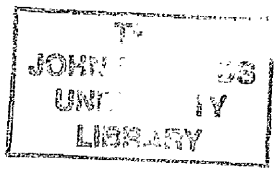
Energy Efficient Functional Unit for a Parallel Asynchronous DSP

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy in the
Faculty of Engineering and Physical Sciences

July 2005

Wannarat Suntiamorntut

School of Computer Science



Contents

Contents 2
List of Figures 5
List of Tables 8
Abstract 9
Declaration 10
Copyright 10
The Author 11
Acknowledgements 12

Chapter 1: Introduction 13
1.1 Motivation 16
1.2 Research Contributions 19
1.3 Organization 20

Chapter 2: Background 23
2.1 Digital Signal Processor 24
2.1.1 How a DSP is different from a general purpose processor? 24
2.1.2 Digital Signal Processor Architecture 25
2.2 A Modern Digital Signal Processor Architecture 27
2.2.1 Important features of DSP processor 27
2.2.2 DSP for next generation mobiles and wireless communication 29
2.2.3 DSP for multimedia signal processing 31
2.3 General Functional Unit for DSPs 32
2.4 Energy Efficient Design 33
2.4.1 Energy Sources 33
2.4.2 Fundamental of low-power design 35
2.5 Low Power Systems 41
2.5.1 Low power design at a technological-level 41
2.5.2 Low-Power design at a circuit-level 43
2.5.3 Low-Power design at a logic-level 52
2.5.4 Timing Approach 53
2.5.5 Low-Power Design at an architecture level 55
2.5.6 Low-Power Design at an Algorithmic Level 58
2.6 Concluding Remarks 58

Chapter 3: Asynchronous Parallel DSP 59
3.1 Background 60
3.2 CADRE-s Top-Level Architecture 61
3.2.1 MIMD with VLIW encoding 64
3.2.2 Five Stage Asynchronous Pipeline Organization 64
3.3 FU Interconnection 67
3.4 Testability 69
3.5 The example of Four-way Parallelism 71

Chapter 4: Energy Efficient Functional Unit - The Adder and Multiplier 74
4.1 Functional Unit Architecture 74

4.1.1 Instruction Set 77
4.1.2 FU datapath 77
4.1.3 A Data Flow 79
4.2 Addition 80
4.2.1 Carry-Look-ahead Tree Background 82
4.2.2 Carry-Look-Ahead Tree Implementation 85
4.2.3 Adder Implementation 86
4.3 Multiplication 88
4.3.1 Background 88
4.3.2 Multiplier Implementation 95
4.3.3 Input Swapping 96
4.3.4 Sign Extension 98
4.4 Multiply Accumulate (MAC) Operation 103

Chapter 5: Energy Efficient Functional Unit - Hamming Distance, Normalization, Timing and Control 105
5.1 Hamming Distance and Normalization 105
5.1.1 Hamming Distance and Normalization Specification 108
5.1.2 Hamming Distance and Normalization Design 109
5.1.3 HD and NORM Results 113
5.2 Others Functions 115
5.3 PTG Shifter 116
5.4 Accumulator 116
5.5 Configurable Memory 117
5.6 Control Unit 118
5.6.1 Asynchronous Control Circuit 118
5.7 Delay Model and Tunable Timing Mechanism 121
5.7.1 Data Encoding 121
5.7.2 Data Dependency 122
5.7.3 Delay Model 123
5.7.4 A Tunable Timing Mechanism 124
5.7.5 Experimental Results 125
5.7.6 Extending the Concept 128
5.8 Voltage Scaling Versus Energy Consumption 131

Chapter 6: Energy Efficient Circuit Design Methodology 134
6.1 Logic Style 135
6.2 Pass Transistor 139
6.2.1 Fan-in and Fan-out of PTG 140
6.2.2 Transistor sizing 142
6.3 Voltage Scaling in Pass Transistor Circuit 145
6.4 XOR/XNOR Design For the Full Adder 146
6.4.1 New full adder using Novel XOR-XNOR Gates 147
6.4.2 Efficiency Evaluation 148
6.5 Energy Efficient 4-2 compressor 154
6.5.1 4-2 Compressor Designs 155
6.5.2 4-2 Compressor Circuitry 155
6.6 Energy Efficient Latch 156
6.7 Layout Consideration 157
6.7.1 Datapath Layout 157

6.7.2 Standard cell layout 160

Chapter 7: Evaluation 163
7.1 Chip Connections 163
7.2 Kernel Benchmarks 164
7.2.1 Digital filters 164
7.2.2 Vector Dot Product 165
7.2.3 Correlation 165
7.2.4 Discrete Cosine Transform (DCT) 166
7.3 Speech / Voice Data 167
7.4 Results 169
7.4.1 Operation Rate 169
7.4.2 Run Time 170
7.4.3 Execution Energy of FIR Filter 171
7.4.4 Instruction Mix and Execution Time 173
7.4.5 Scaling Down the Supply 175
7.4.6 Power Breakdown in the Functional Units 175
7.4.7 Projected CADRE-s 177
7.5 Comparison with other DSPs 179

Chapter 8: Conclusions 184
8.1 Current Directions 186
8.2 Future Research Directions 187
8.3 Conclusions 187

References 189
Appendix A: Instruction set 198
Appendix B: Glossary 201
Appendix C: Simulation Details 202

List of Figures

2.1 An example of FIR operation in a DSP 25
2.2 Typical DSP architecture 26
2.3 Block Diagram of ZSP600 DSP 30
2.4 Improvement in technology[30] 34
2.5 Energy versus voltage scaling 35
2.6 Circuit topology (a) chain structure (b) tree structure 38
2.7 (a) Dynamic logic (b) Dynamic DCVSL (c) Static DCVSL 47
2.8 Pass gates or Pass Transmission Gate (PTG) 49
2.9 Complementary Pass Transistor Logic (CPL) 49
2.10 Energy Economized Pass Transistor Logic (EEPL) 50
2.11 Single-ended Pass Transistor Logic (SPL) 50
2.12 Principle of micro-pipeline 53
2.13 Two possible structures of execution unit-sharing 57
3.1 Average distribution of energy per block in CADRE 62
3.2 Breakdown of power consumption within the FU in CADRE 62
3.3 Top-level architecture for CADRE-s 63
3.4 CADRE-s Data Organization 63
3.5 5-Stage pipeline CADRE-s Organization 65
3.6 (a) CADRE-s 5-stage pipeline operation for a FU. (b) CADRE-s multi-time slot instruction of 5-stage pipeline operation of 4FUs. 67
3.7 Scan-path in a FU of CADRE-s 70
3.8 Test board configuration 71
4.1 Analyse arithmetic instructions in DSP kernels 75
4.2 Top-level functional unit architecture 76
4.3 Functional Unit datapath 77
4.4 MAC instruction flow in FU datapath 79
4.5 Data movement flow in FU datapath 80
4.6 The structure of 4-bit carry-look-ahead tree block 86
4.7 An example of the delay characteristic in an 8-bit CLA multiplexer tree 87
4.8 The adder structure of this FU 87
4.9 An example of the LR and RL in 8-bit parallel multiplier 90
4.10 Carry save adder (CSA) and tree of CSA adder reducing 7 numbers to 2 90
4.11 A general structure of 8-bit parallel or tree multiplier 92
4.12 Energy estimation of serial-parallel and tree multipliers 93
4.13 Determination block 97
4.14 Partial product generator circuit 99
4.15 Eight PPs presented in excel worksheet 99
4.16 Pre-calculated sign extension 100
4.17 Parallel multiplier architecture 101
4.18 The conventional Wallce tree and this addition tree 102
4.19 Average power breakdown of the multiply-accumulator (MAC) 103
5.1 A structure of the parallel comparator and counter 106
5.2 Examples of HD and NORM using 16-b for HD and 40-b for NORM 106
5.3 A basic structure of the HD and NORM circuits 107
5.4 The first stage of HD and NORM operations. 109

5.5 The parallel addition using 4-2 compressor 110
5.6 The population in each step of the NORM operation 111
5.7 The example of the first adjusted component (ADJ5) 112
5.8 The translation stage diagram 113
5.9 The delay and current characteristics of the HD and NORM circuits. 114
5.10 Accumulator register structure 116
5.11 The relation between current and the amount of load in ACC 117
5.12 The control circuit diagram 119
5.13 Bundled Data Asynchronous Structure 122
5.14 Tunable delay principle 124
5.15 (a) a tunable timing mechanism (b) voltage level shifter circuit 125
5.16 Simulation of the timing versus scaling voltage 126
5.17 Plot of the theoretical and simulated delay 127
5.18 Matched delay on different logic families 127
5.19 The relation between delay and voltage scaling of various delay lines. 128
5.20 Different Matched Delay Selection 129
5.21 A simple 4x4 multiplier for tunable timing mechanism evaluation 129
5.22 Relation between the normalised speeds of timing delay circuits for random input data 132
6.1 A test environment for the circuit simulation 135
6.2 Different logic families of XNOR circuits 136
6.3 The relation of energy and load capacitance of various XNOR 137
6.4 2-input multiplexers based on different type of pass transistor 139
6.5 The behaviour of the pass transistor 140
6.6 Fan-in and fan-out of pass transmission gate 141
6.7 The relationship between normalized delay and electrical effort (h) of PTG 141
6.8 PTG multiplexer 143
6.9 PTG 2-1 multiplexer results 144
6.10 Effect of supply voltage on energy delay product in an 8-bit ripple carry adder 145
6.11 The relation between leakage current and scaling voltage of the CMOS and PTG ripple carry adder. 146
6.12 XOR with 3 transistors 148
6.13 New full adders 149
6.14 Test structure for 1-bit full adder. 150
6.15 Architecture of the 4-2 compressor 155
6.16 A proposed 4-2 compressor 156
6.17 A pass transmission gate latch 157
6.18 Functional unit datapath floor-plan 159
6.19 A 4-2 compressor layout in the multiplier showing cell abutment 159
6.20 Layout format for 1-bit cell in full custom functional unit datapath 160
6.21 Layout format for standard cell library (Amust) 161
7.1 CADRE-s Die Photo 164
7.2 Sample of transforming spatial domain of pixels to DCT domain 167
7.3 The proportion of execution and download energy in CADRE-s 173
7.4 The ratio of energy per instruction compared to shift instruction 174
7.5 Breakdown power consumption of the functional unit 177
7.6 Break down average power of the arithmetic blocks in the FU 177
7.7 Comparison of distribution of energy throughout the original CADRE

and the projected CADRE-s 179
7.8 Distribution of percentage energy throughout the Original CADRE versus the projected CADRE-s 180

List of Tables

3.1 Mapping a arrangement FIR 20-tap filter (n = 0, 1, 2, 3) onto four FUs 72
3.2 A different data arrangement for the FIR 20-tap filter (n = 0, 1, 2, 3) mapped onto four FUs 73
4.1 The comparison of 40-b adders 82
4.2 Modified Booth's recording 91
4.3 Truth table of 4-2 compressor 94
4.4 Partial Product Generator 96
4.5 Power dissipation the different dynamic range inputs (multiplicand is constant) 97
4.6 Comparison of energy delay product of multiplier 103
6.1 Power and performance of 2-input multiplexers based on different type of pass transistor 138
6.2 Size of transistor of 2-1 PTG multiplexer 142
6.3 Six different input patterns 150
6.4 Simulation results for the full adders at 1.8 V without output load 150
6.5 Simulation results for the full adders at 1.8 V with output load 151
6.6 Simulation results for full adder sub-set with 8 inverters as the output load at different supply voltages. 152
6.7 Simulation results for full adder sub-set with output load (8 inverters) with the 6 patterns of the sequence inputs 153
6.8 Energy saving of PTG XOR: standard cell versus full custom 158
6.9 Metal usage summation for full custom datapath 160
7.1 Parallel instruction rates and operations per second 170
7.2 Power and energy consumption of CADRE-s operated at 1.8V 170
7.3 Break down average power, download and execution time of the CADRE-s 172
7.4 Instruction breakdown for the benchmark programs 174
7.5 Energy consumption during the execution time of CADRE-s a 1.6V power supply 175
7.6 The average power consumption during the execution of the FU in CADRE-s 176
7.7 Estimated energy consumption of using the FU in the original CADRE architecture 178
7.8 Performance and power factor when process technology scaling to 0.18mm@1.8V 181
7.9 Power per million instruction (mW/MHz2) of the commercial fixed-point DSPs 181
A.1 Top-level instruction specification 198
A.2 Opcodes 198
A.3 Configuration instruction specification 199
A.4 CADRE-s chip pad pin (68-pin PGA package) 200

Abstract

In the last few years, the mobile market has grown extensively. The next generation of these 3G or 4G devices require much higher performance to operate the more complex algorithms required from increased functionality such as multimedia, voice recognition and internet access. As a consequence, the number of transistors required increases dramatically. Meanwhile, the energy density of existing battery technologies is not increasing at the same rate. Thus in order to have a longer battery operating time, energy efficient computing in a DSP assumes greater emphasis.

CADRE was designed and expected to be a minimum power consumption asynchronous DSP whilst meeting the performance requirements of next generation cellular phones. In the original CADRE, most of the research was focused on the algorithmic and architectural level design and resulted in a good energy economical design at these levels. However, the power dissipation of the original CADRE showed that approximately 50% of the overall power consumption was found to be dissipated in the functional units (FUs). Thus, reducing the power consumption of the FU was the primary motivation for the work described in this thesis.

The FU has been re-designed focusing on improving the energy efficiency at the logic, circuit and layout levels. Coherent design techniques have been applied. These include the use of pass gate logic, the sharing of the adder between the multiplier and adder, a combined logic block for performing the Hamming distance and normalization function and a new timing mechanism for better tuning the asynchronous control to the datapath. These functions reduce the amount of logic and possible failure of the system.

CADRE-s has been implemented as a full custom design and simulations are presented to successfully demonstrate the energy-efficiency of the FU. The results show that the FU designed can achieve an energy improvement by a factor of 5 in the multiply accumulator units and a factor of nearly 2 for the overall system compared with the original CADRE system. This demonstrates the importance that energy efficient logic, circuit and layout techniques contribute to a design.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification at this or any other university or other institute of learning.

Copyright

The Author

After being born in the south of Phuket island, Thailand, Wannarat grew up and was educated by many wonderful teachers at many schools around Phuket city. She learnt about microelectronic design at King Mongkut Institute Technology of Ladkrabang (KMITL), one of the top three Thai universities in engineering. After joining the embedded system laboratory of KMITL for 2 years, she decided that the most interesting career would be as a researcher doing a fantastic microelectronic inventory for the country. However, being a researcher in Thailand was limited by funding sources. Thus she became a private research assistant for the laboratory at the Institute. Later, she was encouraged to do a master course in computer engineering at Chulalongkorn University which is the top Thai university in engineering. She spent 18 months in the microprocessor verification field, obtaining a master degree. She became a lecturer at Prince of Songkla University, returning to do research on microelectronics and also to be a member of the national IC design group.

Her main responsibility was teaching and developing the microelectronic laboratory for the 3rd year of the undergraduate computer engineering course. After spending a year setting up the machines and EDA tools, a Microcontroller and FPGA hardware laboratory were established for the third year course. A project of a talking chip using a FPGA, implemented by a senior student under her supervision received a third place award in the national microelectronic contest. In addition other projects such as a non-coding robot implemented on Complex Programmable Logic Device (CPLD), design and implementation of an ARM7 core, and a network protocol soft-core using VHDL, were developed during her time at the Prince of Songkla University. She joined the APT group at The University of Manchester in 2002.

Acknowledgements

I would like to thank my supervisor Dr. Linda Brackenbury to give me a great opportunity and wonderful support both in term of technical questions and English. I also would like to thank my advisor Dr. Jim Garside for the circuit design inspiration and answers to many of my technical questions.

Special thanks go to Mr. Dave Clark for being a wonderful help with many technical problems, especially his brilliant support on the layout and Mr. Jeff Pepper who maintained and solved the Cadence and other CAD tools problems through adversity and gave help whenever difficulties were encountered.

Many thanks to everybody who has helped by proof-reading my thesis, especially Mrs Wendy Warburton who spent her time to correct my English. Thanks also to Dr. Andrew Bardsley and Dr. John Bainbridge for their useful comments on the thesis.

Thank you to the whole APT group who have somehow managed to put up with my silly ways and also badminton players who shared great fun with me.

Thanks to my parent who know nothing about microelectronic but are wonderful advisors of all kinds.

Thanks to Apinetr Unakul for his wonderful advice and inspiring me to do research work in this field when I was in Thailand.

The work presented in this thesis was funded by the EPSRC, grant number GR/N39159/01 and GR/561270/01. The author is grateful for this support.

Chapter 1: Introduction

In the last few years, the number of applications of portable, battery powered electronic equipment has grown considerably. Pocket PCs, handheld PCs, hearing aids, portable military equipment and mobile phones are good examples. Nowadays, the third generation (3G) handsets serve as not only voice communication equipment, but also include data, image, multimedia and other complex functions. Furthermore, the next generations of mobile equipment will require even higher performance to operate more complex systems and to support the growing trend towards portable computing and wireless communication.

The enormous improvements in technology allows many functions to be integrated onto a chip. Thus, one chip can support an expanding range of functions, and multiple devices can be placed into a single unit [1]. The improvement in technology has opened up many possibilities for current and future mobile systems to expand functions. Future mobile systems will comprise a small personal portable computer and wireless communication device. An important feature will be the interface to and interaction with the user. Thus speech and pattern recognition will be key functions. Real-time multimedia data such as video, speech, and music will improve the productivity, usability, quality and enjoyment of future mobile systems. These require not only a high speed computation, but also a significant amount of computing power. Therefore, one of the most compelling issues in portable computing is to keep the energy consumption of the devices low in order to have a longer battery operating time.

High power consumption means a short battery life-time of the mobile application. In addition, high consumption causes a heat problem affecting the reliability and life-time of circuits as these depend on the power consumption. So high power consumption will become more critical in future because of the future mobile system requirements. In

addition, battery weight and life time are becoming more important than processing speed due to various microprocessors now running well in excess of 10 GHz. Thus in portable systems, energy consumption is the limiting factor in the amount of functionality that can be operated. For example, speech or hand writing recognition functions on an embedded system board needs about 20W to realize 20,000 words[2]. Unfortunately, the nickel-cadmium battery technology can provide only 35-57 Wh/Kg (Watt-hour per Kg.)[3]. Thus systems have to be designed to be more efficient in the way they use the energy. Consequently, energy efficiency can be informally defined as doing more work with the same amount of energy resource or doing the same work with less energy. This is the approach needed for portable systems having a limited energy budget.

Most portable systems have a Digital Signal Processor (DSP) operating as the main processing core. DSP programs are typically tight loops of arithmetic operations with fewer branches than general purpose code. Furthermore, DSP algorithms are used in real-time applications with different sampling rates. These require processing speeds from 20 MHz to over 500 MHz depending on their applications. Therefore, a significant challenge in the implementation of a DSP for portable systems is how to use the limited power source efficiently.

Using voltage scaling, low power circuit design and architectural modifications, such as parallelism and pipelining, can reduce the power dissipation by more than 100 fold as demonstrated in custom DSP ASICs[4]. With some low power processors widely used in portable devices[5], this has been at the expense of accepting a lower performance. Therefore, those processors achieve low power operation by running slowly. However, it is not suitable for a DSP application which requires both a high performance and power efficiency.

Another good example of a low power processor, which may not be the highest energy efficient processor for mobile systems, is the StrongARM[6]. [7] compares StrongARM with other contemporary processors because ARM designed StrongARM with energy consumption in mind from the start. However, StrongARM cannot achieve an energy efficient processing core in mobile systems because of the many additional features it includes. For this reason, the ARM7[6] was introduced and widely used in mobile systems thereafter. ARM7 also has many features such as branch prediction, a 16-bit thumb

decoder and so on, which are not required for signal processing systems. This is a large power overhead for a signal processor. Nevertheless, a DSP could take advantage of the superscalar and/or pipelined micro-architecture feature from the StrongARM or ARM7 processor to achieve an energy-efficient improvement. The ARM9E[6] was introduced to DSP systems with enhanced DSP functionality in response to the growing demand for greater signal processing capabilities. The need for a separate DSP can potentially be eliminated by adding DSP functionality into an embedded general-purpose processor such as the ARM9E. However, its ability for keeping its execution units supplied with operands, is limited by the lack of parallel move support.

Contemporary DSPs exploit parallelism by being organised as single-instruction multiple-data (SIMD), multiple-instruction multiple data (MIMD) or very long instruction word (VLIW) architectures. This achieves higher performance irrespective of the clock. These architectures were introduced to exploit the regularity of DSP algorithms. The motivation for these designs was to avoid high clock frequencies. This was because increasing the clock frequency is limited by technology and an increased frequency raises the power consumption in a circuit as power is proportional to frequency. Also, high frequencies usually require more complex logic to meet the clock speed. Therefore, parallel architectures have advantages for future mobile applications, especially in terms of energy efficiency.

VLIW processor chips have been used in non-portable commercial applications; for example, the TI TMS320C6000 DSP[8] is used in 3G base stations, in the Digital Subscriber Line (DSL) access multiplexers and in network concentration units. An alternative, attractive architectural approach employs a parallel 2D array of hundreds of small processors and used in (non-portable) applications such as image and video processing. This has been implemented by the PACT (Processor Array Computing Technology) Company who have been developing an eXtreme processor platform called XPP [9]. The power consumption and performance are set to rise exponentially in the future. Therefore even those architectures which are currently energy efficient will become high dissipation devices in the future. Hence energy efficient architectures are required and even more is the subject of this and other research.

The emerging trend for wireless based stations and voice channels in telecommunications systems are reconfigurable architectures and there is a small group of companies such as Elixent Ltd., Morpho Technologies and Quick Silver Technologies, who are all working on reconfigurable FPGA (Field Programmable Gate Array) Architectures. A reconfigurable DSP was introduced in some research work[10],[11] to achieve the system-on-chip (SOC) concept; this is rapidly becoming a reality, with the time-to-market and product complexity promoting the reuse of complex macromodules. As an alternative, an application domain specific architecture for digital signal processing, the Field Programmable Function Array (FPFA) has been employed[12]. Unfortunately, the draw back of this reconfigurable architecture over ASIC designs is their energy efficiency even though they are more flexible than ASICs.

1.1 Motivation

Even those architectures which claim to be energy efficient digital signal processing components will not meet future needs as the power consumption is set to rise exponentially as will the performance required for portable applications. Therefore, these processors will have high power dissipation when future computing is considered. There is much research[13],[14] in this area resulting in architectural structures which are energy efficient. For examples, a novel asynchronous parallel architecture, named CADRE[14] has been designed in the School of Computer Science, University of Manchester. CADRE has expanded instructions to give a flexible VLIW capability including a large register file, instruction buffer and four functional units. It has been designed based on the sign and magnitude number representation and asynchronous circuit design. The compressed instruction was exploited to reduce long instructions. Even though CADRE has been designed to be an efficient CPU architecture for DSP processors, it requires a large amount of power as demonstrated in the original CADRE power results[14], [72].

Similarly, reducing the number of multiplications for general DSP algorithms such as finite impulse response (FIR) filters, infinite impulse response (IIR) filters, or fast Fourier transforms (FFT), has been developed to improve the energy efficiency[15]. Results for these show that whilst algorithm organization can make some contribution to power efficiency, the power consumption is still large. Moreover, developed or modified

algorithms cannot save the power dissipation if the DSP is implemented on power-hungry circuits, especially in the multiplier and adder, which contribute to a big part of the power consumption in a DSP. Therefore, another research angle to reduce and minimize the power budget is logic and circuit optimization with full-custom layout also contributing to a large part of the energy efficiency.

As is well-known, arithmetic operations, such as multiply or multiply-accumulate are frequently performed and are power hungry in the functional unit (FU) of a DSP. Therefore, this thesis looks at the functionality that a FU needs to perform and makes a low power version. This will then be compared to the (scaled) original design of the CADRE FU to demonstrate the significant power saving realised by low power logic and circuits. Whilst this is a study of a FU for a DSP, the techniques used are generally applicable to all digital logic gate design, yielding similar improvements elsewhere.

The CADRE FU has been re-designed based on two's complement number representation; the circuits for sign-magnitude computation were found to be more complex and larger than those used to compute in two's complement format because an extra circuit was required to produce a precise sign bit result. CADRE performed reasonably in the overall comparison to other DSP's as shown in [72]. CADRE's greatest benefit stems from the fact the complex algorithms can be executed efficiently by the parallel architecture through the use of compressed instructions and the register file. According to the requirement of the third generation of wireless communication, the digital signal processing circuit needs high computational performance, low-power dissipation and a high degree of flexibility. Embedded configurable memory within a FU offers the advantage of combining flexibility and low-energy by providing control signals mapping directly from algorithm to hardware. The encoding instructions are stored in a configuration memory which can be reconfigured by storing the instructions in advance at the start of each algorithm. This reduces the control overhead associated with instruction-set processors. Philips REAL DSP[16] and the Infineon CARMEL DSP[17] have embedded this feature as a single global configurable memory which is only used for special instructions. As a result, the flexibility and performance of the FU is limited. Therefore, an embedded configuration memory in each FU gains the user increased flexibility and performance.

The challenge in the work described here is to meet the requirements of future portable applications, such as multimedia mobile phones. These devices need a very small power budget, higher performance with an increased complexity approaching that of a desktop computer. This can be achieved by adopting a parallel architecture. However, parallelism without dropping the supply voltage leads to increased area and power. This can be reduced by trading any excess performance for power by reducing the voltage. The next challenge is therefore to increase the performance if the voltage is scaled downwards. Parallel FUs give increased overall speed and are relatively easy to exploit in DSP applications. Throughput is roughly proportional to supply voltage, so four-way parallelism allows a voltage reduction of up to four times. This translates to a 16 times power reduction per FU (corresponding to CV^2) or 4 times overall compared with a single FU. Thus, reducing the power dissipation in a FU is the most effective way of power saving in a parallel DSP.

An energy efficient configurable DSP architecture framework with four power efficient FUs has been designed and implemented to demonstrate a high energy-efficiency DSP. Algorithms such as FIR filters and the FFT, have been transformed to parallel assembly codes and then mapped onto the framework. The binary codes are generated by assembler and then stored in each configuration memory at the start of algorithm. Self-time completion detection circuits in FUs are employed because of using a globally asynchronous framework. Additional advantages of this framework that arise from asynchronous timing are power saving, higher performance and low electromagnetic interference.

To support a power efficient design of FU, the major components such as the multiplier, adder, shifter, Hamming distance and normalization need to be optimized. In addition to logic optimization for these component, the logic is implemented with energy efficient circuits using the smallest number of transistors. Thus the FU is aimed at the next generation of wireless and portable applications, and should achieve the ultimate in power efficiency.

1.2 Research Contributions

The goal of this research is to significantly improve the power efficiency of a FU for DSPs and to demonstrate this by combining four FUs running in parallel with configurable memories as well as a demonstrator framework. Several key research contributions are:

- Demonstration of the energy-efficient architectural framework comprising four asynchronous FUs and data memories for executing digital signal processing algorithms.

- Developing a flexible FU for the user and system programmer. This has led to additional cost, particularly in the implementation of the configuration memory. This cost can be justified by the flexibility and performance gained by users. In particular, the use of configurable memory embedded in the FU is unique compared with other DSPs such as REAL and CARMEL. This feature also allows each FU to operate independently on different instruction streams.

- Showing that lowering power at the logic, circuit and layout makes a large contribution to overall power levels. The architectural framework will be fabricated on 0.18 um operating at 1.8V to evaluate the parallel DSP approach.

- Developing a large power improvement on an unusual low power FU datapath particularly suitable for use in DSP's aimed at portable applications. This datapath is an extended knowledge component combining many good low power circuit design techniques in each component. This leads to high performance and keeps the power dissipation of the FU low.

- Developing the FUs ability to keep the execution units supplied with operands by supporting parallel movement.

- Developing a technique to reduce the switching activities and the number of stages of addition in the multiplier by using a parallel wallace tree structure and sharing the final addition circuitry between the adder and multiplier. The use of a shared adder has not previously been described.

- Considering addition and multiplication schemes for an energy-efficient DSP component.

- Developing a combined novel circuit for performing the Hamming distance and normalization functions.

- Designing a novel tunable timing mechanism to better tune the control logic to the data path.

- Developing the power-efficient and novel circuits necessary for a FU. In addition, the comparison of circuit families at 0.18 um geometry operates at 1.8 V. is presented.

- Investigating the power and energy efficiency when voltage scaling is applied.

- Identifying the contribution the design can make to future designs. The implementation can be used on future Super-scalar Asynchronous DSP architectures. This contribution is aimed at next generation designs requiring a high performance and low power.

1.3 Organization

Chapter 2 presents the principles of the hardware and algorithms for digital signal processing, and describes current and future general-purpose DSPs. The advantages and disadvantages of parallel general-purpose DSPs in terms of energy efficiency are discussed here. The features of DSPs for next generation mobile, wireless communication and multimedia applications are presented. This chapter also presents the sources of power dissipation and an overview of low power design techniques for CMOS circuits. How to minimize the power consumption in CMOS circuits is discussed here. The comparison of various logic families such as conventional CMOS, different pass transistor circuits and dynamic logic will also be given in this chapter. Finally, the chapter briefly looks at low power design at the different levels in the design hierarchy.

System design is presented in Chapter 3. The original CADRE architectural features and power results from simulation are discussed here. This chapter shows that despite good

design at the top levels, power efficiency is disappointing and the reason for this is that logic and circuits need optimizing; power consumption at circuit and logic level can be a large contributory factor to power. This chapter also presents an overview of the FU proposed in this research. The FU comprises a sophisticated arithmetic unit. The basic top level operation of the FU is demonstrated by the multiply-accumulate shifting with rounding instruction to show concurrence of many execution paths within each FU. The significant power consumption of the original CADRE multiplication and addition operations are given.

Chapter 4 presents the design of a low power high performance multiplier and adder. Different multiplier architectures are described. The strengths and weakness of these architectures for digital signal processing are discussed and concludes with the choices made for the new FU. The multiplier employs logic optimization. Techniques used in the multiplier such as input selection, pre-sign extension calculations, balanced inputs, completion detection, compressor circuits, partial product generator, Wallace tree reduction and forwarding of the carry signal scheme are explained. The implementation techniques for mapping both in 2-1 multiplexers and XOR/XNOR circuits efficiently onto Pass Transmission Gates are described. This chapter also presents the structure of the 4-input adder design which makes multiplication simpler but adds complexity to the adder which now has a 4-2 compressor at its front end. The comparison with 2-input adder structures are discussed. The carry look-ahead tree adder architecture is explained along with the timing analysis through the tree structure.

Chapter 5 presents the other necessary operations of the function unit such as the Hamming distance and normalization. The other components of the FU such as the shifter, accumulators, accumulate-shifter, limit circuits and control circuits including the timing tunable mechanism are described.

Chapter 6 presents an investigation into the most energy efficient logic family and low power circuit; as a result, Single-ended Pass transistor Logic (SPL) and Pass Transmission Gate (PTG) are identified as the most promising for energy efficient circuits. Chapter 6 also describes the investigation results from energy efficient circuits such as a novel XOR and pass-gate latch which are employed in the functional unit. The logical effort,

transistor sizing, dual supply voltage and layout design techniques are also presented for the design.

Chapter 7 presents the tests performed and the evaluation of the design. The floorplan of the whole chip and its area is given. Algorithms such as FIR, DCT (Discrete Cosine Transform) and simple vector dot product programs run on the FUs for testing are discussed as well as how to map algorithms to the FUs in an optimized way. Performance and power results of the FUs from running these algorithms are given and compared with the (scaled) original CADRE. Multiplier and adder results are discussed together with other research work.

Finally, Chapter 8 summarizes the contributions of the presented work and proposes directions for future research.

Chapter 2: Background

The demand for portable computing and communication devices is growing exponentially. Increasingly such devices contribute to the improvement to the quality of life and are used to increase business productivity. Laptop computers, personal digital assistants and mobile phones are well-known examples of those portable and communication devices which provide the ability to process multimedia information and the ability to communicate information over a wireless communication channel. Digital Signal Processing is the key element underpinning the processing of multimedia information e.g. speed, audio and video.

As is well-known, a general purpose processor is not well-suited to signal processing, control system applications and vice versa. Thus, the Digital Signal Processor (DSP) has been specifically developed for signal processing applications[18]. A DSP is targeted for use in arithmetic computation and contains many power hungry units such as a multiplier, adder, shifter and accumulator registers. These circuits are usually implemented and embedded in a block called a Functional Unit (FU). Therefore, reducing energy in a FU used in battery powered application would enable a DSP to operate within its limited energy resource.

This chapter describes the basic structure of a DSP, distinguishing between the characteristics of a general purpose processor and a signal processor. A modern DSP architecture and design techniques are then discussed in terms of energy efficiency. Then energy efficient design techniques ranging from the circuits and transistor level up to the architecture and algorithm level are explained in this chapter.

2.1 Digital Signal Processor

Nowadays, Digital Signal Processing is widely used in a variety of applications such as IP telephony, radar, audio, digital TV, multimedia and handsets. This processing analyses information which comes from the real world as signals and converts these into digital numbers. DSPs are microprocessors specifically designed to handle digital signal processing tasks. Therefore, the characteristics of a DSP are different from a general purpose processor. In a DSP, one or more outputs, $Y_i[n]$, can be produced for n =..., -1, 0, 1,... and i = 1...N corresponding to one or more discrete-time inputs, $X_i[n]$. The samples of input signals are quantized to a finite number of bits which is either fixed-point or floating-point. Generally, a DSP has to perform many millions of operations per second[19] and hence requires a large memory bandwidth.

Generally, a DSP can be classified into two types; the first type is for a specific application DSP, such as FFT-type (Butterfly operation), and the second type is a general purpose DSP. Both types of DSPs can be implemented as a programmable DSP by using a FPGA or implemented as a hardwired ASIC DSP. However, an ASIC DSP can offer a faster operation time and less power dissipation than a programmable DSP.

2.1.1 How a DSP is different from a general purpose processor?

Since the 1960s, many processors have been developed for use in artificial intelligence equipment. However, the capabilities of those processors can be categorized into two areas; data manipulation and mathematical calculation. The data manipulation concerns storing information or organizing information such as word processing, database management or operating systems. The processor running this kind of task is occasionally used in mathematic calculation. In comparison, digital signal processing is concerned directly with mathematical calculations especially multiplication and addition. A common DSP algorithm is a FIR digital filter as shown in Figure2.1, where 'x' is the input signal and the output signal is 'y'. The input signal is multiplied by a group of coefficients, a0,a1,... and then the products are summed. A group of coefficients is a filter kernel on the input signal. The coefficients will depend on each application.

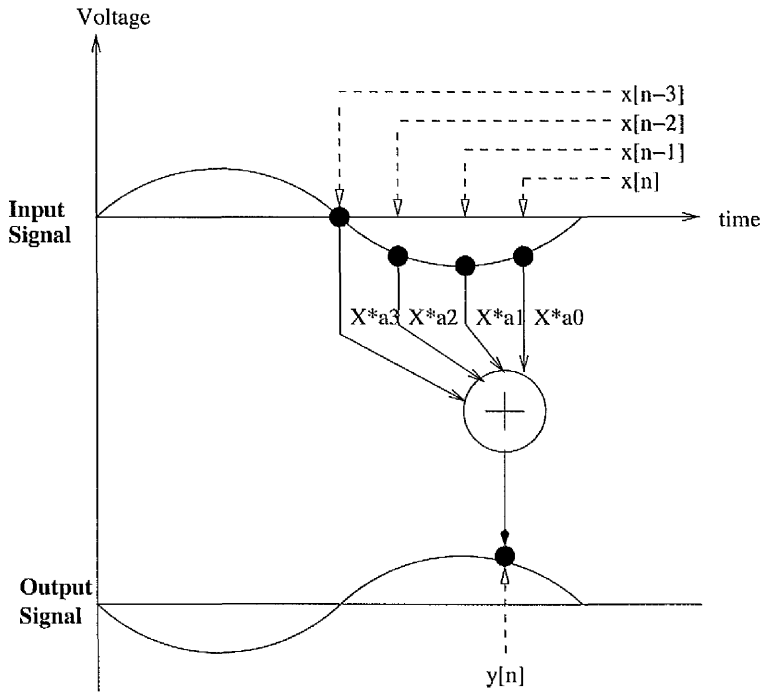


Figure 2.1: An example of FIR operation in a DSP

In addition, DSPs require continuous processing and the timing depends on the sample rate to maintain a sustained throughput. In contrast, the operation of a general purpose processor to perform a word processing job may take two milliseconds or half of a second. This is because users will wait until the job is finished and then assign the next tasks. The most important aim is therefore to make DSPs run faster. Unfortunately, increasing the speed consumes more power and the design becomes more difficult and complex. Particularly, portable applications require a high performance and low power consumption.

2.1.2 Digital Signal Processor Architecture

A traditional microprocessor architecture called *Von Neuman[17],* contains a single memory and single bus for transferring data in and out of the processing unit. Obviously, a multiplication needs at least three clock cycles to get the instruction and two operands. The *Harvard Architecture* is therefore employed to improve the performance of the processor. With a Harvard Architecture, the memories are separated for each data operand

and program instructions, each memory has its own buses. Nowaday, most DSPs use this architecture to improve the speed. However, modern applications require a higher performance from a DSP. Analog Devices therefore invented the *Super Harvard Architecture[21]* by adding features such as an instruction cache and I/O controller to improve the throughput in their SHARC DSPs.

Figure2.2 illustrates a Central Processing Unit (CPU). Within the CPU, a block such as a *Data Address Generator (DAG)* controls the address sent to the data memories. In a DSP, it is designed to be performed with *circular buffers* to avoid the overhead of keeping track of how the data is stored. In addition, *bit-reversed address* mode can be selected by the DAG to efficiently carry out the FFT algorithm. Another block in the CPU is for mathematical processing and this mainly consists of a multiplier, arithmetic logic unit and shifter. Instruction execution can operate with fixed-point or floating-point numbers. With the fixed-point format, the programmer needs to understand the amplitude of the numbers, the accumulation of quantization errors and what scaling needs to take place. There is no need to worry about these factors in a floating-point format. However, there is benefit to using fixed-point hardware rather than floating-point hardware for DSP development because many DSP applications require low-power and cost-effective circuitry, which makes fixed-point hardware a natural choice.

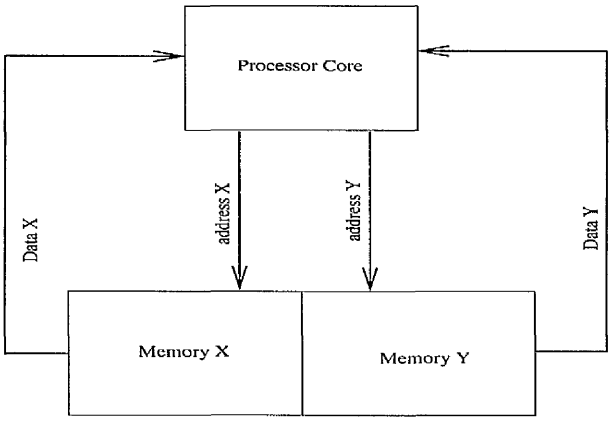


Figure 2.2: Typical DSP architecture

In a DSP processor, the programmer explicity controls which data and instructions are stored in the on-chip memory bank. The programmers have to write programs so that the processor can efficiently use its data and its instruction memory. In contrast, general purpose processors (GPPs) use control logic to determine which data and instruction words reside in the on-chip cache, a process that is typically invisible to the programmer. From the GPPs programmer's perspective, there is generally only one memory space for both data and instruction rather than two memory spaces of the Harvard architecture. Most DSP processors do not have any cache, they use multiple banks of on-chip memory and multiple bus sets to enable several memory accesses per instruction cycle.

DSP processors often support specialized addressing modes that are useful for common signal processing algorithms. Examples include circular addressing (which is useful for implementing digital-filter delay lines) and bit-reversed addressing (which is useful for performing a commonly used DSP algorithm, the fast Fourier transform). These specialized addressing modes are not often found on GPPs, which have to instead rely on software to implement the same functionality.

2.2 A Modern Digital Signal Processor Architecture

It becomes obvious that exploiting parallelism to speed up the processing is necessary in high data rate applications. The parallelism can be achieved by implementing different algorithms onto the different hardware components. However, different designs are required if the applications are changed. Another parallel approach is to employ a processor with a parallel datapath. By using this approach, different algorithms can be performed on a common architecture.

2.2.1 Important features of DSP processor

The important features of DSP processors are repetitive numerical intensive tasks and special instruction sets to exploit hardware efficiency. Two of the most important features of modern DSP processor architecture aimed at low power are the datapath containing an energy efficient multiply-accumulate unit(s) and an energy efficient multiple-access memory architecture.

Datapath

From the reason for using fixed-point given in section2.1.1, only a fixed-point DSP processor is discussed in depth in this thesis. The datapath typically incorporates a multiplier, adder, shifter, logical unit and other specialized functions such as Hamming distance and normalization. Multiplication is an essential operation and frequently used in DSP applications. In many DSP algorithms, multiplication is followed by summing the product with the sum of previous products. Therefore, most DSP processors integrate the multiplier with an adder so that the multiply-accumulate operation can be performed with one instruction. However, multiplication itself is based on an addition generally implemented using XOR and multiplexer circuits.

A shifter is usually required in a DSP processor for two purposes. One is to shift by any number of bits and is used to scale the result of the multiplier and adder which tends to grow in bit width. The shifter for this purpose can be implemented by multiplexers.

The other shift unit (a single bit shifter) involves dealing with overflow and saturation. When the magnitude of the sum exceeds the maximum or minimum value that can be represented in an accumulator register, the quantity overflows. Scaling down the result or saturation is required to handle the overflow situation and so yield a correct value.

Memory Architecture

The memory architecture including its interconnection with the DSP processor's datapath is important for the datapath itself. Consider the N-tap FIR filter. N multiply-accumulate operations are required to produced the new output every N instruction cycles. To achieve this performance, the memory needs to be accessed several times within one instruction cycle.

Typically, three memory banks, comprising one program bank and two data memory banks, are common in current DSP processors. With this feature, it is possible to perform a 1-tap FIR filter operation per instruction cycle. However, using a dual-ported memory which provides two simultaneous read operands only requires one operand memory bank. Therefore, some DSP processors use one single-ported program memory with one dual-

ported data memory. Finally, register-indirect addressing modes are used. Most DSP processors have an address generation unit (AGU) dedicated to address calculations. An AGU is used to calculate complex addressing in parallel with arithmetic operations.

A modern DSP processor has multiple functional units and multiple-access memory banks. Therefore special instructions are needed to specify the several simultaneous operations performed by the different functional units in one instruction cycle. Encoding instructions so that operations are packed into one instruction can also reduce the program memory size.

2.2.2 DSP for next generation mobiles and wireless communication

Currently, multiple time slots and processing in parallel are employed to speed up the throughput. Additional features such as MP3 and speech processing in handsets requires a high performance DSP whilst the power consumption needs to be low. A highly parallel architecture with Very Long Instruction Word (VLIW) architecture can increase the performance. Adelante Technologies produced a DSP called Saturn[23] with these properties. It has a highly parallel 96-bit VLIW architecture that is accessed using encoded 16-bit instruction words. Saturn was demonstrated using Viterbi butterfly operations. Normally, it takes about 11 clock cycles to complete in regular assembly code but in Saturn it takes only 2 clock cycles with two application specific instructions. This is a factor of 5.5 performance improvement. Unfortunately, gains in the power consumption have not been realised yet in this design.

LSI logic[24] presented the second-generation (G2) superscalar ZSP DSP architecture which is able to enhance the instructions per cycle (IPC) performance by nearly three times that of current architectures running various benchmarks. The ZSP600, as shown in Figure2.3, uses a Prefetch unit (PFU) that can prefetch eight 16-bit words per cycle. The instruction cache inside the PFU needs to have the data required by the instruction sequencing unit (ISU) for any given fetch cycle. The ISU has the responsibility for the instruction fetch and decode, instruction grouping, and instruction issue. The ZSP600 has the capability of issuing up to six instructions per cycle to each of the six primary datapaths each comprising: two Address Generation Units (AGUs), two ALUs and two

multiplication/arithmetic units (MAUs). The core of the multiplication and arithmetic unit is multiply accumulate (MAC) logic. This MAC operation is fundamental to most DSP algorithms where successive products are computed and summed. Each of the primary datapaths are controlled by the pipeline control unit (PCU). The interrupt controller, the co-processor and debug interface and timer are also controlled by the PCU.

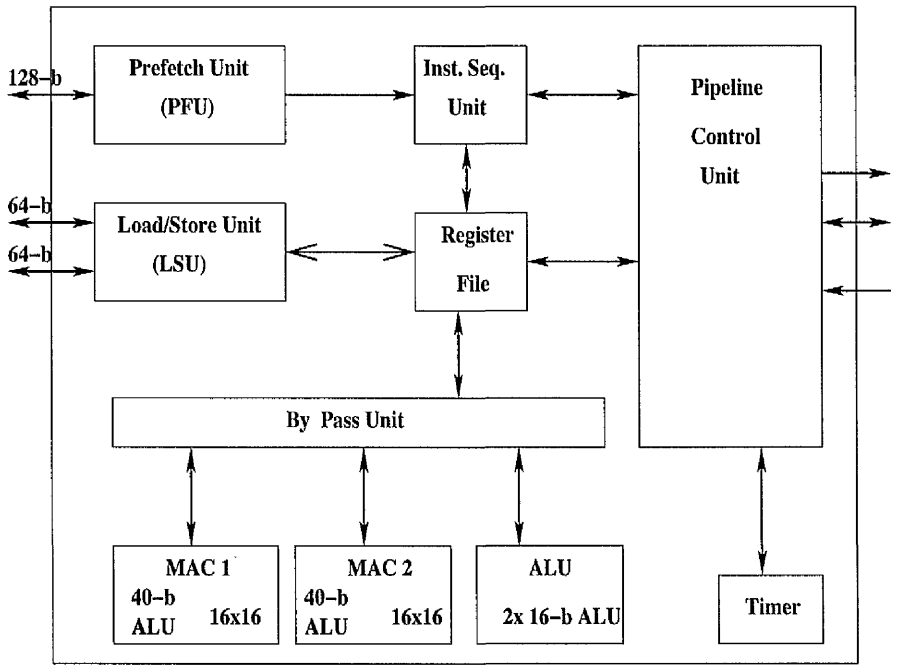


Figure 2.3: Block Diagram of ZSP600 DSP

The ZSP600 has a quad-MAC implementation. A dual-MAC datapath in a MAU allows two 16-bit MAC operations or one 32-bit operation to execute per cycle. Therefore, each MAU can accumulate the two 16-bit multiplications into a single accumulator or into separate accumulators.

In terms of power improvement, the ZSP600 has many levels of power saving. The first level is to control the power consumption by an instruction to idle the core, so the core is effectively shut down if it is not being used. An interrupt is used to wake the core when needed. The second level of power saving is to shut down any part of the primary unit by using the clock control unit; this can prevent unnecessary dynamic power dissipation. The

last level of power reduction is at the register level, allowing control logic within the ALU to determine the operations. So only the appropriate logic is active in the datapath. This prevents switching activity in the unused parts of the datapath. The design can achieve 300 MHz operation on a 0.13μm geometry. However, the ZSP600 core is available to customers as a licensable RTL design, no custom logic is used in any part of the design. Unfortunately, there is no detailed report of the power reduction achieved, only the performance improvement is provided. It should be noted that in an asynchronous DSP, the power dissipation equals zero when the core is not being used. Furthermore there is no power overhead in switching to full performance and so asynchronous operation should be considered for next generation DSPs.

Yuan-Hao Huang et. al[25], proposed a communication digital signal processor (DSP) suitable for massively parallel signal processing operations in orthogonal frequency division multiplexing (OFDM) and code-division multiple-access (CDMA) communication systems. This proposed architecture supports basic butterfly operations, single/double-precision error computation, and the add-compare-select (ACS) operation. The processor chip is fabricated using a 0.35μm CMOS technology. The fabricated DSP chip reaches a speed of 1.1 G MACs/s when operating in high-speed mode, and it achieves 4 M MACs/s/mW in the low-power mode.

The latest series of processors provided by Motorola are called DragonBall[26]. It has been proposed for use in smartphone applications. The DragonBall performs the applications processor processing and provides seamless integration with the Bluetooth wireless chipset and with the Innovative Convergence cellular platform. This new feature leads to a change and enhancement of the functionality in the applications' processor without the need to re-certify the cellular functionality of the Smartphone with carriers. Moreover, it is claimed that DragonBall can provide extremely low power consumption for extended battery life.

2.2.3 DSP for multimedia signal processing

Another approach for performing DSP applications is centred around multimedia processing. Li-Hsun Chen et.al[29], proposed a reconfigurable DSP architecture which has 5 ALUs, 1 multiplier and 2 load/store units. This allows the special purpose DSP

architecture to have a better parallel processing capability for the MPEG-4 Video encoder. In a comparison of the block coding of the MPEG-4 Video, their DSP is nearly 25% more efficient than the TI TMS320C64X architecture. Moreover, it is 80% more efficient in the MPEG-4 video encoding process.

Hyunjune Yoo et. al[13], proposed a multimedia DSP (MDSP) chip for portable applications. Parallel processing techniques such as SIMD, vector processing and DSP schemes were used and four MAC operations ran in parallel. Therefore, the MDSP is able to handle a 2-D video signal and 1-D signal processing. Their DSP speed capability is 30 MHz. The design was implemented using Verilog-HDL and the synthesized core simulation results have shown a performance improvement on various algorithms such as FFT, IIR, FIR, BMA (Block Matching Algorithm) and convolution encoding.

2.3 General Functional Unit for DSPs

The arithmetic unit is the main core of a DSP and operates on almost every cycle. The operands required can be stored in a data register file or be an immediate value. Multiplication, addition, multiply-accumulate and other necessary functions of digital signal processing are performed here. The 40-bit accumulators are also in this unit. As is well-known, the number of processing elements has been increased in a modern DSP. This leads to much research work proposing various low power or high performance arithmetic components such as multipliers and adders to try and achieve the optimal energy requirements of DSPs. However, more complex designs may require extra power and a delay in the overall system.

As already stated, the multiplier is based on adder circuits. The adder can be implemented based on XOR and multiplexer circuits. Therefore, optimal energy of basic cells such as XOR and multiplexer can contribute to the energy saving of the multiplier. However, this can only be applied in some particular multiply and add algorithms which are suitable for implementation with XOR and multiplexer cells.

2.4 Energy Efficient Design

In this section, an overview of low-power design and the techniques to employ them from the structural level down to the logic and circuit level in the FU are described. Design techniques such as avoiding unnecessary activity, reducing the supply voltage, the logical effort for energy efficient circuits, the logic family selection, transistor sizing and layout considerations, are focused on in order to achieve an energy efficient design for a portable computer having a wireless communication system.

The capabilities of computation in portable applications have been rising exponentially but battery technology has improved only slowly. Therefore, the intensive and continuous computing of hand-held computers and other portable devices is becoming restricted by the sources of power and energy management. Consequently, a variety of energy reduction techniques at various levels are required.

2.4.1 Energy Sources

Since 1990, the speed-power efficiency has increased tenfold every 2.5 years for general purpose processors and DSPs. The most difficult task for future mobile system designers is to attempt to pack more capabilities, such as multimedia processing, into a battery operated portable miniature package. The main problem is that there is no equivalent of Moore's Law in the case of battery technology. In particular, only a 20% improvement in capacity is expected over the next 10 years[30]. Figure2.4 shows these trends[30] and it is clear that the energy density of existing battery technologies is far from what is needed.

The Nickel Metal Hydride (NiMH) battery was introduced in the late eighties. Its best qualities were that it was compact in size, had light weight and had a long battery operating time. In the early nineties, more energy was available in a Lithium Ion (Li-ion) cell compared to the Nickel Cadmium (NiCd) battery. [31] examined the batteries not only in terms of energy density but also service life, load characteristics, maintenance requirements, self-discharge and costs. Alternative chemistries were also evaluated against the Nickel Cadmium battery type. However, Ni-Cd is still used in low cost applications like portable CD/tape players. Although Nickel-MH batteries have roughly twice the energy density of Ni-Cd batteries, they have a shorter cycle life and are more

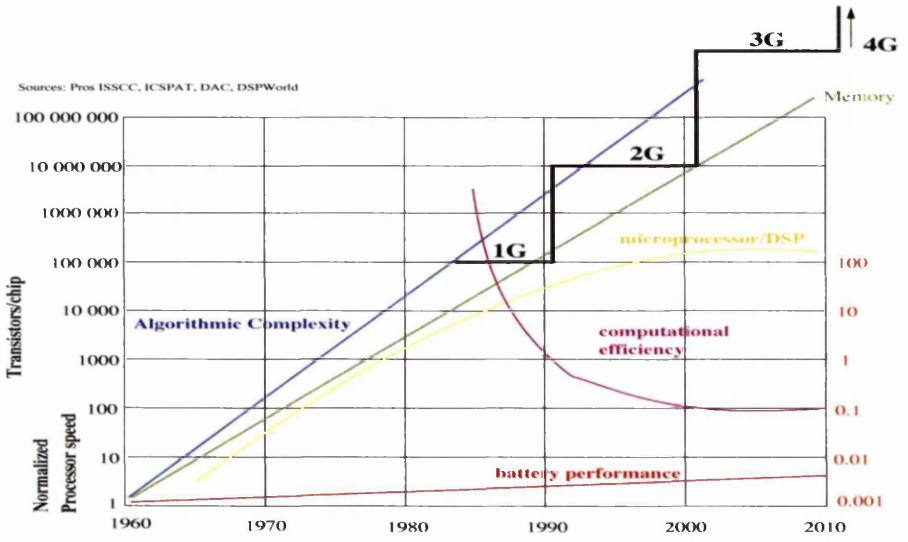


Figure 2.4: Improvement in technology[30]

expensive. So the most popular battery choice for Laptop, PDAs and cellular phones is the Li-ion batteries. However, Li-ion batteries can be unsafe when they are used improperly.

Fuel cells may offer a promising alternative technique. It is an electrochemical device that converts the chemical energy of a fuel directly to usable energy. A fuel cell running on methanol could provide power for more than 20 times longer than traditional Nickel Cadmium batteries in a comparably sized package[32]. Lithium Polymer is one kind of fuel cell. It has been established recently to make ultra thin batteries with less than 1 mm thickness[33]. It is expected to suit the needs of light-weight next-generation portable computing and communication devices. However, these batteries are still expensive to manufacture.

According to the slight improvement of the battery technology, it is clear that the energy density of existing battery technologies is far from what is needed. Hence, an energy efficient DSP becomes vital.

2.4.2 Fundamental of low-power design

Most research work in low power design for battery driven applications is concerned with lowering energy consumption because of the finite energy source. Figure2.5a shows a computation taking time ta when operated from a voltage V. This is followed by an idle time ts. Thus the energy consumed is proportional to V^2ta. When the supply voltage is scaled down by a factor of 2, Figure2.5b, the execution time extends to tb = 2ta and the energy is αV^2ta/2; the power reduces quadratically whilst the performance is reduced linearly. With lowering the supply voltage and running the task slowly so that the idling time is reduced, the implementor needs to be aware of leakage current in a small process geometry and aware of the timing overhead involved in changing the supply voltage. However, there is another approach to gain an energy saving as shown in Figure2.5c. Here high speed logic operating from a supply of V is used to reduce the computation time to tc. Assuming tc = ta/2, the energy consumed is the same as that from scaling the voltage down as shown in Figure2.5b. This approach depicted in Figure2.5c is the one adopted in this work.

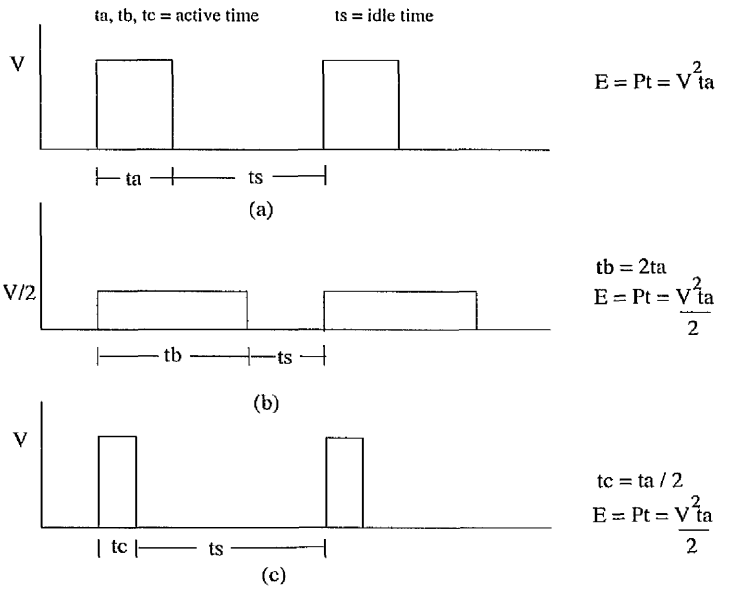


Figure 2.5: Energy versus voltage scaling

The power dissipation in a CMOS circuit needs to be explained before understanding energy and how to minimize it. Following this, an explanation of the several measurement metrics of energy efficiency are then described and the most suitable metric will be chosen to evaluate the FU of a DSP.

2.4.2.1 CMOS Power Consumption

In digital CMOS circuit, there are three major sources of power dissipation as shown in the following equation[4]:

$$\begin{aligned} P_{avg} &= P_{switching} + P_{shortcircuit} + P_{leakage} \\ &= \alpha_{0\to 1} \cdot C_L \cdot V_{DD}^2 \cdot f_{clk} + I_{sc} \cdot V_{DD} + I_{leakage} \cdot V_{DD} \end{aligned} \quad \text{... Eq. 2.1}$$

The switching or dynamic component of power consumption is the product of the load capacitance (C_L), clock frequency (f_{clk}), an activity factor, $\alpha_{0->1}$, is used to denote the average fraction of clock cycles in which a low-to-high transition occurs and the power supply V_{DD}^2. The second term is the power dissipated due to the direct-path short circuit current, I_{sc}, which occurs during switching when both NMOS and PMOS transistor are active. The last term is the power caused by the leakage current, $I_{leakage}$, which arises from substrate injection and sub-threshold effects. The leakage current becomes a significant problem when the fabrication technology is scaled down or when there are significant periods of idle time. In general, in an active circuit, the first term (the switching component) is by far the most dominant term with the short circuit current being generally 10% to 20%.

2.4.2.2 Dynamic Power Consumption

Since each switching event in CMOS circuit expends energy $\alpha_{0->1} C_L V_{DD}^2 f_{clk}$, it is extremely important to reduce this source of power dissipation. Firstly, $\alpha_{0->1} C_L$ needs to be minimized through the choice of logic function, logic style, circuit topology, data statistics and sequencing of operations. Looking at each of these in turn:

Logic function: Generally, each gate will have different static transition probabilities. For example, a 2-input static NOR gate is assumed to have only one possible input transition during a clock cycle and a uniform input distribution of high and low levels. So $\alpha_{0->1}$ of this NOR gate will be 3/16 (=p(0)(1-p(0))=3/4(1-3/4)). For a 2-input static XOR, the 0->1 transition probability equals 1/4 (=1/2(1-1/2)).

Logic Style: The logic style can give a different transition probability. Basically, the activity for the dynamic CMOS depends only on the signal probability, whilst the transition probability in the static CMOS depends on its previous state. So the static CMOS gate cannot be switched, if the inputs do not change. However, the amount of capacitance on the switching node and the sensitivity to supply voltage reduction are other factors for choosing a particular logic style.

Signal Statistics: The signals in a circuit are not always random. It is necessary to consider the effect of signals statistics on power. Assume a 2-input NOR gate has Pa and Pb as the probabilities of a one on its inputs A and B, respectively. The output node has a probability that it is "1" of P_1=(1-Pa)(1-Pb) and the probability of a "0" = 1-P_1. Therefore, the probability of transition from 0 to 1 is (1-(1-Pa)(1-Pb))(1-Pa)(1-Pb). If this probability is plotted, its impact on switching events can be used to significantly improve power dissipation.

Circuit Topology: The manner in which logic gates are connected together can affect the switching activity. The transition probabilities of logic gates are computed for two topologies: chain and tree structure as shown in Figure2.6. The results in [5] indicate that the tree implementation will have an overall lower switching activity than the chain structure for random inputs. However, the extra transition is a function of input patterns; delay skew and logic depth can still cause extra power dissipation. To eliminate these transitions, a careful design such as balanced paths is needed.

2.4.2.3 Short-Circuit Component of Power

The short circuit path exists for direct current flow from V_{DD} to GND, when both the pull down and pull up circuits in conventional CMOS conduct. The short-circuit dissipation is

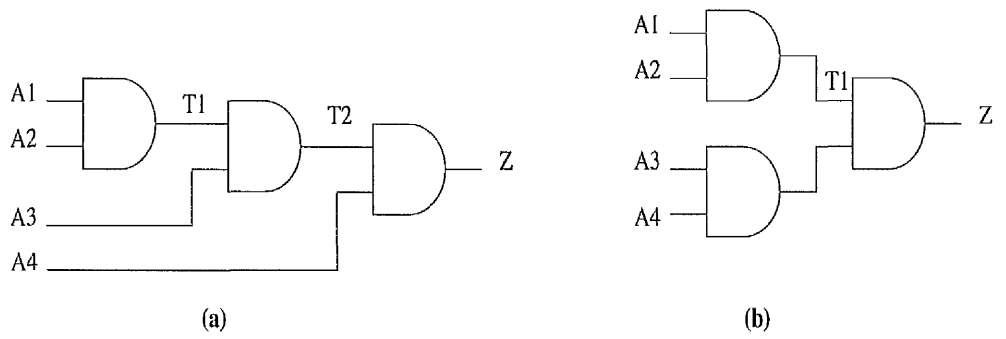


Figure 2.6: Circuit topology (a) chain structure (b) tree structure

a small term as a fraction of total dissipation as the output load is increased. The short-circuit power dissipation $P_{short\ circuit}$ or P_{sc} of an unloaded inverter is[5]:

$$P_{SC} = \frac{\beta}{12}(V_{DD} - V_T)^3 \frac{\tau}{T} \qquad \text{... Eq. 2.2}$$

From this equation, Psc depends upon the frequency (1/T), β the process technology constant, V_{DD} and the fall time of the input signal (τ). In[34], simulation shows the Psc variation of an inverter with a load capacitance, C_L. It is clear that the short-circuit dissipation can be small when the rise and fall times of input signals are about equal to the output rise and fall times.

2.4.2.4 Leakage Component of Power

There are two types of leakage current. The first type is diode leakage which can occur when a transistor is turned off and another transistor charges up/down the drain with the respect to the former's bulk potential. Normally, the leakage current will be approximately $A_D J_S$, where A_D is the area of the drain diffusion and J_S is the leakage current density. This is typically a very small fraction of the total power consumption. However, it depends on how long the circuit spends in stand-by mode as power always dissipates even when there is no switching activity. The second type of leakage current is subthreshold leakage which occurs because of carrier diffusion between the source and

the drain when the gate source voltage (Vgs) exceeds the weak inversion point but is still below the threshold voltage.

Basically, the diode leakage current is very small and can be ignored[35]. The subthreshold leakage current increases exponentially with the downward scaling of both the supply voltage and threshold voltage[36]. Therefore, $I_{leakage}$ values are set to become a critical block to improving battery lifetimes in portable applications. Further information about leakage reduction can be found in many research works[37],[38],[39].

2.4.2.5 Energy Per Operation

The power-delay product (PDP) is a common measure of energy consumption for portable applications. The energy consumed per clock cycle can be derived by dividing the PDP with the number of operations per clock cycle as shown in the equation below:

$$\frac{Energy}{Operation} = \frac{V^2_{DD} \bullet C_{EFF}}{Operations} \quad \text{... Eq. 2.3}$$

where the effective switched capacitance, C_{EFF} is commonly expressed as the product of the physical capacitance C_L and the activity weight factor α, each averaged over the N nodes. The delay of a CMOS gate can be defined as the time required for the output to transition to 50% of the voltage swing and is[40]:

$$Delay = \frac{C_L}{I_{AVE}} \bullet \Delta V_{0 \rightarrow 50percent} = \frac{C_L}{I_{AVE}} \cdot \frac{V_{DD}}{2} \quad \text{... Eq. 2.4}$$

where I_{AVE} is the average device current during the transition. The device remains in saturation during the output transition and the current is approximately constant such that $I_{AVE} = I_D$, so the delay can be given by[41]:

$$Delay \cong \frac{C_L \cdot V^2{}_{DD}}{k_v \cdot W \cdot (V_{DD} - V_T)^2} \quad \text{... Eq. 2.5}$$

where k_v is a technology dependent constant, V_T is the transistor threshold and W is the transistor width.

2.4.2.6 Energy Efficiency Metric

In order to compare designs that run at different speeds and consume different amounts of energy, the energy (E) and the delay or throughput metrics need to be considered. In [41], different energy efficiency metrics are presented. There are three modes of computation: fixed throughput, maximum throughput and burst throughput. Firstly, fixed throughput mode is suitable for use in real-time systems (e.g., speech, audio and video) which require a fixed number of operations per second. The excess throughput will consume unnecessary energy. The metric can be expressed as:

$$Metric_{FIX} = \frac{V^2{}_{DD} \cdot C_{EFF}}{(Operations)/(ClockCycle)} \quad \text{... Eq. 2.6}$$

From the fixed throughput equation, the way to improve energy efficiency is to reduce the supply voltage and the effective switched capacitance while the throughput remains constant.

The maximum throughput mode is the second energy efficient metric. It is used in most multi-user systems such as networked workstations and main-frames which require the processor to run continuously. Hence, the energy efficiency metric must balance the need for low energy/operation with high throughput. Here,

$$Metric_{MAX} = \frac{E_{MAX}}{T_{MAX}} = \frac{Power}{Throughput^2} = \frac{C_L \cdot N_{gates} \cdot C_{EFF} \cdot V_{DD}{}^4}{k_v \cdot W \cdot (V_{DD} - V_T)^2} \quad \text{... Eq. 2.7}$$

where throughput is the number of outputs per second and E_{MAX} is the power/throughput corresponding to maximum throughput T_{MAX}. From this equation, the energy efficiency for a maximum throughput mode can be improved by increasing the throughput with the same amount of energy/operation or power. Considering the last term of the equation, it is clear that the variables have several dependencies. For example, if the width (W) of the transistor is increased, C_L and C_{EFF} will also increase. So the individual parameters cannot be considered separately.

The last energy efficiency metric is the mode which is suitable for portable applications such as the PDA and laptop. The burst throughput mode of the energy efficiency metric must balance the minimized energy consumption for both idling and computing with the maximum throughput when computing. The metric of energy efficiency for the burst throughput mode is:

$$Metric_{BURST} = \frac{E_{MAX} + E_{IDLE}}{T_{MAX}} \qquad \text{... Eq. 2.8}$$

where E_{IDLE} is the average energy/operation while idling and E_{MAX} is the average energy/operation during full activity. According to the three different throughput modes, the energy efficiency metrics can be used to describe the energy efficiency of a processor which has the specification of a target application. The metric that has been chosen for this DSP aimed at the next generation of portable applications such as mobile phones is the burst throughput mode. The reason is because most of the time the mobile phone is in the idle mode. However, it requires a high throughput when computing.

2.5 Low Power Systems

2.5.1 Low power design at a technological-level

There are several factors that need to be considered when lowering the power consumption at a technological-level and applying this to a design. At this level, it is necessary to consider the process technology and how the design is laid out.

Process Technologies for Reducing Power

Each new process results in a reduction in the minimum gate length, more metal layers etc. and this effects a power reduction in the CMOS logic. Unfortunately, designers cannot control these process parameters. However, it is necessary for designers to understand how the process technologies reduce the power so that the designers can select the process for an energy efficient design. In addition, a new process has many other parameters that the designers should be aware of such as the fact that higher leakage currents occur on silicon below 0.13μm.

Each new process has an impact on reducing power dissipation as well as increasing the speed. A proportional decrease in power and increase in the circuit speed are provided by reducing all capacitances in the scaling process. However, the thickness of the interconnect metal is roughly the same across processes[42]. This leads to an increase in the capacitance between adjacent interconnecting segments. As a result, the overall capacitance scaling is below a factor S if the geometry is scaled down by a factor S.

Design Style

The selection of design style can have a large impact on the energy efficiency. The best energy efficient design style is full custom design. In this case, all the energy efficient design techniques such as transistor sizing and logical effort can be applied to the circuits. In addition, full custom design allows designers to select the metal layers which give a power saving. Since the higher metal layers have a smaller physcial capacitance, the high-activity signals should be routed by upper layers of metal. Moreover, high-activity wires should be kept short and local. However, full custom design is a costly method in terms of design time.

Semi-custom design (using standard cells) is a design style which is selected to reduce time to market. Even current standard cells and tools have within their design various low-power methodologies to achieve low power. Most standard cells are designed for the maximum performance with worst-case loading of the routing. Special cells such as pass transistor logic cells where the circuit contains only NMOS (such as single-ended pass transistor logic) or both NMOS and PMOS (such as pass transmission gate) but has no

power or ground connection (see more detail about pass-transistor logic in section 2.5.2), are not included in normal standard cell libraries. The reason is because the area of a pass transistor circuit is smaller than the standard cell area where the conventional CMOS can fit in. Therefore, the standard cell layout style with pass-transistor logic topology cannot achieve the area and energy efficiency of the pass-transistor circuit.

Field Programmable Gate Arrays (FPGAs) offer an alternative method for shorter design time and better flexibility than full custom and standard cells. However, FPGAs use more area and power and have less performance. Even though the FPGA tools can offer a transistor sizing option, its energy consumption is still relatively high and is not acceptable for future requirements of realising energy efficiency in portable applications.

At this level, the circuit is designed and implemented on 0.18um geometry process technology because this is the design kit available to the author. So the process technology factor for reducing power cannot be controlled in this research work. However, this has the advantage that leakage currents can be ignored. The selection of design style is full custom enabling the author to design special circuits such as pass transistor logic effectively and to include other energy efficient design techniques. This decision allows this DSP to have a great potential for energy reduction.

2.5.2 Low-Power design at a circuit-level

The next level up where power dissipation can be lowered involves optimizing the circuits. Here, techniques include transistor sizing, scaling the supply voltage, reduced signal swing and choice of logic styles. The trade-off between performance and power of each technique is illustrated in this section.

2.5.2.1 Transistor Sizing

Transistor sizing is one method for making a power efficient CMOS circuit. This method is independent of the logic topology selection. Generally, it is believed that using minimum size transistors keeps the power of a design low. However, the circuit which aims to use battery-operated applications requires a measurement in term of energy consumption, not just power consumption. Considering these facts, transistor sizes can be

increased to give the optimal sizing for low energy per operation. Hence, it is totally different from that which is required for highest speed or lowest power. There are several research works about transistor sizing such as how to get the minimum size of transistors[43], how to apply a transistor sizing method to an automatic layout generation tool[44], and power reduction using transistor sizing[45],[46],[47],[48].

Transistor sizing was introduced[49],[50],[51] to optimize the circuit speed. The transistor size has been increased mostly in the critical path. The right transistor size should be used in order to meet the demands of energy efficiency. In principle, the delay calculations are usually done by using Elmore's RC delay model reference[52]. Here, each ON transistor is modelled as resistors, therefore a chain of transistors can be represented as an RC ladder. The sum over each node in the ladder of the resistance R_{n-i} multiplied by the capacitance (C_i) is estimated as the delay of an RC ladder (t_{pd}). Here,

$$t_{pd} = \sum_i R_{n-i} C_i \qquad \text{... Eq. 2.9}$$

In [53], a power-efficient driven device sizing of a pass transistor in low-voltage CMOS was reported. This gives useful information about transistor sizing for power efficient pass transistor circuits under variable temperature conditions. The power delay product ($P\tau$) is given by:

$$P \cdot \tau \propto \frac{C_{total}{}^{2} V_{DD}{}^{1.7} L^{0.5} T_{OX}{}^{0.5}}{((0.9 - V_t)/V_{DD})^{1.9}} \cdot \left(\frac{1}{W_n} + \frac{2.2}{W_p}\right) \qquad \text{... Eq. 2.10}$$

where τ is the gate delay, C_{total} may be divided into a factor contributed by the driver devices and the rest of the load, Tox is the oxide thickness, L is the length of transistor, W_n is the width of the NMOS device and W_p the PMOS width.

When the optimal ratio is independent of the load capacitance, Wp/Wn equal to 1.5 will then give the minimized power delay product (τP). However, in practice, the optimal power delay product depends on the total capacitance that may be divided into a factor

contributed by the driver devices and the rest of the load. In this research work, the optimal power delay product can be found out by simulation.

Many transistor sizing methods have been presented in the past. Currently, transistor sizing is applied using a CAD tool enabling the identification of the best circuit arrangement to achieve the best power-delay-product improvement[54],[55],[56]. In addition, transistor sizing can also be used for delay balancing, so that each circuit part has roughly the same delay; this results in no critical path in the design.

2.5.2.2 Voltage Scaling

The equation for power dissipation in a conventional CMOS circuit shows that the largest reduction of power results from reducing the supply voltage. This is due to the power being proportional to the square of the supply voltage. The short-circuit component is a small fraction whilst the leakage current is ignored with the assumption of a relatively large difference between V_{DD} and threshold voltage V_T. If the circuit performs one operation per cycle, then the energy per operation is

$$E = \alpha C_{EFF} V^2 \quad \text{... Eq. 2.11}$$

where C_{EFF} is the effective capacitance being switched to perform a computation. If the supply voltage is scaled down from 3.3 V to 1.8 V, this gives a 70% reduction in energy consumption. However, the energy consumption improvement from reducing the supply with a constant threshold voltage makes the gates slower. So some techniques such as pipelining and parallelism have to be used to keep the throughput constant. Another simple and attractive method is moving to a lower threshold voltage process. This allows the designers to reduce the supply voltage and power without requiring a major change of design, since the gate speed would remain constant[57]. However, further analysis is needed when sleep modes and variations in the supply voltage are taken into account. Some papers have reported that the circuit should use a lower threshold voltage during normal operation and a higher threshold voltage during sleep modes[58]. Alternatively, [59] has proposed using direct control of the threshold voltage instead. However, it is difficult to control the speed of each gate when both the threshold and supply voltage are

varied. This will force a reduction in the levels of logic in the design to maintain the performance. Finally, low voltage operating looks very attractive for energy efficiency but it is very sensitive to manufacturing variations and operating point changes. All advantages for energy efficiency will disappear if all these variations are not taken into account carefully.

2.5.2.3 Signal Swing

Generally, the dynamic power consumption equation is known as the product of load capacitance, the average number of times that node n transitions/cycle and the signal swing V_{sig}; V_{sig} is normally equal to V_{DD}. Consequently, the power consumption can be decreased by reducing the signal swing to lower than the supply voltage. Meanwhile the delay can also be reduced since delay is proportional to signal swing.

To adopt this technique, an extra device is needed to limit the swing because the output normally swings from the rail to ground in both static and dynamic CMOS circuits. With this extra device, its parasitic capacitance adds to the total effective capacitance being switched. Hence, as long as the reducing signal swing is greater than the effect of the increased parasitic capacitance, the energy consumption will be reduced. However, there is some drawback in reducing the swing signal. Static power for a high output voltage can be dissipated, when connected to the next stage because the output does not reach the supply voltage tending to turn on the pull up device in the driven gate. In contrast, if the following stage is a dynamic pull-down network, the problem will be eliminated because the inverter's output is only driving an NMOS device. However, reducing the signal swing will reduce the drive to the pull-down network. As a consequence, NMOS devices of the next stage must be sized up to maintain the speed. Reducing the swing signal has been used in memory circuits in on-chip signalling techniques[60] and low-power arithmetic circuits[61].

2.5.2.4 Logic Topology

Up to this section, many approaches for lowering power consumption have been described. To achieve energy efficient design, many design techniques are needed. This

section is another important factor in identifying whether a design can gain a large energy efficiency improvement or not. The basic circuit topology is an important issue in the design of VLSI circuit especially in battery operated applications which require both high speed and low power. This section describes both the advantage and drawback of each CMOS circuit design style for energy efficiency.

The logic style of the logic gate influences the speed, area, wiring complexity of a circuit and power dissipation. The switched capacitance should be minimized and can be achieved by having as few transistors and circuit nodes as possible. Since a supply voltage reduction technique is usually applied to reduce the power dissipation, a logic style providing fast logic gates is normally used to speed up critical paths in order to achieve the same throughput. For this purpose, a logic style must be robust against the supply voltage reduction. In addition, a distinction must be made between dynamic and static logic styles. In dynamic logic the operation is divided into two modes, *precharge and evaluation*, as shown in Figure2.7(a). In precharge, the pull up circuit operates to pull the output Y high. In evaluation, the pull down operates and point Y either remains high or its output is pulled low. Dynamic logic is attractive for high performance applications. However, a system which has a high signal transition activities such as DSPs can result in excessive high power dissipation if dynamic logic has been used due to the precharging mechanism.

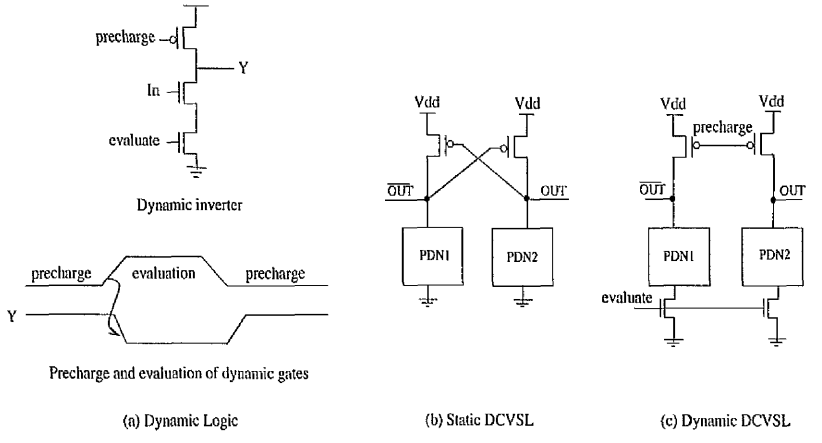


Figure 2.7: (a) Dynamic logic (b) Dynamic DCVSL (c) Static DCVSL

Conventional static CMOS has been a choice in most processor designs including DSPs. However in 1996, Yano et al.[62], suggested using static pass transistor circuits and dynamic circuits have been used in low-power and high speed systems such as processors from MIPS Technologies since 1994.

Conventional static CMOS, complementary pass-transistor, double pass-transistor, Energy Economized Pass Transistor Logic and Single-ended Pass Transistor Logic are now presented:

Conventional Static CMOS: This logic style has been used in most chip designs. It comprises a PMOS network for the pull-up and a NMOS network for the pull-down. In order to have output drive, the width of the transistors must be increased. This increases the input capacitance which in turn increases the power dissipation and propagation delay.

Differential Cascade Voltage Switch Logic (DCVSL): DCVSL was introduced in [63]. It was derived from two complementary DOMINO gates with merged logic trees. There are two kinds of differential logic style: static and dynamic. Static DCVSL as shown in Figure2.7(b) has two complementary NMOS trees connected to a pair of cross-coupled PMOS transistors. The input capacitance is two to three times smaller than conventional static CMOS since only the NMOS transistors are driven. Dynamic DCVSL[64] as shown in Figure2.7(c) is a combination of domino logic and static DCVSL. Dynamic DCVSL has an advantage over domino logic due to its ability to generate any logic function, whilst the domino logic can only generate without logic inversion.

Pass Transistor Logic

Pass gates or transmission gates are the combination of an NMOS and a PMOS pass-transistor as shown in Figure2.8 and are often used for implementing multiplexers, XOR-gates, and latches. The advantage of the pass gate over other pass-transistor logic styles is its robustness against voltage scaling and transistor sizing due to its high noise margins (as the results will show in chapter6). The pass gate also results in a smaller number of transistors and smaller input loads compared to conventional static CMOS. However, an inverter is required to buffer its output and provide output drive.

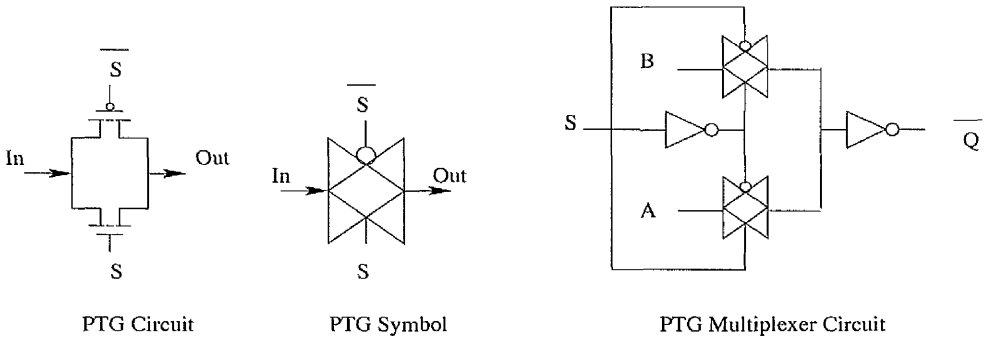


Figure 2.8: Pass gates or Pass Transmission Gate (PTG)

Complementary Pass Transistor Logic (CPL): A CPL gate is shown in Figure2.9 and comprises two NMOS logic networks, two small pull-up PMOS for swing restoration and two inverters for complementary outputs. With CPL logic, any two input logic function, such as AND, OR and XOR can be implemented by this basic structure. However, it is relatively expensive for a simple gate such as NAND and NOR. The advantages of CPL are small input loads, good output driving and fast restoring stages because of the PMOS pull-up transistors. Unfortunately, the number of nodes and high wiring overhead (dual-rail signals) cause a high power dissipation.

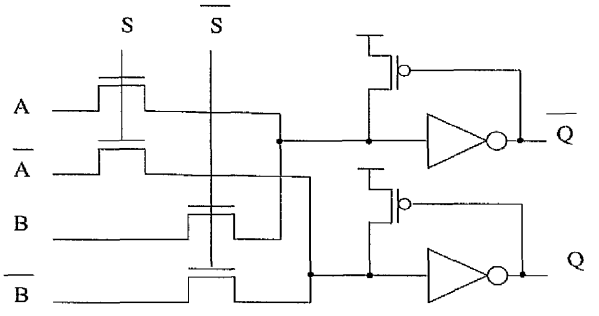


Figure 2.9: Complementary Pass Transistor Logic (CPL)

Energy Economized Pass Transistor Logic (EEPL): EEPL is shown in Figure2.10 The source of the PMOS pull-up transistors are connected to the output signals. This logic gives a shorter delay and smaller power dissipation compared to CPL.

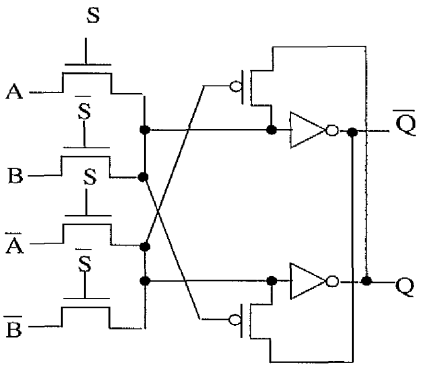


Figure 2.10: Energy Economized Pass Transistor Logic (EEPL)

Single-ended Pass Transistor Logic (SPL): A SPL gate previously known as Lean Integration with Pass Transistors: LEAP) is shown in Figure2.11. It requires only single inter-cell wiring and a single NMOS network. Swing restoration consists of a feedback pull-up PMOS transistor. This swing restoration structure only works for $V_{dd} > V_{tn} + |Vtp|$ due to the threshold voltage drop through the NMOS for a logic '1'; preventing the NMOS transistor of the inverter from turning on. The robustness of the SPL circuits is also affected when the supply voltage is scaled down.

After reviewing the possible low power logic families, two pass transistor logic styles -

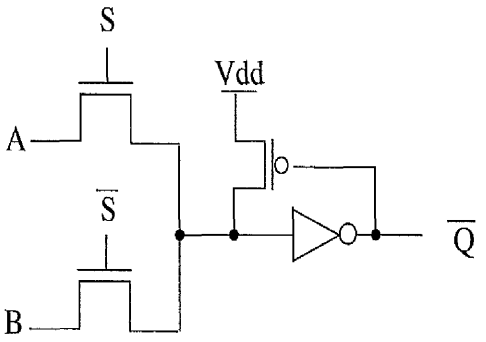


Figure 2.11: Single-ended Pass Transistor Logic (SPL)

the pass transmission gate (PTG) and single-ended pass transistor logic (SPL) were

identified as having the potential for building the low energy circuits required in this research work.

2.5.2.5 Logical Effort

The method of logical effort as presented in [65] is a simple model of the delay through a single CMOS logic element. Logical effort has been used to improve the delay of the conventional CMOS circuits. It provides the sizes of different transistors to make the circuits run with the maximum possible speed. Logical effort also defines the performance cost of computation inherent in the circuit topology.

The delay (d) of a logic gate comprises two parts: a parasitic delay (ρ) and the effort delay or stage effort (f) which is proportional to the output load.

$$d = f + \rho$$

The load and the properties of the logic gate driving a load affect the effort delay which is the product of logical effort (g) and electrical effort (h). The logical effort is a property of the logic gate and captures the effort of the logic gate topology in relation to its ability to produce the output. Logical effort is independent of the size of the transistors in the circuit.

$$f = gh$$

Whilst the load can be characterized by h, the size of transistors in the gate will determine the load-driving capability and affects its performance. These are dependent on h and h can be defined by

$$h = \frac{C_{out}}{C_{in}}$$

Where C_{out} is capacitance at the output of the logic gate and C_{in} is the capacitance at the input terminal of the logic gate.

However, most designers will define the electrical effort (h) in term of *fanout* which is only dependent on the number of gates being driven.

$$d = gh + \rho$$

ρ is dependent on the internal capacitance and is also largely independent of the size of transistors in the logic gate. Electrical effort h combines the effects of the transistor sizes in the logic gate, the input capacitance C_{in} and the external load (C_{out}).

Logical effort has been considered in this research work in order to reduce the delay of the circuits. This leads to a gain in energy saving.

2.5.3 Low-Power design at a logic-level

At the logic level, the opportunities to minimize the power budget exist in both the capacitance and frequency of the power dissipation equation. The most important factor in logic optimization is the minimisation of switching activity.

Data guarding

In most CMOS digital systems, switching activity is the major cause of power dissipation. Switching activities not concerned with useful work should be eliminated. Guard logic is one approach to minimize such switching activities. Transparent latches with an enable signal are used to guard against non-useful switching activities propagating further into a system. The latches are only transparent when the data is to be used.

Glitch activity reduction

Before the correct logic level is reached, there is often spurious transitions from one block to the next block (called critical races and dynamic hazards). These extra transitions waste power. Balanced signal paths and a short logic depth are able to minimize glitching activities. For example, a tree structure has more balanced signal paths than a chain structure, Figure2.6, if all primary inputs arrive at the same time. Thus the capacitance switched for a chain implementation is larger than the tree implementation depending on

the number of inputs. For the chain case, pipelining can be applied to reduce the logic depth. Normally, there is a trade-off between glitching capacitance (the normal chain case with a long logic depth) and register capacitance (the chain implemented with pipelining). However, using the tree implementation the supply voltage can be reduced whilst keeping throughput fixed.

When the low energy circuit is designed at this level, the optimal switching activity is targeted. The datapath of this DSP is therefore designed to stop non-useful switching activities propagating further into the datapath. In addition, balanced signal paths are used in the multiplier whilst overall the design is implemented using a short logic depth.

2.5.4 Timing Approach

Asynchronous Design

The FU unit forms part of an asynchronous pipelined design. Asynchronous timing provides further power improvement as it eliminates clock generation, buffering and distribution. This asynchronous approach also gives a reduction of electromagnetic interference (EMI) as the switching of the logic is spread instead of being concentrated around the clock edge. The FUs are therefore based on the principle of micro-pipelines[66] where the data transfer between blocks uses local handshake signals rather than clocks.

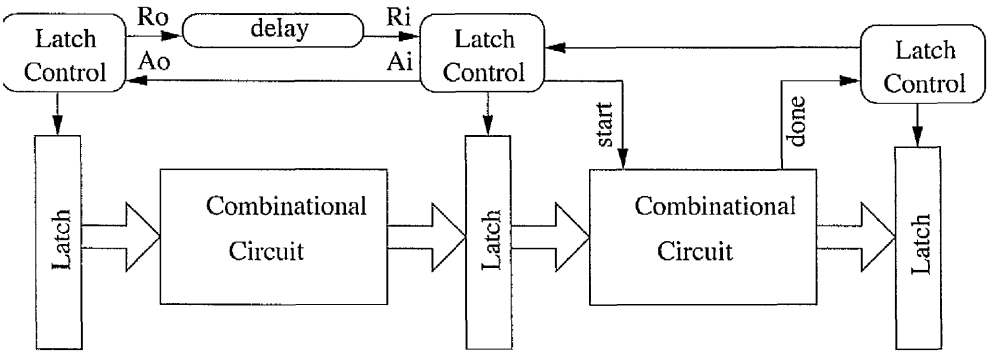


Figure 2.12: Principle of micro-pipeline

The principle is shown in Figure2.12. The done signal allowing an output to propagate to the next micro-pipeline stage is generated either from the combinational logic for a data dependent operation or from a matched delay. Where the operation time is data dependent, as in the adder, the operation can be self-timed by using a completion-detection (CD) circuit and many CD techniques are proposed [67],[68],[69],[70].

The circuit in the FU has the additional requirement of being low power. Therefore dual-rail coding which has two wires per signal which always return to a 00 state, or duplicate logic producing the done signal which has the same delay as the combinational circuit is not suitable for such a low power application. When a single-rail is chosen for the low energy FU, the timing mechanism needs to be considered. The completion-detection techniques referred to in the previous paragraph were reviewed. Current-sensing CDs are not suitable because they require an additional supply voltage and have higher quiescent current than synchronous circuits negating their advantage. Activity-Monitoring CDs (AMCD) have an even higher energy-delay when there is a maximum ripple path. This is disadvantageous in a DSP since a maximum ripple path occurs in many arithmetic operations. Many designs use the bounded delay approach as it is the easiest CD method to implement. However, this is not a proper completion-detection method since the delay is fixed and needs to be longer than the maximum delay path. In [70], it is claimed that if a combinational circuit has a critical path of less than ten gate-delays, then a bounded delay approach works satisfactorily.

The bundled-data asynchronous circuits are based on micropipelines. The request signal indicates that the data is valid and the acknowledge signal is asserted when the data is received. When the sender disasserts the request, the receiving device can remove the acknowledge signal. Therefore the data is valid between the rising edge of request and the falling edge of acknowledge. These handshake operations are well-known as 4-phase timing. Two-phase pipelines could be used for event detection on the request and acknowledge. This makes 2-phase pipelines faster than 4-phase pipelines. But 2-phase control circuits are significantly more complex than 4-phase circuits so 4-phase is chosen for this work.

Clock gating

Most power minimization techniques at the logic level rely on switching frequency. Thus in the clock system, the use of clock gating is the best technique as CMOS power dissipation is proportional to clock frequency; therefore it is an obvious way to reduce power consumption. In clock gating, the clock toggles only when an enable signal is true. This technique is used to shut down some parts of the chip that are inactive. Thus power is saved due to the clock line not being activated all the time. With clock gating, the energy consumed in driving the register's clock input is reduced in proportion to the decrease in average local clock frequency. Moreover, clock gating can be applied globally to achieve larger energy reduction. Meanwhile, the control signal can be done at the hardware level or the operating system or can depend on the application. However, the drawback of this method is that when the design is reactivated the oscillators used require milliseconds to be stable after being re-activated. So chips exhibit less performance if they enter the sleep mode for a long time.

The technique employed in this research work is that the whole system is clock-free. Clockless (or asynchronous) logic is used to eliminate clock generation, buffering and distribution – a major power user – at the system level. Each functional unit is responsible for its own timing; data is passed in and out using handshake signals. This means that a functional unit which is not in use dissipates almost no power. If evaluation in one unit is slow the whole system will adapt on a cycle-by-cycle basis. The final advantage of asynchronous logic here is that it adapts automatically as the supply voltage is changed; a reduced input voltage gives slower processing (counteracted somewhat by the multiple, parallel functional units) but much greater energy efficiency.

2.5.5 Low-Power Design at an architecture level

Architecture with voltage scaling

An effective approach for power dissipation reduction at the architecture level is an architecture driven voltage scaling strategy which can yield more than an order of magnitude saving. It is shown clearly from the CMOS power dissipation equation that the

system can achieve lower energy consumption as the supply voltage is reduced. The task for the architecture designer is to retain the throughput whilst reducing the supply voltage. A parallel architecture is one way to maintain the throughput. However, the capacitance has increased by a factor of 2, whilst the operating frequency has correspondingly decreased by a factor of 2. Unfortunately, the extra routing involved between units can result in a slightly increased capacitance. The principle can be extended to reduce the supply voltage for a fixed throughput by increasing the parallelism further. However, when the supply voltage approaches the threshold voltage of the devices, the delays will increased significantly. [4] examines the optimum voltage and the overhead circuitry arising from the parallelism when a further reduction in the supply is considered.

Another approach is to apply pipelining to the architecture. With this method, only the additional pipeline latch is an area overhead. Thus the capacitance is increased by less than a factor of 2 compared to parallelism, whilst the supply voltage is again reduced by a factor of 2.

Resource sharing

Another approach to reducing power is to share resources.

Time-sharing buses: The choice of bus topology is an important issue which can affect the energy consumption of the system. This is illustrated by considering two cases. In the first a system has two separate buses running at the same frequency whilst in the second case the system has a single bus shared between two components running twice as fast so the throughput is the same in both cases. The power consumption of the system using separate buses is given by:

$$P_{separate-buses} = \Sigma \alpha_i C_{sb} V^2 f$$

where α_i is the transition activity for bit i (both 0 -->1 and 1 --> 0) and C_{sb} is the capacitance per bit of two buses. With this equation, we assume that all bits have the same capacitance and the supply is a fixed voltage. The transition activity can be computed as $\Sigma\alpha_i = 1 + 1/2 + 1/4 ... + 1/128$ due to the least significant bit (lsb) switching every cycle,

the 2nd lsb switching every other cycle and so on. $\Sigma\alpha_i$ approximately equals 2 for each bus. So the system has $\Sigma\alpha_i$ equal to 4.

For the shared bus, running at twice the frequency, the number of transitions per cycle depends on the skew between the outputs, whilst the number of transitions for the separate buses is independent of skew. Thus the separate buses implementation has less switching activity than the bus-sharing implementation.

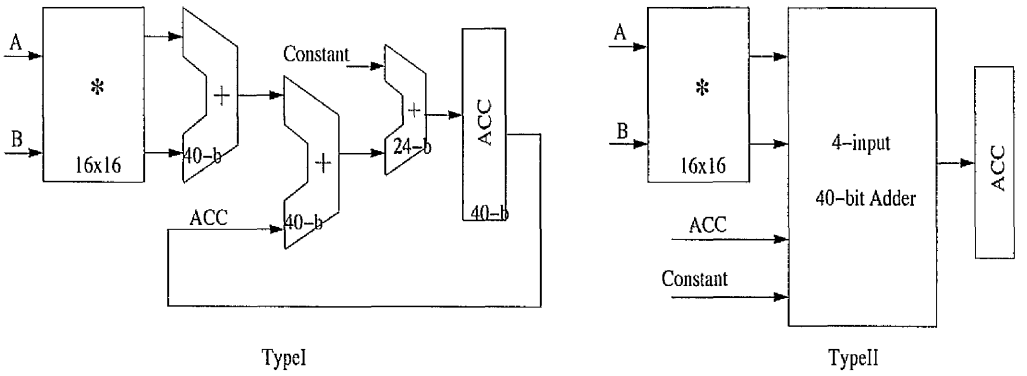


Figure 2.13: Two possible structures of execution unit-sharing

Execution unit-sharing: The multiply-accumulator (MAC) operation is frequently performed in general DSP applications. The multiplication is performed before the addition. One possible implementation might be to have two 40-bit adders and one 24-bit adder. The first 40-bit adder performs the addition on the final stage of multiplication, whilst the other performs the addition of accumulator and the 24-bit adder does the rounding constant addition as shown in Figure2.13a. The alternate implementation might be to have a single adder performing two and half additions as shown in Figure2.13b. In the first implementation style, the critical path for the addition equals the delay through 3 adders. Meanwhile, the second implementation of this design only has a 4-input adder and it is the approach adopted in this work. The detail of the 4-input adder will be described in Chapter 4.

2.5.6 Low-Power Design at an Algorithmic Level

At the highest design level, the choice of algorithm is considered in order to meet the power constraints. A parallelized algorithm having the minimal number of computations are usually used to achieve the optimal power consumption at an algorithmic level. An architecture driven voltage scaling strategy is generally present in architectural parallelism. An algorithm running on architecture which has no feedback path is easily parallelizable. In DSP algorithms such as the FIR or discrete cosine transform (DCT), the computation cannot be easily parallelized, so algorithm transformations are required to avoid the communication bottlenecks. Another approach is to reduce the number of operations, this technique is based on a conversion of multiplications using constants into shift-add operations since multiplication with a fixed coefficient is commonly used in digital signal processing applications.

2.6 Concluding Remarks

Energy efficient design is a problem because so many factors are involved in the design hierarchy. Getting it wrong at any level can make for a poor power efficient design even though the rest of the design is good. For example, the original CADRE design is energy efficient at the architecture and algorithmic levels but was not optimised at the circuit and logic level. According to the above background knowledge, the DSP architecture adopted in this research work uses the parallelism offered from operating four full custom functional units. In addition, a low power logic family, namely pass transistor logic, has been adopted for the datapath and investigated in more detail in order to use it effectively. At the logic level, reducing the switching activity using transparent latches and having well balanced signal design is used; in addition, asynchronous timing makes the whole system clock-free. At the architecture level, supply voltage scaling and resource sharing are employed to gain further energy saving. Finally, mapping the data onto four functional units, so as to reduce the number of memory accesses, is performed at the algorithmic level. With all these features for lowering energy, the design has a great potential to become an energy efficient DSP.

Chapter 3: Asynchronous Parallel DSP

It is well-known that the capabilities of computation in portable applications are increasingly exponentially. However, the intensive and continuous computing of hand-held computers and other portable devices is limited by the source of power. From [71], the technology graph shows clearly that the energy density of existing battery technologies is far from what is needed. Hence, an energy efficient design becomes vital.

Digital Signal Processors (DSPs) have been developed and widely used in wireless applications such as mobile handsets and PDA mobile phones. A modern mobile phone requires more applications such as video decoding, data processing and speech recognition, than in the past. Multiple standards are also needed in each device. Therefore, the trend of DSP architecture in wireless applications is parallelism; exploiting more than one processing unit gains higher throughput. Whilst the area is increased by employing multiple computing units, the energy consumption is able to be reduced by scaling down the voltage supply. For moderate voltage scaling, the logic depth, and the amount of switching and the workload factor of the circuit will determine the energy efficiency. If the power-delay product metric, i.e. energy, is used as the basis of comparison between DSP designs, then the overall energy efficiency can be improved in a DSP with parallel FUs, as each FU can be operated at lower throughput. This is because when the energy-delay product is taken into account, each component uses the smallest energy source per operation. When the speed has been slowed down, each component will dissipate less power. Meanwhile, the overall throughput of the system is still maintained because of the parallel architecture. Therefore, the system can gain an energy efficiency improvement merely by introducing parallelism.

An alternative approach to make an energy efficient system is to structure algorithms to fit the available hardware resource in a DSP. This usually has restrictions with the existing hardware architecture. However, at the algorithmic level, this approach can result in a power improvement in most systems. Therefore, several energy efficient design techniques can be combined to achieve a larger energy improvement in a DSP system.

Another requirement of a DSP is flexibility. Many reconfigurable digital signal processors with designs implemented on Field Programmable Gate Array (FPGA) technology have been proposed. However, the power dissipation is still relatively high. New generation FPGAs include specific arithmetic units, accelerators, IP cores that provide speed and flexibility, but still tend to be high power. The research challenge is to develop both an energy efficient and flexible design. Portable systems need a processing unit that gives them the lowest energy consumption, but at the same time must provide enough flexibility to the user and programmer. The functional unit presented here is an energy efficient and flexible component for an asynchronous parallel DSP performing computationally intense algorithms. A configuration memory has been attached to each energy efficient FU in this design for flexibility. Internally, the design combines coherent energy efficient circuit/logic design techniques.

3.1 Background

The FU has been designed and implemented for a parallel asynchronous DSP named CADRE [72]. CADRE was expected to be a minimum power consumption DSP whilst meeting the performance requirements of next generation cellular phones. However, the simulation results show that the power dissipation is still on the high side. In [72], the power dissipation of CADRE was analysed for random plus speech data and approximately 50% of the overall power consumption was found to be dissipated in the FU. Thus, reducing the power consumption of the FU was the primary motivation behind this research. The original philosophy of the architecture is described to help the reader understand the principles of this FU.

CADRE was implemented by exploiting four-way parallelism, as this appeared to be optimal for power reduction [5]. This was based on the premise that area can be traded for increased speed because silicon area is rapidly becoming less expensive. Most of the DSP

activity can be characterized by frequent repetition of fixed instruction sequences. So, the instruction encoding which determines the selection and passage of data for each operation can be predetermined and stored in advance in a configurable memory which is located locally to each FU. These encodings can then be recalled with a compressed instruction. Because the configuration memories are RAMs, this allows reconfiguration at any point in execution. In addition, these encodings could be expanded within the FUs. This dramatically reduces the size and amount of information that needs to be fetched from main memory. CADRE used a dual Harvard architecture that has one program memory and two separate data memories. A large on-chip register file of 256 16-bit words was included to avoid traffic and power dissipation in the main memories. In this way, the operands required by the FUs were provided directly from a register file. As with other DSPs, a 32-entry instruction buffer was also included to handle loop instructions and reduce traffic to or from the program memory. Finally, all standard hardware components in CADRE were operated using self-timed techniques.

From the previous work using random plus speech data, a large percentage of power was found to be dissipated in the Multiply Accumulator Units (MAC units) amounting to about 50% of the overall power consumption as shown in Figure:3.1. A breakdown of the power consumption within one of MAC units is depicted in Figure:3.2. The multiplier dominated, followed by the adder. Therefore, the FU has been re-designed and re-implemented to demonstrate the energy saving improvement possible. To reduce the power consumption in the new FU, the major dominant components, the multiplier and adder, are designed and implemented with coherent low power techniques. The new four-way parallel asynchronous DSP which has been designed, implemented and tested is called CADRE-s (CADRE successor) and will be referred to as CADRE-s throughout this thesis.

3.2 CADRE-s Top-Level Architecture

The top-level architecture for CADRE-s is shown in Figure:3.3. The architecture has been designed to demonstrate the energy efficiency of the functional unit and the other advantageous features of CADRE-s such as four-way parallelism and configurable memories. In this top-level architecture, the 32x64 bit configuration memory has been attached to each FU and it stores the operation code (opcode). In contrast with the original

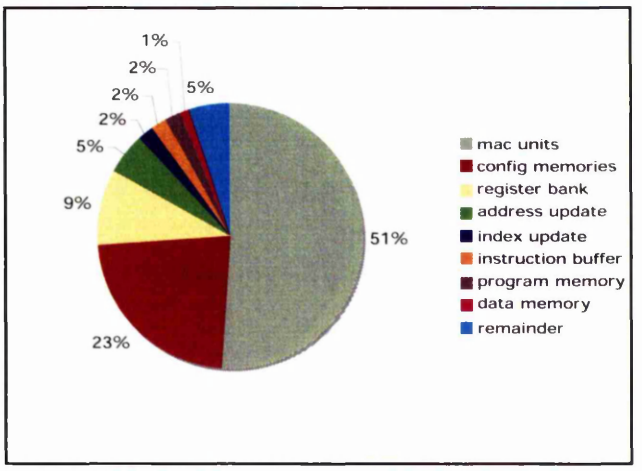


Figure 3.1: Average distribution of energy per block in CADRE

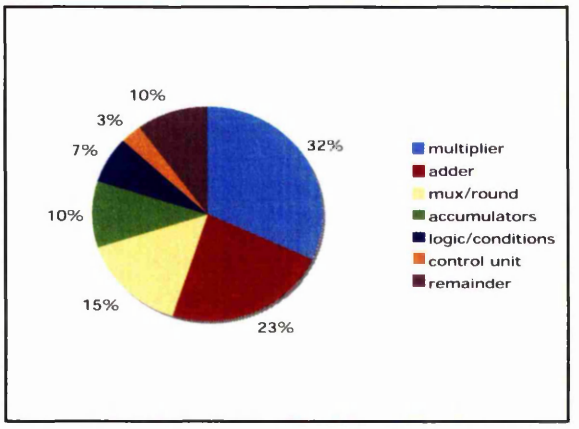


Figure 3.2: Breakdown of power consumption within the FU in CADRE

CADRE, the operands of each FU in the current architecture are fetched directly from on-chip RAMs (OP A and OP B). The new system consists of four FUs connected together with a global bus named the Global Interface Functional Unit (GIFU), whilst each pair of FUs are connected locally via the bus named Local Interface Functional Unit (LIFU). The output data from each FU is stored in another on-chip RAM (Result). Two RAMs are used for the top level control. One is the top-level 14-bit instruction and is stored in a program memory. The second one is the address RAM which effectively performs the function of a 16-bit program counter (PC).

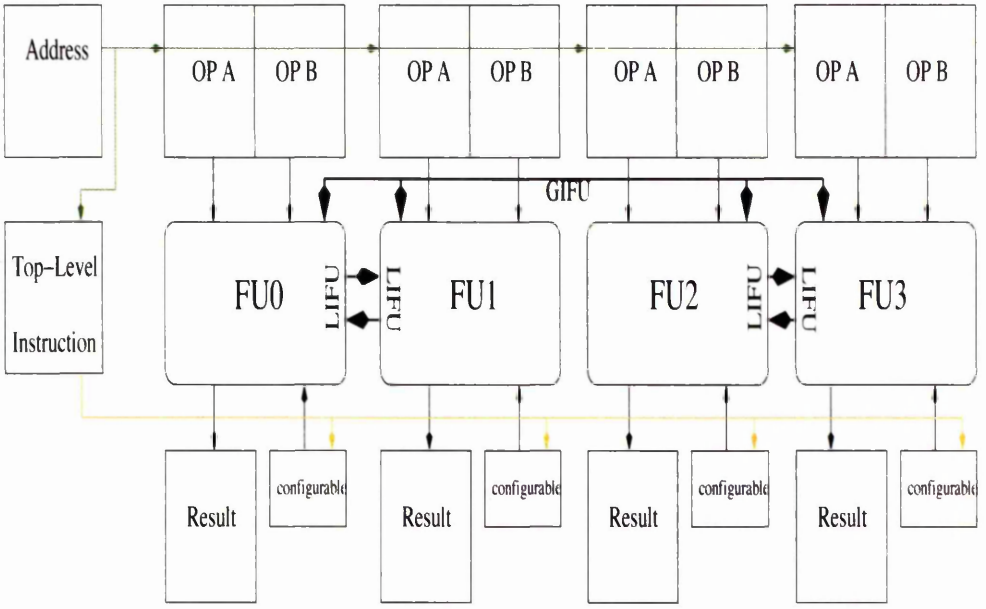


Figure 3.3: Top-level architecture for CADRE-s

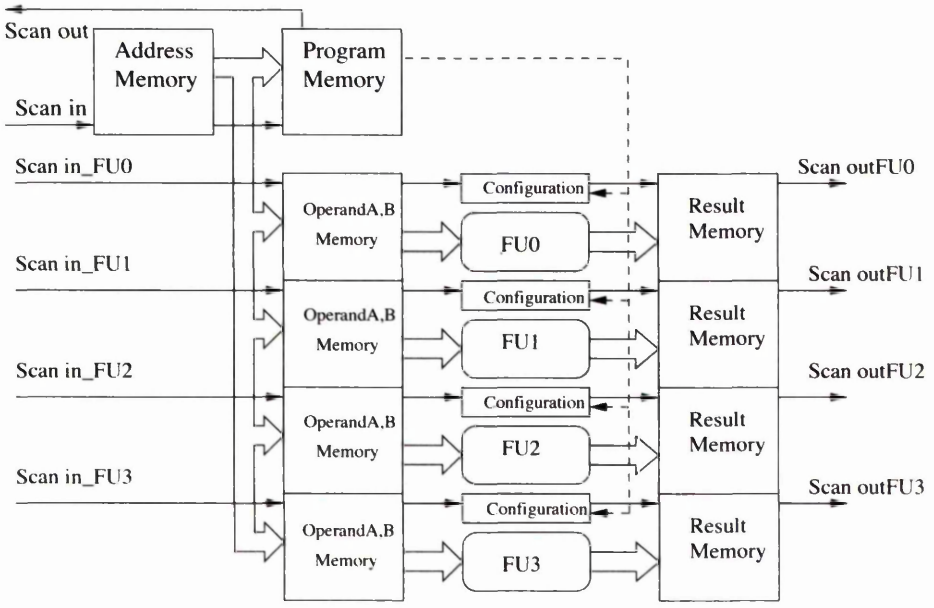


Figure 3.4: CADRE-s Data Organization

CADRE-s employs a scan path structure for downloading instructions/data and uploading results as shown in Figure:3.4. There are five separate serial scan paths in the design, one

is for the address and program memories and the others used for the operand, configuration and result memories. This makes the system fully testable as internal states can be controlled and monitored.

3.2.1 MIMD with VLIW encoding

In CADRE-s, each FU has its own operands, configurable and result memories. Hence, each FU can execute independently. It differs from the asynchronous superscalar pipeline proposed in[73] in that the FU in [73] has shared memories and registers. A special mechanism for the instruction dispatch unit is therefore required. Because CADRE-s contains four FUs which work independently, the instruction is relatively long. To minimize the length, the instruction has been divided into two parts. One part is stored in the configuration memory and the other part in the top level instruction memory. The configuration memory contains the FU opcode, input/output selection, and the shift condition whilst the top level instruction contains four enable signals (so each FU can turn its datapath on/off) plus four accumulator write enable control signals (one per FU) and a common 6-bit address for the configuration memory. Storing the 6-bit address of configuration memory not only reduces the number of instruction bits but the instruction can be compressed. This means that the instructions can be recalled when the same instructions but different data are required. Most DSP algorithms can take advantage of this feature to reduce the size of the program.

3.2.2 Five Stage Asynchronous Pipeline Organization

The CADRE-s pipelined processing unit is organized using self-timing techniques. An asynchronous pipeline operates at a variable rate determined by current conditions, unlike a single clock system, where the whole pipeline will be clocked at a rate determined by the worst-case delay in the slowest stage. Although, a multi-clocked system has been proposed in [74],[75],[76] to allow a variable rate operation, this method is limited by the number of the various clock cycles which are generated by the clock generator. In a clockless system, the next instruction can start as soon as the previous result is generated. Therefore, the self-timed system is able to operate at the average-case performance rather than worst-case. In addition, the absence of a global clock in asynchronous systems is also attractive for low-power and low EMI designs.

As mentioned before, the CADRE-s employs an asynchronous pipeline. However, all 4 FUs are forced to wait until the result of all four FUs become valid before starting the next instruction. Thus the FU's are synchronised at the start of an instruction. This feature still allows the CADRE-s to gain the advantages of a data dependent asynchronous system because each pipeline stage has a different cycle time rather than a fixed cycle time as in the clock system.

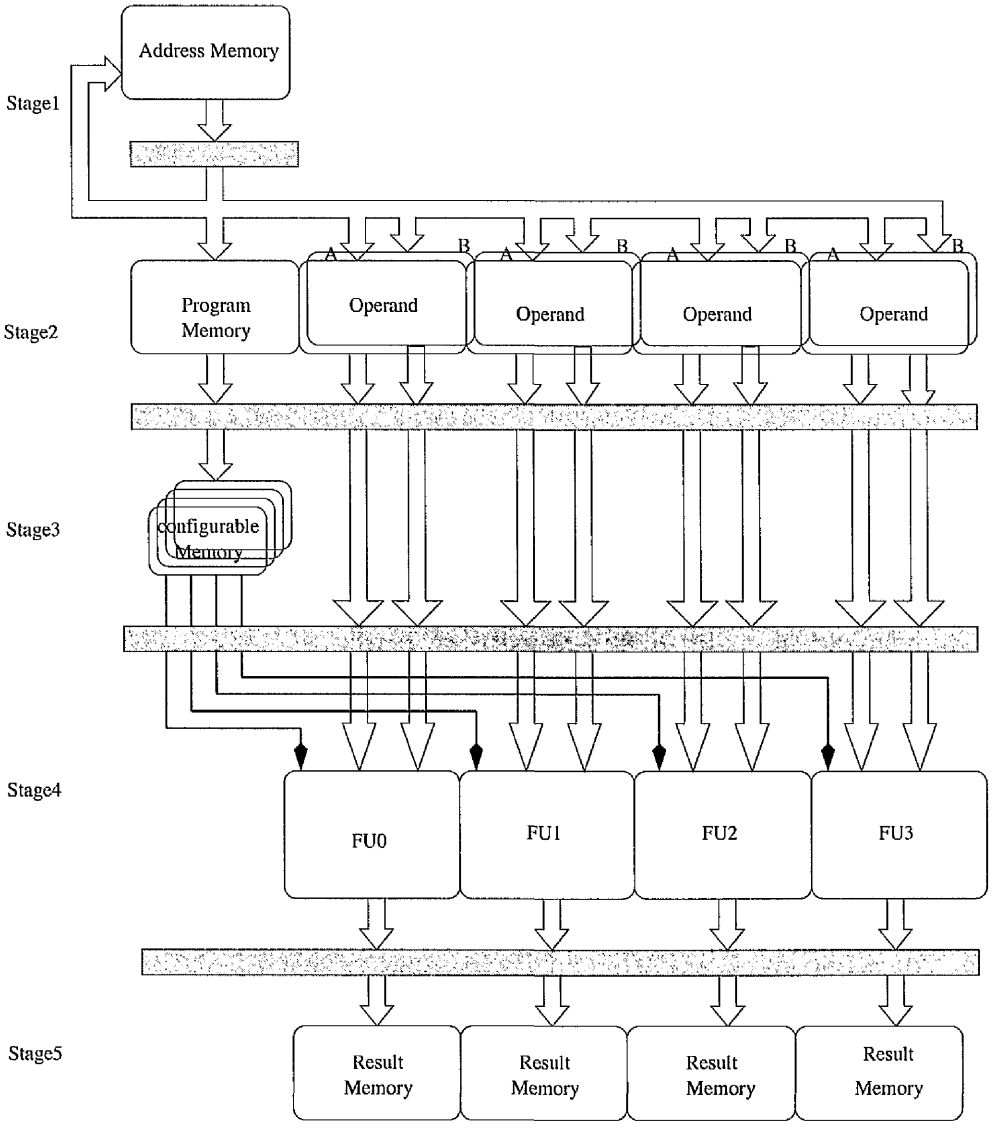


Figure 3.5: 5-Stage pipeline CADRE-s Organization

The principle components of the pipelined CADRE-s are illustrated in Figure:3.5. The shaded areas represent pipeline registers.

The16-b x 2K Address Memory, acts as a program counter. It keeps the next instruction address. This allows the program to jump or branch to any address within a range of 2^{10}. On reset, its address in initialised to zero. This address memory is used to address the Program and Operand memories in the next pipeline stage.

The program RAM, stores up to 2048 top-level instructions whilst the four Operand RAM blocks contain two 16x2K RAMs. These store operand A and B which are sent directly to the FUs in stage4 of the pipeline.

Stage3 comprises the four configuration memories, which store the FU encoded instructions. The address of the configuration memory is provided from the top-level Program Memory in stage2. The look up from the configuration memory is sent directly to the FUs in stage4 and are used to control the actions performed in the FUs.

In stage4, the four functional units (FUs) perform the arithmetic and logic functions required by the basic DSP instruction set. The FU also contains the (pass-transistor) shifter, which can shift the operand up to 32 positions. This FU has been designed and implemented using coherent energy saving techniques, the details of which are presented in Chapter4.

Finally, stage5 comprises four result RAMs, which store the FU result depending on the write-back enabling signal of the FU. The address of the result RAM is generated by an asynchronous 11-bit counter since writing to this memory is sporadic and the counter is incremented every time a result is required to be stored in the result RAM.

Thus the 5 stage pipeline can be regarded as having the following operations: next program address generation, top-level instruction fetch and operand fetch, configuration memory access, instruction execution and result writeback (if required).

These operations are shown in Figure:3.6(a) for a single FU. However, when the pipeline has filled with instructions, the flow becomes more complicated. For example, two MPYs and two ADDs running in parallel on the 4 FUs are shown in the flow of 5-stage pipeline

in Figure:3.6(b). The ADD instructions finish before the MPY instructions. However, the pipeline is forced to wait until the MPYs have finished. In Figure:3.6b, the FU operates in all execution time-slots.

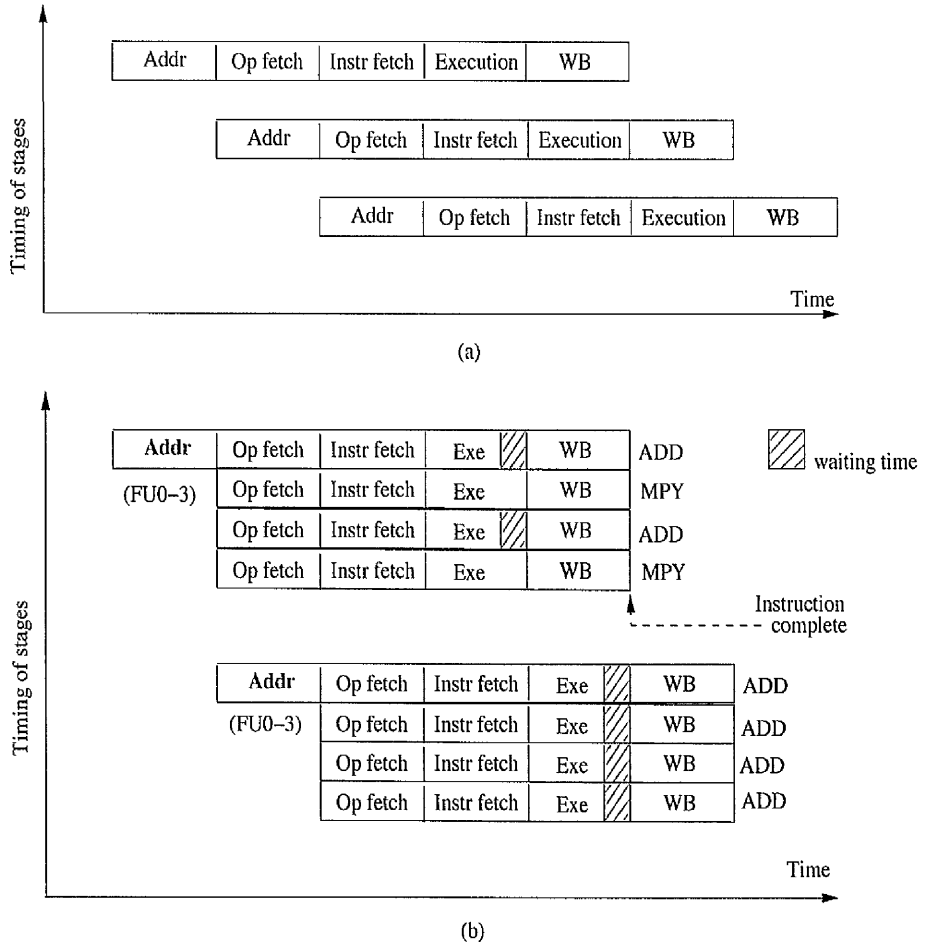


Figure 3.6: (a) CADRE-s 5-stage pipeline operation for a FU. (b) CADRE-s multi-time slot instruction of 5-stage pipeline operation of 4FUs.

3.3 FU Interconnection

The proposed DSP architecture is a hybrid of the MIMD (Multiple Instruction Multiple Data) and VLIW (Very Long Instruction Word) approaches. It allows each FU to execute different instructions at the same time and the VLIW encoding instructions can be loaded

into configurable memories in advance of execution. To implement data passing between FUs two types of bus interconnection are provided:

- Direct FU-to-FU connections, between pairs of adjacent FUs on 40-bit in and out unidirectional buses called the Local Interface FU (LIFU).

- Shared bidirectional bus between all four FUs. This 40-bit bus is called the Global Interface FU (GIFU).

Interconnection Power Awareness

With careful design, the direct and shared bus between FUs should be sufficiently energy efficient for future DSP requirements. In CMOS designs, the power is dominated by the switching power. Switching power is the power dissipation of charging and discharging the node capacitance. To minimise power on LIFU and GIFU, the LIFU bus is disabled at the transmitting end when not required to protect unnecessary data passing onto the bus. Meanwhile, data on the GIFU bus is only picked up by a receiver expecting data.

Load capacitance is another factor of the switching power dissipation. With regard to the power required to transmit and receive bus data, the gate capacitance can be neglected due to the wire capacitance being much more than the gate capacitance. Therefore, reducing the load capacitance can be done by optimizing the long wire. In this DSP architecture, each FU has been placed close to the others to make the wires as short as possible.

Self-timed Communication Protocol

The data communication on GIFU and LIFU uses a four-phase return to zero handshake protocol. This uses a request and an acknowledge signal to accompany the data; these signal its validity from the transmitter and acceptance by the receiver. The sequence used in as follows:

- Sender places valid data onto a bus.

- Sender issues a Request signal.

- Receiver accepts the data when it's ready.

- Receiver sends an acknowledges back to the Sender.

- Sender can remove the data and start the next communication.

- Receiver sees the removal of the request line and removes its acknowledge signal.

Transmitted data must be valid at the rising edge of the request signal and a return-to-zero has to occur before the next communication. LIFU has one input and one output bus with separate handshake control signals for each bus. When FU0 wants to send data to FU1 via LIFU, the LIFU out request signal is produced. FU1 will acknowledge back after the data has arrived. However, to avoid a deadlock on the LIFU communication, at every time slot each FU will produce a LIFU out request. Therefore, on the input side, a FU has to detect whether data is needed or not.

GIFU has only one bidirectional bus which connects all four FUs. When FU0 wishes to send data out onto GIFU, FU0 issues a private request signal to a global communication control unit (at the top level). If the request is granted, all other FUs request for GIFU are blocked. As soon as the target FU receives the data, it places an acknowledge on GIFU and this is broadcast to all FUs. Similar to the LIFU communication, a deadlock is prevented by allowing all FUs to send their GIFU requests but only one FU will be enabled so it can put its data onto the bus.

3.4 Testability

When integrated circuit get larger, testing needs to be seriously considered. This is because in a large system, it is difficult to access internal nodes during testing and the number of test vectors required to test all aspects of the design, with a high probability of not missing any faults, grows exponentially. This results in larger testing time and incomplete testing.

CADRE-s considers the testing to be one of the most important issue for realizing real systems, so CADRE-s includes circuitry whose sole function is to help with the on-chip

testing. Built-in testing with scan-design circuitry has been attached. The testing is accomplished by operating the circuits in two different modes: regular operation and test mode. In test mode, the data can be shifted through additional shift registers of each internal section. After the results are produced in the regular operation, these outputs are shifted serially out of the CADRE-s and compared to the known results. The major limitation of this technique is the large time required to shift in a long stream of data and the time to shift out the outputs of the acyclic logic circuitry.

In 1978, Stewart[78] proposed a technique called the scan/set technique related to scan-path design. This technique used shift-register flip-flops for shifting in data (test vectors) or shifting out results (test-result vectors) which were separate from the shift registers and other flip flops actually used during regular system operation. In addition, multiplexers are required that select system inputs from either the regular inputs or from the additional test-shift registers. With this scan/set technique, the output vectors can be shifted out during regular system operation. CADRE-s uses a normal scan-path with register flip-flops shared between regular operation and the test mode because the main objective of CADRE-s is not the speed of testing.

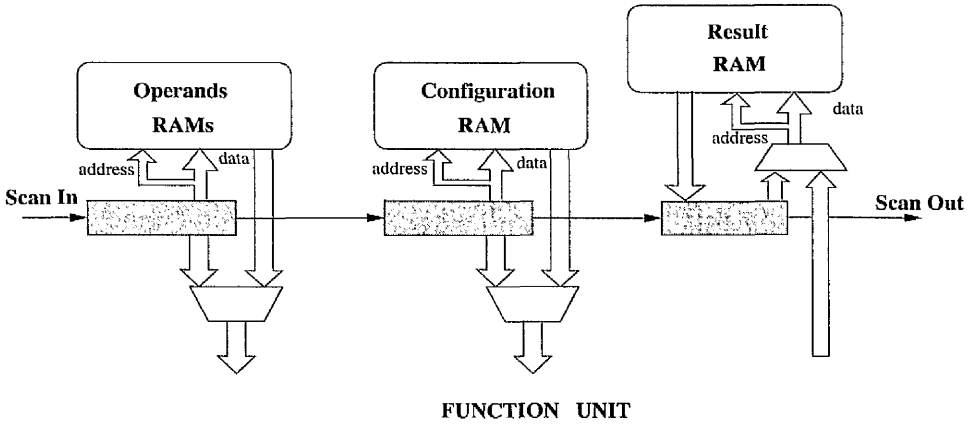


Figure 3.7: Scan-path in a FU of CADRE-s

Figure:3.7 shows an example of the scan-path used in a FU. This scan path carries a long input containing the A and B operands, the configuration (FU instruction) data and the result RAM data and their addresses. When both data and its address have been shifted to

the correct position, the clock will rise to input the data into the RAMs. However, (as a fall back position) if there is something wrong with the RAMs, there is the option to feed the scan input directly to the FU. When the algorithm execution is completed, the data in the result RAM can be read and shifted out via the scanout. This increases flexibility for the testing and operating of the chip.

The test board for CADRE-s is shown in Figure:3.8. The FPGA is used to transmit the binary code of the operand and instructions stored in the RAM/ROM to CADRE-s via the scan in path whilst the results are serially shifted out of CADRE-s via the scanout and stored in RAM or ROM controlled by FPGA.

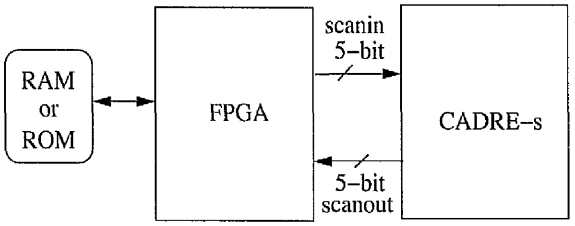


Figure 3.8: Test board configuration

3.5 The example of Four-way Parallelism

To illustrate how the parallelism available can be exploited on CADRE-s, the execution of the FIR filter algorithm expressed by the equation:

$$y(n) = \sum_{k=0}^{M-1} C_k X_{(n-k)}$$

is presented. A simple way to map the FIR filter is if each FU processes four y(n) terms in parallel and each FU performs a quarter of the operations for each term. At the end, a final summation of the four partial sums is then performed. The example is shown in Table3.1 where the arithmetic operation format is *operation source1, source2, destination* for the multiply instruction (MPY), *operation source1, source2, source3, destination* for

multiply-accumulate instruction (MAC) and *operation source1 source2 destination* for the add instruction. NOP is a no-operation instruction. Source1 and 2 are 16 bits whilst source3 and the destination are a 40-bit accumulator[A to D]. In addition, the notation *fu[0-3]* is used to indicated which functional unit is involved. Twenty five time slots are required to perform the example of the FIR filter with 20 coefficients (20-tap) and n = 0, 1, 2 and 3 in Table3.1 and thirteen configuration memory instructions (in each FU) are required. This table also shows that the parallel architecture can gain a large performance improvement by about factor of three compared to a single execution unit that requires 80 cycles time to complete this FIR example.

Table 3.1: Mapping a arrangement FIR 20-tap filter (n = 0, 1, 2, 3) onto four FUs

FU0	FU1	FU2	FU3
mpy X_n,C_0, a	mpy X_{n-1},C_1, a	mpy X_{n-2},C_2, a	mpy X_{n-3},C_3, a
mpy X_{n-1},C_0,b	mpy X_{n-2},C_1,b	mpy X_{n-3},C_2,b	mpy X_{n-4},C_3,b
mpy X_{n-2},C_0,c	mpy X_{n-3},C_1,c	mpy X_{n-4},C_2,c	mpy X_{n-5},C_3,c
mpy X_{n-3},C_0,d	mpy X_{n-4},C_1,d	mpy X_{n-5},C_2,d	mpy X_{n-6},C_3,d
mac X_{n-4},C_4,a	mac X_{n-5},C_5,a	mac X_{n-6},C_6,a	mac X_{n-7},C_7,a
mac X_{n-5},C_4,b	mac X_{n-6},C_5,b	mac X_{n-7},C_6,b	mac X_{n-8},C_7,b
⋮	⋮	⋮	⋮
mac X_{n-16},C_{16},a	mac X_{n-17},C_{17},a	mac X_{n-18},C_{18},a	mac X_{n-19},C_{19},a
mac X_{n-17},C_{16},b	mac X_{n-18},C_{17},b	mac X_{n-19},C_{18},b	mac X_{n-20},C_{19},b
mac X_{n-18},C_{16},c	mac X_{n-19},C_{17},c	mac X_{n-20},C_{18},c	mac X_{n-21},C_{19},c
mac X_{n-19},C_{16},d	mac X_{n-20},C_{17},d	mac X_{n-21},C_{18},d	mac X_{n-22},C_{19},d
add fu1:a,a,a	add fu0:b,b,b	nop	add fu2:a,a,a
add fu3:a,a,a [y(0)]	add fu0:c,c,c	nop	add fu2:b,b,b
add fu1:d,d,d	add fu3:b,b,b [y(1)]	nop	add fu2:c,c,c
nop	nop	add fu3:d,d,d	add fu1:c,c,c [y(2)]
nop	nop	add fu0:d,d,d [y(3)]	nop

The possibility of reducing the power consumption at the algorithmic level depends on how the instructions are selected and mapped onto the parallel structure. As can be seen from the FIR filter example, the FU has to get new operands from the memory in every cycle. This results in a higher amount of memory access which dissipates power on each access. It also increases the switching activity within the multiplier and data buses. To

reduce the amount of memory access, the technique of input buffering can be used if the amount of data is high enough to make a good trade-off between the power saving of minimal memory access and the power of buffers. On the other hand, the way that operations are mapped to the hardware dramatically reduces the amount of switching activity within the multiplier because the value on the data bus within each FU is held constant for four successive instructions compared to a different arrangement for accessing data which minimises the number of time slots; this is illustrated in Table3.2.

Here in Table3.2, a new coefficient is fetched at each time slot but is the same for all four FUs. Comparing it with the arrangement of Table3.1, the switching activity in the datapath of Table3.1 is reduced by about a factor of four. However, the different data arrangement FIR in Table3.2 requires only 20 time slots to produce four outputs (y_0,y_1,y_2 and y_3) and only three configuration memory instructions per FU are required. This is a trade-off between minimizing switching activity with the number of time slots and configuration memory instructions. In chapter7, the power and energy consumption of the FIR filter with these two different mapping styles are presented and discussed in more detail.

Table 3.2: A different data arrangement for the FIR 20-tap filter (n = 0, 1, 2, 3) mapped onto four FUs

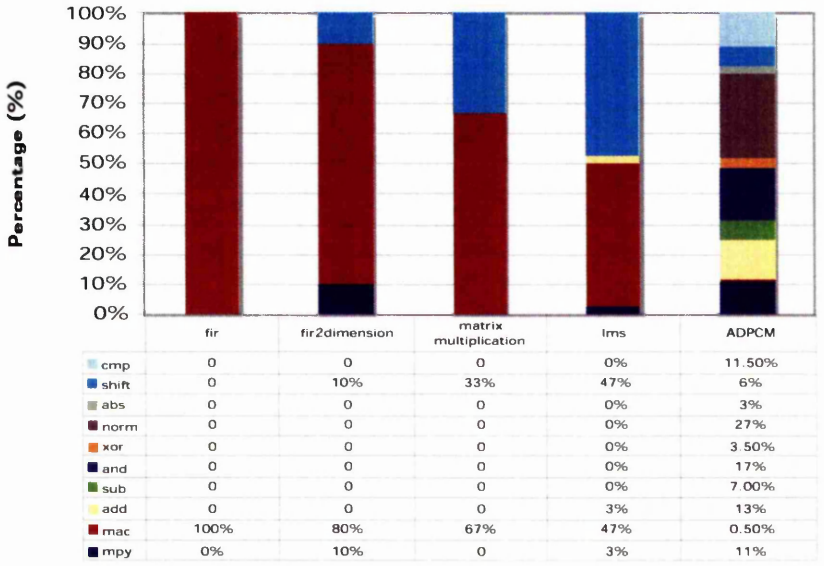
FU0	**FU1**	**FU2**	**FU3**
mpy X_n,C_0, a	mpy X_{n-1},C_0, a	mpy X_{n-2},C_0, a	mpy X_{n-3},C_0, a
mac X_{n-1},C_1,a	mac X_{n-2},C_1,a	mac X_{n-3},C_1,a	mac X_{n-4},C_1,a
mac X_{n-2},C_2,a	mac X_{n-3},C_2,a	mac X_{n-4},C_2,a	mac X_{n-5},C_2,a
mac X_{n-3},C_3,a	mac X_{n-4},C_3,a	mac X_{n-5},C_3,a	mac X_{n-6},C_3,a
⋮	⋮	⋮	⋮
mac X_{n-16},C_{16},a	mac X_{n-17},C_{17},a	mac X_{n-18},C_{18},a	mac X_{n-19},C_{19},a
mac X_{n-17},C_{17},a	mac X_{n-18},C_{17},a	mac X_{n-19},C_{18},a	mac X_{n-20},C_{19},a
mac X_{n-18},C_{18},a	mac X_{n-19},C_{17},a	mac X_{n-20},C_{18},a	mac X_{n-21},C_{19},a
mac X_{n-19},C_{19},a [y_0]	mac X_{n-20},C_{17},a [y_1]	mac X_{n-21},C_{18},a [y_2]	mac X_{n-22},C_{19},a [y_3]

Chapter 4: Energy Efficient Functional Unit - The Adder and Multiplier

When an energy efficient design becomes vital, both high performance and low power design techniques need to be applied to the computing unit such as a functional unit (FU) in a general DSP. Making a high performance and low power arithmetical processing unit is not easy. However, the complex arithmetic functions such as division, multiplication, multiply accumulation and modulation share a common function - addition. Addition can be implemented by an XOR which is essentially a multiplexer circuit. Furthermore, the basic logic functions -XOR and multiplexer- are well suited to a pass transistor logic implementation. The number of transistor is therefore significantly reduced and this leads to a saving in the power dissipation and also increases the performance. Consequently, the functional unit based on pass transistor circuits is a good candidate for a compute intensive arithmetic unit of an energy efficient DSP. In addition, coherent energy saving design techniques have been employed at every design level - architecture, logic and circuit - of the FU.

4.1 Functional Unit Architecture

As is well known, arithmetic operations such as multiplication, addition and multiply-accumulate are the most commonly specified operations and occur in most cycles of a general DSP. This is confirmed by analysing the assembly codes of the DSP kernels for a FIR filter, 2 dimensional FIR filter using a 3x3 coefficient mask, matrix multiplication, Least Mean-Square (LMS) adaptive filter, and Adaptive Differential Pulse Code Modulation (ADPCM). The code for these was based on that for Motorola's DSP56K family[26]. These kernel benchmarks have been analysed by the author for the frequency of arithmetic instruction use and the results are shown in Figure4.1. Looking at the results,



	fir	fir2dimension	matrix multiplication	lms	ADPCM
cmp	0	0	0	0%	11.50%
shift	0	10%	33%	47%	6%
abs	0	0	0	0%	3%
norm	0	0	0	0%	27%
xor	0	0	0	0%	3.50%
and	0	0	0	0%	17%
sub	0	0	0	0%	7.00%
add	0	0	0	3%	13%
mac	100%	80%	67%	47%	0.50%
mpy	0%	10%	0	3%	11%

Figure 4.1: Analyse arithmetic instructions in DSP kernels

the MAC operation is significant in all kernels except the ADPCM. Surprisingly, considerable shifting is performed in the matrix multiplication and LMS code; this is to scale the numbers so they remain within range. The high use of the NORM instruction in the ADPCM code can be attributed to it including the conversion of PCM A-Law to/from U-Law[27]. Since overall the MAC operations dominate, the multiplier and its adder can be considered as a major source of power dissipation in a system. Consequently, great attention has been accorded the architecture, logic and circuit design of these features to ensure their better energy efficiency. Apart from these operations, special functions such as Hamming distance are also needed for complex DSP applications such as image processing and speech recognition whilst the normalization operation is required to convert from fixed-point to floating point numbers. Thus a special unit combining these two functions has been designed and implemented in this FU. The top-level architecture of the processing unit as shown in Figure:4.2, has been divided into two main components: the first component is the functional unit performing all arithmetic and logic operations such as multiplication, addition, multiply-accumulate, Hamming distance, normalization and shifting; the other is the 32x64 configuration memory which contains the encodings opcode i.e. the FU instruction.

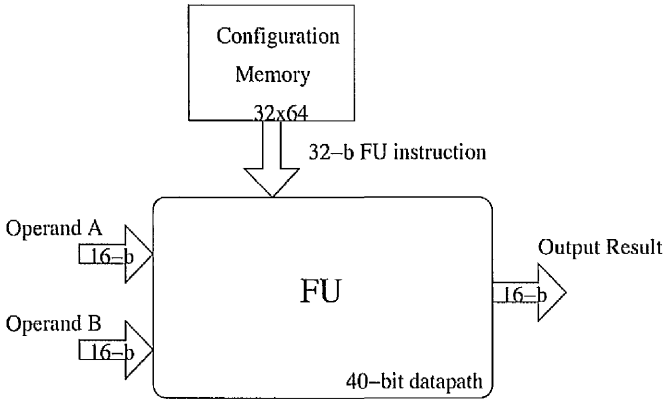


Figure 4.2: Top-level functional unit architecture

The critical path of the FU architecture performs the arithmetic and logic operations. Two 16-bit operands, A and B, are input to execute in the FU as soon as the FU instruction has been decoded. One of the differences between the FU in a general processor and a DSP are that the FU in a DSP requires a 40-bit datapath to support the fractional number calculations which are performed most of the time. Therefore, common arithmetic operations dominate the power consumption. In addition, some special computing such as Hamming Distance and Normalization are also needed to meet the performance target of digital signal processing. Thus the FU requires careful design.

The FU has a 16-bit output bus used to transfer the data out of the FU. At the end of some computations such as in the FIR and DCT algorithms, the output needs to be stored in the result RAM outside the FU. Since the communication between FU and RAM rarely happens during the processing of these algorithms, it should not significantly slow down the whole DSP when it occurs.

As mentioned before, the FU in a DSP has a heavy workload. Lowering the power dissipation of the FU will therefore result in a significant power saving. However, the overall performance is still required to meet its target - a minimum of 100 MIPs[28] for a 3G mobile phone application. To meet these conflicting aims, simple pass logic transistor is adopted in a full custom design; this makes the FU less complex resulting in a smaller and faster design. In turn, this approach results in an energy efficient FU.

4.1.1 Instruction Set

The instruction set of a FU has two types of orders: computational and data movement; these can be performed concurrently.

1) Computational instructions contain arithmetic and logical operations; the arithmetic instructions include Hamming distance, normalization, absolute maximum/minimum, maximum/minimum, shift, ADD, SUB, MPY and MAC (multiply and accumulate). The output destination of these operations is an accumulator register (A to D) within the FU.

2) Data Movement Instructions instructions process the data movement in or out of the FU. These instructions include an accumulator register to accumulator register transfer within the FU itself, an accumulator register to an accumulator register in another FU, or a movement of data to the output RAM.

4.1.2 FU datapath

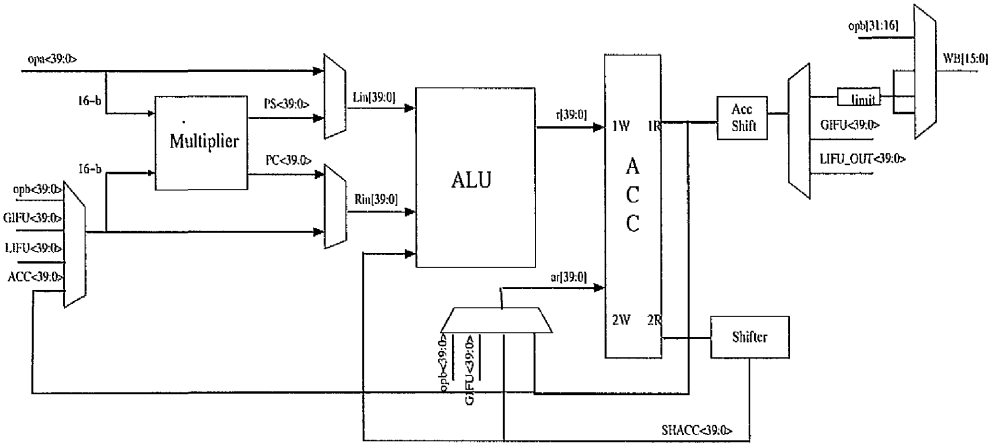


Figure 4.3: Functional Unit datapath

At an architectural level, a resource sharing technique has been used to minimise the logic gates required. Since the multiply-accumulate is a frequent operation in a DSP, the same adder can be used by both the addition and multiplication instructions. The addition of the multiplier outputs, the Partial Sum (PS) and Partial Carry (PC), is performed by the sole adder in the FU. This architecture, as shown in Figure:4.3, is unusual in that most DSP

designs have a dedicated adder for the MPY and MAC operations and a separate adder is provided for addition. Having just one adder in the FU significantly reduces the amount of logic required in the datapath which in turn reduces power. This uses the concept of resource sharing. The MAC operation is performed in most cycles of general DSP applications. The multiplication is performed before the addition and the multiplier also requires the adder. If an adder is located in the multiplier and separately for doing the ADD instruction, full carry propagation can occur twice (once for multiplication and the other for addition); this will slow the MAC down. Thus sharing the adder logic can gain a significant power saving, reduce the amount of logic and enhance performance.

However, if a single adder in the FU is adopted, more than 2 operands may need to be summed in the adder. For example, the MAC operation requires three inputs -two for PS and PC of the multiplier and one for the accumulator (with or without shift). Furthermore, the rounding option in the addition, subtraction and multiplication orders results in another (constant-value) input. An adder in the ALU performing a four-operand addition (PS and PC from the multiplier, an accumulating value from the accumulators and a rounding constant) has a better performance compared to the usual 2 input adder because the four input carry save adder, can exploit the use of 3 input to 2 output compressors or 4 to 2 compressors to reduce the carry propagation time.

Finally, transparent latches are inserted in front of the multiplier, the ALU, the second write port (2W) of the four accumulators (ACC) and the write back (WB) bus to prevent unnecessary switching in the datapath. This approach directly reduces the dynamic power dissipation.

As can be seen in Figure:4.3, the FU datapath allows the data to move in and out in parallel with the data processing. Conflicts in the data communication between each FU or a FU and RAM can be avoided since program scheduling at the software or compilation level can be applied to solve any such data conflict problem.

4.1.3 A Data Flow

The best method to explain the execution of a functional unit instruction is to show the data flow in the FU datapath. The data flow is divided into two groups; the first group is data processing instructions and the second group is data movement instructions.

Data processing instructions

A data processing instruction requires two operands that are fetched from the operand RAMs. The operation corresponding to the opcode then takes place in the multiplier and ALU. Finally, the result is written into an accumulator register. The data processing instruction -MAC- is illustrated in Figure:4.4. Two operands are input to the multiplier and two outputs -PS and PC- produced. These are then summed together with the data on the shift accumulated (SHACC) bus by the ALU. The result is placed in an accumulator register (ACC) via the 1W write port.

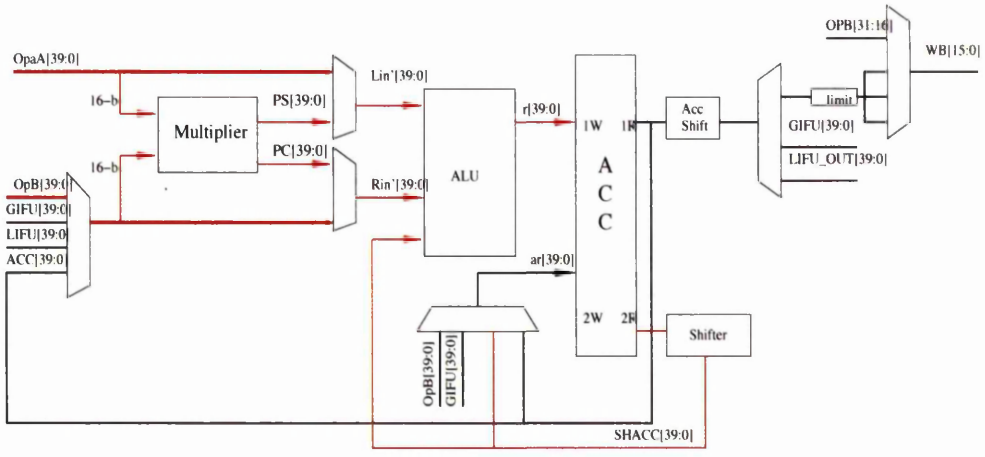


Figure 4.4: MAC instruction flow in FU datapath

Data movement instructions

A data movement instruction involves all data transfers between an accumulator ACC of a FU to the ACC of another FU, or the data transfer between a FU and a RAM as shown in Figure:4.5. For example, the red lines show the movement from a RAM or from other

FUs to an accumulator register. The blue line presents the data movement between an ACC and RAM whilst the green and brown line show the data movement between FUs via GIFU and LIFU, respectively. Finally, the data can use the FU as a bypass unit for a register to register move within the Register File via the gold line. This feature is built to support future requirements when the data has no need to execute as only movement is required. All data movements using the different sources are allowed to perform concurrently. Rather than leave the datapath largely idle during data movement, data processing instructions can also be executed in parallel. However, the accumulator read operation has to occur before the write operation is performed to avoid data hazard problems. More than one operation can occur concurrently within a single timeslot because of the number of available paths and possible input sources at different points. This parallelism gives the system great flexibility, and being able to access and then operate on data from another FU in a timeslot enhances the performance. However, this requires more complex self-timed control circuits.

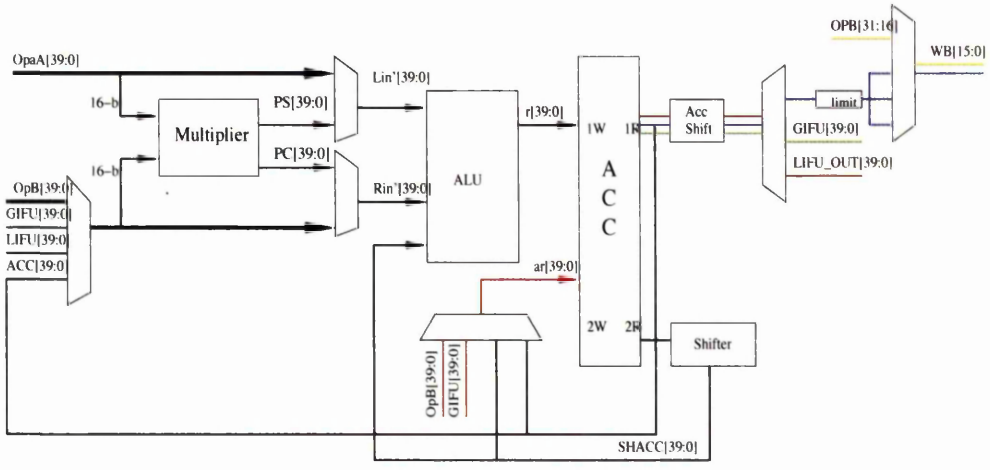


Figure 4.5: Data movement flow in FU datapath

4.2 Addition

A vital arithmetic operation which analysis reveals is frequently used in DSP algorithms as shown in Figure:4.1 is addition. The adder is on the critical path which determines the overall performance of the system. Furthermore, it also dissipates significant power due

to its high usage. It is therefore essential to minimise the power and logic required for this operation.

In an addition, the carry propagation is a bottleneck for the speed. Many fast carry generator schemes such as carry-select, Manchester carry path, carry look-ahead, binary look-ahead, carry-skip and conditional sum need therefore to be considered. All detail of these adders can be found in[88]. These classical fast carry generators can be effected particularly efficiently by pass transistor logic implementing the multiplexer function. The carry select scheme, based on multiplexers, is one type of fast adder. However the number of multiplexers increases exponentially with word length as reported in [79]. The carry skip adder seems to give the lowest power dissipation compared to the other adders. Unfortunately, it takes a longer time to complete the addition compared to other carry accelerator methods[80]. The carry propagation delay of the Manchester adder is about $(W+1)t_{mux}$[81], where W is the word length and t_{mux} is the delay through a multiplexer, whilst the carry-look-ahead adder has a carry propagation delay of $(\log_2 W)t_{mux}$ but has a large fan-in of n. Finally, the carries in a conditional sum adder are generated in a similar manner to the carry-look-ahead and has the same delay; however it requires a larger silicon area[58]. The carry save adder is therefore adopted for use in the FU because it offers a good compromise between performance and power[82]; it is often chosen for high speed application for this reason. It is also widely used in a time multiplexed adder with many operands. Thus it is employed in the first stage of the adder in the FU where four inputs are required. However, the carry-look-head tree adder is adopted and mapped onto pass transistor circuits for the final (full carry propagation) addition to improve performance.

The compressor (which combines n inputs into m outputs where $m < n$) is employed in the first stage of the adder in the FU where four inputs are required. However, the carry-look-ahead tree adder is adopted and mapped onto pass transistor circuits for the final (full carry propagation) addition to improve performance. The first stage of the adder in the FU, involves adding up to four variables as shown in Figure:4.3. Compression is first performed using 4-2 pass gate compressors; these generate two outputs from the four inputs, and the two outputs are then added in a carry-look-ahead (CLA) tree. The carry-look-ahead equation, given in [81], has been modified to enable an easy efficient mapping

onto pass transistor circuits. The carry is generated across blocks of four bits because simulation from [81], showed that this offered the best speed-power product. The carry-look-ahead tree is employed as the carry generator for the second stage of the adder following the post layout of simulations of the different adder types implemented on 0.18μm running at 1.8V as shown in Table4.1. The adder circuits have been simulated by the author using random numbers running on the Nanosim Simulator[150], see more detail in Appendix C.

Table 4.1: The comparison of 40-b adders

Adder Type based on pass transistor logic	Max. Add time (ns)	Avg. Power (mW)	Energy (pJ)	No. of Devices
CLA tree - 4 bit blocks	2.6	0.43	1.12	3080
CLA tree - 2,8,16,8,4,2 blocks	1.5	0.88	1.32	3057
CLA	9.1	0.32	2.91	1520
Carry save - ripple adder blocks	3.8	0.78	2.96	3620
Carry save - CLA adder blocks	3.6	0.75	2.70	3540

The simulation results of the different adder types show that carry-look-ahead tree with 4-bit per block gives the lowest energy whilst the fastest adder is carry-look-ahead tree with blocks of 2, 8, 16, 8, 4 and 2 bits. However, a carry-look-ahead adder is suitable for low power applications where the battery lifetime is not a concern.

4.2.1 Carry-Look-ahead Tree Background

In this section, equations for each carry bit in a 4-bit block are evolved and transformed so that they can be mapped onto a multiplexer implementation. In the addition operation, the carries are generated using the iterative formula:

$$C_{i+1} = C_i P_i + G_i$$

where $P_i = A_i + B_i$ or $K_i = \overline{A_i + B_i}$

$$G_i = A_i B_i$$

here C_{i+1} is the generated carry, C_i is the carry in, P_i is a propagate bit, the inverse of P_i is K_i (Kill propagate) and G_i is a generate carry bit. However, the equation of C_{i+1} can be rewritten in terms of a multiplexer implementation as follows:

$$C_{i+1} = C_i P_i + \overline{C}_i G_i \quad \text{... Eq. 4.1}$$

Because an adder tree can reduce the carry propagate time to be $(\log_2 W)t_{mux}$, it is therefore used as the carry generation approach. Considering a block of 4 bits with the carry into the lsb of the block being C_i, the generating functions can be defined as $K_j(i)$ and $G_j(i)$ and have been adapted by the author to map onto multiplexers from the equations given in [83]:

$$\begin{aligned} \overline{K}_{j+1}(i+1) &= \overline{K}_j(i)\overline{K}_j(i+1) + K_j(i)G_j(i+1) \\ G_{j+1}(i+1) &= G_j(i)\overline{K}_j(i+1) + \overline{G}_j(i)G_j(i+1) \end{aligned} \quad \text{... Eq. 4.2}$$

where the initial values are $\overline{K}_i$ and G_i where j and i are the recursive level and bit position respectively. Note that the updating functions K_j and G_j requires only two multiplexers; these have common inputs but different select signals. Therefore, the carry generation from equation (4.1) can be written based on these recursive functions as follows:

$$\begin{aligned} C_{i+1} &= C_i \overline{K}_0(i) + \overline{C}_i G_0(i) \\ \overline{C}_{i+1} &= C_i K_0(i) + \overline{C}_i \overline{G}_o(i) \end{aligned} \quad \text{... Eq. 4.3}$$

Hence

$C_{i+2} = C_{i+1}\overline{K}_0(i+1) + \overline{C}_{i+1}G_0(i+1)$

$= ([Ci\overline{K}_0(i) + \overline{C}_i G_0(i)]\overline{K}_0(i+1) + [C_i K_0(i) + \overline{C}_i \overline{G}_0(i)]G_0(i+1))$ ..... Eq. 4.4

$= C_i[\overline{K}_0(i)\overline{K}_0(i+1) + K_0(i)G_0(i+1)] + \overline{C}_i[G_0(i)\overline{K}_0(i+1) + \overline{G}_0(i)G_0(i+1)]$

Assigning $[\overline{K}_0(i)\overline{K}_0(i+1) + K_0(i)G_o(i+1)] = \overline{K}_1(i+1)$ and $[G_0(i)\overline{K}_0(i+1) + \overline{G}_0(i)G_o(i+1)] = G_1(i+1)$

$$C_{i+2} = C_i\overline{K}_1(i+1) + \overline{C}_iG_1(i+1)$$
$$\overline{C}_{i+2} = C_iK_1(i+1) + \overline{C}_i\overline{G}_1(i+1) \quad \text{... Eq. 4.5}$$

Similarly

$C_{i+3} = C_{i+2}\overline{K}_0(i+2) + \overline{C}_{i+2}G_o(i+2)$

and replacing C_{i+2} and $\overline{C}_{i+2}$ in the equation for C_{i+3} gives

$C_{i+3}= [C_i\overline{K}_1(i+1) + \overline{C}_iG_1(i+1)]\overline{K}_0(i+2) + [C_iK_1(i+1) + \overline{C}_i\overline{G}_1(i+1)]G_o(i+2)$

$= C_i[\overline{K}_1(i+1)\overline{K}_0(i+2) + K_1(i+1)G_0(i+2)] + \overline{C}_i[G_1(i+1)\overline{K}_0(i+2) + \overline{G}_1(i+1)G_o(i+2)]$

As with C_{i+2}, assigning $\overline{K}_1(i+2) = [\overline{K}_1(i+1)\overline{K}_0(i+2) + K_1(i+1)G_0(i+2)]$ and $G_1(i+2) = [G_1(i+1)\overline{K}_0(i+2) + \overline{G}_1(i+1)G_o(i+2)]$, C_{i+3} can be rewritten as follows:

$C_{i+3} = C_i\overline{K}_1(i+2) + \overline{C}_iG_1(i+2)$ and

$\overline{C}_{i+3} = C_iK_1(i+2) + \overline{C}_i\overline{G}_1(i+2)$

When C_{i+3} is substituted in to the equation for C_{i+4}, the equation becomes:

$C_{i+4} = C_i[\overline{K}_1(i+2)\overline{K}_0(i+3) + K_1(i+2)G_0(i+3)] + \overline{C}_i[G_1(i+2)\overline{K}_0(i+3) + \overline{G}_1(i+2)G_o(i+3)]$

However, C_{i+4} cannot be produced until $\overline{K}_1(i+2)$ and $G_1(i+2)$ are ready. The equation is therefore rewritten to form C_{i+4} faster as:

$C_{i+4} = C_{i+3}\,\overline{K}_0(i+3) + \overline{C}_{i+3}G_0(i+3)$

$= [C_{i+2}\,\overline{K}_0(i+2) + \overline{C}_{i+2}G_0(i+2)]\overline{K}_0(i+3) + [C_{i+2}\,K_0(i+2) + \overline{C}_{i+2}\overline{G}_0(i+2)]G_0(i+3)$

$= C_{i+2}\,[\overline{K}_0(i+2)\overline{K}_0(i+3) + K_0(i+2)G_0(i+3)] + \overline{C}_{i+2}\,[G_0(i+2)\overline{K}_0(i+3) + \overline{G}_0(i+2)G_0(i+3)]$

Assigning $[\overline{K}_0(i+2)\overline{K}_0(i+3) + K_0(i+2)G_0(i+3)] = \overline{K}_1(i+3)$ and $[G_0(i+2)\overline{K}_0(i+3) + \overline{G}_0(i+2)G_0(i+3)] = G_1(i+3)$ and replacing C_{i+2} with equation (4.5) yields:

$C_{i+4} = [C_i\overline{K}_1(i+1) + \overline{C}_iG_1(i+1)]\,\overline{K}_1(i+3) + [C_iK_1(i+1) + \overline{C}_i\overline{G}_1(i+1)]G_1(i+3)$

$= C_i[\overline{K}_1(i+1)\overline{K}_1(i+3) + K_1(i+1)G_1(i+3)] + \overline{C}_i[G_1(i+1)\,\overline{K}_1(i+3) + \overline{G}_1(i+1)G_1(i+3)]$

Assigning $[\overline{K}_1(i+1)\overline{K}_1(i+3) + K_1(i+1)G_1(i+3)] = \overline{K}_2(i+3)$ and $[G_1(i+1)\overline{K}_1(i+3) + \overline{G}_1(i+1)G_1(i+3)] = G_2(i+3)$ reduces the equation of C_{i+4} to:

$C_{i+4} = C_i\overline{K}_2(i+3) + \overline{C}_iG_2(i+3)$ and

$\overline{C}_{i+4} = C_iK_2(i+3) + \overline{C}_i\overline{G}_2(i+3)$

This completes the formation of the mapping of 4 bits of the carries in the carry-look-ahead tree onto a multiplexer implementation.

4.2.2 Carry-Look-Ahead Tree Implementation

As the equations of the carry-look-ahead tree show in the previous section, the circuit can be implemented using multiplexer functions (excluding the formation of the propagate and generate bits, P_i and G_i). A bypass/exchange function is required in the C_{i+2}, C_{i+3} and C_{i+4} carry generator circuits. This takes 2 inputs and either passes them straight through or swaps them, depending on the control input. This can be implemented with two 2-input multiplexers which logically have the functionality:

$Z_1 = S_1IN_1 + \overline{S}_1IN_2$

$Z_2 = S_2IN_1 + \overline{S}_2IN_2$

This leads to the block for the 4-bit carry-look-ahead tree implemented entirely by the multiplexers as shown in Figure:4.6

Using a 4-bit carry-look-ahead tree unit, the 40-bit carry-look-ahead tree can be built by 10 blocks of these 4-bit units. The carry propagates from the bottom (C_0) bit to the top carry bit (C_4) of the 4-bit unit within $3t_{mux}$ and uses $12t_{mux}$ for the delay carry-chain of the 40-bit carry-look-ahead tree. The delay of a W-bit CLA-tree implemented by 4-bit unit is therefore $((W/4)+2)t_{mux}$ as shown in Figure:4.7 for an 8-bit adder where the multiplexer generating the sum is shown only for the LSB in each 4-bit unit. The times t_0,

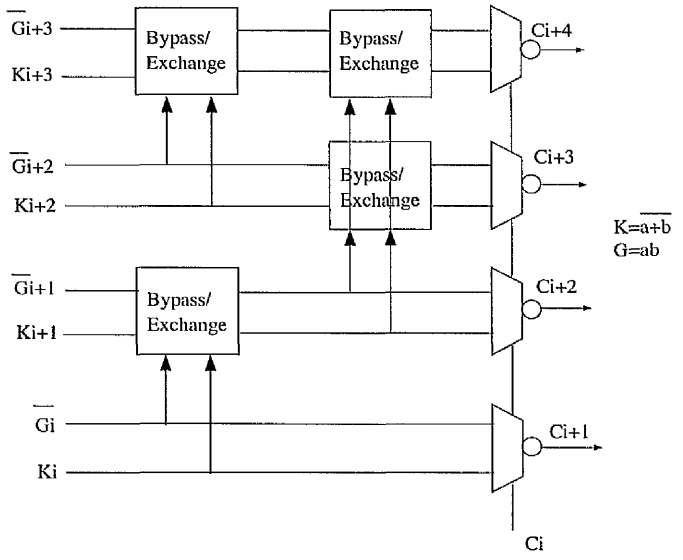


Figure 4.6: The structure of 4-bit carry-look-ahead tree block

t_1, etc. refer to the time at which a particular point in the circuit becomes valid. It can be seen that the carries into the most significant four bits are valid at t_4 so the correct sum is available at t_5. The time through the multiplexer is very small because the pass gate implementation gives good overall performance.

4.2.3 Adder Implementation

The adder of the implemented FU is shown in Figure:4.8. Input A can be selected from PS or operand A and input B from PC or operand B, where PS and PC are the multiplier outputs. The input SHACC is used when the multiply-accumulate operation is performed. The SHACC input is generated from the second read port of the accumulator registers which is then passed through the shifter. There is therefore the option to modify the value of the accumulator input. Finally, the constant input is used when a rounding constant is required by an instruction and this input also carries the eight carry bits from the multiplication. These carry bits are formed to be at bits 0, 2, 4, 6, 8, 10, 12 and 14 in order to match the same position as the LSB of each partial product from the multiplier, these carries can then be dealt with by existing logic can be connect with other parts of the

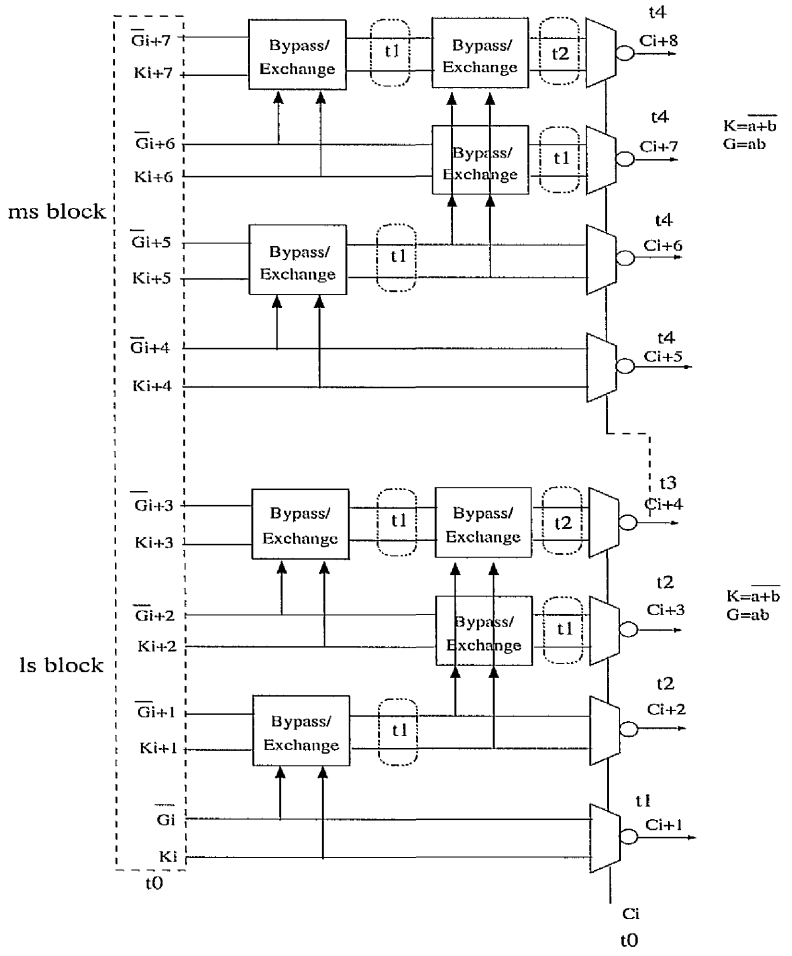


Figure 4.7: An example of the delay characteristic in an 8-bit CLA multiplexer tree

multiplier saving both time and power. As usual, inputs are inverted when subtraction is needed.

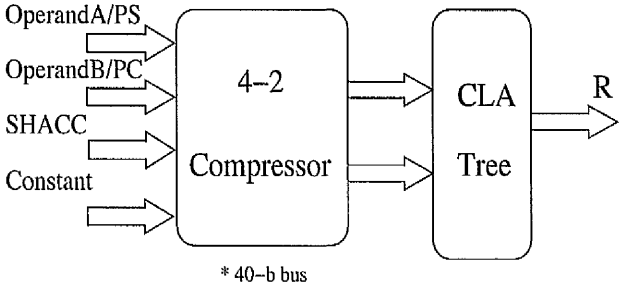


Figure 4.8: The adder structure of this FU

The addition here has two stages; the first stage is used to compress from four inputs to two inputs and these are then summed up in the second stage. Details of the 4-2 compressor will be given in the multiplier section. The carry-look-ahead tree discussed in section 4.2.2 is used in the second stage of the adder in order to reduce the delay time of the carry propagate. Additionally, subtraction can be simply performed in the adder by using the one's complement input and setting the carry input to be 1.

The adder is implemented by using only pass transmission gates. This makes the adder achieve both high performance and low power design. The results from post full-custom layout on 0.18um process technology operating at 1.8V show that the 4 input 40 bit adder consumes an average power of only 7.56mW at a throughput equivalent to 250 MOPs. The energy efficiency of this adder appears to be better than those recently reported[81].

4.3 Multiplication

Multiplication is a basic arithmetic operation that is fundamental to digital signal processing. Both multiplication and multiply accumulate operations are performed by the multiplier. Furthermore, the multiplier is involved in most time slots during the execution of DSP algorithms such as the FIR filter, Discrete Fourier Transform (DFT), DCT or Linear Predictive Coding (LPC). The multiplier in a general DSP processor is required to operate with high performance whilst keeping the power dissipation low. The architecture and circuit implementation of the multiplier therefore needs to be considered carefully.

4.3.1 Background

The basic multiplication scheme is a multi-operand addition process.

Assume

a is the multiplicand = ak-1, ak-2,..., a1, a0

x is the multiplier = Xk-1, Xk-2,..., X1, X0

P is the product (a x X) = P2k-1, P2k-2,..., P1, P0

For example, the multiplication of two k-bit operands can be completed in k cycles of shifting and adding. In multiplication with left shifts, the partial product terms $X_j a$ (each term is either 0 or a), are added up from least to most significant:

$$P^{(j+1)} = 2P^j + X_{k-j-1}a \quad \text{with } P^{(0)} = 0 \text{ and } P^{(k)} = P$$

The basic multiplication can also be implemented with a right shift algorithm. In this multiplier, partial product term $X_j a$ are accumulated from most to least significant:

$$P^{(j+1)} = (P^j + X_j a 2^k)2^{-1} \quad \text{with } P^{(0)} = 0 \text{ and } P^{(k)} = P$$

These two basic multiplications of course yield the same result. There is no difference between the two styles when parallel multiplication is adopted in terms of the logic used, due to all partial products being formed in parallel and needing to use 2k-1 bit wide additions as shown in Figure:4.9. However, left-to-right has a speed advantage due to using a carry free addition. In[84], a left-to-right (LR) carry free array multiplier was proposed where the final addition step to produce the most significant bits of the product was avoided by using concurrent absorption of any carries in parallel with the linear reduction in[85]. This structure is called Leapfrog and takes advantage of the delay imbalances in adders. Its disadvantage is a hardware overhead. The decision of which multiply algorithm to adopt therefore depends on the aims of the applications.

For mobile phones and portable applications, these complex systems are expected to support the requirements of modern features such as multimedia, high quality games and so on. The parallel or tree multiplier with a high performance carry save adder (CSA) tree as shown in Figure:4.10 is justified for these applications [86] and [87]. A dot in Figure:4.10 represents one bit and shows the carry save adder functions in dot notation. The three dotted lines above the dashed line are inputs and these are summed to produce two outputs: the partial sum (PS) and partial carry (PC). The circuit compresses three inputs to two outputs so it is called a 3-2 compressor or 3-2 counter. A CSA adder tree using 3-2 compressors can reduce n partial products to two numbers within 4 levels.

For a k x k multiplier, k partial products are produced and a k-input CSA tree is used to reduce them to two operands for the final addition. However, the number of partial

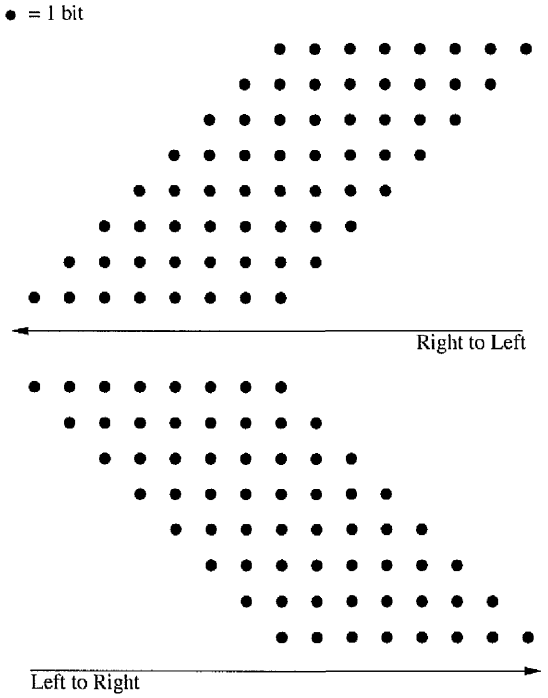


Figure 4.9: An example of the LR and RL in 8-bit parallel multiplier

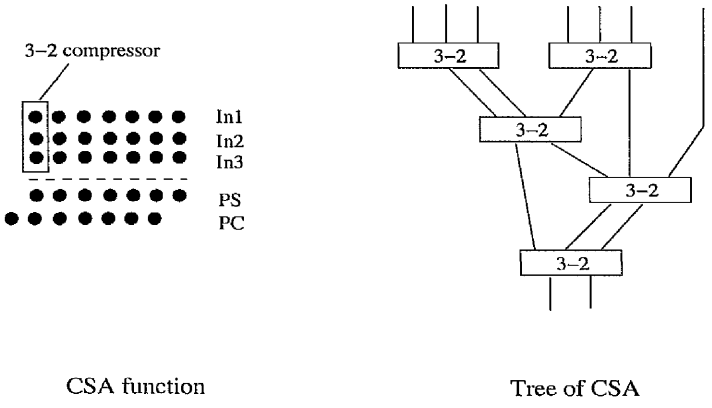


Figure 4.10: Carry save adder (CSA) and tree of CSA adder reducing 7 numbers to 2

products and the logic depth of the CSA can be reduced using modified Booth's algorithm.

The original Booth's algorithm considers a group of ones to be signed. Booth observed that whenever there are a large number of consecutive 1s in the multiplier, multiplication can be speeded up by replacing the corresponding sequence of additions with a subtraction at the least significant end and an addition in the position immediately to the left of its most significant end. The modified Booth's algorithm as shown in Table4.2[88] considers multiplier bits in pairs and treats the pair as a signed two's complement number; this algorithm requires no additional adjustment for negative multipliers. When modified Booth's multiplication is performed, only the multiples ±a and ±2a of the multiplicand (a) will be required, all of which are easily obtained by shifting and/or complementing.

Table 4.2: Modified Booth's recording

Xi+1	Xi	Xi-1	Action	Explanation
0	0	0	0	No string of 1s in sight
0	0	1	1a	End of a string of 1s in X
0	1	0	1a	Isolated 1 in X
0	1	1	2a	End of string of 1s in X
1	0	0	-2a	Beginning of a string of 1s in X
1	0	1	-1a	End one string, begin new string
1	1	0	-1a	Beginning of a string of 1s in X
1	1	1	0	Continuation of string of 1s in X

The higher radix leads to fewer partial products (PP) but requires more complex hardware and a longer time for decoding. The radix-4 Booth's recoding (i.e. the modified Booth's algorithm) has been chosen to generate the PP in this design because for larger radices the number of multiples required goes up exponentially. Furthermore, odd multiples will take time to generate and be power hungry; for example the radix-16 Booth's recording (4-b per cycle) requires the generation of the multiples {±0, ±a, ±2a, ±3a, ±4a, ±5a, ±6a, ±7a, ±8a}. It is hard to implement ±3a, ±5a and ±7a.

Radix-4 modified Booth's algorithm can be viewed as a digit-set conversion with the 2 bits denoting the numbers 0 to 3 converted to the set of {-2,2}. Effectively the two bits are treated as a signed two's complement number. Furthermore, forming the partial product is easy as it is just a set of AND gates. With a radix-4 modified Booth's algorithm, the

number of PPs can be reduced by half, whereas the delay of the PP generation is not as long as if radix-8 were used.

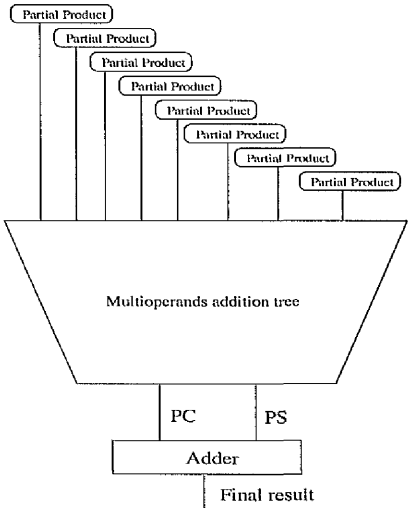


Figure 4.11: A general structure of 8-bit parallel or tree multiplier

Figure:4.11 shows the general structure of a full-tree multiplier. The parallel or tree multiplier consists of two main components: partial product generation (PPG) and the addition of those partial products (PPs). If the multiplier is aimed at a low energy applications, the parallel multiplier should be used. The serial-parallel multiplier does use less logic but it is worse for power because the resister uses a relatively high amount of power and performance comparison is illustrated in Figure4.12 where two basic multiplier structures of a serial-parallel and tree multiplier are compared in terms of execution time and energy for the partial product addition (since the logic to form the PPs in both is the same).

The power P depends on the logic active at anytime. Thus the energy E, which is the product of power and time, is the sum of the power from the active logic times the time t that it active. The time t in turn depends on the logic delay and cycle time. The estimated energy of the serial-parallel multiplier is relative high due to the high power dissipation in the pipeline registers. In addition, the serial-parallel multiplier would require 4 cycles to complete a multiplication which makes the energy per operation high. Thus the tree

multiplier has a better potential to be an energy efficient multiplier than the pipelined multiplier and has thus been adopted.

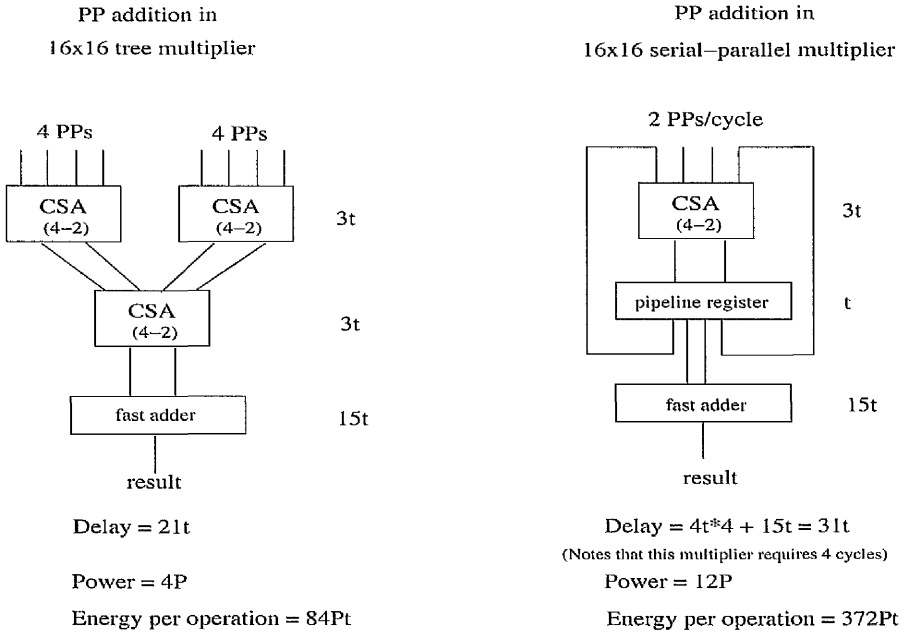


Figure 4.12: Energy estimation of serial-parallel and tree multipliers

Partial Product Generation

In the basic tree multiplier structure, there are many places where fast or low power design techniques are able to be applied. Thus these are many trade-offs between performance and power. Radix-4 modified Booth's algorithm, used in this design, appears a good compromise and the partial products are formed using a set of AND gates.

Partial Product Addition

For partial product addition, several methods such as the ripple carry adder, the CSA, the CSA tree, the Wallace tree[89] and Dadda's strategy[90] are all candidates for the addition scheme in the parallel or tree multiplier. Both Wallace's and Dadda's strategies are based on compression techniques which were first introduced by Weinberger[91]. The differences between Wallace and Dadda are that the Wallace tree tries to combine the

partial product bits at the earliest opportunity, whilst Dadda's scheme combines them as late as possible and keeps the critical path (level) of the tree minimal. So Dadda's structure is simpler but has a wider CPA at the end compared to the Wallace tree. However, combining the partial product bits as soon as possible makes the Wallace tree scheme faster than Dadda.

The 4-2 compressor is popular in many digital multiplication and multi-operand addition schemes because of its performance. However, the compressor differs from Dadda's counter in that it is not necessary to have the pattern of M outputs drawn from 2^M inputs. An N:M compressor in essence is a variation of the Dadda counter that employs a separate path between compressor units in order to generate M final outputs using $N>2^M$ input bits. The 4-2 compressor is based on a 5:3 counter structure because it is impossible to use 2 output bits to represent the 5 binary input bits resulting from the following equations:

$$\begin{aligned} PS &= P1 \oplus P2 \oplus P3 \oplus P4 \oplus Cin \\ PC &= Cin(P1 \oplus P2 \oplus P3 \oplus P4) + P3\overline{(P1 \oplus P2 \oplus P3 \oplus P4)} \\ Cout &= P1(P2 \oplus P4) + P2\overline{(P2 \oplus P4)} \end{aligned} \quad \text{... Eq.4.6}$$

The truth table corresponding to the above equations is shown in Table4.3 where the three output, PS, PC and Cout, have the bit weight of 2^0 2^1 2^1, respectively. The reason for using two bits weight 2^1 for PC and Cout so that the outputs from compressor i need only feed to compressor i and i+1 in the next stage. This leads to a convenient layout implementation.

Table 4.3: Truth table of 4-2 compressor

P1	P2	P3	P4	Cin	Cout	PC	PS
0	0	0	0	1	0	0	1
0	0	0	1	0	0	0	1
0	0	0	1	1	0	1	0
0	0	1	0	0	0	0	1
0	0	1	0	1	0	1	0
0	0	1	1	0	0	1	0
0	0	1	1	1	0	1	1
0	1	0	0	0	0	0	1

Table 4.3: Truth table of 4-2 compressor

P1	P2	P3	P4	Cin	Cout	PC	PS
0	1	0	0	1	0	1	0
0	1	0	1	0	1	0	0
0	1	0	1	1	1	0	1
0	1	1	0	0	1	0	0
0	1	1	0	1	0	1	1
0	1	1	1	0	1	0	1
0	1	1	1	1	1	1	0
1	0	0	0	0	0	0	1
1	0	0	0	1	0	1	0
1	0	0	1	0	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	0	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	0	1	0	1
1	0	1	1	1	1	1	0
1	1	0	0	0	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	0	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	0	1	0	1
1	1	1	0	1	1	1	0
1	1	1	1	0	1	1	0
1	1	1	1	1	1	1	1

4.3.2 Multiplier Implementation

The 16x16 fixed-point multiplier has been implemented using two's complement numbers to produce a 40 bit result for a 40 bit DSP datapath. To explain the multiply scheme, the structure of the row and column in an Excel worksheet has been used. Each cell in Figure:4.15 represents a single bit which can be a zero or one. A horizontal row represents a partial product, whilst the latency of a multiplication is related to the height of the PP section (the maximum number of cells in any vertical column).

As previously stated, the multiplier employs a modified Booth's algorithm (radix-4) to produce the 8 PPs required in parallel. Because of using modified Booth's algorithm, the numbers of PP is reduced by half compared to a conventional add and shift algorithm.

This leads to less hardware and can be faster because the depth of cells is reduced. In modified Booth's logic, six types of control signal are generated denoting the multiple of the multiplicand required to be added. These are plus one times the multiplicand (+a), minus one times the multiplicand (-a), plus two times the multiplicand (+2a), minus two times the multiplicand (-2a), zero(Z) and a carry (C) for the adder when a subtraction is performed in the current cycle rather than addition. These control signals are generated for each pair of multiplier bits and are used to control the partial product generator (PPG) to produce the 8 PPs, as shown in Table4.4.

Table 4.4: Partial Product Generator

Multiplier	+a	-a	+2a	-2a	Z	C	Selection
000	0	0	0	0	1	0	0
001	1	0	0	0	0	0	1*Multiplicand
010	1	0	0	0	0	0	1*Multiplicand
011	0	0	1	0	0	0	2*Multiplicand
100	0	0	0	1	0	1	-2*Multiplicand
101	0	1	0	0	0	1	-1*Multiplicand
110	0	1	0	0	0	1	-1*Multiplicand
111	0	0	0	0	1	0	0

4.3.3 Input Swapping

Power dissipation of a parallel multiplier mainly arises from the switching activities of its functional blocks. One of the two inputs data is sent to the Booth's encoding block where the multiplier is decoded to generate the correct partial products in the PPG. Therefore, in the multiplication process, the control bits for the input with the smaller effective dynamic range should be used for the Booth's encoding to increase the chance of neighbouring partial products being '0'. The results from a 0.18μm@1.8V post-layout multiply-accumulator simulation with a throughput rate at 200 MHz are shown in Table4.5 and illustrate the different power consumption when the multiplier has a different number of ones with the multiplicand remaining constant.

Table 4.5: Power dissipation the different dynamic range inputs (multiplicand is constant)

Number of non-zero multiplier	Power (mW)
010	1.98
01010	2.16
0101010	3.42
010101010	5.04
01010101010	5.04
0101010101010	7.02
010101010101010	7.20
01010101010101010	8.28

From the experimental results in Table4.5, swapping two inputs may reduce the energy of the multiplier. However, this needs further investigation since swapping inputs requires additional hardware and hence extra power. The block determining if inputs should be swapped is as shown in Figure:4.13. It detects the effective dynamic ranges of the input data. This block will make a decision whether the two input data paths should be exchanged or remain unchanged.

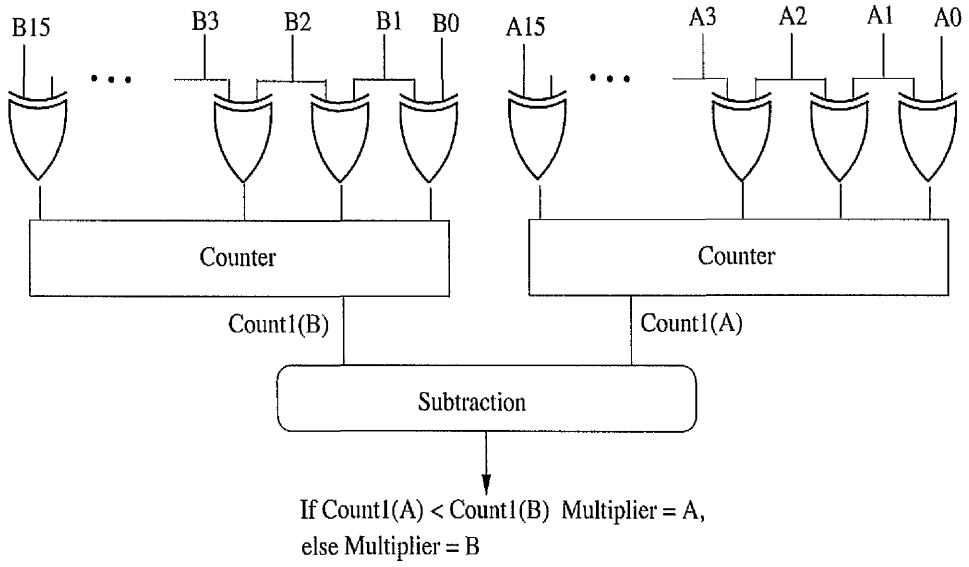


Figure 4.13: Determination block

The determining block mainly contains XOR gates to scan and count the number of non-successive '0's or '1's at the A and B inputs and generates a signal to select the operand with the smaller number of non-successive '0' or '1' inputs as the multiplier. The other input then becomes the multiplicand. However, there is also a latency of this logic that needs to be considered carefully. The simulation result of a 16-bit determining block shows that it takes about 1.4 ns to generate the select signal for switching the inputs of the multiplier, whilst the average power it dissipates is about 9mW at a speed of 666 MHz (where data is fetched every 1.5 ns). Therefore, swapping inputs should be used when the delay and power dissipation of this block are lower than the power that can be saved when it is applied in the multiplier. Simulation of the multiplier power shows that the determination block to swap the inputs does not save energy and therefore input swapping has not been adopted for the design.

The PPG here is implemented by using four pass transmission gates for the +a, -a, +2a, -2a multiplicand selection, and one N-MOS transistor for Z as shown in Figure:4.14; this makes the PPG small and fast. Taking into account the need for true and inverse control signals, 9 control signals as shown in Figure:4.14 are needed to produce each PP. This is another trade-off between a large number of control signals and the small number of transistors in the PPG. This multiplier chooses to have a large number of control signal because the large number of wires required between Booth's logic and the PPG has been wired by hand. Therefore, the wiring load capacitance and the area is relatively small compared to the overall design.

4.3.4 Sign Extension

The eight PPs shown in Figure:4.15 are the output of the PPG which produces eight 16 bit PPs (which is the same as the multiplicand length). However, the output is expected by a general DSP to be in 40-bit format. So each PP has to be sign-extended to 40 bits prior to their addition. The energy saving approach in the multiplier is to reduce the logic for the sign-extended bits by using a pre-calculated sign extension, as shown in Figure:4.16, over a group of 4 PPs. This pre-calculated number assumes that all partial products negative and thus have a sign bit of one.

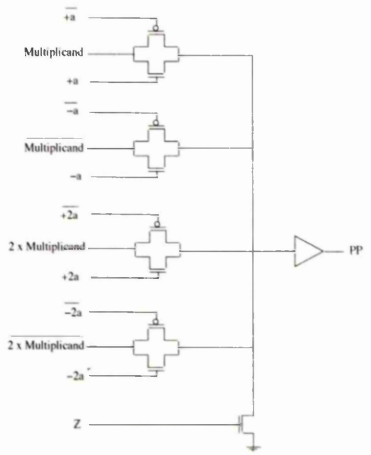


Figure 4.14: Partial product generator circuit

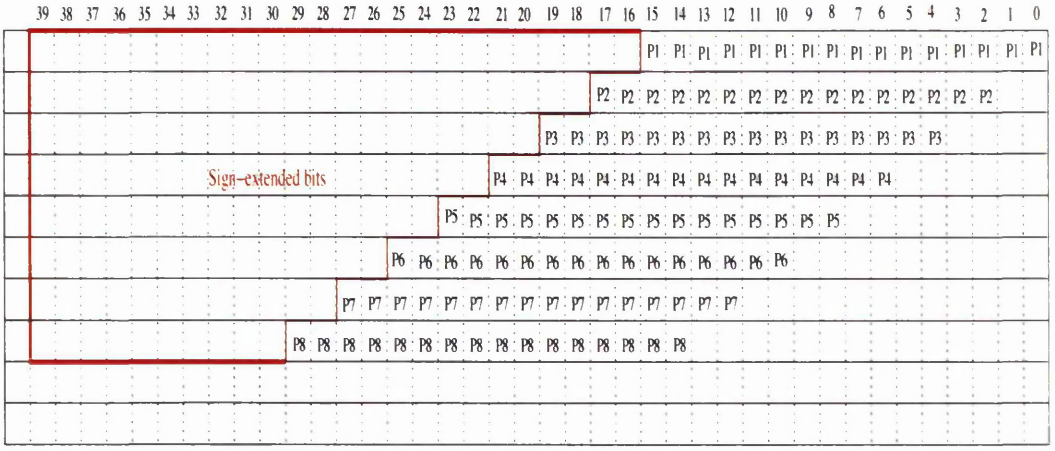


Figure 4.15: Eight PPs presented in excel worksheet

In Figure:4.16, the assumed sign extension for PP1 to PP3 are shown and when added yield the sign extension constant shown along the top line. If the real PP has a most significant bit (msb) equal to *one*, no adjustment is necessary. In contrast, when the real MSB is *zero,* adjustment is required equivalent to all the 'one' bits in the extension being flipped back to zero. This is achieved by adding a $\overline{P}x$ (x = 1 to 3) in the bit position shown; it should be noted that the '0's in the sign extension constant makes it less likely that the carry chain will extend across all bits in the extension constant. It is believed this technique is used commercially although details are not reported because of commercial

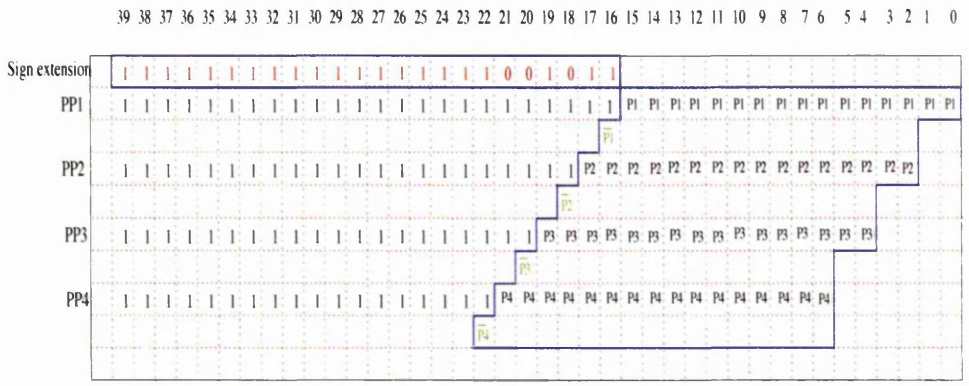


Figure 4.16: Pre-calculated sign extension

sensitivity. With this technique, the constant number required can be computed before the multiplication takes place in the hardware. This allows the addition hardware of the sign-bit part to be reduced.

In the addition of the PPs, most designs use a carry save adder (CSA) to avoid long latency with some designs adding pipelining as well to increase the addition speed. Yet others use an iterative pipeline addition including a shift register to make the circuits small but this gives a long multiply latency. However, they are not suitable for the energy efficient FU in this parallel DSP because whilst the iterative multiply has less power dissipation; it spends a longer time to complete the multiplication due to the number of partial products to be added. Furthermore, the iterative multiplier requires more complex control circuits and shift registers. Therefore, it is not as good in terms of energy efficiency as the parallel multiplier implemented here. The parallel multiplier has been designed as shown in Figure:4.17

The Wallace tree [89] has been applied in the addition of multiplier instead of the well-known Dadda's strategy[90] as the partial products combining takes place as late as possible. Therefore, Wallace's method is faster than Dadda's. The addition of the PPs is usually done in 3 levels of compressors as shown in Figure:4.18(a). At the top level, two groups of compressors compress a group of 4 PPs to two outputs. In the second stage, two groups sum the four outputs from the top level in 3-2 compressors, with 4-2 compressors

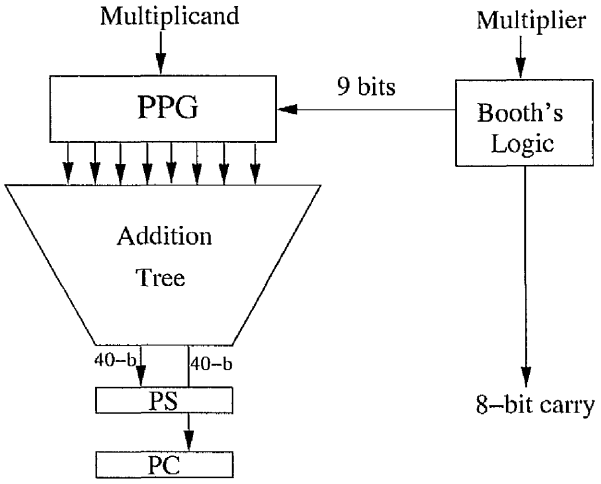


Figure 4.17: Parallel multiplier architecture

used at the final level[91]. If the multiplier bits require a -a or -2a, then the multiplicand bits are inverted and +1 is input to the addition tree to form the two's complement input to the second stage. In the CADRE-s design, the second level compressors are omitted as the carry bits are forwarded directly to the 4-input adder, as shown in Figure:4.18(b).

A significant reduction in logic arises from losing the second level compressors and this both improves performance and saves considerable circuitry as can be seen in Figure:4.18(b). As a result of using this tree structure topology, the number of stages traversed by each input is approximately the same for all inputs. This leads to a balanced delay tree and results in less switching activity due to input skew.

The post full-custom layout simulation on a 0.18um process operating from 1.8V (on Nanosim) shows that the multiplier can produce a PS and PC every 1.5 ns in the worst case of random inputs and consumes an average power of only 10.8 mW at this speed. A comparison against other designs is difficult because of the differences in term of process technology and bit width. However, selecting some other low energy designs that have been reported such as the 16x16 multiplier (32-b output) on a 0.09 micron process technology running from 1.2V[92], the 4x4 wave pipeline on a 0.18 micron process running from 1.8V[93], a 16x16 pass transistor multiplier (32-bit output) on a 0.8 micron running from 3.3V[94] and a 16x16 multiplier (40-bit output) on a 0.13 micron process

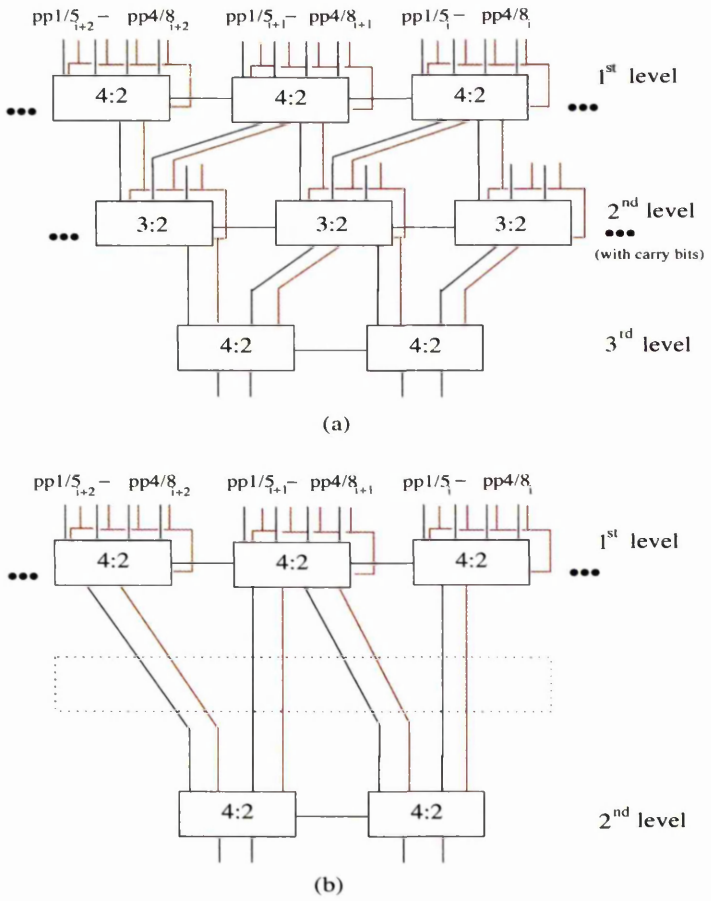


Figure 4.18: The conventional Wallce tree and this addition tree

running from 1.2V[95]. To get some idea of the relative merits of CADRE-s multiplier, all results have been scaled to a 0.18μm geometry running from 1.8V. Whilst scaling does not take into account all effects, the results do give an indication of the energy efficiency of the different multipliers. Table4.6 shows the lowest energy consumption (PDP) and energy delay product (EDP) of the multiplier described here (including the time to add and the adder energy) compared to others reported; it demonstrates a good compromise between performance and power for the 0.18μm. process used. The power delay product (PDP) can be useful for comparisons in which absolute energy values are not known[96] whilst the energy delay product (EDP) is normally used when the circuit-speed is important for an energy efficiency comparison.

Table 4.6: Comparison of energy delay product of multiplier

Types	Scaled Power to 0.18um@1.8V (mW)	Speed (MHz)	Energy (PDP) (pJ)	Energy-Delay (EDP) (pJ x ns)
CADRE-s	3.64	200	18.2	91.0
[92]	49.5	500	99.0	198.0
[93]	74.48	167	446.8	2,681.3
[94]	11.34	44	257.0	5,855.5
[95]	201.9	435	464.4	1,068.0

4.4 Multiply Accumulate (MAC) Operation

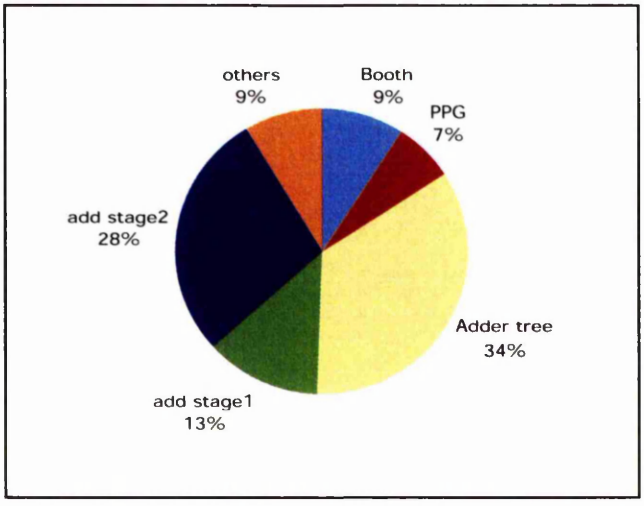


Figure 4.19: Average power breakdown of the multiply-accumulator (MAC)

The MAC instruction is widely used in DSP applications as shown in Figure:4.1 and is performed by the 16x16 bit multiplier and 4-input 40-bit adder previously described. The power dissipation ratio in the MAC unit is analysed in this section. Random numbers have been used in the simulation of the post-layout netlist at a speed of 200 MHz where

Nanosim Simulator has been used for getting the power and timing information of the designs here and others in the thesis. The resistors are not included in the post-layout netlist according to the limit of tools and simulation time. Therefore, it is impossible to put the resistor information into the post-layout netlist. However, from the author's experience simulating a small circuit by using SPICE, this resistors can affect the overall power consumption by 10%.

A breakdown of the power dissipation for the blocks within the multiplier and adder is depicted in Figure:4.19. It can be seen that the dominant source of power consumption is the addition tree where eight partial products are summed to produce the partial carry and the partial sum. The addition stage2 (carry-look-ahead tree) is the next greatest source of power dissipation, whilst the Booth's logic circuit dissipates only 9% which is the same as the other logic such as the multiplexer and bus selection decoder circuits; the smallest power is dissipated in the partial product generator (PPG). The pie chart also shows that, if the ADD instruction is performed, the power dissipation is about 41% that of the MAC or MPY instruction. Therefore, numbers multiplied by a power of 2 should be transformed to use the ADD instruction instead.

In this chapter, the designs for a multiplier and adder in CADRE-s for good energy efficiency have been described. As a result, CADRE-s can operate with energy efficiency when running DSP algorithms which contain a large number of MPY, MAC, ADD and SUB instructions. However, other functions such as Hamming distance and normalization are also necessary. These functions and other components such as shifter, accumulator, timing and control circuits are described in the next chapter.

Chapter 5: Energy Efficient Functional Unit - Hamming Distance, Normalization, Timing and Control

5.1 Hamming Distance and Normalization

Although most operations feature addition or multiplication, other operations such as saturation, Hamming distance (HD) and normalization (NORM) are also required in the DSP algorithms.

The Hamming distance function is a count of the number of bits which differ in two data words. It is an essential operation in image processing applications, pattern recognition, intelligent processing systems and nearest matching applications[97]. It is therefore a necessary instruction in a general DSP. The first step in forming the Hamming distance is to perform a comparison between the two words and some recent publications describe this comparison[98]. A mixed-signal design, using capacitive threshold logic gates to implement the comparator in the HD is described by Fujino and Moshnyaga[99]. Using these gates, the comparator consumes less area but has high power consumption due to having a large capacitance per bit since these gates are charged biased circuits. It dissipates even more power by charging and discharging the capacitances every cycle and this also increases the delay overhead. Asada et al. presented a HD comparison circuit not using a conventional two's complement coding which is targeted at an associative memory[100]. As the largest number is output rather than the number of ones between the two numbers, this approach is unsuited to a general purpose DSP processor.

The Hamming distance unit in this FU uses a parallel bit comparator and then sums the output in parallel count circuits as shown in Figure:5.1. The logic is implemented using pass transistor circuits; this makes the design fast and gives it low power consumption.

An example of finding the 40-bit Hamming distance is shown in Figure:5.2; the input patterns differ in seven bit positions (leading to seven being output).

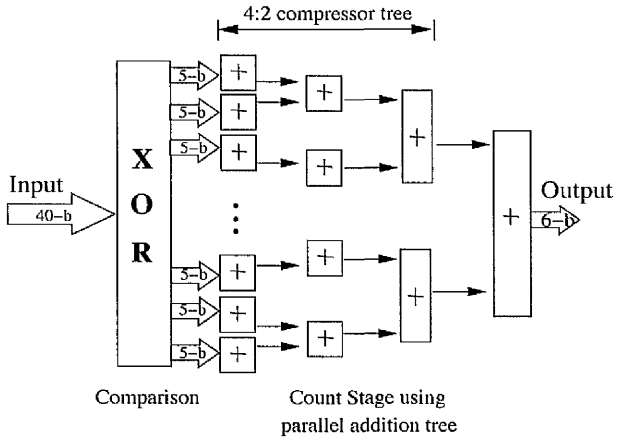


Figure 5.1: A structure of the parallel comparator and counter

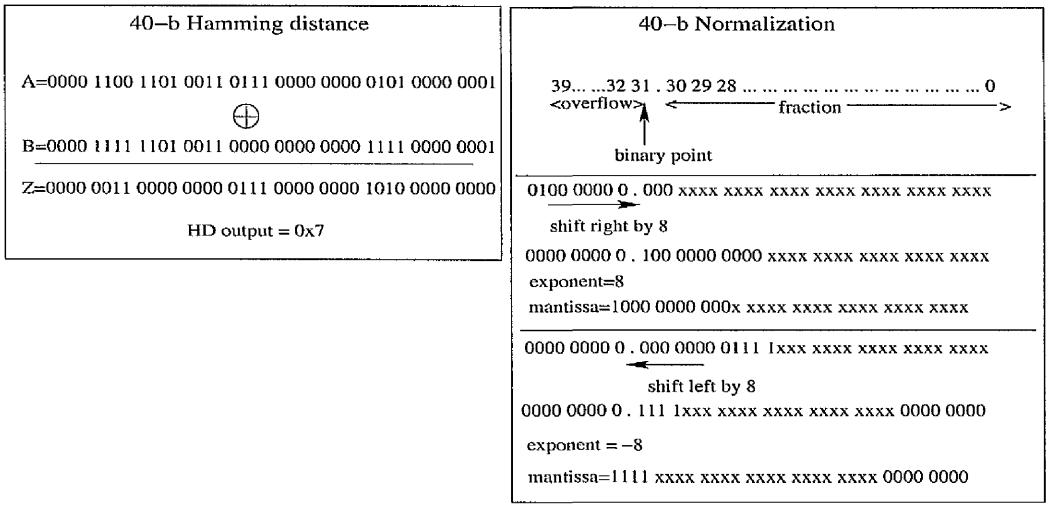


Figure 5.2: Examples of HD and NORM using 16-b for HD and 40-b for NORM

Normalization (NORM) is another useful instruction for DSP algorithms although it is not frequently used. The NORM operation is used for a fixed to floating point conversion (operation) that produces a normalised mantissa. In this instance the input data is represented in 40 bits: one sign bit and 31 bits for the fractional part with the

most significant 8-bits supposedly being a sign extension. However, following a set of calculations it may be that the result has overflowed into this extension, or else underflowed resulting in (extra) sign extension bits. The number is normalised by shifting so that the binary point is again between the most-significant different bits (i.e. either 1.0 or 0.1) and normalisation calculates the shift distance, which is also the correction to the exponent. Examples of left and right shift normalising are shown in Figure:5.2; if the number is greater than 1 it is shifted right n places with n places being lost from the least significant n bits of the fractional part. For purely fractional numbers, a left shift of n places is performed with zero inserted in the least significant n places.

NORM is defined as a special instruction in Motorola's DSP56000 family[101]. Its syntax is NORM Rn,ACC where it performs only one normalization iteration on the specified destination operand ACC, updating the specified register Rn based upon the results of that iteration and storing the result in the destination accumulator. For example, if the accumulator(ACC) is 0x0000000001, then ACC would be arithmetic shifted left and Rn will be updated to be Rn – 1. This can be repeated until the most significant bit '1' is in the position of the binary point. In contrast, if the ACC bits beyond the decimal point are used then the ACC is arithmetic shifted right and Rn is updated to be Rn + 1 iteratively until the most significant bit '1' is in the position of binary point. It is obviously that extra operations are needed to find the position of the most significant bit '1' and the normalization time depends on the number of iterations.

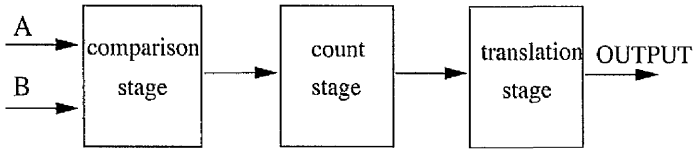


Figure 5.3: A basic structure of the HD and NORM circuits

To perform the NORM operation, the bit position of the most significant one in the number needs to be located as this subtracted from 32 gives the shift distance. The strategy for identifying the bit position of the most significant one is to populate all lower

order bits with '1' and to then count the number of bits set. The counting is exactly the same as that required in the Hamming operation and the location of the most significant '1' is best achieved by comparing the number with a 1-bit shifted version of itself. Again, this input comparison is required by the Hamming code which looks for the difference between 2 numbers. Thus both the HD and NORM operations require the same basic components of a comparator and bit count stage. However, an output translation stage is required to produce the correct output if both operations share the same circuit. A top-level view of their operation is shown in Figure:5.3. Special instructions, HD and NORM, are included in a general DSP so that these operations are performed efficiently when needed. However, their implementation is non-trivial. In addition, their latency can be longer than frequently occurring instructions such as multiply, add or multiply-accumulate. The idea of the logic reduction and improved performance presented here is therefore extremely useful for these instructions in reducing the overall ALU cycle time. The idea of combining these two functions into a common logic block is novel and in addition this method of performing the NORM function has not been presented before.

5.1.1 Hamming Distance and Normalization Specification

Consideration of the HD and NORM functions shows that they perform similar operations. Both the HD and NORM require a comparison at the first stage and need logic to count the number of ones output from the comparison stage. In the HD case this operation is a direct addition of the bits; in the NORM case the single input number needs to have adjacent bits compared to find the position of the most significant 10 or 01 combinations.

The two functions perform different calculations; the HD operation is looking for the total number of ones to define the difference between the two binary numbers. Its output indicates how similar the two numbers are. With a 32-bit input the output of the HD operation could be represented by 5 bits; however here a 6-bit code is used for compatibility with the (signed) NORM result. The NORM operation seeks the position of the most significant place where the bits are different. The NORM operation finds the number of places that need to be shifted to produce a normalised mantissa. The shift range is +8 to -31 which yields a 6-bit (two's complement) result. An input which is already normalised - or an input of 0 - will give a zero result.

There are many different ways to implement the parallel counter such as using a 2-input adder, a 3-2, a 4-2 compressor and so on. However, this design is aimed at use in a full-custom functional unit where the fewer the number of functional cells used, the more implementation time can be saved. Only the 4-2 compressor is therefore employed in this parallel counter as this is used elsewhere. In addition, the logic design is based on the use of XOR and multiplexer functions. This enables circuits to be implemented using pass transistors making the circuits far smaller than conventional static CMOS circuits.

5.1.2 Hamming Distance and Normalization Design

Comparison Stage

In the HD and NORM logic, the first stage (the comparison stage) of these two operations is the same, only the inputs are different. The HD has two inputs and finds the difference in each bit position using XOR gates. The NORM has only one input(A), which is XORed with itself shifted right one place. This produces a '1' wherever these differ with the most significant '1' in the original pattern indicated by the most significant '1' in the comparison output. A multiplexer can also supply the XOR function. Figure:5.4 shows the logic of this first stage; a mode signal (NORM) selects the appropriate input to the XOR stage i.e. B or a shifted copy of A.

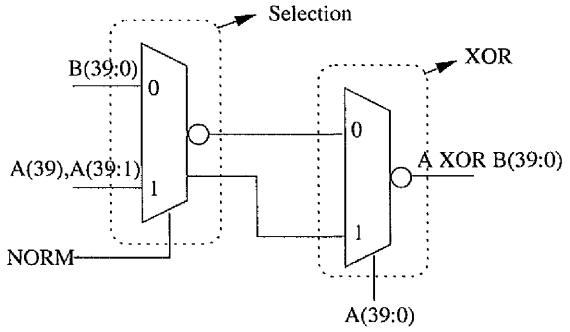


Figure 5.4: The first stage of HD and NORM operations.

Count Stage

a) Hamming Distance (HD)

For the HD operation, the second stage sums the ones from the comparison stage. To increase the performance of this design, a parallel structure has been employed. A multistage process is used which takes the comparator output and transforms it over four addition steps. Carry save adders at each addition stage produce a partial sum, PS(j), and partial carry, PC(j+1). These are then summed in a full adder to give a count result.

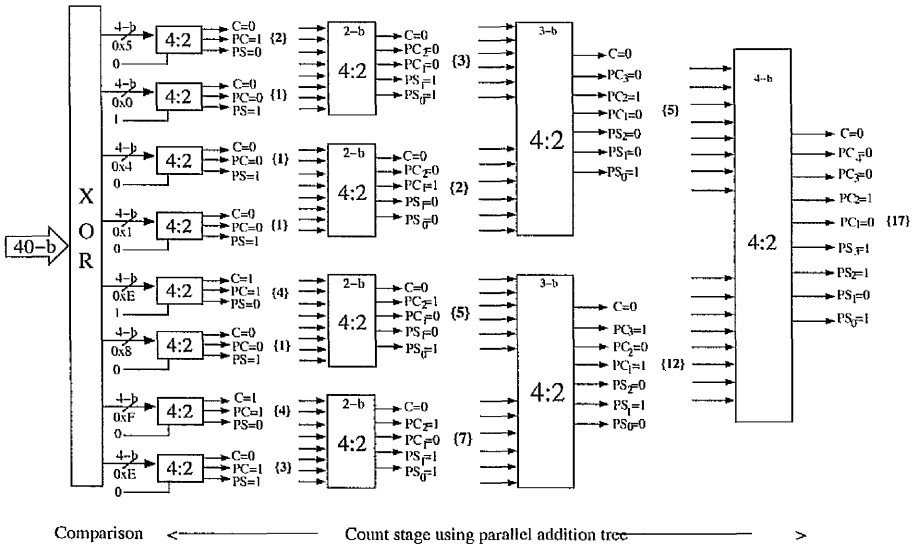


Figure 5.5: The parallel addition using 4-2 compressor

As shown in Figure:5.5, in the first addition step the 40-bit input is divided into ten parallel groups of four bits. The 4-2 compressor takes a 5-bit input (P1, P2, P3, P4 and Cin) and produces an output {Cout, PC, PS} where the first two elements each represent a count of 2^1 and the last a count of 2^0. Thus from 000 to 111 indicates that there are from 0 to 5 '1's present in the input.

b) Normalisation (NORM)

To perform a NORM operation with this hardware, the number needs to be transformed. If bit i is the most significant '1' in the pattern generated by the XOR stage then populating all bits of lesser significance with ones and then counting the number of '1's set will yield bit position i. For example, if A XOR B equals 0x50502EC3DC, then populating all lower order bits to '1' yields (0x7FFFFFFFFF) and counting the number of '1' s in the transformed pattern gives 39 for the count stage which corresponds to the most significant '1' position set in the original number. This number is not quite the same as that needed by normalization; this requires the subtraction of 31 to give the answer 8. The subtraction takes place in translation stage. Since subtraction by 32 is easier than 31, 1 is added into the last stage of the addition tree before subtracting 32 in the translation stage.

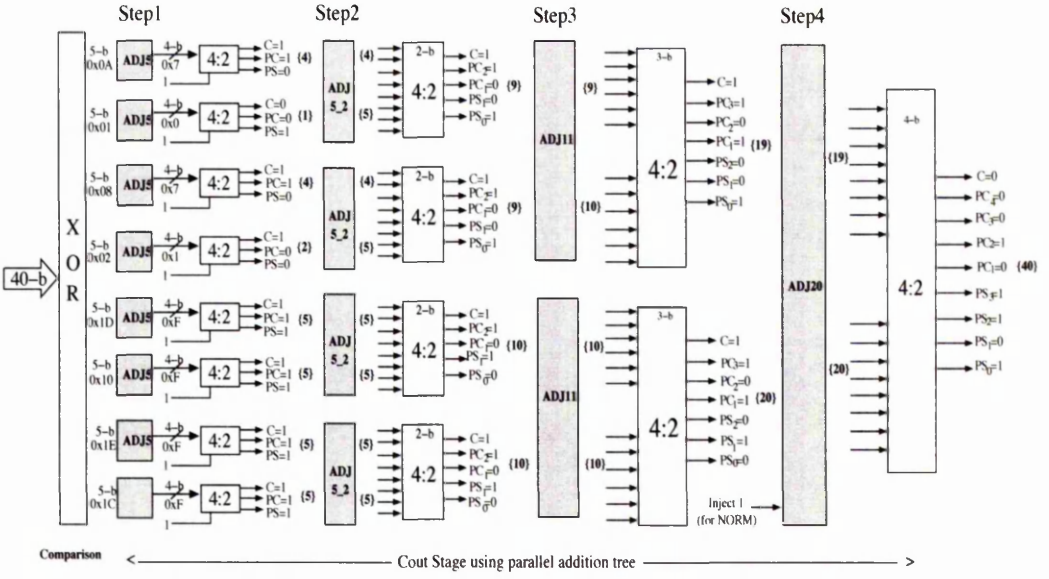


Figure 5.6: The population in each step of the NORM operation

An example of the NORM operation is given in Figure:5.6 with the input A=0x6060348297. It illustrates how each step adjusts the inputs appropriately and then performs a compression to successively encode the data. The first addition step, the ADJ5 output, populates all lower order bits below the most significant '1' in the group. So the output of ADJ5 will be either 0 corresponding to the input 00000, 1 for an input 00001,

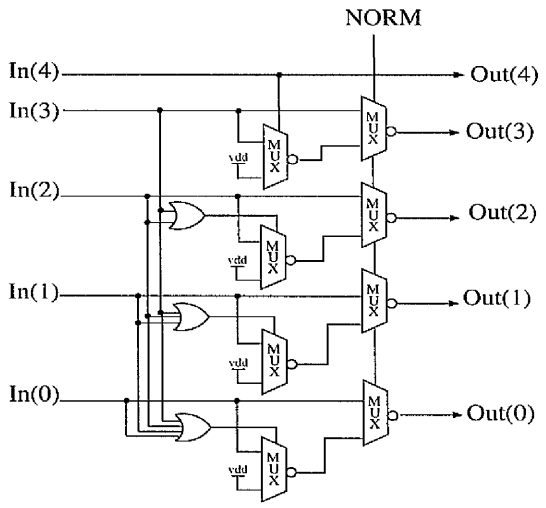


Figure 5.7: The example of the first adjusted component (ADJ5)

3 for an input 0001δ, 7 for an input 001δδ, 0xf for an input 01δδδ and 0x1f for an input 1δδδδ. The logic to perform this is shown in Figure:5.7. The output multiplexer selects between by-pass (HD) and adjusted data (NORM). The first multiplexer stage inserts '1's in lower order bits where bits of higher order are '1'.

In the second addition step, ADJ5_2 forms the inputs for the 4-2 compressor. The outputs of the step1 compressors are grouped in pairs. If the output of the higher set is non zero, then all the bits of the lower set are adjusted to be to 5 represent a maximum count. In addition step3, ADJ10 transforms the step2 compressor outputs. Again the compressor outputs are grouped into pairs starting at the most significant end and if the higher set of the group is non zero, then the lower data set is populated with the maximum count value of 10. The outputs from the step3 4-2 compressors now form two groups which have their formal adjustment performed by ADJ20. Here, the maximum counted value or populated value is 20.

Translation Stage

In the final stage, shown in Figure:5.8, the result of the count stage needs to be translated to the correct format for the HD or NORM output. In the case of the NORM operation, the output (in decimal code) of the last addition needs to be subtracted from 32 to give the

final answer of the distance between the most significant '1' and the binary point as mentioned in the previous section. This gives a 6 bit output which is then sign extended to provide a 16-bit output. The HD output by-passes the subtractor and is just sign extended to 16 bits.

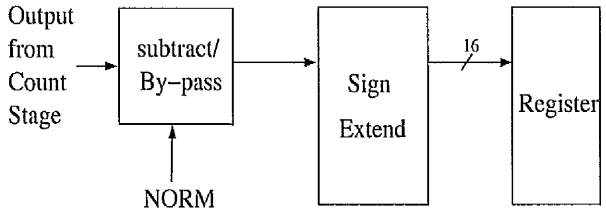


Figure 5.8: The translation stage diagram

5.1.3 HD and NORM Results

The HD and NORM designs have been implemented in two ways using DPL (Double pass transistor logic) and pass gate logic for the compressor because DPL produces the output Z and its inverse output $\overline{Z}$ which give balanced positive and negative transitions whilst PTG uses a smaller number of transistors than DPL. Balancing circuits (as in DPL) and an optimal number of transistors (as in PTG) both have an energy saving potential. The 4-2 compressor results for the DPL are called Hdv1 and NORMv1, respectively, whilst the results for HD and NORM implemented in PTG gates are called Hdv2 and NORMv2, respectively.

The circuit have been implemented on 0.18 micron technology operating from 1.8V. Post layout has been simulated using Nanosim to yield currents and times over an extensive range of data inputs. Note that, apart from the zero detection logic implemented in conventional CMOS static logic, all other logic has been implemented by CMOS pass transistor logic to make the design fast and low power. The XOR function, implemented by a PTG multiplexer, is used in the comparator section.

Figure:5.9 shows the delay characteristics of the proposed circuits. The circuit with the DPL compressor takes 1.7ns to produce the result for the HD (average case) and only 1.5ns to complete the NORM operation, whilst the circuit using pass gate compressors

also takes 1.7ns for the HD operation and 1.5ns for NORM operation. This corresponds to an average of 0.2ns per stage (7 stages in total including comparison, 5-addition stages and the translation phase) for both HD circuit versions. Figure:5.9 also shows the results of the current for the HD and NORM operation. The average power consumption for HD is about 3mW whilst the NORM operation has an average power consumption of about 1.5mW at a throughput rate of 600 MOPS for both circuit versions. Such details for other normalisation circuits are scarce. However, the power consumption running the audio processing algorithms reported in[102] is about 2mW at a throughput rate 56 MOPS. Compared with this, the HD and NORM operations in the new design perform with high speed and low power dissipation.

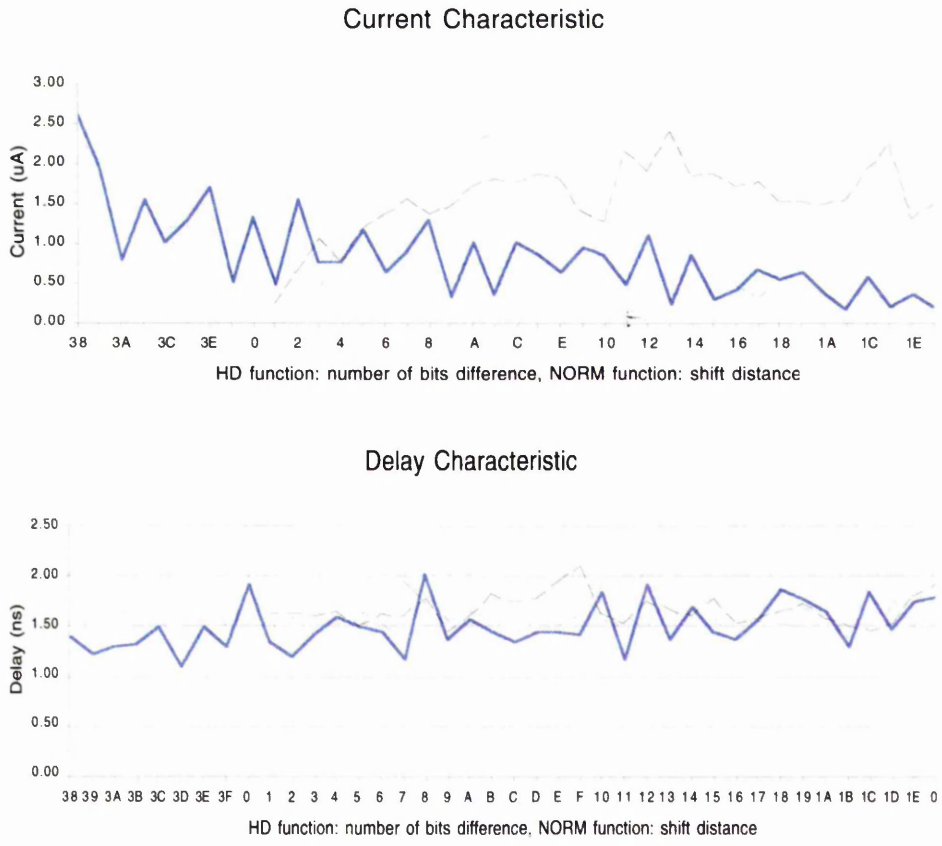


Figure 5.9: The delay and current characteristics of the HD and NORM circuits.

As shown in Figure:5.9, it is obvious that both the NORM and HD functions require approximately the same cycle time to complete the operation whilst the NORM operation dissipates less power than the HD operation. Similarly, the power consumption of the two circuit versions is approximately the same. However, the number of transistors of circuit version II is less than the circuit implemented by the 4-2 DPL compressor by a factor of two leading to a significant saving in area. Therefore, a PTG implementation was adopted.

5.2 Others Functions

Apart from the vital instructions such as multiply, add, sub, Hamming distance and normalization, the maximum and minimum instructions are also necessary. The instruction is used to select the maximum (MAX) or minimum (MIN) number kept in one of four accumulators. The maximum and minimum functions are also applied for the absolute value which is referred to as ABSMAX and ABSMIN, respectively. These instructions are performed using a subtraction. The difference between finding maximum/ minimum of the normal and absolute value is that the sign-bit is not taken into account when computing the absolute value.

Limiting in CADRe-s will occur when the 40-bit result stored in accumulator register is written back to 16-bit output RAM. The limiter logic minimizes the error due to overflow. For example, if the accumulator source was 0x0080000000 (+1 decimal) where the binary point is between bit 30 and 31, the destination 16-bit register would contain 0x8000 (-1 decimal) after the transfer, assuming signed fractional arithmetic. This is clearly in error as overflow has occurred. Thus the limiter would form and write the maximum value to the destination 16-bit register as 0x7fff = +0.999 decimal instead.

CADRE-s uses a fractional data presentation for all data arithmetic operations. The fractional 32-bit data has two parts: 16-bit left half (bits 31-16) and 16-bit right half (bits 15-0). The right half can be rounded into the left half without shifting. This is performed by adding +1 at bit position 15 when doing an arithmetic operation. This rounding makes the final result value of the 16-bit left half more precise when it is required to be written back to the 16-bit result RAM.

5.3 PTG Shifter

A shift function is an important operation in a signal processing unit. This DSP uses the pass-transmission gate to build a shifter where it can shift the content of a bus a specified number of positions left, right or no-shift as specified by the control signals. When shifting to the right position, the position vacated will be filled with values from the left or with the most significant digit if no values are available. This PTG shifter is similar to a barrel shifter except if no-shift is required, the input will be forwarded directly to the output using a bypass route.

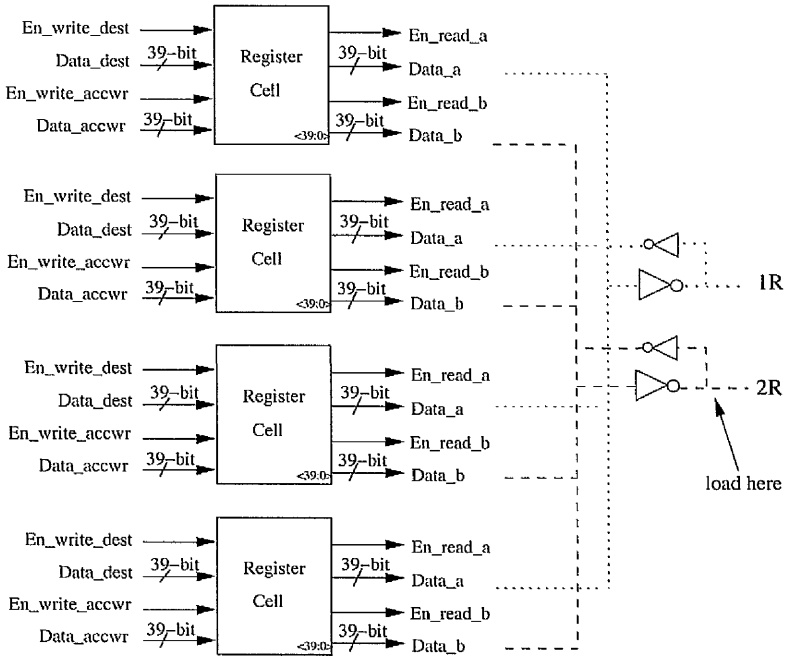


Figure 5.10: Accumulator register structure

5.4 Accumulator

This DSP has four accumulator registers which store a 40-bit result from the arithmetic/ logic operation. Two read ports are designed to support a concurrent read operation, whilst two write ports are used to support writing the result of an arithmetic/logic operation and storing due to data movement concurrently. It is left to the assembler to prevent and report any attempt to write to the same accumulator register concurrently.

A static memory circuit has been used as the register cell. However, the size of transistor becomes a major problem because of driving a big load (including a wiring load). The current consumed by different loads has been investigated. A buffer is placed at the output of the register circuit as shown in Figure:5.10 and this uses the strongest drive that could be fitted into the custom cell. The post-layout netlist of ACC with a various load capacitances has been used in SPICE simulations. Ten random numbers for read and ten for write operations in the ACC have been used and the results averaged; the simulation results in Figure:5.11 show that the current in the ACC increases linearly with increasing output load. Thus not only does a strong drive need to be used but reducing the output load should also be considered. In a large datapath, careful floorplan design is important because the output driver and output load can be minimized when the ACC is placed near to the circuits it drives. The ACC of this FU has two read ports which are driven to different places; one (1R) provides the inputs to the multiplier and the ACC itself, the other (2R) goes to the shifter nearby. Thus a bigger drive has been used at the 1R port compared to the 2R port of the ACC.

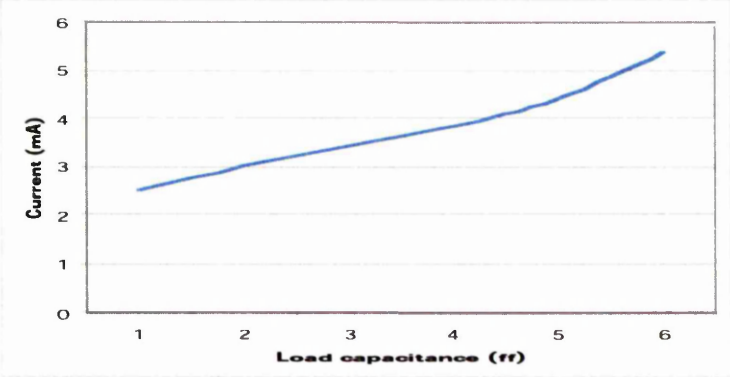


Figure 5.11: The relation between current and the amount of load in ACC

5.5 Configurable Memory

The regularity of typical DSP code allows multiple FUs to be employed without the power and area expense of dynamic scheduling hardware. For example, most modern DSPs, such as the TMS320C55x from Texas Instruments, use very long instruction words (VLIWs). In this case, program memory is fetched at the full rate demanded by the FUs

and a lot of power is dissipated. Although cache would be a possible method to ameliorate this, there is an energy overhead in searching for a hit in cache memory.

In this FU architecture, the VLIW encoded instructions are stored in advance of executing each algorithm in a configuration memory located in each FU. This is different from others commercial DSPs such as the Phillips REAL DSP [16] or the Infineon CARMEL DSP [17] which have a single global configurable memory which is only used for special instructions. In CADRE-s, the VLIW instructions for a particular algorithm are cached by software and looked up using a short-form instruction. Up to 64 instructions are kept in a configuration memory which is more than sufficient for any anticipated algorithm; for example the FIR algorithm with non-data arrangement previously described requires only 3 encoded instructions in a configuration memory.

Furthermore, the configuration memory instructions are completely user definable. The simple look-up avoids any tag overhead associated with a cache. The configuration memory makes the configuration of the FU flexible, enabling optimization by users. This flexibility has led to additional hardware costs, particularly in the implementation of the configuration memory. These costs can be justified by the flexibility and performance gained by users. In particular, the use of configurable memory embedded into the design is unique compared with the other DSPs mentioned above. This feature also allows each FU to operate on an independent instruction stream if required. The RAM is a custom part from the manufacturer and has a separate power supply (RAM supply) to enable the energy cost of operating it to be measured.

5.6 Control Unit

5.6.1 Asynchronous Control Circuit

Self-timed control circuits have been used in the functional units of the parallel asynchronous digital signal processor to avoid electromagnetic interference and clock distribution; these are of particular important in mobile applications. However, an asynchronous control circuit can slow down the whole the system performance as the control is more complex than synchronous control. Therefore, asynchronous design

techniques such as using an early done signal and hiding the recovery time for control circuits have been used.

A four phase handshake protocol is used to allow for a simpler design. This is implemented through the use of Muller-C elements which allow components to synchronize their phase of operation. As the FU is a self-timed unit in the execution state of the asynchronous pipeline, the FU is enabled by a request signal and a done signal is generated by the FU when the execution is completed. However, the done signal is able to rise before the actual result is completed. This is permissible because the input data of the FU has been latched and will only change after the done signal returns low. Therefore, the recovery time (return to zero) of the done signal can be concurrent with computation.

The control circuits of the FU as shown on the top of CADRE-s datapath in Figure:5.12 have been divided into 4 sections which can execute in parallel when the request (req) signal arrives.

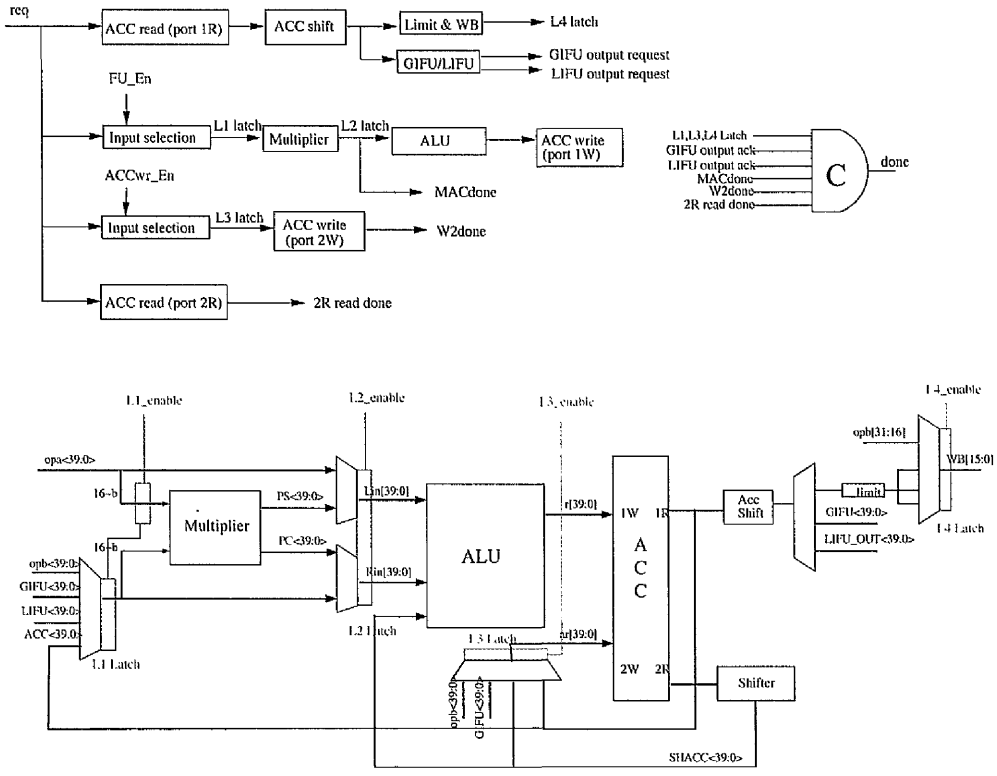


Figure 5.12: The control circuit diagram

Section 1

This part of the circuit control is concerned with sending the data from the FU via LIFU, GIFU, or the write bus (WB) (that is used to send a 16-bit result to the RAM). The control assumes that the data is ready from the preceding instruction. When the request signal (req) arrives, the required data is read from the accumulator register via port 1R. The data can be shifted one bit position at the ACC shift if required. Following this, the data is sent out to the LIFU, GIFU or WB bus depending on the instruction. If the WB bus is selected, the data will be latched at time L4.

Section 2

The second part of the control unit is concerned with the multiplier and ALU. As soon as the request signal arrives, the functional unit enable (FU_en) will control whether the functional unit arithmetic/logical processing is required or if this is to be bypassed on this request. In the case where there is an arithmetic and logic execution, the input selection (in front of the multiplier) will be performed before the data is latched at time L1. It is also necessary to ensure that the FU gets the correct operands since the data on GIFU is broadcast to all four FUs when one of the 4-FUs requires the data. As soon as the execution has finished in the multiplier and the results have been latched at L2, the MAC done signal goes high to inform the control that the adder operation in the ALU is beginning. When the ALU operation is finished, the result is stored in the ACC via port 1W.

Section 3

Section 3 controls the input of data from another FU into a FU's ACC. Similar to the second section, at the beginning of the control flow the ACC write enable signal controls whether this logic is going to be active or not. If active, then as soon as the data has arrived and is latched at L3, the ACC can start to write the data via the second write port (2W). The w2 done signal is generated when the ACC completes writing the data.

Section 4

This section is about reading the data out of an accumulator register via port 2R. This is activated by the request signal. The R2 read done signal signals the completion of the accumulator read.

Note that data written into the ACC has to occur after the ACC read has been performed. When all four sections complete their operations (data at L1, L3 and L4 latched, GIFU/ LIFU output acknowledge received, MAC operation complete, W2 complete and 2R read complete), the FU's done signal will go high. Then the recovery of the handshake signals returning to zero can occur immediately.

5.7 Delay Model and Tunable Timing Mechanism

5.7.1 Data Encoding

One technique for designing asynchronous (or self-timed) circuits is to encode the timing and data onto the same set of signals. For example, in a dual-rail scheme[103] each bit is signalled on a pair of wires, each indicating a particular data value and the arrival of that bit. This has the advantage that delays in processing and interconnection are accommodated automatically[103],[70], the resultant circuit being correct by design; unfortunately this design style results in circuits roughly twice the size of their conventional synchronous equivalent.

A compromise form of self-timed circuit is the single-rail, bundled-delay circuit[104]. In this, a single wire per bit is used to represent each bit of data (as in a synchronous datapath). The data is accompanied by local handshake signals (request and acknowledge) which replace the clock. As previously stated, an example of this style is the micropipeline[66]. The asynchronous single-rail design is shown in Figure:5.13; the timing of each stage can be either data dependent or fixed-delay (with a timing margin).

The method adopted to vary the timing in an asynchronous control circuit is to vary an additional supply voltage. This method is relatively easy to apply, as is demonstrated. In addition it offers many other advantages, such as allowing several different matched delays without incurring extra area and power overheads. Tuning the control delay to the datapath also improves performance and offers a flexibility not apparent in other

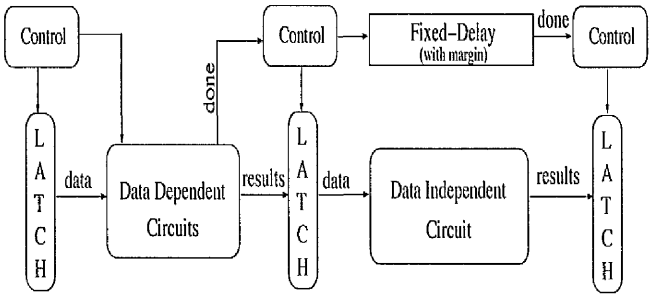


Figure 5.13: Bundled Data Asynchronous Structure

asynchronous timing approaches. It therefore improves the likelihood of obtaining working asynchronous circuits at the first attempt and should lead to greater acceptability of asynchronous design techniques by easing the design problem.

5.7.2 Data Dependency

The best way to exploit asynchronous operation is when data dependency is present. For example, addition can cease when all carries in the adder are detected to be valid. Completion logic for addition can in itself incur additional delay so needs careful design. Designers have attempted to circumvent this problem; for example Nowick et al. proposed a method called '*speculative completion*' for use in a single-rail asynchronous datapath[105]. This uses several different matched delays that allow each component to operate at a different speed. This was primarily to allow a subsystem to complete more rapidly when it has to do less processing. It could be adapted to include the fine-tuning required here, but at the expense of significant extra complexity.

Unfortunately, not every part of an asynchronous system is data dependent. For instance, a FIFO is able to exploit a clock quite easily and gives a better performance compared to using asynchronous circuits because of its control circuit overhead. Only if data dependent operations dominate operations can the asynchronous performance be better than that in a clocked system.

This work is targeted at an asynchronous DSP design and the major component is the multiply-accumulate unit. Thus data dependent operations dominate the computation and

hence careful design of the timing for this unit can yield a significant performance improvement over a clocked system.

5.7.3 Delay Model

In a bundled data, self-timed circuit the processing delay must be modelled rather than measured. This may done in several ways. Using a copy of the critical path for data processing can give a very good model delay. This would be typified by, for example, the '33rd bit' of a 32-bit register which always causes a transition when it is read. By building the timing model in the same way as the circuits are modelled and placing it in the same physical substructure it is reasonable to expect a close match in manufacturing process variation and operating conditions (temperature, supply voltage), and therefore a well-matched delay.

Circuits with more 'random' logic may have an ill defined critical path and are harder to model. They can be simulated using chains of gates as delay elements, but with less confidence that these will suffice under all conditions.

With any matched delay (including an external clock) it is necessary to add some timing margin. The size of this margin depends on how 'good' the delay model is. With an ill-matched delay a large margin-perhaps as much as 50%-must be allowed, leading to performance degradation.

A big advantage of using an external delay model (i.e. a clock) is that, if the silicon is slower than expected the clock frequency can be reduced and the circuit will work. With matched delays built on chip this is not possible; if any single delay is too short the device will not function. This encourages caution in the designer, with the result that delays tend to be conservative and made longer than they need to be. The mechanism proposed here enables a delay to be tuned to the datapath in exactly the same way that a clock is tuned to logic.

The advantage is that the timing of the asynchronous circuit can be slowed if the timing is 'on the edge' giving a fail safe option. However, if as would be usual, the timing is

conservative then the timing can be reduced to speed up the handshakes and improve performance.

5.7.4 A Tunable Timing Mechanism

The bounded delay for a self-timed functional unit is the timing model which indicates that the operation is complete. The normal bounded delay is a fixed timing model. With this model, an asynchronous system will fail if the delay of a functional unit is more than the bounded delay. Since circuit delay is roughly proportional to supply voltage, controlling the delay's power supply independently from the datapath voltage enables the designer to adjust the timing margins. The principle is shown in Figure:5.14. The datapath operates from the normal supply voltage while the matched delay paths are connected to an external supply V_{DD2}. Within margins, V_{DD2} can be greater or less than the normal supply V_{DD}.

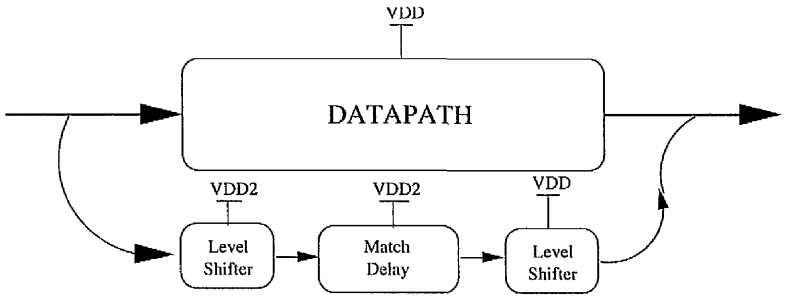


Figure 5.14: Tunable delay principle

Voltage scaling is widely used to gain power savings. Scaling down the voltage also causes the delay to rise since the delay is proportional to $CV_{DD}/K(V_{DD}-V_{th})^{\alpha}$ [4], where V_{th} is the transistor threshold, C is the circuit capacitance, K is a circuit constant from simulation and α is a technology-dependent factor which varies between 1 and 2. Therefore, lowering the supply voltage used in the delay model gives a larger delay margin, and vice versa.

The matched delay is nominally the same as the datapath delay and V_{DD2} is used for fine tuning. V_{DD2} is initially set to V_{DD}. If the margins are too conservative, V_{DD2} is adjusted

to be higher than V_{DD} reducing the timing margin and enhancing system performance. However, if failure results due to 'timing on the edge' then V_{DD2} can be adjusted to be lower than V_{DD}, slowing the control path down. The only disadvantage of this approach is the requirement for an additional power supply. However, the simplicity and flexibility of this approach and its ability to assist in the realisation of fully functional, complex asynchronous systems outweights this extra cost.

Figure:5.15 shows more detail of the timing mechanism. A level shifter is needed to ensure that control signals are supplied to the datapath at a voltage V_{DD}. The arrangement in Figure:5.15(b)[106] is used to ensure the output swing is full rail and avoid static current flow in the subsequent stage. Either T1 or T2 is on, and this pulls down the relevant side of the amplifier sufficiently to turn T4 or T3 on in the opposite side. The other PMOS transistor then turns off and the level shifter stabilises without any further static current flow.

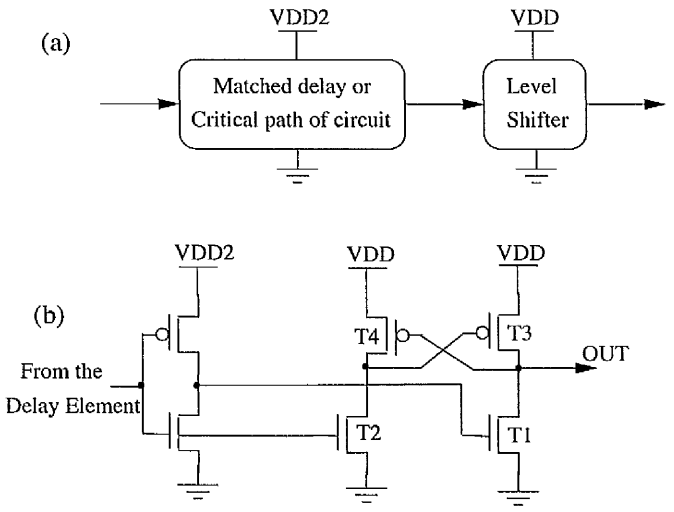


Figure 5.15: (a) a tunable timing mechanism (b) voltage level shifter circuit

5.7.5 Experimental Results

To illustrate the application of this tunable timing mechanism, the delay for the critical path through a 16x16 multiplier has been used as the delay model. This delay model has been implemented on a 0.18 micron process using a mixture of pass transmission gate

(PTG) and conventional CMOS circuits. The nominal delay is approximately 1.5ns (which is roughly equivalent to a string of 12 NAND gates). Two different power rails are included in the design. The results (Figure:5.16) show the percentage variation of the delay as V_{DD2} is varied from its nominal operating value of 1.8V. A exponential shape as would be expected from the delay equation, is shown in Figure:5.16.

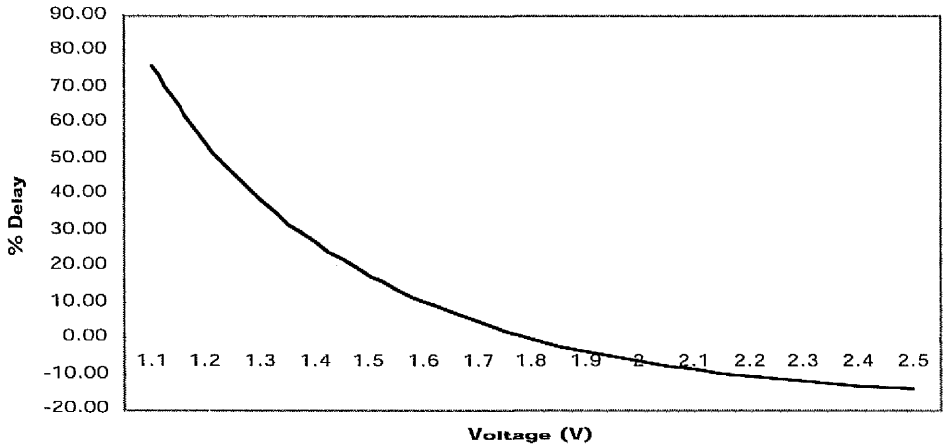


Figure 5.16: Simulation of the timing versus scaling voltage

Figure:5.16 suggests that the delay margin can be increased by about 50% by scaling the voltage down by about 30%. However, such large variations in V_{DD2} are not envisaged as the mechanism is essentially one of fine tuning. Thus margins of ± 10% are more realistic and Figure:5.16 shows that over this range the variation in delay is approximately linear.

Figure:5.17 shows the theoretical plots of the delay $CV_{DD2}/K(V_{DD2}-V_{th})^{\alpha}$ against V_{DD2} for different values of α together with the simulated results. From the graph, it is apparent that $\alpha \cong 1.5$ for the process used. This enables designers to calculate the value of V_{DD2} corresponding to a particular delay.

The use of different logic families to implement the matched delay has been investigated. The different arrangements are shown in Figure:5.18. The CMOS matched delay is built using a chain of eight inverters and a buffer at the output whilst the PTG delay line uses two sets of four pass transmission gates each with a drive inverter; the PTG chain also has

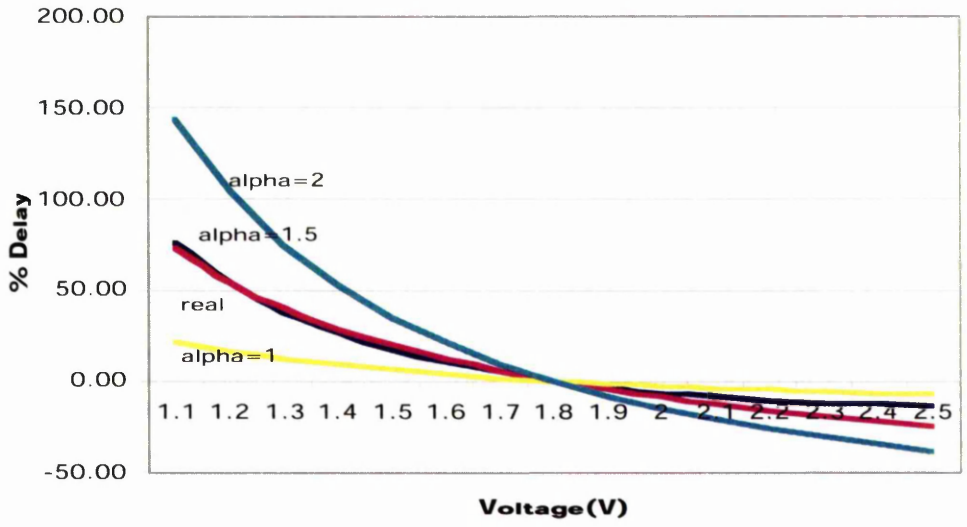


Figure 5.17: Plot of the theoretical and simulated delay

an output buffer. Finally, two sets of four NMOS transistors with a swing restoring inverter and an output buffer are connected together to form the SPL delay line. All delay lines have been attached to a load capacitance of 10 ff, which is equivalent to 4 inverters at the output. In all 3 configurations the voltage has been varied from 1.5V to 2.1V.

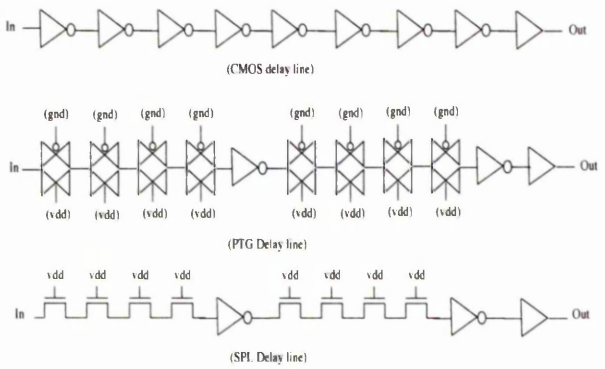


Figure 5.18: Matched delay on different logic families

In Figure:5.19, the simulation results show that the delay of the PTG delay line starts to grow non-linear when the voltage is scaled down from 1.8V to 1.5V. However, it is clear that the SPL delay line gives the greatest delay variation when the voltage is varied from

1.8V to 2.1V. Thus the SPL matched delay is the most suited for use in speeding up circuits in a bundled data asynchronous design. Figure:5.19 also reveals that the CMOS matched delay has the potential to achieve less performance improvement than either the PTG or SPL chains.

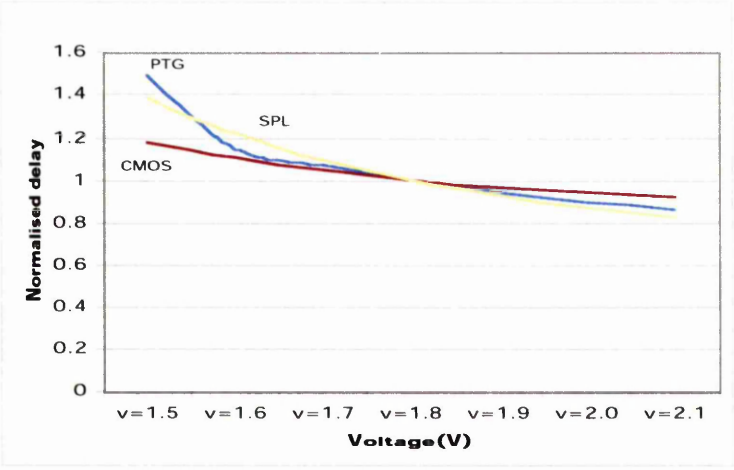


Figure 5.19: The relation between delay and voltage scaling of various delay lines.

5.7.6 Extending the Concept

The mechanism so far described adjusts the timing on all control circuits. However, it is likely that a designer adopting this approach would require a more individual control of each delay element.

Clearly an individual supply for each control circuit is not viable. This leads to the compromise scheme shown in Figure:5.20. Here the logic for a matched delay can be chosen from one of three supplies depending on the state of sw1, sw2 and sw3. One of the these is low so its associated PMOS transistor is on whilst the other two inputs are high, turning their associated transistors off.

The specification of sw1 to sw3 for each matched delay would need to be held in on-chip registers with a default loaded on initialisation. Suitable nominal starting voltage for V_{DD1} to V_{DD3} would be 0%, +15% and -15% of the normal datapath supply voltage V_{DD}.

Again this yields a low area and low complexity solution to a difficult problem, at the expense of an additional two power supplies. The principle can, of course, be further extended, but requiring even more supplies appears excessive and probably indicates poor matched delay design!

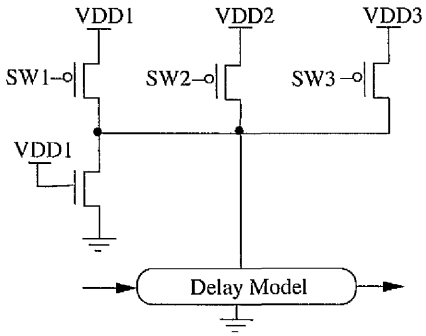


Figure 5.20: Different Matched Delay Selection

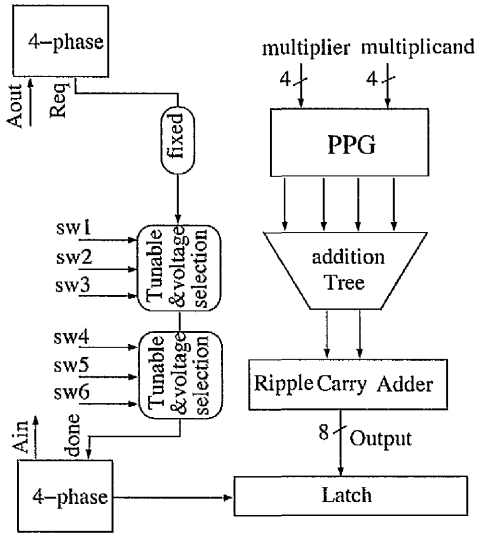


Figure 5.21: A simple 4x4 multiplier for tunable timing mechanism evaluation

A bundled asynchronous circuit is not able to achieve a higher performance if a worst case delay has been incorporated. In addition there is no benefit from tunable timing if most of the operations are executed with a worst case delay. Since the FIR filter is a general DSP

application, the operands used were investigated and it was found that more than 70% of the byte data from random data sampling was zero whilst more than 50% of the byte data from speech data sampling was found to be zero. Therefore, most of the execution time for the multiplication in the FIR algorithm is less than the worst case delay. In this case, a performance improvement can be gained from the tunable timing technique.

As a mean of testing and demonstrating tunable timing mechanism, a data dependent 4x4 tree multiplier was implemented as a bundled data path of an asynchronous pipeline as shown in Figure:5.21. Four-phase handshaking control was used and the matched delay lines with the voltage selection circuits accompanying the data path are shown.The tree multiplier hardware consists of three parts as described previously. The first labelled PPG generates the partial products; it is implemented by multiplexers and has a fixed delay. Then, the partial products are added in a tree structure where carry save adders have been used. Although, the carry save adder can produce the output without waiting for the carry propagation, the timing delay of this stage still depends on the input data; tunable timing is therefore applied here. Finally, the sum and carry of the carry save addition are added in the third part. This section must resolve all the carries progressively from the LSB to MSB. A ripple carry adder has been chosen for this last part because of its small area. In synchronous design, the ripple carry adder can give a very poor performance because the clock has to allow for the worst case time which depends on the length of the words being added. Thus, a self-timed circuit with the tunable timing mechanism is able to provide a significant performance improvement over a clock design when a variable delay is applied on an operation by operation basis.

The tunable delays can be set to one of three values corresponding to a supply voltage of 1.5V, 1.8V or 2.1V as shown in Figure:5.20. To provide a suitable voltage supply, at the beginning of the timing slot, the input (multiplier) is detected and the switching control signals are generated corresponding to this data input. Because a circuit of PPG is implemented by multiplexers and provides a fixed-delay, a tunable timing is not applied at this stage. However, the timing in the addition tree depends on its inputs. If any two bits of the input (multiplier) are one, two partial products become zero and a normal voltage (~1.8V) is set for an average speed by signal 'sw1'. Meanwhile, the higher voltage (~2.1V) is set by signal 'sw2' for a fast speed, when all four bits of the input are zero since

the partial products are zero. Finally, the worst case timing occurs when all four bits input are one and all four partial products are active. Here signal 'sw3' is used to set a lower voltage (~1.5V). Sw4-sw6 are also generated at the ripple carry adder stage in a similar way. It can be expected that the highest voltage will not be set often in the addition tree or ripple carry adder. In fact, since there is no carry propagation in the addition tree, carry propagation can only occur in the ripple carry adder so in practice only the ripple carry adder is likely to benefit from adjusting the voltage according to the number of zero PPs.

The results obtained are shown in Figure:5.22 for random data inputs. This shows that the self-timed design with tunable timing can achieve a significant performance improvement over a clocked design (worst case timing) when the switching voltage control signals are decoded from the input data. With a set of the test numbers extracted from random data sampling, the matched delay implemented with conventional CMOS has a better performance than the clock design of about 15% whilst the performance improvement of a PTG matched delay is about 27%. Finally, the biggest performance improvement, 32%, is found in a SPL matched delay. The results from speech data sampling are similar with the percentage of the performance improvement over the clocked design in the three matched delays being 14%, 25% and 30% for the CMOS, PTG and SPL delay circuits, respectively. The tuning mechanism is also suitable for applying at the instruction or pipeline level since some systems lose performance because the worst case timing has been applied in every instruction or pipeline stage.

In all the cases where the multiplier exhibits data dependent delays the environment interacts with the circuits to determine its performance. Alternatively, this interaction is not required if the voltage control signals are embedded in the instruction set. In addition, the data dependency can be limited by the speed of supplying the inputs in practice. To remove this restriction in an implementation, extra buffering can be used to supply the inputs; however, the cost of increasing the power consumption and area will be a penalty.

5.8 Voltage Scaling Versus Energy Consumption

To minimize energy at the device level, lowering the supply voltage is now generally adopted. The techniques to do this have been studied and centre around the ability to change the voltage dynamically[107],[108],[109],[110],[111] or the use of multiple

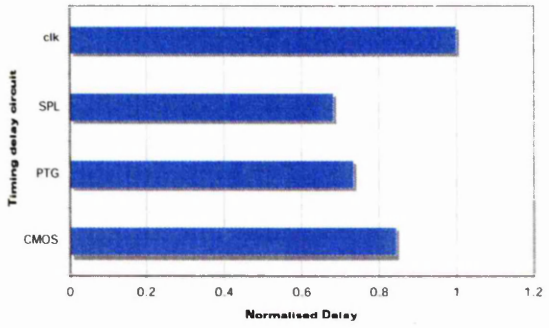


Figure 5.22: Relation between the normalised speeds of timing delay circuits for random input data

voltages[112],[114],[115]. Although dynamic voltage scaling (DVS) incurs a latency cost, typically in the microsecond range, for switching the output of a programmable power supply[114], the coarse grained computational entities, like task scheduling by an operating system, can be used to satisfy this range of switching voltage delay[111]. There are also many research papers[117],[118], [119],[120] proposing solutions to the DVS problems with the operating voltage setup.

With the use of multiple voltages, the algorithm named MOVER[112] (Multiple Operating Voltage Energy Reduction) was proposed to arrange the datapath scheduling with multiple supply voltages on a DSP; results suggested that 0-50% of the energy consumption could be saved compared to a single voltage DSP. Kaushik Roy[112] also found that the datapath resource requirements vary greatly with respect to the number of supplies and that area penalties ranged from 0-170%. Therefore, Dynamic Voltage Supply (DVS) looks a more promising approach and has a high potential to improve the energy efficiency of a DSP processor for burst-mode operation (Emax + Eidle) without the penalty of using multiple supplies. However, DVS designs need to consider carefully the switching voltage delay.

Typically, a DSP has been used in portable applications, where power is a critical issue. Most portable applications such as mobile phones, PDAs, PocketPCs and MP3 players do not require the full processing power all the time. Therefore, lowering the voltage when the DSP is lightly loaded can achieve an energy efficiency since power is proportional to

V^2_{DD}. However, designers need to be aware of the leakage current and the overhead in the voltage when dynamic voltage scaling is applied[113].

In addition, scaling down the supply voltage is usually applied to a parallel architecture to reduce the power consumption caused by increasing area; however the throughput of the system remains the same. The 4-way parallel demonstrated DSP in this thesis has been designed and implemented to support scaling the voltage supply where pass transistor logic is used throughout the design. The capability of pass transistor logic to operate under dynamic, irregular voltage supply conditions needs to be investigated because most research works have not concentrated on voltage scaling effects of pass transistor circuits. This will be discussed further in the next chapter and the effect of applying different voltage supplies to CADRE-s will be given in Chapter7.

The circuits for performing operations such as Hamming distance, normalization, shifter and accumulator and the control and timing mechanisms have been described in this chapter. The power consumption and performance of this logic requires energy efficient, high performance XOR/multiplexer circuits. These are examined in the next chapter.

Chapter 6: Energy Efficient Circuit Design Methodology

According to the consideration of the logic arrangements in the previous chapter, it is clear that the multiplier, adder, shifter, Hamming distance and normalization have 4-2 compressors, XORs and multiplexers as their major circuits. Latches to stop unnecessary transitions also feature heavily in the design. Thus this chapter concentrates on the energy efficient design of circuits for these functions.

An important key to achieving energy efficient designs is to focus on lowering power and increasing performance through all the levels of the design, rather than applying only one or two strategies at the algorithmic and architectural levels. There are several methods that are able to yield an energy efficient circuit implementation. However, a complex block, such as the arithmetic unit requires a coherent energy saving technique, rather than one or two techniques. Brodersen et al[122] and Zyuban et al[121] also suggest that true power minimization can only be achieved when the energy reduction potential of all adjustable variables such as transistor size (W), voltage supply (V) and threshold voltage (V_{th}) are balanced. This is because each of these variables carries a certain energy reduction potential per delay cost at each point of the power-delay space. In addition, circuits used in the low energy functional unit, have to be designed and implemented focusing on energy efficiency rather than just low power dissipation.

The work in this chapter have been done by the author using post-layout netlists to run SPICE simulations in the Cadence environment because the SPICE simulator gives accurate results and is suitable for relatively small circuits. In addition, the testing environment has been set up as shown in Figure:6.1with both input and output loads taken into account to make the post-layout simulation more realistic.

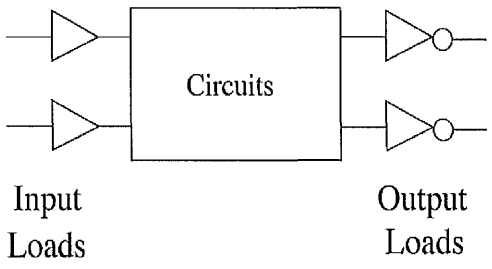


Figure 6.1: A test environment for the circuit simulation

6.1 Logic Style

The choice of logic style needs to be considered for more complex full custom cells such as a latch and 4-2 compressor, as the logic style affects the power dissipation, speed, wiring load and the size of transistor. A different logic style can give a different energy consumption. Therefore, logic styles having a good energy efficiency potential are analysed in this chapter for their power dissipation under different load conditions. The implementation of a XNOR gate in four types of logic style, conventional CMOS, dynamic logic, pass transmission gate (PTG) and single-ended pass transistor (SPL), are discussed here.

The output transitions of static CMOS and both pass transistor logics occur when the inputs change. However in dynamic logic extra transitions will occur during the precharge phase. The circuit design in all four styles is shown in Figure:6.2; the transistor sizes are also shown. It should be noted that the XNOR operation is essentially that of a multiplexer so the circuits in Figure:6.2 can also be viewed as different implementations of the much used 2-1 multiplexer. The output (Z) of the CMOS XNOR is pulled low when inputs A and B are different, whilst the dynamic XNOR gate evaluates its output when ϕ=1. In the PTG XNOR, input B is passed to the output when input A is high otherwise, $\overline{B}$ is passed to the output. Finally, the SPL XNOR gate uses a NMOS 2-1 multiplexer to pass $\overline{B}$ to an inverter when A is high and B to the inverter otherwise. The post-layout netlists have been simulated using SPICE with different load capacitances of 10ff, 20ff, 30ff and 40ff; 10ff

is equivalent to a load of four inverters. The test input for the circuit uses all combination of the two inputs. The energy versus load capacitance is shown in Figure:6.3.

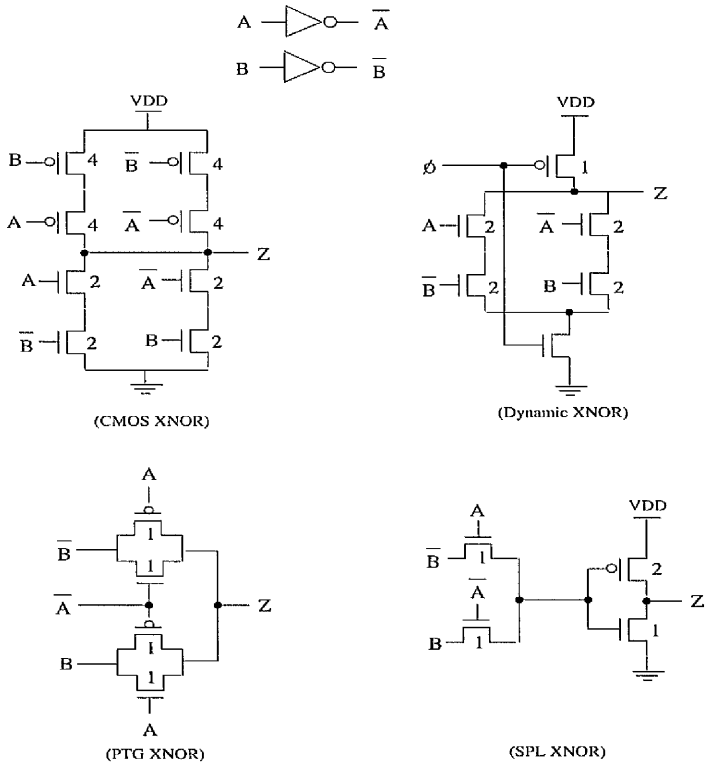


Figure 6.2: Different logic families of XNOR circuits

In Figure:6.3, the PTG circuit gives the best energy efficiency when the load capacitance increases whilst the conventional CMOS circuit is the worst one. Meanwhile, the SPL circuit with an inverter at the output is the second most energy efficient. This explains why an inverter or buffer should always be attached to the output of pass transistor logic. The results confirm that pass transistor logic has the potential to be energy efficient.

From the literature, pass transistor logic is promising for low energy and maps well and easily onto the arithmetic units in CADRE-s. This leads to exploring and analysing other pass transistor logic styles such as double pass transistor logic (DPL)[123] with full-swing restoration, complementary pass transistor logic (CPL)[124], swing restored pass-transistor logic (SRPL)[125], energy economized pass-transistor logic (EEPL)[126], push-pull pass-transistor logic (PPL)[127] and single-ended pass transistor logic (SPL).

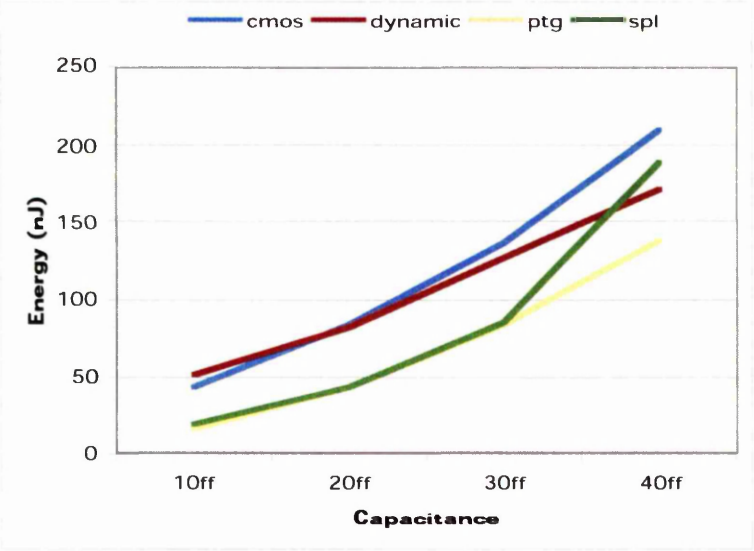


Figure 6.3: The relation of energy and load capacitance of various XNOR

The 2-1 multiplexer circuit with a load capacitance of 10ff is chosen for this comparison and the circuit for each style is shown in Figure:6.4. The multiplexer is the most commonly used cell in arithmetic units. The CPL multiplexer consists of two NMOS logic networks and two inverters which produce the complementary output signals. The CPL has good output driving capability because of the output inverters and is well suited to dual-rail signals but it is a poor choice for low power circuits because of the number of transistors. The SRPL circuit is derived from CPL. The output uses cross-coupled inverters as a latch structure; these perform both swing restoration and output buffering at the same time. However, transistor sizing on this type of circuit becomes difficult because the inverters have to drive the outputs and be overridden by the NMOS network. DPL uses both PMOS and NMOS transistors in parallel and includes two inverters for output buffering; swing restoration is not required because of its full swing outputs. However, the use of a large number of PMOS transistors plus the need for double the number of transistors make DPL uncompetitive compared to other pass transistor logic. In EEPL, the PMOS pull-up transistor is connected to the complementary output signal instead of V_{dd}. PPL can be regarded as a CPL without the output inverters. PPL uses a PMOS pull-up device on one output and a NMOS pull-down on the other. Unfortunately, PPL does not work when $V_{dd} < V_{tn} + |V_{tp}|$.

The simulation results obtained by the author using all combinations of data and control inputs of 2-1 multiplexer in both performance and power terms are reported in Table6.1 and relate to a 0.18μm geometry and 1.8V supply voltage. From the energy and energy delay product results, three circuit types; CMOS, PTG and SPL are attractive for energy efficiency. The PTG multiplexer operates at less than 0.1ns and consumes the smallest energy of 0.018nJ. The SPL multiplexer is the second best for energy although it has one of the longest delay times. CMOS is the third best in energy terms being slightly faster than SPL but consuming slightly more power. Meanwhile, the other pass transistor logic styles can be ignored as they are relatively energy inefficient.

Of these three styles, PTG has been chosen for a full custom implementation of the CADRE-s datapath because it offers the best performance and energy. Furthermore, it occupies less than half the area of CMOS. The long operating time of SPL makes it look unattractive despite having lower energy than conventional CMOS. CMOS would be another good compromise choice offering both good performance at reasonable energy and of course would have the advantage of being able to use a standard cell library leading to a greatly reduced design time. However, the PTG style offers not only the best energy efficiency but also offers the novel opportunity to study a pass gate implementation in a large, complex system.

Table 6.1: Power and performance of 2-input multiplexers based on different type of pass transistor

Circuits types	Delay (ns)	Power (mW)	Energy (pJ)	EDP (pJ*ns)
CMOS	0.128	0.2023	0.0259	0.0033
CPL	0.126	0.6777	0.0854	0.0107
DPL	0.118	0.7740	0.0913	0.0107
EEPL	0.132	0.6561	0.0866	0.0114
PPL	0.120	0.3916	0.7049	0.0846
PTG	0.096	0.1872	0.0180	0.0017
SPL	0.130	0.1890	0.0246	0.0032
SRPL	0.130	0.2772	0.0360	0.0047

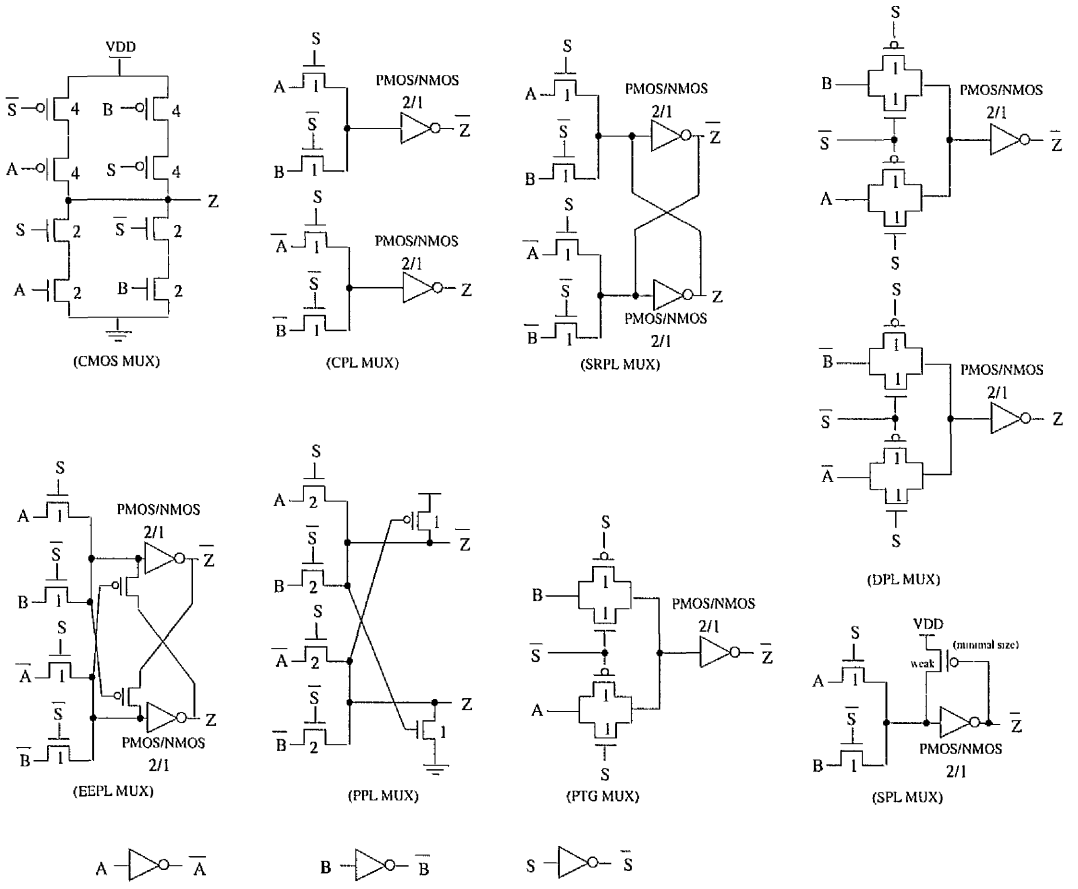


Figure 6.4: 2-input multiplexers based on different type of pass transistor

6.2 Pass Transistor

The pass transmission gates (PTG) is a good design methodology for CMOS because of the small number of transistors required as the results show in the previous section. A PTG gate is simply two mutually exclusive switches plus an output driving inverter where a switch only comprises a single minimum sized NMOS and PMOS transistor. So the area of the whole design is relative small.

The strength of an output is measured by how closely its output '1' is to the supply voltage. NMOS is almost perfect when passing a '0', and this is therefore called a *strong '0'*. However, NMOS has a *weak '1'* output when passing '1' because the voltage level of the output is somewhat less than V_{dd} (=V_{dd} - V_{tn}).

PMOS has the opposite behaviour, passing a strong '1' but a weak '0'. Therefore, by combining a NMOS and PMOS transistor in parallel, called a transmission gate, it forms a gate in which '0' and '1' are both passed strongly as shown in Figure:6.5. The effective resistance of a unit NMOS transistor passing '0' and '1' can be modelled as R and 2R respectively, whilst passing '0' and '1' of a unit PMOS transistor as 4R and 2R[128]. Thus the effective resistance from passing a '0' through the pass transmission gate is R || 4R =(4/5)R and passing a '1' is 2R || 2R = R. Hence, a pass transmission gate can be seen as a unit transistor with an effective resistance of approximately R for both input levels. This makes the propagation delay (t_{pd}) of a PTG small; the resistance of an ideal inverter would be 3RC which is three time larger than that of a PTG.

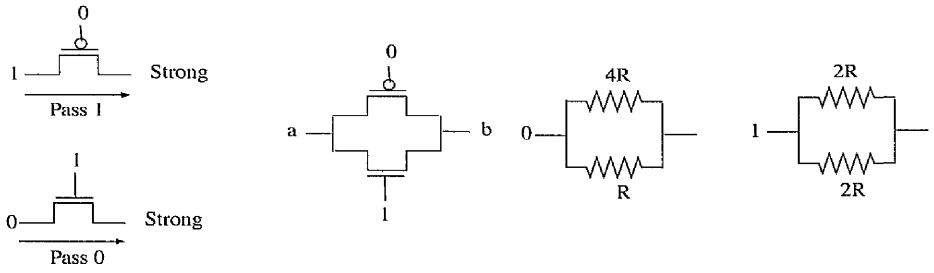


Figure 6.5: The behaviour of the pass transistor

6.2.1 Fan-in and Fan-out of PTG

In an energy efficient system, the importance of fan-in and fan-out has been demonstrated in computing the Logical Effort (section 2.5.2). This can be used for a speed estimation of CMOS circuits. However, the fan-in and fan-out for pass transistor logic has not previously been considered. The inputs (fan-in) of a pass transmission gate are the data input (In) and the select gate inputs (S and $\overline{S}$) as illustrated in Figure:6.6. Therefore, a pass transmission gate has a fan-in of 3. The fan-out of a PTG is the number of gate inputs that is driven by a PTG output. In Figure:6.6, a PTG has a fan-out of 4 since each inverter input connects to 2 transistors. The correct ratio of the increase in transistor sizing in successive logic stages can affect timing in cascaded logic stages and the power consumption. Therefore, logical effort can be used to find out the optimal energy consumption of the design.

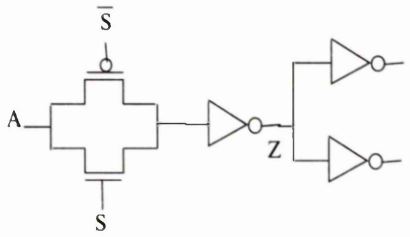


Figure 6.6: Fan-in and fan-out of pass transmission gate

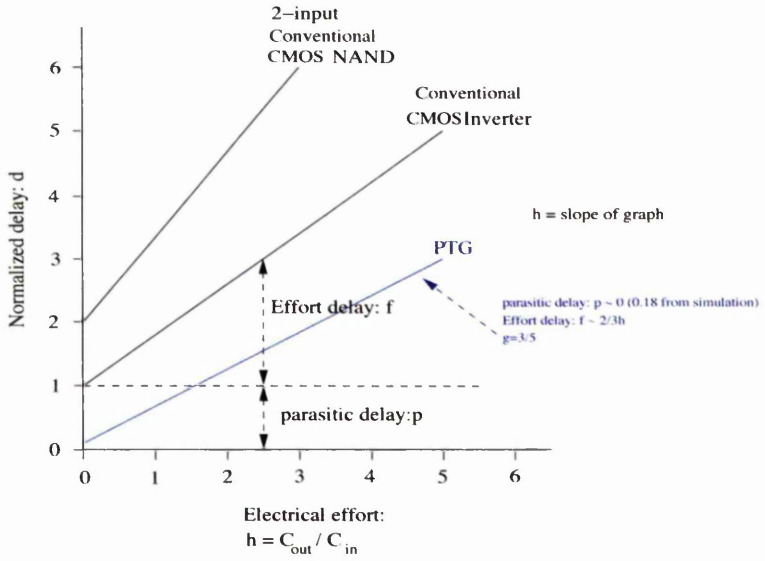


Figure 6.7: The relationship between normalized delay and electrical effort (h) of PTG

The delay comparison of an inverter, 2-input nand and a PTG gate is shown in Figure:6.7. By applying different electrical efforts (h), which is the ratio between C_{out} and C_{in}, the normalized delay can be extracted from a SPICE simulation; the normalised delay is the average delay obtained from all combinations of the input(s). The delay can be calculated from the summation of the effort delay (f) and parasitic delay (p) where the parasitic delay can be found by setting the electrical effort to zero (h=0) and the effort delay is the slope of the graph.

The computation of the logical effort for a 2-input nand gate and inverter is relatively easy. As shown in the graph, the inverter has a parasitic delay equal to 1 and an effort

delay equal to 2 when the electrical effort is 2, so the logical effort (g) is 1. Similarly, the logical effort of the PTG (at point Z in Figure:6.6) can be calculated and is about 3/5. The logical effort of a PTG is a way of showing how difficult it is for that PTG to drive its output and how that relates to its delay. In the logical effort of the PTG above, the PMOS and NMOS devices are minimum size. Its output inverter gives PTG a good logical effort performance. This logical effort can be used to specify the proper number of logic stages on a path and the best transistor sizes for the PTG gates.

6.2.2 Transistor sizing

Another energy efficient circuit design methodology described here is transistor sizing; this is the variation of the transistor widths and lengths: W/L. This is especially important in pass transistor logic. Usually, when the design has been implemented by synthesis tools, cell elements with different transistor sizing are required to improve the maximum drive capability under various load conditions. The general technique which is used in the synthesizer is a simple look-up table to choose the proper gate drive corresponding to the estimated load. This means the synthesis tools cannot provide an energy efficient design due to the load being estimated rather than accurately known; this makes selecting an accurate drive capability impossible.

Table 6.2: Size of transistor of 2-1 PTG multiplexer

CASE	Width of pass gate (μm)		Width of Select inverter (μm)		Width of Output inverter (μm)	
	NMOS	PMOS	NMOS	PMOS	NMOS	PMOS
1	0.28	0.28	0.28	0.28	0.28	0.28
2	0.28	0.28	0.70	1.24	0.28	0.28
3	0.80	0.80	0.70	1.24	0.28	0.28
4	0.80	0.80	0.70	1.24	0.70	1.24
5	1.00	1.00	0.70	1.24	0.70	1.24
6	0.28	0.28	0.70	1.24	0.70	1.24
7	0.28	0.28	0.70	1.24	0.36	1.46
8	0.80	0.80	0.70	1.24	0.36	1.46
9	0.80	0.80	0.28	0.28	0.36	1.46
10	0.28	0.28	0.28	0.28	0.36	1.46

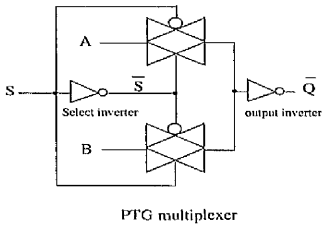
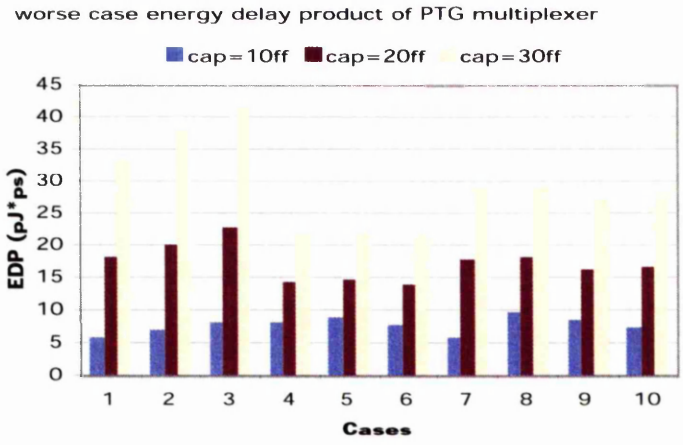


Figure 6.8: PTG multiplexer

The multiplexer circuit which is used as the main cell in the design presented in this thesis has been investigated to see how much improvement the design can achieve when transistor sizing is used. It is implemented using PTGs plus an output driver as shown in Figure:6.8. The multiplexer circuits differ from those in Figure:6.2 in that the PTG circuit has an output inverter driver. The size (width) of transistor of the different sections of the PTG 2-1 multiplexer used in this simulation are shown in Table6.2; the minimum length (of 0.28μm) is used throughout. Ten different sizing options have been used and the circuits have been simulated at load capacitances of 10ff, 20ff and 30ff. The transistor sizes have been chosen by the author from running schematic SPICE simulations and observing the lowest point on the power versus time graph for different transistor sizes. Case1 uses a drive '0' strength at the output. A NMOS width of 0.7μm with a PMOS width of 1.24μm gives a drive strength of 1 whilst a NMOS width of 0.36μm and PMOS width of 1.46μm represents a 1.5 drive strength. The pass gate transistors are always chosen to be the same size and this only affects the transmission time of the data. The transistor size has begun with the minimal transistor width of this 0.18μm geometry process technology, 0.28μm. In the second case, the select inverter has been sized for both NMOS and PMOS to 0.70μm and 1.24μm, respectively giving a drive strength 1. The results of the investigation into the effects of transistor sizing on the driven capacitance and its impact on the maximum energy efficiency of the PTG multiplexer are shown in Figure:6.9. Notes that all combinational inputs of 2-1 multiplexer have been used for the simulation.

The results relate to the maximum output edge time (10% to 90% of maximum rising or falling edge) and the average propagation delay (average of 50% input to output edge times). In practice, load capacitance is very important and has a large effect on both the

a)



b)

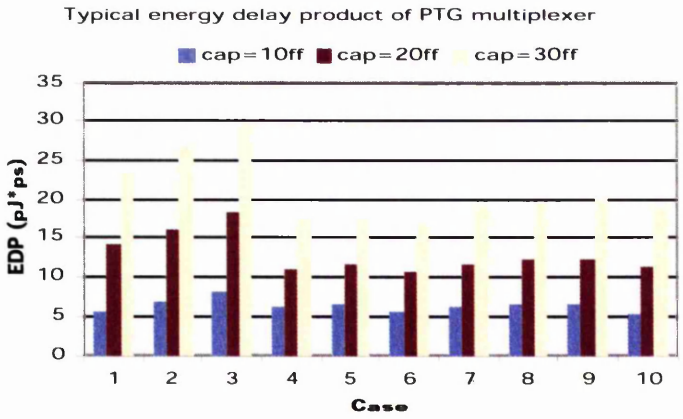


Figure 6.9: PTG 2-1 multiplexer results

(a) maximum output edge time (b) typical propagation delay

performance and power of the system, especially in a large design. So different load capacitances have been attached to the multiplexer in the simulation. As can be seen in Figure:6.9, using minimal transistor size (case1) cannot achieve optimum energy efficiency when the load capacitance is increasing. It can be concluded from these results, that the CMOS pass gate transistors for minimum energy delay should be the minimum size and that the select inverter should be a drive 1 gate (PMOS/NMOS having width = 1.24μm/0.70μm); this is capable of driving the select signals on 2 or 3 multiplexer gates. The optimum size for the output inverter is case 6 which is the best energy efficiency to the driven load (PMOS/NMOS width = 1.24μm/0.70μm). The energy delay product

figures do not include the leakage power which is small for this process provided the transistor sizing above is adopted.

6.3 Voltage Scaling in Pass Transistor Circuit

This section considers how a multiple or dynamic voltage supply will affect a PTG circuit in terms of energy efficiency. This is because it is very important to know how robust the circuit is under non-standard conditions. As mentioned before a dynamic voltage supply (DVS) can significantly improve the energy efficiency of a DSP because of the nature of a DSP processor. If operating at maximum speed, then the highest supply voltage should be used whilst the supply voltage should be scaled down when the system is idle.

Furthermore, scaling the voltage down in pass transistor logic may increase the leakage current; this may become important especially when the technology is scaled down. This section presents an example of an 8-bit ripple carry adder (RCA) based on pass gates to perform the XOR functions to form the sum and conventional CMOS circuits on to form the carry. The 0.18μm process technology, operates correctly when the voltage is scaled up or down by ±10%. The post layout SPICE simulation measurements enabled the relation between the energy delay product and voltage scaling to be calculated and is shown in Figure:6.10. This histogram shows that as the voltage is scaled down, PTG retains its better energy efficiency compared with CMOS.

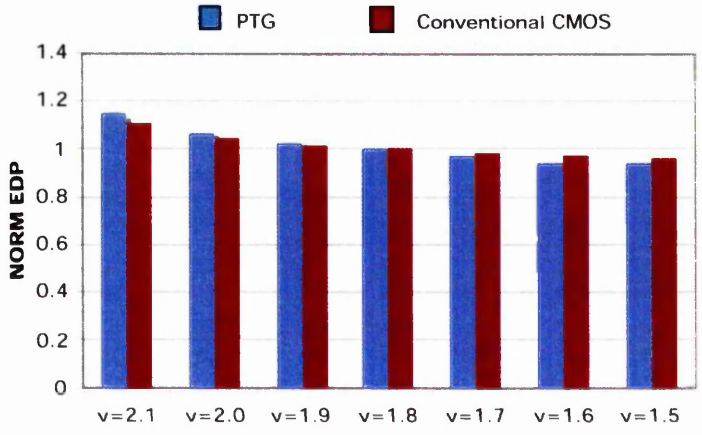


Figure 6.10: Effect of supply voltage on energy delay product in an 8-bit ripple carry adder

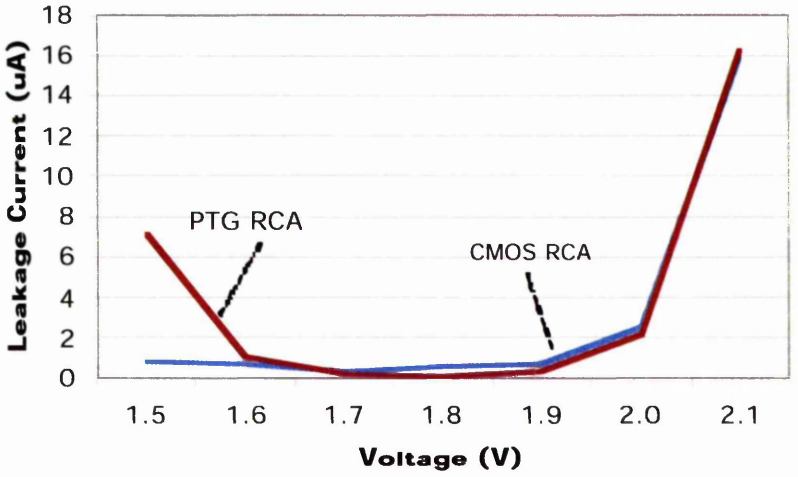


Figure 6.11: The relation between leakage current and scaling voltage of the CMOS and PTG ripple carry adder.

Both the CMOS and PTG ripply carry adder are also investigated for leakage current using the SPICE simulator when the adder inputs are constant after operating some random inputs to observe the leakage current. Figure:6.11 shows the leakage current which is drawn from the supply for the CMOS and PTG ripple carry adders at different supply voltages from 1.5V to 2.1V. As can be seen, the leakage current of the pass transmission gate is lower than the conventional CMOS when operated at the normal supply of 1.8V. However, the leakage current of the PTG ripple carry adder increases exponentially for voltages below 1.6V whilst there is no difference in the leakage current growth of the PTG and CMOS ripple carry adder above 1.9V. This makes the designer aware of the limitations of varying the voltage supply on the leakage power dissipation, especially when the PTG circuit operates at less than 1.6V.

6.4 XOR/XNOR Design For the Full Adder

The functional unit makes extensive use of XOR/XNOR gates and the data path design previously presented in chapter 4 and 5 mainly uses the PTG circuit shown in Figure:6.4. However, in some parts of the datapath, an even more economically energy efficient gate is used. The new XOR gate described in this section is novel and used in the final stage

of the adder in the Hamming distance and normalization logic. It could also be employed elsewhere in the data path such as the XOR function in the logical unit, adder and compressor circuits. Unfortunately, by the time the new XOR gate was designed, the full custom logical unit and compressors were finished. Therefore, this XOR could be used in any future implementation for a more economical energy efficient version.

6.4.1 New full adder using Novel XOR-XNOR Gates

The three full adder inputs A, B and Carry in (Cin) produce the outputs Sum (S) and Carry out (Cout), where

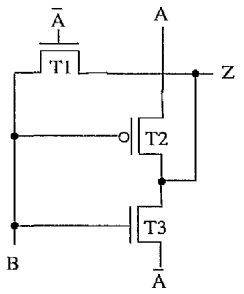
$$S = A \oplus B \oplus Cin$$

$$Cout = [\langle A \oplus B \rangle \cdot Cin] + [\langle not \langle A \oplus B \rangle \rangle \cdot A]$$

Only the XOR-XNOR and multiplexer functions are required to implement a full adder. It is clear from recently proposed new XOR-XNOR gates in[129],[130],[131],[132] and[134] that pass transistor logic is a good candidate for low power and it has a promising high performance due to its low transistor count. However, loading of the adder inputs is required for realistic testing which will enable a meaningful comparison of different structures.

A new XOR gate shown in Figure:6.12 is proposed for the adders design in CADRE-s. This XOR contains only 3 transistors. Similarly, a new XNOR gate is achieved by swapping the A and $\overline{A}$ inputs. In Figure:6.12(a) if A is high, T2 and T3 acts as inverter and $\overline{B}$ is formed at Z whilst if A is low, B passes to Z via T1. This design has the advantage of not requiring the formation of $\overline{B}$. Z will be V_{DD}-V_{tn} when A is low and B is high; for all other input combinations Z is either 0V or V_{DD}. Thus Z may need to be swing restored to V_{DD} depending on the circuit it drives.

The new XOR-XNOR gate can be used to implement the full adder in three different ways as illustrated in Figure:6.16 for designs I, II and III; these designs are based on the full adder logic design with the multiplexer realised with a CMOS transmission gate. The two XOR gates in design I are swing restored. In design II no restoring buffer is placed after the first XOR but as its output signal is only used to drive the select signals of the CMOS



(a) circuit

A	B	Z	T1	T2	T3
0	0	0	ON	OFF	OFF
0	1	1	ON	OFF	OFF
1	0	1	OFF	ON	OFF
1	1	0	OFF	OFF	ON

(b)Truth Table

Figure 6.12: XOR with 3 transistors

transmission gate, the carry output (Cout) still appears as full swing, whilst the XNOR gate with a restoring buffer is used to produce the sum output. In design III, the XOR gate is used with a restoring buffer together with the pass gate full-swing XOR proposed in[130] to produce the sum output. These designs are compared with 4 designs using other logic families. Design IV is a SPL design[135], design V uses PTG gates, design VI uses PTG gates plus output drivers whilst the reference design comprises conventional static CMOS logic. All the full adders I-VI and the reference design are summarized in Figure:6.13. All designs also include inverters to form the inverse of A, B and C as required.

6.4.2 Efficiency Evaluation

To be a fair test, all possible combinations of inputs are applied to the adder cell. Furthermore, since one sequence of inputs, which maximizes the energy consumption for a given cell, could exhibit less energy for another cell[136], six different patterns of input sequences as shown in Table6.3, are used which all generate the same output sequence of Sum and Cout. The average power dissipation and the worst case delay are used to calculate the energy-delay product.

The simulation structure shown in Figure:6.14 is designed to reflect the realistic operating conditions of the full adder being driven by another circuit and also itself driving some load. Buffers of drive 1 strength are used to drive the adder inputs and the output load comprises either no or 8 inverters. Output loading tends to be larger in an asynchronous circuit because an extra load is imposed by driving a delay model or completion detection

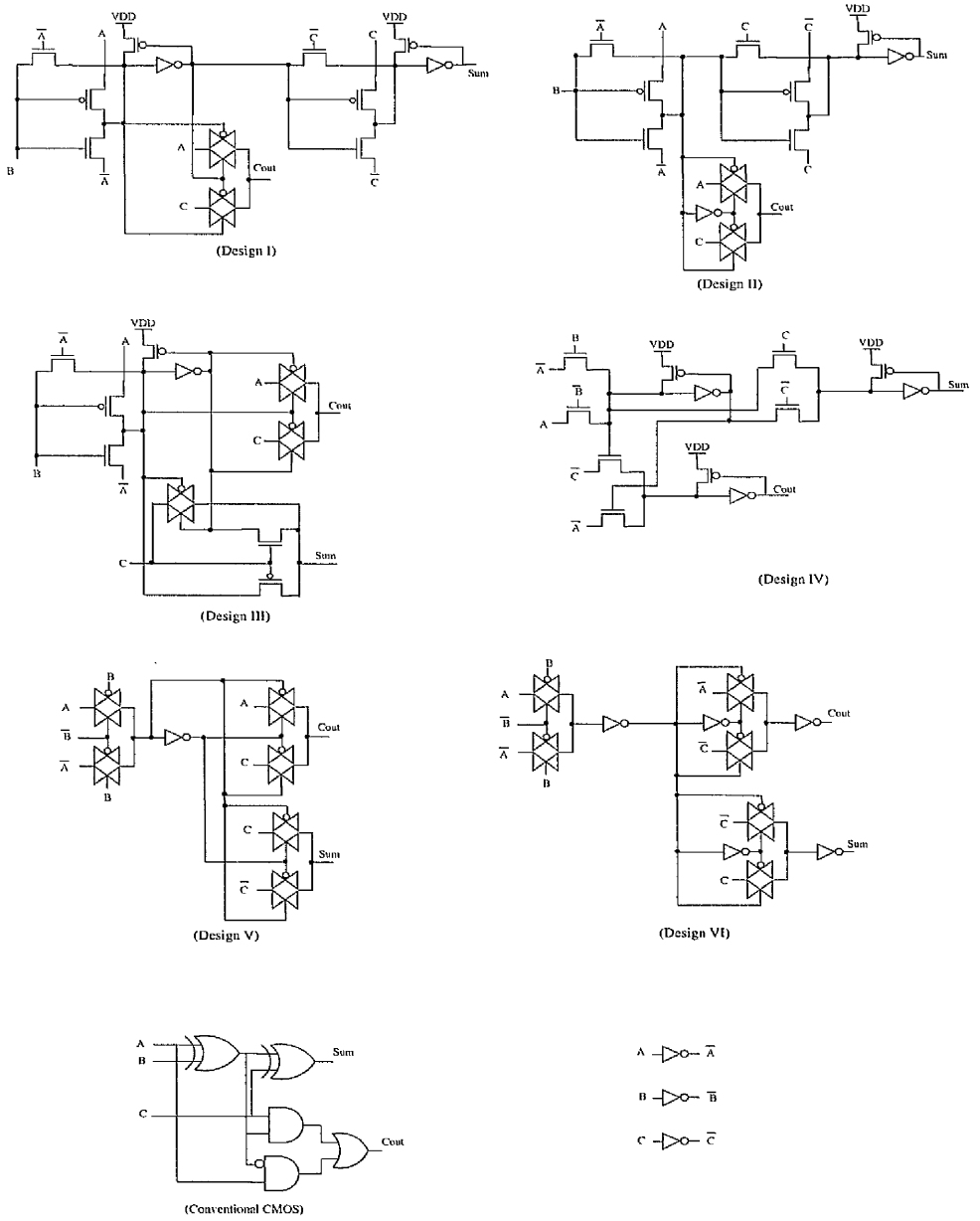


Figure 6.13: New full adders

circuit and this needs to be taken into account. In addition, transistor sizing has been chosen on the basis of minimising energy in each design being evaluated as explained in section 6.2.2. The SPICE simulations target an 0.18um geometry and have been performed under identical environmental conditions.

Table 6.3: Six different input patterns

Pattern1	000	001	010	011	100	101	110	111	000
Pattern 2	000	010	001	011	100	110	101	111	000
Pattern 3	000	100	010	110	001	101	011	111	000
Pattern 4	000	100	001	101	010	110	011	111	000
Pattern 5	000	010	100	110	001	011	101	111	000
Pattern 6	000	001	100	101	010	011	110	111	000

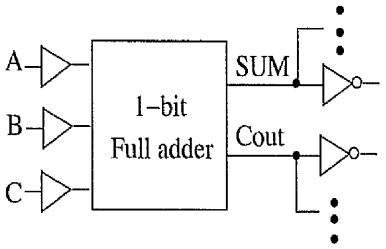


Figure 6.14: Test structure for 1-bit full adder.

All the full adders shown operate correctly at the regular supply voltage of 1.8V and have the full voltage swing at their outputs.

Table 6.4: Simulation results for the full adders at 1.8 V without output load

Design	No. of Transistors	Without output load				
		Power (uW)	Delay (ps)	Energy (pJ)	EDP (pJxps)	NORM % EDP
I	20	56.63	332	0.019	6.24	-21%
II	19	50.24	279	0.014	3.91	-50%
III	16	48.55	336	0.016	5.48	-31%
IV	21	62.17	262	0.016	4.27	-46%
V	20	51.30	193	0.010	1.91	-76%
VI	28	51.48	222	0.011	2.54	-68%
Reference Design	36	71.17	333	0.024	7.90	0

Table 6.5: Simulation results for the full adders at 1.8 V with output load

Design	No of Transistor	With output load (8 inverters)				
		Power (uW)	Delay (ps)	Energy (pJ)	EDP (pJxps)	NORM % EDP
I	20	138.46	456	0.063	28.79	+68%
II	19	100.24	344	0.034	11.86	-31%
III	16	143.06	588	0.084	49.46	+189%
IV	21	139.99	322	0.045	14.51	-15%
V	20	127.80	289	0.037	10.67	-38%
VI	28	98.64	263	0.026	6.82	-60%
Reference Design	36	119.00	379	0.045	17.09	0

The simulation results of the sequence of input patterns running on the test structure without and with an output load are shown in Table6.4 and Table6.5, respectively. The area is approximately proportional to the number of transistors so apart from design VI, a design occupies approximately 55% of the static CMOS adder area. The power consumed is also proportional to the number of devices and when loaded also depends on whether an output driver is used. Performance is dependent upon the number of transistor stages, the drive strength of internal signals and when loaded whether an output buffer is present. Thus design III when loaded has the worst power and performance because there is no drive on the Sum output and passing a low through the output PMOS transistor is relatively slow. The power and performance benefit from incorporating an output driver is illustrated by comparing the loaded results for design V and VI. Design VI is the most energy efficient of all the designs because as well as the output driver it also has strong internal driving due to internal buffering.

Of the three new designs, design II is the best being faster than design I because an inverter has been removed from the sum path. Performance and power are further lowered as the internal capacitance has also been reduced. Design II's power is similar to design VI but its performance is not as good due to its weaker internal drive. Nevertheless design II when loaded has the third best energy efficiency. Design IV uses the pass network and restoring buffer shown in Figure:6.4 and its high short circuit current results in its relatively high power consumption.

Table 6.6: Simulation results for full adder sub-set with 8 inverters as the output load at different supply voltages.

Design		II	V	VI	Reference Design
Number of Transistor		19	20	28	36
V=1.8V	EDP (pJ x ps)	11.86	10.67	6.82	17.09
	Power (uW)	100.24	127.80	98.64	119.00
	Delay (ps)	344	289	263	379
	Comparison	-31%	-38%	-60%	0
V=1.5V	EDP (pJ x ps)	10.72	14.79	7.53	19.03
	Power (uW)	73.08	89.75	73.12	89.95
	Delay (ps)	383	406	321	460
	Comparison	-44%	-22%	-60%	0
V=1.3V	EDP (pJ x ps)	10.99	18.53	8.85	22.69
	Power (uW)	57.02	66.71	58.81	73.13
	Delay (ps)	439	527	388	557
	Comparison	-52%	-18%	-61%	0
V=1.0V	EDP (pJ x ps)	18.32	39.98	14.77	38.04
	Power (uW)	38.14	42.32	40.90	51.91
	Delay (ps)	693	972	601	856
	Comparison	-52%	5%	-61%	0

The lowest possible voltage consistent with the desired performance is normally applied because the power consumption is proportional to the square of the supply voltage. Therefore, three lower levels of voltage (1.5, 1.3 and 1.0 V) below the regular supply voltage (1.8 V) have also been evaluated for their energy-delay product for the three best adders (design II, design V and design VI) and for the static CMOS full adder. This analysis has used the same input patterns. The simulation results in Table6.6, show that full adder design VI is the most promising approach for energy efficiency at all the different voltages. Whilst full adder design V has a better energy saving than the new proposed adder design II at the regular voltage supply, design II is better when the voltage is reduced. Furthermore, design II dissipates less power than design VI as the supply is lowered. This is because the new XOR-XNOR gate is designed to pass inputs strongly by

one of the NMOS transistors whilst a '1' is passed strongly by the PMOS device for the AB="10" combination.

Table 6.7: Simulation results for full adder sub-set with output load (8 inverters) with the 6 patterns of the sequence inputs

Design (V=1.8V)		II	V	VI	Reference Design
Pattern1	EDP (pJ x ps)	11.86	10.67	6.82	17.09
	Power (uW)	100.24	127.80	98.64	119.00
	Comparison	-31%	-38%	-60%	0
Pattern2	EDP (pJ x ps)	7.47	6.63	6.35	16.26
	Power (uW)	109.62	99.56	92.50	123.39
	Comparison	-54%	-59%	-61%	0
Pattern3	EDP (pJ x ps)	16.69	9.29	9.81	19.47
	Power (uW)	98.32	118.51	104.78	125.41
	Comparison	-14%	-52%	-50%	0
Pattern4	EDP (pJ x ps)	5.95	15.22	3.86	7.63
	Power (uW)	111.40	124.25	105.86	124.07
	Comparison	-22%	+99%	-49%	0
Pattern5	EDP (pJ x ps)	14.60	8.09	8.65	19.03
	Power (uW)	86.00	103.14	93.01	122.00
	Comparison	-23%	-58%	-55%	0
Pattern6	EDP (pJ x ps)	11.97	14.98	7.05	20.16
	Power (uW)	101.16	122.27	101.97	140.36
	Comparison	-41%	-26%	-65%	0

The simulation results from Table6.7 show the significant variation in results obtained according to the input sequence applied and it is possible that some systems using DSPs may be able to organize their adder inputs to exploit this feature. Table6.7 shows that for most input sequence patterns design VI is the most energy efficient full adder although it occupies approximately 50% more area than design II. In some patterns, design II dissipates less power than design VI.

The full adder design II using the new XOR-XNOR is an alternative choice for designs where both power and area are the critical issues. Design V is a poor choice because it has

the worst energy consumption for some cases of the input patterns. Both design II and design VI offer significant energy savings over the static CMOS design.

For robust systems, circuits need to generate the full voltage swing at the outputs. A number of full adder designs have been presented and compared. The best energy efficient 1-bit full adder for practical design is based on pass transmission gates with output drive buffers. The results also demonstrate that using CMOS transmission gates without the output driver cannot achieve energy efficiency. The new full adder (design II) implemented using the novel XOR-XNOR gates with restoring buffers yields the best circuit for low power when the voltage supply is scaled down, making it suitable for use with a voltage scaling technique. In addition, the new full adder is the best choice for applications where area is a critical issue.

The new full adder design II has been used in the final addition of the Hamming distance and normalization unit because it dissipates low power. Additionally, a significant advantage of using the new adder design is that the datapath becomes more compact. The new XOR/XNOR could also have been used in other arithmetic functions such as the 4-2 compressor for a more economical area and power budget; this is discussed in the next section.

6.5 Energy Efficient 4-2 compressor

A conventional 4-2 compressor has typically five inputs (P1 to P4 and Cin) and three outputs (PC, PS and Cout). The four inputs (P1 to P4) and the output PS have the same weight (2^0). The output PC is weighted one binary bit higher (2^1). The final input, Cin, is from the preceding compressor module of one bit lesser significance. The carry output from the compressor, Cout (weight 2^1), is passed to the compressor of one bit higher significance. The layout can be very compact as the logic corresponds to the data flow with the data inputs and outputs P1 to P4, PS and PC, flowing vertically and the input and output carry Cin and Cout flowing horizontally.

6.5.1 4-2 Compressor Designs

Conventional 4-2 compressors can be implemented with two stages of full adders connected in series. There are many compressor designs. For example, Nagamatsu[137] introduced a logic circuit optimization to shorten the resulting critical path. The 4-2 compressor implemented in pseudo-CMOS logic was given in [138]. However, it increased the switching activity because of the large number of inverters used in the design. Many 4-2 compressor circuits use the equations 4.6, for example[139], [140]. The block diagram of 4-2 compressor architecture is shown in Figure:6.15.

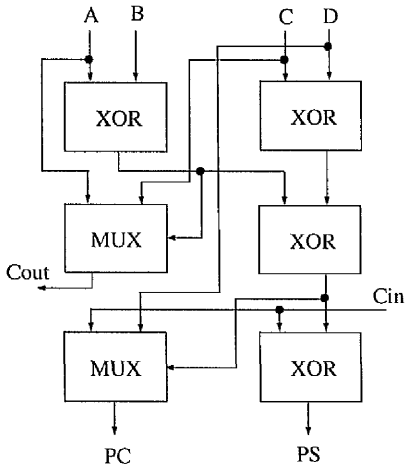


Figure 6.15: Architecture of the 4-2 compressor

6.5.2 4-2 Compressor Circuitry

There are several designs available to implement the 4-2 compressor. A multiplexer based implementation of a 4-2 compressor using pass transmission gates was given in [143]. A double pass transistor logic (DPL) based on compression technique[144] was first employed in AMULET3i[145] and this exhibits lower power consumption and higher speed compared to earlier designs. But this increases the overall transistor count to 58 transistors.

The new 4-2 compressor used in CADRE-s uses the new three transistor XOR-XNOR gate. It is shown in Figure:6.16 and minimizes the number of transistors. Due to the new

XOR-XNOR gate using only three transistors and not needing to generate inverse outputs (such as required in the DPL circuit), the total number of transistors used in the CADRE-s compressor, shown in Figure:6.16, is only 29 transistors against the 58 reported for DPL. Circuit simulation shows that the energy consumption per operation of the DPL 4-2 compressor is about 0.11pJ whilst the new 4-2 compressor consumes only 0.05 pJ which is better than the DPL by about a factor of two. Since the design in Figure:6.16 was proposed after most of the data path was completed, it became impractical to include it in the final design. However, the proposed design should be considered for inclusion in any subsequent version of the CADRE-s design. It would also be suitable for inclusion in other arithmetic unit designs.

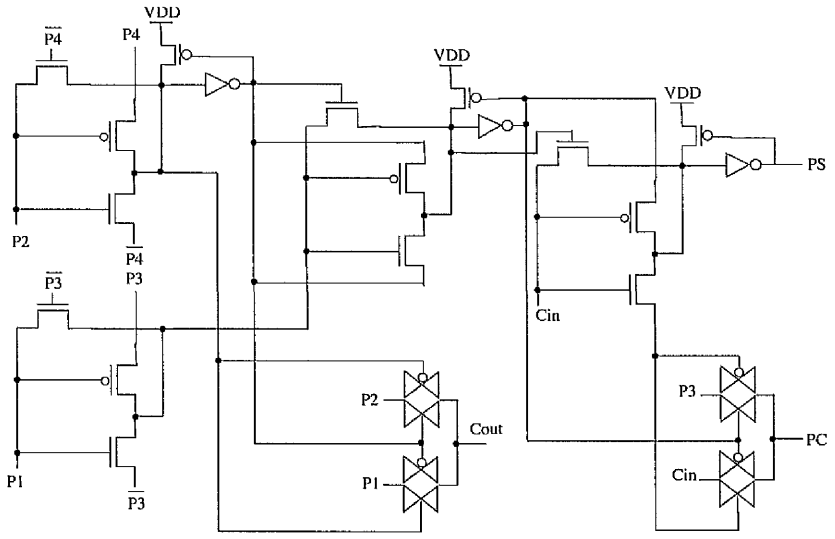


Figure 6.16: A proposed 4-2 compressor

6.6 Energy Efficient Latch

The pass transmission gate latch shown in Figure:6.17 is used for preventing unnecessary transitions from passing through to the next unit in the data path. When En is high and $\overline{\text{En}}$ is low, the NMOS and PMOS transistors of the pass transmission gate are on and the latch is loaded. However, when En is low and $\overline{\text{En}}$ is high, the PTG is off. The only significant problem is about charge sharing at node X between the PTG and inverter. A very weak inverter is therefore placed in the feedback path to statically maintain the output state.

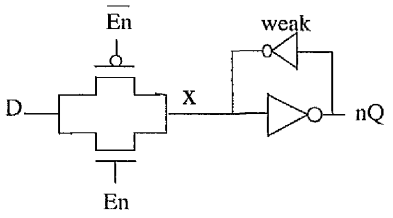


Figure 6.17: A pass transmission gate latch

6.7 Layout Consideration

6.7.1 Datapath Layout

Both the full custom datapath and the standard cells used for the control and scan paths have been carefully optimized for both performance and area efficiency. This improves the circuit energy efficiency. Efficient layout is a significant goal for any VLSI design, whether it aims at low power or high performance. Both the datapath and standard cells layouts for energy efficient systems have the same goal of minimizing the layout area. In addition, commercial cell libraries only typically provide strength 1 and 2 output drives. Therefore, full-custom design gives greater flexibility and energy efficiency compared to semi-custom design. Moreover pass transistor design implemented by adopting a semi-custom style cannot gain the advantage of optimal area because pass transistor cells then need to conform strictly to the same design rules and area as other conventional CMOS cells. Table6.8 shows experimental results from a standard cell and full custom XOR circuit.

In term of energy efficiency, it is clear that comparing the layout results, implementing the circuits using full custom gives a significant energy saving compared with using standard cell; the advantage of using full custom is not really apparent if just a schematic comparison is performed. For this reason, the CADRE-s data path has been implemented with the full custom style.

Table 6.8: Energy saving of PTG XOR: standard cell versus full custom

	Standard cell		Full custom		% Saving
	Power (uW)	Delay (ps)	Power (uW)	Delay (ps)	
Schematic	12.204	40.0	11.736	40.0	
Layout	16.488	47.4	13.50	43.0	
Energy(Schematic)($x10^{-18}$J)		489		470	3.73%
Energy(Layout)($x10^{-18}$J)		782		581	25.68%
Et(Schematic)($x10^{-30}$Js) (energy - delay product)		19526		18797	3.73%
Et(Layout)($x10^{-30}$Js)		37060		24984	32.58%

The pitch of the datapath is set when the entire schematic of 1-bit of the FU datapath is completed. Of the full custom cells designed for the data path, the 4-2 compressor circuit is the biggest cell in the FU datapath. From this, the maximum datapath width of 1 cell is able to be calculated (7.28um). Following this, the FU datapath has been partitioned to minimise the length and number of buses across the datapath. In Figure:6.18, the floor-plan of the CADRE-s datapath layout shows each stripe of the functional unit datapath which gives an optimal bus arrangement and area where m1,m2,m4 and m5 are multiplexer stages. The layout reflects the flow of data to minimise the length of wiring required.

The 4-2 compressors in the multiplier, adder, and hamming distance and normalization logic have been laid out in a line so as to minimise their area and interconnection lengths (at the expense of greater design and layout effort). Cells abut where possible and the key to butting up cells is to match the pitch of the cell's input and output signals. Cells connected together in a vertical slice must have the output pitch lined up with the input pitch at the cell edge. Figure:6.19 shows the line-up for 4-2 compressor cells. Reducing the vertical connection of a 4-2 compressor row makes the layout smaller, lowers power and increases performance. This line-up technique is also applied in the carry-look-ahead tree and to every component that has a vertical connection in the CADRE-s datapath. The cells are designed with power and ground rails on the far left and right of the cell as shown

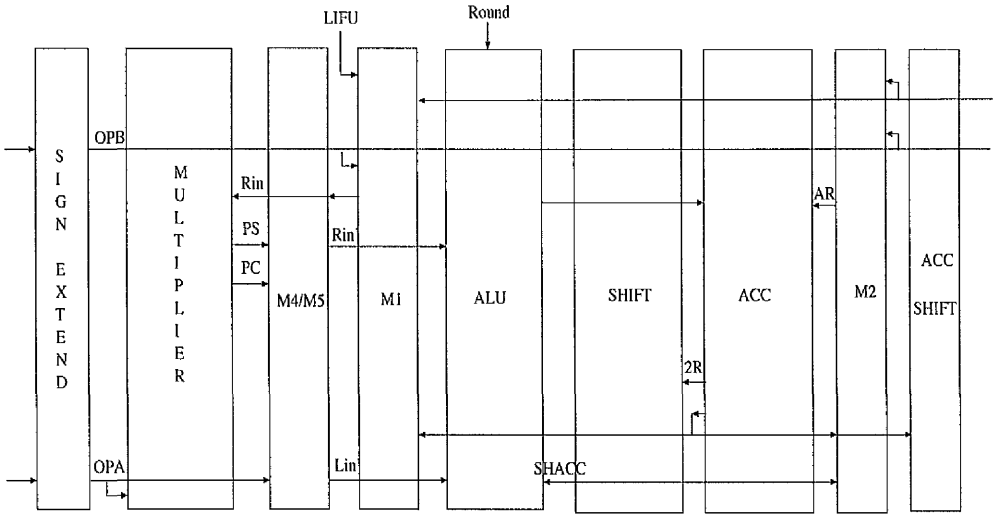


Figure 6.18: Functional unit datapath floor-plan

in Figure:6.20; this enables the cell to connect directly to other datapath cells in the next slice on either side of it if routing through the cell is not required.

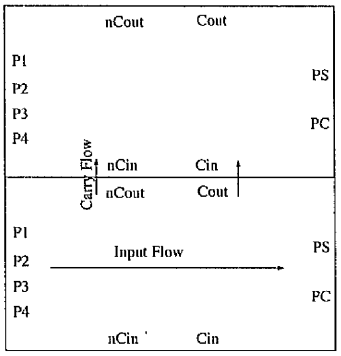


Figure 6.19: A 4-2 compressor layout in the multiplier showing cell abutment

In the process used, six metal layers are available with Metal1 nearest the substrate and Metal6 the furthest away. Metal1 has been used for local connections. Metal2 is utilized for power (VDD) and ground (GND) and its line widths are dictated by the maximum current a cell can draw. Metal2 and Metal4 are utilized for control lines whilst Metal3 and

Metal5 are used for local and global data buses. The summary of metal use is shown in Table6.9.

Table 6.9: Metal usage summation for full custom datapath

Metal	Usage
1	local connections inside cells
2	power, ground, control signals
3	local data buses
4	control signals
5	global data buses
6	power, ground

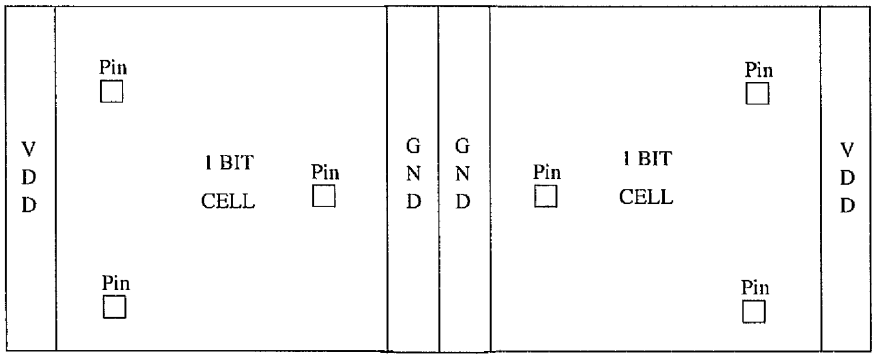


Figure 6.20: Layout format for 1-bit cell in full custom functional unit datapath

6.7.2 Standard cell layout

An in-house Standard Cell Library named Amust, has been used in the control unit. It contains basic static CMOS gates (AND, OR, NAND, NOR, etc.) and Muller C-gate elements; Muller-C gates are required for the asynchronous handshake control. The layout of each cell in this Amust library has been laid out by hand to have a minimal overall area compared to synthesized layout. The pitch for the standard cell control was set to 6.4um. There are four different versions of each logic gate capable of driving different loads (drive1, drive2, drive3 and drive4). In Figure:6.21, a summary of the

layout rules for the Amust library is shown for two adjacent cells in a slice. The power and ground rails are 8λ in Metal1. Metal2 is avoided as much as possible in standard cells to allow Metal2 to be used for routing over the cells. Meanwhile, all pins have to be placed in the middle in Metal1 on a 6.4um routing pitch.

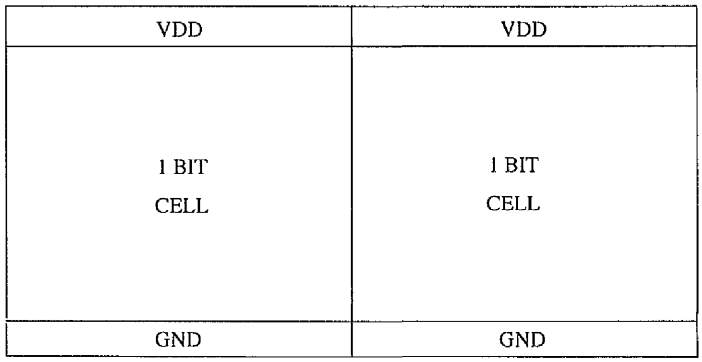


Figure 6.21: Layout format for standard cell library (Amust)

Interconnect Resistance

For the 180nm process, the upper layers of metal have less resistivity due to being thicker. Hence, a high switching transition bus is recommended to use the upper metal layers such as Metal4, Metal5 or Metal6. On the other hand, contacts and vias also have a resistance associated with them that is dependent on the contacted materials and size of the contact. Multiple contacts can form low-resistance connections. When current turns at the corner, then a square array of contacts is generally implemented.

Layout for Dual Supply Voltage

The incorporation of the tunable delay mechanism requires the use of two different supply voltages. A conventional dual-supply layout style uses two n-wells (one for each supply) in order to isolate the n-wells connected to different supplies from one another. By doing this, an area penalty is incurred corresponding to the minimum spacing between two cells operating at different voltages. Therefore, sharing the n-well for the two supplies is introduced in the full-custom database to reduce the area overhead. This shared n-well

layout is connected to the regular supply and employs an extra power rail for the source voltage in a higher metal layer, whilst the power rail of the substrate voltage runs underneath using a lower metal layer. The power line to the tunable delay does not need to be very wide because the current required is relatively small.

In this chapter, energy efficient circuits implemented by pass gate logic, and layout considerations for energy efficiency have been presented. This completes the description of the CADRE-s design and its implementation. The next chapter presents the results of running programs on the design.

Chapter 7: Evaluation

7.1 Chip Connections

CADRE-s is implemented on a 0.18μm CMOS process having 6-metal layers and runs from 1.8V. Figure:7.1 shows the final chip which is in the final stages of preparation for fabrication. Thus results presented in this chapter are the result of simulating the layout following the layout versus schematic (LVS) checks. The tests described seek to demonstrate the energy efficiency of the FU designed by the coherent low energy design techniques described in the thesis. The kernel benchmarks are translated into binary codes by the CADRE assembler developed by Mike Lewis[72]. A Verilog module then re-arranges this binary code into the format for serially shifting into CADRE-s. The CADRE-s die, shown in Figure:7.1, measures 4.5 x 3.9mm and contains over 230,000 transistors plus 14 RAM memories of 2k x 16 bits plus a further 4 RAM memories of 64 x 32 bits.

The CADRE-s die is to be placed into a 68-pin Pin Grid Array (PGA) package. There are 25 signal pins with 9 pins required for the DSP input/output and 16 required for the scan chain pins. Thus, there are no large data buses into or out of the chip. In addition to the signal pins, there are 13 power pins and 24 ground pins. The power pins are further subdivided into 1 pin for the tunable timing logic (vddtune), 4 for the FU voltage (one per FU) (vddfu), 4 pins for the 1.8V input/output pads and RAMs, and 4 pins for the 3.3V input/output pads. The complete chip pad pins are given in Appendix A. There are 4 additional power pins and 3 signal pins for test circuits external to the FU design.

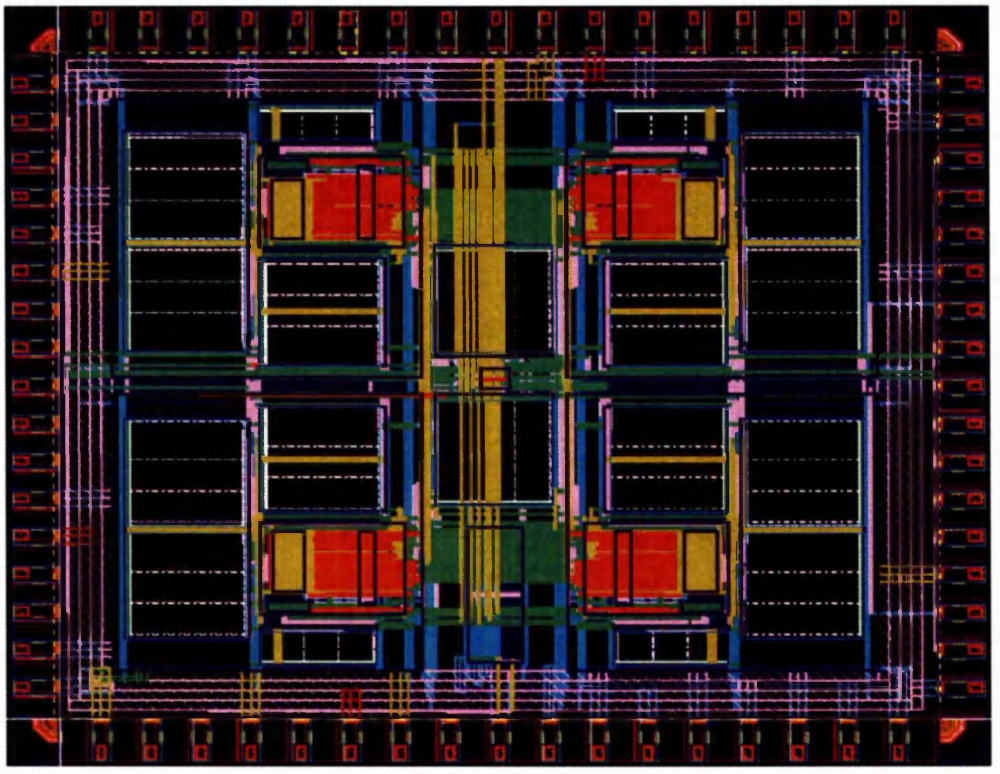


Figure 7.1: CADRE-s Die Photo

7.2 Kernel Benchmarks

This section outlines the four kernel benchmarks used to evaluate CADRE-s.

7.2.1 Digital filters

DSPs have become very popular because of their excellent performance when performing as digital filters. Normally, filters are used for signal separation and restoration. For example, if a signal suffers from interference or noise, then a filter can be used to separate the signal from the noise. Similarly, if a signal has been distorted, a filter can be employed to obtain a better signal. A filter is able to be either analog or digital. However, a digital filter is better than an analog filter in terms of performance.

The well-known and widely used method of filtering is performed by convolution. The equation for the Finite Impulse Response (FIR) filter is shown in Eq.7.1. A FIR with 256 data samples and 20 coefficients has been used to evaluate the CADRE-s system.

$$y(n) = \sum_{k=0}^{M-1} C_k \cdot X_{(n-k)} \qquad \text{..... Eq.7.1}$$

7.2.2 Vector Dot Product

The dot product operation multiplies two one dimensional vectors and sums the products. It is useful for matrix calculations and is also common in filtering operations. The operation performed is given in Eq 7.2.

$$Y = \sum_{n=1}^{N} a(n) \cdot X(n) \qquad \text{..Eq.7.2}$$

where a(n) and X(n) are the vector elements of length N. CADRE-s has evaluated this function for N = 100.

7.2.3 Correlation

The correlation function shows the similarity of two signals and finds out how long they remain similar when one is shifted with respect to the other. Correlation is a widely used computational operation in digital communications and other systems processing random signals. The correlation of 2 sequences a(n) and x(n) is defined in [81] as

$$y(n) = \sum_{k=-\infty}^{\infty} a(k) \cdot x(n+k) \qquad \text{.... Eq.7.3}$$

If a(n) and x(n) have finite length N, the digital correlation operation is given as follows:

$$y(n) = \sum_{k=0}^{N-1} a(k) \cdot x(n+k) \qquad \text{... Eq.7.4}$$

In the evaluation performed on CADRE-s, N = 4 have been used for running the correlation kernel. The operation in Eq.7.4 can be written in matrix-vector multiplication form as

$$\begin{bmatrix} y(-3) \\ y(-2) \\ y(-1) \\ y(0) \\ y(1) \\ y(2) \\ y(3) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & x(0) \\ 0 & 0 & x(0) & x(1) \\ 0 & x(0) & x(1) & x(2) \\ x(0) & x(1) & x(2) & x(3) \\ x(1) & x(2) & x(3) & 0 \\ x(2) & x(3) & 0 & 0 \\ x(3) & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a(0) \\ a(1) \\ a(2) \\ a(3) \end{bmatrix}$$

for N=4.

7.2.4 Discrete Cosine Transform (DCT)

DCT is a mathematical operation that transforms a set of data to its frequency components. The number of samples should be finite and a power of two for optimal computation time. A one-dimensional DCT is used to convert an array of numbers (signal amplitudes at various points in time) into another array of numbers. The first element in the result array is a simple average of all the samples in the input array and is referred to as the DC coefficient. The remaining elements in the result array each indicate the amplitude of a specific frequency component of the input array and are known as AC coefficients. The frequency content of the sample set at each frequency is calculated by taking a weighted average of the entire set. The weight coefficients are those of a cosine

wave, whose frequency is proportional to the result array index. The procedure of transforming the spatial domain of pixels to the DCT domain is given in Figure:7.2.

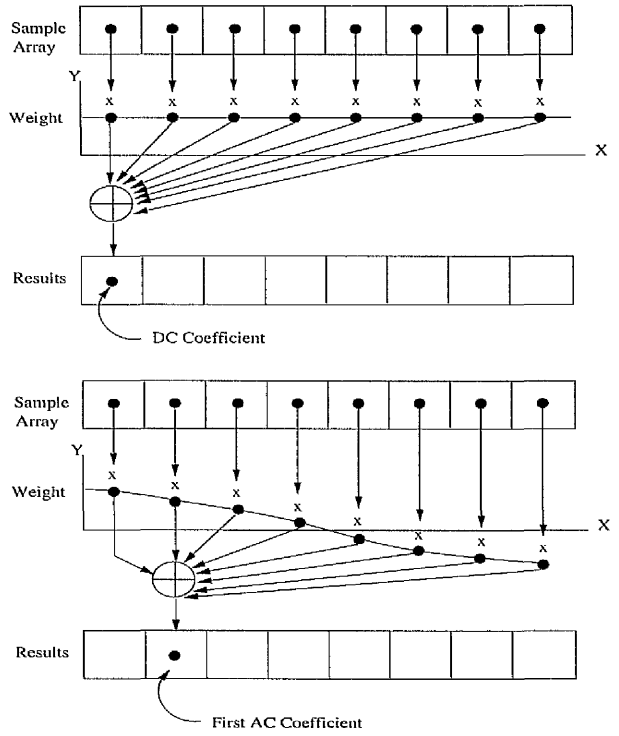


Figure 7.2: Sample of transforming spatial domain of pixels to DCT domain

The two-dimensional DCT uses the same fundamental principle as the 1-D DCT. It assumes an 8x8 array of pixels i.e. eight rows of eight pixels. So a 1-D DCT is applied separately to each row of eight pixels to produce eight rows of frequency coefficients. These eight coefficients are then taken as eight columns with the first column containing the DC coefficient, the second containing the fundamental frequency f, the third column coefficients of 2f etc. More details about the DCT can be found in [81],[154].

7.3 Speech / Voice Data

As mentioned before, the functional unit design is aimed at mobile applications. It is also expected to run multimedia applications such as 3-D gaming, enhanced graphics, video and digital audio features on 3G mobile hand-held devices (smart phones and PDAs). In

this section, the speech/voice data employed in running the kernel benchmarks is described.

The main speech coding techniques which are used today can be divided into three classes - waveform codecs, source voice codecs and hybrid codecs. The waveform codecs are used at high bit rates and give very good quality speech. In contrast, the source codecs operate at very low bit rates and tend to produce speech which sounds synthetic. Finally, hybrid codecs use techniques from both source and waveform coding and give good quality speech with intermediate bit rates.

The evaluation on CADRE-s uses waveform codecs to generate the test data. The waveform codecs are low complexity codecs which produce high quality speech at rates above about 16 kbps. The simplest form of waveform coding is Pulse Codec Modulation (PCM), which involves sampling and quantizing the input waveform. Narrow-band speech is typically band-limited to 4kHz and sampled at 8kHz. If linear quantization is used then 12 bits per sample (bit rate = 96 kbps) is needed to produce good quality speech. A logarithmic quantizer is often used in speech coding and gives a signal to noise ratio which is almost constant over a wide range of input levels. At a rate of 8 bits per sample or 64 kbps, it gives a signal which is almost indistinguishable from the original.

Another type of waveform codec uses Adaptive Differential Pulse Code Modulation (ADPCM) which quantizes the difference between the speech signal and a prediction that has been made of the speech signal. If the prediction is accurate then the difference between the real and predicted speech samples will have a lower variance then the real speech samples, and will be accurately quantized with fewer bits than would be needed to quantize the original speech samples. At the decoder the quantized difference signal is added to the predicted signal to give the reconstructed speech signal. The performance of the codec is aided by using adaptive prediction and quantization, so that the predictor and difference quantizer adapt to the changing characteristics of the speech being coded. In the mid 1980s, the CCITT standardised[141] a 32 kbits/s ADPCM, known as G721, which gave reconstructed speech almost as good as the 64 kbits/s PCM codecs. Later G726 and G727 codecs operating at 40, 32, 24 and 16 kbits/s were standardised.

7.4 Results

7.4.1 Operation Rate

The kernel benchmarks are supplemented by a validation test program written by the author. This tests every instruction of CADRE-s for correct functionality repeatedly testing each instruction 50 or 100 times with random data. The kernel benchmarks used comprised a FIR filter (called FIR-20v1) with 20 coefficients and 256 sampling points mapped onto 4FUs to minimise data fetches, a FIR filter called FIR-20v2 similar to FIR-20v1 but minimising the number of timing slots, a 2 dimensional DCT where n=8 (the code, data and coefficients are contained in [151]), a dot product program with N=100 and a correlation function with N=4. Note that the data and coefficients of FIR-20 have been provided from Motorola's DSP56K benchmark, otherwise the random numbers have been used as input data.

Table7.1 shows the average instruction rate, the overall rate of performing arithmetic operations and the occupancy in the 4 FUs during program execution. The results are slightly over-optimistic because the time to set up the configuration memory prior to running the programs is ignored. However, since the number of configuration memory lines in general is relatively small compared with the number of operations performed in the programs, the results in Table7.1 are indicative of the performance to be expected from the system when running.

From Table 7.1, CADRE-s can perform at over 190MOPS which more than meets the expected target at 160MOPS higher than the 3G mobile phones of 100MOPs. Furthermore, Table7.1 shows, as expected from the algorithms, the parallelism of the system can be fully exploited and the occupancy figures indicate that a very high level of concurrency is achieved. The modest and variable instruction rate of 50-66MHz is indicative of simpler units working at their natural rate rather than being coerced to conforming to a clock. The ability of the design to adapt to the calculations being performed is of course one of the advantages of asynchronous design.

Table 7.1: Parallel instruction rates and operations per second

Benchmarks	Instruction rate	Arithmetic operation rate	Parallelism Achieved
Validation	66MHz	264MOPS	100%
FIR-20v1	50MHz	192MOPS	96%
FIR-20v2	50MHz	190MOPS	95%
2-D DCT 8x8	59MHz	236MOPS	100%
Dot Product (N=100)	50MHz	200MOPS	100%
Correlation (N=4)	50MHz	200MOPS	100%

7.4.2 Run Time

The post-layout netlist of CADRE-s in SPICE format is extracted by the DIVA[149] software which is part of the Cadence EDA tool flow; this takes the parasitic capacitance into account. Following this, the CADRE-s netlist is simulated using the Verilog co-simulator (VCS_nanosim)[150]. In order to get accurate results, similar to those of fabricated silicon, the Verilog timing models with necessary process parameters such as the load capacitance of the RAMs and the pad input/outputs are also included in the simulation. All binary codes of the test programs are generated by a Verilog test module.

Table 7.2: Power and energy consumption of CADRE-s operated at 1.8V

Benchmarks	Avg. Power consump-tion (mW)	run time (us)	Energy (uJ)	No. of instructions	Avg. energy (nJ)/ instruction
Validation	21.18	13.14	0.278	80	3.48
FIR-20v1	23.42	1,316	30.82	6,400	4.81
FIR-20v2	25.47	1,107	28.19	5,376	5.24
2-D DCT 8x8	22.67	170.25	3.86	832	4.64
Dot Product (N=100)	24.04	22.13	0.53	108	4.92
Correlation (N=4)	23.97	8.2	0.19	40	4.91

Note that CADRE-s uses the manufacturer's RAMs[142] and only a top level timing model in Verilog for these is provided; no internal detail is included in the RAM model. Therefore the average power consumption of the RAM based on the specification on the data sheet has been analysed. According to this, the power consumption per megahertz (mW/MHz) for reading and writing is approximately the same at 0.24mW/MHz; this gives a power consumption per operation of 12mW@50MHz. This figure is used in subsequent power consumption analyses.

The average power consumption and run time shown in Table 7.2 are measured on the CADRE-s system. The run time includes the FU execution time and the time to download the data and instructions. It should be noted that the energy calculations are based on the metric of energy for the burst throughput mode which is $(E_{max} + E_{idle})/T_{max}$ as stated in chapter2.

The download period depends on the size of programs and uses a clock frequency of 167MHz for shifting data serially. The longest scan path is 113-bit found in the functional unit consisting of address, data and control signals for two operand 2Kx16-bit RAMs, one 64x32-bit configuration memory and one 2Kx16-bit result RAM. So even with a download frequency of 167MHz, the download time dominates the run time. The breakdown between the download and execution run time is given in Table 7.3. This enables the actual energy efficiency of the FUs to be analysed.

7.4.3 Execution Energy of FIR Filter

The overall power and run time figures can be broken down into those during execution and those during download. This is shown in Table 7.3. The figures have been obtained from the trace file generated by the Nanosim simulation. As discussed in chapter3 about the trade-off between minimizing switching activity with the number of time slots and configuration memory instructions in the FIR filter, the results in Table 7.3 shows that the energy consumption of the FIR filter with a smaller number of time slots and configuration memory instruction (FIR-20v2) is less than the FIR filter with a data

arrangement for minimizing the switching activity by about 7%; this is because of its shorter run time.

Table 7.3: Break down average power, download and execution time of the CADRE-s

Benchmarks	Average power consumption (mW)		run time (us)		Energy (uJ)	
	Execution	Download	Execution	Download	Execution	Download
Validation	18.49	23.88	0.25	13.00	0.004	0.310
FIR-20v1	22.97	23.88	30.4	1,285.63	0.698	30.701
FIR-20v2	24.06	23.88	26.9	1,080.70	0.647	25.807
2-D DCT 8x8	21.47	23.88	3.52	166.73	0.075	3.981
Dot Product (N=100)	24.21	23.88	0.54	21.59	0.013	0.515
Correlation (N=4)	24.06	23.88	0.19	8.11	0.004	0.194

As can be seen in Figure:7.3, which shows the difference between the execution and download energy, CADRE-s dissipates more than 96% of its overall energy consumption in downloading the data and instructions. It is only to be expected that the top level framework surrounding the CADRE-s FUs will dominate the power as it contains 14 2Kx16 bit RAMs. In a complete system comprising a register file, instruction buffer etc., the RAM power will drop dramatically; for example, in the original CADRE, on-chip RAM was estimated to consume only 3% of the overall power. For this reason in further analysis of CADRE-s, the download time will be ignored and attention concentrated only on the execution time and power of the functional units.

Looking at the execution energy of the FIR-20v1 on the CADRE-s FUs, it is only 0.7μJ. To put this result into context, the original CADRE was designed on a 0.35μm process running from 3.3V. Furthermore the original design was only simulated at the schematic level and realistically the energy will (at least) double in going down to layout. Taking all these factors into account and scaling the original CADRE results to 1.8V and 0.18μm, the FUs in CADRE-s result in less energy than in the original CADRE by a factor of 4.6 for the FIR-20v1 program.

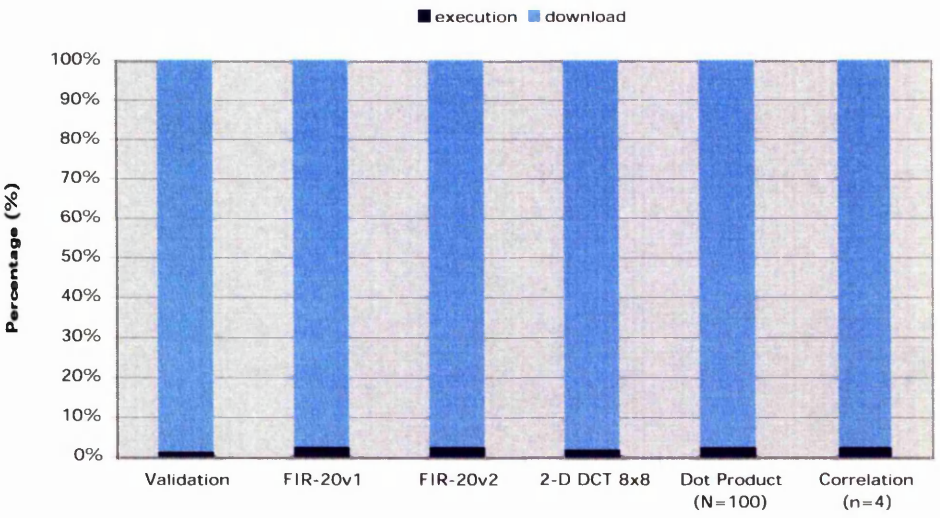


Figure 7.3: The proportion of execution and download energy in CADRE-s

7.4.4 Instruction Mix and Execution Time

The relative power consumption for each instruction type has been extracted for the programs run and is shown in Figure:7.4. The shifter operation consumes the smallest power dissipation (=1) compared to others. From the pie chart in Figure:7.4, multiplication and multiply accumulate instructions dissipate five times as much power as a shift instruction. Therefore, this suggests that multiplying by a power of two should be performed by a shift operation instead. Addition consumes half the power of multiplication. So using the addition instead of multiplication needs to be considered carefully. The other instructions such as MAX, MIN, ABSMAX and ABSMIN consume approximately the same power as addition because these operations are based on addition. The Hamming distance and normalization instructions dissipate slightly less power than addition. Thus, re-arranging instructions carefully can minimize the energy consumption in the system.

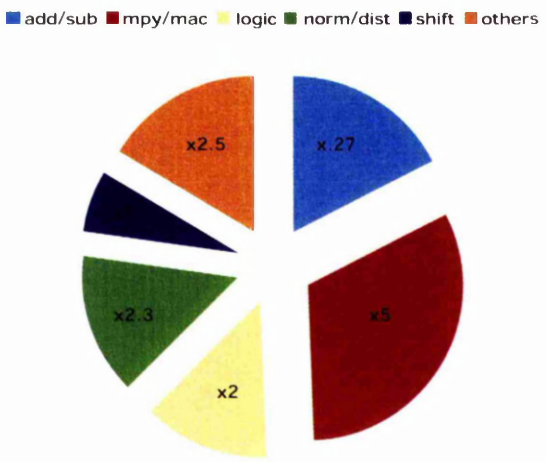


Figure 7.4: The ratio of energy per instruction compared to shift instruction

Table7.4 gives a summary of the instruction breakdown for each benchmark program. It

Table 7.4: Instruction breakdown for the benchmark programs

Instruction	Validation	FIR-20v1	FIR-20v2	2-D DCT	Dot Prod-uct	Correla-tion
ADD	10%	12%	0%	34.6%	0%	0%
MPY	5%	16%	4.7%	19.2%	3.7%	40%
MAC	5%	64%	90%	15.3%	96.3%	60%
SHIFT	5%	0%	0%	0%	0%	0%
Others	75%	8%	5.3%	30.9%	0%	0%

shows that in general the CADRE-s FUs spend most of their time multiplying or adding. For typical algorithms this is at least 70% of the time and for many algorithms it is over 90%. Thus the time spent in fully optimising the adder and multiplier designs is more than justified.

7.4.5 Scaling Down the Supply

Three DSP kernels, the FIR-20v2, the two dimensional DCT 8x8 and correlation are used in the simulation to find the energy saving when CADRE-s operates from a 1.6V supply. The results in Table 7.5 show the energy saving from running FIR-20v2, the 2-D DCT and correlation to be about 24%, 22% and 22%, respectively compared with running at 1.8V. It is clear that CADRE-s using a voltage scaling technique can gain an energy saving. This implies that the voltage should be reduced to provide just the required throughput whenever possible.

Table 7.5: Energy consumption during the execution time of CADRE-s a 1.6V power supply

Bench-marks	Average Power Consumption (mW)	Execution Time (us)	Energy (uJ)	Energy Saving compared to V=1.8V
FIR-20v2	18.51	28.24	0.523	~ 24 %
2-D DCT 8x8	16.52	3.78	0.062	~22 %
Correlation	18.50	0.20	0.0037	~22 %

7.4.6 Power Breakdown in the Functional Units

The average power consumption during the execution time can be broken down into the power to supply operands from the scan path, the power dissipated in the FUs and the power taken by the tunable delay circuits. The results of this breakdown are shown in Table 7.6. From Table 7.6, the average power consumption of the FUs in FIR-20v2 is higher than FIR-20v1 even though the total energy consumption of FIR-20v2 is less than FIR20-v1 as shown in Table 7.3. This is because although the FIR-20v1 has less switching activity than FIR-20v2, FIR-20v2 uses less execution time to complete. This makes FIR-20v2 consume less energy than FIR-20v1.

The power consumption for the tunable delay of the asynchronous design is very small at < 6% of the overall power consumption in the FUs. Considering that the clock generator can dissipate about 30% of the overall system power in synchronous design, it can be concluded that the overhead of the delay model in this asynchronous design is relatively small and thus asynchronous design can achieve an energy saving of about 24%. As mentioned in chapter3, CADRE-s employs scan path circuits to load data and instructions

into the system. The pipelined latches are also attached to the scan path output. These features consume power of about 10-20% of the overall power consumption during the execution time.

Table 7.6: The average power consumption during the execution of the FU in CADRE-s

Benchmarks	Average power consumption in FUs (mW)	Average power consumption in tunable (mW)	Average power consumption from scan path&pipe-lined latches (mW)	Average power consumption in total (mW)
Validation	14.27	0.55	3.67	18.49
FIR-20v1	18.95	0.72	3.29	22.97
FIR-20v2	19.67	1.04	3.35	24.06
2-D DCT 8x8	16.77	0.69	4.01	21.47
Dot Product (N=100)	19.96	1.08	3.17	24.21
Correlation (N=4)	19.67	1.03	3.35	24.06

Figure:7.5 shows a breakdown of the power consumption of the arithmetic unit block (MAC), the tunable mechanism, and the control and decoder circuits. The MAC unit and tunable mechanism circuits have been implemented by full-custom design which consumes about 75% of the power. The control and decoder circuits are implemented in standard cell and have a consumption around 1/3 of that of the MAC unit, although the number of transistors in the control and decoder circuit is less than those in the MAC unit and tunable mechanism by a factor of 5. This shows the power inefficiency of circuits implemented by conventional CMOS standard cells. Therefore the power in the control and decoder unit could be further minimized by building it with full-custom and performing circuit optimization.

The total energy consumed during the execution time in a FU is calculated by multiplying the average power consumption by the execution time. A breakdown of the power consumption within one MAC unit is depicted in Figure:7.6. These figures are similar for all the programs. The adder consumes the largest power at 36%, followed by the logic unit at 20%. Note that the power consumption in the multiplier does not include the final addition of the partial sum and partial carry which is performed in the adder and included in the power figure for the adder. Thus the power consumption of the multiplier block

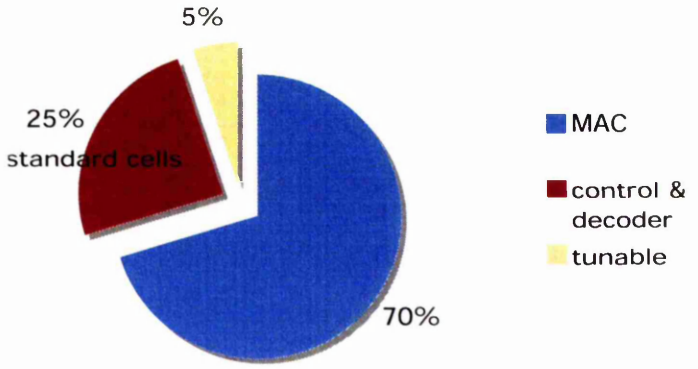


Figure 7.5: Breakdown power consumption of the functional unit

becomes only the fourth highest figure at 7%. The buffer drivers within the datapath have the same dissipation as the multiplier.

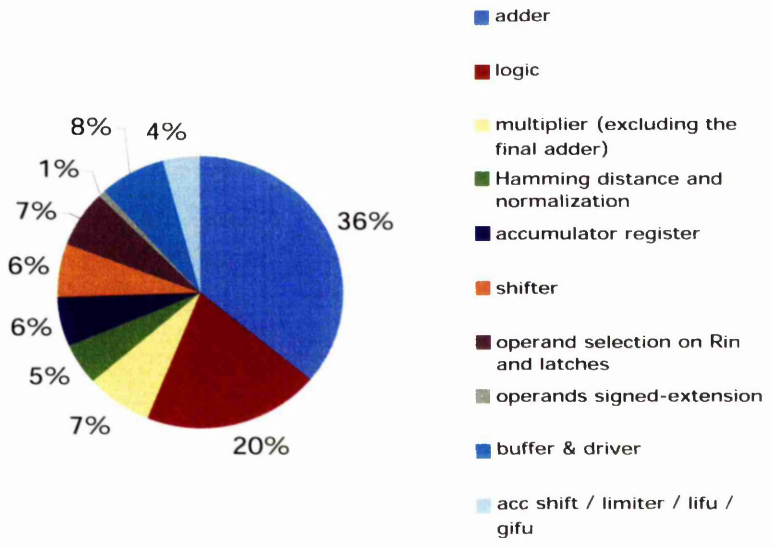


Figure 7.6: Break down average power of the arithmetic blocks in the FU

7.4.7 Projected CADRE-s

In the original CADRE, most of the research was focused on the algorithmic and architectural level design for a low power DSP and the features incorporated resulted in a

good energy economical design at these levels. The FU designed and implemented in CADRE-s has focused on the optimization of the power consumption at the logic, circuit and layout levels. Further analysis in CADRE-s can be made by replacing the FUs in the original CADRE with the CADRE-s FUs. 'Plugging' the new FUs into the original design enables further examination of the merits of the new FU provided the remaining parts of the CADRE architecture are scaled to reflect a 0.18μm geometry layout running from 1.8V. The results obtained are shown in Table 7.7 with results from the original CADRE design shown on the left and those projected for CADRE-s on the right.

Table 7.7: Estimated energy consumption of using the FU in the original CADRE architecture

	FIR			
	% total energy	Original CADRE assuming a 0.18μm geometry from 1.8V (uJ)	% total energy	Projected CADRE-s 0.18μm, 1.8V (uJ)
Instruction fetch	0.2	0.013	0.3	0.013
Instruction decode	0.7	0.045	1.2	0.045
Data memory	1.2	0.078	2.0	0.078
Program memory	1.9	0.123	3.2	0.123
Instruction buffer	2.1	0.136	3.5	0.136
Index update	2.2	0.143	3.7	0.143
Address update	5.4	0.351	9.1	0.351
Register bank	8.7	0.565	14.6	0.565
Config. memories	23.2	1.507	39.0	1.507
MAC units	50.5	3.280	16.8	0.647
Remainder	3.9	0.253	6.6	0.253
Total (uJ)		6.496		3.861

The most important improvement is that the new MAC unit now only requires 17% of the total execution energy compared with over 50% in the original design demonstrating the contribution that energy efficient logic, circuit and layout can make to the design. The energy saving in this important unit amounts to an energy improvement factor of 5 and overall the new FU design of CADRE-s contributes an overall system energy saving of a

factor of 1.7. Therefore, a significant energy saving of the overall system can be gained when low energy techniques at the logic and circuit level are applied to a design.

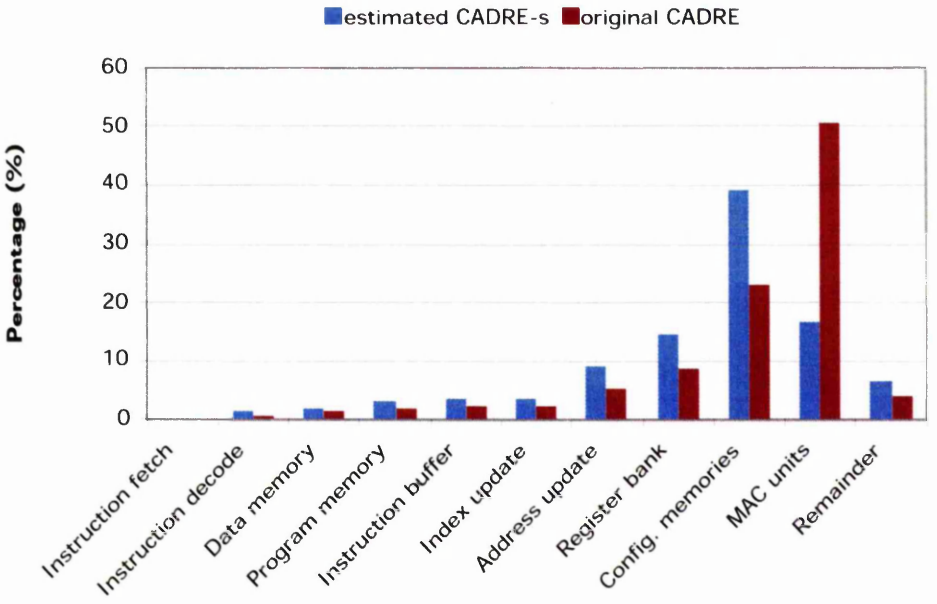


Figure 7.7: Comparison of distribution of energy throughout the original CADRE and the projected CADRE-s

Figure:7.7 and Figure:7.8 show the percentage breakdown of energy in the projected CADRE-s and the original CADRE scaled to 0.18μm and 1.8V. These show that the largest energy dissipated in the projected CADRE-s is from the configurable memories at 39%. The MAC Units consume the second largest energy at about 17%, which is roughly the same as the register bank. These results suggest that the configurable memories and register bank are the next units which should be targeted for lowering the power dissipation.

7.5 Comparison with other DSPs

A comparison with the DSPs developed by other research groups or commercial manufacturers is difficult because CADRE-s does not include the same range of blocks as the others. CADRE-s is built to demonstrate an energy efficient functional unit. Therefore

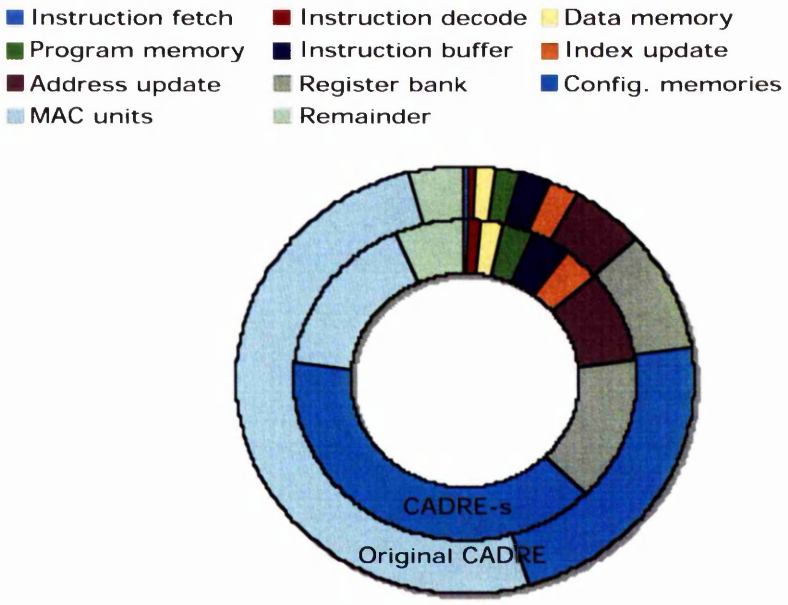


Figure 7.8: Distribution of percentage energy throughout the Original CADRE versus the projected CADRE-s

other necessary features such as memory addressing, register operations, jump and branch are not included in this design. For this reason, the energy of a complete CADRE-s as projected in Table7.6 is used.

In addition, the difference in process technology can make comparisons difficult because of different process parameters such as capacitance, delay and the supply voltage. For this reason, the power (mW) per MHz^2 for some current selected fixed-point DSPs has been scaled to 0.18m@1.8V. This has been done by calculating performance and power factors where the performance factors and power factors are linearly scaled by the geometry size and V_{DD}^2, respectively. The original power and performance features of the selected DSPs are then multiplied by the performance and power factors in Table 7.8, to give the scaled power and performance of the DSPs shown in Table 7.9. These comparisons need to be treated with caution because the scaling calculation can only be a rough guide to power and performance; effects which are becoming apparent at geometries below 0.18μm are ignored. Furthermore the benchmarks producing the power and performance for other DSP's are not known. The decision for using any DSP should be based on the

evaluation results when performing the benchmarks and it has to fit with the specification of target applications.

Table 7.8: Performance and power factor when process technology scaling to 0.18μm@1.8V

Voltage (V)	Geometry (um)	Performance factor	Power factor
1.4	0.12	0.67	1.65
2.5	0.25	1.39	0.52
1.2	0.09	0.5	2.25

Table 7.9: Power per million instruction (mW/MHz^2) of the commercial fixed-point DSPs

DSP	Scaled Power Consumption (mW)	Scaled Frequency (MHz)	mW/ MHz	mW/ MHz^2	Factor comparing to worst case CADRE-s mW/MHz^2
TMS320c6414[152]	2095	402	5.21	0.0130	~1.35
Carmel[153]	104	167	0.62	0.0037	~0.38
Saturn[23]	49.5	100	0.49	0.0049	~0.51
Projected CADRE-s (worst)	24	50	0.48	0.0096	1
Projected CADRE-s (best)	18.5	66	0.28	0.0040	~0.42

The projected worst case CADRE-s represents the slowest performance when the MAC operation is continually performed whilst the projected best case CADRE-s occurs when only addition and logic functions are continually performed.

The TMS320C6414 is the highest-performance fixed-point DSP and based on the VLIW architecture developed by Texas Instruments (TI). Its C64x DSP processor has 64 general-purpose registers of 32-bit word length and eight highly independent functional units comprising two multipliers computing a 32-bit result and six ALUs. It is implemented on a 0.12μm/ 6-level Metal Process (CMOS) and operated from a 1.4V supply.

The second generation of Infineon's Carmel family of 16-bit fixed-point processors operates from a 2.5V supply, and Siemens report a typical power consumption (excluding

peripherals and memory) of 200mW at 120 MHz on a 0.25μm process. Carmel's data paths contain six execution units (exponent unit, shifter, two ALUs and two MACs) and the core can issue and execute up to six native instructions at a time via its user-defined super instructions. Carmel's data paths share a common set of registers. Carmel provides a total of six 40-bit accumulators plus 26 address registers. Unlike the 'C6xxx', Carmel is not a load/store architecture and operands can come directly from memory as well as from registers.

The Saturn DSP core is a 16-bit general-purpose open configurable DSP, targeted at wireless systems like GSM and 3G. The Saturn DSP core consists of a dual Harvard-VLIW architecture with two independent 16x16 multipliers, four 16-bit ALUs and multiple additional parallel resources. This DSP is an open synthesizable core with a typical system performance in a standard 90nm low power CMOS process and operated from a 1.2V supply.

From Table 7.9, the projected CADRE-s performs reasonably on the basis of power per Megahertz^2 (mW/MHz^2) compared to other DSPs. The projected best case CADRE-s has a better energy efficiency than other selected DSPs except Carmel DSP. The projected worst case CADRE-s has a better power efficiency than TMS320C6414 by a factor of 1.35.

The timing delay used in CADRE-s is over estimated by about 30-40%. The author has used too much delay because an asynchronous system is very sensitive to the timing delay and could easily result in a system failure. In practice, taking the delay model out of the data path at the layout stage is much easier than adding a delay into the data path and the tunable delay mechanism will assist with this. However, CADRE-s has a big advantage over commercial DSPs when the supply voltage is scaled down to save energy. This is because an asynchronous design automatically adjusts to different supplies without needing to consider a clock frequency.

The design for low energy or energy efficiency requires correct decisions for applying design techniques at all levels, especially in the logic and circuit levels. Due to time constraints, most of the work on CADRE-s has been at the logic, circuit and layout levels as applied to a full custom data path design. The original CADRE itself had a complex

architecture which would require a large research team to implement. In addition, limitation of the EDA tools and computers to design and simulate a large system such as CADRE-s has extended the time to produce the chip.

Chapter 8: Conclusions

Digital signal processors are widely employed in portable devices and these demand that coherent low energy design techniques are applied at every design level in order to cope with increasingly higher performance. This thesis has demonstrated that coherent design techniques can improve FU energy efficiency to enable smaller area and longer running portable devices. In addition, new logic for performing the Hamming distance and normalization, energy efficient circuits based on PTG circuits and a new timing mechanism have been added to the FU to reduce the amount of logic and possible failure of the system.

The goal of this research from the beginning was to significantly improve the power efficiency of a FU for DSPs. The results in the previous chapter demonstrate that the FU designed can achieve an energy improvement by a factor of 5 in the MAC units and a factor of nearly 2 for the overall system compared with the original CADRE system. However, there are also other research contributions mentioned in chapter1. Looking at each of these in turn:

- CADRE-s has successfully demonstrated an energy-efficient architectural framework comprising four asynchronous FUs and data memories for executing digital signal processing algorithms. Kernel benchmarks have been used to evaluate CADRE-s and the results show that CADRE-s mostly has a better energy efficiency than other selected current DSPs.

- The configuration memory embedded in the FU of the CADRE-s architecture is unique. CADRE-s also shows that users can gain better performance and energy saving by judicious use of the configuration memory as shown in the results for the FIR-20v1 and FIR-20v2 in the previous chapter. The main advantage of the

configuration memory however is the flexibility it gives the user and system programmer. It enables FUs to be controlled directly by a configuration instruction. The implementation of the configuration memory has led to additional power and area cost. However, this cost can be justified by the flexibility and performance gained by users. This feature also allows each FU to operate independently on different instruction streams. However, energy figures for CADRE-s suggest that the configurable memories are the largest energy consumer and a lowering of their energy levels is a topic for future research work.

- The estimated power for CADRE-s based on the scaled original CADRE but using the FU from this research work shows that the energy saving is a factor of 2, whilst the energy saving in the MAC unit achieves a factor of 5. This shows that lowering power at the logic, circuit and layout makes a large contribution to overall power levels.

- A large power improvement in an unusual low power FU datapath, which is particularly suitable for use in DSP's aimed at portable applications, provides a component which combines many good low power circuit design techniques in each internal block. This leads to high performance and keeps the power dissipation of the FU low. The low energy techniques used are widely applicable to other digital designs.

- The FUs have the ability to keep the execution units supplied with operands by supporting parallel movement. Four arithmetic operations can operate concurrently whilst the data movement of four FUs can occur in parallel.

- A technique to reduce the switching activities and the number of stages of addition in the multiplier has been developed by using a parallel Wallace tree structure with balanced inputs and sharing the final addition circuitry between the adder and multiplier. The use of a shared adder has not previously been described. This makes the energy consumption of the addition and multiplication in CADRE-s relatively small as the comparison results show in chapter4. This technique is generally applicable to all DSP design. Furthermore, the MAC unit of CADRE-s gives the best energy efficiency compared to other designs.

- A new design for a logic block which combines the Hamming distance and normalization functions is used in the FU. This should reduce the overall power level and area for these functions.

- A novel tunable timing mechanism to better tune the asynchronous control logic to the data path is employed in the FU. Simulations show that this approach should be able to easily reduce the over-conservative timing margins by at least 10% without incurring any operating problems.

- CADRE-s can gain better energy efficiency by about 20-30% when voltage scaling is applied. Futhermore, PTG logic appears to scale well with regard to energy efficiency as the voltage scales down. The only potential problem is the increasing effect of leakage and further research here is needed.

- CADRE-s can be regarded as an energy efficient IP block which can be used in future designs. In particular, the implementation can be used on future super-scalar asynchronous DSP architectures. These architectures are aimed at next generation designs requiring a high performance and low power.

8.1 Current Directions

In the rapidly growing DSP industry, some techniques used in this thesis are already used commercially. A parallel MAC unit can be found in some commercial DSPs such as the TMS320C6414 and Carmel's DSP. An adaptive supply and frequency can be seen in some commercial general purpose processors such as the AMD processor, which dynamically varies voltage and frequency over the range of 1.4-1.8V and 200-500MHz, and the Intel XScale processor (the second generation of StrongARM), which dynamically operates over the voltage and frequency range of 0.7-1.75V and 150-800MHz. However, there is no report of using a variable supply voltage in a DSP as yet. Furthermore, a commercial asynchronous DSP processor has not been reported either. Therefore, these are the good reasons why CADRE-s is a good candidate for an energy efficient DSP in future portable applications.

8.2 Future Research Directions

CADRE-s provides the groundwork for a variety of continuing research directions. Further research is required on the software as well as all aspects of energy efficient hardware design.

Integrating the compiler which can convert a high level language such as the C language onto the parallel architecture effectively needs to be explored. The compiler should take care of the code optimization, the use of resources in a parallel system, and avoiding data conflicts. Fabrication of the current design would enable further research on multiple and variable voltage supplies, without impacting on system cost and complexity. The compiler could assign a different voltage supply to a DSP based on the tasks or program execution.

Another research direction would further explore the instruction set and architecture for improving energy efficiency. A particular problem on VLIW and parallel DSP architectures is to ensure that the units can be supplied with operands at a sufficient rate and mechanisms to support this are required. A register file which has the ability to deal with parallel problems such as data conflicts should be explored. Because of using configuration memories in CADRE-s to reduce the length of a VLIW instruction, the optimal number of such memories could be investigated.

As process technology continues to advance, a smaller geometry could increase the leakage power, especially in pass transistor logic. Thus, further investigation of controlling the leakage could yield additional improvements to the DSP energy efficiency.

Finally, since full custom design consumes so much design effort, the FU in this thesis could be seen as a macro cell and used in future designs to produce a complete system.

8.3 Conclusions

This PhD set as its goal the significant reduction of the power consumption in the original CADRE. Initial post-layout simulations show that CADRE-s (best case) has already reduced the mW/MHz figure by a factor of 3 compared to the TMS320C6414, which is

targeted at wireless applications such as the mobile phone. The functional unit is energy efficient and can be used in the next generation of DSP architectures whilst the multiply accumulator unit in CADRE-s gives the best energy efficiency compared to other designs.

Finally, the most important contribution of this work is to show that energy efficient logic, circuit and layout techniques make a significant contribution to saving energy. Thus it is in the designers interest to not only consider energy efficiency at the algorithmic and architectural levels. Energy efficiency requires a holistic approach to design.

References

[1] J.L.Hennessy and D.A.Patterson, "Computer Architecture: A Quantitative Approach.", 3rd Edition, *Morgan Kaufmann,* 2003, ISBN:1-55860-596-7.

[2] J.M.C. Stork, "Technology Leverage for Ultra-low Power Information Systems", *Proceedsing of IEEE*, vol. 83, no.4, 1995.

[3] http://www.thermoanalytics.com/support/publications/batterytypesdoc.html

[4] Anantha P. Chadrakasan and Robert Broderson, "Low Power CMOS Design", *John Wiley & Sons Inc.,* ISBN 0780334299, 1997.

[5] Anantha P. Chandrakasan, Robert W. Brodersen, "Minimizing Power Consumption in CMOS Circuits", *Proceedings of the IEEE,* vol.83, April, pp.498-523, 1995.

[6] http://www.arm.com

[7] R. Witek and J. Montanaro, "StrongARM: a high-performance ARM processor", *Technologies for the Information Superhighway Compcon'96,* February, pp.188-191, 1996.

[8] http://www.ti.com

[9] V.Baumgarte, G.Ehlers and et al, "PACT XPP-A Self-reconfigurable data processing architecture", *Journal of Supercomputing*, vol.26, September, pp.167-184, 2003.

[10] M.Wan, Hui Zhang, M.Benes and J.Rabaey, "A low-power reconfigurable data-flow driven DSP system", *Proceedings 1999 IEEE Workshop on Signal Processing Systems,* October, pp.191-200, 1999.

[11] H.Zhang, V.Prabhu, V.George, M.Wan, et. al., "A 1V Heterogeneous Reconfigurable Processor IC for Baseband Wireless Applications", *International Solid State Circuits Conference,* February, pp.68-69, 2000.

[12] Paul M. Heysters, Henri Bouma, Jaap Smit, and et al, "Reconfigurable System Design: The Control Part", *Proceedings 2nd workshop on Embedded Systems*, October, pp. 89-94, 2001.

[13] Hyujune Yoo, Ecksang Ko, Soohwan Ong and Myung H. Sunwoo, "A Multimedia DSP Chip for Portable Applications", *The First IEEE Asia Pacific Conference on ASICs*, August, pp.147-150, 1999.

[14] M.Lewis and L. Brackenbury, "CADRE: An Asynchronous Embedded DSP for Mobile Phone Applications", *Design Automation for Embedded Systems,* Vol.6, No.4, pp.451-475, 2002.

[15] E.E.Swartzlander, W.K.W.Young and S.J.Joseph, "A Radix-4 delay commutator for fast Fourier transform processor implementation", *Journal Solid-State Circuits,* vol.SC-19, October, pp.702-709, 1984.

[16] P. Kievits, E. Lambers, C. Moerman and R. Woudsma, "R.E.A.L. DSP Technology for Telecom Based-band Processing", *Proceeding 9th International Conference on Signal Processing Applications and Technology*, Miller Freeman Inc., 1998.

[17] Carmel DSP Core Technical Overview Handbook, *Infineon Technologies*, 2000.

[18] http://www.ieee.org/organizations/society/sp/mongrphs.html

[19] Jennifer Eyre and Jeff Bier "DSP Processors Hit the Mainstream", *IEEE Computer Society,* vol.31, no.8, August, pp.51-59, 1998.

[20] J. Von Neumann, "First draft of a report on the EDVAC", Reprinted in W.Aspray and A.Burks, eds., *Papers of John von Neumann on Computing and Computer Theory,* The MIT Press, pp.592, 1945.

[21] N. Mozaffar and N.Z. Azeemi, "Design and Implementation of a SHARC Digital Signal Processor Core in Verilog HDL", *7th International Multi Topic Conference*, December, pp.247-252, 2003.

[22] S.K.Tewksbury, K.Devabattini and V. Gandikota, "A Parallel DSP testbed with a Heterogeneous and Reconfigurable network fabric", *Proceedings on 2nd IEEE international Conference on Innovative Systems in Silicon*, October, pp.310-322, 1997.

[23] Kee Moerman, "Digital signal processing options for wireless handsets.", *Adenlante Technologies*, White-paper, http://www.adelantetech.com/upload/search/wireless_handsets.pdf

[24] Shannon Wichman and Nagendra Goel, "The second generation ZSP DSP.", *LSI Logic Corp.*, White-paper.

[25] Yuan-Hao Huang, Hsi-Pin Ma, Ming-Luen Lioy and Tzi-Dar Chiueh, "A 1.1G MAC/s Sub-Word-Parallel Digital Signal Processor for Wireless Communication Applications.", IEEE Journal of Solid-state Circuits, vol.39, No.1, January, pp.169-183, 2004.

[26] http://www.motorola.com/semiconductors

[27] http://www.epanorama.net/documents/telecom/ulaw_alaw.html

[28] B. Kim et al., "MDSP-II : A 16-bit DSP with Mobile Communication Accelerator", *IEEE Journal of Solid-State Circuits,* vol.34, no.3, March, pp.397-404, 1999.

[29] Li-Hsun Chen, Wei-Lung Liu, Oscal T.-C. Chen and Rueng-Liang Ma, "A Reconfigurable Digital Signal Processor Architecture for High-Efficiency MPEG4 video encoding", *IEEE International Conference on Multimedia and Expo.,* vol.2, August, pp.165-168, 2002.

[30] Reiner H. "Reconfigurable Computing: A New Business Model and its Impact on SoC Design", *Keynote speeches in EUROMICRO Symposium on Digital System Design, Architectures, Methods and Tools,* September, 2001.

[31] Buchmann I., "Understanding your batteries in a portable world. Article on battery choice and how to maximize service life", *The 4th Battery Conference on Applications and Advances*, January, pp.369-373, 1999.

[32] Dyer C. K., "Replacing the battery in portable electronics", *Scientific American*, July, pp.70-75, 1999.

[33] S. Li and J. W. Evans, "Electrochemical-thermal model of lithium polymer batteries", *Journal Electrochemical Society*, vol.147, June, pp. 2086-2095, 2000.

[34] H. Veendrick, "Short Circuit Dissipation of Static CMOS Circuitry and its impact on the design of buffer circuits", *IEEE Journal of Solid-State Circuits*, vol.19, no.4, August, pp.468-473, 1984.

[35] http://www.allaboutcircuits.com/vol_3/chpt_3/1.html

[36] C. Hu, "Device and Technology Impact on Low Power Electronics: Low Power Design Methodologies", *Kluwer Academic*, pp. 21-36, 1996.

[37] Calhoun B.H., Honore F.A., Chandrakasan A.P, "A Leakage Reduction Methodology for Distributed MTCMOS", *IEEE Journal of Solid-State Circuits*, vol. 39, no. 5, May, pp. 818-826, 2004.

[38] Hanchate N., Ranganathan N, "LECTOR: A Technique for Leakage Reduction in CMOS Circuits", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 2, February, pp. 196-205, 2004.

[39] Tsai Y.-F., Duarte D., Vijaykrishnan N., Irwin M.J, "Implications of Technology Scaling on Leakage Reduction Techniques", *Proceedings Design Automation Conference*, June, pp. 187-190, 2003.

[40] J. Rabaey, "Digital Integrated Circuits: A Design Perspective.", *Prentice Hall*, Upper Saddle River, NJ, 1996.

[41] T.D.Burd, "Energy-Efficient Processor System Design", *Doctor of Philosophy Thesis,* Engineering-Electrical Engineering and Computer Sciences, University of California, Berkeley, 2001.

[42] H.B. Bakoglu, "Circuits, Interconnections, and Packaging for VLSI.", *Addison-Wesley*, Menlo Park, CA, 1990.

[43] Borah M., Owens R.M., Irwin M.J, "Transistor sizing for low power CMOS circuits.", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.15, no.6, June, pp. 665-671, 1996.

[44] Santos C., Wilke G., Lazzari C., Reis R., Guntzel J.L, "A transistor sizing method applied to an automatic layout generation tool", *Proceedings 16th Symposium on Integrated Circuits and Systems Design*, September, pp. 303-307, 2003.

[45] Jiren Yuan, Svensson C., "Principle of CMOS circuit power-delay optimization with transistor sizing", *IEEE International Symposium on Circuits and Systems*, vol.1, May, pp.637-640, 1996.

[46] Augsburger S., Nikolic B., "Combining dual-supply, dual-threshold and transistor sizing for power reduction", *Proceedings IEEE International Computer Design: VLSI in Computers and Processors*, September, pp. 316-321, 2002.

[47] P.Pant, R.K.Roy and A.Chattejee, "Dual-threshold voltage assignment with transistor sizing for low power CMOS circuits", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol.9, no.2, April, pp.390-394, 2001.

[48] P. Penzes, M. NystrOm and A. J. Martin, "Transistor sizing of energy-delay-efficient circuits", 10th TAU workshop, 2002.

[49] Edward T. Lewis, "Optimization of device area and overall delay for CMOS VLSI designs", *Proceedings of IEEE*, vol. 72, pp. 670-689, 1984.

[50] M. A. Cirit, "Transistor sizing in CMOS circuit", *Proceedings of 24th ACM/IEEE Design Automation Conference*, pp. 121-124, 1987.

[51] J. Yuan and C. Svensson, "CMOS circuit speed optimization based on switch level simulation", *Proceedings of 1988 IEEE International Symposium on Circuits and Systems*, vol. 3, pp.2109-2112, 1988.

[52] W.Elmore, "The transient response of damped linear networks with particular regard to wide band amplifiers", *Journal of Applied Physics,* vol.19, no.1, January, pp.55-63, 1948.

[53] L. Chen, M. Margala, "Power-Efficiency Driven Device Sizing of Pass-Transistor Digital Circuits in Low-Voltage CMOS", *International Workshop in Power and Timing Modelling, Optimization and Simulation (PATMOS)*, September, pp.6.2.1-6.2.13, 2001.

[54] Santos C.,Wilke G., Lazzari C., Reis R., Guntzel J.L., "A transistor sizing method applied to an automatic layout generation tool", *Proceedings of 16th Symposium on Integrated Circuits and Systems Design*, September, pp. 303-307, 2003.

[55] Sundararajan V., Sapatnekar S.S., Parhi K.K., "Fast and exact transistor sizing based on iterative relaxation", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.21, no.5, May, pp.568-581, 2002.

[56] Wroblewski A., Schumecher O., Schimpfle C.V., Nossek J.A., "Minimizing gate capacitances with transistor sizing", *The 2001 IEEE International Symposium on Circuits and Systems*, vol.4, May, pp.186-189,2001.

[57] R. Gonzalez, B. M. Gordon and M. A. Horowitz, "Supply and Threshold Voltage Scaling for Low Power CMOS", *IEEE Journal of Solid-State Circuits*, vol.32, no.8, August, pp.1210-1216, 1997.

[58] S. Mutoh, S. Shigematsu, Y. Matsuya, et al., "A 1V multi-threshold voltage CMOS DSP with an efficient power management technique for mobile phone applications", *IEEE international Solid-State Circuits*, vol.39, February, pp.168-169, 1996.

[59] K. Seta, H. Hara, T. Kuroda, et al., "50% active-power saving without speed degradation using stand-by power reduction (SPR) circuit", *IEEE international Solid-State Circuits*, vol.38,February, pp.318-319, 1995.

[60] Hui Zhang, V. George, Jan M. Rabaey, "Low-Swing On-Chip Signaling Techniques: Effectiveness and Robustness", *IEEE Transactions on very large scale integration (VLSI) systems*, vol.8, no.3, June, pp.264-272, 2000.

[61] Ram K. Krishnamurthy, Herman Schmit, L.Richard Carley, "A Low-Power 16-bit Multiplier-Accumulator using Series-regulated Mixed Swing Techniques", *Proceedings of the IEEE Custom Integrated Circuits Conference,* May, pp.499-502, 1998.

[62] K.Yano, Y.Sasaki, K.Rikino and K.Seki, "Top-down Pass-transistor Logic Design", *Journal of Solid-State,* vol.31, June, pp.792-803, 1996.

[63] L. G. Heller, W. R. Griffin, J. W. Davis and N. G. Thoma, "Cascode Voltage Switch Logic: A Differential CMOS Logic Family", Proceedings of International Solid-State Circuits Conference, pp. 16-17, 1984.

[64] K. Chu and D.L. Pulfrey, "A Comparison of CMOS Circuit Techniques: Differential Cascode Voltage Switch Logic Versus Conventional Logic", *IEEE Journal of Solid-State Circuits*, vol.21, August, pp. 528-532, 1987.

[65] Ivan Sutherland, Bob Sproull and David Harris, "Logical Effort Designing Fast CMOS Circuits", *Morgan Kaufmann Publishers*, 1999.

[66] I.E. Sutherland, "Micropipelines", *Communications of the ACM*, vol.32, no.6, June, pp.720-738, 1980.

[67] M.E. Dean, T.E. Williams and D.L. Dill, "Efficient self-timing with level-encoded 2-phase dual-rail (LEDR)", *MIT Conference on Advanced Research in VLSI*, March, 1991.

[68] V. Varshavsky, V. Marakhovsy and M. Tsukisaka, "Data-controlled delays in the asynchronous design", *Proceeding Of the 1996 IEEE International Symposium Circuits and Systems (ISCAS96)*, vol. 4, May, pp. 153-155, 1996.

[69] E. Grass, Viv Bartlett and Izzet Kale, "Completion-Detection Techniques for Asynchronous Circuits", *IEICE Trans. Inf.&Syst.*, vol.E80-D, no. 3, March, pp.344-350, 1997.

[70] E.Gras, R.C.S.Morling and I.Kale, "Activity-Monitoring Completion-Detection (AMCD): A new single rail approach to achieve self timing", *Proceeding in 2nd International Symposium on Advanced Research in Asynchronous Circuits and Systems*, Aizu Japan IEEE Computer Society Press, 1996.

[71] Li-Hsun C., Chen O.T.-C. and Ruey-Ling M., "A high-efficiency reconfigurable digital signal processor for multimedia computing", *Proceedings of the 2003 International Symposium on Circuits and Systems*, vol.2, May, pp. 768-771, 2003.

[72] M. J. G. Lewis, "Low Power Asynchronous Digital Signal Processing", *Doctor of Philosophy Thesis*, Faculty of Science and Engineering, University of Manchester, 2000.

[73] D.K.Arvind and Robert D. Mullins, "A Fully Asynchronous Superscalar Architecture", *International Conference on Parallel Architecture and Compilation Techniques,* October, p.17-22, 1999.

[74] R.Fried, "High-efficiency low-power one-clock solutions for multi-clock chips and systems", *IEEE-CAS Region 8 Workshop on Analog and Mixed IC Design*, September, pp. 60-65, 1996.

[75] D. Peiliang, Y. Rilong, X. Hongbo and Y. Chengfang, "Multi-clock driven system: a novel VLSI architecture", *Proceedings 4th International Conference on ASIC,* October, pp.555-558, 2001.

[76] M. Singh and M. Theobald, "Generalized latency-insensitive systems for single-clock and multi-clock architectures", *Proceedings on Design, Automation and Test in Europe Conference and Exhibition,* vol.2, February, pp.1008-1013, 2004.

[77] M. Pedram, "Power Minimization in IC Design: Principles and Applications", *ACM Transactions on Design Automation of Electronic Systems (TODAES),* vol.1, January, pp.6-56, 1996.

[78] J.H.Stewart, "Application of Scan/set for Error Detection and Diagnostics", *Semiconductor Test Conference*, Cherry Hill, New Jersey, October, pp.152-158, 1978.

[79] Aamir A. Farooqui, Vojn G. Oklobdzija and Farzad Chechrazi, "Multiplexer Based Adder for Media Signal Processing", *International Symposium on VLSI Technology System and Applications*, June, pp. 100-103, 1999.

[80] Wayne Wolf, "Modern VLSI Design: System-on-Chip Design", *Prentice-Hall Inc.*, Third Edition, 2002.

[81] Keshab K. Parhi, "VLSI Digital Signal Processing Systems: Design and Implementation", *A Wiley-Interscience Publication*, 1999.

[82] T.G. Noll "Carry-save architectures for high-speed digital signal processing", Journal VLSI Signal Processing, vol.3, pp. 121-140, 1991.

[83] P.M. Kogge and H.S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Transaction computer*, vol. C-22, pp.786-792, August, 1973.

[84] S.S.Mahanant-Shetti, P.T.Balsara, and C. Lemonds, "High Performance low power array multiplier using temporal tiling", *IEEE Transaction Very Large Scale Integration (VLSI) Systems,* vol.7,no.1, March, pp.121-124, 1999.

[85] M.D.Ercegovac and T.Lang, "Fast multiplication without carry-propagate addition", *IEEE Transaction Computer,* vol.39, no.11, pp.1385-1390, 1990.

[86] H. Al-Twaijry and M.J.Flynn, "Multipliers and Datapaths", *Technical Report CSL-TR94-654*, Standford University, 1994.

[87] Alok A. Katkar and James E. Stine, "Modified booth truncated multipliers", *Proceedings of the 14th ACM Great Lakes Symposium on VLSI*, pp. 444-447, 2004.

[88] B.Parhami, "Computer Arithmetic: Algorithms and Hardware Designs", *Oxford University Press,* 2002.

[89] C.S. Wallace, "A Suggestion for Fast Multipliers", *IEEE Transactions Electronic Computer*, vol. EC-13, February, pp. 14-17, 1964.

[90] L. Dadda and D.Ferrai, "Digital multipliers: A unified approach", *Alta Frequenza,* vol. 37, November, pp.1079-1086, 1968.

[91] A Weinberger, "4:2 Carry-Save Adder module", *IBM Technical Disclosure Bullentin*, vol.23, January, 1981.

[92] B. R. Zeydel, V. G. Oklobdzija, S. Mathew, R. K. Krishnamurthy and S. Borkar, "A 90nm 1GHz 22mW 16x16-bit 2's Complement Multiplier for Wireless Baseband", *Symposium on VLSI Circuits Digest of Technical Papers*, pp.235-236, 2003.

[93] J.B.Sulistyo and D. Sam Ha, "5GHz pipelined multiplier and MAC in 0.18um complementary static CMOS", *ISCAS'03*, May, pp.117-120, 2003.

[94] C.F.Law, S.S.Rofail and K.S.Yeo, "A low power 16x16-b parallel multiplier utilizing pass transistor logic", *IEEE Journal of Solid-State Circuits*, Vol.34, No.10, October, pp.1395-1399, 1999.

[95] S.Agarwala et al, "A 600MHz VLIW DSP", *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, vol.1, pp.56-444, 2002.

[96] Viv A. Bartlett, "Exploiting data-Dependencies in Low-Power Asynchronous VLSI Signal Processors", *Doctor of Philosophy Thesis*, Department of Electronic Systems, University of Westminster, 2000

[97] Y. Oike, M. Ikeda and K. Asada, "A High-Speed and Low-Voltage Associate Co-Processor with Hamming Distance Ordering using Word-Parallel and Hierarchical Search Architecture", *IEEE Custom Integrated Circuits Conference*, pp.643-646, 2003.

[98] H.J.Mattausch et al, "Fully-Parallel Pattern-Matching Engine with Dynamic Adaptability to Hamming or Manhattan Distance", *Symp. on VLSI Circuits Digest Technical Papers,* pp.252-255, 2002.

[99] M.Fujino and V.G.Moshnyaga, "An efficient Hamming distance comparator for low-power applications", *International Conference on Electronics Circuits and Systems,* vol.2, September, pp.641-644, 2002.

[100] K. Asada, S. Komatsu and M. Ikeda, "Associative Memory with Minimum Hamming Distance Detector and it's application to Bus Data Encoding", *Proceedings of Asia-Pacific Application Specific Integrated Circuits*, August pp.16.1-16.4, 1999.

[101] DSP56000/DSP56001 Digital signal processor user's manual, *MOTOROLA INC.*, 1990.

[102] T. Schneider, R. Brennan, P. Balsiger and A. Heubi, "An Ultra Low-power Programmable DSP System for Hearing Aids and Other Audio Applications", *Proceedings of Conference on Signal Processing Applications and Technology*, November, 1999.

[103] L. S. Nielsen, C. Niessen, J. Sparso, "Low-power operation using self-timed and adaptive scaling of the supply voltage," *IEEE Trans. VLSI Systems*, vol.2, pp.391-397, 1994.

[104] Asynchronous Design - A Systems Perspective, editors J. Sparsø and S. B. Furber, Kluwer Academic Publishers, 2001.

[105] S.M.Nowick, K.Y.Yun, P.A.Beerel and A.E.Dooply, "Speculative Completion for the Design of High-Performance Asynchronous Dynamic Adders", *Proceedings of Async97*, April, pp.210-223, 1997.

[106] K. Usami and M. Horowitz, "Clustered voltage scaling technique for low-power design", *Proceedings of International Symposium on Low Power Design*, April, pages. 3-8, 1995.

[107]A.Chandrakasan, V.Gutknik, T.Xanthopoulos, "Data driven Signal Processing: An approach for Energy Efficient Computing", *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, August, pp.347-352, 1996.

[108]T.Ishihara, H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors", *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED),* August, pp.197-202, 1998.

[109]W.Kuang, J.S. Yuan, A.Ejnioui, "Supply Voltage Scalable System Design Using Self-Timed Circuits", *Proceedings of IEEE Computer Society Annual Symposium on VLSI,* February, pp.161-166, 2003.

[110]S.Raje and M. Sarrafzadeh, "Scheduling with multiple voltages under resource constraints", *IEEE International Symposium on Circuits and Systems,* vol.1, June, pp.350-353, 1999.

[111]J.A.Barnett, "Dynamic Task-Level Voltage Scheduling Optimization", *IEEE Transactions on Computers,* vol.54, no.5, May, pp.508-520, 2005.

[112]M.C.Johnson and K.Roy, "Datapath Scheduling with Multiple Supply Voltage and Level Converters", *ACM Transactions on Design Automation of Electronic Systems,* vol.2, no.3, July, pp.227-248, 1997.

[113]R.Jejurikar, C. Pereira and R. Gupta, "Leakage Aware Dynamic Voltage Scaling for Real-Time Embedded Systems", *Proceedings of Design Automation Conference (DAC'04),* June, pp.275-280, 2004.

[114]A.Menon, S.K.Nandy and M.Mehendale, "Multivoltage Scheduling with Voltage-Partitioned Variable Storage", *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED),* August, pp.298-301, 2003.

[115]J.Choi, C.Lin and H.Kim, "A Low Power Register Scheduling and Allocation Algorithm", *Proceedsings of IEEE Region 10 Electrical and Electronic Technology (TENCON),* August, pp.627-630, 2001.

[116]A.Manzak and C.Chakrabarti, "A Low Power Scheduling Scheme with Resources Operating at Multiple Voltages", *IEEE Transactions on VLSI,* vol.10, no.1, February, pp.6-14, 2002.

[117]J.Seo, T. Kim and K. Chung, "Profile-based Optimal Intra-task Voltage Scheduling for Hard Real-time Applications", *Design Automation Conference,* June, pp.87-92, 2004.

[118]S.Hua, and G.Qu, "Approaching the Maximum Energy Saving on Embedded Systems with Multiple Voltages", *International Conference on Computer-Aided Design,* November, pp.26-29, 2003.

[119]M.Buss, T.Givargis and N.Dutt, "Exploring Efficient Operating Points for Voltage Scaled Embedded Processor Cores", *IEEE International Real-time Systems Symposium,* December, pp.275-281, 2003.

[120] J.Seo and N.Dutt, "A Generalized Technique for Energy-Efficient Operating Voltage Set-up in Dynamic Voltage Scaled Processors", *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp.836-841, 2005.

[121] V. Zyuban and P. Strenski, "Unified Methodology for resolving power-performance trade-off at the microarchitectural and circuit levels", *International Symposium on low-power Electronics and Design,* August, pp.166-171,2002.

[122] R.W. Brodersen, M.A. Horowitz, D. Markovic, B. Nikolic and V. Stojanovic, "Methods for True Power Minimization", *IEEE/ACM International Conference on Computer Aided Design*, November, pp.35-42, 2002.

[123] M. Suzuki, N. Ohkubo, and et al, "A 1.5ns 32-b CMOS ALU in double pass-transistor logic", *IEEE Journal of Solid-State Circuits*, vol. 28, pp.1145-1151, 1993.

[124]K. Yao et al, "A 3.8ns CMOS 16x16-b multiplier using complementary pass-transistor logic", *IEEE Journal of Solid-State Circuits*, vol. 25, April, pp.388-393, 1990.

[125]A. Parameswar, H. Hara and T. Sakurai, "A swing restored pass-transistor logic-based multiply and accumulate circuit for multimedia applications", *IEEE Journal of Solid-State Circuits*, vol. 31, June, pp.805-809, 1996.

[126]M. Song, G. Kang, S. Kim, and B. Kang, "Design methodology for high speed and low power digital circuits with energy economized pass-transistor logic (EEPL)", *Proceedings 22nd European Solid-State Circuits Conference*, September, pp.120-123, 1996.

[127]W. H. Paik, H. J. Ki and S. W. Kim, "Push-pull pass transistor logic family for low-voltage and low-power", *Proceedings 22nd European Solid-State Circuits Conference*, September, pp. 116-119, 1996.

[128]Neil H.E. Weste and David Harris, "CMOS VLSI Design: A Circuits and Systems Perspective", *Addision Wesley*, 3rd Edition, 2005.

[129]R.Shalem E. John and L. K. John, "A novel low power energy recovery full adder cell", *Proceedings of the IEEE Great Lakes Symposium of VLSI*, February, pp. 380-383, 1999.

[130]Hung Tien Bui, et al, "Design and analysis of 10-transistor full adders using novel XOR-XNOR gates", *Proceedings of ICSP*, pp.619-622, 2000.

[131]W.M.Badawy, et al, "An enhanced low-power computational kernel for a pipelined multiplier-accumulator", *Proceedings of the Tenth International Conference on Microelectronics,* December, pp.33-36, 1998.

[132]Ayman A. Fayed and Magdy A. Bayoumi, "A low power 10-transistor full adder cell for embedded architectures", *Proceedings of IEEE International Symposium on Circuits and Systems*, vol.4, May, pp.226-229, 2001.

[133]I-Chyn Wey and et al, "A now low-voltage CMOS 1-bit full adder for high performance applications", *Proceedings of Asia-Pacific Conference on ASIC*, August, pp.21-24, 2002.

[134]Yingtao Jiang, et al, "A novel multiplexer-based low-power full adder", *IEEE Transactions on circuits and systems-II: Express briefs*, vol.51, no.7, July, pp.345-348, 2004.

[135]M. Munteanu, et al, "Single-ended pass transistor logic for low-power design", *33th Asilomar Conference on Signals Systems and Computers*, vol.1, October, pp.364-368, 1999.

[136]Ahmed M. Shams, et al, "Performance analysis of low-power 1-bit CMOS full adder cells", *IEEE Transactions on VLSI systems*, vol.10, no.1, february, pp.20-29, 2002.
[137]M. Nagamatsu, S. Tanaka, J. Mori, T. Noguchi and K. Hatanaka, "A 1.5ns 32x32-bit CMOS Multiplier with an Improved Parallel Structure", *Proceedings IEEE Custom Integrated Circuits Conference*, pp. 10.3.1-10.3.4, 1989.
[138]J. Mori, M. Nagamatsu, M. Hirano, S. Tanaka, M. Noda, Y. Toyoshima, K. Hashimoto, H. Hayashida and K. Maeguchi, "A 10ns 54x54-b Parallel Structured Full Array Multiplier with 0.5um CMOS Technology", *IEEE Journal Solid-State Circuits*, vol.26, April, pp. 600-606, 1991.
[139]D. Ghosh, S.K. Nandy and K. Parthasarathy, "TWTXBB: A Low Latency, High Throughput Multiplier Architecture Using a New 4-2 Compressor", *7th International Conference on VLSI Design*, January, pp. 77-82, 1994.
[140]J. Gu and C.H. Chang, "Ultra Low Voltage Low Power 4-2 Compressor for High Speed Multiplications", *Proceedings of the International Symposium on Circuits and Systems,* pp. 321-324, 2003.
[141]http://www.itu.int/home/
[142]ST Microelectronic SPS2HD RAMs datasheet.
[143]N. Ohkubo, M. Suzuki, T.Shinbo, T. Yamanaka, A. Shimizu, K. Sasaki and Y. Nakagome, "A 4.4ns CMOS 54x54-b Multiplier Using Pass-transistor Multiplier", *Proceedings of the IEEE Custom Integrated Circuit Conference*, pp. 26.4.1-26.4.4, 1994.
[144]S.F. Hsiao, M.R. Jiang and J.S. Yeh, "Design of high-speed low-power 3-2 counter and 4-2 compressor for fast multipliers", *Electronics Letters*, vol.34, no.4, February, pp.341-342, 1998.
[145]S.B.Furber and J.Liu, "A novel area-efficient binary adder", *Asilomar Conference on Signals Systems and Computers,* vol.1, October, pp.119-123, 2000.
[146]Sanjit K. Mita "Digital Signal Processing - A Computer-Based Approach", *The McGraw-Hill Companies Inc.*, 1998.
[147]S.W. Smith, "The Scientist and Engineer's Guide to Digital Signal Processing", *California Technical Publishing,* 2nd edition, 1999.
[148]International Standard, "Information Technology Coding of Moving Pictures and Associated Audio Information: Audio", *ISO/IEC 11172-3*, 1993.
[149]http://www.cadence.com/products/dfm/diva/index.aspx
[150]http://www.synopsys.com/products/success/nanosim_ss.pdf
[151]http://momonga.t.u-tokyo.ac.jp/~ooura/fft.html
[152]Texas Instruments (TI), "TMS320C6414/5/6 Power Consumption Summary", *Application report (Spra811c),* July, pp.7, 2003.
[153]Jennifer Eyre and Jeff Bier, "Carmel Enables Customizable DSP: User-Defined Instructions, Licensing Plan Set New Siemens Core Apart", *Microprocessor Report: The Insiders' Guide to Microprocessor Hardware,* vol.12, December, pp.1-6, 1998.
[154]N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform", *IEEE Transactions on Computers,* January, pp.90-93, 1974.
[155]http://www.verilog.com

Appendix A: Instruction set

This instruction causes the stored parallel instruction in the top-level RAM specified by operation to be executed. The encoding is shown in Table A.1.

Bit position	Function
5 - 0	Configuration address
9 - 6	functional unit 0-3 enables
13 - 10	accumulator write back of FU 0-3 enables

Table A.1: Top-level instruction specification

00000	MPY	10000	CMP
00001	MAC	10001	SIGN
00010	MPYR	10010	AND
00011	MACR	10011	OR
00100	ADD	10100	reserved
00101	ADC	10101	reserved
00110	ADDC	10110	ASHIFT
00111	ADCR	10111	LSHIFT
01000	SUB	11000	SCLNONE
01001	SBC	11001	SCLUP
01010	SUBR	11010	SCLDOWN
01011	SBCR	11011	reserved
01100	ABSMAX	11100	NORM
01101	ABSMIN	11101	DIST
01110	MAX	11110	XOR
01111	MIN	11111	reserved

Table A.2: Opcodes

The encoding instruction in configuration memory is shown in Table A.3.

Bit position	Function
0	Left operand A bus position (OpA[15:0]->Lin[31:16]/Lin[15:0])
1	Right operand B bus position (OpB[15:0]->Rin[31:16]/Rin[15:0])
3 - 2	Right input select 00 :GIFU, 01:ACC, 10 :LIFU, 11 : OpB
8 - 4	Opcode .. see Table A.2
13 - 9	SHACC shift distance
14	SHACC shifter direction (1=left, 0=right)
16 - 15	SHACC modification 00 : none, 01 : invert bit [31:0], 10 : invert bit [39:32], 11 : illegal
18 - 17	ACC shifter control 00 : no shift, 01 : shift left, 10 : shift right, 11 : conditional shift
21 - 19	Write back source and ACC shift/limiter output 000 : OpB, 001 : ACC[15:0], 010 : ACC[31:16], 011 : ACC[39:32], 100 : bus not in use, 101 : GIFU, 110 : LIFU, 111 : illegal
23 - 22	ACCWR source: 00 : GIFU, 01 : OpB, 10 : ACC, 11 : SHACC
25 - 24	ACCWR destination: 00 : ACC A, 01 : ACC B, 10 : ACC C, 11 : ACC D
27 - 26	Arithmetic destination: 00 : ACC A, 01 : ACC B, 10 : ACC C, 11 : ACC D
29 - 28	ACC source: 00 : ACC A, 01 : ACC B, 10 : ACC C, 11 : ACC D
31 - 30	SHACC source: 00 : ACC A, 01 : ACC B, 10 : ACC C, 11 : ACC D

Table A.3: Configuration instruction specification

The complete chip pad pins are given in Table A.4.

Signal	I/O	Pad name	Supply	Description
gnd	-	VSSIOCO, VSSCO	0.0V	Ground for PAD ring and CORE
vdd3io	-	VDD3IOCO	3.3V	Pad ring 3.3V supply
vdd	-	VDDIOCO	1.8V	1.8V pad ring and RAM supply
vddfu	-	VDDCO	1.8V	functional unit supply
vddtune	-	VDDCO	1.8V	tunable delay supply
ClkBootFU3	i/p	TLCHTD_TC	-	FU3 scan clock (strobe data into RAMs)
ScaninFU3	i/p	TLCHTD_TC	-	FU3 scan in
ScanoutFU1	o/p	B4TR_TC	-	FU1 scan out
Oin31	o/p	B4TR_TC	-	Observation pin (input of data path bit 31)
Or39	o/p	B4TR_TC	-	Observation pin (ALU output bit 39, sign-bit)
Or31	o/p	B4TR_TC	-	Observation pin (ALU output bit 31)
OC	o/p	B4TR_TC	-	Observation pin (carry out)
ScanoutFU3	o/p	B4TR_TC	-	FU3 scan out
ScaninFU1	i/p	TLCHTD_TC	-	FU1 scan in
ClkBootFU1	i/p	TLCHTD_TC	-	FU1 scan clock (strobe data into RAMs)
REQ	o/p	B4TR_TC	-	Observation pin (request)
ClkBootAdr	i/p	TLCHTD_TC	-	Top level RAMs clock (strobe data into RAMs)
ScaninAdr	i/p	TLCHTD_TC	-	Top level RAMs scan in
ScanoutAdr	o/p	B4TR_TC	-	Top level RAMs scan out
Gclk	i/p	TLCHTD_TC	-	Global clock for shifting the scan path data
nreset	i/p	TLCHTD_TC	-	reset pin
mx	i/p	TLCHTD_TC	-	mode pin to select between test and normal mode
HALT	i/p	TLCHTD_TC	-	halt pin
ScaninFU0	i/p	TLCHTD_TC	-	FU0 scan in
ClkBootFU0	i/p	TLCHTD_TC	-	FU0 clock (strobe data into RAM)
ScanoutFU0	o/p	B4TR_TC	-	FU0 scan out
ScanoutFU2	o/p	B4TR_TC	-	FU2 scan out
req_mac	o/p	B4TR_TC	-	Observation pin (request signal to FU)
done_fu	o/p	B4TR_TC	-	Observation pin to signal that all 4 FUs have completed their current instruction
ClkBootFU2	i/p	TLCHTD_TC	-	FU2 clock (strobe data into RAM)
ScaninFU2	i/p	TLCHTD_TC	-	FU2 scan in

Table A.4: CADRE-s chip pad pin (68-pin PGA package)

Appendix B: Glossary

ADPCM: Adaptive Differential Pulse Code Modulation

ALU: Arithmetic and logic unit

CADRE: Configurable Asynchronous DSP for Reduced Energy

CADRE-s: CADRE successor

DCT: Discrete Cosine Transform

DSP: Digital signal processing

EDA: Electronic design automation

FIR: Finite Impulse Response

FU: Functional unit

MIMD: Multiple Instruction Multiple Data

NORM: Normalization

PC: Partial carry

PCM: Pulse Codec Modulation

PTG: pass transmission gate

PS: Partial sum

SIMD: Single Instruction Multiple Data

SPL: Single-ended pass transistor logic

VCS: Verilog co-simulation

Appendix C: Simulation Details

This appendix concerns the generation of input test vectors and the simulation of the layout of five different 40-bit adder configuration using the Nanosim simulator[150]. It details how the test vectors were generated by mean of a C program and how to produce the timing and power measurements to obtain the results in Table 4.1.

1. Generating the input test vectors

A sequence of 40-bit uniformly distributed pseudo random numbers were produced by mean of the 'rand' function in a C program. The numbers were then converted and stored as hexadecimal numbers in a text file readable by the Verilog[155] test bench. These numbers are delivered by the test bench directly to the Nanosim simulator. A data set of pseudo random 100 numbers were used in the experiment to evaluate the five 40-bit adder simulations. The simulation speed depends on the complexity of the design and the number of test inputs. Processing 100 data inputs takes in the order of 6 hours on a Sun machine. Therefore, 100 numbers was considered to be a reasonable quantity for providing the results shown in Table 4.1. In addition, a experiment carried out by the author, it was found a variation of the average current of as estimated from 100 additions and 1,000 additions was less than 1%. We therefore believe that the error incurred by restoring the sequence length to 100 per number's of the error of 1%.

2. Generating the post layout netlist of the 40-bit adder

The layouts of five 40-bit adders were placed and routed using Cadence's Encounter tool. The post layout netlists (called SPICE netlists) including the load capacitance information of these five 40-bit adders were generated by Diva in the Cadence framework[149].

3. Preparing the Verilog for simulation

A Verilog module was written by the author separate to be two parts. The first part is a test bench which was used to read the numbers. The second part is a top level module which was used to connect the test bench and post layout netlists (SPICE netlists) of 40-bit adder. This Verilog module can be recognized by Nanosim.

4. Getting the timing and power results

As soon as the Verilog test bench provides the test inputs, Nanosim starts the simulation. The average current can be calculated by specifying 10ns as a start and allowing time for all 100 instructions to complete, making a final time of 1,000ns. This allows the designer to ignore the initial time at the beginning of the simulation. Meanwhile, the timing information is produced in the trace file (the output file of Nanosim) using the command 'print node logic *'. From the experiment, the average delay for the addition is measured and is shown in Table 4.1. The average power is calculated as the product of the average current and V_{DD}(1.8V).