

**ENERGY EFFICIENT HARDWARE ACCELERATION OF MULTIMEDIA  
PROCESSING TOOLS**

by

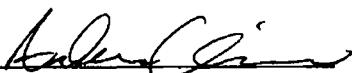
**Andrew Kinane, B.Eng.**

Submitted in partial fulfilment of the requirements  
for the Degree of Doctor of Philosophy

Dublin City University  
School of Electronic Engineering  
Supervisor: Dr. Noel E. O'Connor  
May, 2006



I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work

Signed   
Andrew Kinane (Candidate)

ID 98069098

Date 15<sup>th</sup> May 2006

# Abstract

The world of mobile devices is experiencing an ongoing trend of feature enhancement and general-purpose multimedia platform convergence. This trend poses many grand challenges, the most pressing being their limited battery life as a consequence of delivering computationally demanding features. The envisaged mobile application features can be considered to be accelerated by a set of underpinning hardware blocks. Based on the survey that this thesis presents on modern video compression standards and their associated enabling technologies, it is concluded that tight energy and throughput constraints can still be effectively tackled at algorithmic level in order to design re-usable optimised hardware acceleration cores.

To prove these conclusions, the work in this thesis is focused on two of the basic enabling technologies that support mobile video applications, namely the Shape Adaptive Discrete Cosine Transform (SA-DCT) and its inverse, the SA-IDCT. The hardware architectures presented in this work have been designed with energy efficiency in mind. This goal is achieved by employing high level techniques such as redundant computation elimination, parallelism and low switching computation structures. Both architectures compare favourably against the relevant prior art in the literature.

The SA-DCT/IDCT technologies are instances of a more general computation – namely, both are Constant Matrix Multiplication (CMM) operations. Thus, this thesis also proposes an algorithm for the efficient hardware design of any general CMM-based enabling technology. The proposed algorithm leverages the effective solution search capability of genetic programming. A bonus feature of the proposed modelling approach is that it is further amenable to hardware acceleration. Another bonus feature is an early exit mechanism that achieves large search space reductions. Results show an improvement on state of the art algorithms with future potential for even greater savings.

# Acknowledgements

This thesis would never have reached the printer but for the support and encouragement I have received from many people, some I've known for a short while, others a very long time indeed. Firstly I would like to thank Aisling, whose unwavering love has always been the key source of inspiration. I would also like to thank my parents and my sister who have been supporting me all the way through twenty plus years of schooling! Credit is due to the CDVP hardware guys, past and present, who were always willing to give encouragement and technical insights. In particular I must sincerely thank Val, who helped me through the Ph D "white nights" and convinced me that there is light at the end of the tunnel. Finally I would like to thank my supervisor, Dr Noel O'Connor, for his guidance and support from start to finish.

*Courage is resistance to fear mastery of fear not absence of fear*

– Mark Twain

## TABLE OF CONTENTS

<b>Table of Contents</b>	<b>v</b>
<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>xiii</b>
<b>List of Peer-Reviewed Publications</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 The Emergence of Mobile Multimedia	1
1.2 Grand Challenges Facing Mobile Computing	1
1.2.1 Application Development Challenges	2
1.2.2 Networking and Communication Challenges	3
1.2.3 Hardware Device Design Challenges	4
1.2.4 Motivation for Thesis	7
1.3 Research Objectives	8
1.4 Thesis Structure	8
1.5 Summary	9
<b>2 Technical Background</b>	<b>10</b>
2.1 Introduction	10
2.2 Digital Video Processing	10
2.2.1 A Generic Video Compression System	12
2.2.2 Transform Theory Overview	17
2.2.3 The Discrete Cosine Transform (DCT)	20
2.2.4 Image and Video Compression Standards Overview	23
2.2.5 Object Based Processing	26
2.2.6 The Shape Adaptive Discrete Cosine Transform (SA-DCT)	31
2.2.7 Digital Video Implementation Approaches	34
2.2.8 Conclusions	36
2.3 Low Power Design	36

2 3 1	Circuit Power Dissipation Phenomena	37
2 3 2	Power Analysis & Estimation Techniques	44
2 3 3	Low Power Design Techniques	54
2 3 4	Electronic Design Automation (EDA)	64
2 3 5	Conclusions	68
2 4	Summary	68
<b>3</b>	<b>SA-DCT Architecture</b>	<b>69</b>
3 1	Introduction	69
3 1 1	Vector Shape Parsing	70
3 1 2	Data Alignment	71
3 1 3	Variable N-point 1D DCT	71
3 1 4	SA-DCT Computational Requirements Summary	71
3 2	DCT/SA-DCT Hardware State of the Art Review	73
3 2 1	Classes of DCT Implementation Approaches	73
3 2 2	SA-DCT Specific Approaches	82
3 3	Design Methodology	85
3 4	SA-DCT Datapath Architecture	86
3 4 1	Even-Odd Decomposition	88
3 4 2	Reconfiguring Adder-Based Distributed Arithmetic Dot Product	89
3 4 3	Partial Product Summation Tree	92
3 5	SA-DCT Memory and Control Architecture	95
3 5 1	Addressing/Routing Control Logic	95
3 5 2	Transpose Memory	97
3 6	Experimental Results	100
3 6 1	Fixed-Point Variable $N$ -point 1D DCT Design	100
3 6 2	Evaluation Against Prior Art	101
3 6 3	Conformance Testing	108
3 6 4	System Prototyping	110
3 7	Possible Architectural Variations	112
3 8	Summary of Contributions	112
<b>4</b>	<b>SA-IDCT Architecture</b>	<b>114</b>
4 1	Introduction	114
4 1 1	Additional Complications Compared to SA-DCT	115
4 2	IDCT/SA-IDCT Hardware State of the Art Review	117
4 2 1	Classes of IDCT Implementation Approaches	117
4 2 2	SA-IDCT Specific Approaches	120
4 3	SA-IDCT Datapath Architecture	122
4 3 1	Reconfiguring Adder-Based Distributed Arithmetic Dot Product	124
4 3 2	Partial Product Summation Tree	125
4 3 3	Even-Odd Recomposition	125
4 4	SA-IDCT Memory and Control Architecture	128

4 4 1	Addressing/Routing Control Logic	129
4 4 2	Transpose Memory	130
4 5	Experimental Results	131
4 5 1	Fixed-Point Variable $N$ -point 1D IDCT Design	131
4 5 2	Evaluation Against Prior Art	132
4 5 3	Conformance Testing	139
4 5 4	System Prototyping	141
4 6	Summary of Contributions	142
<b>5</b>	<b>Dot Product High Level Synthesis</b>	<b>143</b>
5 1	Introduction	143
5 2	Multiplication by Constant Applications	144
5 2 1	Preliminaries	144
5 2 2	Single Constant Multipliers	145
5 2 3	Constant Multiplier Blocks	145
5 2 4	Digital Filters	146
5 2 5	The Constant Matrix Multiplication (CMM) Problem	149
5 3	Multiplication by Constant State of the Art Review	151
5 3 1	CMM – A Closer Look	151
5 3 2	Optimisation Approaches	153
5 3 3	Proposed Optimisation Approach	158
5 4	The CMM Optimisation Algorithm	162
5 4 1	Signed-Digit Permutation Extraction Stage	162
5 4 2	Dot Product Level (DPL) Stage	164
5 4 3	CMM Level (CMML) Stage	176
5 4 4	CMML – A Genetic Programming Approach	178
5 4 5	Genetic Algorithm Parameter Selection	185
5 4 6	CMML Results	187
5 5	MCMM Extension	188
5 6	Summary of Contributions	190
<b>6</b>	<b>Conclusions &amp; Future Work</b>	<b>192</b>
6 1	Conclusions	192
6 1 1	Motivation for Proposed Research – A Summary	192
6 1 2	Summary of Thesis Contributions	193
6 1 3	Research Objectives Achieved	195
6 2	Future Work	196
6 2 1	SA-DCT/IDCT Accelerator Variations and Improvements	196
6 2 2	SA-DCT/IDCT Accelerator Pre/Post Processing	196
6 2 3	SA-IDCT Power Efficiency Enhancements	200
6 2 4	CMM Optimisation Algorithm Improvements	201
6 2 5	Hardware Accelerator Adaptability	203
6 3	A Vision for the Future	205

<b>A Leaf-Labelled Complete Rooted Binary Trees</b>	<b>208</b>
A 1 Introduction	208
A 2 The Problem	208
A 2 1 Problem 1 – Valid Patterns	209
A 2 2 Problem 2 – Topologies	210
A 2 3 Problem 3 – Addend Permutations	210
A 3 The Solution	216
A 3 1 Number of Non-Isomorphic Binary Bracket Structures of $n$ -Operand Operations	217
A 3 2 Number of Leaf-Labelled Complete Rooted Binary Trees	219
<b>List of Acronyms</b>	<b>221</b>
<b>List of MPEG Meeting Contributions</b>	<b>225</b>
<b>List of Patents</b>	<b>227</b>
<b>Bibliography</b>	<b>228</b>



## LIST OF FIGURES

1 1	Power Density of Silicon Versus Process Technology According to Intel	6
1 2	Power Savings at Various Levels of Design Abstraction	6
2 1	The 4 2 0 Image Format	12
2 2	A Generic Video Compression Codec	13
2 3	Intra Mode Coding Flow	13
2 4	Zigzag Scan Order	14
2 5	A Generic Video Decoder	15
2 6	Motion Estimation and Compensation	16
2 7	DCT-II Basis Functions for $N = 8$	22
2 8	Basis Images for 2D DCT-II with $N = 8$	22
2 9	Evolution of Video Coding Standards by the ITU-T and ISO/IEC	24
2 10	Mobile Telephony Application – Human Face is Object of Interest	28
2 11	Example Face Detection Based on Colour Filtering	29
2 12	Three Successive VOPs with Alpha Masks of a VO	29
2 13	A Generic Object-Based Video Compression Codec	30
2 14	Various BAB Types	30
2 15	The SA-DCT Algorithm Step-by-Step	33
2 16	Dynamic Charging and Discharging a Capacitive Load	38
2 17	CMOS Inverter Transfer Characteristic and Short-Circuit Current	39
2 18	Diode Leakage Current in a CMOS Inverter	41
2 19	Sub-threshold Current as a Function of Gate Voltage	41
2 20	Degenerated Voltage Level Driver	42
2 21	A Pseudo-NMOS NOR Gate	42
2 22	Static vs Dynamic Power	43
2 23	Using Power [W] as a Design Metric	44
2 24	Energy Delay Product (EDP) Solution Space	45
2 25	NMOS Transistor	50
2 26	Simple Multiply-Accumulate Circuit	54

2 27	An Example of a Pipelined System	58
2 28	Sample Comparator Circuit	59
2 29	Sample Parallel Comparator	59
2 30	Sample Pipelined Comparator	59
2 31	Reducing Glitching by Balancing Signal Paths	61
2 32	Low Power Design Flow	64
3 1	The SA-DCT Algorithm Data Addressing	70
3 2	Mismatch Between Data Memory Index and Data Transform Index	71
3 3	The SA-DCT Basis Functions	72
3 4	Taxonomy of DCT Implementation Approaches	74
3 5	Sample ROM-based DA Architecture	79
3 6	Conceptual NEDA Architecture	81
3 7	Recursive Architecture (Le et al )	83
3 8	Feed-Forward Architecture (Le et al )	83
3 9	Reconfigurable Processor Architecture (Tseng et al )	84
3 10	Programmable Processor Architecture (Chen et al )	84
3 11	Auto-Aligning Architecture (Lee et al )	85
3 12	Top-Level SA-DCT Architecture	87
3 13	SA-DCT Vertical Raster Data Scanning	87
3 14	Even-Odd Decomposition (EOD) Architecture	90
3 15	Multiplexed Weight Generation Module (MWGM) Architecture	91
3 16	Simple 4 1 MUX	92
3 17	Data Optimised 4 1 MUX	92
3 18	General Counter and Compressor	93
3 19	Partial Product Summation Tree (PPST) Behavioural Architecture	93
3 20	Original Alignment of PPST Weights	94
3 21	First Sign Extension Exploit	94
3 22	Reformulated PPST Weights	94
3 23	Pre-addition of Constants	94
3 24	Second Sign Extension Exploit	94
3 25	Optimised PPST Logic	94
3 26	Addressing & Control Logic (ACL) Architecture	96
3 27	Sample SA-DCT ACL Timing Diagram	98
3 28	TRANspose Memory (TRAM) Architecture	99
3 29	Example SA-DCT TRAM Addressing	99
3 30	Example SA-DCT TRAM Write	99
3 31	SA-DCT IP Core Verification Platform	100
3 32	Sample Object Reconstruction with $Q = -10$ and $T = 11 0$	101
3 33	Andy	105
3 34	Coastguard Object 1	105
3 35	Container Object 1	105

3 36	Hall Monitor Object 2	105
3 37	Sean	105
3 38	Weather	105
3 39	MPEG-4 Part7 SA-DCT ARM920T Power Consumption	107
3 40	MPEG-4 Part 7 Fast SA-DCT Energy Characteristics on StrongARM SA-1100	108
3 41	MPEG-4 Conformance Integration Hardware	108
3 42	ARM Virtual Socket Platform	111
4 1	The SA-IDCT Algorithm Data Addressing	115
4 2	Sample SA-DCT Shape Parsing	116
4 3	Sample SA-IDCT Shape Parsing	116
4 4	Probabilities of Nonzero Occurrence for $8 \times 8$ DCT Coefficient Blocks	119
4 5	Auto-Aligning Architecture (Hsu et al )	121
4 6	Top-Level SA-IDCT Architecture	123
4 7	SA-IDCT Vertical Alpha Scanning and Horizontal Coefficient Scanning	123
4 8	Even-Odd Recomposition (EOR) Architecture	125
4 9	SA-IDCT ACL Architecture	129
4 10	SA-IDCT ACL Data Processing Pipeline	129
4 11	SA-IDCT TRAM Architecture	131
4 12	SA-IDCT IP Core Verification Platform	132
4 13	Andy	136
4 14	Coastguard Object 1	136
4 15	Container Object 1	136
4 16	Hall Monitor Object 2	136
4 17	Sean	136
4 18	Weather	136
4 19	MPEG-4 Part7 SA-IDCT ARM920T Power Consumption	138
4 20	MPEG-4 Part 7 Fast SA-IDCT Energy Characteristics on StrongARM SA-1100	139
5 1	Single Constant Multiplier Behavioural Architecture	145
5 2	Constant Multiplier Block Behavioural Architecture	146
5 3	FIR Filter Behavioural Architecture	146
5 4	FIR Sub-Expression Selection Example	148
5 5	Example FIR Architecture Using Sub-Expressions	148
5 6	Simple FIR Sub-Expression Selection Options	148
5 7	Simple FIR Sub-Expression Selection Options	149
5 8	Dot Product Behavioural Architecture	150
5 9	CMM Distributed Signed Digit 3D Matrix	150
5 10	P1D Architecture	152
5 11	P2D Architecture	152
5 12	CMML Divide and Conquer	153
5 13	Sample Acyclic Graph of a Constant Multiplier Block	154
5 14	Sample P1D Architecture	160

5 15	Sample P2D Architecture	160
5 16	Summary of the CMM Algorithm	163
5 17	Architecture Options Implied by Example SOP	168
5 18	Permutation SOP Building Finite State Machine	169
5 19	Node Combination at DPL	170
5 20	DPL Unique Implementation Skip List	172
5 21	DPL Result Stacking Prior to CMML Processing	177
5 22	CMML Processing	177
5 23	Example CMML Search Space Reduction	178
5 24	Boltzmann Decision Based Simulated Annealing	182
5 25	Uniform Crossover Example	184
5 26	Mutation Example	185
5 27	Binomial Distribution Function	186
5 28	Re-Aligned Distribution	186
5 29	SA-DCT as an MCMM Problem	189
5 30	Time-MUX Addends & PPST	190
5 31	Time-MUX PPST	190
6 1	Regular Adder and Guarded Evaluation Adder	201
6 2	Sample Images Captured by the Philips FluidFocus Lens	204
6 3	Near Focused Image	205
6 4	Far Focused Image	205
6 5	Rough Foreground Mask	206
6 6	Post-Processed Foreground Mask	206
6 7	Segmented Foreground Object	206
A 1	Multi-Operand Addition	209
A 2	Adder Topology Options for Different Number of Addends	210
A 3	Hardware Associativity and Commutativity Examples	211

## LIST OF TABLES

2 1	Weighting Factors for $K=4$ on the StrongARM Processor	47
3 1	SA-DCT $N$ -point Data Decomposition	89
3 2	Total Number of Unique Possibilities for Each Distributed SA-DCT Weight	92
3 3	SA-DCT Core IEEE 1180-1990 Results for $Q = -12, f = 4$	101
3 4	SA-DCT 90nm TSMC Synthesis Results and Benchmarking	102
3 5	Available MPEG-4 Standard Test Sequences	104
3 6	SA-DCT 90nm TSMC Power Consumption	104
3 7	Normalised SA-DCT Power and Energy Benchmarking	106
3 8	Software SA-DCT StrongARM SA-1100 Energy Statistics	107
3 9	Normalised Power and Energy Benchmarking Against Software SA-DCT	108
3 10	SA-DCT FPGA Synthesis Results	110
4 1	Total Number of Unique Possibilities for Each Distributed SA-IDCT Weight	125
4 2	SA-IDCT EOR Ping Pong Data Ordering	128
4 3	SA-IDCT Core IEEE 1180-1990 Results for $Q = -12, f = 4$	132
4 4	SA-IDCT 90nm TSMC Synthesis Results and Benchmarking	133
4 5	SA-IDCT 90nm TSMC Power Consumption	135
4 6	Normalised SA-IDCT Power and Energy Benchmarking	137
4 7	Software SA-IDCT StrongARM SA-1100 Energy Statistics	138
4 8	Normalised Power and Energy Benchmarking Against Software SA-IDCT	139
4 9	SA-IDCT FPGA Synthesis Results	141
5 1	16-bit Daubechies D4 Full Adder Requirements	176
5 2	CMMML Genetic Algorithm Parameters	186
5 3	1D 8-point DCT/IDCT Adder Unit / Full Adder Requirements	187
5 4	Operating Conditions of Proposed CMM Optimisation Algorithm Test Cases	187
A 1	Topology 1 Options	211
A 2	Topology 2 Options	211

A 3	Unique Adder Permutations	212
A 4	The First 32 Values of $T_n$	218
A 5	The First 32 Values of $P_n$	220

## List of Peer-Reviewed Publications

N O'Connor, V Muresan, A Kinane, D Larkin, S Marlow, and N Murphy, *Hardware Acceleration Architectures for MPEG-based Mobile Video Platforms A Brief Overview*, in Proc 4th European Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS), London, April 9–11, 2003, pp 456-461

A Kinane, V Muresan, N O'Connor, N Murphy, and S Marlow, *Energy-Efficient Hardware Architecture for Variable N-point 1D DCT*, in Proc International Workshop on Power and Timing Modelling, Optimization and Simulation (PATMOS), Santorini, Greece, September 15-17, 2004, pp 780-788

A Kinane, V Muresan, and N O'Connor, *An Optimal Adder-Based Hardware Architecture for the DCT/SA-DCT*, in Proc SPIE Video Communications and Image Processing (VCIP), Beijing, China, July 12-15, 2005, pp 1414–1417

D Larkin, A Kinane, V Muresan, and N O'Connor, *Efficient Hardware Architectures for MPEG-4 Core Profile*, in Proc 9th Irish Machine Vision and Image Processing Conference (IMVIP), Belfast, Northern Ireland, August 30-31, 2005, pp 249–252

A Kinane, A Casey, V Muresan, and N O'Connor, *FPGA-Based Conformance Testing and System Prototyping of an MPEG-4 SA-DCT Hardware Accelerator*, in Proc IEEE International Conference on Field Programmable Technology (FPT), Singapore, December 11-14, 2005, pp 317–318

A Kinane, V Muresan, and N O'Connor, *Optimisation of Constant Matrix Multiplication Operation Hardware Using a Genetic Algorithm*, in Proc 3rd European Workshop on Evolutionary Computation in Hardware Optimisation (EvoHOT), Budapest, Hungary, April 10-12, 2006

A Kinane, V Muresan, and N O'Connor, *Towards an Optimised VLSI Design Algorithm for the Constant Matrix Multiplication Problem*, in Proc IEEE International Symposium on Circuits and System (ISCAS), Kos, Greece, May 21-24, 2006

D Larkin, A Kinane, V Muresan, and N O'Connor, *An Efficient Hardware Architecture for A Neural Network Activation Function Generator*, in Proc International Symposium on Neural Networks (ISNN), Chengdu, China, May 29-31, 2006

A Kinane and N O'Connor, *An Adder Based Hardware Architecture for the MPEG-4 Shape Adaptive IDCT*, in Proc IEE Irish Signals and Systems Conference (ISSC), Dublin, Ireland, 28-30 June 2006



## 1.1 The Emergence of Mobile Multimedia

Communication is arguably the most fundamental facet of human behaviour. We experience audio-visual interactions daily and they occur in all areas of our lives. Considering that this is true for people of every race, age, profession and any other qualifier one can imagine, it is easy to see why there is such a continual demand for more sophisticated communication systems. The medium used for communication has evolved with this demand due to the astonishing advancements in the electronics industry and telecommunications infrastructure over the past fifty years [1, 2, 3, 4]. For example, the mobile phone has permeated our society to the extent that it is now an essential device for daily living. According to American research firm IDC, the number of mobile phone users approached 1.4 billion in 2004 [5]. Developments have not been restricted to the audio domain, there have also been advances in the visual domain in areas such as video telephony, video conferencing as well as the development of interactive digital television [6, 7]. The most recent trend has seen audio-visual communication systems shift to mobile platforms. Indeed mobile devices are serving as a point of convergence for communication, internet and personal computing applications yielding a truly ubiquitous multi-purpose mobile computing device [8, 9].

## 1.2 Grand Challenges Facing Mobile Computing

The emergence of the convergent mobile device poses huge technical challenges especially when end users demand increasingly complex content-based interactive services on such a small piece of hardware. The challenges involved can be broadly categorised into three problems: application development, content delivery and content processing. This section briefly outlines these challenges to give context for the research undertaken in this thesis. A more technical overview and discussion of these issues is given in Chapter 2.

## 1.2.1 Application Development Challenges

The modern mobile phones now available offer many more applications than its name implies – it is no longer merely a device for making voice calls. There is a multitude of applications emerging that include: personal organiser, e-mail, mobile internet, MP3 audio codec, digital camera, video telephony, mini-camcorder, mobile gaming, mobile TV, digital radio... and the list is ever increasing! An example of such a device is the new Nokia 7710 that offers many of these features [10]. We are witnessing the emergence of a ubiquitous personal gadget that covers the entire spectrum of daily activities and their associated data [11, 12]. As such, it represents a mass market product that is an ideal technical platform for cutting edge research.

### 1.2.1.1 Mobile Computing Market Demographics

Although technophiles will always provide a market for multi-purpose gadgets, it remains to be seen whether such a convergent device will be considered necessary by the wider population, especially by people who are not technically minded. Sceptics worry about possible market disinterest due to feature overload [13]. However it is short-sighted to predict that just because there may be no perceived widespread demand for a convergent device at present, there will be no such demand in the future. To illustrate how a market can boom so quickly consider that in 1999, 12.8% of US internet users had at least one MP3 music file on their hard disks [9]. Since then, the adoption of digital audio jukebox products such as the Apple iPod has mushroomed. It is estimated that shipments of MP3 players rose by 116% in 2004 and that the total number of shipments is estimated to rise by a factor of four by 2009 [14]. Now consider that in 2004, 13% of US internet users had at least one video file (> 150MB) on their hard disks. It is reasonable to assume that there will be a similar growth in the mobile video market, considering how quickly digital cameras have become almost a standard feature in new mobile phones. Evidence of this emerging trend is clear given that Apple have recently released their fifth generation iPod, which is capable of video playback and can store up to 150 hours of video content [15].

To guarantee the widespread adoption of the convergent device, any proposed applications must be easy to use and be of comparable performance and quality to their desktop equivalent (albeit allowing for the device limitations as discussed in Section 1.2.3). One only has to consider the failure of the initial Wireless Application Protocol (WAP) service compared to the facile nature of web surfing on a desktop to see how easily the consumer can be turned off by poor application usability and quality [16]. The work in this thesis is primarily concerned with digital image and video applications on a mobile device, and for such applications to be successful the user must feel satisfied with the same usability and quality criteria. The rapid advancement of digital cameras integrated in mobile phones implies that users demand quality if they are to abandon their dedicated camera devices and trust their convergent device for quality photo capture. Samsung aim to push toward the 5-Megapixel camera phone leaving dedicated digital still cameras as devices only used by professional photographers [13]. For digital video on a mobile device to be successful, there is a need for better network infrastructure (see Section 1.2.2) and improved video compression. Video data requires compression when you consider that the bit rate for uncompressed Common Interchange Format (CIF) resolution video is approximately 4.5MB/s (see Chapter 2 for details). High compression ratios and low bit rate coding is especially necessary in the mobile domain where there is relatively low bandwidth.

### 1 2 1 2 MPEG-4 – Low Bit-rate Mobile Video Telephony

The emerging industry compression standard for low bit-rate video applications is MPEG-4 [17]. A comprehensive survey of video compression standards and techniques is given in Chapter 2. One of the essential advances achieved by MPEG-4 is that it considers a scene as being composed of distinct audio-visual objects and each is coded independently [18, 19]. This is revolutionary, particularly in terms of how the video data is coded as explained in detail in Chapter 2. One can imagine in a mobile video telephony application that only encoding and transmitting the object of interest (e.g. the human face) while ignoring the background is very bandwidth efficient (and thus the service is cheaper). This additional compression unfortunately comes at the cost of increased computational complexity, and this is particularly troublesome given mobile device limitations (see Section 1 2 3). One of the most computationally demanding algorithms at the heart of the MPEG-4 object-based compression scheme is the Shape Adaptive Discrete Cosine Transform (SA-DCT) and its inverse the SA-IDCT [20, 21]. When aiming to achieve real time video applications on mobile devices, the computational burden of the associated algorithms is a primary obstacle – especially given that the platform has limited processing capability and is battery powered. Efficient hardware acceleration implementations of the SA-DCT and SA-IDCT suitable for a mobile device are thus key requirements and are the topics of Chapters 3 and 4.

A key point to note is that an MPEG-4 codec is not technically orthogonal to the other applications emerging for the mobile device (e.g. gaming, mobile TV, etc). The envisaged applications can be considered to be built using a set of lower-level basic video processing blocks that may be termed *basic enabling technologies*. The SA-DCT/IDCT is an example of one such basic enabling technology. The SA-DCT block is used in an MPEG-4 application, but may also be used to estimate image regions of high spatial frequency to aid object-segmentation (i.e. breaking the image up into semantic regions) in a content management or security application. Similarly specific image filters can be implemented for visual quality enhancement of highly compressed video, but again suitably configured can be used for spatial segmentation, or for identifying moving objects when used in conjunction with the results of a motion estimator. Since basic enabling technologies such as the SA-DCT have great potential for re-use across many mobile applications, efficient implementation of such technologies is critical to the successful development of the convergent device.

There will always be a demand by end users for more advanced features and functionality. Users demand multi-feature integration on a reasonably small mobile platform with improved battery life. As discussed in Section 1 2 3, these are conflicting demands as far as a design engineer is concerned and as such, research in the area of mobile multimedia applications and devices is a very active field.

### 1 2 2 Networking and Communication Challenges

Mobile networks have developed significantly in recent years and more and more applications are shifting to the mobile domain. Second generation mobile devices (2G) use Global System for Mobile Communication (GSM) technology, which in its basic form offers 9.6 kb/s. Third generation (3G) networks use the Universal Mobile Telecommunications System (UMTS) which offers more than 2 Mb/s and it is estimated that 48 million 3G handsets were shipped in 2004 [5]. Most mobile communication systems around the world at the moment use 2G technology but 3G is becoming widespread. Despite the growth of data rates in mobile networks, bandwidth is still a valuable commodity, especially if the data being

carried over these networks is audio-visual data [22].

There are other network design challenges in addition to the need for higher bandwidth [23]. For “anytime anywhere” computing, next generation networks should support ubiquitous intelligent connectivity. Users need to be able to roam seamlessly between networks without needing detailed (or ideally any) knowledge at the technology level. Each device should be uniquely addressable regardless of global location – fusing the conventional idea of a telephone number and a computer internet MAC address (note IPv6 will enable this [23]). The convergent device should provide personalised content and be location aware. One could envisage a scenario that a user is abroad and the device lists nearby restaurants that serve his/her favourite dishes. Global Positioning System (GPS) technology is already appearing in consumer electronic devices and its evolution is key to the enabling of personalised, location aware technology. Next generation networks need to be robust and reliable as well as delivering excellent quality of service. The user needs assurance of privacy and security if the device is being used with sensitive data, e.g. in a mobile banking application. Clearly there are challenges facing the networking and communications community to provide the backbone that will support the envisaged convergent device. Although not tackled directly in this work, progress in this field is essential to the success of ubiquitous mobile computing.

### **1.2.3 Hardware Device Design Challenges**

The applications conceived for the convergent device as outlined in Section 1.2.1 are sometimes more artistically referred to as “e-Dreams” – the concept whereby the world starts to adapt to the wishes of the consumer wherever he/she may be [24]. This requires the convergence of computing technology, wired/wireless connectivity, non-keyboard user interfaces and distributed transducer networks. However, the technology push for these e-Dreams (“7th heaven of software”) leads to relentless CMOS scaling and the “nano-scale hell of physics” [25]. These are two paradigms that on the one hand enable each other, but on the other hand the conceptual gap between them is diverging at a Moore type rate. The e-Dream creators expect increased reliability, security, adaptivity, interoperability, battery life and so on but hardware engineers must struggle with the practical energy-flexibility-cost issues associated with deep-submicron design.

#### **1.2.3.1 Limited Interfaces**

There are many technical obstacles to implementing the “digital dream” applications that are currently evolving [25]. The convergent mobile device has some obvious limitations compared to a desktop platform e.g. limited display size, limited input capability [22]. As feature integration continues to increase, designers are facing aggressive miniaturisation requirements and the permissible footprint of each component (both software and hardware based) is very restricted if the device is to be as small and portable as possible.

#### **1.2.3.2 Limited Processing Capability**

Mobile embedded systems have limited processing power compared to a desktop. It is not feasible to use a large multi-GHz Complex Instruction Set Computer (CISC)-based processor such as the Intel Pentium4 [26] which would need a bulky cooling fan subsystem. Embedded systems tend to use Reduced

Instruction Set Computer (RISC) such as those designed by companies such as ARM and MIPS [27, 28] Operating frequencies of such processors tend to be in the order of hundreds of MHz (e.g. ARM920T processor can run up to 250MHz [29]) While it must be said that it is naive to compare processors based solely on operating frequency [27] (memory access time, instruction set parallelism etc. should all be considered), it is safe to claim that embedded processors have less “horsepower” compared to their desktop counterparts. As such, computationally demanding multimedia applications with heavy data throughput requirements pose a big problem in the mobile domain. Coupled with this issue, embedded systems also have a reduced memory capacity (both RAM and HDD) compared to desktops (RAM > 1GB and HDD > 100GB). For example the new Nokia 7710 has only 90MB of RAM and a maximum of 512MB of flash memory for storage [10]. Progress in the field of embedded memories and mobile hard disk drives is ongoing [30] and the Samsung SGH-i300 released in early 2005 comes with a 3GB HDD.

### **1.2.3.3 Limited Battery Life**

Perhaps the most pressing challenge facing mobile system designers is that of limited battery energy and short battery life. In VLSI engineering, the trade off between high speed and minimum area has traditionally been the most dominating design constraint. Power consumption issues were usually a mere afterthought and any power savings gained were considered a bonus. More recently, power has become the high priority constraint due to the ongoing trend that has seen a shift to battery operated wireless platforms for multimedia applications [31, 32, 33]. The ever-increasing device capability coupled with unremitting miniaturisation trends has had an enormous impact on circuit power consumption and requires serious consideration [34]. It is now crucial for designers to tackle this issue at every stage in the design process and at all levels of abstraction to make “digital dreams” a reality.

Technology scaling is improving continuously and showing no immediate signs of reaching fundamental limits [35]. Such improvements mean packing more functionality into a smaller area [33]. Unfortunately however, scaling causes increased power density as demonstrated in Figure 1.1 and this is a problem, particularly in a wireless application where cooling fans and subsystems are not feasible [36]. As this graph shows, the constraint for producing smaller devices in the future is not necessarily manufacturability but power consumption.

The requirement for advanced functionality coupled with portability means more power hungry computation and bigger, bulkier batteries. Batteries are usually the most dominant component of a mobile device in terms of its weight. There are two complementary ways to approach the problem of power constraints. One obvious way involves developing better battery technology since conventional nickel-cadmium batteries only provide 20Wh per pound of weight [34]. Although advances in battery technology are being made (as discussed in Chapter 2), progress is relatively slow compared to advances in technology scaling and wireless application development. Thus, in tandem with developing better batteries, hardware engineers should give prime importance to designing circuits that are as energy efficient as possible since the total battery energy is finite. Energy efficiency is critical if the intended application is as demanding as real time wireless MPEG-4 decoding and there is a constraint on the size of battery being used. Battery life has become a prime product differentiator and this has led to wide research in the field of low power design.

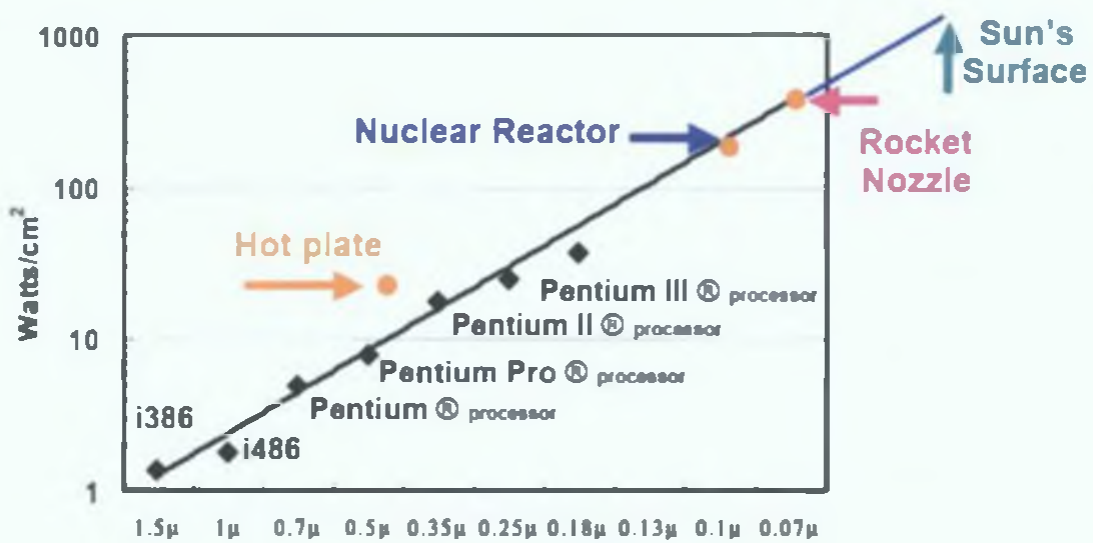


Figure 1.1: Power Density of Silicon Versus Process Technology According to Intel [33]

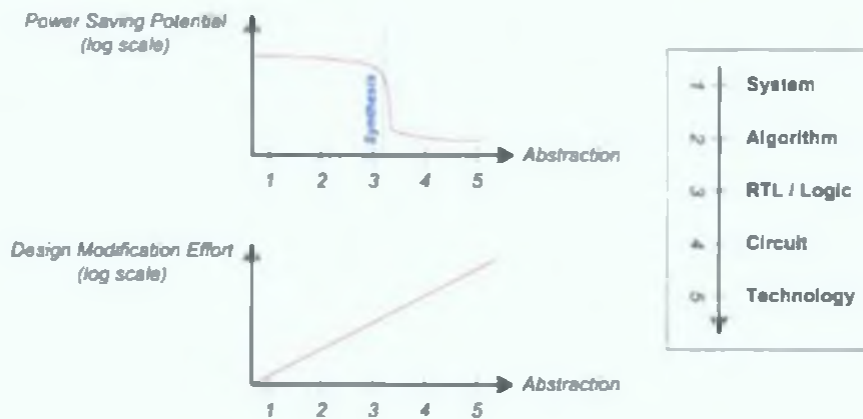


Figure 1.2: Power Savings at Various Levels of Design Abstraction

#### 1.2.3.4 Power – A Primary Design Constraint

Chapter 2 focuses in detail on why power has become a primary design constraint. The sources of power consumption in an integrated circuit are elaborated, and the common techniques used to reduce this dissipation are detailed. The overriding theme is that most potential energy savings exist at the higher levels of abstraction (as illustrated in Figure 1.2), since there exists greater degrees of design freedom [37]. This is the motivating argument for focusing on system/algorithmic level optimisations as opposed to technology/layout/circuit level tweaks when designing the SA-DCT and SA-IDCT implementations proposed in Chapters 3 and 4.

Digital video processing and compression involves a lot of complex digital signal processing (DSP). As discussed in more detail in Chapter 2, the central processor (e.g. ARM RISC) in an embedded system is invariably relieved of the most computationally demanding DSP tasks by implementing some parallel co-processor or dedicated hardware accelerator [38, 21]. As mentioned previously, silicon real estate on

the convergent device is limited so the overhead of any extra logic should be worthwhile in the sense that this extra hardware should ideally be flexible enough to be re-usable for different applications and is thus not sitting idle for long periods of time. This hardware flexibility is a key concern, especially given the range of applications being constantly dreamt up.

#### **1.2.4 Motivation for Thesis**

The grand challenges facing mobile computing mean that there is huge scope for research in this domain. The work in this thesis is focused on two of the basic enabling technologies that support mobile video applications, namely the SA-DCT and the SA-IDCT. It should be noted that both of these technologies are instances of a more general computation – both are constant matrix multiplication operations. Thus, this thesis also proposes an algorithm for the efficient hardware design of any general constant matrix multiplication equation. This algorithm can be used to design efficient hardware for a wide variety of signal processing basic enabling technologies that are needed on a convergent mobile device.

##### **1.2.4.1 Power Efficient Hardware for SA-DCT/IDCT**

As mentioned in Section 1.2.1, the SA-DCT and SA-IDCT have re-use potential since they may be leveraged in various video processing applications. As such, they are valid candidates for implementation as hardware accelerated cores designed with power as a primary design constraint. It is proposed that power should be on an equal footing with the traditional constraints of speed and area. Much of the prior art in terms of hardware to implement the SA-DCT/IDCT does not address all of the required processing steps, and indeed none of the identified techniques are designed with power as a primary design constraint. With a mobile platform in mind, the latter is very important and the SA-DCT/IDCT architectures proposed in Chapters 3 and 4 aim to fill this void.

##### **1.2.4.2 Efficient Hardware Resource Allocation for Constant Matrix Multiplication Operations**

From a more general viewpoint, many of the DSP basic enabling technologies (e.g. SA-DCT/IDCT) are dominated by the so-called “fundamental” operations of multiplication and addition [39]. More specifically, they contain many instances of “multiplication by constants” (i.e. in the product  $a \times x$  when the multiplier  $a$  is a constant and the multiplicand  $x$  is a variable). Research has shown that for a survey of more than two hundred industrial examples, over 60% have greater than 20% operations that are multiplications with constants [40]. Apart from the DCT/SA-DCT, some other common enabling technologies that use multiplication by constants include digital filters (both FIR and IIR), the Discrete Fourier Transform (DFT), colour space conversion transforms, and the Discrete Wavelet Transform (DWT) to name but a few [41]. Optimisation of these kind of multiplications will significantly impact the performance of such tasks and the global system that leverages them. Since multiplication by constant operations are widespread in DSP applications, there has been widespread research into designing efficient datapaths for such computations [42] (a detailed review is given in Chapter 5). The design problem is non-trivial and many of the previously proposed approaches are sub-optimal since they derive solutions based on greedy heuristics. Chapter 5 discusses these issues in detail and proposes a set of electronic design automation (EDA) algorithms that specifically tackle the resource allocation optimisation problem for Constant Ma-

trix Multiplication (CMM) operations. Such algorithms are important given the large amount of mobile applications that are built with fundamental CMM operations.

### 1.3 Research Objectives

The research goals of this thesis may be summarised as follows:

- 1 To design and implement energy-efficient hardware architectures for the SA-DCT and the SA-IDCT.
- 2 Verify that both architectures conform to the official MPEG-4 standard requirements by comparing the bitstreams of a software-only MPEG-4 codec to an MPEG-4 codec that computes the SA-DCT/IDCT using the proposed hardware architectures.
- 3 To derive a power and energy normalisation framework that facilitates fair benchmarking of competing architectures. Benchmarking is difficult due to the unavailability of source code for the prior art and the fact that in general other approaches are implemented in different technologies. For these reasons, some normalisations are required to justify any comparisons made.
- 4 Using the derived framework, evaluate both architectures against the prior art in the literature in terms of area, speed, power and energy.
- 5 To design and implement a hardware resource allocation algorithm for a general constant matrix multiplication operation (of which the SA-DCT/IDCT are particular cases).
- 6 Evaluate the proposed CMM algorithm against prior art proposed in the literature.

### 1.4 Thesis Structure

This thesis is structured as follows:

- *Chapter 1 – Introduction*  
The introduction (this chapter) outlines the motivations for the research proposed in this thesis providing a global context. The scope and objectives of the work are clearly outlined.
- *Chapter 2 – Technical Background*  
The technical background chapter outlines a more in-depth technical context for the work. The general topics covered are digital video processing (in particular transform based-compression and object-based processing) and low power design (power dissipation phenomena, power analysis techniques and low power design techniques). A high level justification is also given for the CMM optimisation algorithm proposed in Chapter 5.
- *Chapter 3 – SA-DCT Architecture*  
The SA-DCT chapter provides a review of prior hardware implementations for the SA-DCT. The proposed SA-DCT architecture is subsequently described in detail. This is followed by conformance testing results and evaluation against the prior art.



- *Chapter 4 – SA-IDCT Architecture*

The SA-IDCT chapter provides a review of prior hardware implementations for the SA-IDCT, with special emphasis on the distinctions between the SA-DCT and the SA-IDCT. The proposed SA-IDCT architecture is subsequently described in detail. This is followed by conformance testing results and evaluation against the prior art.

- *Chapter 5 – Dot Product High Level Synthesis*

This chapter firstly outlines the taxonomy of multiplication by constant applications to illustrate where the CMM operation sits in relation to other similar operations. This is followed by a characterisation of the dimensionality of the CMM optimisation problem parameters and a survey of prior art with these parameters in mind. Subsequently, the proposed optimisation algorithm is described in detail followed by experimental results that justify the approach taken and facilitate evaluation against relevant prior art.

- *Chapter 6 – Conclusions & Future Work*

A summary of the main results achieved in Chapters 3, 4 and 5 is presented. This is followed by a discussion of possible future research ideas to improve and/or extend the work presented in this thesis.

- *Appendix A – Leaf-Labelled Complete Rooted Binary Trees*

This appendix provides some mathematical proofs that are related to the permutation space modelling of the proposed CMM optimisation algorithm in Chapter 5. In high-level terms, the permutations are related to the number of two input adder topologies possible for a generic multi-operand addition. Then, for each topology, the task is to find how many different ways there are of mapping addends to input pins, such that the resultant injective mapping from the set of input pins to the set of 'network adder output pins' is unique. Essentially, the issue is finding out how many different associative combinations of two addends are possible while ensuring duplicates due to the commutative property of addition are not counted.

## 1.5 Summary

We are currently witnessing the convergence of many diverse applications (communication, multi-media, internet, etc.) onto a mobile platform. Users constantly demand the conflicting goals (from a system design perspective) of enhanced performance and longer battery life. Mobile digital video applications are certain to emerge in the near future, especially given that many of the key hardware components such as colour display, camera sensor and memory card are already present in the popular camera phones of today. It is estimated that by 2009, more than 31 million Americans will use video messaging generating 5.4B USD in revenue [43]. The key challenges facing the engineering community have been outlined in this chapter and it is clear that there are two dominant underpinning paradigms – efficient video compression and efficient hardware design. This work aims to help in this regard by proposing efficient multimedia co-processors (SA-DCT, SA-IDCT) and a set of algorithms that aid the design of a dominant general DSP operation – the constant matrix multiplication (of which the SA-DCT and SA-IDCT are particular cases).

## 2.1 Introduction

This chapter gives a technical overview of digital video processing and low power design to provide context for the research in subsequent chapters. The starting point for this thesis was to investigate power efficient hardware implementations for the SA-DCT/IDCT algorithms suitable for mobile devices (see Chapter 3 and Chapter 4). Research in this has led the author to realise that the SA-DCT/IDCT are instances of a more general task in DSP and video processing – the constant matrix multiplication (CMM). Since CMMs are a common task, and power optimisation is vital, this has led the author to research an electronic design automation (EDA) algorithm capable of optimising the hardware for a general CMM task (see Chapter 5). The algorithm can be leveraged when implementing any application that uses CMM tasks.

Digital video applications are primed to become mainstream on the emerging convergent mobile platform as discussed in Chapter 1. This chapter outlines why digital video requires complex compression algorithms and gives a detailed overview of the theoretical foundations of these algorithms and the associated industry standards (see Section 2.2). MPEG-4 object-based compression, which requires the computationally intensive SA-DCT and SA-IDCT algorithms, is discussed in Section 2.2.5. With the application and algorithmic foundations outlined, there then follows a survey of approaches to implementing such complex compression schemes on mobile embedded platforms which have extremely limited energy sources. As more features are integrated on the convergent device, the strain on its battery increases. This is problematic since consumers demand the conflicting goals of more features and increased battery life. Because power consumption properties are so important, a comprehensive survey of the sources of power consumption in CMOS circuitry is given along with the most popular techniques for estimating and tackling this issue (see Section 2.3).

## 2.2 Digital Video Processing

Prior to discussing how video data is processed, it is essential to understand how an electrical representation of a video sequence is formed [44]. At the most basic level an analogue image signal is generated

when a camera scans a two-dimensional scene and converts it into an electrical signal. A video signal consists of several successive image scans where each image is referred to as a frame. The scanning process actually generates three signals for each frame corresponding to the primary colours red ( $R$ ), green ( $G$ ) and blue ( $B$ ). To generate a digital image, each of the RGB signals are sampled and then quantised to a certain number of bits (usually eight) per sample. Each quantised sample is referred to as a pixel or pel and these are the fundamental unit of image processing and analysis. A frame is therefore a rectangular 2D array of pels where each pel location has three eight-bit values – one for each of the RGB colour components at that location.

RGB signals are usually transformed into another colour space called  $YC_bC_r$  where the  $Y$  signal represents the luminance (or brightness) component of the image whilst the  $C_b$  and  $C_r$  components represent the chrominance (or colour) components. This facilitates compatibility with black and white video but also helps compression since  $C_b$  and  $C_r$  are amenable to sub-sampling as discussed subsequently. The transformation equations are outlined in Equation 2.1

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.257 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} \quad (2.1)$$

$R'G'B'$  refers to the gamma-corrected version of the RGB colour space. Gamma-correction is a necessary step to ensure that an image is displayed accurately on a computer screen. All monitors scale input pixel voltages depending on their gamma function when displaying to a screen so the original RGB input voltages must be adjusted to  $R'G'B'$  to counteract the effect.

Most applications require standardised video formats, of which there are many. One example is the Common Intermediate Format (CIF), which has a luminance ( $Y$ ) frame resolution of  $352 \times 288$  pels with a frame rate of 30Hz<sup>1</sup>. The chrominance bandwidths are half that of the luminance since the human visual system is less sensitive to chrominance information [44]. CIF is an example of the 4:2:0 image format – here the chrominance components are sub-sampled by a ratio of two horizontally and by a ratio of two vertically as shown in Figure 2.1. A 4:2:0 frame is composed of three planes (one for each of  $Y$ ,  $C_b$  and  $C_r$ ) as illustrated in the figure, and these planes are usually passed to video processing tools in sequence when processing the frame. Each plane is sub-divided into non-overlapping 2D blocks of  $8 \times 8$  pels. For every four  $Y$  blocks there are two chrominance blocks – one for each of  $C_b$  and  $C_r$  (if the format is 4:2:0). Each group of corresponding six blocks is referred to as a macroblock as illustrated in Figure 2.1. In a single CIF frame there are 396 ( $22 \times 18$ ) macroblocks.

Chrominance sub-sampling is the first step taken in compressing the video data without any loss perceptible to the human eye. However, further compression of video data is necessary when you consider that the bit rate for uncompressed CIF is approximately 4.5MB/s (152064 bytes per frame  $\times$  30 frames per second)<sup>1</sup>. The remainder of this section summarises the techniques used to compress video into the most compact form possible for storage or transmission over a network to a compatible decoder<sup>2</sup>. Typically, compression ratios are traded off against decoded video quality and the acceptable quality depends on the application, but it is usually possible to compress video data to a very high degree with only minor

<sup>1</sup>The lower resolution Quarter Common Intermediate Format (QCIF) has a  $Y$  component resolution of  $176 \times 144$

<sup>2</sup>This thesis primarily concentrates on DCT based codecs – wavelet based compression codecs (as used in JPEG-2000) are only briefly described

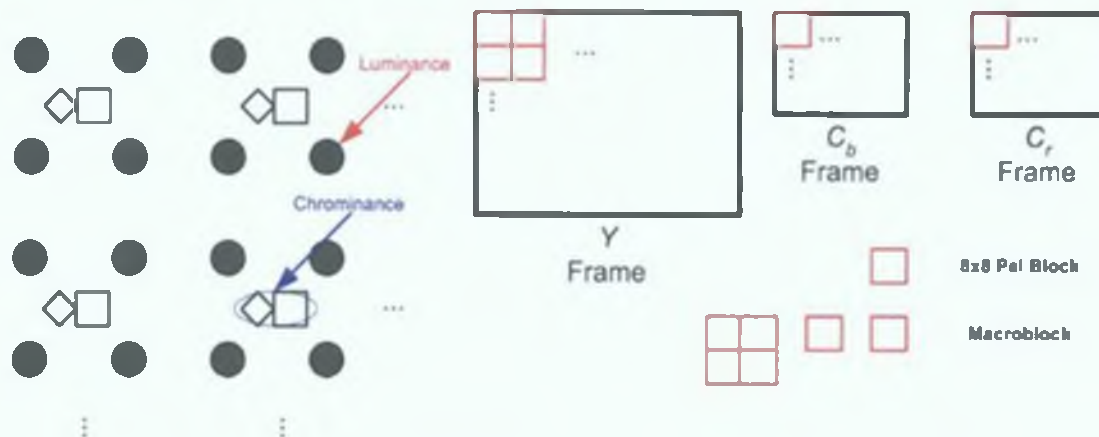


Figure 2.1: The 4:2:0 Image Format

quality degradation.

The achievement of high compression ratios for video data is made possible primarily because of the huge amount of redundancy present in such data. These redundancies may be broadly classified as follows:

- *Spatial Redundancy* – Taking a single frame in isolation, it is highly probable that neighbouring pixels are highly correlated.
- *Temporal Redundancy* – Most of the time it is likely that successive frames will be very similar in terms of their pixel content.
- *Perceptual Redundancy* – The human visual system is less sensitive to high frequency information [44].

The above are very intuitive properties but they can be exploited by very powerful compression techniques to remove as much redundant data as possible with minimal degradation of perceptual decoded quality for efficient transmission or storage. The amount of compression acceptable depends on the target application. In some cases (such as medical imaging) it is necessary that the decoded image be identical to the original image prior to encoding. This is called lossless coding, and hence redundancy can only be exploited to a limited extent. In lossy coding applications the compression can be more aggressive if defects in the decoded image are permitted. Again, the acceptable level of degradation depends on the target application and also the available bandwidth. In a low bit rate mobile environment it may be necessary to sacrifice some quality to ensure the compression rates are high enough to guarantee real time decoding.

## 2.2.1 A Generic Video Compression System

This section will briefly explain how a general video compression scheme works to illustrate how the redundancies outlined previously are exploited. The concept behind the system as a whole is explained and each of the tools involved are introduced. It is not the intention of this document to discuss each tool

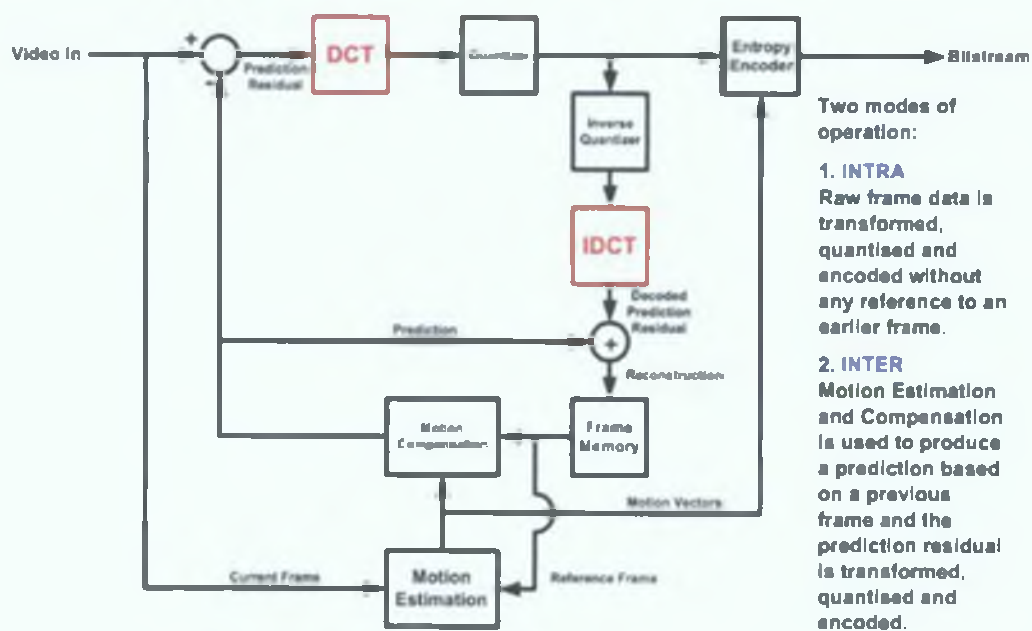


Figure 2.2: A Generic Video Compression Codec



Figure 2.3: Intra Mode Coding Flow

in depth as each are worthy research topics in their own right. The author's work focuses on the object-based derivative of the Discrete Cosine Transform (DCT) tool and this section gives a system context for the DCT. The DCT itself is given an in-depth treatment in Section 2.2.3.

A generic video coder/decoder (codec) is illustrated in Figure 2.2 [45]. A system like this is commonly referred to as a hybrid coder as it employs a combination of intraframe (transform-based compression) and interframe (differential pulse code modulation (DPCM) theory) techniques. The interframe processing stage exploits temporal redundancy and this is followed by intraframe coding, which exploits the spatial and perceptual redundancy of the result.

### 2.2.1.1 Intraframe Compression

Intraframe coding means that the frame is coded without any references to other frames. In relation to the codec in Figure 2.2, the feedback loop is not used in intra mode and the equivalent system is illustrated in Figure 2.3. If a frame is to be intra coded, it is fed to the above system in non-overlapping  $8 \times 8$  pel blocks. Each of the blocks for all planes undergoes transformation, quantisation and entropy encoding. Recall that each frame has a  $Y$ ,  $C_b$  and  $C_r$  plane and these planes are usually processed in turn per macroblock in a horizontal raster order.

**$8 \times 8$  Block Transform** The transform tool exploits spatial and perceptual redundancy in a frame. It maps spatial pel data to the frequency domain where each transform coefficient represents a spatial

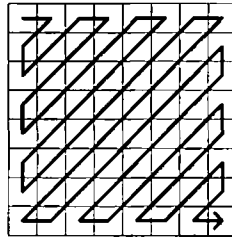


Figure 2.4 Zigzag Scan Order

frequency from DC up to the Nyquist limit. The reason for doing this stems from the fact that the image energy of most natural scenes is concentrated in the low frequency region and that humans are less sensitive to high frequency information. This suggests that operating in the frequency domain may make compression operations easier. Therefore the transform block converts data to the frequency domain to make subsequent quantisation and coding more efficient. This is achieved by decorrelating-correlating the signal energy (Section 2.2.2 explains this in much greater depth). The transform usually used for image and video coders is the Discrete Cosine Transform (DCT).

**Quantiser** It should be noted that the transform block does not achieve any compression in isolation – it merely converts the data to an alternate representation. Subsequent quantisation and variable length coding give the necessary bit rate reduction. Since the higher frequency coefficients are less important and have quite low values, they can be more coarsely quantised and may indeed be quantised to zero. Depending on the activity in the block, it may only be necessary to retain the DC value and a few low frequency coefficients as opposed to all 64 pel values without affecting the decoded frame quality. More detailed information on quantisers and quantiser design may be found in [44, 45, 46].

**Variable Length Encoder** To achieve further compression, the quantised transform coefficients undergo variable length coding (VLC). The idea behind VLC is that the length of the code assigned should vary inversely with the probability of occurrence of that code thus reducing the amount of bits required to represent the information [45]. The first step involves zigzag scanning (as shown in Figure 2.4) and run-length coding of the quantised transform coefficients (referred to as symbols in information theory parlance) into coding events. These coding events have lower entropy than the original symbols. The quantised coefficients are zigzag scanned in order of ascending frequency since there is a high probability that the high frequency coefficients will be long runs of zeros after quantisation. The next step is referred to as source coding, which involves replacing the coding events with binary codewords, the length of which depends on their probability of occurrence [44]. The most commonly used source coding schemes are Huffman Coding and Arithmetic Coding [44]. The VLCs generated for each block are then routed out to the bit-stream where the system level controller multiplexes them as appropriate.

### 2.2.1.2 Interframe Compression

To exploit the temporal redundancy in video data, the codec in Figure 2.2 operates in inter mode. A prediction of the current frame is formed (using a technique called motion estimation and compensation) and the prediction residual is then coded using the techniques outlined in the previous section. This is

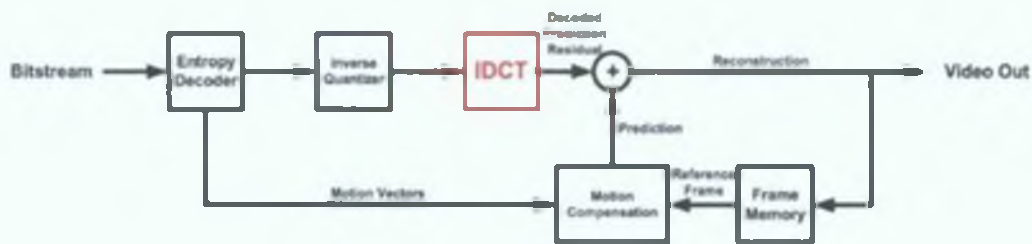


Figure 2.5: A Generic Video Decoder

done because the entropy of the prediction residual is much lower than that of the frame itself – this is especially true at high frame rates when successive frames are likely to be almost identical.

**Generating the Reconstructed Reference Frame – The Decoder** To create a prediction for a current frame based on one or more reference frames it is inappropriate to use the raw uncompressed data as a reference. The reference frames used must be identical to the frames that will be reconstructed in the decoder. This is because the prediction generated in the encoder must be identical to that generated in the decoder in order for the coded prediction residual that is transmitted to be appropriate in both systems. As a consequence, a decoder must also be present in the encoder – hence the name codec (coder/decoder). A generic video decoder is shown in Figure 2.5. To create a decoded prediction residual the data must undergo the opposite operations to the intra coding operations (i.e. Variable Length Decoding (VLD), inverse quantisation and an inverse transformation). This residual is added to the prediction (inter mode only) to generate the reconstructed frame. It is intuitive to see how the decoder in Figure 2.5 fits into the feedback loop in the codec in Figure 2.2.

**Motion Estimation and Compensation** Given a current frame and a reference frame<sup>3</sup> (a previously reconstructed frame) a prediction is formed for the current frame by using motion estimation and compensation. The motion estimation tool starts by loading the current frame into memory and processes the frame on a macroblock-by-macroblock basis. For each  $16 \times 16$  Y component of the macroblock in the current frame, the tool searches a certain region of the reference frame for the closest match. The search strategy used can vary and some examples include [47]:

- Full Search
- Logarithmic Search
- Hierarchical Search

This match is evaluated using a block-matching algorithm (BMA). Some of the more popular BMAs include:

- Cross Correlation Function (CCF)
- Mean-Squared Error (MSE)

<sup>3</sup>The reference frame is typically the previous frame in the video sequence, although some standards (e.g. MPEG-1, MPEG-2 & MPEG-4) allow predictions based on future frames.

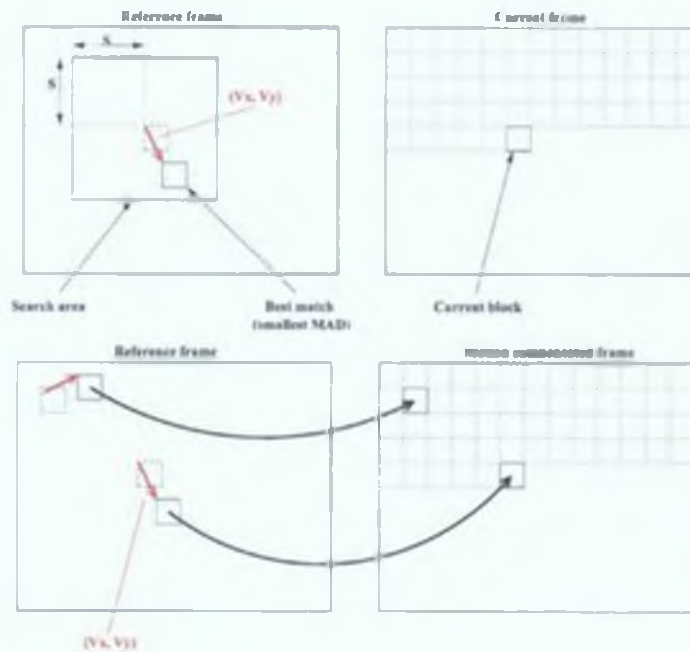


Figure 2.6: Motion Estimation and Compensation

- Mean Absolute Difference (MAD)
- Sum of Absolute Differences (SAD)

The motion estimation tool searches the reference data search window for the corresponding  $16 \times 16$  macroblock component with the best value according to the adopted BMA. In general this macroblock component will not be at the same coordinates as the current macroblock component so a motion vector (MV) is calculated to represent the offset. To generate the prediction frame (or motion compensated frame) for the current frame the MVs for each macroblock are used to translate the associated macroblock component with the best match in the reference frame to the location of the current macroblock component in the current frame. This process is more clearly illustrated in Figure 2.6 [44].

The motion compensated frame is the prediction of the current frame and this is subtracted from the current frame to create the prediction residual as shown in Figure 2.2. The prediction residual based upon motion estimation and compensation in general will have much lower entropy than a prediction residual formed by a simple frame difference. As a consequence, the temporal redundancy of the data is extensively exploited resulting in improved compression efficiency. To be clear – the outputs from the motion estimation and compensation tool are a predicted frame and a set of motion vectors (one for each macroblock). These MVs then undergo entropy coding and the VLCs produced are multiplexed appropriately into the output bit-stream. As shown in Figure 2.2, the prediction residual is encoded using the DCT-based intraframe processing steps. This fact re-enforces the importance of the DCT operation in a video compression system. Usually the BMA is evaluated for the Y component of each macroblock only and a scaled version of the resulting MV is assigned to the corresponding chrominance components when generating the chrominance motion compensated frame planes. Motion estimation and compensation is the most computationally demanding task in a video codec and a survey of power



efficient hardware implementations are discussed in [48]

### 2.2.1.3 Inter/Intra Mode Decision

The mode decision logic that decides whether the codec in Figure 2.2 operates in inter or intra mode depends on the specific standard adopted (a brief survey of the popular industry standards is given in Section 2.2.4). This decision may be made at the frame or macroblock level, although individual blocks can be coded in inter or intra mode. Usually an entire frame is intra coded periodically to create a new reference frame to guard against the current frame and the reference frame becoming too dissimilar (and the prediction residual becoming too large). Big differences may occur at a shot cut or at a scene cut so refreshing the reference frame is necessary. Periodic intra frames also provide “random” access points in the bit-stream since they can be decoded without reference to a previous or future frame.

## 2.2.2 Transform Theory Overview

Many image and video compression standards use transform-based compression techniques to achieve highly efficient compression ratios. The transformation process literally transforms the input data to an alternate representation, which is more amenable to compression than the original representation. By processing the data in the transform domain, it is easier to exploit the signal redundancy and achieve lower bit-rates for transmission. Both representations are equivalent and we can switch between the two seamlessly. This is possible because the transformation itself is a lossless process and performing the inverse transformation retrieves the original image data. It should be noted that it is a misnomer to talk about transform compression since the transformation process itself does not achieve any compression in isolation – hence the term *transform-based compression*. This section introduces in detail the mathematical principles behind transform-based compression with special emphasis on the Discrete Cosine Transform (DCT) and its Shape Adaptive extension (SA-DCT). A survey of the algorithms and architectures proposed in the literature for the DCT/SA-DCT are deferred until Chapter 3 where the proposed SA-DCT implementation is outlined.

### 2.2.2.1 Discrete Linear Orthonormal Transforms

**1D Transforms** Video data is generally arranged in sequential rectangular frames, which are arrays of pixels whose values are sampled values produced by an imaging device. Hence, any transform that operates on such discrete data must also be discrete. Discrete transforms find applications in many diverse areas of science, engineering and technology. In general, 1D discrete transforms transform an  $N$ -point data sequence into  $N$  transform coefficients. The transformation is done by comparing, using a dot product, the  $N$ -point data sequence with a set (of size  $N$ ) basis functions. To be clear, there are  $N$  basis functions – each with  $N$  samples/elements since the transform is discrete. A geometrical interpretation of the process is that the data is being transformed from one system of coordinates to another system of coordinates by a rotation and possibly a reflection. This second system is chosen so that the most of the coefficients (that define locations in this second system) have low values, and this fact can be used to aid compression.

Since most practical applications require encoding and decoding, a (most probably) linear forward  $\mathbf{A}$  and inverse  $\mathbf{A}^{-1}$  transform pair are necessary. It is usually convenient to represent such a forward and

inverse transform pair in terms of linear matrix multiplications as in Equation 2.2 [44]

$$X = \mathbf{A} \times x \Leftrightarrow x = \mathbf{A}^{-1} \times X \quad (2.2)$$

Here the vector  $x$  contains the original (spatial) representation of the data, and the vector  $X$  contains the transformed representation. The 2D matrix  $\mathbf{A}$  contains the set of basis functions, and the choice of  $\mathbf{A}$  corresponds to the choice of transform, which is application dependent. Equation 2.2 is a 1D transform pair – vectors  $x$  and  $X$  are 1D vectors (of size  $N$ ) and  $\mathbf{A}$  and  $\mathbf{A}^{-1}$  are 2D  $N \times N$  matrices that contain  $N$  1D basis vectors each. The transformation is an orthogonal transform if the inverse of the basis matrix  $\mathbf{A}$  is its transposition as shown in Equation 2.3

$$X = \mathbf{A}^{-1} = \mathbf{A}^T \quad (2.3)$$

This means that the implementation of the inverse transform is much the same as implementing the forward transform since it uses the same basis matrix albeit transposed. This aids efficient software/hardware design and permits the integration of a single forward/inverse transform module with mode selection logic. The set of basis functions discussed earlier form an orthogonal set in an  $N$ -dimensional space (for an  $N$ -point transform). Geometrically, this means that each basis function may be considered as an  $N$ -dimensional vector – each pointing in a mutually exclusive direction in an  $N$ -dimensional space. In fact, a more general form of orthogonality is unitarity. A unitary matrix has the property that its inverse is equal to the transpose of its complex conjugate (Equation 2.4)

$$X = \mathbf{A}^{-1} = \mathbf{A}^{*T} \quad (2.4)$$

In this case, an orthogonal matrix is also a unitary matrix. It is useful to keep in mind the distinction between orthogonality and unitarity if a complex transform is being considered. However, the discussion in this thesis restricts itself to dealing with basis functions that are purely real.

The set of  $N$  orthogonal basis functions (each with  $N$  samples) described by  $\mathbf{A}$  are in a sense complete in that they capture every way that the data in vector  $x$  can change – i.e. capture all the spatial frequencies. This means that  $N$  transform coefficients are enough to represent  $N$  data samples exactly. The proof of this concept stems from Fourier's theorem but may be understood intuitively with the ideas of sampling theory [49]. In the discrete sampled domain we know that samples are meaningful only if the frequencies captured are less than or equal to the Nyquist frequency of the input analogue waveform (i.e.  $f_s/2$  – half the sampling frequency). Therefore, the highest frequency we need consider is that represented by two adjacent samples. Even if higher frequencies existed before sampling they are now aliased down below  $f_s/2$ . So for an  $N$  point vector only  $N$  transform coefficients are needed for each basis function from frequency  $f_s/N$  up to the Nyquist frequency to capture the information completely.

An orthogonal transform becomes an orthonormal transform if each of the orthogonal basis vectors is also a unit vector in its corresponding direction. An orthonormal transform is useful since the energy of the data is preserved in the transform domain (the sum of the variances of  $x$  is equal to the sum of the variances of  $X$ ). This equivalence can be proved by demonstrating that the squares of the vector norms

are equal [50]:

$$\|X\|^2 \equiv \sum_{n=0}^{N-1} |X(n)|^2 \equiv \|x\|^2 \quad (2.5)$$

This by definition allows perfect reconstruction of the original data when the inverse transform is applied.

Although an orthonormal transform preserves signal energy in both domains, the weighting of this energy is distributed very differently when one compares the spatial and the transform domains. Orthonormal transforms have the ability to concentrate a large fraction of the average energy of the data into a few transform coefficients. It is this idea that is exploited in transform-based compression systems for image and video data. Some transforms do a better job than others depending on the statistical properties of the data being transformed.

In the field of image and video compression, the choice of transform is based on the fact that neighbouring pixels in a frame are likely to be very similar, i.e. highly correlated. An image is often modelled as a first-order Markov process for which the correlation between pixels in the image is proportional to their geometric separation. If this is the case, there is a certain amount of spatial redundancy present in the data. The transform process exploits this redundancy and subsequent clever quantisation and coding stages achieve a high compression ratio, which leads to the average bit rate being reduced significantly.

To summarise, transform-based compression using discrete orthonormal transforms is very efficient since:

- Most of the signal energy is decorrelated-correlated and compacted into relatively few coefficients, so not all coefficients need to be transmitted to maintain quality.
- The coefficients that are sent can be quantised to various degrees of accuracy since "not all transform coefficients are created equal" (The human visual system is less sensitive to high frequency data so the higher frequency coefficients can be coarsely quantised).
- Coefficients have a very non-uniform probability distribution, so are potentially suitable for entropy coding.

**Extension to 2D** For video processing, a 2D transform is generally used since uncompressed video is arranged into sequential rectangular 2D frames. It is possible to rearrange a 2D frame into a long 1D vector and use a 1D transform but this method does not exploit correlation in both horizontal and vertical directions. A 2D transform of a frame tends to be an iterative process of transforming successive non-overlapping  $N \times N$  blocks within that frame where  $N$  divides evenly into the frame resolution. So reconsidering Equation 2.2 the vectors  $x$  and  $X$  are now 2D  $N \times N$  blocks  $y$  and  $Y$ . Therefore the set of basis functions  $A$  becomes a set of basis functions/images  $B$ , i.e.  $B$  is a block matrix where each element of  $B$  is a matrix itself. A 2D transform pair is shown in Equation 2.6.

$$Y = B \times y \Leftrightarrow y = B^{-1} \times Y \quad (2.6)$$

There are now  $N \times N$  basis images each with  $N \times N$  elements since the transform is still discrete. To visualise this, consider that the basis images represent  $N \times N$  different arrays of pixel values. Given a set of  $N \times N$  transform coefficients, the inverse transformation operation consists of multiplying each basis image by its corresponding coefficient and summing the  $N \times N$  sets of resultant pixel values. The pixel

values of the basis images, and the different distribution of these values, mean that with an appropriate set of coefficients we can regenerate any pattern of pixels. The forward transformation process may be thought of as finding the set of  $N \times N$  coefficients that, when multiplied by the basis images, yield the original  $N \times N$  pixel block. Implementing a 2D transform directly is generally inefficient and most implementations exploit a useful property of  $\mathbf{B}$ . If  $\mathbf{B}$  can be factorised into two 1D kernels as shown in Equation 2.7a then the orthonormal transform is said to be separable and the transformation equations may be re-written as in Equation 2.7b [51].

$$\mathbf{B} = \mathbf{A}^T \mathbf{A} \quad (2.7a)$$

$$Y = \mathbf{A} \times (y \times \mathbf{A}^T) \Leftrightarrow y = \mathbf{A}^T \times (Y \times \mathbf{A}) \quad (2.7b)$$

Equation 2.7b implies that a 2D orthonormal transform can be implemented by an  $N$ -point 1D transform on each of the  $N$  rows of  $y$  and subsequently performing an  $N$ -point 1D transform on each of the  $N$  columns of the result. This fact is exploited when implementing fast algorithms and efficient hardware architectures for the transform. This separable property means that all the techniques for implementing a 1D transform can be applied to the 2D case, and indeed for any higher order dimension.

Two issues arise from this discussion; an appropriate choice of  $N$  and whether or not the extension of the transforms to higher dimensions is necessary. Considering the former, in general  $N = 8$  or  $16$  is used in image coding. Although there may be correlation between neighbouring  $N \times N$  blocks, it is too insignificant to warrant a larger value of  $N$ . Small block sizes also allow the use of sophisticated adaptive transforms that can be tailored to local statistics. This also means that the subsequent quantisation error spreads to a smaller area and is therefore easier to control. Indeed, the most recent video conferencing standard (H.264 as introduced in Section 2.2.4) advocates the use of flexible block sizes of  $N = 2, 4, 8, 16$  depending on the context. From an implementation point of view, hardware complexity in terms of memory size and logic is reduced considerably with small  $N$  compared with a 2D transform of an entire frame. For high-throughput applications, parallel processing is possible since each of the  $N \times N$  blocks within a frame can be transformed independently. The predictor generated by the motion estimator also improves with a smaller block size, and this is why the latest H.264 video compression standard is based primarily on a  $4 \times 4$  transform [52].

Secondly, it may seem intuitive to extend the transform to 3D to decorrelate between corresponding blocks in successive frames. The reason why this is generally not done in a typical video codec is that the motion estimation and compensation process can decorrelate this data more easily than a 3D transform could so in most video codecs an  $8 \times 8$  2D transform is used. The implications for implementation complexities are another reason why 3D transforms are not commonly used.

### 2.2.3 The Discrete Cosine Transform (DCT)

Good transforms compact the largest amount of energy into the smallest number of coefficients. However, not all pictures have the same statistical characteristics. Therefore the optimum transform is not constant, but depends on the block-by-block video content. It is possible to recalculate the optimum transform matrix for each new data block being transformed, and this is exactly what the Karhunen-Loeve Transform (KLT) does [53]. The basis functions for a KLT are the eigenvectors of the cross-covariance

matrix for the particular data block being transformed. Therefore the KLT completely decorrelates the data. Statistically it packs the most variance into the fewest number of transform coefficients thus minimising the mean square error for subsequent compression. Although the KLT is statistically optimal, it is too complex to implement in a practical codec. This is because the transform matrix must be calculated and then indicated to the decoder for each block, which is a significant overhead for any compression system – especially in a low-bandwidth mobile environment.

The Discrete Cosine Transform (DCT) offers a compromise, and is actually quite close statistically to the optimal KLT but without the added complexity. There are numerous definitions for the DCT, and the most popular is the so-called DCT-II that was proposed by Ahmed, Natarajan and Rao in 1974 [54]. Equation 2.8a defines the 1D DCT-II for an  $N$ -point vector, where  $k = \{0, 1, \dots, N-1\}$  and  $F(k)$  denotes the  $k^{\text{th}}$  DCT coefficient. The vector  $f(x)$  represents the data vector with  $C(0) = 1/\sqrt{2}$ , and  $C(k) = 1$  otherwise. Equation 2.8b defines the corresponding inverse transform (1D IDCT-II).

$$F(k) = \sqrt{\frac{2}{N}} C(k) \sum_{x=0}^{N-1} f(x) \cos \left[ \frac{(2x+1)k\pi}{2N} \right] \quad (2.8a)$$

$$f(x) = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} C(k) F(k) \cos \left[ \frac{(2x+1)k\pi}{2N} \right] \quad (2.8b)$$

The shape of the basis functions for the DCT-II are illustrated in Figure 2.7 for  $N = 8$ . It can be seen that the first cosine is used to evaluate the average or DC value of the  $N$ -point sequence. The subsequent cosines have frequencies that are half integer multiples of  $f_s/N$  up to the Nyquist frequency. The reasons why these basis functions are powerful are quite subtle and can be explained using Fourier's Theorem [44]. This theorem is defined for periodic and theoretically infinite signals, but image data is not infinite or in general periodic. Therefore to use Fourier theorem on image data we need to take successive windowed blocks of this non-periodic data, and an artificially periodic waveform must be created for each block using the data in that particular block. How to create that periodicity is a design decision – but it turns out that using a windowed data block to create half-range even periodicity gives good energy compaction and does not introduce any undesirable jump discontinuities (which lead to Gibbs phenomenon). Again from Fourier theory, this leads to the need for cosine basis functions that have frequencies that are half integer multiples of  $f_s/N$  up to the Nyquist frequency, i.e. the DCT basis functions. This is why the DCT is popular – the relatively good energy compaction and the data independent basis functions make the design of a video codec much easier.

The DCT-II and IDCT-II may be generalised to 2D, as illustrated by Equations 2.9a and 2.9b. In this case  $k, l = \{0, 1, \dots, N-1\}$  and  $F(k, l)$  denotes the DCT coefficient at coordinate  $(k, l)$ . Data  $f(x, y)$  is the input data at coordinate  $(x, y)$  with  $C(0) = 1/\sqrt{2}$ , and  $C(k) = C(l) = 1$  otherwise.

$$F(k, l) = \frac{2}{\sqrt{N^2}} C(k) C(l) \sum_{y=0}^{N-1} \sum_{x=0}^{N-1} f(x, y) \cos \left[ \frac{(2x+1)k\pi}{2N} \right] \cos \left[ \frac{(2y+1)l\pi}{2N} \right] \quad (2.9a)$$

$$f(x, y) = \frac{2}{\sqrt{N^2}} \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} C(k) C(l) F(k, l) \cos \left[ \frac{(2x+1)k\pi}{2N} \right] \cos \left[ \frac{(2y+1)l\pi}{2N} \right] \quad (2.9b)$$

The basis images for equation 2.9a with  $N = 8$  are illustrated in Figure 2.8. It is intuitive to see that each

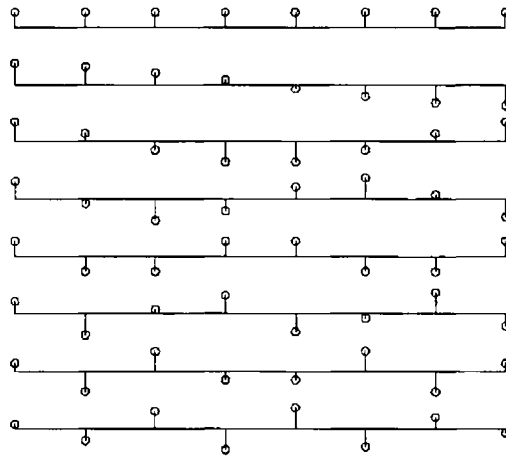


Figure 2.7 DCT-II Basis Functions for  $N = 8$

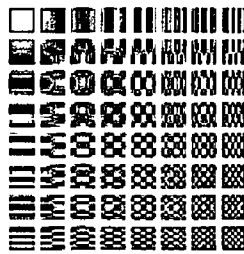


Figure 2.8 Basis Images for 2D DCT-II with  $N = 8$

basis image captures different frequency information from DC (top left corner image) to the Nyquist limit (bottom right corner image). Each DCT coefficient  $F(k, l)$  produced by the  $N \times N$  2D DCT is a dot product of all the  $N \times N$  input samples  $f(x, y)$  with a corresponding cosine-based weighting image that depends on the coefficient's position in the output array and the position of the sample in the input array.

### 2.2.3.1 The Discrete Wavelet Transform (DWT) - An Alternative?

The choice of the DCT is not inevitable, and there are many alternatives. A lot of research has been carried out into efficient implementations of the Discrete Wavelet Transform (DWT) for example, and it has been incorporated into the JPEG-2000 and MPEG-4 (for static textures) standards. One reason for the adoption of the DWT is that it is inherently scalable. The DWT process can be repeated for as many iterations as desired or required. The decoder may stop at any time by using a subset of the bit-stream depending on its capabilities if full resolution is not required. Wavelet compression is claimed to be more efficient at low bit rates. At low bit rates, the DCT introduces block artefacts, which are precisely the artefacts to which the human psycho-visual system is most sensitive. On the other hand, excessive quantisation of DWT coefficients leads to smearing of detail, which is more difficult for humans to see.

A comparative study of the DCT and DWT coding schemes is given in [55] by Xiong et al., and concludes that for still images the DWT outperforms the DCT by the order of about 1dB PSNR. It also concludes that the advantage of the DWT is less obvious for video coding. Xiong et al. also claim that

a hardware or software implementation of the DCT is less expensive than that of the DWT. So although the DWT gives better performance at low bit-rates, the additional computational complexity makes the DCT seem a better alternative to the DWT from a low power hardware accelerator design point of view.

Even with the emergence of the DWT, the DCT will hold favour for a long time, as there has been a lot of research effort invested in it and it is employed by many industrial standards. The advantages of using the DCT for image and video compression are clear and are based on the properties exhibited by the DCT [46]:

- Has all the advantages of any discrete orthonormal transform as described previously.
- Close to statistically optimal KLT (in terms of MSE and energy compaction) compared to other alternatives for image and video data and is much simpler to implement since the basis functions are independent of the image data.
- Real arithmetic is sufficient for a cosine series.
- It is a separable transform, so can be easily extended to higher dimensions.
- Has many fast algorithms.
- Has many VLSI implementations suitable for various constrained environments (see the survey in Section 3.2).
- Has been adopted by most standards including the MPEG series and JPEG.

## 2.2.4 Image and Video Compression Standards Overview

This section aims to give a brief overview of the evolution of the most popular digital image and video standards including how they differ from each other. More details on these standards can be found in the cited references. In general, the standards outlined here all have some flavour of the generic codec engine detailed in Section 2.2.1. Broadly, the video standards have evolved under the two brand names of H.26x and MPEG-x [45]. The H.26x codecs are recommended by the International Telecommunication Union (ITU-T), and are targeted to telecommunication applications such as video conferencing and video telephony. The MPEG-x codecs are recommended by the Motion Picture Experts Group (of the International Standardisation Organisation and the International Electrotechnical Commission, Joint Technical Committee number 1 – ISO/IEC JTC 1), and are designed to address the needs of video storage, broadcast TV and video streaming. A flowchart illustrating when the main digital video standards were released is given in Figure 2.9. The still image standards released by the Joint Picture Experts Group (JPEG) are also included for completeness.

### 2.2.4.1 Still Picture Coding – JPEG and JPEG-2000

In the mid 1980s, a collaboration between the ITU-T and ISO led to the image compression standard known as JPEG: the Joint Photographic Experts Group [45]. The JPEG compression scheme is essentially an intra-mode only video codec as shown in Figure 2.2 since a still image has no reference frames. A JPEG standard specifies two classes of operation, namely lossy or lossless<sup>4</sup> compression. The

---

<sup>4</sup>The lossless compression class is based on a predictive coding algorithm and is not transform based.

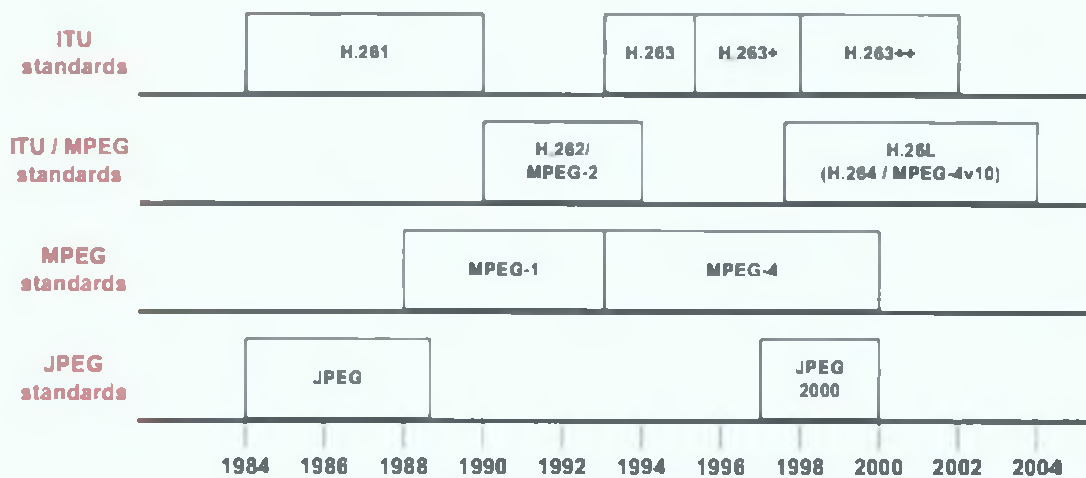


Figure 2.9: Evolution of Video Coding Standards by the ITU-T and ISO/IEC [45]

JPEG-2000 standard has subsequently been developed to cater for the wider spectrum of modern multimedia applications. It is similar to JPEG, but has greater flexibility with more sophisticated pre and post processing. A key difference between JPEG and JPEG-2000 is that JPEG-2000 uses the DWT instead of the DCT. JPEG-2000 has superior low bit rate performance, and includes features like image security watermarking and region of interest coding [19, 45].

#### 2.2.4.2 H.261

The demand for low bit rate real time video telephony and video conferencing provided the impetus for the development and release of the H.261 video standard in the late 1980s. The bandwidth targeted by H.261 was baseline ISDN (Integrated Services Digital Network) of 364 kb/s but this was later extended to systems based on multiples (1 to 30) of 64 kb/s. The H.261 standard adopted the hybrid codec architecture and was a milestone for real time low bit rate coding of video of reasonable quality. An attractive feature of the H.261 standard (as with all digital video standards it must be said) is that it is defined in terms of the bit-stream and decoder sequencing. This leaves scope for designers to incorporate new innovations on the encoder side to meet specific needs while maintaining compliance with the standard decoder. For example, the standard specifies that the use of motion estimation to generate the predictor is normative, but the algorithm used to implement the motion estimation is non-normative. Hence different implementations of the standard can use distinct motion estimation search strategies etc., but all are compliant once they generate a normative bit-stream.

#### 2.2.4.3 MPEG-1

In the early 1990s the MPEG body started to investigate coding techniques for storage of video on CD-ROMs at a bit-rate of about 1.5 Mb/s (the maximum rate that could be handled by CD-ROM drives at the time). CD-ROMs were targeted since at the time they were the only portable medium that could support the storage requirements for digital video. H.261 was used as the starting point in the development of this new standard (referred to as MPEG-1), although MPEG-1 and H.261 do not have compatible bitstreams. Since MPEG-1 was targeted to storage applications (as opposed to real time video conferencing)



ing), compression efficiency could be improved at the expense of latency (e.g. by allowing bi-directional prediction). Another goal of the MPEG-1 standard was to be a viable alternative to analogue VHS storage. As such, MPEG-1 was designed to offer the same features as VHS (fast forward/reverse, freeze, random access, ...) although the degree of access and manipulation is inversely proportional to the compression efficiency (since only intra coded frames can be manipulated for these features)

#### 2.2.4.4 MPEG-2

Following the success of H.261 and MPEG-1 there was a need for a video codec to support a wider variety of application areas [45], and ISO/IEC and the ITU-T collaborated to release MPEG-2/H.262 (more commonly referred to as MPEG-2) in 1995. The target bit rate for MPEG-2 is approximately 4 – 15 Mb/s and the standard is used in areas such as digital TV (satellite and cable) and DVD. The primary differences between MPEG-1 and MPEG-2 are the picture resolutions supported and scalability. MPEG-2 can support a wider range of picture resolutions and frame rates from Source Input Format (SIF) to High Definition Television (HDTV)<sup>5</sup>, and also supports coding of interlaced video sources. The scalability feature was introduced to increase flexibility and to accommodate the varying capabilities of different receivers on the same service. Different clients can decode a subset of the same bitstream at various qualities, temporal or spatial resolutions. Alternatively in a networking application the less essential parts of the bitstream may be discarded based on network congestion. HDTV resolution coding was seen as the next target (and work started on MPEG-3) but it was found that MPEG-2 was versatile enough to support HDTV and MPEG-3 was abandoned. MPEG-2 has also proven to be a success and it is foreseen that in the USA by 2014 the existing transmission of analogue National Television Standards Committee (NTSC) video will cease and HDTV MPEG-2 will be the only terrestrial broadcasting format [45].

#### 2.2.4.5 Very Low Bit Rate Coding – MPEG-4 and H.263

Despite the successes of the earlier MPEG standards, there was still a need to accommodate the coding of video at very low bit rates (under 64 kb/s). The emergence of the wireless industry has been a major driver towards this goal since the channel capacities of mobile networks are very limited, hence a new MPEG standard called MPEG-4 was formulated to meet these demands. However, MPEG-4 was also used as a vehicle for meeting other requirements such as coding of multi-viewpoint scenes, graphics and synthetic scenes (as well as improved compression efficiency for conventional natural scene coding). MPEG-4 has evolved into a huge standard with something for all applications, and this has caused criticism of the standard as well as praise. As such, MPEG-4 has been optimised to operate at a range of bit rates. It was explicitly optimised for three ranges (below 64kb/s, 64 – 384 kb/s and 384 kb/s – 4 Mb/s), but can also operate at higher bit rates for studio and broadcast applications [49]. Perhaps the most significant feature of MPEG-4 is that it has adopted a scene representation in terms of the individual audio-visual objects that make up a scene and each object is encoded independently. Object-based coding is discussed in Section 2.2.5 in more depth, but its main benefit is that it helps to make best use of the bit rate and opens up many new application areas.

---

<sup>5</sup>SIF has a resolution of 384 × 288. HDTV has a resolution from 768 × 576 up to 1920 × 1280.

Work has also been undertaken by the ITU-T to meet the needs of very low bit rate coding and the resultant standard is called H 263. This standard is built on H 261 although it uses more sophisticated methods to code the transform coefficients and the motion vectors [44, 45]. Some subsequent improvements to H 263 have been proposed (H 263+/H 263++) that enhance the compression efficiency even further and these have become annexes to the standard. However, H 263 does not support object-based coding.

#### **2.2.4.6 Advanced Video Coding (AVC) – H 264 / MPEG-4 Part 10**

After the successful collaboration resulting in H 262/MPEG-2, the ITU-T and MPEG decided to work together again on very low bit rate video. They founded a project called H 26L (where L stands for long term objectives), whose objective is to create a single video standard to outperform the more optimised H 263 and MPEG-4 codecs. The official title of the new standard is Advanced Video Coding (AVC), but it is known by the ITU-T group as H 264 and by MPEG as MPEG-4 Part 10.

AVC represents a significant step forward and it has developed the video coding tools in synchronisation with the VLSI technology that is available today or will be available in the near future. Some features that were too expensive with the state of the art technologies when earlier standards were being developed are included in the AVC standard. It is claimed that AVC improves over MPEG-2 in terms of coding efficiency by a factor of two while keeping the cost within an acceptable range [52]. Some of the new features include enhanced motion-prediction capability, use of a small block-size integer arithmetic transform, adaptive in-loop de-blocking filter and enhanced entropy coding methods [52]. Smaller block-size transforms reduce blocking artefacts and facilitates better prediction. Since H 264 makes extensive use of prediction it is very sensitive to prediction “drift”. This drift exists due to arithmetic round-off in the DCT/IDCT. The use of integer arithmetic lessens the computational burden of the transform and eliminates this drift [56]. As well as improved compression efficiency, AVC offers greater flexibility for effective use over a wide variety of network types and application areas. However, H 264 is not capable of object-based compression.

#### **2.2.5 Object Based Processing**

When introducing the MPEG-4 standard in Section 2.2.4 it was mentioned that it uses a radically different approach to coding a video scene. MPEG-4 considers a scene as being composed of distinct audio-visual objects and each is coded independently. In this context, an object may be defined as “something in the scene with which the user or system would like to interact”. This is revolutionary in many respects (e.g. user interaction, scalability, error robustness etc.), but has significant coding implications. Earlier standards encoded a frame as a single rectangular object. In MPEG-4, a video frame may contain different natural or synthetic items and these are classed as objects and may be coded independently. Video objects can be any arbitrary shape in general. MPEG-4 is also backward compatible in that it can code the entire frame as the sole object (these are referred to as the simple profiles).

The advantages of object-based coding are manifold and include [17, 19, 44, 57]

- Encoding of objects at different qualities and spatial/temporal resolutions according to their importance

- Content-based interaction with objects giving web-like manipulation of a scene (e.g. an interactive shopping or teaching experience)
- Object re-use and virtual scene creation (e.g. insertion or deletion of advertisements)

Coding each object independently facilitates many new applications as well as improving the compression efficiency. Of course this process assumes that the arbitrarily shaped objects have been defined and segmented from the frame prior to encoding. The segmentation process produces an alpha mask, which defines which pixels in the frame belong to the segmented object. This is a non-trivial issue and a significant research challenge in itself.

### 2.2.5.1 Video Object Segmentation

The segmentation process involves the identification of semantically meaningful video objects from a frame. The difficulty with this is that what constitutes an object depends on the application. For example, in a video telephony application the human face could be the object of interest. However, in a biometric identification application the eyes could be the object of interest (e.g. for retinal scan identification). Hence video object segmentation is an ill-posed problem in general. However, as briefly summarised here, there are some generic approaches based on extracting and labelling image regions of similar properties such as brightness, colour, texture, motion and others [45]. The segmentation process can be performed either automatically or semi-automatically (i.e. guided by some user interaction). It must be emphasised that the applicability of these general approaches depends heavily on the application.

A comprehensive survey of approaches to video object segmentation is given by Zhang and Lu in [58]. The broadest classification groups algorithms into motion-based approaches and spatio-temporal approaches. Motion-based segmentation approaches generally have three main properties: the data primitives used (pixels, corners, regions, ...), the motion model used (2D, 3D, parameter estimation, motion estimation, ...) and the segmentation criteria used (maximum a posteriori, Hough Transform, expectation and maximisation, ...). Traditionally motion-based segmentation methods are usually only applicable to scenes with rigid motion or piecewise rigid motion. Spatio-temporal approaches leverage spatial features (edges, colour, morphological filtering, ...) and temporal (motion) features to give a more robust solution. These approaches are applicable to non-rigid motion and hence more generic scenes. Since it is impossible to generalise for all applications, effective object segmentation algorithms will always be an open research problem. It is a very active area of research and some state of the art algorithms are presented in [59].

Object segmentation is computationally demanding in general, and as a result there has been some research work published that aims to hardware accelerate the task. For example, Chien et al. propose a systolic array scheme based on frame differencing and morphological filtering (dilation, erosion and conditional/geodesic erosion) [60]. Systolic arrays are arrays of processing elements that are connected in a mesh-like topology to a small number of nearest neighbours, and are generally used for tasks that are highly computationally intensive. Processors perform a sequence of operations on the data that flows between them. Another systolic array approach by Kim et al. extracts, filters and normalises spatial and temporal low-level features [61, 62]. These features are combined by a neural network to generate a segmentation mask.



Figure 2.10: Mobile Telephony Application – Human Face is Object of Interest

A likely application on a mobile multimedia terminal with video processing capability is video telephony. In this scenario, an object-based compression scheme is advantageous since it allows the semantic video object of interest (e.g. the human face) be coded at higher quality and the background can be coded at a much lower quality since it has less semantic significance (see Figure 2.10). Indeed, if the background is static it need only be transmitted once to the receiver. Such a strategy is contingent on accurate real-time face segmentation. Face detection is difficult in general due to complications like pose, occlusion, facial expression and orientation. Hence, it is another huge research topic in its own right, and a survey of approaches is presented in [63]. However, in a constrained application, real-time face detection is possible using some of the simpler face detection schemes (e.g. colour filtering). For example, it is not unreasonable to assume that in a video conference application, the face is probably in the centre of the frame looking directly at the camera, with not much rotation or occlusion. Despite the wider range of applications possible, MPEG-4 object-based compression has its detractors due to the difficulty of the segmentation problem in general. However, it is the belief of the author that in a constrained application such as mobile video telephony, valid assumptions simplify the segmentation problem. Hence certain object-based compression applications and associated benefits become possible. A screen-shot of a face detection algorithm using simple *RGB* thresholding [64] is shown in Figure 2.11. Although the segmentation mask quite rough, the computational complexity of the algorithm is very simple and the quality could be improved by aggregating a number of simple features based on some assumptions made possible by the known constraints.

### 2.2.5.2 An Object-based Video Compression System

Before discussing how the video codec in Figure 2.2 needs to be developed to facilitate object-based coding, some new terminology needs to be defined. In MPEG-4 video, objects are referred to as Video Objects (VOs) and these are irregular shapes in general but may indeed represent the entire rectangular frame as mentioned earlier. A VO will evolve temporally at a certain frame rate and a snapshot of the state of a particular VO at a particular time instant is termed a Video Object Plane (VOP). The alpha mask defines the shape of the VOP at that instant and this mask also evolves over time. Figure 2.12 shows 3 VOPs to illustrate how a VO and its corresponding alpha mask evolve over time [44].

VOPs are passed to a shape-based codec (Figure 2.13) in the same way frames are passed to a frame-based codec (Figure 2.2). In this case the shape information (the alpha plane information) as well as the texture information needs to be coded and transmitted to the decoder. As such, a shape encoder and



Figure 2.11: Example Face Detection Based on Colour Filtering



Figure 2.12: Three Successive VOPs with Alpha Masks of a VO

decoder are needed in the object-based codec as well as a shape-adaptive transform block that will only transform pels that are part of the VOP.

**Shape Coder / Decoder** The input to the shape coder is the alpha frame defining the current VOP to be coded. The alpha frame is subdivided into non-overlapping 2D  $16 \times 16$  blocks (corresponding to the Y texture macroblock components discussed earlier), and these are referred to as Binary Alpha Blocks (BABs) for binary shape coding. Each element of the BAB defines whether the corresponding texture pixel is part of the current VOP. If the BAB element value is 255 the corresponding texture pel is part of the VOP and if it is 0, the pel is not included<sup>6</sup>. There are three BAB types – either the BAB is completely inside the VOP, completely outside the VOP or on the VOP boundary as illustrated in Figure 2.14.

Each BAB is encoded using motion compensation and a technique referred to as Context-based

<sup>6</sup>Alpha blocks are not restricted to binary—they can also be grey level indicating the transparency of a pixel inside a VO.

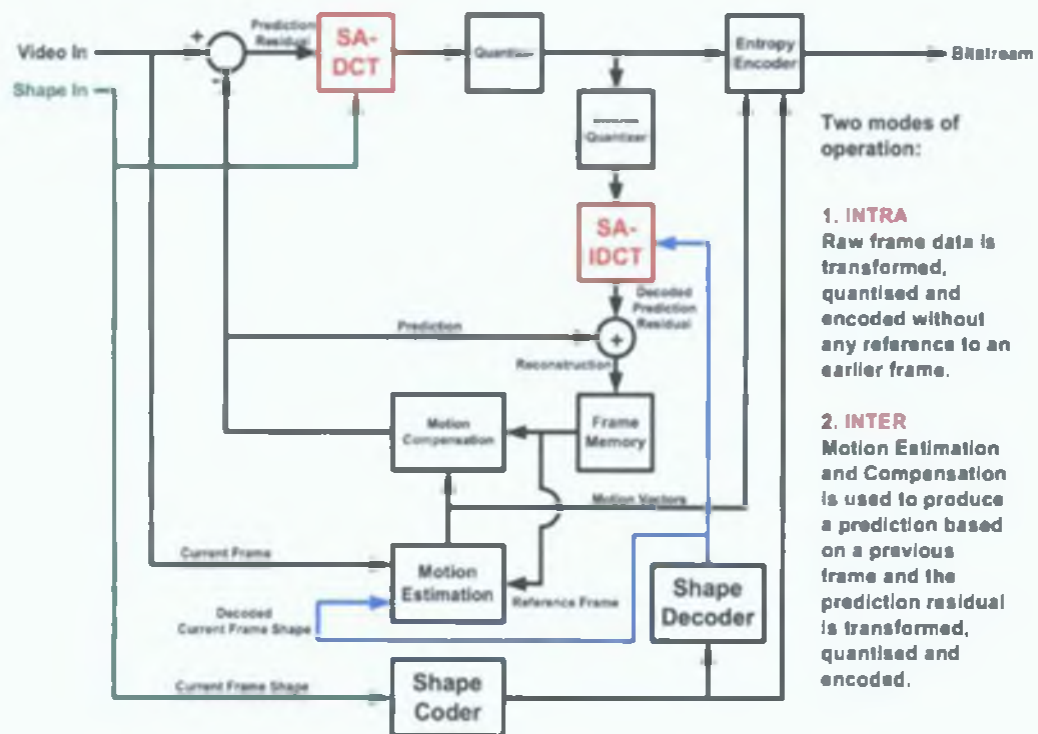


Figure 2.13: A Generic Object-Based Video Compression Codec

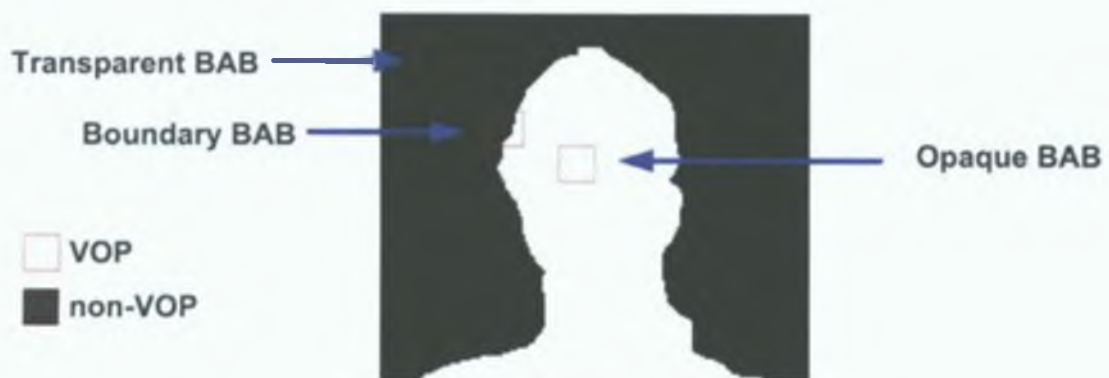


Figure 2.14: Various BAB Types

Arithmetic Encoding (CAE) [65] Motion compensation produces a motion vector for each BAB encoded using CAE The motion vector for each BAB is passed to the entropy encoder and encoded using a prediction based on neighbouring shape or texture motion vectors The idea behind CAE itself is to use efficient arithmetic coding to represent the state (transparent/opaque) of each pixel in the BAB Figure 2 13 shows the shape coder separated from the entropy encoder even though CAE is a form of entropy coding itself Conceptually, the shape coder block produces motion vector differences for shape that are passed to the entropy coder as well as the encoded shape (the result of the CAE process) Both components are multiplexed into the bitstream by the system level controller A shape decoder is also included in Figure 2 13 since the inverse motion compensation tool and the inverse transform tool in the downstream decoder will only have the decoded shape information and not the raw input shape information Since shape coding may be a lossy process depending on the mode of operation, it is important to maintain compatibility in the decode feedback loop of the codec Shape coding is a complex task and a survey of hardware implementations is discussed in [48]

**Shape Adaptive Transform** As well as coding the shape definition of the VOP using the shape coder, the VOP texture must also be encoded A frame-based codec uses an  $8 \times 8$  transform block to do this in intra mode (see Section 2 2 1) In a shape-based codec a shape-adaptive transform is required to exploit the spatial and perceptual redundancy of the VOP, ignoring any texture not encompassed by the VOP The tool used by MPEG-4 is the Shape Adaptive Discrete Cosine Transform (SA-DCT) [20] A power efficient hardware implementation of this tool is a primary focus of the author's research The SA-DCT algorithm is outlined in more detail in Section 2 2 6, and the proposed implementation of the SA-DCT is outlined in Chapter 3 For boundary BABs the SA-DCT only transforms the pels that are defined as part of the VOP The SA-DCT will ignore transparent BABs and revert to a regular  $8 \times 8$  transform for opaque BABs The inverse transform (SA-IDCT) uses the decoded BABs to reproduce the texture information

**Shape-based Motion Estimation and Compensation** To exploit VOP temporal redundancy when the codec is operating in inter mode, the motion estimation and compensation tools need to operate in a shape aware fashion The two techniques most commonly used are referred to as padding and polygon matching [44] As the name suggests, padding pads out the boundary BAB texture blocks (and indeed the opaque BABs adjacent to the boundary BABs) and conventional motion estimation and compensation is carried out on the result Polygon matching is more sophisticated in that only pixels within the VOP are considered in the matching criterion used in the motion estimation process

### 2 2 6 The Shape Adaptive Discrete Cosine Transform (SA-DCT)

When the MPEG-4 standard was being defined, a transform tool was needed that supported object-based coding and the Shape-Adaptive DCT (SA-DCT) defined by Thomas Sikora and Bela Makai was adopted [20] The SA-DCT has the advantage that is based on the  $8 \times 8$  DCT-II [53] so is backward compatible with traditional block-based video codecs

The SA-DCT operates on  $N \times N$  texture blocks as before but has three modes of operation depending on the corresponding alpha  $N \times N$  block

- Alpha block completely outside the VOP  $\Rightarrow$  Transparent Block  $\Rightarrow$  Do nothing
- Alpha block completely inside the VOP  $\Rightarrow$  Opaque Block  $\Rightarrow$  Standard 2D DCT
- Alpha block on VOP boundary  $\Rightarrow$  Boundary Block  $\Rightarrow$  SA-DCT

The first case is trivial and the second case requires the conventional  $8 \times 8$  2D DCT as discussed in Section 2.2.3. In the third case, the SA-DCT is required and the steps taken are illustrated by the example in Figure 2.15. In this figure the example boundary block is on the boundary of the boat (part of the VO) and the background (non-VO). As with the block-based 2D DCT, the SA-DCT is separated into processing the columns first and then processing the resultant rows. The processing order (columns or rows first) did not matter with the block-based 2D DCT (since the formula for the 2D  $8 \times 8$  DCT is fully separable) but this becomes significant for the SA-DCT. This is because there is no formula for a 2D SA-DCT – the formula is defined for one dimension only. To resolve incompatibility issues the MPEG-4 visual standard defines that the forward SA-DCT must process columns and then rows (hence the inverse SA-IDCT must process rows then columns) [66].

The first step involves shifting all pels that belong to the VOP (as highlighted by the grey pel locations in the top left block) vertically filling any gaps. Each column is then transformed using the conventional  $N$ -point 1D DCT formula as given in Equation 2.8a. For the shape adaptive mode  $N$  may have a different value for each column depending on the number of pels in that column that are part of the VOP (previously  $N = 8$  for all columns and rows). The resultant coefficient vectors for each column are then shifted and packed horizontally and an  $N$ -point 1D DCT is performed on each row. The final coefficients for the sample  $8 \times 8$  block are shown in the bottom left pixel grid in Figure 2.15, and the top left value in this block is the DC coefficient. This process is adaptive in the sense that the DCT performed on each row and column adapts to the value of  $N$  for that row and column. The number of coefficients generated is equal to the number of pels inside the VOP for that particular texture block as defined by the corresponding alpha block.

The choice of SA-DCT is not inevitable and there are other methods for object-based transform coding, most notably by Gilge et. al. [67]. Gilge's approach is a 2D non-separable generalised orthogonal transform for coding arbitrarily sized image segments. Essentially a specific set of orthogonal basis images are derived for each kind of 2D shape. A study was carried out comparing the efficiency of the Sikora SA-DCT, the Gilge Transform and the theoretically optimal Shape Adaptive Karhunen-Loeve Transform (SA-KLT) [68]. Results show that for correlated data the SA-KLT outperforms the two sub-optimal techniques (Sikora SA-DCT and Gilge Transform) by about 2dB (considering only boundary blocks) and by only 0.5dB for a mixture of boundary and  $8 \times 8$  blocks. The study concludes that the Sikora SA-DCT and Gilge Transform have almost identical performance. The advantages of choosing the SA-DCT include:

- The SA-DCT is much less computationally complex compared to the Gilge Transform and the SA-KLT.
- It is fully backward compatible with the  $8 \times 8$  DCT used in earlier codecs.
- Its relative similarity with the  $8 \times 8$  DCT make it suitable for hardware implementation [69].



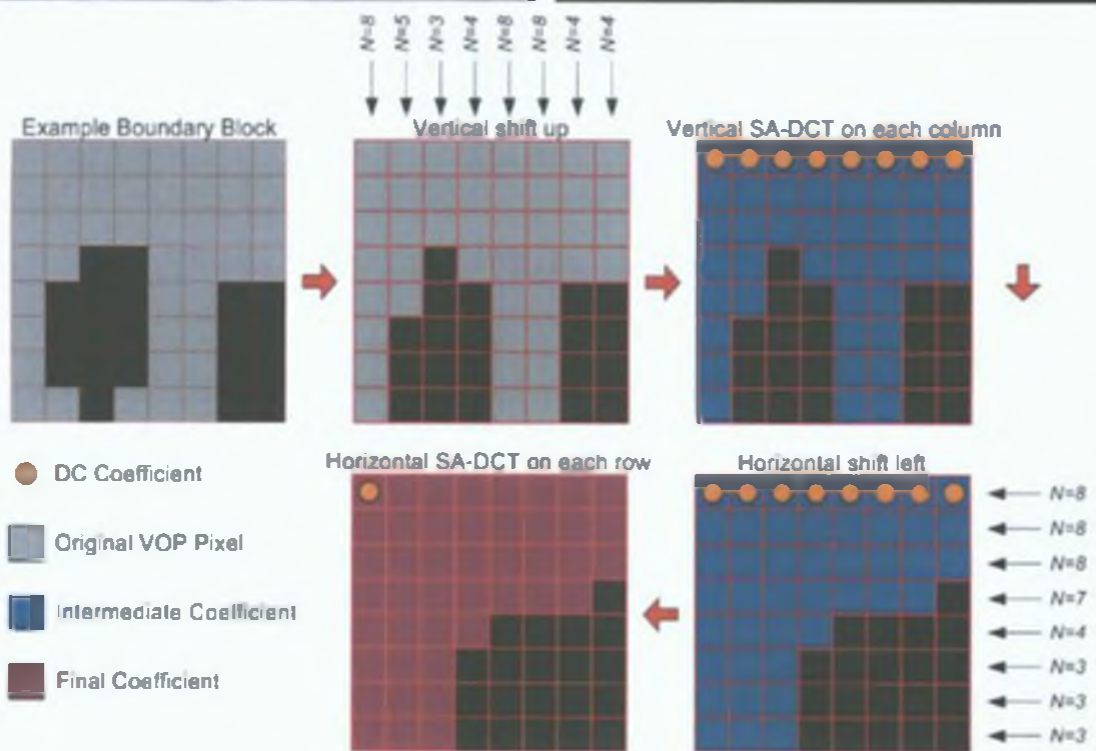


Figure 2.15: The SA-DCT Algorithm Step-by-Step

Since the SA-DCT is more computationally demanding compared to the regular  $8 \times 8$  DCT, some justification in terms of compression efficiency for adopting the SA-DCT is required. In a paper by Kauff et. al [18], the authors exploit the fact that different semantic objects in a video sequence will have different motion characteristics (e.g. rigid background with global camera motion or flexible foreground object with coherent motion). It may be more efficient to encode only global motion parameters rather than block vectors. This highlights the increased compression potential with an object-based compression scheme with semantic objects. The authors show that for the same bit rate an SA-DCT based codec outperforms an optimised MPEG-2 block-based codec by 3.3dB using global motion compensation [18]. Subjective tests show that the bit rate of the object-based codec can be dropped by a factor of three without affecting picture quality in comparison with the standard codec. Thus the adoption of the SA-DCT and an object-based MPEG-4 encoder is justified both in terms of compression efficiency and bit rate reduction. Object-based coding also has a wider application potential as discussed previously.

In addition to object-based compression, the SA-DCT also has applications in fractal based compression [70], digital watermarking [71] and image denoising [72]. Indeed, in Chapter 6 the author proposes another use of the SA-DCT – to tackle the very difficult challenge of semantic video object segmentation. This flexibility coupled with its relative computational complexity justifies hardware implementation and research into energy efficient and flexible architectures for the SA-DCT/DCT. Hence, the focus of Chapters 3 and 4 is on energy efficient hardware accelerator implementations of the SA-DCT and SA-IDCT respectively. In these chapters a detailed state-of-the-art review is given of previous approaches in the literature before outlining the proposed approach. The target platform for such accelerators is the convergent mobile device as introduced in Chapter 1. In the mobile domain, power consumption is emerging as the most important design constraint. Section 2.3 outlines the technical reasons for this trend and the architectures proposed in Chapters 3 and 4 are designed with this in mind.

### 2.2.7 Digital Video Implementation Approaches

The high computational demands of advanced video codecs such as MPEG-4 pose significant challenges for their implementation if high quality applications are to be feasible on a mobile device. Implementation strategies may be generally classified into two approaches – dedicated accelerators and programmable processors [38, 73, 21]. In general, dedicated approaches offer the greatest scope for efficient implementations of a specific task – particularly if the task is regular and has inherent parallelism (e.g. DCT, motion estimation block-matching) [74]. The architecture can be tailored with respect to the particular area, speed and power constraints of a given system. The main disadvantage of dedicated architectures, however, is their lack of flexibility (although this is not such a problem if a programmable fabric such as a Field Programmable Gate Array (FPGA) is used). A programmable based approach offers greater flexibility since modifications only require a software edit as opposed to a costly hardware re-design. However, this additional flexibility leads to decreased performance and architecture efficiency [38]. For these reasons, a common design approach is a hybrid scheme where dedicated architectures are adopted for the implementation of very frequent computationally demanding tasks, such as DCT and motion estimation, where the additional silicon overhead is deemed worthwhile. Programmable modules are used for the less demanding tasks such as codec control where the computational flow is less regular. The programmable processor and dedicated accelerators may then execute in parallel yielding speed and power benefits.

Some of the key issues requiring consideration when choosing a system architecture for a particular coding approach include [73]

- Detailed profiling of the multimedia codec to identify processing “hot spots” These hot spots may be candidates for dedicated accelerator cores
- Evaluation of real-time constraints of tasks taking into account task switching
- Exploration of inherent algorithm parallelism and redundancy that could be exploited to accelerate the computations
- Memory bandwidth requirements and memory subsystem design (on chip and off chip partitioning, suitable cache design)
- Power, area and throughput trade-offs according to design constraints

This section briefly discusses the trends and popular approaches to dedicated accelerator and processor design in a hybrid system to support mobile multimedia processing

#### **2.2.7.1 Dedicated Accelerator Options**

Traditionally, highly demanding tasks were mapped to systolic array processors based on their inherent parallelism [75]. However, the more recent trend is to move away from blindly implementing highly parallel, systolic array type structures to a more considered approach with power consumption as a primary concern [75, 76]. Multimedia applications are becoming more diverse and less predictable in their computational flow and load – MPEG-4 object-based processing is a prime example of this. In this scenario the efficiency of highly parallel architectures diminishes significantly [76]. This makes the design of dedicated architectures more challenging, and ideally these architectures should be flexible enough to be re-usable by more than one of the ever expanding application features being integrated on the convergent mobile device [75]. For example, in Chapter 6 it is argued that the SA-DCT architecture proposed in Chapter 3 could be used to tackle the ill-posed problem of video object segmentation. This is obviously advantageous if the same architecture is flexible enough to support both the segmentation processing and compression processing in an MPEG-4 object-based encoding application. As hinted at in Chapter 1 and discussed in detail in Section 2.3, the more considered design approach requires optimisations at all levels of design abstraction. The variable nature of MPEG-4 object-based processing creates the need for power efficient reconfigurable accelerator cores flexible enough to handle the irregular evolution of VOs [76]. A commercial example of a dedicated hardware core with acceleration units for DCT and others is the Hantro 4250 [77].

#### **2.2.7.2 Programmable Processor Options**

General purpose processors (GPPs), whether they be RISC or CISC based, generally do not have the right mix of processing performance, power consumption and silicon area to be viable for efficient mobile multimedia processing [75, 73]. This is despite extensions to the instruction set that exploit data level parallelism (e.g. Intel MMX) since such extensions must be backward compatible with the existing instruction set [73, 78]. This deficiency has spawned the emergence of special purpose processors (i.e.

DSPs, and media processors) to fill the void. DSPs differ from GPPs in the sense that their architectures are more efficient for mathematical computations (multiplications, additions) whereas GPPs are more suited to data movement and conditional testing. Media processors may be thought of as niche DSPs that are specifically tailored for media processing, rather than for a broad range of signal processing tasks [79]. Such architectures are based on concepts such as data parallelism, instruction level parallelism and multi-threading. They are generally a group of heterogeneous processors with highly integrated fixed-function accelerators for multimedia processing (essentially the hybrid system outlined earlier). The dedicated accelerators tend to be more integrated with the central processing core as opposed to residing as external peripherals addressable by a GPP. An example of a commercial media processor is the Philips PNX1500 [80]. Another variation in the processor domain is that of configurable and reconfigurable processors. A configurable processor allows the designer to tailor the architecture and instruction set to the desired application prior to manufacture of the processor, as opposed to tweaking the application to suit the processor (e.g. Tensilica Xtensa [81]). Reconfigurable computing engines are implemented on FPGAs. The programmable nature of FPGAs allow easy hardware edits and upgrades. However, FPGAs are also run-time configurable so hardware may change in response to the demands placed upon the system while it is running. Although FPGAs are widely considered to be very inefficient in terms of power consumption, the gap is narrowing with innovations such as the triple oxide technology introduced by Xilinx in their latest Virtex4 range [82].

### 2.2.8 Conclusions

The enabling compression technologies that underpin digital video processing have been outlined. In particular, the mathematical foundations of transform theory have been summarised to explain the behaviour of the SA-DCT/IDCT algorithms. Hardware implementations of these algorithms are a primary focus of this thesis. The evolution of the popular digital video compression standards has been described, and the clear trend is that to support enhanced applications, compression efficiency is improving at the cost of algorithm complexity. As consumers demand more features on their mobile devices, the complexity of the algorithms that underpin such features can only increase. This trend conflicts with the other trend of hardware device miniaturisation, and the consequential problem of excessive power consumption as devices become more integrated (see Figure 1.1 in Chapter 1). To the consumer, this conflict translates to an unacceptably short battery life despite the added features. Section 2.3 outlines the causes for this power consumption, how it can be analysed, and the popular techniques for addressing the issue. These techniques are exploited by the author in the design of the SA-DCT/IDCT hardware accelerators in Chapters 3 and 4 and also in the design of the proposed EDA algorithm for constant matrix multiplication hardware optimisation in Chapter 5.

## 2.3 Low Power Design

As outlined in Chapter 1, the energy dissipation in VLSI circuitry is increasing at a troublesome rate due to ever increasing levels of integration and device scaling. Consequentially, designers of video accelerators must take power consumption seriously as a design constraint alongside the traditional constraints of area and speed. This section outlines the sources of power consumption in digital CMOS circuitry, methods of estimating and analysing power, as well as popular techniques for reducing power.

### 2 3 1 Circuit Power Dissipation Phenomena

It is essential to understand the sources of power dissipation in an integrated circuit before discussing techniques to reduce it. A commonly used model to describe power consumption is given in Equation 2 10, which has four terms – dynamic power (due to switching), short-circuit power, leakage power (incorporating reverse biased junctions and sub-threshold effects) and static power (due to circuits with a constant source of current between power rails) [83]

$$P_{avg} = P_{dynamic} + P_{short} + P_{leakage} + P_{static} \quad (2 10)$$

Each component will be discussed in turn in this section. It should be noted that the distribution of power consumption between these components is inextricably linked to the specific application and the technology employed.

#### 2 3 1 1 Dynamic Power Consumption

Switching activity in a circuit causes dynamic power consumption, and the most significant source of dynamic power in a CMOS circuit is the charging and discharging of capacitances [84]. These capacitances may be intentionally fabricated in the circuit but it is more likely that they are parasitic. The latter arise due to the parasitic effects of interconnection wires and transistors. This is inherent in a circuit and cannot be ignored – especially in deep sub-micron devices.

Toggling between logic zero and logic one at a logic node discharges and charges the equivalent capacitance at that node as demonstrated in Figure 2 16. In this figure (taken from the derivation in [84])  $V$  is an ideal constant voltage source and  $R_c$  and  $R_d$  are the resistances of the charging and discharging circuits respectively. The switch is a model for the change in logic state at the node. The capacitor  $C_L$  is a model of the capacitive load at the node. This load is caused by parallel plate capacitive effects that are present as a result of the layout and fabrication process of the transistors and wire interconnect. Accurate modelling of this capacitance is complicated, but to a first order  $C_L \propto W \times L$ , where  $W$  is the transistor channel width and  $L$  is the transistor channel length [85, 86, 87]. It is more common to find  $L$  referred to as the “process technology” or similar in the literature. For example, if discussing 90nm process technology this implies that  $L = 90\text{nm}$ .

During the charging cycle from time  $t_0$  and  $t_1$  the energy  $E_s$  is given by Equation 2 11 assuming that the capacitive load  $C_L$  is fully charged.

$$\begin{aligned} E_s &= \int_{t_1}^{t_0} V_{Lc} dt \\ &= C_L V \int_{t_1}^{t_0} \frac{dv_c(t)}{dt} dt \\ &= C_L V^2 \end{aligned} \quad (2 11)$$

Part of  $E_s$  is stored in the capacitor ( $E_{cap}$  in Equation 2 12) and the rest is dissipated as heat energy in

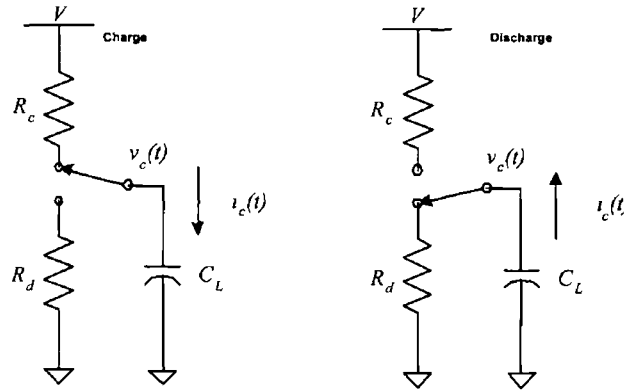


Figure 2 16 Dynamic Charging and Discharging a Capacitive Load

$R_c$  ( $E_c$  in Equation 2 13)

$$\begin{aligned}
 E_{cap} &= \int_{t_1}^{t_0} v_c(t) i_c(t) dt \\
 &= \frac{1}{2} C_L V^2
 \end{aligned} \tag{2 12}$$

$$\begin{aligned}
 E_c &= E_s - E_{cap} \\
 &= \frac{1}{2} C_L V^2
 \end{aligned} \tag{2 13}$$

Assuming that the capacitor is fully discharged from time  $t_1$  to  $t_2$  the energy dissipated  $E_d$  in the discharge resistor  $R_d$  is identical to  $E_{cap}$ . If the circuit operates on a clock at  $f$ Hz and  $C_L$  is charged and discharged with a probability  $\alpha$ , the power dissipation at the node is given by Equation 2 14

$$P = C_L V^2 f \alpha \tag{2 14}$$

A more general version of Equation 2 14 for a complete system operating at  $f$ Hz involves summing the above equation over all nodes in the circuit as shown in Equation 2 15

$$P_{dynamic} = \sum_i P_i = \sum_i C_i V_i^2 f \alpha_i \tag{2 15}$$

It is immediately clear from the above equation that the supply voltage is the dominant term and has a quadratic effect on dynamic power consumption. Although reducing  $V$  will reduce the power consumption, it has other repercussions, including increasing the propagation delay, as discussed in Section 2 3 3. There are some assumptions made in deriving Equation 2 15 that must be taken into account, although most digital CMOS circuits satisfy them. They are as follows:

- 1 The capacitance  $C_L$  is constant
- 2 The voltage  $V$  is constant
- 3 The capacitor is fully charged and discharged

It is interesting to note that although these restrictions apply, the result is independent of the charging

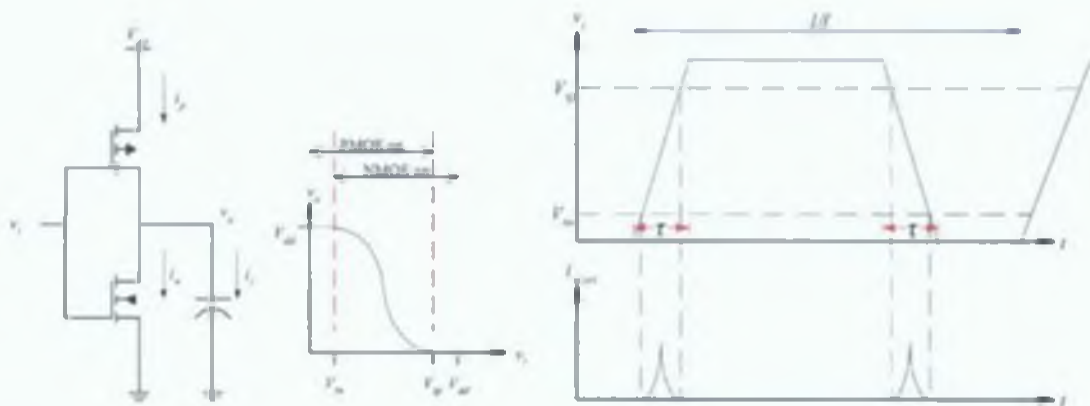


Figure 2.17: CMOS Inverter Transfer Characteristic and Short-Circuit Current

and discharging circuitry ( $R_c$  and  $R_d$  may be non-linear or time varying), the length of the charging and discharging cycle and the voltage and current waveform. Equation 2.15 is the dominant term in Equation 2.10 for CMOS circuits operating at mid to high frequencies.

### 2.3.1.2 Short-Circuit Power Consumption

Short-circuit power is another source of power consumption related to the switching activity in a CMOS digital circuit. The dynamic power consumption term derived in the previous section did not take into account the fact that input waveforms have finite rise and fall times. In the CMOS inverter circuit as shown in Figure 2.17, ideally only one transistor is on at any one time. However, owing to the finite transition time of the input waveform, both transistors will be on for a very short time thus providing a conductive path between the supply rail and ground, which dissipates power. The amount of short circuit power consumed depends on the short circuit current characteristic when the input voltage  $v_i$  lies between the thresholds for the NMOS ( $V_{in}$ ) and PMOS ( $V_{ip}$ ) devices. This current is dependent on the following factors [84, 83, 88]:

- The duration and slope of  $v_i$
- The I-V curves of the PMOS and NMOS transistors, which in turn depends on their sizes, process technology, temperature, etc.
- The output load capacitance

Short circuit is difficult to formulate but a commonly used first order model is given in Equation 2.16 [84]. This equation assumes a zero output load capacitance, which is not practical, so is rarely used to actually compute short circuit power. It is used more to show how short circuit power depends on each of the contributing factors.

$$P_{short} = \frac{\beta}{12} (V_{ip} - V_{in})^3 \tau f \quad (2.16)$$

The parameters in Equation 2.16 are  $\beta$  (related to the size of the transistors),  $f$  (the switching frequency) and  $\tau$  (the switching transition time). The short circuit current characteristic for a CMOS inverter is given in Figure 2.17. Taking output capacitance into account, it follows that the larger the output capacitance the smaller the short circuit current. This is because the output voltage responds more slowly and both

transistors are on for a shorter duration. However, by Equation 2.15, a larger capacitance results in a larger dynamic power component. Although it may seem that there is an optimal load capacitance for minimal power consumption this is not the case [84]. The short circuit power component is typically 10% of the dynamic component so increasing the load capacitance will increase the overall power figure even though the short circuit component is decreasing [83]. It is therefore necessary to minimise the load capacitance for low power design. It is also worth noting that although a sharper input transition time will improve the power consumption of the receiving network, a more power consumptive driving network is required to sustain a short  $\tau$ . As with all decisions in low power design, a trade-off is necessary and the rule of thumb used is that the input transition time should be comparable to the propagation delay of the gate.

### 2.3.1.3 Leakage Power Consumption

Leakage power consumption arises from the non-ideality of semiconductor junctions. Leakage is a form of current that is generally not desired for the normal operation of a digital circuit but exists due to the nature of the devices employed. For circuits that use MOS transistors there are two main sources of leakage current (and thus power) – reverse biased PN-junction current and sub-threshold channel conduction current.

**PN-junction Leakage** To illustrate where reverse biased currents come from in a CMOS circuit consider the inverter in Figure 2.17. A physical implementation of the circuit is shown in Figure 2.18 where the input is logic '1' and the output is logic '0'. In this case the PMOS transistor is off forming a reverse biased PN-junction between the bulk and the PMOS drain as shown in the figure. A similar diagram can be deduced if the input is logic '0' for reverse bias in the NMOS transistor. In both cases the leakage power is based on the standard formula for a reverse biased PN-junction as shown in Equation 2.17 [83].

$$P_{\text{leakage, junction}} = V_{dd} \left| I_s \left( e^{\frac{V}{V_T}} - 1 \right) \right| \quad (2.17)$$

In the above equation,  $V$  is the voltage across the junction (negative if reverse biased),  $V_T$  is the thermal voltage (usually about 26mV at room temperature) and  $I_s$  is the reverse saturation current. This power component is dependent on the following factors [84]:

- Operating temperature ( $I_s$  doubles with every ten degree increase)
- Fabrication process
- Junction area

It is worth noting that this component of leakage power is largely independent of operating voltage and even with a small reverse bias the leakage current will equal the reverse saturation current. Generally reverse biased leakage is not a problem in terms of power for a large chip and there is little scope for a designer to reduce it in any case.

**Sub-threshold Channel Leakage** The second leakage component is sub-threshold conduction through a MOS device channel. Even if a device is logically off (Gate-Source voltage  $V_{GS}$  is less than the device



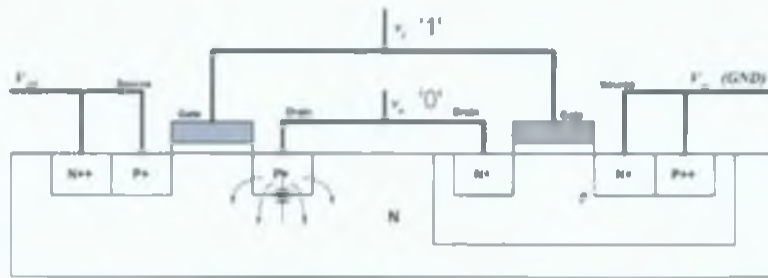


Figure 2.18: Diode Leakage Current in a CMOS Inverter

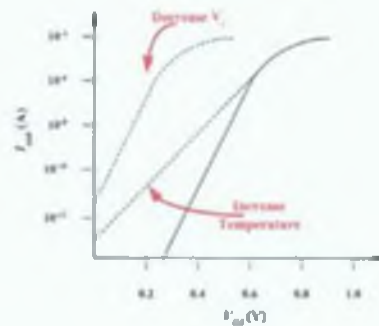


Figure 2.19: Sub-threshold Current as a Function of Gate Voltage

threshold voltage  $V_t$  for NMOS) there is still a non-zero leakage current through the channel due to weak inversion effects. The formula for sub-threshold current is given in Equation 2.18 [84].

$$I_{leakage\_sub} = I_0 e^{\left(\frac{V_{GS} - V_t}{nV_T}\right)} \left(1 - e^{-\frac{V_{DS}}{V_T}}\right) \quad (V_{DS} \gg V_T \text{ if the device is off})$$

$$I_{leakage\_sub} \approx I_0 e^{\left(\frac{V_{GS} - V_t}{nV_T}\right)} \quad (2.18)$$

$$\therefore P_{leakage\_sub} \approx V_{DD} I_{leakage\_sub}$$

$I_0$  is the channel current when  $V_{GS}$  is equal to  $V_t$  and  $\gamma$  is a constant parameter that depends on the device fabrication (a value typically between 1.0 and 2.5) [84]. The exponent factor has a negative value when the device is off ( $V_{GS} < V_t$ ) so  $I_{leakage\_sub}$  decays exponentially as  $V_{GS}$  decreases. The slope at which  $I_{leakage\_sub}$  decays is the important factor in low power design and is dependent on device temperature and threshold voltage  $V_t$ . The influence of varying temperature and threshold voltage on the sub-threshold current is illustrated in Figure 2.19. Traditionally, the sub-threshold component did not give rise to concern but with device scaling ( $V_t$  decreasing and associated tolerances tightening) and higher levels of integration (giving rise to temperature increases), its importance is increasing. It is foreseen that sub-threshold conduction will be a limiting design factor in the future [89].

### 2.3.1.4 Static Power Consumption

The final power consumption component is due to steady-state static dissipation. As the name suggests, this type of dissipation is not caused by switching activity but by the static circuit architecture. Strictly speaking, pure CMOS circuits only consume power during switching. However, there are two situations

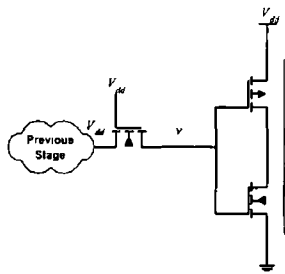


Figure 2 20 Degenerated Voltage Level Driver

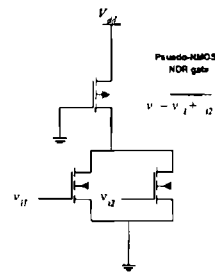


Figure 2 21 A Pseudo-NMOS NOR Gate

when CMOS circuits may consume power in steady state mode – degenerated voltage levels feeding CMOS gates and pseudo NMOS circuit architectures [83]

**Reduced Voltage Levels Feeding CMOS Gates** Consider the common scenario of a CMOS inverter being driven by a previous stage buffered by an NMOS pass transistor as shown in Figure 2 20 If the output voltage of the previous stage is  $V_{dd}$  and the pass transistor is conducting (i.e. gate voltage is  $V_{dd}$ ) then  $v_i$  will equal  $V_{dd} - V_{dtn}$  (a degraded logic '1') where  $V_{tn}$  is the threshold voltage for the NMOS pass transistor In this scenario the PMOS transistor should be fully off and the output voltage will be at logic '0' However, since  $v_i$  is a degraded logic '1' the PMOS transistor is weakly on (the gate-source voltage is very close to the threshold  $V_{tp}$ ) and hence there is a static current path between the supply rails A similar problem occurs if the previous stage is driving logic '0' and the pass transistor is PMOS This component of power consumption may become appreciable if the circuit is idle for most of the time

**Pseudo-NMOS Logic** The PMOS pull-up network of a pseudo-NMOS circuit is a single PMOS device whose gate is grounded (i.e. always on) If the output voltage is driven low, there is a static conduction path between supply rails To illustrate, consider the pseudo-NMOS style NOR gate as shown in Figure 2 21 If any of the input signals are logic '1' there will be static current flow between the rails thereby dissipating power Although they contribute more to static power consumption than pure CMOS circuits, pseudo-NMOS architectures are commonly used because of their area efficiency They tend to be used only for implementations that require complex logic switching at high frequencies since their area efficiency implies a reduced switching capacitance (and hence lower dynamic power consumption) Pseudo-NMOS circuits should not be used if the circuit is likely to be idle at an output logic '1' for long periods of time as the extra static power consumption then negates the area gain

### 2 3 1 5 Summary and Trends

The main power consumption components in a digital CMOS circuit have been introduced They can be generally categorised as follows

- Dynamic/Active Components ( $\approx 90\%$ )
  - Node switching (dominant factor)
  - Short circuit currents
- Static/Standby Components ( $\approx 10\%$ )

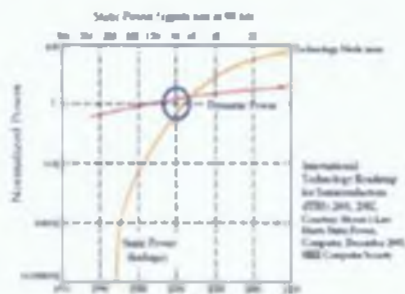


Figure 2.22: Static vs. Dynamic Power

- Leakage currents
  - Reverse-biased junctions
  - Sub-threshold conduction
- Steady-state static current
  - Reduced Voltage Level Drivers
  - Pseudo NMOS logic

Traditionally, the dominant component has been dynamic power consumption and engineers focused on minimising Equation 2.15. However, this is changing due to physical phenomena (mainly sub-threshold leakage) as process technologies shrink below 90nm. It is estimated that when devices are scaled to 45nm (around the year 2010) the static component will be equal in proportion to the dynamic component [89]. This trend is illustrated by Figure 2.22 [90]. This is causing great concern in the low power research community and it is seen as the primary obstacle in keeping power levels under control in deep sub-micron devices (sub-100nm process technologies) [32, 91, 92]. Indeed, in deep sub-micron design, other significant sources of leakage power consumption are emerging that were previously negligible with larger process technologies [93]. These include gate leakage (tunnelling), gain induced drain leakage and punchthrough. However, the trend projected by Figure 2.22 will only happen if appropriate steps are not taken to address the causes of excessive static power in deep sub-micron devices. As discussed in Section 2.3.3, however, this is already happening with many innovative techniques emerging that specifically target static power. Clearly, in deep sub-micron design, all of the power consumption terms in Equation 2.10 are equally important and each warrants significant research attention.

For successful low power implementation of a convergent mobile multimedia device, power issues must be analysed and considered at each stage of the design process from system design to manufacture (see Section 1.2.3). Section 2.3.2 discusses meaningful metrics for power analysis and outlines some common techniques for power analysis of software and hardware design flows. Subsequently, Section 2.3.3 summarises a range of low power design techniques spanning all abstraction levels. Most of these approaches are based on common sense ideas that either directly or indirectly aim to reduce some or all of the terms in Equation 2.10. Unfortunately, reduction of Equation 2.10 usually comes at the cost of other system constraints and these repercussions are also discussed in Section 2.3.3. The overall goal is to find a sweet spot between speed, area and power that enables the digital dream applications envisaged for the emerging convergent device.



Figure 2.23 Using Power [W] as a Design Metric

## 2.3.2 Power Analysis & Estimation Techniques

### 2.3.2.1 Power Metrics

Having established that power has become the most important design constraint and discussed the sources of power consumption in a digital CMOS circuit, it is now necessary to elaborate on the techniques designers use to tackle the issue. As with everything else in engineering design, there is no golden procedure to follow and trade-offs are inevitable. Therefore, there are many solutions to a given problem and designers need a metric to compare alternatives in terms of their low power properties. As will be discussed in this section, the obvious metrics of power (in Watts) and energy (Joules) have serious flaws if taken in isolation [94].

Using instantaneous or average power in Watts as a metric is useful for designers when selecting a power supply, figuring out cooling requirements and ensuring that physical packaging limits are not exceeded. Also, given a battery with a certain amount of energy, the average power in Watts can be used to derive the battery life in hours. Power in Watts may be misleading, however, as a metric for comparing designs, it gives no indication as to how long it takes a circuit to complete an operation. Consider two designs **A** and **B**. The two designs may have different peak power levels as shown in Figure 2.23 for the same computation [95]. Although **A** seems worse than **B** in terms of Watts, **B** takes much longer to perform the operation. Thus, comparing **A** and **B** based solely on Watts is not a good way to compare design efficiency, as it does not take into account the temporal factor, and designs are usually specified to complete a computation within a certain temporal window. As shown in Equation 2.15, power is proportional to operating frequency. Designs **A** and **B** could have identical architectures with **B** simply operating at a lower frequency. Both designs will perform the same task and draw the same amount of energy from the battery but it will take **B** longer to do this than **A**.

The previous discussion hints that using energy (Joules) is a better metric for comparing design alternatives. Energy per operation is obtained by multiplying power by operation delay (it is also equal to the area under the curves in Figure 2.23). Although the base unit is the Joule, the more commonly quoted unit is W/Mhz (i.e. dissipation per clock cycle). When comparing the performance of two processors however, W/Mhz can be misleading since different processors may require different numbers of clock cycles for the same instruction. As a consequence, it is usually normalised to  $\mu\text{W}/\text{MIPS}$ , a more objective metric since it has the unit of Watt-Second per Instruction. It is argued, however, that  $\mu\text{W}/\text{MIPS}$  favours processors with simpler architectures with inferior computation engines [84]. In addition, energy has a quadratic dependence on voltage so if designs **A** and **B** are identical but **B** is operating at a lower voltage, theoretically **B** will consume less energy per operation. However, scaling the voltage has complex effects on delay and power performance (discussed in detail in the next section), so whether **B** is truly a better low power alternative depends on many other factors. As a result, using energy as a metric in isolation suffers from similar problems as using Watts.

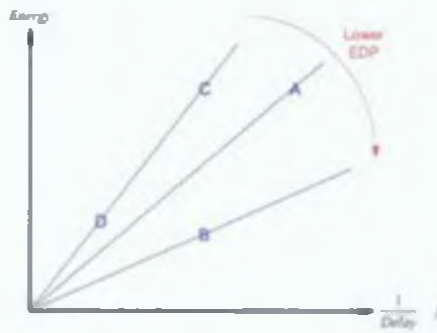


Figure 2.24: Energy Delay Product (EDP) Solution Space

Essentially, what is required is a metric that takes temporal performance into account as well as energy efficiency. A solution is the energy-delay product (EDP), which as the name suggests is the product of the energy per operation by the delay per operation [94, 96]. The advantage of this metric is that a design with a low EDP is guaranteed to be better in terms of energy efficiency and performance than one with a high EDP. Consider Figure 2.24 where the straight lines represent constant EDPs in a 2D solution space, and four design solutions are also mapped to this space. The graph illustrates that design B is the best solution in terms of low power viability since it has the lowest EDP. The question is how can a designer differentiate two designs with identical EDPs? In Figure 2.24 designs C and D have the same EDP but D is more energy efficient than C but runs at a lower frequency. This again implies that a designer can trade energy for speed. Although it improves again over Watts and  $\mu\text{W/MIPS}$ , the EDP is not sufficient in isolation if designs have the same EDP.

In general, the choice of metric depends on the type of analysis being carried out and the intended application. No single measure is sufficient for all purposes and it is a designer's responsibility to correctly interpret any analysis carried out.

### 2.3.2.2 Software Power Analysis

The power consumption due to the execution of software is heavily dependent on the underlying hardware processor that the software runs on. Software power analysis generally refers to the analysis of a processor instruction set and the power consumed by the processor due to the instructions themselves, operand values, addressing modes and inter-instruction effects.

By Ohm's Law, the instantaneous power  $p(t)$  consumed by a processor at time  $t$  is the product of instantaneous current  $i(t)$  and instantaneous voltage  $v(t)$  (i.e.  $p(t) = i(t)v(t)$ ). Assuming that the voltage is kept constant (i.e. ignoring the possibility of dynamic voltage scaling), then  $p(t) = i(t)V_{dd}$  where  $V_{dd}$  is the constant supply voltage. If a software program takes  $T_{sw}$  seconds to execute on the processor, then the energy consumed by the program can be obtained by integrating the  $i(t)V_{dd}$  waveform over  $T_{sw}$ . Assuming an accurate value for  $T_{sw}$  can be obtained using a profiling tool, the only difficult task is recording the current waveform  $i(t)$ . One option is to use simulation based methods where the software is simulated on some low level model of the CPU (HDL or even SPICE). In reality it may be impossible to obtain such a model or impractical if the program is very long or the processor is very complex. An alternative approach is to make a physical measurement by placing an ammeter in series

between  $V_{dd}$  and the power pin of the processor. This option is only viable, of course, if a prototype of the processor is available. The approach here is to code the program in a long iteration loop. This should result in a periodic  $i(t)$  waveform, where one period represents the current drawn by one iteration of the program. However, this may require expensive data acquisition systems for accurate readings.

From a software design perspective, the disadvantage of the simulation approach, apart from the size and slowness of the run, is the fact that processor models are rarely available to software designers so power consumption due to software execution can typically only be measured after manufacture [97]. Since physical measurement is a laborious error-prone process, it makes sense to make measurements that characterise the processor's instruction set [98]. Instruction level analysis measures the current drawn for specially created instruction sequences. This then provides all the required information for instruction level analysis and the fundamental information to quantify software power at algorithmic level for any program that runs on the particular processor. With an accurate power model of an instruction set, this information may be integrated into a compiler that could generate machine code optimised for power as well as execution time and code size.

In [98], each instruction is characterised with a base cost that represents the power cost for basic processing needed for the instruction (ignoring stalls and cache misses which are modelled separately). The current is measured for a loop of several instances of the given instruction. Clearly, the cost varies with operand and address values so averages are used (e.g. a variation of only 5% for 486DX2 and SPARClike but greater for DSPs). Instructions can then be grouped into classes depending on the amount of current they draw. Inter-instruction effects are also modeled since the effect of circuit state is important. Again it was found that inter-instruction effects are limited for GPPs but more pronounced for DSPs (since they are smaller and have smaller caches). Other inter-instruction effects include pipeline stalls, write buffer stalls and cache misses. These are modeled as energy overheads characterised by artificial programs where these effects occur repeatedly. Tiwari's model for algorithm energy requirements  $E_{alg}$  is as follows:

$$E_{alg} = \sum_i (Base_i \cdot N_i) + \sum_{i,j} (Ovhd_{i,j} \cdot N_i) + \sum_k Energy_k \quad (2.19)$$

where  $N_i$  is the number of times instruction  $i$  is executed,  $Base_i$  is the base energy cost of  $i$ ,  $Ovhd_{i,j}$  is the circuit state overhead when  $i$  and  $j$  are adjacent and  $Energy_k$  is the energy overhead of stalls and cache misses. Tiwari's model with extensions and variations has been used to characterise a wide variation of processor architectures, including the ARM7TDMI [99, 100, 101] and the TI VC5510 DSP [97].

A drawback to this approach is that it requires exhaustive characterisation of the entire instruction set plus inter-instruction effects. Work by Sinha and Chandrakasan has shown that a lot of overheads are common across instructions and as a result the overall current consumption of a program is a weak function of the instruction stream and to a first order only depends on the operating frequency and voltage [102, 103, 104]. Second order effects (due to instructions) exist but their experiments show that they are less than 7% for a set of benchmark programs. The technique proposed by Sinha and Chandrakasan is simple and fast with experiments showing that the accuracy is within 3% of actual measurements on a StrongARM SA-1100 and Hitachi SH-4. They conclude that the common overheads (caches, decode etc.) in modern microprocessors are large and overshadow instruction specific variations.

When deriving their first order model, Sinha and Chandrakasan show that the maximum average

Table 2.1 Weighting Factors for  $K=4$  on the StrongARM Processor

Class	Weight	Value
Instruction	$w_1$	2.1739
Sequential Memory Access	$w_2$	0.0311
Non-sequential Memory Access	$w_3$	1.2366
Internal Cycles	$w_4$	0.8289

current variation over a range of programs is approximately 8%, so to a first order

$$E_{alg} = V_{dd} I_0(V_{dd}, f) \Delta t \quad (2.20)$$

This is surprising because it implies that the power dissipated by a processor is program independent! However, what is more important is the energy consumed – algorithms are differentiated to first order by their execution time and not by their instruction stream. Indeed this seems to be true for the StrongARM but may not be true in the case of datapath dominated processors such as DSPs. Their second order model considers instruction effects and classifies  $K$  energy differentiated classes  $C_k$  of instructions for a processor (For StrongARM  $K = 4$ : Instruction, Sequential Memory Access, Non-Sequential Memory Access and Internal Cycles). The proposed second order current consumption equation is

$$I(V_{dd}, f) = I_0(V_{dd}, f) \sum_{k=0}^{K-1} w_k c_k \quad (2.21)$$

where  $c_k$  is the fraction of total cycles in a given program that can be attributed to instructions in the class  $C_k$ , and  $w_k$  are a set of weights. The weights for the StrongARM are shown in Table 2.1 which have been solved using the least mean square approach for values obtained for a variety of benchmark programs [104]. The key advantage of this approach is that the instruction effects have been modelled without resorting to elaborate instruction level profiling.

Despite the advantages of the instruction set characterisation approach to software power analysis, the physical measurement approach is used in this thesis. This is because efficient software implementation is not the primary focus of this thesis and the appropriate equipment for physical measurement was available to the author. This facilitates rapid software power analysis – see Chapters 3 and 4 for details on the experiments conducted. The results presented there justify implementing the SA-DCT/IDCT in hardware as opposed to software for energy and power reasons.

### 2.3.2.3 Hardware Power Analysis

In this thesis, “hardware power” refers to the power consumed by a dedicated hardware accelerator. This is distinct to the “software power” consumed by a programmable processor caused by software code executing on it. Unlike the case of software, it is rare that an engineer has a physical embodiment of the hardware for power analysis early in the design flow. Hence, physical measurement approaches like Tiwari’s [98] are not suitable and some simulation-based approach is necessary. There are three broad categories of simulation-based approaches: probabilistic, statistical and event-based. All three are closely related.

Since power consumption is highly dependent on switching activity, the power consumed by a hard-

ware module is input pattern dependent. For this reason power estimation is a “moving target” [105]. A probabilistic approach uses the switching probabilities of each of the module input pins and propagates this data into the circuit. Using a delay model for the intended process technology, the propagated switching probabilities can be used to estimate the power consumption of the entire block. However such an approach requires intimate knowledge of the data characteristics. To get around this, a statistical approach simulates the circuit repeatedly with random data. The switching power is estimated using the delay model for the circuit. The power value should converge to an average value and a technique like the Monte Carlo method can be used to decide when to stop [105]. It is clear that the probabilistic and statistical approaches are very similar. The statistical approach is needed if the input signal transition probabilities are unknown, and essentially the statistical approach uses simulation to experimentally estimate these probabilities.

Event-based power estimation involves simulating a circuit using typical input patterns. The switching activity at each node in the circuit is monitored, and after the simulation the average power consumption can be calculated by summing the average power consumed by each node in the circuit. The potential difficulty with this approach is that the designer needs to know what “typical patterns” are for the circuit in question. This could be a problem if a designer is in the early phases of working on a sub-module that is part of a larger design, and the timing of the interface signals to the sub-module are not yet established. Also, for large circuits, monitoring each node can be very computationally intensive.

#### **2.3.2.4 Power Measurement Approach Adopted**

The PrimePower tool by Synopsys is capable of probabilistic, statistical and event-based power analysis [106]. Since the SA-DCT and SA-IDCT circuits proposed in this thesis are relatively small, and “typical patterns” are available (i.e. video test sequences), event-based power analysis is used due to its simplicity. A Verilog testbench has been designed to stimulate the SA-DCT/IDCT accelerators with test data from a range of CIF/QCIF resolution test sequences. The testbench can also run in random data mode. For a given simulation run, the simulator saves a Value Change Dump (VCD) file. This records the switching transitions on each node in the hardware accelerator for the duration of the simulation, and PrimePower uses this VCD file in association with the target technology library information to calculate the average power consumed. VCD analysis causes problems if a timing simulation is run for a long time since VCD files may grow large (in excess of 1GB is not uncommon). Large VCD files means that running power analysis is slow and laborious, and indeed some tools cannot handle VCD files larger than 2GB. Future research is needed to find solutions to this file size problem.

It must be noted that the simulations run to generate results in this thesis are back-annotated dynamic simulations using the Synopsys VCS simulator of the gate level netlist generated by the Synopsys Design Compiler synthesis tool. In this context, back-annotation means that the netlist is annotated with the delays and parasitics (capacitances and resistances) of the circuit logic elements as well as the interconnect delays. Hence the VCD created should reflect the switching that would occur in a real physical realisation of the circuit. However, analysis using the post-synthesis netlist uses wire load models to estimate interconnect delays and parasitics. There is no layout information to indicate the length of the wires. This is acceptable for older technologies where the delays through the cells (logic gates and registers) is much greater than the wire delays. However in deep sub-micron technologies like 90nm the interconnect delays are actually the longest – hence accurate layout info is crucial for accurate timing.



information. Ideally the netlist should be passed to a physical layout tool. This tool would then extract accurate parasitic information for PrimePower to achieve more accurate power analysis. This additional step has not yet been completed as it is deemed to be outside the scope of this thesis.

PrimePower is sophisticated enough to be able to break down the average power value into dynamic and static components. Synopsys defines dynamic power to comprise of two sub-components (switching power and internal power). They define switching power as follows [106]:

“The switching power of a driving cell is the power dissipated by the charging and discharging of the load capacitance at the output of the cell. The total load capacitance at the output of a driving cell is the sum of the net and gate capacitances on the driving output.”

“Because such charging and discharging are the result of the logic transitions at the output of the cell, switching power increases as logic transitions increase. Therefore, the switching power of a cell is a function of both the total load capacitance at the cell output and the rate of logic transitions.”

And internal power [106]:

“Internal power is any power dissipated within the boundary of a cell. During switching, a circuit dissipates internal power by the charging or discharging of any existing capacitances internal to the cell. Internal power includes power dissipated by a momentary short circuit between the P and N transistors of a gate, called short-circuit power.”

Synopsys define static power as follows [107]:

“Static power is the power dissipated by a gate when it is not switching, that is, when it is inactive or static. Static power is dissipated in several ways. The largest percentage of static power results from source-to-drain sub-threshold leakage, which is caused by reduced threshold voltages that prevent the gate from completely turning off. Static power is also dissipated when current leaks between the diffusion layers and the substrate. For this reason, static power is often called leakage power.”

Clearly power analysis of a hardware accelerator is not useful in isolation. Rather, benchmarking against other implementations is required. Ideally the designer has access to the HDL code of the other implementations and can run simulations using a common testbench for a common technology. Unfortunately, this scenario is unlikely so typically some normalisations are required for benchmarking as discussed in the next section.

### 2.3.2.5 Power and Energy Benchmarking

The power  $P$  dissipated by a circuit depends on the input stimulus pattern, but also on other factors like process technology  $L$ , voltage  $V$  and frequency  $f$  as discussed in Section 2.3.1. The energy  $E$  consumed by a circuit is defined as  $E = P \times D$ , where  $D$  is the delay required by the circuit to compute a given task. This delay can be calculated as  $D = C \times 1/f$ , where  $C$  is the clock cycle latency of the circuit.

If HDL code were available for other prior art architectures, accurate energy and power benchmarking is relatively straightforward. This is because the flow described in section 2.3.2.4 could be followed for all competing architectures, using the same testbench stimulus and the same parameter values for  $V$

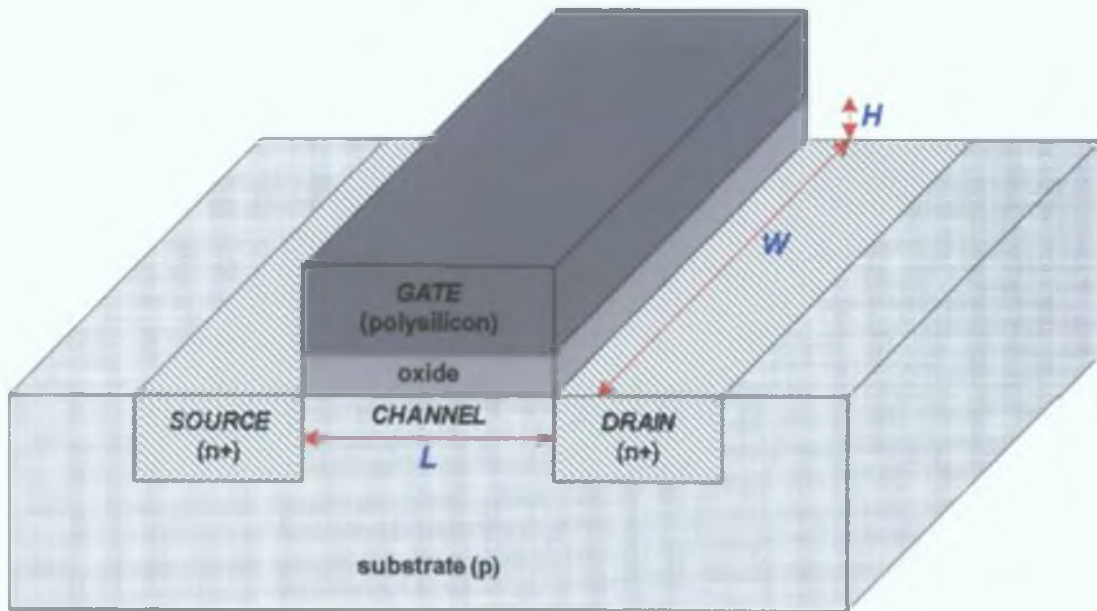


Figure 2.25: NMOS Transistor

and  $L$ . An important point to note is that the parameter  $f$  may not be equal for competing architectures, since it is design dependent as discussed subsequently. In the absence of HDL code for other prior art architectures, energy and power benchmarking against prior art must be normalised with respect to the parameters  $L$ ,  $V$ ,  $f$  and  $D$  where appropriate.

To understand the reasoning behind the normalisation formulae adopted in this thesis, device scaling according to Moore's Law [1] must be analysed more closely. This is required since any attempt to normalise two different technologies is effectively the same process as device scaling because all parameters must be normalised according to the scaling rules. To illustrate what device scaling means at the technology level, consider a simplified diagram of an NMOS transistor in Figure 2.25. This diagram clearly illustrates the channel length  $L$ , the channel width  $W$  and the gate oxide thickness  $H$ . The scaling formula when normalising from a given process  $L$  to a reference process  $L'$  is given by Equation 2.22.

$$\begin{aligned} \text{Reference Process} &= S \times \text{Process} \\ L' &= S \times L \end{aligned} \quad (2.22)$$

If  $S < 1$ , this effectively means that  $L'$  is a smaller process technology compared to  $L$  (e.g. 90nm versus 130nm). The converse is true if  $S > 1$ . Usually the channel width  $W$  is also scaled by the same factor  $S$ , as shown in Equation 2.23.

$$W' = S \times W \quad (2.23)$$

The voltage  $V$  is scaled by a factor  $U$  as shown in Equation 2.24.

$$\begin{aligned} \text{Reference Voltage} &= U \times \text{Voltage} \\ V' &= U \times V \end{aligned} \quad (2.24)$$

Again if  $U < 1$  this means that  $V'$  is a smaller voltage compared to  $V$  and vice versa. If  $S = U$ , this is commonly referred to as constant field scaling. The case where  $S \neq U$  is termed general scaling. The gate oxide thickness  $H$  is usually scaled by the factor  $U$ , i.e.  $H' = U \times H$ .

With the scaling factors established, the task now is to investigate how the various factors influence the power  $P$  and energy  $E$ . The normalisations used in this thesis assume that dynamic power is the dominant component (i.e. leakage power is ignored). This assumption gives reasonable first order normalisations [85]. The dynamic power equation for a circuit node (Equation 2.14) is restated here for convenience.

$$P = C_L V^2 f \alpha \quad (2.25)$$

From Equation 2.25 it is clear that  $P$  depends on the capacitive load switched  $C_L$ , the voltage  $V$ , the operating frequency  $f$  and the node switching probability  $\alpha$ . The switching probability  $\alpha$  depends on the architecture of the design itself and is independent of  $V$ ,  $L$  and  $f$ . Hence it does not need to be scaled when normalising  $P$  and  $E$ .

To normalise  $P$  with respect to  $V$ , Equation 2.26 is applied since by Equation 2.25  $P$  is quadratically dependent on  $V$ .

$$P'(V) = P \times U^2 = \text{Power After } V \text{ Scaling} \quad (2.26)$$

It is clear that if  $U < 1$ ,  $P'(V) < P$ . This is expected since the voltage has been reduced. Normalising  $P$  with respect to  $W$  and  $L$  is not as straightforward since neither  $W$  or  $L$  appear explicitly in Equation 2.25. However, consider Equation 2.27 which approximately<sup>7</sup> expresses the load capacitance of the NMOS transistor in Figure 2.25 [86, 85].

$$C_L = \frac{\epsilon_0 L W}{H} \quad (2.27)$$

In this equation  $\epsilon_0$  is the permittivity of the gate oxide. By the scaling equations presented earlier,  $C_L$  scales according to Equation 2.28.

$$C'_L = C_L \times \frac{S^2}{U} \quad (2.28)$$

Thus, normalising  $P$  with respect to  $V$  and  $L$  is achieved by Equation 2.29.

$$\begin{aligned} P'(V, L) &= P \times U^2 \times \frac{S^2}{U} \\ &= P \times S^2 \times U = \text{Power After } VL \text{ Scaling} \end{aligned} \quad (2.29)$$

Normalising  $P$  with respect to operating frequency  $f$  is also difficult, and this thesis presents three options and adopts the third to generate the results in Chapters 3 and 4.

**Option 1** According to Intel, each generation  $G$  of  $L$  scaling reduces the feature size by approximately 0.7 and increases the transistor speed 1.5 times. For example, scaling from 130nm to 90nm means the maximum operating frequency can increase by a factor of 1.5 all other factors being equal [108]. Using the Intel approximation  $S \approx 0.7^G$ , the number of generations  $G$  when scaling from  $L$  to  $L'$  is determined

<sup>7</sup>This is the gate capacitance of the transistor and ignores other gate and interconnect parasitics although most these other components scale similarly.

by Equation 2 30

$$G = \begin{cases} \frac{\ln(\frac{1}{S})}{\ln(0.7)} & S > 1 \\ \frac{\ln(S)}{\ln(0.7)} & S \leq 1 \end{cases} \quad (2.30)$$

Therefore, when normalising from  $f_{max}$  to  $f'_{max}$ ,  $f_{max}$  is multiplied by  $1.5G$  if  $S < 1$  (scaling to a smaller and faster technology), or it is divided by  $1.5G$  if  $S > 1$  (scaling to a larger and slower technology)

To a first order approximation, the maximum operating frequency also depends linearly on  $V$  (see [31] and subsequent discussion in Section 2.3.3). Hence  $f_{max}$  should be scaled by  $U$  when normalising to  $f'_{max}$ . Using the Intel approximation, normalising the maximum operating frequency with respect to  $V$  and  $L$  is accomplished by Equation 2.31

$$f'_{max} = \begin{cases} f_{max} \times \frac{U}{1.5 \times G} & S > 1 \\ f_{max} \times U \times 1.5 \times G & S \leq 1 \end{cases} \quad (2.31)$$

Using Equation 2.31 and Equation 2.29, normalising  $P$  with respect to  $V, L$  and  $f_{max}$  is achieved by Equation 2.32

$$\begin{aligned} P'(V, L, f_{max}) &= \begin{cases} P \times S^2 \times U \times \frac{U}{1.5 \times G} & S > 1 \\ P \times S^2 \times U \times U \times 1.5 \times G & S \leq 1 \end{cases} \\ &= \text{Power After } VLf \text{ Scaling} \end{aligned} \quad (2.32)$$

**Option 2** Another approach to scaling  $f_{max}$  is to consider Equation 2.33, which is an attempt to express the maximum operating frequency of a device in terms of its physical parameters, where  $\mu$  is the carrier mobility in the channel and  $C_g$  is the gate capacitance (see Equation 2.27 where it is equal to  $C_L$ ) [85]

$$f_{max} = \frac{W}{L} \times \frac{\mu \epsilon_0 V}{C_g H} \quad (2.33)$$

Applying the previously outlined scaling factors to  $W, L, H$  and  $V$ , it follows that  $f'_{max}$  can also be expressed by Equation 2.34

$$f'_{max} = f_{max} \times \frac{U}{S^2} \quad (2.34)$$

Comparing Equation 2.34 with Equation 2.31, it is clear that both scale  $f_{max}$  by  $U$ . Given that each generation corresponds to  $S \approx 0.7$ , the  $1/S^2$  term in Equation 2.34 implies that each generation scale down ( $S < 1$ ) represents an approximate speed-up of 2. In contrast, Equation 2.31 assumes that each generation offers a speed up of 1.5. Hence Equation 2.34 represents more aggressive scaling of  $f_{max}$ . Using Equation 2.34 and Equation 2.29, normalising  $P$  with respect to  $V, L$  and  $f_{max}$  is achieved by Equation 2.35

$$\begin{aligned} P'(V, L, f_{max}) &= P \times S^2 \times U \times \frac{U}{S^2} \\ &= P \times U^2 = \text{Power After } VLf \text{ Scaling} \end{aligned} \quad (2.35)$$

Interestingly, Equation 2.35 is independent of  $S$

**Option 3** From a low-power design perspective, a circuit is probably not going to be operating at its maximum possible frequency  $f_{max}$ . Indeed it is likely that its operating frequency  $f \ll f_{max}$ , and  $f$  is as low as possible. This is because by Equation 2.25, a low  $f$  will result in lower dynamic power dissipation. The limit on how low  $f$  can be is design dependent and is constrained by any throughput requirements that must be met and the clock cycle latency of the circuit. Hence, in this case it is likely that  $f' = f$  when scaling from  $L$  to  $L'$ . This assumes that there is no negative slack for the circuit in technology  $L'$  when operating at  $f'$ , and this is only likely to be a concern if  $L'$  is a really old technology. So if  $f' = f$ , then normalising  $P$  with respect to  $V, L$  and  $f$  is achieved by Equation 2.36.

$$\begin{aligned} P'(V, L, f) &= P'(V, L) \\ &= P \times S^2 \times U = \text{Power After } V, L, f \text{ Scaling} \end{aligned} \quad (2.36)$$

Given that the SA-DCT and SA-IDCT architectures proposed in this thesis have been designed with low power as a primary design constraint, they operate at  $f \ll f_{max}$  (for  $L = 90\text{nm}$ ). Hence Equation 2.36 is used in this thesis for power normalisation across technologies when benchmarking against prior art.

With an expression for the normalised power consumption established by Equation 2.36, the normalised energy  $E'$  consumed by the proposed design with respect to the reference technology is expressed by Equation 2.37, and the normalised Energy-Delay Product is given by Equation 2.38.

$$\begin{aligned} E'(V, L, f) &= P'(V, L, f) \times D \\ &= P'(V, L, f) \times \frac{1}{f_{scale}} \times C \end{aligned} \quad (2.37)$$

$$EDP'(V, L, f) = P'(V, L, f) \times D^2 \quad (2.38)$$

This section has presented a set of formulae that attempt to normalise the power and energy properties of circuits when benchmarking them against each other. In particular, Equations 2.29, 2.36, 2.37 and 2.38 are used in Chapters 3 and 4 when benchmarking the SA-DCT/IDCT architectures proposed in this thesis against prior art.

### 2.3.2.6 Area and Speed Benchmarking

Although power and energy are vitally important benchmarking metrics for circuits that are implemented on mobile devices, the traditional metrics of area and speed cannot be ignored. The ideal goal for any circuit is always “smaller, faster and lower power”. In the previous section it has been explained that comparing designs in terms of their operating frequency does not always make sense, particularly if the designs are implemented in different technologies and the goal is low power operation. A more sensible metric when benchmarking two designs for the same task in terms of speed is the number of clock cycles required to compute that task, since this is independent of implementation technology.

When benchmarking SA-DCT implementations, Chen et. al. propose the use of a metric referred to as the *product of unit area and computation cycle* (PUAC) [109]. The PUAC is obtained by multiplying the normalised unit area (in units of equivalent ripple carry adders) with the number of computational cycles required for a 1D 8-point DCT. The aim of the PUAC is to combine the latency and area of a circuit into a single metric, where a lower PUAC represents a better implementation. However, the

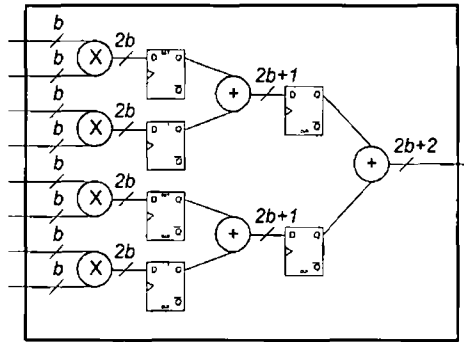


Figure 2.26 Simple Multiply-Accumulate Circuit

PUAC results presented in [109] only consider the number of adders and multipliers when deriving area, and do not consider the bitwidths of these arithmetic units

To illustrate the flaw with the PUAC metric, consider the simple multiply-accumulate circuit in Figure 2.26, where the input pins have bitwidth  $b$ . Consider two circuits  $X$  and  $Y$  that have the same architecture as shown in Figure 2.26, but the input bitwidth of circuit  $X$  is  $b_X = 16$ , and the input bitwidth of circuit  $Y$  is  $b_Y = 32$ . Using the PUAC metric according to [109], both  $X$  and  $Y$  have the same PUAC since they both have the same number of adders, multipliers and clock cycle latency. However,  $Y$  is twice as big as  $X$ ! Also, if either  $X$  or  $Y$  had additional non-arithmetic decode logic (e.g. some Boolean logic network), the PUAC would not take this extra circuit area into account.

We propose that a fairer metric when comparing designs would be the *product of gate count and computation cycle* (PGCC), since the gate count considers all the logic in the design including the arithmetic units. For the simple example described earlier, the PGCC of circuit  $Y$  would be twice as big as the PGCC for circuit  $X$ . When benchmarking the SA-DCT/IDCT architectures proposed in this thesis, the PGCC metric is used along with the power and energy metrics outlined in Section 2.3.2.5.

### 2.3.3 Low Power Design Techniques

With the causes of power consumption in CMOS circuitry established, this section aims to survey the power saving techniques commonly employed by designers. For maximal effectiveness, power considerations must be integrated into the design flow at all abstraction levels from concept to layout. Electronic Design Automation (EDA) companies have foreseen this need [14] and there are many new tools and libraries coming on the market that recognise power alongside area and timing as a high priority design constraint.

#### 2.3.3.1 Battery Technology

Perhaps the most obvious target for improving battery life is by improving the battery technology itself. However, due to slow incremental improvements in battery technology relative to the pace of Moore's law, hardware engineers must focus on energy-efficient circuit design to alleviate the exponential increase in power density of modern integrated circuits [34]. Nevertheless, research into improved longer-lasting physically lighter energy sources is an important complementary field. The most common secondary cells in use today are Sealed Nickel-Cadmium (NiCd), Nickel-Metal Hydride (NiMH) and Lithium-Ion.

(Li-ion) and significant research has been invested in these technologies due to the explosion of cellular phones, portable computers, camcorders and entertainment devices [110]. Of the three, Li-ion is the most advanced (in terms of lifetime and weight) but is also the most expensive. It is currently dominant in the laptop computer market and the technology is currently maturing and is permeating the mobile consumer electronics market [111]. A proposed low cost alternative to Li-ion is Lithium polymer (Li-polymer) [110, 112].

Researchers have been attempting to develop a revolutionary power source solution that could complement work in low power electronics, and the most promising solution is the fuel cell [113]. A fuel cell is a device that generates energy using electrochemical reactions. There is enormous interest in fuel cells since they deliver cheap, clean energy [114]. The fuels being looked at most seriously for portable fuel cells are hydrogen, metal hydrides, ethanol and methanol, with methanol appearing to be the most promising option [113]. One issue with fuel cells is making them powerful enough at the small sizes necessary for mobile platforms. Research in the field is ongoing, and Toshiba appear to be the leading company in the field [113, 115].

### 2.3.3.2 System Level Techniques

It is generally accepted that most power savings are achieved at the higher levels of design abstraction since there exists greater degrees of design freedom as illustrated in Fig. 1.2 [83, 116]. The largest savings are possible prior to design synthesis, therefore it is important to consider power consumption at an early stage in a design. It is estimated that at the system and algorithm design phases, optimisations can yield a 10–20× improvement in terms of power consumption as compared to only a 10–20% possible improvement with post-synthesis stage (circuit/technology level) techniques [117].

**Hardware / Software Partitioning** When implementing a system specification, partitioning the functionality between hardware and software is an important step. Hardware accelerators for computational hot spots (e.g. motion estimation and DCT in an MPEG-4 codec) are generally more power efficient compared with software but are less flexible and require extra logic on the chip [21, 117]. Accelerating demanding algorithms in hardware will increase throughput, since the processor is relieved of these tasks, but also save power if designed correctly. The SA-DCT/IDCT architectures proposed in this thesis achieve precisely this – they are low power hardware accelerators. Designers have great control over the architectures of the hardware accelerators and so can design them with energy-efficiency in mind. There is less design freedom for a general-purpose processor since by definition they must cater for numerous processing tasks. On the software side, the latest trend in microprocessor design is the shift from high frequency single core systems to multiple parallel cores operating at lower frequencies [118, 119].

**Board Level Layout and I/O** The initial high-level layout of the chips and interconnects on a board is very important in terms of power consumption. A system designer should aim to minimise the physical length of interconnects that will have the heaviest switching load. Careful layout of chips (processors, memory, I/O, custom peripherals etc.) on a board can potentially save a lot of needless power consumption. It is estimated that one third of the power consumed by a single chip is at the I/O ports [116]. Therefore for board designs containing discrete custom peripherals (ASIC or FPGA), the amount and position of I/O pins must be carefully traded off against speed and area requirements. Data buses usually

have heavy loads and long interconnects between cores on a board. It may not be possible to reduce the number of I/O lines but effort can still be made to reduce the switching activity on the bus. Some common techniques include one-hot coding, Gray coding and bus-inversion coding [84, 116].

**Memory Subsystems** The memory bandwidth available for a given application is another important design parameter for hardware designers. There are two components that need consideration – firstly there is an off-chip to on-chip memory bandwidth, and secondly an on-chip memory to processing data-path bandwidth. The optimal partitioning between on-chip and off-chip memories depends on the application and the design constraints. Both have to be optimised in order to achieve the appropriate speed, area and power efficiency. In general, the same source video data needs to be accessed and processed by different video tools (e.g. motion estimation and DCT) so it makes sense to read the current source data once from an off-chip memory and store it in an on-chip memory where the various tools can access the data faster and with less energy. Unfortunately, embedded/on-chip memory is usually more expensive so again trade-offs must be made. However, the power requirements of modern wireless hardware heavily favour the energy-efficiency of embedded memory. Another complicating factor for memory design is the likely usage of a cache architecture to minimise the probability of expensive off-chip accesses, both in terms of power and speed [120]. Other techniques employed to reduce switching on memory data and address lines include memory banking, selective line activation [88]. The aim of techniques like these is to localise the amount of memory circuitry activated for each data access. Energy is saved if irrelevant areas of the memory are not activated.

When choosing the memory technology for a low power application, a designer must consider both the dynamic and static power consumption of the various alternatives. In terms of dynamic power consumption, a Static RAM (SRAM) cell consumes more power than a Dynamic RAM (DRAM) cell due to its latching action and inherent size [121]. Traditionally, DRAM was said to be at a disadvantage in terms of leakage power due to its need for refresh circuitry but as process technologies shrink below 130nm, SRAM cells become very susceptible to leakage currents [121]. In fact, modern techniques have reduced the refresh current needed for a DRAM cell significantly lower than the leakage currents associated with an SRAM cell. However, DRAM is relatively slower than SRAM so again speed must be traded for power depending on the overall system specifications.

**Dynamic Power Management (DPM)** As mentioned, a system can be thought of as a collection of functional components (processor, memory, I/O, dedicated hardware accelerators etc.) which themselves are composed of various functional modules. In a power manageable system all of these components have one active state in which it carries out its task and one or more inactive low power states. The components may self-manage the power state transitions or may be controlled externally. A power manager observes the component workload and controls its power state accordingly, where the power state corresponds to a particular operating frequency and voltage. Such strategies are commonly referred to as Dynamic Voltage/Frequency Scaling (DVS/DFS). In this scenario, components can be put in a low power state or shutdown to save power when not needed. However, there are many complications regarding the implementation of a DPM system [122]. For example, components should only be shut down if the shutdown period is long enough to amortise the cost of turning off the component. This means that the DPM controller needs to be able to predict with a certain degree of accuracy the fluctuations in a



components workload. Even though the power controller is in itself a complex piece of circuitry, it has the potential to significantly extend the battery life of a mobile device by ensuring that needless power is not wasted by idle components.

### 2.3.3.3 Algorithmic Level Techniques

Assuming that the system has been partitioned in an energy-efficient manner and certain algorithms have been earmarked for hardware acceleration, the important question then becomes how a designer ensures that the hardware accelerator for a particular algorithm is indeed low power. Probably the simplest yet most effective technique is to reduce the number/complexity of basic operations that the algorithm requires. If the operations can be simplified, the number of primitive steps required decreases and consequently the energy and delay for the operation will decrease (lower EDP). If there is less processing steps to execute, the design may possibly operate at a lower voltage and/or frequency to maintain the same throughput. The hardware complexity will also reduce meaning a reduction in switching capacitance. Such properties have very positive effects on all components of power consumption as listed in Section 2.3.1. Simply by careful reformulation of the algorithm, there exists scope for large power savings. The particular optimisations employed at this level are very application specific. For example, a hardware implementation of a Fast Fourier Transform (FFT) algorithm would certainly be more energy-efficient than a direct implementation of the DFT equation itself. The same is true for the SA-DCT/IDCT, and the approach taken by the author is discussed in Chapters 3 and 4.

### 2.3.3.4 Register Transfer Level (RTL) and Logic Level Techniques

Once the optimised algorithm has been formulated, the next stage of design generally involves coding the architecture at the register transfer level (RTL) using a Hardware Description Language (HDL) such as Verilog, VHDL or SystemC. There are numerous techniques that can be employed to reduce power consumption at this level and the most common are outlined in this section.

**Parallelism and Pipelining** Parallelism and pipelining are techniques that allow the designer to reduce the operating frequency and voltage of a design while maintaining throughput. If an architecture without pipelining takes  $n$  clock cycles to compute its final results for each set of input samples and there are  $m$  sets of input samples, it will take  $n \times m$  clock cycles to compute all  $m$  inputs. The latency in this case is the product of  $n \times m \times T$  where  $T$  is the clock period. However, by enhancing the architecture with added extra pipeline control circuitry, it may take as little as  $n + (m - 1)$  cycles. This will decrease the latency of the design to  $(n + (m - 1)) \times T$ . This concept is illustrated more clearly in Figure 2.27 where  $n = 4$  and  $m = 3$ . However, if power savings are important, the clock period could be increased in the pipelined design to match the latency of the non-pipelined design. In this case, the throughput has been maintained but the architecture is operating at a lower frequency thus lowering the power consumption. Lowering the operating frequency may also afford the possibility of operating at a lower voltage thus saving even more power.

The same concept applies to incorporating parallelism in a design. At one extreme is there are  $m$  sets of inputs there could be  $m$  parallel processing units and all  $m$  inputs could be processed in only  $n$  clock cycles, thus saving power since again frequency and voltage may be reduced. However, there

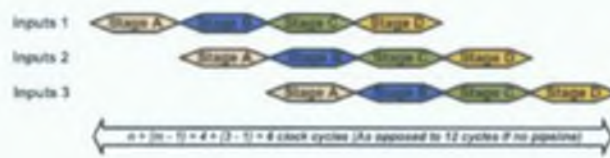


Figure 2.27: An Example of a Pipelined System

is an obvious drawback since the greater the extent of parallelism the greater the effective switched capacitance. Parallelism and/or pipelining can only save power if the correct balance between area and power is used [32], and both techniques are used to a certain extent in the design of the SA-DCT/IDCT architectures proposed in this thesis in Chapters 3 and 4.

As an illustrative example, consider the simple adder comparator circuit shown in Figure 2.28 [83] (the following discussion is a summarised version of the example in [83]). In this circuit, inputs  $A$  and  $B$  are added and the result is compared to  $C$ . If the circuit is synthesised using  $2.0\mu\text{m}$  technology the worst-case delay through the adder, comparator and latch is approximately  $25\text{ns}$  at a voltage of  $5\text{V}$ . Therefore (ignoring metastability effects) if the clock period  $T$  needs to be  $25\text{ns}$  to match throughput specifications, the voltage cannot be reduced any lower than  $5\text{V}$  since no extra delay can be tolerated. If this is taken as the reference circuit, the power consumed by this circuit is  $P_{ref} = C_{ref}V_{ref}^2f_{ref}$ . Figures 2.29 and 2.30 show parallel and pipelined implementations respectively of the previous circuit. In the parallel implementation, two identical adder-comparator datapaths are used, allowing each unit to work at half the original rate (half the clock frequency) to maintain a throughput of  $25\text{ns}$ . Since the speed requirements for each unit has now decreased to  $50\text{ns}$ , the voltage may be dropped from  $5\text{V}$  to  $2.9\text{V}$ . However, the datapath capacitance has increased by a factor of 2 but the frequency has correspondingly decreased by a factor of 2. In fact, the total effective capacitance has increased by a factor of 2.15 owing to the extra interconnect routing. Comparing to the reference power consumption  $P_{ref}$  the parallel consumption evaluates to  $P_{par} = C_{par}V_{par}^2f_{par} = 2.15C_{ref}(0.58V_{ref}^2)\frac{f_{ref}}{2} \approx 0.36P_{ref}$ . So even though the area has more than doubled, the power consumption has been significantly reduced. With the pipelined architecture, the additional pipeline latch between the adder and the comparator reduces the critical path to be  $\max(T_{adder}, T_{comparator})$  allowing the adder and comparator to operate at a lower rate. If the two delays are equal (both  $12.5\text{ns}$ ), the voltage can again be reduced to  $2.9\text{V}$  with no loss of throughput if the clock frequency is maintained (assuming the pipeline is never stalled). There is a much lower area overhead incurred with this technique, since only two extra latches are needed, and the total effective capacitance increases by approximately a factor of 1.15. The pipelined power consumption is  $P_{pipe} = C_{pipe}V_{pipe}^2f_{pipe} = 1.15C_{ref}(0.58V_{ref}^2)f_{ref} \approx 0.39P_{ref}$ , which is comparable to the parallel savings with much less overhead. One obvious extension is to combine both parallelism and pipelining. An architecture incorporating both reduces the critical path by a factor of 4, and as such the voltage can be dropped until the delay increases by a factor of 4. The power consumption in this case is  $P_{both} = C_{both}V_{both}^2f_{both} = 2.5C_{ref}(0.4V_{ref}^2)\frac{f_{ref}}{4} \approx 0.2P_{ref}$ . The power consumption in this case has been reduced to only 20% of the original circuit.

The issue of how far the parallelism/pipelining approach can be taken warrants further discussion. Obviously there is an area overhead involved and it may not always be viable to increase the area to reach the optimum power levels. The optimal amount of parallelism and pipelining in terms of power will

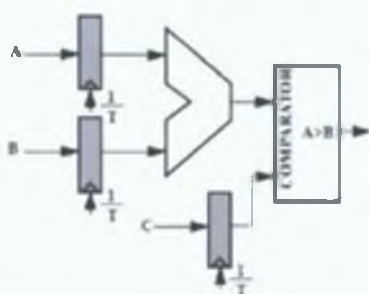


Figure 2.28: Sample Comparator Circuit

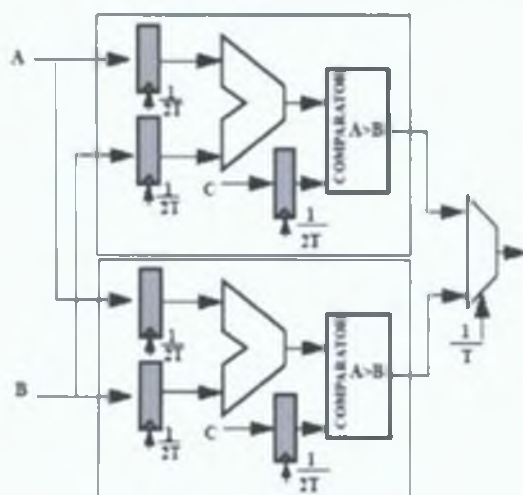


Figure 2.29: Sample Parallel Comparator

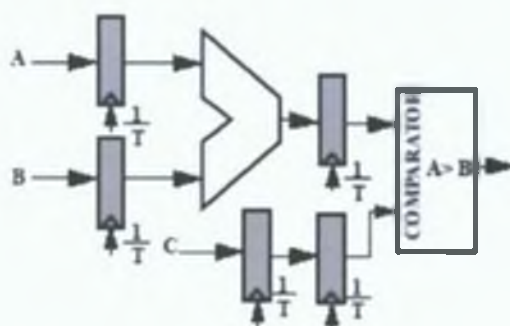


Figure 2.30: Sample Pipelined Comparator

depend on the specific circuit and the technology employed (process line widths, transistor thresholds, noise margins etc.). Usually the overhead in increased parallelism to compensate for the reduced logic speed will begin to dominate, increasing the power faster than the voltage reduction decreases it [83]. The authors of [34] when dealing with the adder-comparator circuit found that using  $2.0\mu\text{m}$  technology with device thresholds at approximately 1V, the optimal supply voltage yielding the lowest power figures is 1.5V. Using  $0.8\mu\text{m}$  technology, they found the optimal voltage to be approximately 1V yielding even greater power savings. Using a number of examples they found that the optimum voltage occurs between 1 – 1.5V and they speculate that scaling the device threshold voltages would result in a scaling of supply voltages into the sub 1V range. However, it should be noted that scaling the threshold voltages too aggressively can result in unwanted sub-threshold conduction effects as outlined in Section 2.3.1.

**Synchronous Techniques and Clock Gating** The clock tree can consume between 30% and 50% of the power in a digital system since by definition the clock toggles every cycle [32, 123]. Clock gating is a commonly employed technique used to turn off clock tree branches to latches or flip-flops when they are not used. Until recently, developers considered clock gating to be a poor design practice because clock tree gates can exacerbate clock skew. Conventionally, placement and routing is optimised for timing, with little or no consideration for power or signal integrity. This is changing with the newer clock tree

synthesis approaches [123] and tools like Synopsys' Power Compiler now has features such as automatic clock gating and power aware placement [107]. Since by their adaptive nature, the SA-DCT/IDCT have a variable computational load, clock gating is used to gate off logic in the SA-DCT/IDCT architectures proposed in Chapters 3 and 4 according to the dynamic nature of the input shape information.

Other techniques related to clock tree optimisation include half-frequency and half-swing clocks [32]. A half-frequency clock triggers logic on both the rising and the falling edge, thus enabling it to run at half the frequency and hence reduce switching by 50%. One drawback of this technique is the increased area requirements of double edge-triggered latches and flip-flops. Another issue is the complexity involved in interfacing logic that is double edge-triggered with logic that is single edge-triggered. A half swing clock swings only half of the supply voltage on each edge, which again saves power. However, this increases latch and flip-flop design requirements and is difficult to apply in systems with a highly scaled supply voltage.

The concept of asynchronous logic is a much-debated topic in the design community and its supporters claim that a system without a clock can save considerable power. However, asynchronous logic requires extra handshaking signalling and is in general difficult to test. The fact that most EDA tool vendors warn against the use of asynchronous logic is a strong sign that the synthesis results may be unpredictable.

**Glitching/Switching Minimisation** Circuits can sometimes exhibit spurious transitions due to finite propagation delays through logic blocks. This is especially true if logic depths are unequal between registers. To minimise these glitches it is important to balance all signal paths as much as possible. For example, consider the two implementations for adding four numbers shown in Figure 2.31. Assume all inputs arrive at the same time. If there is a propagation delay through the first adder in the cascaded case, the second adder is computing with the new  $C$  input and the old value for  $A + B$ . When the new value of  $A + B$  finally propagates, the sum is recomputed. The same idea is true for the third adder, except this time the addition is computed three times per cycle. The capacitive load switched per cycle is greater for the cascaded case compared to the tree implementation due to the spurious transitions. The capacitive load for the registers holding the primary inputs is also distributed more evenly in the tree circuit. Sometimes, however, the primary inputs may have different arrival times and depending on the specific signal timing statistics, a hybrid cascaded tree implementation may be optimal. Such decisions can only be taken with certainty after synthesis and placement details are available. As well as balancing logic paths, other common techniques to reduce general node switching (as opposed to glitching in particular) include conditional computation [84] and clever finite state machine encodings (Gray, One-Hot etc.). Both adder tree balancing and Gray coding of finite state machines are used in the SA-DCT/IDCT architectures proposed in this thesis.

**Reduced Precision Computation** The previous techniques that have been mentioned aim at saving energy without loss of performance. Speed may be sacrificed, but the precision and accuracy of the output produced by the low power hardware accelerator is maintained. However there are cases when degradation in quality may be acceptable if it can be traded for increased battery life. Signal processing applications generally require many additions and multiplications and energy can be saved if the bit widths of some of these computations is reduced, or if the results of a computation are rounded to a

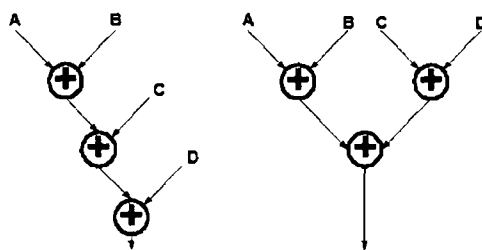


Figure 2.31 Reducing Glitching by Balancing Signal Paths

smaller number of bits. A detailed examination of the algorithm to be implemented may hint at certain computations that may afford less accurate or precise computation without a significant degradation in quality. For example, an  $8 \times 8$  DCT computation produces 64 frequency coefficients all of which require the same number of additions and multiplications to compute. However, as outlined in Section 2.2.2, the lower frequency coefficients are most important if the DCT is being applied to natural video data. Therefore it may be energy-efficient to compute the high frequency coefficients less precisely. It may also be possible to adapt the precision of the computations based on the input signal statistics and only dissipate the minimal power necessary for each individual computation. This idea has been explored for a hardware accelerator implementing the forward  $8 \times 8$  DCT [124]. The SA-DCT/IDCT architectures proposed in this thesis use fixed-point arithmetic using the lowest possible bitwidth that maintains the quality required by the MPEG-4 standard.

### 2.3.3.5 Circuit / Technology Level Techniques

The energy saving techniques discussed up to this point have been relatively independent of the technology used to implement hardware accelerators. The focus has been on optimising the system architecture, the algorithm to be implemented and the local architecture of the hardware accelerator. These techniques are applied before the synthesis process enhanced by sophisticated EDA tools during logic synthesis. Applying low power methodologies effectively at the circuit, transistor or silicon levels requires a detailed understanding of the technology being used and the associated semiconductor physics. The main technique used is to scale down the voltage to save power, but this can also cause subsequent adverse behaviour depending on the technology employed. Since these techniques are not exploited directly by the SA-DCT/IDCT architectures proposed in this thesis, this section aims to only briefly outline the most commonly used techniques at these levels.

**Logic Circuit Options** There are numerous options available in choosing the basic circuit approach and topology for implementing various logic functions. The choice between static versus dynamic implementations and pass-gate versus conventional CMOS are two such examples [83]. In terms of low-power properties, dynamic logic has the edge over static logic since it has lower glitch switching, eliminates short-circuit dissipation and reduced parasitic node capacitances. Pass-gate logic is attractive, as fewer transistors are required to implement logic functions such as XOR [34].

**Voltage Scaling and its Implications** As noted earlier, ideally the most effective method to reduce power consumption is to reduce the supply voltage since the energy per switching transition is quadratically dependent on this term. Unfortunately one adverse effect of reducing  $V_{dd}$  is that the speed of the logic gates is reduced. A first order approximation of the relationship between voltage and delay is given in Equation 2.39

$$T = \frac{C_L}{\frac{\mu C_{ox}}{2} \frac{W}{L}} \frac{V_{dd}}{(V_{dd} - V_t)^2} \quad (2.39)$$

Since the objective is to reduce power consumption while keeping the latency of the overall system fixed, compensation for these increased delays is required. At higher levels of abstraction this is achieved by parallelism and pipelining as outlined earlier. At the lower levels of abstraction, two primary techniques based on Equation 2.39 involve resizing the transistors (the  $W/L$  ratio – see Figure 2.25) and scaling the device threshold voltages. The latter technique has serious implications in deep sub-micron designs and is dealt with separately in the next section.

With regard to transistor sizing, it can be seen from Equation 2.39 that increasing the  $W/L$  ratio will decrease the delay but it will also increase the gate capacitance and increase the switching power consumption. In circuits where the parasitic capacitances dominate over the gate capacitances, the trade-off is not so distinct and there exists an optimal  $W/L$  ratio that actually minimises the switching power consumption with a certain decrease in gate delay [83, 96]. When the parasitic capacitance is negligible, the minimum sized devices should be used. Adjusting the transistor sizes and the supply voltage optimally to balance speed and power is very much circuit dependent.

One way to improve both power and delay is to use a smaller technology where all voltages and linear dimensions are scaled down [96]. The scaling formulae are presented in Section 2.3.2.5 in the context of circuit benchmarking. The difficulty with ideal scaling is the requirement for the threshold voltage to scale along with the supply voltage as well as reliability issues. In addition, with the latest technologies such as 90nm the resistances associated with interconnects are no longer negligible. This may cause voltage drop problems and electromigration effects when routing the power supply rails and the clock tree [125].

In general, reducing the supply voltage will reduce the power consumption but increase the device delay. Techniques such as transistor sizing, technology scaling and threshold voltage scaling are commonly employed to compensate for the increase in delay. However, with the most modern technologies these techniques cause problems with static power dissipation which if not tackled threaten the viability of future feature size miniaturisation.

### 2.3.3.6 Tackling Static Power Dissipation

As technologies are scaled to make devices smaller and faster, the supply voltages are also scaled. The supply voltages are scaled to keep the electric fields the same across generations, since many of the device reliability parameters are a function of the fields that exist within the device. This approach is called constant field scaling [89]. Scaling the supply voltage has the added benefit of reducing the dynamic power dissipation as outlined in the previous section. To achieve a device delay improvement as it is scaled, the device drive current must be kept constant [91]. The drive current is a complicated function of many parameters including  $(V_{dd} - V_t)$ . The quantity  $(V_{dd} - V_t)$  is referred to as the gate overdrive, where  $V_{dd}$  is the supply voltage and  $V_t$  is the device threshold voltage. As  $V_{dd}$  is scaled,  $V_t$

must be scaled accordingly to keep the drive current constant. Unfortunately as Equation 2.18 illustrates, the sub-threshold leakage current is exponentially dependent on  $-V_t$ . This is the reason why static power dissipation is increasing as technologies shrink below 90nm.

Research into the static power issue is widespread and one solution is to use multiple voltage domains [125]. In this approach, any performance critical functions can be located in a higher voltage domain while non-critical functions can be allocated to a lower voltage domain. Many vendors also offer multiple  $V_t$  libraries where each type of gate is available in different forms: low-threshold devices that switch quickly but have high leakage, and high-threshold devices that switch more slowly but have low leakage. Again, low  $V_t$  devices could be used on critical paths only. Another technique is to selectively power down low  $V_t$  chip areas when they are not needed [89]. This technique is called power gating, and is based on the same concept as dynamic power management of subsystems, except at the transistor level. For this technique to work effectively, accurate prediction and speculation circuitry is needed which obviously has an associated area overhead. There is also widespread research at the silicon and physical level, and some common themes appear to be strained silicon and high-k dielectrics [91]. Strained silicon is a processing technique that essentially stretches the silicon molecules further apart, allowing electrons to flow through transistors up to 70% faster. This will reduce the need to scale  $V_t$  as aggressively to maintain the gate drive current. "High-k" stands for high dielectric constant, which is a measure of how much charge a material can hold. By moving to a new high-k material, the drive current can be maintained at the level it could have achieved with silicon dioxide dielectrics – and overcome the leakage. Intel aims to introduce a high-k dielectric with its 45nm technology before 2010.

### 2.3.3.7 Recurring Themes in Low-Power Design

Section 2.3.1 has provided a detailed explanation of the sources of power consumption in modern CMOS devices. This discussion has shown that the primary controlling parameters are voltage, physical capacitance, operating frequency and node activity rate. The sub-sections thus far in Section 2.3.3 have described techniques at the different abstraction levels that aim to jointly optimise these parameters. Many of these techniques follow two common themes – trading area/performance for power and avoiding waste [37].

It has been demonstrated that decreasing the system supply voltage to save power degrades the maximum operating frequency of the system. Clearly this represents trading power for performance. In deep-submicron design where power is a primary design constraint, a designer should aim to meet rather than beat performance requirements. If performance constraints are also tight, the designer can employ parallelism which has been shown to save power as well as maintain performance. However, excessive parallelism incurs an area penalty, which is clearly important on a mobile platform where area is limited. It is clear that with modern technology, a designer must carefully trade area, performance and power according to the system constraints.

The other recurring theme of low power design involves avoiding waste. Low power techniques for avoiding waste exist at all levels of abstraction (e.g. DPM, algorithm re-formulation, clock gating, glitching elimination and power gating). These techniques almost come for free in the sense that performance and area are not significantly affected if they are employed (ignoring relatively negligible overheads). These techniques are mainly common sense ideas but can significantly improve the power consumption of a system.

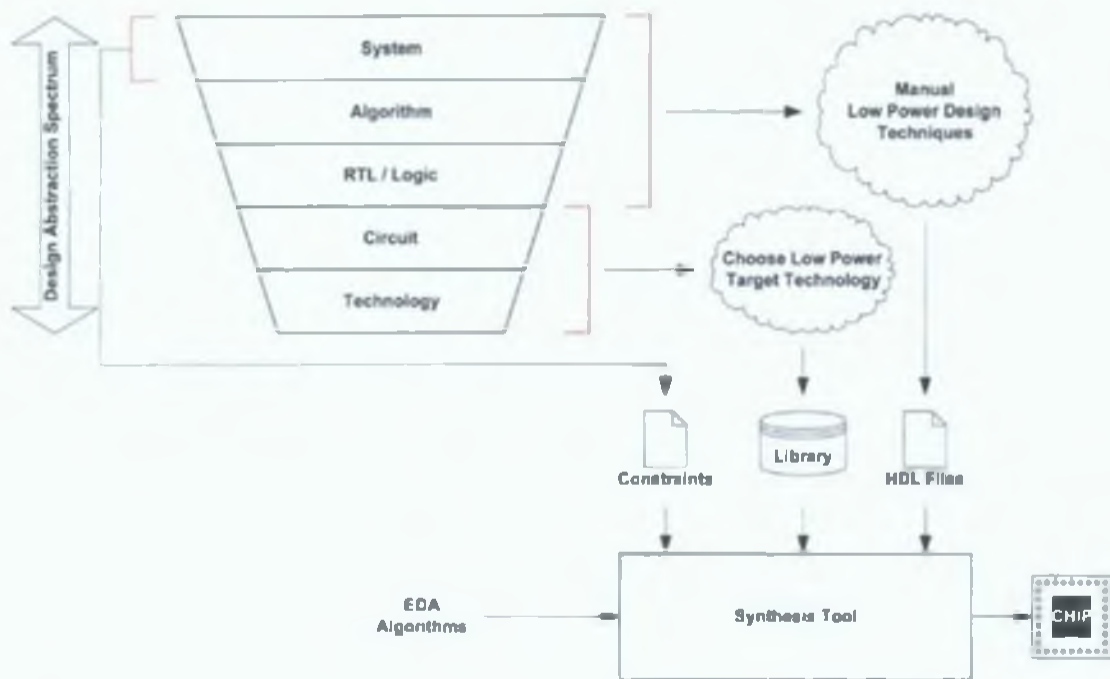


Figure 2.32: Low Power Design Flow

### 2.3.4 Electronic Design Automation (EDA)

The low power design techniques surveyed in Section 2.3.3 are generally leveraged by a digital designer before committing the optimised design to a HDL description as shown in Figure 2.32. The system, algorithmic and RTL/logic level techniques discussed in Section 2.3.3 are manually applied by the designer. For example, a software designer could include a dynamic power management scheme as part of the operating system. A hardware designer could choose reduced precision multipliers or use Gray coding of finite state machines. The hardware designer implements these ideas by describing the low power hardware architecture in a HDL like VHDL or Verilog as shown in Figure 2.32. A hardware designer indirectly applies the circuit and technology level low power techniques of Section 2.3.3 by choosing an appropriate target process technology (e.g. a low power ASIC library or a low power FPGA), which the HDL description will be mapped to by a synthesis tool. By analysing the global system itself, a set of design constraints (area, speed and power) are derived for each of the hardware modules in the system. As shown in Figure 2.32, the HDL modules, the target technology library and the design constraints are fed to a sophisticated synthesis tool that maps the HDL description to the technology library based on the constraints, and the final result (after extensive verification) is a chip mask that is ready for manufacture.

An important point about Figure 2.32 is that prior to the HDL coding, the low power techniques are applied manually by the user. The synthesis tool uses sophisticated Electronic Design Automation (EDA) algorithms to further optimise the design according to constraints and library characteristics. The optimisations achieved by synthesis tools are generally difficult to employ manually by a user, hence the automation. Indeed many of the optimisations that EDA algorithms attempt may be characterised mathematically as NP-complete problems, so in general, clever heuristic approaches are required [126,



[27] This section briefly summarises the kind of EDA algorithms used by a synthesis tool, which gives context for the EDA algorithm proposed by the author in Chapter 5 that optimises the hardware resource allocation for general CMM operations

### 2.3.4.1 Trends in EDA

In general, the synthesis process is a multi-stage process that takes a specification of the behaviour required of a system and a set of constraints and goals to be satisfied, and finds a structure that implements the behaviour while satisfying the goals and constraints [128]. The synthesis flow is an important topic in itself, and a proper treatment is beyond the scope of this thesis.

It must be noted that differing definitions of the scope of the synthesis stages exist in the literature, but they all refer to translation from behaviour to structure at some abstraction level. Behavioural synthesis translates an untimed algorithmic description (possibly C/C++/SystemC) into a cycle accurate Register Transfer Level (RTL) description (SystemC/Verilog/VHDL). RTL synthesis transforms behavioural HDL constructs (such as if-else and case statements) into a structural representation (Boolean networks for combinational logic and finite state machines for sequential logic). Logic synthesis is the fine tuning of an existing structural description based on design constraints. Layout synthesis is the place and route step where a layout mask of the final chip for manufacture is produced.

The escalating complexity of hardware designs led to the emergence of HDLs in the 1990's, since schematic capture became no longer feasible. Today, there is no established automatic design flow from the system level to the timed RTL description, although this is slowly changing. In the workplace, digital designers usually manually interpret a behavioural system specification document and directly code HDL timed RTL descriptions of the various sub-modules. As designs have become even more complex and time to market pressures increase, there is a necessity for a more integrated design flow incorporating behavioural synthesis. Behavioural synthesis and RTL synthesis can be grouped by the term "High-Level Synthesis" (HLS). The HLS process is discussed in more detail in Section 2.3.4.2.

The most recent EDA trend is the emergence of Electronic System Level (ESL) design flows – a fully integrated design flow from concept to manufacture. The perceived advantages of such a flow include

- More abstract models facilitate faster and easier design space exploration
- No manual translation necessary from behavioural software description to RTL hardware description
- Virtual prototypes for the hardware are immediately available so software development can begin in parallel as opposed to the traditional staggered design flow where a physical hardware prototype is necessary first
- Verification time decreases significantly compared to RTL simulation

ESL tools have been slow to emerge due to concerns about the quality of results compared to hand crafted RTL design. However, there is significant work ongoing in this field and recent years has seen the release of many ESL tools. Examples include Cynthesizer by Forte [129], DK Design Suite by Celoxica [130] and Catapult C by Mentor Graphics [131]. The overriding trend is the drive to more integrated design flows with designs described at more abstract levels. However, the EDA algorithm proposed in Chapter 5 is still in the HLS domain.

### 2.3.4.2 The High-Level Synthesis (HLS) Process

HLS may be thought of as an intermediate abstraction level of synthesis between pure RTL synthesis and true ESL synthesis. The process may start with an untimed or timed description of the system, depending on whether the behavioural synthesis step is included or not. Once a timed RTL description is available, the following synthesis steps take place to translate it into a structure transparently mappable to hardware gates [127, 126]

- Compilation
- High Level Transformations
- Scheduling
- Allocation
- Binding

These are the steps that underpin the “synthesis tool” black-box in Figure 2.32. The compilation step is a one-to-one transformation of the input HDL specification into a new internal representation of the behaviour, better suited to synthesis [128, 127]. No optimisation occurs in this step and the result is usually graph based (a data flow graph and a control flow graph) [127]. High Level Transformations (HLTs) aim to optimise circuit behaviour by transforming the description of the circuit without altering its functionality. Some common HLTs include dead-code elimination, common sub-expression sharing, algebraic transformations and constant propagation [126]. The goal of scheduling is to optimise the number of control steps needed to complete a function, according to design constraints. A scheduling algorithm assigns each operation into one or more control steps [128]. The allocation and binding steps carry out the selection and assignment of hardware resources for a given design. Based on the schedule, allocation determines the type and number of resources (registers, adders, gates etc.) required by the design. Binding assigns the instance of an allocated hardware resource to a given data path node [126].

The HLS steps are non-trivial and the optimisations used depend on the design constraints. Indeed the HLS process can be described as an NP-complete problem, so in general clever heuristic approaches are required [126, 127]. The design space is large and a good HLS process aims to search the design space cleverly to converge on an optimal realisation based on constraints. The complexity of the problem means that it will always remain an active area of research, especially since the optimal synthesis of complex VLSI systems is vitally important.

### 2.3.4.3 High-Level Transformations of VLSI for Constant Matrix Multiplications

Applications involving the multiplication of variable data by constant values are prevalent throughout signal processing, image processing, control and data communication [132]. Some common tasks that involve these operations are finite impulse response filters (FIRs), the discrete Fourier transform (DFT) and the discrete cosine transform (DCT). Optimisation of these kind of multiplications will significantly impact the performance of such tasks and the global system that uses them. The tasks listed as examples are instances of a more generalised problem – that of a linear transform involving a constant matrix multiplication (CMM).

The general properties of a CMM operation can be exploited by a designer to realise an efficient hardware implementation. For example if one operand of a multiplication is constant it is possible to implement the operation using only additions and/or subtractions coupled with shift operations instead of implementing a full multiplier (referred to as distributed arithmetic) [133]. The multiplication by constant problem has been widely researched and there are many optimisation algorithms in the literature, as surveyed in [134]. These algorithms may be broadly classified as high-level transformational algorithms since they transform the realisation of the CMM operation to a more optimal representation. As such, they may form part of a larger HLS tool as described in Section 2.3.4.2.

Traditionally an “optimal” realisation of a CMM circuit referred to the realisation with the lowest adder count (hence smallest silicon area). Based solely on this criterion the CMM optimisation problem itself may seem simple but in reality is very difficult due to the huge combinatorial possibilities. Most proposed approaches sacrifice optimality by using heuristics to converge on a local optimum. With the trend towards mobile computing platforms, power consumption properties have become as (if not more) important than circuit area. Indeed, there are many criteria that may be considered when building a CMM optimisation approach some of which are complementary and some of which conflict:

- Adder count (the lower the better)
- Logic depth (the smaller the better)
- Operating frequency (the faster the better)
- Power consumption (the lower the better)
- Datapath re-use (the more the better)
- Degree of parallelism (depends on constraints)
- Net fan-out (related to speed and power)
- ...

The important point here is that they all trade-off against each other – the optimal solution depends entirely on the target application design constraints. From an algorithmic point of view, the factor that most influences the relative trade-offs of the above criteria is the numeric representation of the distributed constant weights. The approach proposed in this thesis (as described in Chapter 5) uses adder count as the sole optimisation criterion but it is envisaged that more criteria be integrated in future. The proposed solution searches for an absolute optimal solution (fewest adders) without resorting to exploring the entire permutation space by using some intuitive early exit strategies. Such an algorithm can be employed to realise optimal circuitry for any CMM problem.

Some commercial approaches to automated distributed arithmetic architectures implementation include the Xilinx DA FIR [135], the eInfochips (Xilinx 3rd Party IP Partner) Parallel Distributed Arithmetic (PDA) architecture [136] and the Actel CoreFIR [137]. These approaches are based on automated look-up-table synthesis. A more sophisticated approach is the Graph Synthesis Algorithm by Xignal Technologies [42]. Such EDA algorithms underpin logic synthesis tools such as Synopsys Module Compiler that try to optimise various kinds of datapaths based on design constraints [138]. Chapter 5 presents

a detailed state-of-the-art review of work in this field to give context for the contributions made by the author's work

### **2.3.5 Conclusions**

The discussion in this section has illustrated clearly that with the ever-increasing miniaturisation trends in microelectronics power consumption has become the high priority design constraint. The sources of power consumption in a CMOS circuit have been derived and the typical approaches taken by designers to tackle the issue have been introduced. To achieve significant power gains, energy efficiency must be considered at all abstraction levels and at all stages of the design cycle. Most power savings can be made at the higher levels and the initial stages of the design where by reformulating the problem, energy is not wasted on computation that is not necessary. It is this high-level approach that has been adopted for the design of the SA-DCT and SA-IDCT cores proposed in Chapters 3 and 4, as well as the CMM optimisation algorithm proposed in Chapter 5. In particular, the main contributions of the SA-DCT/IDCT architectures proposed in this thesis are at the algorithmic and RTL/logic abstraction levels. It is envisaged that the proposed CMM optimisation algorithm could be integrated into a sophisticated EDA synthesis tool to help achieve superior results for a complex hardware resource allocation problem.

## **2.4 Summary**

This chapter has presented a comprehensive technical background to give context and motivation for the author's own work. Digital video applications are emerging as key selling points on the latest mobile handsets. Due to the large amount of data involved, a data compression scheme is required to make transmission and/or storage feasible. This chapter has described the basic enabling technologies that underpin the modern video compression schemes. A taxonomy of compression standards has been presented, with emphasis on object-based MPEG-4, which uses the SA-DCT/IDCT. It was explained how video compression schemes are very computationally demanding, and how on a mobile platform this gives rise to power consumption concerns. This chapter presented the sources of power consumption in modern CMOS technology, discussed some benchmarking metrics, as well as outlining power saving techniques for all levels of design abstraction. It is concluded that most savings are possible at the higher abstraction levels (system, algorithmic, RTL). Such "high-level" techniques include effectively partitioning the system between hardware and software, avoiding redundant computation, and hardware resource re-use. The SA-DCT/IDCT architectures of Chapters 3 and 4, as well as the CMM optimisation algorithm proposed in Chapter 5 have been designed with this high-level approach in mind.

## CHAPTER 3

# Shape Adaptive Discrete Cosine Transform Architecture

Modern image and video compression standards use transform based compression techniques to achieve highly efficient compression ratios as outlined in Chapter 2. The transformation process literally transforms the input data to an alternate representation, which is more amenable to compression than the original representation. By processing the data in the transform domain, it is easier to exploit spatial signal redundancy and thereby achieve lower bit-rates for transmission. Both representations are equivalent and the forward and inverse transform equations are used to alternate between one and the other. This is possible because the transformation itself is a lossless process and performing the inverse transformation restores the original image data.

The benefits of transform-based compression come at a cost – the transform functions are among the most computationally demanding in a video codec. Hence, there has been extensive research into efficient implementations, both software and hardware, of mathematical transforms. The stringent power and real-time design constraints imposed by modern mobile multimedia devices has seen a recent focus on efficient hardware implementations of transform functions. The most popular transform adopted by multimedia standards is the Discrete Cosine Transform (DCT) [54] for block-based codecs and its Shape Adaptive extension, the SA-DCT, for object-based codecs [20]. This chapter firstly surveys state of the art implementations for the DCT/SA-DCT proposed in the literature. Then, a novel hardware implementation of the SA-DCT is proposed that offers a good trade-off between area, speed and power consumption. This architecture is extremely attractive if implementing a mobile multimedia terminal that supports MPEG-4 object-based processing.

### 3.1 Introduction

To illustrate the additional complications associated with the SA-DCT compared to the regular  $8 \times 8$  DCT, consider Figure 3.1 which is similar to Figure 2.15 (see Section 2.2.6 for an introduction to the algorithm). In Figure 3.1, the data in the block relevant to the VO is labelled with integers to explicitly illustrate how the data is shifted and packed during the different SA-DCT steps. The additional factors that make the SA-DCT more computationally complex with respect to the  $8 \times 8$  DCT are vector shape parsing, data alignment and the need for a variable  $N$ -point 1D DCT transform. The SA-DCT is less

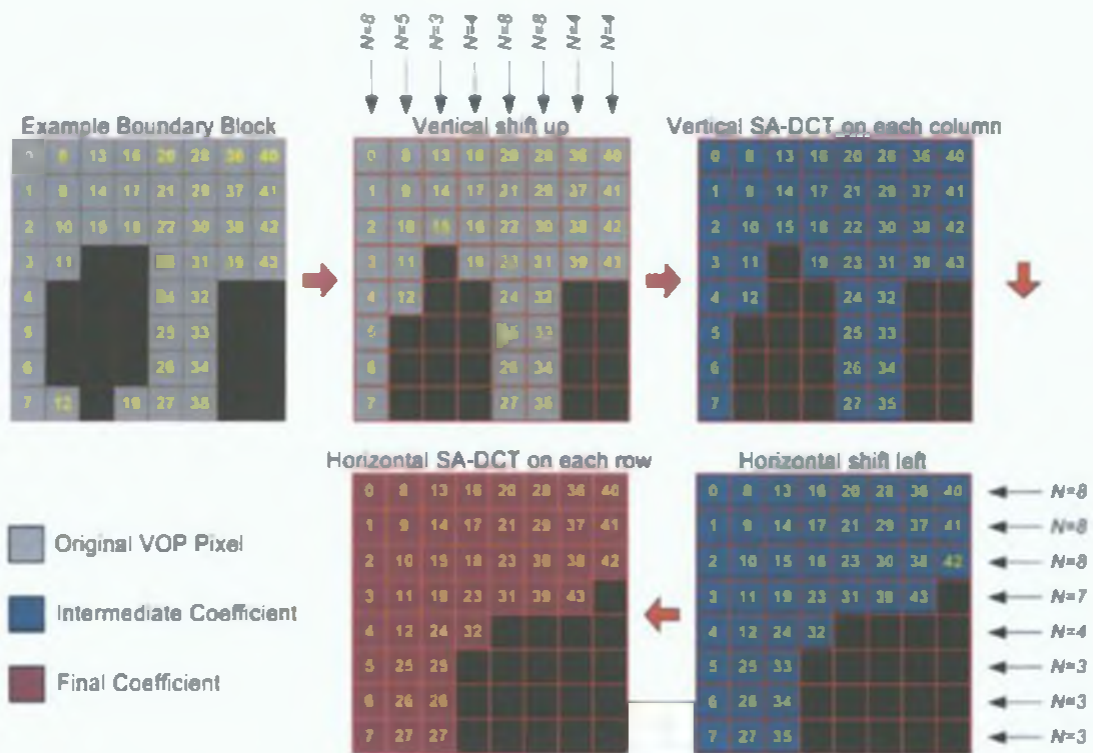


Figure 3.1: The SA-DCT Algorithm Data Addressing

regular compared to the  $8 \times 8$  block-based DCT since its processing decisions are entirely dependent on the shape information associated with each individual block. To re-cap, the SA-DCT algorithm steps are as follows (as illustrated by Figure 3.1):

- Parsing of column shape information and vertical packing of data
- Vertical  $N$ -point DCT on each column according to results of previous step
- Parsing of row shape information and horizontal packing of data
- Horizontal  $N$ -point DCT on each row according to results of previous step

### 3.1.1 Vector Shape Parsing

The  $8 \times 8$  alpha block that defines the VO region in the corresponding  $8 \times 8$  texture block must be parsed to evaluate the  $N$  values that configure the transform kernels for each of the row and column transforms. Since the shape vector of a general row or column is 8 pixels wide, there are  $2^8 = 256$  permutations of possible shape patterns for shape vectors. This step is not required for the  $8 \times 8$  DCT, but for the SA-DCT there are 16 parsings necessary (for the 8 columns and 8 rows). Sample parsed  $N$  values are shown in Figure 3.1.

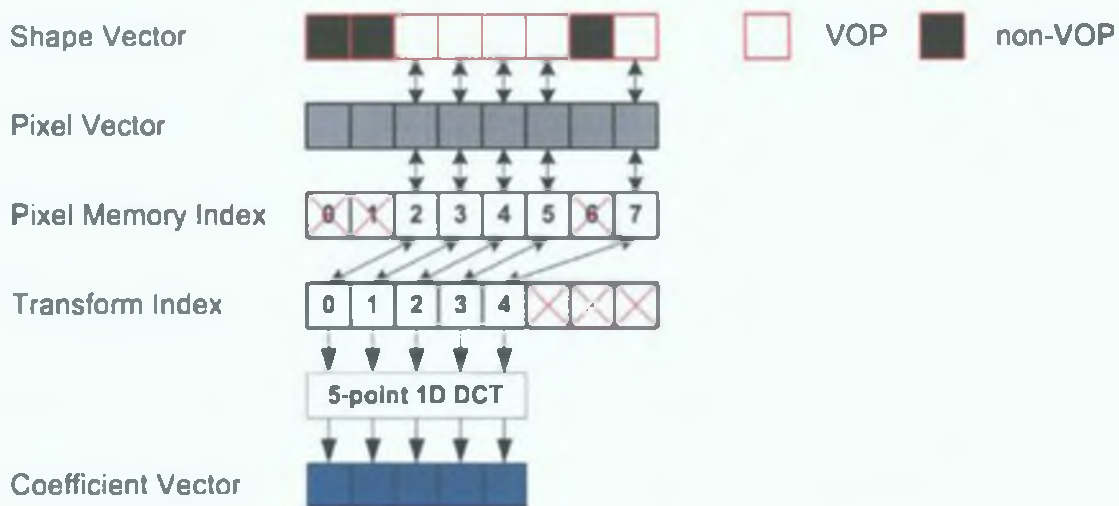


Figure 3.2: Mismatch Between Data Memory Index and Data Transform Index

### 3.1.2 Data Alignment

As well as parsing the  $N$  values for each column and row, it is necessary to identify which data is being addressed for transformation. This is challenging from an implementation point of view since vertical and horizontal packing steps are necessary prior to transformation to (as illustrated in Figure 3.1). The offset index of the data in memory may not correspond to its index for transformation as shown in Figure 3.1 and more explicitly in Figure 3.2. With the  $8 \times 8$  DCT, all data is by default aligned and no packing is required. This alignment is reversed by the SA-IDCT algorithm – this is dealt with in detail in Chapter 4.

### 3.1.3 Variable $N$ -point 1D DCT

The  $8 \times 8$  DCT requires 16 1D 8-point DCT computations if implemented using the row-column approach. Each 1D transformation has a fixed length of 8, with fixed basis functions (i.e. Equation 2.8a with  $N = 8$ ). This is amenable to hardware implementation since the data path is fixed and all parameters are constant. The SA-DCT requires up to 16 1D  $N$ -point DCT computations (i.e. Equation 2.8a with  $N \in \{0, 1, 2, \dots, 8\}$ ). The key points to note are that  $N$  may vary (depending on the particular shape pattern) and that perhaps not all 16 1D transforms are necessary (i.e. there is one less 1D transform required for each case that  $N = 0$ ). The fact that  $N$  may vary between 0 and 8 complicates hardware implementation since there are now 35 distinct basis functions, whereas the  $8 \times 8$  DCT has only 8 as shown in Figure 3.3. Note that  $N \in \{0, 1\}$  are trivial cases. For  $N = 1$  the single DCT coefficient is equal to the single input data sample (a simple data copy), and for  $N = 0$  no transform is necessary. The cosine basis elements in Figure 3.3 are derived from the  $N$ -point 1D DCT equation (i.e. Equation 2.8a).

### 3.1.4 SA-DCT Computational Requirements Summary

The advantages of MPEG-4 object-based compression, which requires the SA-DCT/IDCT, have been outlined in Chapter 2. However, compared to the block-based  $8 \times 8$  DCT, the SA-DCT is more computationally demanding since there are additional processing steps necessary. These additional steps

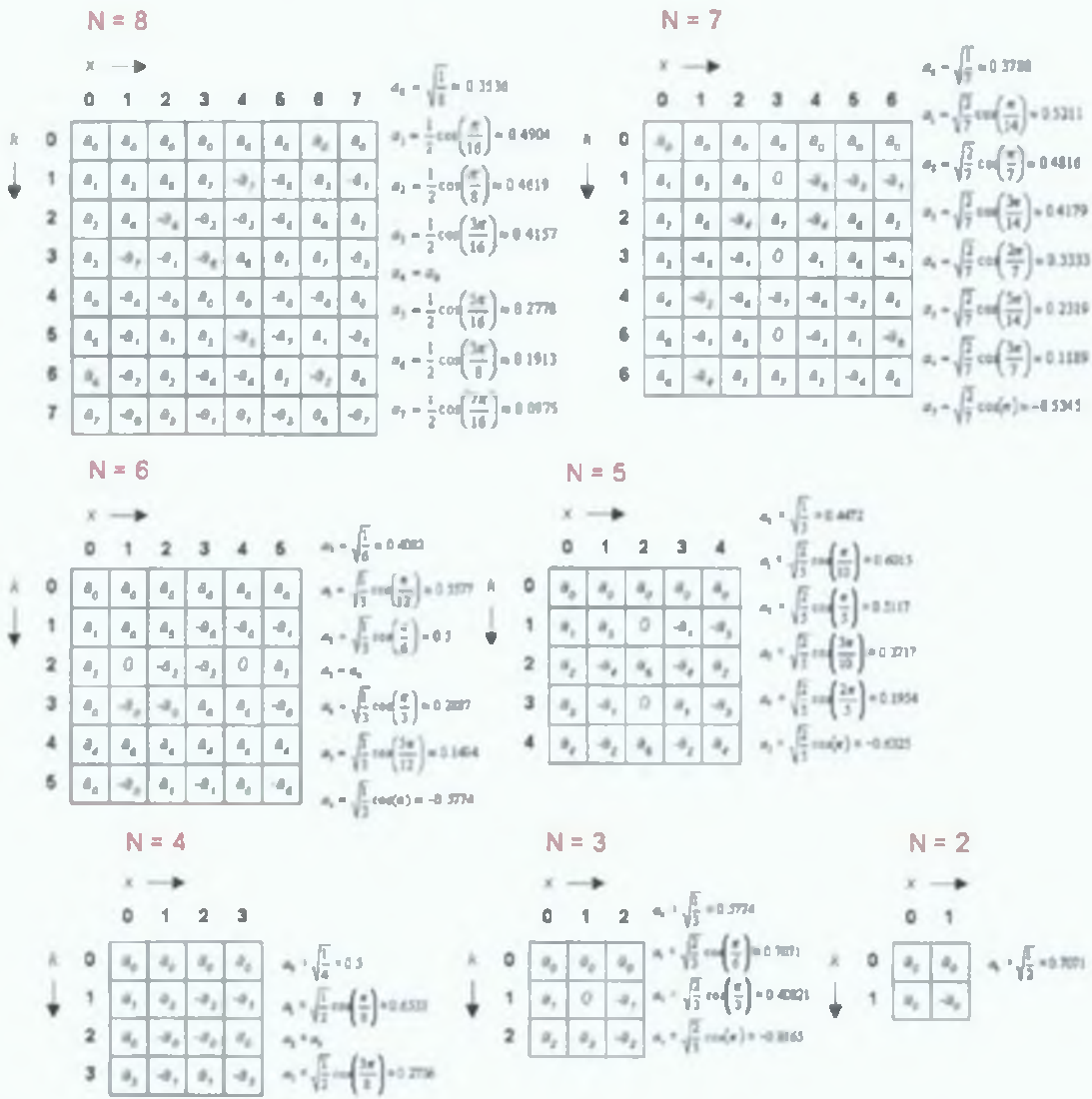


Figure 3.3: The SA-DCT Basis Functions



mean that real-time processing is more difficult to achieve. As such, there is a valid case for hardware acceleration of the SA-DCT tool especially given the relatively reduced performance and computational capability of mobile processors. Any hardware acceleration solution proposed for the SA-DCT on a mobile platform must make power efficiency a primary design feature as well as matching the other requirements for throughput and low silicon area. The proposed solution tackles these issues at a high level of abstraction, yielding a low power solution without sacrificing throughput or area, nor does it impact on design-time by requiring exotic process technologies or circuit design techniques.

## **3.2 DCT/SA-DCT Hardware State of the Art Review**

This section outlines the algorithmic and architectural approaches to implementing the  $8 \times 8$  DCT and in particular the SA-DCT. There is a huge amount of literature discussing  $8 \times 8$  DCT implementations – this is due to the computational complexity of the algorithm and its importance in various signal processing applications. As such, Section 3.2.1 aims to classify DCT implementations by general approach. Section 3.2.2 discusses SA-DCT specific implementations in detail giving context for the proposed SA-DCT architecture outlined subsequently. A condensed version of this DCT/SA-DCT implementation overview is given in [48].

### **3.2.1 Classes of DCT Implementation Approaches**

Due to the computational complexity of the DCT, many fast algorithms have been proposed in the literature. Initially, the focus in the literature was on developing fast algorithms without consideration for VLSI implementation. The fastest algorithms generally result in complex signal-flow graphs with irregular routing, complex architectures and numerous I/O pins. As a result, most VLSI implementations of the DCT in the literature use simpler, more regular, hardware-oriented algorithms. This section outlines some of the fast algorithm approaches before discussing the hardware-oriented approaches (systolic arrays, recursive structures, CORDIC, approximation-based, integer encoding and distributed arithmetic). The taxonomy of approaches is summarised in Figure 3.4.

There are two broad categories of 2D DCT computation schemes that can be implemented in both hardware (architecture) or software (algorithm). These are the direct approach (which operates directly on the 2D pixel data) and the so-called row-column approach, which exploits the separability of the DCT (see Section 2.2.2). The row-column approach is a two stage process that involves computing a 1D DCT for each of the columns in the  $8 \times 8$  pixel block, followed by a 1D DCT on each of the rows of the result. Some authors claim that the direct approach is more computationally efficient than the row-column approach since the latter is more complicated and requires additional matrix transposition architecture to store the intermediate data [139]. However, direct implementations generally suffer from irregular routing [140]. The direct and row-column approaches are very general, and can be implemented with any of the strategies listed in Figure 3.4.

#### **3.2.1.1 Fast Algorithm Approaches**

The primary goal of fast DCT algorithms is to minimise the number of arithmetic operations – especially multiplications. The early fast DCT algorithms are based around the ideas similar to the Cooley-Tukey

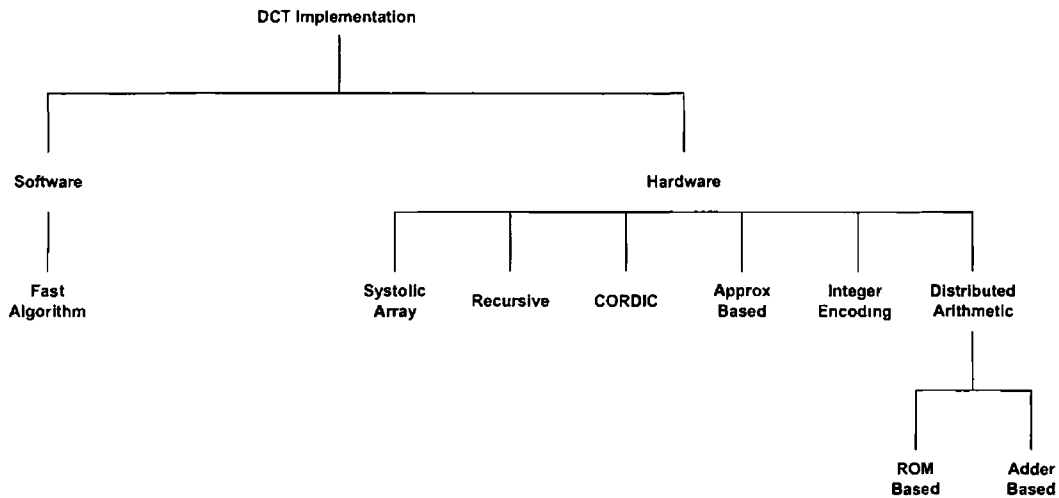


Figure 3 4 Taxonomy of DCT Implementation Approaches

FFT algorithms, namely data-reordering and butterfly processing to avoid redundant computation. The “Chen” algorithm is a common approach for implementing an  $N$  point DCT where  $N$  is a power of 2 [141]. It uses the fact that a unitary matrix can theoretically be factorised into products of relatively sparse matrices. These matrices are tailored in such a way as to reduce the number of computations. This algorithm requires 16 multiplications and 26 additions for the 8-point DCT. It also has the advantage that it uses real arithmetic, which is more hardware efficient compared to complex arithmetic approaches that exploit the Discrete Fourier Transform [142]. An approach based on the derivation of the FFT is proposed by Lee requiring 12 multiplications and 29 additions [143]. A similar approach to the Chen algorithm is proposed by Loeffler et al whose signal flow graph is based on scale-rotation operators [144]. This approach needs 11 multiplications and 29 additions and a recent paper has implemented it on an FPGA platform [145].

Another approach proposed by Feig exploits the fact that scaling and quantisation follow the DCT in a video codec [146]. The subsequent scaling means a scalar multiple of the DCT basis functions might do instead of a two step DCT and quantisation process, with appropriate compensation incorporated into the scaling. The algorithm also factorises the DCT basis matrix using the so-called “Winograd construction”. The Feig algorithm requires 13 multiplications and 29 additions. An extremely fast algorithm based on similar ideas as the Feig algorithm is the so-called “AAN” algorithm [147], which needs only 5 multiplications and 29 additions. For more details and a broader survey of fast DCT algorithms, the reader is referred to Chapter 14 of [148] and Chapter 4 of [142].

### 3 2 1 2 Systolic Approaches

The advantage of a systolic array approach is that the resulting architecture is very modular, regular and easily pipelined. Since there is only local communication between processing elements, high speed is possible. The disadvantage of such an architecture is the relatively large area requirement.

A systolic approach using single instruction multiple data (SIMD) techniques is proposed in [149]. The architecture is based around the “Lee” fast algorithm [143]. Pan et al propose a unified systolic

array architecture capable of computing not only the DCT but also the Discrete Sine Transform (DST) and the Discrete Hartley Transform (DHT) [150]. Depending on the transform required, each processing element is capable of either computing two parallel multiply-accumulate operations, or one multiply-accumulate and one multiply-deaccumulate. Aggoun et al. propose a fully parallel systolic architecture with multiply-accumulate processing elements [151]. The multiplier has an array architecture and the partial products are accumulated using a column compressor tree. A recent paper by Cheng et al. proposes a systolic array structure for an  $N$ -point DCT with the restriction that  $N$  must be a prime number [152]. In this paper the DCT is reformulated into two cyclic convolutions implemented as systolic arrays. Other references proposing systolic array architectures to compute the DCT include [153, 154, 155]. However, despite their regularity and locality, the high hardware area cost (usually proportional to  $N^2$  [156]) of systolic architectures generally make them unsuitable for VLSI implementations on a mobile platform.

### 3.2.1.3 Recursive Approaches

A recursive approach can be used to compute the DCT at varying levels of granularity. At the coefficient level, a recursive operation could be executed iteratively to yield a single DCT coefficient. However, the computation may suffer from round-off noise depending on the bitwidth of the datapath. At a higher level, recursive structures could be used to generate higher order DCTs from lower order DCTs. A recursive structure is area efficient and allows programming flexibility. This affords the possibility to selectively control the amount of coefficients computed and their accuracy to a certain degree. This could be leveraged by a dynamic power management system to reduce the amount of computation in low power mode.

Chan et al. propose an algorithm that converts a length  $2^n$  DCT into  $n$  groups of equations with the group sizes  $2^{n-1}$ ,  $2^{n-2}$ , ...,  $2^0$  respectively [157]. The result is a regular formulation suitable for implementation using a regular filter structure. Although the structure is quite simple, it does require more multiplications than some of the fast algorithms. Elnaggar et al. propose a recursive architecture based on the fact that a large 2D DCT can be decomposed exactly into a single parallel stage of four smaller 2D DCTs with additional pre and post processing [158]. A recent paper by Chen [159] outlines a family of recursive processing elements capable of computing an  $M$ -dimensional DCT that could be used in a 3-D DCT coding scheme as opposed to conventional 2-D DCT with motion compensation. The structures exploit the symmetrical properties of the DCT basis functions to reduce recursion iterations.

### 3.2.1.4 CORDIC Approaches

Another class of approaches to implementing the DCT is based on the COordinate Rotation Digital Computer (CORDIC) algorithm [160]. CORDIC is an iterative algorithm for calculating trigonometric functions including sine, cosine, magnitude and phase. It is particularly suited to hardware implementations because it does not require any multipliers. CORDIC is based upon the concept of rotating the phase of a complex number by multiplying it by a succession of constant values. If the multipliers are powers of 2 they can be done using just shifts and adds, no actual multiplier is needed.

A DCT architecture that makes use of CORDIC rotators is proposed by Jeong et al. [161]. The architecture exploits the fact that image data is highly correlated so local differences will have small

values. If these values are small, some of the CORDIC iterations may be skipped, avoiding power wastage. Another recent CORDIC approach is by Hsiao et al. [162]. The architecture for the DCT is derived from a decomposition of the DFT and uses butterfly operators and rotators as processing elements.

### 3.2.1.5 Approximation Approaches

Some recent approaches are designed for low power by using specific low power architectural techniques and algorithm approximation. In [163], Christopoulos et al. propose an approach that exploits the fact that the most useful information about image data is kept in the low-frequency DCT coefficients. Only the low-frequency DCT coefficients are computed by using a so-called ‘pruning’ algorithm. The algorithm is used for the computation of  $N_0 \times N_0$  coefficients, which are a subset of the possible  $N \times N$  coefficients with the restriction that  $N$  and  $N_0$  are powers of two. This results in computational savings, and the recursive nature of the algorithm allows the computation of the DCT coefficients in zig-zag order saving the entropy encoder this task. The authors claim that the algorithm is suitable for high-speed, limited memory applications.

Another hardware-oriented pruning scheme is proposed in [164] which investigates the quantisation parameters and the last non-zero DCT coefficient after quantisation. This information is used to adaptively make the decision of calculating all  $8 \times 8$  DCT coefficients or merely a subset of these. This is because the magnitudes of the high-frequency DCT coefficients are usually small and tend to go to zero after quantisation anyway. This paper also proposes a method to approximate the DCT coefficients, which leads to further computational savings. This is possible since, when the quantisation parameters are large, the coefficients will be coarsely quantised. Such methods are useful for low bit-rate applications such as mobile MPEG-4 codecs. This work is extended in [165] where a theoretical model for the DCT coefficients is proposed which generalises the work in [164].

August et al. propose some low power optimisations on a ‘Chen’ datapath  $8 \times 8$  DCT [166]. One technique is skipping macroblocks that have small magnitude data after motion compensation. This decision is based on a simple function of the quantisation parameter and the sum of absolute differences computed by the motion estimator. A similar approximation approach that incorporates the data-driven architectural ideas by Xanthopoulos [124] is proposed by Lin et al. [167].

### 3.2.1.6 Algebraic Integer Encoding

Algebraic integer encoding is an interesting approach that expresses each DCT cosine scale factor as an integer polynomial of a base cosine [168, 169, 170]. The underlying mathematics is too involved for a full treatment here, but a detailed explanation can be found in [171]. The advantages of algebraic integer encoding are its low complexity and high precision. The internal datapath precision only grows maximally to 13 bits, and this is considerably less than the typical 22 bits of conventional approaches. The inherent description of the integer encoding scheme is such that it is easily mappable to a distributed arithmetic structure. Distributed arithmetic structures are discussed in detail in Section 3.2.1.7, and their primary advantage is that power consumptive multipliers are not required.

There is currently a call for proposals on a fixed-point  $8 \times 8$  IDCT and DCT standard for use in the MPEG video coding standards [172], hence approaches such as algebraic integer encoding should come

to prominence in the near future. The broad goal of this activity is to come up with a low computational complexity transform scheme that can achieve acceptable performance as well as eliminating the IDCT mismatch problem and drift between encoders and decoders [173]. This mismatch problem is discussed in greater depth in Chapter 4 in the context of IDCT implementations.

### 3.2.1.7 Distributed Arithmetic Approaches

Although the DCT is an extremely complex operation, it is essentially a collection of dot products, each involving multiply-accumulate operations. Therefore, any algorithm or architecture that implements the DCT should focus on reducing this computation – particularly the multiplications. This is especially true for low power hardware solutions. One architecture technique that has evolved is Distributed Arithmetic (DA), which is a bit-serial operation that computes the dot product of two vectors, one of which is a constant (the DCT basis vectors in this case) in parallel. The advantage of DA is its efficiency and the fact that no multiplications are necessary. A more detailed explanation of the theory behind DA is provided in this section as it is the technology underpinning of the SA-DCT and SA-IDCT architectures proposed in this thesis. It is also the foundation of the proposed EDA algorithm for constant matrix multiplication circuit design proposed in Chapter 5.

DA is widely acknowledged to be extremely efficient at computing a dot product calculation when one of the vectors contains constants. The paper by White is a seminal paper in this area where the theory is explained in detail along with possible architecture optimisations [174]. A frequently argued disadvantage is apparent slowness because of its bit-serial nature. However in his paper, White discusses techniques such as offset-binary coding, bit-pairing and word partitioning to speed up the computation and claims that DA is at its most efficient when the number of input lines is equal to the number of clock cycles required to load the data. Essentially this means balancing the area of the circuit (a consequence of the number of parallel input lines) with the circuit latency (the clock cycles required to compute the dot product). DA implementations of dot product circuits where one of the vectors is constant have been shown to have attractive power consumption properties [124]. There are two broad approaches to implementing a DA computation unit – ROM-based and adder-based, both of which are introduced here. The SA-DCT/IDCT architectures proposed in this thesis adopt the adder-based approach since it requires no ROM lookups and is more scalable, as explained in the following sections.

**ROM-Based DA** Consider Equation 3.1 which shows a single dot product of variable data vector  $\mathbf{x}$  with a vector of constants  $\mathbf{A}(\mathbf{u})$

$$X(u) = \sum_{k=0}^{N-1} A(u)_k x_k \quad (3.1)$$

Assume that each data vector element  $x_k$  is a  $(|P + 1 - Q|)$ -bit two's complement binary number ( $P$  is MSB index,  $Q$  is LSB index) and so can be represented as shown in Equation 3.2 where  $b_{k(i)} \in \{0, 1\}$  is the  $i$ th bit of data vector element  $x_k$ . To be clear,  $b_{k(Q)}$  is the least significant bit (LSB) of  $x_k$  and  $b_{k(P)}$  is the sign bit (also the MSB).

$$x_k = -b_{k(P)}2^P + \sum_{i=Q}^{P-1} b_{k(i)}2^i \quad (3.2)$$

Substituting Equation 3 2 into Equation 3 1 yields

$$X(u) = \sum_{k=0}^{N-1} A(u)_k \left[ -b_{k(P)}2^P + \sum_{i=Q}^{P-1} b_{k(i)}2^i \right] \quad (3 3)$$

$$X(u) = - \sum_{k=0}^{N-1} A(u)_k b_{k(P)}2^P + \sum_{k=0}^{N-1} A(u)_k \sum_{i=Q}^{P-1} b_{k(i)}2^i \quad (3 4)$$

$$X(u) = - \sum_{k=0}^{N-1} A(u)_k b_{k(P)}2^P + \sum_{i=Q}^{P-1} \underbrace{\left[ \sum_{k=0}^{N-1} A(u)_k b_{k(i)} \right]}_Z 2^i \quad (3 5)$$

The rearrangement of the summation to arrive at Equation 3 5 is the key step in deriving the ROM-based DA computation. Consider the bracketed term  $Z$  in Equation 3 5. Since  $b_{k(i)} \in \{0, 1\}$ ,  $Z$  has only  $2^N$  possible values. The set of possible values  $Z$  can take can be precomputed, since vector  $A(u)$  is constant, and stored in a ROM of size  $2^N$ .

The corresponding bits from each of the  $N$  data vector elements  $x_k$  are fed serially to an address generation unit (i.e.  $|P + 1 - Q|$  iterations). At each serial iteration  $i$ , the corresponding  $N$  bits ( $b_{0(i)}, b_{1(i)}, \dots, b_{N-1(i)}$ ) are aggregated to form a ROM address to access one of the  $2^N$  possible  $Z$  values. The architecture is referred to as distributed because the binary bit values of  $x_k$  are distributed across  $|P + 1 - Q|$  weights. The value of  $|P + 1 - Q|$  is generally termed as the *DA precision*. The ROM value for each weight then needs to be scaled and accumulated as described in Equation 3 5. Each weight is scaled by a power of 2 so the entire process can be achieved using successive shifts and accumulations. Special care is needed with the first term of Equation 3 5 (the sign bit i.e. when  $i = P$ ) to ensure that the accumulator subtracts the weight from the partial sum. After  $|P + 1 - Q|$  cycles, the final value of  $X(u)$  is present in the result register. A sample ROM-based DA architecture with  $P = 5, Q = 0, N = 4$  is shown in Figure 3 5. The benefit associated with this approach is that no multiplications are required to implement the dot product – only additions are required. A recent proposal for a low-power FIR filter architecture uses ROM-based DA coupled with unsigned LUT data to reduce net switching [175].

In terms of ROM-based DA DCT architectures, [176] is based on the ‘‘Chen’’ algorithm, and to improve latency two 16-word ROMs are used with a Wallace column compressor accumulator. Paek et al. use the row-column decomposition scheme based on the ‘‘Chen’’ and ‘‘Lee’’ algorithms and the architecture is capable of both DCT/IDCT with ROM-based DA [177]. Karathanasis claims that most of the traditional ROM-based DA DCT implementations have large ROM requirements, and the algorithm proposed uses offset binary coding to alleviate this drawback [178]. Offset binary encoding effectively reformulates Equation 3 2 to enable ROM compression – further details on this scheme are available in [174]. Scopa et al. present a low power architecture for the 2D DCT [179]. It exploits the fact that coefficients will be subsequently quantised by adapting the precision of the calculations to the minimum required. Sub-systems are turned off when not necessary for the current computation thus saving power. The approach uses ROM-based DA with extensive use of pipelining to improve latency. Srinivasan et al. employ a recursive structure (similar to the family of approaches described in Section 3 2 1 3), but use ROM-based DA to implement the constant multipliers. Hence, the regular compact properties of the recursive structure are combined with the power efficient properties of ROM-based DA. A similar

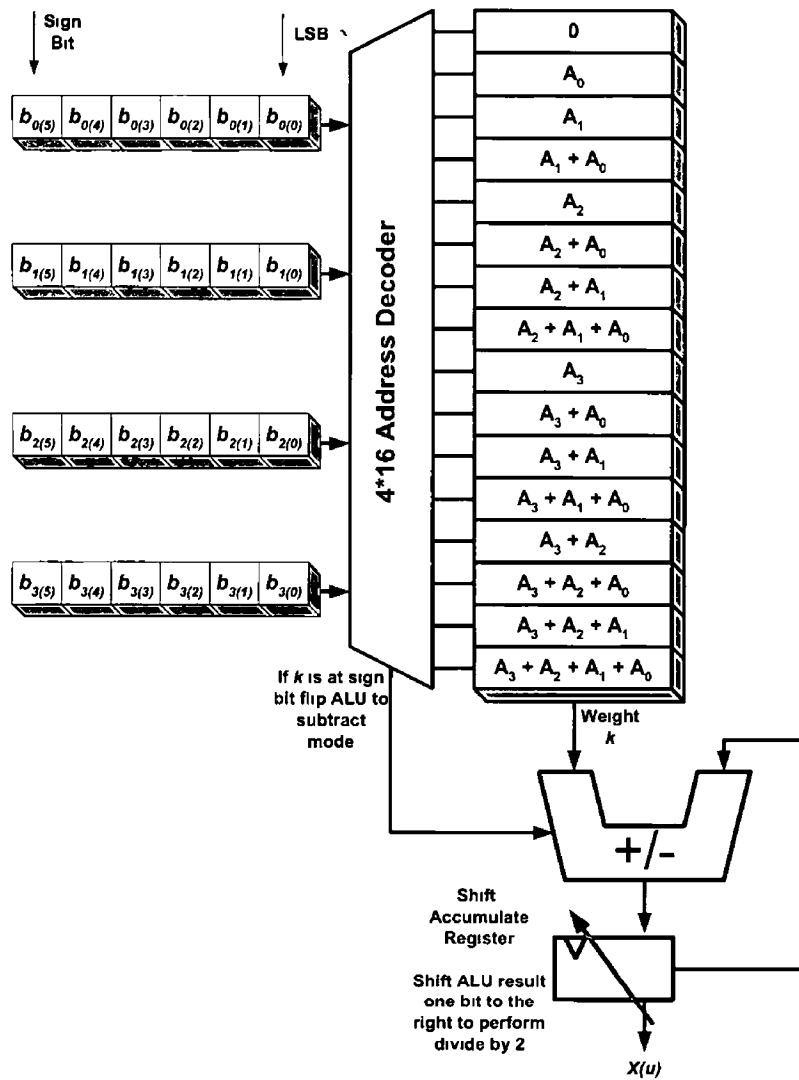


Figure 3 5 Sample ROM-based DA Architecture

approach is used in [180], where arithmetic units are turned off whenever possible. The linear array architecture used is fully pipelineable and easily scalable. Xanthopoulos et al. concentrate on low power optimisations for a ROM-based DA architecture [124]. Firstly, they exploit local pixel correlation by reducing computation dynamic range that leads to switching activity reduction (MSB Rejection). Secondly, they use precision reduction for computation related to perceptually insignificant data (Row-Column Classification). The architecture implements a row-column DA version of the “Chen” algorithm, enhanced with these activity-reduction methods. An area efficient architecture that uses ROM-based DA is proposed in [181], where a single 1D DCT/IDCT module is used to evaluate the 2-D DCT/IDCT. In essence, the module is used four times for each transform/reconstruction carried out in the codec, and is based on the row-column technique. Further approaches that adopt ROM-based DA DCT architectures can be found in [182, 183, 184, 185, 186, 187, 188, 167].

**Adder-Based DA** With adder-based DA the values in the vector of constants are distributed as opposed to the data vector, as was the case with ROM-based DA. The bits of the constant vector elements are distributed, as shown in Equation 3.6, over successive stages of adders, which combine elements of the input vector  $x$  to form weighted values. These weights are then shift-accumulated together to form the result.

$$A(u)_k = -b_{k(P)}2^P + \sum_{i=Q}^{P-1} b_{k(i)}2^i \quad (3.6)$$

Combining Equation 3.1 and Equation 3.6 results in Equation 3.9 (see Equations 3.7 and 3.8). Note that weight  $W_{(P)}$  (corresponding to the sign weight  $P$ ) needs to be 2’s complemented (change sign) since it is derived from the sign bits and has a negative weight. Since  $b_{k(i)} \in \{0, 1\}$ , each weight  $W_{(i)}$  has only  $2^N$  possible values and is a linear additive combination of input variable vector  $x$ . An element of  $x$  (namely  $x_k$ ) is included in the linear addition forming  $W_{(i)}$  if  $b_{k(i)} = 1$ , where  $k = 0, 1, \dots, N-1$ . To generate the final coefficient  $X(u)$ , all  $|P+1-Q|$  weights need to be scaled appropriately (i.e. shifted) and accumulated together.

$$X(u) = [A(u)_0 \quad A(u)_{N-1}] \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \end{bmatrix} \quad (3.7)$$

$$X(u) = \begin{bmatrix} -2^P \\ 2^{P-1} \\ \vdots \\ 2^{Q+1} \\ 2^Q \end{bmatrix}^T \underbrace{\begin{bmatrix} b_{0(P)} & b_{N-1(P)} \\ b_{0(P-1)} & b_{N-1(P-1)} \\ \vdots & \vdots \\ b_{0(Q+1)} & b_{N-1(Q+1)} \\ b_{0(Q)} & b_{N-1(Q)} \end{bmatrix}}_{A(u)_d} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \quad (3.8)$$

$$X(u) = \begin{bmatrix} -2^P \\ 2^{P-1} \\ \vdots \\ 2^{Q+1} \\ 2^Q \end{bmatrix}^T \begin{bmatrix} W_{(P)} \\ W_{(P-1)} \\ \vdots \\ W_{(Q+1)} \\ W_{(Q)} \end{bmatrix} \quad (3.9)$$



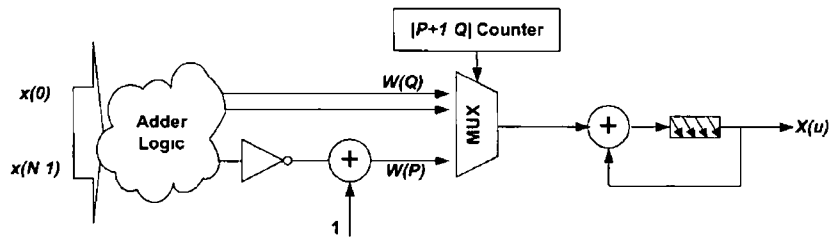


Figure 3.6 Conceptual NEDA Architecture

Again no multipliers are required, and adder-based DA has the extra bonus of requiring no ROM-lookups. This is particularly important if  $N$  is large or if the DA precision  $|P + 1 - Q|$  is large [133]. It has been shown that if implementing the DCT, an adder-based DA architecture requires less silicon area compared to ROM-based DA architecture [189]. Adder-based DA is more promising in terms of hardware reuse because the adder inputs can be easily reconfigured as demonstrated with the SA-DCT architecture presented in Section 3.4. Shams et al. refer to adder-based DA as New Distributed Arithmetic (NEDA) and this acronym will be used in the remainder of this thesis [133]. A conceptual NEDA architecture is shown in Figure 3.6. NEDA based architectures offer the possibility of optimisation in the sense that the number of adders required to compute a dot product depends on

- The length of the dot product  $N$
- The precision  $|P + 1 - Q|$  of the constant vector elements  $A(u)_k$
- The binary (or otherwise) representation of  $A(u)_k$

The impact of these parameters is discussed in greater detail in Chapter 5, Section 5.3.3.

NEDA DCT architectures have become popular recently, and Chang et al. propose a fast direct 2D algorithm with NEDA and common sub-expression sharing to reduce hardware cost and increase throughput [139]. It claims to combine a flexible programmable processor style approach with efficient dedicated hardware units. Generally, direct DCT algorithms (as described in the introduction of Section 3.2.1) require several stages of butterfly additions making routing difficult for parallel hardware implementations. Chang et al. claim to overcome this by using a programmable processor style approach at the cost of longer computation latency. Another proposal based on the “Chen” algorithm uses NEDA with radix-2 multi-bit coding to minimise resource usage [190]. It also uses symmetric transpose memory to improve latency. Another NEDA architecture that computes DCT coefficients in parallel for high throughput is proposed by Shams et al. [133]. The adder network required for each coefficient is derived using common sub-expression sharing and 35 adders are required in total to generate the DA weights in Equation 3.9. Gundimada et al. extend this idea by employing the “Chen” fast algorithm which reduced the number of weight adders to 26 [191]. The work by Shams has been developed further by Alam et al. who have proposed a time-multiplexed NEDA datapath (i.e. a serial computation scheme) to increase resource usage while maintaining low bandwidth serial I/O [192, 193]. This architecture uses 12 time-multiplexed adders to generate the DA weights. A similar approach based on the time-multiplexed idea is proposed by Guo et al. [194]. They exploit canonic signed digit representations of the basis functions and their weight generation logic only requires 10 adders. It should be noted that time-multiplexed architectures such as these achieve lower area at the expense of lower throughput.

### 3 2 1 8 Future Evolution of DCT Hardware Cores

As mobile implementations of MPEG-4 become more commonplace, research attention will focus more centrally on low power hardware acceleration cores. The DCT/IDCT tool is one of the most computationally intensive tools in the MPEG-4 toolbox and power efficient algorithms and architectures are becoming increasingly important. To include full object-based processing, architectures will need to incorporate the regular block-based DCT with a shape adaptive DCT for VO boundaries. Some of the fastest algorithms that appear in the literature are not getting much attention because they translate to very complex hardware layouts. Future DCT/IDCT hardware tools for mobile MPEG-4 will probably use simpler but slower (i.e. more serial) algorithms that have a regular and power efficient architecture. Some current low power techniques exploit the mathematical properties of the DCT/IDCT such as DCT pruning algorithms, low energy macroblock skipping and truncated multiplication. Other techniques exploit the fact that the DCT/IDCT are essentially a large multiply-accumulate operation and use distributed arithmetic as an alternative to power consumptive multiplications. There are also some general low-level techniques that can be used such as clock gating, low-transition data paths and voltage scaling [166]

### 3 2 2 SA-DCT Specific Approaches

The SA-DCT is a column-row process and cannot be easily implemented directly as a 2D equation given the large number of possible  $8 \times 8$  shape configurations. The primary computation engines required to implement the SA-DCT are a variable  $N$ -point 1D DCT processor, an alpha shape parser (for the packing steps) and a memory scheme to store the intermediate coefficients after the vertical transform. The dynamic nature of the SA-DCT processing steps pose significant VLSI implementation challenges and many of the previously proposed approaches use area and power consumptive multipliers to implement the  $N$ -point 1D DCT. Most also ignore the subtleties of the packing steps and manipulation of the shape information.

Gause et al. [195, 196] have proposed two FPGA-specific architectures for the SA-DCT. Because shape adaptive algorithms are less regular and predictable compared to the classic DCT, Gause et al. propose reconfigurable architectures, one using “static reconfiguration” and the other using “dynamic reconfiguration” of the FPGA fabric. They implement the SA-DCT packing stages using a shift register controlled by the incoming shape information. This approach induces needless register switching that can be avoided using a simple addressing scheme as proposed in Section 3.5.1. The static design has 8 parallel  $N$ -point 1D DCT processors implemented using ROM-based DA, although no details are given about the precision of the cosines or indeed the style of ROM used. The dynamic design re-programs the FPGA fabric with a dedicated  $N$ -point processor for a particular  $N$  according to the dynamic value of  $N$  (determined by the shape), although the authors admit that the time overhead involved in reprogramming the FPGA on the fly is not worth the effort [196].

Le et al. have proposed two SA-DCT architectures – a recursive structure (Figure 3.7) and a feed-forward structure (Figure 3.8) [197, 198, 199, 200]. The former employs a bank of second order IIR filters (16 multipliers and 24 adders in total) but suffers from numerical instability due to bitwidth growth, as well as using area inefficient multipliers. The latter exploits a factorised SA-DCT equation implemented by multiplexed datapath with adders and multipliers with improved numerical accuracy. The authors

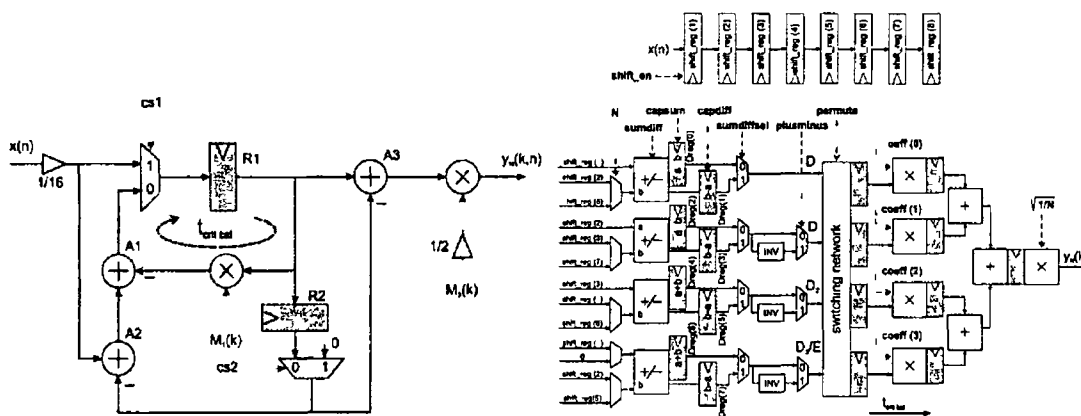


Figure 3 7 Recursive Architecture (Le et al [197]) Figure 3 8 Feed-Forward Architecture (Le et al )

favour the feed-forward architecture and as is clear from Figure 3 8, this has a hardware cost of 11 adders and 5 multipliers, with a cycle latency of  $N + 2$  for an  $N$ -point transform. However, neither of the architectures address the horizontal packing required to identify the lengths of the horizontal transforms and have the area and power disadvantage of using expensive hardware multipliers.

Tseng et al propose a reconfigurable pipeline that is dynamically configured according to the shape information [201]. The architecture (as shown in Figure 3 9) is hampered by the fact that the entire  $8 \times 8$  shape information must be parsed to configure the datapath “contexts” prior to texture processing. The paper does not describe the architecture in detail, but it seems that the SA-DCT data packing steps are computed using explicit shifts which would cause needless power consumption and add extra processing latency. Furthermore, there is no breakdown on adder and multiplier requirement and no cycle latency figure is provided.

Chen et al developed a programmable datapath that avoids multipliers by using canonic signed digit (CSD) adder-based distributed arithmetic [202, 109]. The datapath is the structure on the right in Figure 3 10. The datapath functionality can be described by the equation  $bus = Ra \pm Rb \gg s$  where  $bus$  denotes the output of the datapath and  $\gg s$  means right shift by  $s$ -bits. The hardware cost of the datapath is 3100 gates requiring only a single adder, which is re-used recursively when computing the multiply-accumulates. This small area is traded off against cycle latency – 119 cycles in the worst case scenario. The architecture is energy-aware in the sense that the data word length precision can be configured by a supervising dynamic power manager. Although it has good area properties it approximates odd length DCTs by padding to the next highest even DCT, and also does not address the SA-DCT horizontal packing requirement. The authors do not comment on the perceptual performance degradation or otherwise caused by approximating odd length DCTs with even DCTs.

Lee et al considered the packing functionality requirement and developed a resource shared datapath using adders and multipliers coupled with an auto-aligning transpose memory (Figure 3 11) [203]. As is clear from Figure 3 11, the datapath is implemented using 4 multipliers and 11 adders. The worst case computation cycle latency is 11 clock cycles. The transpose memory in Figure 3 11 is used to store the intermediate coefficients computed by the vertical transformations prior to horizontal transformation. The horizontal packing step is inherent in the addressing scheme and is similar to the independently conceived scheme proposed in Section 3 5 2. Essentially, the transpose memory is partitioned into eight

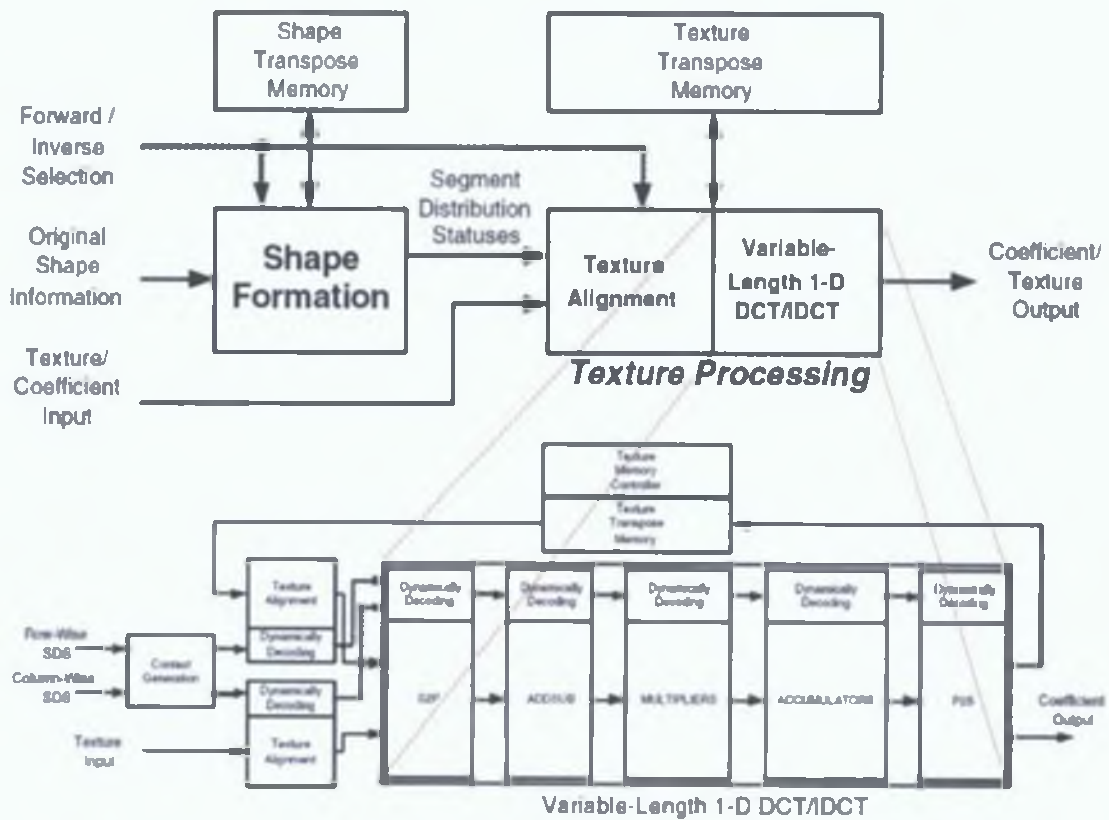


Figure 3.9: Reconfigurable Processor Architecture (Tseng et. al. [201])

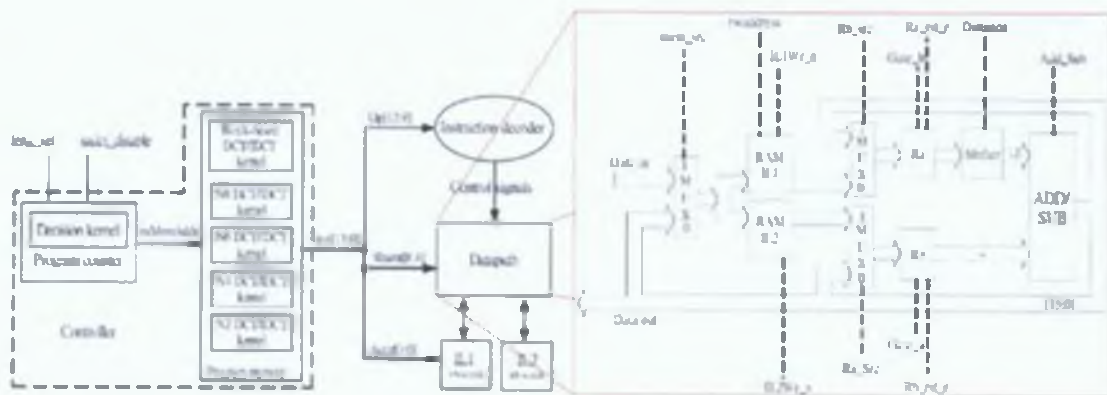


Figure 3.10: Programmable Processor Architecture (Chen et. al. [109])

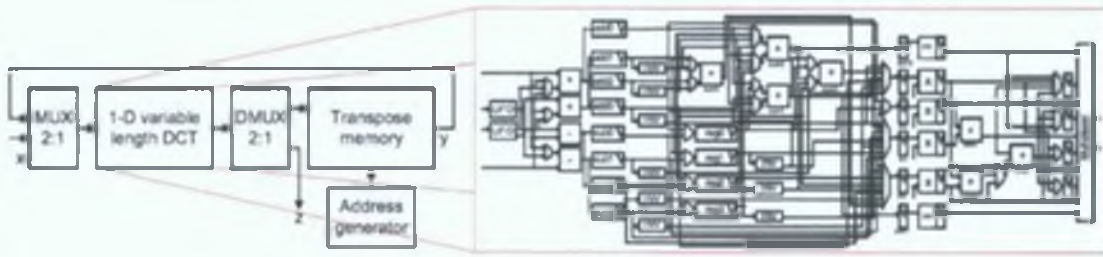


Figure 3.11: Auto-Aligning Architecture (Lee et. al.) [203]

rows. Using eight address pointers for these rows  $row\_ptr(k)$  where  $0 \leq k \leq 7$ , coefficient  $k$  from each column is stored in location  $row\_ptr(k)$  and then  $row\_ptr(k)$  is incremented by one. As such, the data is horizontally aligned immediately after the vertical transforms are complete. This is the most advanced implementation, but the critical path caused by the multipliers in this architecture limits the maximum operating frequency and has negative power consumption consequences.

The SA-DCT architecture proposed in this thesis (Sections 3.4 and 3.5) tackles these issues by employing a reconfiguring adder-only based distributed arithmetic structure. It is estimated that an  $m$ -bit Booth multiplier costs approximately 18-20 times the area of an  $m$ -bit ripple carry adder [109]. In terms of power consumption, the ratio of multiplier power versus adder power is not the same as the area ratio since the transition probabilities for the individual nodes are different for both circuits. To get an estimate of the power cost of multipliers versus adders, a  $16 \times 16$  radix-4 Booth multiplier and a 16-bit ripple carry adder were implemented in Verilog. Using 1000 random test vectors, gate-level power analysis was carried out according to the steps outlined in Section 2.3.2.4. The results show that in the 16-bit case, the multiplier is approximately 12 times more power consumptive compared to the adder. It is primarily for this reason that the SA-DCT architecture proposed in this thesis is implemented with adders only. As discussed in Section 3.6, the author believes that compared to the prior art, the architecture proposed in this thesis offers a better trade-off between speed, area and power. The proposed datapath serially computes each coefficient  $k$  ( $k = 0, \dots, N - 1$ ) of an  $N$ -point 1D DCT by reconfiguring the datapath based on the value of  $k$  and  $N$ . Local clock gating is employed based on  $k$  and  $N$  to ensure that redundant switching is avoided for power efficiency. A transpose memory (TRAM) has been designed whose surrounding control logic ensures that the SA-DCT horizontal packing is computed efficiently without needless switching or shifting and does not delay the commencement of the horizontal transformation process.

### 3.3 Design Methodology

As outlined in Chapter 2, the best opportunity for power savings when designing a hardware accelerator is at the system and algorithmic level since there are wider degrees of design freedom. The design techniques adopted when implementing the proposed SA-DCT accelerator may be summarised as follows:

- An “algorithm washing” methodology is adopted that involves translating the behavioural specification of the SA-DCT algorithm into a hardware-oriented concurrent system with task scheduling, timing and communication [25]. This step serves to illuminate task concurrency, memory transfer requirements and the tasks required in mapping of the algorithm into discrete hardware processing

units.

- The behaviour of the hardware system should reduce the complexity of the required processing wherever possible on the basis that if there are fewer operations to be carried out, there will be less switching and hence less energy dissipation [37]. This can be achieved by avoiding needless computation and via early task termination.
- A balance should be struck between a fully parallel architecture and a time-multiplexed resource sharing architecture according to design constraints. Parallel processing enables quicker task computation but requires more silicon area. Time-multiplexing and re-use of hardware resources keeps circuit area low but increases computation latency.
- Implementation of algorithm tasks with power efficient hardware structures (e.g. implementation of DCT multiply-accumulate operations using adder-based DA).

The properties of the proposed SA-DCT architecture may be summarised as follows:

- Even/odd decomposition of SA-DCT basis matrices to reduce computational complexity (Section 3.4.1).
- Reconfiguring adder-based DA implementation of a variable  $N$ -point DCT processor avoiding power consumptive hardware multipliers (Section 3.4.2 & 3.4.3).
- Efficient addressing scheme avoiding explicit shifting to identify column/row lengths and associated data (Section 3.5).
- General low power properties such as a pipelined/interleaved datapath and local clock gating controlled by the processing load requirements.

### 3.4 SA-DCT Datapath Architecture

The top-level SA-DCT architecture is shown in Figure 3.12, comprising of the TRAM and datapath with their associated control logic. For all modules, local clock gating is employed based on the nature of the data to be processed (i.e. row/column length and coefficient index) to avoid wasted power. The addressing control logic (ACL) reads the  $8 \times 8$  pixel and shape information in a vertical raster fashion (as in Figure 3.13) into a set of interleaved pixel vector buffers that store the pixel data and evaluate  $N$  with minimal switching avoiding explicit vertical packing. When loaded, a vector is then passed to the variable  $N$ -point 1D DCT module, which computes all  $N$  coefficients serially starting with  $F[N - 1]$  down to  $F[0]$ . This is achieved using even/odd decomposition, followed by adder-based distributed arithmetic using a multiplexed weight generation module (MWGM) and a partial product summation tree (PPST). The TRAM has a 64 word capacity, and when storing data, the index  $k$  is manipulated to store the value at address  $8 \times k + N_{horz}[k]$ . Following this,  $N_{horz}[k]$  is incremented by 1. In this way, when an entire block has been vertically transformed, the TRAM has the resultant data stored in a horizontally packed manner with the horizontal  $N$  values ready immediately without shifting. The ACL addresses the TRAM to read the appropriate row data and the datapath is re-used to compute the final SA-DCT coefficients that are routed to the module output.

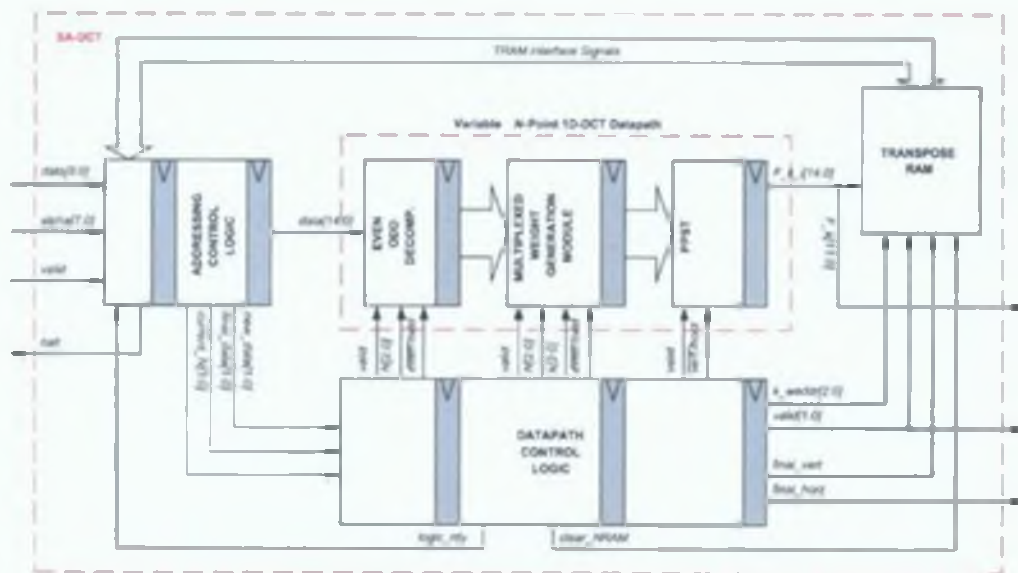


Figure 3.12: Top-Level SA-DCT Architecture

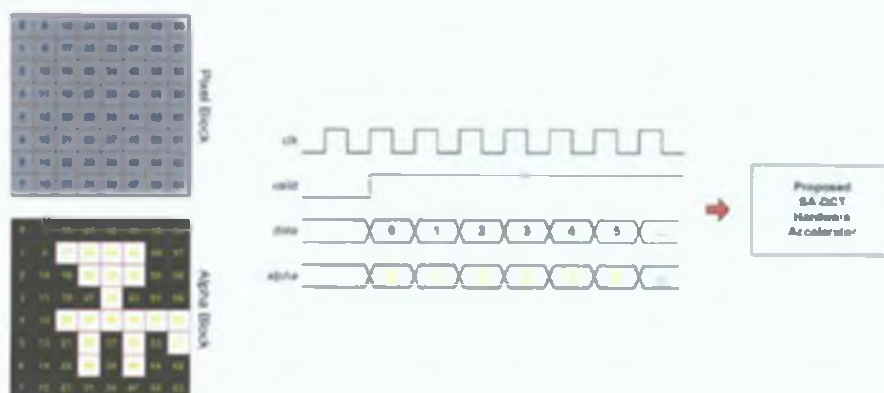


Figure 3.13: SA-DCT Vertical Raster Data Scanning

A serial coefficient computation scheme was chosen because it facilitates simpler parsing of shape information and hence simpler data in interpretation and addressing. Also, the area of the serial datapath is smaller compared to a parallel scheme although the processing latency increases slightly. The reason for only a slight increase is due to the way the SA-DCT packing stages have been incorporated into the architecture. Architectural variations are discussed in more detail in Section 3.7.

### 3.4.1 Even-Odd Decomposition

Even-odd decomposition exploits the inherent symmetries in the SA-DCT cosine basis functions to reduce the complexity of the subsequent MWGM computation. Referring to Figure 3.3, it is clear that the rows of each of the  $N$ -point DCT basis functions exhibit either symmetry (for even  $k$ ) or anti-symmetry (for odd  $k$ ). This can be exploited to reduce the amount of computation required. For example, consider the 8-point DCT according to Equation 2.8a re-formulated as a matrix multiplication as shown in Equation 3.10 (with basis elements as labelled Figure 3.3)

$$\begin{bmatrix} F(0) \\ F(1) \\ F(2) \\ F(3) \\ F(4) \\ F(5) \\ F(6) \\ F(7) \end{bmatrix} = \begin{bmatrix} a_0 & a_0 & a_0 & a_0 & a_0 & a_0 & a_0 & a_0 \\ a_1 & a_3 & a_5 & a_7 & -a_7 & -a_5 & -a_3 & -a_1 \\ a_2 & a_6 & -a_6 & -a_2 & -a_2 & -a_6 & a_6 & a_2 \\ a_3 & -a_7 & -a_1 & -a_5 & a_5 & a_1 & a_7 & -a_3 \\ a_0 & -a_0 & -a_0 & a_0 & a_0 & -a_0 & -a_0 & a_0 \\ a_5 & -a_1 & a_7 & a_3 & -a_3 & -a_7 & a_1 & -a_5 \\ a_6 & -a_2 & a_2 & -a_6 & -a_6 & a_2 & -a_2 & a_6 \\ a_7 & -a_5 & a_3 & -a_1 & a_1 & -a_3 & a_5 & -a_7 \end{bmatrix} \begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \\ f(6) \\ f(7) \end{bmatrix} \quad (3.10)$$

Direct implementation of Equation 3.10 requires 64 multiplications and 56 additions. However, the symmetry and anti-symmetry inherent in the basis matrix in Equation 3.10 means it can be factorised into two smaller matrix multiplications (Equations 3.11a and 3.11b)

$$\begin{bmatrix} F(0) \\ F(2) \\ F(4) \\ F(6) \end{bmatrix} = \begin{bmatrix} a_0 & a_0 & a_0 & a_0 \\ a_2 & a_6 & -a_6 & -a_2 \\ a_0 & -a_0 & -a_0 & a_0 \\ a_6 & -a_2 & a_2 & -a_6 \end{bmatrix} \begin{bmatrix} f(0) + f(7) \\ f(1) + f(6) \\ f(2) + f(5) \\ f(3) + f(4) \end{bmatrix} = \mathbf{A}_{\text{even}} \begin{bmatrix} s(0) \\ s(1) \\ s(2) \\ s(3) \end{bmatrix} \quad (3.11a)$$

$$\begin{bmatrix} F(1) \\ F(3) \\ F(5) \\ F(7) \end{bmatrix} = \begin{bmatrix} a_1 & a_3 & a_5 & a_7 \\ a_3 & -a_7 & -a_1 & -a_5 \\ a_5 & -a_1 & a_7 & a_3 \\ a_7 & -a_5 & a_3 & -a_1 \end{bmatrix} \begin{bmatrix} f(0) - f(7) \\ f(1) - f(6) \\ f(2) - f(5) \\ f(3) - f(4) \end{bmatrix} = \mathbf{A}_{\text{odd}} \begin{bmatrix} d(0) \\ d(1) \\ d(2) \\ d(3) \end{bmatrix} \quad (3.11b)$$

A direct implementation of the factorised equations requires only 32 multiplications and 32 additions. This technique is referred to as even-odd decomposition (EOD), and can be applied to all  $N$ -point DCT basis matrices shown in Figure 3.3. The appropriate decomposition of the  $N$ -point data vector obviously depends on  $N$  and whether the coefficient  $k$  being computed is even or odd. These decompositions are summarised in Table 3.1 where  $X$  signifies a "don't care" value. In this context, a "don't care" value for a



Table 3 1 SA-DCT  $N$ -point Data Decomposition

$N = 8$	$N = 7$	$N = 6$	$N = 5$
$s(0) = f(0) + f(7)$	$s(0) = f(0) + f(6)$	$s(0) = f(0) + f(5)$	$s(0) = f(0) + f(4)$
$s(1) = f(1) + f(6)$	$s(1) = f(1) + f(5)$	$s(1) = f(1) + f(4)$	$s(1) = f(1) + f(3)$
$s(2) = f(2) + f(5)$	$s(2) = f(2) + f(4)$	$s(2) = f(2) + f(3)$	$s(2) = f(2)$
$s(3) = f(3) + f(4)$	$s(3) = f(3)$	$s(3) = X$	$s(2) = X$
$d(0) = f(0) - f(7)$	$d(0) = f(0) - f(6)$	$d(0) = f(0) - f(5)$	$d(0) = f(0) - f(4)$
$d(1) = f(1) - f(6)$	$d(1) = f(1) - f(5)$	$d(1) = f(1) - f(4)$	$d(1) = f(1) - f(3)$
$d(2) = f(2) - f(5)$	$d(2) = f(2) - f(4)$	$d(2) = f(2) - f(3)$	$d(2) = X$
$d(3) = f(3) - f(4)$	$d(3) = X$	$d(3) = X$	$d(2) = X$
$N = 4$	$N = 3$	$N = 2$	$N = 1$
$s(0) = f(0) + f(3)$	$s(0) = f(0) + f(2)$	$s(0) = f(0) + f(1)$	$s(0) = f(0)$
$s(1) = f(1) + f(2)$	$s(1) = f(1)$	$s(1) = X$	$s(1) = X$
$s(2) = X$	$s(2) = X$	$s(2) = X$	$s(2) = X$
$s(3) = X$	$s(3) = X$	$s(3) = X$	$s(2) = X$
$d(0) = f(0) - f(3)$	$d(0) = f(0) - f(2)$	$d(0) = f(0)$	$d(0) = X$
$d(1) = f(1) - f(2)$	$d(1) = X$	$d(1) = f(1)$	$d(1) = X$
$d(2) = X$	$d(2) = X$	$d(2) = X$	$d(2) = X$
$d(3) = X$	$d(3) = X$	$d(3) = X$	$d(2) = X$

register (one of  $s(0)$ ,  $s(3)$ ,  $d(0)$ ,  $d(3)$ ) indicates that the register is not needed for that particular  $N$ -point transform. All 8 registers in Figure 3 14 are required for the  $N = 8$  1D DCT, but if  $N < 8$ , the registers with “don’t care” values can be clock gated. The EOD module (Figure 3 14) decomposes the input vector and re-uses the same adders for both even and odd  $k$ . This adder re-use requires MUXs but the savings in terms of adders offsets this and results in an overall area improvement with only a slight increase in critical path delay. Register clocking of  $s$  and  $d$  is controlled so that switching only occurs when necessary. For example, when computing coefficient  $k = 0$  when  $N = 4$ , only registers  $s(0)$  and  $s(1)$  are clocked since the other data is irrelevant and ignored by the subsequent DA circuitry anyway. It is clear that register data is only switched when necessary to save needless power consumption.

### 3 4 2 Reconfiguring Adder-Based Distributed Arithmetic Dot Product

The dot product of the data vector (decomposed by the EOD module) with the appropriate  $N$ -point DCT basis vector yielding SA-DCT coefficient  $k$  is computed using a reconfiguring adder-based distributed arithmetic structure (the MWGM in Figure 3 15) followed by a PPST (see Section 3 4 3). Multipliers are avoided since they are expensive if system power and area is constrained, particularly if single cycle multipliers are employed. In our case, in order to use only adders in a serial computation scheme, the approach comes at a cost of additional multiplexing logic, but as evidenced by the synthesis results, overall gains are achieved (see Section 3 6). Using dedicated units for different  $\{k, N\}$  combinations (where at most only one will be active at any instant) is avoided by employing a reconfiguring multiplexing structure based on  $\{k, N\}$  that re-uses single resources.

Experimental results using the proposed approach have shown that for a range of video test sequences, 13 distributed binary weights are needed to adequately satisfy reconstructed image quality requirements [204]. This effectively means that the constant cosine scale factors are implemented using

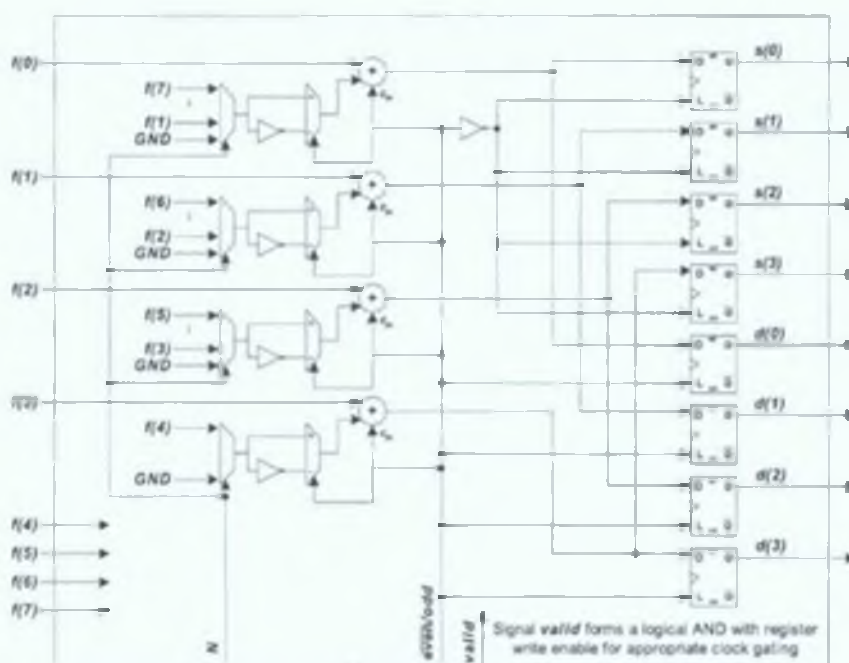


Figure 3.14: Even-Odd Decomposition (EOD) Architecture

13 bit fixed-point precision. The details of these experiments are outlined in Section 3.6.1. The adder requirement (11 in total) for the 13-weight MWGM has been derived using a recursive iterative matching algorithm [40]. This algorithm uses common sub-expression elimination to exploit data re-use potential. However, the author is currently researching an approach investigating other signed-digit representations of the cosine basis functions (not restricted to two's complement or canonic signed digit) and the effects on the inferred adder-MUX structure in terms of area, speed and power. This problem is discussed in greater depth in Chapter 5.

The datapath of the MWGM is configured to compute the distributed weights (see Equation 3.9) for  $N$ -point DCT coefficient  $k$  using the 6-bit vector  $\{k, N\}$  as shown in Figure 3.15. Even though  $0 \leq N \leq 8$ , the case of  $N = 0$  is redundant so the range  $1 \leq N \leq 8$  can be represented using three bits (range  $0 \leq k \leq N - 1$  also requires three bits). Even though the select signal is 6 bits wide only 36 cases are valid (of the  $2^6 = 64$  possible) since the 28 cases where  $k \geq N$  do not make sense so the MUX logic complexity is reduced, since only 36:1 MUXs are needed. In addition, for each of the weights there is a certain degree of equivalence between subsets of the 36 valid cases which again decreases the MUX complexity. To illustrate this concept, consider the simple 4:1 MUX as shown in Figure 3.16, where the inputs  $D$  are analogous to the data values for the weights in the proposed MWGM. If  $D_0$  is always zero, by Boolean algebra the bottom AND gate in Figure 3.16 is not needed so can be optimised away. If  $D_2 = D_3$ , then by Boolean algebra the logic for the top two AND gates in Figure 3.16 can be simplified to a single two input AND gate as shown in Figure 3.17. The optimisations in Figure 3.17 are quite simple, but can be widely applied to each of the 36:1 MUXs in Figure 3.15. Signal *even/odd* (equivalent to the LSB of  $k$ ) selects the even or odd decomposed data vector and the selected vector (signals  $x_0, x_1, x_2$ , and  $x_3$  in Figure 3.15) drive the 11 adders. Based on  $\{k, N\}$ , the MUXs select the appropriate value for each of the 13 weights. There are 16 possible values (zero and signals  $x_0, x_1, x_2,$

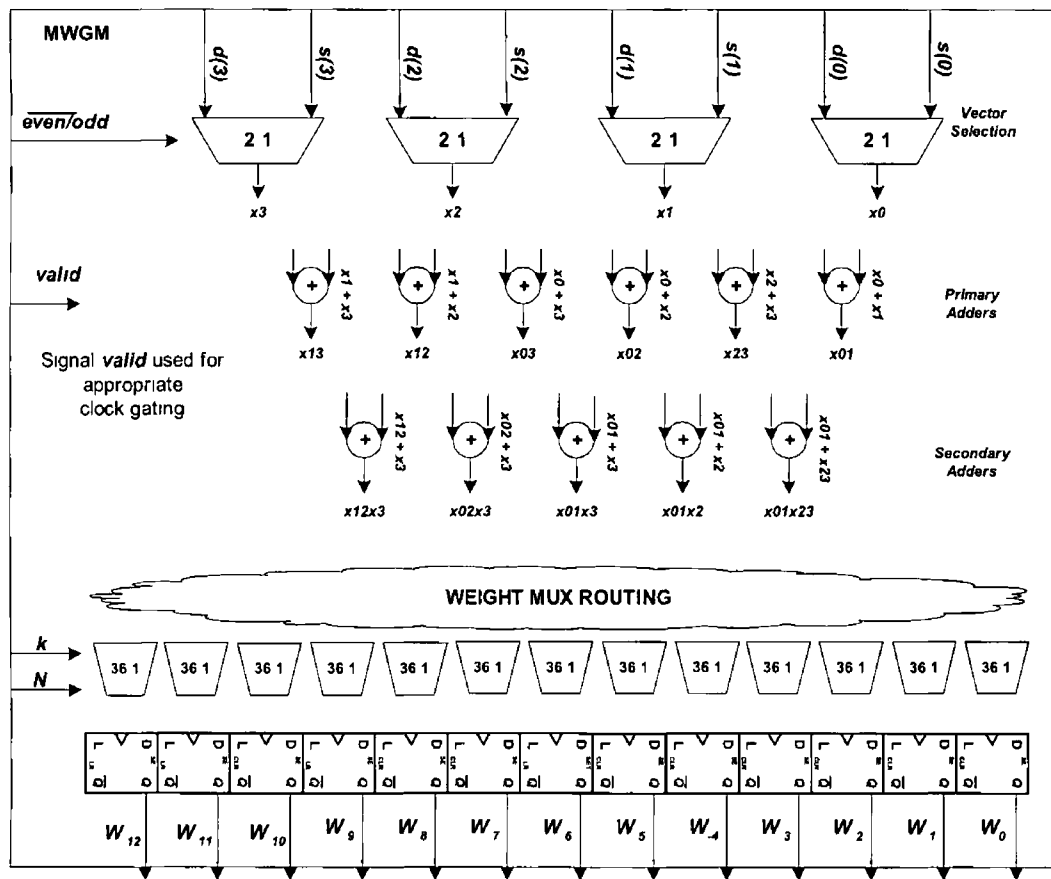


Figure 3 15 Multiplexed Weight Generation Module (MWGM) Architecture

$x_3$   $x_{01}$   $x_{23}$   $x_{02}$   $x_{03}$   $x_{12}$   $x_{13}$   $x_{01x_{23}}$   $x_{01x_2}$   $x_{01x_3}$   $x_{02x_3}$   $x_{12x_3}$  in Figure 3 15) although each weight only chooses from a subset of these possibilities as illustrated in Table 3 2 The weights are then combined by the PPST to produce coefficient  $F(k)$  as described in Section 3 4 3

As is clear from Figure 3 15, the primary adders are all six possible two input addition combinations of the selected 4-point vector (e.g.  $x_{01} = x_0 + x_1$ ). The secondary two-input adders combine a subset of the possible primary adder sums with elements of the selected vector (e.g.  $x_{01x_2} = x_{01} + x_2$ ). Note that one of the secondary adders  $x_{01x_{23}}$  combines primary adder sums  $x_{01}$  and  $x_{23}$  together representing the summation of the entire selected vector  $x_0 + x_1 + x_2 + x_3$ . The important point to note is that not all possible additive combinations of the selected vector  $\{x_0, x_1, x_2, x_3\}$  are used to form the weights forming SA-DCT coefficients for all allowed  $\{k, N\}$ . It is possible that this adder count of 11 can be reduced depending on the representation of the cosine basis functions and this is presently being investigated.

Again, power consumption issues have been considered by providing a *valid* signal that permits the data in the weight registers to only switch when the control logic flags that this is necessary. The logic paths have been balanced in the implementation in the sense that the delay paths from each of the MWGM input ports to the data input of the weight registers are as similar as possible. This has been achieved by designing the adders and multiplexers in a tree structure as shown in Figure 3 15, reducing the probability of net glitching when new data is presented at the input ports.

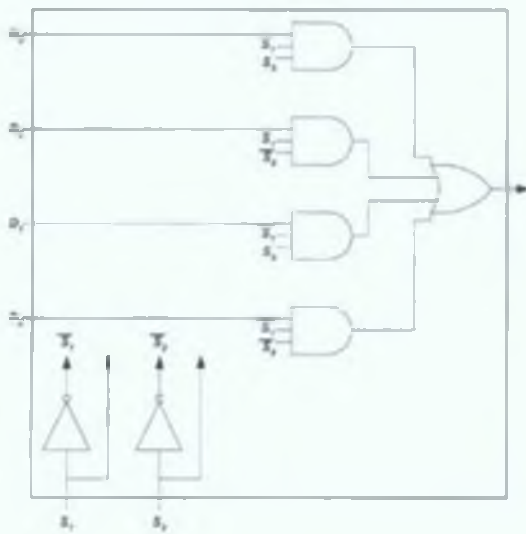


Figure 3.16: Simple 4:1 MUX

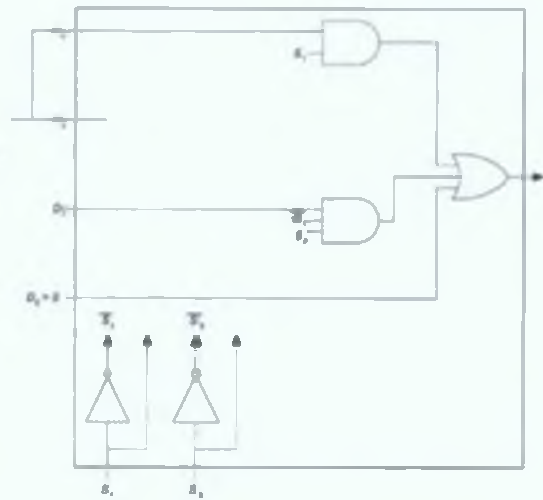


Figure 3.17: Data Optimised 4:1 MUX

Table 3.2: Total Number of Unique Possibilities for Each Distributed SA-DCT Weight

	$W_0$	$W_{-1}$	$W_{-2}$	$W_{-3}$	$W_{-4}$	$W_{-5}$	$W_{-6}$	$W_{-7}$	$W_{-8}$	$W_{-9}$	$W_{-10}$	$W_{-11}$	$W_{-12}$
# Unique Selections	8	11	11	13	13	14	10	11	9	13	13	12	12

### 3.4.3 Partial Product Summation Tree

The use of adder-based distributed arithmetic necessitates a PPST to combine the weights together to form the final coefficient as shown in Figure 3.19. On the left hand side of Figure 3.19, the appropriate alignment of the 13 weights produced by the MWGM requiring summation is illustrated. Equation 3.12 describes the accumulation of the weights mathematically.

$$F(k) = \sum_{i=0}^{-12} sW_i 2^i \quad s = -1 \text{ when } i = 0, s = 1 \text{ otherwise} \quad (3.12)$$

The PPST architecture is illustrated on the right hand side of Figure 3.19. Since this multi-operand addition is a potentially critical path, a carry-save Wallace tree structure using (3:2) counters and (4:2) compressors has been used to postpone carry propagation until the final ripple carry addition [160, 205, 206]. A general  $(k : r)$  counter, see Figure 3.18, is a combinational logic processing element where  $r$  represents a binary number corresponding to the number of 1's at the  $k$ -bit input [207]. A general  $(k; 2)$  compressor, see Figure 3.18, is also a combinational logic processing element, but there are  $l_{in}$  additional inputs and  $l_{out}$  additional outputs ( $l_{in} = l_{out}$ ). The compressor performs the operation  $k + l_{in} = s + 2(c + l_{out})$  [160]. A Wallace tree topology arranges instances of these processing elements as shown in Figure 3.19. To understand why this approach was adopted, consider that if a simple ripple-carry adder tree is used to add  $p \times k$ -bit numbers, the delay through the network is  $O(k + \ln(p))$  [206]. The absolute minimum time is  $O(\ln(kp))$ , where  $kp$  is the number of bits, and using a carry save tree this minimum is achievable since all carry propagation is deferred until a final single ripple-carry adder, which combines the two outputs of the final compressor [206]. The design has been implemented in a modular fashion so

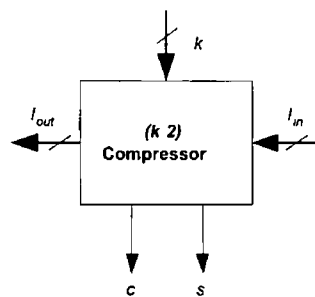


Figure 3 18 General Counter and Compressor

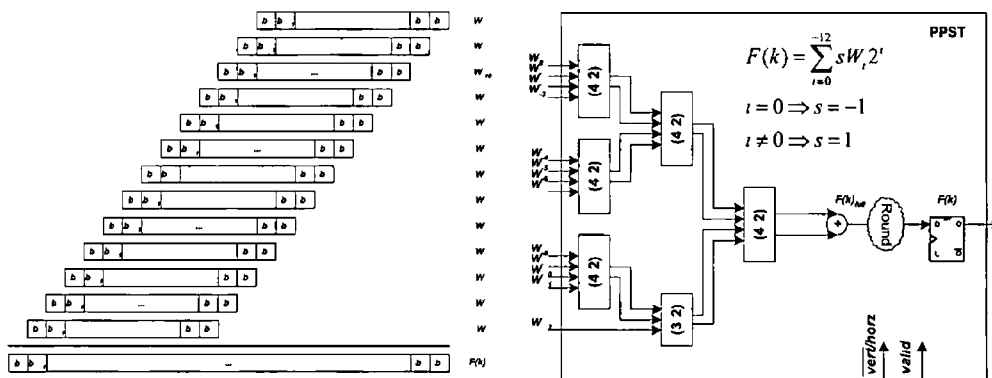


Figure 3 19 Partial Product Summation Tree (PPST) Behavioural Architecture

it is straightforward to swap in compressor structures other than the Wallace arrangement, e.g. Dadda, overturned-stairs tree, balanced tree [160]

The weighted nature of the inputs means that the sign extension can be manipulated to reduce the circuit complexity of the high order compressors [160]. This manipulation can be illustrated by an example. Consider a general 6 bit two's complement number  $sb_4b_3b_2b_1b_0$ , where  $s$  is the sign bit. Now consider for illustration five of these 6 bit numbers, which represent distributed weights that require accumulation using a PPST according to Equation 3.12 (with  $i = 0, \dots, -4$  as opposed to  $i = 0, \dots, -12$  for ease of illustration). The appropriate alignment of the five weights are illustrated in Figure 3.20, where the shaded bits are the necessary sign extension bits. The size of the compressor for each column is listed underneath each column. Using the identity  $s = 1 - \bar{s}$  illustrated in Figure 3.21, the weights in Figure 3.20 may be reformulated as shown in Figure 3.22. By pre-adding the '1' values (since they are constant), the PPST inputs look like Figure 3.23. It is clear that the required size of the leading bit compressors are reduced. Another useful identity is  $\bar{s} + 1 = \{\bar{s}, s\}$  (see Figure 3.24). This identity can be applied to  $W_{-4}$  to reduce the height of the (6,2) compressor in Figure 3.23 to a (5,2) compressor, as illustrated in Figure 3.25. This means increasing the size of the left adjacent compressor from (4,2) to (5,2), but from a circuit delay perspective it is better to have two (5,2) compressors as opposed to a (6,2) and a (4,2). It is clear that both identities exploited in this example reduce the complexity of the PPST logic. Hence both are applied to the design of the PPST required to implement the accumulation of the 13 distributed weights produced by the MWGM outlined in Section 3.4.2.

A convergent rounding scheme controlled by the  $\overline{vert}/horz$  signal is employed to round the full-precision coefficient to the appropriate precision. Vertical coefficients are rounded to 11  $f$  fixed-point

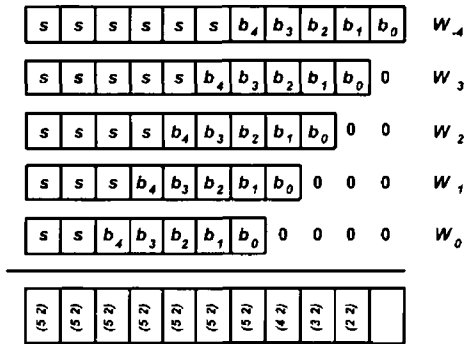


Figure 3 20 Original Alignment of PPST Weights

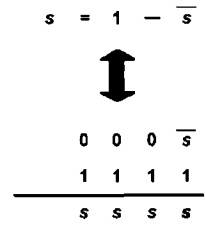


Figure 3 21 First Sign Extension Exploit

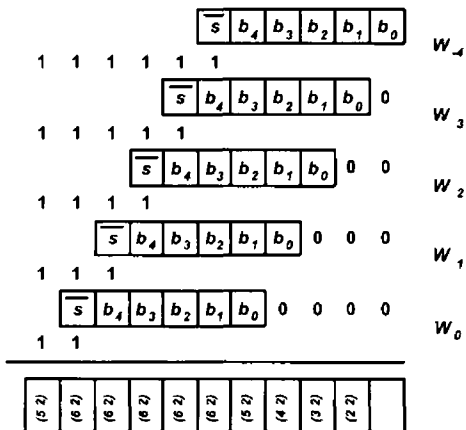


Figure 3 22 Reformulated PPST Weights

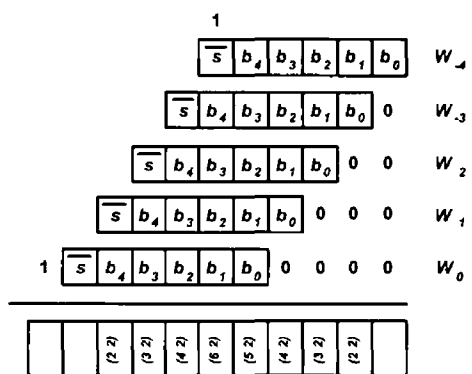


Figure 3 23 Pre-addition of Constants

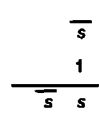


Figure 3 24 Second Sign Extension Exploit

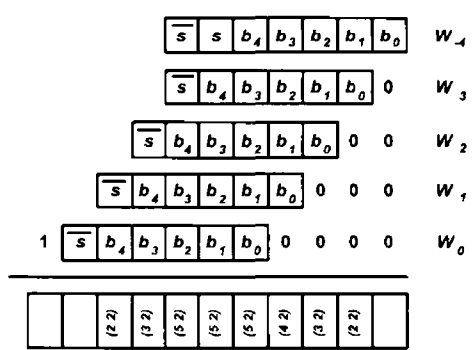


Figure 3 25 Optimised PPST Logic

bits (11 bits for integer and  $f$  bits for fractional) and the experiments show that  $f = 4$  represents the smallest bitwidth that still meets the performance requirements defined by the MPEG-4 standard (see Section 3.6.1). This implies that each word in the TRAM is 15 bits wide. Horizontal coefficients are rounded to 12.0 bits and are routed directly to the module outputs.

### 3.5 SA-DCT Memory and Control Architecture

The memory and addressing modules in Figure 3.12 avoid redundant register switching and latency when reading data and storing intermediate values. SA-DCT vertical and horizontal packing are computed implicitly by using an addressing scheme that tracks the VOP block shape (row and column masks). The proposed addressing scheme reads pixel and shape information from the input bus in an efficient manner (ACL in Figure 3.26). The ACL implicitly performs SA-DCT vertical packing as it reads data from the bus with minimal switching and routes data to the datapath for vertical transformation. Vertical coefficients require buffering prior to horizontal transformation so are stored in a local transpose memory (TRAM in Figure 3.28). When writing to the TRAM the addressing implicitly performs SA-DCT horizontal packing meaning that the horizontal transform process can begin without delay after the vertical transforms are complete. The ACL reads these transpose coefficients from the TRAM for horizontal transformation. Since the shape information for this data has already been determined after the initial shape parsing stage, the number of TRAM accesses is minimised to shape only data. Hence the processing latency for smaller shapes is less compared to the latency for opaque  $8 \times 8$  block.

#### 3.5.1 Addressing/Routing Control Logic

The ACL has a set of pipelined data registers (*BUFFER* and *CURRENT*) that are used to buffer up data before routing to the variable  $N$ -point DCT datapath. There are also a set of interleaved modulo-8 counters (*N\_buff\_A<sub>r</sub>* and *N\_buff\_B<sub>r</sub>*). Each counter either stores the number of VOP pels in *BUFFER* or the number of VOP pels in *CURRENT*, depending on the selection signal *buff\_sel<sub>r</sub>* as is clear from Figure 3.26. This pipelined/interleaved structure means that as soon as the data in *CURRENT* has completed processing, the next data vector has been loaded into *BUFFER* with its shape parsed. It is immediately ready for processing, thereby maximising throughput and minimising overall latency.

Data is read serially from the external data bus if in vertical mode or from the local TRAM if in horizontal mode. In vertical mode when valid VOP pixel data is present on the input data bus, it is stored in location *BUFFER[N\_buff<sub>i,r</sub>]* in the next clock cycle ( $i \in \{A, B\}$ ). The 4-bit register *N\_buff<sub>i,r</sub>* is also incremented by 1 in the same cycle, which represents the number of VOP pels in *BUFFER* (i.e. the vertical  $N$  value). In this way vertical packing is done without redundant shift cycles. In horizontal mode a simple FSM is used to address the TRAM, using the  $N$  values already parsed in the vertical process to minimise the number of accesses. The same scheme ensures horizontal packing is done without redundant shift cycles.

The ACL effectively provides logic to compute the SA-DCT data alignment requirements as well as the row/column shape parsing. The serial loading of data from memory and SA-DCT alignment and shape parsing is achieved without needing more than the number of clock cycles for a conventional  $8 \times 8$  DCT that does not require alignment or shape parsing. By using the interleaved addressing scheme, register data is only switched when necessary eliminating unnecessary power consumption.

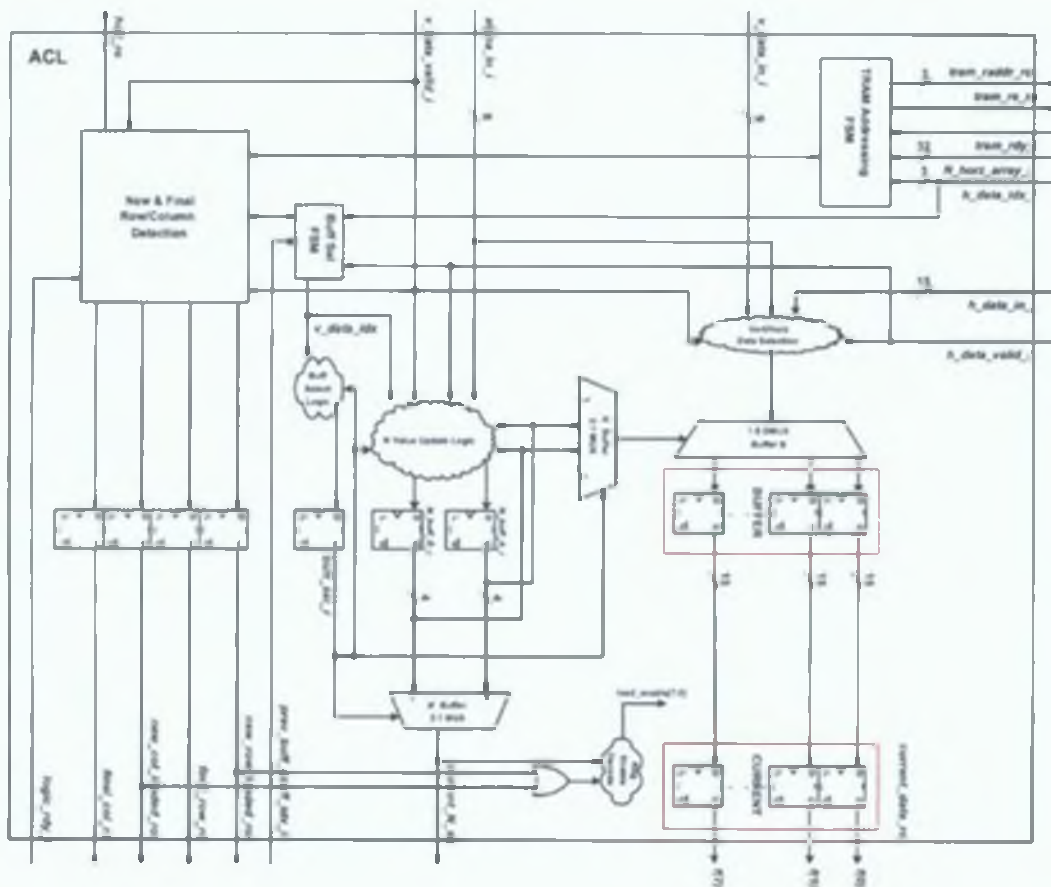


Figure 3.26: Addressing & Control Logic (ACL) Architecture

The behaviour of the ACL module is illustrated by the example timing diagram in Figure 3.27. In this diagram, the signal names correspond to the signals and registers labelled in Figure 3.26. The example shows the ACL module operating in vertical mode, hence it is reading pixel and alpha data from the external data bus. The pixel and alpha blocks shown in Figure 3.26 are being read serially in a column-wise manner. The  $v\_data\_valid_j$  signal is asserted without interruption implying that the SADC architecture can handle a constant stream of new data presented at the input ports. In clock cycle 0, the final data value in input column 0 is presented at the input ports. The "BuffSel FSM" in Figure 3.26 detects this final column data condition, referred to as "condition C" for clarity. Condition C causes  $buff\_sel_j$  to be inverted, as clear from in clock cycle 1. Condition C coupled with the negative transition on  $buff\_sel_j$  also causes  $N\_buff\_B_j$  to be reset to 0 (note that a positive transition causes  $N\_buff\_A_j$  to be reset to 0). In cycle 2,  $CURRENT$  is updated with the newly ready column 0 pixel data in  $BUFFER$ . Note that since  $current\_N\_so$  is equal to 1, only one of the registers in the  $CURRENT$  register bank is actually clocked. Also in cycle 2  $new\_col\_loaded_m$  is pulsed to indicate to the variable  $N$ -point DCT datapath circuit that column 0 has been loaded and automatically aligned in  $CURRENT$  and is ready for processing with its  $N$  value stored in  $N\_buff\_A_j$  ( $N = 1$ ). The vertical DCT circuit then reads the data from  $CURRENT$  and the associated  $N$  value from port  $current\_N\_so$ . In clock cycle 1 since the  $alpha\_in_j$  is 255 (i.e. part of the VOP), the data DB0 on  $v\_data\_in_j$  is stored in  $BUFFER$  in cycle 2 at location



$BUFFER[N\_buff\_B_r]$ , and the value in  $N\_buff\_B_r$  is also incremented by 1. This process continues in a similar manner until in clock cycle 5 where  $alpha\_in\_1$  is 0 (i.e. not part of the VOP). Therefore in cycle 6 the corresponding data DB4 is not stored (since it does not form part of the VOP) and  $N\_buff\_B_r$  is not incremented. In cycle 6  $alpha\_in\_1$  is 255 indicating that DB5 is part of the VOP so requires storage in cycle 7. However since  $N\_buff\_B_r$  was not incremented in cycle 6, DB5 is stored in the next  $BUFFER$  location after the previous VOP pixel, which is DB3. It is clear that no actual circuit register shifting takes place but the data is stored in the appropriately packed manner. Column 1 contains 6 VOP pixels and only 6 registers in  $BUFFER$  get switched. Registers  $BUFFER[7:6]$  are not unnecessarily switched and hence no unnecessary energy is consumed. Although these registers now contain dead data, the subsequent DCT circuitry will know this based on the value of  $N\_buff\_B_r$  and will not access them. In clock cycles 10–17 the DCT processor pipeline has time to process column 1. Also during this interval column 2 is stored in  $BUFFER$  and in this case all 8 pixels form part of the VOP. This example illustrates how the ACL performs the SA-DCT shifting steps efficiently without any explicit shifting cycles by exploiting the shape information.

### 3.5.2 Transpose Memory

The TRAM in Figure 3.28 is a 64-word  $\times$  15-bit RAM that stores the coefficients produced by the datapath when computing the vertical 1D transforms. These coefficients are then read by the ACL in a transposed fashion and horizontally transformed by the datapath yielding the final SA-DCT coefficients. As is clear from Figure 3.28, when storing data the index  $k$  is manipulated to store the value at address  $8 \times k + NRAM[k]$ , where  $NRAM[k]$  is routed to  $N_k\_curr$  in Figure 3.28. In this way, when an entire block has been vertically transformed, the TRAM has the resultant data stored in a horizontally packed manner with the horizontal  $N$  values ready immediately without shifting. Then  $NRAM[k]$  is incremented by 1. The 8 horizontal  $N$  values are implemented as a bank of 3-bit counters (NRAM in Figure 3.28). When an entire block has been stored in the TRAM, there are 8 values in the NRAM that represent the  $N$  values for each of the transpose rows, and these values are used by the ACL to minimise TRAM reads.

To illustrate the behaviour of the proposed TRAM, consider the example shown in Figure 3.29. The left block shows a vertically aligned block of SA-DCT coefficients as produced by the vertical variable length transform module. Each column of coefficients are computed in turn and each column coefficient  $k$  is produced serially. The SA-DCT requires these coefficients to be re-aligned as shown in the middle block in Figure 3.29 before horizontal transformation of each of the rows. To avoid explicit shifting of data to achieve this re-alignment, the proposed TRAM exploits coefficient index  $k$  and the set of 8  $NRAM[k]$  counters as described previously. To illustrate this behaviour with timing diagrams is too unwieldy, but the general concept can be illustrated by Figure 3.30. This diagram shows coefficient  $k = 2$  of column 3 being stored in the TRAM. Using the formula  $8 \times k + NRAM[k]$ , the coefficient is stored in the aligned TRAM address without explicit shifting. This addressing scheme makes it easier for the ACL to read data row-wise from the TRAM since the data is nicely aligned, as shown in the rightmost block in Figure 3.29. The ACL uses the  $N$  values in the NRAM to limit TRAM accesses to VOP data only. The TRAM proposed in this thesis is similar to the auto-aligning TRAM proposed by Lee et al. [203]. However, the scheme by Lee et al. partition their TRAM into 4 banks to support their two data per cycle read/write processing scheme. The TRAM proposed by the author is simpler in that it only uses one bank since only one read or write is ever required in a single cycle.

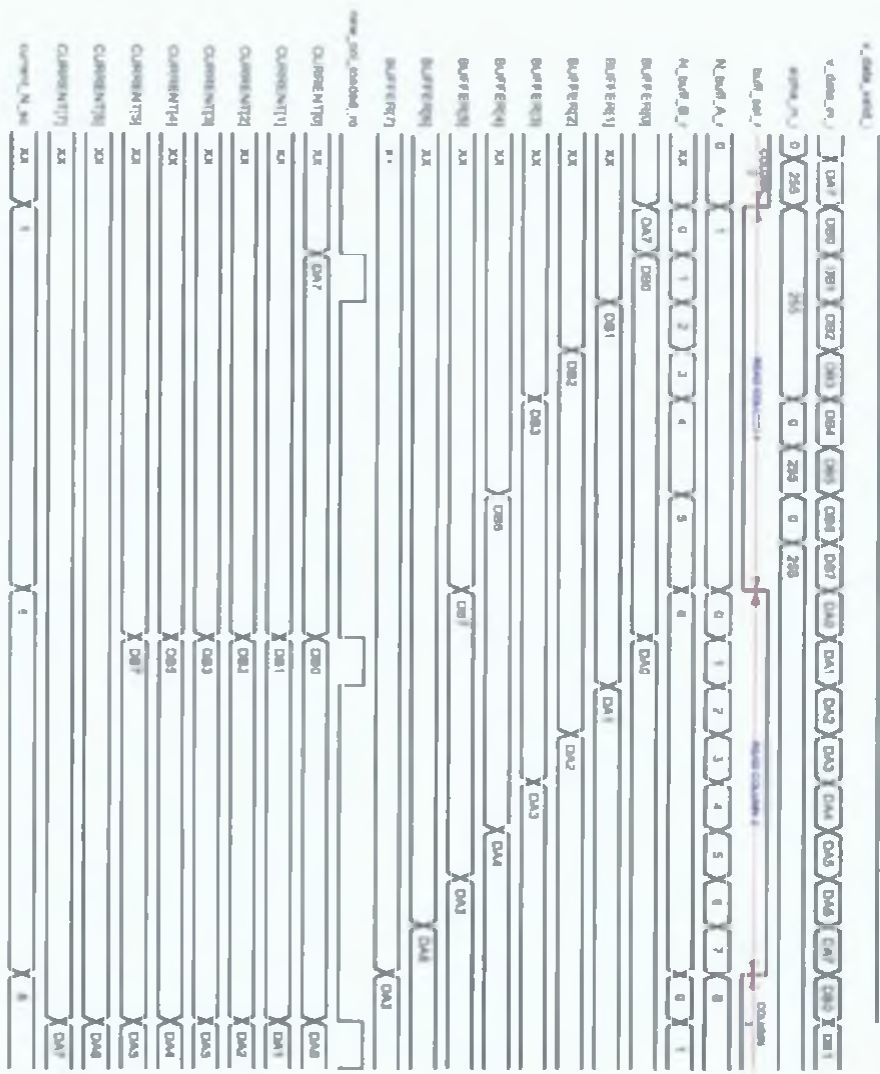
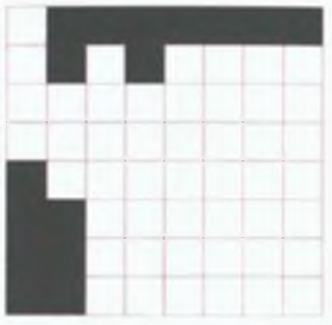
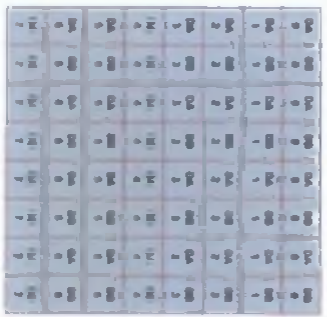


Figure 3.27: Sample SA-DCT ACL Timing Diagram

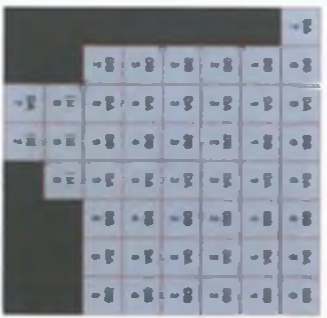


Sample Alpha Block

- ▶ column 0 ( $N = 1$ )
- ▶ column 1 ( $N = 6$ )
- ▶ column 2 ( $N = 8$ )
- ▶ column 3 ( $N = 8$ )
- ▶ column 4 ( $N = 7$ )
- ▶ column 5 ( $N = 6$ )
- ▶ column 6 ( $N = 6$ )
- ▶ column 7 ( $N = 6$ )



Sample Pixel Block



Vertically Aligned Pixel Block



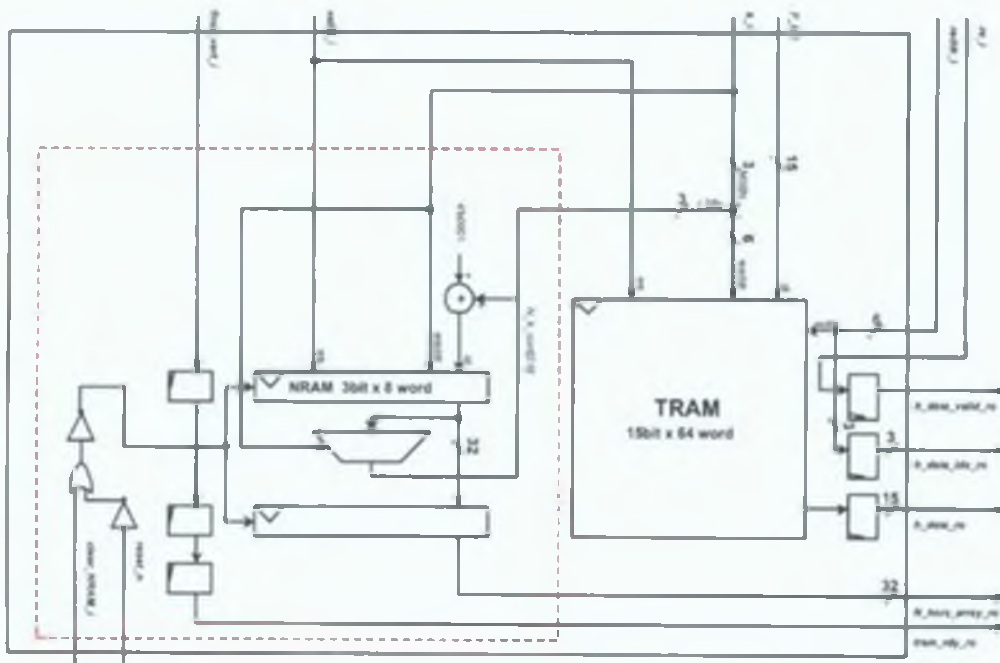


Figure 3.28: TRANSPOSE Memory (TRAM) Architecture

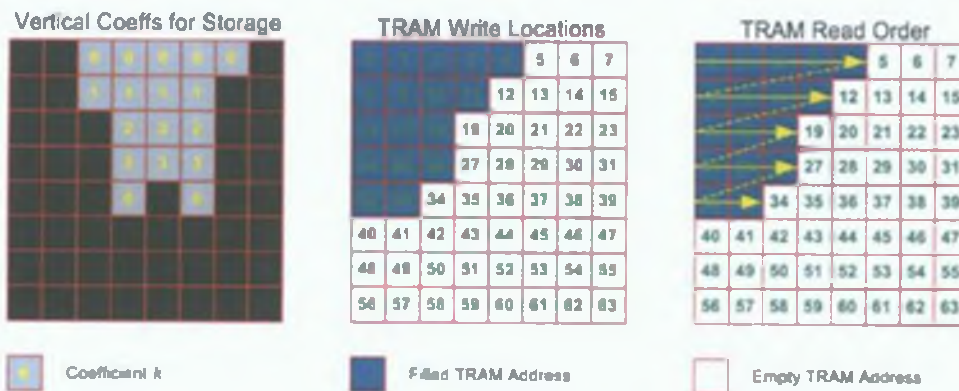


Figure 3.29: Example SA-DCT TRAM Addressing

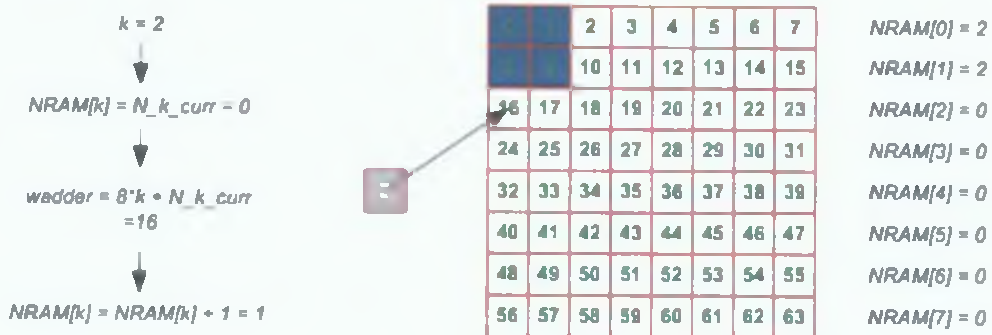


Figure 3.30: Example SA-DCT TRAM Write

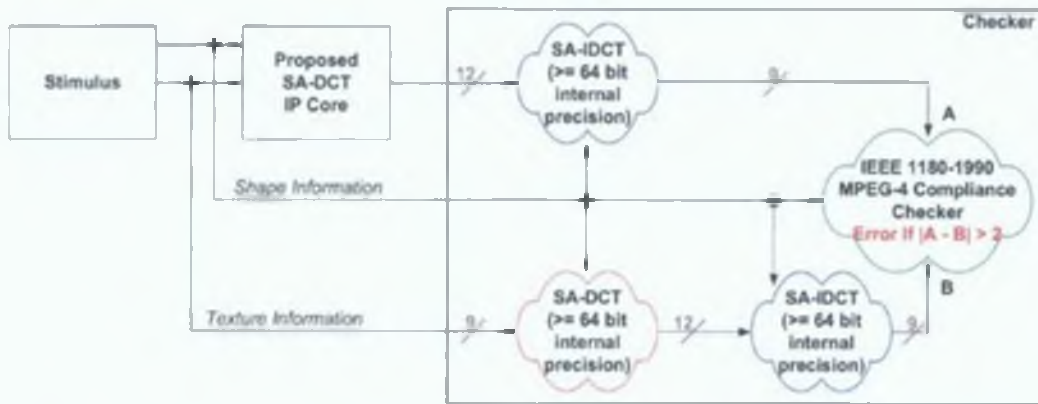


Figure 3.31: SA-DCT IP Core Verification Platform

## 3.6 Experimental Results

### 3.6.1 Fixed-Point Variable $N$ -point 1D DCT Design

The SA-DCT core presented in this thesis has two sources of round-off error inherent in its behaviour, namely the DA precision  $|P + 1 - Q|$  (see Section 3.2.1.7) and the intermediate coefficient precision  $T = 11.f$  (see Section 3.4.3). The DA precision in this case represents the number of bits used to represent the cosine constants of the variable  $N$ -point 1D DCT basis functions in Figure 3.3. Since cosine values are guaranteed to be in the range  $[-1, 1]$ , the value of  $P = 0$  in Equation 3.6 introduces no error. The mathematical properties of a 1D  $N$ -point DCT imply that 11 bits is enough to capture the integral part without any loss incurred [142]. Since the cosine functions are theoretically infinitely precise, rounding the fractional part of the result to  $T = 11.f$  bits introduces loss.

The values of  $Q$  and  $f$  are design parameters. Since both increase the circuit area the larger they are, the goal is to keep these parameters as low as possible according to design constraints. For MPEG-4 standard compliance, the SA-DCT implementation must satisfy the IEEE 1180-1990 [173] standard with some simplifications as specified by the MPEG-4 standard [66] as summarised in Figure 3.31. The coefficients are said to be compliant if, when reconstructed to pixels using the inverse transform, they are within two grey levels of the reference reconstructed data produced by a double precision floating point SA-DCT/SA-IDCT process. The test vectors used were generated by the random number generator specified by the IEEE 1180-1990 standard [173]. This generator produces pixel data with values in the range  $[-L, H]$ . The MPEG-4 standard specifies that 1 million  $8 \times 8$  random blocks should be generated for each of the following configurations:  $(L=256, H=255)$ ,  $(L=H=5)$  and  $(L=384, H=383)$ . The same stimuli are used with the sign changed on each pixel value and the tests are re-run.

With these stimuli, experimentation has been carried out using various values of  $Q$  and  $f$  for the proposed architecture. It has been found that MPEG-4 precision requirements are met with  $Q = -12$  (13-bit cosine constants according to Equation 3.6) and  $f = 4$  ( $11.4 = 15$ -bit transpose memory). Table 3.3 presents the results of each standard experiment for  $Q = -12$  and  $f = 4$ . The results for each experiment include the overall mean square error (OMSE) and the Peak Signal to Noise Ratio (PSNR). The PSNR is an objective measure of image quality, and is calculated in this case on the basis of the mean square error between the fixed-point reconstructed image and the reference reconstructed image

Table 3.3: SA-DCT Core IEEE 1180-1990 Results for  $Q = -12, f = 4$

$[-L, H]$	Sign	OMSE	PSNR [dB]	Sign	OMSE	PSNR [dB]
[-256,255]	+1	0.586	50.449	-1	0.586	50.449
[-5,5]	+1	0.551	50.720	-1	0.551	50.720
[-384,383]	+1	0.320	53.075	-1	0.320	53.075



Figure 3.32: Sample Object Reconstruction with  $Q = -10$  and  $T = 11.0$

[45].

As  $Q$  and  $f$  reduce from -12 and 4 respectively, less hardware is required but the standard requirements are violated. However, the reconstructed pixels do not deviate significantly from the required accuracy. For example, with the “akiyo” test sequence with  $Q = -10$  and  $T = 11.0$ , only 0.9333% of the VOP pixels reconstructed violate the standard and the maximum error is only a difference of 6 grey-levels. It is clear from Figure 3.32 that this degradation, even though it violates the MPEG-4 standard requirements, only has a minor effect on the perceptual quality of the reconstructed object. A user may tolerate such degradation on a mobile platform if it means a longer battery life. Future work in this area will be to investigate the relationship between the power gains achievable with coarse precision arithmetic and perceptually degraded video quality.

### 3.6.2 Evaluation Against Prior Art

The SA-DCT architecture has been implemented in Verilog and synthesised targeting three technologies: a  $0.09\mu\text{m}$  low power standard cell ASIC library by Taiwan Semiconductor Manufacturing Company (TSMC), a Xilinx Virtex-II FPGA and a Xilinx Virtex-E FPGA. A SystemC flow was also followed but due to the discontinuation of the “SystemC Compiler” tool by Synopsys, this flow was abandoned. The ASIC flow is necessary for benchmarking against prior art architectures, and the FPGA flows are used for conformance testing with the MPEG-4 standard requirements and embedded system prototyping. The ASIC synthesis results are summarised in Table 3.4. The SA-DCT architecture has been physically implemented on the two FPGAs listed as part of two integration frameworks. The wrapper nature of both frameworks make it straightforward to migrate any general IP core between platforms since the wrappers shield it from platform specific protocols. The FPGA frameworks and synthesis results are discussed in Section 3.6.3 and Section 3.6.4.

Table 3.4: SA-DCT 90nm TSMC Synthesis Results and Benchmarking

Architecture	Process [ $\mu\text{m}$ ]	Cycle Count			+	*	Gates	PGCC	Speed [MHz]	
		General	Max	Min						
Le[197]	◇	0.7	$N + 2$	10	3	11	5	n/a	n/a	40
Tseng[201]	★	0.35	n/a	n/a	n/a	n/a	n/a	n/a	n/a	66
Chen[109]	◇	0.35	n/a	119	14	1	0	3100	$3.7 \times 10^5$	83
	★		n/a	1904	224	n/a	n/a	5715	$1.1 \times 10^7$	
Lee[203]	◇	0.35	n/a	11	3	11	4	17341	$1.9 \times 10^5$	66.7
Proposed Approach	◇	0.09	$N + 2$	10	3	27	0	12016	$1.2 \times 10^5$	556
	★		n/a	142	79	31	0	25583	$3.6 \times 10^5$	

Key: ◇ ⇒ Datapath Only, ★ ⇒ Datapath & Memory

### 3.6.2.1 Area and Speed

As discussed in detail in Chapter 2, it is difficult to compare designs implemented with different technologies based on operating frequency or power consumption. Technology independent metrics must be used or else some normalisations are required for meaningful analysis. This section attempts to compare the SA-DCT architecture proposed in this thesis against prior art in terms of speed and area (power and energy are dealt with in Section 3.6.2.2).

When benchmarking circuits in terms of speed and area, metrics such as maximum operating frequency, throughput (pixels/second) and circuit die area depend on the implementation technology so are difficult to use directly. Gate counts coupled with processing cycle latency offer better benchmarks since they are technology independent. The corresponding values for the prior art and the proposed architecture are summarised in Table 3.4 (also presented in [208]). Synthesising the design with Synopsys Design Compiler targeting TSMC 0.09 $\mu\text{m}$  TCBN90LP technology yields a gate count of 25583. The area of the variable  $N$ -point 1D DCT datapath is 12016 (excluding TRAM memory and ACL). Both gate counts are used to facilitate equivalent benchmarking with other proposals based on the information available in the literature. The results show the proposed design is an improvement over Lee [203] and offers a better trade-off in terms of cycle count versus area compared with Chen [109], as discussed subsequently. Area and power consuming multipliers have been eliminated by using only adders – 27 in total – divided between the EOD module (4), the MWGM (11) and the PPST (12). Using the estimation that a multiplier is equivalent to about 20 adders in terms of area, the adder count of the proposed architecture (27) compares favourably with Le [197] ( $5 \times 20 + 11 = 111$ ) and Lee [203] ( $4 \times 20 + 11 = 91$ ). This is offset by the additional MUX overhead, but as evidenced by the overall gate count figure of the proposed architecture, it still yields an improved circuit area. By including the TRAM (1) and the ACL controller (3), an additional 4 adders are required by the entire proposed design. In total, the entire design therefore uses 31 adders and no multipliers.

The worst case clock cycle latency of the SA-DCT core is 142 cycles, and this latency is incurred when processing an  $8 \times 8$  opaque block (an internal block in the VO). For boundary blocks the latency is  $142 - (64 - \text{opaque\_pels})$ , where *opaque\_pels* is the number of VO pels. An empty block can be determined in 64 cycles, the time it takes to parse the alpha block. Hence when working out the real-time processing constraints for the proposed SA-DCT module, the worst case latency of 142 cycles must be used. CIF resolution with a frame rate of 30 fps requires  $71.28 \times 10^3$  blocks to be processed per second.

This translates to a maximum block processing latency of approximately  $14\mu\text{s}$ . Given that the proposed SA-DCT module requires at worst 142 cycles to process a block, this means that the longest allowable clock period for the proposed SA-DCT module is approximately 98.8ns. This translates to a minimum operating frequency of about 11MHz. Table 3.4 shows that the maximum achievable frequency with TSMC 0.09 $\mu\text{m}$  TCBN90LP technology is 556MHz, so the proposed SA-DCT core can comfortably meet real-time constraints. Running at 556MHz, the proposed core can process a single  $8 \times 8$  block in 256ns. At this frequency, the design is capable of a throughput of 250 Mpixel/s. As discussed in Chapter 2, the design should run as slow as possible (11MHz) for low power consumption, and the maximum throughput in this case is 5Mpixel/s.

In Chapter 2, the *product of gate count and computation cycle* (PGCC) was proposed as a good metric that combines speed and area properties for benchmarking competing circuit architectures that implement the same task (in this case the SA-DCT). As evidenced by the results in Table 3.4, the proposed design represents an improvement in terms of PGCC compared to prior art in the cases where benchmarking is possible. An “n/a” field in Table 3.4 indicates that this value is not available from the literature, or is not appropriate. These results indicate that the proposed SA-DCT architecture strikes a better balance between area and speed compared to the prior art.

### 3.6.2.2 Power Consumption and Energy Dissipation

Comparing hardware accelerators meaningfully in terms of their power consumption properties is difficult to achieve in practice. To ensure a fair comparison, competing architectures must be compared using the same synthesis tools and configurations. In practice this is difficult to achieve due to the general unavailability of HDL source code for competing architectures published in the literature. Some approximate normalisations must be used to aid benchmarking from the available parameters generally quoted in the literature (power, voltage, frequency and process technology). The benchmarking outlined in this section is based on the normalisation formulae outlined in Section 2.3.2.5 of Chapter 2). To analyse the power consumption of the proposed SA-DCT IP core, the method described in Section 2.3.2.4 is followed. In summary this involves a back-annotated dynamic simulation of the gate level netlist. This netlist is generated by the synthesis tool for TSMC 0.09 $\mu\text{m}$  TCBN90LP technology. The Synopsys Prime Power tool is used to analyse the annotated switching information from a VCD file.

As discussed in Chapter 2, the power consumption of a circuit is heavily dependent on the nature of the input stimulus. In the case of the SA-DCT, typical input stimulus are readily available from the standard video test sequences (and segmentation masks) defined by the MPEG-4 Video Verification Model [209]. The sequences available to the author are listed in Table 3.5. Since running power analysis is a slow process, only 5 of these sequences were used for power analysis (indicated by an asterisk in Table 3.5). The sequences chosen are semantically meaningful objects – they represent the kind of objects that may be of interest to a user in an object-based MPEG-4 application. It is argued that objects that represent the background (such as the water) are not as important, and given that power analysis is a slow process these are not included in the analysis presented in this thesis. In addition to these 5 sequences, an additional test sequence was generated by the author, called “Andy”, that simulates video telephony where the semantic object is the human face. The segmentation mask was generated using a simple colour thresholding segmentation algorithm [64]. Power analysis was carried out for each of the 6 test sequences. The power waveforms for each of these sequences are shown in Figure 3.33 to



Table 3.5: Available MPEG-4 Standard Test Sequences

Test Sequence	Description	Test Sequence	Description
Akiyo	Head and Shoulders	Sean*	Head and Torso
Container Object 0	Water	Coastguard Object 0	Water
Container Object 1*	Big Boat	Coastguard Object 1*	Big Boat
Container Object 2	Small Boat	Coastguard Object 2	Small Boat
Container Object 3	Near Shore	Coastguard Object 3	Embankment
Container Object 4	Far Shore & Sky	News Object 0	TV Set Background
Container Object 5	Flag	News Object 1	Screen
Hallmonitor Object 0	Office Background	News Object 2	Newsreaders
Hallmonitor Object 1	Walking Person 1	News Object 3	Text Logo
Hallmonitor Object 2*	Walking Person 2	Weather*	Head and Torso

Table 3.6: SA-DCT 90nm TSMC Power Consumption

Test Sequence	Average Power [mW]
Andy	0.340
Coastguard Object 1	0.341
Container Object 1	0.359
Hall Monitor Object 2	0.307
Sean	0.412
Weather	0.390
Random Data	0.519

Figure 3.38. It is clear that the power consumption varies in proportion to the size of the object. For example, as the person in the “hall monitor” test sequence approaches the camera, the power increases until it suddenly drops when the person leaves the scene through a door. This is expected since the proposed SA-DCT architecture terminates processing earlier for smaller shape blocks. The average power for each simulation is given in Table 3.6, and the average power figure over all 6 test sequences is 0.36mW.

Another set of power analysis simulations were run using random data as stimulus. In the context of an SA-DCT hardware architecture, it is argued that random data simulations generate an estimate for worst case power consumption. This is because unlike natural video data, there is a very low probability of adjacent pixel or shape correlation with random data so there will be a lot of switching. To get an estimate for worst case power, 10 simulations were run using 50 random pixel and alpha  $8 \times 8$  block pairs as stimulus, with different seeds used for the random number generators to guarantee 10 unique data sets. The power figure quoted in Table 3.6 is the average value over the 10 random data simulations. As expected, the value of 0.519mW is slightly higher compared to 0.36mW, which is the average power consumption over all the video test sequence simulations.

Only two of the SA-DCT implementations in the literature quote power consumption figures and the parameters necessary against which to perform normalised benchmarking: the architectures by Tseng et. al. [201] and Chen et. al. [109]. The normalised power, energy and Energy-Delay Product (EDP) figures are summarised in Table 3.7. Note that the energy figures quoted in the table are the normalised energies required to process a single opaque  $8 \times 8$  block. Results show that the proposed SA-DCT

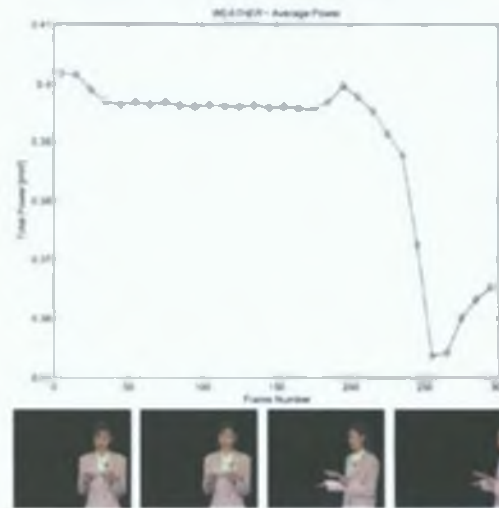
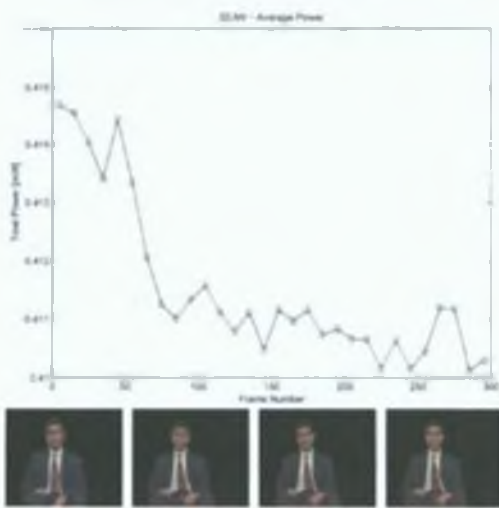
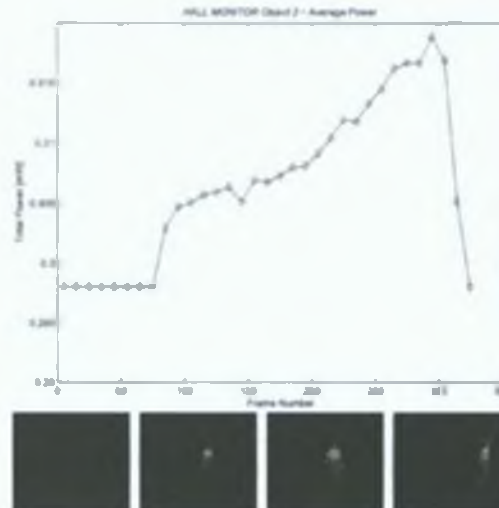
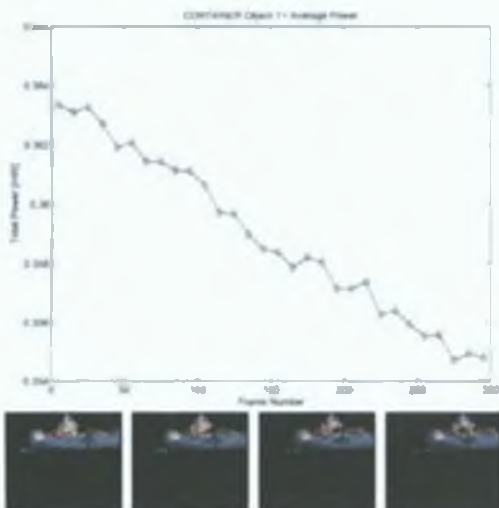
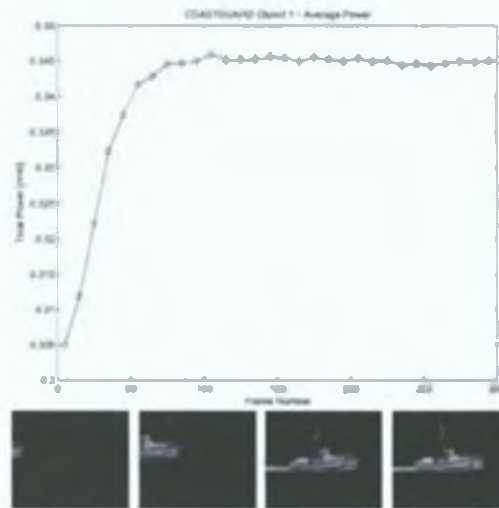
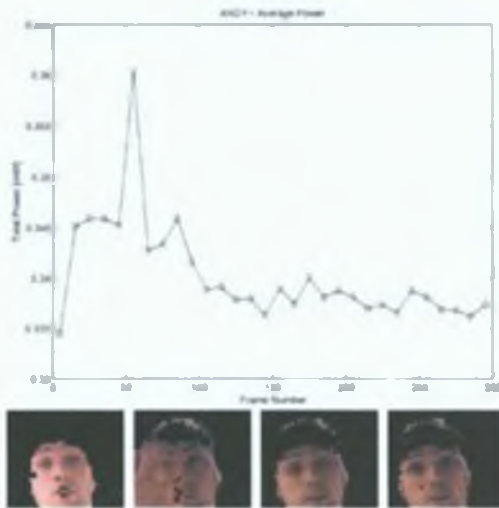


Table 3.7: Normalised SA-DCT Power and Energy Benchmarking

	Process	Voltage	Power	Power (VL)	Delay	Speed	Power (VLf)	Energy	EDP
	[ $\mu\text{m}$ ]	[V]	[mW]	[mW]	[Ckcs]	[MHz]	[mW]	[nJ]	[pJs]
Tseng[201]	0.35	3.3	180.00	180.00*	n/a	66	180.00*	n/a*	n/a*
Chen[109]	0.35	1.2	12.44	12.44*	1904	25	12.44*	285.37*	6.55*
Proposed Approach	0.09	1.2	0.36	15.00*	142	11	15.00*	193.28*	2.50*
				8.17*			8.17*	105.42*	1.37*

architecture compares favourably against both the Tseng and the Chen architectures. The normalised energy dissipation and EDP figures are the most crucial in terms of benchmarking, since the energy dissipated corresponds to the amount of drain on the battery and the lifetime of the device.

It must be conceded that despite these normalised benchmarking figures, the validity of these figures is still open to debate. This is because the power figures for each of the target technologies are obtained using different power estimation tools and configurations. Since the switching activity in a module is dependent on its data load (and this is particularly true for SA-DCT), the same testbench should really be used for fair comparisons when generating VCD files.

### 3.6.2.3 Benchmarking Power & Energy Against Software

Another interesting power and energy benchmarking experiment is to compare the power consumption and energy dissipation of the proposed SA-DCT hardware accelerator against the MPEG-4 Part 7 optimised software implementation of the SA-DCT [210]. Experiments targeting two processors have been conducted. Clearly, the figures should be much better for the hardware otherwise there is no benefit to the hardware accelerator in terms of power and energy.

The first approach involved taking a physical measurement of the current drawn by an ARM920T processor [211], a typical embedded processor for mobile devices. The ARM prototyping platform used is described in more detail in Section 3.6.4. The SA-DCT C language software implementation was isolated without modification from the MPEG-4 part 7 optimised software codec and cross-compiled targeting the ARM920T. This C function was wrapped in a long loop in a simple program that was then run on the ARM920T. Because of the loop, the current drawn by the processor is periodic and hence can be used to self-trigger an oscilloscope. Using the same setup as [97], a period of this current waveform was recorded and analysed in MATLAB. The power waveform for a single SA-DCT iteration processing an opaque  $8 \times 8$  block is shown in Figure 3.39, and the average power is approximately 15mW. Using the same power and energy normalisations as before, the results presented in Table 3.9 clearly show that the SA-DCT hardware architecture proposed in this thesis is much more efficient compared to a software implementation running on that ARM920T processor.

The second approach leverages the web-based JouleTrack tool [103, 104]. JouleTrack is an online tool for obtaining power and energy statistics for C programs running on a StrongARM SA-1100 processor, another typical embedded processor. The simple program as described earlier was uploaded to JouleTrack executing 1000 iterations of the SA-DCT algorithm, and the results are shown in Table 3.8. The profiling results for the different processor cycle modes as well as a breakdown of leakage and switching power are illustrated in Figure 3.40. As the results in Table 3.9 illustrate, the proposed hard-

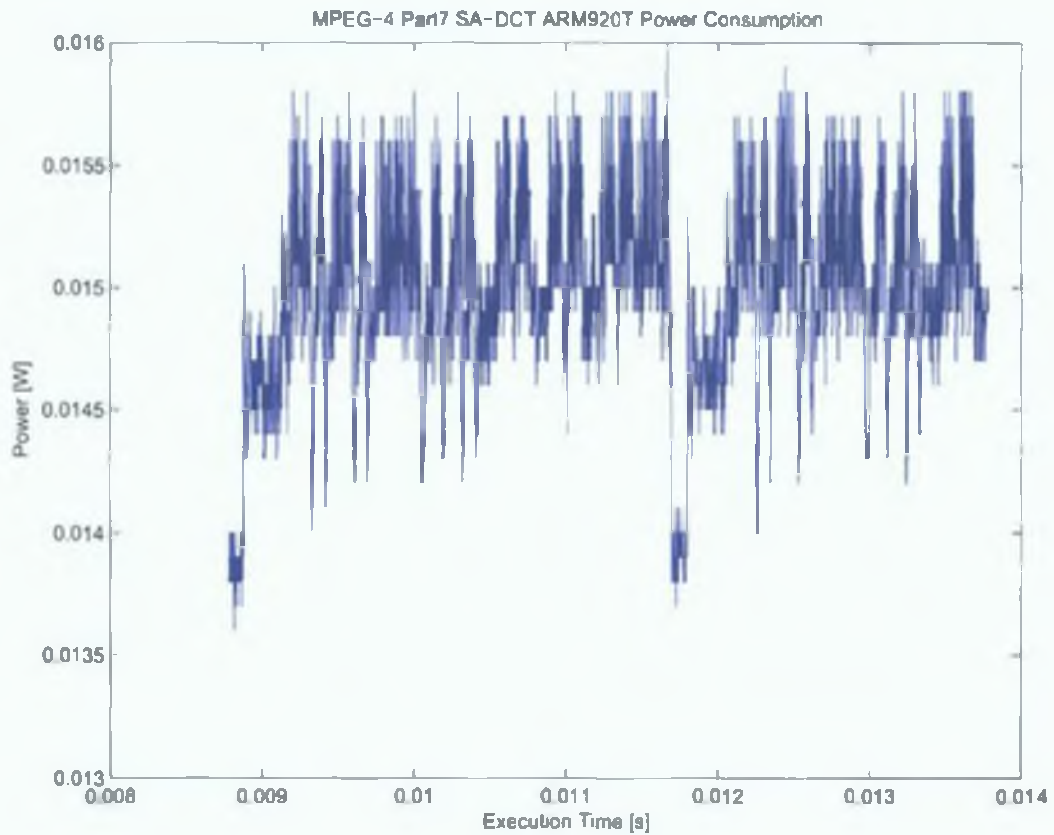


Figure 3.39: MPEG-4 Part7 SA-DCT ARM920T Power Consumption

ware SA-DCT architecture is much more efficient compared to the reference software implementation executing on the SA-1100 processor.

It must be conceded that although the SA-DCT software implementation used for these experiments is the official MPEG-4 Part 7 optimised software, it has not been optimised for an embedded processor like the ARM920T or the StrongARM SA-1100. Hence, it is likely that the results presented in this section could be improved if processor specific optimisations were made to the software. For example, Sloss et. al. provide an efficient C programming guide specifically for ARM processors [120]. However, power efficient software implementation is not the focus of this thesis. Even allowing for improved software power results, the results presented in this section validate the energy efficiency of SA-DCT architecture proposed in this thesis compared to a software implementation.

Table 3.8: Software SA-DCT StrongARM SA-1100 Energy Statistics

<b>Operating Frequency</b>	206 MHz
<b>Operating Voltage</b>	1.5 V
<b>Execution Time</b>	433.223 $\mu$ s
<b>Average Switching Current</b>	0.2041 A
<b>Average Leakage Current</b>	0.0330 A
<b>Total Energy</b>	154.02 $\mu$ J
<b>Average Power</b>	356 mW



Figure 3.40: MPEG-4 Part 7 Fast SA-DCT Energy Characteristics on StrongARM SA-1100

Table 3.9: Normalised Power and Energy Benchmarking Against Software SA-DCT

	Process [ $\mu\text{m}$ ]	Voltage [V]	Power [mW]	Power ( $V, L$ ) [mW]	Delay [ $\mu\text{s}$ ]	Speed [MHz]	Power ( $V, L, f$ ) [mW]	Energy [nJ]	EDP [pJs]
ARM920T	0.25	2.5	15.00	15.00*	3000*	140	15.00*	$45 \times 10^{-3}$ *	$135 \times 10^{-3}$ *
SA-1100	0.35	1.5	356.00	356.00*	433*	206	356.00*	$154 \times 10^{-3}$ *	$67 \times 10^{-3}$ *
Proposed Approach	0.09	1.2	0.36	5.79*	13*	11	5.79*	75*	0.96*
				6.80*	13*		6.80*	88*	1.13*

### 3.6.3 Conformance Testing

The MPEG-4 reference hardware initiative ('Part 9') is a working group dedicated to proposed VLSI architectures for the most computationally demanding tools in the MPEG-4 standard. All proposed architectures must pass conformance tests that validate that the bitstream produced by the hardware accelerator is identical to that produced by the MPEG-4 reference software [212]. For the proposed SA-DCT implementation, the MPEG-4 reference software was compiled with the SA-DCT software replaced by a user defined API call to the SA-DCT hardware accelerator residing on an FPGA. Further details on the MPEG-4 conformance platform (proposed by the University of Calgary) may be found in [213] and details on the proposed SA-DCT integration process (including hardware and software APIs) may be found in [214, 215, 216].

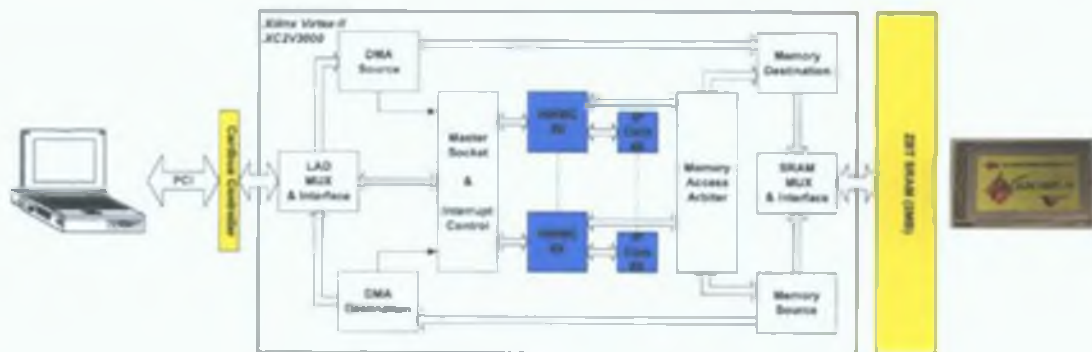


Figure 3.41: MPEG-4 Conformance Integration Hardware

### 3 6 3 1 Conformance Hardware System

The SA-DCT module has been integrated with an adapted version of the multiple IP core hardware accelerated software system framework developed by the University of Calgary (UoC) [217]. The hardware is implemented on an Annapolis WildCard-II PCMCIA FPGA prototyping platform [218], and a system block diagram is shown in Figure 3.41. The MPEG-4 software runs on a host laptop Pentium4 processor and transfers data to and from the 2MB SRAM on the WildCard-II. This is done across the PCI bus via a DMA controller. Integrated IP cores are memory mapped peripherals and are activated by the host software by writing configuration data to a specific address. This write is detected by the master socket block and a strobe is issued by the interrupt controller to the specific IP core hardware module controller (HWMC).

When integrating a new IP core with the UoC framework, a HWMC specific to the IP core is required to control activation of the IP core and memory transfers to and from it. The HWMC shields the IP core from the WildCard-II platform specific interface signals (PCI and SRAM) making it easy to port the core to another platform (e.g. ARM based in as Section 3.6.4). The SA-DCT has a variable load depending on the shape information and the SA-DCT core has been designed such that it finishes processing smaller video object plane (VOP) blocks earlier. The SA-DCT HWMC has been designed with this in mind so that the SA-DCT relinquishes control of the SRAM at the earliest possible moment. Due to the nature of the SA-DCT computation, the IP core will not respond with output data until it has sampled all 128 bytes (32 DWORDs) of the input texture and alpha blocks (64 bytes each). Hence the SA-DCT HWMC performs a burst read of 32 DWORDs and will not need to write back until after this transaction is complete. The latency between the last DWORD read and the final IP core response is 142 cycles. Performance could be enhanced by some prefetching and buffering of the next data block while waiting for the IP core to respond for the current block and this will be investigated in future work. The WildCard-II SRAM is single port so does not support dual write and read accesses, but if another system with dual port memory is targeted, the SA-DCT HWMC could be modified to operate with two parallel state machines to further speed up processing.

### 3 6 3 2 MPEG-4 Reference Software API

The SA-DCT hardware accelerator has been integrated with the publicly available MPEG-4 optimised reference software (Part 7) [210], as required to validate functionality and pass conformance testing. For conformance testing of hardware, ideally the software should require minimal modification for rapid verification. For the SA-DCT accelerator two minor modifications are required (see [214, 216] for more details). The first modification involves editing the configuration steps of the MPEG-4 software to also program the WildCard-II FPGA fabric with the SA-DCT accelerator, and allocate and bind the DMA pointers. This allows both the application software and hardware accelerator to read/write in the same area of the WildCard-II SRAM transparently. The second modification is in the *CFwdSADCT* C++ class in file *sadct.cpp*. A new pre-processor directive is added to allow the codec to use either the SA-DCT software algorithm or SA-DCT accelerator on the WildCard-II.

It would be more efficient to send unprocessed data to FPGA SRAM in frame size bursts as opposed to initiating a transfer for each  $8 \times 8$  block. Although transparent to the HWMC which can be programmed for any amount of burst processing, such a setup requires more extensive modification of the

Table 3 10 SA-DCT FPGA Synthesis Results

Target	Module	Area [Gates]	$f_{max}$ [MHz]	Power [mW]		Throughput [Mpixel/s]		*	CLB Slices	Block RAMs
				$(f_{max})$	(11MHz)	$(f_{max})$	(11MHz)			
WildCard II Virtex II XC2V3000	SA DCT	39854	78.2	318.37	49.26	35.14	4.96	0	2605 (18%)	0
	SA DCT HWMC	4354	85.6	n/a	n/a	n/a	n/a	0	201 (1%)	0
	Mod UoC Fw	102085	77.6	n/a	n/a	n/a	n/a	0	1627 (11%)	1
Integrator/CP Virtex E XCV2000E	SA DCT	36088	47.2	352.55	91.41	21.27	4.96	n/a	2018 (10%)	0
	SA DCT HWMC	6053	81.0	n/a	n/a	n/a	n/a	n/a	283 (1%)	0
	ARM VS	7020	89.7	n/a	n/a	n/a	n/a	n/a	381 (1%)	0

MPEG-4 reference software Since the purposes of this platform is for conformance testing, the minimal modification approach is suitable since it allows designers to concentrate on the hardware accelerator development and facilitates rapid validation with the reference software

### 3 6 3 3 Conformance Results

End to end conformance has verified that the bitstreams produced by the encoder with and without SA-DCT hardware acceleration are identical The test vectors used were 39 of the CIF and QCIF object-based test sequences as defined by the MPEG-4 Video Verification Model [209] Synthesis results for the WildCard-II platform are shown in Table 3 10 The modified University of Calgary integration framework takes up only approximately 11% of the FPGA resources leaving almost 90% for hardware accelerators The SA-DCT along with its HWMC require 19% The post place and route timing analysis indicates a theoretical operating frequency of 78.2MHz so the IP core is able to handle real time processing of CIF sequences quite comfortably (assuming the same 11MHz constraint as before) Post place and route simulations were carried out on the netlist for subsequent VCD power analysis with the Xilinx XPower FPGA power analysis tool These simulations were run at 78.2MHz (high throughput mode) and at 11MHz (low power mode) For each mode, 10 simulations were run using 50 random coefficient and alpha  $8 \times 8$  block pairs as stimulus, with different seeds used for the random number generators to guarantee 10 unique data sets The power figures quoted in Table 3 10 are the average value over the 10 simulations for each mode As described previously, random data is useful in this case to estimate a value for worst case power consumption

### 3 6 4 System Prototyping

Although the WildCard-II platform is useful for rapid validation and conformance testing, the system itself does not represent a realistic architecture for a mobile embedded system Since it interfaces with the PCI bus it is more suitable for emulating hardware accelerators for desktop PC graphics cards For wireless embedded systems, the processor of choice is the ARM family, so the author proposes a plug and play "ARM virtual socket" prototyping platform built around an ARM processor and the AMBA bus architecture The ARM virtual socket has been implemented on an ARM Integrator/CP prototyping platform [219] with an ARM920T processor [211] running embedded Linux and a Xilinx Virtex-E FPGA [220] for AMBA IP core prototyping The platform facilitates the rapid prototyping of any number of "virtual component" hardware accelerators with AMBA interfaces

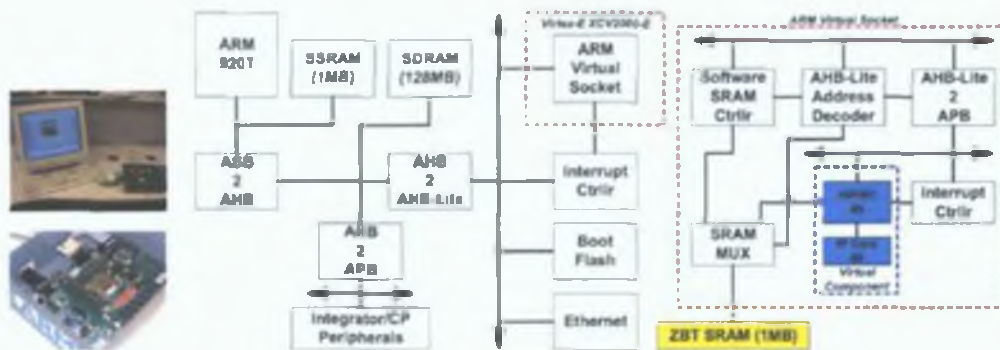


Figure 3.42: ARM Virtual Socket Platform

### 3.6.4.1 ARM Virtual Socket Hardware

A system block diagram showing the ARM virtual socket hardware (implemented on the Virtex-E FPGA) is shown in Figure 3.42. The virtual component block encompasses a single hardware accelerator IP core and its associated HWMC. In general there may be multiple virtual components residing on the AMBA Advanced Peripheral Bus (APB) but for diagram clarity only one is shown. The virtual component was designed as an APB peripheral since the APB bus is typically used for low power peripherals and has a simple interface. The Integrator/CP platform interfaces to the prototyping FPGA via the AHB-Lite bus as shown in Figure 3.42. The AHB-Lite protocol limits the bus to one master (in this case the AHB 2 AHB-Lite bridge in Figure 3.42). Consequently only AHB slaves may be implemented on the FPGA preventing the implementation of a DMA controller. The HWMC of a virtual component has addressable registers for configuring frame dimensions, frame burst length, SRAM start read address and SRAM start write address. When strobed by a software API call, the SA-DCT HWMC initiates SRAM transfers to and from the IP core and has similar behaviour to the HWMC developed for the WildCard-II platform (outlined in Section 3.6.3.1) but with an APB interface. Further details on the ARM virtual socket may be found in [216, 221].

### 3.6.4.2 ARM Virtual Socket Software API

A software API has been developed to allow transparent interaction for ARM compiled application software with virtual component hardware accelerators. The API provides a simple user interface for software developers targeting an ARM processor with virtual component hardware accelerators. Further details on the ARM virtual socket software API may be found in [216, 221].

### 3.6.4.3 Prototyping Results

The synthesis results in Table 3.10 illustrate that the SA-DCT IP core also meets the real time frequency constraint of 11MHz on the ARM virtual socket platform. The maximum possible operating frequency of the SA-DCT core on this FPGA is 47.2MHz. The ARM virtual socket platform itself has a much smaller equivalent gate count (7020) compared to the PCI-based conformance framework currently used by MPEG-4 Part 9 (102085). This leaves more space on the ARM platform's FPGA for prototyping



more IP cores together and the system architecture in Figure 3.42 is more representative of a real mobile multimedia system compared with the PCI based system in Figure 3.41. Post place and route simulations were carried out on the netlist for VCD power analysis. These simulations were run at 47.2MHz (high throughput mode) and at 11MHz (low power mode). For each mode, the random data configurations were used as described previously. The power figures quoted in Table 3.10 are the average value over the 10 simulations for each mode. These results show that in low power mode, the worst case power consumption is approximately 91.41mW.

### 3.7 Possible Architectural Variations

The SA-DCT architecture proposed in this thesis computes each coefficient serially and re-uses the same variable  $N$ -point 1D DCT processor for the vertical and horizontal processing steps, with a worst case clock cycle latency of 142 cycles. The experimental results outlined in Section 3.6 show that the architecture is suitable for implementation on a low power mobile platform, and it compares favourably with prior art. In a high throughput application where power may be less of an issue, 142 cycles may be too long. However, it is still possible to leverage a parallel version of the architecture proposed in this thesis for such an application.

Consider the architecture in Figure 3.12, which uses a serial computation scheme. The most obvious technique to reduce latency is to implement 8 parallel variable  $N$ -point 1D DCT processors. In this scenario the number of clock cycles required to compute an  $N$ -point DCT is 3 (the pipeline depth) as opposed to the  $N + 2$  cycles required by the serial scheme. This translates to an overall SA-DCT processing latency of 30 clock cycles (compared to 142 with the current serial scheme). However, this speed up comes at the cost of an circuit area increase of a factor of 8. Another possibility is to implement dedicated datapaths for both the vertical and horizontal DCT processing (i.e. 16 variable  $N$ -point 1D DCT processors). In this scenario, the worst case clock cycle latency is still 30 cycles, since the horizontal processing can not begin until the vertical processing is complete. However, the overall throughput of the module increases by a factor of 2 since when the module is processing the horizontal transforms for a certain  $8 \times 8$  block, it can begin the vertical transforms for the next  $8 \times 8$  block. The downside is that this throughput gain comes at the cost of the circuit area increasing by a factor of 16.

Since the focus of this thesis is low energy dissipation, the parallel SA-DCT architectures outlined here have not been thoroughly investigated. If a real system specification was provided with specific throughput and power requirements, an interesting research task would involve experimenting with varying degrees of parallelism in the SA-DCT architecture to achieve a "sweet-spot" architecture.

### 3.8 Summary of Contributions

This chapter has described the SA-DCT algorithm in detail with a discussion on the hardware implementation challenges caused by the additional processing steps involved over the classical  $8 \times 8$  DCT. A comprehensive survey of prior art implementations of the  $8 \times 8$  DCT is provided, followed by a detailed analysis of the prominent prior art related specifically to the SA-DCT. Based on this discussion, it is concluded that some of the prior SA-DCT implementations do not address all of the required processing steps, and none have been designed specifically with low energy in mind. Specifically, the Le [197]

Tseng [201] and Lee [203] architectures have multiplier-based datapaths and only the Lee architecture addresses all of the SA-DCT processing steps. Compared to these approaches, the Chen architecture [109] is more energy-efficient since it uses an adder-based datapath, but has a long computational latency and approximates odd  $N$  DCTs with the nearest even DCT. Following this discussion on prior art, the author's proposed SA-DCT architecture is described in detail. In Chapter 2 it was concluded that most energy savings are achievable at the high levels of design abstraction, so the proposed SA-DCT has been designed accordingly. The primary low energy features include minimal switching shape parsing and data alignment, data dependent clock gating, multiplier-free datapath (using adder-based DA), balanced delay paths and hardware resource re-use.

The chapter concludes by benchmarking the proposed SA-DCT architecture against the prior art where possible. In terms of area and latency, the PGCC metric shows that the proposed architecture outperforms the Chen [109] and Lee [203] architectures. In terms of normalised energy and power, the proposed architecture outperforms the Chen [109] and Tseng [201] architectures. The proposed SA-DCT has also undergone conformance testing via the MPEG-4 Part 9 initiative, validating that it is compliant with official MPEG requirements.

Future work on this topic could investigate different parallel variations and the trade-offs involved between processing latency and power consumption. The proposed architecture could be extended to include a  $\Delta$ DC-SA-DCT pre-processing module as required for MPEG-4 intra-VOP boundary blocks, and this is discussed in detail in Chapter 6.

## CHAPTER 4

# Shape Adaptive Inverse Discrete Cosine Transform Architecture

To complement the SA-DCT accelerator proposed in Chapter 3, an inverse transform architecture is necessary since both are required for a video encoder (see Figure 2.13) and the inverse transform is needed for the decoder (see Figure 2.5). This chapter surveys state of the art implementations for the IDCT/SA-IDCT proposed in the literature. A novel hardware implementation of the SA-IDCT is then proposed that offers a good trade-off between area, speed and power consumption. This architecture can be leveraged if implementing a mobile multimedia terminal that supports MPEG-4 object-based processing.

### 4.1 Introduction

The SA-IDCT has a similar algorithm to the SA-DCT discussed in Chapter 3, but it has some distinctive features that pose additional hardware implementation challenges. Like the SA-DCT, the additional factors that make the SA-IDCT more computationally complex with respect to the  $8 \times 8$  IDCT are vector shape parsing, data alignment and the need for a variable  $N$ -point 1D IDCT transform (according to Equation 2.8b). The SA-IDCT is less regular compared to the  $8 \times 8$  block-based IDCT since its processing decisions are entirely dependent on the shape information associated with each individual block. Also, peculiarities of the SA-IDCT algorithm mean that the shape parsing and data alignment steps are more complicated compared to the SA-DCT. To illustrate the SA-IDCT algorithm, consider Figure 4.1 which illustrates the following steps:

1. Parsing of the  $8 \times 8$  block shape information
2. Horizontal  $N$ -point IDCT on each row of the block of DCT coefficients to produce intermediate reconstructed pixels
3. Horizontal re-alignment of intermediate reconstructed pixels according to parsed shape
4. Vertical  $N$ -point IDCT on each column of the block of intermediate reconstructed pixels to produce the final reconstructed pixels
5. Vertical re-alignment of final reconstructed pixels according to parsed shape

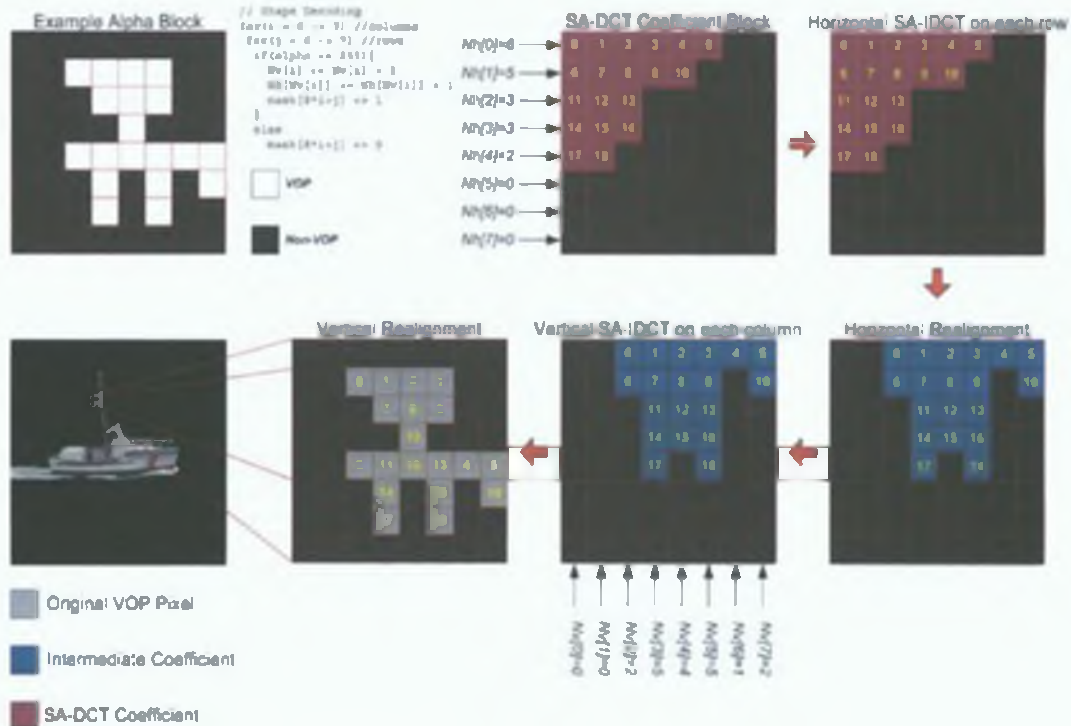


Figure 4.1: The SA-IDCT Algorithm Data Addressing

#### 4.1.1 Additional Complications Compared to SA-DCT

##### 4.1.1.1 Shape Parsing

The first subtle difference to note between the SA-DCT and the SA-IDCT algorithms is the way the shape information is parsed in each. In the case of the SA-IDCT, the entire  $8 \times 8$  alpha block must be parsed before any of the  $N$ -point 1D IDCT computations can be processed. However, in the case of the SA-DCT, an  $N$ -point 1D DCT computation can be processed after each column of the  $8 \times 8$  alpha block is parsed. The consequence of this subtle difference is that it is more difficult to pipeline the SA-IDCT processing steps compared to the SA-DCT from a hardware implementation perspective.

This subtle difference is best illustrated by an example. Consider Figure 4.2, which shows sample input data for an SA-DCT computation. It is clear that after parsing each column of the alpha block, the  $N$  value for that column is available without needing to parse any other alpha column. Hence the vertical 1D DCT  $N$ -point transforms can be pipelined with the column alpha parsing as illustrated by the simple timing diagram in Figure 4.2. Such pipelining is not possible in the case of the SA-IDCT. To illustrate why, consider the sample input data for an SA-IDCT computation in Figure 4.3. Looking at the first row of the coefficient block, it is clear that it requires a 6-point 1D IDCT transform. However, an SA-IDCT architecture has no way of knowing that  $N = 6$  for  $row[0]$  by parsing  $row[0]$  of the original alpha block. As shown in Figure 4.3, the entire  $8 \times 8$  alpha block must be parsed before any  $N$  value can be known for certain. This implies that the particular pipelining discussed in this section is not possible for the SA-IDCT, as illustrated by the simple timing diagram in Figure 4.3.

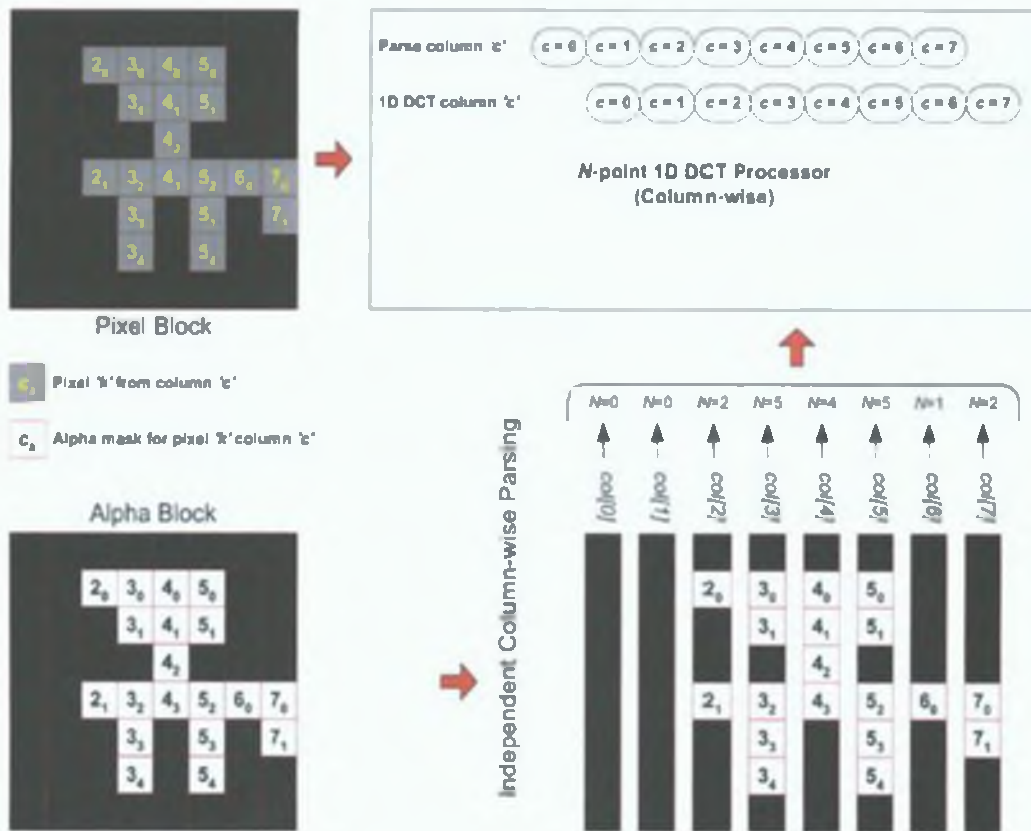


Figure 4.2: Sample SA-DCT Shape Parsing

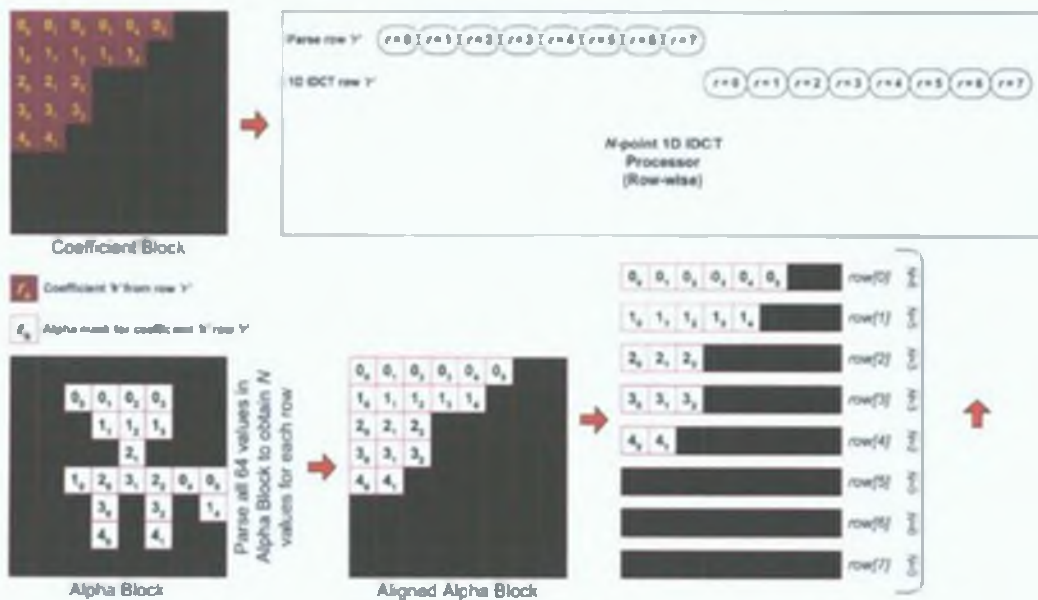


Figure 4.3: Sample SA-IDCT Shape Parsing

#### 4.1.1.2 Data Re-Alignment

The data re-alignment steps for the SA-IDCT are less straightforward compared with the corresponding steps necessary for the SA-DCT. In particular, the vertical re-alignment of reconstructed pixels to their original VO position is not trivial. Essentially, the inverse alignment steps undo the forward SA-DCT packing steps as discussed in Section 3.1 and re-insert any holes (see Figure 3.2). This is more difficult compared to the forward SA-DCT packing of pixel data to the block borders (Figure 3.1) since precise knowledge of the shape pattern is required, whereas only an incremental count of shape pixels in the vector is required for packing. Once the  $16 \times N$  values for the rows and columns have been interpreted during the forward SA-DCT process, the actual shape pattern is no longer needed since the subsequent transform and alignment steps are not contingent on it (as illustrated by Figure 3.1). However, the vertical alignment step during the SA-IDCT process does depend on knowing the shape pattern as is clear from Figure 4.1. For hardware implementation, this means an SA-IDCT accelerator block must somehow retain this information locally to be able to compute for the vertical re-alignment step. Otherwise, it must waste a second access to main memory to re-parse the shape information in order to reconstruct the pixels correctly.

## 4.2 IDCT/SA-IDCT Hardware State of the Art Review

This section outlines the algorithmic and architectural approaches to implementing the  $8 \times 8$  IDCT and in particular the SA-IDCT. Like the DCT, there is a vast amount of literature dedicated to  $8 \times 8$  IDCT implementations due to its computational complexity and importance in signal processing applications. From a video compression point of view, it may be argued that the IDCT is actually more important than the DCT since it is required to encode and decode data whereas the DCT is only needed for encoding. Section 4.2.1 classifies IDCT implementations by general approach. Taking the DCT and IDCT as “face value”, both are constant matrix multiplication computations so the DCT approach classifications discussed in Section 3.2.1 also apply to the IDCT. Special emphasis is given in Section 4.2.1 to IDCT specific approaches that exploit the fact that the nature of the data processed by the DCT and the IDCT is very different. Section 4.2.2 discusses SA-IDCT specific implementations in detail giving context for the proposed SA-IDCT architecture outlined subsequently.

### 4.2.1 Classes of IDCT Implementation Approaches

#### 4.2.1.1 Standard Matrix Multiplication Approaches

As clear from Equations 2.8a and 2.8b, the DCT and IDCT are very similar computations. Both are linear transforms involving a constant matrix multiplication. Hence many implementation proposals in the literature for IDCT have a lot in common with the DCT approaches outlined in Section 3.2. Indeed many architectures are capable of computing both DCT and IDCT and the choice is governed by some mode selection logic.

In Section 3.2.1, DCT approaches are classified under the following groupings: fast-algorithm, systolic, recursive, CORDIC, approximation-based, integer encoding and distributed arithmetic. The technical merits or otherwise of each will not be restated here. Rather, a few examples of each approach that

implement the IDCT are given. Note however, the approximation-based approaches tend to be specifically tailored to the DCT or the IDCT (as discussed in Section 4.2.1.2).

The DCT fast algorithms are directly mappable to IDCT. These include the popular “Chen” algorithm [141], “Lee” [143], “Loeffler” [144], “Feig” [146] and the “AAN” algorithm [147].

The systolic DCT architecture proposed by Aggoun et al. can be used to compute the IDCT but a unified architecture is not outlined [151]. The systolic array architecture based on the “Lee” algorithm proposed by Wu et al. [149] is capable of both DCT and IDCT on a unified structure. For high throughput a SIMD datapath is used. The datapath interconnect is a dynamic switching network controlled by a DCT/IDCT mode select signal. Hsiao et al. propose a scalable  $N \times N$  2D DCT/IDCT systolic array architecture [153]. The architecture has a shared DCT/IDCT kernel processor and a post-processor for DCT mode and pre-processor for IDCT mode.

Chen et al. propose low complexity recursive kernels for computing the DCT and the IDCT [222]. They exploited the symmetry properties of the DCT/IDCT basis matrices to reduce the iteration cycles. Chen et al. also propose a method for designing recursive filter structures that compute an  $M$ -dimensional DCT or IDCT of arbitrary length [159]. This is achieved by re-formulating the equations in a compact form and designing the filters using Chebyshev polynomials.

Hu et al. propose a parallel CORDIC IDCT architecture for high throughput applications [223]. Zhou et al. propose a CORDIC architecture that can compute both the DCT and the IDCT on a unified architecture [224]. The CORDIC iterations are mapped to a pipeline to increase computation speed. Yang et al. compute the IDCT using the CORDIC algorithm but they implement it using a ROM-based distributed arithmetic structure [225]. To speed up the processing, they use a radix-4 redundant signed digit number system to limit the carry propagation delay to a few bit positions in the adders.

The algebraic integer encoding scheme introduced in Chapter 3 in the context of DCT implementations may also be used to implement the IDCT. In the case of the IDCT, algebraic integer encoding is particularly interesting since it eliminates the DCT mismatch problem, and drift between encoders and decoders [173]. To understand this mismatch issue, consider that since the implementation approach to the IDCT is not normative in any of the common video standards, each different implementation could have a different numerical accuracy. Therefore, for a set of given inputs, the outputs of IDCTs from various implementations will likely be slightly different. In the general hybrid video codec presented in Chapter 2, the IDCT results are used in the reconstruction loop in both the encoder and the decoder to reconstruct pictures. When different IDCTs are used in the encoder and the decoder, the difference between the two IDCT outputs, referred to as the IDCT mismatch error, will accumulate. The mismatch error appears as an additional noise in the reconstructed pictures and causes quality degradation. Due to the nature of error accumulation, even a slight IDCT mismatch may cause severe picture quality degradation [173]. An integer encoding implementation of the IDCT is proposed in [168].

Distributed arithmetic, both ROM-based and adder-based, is a very popular architecture for implementing the IDCT as well as the DCT. Uramoto et al. propose a ROM-based DA architecture capable of both DCT and IDCT [226]. They propose a “dual-plane ROM” that allows two bit data to be stored in one transistor memory cell. This means only a single ROM is required as opposed to having discrete ROMs for DCT and IDCT cosines. Guo et al. propose a unified DCT/IDCT ROM-based DA architecture with partitioned ROMs to increase speed [184].

The architecture by Chang et al. [139] is capable of computing the DCT and IDCT using adder-

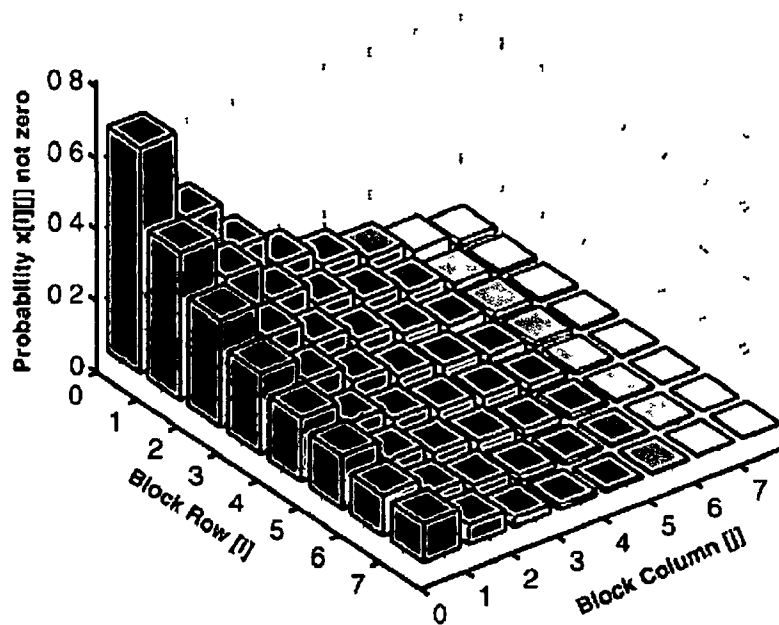


Figure 4.4 Probabilities of Nonzero Occurrence for  $8 \times 8$  DCT Coefficient Blocks [227]

based DA (NEDA) It is designed as a programmable general purpose NEDA processor built around an ALU of an adder/subtractor with programmable shifting of the addends Kim et al also use NEDA in their unified architecture capable of both DCT and IDCT [190] The time-multiplexed NEDA unified DCT/IDCT architecture by Guo et al reconstructs pixels serially in IDCT mode and only requires 10 adders in the distributed weight generation logic [194]

#### 4.2.1.2 IDCT Specific Approaches

In a video codec, the nature of the input data processed by the IDCT compared to the DCT is very different The DCT is fed pixel blocks in intra mode and motion compensated residual blocks in inter mode The statistical nature of such 2D pixel data from natural scenes is commonly modelled as a first order Markov process [53, 50] As discussed in Chapter 2, the DCT decorrelates pixel data and packs most of the pixel block energy for natural scenes into a few low frequency coefficients The AC DCT coefficients exhibit sharp zero-centred dual-sided Laplacian probability distributions [227], and this implies a large number of zero-valued coefficients per  $8 \times 8$  block for image and video data Subsequent quantisation increases the chances of high frequency components being reduced to zero This phenomenon is illustrated clearly in Figure 4.4 [227], and is exploited by some hardware implementations of the IDCT proposed in the literature

McMillan et al express the IDCT as a forward-mapping formulation, i.e. each input element's contribution to the entire set of output elements is expressed independently [228] This can be exploited since the IDCT input coefficient matrix is sparse The authors claim that for a set of sample images there are only between four and twelve non-zero values in the input matrix Since each input is treated



independently, this allows iterative computation of a reconstructed block according to a quality-speed trade-off (e.g. by parsing the input matrix in a zig-zag scan). The hardware architecture proposed is a systolic array of 64 multiply-accumulate (MAC) units. A similar approach is used in [229], but the architecture is serial. The approach by Lee et al. is also similar to McMillan's approach since it uses the forward-mapping formulation and has a systolic array structure [230, 231]. However, Lee et al. exploit the symmetry of the IDCT basis functions when deriving their formulation. This reduces the multiplicative complexity and processing latency of their architecture.

A data-driven IDCT architecture is presented by Xanthopoulos et al. in [232, 233], which also exploits the fact that a large percentage of DCT coefficients are zero valued. As well as avoiding zero argument computations, the architecture also has dynamic voltage and frequency scaling for greater power savings. The variability of the workload is exploited by adaptively changing the power supply and clock frequency of the main computation units. The architecture is a ROM-based DA implementation of the "Chen" IDCT algorithm. It claims to be more energy efficient than a conventional row-column DA IDCT by more than an order of magnitude for the same sample rate. Xanthopoulos et al. have also published an architecture based on similar principals except that the "Chen" IDCT algorithm is implemented using a MAC structure as opposed to DA [227].

Kim et al. propose MAC structure [234] similar to that by Xanthopoulos [227]. Based on the number of non-zero DCT coefficients at the input, it is possible to clock gate some of the registers/multipliers in the pipeline. The register pipeline is reconfigurable to allow stages to be skipped using bypass multiplexers if the data rate is low. Their experimental results show that most of the MPEG video sequences have DCT blocks with typically five or six non-zero coefficients, mainly located in the low spatial frequency positions. The architecture assumes that the number of non-zero coefficients has been established a priori by simply counting the outputs of the variable length decoder or inverse quantiser.

Fanucci et al. propose a data driven unified architecture capable of both DCT and IDCT [183]. Previous approaches that exploit IDCT data statistics used two separate architectures for DCT and IDCT. The architecture in [183] is based on ROM-based DA. Since such an architecture is a serial computation scheme, sign extension of input data does not contribute to overall result. Such bits are discarded when detected by the architecture to eliminate redundant computation. The control logic also detects the presence of so-called "null rows" (a row in the input DCT coefficient matrix with all-zero values). If a null row is detected, the datapath is clock gated avoiding unnecessary computation and the corresponding row in the transpose memory is reset to all-zeros.

#### 4.2.2 SA-IDCT Specific Approaches

Similar to the SA-DCT, the SA-IDCT is a column-row process by definition and cannot be easily implemented directly as a 2D equation given the large number of possible  $8 \times 8$  shape configurations. This precludes the possibility of expressing the SA-IDCT in a single forward-mapping formulation. The primary computation engines required to implement the SA-IDCT are a variable  $N$ -point 1D IDCT processor, an alpha shape parser (for data re-alignment) and some form of memory scheme to store the intermediate reconstructed pixels after the horizontal inverse transforms. The dynamic nature of the SA-IDCT processing coupled with the additional complexities compared to SA-DCT (see Section 4.1.1) make efficient hardware implementation difficult. Previously proposed implementations of the SA-IDCT do not explicitly explain how they deal with the issues described in Section 4.1.1, and indeed some assume a

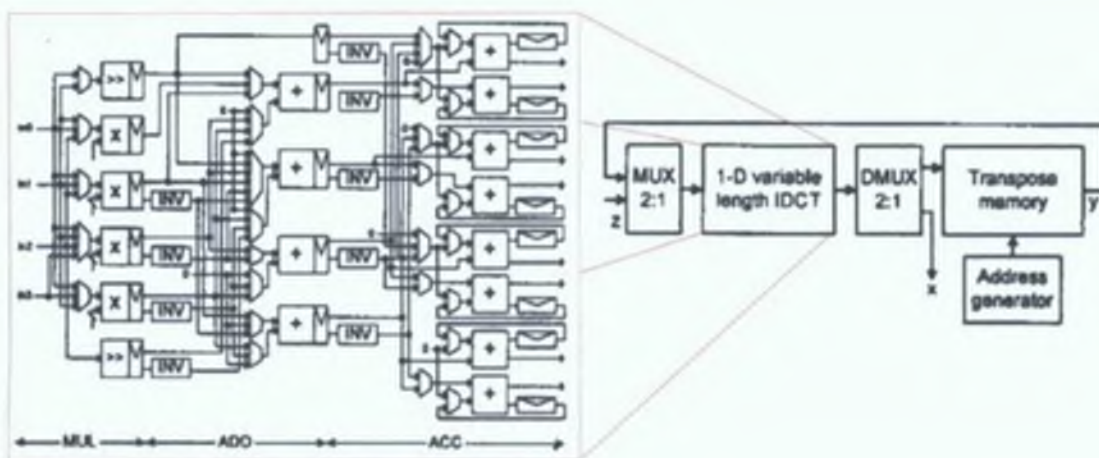


Figure 4.5: Auto-Aligning Architecture (Hsu et. al. [235])

priori information about the shape information.

A unified architecture capable of computing both SA-DCT and SA-IDCT is proposed by Tseng et. al. [201] (discussed in Section 3.2.2 in the context of SA-DCT). They propose a reconfigurable variable  $N$ -point 1D DCT/IDCT processor, although only a block diagram is provided without a discussion on architectural features (see Figure 3.9). The pipeline is dynamically configured according to the shape information. Details on the shape parsing and alignment are also unclear but it seems that the shape is parsed serially in a vertical raster and fed to a look-up table that provides control configuration for the datapath. The re-alignment scheme is mentioned but not described apart from stating that the look-up table outputs re-shift the data. Explicit shifting of data is inefficient in terms of power consumption and also costs extra computation cycles.

Chen et. al. have also developed a unified architecture for SA-DCT and SA-IDCT [202, 109]. Its architectural features are discussed in Section 3.2.2 in the context of SA-DCT. The paper proposes a variable  $N$ -point 1D DCT/IDCT datapath and does not include shape parsing or data re-alignment logic. As outlined in Section 3.2.2, the datapath requires approximately 3100 gates (including a single adder and no multipliers) and has a worst case clock cycle latency of 124.

Hsu et. al. have proposed an architecture specifically for the SA-IDCT [235]. A block diagram is shown in Figure 4.5, and it has a similar structure to the SA-DCT architecture proposed by the same authors [203] (as discussed in Section 3.2.2). The datapath uses time-multiplexed adders and multipliers coupled with an auto-aligning transpose memory. As is clear from Figure 4.5, the datapath is implemented using 4 multipliers and 12 adders. The worst case computation cycle latency is 8 clock cycles. The transpose memory in Figure 4.5 is used to store the intermediate reconstructed pixels computed by the horizontal inverse transformations prior to vertical inverse transformation. In contrast to their SA-DCT architecture described in [203], it is not clear how their SA-IDCT auto-alignment address generation logic operates. The architecture also employs skipping of all-zero input data to save unnecessary computation, although again specific details discussing how this is achieved are omitted. As is the case for their SA-DCT architecture, the critical path caused by the multipliers in this SA-IDCT architecture limits the maximum operating frequency and has negative power consumption consequences.

The SA-IDCT architecture proposed in this thesis addresses the issues outlined by employing a

reconfiguring adder-only based distributed arithmetic structure. As discussed in Chapter 3, multipliers can be prohibitive compared to adders in terms of area and power consumption. Hence by avoiding multipliers, the author believes that the proposed architecture offers a better trade-off between speed, area and power. The datapath computes serially each reconstructed pixel  $k$  ( $k = 0, \dots, N - 1$ ) of an  $N$ -point 1D IDCT by reconfiguring the datapath based on the value of  $k$  and  $N$ . Guarded evaluation and local clock gating are employed using  $k$  and  $N$  to ensure that redundant switching is avoided for power efficiency. A transpose memory (TRAM) has been designed whose surrounding control logic ensures that the SA-IDCT data re-alignment is computed efficiently without needless switching or shifting. A pipelined approach alleviates the computational burden of needing to parse the entire shape  $8 \times 8$  block before the SA-IDCT can commence.

Due to the additional algorithmic complexity, it is more difficult to design a unified SA-DCT/SA-IDCT module compared to a unified  $8 \times 8$  DCT/IDCT module. The reasons for not attempting to do so in the proposed work may be summarised as follows:

- A video decoder only requires the SA-IDCT. Since SA-DCT and SA-IDCT require different addressing logic, embedding both in the same core will waste area if the final product is a video decoder application only.
- A video encoder needs both SA-DCT and SA-IDCT, but if real-time constraints are tight it may be required to have the SA-DCT and SA-IDCT cores executing in parallel. If this is the case, it makes sense to have a dedicated task-optimised core. Admittedly, this has negative silicon area implications.
- Even though the addressing logic for SA-DCT and SA-IDCT are quite different, the core datapaths that compute the transforms are very similar. Therefore it may be viable to design a unified variable  $N$ -point 1D DCT/IDCT datapath and have separate dedicated addressing logic for each. Future work could involve designing such an architecture and comparing its attributes against the two distinct dedicated cores presented in this thesis.

### 4.3 SA-IDCT Datapath Architecture

The top-level SA-IDCT architecture is shown in Figure 4.6, comprising of the TRAM and datapath with their associated control logic. It has a similar architecture to the SA-DCT proposed in Chapter 3 (see Figure 3.12). For all modules, local clock gating is employed based on the computation being carried out to avoid wasted power. The addressing control logic (ACL) firstly parses the  $8 \times 8$  shape information in a vertical raster fashion (as in Figure 4.7) to interpret the horizontal and vertical  $N$  values as well as saving the shape pattern into interleaved buffer register files. Each register file is 128 bits wide ( $16 \times 4$ -bits for  $N$  values plus 64 bits for shape pattern). Interleaving the data increases throughput since when one  $8 \times 8$  block is being processed by the SA-IDCT logic, the ACL can begin to parse the next block into the alternate interleaved buffer. The ACL then uses the parsed shape information to read the input SA-DCT coefficient data from memory – this requires  $num\_vo\_pels$  memory accesses where  $num\_vo\_pels$  is the subset of pels in the  $8 \times 8$  block that form part of the VO. The SA-DCT coefficients are parsed in a horizontal raster fashion (e.g. Figure 4.7). Each successive row vector is stored in an alternate interleaved buffer to improve latency.

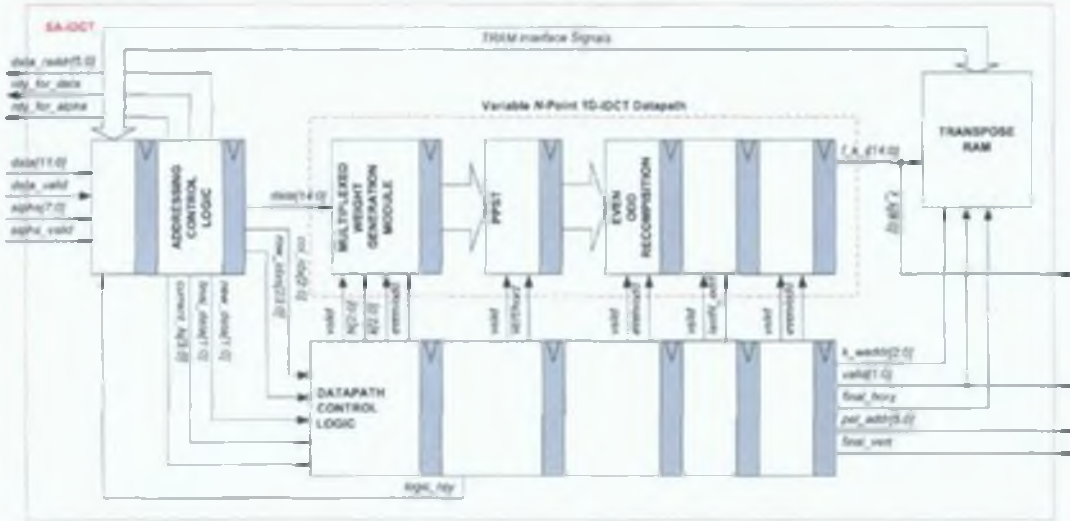


Figure 4.6: Top-Level SA-IDCT Architecture

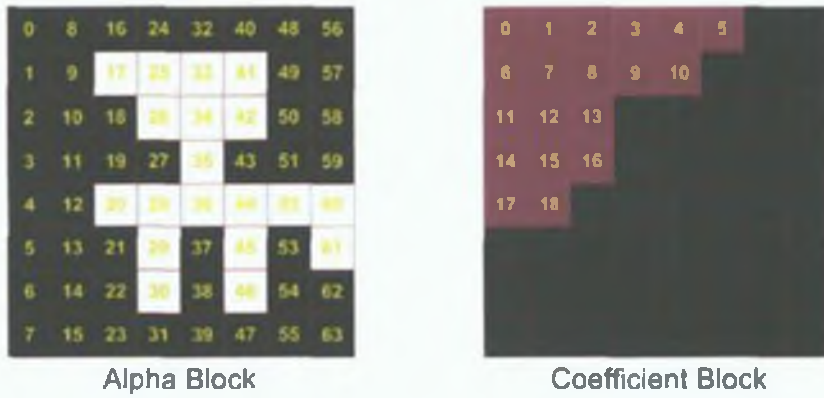


Figure 4.7: SA-IDCT Vertical Alpha Scanning and Horizontal Coefficient Scanning

When loaded, a vector is then passed to the variable  $N$ -point 1D IDCT module, which computes all  $N$  reconstructed pixels serially in a ping-pong fashion (i.e.  $f[N-1], f[0], f[N-2], f[1], \dots$ ). The reason for this order is discussed in Section 4.3.3. The variable  $N$ -point 1D IDCT module is a five stage pipeline and employs adder-based distributed arithmetic using a multiplexed weight generation module (MWGM) and a partial product summation tree (PPST), followed by even-odd recombination. The MWGM and the PPST are very similar in architecture to the corresponding modules used for the SA-DCT as discussed in Chapter 3. The even-odd recombination module has a three clock cycle latency whereas the corresponding decomposition module used for the SA-DCT in Chapter 3 has only a single cycle latency. This accounts for the additional two cycles over the variable  $N$ -point 1D DCT module.

The TRAM has a 64 word capacity, and when storing data the index  $k$  is manipulated to store the value at address  $8 \times N\_vert[k] + k$  after which  $N\_vert[k]$  is incremented by 1. The TRAM addressing scheme used by the ACL looks after the horizontal and vertical re-alignment steps by using the parsed shape information. The data read from the TRAM is routed to the datapath which is re-used to compute the final reconstructed pixels that are driven to the module output along with its address in the  $8 \times 8$  block according to the shape information.

#### 4.3.1 Reconfiguring Adder-Based Distributed Arithmetic Dot Product

The dot product of a 1D coefficient data vector with the appropriate  $N$ -point IDCT basis vector yielding reconstructed pixel  $k$  is computed using a reconfiguring adder-based distributed arithmetic structure (the MWGM) followed by a PPST. The architecture is almost identical to the one used for the forward SA-DCT (as shown in Figure 3.15 and Figure 3.19) and has been designed in the same manner using the recursive iterative matching algorithm [40]. The advantages of such a structure may be summarised as follows:

- Power consumptive multipliers are avoided in favour of an adder-only structure
- The re-configuring structure promotes hardware re-use
- Balanced adder network for regularity and to reduce the probability of glitching
- Local clock gating of distributed weight registers

The primary difference for the SA-IDCT MWGM is that the multiplexer configurations controlled by  $\{k, N\}$  are different since the IDCT basis vectors are different to those of the forward DCT. In fact the MUX configurations for odd  $k$  for all  $N$  are actually identical to the forward transform but not so for even  $k$ . This is due to anti-symmetry of the odd transform basis vectors.

Experimental results have shown that for a range of video test sequences, 13 distributed binary weights are needed to adequately satisfy reconstructed image quality requirements (see Section 4.5.1). As expected, this is identical to the precision required for the forward SA-DCT as discussed in Chapter 3. The 11 adders needed for the SA-IDCT MWGM are identical to those used for the forward SA-DCT. Hence it is possible to extend the capability of a single MWGM module for both forward and inverse transform processing by extending the multiplexer configuration signal from the current 6-bit  $\{k, N\}$  to a 7-bit signal  $\{k, N, mode\}$  ( $mode = 0 \Rightarrow$  SA-DCT  $mode = 1 \Rightarrow$  SA-IDCT).

Table 4 1 Total Number of Unique Possibilities for Each Distributed SA-IDCT Weight

	$W_0$	$W_{-1}$	$W_{-2}$	$W_{-3}$	$W_{-4}$	$W_{-5}$	$W_{-6}$	$W_{-7}$	$W_{-8}$	$W_{-9}$	$W_{-10}$	$W_{-11}$	$W_{-12}$
# Unique Selections	8	9	11	13	12	13	12	11	9	12	15	12	12

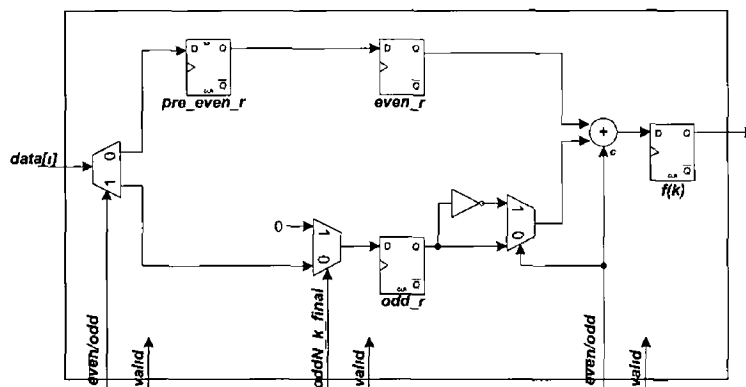


Figure 4 8 Even-Odd Recomposition (EOR) Architecture

Like the SA-DCT MWGM, a certain degree of overlap exists for the distributed weight values over the 36 valid cases possible using configuration signal  $\{k, N\}$ . This case overlap decreases the synthesised MUX complexity. For each of the 36 cases there are 16 possible values for a weight (zero and signals  $x_0, x_1, x_2, x_3, x_{01}, x_{23}, x_{02}, x_{03}, x_{12}, x_{13}, x_{01x23}, x_{01x2}, x_{01x3}, x_{02x3}, x_{12x3}$  in Figure 3 15). However, of the 36 valid configurations for each weight, only a subset of these 16 possibilities are used as illustrated in Table 4 1. The weights are then combined by the PPST to produce pixel  $f(k)$  as described in Section 4 3 2.

#### 4 3 2 Partial Product Summation Tree

The use of adder-based distributed arithmetic necessitates a PPST to combine the weights together to form the reconstructed pixel. The architecture used for the PPST in Figure 4 6 is identical to that shown in Figure 3 19 (apart from the rounding ranges used). The reconstructed pixels after the horizontal inverse transforms are rounded to 11  $f$  fixed-point bits (11 bits for integer and  $f$  bits for fractional) and the experiments show that  $f = 4$  represents a good trade-off between area and performance. This implies that each word in the TRAM is 15 bits wide. The final reconstructed pixels produced after the vertical inverse transforms are rounded to 9 0 bits and are routed directly to the module outputs.

#### 4 3 3 Even-Odd Recomposition

The purpose of the even-odd recombination (EOR) circuit is to combine successive values produced by the PPST and reform the reconstructed pixel values. The reason for this is explained by the mathematics of EOD/EOR. Consider Equations 4 1a and 4 1b which are the general  $N$ -point even-odd decomposition equations (Equations 3 11a and 3 11b take the value  $N = 8$ ). Substituting into Equations 4 1c and 4 1d yield the corresponding inverse transform equations by even-odd recombination (refer to Section 2 2 2 for a discussion on the mathematics of orthogonal forward and inverse transform pairs).

$$F_{\text{even}} = \begin{cases} \begin{bmatrix} F(0) \\ F(2) \\ \\ F(N-4) \\ F(N-2) \end{bmatrix} \\ \begin{bmatrix} F(0) \\ F(2) \\ \\ F(N-3) \\ F(N-1) \end{bmatrix} \end{cases} = \mathbf{A}_{\text{even}} \times \mathbf{s} = \mathbf{A}_{\text{even}} \times \begin{cases} \begin{bmatrix} f(0) + f(N-1) \\ f(1) + f(N-2) \\ \\ f(\frac{N}{2} - 2) + f(\frac{N}{2} + 1) \\ f(\frac{N}{2} - 1) + f(\frac{N}{2}) \end{bmatrix} \\ \begin{bmatrix} f(0) + f(N-1) \\ f(1) + f(N-2) \\ \\ f(\frac{N-1}{2} - 1) + f(\frac{N-1}{2} + 1) \\ f(\frac{N-1}{2}) \end{bmatrix} \end{cases} \begin{cases} \text{if } N \text{ even,} \\ \text{if } N \text{ odd} \end{cases} \quad (4 \text{ la})$$

$$F_{\text{odd}} = \begin{cases} \begin{bmatrix} F(1) \\ F(3) \\ \\ F(N-3) \\ F(N-1) \end{bmatrix} \\ \begin{bmatrix} F(1) \\ F(3) \\ \\ F(N-4) \\ F(N-2) \end{bmatrix} \end{cases} = \mathbf{A}_{\text{odd}} \times \mathbf{d} = \mathbf{A}_{\text{odd}} \times \begin{cases} \begin{bmatrix} f(0) - f(N-1) \\ f(1) - f(N-2) \\ \\ f(\frac{N}{2} - 2) - f(\frac{N}{2} + 1) \\ f(\frac{N}{2} - 1) - f(\frac{N}{2}) \end{bmatrix} \\ \begin{bmatrix} f(0) - f(N-1) \\ f(1) - f(N-2) \\ \\ f(\frac{N-1}{2} - 2) - f(\frac{N-1}{2} + 2) \\ f(\frac{N-1}{2} - 1) - f(\frac{N-1}{2} + 1) \end{bmatrix} \end{cases} \begin{cases} \text{if } N \text{ even,} \\ \text{if } N \text{ odd} \end{cases} \quad (4 \text{ lb})$$

$$\begin{aligned}
\mathbf{A}_{even}^T \times F_{even} &= \mathbf{A}_{even}^T \times \mathbf{A}_{even} \times \mathbf{s} \\
&= \begin{cases} \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0.5 \end{bmatrix} \times \mathbf{s} & \text{if } N \text{ even,} \\ \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \mathbf{s} & \text{if } N \text{ odd} \end{cases} \quad (4 \text{ lc})
\end{aligned}$$

$$\begin{aligned}
\mathbf{A}_{odd}^T \times F_{odd} &= \mathbf{A}_{odd}^T \times \mathbf{A}_{odd} \times \mathbf{d} \\
&= \begin{cases} \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0.5 \end{bmatrix} \times \mathbf{d} & \text{if } N \text{ even,} \\ \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0.5 \end{bmatrix} \times \mathbf{d} & \text{if } N \text{ odd} \end{cases} \quad (4 \text{ ld})
\end{aligned}$$

The crucial point to note from Equations 4 1c and 4 1d is that recomposition produces added/subtracted pairs of the original pixel values scaled by 0.5. This is except for the one particular case of the middle odd index pixel when  $N$  is odd (e.g. pixel  $k = 3$  when  $N = 7$ ). This is because this middle odd pixel does not contribute to any of the odd coefficients when  $N$  is odd (as is clear from Table 3 1). Mathematically speaking the data is scaled since the EOD basis matrices are not orthogonal so  $\mathbf{A}_{odd}^T \times \mathbf{A}_{odd}$  or  $\mathbf{A}_{even}^T \times \mathbf{A}_{even}$  will not result in the identity matrix.



Table 4 2 SA-IDCT EOR Ping Pong Data Ordering

N	data[0]	data[1]	data[2]	data[3]	data[4]	data[5]	data[6]	data[7]
8	$\frac{f(0)+f(7)}{2}$	$\frac{f(0)-f(7)}{2}$	$\frac{f(1)+f(6)}{2}$	$\frac{f(1)-f(6)}{2}$	$\frac{f(2)+f(5)}{2}$	$\frac{f(2)-f(5)}{2}$	$\frac{f(3)+f(4)}{2}$	$\frac{f(3)-f(4)}{2}$
7	$\frac{f(0)+f(6)}{2}$	$\frac{f(0)-f(6)}{2}$	$\frac{f(1)+f(5)}{2}$	$\frac{f(1)-f(5)}{2}$	$\frac{f(2)+f(4)}{2}$	$\frac{f(2)-f(4)}{2}$	$f(3)$	
6	$\frac{f(0)+f(5)}{2}$	$\frac{f(0)-f(5)}{2}$	$\frac{f(1)+f(4)}{2}$	$\frac{f(1)-f(4)}{2}$	$\frac{f(2)+f(3)}{2}$	$\frac{f(2)-f(3)}{2}$		
5	$\frac{f(0)+f(4)}{2}$	$\frac{f(0)-f(4)}{2}$	$\frac{f(1)+f(3)}{2}$	$\frac{f(1)-f(3)}{2}$	$f(2)$			
4	$\frac{f(0)+f(3)}{2}$	$\frac{f(0)-f(3)}{2}$	$\frac{f(1)+f(2)}{2}$	$\frac{f(1)-f(2)}{2}$				
3	$\frac{f(0)+f(2)}{2}$	$\frac{f(0)-f(2)}{2}$	$f(1)$					
2	$\frac{f(0)+f(1)}{2}$	$\frac{f(0)-f(1)}{2}$						
1	$f(0)$							

Using a serial computation scheme the order of data entering the EOR module for different  $N$  is summarised in Table 4 2 The EOR module converts these sequences into the original pixel values Taking  $N = 8$  as an example,  $data[0] + data[1] = f(0)$  and  $data[0] - data[1] = f(7)$  The EOR module takes successive pairs of samples and recomposes the original pixel values but in a ping-pong order ( $f(0), f(7), f(1), f(6), \dots$ ) This data ordering eliminates the need for data buffering that is required if the sequence generated is ( $f(0), f(1), f(2), f(3), \dots$ ) The ping-pong ordering must also be taken into account by the addressing and control logic responsible for intermediate data storage in the TRAM and vertical re-alignment of the final coefficients (see Section 4 4)

A schematic of the EOR module is shown in Figure 4 8 It has a three stage pipeline and the data flow is controlled by three multiplexers The input signal  $data[i]$  is updated on every cycle for  $N$  cycles from  $i = 0, 1, \dots, N - 1$  The pipelined *even/odd* control signal is set or cleared depending on whether the corresponding pipelined value of  $i$  is even or odd The signal *oddN\_k\_final* is asserted whenever the condition  $i = N - 1$  and  $N$  is odd is detected at the second pipeline stage This deals with the special case of the middle odd index pixel when  $N$  is odd as outlined earlier The advantageous features of the EOR module are its simplicity and efficient resource usage The interleaved pipeline means that the module can sample new data every clock cycle

#### 4 4 SA-IDCT Memory and Control Architecture

The primary feature of the memory and addressing modules in Figure 4 6 is that they avoid redundant register switching and latency when addressing data and storing intermediate values This is achieved by manipulating the shape information The architectures are similar to the corresponding forward SA-DCT counterparts outlined in Section 3 5 However, the distinctions between the SA-DCT and SA-IDCT algorithms necessitate different addressing strategies

The ACL (Figure 4 9) parses shape and SA-DCT coefficient data from an external memory and routes the data to the variable  $N$ -point 1D IDCT datapath for processing in a row-wise fashion The intermediate coefficients after the horizontal processing are stored in a transpose memory (Figure 4 11) The ACL then reads each vertical data vector from this transpose memory for vertical inverse transformation by the datapath

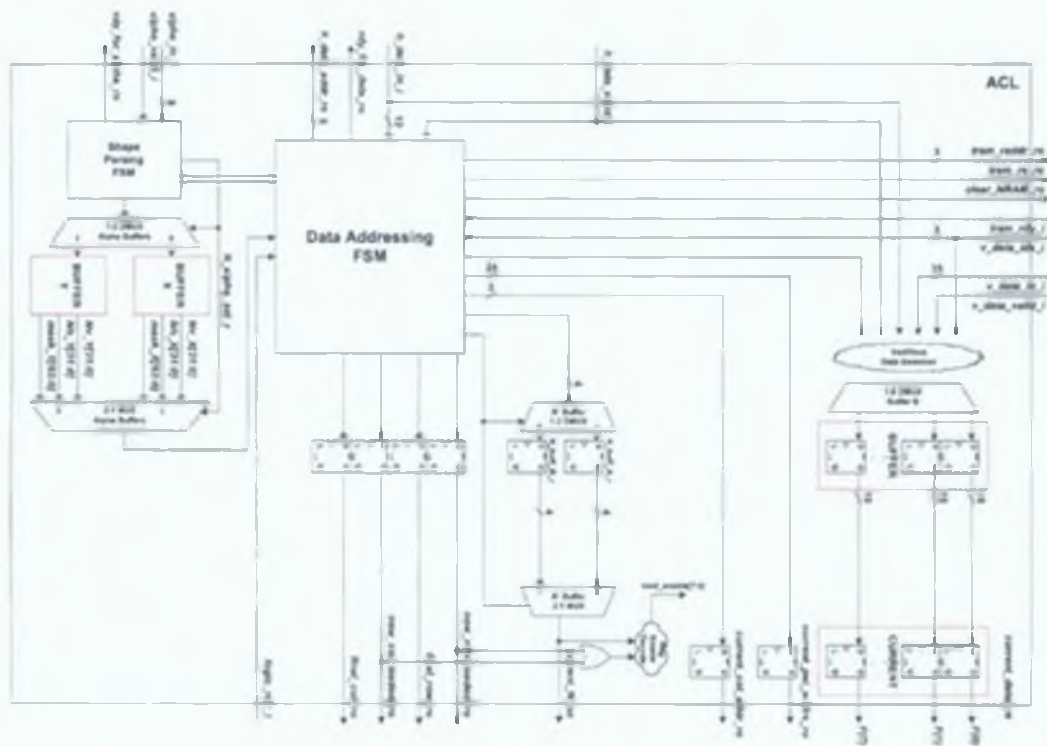


Figure 4.9: SA-IDCT ACL Architecture



Figure 4.10: SA-IDCT ACL Data Processing Pipeline

#### 4.4.1 Addressing/Routing Control Logic

Since the alpha information must be fully parsed before any horizontal IDCTs can be computed, the SA-IDCT algorithm requires more computation steps compared to the forward SA-DCT. The proposed ACL tackles this by employing two parallel finite state machines – one for alpha parsing and the other for data addressing. This reduces the algorithm processing latency. As clear from Figure 4.9, the ACL has two interleaved buffers for storing parsed shape information (*bufferX* and *bufferY*). The parallel processing is described by the following sequential steps (also illustrated by Figure 4.10):

1. Parse alpha block  $b$  into shape *bufferX* if  $b$  even or *bufferY* if  $b$  odd while in parallel the datapath computes vertical IDCTs on block  $b - 1$ .
2. The datapath computes horizontal IDCTs on block  $b$ . Then  $b$  is incremented by one and the processing returns to step 1.

As is clear from Figure 4 10, the parallel state machines mean that the variable  $N$ -point IDCT datapath is continuously fed with data after the first pipeline stage. Apart from the parsing of the very first alpha block, no additional computation cycles are caused by the requirement to parse the shape information prior to IDCT processing. The shape information is parsed according to the pseudo code in Figure 4 1. Each of the 16  $N$  values are stored in a 4-bit register and the shape pattern is stored in a 64-bit register. The shape pattern requires storage for the vertical re-alignment step, since this re-alignment cannot be computed from the  $N$  values alone. A run-length encoding scheme was considered for storing the shape pattern but was dismissed as too complicated given that the datapath produces reconstructed data in a ping-pong order. This would either require a complex run length decoder or a sophisticated ping-pong run length encoding scheme. Hence, a 64-bit register with very simple control logic was chosen.

Once an alpha block has been parsed, the data addressing finite state machine uses the horizontal  $N$  values to read SA-DCT coefficient data from an external memory row by row. Since the shape information is now known, the state machine only reads from memory locations relevant to the VO (see Figure 4 7). Like the SA-DCT ACL discussed in Section 3 5 1, the SA-IDCT ACL uses a parallel/interleaved data buffering scheme to maximise throughput. By virtue of the fact that the SA-DCT coefficients are packed into the top left hand corner of the  $8 \times 8$  block, early termination is possible for the horizontal IDCT processing steps. If the horizontal  $N$  value for row index  $j$  has been parsed as 0 it is guaranteed that all subsequent rows with index  $> j$  will also be 0 since the data is packed. Hence the vertical IDCT processing can begin immediately if this condition is detected.

The data addressing finite state machine reads intermediate coefficients column-wise from the TRAM for the vertical IDCT processing. Early termination based on  $N = 0$  detection is not possible in this case since the data is no longer packed (e.g. Figure 4 1). When a column is being read from the TRAM, the data addressing FSM also re-generates the original pixel address from the 64-bit shape pattern. This 64-bit register is divided into an 8-bit register for each column. Using a 3-bit counter, the state machine parses the 8-bit register for the current column until all  $N$  addresses have been found. These addresses are read serially by the  $N$ -point IDCT datapath as the corresponding pixel is reconstructed.

#### 4 4 2 Transpose Memory

The TRAM in Figure 4 11 is a 64-word  $\times$  15-bit dual port RAM that stores the reconstructed data produced by the horizontal inverse transform process. This data is then read by the ACL in a transposed fashion and vertically inverse transformed by the datapath yielding the final reconstructed pixels. The TRAM itself is identical to the structure proposed in Chapter 3 for the SA-DCT, but the addressing logic is different. As is clear from Figure 4 11, when storing data here the index  $k$  is manipulated to store the value at address  $8 \times N\_curr[k] + k$ . Then  $N\_curr[k]$  is incremented by 1. After the entire block has been horizontally inverse transformed, the TRAM has the resultant data packed to the top left corner of the block (e.g. top right block in Figure 4 1). For the subsequent vertical inverse transformations, the ACL data addressing state machine combined with the  $N$  value registers beside the TRAM read the appropriate data from the TRAM. Horizontal re-alignment is intrinsic in the addressing scheme meaning explicit data shifting is not required. Instead, manipulating the shape information combined with some counters control the re-alignment.



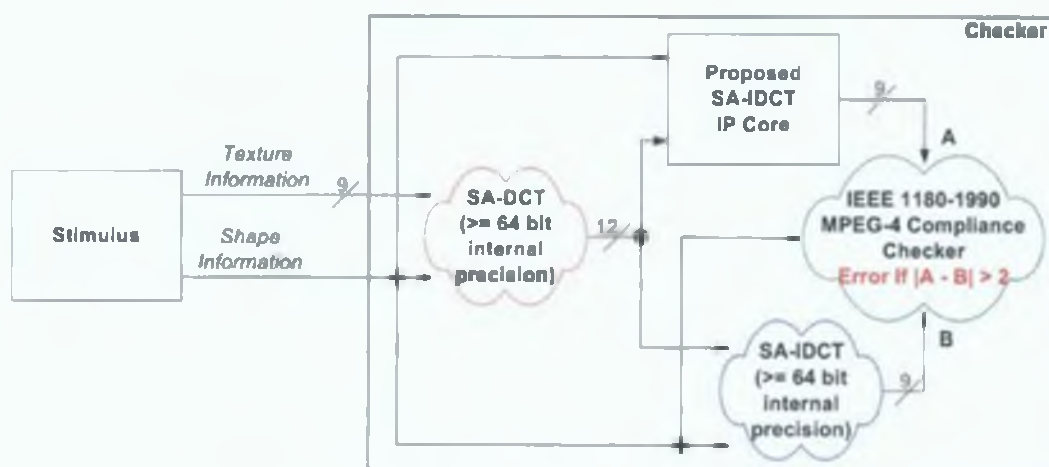


Figure 4.12: SA-IDCT IP Core Verification Platform

Table 4.3: SA-IDCT Core IEEE 1180-1990 Results for  $Q = -12$ ,  $f = 4$

$[-L, H]$	Sign	OMSE	PSNR [dB]	Sign	OMSE	PSNR [dB]
[-256,255]	+1	0.545	50.766	-1	0.544	50.774
[-5,5]	+1	0.494	51.196	-1	0.494	51.196
[-384,383]	+1	0.357	52.602	-1	0.357	52.602

reconstructed pixels are said to be compliant if they are within two grey levels of the same data that has undergone a double precision SA-DCT/SA-IDCT process. The test vectors used were generated by the random number generator specified by the IEEE 1180-1990 standard [173], as described in Chapter 3. Experiments have shown that indeed  $Q = -12$  and  $f = 4$  are required to meet the standard requirements. The OMSE and PSNR results for each standard experiment are summarised in Table 4.3.

## 4.5.2 Evaluation Against Prior Art

The SA-IDCT architecture has been implemented in Verilog and synthesised targeting the same three technologies introduced in Chapter 3: a  $0.09\mu\text{m}$  low power standard cell ASIC library by Taiwan Semiconductor Manufacturing Company (TSMC), a Xilinx Virtex-II FPGA and a Xilinx Virtex-E FPGA. The ASIC flow is necessary for benchmarking against prior art architectures, and the FPGA flows are used for conformance testing with the MPEG-4 standard requirements and embedded system prototyping. The ASIC synthesis results are summarised in Table 4.4. The SA-IDCT architecture has been physically implemented on the two FPGAs listed as part of the two integration frameworks introduced in Chapter 3. The SA-IDCT FPGA synthesis results are discussed in Section 4.5.3 and Section 4.5.4.

### 4.5.2.1 Area and Speed

This section attempts to compare the SA-IDCT architecture proposed in this thesis against prior art in terms of speed and area using the normalisations introduced in Chapter 2 for meaningful analysis. The corresponding values for the prior art and the proposed architecture are summarised in Table 4.4.

Table 4.4: SA-IDCT 90nm TSMC Synthesis Results and Benchmarking

Architecture	Process [ $\mu\text{m}$ ]	Cycle Count			+	*	Gates	PGCC	Speed [MHz]	
		General	Max	Min						
Chen[109]	◇	0.35	n/a	124□	14	1	0	3157	$3.9 \times 10^5$	83
	★		n/a	1984□	14	n/a	n/a	5775	$3.9 \times 10^7$	
Hsu[235]	◇	0.18	n/a	8□	3	12	4	n/a	n/a	62.5
	★		n/a	84□	n/a	n/a	n/a	377685	$3.2 \times 10^7$	
			n/a	10■	n/a				$3.8 \times 10^5$	
Proposed Approach	◇	0.09	$N + 4$	12□	5	24	0	9780	$1.2 \times 10^5$	588
	★		n/a	188□	125	32	0	27518	$5.2 \times 10^5$	

Key: ◇ ⇒ Datapath Only, ★ ⇒ Datapath & Memory

Operating Modes: □ ⇒ No Zero Skipping, ■ ⇒ Zero Skipping Implemented

Synthesising the proposed design with Synopsys Design Compiler targeting TSMC 0.09 $\mu\text{m}$  TCBN90LP technology yields a gate count of 27518. The area of the variable  $N$ -point 1D IDCT datapath is 9780 (excluding TRAM memory and ACL). Both gate counts are used to benchmark against other proposals according to the results available in the literature. This is an improvement over Hsu [235] and offers a better trade-off in terms of cycle count versus area compared with Chen [109], as discussed subsequently. Similar to the SA-DCT architecture proposed in Chapter 3, no multipliers have been used. The datapath uses adders only – 24 in total – divided between the EOR module (1), the MWGM (11) and the PPST (12). Using the estimation that a multiplier is equivalent to about 20 adders in terms of area, the adder count of the proposed architecture (24) compares favourably with Hsu[235] ( $4 \times 20 + 12 = 92$ ). This is offset by the additional MUX overhead, but as evidenced by the overall gate count figure of the proposed architecture, it still yields an improved circuit area. By including the TRAM (1) and the ACL controller (7), an additional 8 adders are required by the entire proposed design. In total, the entire design therefore uses 32 adders and no multipliers.

The worst case clock cycle latency of the SA-DCT core is 188 cycles, and this latency is incurred when processing an  $8 \times 8$  opaque block (an internal block in the VO). For boundary blocks the latency is  $188 - (64 - \text{opaque\_pels})$ , where *opaque\_pels* is the number of VO pels. Hence, when working out the real-time processing constraints for the proposed SA-IDCT module, the worst case latency of 188 cycles must be used. CIF resolution with a frame rate of 30 fps requires  $71.28 \times 10^3$  blocks to be processed per second. This translates to a maximum block processing latency of approximately 14 $\mu\text{s}$ . Given that the proposed SA-IDCT module requires at worst 188 cycles to process a block, this means that the longest allowable clock period for the proposed SA-DCT module is approximately 74.6ns. This translates to a minimum operating frequency of about 14MHz. Table 4.4 shows that the maximum achievable frequency with TSMC 0.09 $\mu\text{m}$  TCBN90LP technology is 588MHz, so the proposed SA-IDCT core can comfortably meet real-time constraints. Running at 588MHz, the proposed core can process a single  $8 \times 8$  block in 338ns. At this frequency, the design is capable of a throughput of 200 Mpixel/s. As discussed in Chapter 2, the design should run as slow as possible (14MHz) for low power consumption, and the maximum throughput in this case is 4.8Mpixel/s.

In Chapter 2, the *product of gate count and computation cycle* (PGCC) was proposed as a good metric that combines speed and area properties for benchmarking competing circuit architectures that implement

the same task (in this case the SA-IDCT). Table 4.4 shows that the proposed SA-IDCT architecture improves upon the Chen architecture [109] in terms of PGCC by an order of magnitude ( $5.2 \times 10^6$  versus  $3.9 \times 10^7$ ). Benchmarking against the Hsu architecture [235] is less straightforward since that architecture can operate in zero skipping mode as described in Section 4.2.2. Also, Hsu et. al. do not mention specifically the computational cycle latency of their architecture. They do quote the average throughput in Mpixel/sec of their module in both no skip mode and zero skipping mode. From these figures, the author estimates that the cycle latency in no skip mode is 84 cycles and averages 10 cycles in zero skipping mode. Compared to the Hsu et. al. no skip mode, the proposed architecture, which does not implement zero skipping, improves upon the Hsu architecture in terms of PGCC by an order of magnitude ( $5.2 \times 10^6$  versus  $3.2 \times 10^7$ ). When comparing the proposed architecture against the zero skipping mode of the Hsu architecture, the Hsu architecture is slightly better, although the results have the same order of magnitude ( $5.2 \times 10^6$  versus  $3.8 \times 10^6$ ). However, since the proposed architecture represents an order of magnitude improvement when both are in no skip mode, it is reasonable to assume that in the future if a zero skipping mode is incorporated into the proposed architecture, it will also improve on the zero skipping mode of the Hsu architecture. A zero skipping extension of the proposed architecture is a clear future research task for the author, and this is discussed in Chapter 6. However, it must be noted that the gate count of the current implementation of the proposed design is much smaller than the Hsu architecture (27518 versus 377685).

#### 4.5.2.2 Power Consumption and Energy Dissipation

As discussed in Chapters 2 and 3, some approximate normalisations must be used to aid power consumption benchmarking of hardware accelerators from the available parameters generally quoted in the literature (power, voltage, frequency and process technology). The benchmarking outlined in this section is based on the normalisation formulae outlined in Section 2.3.2.5 of Chapter 2). To analyse the power consumption of the proposed SA-IDCT IP core, the method described in Section 2.3.2.4 is followed. In summary, this involves a back-annotated dynamic simulation of the gate level netlist. This netlist is generated by the synthesis tool for TSMC 0.09 $\mu$ m TCBN90LP technology. The Synopsys Prime Power tool is used to analyse the annotated switching information from a VCD file.

As discussed in Chapter 2, the power consumption of a circuit is heavily dependent on the nature of the input stimulus. In the case of the SA-IDCT, typical input stimulus are readily available from the standard video test sequences (and segmentation masks) defined by the MPEG-4 Video Verification Model [209]. The set of 6 test sequences used for the SA-DCT power analysis experiments presented in Chapter 3 are also used for the SA-IDCT power analysis. Firstly, each of the 6 test sequences are transformed into their frequency domain representation using a double precision software implementation of the SA-DCT. Then, as defined by the MPEG-4 standard, the frequency domain coefficients are rounded to 12-bit integer precision. It is this 12-bit SA-DCT coefficient data that is used to as input data to the proposed SA-IDCT module along with the corresponding segmentation mask for power analysis simulations. Power analysis was carried out for each of the 6 test sequences. The power waveforms for each of these sequences are shown in Figure 4.13 to Figure 4.18. Just like the SA-DCT waveforms presented in Chapter 3, it is clear that the power consumption of the SA-IDCT architecture varies in proportion to the size of the object. For example, as the person in the "hall monitor" test sequence approaches the camera, the power increases until it suddenly drops when the person leaves the scene through a door. This is ex-

Table 4 5 SA-IDCT 90nm TSMC Power Consumption

Test Sequence	Average Power [mW]
Andy	0 404
Coastguard Object 1	0 465
Container Object 1	0 482
Hall Monitor Object 2	0 429
Sean	0 504
Weather	0 488
Random Data	0 686

pected since the proposed SA-IDCT architecture terminates processing earlier for smaller shape blocks. The average power for each simulation is given in Table 4 5, and the average power figure over all 6 test sequences is 0 46mW.

Another set of power analysis simulations were run using random data as stimulus. To achieve this, random  $8 \times 8$  pixel and alpha block pairs were generated using a software program. This random data is then transformed using a software implementation of the SA-DCT as described previously, and the resultant random coefficients are used as stimulus to the SA-IDCT block. As argued in Chapter 3, random data simulations generate an estimate for worst case power consumption. This is because unlike natural video data, there is a very low probability of adjacent pixel or shape correlation with random data. For natural video blocks, the SA-DCT decorrelates most of the information into relatively few low frequency coefficients and much of the high frequency coefficients are zero valued. In the case of random coefficient blocks, there is not as much decorrelation – hence the input stimulus used in the random simulations is more likely to generate circuit switching. To get an estimate for worst case power, 10 simulations were run using 50 random coefficient and alpha  $8 \times 8$  block pairs as stimulus, with different seeds used for the random number generators to guarantee 10 unique data sets. The power figure quoted in Table 4 5 is the average value over the 10 random data simulations. As expected, the value of 0 686mW is slightly higher compared to 0 46mW, which is the average power consumption over all the video test sequence simulations.

Two of the SA-IDCT implementations in the literature quote power consumption figures and the parameters necessary against which to perform normalised benchmarking: the architectures by Hsu et al [235] and Chen et al [109]. The normalised power, energy and Energy-Delay Product (EDP) figures are summarised in Table 4 6. Note that the energy figures quoted in the table are the normalised energies required to process a single opaque  $8 \times 8$  block. The proposed SA-IDCT architecture improves upon the Chen architecture in terms of normalised power, energy and EDP. The EDP results are particularly impressive (0 50 pJs versus 7 11 pJs). This gain is predominantly attributable to the much improved clock cycle latency (188 versus 1984). Compared to the Hsu architecture (in no skip mode), the proposed architecture is again better in terms of normalised power, energy and EDP. Table 4 6 shows that the Hsu architecture in zero skipping mode outperforms the current implementation of the proposed design (no zero skipping) in terms of Energy and EDP, despite the fact that the current implementation of the proposed design has a better normalised power consumption performance. This is a direct consequence of the reduced clock cycle latency achievable with a zero skipping scheme. As mentioned in Section 4 5 2 1, future work on the proposed SA-IDCT architecture will involve incorporating an appro-



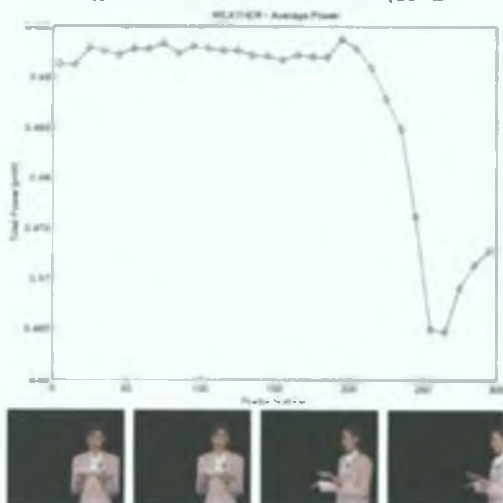
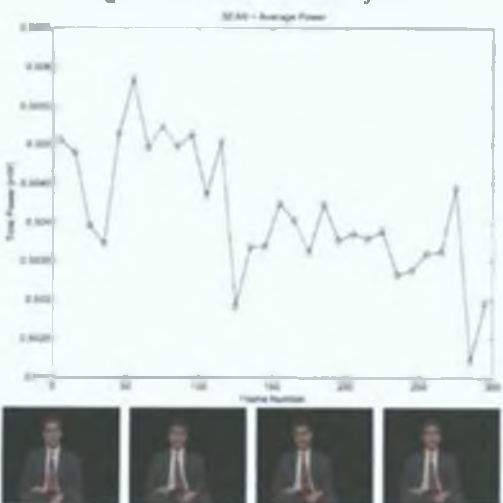
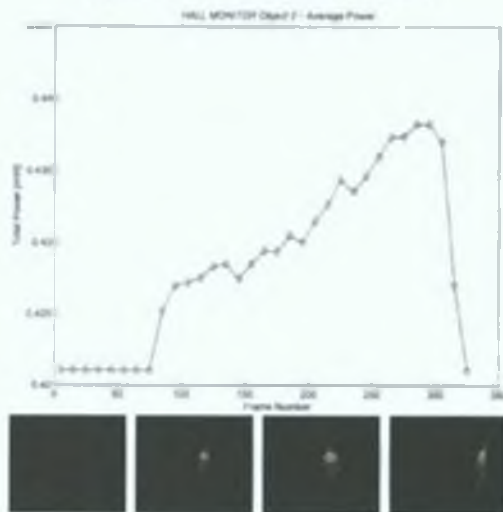
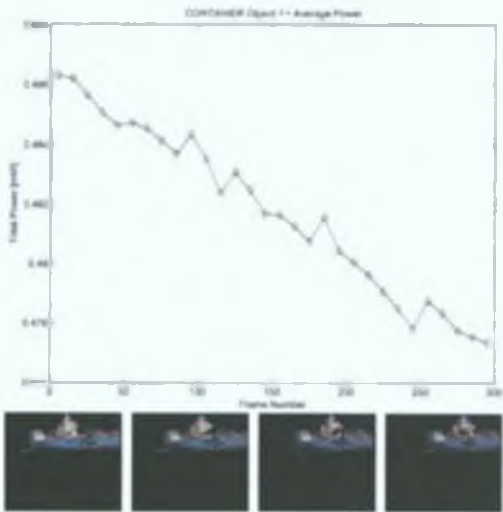
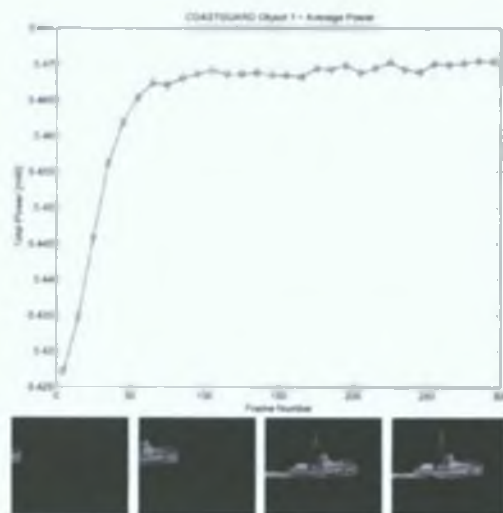
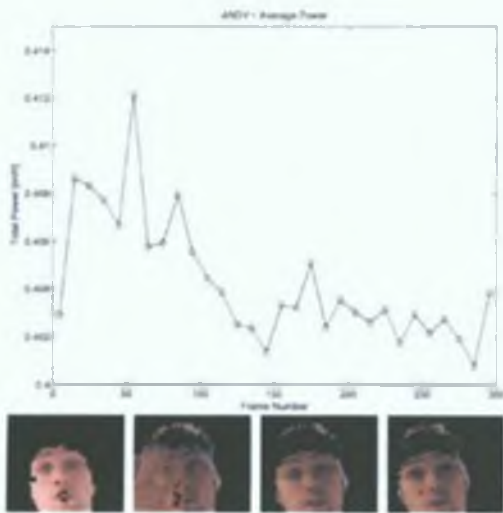


Table 4.6: Normalised SA-IDCT Power and Energy Benchmarking

Architecture	Process [ $\mu\text{m}$ ]	Voltage [V]	Power [mW]	Power (V,L) [mW]	Delay [Clks]	Speed [MHz]	Power (V,L,f) [mW]	Energy [nJ]	EDP [pJs]
Hsu[235]	<input type="checkbox"/>	0.18	1.8	467.04	84	62.5	467.04 <sup>a</sup>	627.70 <sup>a</sup>	0.84 <sup>a</sup>
				55.60	10		55.60 <sup>b</sup>	8.90 <sup>b</sup>	0.001 <sup>b</sup>
Chen[109]	<input type="checkbox"/>	0.35	1.2	12.44	1984	25	12.44 <sup>a</sup>	297.36 <sup>a</sup>	7.11 <sup>a</sup>
Proposed Approach	<input type="checkbox"/>	0.09	1.2	0.46	188	14	2.76 <sup>a</sup>	37.06 <sup>a</sup>	0.50 <sup>a</sup>
							2.76 <sup>b</sup>	37.06 <sup>b</sup>	0.50 <sup>b</sup>
							10.44 <sup>a</sup>	140.13 <sup>a</sup>	0.50 <sup>a</sup>

Operating Modes:   $\Rightarrow$  No Zero Skipping,   $\Rightarrow$  Zero Skipping Implemented

appropriate zero skipping scheme, as discussed in Chapter 6. It is expected that this future work will improve the performance of the proposed architecture significantly.

As is the case for the SA-DCT results presented in Chapter 3, it must be conceded that despite these normalised benchmarking figures, the validity of these figures is still open to debate. This is because the power figures for each of the target technologies are obtained using different power estimation tools and configurations. Since the switching activity in a module is dependent on its data load (and this is particularly true for SA-IDCT), the same testbench should really be used for fair comparisons when generating VCD files.

#### 4.5.2.3 Benchmarking Power & Energy Against Software

In Chapter 3, experimental results are presented that compare the power and energy performance of the proposed SA-DCT accelerator against the MPEG-4 Part 7 optimised software implementation of the SA-DCT [210] running on two embedded processors. The same set of experiments were conducted for the MPEG-4 Part 7 optimised software implementation of the SA-IDCT, and the results are presented in this section.

The first experiment involved taking a physical measurement of the current drawn by an ARM920T processor [211], and the steps taken are identical to the corresponding SA-DCT experiment outlined in Section 3.6.2.3. The ARM prototyping platform used is described in more detail in Section 3.6.4. The power waveform obtained for a single SA-IDCT iteration processing an opaque  $8 \times 8$  block is shown in Figure 4.19, and the average power is approximately 15mW. Interestingly, this power figure is the same as the ARM920T power figure estimated for the SA-DCT software implementation in Chapter 3. This similarity verifies the first order approximation by Sinha and Chandrakasan that the power dissipated by a processor is program independent, as discussed in Chapter 2. Using the same power and energy normalisations as before, the results presented in Table 4.8 clearly show that the SA-IDCT hardware architecture proposed in this thesis is much more efficient compared to a software implementation running on that ARM920T processor.

The second experiment leverages the JouleTrack tool, as introduced in Section 3.6.2.3 and described in detail in [103, 104]. The simple program as described earlier was uploaded to JouleTrack executing 1000 iterations of the SA-IDCT algorithm, and the results are shown in Table 4.7. The profiling results for the different processor cycle modes as well as a breakdown of leakage and switching power are illustrated in Figure 4.20. Again, the Sinha and Chandrakasan approximation is verified since the power figure of

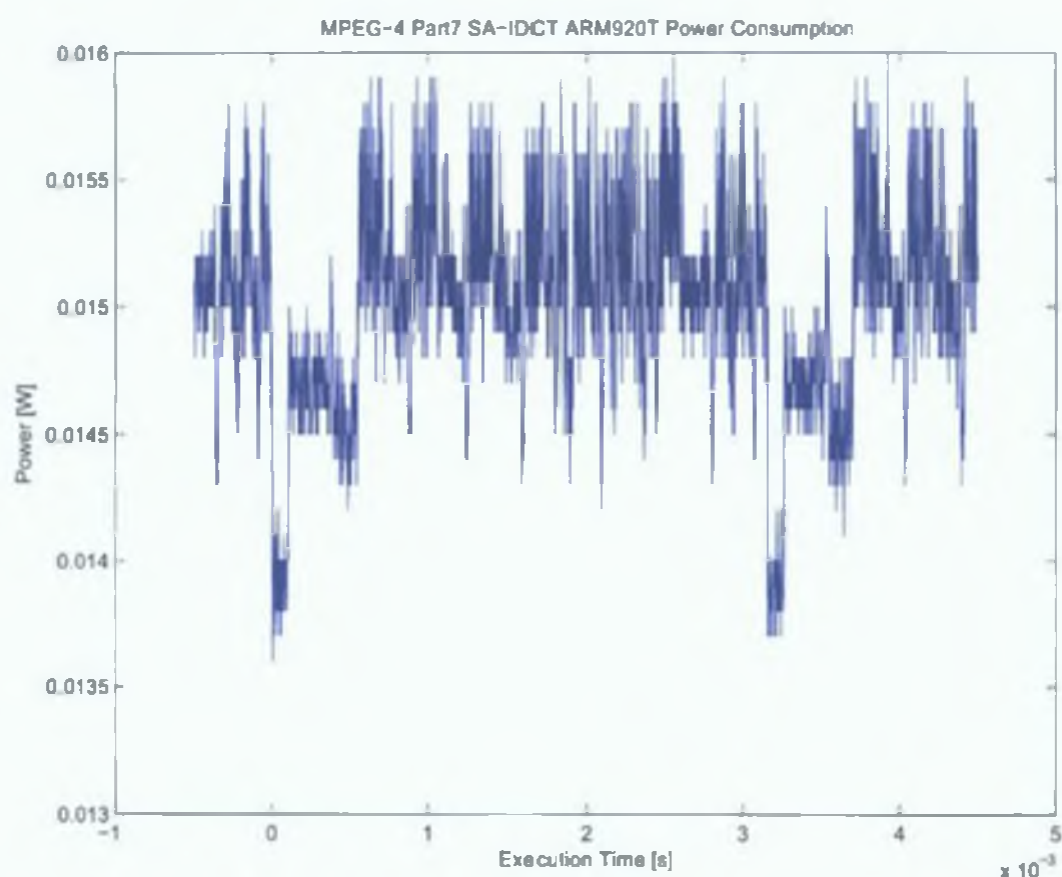


Figure 4.19: MPEG-4 Part7 SA-IDCT ARM920T Power Consumption

356mW is the same as the SA-1100 power figure estimated for the SA-DCT software implementation in Chapter 3. Table 4.8 illustrate that the proposed hardware SA-IDCT architecture is much more efficient compared to the reference software implementation executing on the SA-1100 processor.

As is the case for the SA-DCT software implementation discussed in Chapter 3, the SA-IDCT software implementation used for these experiments has not been optimised for an embedded processor like the ARM920T or the StrongARM SA-1100, despite being the official MPEG-4 Part 7 optimised software implementation. However, following the same arguments presented in Section 3.6.2.3, the results presented in Table 4.8 validate the energy efficiency of SA-IDCT architecture proposed in this thesis compared to a software implementation.

Table 4.7: Software SA-IDCT StrongARM SA-1100 Energy Statistics

<b>Operating Frequency</b>	206 MHz
<b>Operating Voltage</b>	1.5 V
<b>Execution Time</b>	442.857 $\mu$ s
<b>Average Switching Current</b>	0.2046 A
<b>Average Leakage Current</b>	0.0330 A
<b>Total Energy</b>	157.83 $\mu$ J
<b>Average Power</b>	356 mW

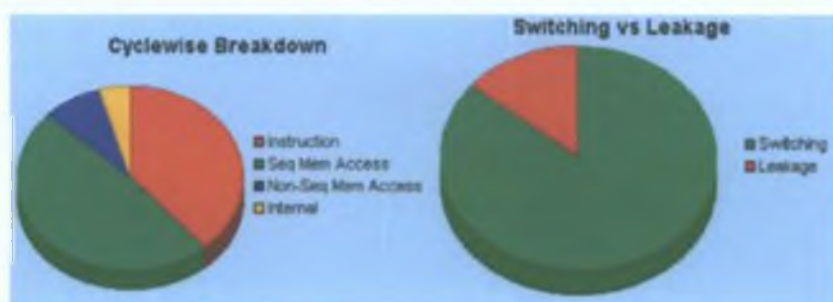


Figure 4.20: MPEG-4 Part 7 Fast SA-IDCT Energy Characteristics on StrongARM SA-1100

Table 4.8: Normalised Power and Energy Benchmarking Against Software SA-IDCT

	Process [ $\mu\text{m}$ ]	Voltage [V]	Power [mW]	Power ( $V_L$ ) [mW]	Delay [ $\mu\text{s}$ ]	Speed [MHz]	Power ( $V_L, f$ ) [mW]	Energy [nJ]	EDP [pJs]
ARM920T	0.25	2.5	15.00	15.00*	3100*	140	15.00*	$47 \times 10^{3*}$	$150 \times 10^{3*}$
SA-1100	0.35	1.5	356.00	356.00*	442*	206	356.00*	$158 \times 10^{3*}$	$70 \times 10^{3*}$
Proposed Approach	0.09	1.2	0.46	7.40*	14*	14	7.40*	99*	1.33*
				8.70*	14*		68.70*	117*	1.57*

### 4.5.3 Conformance Testing

The SA-IDCT implementation proposed in this chapter must pass the same conformance tests applied to the proposed SA-DCT implementation discussed in Chapter 3. These tests validate that the bitstream produced by the hardware accelerator is identical to that produced by the MPEG-4 reference software [212]. For the proposed SA-IDCT implementation, the MPEG-4 reference software was compiled with the SA-IDCT software replaced by a user defined API call to the SA-IDCT hardware accelerator residing on an FPGA. Further details on the MPEG-4 conformance platform (proposed by the University of Calgary) may be found in [213] and more details than practical to include here on the proposed SA-IDCT integration process (including hardware and software APIs) may be found in [236].

#### 4.5.3.1 Conformance Hardware System

The SA-IDCT module has been integrated with an adapted version of the multiple IP core hardware accelerated software system framework developed by the University of Calgary (UoC) [217]. The hardware is implemented on an Annapolis WildCard-II PCMCIA FPGA prototyping platform [218]. The block level structure of the system as shown in Figure 3.41 in Chapter 3 is also leveraged to integrate the SA-IDCT module. The MPEG-4 software runs on a host laptop Pentium4 processor and transfers data to and from the 2MB SRAM on the WildCard-II. This is done across the PCI bus via a DMA controller. The SA-IDCT core is a memory mapped peripheral, and is activated by the host software by writing configuration data to a specific address in memory. This write is detected by the master socket block and a strobe is issued by the interrupt controller to the specific SA-IDCT IP core hardware module controller (HWMC).

For integration with the UoC framework, a HWMC specific to the SA-IDCT IP core is required to control activation of the IP core and memory transfers to and from it. The HWMC shields the IP core

from the WildCard-II platform specific interface signals (PCI and SRAM) making it easy to port the core to another platform (e.g. ARM based as in Section 3.6.4). The SA-IDCT has a variable load depending on the shape information and the SA-IDCT core has been designed such that it finishes processing smaller video object plane (VOP) blocks earlier. The SA-IDCT HWMC has been designed with this in mind so that the SA-DCT relinquishes control of the SRAM at the earliest possible moment.

Due to the nature of the SA-IDCT computation, the IP core will not respond with output data until it has sampled all 64 bytes of the alpha block and *opaque\_pels* number of coefficients from the coefficient block memory. The variable *opaque\_pels* represents the number of object pixels in the current block belonging to the object. Hence the SA-IDCT HWMC processing latency could be reduced by implementing some prefetching and buffering of the next data block while waiting for the IP core to respond for the current block. This idea is essentially the same as the prefetching idea mooted in Section 3.6.3 for enhancing the SA-DCT HWMC. This investigation is targeted as future work.

#### 4.5.3.2 MPEG-4 Reference Software API

The steps involved to integrate the SA-IDCT hardware accelerator with the MPEG-4 optimised reference software (Part 7) [210] are very similar to the steps followed for the SA-DCT accelerator (see Chapter 3). This integration is required to validate functionality and pass conformance testing according to MPEG-4 Part 9 requirements. For conformance testing of hardware, ideally the software should require minimal modification for rapid verification. For the SA-IDCT accelerator two minor modifications are required (see [236] for more details). The first modification involves editing the configuration steps of the MPEG-4 software to also program the WildCard-II FPGA fabric with the SA-IDCT accelerator, and allocate and bind the DMA pointers. This allows both the application software and hardware accelerator to read/write in the same area of the WildCard-II SRAM transparently. The second modification is in the *ClmSADCT* C++ class in file *sadct.cpp*. A new pre-processor directive is added to allow the codec to use either the SA-IDCT software algorithm or SA-IDCT accelerator on the WildCard-II. By only making these straightforward changes, the minimal modification approach described in Chapter 3 is followed for the SA-IDCT conformance integration to facilitate rapid validation of the hardware accelerator with the reference software.

#### 4.5.3.3 Conformance Results

End to end conformance has verified that the bitstreams produced by the encoder with and without SA-IDCT hardware acceleration are identical. The test vectors used were 39 of the CIF and QCIF object-based test sequences as defined by the MPEG-4 Video Verification Model [209]. Synthesis results for the WildCard-II platform are shown in Table 4.9. The modified University of Calgary integration framework takes up only approximately 11% of the FPGA resources leaving almost 90% for hardware accelerators. The SA-DCT along with its HWMC require 27%. The post place and route timing analysis indicates a theoretical operating frequency of 75.3MHz so the IP core is able to handle real time processing of CIF sequences quite comfortably (assuming the same 14MHz constraint as before). Post place and route simulations were carried out on the netlist for VCD power analysis. These simulations were run at 75.3MHz (high throughput mode) and at 14MHz (low power mode). For each mode, 10 simulations were run using 50 random coefficient and alpha  $8 \times 8$  block pairs as stimulus, with different seeds used for the

Table 4.9: SA-IDCT FPGA Synthesis Results

Target	Module	Area [Gates]	$f_{max}$ [MHz]	Power [mW]		Throughput [Mpixels]		* [ns]	CLB Slices	Block RAMs
				$f_{max}$ (14MHz)	(14MHz)	$f_{max}$ (14MHz)	(14MHz)			
WildCard-II	SA-IDCT	42787	75.3	301.13	56.34	25.63	4.77	0	2763 (19%)	0
Virtex-II XC2V3000	SA-IDCT HWMC	20650	97.8	n/a	n/a	n/a	n/a	0	1196 (8%)	0
	Mod. UoC Fw.	102085	77.6	n/a	n/a	n/a	n/a	0	1627 (11%)	1
Integrator/CP	SA-IDCT	37321	43.1	348.19	113.85	33.06	4.77	n/a	2204 (11%)	0
Virtex-E XCV2000E	SA-IDCT HWMC	20945	75.3	n/a	n/a	n/a	n/a	n/a	1227 (6%)	0
	ARM VS	7020	89.7	n/a	n/a	n/a	n/a	n/a	381 (1%)	0

random number generators to guarantee 10 unique data sets. The power figures quoted in Table 4.9 are the average value over the 10 simulations for each mode. Random data is useful in this case to estimate a value for worst case power consumption, as discussed in Section 4.5.2.2.

#### 4.5.4 System Prototyping

In Section 3.6.4 of Chapter 3, it is argued that although the WildCard-II platform is useful for rapid validation and conformance testing, the system itself does not represent a realistic architecture for a mobile embedded system. Hence Section 3.6.4 proposes an "ARM virtual socket" prototyping platform built around an ARM processor and the AMBA bus architecture, since the ARM family is the processor of choice for wireless embedded systems. The proposed SA-IDCT IP core has been instantiated as a "virtual component" in this platform, as discussed subsequently.

##### 4.5.4.1 ARM Virtual Socket Hardware

A system block diagram showing the ARM virtual socket hardware (implemented on the Virtex-E FPGA) is shown in Figure 3.42 in Chapter 3. The virtual component block encompasses a single hardware accelerator IP core (in this case the SA-IDCT), and its associated HWMC. When strobed by a software API call, the SA-IDCT HWMC initiates SRAM transfers to and from the IP core and has similar behaviour to the HWMC developed for the WildCard-II platform (outlined in Section 4.5.3.1) but with an APB interface. Further details on the ARM virtual socket may be found in [216, 221].

##### 4.5.4.2 ARM Virtual Socket Software API

The same software API presented in Section 3.6.4 of Chapter 3 is used to allow transparent interaction for ARM compiled application software with the SA-IDCT hardware accelerator. Further details on the ARM virtual socket software API may be found in [216, 221].

##### 4.5.4.3 Prototyping Results

The synthesis results in Table 4.9 illustrate that the SA-IDCT IP core also meets the real time frequency constraint of 14MHz on the ARM virtual socket platform. The maximum possible operating frequency of the SA-IDCT core on this FPGA is 43.1MHz. Post place and route simulations were carried out on the netlist for VCD power analysis. These simulations were run at 43.1MHz (high throughput mode) and at 14MHz (low power mode). For each mode, the random data configurations were used as described previously. The power figures quoted in Table 4.9 are the average value over the 10 simulations for each

mode. These results show that in low power mode, the worst case power consumption is approximately 113.85mW.

## 4.6 Summary of Contributions

This chapter described the SA-IDCT algorithm in detail with special emphasis on the differences between it and the forward SA-DCT. The SA-IDCT is effectively the reverse process of the SA-DCT, and this means data addressing is more challenging. A comprehensive survey of prior art implementations of the  $8 \times 8$  IDCT is provided, with emphasis on techniques that exploit the fact that much of the coefficient data processed by the IDCT is zero valued (thanks to the decorrelation of the DCT process and subsequent quantisation). Prior art hardware implementations of the SA-IDCT are then discussed in depth. Based on this discussion, it is concluded that just like the SA-DCT, some of the prior SA-IDCT implementations do not address all of the required processing steps, and none have been designed specifically with low energy in mind.

Then, the author's proposed SA-IDCT architecture is described in detail. In Chapter 2 it was concluded that most energy savings are achievable at the high levels of design abstraction, so the proposed SA-IDCT has been designed accordingly. The primary low energy features include efficient data addressing and reconstruction, data dependent clock gating, multiplier-free datapath, balanced delay paths and hardware resource re-use.

The chapter concludes by benchmarking the proposed SA-IDCT architecture against the prior art, where possible. In terms of area and latency, the PGCC metric shows that the proposed architecture outperforms the Chen [109] architecture and the Hsu [235] architecture (in no skip mode). Considering normalised energy and power, the proposed architecture also outperforms the Chen [109] architecture and the Hsu [235] architecture (again in no skip mode). However, the Hsu architecture also has a zero skipping mode that exploits the likelihood of zero valued coefficient data to reduce computational latency. Comparing the zero skipping mode of the Hsu architecture against the proposed SA-IDCT architecture, which has no zero skipping mode, it has been shown that the Hsu architecture in this mode slightly outperforms the proposed design in terms of PGCC and significantly in terms of energy and EDP. Interestingly, the proposed architecture is still better in terms of normalised power despite the zero skipping mode of Hsu. However, since the proposed design is better for all metrics with both in no skip mode, it is expected that in future if a zero skipping mode is integrated into the proposed design, it will improve upon the zero skipping mode of the Hsu architecture. This is because the savings achievable due to zero skipping are data dependent and independent of the architecture. As well as benchmarking against the prior art, the proposed SA-IDCT has also undergone conformance testing via the MPEG-4 Part 9 initiative, validating that it is compliant with official MPEG requirements.

In addition to the development of a zero skipping extension to the proposed SA-IDCT architecture, future work could include the development of a  $\Delta$ DC-SA-IDCT post-processing module as required for MPEG-4 intra-VOP boundary blocks (see Chapter 6 for details). Similar to the SA-DCT architecture discussed in Chapter 3, a future research task is to investigate different parallel variations and the trade-offs involved between processing latency and power consumption.

## 5.1 Introduction

The multiplication of variable data by constant values is a common operation required by many signal processing algorithms and applications [132]. Indeed it has been shown that for a survey of more than two hundred industrial examples, over 60% have greater than 20% operations that are multiplications with constants [40]. Some commonly used tasks that involve such operations include digital filters and linear transforms such as the Discrete Fourier Transform (DFT) and the Discrete Cosine Transform (DCT). Since these operations are so common, efficient optimisation of hardware architectures to implement these operations is key to improving the performance of the algorithms and applications that use them. The DFT, DCT and indeed the SA-DCT/IDCT discussed in detail in previous chapters, are instances of a more generalised problem – that of a linear transform involving a constant matrix multiplication (CMM).

The general properties of a CMM operation can be exploited by a designer to realise an efficient hardware implementation. Given that one operand of a multiplication is constant, it is possible to implement the operation using only additions and or subtractions coupled with shift operations instead of implementing a full multiplier. The multiplication by constant problem has been widely researched and there are many optimisation algorithms in the literature, as surveyed in Section 5.3. The CMM optimisation problem itself seems simple but in reality is very difficult due to the huge combinatorial possibilities. As a result, most prior art sacrifice optimality by using heuristics to converge on a local optimum. These heuristics are usually “greedy” in nature meaning that the algorithm is steered such that it may exclude an area of the search space that contains the absolute optimum solution. Since it builds potential solutions in parallel, the algorithm proposed in this chapter guarantees (in theory) an absolute optimal solution without resorting to exploring the entire permutation space by using some intuitive early exit strategies. Such an algorithm can be employed to realise optimal circuitry for any CMM problem. In reality, due to the huge permutation space and computational tractability limitations it is impossible to search the entire search space in a reasonable amount of time even with the early exit strategies proposed. This has led to the proposal of a genetic algorithm that searches the permutation space intelligently to converge on the optimal result relatively quickly. Mutation strategies have been built in to the algorithm to avoid getting stuck in local optima.



## 5.2 Multiplication by Constant Applications

As mentioned previously, multiplication by constants is a prevalent task in many applications. The task itself is used in many different guises depending on the application and a recent paper by Dempster and Macleod classifies them into four sub-categories [134]. These categories are single coefficient multipliers, constant multiplier blocks, digital filters and constant matrix multipliers (CMMs). All four are summarised here to illustrate where the CMM problem sits in relation to the others.

### 5.2.1 Preliminaries

The work presented in this chapter is built upon the theory of adder-based distributed arithmetic as described in Section 3.2.1.7 of Chapter 3. Distributed arithmetic substitutes all multiplications by constants with a number of shifts and additions/subtractions (we refer to both as “additions”). Restating Equation 3.6, a general  $|P + 1 - Q|$ -bit 2’s complement constant  $a$  may be expressed as follows:

$$a = -b_P 2^P + \sum_{k=Q}^{P-1} b_k 2^k, \quad b_k \in \{0, 1\} \quad (5.1)$$

where bit position  $Q$  is the LSB and bit position  $P$  is the MSB. Bit position  $P$  is also the sign bit. For notational convenience in this chapter, let  $|P + 1 - Q| = M$  and re-align bit position 0 to be the LSB and bit position  $M - 1$  to be the MSB and sign bit without loss of generality. So the above equation can be re-stated as follows:

$$a = -b_{M-1} 2^{M-1} + \sum_{k=0}^{M-2} b_k 2^k, \quad b_k \in \{0, 1\} \quad (5.2)$$

In Section 3.2.1.7, the constants are assumed to be 2’s complement numbers. However, such constants may be expressed in different number systems. For example, consider the radix-2 signed digit (SD) numbering scheme with digit set  $\{\bar{1}, 0, 1\}$  where  $\bar{1} \equiv -1$ . With  $M = 4$ , the constant  $(-3)_{10}$  can be represented as either  $(00\bar{1}\bar{1})_2, (0\bar{1}01)_2, (\bar{1}101)_2, (0\bar{1}1\bar{1})_2, (\bar{1}11\bar{1})_2$ . The reasons for considering such a number system will become clear when the proposed design algorithm is outlined in this chapter. Note that with the radix-2 SD system there is no specific sign bit, the sign of the overall constant is spread among the distributed weights  $b_k$ . So restating the previous equation we have:

$$a = \sum_{k=0}^{M-1} b_k 2^k, \quad b_k \in \{\bar{1}, 0, 1\} \quad (5.3)$$

All subsequent discussions in this chapter assume that the constants are  $M$ -bit radix-2 SD numbers without loss of generality. Two other terms used when describing the SD representation of a constant throughout this chapter are **Canonic Signed Digit (CSD)** and **Minimal Signed Digit (MSD)**. Park et al. clearly define the distinction between CSD and MSD [237]. They say that given a constant, the corresponding CSD representation is unique and has two properties, the first is that the number of non-zero digits is minimal and the second is that two non-zero digits are not adjacent. They define MSD to be a superset of CSD where the first property still holds but the second is relaxed.

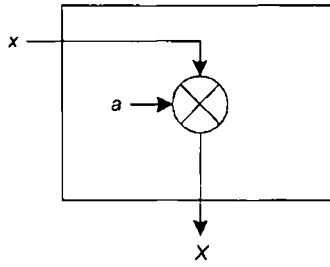


Figure 5.1 Single Constant Multiplier Behavioural Architecture

### 5.2.2 Single Constant Multipliers

A single constant multiplier is defined by Equation 5.4 where  $a$  is a constant scalar multiplicand and  $x$  is the input scalar multiplier. Since  $a$  is a constant, Equation 5.3 may be substituted into Equation 5.4 giving Equation 5.5. Equation 5.5 is expressed in matrix form in Equation 5.6 where  $\mathbf{a}_d = [b_0, b_1, \dots, b_{M-1}]^T$  are the distributed signed digits of  $a$  and the corresponding elements of  $\mathbf{t} = [2^0, 2^1, \dots, 2^{M-1}]$  represent their weights.

$$X = ax \quad (5.4)$$

$$X = \sum_{k=0}^{M-1} 2^k b_k x \quad (5.5)$$

$$X = \mathbf{t} \mathbf{a}_d x \quad (5.6)$$

A behavioural architecture is shown in Figure 5.1. The optimisation challenge in this case is to search for patterns in  $\mathbf{a}_d$  that minimises the number of adders, shifts, latches or any other criterion deemed important. For example consider  $a = 85, M = 8$ . Therefore  $\mathbf{a}_d = [01010101]^T$ . Direct implementation of  $X$  requires 3 adders ( $X = x2^0 + x2^2 + x2^4 + x2^6$ ), however considering the sub-expression  $S_1 = x[1 + 2^2]$ ,  $X$  can be implemented with only 2 adders ( $X = S_1 + S_1 2^4$ ).

### 5.2.3 Constant Multiplier Blocks

A constant multiplier block is a simple extension to the single constant multiplier except there are  $N$  products with the input data  $x$  simultaneously. This is shown in Equation 5.7 where  $\mathbf{A}, \mathbf{X}$  are now  $N$ -point 1D vectors. The distributed signed digits  $b_{jk}$  for each of the  $N$  constants are represented by a 2D matrix  $\mathbf{A}_d$  with each column  $j$  representing a particular  $M$ -bit constant  $a_j$  (Equation 5.9).

$$\mathbf{X} = \mathbf{A}x \quad (5.7)$$

$$\mathbf{X} = \mathbf{t} \mathbf{A}_d x \quad (5.8)$$

$$\mathbf{X} = \begin{bmatrix} 2^0 \\ 2^1 \\ \vdots \\ 2^{M-1} \end{bmatrix}^T \begin{bmatrix} b_{0,0} & b_{N-1,0} \\ b_{0,1} & b_{N-1,1} \\ \vdots & \vdots \\ b_{0,M-1} & b_{N-1,M-1} \end{bmatrix} x \quad (5.9)$$

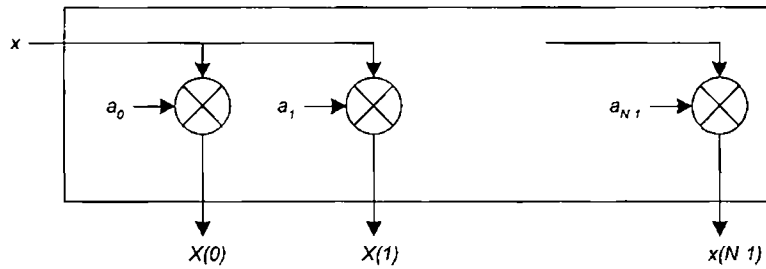


Figure 5.2 Constant Multiplier Block Behavioural Architecture

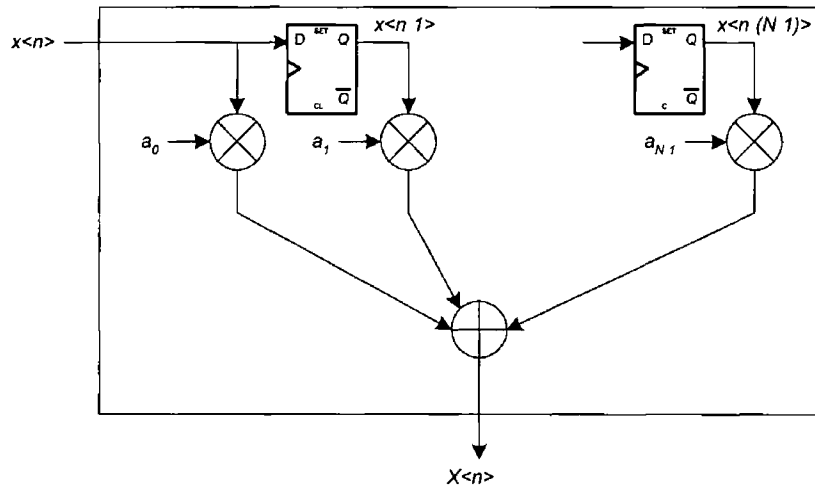


Figure 5.3 FIR Filter Behavioural Architecture

A behavioural architecture is shown in Figure 5.2. The optimisation challenge in this case is to search for vertical patterns across the 2D matrix  $\mathbf{A}_d$  that aid minimisation of the optimisation criterion.

### 5.2.4 Digital Filters

A digital filter is a more complicated scenario that involves a sum of products of a vector of delayed “versions” of a multiplicand (essentially a shift register) with a vector of constants. This kind of filter is referred to as a Finite Impulse Response (FIR) filter. An alternative scenario is an Infinite Impulse Response (IIR) filter which includes delayed versions of previous outputs (i.e. feedback) as well as delayed versions of the input multiplicand. The following discussion relates to FIR filters for illustration but may be generalised to cover the IIR case. An  $N$ -point FIR filter may be described by Equation 5.10. In this notation  $x\langle n-j \rangle$  represents sample  $x\langle n \rangle$  delayed by  $j$  clock cycles. In this application, matrix  $\mathbf{A}$  is again a 1D  $N$ -point vector of constants  $a_j$ . Matrix  $\mathbf{A}$  has a 2D matrix distributed form  $\mathbf{A}_d$  as before as shown in Equation 5.12. A behavioural architecture is shown in Figure 5.3.

$$X \langle n \rangle = \sum_{j=0}^{N-1} a_j x \langle n - j \rangle \quad (5.10)$$

$$X \langle n \rangle = \sum_{j=0}^{N-1} a_j q^{-j} x \langle n \rangle \quad (5.11)$$

$$X \langle n \rangle = \begin{bmatrix} 2^0 \\ 2^1 \\ \vdots \\ 2^{M-1} \end{bmatrix}^T \begin{bmatrix} b_{0,0} & b_{N-1,0} \\ b_{0,1} & b_{N-1,1} \\ \vdots & \vdots \\ b_{0,M-1} & b_{N-1,M-1} \end{bmatrix} \begin{bmatrix} x \langle n \rangle \\ x \langle n - 1 \rangle \\ \vdots \\ x \langle n - (N - 1) \rangle \end{bmatrix} \quad (5.12)$$

In this case, the optimisation challenge is to search for vertical, horizontal and diagonal patterns in  $\mathbf{A}_d$  that aid minimisation of the optimisation criterion. It is clear that the optimisation is becoming more challenging as the applications become more complex. Consider a sample 4 tap FIR ( $M = 12$ ) with matrix  $\mathbf{A}_d$  as shown in Figure 5.4. Step 1 and step 2 illustrate how patterns<sup>1</sup> are found in row, columns and diagonal directions in  $\mathbf{A}_d$ . Direct implementation of the FIR requires 3 registers and 15 adders. The resultant architecture after the patterns selected in Figure 5.4 needs 7 registers and 10 adders and is shown in Figure 5.5.

The example outlined illustrates a peculiarity of searching for patterns in FIR problems. Since the data vector is a shift register of delayed samples, step 1 in Figure 5.4 finds four instances of pattern  $S_1$ , two of which are  $S \langle n \rangle_1$  and two delayed by a clock cycle  $S \langle n - 1 \rangle_1$ . Assuming that a new  $X \langle n \rangle$  is computed every clock cycle, the data vector  $x$  for subsequent cycles is shifted by one location and an architecture with delayed versions of patterns is possible as shown in Figure 5.5. However, for a general dot product computation architecture at sample  $n$  the data vector  $x$  in Equation 5.12 could be independent of the previous vector for dot product at  $n - 1$ . This means that storing patterns such as  $S_1$  in registers may not make sense for such a general dot product computation engine.

It also must be noted that the order in which patterns are selected can greatly influence the resultant architecture. To illustrate this point more clearly, consider the simple example 4 tap FIR ( $M = 4$ ) in Figure 5.6. Although the filter itself is trivial, it serves to illustrate the point. Figure 5.6 shows two solutions for the order of selecting sub-expression patterns, and indeed these are not the only two solutions. Figure 5.7 shows the resultant architectures for both options. It is clear that solution 1 requires less hardware resources, but this is not immediately obvious from the sub-expression choices. In step one, both solutions choose a sub-expression that occurs four times and in step two, both choose a sub-expression that occurs twice. Despite this similarity, the resultant architectures are quite different. It is intuitive to see that in a real optimisation problem with a more complex filter, the choice of sub-expression patterns and the order in which they are chosen becomes a difficult problem to solve optimally. As discussed in Section 5.3.2, most prior art algorithms in the literature use heuristics to guide the decision process. However, there is no guarantee that the heuristic will produce the best result (whatever "best" is taken to represent), since choosing a pattern at a particular step will influence the potential patterns available in subsequent steps.

<sup>1</sup>For the moment we ignore how these patterns are selected

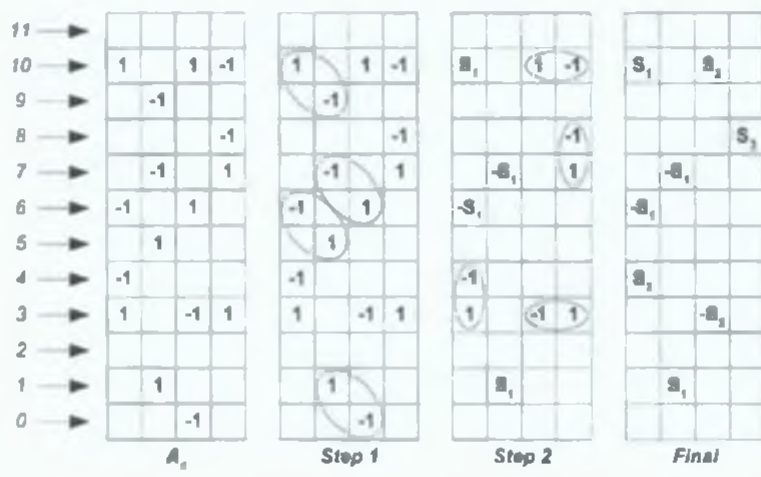


Figure 5.4: FIR Sub-Expression Selection Example

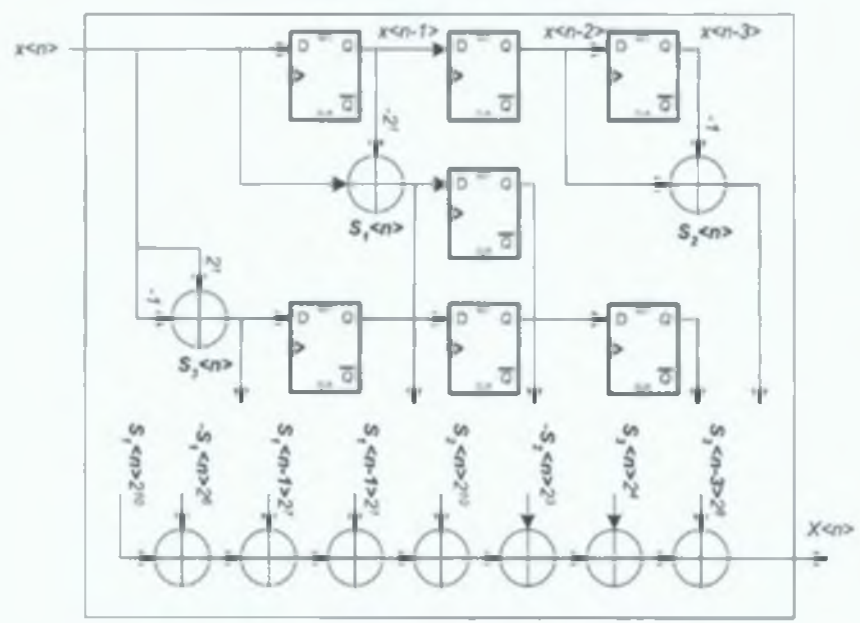


Figure 5.5: Example FIR Architecture Using Sub-Expressions

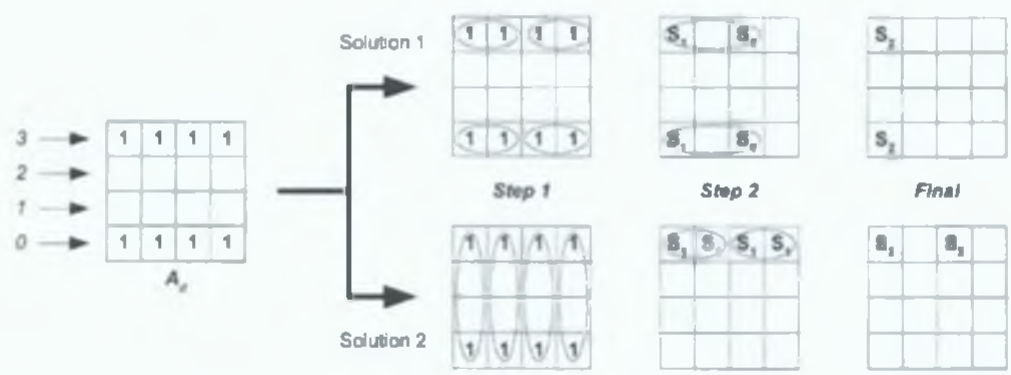


Figure 5.6: Simple FIR Sub-Expression Selection Options

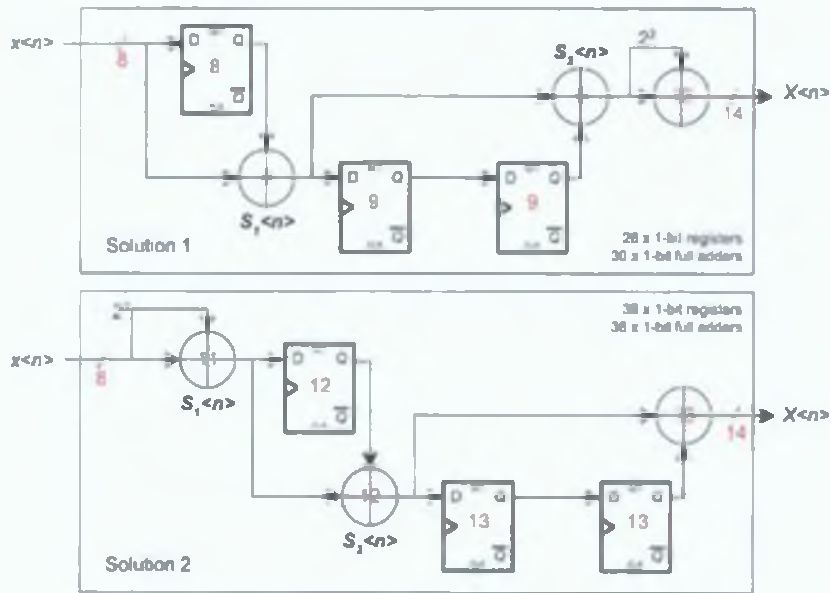


Figure 5.7: Simple FIR Sub-Expression Selection Options

### 5.2.5 The Constant Matrix Multiplication (CMM) Problem

A general CMM operation involves the linear transform of an  $N$ -point input data vector  $\mathbf{x}$  to an  $N$ -point output data vector  $\mathbf{X}$  as shown in Equation 5.13. The transformation itself depends on the values in the  $N \times N$  matrix  $\mathbf{A}$ . Such a CMM involves a set of  $N$  dot products as shown in Equation 5.14. Each of these dot products is formed using the data vector  $\mathbf{x}$  with one of the  $N$  rows of the matrix  $\mathbf{A}$ . The result for each dot product is stored in row  $i$  of the output vector  $\mathbf{X}$ , and the dot product at row  $i$  can be described by Equation 5.15. A behavioural architecture that implements dot product  $i$  is shown in Figure 5.8. To implement the entire CMM,  $N$  of these computation units are needed in theory.

$$\mathbf{X} = \mathbf{A}\mathbf{x} \quad (5.13)$$

$$\begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(N-2) \\ X(N-1) \end{bmatrix} = \begin{bmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,N-2} & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \dots & a_{1,N-2} & a_{1,N-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{N-2,0} & a_{N-2,1} & \dots & a_{N-2,N-2} & a_{N-2,N-1} \\ a_{N-1,0} & a_{N-1,1} & \dots & a_{N-1,N-2} & a_{N-1,N-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} \quad (5.14)$$

$$X(i) = \sum_{j=0}^{N-1} \sum_{k=0}^{M-1} b_{ijk} 2^k x_j, \quad i = 0, \dots, N-1 \quad (5.15)$$

The optimisation challenge in this case is similar to the FIR filter except that there are now  $N$  dot products (FIR has only one) and the values in vector  $\mathbf{x}$  are independent (in the FIR they are delayed versions of the input sample). It is intuitive to recognise how the optimisation of a CMM circuit is a difficult problem to solve since the number of potential patterns increases. The pattern possibilities increase because the distributed signed digit matrix for the CMM problem is actually a 3D matrix  $\mathbf{A}_d$  of digits  $b_{ijk}$  as clear from Figure 5.9. Proposed algorithms that seek an optimal solution based on some

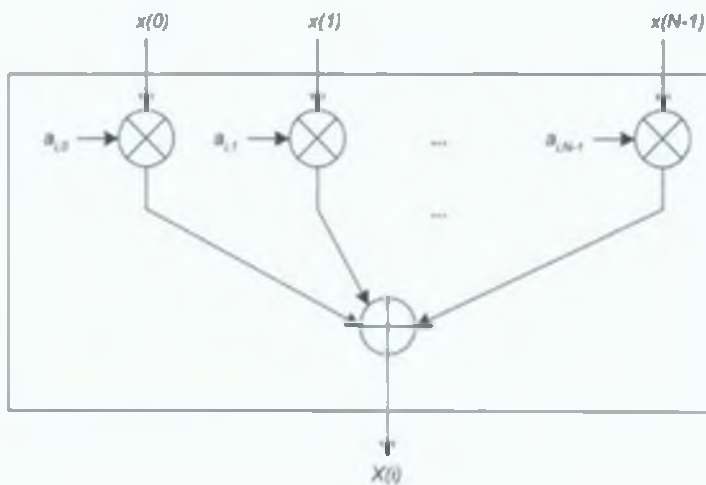


Figure 5.8: Dot Product Behavioural Architecture

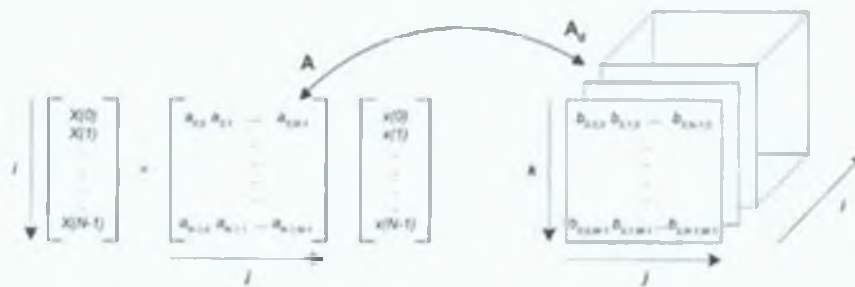


Figure 5.9: CMM Distributed Signed Digit 3D Matrix

criteria must be wary of getting stuck in local optima.

A special case of the CMM problem is the multiplication free CMM (MF-CMM), where the constants  $a_{ij}$  themselves in Equation 5.14 are limited to the values  $\{\bar{1}, 0, 1\}$ . Examples of an MF-CMM include the Discrete Walsh Transform, the Discrete Hadamard Transform or some error-correcting codes (e.g. Reed). With respect to optimisation, the matrix  $A_d$  of digits  $b_{ij,k}$  is really only a 2D matrix because the  $k$  dimension does not really exist (since effectively  $M = 1$  if the constants are limited to  $\{\bar{1}, 0, 1\}$ ). As such the MF-CMM problem resembles the constant multiplier block problem discussed in Section 5.2.3.

The 1D 8-point DCT as described by Equation 3.10 in Chapter 3 on the other hand is an example of a CMM computation. It involves 8 dot products of an input data vector with 8 orthogonal cosine basis functions. The 8-point 1D DCT will be used as a vehicle to outline the concepts explored in this work. For reference, the SA-DCT uses 1D  $N$ -point DCTs  $\forall N = 0, 1, \dots, 8$  increasing the number of potential dot products. Essentially the SA-DCT involves multiple related CMM problems (MCMM) and adds another layer of complexity to the optimisation requirements. An optimisation solution to a generic CMM problem is described in Section 5.4 and the extensions to deal with an MCMM problem are described in Section 5.5. The 8-point 1D DCT CMM computation will be used as a vehicle to illustrate the problem complexity and to explain the generic optimisation approach developed.

## 5.3 Multiplication by Constant State of the Art Review

This section explores in more detail some properties of the CMM problem and the associated impact on an optimisation algorithm for designing VLSI implementations of a CMM equation. Subsequently, a survey is given of state of the art design algorithms for the constant multiplication applications described in Section 5.2. Special emphasis is placed on algorithms that deal with the CMM problem specifically, as opposed to constant multiplier blocks or digital filters.

### 5.3.1 CMM – A Closer Look

The simplest optimisation criterion for the CMM problem is to find the optimal sub-expressions across all  $N$  dot products in Equation 5.15 that lead to the fewest adder resources. Three properties of solutions can be used to classify approaches: SD permutation, pattern search strategy and problem subdivision, and each of these properties are briefly reviewed below.

#### 5.3.1.1 SD Permutation

Consider that each of the  $N \times N$   $M$ -bit fixed point constants  $a_{ij}$  have a finite set of possible radix-2 SD representations. For example, it was shown in Section 5.2.1 that there are five radix-2 SD representations of the constant  $(-3)_{10}$  with  $M = 4$ . To find the optimal number of adders, all SD representations of  $a_{ij}$  should be considered since for a CMM problem CSD representation is not guaranteed to be optimal (as shown in Section 5.4.2.7 and 5.4.6 and as also mentioned in other publications [237, 238, 134]). The difficulty is that the solution space is very large [239], hence SD permutation has only thus far been applied to simpler problems [239, 240]. Potkonjak et al. acknowledge the potential of SD permutation but choose a single SD representation for each  $a_{ij}$  using a greedy heuristic. Neither of the recent CMM-specific algorithms in the literature apply SD permutation [241, 242], whereas the algorithm proposed in this thesis does. The approaches in the literature that use SD permutation are described in more detail in Section 5.3.2.

#### 5.3.1.2 Pattern Search

The goal of pattern searching is to find the optimal number of sub-expressions in the 3D bit matrix  $\mathbf{A}_d$  resulting in fewest adders. Usually the elements  $b_{ijk}$  of  $\mathbf{A}_d$  are divided into  $N$  2D slices along one of the planes. An example showing 2D slices along the  $i$  plane is shown in Figure 5.9. Slices along the  $i$  plane essentially mean that the CMM problem is divided into  $N$  dot product problems. Slices along the  $j$  plane mean that the CMM problem is divided into  $N$  constant multiplier block problems. Although not usually used, slices along the  $k$  plane mean that the CMM problem is divided into  $M$  MF-CMM problems. Patterns are searched for in the 2D slices independently before combining the results for 3D. An example  $i$  plane 2D slice for a particular SD permutation is shown in Equation 5.16. It is a simple 4-point dot product with  $M = 13$  constants  $\frac{1}{2}\cos(\frac{\pi}{16})$ ,  $\frac{1}{2}\cos(\frac{3\pi}{16})$ ,  $\frac{1}{2}\cos(\frac{5\pi}{16})$  and  $\frac{1}{2}\cos(\frac{7\pi}{16})$ . It is actually the dot product to compute coefficient 1 of the 8-point 1D DCT using even-odd decomposition (see



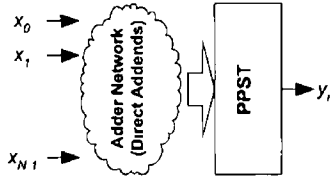


Figure 5 10 P1D Architecture

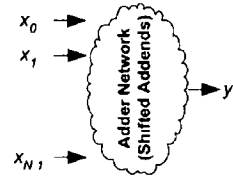


Figure 5 11 P2D Architecture

Section 3 11b for more details)

$$X(z) = \begin{bmatrix} 2^0 \\ 2^{-1} \\ 2^{-11} \\ 2^{-12} \end{bmatrix}^T \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & \bar{1} & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & \bar{1} \\ 0 & 0 & 0 & 0 \\ \bar{1} & 1 & 0 & 0 \\ 0 & 0 & \bar{1} & 1 \\ \bar{1} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \bar{1} & 0 & 0 \\ 0 & 0 & 1 & \bar{1} \end{bmatrix}}_{2D \text{ slice of } b_{i,j,k}} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2^0 \\ 2^{-1} \\ 2^{-11} \\ 2^{-12} \end{bmatrix}^T \begin{bmatrix} W_0 \\ W_{-1} \\ W_{-11} \\ W_{-12} \end{bmatrix} \quad (5 16)$$

The algorithm proposed in this thesis uses  $i$  plane 2D slices for reasons outlined in Section 5 3 3. As such, a 2D slice is taken to mean an  $i$  plane 2D slice unless explicitly stated otherwise. Algorithms may search for horizontal/vertical patterns (P1D) or diagonal patterns (P2D) in the 2D slice. The P1D strategy infers a two-layer architecture of a network of adders (with no shifting of addends) to generate distributed weights for each row followed by a fast partial product summation tree (PPST) to carry out the shift accumulate (Figure 5 10). The P2D strategy infers a one-layer architecture (Figure 5 11) of a network of adders that in general may have shifted addends (essentially merging the two layers of the P1D strategy). Potkonjak et al. use the P1D strategy and search for horizontal patterns, while Boullis and Tisserand use the P1D strategy and search for vertical patterns [242]. Dempster and Macleod use the P2D strategy [241]. However, all of these proposals select sub-expressions iteratively based on some heuristic criteria that may preclude an optimal realisation of the global problem. This is because the order of sub-expression elimination affects the results [243, 244]. The proposed algorithm sidesteps this issue by building parallel solutions using the P1D strategy as discussed in Section 5 4 2.

### 5 3 1 3 Problem Sub-Division

As in any hardware optimisation problem, synthesis issues should be considered when choosing sub-expressions for an  $N$ -point dot product (a 2D slice). If  $N$  is large (e.g. 1024-point FFT) then poor layout regularity may result from complex wiring of sub-expressions from taps large distances apart in

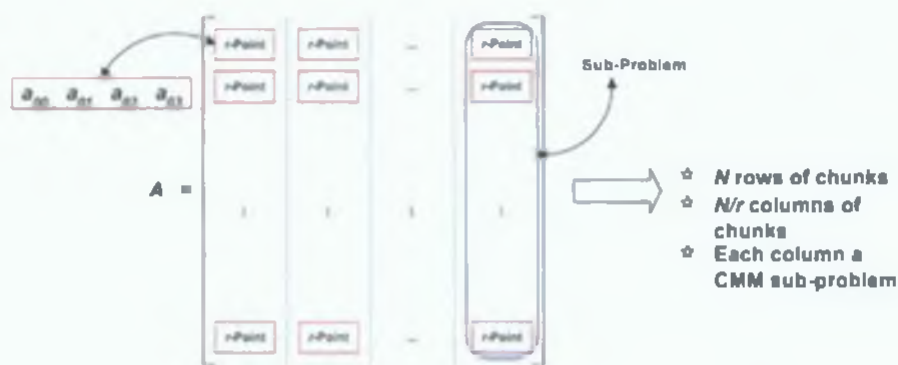


Figure 5.12: CMML Divide and Conquer

the data vector. Indeed a recent paper has shown that choosing such sub-expressions can result in a speed reduction and greater power consumption [245]. It is therefore sensible to divide each  $N$ -point dot product into  $N/r$   $r$ -point chunks and optimise each sub dot product independently. This approach is favoured by Paško et. al. to deal with large matrix problems [243]. The CMM problem hence becomes  $N/r$  independent sub problems, each with  $N$  dot products of length  $r$  (Figure 5.12). The optimal choice of  $r$  is problem dependent, but the proposed algorithm currently uses  $r = 4$  for reasons outlined in Section 5.4.2.

### 5.3.2 Optimisation Approaches

There are two basic types of design algorithms aiming to minimise a cost function such as adder count in a multiplication by constant application. On the one hand there are graph synthesis based algorithms and on the other there are common sub-expression elimination (CSE) algorithms. In their comparison work [134], Dempster et. al. conclude that for simpler problems graphical methods are best, while CSE works better for the more complex problems (such as CMM). In this section, some of the prominent graphical methods are reviewed as well as a more comprehensive survey of CSE approaches, as this is the approach adopted in this thesis.

#### 5.3.2.1 Graph Synthesis Based Approaches

Graph synthesis based approaches are founded on the idea that multiplication by a constant can be described by an acyclic graph. A sample graph for a multiplier block with constants 3,7,21 and 33 is shown in Figure 5.13. Each vertex (except the extreme left vertex) represents an adder and each edge represents a multiplication by a power of two (a hardware shift). The goal of graphical algorithms is to search for the optimal graph to implement the problem. If adder count is the criterion, the algorithm aims to find the graph with minimal vertices.

Bull and Horrocks [246] propose a set of graph-based heuristic algorithms for designing digital filter realisations. Their approach is commonly cited as the "Bull Horrocks Algorithm" (BHA). They prove that the problem is in general NP-complete by local replacement using the known NP-complete "ensemble computation" problem. Their algorithms do not claim to be optimal and search iteratively until enough partial summations exist so that all the filter coefficients are realised.

Dempster and Macleod consider all of the constant multiplier block multiplications together and

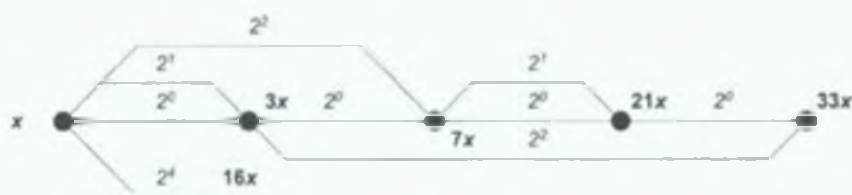


Figure 5.13: Sample Acyclic Graph of a Constant Multiplier Block

develop a graph-based approach to minimise the number of adders required assuming that precision of the constants has been pre-determined [247]. In contrast, early approaches such as [248] and [249] use the filter performance error as the optimisation function to derive word-length and representation of the constant values. This technique is commonly referred to as the “modified Bull Horrocks” algorithm (BHM). One advantage of the BHM over the BHA is that it considers values with a larger magnitude than the coefficient (e.g. coefficient 7 is obtained as  $7 = 8 - 1$  whereas BHA uses  $7 = 4 + 2 + 1$ ). Essentially this means that the BHM considers subtractors as well as adders. The BHM also has advantages since it initially factorises the constants into “fundamental” factors that are definitely required regardless of the final graph topology. These fundamentals are then used to build more effective partial sums and more flexible graph topologies. Dempster and Macleod also propose the n-Dimensional Reduced Adder Graph (RAG-n) algorithm in their paper [247]. They show that the RAG-n algorithm performs better than the BHM algorithm although it has a longer run-time and they also detail a hybrid between BHM and RAG-n allowing a trade-off between performance and computation time. Both the BHM and RAG-n algorithms are suitable for FIR and IIR optimisations when the constant vectors are being multiplied by the same data value.

Vena and Melodia [42] claim that graph synthesis algorithms are advantageous in terms of adder reduction but that CSE has the advantage in terms of logic depth. They propose a graph-based approach to alleviate this disadvantage in terms of logic depth and also further reduce the number of adders. The approach is similar to the BHM algorithm but is not disclosed in detail for patent and IP reasons.

### 5.3.2.2 Common Sub-Expression Elimination Based Approaches

CSE algorithms aim to minimise the cost function by combining pairs or groups of digits that appear in recurring patterns in SD representations of the fixed-point constants [134]. Each pattern is computed using a single adder, and then re-used repeatedly reducing the overall adder count. An example of CSE is outlined in Section 5.2.4 in the context of FIR filters. CSE algorithms are differentiated based on the way they choose sub-expressions (as hinted at in Section 5.3.1).

Potkonjak et. al. propose an iterative pairwise matching CSE algorithm targeting the constant multiplier block problem (Section 5.2.3) [40]. The algorithm has been generalised to address the CMM problem by decomposing an  $N$ -point linear transform CMM into  $N$  constant multiplier block problems. This is effectively done by firstly applying their algorithm to each of the columns of  $A$  in Equation 5.14 in isolation. Then the algorithm is applied to the results of the first step. The iterative pairwise matching algorithm itself iteratively searches for horizontal sub-expression patterns in  $A_d$  of Equation 5.9. The choice of sub-expression at each iteration is determined by a heuristic function that combines the immediate saving gained by the chosen sub-expression coupled with an estimate of likely savings in later

iterations by choosing this sub-expression. They also propose a scaling preprocessing transformation of the constants to further improve the results. Also mentioned is the use of alternative SD representations of  $\mathbf{A}_d$  based on a greedy algorithm that minimises the number of shifts, but details are not given. Matsuura et al. claim to optimise a variation on Potkonjak's approach but it is not clear exactly how the problem is being modelled from the detail given in the paper [250].

Hartley uses CSE but also considers the influence of a sub-expression choice on latch count as well as adder reduction [251]. He targets the FIR problem specifically, and uses the CSD representation of the constants. The algorithm searches for vertical, horizontal and diagonal pairwise patterns in the  $\mathbf{A}_d$  matrix of Equation 5.12. Hartley derives the latch count cost caused by choosing a sub-expression by running the scheduling algorithm in [252] to estimate the lifetime of the sub-expression. These ideas are extended in [253] where he derives a heuristic cost function for choosing the best sub-expression as a weighted difference of the number of adders saved and the additional latch count. This heuristic is adopted since repeatedly re-scheduling the circuit is a lengthy process. It is argued that all other factors being equal, when forming sub-expressions it is best to combine signals with overlapping lifetimes since otherwise, the lifetime of one of the signals must be extended. In fact it is commented that the cost function discriminates quite strongly against choosing diagonal pairwise patterns whose elements are in rows far away from each other in the  $\mathbf{A}_d$  matrix. This is because for a FIR problem many more latches are needed if such sub-expressions are chosen. When considering routability, Hartley proposes using only the two most common sub-expressions to avoid complex layout. It is shown that statistically the vertical patterns  $10\bar{1}$  and  $101$  in matrix  $\mathbf{A}_d$  occur most often if CSD encoding of the constants is used. Hartley also hints at the potential savings of using other representations that are not strictly CSD. Although in his paper Hartley does not consider the CMM problem, later papers (as outlined subsequently in this review) base their CMM algorithms on Hartley's ideas.

Paško et al. propose a steepest descent (greedy) approach when choosing the best matches for CSE [243]. They justify a greedy approach by claiming that CSE optimisation may be an NP-complete problem (although they state that the proof of this claim is still an open problem). The approach is targeted to the MF-CMM problem. They search for horizontal sub-expressions in the  $\mathbf{A}_d$  matrix and always choose the pattern with highest frequency (i.e. greedy selection). The algorithm does not limit itself to only searching for 2-bit sub-expressions and can search for any possible horizontal multi-bit patterns within the scope of the problem being optimised. The paper proposes simple iterative splitting of constant matrix to reduce run time for large problems acknowledging the fact that detection of patterns across boundaries will not be found. This "matrix-splitting" technique is proposed to reduce algorithm run time, but also makes sense from a synthesis perspective as discussed in Section 5.3.1. They outline extensions of their algorithm for the FIR and CMM problems but their approach is exactly the same as the work by Potkonjak et al. [40].

The non-recursive signed common sub-expression (NR-SCSE) algorithm proposed by Peiro et al. [245] aims to optimise in terms of adder count and logic depth simultaneously. Their algorithm specifically targets the FIR problem and they use the CSD representation of the constants. When surveying other algorithms they comment that graphical methods such as the BHM algorithm obtain the best adder count but the highest logic depth. In contrast, they claim that CSE algorithms, like Hartley's, require more adders but obtain the best logic depth and the results infer a regular layout. The NR-SCSE only considers vertical 2-bit patterns in the distributed  $\mathbf{A}_d$  matrix of Equation 5.12. This pattern search strat-

egy results in independent structures for each multiplier, and they claim that this simplifies both logic depth and number of adders. When choosing the best match they adopt a greedy selection approach. Peiro et al. also comment that sharing few common sub-expressions (i.e. using a low number of adders) will result in a large wiring overhead for large problems. This large fanout can have a detrimental effect on speed and power consumption since large capacitive loads are being switched.

The work by Boullis and Tisserand [132] concentrates on the CMM problem specifically although they have extended their work to cover the FIR problem also [242]. They acknowledge that the problem seems simple but in reality it is a “hard” problem due to its combinatorial properties. As more multiplications by constants become involved, the search space grows huge so usually some heuristics are required. Like Paško et al., the authors claim that the theoretical complexity of the problem is still unknown! The method proposed uses CSE and CSD encoding of the constants. The CSE method proposed by Boullis and Tisserand improves on the work by Paško et al. [243] using the Lefèvre algorithm [132] which allows more complex patterns to be used. The elements  $b_{ijk}$  of a CMM  $\mathbf{A}_d$  are divided into  $N$  2D slices along the  $j$  plane, essentially dividing the CMM problem into  $N$  constant multiplier block problems, although the paper does not clarify this clearly. They use a heuristic method of choosing the patterns at each iteration based on the pattern with maximal non-conflicts and minimal conflicts with other patterns. The authors hint at the fact that perturbing the fixed-point representations of the coefficients (i.e. adjusting the value of the constant slightly) may save adders at the expense of some additional quantisation noise that may be acceptable depending on the synthesis constraints and the quality requirements.

Vinod et al. concentrate on linear phase FIRs and discuss the trade-offs involved between searching for intra-coefficient patterns and inter-coefficient patterns [254, 255]. Intra-coefficient patterns refer to patterns along the  $k$  plane in the notation in this thesis. Inter-coefficient patterns refer to patterns along the  $j$  plane in the notation in this thesis. They do not mention diagonal patterns. With adder count and critical path as the optimisation criteria, they conclude that vertical pattern CSE is better than horizontal pattern CSE only when constant bitwidth is small. They use the Hartley “two most common” CSE approach [253, 255]. In [254], they propose a heuristic for searching for a certain subset of horizontal sub-expressions, followed by a subset of vertical sub-expressions to give a better overall result. Further variations of their work are presented in [256, 257].

While all the approaches listed thus far have concentrated primarily on the optimisation criterion and the sub-expression pattern search strategy, there has not been much emphasis on the potential benefits of trying different representations of the constant coefficients themselves. As illustrated in Equation 3.9, the representations of the constants has the most dominant influence on the number of adders required to realise the circuit, regardless of the pattern searching techniques used. The work by Potkonjak et al. hints at this but it is not explored in any depth [40].

The algorithm by Park and Kang generates all MSD representations for a given constant [237]. The authors acknowledge the potential savings using non-CSD representations and approximate that from a set of three numbers  $\{-1, 0, 1\}$  the average number of representations  $R_{avg}$  per  $n$ -bit number is as shown in Equation 5.17

$$R_{avg} \approx \frac{\left(\frac{3}{2}\right)^n}{2} \quad (5.17)$$

Equation 5.17 shows that  $R_{avg}$  increases exponentially with  $n$  and searching these permutations is a very “hard” problem. Park and Kang propose only searching the MSD representations to get a “good”

result in a reasonable computation time. They propose an algorithm for searching the MSD search space one coefficient at a time for the FIR filter problem. For a selection of 8 FIR filters, their algorithm outperforms Hartley, Potkonjak and Paško. They acknowledge that all coefficients cannot be considered simultaneously and try to counteract this by applying some iterative techniques. However, the selection approach is greedy and is not guaranteed to give optimal results.

A recent paper by Dempster et al. makes a very important point in regard to the representation adopted for the constant values in any general multiplication by constant problem [134]. In their paper, Dempster et al. publish results (in terms of adder count) for some small single multiplier and FIR filter problems using the RAG- $n$  graphical algorithm, the Hartley CSE algorithm [253] (referred to as “HH” in the paper since it uses CSD for the constants and searches for only horizontal intra constant sub-expressions) and their own HHS( $k$ ) algorithm (Hartley CSE using only horizontal intra constant sub-expressions for SD representations with at most  $k$  bits more than a MSD representation). They conclude that the HHS( $k$ ) gives good results – always at least as good as the HH algorithm and better in many cases compared to the RAG- $n$  algorithm. They state that as the number of constants increases, the number of SD representations grows exponentially so they prefer the RAG- $n$  algorithm in this case for computational complexity reasons. However, these results hint at the potential of unconventional SD representations of the constants if the associated computational complexity could be addressed. They cite another paper by Dempster and Macleod as the best candidate for the CMM problem as discussed subsequently in this review [241].

Dempster and Macleod propose an algorithm to generate SD representations for a given integer that can be generalised to any fixed-point number [238]. They state that any integer has an infinite number of SD representations. This is true, but for any real implementation there will be a finite number of possibilities due to the finite number of bits chosen to represent that integer. The algorithm they propose generates the SD representations hierarchically. In other words the algorithm searches for all  $n$ -bit solutions (if any) from  $n = 1$  up to whatever resolution the user desires. They give numerous examples highlighting the potential benefit of non-CSD representations for integer multipliers. For example, integer 363 has CSD representation  $10\bar{1}00\bar{1}0\bar{1}0\bar{1}$ , which has 5 non-zero digits requiring 4 adders to implement multiplication by 363. However, another SD representation 101101011 with 6 digits can be synthesised using three adders because of its patterns:  $3 = 2 + 1$  (pattern 11),  $11 = 3 + 8$  (pattern 1011),  $363 = 11(32 + 1)$ .

Dempster and Macleod have come up with a new method that leverages [238] to examine a subset of the SD representations when designing single constant multipliers [240] and FIR filters [239]. For the single constant multiplier problem they suggest using an extension of the Hartley [253] algorithm – namely the H( $k$ ) algorithm where the constants have at most  $k$  bits more than a MSD representation. Their results show that for 19-bit constants, the gains above  $k = 2$  are quite low, thereby posing the question – how far is worth searching? They state that they have a forthcoming publication with theoretical proofs showing that there is a limit to  $k$  worth searching. For the FIR filter problem they apply H( $k$ ) and illustrate the potential gains [239]. However they acknowledge the problem of exponential growth of the search space. They state that for Samueli’s filter [248] using H(1) there are  $6.4 \times 10^{64}$  different sets that need consideration and conclude that a sensible heuristic needs to be developed.

Another recent paper by Dempster and Macleod specifically targets the CMM problem using CSE [241]. This work is essentially an implementation of the Hartley CSE algorithm that searches for diagonal

patterns (the P2D strategy outlined in Section 5.3.1). The CMM constants are represented using CSD. When choosing the sub-expressions they choose the pair that occurs most often (a greedy approach). Experimental results show an adder cost improvement of 20–40% over Paško et. al. [243] and a 12–40% improvement over Potkonjak et. al. [40]. However, they do not compare favourably compared to the work by Boullis and Tisserand [242]. Although they do not leverage their SD search approach for the CMM problem, there is no reason to believe that they will not publish such work in the near future.

Dempster and Macleod present a survey of CSE algorithms for FIR filter applications in a recent paper [244] where they affirm that no proposed algorithm is optimal and the results depend on the length of the filters, their bitwidths and the actual coefficients involved. They also affirm that the order of sub-expression elimination affects the results obtained and that the search space is huge when one considers different SD representations. This motivates the work presented in this thesis.

### 5.3.2.3 Summary of Approaches

A detailed survey of multiplication by constant optimisation algorithms has been outlined. The most prominent proposals that encompass the CMM problem specifically are Potkonjak et. al. [40], Paško et. al. [243], Boullis et. al. [242] and Dempster et. al. [241]. The Boullis and Tisserand algorithm is essentially an extension of the Paško algorithm. Boullis and Tisserand also present a range of experimental results for different CMM problems and their results improve upon both Potkonjak's and Dempster's algorithms where valid comparisons can be made. However, none of these approaches explore SD permutation of the matrix constants to try to achieve further savings because of the huge permutation space, despite the reported validity of such a technique [237, 238, 134]. These proposals also select sub-expression patterns iteratively based on some heuristic criteria that may preclude an optimal realisation of the global problem. This is because the order of sub-expression elimination affects the results [243, 244].

The CMM problem is a difficult discrete combinatorial problem and currently requires a shift to a higher class of algorithms for more robust near-optimal solutions. This is because the current approaches are greedy hill-climbing algorithms and the associated results are very problem dependent [244]. The challenge is in the modelling of the problem to make it amenable to efficient computation. The algorithm proposed in this thesis models the problem in such a way as to make it amenable to so-called near-optimal algorithms, for example genetic algorithms (GAs), simulated annealing and tabu-search. A by-product of the modelling approach used is that the algorithm is very amenable to hardware acceleration. The proposed approach incorporates SD permutation of the matrix constants and avoids hill-climbing by evaluating parallel solutions for each permutation. Such an approach is computationally demanding but the algorithm has been modelled with this in mind and incorporates innovative fast search techniques to reduce this burden. Results show an improvement on state of the art algorithms with future potential for even greater savings (see Section 5.4.6). The concepts underpinning the proposed approach are introduced in Section 5.3.3 before a full discussion on the algorithm details in Section 5.4.

### 5.3.3 Proposed Optimisation Approach

As discussed in Chapter 2, the ever increasing computational burden of multimedia applications points to the need for dedicated hardware acceleration solutions – IP cores that implement the most demand-

ing tools in the application (such as the SA-DCT core for an MPEG-4 codec presented in Chapter 3). However in a price-sensitive mobile device where silicon real estate is at a premium, it is not feasible to simply lay down multiple accelerator IP cores ad nauseum – especially if the core is only useful for a single task and remains idle otherwise. Ideally, any accelerators that are implemented should be somewhat configurable for multiple tasks to increase their value. In this context, and given that the dot product operation where one of the vectors is constant is a frequent low-level task in many applications, it makes sense for a mobile multimedia device to include a dot product IP core with energy efficient properties configurable for many dot product tasks.

As outlined in Section 3.2.1.7 a distributed arithmetic (DA) implementation of a dot product has power efficient properties. Section 3.2.1.7 also discusses the advantages of adder-based DA (also known as NEDA) over ROM-based DA. Hence, as with the approaches surveyed in Section 5.3.2, the proposed optimisation approach is targeted towards an adder-based DA realisation of the final CMM IP core. The key differentiating attributes of the proposed approach over existing work are as follows:

1. The algorithm considers all radix-2 SD representations of the CMM fixed-point constants when building solutions. SD permutation has only thus far been applied to simpler problems like single constant multipliers [239] and FIR filters [240].
2. The algorithm uses CSE and avoids getting stuck in local optima by maintaining parallel solutions. Previously proposed approaches build a single solution by choosing sub-expression patterns iteratively using greedy heuristics. The proposed algorithm evaluates all sub-expression choice options in parallel, and decides on the best choice afterwards.
3. The improved results achievable by using SD permutation and building parallel solutions comes at the cost of additional computational complexity. Previous work has commented that the SD permutation search space is huge and whilst this is indeed true the modelling and implementation of the proposed approach aims to alleviate this issue by cleverly ordering the search space. The proposed modelling solution, summarised in Section 5.3.3.3, makes the implementation amenable to genetic programming and hardware acceleration.
4. As hinted at in Section 5.3.1.3, real-world synthesis issues should be factored into the design of any hardware resource allocation algorithm. The proposed algorithm is scalable to large CMM problems by adopting a “divide and conquer” approach. The division of a large CMM into smaller independent sub-problems makes sense from a layout and power consumption perspective [243, 245].
5. In the context of the discussion in Section 5.3.1.2, the algorithm processes each CMM dot product independently by dividing the elements  $b_{i,j,k}$  of 3D matrix  $\mathbf{A}_d$  into  $N$  2D slices along the  $i$  plane. Horizontal patterns are searched for (i.e. using the P1D strategy) in each 2D slice. The results for each dot product are then combined to yield an optimised solution for the entire CMM problem. Such a strategy facilitates effective re-use of logic and the logic itself has good area, layout, speed and power characteristics. The reasons for this are elaborated on in Section 5.3.3.1.



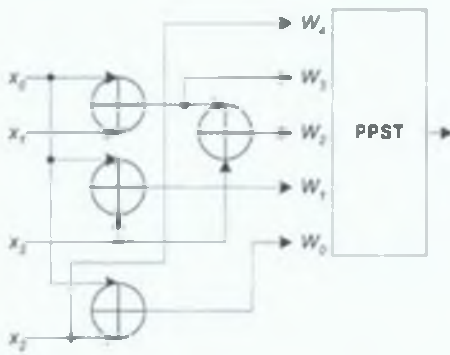


Figure 5.14: Sample PID Architecture

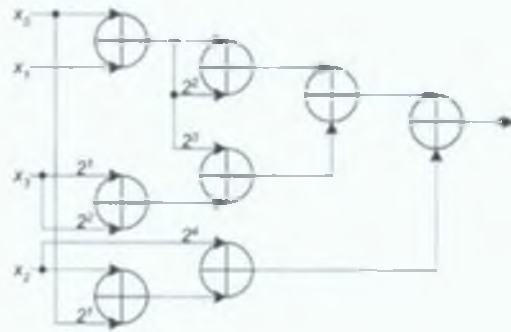


Figure 5.15: Sample P2D Architecture

### 5.3.3.1 Proposed CSE Pattern Search Strategy

Since the elements  $b_{ijk}$  of the 3D matrix  $A_{i,l}$  are divided into  $N$  2D slices along the  $i$  plane, the proposed algorithm optimises each dot product in the CMM equation independently before combining the results. Since a CMM is essentially a collection of dot products, slices along the  $i$  plane were chosen since this means the algorithm can readily be applied to single dot product problems (essentially a single 2D slice). It is important to note that dividing a CMM problem up like this does not prevent the algorithm finding certain solutions for the global CMM problem because all possible solutions for each dot product are retained to ensure no potential solutions are missed when the dot product results are combined for the global solutions. The methods by which this is achieved are explained in detail in Section 5.4.

For each 2D slice the PID search strategy as outlined in Section 5.3.1.2 is used. The reasons for choosing this strategy as opposed to the P2D strategy are best illustrated by considering a simple example. Equation 5.18 shows a simple 2D slice with  $N = 4$  and  $M = 5$ .

$$X(i) = \begin{bmatrix} 2^0 \\ 2^1 \\ 2^2 \\ 2^3 \\ 2^4 \end{bmatrix}^T \underbrace{\begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{2D slice of } b_{ijk}} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (5.18)$$

Referring back to Equation 3.9, a PID approach searches only for horizontal patterns of digits  $\{\pm 1\}$  in the 2D slice and uses these to form weights  $W$ . These weights are then accumulated according to their binary digit position ( $2^0, 2^1, \dots$ ) with some adder-tree (e.g. a highly optimised PPST as discussed in Chapter 3). For the given example, a potential architecture is shown in Figure 5.14. A P2D strategy searches for vertical, horizontal and diagonal patterns of digits  $\{\pm 1\}$  in the 2D slice and a potential solution architecture is shown in Figure 5.15. At a first glance it appears that the P2D architecture requires 8 adders while the PID architecture requires 4+PPST adders. The number of adders in the PPST can be determined by the formula  $(M - 3) + 1 + 1 = M - 1$  if carry save adders are used [160]. Using this formula the PID architecture also requires the equivalent of 8 adders. However, the number of

adder “units” is not a true benchmark since the bitwidths of these adder units must be taken into account. Analysis of the structures in Figure 5.14 and Figure 5.15 shows that the P1D architecture requires 69 1-bit full adders (FAs) whereas the P2D architecture requires 89 1-bit FAs. Analysis of the critical paths reveal that for the P1D architecture, the critical path is 36 FAs and for the P2D it is 47 FAs. It is clear that the P1D is advantageous both in terms of area and speed. The primary reason for the advantage is that the PPST has attractive speed, layout and power characteristics since the carry save tree accumulates the weights  $W$  with minimal carry propagation [160]. Indeed, such characteristics are leveraged in the SA-DCT and SA-IDCT architectures of Chapters 3 and 4.

### 5.3.3.2 Architectural Considerations

When designing a CMM optimisation algorithm, it is useful to keep in mind the implementation options that exist for hardware implementation of the CMM. Since the CMM is a collection of  $N$  dot products, each dot product could be computed using  $N$  computation units in parallel. Alternatively a single computation unit could be implemented that is time multiplexed to compute each dot product in turn (e.g. the dot product architecture proposed in Chapter 3 is capable of computing 8 different  $N$ -point DCTs). It is also possible to adopt a hybrid serial-parallel architecture between the aforementioned extremes. The size of the CMM problem (defined by  $N$  and  $M$ ) is a crucial factor in deciding on the appropriate option. The correct choice also depends on the throughput requirements of the CMM block, bandwidth limitations, any area constraints imposed and whether or not power consumption is a priority. All optimisation approaches thus far have assumed a parallel approach, but for a large  $N$  CMM this is not always possible due to area, bandwidth and power constraints. This suggests a divide and conquer approach is sensible for large problems as hinted at in Section 5.3.1.3.

The dot product circuit need not be restricted to a single CMM – it could be configured to implement a set of dot products spanning multiple CMMs and if the final IP core can be configured in this way, the value of the core itself is greatly improved. Indeed if the dot products involved are long (e.g. 1024-point FFT), it is possible that the individual dot products themselves could be serialised into discrete segments (again hinting at divide and conquer optimisation). The dot product circuit could then be reconfigured to compute each segment sequentially, accumulating the results. For example consider  $X = A_0x_0 + A_1x_1 + A_2x_2 + A_3x_3$  which could be segmented as  $X = \alpha + \beta$  where  $\alpha = A_0x_0 + A_1x_1$  and  $\beta = A_2x_2 + A_3x_3$ . In this simple case the dot product circuit could be configured to compute  $\alpha$  followed by  $\beta$  in the next clock cycle.

Hence, if a reconfigurable dot product architecture is required capable of implementing a set of dot products, it makes sense to adopt a two-layer P1D architecture, since both layers can be independently highly optimised. In this way, the reconfiguring multiplexers (such as those proposed in Chapter 3) are confined to one layer. This promotes hardware resource re-allocation/re-use and makes the architecture less application specific. This discussion further justifies choosing the P1D search strategy.

### 5.3.3.3 Proposed Efficient Modelling Solution

The proposed algorithm permutes the SD representations of the constants in  $\mathbf{A}$  of Equation 5.13. For each permutation, parallel solution options are built based on different sub-expression choices. These parallel implementations are expressed as a sum of products (SOP), where each product term in the

SOP represents a particular solution (with an associated adder count). The SD permutation is done on each CMM dot product in isolation (see Section 5.4.2), and the results are subsequently combined (see Section 5.4.3). The combined SOPs are ordered yielding the overall best (in terms of adder count) sub-expression configuration to implement the CMM equation. Essentially, previous approaches derive one implementation option (akin to a single term SOP) whereas the proposed approach derives parallel implementations (a multi-term SOP). It is this multi-term SOP approach and its manipulation (see Section 5.4) that make the algorithm suitable for GAs and hardware acceleration.

The proposed algorithm currently uses the P1D strategy, so it searches for horizontal sub-expression patterns of  $\{\pm 1\}$  digits in a 2D slice. The proposed SOP modelling idea can be extended to cover the P2D strategy by simply extending the digit set from  $\{\pm 1\}$  to  $\{\pm 1, \pm 2, \pm \frac{1}{2}, \pm 4, \pm \frac{1}{4}, \dots\}$ . However, the P1D approach was chosen since it generates results with attractive hardware characteristics.

The value of the algorithm is that it can be applied to any CMM operation that can be expressed in the form of Equation 5.13. The primary challenge facing the proposed approach is the size of the solution space. This space is huge even before considering SD permutation, which only increases the search space even further. The size of the search space is exponentially dependent on both  $M$  and  $N$ . This problem is addressed by the proposed algorithm in Section 5.4, which incorporates intuitive divide and conquer techniques as well as effective search space ordering to facilitate redundant search space area elimination.

## 5.4 The CMM Optimisation Algorithm

The proposed approach is a three stage algorithm as depicted in Figure 5.16. Firstly, all SD representations of the  $M$ -bit fixed point constants are evaluated using an  $M$ -bit radix-2 SD counter (see Section 5.4.1). Then, each dot product in the CMM is processed independently by the dot product level (DPL) algorithm (see Section 5.4.2). Finally the DPL results are merged by the CMM level (CMML) algorithm (see Section 5.4.3). The three steps may execute in a pipelined manner with dynamic feedback between stages. This offers search space reduction potential as outlined subsequently.

### 5.4.1 Signed-Digit Permutation Extraction Stage

Equation 5.2 defines the conventional radix-2 2's complement representation of an  $M$ -bit fixed-point number. However there are alternative representations possible, for example the radix-2 SD number system as introduced in Section 5.2.1. Using the radix-2 SD number system there are a finite number of different SD representations of an  $M$ -bit fixed-point constant. The number of possibilities increases with  $M$ . As a simple example, consider the possible representations of  $-3$  with  $M = 4$ . In this scenario there are five legitimate SD representations that yield the value  $-3$ .

$$\begin{aligned} (00\bar{1}\bar{1})_2 &= -2 - 1 = -3 \\ (0\bar{1}01)_2 &= -4 + 1 = -3 \\ (\bar{1}101)_2 &= -8 + 4 + 1 = -3 \\ (0\bar{1}1\bar{1})_2 &= -4 + 2 - 1 = -3 \\ (\bar{1}11\bar{1})_2 &= -8 + 4 + 2 - 1 = -3 \end{aligned}$$

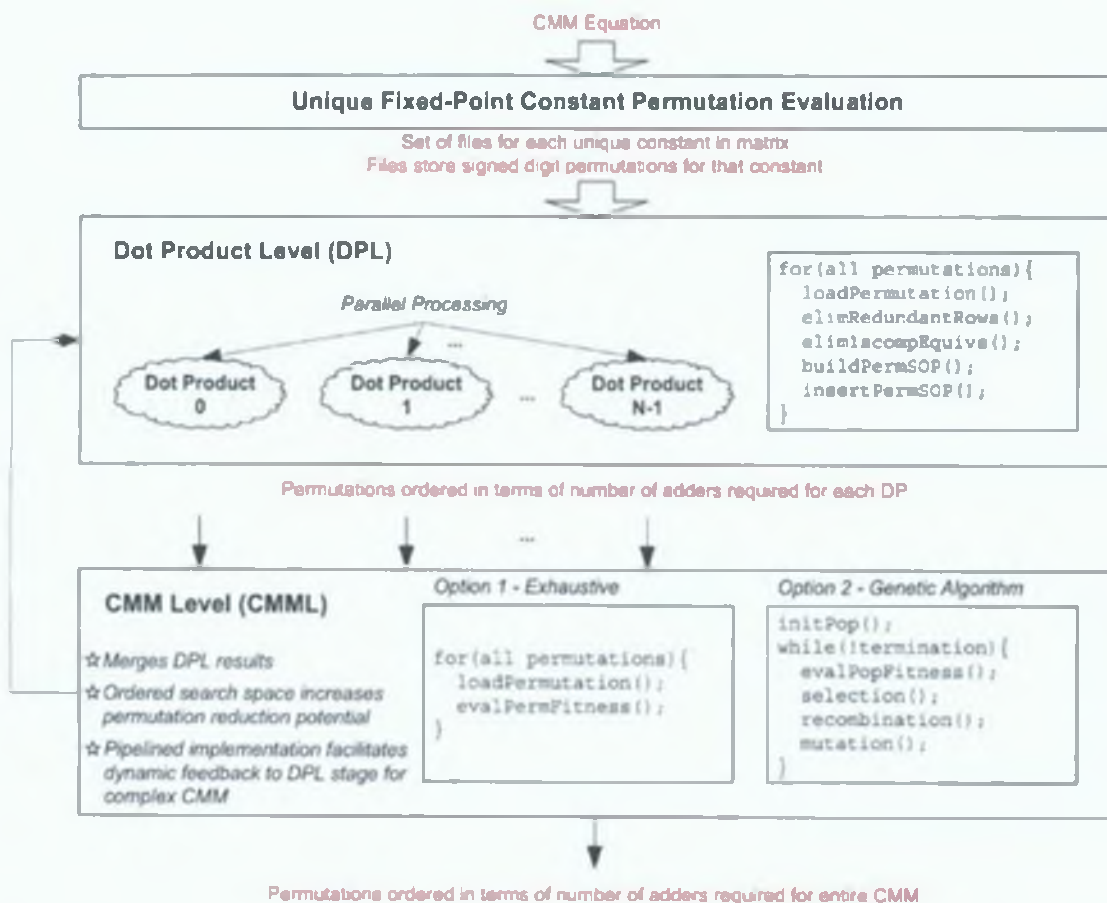


Figure 5.16: Summary of the CMM Algorithm

These representations have different zero and non-zero digit distributions and the significance of this is apparent when one considers that the constant  $-3$  could be a column in a CMM 2D slice. Clearly, the different zero and non-zero digit distributions affect the sub-expression patterns that could be chosen by an optimisation algorithm.

Intuitively, one may expect that representations with the minimal non-zero digits (CSD or any MSD) would be the best choice to obtain the minimal number of adders for a NEDA implementation. This is not true in general since adders are only inferred if two or more non-zero digits are present in the same rows of a 2D slice. If the PID sub-expression search strategy is employed, patterns in the 2D slice columns do not matter since the PPST looks after accumulation in this direction. The impact of this is that it is the permutation of SD representations for *all* the constants of the CMM matrix that determines the adder count. An MSD representation only guarantees minimal non-zero digits along each column of a 2D slice.

This step of the algorithm finds all of the unique radix-2 SD representations for all the unique basis elements in the CMM, i.e. all of the unique  $a_i$  in Equation 5.14. This is implemented using an exhaustive  $M$ -bit radix-2 counter. If an  $M$ -bit SD string is equal to one of the constants  $a_i$ , the string is saved in a file associated with that particular  $a_i$ . This is a pre-processing stage before the SD values are permuted by the DPL and CMML stages of the algorithm. For the 8-point 1D DCT there are 13 unique constants

( $\pm a_2$  are considered as two distinct constants)

In future work it is worth investigating other algorithms for generating the possible SD representations for a given constant. For example it may be useful to order the representations based on their Hamming weight (i.e. the number of non-zeros set) as suggested by Dempster and Macleod [238]. Alternative sorting will not affect the optimality of the results but they may produce the results quicker (this may be significant for larger CMM problems)

Other future research could investigate number systems other than radix-2 SD to investigate whether they improve results. This could include radix-4, radix-8 or even higher-radix systems. Indeed it is probably worth investigating the ideas in [258] and [259] to understand how SD number systems can be generalised and how “hybrid SD” arithmetic may be beneficial. A recent paper by Phillips and Burgess proposes a general sliding window scheme for recoding a number (in a variety of popular recoding schemes) with a minimal Hamming weight [260].

#### 5.4.2 Dot Product Level (DPL) Stage

The DPL algorithm iteratively builds a SOP, and the final SOP terms are the unique sub-expression selection options after considering all SD permutations of the dot product constants in question. The final SOP terms are listed in increasing order of the number of adders required by the underlying sub-expressions. The DPL algorithm executes the following steps for each SD permutation:

- 1 Load the next SD permutation of the fixed-point constants from a CMM 2D slice (a single dot product forming part of the CMM)
- 2 Eliminate redundant rows in 2D slice
- 3 Eliminate 1's complement equivalences in 2D slice
- 4 Build a SOP where each term in the SOP represents a unique way of implementing this dot product using NEDA according to the current SD permutation
- 5 Integrate the SOP terms into the overall ordered list of unique implementations obtained so far

All dot products are treated independently at the DPL stage and the results are combined in the CMML stage. Due to the large number of permutations the above steps are carried out many times. Hence the software implementation of the algorithm is vital and must be as efficient as possible. Potential early exit strategies are outlined to help in this regard.

The DPL software allows the user to interrupt the algorithm at any stage. When interrupted, the program continues until a safe point to stop and reports how much of the permutation space for the particular dot product being examined has been searched so far. It also reports the best permutation found thus far and the associated number of adders required. At this point, the user must decide whether to continue execution of the program from that point or terminate and save the configuration and algorithm state. If quit, the configuration state can be used to reconstruct the data structures and file pointers and restart from the interrupted state.

### 5.4.2.1 Step 1 – Load Permutation

For illustration, consider the dot product to compute 8-point 1D DCT coefficient  $X(1)$  in Equation 3.11b where  $N = 4$ . With  $M = 13$  there are a finite number of 13-bit SD representations for each of  $a_1, a_3, a_5$  and  $a_7$ . In fact there are 37, 193, 155 and 133 respectively giving a total number of 147211715 permutations, and one such permutation is listed as an example in 2D slice form in Equation 5.16. The key idea is to examine each of these permutations and see which one leads to the minimal number of adders necessary. The CMML stage will then permute over all dot products in the CMM. This section details the permutation processing at the DPL only.

For the current SD permutation, each of the 2D slice columns is loaded into a char pointer that has been allocated  $M$  amount of bytes using the function in Listing 5.1. Each vertex in the 2D slice matrix is stored in a byte. Although it is possible to pack four radix-2 digits into a byte, having a byte for each digit allows more efficient manipulation.

Listing 5.1 Load SD Permutation

```
void opt_load_vector(char data, ifstream fileptr)
{
    fileptr.read(data, M);
}
```

### 5.4.2.2 Step 2 – Eliminate Redundant Rows

Each row in the  $M \times N$  2D slice matrix (e.g. the  $13 \times 4$  matrix Equation 5.16) when multiplied with the data vector forms a weight  $W$  that is a linear additive combination of the 4-point data vector. Any row with the string 0000 implies no additions are necessary for this weight so may be ignored for optimisation. Any row with only one non-zero digit also implies that no additions are necessary since only 1 element of the data vector is selected to form the weight and no additions are necessary. Therefore if any row is represented by an element from the set 0001, 0010, 0100, 1000, 000 $\bar{1}$ , 00 $\bar{1}$ 0, 0 $\bar{1}$ 00,  $\bar{1}$ 000 it can also be ignored for subsequent optimisation assuming that all  $x_i$  and  $-x_i$  are available. For example  $W_{-11}$  in Equation 5.16 is formed by the row 0 $\bar{1}$ 00 which implies  $W_{-11} = -x_1$ . Clearly if two or more rows with non-zero digits are identical, they represent the same set of additions so only one of these rows needs consideration for optimisation. For  $N = 4$  there are 36 possible row patterns with a Hamming weight  $\geq 2$  that imply unique additions as outlined in Table A.3 in Appendix A.

Referring back to Equation 5.16 if we ignore any rows of the  $13 \times 4$  matrix with all zeros or only 1 non-zero digit we are left with the following rows

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & \bar{1} & 0 & 1 \\ 0 & 1 & 1 & \bar{1} \\ \bar{1} & 1 & 0 & 0 \\ 0 & 0 & \bar{1} & 1 \\ \bar{1} & 1 & 0 & 0 \\ 0 & 0 & 1 & \bar{1} \end{bmatrix}$$

Each of the rows in the above matrix are pushed onto a dynamic singly linked list – used for code efficiency and memory conservation. They are pushed on at the top for efficiency because order does

not matter at this stage (they are ordered later on) Each node corresponds to a row in the above matrix and has four data elements corresponding to the signed digit value of each column. The node structure is shown in Listing 5.2

Listing 5.2 NEDA Node

```
// NEDA row SLL node definition
struct nodeNEDA {
    unsigned char rbit0
    unsigned char rbit1
    unsigned char rbit2
    unsigned char rbit3
}
struct nodeNEDA next
}
```

Next, if any rows have the same pattern all but one are redundant. In the above matrix, the pattern  $\bar{1}100$  occurs twice so one is removed from the list leaving

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & \bar{1} & 0 & 1 \\ 0 & 1 & 1 & \bar{1} \\ \bar{1} & 1 & 0 & 0 \\ 0 & 0 & \bar{1} & 1 \\ 0 & 0 & 1 & \bar{1} \end{bmatrix}$$

#### 5.4.2.3 Step 3 – Eliminate 1’s Complement Equivalences

After eliminating redundant rows we are left with a number of unique patterns that have at least two non-zero digits per row. However, it may be the case that there are occurrences where a certain pattern remains on one row and its 1’s complement occurs on another row. Table A.3 shows that a unique pattern and its ones complement infer the same unique set of additions, albeit the final output is the  $\pm$  of the other. Therefore for subsequent optimisation, one of these rows can be eliminated since only additional inverters are required and the ‘1’ added to the LSB can be subsumed by the PPST. The DPL algorithm checks for this condition for each row and as soon as it is found, one row is eliminated and the next row is checked. By virtue of the redundant row elimination, it is guaranteed that if a 1’s complement pair is found there will be no other row in the rows that haven’t yet been checked that form the appropriate 1’s complement pair, which saves unnecessary iterations. Applied to this particular example this step reduces the permutation matrix to

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & \bar{1} & 0 & 1 \\ 0 & 1 & 1 & \bar{1} \\ \bar{1} & 1 & 0 & 0 \\ 0 & 0 & 1 & \bar{1} \end{bmatrix} \tag{5.19}$$

#### 5.4.2.4 Step 4 – Build Permutation SOP

Steps 2 and 3 reduce the example 2D slice in Equation 5.16 to the residual matrix shown in Equation 5.19. The DPL algorithm considers each residual row in turn and builds an implementation SOP for that row,

as in (5 20)

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & \bar{1} & 0 & 1 \\ 0 & 1 & 1 & \bar{1} \\ \bar{1} & 1 & 0 & 0 \\ 0 & 0 & 1 & \bar{1} \end{bmatrix} \Rightarrow \begin{bmatrix} (p_0) \\ (p_{10}) \text{ AND} \\ ((p_3)(p_{51}) \text{ OR } (p_{10})(p_{52}) \text{ OR } (p_{11})(p_{53})) \text{ AND} \\ (p_6) \text{ AND} \\ (p_{11}) \text{ AND} \end{bmatrix} \quad (5 20)$$

Each SOP term is represented internally as a data structure with elements `prod_vector` (a bit vector where each set bit represents a specific adder to be resource allocated) and `num_nonzeros` (the Hamming weight of `prod_vector` that records the total adder requirement) In Equation 5 20,  $p_v$  means bit  $v$  is set in the `prod_vector` for that SOP term The number of possible two input additions is equivalent to the combinatorial problem of leaf-labelled complete rooted binary trees [261] With  $N = 4$ , the number of possibilities is 180 as shown in Table A 3 (in Appendix A) and the general series in  $N$  increases quickly for  $N > 4$  (see proof in Appendix A) Future work will investigate an automated method for configuring the DPL algorithm for any  $N$ , which will leverage the expression for the general series presented in Appendix A To get around this issue in the meantime for larger  $N$  CMM problems, the author proposes to leverage the divide and conquer approach as suggested in Section 5 3 1 3 By dividing an  $N$ -point dot product into  $N/r$  smaller  $r$ -point chunks, a large problem can be divided into more manageable sub-problems (Section 5 3 1 3 also outlines why this also makes sense from an implementation perspective) In the example being used here,  $N = 4$  so sub-division is not necessary

So, each `prod_vector` is a 180-bit vector with a `num_nonzeros` variable equal to the number of required adders The SOPs for each row are logically ORed together to form a permutation SOP that is an exhaustive set of sub-expressions options that implement the entire permutation The permutation SOP for Equation 5 16 is given by the three-term SOP in Equation 5 21

$$\begin{aligned} & ((p_{11})(p_6)(p_{53})(p_{10})(p_0)) \\ & ((p_{11})(p_6)(p_{10})(p_{52})(p_0)) \text{ OR} \\ & ((p_{11})(p_6)(p_3)(p_{51})(p_{10})(p_0)) \text{ OR} \end{aligned} \quad (5 21)$$

The last term in Equation 5 21 has `num_nonzeros = 6` so it requires 6 unique additions (+PPST) to implement Equation 5 16 whereas the first two options only require 5 unique additions (+PPST) The three possible architectures corresponding to each SOP term are illustrated in Figure 5 17 Clearly one of the first two options is more efficient if implementing this dot product in isolation However, when targeting a CMM problem one must consider the CMM level, and it may be that permuting the first option at CMML gives in a better overall result since it may overlap better with requirements for the other dot products Hence it is necessary to store the entire SOP for each permutation at DPL and then permute these at CMML to obtain the guaranteed optimal

To appreciate how a permutation SOP like the example in Equation 5 21 is generated, some implementation details are subsequently outlined Ordinarily these may not be of interest but since the DPL algorithm generally may be executed for a large number of permutations, efficient implementation is crucial to the feasibility of the algorithm The SOP for each permutation is built using a singly linked list strategy to conserve the memory allocated by the operating system running this optimisation algorithm



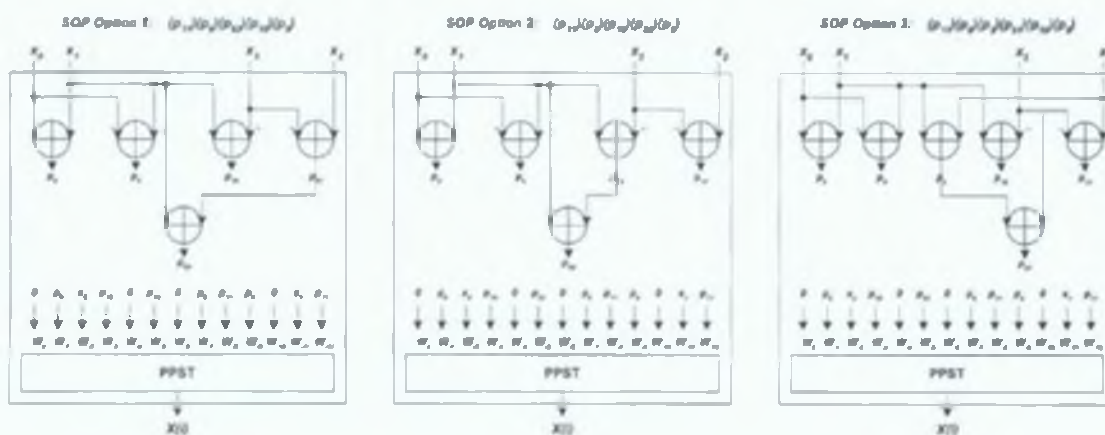


Figure 5.17: Architecture Options Implied by Example SOP

Each SOP term is allocated a node in the list and the node definition is illustrated in Listing 5.3. A key implementation decision is the representation of `prod_vector` as discussed in detail in Section 5.4.2.6. The current implementation of the 180-bit `prod_vector` as shown in Listing 5.3 is a collection of six `int` variables representing 32-bit chunks of `prod_vector`. Since the `int` data type is typically 32 bits wide on a conventional desktop PC,  $6 \times \text{int} = 192$  bits so the top 12 bits of `pv_6` are unused.

Listing 5.3: SOP Term Node Data Structure

```
// CMM SOP node definition
struct node4 {
    // Hamming Weight of prod_vector
    unsigned char num_nonzeros;
    // 180-bit prod_vector implemented as an aggregate of int data chunks
    // prod_vector $[equiv] [pv_6,pv_5,pv_4,pv_3,pv_2,pv_1]
    unsigned int pv_1;
    unsigned int pv_2;
    unsigned int pv_3;
    unsigned int pv_4;
    unsigned int pv_5;
    unsigned int pv_6;
    // Pointer to next list node
    struct node4* next;
};
```

The implementation challenge in this step is to build the linked list for the permutation SOP from the matrix in Equation 5.20. The approach taken is to use three separate linked lists whose list head pointers switch dynamically and the SOP is formed in an accumulated manner by taking each row of the residual matrix (such as Equation 5.20) in turn. The implementation can be conceived as a finite state machine as described in Figure 5.18. When the SOP for the very first row is encountered the nodes for each product ( $P$ ) in the SOP are pushed onto the singly linked list `s11_1` (the order doesn't matter in this case since `num_nonzeros` is equal for each  $P$  in the SOP for this row). The state machine moves to state `STORE_3` where the second row SOP is pushed onto singly linked list `s11_2` (again the order doesn't matter in this case since `num_nonzeros` is equal for each  $P$  in the SOP for this row). Then `s11_1` and `s11_2` are combined and the results are stored in `s11_3`. This combination process involves combining each node from `s11_1` with each of the nodes in `s11_2` forming new nodes that are stored in `s11_3`. So if `s11_1` has  $L_1$  nodes and `s11_2` has  $L_2$  nodes, `s11_3` will have  $L_1 \times L_2$  nodes. After combining all nodes from the in-list (`s11_2`) and the current-list (`s11_1`) into the out-list (`s11_3`), the out-list represents the partially accumulated SOP. The state machine iterates until the partial SOP for each row has been incorporated into the overall permutation SOP. Each term in this final SOP represents

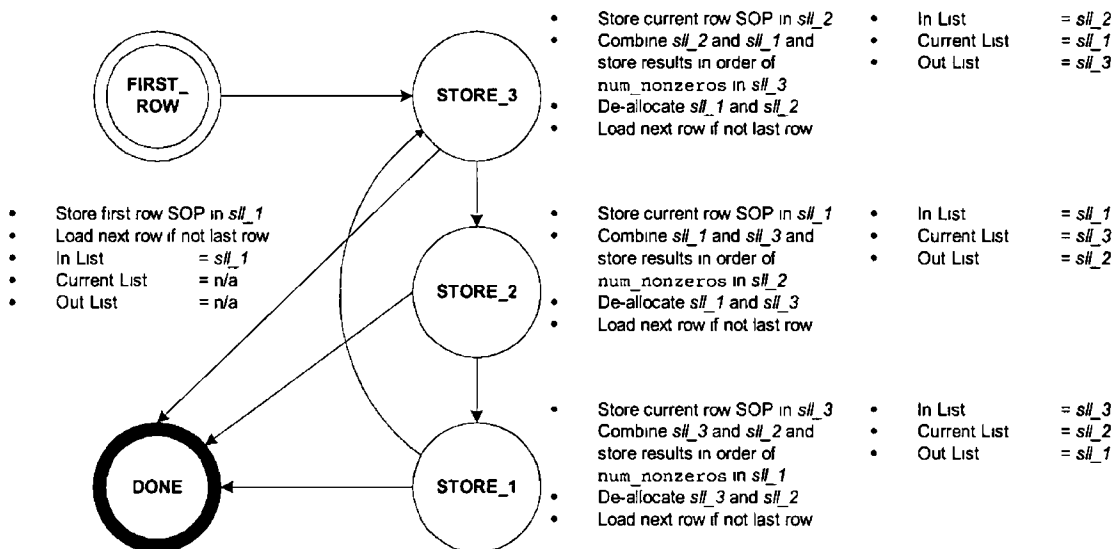


Figure 5 18 Permutation SOP Building Finite State Machine

an implementation option for the associated signed digit representation permutation of the associated dot product constants

The combination of two nodes is illustrated in Figure 5 19 When combining two particular nodes the  $prod\_vector_{in}$  and  $prod\_vector_{cur}$  are combined in a bit wise OR fashion to generate a new  $prod\_vector_{out}$  for the output node This new vector  $prod\_vector_{out}$  represents a total number of unique adders needed for all rows combined thus far (possibly one option of many – the number of options depends on the number of nodes in the out-list) The node value for  $num\_nonzeros$  for the new node ( $num\_nonzeros_{out}$ ) in  $sll\_3$  is obtained by counting the number of ones in  $prod\_vector_{out}$  However, this is not necessary in the case where  $prod\_vector_{in}$  bit wise AND  $prod\_vector_{out}$  equals zero In this specific case  $num\_nonzeros_{out} = num\_nonzeros_{in} + num\_nonzeros_{cur}$  This check saves the computational burden of iterating over  $prod\_vector_{out}$  After each set of combinations, each P in the output SOP represents an alternative way of implementing all the rows combined thus far in terms of unique adders For the cases where the bits must be counted, some techniques that have been implemented to speed up the process include

- If the count exceeds a user-defined maximum, terminate the count and don't add the term to the SOP For example if one realisation requires more adders than using regular 2's complement adder-based DA there is no point in considering it any further This maximum value can also be dynamically updated according to dynamic CMML stage results if DPL and CMML are executing in a pipelined manner Such strategies help to avoid searching useless expanses of the search space – hopefully making the task more tractable An intuitive initial value for the user-defined maximum would be one less than the adder requirement to implement the 2D slice if the constants are coded using regular 2's complement implementation Hence the algorithm will only consider solutions that are better than can be obtained directly without running the optimisation algorithm
- The  $prod\_vector$  data element can be represented internally in different ways as discussed in

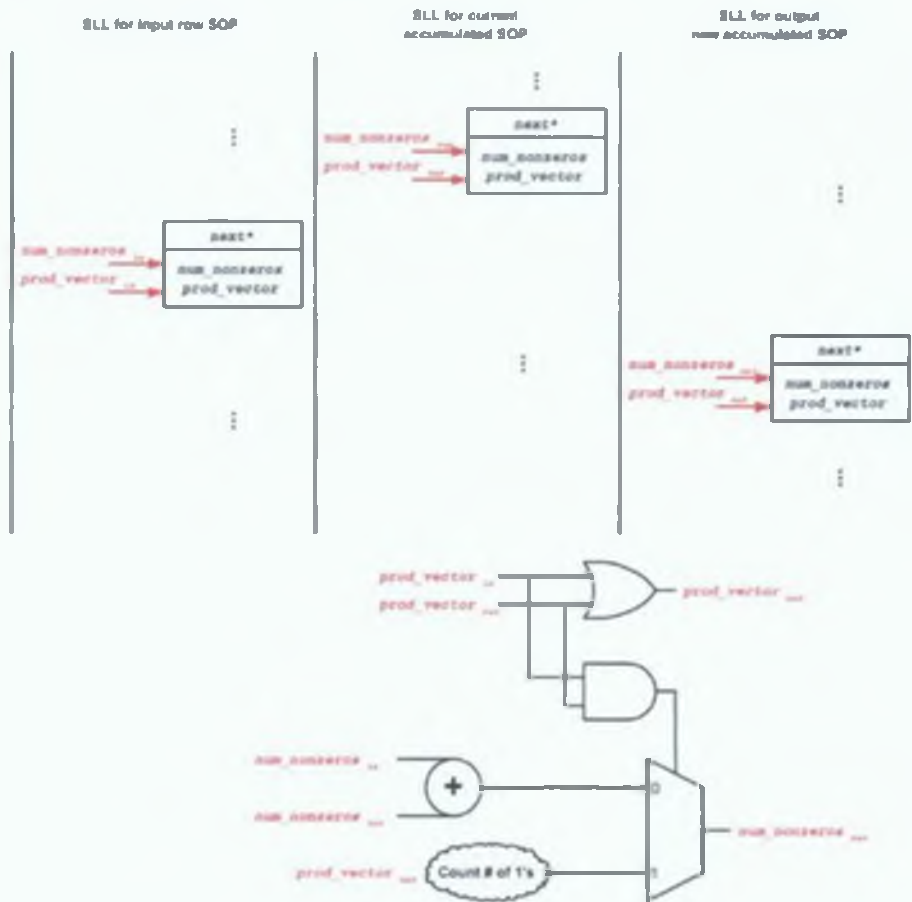


Figure 5.19: Node Combination at DPL

Section 5.4.2.6. If it is broken up into a sequence of `char` or `int` chunks the bits for each can be counted by using a look-up table.

- Also it is possible that when counting the set bits in an `int` or `char` chunk in `prod_vectorout` the associated chunk in either `prod_vectorin` or `prod_vectorcur` is all zeros. Therefore `num_nonzerosout` can be simply updated by the number of bits set in the chunk that isn't zero.

Some more techniques that have not been yet implemented, but are targeted for future investigation include:

- If counting the number of bits set in `prod_vectorout` and at some point during the process `num_nonzerosout = num_nonzerosin + num_nonzeroscur` there is no point in continuing any further since it is guaranteed that no more set bits will be found. However, careful investigation must be carried out to ensure that the extra overhead of the comparison operations does not negate the potential gains achievable with the early bit count termination.
- Essentially the goal is fast bit counting – it is therefore worthwhile investigating the feasibility of implementing a highly parallel bit counting hardware accelerator.

It is clear from Figure 5 18 that by following this state machine for all rows in the residual matrix (such as in Equation 5 20), the final SOP linked list will be a list of implementation options where each node is a term in the permutation SOP (e.g. Equation 5 21)

#### 5 4 2 5 Step 5 – Update Global DPL SOP

The algorithm checks each term in the current SOP produced by step 4 to see if it has already been found with a previous permutation. If so, it is discarded – only unique implementations are added to the global list. This global list is implemented using a 2D skip list to minimise the overhead of searching it with a new term from the current permutation SOP (Figure 5 20). In the horizontal direction there are “skip nodes” ordered from left to right in order of increasing `num_nonzeros` in the skip node list (SNL). In the vertical direction there are “product nodes” and each skip node points to a product node list (PNL) of ordered product nodes where each product node in the PNL has the same number of bits set (i.e. `num_nonzeros`) in its `prod_vector` bit vector. Therefore, if a new node is presented with `num_nonzeros = num_nonzerosnew` and `prod_vector = prod_vectornew`, it makes sense to only search the subset of nodes in the global list that have the same value `num_nonzerosnew` (i.e. only search one particular PNL). The PNLs are ordered in increasing order of their `prod_vector`, if `prod_vector` is considered as a 180-bit integer. Therefore, when iterating over a PNL at a given skip node, and the current iteration product node has a `prod_vector` variable value of `prod_vectorcur` such that `prod_vectorcur > prod_vectornew`, it is guaranteed that `prod_vectornew` is not already in the list and can be inserted at this point. When inserting into the list a unique permutation ID (`pid`) is added to the node along with `prod_vector` so that the SD permutation that generated it can be reconstructed. If the condition `prod_vectorcur = prod_vectornew` is true, then the SOP term already exists in the PNL so the new node is discarded and the search terminates immediately for this SOP term.

Essentially, step 5 of the DPL algorithm may be thought of more abstractly as follows. Each iteration of steps 1–4 of the algorithm produce a set of new numbers and in step 5 any new unique numbers presented are recorded in a list. These unique numbers correspond to unique values of `prod_vector`. The implementation challenge for step 5 is how to sort and search the list such that when a new number is presented the following needs to be determined as quickly as possible:

- Is this number already in the list? If so – discard (The **search** and **matching** tasks)
- If not, where to insert into list to aid the speed of the search task? (The **sort** task)

For the sort task the author proposes to use the 2D linked list structure so that it will reduce the number of nodes that need to be inspected when a new node is presented. The search algorithm then jumps to the appropriate `num_nonzeros` PNL and iterates down through the list until it is determined if the presented node is new or not. A consequential bonus of such an arrangement is that the final list is ordered in an attractive way for efficient CMML optimisation as discussed in Section 5 4 3. Since the CMML search space is huge, clever sorting is important. The implementation of the node comparison (the matching task) is crucial since as discussed in Section 5 4 2 6, it is the computational bottleneck of the DPL algorithm. The comparison performance is influenced heavily by the internal representation of the `prod_vector` structure data element. Since `prod_vector` is a 180-bit vector is outside the scope

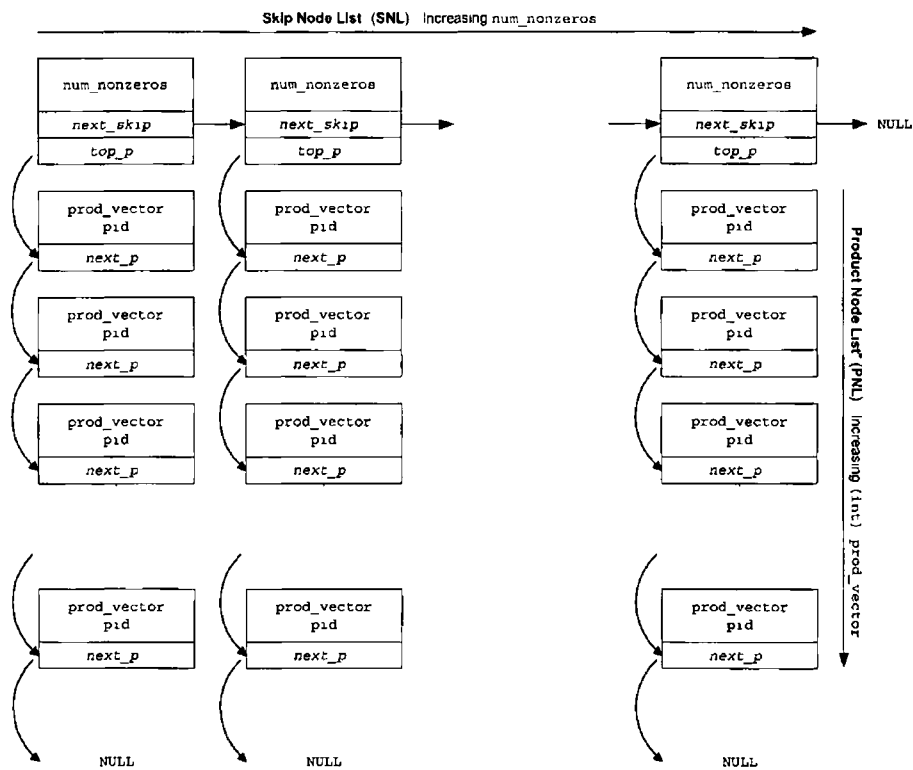


Figure 5 20 DPL Unique Implementation Skip List

of regular C/C++ data types so some creativity is required. Potential solutions to this issue are discussed in detail in the next section.

#### 5 4 2 6 DPL Algorithm Analysis and Profiling

The DPL algorithm was profiled for 11,936,085 permutations of the dot product to compute the 8-point 1D DCT coefficient  $X(1)$  in Equation 3 11b where  $N = 4$ . This number of permutations represents approximately 8 65% of the entire permutation space for this dot product. Results show that approximately 60% of the computation time is consumed by bit counting and bitwise OR operations (broken up into 50% count and 10% OR). Also, approximately 30% of the time is consumed by dynamic memory allocation and deallocation (C++ new and delete operators).

The runtime for the DPL algorithm for each of the 8 dot products in the 8-point 1D DCT CMM with  $M = 13$  is approximately 24 hours (using a Pentium-4M 2GHz running Windows 2000). This runtime increases as  $M$  increases so efficient implementation is key to the viability of the algorithm. Some general implementation techniques that were used to minimise runtime include

- Keep the number of function arguments low and the number of live variables low to minimise probability of “spilling” which means data is kept in processor registers as opposed to out in memory. By keeping live algorithm data close to the processor core, frequent slow memory accesses are avoided.
- The algorithm has inherent early-exit strategies that avoid unnecessary computation without compromising the optimality of the results.

- Many other implementation tweaks are of course possible and may be introduced in the future

Since node bit counting and node combination (using bitwise OR) are the bottlenecks of the algorithm, the chosen data representation of the nodes is important and some potential candidates for representing the nodes are presented in Listing 5 4 (`prod_vector` is 180-bits wide) and discussed in the following

Listing 5 4 SOP Term Node Data Structure Options

```

// Option 1
struct pnode {
  sc_bv<180>   prod_vector
  double      pid
  struct pnode next_p
}

// Option 2
struct pnode {
  unsigned int pv_1
  unsigned int pv_2
  unsigned int pv_3
  unsigned int pv_4
  unsigned int pv_5
  unsigned int pv_6
  double      pid
  struct pnode next_p
}

// Option 3
struct pnode {
  unsigned char pv_1
  unsigned char pv_2
  unsigned char pv_3
  unsigned char pv_4
  unsigned char pv_5
  unsigned char pv_6
  unsigned char pv_7
  unsigned char pv_8
  unsigned char pv_9
  unsigned char pv_10
  unsigned char pv_11
  unsigned char pv_12
  unsigned char pv_13
  unsigned char pv_14
  unsigned char pv_15
  unsigned char pv_16
  unsigned char pv_17
  unsigned char pv_18
  unsigned char pv_19
  unsigned char pv_20
  unsigned char pv_21
  unsigned char pv_22
  unsigned char pv_23
  double      pid
  struct pnode next_p
}

```

**Option 1** The `prod_vector` element is represented by a single data variable using the SystemC bit-vector data type `sc_bv<180>` Using the SystemC language is interesting since it defines hardware-like data types such as user defined precision bit vectors, making coding more straightforward and easy to maintain Also, SystemC has recently been ratified by the IEEE as a standard so there is growing support for the language and plenty of resources to refer to [262]

Advantages

- It is a single variable so is easy to manipulate in code

Disadvantages

- When comparing two values of `prod_vector` all 180 bits must be compared It is possible to iterate over the individual bits from MSB down towards the LSB to determine whether one is bigger than the other (according to the highest order set bit in both) but this is very slow since SystemC data types are themselves structures and comparing elements of structures involve a lot of accesses
- The data types `sc_bv<X>` actually need 16 bytes in memory regardless of the size of X so the single variable advantage is not as clear-cut as it appears
- If  $X > 64$  a cast is required to the type `sc_biguint<X>` when comparing `sc_bv<X>` strings This is undesirable in terms of computational processing In fact the type `sc_biguint<X>` also requires 16 bytes of memory so in fact `sc_biguint<X>` possibly represents an improved alternative to `sc_bv<X>` since the casting will be eliminated

- Profiling variations of the bit counting function has shown that counting bits in SystemC data types is very inefficient compared to the native C/C++ data types. Hence SystemC types do not seem a viable solution and native data types were investigated instead.

**Option 2** The `prod_vector` element is distributed over a group of `int` chunks where `pv_1` contains the LSBs up to `pv_6` which contains the MSBs. Since a collection of `int` variables can only hold multiples of 32 bits there is obviously redundancy for 180 bits (closest is 192 bits).

Advantages:

- Native C/C++ types are faster for bit counting compared to SystemC types.
- Dividing `prod_vector` into explicit chunks helps to avoid some redundant bit counting. By only counting the bits set in an `int` chunk whose value  $\neq 0$  some time is saved. Referring back to Figure 5.19 when combining two nodes the two sets of input chunks are bit wise ORed together individually. If corresponding chunks are both zero valued, no bitwise OR is necessary and the combined chunk is also zero. If one chunk is zero valued, then the combined chunk can take on the value of the other input chunk directly. Comparing an `int` variable to 0 is much faster than the corresponding operation for SystemC types.
- Dividing `prod_vector` into explicit chunks also enables early-exit strategies for step 5 of the DPL algorithm (the searching and sorting of the skip list). Such a strategy is illustrated by the pseudo code in Listing 5.5. When comparing two `prod_vector` values, the goal is to find out as soon as possible if they are equal so that the presented node can be discarded. If the nodes are not equal the next goal is to establish whether the presented node is “less than” the current node. If this condition is true, the presented node is inserted in the PNL at this location. The faster these comparison results can be determined the better because if it is not equal and not less than the current node, it obviously must be greater than the current node and we must continue searching the PNL.

Listing 5.5: SOP Term Search and Sort

```
// pv_in[0..5] -- Array of int variables for presented node chunks
// pv_current[0..5] -- Array of int variables for chunks of current PNL node

int i = 1;
// Check to see if presented node is equal to the current node in the PNL.
for( i = 0; i < 6; i++)
    // Chance quit equality check early on every byte
    if(pv_in[i] != pv_current[i])
        break;
}
if(i == 6){
    // The for loop was not broken so pv_in[0..5] == pv_current[0..5]
    deletePresentedNode();
    return; // Exit early and retrieve next presented node in SOP
}
// If execution reached this point nodes are not equal
for( i = 0; i < 6; i++){
    if(pv_in[i] < pv_current[i]){
        insertHere(i); // pv_in[i] < pv_current[i]
        return; // Insert here and exit to check next node in SOP
    }
    else if(pv_in[i] == pv_current[i]){
        continue; // Still not sure is < or >
    }
    else // pv_in[i] > pv_current[i]
        break; // Move to next node on PNL and keep checking
}
```

Disadvantages:

- The `prod_vector` element is not a single variable so it makes coding more verbose
- The `int` type offers less granularity compared to the `char` type
- Implementing the bit count of an `int` type using a single look-up table is too big ( $2^{32}$  possibilities)

**Option 3** The `prod_vector` element is distributed over a group of `char` chunks where `pv_1` contains the LSBs up to `pv_23` which contains the MSBs. Since a collection of `char` variables can only hold multiple of 8 bits there is obviously redundancy for 180 bits (closest is 184 bits)

Advantages

- Similar to option 2 but with finer granularity
- Implementation as a software look-up table is feasible (only  $2^8$  possibilities)

Disadvantages

- Similar disadvantages as option 2

**Further Options** Clearly there are many other data types and partitioning options that could be used when implementing the `prod_vector` data element that would be suitable for fast bit counting. However, another interesting approach would be to investigate potential hardware acceleration solutions for bit counting and comparisons of data. A dedicated hardware accelerator solution offers greater flexibility and can be tailored specifically for the problem in hand. It is possible to count the number of set bits in a string of data in a single clock cycle by using a highly parallel hardware loop-up table. The comparator could also be implemented in hardware. In a single clock cycle it is possible to determine whether two data variables are equal, and if not, which is bigger. The current implementation of the DPL algorithm uses option 2 (`int` chunks), but hardware acceleration solutions will be investigated in the future. The bit counting is currently implemented by masking the LSB and accumulation of bit count by 1 if the LSB is set. The `int` variable is then shifted right one position and the mask accumulate repeats. This happens for all 32 bit positions.

#### 5.4.2.7 DPL Results

To verify the implementation of the DPL algorithm and to validate the claim that improved adder resource savings are possible by permuting the radix-2 SD representations of CMM dot product constants, the DPL was run to optimise two simple 4-point dot products. The selected test cases were the Daubechies D4 wavelet and scaling functions (useful in wavelet-based image compression and analysis) [45]. The Daubechies D4 scaling and wavelet functions are shown in Equations 5.22 and 5.23

$$X_{scale} = h_0x_0 + h_1x_1 + h_2x_2 + h_3x_3 \quad (5.22)$$

$$X_{wavelet} = h_3x_0 - h_2x_1 + h_1x_2 - h_0x_3 \quad (5.23)$$

where

$$h_0 = \frac{1 + \sqrt{3}}{4\sqrt{2}} \quad h_1 = \frac{3 + \sqrt{3}}{4\sqrt{2}} \quad h_2 = \frac{3 - \sqrt{3}}{4\sqrt{2}} \quad h_3 = \frac{1 - \sqrt{3}}{4\sqrt{2}}$$



Table 5.1 16-bit Daubechies D4 Full Adder Requirements

CMM	2's Complement		CSD		DPL		
	+	FA	+	FA	+	FA	FA% Saving
D4 Scaling	22	205	20	186	18	169	9.1
D4 Wavelet	20	188	22	202	18	169	10.1

The DPL algorithm was run exhaustively with 16-bit constants ( $M = 16$ ) for both dot products. Table 5.1 compares the number of physical adder “units” required and the associated 1-bit full adder (FA) requirements. The results generated by the DPL algorithm are compared to the adder requirements if simple 2’s complement or CSD is used. It is clear that the DPL algorithm improves on both 2’s complement and CSD for both the D4 scaling and wavelet functions. The percentage FA savings achieved by the DPL algorithm compared to the second best implementation are also listed (CSD for scaling and 2’s complement for wavelet). Interestingly, the DPL algorithm found a total 59 different SD permutations that can implement the D4 scaling function with 18 adders, and 63 SD permutations that can implement the D4 wavelet function with 18 adders. None of these SD permutations are the 2’s complement or CSD permutations. The improvements illustrated by Table 5.1 validate the claim that permuting the SD representations of the CMM constants can improve the results found by a CSE optimisation algorithm. Benchmarking against state of the art CMM optimisation algorithms is deferred until Section 5.4.6 after the CMML stage of the algorithm has been outlined.

### 5.4.3 CMM Level (CMML) Stage

Once the DPL algorithm has run for each of the dot products in the CMM, there will be  $N$  2D skip lists (such as that illustrated in Figure 5.20) – one for each of the  $N$  dot products examined. The task now is to find the best overlapping product nodes for all CMM dot products. Overlapping nodes have similar `prod_vector` set bits, and this results in adder resource sharing when implementing the CMM. It is expected (though not guaranteed) that since the skip lists are ordered with the lowest `num_nonzeros` PNL first, the optimal result will be converged upon quickly saving needless searching of large areas of the permutation space. The CMM Level (CMML) algorithm searches for the optimal overlapping nodes from each of the DPL lists.

Figure 5.21 illustrates how a 2D skip list produced by the DPL algorithm is rearranged prior to processing by the CMML algorithm. Each PNL is stacked into a 1D linked list in order of associated `num_nonzeros` value (lowest `num_nonzeros` at the top). This stacking is done for each of the DPL components of the CMM. Further discussion of the CMML algorithm in this section assumes that the skip lists are stacked, even if this is not mentioned explicitly. The CMML algorithm permutes the terms in each skip list with terms from others, starting from the top of each. The CMML algorithm executes the following steps for all permutations of the skip list pointers `slp[i]`,  $i = 0, 1, \dots, N - 1$  (e.g. Figure 5.22)

1. Load the next permutation by reading the appropriate node from each of the DPL component skip lists (the very first permutation will be the top nodes in each of the stacked skip lists)
2. Combine these nodes (using bit wise OR and bit counting similar to the techniques used in Step 4 of the DPL algorithm and as shown in Figure 5.22 for  $N = 4$ ). The value of `num_nonzeros`

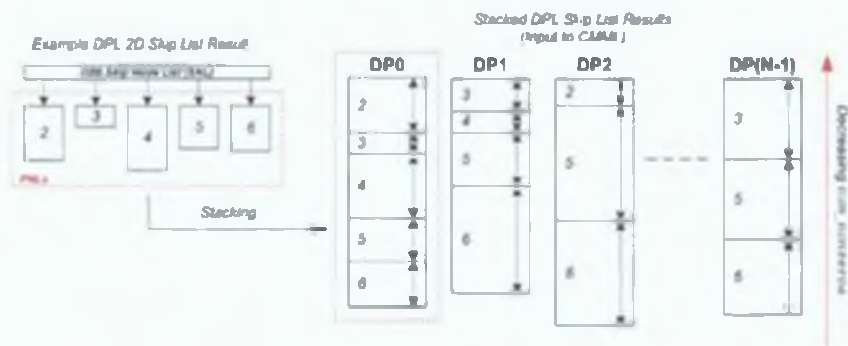


Figure 5.21: DPL Result Stacking Prior to CMML Processing

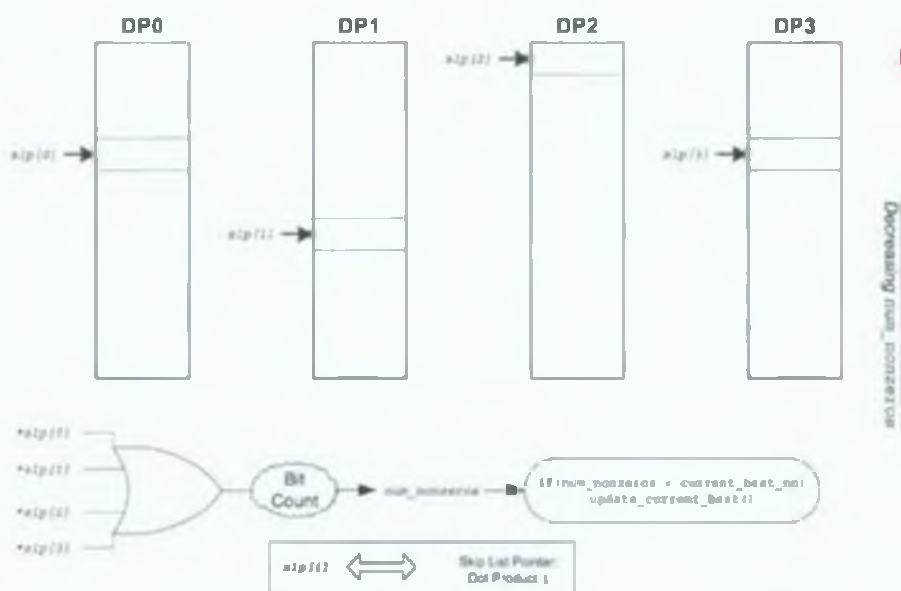


Figure 5.22: CMML Processing

of the combined node represents the number of adders necessary to implement the CMM for the current permutation.

3. Check if the combined node produced by the previous step is better (fewer adder resources required) than the best solution found from previous permutations. If an improvement is found, record this permutation as the current best solution found thus far.

If the CMML algorithm examines all permutations, the final best solution recorded is guaranteed to be the permutation that requires fewest adders to implement the CMM in question.

The potential exists to use the lowest `num_nonzeros` value found thus far to rule out areas of the search space – hence the early exit mechanism referred to previously. For example, if an improved value of `num_nonzeros = 5` is found for a CMML solution, there is no point in searching DPL PNLs with `num_nonzeros > 5` since they are guaranteed not to overlap with other DPL PNLs and give a better result than 5. This concept is illustrated in Figure 5.23, and the technique gives rise to significant search

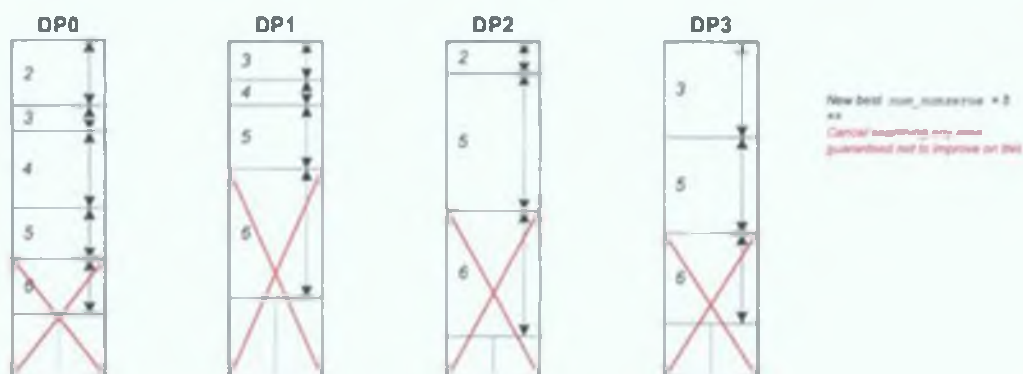


Figure 5.23: Example CMML Search Space Reduction

space reduction as discussed in Section 5.4.6. The benefit of ordering the search space is that it allows the algorithm to recognise regions of the search space that do not require examination (without sacrificing optimality) as the execution evolves and the CMML result becomes iteratively better.

The current best value of `num_nonzeros` at CMML level could also be fed back to the DPL algorithm to reduce the size of the skip lists generated by DPL (and hence permutation space) without compromising optimality. This requires that the DPL and CMML algorithms are executing in a pipelined manner. This idea has not yet been implemented and is discussed in Chapter 6.

#### 5.4.4 CMML – A Genetic Programming Approach

Despite the DPL skip list ordering, the huge permutation space means that the exhaustive CMML approach is not tractable, especially as  $N$  increases. However, the proposed modelling of the CMM problem and bit vector representation of candidate solutions means that the CMML algorithm is very amenable to genetic programming. The bit vectors can be interpreted as chromosomes and the value of `num_nonzeros` can be used to build an empirical fitness function (i.e. the less adders required, the fitter the candidate). The current fitness function is based upon the number of adders but could be extended to include parameters such as layout, and speed. The use of genetic programming should enhance the search strategy for a solution to a CMM design problem. If the search space proves too large to examine exhaustively, a genetic algorithm (GA) could be used to search the search space more cleverly. This is because a GA tends to gravitate toward finding solutions that have greater “fitness” and should converge on the optimal result quicker than a linear search of the search space.

##### 5.4.4.1 Related Work

There has not been much work in the field that harnesses genetic programming to tackle the optimisation of multiplication by constant applications. The most relevant work is by Safiri et. al., who use genetic programming techniques targeting the digital filter problem (FIR and IIR) [263, 264]. Firstly, each of the filter taps are expressed as a rooted binary summation tree assuming each constant is represented with CSD. A genetic algorithm is used that selects sub-expressions from the binary trees according to a fitness function. As well as adder count, their fitness function includes weights for fixed-point accuracy, latch count and layout efficiency. Genetic operations such as crossover and mutation are applied with

certain probabilities to the binary trees and the algorithm moves to the next generation for a new round of selection. The modelling by Safiri et al. does not explore the potential of SD permutation and is less amenable to hardware acceleration. Unlike the algorithm proposed in this thesis, it does not arrange the search space in any particular order to save processing time. Safiri et al. only consider the digital filter problem and not the more complicated CMM problem. However, the fitness function used by Safiri et al. is interesting because it includes other factors as well as adder count so may be a useful guide in extending the fitness function used in this thesis in the future.

#### 5.4.4.2 Proposed Genetic Algorithm for CMML

A proposed GA to implement the CMML algorithm is summarised in Algorithm 1. It contains the five components necessary to be classified as a GA [265], namely

- 1 A genetic representation for potential solutions to the problem (The bit string `prod_vector` produced by a bit wise OR of candidate DPL components)
- 2 A way to create an initial population of potential solutions
- 3 An evaluation function that plays the role of the environment to rate solutions in terms of their “fitness” (the bit counting operation – a lower value of `num_nonzeros` represents a fitter candidate)
- 4 Genetic operators that alter the composition of children
- 5 Values for various parameters that the genetic algorithm uses (population size, probabilities of applying genetic operators etc.)

---

#### Algorithm 1 GA-based CMML Algorithm

---

```

init_pop(),
while 'termination condition do
    eval_pop_fitness(),
    selection(),
    recombination(),
    mutation(),
end

```

---

A candidate solution  $c$  is represented by a set of  $N$  skip list pointers `slp[1][c]`, where each pointer addresses a product node in dot product skip list 1 ( $i = 0, 1, \dots, N - 1$ ). By combining these pointers the number of adders required to implement the CMM is determined. The task of the GA is to find the DPL components that overlap as much as possible resulting in the fewest adders necessary to implement the CMM with a PID architecture (see Figure 5.10). This goal is obviously the same as the goal of the exhaustive CMML algorithm, but by its very nature, the GA should search the huge solution space more effectively. A GA is an iterative process where each iteration is referred to as a “generation”. Each generation contains a number of candidate solutions in parallel – referred to as a “population” – and the idea is that candidates with greater fitness have a better chance to breed and produce offspring that contain the best genetic material from their parents (converging towards the optimum solution based on survival of the fittest). The exhaustive algorithm essentially maintains one candidate solution at a time.

(e.g. Figure 5.22) and searches the pointer permutations linearly without regard to fitness. On the other hand, the GA maintains multiple candidates in parallel and the search is steered by the relative fitness of regions of the permutation space.

#### 5.4.4.3 Step 0 – Initialise Population

The setup stage of the GA involves initialising the population for generation 0. The size of the population is determined by the parameter *pop\_size*. This parameter is usually determined heuristically and tends to range from 30 – 100 [266]. Indeed *pop\_size* could dynamically adapt across generations [265], although this has not been implemented. Since the DPL algorithm results are ordered as described in Section 5.4.2, it makes intuitive sense to initialise the population with candidates (sets of pointers) near the top of the DPL lists. This is achieved by weighting the selection of the initial candidates. For notational convenience, let  $z_{i,c} \equiv \text{slp}[i][c]$ . As mentioned in the previous section, each pointer  $z_{i,c}$  points at an address in the skip list for dot product  $i$  for a particular candidate solution  $c$ , where  $0 \leq i \leq N - 1$  and  $0 \leq c \leq \text{pop\_size} - 1$ . Each pointer  $z_{i,c}$  is in the range  $0 \leq z_{i,c} \leq \text{NP}_i$ , where  $\text{NP}_i$  is the number of product nodes in the skip list of dot product  $i$ . The algorithm randomly sets the pointer address  $z_{i,c}$  for all  $N$  pointers for each of the initial *pop\_size* candidates according to an exponential probability mass function (Equation 5.24).

$$p(z_{i,c}) = \frac{1}{\mu} \exp(-z_{i,c}/\mu) \quad (5.24)$$

According to Equation 5.24, the lower the value of parameter  $\mu$ , the more likely a candidate is to have DPL component pointers nearer the top of the respective DPL skip lists (i.e.  $z_{i,c}$  tends to zero for each of the  $N$  pointers).

The initialisation is further constrained to ensure that one of the initial candidates, candidate  $c = 0$ , points to the very top of each of the DPL skip lists (i.e.  $z_{i,0} \equiv \text{slp}[i][0] = 0, \forall i = 0, 1, \dots, N - 1$ ). Although not implemented, another approach would be to ensure that each of the initial candidates are unique and that no duplicates are present to give the algorithm the best chance of maintaining candidate diversity and not converge too quickly on a “superfit” candidate. If the algorithm is further constrained to only record unique solutions as it iterates, and keeps a record of solutions found to prevent the search from retracing a previous path, this is called a tabu search. Tabu searching has not been implemented and it is argued that these extra constraints are not really necessary, since the user should rely on the GA itself to inherently converge on the best areas of the solution space.

#### 5.4.4.4 Step 1 – Population Fitness Evaluation

The fitness of a candidate solution is obtained by doing a bitwise OR of all of the component dereferenced pointers followed by bit counting as shown in Figure 5.22. The lower the resultant bit count the better as this means less adder resources are required to implement the CMM problem with a PID hardware architecture (Figure 5.10). A possible issue with the proposed algorithm is that the variance of the CMML candidate fitness values is quite small and many candidates will have similar fitness. The logical question is – how to distinguish between equally fit candidates? The present implementation chooses one candidate randomly, but it is possible that two candidates require the same number of adders but require different additions (i.e. the addends are different). In future, if more optimisation criterion are built in to the fitness function, the increased granularity of the fitness function should give a more diverse range and

greater variance. For example, the fitness function  $f$  could include factors like fanout and logic depth as well as adder count, e.g. Equation 5.25

$$f = \alpha(\text{Adder Count}) + \beta(\text{Fanout}) + \gamma(\text{Logic Depth}) + \dots \quad (5.25)$$

Another possible variation could be to scale the fitness function using power law scaling or some other method [265]. This scaling means the fitness function has an exponential curve, e.g. a new solution with 4 adders compared to a solution with 5 adders is a much more impressive improvement than a new solution of 49 adders from 50 adders.

#### 5.4.4.5 Step 2 – Selection

The selection process decides which candidates in the current generation are worth selecting for reproduction and/or copying directly to the next generation. GA selection should maintain an appropriate balance between selective pressure and population diversity. Selective pressure refers to how strongly weighted fitter candidates are likely to be selected ahead of weaker ones. Of course this weighting is desired but too much selective pressure leads to premature GA convergence (possibly at a local optimum) so a healthy population diversity is needed. However, weak selective pressure can make the search ineffective and the GA can become a purely random search of the permutation space. Tuning the parameters of the GA effectively plays with the balance of selective pressure and diversity.

A good selection method will lead to effective searching and convergence. Again, the best approach is problem dependent and determined heuristically. The chosen selection mechanism may be classified as “tournament selection using simulated annealing based on Boltzmann decision”. The proposed method in this thesis is a variation on the original algorithm by Goldberg [267], and the reasons for choosing this method are explained subsequently.

Tournament selection involves a pure random selection of  $t$  individuals ( $t \leq \text{pop\_size}$ ) that compete in terms of fitness against each other and the winner is selected. This process is repeated  $\text{pop\_size}$  times. As  $t$  increases the selective pressure increases but typically  $t = 2$ . However, the author proposes to use a simulated annealing strategy with essentially a “fuzzy” selection decision with  $t = 2$ . Simulated annealing optimisation is based on the analogy of the slow cooling of a material from a temperature above its melting point [126]. At high temperatures there is a greater chance that weak candidates may be selected, which enhances population diversity and makes it less likely that the algorithm will get stuck in local optima. As the temperature cools the strong candidates begin to dominate selection since the algorithm should be converging on the true optimum. The choice of temperature  $T$  and its associated rate of cooling are important design decisions for a simulated annealing based algorithm.

The term “Boltzmann decision” refers back to the thermodynamic analogy mentioned earlier [126]. If the energy of a system has increased by  $\Delta E$ , the new state is accepted according to the laws of thermodynamics with probability  $p(\Delta E) = \exp(-\Delta E/kt)$ , where  $k$  in this context is Boltzmann’s constant. The proposed approach uses Equation 5.26 which is plotted along with the exponent of  $X = \frac{f(j)-f(k)}{T}$  in Figure 5.24 where  $f(j)$  and  $f(k)$  are the fitness values of candidates  $j$  and  $k$  respectively.

$$W = \frac{1}{1 + e^{\frac{f(j)-f(k)}{T}}} = \frac{1}{1 + e^X} \quad (5.26)$$

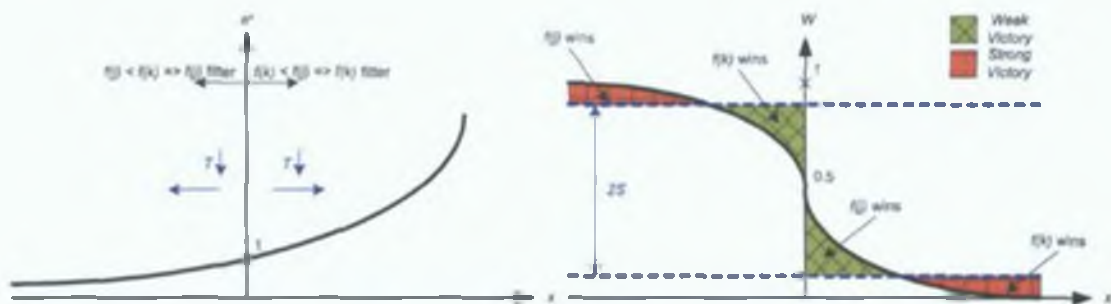


Figure 5.24: Boltzmann Decision Based Simulated Annealing

As is clear from Figure 5.24, as the temperature  $T$  decreases, the value of the exponential term  $X$  moves further from the central vertical axis for a fixed  $f(j)$  and  $f(k)$ . As  $T$  decreases  $W \rightarrow 1$  when  $f(j) < f(k)$  and  $W \rightarrow 0$  when  $f(k) < f(j)$ . The original Boltzmann tournament selection algorithm proposed by Goldberg lets  $W$  equal the probability that  $j$  wins the tournament and  $1 - W$  be the probability that  $k$  wins the tournament [267]. Goldberg proposes a tournament size  $l = 3$ , and the algorithm is summarised in Algorithm 2.

---

**Algorithm 2: Boltzmann Tournament Selection Algorithm**

---

```

j and k compete:
  Generate a random (float) number r in the range [0,1]
  if r < W then
    j wins => win1 = j
  end
  else
    k Wins => win1 = k
  end
win1 and l compete:
  Generate a random (float) number r in the range [0,1]
  if r < W then
    win1 wins => win2 = win1
  end
  else
    l Wins => win2 = l
  end
end

```

---

The author proposes a variation on Goldberg's algorithm by introducing a fuzzy select threshold  $S$  to enhance the population diversity (given that the variance of fitness values is low). Using  $S$ , the selection algorithm can be programmed to have a higher probability of selecting a weak candidate as a tournament victor when the temperature  $T$  is high in the early generations. As the temperature decreases and the algorithm converges on the optimum, the stronger candidate has a greater chance of victory. The approach is summarised in Algorithm 3.

To summarise, the proposed selection method maintains a balance between population diversity and selection strength. The selection decision depends on the relative fitness of competing individuals, the temperature  $T$  and the fuzzy select threshold  $S$ . Since the GA should converge on globally optimal solutions as the generations iterate, the parameters  $T$  and  $S$  should decay over the generations to select the strong candidates with higher probability.

---

**Algorithm 3** Fuzzy Boltzmann Tournament Selection Algorithm

---

```
if  $f(j) < f(k)$  then
  if  $W > (0.5 + S)$  then  $j$  wins (strong victory),
  else  $k$  Wins (weak victory)
end
else if  $f(j) > f(k)$  then
  if  $W < (0.5 - S)$  then  $k$  wins (strong victory),
  else  $j$  Wins (weak victory)
end
else
  What to do if they are equal ( $W = 0.5$ )? Choose pure random winner?
end
```

---

It is worth noting that the selection algorithm requires a lot of computational power – and must be run *pop\_size* times per generation. It may be more prudent to use a simpler selection method like stochastic universal sampling (spinning a roulette wheel once with *pop\_size* equally-spaced spokes). However, tournament selection using simulated annealing based on Boltzmann decision was chosen because the parameters *T* and *S* allow fine-grained control of the algorithm behaviour.

#### 5.4.4.6 Step 3 – Recombination

Recombination is the process of producing new offspring by somehow combining the information contained in two or more parents. The method used by this approach is uniform crossover. After *pop\_size* individuals have been selected for the new population (step 2) a proportion of these are further selected for crossover based on a parameter representing the probability of XOVER  $p_c$ . The process is summarised in Algorithm 4.

---

**Algorithm 4** Uniform Crossover Algorithm

---

```
for Each candidate in new population do
  Generate a random (float) number  $r$  in the range [0,1]
  if  $r < p_c$  then
    Append current candidate to XOVER list
  end
end
for Each pair of adjacent candidates on XOVER list do
  crossover(),
  Move to next pair on XOVER list
end
```

---

Note that the number of candidates selected for crossover should be even for easy pairing. If the number selected were odd, an extra candidate could be added or removed – this choice is made randomly. An example showing two selected candidates paired for crossover is shown in Figure 5.25. Since each candidate is represented by *N* pointers (in this case  $N = 4$ ) the uniform XOVER process generates a random *N*-bit mask. Each bit location in the mask determines the mixture of genetic material from the parents: each offspring is created with. Consider Figure 5.25. If a bit location is '0' the corresponding pointer component for offspring '0' is created respectively from parent '0' and that for offspring '1' is created from parent '1'. The opposite creation process occurs if the bit is '1'.



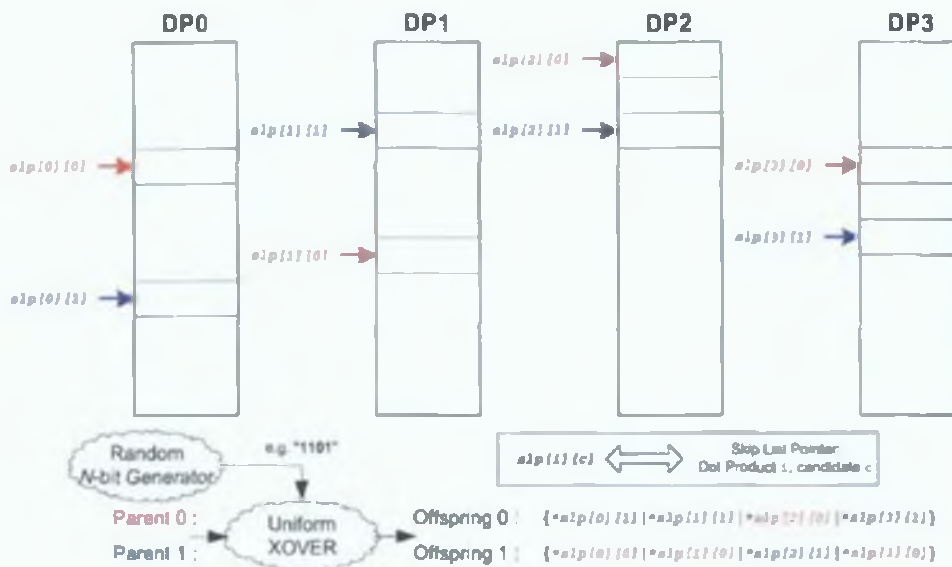


Figure 5.25: Uniform Crossover Example

#### 5.4.4.7 Step 4 – Mutation

Mutation is a genetic operator whereby individuals are randomly altered. After selection and crossover, each candidate undergoes mutation based on a probability of mutation  $p_{mut}$ , which is usually quite low. The process for applying mutation to the population is shown in Algorithm 5.

---

#### Algorithm 5: Mutation Algorithm

---

```

for Each candidate in new population do
  for Each DPL pointer component in current candidate do
    Generate a random (float) number  $r$  in the range  $[0,1]$ 
    if  $r < p_{mut}$  then
       $M = \text{generate\_mutation\_size}()$ ;
       $\text{apply\_mutation}(M)$ ;
    end
  end
end

```

---

If according to Algorithm 5 mutation is to be applied, the degree of mutation is determined by a value  $M$ , where  $M \in \mathbb{Z}$ . A pointer selected for mutation moves  $M$  pointer locations up ( $M < 0$ ) or down ( $M > 0$ ) its associated DPL skip list. This is illustrated in Figure 5.26 where the example shows  $M < 0$ . The range of mutations possible depends on the value of a parameter  $M_{max}$ . The value for  $M$  is determined based on a binomial probability density function  $p(M)$  Eqn. 5.27. This distribution means that if mutation is applied, smaller mutations are more likely than large mutations.

$$p(M) = \frac{(2M_{max} - 1)!}{M!((2M_{max} - 1) - M)!} 0.5^M (0.5)^{((2M_{max} - 1) - M)} \quad (5.27)$$

To allow positive or negative mutations (up or down the DPL skip lists) the binomial distribution is re-aligned about  $M = 0$  (where  $p(0) = 0$  because  $M = 0$  means no mutation). This re-alignment is

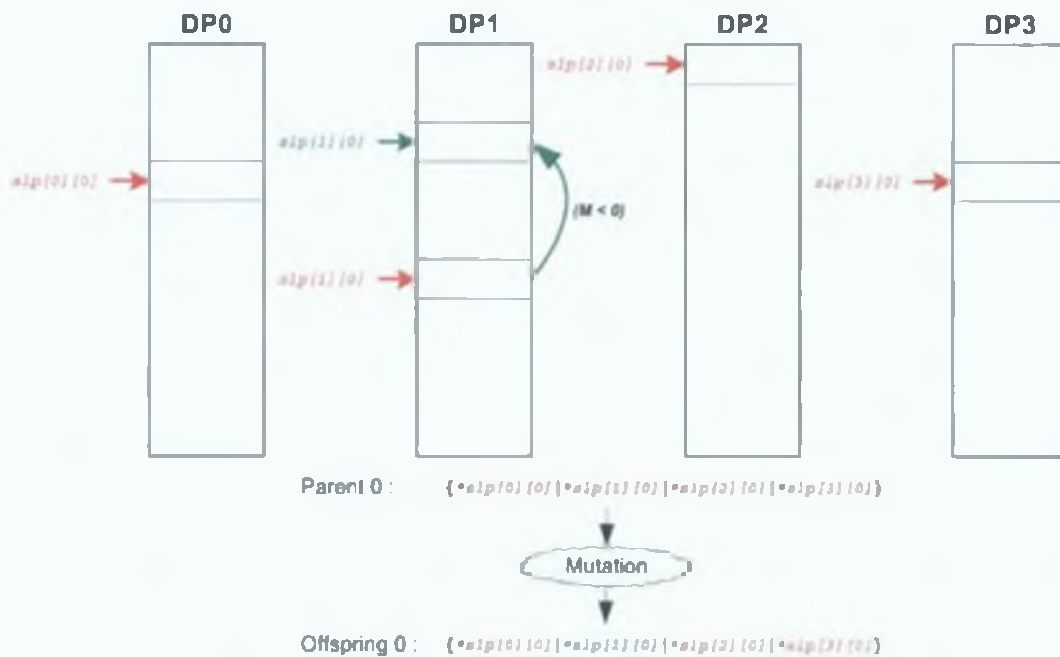


Figure 5.26: Mutation Example

illustrated in Figure 5.27 and Figure 5.28 for  $M_{max} = 6$ .

After this step the new population forming the next generation is ready and the process loops back to step 1. The process continues looping through steps 1–4 until a termination condition is met. The termination condition employed in the current implementation is a fixed number of generations, but could equally be a time constraint.

#### 5.4.5 Genetic Algorithm Parameter Selection

Choosing the values for the parameters that steer a GA is widely acknowledged to be the most difficult step in the implementation of a successful GA [266]. This is even more complicated if the parameters are allowed to change across the generations since their direction and rate of change must be determined. A summary of the parameters controlling the proposed GA is outlined in Table 5.2.

The parameter values in Table 5.2 have been obtained empirically by trial and error, starting with “sensible” values for each of the parameters determined intuitively. The population size (*pop\_size*) is quite large because in the CMM problem, the variance of the fitness values of the solution space is relatively small compared to the size of the solution space. A large value for *pop\_size* increases the diversity of the solutions at the expense of computation time per generation. The smaller the initialisation weight  $\mu$ , the nearer the initial population candidate component pointers are to the top of the DPL skip lists. Good solutions are expected to have component pointers near the top of the skip lists, but if  $\mu$  is too small, the population diversity is not sufficient. Higher values for temperature  $T$  give weaker solutions a better chance of being selected ahead of stronger solutions to encourage diversity. However,  $T$  decays exponentially as the generations iterate since the algorithm should converge on “good” solutions. The

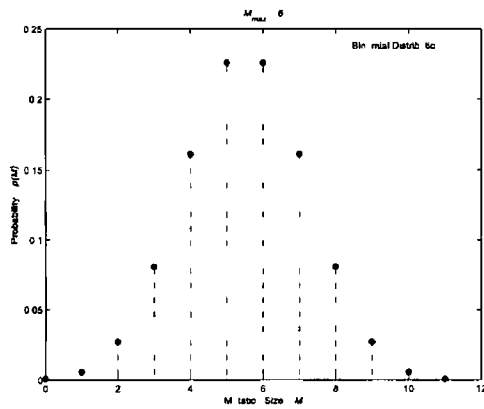


Figure 5 27 Binomial Distribution Function

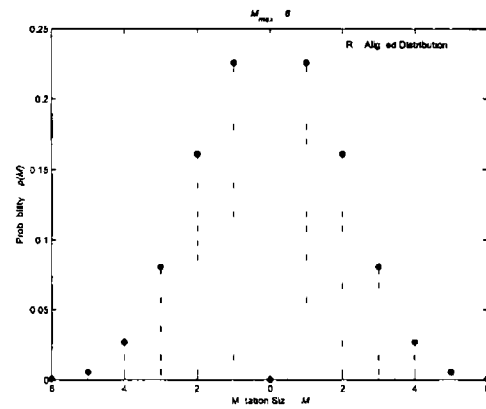


Figure 5 28 Re-Aligned Distribution

Table 5 2 CMML Genetic Algorithm Parameters

Parameter	Name	Value
$pop\_size$	Population Size	3000
$\mu$	Initialisation Weight	10 0
$T$	Selection Temperature	0 001
$S$	Selection Threshold	0 4
$p_c$	Crossover Probability	0 98
$p_{mut}$	Mutation Probability	0 08
$M_{max}$	Max Mutation Size	6

fuzzy selection threshold  $S$  also controls the balance between selective pressure and diversity. A larger  $S$  implies better chance for a weak victory in the tournament selection process. A typically quoted range in the literature for crossover probability  $p_c$  is  $0.8 \leftrightarrow 0.9$  [266]. Frequent crossover encourages good solutions to breed with each other in the hope of finding even better solutions. A degree of mutation encourages diversity, but mutation probability  $p_{mut}$  is usually quite small ( $<0.01$ ), since large  $p_{mut}$  tends to lose good solutions. However, trial and error experimentation has shown that for the proposed CMM optimisation algorithm, a relatively large value for  $p_{mut}$  leads to better solutions. A large value for  $M_{max}$  also increases diversity, since it determines the size of a mutation when mutation does occur according to  $p_{mut}$ .

Based on trial and error experimentation, the tuned parameter values in Table 5 2 have been established. The values for these parameters imply that the proposed CMML genetic algorithm produces better results when there is weak selective pressure (strong diversity). It is estimated that the reason for this is because the variance of the solution space fitness values is quite low, according to the current fitness function. In a sense, the current search is almost a “needle in a haystack” search, so a healthy diversity is needed. Extending the fitness function should increase the granularity of the fitness values in the solution space. Hence the tuned genetic algorithm parameters are likely to change so that the selective pressure will increase.

Table 5.3: 1D 8-point DCT/IDCT Adder Unit / Full Adder Requirements

CMM	Initial	[40]	[241]	[242]		Proposed Approach					
	+	+	+	+	FA	Untuned GA [268]			Tuned GA [269]		
						+	FA	FA%	+	FA	FA%
DCT 8bit	300	94	63	56	739	78	730	1.2	77	712	3.7
DCT 12bit	368	100	76	70	1202	109	1056	12.1	108	1048	12.8
DCT 16bit	521	129	94	89	2009	150	1482	26.2	141	1290	35.8
IDCT 14bit	444	n/a	n/a	81	1196	n/a	n/a	n/a	93	934	21.9

Table 5.4: Operating Conditions of Proposed CMM Optimisation Algorithm Test Cases

CMM	Parallel Machines*	Operating System	Processor	SD Search Space		DPL Runtime	CMML GA Runtime
				DPL	CMML		
DCT 8bit	1	Windows 2000	Pentium-4M 2GHz	10 <sup>6</sup>	10 <sup>40</sup>	Minutes	Minutes
DCT 12bit	8	Windows 2000	Pentium-4 2GHz	10 <sup>8</sup>	10 <sup>62</sup>	≈24 hours	Minutes
DCT 16bit	8	Windows 2000	Pentium-III 733MHz	10 <sup>10</sup>	10 <sup>81</sup>	≈6 days	Minutes
IDCT 14bit	8	Windows 2000	Pentium-4M 2GHz	10 <sup>9</sup>	10 <sup>65</sup>	≈24 hours	Minutes

Note\*: Parallel machines used to run the DPL algorithm

#### 5.4.6 CMML Results

For a fair comparison with other approaches, the number of 1-bit full adders (FAs) allocated in each optimised architecture should be used as opposed to "adder units". For example, a solution that requires 10 × 8-bit adder units (80 FAs) is more attractive than a solution requiring 5 × 32-bit adder units (160 FAs) in terms of circuit area, despite the fact that it requires 10 adder units as opposed to 5. It is clear that FA count more accurately represents circuit area requirements. Unfortunately, the bitwidth of each unit of the solutions are unspecified in other publications apart from in [242]. Using the 8-point 1D DCT/IDCT ( $N = 8$  with various  $M$ ) as a benchmarking CMM problem, Table 5.3 compares results with other approaches based on adder units and FAs where possible. Two sets of results are presented in Table 5.3 – one set are preliminary solutions found after 100000 generations of the proposed GA with un-tuned parameters [268], and the second set are found after 1000 generations of the GA configured with the tuned parameters listed in Table 5.2 [269]. The proposed approach compares favourably with Boullis and Tisserand [242] in terms of FAs (see FA% savings in Table 5.3), even though this gain is not reflected by the number of adder units required. Gains are evident using the untuned parameters – from 1.2% for the 8-bit DCT increasing to 26.2% for the 16-bit DCT. As expected, increased gains are achieved using the tuned GA – from 3.7% for the 8-bit DCT to 35.8% for the 16-bit DCT. It is clear that more gains are achieved as the size of the problem increases (increasing  $M$ ). This is because there are many more SD permutations possible as  $M$  increases, coupled with the fact that the proposed algorithm has more scope to intelligently find near optimal solutions using the genetic algorithm. The algorithm by Boullis and Tisserand does not search the SD permutation space and uses heuristic search techniques, which results in sub-optimal results compared to the results achieved by the proposed algorithm.

The improved results achieved by the proposed algorithm come at the cost of execution time. However, despite the large search spaces involved when examining all SD permutations, the proposed al-

gorithm is still tractable. Unfortunately, it is not possible to compare the proposed CMM optimisation algorithm against relevant prior art in terms of execution time. For most prior art algorithms, execution time is not discussed since it is not necessarily an issue because those algorithms do not address the huge search spaces associated with SD permutation. However, even without SD permutation, the general sub-expression elimination problem is quite difficult. In their paper, Boullis and Tisserand state that the execution time of their algorithm ranges from a few minutes to a few hours depending on the run-time optimisation effort of the algorithm [242]. The execution time for the proposed algorithm is important since the gains achieved by the algorithm come at the cost of extra computation. The computational bottleneck with the proposed algorithm is the DPL stage, which is described in Section 5.4.2. However, as clear from Figure 5.16, the DPL optimisation of each dot product in a CMM can be optimised independently in parallel and this has been exploited when generating the results presented in Table 5.3. The operating environment for each test scenario listed in Table 5.3 are presented in Table 5.4. It is clear that as  $M$  increases, the execution time of the DPL algorithm increases significantly. Unfortunately the computers used for each test run were different due to the limited resources available to the author, so it is difficult to extrapolate an accurate relationship between problem complexity  $M$  and the execution time. It is interesting to note that the execution time of the CMML GA is independent of  $M$ . It is postulated that the reason for this is that the ordering of the DPL solutions by step 5 of the DPL algorithm (see Section 5.4.2) means that the CMML GA searches the most promising regions of the overall solution space first. Nevertheless, it is clear that the execution time of the DPL algorithm and the computing resources required are concerns for large  $M$  or for larger CMM problems. As outlined in Section 5.4.2.6, the computational bottleneck of the DPL algorithm is the bit counting task. Section 5.4.2.6 also outlines potential avenues for future research to improve the performance of the current implementation.

Despite the execution time concerns for large  $M$ , the modelling approach used means that the proposed DPL algorithm is tractable and can search the DPL SD permutation space exhaustively. As is evident from Table 5.4, the CMML search spaces are huge, and this is why the genetic programming approach was followed when designing the algorithm. The results in Table 5.3 are based in searching much less than 1% of the CMML search space, and this is why the CMML execution time is of the order of minutes. However, it is clear that despite searching such a small amount of the potential solutions, the solutions obtained improve on the prior art, and this is attributable to the way in which the problem is modelled and the search space is organised. Also, the hypothesis of achieving extra saving by permuting the SD representations is validated by the fact that the best SD permutation yielding the results in Table 5.3 are not the CSD permutation. It is also interesting to note that for each of the benchmarks in Table 5.3, the tuned GA parameters cause the proposed algorithm to invoke its search space reduction mechanism (see Section 5.4.3 for details). This reduces the search space for each DCT benchmark from the orders listed in Table 5.4 to the order of  $10^{17}$  without compromising the quality of the results. Even though a search space of  $10^{17}$  is still very large, the fact that the reduction mechanism is invoked shows that the genetic algorithm is converging on near-optimal solutions quite quickly.

## 5.5 MCMM Extension

This section explains how the CMM optimisation algorithm proposed in this chapter may be leveraged to optimise the SA-DCT problem as described in Chapter 3 (also covers SA-IDCT problem of Chapter 4)

Section 5.2 classifies multiplication by constant applications into four categories: single constant multipliers, constant multiplier blocks, digital filters and constant matrix multiplication (CMM). The adaptive nature of the SA-DCT means it is essentially multiple related CMM (MCMM) problems. Depending on the amount of shape pixels  $N$  in a vector  $x$ , an  $N$ -point 1D DCT CMM is computed on  $x$ . There are 8 possible CMMs ( $1 \leq N \leq 8$ ), as is clear from Figure 5.29. The key point to note is that all 8 CMMs are related since they operate on the same data vector  $x$  (actually different ranges of  $x$  depending on  $N$ ). The MCMM interpretation of the SA-DCT is to combine the 8 basis matrices into one large  $36 \times 8$  basis matrix that operates on  $x$  as shown in Figure 5.29. When combining the constant basis matrices to get the  $36 \times 8$  matrix, all  $N \times N$  matrices ( $N \neq 8$ ) are padded to  $8 \times 8$  matrices by padding an appropriate amount on zero-valued columns.

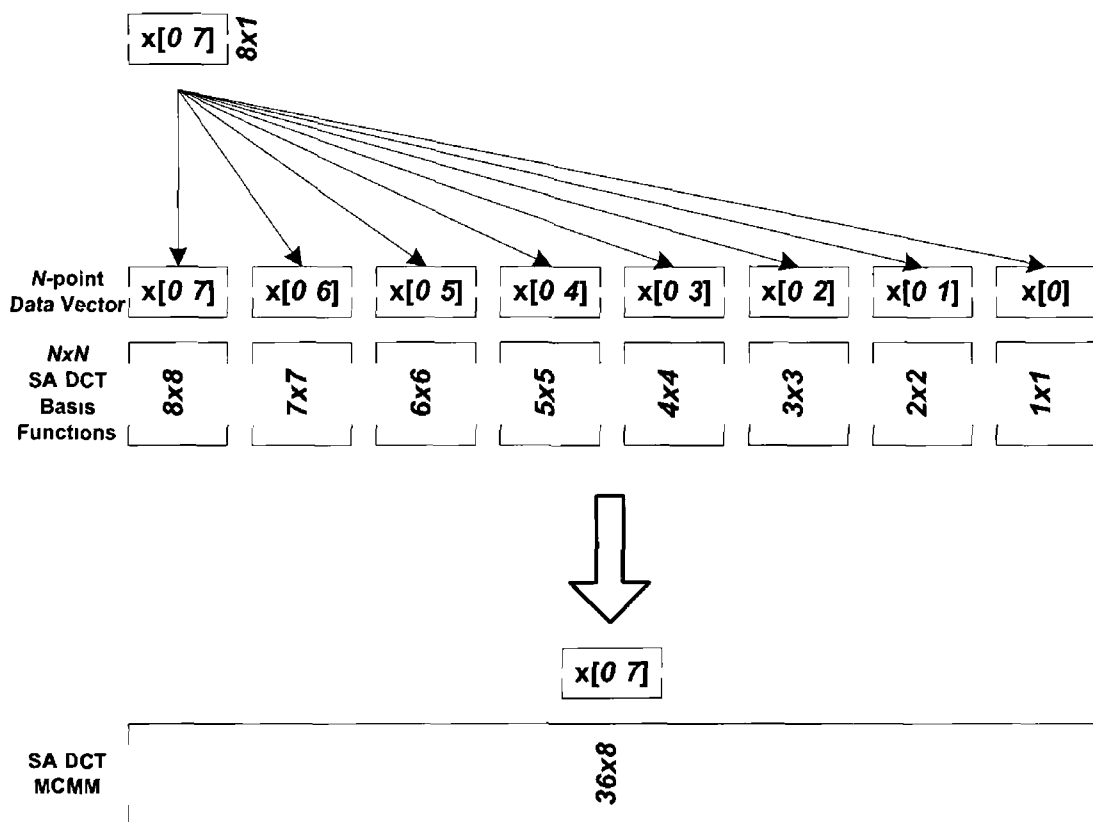


Figure 5.29 SA-DCT as an MCMM Problem

Using the MCMM representation, the SA-DCT may be regarded as a collection of 36 dot products. For reasons including area, bandwidth, power and the nature of the incoming data, the SA-DCT architecture proposed in Chapter 3 does not use an MCMM computation unit capable of computing the 36 dot products in parallel. Instead, a time-multiplexed computation unit is used that can be configured to compute one of the 36 dot products per cycle (a serial computation scheme). The core SA-DCT architecture described in Chapter 3 may be described at a high level by the diagram in Figure 5.30. The sub-expression addends from data vector  $x$  are time-multiplexed depending on which one of the 36 SA-DCT MCMM dot products is being computed at that instant. The sub-expression sums (the adder

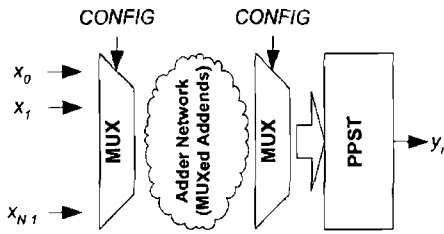


Figure 5 30 Time-MUX Addends & PPST

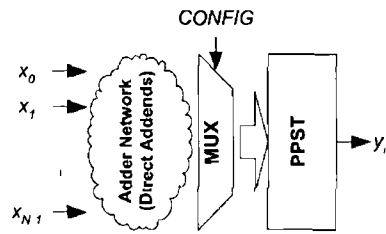


Figure 5 31 Time-MUX PPST

outputs) that drive the PPST are also time-multiplexed in the same manner. Both time-multiplexing stages are clearly illustrated in Figure 5 30.

There are two possible approaches for optimising the SA-DCT MCMM with the DPL/CMML algorithms. The first approach is based on the fact that only one of the 36 SA-DCT MCMM dot products are ever computed at any one instant (at least according to the computation scheme proposed in Chapter 3). The DPL algorithm could be run on each of the 36 dot products. Since the addends are time-multiplexed, the dot product with the worst (highest) value of `num_nonzeros` at the top of its SNL (top left node in Figure 5 20) represents the maximum number of sub-expression adders that will ever be needed for a single dot product computation. Essentially this is the dot product with the worst results in terms of adder resource requirements according to the DPL algorithm. The CMML algorithm is not necessary with this approach since the addends are time-multiplexed depending on which dot product is being computed.

The alternative optimisation approach is to run the CMML algorithm on the DPL results for the 36 dot products. The essential difference here is that the sub-expression addends are not time-multiplexed since the CMML algorithm finds all the adders that are necessary to implement all 36 dot products. The architecture in this case would look like Figure 5 31. It is likely that this second approach will require more adder resources since the CMML algorithm searches builds adder resource solutions as if all MCMM dot products are computed in parallel. However, this second approach eliminates the first stage of multiplexers that configure the sub-expression addends.

## 5.6 Summary of Contributions

Digital signal processing algorithms contain many instances of various kinds of multiplication by constant operations. This chapter has classified these operations into four broad categories: single constant multipliers, constant multiplier blocks, digital filters and CMMs. Multiplication by constants can be implemented efficiently in hardware using only additions and/or subtractions coupled with shift operations instead of implementing a full multiplier. The optimisation task to find the optimal adder/subtractor sub-expressions is a very difficult problem, and it is claimed to be NP-complete although a proof has not yet been published [243]. In this context, Section 5 2 explains why CMMs are the most challenging application in terms of optimising multiplication by constant operations. Section 5 3 explains the CMM problem in more detail and three properties are discussed that can be used in the classification of approaches to this problem: SD permutation, pattern search strategy and problem subdivision. Then, a comprehensive prior art review is outlined based on these properties. The conclusion is that not much prior art leverages the potential of SD permutation, and those that do only apply it to a limited extent on simpler problems and not for CMMs. Much of the prior art also risks sub-optimal results because they

use greedy heuristics to select sub-expressions

The CMM optimisation algorithm proposed in this thesis (Section 5.4) incorporates SD permutation. The algorithm splits the problem into tractable sub-problems, and avoids risking sub-optimal results by building parallel partial solutions for each of the sub-problems. The partial solutions for each of the sub-problems are ordered cleverly allowing the algorithm to combine the partial solutions effectively when searching for global solutions. Section 5.4 outlines the algorithm in detail. The algorithm partitions the global constant matrix multiplier into its constituent dot products, and all possible solutions are derived for each dot product in the first two stages. The third stage leverages the effective search capability of genetic programming to search for global solutions created by combining dot product partial solutions. A bonus feature of the algorithm is that the modelling is amenable to hardware acceleration. Another bonus feature is a search space reduction early exit mechanism, made possible by the way the algorithm is modelled. Section 5.4.6 benchmarks the proposed algorithm where possible against prior art. The experimental results show an improvement on state of the art algorithms with future potential for even greater savings.



## 6.1 Conclusions

This section summarises the research goals of this thesis and discusses the results achieved by the work outlined in previous chapters

### 6.1.1 Motivation for Proposed Research – A Summary

The world of portable electronics is experiencing an ongoing trend of feature enhancement and device convergence. This is perhaps most evident in the field of mobile communications where ever more sophisticated smartphones are constantly appearing that incorporate a broader spectrum of applications. For many, the mobile phone is now their phone, their PDA (with calendars, contacts, etc), their email client and their camera all in one device. As silicon implementation technology continues to improve at an astonishing rate, the range of possible applications is seemingly only limited by human imagination. From a phone manufacturer's perspective, intent on selling more of their devices, this is an ideal state of affairs – the device is now no longer “just” a phone but a general-purpose multimedia platform that can be leveraged by many different applications. A vision of the future evolution of these mobile applications leads to an interactive personalised multimedia experience that is context aware (i.e. adapts to location and environment) [24]. This vision of a futuristic convergent device and the associated applications are sometimes referred to as “e-Dreams” (see Chapter 1 and [25]).

Unfortunately, there are many grand challenges that impede the emergence of the convergent device and truly ubiquitous computing. The challenges involved can be broadly categorised into three problems: application development, content delivery and content processing. These challenges are described in Chapter 1 and summarised here. Firstly, the applications developed should be user friendly. Engineers should tailor the design of mobile applications to people, rather than expecting people to adapt to technology. The applications on a convergent device should be an adequate substitute for a dedicated device for that application. For example, the camera module on a mobile phone should be of sufficient quality compared to the performance of a dedicated camera (at least for the majority of users). Secondly, the current networking infrastructure should be improved to support the envisaged applications (sufficient bandwidth, security and ease of use, etc). Finally, and arguably most importantly, the emergence

of e-Dreams is hampered by hardware limitations of the convergent mobile device itself and the “nano-scale hell of physics” caused by relentless CMOS scaling [25]. These limitations include a limited user interface, limited processing capability and most pressingly limited battery energy. Since users expect extended battery life on each new generation of device, this poses significant problems to those who are designing the hardware for the multimedia platforms.

Since video processing is arguably the most computationally demanding form of data to process, and increased computation means increased power consumption, the e-Dream applications being dreamt up are certain to lead to an unacceptably poor battery life unless this problem is tackled. In light of these trends, this thesis proposes power efficient hardware accelerators for some of the basic enabling technologies (SA-DCT/IDCT) that are required to implement a selection of video processing functionalities on mobile devices. The power consumption problem is addressed at the algorithmic and architectural abstraction levels since these levels provide the greatest scope for savings (see Chapter 2). Investigation into architectures for the SA-DCT/IDCT has led the author to realise that these transforms are instances of a more general class of computation that is commonplace in multimedia processing, namely the constant matrix multiplication (CMM) operation. Thus, this thesis also proposes a novel optimisation algorithm for the design of circuitry for any general CMM equation. The algorithm chooses a set of adder sub-expressions using a comprehensive search strategy to reduce the operator count required to compute a CMM equation, and consequentially circuit area and power are reduced.

## **6.1.2 Summary of Thesis Contributions**

Chapter 1 gives a broad context for the research work described in this thesis. This context is summarised in Section 6.1.1, and the essential theme is that shortened battery life due to the excessive power consumption required by video processing is a key challenge facing hardware designers of next generation mobile multimedia platforms. Chapter 1 also outlines the goals and objectives of the thesis, followed by a synopsis of the thesis structure. The objectives of this work are listed in Section 1.3. As summarised in this section, Chapters 3 and 4 address objectives 1–3, and Chapter 5 addresses objectives 4 and 5.

Chapter 2 provides a more technical context for the research work in this thesis. To achieve this objective, a generic video compression system is described to illustrate how the DCT/IDCT contribute to the global application. A mathematical foundation for transform theory and the DCT/IDCT in particular is outlined to explain how the DCT/IDCT can transform video data into a form more amenable to compression. This is followed by an overview of industrial image and video compression standards, which plots their evolution from block-based compression standards like MPEG-1 and H.261 to object-based compression and the MPEG-4 standard. MPEG-4 supports content-based functionalities by encoding arbitrarily-shaped image segments corresponding to semantic objects. Object-based compression enables a whole new paradigm of multimedia applications where the user can manipulate semantic objects in a scene. The overview of MPEG-4 leads on to an outline of an object-based compression system which highlights how the SA-DCT/IDCT fits into this system. The SA-DCT algorithm is outlined and the reasons why the MPEG-4 standardisation body adopted the SA-DCT as opposed to alternative algorithms are summarised.

The low power design paradigm is also outlined in Chapter 2. The circuit power dissipation phenomena in digital CMOS circuitry are enumerated highlighting the alarming trends evident as CMOS scaling reaches deep sub-micron linewidths. Aggressive scaling coupled with increasing application conver-

gence render power as a primary design constraint along with area and speed. With the sources of power consumption described, Chapter 2 outlines some common metrics and techniques used by engineers to analyse their designs in terms of power consumption. Then, the traditional HDL design flow is summarised highlighting where power optimisations can be made (both pre-synthesis and by the synthesis tool itself). A comprehensive survey is given detailing the most popular techniques for addressing the power consumption problems in modern CMOS circuits. Different techniques are used at the various design abstraction levels pre-synthesis, but the survey concludes that there is greatest scope for savings at the system and algorithmic/architectural levels because of the wider degree of design freedom at these abstractions. Recurring themes include common sense approaches such as avoiding needless computation and trading area/speed for power [37]. It is this high-level approach that has been adopted for the design of the SA-DCT and SA-IDCT cores proposed in Chapters 3 and 4. Heavy duty synthesis tools can further boost power savings by leveraging sophisticated EDA algorithms. The optimisations achieved by synthesis tools are generally difficult to employ manually by a user. Indeed many of the optimisations that EDA algorithms attempt may be characterised mathematically as NP-complete problems, so in general clever heuristic approaches are required [126, 127]. This scenario provides the backdrop for the author's CMM optimisation EDA algorithm presented in Chapter 5.

The SA-DCT hardware architecture proposed in this thesis is described in Chapter 3. Its distinguishing low energy features include minimal switching shape decoding and data alignment, data dependent clock gating, multiplier-free datapath (using adder-based DA), balanced delay paths and hardware resource re-use. With respect to prior art, benchmarking is difficult but Chapter 3 uses some normalisation metrics to facilitate comparisons where possible. In terms of area and latency, the PGCC metric shows that the proposed architecture outperforms the Chen [109] and Lee [203] architectures. In terms of energy and power, the proposed architecture outperforms the Chen [109] and Tseng [201] architectures. The proposed SA-DCT has also undergone conformance testing via the MPEG-4 Part 9 initiative, validating that it is compliant with official MPEG requirements.

Chapter 4 presents the SA-IDCT architecture proposed in this thesis, which has similar architectural properties to the forward SA-DCT architecture described in Chapter 3. However, peculiarities of the SA-IDCT algorithm mean that the shape parsing and data alignment steps are more complicated compared to the SA-DCT. To address this issue, Chapter 4 presents a parallel addressing scheme that regenerates the reconstructed pixel address location as the pixel value itself is being computed by the datapath. Chapter 4 uses the same normalisation metrics as Chapter 3 to benchmark the proposed SA-IDCT architecture against prior art works. In terms of area and latency, the PGCC metric shows that the proposed architecture outperforms the Chen [109] architecture and the Hsu [235] architecture (in no skip mode). Considering normalised energy and power, the proposed architecture also outperforms the Chen [109] architecture and the Hsu [235] architecture (again in no skip mode). However, the Hsu architecture also has a zero skipping mode that exploits the likelihood of zero valued coefficient data to reduce computational latency and hence both PGCC and energy. Since the savings achievable due to zero skipping are data dependent and independent of the architecture, it is expected that in future if a zero skipping mode is integrated into the proposed design, it will improve upon the zero skipping mode of the Hsu architecture. As well as benchmarking against the prior art, the proposed SA-IDCT has also undergone conformance testing via the MPEG-4 Part 9 initiative, validating that it is compliant with official MPEG requirements.

In Chapter 5, an optimisation algorithm is presented that can be employed to realise optimal circuitry

for any general CMM problem. The approach proposed in Chapter 5 presently uses adder count as the sole optimisation criterion, but is envisaged that more criteria could be integrated in future. Before detailing the proposed algorithm itself, Chapter 5 outlines a taxonomy of multiplication by constant operations to give context for the CMM problem. Then, a comprehensive review is presented of relevant prior art which is divided between graph-based approaches and common sub-expression elimination (CSE) approaches. In their comparison work [134], Dempster et al. conclude that for simpler problems graphical methods are best, while CSE works better for the more complex problems (such as CMM). For this reason, the algorithm proposed in this thesis is CSE-based. The algorithm itself incorporates signed digit permutation of the CMM constants since greater potential exists for improved solutions. The algorithm splits the problem into tractable sub-problems, and avoids risking sub-optimal results by building parallel partial solutions for each of the sub-problems. The partial solutions for each of the sub-problems are ordered cleverly allowing the algorithm to combine the partial solutions effectively when searching for global solutions. Chapter 5 concludes by benchmarking the algorithm against relevant prior art works. The experimental results show an improvement on state of the art CMM optimisation algorithms such as the Boullis and Tisserand algorithm [242]. It must be noted that even though the proposed algorithm improves upon prior art, there is scope for improvement as described in Chapter 5 and summarised in Section 6.2.

### 6.1.3 Research Objectives Achieved

The contributions of this thesis as discussed in Section 6.1.2 may be summarised by the following list of achieved research objectives:

1. Energy-efficient hardware architectures for the SA-DCT/IDCT enabling technologies have been designed and implemented according to the low power design techniques described in Chapter 2. The discussion in Chapter 2 demonstrates that most power and energy savings are achievable at the higher levels of design abstraction. Hence the techniques employed are at the algorithmic and RTL abstraction levels.
2. It has been verified that both the SA-DCT and SA-IDCT architectures conform to the official MPEG-4 standard requirements by comparing the bitstreams of a software-only MPEG-4 codec to an MPEG-4 codec that computes the SA-DCT/IDCT using the proposed hardware architectures. Hence the proposed designs are viable solutions for MPEG-4 products that have tight power and energy constraints.
3. A set of formulae have been derived to facilitate normalised power and energy benchmarking of different implementations of any general basic enabling technology. These formulae are important as they allow fair comparisons to be made by normalising the technology specific parameters of different implementations.
4. Using the derived normalisation formulae, the SA-DCT/IDCT architectures have been evaluated as fairly and accurately as possible against prior art in the literature in terms of area, speed, power and energy. Chapters 3 and 4 demonstrate that the proposed architectures compare favourably against the prior art.

- 5 An innovative hardware resource allocation EDA algorithm has been designed that optimises the hardware implementation of any general CMM operation. The proposed algorithm extends the state of the art since it considers different SD representations of the matrix constants as opposed to limiting the analysis to 2's complement or CSD. This approach leads to very large search spaces, but this has been addressed by clever data modelling and search space organisation. A genetic algorithm has also been designed to search these large search spaces intelligently.
- 6 The proposed CMM algorithm has been evaluated where possible against prior art that targets the CMM problem specifically. Chapter 5 demonstrates promising results that validate the approach adopted, with scope for even greater improvement.

## 6.2 Future Work

This section indicates potential directions for future research based on the work outlined in this thesis. Potential tweaks and variations of the SA-DCT/IDCT architectures that may improve power consumption performance are suggested, as well as extensions for the pre and post processing steps used by the MPEG-4 standard. Since the EDA CMM optimisation algorithm proposed in Chapter 5 tackles a very difficult problem with very little previous work in the specific area, there is significant scope for further improvements and some suggestions are outlined. Another potential use for the SA-DCT hardware accelerator (apart from object-based compression) is also proposed. This is important since silicon is limited on a mobile platform, and ideally hardware accelerators should be configurable for multiple applications to add value to the accelerator and further justify its existence.

### 6.2.1 SA-DCT/IDCT Accelerator Variations and Improvements

The SA-DCT/IDCT architectures proposed in this thesis compute their output results serially, i.e. one coefficient or reconstructed pixel is produced at a time. This approach was adopted since the designs have low silicon area and still meet real time requirements. The experimental results in Chapters 3 and 4 show that the serial architectures also have attractive power and energy properties. Clearly though, it is possible to vary the amount of parallelism in the computation method of the designs. A more parallel datapath will compute results in a shorter amount of time at the expense of additional silicon area. This trade off is discussed in more depth in Section 3.7 in the context of the SA-DCT architecture proposed in this thesis, but the same ideas apply to the proposed SA-IDCT architecture. However, the effect of varying the parallelism on power and energy is not immediately obvious and future work should investigate this for the proposed SA-DCT/IDCT designs.

### 6.2.2 SA-DCT/IDCT Accelerator Pre/Post Processing

The SA-DCT/IDCT algorithms defined by Sikora et al. [20] are classified as a pseudo-orthonormal transform pair (orthonormality is defined in Chapter 2). This means that the SA-DCT/IDCT is not orthonormal as a 2D transform, but each of the variable  $N$ -point 1D transforms are individually orthonormal, and in the context of this discussion, it is referred to as the PO-SA-DCT/IDCT. However, it has been shown that the PO-SA-DCT causes what is referred to as the *mean weighting defect* [270]. This means that applying the PO-SA-DCT on a boundary pixel block with all the same grey levels generates some

AC coefficients along with a DC coefficient. Based on the transform theory described in Chapter 2, a true orthonormal transform should generate a DC coefficient only and no AC coefficients if all pixels have the same grey level. What this means in practice is that after AC quantisation and PO-SA-IDCT, the reconstructed grey pattern is unacceptably degraded. A way around this would be to use a non-orthonormal SA-DCT/IDCT (NO-SA-DCT/IDCT). However the use of the NO-SA-DCT/IDCT causes what is called the *noise weighting defect* [270]. Essentially this means that the quantisation noise is not exclusively controlled by the quantiser, but is additionally influenced in an uncontrollable manner by the shape.

Since the PO-SA-DCT/IDCT causes the mean weighting defect, and the NO-SA-DCT/IDCT causes the noise weighting defect, it seems that one must accept one form of degradation, with the noise weighting defect more tolerable. However, both defects are avoidable using the PO-SA-DCT/IDCT if and only if the image data has a mean of zero. Kauff et al. have exploited this idea and have developed an algorithm called the  $\Delta$ DC-SA-DCT/IDCT [270], which is used by the MPEG-4 standard codecs for intra coded macroblocks on the VOP boundary. For inter coded macroblocks the regular PO-SA-DCT/IDCT is used. Because of this, it is clear that an interesting path for future research would be to integrate a  $\Delta$ DC pre-processing block for the SA-DCT architecture proposed in this thesis and a  $\Delta$ DC post-processing block for the SA-IDCT architecture. The following two sections outline the  $\Delta$ DC-SA-DCT/IDCT algorithms and hint at possible steps for future work.

#### 6.2.2.1 MPEG-4 SA-DCT Pre-Processing

In a codec that is MPEG-4 compliant, the following coding steps are followed for intra coded macroblocks on the VOP boundary. These steps are referred to as the  $\Delta$ DC-SA-DCT as defined by Kauff et al. [270].

1. Establish the summation of all of the valid VOP pixel values in the current block

$$check\_sum = \sum_{i=0}^7 \sum_{j=0}^7 f[j][i] \times \alpha[j][i] \quad (6.1)$$

2. Establish the number of VOP pixels in the current block

$$num\_pixels = \sum_{i=0}^7 \sum_{j=0}^7 \alpha[j][i] \quad (6.2)$$

3. Establish the mean pixel value in the current block

$$mean = \frac{check\_sum}{num\_pixels} \quad (6.3)$$

4. Subtract *mean* from each VOP pixel to achieve zero mean image data
5. Compute the regular SA-DCT steps (Sikora's PO-SA-DCT) as defined by [20]
6. Overwrite  $F[0][0]$  with  $F[0][0] = mean \times 8$ . This essentially sets the DC coefficient to be a scaled mean value of the VOP pixels in the block. The value is scaled to ensure it is scaled in the same way as it would be scaled in the standard orthonormal DCT.

From the above steps it is clear that a  $\Delta$ DC-SA-DCT extension to the current SA-DCT architecture proposed in Chapter 3 is a pre-processing module. The only other modification is  $F[0][0] = \text{mean} \times 8$ , which can be achieved with a simple shift by three binary digit places to the left ( $8 = 2^3$ ). Some ideas that could be explored in future work include

- Presently the SA-DCT architecture reads pixel data and the co-located alpha value in a serial vertical raster of the  $8 \times 8$  block. During this process the architecture could compute the following pseudo-code

```
for(col = 0, col < 8, col++)
  for(row = 0, row < 8, row++)
    if (alpha == 255)
      begin
        check_sum    <= check_sum + data_in_1[row][col],
        dc_buffer[Nv[col]][col] <= data_in_1[row][col],
        num_pixels   <= num_pixels + 6'b000001,
        Nh[Nv[col]]  <= Nh[Nv[col]] + 4'b0001,
        Nv[col]      <= Nv[col] + 4'b0001,
      end
```

- After 64 cycles the architecture will know all vertical/horizontal  $N$  values for each column/row, and be able to calculate the mean value. Therefore it is possible to access `dc_buffer` in `num_pixels` clock cycles and subtract the mean from the data as it is being read.
- A divider architecture is required for the mean calculation. One possible option is a Sweeney Robertson Tocher (SRT) divider [160]. An SRT divider is analogous to a Booth multiplier in that it can terminate the computation earlier by exploiting strings of consecutive zeros in the dividend.
- This mean subtracted data is then routed to the SA-DCT module along with the appropriate  $N$  value. In this way the SA-DCT circuit no longer needs to calculate the  $N$  values and can speed up vertical processing since it can process vectors in  $N$  cycles since data is already aligned in `dc_buffer`.
- The final overwriting of  $F[0][0]$  can be done by simply shifting `mean` three bits to the left to implement the multiply by 8.
- It is possible to pipeline the  $\Delta$ DC pre-processing steps and the regular SA-DCT processing. When computing the SA-DCT on block  $b$ , the  $\Delta$ DC pre-processing module can be processing block  $b+1$ .

If the additional silicon area of a  $\Delta$ DC pre-processing module is a concern, it is possible to use the existing SA-DCT architecture without any additional logic at the cost of computation latency. This can be achieved by processing a given block twice with the existing architecture. On the first pass only  $F[0][0]$  needs to be calculated since it can be reverse engineered to compute mean very easily. Then, on the second pass mean could be subtracted from the incoming pixel data as it is read. Hence, the  $\Delta$ DC-SA-DCT can be computed with little extra hardware expense (since `dc_buffer` is not needed) at the cost of latency since each block must be processed twice by the architecture. However, the second pass can be computed faster since all the  $N$  values have already been established with the first pass.

### 6.2.2.2 MPEG-4 SA-IDCT Post-Processing

The inverse process corresponding to the steps outlined in the previous section is referred to as the  $\Delta$ DC-SA-IDCT [270]. In a codec that is MPEG-4 compliant, the following coding steps are followed for intra-coded macroblocks on the VOP boundary:

- 1 Regenerate the mean pixel value from the DC coefficient

$$mean = \frac{F[0][0]}{8} \quad (6.4)$$

- 2 Compute the regular SA-IDCT steps (Sikora's PO-SA-IDCT) as defined by [20]
- 3 Establish the summation of all of the valid VOP pixel values in the reconstructed block

$$check\_sum = \sum_{i=0}^7 \sum_{j=0}^7 f[j][i] \times \alpha[j][i] \quad (6.5)$$

- 4 Compute the term  $sqrt\_sum$

$$sqrt\_sum = \sum_{j=0}^7 \sqrt{N\_horz[j]} \quad (6.6)$$

- 5 Compute the term  $corr\_term$ , which is the DC correction term

$$corr\_term = \frac{check\_sum}{sqrt\_sum} \quad (6.7)$$

- 6 The reconstructed pixel block after DC correction is defined by the following expression

$$f[j][i] = f[j][i] + mean - \frac{corr\_term}{\sqrt{N\_vert[i]}} \quad (6.8)$$

From the above steps it is clear that a  $\Delta$ DC-SA-IDCT extension to the current SA-IDCT architecture proposed in Chapter 4 is a post-processing module. The only other modification is  $mean = \frac{F[0][0]}{8}$ , which can be achieved with a simple shift by three binary digit places to the right ( $8 = 2^3$ ). Some ideas that could be explored in future work include:

- The value  $check\_sum$  can be computed iteratively as the SA-IDCT core produces reconstructed pixels
- All 16  $\sqrt{N\_horz[j]}$  and  $\sqrt{N\_vert[i]}$  values can be computed during the initial alpha parsing using a look-up table (LUT), since there are only 8 possible values ( $0 \leq N \leq 8$ )
- A divider architecture such as an SRT divider could be used to compute  $corr\_term$
- The reconstructed pixel data produced by the SA-IDCT core can be stored in a local memory, and when complete this data can be read and step 6 above can be done on the fly as each data value is read from the local memory



### 6.2.3 SA-IDCT Power Efficiency Enhancements

In Chapter 4, work by McMillan et al. is described which leverages the fact that the  $8 \times 8$  IDCT can be expressed as a forward-mapping formulation, i.e. each input element's contribution to the entire set of output elements is expressed independently [228]. Since the quantisation process causes many of the 64 coefficients processed by an IDCT block to be zero valued, the forward mapping formulation can be exploited to skip arithmetic operations on zero valued data. Since the SA-IDCT cannot be expressed in 2D form it is difficult to envisage a forward-mapping formulation. However, there are still ways to manipulate the high probability that there will be a lot of zero valued data fed to the datapath for the horizontal inverse transformation. Two ideas are proposed: null row checking and the use of guarded evaluation adders. Hsu et al. propose variations on these ideas appropriate for their specific SA-IDCT architecture [235]. The current implementation of the proposed SA-IDCT architecture in this thesis does not incorporate zero skipping, and the normalised metrics used in Chapter 4 show that it performs better compared to the Hsu architecture in non-skip mode. It is expected that in future if the following zero skipping schemes are integrated into the proposed design, it will improve upon the zero skipping mode of the Hsu architecture since the savings achievable with zero skipping are purely data dependent.

#### 6.2.3.1 Null Row Checking

The null row checking idea is similar to that proposed by Fanucci et al. for their data driven IDCT architecture [183] (see Section 4.2.1.2). If coarse quantisation causes an entire  $N$ -point row of SA-DCT coefficients to be rounded to zero, an  $N$ -point inverse 1D transform on this data is guaranteed to yield an  $N$ -point vector of zeros. If the ACL in Figure 4.6 detects such a null row, the variable  $N$ -point 1D IDCT datapath can be bypassed, and the corresponding row in the TRAM can be reset immediately to all-zeros (to simulate the computation of the all-zero output vector that did not require explicit computation). Since the detection of null rows for all rows in the input coefficient block is unlikely (usually there is at least a DC coefficient), there is no real need for a “null column” check since a DC coefficient guarantees one non-zero data value in each of the intermediate column vectors stored in the TRAM.

#### 6.2.3.2 Guarded Evaluation Adders

It is also likely that even if a data row/column is not a null vector, it may have some (but not all) zero values. Hence the data entering the adders in the MWGM may have one or more of the addends equal to zero. In such circumstances the adder can be bypassed. Bypassing the adder avoids needless carry propagation and power consumption. To achieve this behaviour in hardware, regular adders are replaced by guarded evaluation adders as shown in Figure 6.1. Such a scheme requires two additional latches, two MUXes and two comparators as is clear from the circuit diagram. The circuit uses the comparators to detect if one or both of the addends are zero. If an addend is zero, its latch is closed and the zero does not propagate. The MUXes then route the non-zero addend to the sum output port (or a zero if both addends are zero). In the case where both addends are non-zero, both latches are transparent and the sum is computed as normal.

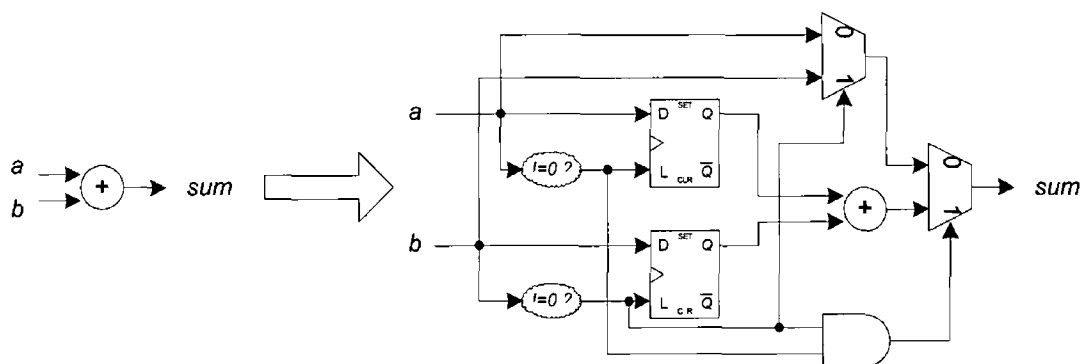


Figure 6.1 Regular Adder and Guarded Evaluation Adder

## 6.2.4 CMM Optimisation Algorithm Improvements

The results presented in Section 5.4.6 show that the EDA CMM optimisation algorithm presented by the author in this thesis achieves improved results compared to other state of the art algorithms in this domain (most notably the Dempster and Macleod algorithm [241] and the Boullis and Tisserand algorithm [132, 242]). Even so, the proposed algorithm is still very computationally demanding. This section briefly outlines some ideas for possible algorithm variations, some ideas for improving the execution time of the current algorithm and also future experiments worth investigating.

### 6.2.4.1 Algorithm Extensions and Variations

**Fitness Function** The most obvious step is to extend the fitness function used by the CMML genetic algorithm. At present, the fitness of a candidate solution is based solely on the number of adder resources it requires. A lower adder count means less circuit area and less power consumption since there is less switching because fewer addition operations are being carried out. However, this is an overly simplistic view. If power savings are the ultimate goal, a more accurate fitness metric should consider factors like fanout, layout and path balancing. Future research work could investigate how such factors could be incorporated into the current chromosomal model.

**Pattern Search Strategy** The current algorithm uses the P1D sub-expression search strategy (i.e. horizontal patterns – see Section 5.3.1.2 for more details). Future research could investigate whether using the P2D strategy (i.e. diagonal patterns – see Section 5.3.1.2 for more details) would improve the results. From a hardware perspective though, it is likely that there exists an upper bound on the number of rows apart within the  $b_{i,j,k}$  slice between which useful sub-expressions will be found. This is because if sub-expression addends come from rows far apart in  $b_{i,j,k}$ , the adders inferred have a large bitwidth.

**Divide and Conquer Variations** Another avenue worth investigating is discussed in step 4 of the DPL algorithm (see Section 5.4.2). The DPL algorithm builds parallel solutions (i.e. different sub-expression choices) based on the permutations possible in the horizontal rows of the  $b_{i,j,k}$  slices. For a given row pattern in  $b_{i,j,k}$ , the number of sub-expression permutations is equivalent to the combinatorial problem of leaf-labelled complete rooted binary trees [261]. As the length  $N$  of the rows in  $b_{i,j,k}$  increases (corresponding to a larger CMM problem) the number of sub-expression permutations increases. Appendix A

outlines the general series, and the number of possibilities increases very quickly for  $N > 4$ . In terms of the algorithm, this means that the bit-string chromosomes become very long for large  $N$ . Since the algorithm bottleneck is attributable to bit counting and bitwise OR of these strings, long bit-strings will slow down the algorithm. However, consider that if  $N$  is very large, sub-expressions formed by addends far apart in the data vector may result in complex wiring and poor layout (see Section 5.3.1.3). As such, a divide and conquer approach to a large CMM problem using smaller independent CMM problems makes sense from a hardware perspective. This could be done by dividing each  $N$ -point row in the CMM matrix into  $N/r$  smaller  $r$  point chunks (see Figure 5.12). This subdivision means that excessively long chromosomal bit-strings are not needed – even for large  $N$  CMMs. An obvious research task would be to investigate the *optimal* subdivision factor  $r$ . A large  $r$  may possibly reduce the adder resources required since there is greater scope for sub-expression sharing. However, large  $r$  means poor layout and a slower algorithm. On the other hand, a small  $r$  speeds up the algorithm, gives a better layout, but there is less scope for sub-expression sharing.

**Pipelined Algorithm** Given that the proposed algorithm is extremely computationally demanding, it makes sense to target a parallel processing platform (if the resources were available). On an ideal parallel platform it would be possible to execute parallel instances of the DPL algorithm for each of the dot products in the CMM being optimised. When each DPL instance reaches certain percentage increments of their respective search spaces, the intermediate results could be forwarded to the CMML algorithm. As the solutions found by the CMML algorithm improve across the generations, this information could be fed back to the DPL instances to reduce the scope of the solutions they produce. Optimality is not sacrificed since the DPL instances are merely restricted from generating solutions that are guaranteed not to contribute to overall solutions that are better than the dynamic results maintained by the CMML stage. Such a pipelined algorithm has not been investigated further, and is targeted for future work.

#### 6.2.4.2 Run-Time Acceleration

The problem modelling used by the author and in particular the chromosomal bit-string representation of the fundamental data structures being manipulated mean that the most demanding sub-tasks of the algorithm are amenable to hardware acceleration. Profiling has shown that the dominant tasks (approximately 60%) are the counting of set bits in the bit-strings and also the bitwise OR operation used when combining bit-strings. These tasks are ideally suited for hardware implementation as they are very regular operations. The most likely way of implementing these as hardware accelerators is on an FPGA prototyping chip which can be addressed by the central CPU. To this end, it makes sense to leverage the prototyping platform used for the SA-DCT/IDCT conformance testing, i.e. the WildCard-II PCM-CIA FPGA platform [218]. The main CMM algorithm could run on the host laptop (e.g. Pentium-4M processor) and offload the bit counting and bitwise OR operations to highly parallel hardware accelerators residing on the WildCard-II. An efficient protocol is required to buffer the appropriate data down to the WildCard-II for fast processing and write-back of results to the main system memory. Clearly the protocol latency overhead between the WildCard-II and the host memory should not negate the potential gains achievable by the hardware accelerated tasks.

### 6 2 4 3 Future Experiments

Section 5 5 describes two options for configuring the current DPL and CMML algorithms to optimise an MCM problem such as the SA-DCT/IDCT. A worthwhile research experiment would be to try both methods and compare the results according to the algorithms (adder count) but also implement the architectures inferred by both in hardware and benchmark the synthesis results. The first approach can be run exhaustively since it only uses the DPL algorithm. The second approach is much more computationally demanding since it uses the CMML algorithm, and the results achieved depend upon the performance of the genetic algorithm.

### 6 2 5 Hardware Accelerator Adaptability

A key theme of this thesis is that shortened battery life due to the excessive power consumption required by video processing is a key challenge facing hardware designers of next generation mobile multimedia platforms. In this spiral of feature enhancement and device convergence, users expect extended battery life on each new device generation. The envisaged applications all leverage different video processing functionalities (in fact, often combinations thereof). For example, multimedia messaging requires image/video compression so that the image data can be efficiently represented for both storage on the device and transmission over mobile networks. Video blogging requires this as well but also indexing and annotation of the content so that it can be subsequently organised for effective browsing or searching. Video gaming requires mixed 3-D synthetic and natural video processing. Security applications require functionalities such as encryption, motion detection, object segmentation (i.e. extracting the real world objects from the scene) and face detection and tracking. As stated in Chapter 1 – these functionalities are not technically orthogonal. They can be considered to be built using a set of lower-level basic video processing blocks that may be termed *basic enabling technologies*. For example, a key component of a video compression algorithm is a process known as motion estimation, which is implemented in practice by matching blocks of pixels. The same block matching process, albeit configured in a different way, can be used for feature extraction for indexing or for crude depth estimation (when two cameras are available).

Given the power consumption issues with the emerging convergent devices what is required is a suite of power efficient hardware implementations of these enabling technologies that are adaptable and that can be configured either in isolation or in combination in order to implement a range of different video processing functionalities for a range of different devices. This thesis proposes energy-efficient architectures for two enabling technologies – the SA-DCT in Chapter 3 and SA-IDCT in Chapter 4. Chapter 5 proposes an algorithm that can be leveraged to design a broader range of enabling technologies – those that may be classified more generally as a CMM, which the SA-DCT/IDCT are specific instances. Given the range of applications constantly being dreamt up, a key concern is hardware flexibility – the ability to be able to adapt the same multimedia platform to many different applications thereby efficiently re-using the same basic architecture. Clearly the SA-DCT/IDCT play a vital role in applications requiring video compression. However, the author believes that the SA-DCT can also be leveraged to help with another difficult video processing task – that of video object segmentation. If proven to be viable, this is interesting since robust video object segmentation is required as a pre-processing stage to MPEG-4 object-based compression and both tasks could re-use the same SA-DCT architecture.



Figure 6.2: Sample Images Captured by the Philips FluidFocus Lens

The author's idea is to use a depth from defocus approach to video object segmentation that leverages the SA-DCT to perform the analysis. More generally depth from focus/defocus attempts to solve the problem of estimating the 3D surface of a scene from a set of two or more images of that scene [271, 272]. The images are taken from the same point of view and only differ by changing the camera parameters (typically the focal setting). Essentially the more defocused an image becomes the more and more attenuated the high spatial frequencies become. If there are multiple objects in the scene at different depths – each object will be in focus at different discrete focal settings while the other objects will be blurred. This fact can be exploited to segment semantic objects in a scene. Depth from defocus is typically used for tasks like depth map estimation, 3D scene reconstruction, range sensing, surveillance etc., but the author proposes to use it for the purpose of object segmentation.

Employing defocus-based segmentation on a mobile platform may be problematic since it requires a camera module capable of fine-grained focal length adjustment, and a bulky mechanical lens is not desirable. However Philips Laboratories have recently devised a non-mechanical “FluidFocus” lens for use in camera phones [273]. The device uses electrostatic forces to alter the shape of a drop of slightly salty water inside a glass cylinder 3mm in diameter and 2.2 mm long. One end of the cylinder points toward the image plane; the other is directed at the subject being imaged. By changing the voltage on the electrode of the liquid lens, the camera is able to focus on objects at distances anywhere from 2cm up to infinity. The voltage adjusts the focal length between 2.85mm and 3.55mm. The size and speed of the lens make it ideal for use in a mobile camera phone style device. A set of sample images captured by the FluidFocus lens are shown in Figure 6.2 [273]. With technology like the FluidFocus lens emerging, it is certainly worth researching further defocus-based semantic object segmentation as a potential low-cost pre-processing step to object-based video compression on a mobile platform.

The author proposes in future work to investigate the potential of leveraging depth from defocus for semantic object segmentation. Since the amount of object blurring is akin to its spatial frequency component values (focused objects have higher frequencies), the SA-DCT could possibly be used to analyse the amount of blurring in  $N \times N$  image blocks ( $N \leq 8$ ). This promotes hardware re-use since the SA-DCT accelerator proposed in Chapter 3 may be used to aid the segmentation process as well as its conventional use in an MPEG-4 encoder. As an illustration, consider the images in Figure 6.3 and Figure 6.4. In Figure 6.3 the focal length of the camera is short so the near objects are in focus. The opposite is true in Figure 6.4 where the focal length is longer.

The first step is to compute the block-wise  $8 \times 8$  DCT on both the near focused and far focused images. This provides two sources of frequency information for each  $8 \times 8$  block in the scene. Blocks

with high frequency information in the near focused image are labelled as part of the foreground object. Conversely, blocks with relatively higher frequency information in the far focused image are labelled as part of the background object. The adaptive nature of the SA-DCT could be exploited to compute a more finely grained segmentation mask.  $8 \times 8$  blocks that are on the object boundary could be sub-divided into smaller variable  $N \times N$  sized blocks and each of these could be classified independently. Some very preliminary illustrative results are shown in Figure 6.5, where white regions have been classified as foreground while black regions have been classified as background using this technique. With some morphological post-processing, noise can be eliminated resulting in the mask shown in Figure 6.6. The final segmented foreground object is shown in Figure 6.7.

Although the segmentation mask in Figure 6.7 has some noise, it is achieved with relatively low complexity given that the dominant computation is the SA-DCT and a hardware accelerator for the SA-DCT (as proposed in Chapter 3) may already be present on a mobile device capable of MPEG-4 encoding. If a more accurate mask is required, the noisy mask produced by the SA-DCT technique could be used as an input seed to a more sophisticated region-based segmentation algorithm. It is mentioned in Chapter 2 that if the application is constrained, i.e. there is some prior knowledge about the object being segmented, this can be used to the advantage of a segmentation algorithm. For example, in the video telephony example discussed in Chapter 2, a human face is the target object. Knowledge of the geometry of the human face could also be used to clean up a noisy segmentation mask generated by the defocus based segmentation. Future work in this area will require the collection of a large image data set as captured by a lens configuration similar to the FluidFocus for testing purposes.

It is clear that in an era where multimedia applications are converging on the mobile platform and short battery life is a concern, hardware flexibility and re-use is a primary goal. Given that technology like the Philips FluidFocus lens is emerging, research into using the SA-DCT for robust object segmentation as well as video compression is an exciting future path to build upon the work presented in this thesis.



Figure 6.3: Near Focused Image



Figure 6.4: Far Focused Image.

### 6.3 A Vision for the Future

It is clear that to satisfactorily support all of the emerging multimedia applications on mobile platforms, the research community must focus on developing low power and low energy implementations of the

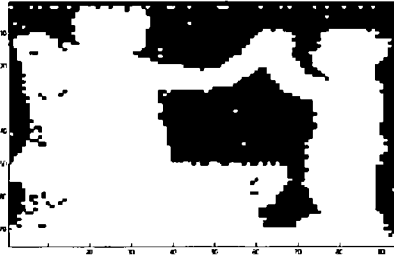


Figure 6 5 Rough Foreground Mask

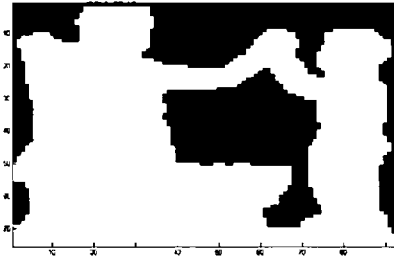


Figure 6 6 Post-Processed Foreground Mask

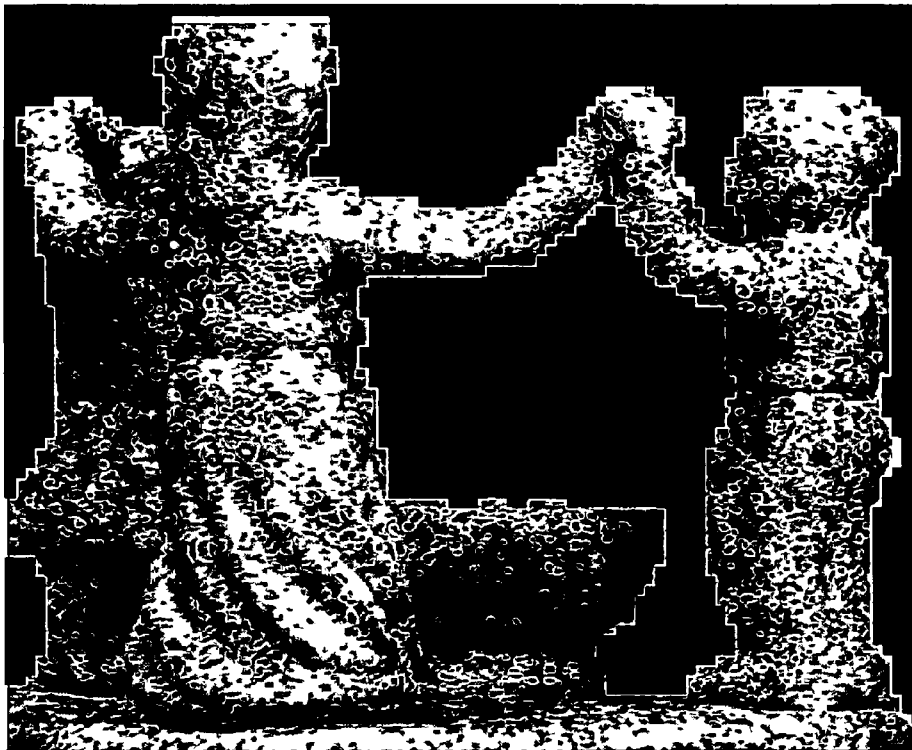


Figure 6 7 Segmented Foreground Object

basic enabling technologies (e.g. SA-DCT/IDCT) that underpin all of these applications. This is because in the highly integrated deep sub-micron era, power consumption and short battery life are emerging as a primary design constraints alongside area and speed. These implementations should also be available in a flexible and adaptable manner to allow various applications to be delivered on the same platform. Although low power is now a prime product differentiator, there is no common platform or forum that enables accurate benchmarking of competing implementations of the basic enabling technologies. Normalisation formulae are presented in this thesis to facilitate this, but what is required is a robust industry standard for benchmarking power and energy properties much like the Berkeley Design Technology Inc (BDTI) standard benchmarks for DSP processor speed performance.

The rapid evolution of multimedia application convergence has caused an increase in time-to-market pressure in industry. This is to guarantee that companies stay at the leading edge in a very competitive market. From a design perspective, such convergence has led to increased design complexity. The fact that there is less time for implementation has driven the need for more abstract “push-button” design flows. This is referred to as the Electronic System Level (ESL) design paradigm. Concerns about the quality of the results generated by these automatic tools compared to hand-crafted solutions has thus far limited the widespread adoption of ESL design. As such, there is an opportunity for the research community to develop robust sophisticated EDA algorithms that tackle optimisation problems that are too complex to be addressed manually.

Whilst both are difficult issues to address and will provide the hardware research community with challenges for many years to come, it is clear that the research presented in this thesis that key advances in crucial aspects of these challenges are now possible.



# APPENDIX A

## Leaf-Labelled Complete Rooted Binary Trees

### A 1 Introduction

The CMM optimisation algorithm proposed in this thesis is described in detail in Chapter 5. A non-trivial step in the algorithm is step 4 of the DPL stage “Build Permutation SOP” (see Section 5.4.2). Step 4 of the DPL algorithm analyses the bit patterns in the rows of the residual matrix of a 2D slice. For example, Equation 5.19 is the residual matrix for the 2D slice shown in Equation 5.16 (see Section 5.4.2 for details). Step 4 of the DPL algorithm considers each residual row in turn. The bit pattern on a particular row implies a multi-operand addition of certain elements of the data vector  $\mathbf{x}$  (in a CMM  $\mathbf{y} = \mathbf{A}\mathbf{x}$ ). For example  $0\bar{1}01$  implies  $-x_1 + x_3$  or  $1101$  implies  $x_0 + x_1 + x_3$ . A general multi-operand addition may be implemented in hardware using various two-input adder sub-expression topologies, and each topology can have different operand ordering. The proposed optimisation algorithm analyses the bit patterns and decodes all possible ways of implementing the multi-operand addition implied. This is represented internally by the algorithm as a Sum of Products (SOP), where each term in the SOP represents a unique choice of two-input adder sub-expressions. Deriving these options is equivalent to the combinatorial problem of leaf-labelled complete rooted binary trees [261]. This appendix discusses this general problem in detail to give background to the algorithm described in Chapter 5.

### A 2 The Problem

Figure A.1 shows a black-box system diagram of an  $N$ -input multi-operand addition. Based on the  $N$ -bit vector  $s[0 : N - 1]$  ( $S$  corresponds to a row of a residual matrix), the single output  $F$  is computed according to equation A.1. Note that additions and subtractions are possible and for the purposes of this discussion are considered equivalent in terms of hardware cost.

$$F = \sum_{n=0}^{N-1} s[n] * x_n \quad s[n] \in \{-1, 0, 1\} \quad (\text{A } 1)$$

There are three problems to be solved

1. Given that there are  $N$  inputs that have three possible select values  $(\bar{1}, 0, 1)$  ( $-1 \equiv \bar{1}$ ), how many

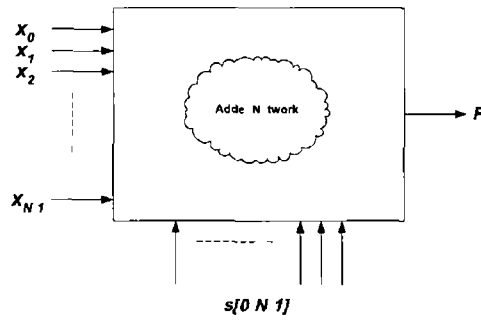


Figure A 1 Multi-Operand Addition

possible select patterns are there? It is not simply  $3^N$  due to certain properties as outlined subsequently

- 2 For each select pattern found by Problem 1 – how many possible adder topologies are there?
- 3 For each topology found by Problem 2 – how many addend arrangement permutations are possible? (being careful to avoid redundant duplicates due to commutative equivalences)

### A 2 1 Problem 1 – Valid Patterns

The summation result  $F$  is a multi-operand addition based on the radix-2 SD  $N$ -bit select vector  $s$  so there are maximally  $3^N$  possible patterns. However, due to the nature of the operation the following cases require special attention:

- 1 If  $s$  is all zeros no addition takes place since  $F = 0$ . Therefore this case must be discounted from the set of possible patterns.
- 2 If  $s$  only has a single non-zero digit set then no addition takes place since  $F = s[n] * x_n$ . Since  $s$  is an  $N$ -bit vector and each of its bits (if set) can take on either 1 or -1, then a further  $2N$  cases can be discounted from the set of possible patterns because they do not require any adder (since all other bits are zero).
- 3 Consider the remaining cases where at least 2 bits are set in the vector  $s$ . This subset contains  $3^N - 1 - 2N$  elements. However, half of these patterns are the “inverse” of the other (assuming  $inverse(0) = 0, inverse(1) = -1, inverse(-1) = 1$ ). Consider as a simple example  $N = 4$  and the two inverse pattern select pairs  $A = \bar{1}100$  and  $B = 1\bar{1}00$ . In this case  $F_A = -x_0 + x_1$  and  $F_B = x_0 - x_1, i.e. F_A = -F_B$ . For the purposes of this work cases  $A$  and  $B$  are considered as “inverse equivalent” since a simple two’s complementer circuit on the output of the circuit with a MUX can switch between either and the same adder can be used to carry out both operations  $F_A$  and  $F_B$ .

As a result of the three cases above the number of valid unique patterns reduces to equation A 2

$$R_N = \frac{3^N - 1 - 2N}{2} \quad (A 2)$$

### A 2 2 Problem 2 – Topologies

Given that there are  $R_N$  possible patterns, how many topologies are there for each pattern? The number of topologies possible for each pattern depends on the number of bits set in the select vector  $s$ . Fig A 2 shows the various topology options for the case of 2, 3, 4 and 5 addends. A formula is required for working out the number of possible topologies  $T_n$  for a general number of addends  $n$  (i.e. when there are  $n$  non-zero bits in the select vector  $s$ ). It is based on the associative nature of addition and permutations based on taking two at a time (possibly in parallel also if there is no overlap between selected pairs).

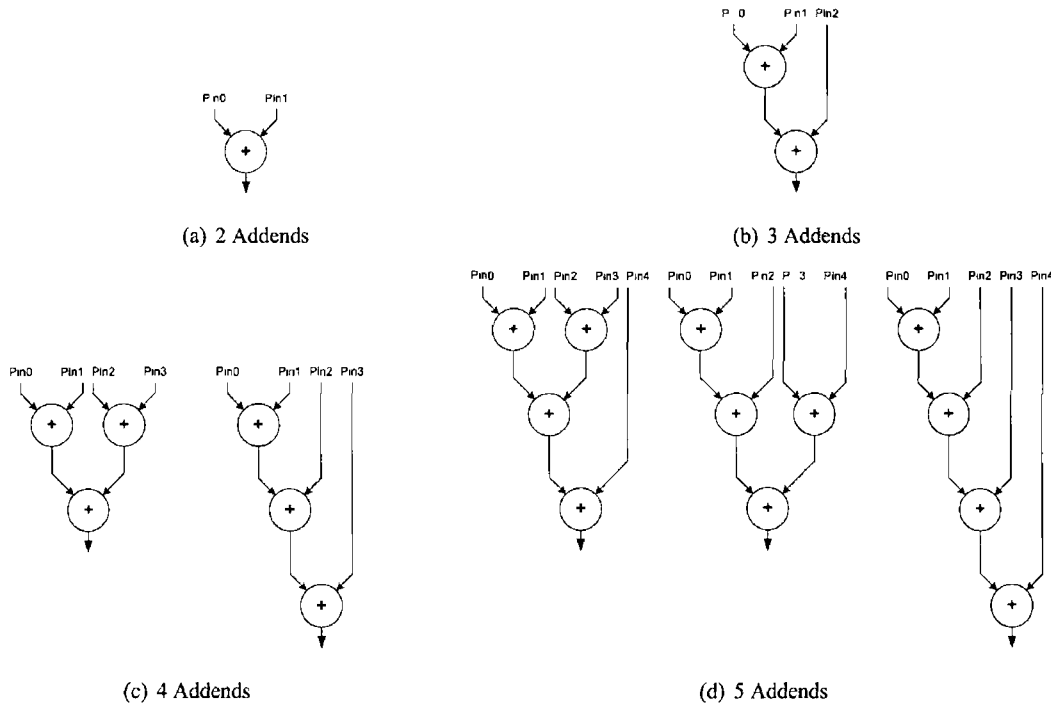


Figure A 2 Adder Topology Options for Different Number of Addends

### A 2 3 Problem 3 – Addend Permutations

For a given topology – how many ways are there to arrange the addends to the input pins of the network avoiding commutative equivalences ( $A + B \equiv B + A$ )? The number of permutations is based on the associative property of addition i.e.  $[(A + B) + C]$  versus  $[A + (B + C)]$ . From a hardware perspective, the grouping of addends (mapping of addends to input pins) is important if there are multiple addend networks where there exists the potential to share common sub-expressions and reduce the number of hardware adders necessary. How many different ways are there to arrange the addends such that the associative nature is different? Two associative options are equivalent if the summations at the output of each of the adders are equivalent taking into account the commutative property of addition (an example follows in the next paragraph).

Consider for example the 4 addend case ( $n = 4$ ) which has two unique topologies as shown in Fig A 2 and that  $N = 4, s = 1111$ . The unique associative permutations for the first topology are shown in Table A 1 and those for the second topology are shown in Table A 2. There are 15 unique

Table A 1 Topology 1 Options

P <sub>in0</sub>	P <sub>in1</sub>	P <sub>in2</sub>	P <sub>in3</sub>
$x_0$	$x_1$	$x_2$	$x_3$
$x_0$	$x_2$	$x_1$	$x_3$
$x_0$	$x_3$	$x_1$	$x_2$

Table A 2 Topology 2 Options

P <sub>in0</sub>	P <sub>in1</sub>	P <sub>in2</sub>	P <sub>in3</sub>
$x_0$	$x_1$	$x_2$	$x_3$
$x_0$	$x_1$	$x_3$	$x_2$
$x_0$	$x_2$	$x_1$	$x_3$
$x_0$	$x_2$	$x_3$	$x_1$
$x_0$	$x_3$	$x_1$	$x_2$
$x_0$	$x_3$	$x_2$	$x_1$
$x_1$	$x_2$	$x_0$	$x_3$
$x_1$	$x_2$	$x_3$	$x_0$
$x_1$	$x_3$	$x_0$	$x_2$
$x_1$	$x_3$	$x_2$	$x_0$
$x_2$	$x_3$	$x_0$	$x_1$
$x_2$	$x_3$	$x_1$	$x_0$

permutations in total and the permutations that are not shown in the tables are not included because they are commutatively equivalent to one of the permutations in the table. Table A 3 shows the permutations for various 4 bit patterns. For example consider Table A 2 and why  $P_{in0} = x_1, P_{in1} = x_0, P_{in2} = x_2, P_{in3} = x_3$  is not in this table. It is not included because it is equivalent to  $P_{in0} = x_0, P_{in1} = x_1, P_{in2} = x_2, P_{in3} = x_3$  (first row in Table A 2) since the outputs of each adder in the topology are equal. However  $P_{in0} = x_0, P_{in1} = x_1, P_{in2} = x_2, P_{in3} = x_3$  and  $P_{in0} = x_0, P_{in1} = x_1, P_{in2} = x_3, P_{in3} = x_2$  are distinct and both included in Table A 2 since the outputs of each adder in the topology are not equal. These distinctions are illustrated graphically in Fig A 3.

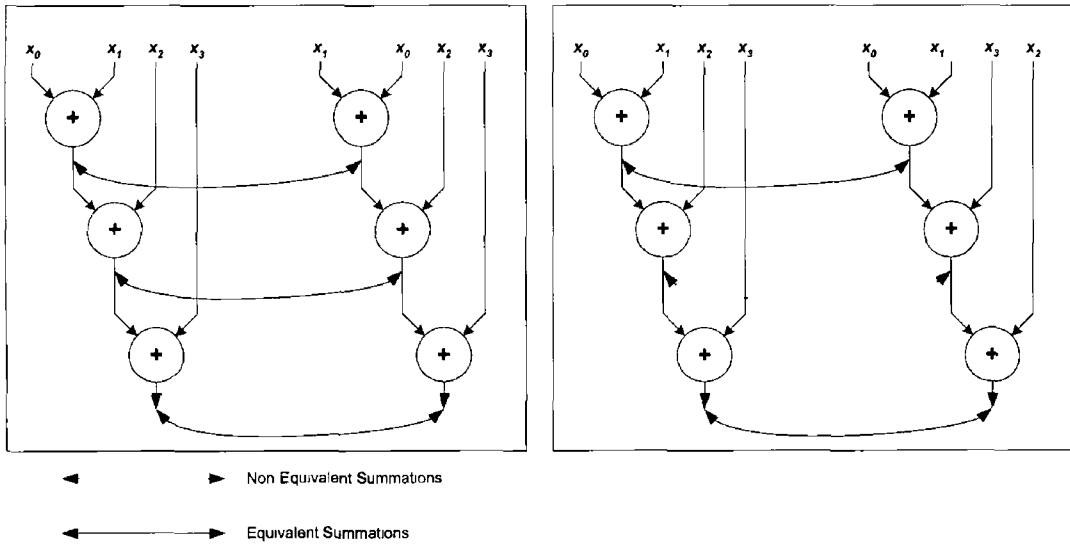


Figure A 3 Hardware Associativity and Commutativity Examples

Table A 3 Unique Adder Permutations for  $N = 4$

<i>prod_vector</i> Bit	Row Permutation	Pattern	1's Comp	Additions
0	0	1100	$\overline{1100}$	$\pm(x_0 + x_1) = \pm A$
1	1	1010	$\overline{1010}$	$\pm(x_0 + x_2) = \pm B$
2	2	1001	$\overline{1001}$	$\pm(x_0 + x_3) = \pm C$
3	3	0110	$0\overline{110}$	$\pm(x_1 + x_2) = \pm D$
4	4	0101	$0\overline{101}$	$\pm(x_1 + x_3) = \pm E$
5	5	0011	$00\overline{11}$	$\pm(x_2 + x_3) = \pm F$
6	6	$1\overline{100}$	$\overline{1100}$	$\pm(x_0 - x_1) = \pm U$
7	7	$10\overline{10}$	$\overline{1010}$	$\pm(x_0 - x_2) = \pm V$
8	8	$100\overline{1}$	$\overline{1001}$	$\pm(x_0 - x_3) = \pm W$
9	9	$01\overline{10}$	$0\overline{110}$	$\pm(x_1 - x_2) = \pm X$
10	10	$010\overline{1}$	$0\overline{101}$	$\pm(x_1 - x_3) = \pm Y$
11	11	$001\overline{1}$	$00\overline{11}$	$\pm(x_2 - x_3) = \pm Z$
12	12	1110	$\overline{1110}$	$\pm(A + x_2) = \pm Ap_2$
13				$\pm(B + x_1) = \pm Bp_1$
14				$\pm(D + x_0) = \pm Dp_0$
15	13	$11\overline{10}$	$\overline{1110}$	$\pm(A - x_2) = \pm Am_2$
16				$\pm(V + x_1) = \pm Vp_1$
17				$\pm(X + x_0) = \pm Xp_0$
18	14	$1\overline{110}$	$\overline{1110}$	$\pm(U + x_2) = \pm Up_2$
19				$\pm(B - x_1) = \pm Bm_1$
20				$\pm(-X + x_0) = \pm Xm_0$
21	15	$1\overline{110}$	$\overline{1110}$	$\pm(U - x_2) = \pm Um_2$
22				$\pm(V - x_1) = \pm Vm_1$
23				$\pm(-D + x_0) = \pm Dm_0$
24	16	1101	$\overline{1101}$	$\pm(A + x_3) = \pm Ap_3$
25				$\pm(C + x_1) = \pm Cp_1$
26				$\pm(E + x_0) = \pm Ep_0$
27	17	$110\overline{1}$	$\overline{1101}$	$\pm(A - x_3) = \pm Am_3$
28				$\pm(W + x_1) = \pm Wp_1$
29				$\pm(Y + x_0) = \pm Yp_0$
30	18	$1\overline{101}$	$\overline{1101}$	$\pm(U + x_3) = \pm Up_3$
31				$\pm(C - x_1) = \pm Cm_1$
32				$\pm(-Y + x_0) = \pm Ym_0$
33	19	$1\overline{101}$	$\overline{1101}$	$\pm(U - x_3) = \pm Um_3$
34				$\pm(W - x_1) = \pm Wm_1$
35				$\pm(-E + x_0) = \pm Em_0$
36				$\pm(B + x_3) = \pm Bp_3$
Overleaf	20	1011	$\overline{1011}$	

Table A 3 – Continued

<i>prod_vector</i> Bit	Row Permutation	Pattern	1's Comp	Additions
37				$\pm(C + x_2) = \pm Cp_2$
38				$\pm(F + x_0) = \pm Fp_0$
39				$\pm(B - x_3) = \pm Bm_3$
40	21	101 $\bar{1}$	$\bar{1}0\bar{1}1$	$\pm(W + x_2) = \pm Wp_2$
41				$\pm(Z + x_0) = \pm Zp_0$
42				$\pm(V + x_3) = \pm Vp_3$
43	22	10 $\bar{1}1$	$\bar{1}0\bar{1}\bar{1}$	$\pm(C - x_2) = \pm Cm_2$
44				$\pm(-Z + x_0) = \pm Zm_0$
45				$\pm(V - x_3) = \pm Vm_3$
46	23	10 $\bar{1}\bar{1}$	$\bar{1}011$	$\pm(W - x_2) = \pm Wm_2$
47				$\pm(-F + x_0) = \pm Fm_0$
48				$\pm(D + x_3) = \pm Dp_3$
49	24	0111	0 $\bar{1}\bar{1}\bar{1}$	$\pm(E + x_2) = \pm Ep_2$
50				$\pm(F + x_1) = \pm Fp_1$
51				$\pm(D - x_3) = \pm Dm_3$
52	25	011 $\bar{1}$	0 $\bar{1}\bar{1}1$	$\pm(Y + x_2) = \pm Yp_2$
53				$\pm(Z + x_1) = \pm Zp_1$
54				$\pm(X + x_3) = \pm Xp_3$
55	26	01 $\bar{1}1$	0 $\bar{1}1\bar{1}$	$\pm(E - x_2) = \pm Em_2$
56				$\pm(-Z + x_1) = \pm Zm_1$
57				$\pm(X - x_3) = \pm Xm_3$
58	27	01 $\bar{1}\bar{1}$	0 $\bar{1}11$	$\pm(Y - x_2) = \pm Ym_2$
59				$\pm(-F + x_1) = \pm Fm_1$
60				$\pm(A + F)$
61	28	1111	$\bar{1}\bar{1}\bar{1}\bar{1}$	$\pm(Ap_2 + x_3)$
62				$\pm(Ap_3 + x_2)$
63				$\pm(Fp_0 + x_1)$
64				$\pm(Fp_1 + x_0)$
65				$\pm(B + E)$
66				$\pm(Bp_1 + x_3)$
67				$\pm(Bp_3 + x_1)$
68				$\pm(Ep_0 + x_2)$
69				$\pm(Ep_2 + x_0)$
70				$\pm(C + D)$
71				$\pm(Cp_1 + x_2)$
72				$\pm(Cp_2 + x_1)$
73				$\pm(Dp_0 + x_3)$
74				$\pm(Dp_3 + x_0)$
75				$\pm(A + Z)$

Overleaf

Table A 3 – Continued

<i>prod_vector</i> Bit	Row Permutation	Pattern	1's Comp	Additions			
76				$\pm(Ap_2 - x_3)$			
77				$\pm(Am_3 + x_2)$			
78				$\pm(Zp_0 + x_1)$			
79				$\pm(Zp_1 + x_0)$			
80				$\pm(B + Y)$			
81				$\pm(Bp_1 - x_3)$			
82				$\pm(Bm_3 + x_1)$			
83				$\pm(Yp_0 + x_2)$			
84				$\pm(Yp_2 + x_0)$			
85				$\pm(W + D)$			
86				$\pm(Wp_1 + x_2)$			
87				$\pm(Wp_2 + x_1)$			
88				$\pm(Dp_0 - x_3)$			
89				$\pm(Dm_3 + x_0)$			
90				30	$11\bar{1}1$	$\bar{1}\bar{1}1\bar{1}$	$\pm(A - Z)$
91							$\pm(Ap_2 + x_3)$
92							$\pm(Am_3 - x_2)$
93							$\pm(-Zm_0 + x_1)$
94							$\pm(-Zm_1 + x_0)$
95	$\pm(V + E)$						
96	$\pm(Vp_1 + x_3)$						
97	$\pm(Vp_3 + x_1)$						
98	$\pm(Ep_0 - x_2)$						
99	$\pm(Em_2 + x_0)$						
100	$\pm(C + X)$						
101	$\pm(Cp_1 - x_2)$						
102	$\pm(Cm_2 + x_1)$						
103	$\pm(Xp_0 + x_3)$						
104	$\pm(Xp_3 + x_0)$						
105				$\pm(A - F)$			
106				$\pm(Am_2 - x_3)$			
107				$\pm(Am_3 - x_2)$			
108				$\pm(-Fm_0 + x_1)$			
109				$\pm(-Fm_1 + x_0)$			
110				$\pm(V + Y)$			
111				$\pm(Vp_1 - x_3)$			
112				31	$11\bar{1}\bar{1}$	$\bar{1}\bar{1}11$	$\pm(Vm_3 + x_1)$
113							$\pm(Yp_0 - x_2)$
114							$\pm(Ym_2 + x_0)$

Overleaf

Table A 3 – Continued

<i>prod_vector</i> Bit	Row Permutation	Pattern	1's Comp	Additions
115				$\pm(W + X)$
116				$\pm(Wp_1 - x_2)$
117				$\pm(Wm_2 + x_1)$
118				$\pm(Xp_0 - x_3)$
119				$\pm(Xm_3 + x_0)$
120	32	$1\bar{1}\bar{1}1$	$\bar{1}\bar{1}\bar{1}\bar{1}$	$\pm(U + F)$
121				$\pm(Up_2 + x_3)$
122				$\pm(Up_3 + x_2)$
123				$\pm(Fp_0 - x_1)$
124				$\pm(Fm_1 + x_0)$
125				$\pm(B - Y)$
126				$\pm(Bm_1 + x_3)$
127				$\pm(Bp_1 - x_1)$
128				$\pm(-Ym_0 + x_2)$
129				$\pm(-Ym_2 + x_0)$
130				$\pm(C - X)$
131				$\pm(Cm_1 + x_2)$
132				$\pm(Cp_2 - x_1)$
133				$\pm(-Xm_0 + x_3)$
134				$\pm(-Xm_3 + x_0)$
135	33	$1\bar{1}\bar{1}\bar{1}$	$\bar{1}\bar{1}\bar{1}1$	$\pm(U + Z)$
136				$\pm(Up_2 - x_3)$
137				$\pm(Um_3 + x_2)$
138				$\pm(Zp_0 - x_1)$
139				$\pm(Zm_1 + x_0)$
140				$\pm(B - E)$
141				$\pm(Bm_1 - x_3)$
142				$\pm(Bm_3 - x_1)$
143				$\pm(-Em_0 + x_2)$
144				$\pm(-Em_2 + x_0)$
145				$\pm(W - X)$
146				$\pm(Wm_1 + x_2)$
147				$\pm(Wp_2 - x_1)$
148				$\pm(-Xm_0 - x_3)$
149				$\pm(-Xp_3 + x_0)$
150				$\pm(U - Z)$
151				$\pm(Um_2 + x_3)$
152				$\pm(Up_3 - x_2)$
153				$\pm(-Zm_0 - x_1)$

Overleaf



Table A 3 – Continued

<i>prod_vector</i> Bit	Row Permutation	Pattern	1's Comp	Additions			
154				$\pm(-Zp_1 + x_0)$			
155				$\pm(V - Y)$			
156				$\pm(Vm_1 + x_3)$			
157				$\pm(Vp_3 - x_1)$			
158				$\pm(-Ym_0 - x_2)$			
159				$\pm(-Yp_2 + x_0)$			
160				$\pm(C - D)$			
161				$\pm(Cm_1 - x_2)$			
162				$\pm(Cm_2 - x_1)$			
163				$\pm(-Dm_0 + x_3)$			
164				$\pm(-Dm_3 + x_0)$			
165				35	$\overline{1111}$	$\overline{1111}$	$\pm(U - F)$
166							$\pm(Um_2 - x_3)$
167							$\pm(Um_3 - x_2)$
168	$\pm(-Fm_0 - x_1)$						
169	$\pm(-Fp_1 + x_0)$						
170	$\pm(V - E)$						
171	$\pm(Vm_1 - x_3)$						
172	$\pm(Vm_3 - x_1)$						
173	$\pm(-Em_0 - x_2)$						
174	$\pm(-Ep_2 + x_0)$						
175	$\pm(W - D)$						
176	$\pm(Wm_1 - x_2)$						
177	$\pm(Wm_2 - x_1)$						
178	$\pm(-Dm_0 - x_3)$						
179	$\pm(-Dp_3 + x_0)$						

### A 3 The Solution

The solution to Problem 1 is straightforward and is expressed by Equation A 2. The solutions to Problems 2 and 3 are more complicated, and the solutions presented here are as a result of a collaborative discussion on the problem with Dr S D Andres (Centre for Applied Information, University of Cologne) [261]. Problem 2 is akin to the problem of setting pairs of brackets between  $n$  identical items, so that each pair of brackets has exactly two entries (which might be items and/or pairs of brackets). Two of these bracket structures are said to be *isomorphic* if they can be obtained from each other by a sequence of commutative operations (not using associative operations). Section A 3.1 determines a recursive formula for the total number of bracket structures (where isomorphic but non-identical structures may be counted several times). This is a solution to Problem 2 of Section A 2.2.

### A 3 1 Number of Non-Isomorphic Binary Bracket Structures of $n$ -Operand Operations

Consider the following problem Given  $n$  variables  $x_1, \dots, x_n$  and an associative binary operation  $(, )$ , how many “essentially different” ways are there to set the brackets when calculating the value of the composition of the  $x_i$  (each once)? To be precise, we do not care about permutations of the variables which lead to the same bracket structure For instance  $((x_1x_2)(x_3x_4))x_5$  and  $(x_2((x_1x_5)(x_4x_3)))$  are regarded as the same bracket structure, whereas  $((x_1x_2)((x_3x_4)x_5))$  is essentially different from or for formally *non-isomorphic* with those

This problem is the same problem as described in Section A 2 2 with regard to the number of possible 2-input adder topologies for an  $n$ -input multi-operand addition A more mathematical way of posing this questions is How many non-isomorphic complete binary rooted trees with  $2n - 1$  vertices exist? A complete binary rooted tree  $(T, r)$  is a connected finite graph  $T = (V, E)$  without cycles with  $r \in V$  where the degree of  $r$  is 2, and the degree of each other vertex is 1 or 3 Obviously, each vertex of degree greater than 1 corresponds to a 2-input adder and respectively to a bracket operation An *isomorphism*  $f$  of rooted trees  $(T_1, r_1), (T_2, r_2)$  is a graph isomorphism with  $f(r_1) = r_2$

**Lemma 1** Let  $A, B$  resp  $C, D$  be bracket structures with  $k_1$  resp  $k_2$  variables and  $k_1 \neq k_2$  ( $A, C$ ) and  $(B, D)$  are isomorphic if and only if  $A$  and  $B$  are isomorphic and  $C$  and  $D$  are isomorphic

**Proof** “ $\Rightarrow$ ” Let  $(A, C)$  and  $(B, D)$  be isomorphic So all substructures must have an isomorphic counterpart Since  $k_1 \neq k_2$  it is impossible that  $A$  and  $D$  (resp  $B$  and  $C$ ) are isomorphic, so  $A$  and  $B$  are isomorphic as well as  $C$  and  $D$

“ $\Leftarrow$ ” The inverse implication trivially holds

□

**Lemma 2** Let  $A, B, C, D$  be bracket structures with  $k$  variables ( $A, C$ ) and  $(B, D)$  are isomorphic if and only if one of the following hold

1  $A$  and  $B$  are isomorphic and  $C$  and  $D$  are isomorphic

2  $A$  and  $B$  are isomorphic and  $C$  and  $B$  are isomorphic

Let  $T_n$  be the number of non-isomorphic complete binary rooted trees with  $n$  vertices (resp , the number of 2-input adder topologies for a  $N$ -input summation, resp , the number of essentially different binary bracket structures) Then we conclude

**Theorem 1**  $T_n$  can be calculated recursively by

$$T_1 = 1 \tag{A 3}$$

$$T_{2n+2} = \sum_{i=1}^n T_i T_{2n+2-i} + \frac{T_{n+1}(T_{n+1} + 1)}{2} \quad \text{for } n \geq 0, \tag{A 4}$$

$$T_{2n+1} = \sum_{i=1}^n T_i T_{2n+1-i} \quad \text{for } n \geq 1 \tag{A 5}$$

Table A 4 The First 32 Values of  $T_n$

$n$	$T_n$
1	1
2	1
3	1
4	2
5	3
6	6
7	11
8	23
9	46
10	98
11	207
12	451
13	983
14	2179
15	4850
16	10905
17	24631
18	56011
19	127912
20	293547
21	676157
22	1563372
23	3626149
24	8436379
25	19680277
26	46026618
27	107890609
28	253450711
29	596572387
30	1406818759
31	3323236238
32	7862958391

**Proof** Clearly,  $T_1 = 1$  since we have no brackets, hence no addition (Equation A 3) If we consider  $T_n$  for odd  $n$ , we have split  $n$  into two unequal parts,  $k_1, k_2$  Consider the combinations  $(A, B)$  or bracket structures  $A$  with  $k_1$  variables and  $B$  with  $k_2$  variables By Lemma 1, two different such structures are non-isomorphic as long as the  $A$ 's are non-isomorphic or the  $B$ 's are non-isomorphic This gives us Equation A 5 Equation A 4 ( $T_n$  for even  $n$ ) uses the same idea whenever  $n$  is split into unequal parts However,  $n = n/2 + n/2$  is also feasible In this case, by Lemma 2, we have to pay attention that we do not count  $(A, B)$  twice (in the case where  $A$  and  $B$  are non-isomorphic), which is isomorphic to  $(B, A)$

□

The first 32 values of the  $T_n$  series are displayed in Table A 4 The relationship between the first few terms in Table A 4 and the topologies in Figure A 2 is clear

### A 3 2 Number of Leaf-Labelled Complete Rooted Binary Trees

For each topology found according to  $T_n$ , Problem 3 (see Section A 2 3) poses the question how many addend arrangement permutations are possible? (being careful to avoid redundant duplicates due to commutative equivalences) Consider each topology  $\iota$  as a rooted binary tree  $(T, r)$ . Let  $A_\iota$  be the automorphism group of  $(T, r)$ . Then, the number of valid addend permutations  $a_n(\iota)$  (for topology  $\iota$  is determined by Equation A 6 where  $n$  is the number of set bits as before and  $\#A_\iota$  is the cardinal number of  $A_\iota$

$$a_n(\iota) = \frac{n!}{\#A_\iota} \quad (\text{A } 6)$$

To illustrate that Equation A 6 is indeed the solution being sought for a particular topology  $\iota$ , consider that  $n!$  is the order of the symmetric group of *all* permutations of the inputs, which may include commutative equivalences. These equivalences form a group, which is mathematically defined to be the automorphism group  $A_\iota$ . Since  $\#A_\iota$  gives the size of the automorphism group, it follows that Equation A 6 is the desired solution to Problem 3 for topology  $\iota$ .

Hence the number of valid permutations  $P_n$  for a given pattern with  $n$  bits set may be obtained by summing  $a_n(\iota)$  for each topology  $\iota$ , as expressed by Equation A 7

$$P_n = \sum_{\iota=1}^{T_n} a_n(\iota) \quad (\text{A } 7)$$

To illustrate this solution, consider the  $n = 4$  case where  $T_n = 2$  (see both topologies in Figure A 2). For topology 1 (bracket structure  $((P_{1n1}, P_{1n2}), (P_{1n3}, P_{1n4}))$ ),  $a_n(1) = 3$ . For topology 2 (bracket structure  $((P_{2n1}, P_{2n2}), P_{2n3}), P_{2n4}$ ),  $a_n(2) = 12$ . Hence  $P_n = 15$ . These 15 permutations are illustrated in Table A 3 for each of the cases where  $n = 4$ . The first 32 terms of the  $P_n$  series are illustrated in Table A 5. It is clear that  $P_n$  starts increasing dramatically with  $n > 4$ .

The solution for Problem 3 described by Equation A 6 and Equation A 7 builds on the solution found for  $T_n$  in Section A 3 2. The author in collaboration with S D Andres (Centre for Applied Information, University of Cologne) have investigated whether a solution for Problem 3 can be expressed independently. By using Equation A 7, the first few terms of the series were entered into "The On-Line Encyclopedia of Integer Sequences" [274]. This query returned the series shown in Equation A 8

$$P_n = (2n - 1)!! \quad (\text{A } 8)$$

Equation A 8 is the series of double factorial numbers, and also represents another solution for Problem 3. However Equation A 7 has the advantage that it explicitly derives the permutations for each individual topology (given by Equation A 6), which is useful from an implementation point of view.

Table A 5 The First 32 Values of  $P_n$

$n$	$P_n$
1	1
2	1
3	3
4	15
5	105
6	945
7	10395
8	135135
9	2027025
10	34459425
11	654729075
12	13749310575
13	316234143225
14	7905853580625
15	213458046676875
16	6190283353629375
17	191898783962510656
18	6332659870762851328
19	221643095476699725824
20	8200794532637891887104
21	319830986772877685030912
22	13113070457687992065589248
23	563862029680583581510926336
24	25373791335626255807872499712
25	1192568192774434313241077219328
26	58435841445947288807899666579456
27	2980227913743311873318071071408128
28	157952079428395541967994317459947520
29	8687364368561750048979716443232796672
30	495179769008019869670414288287988449280
31	29215606371473156745634515470640694165504
32	1782151988659863441581702123159121530191872

<b>ACL</b>	Addressing & Control Logic
<b>AMBA</b>	Advanced Microcontroller Bus Architecture
<b>ASIC</b>	Application Specific Integrated Circuit
<b>BAB</b>	Binary Alpha Block
<b>BMA</b>	Block Matching Algorithm
<b>CAE</b>	Context-based Arithmetic Encoding
<b>CCF</b>	Cross Correlation Function
<b>CIF</b>	Common Intermediate Format
<b>CISC</b>	Complex Instruction Set Computer
<b>CMM</b>	Constant Matrix Multiplication
<b>CMML</b>	Constant Matrix Multiplication Level
<b>CMOS</b>	Complementary Metal-Oxide Semiconductor
<b>CORDIC</b>	COordinate Rotation DIgital Computer
<b>CSD</b>	Canonic Signed Digit
<b>CSE</b>	Common Sub-Expression Elimination
<b>DA</b>	Distributed Arithmetic
<b>DCT</b>	Discrete Cosine Transform
<b>DFS</b>	Dynamic Frequency Scaling
<b>DFT</b>	Discrete Fourier Transform
<b>DPCM</b>	Differential Pulse Code Modulation

<b>DPL</b>	Dot Product Level
<b>DSP</b>	Digital Signal Processing
<b>DVS</b>	Dynamic Voltage Scaling
<b>DWT</b>	Discrete Wavelet Transform
<b>EDA</b>	Electronic Design Automation
<b>EDP</b>	Energy Delay Product
<b>EOD</b>	Even-Odd Decomposition
<b>EOR</b>	Even-Odd Recombination
<b>ESL</b>	Electronic System Level
<b>FFT</b>	Fast Fourier Transform
<b>FIR</b>	Finite Impulse Response
<b>FPGA</b>	Field Programmable Gate Array
<b>GPP</b>	General Purpose Processor
<b>GPS</b>	Global Positioning System
<b>GSM</b>	Global System for Mobile Communication
<b>HDD</b>	Hard Disk Drive
<b>HDL</b>	Hardware Description Language
<b>HDTV</b>	High Definition TeleVision
<b>HWMC</b>	HardWare Module Controller
<b>IDCT</b>	Inverse Discrete Cosine Transform
<b>IEC</b>	International Electrotechnical Commission
<b>IIR</b>	Infinite Impulse Response
<b>ISDN</b>	Integrated Services Digital Network
<b>ISO</b>	International Standardisation Organisation
<b>JPEG</b>	Joint Photographic Experts Group
<b>JTC</b>	Joint Technical Committee
<b>KLT</b>	Karhunen-Loeve Transform
<b>MAC</b>	Multiply ACcumulate

<b>MAD</b>	Mean Absolute Difference
<b>MCMM</b>	Multiple Constant Matrix Multiplication
<b>MF-CMM</b>	Multiplication Free Constant Matrix Multiplication
<b>MMX</b>	Multi-Media Extensions
<b>MPEG</b>	Motion Picture Experts Group
<b>MSD</b>	Minimal Signed Digit
<b>MSE</b>	Mean Square Error
<b>MV</b>	Motion Vector
<b>MWGM</b>	Multiplexed Weight Generation Module
<b>NEDA</b>	NEw Distributed Arithmetic
<b>NO-SA-DCT/IDCT</b>	Non Orthonormal SA-DCT/IDCT
<b>NTSC</b>	National Television Standards Committee
<b>OMSE</b>	Overall Mean Square Error
<b>PGCC</b>	Product of Gate count and Computation Cycles
<b>PNL</b>	Product Node List
<b>PO-SA-DCT/IDCT</b>	Pseudo Orthonormal SA-DCT/IDCT
<b>PPST</b>	Partial Product Summation Tree
<b>PSNR</b>	Peak Signal-to-Noise Ratio
<b>QCIF</b>	Quarter-Common Intermediate Format
<b>RAM</b>	Random Access Memory
<b>RISC</b>	Reduced Instruction Set Computer
<b>ROM</b>	Read Only Memory
<b>RTL</b>	Register Transfer Level
<b>SAD</b>	Sum of Absolute Differences
<b>SA-DCT</b>	Shape Adaptive Discrete Cosine Transform
<b>SA-IDCT</b>	Shape Adaptive Inverse Discrete Cosine Transform
<b>SD</b>	Signed Digit
<b>SIF</b>	Source Input Format



<b>SIMD</b>	Single Instruction Multiple Data
<b>SNL</b>	Skip Node List
<b>SPICE</b>	Simulation Program With Integrated Circuit Emphasis
<b>SRT</b>	Sweeney Robertson Tocher
<b>TRAM</b>	TRAnspose Memory
<b>UMTS</b>	Universal Mobile Telecommunications System
<b>VCD</b>	Value Change Dump
<b>VLC</b>	Variable Length Coding
<b>VLD</b>	Variable Length Decoding
<b>VLSI</b>	Very Large Scale Integration
<b>VO</b>	Video Object
<b>VOP</b>	Video Object Plane
<b>WAP</b>	Wireless Application Protocol

## List of MPEG Meeting Contributions

A Kinane, V Muresan, N OConnor, N Murphy and A Smeaton, *Hardware Acceleration Module for Shape Adaptive Discrete Cosine Transform for MPEG-4 Part 2*, ISO/IEC JTC1/SC29/WG11 M10883, in 69th MPEG-4 Meeting Document Register, Redmond, USA, July 19–23, 2004

V Muresan, D Larkin, A Kinane and N OConnor, *Status of CDVP's contribution to MPEG-4 Part 9 - Implementation Study Group*, ISO/IEC JTC1/SC29/WG11 M11104, in 69th MPEG-4 Meeting Document Register, Redmond, USA, July 19–23, 2004

A Kinane, V Muresan, and N OConnor, *Updated Status and Documentation on Hardware Acceleration Module for SA-DCT for MPEG-4 Part 2*, ISO/IEC JTC1/SC29/WG11 N6756, in 70th MPEG-4 Meeting Document Register, Palma de Mallorca, Spain, October 18–22, 2004

A Kinane, V Muresan, and N OConnor, *Architecture Update and Conformance Testing for SA-DCT Hardware Module*, ISO/IEC JTC1/SC29/WG11 M12093, in 72nd MPEG-4 Meeting Document Register, Busan, South Korea, April 18-22, 2005

A Kinane, V Muresan, and N OConnor, *Comments on Module Integration using University of Calgary's Multiple IP-Core Hardware Accelerated Software System Framework for MPEG4-Part9*, ISO/IEC JTC1/SC29/WG11 M11810, in 72nd MPEG-4 Meeting Document Register, Busan, South Korea, April 18–22, 2005

M Mattavelli, I Amer, W Badawy and A Kinane, *Overview of MPEG-4 Part-9 Reference HW Description*, ISO/IEC JTC1/SC29/WG11 N7543, in 74th MPEG-4 Meeting Document Register, Nice, France, October 17–21, 2005

A Kinane, V Muresan, and N OConnor, *Architecture Description and Conformance Testing of an SA-IDCT Hardware Module for MPEG-4 Part 2*, ISO/IEC JTC1/SC29/WG11 M12809, in 75th MPEG-4 Meeting Document Register, Bangkok, Thailand, January 16–20 2006

D Larkin, A Kinane, and N O'Connor, *Conformance testing of Binary Motion Estimation Hardware Core for MPEG-4 Binary Shape Coding*, ISO/IEC JTC1/SC29/WG11 M13182, in 76th MPEG-4 Meeting Document Register, Montreux, Switzerland, April 3–7, 2006

A Kinane, M Mattavelli, J Dubois and C Lucarz, *Suggested directory structure and linkage for IP cores on CVS server*, ISO/IEC JTC1/SC29/WG11 N8059, in 76th MPEG-4 Meeting Document Register, Montreux, Switzerland, April 3–7, 2006

List of Patents

"Power-Efficient Shape Adaptive Discrete Cosine Transformation", filed 23rd July 2005, Application number 2004/0498

## BIBLIOGRAPHY

- [1] G E Moore, "Cramming More Components onto Integrated Circuits," *Electronics*, vol 38, no 8, Apr 19 1965
- [2] G E Moore, "Lithography and the Future of Moore's law," in *Proc SPIE Electron-Beam, X-Ray, EUV and Ion-Beam Submicrometer Lithographies for Manufacturing V*, vol 2437, May 1995
- [3] Intel, "Moore's Law Raising the Bar," Intel Corporation," Backgounder, 2005 [Online] Available [ftp //download intel com/museum/Moores\\_Law/Printed\\_Materials/Moores\\_Law\\_Backgounder pdf](ftp://download.intel.com/museum/Moores_Law/Printed_Materials/Moores_Law_Backgounder.pdf)
- [4] Intel (2004, Aug 30,) Intel Drives Moores Law Forward with 65 Nanometer Process Technology [Online] Available [ftp //download intel com/museum/Moores\\_Law/Articles-Press\\_Releases/Press\\_Release\\_Aug2004 pdf](ftp://download.intel.com/museum/Moores_Law/Articles-Press_Releases/Press_Release_Aug2004.pdf)
- [5] M Hines (2003, Sept 5,) Mobile Subscribers Surge Worldwide [Online] Available [http //news zdnet co uk/hardware/mobile/0,39020360,39116137,00 htm](http://news.zdnet.co.uk/hardware/mobile/0,39020360,39116137,00.htm)
- [6] G Lawton, "New Technologies Place Video in Your Hand," *IEEE Computer*, vol 34, no 4, pp 14–17, Apr 2001
- [7] S Kelly (2004, July 11,) Mobile TV Coming of Age [Online] Available [http //news bbc co uk/2/hi/technology/3880069 stm](http://news.bbc.co.uk/2/hi/technology/3880069.stm)
- [8] K Lowe (2004, Aug 2,) Nearing the Age of Digital Media Portability [Online] Available [http //www eetasia com/ART\\_8800343444\\_499481,499483 HTM](http://www.eetasia.com/ART_8800343444_499481,499483.HTM)
- [9] M -T Sun, "Mobile Computing," Keynote Speech at Proc SPIE Video Communications and Image Processing (VCIP), Beijing, China, July 14, 2005
- [10] Nokia 7710 Technical Specifications Nokia [Online] Available [http //www nokia ie/nokia/0,8764,68685,00 html](http://www.nokia.ie/nokia/0,8764,68685,00.html)
- [11] L Mittag (2004, Nov 1,) The Future of Consumer Electronics Platforms [Online] Available [http //www embedded com/showArticle.jhtml?articleID=51201908](http://www.embedded.com/showArticle.jhtml?articleID=51201908)

- [12] J Yoshida (2005, Jan 24,) Video Trumps Voice in Cellphone of the Future [Online] Available [http://www.commsdesign.com/news/tech\\_beat/showArticle.jhtml?articleID=57702514](http://www.commsdesign.com/news/tech_beat/showArticle.jhtml?articleID=57702514)
- [13] R Merritt (2004, May 10,) Camera Phones Get Sexy, Do Users Care? [Online] Available <http://www.commsdesign.com/printableArticle/?articleID=20000183>
- [14] I Bell (2005, Mar 17,) MP3 Market To Nearly Quadruple By 2009 [Online] Available <http://news.designtechnica.com/article6862.html>
- [15] J Schopflin (2005, Oct 12,) Apple Unveils the New iPod [Online] Available <http://www.apple.com/pr/library/2005/oct/12ipod.html>
- [16] R Merritt (2000, Aug 11,) WAP Is Dead [Online] Available [http://www.clickz.com/experts/archives/ebiz/ebiz\\_report/article.php/827681](http://www.clickz.com/experts/archives/ebiz/ebiz_report/article.php/827681)
- [17] M Jacklin, "MPEG-4 – The Media Standard – The Landscape of Advanced Multimedia Coding," MPEG-4 Industry Forum," Internal Technical Report, Nov 2002
- [18] P Kauff, B Makai, S Rauthenberg, U Golz, J L D Lameilleure, and T Sikora, "Functional Coding of Video Using a Shape-Adaptive DCT Algorithm and an Object-Based Motion Prediction Toolbox," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 7, no 1, pp 181–196, Feb 1997
- [19] T Sikora, "Trends and Perspectives in Image and Video Coding," *Proceedings of the IEEE*, vol 93, no 1, pp 6–17, Jan 2005
- [20] T Sikora and B Makai, "Shape-Adaptive DCT for Generic Coding of Video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 5, no 1, pp 59–62, Feb 1995
- [21] P-C Tseng, Y-C Chang, Y-W Huang, H -C Fang, C -T Huang, and L -G Chen, "Advances in Hardware Architectures for Image and Video Coding – A Survey," *Proceedings of the IEEE*, vol 93, no 1, pp 184–197, Jan 2005
- [22] J Arreymb and M Dastbaz, "Issues in Delivering Multimedia Content to Mobile Devices," in *Proc of the Sixth International Conference on Information Visualisation (IV)*, July 10–12, 2002, pp 622–626
- [23] J F Huber, "Mobile Next-Generation Networks," *IEEE Multimedia*, vol 11, no 1, pp 72–83, January–March 2004
- [24] M Weiser, "The Computer for the 21st Century," *Scientific American*, vol 265, no 3, pp 94–104, Sept 1991
- [25] H D Man, "Connecting E-Dreams to Deep-Submicron Realities," Keynote Speech at International Workshop on Power and Timing Modelling, Optimization and Simulation (PATMOS), Santorini, Greece, Sept 15–17, 2004
- [26] Pentium-4 Datasheet Intel [Online] Available <http://developer.intel.com/design/Pentium4/documentation.htm>

- [27] S Furber, *ARM System-on-Chip Architecture*, 2nd ed Addison Wesley, 2000
- [28] RISC Wikipedia – The Free Encyclopedia [Online] Available <http://en.wikipedia.org/wiki/RISC>
- [29] ARM920T Processor ARM Ltd Cambridge, UK [Online] Available <http://www.arm.com/products/CPUs/ARM920T.html>
- [30] K Itoh, K Osada, and T Kawahara, “Low-Voltage Embedded RAMs – Current Status and Future Trends,” in *Proc International Workshop on Power and Timing Modelling Optimization and Simulation (PATMOS)*, Santorini, Greece, Sept 15–17, 2004, pp 3–15
- [31] A P Chandrakasan and R W Brodersen, “Minimizing Power Consumption in Digital CMOS Circuits,” *Proceedings of the IEEE*, vol 83, no 4, pp 498–523, Apr 1995
- [32] T Mudge, “Power A First-Class Architectural Design Constraint,” *IEEE Computer*, vol 34, no 4, pp 52–58, Apr 2001
- [33] S Rusu, “Trends and Challenges in High-Performance Microprocessor Design,” Keynote Speech at Electronic Design Processes (EDP) Workshop Monterey, California Intel Corporation, Apr 27, 2004
- [34] A P Chandrakasan, S Sheng, and R W Brodersen, “Low Power Digital CMOS Design,” *IEEE Journal of Solid-State Circuits*, vol 27, no 4, pp 473–483, Apr 1992
- [35] D J Frank, R H Dennard, E Nowak, P M Solomon, Y Taur, and H-S P Wong, “Device Scaling Limits of Si MOSFETs and Their Application Dependencies,” *Proceedings of the IEEE*, vol 89, no 3, pp 259–288, Mar 2001
- [36] F Pollack, “Challenges in Microarchitecture and Compiler Design,” Keynote Speech at Proc International Conference on Parallel Architectures and Compilation Techniques (PACT), Philadelphia, USA, Oct 16, 2000
- [37] P Landman, “Low-Power Architectural Design Methodologies,” Ph D dissertation, Univ of California at Berkeley, Aug 1994 [Online] Available [http://bwrc.eecs.berkeley.edu/Publications/1994/theses/lw\\_pwr\\_arch\\_des\\_meth\\_Landman/Landman94.pdf](http://bwrc.eecs.berkeley.edu/Publications/1994/theses/lw_pwr_arch_des_meth_Landman/Landman94.pdf)
- [38] P Pirsch, N Demassieux, and W Gehrke, “VLSI Architectures for Video Compression – A Survey,” *Proceedings of the IEEE*, vol 83, no 2, pp 220–246, Feb 1995
- [39] S W Smith, *Digital Signal Processing – A Practical Guide for Engineers and Scientists*, 1st ed Newnes, 2003
- [40] M Potkonjak, M B Srivastava, and A P Chandrakasan, “Multiple Constant Multiplications Efficient and Versatile Framework and Algorithms for Exploring Common Subexpression Elimination,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol 15, no 2, pp 151–165, Feb 1996
- [41] A Bovik, *Handbook of Image and Video Processing*, 2nd ed Elsevier Inc , 2005

- [42] M Vena and A Melodia, "Improving Data-Path Synthesis Performances by Means of Advanced Parallel Multiplication Algorithms for High-Speed, Low-Power and Low Area Digital Signal Processing," in *Synopsys Users Group Europe (SNUG)*, Munich, Germany, May 6–7, 2004
- [43] T Hope, "Enabling a Mobile Video (R)evolution," *ARM Information Quarterly*, vol 4, no 1, pp 43–45, 2005
- [44] N O'Connor and N Murphy (2003) Image and Video Compression – EE554 Taught M Eng Course Notes Dublin City University [Online] Available <http://www.eeng.dcu.ie/~oconnorm>
- [45] M Ghanbari, *Standard Codecs Image Compression to Advanced Video Coding*, 2nd ed IEE, 2003
- [46] K Rao and J Hwang, *Techniques and Standards for Image Video and Audio Coding*, 1st ed Prentice Hall, 1996
- [47] P Kuhn, *Algorithms Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*, 1st ed Dordrecht, The Netherlands Kluwer Academic, 1999
- [48] N O'Connor, V Muresan, A Kinane, D Larkin, S Marlow, and N Murphy, "Hardware Acceleration Architectures for MPEG-based Mobile Video Platforms A Brief Overview," in *Proc 4th European Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, London, Apr 9–11, 2003, pp 456–461
- [49] P Symes, *Video Compression Demystified*, 1st ed McGraw Hill, 2001
- [50] A K Jain, *Fundamentals of Digital Image Processing*, 1st ed Prentice Hall, 1988
- [51] W K Pratt, *Digital Image Processing*, 1st ed John Wiley & Sons, 1978
- [52] T Wiegand, G J Sullivan, G Bjontegaard, and A Luthra, "Overview of the H 264/AVC Video Coding Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 13, no 7, pp 560–576, July 2003
- [53] K Rao and P Yip, *Discrete Cosine Transform – Algorithms Advantages Applications*, 1st ed Academic Press, 1990
- [54] N Ahmed, T Natarajan, and K Rao, "Discrete Cosine Transform," *IEEE Transactions on Computers*, vol 23, pp 90–93, Jan 1974
- [55] Z Xiong, K Ramchandran, M T Orchard, and Y-Q Zhang, "A Comparative Study of DCT- and Wavelet-Based Image Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 9, no 5, pp 692–695, Aug 1999
- [56] H S Malvar, A Hallapuro, M Karczewicz, and L Kerofsky, "Low-Complexity Transform and Quantization in H 264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 13, no 7, pp 598–603, July 2003
- [57] T Sikora "MPEG Digital Video-Coding Standards," *IEEE Signal Processing Magazine*, vol 14, no 5, pp 82–100, Sept 1997



- [58] D Zhang and G Lu, "Segmentation of Moving Objects in Image Sequence A Review," *Journal of Circuits Systems and Signal Processing (Special Issue on Multimedia Communication Services)*, vol 20, no 2, pp 143–183, Feb 2001
- [59] K N Ngan, S Panchanathan, T Sikora, and M-T Sun, "Special Issue on Segmentation, Description and Retrieval of Video Content," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 8, no 5, Sept 1998
- [60] S-Y Chien, Y-W Huang, B-Y Hsieh, and L-G Chen, "Single Chip Video Segmentation System with a Programmable PE Array," in *Proc IEEE Asia Pacific Conference on ASIC*, Taipei, Taiwan, Aug 6–8, 2002
- [61] J Kim and T Chen, "A VLSI Architecture for Video-Object Segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 13, no 1, pp 83–96, Jan 2003
- [62] J Kim and T Chen, "Combining Static and Dynamic Features Using Neural Networks and Edge Fusion for Video Object Extraction," *IEE Proceedings on Vision Image and Signal Processing*, vol 150, no 3, pp 160–167, June 2003
- [63] M-H Yang, D J Kriegman, and N Ahuja, "Detecting Faces in Images A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol 24, no 1, pp 34–58, Jan 2002
- [64] D Chai and K N Ngan, "Face Segmentation Using Skin-Color Map in Videophone Applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 9, no 4, June 1999
- [65] N Brady, "MPEG-4 Standardized Methods for the Compression of Arbitrarily Shaped Video Objects," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 9, no 8, pp 1170–1189, Dec 1999
- [66] *MPEG-4 Information Technology – Coding of Audio Visual Objects – Part 2 Visual ISO/IEC 14496-2*, ISO/IEC Std, Rev Amendment 1, July 2000
- [67] M Gilge, T Engelhardt, and R Mehlan, "Coding of Arbitrarily Shaped Image Segments Based on a Generalized Orthogonal Transform," *Signal Processing and Image Communication*, vol 1, pp 153–180, 1989
- [68] T Sikora, S Bauer, and B Makai, "Efficiency of Shape-Adaptive 2-D Transforms for Coding of Arbitrarily Shaped Image Segments," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 5, no 3, pp 254–258, June 1995
- [69] M Owzar, M Talmi, G Heising, and P Kauff, "Evaluation of SA-DCT Hardware Complexity, ISO/IEC JTC1/SC29/WG11 N4407," in *47th MPEG-4 Meeting Document Register*, Seoul, South Korea, Mar 15–19, 1999
- [70] K Belloulata and J Konrad, "Fractal Image Compression With Region-Based Functionality," *IEEE Transactions on Image Processing*, vol 11, no 4, pp 351–362, Apr 2002
- [71] J Guo and P-F Shi "Object-Based Watermarking Scheme Robust to Object Manipulations," *IEE Electronics Letters*, vol 38, no 25, pp 1656–1657, 2002

- [72] A Foi, V Katkovnik, and K Eziagarian, "Pointwise Shape-Adaptive DCT as an Overcomplete Denoising Tool," in *Proc International Workshop on Spectral Methods and Multirate Signal Processing (SMMSP)*, Riga, Latvia, June 20–22, 2005
- [73] A Dasu and S Panchanathan, "A Survey of Media Processing Approaches," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 12, no 8, pp 633–645, Aug 2002
- [74] H -C Chang, L -G Chen, M -Y Hsu, and Y -C Chang, "Performance Analysis and Architecture Evaluation of MPEG-4 Video Codec System," in *Proc IEEE International Symposium on Circuits and Systems (ISCAS)*, vol 2, Geneva, Switzerland, May 28–31, 2000, pp 449–452
- [75] P Pirsch and H -J Stolberg, "VLSI Implementations of Image and Video Multimedia Processing Systems," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 8, no 7, pp 878–891, Nov 1998
- [76] V Muresan, N O'Connor, N Murphy, S Marlow, and S McGrath, "Low Power Techniques for Video Compression," in *Proc Irish Signals and Systems Conference (ISSC)*, University College Cork, June 25–26, 2002
- [77] Hantro 4250 CIF Encoder Hantro [Online] Available [http //www.hantro.com/en/products/codecs/hardware/4250.html](http://www.hantro.com/en/products/codecs/hardware/4250.html)
- [78] V Lappalainen, T D Hamalainen, and P Liuha, "Overview of Research Efforts on Media ISA Extensions and Their Usage in Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 12, no 8, pp 660–670, Aug 2002
- [79] B Hori and J Eyre (2005, Mar 14,) Inside DSP on Digital Video Processors for Video [Online] Available [http //insidedsp.eetimes.com/features/showArticle.jhtml?articleID=159401710](http://insidedsp.eetimes.com/features/showArticle.jhtml?articleID=159401710)
- [80] Philips PNX1500 Media Processor Philips Semiconductor [Online] Available [http //www.semiconductors.philips.com/acrobat/literature/9397/75010486.pdf](http://www.semiconductors.philips.com/acrobat/literature/9397/75010486.pdf)
- [81] Xtensa Product Overview Tensilica [Online] Available [http //www.tensilica.com/html/products.html](http://www.tensilica.com/html/products.html)
- [82] A Telikepalli, "Power vs Performance The 90 nm Inflection Point," Xilinx Inc , White Paper WP223, May 2005
- [83] A P Chandrakasan and R W Brodersen, *Low Power Digital CMOS Design*, 1st ed Dordrecht The Netherlands Kluwer Academic Publishers, 1995
- [84] G K Yeap, *Practical Low Power Digital VLSI Design*, 1st ed Dordrecht The Netherlands Kluwer Academic Publishers, 1998
- [85] D A Pucknell and K Eshragian, *Basic VLSI Design*, 3rd ed Australia Prentice Hall, 1994
- [86] J M Rabaey, A Chandrakasan, and B Nikolic, *Digital Integrated Circuits A Design Perspective*, 2nd ed Prentice Hall 2003

- [87] W Wolf, *Modern VLSI Design*, 3rd ed New Jersey Prentice Hall, 2002
- [88] K Roy and S C Prasad, *Low Power CMOS VLSI Circuit Design*, 1st ed New York Wiley, 2000
- [89] J A Butts and G S Sohi, "A Static Power Model for Architects," in *Proc 33rd Annual IEEE/ACM International Symposium on Microarchitecture*, Monterey, Canada, Dec 2000, pp 191–201
- [90] N S Kim, T Austin, D Baauw, T Mudge, K Flautner, J S Hu, M J Irwin, M Kandemir, and V Narayanan, "Leakage Current Moore's Law Meets Static Power," *IEEE Computer*, vol 36, pp 68–75, Dec 2003
- [91] G C-F Yeap, "Leakage Current in Low Standby Power and High Performance Devices Trends and Challenges," in *International Symposium on Physical Design*, San Diego, CA, Apr 2002, pp 22–27
- [92] L D Paulson, "Low Power Chips for High-Powered Handhelds," *IEEE Computer*, vol 36, no 1, pp 21–23, Jan 2003
- [93] W M Elgharbawy and M A Bayoumi, "Leakage Sources and Possible Solutions in Nanometer CMOS Technologies," *IEEE Circuits and Systems Magazine*, vol 5, no 4, pp 6–17, Fourth Quarter 2005
- [94] M Horowitz EE271 – Introduction to VLSI Systems (Course Notes) Stanford University [Online] Available <http://www.stanford.edu/class/ee271/lectures.html>
- [95] R Saleh ELEC481 – Digital Integrated Circuit Design (Course Notes) University of British Columbia [Online] Available <http://www.ece.ubc.ca/~elec481/>
- [96] M Horowitz, T Indermaur, and R Gonzalez, "Low-Power Digital Design," in *IEEE Symposium on Low Power Electronics*, San Diego, CA, Oct 1994, pp 8–11
- [97] M Casas-Sanchez, J Rizo-Morente, and C J Bleakley, "Power Consumption Characterisation of the Texas Instruments TMS320VC5510 DSP," in *Proc International Workshop on Power and Timing Modelling Optimization and Simulation (PATMOS)*, Leuven, Belgium, Sept 20–23, 2005
- [98] V Tiwari, S Malik, and A Wolfe, "Power Analysis of Embedded Software A First Step Towards Software Power Minimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol 2, no 4, pp 437–445, Dec 1994
- [99] S Steinke, M Knauer, L Wehmeyer, and P Marwedel, "An Accurate and Fine Grain Instruction-Level Energy Model Supporting Software Optimizations," in *Proc International Workshop on Power and Timing Modelling Optimization and Simulation (PATMOS)*, Yverdon-Les-Bains, Switzerland, Sept 26–28, 2001
- [100] N Chang, K Kim, and H G Lee, "Cycle-Accurate Energy Consumption Measurement and Analysis Case Study of ARM7TDMI" in *Proc International Symposium on Low Power Electronics and Design (ISLPED)*, Rapallo, Italy, July 25–27, 2000, pp 185–190

- [101] N Chang, K Kim, and H G Lee, "Cycle-Accurate Energy Measurement and Characterization With a Case Study of the ARM7TDMI," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol 10, no 2, pp 146–154, Apr 2002
- [102] A Sinha and A P Chandrakasan, "Energy Aware Software," in *Proc Thirteenth International Conference on VLSI Design*, Calcutta, Jan 3–7, 2000, pp 50–55
- [103] A Sinha and A P Chandrakasan, "JouleTrack – A Web Based Tool for Software Energy Profiling," in *Proc ACM 38th Design Automation Conference (DAC)*, Las Vegas, NV, June 18–21, 2001
- [104] A Sinha, N Ickes, and A P Chandrakasan, "Instruction Level and Operating System Profiling for Energy Exposed Software," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol 11, no 6, pp 1044–1057, Dec 2003
- [105] F N Najm, "A Survey of Power Estimation Techniques in VLSI Circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol 2, no 4, pp 446–455, Dec 1994
- [106] (2005) Synopsys PrimePower Manual X-2005 06 Synopsys Inc USA
- [107] (2005) Synopsys Power Compiler User Guide W-2004 12 Synopsys Inc USA
- [108] M Bohr, "Intel's 90nm Technology Moore's Law and More," in *Intel Developers Forum*, Sept 12, 2002
- [109] K-H Chen, J-I Guo, J-S Wang, C-W Yeh, and J-W Chen, "An Energy-Aware IP Core Design for the Variable-Length DCT/IDCT Targeting at MPEG4 Shape-Adaptive Transforms," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 15, no 5, pp 704–715, May 2005
- [110] R A Powers, "Batteries for Low Power Electronics," *Proceedings of the IEEE*, vol 83, no 4, pp 687–693, Apr 1995
- [111] J D Shepard and L Brush, "Market Trends in Smart Battery Packs for Portable Electronics," in *Proc The Thirteenth Annual Battery Conference on Applications and Advances*, Jan 13–16, 1998
- [112] I Buchmann, "Understanding your Batteries in a Portable World," in *Proc The Fourteenth Annual Battery Conference on Applications and Advances*, Jan 12–15, 1999
- [113] D Ross, "Power Struggle (Power Supplies for Portable Equipment)," *IEE Review*, vol 49, no 7, pp 34–38, July 2003
- [114] (2003, Sept 10,) Micro Fuel Cells May Replace Batteries in Mobile Phones ST Microelectronics [Online] Available <http://www.st.com/stonline/press/news/year2003/t1346h.htm>
- [115] Y Hara (2003, Sept 23,) Fuel Cells Power Up for Notebooks [Online] Available <http://www.techweb.com/wire/story/TWB20030923S0002>

- [116] F Poppen (2002, May) Low Power Design Guide Low Power Design and Services Group – OFFIS Research Institute [Online] Available <http://www.lowpower.de/charter/lpdesignguide.pdf>
- [117] A Krishnamoorthy (2004, July 15,) Minimize IC Power Without Sacrificing Performance [Online] Available <http://www.eedesign.com/article/showArticle.jhtml?articleId=23901143>
- [118] A Jerraya, H Tenhunen, and W Wolf, “Multiprocessor Systems-on-Chips,” *IEEE Computer*, vol 38, no 7, pp 36–40, July 2005
- [119] (2004, May 17,) ARM Announces First Integrated Multiprocessor Core ARM Ltd [Online] Available <http://www.arm.com/news/5346.html>
- [120] A N Sloss, D Symes, and C Wright, *ARM System Developer's Guide*, 1st ed Morgan Kaufman, 2004
- [121] J Bond (2003, July 30,) Memory Amnesia Could Hurt Low-Power Design [Online] Available <http://www.usdesign-reuse.com/articles/article6012.html>
- [122] V Muresan, “Power Management for MPEG-4 Peripherals Targeted to Mobile Multimedia Platforms,” Centre for Digital Video Processing – Dublin City University,” Internal Technical Report, May 2002
- [123] K Chun and A Ling (2003, Nov 24,) Placement Approach Cuts SoC Power Needs [Online] Available [http://www.eetimes.com/in\\_focus/sihcon\\_engineering/OEG20031121S0035](http://www.eetimes.com/in_focus/sihcon_engineering/OEG20031121S0035)
- [124] T Xanthopoulos and A P Chandrakasan, “A Low-Power DCT Core Using Adaptive Bitwidth and Arithmetic Activity Exploiting Signal Correlations and Quantization,” *IEEE Journal of Solid-State Circuits*, vol 35, no 5, pp 740–750, May 2000
- [125] S Patel (2003, June 9,) Low-Power Design Techniques Span RTL-to-GDSII Flow [Online] Available <http://www.eedesign.com/story/OEG20030609S0059>
- [126] P Eles, K Kuchcinski, and Z Peng, *System Synthesis with VHDL* Dordrecht The Netherlands Kluwer Academic Publishers, 1998
- [127] X Wang VHDL and High Level Synthesis – EE540 Taught M Eng Course Notes Dublin City University [Online] Available <http://www.eeng.dcu.ie/~wangx>
- [128] M C McFarland, A C Parker, and R Camposano, “The High-Level Synthesis of Digital Systems,” *Proceedings of the IEEE*, vol 78, no 2, pp 301–318, Feb 1990
- [129] Synthesizer Data Sheet Forte Design Systems [Online] Available [http://www.forted.com/products/c synthesizer\\_datasheet.pdf](http://www.forted.com/products/c synthesizer_datasheet.pdf)
- [130] DK Design Suite Celoxica Ltd [Online] Available <http://www.celoxica.com/products/dk/default.asp>
- [131] Catapult C Synthesis Data Sheet Mentor Graphics [Online] Available [www.mentor.com/products/c-based\\_design/catapult\\_c\\_synthesis/index.cfm](http://www.mentor.com/products/c-based_design/catapult_c_synthesis/index.cfm)

- [132] N Boullis and A Tisserand, "Some Optimizations of Hardware Multiplication by Constant Matrices," in *Proc 16th IEEE Symposium on Computer Arithmetic*, June 2003, pp 20–27
- [133] A Shams, W Pan, A Chidanandan, and M A Bayoumi, "A Low Power High Performance Distributed DCT Architecture," in *Proc IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Apr 2002, pp 21–27
- [134] A G Dempster, M D Macleod, and O Gustafsson, "Comparison of Graphical and Sub-expression Elimination Methods for Design of Efficient Multipliers," in *Proc Thirty Eighth Asilomar Conference on Signals Systems and Computers*, vol 1, Monterey, CA, Nov 7–10, 2004, pp 72–76
- [135] Distributed Arithmetic FIR Xilinx Inc California, USA [Online] Available [http //www xilinx com/ipcenter/catalog/logiccore/docs/da\\_fir.pdf](http://www.xilinx.com/ipcenter/catalog/logiccore/docs/da_fir.pdf)
- [136] PDA FIR Filter Product Specification eInfochips Inc California, USA [Online] Available [http //www xilinx com/bvdocs/ipcenter/data\\_sheet/PDA\\_FIR\\_filter.pdf](http://www.xilinx.com/bvdocs/ipcenter/data_sheet/PDA_FIR_filter.pdf)
- [137] CoreFIR Finite Impulse Response FIR Filter Generator Data Sheet Actel Corporation California, USA [Online] Available [http //www actel com/ipdocs/CoreFIR\\_IP\\_DS.pdf](http://www.actel.com/ipdocs/CoreFIR_IP_DS.pdf)
- [138] S Haider (1997, Oct ) Datapath Synthesis and Optimization for High-Performance ASICs Synopsys Inc California, USA [Online] Available [http //www synopsys com/products/datapath/datapath\\_bgr.html](http://www.synopsys.com/products/datapath/datapath_bgr.html)
- [139] T-S Chang, C-S Kung, and C-W Jen, "A Simple Processor Core Design for DCT/IDCT," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 10, no 3, pp 439–447, Apr 2000
- [140] Y-K Lai and H-J Hsu, "A Cost-Effective 2-D Discrete Cosine Transform Processor with Reconfigurable Datapath," in *Proc IEEE International Symposium on Circuits and Systems (ISCAS)*, vol 2, Bangkok, Thailand, May 23–26, 2003, pp 492–495
- [141] W-H Chen, C H Smith, and S Frahck, "A Fast Computational Algorithm for the Discrete Cosine Transform," *IEEE Transactions on Communications*, vol 25, no 9, pp 1004–1009, Sept 1977
- [142] W B Pennebaker and J L Mitchell, *JPEG Still Image Data Compression Standard* Chapman and Hall, 1993
- [143] B G Lee, "A New Algorithm to Compute the Discrete Cosine Transform," *IEEE Transactions on Acoustics Speech and Signal Processing*, vol 32, no 6, pp 1243–1245, Dec 1984
- [144] C Loeffler, A Ligtenberg, and G S Moschytz, "Practical Fast 1-D DCT Algorithms with 11 Multiplications," in *Proc IEEE International Conference on Acoustics Speech and Signal Processing ICASSP*, vol 2, May 23–26, 1989, pp 988–991

- [145] M Martina, A Molino, and F Vacca, "Reconfigurable and Low Power 2D-DCT IP for Ubiquitous Multimedia Streaming," in *Proc IEEE International Conference on Multimedia and Expo (ICME)*, vol 2, Lausanne, Switzerland, Aug 26–29, 2002, pp 177–180
- [146] E Feig and S Winograd, "Fast Algorithms for the Discrete Cosine Transform," *IEEE Transactions on Signal Processing*, vol 40, no 9, pp 2174–2193, Sept 1992
- [147] Y Arai, T Agui, and M Nakajima, "A Fast DCT-SQ Scheme for Images," *The Transactions of the IEICE*, vol E 71, no 11, pp 1095–1097, Nov 1988
- [148] K K Parhi and T Nishitani, *Digital Signal Processing for Multimedia Systems*, 1st ed Dekker, 1999
- [149] C-M Wu and A Chiou, "A SIMD-Systolic Architecture and VLSI Chip for the Two-Dimensional DCT and IDCT," *IEEE Transactions on Consumer Electronics*, vol 39, no 4, pp 859–869, Nov 1993
- [150] S B Pan and R-H Park, "Unified Systolic Arrays for Computation of the DCT/DST/DHT," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 7, no 2, pp 413–419, Apr 1997
- [151] A Aggoun and I Jalloh, "Two-Dimensional DCT/IDCT Architecture," *IEE Proceedings Computers and Digital Techniques*, vol 150, no 1, pp 2–10, Jan 2003
- [152] C Cheng and K K Parhi, "A Novel Systolic Array Structure for DCT," *IEEE Transactions on Circuits and Systems II Express Briefs*, vol 52, no 7, pp 366–369, July 2005
- [153] S Hsiao and J Tseng, "New Matrix Formulation for Two-Dimensional DCT/IDCT Computation and its Distributed-Memory VLSI Implementation," *IEE Proceedings Vision Image and Signal Processing*, vol 149, no 2, pp 97–107, Apr 2002
- [154] M P Leong and P H W Leong, "A Variable-Radix Digit-Serial Design Methodology and its Application to the Discrete Cosine Transform," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol 11, no 1, pp 90–104, Feb 2003
- [155] S Bique, "New Characterizations of 2D Discrete Cosine Transform," *IEEE Transactions on Computers*, vol 54, no 9, pp 1054–1060, Sept 2005
- [156] S-F Hsiao and W-R Shue, "A New Hardware-Efficient Algorithm and Architecture for Computation of 2-D DCTs on a Linear Array," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 11, no 11, pp 1149–1159, Nov 2001
- [157] Y-H Chan, L-P Chau, and W-C Siu, "Efficient Implementation of Discrete Cosine Transform using Recursive Filter Structure," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 4, no 6, pp 550–552, Dec 1994
- [158] A Elnaggar and H M Alnuweiri, "A New Multidimensional Recursive Architecture for Computing the Discrete Cosine Transform," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 10, no 1, pp 113–119, Feb 2000

- [159] C -H Chen, B -D Liu, and J -F Yang, "Condensed Recursive Structures for Computing Multi-dimensional DCT/IDCT With Arbitrary Length," *IEEE Transactions on Circuits and Systems I Regular Papers*, vol 52, no 9, pp 1819–1831, Sept 2005
- [160] I Koren, *Computer Arithmetic Algorithms*, 2nd ed Natick, MA AK Peters, 2002
- [161] H Jeong, J Kim, and W kyung Cho, "Low-Power Multiplierless DCT Architecture using Image Correlation," *IEEE Transactions on Consumer Electronics*, vol 50, no 1, pp 262–267, Feb 2004
- [162] S -F Hsiao, Y H Hu, T -B Juang, and C -H Lee, "Efficient VLSI Implementations of Fast Multiplierless Approximated DCT Using Parameterized Hardware Modules for Silicon Intellectual Property Design," *IEEE Transactions on Circuits and Systems I Regular Papers*, vol 52, no 8, pp 1568–1579, Aug 2005
- [163] C A Christopoulos, J G Bormans, A N Skodras, and J P Cornelis, "Efficient Computation of the Two-Dimensional Fast Cosine Transform," in *Proc SPIE Hybrid Image and Signal Processing IV*, vol 2238, June 1994, pp 229–237
- [164] I -M Pao and M -T Sun, "Approximation of Calculations for Forward Discrete Cosine Transform," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 8, no 3, pp 264–268, June 1998
- [165] I -M Pao and M -T Sun, "Modeling DCT Coefficients for Fast Video Encoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 9, no 4, pp 608–616, June 1999
- [166] N J August and D S Ha, "Low Power Design of DCT and IDCT for Low Bit Rate Video Codecs," *IEEE Transactions on Multimedia*, vol 6, no 3, pp 414–422, June 2004
- [167] C -P Lin, P -C Tseng, and L -G Chen, "Nearly Lossless Content-Dependent Low-Power DCT Design for Mobile Video Applications," in *Proc IEEE International Conference on Multimedia and Expo (ICME)*, vol 3, Amsterdam, The Netherlands, July 6–8, 2005
- [168] V Dimitrov and G Jullien, "Multidimensional Algebraic-Integer Encoding for High Performance Implementation of DCT and IDCT," *IEE Electronics Letters*, vol 39, no 7, pp 602–603, Apr 2003
- [169] M Fu, G Jullien, V Dimitrov, M Ahmadi, and W Miller, "The Application of 2D Algebraic Integer Encoding to a DCT IP Core," in *Proc IEEE International Workshop on System-on-Chip for Real-Time Applications*, Jun 30 — Jul 2 2003, pp 66–69
- [170] M Fu, G Jullien, V Dimitrov, and M Ahmadi, "A Low-Power DCT IP Core Based on 2D Algebraic Integer Encoding," in *Proc IEEE International Symposium on Circuits and Systems (ISCAS)*, vol 2, Vancouver, Canada, May 23–26, 2004, pp 765–768
- [171] V Dimitrov, G Jullien, and W Miller, "A New DCT Algorithm Based on Encoding Algebraic Integers," in *Proc IEEE International Acoustics Speech and Signal Processing (ICASSP)*, vol 3, Seattle, WA, May 12–15, 1998 pp 351–356



- [172] G Sullivan and A Luthra, "Call for Proposals on Fixed-Point 8x8 IDCT and DCT Standard, ISO/IEC JTC1/SC29/WG11 N7099," in *72nd MPEG-4 Meeting Document Register*, Busan, Korea, Apr 18-22, 2005
- [173] *Standard Specification for the Implementations of 8x8 Inverse Discrete Cosine Transform*, IEEE Std 1180-1990, Mar 1991
- [174] 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
- [175] S Hwang, G Han, S Kang, and J Kim, "New Distributed Arithmetic Algorithm for Low-Power FIR Filter Implementation," *IEEE Signal Processing Letters*, vol 11, no 5, pp 463-466, May 2004
- [176] T Masaki, Y Morimoto, T Onoye, and I Shirakawa, "VLSI Implementation of Inverse Discrete Cosine Transformer and Motion Compensator for MPEG2 HDTV Video Decoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 5, no 5, pp 387-395, Oct 1995
- [177] S-K Paek, J-H Kim, B-S Kwon, D-H Chung, and L-S Kim, "A Mode-Changeable 2-D DCT/IDCT Processor for Digital VCR," *IEEE Transactions on Consumer Electronics*, vol 42, no 3, pp 606-616, Aug 1996
- [178] H C Karathanasis, "A Low ROM Distributed Arithmetic Implementation of the Forward/Inverse DCT/DST using Rotations," *IEEE Transactions on Consumer Electronics*, vol 41, no 2, pp 263-272, May 1995
- [179] E Scopa, A Leone, R Guerrieri, and G Baccarani, "A 2D-DCT Low-Power Architecture for H 261 Coders," in *Proc IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, vol 5, May 9-12, 1995, pp 3271-3274
- [180] S-F Hsiao, W-R Shue, and J-M Tseng, "A Cost-Efficient and Fully-Pipelined Architecture for DCT/IDCT," *IEEE Transactions on Consumer Electronics*, vol 45, no 3, pp 515-525, Aug 1999
- [181] K Kim and J-S Koh, "An Area Efficient DCT Architecture for MPEG-2 Video Encoder," *IEEE Transactions on Consumer Electronics*, vol 45, no 1, pp 62-67, Feb 1999
- [182] S Yu and E E S Jr, "DCT Implementation with Distributed Arithmetic," *IEEE Transactions on Computers*, vol 50, no 9, pp 985-991, Sept 2001
- [183] L Fanucci and S Saponara, "Data Driven VLSI Computation for Low Power DCT-Based Video Coding," in *Proc 9th International Conference on Electronics Circuits and Systems*, vol 2, Sept 15-18, 2002, pp 541-544
- [184] J-I Guo, J-W Chen, and J-C Chen, "A New 2-D 8x8 DCT/IDCT Core Design Using Group Distributed Arithmetic," in *Proc IEEE International Symposium on Circuits and Systems (ISCAS)*, vol 2, Bangkok, Thailand, May 23-26, 2003, pp 752-755

- [185] T Darwish and M Bayoumi, "Energy Aware Distributed Arithmetic DCT Architectures," in *Proc IEEE International Workshop on Signal Processing Systems (SIPS)*, Seoul, Korea, Aug 27–29, 2003, pp 351–356
- [186] J Park and K Roy, "A Low Power Reconfigurable DCT Architecture to Trade Off Image Quality for Computational Complexity," in *Proc IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, Montreal, Canada, May 17–21, 2004, pp 17–20
- [187] R -C Ju, J -W Chen, J Guo, and T -F Chen, "A Parameterized Power Aware IP Core Generator for the 2-D 8x8 DCT/IDCT," in *Proc IEEE International Symposium on Circuits and Systems (ISCAS)*, vol 2, Vancouver, Canada, May 23–26, 2004, pp 769–772
- [188] H -C Chen, J -I Guo, T -S Chang, and C -W Jen, "A Memory-Efficient Realization of Cyclic Convolution and its Application to Discrete Cosine Transform," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 15, no 3, pp 445–453, Mar 2005
- [189] A Madiseti and A N W Jr, "A 100 MHz 2-D  $8 \times 8$  DCT/IDCT Processor for HDTV Applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 5, no 2, pp 158–165, Apr 1995
- [190] D W Kim, T W Kwon, J M Seo, J K Yu, S K Lee, J H Suk, and J R Choi, "A Compatible DCT/IDCT Architecture using Hardwired Distributed Arithmetic," in *Proc IEEE International Symposium on Circuits and Systems (ISCAS)*, vol 2, May 6–9, 2001, pp 457–460
- [191] S Gundimada and V K Asari, "Data Dependency Considerations in Low Power Design of Discrete Cosine Transform Architecture," in *Proc International Conference on Embedded Systems and Applications (ESA)*, Las Vegas, NV, June 2003, pp 275–284
- [192] M Alam, W Badawy, and G Jullien, "Time Distributed DCT Architecture for Multimedia Applications," in *Proc IEEE International Conference on Consumer Electronics (ICCE)*, LA, USA, June 17–19, 2003, pp 166–167
- [193] M Alam, W Badawy, and G Jullien, "A New Time Distributed DCT Architecture for MPEG-4 Hardware Reference Model," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 15, no 5, pp 726–730, May 2005
- [194] J -I Guo, R -C Ju, and J -W Chen, "An Efficient 2-D DCT/IDCT Core Design Using Cyclic Convolution and Adder-Based Realization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 14, no 4, pp 416–428, Apr 2004
- [195] J Gause, P Y Cheung, and W Luk, "Static and Dynamic Reconfigurable Designs for a 2D Shape-Adaptive DCT," in *Proc 10th International Workshop on Field-Programmable Logic and Applications (FPL)*, 2000, pp 96–105
- [196] J Gause, P Cheung, and W Luk, "Reconfigurable Computing for Shape-Adaptive Video Processing" *IEE Proceedings Computers and Digital Techniques*, vol 151, no 5, pp 313–320, Sept 2004

- [197] T Le and M Glesner, "Flexible Architectures for DCT of Variable-Length Targeting Shape-Adaptive Transform," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 10, no 8, pp 1489–1495, Dec 2000
- [198] T Le and M Glesner, "Configurable VLSI Architectures for Both Standard DCT and Shape-Adaptive DCT in Future MPEG-4 Circuit Implementations," in *Proc IEEE International Symposium on Circuits and Systems (ISCAS)*, vol 2, Geneva, Switzerland, May 28–31, 2000, pp 461–464
- [199] T Le and M Glesner, "A New Flexible Architecture for Variable-Length DCT Targeting Shape-Adaptive Transform," in *Proc IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, Phoenix, Arizona, Mar 1999, pp 1949–1952
- [200] T Le, T Dombek, and M Glesner, "On the Design of a Novel Architecture for Shape-Adaptive DCT Targeting Image Coding," in *Proc of the IEEE International Conference on Electronics Circuits and Systems (ICECS)*, Paphos, Cyprus, Sept 5–8, 1999, pp 1139–1142
- [201] P-C Tseng, C -T Haung, and L -G Chen, "Reconfigurable Discrete Cosine Transform Processor for Object-Based Video Signal Processing," in *Proc IEEE International Symposium on Circuits and Systems (ISCAS)*, vol 2, Vancouver, Canada, May 23–26, 2004, pp 353–356
- [202] K -H Chen, J -I Guo, J -S Wang, C -W Yeh, and T -F Chen, "A Power-Aware IP Core Design for the Variable-Length DCT/IDCT Targeting at MPEG-4 Shape-Adaptive Transforms," in *Proc IEEE International Symposium on Circuits and Systems (ISCAS)*, vol 2, Vancouver, Canada, May 23–26, 2004, pp 141–144
- [203] K -B Lee, H -C Hsu, and C -W Jen, "A Cost-Effective MPEG-4 Shape-Adaptive DCT with Auto-Aligned Transpose Memory Organization," in *Proc IEEE International Symposium on Circuits and Systems (ISCAS)*, vol 2, Vancouver, Canada, May 23–26, 2004, pp 777–780
- [204] A Kinane, V Muresan, N O'Connor, N Murphy, and S Marlow, "Energy-Efficient Hardware Architecture for Variable N-point 1D DCT," in *Proc International Workshop on Power and Timing Modelling Optimization and Simulation (PATMOS)*, Santorini, Greece, Sept 15–17, 2004, pp 780–788
- [205] K Hwang, *Computer Arithmetic Principles Architecture and Design* Wiley, 1979
- [206] B Parhami, *Computer Arithmetic Algorithms and Hardware Designs* Oxford University Press, 1999
- [207] V G Oklobdzija and D Villegier, "Improving Multiplier Design by Using Improved Column Compression Tree and Optimized Final Adder in CMOS Technology," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol 3, no 2, pp 292–301, June 1995
- [208] A Kinane, V Muresan, and N O'Connor, "An Optimal Adder-Based Hardware Architecture for the DCT/SA-DCT," in *Proc SPIE Video Communications and Image Processing (VCIP)*, Beijing, China, July 12–15, 2005, pp 1410–1417

- [209] W Li, J-R Ohm, M van der Schaar, H Jiang, and S Li, "MPEG-4 Video Verification Model version 18 0," in *ISO/IEC JTC1/SC29/WG11 N3908*, Pisa, Italy, Jan 2001
- [210] *Text of ISO/IEC DTR 14496-7 Information Technology – Coding of Audio-Visual Objects – Part 7 Optimized Reference Software for Coding of Audio-Visual Objects*, ISO/IEC Std, Rev microsoft-v2 4-070710-NTU [Online] Available <http://megaera.ee.nctu.edu.tw/mpeg/>
- [211] Integrator/CM920T ARM Ltd Cambridge, UK [Online] Available <http://www.arm.com/products/DevTools/IntegratorCM920T.html>
- [212] *Text of ISO/IEC TR 14496-9 Information Technology – Coding of Audio Visual Objects – Part 9 Reference Hardware Description*, ISO/IEC Std, Rev 2, 2005
- [213] T S Mohamed and W Badawy, "Integrated Hardware-Software Platform for Image Processing Applications," in *Proc 4th IEEE International Workshop on System-on-Chip for Real-Time Applications (IWSOC)*, Alberta, Canada, July 19–21, 2004
- [214] A Kinane, V Muresan, and N O'Connor, "Architecture Update and Conformance Testing for SA-DCT Hardware Module, ISO/IEC JTC1/SC29/WG11 M12093," in *72nd MPEG-4 Meeting Document Register*, Busan, South Korea, Apr 18–22, 2005
- [215] D Larkin, A Kinane, V Muresan, and N O'Connor, "Efficient Hardware Architectures for MPEG-4 Core Profile," in *Proc 9th Irish Machine Vision and Image Processing Conference (IMVIP)*, Belfast, Northern Ireland, Aug 30–31, 2005, pp 249–252
- [216] A Kinane, A Casey, V Muresan, and N O'Connor, "FPGA-Based Conformance Testing and System Prototyping of an MPEG-4 SA-DCT Hardware Accelerator," in *Proc IEEE International Conference on Field Programmable Technology (FPT)*, Singapore, Dec 11–14, 2005, pp 317–318
- [217] T S Mohamed and W Badawy, "Multiple IP-Core Hardware Accelerated Software System Framework for MPEG-4 Part 9," in *ISO/IEC JTC1/SC29/WG11 M10954 Contribution to AHG on MPEG-4 Part 9 Reference Hardware*, Redmond, USA, July 2004
- [218] Annapolis WildCard-II Datasheet Annapolis Micro Systems Inc Maryland, USA [Online] Available <http://www.annapmicro.com/wildcard2.html>
- [219] Integrator Compact Platform ARM Ltd Cambridge, UK [Online] Available <http://www.arm.com/products/DevTools/IntegratorCP.html>
- [220] Integrator Xilinx Logic Module ARM Ltd Cambridge, UK [Online] Available <http://www.arm.com/products/DevTools/IntegratorLM-XCV2000E.html>
- [221] A Casey, "Virtual Socket Framework for the Integration of MPEG-4 Hardware Accelerators and MPEG-4 Software Tools on the ARM Integrator Compact Platform," Master's thesis, School of Electronic Engineering, Dublin City University, Ireland, 2004

- [222] C-H Chen, B-D Liu, J-F Yang, and J-L Wang, "Efficient Recursive Structures for Forward and Inverse Discrete Cosine Transform," *IEEE Transactions on Signal Processing*, vol 52, no 9, pp 2665–2669, Sept 2004
- [223] Y H Hu and Z Wu, "An Efficient CORDIC Array Structure for the Implementation of Discrete Cosine Transform," *IEEE Transactions on Signal Processing*, vol 43, no 1, pp 331–336, Jan 1995
- [224] F Zhou and P Kornerupt, "High Speed DCT/IDCT Using a Pipelined CORDIC Algorithm," in *Proc 12th IEEE Symposium on Computer Arithmetic*, July 1995, pp 180–187
- [225] Y Yang, C Wang, M O Ahmad, and M Swamy, "An FPGA Implementation of an On-Line Radix-4 CORDIC 2-D IDCT Core," in *Proc IEEE International Symposium on Circuits and Systems (ISCAS)*, vol 4, Scottsdale, Arizona, May 26–29, 2002, pp 763–766
- [226] S ichi Uramoto, Y Inoue, A Takabatake, J Takeda, Y Yamashita, H Terane, and M Yoshimoto, "A 100-MHz 2-D Discrete Cosine Transform Core Processor," *IEEE Journal of Solid-State Circuits*, vol 27, no 4, pp 492–499, Apr 1992
- [227] T Xanthopoulos and A P Chandrakasan, "A Low-Power IDCT Macrocell for MPEG-2 MP@ML Exploiting Data Distribution Properties for Minimal Activity," *IEEE Journal of Solid-State Circuits*, vol 34, no 9, pp 693–703, May 1999
- [228] L McMillan and L Westover, "A Forward-Mapping Realization of the Inverse Discrete Cosine Transform," in *Proc Data Compression Conference (DCC)*, Mar 24–27, 1992, pp 219–228
- [229] S-C Hsia, B-D Liu, J-F Yang, and B-L Bai, "VLSI Implementation of Parallel Coefficient-by-Coefficient Two-Dimensional IDCT Processor," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 5, no 5, pp 396–406, Oct 1995
- [230] J Lee, N Vijaykrishnan, and M Irwin, "Efficient VLSI Implementation of Inverse Discrete Cosine Transform," in *Proc IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, vol 5, Montreal, Canada, May 17–21, 2004, pp 177–180
- [231] J Lee, N Vijaykrishnan, M Irwin, and R Radhakrishnan, "Inverse Discrete Cosine Transform Architecture Exploiting Sparseness and Symmetry Properties," in *Proc IEEE International Workshop on Signal Processing Systems (SIPS)*, Oct 13–15, 2004, pp 361–366
- [232] T Xanthopoulos, Y Yaoi, and A P Chandrakasan, "A Data-Driven IDCT Architecture for Low Power Video Applications," in *Proc European Solid State Circuits Conference (ESSCIRC)*, Sept 1996
- [233] T Xanthopoulos, A P Chandrakasan, C Sodini, and W Dally, "Architectural Exploration Using Verilog-Based Power Estimation A Case Study of the IDCT," in *Proc ACM 38th Design Automation Conference (DAC)*, Anaheim, CA, June 9–13, 1997, pp 415–420
- [234] S Kim, C H Ziesler, and M C Papaefthymiou, "A Reconfigurable Pipelined IDCT for Low-Energy Video Processing," in *Proc 13th Annual International ASIC/SOC Conference*, Sept 13–16, 2000, pp 13–17

- [235] H-C Hsu, K-B Lee, N Y-C Chang, and T-S Chang, "An MPEG4 Shape-Adaptive Inverse DCT with Zero Skipping and Auto-Aligned Transpose Memory," in *Proc IEEE Asia-Pacific Conference on Circuits and Systems*, Dec 6–9, 2004, pp 773–776
- [236] A Kinane, V Muresan, and N O'Connor, "Architecture Description and Conformance Testing of an SA-IDCT Hardware Module for MPEG-4 Part 2, ISO/IEC JTC1/SC29/WG11 M12809," in *75th MPEG-4 Meeting Document Register*, Bangkok, Thailand, Jan 16–20, 2006
- [237] I-C Park and H-J Kang, "Digital Filter Synthesis Based on an Algorithm to Generate All Minimal Signed Digit Representations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol 21, no 12, pp 1525–1529, Dec 2002
- [238] A G Dempster and M D Macleod, "Generation of Signed-Digit Representations for Integer Multiplication," *IEEE Signal Processing Letters*, vol 11, no 8, pp 663–665, Aug 2004
- [239] A G Dempster and M D Macleod, "Digital Filter Design Using Subexpression Elimination and all Signed-Digit Representations," in *Proc IEEE International Symposium on Circuits and Systems (ISCAS)*, vol 3, Vancouver, Canada, May 23–26, 2004, pp 169–172
- [240] A G Dempster and M D Macleod, "Using all Signed-Digit Representations to Design Single Integer Multipliers using Subexpression Elimination," in *Proc IEEE International Symposium on Circuits and Systems (ISCAS)*, vol 3, Vancouver, Canada, May 23–26, 2004, pp 165–168
- [241] M D Macleod and A G Dempster, "Common Subexpression Elimination Algorithm for Low-Cost Multiplierless Implementation of Matrix Multipliers," *IEE Electronics Letters*, vol 40, no 11, pp 651–652, 2004
- [242] N Boullis and A Tisserand, "Some Optimizations of Hardware Multiplication by Constant Matrices," *IEEE Transactions on Computers*, vol 54, no 10, pp 1271–1282, Oct 2005
- [243] R P sko, P Schaumont, V Derudder, S Vernalde, and D Durackova, "A New Algorithm for Elimination of Common Subexpressions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol 18, no 1, pp 58–68, Jan 1999
- [244] M D Macleod and A G Dempster, "Multiplierless FIR Filter Design Algorithms," *IEEE Signal Processing Letters*, vol 12, no 3, pp 186–189, Mar 2005
- [245] M Martinez-Peiro, E I Boemo, and L Wanhammar, "Design of High-Speed Multiplierless Filters Using a Nonrecursive Signed Common Subexpression Algorithm," *IEEE Transactions on Circuits and Systems—Part II Analog and Digital Signal Processing*, vol 49, no 3, pp 196–203, Mar 2002
- [246] D R Bull and D H Horrocks, "Primitive Operator Digital Filters," *IEE Proceedings-G*, vol 138, no 3, pp 401–412, June 1991
- [247] A G Dempster and M D Macleod, "Use of Minimum-Adder Multiplier Blocks in FIR Digital Filters," *IEEE Transactions on Circuits and Systems—Part II Analog and Digital Signal Processing*, vol 42, no 9, pp 569–577, Sept 1995

- [248] H. Samueli, "An Improved Search Algorithm for the Design of Multiplierless FIR Filters with Powers-of-Two Coefficients," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 7, pp. 1044–1047, July 1989.
- [249] B.-R. Horng, H. Samueli, and A. N. Willson, "The Design of Low-Complexity Linear-Phase FIR Filter Banks Using Powers-of-Two Coefficients with an Application to Subband Image Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 1, no. 4, pp. 318–324, Dec 1991.
- [250] A. Matsuura, M. Yukishita, and A. Nagoya, "An Efficient Hierarchical Clustering Method for the Multiple Constant Multiplication Problem," in *Proc. of the Asia and South Pacific Design Automation Conference ASP-DAC*, Jan. 28–31, 1997, pp. 83–88.
- [251] R. Hartley, "Optimization of Canonic Signed Digit Multipliers for Filter Design," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, Singapore, June 1991, pp. 1992–1995.
- [252] R. Hartley, "Tree-Height Minimization in Pipelined Architectures," in *Proc. IEEE International Conference on Computer Aided Design (ICCAD)*, Santa Clara, CA, Nov. 5–9, 1989, pp. 112–115.
- [253] R. I. Hartley, "Subexpression Sharing in Filters Using Canonic Signed Digit Multipliers," *IEEE Transactions on Circuits and Systems—Part II: Analog and Digital Signal Processing*, vol. 43, no. 10, pp. 677–688, Oct. 1996.
- [254] A. Vinod, E.-K. Lai, A. Premkumar, and C. Lau, "FIR Filter Implementation by Efficient Sharing of Horizontal and Vertical Common Subexpressions," *IEE Electronics Letters*, vol. 39, no. 2, pp. 251–253, Jan. 2003.
- [255] A. Vinod and E.-K. Lai, "Comparison of the Horizontal and the Vertical Common Subexpression Elimination Methods for Realizing Digital Filters," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, Kobe, Japan, May 23–26, 2005, pp. 496–499.
- [256] A. Vinod and E. M.-K. Lai, "On the Implementation of Efficient Channel Filters for Wideband Receivers by Optimizing Common Subexpression Elimination Methods," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 2, pp. 295–304, Feb. 2005.
- [257] A. Vinod and E. M.-K. Lai, "An Efficient Coefficient-Partitioning Algorithm for Realizing Low Complexity Digital Filters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 12, pp. 1936–1946, Dec. 2005.
- [258] D. S. Phatak and I. Koren, "Hybrid Signed-Digit Number Systems: A Unified Framework for Redundant Number Representations With Bounded Carry Propagation Chains," *IEEE Transactions on Computers*, vol. 43, no. 8, pp. 880–891, Aug. 1994.
- [259] D. S. Phatak, S. Kahle, H. Kim, and J. Lue, "Hybrid Signed Digit Representation for Low Power Arithmetic Circuits," in *Proc. Low Power Workshop in Conjunction with ISCA*, Barcelona, Spain, June 1998.

- [260] B Phillips and N Burgess, "Minimal Weight Digit Set Conversions," *IEEE Transactions on Computers*, vol 53, no 6, pp 666–677, June 2004
- [261] S D Andres, "On the Number of Bracket Structures of  $n$ -Operand Operations Constructed by Binary Operations," 2005, private communication
- [262] J Bhasker, *A SystemC Primer*, 1st ed Star Galaxy Publishing, 2002
- [263] H Safiri, M Ahmadi, G Jullien, and W Miller, "A New Algorithm for the Elimination of Common Subexpressions in Hardware Implementation of Digital Filters by Using Genetic Programming," in *Proc International Conference on Application-Specific Systems Architectures and Processors*, July 10–12, 2000, pp 319–328
- [264] H Safiri, M Ahmadi, G Jullien, and W Miller, "A New Algorithm for the Elimination of Common Subexpressions in Hardware Implementation of Digital Filters by Using Genetic Programming," *The Journal of VLSI Signal Processing*, vol 31, no 2, pp 91–100, Feb 2002
- [265] Z Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed New York Springer, 1998
- [266] J Bruton (2005) Intelligent Systems – EE551 Taught M Eng Course Notes Dublin City University [Online] Available <http://www.eeng.dcu.ie/~brutonj>
- [267] D E Goldberg, "A Note on Boltzmann Tournament Selection for Genetic Algorithms and Population-Oriented Simulated Annealing," *Complex Systems*, vol 4, pp 445–460, 1990
- [268] A Kinane, V Muresan, and N O'Connor, "Towards an Optimised VLSI Design Algorithm for the Constant Matrix Multiplication Problem," in *Proc IEEE International Symposium on Circuits and Systems (ISCAS)*, Kos, Greece, May 21–24, 2006
- [269] A Kinane, V Muresan, and N O'Connor, "Optimisation of Constant Matrix Multiplication Operation Hardware Using a Genetic Algorithm," in *Proc 3rd European Workshop on Evolutionary Computation in Hardware Optimisation (EvoHOT)*, Budapest, Hungary, Apr 10–12, 2006
- [270] P Kauff and K Schuur, "Shape-Adaptive DCT with Block-Based DC Separation and  $\Delta$ DC Correction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol 8, no 3, pp 237–242, June 1998
- [271] E Krotkov, "Focusing," *International Journal of Computer Vision*, vol 1, pp 223–237, 1987
- [272] A P Pentland, "A New Sense for Depth of Field," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol 9, no 4, pp 523–531, July 1987
- [273] B Hendriks and S Kuiper, "Through a Lens Sharply [FluidFocus lens]," *IEEE Spectrum*, vol 41, no 12, pp 32–36, Dec 2004
- [274] The On-Line Encyclopaedia of Integer Sequences AT&T Research [Online] Available <http://www.research.att.com/~njas/sequences/Seis.html>