Wavelets are thus more appropriate for the reconstruction of sharp point-like singularities than lines or edges. These shortcomings of wavelets are well addressed by the ridgelet and curvelet transforms, as they extend the functionality of wavelets to higher dimensional singularities, and are effective tools to perform sparse directional analysis. The basic building block of these transforms is the FRAT [35].

C.1.1 The Continuous Radon Transform

The 2D Radon transform of an image in spatial domain is defined as the line integral of each projection of the image taken at regular rotational intervals given by $x \cdot \cos\theta$. For an image represented by rectangular coordinates, the Radon Transform is a set

of projections of the image taken by integrating along the set of lines defined by $x\cos\theta + y\sin\theta = d$ for $0 \leq \theta \leq \pi$, where θ is the angle of the line with respect to the positive y axis and d corresponds to the distance from the origin. Mathematically, it is expressed as:

$$R(\theta, t)[f] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \delta(x \cos \theta + y \sin \theta - t) dx dy \quad (\text{C.1.1})$$

The projection slice theory gives us a mathematically elegant and computationally efficient inversion called the Filtered Back Projection. This approach is more efficient, as the Radon domain image is first filtered, and then back projected. The filtered back projection can be mathematically obtained as follows:

$$f(x, y) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{d}{dy} H[R(\theta, y - tx)] d\theta \quad (\text{C.1.2})$$

C.1.2 The Finite Radon Transform and its Inverse- A Primer

The FRAT was first introduced in [148] as the finite analogue of integration in the continuous radon transform, with origins in the field of combinatorics. The mathematical representation of an injective form of the FRAT to ensure invertibility when applied on finite Euclidian planes has been presented in [35]. A pseudocode for the implementation of FRAT has also been provided in [35], and is reproduced in Codeblock 2. In all previous implementations of the FRAT, it has become convention to label an architecture based on the straightforward implementation FRAT pseudocode as the “*reference architecture*”.

It is worth mentioning that the FRAT is not a discretised version of the RT, but a discrete finite version. Consider a cyclic group Z_p denoted by $Z_p = (0, 1, \dots, p - 1)$ 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 subgroups, given by:

$$L_{k,l} = \{(i, j) : j = (ki + l)(\text{mod } p), i \in Z_p\}, \quad k < p \quad (\text{C.1.3})$$

Algorithm 2 Pseudocode for the FRAT

```

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

```

and

$$L_{p,l} = \{(l, j) : j \in Z_p\} \quad (C.1.4)$$

where each subgroup $L_{k,l}$, 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 then given by:

$$r_k[l] = FRAT_f(k, l) = \frac{1}{\sqrt{p}} \left(\sum_{(i,j) \in L_{k,l}} f[i, j] \right) \quad (C.1.5)$$

From Eq. C.1.3, C.1.4 and C.1.5, it can be seen that the function f is treated as a periodic function, and hence the digital representation of the line displays a wrap around effect, as illustrated in Fig. C.1.

Fig. C.1 represents all the set of lines corresponding to the FRAT function for blocksize $p = 7$ and $k = 4$. There are $p + 1$ vectors in the FRAT domain, each corresponding to one “direction”. The line corresponding to $k = 4, l = 4$ corresponds to the 4th digital line in the 4th rotational direction. The concept of “periodicity”

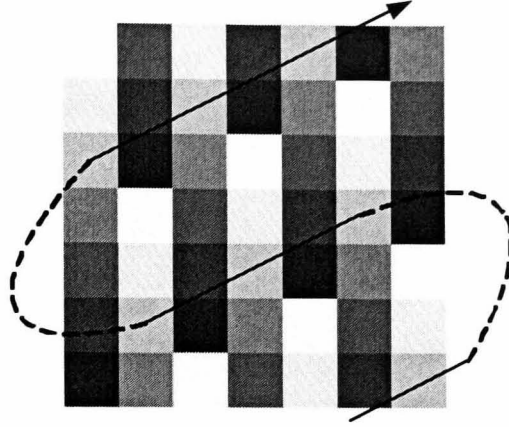


Figure C.1: FRAT basis function (projection kernel) for blocksize $p = 7$ and $k = 4$. The digital line superimposed over the kernel corresponds to $l = 4$

is highlighted for the specific case where $l = 4$, and is superimposed over the kernel in Fig. C.1. We can see that there are three (Euclidian) parallel lines that actually constitute a *single* digital line, connected by the dots. This is known as the 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 FBP is given by:

$$FBP_r(i, j) = \frac{1}{\sqrt{p}} \left(\sum_{(k,l) \in P_{i,j}} r_k[l] \right), (i, j) \in Z_p^2 \quad (C.1.6)$$

where

$$P_{i,j} = \{(k, l) : l = (j - ki)(\text{mod } p), k \in Z_p\} \cup \{(p, i)\} \quad (C.1.7)$$

Substituting Eq. C.1.5 in C.1.6, we get:

$$FBP_r(i, j) = \frac{1}{p} \left(\sum_{(k,l) \in P_{i,j}} \sum_{(k',l') \in P_{i,j}} f[i', j'] \right) \quad (C.1.8)$$

$$= f[i, j] \quad (C.1.9)$$

C.1.3 Ridgelets and Curvelets

The continuous ridgelet transform [33] of a bivariate function $f(x)$ is given by:

$$RT = \int_{R^2} \psi_{a,b,\theta}(x) f(x) dx \quad (\text{C.1.10})$$

However, the use of digital images necessitates the development of suitable variations of the ridgelets to deal with images in the digital domain. An orthonormal, invertible and discrete form of the ridgelet, called finite ridgelet transform was first proposed in [31]. THE FRIT is obtained by performing the DWT on each FRAT projection sequence with fixed value of k . This process is pictorially represented in Fig. C.2.

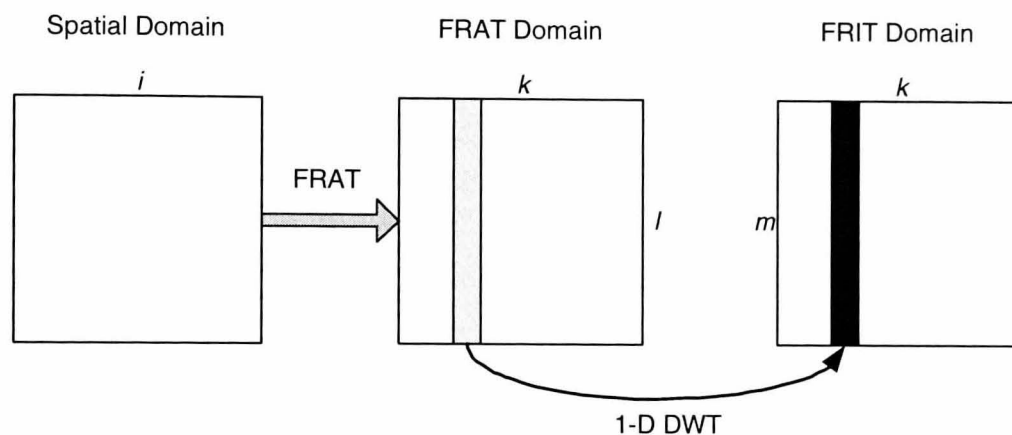


Figure C.2: Finite Ridgelet transform obtained by performing DWT on the FRAT vectors

The finite and discretised variant of curvelets can be obtained by repeated application of the FRIT. The directional attributes of these higher dimensional generalisations of wavelets make them ideal for a number of applications such alternate image representation, compression, denoising, etc.

C.2 Colour Spaces and Conversion Formulae

A colour in the RGB colour space is converted to the YCrCb colour space using the following equation:

$$\begin{pmatrix} Y \\ Cr \\ Cb \end{pmatrix} = \begin{pmatrix} 0.257 & 0.504 & 0.098 & 16 \\ 0.439 & -0.368 & -0.071 & 128 \\ -0.148 & -0.291 & 0.439 & 128 \end{pmatrix} \times \begin{pmatrix} R \\ G \\ B \\ 1 \end{pmatrix} \quad (\text{C.2.11})$$

While the inverse conversion can be carried out using the following equation:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1.164 & 1.596 & 0.0 & -222.912 \\ 1.164 & -0.813 & -0.392 & 135.616 \\ 1.164 & 0.0 & 2.017 & -276.8 \end{pmatrix} \times \begin{pmatrix} Y \\ Cr \\ Cb \\ 1 \end{pmatrix} \quad (\text{C.2.12})$$

Consider an $N \times M$ image (N : image height, M : image width). The steps in the conversion of this image between RGB and YCrCb spaces is presented in Fig. C.3

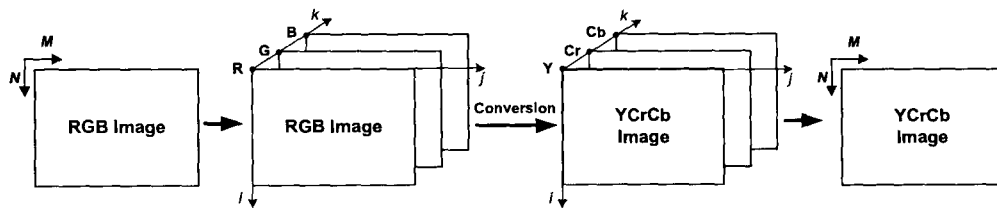


Figure C.3: RGB to YCrCb conversion

C.3 Distributed Arithmetic

DA was first presented in [187] and later reported in [188, 189]. “Distributed arithmetic is used to design bit-level architectures for vector-vector multiplications. In distributed arithmetic, each word in the vectors is represented as a binary number, the multiplications are reordered and mixed such that the arithmetic becomes ‘distributed’ through the structure. Distributed arithmetic is commonly used for implementation of convolution operations and Discrete Orthogonal Transforms” [57].

C.3.1 Suitability of DA for FPGAs

ROM-based DA uses a ROM table to store the pre-computed data, which makes it regular and efficient in the use of silicon area, in a VLSI implementation [57]. The advantage of a DA based ROM approach is its efficiency of implementation. The basic operations required are a sequence of ROMs, addition, subtraction and shift operations of the input data sequence. All of these functions are efficiently mapped to FPGA structures [11].

C.3.2 Mathematical Formulation of DA

Consider an inner product of two vectors A and B of length N .

$$Y = \sum_{k=1}^N A_k B_k \quad (\text{C.3.13})$$

where B_k is represented in 2's complement binary notation and is defined as $B_k : b_{k0}; b_{k1} \dots; b_{k(W-1)}$ such that B_k has a wordlength W and b_{k0} is the sign bit.

$$B_k = -b_{k0} + \sum_{n=1}^{W-1} b_{kn} 2^{-n} \quad (\text{C.3.14})$$

Substituting Eq. C.3.14 in Eq. C.3.13, we get:

$$Y = \sum_{k=1}^N A_k \left[-b_{k0} + \sum_{n=1}^{W-1} b_{kn} 2^{-n} \right] \quad (\text{C.3.15})$$

By rearranging the above expression, we get:

$$Y = \sum_{n=1}^{W-1} \left[\sum_{k=1}^N A_k b_{kn} \right] 2^{-n} + \sum_{k=1}^N A_k (-b_{k0}) \quad (\text{C.3.16})$$

Consider Eq. C.3.16. We can see that the term $\left[\sum_{k=1}^N A_k b_{kn} \right]$ has only 2^N possible values. The term $\sum_{k=1}^N A_k (-b_{k0})$ also has 2^N possible values. This means that all possible values can now be stored in a ROM of size 2^{N+1} .

However, it is possible to reduce the size of the ROM by half by applying a simple modification. Consider the term:

$$\left[\sum_{k=1}^N A_k b_{kn} \right] = A_1 b_{1n} + A_2 b_{2n} + \dots + A_N b_{Nn} \quad (\text{C.3.17})$$

By applying the above expression, the second term in Eq. C.3.16 can be replaced as follows:

$$\sum_{k=1}^N A_k (-b_{k0}) = - \left[\sum_{k=1}^N A_k (b_{k0}) \right] \quad (\text{C.3.18})$$

Eq. C.3.16 can now be rewritten as:

$$Y = \sum_{n=1}^{W-1} \left[\sum_{k=1}^N A_k b_{kn} \right] 2^{-n} - \sum_{k=1}^N A_k (b_{k0}) \quad (\text{C.3.19})$$

This modification can be easily adapted in hardware by replacing the adder in the shift-add structure with an adder/subtractor.

C.4 ROM Size Reduction in DA

However, when the size of the inner products increases the ROM area increases exponentially and becomes impractically large, even when using ROM partitions [115, 190, 191]. A number of techniques are available to further reduce the size of the memory for DA implementation. An effective technique is Offset Binary Coding (OBC) which involves a change in the internal data representation from binary to signed-digit. However, the reduced area is traded off for additional complexity in the control circuitry. Another method for memory reduction is ROM decomposition.

The disadvantages are increase in effective latency and control circuit complexity. A complete description of these techniques is beyond the scope of this Appendix and can be referred from [57, 115].

Appendix D

Various Compound Energy Cost Functions

Power is an important metric in the design of FPGA based systems, and most studies of designs implemented on FPGAs focus on average power dissipation of the overall circuit. However, in the case of high-throughput DSP circuits, energy is a more appropriate measure to quantify the efficiency of an operation. This is because energy metrics combine power data with performance figures, and yield better insights into the efficiency of a design. Energy metrics also enable comparisons across different design parameters. Understanding energy consumption will enable the design choices that meet a specific throughput constraint while minimising power consumption.

In this work, various energy metrics have been defined and applied as appropriate for analysing different cores.

D.1 Energy per Operation

Energy per Operation (*EOP*) is a product of power and number of clock edges per operation over the frequency. The term “Operation” in *EOP* can take different meanings depending on the context. For example, in the case of a simple inner product calculation, an operation means the complete calculation of the product of one set of vectors. In an image processing / transformation algorithm, operation

indicates the completion of the processing of a single computational block (or image tile). *EOP* is useful for comparing energy efficiency of two or more circuits that employ different architectural approaches to perform the same operation. Also, it is useful for comparing circuits that require different numbers of clock cycles to complete one operation. Effectively, *EOP* is defined as:

$$\frac{1}{T} \int_T P(t) dt \quad (\text{D.1.1})$$

where $P(t)$ is the instantaneous power dissipated and T is the number of clock cycles needed to complete one operation. Assuming constant power consumption, *EOP* can be estimated by the following expression:

$$EOP = \frac{P_{av} \times l}{f} \quad (\text{D.1.2})$$

where P_{av} is the average power consumed, l is the latency of the one “operation” of the core under consideration and f is the frequency.

D.2 Energy Area

Energy Area (*EA*) is a compound cost metric, and is useful to determine the tradeoffs between energy efficiency and area occupied by a core. It is determined from *EOP* as follows:

$$EA = \frac{EOP}{s} \quad (\text{D.2.3})$$

D.3 Energy Throughput

Energy Throughput (*ET*) combines energy dissipation and the actual volume of data processed per cycle into a combined metric. In other words, it is the amount of energy dissipated per bit of data processed, and enables us to make a fair comparison between different architectures that perform the same operation at different mathematical scales. *ET* is given by the expression in Eq. D.3.4.

$$ET = EOP/(L \cdot N) \quad (\text{D.3.4})$$

D.4 Energy Per Pixel and Energy Per Frame

The following energy metrics are specific to image processing based cores. The fundamental component of a digital image is the pixel. The whole of an image is also known as a frame. Evaluating Energy Per Pixel (EPP) and Energy Per Frame (EPF) is of interest in some image processing applications where different block sizes can be chosen to perform the same transformation process on a digital image. The equation for EPP is obtained by rationalising energy per operation with respect to the number of pixels processed.

$$EPP = \frac{EOP}{b^2} \quad (\text{D.4.5})$$

where b is the blocksize (and the complete processing of a single block constitutes and “operation”, in this case)

For an image of size $M \times N$, the Energy Per Frame (EPF) is calculated as follows:

$$EPF = \frac{EOP \cdot M \cdot N}{b^2} \quad (\text{D.4.6})$$