

ULRR

Digital signal processor based controller for multi-rail DC-DC convertor systems

Item Type	Thesis
Authors	Mooney, James
Download date	2026-05-20 13:47:22
Item License	https://creativecommons.org/licenses/by-nc-sa/1.0/
Link to Item	https://hdl.handle.net/10344/1680



Digital Signal Processor Based Controller for Multi-Rail DC-DC Converter Systems

Submitted to

Department of Electronic and Computer Engineering
Faculty of Science and Engineering
University of Limerick

For the degree of

Doctor of Philosophy

By

James Mooney

Supervisors:

Dr. Abdulhussain Mahdi

Dr. Mark Halton

Submitted to the University of Limerick,
April 2011

Abstract

Digital Signal Processor Based Controller for Multi-Rail DC-DC Converter Systems

James Mooney

The widespread adoption of digital control has resulted in many improvements in low power switching mode power converters. Digital control has introduced alternative control laws and new features to previously analogue-controlled converters. It has also enhanced performance and cost factors in particular applications. This is evident in systems comprising of multiple ASICs, DSPs, memory devices, FPGAs, etc., which require multiple DC-DC converters. A single digital controller can perform the control for all of the DC-DC converters even though the specifications of each may be different. This thesis addresses fundamental issues in the architectural design and hardware implementation of a digital controller for such multi-rail systems.

A review of conventional digital controllers indicates that they are not specifically optimised for multi-rail applications. In some cases existing digital signal processors are used, which have superfluous features to those required for the simple control task. In other cases fixed-algorithm controllers are used, which are unsuitable for applications having multiple converters with individual control algorithm requirements.

A new digital signal processor core that overcomes the shortcomings of existing controllers has thus been designed, implemented and evaluated. The requirements of computational, memory and program control elements have been analysed in relation to the algorithms to be executed and the power converter system to be controlled, where linear compensators for multi-rail DC-DC converter systems are considered in particular. This analysis has led to the design of an instruction set and a corresponding dual-datapath architecture. The proposed processor has been implemented using an FPGA and verified in a closed-loop power converter system. The benefits of the specialised processor are illustrated in the thesis through a comparison with a conventional single datapath processor. Experimental results demonstrate improved performance over existing digital signal processor based controllers when controlling multiple DC-DC converters. In the case of multiple converters that have a non-integer switching frequency ratio more significant performance improvements are exhibited due to the processor's novel interrupt controller.

Declaration

I hereby declare that this thesis is entirely my own work and does not contain material previously published by any other author, except where due reference or acknowledgement has been made. Furthermore, I declare that it has not been submitted to any other university or higher education institution, or for any other academic award in this university.

James Mooney

April 2011

Acknowledgements

I would like to acknowledge the invaluable role my supervisors, Dr. Hussain Mahdi and Dr. Mark Halton have played in assisting me in completing this research and thesis. They have both contributed immensely to my development and have allowed me to attain my goals by generously sharing their time and knowledge. I am also grateful to Dr. Karl Rinne and Dr. Anthony Kelly who initially introduced me to the world of digital power control.

I wish to thank all those friends and colleagues in the Circuits and Systems Research Centre, too numerous to mention, who assisted me in any way during my time there. Special thanks to Dr. Simon Effler for providing the power system PCB and DPWM for the research and for his assistance in completing the experimental work. Thanks also to Simon, Martin and Vincent for their fruitful discussions on both “digital power control” related and non-“digital power control” related topics.

I would like to acknowledge the financial support provided by the Irish Research Council for Science Engineering and Technology: funded by the National Development Plan during the course of this work.

The kindness and patience of my family and friends throughout the entire duration of the work is very much appreciated. Thanks to my sister, Janice for her endless guidance and inspiration. Thank you especially to Michelle, for persevering with me through it all.

Finally, I would like to dedicate this thesis to my parents in recognition of their infinite support and encouragement.

Contents

Abstract.....	iii
Declaration.....	iv
Acknowledgements.....	v
List of Figures.....	xi
List of Tables	xv
List of Acronyms and Abbreviations	xvi
1 Introduction	1
1.1 DSP-Based Control – Context in Digital Power Conversion.....	1
1.2 Thesis Objectives	3
1.3 Research Methodology	4
1.4 Thesis Contribution.....	5
1.5 Thesis Overview	7
2 Digital Controller Implementations for SMPC Applications	9
2.1 Introduction.....	9
2.2 Digital Control System Overview	10
2.2.1 Voltage Mode Control	11
2.2.2 Current Mode Control.....	12
2.2.3 Analogue to Digital Conversion	13
2.2.4 Digital Pulse Width Modulation	14
2.3 Multi-Rail Point-of-Load Converter Applications.....	15
2.4 Digital Voltage Mode Control Algorithms	19
2.4.1 Linear control.....	19
2.4.2 Adaptive control.....	22
2.4.3 Multi-mode control	22

2.5	Digital Controller Implementations	23
2.5.1	Dedicated Hardware Controllers	24
2.5.2	Digital Signal Processor Based Controllers	29
2.6	Summary	34
3	Analysis of DSP Core Architectures.....	35
3.1	Introduction	35
3.2	Potential Architectures	37
3.2.1	Multiple-Issue Processor Architectures	38
3.2.2	Multiprocessor Architectures	40
3.2.3	Single Instruction Multiple Data Architectures	41
3.3	Algorithm Analysis	43
3.3.1	Linear Control Algorithm	44
3.3.2	Adaptive Control Algorithm	51
3.4	Architecture Performance Comparisons	58
3.4.1	Execution Times for Multiple Rails	59
3.4.2	Performance over a Range of Switching Frequencies	62
3.4.3	Algorithm Comparisons	65
3.5	Summary	66
4	A Dual-MAC DSP Core Architecture	67
4.1	Introduction	67
4.2	Processor Architecture	69
4.2.1	Processor Core Overview	69
4.2.2	Instruction Set	70
4.2.3	I/O Interface and Peripherals	73
4.3	Datapath	74
4.3.1	Functional Units	76
4.3.2	Number Representation and Arithmetic Issues	77
4.4	Memory	80
4.4.1	Register File Memory	80
4.4.2	Addressing.....	83

4.5	Program Controller	84
4.5.1	Execution Control	84
4.5.2	Interrupt Control	88
4.5.3	Instruction Encoding	90
4.5.4	Pipelining	91
4.6	Summary	93
5	Implementation of DSP Core and Application to Multi-Rail SMPC.....	95
5.1	Introduction	95
5.2	Simulation-Based Verification.....	96
5.3	FPGA Synthesis	100
5.3.1	Synthesis Platform	100
5.3.2	Hardware Cost.....	102
5.4	Comparison with Alternative Architectures.....	104
5.4.1	Single MAC Architecture	104
5.4.2	Hardware Cost Comparison	107
5.5	Experimental Platform	109
5.5.1	Digital Hardware.....	109
5.5.2	SMPS System.....	110
5.5.3	DSP Core Programming.....	112
5.6	Multi-Rail Voltage Mode Control.....	115
5.6.1	Multiple Control Algorithm Execution.....	116
5.6.2	Impact on DC-DC Converter Performance.....	118
5.7	Summary	122
6	Control of SMPC Systems with Non-Integer Switching Frequency Ratios..	123
6.1	Introduction	123
6.2	Interrupt Triggered Control.....	124
6.2.1	Overview	124
6.2.2	Delays in Non-Integer Switching Frequency Ratio Systems.....	125
6.2.3	Program Code Priority	129
6.2.4	Maximum ADC-Sample to Duty-Cycle-Update Delay	131

Contents

6.3	Modified Interrupt Controller	132
6.3.1	Automatically Enabled Interrupts	132
6.3.2	Modified Interrupt Controller Hardware.....	135
6.3.3	Comparison With Standard Interrupt Controllers	137
6.4	Experimental Verification	139
6.4.1	Experimental Platform	139
6.4.2	Interrupt Controller Operation	141
6.4.3	Performance Improvements	143
6.5	Summary	145
7	Conclusions	147
7.1	Summary and Achievements.....	147
7.2	Future work	149
7.2.1	Potential Improvements to the DSP Core	150
7.2.2	Application and System Level Considerations	153
7.3	Concluding Remarks	154
	Bibliography	155
	Appendix A. Instruction Set	165
	Appendix B. Assembly Code	169

List of Figures

Figure 2.1. Digital voltage mode control of buck converter.	11
Figure 2.2. Digital current mode control of buck converter.	13
Figure 2.3. Comparison of pulse width modulation methods over three switching cycles.	15
Figure 2.4. Intermediate bus architecture with multiple voltage rails.	16
Figure 2.5. Digital compensator applied to multiple DC-DC converters.	17
Figure 2.6. Analogue compensators applied to multiple DC-DC converters.	17
Figure 2.7. Multi-phase buck converter.	18
Figure 2.8. Multi-mode digital controller optimised for fast transient response.	23
Figure 2.9. Typical digital power controller components.	24
Figure 2.10. Look-up table based PID compensator.	26
Figure 2.11. Parallel architecture dedicated to implementing 2P2Z algorithm for controlling multiple POL converters.	27
Figure 2.12. Multiplexed architecture dedicated to implementing 2P2Z algorithm for controlling multiple POL converters.	28
Figure 2.13. Interrupt-based execution of control algorithm using a digital signal processor.	30
Figure 2.14. Resource utilisation of DSP controlling dual rail power supply system.	30
Figure 2.15. PWM and inductor current waveforms for buck converter.	33
Figure 3.1. Typical computational elements in the datapath of a DSP.	37
Figure 3.2. Multiple-issue architecture for controlling multiple DC-DC converters.	39
Figure 3.3. Multicore architecture for controlling multiple DC-DC converters.	41
Figure 3.4. SIMD architecture for controlling multiple DC-DC converters.	42
Figure 3.5. Block diagram of direct form realisation of 2P2Z algorithm.	45
Figure 3.6. Ordering of 2P2Z algorithm operations to minimise duty cycle calculation latency.	46
Figure 3.7. Block diagram of pre-calculated 2P2Z algorithm.	47
Figure 3.8. Precedence graph of pre-calculated 2P2Z algorithm.	48
Figure 3.9. Scheduling of 2P2Z algorithm operations on single MAC architecture.	49
Figure 3.10. Scheduling of 2P2Z algorithm operations on dual MAC architecture.	50

List of Figures

Figure 3.11. Block diagram of adaptive algorithm.	53
Figure 3.12. Precedence graph of adaptive algorithm.....	55
Figure 3.13. Scheduling of adaptive algorithm operations on single MAC DSP architecture.....	56
Figure 3.14. Scheduling of adaptive algorithm operations on dual MAC DSP architecture.....	57
Figure 3.15. ASAP scheduling of adaptive algorithm operations.....	58
Figure 3.16. Execution time of 2P2Z algorithm on range of processor architectures for up to 10 rails.	61
Figure 3.17. Execution time of adaptive algorithm on range of processor architectures for up to 10 rails.	62
Figure 3.18. Maximum switching frequency controllable by range of processor architectures using 2P2Z algorithm for up to 10 rails.....	63
Figure 3.19. Maximum switching frequency controllable by range of processor architectures using adaptive algorithm for up to 10 rails.....	64
Figure 3.20. Maximum switching frequency controllable by single MAC, dual MAC and dual core architectures using 2P2Z and adaptive algorithms for up to 10 rails.....	65
Figure 4.1. Main DSP core hardware elements.....	70
Figure 4.2. DSP core with I/O interface elements.	74
Figure 4.3. Main datapath elements.	75
Figure 4.4. Wordlengths of main datapath buses.....	78
Figure 4.5. Memory map of multiple register bank implementation.	82
Figure 4.6. Main program controller elements.....	85
Figure 4.7. DSP program code structure for controlling three power converters.....	87
Figure 4.8. Interrupt controller finite state machine.	89
Figure 4.9. Encoding for standard instruction with two sub-instructions.....	90
Figure 4.10. Encoding for double-length instruction.....	90
Figure 4.11. DSP core pipeline stages.	92
Figure 4.12. JMP instruction in DSP core instruction pipelines.....	92
Figure 4.13. Effects of an interrupt on DSP core instruction pipelines.	93
Figure 5.1. Hierarchical structure of Verilog modules of DSP core.....	97

List of Figures

Figure 5.2. Simulink/ModelSim co-simulation configuration.	97
Figure 5.3. Simulink/ModelSim closed-loop co-simulation of dual MAC DSP core - buck converter output voltage response.	98
Figure 5.4. DSP control and data signals from ModelSim simulation of three-pole, three-zero control algorithm.	99
Figure 5.5. Critical path of dual MAC DSP core.	101
Figure 5.6. Contribution of main elements to total dual MAC DSP core logic element usage.	103
Figure 5.7. Main datapath elements of single MAC DSP core.	105
Figure 5.8. Contribution of main elements to total single MAC DSP core logic element usage.	106
Figure 5.9. Experimental platform for prototyping of DSP core.	109
Figure 5.10. Digital hardware units implemented on FPGA.	110
Figure 5.11. Custom PCB featuring multi-rail SMPC system.	112
Figure 5.12. Program code assembly and DSP core programming.	114
Figure 5.13. Assembly code for 3P3Z algorithm on dual MAC DSP.	115
Figure 5.14. Single MAC DSP core executing three control algorithms and background code for three independent buck converters.	117
Figure 5.15. Dual MAC DSP core executing three control algorithms and background code for three independent buck converters.	117
Figure 5.16. Duty cycle calculation instant during execution of control algorithm by single MAC DSP.	119
Figure 5.17. Duty cycle calculation instant during execution of control algorithm by dual MAC DSP.	119
Figure 5.18. Single MAC DSP core regulating output voltage of buck converter to 1.5 V in presence of load current step.	121
Figure 5.19. Dual MAC DSP core regulating output voltage of buck converter to 1.5 V in presence of load current step.	121
Figure 6.1. Interleaved scheduling for interrupts with integer multiple frequency ratio.	125
Figure 6.2. Delayed scheduling for overlapping interrupts with non-integer multiple frequency ratio.	126

List of Figures

Figure 6.3. Delayed scheduling for simultaneous interrupts with non-integer multiple frequency ratio.....	127
Figure 6.4. Application of duty cycle from previous cycle caused by delay in duty cycle update.....	127
Figure 6.5. Duty cycle updated just in time using maximum fixed delay.	128
Figure 6.6. Duty cycle updated well in advance of application time using maximum fixed delay.....	128
Figure 6.7. Priority levels of code executed by DSP-based digital power controller.	130
Figure 6.8. DSP delay with standard interrupt control.....	131
Figure 6.9. DSP delay with modified interrupt control.....	132
Figure 6.10. Flowchart of modified interrupt controller behaviour.	134
Figure 6.11. Modified interrupt controller with additional hardware blocks highlighted.....	136
Figure 6.12. Comparison of modified interrupt control method with a standard method involving manually disabling and enabling interrupts.	138
Figure 6.13. Comparison of modified interrupt control method with a standard method involving separate interrupt service routines for the duty cycle calculation and pre-calculation code sections.....	138
Figure 6.14. Experimental platform consisting of three rail power converter system and interrupt controller implemented as part of the dual MAC DSP core on a Cyclone II FPGA.	140
Figure 6.15. Standard interrupt control operation (Horizontal axis: 500 ns/division).	142
Figure 6.16. Modified interrupt control operation (Horizontal axis: 500 ns/division).	142
Figure 6.17. Load transient response for standard interrupt controller.....	144
Figure 6.18. Load transient response for modified interrupt controller with wider bandwidth compensator.....	144
Figure 7.1. Register access possibilities for dual datapath processor with four bank register file.....	152

List of Tables

Table 3.1. Classification of adaptive algorithm computational operations.....	54
Table 3.2. Minimum clock cycles required to execute 2P2Z and adaptive algorithms on single MAC and multiple datapath architectures.	60
Table 3.3. Comparison of execution times (in terms of DSP clock cycles) required to implement 2P2Z algorithm on a range of processors for single and multi-rail systems.....	60
Table 3.4. Comparison of execution times (in terms of DSP clock cycles) required to implement adaptive algorithm on a range of processors for single and multi-rail systems.	61
Table 3.5. Comparison of 2P2Z and adaptive algorithm execution times (in terms of DSP clock cycles) on single MAC, dual MAC and dual core architectures.	65
Table 4.1. DSP core instructions.....	71
Table 4.2. DSP core special function registers.	86
Table 5.1. Delay contributions of datapath elements to critical path.....	100
Table 5.2. FPGA resource utilisation for synthesised dual MAC DSP core.....	103
Table 5.3. FPGA resource utilisation for synthesised single MAC DSP core.....	106
Table 5.4. Resource utilisation comparison of single MAC, dual MAC and dual core architectures.	107
Table 5.5. Additional resources required for dual MAC DSP compared with single MAC DSP.	108
Table 5.6. Additional resources required for dual core DSP compared with dual MAC DSP.	108
Table 5.7. Parameters of buck converters in experimental platform.	111
Table 6.1. Digital control system delay parameters.....	141
Table 6.2. Maximum ADC-sample to duty-cycle-update delays for 3-rail power converter system.....	141
Table 6.3. Maximum ADC-sample to duty-cycle-update delays for up to 8 rails....	141

List of Acronyms and Abbreviations

2P2Z	Two-Pole Two-Zero
3P3Z	Three-Pole Three-Zero
ACC	Accumulator
ADC	Analogue to Digital Converter
ALU	Arithmetic Logic Unit
ASAP	As Soon As Possible
ASIC	Application Specific Integrated Circuit
BG	Background
CCM	Continuous Conduction Mode
CLA	Control Law Accelerator
CMC	Current Mode Control
CPU	Central Processing Unit
DAC	Digital to Analogue Converter
DCM	Discontinuous Conduction Mode
DPWM	Digital Pulse Width Modulator
DSP	Digital Signal Processor
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HDL	Hardware Description Language
IBA	Intermediate Bus Architecture
IIR	Infinite Impulse Response
IC	Integrated Circuit
I²C	Inter-Integrated Circuit
ISR	Interrupt Service Routine
LDO	Low Drop Out
LE	Logic Element
LSB	Least Significant Bit

List of Acronyms and Abbreviations

LUT	Look-Up Table
MAC	Multiplier Accumulator
MIMD	Multiple Instruction Multiple Data
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MSB	Most Significant Bit
OSC	Oscillator
PCB	Printed Circuit Board
PID	Proportional Integral Derivative
PLL	Phase-Locked Loop
POL	Point of Load
POR	Power-On Reset
PMBus	Power Management Bus
PWM	Pulse Width Modulation
RETI	Return from Interrupt
RISC	Reduced Instruction Set Computer
RTL	Register Transfer Level
SIMD	Single Instruction Multiple Data
SMBus	System Management Bus
SMPC	Switching Mode Power Converter
SMPS	Switching Mode Power Supply
SoC	System-on-Chip
UART	Universal Asynchronous Receiver Transmitter
VLIW	Very Long Instruction Word
VMC	Voltage Mode Control
VREF	Voltage Reference
VRM	Voltage Regulator Module

Chapter 1

Introduction

1.1 DSP-Based Control – Context in Digital Power Conversion

Digital circuits have long formed part of various switching mode power converter (SMPC) systems, implementing numerous functions in order to assist in the power conversion process. The reduced cost of digital circuitry has allowed the compensator of the power converter to be implemented in the form of a single digital integrated circuit (IC), thus realising increased reliability and reduced cost compared with compensators with multiple analogue components.

In this work the implementation of a digital compensator is addressed for SMPC applications with fast transient performance demands. In particular, the focus is on the controller requirements for point-of-load (POL) converter systems. The defining characteristic of POL systems is that the power converter is located close to the load in order to maintain tight voltage regulation in the presence of fast dynamic changes in the load current. Typical examples of the loads are ICs that require multiple voltage rails and sink large currents. These range from field programmable gate arrays (FPGAs), high-end digital signal processors (DSPs) and memory devices to application specific integrated circuits (ASICs), graphics processing units (GPUs) and microprocessors. POL converters are usually non-isolated step-down converters, i.e.

they convert an input DC voltage to a lower output DC voltage. To meet the high current requirements in an efficient manner the multi-phase buck converter architecture is often utilised.

Early digital compensators applied to POL converters sought to emulate their analogue predecessors by implementing proportional-integral-derivative (PID) control algorithms [1, 2]. In addition to achieving accurate voltage regulation with these simple digital compensators, recent advances in digital implementation have enabled the use of alternative compensators, whose features surpass those offered by analogue solutions [3]. These include the implementation of compensators with complex conjugate zeroes that yield improved power converter performance and adaptive algorithms that can instantly modify the compensator's control law to accurately regulate the output voltage when conditions in the power converter and load change [4-7]. Such adaptive algorithms can achieve high power conversion efficiency over a wide load current range or maintain accurate voltage regulation in spite of the effects of aging of the power converter's components over time [8-10].

The hardware implementation of the digital control algorithm can take many forms. Approaches include the use of dedicated logic, look-up tables (LUTs), finite state machines (FSMs), DSPs or microcontroller architectures. LUT- and FSM-based compensators serve their purpose well in low-cost switching mode power supplies (SMPS), which have relatively undemanding power conversion requirements [11-13]. Higher performance converters have more complex specifications, which necessitate a more flexible compensator approach. Digital signal processors are suitable for these specific applications because they provide sufficient computational functionality in addition to compensator flexibility through their ability to be programmed to execute custom control algorithms. In addition to their computational features, general purpose DSPs usually contain other hardware that is required in the digitally controlled SMPS system, for example, pulse width modulators.

While most existing DSPs perform their role in the digital power control process satisfactorily, many of them are not strictly optimised for the needs of the SMPCs to

which they are being applied. Aside from the fact that these DSPs contain superfluous peripheral hardware, their datapaths are unable to execute multiple complex control algorithms within the time frame dictated by the multi-rail POL converter application. The growing complexity of the digital control algorithms being applied has meant that the existing DSPs [14-17], which were originally designed for implementing digital filtering and Fast Fourier Transform (FFT)-type algorithms, are struggling to meet the constraints of digital power control. The supplementing of existing DSPs and microcontrollers with hardware accelerators and co-processors has been proposed in an attempt to hastily provide a solution to the rapid development of a wide range of advanced power control techniques [18, 19]. Although such architectures can temporarily address deficiencies in existing processors, they do not take into account the idiosyncrasies of digital power control systems at a basic level.

There is a need therefore to examine evolving digital control techniques to identify the most suitable DSP architecture that can take account of constraints in the application system in order to yield improved regulation and performance in current and future switching mode power supplies. With recent emphasis on maximising power conversion efficiency, it is imperative that appropriate features can be incorporated into the computational hardware so that significant benefits can be attained at the system level. This thesis therefore investigates, proposes and develops a DSP core architecture that can act as an exemplary template for the computational engine of future intelligent SMPS systems.

1.2 Thesis Objectives

As explained in the previous section, existing DSPs do not optimally meet the digital control requirements of modern POL converter applications. This thesis therefore aims to investigate more suitable architectures based on a thorough analysis of the computational requirements of typical power control algorithms. The constituent components of the most suitable architecture should be selected and customised, if necessary, in accordance with the unique demands of the power converter system,

whereby all choices and trade-offs are fully justifiable in the end-application. Furthermore the thesis aims to demonstrate the advantages of a new flexible processor core architecture when implemented as the digital controller in a relevant power conversion system. The proposed processor should exceed the computational performance of similar existing power control specialised DSPs, when executing control algorithms for multi-rail DC-DC converter systems.

1.3 Research Methodology

In order to achieve the objectives outlined above, the following steps were taken:

- Review of the state of the art in the area of digital control of SMPCs
- Investigation of multi-rail POL converter systems, in particular their typical specifications and requirements
- Characterisation of existing digital controller architectures, identifying drawbacks and shortcomings
- Review of alternative DSP-based digital controller architectures
- Comprehensive study of the computational requirements of common digital control algorithms
- Comparative analysis of existing and alternative DSP-based architectures based on suitability for multi-rail POL converter applications and on computational performance when executing digital control algorithms
- Identification of the optimal architecture for SMPC control requirements based on comparative analysis

- Investigation of architectural features of selected processor including datapath, memory and program controller requirements based on existing techniques and closed-loop digital control system demands
- Specification and register transfer level (RTL) implementation of proposed processor using Verilog hardware description language
- Simulation of Verilog-specified processor architecture in closed-loop power converter system with accurate power converter, analogue to digital converter and digital pulse width modulator models
- Integration of DSP core with other digital control system hardware, synthesis and FPGA implementation
- Experimental verification and evaluation of DSP core applied to multi-rail power converter system
- Demonstration of improved power converter performance provided by proposed DSP core compared with existing methods under varying load conditions

1.4 Thesis Contribution

The content of this thesis represents a significant contribution to the area of digital control of DC-DC converters. The key achievements of the work are described here.

The initial analyses and exploratory work presented in the thesis provide the reader with an in-depth understanding of the process involved in defining and characterising a DSP core architecture for digital power control applications.

Applying the knowledge developed through this investigation resulted in the creation of an optimised digital signal processor architecture for executing control algorithms in multi-rail POL converter applications.

The two-way, very long instruction word, dual multiplier-accumulator (MAC) architecture was identified as the architecture that could best serve the computational requirements of the target application.

A custom instruction set, an optimised datapath, a specialised register file memory and a context switching interrupt controller were developed for the power control specific dual MAC processor.

A particularly notable adaptation of the processor's interrupt controller permitted application of the processor to be extended to multi-rail power converter systems where the converter switching frequencies are in a non-integer ratio.

Although the operation of the programmable DSP core was validated through simulation, it was also implemented on FPGA in order to demonstrate its capabilities in executing custom control algorithms for the power conversion application. The experimental results obtained using this FPGA implementation confirmed the significant performance advantages of the proposed architecture when executing multiple control algorithms compared with conventional digital controller architectures.

The following technical papers were published based on the work contained in this thesis:

J. Mooney, S. Effler, M. Halton, and A. E. Mahdi, "Interrupt controller for DSP-based control of multi-rail DC-DC converters with non-integer switching frequency ratio", in IEEE International Conference on Electronics, Circuits and Systems, 2010, pp. 1211-1214.

J. Mooney, S. Effler, M. Halton, and A. E. Mahdi, “DSP-based control of multi-rail DC-DC converter systems with non-integer switching frequency ratios”, in IEEE International Conference on Energy, Power and Control, 2010, pp. 200-204.

J. Mooney, M. Halton, and A. E. Mahdi, “Specialized Digital Signal Processor for control of multi-rail/multi-phase high switching frequency power converters”, in IEEE Applied Power Electronics Conference, 2010, pp. 2207-2211.

J. Mooney, A. E. Mahdi, A. Kelly, and K. Rinne, “DSP-based controller for multi-output/multi-phase high switching frequency DC-DC converters”, in IEEE Workshop on Control and Modeling for Power Electronics, 2008.

J. Mooney, A. E. Mahdi, A. Kelly, and K. Rinne, “Dual MAC processor for adaptive control of multiple high switching frequency DC-DC converters”, in IET Irish Signals and Systems Conference, 2008, pp. 116-121. (Best Paper Award Winner)

1.5 Thesis Overview

This thesis describes the design, development and implementation of the dual MAC DSP core as a digital controller for multi-rail DC-DC conversion applications. Chapter 2 provides a review of the characteristics of the application system, in addition to examining the benefits and pitfalls of previous and existing digital controller implementations. Chapter 3 investigates the performance of a number of potential digital controller architectures in a more quantitative manner, through a detailed analysis of the implementation of typical power control algorithms on those architectures. Chapter 4 describes the architectural features of the proposed dual MAC DSP core, which are particularly targeted towards the execution of multiple digital control algorithms for multi-rail systems. Hardware implementation of the architecture is considered in Chapter 5 and an evaluation of its performance when applied to a multi-rail power conversion system is also presented. Chapter 6 focuses on multi-rail system applications where there is a non-integer ratio between the switching

frequencies of the power converters and proposes a modified DSP-based controller solution that provides multiple benefits in such applications. Chapter 7 provides a summary of the thesis and highlights the main achievements of the work. Potential future research avenues are also proposed.

Appendix A concerns details of the instruction set that is implemented by the dual datapath processor. The reader is encouraged to refer to Appendix A for an explanation of the instruction set mnemonics that are referenced frequently throughout the text. Appendix B presents the assembly language program that was used to program the dual MAC DSP in verifying its operation in the multi-rail system application.

Chapter 2

Digital Controller Implementations for SMPC Applications

2.1 Introduction

This chapter introduces the theory and current implementation practice of digitally controlled multi-rail DC-DC converter systems. The trend towards lower cost digital hardware has allowed digital controllers to compete with analogue controllers in low power SMPC applications. Following the successful emulation of analogue controllers using digital hardware, digital controllers have been implemented that surpass their analogue counterparts in terms of functionality and performance. Digital controllers have not completely replaced analogue controllers in the SMPC domain but rather have found their niche in certain applications where digital control is the more suitable choice. One such application is the multi-rail DC-DC converter system, which will be introduced in this chapter.

Digital controllers have brought improved performance to switching mode power supplies through the implementation of complex algorithms, which could not be practically implemented using analogue methods. While it is possible to implement these complex algorithms using a variety of digital implementation methods, the cost of the chosen implementation method is critical in the typically cost-sensitive SMPC

application; where cost in this case explicitly refers to the cost of the digital compensator implementation. It is acknowledged that other elements of the SMPC such as the output filter components also contribute significantly to the cost of the switching mode power supply, especially in multi-rail systems where multiple inductors and capacitors are required. The standard method of reducing the size and thus the cost of these components is to increase the switching frequency of the power converters [20]. Indeed, one of the primary objectives of this work is to accommodate the use of higher switching frequencies by increasing the execution speed of the control algorithms by the digital controller. This chapter examines existing digital controller implementations with a particular focus on their suitability for executing multiple algorithms for controlling multi-rail power converter systems.

The chapter has the following layout. An overview of a basic digital power control system is presented first, where the role of the digital controller is examined. Aspects of the other control system elements that are relevant to the implementation of the digital controller are also discussed. This is followed by an introduction to the multi-rail DC-DC converter system. The next section describes the algorithms and control techniques that a digital controller is expected to perform, based on the direction of recent research and market trends in digital power control. The final section is devoted to the implementation of digital controllers and includes a review of the state of the art.

2.2 Digital Control System Overview

The concept of digital power control is depicted in Figure 2.1 where the switching mode power converter (SMPC) being controlled is a single phase non-isolated synchronous buck converter. The analogue to digital converter (ADC), digital compensator and digital pulse width modulator (DPWM) hardware blocks add digital control functionality to the DC-DC converter. Figure 2.1 illustrates digital voltage mode control (VMC), which is the most common digital control scheme for SMPCs. It should be noted that a practical implementation would require additional components to those illustrated in the simplified circuit of Figure 2.1.

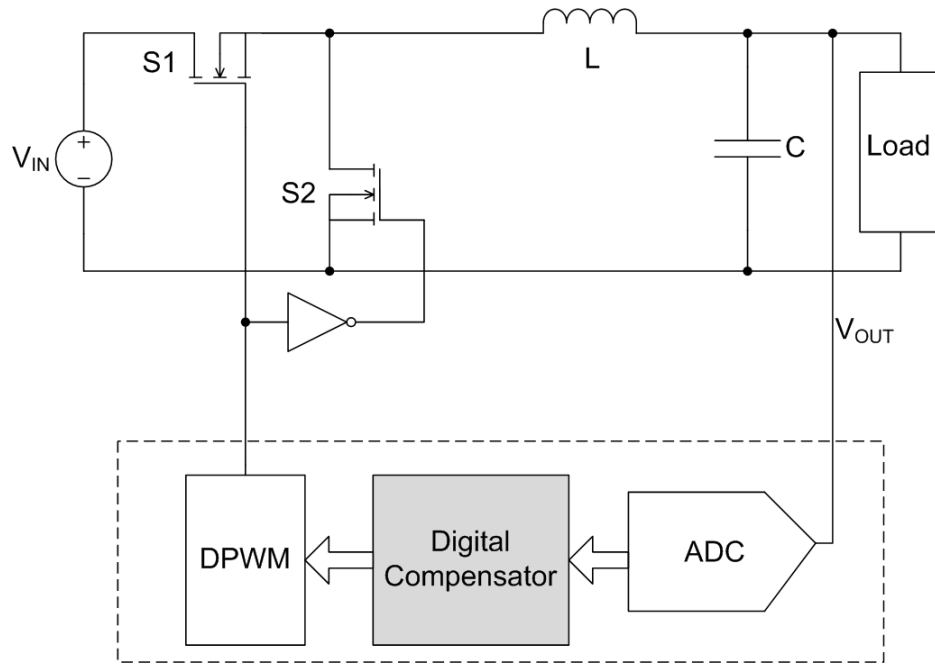


Figure 2.1. Digital voltage mode control of buck converter.

Although the non-isolated synchronous buck converter is the only power converter type that is referred to in this text, it is likely that much of the work is applicable to systems that contain other power converter types, for example boost, buck-boost and their isolated versions. This is evident from the similar types of control algorithms that are implemented by digital controllers for each of these power converters [21-23].

2.2.1 Voltage Mode Control

In voltage mode control (VMC) the ADC samples the output voltage of the power converter and converts it to a digital value. The digital output voltage is subtracted from a reference value, which represents the required output voltage, to generate a digital error value. This error value is passed to the compensator, which implements the control law for regulation of the output voltage. The compensator produces a new digital duty cycle output value every switching cycle, based on the deviation of the output voltage from the reference value. The DPWM creates on-off signals according to the duty cycle value, which are applied to the metal oxide semiconductor field effect transistor (MOSFET) switches $S1$ and $S2$, as illustrated in Figure 2.1. The switches turn on and off once per switching period, where the fraction of the

switching period for which $S1$ remains on is determined by the duty cycle value. $S1$ and $S2$ operate synchronously, one turning on when the other turns off and vice versa. The fraction of the switching period for which the input voltage V_{IN} is passed to the output determines the ratio of the output to input voltage, where the output voltage V_{OUT} is always less than the input voltage for the buck converter. The inductor L and capacitor C act as a low pass filter so that the output voltage remains at a constant level for the entire duration of the switching period. In reality there is a small but acceptable ripple in the output voltage [24].

It should be noted that continuous conduction mode (CCM) operation of the buck converter is only considered here, i.e. it is assumed that the inductor current is always non-zero during operation, though many of the techniques and implementations discussed in this work are also compatible with discontinuous conduction mode (DCM) operation.

2.2.2 Current Mode Control

An alternative control scheme to VMC is current mode control (CMC), which uses the buck converter's inductor current instead of the output voltage as the controlled variable in the closed-loop system. CMC is usually implemented as a dual loop system, whereby the outer voltage loop provides the reference for the inner current loop as illustrated in Figure 2.2.

While CMC is popular for analogue controlled SMPCs, the difficulties associated with current sampling currently prohibit its widespread implementation in digitally controlled power converters. The main disadvantage of digital current mode control is the need for a high performance ADC that can convert the fast-changing inductor current. Recent efforts have aimed to eliminate the need for such an ADC through current estimation techniques [25, 26]. However the vast majority of digital controllers are VMC-based and therefore these are the main focus of this work. Digital voltage mode control techniques will be elaborated upon in Section 2.4.

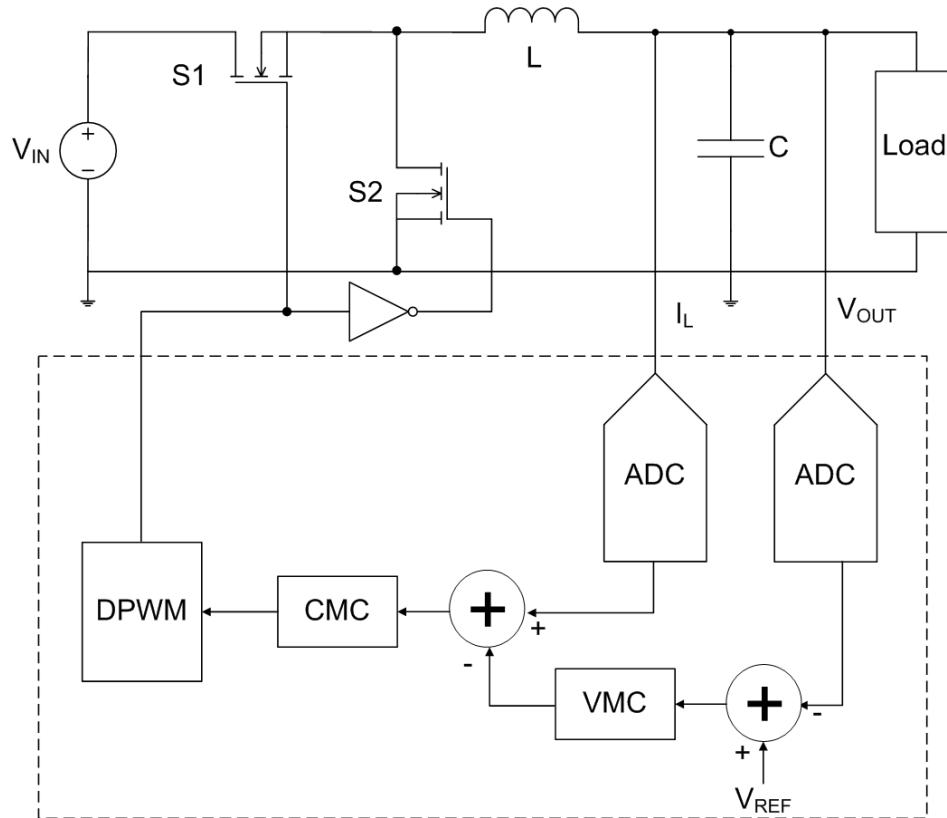


Figure 2.2. Digital current mode control of buck converter.

2.2.3 Analogue to Digital Conversion

SMPS applications typically require relatively high resolution, low latency ADCs with low power consumption. The physical implementations of the ADCs must have low silicon area to minimise cost and to allow them to be integrated on a single chip with the other digital control hardware blocks. Digital power control specialised ADCs therefore have windowed architectures, which only convert voltages in a narrow range about a reference value. Delay-line, ring-oscillator and track-and-hold based ADCs have been introduced as alternatives to the standard flash ADC architecture but with lower silicon area and lower power consumption [27-30]. Supplying both the analogue output voltage and the reference voltage to the ADC allows a digital error value to be directly generated [31]. This eliminates the need to carry out a subtraction operation using arithmetic hardware, which contributes towards reducing the overall execution time of the control algorithm.

Most digital controllers sample the output voltage error once per switching period. The sampling rate is therefore the same as the switching frequency. MOSFET driver ICs typically cannot switch at frequencies higher than 2 MHz [32], thus the ADC sampling rate required is usually not greater than 2 MHz. Higher sampling rates are required in applications using multiple sampling, which involves sampling the output voltage and computing the duty cycle more than once per switching period.

Multiple sampling techniques have been implemented as a means to further improve the transient response by reducing the DPWM delay [33-35], while asynchronous sampling has also been considered in an attempt to widen the bandwidth of the controller [36]. Under-sampling has been used to reduce power consumption of digital controllers in low power applications [37]. This is achieved by performing sampling and duty cycle updating less often i.e. at intervals of multiple switching periods.

2.2.4 Digital Pulse Width Modulation

There are three possible methods to update the duty cycle in DPWMs, namely trailing-edge, leading-edge and dual-edge modulation. Figure 2.3 compares the three modulation schemes as the duty cycle decreases over three switching cycles, where the modulated edges are highlighted in each case. For trailing-edge modulation the pulse is set high at the beginning of a switching period and is set low when the fraction of the switching period determined by the digital duty cycle value has elapsed. For leading-edge modulation the pulse is at the end of the switching cycle, while for dual-edge modulation the pulse is centred about the mid-point of the switching period. In order to avoid limit cycling in the control loop due to quantization effects, the resolution of the DPWM must exceed that of the ADC [38].

High resolution counter-based DPWMs use high frequency clock signals which consume excessive power for digital power control applications. Hybrid delay-line/counter DPWM architectures are favoured because of their lower power and area requirements [27, 39], though it should be noted that the architecture used does not influence the requirements of the other hardware blocks in the digital controller.

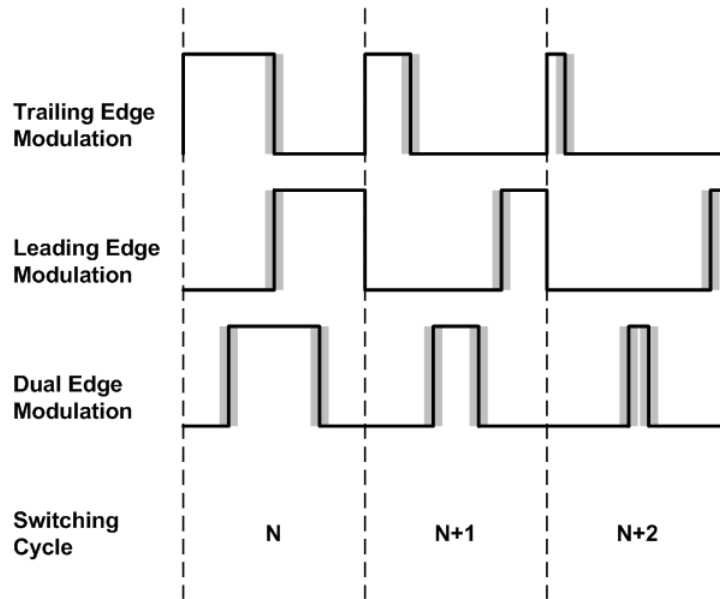


Figure 2.3. Comparison of pulse width modulation methods over three switching cycles.

Other strategies have been adopted to reduce the resolution of the physical DPWM while maintaining a sufficiently high effective duty cycle resolution in the control system. These include sigma-delta and dithering techniques which average the high resolution digital duty cycle value over multiple switching cycles [38, 40, 41].

2.3 Multi-Rail Point-of-Load Converter Applications

Multi-rail systems require multiple DC-DC converters to supply power to a variety of load devices. A common example of such a system is found in computing and telecommunications devices which use an intermediate bus architecture (IBA). In this power distribution architecture the output of an isolated DC-DC converter forms an intermediate voltage bus that supplies multiple voltage rails [19, 20], as illustrated in Figure 2.4. The non-isolated DC-DC converters that convert the bus voltage to the required output levels are known as Point-of-Load (POL) converters because they are located close to the IC that they supply. This helps to minimise power loss and also to reduce parasitic effects between the power converters and their loads that would otherwise negatively impact the transient response of the power converters. The POL converters supply a wide range of load ICs for example ASICs, DSPs, FPGAs, memory devices, etc. [42-45].

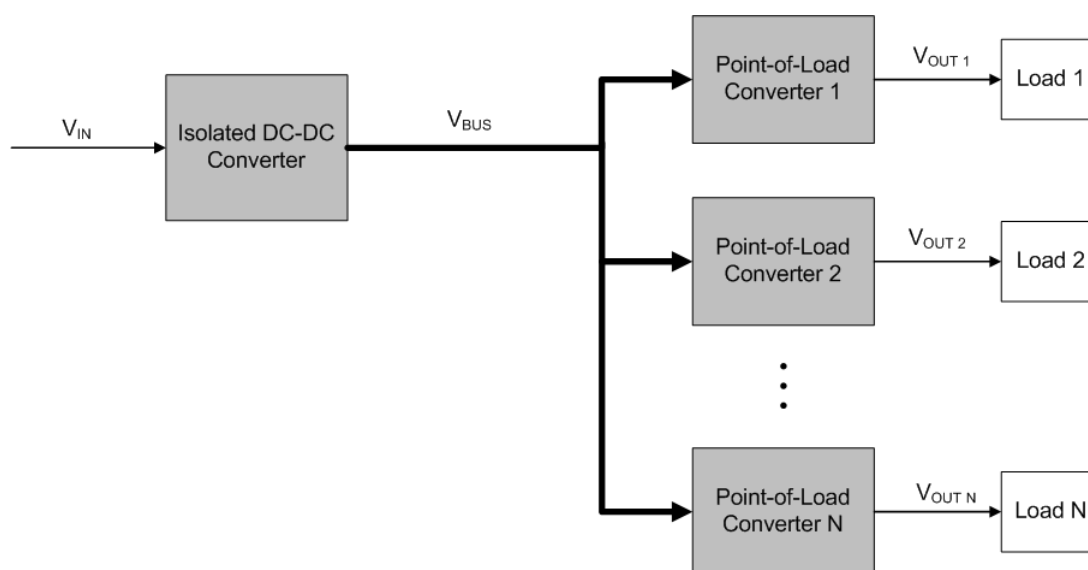


Figure 2.4. Intermediate bus architecture with multiple voltage rails.

While individual digital controllers could be applied to each of the POL converters, it is usually beneficial to share a single digital controller to perform the compensation for all of the power converters as shown in Figure 2.5. This is in contrast with the analogue system illustrated in Figure 2.6, which requires individual compensators for each POL converter resulting in a potentially more expensive power supply that consumes more board area. The POL converters may have either single or multi-phase buck converter topologies.

A multi-phase (also known as interleaved) converter has multiple inductors where the input voltage is applied to each of the inductors at evenly spaced intervals equal to T_S divided by N , where T_S is the switching frequency and N is the number of phases in the power converter. The sum of the inductor currents which is applied to the output capacitor has reduced ripple compared with that of a single phase power converter and the output voltage ripple is also reduced. A smaller output capacitor can therefore be used in multi-phase DC-DC converters. The sharing of current among multiple inductors means that the heat loss is dissipated over a larger area thus allowing the power converter to be applied to loads with high current demands.

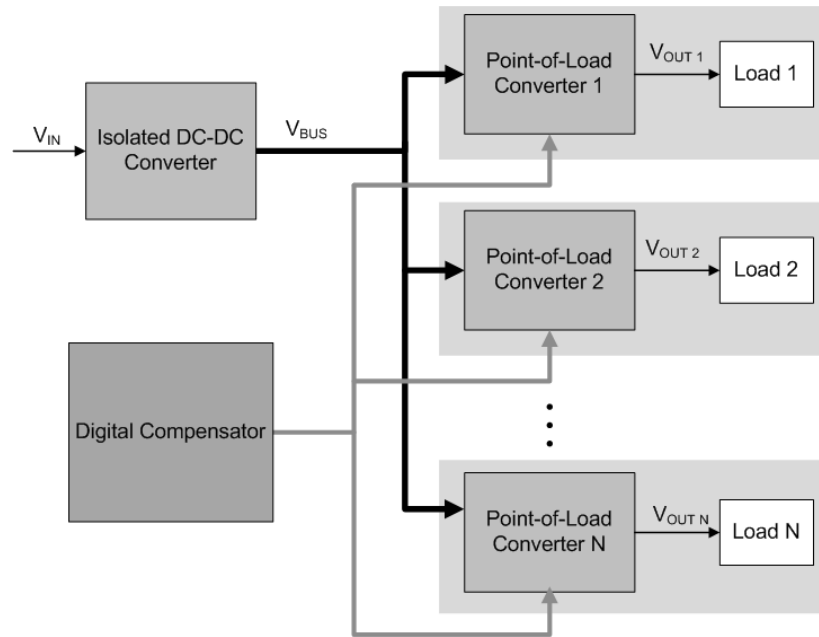


Figure 2.5. Digital compensator applied to multiple DC-DC converters.

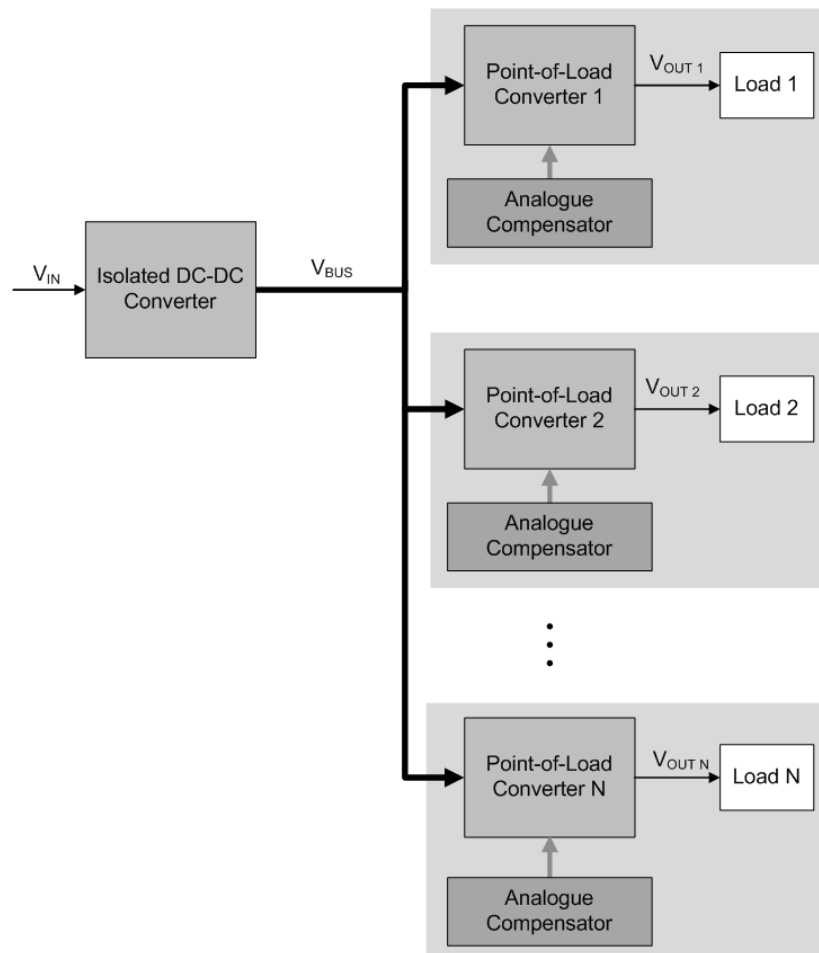


Figure 2.6. Analogue compensators applied to multiple DC-DC converters.

For an N -phase buck converter the frequency of the current ripple is N times the frequency of a single-phase buck converter's ripple. This allows smaller inductors to be used, which gives the SMPC a faster transient response. Figure 2.7 illustrates a four phase buck converter excluding control circuitry.

Although multi-phase buck converters have increased component count and require additional hardware to ensure accurate current sharing between the phases, their use in combination with single phase buck converters is essential in voltage regulator modules (VRMs) to achieve the rigorous specifications of today's microprocessor power supplies [46]. Modern VRMs use multi-phase buck topologies to allow them to react quickly to high slew rate transients in the load current. High switching frequency single phase power converters suffer excessive power loss due to switching and are therefore not a suitable VRM solution [47].

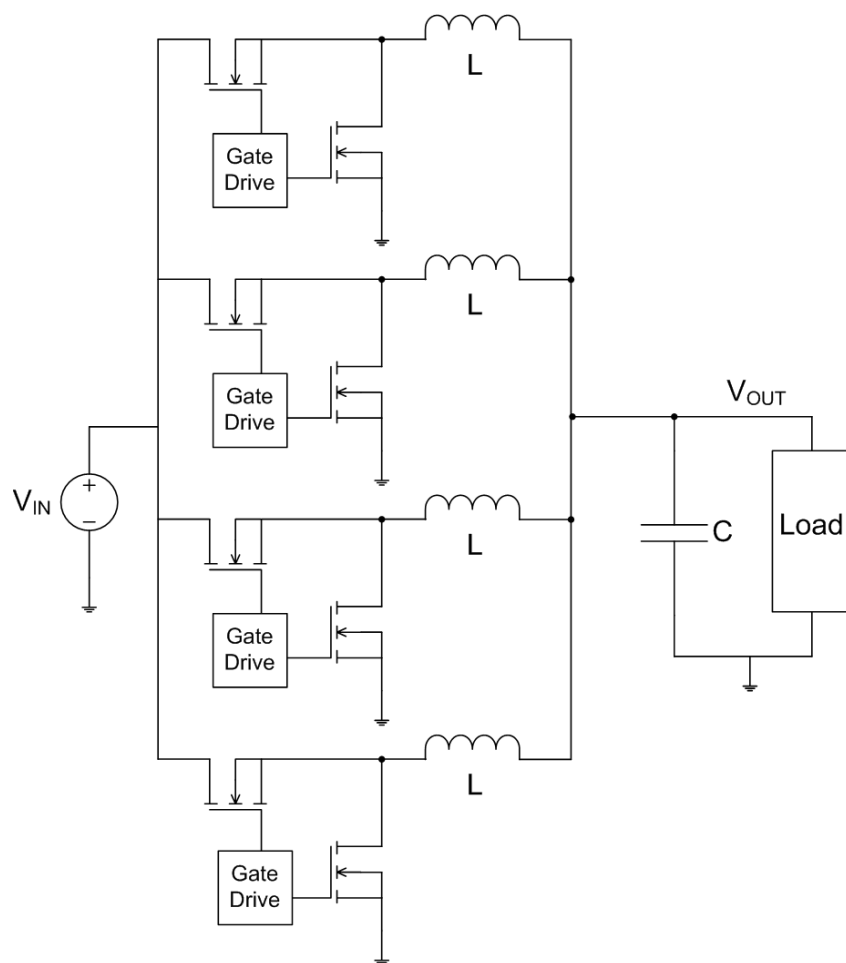


Figure 2.7. Multi-phase buck converter.

2.4 Digital Voltage Mode Control Algorithms

The voltage mode control law implemented by the digital compensator endeavours to maintain a fixed output voltage, determined by a reference value. It does this by rejecting disturbances in the DC-DC converter which are due to variations in the power converter's input voltage supply or changes in the current demand of the load circuit to which the SMPC is applied. The choice of control algorithm depends on the required behaviour or transient response of the converter. Some control strategies include minimising settling time or overshoot. In this section we consider both linear control and adaptive control algorithms, while emerging multi-mode control algorithms are also briefly reviewed.

2.4.1 Linear control

Linear control is one of the simplest methods used to yield a satisfactory closed-loop response for the DC-DC converter plant. There are two main approaches to obtaining a discrete-time (z -domain) transfer function of a linear compensator. The first approach is to design a suitable analogue compensator in the s -domain using standard control techniques and then use a discretisation method to obtain the corresponding z -domain transfer function. Examples of standard control techniques include the frequency response, root locus and pole placement methods [48]. The second approach involves first discretising the DC-DC converter plant to be compensated and then designing an appropriate compensator in the z -domain.

A digital version of the classical analogue proportional-integral-derivative (PID) compensator given in (2.1) is commonly used for voltage mode control of SMPCs [27, 31, 49]:

$$d(t) = K_p \left(1 + \frac{1}{T_i} \int_0^t e(t) + T_D \frac{de(t)}{dt} \right), \quad (2.1)$$

where K_P is the proportional gain, T_I is the integral time, T_D is the derivative time and $e(t)$ is the error input. The Laplace transform can be used to obtain a frequency-domain representation of the continuous time function:

$$G(s) = K_P \left(1 + \frac{1}{T_I} \cdot \frac{1}{s} + T_D s \right). \quad (2.2)$$

A number of methods can be used to discretise this compensator, for example the Backward Euler (also known as Backward Rectangular) or the Bilinear (also known as Trapezoidal or Tustin) methods, which are based on numerical integration. Pole-zero mapping or hold equivalence methods may also be used. In this case the Backward Euler method is applied which uses the following approximation:

$$s = \frac{1}{T} \cdot \frac{z-1}{z}, \quad (2.3)$$

where T is the sampling frequency. The resulting z -domain transfer function of the PID compensator is:

$$G(z) = \frac{D(z)}{E(z)} = \frac{b_0 z^2 + b_1 z + b_2}{z \cdot (z-1)}, \quad (2.4)$$

where the coefficients b_0 , b_1 and b_2 are:

$$b_0 = K_P \left(1 + \frac{T}{T_I} + \frac{T_D}{T} \right), \quad (2.5)$$

$$b_1 = -K_P - 2 \frac{K_P T_D}{T}, \quad (2.6)$$

$$b_2 = \frac{K_P T_D}{T}. \quad (2.7)$$

Using the inverse Z-transform the z -domain transfer function (2.4) can be converted to a time-domain difference equation, involving a set of delay elements multiplied by coefficients:

$$d(n) = d(n-1) + b_0 e(n) + b_1 e(n-1) + b_2 e(n-2), \quad (2.8)$$

where $e(n)$ is the current error value, $e(n-1)$ is the error that occurred in the previous switching cycle, $e(n-2)$ is the error that occurred two switching cycles ago, while $d(n-1)$ denotes the duty cycle value in the previous switching cycle. This difference equation determines the mathematical operations that must be executed in digital hardware to implement the control algorithm. A more general second-order two-pole, two-zero algorithm is also used, which features additional coefficients a_1 and a_2 , allowing the poles of the compensator to be located at more suitable positions for a given application:

$$d(n) = b_0 e(n) + b_1 e(n-1) + b_2 e(n-2) + a_1 d(n-1) + a_2 d(n-2). \quad (2.9)$$

The second order digital PID compensator in (2.4) is commonly applied in a wide range of power control applications [27]. However adding extra poles and zeros to the transfer function of the compensator gives extra degrees of freedom, thus allowing better compensators to be designed, which are required by certain applications with more severe performance requirements. By including additional error and duty cycle terms, $e(n-3)$ and $d(n-3)$, and their associated coefficients b_3 and a_3 , a third order, three pole, three zero compensator is formed:

$$d(n) = b_0 e(n) + b_1 e(n-1) + b_2 e(n-2) + b_3 e(n-3) + a_1 d(n-1) + a_2 d(n-2) + a_3 d(n-3). \quad (2.10)$$

Third order compensators have been reported in research publications [11], while state-of-the-art digital power control products are also using higher order compensators [50].

2.4.2 Adaptive control

Adaptive or auto-tuning algorithms have been developed for digital VMC applications in order to take into account variations in the tolerances of the power converter's components and variations due to the aging of components [51-54]. An adaptive digital regulator achieves this by modifying the coefficients of a linear or non-linear compensator during closed-loop operation. This allows inaccuracies in voltage regulation to be reduced. A disadvantage of adaptive algorithms is their increased computational complexity compared with non-adaptive compensation methods. A more computationally powerful processor is therefore required to execute an adaptive algorithm within the time constraints posed by high switching frequency multi-rail DC-DC converter applications. Chapter 3 presents a thorough analysis of the computation requirements of adaptive and linear compensators in SMPC applications.

2.4.3 Multi-mode control

In an attempt to improve dynamic performance of SMPCs, multi-mode digital control has been introduced [30, 41, 50, 55, 56]. This involves implementing one algorithm when a transient condition occurs and a second algorithm when the SMPC is in steady state as depicted in Figure 2.8. The steady state algorithm is typically a standard PID algorithm while the dynamic algorithm may be either linear or non-linear with a focus on optimising the control action to minimise the output voltage settling time.

Time optimal control for DC-DC conversion is also an on-going research topic, which allows both overshoot and settling time to be minimised when a transient condition is detected [57, 58]. The implementation of these algorithms is not addressed in this work because from the numerous diverse approaches that have recently been proposed in the literature there is no specific outstanding method that can be selected for implementation.

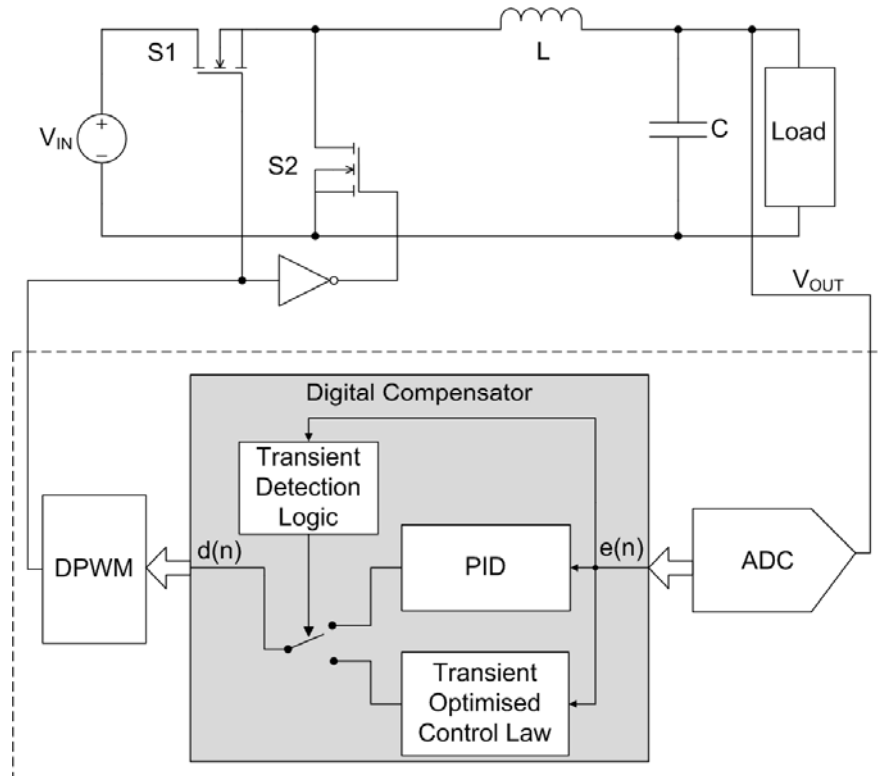


Figure 2.8. Multi-mode digital controller optimised for fast transient response.

2.5 Digital Controller Implementations

One of the main goals in digital controller design is the minimisation of cost, i.e. the minimisation of the quantity and size of the hardware components required. It is therefore necessary to integrate as many functions of the digital controller as possible on a single IC. Existing digital controller implementations feature three essential elements: an ADC, a digital compensator and a DPWM and usually some auxiliary hardware. The auxiliary hardware may consist of a finite state machine (FSM) or reduced instruction set computer (RISC) processor for power management purposes. Memory, communications and sensing components as well as digital to analogue converters (DACs) are also featured as illustrated in Figure 2.9. Timers are usually included to determine relevant time intervals, for example to implement soft-start functionality. Other system requirements include power-on reset (POR) hardware, voltage references (VREF), oscillators (OSC) and voltage regulators, for example, low drop out (LDO) regulators.

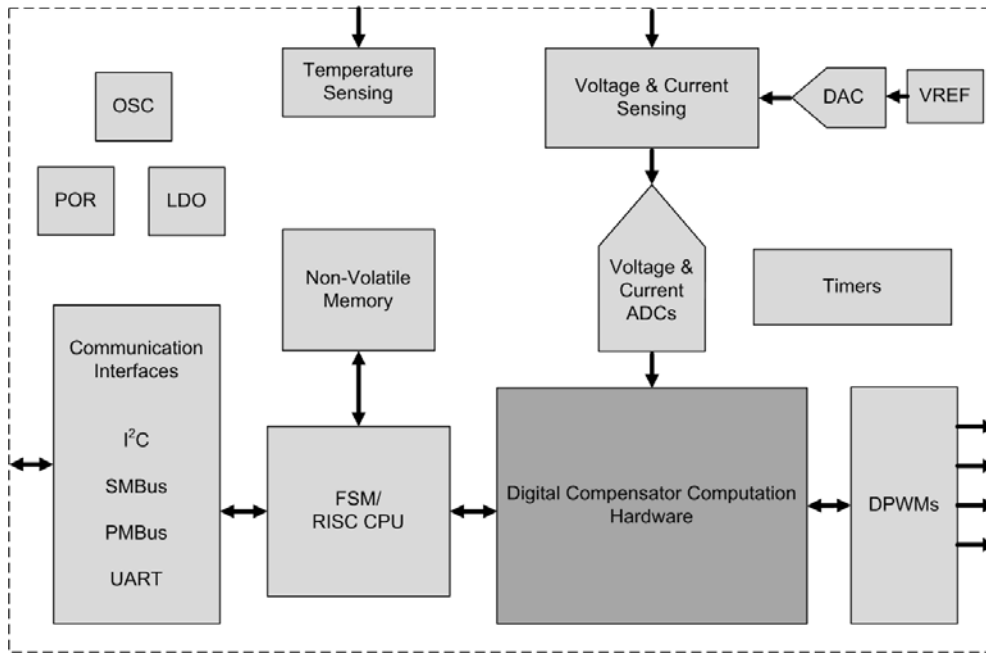


Figure 2.9. Typical digital power controller components.

This section primarily addresses the implementation of the digital compensator computation hardware, however it is important to also consider the interface to the ADC, DPWM and auxiliary hardware elements. This topic will be dealt with in more detail in Chapter 4. Approaches to digital compensator implementation can be divided into two distinct categories: dedicated hardware controllers and digital signal processor based controllers.

2.5.1 Dedicated Hardware Controllers

Dedicated ICs consist of digital hardware that implements one type of control algorithm. In many cases the coefficients relating to the control algorithm can be programmed so that the IC can be used to control a variety of power converters. Although fixed coefficient digital controllers could exploit multiplier reduction techniques to achieve a low area design [59], this type of controller is rarely implemented in practice due to its lack of flexibility.

In the implementation of dedicated hardware controllers, it is not feasible to allocate an individual processing unit to each operation that is to be executed. This would yield an excessively large silicon design, which would be too expensive for use in a power converter application. Folding techniques can be used to reduce the number of area intensive multipliers required to implement complex DSP algorithms [60]. Thus a single hardware multiplier and adder can be multiplexed to execute multiple arithmetic operations in a digital power controller [61]. The compensator hardware implementation can alternatively be minimised by replacing the large multiplier units with shifters [62]. In this case the coefficients are constrained to powers of two, thus significantly reducing the accuracy of the computation. Area costly arithmetic hardware can also be avoided if a finite state machine control approach is chosen instead of a PID or filter based controller [13].

In recent years a number of commercial digital power controllers have become available. These controllers typically contain dedicated hardware to implement a PID-type control algorithm [18, 50, 63], though more advanced adaptive and multi-mode control solutions are also being developed [64, 65]. While these dedicated IC solutions have advantages of low area and low latency, their main drawback is that they are rigid and therefore do not allow a variety of control algorithms to be executed.

A look-up table (LUT) approach has been used to implement PID compensators for applications requiring minimum hardware implementations [11, 27, 66]. Figure 2.10 illustrates the main features of this design. The input of the LUT compensator is a low resolution digital error value, $e(n)$ from a windowed ADC. For the case of the PID compensator described by (2.8), a look-up table stores the product of the error value and the b_0 coefficient for all possible values of $e(n)$. Similar LUTs store the results of the multiplications involving the b_1 and b_0 coefficients and the low resolution error value. For a single switching cycle, the current error, $e(n)$ and previous two error values, $e(n-1)$ and $e(n-2)$, act as addresses to the LUTs which output the products $b_0.e(n)$, $b_1.e(n-1)$ and $b_2.e(n-2)$. These three products are summed with the previous duty cycle value, $d(n-1)$ using a hardware adder to generate the new duty cycle value, $d(n)$.

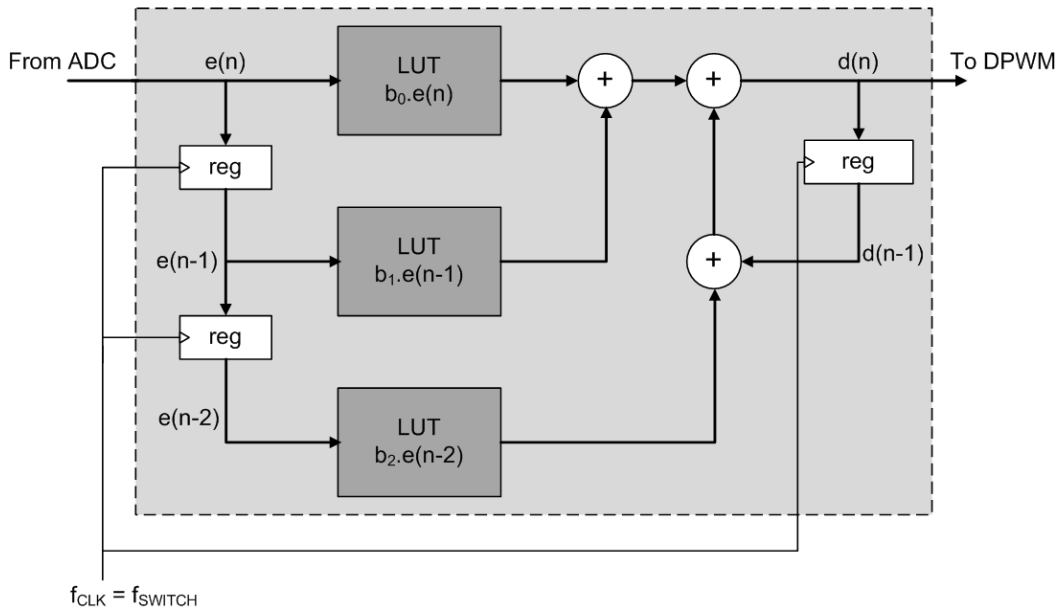


Figure 2.10. Look-up table based PID compensator.

As well as having low area requirements, the latency of the duty cycle calculation carried out by the LUT compensator is low. The three product values may be generated in parallel while two additions may also be performed in parallel if more than one hardware adder is included in the design. The compensator is programmable in that the coefficient values may be changed but varying the number of poles and zeros in the control law is not possible. Multi-mode operation may be enabled by selection of a second set of coefficient LUTs depending on the required mode of operation of the compensator [41]. However more sophisticated adaptive algorithms [54, 67, 68] cannot be executed using this simple hardware structure because the coefficients cannot be updated during operation.

In order to achieve fast transient response through the implementation of time optimal algorithms, asynchronous digital hardware can be used. A continuous time digital controller incorporating asynchronous logic can respond quickly to transient conditions without having to wait for a synchronising clock signal [69], while steady state operation is still controlled by a synchronous LUT compensator.

A possible digital controller solution for the multi-rail system application is a parallel-dedicated architecture. This is a highly hardware intensive structure for high speed execution of control algorithms. Each power converter has a dedicated computational element for each operation in its control algorithm. Figure 2.11 shows an example of this architecture for the 2P2Z algorithm for N POL converters. The area and cost of such a structure is not justifiable for the power converter application, thus highlighting the need for some form of resource sharing in the implementation of the processing hardware. Although the LUT approach would be preferable to dedicated hardware in this case, there would still be a restriction on the type of control laws that could be implemented by the LUTs, as mentioned previously.

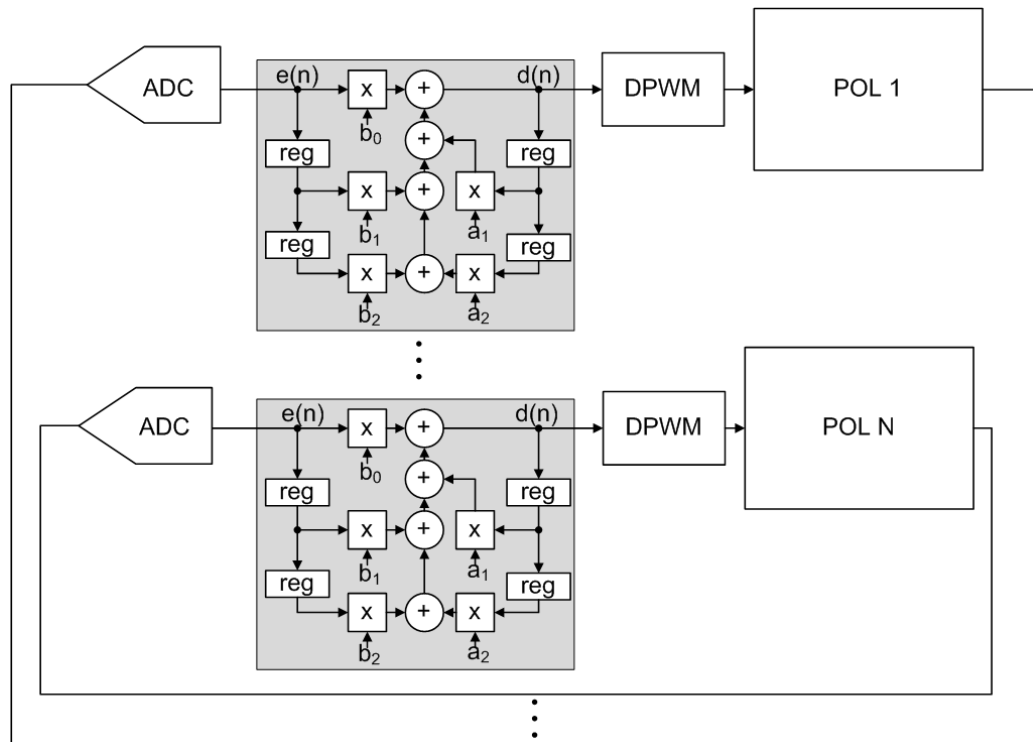


Figure 2.11. Parallel architecture dedicated to implementing 2P2Z algorithm for controlling multiple POL converters.

A multiplexed architecture, which offers a less area intensive implementation compared with the parallel architecture, is illustrated in Figure 2.12. Again, this architecture has a dedicated computational element for each operation of its control algorithm but the single set of computational elements is multiplexed to execute the same algorithm for all of the DC-DC converters to be controlled. Each POL converter can use different coefficients, for example *POL 1* would use coefficients $b_0^{(1)}$, $b_1^{(1)}$, $b_2^{(1)}$, $a_1^{(1)}$ and $a_2^{(1)}$ while *POL 2* would use coefficients $b_0^{(2)}$, $b_1^{(2)}$, $b_2^{(2)}$, $a_1^{(2)}$ and $a_2^{(2)}$. The POL converters are then allocated time slots when the control algorithm runs on the dedicated processor. The throughput of such an architecture could be increased by inserting pipelining registers in the datapath. The problem with having dedicated hardware components for each algorithm operation is that area increases if more complex algorithms are used. This architecture is also inflexible and does not allow for the possibility of different power converters being controlled by different control laws. The inadequate performance of the architecture in implementing large numbers of control algorithms is a further disadvantage.

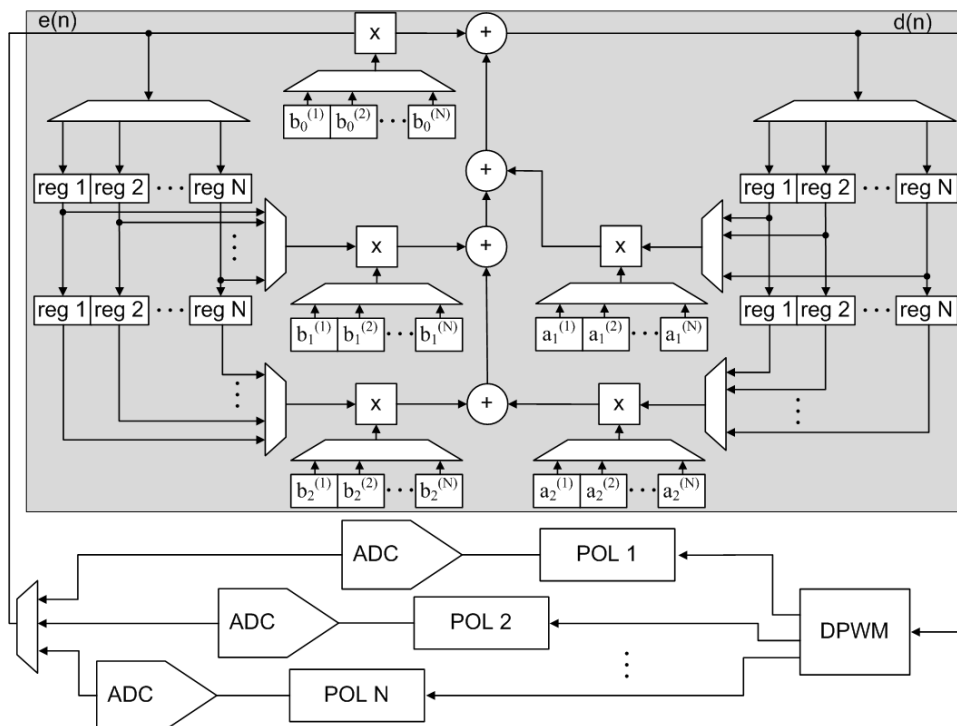


Figure 2.12. Multiplexed architecture dedicated to implementing 2P2Z algorithm for controlling multiple POL converters.

2.5.2 Digital Signal Processor Based Controllers

In digital signal processor based controllers a DSP core executes the control algorithms required by the power converter application. The main constituents of the DSP core are multiplication and arithmetic hardware, program control hardware and memory. The primary element is the multiplier-accumulator (MAC) unit, which can perform one multiply operation and add the result to the value in the accumulator register in a single clock cycle. General purpose digital signal processors usually contain many other peripherals and computational features that are irrelevant to power converter control applications [14, 15, 17].

The advantages of using DSP-based controllers have been demonstrated in numerous DC-DC converter applications to date [51, 70-72]. The inherent flexibility of a DSP means that it can be programmed to execute different user-specified control laws depending on the regulation requirements of the power converter. The possibility of implementing adaptive algorithms has also been demonstrated in the research field [53, 73, 74]. FPGAs offer similar flexibility and are thus widely used to prototype digital power controllers [33, 75, 76]. However the cost and size of such devices means that they cannot be used in commercial power supplies.

Interrupts govern the execution of control algorithms in the DSP core whereby the control algorithm is executed as part of the interrupt service routine (ISR), which is triggered periodically by the interrupt pulse signal, as illustrated in Figure 2.13. Background code consisting of monitoring and communications tasks can be executed during the idle time of the DSP when it is not executing the control algorithm code.

Multiple independent power converters can be simultaneously controlled by a single DSP. The DSP's resources are time-multiplexed so that the different control algorithms for each DC-DC converter are executed sequentially. This is in contrast with a system using dedicated hardware based controllers in which case separate controllers would be required to execute individual control laws. Figure 2.14 illustrates how a DSP can execute control algorithms for a power supply system consisting of two voltage rails.

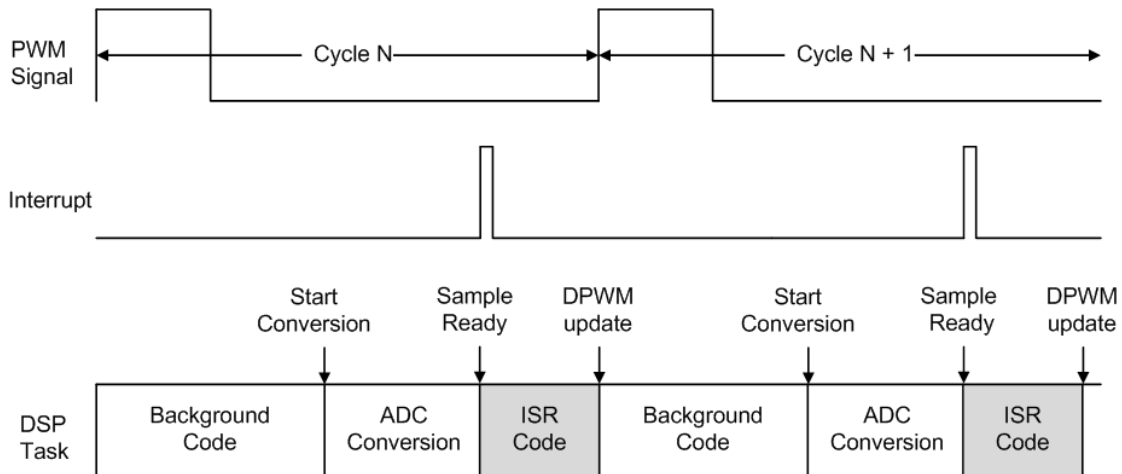


Figure 2.13. Interrupt-based execution of control algorithm using a digital signal processor.

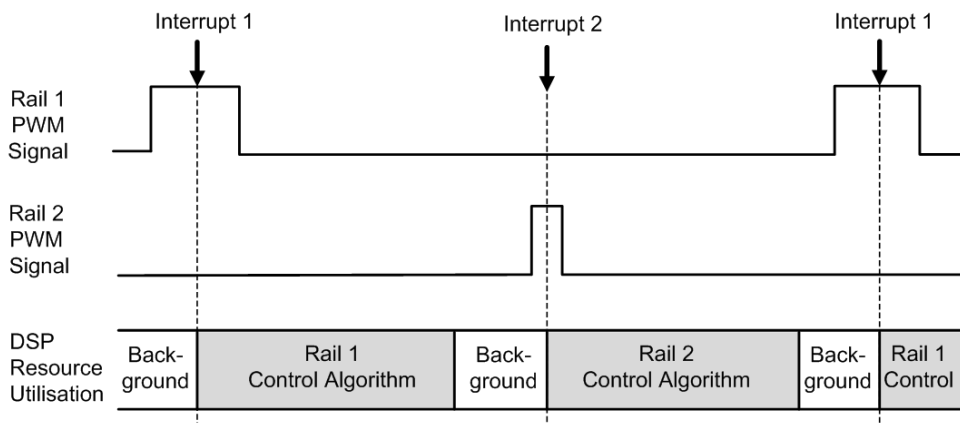


Figure 2.14. Resource utilisation of DSP controlling dual rail power supply system.

The number of power converters that can be controlled by a single DSP is dependent on the time it takes the DSP to execute the control algorithm and also on the execution frequency of the algorithm, which is usually the same as the switching frequency of the power converter. Until recently no more than three SMPCs with a switching frequency of 1 MHz could be controlled by commercial single MAC DSPs [77]. This was primarily due to the fact that the low-end DSPs that were typically applied to power converters had only a single datapath and therefore all operations had to be executed sequentially [14, 15, 17, 70, 78]. Even though the DSP may have had more

than five pulse width modulated (PWM) outputs, they could not all be applied to 1 MHz power converters.

A more recent solution that attempts to meet the performance requirements of high switching frequency multi-rail systems uses a 32-bit floating point math co-processor in parallel with its main 32-bit DSP datapath [19]. The math processor acts as a control law accelerator (CLA) which executes the control algorithms while the main DSP core performs communication and housekeeping tasks. Although it appears that this solution meets the computational requirements of the application, its recent release means that no performance details have been published in the literature as of yet. The application problem is solved by this product by adding an additional processor without regard to possible architectural optimisations. Hardware minimisation which is vital in the digital power control application is not apparent in this architectural design. Even though the 32-bit floating point number representation offers flexibility it also negatively impacts the area of the processor. This approach is very much in contrast with fixed-point digital power controllers which are successfully implemented with much lower resolution [38, 41, 52, 70]. A more comprehensive analysis of the suitability of a range of existing DSP architectures for multi-rail SMPC applications is provided in Chapter 3.

The increased prevalence of digital control in low power DC-DC converter systems has meant that power electronics engineers have had to become more familiar with DSP and microcontroller programming. In most cases the programming is assembly language based because this allows optimisation of the control algorithm code so that it runs in the shortest possible time. Companies providing digital power control products offer graphical user interface (GUI) based software programs for programming of the digital controller [79]. This eliminates the need for end-users of the DSP core to perform assembly programming, thus enabling them to focus on the controller's functionality. Consequently when designing a DSP-based digital controller the ease of use of the programming interface, although important, is not a high priority because it will not be visible to the end-user.

A problem with conventional DSPs is that the switching frequencies of the individual SMPCs being controlled in a multi-rail system are normally restricted to being identical or integer multiples of each other. This is due to variations in the delay between ADC-sampling and DPWM duty-cycle-updating, which are caused by simultaneously occurring interrupts if the switching frequencies have a non-integer ratio. In designing a buck converter, one of the first specifications to be decided upon is the inductor current ripple ΔI_L , which is illustrated in the waveform of Figure 2.15 for continuous conduction mode operation. A simplified representation for ΔI_L can be derived from the equation for the voltage across the inductor L [24]:

$$V_L = L \frac{di}{dt} \quad (2.11)$$

Referring to Figure 2.1, when switch $S1$ is on and switch $S2$ is off, the voltage across L is:

$$V_L = V_{IN} - V_O \quad (2.12)$$

The buck converter is in this state for a time interval ΔT given by:

$$\Delta T = D \cdot T_S \quad (2.13)$$

where D is the steady state duty cycle for the buck converter defined as the ratio of the output voltage V_O to the input voltage V_{IN} :

$$D = \frac{V_O}{V_{IN}} \quad (2.14)$$

and T_S is the switching period. Applying these expressions to (2.11) and rearranging yields:

$$\Delta I_L = \frac{V_O}{V_{IN}} \cdot \frac{V_{IN} - V_O}{L \cdot f_S} \quad (2.15)$$

where the switching frequency f_s is:

$$f_s = \frac{1}{T_s} \quad (2.16)$$

The value of ΔI_L is thus only dependent on the values of V_O , V_{IN} , L and f_s . The input and output voltages will be specified by the application, therefore the required inductor current ripple will be determined by choice of L and f_s . If f_s is restricted to an integer multiple of another frequency, this also restricts the value of L that can be chosen. Having to choose a switching frequency that is larger than necessary will impact the efficiency of the buck converter due to the relationship between switching frequency and efficiency [4]. Similarly having to choose an inductor that is larger than necessary also affects the efficiency [80]. Removing the restriction of integer multiple switching frequencies thus allows more suitable values of f_s and also L to be chosen. A detailed analysis of this issue is presented in Chapter 6.

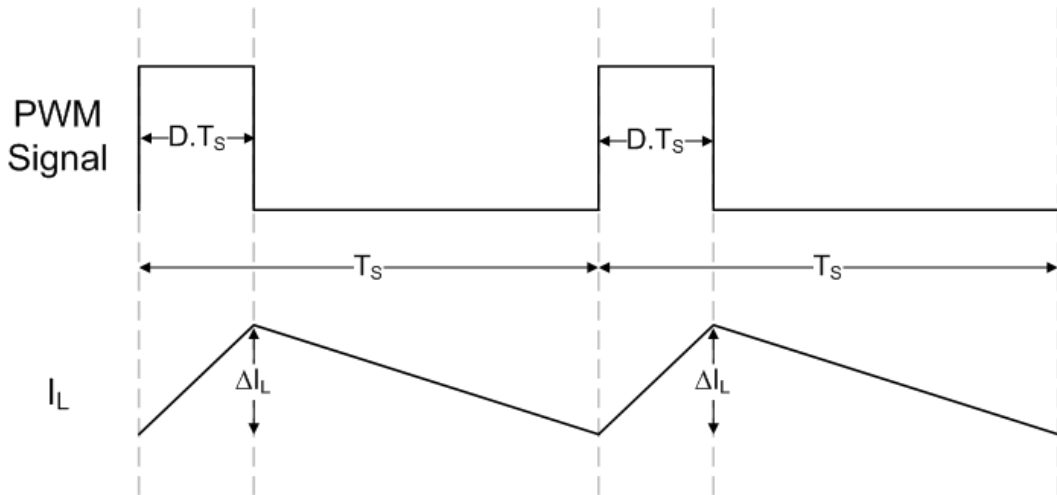


Figure 2.15. PWM and inductor current waveforms for buck converter.

2.6 Summary

It has been shown in this chapter that the demands of modern microprocessors and other ICs are posing power supply designers with more challenging specifications. This has resulted in advanced digital control algorithms and DC-DC converter applications of increased complexity being proposed.

The slow adoption of advanced digital control algorithms in commercial products emphasises the need to examine more computationally powerful digital hardware options that can meet the critical timing constraints of the DC-DC converter applications in an area efficient manner.

Dedicated hardware based controller architectures are deemed to be impractical for multi-rail DC-DC converter systems because of their excessiveness of area and lack of flexibility. The suitability of DSP-based controllers for multi-rail applications has been outlined, though the need for an optimised processor architecture has also been highlighted.

Chapter 3

Analysis of DSP Core Architectures

3.1 Introduction

In order to select an appropriate digital controller architecture for application to multi-rail power supply systems, it is imperative to examine the merits of all potentially relevant architectures. In Chapter 2, existing digital signal processor based controllers were discussed. This chapter explores a number of alternative DSP architecture types with respect to their performance in executing power control algorithms.

In Section 2.5.2 the limitations of single MAC DSPs in executing control algorithms for multiple power converters with switching frequencies in excess of 1 MHz were highlighted. While more computationally powerful platforms have been proposed, the benefits of arbitrarily adding auxiliary resources remain to be proven. The relationship between the characteristics of alternative potential architectures and the requirements of the multi-rail power converter systems needs to be clearly identified.

Up to now mostly linear PID-type control algorithms have been used in power converter applications. There is a need therefore to re-examine the DSP architectures being used in the context of the execution of the more complex adaptive algorithms. In order to facilitate more widespread adoption of adaptive compensation methods in

digital power controllers, it is necessary to define the exceptional features of a programmable digital signal processor that can efficiently implement such algorithms. Although adaptive compensators can be implemented on currently available general purpose DSPs [73], the increased computational complexity of adaptive algorithms requires a more computationally powerful platform especially when considering the trend towards multi-rail converter systems and higher switching frequencies, which pose more demanding time constraints [81, 82].

An alternative to investigating more computationally powerful DSP architectures is to apply faster clock frequencies to existing single MAC based architectures that are used in multi-rail DC-DC converter systems. Even though this approach increases the computational performance, its high clock frequency implementation does not comply with the requirements of the majority of power conversion applications which demand a low power consumption digital controller. Indeed, current solutions are trying to meet this requirement of minimising the power consumption of the digital controller by applying architectures based on look-up tables or finite state machines; however these architectures cannot implement the more complex adaptive type algorithms and are therefore deemed unsuitable for multi-rail SMPC systems, as discussed in Section 2.5.1. Moreover, this work focuses on increasing the computation speed of typical control algorithms by identifying a suitable DSP architecture that can exploit parallelism in the algorithms rather than simply attempting to use a higher frequency clock.

The performances of a number of DSP architecture types are evaluated in this chapter by applying a number of pertinent power control algorithms to these architectures. An analysis of potential multi-rail controller architectures is given initially. This is followed by an introduction to the relevant control algorithms. Finally the performance characteristics of the processors are compared and conclusions are drawn regarding the most suitable processor architecture based on the time constraints of the multi-rail power control application.

3.2 Potential Architectures

The datapath of a DSP contains all of its computational and data processing elements, including a multiplier-accumulator (MAC), a shifter, an arithmetic and logic unit (ALU) and a register file, as illustrated in Figure 3.1. In the context of control algorithm execution, the datapath's most frequently used element is the multiplier-accumulator.

Conventional DSPs used for control algorithm execution have only a single multiplier-accumulator unit, therefore all MAC operations in the control algorithm must be carried out sequentially. In general, all computational operations are executed sequentially on conventional DSPs, thus the architectures do not permit multiple computational operations to be executed in parallel.

It has already been noted in Section 2.5.2 that control of multiple power converters is achieved by time-multiplexing the DSP's resources so that individual algorithms are executed sequentially. In this section, a number of architectures consisting of more than one MAC unit are investigated, in an attempt to identify higher performance successors to the current single-MAC, DSP-based power controllers.

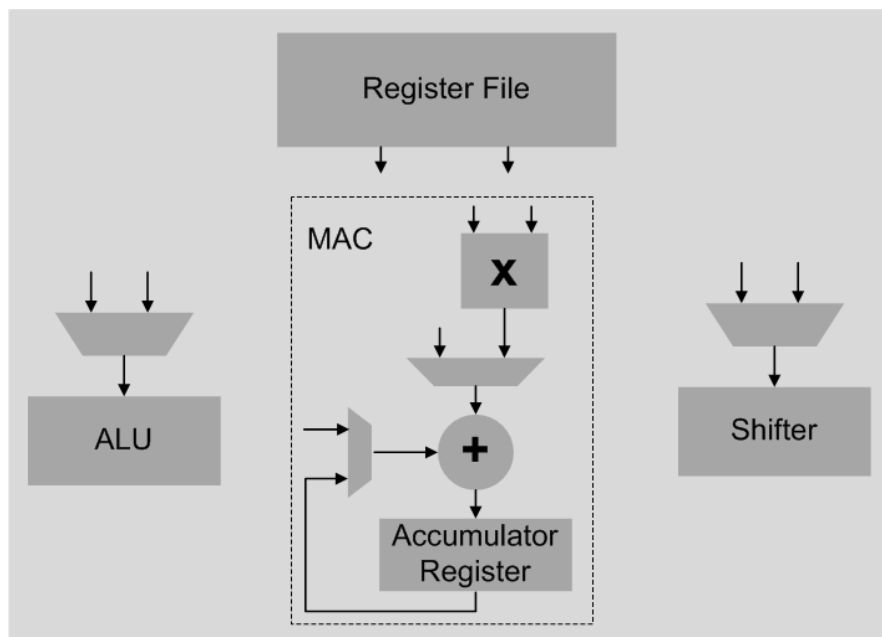


Figure 3.1. Typical computational elements in the datapath of a DSP.

3.2.1 Multiple-Issue Processor Architectures

High-end DSPs exploit instruction level parallelism to execute algorithms in a shorter time interval than conventional single-issue DSPs. This is achieved by adding supplementary processing elements to the datapath of the DSP, thereby allowing multiple operations to be executed in parallel. For example, a dual-MAC processor can carry out two multiply-accumulate operations simultaneously [83, 84]. Very long instruction word (VLIW) and superscalar architectures typically have sufficient computational hardware to allow them to execute greater than two computational operations in parallel [85].

VLIW architectures have variable length instructions, which allow a variable number of operations to be executed in parallel. The maximum instruction length of a VLIW processor is many times longer than the instruction word of a non-VLIW processor because its instruction word includes opcodes and operands for multiple parallel operations. Typically, the operations to be performed in parallel are explicitly specified by the programmer. The programmer therefore considers all data and resource constraints when choosing whether operations should be executed sequentially or in parallel. The compiler may also detect parallelism in the code provided by the programmer in order to generate machine-level code that further exploits the parallel functional units of the processor. Although programming for a VLIW processor requires more effort than for a single-issue architecture, this drawback is not as significant for DSP-based power controllers as it would be for general purpose processors. This is because end-users commonly configure commercial DSP-based power controllers using a graphical user interface, which allows them to select a number of pre-programmed control algorithms, as stated in Section 2.5.2.

Superscalar architectures similarly allow multiple operations to be executed in parallel. In contrast to VLIW processors, the programmer usually writes program code for superscalar processors that does not specify which operations may be executed in parallel, while the compiler also does not specify parallelism in its machine-level code. Instead, a specialised hardware unit in the processor's program controller

decides on which operations to execute in parallel by detecting data and resource constraints as it decodes and executes the program code. VLIW architectures are favoured over their superscalar counterparts for embedded applications because they do not require this additional hardware.

The concept of applying a multiple-issue architecture in a digital power control system is illustrated in Figure 3.2. The architecture consists of multiple functional units, which are controlled by a single program control hardware unit. The functional units work simultaneously in executing the algorithm for a single power converter. When this algorithm is completed the hardware executes the algorithm for the next power converter. This allows different power converters to be controlled by different algorithms.

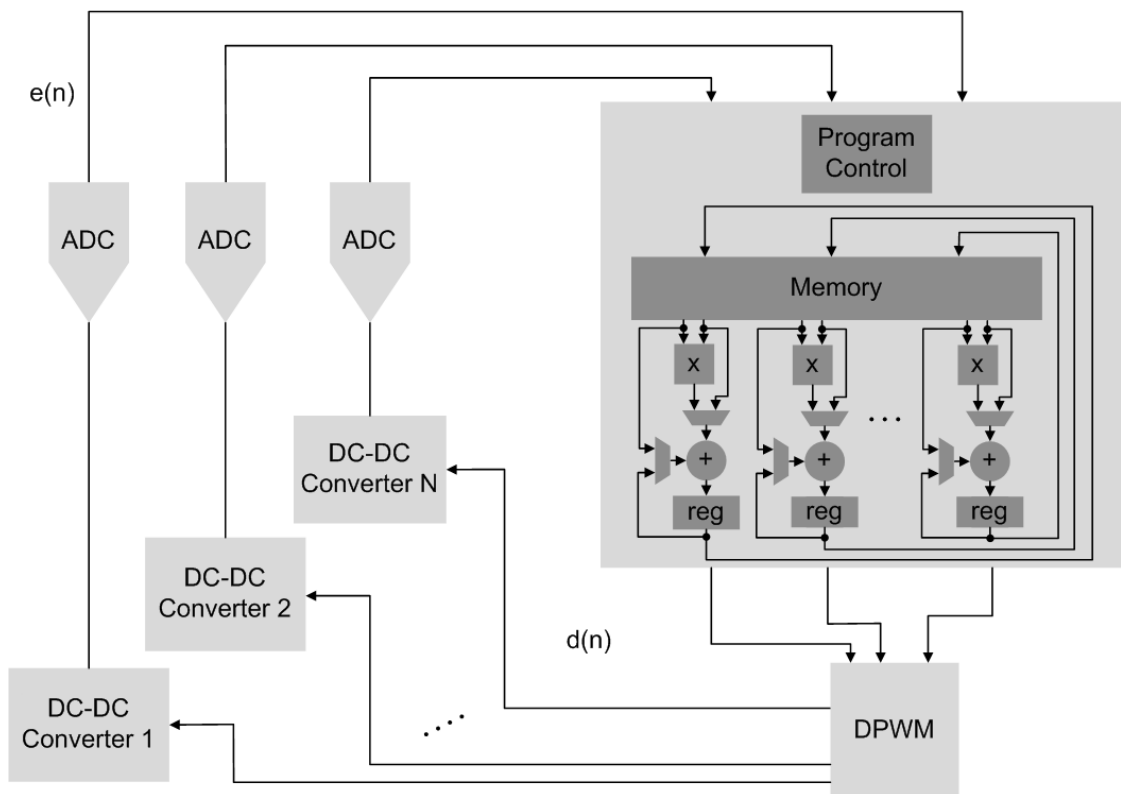


Figure 3.2. Multiple-issue architecture for controlling multiple DC-DC converters.

3.2.2 Multiprocessor Architectures

Developments in processor design have led to a possible characterisation of processors into two categories: uniprocessor and multiprocessor types. Uniprocessors have been replaced by multiprocessors in applications where a single processor can no longer meet the demanded performance requirements. Multiprocessors aim to increase execution speed by applying multiple processors in parallel. Multiprocessing DSP systems are typically in the form of multicore processors, where each of the individual processors is placed on the same die [86]. The multicore processor has effectively a type of multiple instruction multiple data (MIMD) architecture [87].

Application of a multicore processor would allow individual DSP cores to be dedicated to the execution of one control algorithm for one power converter, so that multiple duty cycles would be calculated in parallel, as illustrated in Figure 3.3. This is in contrast with multiple-issue architectures, where only one control algorithm may be executed at any one time. Each individual core of the multicore processor could also divide its time among executing more than one algorithm, provided that it had sufficient processing power to accomplish this. This would permit the number of power converters controlled to be greater than the number of processor cores.

The disadvantage of using multicore architectures is that they do not improve on the latency of a duty cycle calculation executed on a single DSP core. They would therefore not be optimal for repetitively executing a single control algorithm at a high rate, as is required by multi-phase SMPC applications that require fast transient response [88, 89].

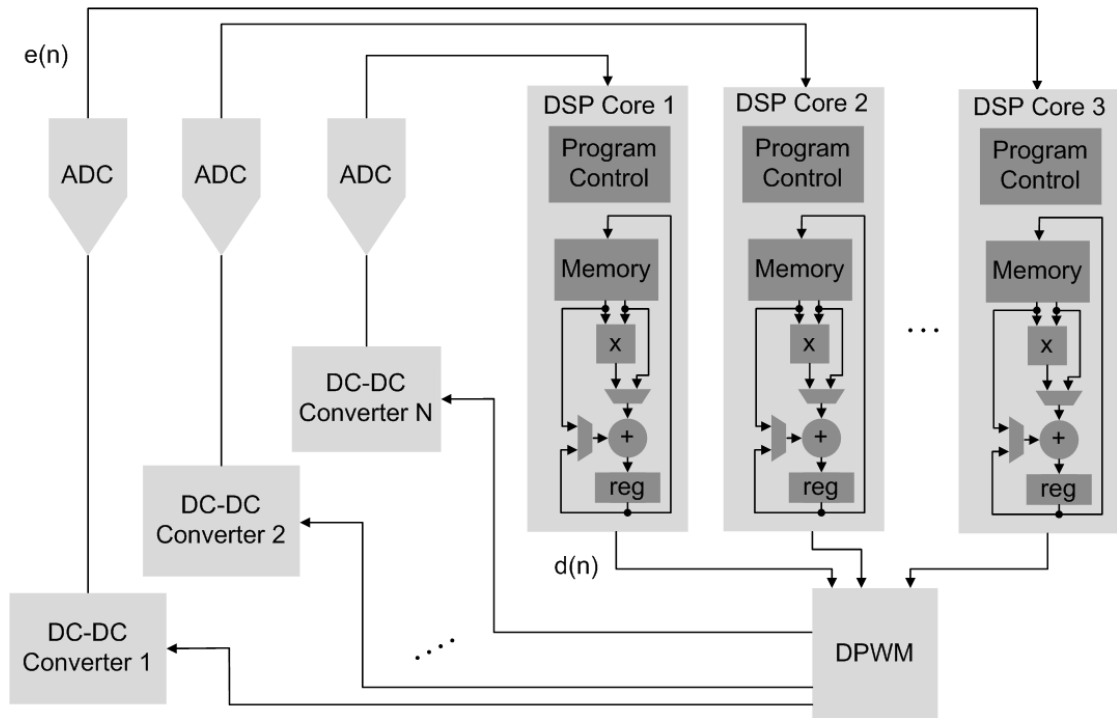


Figure 3.3. Multicore architecture for controlling multiple DC-DC converters.

3.2.3 Single Instruction Multiple Data Architectures

In addition to the multiple-issue and multiprocessor architectures that are featured in high-end DSPs, the single instruction multiple data (SIMD) architectural technique is also used. SIMD exploits data level parallelism rather than instruction level parallelism to increase execution speed [90]. The issuing of a single SIMD instruction results in the execution of multiple instances of that instruction in parallel using a separate data set for each instance.

Some processor architectures only facilitate SIMD-based instructions for certain operations, for example processors that only execute dual-MAC SIMD-based operations. In that case a specialised SIMD-based MAC unit is often utilised that executes either single MAC operations or multiple parallel MAC operations where the input operands are sub-words of the full-length operands. Alternative SIMD-based processor architectures use multiple independent clusters consisting of registers and functional units to execute a wide range of SIMD operations.

Separate SMPCs can be potentially controlled in parallel with the same control algorithm simultaneously applied to separate power converters using a single instruction multiple data (SIMD) architecture, as illustrated in Figure 3.4. However, different algorithms cannot be executed in parallel using such a SIMD-based approach. For example, a dual MAC SIMD-based DSP cannot execute two different control algorithms for two independent SMPCs in parallel. In contrast, dual MAC processors based on the 2-way VLIW architecture can fulfil this requirement and are therefore preferable for use in multi-rail power control applications.

Due to the inflexible nature of the SIMD architecture, only the performances of the multi-core and multiple-issue VLIW processors are examined in further detail with respect to executing multiple power control algorithms.

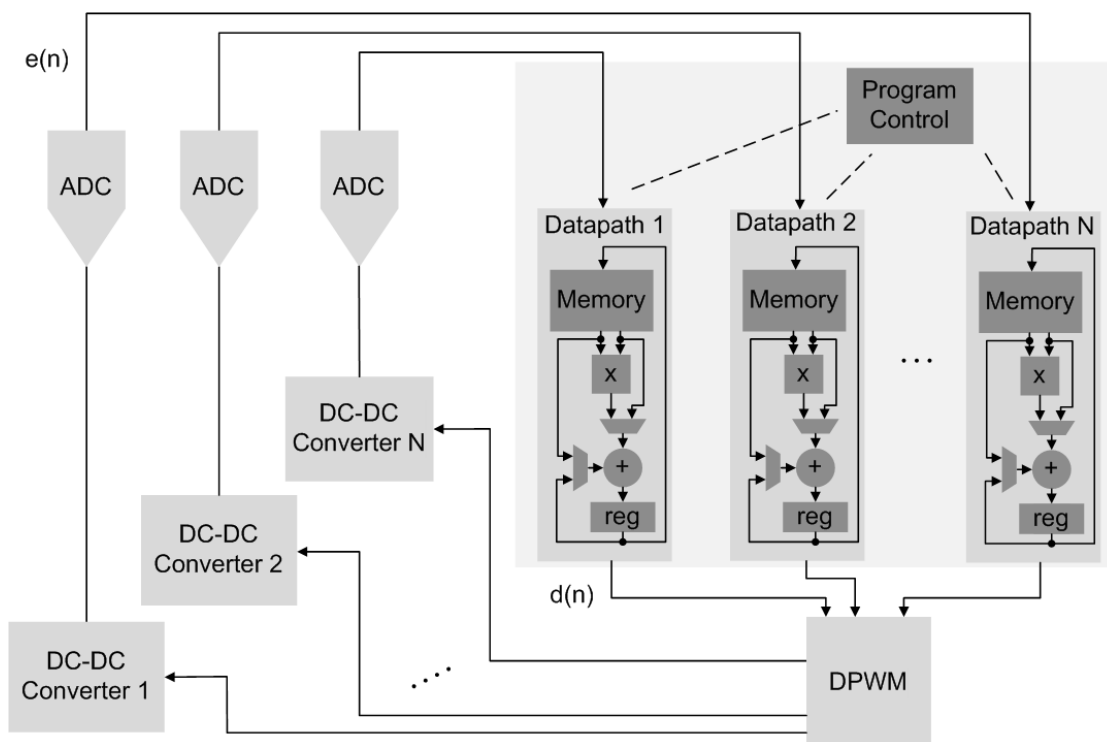


Figure 3.4. SIMD architecture for controlling multiple DC-DC converters.

3.3 Algorithm Analysis

Similar to the filtering, coding and FFT benchmarks that are used to evaluate the performance of general purpose digital signal processors [91], this section analyses specific control algorithms that are relevant to digital power controller applications. The linear and adaptive compensators that were introduced in Section 2.4 are addressed in this context. The computation and data movement operations necessary to yield the required control output are quantified and classified for each of the control algorithms. By examining the individual operations that make up the algorithm, the type of functional elements needed to produce the duty cycle result is determined. The precedence of the operations in the algorithm is also examined.

Precedence in an algorithm refers to the order of execution of the individual operations that make up the algorithm [59]. Precedence limits parallelism in the execution of an algorithm and therefore also limits the maximum execution speed of the algorithm. It is vital to consider precedence when scheduling operations on multiple-issue DSPs.

Scheduling involves ordering of the algorithm's operations while observing a set of defined constraints. There is always a fixed constraint on the area or performance in the specification of the required hardware architecture. In the case of the digital power controller the area of the design, in terms of computational resources, is the fixed constraint. In particular, for multiple-issue VLIW architectures, the quantity of datapaths available is the fixed constraint, while for multi-core architectures the quantity of cores available is the fixed constraint. For an architecture that is defined by a particular set of computational resources, maximum performance is obtained by scheduling the algorithm's operations so that as few clock cycles as possible are required to implement the algorithm. High performance is obtained by scheduling the operations so that the utilisation of the DSP's resources is high over the entire duration of the algorithm's execution. The linear and adaptive algorithms are examined here in terms of their scheduling on a single-core, single-MAC architecture and a two-way VLIW, dual-MAC architecture. This analysis permits a more thorough comparison of a wider range of architectures to be undertaken in Section 3.4

3.3.1 Linear Control Algorithm

As discussed in Section 2.4.1, the use of second order compensators is prevalent in existing buck converter controllers. The second order, two-pole two-zero (2P2Z) compensator has a difference equation of the following form:

$$d(n) = b_0 e(n) + b_1 e(n-1) + b_2 e(n-2) + a_1 d(n-1) + a_2 d(n-2) \quad (3.1)$$

where b_0 , b_1 , b_2 , a_1 and a_2 are the compensator coefficients, $e(n)$ is the current error value, $e(n-1)$ is the error that occurred in the previous switching cycle, $e(n-2)$ is the error that occurred two switching cycles ago, while $d(n)$, $d(n-1)$ and $d(n-2)$ denote the duty cycle values for the current and two previous switching cycles respectively. The duty cycle, $d(n)$ is updated once per switching cycle.

It is clear from (3.1) that execution of these algorithms involves multiplying previous error and duty cycle values by coefficients and summing them to obtain the new duty cycle value, which is similar to the form of the digital infinite impulse response (IIR) filter. Apart from the main computational operations a number of other tasks must be executed to obtain a correct duty cycle output when the algorithm is being executed using fixed-point arithmetic. Shifting and rounding of the duty cycle value are required before passing it to the DPWM. Data access is another issue to be considered because values must be moved along the error and duty cycle delay lines, as in standard digital filters, so that they may be accessed correctly in each iteration of the algorithm. The current error and duty cycle values must also be stored in each iteration.

The block diagram in Figure 3.5 represents the direct form realisation of the 2P2Z algorithm given in (3.1), where the \ll symbol designates a shift operation. The direct form digital filter realisation is used here because it is the preferred method for the implementation of low-order digital filters on programmable processors [92]. This is due to the fact that it generally has a faster execution time than other realisations, such as the canonic form, because of its simpler memory addressing requirements.

In order to ensure fast transient response the duty cycle should be updated as soon as possible after receiving the voltage error sample. This can be achieved by performing some operations in the control algorithm prior to obtaining the voltage error sample to reduce the duty cycle calculation latency, as Figure 3.6 illustrates.

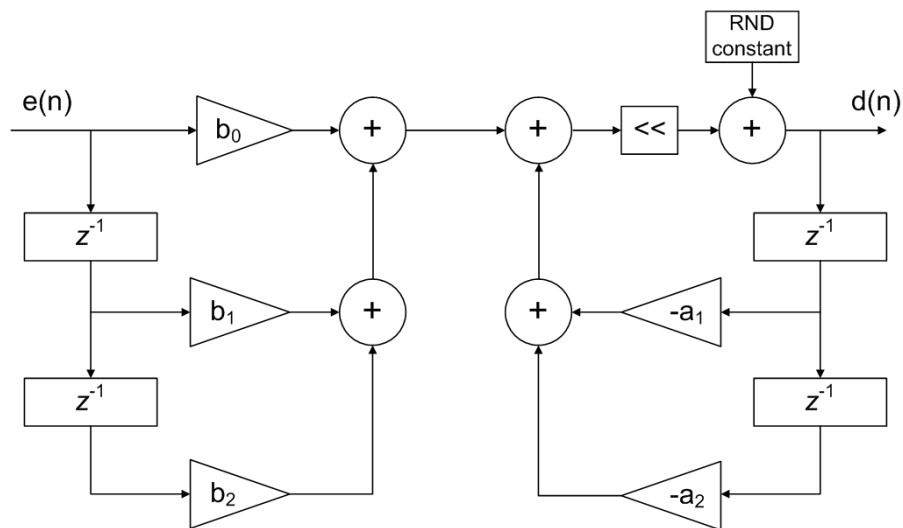


Figure 3.5. Block diagram of direct form realisation of 2P2Z algorithm.

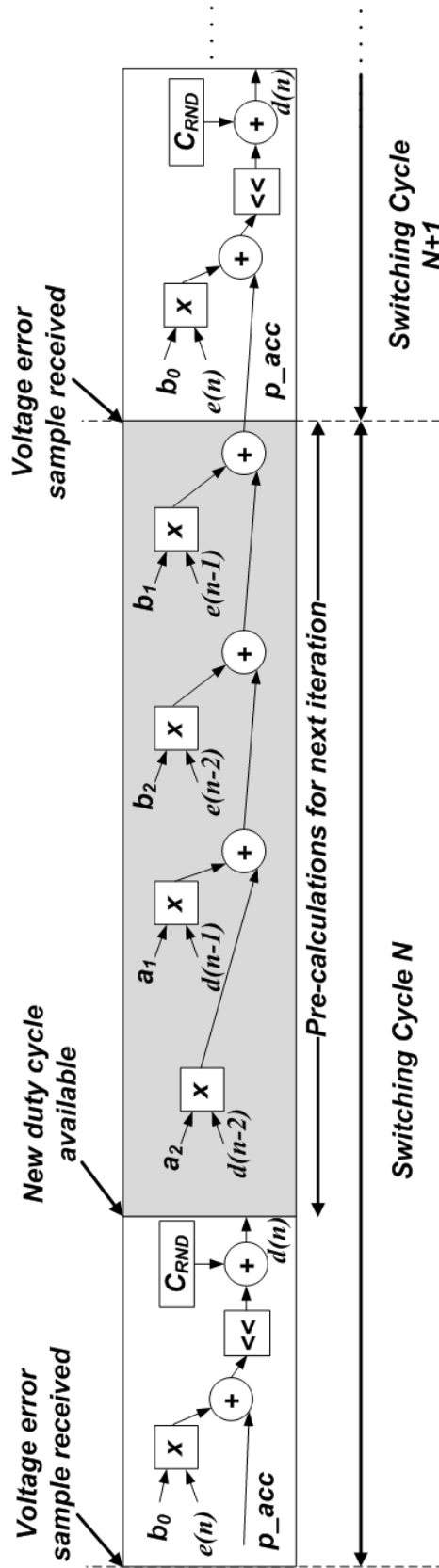


Figure 3.6. Ordering of 2PZ algorithm operations to minimise duty cycle calculation latency.

Only a few operations remain to be executed after sampling and before duty cycle updating. The pre-calculated accumulated value p_acc only needs be added to the error sample $e(n)$ which has been multiplied by the coefficient b_0 , before the result is scaled, rounded and applied to the DPWM. The ‘pre-calculations’ for the next iteration of the algorithm can be executed after the calculation of the duty cycle for the previous iteration. Figure 3.7 illustrates a block diagram of the direct form realisation of the 2P2Z algorithm, which has been modified to account for the re-ordering of the operations to the implement the pre-calculations.

Figure 3.8 is a precedence graph of the pre-calculated 2P2Z algorithm represented in Figure 3.7. The precedence graph is effectively a dataflow graph that illustrates the precedence constraints for only one iteration of the algorithm [60]. The numbers next to each of the operations in Figure 3.8 indicate which operations these correspond to in the dataflow graph in Figure 3.7. It is important to note that there are a minimum number of operations that must be carried out sequentially due to the precedence constraints of the algorithm.

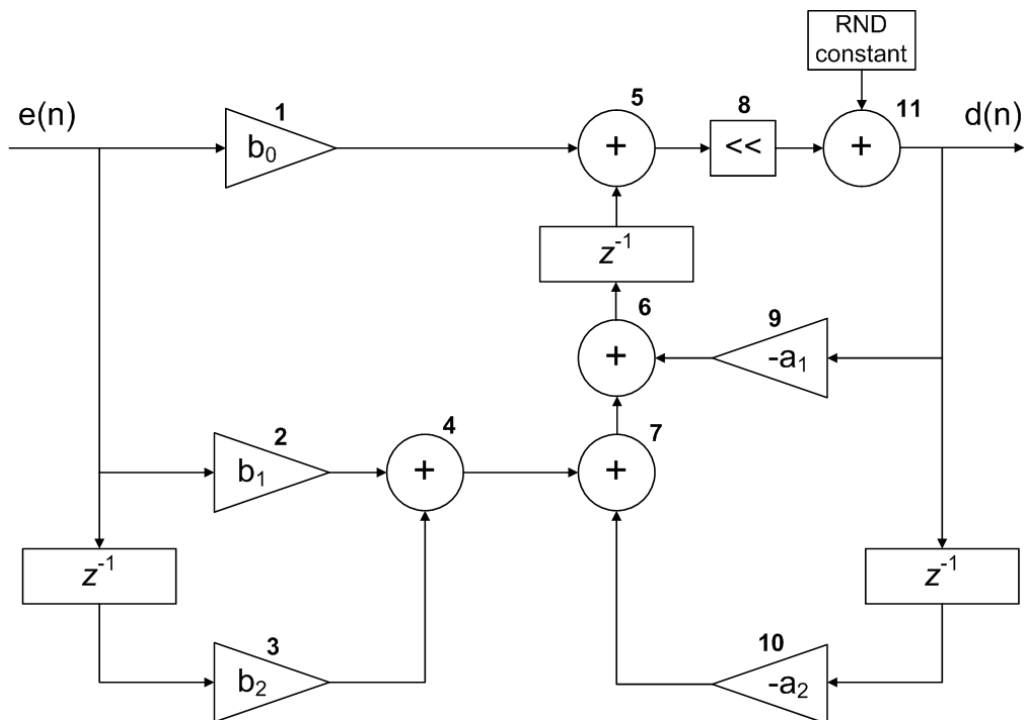


Figure 3.7. Block diagram of pre-calculated 2P2Z algorithm.

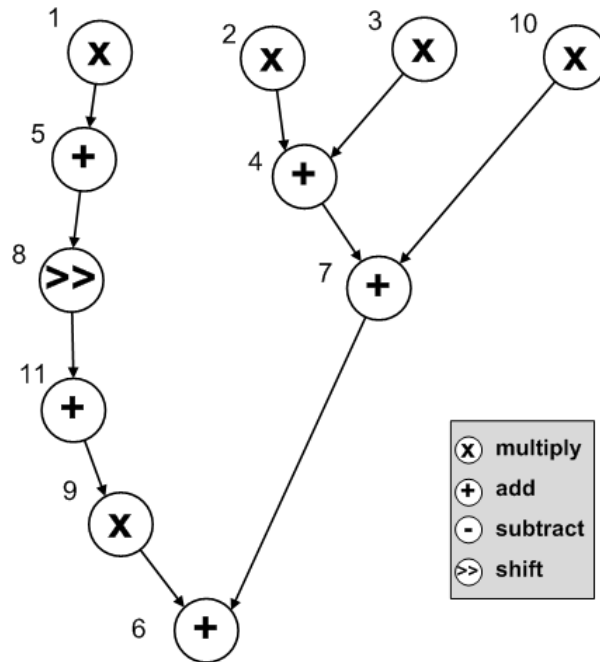


Figure 3.8. Precedence graph of pre-calculated 2P2Z algorithm.

Two resource constrained scheduling options are investigated in order to determine a relationship between the area and performance of the hardware when implementing the 2P2Z algorithm. It is assumed that multiplication, addition, shifting and multiply-accumulate operations are each executed in a single clock cycle. Updating of filter delay lines and other data movement operations are assumed to be executed in parallel with the computational operations.

When programming a DSP to execute control algorithms, a ‘return from interrupt’ instruction is required at the end of each iteration of each algorithm. This program flow instruction is required due to the interrupt triggered nature of control algorithm execution on DSPs, as was described in Section 2.5.2. The instruction directs the processor to return to execute background code until the next interrupt signal occurs, which triggers the execution of the next algorithm. Such program flow instructions often have latencies of multiple clock cycles because they require the instruction pipeline of the DSP to be refilled before execution of the next instruction. The latency of a program flow instruction depends on the implementation of the processor’s program control hardware. The number of clock cycles of latency will at most be

equal to the pipeline length of the processor, although the latency can be eliminated through the inclusion of specialised logic, which is assumed for each of the algorithms considered here.

Taking the preceding assumptions into account, maximum performance schedules for executing the 2P2Z algorithm on single and dual MAC DSPs are derived. The maximum performance schedule for the single MAC DSP is illustrated in Figure 3.9 while Figure 3.10 illustrates the schedule for the dual MAC processor.

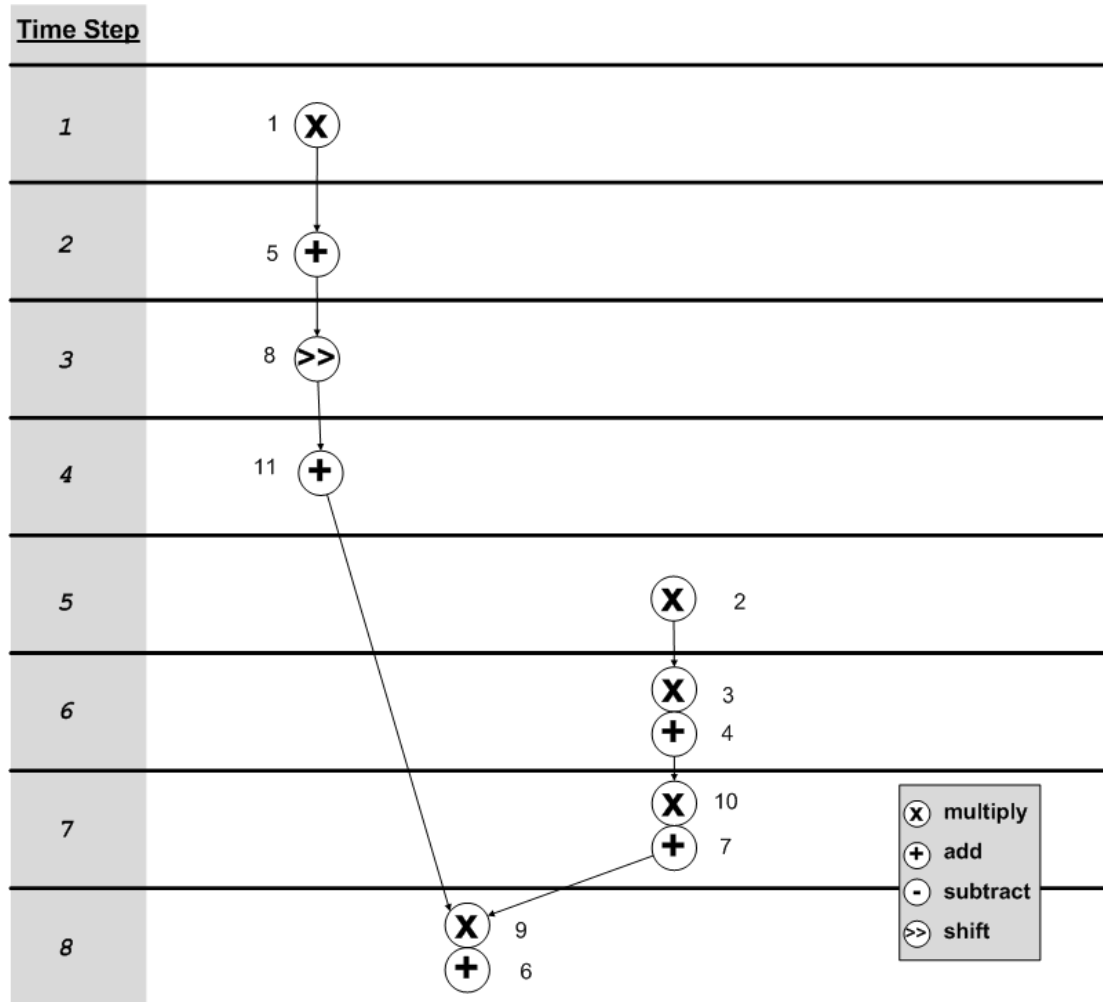


Figure 3.9. Scheduling of 2P2Z algorithm operations on single MAC architecture.

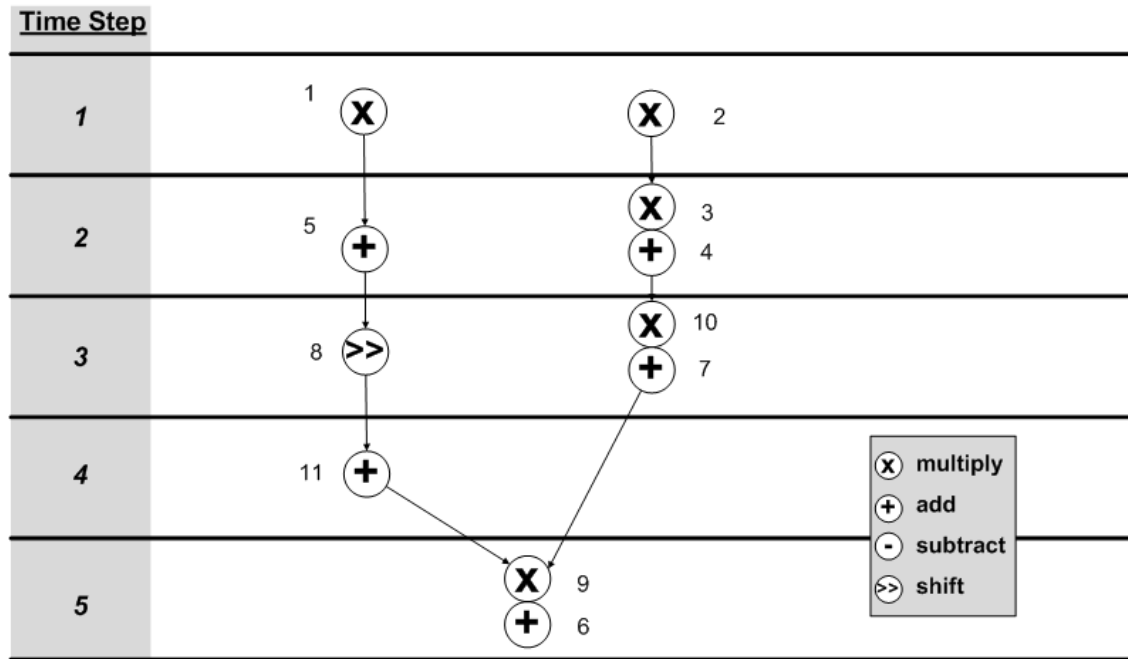


Figure 3.10. Scheduling of 2P2Z algorithm operations on dual MAC architecture.

The time steps shown in each of these figures represent DSP clock cycles. For the most part arbitrary scheduling of operations is restricted by the algorithm's intra-iteration and inter-iteration precedence constraints. The operations are ordered so that the duty cycle value is available in the minimum number of clock cycles after the output voltage error sample is available. Updating operations are carried out after the duty cycle calculation.

The minimum number of clock cycles required to execute an algorithm can be determined using As Soon As Possible (ASAP) scheduling [60]. Unlimited computational resources are assumed for ASAP scheduling where only the precedence constraints restrict the scheduling of the algorithm's operations. The schedule illustrated in Figure 3.10 is in fact an ASAP schedule for the 2P2Z algorithm therefore the minimum execution time of the algorithm is five clock cycles.

3.3.2 Adaptive Control Algorithm

An algorithm based on linear prediction can be used to implement adaptive control for SMPCs. This type of algorithm is particularly suited to implementation using a DSP-based controller because it does not necessitate the use of custom hardware components, as is required by some other adaptive control techniques [51, 68]. It simply involves performing some additional computational operations, similar to those executed in a linear control algorithm. The following equations define the second-order adaptive algorithm that was proposed in [73]. The duty cycle $d(n)$ is:

$$d(n) = K_0.c(n) + V_{SET}.W_a, \quad (3.2)$$

where K_0 is the adaptive proportional gain factor, V_{SET} is the converter's output voltage set-point or reference value, W_a is an adjustment gain factor that compensates for deviations in the input voltage of the power converter and $c(n)$ is the output of the compensator, which is given by:

$$c(n) = e(n) + b_1.e(n-1) + a_1.c(n-1) + a_2.c(n-2). \quad (3.3)$$

In (3.3), b_1 , a_1 and a_2 are adaptive coefficients, $c(n-1)$ and $c(n-2)$ are the values of $c(n)$ in the previous two iterations of the algorithm and $e(n)$ is given by:

$$e(n) = \frac{e_c(n)}{2} + \frac{e_c(n-1)}{2}, \quad (3.4)$$

where the control error $e_c(n)$ is:

$$e_c(n) = V_{SET}.W_a - V_0.G.W_a, \quad (3.5)$$

for a constant gain value G . The adaptive coefficients are updated in each iteration of the algorithm as follows:

$$b_1 = b_{d1}.\alpha(n) + b_{o1}, \quad (3.6)$$

$$a_1 = a_{d1}.\alpha(n) + a_{o1}, \quad (3.7)$$

$$a_2 = a_{d2}.\alpha(n) + a_{o2}, \quad (3.8)$$

$$K_0 = K_{d1}.\alpha(n) + K_{o0}, \quad (3.9)$$

where b_{d1} , b_{o1} , a_{d1} , a_{o1} , a_{d2} , a_{o2} , K_{d0} and k_{o0} are constants and the updating factor $\alpha(n)$ is calculated as:

$$\alpha(n) = \alpha(n-1) + 128 e(n-1).p(n) \quad (3.10)$$

and in (3.10) $p(n)$ is:

$$p(n) = e(n) + b_1e(n-1). \quad (3.11)$$

The adjustment gain factor, W_a is also updated in each iteration as:

$$W_a(n) = \frac{e(n)}{8} + W_a(n-1). \quad (3.12)$$

The block diagram in Figure 3.11 illustrates how these equations are combined to execute the adaptive algorithm. The operations of the adaptive algorithm can be broken into the two main categories: duty cycle calculation and coefficient updating.

Table 3.1 shows the number of individual computational operations in each category for the algorithm described in Figure 3.11. The duty cycle calculation is the main priority in the execution of the algorithm and its operations should be those first executed when the voltage sample is obtained. After the duty cycle has been computed, the coefficients can be updated for the next iteration of the algorithm.

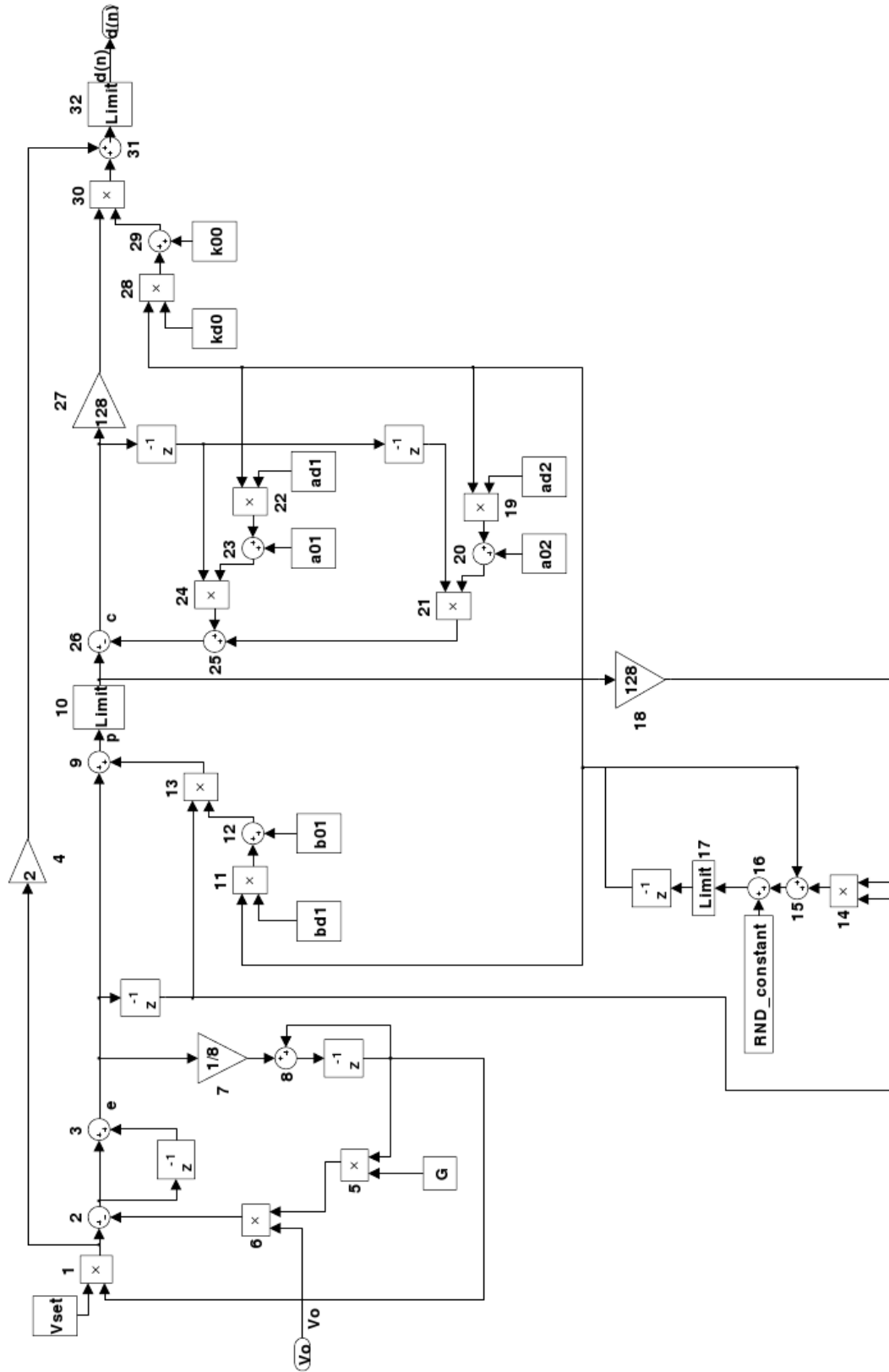


Figure 3.11. Block diagram of adaptive algorithm.

Table 3.1. Classification of adaptive algorithm computational operations.

Operations	Addition	Multiplication	Shifting	Limiting
Duty Cycle	6	7	2	2
Coefficient Updating	7	5	2	1
All	13	12	4	3

The main operations of the algorithm include addition, multiplication, shifting and limiting. Limiting refers to constraining a value between upper and lower thresholds. Fewer shifting and limiting operations are required compared with multiplication and addition operations as noted in Table 3.1. In a number of cases multiplication and addition can be combined in a single MAC operation where the product is added to the previous result. In addition to accumulating the previous result, as is common in standard DSP implementations, it is also necessary for the previous result to be shifted or used as a multiplicand in the next operation.

From the data storage perspective, the results of intermediate operations frequently need to be stored for use in the next iteration of the algorithm or for later use in the current iteration. This is in contrast with the linear compensator algorithms discussed in Section 3.3.1. Such algorithms consist primarily of sequential accumulation operations and as a result they do not require many intermediate storage operations. The execution of the adaptive algorithm therefore has unique requirements in terms of both computations and data movement operations.

Figure 3.12 is a precedence graph of the algorithm represented in Figure 3.11. The numbers next to each of the operations in Figure 3.12 indicate which operations these correspond to in the dataflow graph in Figure 3.11. It can be seen that some of the results of the operations in the precedence graph are not used by other operations. This is because these results are not required until the next iteration. It is also clear that updating of coefficients for the next iteration cannot take place until certain intermediate values in the duty cycle calculation have been computed.

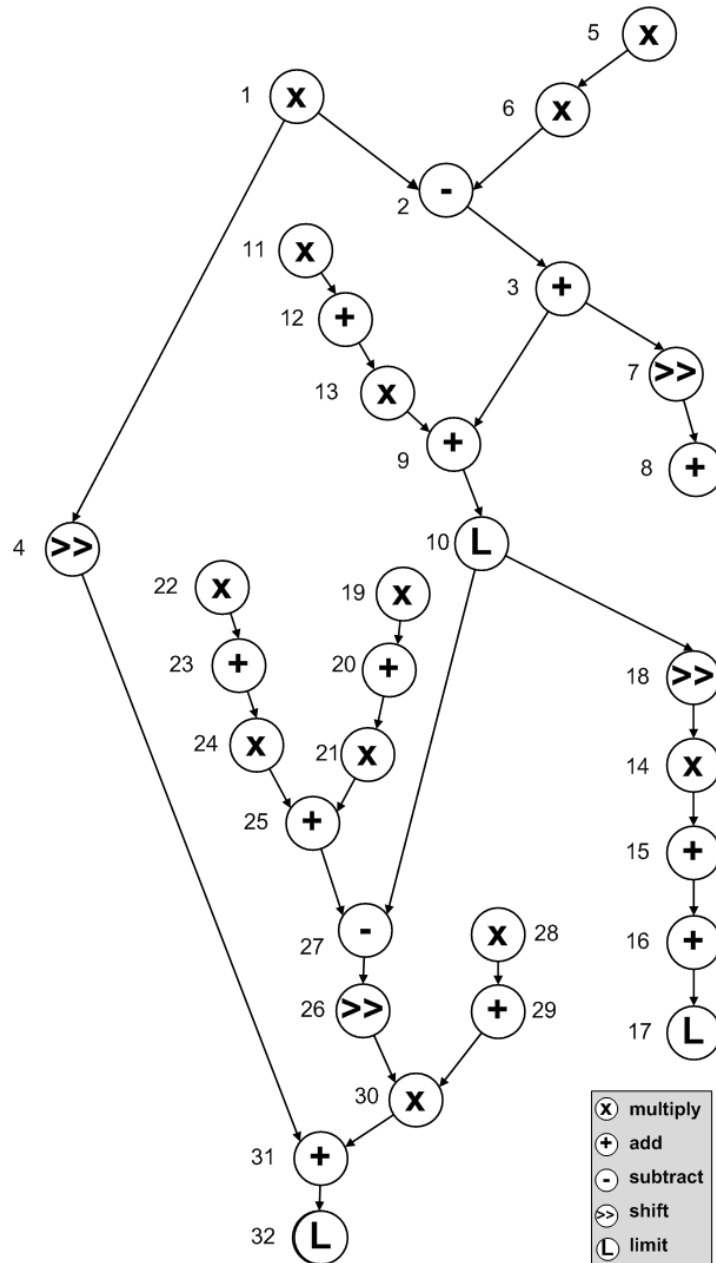


Figure 3.12. Precedence graph of adaptive algorithm.

The maximum performance schedules for executing the adaptive algorithm on single and dual MAC DSPs are illustrated in Figure 3.13 and Figure 3.14 respectively, where data movement operations are assumed to be executed in parallel with computational operations.

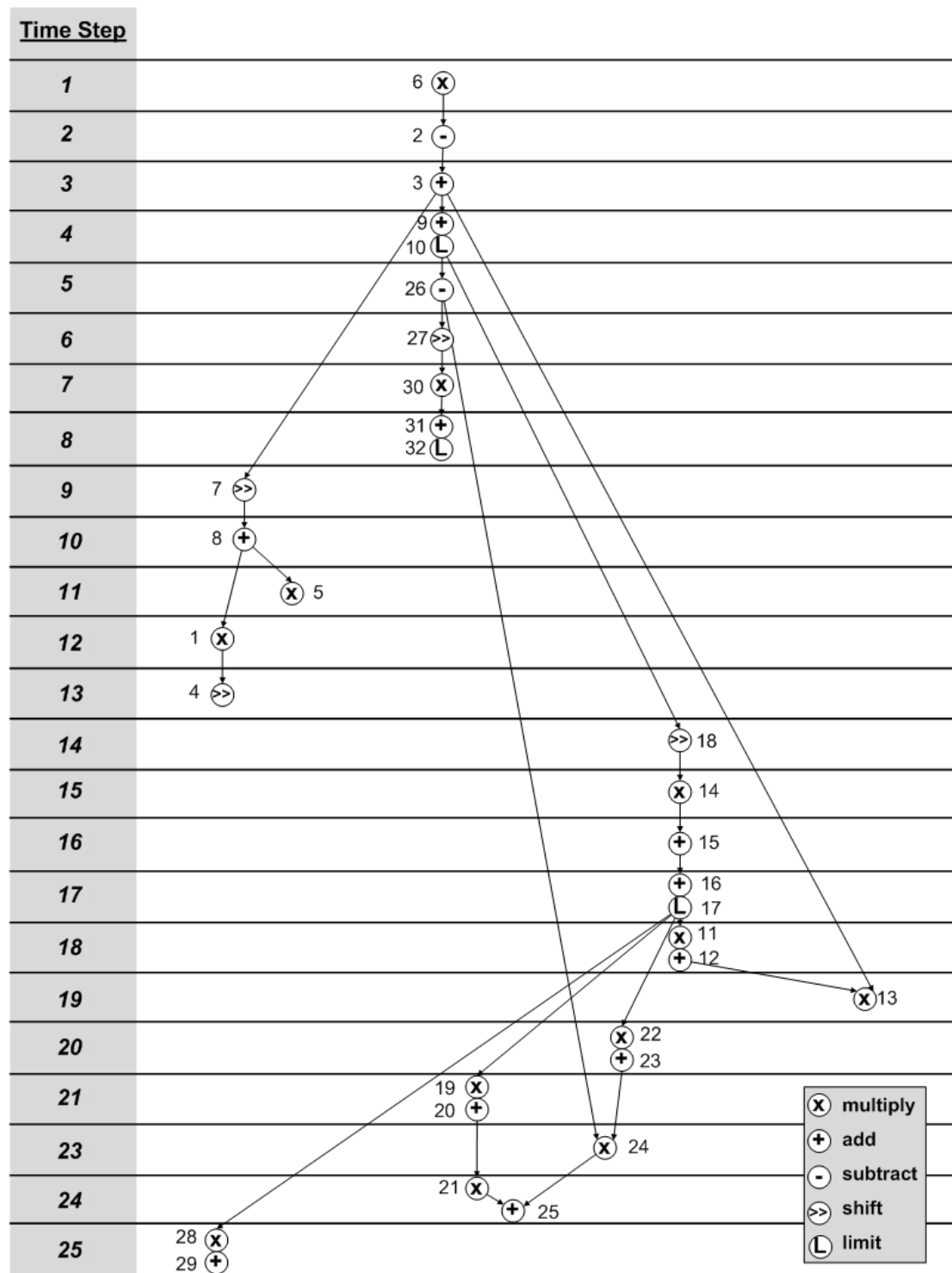


Figure 3.13. Scheduling of adaptive algorithm operations on single MAC DSP architecture.

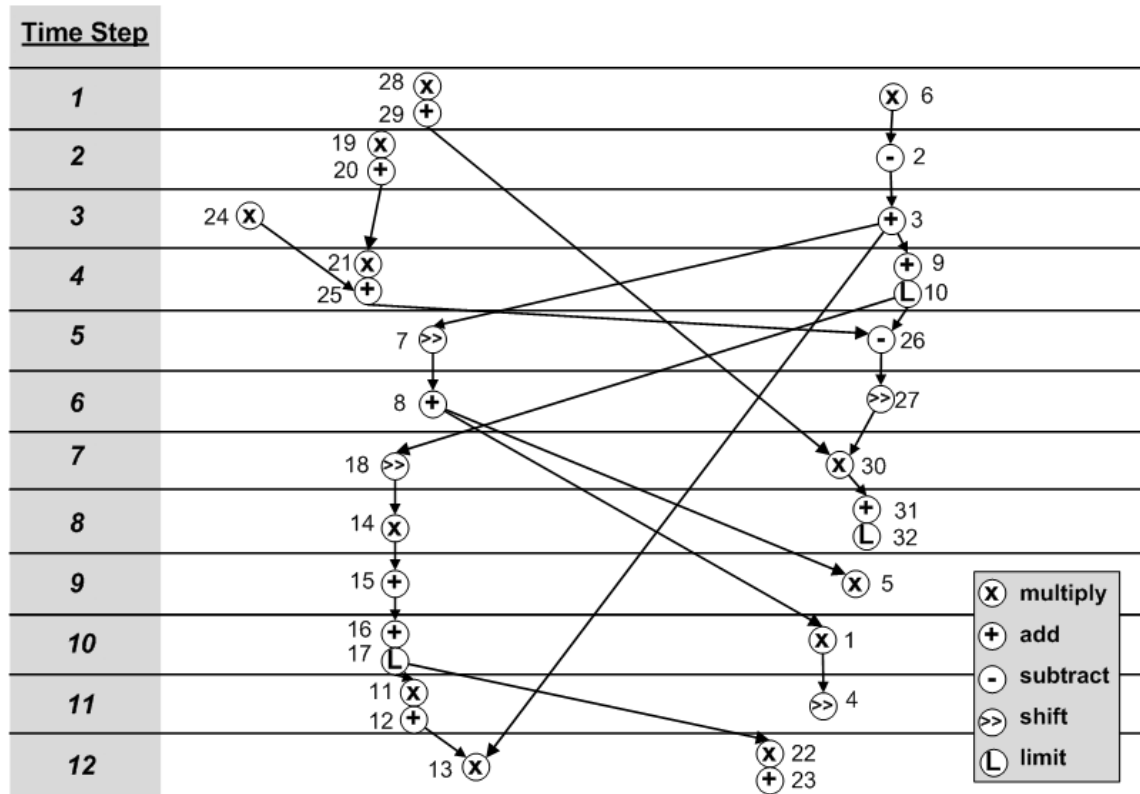


Figure 3.14. Scheduling of adaptive algorithm operations on dual MAC DSP architecture.

Figure 3.15 illustrates the ASAP schedule for the adaptive algorithm. This schedule determines that the minimum number of DSP clock cycles required to execute the adaptive algorithm with a DSP with an unlimited number of computational resources is ten.

In general, scheduling also impacts the number of storage elements required to execute the algorithm, thereby influencing the area of the design. However for the adaptive algorithm these differences in memory requirements are insignificant compared with the overall silicon area occupied by the DSP.

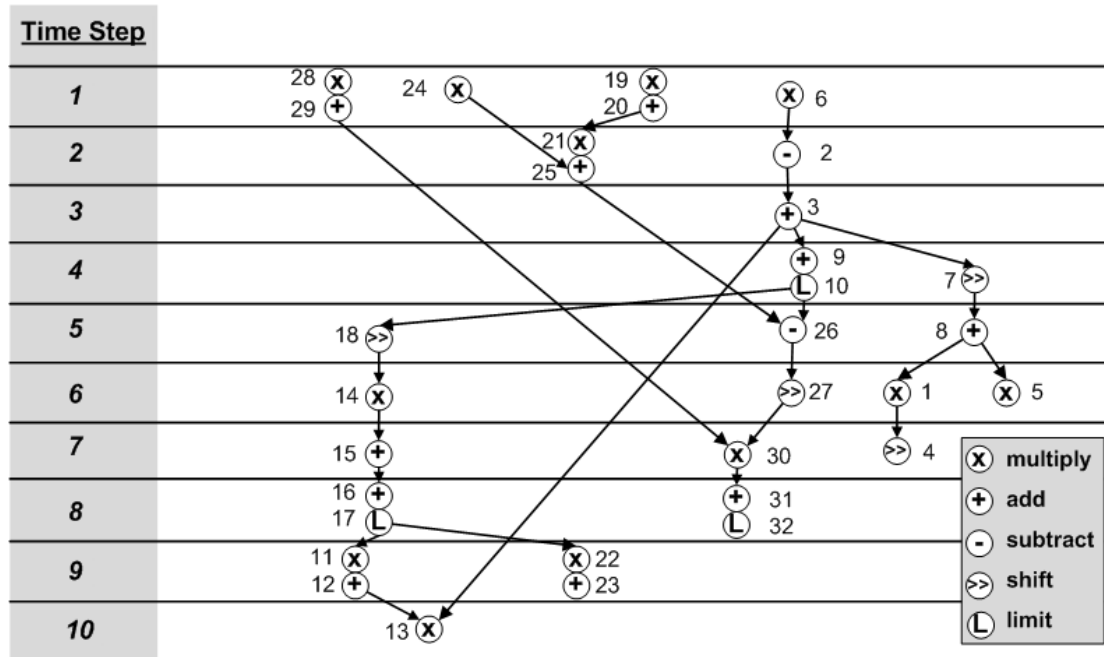


Figure 3.15. ASAP scheduling of adaptive algorithm operations.

3.4 Architecture Performance Comparisons

This section compares the execution of the previously analysed control algorithms on architectures with up to three MAC elements. The architectures are compared in terms of the number of DSP clock cycles required to implement the control algorithms for multiple voltage rails. Both multiple-issue VLIW DSPs and multicore DSPs are examined and they are also compared with a single MAC architecture. The multicore DSPs consist of multiple single-MAC cores. The five architectures that are compared are listed below:

- 1) Single MAC (SM)
- 2) Dual MAC, Two-way VLIW (V2)
- 3) Triple MAC, Three-way VLIW (V3)
- 4) Dual Core (C2)
- 5) Triple Core (C3)

3.4.1 Execution Times for Multiple Rails

The single and dual MAC schedules presented in Section 3.3 can be used to formulate an equation for the execution time of the control algorithms on multicore and multiple-issue processors. The number of clock cycles N required to implement an algorithm on a processor with D MAC elements for controlling P rails is given by:

$$N_{core} = C_{SM} \times \left\lceil \frac{P}{D} \right\rceil \quad (3.13)$$

for multicore architectures and

$$N_{issue} = P \times \max \left(C_{MIN}, \left\lceil \frac{C_{SM}}{D} \right\rceil \right) \quad (3.14)$$

for multiple-issue architectures, where C_{SM} is a constant which represents the minimum number of clock cycles required to implement a particular algorithm on a single MAC DSP and C_{MIN} is a constant which represents the minimum number of clock cycles required to implement a particular algorithm on a multiple datapath DSP with an unlimited number of datapaths. Note $\lceil x \rceil$ denotes the ceiling function of x where the value of the ceiling function at any number x is defined to be the smallest integer greater than or equal to x . The values of C_{SM} and C_{MIN} for the 2P2Z and the adaptive algorithms that were determined in Section 3.3 are summarised in Table 3.2.

Table 3.3 compares the number of DSP clock cycles required to implement the 2P2Z algorithm. The values show the increased computing power of the multiple MAC processors compared with the single MAC one. The two-way and three-way VLIW processors (V2 and V3) have identical execution times. This is due to the fact that there is a minimum number of clock cycles required to execute the 2P2Z algorithm, no matter how many datapaths the processor has.

Table 3.2. Minimum clock cycles required to execute 2P2Z and adaptive algorithms on single MAC and multiple datapath architectures.

Algorithm	Single MAC Execution (C_{SM})	Multiple Datapath Execution (C_{MIN})
2P2Z	8	5
Adaptive	25	10

Table 3.3. Comparison of execution times (in terms of DSP clock cycles) required to implement 2P2Z algorithm on a range of processors for single and multi-rail systems.

Rails	Processor Architectures				
	SM	V2	V3	C2	C3
1	8	5	5	8	8
2	16	10	10	8	8
3	24	15	15	16	8
4	32	20	20	16	16

The performance in terms of computation time of the 2P2Z algorithm in nanoseconds for a range of rails is given in Figure 3.16 for four architectures, assuming a 50 MHz DSP clock. Multi-rail systems with up to ten rails are examined here, though a typical application system of the DSP core would have between three and eight rails. The V3 architecture is omitted due to its poor performance-area ratio compared with the V2 architecture. The difference between the dual and triple MAC architectures increases for increasing numbers of rails. The zig-zag behaviour in the multicore graphs is due to the non-utilization of some of the cores when the number of rails is not an integer multiple of the number of cores. The resource utilisation is best when the ratio of rails to execution time is the largest, however in many instances it can be seen that the resource utilisation of the multicore architectures is non-optimal, i.e. when the number of rails is not an integer multiple of the number of cores.

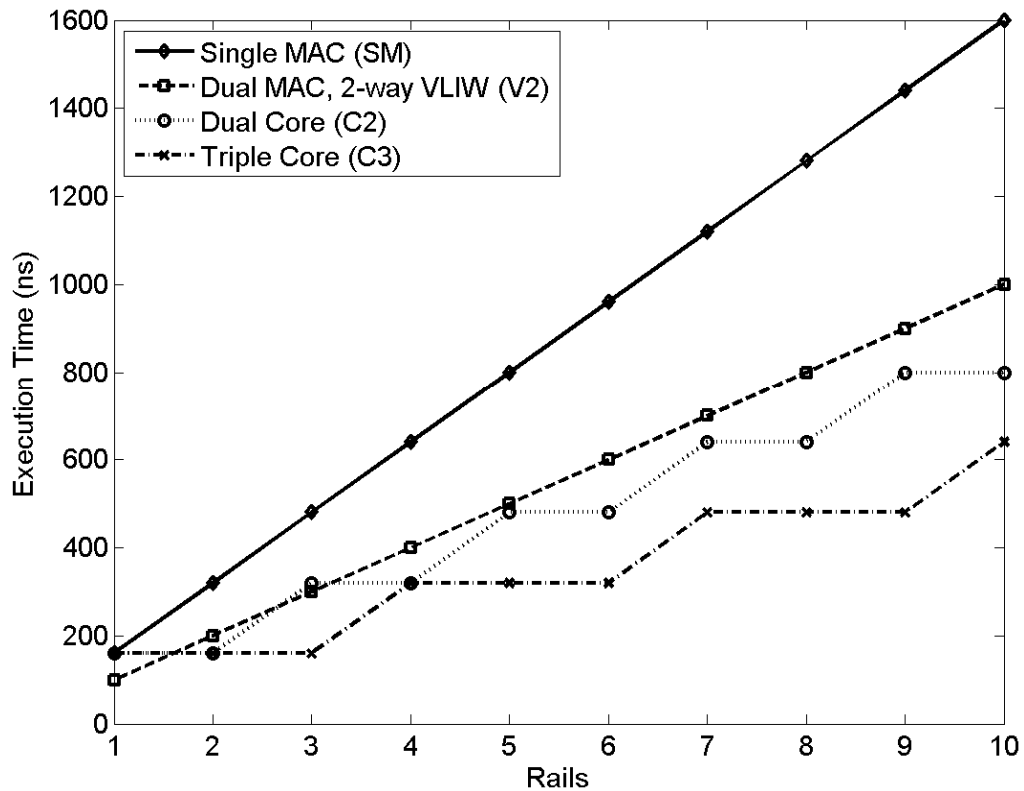


Figure 3.16. Execution time of 2P2Z algorithm on range of processor architectures for up to 10 rails.

The number of clock cycles each architecture requires to execute the adaptive algorithm is given in Table 3.4. The multiple datapath architectures bring a significant increase in execution speed of the adaptive algorithm compared with execution by the single MAC architecture. In this case the V3 architecture takes advantage of the parallelism in the adaptive algorithm so that it further reduces the execution time of the algorithm compared with execution on the V2 architecture. This trend is also visible in Figure 3.17, where the execution time of the adaptive algorithm on the architectures is compared assuming a 50 MHz DSP clock.

Table 3.4. Comparison of execution times (in terms of DSP clock cycles) required to implement adaptive algorithm on a range of processors for single and multi-rail systems.

Rails	Processor Architectures				
	SM	V2	V3	C2	C3
1	25	13	10	25	25
2	50	26	20	25	25
3	75	39	30	50	25
4	100	52	40	50	50

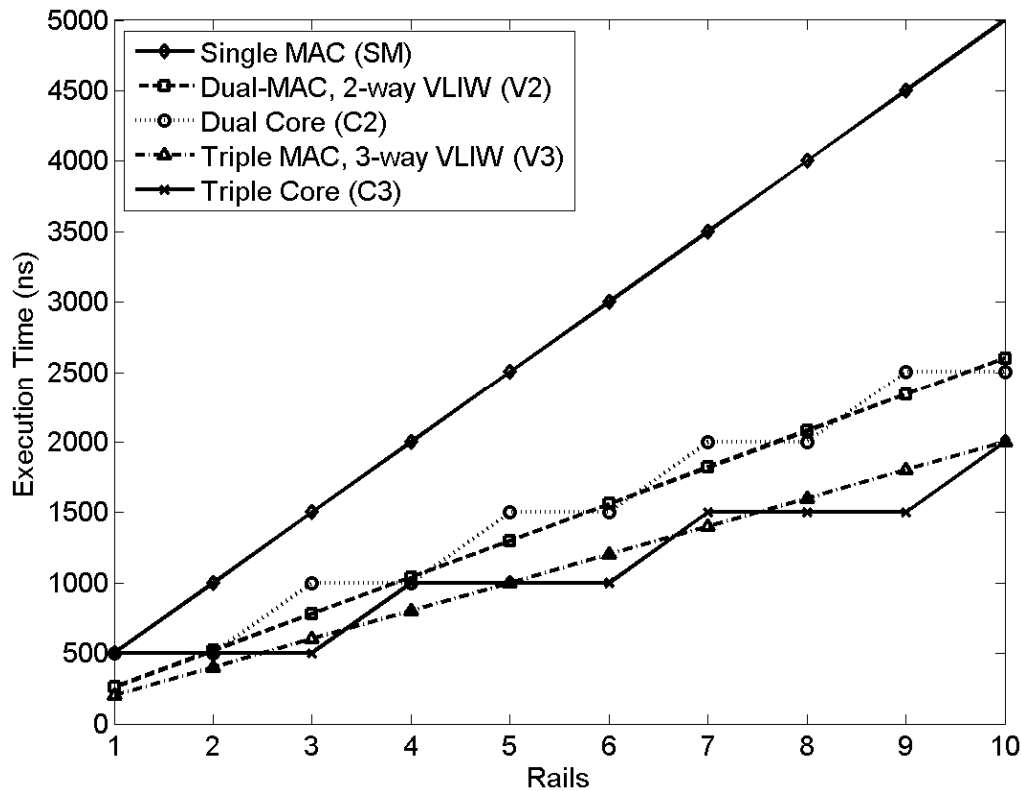


Figure 3.17. Execution time of adaptive algorithm on range of processor architectures for up to 10 rails.

3.4.2 Performance over a Range of Switching Frequencies

An important specification of a digital power controller is the maximum switching frequency of the SMPC that it can be applied to. This is determined by the execution speed of the control algorithm, if it is assumed that the duty cycle is updated once per switching period. If the algorithm can be executed more than once per switching period, then multiple power converters can be controlled. For multi-rail systems it is assumed here that the rails have identical switching frequencies when determining the maximum switching frequency SMPC that can be controlled.

Figure 3.18 shows the maximum switching frequency that can be handled by the various architectures executing the 2P2Z algorithm for a variable quantity of rails, assuming a 50 MHz DSP clock. From Figure 3.18 it can be seen that the architectures with multiple datapaths can be used to control up to eight DC-DC converters with switching frequencies in excess of 1 MHz, which is not possible with the single MAC architecture. The V2 architecture is the most effective at implementing compensators for single power converters, allowing SMPCs with switching frequencies over 8 MHz to be controlled. The C3 architecture allows control of the highest switching frequencies over the range of rails examined here for the most part, though the difference between the performance of the C3 architecture and the other architectures decreases for increasing number of rails.

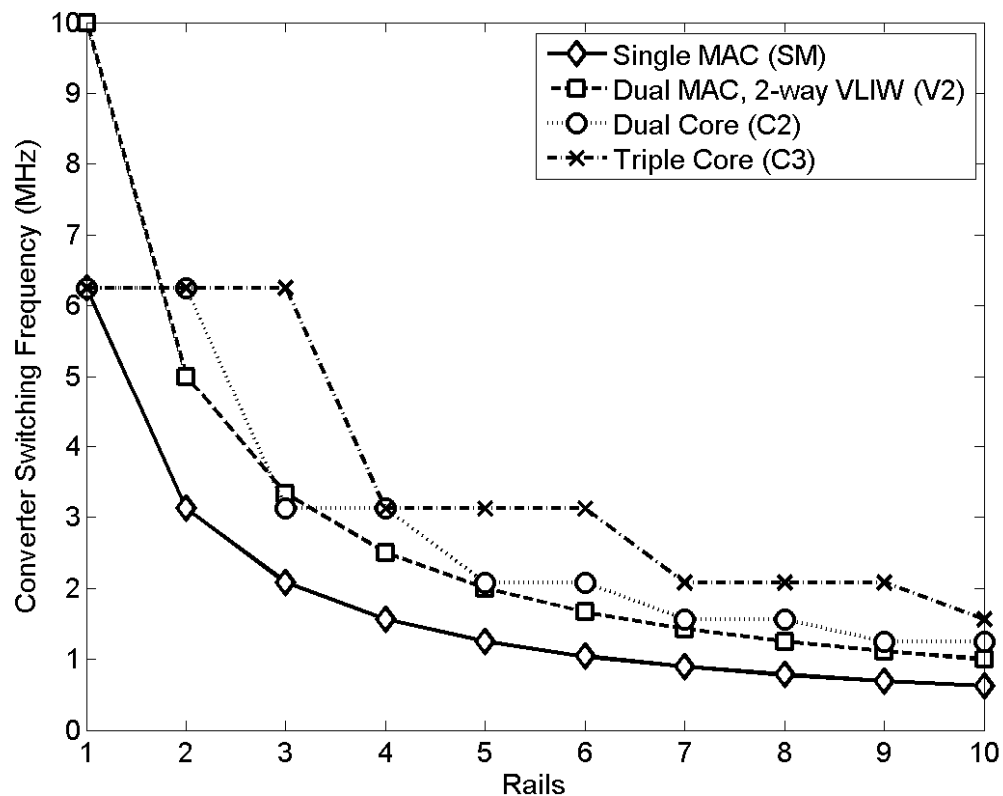


Figure 3.18. Maximum switching frequency controllable by range of processor architectures using 2P2Z algorithm for up to 10 rails.

Figure 3.19 shows the corresponding graph for implementation of the adaptive algorithm, again assuming a 50 MHz DSP clock. This graph determines the maximum switching frequency of a power converter controllable by a single MAC DSP to be in the region of 2 MHz while power converters with switching frequencies greater than 3.5 MHz can be controlled by the VLIW architectures. The limitations of the multicore architectures when executing the adaptive algorithm for single rail power converters with very high switching frequencies are highlighted here. It is also clear that the increased computational resources of the multiple datapath DSPs permit them to execute more algorithms within the same switching period as the single MAC processor, therefore enabling them to control more power converters. The single MAC processor can implement adaptive control for two rails with a 1 MHz switching frequency while the multiple datapath DSPs can execute the adaptive algorithm for at least four 1 MHz rails.

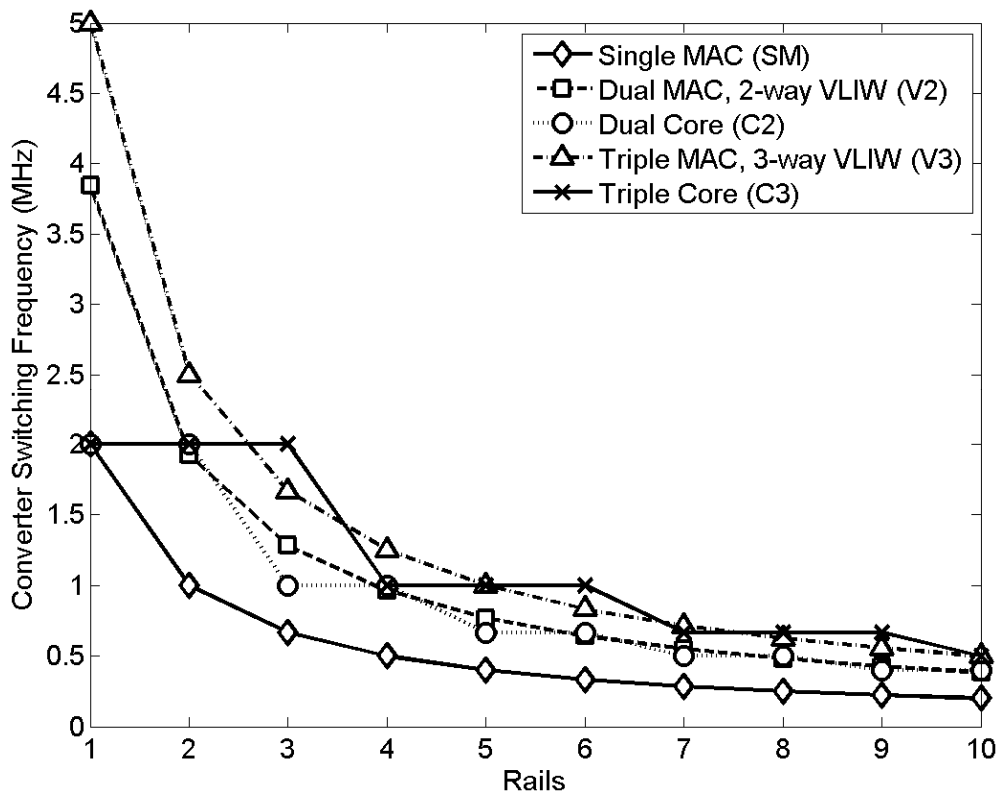


Figure 3.19. Maximum switching frequency controllable by range of processor architectures using adaptive algorithm for up to 10 rails.

3.4.3 Algorithm Comparisons

The computational requirements of the adaptive algorithm and the two-pole, two-zero algorithm are compared for the single and dual MAC DSPs in Table 3.5. The values highlight the greater computational demands of the adaptive algorithm. Figure 3.20 compares the number of power converters that can be controlled using the adaptive and 2P2Z algorithms for each of the processors. Examining Figure 3.20 it can be seen that the application of multiple datapath DSPs bring the execution time of the adaptive algorithm closer to that of the execution time of the 2P2Z algorithm when implemented on a single MAC architecture.

Table 3.5. Comparison of 2P2Z and adaptive algorithm execution times (in terms of DSP clock cycles) on single MAC, dual MAC and dual core architectures.

Architecture	2P2Z	Adaptive
SM	9	25
V2	6	13
C2	9	25

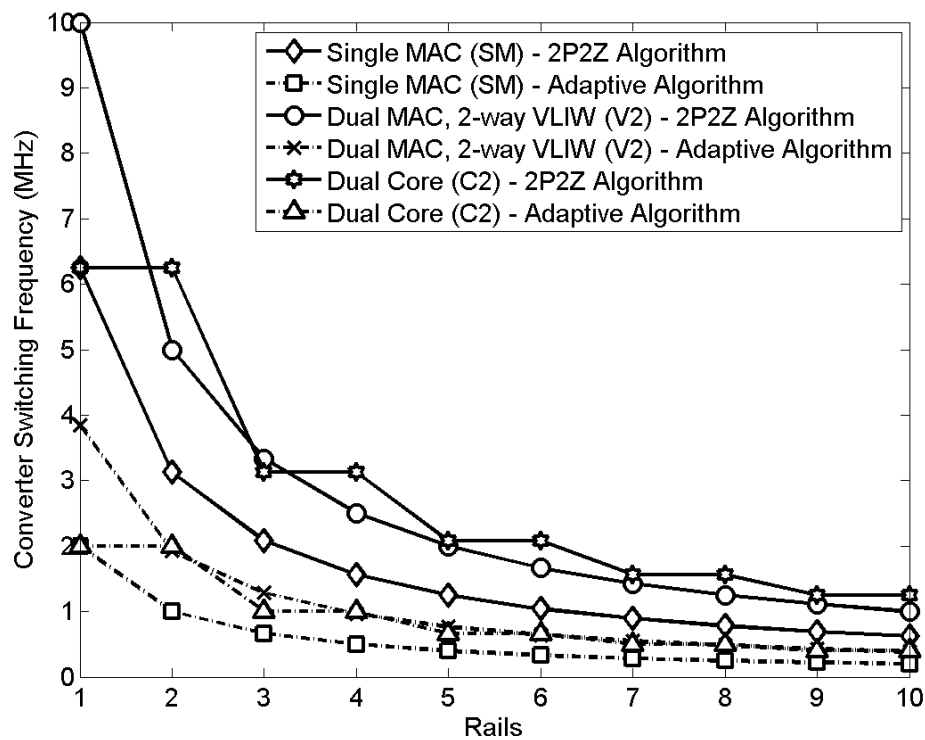


Figure 3.20. Maximum switching frequency controllability by single MAC, dual MAC and dual core architectures using 2P2Z and adaptive algorithms for up to 10 rails.

3.5 Summary

The hardware requirements for the implementation of multiple 2P2Z and adaptive algorithms for controlling multi-rail systems have been analysed in this chapter based on the comparison of a number of potential architectures.

The comparisons showed that multiple datapath DSPs are needed for adaptive algorithm execution to meet the performance requirements of SMPC applications that are typically controlled by single MAC 2P2Z implementations. The results highlighted the performance advantage of multiple datapath architectures, while also demonstrating their limitations due to precedence constraints. This was particularly obvious in the analysis of the execution time of 2P2Z algorithms on VLIW architectures with greater than two datapaths. Comparing the number of voltage rails that could be controlled by the various architectures also illustrated the relative computational power of each of the architectures. The multicore architectures examined were shown to have high resource utilisation only when the number of voltage rails being controlled was an integer multiple of the number of cores in the processor.

Altogether the results of the comparative investigation results serve as a guideline for the implementation of a flexible digital controller for a wide range of multi-rail applications. In the context of trends towards higher switching frequencies and more complex algorithms controlling SMPCs, the two-way VLIW dual-MAC processor is selected as a suitably flexible architecture to fulfil the requirements of a modern digital power controller. It shows higher resource utilisation for a wider range of multi-rail applications compared with the multicore architectures examined, while it is also superior to multiple-issue architectures with greater than two MAC units in this respect. The next chapter follows from this conclusion with an investigation of the requirements of the instruction set and microarchitecture of the dual-MAC processor in order to determine its optimal implementation in a multi-rail SMPC system.

Chapter 4

A Dual-MAC DSP Core Architecture

4.1 Introduction

Following on from the analysis of the computational requirements for the DSP-based multi-rail controller in the previous chapter, a more detailed examination of the hardware requirements of the 2-way VLIW dual-MAC architecture is presented in this chapter. Throughout the chapter, the characteristics of the control algorithms and the power converter application are taken into account in defining the architectural features of the processor. By examining each of elements of the DSP core individually, it can be ensured that each element satisfies the requirements of the multi-rail power converter application, while not adding superfluous functionality with an additional hardware cost penalty [93].

The datapath of the specialised DSP should include all of the computational elements required to execute the relevant algorithms for controlling SMPCs. In addition to the quantity of computational elements that make up the datapath of the processor, the resolution of the data values must also be defined. The limited wordlength affects the accuracy of the computations, therefore precautions must be taken to ensure the required level of accuracy is maintained while the algorithm is being executed.

Storage and movement of data is a key feature in every processor. Different types of data will be handled in different ways by the processor, for example coefficients, delayed error and duty cycle values and data from different loops. These must all be accommodated by the memory and data movement features of the architecture.

The program controller processes the instructions in the program memory of the processor to determine the routing of data through the datapath. The design of the program control hardware is therefore based on the type of operations described by the instruction set. In order to minimise algorithm execution time, the instruction set should be designed to ensure efficient use of the dual-MAC hardware by employing instructions that allow multiple computations or data movement and computation operations to be performed in a single clock cycle. Execution of multiple control algorithms is also necessary in order to facilitate multi-rail DC-DC converter control. The method of switching between these algorithms is also an important aspect to be considered in the design of the program controller.

Although each of the primary elements of the DSP core are addressed in this chapter the main focus is on incorporating the dual MAC datapath into a suitable processor architecture in order to be able to evaluate the impact on power converter performance when using a dual MAC DSP based controller instead of a standard single MAC based approach. The implementation of the main data memory source for the dual MAC datapath and the handling of multiple interrupt input signals is also focused on in the context of the execution of multiple control algorithms for multi-rail DC-DC converter systems. Although aspects such as the program memory, pipeline architecture and instruction decoder are commented on, their detailed implementation is outside the scope of this work.

This chapter is divided into four main sections. The first section gives an overview of the architecture of the processor and includes considerations on the input and output interface, while also discussing the instruction set of the processor. The other three sections address the three main hardware elements of the DSP core in detail. These elements are the datapath, the data memory and the program controller.

4.2 Processor Architecture

This section introduces the 2-way VLIW (Very Long Instruction Word) dual MAC DSP core that was selected in the previous chapter as the most suitable architecture for implementing digital control in multi-rail power converter applications based on the efficient utilisation of its primary computational components.

4.2.1 Processor Core Overview

Figure 4.1 illustrates the main hardware blocks of the proposed 2-way VLIW dual MAC DSP core that are discussed in this chapter. In order to implement this architecture, the 16-bit datapath is divided into two identical groups of computational elements that form two individual datapaths. These datapaths contain the necessary computational elements to execute multiply-accumulate (MAC) operations. Computational elements for shifting, limiting and performing logical operations are also required in most power control algorithms but are less frequently used throughout the execution of the algorithms compared with the MAC elements, as seen in Section 3.3. These computational elements are shared by the individual datapaths to avoid unnecessary duplication and under-utilisation of resources.

The processor has a Harvard-based memory architecture, similar to the majority of modern DSPs, whereby separate memory banks and buses are dedicated to data and program storage in order to reduce the execution time of operations which require multiple memory accesses [60]. The primary data memory consists of multiple register banks that facilitate fast access to the coefficients, constants and temporary variables during the execution of multiple control algorithms. Data can be transferred between the register banks and external memory using load and store operations.

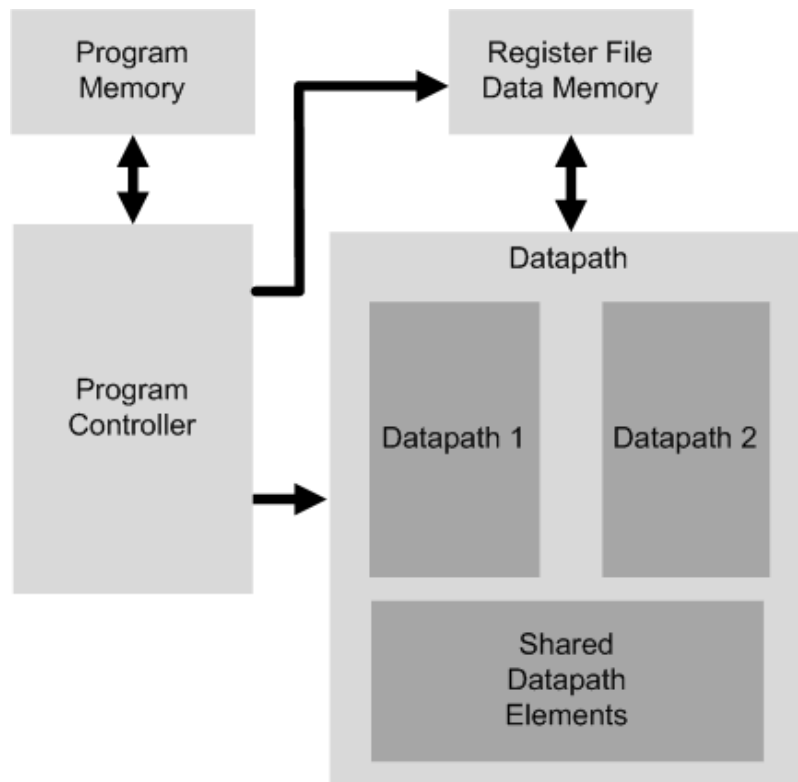


Figure 4.1. Main DSP core hardware elements.

The program controller contains all of the housekeeping functionality of the processor, which includes the program counter, instruction decoder, interrupt controller and some mode configuration registers. The interrupt controller contains a finite state machine that determines the exact mode of operation or context of the processor. This element is vital in executing multiple algorithms in a time-efficient manner and ensuring a fast response time when controlling multiple power converter rails.

4.2.2 Instruction Set

An assembly language-based instruction set was created for the specialised DSP core that only contains instructions that are relevant to the execution of power control algorithms. The instructions are listed by category in Table 4.1. Although the instruction set listed here contains thirty-seven instructions, resulting in a 6-bit

instruction opcode, a smaller instruction set of thirty-two instructions or less would be sufficient to facilitate the execution of standard digital control algorithms and simple background tasks. A 6-bit opcode was chosen in this case in order to implement additional instructions to accommodate the execution of a wider range of background tasks.

Multiplication, addition and MAC instructions are essential operations in the specialised power control instruction set architecture. Shifting is also required for scaling of results. Other operations which are included in the instruction set to deal with the limitations of digital representation include saturation and rounding operations. A compare instruction is also required in constraining the output duty cycle before passing it on to the DPWM.

Table 4.1. DSP core instructions.

Arithmetic & Logic	Multiplication & Accumulation	Load/Store	Other
ADD	MUL	LOADP	RNDSAT
ADDSAT	MULRND	LOADPI	SAT
SUB	MAC	LOADPD	SHIFTSAT
SUBSAT	MACSAT	LOADI	NOP
AND	MACU	LOADCI	MOV
OR		LOADD	CMP
XOR		STORES	JMP
NOT		STOREP	RETI
		STOREPI	
		STOREPD	
		STORECS	
		STORED	
		STORECD	
		STORECP	
		STORECPI	
		STORECPD	

Data move instructions are required to move data between registers for temporary storage of variables and also to transfer data between the processor's register file memory and external memory or peripherals including the ADC and DPWM. In order to facilitate the movement of data along the voltage error and duty cycle delay lines of the control algorithm's digital filter structure, two different MAC instructions are available. The basic *MAC* instruction simply multiplies the coefficient by the delay-line value and accumulates the result, which is necessary for end-of-delay-line values. The *MACU* instruction performs the same computation but also moves the delay-line value to its next position in the delay line.

In order to take advantage of the flexibility of the dual datapath architecture, the two datapaths can perform different operations in a single instruction cycle. Each very long instruction word may consist of the opcodes and operands for two independent operations, which are executed concurrently on the separate datapaths. Using instructions that can perform a number of operations in parallel in a single clock cycle therefore minimises the number of clock cycles taken to execute control algorithms.

Operations that involve the use of shared datapath elements may only be specified in one of the two operations that form a single instruction. For example, two shifting operations cannot be performed in parallel because there is only one shifter in the main datapath of the DSP. A number of operations are also included in the instruction set that may not be executed in parallel with any other operations. These include branch or program flow operations and also operations that have an immediate data value or memory location in the instruction word. It should be noted that there are no conditional branch instructions in the instruction set because these are not necessary in the execution of standard power control algorithms and simple background tasks, but they could be included in an alternative implementation if there was a requirement to execute tasks depending on certain conditions, for example in the execution of the multi-mode control algorithms described in Section 2.4.3. A full list of the DSP's instructions and details on their usage is provided in Appendix A.

4.2.3 I/O Interface and Peripherals

Other digital hardware components that are required in addition to the primary elements of the DSP core are illustrated in Figure 4.2. These components facilitate the flow of input/output data to the peripheral hardware elements of the DSP, which are required by the multi-rail SMPC application. The essential peripherals for digital power control are the ADC and the DPWM, which are mapped into the memory space of the processor. The processor may be interfaced to multiple ADCs in order to obtain the necessary data to implement a range of digital control algorithms including those for VMC, CMC, efficiency-based control, current estimation, etc. Other data, such as temperature sensor data and data from on-chip analogue comparators, may also be extracted by the processor for monitoring purposes. Such data might provide an input to event-based control algorithms, for example to deal with and recover from over-current conditions in the power converter being controlled.

In addition to the processor's program memory and register-file based data memory, 64 KB of additional data memory can be addressed by the DSP core's 16-bit address bus, which may be used for debug or data logging purposes. On-chip data-transfer between the DSP core and its associated peripherals is achieved using an appropriate System-on-Chip (SoC) bus interface [94]. In this case a simple custom SoC bus architecture was used, however in practise an open standard bus architecture such as the Advanced Microcontroller Bus Architecture (AMBA) Advanced Peripheral Bus (APB) could be used [95].

A communication interface is typically included in a digital power controller to facilitate programming of the device. It may also be used for error reporting or to feed back monitoring data at regular intervals to a supervisory controller IC which implements power management functions on a system level. Although the open-standard PMBus interface has specific power management features [96], many other communication interfaces are also found in commercially available digital power controllers [16, 18], such as UART, SMBus, I²C, etc. In this case a UART interface has been used, which also facilitates debugging of the processor's program code.

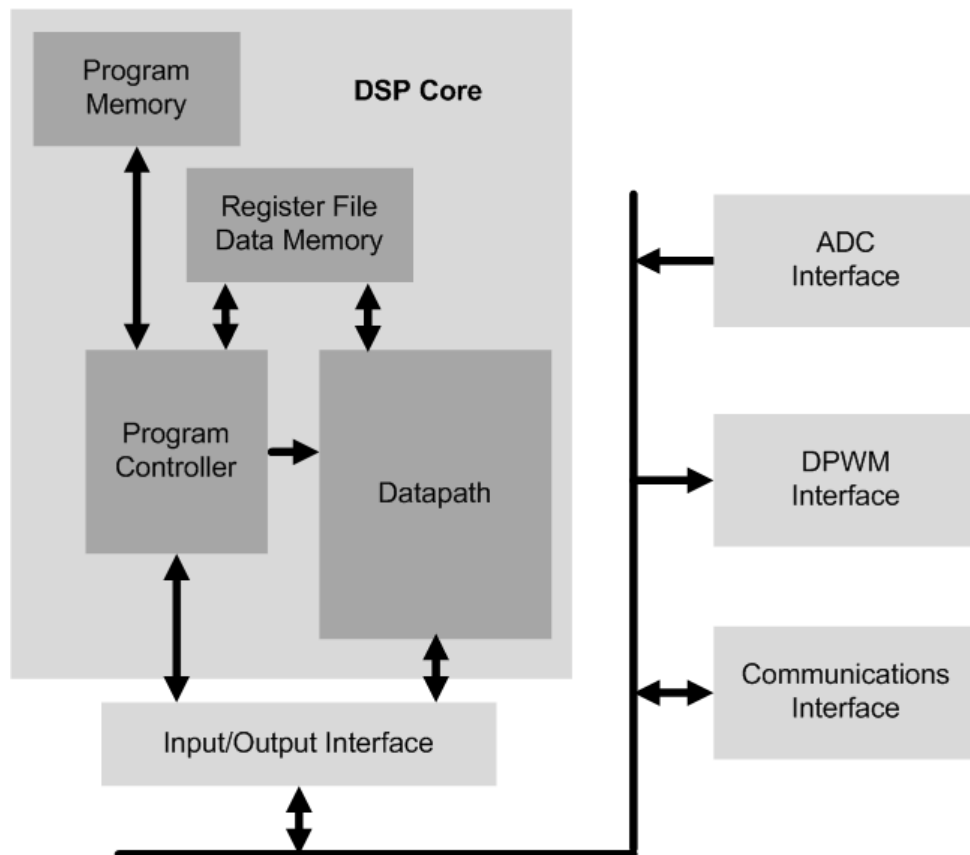


Figure 4.2. DSP core with I/O interface elements.

4.3 Datapath

The datapath component effectively consists of two interconnected datapaths which receive data from a common register file and share a number of functional elements, as illustrated in Figure 4.3. Data is transferred between the datapath and the peripheral elements of the digital controller system through the *dat_in* and *dat_out* ports. Immediate data, that is part of the instruction word of the DSP core, is also a data source for the functional units of the datapath.

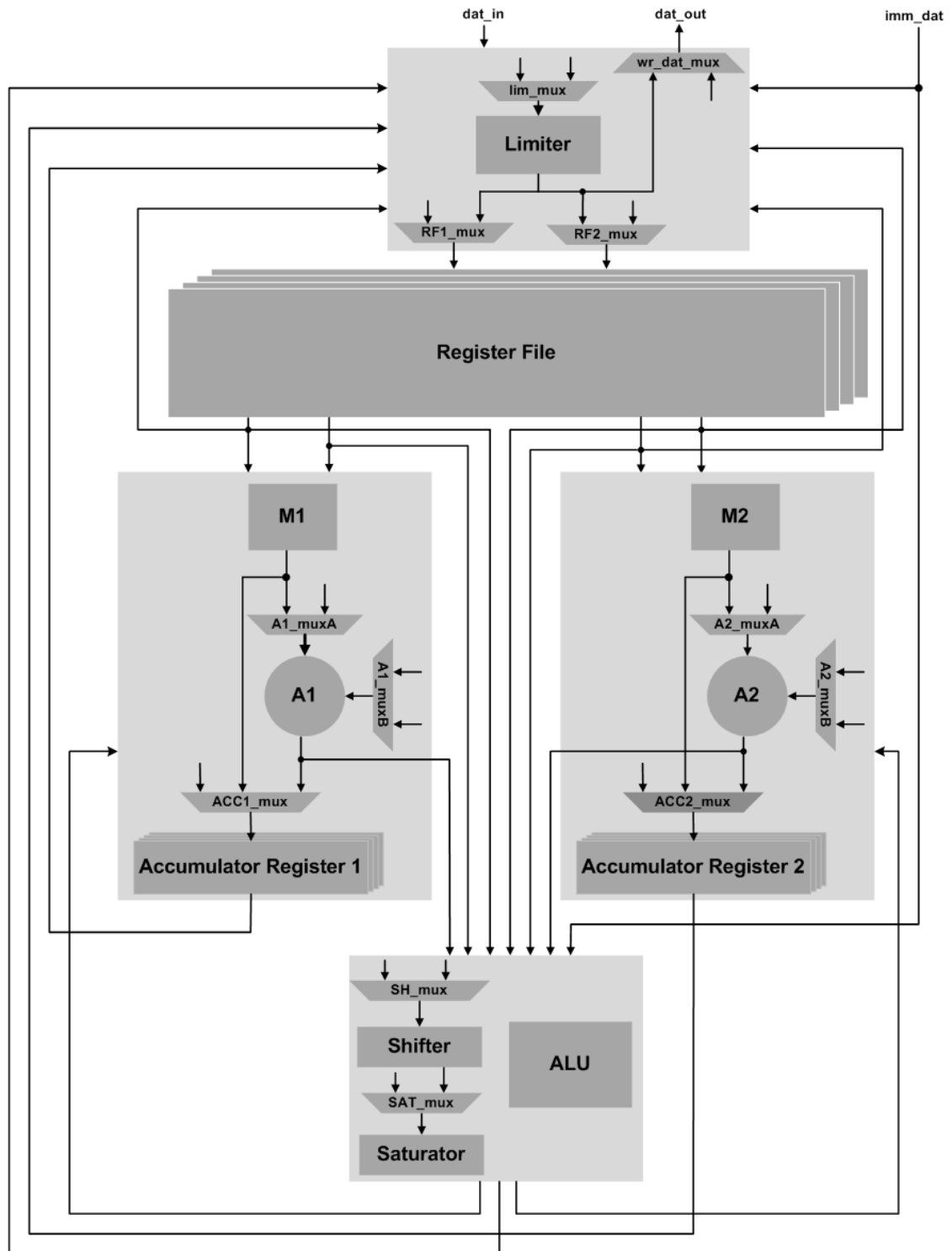


Figure 4.3. Main datapath elements.

4.3.1 Functional Units

Each of the interconnected datapaths has a MAC unit to execute multiplication, addition, subtraction, rounding or combined multiply-accumulate operations in a single clock cycle. These operations are the main operations found in power control algorithms. The execution time of the control algorithms is therefore reduced by carrying out two such operations simultaneously. Data movement operations may also be executed in parallel with computational operations for updating filter delay lines.

The sequencing of shifting and saturation operations in power control algorithms means that more than one of these operations never needs to be executed simultaneously. A barrel shifter and a saturator can therefore be shared between the two datapaths. Data can be shifted by up to 16-bits left or right per clock cycle using the shifter. The internal saturation logic limits the output of the accumulator registers to the maximum representable value to prevent errors due to overflow.

Other less frequently required functional elements are also shared between the two datapaths, for example the Arithmetic Logic Unit (ALU), which can perform bitwise logical AND, OR, XOR and NOT operations. The datapath also includes a separate limiter unit that can limit input, output and register data values to any given threshold in a single clock cycle operation. This facilitates limiting of the error from the ADC or limiting of the duty cycle that is applied to the DPWM in a straightforward manner.

The output data of most of these functional units is routed to the accumulator registers. Similar to the register file memory, there is a bank of accumulator registers for each datapath, where each register is associated with a separate context. Context switching is discussed in more detail in Section 4.5.2. The provision of multiple accumulator registers means that an individual accumulator can hold its value while switching between contexts, without the requirement to repetitively save and restore its value when exiting or entering a context.

4.3.2 Number Representation and Arithmetic Issues

The datapath uses fractional fixed-point two's complement arithmetic, similar to the majority of commercial digital power controllers [7]. The limited range of values being computed in power control algorithms means that the wide dynamic range of a floating-point DSP and its associated hardware overhead is not required.

The correct resolution choices of the inputs and outputs of the processor are vital for stable operation of the closed-loop digitally controlled power converter system. The input resolution is determined by the resolution of the ADC while the output resolution is determined by the resolution of the DPWM. The ADC resolution is typically in the range of ten to twelve bits. Conversion rate and ADC cost are the main deciding factors on the resolution of the ADC. In order to avoid creating limit cycle oscillations in the output voltage of the power converter the DPWM must have greater resolution than the ADC [38], usually by one or two bits, resulting in an 11- to 14-bit resolution. This ensures that there is a DPWM level in each ADC bin, thus eliminating the possibility of the output voltage continuously oscillating between DPWM levels while attempting to settle at the reference value. Additionally by keeping the internally fed-back duty cycle signal at a higher resolution than the duty cycle applied to the DPWM, the effects of limit cycling due to quantization errors are also reduced. This dithering of the duty cycle is achieved by maintaining a higher resolution in the duty cycle delay line compared with that in the voltage error delay line of a VMC compensator. The dual MAC architecture simplifies the implementation of this, whereby operations in the duty cycle delay line are executed on one MAC unit while voltage error delay-line operations are executed on the other MAC unit. In contrast this would require more operations to be executed sequentially on a single MAC architecture, thus leading to an overall increased algorithm execution time.

A standard 16-bit wordlength is preferred for the datapaths to facilitate straightforward interfacing with external digital hardware, though this resolution could be reduced if there were strict area requirements. The wordlengths of the main datapath buses are illustrated in Figure 4.4. Deviation from the 16-bit wordlength is necessary when performing multiplication. In order to maintain complete accuracy, $2N$ bits are required to represent the product when two N bit numbers are multiplied. If one of the multiplicands is not the largest negative number then all products can be represented using $2N - 1$ bits. The product will have one sign bit and $2N - 2$ magnitude bits. The product of two 16-bit numbers will therefore have one sign bit and thirty magnitude bits.

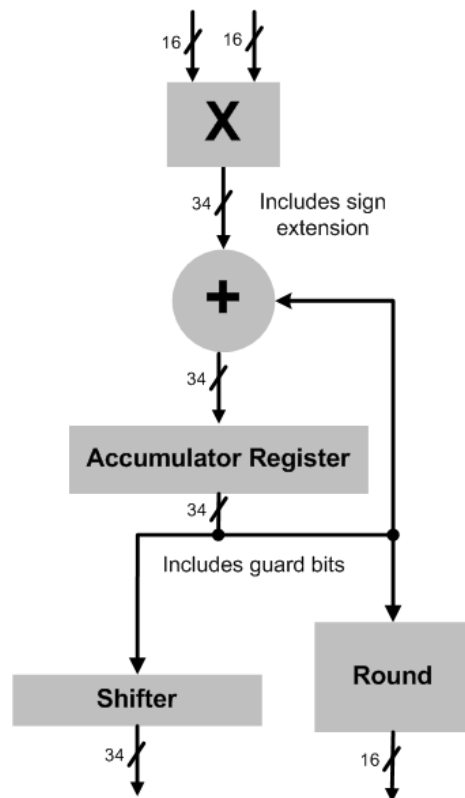


Figure 4.4. Wordlengths of main datapath buses.

Overflow is a problem that must be dealt with in datapaths based on fixed-point two's complement arithmetic. Using two's complement arithmetic allows multiple values to be accumulated correctly while only checking the final result for overflow, even if intermediate results exceeded the maximum possible value [60]. Overflow can be correctly detected by appending multiple additional sign bits, known as guard bits, to the accumulator [90]. If 2^N values have been accumulated, overflow can be correctly detected if the accumulator has at least N guard bits. Thus, using three guard bits means that eight values can be accumulated. Using more than three guard bits in the accumulator is not necessary since higher order filters are not usually used in the digital compensation of power converters. If overflow is detected saturation logic can be used to clamp the result at its maximum positive or negative value to prevent large errors in calculations. Saturation is also necessary to eliminate overflow caused by shifting operations. Taking into consideration the additional guard bits required for prevention of overflow, a 34-bit accumulator is proposed. In order to provide for the shifting of accumulator data, the wordlength of the datapath's shifter is also 34-bit.

The extended wordlength maintains accuracy during accumulation however such precision is not required in the final result. When storing the result in memory or passing it to the DPWM register it will be represented as a 16-bit value. Simply discarding the extra least significant bits (LSBs) results in errors as the truncated result is always less than or equal to the original value. By including rounding hardware on the path from the accumulator to the native wordlength registers, the errors can be reduced. The most common method of rounding in DSPs adds a one to the bit to the right of where truncation is to occur before discarding the extra bits [97]. This simple method does not completely eliminate errors and biasing but reduces them sufficiently for the purposes of power control algorithms and is therefore the preferred method. Less bias can be achieved by implementing convergent rounding, which only adds one to the bit to the right of where truncation is to occur if the bit to the left of that point is one [98].

4.4 Memory

All data relating to the execution of algorithms must be stored in memory throughout the execution of the algorithms and while waiting for the next data input to the system. Temporary result values must also be stored during the execution of the algorithms. The total amount of memory required depends on the complexity of the algorithms being implemented and on the number of algorithms. The storage of data must be such that it can be accessed quickly when needed so that it does not cause significant delay in computing duty cycle values. For this reason a Harvard memory architecture has been selected, where data and program memories are separate.

This section deals mainly with data memory issues while the program memory is discussed in Section 4.5.1. The processor uses a load-store instruction set architecture where the operands for most computational operations are from the register file data memory [87]. Another important issue relating to memory implementation is addressing. Addressing of data contributes to the hardware area of the processor in the form of instructions in the program memory as well as through any specialised address generation hardware that is used. An addressing scheme has been chosen that accounts for the amount of memory to be addressed as well as the frequency and sequencing of memory accesses in power converter control algorithms.

4.4.1 Register File Memory

The register file, which acts as the main data memory, consists of four banks of thirty-two 16-bit registers and is the primary storage location of the operands required during the execution of multiple control algorithms. It provides fast access to the coefficients, delay-line values and intermediate results and facilitates multiple parallel read and write operations in each clock cycle. The register file memory architecture eliminates the need to frequently access a large data memory, which would reduce the time available to execute computational operations.

The processor has a concise addressing scheme in spite of the large volume of data that needs to be accessed. This is achieved by dividing the register file memory into multiple register banks and restricting access to only a single pre-selected bank in any single instruction. Thus by limiting access to a bank of thirty-two registers, a register address length of only five bits is required in each instruction, which contributes towards minimizing the size of the processor's program memory and instruction decoding hardware.

Three of the register banks are dedicated to the storage of data associated with the control algorithm for an individual power converter, while the fourth stores data pertaining to the execution of the background code. Using greater than four register banks would lead to increased multiplexer delays and would more likely result in under-utilisation of register banks due to there being fewer power converters compared with register banks.

Although allocating sixteen registers per bank with a 4-bit register address would provide sufficient storage for the data associated with the execution of two-pole, two zero algorithms, it would not be sufficient to store the significantly larger quantity of data associated with the execution of more complex algorithms. In contrast with the two-pole, two-zero algorithm implementation which requires at least twelve registers, the adaptive algorithm described in Section 3.3.2 requires in excess of twenty-two registers to store its associated data variables. For this reason the next largest register bank size of thirty-two registers was chosen, corresponding to a 5-bit register address, which is adequate for use with more data-intensive control algorithms. If the number of registers per register bank far exceeds the number required by a single algorithm, then the data for more than one algorithm can be assigned to a single register bank.

Usually only the data associated with one power converter needs to be accessed while the algorithm for that power converter is being executed, therefore frequent execution of register bank switching instructions is not required, which would have a negative impact on algorithm execution speed. The processor actually allows a different bank to be selected automatically before executing the algorithm for the next power

The number of input and output ports of the memory also needs to be considered. The dual datapath architecture can provide up to four operands per clock cycle as illustrated in Figure 4.3. An architecture that allows simultaneous reading of operands and writing of results is desirable as it reduces algorithm execution time. Two write ports and four read ports are needed to accommodate the memory accesses when two *MACU* (multiply-accumulate with update) operations are executed simultaneously. The *MACU* instruction requires reading two operands and writing one of the operands to the next location in the memory file. Two *MOV* (register to register move) operations can be executed simultaneously where data can be written from one register to another register in the same register file. The instruction being executed in Datapath 1 has priority over instructions executing in Datapath 2 in the case where an assembly instruction attempts to write two different data values to the same register.

4.4.2 Addressing

The benefits of the register file based memory architecture have been highlighted in the previous section. The register file memory architecture allows direct access to an operand by referring to it by its specific name or address in the instruction word, i.e. by performing register-direct addressing, rather than having to specify a longer external memory address. When power control algorithms are being executed, the result of the majority of the computational operations will be added to the result of the next operation, thus the default destination of the result for most operations is the accumulator register. Other data values may be moved from one register to another, if required, by using the *MOV* instruction where both the source and destination register are specified. Previous error and duty cycle values must be moved within the register file so that they are accessed correctly in consecutive iterations of the control algorithm. This updating of the delay lines of the algorithm can be done in parallel with computations by using *MACU* instructions, whose operands are automatically written to the next location in the register file memory following the computation. Constants may be loaded into the register file memory using immediate addressing, whereby the 16-bit data value is specified in the 32-bit instruction word.

Only duty cycle results and error values from the ADC need to be moved between the register file memory and external memory locations during algorithm execution. This may be performed using a variety of addressing modes. Direct addressing, indirect addressing and indexed addressing methods are all facilitated by the processor architecture for convenience when programming. For direct addressing the 16-bit source or destination memory address is specified in the 32-bit instruction word. In indirect addressing mode the memory address is located in the processor register specified in the instruction word. For indexed addressing the base address located in a dedicated pointer register is combined with a 5-bit offset in the instruction word to form the required memory address.

4.5 Program Controller

Figure 4.6 illustrates the main elements of the program controller, which manage execution and interrupt control. In this case program memory and instruction decoding logic are also considered part of the program controller. A small amount of logic is also included to implement the addressing scheme which has been discussed in Section 4.4.2.

4.5.1 Execution Control

The program counter together with a number of special function registers and interrupt inputs determine instruction sequencing during execution of a program. Table 4.2 lists the main configuration or special function registers and describes their purpose. The output of the program counter is the program memory address of the next instruction to be executed. The program memory address is incremented by one in each clock cycle unless a program flow operation is taking place, in which case an alternative jump, return or ISR address is selected depending on the operation in question.

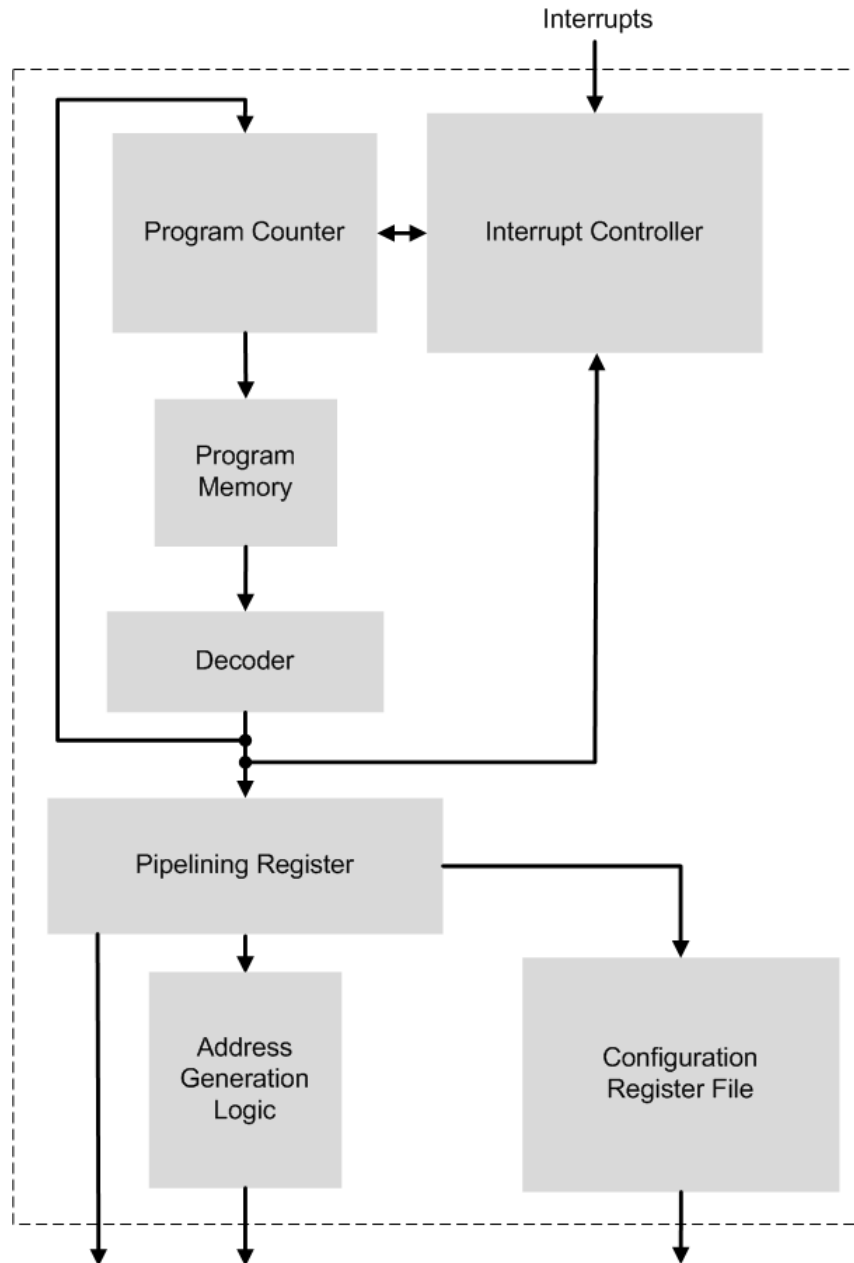


Figure 4.6. Main program controller elements.

On start-up, a default program is automatically loaded into the 32-bit program memory from a small on-chip ROM. The program controller's decoder interprets the binary-coded instructions in the program memory to generate the signals that allow the specified control algorithms to be executed on the processor's datapath.

Table 4.2. DSP core special function registers.

Register	Function	Wordlength	Read/ Write
pc_reg	Program counter	16	R
jmp_adr1	Address of Interrupt 1 in program memory	16	R/W
jmp_adr2	Address of Interrupt 2 in program memory	16	R/W
jmp_adr3	Address of Interrupt 3 in program memory	16	R/W
jmp_adr4	Address of Interrupt 4 in program memory	16	R/W
jmp_adr5	Address of Interrupt 5 in program memory	16	R/W
jmp_adr6	Address of Interrupt 6 in program memory	16	R/W
jmp_adr7	Address of Interrupt 7 in program memory	16	R/W
jmp_adr8	Address of Interrupt 8 in program memory	16	R/W
jmp_adrBG	Address of background code in program memory	16	R/W
jmp_adrD	Default program memory jump address	16	R/W
ptr_reg	Default data memory pointer address	16	R/W
adc_lim_reg	Limit threshold for input data	16	R/W
pwm_lim_reg	Limit threshold for output data	16	R/W
status_reg	DSP core status	16	R
base_reg	Base address for indexed addressing	11	R/W
bank_sel_reg	Active register bank	2	R/W
contxt_reg	Current program mode	2	R
adc_lim_mode	Input data limit enable	1	R/W
pwm_lim_mode	Output data limit enable	1	R/W
int1_pend	Pending Interrupt 1 indicator	1	R
int2_pend	Pending Interrupt 2 indicator	1	R
int3_pend	Pending Interrupt 3 indicator	1	R
int4_pend	Pending Interrupt 4 indicator	1	R
int5_pend	Pending Interrupt 5 indicator	1	R
int6_pend	Pending Interrupt 6 indicator	1	R
int7_pend	Pending Interrupt 7 indicator	1	R
int8_pend	Pending Interrupt 8 indicator	1	R

Figure 4.7 illustrates the structure of the program code that is loaded into the program memory. The program consists of initial instructions for loading registers and setting up ADC and DPWM interfaces as well as the instructions for executing the control algorithm operations. In this case three different algorithms are used to control three power converters. Alternatively, if the same algorithm is used for controlling each of the SMPCs, only program code for one power converter needs to be included in the memory. The multiple-banked register file structure allows unique input data and coefficient values of each power converter to be applied to the same program code that specifies the required control algorithm.

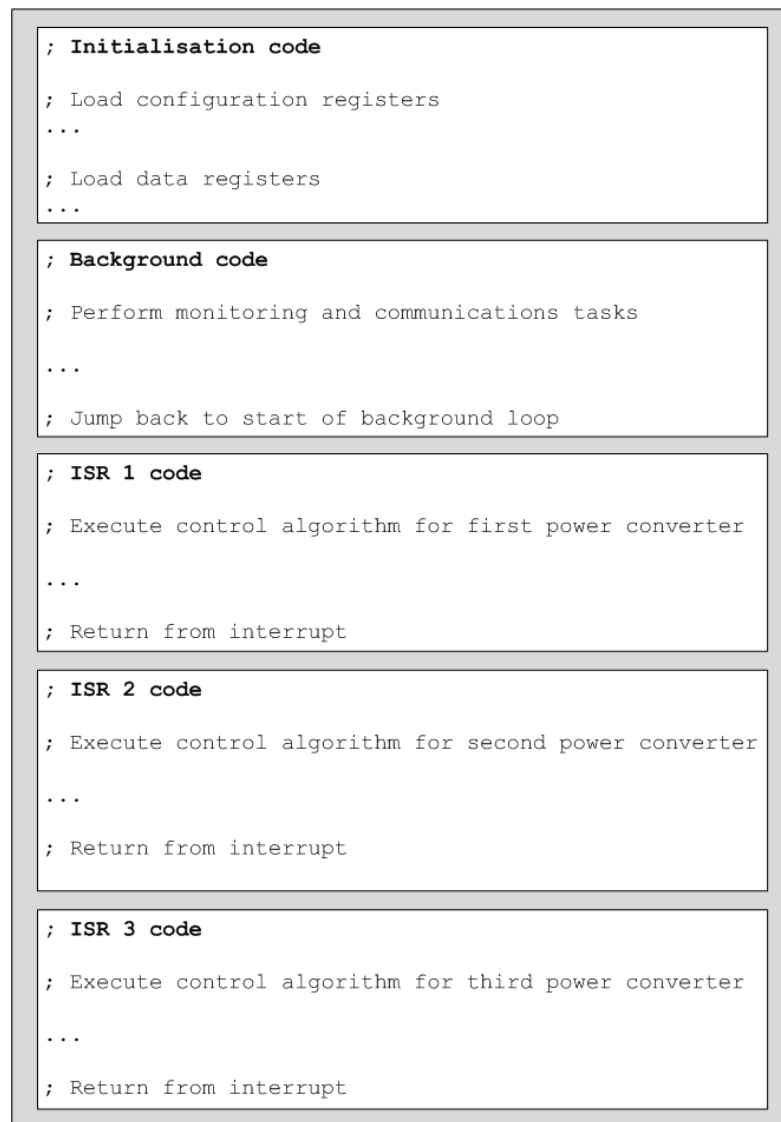


Figure 4.7. DSP program code structure for controlling three power converters.

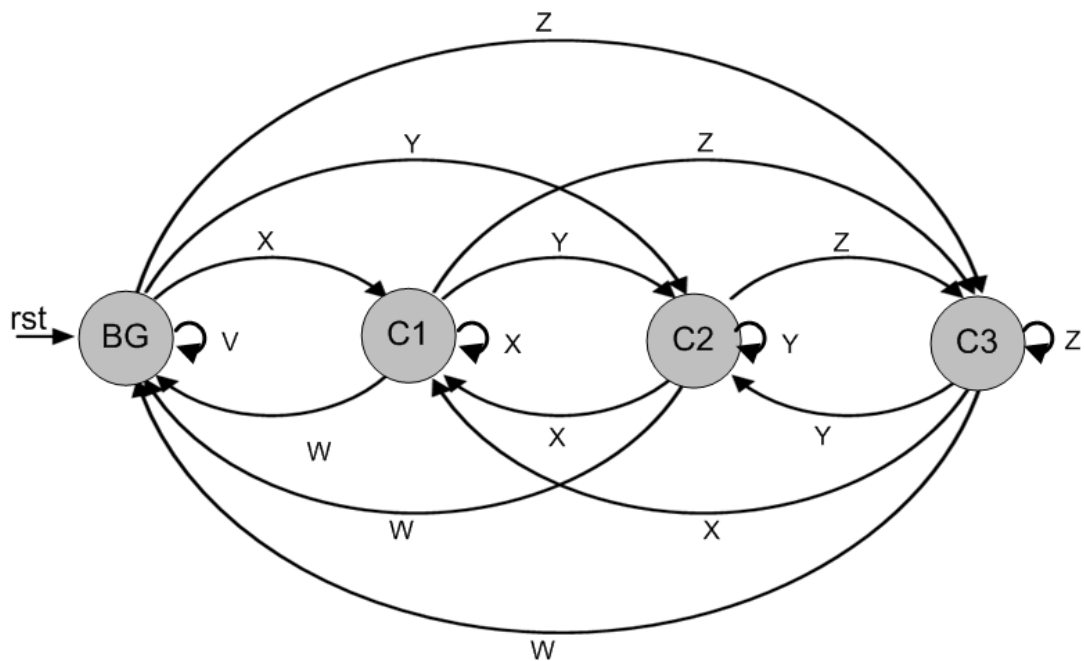
4.5.2 Interrupt Control

The interrupt control hardware governs how the computational resources of the DSP are time-multiplexed to execute control algorithms for multiple independent power converters. Each control algorithm is associated with an interrupt service routine whose execution is triggered by an external interrupt signal, which comes from the DPWM. The DSP has eight interrupt inputs, *Int1* to *Int8*, each of which is of the form of a periodic pulse, which is active for only one DSP clock cycle for each PWM switching cycle. The pulse occurs at a pre-configured offset from the beginning of the PWM switching cycle. The interrupt signals could alternatively be supplied by a timer or other synchronization hardware.

The interrupt controller does not permit an ISR to be interrupted by another interrupt signal, hence the ISR must run to completion before the next ISR can begin. The next ISR can therefore not begin until a return from interrupt (*RETI*) instruction has been executed at the end of the ISR. Interrupts that occur simultaneously are serviced in terms of their fixed priority setting. *Int1* has the highest priority while *Int8* has the lowest priority. Lower priority pending interrupts are serviced when the higher priority ISR has completed. This allows SMPCs with different switching frequencies to be controlled by the DSP. This is explored in more detail in Chapter 6.

Context switching delay is an important factor in the interrupt-based execution of multiple control algorithms when switching between the program code being interrupted and the ISR. This delay is caused by carrying out data movement operations which involve saving the data contained in the register set so that it will not be overwritten by new data during the execution of the ISR [90]. The data must be restored to the appropriate registers of the register set after execution of the ISR. In this case, a finite state machine controls context switching operations in order to minimise context switching delay. The processor has four pre-defined contexts (*C1*, *C2*, *C3* and *BG*), with a register bank associated with each one of them. After executing an algorithm for the control of one power converter, the processor is prepared for control of the next power converter by the FSM, a simplified version of

which is illustrated in Figure 4.8. The Boolean expressions determining the transitions between the states are provided in Figure 4.8, whereby logical AND is denoted by a dot symbol (.), logical OR is denoted by a plus symbol (+) and logical negation is denoted by a bar symbol ($\bar{}$) overhead the Boolean variable. The input interrupt signals modify the output of the FSM to determine the context of the processor. The FSM thus selects the appropriate bank of the register file before any other operation of the ISR is executed, whereby two ISRs share each register bank. The FSM also independently determines the program memory address of the first instruction of the control algorithm to be executed based on the status of the input interrupt signals.



$$V = \text{RETI} \cdot \overline{(\text{Int1} + \text{Int2} + \text{Int3} + \text{Int5} + \text{Int6} + \text{Int7})}$$

$$W = \text{RETI} \cdot \overline{(\text{Int1} + \text{Int2} + \text{Int3} + \text{Int5} + \text{Int6} + \text{Int7})} + \text{Int4} + \text{Int8}$$

$$X = \text{RETI} \cdot (\text{Int1} + (\text{Int5} \cdot \overline{(\text{Int2} + \text{Int3} + \text{Int4})}))$$

$$Y = \text{RETI} \cdot (\text{Int2} \cdot \overline{\text{Int1}} + (\text{Int6} \cdot \overline{(\text{Int1} + \text{Int3} + \text{Int4} + \text{Int5})}))$$

$$Z = \text{RETI} \cdot (\text{Int3} \cdot \overline{(\text{Int1} + \text{Int2})} + (\text{Int7} \cdot \overline{(\text{Int1} + \text{Int2} + \text{Int4} + \text{Int5} + \text{Int6})}))$$

Figure 4.8. Interrupt controller finite state machine.

When no control algorithm is being executed by the DSP, it operates in background mode (*BG*), which provides the option of executing monitoring and communications instructions in a loop so that the processor does not remain idle. A *RETI* instruction at the end of an ISR causes the processor to enter background mode if no algorithm needs to be executed immediately. ISRs triggered by *Int4* or *Int8* also use the background mode register bank.

4.5.3 Instruction Encoding

The dual MAC processor has a 32-bit instruction word where the addresses of four separate registers may be included, as illustrated in Figure 4.9. The first two register addresses provide the operands for the operation defined by *opcode1*, which is executed on the processor's first datapath. Addresses 3 and 4 provide the operands for the second operation to be executed in parallel on the second datapath. Only a limited area of memory can be addressed by any one instruction because the memory is divided into banks so that the lengths of the register addresses can be significantly reduced. The register bank is selected before starting execution of the control algorithm. A second type of instruction is illustrated in Figure 4.10 which contains a single register address and also a 16-bit field, which may hold a memory address or a 16-bit data word. This double-length instruction has only one opcode as it determines the behaviour of both datapaths.

31-26	25-20	19-15	14-10	9-5	4-0
opcode1	opcode2	register address1	register address2	register address3	register address4

Figure 4.9. Encoding for standard instruction with two sub-instructions.

31-21	20-5	4-0
double-length instruction opcode	memory address/data	register address

Figure 4.10. Encoding for double-length instruction.

4.5.4 Pipelining

Pipelining speeds up the execution of control algorithms by breaking down individual instructions into tasks. Tasks relating to consecutive instructions may then be executed in parallel. The number of main tasks each instruction is divided into determines the pipeline depth of the DSP core.

One problem relating to pipelining is the latency that can occur when program flow instructions are executed, for example the delay after an interrupt is triggered or a branch instruction is decoded. The number of clock cycles delay caused by these pipeline hazards is usually the same as the pipeline depth [97]. This is relevant to the execution of multiple power control algorithms because such instructions are executed frequently on entry and exit from the relatively short interrupt service routines. For example if an algorithm consists of six instructions, for a three-stage pipelined processor an extra three clock cycles of latency are produced by a single branching instruction. It is therefore essential to consider the trade-off between the performance improvements and the delays incurred in implementing pipelining in the DSP core [99]. Minimizing the pipeline latency is vital in ensuring efficient use of processor resources over the execution time of the multiple power control algorithms.

Dedicated logic is included in the DSP core's program controller to quickly detect instructions that cause pipeline hazards, in order to reduce the latency. The use of a short pipeline simplifies the implementation of this specialised logic. For this reason a pipeline depth of three is used in the DSP core as illustrated in Figure 4.11, where it can be seen that all of the instructions consist of three tasks: fetch, decode and execute. In the first stage the instruction is fetched from the address in the program memory determined by the program counter or other pointer registers. The second stage involves decoding the instruction that has been fetched and loading the required operands from the register file into operand registers. In the third stage the decoded instruction is executed on the DSP's datapath and the result is stored to the accumulator registers or the register file memory.

Figure 4.12 illustrates the effect of a *JMP* instruction on the DSP core’s pipeline, comparing it with a standard three-stage pipeline where the entire pipeline is flushed on detection of the *JMP* instruction. Similarly Figure 4.13 shows how the benefit of including dedicated detection logic impacts the movement of instructions along the pipeline for the proposed DSP core and a standard processor when an interrupt is triggered.

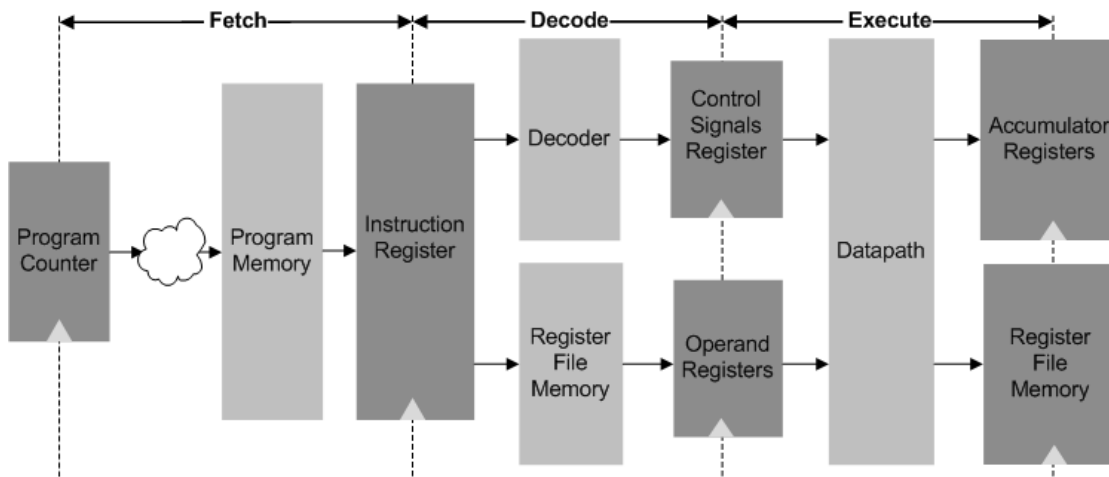


Figure 4.11. DSP core pipeline stages.

Fast jump response

Cycle	1	2	3	4	5	6	7	8
Fetch	MUL	MAC	JMP	ADD
Decode	...	MUL	MAC	JMP	ADD
Execute	MUL	MAC	JMP	ADD

Standard pipeline-flush jump response

Cycle	1	2	3	4	5	6	7	8
Fetch	MUL	MAC	JMP	SUB	—	ADD
Decode	...	MUL	MAC	JMP	—	—	ADD	...
Execute	MUL	MAC	JMP	NOP	NOP	ADD

Figure 4.12. JMP instruction in DSP core instruction pipelines.

Fast interrupt response

Cycle	1	2	3	4	5	6	7
Fetch	MUL	MAC	MACU	LOADP	ADD
Decode	...	MUL	MAC	—	LOADP	ADD	...
Execute	MUL	MAC	NOP	LOADP	ADD

↑ Interrupt detected (between cycle 2 and 3)
 ↑ First ISR instruction fetch (between cycle 3 and 4)

Standard pipeline-flush interrupt response

Cycle	1	2	3	4	5	6	7
Fetch	MUL	MAC	MACU	—	LOADP	ADD	...
Decode	...	MUL	MAC	INT	—	LOADP	ADD
Execute	MUL	MAC	INT	NOP	LOADP

↑ Interrupt detected (between cycle 2 and 3)
 ↑ First ISR instruction fetch (between cycle 4 and 5)

Figure 4.13. Effects of an interrupt on DSP core instruction pipelines.

4.6 Summary

This chapter has presented a comprehensive specification of a 2-way, VLIW dual MAC DSP core for multi-rail SMPC applications. The arguments for each aspect of the DSP core's architecture are relevant for a wide range of multi-rail DC-DC converter applications where the switching frequency, number of rails and type of control algorithms may vary.

The parallel multiply-accumulate approach presented in this chapter is likely to be inherent in future digital power control designs as more computationally intensive algorithms are implemented to meet more stringent power converter specifications. The resource optimised dual-MAC datapath provides increased computational power

and flexibility to implement typical power control algorithms, as analysed in Chapter 3, at a reasonable hardware cost.

Key architectural features such as the multiple-banked register file memory and the context-switching-based interrupt controller are novel additions to existing power control specialised DSP architectures that provide computational efficiency enhancements. Fast access to relevant data and automatic context switching contribute to the reduction of the DSP cycle time and to the number of instructions required to execute the control algorithms, which therefore reduces algorithm execution times and permits more algorithms to be executed by a single digital controller.

The power converter performance benefits afforded by the dual MAC DSP core in multi-rail DC-DC converter systems are detailed in the next chapter, through the implementation of the DSP core architecture proposed here and its application in a realistic power conversion system.

Chapter 5

Implementation of DSP Core and Application to Multi-Rail SMPC

5.1 Introduction

This chapter describes the implementation of the dual MAC DSP core described in Chapter 4. The steps involved in realising the DSP core specification in hardware by means of FPGA implementation are dealt with in detail. Complementary to verifying the DSP core design using a closed-loop software simulation, FPGA implementation allows a prototype DSP core to be applied to a real DC-DC converter system as in the target application. Thus, if any timing or quantisation issues occur, they can be easily detected by observing the output response of the DC-DC converter.

FPGA implementation tools also facilitate analysis of area and clock requirements, providing insights into the final cost of a DSP-based digital controller chip. An advantage of the FPGA-based implementation approach over ASIC implementation is the ability to perform rapid prototyping, which permits exploration of design trade-offs by implementing multiple design variations [100].

The simulation platform used to verify the DSP core design, together with illustrative simulation results are presented initially. This is followed by a description of the

synthesis process undertaken and a discussion of the results obtained. A comparison of these results with those obtained from the synthesis of a single MAC DSP core design is also presented. The experimental platform used to verify the dual MAC DSP core, consisting of the FPGA and the SMPC application system, is introduced and the procedure for programming the DSP core with a typical power control algorithm is outlined. Finally, the capability of the DSP core to successfully execute control algorithms to regulate the output voltage of multiple DC-DC converters is illustrated in the experimental results obtained from the purpose-built test platform.

5.2 Simulation-Based Verification

The dual MAC DSP core architecture described in Chapter 4 was implemented using the Verilog hardware description language (HDL). The hierarchical structure of the Verilog modules that specify the design is illustrated in Figure 5.1.

Functional verification of the Verilog-specified DSP design was performed using ModelSim in conjunction with MATLAB and Simulink. The HDL code includes a DPWM design and ADC interface hardware. More detail on these auxiliary digital hardware blocks is provided in Section 5.5.1. The power electronics specialised PLECS toolbox from Plexim was used to create an accurate buck converter model in Simulink [101]. This allowed the register transfer level (RTL) design to be evaluated in a simulated closed-loop environment, as illustrated in Figure 5.2. Conversion and gain blocks were required in Simulink in order to combine the ModelSim and PLECS models, while changes in the load current were simulated using an appropriate input waveform.

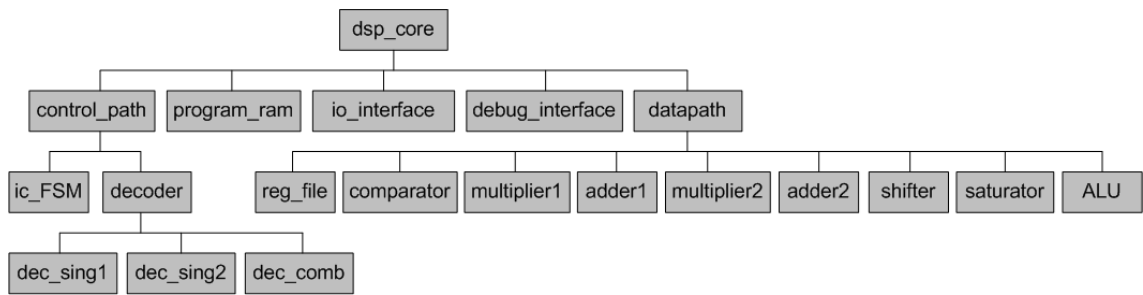


Figure 5.1. Hierarchical structure of Verilog modules of DSP core.

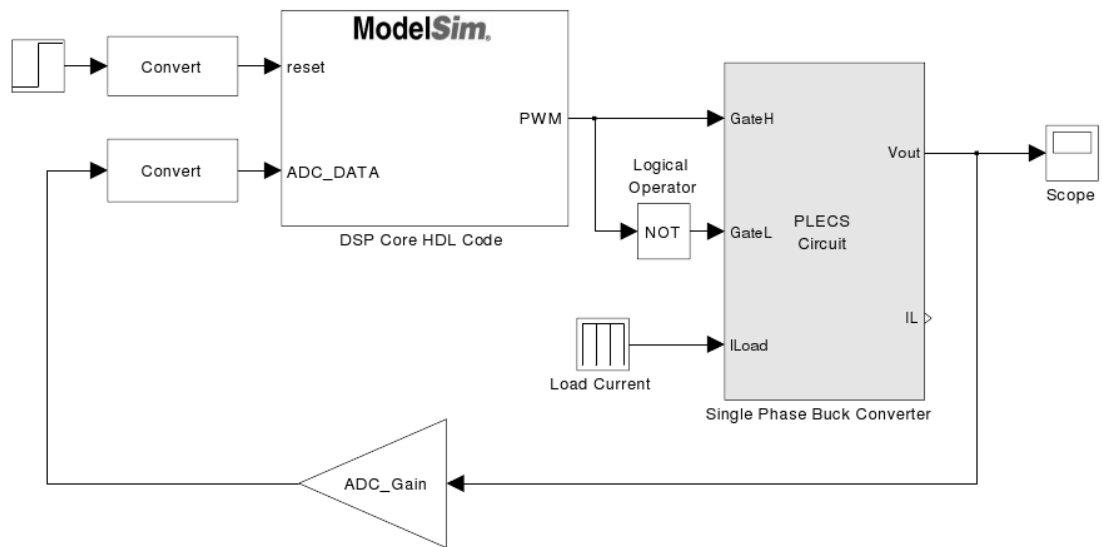


Figure 5.2. Simulink/ModelSim co-simulation configuration.

Figure 5.3 illustrates the resulting output voltage of a single-phase buck converter model which is regulated by the HDL-based controller. The DSP core was programmed to execute a standard three-pole, three-zero (3P3Z) control algorithm, which is typical of the type of control algorithm implemented by commercial digital controllers, as was highlighted in Section 2.4.1. The output voltage is regulated successfully in the presence of load current steps at 2.25 ms and 2.75 ms, as illustrated by the acceptable levels of overshoot and undershoot in the output voltage waveform in Figure 5.3. The voltage deviations are close to the requirements of the most stringent modern voltage regulator specifications [46].

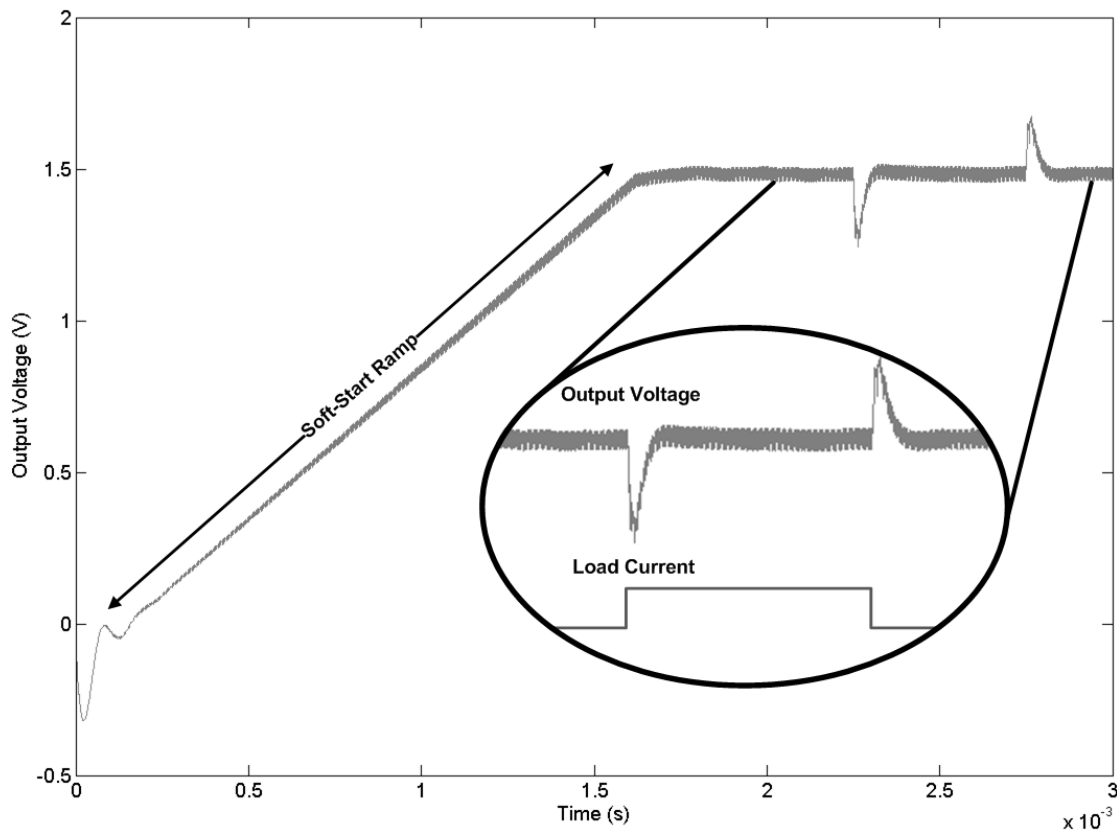


Figure 5.3. Simulink/ModelSim closed-loop co-simulation of dual MAC DSP core - buck converter output voltage response.

The soft-start ramp which occurs when the power supply is first switched on is also simulated. The output voltage is increased from 0 V to its set-point value of 1.5 V by limiting the output voltage error for a pre-defined interval after the DSP core is reset.

The operation of the DSP was also analysed in more detail by examining its internal data and control signals using the waveform viewer of ModelSim, as the waveform snapshot in Figure 5.4 demonstrates. The updating of the data registers is visible for the execution of one iteration of the 3P3Z control algorithm.

5.3 FPGA Synthesis

5.3.1 Synthesis Platform

The Verilog code was synthesised using the integrated synthesiser in the Quartus II design software from Altera, targeting implementation on a Cyclone II FPGA device.

The two 16 x 16 bit multipliers in the datapath of the DSP core were implemented using the embedded multipliers of the FPGA, while the program memory was implemented using the embedded M4K memory blocks, each of which have 4,608 RAM bits.

In order to maximize execution speed, the speed optimisation synthesis option was selected. The synthesis process yielded a maximum achievable DSP clock frequency of 64 MHz due to the critical path from the register file memory, through the datapath to the accumulator register. The multiplier and adder were found to be the main sources of latency in the datapath as can be seen in Table 5.1. The delay contributions of the datapath elements are also shown in the illustration of the critical path in Figure 5.5.

Table 5.1. Delay contributions of datapath elements to critical path.

Datapath Element	Delay (ns)
Adder (A2)	4.249
Multiplier (M2)	3.960
Register file input	2.491
Saturator	2.401
Adder input multiplexer (A2_muxA)	1.344
Saturator input multiplexer (SAT_mux)	0.974
Register file output	0.250

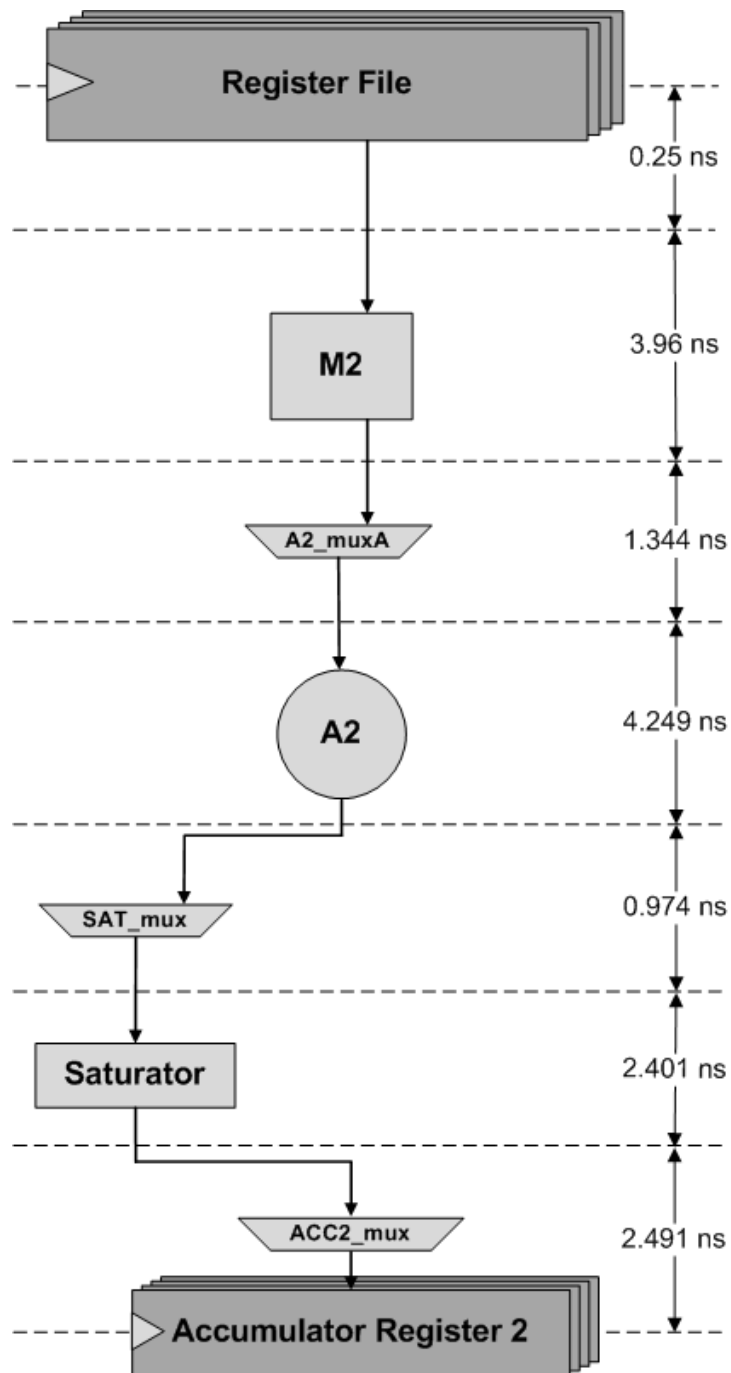


Figure 5.5. Critical path of dual MAC DSP core.

5.3.2 Hardware Cost

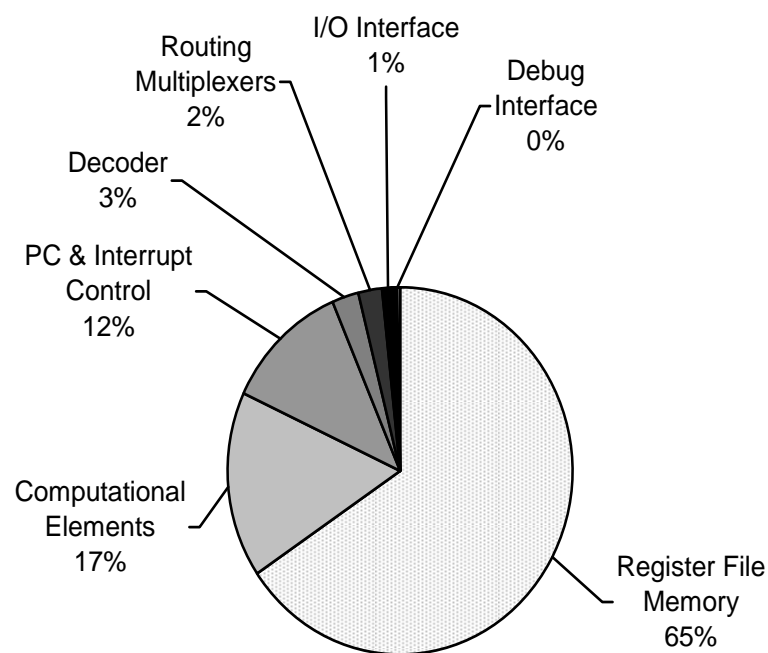
The resource utilisation report obtained from the synthesis process may be examined in order to determine the relative size of the individual components of the DSP core.

The DSP core uses twenty-five percent of the total logic elements of the Cyclone II FPGA. Table 5.2 compares the main constituent elements of the DSP in terms of the quantity of FPGA resources they require and their percentage contribution toward the total quantity of resources required by the DSP core. In addition to the logic elements (LEs) and registers listed in the table, the DSP also uses the embedded memory blocks of the FPGA for the program memory. It should be noted that the contribution of these embedded memory components is not included in the table. The multipliers of the DSP core are in this case implemented using the logic elements of the FPGA rather than the embedded multipliers by selecting the relevant synthesis option. This permits a clearer indication to be obtained of the contribution of the computational elements to the overall resource utilisation. As expected the majority of the registers are required by the register file, while the remainder are needed as configuration and pipelining registers in the program controller. The percentage contribution of each of the individual components towards the total quantity of the logic elements used by the DSP core is illustrated in the pie chart of Figure 5.6.

Figure 5.6 indicates that a major proportion of the logic utilised is allocated to the register file memory block. The register file memory consists of four segments, each with thirty-two 16-bit registers. Read access to four registers and write access to two registers is required in each clock cycle. The input de-multiplexers and output multiplexers therefore contribute to a significant proportion of the overall logic requirements. The computational elements together with the program control hardware consume the next largest quantities of resources, while the remainder is consumed by the decoder, the routing multiplexers of the datapath and other miscellaneous components. It can be seen in Table 5.2 that the majority of registers required are used to form the register file data memory.

Table 5.2. FPGA resource utilisation for synthesised dual MAC DSP core.

Hardware Components	Logic Elements	% of Total Logic Elements of DSP	Registers	% of Total Registers of DSP
Datapath				
Register File Memory	5479	65 %	2384	84 %
Computational Elements	1383	17 %	0	0 %
Routing Multiplexers	171	2 %	0	0 %
Program Controller				
Decoder	217	3 %	0	0 %
PC & Interrupt Control	977	12 %	406	14 %
Miscellaneous				
I/O Interface	125	1 %	65	2 %
Debug Interface	18	0 %	7	0 %
Total	8370	100 %	2862	100 %

**Figure 5.6.** Contribution of main elements to total dual MAC DSP core logic element usage.

5.4 Comparison with Alternative Architectures

In order to evaluate the FPGA resource utilisation of the dual MAC core, a fair comparison can be performed with equivalent single MAC and dual core architectures. For this purpose, a single MAC based architecture was derived from the existing dual MAC design. A dual core architecture consisting of two of these single MAC cores was also used in the comparison under the assumption that its resource utilisation was exactly twice that of the single MAC architecture.

5.4.1 Single MAC Architecture

The single MAC architecture only allows one operation to be executed per DSP clock cycle, for example one MAC operation. The register file memory therefore requires only two output ports and one input port. Apart from the number of ports, the register file has the same four-bank architecture as that of the dual MAC core.

The datapath components are the same as those of the dual MAC architecture except that it has only one multiplier and one accumulator register set. This is illustrated in Figure 5.7, which depicts the datapath elements of the single MAC DSP core.

The instruction decoder hardware requirements are much reduced compared with the dual MAC architecture because only one operation is decoded per instruction cycle for the single MAC architecture.

The single MAC architecture was synthesised using an identical process to that used for the dual MAC architecture, as described in Section 5.3.1. In obtaining the synthesis results, the multiplier in the datapath was implemented using the logic elements of the FPGA, as in the case of the synthesis of the dual MAC architecture. Table 5.3 lists the resulting FPGA resource requirements obtained from the synthesis process, while Figure 5.8 illustrates the percentage contribution of each DSP core component to the overall logic element usage of the DSP core. The single MAC DSP core was also found to use fifteen percent of the total logic elements of the Cyclone II

FPGA. Similar to the dual MAC architecture, the largest percentage of the LEs used are dedicated to implementing the register file memory. The computational elements required the next largest quantities of LEs, while the program control hardware also consumes a significant percentage of the total resources used. The register file memory again dominates the register requirements.

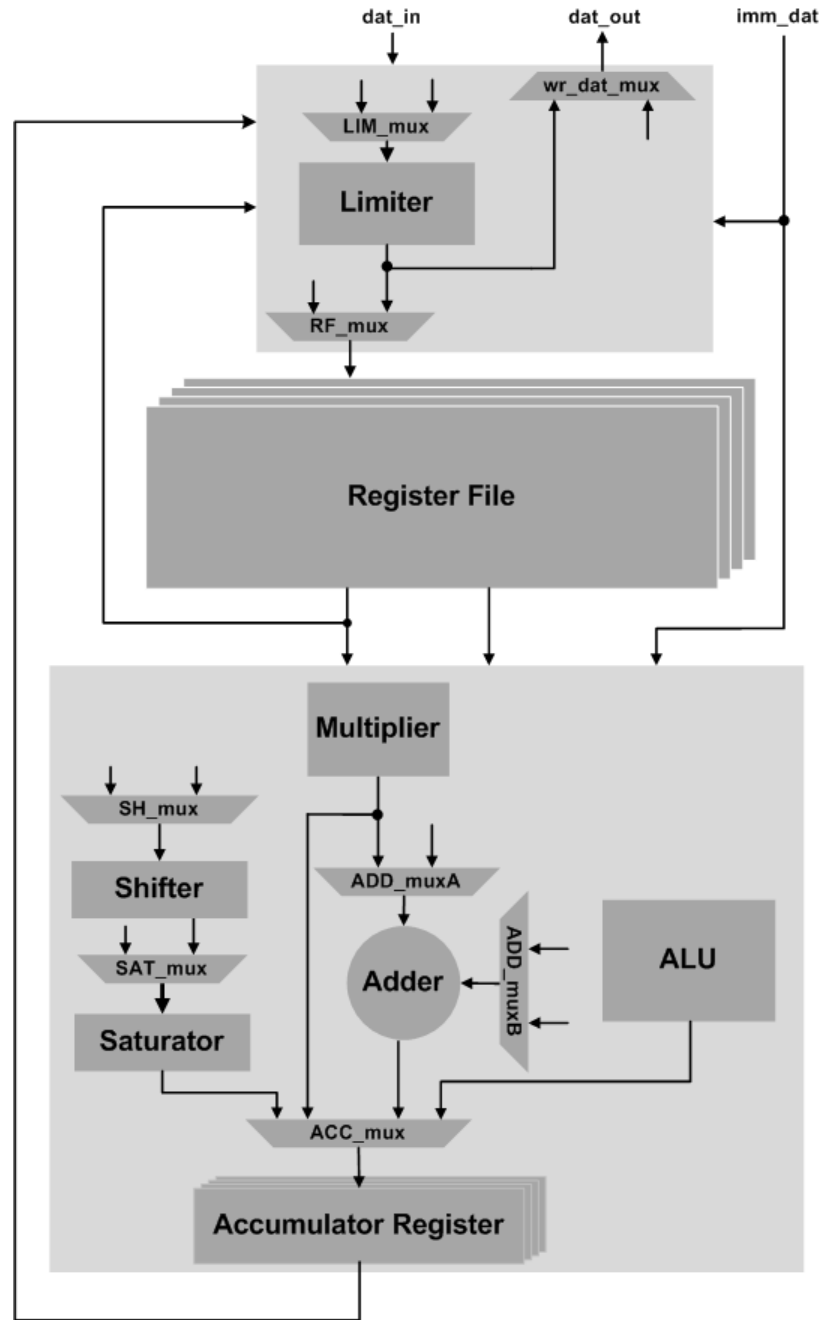
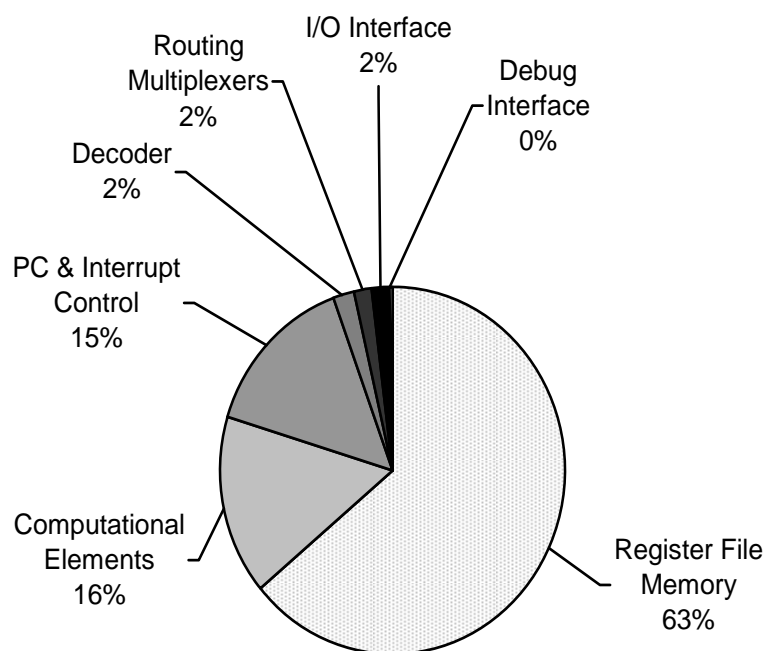


Figure 5.7. Main datapath elements of single MAC DSP core.

Table 5.3. FPGA resource utilisation for synthesised single MAC DSP core.

Hardware Components	Logic Elements	% of Total Logic Elements of DSP	Registers	% of Total Registers of DSP
<i>Datapath</i>				
Register File Memory	3154	63 %	2216	83 %
Computational Elements	775	16 %	0	0 %
Routing Multiplexers	98	2 %	0	0 %
<i>Program Controller</i>				
Decoder	82	2 %	0	0 %
PC & Interrupt Control	726	15 %	405	15 %
<i>Miscellaneous</i>				
I/O Interface	82	2 %	65	2 %
Debug Interface	15	0 %	7	0 %
Total	4932	100 %	2693	100 %

**Figure 5.8.** Contribution of main elements to total single MAC DSP core logic element usage.

5.4.2 Hardware Cost Comparison

The resource utilisations of the components of the single MAC architecture, the dual MAC architecture and the dual core architecture are summarised in Table 5.4, where the dual core architecture has twice the resource usage of the single MAC architecture.

The resource utilisations of the single and dual MAC DSP cores are directly compared in Table 5.5 in terms of the additional resources required by the dual MAC architecture. The largest percentage increase in logic elements is required by the decoder, which is attributable to the additional hardware required to decode two sub-instructions in parallel. The increases in LEs required for the computational elements, datapath multiplexers, and register file are also significant, while minor increases in resources are required by the other components. The dual MAC DSP only requires a minor increase in the number of registers due to the additional accumulators and pipelining registers.

The resource utilisations of the dual MAC architecture and the dual core architecture are also directly compared in Table 5.6 in terms of the additional resources required by the dual core architecture.

Table 5.4. Resource utilisation comparison of single MAC, dual MAC and dual core architectures.

Hardware Components	Logic Elements			Registers		
	Single MAC	Dual MAC	Dual Core	Single MAC	Dual MAC	Dual Core
<i>Datapath</i>						
Register File Memory	3154	5479	6308	2216	2384	4432
Computational Elements	775	1383	1550	0	0	0
Routing Multiplexers	98	171	196	0	0	0
<i>Program Controller</i>						
Decoder	82	217	164	0	0	0
PC & Interrupt Control	726	977	1452	405	406	810
<i>Miscellaneous</i>						
I/O Interface	82	125	164	65	65	130
Debug Interface	15	18	30	7	7	14
Total	4932	8370	9864	2693	2862	5386

Analysing Table 5.6 it is clear that the dual core architecture requires additional resources compared with the dual MAC architecture for the majority of the DSP core's components. The register file memory and program controller components in particular contribute to the required additional resources of the dual core architecture. Thus, it can be concluded that the optimised dual MAC architecture can achieve its goal of increased computational performance with significantly fewer resources than the dual core architecture.

Table 5.5. Additional resources required for dual MAC DSP compared with single MAC DSP.

Hardware Components	Additional Logic Elements	% Increase in Logic Elements	Additional Registers	% Increase in Registers
<i>Datapath</i>				
Register File	2325	74 %	168	8 %
Computational Elements	608	78 %	0	0 %
Routing Multiplexers	73	74 %	0	0 %
<i>Program Controller</i>				
Decoder	135	165 %	0	0 %
PC & Interrupt Control	251	35 %	1	0 %
<i>Miscellaneous</i>				
I/O Interface	43	52 %	0	0 %
Debug Interface	3	20 %	0	0 %
Total	3438	70 %	169	6 %

Table 5.6. Additional resources required for dual core DSP compared with dual MAC DSP.

Hardware Components	Additional Logic Elements	% Increase in Logic Elements	Additional Registers	% Increase in Registers
<i>Datapath</i>				
Register File	829	15 %	2048	86 %
Computational Elements	167	12 %	0	0 %
Routing Multiplexers	25	15 %	0	0 %
<i>Program Controller</i>				
Decoder	0	0 %	0	0 %
PC & Interrupt Control	475	49 %	404	100 %
<i>Miscellaneous</i>				
I/O Interface	39	31 %	65	100 %
Debug Interface	12	67 %	7	100 %
Total	1494	18 %	2524	88 %

5.5 Experimental Platform

The experimental platform used to evaluate the DSP core in a multi-rail SMPC application consists of two interconnected parts as illustrated in Figure 5.9. The first part is a commercial FPGA evaluation board, the Altera DE2 board, which includes the Cyclone II FPGA, on which the DSP core and all other digital hardware is implemented. The second part is a custom printed circuit board (PCB) featuring the buck converter, load, ADCs and sensing circuitry.

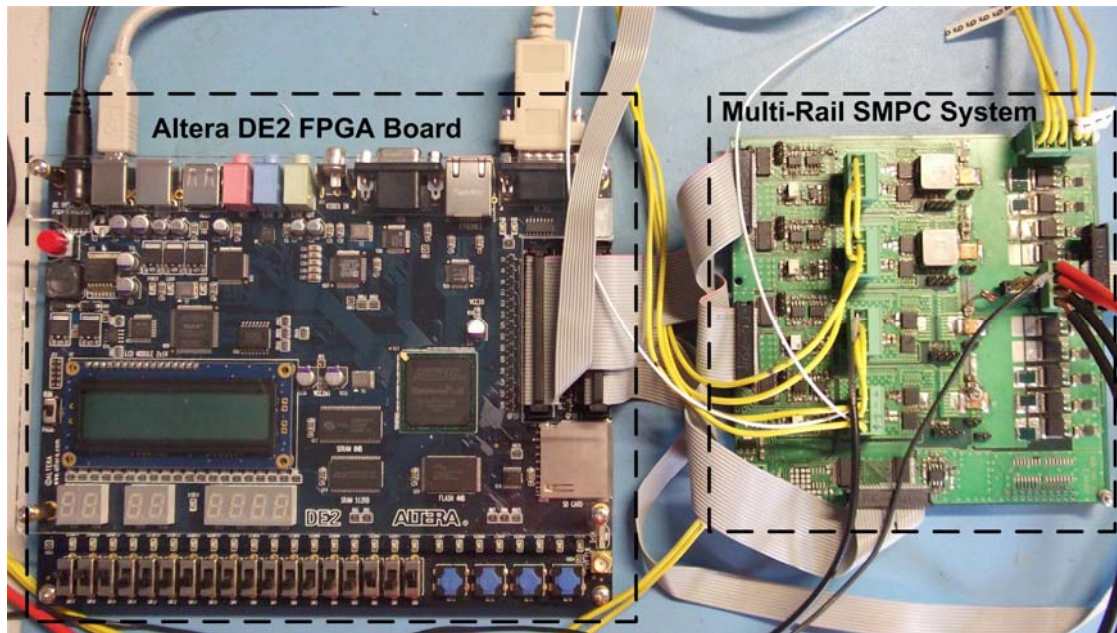


Figure 5.9. Experimental platform for prototyping of DSP core.

5.5.1 Digital Hardware

Figure 5.10 illustrates the main elements of the digital system that was synthesised and downloaded on to the Cyclone II FPGA. The synthesised DSP core design was combined with the necessary digital interface hardware to allow data acquisition from the voltage-sampling ADCs. An existing DPWM design was also interfaced to the processor. Fast on-line programming of the DSP core was achieved using a UART

connection from the FPGA board to a personal computer. A custom debug interface was incorporated into the digital hardware to aid troubleshooting in conjunction with the UART connection. Additional logic was also required to implement clock synchronisation and soft-start functionality. The system clock frequency of 33 MHz was derived from the 50 MHz oscillator on the FPGA board using one of the embedded phase-locked loops (PLLs) on the FPGA.

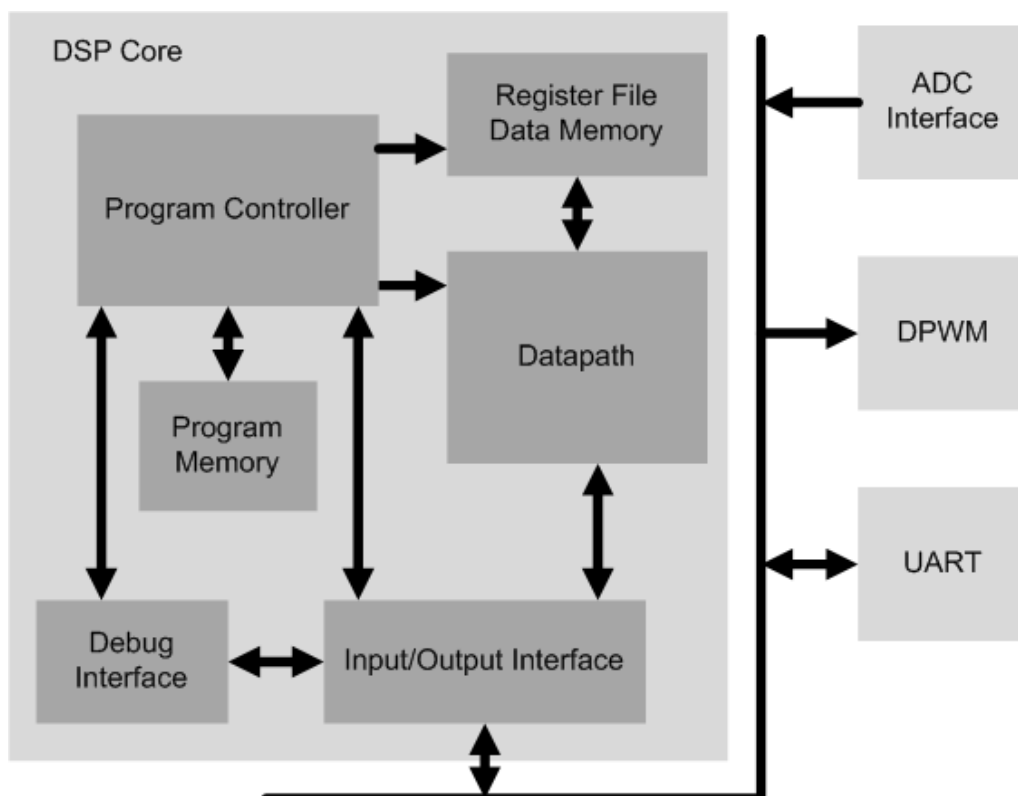


Figure 5.10. Digital hardware units implemented on FPGA.

5.5.2 SMPS System

The prototype power supply system featured on the custom PCB consists of three identical single-phase 12 V to 1.5 V buck converters each with a 500 kHz switching frequency, as illustrated in Figure 5.11. Other parameters associated with the buck converters are listed in Table 5.7.

A variable current sink at the output of each of the buck converters acts as the dynamic load. The current sink consists of a network of resistors and MOSFET switches. This permits the application of load current steps up to 3 A to the DC-DC converter to evaluate its transient performance. The load steps are enabled using the toggle switches on the FPGA evaluation board. Larger static loads are applied to the output of the DC-DC converter by means of a connection to a DC electronic load [102], which is a specialised device used for testing power supplies.

Individual ADCs are used to independently sample the output voltages of the three SMPCs once per switching period. Although a multi-channel ADC would be more appropriate in a multi-rail power converter application, using individual ADCs permits more flexibility in sampling multiple signals when evaluating experimental prototype designs.

Table 5.7. Parameters of buck converters in experimental platform.

Buck Converter Parameter	Value
Input voltage	12 V
Output voltage	1.5 V
Switching frequency	500 kHz
Inductor	680 nH
Output capacitor	450 μ F
Maximum output current	15 A

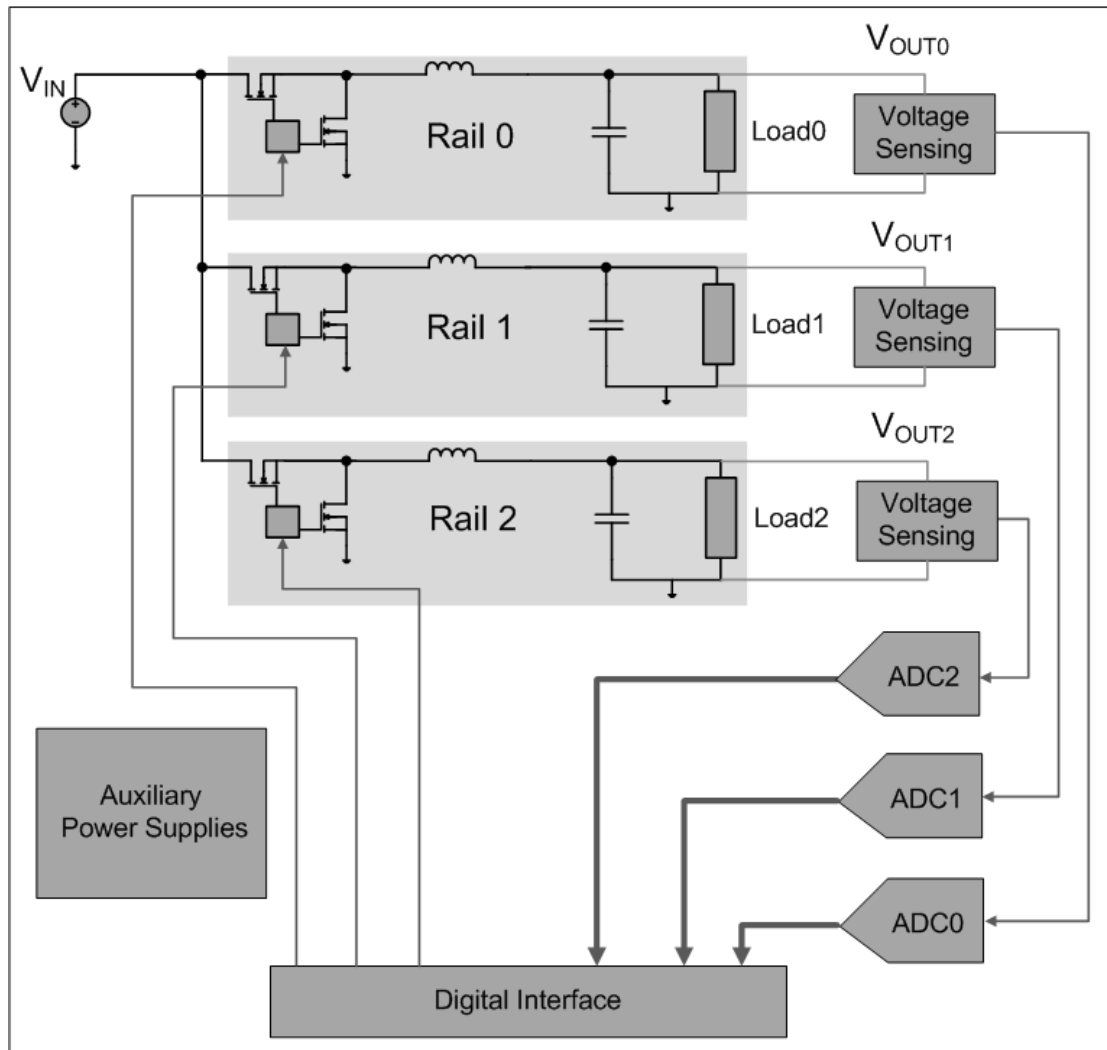


Figure 5.11. Custom PCB featuring multi-rail SMPC system.

5.5.3 DSP Core Programming

In order to convert the assembly language based instruction set mnemonics of the DSP core’s instruction set to binary code, an assembler was developed using the Ruby programming language [103]. The input to the assembler is a text file, which consists of the instructions that define the control algorithms to be executed, as well as a number of initialisation instructions for loading data and configuration registers. The output of the assembler is a file that contains the binary representation of the instructions listed in the input assembly language based program.

The machine code output of the assembler is loaded into the program memory by means of the UART interface between a PC and the DSP core, as illustrated in Figure 5.12. The machine code could alternatively be used to program a ROM device, whose contents could be loaded into the program memory on start-up. However, using the UART is more convenient in this case because it facilitates frequent re-programming while validating the correct operation of the DSP core.

A 3P3Z voltage mode control algorithm was implemented using the DSP core's assembly language instructions for application to each of the buck converters. The operations involved in executing the 3P3Z control algorithm are reflective of the operations required in a wide range of control algorithms that are commonly applied to multi-rail DC-DC converter systems. The implementation details of these algorithms were discussed comprehensively in Section 3.3. Example program code for the control algorithm is illustrated in the ISR code excerpt of Figure 5.13, while the entire assembly program used in verifying the operation of the DSP core is provided in Appendix B. The lines beginning with a semicolon are comments and are not executed by the DSP core. Each line of code corresponds to a single instruction, thus two operations are executed in parallel where two operations are given on the same line of code. The result of the second operation on each line is stored in the second accumulator of the DSP core's datapath. If only one operation is specified in an instruction the result is stored in the first accumulator. In the case when only one operation is to be executed where it is required that the result be stored in the second accumulator, the computational operation is preceded on the same line of code by a *NOP* instruction.

The control algorithm described in Figure 5.13 uses pre-calculations to minimise the delay between ADC sampling and duty cycle updating, as described in Section 3.3.1. The pre-calculated sum is stored in the first accumulator between separate iterations of the algorithm. When it is required to store the value in one of the accumulator registers to a 16-bit register, the most significant bits (MSBs) of the accumulator are selected by specifying *ACC1H* or *ACC2H* in the instruction, while the least significant bits (LSBs) are selected using *ACC1L* or *ACC2L*.

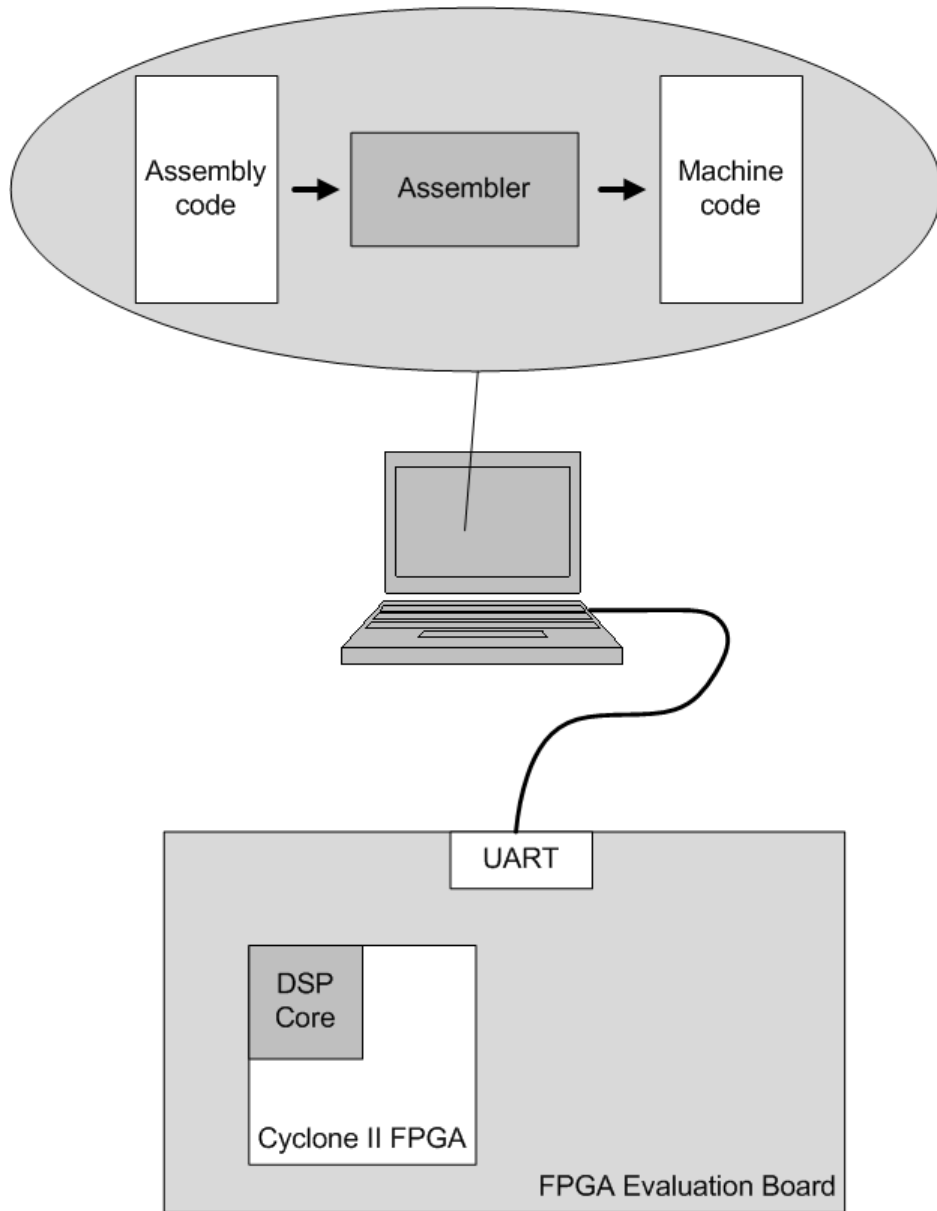


Figure 5.12. Program code assembly and DSP core programming.

```

; Load R8 with error value from ADC
LOADP  R8

; b0 * e(n) + ACC1
MAC  R8  R0

; Convert to Q12 by left-
; shifting by 4
SHIFTSAT  ACC1H  -4

; Round ACC1
RNDSAT  ACC1H

; Output duty cycle to PWM, Load d(n-1) reg with d(n)
STOREP  ACC1H  MOV  ACC1H  R11

; NOP, a3 * d(n-3)
NOP  MUL  R13  R6

; b3 * e(n-3), a2 * d(n-2) + ACC2
MUL  R10  R3  MACU  R12  R5

; b2 * e(n-2), a1 * d(n-1) + ACC2
MACU  R9  R2  MACU  R11  R4

; b1 * e(n-1) + ACC1, Convert to Q12 by right-
; shifting by 2
MACU  R8  R1  SHIFTSAT  ACC2H  2

; Sum b's and a's
ADD  ACC1H  ACC2H

; Return from interrupt
RETI

```

Figure 5.13. Assembly code for 3P3Z algorithm on dual MAC DSP.

5.6 Multi-Rail Voltage Mode Control

The application of the dual MAC DSP core to the test system described in the previous section allowed an evaluation of the DSP's operation and performance to be undertaken. The application of the single MAC DSP core to the same test system facilitated a direct comparison with the dual MAC DSP in terms of control algorithm

execution time and impact on power converter performance. The results of this comparison are particularly important because the single MAC DSP core is representative of typical commercial DSP-based controller architectures, such as those reviewed in Section 2.5.2. Although the dual core architecture was not applied to the test system, aspects of its potential performance could be deduced by analysing the performance of the single MAC DSP.

5.6.1 Multiple Control Algorithm Execution

Figure 5.14 illustrates the time-multiplexing of the single MAC DSP core to execute three identical 3P3Z control algorithms, where each control algorithm is executed in 600 ns. The bus signal at the bottom of Figure 5.14 indicates the interrupt service routine or control algorithm that is currently being executed. In between the execution of the algorithms, background code is executed, which is indicated by '3' in the bus signal.

The dual MAC core was also used to execute three 3P3Z control algorithms, which were identical to those used by the single MAC core. Figure 5.15 illustrates the time-multiplexing of the three control algorithms on the dual MAC core. Although the single MAC DSP can successfully execute the same control algorithms for the multi-rail power converter system, it can be seen by comparing Figure 5.14 with Figure 5.15 that the single MAC DSP does not execute each algorithm as quickly as the dual MAC DSP. The dual MAC architecture only requires 360 ns to execute the same 3P3Z algorithm, which is 60% of the execution time required by the single MAC DSP. This results in much less background code being executed in the same time interval by the single MAC core, which is clearly visible in Figure 5.14. Application of the same control algorithm code to the dual core DSP would result in one of the cores having the same utilisation as the single MAC architecture, while the other core would be available to execute additional background code.

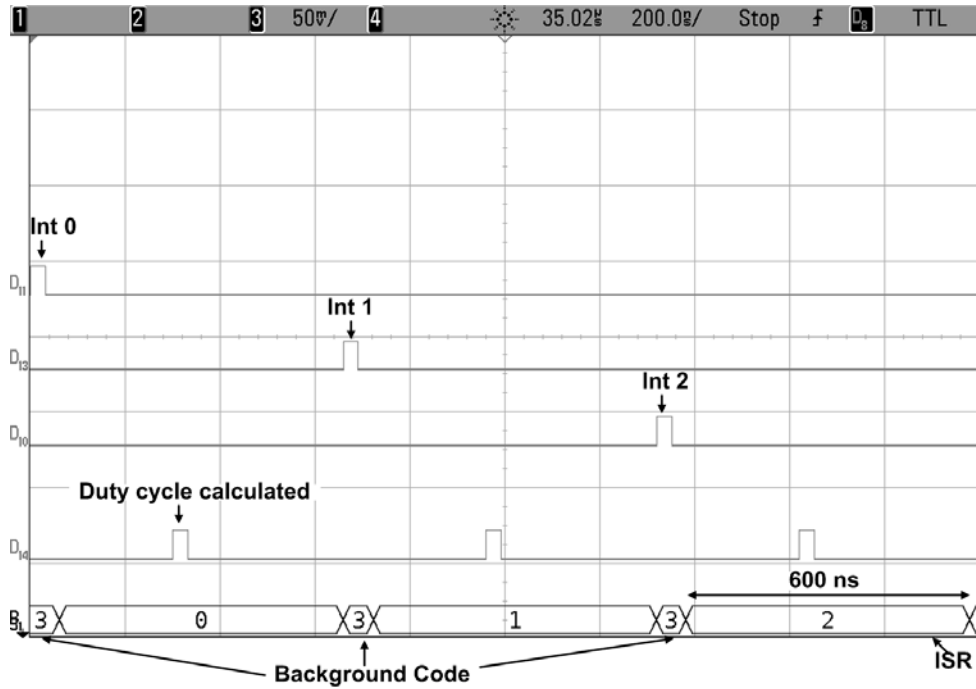


Figure 5.14. Single MAC DSP core executing three control algorithms and background code for three independent buck converters.

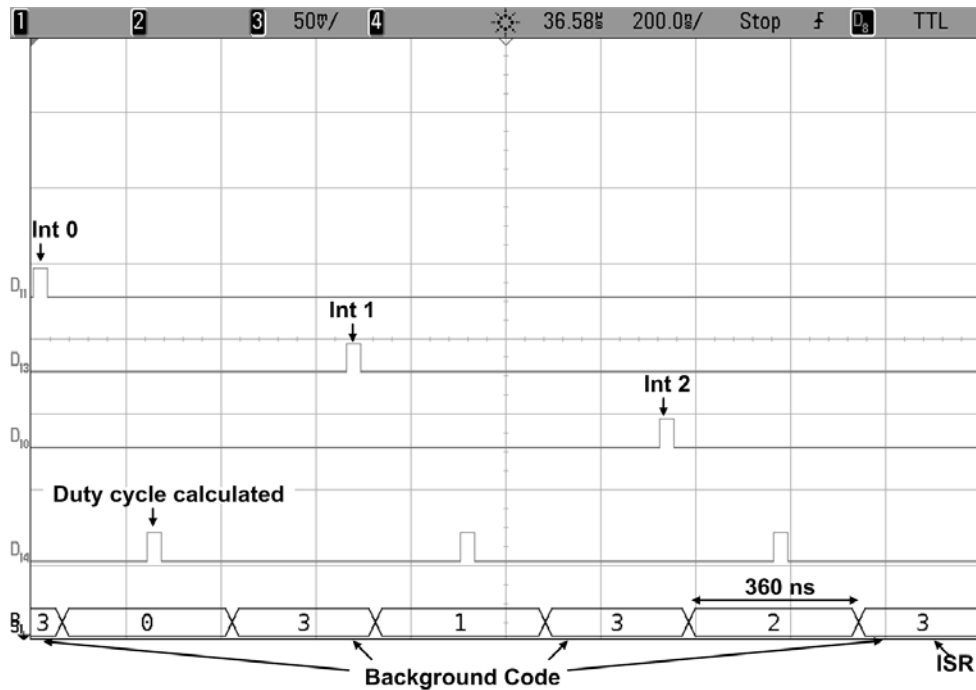


Figure 5.15. Dual MAC DSP core executing three control algorithms and background code for three independent buck converters.

5.6.2 Impact on DC-DC Converter Performance

Figure 5.16 illustrates the execution of one control algorithm on the single MAC core. When the *Int0* pulse is activated, the ADC samples the output voltage and the execution of the control algorithm also commences. The duty cycle is calculated at around the half-way point of execution of the control algorithm because pre-calculations are used. If pre-calculations had not been used the duty cycle would not be calculated until the end of the algorithm. The time between the activation of the interrupt signal and the calculation of the duty cycle is 330 ns. The DPWM is updated with the newly calculated duty cycle value immediately before the beginning of the next switching cycle.

The calculation of the duty cycle on the dual MAC core is illustrated in Figure 5.17, where there is an interval of 270 ns between when the interrupt signal is activated and when the duty cycle is calculated. Thus, the shorter computation time of the dual MAC core means that it has a shorter delay between when the output voltage is sampled and when the duty cycle is calculated compared with the single MAC core.

For the dual core DSP, the computation time of the duty cycle would be identical to that of the single MAC architecture in spite of the additional computational components, thus resulting in the same processing delay.

By sampling the output voltage as close as possible to the end of the switching cycle, the controller can react quickly to any changes in the load. The proximity of the sampling instant to the end of the switching cycle is limited by the calculation time of the duty cycle value after the voltage sample is available to the DSP core. Thus, the longer calculation time required by the single MAC processor results in the sampling instant being further away from the end of the switching cycle compared with the dual MAC processor. The consequence of the longer delay between sampling and calculation of the duty cycle means that the single MAC processor requires a longer time to react to changes in the output voltage of the DC-DC converter. Similarly, the dual core DSP would also have a longer reaction time than the dual MAC DSP.

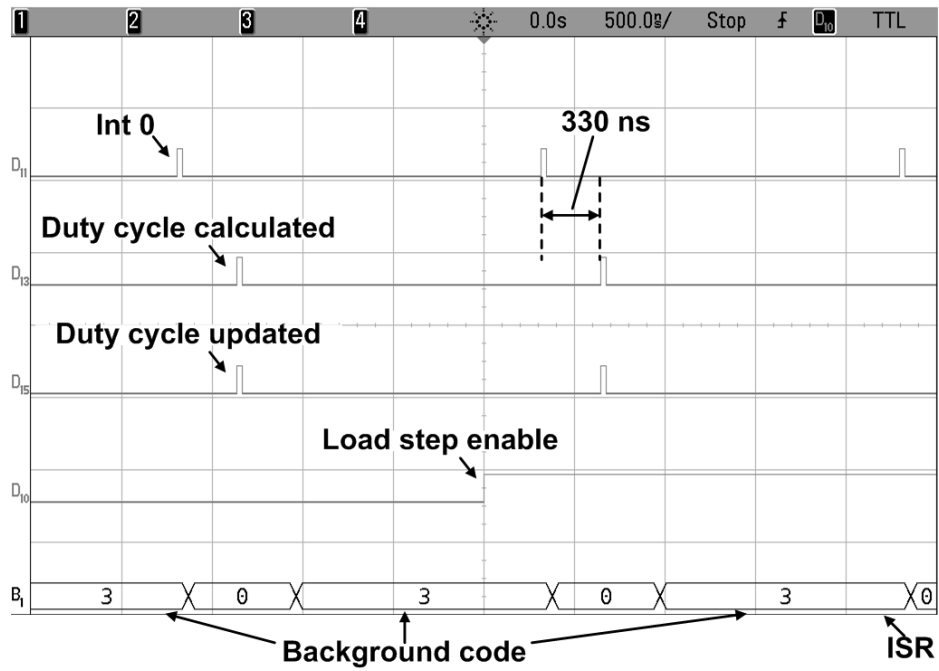


Figure 5.16. Duty cycle calculation instant during execution of control algorithm by single MAC DSP.

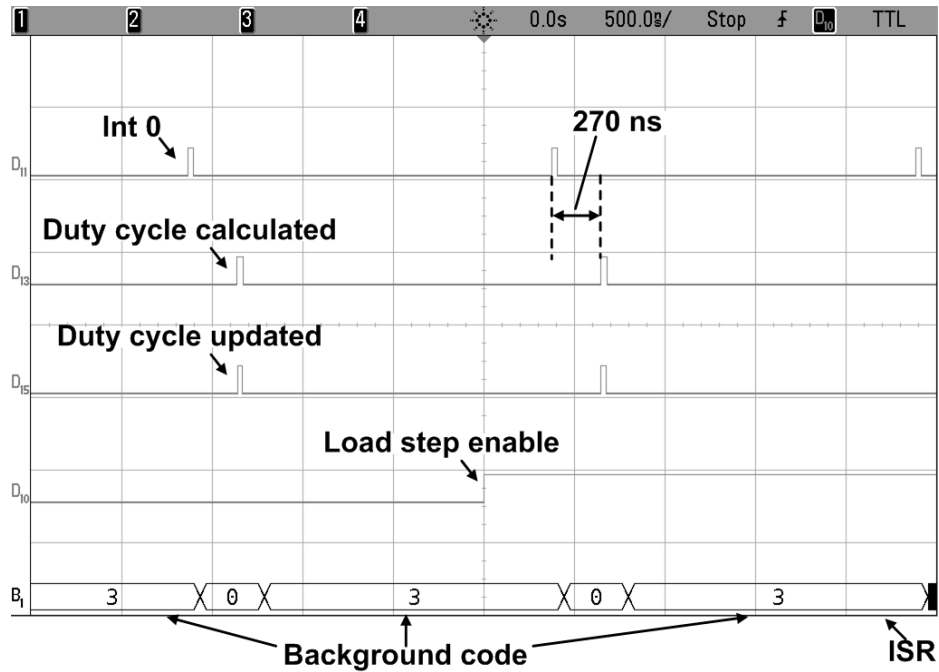


Figure 5.17. Duty cycle calculation instant during execution of control algorithm by dual MAC DSP.

The regulation of the output voltage of one of the DC-DC converters to 1.5 V is illustrated in Figure 5.18, where the single MAC DSP core has been used to execute the 3P3Z control algorithm and a load step of 3 A has been applied. This voltage response is representative of what would be obtained using a standard commercial DSP based controller with only one MAC element. An identical response to that in Figure 5.18 would be obtained for the dual core DSP due its having an identical computation time to the single MAC DSP. Executing the same 3P3Z control algorithm using the dual MAC DSP core for application to the same DC-DC converter, for the same 3 A load step, results in the output voltage response illustrated in Figure 5.19. Comparing Figure 5.18 with Figure 5.19, it can be observed that the transient performance is considerably improved in Figure 5.19 in terms of voltage drop and settling time, which thus results in a more desired transient response due to the use of the dual MAC core. The shorter computational delay of the dual MAC core compared with the single MAC core results in close to a 10 % reduction in the output voltage drop from 316 mV to 285 mV, which in turn leads to a 40 % reduction in settling time from 100 μ s to 60 μ s. The faster dual MAC response also has less overshoot and is therefore a more stable and more favourable response than that obtained using the single MAC core. The transient response obtained using the dual MAC DSP core is thus a very significant result in terms of the power converter performance.

From these results it is clear that applying a dual MAC DSP architecture to execute control algorithms for multi-rail DC-DC converter systems leads to improved power converter performance compared with applying a single MAC DSP. As discussed earlier, by illustrating improved performance compared with a single MAC architecture it can be concluded that the dual MAC DSP is a more suitable digital controller than the standard single MAC based DSPs that are commonly applied in commercial applications.

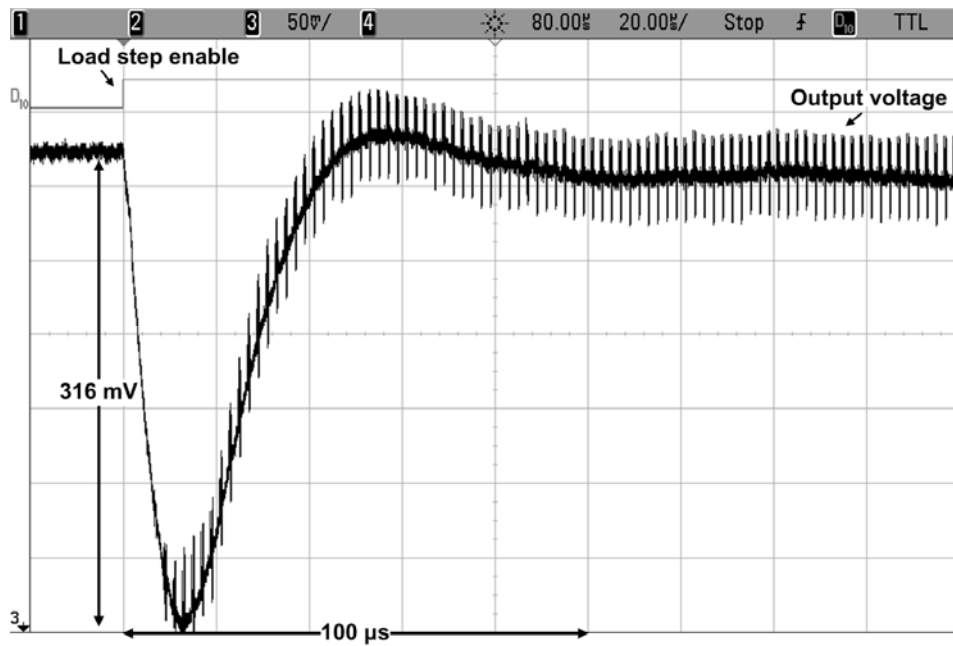


Figure 5.18. Single MAC DSP core regulating output voltage of buck converter to 1.5 V in presence of load current step.

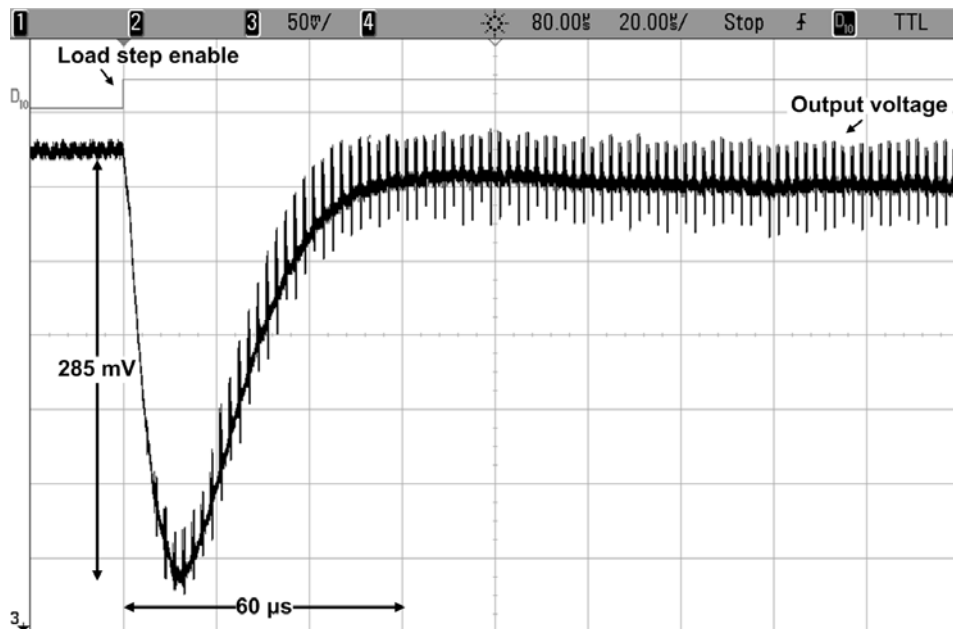


Figure 5.19. Dual MAC DSP core regulating output voltage of buck converter to 1.5 V in presence of load current step.

5.7 Summary

This chapter demonstrated the successful implementation of the dual MAC DSP core as a digital controller in a multi-rail SMPC system. The dual MAC architecture was shown to be better in terms of hardware cost compared with an equivalent dual core architecture. The dual MAC architecture was also shown to have significant performance advantages over equivalent single MAC and dual core architectures through its ability to execute control algorithms in a shorter time interval. This capability enables the dual MAC processor to be applied in power conversion applications where there is a requirement for the duty cycle to be calculated as fast as possible in order to react to rapid changes in the load current. It also allows the DSP to either execute more complex control algorithms that require additional operations compared with standard PID-type compensators or alternatively to implement more advanced power management features in the background code.

Chapter 6

Control of SMPC Systems with Non-Integer Switching Frequency Ratios

6.1 Introduction

The ability of the dual-MAC DSP core to provide digital control for multi-rail DC-DC converter systems was verified in the previous chapter. The switching frequencies of the individual SMPCs examined there were synchronised i.e. there was an integer multiple ratio between the switching frequencies. Although synchronisation is sometimes required to avoid creating beat frequency signals which can degrade the performance of the DC-DC converters or components of the application system [104], it can also have a negative impact on the efficiency or performance of the power converters because the designer is forced into selecting non-optimal switching frequencies. If arbitrary switching frequencies can be chosen, the task of meeting the unique efficiency and performance specifications of each of the individual power converters is easier.

Although individual analogue or digital compensators could be used to control each of the DC-DC converters with no constraint on the ratio between their switching frequencies, the hardware requirements of multiple compensator solutions are excessive for many multi-rail applications. The DSP-based compensator approach

presented in Chapter 5 is preferable, whereby multiple power converters are controlled by a single DSP. However implementing control for non-integer switching frequency ratio systems using a standard DSP can lead to performance or efficiency related issues, as described in Section 2.5.2. This chapter introduces a hardware solution that overcomes the drawbacks of conventional DSPs, thereby enabling the benefits of non-integer switching frequency ratios to be obtained in multi-rail applications [105].

The chapter has the following structure. The principles of interrupt triggered control are investigated first. This is followed by an analysis of the delay caused by the use of interrupts in multi-rail DC-DC converter systems. The impact of this delay in terms of power converter performance is also discussed. A method to overcome the problems associated with standard interrupt control methods is outlined and the improved performance obtained by applying that method is demonstrated experimentally in a multi-rail SMPC system with a non-integer switching frequency ratio.

6.2 Interrupt Triggered Control

6.2.1 Overview

Interrupts are used to trigger the execution of control algorithms in DSP-based digital power controllers [106]. The interrupt signal may be derived from a pulse signal, which is generated in the ADC module when a new sample has been acquired [71]. This allows control algorithm execution to immediately follow ADC sampling. In general, only one sample is processed per power converter switching cycle. Alternatively, the interrupt signal can be generated from a timer or derived from a period-match pulse, which is generated in the PWM module at the beginning of each switching cycle.

In a multi-rail power converter system a separate interrupt signal is assigned to each power converter, which triggers the processor to execute the control algorithm for the

assigned power converter as part of the corresponding interrupt service routine (ISR). The DSP's execution time is thus divided up among each of the individual algorithms, which is achieved by interleaving the interrupt signals so that each control loop has its own fixed time slot. This is illustrated in Figure 6.1 where two power converters are being controlled by a single DSP. In this case the power converter that is being controlled by the algorithm in *ISR1* has a switching period, T_s , while the SMPC being controlled by the algorithm in *ISR2* has a switching period that is twice T_s .

This method is only practical when all algorithms are either executed at the same frequency or at different frequencies that are integer multiples of each other. When the execution frequencies have non-integer ratios it is impossible to simply assign time slots to each of the algorithms so that they do not overlap. It should be noted that although the algorithms illustrated in Figure 6.1 have identical execution times, this is not a requirement. Multiple different control algorithms can be executed in this way, provided there is sufficient time for each of the required algorithms to be executed without overlap.

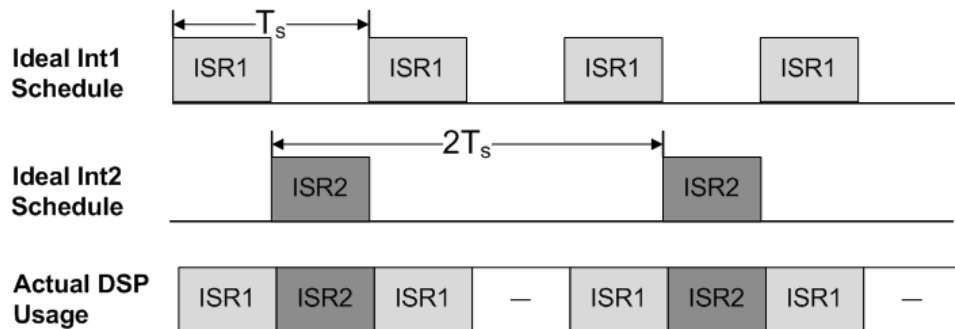


Figure 6.1. Interleaved scheduling for interrupts with integer multiple frequency ratio.

6.2.2 Delays in Non-Integer Switching Frequency Ratio Systems

In a typical DSP, control algorithms should not be interrupted during their execution to ensure that the duty cycle is calculated as fast as possible. This is usually achieved by disabling the interrupt nesting mode or by manually masking or disabling all

interrupts each time an ISR is entered. If an interrupt occurs when a control algorithm is already being executed, as in the case where the interrupt frequencies have a non-integer ratio, the interrupt is not serviced until the execution of the algorithm has completed. This results in a delay in the calculation and updating of the duty cycle of the pending interrupt service routine. Figure 6.2 illustrates an example of this where there is a ratio of 5/3 between the interrupt frequencies.

When multiple interrupt signals occur simultaneously, the algorithms are typically executed according to their pre-defined priority, where each interrupt has a different priority level. Thus an extra delay is again introduced between ADC-sampling and duty-cycle updating for the power converter controlled by the lower priority interrupt. This is illustrated in Figure 6.3, where *Int1* has higher priority than *Int2* and the interrupt frequencies have a non-integer ratio of 3/2.

Consequently, the delay between ADC-sampling and duty-cycle-updating can vary each time an interrupt is triggered, depending on whether or not multiple interrupts have occurred simultaneously or if an algorithm is already being executed. If the duty cycle has not been calculated by the beginning of the switching cycle, the DPWM will apply the duty cycle from the previous cycle, as illustrated in Figure 6.4, where the execution of *ISR3* has been delayed by the higher priority *Int1* and *Int2* interrupts.

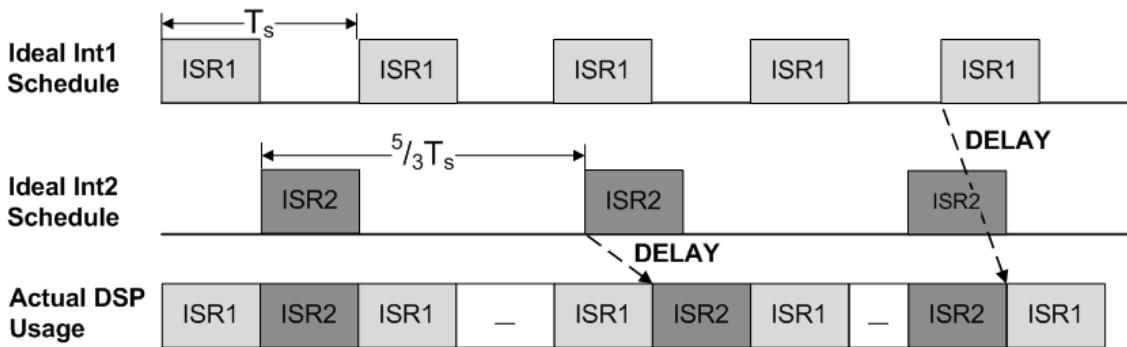


Figure 6.2. Delayed scheduling for overlapping interrupts with non-integer multiple frequency ratio.

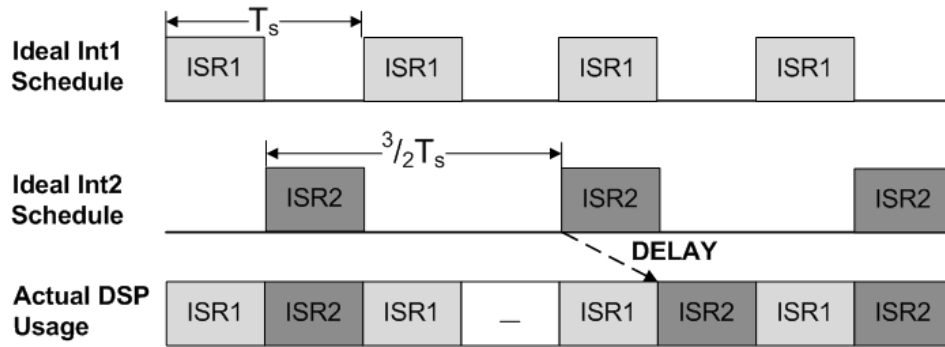


Figure 6.3. Delayed scheduling for simultaneous interrupts with non-integer multiple frequency ratio.

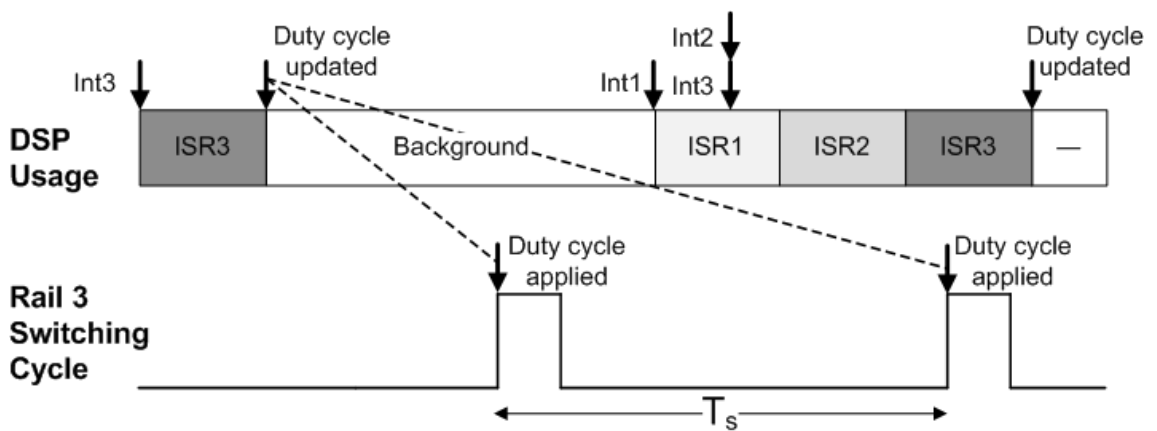


Figure 6.4. Application of duty cycle from previous cycle caused by delay in duty cycle update.

This behaviour is undesirable because a fixed loop delay is assumed when designing the compensator for the closed-loop system [107, 108]. Additionally if a load transient occurs around this time, the delay in updating the duty cycle will result in a much slower response in the output voltage. The power converter could also become unstable if the delay occurs for a number of consecutive cycles [48].

To overcome the aforementioned problems, the delay can be fixed at its maximum possible value for each iteration of each algorithm. This is achieved by setting the interrupt instant at a sufficient offset from the beginning of the next switching period, such that when the maximum number of interrupts occurs simultaneously, the duty cycle will be calculated just in time for the beginning of the next switching cycle, as illustrated in Figure 6.5. Conversely, when only one interrupt occurs there will be an

idle interval between when the duty cycle is calculated and the beginning of the next switching cycle if no other interrupt occurs during that time, as Figure 6.6 illustrates. The execution point of the lower priority algorithms thus jitters within a permitted time interval.

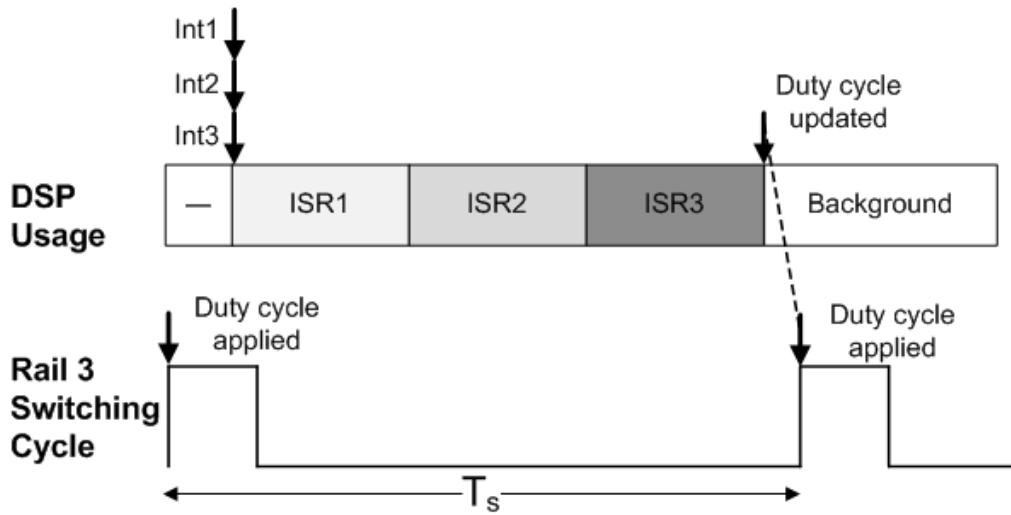


Figure 6.5. Duty cycle updated just in time using maximum fixed delay.

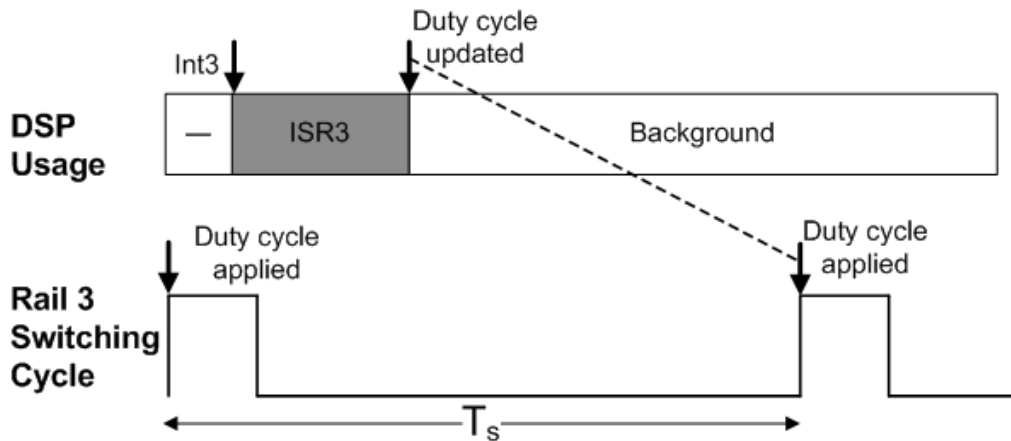


Figure 6.6. Duty cycle updated well in advance of application time using maximum fixed delay.

A problem with using the maximum fixed delay is that it is excessive and prohibits the use of wide bandwidth compensators. The performance of the voltage regulator is therefore degraded due to a much slower response to load transients [31]. Improved performance can be obtained through a reduction of this delay [109], i.e. reducing the time between when the ADC is sampled and when the calculated duty cycle is applied.

6.2.3 Program Code Priority

In an attempt to reduce the delay between ADC sampling and duty cycle updating, it is insightful to examine the priority of all of the tasks being executed by the DSP. The instructions of a control algorithm are usually ordered so that the highest priority operations are executed first. When a new ADC sample becomes available, the duty cycle is calculated immediately using the pre-calculated sum from the previous iteration of the algorithm. Subsequently, the DPWM is updated and the pre-calculations are completed in advance of the next iteration of the algorithm.

Figure 6.7 illustrates the priority levels of the various sections of program code executed by a DSP-based power controller, where the interrupt for *ISR1* has the highest priority, followed by the interrupt for *ISR2* and so on, with the interrupt for *ISR_N* having the lowest priority. The control algorithm for the power converter with the strictest transient performance requirements should be executed in *ISR1* to avail of the shortest possible ADC to duty cycle calculation delay provided by the DSP. It can be seen in Figure 6.7 that the highest priority levels are occupied by the time-critical duty cycle calculation sections of code for each of the control algorithm ISRs, which should be executed as soon as possible after the corresponding interrupt signal is triggered. These are followed by the pre-calculation sections, which must be executed before the next iteration of each of the algorithms begins, in order to use the pre-calculated value in computing the new duty cycle in the next iteration. The pre-calculation sections are followed by the non-critical background code, which can be executed in a flexible manner as it does not have any imminent deadlines.

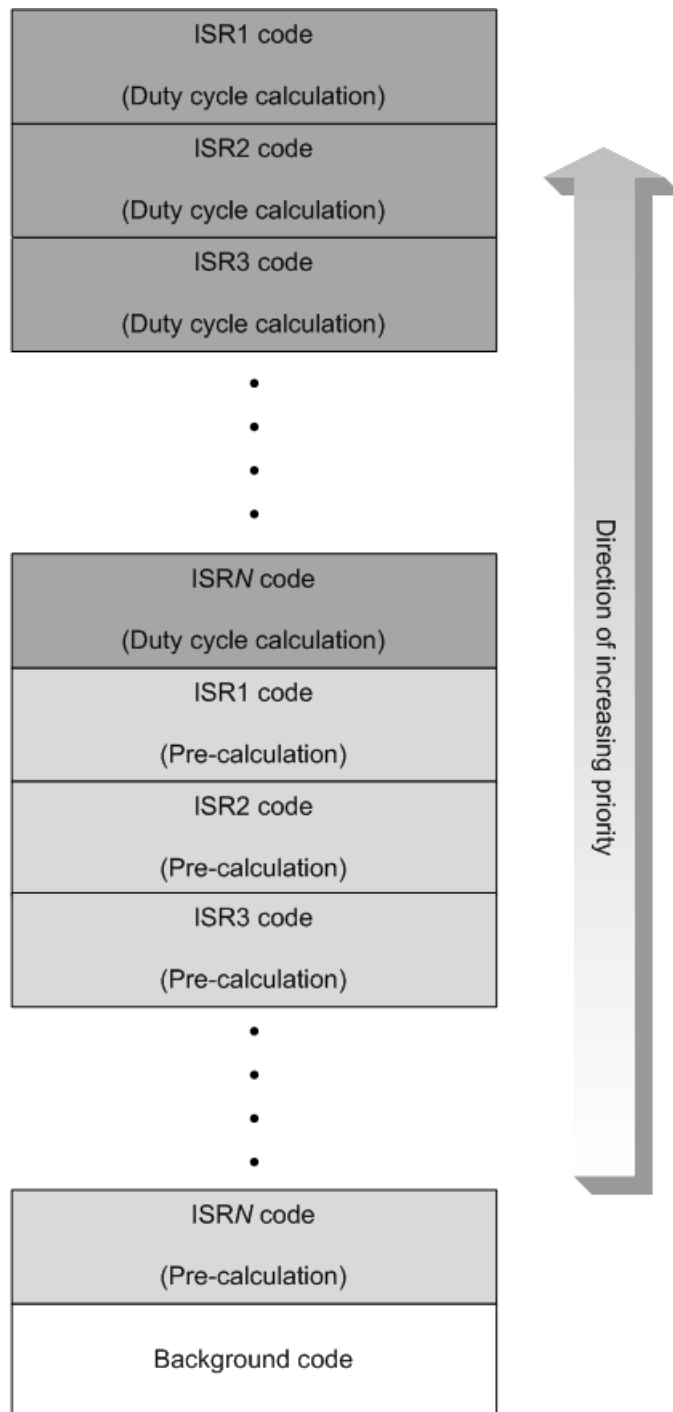


Figure 6.7. Priority levels of code executed by DSP-based digital power controller.

6.2.4 Maximum ADC-Sample to Duty-Cycle-Update Delay

Figure 6.8 shows the maximum ADC-sample to duty-cycle-update delays (T_{DMAX}) in the situation where three interrupt signals coincide, where T_{ADC} is the ADC conversion delay, T_{DC} is the duty cycle calculation time and T_{PC} is the pre-calculation time. It is assumed that the ADC hardware can convert multiple inputs in parallel. Examining Figure 6.8, an equation for T_{DMAX} for any interrupt can be determined as:

$$T_{DMAX} = T_{ADC} + T_{DC} + \sum_{i=1}^N (T_{DCHP_i} + T_{PCHP_i}), \quad (6.1)$$

where T_{DC} is the duty cycle calculation time for that interrupt, N is the number of interrupts with higher priority than that interrupt, while T_{DCHP} and T_{PCHP} are the duty cycle and pre-calculation times of each of the interrupts with higher priority than that interrupt. For example, the value of T_{DMAX} for the highest priority interrupt, *Int1* is given by:

$$T_{DMAX1} = T_{ADC} + T_{DC1}, \quad (6.2)$$

while the value of T_{DMAX} for *Int3* is given by:

$$T_{DMAX3} = T_{ADC} + T_{DC3} + (T_{DC1} + T_{DC2} + T_{PC1} + T_{PC2}). \quad (6.3)$$

Although the T_{DC} and T_{PC} values are equal for each of the algorithms in Figure 6.8, it should be noted that (6.1) is also valid for different values of T_{DC} and T_{PC} , if different control algorithms are executed in each of the ISRs.

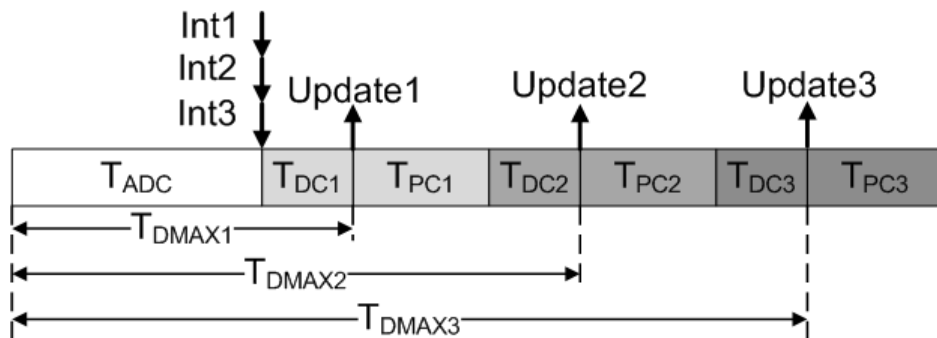


Figure 6.8. DSP delay with standard interrupt control.

6.3 Modified Interrupt Controller

6.3.1 Automatically Enabled Interrupts

In order to avoid the effects of variable DSP delays by fixing the delay at its maximum, it is proposed to modify the interrupt controller so that T_{DMAX} can be reduced to an acceptable value. Figure 6.9 illustrates the resulting delays if all duty cycle calculations for coinciding interrupts are executed before any pre-calculations for the next iteration are carried out. By postponing the pre-calculations until after duty-cycle-updating, the total ADC-sample to duty-cycle-update delay, T_{DMAX} as given in (6.1), is reduced. The reduced delay value for each of the algorithms can be obtained from:

$$T_{DMAX}^* = T_{ADC} + T_{DC} + \sum_{i=1}^N T_{DCHP_i}, \quad (6.4)$$

where the reduction in delay is given by:

$$T_R = \sum_{i=1}^N T_{PCHP_i}. \quad (6.5)$$

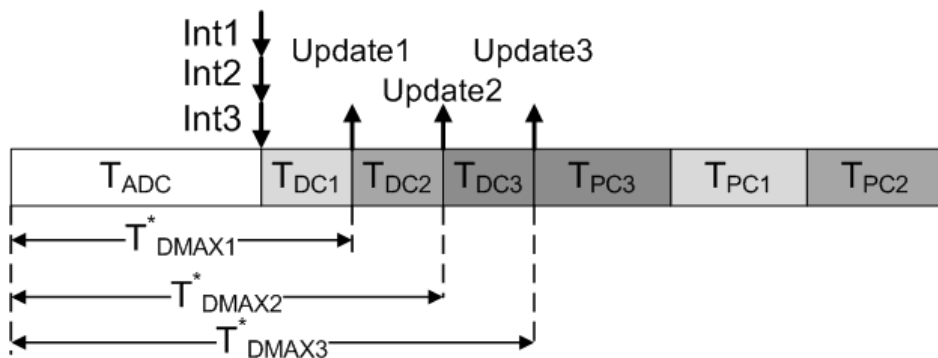


Figure 6.9. DSP delay with modified interrupt control.

The modified interrupt controller can achieve these reduced delays by performing the tasks illustrated in the flowchart of Figure 6.10. If multiple interrupts occur simultaneously, as in Figure 6.9, the highest priority control algorithm is selected first. At this point all other interrupts are disabled, a dedicated counter is loaded with a pre-configured duty cycle calculation time. Counting is subsequently enabled and the execution of the control algorithm commences. After the counter determines that the duty cycle calculation time has elapsed, all interrupts are re-enabled. At this stage the DPWM should have been updated with the newly calculated duty cycle value. Before the pre-calculations can begin, execution is interrupted by the highest priority pending interrupt. Again all other interrupts are disabled, the counter is reloaded and counting is enabled. The same applies for the next priority interrupt and so on. After no further interrupts are pending, the DSP continues with the execution of the pre-calculations for each of algorithms that were interrupted.

If no interrupts are pending after the duty cycle calculation time counter expires, the execution of the control algorithm can continue immediately with the execution of the pre-calculations for the next iteration of the control algorithm. The pre-calculations may be interrupted at any stage if another interrupt signal becomes active before they have been completed. It should be noted that the duty cycle calculation time is configurable for each of the algorithms in order to provide the flexibility to execute a different algorithm for each individual power converter.

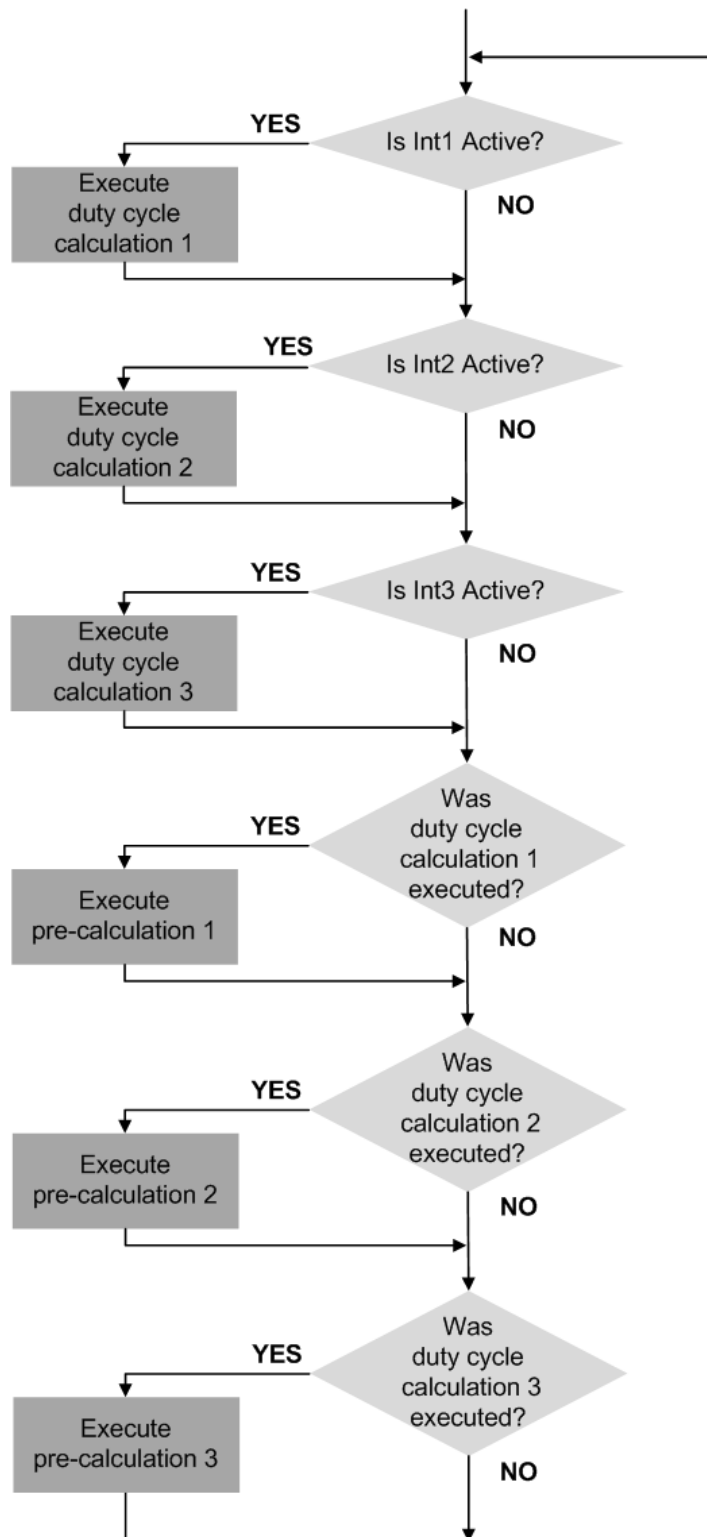


Figure 6.10. Flowchart of modified interrupt controller behaviour.

6.3.2 Modified Interrupt Controller Hardware

The substantial benefits of the modified interrupt scheme can be achieved by augmenting a conventional interrupt controller with minimal additional hardware, as illustrated in Figure 6.11. The highlighted regions of Figure 6.11 indicate the additional components that were added to the interrupt controller described in Section 4.5.2.

The main enhancement is a counter to determine when to re-enable interrupts. Some extra registers are also required. A *'ret_adr'* register is required for each of the interrupts to store the return address for the ISR if its execution is interrupted by a higher priority interrupt. The duty-cycle calculation times must also be stored for each algorithm in terms of the number of instructions required, in order to be accessed by the interrupt disable counter. These values should be loaded into special function *'int_cnt'* registers during the initialisation section of the program code.

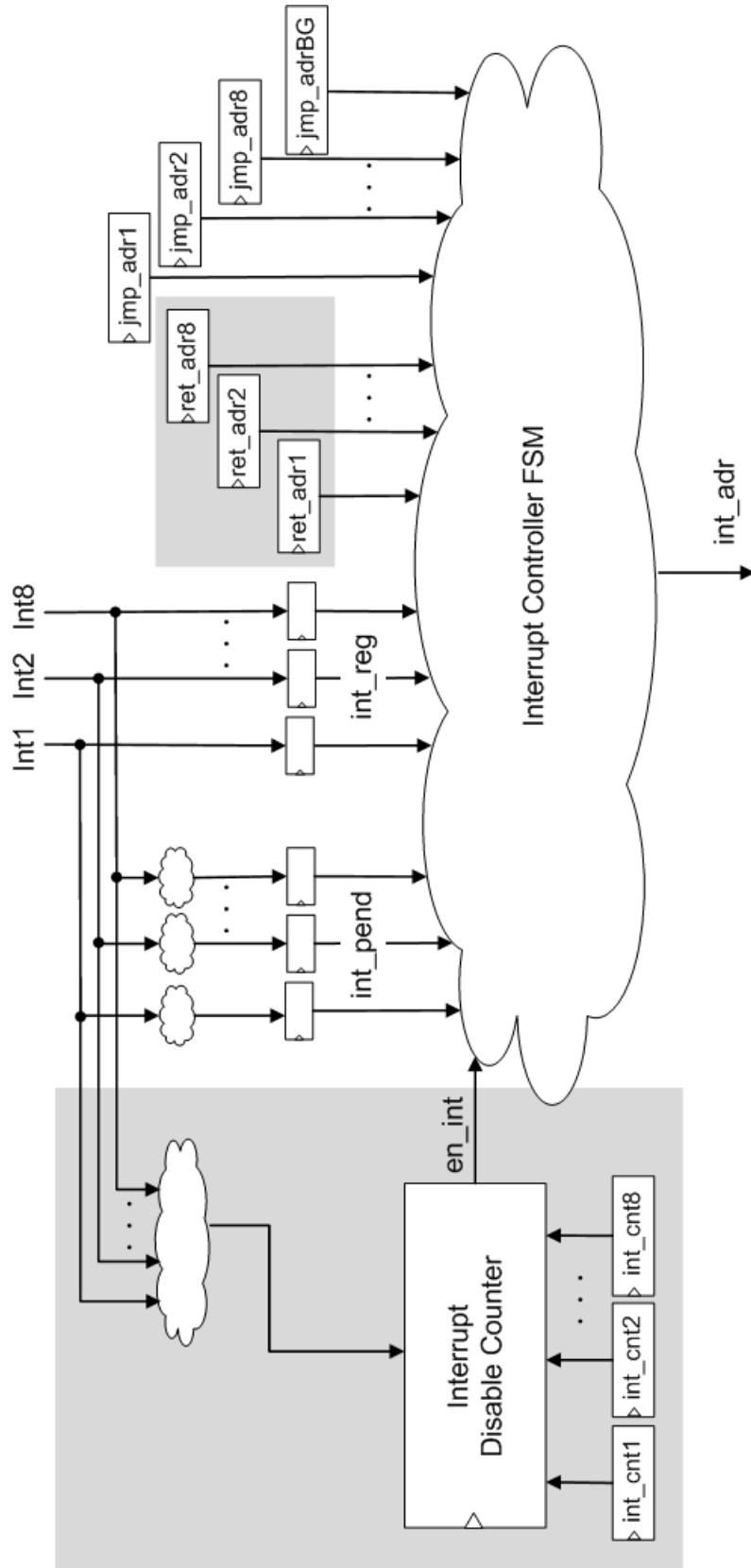


Figure 6.11. Modified interrupt controller with additional hardware blocks highlighted.

6.3.3 Comparison With Standard Interrupt Controllers

By inspecting the operation of a number of existing DSPs that are used in power converter control applications [14, 15, 17], it can be seen that they do not facilitate the automatic disabling of interrupts. For this reason, the modified interrupt controller technique cannot be directly implemented by such DSPs. Alternative methods that can be implemented using these DSPs and their drawbacks are discussed here.

An alternative method that can be implemented by existing DSPs is to manually re-enable interrupts after the duty cycle has been calculated. This requires an additional instruction at the end of each duty cycle calculation (*DC*) before the pre-calculation (*PC*) begins, as illustrated in Figure 6.12. This method clearly adds to the ADC-sample to duty-cycle-update delay. By re-enabling interrupts automatically the proposed scheme frees up more execution time in each switching cycle which could be used to execute instructions of a more complex control algorithm or to perform additional monitoring operations.

Some existing DSPs also only facilitate interrupt nesting where higher priority interrupts are allowed to interrupt the currently executing ISR. This does not comply with the requirement for every algorithm to be interruptible by every other interrupt after a certain number of instructions have been executed. The priority of the interrupts in the proposed method relates to the order in which they will be processed if they occur simultaneously and bears no influence on whether or not they can interrupt other ISRs. In order to overcome the difficulties associated with interrupt nesting where only higher priority interrupts are allowed to interrupt the currently executing ISR, an alternative method can be implemented using such DSPs which has separate interrupts for the duty cycle calculation and pre-calculation sections of each control algorithm. After the duty cycle calculation has been completed a lower priority software interrupt (*SW INT*) is triggered for the pre-calculation. This allows other pending duty cycle calculation interrupts to be processed before any pre-calculations take place. The drawback of this method is that a return from interrupt (*RETI*) instruction must be executed after each duty cycle calculation to exit from the ISR

before processing the next pending interrupt, as illustrated in Figure 6.13. This adds a significant number of additional clock cycles between the duty cycle calculations for each algorithm and thus increases T_{DMAX} . The proposed method avoids this problem by keeping the duty cycle calculation and pre-calculation in the same ISR and only exiting from that routine if another interrupt is pending.

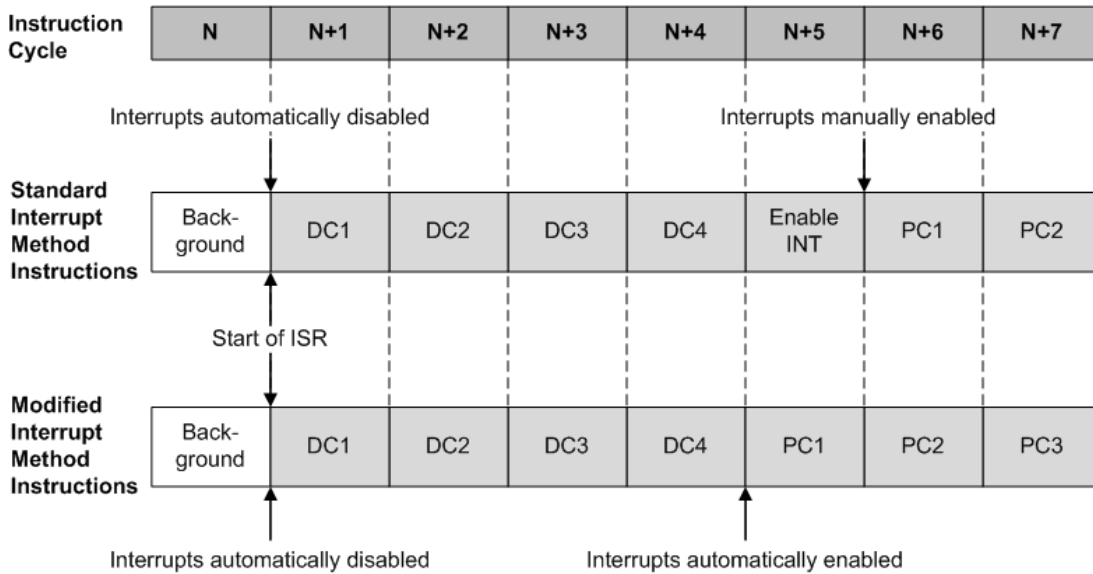


Figure 6.12. Comparison of modified interrupt control method with a standard method involving manually disabling and enabling interrupts.

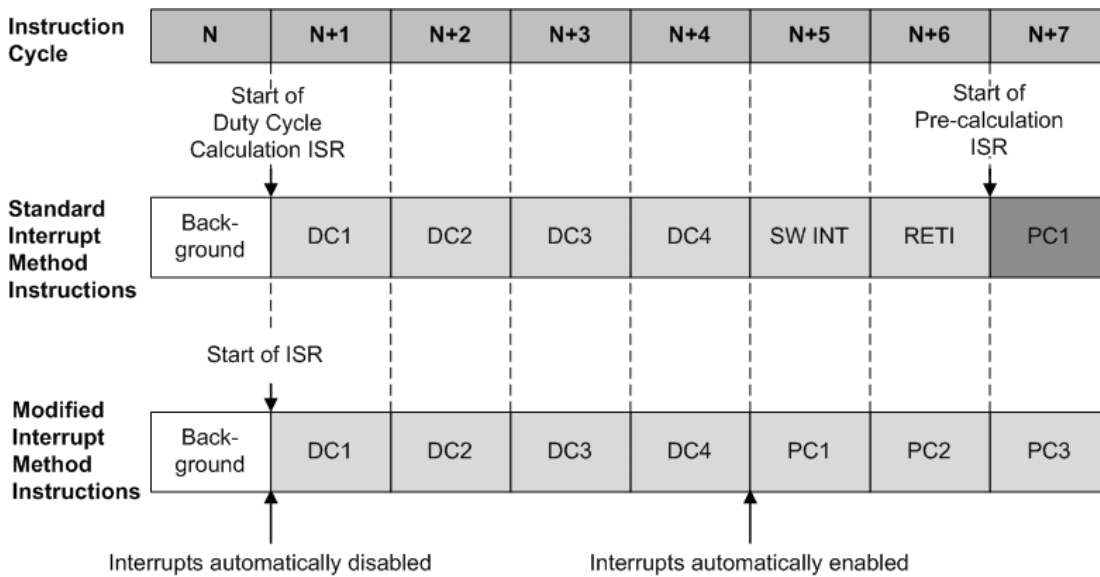


Figure 6.13. Comparison of modified interrupt control method with a standard method involving separate interrupt service routines for the duty cycle calculation and pre-calculation code sections.

6.4 Experimental Verification

6.4.1 Experimental Platform

In order to evaluate the performance improvement provided by the proposed interrupt controller, it has been compared with the standard interrupt control method that is applied in multi-rail DC-DC converter systems, whose operation was illustrated in Figure 6.8. The comparison is based on the application of both interrupt control methods to the power converter system described in Section 5.5.2 consisting of three independent 12 V to 1.5 V buck converters.

Rail 0 and Rail 2 were configured to operate at a switching frequency of 500 kHz, while Rail 1 was configured to operate at a switching frequency of 495 kHz. Thus the ratio of the Rail 1 switching frequency to the Rail 0 switching frequency is 0.99. The control algorithm for Rail 0 has the highest interrupt priority, followed by Rail 1 and then Rail 2. The proposed interrupt controller was incorporated into the dual MAC DSP core design discussed in Chapter 4 and implemented using the same FPGA platform as described in Section 5.5.1, which is also illustrated in Figure 6.14. Each DSP clock cycle is 30 ns, corresponding to a clock frequency of 33 MHz. The Verilog code was synthesised separately using both the standard and the modified interrupt controllers in order to compare each of the techniques.

A third-order linear compensator was used in turn to regulate the output voltage of each of the DC-DC converters. Details of the compensator delays and ADC-conversion delay are given in Table 6.1. These values were applied to (6.1) and (6.4) to determine the T_{DMAX} delays for the standard and proposed interrupt methods. Table 6.2 summarises the T_{DMAX} delays for each of the voltage rails for the third-order compensator. It also includes the percentage reduction in delay provided by the proposed interrupt scheme. Although the modified interrupt method does not provide any reduction in T_{DMAX} for the highest priority interrupt, it significantly reduces T_{DMAX} for all other interrupts. It should be noted that the delay for Rail 1 corresponds to the delay that would occur for each of the rails in a multi-rail system with an integer

switching frequency ratio. Table 6.3 illustrates the benefit of the proposed interrupt control scheme for a hypothetical system involving up to eight rails, where it can be seen that the percentage reduction in T_{DMAX} is similar for each added rail when the number of voltage rails to be controlled by the system is increased.

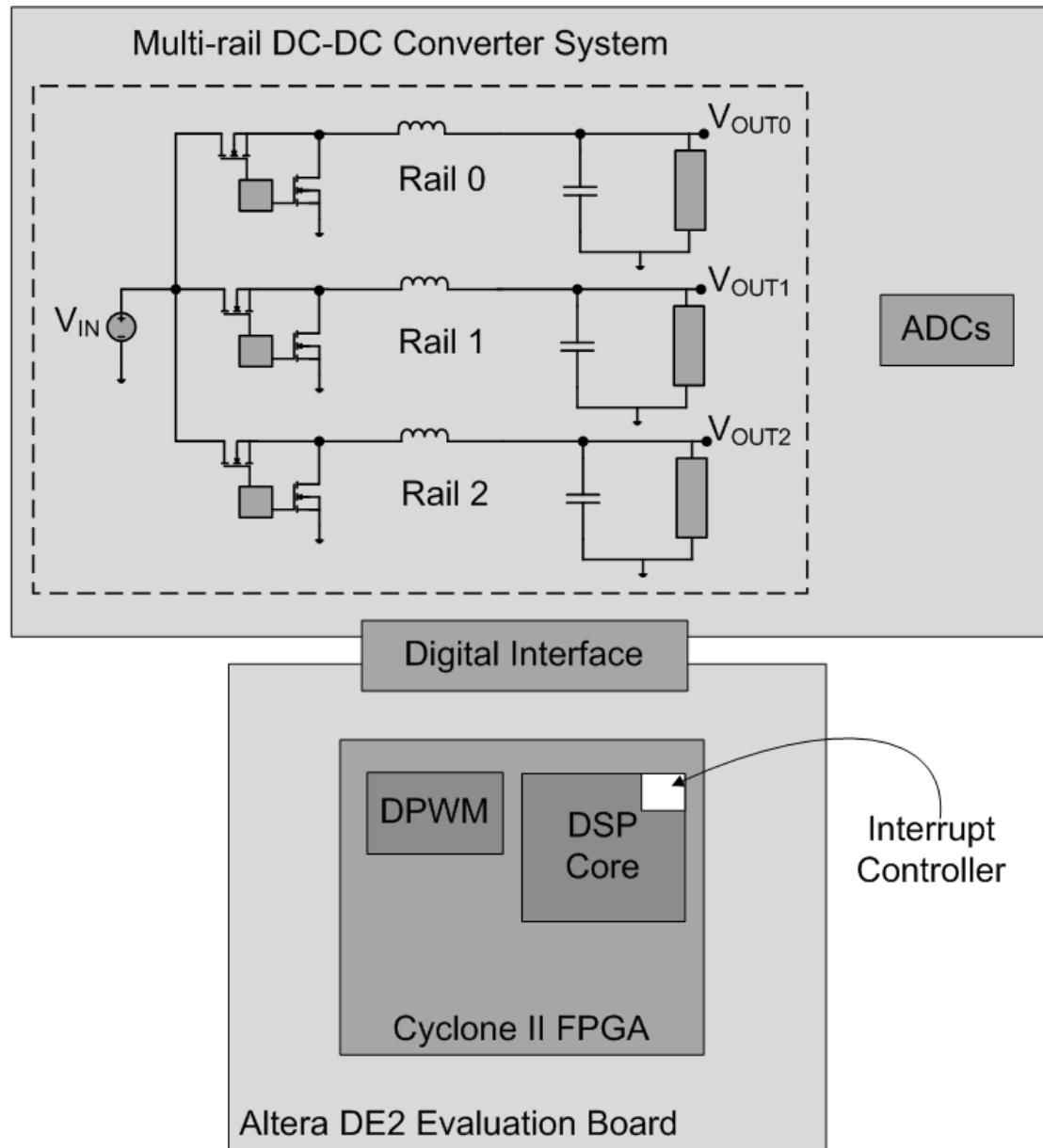


Figure 6.14. Experimental platform consisting of three rail power converter system and interrupt controller implemented as part of the dual MAC DSP core on a Cyclone II FPGA.

Table 6.1. Digital control system delay parameters.

Operation	Delay (DSP clock cycles)	Delay (ns)
Duty cycle calculation	7 cycles	210 ns
Pre-calculation	5 cycles	150 ns
ADC-conversion	–	180 ns

Table 6.2. Maximum ADC-sample to duty-cycle-update delays for 3-rail power converter system.

Interrupt Method	Rail 0	Rail 1	Rail 2
T_{DMAX} for Standard Interrupt Control	390 ns	750 ns	1110ns
T_{DMAX} for Modified Interrupt Control	390 ns	600 ns	810 ns
Reduction in T_{DMAX} (%Reduction in T_{DMAX})	0 ns (0 %)	150 ns (20 %)	300 ns (27 %)

Table 6.3. Maximum ADC-sample to duty-cycle-update delays for up to 8 rails.

Interrupt Method	Rail 3	Rail 4	Rail 5	Rail 6	Rail 7
T_{DMAX} for Standard Interrupt Control	1470	1830	2190	2550	2910
T_{DMAX} for Modified Interrupt Control	1020	1230	1440	1650	1860
Reduction in T_{DMAX} (%Reduction in T_{DMAX})	450 (31 %)	600 (33 %)	750 (34 %)	900 (35 %)	1050 (36%)

6.4.2 Interrupt Controller Operation

The operation of both the standard interrupt controller and the modified interrupt controller was verified using the experimental platform, as illustrated in the waveforms of Figure 6.15 and Figure 6.16 respectively. The ISR signal indicates when control algorithms are running for Rails 0, 1 and 2, while ‘3’ indicates when the DSP is in background mode during the idle time between the ISRs.

For standard interrupt control all ISRs run without interruption, as illustrated in Figure 6.15. During the time interval shown, interrupt signals *Int0* and *Int1* occur close together. In the first instance *Int0* occurs before *Int1*, while in the second instance *Int1* occurs just before *Int0*. *ISR 0* is therefore not executed until after *ISR 1* has completed. Figure 6.16 illustrates the contrasting operation of the modified interrupt controller. In this case *ISR 0* is interrupted by *ISR 1* after the duty cycle has been calculated. After *ISR 1* has finished, execution returns to *ISR 0* to complete the pre-calculations for the next iteration. Similarly when *Int1* occurs next, *ISR 1* is interrupted by *ISR 0* after the duty cycle has been calculated. *ISR 2* runs without interruption.

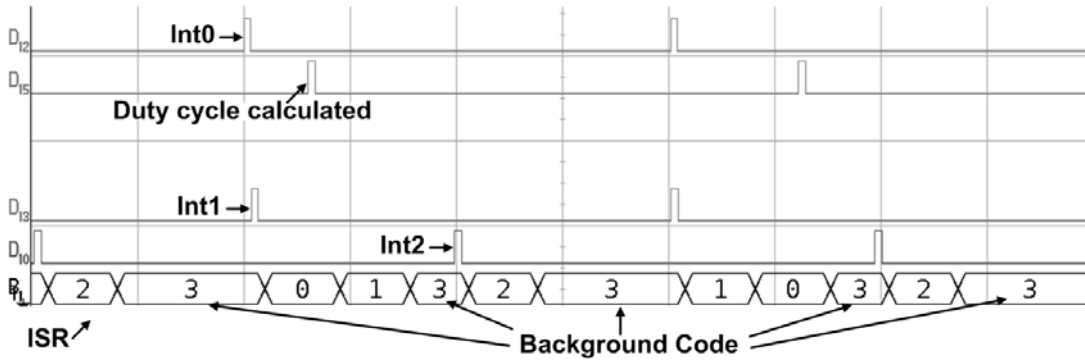


Figure 6.15. Standard interrupt control operation (Horizontal axis: 500 ns/division).

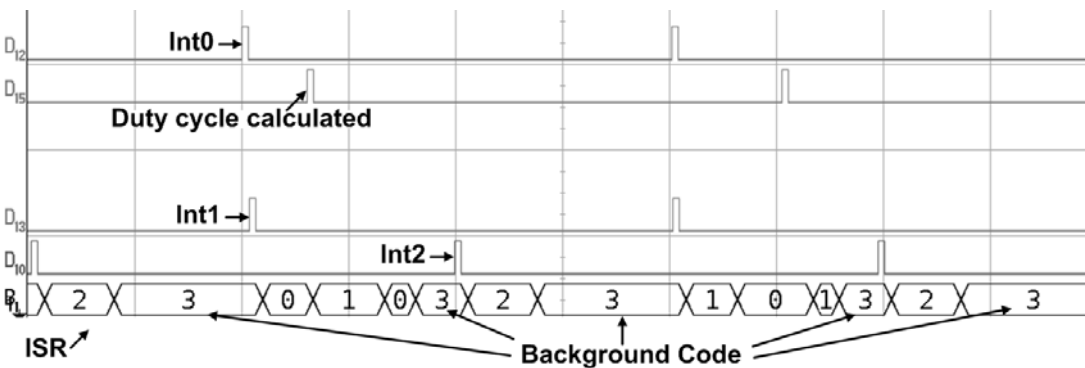


Figure 6.16. Modified interrupt control operation (Horizontal axis: 500 ns/division).

6.4.3 Performance Improvements

Applying load current steps to the output of DC-DC converters is a standard method of evaluating their dynamic performance. In the test system identical 3 A current steps were applied to the output of the DC-DC converters while using both the standard and modified interrupt control methods. A 3P3Z compensator was used in each case to regulate the output voltage, where the shorter T_{DMAX} delay of the modified interrupt controller facilitated the use of a wider bandwidth compensator compared with the standard interrupt controller.

The resulting Rail 1 output voltage transient responses caused by the current steps, for the cases of the standard and modified interrupt controllers, are demonstrated in Figure 6.17 and Figure 6.18. The restriction of the long T_{DMAX} delay for the standard method results in a slow response to the load current step as illustrated in Figure 6.17. The modified interrupt controller provides better performance and with the wider bandwidth compensator enables a faster response with less overshoot to be obtained as Figure 6.18 shows.

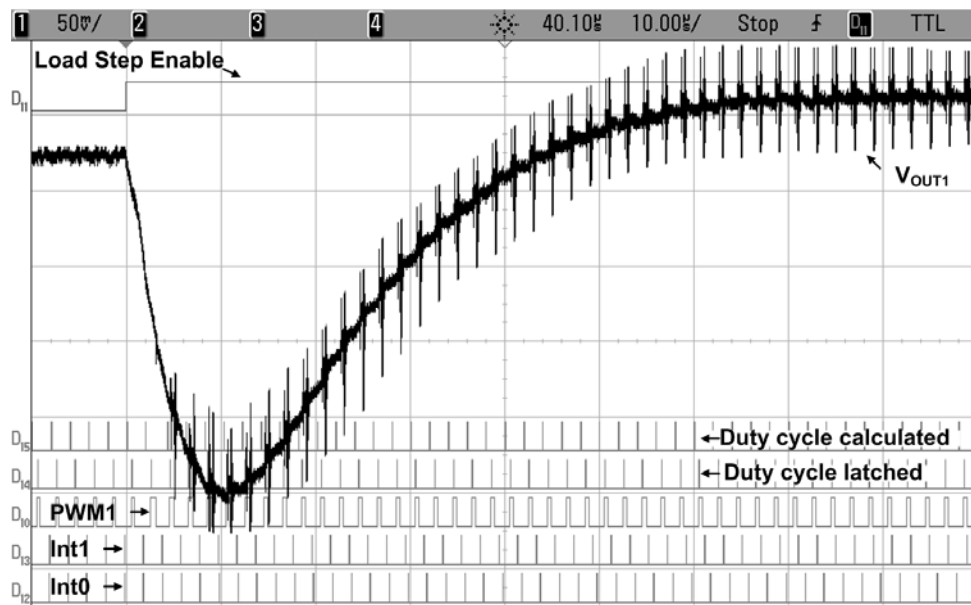


Figure 6.17. Load transient response for standard interrupt controller.

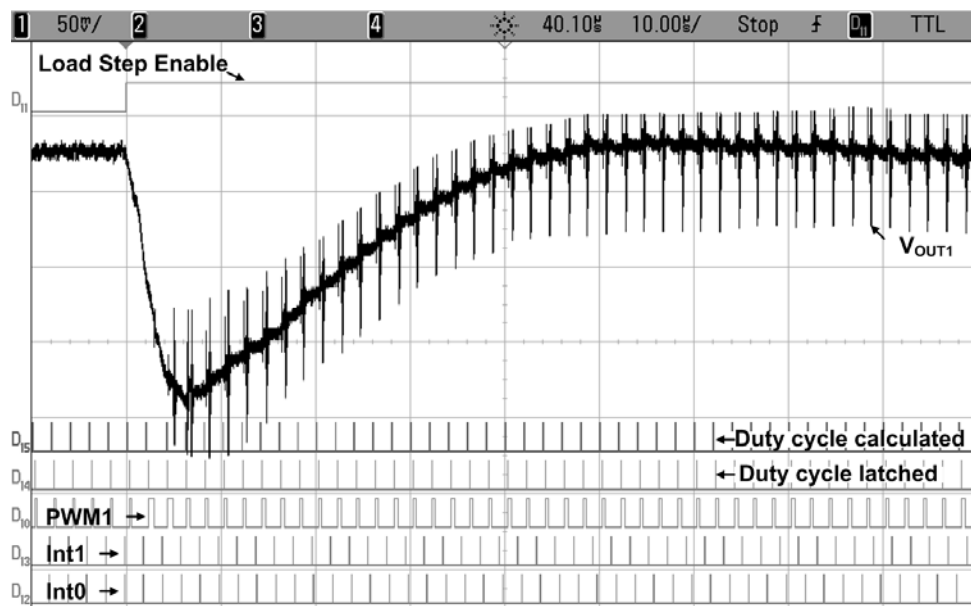


Figure 6.18. Load transient response for modified interrupt controller with wider bandwidth compensator.

6.5 Summary

A drawback of using a standard DSP to control multiple power converters is its limitation in dealing with switching frequencies with non-integer ratios. Using multiple analogue or digital compensators in such systems is not feasible due to cost constraints. A modified interrupt controller for digital signal processors has been proposed in this chapter that performs significantly better than standard DSPs and at a reduced hardware cost compared with multiple compensator solutions.

The variation in delay from when the ADC samples the output voltage to when the duty cycle is updated has been identified as a critical factor to be considered when implementing the digital control system for such an application. Fixing the delay to its maximum value produces reasonable performance using a conventional DSP. By incorporating modified interrupt control logic the worst case delay is reduced. In comparison with the varying or excessive ADC-sample to duty-cycle-update delay in existing DSPs, the proposed interrupt method uses a constant, reduced and hence more desirable delay. As shown, this yields improved performance compared with that obtained using a standard interrupt controller.

Applying this DSP to a multi-rail power supply system provides the designer with flexibility in choosing arbitrary switching frequencies and also optimal component values for the individual SMPCs, thereby allowing optimization of the efficiency and performance of the individual power converters.

Chapter 7

Conclusions

7.1 Summary and Achievements

This thesis has described the architectural characterisation, optimisation and implementation of a new DSP core designed specifically for executing control algorithms for multi-rail DC-DC converter applications.

The characteristics of the application system were initially reviewed, while also examining the advantages and disadvantages of existing digital controller implementations. The performances of a number of potential digital controller architectures were investigated and compared, by analysing the implementation of typical power control algorithms on those architectures. From this analysis the proposed dual MAC DSP core was selected as the most suitable architecture.

The architectural features of the dual MAC DSP were specified, which were particularly targeted towards the execution of multiple digital control algorithms for multi-rail systems. The hardware implementation of the architecture and an evaluation of its performance when applied to a multi-rail power conversion system were subsequently presented. Multi-rail system applications where there is a non-integer ratio between the switching frequencies of the power converters were also examined

and a modified DSP-based controller solution that provides multiple benefits in such applications was proposed and verified.

Details of the work have been published in a number of peer-reviewed technical papers [81, 82, 93, 105, 110]. The work has resulted in a number of achievements, which are significant to the area of digital control for power conversion. The main achievements are listed below:

- Characterisation of the computational demand of typical digital control algorithms in multi-rail POL converter systems
- Analysis and evaluation of a variety of processor architectures for their suitability for implementation in DSP-based digital controllers
- Specification and RTL implementation of a two-way VLIW dual MAC DSP core architecture for SMPC control including:
 - Custom assembly language based instruction set
 - Resource optimised dual MAC datapath
 - Concise addressing and instruction encoding
- Implementation of novel features of a DSP-based digital controller for multi-rail power conversion applications, for example:
 - Register file memory architecture with context-based memory segmentation facilitating fast access to relevant control algorithm data
 - Interrupt triggered context switching for efficient multiple control algorithm execution
 - Automatic interrupt enable control permitting fast duty cycle computations in non-integer switching frequency ratio systems
- Verification of an FPGA-based implementation of the dual MAC DSP core in a digital power controller system by means of its application to a prototype multi-rail power supply

- Enabling improved transient response of power converters controlled by DSP-based digital controllers by reducing control algorithm execution time
- Facilitating the execution of an increased number of complex control algorithms and additional background code, thus enabling the enhancement of power conversion performance, efficiency and adaptability
- Reduction in cost of digital controller requirements in multi-rail systems with a non-integer switching frequency ratio through the use of a single DSP-based controller with specialised interrupt controller
- Enabling flexible choice of power converter switching frequencies in multi-rail systems using specialised interrupt controller, in order to obtain optimal efficiency or transient response of individual power converters
- Reduction of digital controller delay and subsequent power converter performance improvement in multi-rail systems with a non-integer switching frequency ratio using DSP core with specialised interrupt controller

7.2 Future work

Throughout the course of this study a number of interesting alternative research avenues were identified. These include exploring the integration of additional optional features in the DSP core that could further enhance its performance and functionality. A number of other issues also arose during the completion of the work that were relevant but not central to the theme of the thesis.

7.2.1 Potential Improvements to the DSP Core

Additional digital design strategies could be applied and evaluated in an attempt to shorten the execution time of digital power control algorithms on the dual MAC DSP.

By using a longer instruction word the destination of the result of a computational operation could be included in the instruction word. This would mean that fewer data movement instructions would be required, which would mean a reduced algorithm execution time at the cost of a larger, more expensive program memory.

The pipeline of the dual MAC DSP core could be further optimised in order to attain a higher clock frequency. While a basic three stage pipeline was implemented for the dual MAC DSP, a more detailed analysis could be undertaken to determine a more suitable pipeline depth and to reduce the critical path through the datapath. The architecture of the multiplier-accumulator unit could also be examined in detail in an attempt to reduce the datapath latency, particularly in the case of an ASIC-based implementation of the DSP core.

Additional shifters could be added to the datapath of the processor in order to correctly align data values with different fractional formats before using them as operands in computational operations.

The area and data access time of the register file memory architecture is dependent on the number of banks, the number of registers per bank and the number of read/write ports [111]. In order to reduce the logic requirements, the number of registers per bank could be reduced. Register file memory banks of sixteen registers could be used if only low-order algorithms, having smaller memory requirements, were to be executed using this DSP core architecture. However, this would confine the DSP to executing basic algorithms, which would require fewer coefficients and data variables, resulting in a lower performance compensator solution. On the other hand, depending on the number of power rails to be controlled, additional register banks may need to be included.

Memory segregation may also be necessary to alleviate area and timing issues. For example, it might be beneficial to separate the register file memory into two individual sources with half the number of ports, each dedicated to one datapath section, with the possibility of having limited access to the memory of the other datapath section. Figure 7.1 shows a three-dimensional memory block illustrating the possibilities in sharing data among the two datapath sections and across register banks. Data can either be accessible by both datapath sections for individual algorithms or by both datapath sections for all algorithms or else be restricted to individual datapath sections for individual algorithms. However, this may also lead to reduced flexibility in terms of the possible algorithms that can be executed as it can render the DSP unsuitable for executing more complex algorithms.

Increasing the functionality of the DSP core could also be examined as a means of broadening the applicability of the processor. The instruction set could be supplemented with additional instructions that would permit alternative computational operations, such as division, logic operations or non-linear functions, to be executed. Conditional branch instructions could also be added to the instruction set to facilitate the execution of more complex algorithms and background tasks. Although the dual MAC DSP core is intended to be a stand-alone processor, it could alternatively act as a co-processor to a supervisory microcontroller in the event that the quantity of power management related instructions was excessive for them to be executed by the dual MAC processor alone.

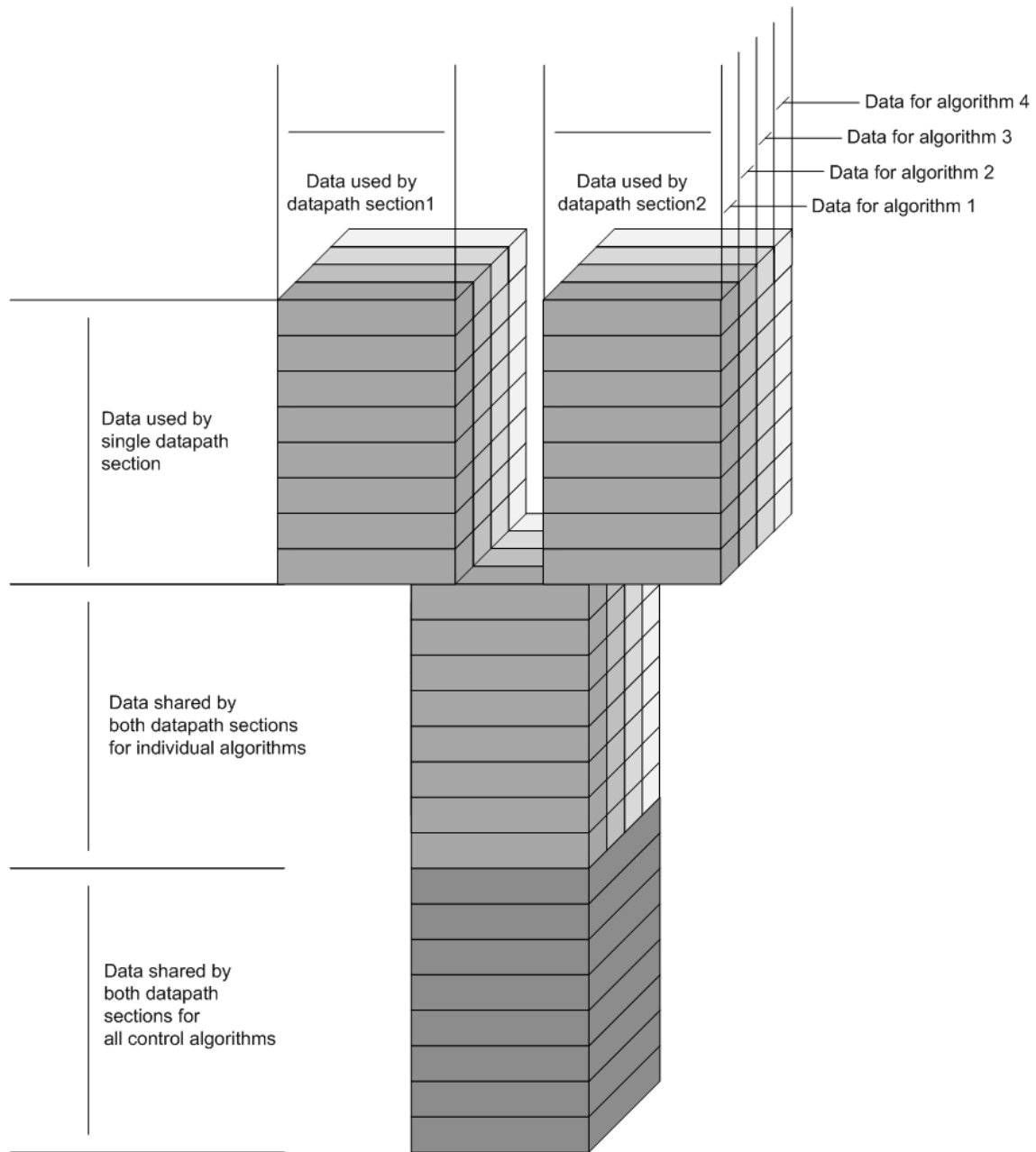


Figure 7.1. Register access possibilities for dual datapath processor with four bank register file.

7.2.2 Application and System Level Considerations

In addition to modifying the dual MAC architecture, a variety of system level changes could be made which would be beneficial in applying the processor in a multi-rail application.

A number of peripherals could be added, for example a mixed-signal current estimator block, which would facilitate the detection of transient events in the output voltage caused by changing load conditions [30, 112]. Such elements could facilitate the implementation of multi-mode control algorithms incorporating non-linear control that are applicable to SMPCs with stringent transient performance requirements [56]. Transition between various operating modes of the buck converter could also be controlled by the DSP core using these event detectors in order to maintain high power conversion efficiency across a wide load current range.

An in-depth investigation of the sampling requirements of multi-rail power conversion systems could be undertaken to determine an optimal analogue to digital conversion solution. The limitations of using multiple ADCs and multi-channel ADCs could be explored and compared for the multi-rail application, especially in the context of implementing multi-sampling digital controllers that process more than one sample per switching period [35]. An important aspect in the development of the ADC solution would be the inclusion of an error generator that would subtract the sampled value from a supplied digital reference for each of the rails being controlled, similar to existing ADCs for single rail applications [29]. This would relieve the computational burden on the DSP core of having to calculate the error value in each iteration of each control algorithm.

Although the assembly language based programming of the DSP core facilitates reasonably straightforward configuration, alternative configuration approaches based on using the C programming language or more favourably using a GUI would permit those with less experience of assembly language programming to exploit the benefits of the DSP core.

The requirements of multi-rail systems with a large number of rails, for example more than ten rails, could be explored in order to determine the applicability of the dual MAC processor to such systems. A multi-core digital controller approach could be envisaged where each core would be based on the dual MAC architecture proposed here.

7.3 Concluding Remarks

The work in this thesis has contributed towards the advancement of the constantly evolving intelligent power supply. The thesis itself has outlined an appropriate method for devising, developing, implementing and testing a high performance DSP architecture for executing digital power control algorithms.

The thesis serves to promote the adoption of high performance optimised DSP architectures for multi-rail power control applications. Such architectures are ideally suited for integration in future mainstream power supplies, which are expected to be generic, programmable devices [113].

Bibliography

- [1] T. W. Martin and S. S. Ang, "Digital control for switching converters", in *IEEE International Symposium on Industrial Electronics*, 1995, pp. 480-484.
- [2] A. Prodic and D. Maksimovic, "Digital PWM controller and current estimator for a low-power switching converter", in *IEEE Workshop on Computers in Power Electronics*, 2000, pp. 123-128.
- [3] Y. F. Liu, E. Meyer, and X. Liu, "Recent Developments in Digital Control Strategies for DC/DC Switching Power Converters", *IEEE Transactions on Power Electronics*, vol. 24, no. 11, pp. 2567-2577, 2009.
- [4] X. Zhang and D. Maksimovic, "Multimode Digital Controller for Synchronous Buck Converters Operating Over Wide Ranges of Input Voltages and Load Currents", *IEEE Transactions on Power Electronics*, vol. 25, no. 8, pp. 1958-1965, 2010.
- [5] L. Corradini, A. Babazadeh, A. Bjeletic, and D. Maksimovic, "Current-Limited Time-Optimal Response in Digitally Controlled DC-DC Converters", *IEEE Transactions on Power Electronics*, vol. 25, no. 11, pp. 2869-2880, 2010.
- [6] L. Jia, D. Wang, E. Meyer, Y. F. Liu, and P. C. Sen, "A novel digital capacitor charge balance control algorithm with a practical extreme voltage detector", in *IEEE Energy Conversion Congress and Exposition*, 2010, pp. 514-521.
- [7] H. Al-Atrash and I. Batarseh, "Digital Controller Design for a Practicing Power Electronics Engineer", in *Applied Power Electronics Conference and Exposition*, 2007, pp. 34-41.
- [8] A. Costabeber, P. Mattavelli, and S. Saggini, "Digital Time-Optimal Phase Shedding in Multiphase Buck Converters", *IEEE Transactions on Power Electronics*, vol. 25, no. 9, pp. 2242-2247, 2010.
- [9] J. A. Abu Qahouq, "Control Scheme for Sensorless Operation and Detection of CCM and DCM Operation Modes in Synchronous Switching Power Converters", *IEEE Transactions on Power Electronics*, vol. 25, no. 10, pp. 2489-2495, 2010.
- [10] A. Costabeber, P. Mattavelli, S. Saggini, and A. Bianco, "Digital autotuning of DC-DC converters based on model reference impulse response", in *IEEE Applied Power Electronics Conference and Exposition*, 2010, pp. 1287-1294.
- [11] T. Takayama and D. Maksimovic, "Digitally controlled 10 MHz monolithic buck converter", in *IEEE Workshop on Computers in Power Electronics*, 2006, pp. 154-158.

- [12] H. C. Foong, M. T. Tan, and Y. Zheng, “A dual-mode digital DC-DC converter based on distributed arithmetic”, in *International Symposium on Power Electronics Electrical Drives Automation and Motion*, 2010, pp. 1146-1149.
- [13] D. W. Kang, Y. B. Kim, and J. T. Doyle, “A high-efficiency fully digital synchronous buck converter power delivery system based on a finite-state machine”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 3, pp. 229-240, 2006.
- [14] Analog Devices Reference Guide. (2003, Jan.). *ADSP-2199x Mixed Signal DSP Controller Hardware Reference* [Online]. Available: http://www.analog.com/static/imported-files/processor_manuals/50171261290224ADSP_2199x_Hardware_Reference.pdf
- [15] Texas Instruments Reference Guide. (2009, Jan.). *TMS320C28x CPU and Instruction Set Reference Guide* [Online]. Available: <http://focus.ti.com/lit/ug/spru430e/spru430e.pdf>
- [16] Microchip Technology Inc. Data Sheet. (2010, Feb.). *dsPIC33FJ32GS406/606/608/610 and dsPIC33FJ64GS406/606/608/610 High-Performance, 16-bit Digital Signal Controllers Data Sheet* [Online]. Available: <http://ww1.microchip.com/downloads/en/DeviceDoc/70591C.pdf>
- [17] Freescale Semiconductor Reference Guide. (2005, Nov.). *DSP56800E Reference Manual* [Online]. Available: http://cache.freescale.com/files/dsp/doc/ref_manual/DSP56800ERM.pdf
- [18] Silicon Laboratories Data Sheet. (2007, June). *Si8250 Digital Power Controller* [Online]. Available: <http://www.silabs.com/Support Documents/TechnicalDocs/Si8250.pdf>
- [19] Texas Instruments Data Sheet. (2010, June). *TMS320F2803x Piccolo Microcontrollers* [Online]. Available: <http://focus.ti.com/lit/ds/symlink/tms320f28033.pdf>
- [20] S. Abedinpour, B. Bakkaloglu, and S. Kiaei, “A Multistage Interleaved Synchronous Buck Converter With Integrated Output Filter in 0.18 μm SiGe Process”, *IEEE Transactions on Power Electronics*, vol. 22, no. 6, pp. 2164-2175, 2007.
- [21] L. Guo, J. Y. Hung, and R. M. Nelms, “Evaluation of DSP-Based PID and Fuzzy Controllers for DC-DC Converters”, *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 2237-2248, 2009.

- [22] S. M. Ahsanuzzaman, A. Parayandeh, A. Prodic, and D. Maksimovic, "Load-interactive steered-inductor dc-dc converter with minimized output filter capacitance", in *Applied Power Electronics Conference and Exposition*, 2010, pp. 980-985.
- [23] A. Pizzutelli, A. Carrera, and S. Saggini, "Relay based digital auto-tuning techniques for isolated DC-DC switching converters", in *International Telecommunications Energy Conference*, 2007, pp. 533-540.
- [24] R. W. Erickson and D. Maksimovic, *Fundamentals of Power Electronics*, 2nd ed.: Kluwer Academic Publishers, 2001.
- [25] Z. Lukic, Z. Zhao, S. Ahsanuzzaman, and A. Prodic, "Self-Tuning Sensorless Digital Current-Mode Controller with Accurate Current Sharing for Multi-Phase DC-DC Converters", in *IEEE Applied Power Electronics Conference and Exposition*, 2009, pp. 264-268.
- [26] A. Kelly and K. Rinne, "Sensorless current-mode control of a digital dead-beat DC-DC converter", in *Applied Power Electronics Conference and Exposition*, 2004, pp. 1790-1795.
- [27] B. J. Patella, A. Prodic, A. Zirger, and D. Maksimovic, "High-frequency digital PWM controller IC for DC-DC converters", *IEEE Transactions on Power Electronics*, vol. 18, no. 1, pp. 438-446, 2003.
- [28] J. Xiao, A. V. Peterchev, J. Zhang, and S. R. Sanders, "A 4-uA quiescent-current dual-mode digitally controlled buck converter IC for cellular phone applications", *IEEE Journal of Solid-State Circuits*, vol. 39, no. 12, pp. 2342-2348, 2004.
- [29] A. Parayandeh and A. Prodic, "Programmable Analog-to-Digital Converter for Low-Power DC-DC SMPS", *IEEE Transactions on Power Electronics*, vol. 23, no. 1, pp. 500-505, 2008.
- [30] A. Radic, Z. Lukic, A. Prodic, and R. de Nie, "Minimum deviation digital controller IC for single and two phase dc-dc switch-mode power supplies", in *IEEE Applied Power Electronics Conference and Exposition*, 2010, pp. 1-6.
- [31] A. V. Peterchev, J. Xiao, and S. R. Sanders, "Architecture and IC implementation of a digital VRM controller", *IEEE Transactions on Power Electronics*, vol. 18, no. 1, pp. 356-364, 2003.
- [32] Intersil Data Sheet. (2007, Feb.). *PX3511A, PX3511B Advanced Synchronous Rectified Buck MOSFET Drivers with Protection Features* [Online]. Available: <http://www.intersil.com/data/fn/fn6462.pdf>

- [33] L. Corradini, P. Mattavelli, E. Tedeschi, and D. Trevisan, “High-Bandwidth Multisampled Digitally Controlled DC-DC Converters Using Ripple Compensation”, *IEEE Transactions on Industrial Electronics*, vol. 55, no. 4, pp. 1501-1508, 2008.
- [34] S. Effler, Z. Lukic, and A. Prodic, “Oversampled digital power controller with bumpless transition between sampling frequencies”, in *IEEE Energy Conversion Congress and Exposition*, 2009, pp. 3306-3311.
- [35] Z. Lukic, A. Radic, A. Prodic, and S. Effler, “Oversampled digital controller IC based on successive load-change estimation for dc-dc converters”, in *IEEE Applied Power Electronics Conference and Exposition*, 2010, pp. 315-320.
- [36] L. Corradini, E. Orietti, P. Mattavelli, and S. Saggini, “Digital Hysteretic Voltage-Mode Control for DC-DC Converters Based on Asynchronous Sampling”, *IEEE Transactions on Power Electronics*, vol. 24, no. 1, pp. 201-211, 2009.
- [37] Z. Lukic, K. Wang, and A. Prodic, “High-frequency digital controller for dc-dc converters based on multi-bit sigma-delta pulse-width modulation”, in *IEEE Applied Power Electronics Conference and Exposition*, 2005, pp. 35-40.
- [38] A. V. Peterchev and S. R. Sanders, “Quantization resolution and limit cycling in digitally controlled PWM converters”, *IEEE Transactions on Power Electronics*, vol. 18, no. 1, pp. 301-308, 2003.
- [39] E. O'Malley and K. Rinne, “A programmable digital pulse width modulator providing versatile pulse patterns and supporting switching frequencies beyond 15 MHz”, in *IEEE Applied Power Electronics Conference and Exposition*, 2004, pp. 53-59.
- [40] A. Kelly and K. Rinne, “High Resolution DPWM in a DC-DC Converter Application Using Digital Sigma-Delta Techniques”, in *IEEE Power Electronics Specialists Conference*, 2005, pp. 1458-1463.
- [41] Z. Lukic, N. Rahman, and A. Prodic, “Multibit Sigma-Delta PWM Digital Controller IC for DC-DC Converters Operating at Switching Frequencies Beyond 10 MHz”, *IEEE Transactions on Power Electronics*, vol. 22, no. 5, pp. 1693-1707, 2007.
- [42] Intersil Application Guide. (2004, March). *Power Management Application Guide for Xilinx FPGAs* [Online]. Available: <http://www.intersil.com/data/ag/ag0001.pdf>
- [43] National Semiconductor Solutions Guide. (2010, March). *Analog for Altera FPGAs Solutions Guide* [Online]. Available: <http://www.national.com/appinfo/power/files/NationalAlteraDesignGuide.pdf>

- [44] Texas Instruments Selection Guide. (2010, Jan.). *Power Management Guide* [Online]. Available: <http://focus.ti.com/lit/sg/slv145j/slv145j.pdf>
- [45] National Semiconductor Design Guide. (2005, March). *Power Management Design Guide for Altera FPGAs and CPLDs* [Online]. Available: <http://www.national.com/appinfo/power/files/NationalAlteraDesignGuide.pdf>
- [46] Intel Design Guide. (2009, Sept.). *Voltage Regulator-Down (VRD) 11.1: Processor Power Delivery Design Guidelines* [Online]. Available: <http://www.intel.com/Assets/pdf/designguide/322172.pdf>
- [47] A. M. Wu, J. Xiao, D. Markovic, and S. R. Sanders, "Digital PWM control: application in voltage regulation modules", in *IEEE Power Electronics Specialists Conference*, 1999, pp. 77-83 vol.1.
- [48] K. Ogata, *Discrete-Time Control Systems*: Prentice Hall, 1995.
- [49] L. Ka and D. Alfano, "Design and implementation of a practical digital PWM controller", in *IEEE Applied Power Electronics Conference and Exposition*, 2006, pp. 1437-1442.
- [50] Texas Instruments Data Sheet. (2008, Nov.). *UCD9240 Digital PWM System Controller* [Online]. Available: <http://focus.ti.com/lit/ds/symlink/ucd9240.pdf>
- [51] Z. Zhao and A. Prodic, "Limit-Cycle Oscillations Based Auto-Tuning System for Digitally Controlled DC-DC Power Supplies", *IEEE Transactions on Power Electronics*, vol. 22, no. 6, pp. 2211-2222, 2007.
- [52] Z. Lukic, Z. Zhao, S. M. Ahsanuzzaman, and A. Prodic, "Self-tuning digital current estimator for low-power switching converters", in *IEEE Applied Power Electronics Conference and Exposition*, 2008, pp. 529-534.
- [53] L. Corradini, P. Mattavelli, W. Stefanutti, and S. Saggini, "Simplified Model Reference-Based Autotuning for Digitally Controlled SMPS", *IEEE Transactions on Power Electronics*, vol. 23, no. 4, pp. 1956-1963, 2008.
- [54] A. Kelly and K. Rinne, "A self-compensating adaptive digital regulator for switching converters based on linear prediction", in *IEEE Applied Power Electronics Conference and Exposition*, 2006, pp. 712-718.
- [55] F. Guang, E. Meyer, and Y. Liu, "A New Digital Control Algorithm to Achieve Optimal Dynamic Performance in DC-to-DC Converters", *Power Electronics, IEEE Transactions on*, vol. 22, no. 4, pp. 1489-1498, 2007.
- [56] S. Pan and P. K. Jain, "Analysis of a high performance voltage regulator with non-linear multi-mode control: Bandwidth and large transient response", in *IEEE Applied Power Electronics Conference and Exposition*, 2010, pp. 499-506.

- [57] A. Costabeber, L. Corradini, P. Mattavelli, and S. Saggini, "Time optimal, parameters-insensitive digital controller for DC-DC buck converters", in *IEEE Power Electronics Specialists Conference*, 2008, pp. 1243-1249.
- [58] V. Yousefzadeh, A. Babazadeh, B. Ramachandran, L. Pao, D. Maksimovic, and E. Alarcon, "Proximate Time-Optimal Digital Control for DC-DC Converters", in *IEEE Power Electronics Specialists Conference*, 2007, pp. 124-130.
- [59] L. Wanhammar, *DSP Integrated Circuits*: Academic Press, 1999.
- [60] K.K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*: John Wiley & Sons, 1999.
- [61] Y. T. Lin, Y. C. Wang, and Y. Y. Tzou, "Single-Chip FPGA Implementation of a Digital VRM Controller with Interlaced Sampling and Control Technique", in *IEEE Power Electronics Specialists Conference*, 2007, pp. 1441-1447.
- [62] M. Y. K. Chui, W. H. Ki, and C.Y. Tsui, "A programmable integrated digital controller for switching converters with dual-band switching and complex pole-zero compensation", *IEEE Journal of Solid-State Circuits*, vol. 40, no. 3, pp. 772-780, 2005.
- [63] Exar Data Sheet. (2009, Sept.). *XRP7740 Quad-Output Digital PWM Buck Controller* [Online]. Available: <http://www.exar.com/Common/Content/Document.ashx?id=20057&LanguageId=1033>
- [64] CHiL Semiconductor Preliminary Product Brief. (2008, Oct.). *CHL8314 Digital Multi-Phase Buck Controller* [Online]. Available: <http://www.chilsemi.com/wp-content/uploads/chl8314-product-brief.pdf>
- [65] Intersil Data Sheet. (2010, Dec.). *ZL6100 Adaptive Digital DC/DC Controller with Drivers and Current Sharing* [Online]. Available: <http://www.intersil.com/data/fn/fn6876.pdf>
- [66] A. Prodic and D. Maksimovic, "Design of a digital PID regulator based on look-up tables for control of high-frequency DC-DC converters", in *IEEE Workshop on Computers in Power Electronics*, 2002, pp. 18-22.
- [67] W. Stefanutti, P. Mattavelli, S. Saggini, and M. Ghioni, "Autotuning of Digitally Controlled DC-DC Converters Based on Relay Feedback", *IEEE Transactions on Power Electronics*, vol. 22, no. 1, pp. 199-207, 2007.
- [68] J. Morroni, R. Zane, and D. Maksimovic, "Design and Implementation of an Adaptive Tuning System Based on Desired Phase Margin for Digitally Controlled DC-DC Converters", *IEEE Transactions on Power Electronics*, vol. 24, no. 2, pp. 559-564, 2009.

- [69] Z. Zhao and A. Prodic, "Continuous-Time Digital Controller for High-Frequency DC-DC Converters", *IEEE Transactions on Power Electronics*, vol. 23, no. 2, pp. 564-573, 2008.
- [70] E. O'Malley and K. Rinne, "A 16-bit fixed-point digital signal processor for digital power converter control", in *IEEE Applied Power Electronics Conference and Exposition*, 2005, pp. 50-56.
- [71] A. R. Oliva, S. S. Ang, and G. E. Bortolotto, "Digital control of a voltage-mode synchronous buck converter", *IEEE Transactions on Power Electronics*, vol. 21, no. 1, pp. 157-163, 2006.
- [72] S. Bibian and H. Jin, "High performance predictive dead-beat digital controller for DC power supplies", *IEEE Transactions on Power Electronics*, vol. 17, no. 3, pp. 420-427, 2002.
- [73] A. Kelly, "A Self Compensating Adaptive Digital Regulator", PhD Thesis, University of Limerick, 2006.
- [74] S. Saggini, W. Stefanutti, E. Tedeschi, and P. Mattavelli, "Digital Deadbeat Control Tuning for dc-dc Converters Using Error Correlation", *IEEE Transactions on Power Electronics*, vol. 22, no. 4, pp. 1566-1570, 2007.
- [75] B. Miao, A. Bakker, M. Soldano, and B. Daly, " $\Sigma\Delta$ ADC based current mode power supply controller with digital voltage loop", in *IEEE Applied Power Electronics Conference and Exposition*, 2008, pp. 1582-1588.
- [76] S. Saggini, M. Ghioni, and A. Geraci, "An innovative digital control architecture for low-voltage, high-current DC-DC converters with tight voltage regulation", *IEEE Transactions on Power Electronics*, vol. 19, no. 1, pp. 210-218, 2004.
- [77] D. Figoli. (2006, Aug.). *Software Design for Digital Power - Programming 101 for Analog Designers* [Texas Instruments Training Course] [Online]. Available: <http://focus.ti.com/download/trng/docs/seminar/Topic6DF.pdf>
- [78] E. O'Malley, "System-on-chip architecture and building blocks for future digital control of switching power converters", PhD Thesis, University of Limerick, 2006.
- [79] Powervation Ltd. (2010). *Digital Power Interface Center (DPIC)* [Online]. Available: <http://www.powervation.com/products/graphical-user-interface-gui>
- [80] J. A. Abu-Qahouq, O. Abdel-Rahman, L. Huang, and I. Batarseh, "On Load Adaptive Control of Voltage Regulators for Power Managed Loads: Control Schemes to Improve Converter Efficiency and Performance", *IEEE Transactions on Power Electronics*, vol. 22, no. 5, pp. 1806-1819, 2007.

- [81] J. Mooney, A. E. Mahdi, A. Kelly, and K. Rinne, “Dual MAC processor for adaptive control of multiple high switching frequency DC-DC converters”, in *IET Irish Signals and Systems Conference*, 2008, pp. 116-121 (Best Paper Award Winner).
- [82] J. Mooney, A. E. Mahdi, A. Kelly, and K. Rinne, “DSP-based controller for multi-output/multi-phase high switching frequency DC-DC converters”, in *IEEE Workshop on Control and Modeling for Power Electronics*, 2008, pp. 1-6.
- [83] Texas Instruments Reference Guide. (2010, July). *TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide* [Online]. Available: <http://focus.ti.com/lit/ug/spru732j/spru732j.pdf>
- [84] Analog Devices Reference Guide. (2010, June). *ADSP-BF50x Blackfin Processor Hardware Reference* [Online]. Available: http://www.analog.com/static/imported-files/processor_manuals/ADSP-BF50x_hwr_rev.0.2.pdf
- [85] J. A. Fisher, P. Faraboschi, and C. Young, *Embedded Computing: A VLIW Approach to Architecture, Compilers and Tools*: Morgan Kaufmann, 2005.
- [86] L. J. Karam, I. AlKamal, A. Gatherer, G. A. Frantz, D. V. Anderson, and B. L. Evans, “Trends in multicore DSP platforms”, *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 38-49, Nov. 2009.
- [87] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 4th ed.: Morgan Kaufmann, 2007.
- [88] O. Garcia, A. de Castro, P. Zumelis, and J. A. Cobos, “Digital-Control-Based Solution to the Effect of Nonidealities of the Inductors in Multiphase Converters”, *IEEE Transactions on Power Electronics*, vol. 22, no. 6, pp. 2155-2163, 2007.
- [89] A. Kelly, “Current Share in Multiphase DC-DC Converters Using Digital Filtering Techniques”, *IEEE Transactions on Power Electronics*, vol. 24, no. 1, pp. 212-220, 2009.
- [90] D. Liu, *Embedded DSP Processor Design: Application Specific Instruction Set Processors*: Morgan Kaufmann, 2008.
- [91] Berkeley Design Technology Inc. White Paper. (2002, March). *Evaluating DSP Processor Performance* [Online]. Available: http://www.bdti.com/articles/benchmk_2000.pdf
- [92] E.C. Ifeachor and B.W. Jervis, *Digital Signal Processing: A Practical Approach*: Pearson Education Limited, 2002.

- [93] J. Mooney, M. Halton, and A. E. Mahdi, "Specialized Digital Signal Processor for control of multi-rail/multi-phase high switching frequency power converters", in *IEEE Applied Power Electronics Conference and Exposition*, 2010, pp. 2207-2211.
- [94] R. Rajsuman, *System-on-a-Chip: Design and Test*: Artech House, 2000.
- [95] ARM Specification. (2010, Apr.). *AMBA APB Protocol Version: 2.0 Specification* [Online]. Available: www.arm.com
- [96] System Management Interface Forum Inc. (2010, Sept.). *PMBus Power System Management Protocol Specification* [Online]. Available: www.pmbus.org
- [97] P. Lapsley, J. Bier, A. Shoham, and E.A. Lee, *DSP Processor Fundamentals: Architectures and Features*: Wiley-IEEE Press, 1997.
- [98] R. S. H. Istepanian and J. F. Whidborne, *Digital Controller Implementation and Fragility: A Modern Perspective*: Springer, 2001.
- [99] A. Hartstein and T. R. Puzak, "The optimum pipeline depth for a microprocessor", in *IEEE International Symposium on Computer Architecture* 2002, pp. 7-13.
- [100] J. P. Deschamps, G. J. A. Bioul, and G. D. Sutter, *Synthesis of Arithmetic Circuits: FPGA, ASIC and Embedded Systems*: John Wiley & Sons, 2006.
- [101] Plexim User Manual. (2011, Mar.). *Piece-wise Linear Electrical Circuit Simulation: User Manual Version 3.1* [Online]. Available: <http://www.plexim.com/files/plecsmanual.pdf>
- [102] Chroma ATE Data Sheet. (2009, Feb.). *High Slew Rate DC Electronic Load Model 6345/6346* [Online]. Available: <http://www.chromaate.com/chroma/product/172/file/314/6340-E-09.pdf>
- [103] L. Carlson and L. Richardson, *Ruby Cookbook*: O'Reilly Media, Inc., 2006.
- [104] H. Hong, "Coordination of design issues in the intermediate bus architecture", in *IEEE Applied Power Electronics Conference and Exposition*, 2005, pp. 169-175.
- [105] J. Mooney, S. Effler, M. Halton, and A. E. Mahdi, "Interrupt controller for DSP-based control of multi-rail DC-DC converters with non-integer switching frequency ratio", in *IEEE International Conference on Electronics, Circuits and Systems*, 2010, pp. 1211-1214.
- [106] A. Emadi, A. Khaligh, Z. Nie, and Y. J. Lee, *Integrated Power Electronic Converters and Digital Control*: CRC Press, 2009.

- [107] D. Maksimovic and R. Zane, "Small-Signal Discrete-Time Modeling of Digitally Controlled PWM Converters", *IEEE Transactions on Power Electronics*, vol. 22, no. 6, pp. 2552-2556, 2007.
- [108] Y. Sun, F. C. Lee, and J. Li, "Modeling of digitally controlled voltage regulator modules", in *IEEE Applied Power Electronics Conference and Exposition*, 2010, pp. 176-182.
- [109] L. Corradini, S. Saggini, and P. Mattavelli, "Analysis of a high-bandwidth event-based digital controller for DC-DC converters", in *IEEE Power Electronics Specialists Conference*, 2008, pp. 4578-4584.
- [110] J. Mooney, S. Effler, M. Halton, and A. E. Mahdi, "DSP-based control of multi-rail DC-DC converter systems with non-integer switching frequency ratios", in *1st International Conference on Energy, Power and Control, EPC-IQ'01*, 2010, pp. 200-204.
- [111] J. L. Cruz, A. Gonzalez, M. Valero, and N. P. Topham, "Multiple-banked register file architectures", in *IEEE International Symposium on Computer Architecture*, 2000, pp. 316-325.
- [112] S. Effler, A. Kelly, M. Halton, T. Kruger, and K. Rinne, "Digital control law using a novel load current estimator principle for improved transient response", in *Power Electronics Specialists Conference*, 2008, pp. 4585-4589.
- [113] R. V. White, "Digital Power After The Hype", in *Applied Power Electronics Conference and Exposition*, 2010.

Appendix A. Instruction Set

This appendix lists the instructions in the instruction set of the dual MAC DSP core. The instructions are grouped into categories according to their function. A general description of the function of the instructions is provided in addition to details of operand requirements and result destinations.

Instruction	Instruction Description	Operand Constraints	Number of Operands	Destination of Result
Add				
ADD	Adds operand1 and operand2	Both operands are register file registers or ACC1 or ACC2	2	ACC on relevant datapath
SUB	Subtracts operand2 from operand1	Both operands are register file registers or ACC1 or ACC2	2	ACC on relevant datapath
ADDSAT	Adds operand1 and operand2 and saturates result	Both operands are register file registers or ACC1 or ACC2	2	ACC on relevant datapath
SUBSAT	Subtracts operand2 from operand1 and saturates result	Both operands are register file registers or ACC1 or ACC2	2	ACC on relevant datapath
Multiply				
MUL	Multiplies operand1 and operand2	Both operands are register file registers or ACC1 or ACC2	2	ACC on relevant datapath
MULRND	Multiplies operand1 and operand2 and rounds result to 16 bits	Both operands are register file registers or ACC1 or ACC2	2	ACC on relevant datapath
MAC				
MAC	Multiplies operand1 and operand2 and adds result to relevant ACC	Both operands are register file registers or ACC1 or ACC2	2	ACC on relevant datapath
MACSAT	Multiplies operand1 and operand2, adds result to relevant ACC and saturates final result	Both operands are register file registers or ACC1 or ACC2	2	ACC on relevant datapath
MACU	Multiplies operand1 and operand2, adds result to relevant ACC and moves data in operand1 register to next register in register file	Both operands are register file registers or ACC1 or ACC2	2	ACC on relevant datapath

Instruction	Instruction Description	Operand Constraints	Number of Operands	Destination of Result
Round				
RNDSAT	Rounds up ACC to 16 bit value and saturates result	Operand1 is ACC1 or ACC2	1	ACC on relevant datapath
Logic				
AND	Bitwise logical AND of operand1 and operand2	Both operands are register file registers or ACC1 or ACC2	2	ACC on relevant datapath
OR	Bitwise logical OR of operand1 and operand2	Both operands are register file registers or ACC1 or ACC2	2	ACC on relevant datapath
XOR	Bitwise logical XOR of operand1 and operand2	Both operands are register file registers or ACC1 or ACC2	2	ACC on relevant datapath
NOT	Bitwise logical NOT of operand1	Operand1 is register file register or ACC1 or ACC2	1	ACC on relevant datapath
Other				
SHIFTSAT	Shifts operand1 by given shift factor and saturates result, performing sign extension if necessary	Operand1 is a register file register or ACC1 or ACC2, operand2 is a 5-bit shift factor where the MSB indicates the direction of the shift (0 - left, 1 - right)	2	ACC on relevant datapath
SAT	Saturates operand1 if overflow has occurred	Operand1 is ACC1 or ACC2 (if a register file register is used, its value is moved to the relevant ACC)	1	ACC on relevant datapath
MOV	Moves data from source register to destination register	Both operands are register file registers or ACC1 or ACC2, operand1 = source, operand2 = destination	2	Destination register
CMP	Compares data in source register to data in limit register and moves value of least magnitude back to source register	Both operands are register file registers or ACC1 or ACC2	1	Source register
Program Flow				
NOP	No operation i.e. idle	N/A	0	N/A
JMP	Loads program counter with value in jmp_adr_regE	N/A	0	N/A
RETI	Loads program counter with value in jmp_adr_regBG	N/A	0	N/A

Instruction	Instruction Description	Operand Constraints	Number of Operands	Destination of Result
Load				
LOADP	Moves data from memory location given by ptr_reg to destination register	Operand1 is a register file register or ACC1 or ACC2	1	Destination register
LOADPI	Moves data from memory location given by ptr_reg to destination register and increments ptr_reg by 1	Operand1 is a register file register or ACC1 or ACC2	1	Destination register
LOADPD	Moves data from memory location given by ptr_reg to destination register and decrements ptr_reg by 1	Operand1 is a register file register or ACC1 or ACC2	1	Destination register
Store				
STORES	Moves data from source register to memory location given by seg_reg + given offset	Operand1 is a register file register or ACC1 or ACC2, operand2 is a 5-bit memory offset	2	Memory
STOREP	Moves data from source register to memory location given by ptr_reg	Operand1 is a register file register or ACC1 or ACC2	1	Memory
STOREPI	Moves data from source register to memory location given by ptr_reg and increments ptr_reg by 1	Operand1 is a register file register or ACC1 or ACC2	1	Memory
STOREPD	Moves data from source register to memory location given by ptr_reg and decrements ptr_reg by 1	Operand1 is a register file register or ACC1 or ACC2	1	Memory
STORECS	Moves data from source register to memory location given by seg_reg + given offset	Operand1 is a configuration register, operand 2 is a 5-bit memory offset	2	Memory
STORECP	Moves data from source register to memory location given by ptr_reg	Operand1 is a configuration register	1	Memory
STORECPI	Moves data from source register to memory location given by ptr_reg and increments ptr_reg by 1	Operand1 is a configuration register	1	Memory
STORECPD	Moves data from source register to memory location given by ptr_reg and decrements ptr_reg by 1	Operand1 is a configuration register	1	Memory

Instruction	Instruction Description	Operand Constraints	Number of Operands	Destination of Result
Double-Length Load/Store				
LOADI	Loads immediate data given in instruction to destination register	Operand1 is a 16-bit data value, operand 2 is a register file register not or ACC1 or ACC2	2	Destination register
LOADCI	Loads immediate data given in instruction to destination register	Operand1 is a 16-bit data value, operand 2 is a configuration register	2	Destination register
LOADD	Moves data from memory location given by immediate address in instruction to destination register	Operand1 is a 16-bit memory address, operand2 is a register file register or ACC1 or ACC2	2	Memory
STORED	Moves data from source register to memory location given by immediate address in instruction	Operand1 is a 16-bit memory address, operand2 is a register file register or ACC1 or ACC2	2	Memory
STORECD	Moves data from source register to memory location given by immediate address in instruction	Operand1 is a 16-bit memory address, operand2 is a configuration register	2	Memory

Appendix B. Assembly Code

The assembly code used to program the dual MAC processor in validating its performance is presented below. The code includes instructions for three three-pole, three-zero control algorithms for the regulation of three independent buck converters.

```
; Initialisation Code

; Load data for control algorithm 1
LOADCI  0          RBANK
LOADI   0.8691_q12 R0
LOADI  -1.5756_q12 R1
LOADI   0.7198_q12 R2
LOADI   0          R3
LOADI   0.4476_q14 R4
LOADI   0.2760_q14 R5
LOADI   0.2764_q14 R6
LOADI   0          R7
LOADI   0          R8
LOADI   0          R9
LOADI   0          R10
LOADI   0          R11
LOADI   0          R12
LOADI   0          R13
LOADI   1.0_q12    R14
LOADI   0.75_q12   R15
LOADI   16         R16
LOADI   17         R17
LOADI   18         R18
LOADI   19         R19
LOADI   20         R20
LOADI   21         R21
LOADI   22         R22
```

```
LOADI 23      R23
LOADI 24      R24
LOADI 25      R25
LOADI 26      R26
LOADI 27      R27
LOADI 28      R28
LOADI 29      R29
LOADI 30      R30
LOADI 31      R31
```

```
; Load data for control algorithm 2
```

```
LOADCI 1      RBANK
LOADI 0.8691_q12 R0
LOADI -1.5756_q12 R1
LOADI 0.7198_q12 R2
LOADI 0        R3
LOADI 0.4476_q14 R4
LOADI 0.2760_q14 R5
LOADI 0.2764_q14 R6
LOADI 0        R7
LOADI 0        R8
LOADI 0        R9
LOADI 0        R10
LOADI 0        R11
LOADI 0        R12
LOADI 0        R13
LOADI 1.0_q12  R14
LOADI 0.75_q12 R15
LOADI 16      R16
LOADI 17      R17
LOADI 18      R18
LOADI 19      R19
LOADI 20      R20
LOADI 21      R21
LOADI 22      R22
```

```
LOADI 23          R23
LOADI 24          R24
LOADI 25          R25
LOADI 26          R26
LOADI 27          R27
LOADI 28          R28
LOADI 29          R29
LOADI 30          R30
LOADI 31          R31
```

```
; Load data for control algorithm 3
```

```
LOADCI 2          RBANK
LOADI 0.8691_q12  R0
LOADI -1.5756_q12 R1
LOADI 0.7198_q12  R2
LOADI 0           R3
LOADI 0.4476_q14  R4
LOADI 0.2760_q14  R5
LOADI 0.2764_q14  R6
LOADI 0           R7
LOADI 0           R8
LOADI 0           R9
LOADI 0           R10
LOADI 0           R11
LOADI 0           R12
LOADI 0           R13
LOADI 1.0_q12     R14
LOADI 0.75_q12    R15
LOADI 16          R16
LOADI 17          R17
LOADI 18          R18
LOADI 19          R19
LOADI 20          R20
LOADI 21          R21
LOADI 22          R22
```

```
LOADI 23      R23
LOADI 24      R24
LOADI 25      R25
LOADI 26      R26
LOADI 27      R27
LOADI 28      R28
LOADI 29      R29
LOADI 30      R30
LOADI 31      R31
```

```
; Load configuration registers
```

```
LOADCI 3      RBANK
LOADCI 32767  RPLIM
LOADCI 0      RPLIMM
LOADCI 15     RINTCNT1
LOADCI 15     RINTCNT2
LOADCI 15     RINTCNT3
LOADCI _BKGND RJMPE
LOADCI _BKGND RJMPBG
LOADCI _INT1  RJMP1
LOADCI _INT2  RJMP2
LOADCI _INT3  RJMP3
```

```
; Enable all interrupts
```

```
LOADCI 1      INTEN
```

```
; Background code - random operations
```

```
_BKGND
XOR  R31  R30
NOP
MOVR ACC1H R31
JMP
```

```
; Interrupt Service Routines

; 3p3z interrupt for Rail 1
_INT1

; Load R7 with ADC value
LOADP R7

; NOP, Subtract ADC value from reference
NOP                ADD R7 R15

; b0 * e(n) + ACC1, Load R8 with error
MAC ACC2H R0      MOVR ACC2H R8

; Convert to Q12 by left-shifting by 4
SHIFTSAT ACC1H -4

; Round ACC1
RNDSAT ACC1H

; Output duty cycle to PWM, Load d(n-1) register with d(n)
STOREP ACC1H      MOVR ACC1H R11

; NOP, a3 * d(n-3)
NOP                MUL R13 R6

; b3 * e(n-3), a2 * d(n-2) + ACC2
MUL R10 R3        MACU R12 R5

; b2 * e(n-2), a1 * d(n-1) + ACC2
MACU R9 R2        MACU R11 R4

; b1 * e(n-1) + ACC1, Convert to Q12 by right-shifting by 2
MACU R8 R1        SHIFTSAT ACC2H 2
```

```
; Sum b's and a's
ADD  ACC1H  ACC2H

; Return from interrupt
RETI

; 3p3z interrupt for Rail 2
_INT2

; Load R7 with ADC value
LOADP  R7

; NOP, Subtract ADC value from reference
NOP                                ADD  R7  R15

; b0 * e(n) + ACC1, Load R8 with error
MAC  ACC2H  R0                    MOVR  ACC2H  R8

; Convert to Q12 by left-shifting by 4
SHIFTSAT  ACC1H  -4

; Round ACC1
RNDSAT  ACC1H

; Output duty cycle to PWM, Load d(n-1) register with d(n)
STORES  ACC1H  14                    MOVR  ACC1H  R11

; NOP, a3 * d(n-3)
NOP                                MUL  R13  R6

; b3 * e(n-3), a2 * d(n-2) + ACC2
MUL  R10  R3                    MACU  R12  R5

; b2 * e(n-2), a1 * d(n-1) + ACC2
MACU  R9  R2                    MACU  R11  R4
```

```
; b1 * e(n-1) + ACC1, Convert to Q12 by right-shifting by 2
MACU R8 R1          SHIFTSAT ACC2H 2

; Sum b's and a's
ADD ACC1H ACC2H

; Return from interrupt
RETI

; 3p3z interrupt for Rail3
_INT3

; Load R7 with ADC value
LOADP R7

; NOP, Subtract ADC value from reference
NOP          ADD R7 R15

; b0 * e(n) + ACC1, Load R8 with error
MAC ACC2H R0          MOVR ACC2H R8

; Convert to Q12 by left-shifting by 4
SHIFTSAT ACC1H -4

; Round ACC1
RNDSAT ACC1H

; Output duty cycle to PWM, Load d(n-1) register with d(n)
STORES ACC1H 15      MOVR ACC1H R11

; NOP, a3 * d(n-3)
NOP          MUL R13 R6
```

```
; b3 * e(n-3), a2 * d(n-2) + ACC2
MUL  R10  R3                MACU  R12  R5

; b2 * e(n-2), a1 * d(n-1) + ACC2
MACU  R9   R2                MACU  R11  R4

; b1 * e(n-1) + ACC1, Convert to Q12 by right-shifting by 2
MACU  R8   R1                SHIFTSAT ACC2H  2
; Sum b's and a's
ADD  ACC1H  ACC2H

; Return from interrupt
RETI
```