DEVELOPMENT OF A SOFT-CORE BASED POWER ELECTRONIC CONVERSION CONTROLLER

Thesis submitted in partial fulfilment of the requirements for the degree Master of Technology: Electrical Engineering in the Faculty of Engineering

at the

CAPE PENINSULA UNIVERSITY OF TECHNOLOGY

by

Cassandra Daviane Nsumbu

Supervisor: Mr Daan de Beer

Co-supervisor: Mr Andrew Van der Byl

Cape Town Campus

June 2014

DECLARATION

I, Cassandra Daviane Nsumbu, declare that the contents of this dissertation/thesis represent my own unaided work, and that the thesis has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.

Signed	**Date**

ABSTRACT

The application of digital control techniques has become dominant in power electronics owing to several advantages they present, when compared to analogue solutions. Their development is based on the use of microprocessors and microcontrollers, such as Application Specific Integrated Circuit (ASIC), Digital signal processors (DSP), Field Programmable Gate Arrays (FPGA), or a combination of these devices.

This thesis presents an investigation of a soft-core based FPGA control system as a solution for power electronic applications. The aim was the development and implementation of a conversion controller, which purpose is to supply control inputs in the form of digital Pulse Width Modulation (PWM) signals, to a number of power electronic applications, such as single half and full bridge DC-DC converters, three phase and multicell inverters. The PWM control technique is achieved via their power semiconductor switching devices. These PWM control signals are necessary for the high frequency conversion of an analog input voltage (AC, DC or unregulated) to an analog output voltage of another level (AC or DC). This was intended to be achieved by exploiting and combining the advantages that FPGA and embedded processors provide such as high reconfigurability and multipurpose ability. This controller's digital outputs, namely PWM switching signals, can be directly delivered to an analog signal amplification circuit to create an adequate voltage level before being processed by the converters' switches.

The full design consists of the Nios® II processor, which executes the control algorithm in the form of software codes; a Hardware Descriptive Language (HDL) peripheral that forms the Pulse Width Modulator (PWM), which is necessary in the operation of power electronic applications in open-loop control mode; and MATLAB® Graphical User Interfaces (GUI) to serve as communication platforms between the controller user and the soft-core. The proposed soft-core is the Nios® II processor from Altera® Corporation, which acts as the central method of implementation in this work. The soft-core is integrated along with several other modules, and is configured onto an FPGA to produce a reconfigurable and expandable hardware/software solution in the form of a System-On-Programmable-Chip (SOPC).

Software is developed in C and implemented to control the interaction and operation of the SOPC hardware components via the Nios® II processor. It also ensures data transfer between MATLAB® GUIs and the processor. Changes can be easily implemented in the software in order to ensure the control of the hardware aspect of the soft-core based conversion controller (SCBC).

The design software, Quartus® II and its development tool, DSP Builder are used in this work, for the creation of the HDL firmware and SOPC system. Another development tool from Quartus® II, the Nios® II Integrated Development Environment (IDE), is used to create and debug the software code, which is intended to program the Nios® II soft-processor. MATLAB was used to serve to develop the GUIs, which serve as a user communication interface between a host PC and the soft-core based embedded system.

Simulation, synthesis and experimental test results are presented and discussed in this research study to evaluate the operation of the processor core and its overall SOPC system. Simulation and synthesis tests were conducted at each stage of the controller's development process in order to check and monitor the progress in both hardware and software development. They also allowed for a comparison with the expected experimental test results. The latter was performed on the overall Nios® II system, while using the MATLAB® GUIs, which were designed to enter user inputs. The controller hardware's outputs, Pulse Width Modulation (PWM) control signals, were measured at various instances following different inputs, accordingly, to evaluate the controller's performance. These signals are responsible for the control and operation of power electronic applications in real hardware. The conclusion drawn from these results is that this method of implementation was successful in its intended objectives. Further research should address the application of the closed-loop control mode, and a more independent method of communication between the user inputs into the controller and the Nios® II processor.

ACKNOWLEDGEMENTS

I wish to thank:

- Our Almighty God;
- Prof Robert van Zyl, Daniel de Beer, Dr Wilkinson, Mr Van der Byl, Prof Biermann and Mr Kaplan for their enthusiasm, guidance and knowledge;
- My family, especially my mother, for their constant support;
- The CIR staff; and
- My colleagues for their support during the development phase.

I am very grateful for the financial support from the National Research Foundation (NRF) and the French South African Institute of Technology (F'SATI).

Finally I extend my gratitude to all who has indirectly contributed to the successful completion of this work.

LIST OF ABBREVIATIONS

AC	Alternating Current
ASIC	Application Specific Integrated Circuit
CAD	Computer-Aided Design
CIR	Centre of Instrumentation Research
CLK	Clock
CPU	Central Processing Unit
DAC	Digital-to-analog
DC	Direct Current
DDS	Direct Digital Synthesis
DPWM	Digital Pulse Width Modulation
DSP	Digital Signal Processing
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
F'SATI	French South African Institute of Technology
GUI	Graphical User Interface
HDL	Hardware Descriptive Language
HSMC	High Speed Mezzanine Card

IC	Integrated Circuit
IDE	Integrated Development Environment
IGBT	Insulated Gate Bipolar Transistor
I/O	Input/Output
IP	Intellectual Property
IRQ	Interrupt Request
JTAG	Join Test Action Group
LED	Light Emitting Diode
LE	Logic Element
LHD	Low Harmonic Distortion
LUT	Lookup Table
MM	Memory Mapped
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MSPS	Million samples per second
PCB	Printed Circuit Board
PE	Power Electronics
PIO	Parallel Input/Output
PLL	Phase Locked Loop
SCBC	Soft-Core Based power conversion Controller
SOC	System On Chip
SRAM	Static Random Access Memory
ST	Streaming Interface

RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
RTL	Register Transfer Level
UART	Universal Asynchronous Receiver/Transmitter
UPS	Uninterruptible Power Supply
USB	Universal Serial Bus
VHDL	VHSIC Hardware Descriptive Language
VHSIC	Very High Speed Integrated Circuit

TABLE OF CONTENTS

DECLARATION I

ABSTRACT II

ACKNOWLEDGEMENTS IV

LIST OF ABBREVIATIONS V

LIST OF FIGURES XII

CHAPTER 1 1

INTRODUCTION 1

1.1 Project background 1

1.2 Motivation 6

1.3 Objectives 6

1.4 Research questions 6

1.5 Research methodology 7

1.6 Delineation of the research 7

1.7 Contributions of the research 8

1.8 Thesis outline 8

1.9 Summary 9

CHAPTER 2 10

BACKGROUND OF DIGITAL CONTROL FOR POWER ELECTRONIC APPLICATIONS . 10

2.1 Introduction 10

2.2 Open-loop control mode 10

2.3 Introduction to Pulse Width Modulation (PWM) 11
2.3.1 PWM techniques 11
2.3.2 Deadtime 13

2.4 Overview of FPGA based digital controllers 14
2.4.1 DSP/ FPGA based digital controller 14
2.4.2 FPGA based digital controller 15
2.4.3 FPGA embedded soft-core processors 15

2.5 Nios® II Soft Processor core 17

2.6 Contributions of the Nios® II soft-core application in power electronics 20

2.7 Related and past work on digital control for power electronics 20

2.8 Summary 23

CHAPTER 3 24

HARDWARE AND SOFTWARE TOOLS 24

3.1 Introduction 24

3.2 Cyclone® III EP3C25 Starter Development board 24

3.3 Controller software components 27
3.3.1 DSP Builder and MATLAB/Simulink 28
3.3.2 Altera® Quartus II 29
3.3.3 SOPC builder 29
3.3.4 Nios® II IDE 30

3.4 Software languages 30
3.4.1 VHDL 30
3.4.2 C language 31

3.5 Summary 31

CHAPTER 4 32

HARDWARE/SOFTWARE CO- DESIGN AND IMPLEMENTATION 32

4.1 Introduction 32

4.2 Soft-core based controller specifications 33

4.3 Controller overview **34**
4.3.1 Hardware design overview 35
4.3.2 Software design overview 36

4.4 Project design flow **36**
4.4.1 Stage 1 – DSP Builder design and implementation 37
4.4.2 Stage 2 – SOPC creation 56
4.4.3 Stage 3 – Quartus® II system completion 60
4.4.4 Stage 4 - MATLAB® GUI design and implementation 61
4.4.5 Stage 5 – Software code development 65

4.5 Summary **72**

CHAPTER 5 .... **73**

SIMULATION AND EXPERIMENTAL RESULTS **73**

5.1 Introduction **73**

5.2 Nios® II and DSP Builder simulation **74**

5.3 Experimental testing **76**

5.4 Switching and reference signal frequency control results **77**

5.5 Application control results **90**
- Five-cell inverter 90
- Three phase inverters 92
- Half bridge converter 95
- Full bridge converter 96

5.6 Deadtime control results **97**

5.7 PWM duty cycle control **99**
- 15 % duty cycle 100
- 25 % duty cycle 101
- 50 % duty cycle 102
- 75 % duty cycle 103
- 90 % duty cycle 104

5.8 Summary **105**

CHAPTER 6 .... **106**

CONCLUSIONS AND FUTURE WORK **106**

6.1 Introduction **106**

6.2 Conclusions **106**

6.4 Encountered problems **108**

6.5 Future work **109**

LIST OF REFERENCES **110**

APPENDIX A – SOPC BUILDER CONFIGURATION SETTINGS **119**

LIST OF FIGURES

Figure 1.1: Power electronics and interdisciplinary concepts (Mohan, 2003) 2
Figure 1.2: Structure of a power electronics switching converter (Erickson & Maksimovic, 2004) 4
Figure 1.3: Open-loop controller block diagram 11

Figure 2.1: Sinusoidal PWM generation (Klumpner, 2004) 13
Figure 2.2: DC based PWM generation 13
Figure 2.3: Illustration of dead time (Venkatesh, 1994) 14

Figure 3.1: FPGA architecture (Zeidman, 2009) 25
Figure 3.2: Top View of the Cyclone® III FPGA Starter Board from (Altera Corporation, 2010b) 26

Figure 4.18: RTL diagram of the soft-core based controller's Nios II system 61

Figure 5.1: Testing setup block diagram 73
Figure 5.3: Sine samples validity verification process block diagram 75
Figure 5.4: Activation of application and signal generation control GUI 78
Figure 5.5: DSP Builder simulation for a switching frequency of 400 kHz and a 25 kHz reference signal 79
Figure 5.6: Experimental result of a switching frequency of 400 kHz and a 25 kHz reference signal 79
Figure 5.7: Measured FFT of a single 400 kHz PWM gating signal displaying the 25 kHz reference frequency 80
Figure 5.8: DSP Builder simulation of interleaved switching for a switching frequency of 100 kHz and a 6.25 kHz reference signal 81
Figure 5.9: Measured 100 kHz PWM signal 81
Figure 5.10: Measured FFT of a single 100 kHz PWM gating signal displaying 6.25 kHz reference frequency 82
Figure 5.11: DSP Builder simulation of interleaved switching for a switching frequency of 200 kHz and a 6.25 kHz reference signal 83

Figure 5.12: Simulation GUI for Half bridge converter 83
Figure 5.13: Simulation GUI zoom screenshots of a) interleaved switching and b)the ed PWM signals 84
Figure 5.14: DSP Builder simulation of interleaved switching for a switching frequency of 200 kHz and a 6.25 kHz reference signal 85
Figure 5.15: Three phase inverter - interleaved switching and PWM signals for a switching frequency of 200 kHz and reference signal of 50 kHz. 86
Figure 5.16: Measured 200 kHz high –side PWM signal 86
Figure 5.17: Measured FFT of a single 200 kHz PWM gating signal displaying 50 kHz reference frequency 87
Figure 5.18: Measured 1.6 kHz high –side PWM signal 88
Figure 5.19: Measured FFT of a single 1.6 kHz PWM gating signal displaying 100 Hz reference frequency. 88
Figure 5.20: Measured 833 Hz high –side PWM signal 89
Figure 5.21: Measured FFT of a single 1.6 kHz PWM gating signal displaying 100 Hz reference frequency. 89
Figure 5.22: 5-cell multilevel inverter PWM signals (simulation – DSP Builder) 91
Figure 5.23: 5-cell multilevel inverter PWM signals (hardware level) 91
Figure 5.24: Three phase inverter PWM signals (simulation – DSP Builder) 92
Figure 5.25: Three phase inverter inverter PWM signals (simulation - MATLAB) 93
Figure 5.26: Enlarged screenshot of Three phase inverter inverter PWM signals (simulation - MATLAB) 93
Figure 5.27: Three High –side PWM signals running at 833 Hz (hardware level) 94
Figure 5.28: Half bridge converter PWM signals (simulation – DSP Builder) 95
Figure 5.29: Half bridge converter PWM signals (hardware level) 95
Figure 5.30: Full bridge converter PWM signals (simulation – DSP Builder) 96
Figure 5.31: Full bridge converter PWM signals (hardware level) 96
Figure 5.32: Deadtime control GUI 97
Figure 5.33: 40 ns deadtime implementation results, in simulation and hardware level 98
Figure 5.34: 100 ns deadtime implementation results, in simulation and hardware level 98
Figure 5.35: 200 ns deadtime implementation results, in simulation and hardware level 98
Figure 5.36: PWM duty cycle control GUI 99
Figure 5.37: 15 % duty cycle (simulation) 100
Figure 5.38: 15 % duty cycle (hardware level) 100
Figure 5.39: 25 % duty cycle (simulation) 101
Figure 5 40: 25 % duty cycle (hardware level) 101
Figure 5.41: 50 % duty cycle (simulation) 102
Figure 5.42: 50 % duty cycle (hardware level) 102
Figure 5.43: 75 % duty cycle (simulation) 103
Figure 5.44: 75 % duty cycle (hardware level) 103
Figure 5.45: 90 % duty cycle (simulation) 104

Figure 5.46: 90 % duty cycle (hardware level)...104

CHAPTER 1

INTRODUCTION

Over the last two decades there has been a rapid growth of technological advances in power electronics, microelectronics, improvements of control algorithms, and demand for new applications. Hence, a generic digital controller is necessary as an educational contribution and basis for further research on digital controllers in the Centre of Instrumentation (CIR), where a high number of power electronics projects are developed. This controller should be efficient by providing as much reconfigurability as possible without any or little change in the developed hardware.

This thesis presents the research work, development tools, design, implementation and simulation and experimental results of a solution in the form of a Nios® II processor based controller. This controller is intended to be operational on several power electronic applications such as half bridge, full bridge, three phase, as well as multicell converters. The purpose of this work is to acquire an understanding of a FPGA embedded processor, and its application and contributions in power electronics. The chapter details the project's background, motivation, objectives, research questions, methodology, delineation and, finally, the thesis' structure.

1.1 Project background

Power electronics is a field of electrical engineering, which involves electrical energy processing and control through circuits that are termed switching converters. Switching converters are found in applications, which range from milliWatts to GigaWatts (Erickson & Maksimovic, 2004). Power electronics deals with the interdisciplinary concepts from the same field, as shown in Figure 1.1 below.

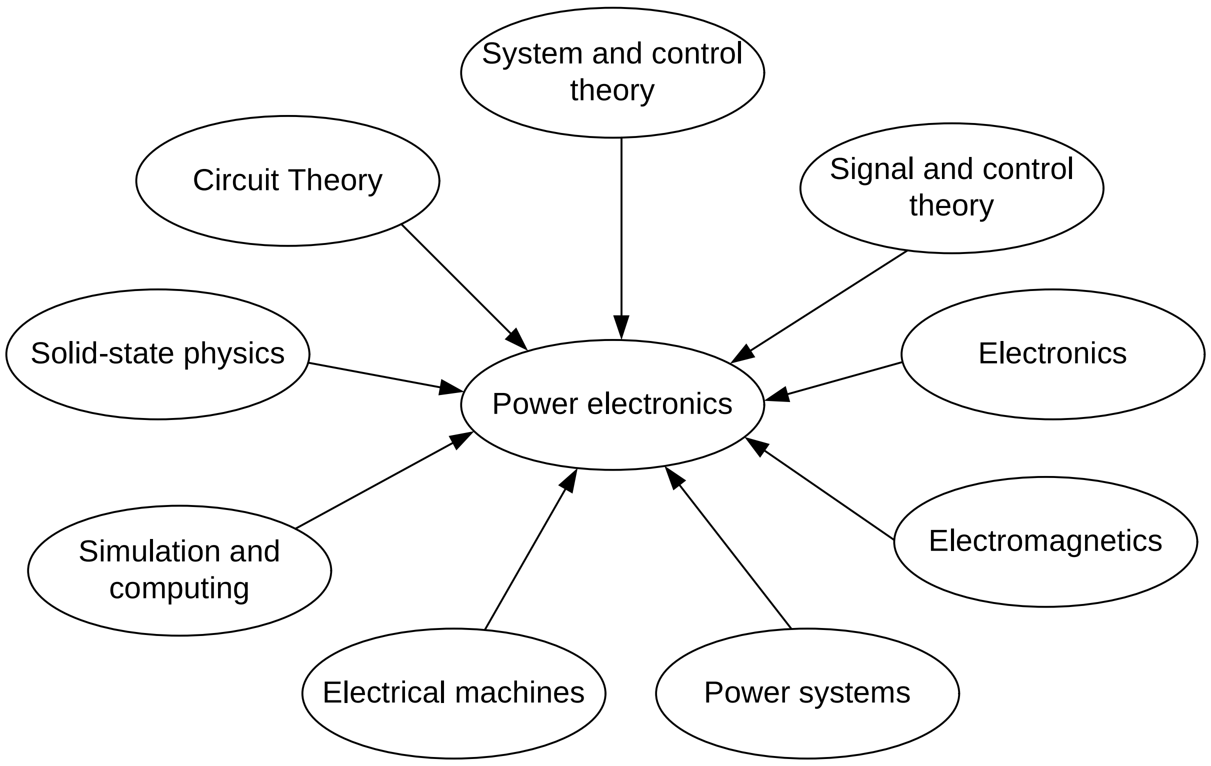


Figure 1.1: Power electronics and interdisciplinary concepts (Mohan, 2003)

Switching converters are the power heart of any power electronic system, as they convert an input source signal (AC or DC) into an output signal (AC or DC) with a voltage of a different magnitude, current or frequency. These converters are classified into four groups according to the type of the input and output signal that they deliver, as listed in Table 1.1.

Figure 1.2 is an illustration of a basic PE switching converter, which consists of an input, control and output signal ports. In PE applications, control and monitoring systems such as digital controllers are required to ensure production of the required regulated and conditioned power output signal from converters.

Switching converter	Function	Applications used in:
AC to DC (rectifier)	Conversion of an AC input signal into a DC output signal	High voltage DC transmission systems, regulated DC power supplies, DC motor drives, electronic devices connected to the mains such as computers and TV.
DC to AC (inverter)	Conversion of a DC input signal into an AC output signal of controllable magnitude and frequency	Uninterruptible Power Supplies (UPS), emergency lighting systems, aircraft and space power supplies, induction heating supplies and induction and synchronous motor drives.
DC to DC	Conversion of a DC input signal into an DC output signal of different magnitude or polarity	Buck, boost, flyback and buck/boost converters, battery chargers, switch mode power supplies, mobile devices, distributed power systems, electrical isolation and power factor correction.
AC to AC	Conversion of an AC input signal into an AC output signal of different magnitude or frequency	International power adaption, power distribution networks, cycloconverters and light dimmers

Table 1.1: Different types of switching converters, their functions and applications (Mohan, 2003; Singh, 2008)

They make up a vital part of any power processing system (Erickson & Maksimovic; Mohan *et al.*, 2003). Power regulation is achieved through the control of transitional on-off states of semiconductor switching devices (Maksimovic & Erickson, 2004). This control process is known as Pulse Width Modulation (PWM), which is a widely used control technique in power electronic systems.

Converters use power semiconductor devices such as diodes, thyristors (SCR), power transistors such as Bipolar Junction Transistors (BJT), Metal Oxide Silicon Field Effect Transistor (MOSFET), Insulated Gate Bipolar Transistors (IGBT), Gate Turn-Off thyristors (GTO) and Insulated Gate Controlled Thyristors (IGCT). All these elements operate within a switching state, with the exception of the diode (Luo, 2005).

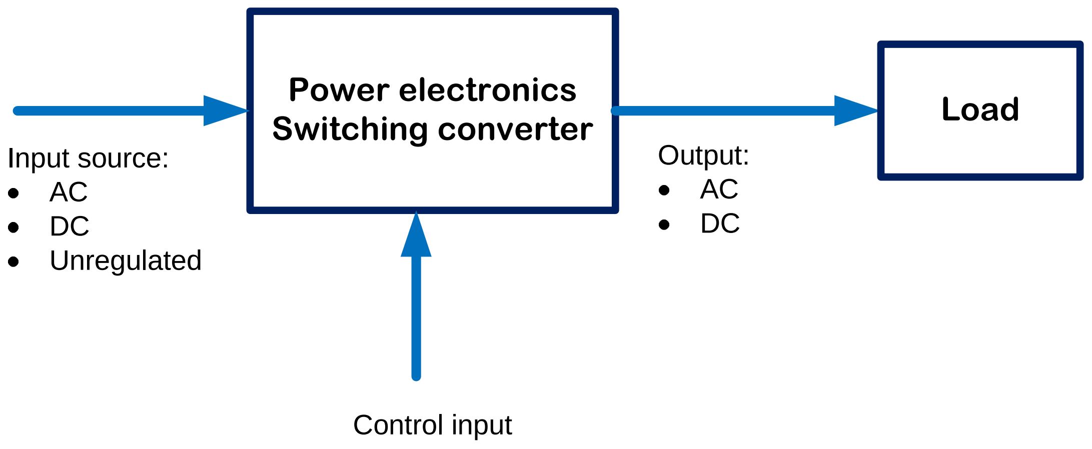


Figure 1.2: Structure of a power electronics switching converter (Erickson & Maksimovic, 2004)

Figure 1.3 illustrates the inclusion of the proposed SCBC block and its role in the control of a switching power converter. In this work, Altera®'s Nios® II soft-processor is the heart of the controller, as it provides control of the entire system and its peripherals. The controller delivers the control input in the form of digital Pulse Width Modulation (PWM) signals to switching converters via the semiconductor switches that they consist of. These PWM signals are necessary for the conversion of an analog input voltage (AC, DC or unregulated) to an analog output voltage of another level (AC or DC).

The growing replacement of analogue control techniques by digital counterparts is promoted by the advantages the latter provide in the area of power electronics. Digital controllers offer configurability, greater noise and ageing immunity, greater resistance to environmental interference, ease of other digital technology integration (Martin *et al.*, 1995), and flexibility Kariyappa & Kumari, 2009).
Most digital controllers that are developed today have been implemented by using Digital Signal Processors (DSPs), Field-Programmable Gate Arrays (FPGAs), a combination of both (Prodic *et al.*, 2001; de Castro *et al.*, 2003), Application-Specific (ASIC), microcontrollers or microprocessors. FPGAs involve flexible technology that has become extremely popular as a platform of choice in digital control. This is owing to the various advantages that it presents, which include higher density, higher performance, higher speed, flexibility and reasonable cost. FPGAs increasingly integrate embedded resources such as hardware multipliers, processors and RAM blocks on a single integrated circuit (Quasim *et al.*, 2010).

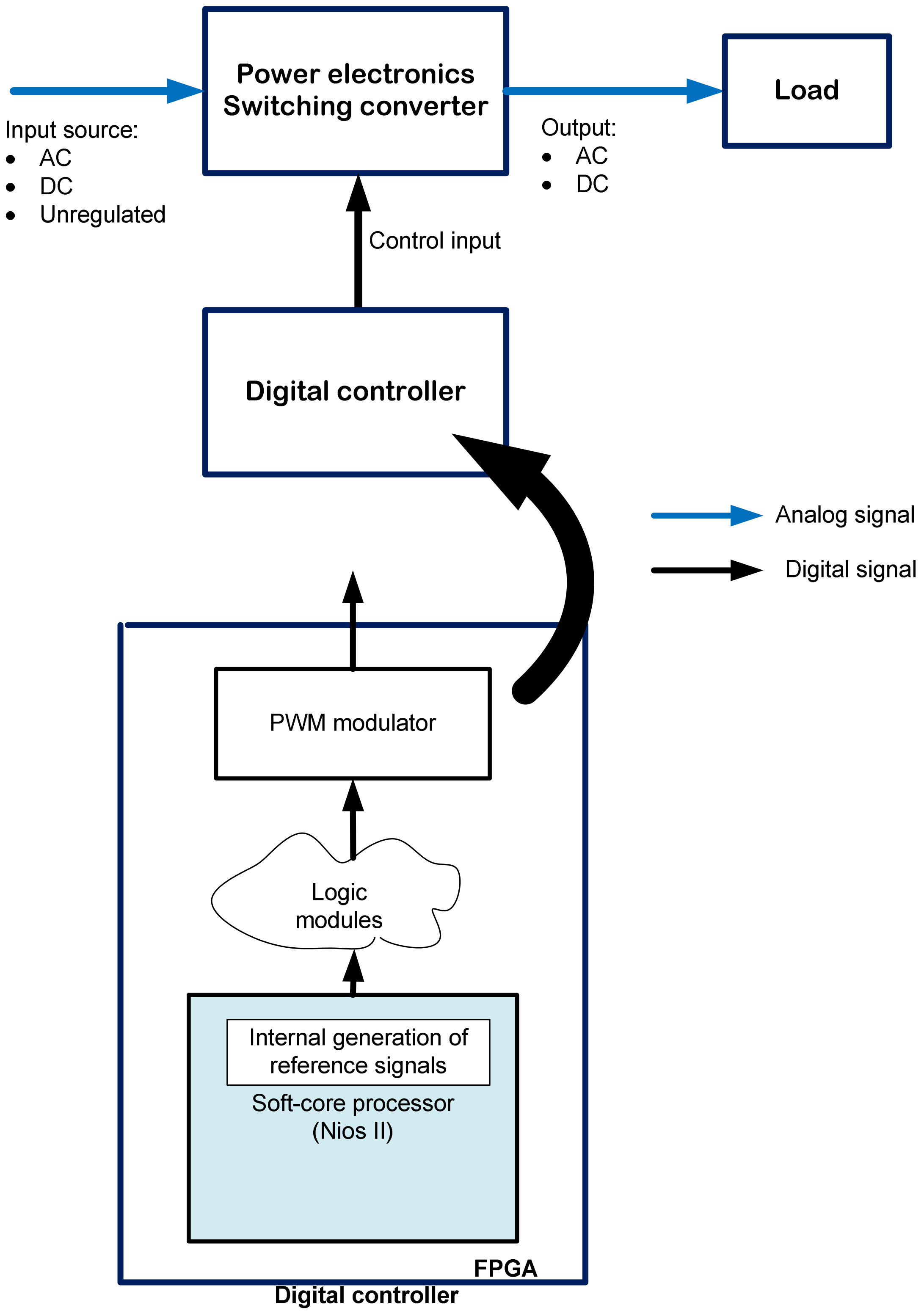


Figure 1.3: Block diagram of the soft-core based controller, modified from (Erickson & Maksimovic, 2004)

FPGAs, along with configurable processor cores integrated into FPGA technology have become an appealing option in producing a digital controller (Nurmi, 2007; Chamberlain et al., 2005). This combination provides a high level of configurability of hardware peripherals in the production of System-On-Programmable-Chip (SOPC) systems (Urriza *et al*, 2009).

1.2 Motivation

There is a need for a generic power electronic controller at the Centre of Instrumentation research (CIR) owing to an increase in power electronic projects. Even though a DSP-FPGA based controller was recently developed (Jooste & Wilkinson, 2009), the need for a more integrated and reconfigurable controller arose, where only a single Integrated Circuit (IC) would be required. This is realizable, as a processor would be embedded in a FPGA chip to produce a complete SOPC system. Centre for Instrumentation Research (CIR) students and researchers can benefit from the advantages that SOPCs provide by applying them in the development of control system solutions in power electronics.

1.3 Objectives

The aims of this study are:

- To investigate embedded system based controllers and FPGA/soft-core technology;
- To design and implement a digital controller, which incorporates a Nios® II processor, both in hardware and software aspects. The controller should be able to work on the above - mentioned PE; and
- To use Pulse Width Modulation (PWM) as the control technique, with the internal generation of carrier signals and reference signals with adjustable frequencies as inputs from the Nios® II processor.

1.4 Research questions

The following research questions have been formulated in respect of the present study:

- Is it feasible or possible to use a soft-core processor as the main core of a generic power electronic controller?

- Is it possible to define a soft-core-based controller for power electronic applications?
- Can it work on a variety of power electronic applications?

1.5 Research methodology

A literature investigation of the implementation of the Nios® II processor, SOPC design and the use of the Very High Speed Integrated Circuit Hardware Description language (VHDL), was conducted. Power converters and their operation and topologies were also investigated. The hardware aspect of the work was developed in VHDL through Altera®'s development tools, DSP Builder and Quartus® II. The software was developed and written in C by using the Nios® II Integrated Development Environment (IDE). MATLAB® was used to create Graphical User Interfaces (GUI) to develop a communication platform for user inputs into the controller's Nios® II system. Several measurement tests were performed to ensure the feasibility of the entire hardware/software co-design.

1.6 Delineation of the research

The proposed work consists of the development and implementation of a conversion controller for power electronic applications based on a Nios® II soft-core, which is implemented in FPGA and co-processing logic for PWM control. This study does not focus on the theory or practical nature of digital control, but rather on the rapid development of an efficient and fast controller for power electronics by using low-cost FPGA-based development tools. A Nios® II-based controller is designed to work on PE applications such as:

- Single half bridge DC-DC converters and inverters;
- Single phase full bridge DC-DC converters and inverters;
- Three phase inverters; and
- Multicell inverters.

The physical construction and implementation of these individual applications were not considered. The research only focused on the open-loop control mode for design simplicity and easy implementation, and did not consider the closed-loop control mode. Both simulation and experimental tests were performed for comparison and performance monitoring purposes. The experimental tests were based on the final product's outputs, which are PWM signals.

1.7 Contributions of the research

An interest in a combination of methods, in this case, a soft-core processor and the FPGA arose at the CIR in an attempt to provide a new, easy and feasible hardware/software solution for the control of power electronic applications.

The use of a single chip to construct an entire and expandable PE control system limits the cost of the work. The reconfigurability offered by this work is beneficial, as the FPGA chip can be re-programmed to adjust to different applications' specifications whenever it is required. The amount of flexibility that is gained through SOPCs is also extremely advantageous. Since software is required to run on the processor, late modifications and simplified debugging can be implemented whenever necessary during the design process without a need for hardware changes. Thus, the controller's development process becomes less expensive. More costs are reduced as the system development cycle is considerably reduced, and this increases the efficiency of the FPGA chip. Flexibility in the selection of a specific set of CPUs, peripherals and interfaces helps to accelerate only the necessary functions that are involved in the operation of the SOPC system, and to prevent processor obsolescence (Alcade, 2009).

This Nios® II processor-based controller provides a better understanding on embedded soft-core processors, SOPCs and their implementation methods, as well as a solid foundation for further research and practice for power electronics at the CIR.

1.8 Thesis outline

Chapter 2 presents a literature review on Pulse Width Modulation (PWM), several controller topologies and contributions of soft-core processors within power electronics.

Chapter 3 describes different hardware and software tools, which are involved in the development of the soft-core based controller. The Cyclone® 3C25 FPGA development board, software tools such as Quartus® II, DSP Builder, MATLAB®, the Nios® II IDE and software languages, are all presented in this chapter.

Chapter 4 elaborates on details of the hardware and software co-design. It presents and describes all the stages that are involved in the development and completion of the entire controller system. The chapter begins with project specifications, followed by brief overviews on hardware/software co-design, DSP Builder firmware, implementation of the Nios® II processor and its peripherals in a SOPC system, and ends with implementation stages of the operating controller system.

Chapter 5 discusses various test results that were obtained from the research study, and analyses from the project from DSP Builder/Simulink, Nios® II IDE (Integrated Development Environment) and oscilloscope, and digital analyser measurement readings.

Chapter 6 summarizes the findings of the thesis, as well as the advantages of the method that was used in the study. The chapter also recommends future research in terms of further development of the study.

1.9 Summary

This chapter dealt with the project's background, motivation, objectives, research questions, research methodology, delineation, contributions and, finally, the thesis outline.

CHAPTER 2

BACKGROUND OF DIGITAL CONTROL FOR POWER ELECTRONIC APPLICATIONS

2.1 Introduction

The purpose of this chapter is to provide a review of the open-loop control mode, Pulse Width Modulation (PWM), different embedded systems based controller topologies and the Nios® II processor. It also elaborates on the importance and contributions of work that have been done in power electronics control during the last decade.

2.2 Open-loop control mode

The digital systems mentioned above act as system controllers while dealing with dynamical systems behaviour. There are two types of digital control: open-loop and closed-loop control modes. The closed-loop control mode is beyond the scope of this work, hence only the open-loop control mode will be examined. Open-loop control is commonly applied in converters' industrial applications as the main control mode in digital control (Luo, 2005). It is mostly useful for well-defined and simple systems where the relationship between input and output can be modelled by a mathematical formula, and output values are predictable. This control system is controlled directly and only by its input (Mastascusa, 2002). The proposed controller is a good representation of this control scheme, and computes its input into the system by using only the current state and the system's model. The advantages that the open-loop control presents and, which makes it more beneficial to apply are simplicity, ease in construction of the system's layout and stability owing to the system's simplicity (Speaking Technology, 2012). Figure 1.3 displays a basic block diagram of the open loop control mode, where the system block represents the switching converter, and the controlled variables represent the regulated power output signal. Input signals are fed into the control system to generate the required output signals. In this work, the input source signal is processed, as specified by the control input, namely PWM signals, which are generated by the soft-core based controller (SCBC), as shown in Figure 1.4. This yields conditioned output power, which is required (Erikson & Maksimovic, 2004).

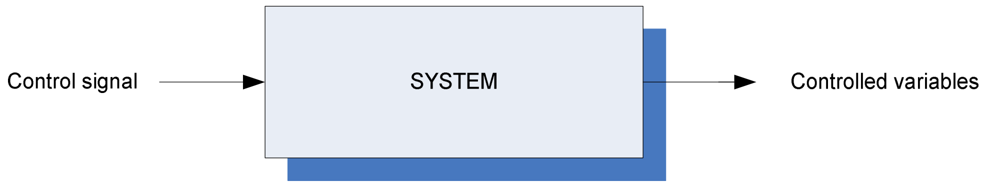


Figure 1.3: Open-loop controller block diagram

2.3 Introduction to Pulse Width Modulation (PWM)

As mentioned in Chapter One, PWM is the most widely used control technique in power electronic systems. It performs output voltage regulation by producing a wave of pulses as an output. These pulses have the same voltage level, but might have different widths, depending on the technique that is used to produce the wave. These widths are responsible for output frequencies and voltages control. The important application of PWM lies in power electronics applications as means to control power converters through their electronic power switches (Koutroulis *et al.*, 2006). PWM control is achieved by controlling power switches' gate voltages and currents. These power switches operate in complementary pairs, and each has one PWM signal to activate them. The main advantage that the PWM method offers is the reduction of systems costs, reduced component space usage, reduced total harmonic distortion of load current (Bakar, 2009), increased noise immunity, high performance and efficiency, high energy saving and power consumption reduction (Ababu, 2007).

2.3.1 PWM techniques

Different PWM schemes and techniques are employed to provide variable voltage, frequency and power to applications with the adequate efficiency, and the necessary performance for the application in use. The average value of these parameters is controlled by turning the power switches on and off at a chosen pace.

This work employs two different carrier-based PWM techniques to control the SCBC system outputs. These digital techniques involve a comparator, a triangular (carrier) signal and a reference sinusoidal or DC signal. A counter is developed to produce the triangular wave by counting from zero to a specific value. Then the counter counts from this maximum value back to zero. The produced triangular wave is then compared to the sinusoidal or DC signal. Each time the two signals intersect, the PWM signal goes from a 'high' to a 'low', or from a 'low' to a 'high'. This results in the formation of duty-cycles. The maximum value of the counter determines the resolution, the period time of the PWM signal (Bengtsson & Jonsson, 2009) and the PWM switching frequency. It also contributes towards determining the maximum value of the duty cycle, just as the maximum amplitude of the reference signal does when both signals are compared. The word duty cycle describes the ratio of 'on' time over 'off' time. The power and voltage, which is delivered to the loads is proportional to the duty cycle. The duty cycle formula is shown below:

$$\text{Duty cycle} = \frac{\text{T on}}{\text{T signal}} \tag{2.1}$$

where: T on is the ON signal period

T signal is the signal period.

The sinusoidal PWM technique involves the carrier and a sinusoidal signal for inverters. This technique results in the generation of a string of pulses of varying duty cycles. This technique promotes harmonic content reduction, and makes it favourable for power electronics inverters, which take pulses with varying duty cycles as inputs and generate AC voltage outputs (Venkatesh, 1994). The sinusoidal PWM technique is illustrated in Figure 2.1 below.

In the case of DC-DC converters, PWM is produced as a string of pulses with a fixed duty cycle value like a D.C component, and with a specific amplitude which is compared to the carrier signal. This technique is shown in Figure 2.2. Since the DC signal's amplitude is directly proportional to the PWM duty cycle, its amplitude can range between the minimum and maximum amplitude of the carrier wave, according to the required duty cycle.

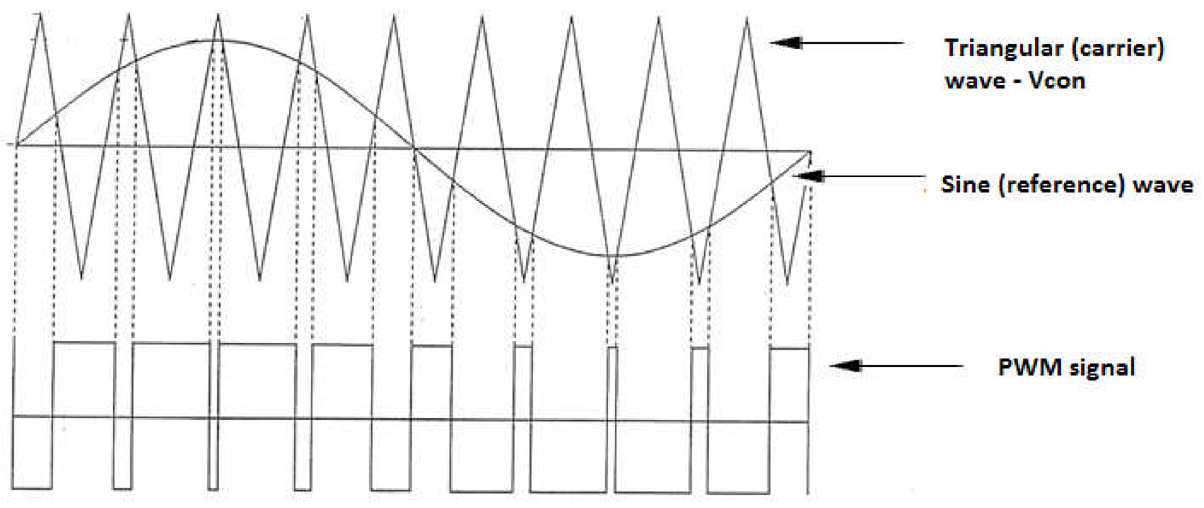


Figure 2.1: Sinusoidal PWM generation (Klumpner, 2004)

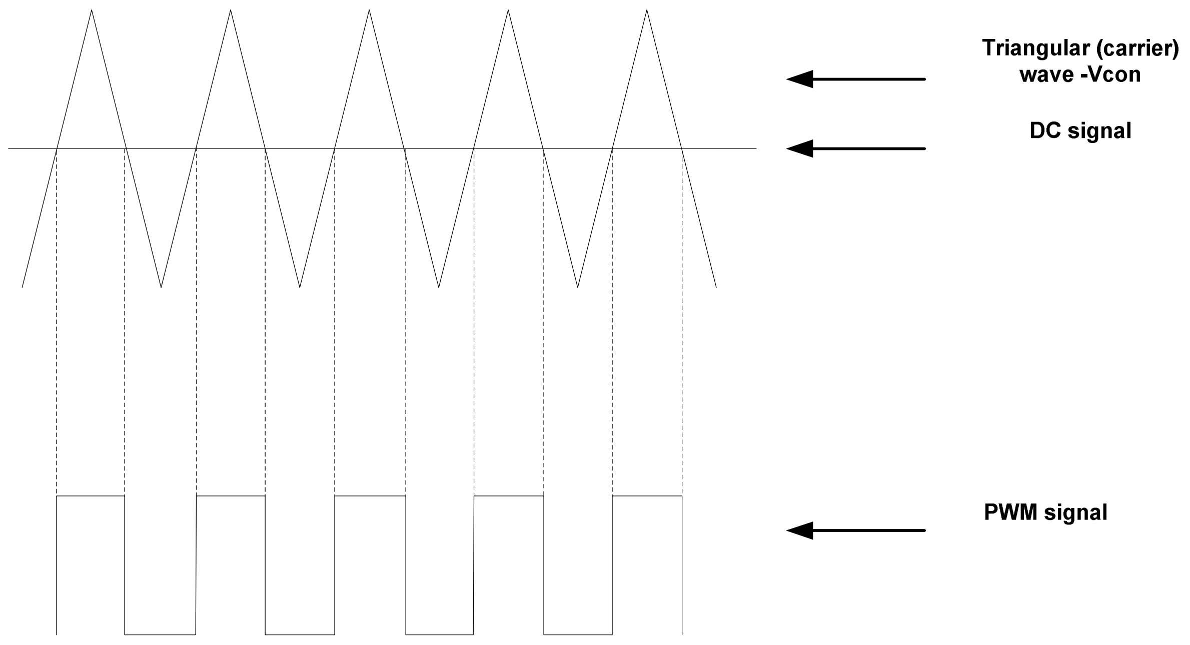


Figure 2.2: DC based PWM generation

2.3.2 Deadtime

During switches control via PWM, switches from a complementary pair are supplied with one inverted and a non-inverted PWM signal. The dead time should be inserted between each intersection between the carrier and reference signals in the digital controller system. This is implemented to prevent unfavourable implications in the form of short circuits and system failure, caused by switching pairs that are on and off at the same. It should be ensured in order to prevent. Hence, dead time should be inserted in the digital controller system. The dead time is inserted between each intersection between the carrier and reference signal. The implementation of dead time on the low side switch is displayed in Figure 2.3. The high-side and low-side switches correspond to the two switches belonging to a complementary pair.

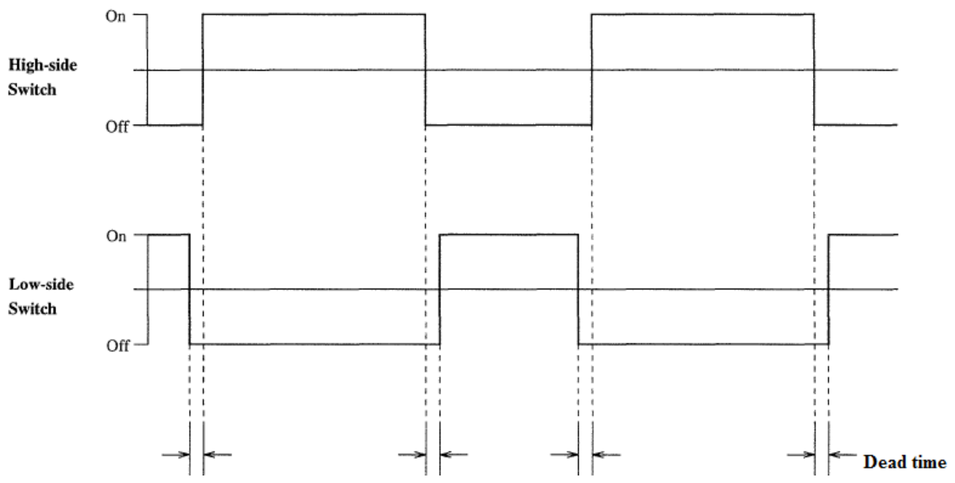


Figure 2.3: Illustration of deadtime (Venkatesh, 1994)

2.4 Overview of FPGA based digital controllers

2.4.1 DSP/ FPGA based digital controller

Recently, DSPs are commonly used in the control of power electronics. In digital signal processing applications such as wireless signals, DSPs are the best option. They perform complex digital signal processing and math-intensive tasks more efficiently, with more functionality, much faster and at a lower cost when compared to microprocessors, ASICs and FPGA hardware (Daya, 2009). These characteristics render DSPs suitable for power converters control, but their application for high speed switching is less common, where massive parallel algorithms are required (Zhang, 2006).

The DSP/FPGA-based controller provides more advantages than the single DSP based controller. The division task implemented by this solution provides faster control speed, extra system flexibility and easier software design. However, this structure has disadvantages of increasing the controller's complexity and costs. It also makes measurement testing more difficult owing to its complicated structure. Another approach comprises implementing control algorithms in one single FPGA chip (Zhang, 2006).

2.4.2 FPGA based digital controller

A FPGA based digital controller has more advantages compared to single DSP and DSP/FPGA controllers (Zhang, 2006). FPGAs are part of flexible technology that has become extremely popular as a platform of choice in digital control. This is owing to the various advantages that it presents, which include higher density, higher performance, higher speed, flexibility and reasonable costs. They can integrate embedded resources such as hardware multipliers, processors and RAM blocks on a single integrated circuit (Quasim *et al.*, 2010). They are configured by using a hardware description language (HDL) such as Verilog or VHDL or a schematic design diagram. HDL codes offer a better alternative when working large systems designs, while schematic entry is more suitable for smaller designs, which can be drawn piece by piece.

This provides direct hardware implementation with the same flexibility that software alternatives provide. VHDL codes can also be easily reused and synthesized into any FPGA platform. This makes FPGAs significantly more advantageous than DSP devices, microcontrollers and ASICs in terms of reconfigurability (Zhang, 2006; Leonov, 2009). Digital system designs such as a PWM system for power electronics are known to be easily implemented on FPGAs. Designs are implemented on FPGA in a way that they can be easily modified to address problems, or to add or remove new functions and features, if necessary. This flexibility amounts to ease of changing interconnections between logic blocks. FPGA designs can provide design complexity and time spent on design and implementation (Daya, 2009; Zhang, 2006)). Reconfiguration is then easily performed by using the JTAG interface.

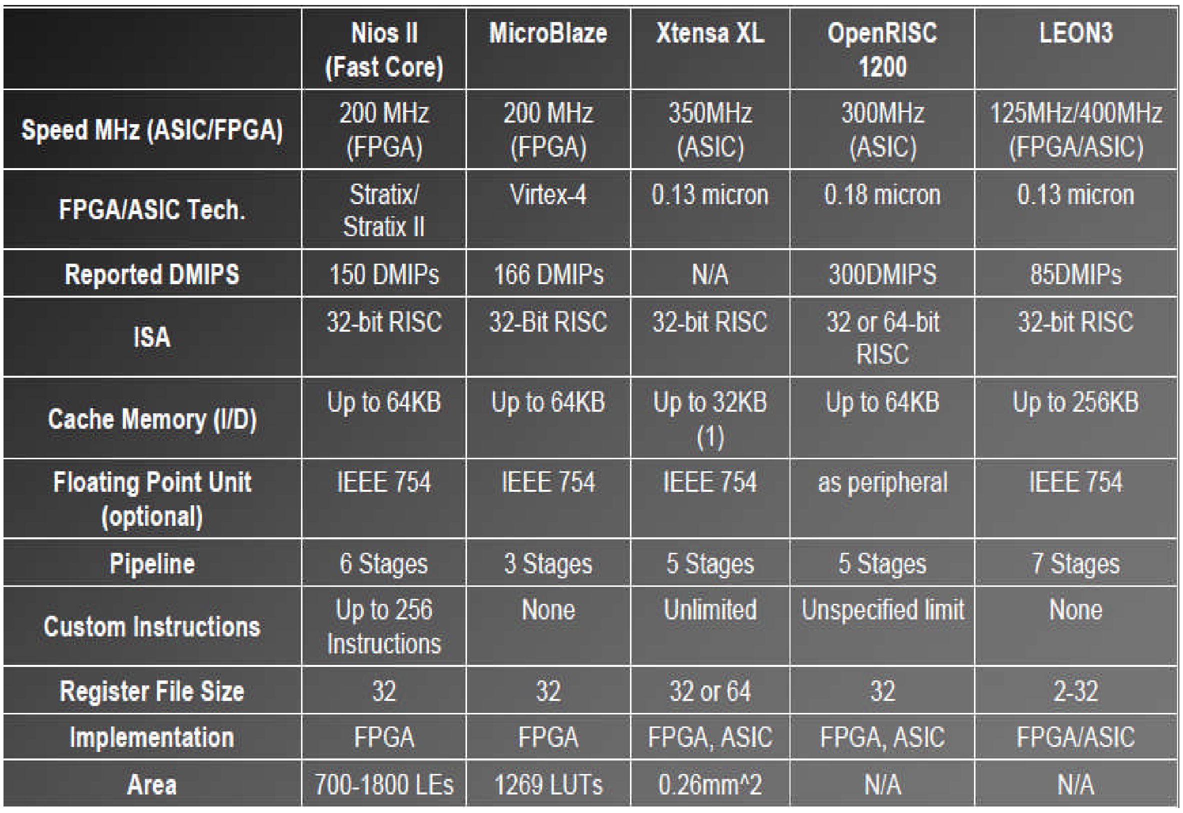
2.4.3 FPGA embedded soft-core processors

FPGAs, along with embedded microprocessors, have become a powerful option as means to produce a digital controller (Nurmi, 2007; Chamberlain *et al.*, 2005). This combination allows the development and control of a configurable System-On-Programmable-Chip (SOPC) system, by using a soft-core processor and a set of hardware peripherals (Urriza *et al.*, 2009).

A soft-core processor is a processor integrated into FPGA architecture. It is implemented in a hardware description language to become customized, by configuring their architecture and behaviour, for given applications. A FPGA embedded soft-core processor and custom peripherals, which should meet the exact design requirements, are described by using an HDL (which is flexible if an open source), synthesized and mapped to an FPGA, which is a reconfigurable device (Tong *et al.*, 2006). The entire design can be mapped to different FPGA platforms. The use of a soft-core on an FPGA holds several advantages, which are outlined below:

- A soft-core can be removed from the design when it is not needed. This saves a lot of area, which can be used for other elements.
- It can be easily customized for a specific application owing to its reconfigurability properties.
- A soft-core is feature-rich and allows peripherals to be added or subtracted from the SOPC system with ease (Tong *et al.*, 2006).
- Open source soft-cores can have their source code taken and modified by the designer to meet the needs of the applications (Anemaet & van As, 2008).
- It offers more flexibility and parallelism.

The main advantage that this type of processor provides digital control systems is the important amount of flexibility, as the existing reconfigurable logic elements from the FPGA are used to implement the processor (Arbinger & Erdmann, 2006). It can be implemented in any FPGA, as long as there are enough configurable logic resources available (Maxwell, 2004). Soft-core processors are provided by FPGA manufacturers to operate on their own FPGAs and have them mapped onto the chip. They are provided by open source communities and commercial vendors. The most widely used commercial soft-cores are Nios® II, MicroBlaze®, PicoBlaze® and Xtensa from Altera®, Xilinx® and Tensilica®, respectively. Some of the most widely used open source soft-cores are OpenRISC® 1200 and LEON 3®. Table 2.1 below clarifies the differences between some soft-cores in terms of features and characteristics.



	Nios II (Fast Core)	MicroBlaze	Xtensa XL	OpenRISC 1200	LEON3
Speed MHz (ASIC/FPGA)	200 MHz (FPGA)	200 MHz (FPGA)	350MHz (ASIC)	300MHz (ASIC)	125MHz/400MHz (FPGA/ASIC)
FPGA/ASIC Tech.	Stratix/ Stratix II	Virtex-4	0.13 micron	0.18 micron	0.13 micron
Reported DMIPS	150 DMIPs	166 DMIPs	N/A	300DMIPS	85DMIPs
ISA	32-bit RISC	32-Bit RISC	32-bit RISC	32 or 64-bit RISC	32-bit RISC
Cache Memory (I/D)	Up to 64KB	Up to 64KB	Up to 32KB (1)	Up to 64KB	Up to 256KB
Floating Point Unit (optional)	IEEE 754	IEEE 754	IEEE 754	as peripheral	IEEE 754
Pipeline	6 Stages	3 Stages	5 Stages	5 Stages	7 Stages
Custom Instructions	Up to 256 Instructions	None	Unlimited	Unspecified limit	None
Register File Size	32	32	32 or 64	32	2-32
Implementation	FPGA	FPGA	FPGA, ASIC	FPGA, ASIC	FPGA/ASIC
Area	700-1800 LEs	1269 LUTs	0.26mm^2	N/A	N/A

Table 2.1: Comparisons between known soft-cores **(**Tong et al, 2006)

This project used the Nios® II soft-core from Altera® Corporation, which provides a wide range of customizable options such as those shown above, and possesses Harvard memory architecture. It is available for use in academia for research, and comes in three versions, which include economy, standard and fast core. The fast core version should be used, since it provides the most features and performance. It has the capability to execute up to 256 custom instructions, and has a fair performance of 150 DMIPS and one of the highest operating frequencies on a FPGA platform (Tong *et al.*, 2006).

2.5 Nios® II soft-core processor

The Nios® II soft-core processor is programmed by software code in C/C++ language. As this processor is incorporated into the FPGA hardware, the latter is controlled to work in conjunction with the code and help in accelerating some functions (Joshi, 2010). For reasons mentioned above, it is the most essential element involved in this entire project. As an FPGA embedded processor, the benefits that contributed to its selection as the major element of this project, are described in the rest of this chapter.

The Nios® II processor is a popular 32 bit RISC embedded processor by Altera®, which was designed to address a wide range of embedded applications. It exists in three cores, which are described below:

- Fast (Nios® II/f) - Optimized for best and high performance on computationally intensive, arithmetic performance-critical applications, and also for applications, which require large amounts of data and codes, such as a full-featured operating system. Hence, it consumes the most LEs.
- Economy (Nios® II/e) - Optimized to use the fewest LE, area and memory. But it is efficient enough to provide sufficient performance for small applications.
- Standard (Nios® II/s) - This core offers better performance and consumes more LEs and memory than the economy core. The Nios® II/s core is optimal for applications, which require medium performance and more amounts of code and/or data for which an economy core would be less suitable (Altera Corporation, 2010f).

Following the level of complexity and for optimization, the Fast Nios® II core was selected for this project. The set of processor elements includes:

- Full 32-bit instruction set, data path, and address space;
- 32 general-purpose registers;
- 32 external interrupt sources;
- Single-instruction 32 × 32 multiply and divide producing a 32-bit result;
- Dedicated instructions for computing 64-bit and 128-bit products of multiplication;
- Floating-point instructions for single-precision floating-point operations;
- Single-instruction barrel shifter;
- Access to a variety of on-chip memories; and
- Instruction set architecture systems (Albiach, 2006).

The flexible nature of the Nios® II processor is promoted by the ability given to designers to implement the core, which is tailored according to system specifications and requirements by defining custom instructions and peripherals through the SOPC Builder tool.

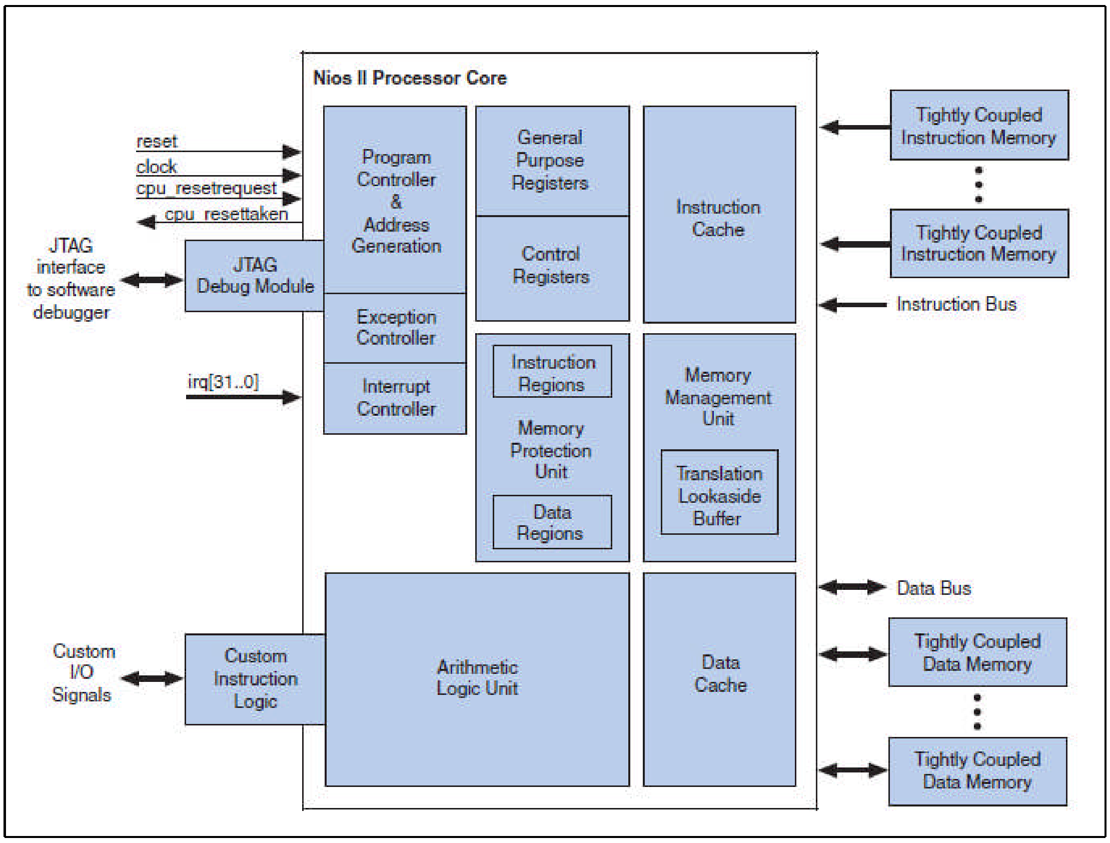


Figure 4.2: Nios® II processor core block diagram from (Altera Corporation, 2010f)

In order to provide memory and I/O access, the core presents several features, as displayed above in Figure 4.2, and are presented below:

- Instruction master port – Its role is to fetch instructions that will be executed by the processor and connects to instruction memory via the Avalon switch fabric. The instruction master port does not perform any write operations.
- Instruction cache - Fast cache memory internal to the Nios® II processor.
- Data master port – It is a 32-bit Avalon master port that connects to data memory and peripherals via Avalon switch fabric. It reads data from memory or a peripheral, when the processor executes a load instruction and writes data to memory or a peripheral when the processor executes a store instruction.

- Data cache - Fast cache memory internal to the Nios® II processor.
- Tightly coupled instruction or data memory port - Interface to fast memory outside the processor.
- JTAG debug module - Its function is to provide a communication bridge between the design's system and the host PC running on Nios® II IDE by downloading programs to memory and starting and stopping code execution. The debug module connection system consists of a download cable such as the Altera® USB Blaster or a system analyzer probe (Altera Corporation, 2010f).

The Nios® II system custom peripheral, via the Avalon switch fabric, is responsible for communication, and for linking both internal logic in FPGA and external hardware in the FPGA development board.

2.6 Contributions of the Nios® II soft-core application in power electronics

The application of a Nios® II soft-core allows for low power consumption, high performance and reduced design time to design reconfigurable power electronic systems. It also presents advantages of flexibility over DSP and other microcontrollers when selecting, configuring, customizing, duplicating or removing easily needed intellectual property (IP) cores, CPUs, peripherals, and interfaces to design power electronic systems at any time by reprogramming the FPGA device. This results in a better hardware performance (Alcalde *et al.*, 2009). Large systems can also be produced in a small design size on a single FPGA chip. Some soft-core processors' internal architecture can be changed in order to suit a particular design. This method of implementation is also extremely fast owing to the straight forward nature of VHDL (Krah *et al.*, 2011). Another advantageous characteristic is the frequency adjustment of the Nios® II processor using Phase Locked Loops (PLL) which plays a crucial role in developing the different ranges of PWM signal frequencies that the SCBC must generate.

2.7 Related and past work on digital control for power electronics

A number of solutions for digital control in the field of power electronics have been proposed and implemented in various academic institutions. These are discussed in the following section.

According to Francis and Boroyevich (2003), the Virginia Polytechnic Institute and State University designed and implemented a Universal Controller for distributed control and Power Electronics Applications was designed and implemented, based on a previous prototype, which was not specific enough to the needs of a distributed controller. The purpose of this work was to save a large amount of time and engineering costs by developing a universal controller that can satisfy the needs of most medium and high power applications in power electronics. The controller interfaces include a dual ring fibre optic interface, a PMC (PCI based mezzanine interface, a generic pin header interface, an upper level control (ControlNet) interface and synchronous serial communications interfaces. The controller uses an Analog Devices ADSP-21160 DSP. The controller is multiprocessor capable (self-stacking), and can be configured to work in single processor mode, multiprocessor cluster mode, or multiprocessor data flow mode. This project provides a solution for the digital control of several power electronic applications.

Heerden (2003) states that the PEC33 reconfigurable controller for power electronic applications was developed at the University of Stellenbosch. This controller is a follow up of the PEC31 controller, which was also developed at the same institution. Its control algorithm was implemented by a TMS320VC33 DSP, and the digital logic design was implemented in PLDs. The development of the control system also involved the design and construction of voltage and current probes for ADC measurements. This work provided information on the overall structure of a digital controller for power electronics applications.

Rauma (2006) asserts that an FPGA-based controller for power electronics was designed at the Laapeeranta University of Technology (LUT) in Finland by Dr Kimmo Rauma. This work compares the control electronics of an analog design and to that of the FPGA-based design. A new generic communication architecture, effective implementation and testing methods were developed. Evaluation of the use of FPGAs and the proposed development methodology is performed with simulations and laboratory measurements by using two power electronic applications; a switched-mode welding machine; and a frequency converter. In both test cases, the implemented algorithms exploited the parallel calculation. This project’s results show FPGA-based digital control’s high efficiency and superior performance over traditional control design.

According to Jooste and Wilkinson (2009), a generic digital controller for power electronics applications was developed on an FPGA at the Cape Peninsula University of Technology

(CPUT). A Pulse Width Modulation (PWM) modulator for a 5-multicell inverter was developed with the use of VHDL and Quartus® II software. Its purpose is to drive a 5-cell multicell inverter with a 20 kHz sinusoidal reference waveform and 300 kHz switching frequency. A DSP development board, in conjunction with a code written in C language, generates sinusoidal reference waveforms, which are required to produce PWM signals. Another code was developed for the transfer of data and communication between the DSP and the FPGA, and the development of some external peripherals. This project provides a lot of information on generic digital control and hence served its purpose, but required two ICs to work with.

Alcade *et al*. (2009) state that the application of the Nios® II processor in a PFC converter was performed at the Federal University of Santa Catarina (Brazil). The focus of this work was to apply the Nios® II soft-core processor to the digital control of a single-phase pre-regulated rectifier, which showed advantages and disadvantages of the use of this technology. A particular control strategy was implemented to obtain Power Factor Correction
(PFC) of a single-phase voltage doubler rectifier with a center tap at the voltage output. This project served its purpose by providing information and displaying multiple characteristics of the FPGA, incorporating the Nios® II soft-core processor and its application in power electronics.

According to Urriza *et al*. (2009), a team of researchers from the Departamento de Ingenieria Electronica y Communicaciones, Universidade de Zaragoza in Spain developed a soft-core based FPGA digital controller for a DC-DC converter. In their design, the controller combines a Microblaze® core with a specific customized peripheral. The Microblaze® core was programmed in C language. It facilitated control algorithm by using floating point type variables, which are usually difficult to implement in FPGAs. The customized was described by using a Hardware Description Language (VHDL) to generate the PWM signals, and to transmit signals to the analog to digital converter (ADC). This project has provided insight for the implementation of a commercial soft-core based digital controller by using floating point data type. However, its drawback is that it can operate on only one power electronic application.

2.8 Summary

This chapter briefly described the open-loop control and PWM, and provided an overview of soft-core processors. Controller topologies such as DSP, FPGA, and DSP/FPGA controllers were discussed and compared. An overview of the Nios® II processor was also provided. Finally, several digital controllers' implementation methods in power electronics from previous work were also mentioned. A FPGA embedded processor was chosen as the core of this work, and regarded as a potential tool to create an efficient and generic digital controller for power electronic applications.

CHAPTER 3

HARDWARE AND SOFTWARE TOOLS

3.1 Introduction

This chapter presents the required hardware and software for the development of this project are presented, while the features of the FPGA development board, and Cyclone® 3C25F324C8, which is involved in the realization of the digital controller, are also presented.

3.2 Cyclone® III EP3C25 Starter Development board

FPGAs are programmable ICs, which consist of programming logic components called logic blocks, which are surrounded by programmable I/O cells and are interconnected together with routing programmable wires, forming logic blocks, as shown in Figure 3.1 below. Logic blocks consist of programmable Look-Up Table and registers (flip-flops). The LUT can be configured to perform any logic function on its inputs to produce one single logic output. The final output is either this new value or the previous value that is stored in the register (Grout, 2008: 28). Their reprogrammability is promoted by the presence of memory elements such as EPROM, EEPROM, flash and SRAM chips.

The FPGAs' structure enables the implementation of digital hardware functions, simple ones such as AND and XOR logic gates or more complex ones. These can be accomplished by utilizing programming procedures, which can be performed without a long and expensive design process. The applications of FPGAs are seen in a variety of industrial applications and areas such as digital signal processing, medical systems, embedded systems and electrical systems.

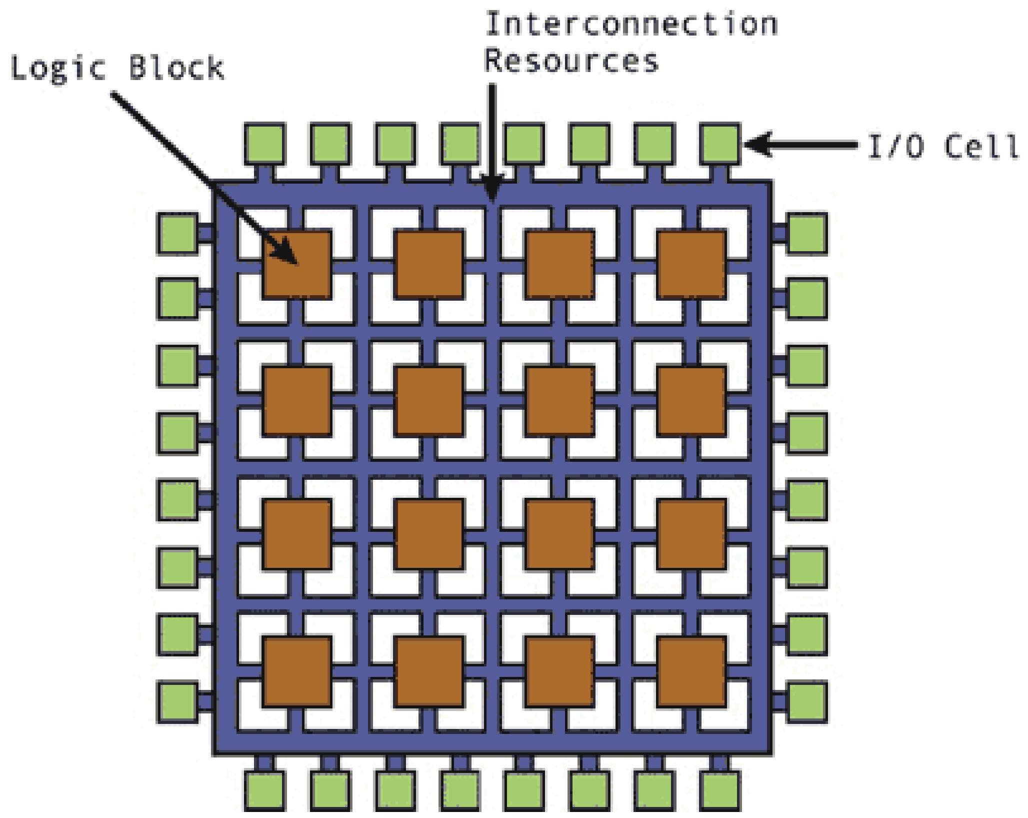


Figure 3.1: FPGA architecture (Zeidman, 2009)

An FPGA design is made possible with the use of CAD tools that convert the designer's specifications into the desired FPGA configuration. It is described by using an HDL such as Verilog or VHDL or a schematic design diagram. They are then synthesized, placed and routed to form a bit file, which is used to program the FPGA. The HDL codes offer a better alternative when working large systems designs, while schematic entry is more suitable for smaller designs, which can be drawn piece by piece. HDL codes and schematic diagrams are inputs into these CAD tools, while the output is a .sof file. This file is a Quartus® II SRAM Object File for FPGA configuration.

The Cyclone ® III EP3C25F324C8 Starter Development board is the chosen FPGA platform, among the others, for the development of the SCBC. This board provides a low cost and easy introduction to FPGA technology. Other advantages are its low power consumption, high functionality. This kit includes all components, which are necessary to create and implement automotive, consumer, wireless communications, video processing, or other high-volume, cost-sensitive application designs.

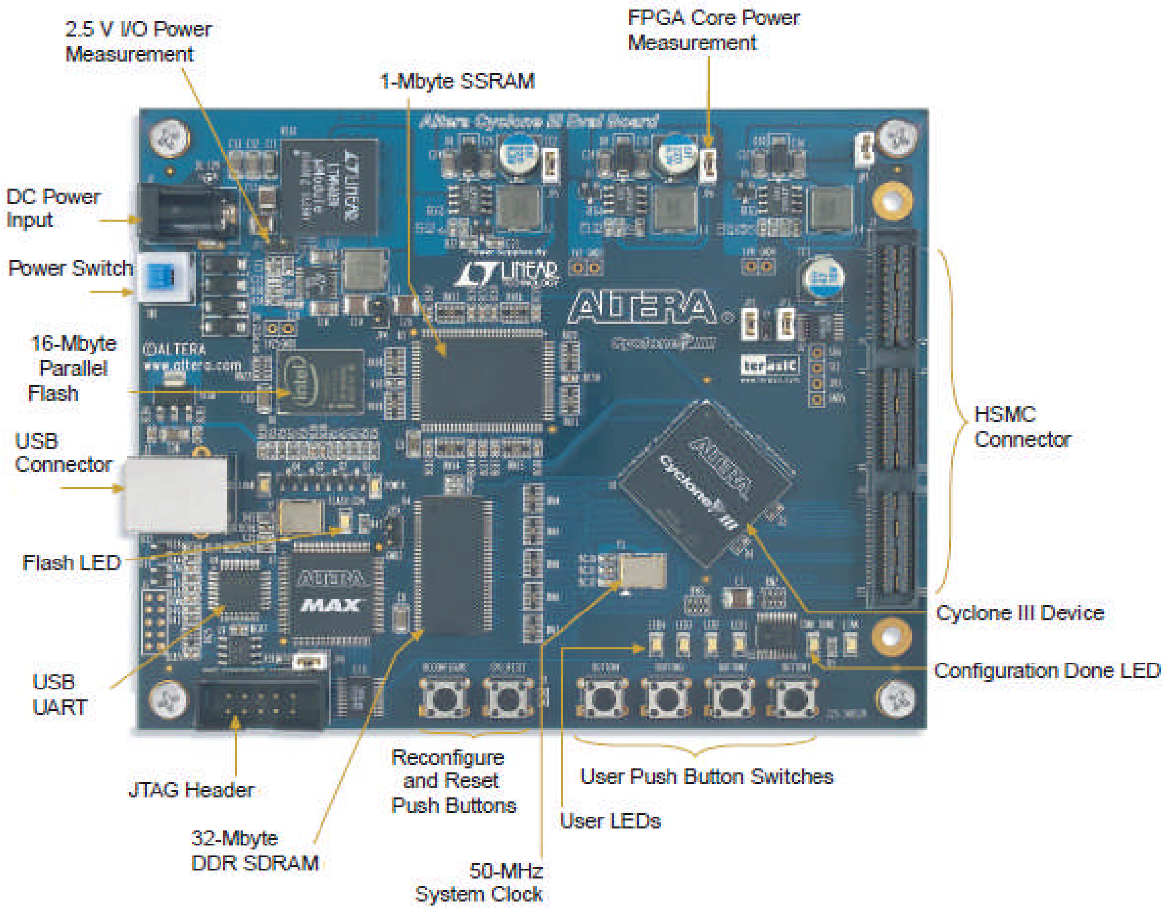


Figure 3.2: Top View of the Cyclone® III FPGA Starter Board from (Altera Corporation, 2010b)

It provides enough I/O pins to work with and Intellectual Properties (IP) are available for this series of FPGA chip (Altera Corporation, 2010b). Figure 3.2 displays some of the hardware components of the Cyclone® 3C25F324C8 board, while all the main components of the Cyclone III starter development board are listed below:

- Low-power consumption Altera® Cyclone III EP3C25 FPGA chip;
- 25K logic elements (LEs);
- 66 M9K memory blocks;
- 16 18x18 multiplier blocks;
- 4 PLLs;

- 214 I/O pins (of which 84 I/O pins for communicating with HSMC daughter cards);
- HSMC connector;
- USB type B connector;
- 32-Mbyte DDR SDRAM;
- 50 MHz on-board oscillator;
- 16-Mbyte parallel flash device (CFI);
- 1 Mbyte high-speed SSRAM memory;
- Four user push-button switches;
- External power supply with U.S. adaptor;
- Four user LEDs;
- Power switch;
- DC power input; and
- Built-in USB-Blaster interface (Altera Corporation, 2010f).

Digital systems such as the PWM modulator for power electronics is easily implemented on FPGAs. Designs are implemented on FPGA so that they can be easily modified to address problems or to add or remove new functions and features, if necessary. This flexibility amounts to the ease of changing interconnections between logic blocks. Their designs provide a reduction in costs, design complexity and time spent on design and implementation. The reconfiguration is then easily performed by using the JTAG interface.

3.3 Controller software components

The software tools that were involved in the process of designing and development are discussed in this section. Table 3.1 lists these tools and are described in the next section below.

Tools /Package	Usage
MATLAB R2009 version 7.8.0	Mathematical computation and analysis tool
Quartus® II software version 9.1	Integrated synthesis and implementation tool from Altera®
DSP Builder version 9.1	Mathematical computation, simulation and analysis tool from Altera®
Nios® II IDE version 9.1	C/C++ compiler, design and verification environment for Nios® II systems

Table 3.1: Software tools used for the SCBC development

3.3.1 DSP Builder and MATLAB/Simulink

In this work, DSP Builder was the first software tool to be used for the PWM modulator's development. DSP Builder allows designers to perform algorithmic design, system integration and simulation of VHDL based descriptions in a single GUI, on MATLAB®/Simulink models in sampled time, whilst porting them to hardware description language (HDL) files and Tcl scripts for synthesis, hardware implementation and simulation with the Quartus® II software. Algorithms that are developed in DSP Builder can be connected to a Nios® II processor as a custom peripheral by involving SOPC Builder.
Figure 3.3 shows this possibility of a DSP Builder design to be inserted into a SOPC system as a peripheral PIO or custom instructions.

MATLAB® (MATrix LABoratory), which is commercially available by MathWorks®, is a most popular software for an algorithm design as it enables numerical computation, system simulations and data visualization by providing a technical computing environment, which is designed to support the implementation of computational tasks (EE TimesAsia, 2007; Chapman, 2008). MATLAB® guide, the GUI Development Environment, is the tool, which is used to create MATLAB® GUIs. The programmer can layout the GUI, selecting and placing various components to be placed in it, hence creating a graphical display, while eliminating the need of typing commands or writing command scripts.

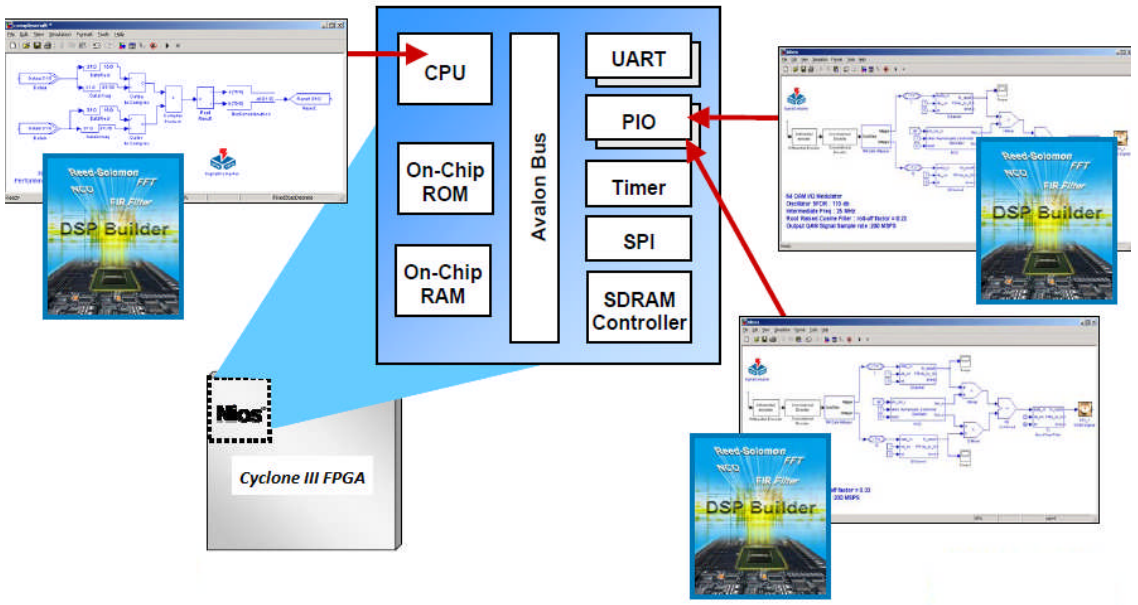


Figure 3.3: Diagram of DSP Builder designs as custom peripherals or custom instructions from (Altera Corporation, 2002)

3.3.2 Altera®'s Quartus® II software

This is the principal software, which was used in this project and operates with several important, highly efficient and convenient CAD tools, depending on the design's complexity to develop VHDL or Verilog structures as overall projects in order to download to FPGA boards. These Quartus® II tools include the SOPC builder, the Nios® II IDE and ModelSim® (Albiach, 2006). Even in the absence of the other tools mentioned above, Quartus® II can still provide a complete design environment for low complexity designs.

3.3.3 SOPC builder

SOPC Builder is part of the Quartus® II software, and is a powerful system development tool that creates the hardware structure of an embedded system-on-programmable chip system by specifying and integrating hardware components such as processors, peripherals and memories. It facilitates both hardware and software design. All a designer needs to do is to specify the components when using SOPC Builder. It automatically generates the interconnect logic, simulation projects files, testbenches, execution scripts for simulation, header files and driver routines.

3.3.4 Nios® II IDE

Nios® II IDE is an Altera® software development environment for all Nios® II processors. It is separate from the SOPC Builder tool, but it acts as the software development platform by utilizing SOPC Builder outputs. It accomplishes this by providing the software toolset that allows the designer to enter the design software source code to carry out all crucial tasks such as creating, editing, building, running and debugging of software. It also goes through essential phases, which are necessary for the source code to be executed, to accommodate project management, and to offer software code debugging.

3.4 Software languages

The software part of an embedded system consists of codes, which are written with programming languages such as C/C++, VHDL, VHDL-AMS, SPICE, Verilog, assembler and Java that control the functioning and interaction of individual hardware parts. This work utilized VHDL and C/C++, MATLAB® to develop the controller.

3.4.1 VHDL

VHDL stands for VHSIC Hardware Description Language, where VHSIC stands for Very High Speed Integrated Circuits. VHDL is a modern, powerful, portable, readable and general-purpose hardware language defined by IEEE standards. It describes electronic circuits or systems on behavioural, structural, physical aspects, which can be implemented as a physical circuit or system (Grout, 2008: 195). CPLDs, ASICs and FPGAs are the three main digital architectures, which utilize VHDL. One important benefit of VHDL is that it allows design systems to be accurately described for automatic circuit synthesis or/and simulation without any hardware being manufactured. Modifications can easily be applied at any time, while sparing unnecessary costs for hardware prototyping (Pedroni, 2004: 3; Guan, 2010).

3.4.2 C language

C language is compiled on UNIX™, Windows, and Linux operating systems and is one of the most common programming languages today. Its closeness with hardware language makes it widely used in embedded systems development, and is mostly employed to develop software codes to program processors in embedded system designs. This software description language is processor-independent and can be easier to define complex algorithms than HDL, and can thus facilitate FPGA design. Another advantage is that C Compilers are easily available for most processors (Barr, 1999: 18).

3.5 Summary

This chapter discussed the different hardware and software parts involved in the soft-core based controller design. The hardware components consist of the embedded Nios® II core in a Cyclone® 3C25 FPGA, and a peripheral board acting as a communication interface between the controller SOPC system and power electronic applications. The software elements involved consist of MATLAB®, DSP Builder, Quartus® II and Nios® II IDE software.

CHAPTER 4

HARDWARE/SOFTWARE CO- DESIGN AND IMPLEMENTATION

4.1 Introduction

This project involves the development of an embedded system-on-programmable-chip (SOPC). Many embedded systems consist of processors, which run software codes and are interfaced to external hardware circuitry. Such designs require that both hardware and software parts should be designed together in parallel (Grout, 2008: 63). In this case, the referred hardware is the HDL system design. A System-on-a-Programmable-Chip (SoPC or SOPC) is developed by integration of an electronic system's components into a single programmable integrated circuit (chip). It may contain various types of signals and performs various functions. SOPCs are the core of modern embedded systems (Koskinen, 2009).

The hardware comprises of blocks that are placed together by using CAD tools. The software aims to control the soft-processor and the external hardware peripherals the SOPC consists of. This is achieved by creating codes in a software development environment.

This chapter elaborates on the specifications of this project, and provides details about the portioning of the overall design into hardware and software parts, while it also discusses the interfacing of these parts to complete the overall system development and design implementation, both in hardware and software.

4.2 Soft-core based controller specifications

In order to ensure the generic, reprogrammable and flexible characteristics of the SCBC, the following project specifications were established and are listed in Table 4.1 below.

SOFT-CORE BASED CONTROLLER SPECIFICATIONS	
Programmable Logic Device	Altera® Cyclone 3C25F324C8 FPGA chip, from the Cyclone series FPGAs
FPGA embedded processor	Reconfigurable Altera® 32-bit Reduced Instruction Set Computer (RISC) Nios II soft-core processor, running at 50 MHz.
Switching techniques	Pulse Width Modulation (PWM), with fixed or variable duty-cycle
Switching frequency range	50 Hz – 3 MHz
Number of reference signal inputs	A single internal reference (sine) signal from the Nios® II processor
Number of gate switching digital outputs	10 (5 complimentary pairs of high and low side switches), to accommodate the maximum number of switching signals for the 5-cell multilevel
User Communication Interface	MATLAB® GUIs
Memory	High capacity volatile and non volatile memory for program and data storage. SSRAM: 1Mbyte Flash: 16 Mbytes
P.E. Applications to operate on	• Half bridge converter • Half bridge inverter • Full bridge converter • Full bridge inverter • Three phase inverter • 5-cell multilevel inverter
Connectors	(Internal) Avalon Interconnect Fabric, connecting all the internal hardware modules of the Nios® II system. (External) USB-blaster download cable, connecting the FPGA to the PC.

Table 4.1: Soft-core based controller specifications

4.3 Controller overview

As mentioned earlier in Chapter Two, embedded processors provide overall system integration, faster execution and flexibility, while partitioning the system between hardware and software. In order for these specifications to be realized, the digital controller integrates four crucial parts, which are developed in parallel. These are described below:

- A DSP builder based PWM modulator firmware - This module generates PWM signals that are required to drive and control the converters and inverters' switches gates via the FPGA I/O pins. They are produced from sine and triangular waveform comparison.

- Quartus® II/SOPC system module - It consists of the digital controller's architecture in terms of FPGA hardware and is generated by the SOPC Builder. The hardware system is completed once the SOPC Builder module is instantiated in the Quartus® II project.

- The Nios® II software module - Based on the Nios® II IDE project which involves the building, compiling and running of the C source programs that run on the processor.

- MATLAB® GUI-based communication interfaces - It will provide the digital controller user with choices on the controller's system input parameters.

- PCB production - It holds a connection platform between the SCBC's outputs and the external world.

Figure 4.1 gives an illustration of the proposed controller's architecture in terms of this hardware / software co-design and implementation.

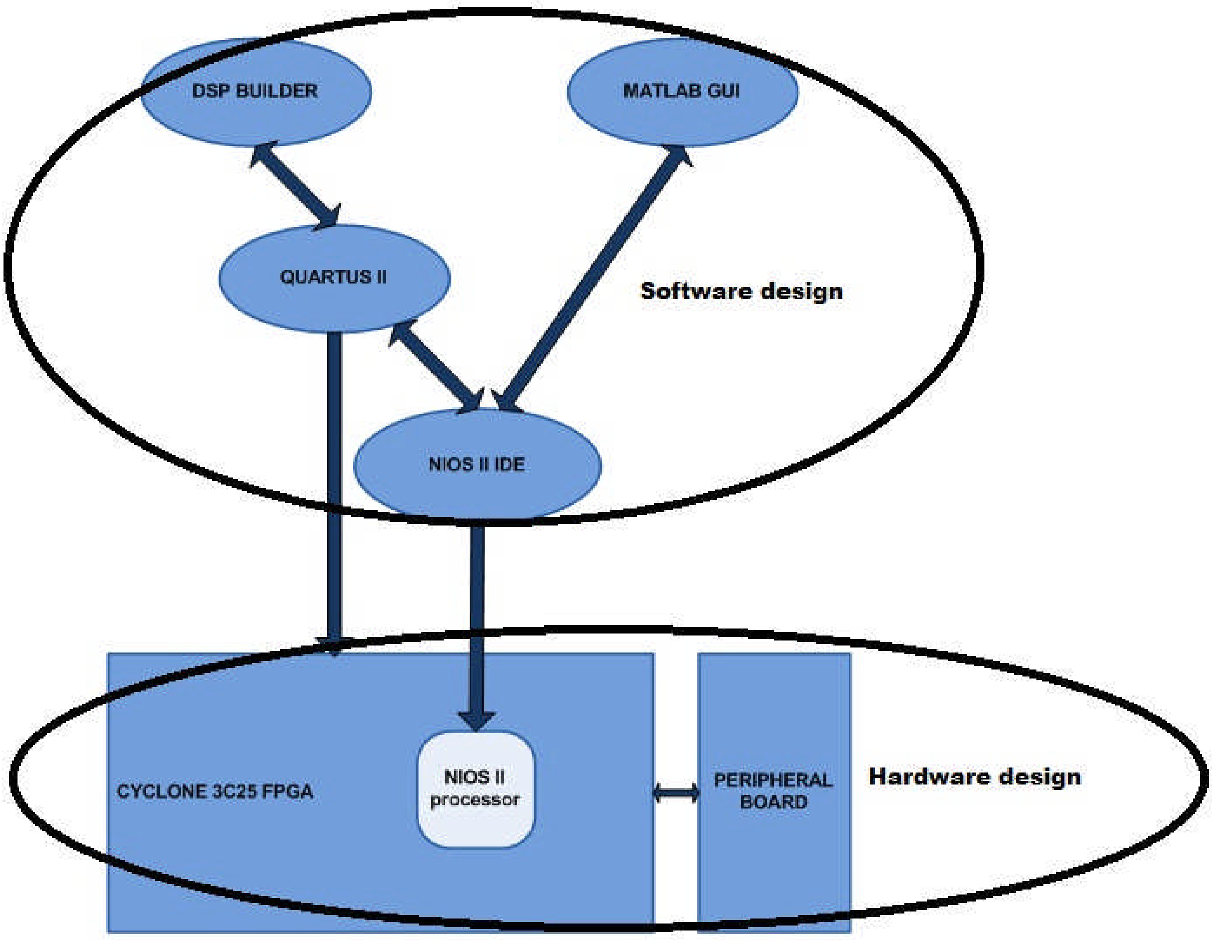


Figure 4.1: Nios® II - based controller's hardware and software architecture

4.3.1 Hardware design overview

In this work the hardware development of the SCBC system was implemented on a DSP Builder design, which was inserted into a Quartus® II project, and incorporated the Nios® II processor. The FPGA was a Cyclone® 3C25F324C8 FPGA, as mentioned in the previous chapter and was available at the CIR laboratory. The architecture of the SOPC system was determined by using the SOPC Builder tool in the Quartus® II software, while the hardware aspect of the PWM modulator was realized by using VHDL imports on DSP Builder. A small peripheral PCB was also developed to place a communication platform between the embedded Nios® II system and power electronic applications by placing a High speed Mezzanine Card on it.

In order to make the hardware design possible, some factors had to be taken into consideration during the hardware design of the digital controller, which are the following:

- The amount of available LE and memory space;
- The compatibility and interconnection of the DSP Builder design with the rest of the FPGA and Nios® II systems;
- The number of I/O FPGA pins available to drive PE application gates; and
- Availability of Intellectual Properties (IP) cores.

4.3.2 Software design overview

In this work the system's software was executed by the Nios® II processor. It is a VHDL defined circuit built around a CPU and, which used the FPGA, m's general logic resources in order to implement hardware functions in parallel. The software development of the soft-core based digital controller was based on all the software tools, mentioned in Chapter Three. As in the case of hardware design, certain factors were taken into consideration for the software design:

- The compatibility of HDL imports and data formatting within the DSP Builder design;
- Programming methods of controlling both DSP Builder design and Nios® II system and their intercommunication via a C code;
- An adequate description of SOPC modules and tasks in a C code;
- Adequate usage of memory bits; and
- Possible communication methods between the Nios® II system and PC.

4.4 Project design flow

In order to develop and demonstrate the capabilities of the controller, five stages were undertaken together, in parallel, namely:

- **Stage 1**. PWM modulator hardware design was created and implemented on a DSP Builder/Simulink model via HDL imports and Altera® Avalon Slave blocks;

- **Stage 2**. Creation of the controller’s SOPC system;
- **Stage 3**. Completion of the overall Quartus® II system;
- **Stage 4**. Development of C code for sine wave samples generation and overall control of the digital controller;
- **Stage 5**. Creation of MATLAB® GUIs and m-files for the controller’s parameters control, signal capture and communication between the controller’s user, the DSP Builder and the Nios® II system; and
- **Stage 6.** Production of a peripheral PCB to place a communication interface between the controller and the external world.

4.4.1 Stage 1 – DSP Builder design and implementation

The DSP Builder tool provides an integrated environment where hardware peripherals are developed on Simulink models. This software tool was used for the PWM modulator’s hardware implementation. The modulator design was made up of HDL import IPs and DSP Builder blockset library elements, which were converted into a HDL Avalon custom peripheral, integrated in a SOPC system and finally downloaded onto the FPGA platform via programming. Each hardware block of the DSP Builder design is presented in Figure 4.3, and are listed below:

- Avalon Slave Interface block;
- RAM block;
- Reference (sinusoidal) signal generator block;
- Carrier (triangular) signal generator block;
- Comparator block;
- Shift and deadtime block; and
- PWM block.

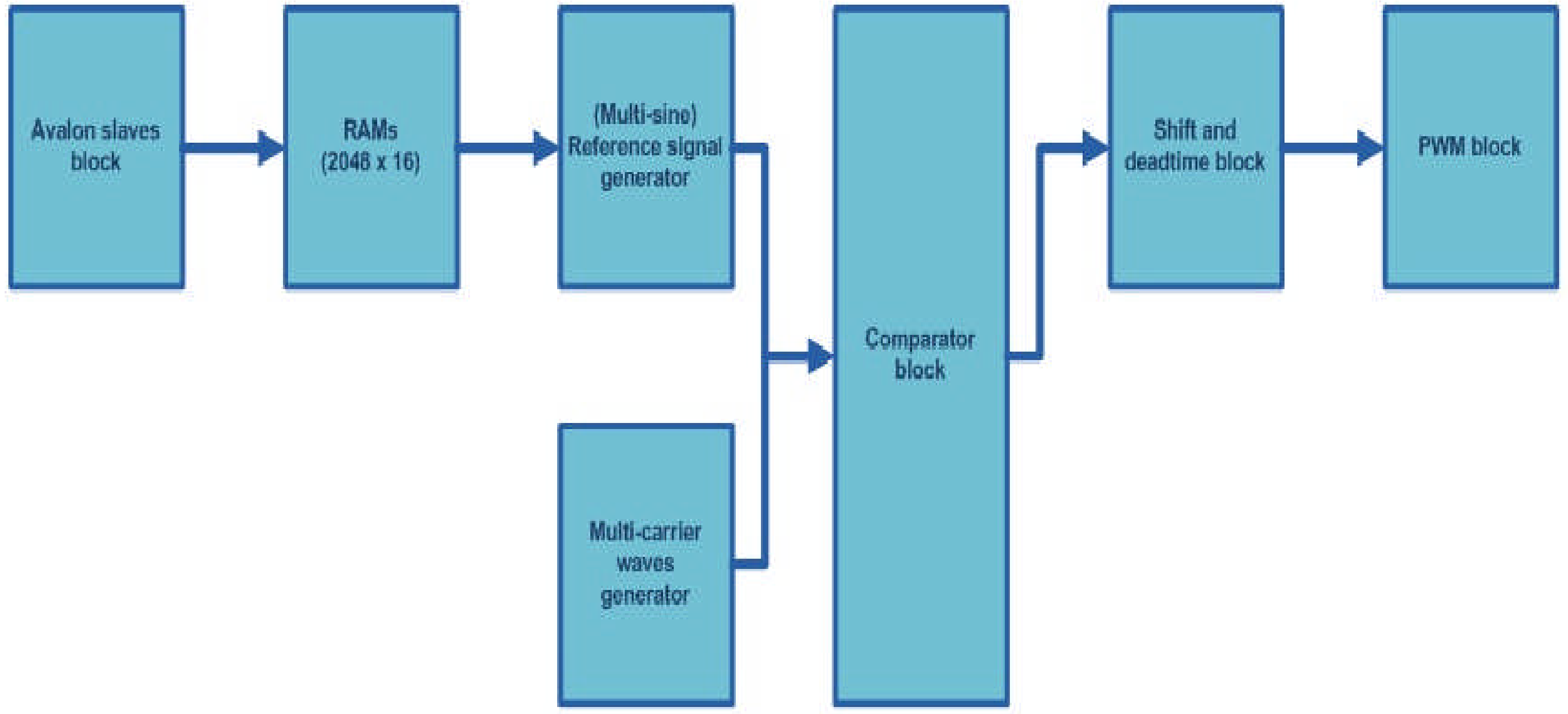


Figure 4.2: The controller’s DSP Builder design block diagram

4.4.1.1 Avalon Slave Interface block

To integrate the DSP Builder design as a hardware slave peripheral into the SOPC system, Avalon Memory Mapped (MM) Slave Interface blocks have to be placed onto the model. This also ensures a connection to the Avalon Interconnect Fabric, which in turn allows communication with the Nios® II CPU, the master peripheral, and the rest of the SOPC slave peripherals. This communication is performed by data transfer through the SOPC components addresses.

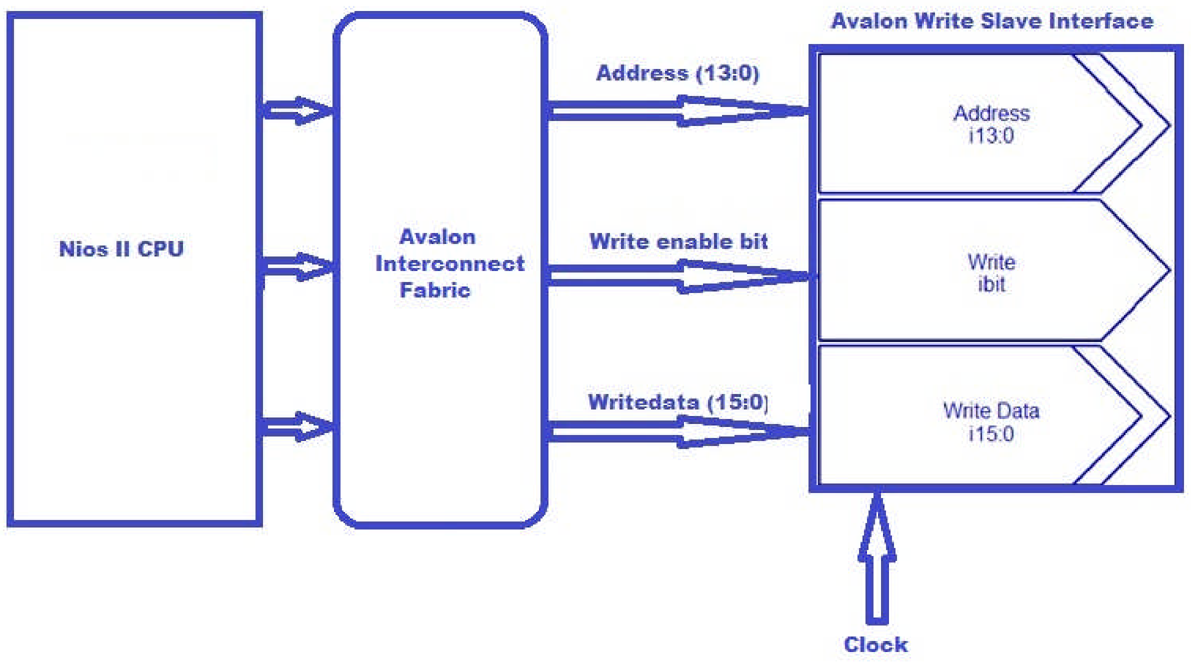


Figure 4.3: Avalon Slave Write block as a sine wave generator

In order for these interfaces to be supported by the Avalon Interconnect fabric, their address alignment must be set to 'dynamic'. This setting allows the processor master and slave address widths to match by making the Slave Interfaces byte addressable (Altera forums, 2011). Figure 4.3 shows the communication interfacing between the Nios® II CPU and Avalon Write Slave Interface signals (Address, WriteBit, WriteData) through the Avalon Interconnect Fabric.

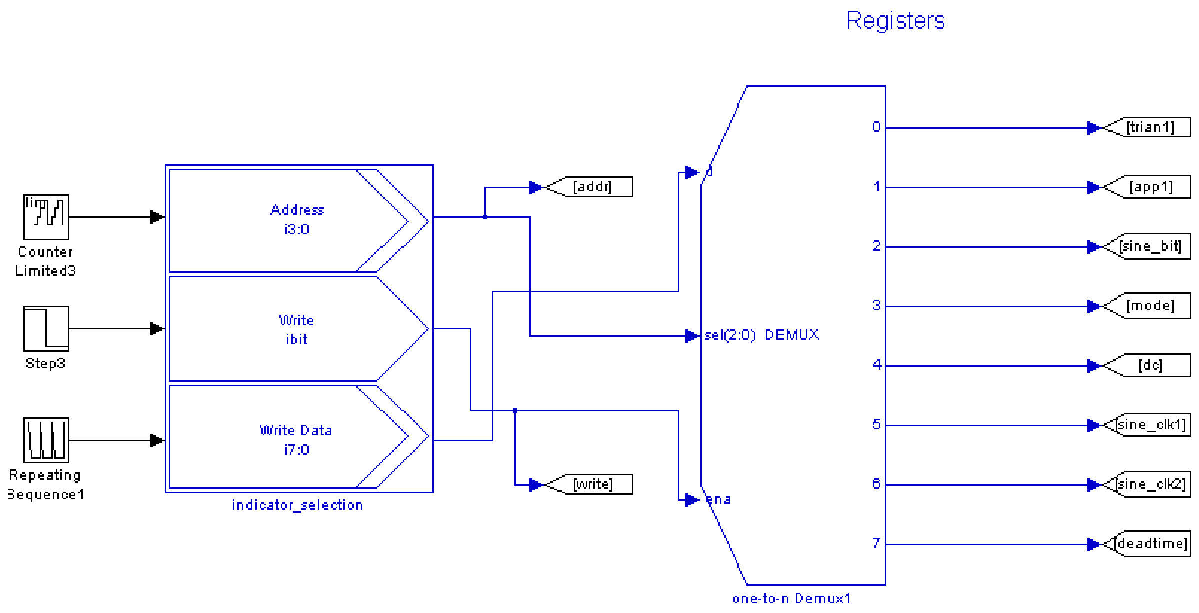


Figure 4.4: Avalon MM write slave interface for control data generation in DSP Builder

In general, an Avalon Slave peripheral pasted on a DSP Builder model is directly connected to the FPGA®'s global clock signal. In this case, it should have been connected to the 50 MHz clock signal, with a 20 ns as the real-world clock period. In order to make the generation of a 50 Hz sinusoidal signal for three phase inverters or other reference signal frequencies possible, different base clock signals were employed.

To transfer the sine wave samples, generated by the Nios® II processor, to the rest of DSP Builder hardware implementation, a pair of 16-bit Avalon Write Slave interface blocks was employed. Another single 8-bit Avalon Write Slave interface block was connected to a 1-to-8 bit demultiplexer, as shown in Figure 4.4 to create a selection system that generates coefficient values to serve as control inputs to the DSP Builder hardware design. Data generated from this slave interface cannot be higher than 8 bits.

4.4.1.2 RAM Block

The memory and LUT blocks of the FPGA architecture were configured as the external memory SSRAM. This memory module is essential to the software part, since it stores sine samples generated by the Nios® II core (Schietekat, 2011). The wordlength of the processor memory is directly proportional to the amount of available memory.

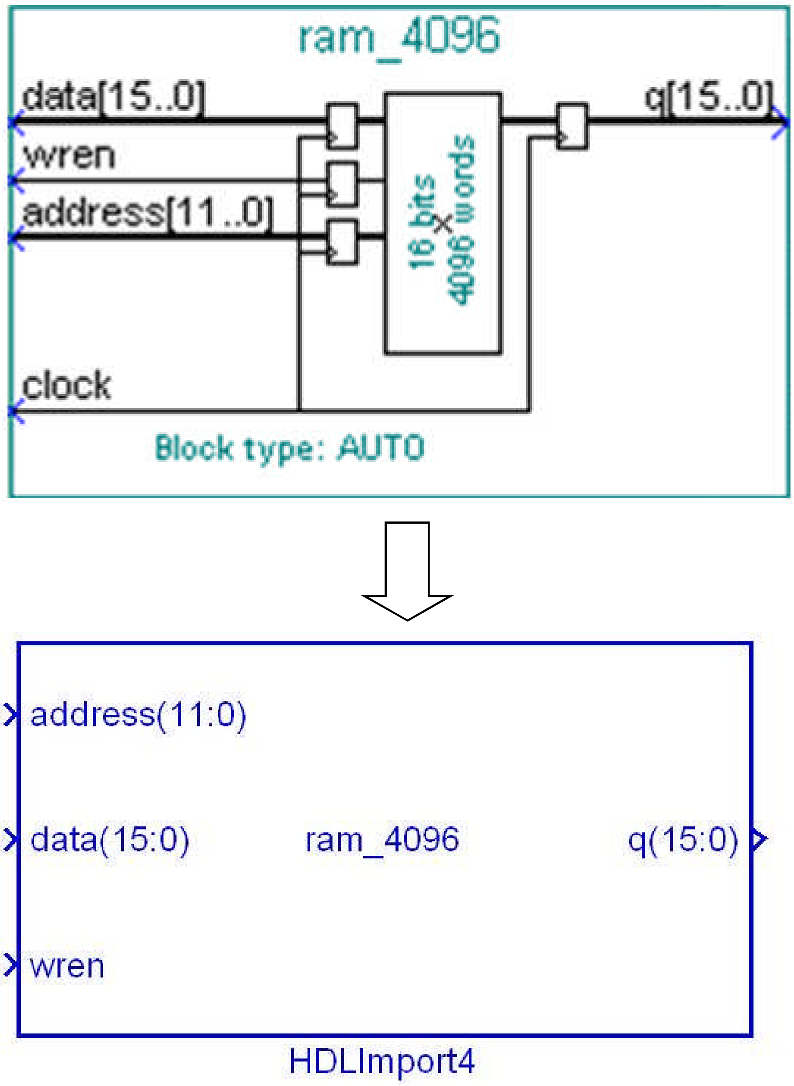


Figure 4.5: Structure of "RAM_4096" IP into a HDL import unit

A pair of MegaFunction single-port RAMs, displayed by Figure 4.5, was employed in the DSP Builder design. The advantage they offer compared to regular RAM blocks is the reduction in the number of LEs to be used in the design. Each of these RAM units forms a sine Lookup table (LUT), as it stores a quarter of a sinusoidal wave. Their memory capacity of 65536 ram_bits allows signal quarter to run at the lowest frequency of 50 Hz. The "addr(11:0)" parameter represents the sample addresses, namely 2^{12} addresses, which are generated and controlled by the Nios® II through the C code. The "d(15:0)" parameter represents the magnitude of the 16 bit (unsigned integer) sine samples, which are generated by the Avalon Write Slave. The "wren" parameter is the write enable bit of the RAMs. The SSRAM was configured with a size of 1 MByte during the SOPC Builder development process. The formula below was employed to determine the capacity of the RAMs and the number of sine samples required to generate the reference signal of a specific frequency:

$$f \text{ reference } = \frac{\Delta \text{ x } f \text{ clock}}{2^{\text{n}} \text{ samples}} \qquad (4.1)$$

Where: f reference is the reference (sine) signal frequency

Δ is the incremental step

f clock is the FPGA clock signal, which controls the DSP Builder model

n is the number of bits

2^n is the number of clock count, samples for a full wave period. This 2^n value divided by 4 is the number of RAM address lines.

Figure 4.6 shows the whole section of the Avalon slave blocks and the RAMs from the overall DSP Builder's design. The output signals from "Avalon_MM_Write_Slave" and "Avalon_MM_Write_Slave1" interfaces serve as input signals to the RAM block, as shown in Figure 4.9. The output signals from the "indicator_selection" interface are inputs to the sections, which involve the development of both carrier and reference signals.

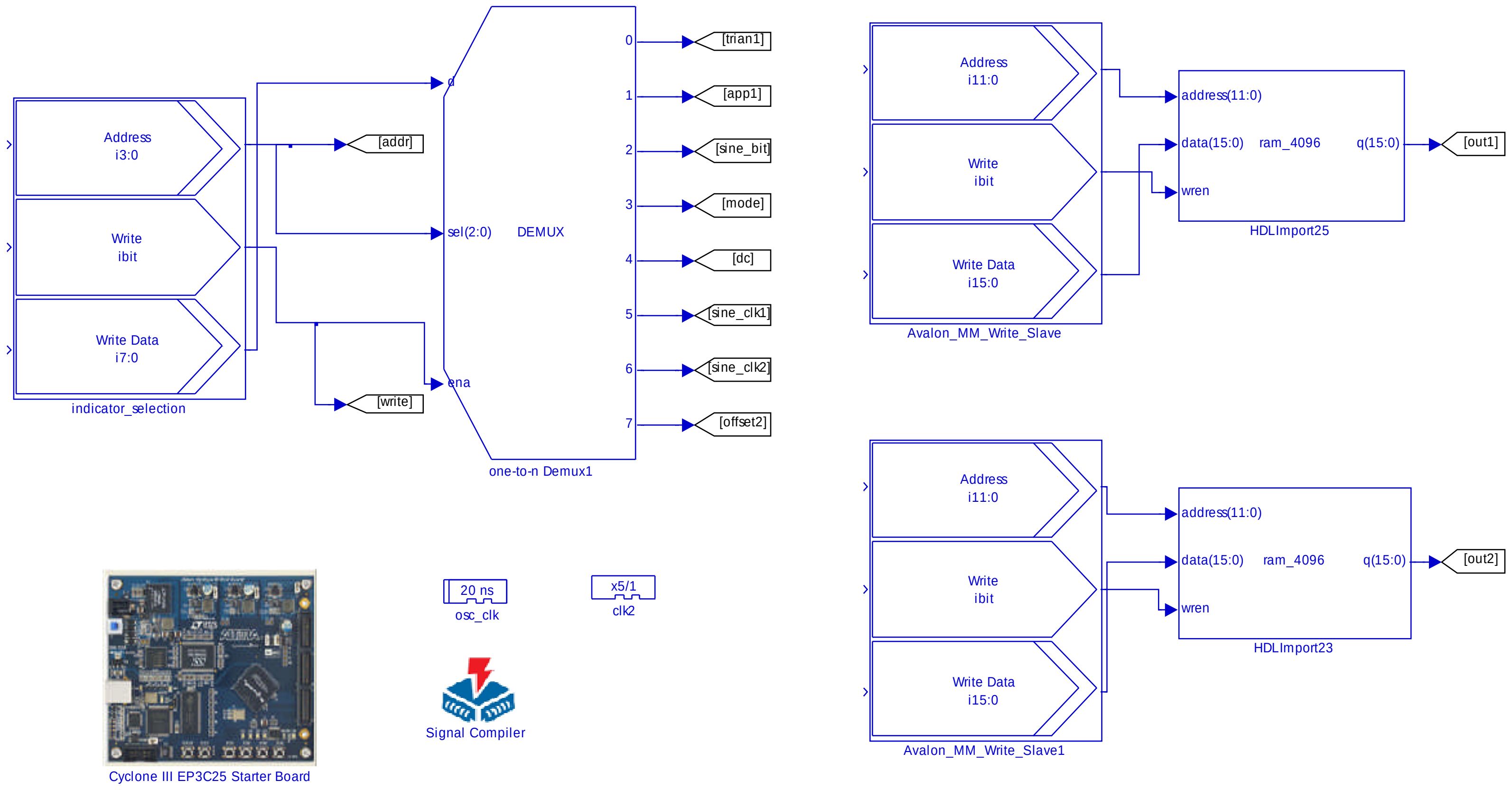


Figure 4.6: Avalon Memory-Mapped slaves and RAMs section

4.4.1.3 Reference (sine) wave signal generation block

The generation of sinusoidal waves was achieved by exploiting their symmetrical properties, as it helps in saving large amounts of logic and memory bits in the SOPC system. Own-written HDL imports and DSP Builder blockset library elements were used. As mentioned above and displayed below in Figure 4.5, the process started as a pair of MegaFunction single-port RAMs. They were used to produce two sine Lookup tables (LUTs), storing the first and second quarters of a sinusoidal signal. Both quarters are connected together by some HDL logic block to produce the first half of the sine wave.

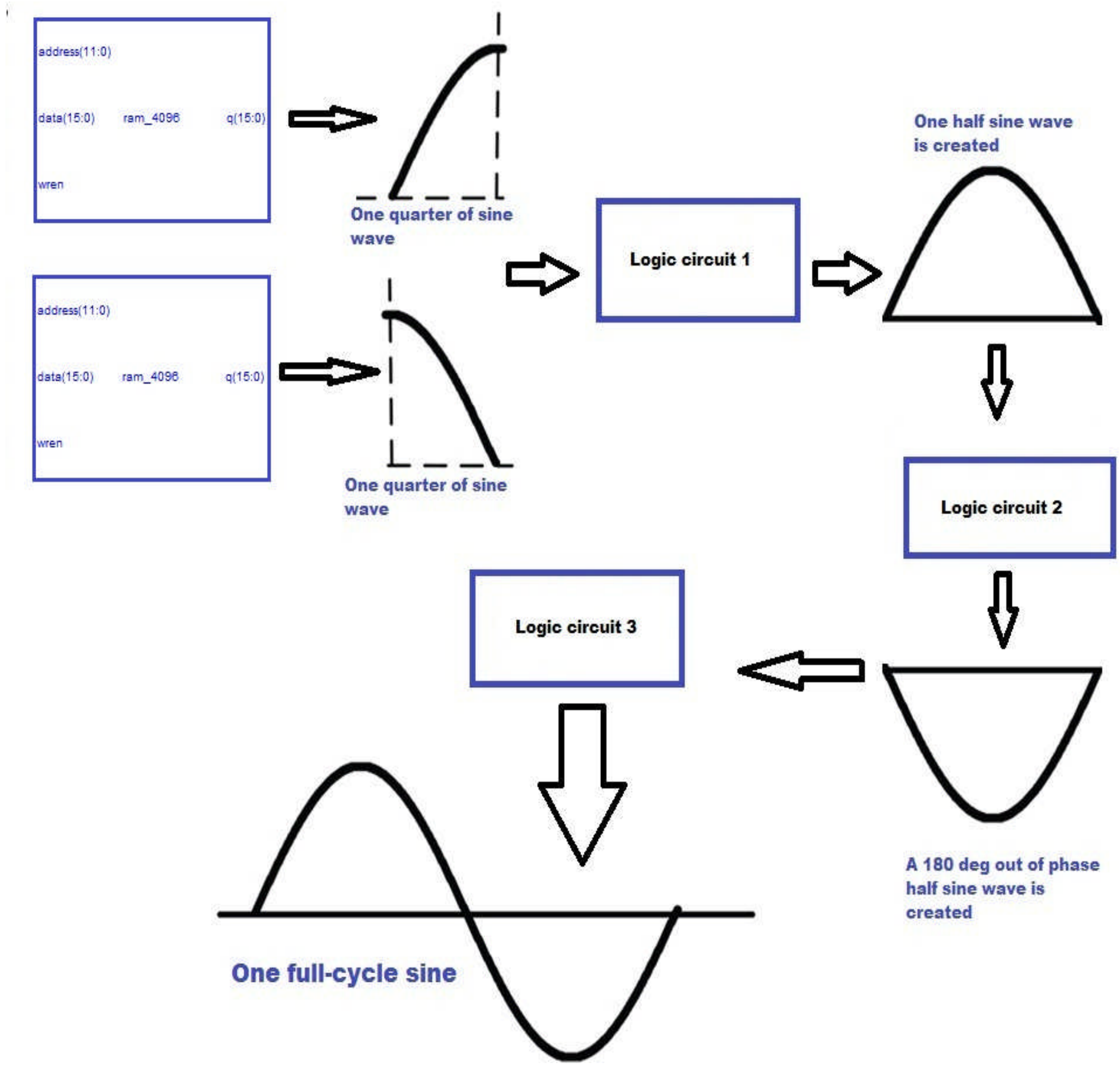


Figure 4.7: Full reference (sine) signal development

Figure 4.7 illustrates the full sine wave development process. Another HDL logic block was employed to create a reverse half signal from the first half one. Both signals were then finally attached to form a full sine wave.

The incremental step worked on an address hopping method, where sine samples were skipped and thus higher frequencies were achieved this way. This was easily performed by “for” loops in the C code program, which operated on the Nios® II core. Figure 4.8 below describes the method.

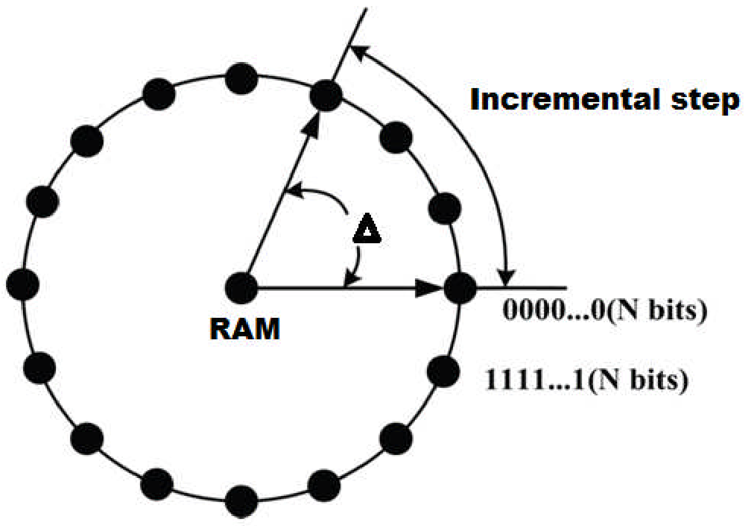


Figure 4.8: Illustration of the incremental step along a circle, which represents a sine wave (Zhang, 2010)

The circle represents the 360° full cycle of a sine wave, while the dots represent the samples of the signal. The incremental value represents the increase of advance along the sine wave. As *n* dots are skipped, the incremental step value becomes 2^n. This results in the generation of higher frequencies and in turn, fewer samples make up LUTs to make up a full circle. It was determined that 4096 was the maximum number of words for each RAM block, which could be implemented to obtain the lowest required frequency on the Cyclone® 3C25F248 FPGA. Since the soft-core controller was required to work on a range of sine wave frequencies, the lowest frequency was taken as the reference frequency when frequency adjustments were necessary. The incremental step in the “for” loop in the C code was increased whenever the reference signal frequency was required to decrease as the sine samples were generated.

The global clock of the FPGA is 50 MHz and the overall number of sine samples was 16384. This gave the lowest reference frequency of 3.125 kHz, as the calculations below show after using (4.5):

$$f\ \text{reference} = \frac{1 \text{ x } 50 \text{ MHz}}{16384 \text{ samples}} \qquad (4.2)$$

$$= 3.125 \text{ kHz} \qquad (4.3)$$

From this reference frequency, the sine wave could operate at higher frequencies of up to 100 kHz. For power electronic applications such as half, full bridge and 5-cell multilevel inverters, such a range proved satisfactory. But in order to generate a 50 Hz reference signal, which is required for three phase inverters, a large memory space was required, since the lower the frequency, the more samples are needed. Two options are available to remedy to this problem:

- Memory size increase; and
- Change of FPGA’s operating clock frequency.

With a 50 MHz global clock, approximately one million sine samples would be required to generate a 50 Hz reference signal. The first option could involve the introduction of a 32 Mbyte SDRAM module in the SOPC system. This would be the best solution, as a large amount of logic cells would be consumed, which would make the overall unable to fit into the FPGA device.

The second option is realized by having a phase lock loop (PLL) clock input to the DSP Builder design with the use of the SOPC Builder tool. The 50 MHz FPGA global clock is either multiplied or divided to produce a clock signal that allows both carrier and reference signals to run in a specific lower frequency range. In this work this clock signal was termed “peripheral_clk**”** and was the DSP Builder design operating clock. According to the reference signal frequency formula above, to generate a 50 Hz sine wave of 16384 samples “peripheral_clk” should run at approximately 833 kHz. The 50 MHz clock would need to be divided by 60. By decreasing “peripheral_clk”, which is the clock that the DSP Builder design works on, other ranges for both carrier and reference signal frequencies were created. Possible input signals specifications are summarized in Table 4.2 and Table 4.3.

Input Signal Frequency ranges			
Peripheral_clk	50 MHz	10 MHz	833 kHz
Reference frequency range	100 – 3.125 kHz	20 kHz – 625 Hz	1.66 kHz - 52 Hz
Carrier frequency range	1.6MHz – 50 kHz	800 – 10 kHz	26kHz - 833 Hz

Table 4.2: Input signal frequency ranges

As the 50 MHz FPGA's global clock signal is divided by a specific denominator, reference and carrier signals are divided by the same denominator, as demonstrated by Table 4.1 above.

Frequency ratio = 16						
Number of memory words (LUT Samples)	**128**	**256**	**512**	**1024**	**2048**	**4096**
Samples size	**16-bit**	**16-bit**	**16-bit**	**16-bit**	**16-bit**	**16-bit**
Reference signal frequency	**100 kHz**	**50 kHz**	**25 kHz**	**12.5 kHz**	**6.25 kHz**	**3.125 kHz**
Carrier signal frequency	**1.6 MHz**	**800 kHz**	**400 kHz**	**200 kHz**	**100 kHz**	**50 kHz**
	Frequency ratio = 8					
Reference signal frequency	**200 kHz**	**100 kHz**	**50 kHz**	**25 kHz**	**12.5 kHz**	**6.25 kHz**
Carrier signal frequency	**1.6 MHz**	**800 kHz**	**400 kHz**	**200 kHz**	**100 kHz**	**50 kHz**

Table 4.3: Examples of input signal frequencies in respect to frequency ratios

According to the Nyquist theorem, a frequency ratio, also termed as a frequency modulation ratio, at least 2 is required.

Fortunately, different frequency ratios can be established on the SCBC by using the formula below (Luo *et al.*, 2005):

$$\text{Frequency ratio } = \frac{\text{Carrier frequency}}{\text{Reference frequency}} \qquad (4.4)$$

Ratios higher than 9 are the best ones for good PWM representations and lowered harmonic distortion (LHD) (Luo *et al.*, 2005). The formula below was used to determine both carrier and sine signal amplitude (Luo *et al.*, 2005; Wilkinson, 2004):

$$m_{\text{a}} = \frac{\text{Reference signal amplitude peak}}{\text{Carrier signal amplitude peak}} \qquad (4.5)$$

$$\text{Reference signal (t)} = m_{\text{a}} \text{ x} \sin(\text{w}_{\text{r}}\, \text{t}) \qquad (4.6)$$

where: m_a is the modulation index

w_r is the phase angle.

The generation of sine waves is performed in the software code. This was achieved with the "sin" function by including the "math.h" header from the C standard library. A modulation index of 0.8 is already set in the code, but can be modified to generate other sine values, if required.

Figure 4.9 shows the entire section responsible for the generation of full sine waves in the DSP Builder's design. Inputs, "out1" and "out2" feed the section with the one quarter of a sine wave's sample values, written by the Nios® II core, which are processed by the rest of this part to produce the required reference signals to be compared with the generated carrier waves.

PWM was also produced as a DC signal is compared to the carrier signal, instead of the sine wave, as stated in Chapter Two.

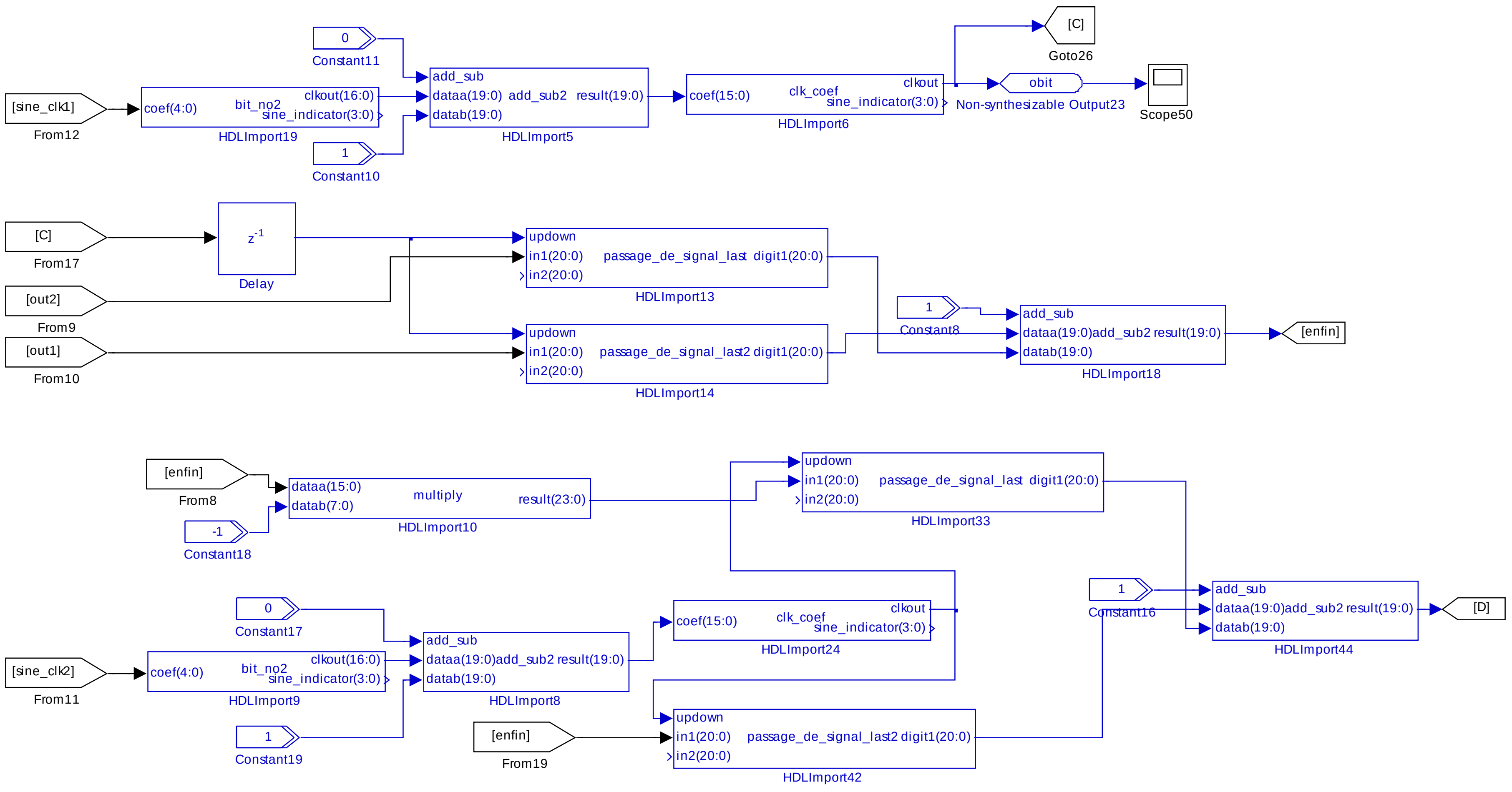


Figure 4.9: Reference (sine) wave signal generation block

More values can be included by modifying the “pre-comparator” VHDL firmware code (see Appendix C) by employing the following formula:

$$\text{DC level } = \text{Maximun clock count value } \text{ x } \text{ duty cycle} \qquad (4.7)$$

Where the maximum clock count value was determined by the 2^n value from (4.8) in the next section.

4.4.1.4 Carrier (triangular) signal generator block

HDL frequency dividers were employed the production of carrier waves. As the SCBC is required to operate on several PE applications, the adjustable frequency divider block acts on the 50 MHz FPGA’s global clock signal to generate carrier signals of adjustable amplitude and frequency. It is controlled by the n coefficient that a user enters, as the generation of carrier waves was based on the formula below:

$$f \text{ carrier} = \frac{f \text{ clock}}{(2^n - 1) \text{ x } 2} \qquad (4.8)$$

Where *f* carrier is the carrier frequency

f clock is the main clock frequency

n is the number of counter bit

2^n is the number of clock count

For instance, for a 50 kHz switching frequency:

$$50\,000 \approx \frac{50\,000\,000}{(2^n - 1) \text{ x } 2} \qquad (4.9)$$

$$(2^n - 1) \approx \frac{50\,000\,000}{100\,000} \qquad (4.10)$$

$$(2^n - 1) \approx 500 \qquad (4.11)$$

According to calculations for approximately 50 kHz, the counter’s resolution to enter must be 9 bits, to ensure that the PWM signal’s resolution is the same. The counter was synchronized with the system's clock and started counting up from 0 to approximately 511, and then back down to 0. This would produce a triangular shape and 1022 steps for every carrier wave cycle. As mentioned above, the maximum value of the counter determines the PWM switching frequency, since it is the same as the carrier frequency. The choice of switching frequency is based on the switching losses in the inverter, the amount of LTD and the properties of the switching devices used (Klumpner, 2004). The carrier waves can run on various switching frequencies, ranging from 833 Hz to approximately 1.6 MHz.

4.4.1.5 Comparator, application selection and signal shift control

A number of MATLAB® GUIs are developed and are available for users to have the option of choosing the PWM switching frequency over a specific range, and the application of the digital controller is required to work on. Parameters are written onto binary files and are read by the soft-processor via the C code. This results in numerical values to be written as inputs to some HDL import modules for carrier signal generation and control, reference signal frequency control, application selection control and signal shift control. This is achieved by the connection of an 8-bit Avalon Write slave interface to a demultiplexer on the DSP Builder design, as mentioned above.

As the reference and carrier signals are generated, they are directly routed to a data comparator block, which is used to support two signals of different magnitudes to process their intersection. This generates one PWM output signal, which is directly routed to another HDL import block, which is involved in the application selection control, where a choice can be made between single phase half bridge, full bridge DC-DC converters or half bridge, full bridge, three phase and multicell inverters. The same block is responsible for signal shift, according to the requested application. As this work focused on the control of the PE applications mentioned above, it was vital to first obtain a simple understanding of their topology and implementation of PWM signals on them. Figures 4.10 - 13 display the topologies of the applications mentioned above. Figure 4.10 displays the single phase half bridge converter topology, which is the simplest one and consists of only one pair of complimentary switches. Each switches' gates are supplied with PWM signals that are represented by the "S1a" and "S1b" PWM elements.

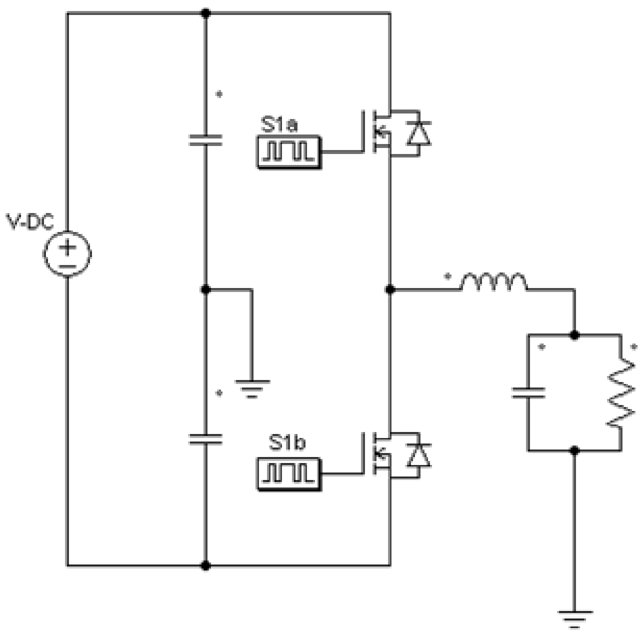


Figure 4.10: Half bridge converter topology

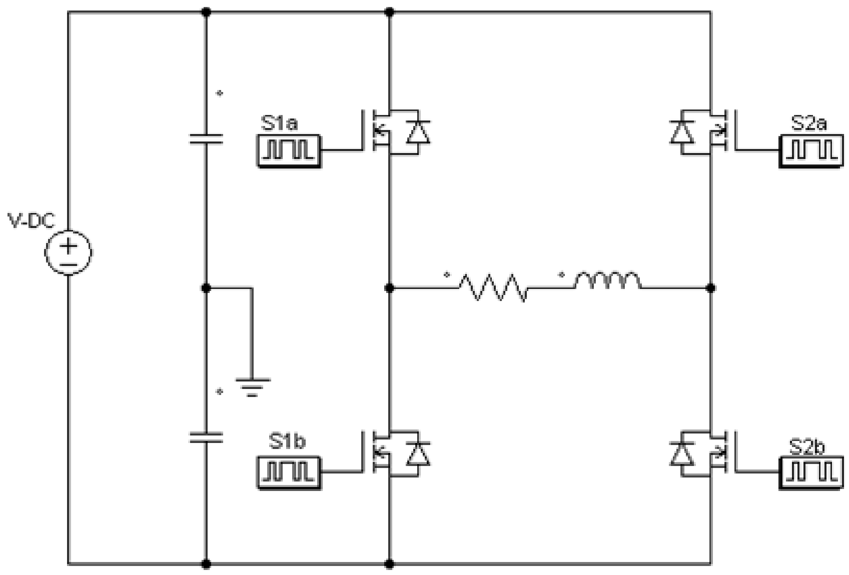


Figure 4.11: Full bridge converter topology

Figure 4.11 displays the full bridge converter topology, which consists of two complimentary switches. Two pairs of PWM signals are delivered to the switches and are phase-shifted by 2 $\times \frac{pi}{2}$ radians (180°) from one another.

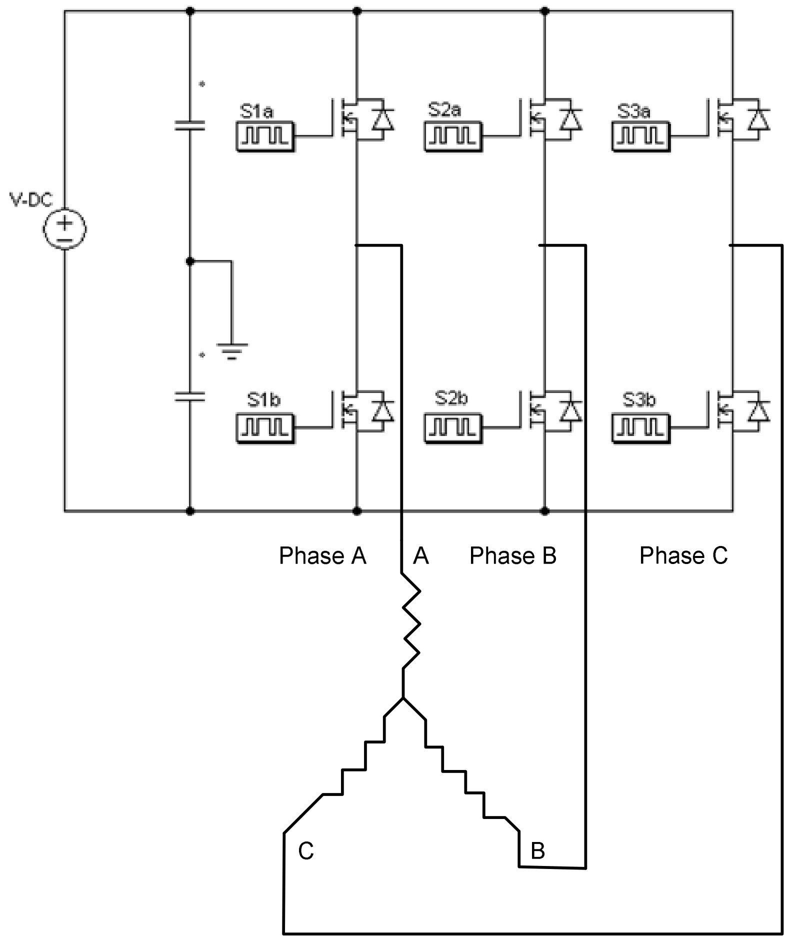


Figure 4.12: Three phase inverter topology

The three phase converter has three switches pairs, as displayed in Figure 4.12. Three reference and one carrier signals are necessary for this P.E application operation. Reference signals are each phase-shifted from one another by $2 \times \frac{pi}{3}$ radians (120 °).

The number of PWM signals delivered to multicell inverters depends on the number of cells that they incorporate. For one cell, two PWM signals are required. This work focused mostly on the 5-cell inverter, since this topology is already available at the CIR laboratory. In this instance, ten PWM signals were required as five complimentary switch pairs were involved in the application's operation, as shown by Figure 4.13 below. For realization of PWM production for the inverter, five carrier waves were required and were phase-shifted by 1.257 radians, namely 72° (Molepo, 2003).

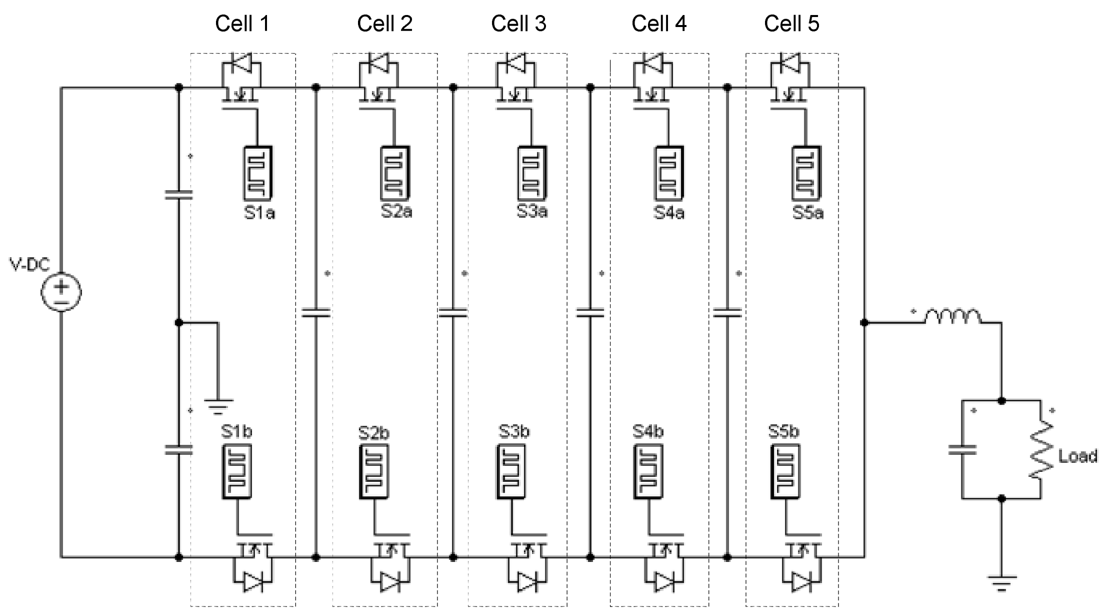


Figure 4.13: 5-cell inverter topology

4.4.1.6 Deadtime control

Another HDL block was inserted to complete the PWM production process. It consisted of an adjustable deadtime block created and implemented to prevent the semiconductors' complimentary switching signals to go on and off at the same time. This prevents system damage caused by short circuits. The controller user has the option of choosing between 40, 80, 100, 160 and 200 ns deadtime. A range of different deadtime values are necessary since the controller runs at different frequencies. This would cause some signals to be destroyed whenever too much deadtime is implemented. Another reason is the compensation for different rise and fall times from different switching semiconductor devices, which can be accommodated.

Figure 4.14 shows the entire section that is responsible for the comparison of reference and carrier signals and the production of PWM switching signals in the DSP Builder's design. Inputs, "out1" and "out2" feed the section with the one quarter of a sine wave's sample values, written by the Nios® II core and these are processed by the rest of this part to produce the required reference signals that are compared with the generated carrier waves.

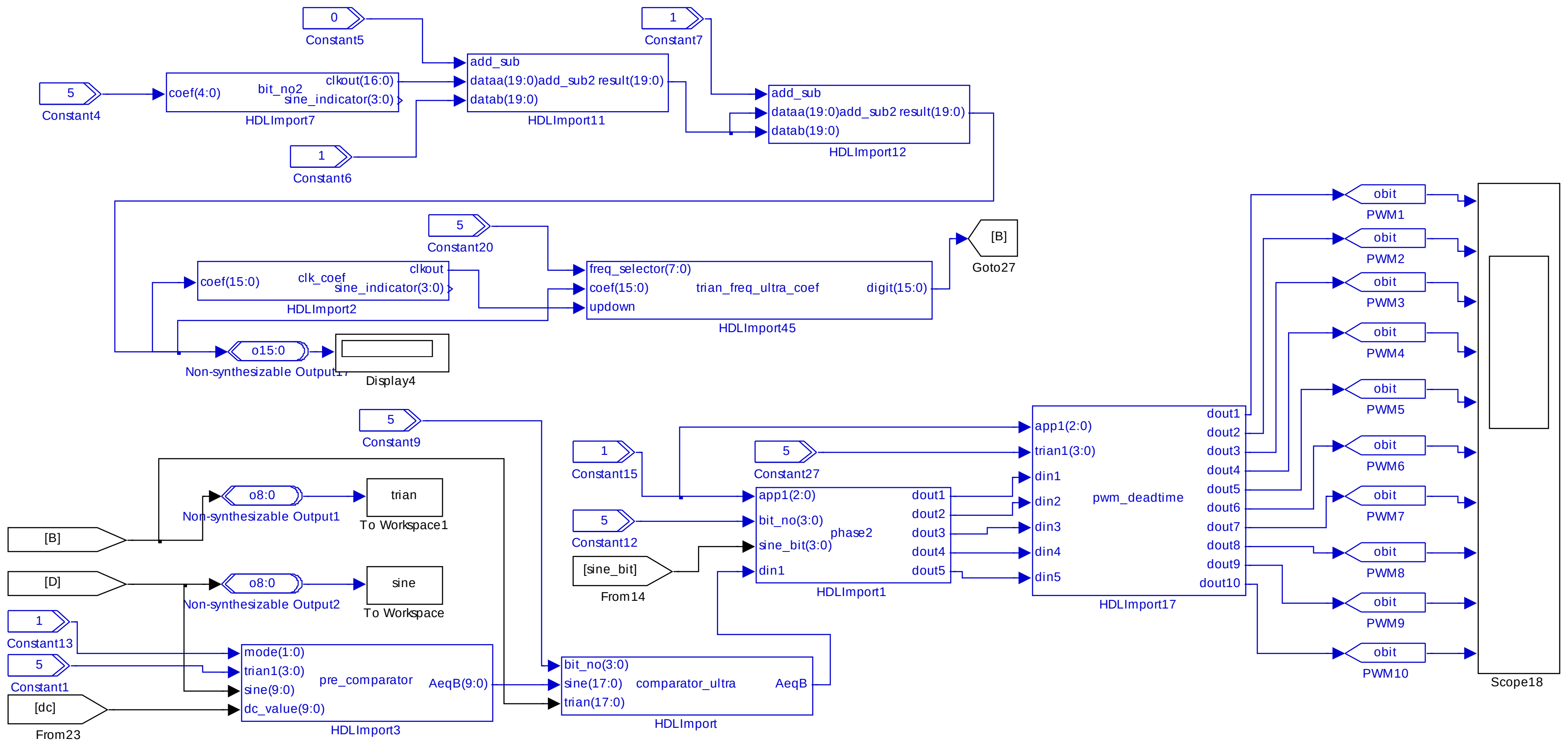


Figure 4.14: Carrier (triangular) signal and PWM generators

The “pre_comparator” and “comparator_ultra” HDL imports make up the comparator, “phase2”, and involve the control of signal phase, while “pwm_deadtime” controls the PWM’s deadtime. The numbered HDL output elements, from PWM1 to PWM10 on the far right side, directly send the PWM signals to the FPGA’s output pins to the HSMC connector.

4.4.2 Stage 2 – SOPC creation

After satisfactory development of the PWM modulator on the DSP Builder/Simulink model and the generation of the SignalCompiler block output files, the SOPC Builder tool was used to advance further in the development of the overall project.

Once the DSP Builder design compilation was completed, its logic design had to be instantiated as a custom peripheral to the soft-processor in SOPC Builder. For this to be accomplished, a new top level project was created where the logic design was included in the form of Tcl scripts in Quartus® II project, generated from SignalCompiler in DSP Builder. The DSP Builder design and all the necessary hardware peripherals are assembled into the SOPC system, which makes up the SCBC. Following system generation, the rest of the embedded processor system was completed by using the standard Nios® II development design flow. The different hardware modules of the controller system are described below:

- **CPU**

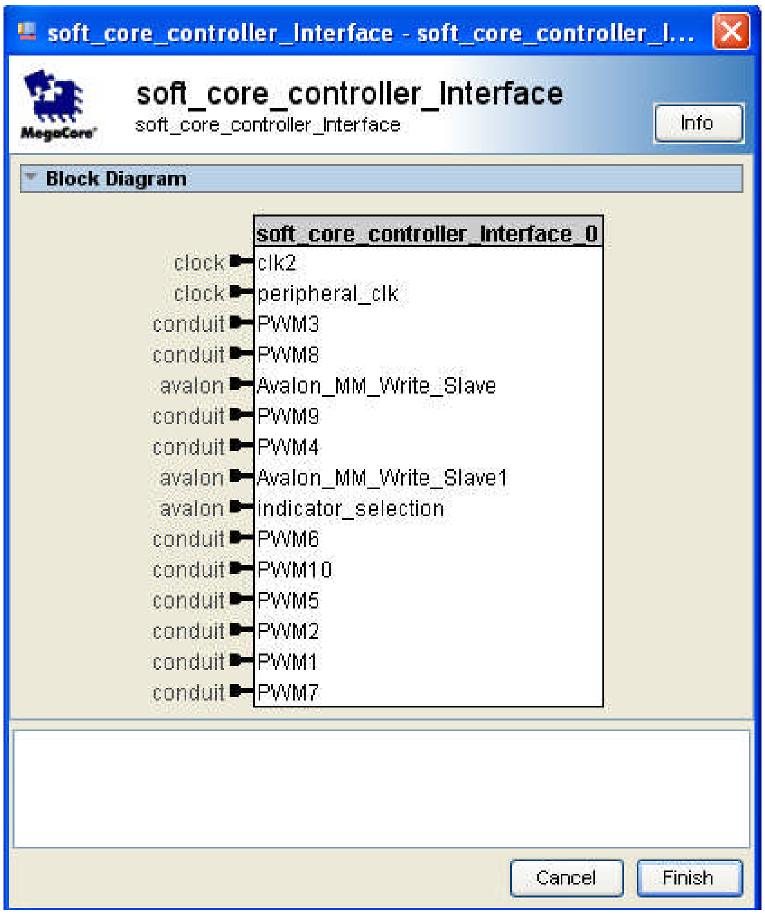
  Communication between custom hardware components and the CPU on which the software runs is best done through the Avalon Interconnect Fabric bus. The Nios® II CPU acts as a master peripheral on the Avalon bus, and initializes the actual transfer of data with slave peripherals. During a read transfer, the CPU transfers data from the hardware peripheral to the CPU, and during a write transfer, data is transferred from the CPU to the hardware peripheral (Sicking, 2005). Since the SCBC’s operation requires large amounts of data and code and arithmetic calculations, the fast (Nios® II/f) core configuration was used.

- **Sysid**

  The Altera® System ID provides a SOPC Builder system with a unique identifier, as it verifies that the system compilation targets a specific board.

- **“Soft_core_controller_Interface_0” custom peripheral**

  The **“**Soft_core_controller_Interface_0**”** is the custom peripheral, which is created as the DSP Builder that involves hardware implementation of the digital controller. The “soft_core_controller_Interface_0**”** is attached to the Nios® II system as a co-processor. The module involves the three Avalon write interfaces, carrier signal, sine wave, transfer function, shift and delay, comparator and PWM blocks.
  The main advantage of custom peripherals is their fast data transfer compared to the one provided by PIOs.


soft_core_controller_Interface - soft_core_controller_I...
soft_core_controller_Interface
soft_core_controller_Interface
Info
Block Diagram
Cancel
Finish

soft_core_controller_Interface_0

clock	clk2
clock	peripheral_clk
conduit	PWM3
conduit	PWM8
avalon	Avalon_MM_Write_Slave
conduit	PWM9
conduit	PWM4
avalon	Avalon_MM_Write_Slave1
avalon	indicator_selection
conduit	PWM6
conduit	PWM10
conduit	PWM5
conduit	PWM2
conduit	PWM1
conduit	PWM7

Figure 4.15: Input and output signals from the DSP Builder design- turned SOPC peripheral interface.

Figure 4.15 shows and lists all the clock, input and output signals involved in the DSP Builder design. PWM1 – 10 signals are the outputs of the PWM generator block and of the SCBC.
“Avalon_MM_Write_Slave”, “Avalon_MM_Write_Slave1” and “indicator_selection” are the Avalon Write slaves address signals, while “peripheral_clk” and “clk2” are the operating clocks. They all become part of the SOPC system after the model compilation via SignalCompiler.

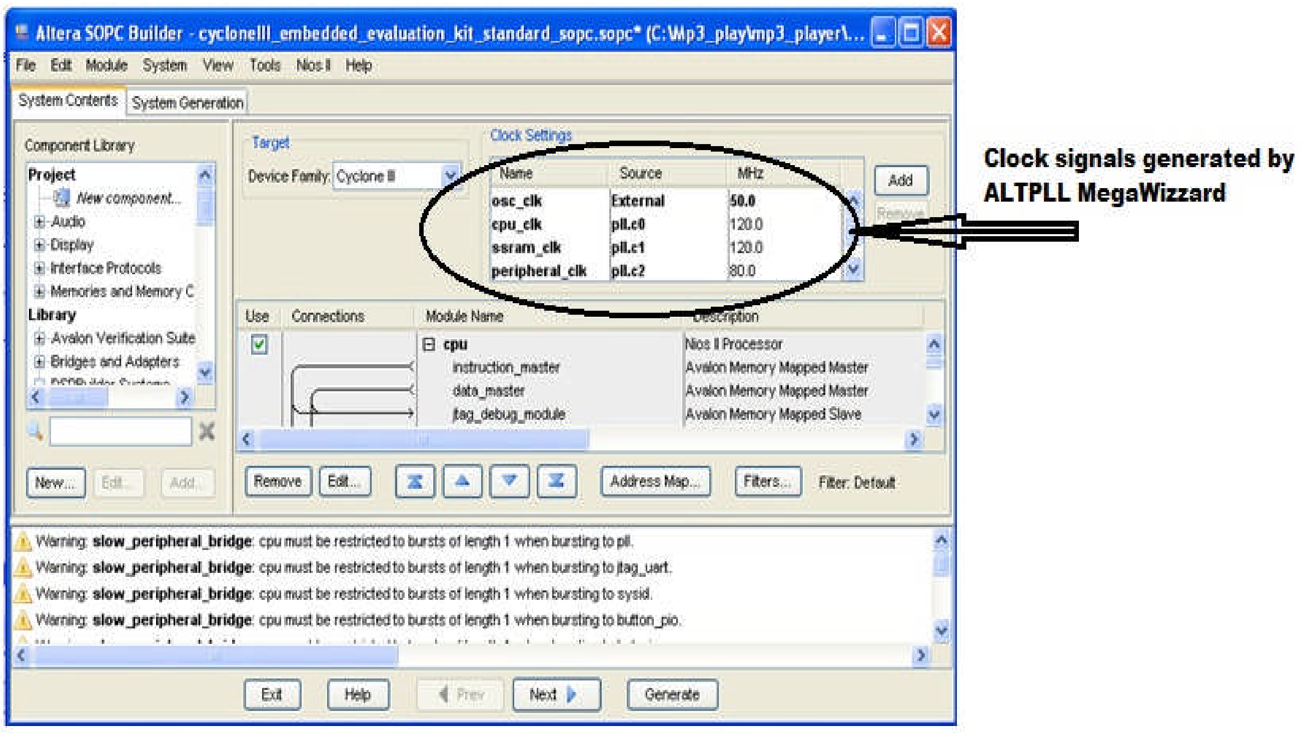
- **Off-chip memory – SRAM and FLASH memories**

Memory is vital for any embedded system. The design of the controller required the use of file management C commands and DSP Builder model, consisting of several VHDL elements. These occupy a lot of memory space in the FPGA. In this work off-chip memory was used owing to their high storage capabilities. It is used to store, provide access to, and allow modification of data and program code to run within the Nios® II processor and program code used by the entire SOPC system.

The off-chip memory consists of the Cypress® CY7C1380 SSRAM and CFI FLASH memories. The SSRAM is configured with a memory capacity of 1 Mbyte and stores the software program code, stack, and exception sections. The flash memory is a non-volatile memory, which retains the program’s contents when the FPGA’s power is off. The flash memory device is configured with a memory capacity of 16 Mbytes, and stores Nios® II application software, and the CPU reset vector (Altera Corporation, 2010e).

- **PHASE LOCKED LOOP (PLL)**

The Nios® II system usually consists of different modules, which operate at different frequencies. Hence, Altera® ATLPLL MegaWizard Plug-In-Manager was used in order to specify and generate other frequencies, apart from the 50 MHz main frequency in this work. This helped to generate various sampling rates to synchronise the inputs and outputs of the digital controller. Figure 4.16 displays all the operating clocks of the overall Nios® II system. “Osc_clk” represents the global clock signal, “cpu_clk**”** and “ssram_clk**”** are clock signals that are required for the SSRAM and other modules, and “peripheral_clk” for the “soft_core_controller_Interface_0” peripheral. “Peripheral_clk” can be modified to meet requirements in terms of frequency range, simply by opening the MegaWizard Plug-In-Manager in SOPC Builder for configuration.


Altera SOPC Builder - cycloneIII_embedded_evaluation_kit_standard_sopc.sopc* (C:\Mp3_play\mp3_player\...

Target: Device Family: Cyclone III

Clock Settings

Name	Source	MHz
osc_clk	External	50.0
cpu_clk	pll.c0	120.0
ssram_clk	pll.c1	120.0
peripheral_clk	pll.c2	80.0

Use	Connections	Module Name	Description
✓		cpu	Nios II Processor
		instruction_master	Avalon Memory Mapped Master
		data_master	Avalon Memory Mapped Master
		jtag_debug_module	Avalon Memory Mapped Slave

Filter: Default

```
Warning: slow_peripheral_bridge: cpu must be restricted to bursts of length 1 when bursting to pll.
Warning: slow_peripheral_bridge: cpu must be restricted to bursts of length 1 when bursting to jtag_uart.
Warning: slow_peripheral_bridge: cpu must be restricted to bursts of length 1 when bursting to sysid.
Warning: slow_peripheral_bridge: cpu must be restricted to bursts of length 1 when bursting to button_pio.
```

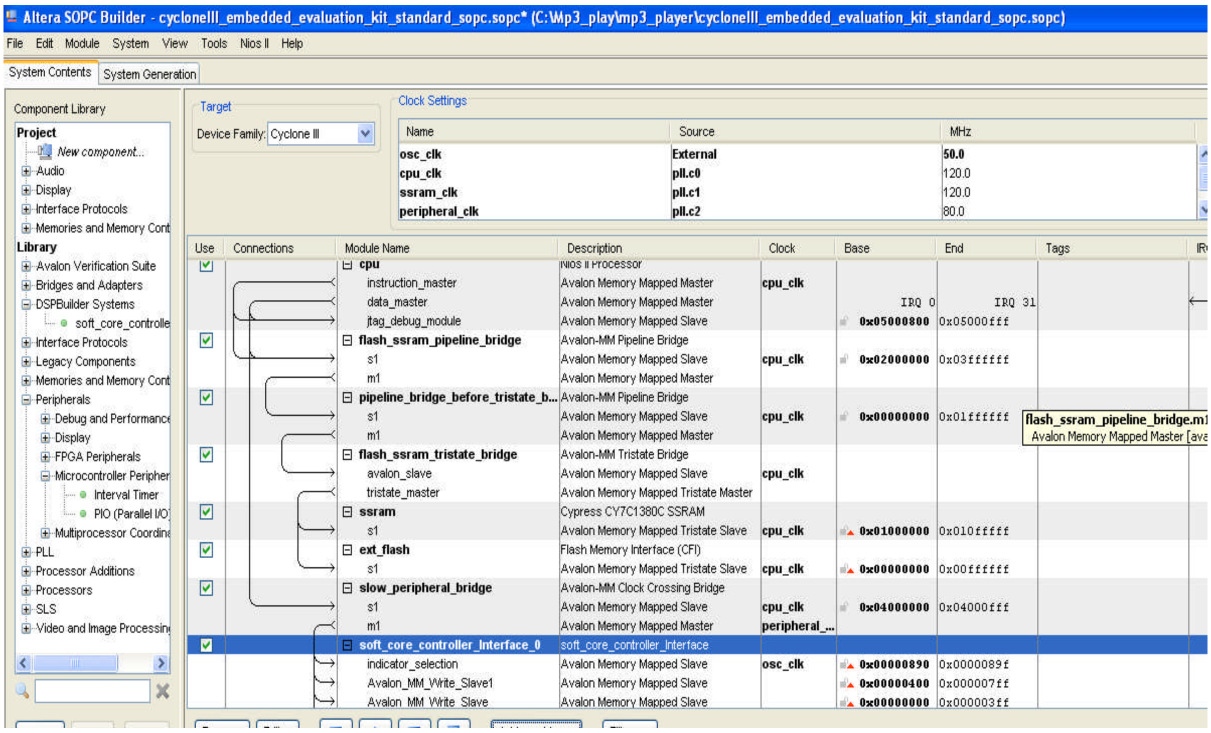
Figure 4.16: PLL clock signals integrated into the SOPC system

- **JTAG UART**

  The Altera® JTAG UART core provides data communication between a PC host and the SOPC Builder system. The host PC can connect to the FPGA® via any Altera® JTAG download cable such as the USB-Blaster cable or Blyster cable. This core enables software files’ downloading, running and debugging by either using the Nios® II integrated development environment (IDE) debugger or the SignalTap II Logic Analyzer (Altera Corporation, 2010d).

Figure 4.17 shows all the hardware modules that are involved in the development of the controller’s complete Nios® II SOPC system, its modules connections and clock settings.

The section highlighted in blue is the custom peripheral, **“Soft_core_controller_Interface_0”,** which was created by DSP Builder.



Device Family: Cyclone III

Name	Source	MHz
osc_clk	External	50.0
cpu_clk	pll.c0	120.0
ssram_clk	pll.c1	120.0
peripheral_clk	pll.c2	80.0

Use	Connections	Module Name	Description	Clock	Base	End	Tags
✓		cpu	Nios II Processor				
		instruction_master	Avalon Memory Mapped Master	cpu_clk			
		data_master	Avalon Memory Mapped Master		IRQ 0	IRQ 31	
		jtag_debug_module	Avalon Memory Mapped Slave		0x05000800	0x05000fff	
✓		flash_ssram_pipeline_bridge	Avalon-MM Pipeline Bridge				
		s1	Avalon Memory Mapped Slave	cpu_clk	0x02000000	0x03ffffff	
		m1	Avalon Memory Mapped Master				
✓		pipeline_bridge_before_tristate_b...	Avalon-MM Pipeline Bridge				
		s1	Avalon Memory Mapped Slave	cpu_clk	0x00000000	0x01ffffff	
		m1	Avalon Memory Mapped Master				
✓		flash_ssram_tristate_bridge	Avalon-MM Tristate Bridge				
		avalon_slave	Avalon Memory Mapped Slave	cpu_clk			
		tristate_master	Avalon Memory Mapped Tristate Master				
✓		ssram	Cypress CY7C1380C SSRAM				
		s1	Avalon Memory Mapped Tristate Slave	cpu_clk	0x01000000	0x010fffff	
✓		ext_flash	Flash Memory Interface (CFI)				
		s1	Avalon Memory Mapped Tristate Slave	cpu_clk	0x00000000	0x00ffffff	
✓		slow_peripheral_bridge	Avalon-MM Clock Crossing Bridge				
		s1	Avalon Memory Mapped Slave	cpu_clk	0x04000000	0x04000fff	
		m1	Avalon Memory Mapped Master	peripheral_...			
✓		soft_core_controller_Interface_0	soft_core_controller_Interface				
		indicator_selection	Avalon Memory Mapped Slave	osc_clk	0x00000890	0x0000089f	
		Avalon_MM_Write_Slave1	Avalon Memory Mapped Slave		0x00000400	0x000007ff	
		Avalon_MM_Write_Slave	Avalon Memory Mapped Slave		0x00000000	0x000003ff	

Figure 4.17: Hardware configuration of the controller’s Nios® II system

4.4.3 Stage 3 – Quartus® II system completion

This stage involved the completion of the overall project by adding the created SOPC system to the top-level Quartus® II project. It was done through compilation as analysis, synthesis, fitting, place-and-route, and timing analysis were performed. As the system was generated in SOPC Builder, input and output signals, from the DSP Builder design’s hardware modules, were exported to the top level system files. They became PIO input and output signals as pin outs were assigned to them via the Quartus® II Assignment Editor. The Register Transfer Level (RTL) of the system design is first checked visually via Quartus® II RTL Viewer before any compilation to ensure all the necessary logic and connections are included in the system for its full functioning. Figure 4.18 displays the generated RTL viewer diagram of the digital controller with the SCBC‘s outputs, which are the PWM switching signals on the upper left hand side.

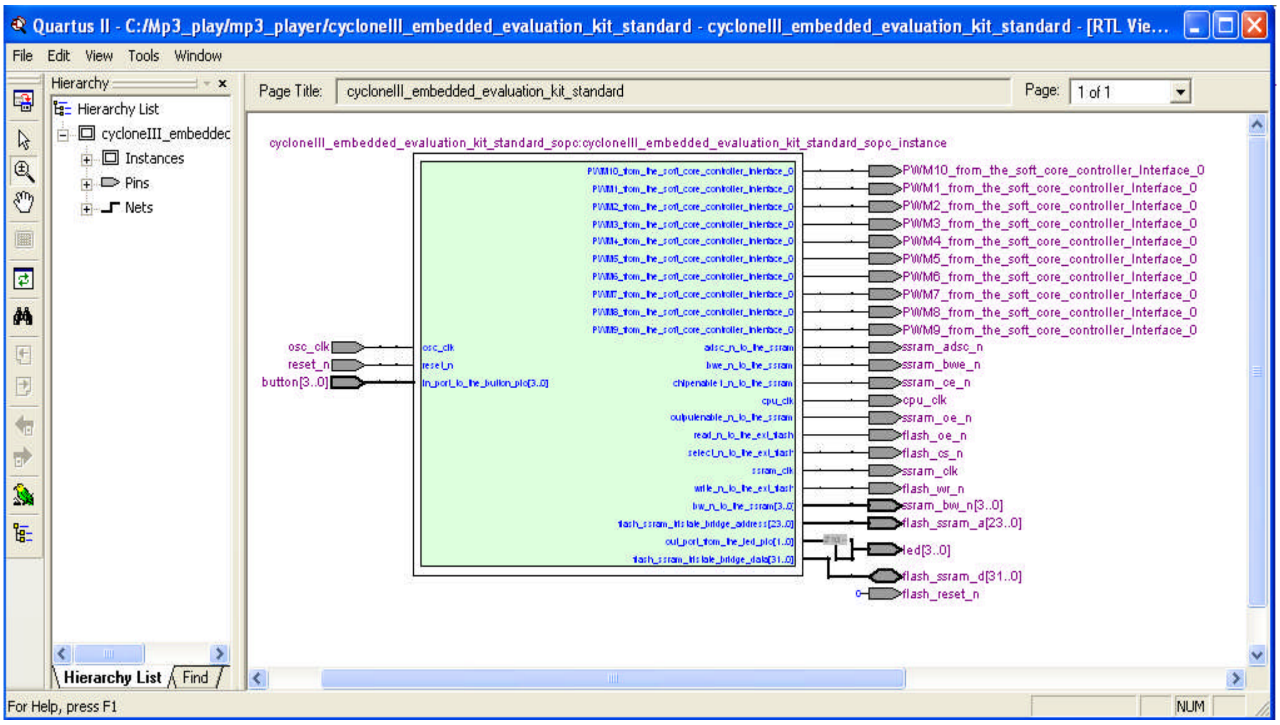

Quartus II - C:/Mp3_play/mp3_player/cycloneIII_embedded_evaluation_kit_standard - cycloneIII_embedded_evaluation_kit_standard - [RTL Vie...
File Edit View Tools Window
Hierarchy
Hierarchy List
cycloneIII_embeddec
Instances
Pins
Nets
Page Title: cycloneIII_embedded_evaluation_kit_standard
Page: 1 of 1
cycloneIII_embedded_evaluation_kit_standard_sopc:cycloneIII_embedded_evaluation_kit_standard_sopc_instance
osc_clk
reset_n
button[3..0]
PWM10_from_the_soft_core_controller_Interface_0
PWM1_from_the_soft_core_controller_Interface_0
PWM2_from_the_soft_core_controller_Interface_0
PWM3_from_the_soft_core_controller_Interface_0
PWM4_from_the_soft_core_controller_Interface_0
PWM5_from_the_soft_core_controller_Interface_0
PWM6_from_the_soft_core_controller_Interface_0
PWM7_from_the_soft_core_controller_Interface_0
PWM8_from_the_soft_core_controller_Interface_0
PWM9_from_the_soft_core_controller_Interface_0
ssram_adsc_n
ssram_bwe_n
ssram_ce_n
cpu_clk
ssram_oe_n
flash_oe_n
flash_cs_n
ssram_clk
flash_wr_n
ssram_bw_n[3..0]
flash_ssram_a[23..0]
led[3..0]
flash_ssram_d[31..0]
flash_reset_n
Hierarchy List
Find
For Help, press F1
NUM

Figure 4.181: RTL diagram of the soft-core based controller’s Nios® II system

These signals, from the DSP Builder project, are connected to the FPGA host High Speed Mezzanine Connector (HSMC) pins through the Quartus® II Assignment Editor. Through this process, they became PIO signals which are directly controlled by the software components of the project.

This stage is concluded by the programming of the FPGA chip, as a (.sof) SRAM object file is produced after the Quartus® II project compilation and downloaded on the FPGA board. A compilation report is also generated and displays all the hardware resource usage of the Quartus® II project design. As Figure A.6 shows (see Appendix A), the Cyclone® 3C25F324C8 FPGA has sufficient resources to sustain the chosen RAM capacity in the design. Only about 55% memory bits were used to allow for hardware expansion. This provided enough memory space for the .elf file to be generated in Nios® II IDE.

4.4.4 Stage 4 - MATLAB® GUI design and implementation

MATLAB® GUIs were developed as additional peripherals to the Nios® II system of the SCBC. Some of these help the user to control the SCBC inputs and outputs. They help in saving time and the need to do all the calculations that are required, while modifying the C codes on the Nios® II IDE each time change is required. Figures 4.19 – 21 present the control GUIs. They are associated with application selection control, carrier and reference signal frequency control, deadtime and duty cycle control. They serve as a communication interface between the user and the SCBC, via MATLAB®, the DSP Builder design and the Nios® II processor, as they replace the absence of a UART component on the FPGA board. Users can enter numerical values, which represent signal parameters into the C code running on the Nios® II processor via the GUIs by selecting radiobuttons. As new configuration values are entered and loaded into the FPGA design from the processor, application selection and frequencies are changed, according to the new values. New configuration data can only be loaded and read after re-running the Nios® II system project on the IDE each time that change is implemented.

The GUI in figure 4.19 represents the communication interface to the Nios® II core. It consists of three panels of radiobuttons that represent the four power electronic applications, and the carrier and reference signal frequencies. Figure 4.20 shows the GUI that is responsible for duty cycle selection, while the one from Figure 4.21 involves deadtime selection.

As a radiobutton is selected, input data is sent and read by the processor, which in turn sends off control data and generates sine wave samples to the rest of the logic design. Some of the MATLAB® .m files were created in order to write numerical values onto binary and text files following the PC host based file system. Pushbuttons are also available for converter and inverter mode, as well as deadtime selection.

Other GUIs such as those that are one presented in Figure 4.22 are developed as DSP Builder simulation platforms as graphing and monitoring tools. These GUIs display signals that are stored in the Simulink ”To Workspace” blocks, which are inserted on the DSP Builder design. In turn, these signals are simulated and displayed in MATLAB® to monitor their validity against their validity in VHDL. MATLAB® GUIs development involves two parts, the MATLAB® .m code and the graphic skin. The .m code actively communicates with binary files by writing numerical values onto them.

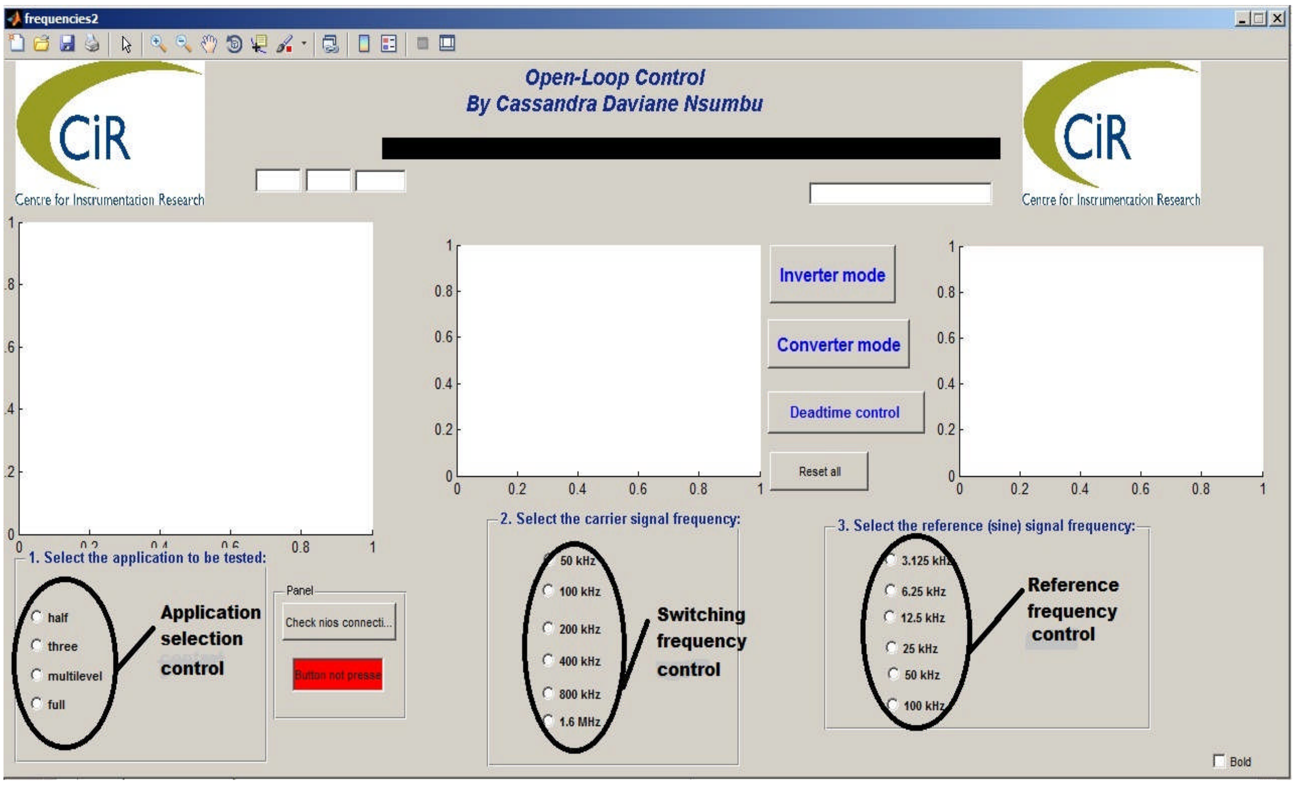

frequencies2
Open-Loop Control
By Cassandra Daviane Nsumbu
CiR
Centre for Instrumentation Research
Inverter mode
Converter mode
Deadtime control
Reset all
1. Select the application to be tested:
half
three
multilevel
full
Application selection control
Panel
Check nios connecti...
Button not presse
2. Select the carrier signal frequency:
50 kHz
100 kHz
200 kHz
400 kHz
800 kHz
1.6 MHz
Switching frequency control
3. Select the reference (sine) signal frequency:
3.125 kHz
6.25 kHz
12.5 kHz
25 kHz
50 kHz
100 kHz
Reference frequency control
Bold

Figure 4.19: MATLAB® GUI for application, frequency ratio and carrier (switching) frequency selection control

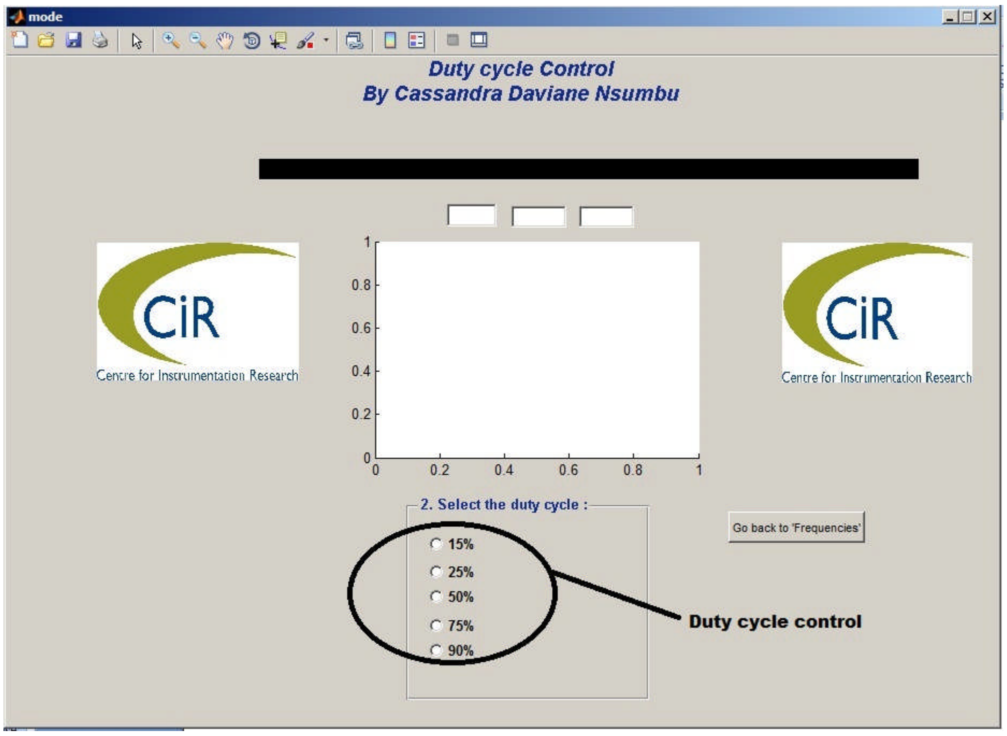

mode
Duty cycle Control
By Cassandra Daviane Nsumbu
CiR
Centre for Instrumentation Research
2. Select the duty cycle :
15%
25%
50%
75%
90%
Duty cycle control
Go back to 'Frequencies'

Figure 4.20: MATLAB® GUI for duty cycle control

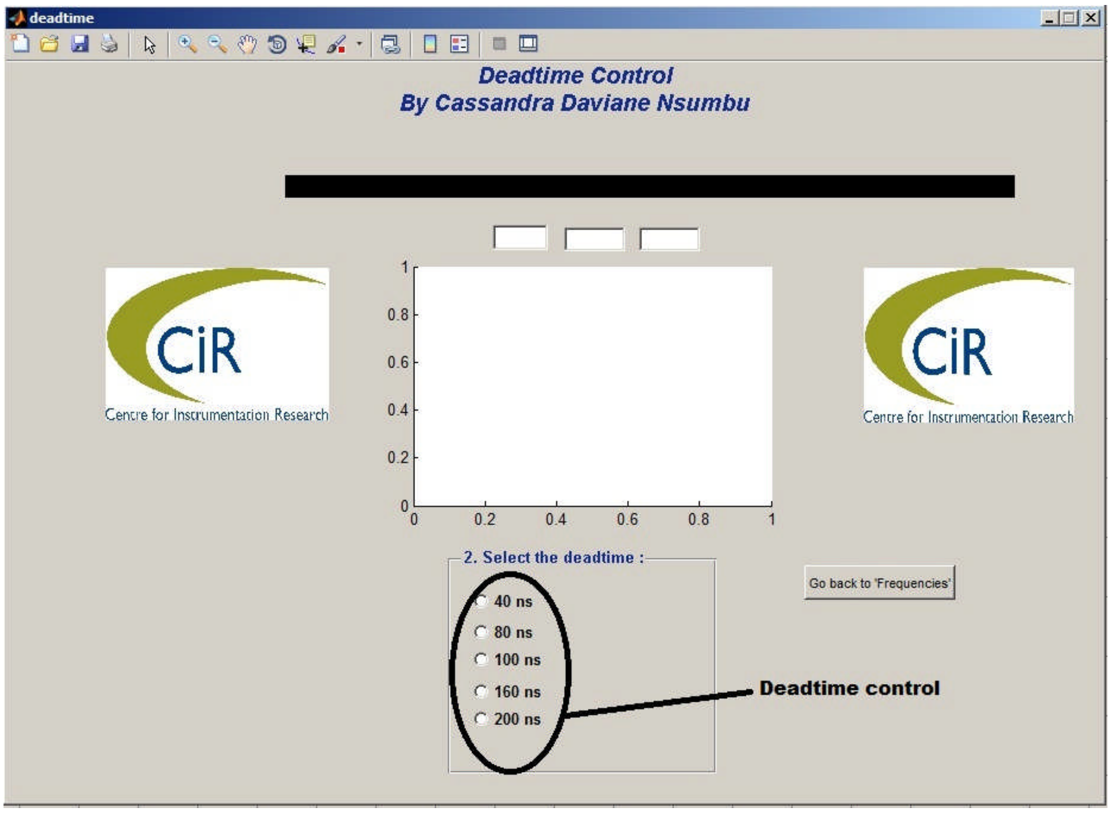


Figure 4.21: MATLAB® GUI for deadtime control

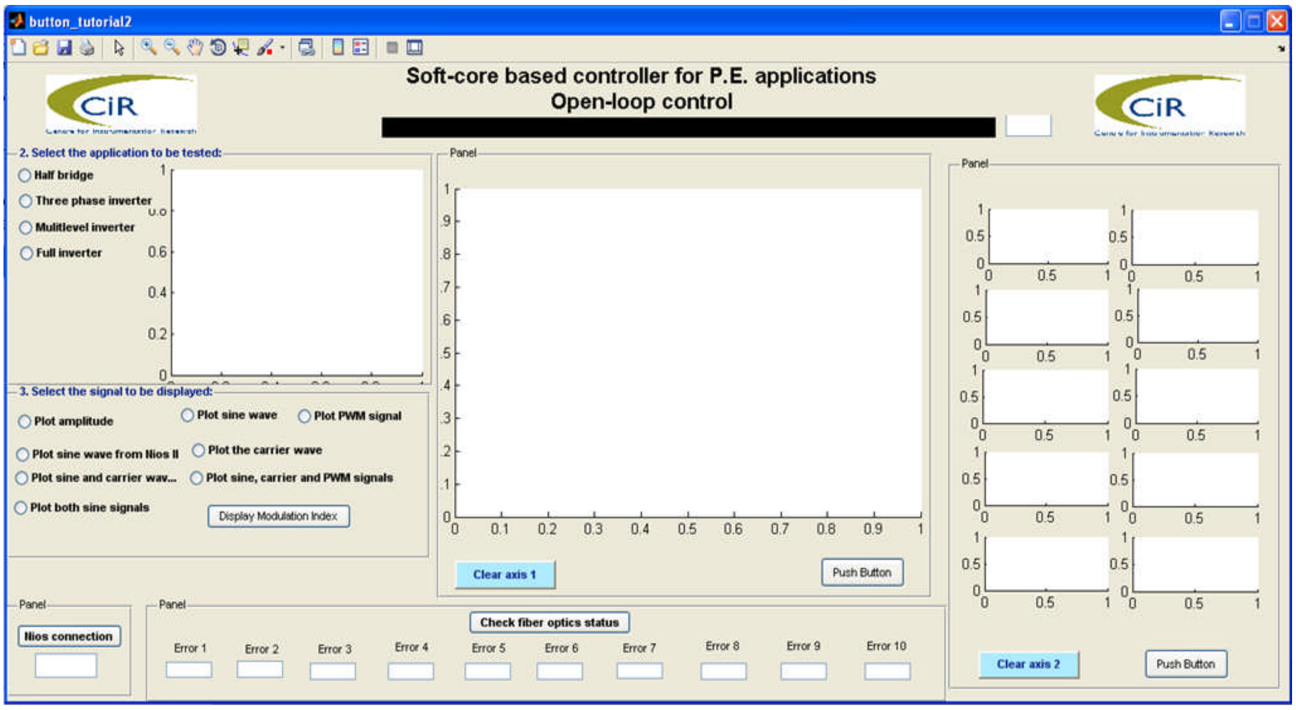


Figure 4.22: MATLAB GUI for simulation (5-cell inverter)

4.4.5 Stage 5 – Software code development

The fifth stage involves C code development and compilation. Software implementation is advantageous in saving the hardware area and in rapidly performing complex functions that are more difficult to implement in hardware by involving a processor and some memory.

Hence, a software code was implemented to control the interaction and operation of the SOPC hardware components via the Nios® II processor. The other function of the processor involves sine waves generation on the Nios® II IDE, and generation of additional logic control, which is necessary for PWM production and file handling to communicate with binary and text files that are contained by a PC host-based file system via the RS232 communication interface. This Nios® II software intercommunication is illustrated in Figure 4.23 below.

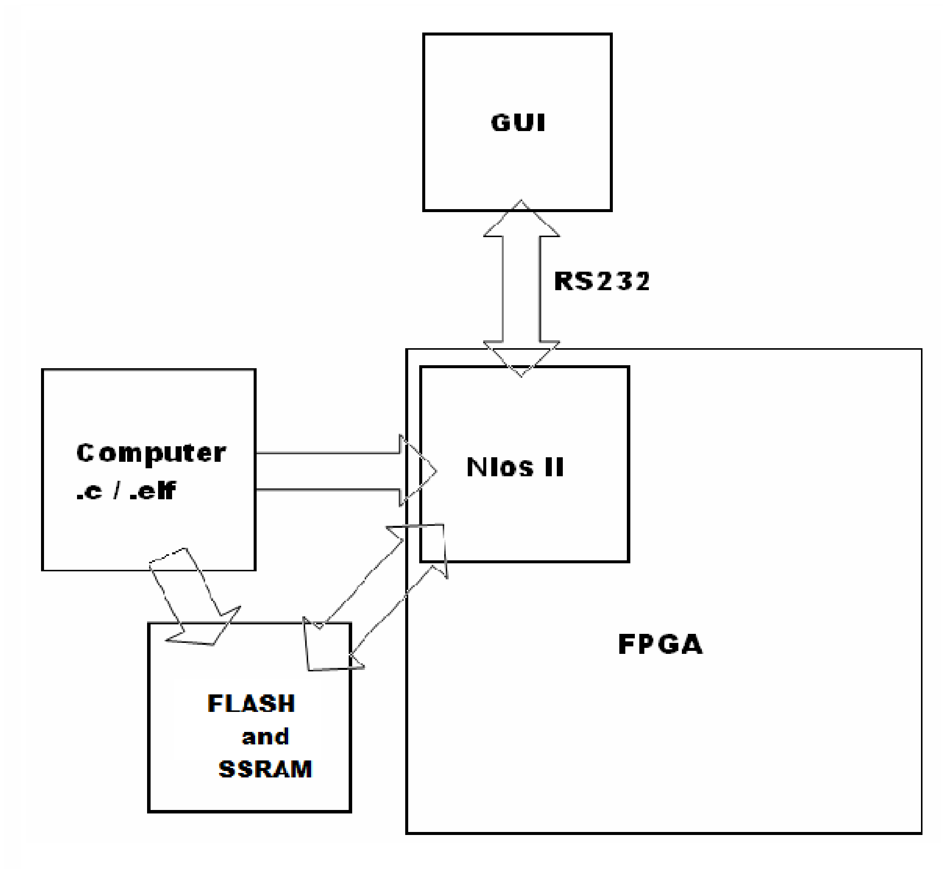


Figure 4.23: Nios® II IDE design flow (Hong, 2008)

4.4.5.1 Source C code design

Two C codes were created to control the Nios® II processor in the system. The first one handles the sine wave samples generation as signal parameters are established, while the other one deals with writing these same sine wave samples into the SSRAM. The second code contains one " int main" function, which calls all the subfunctions, which are categorized into several main functions: Code initialization, reading of binary and text files from the PC host file system, writing of sine samples into SSRAM memory, carrier wave production, signal frequency, application selection and PWM production control.

Initialization

Initialization functions initialize all peripherals and parameters to the Nios® II processor to be used for the digital controller operation.

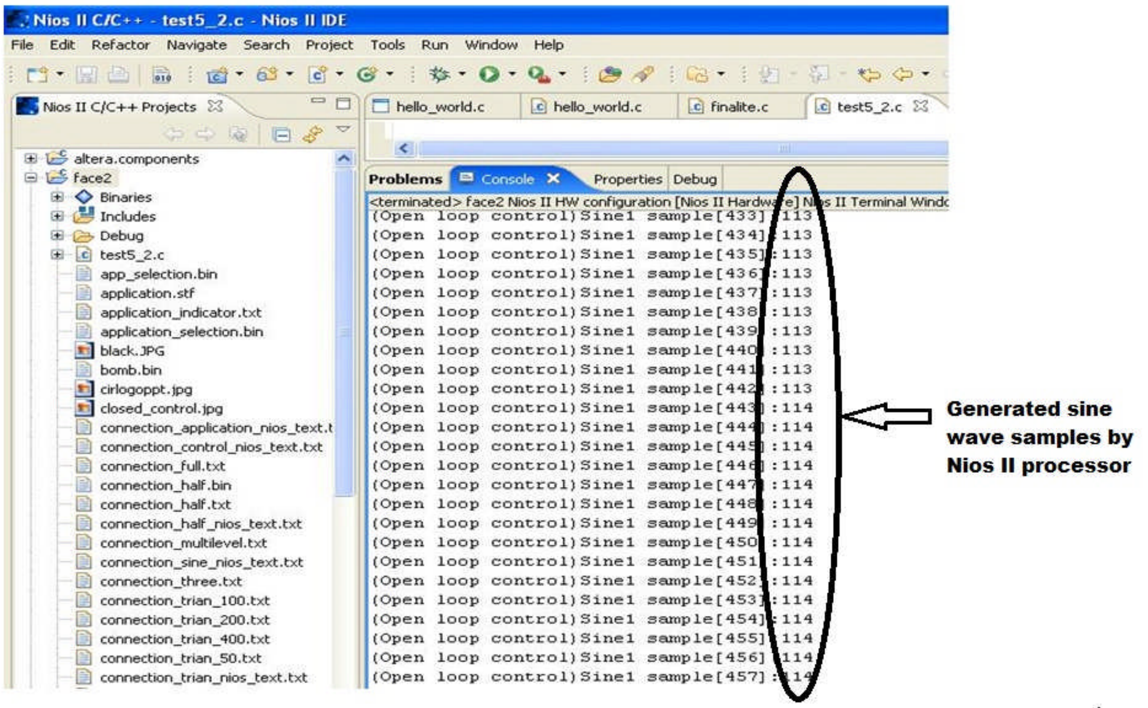
PC host-based file system

This work made use of an Altera®'s GNU debugger filing system, as it offers a simpler alternative way of communication between the Nios® II core and the digital controller user. Parameter values are entered into MATLAB® GUIs, which in turn are written into binary and text files. File handling commands such as "fopen", "fscanf", "fread" and "fclose" are each employed in order to make binary and text files from the PC – Host File System possible. However, these commands were employed as little as possible by having a few files to read from, since such commands require a large amount of memory bits.

Reference wave generation

The Nios® II processor generates the sine wave samples into an array by performing float type calculations with the mathematical sin function from the included "math.h" library. As Appendix D shows, parameters such as "MAX_NUM" represents the number of samples that is required to generate the sine wave; "h" is its offset and "f" is the carrier wave amplitude multiplied by the modulation index then subtracted by the offset. Two arrays "buff[i]" and "buff2[i]" generate the required sine samples values for two separate RAM components and print them on the IDE console window.

Figure 4.24 below displays the sine wave sample values generated by the processor, as they are also displayed on the IDE console after the "printf" command was utilized to display them. The sine samples are extracted from the arrays and converted into sine tables in the form of arrays. Since the processor generates sine samples for sine signals of varying known amplitudes according to frequency ranges, several sine tables of similar magnitude were created. As mentioned above, the lowest reference frequency is taken as the reference frequency from which the design operates. It also provides a higher quality signal, as this frequency presents the highest number of different sample values. As higher frequencies are required, the samples values are divided by the "mul_div" parameter in the "for" loop.



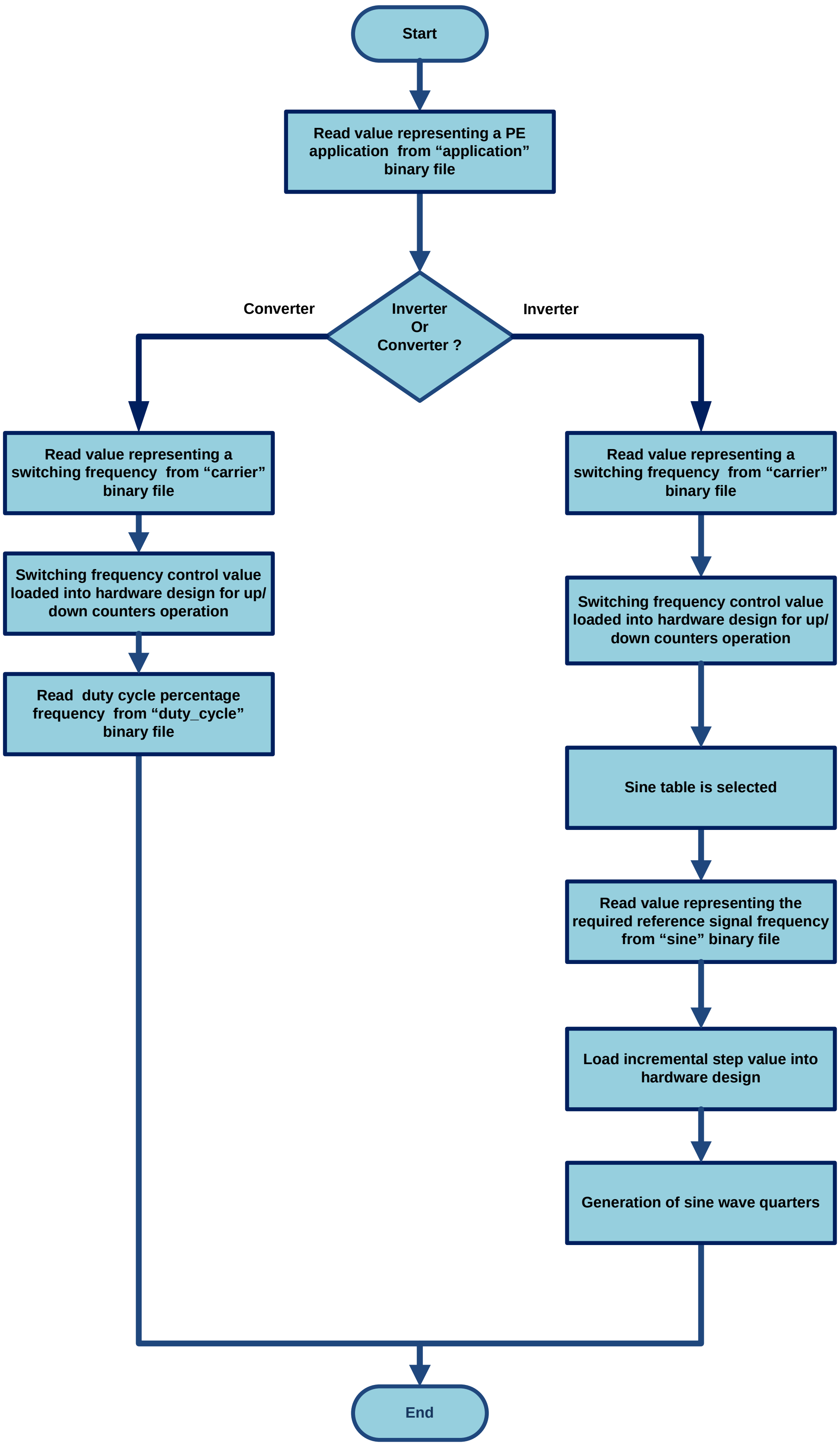
```
<terminated> face2 Nios II HW configuration [Nios II Hardware] Nios II Terminal Windc
(Open loop control)Sine1 sample[433]:113
(Open loop control)Sine1 sample[434]:113
(Open loop control)Sine1 sample[435]:113
(Open loop control)Sine1 sample[436]:113
(Open loop control)Sine1 sample[437]:113
(Open loop control)Sine1 sample[438]:113
(Open loop control)Sine1 sample[439]:113
(Open loop control)Sine1 sample[440]:113
(Open loop control)Sine1 sample[441]:113
(Open loop control)Sine1 sample[442]:113
(Open loop control)Sine1 sample[443]:114
(Open loop control)Sine1 sample[444]:114
(Open loop control)Sine1 sample[445]:114
(Open loop control)Sine1 sample[446]:114
(Open loop control)Sine1 sample[447]:114
(Open loop control)Sine1 sample[448]:114
(Open loop control)Sine1 sample[449]:114
(Open loop control)Sine1 sample[450]:114
(Open loop control)Sine1 sample[451]:114
(Open loop control)Sine1 sample[452]:114
(Open loop control)Sine1 sample[453]:114
(Open loop control)Sine1 sample[454]:114
(Open loop control)Sine1 sample[455]:114
(Open loop control)Sine1 sample[456]:114
(Open loop control)Sine1 sample[457]:114
```

Figure 4.24: Sine waves sample values generated and displayed on the Nios® II IDE console

In order to generate sine wave samples and load them directly into Avalon slave interfaces, custom instructions should be incorporated into the Nios® II CPU. It will guarantee the production of adequate PWM signals, as these instructions accelerate time-consuming software algorithms. Their absence will result in timing delays and, subsequently, in inadequate sine wave generation, however, also occupy a large amount of memory bits in the system. Since a few simple arithmetic operations such as multiplication and division were performed by the processor, and the "sin" trigonometric function posed a problem, and there was an absence of such instructions, the sine table approach was adopted in this work and another C code was required to run on the processor. Figure 4.25 shows the design flow chart of the second C code.

The sine tables' content was then written into the SSRAM via the Avalon Write MM interfaces by utilizing writing commands for dynamic address alignment and Avalon peripheral commands come in specific formats such as IORD_8DIRECT(BASE, OFFSET) or IOWR_16DIRECT(BASE, OFFSET, DATA). These commands are declared by the "alt_types.h" file, included in the C code as well. The sine table method was adopted owing to the large amount of memory space that inclusion of Nios® II custom instructions requires. This method is implemented for inverters, but in case of converters, numerical values are generated instead of sine waves to produce PWM signals with a duty cycle. File pointers were required for binary file handling and access in the PC host file system by the second C code.

There are three main software methods to communicate with the processor peripherals: Direct register access, Hardware abstraction layer (HAL) interface and standard ANSI C library functions. In this project all three methods were used. The HAL approach provides the C-language macros IOWR and IORD (Hamblen & Furman, 2001: 326). The "system.h" file is included in the source code by declaring it in the beginning. This file provides a software description of the microprocessor hardware and information about each peripheral in the specified SOPC Builder system analyzer (Altera Corporation, 2010d).



Start
Read value representing a PE application from “application” binary file
Inverter Or Converter ?
Converter
Inverter
Read value representing a switching frequency from “carrier” binary file
Switching frequency control value loaded into hardware design for up/ down counters operation
Read duty cycle percentage frequency from “duty_cycle” binary file
Read value representing a switching frequency from “carrier” binary file
Switching frequency control value loaded into hardware design for up/ down counters operation
Sine table is selected
Read value representing the required reference signal frequency from “sine” binary file
Load incremental step value into hardware design
Generation of sine wave quarters
End

Figure 4.25: The second C code’s flow chart

4.4.5.2 Nios® II IDE software implementation

The steps involved in software code implementation are described below:

- Creation of a new C/C++ application, consisting of two projects, an application project and a library project. The application project is the C/C++ source code developed by the designer, and the library project consists of BSP automated device component configuration files referencing to SOPC Builder processing (Tang, 2008);
- Hardware system.ptf file assignment to the newly created C/C++ application;
- A source C/C++ code development and insertion in the application and projects configuration;
- Project building, using the two projects, where an Executable and Linked Format (.elf) file is generated and loaded into the Nios® II system memory if no error has been located; and
- Project loaded and running by the Nios® II processor on hardware.

4.4.6. PCB production and completion of overall system

The creation of a custom FPGA board is beyond the scope of this project, but an additional hardware platform is needed to work in conjunction with the embedded Nios® II system to complete the digital controller design. One peripheral PCB was developed by using Altium® Designer v.6. This platform serves as a communication interface between the embedded Nios® II system and power electronic applications. It also operates for measurement and testing purposes for signals from the designed Nios® II system. For these activities to be made possible, this board accommodates a High Speed Mezzanine Card (HSMC). This card was a SAMTEC® ASP-122953-01 host board socket, which was plugged into the corresponding HSMC on the FPGA board. As the development of the peripheral PCB was completed and connected to the FPGA board through the HSMC connecter, the soft-core based controller is ready to be operational. The block diagram of the developed soft-core based controller is presented below in Figure 4.26 as a summary of the internals of the Nios® II system design and their interaction within the system.

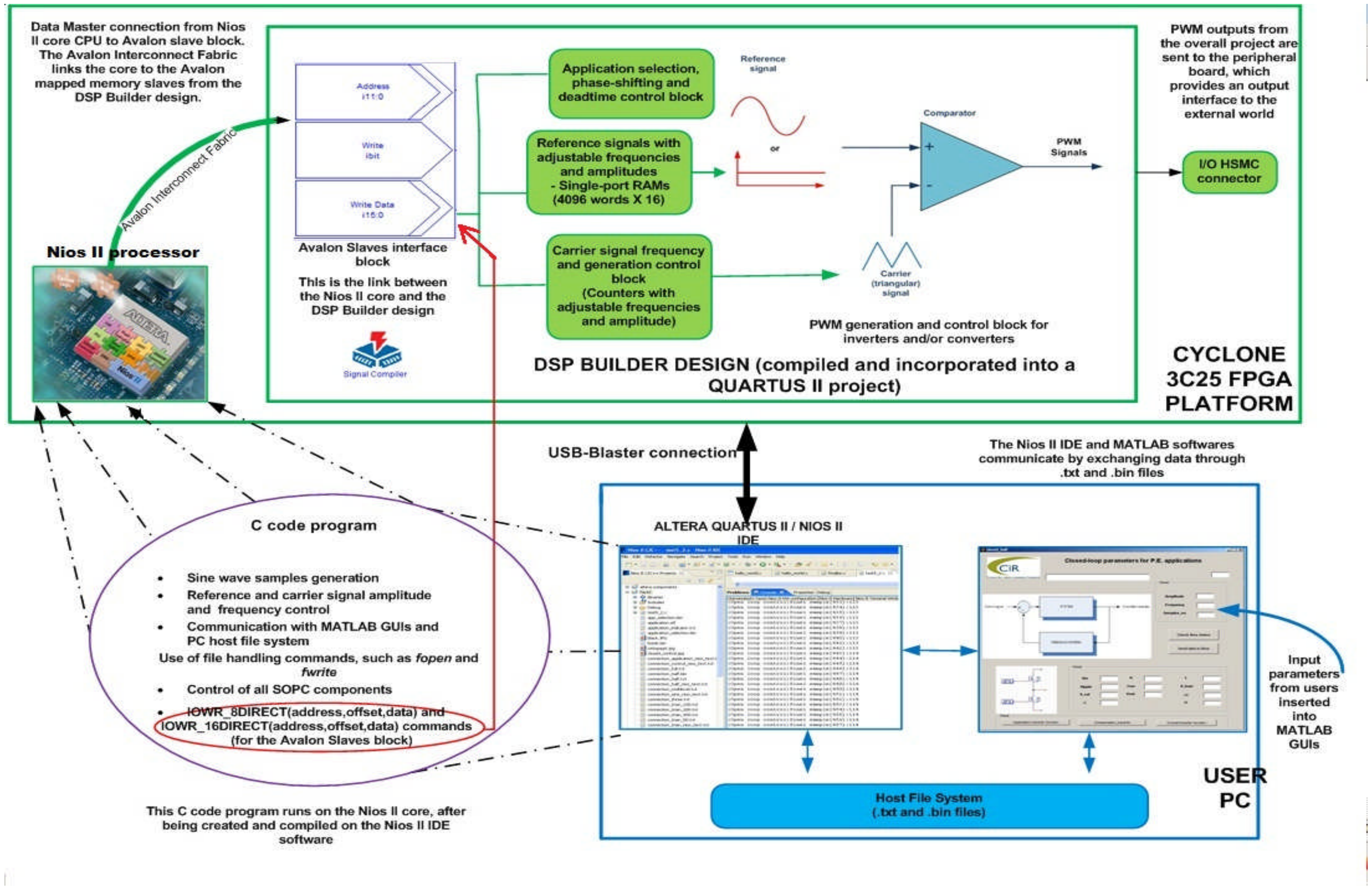

Data Master connection from Nios II core CPU to Avalon slave block. The Avalon Interconnect Fabric links the core to the Avalon mapped memory slaves from the DSP Builder design.
Avalon Interconnect Fabric
Nios II processor
Address i11:0
Write ibit
Write Data i15:0
Avalon Slaves interface block
This is the link between the Nios II core and the DSP Builder design
Signal Compiler
Application selection, phase-shifting and deadtime control block
Reference signals with adjustable frequencies and amplitudes - Single-port RAMs (4096 words X 16)
Carrier signal frequency and generation control block (Counters with adjustable frequencies and amplitude)
Reference signal
or
Comparator
PWM Signals
Carrier (triangular) signal
PWM generation and control block for inverters and/or converters
DSP BUILDER DESIGN (compiled and incorporated into a QUARTUS II project)
PWM outputs from the overall project are sent to the peripheral board, which provides an output interface to the external world
I/O HSMC connector
CYCLONE 3C25 FPGA PLATFORM
USB-Blaster connection
The Nios II IDE and MATLAB softwares communicate by exchanging data through .txt and .bin files
ALTERA QUARTUS II / NIOS II IDE
Closed-loop parameters for P.E. applications
Input parameters from users inserted into MATLAB GUIs
Host File System (.txt and .bin files)
USER PC
C code program
Sine wave samples generation
Reference and carrier signal amplitude and frequency control
Communication with MATLAB GUIs and PC host file system
Use of file handling commands, such as fopen and fwrite
Control of all SOPC components
IOWR_8DIRECT(address,offset,data) and IOWR_16DIRECT(address,offset,data) commands (for the Avalon Slaves block)
This C code program runs on the Nios II core, after being created and compiled on the Nios II IDE software

Figure 4.26: Block diagram of the developed soft-core based controller

4.5 Summary

This chapter presented the controller specifications, its design flow and the Nios® II processor functions in the SOPC system. It also provided details of the design and implementation steps involved in both software and hardware aspects. The controller's flexibility is maintained by the ability of input reconfiguration on both the DSP Builder design and C code.

CHAPTER 5

SIMULATION AND EXPERIMENTAL RESULTS

5.1 Introduction

This chapter presents and discusses results following performance tests at various points of the controller's design and implementation process.

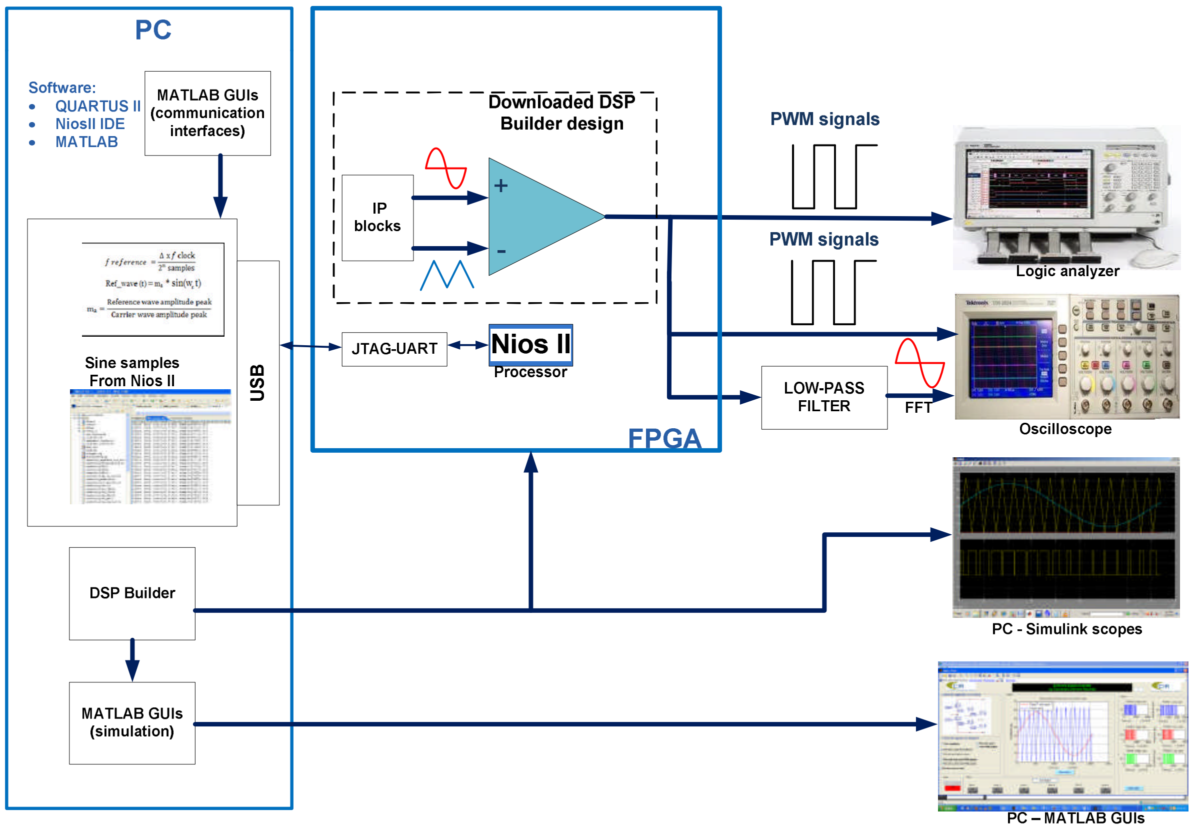


Figure 5.1: Testing setup block diagram

The simulation test comprised evaluating the Nios® II processor's performance via the validity of sine samples that it generates by DSP Builder simulation. The hardware level of the completed PWM modulator was also tested by DSP Builder. The experimental test involved evaluation on the Nios® II core performance of the digital controller from the FPGA hardware level. The controller's flexibility in terms of signal frequency ranges and application selection was monitored, as all the necessary modules and units were integrated to complete the controller system. The block diagram of the Nios® II-based controller and its test environment are illustrated in Figure 5.1.

5.2 Nios® II and DSP Builder simulation

The PWM modulator design was verified by testing it entirely in synthesizable VHDL format before it was implemented into the FPGA. The entire DSP Builder simulation was performed in the DSP Builder/Simulink environment.

As stated in the previous chapter, the advantage of the DSP Builder model simulation is that it is the visualization of the exact hardware implementation of the controller system without any hardware manufacturing. It allows the designer to rapidly identify any problems, troubleshoot them and re-simulating the model until satisfactory results are obtained.

In this work, the testing process began with sine samples' data which was extracted from the Nios® II IDE console. These sine sample values were generated digitally from trigonometric calculations based on the formulae that were given in Chapter Four to determine the right parameters of the signal running on the Nios® II processor, and are expected to be the exact hardware representation of a fourth of a sine wave.

It was first intended to have floating point custom instructions integrated into the processor core to increase the calculations' speed, but they occupy a large amount of memory space, as mentioned in Chapter Four. A simple C code was developed by using some parameter values that were determined through calculations. The carrier signal and application selection blocks were configured manually, according to calculations. As C codes were run onto the processor, the sine waves values, which were generated, could not be viewed and monitored directly from the Write Interface. Hence, in this testing process the samples were not written directly into the Avalon MM Write Slave Interfaces ‘s addresses, but were rather collected from the IDE console to form HDL sine tables, as Figure 5.3 shows.

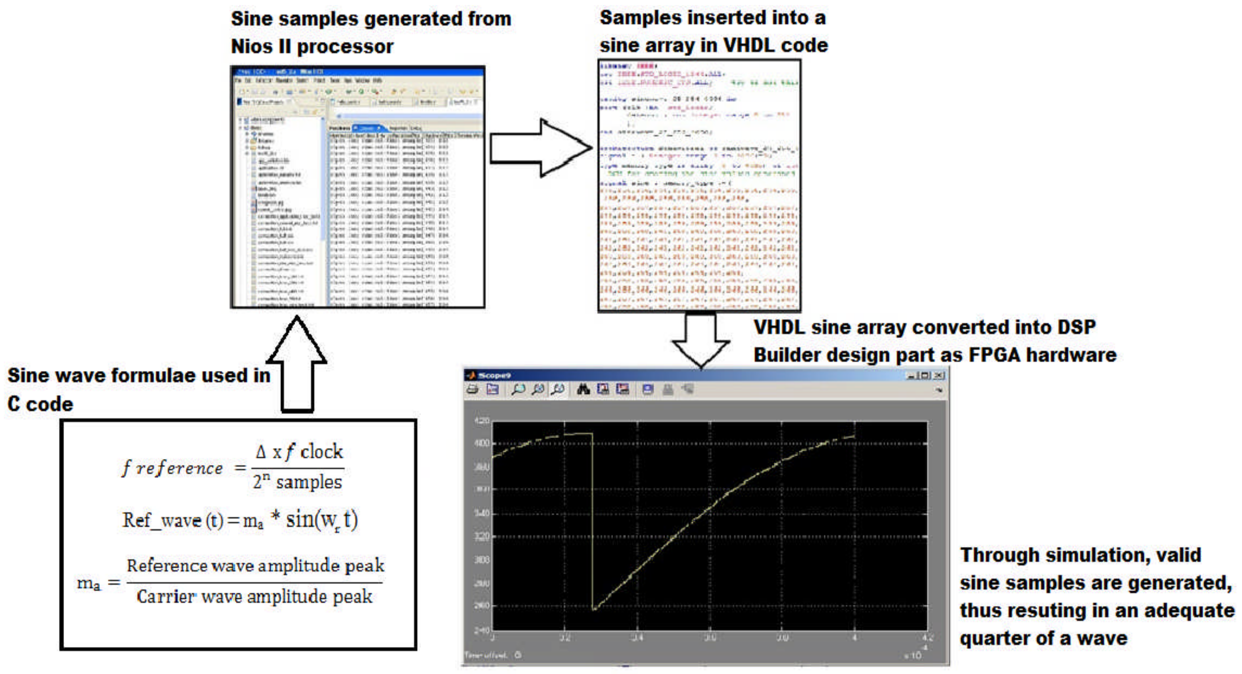

Sine samples generated from Nios II processor
Samples inserted into a sine array in VHDL code
VHDL sine array converted into DSP Builder design part as FPGA hardware
Sine wave formulae used in C code
$f\,reference = \frac{\Delta \; x\, f \text{ clock}}{2^n \text{ samples}}$
$\text{Ref\_wave}(t) = m_a * \sin(w_r t)$
$m_a = \frac{\text{Reference wave amplitude peak}}{\text{Carrier wave amplitude peak}}$
Through simulation, valid sine samples are generated, thus resuting in an adequate quarter of a wave

Figure 5.2: Sine samples validity verification process block diagram

This specific testing process showcased the validity of the sample values that were generated by the Nios® II core trigonometric and arithmetic calculations and their implementation in FPGA hardware. In this instance, the sine sample values that were acquired from the Nios® II core were used as HDL input data. Some “constant” DSP Builder blocks were configured to emulate Nios® II control data, and used as control inputs to complete the hardware design of the digital controller for simulation. The operating clock “peripheral_clk” on the DSP Builder design could be altered at any time on the DSP Builder design to emulate a different ALTPLL clock signal. This change generated other frequency ranges for switching and reference sine signals that could not be obtained with the 50 MHz FPGA global clock signal. Some MATLAB® GUIs were created and used to display simulation results by using Simulink “To Workspace” blocks on the DSP Builder to store signal data into MATLAB®’s workspace. All this data appeared to be similar to the ones displayed by the DSP Builder scopes and proved their compatibility with MATLAB®.

5.3 Experimental testing

In order to verify the possibility of applying the SCBC’s in a practical PE environment, simulations are not enough to prove its reliability and adequate operation. Hence an experimental test on the HDL design had to be realized in order to evaluate the performance of the controller. The controller’s outputs, the PWM switching signals, are responsible for the control of the required PE applications, as stated in Chapter Two. Thus they were the focus in this hardware test. The Nios® II processor’s functionality was tested, as it controls the digital controller’s hardware through both its input and output signals by producing the required PWM signals for specific applications and controlling their frequency, duty cycle and deadtime.

This testing process was carried out upon completion of the digital controller’s SOPC system. The configuration of the FPGA was performed by downloading the generated .sof programming file onto the board with the USB-Blaster download cable and making the Nios® II system operate as the direct brain of the controller. The latter was accomplished by building and running another C program code on Nios II® IDE, based on the sine samples collected from the IDE console and, which compute data read from binary files stored in the PC- host based file system. The data on the binary files originate from MATLAB® GUIs.

The experimental results were captured in the form of digital waveform signals at the output of a SAMTEC® ASP-122952-01 High Speed Mezzanine daughter Card (HSMC) connector placed on a peripheral board to be plugged into the corresponding SAMTEC® ASP-122953-01 HSMC host connector on the FPGA kit. This facilitates the use of measurement equipment probes on the FPGA development kit. The controller’s hardware operation is verified by using measurement equipment, which is displayed in Figure 5.1 and listed in Table 5.1 below.

Measurement equipments	
Oscilloscope	Tektronix® TDS2024B
Logic analyser	Agilent® Technologies 1683A

Table 5.1: Measurement equipments

The oscilloscope was used to display the switching signal frequency, high side PWM signals, and to perform FFT analysis for reference sine waves. PWM outputs consist of a spectrum of several harmonic frequency components. FFT signals were obtained by passing a single PWM signal out through a second order LC low-pass filter acting as some kind of a DAC. The LC filter filters out all the higher frequency components in the PWM signal, and omits out the reference signal frequency component to be captured on the oscilloscope.

The logic analyser is advantageous for capturing and displaying multiple switching PWM signals in one time and over varying ranges of time periods.

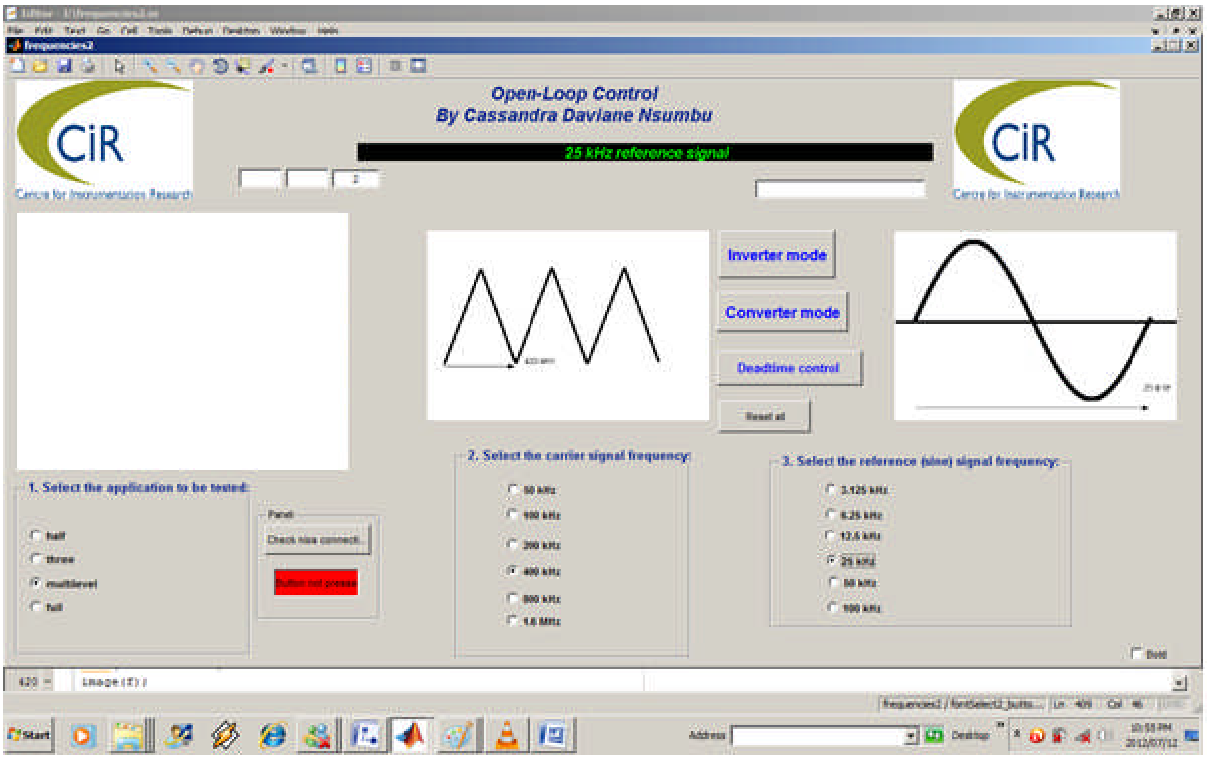
As mentioned in Chapter Four and shown in Figure 5.4, some MATLAB® GUIs were created to serve as a communication interface between the SCBC user and the Nios® II processor. They were also tested and were successful in communicating with the processor by making it control the FPGA hardware design as expected. To demonstrate the effects of altering the size of sine tables on the C code by selecting different frequencies from the MATLAB® GUIs, the results of this aspect were presented. The adequate frequencies were captured and displayed on the oscilloscope. It should be remembered that altering the PLL clock on the SOPC Builder will produce other frequencies than the ones termed on the MATLAB® GUIs. These frequencies were altered in the same way by the same value that was used for the PLL clock signal.

This chapter presents both simulation and experimental test results for comparison purposes. During the simulation testing, carrier, reference and PWM signals were analyzed and displayed on the Simulink scopes. During the experimental testing of the SCBC, only PWM and FFT signals were captured, analysed, displayed and compared to the simulation results.

5.4 Switching and reference signal frequency control results

The SCBC's reconfigurability, in terms of input and output frequency variation was verified in simulation and on the hardware level. As mentioned earlier, "constant" blocks were inserted on the DSP Builder design to represent the number of signal bits that were required by the hardware. The GUI in Figure 5.4 was used as the input generator for the SCBC's signal frequency control in the experimental testing. As the user activated the GUI's radiobuttons, data was written onto binary files, which was then computed by the Nios® II processor to control the DSP Builder design to generate the required switching and reference frequencies.

By altering the “peripheral_clk” signal on SOPC Builder, carrier and reference signals running in different frequency ranges were produced; were equally tested; and proved to be adequate for the SCBC’s operation. Different frequency ratios were also implemented and verified throughout the testing process. The reference signal frequency component was captured and analyzed from the PWM harmonic spectrum in order to validate adequate reference signal generation.



1. Select the application to be tested:
- half
- three
- [✓ multilevel]
- full

2. Select the carrier signal frequency:
- 50 kHz
- 100 kHz
- 200 kHz
- [✓ 400 kHz]
- 800 kHz
- 1.6 MHz

3. Select the reference (sine) signal frequency:
- 3.125 kHz
- 6.25 kHz
- 12.5 kHz
- [✓ 25 kHz]
- 50 kHz
- 100 kHz

Figure 5.3: Activation of application and signal generation control GUI

All the frequencies for both carrier range (50 kHz to 1.6 MHz) and reference range (3.125 to 100 kHz) were tested and proved to be correctly generated, but only some of them are displayed below.

- Figures 5.5 and 5.6 display one cycle of the 400 kHz PWM signal, while Figure 5.7 displays the 25 kHz reference signal component in the 400 kHz PWM harmonic spectrum. Figure 5.5 also displays the simulation of the carrier and reference signals involved in the PWM production on the upper section of the scope window.

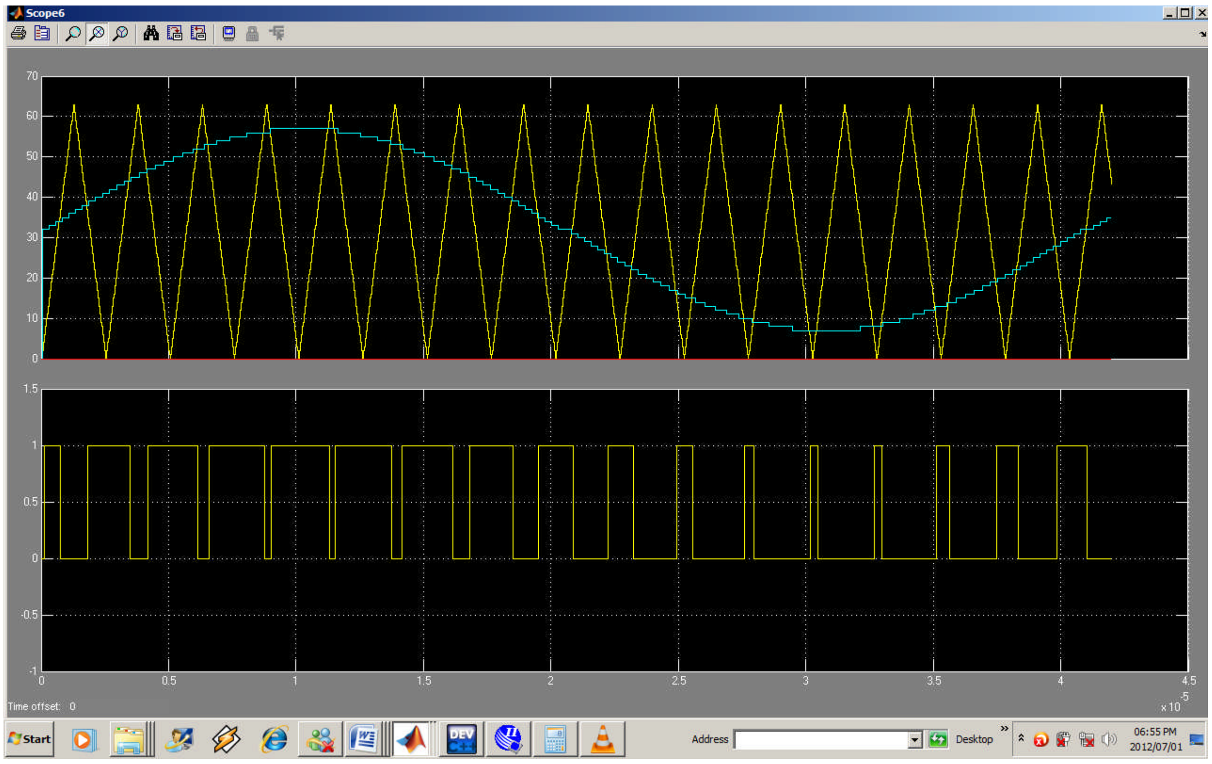


Figure 5.4: DSP Builder simulation for a switching frequency of 400 kHz and a 25 kHz reference signal

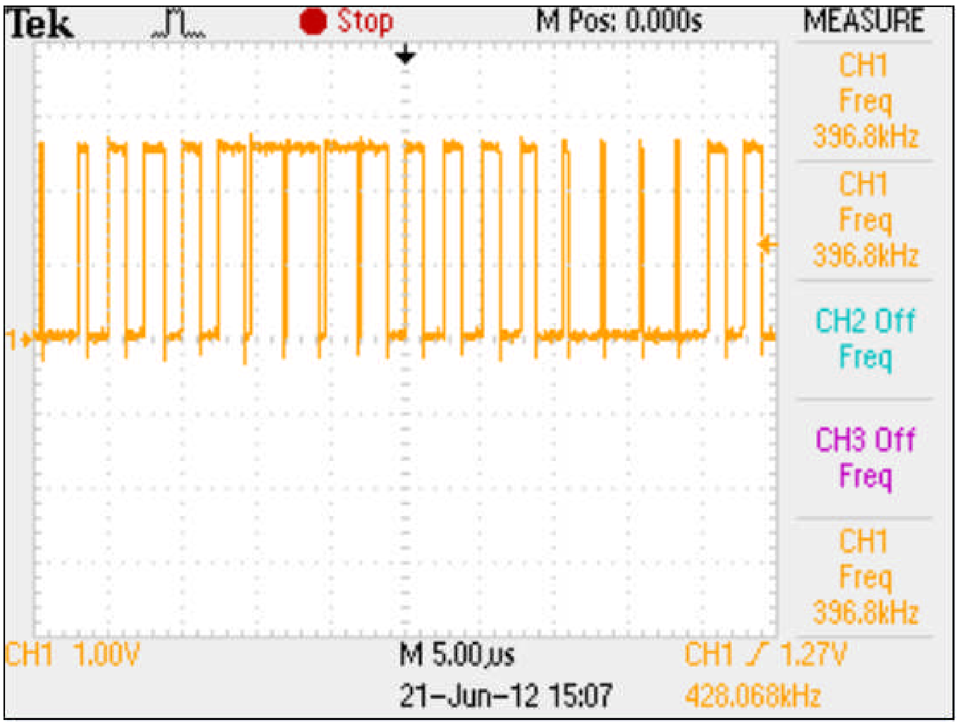


Figure 5.5: Experimental result of a switching frequency of 400 kHz and a 25 kHz reference signal

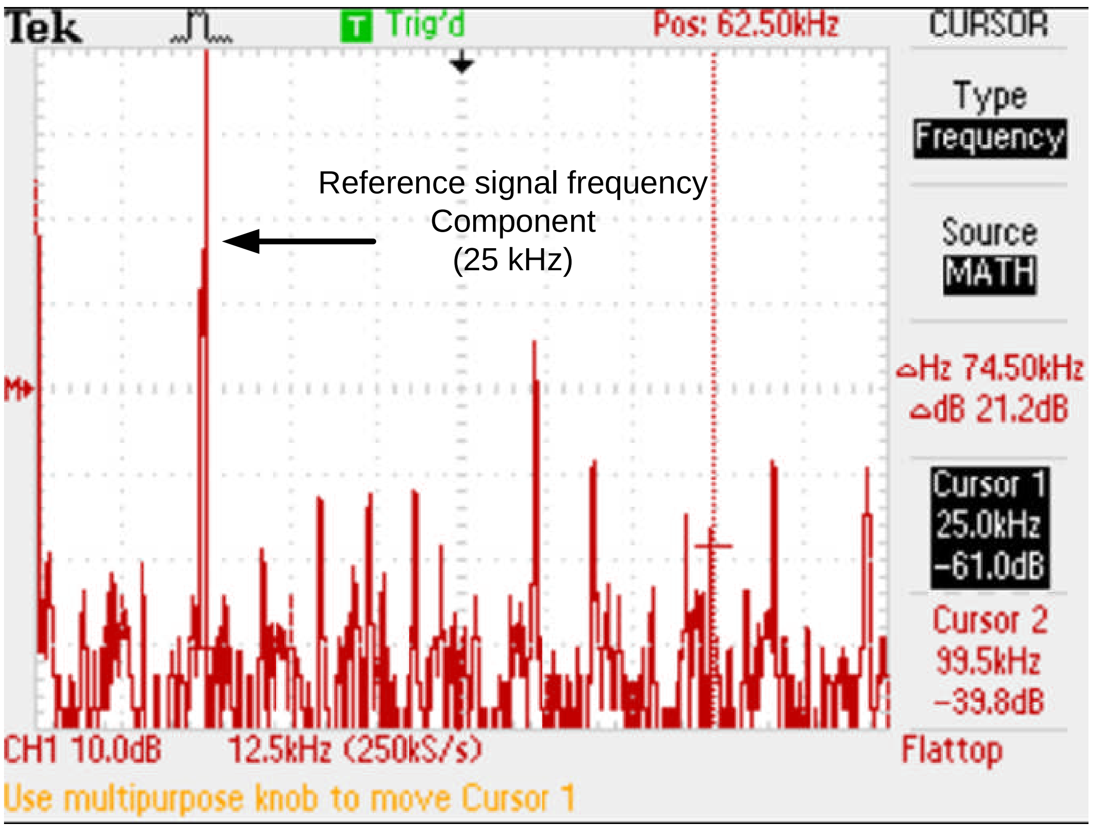


Figure 5.6: Measured FFT of a single 400 kHz PWM gating signal displaying the 25 kHz reference frequency

- Figures 5.8 and 5.9 display one cycle of the 100 kHz PWM signal, while Figure 5.10 displays the 6.25 kHz reference signal component in the 100 kHz PWM harmonic spectrum. Figure 5.8 also displays the simulation of the carrier and reference signals involved in the PWM production on the upper section of the scope window.

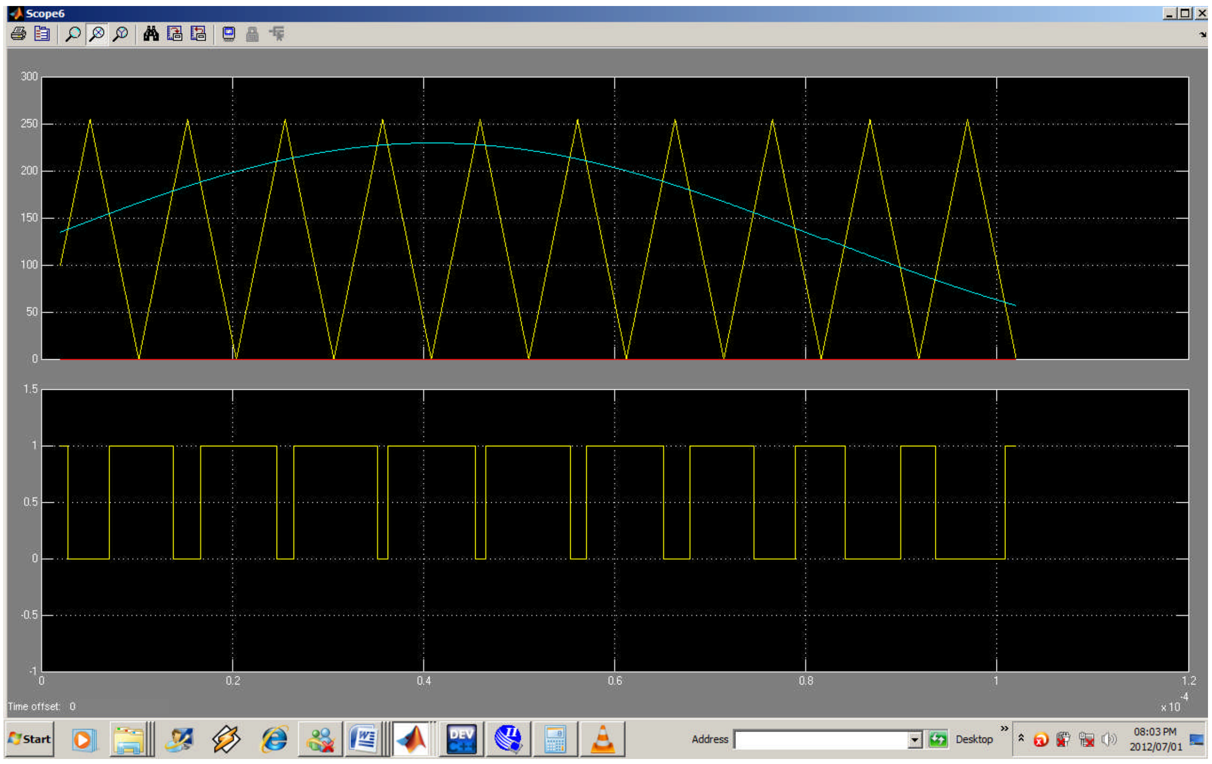


Figure 5.7: DSP Builder simulation of interleaved switching for a switching frequency of 100 kHz and a 6.25 kHz reference signal

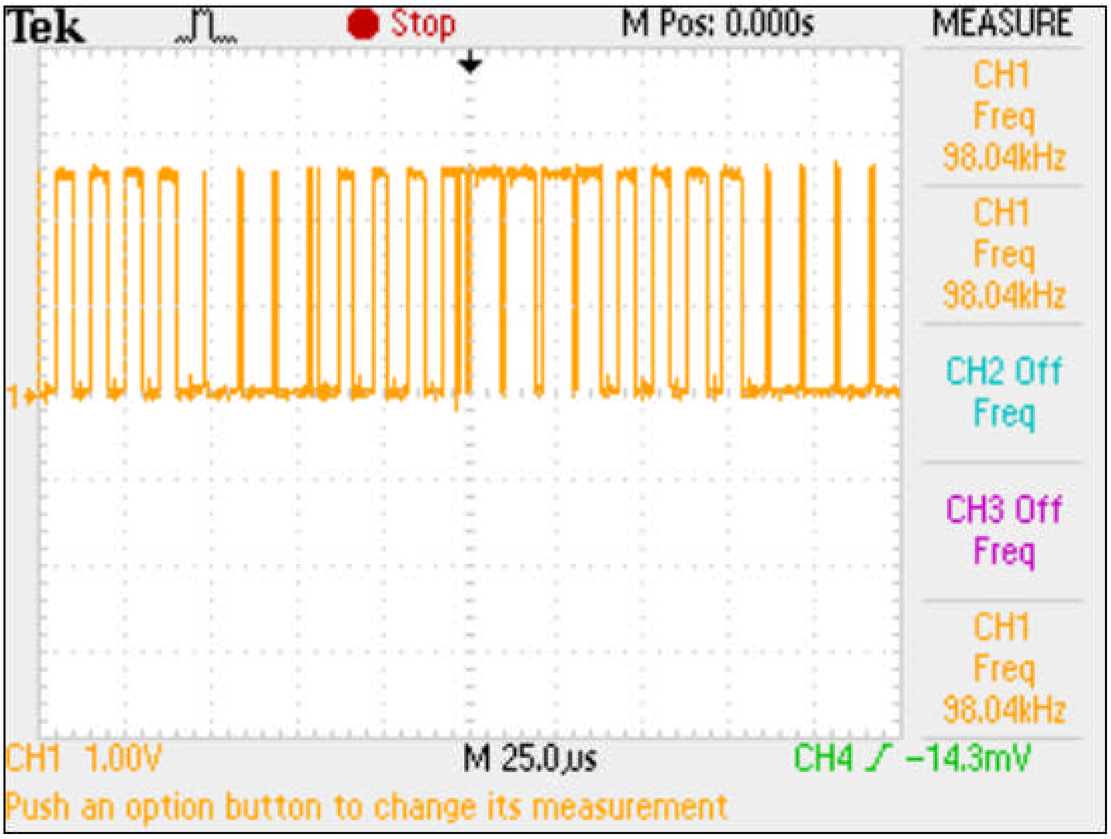


Figure 5.8: Measured 100 kHz PWM signal

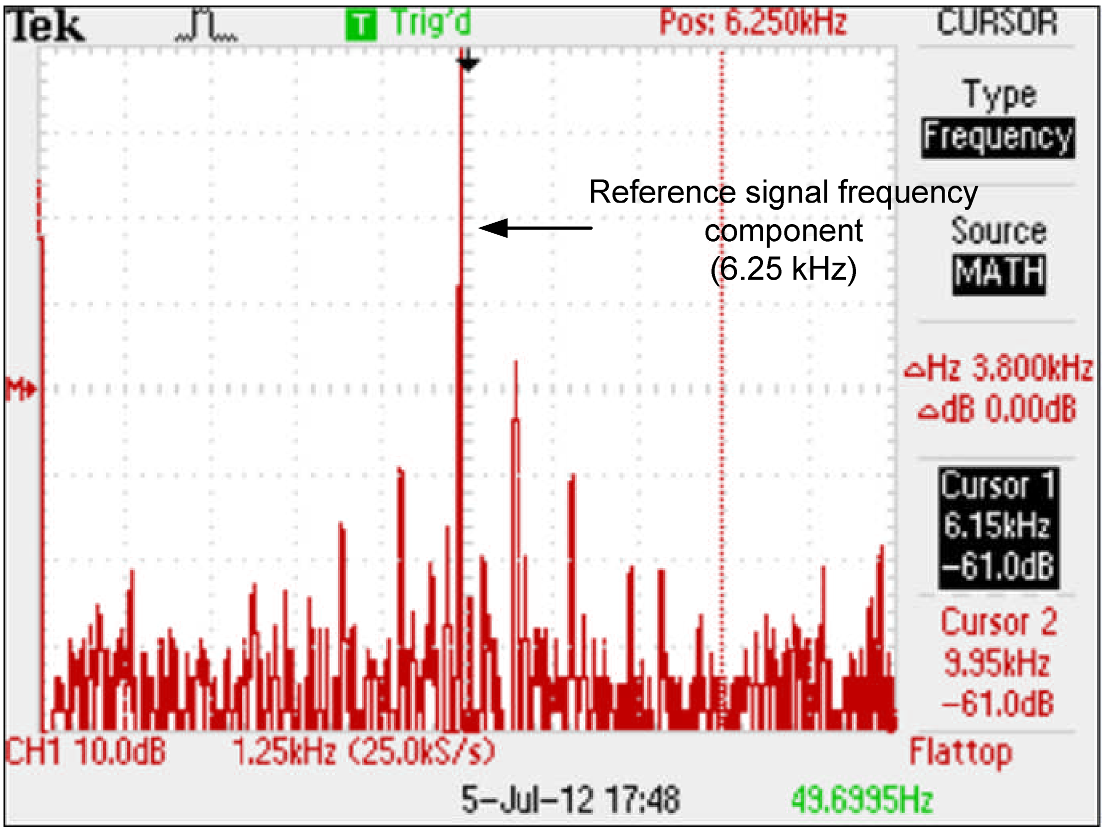


Figure 5.9: Measured FFT of a single 100 kHz PWM gating signal displaying 6.25 kHz reference frequency

- Figures 5.11 and 5.15 display the 200 kHz PWM signal. Figure 5.8 also displays the simulation of the carrier and reference signals involved in the PWM production on the upper section of the scope window. A simulation GUI for 3-cell inverters is shown in Figure 5.12. Figure 5.13 and 5.14 are enlarged screenshots of interleaved switching signals and PWM signals, respectively, from the simulation GUI.

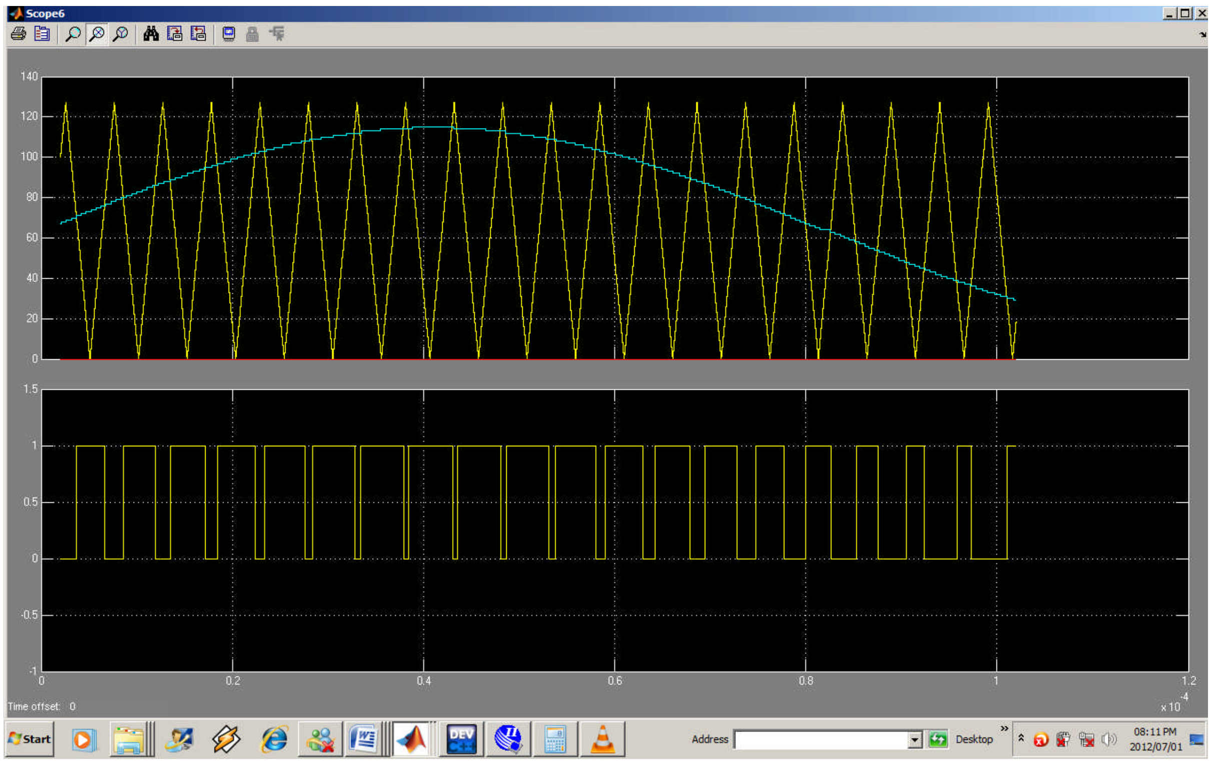


Figure 5.10: DSP Builder simulation of interleaved switching for a switching frequency of 200 kHz and a 6.25 kHz reference signal

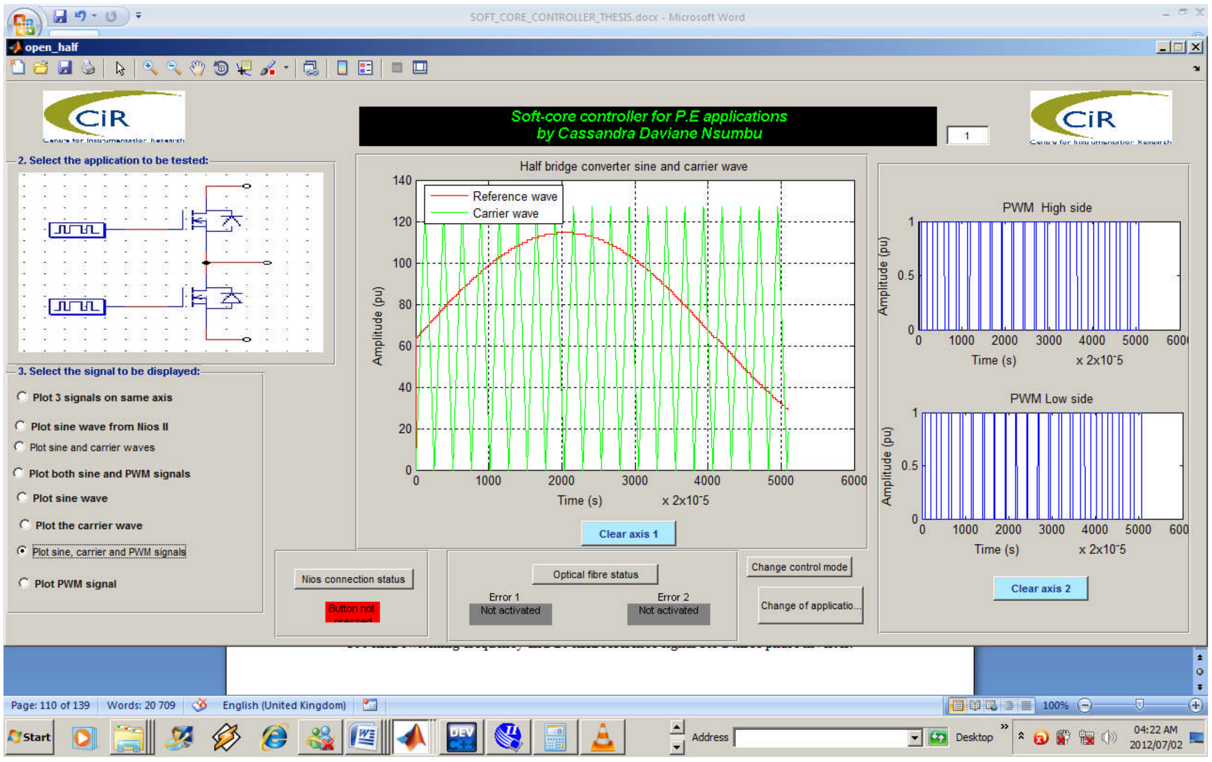
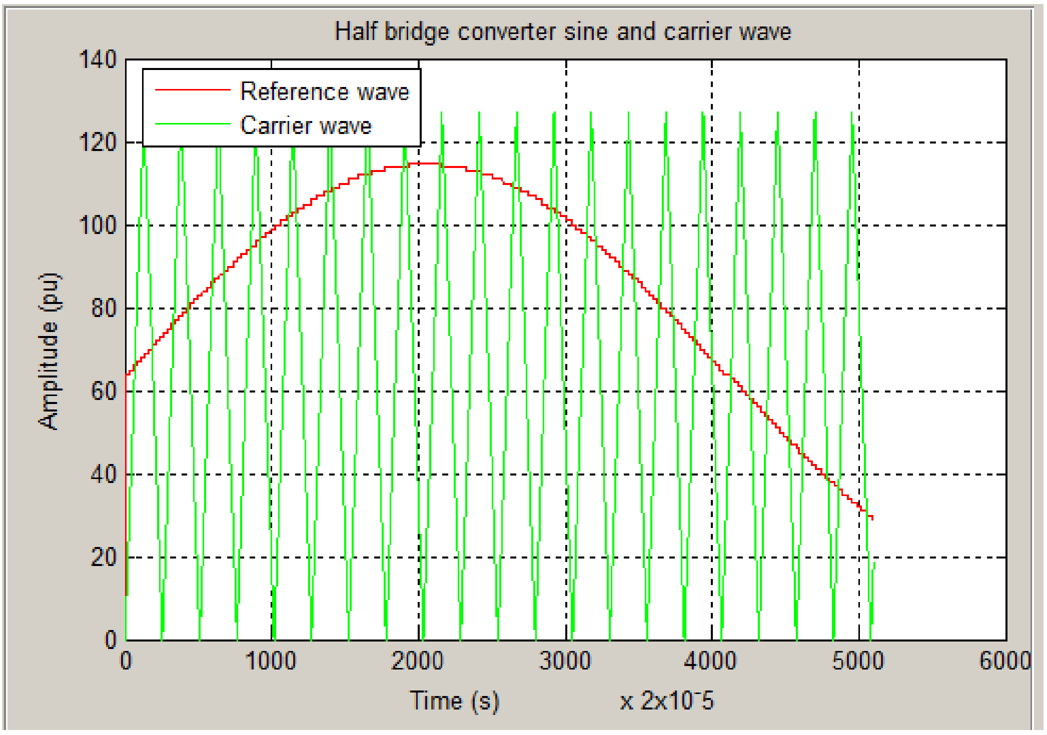
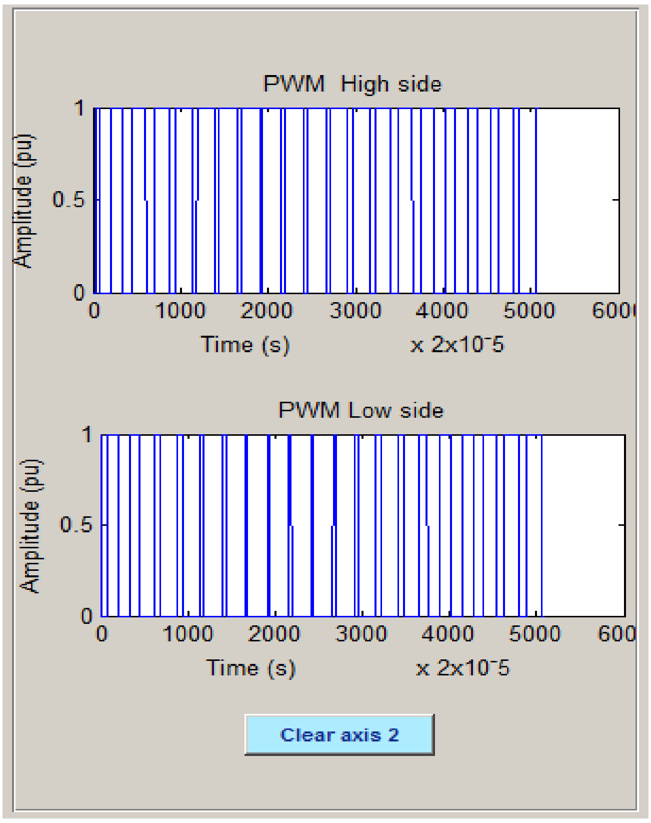

open_half
Soft-core controller for P.E applications
by Cassandra Daviane Nsumbu
2. Select the application to be tested:
3. Select the signal to be displayed:
Plot 3 signals on same axis
Plot sine wave from Nios II
Plot sine and carrier waves
Plot both sine and PWM signals
Plot sine wave
Plot the carrier wave
Plot sine, carrier and PWM signals
Plot PWM signal
Half bridge converter sine and carrier wave
Reference wave
Carrier wave
Amplitude (pu)
Time (s)
Clear axis 1
PWM High side
PWM Low side
Clear axis 2
Nios connection status
Button not pressed
Optical fibre status
Error 1
Not activated
Error 2
Not activated
Change control mode
Change of applicatio...

Figure 5.11: Simulation GUI for Half bridge converter



a)



b)

Figure 5.12: Simulation GUI zoom screenshots of a) interleaved switching and b)the ed PWM signals

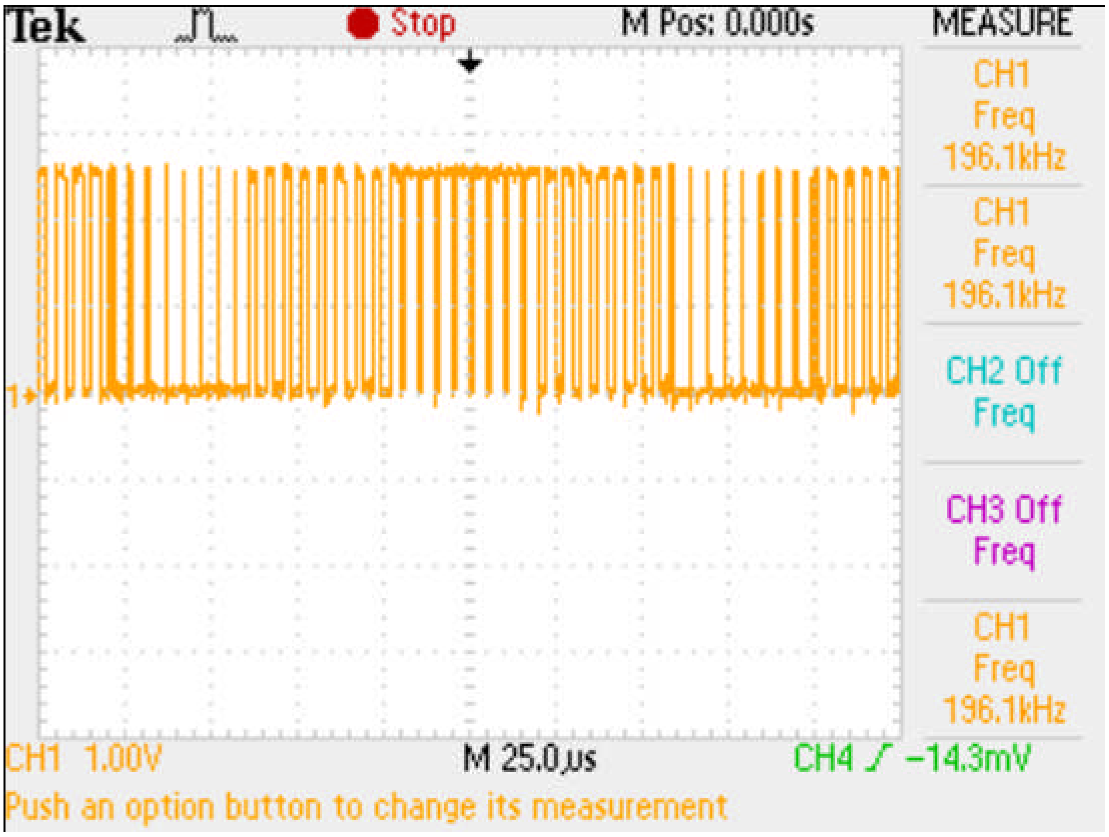


Figure 5.13: DSP Builder simulation of interleaved switching for a switching frequency of 200 kHz and a 6.25 kHz reference signal

- Figures 5.15 and 5.16 display one cycle of the 200 kHz PWM signal, while Figure 5.17 displays the 50 kHz reference signal component in the 200 kHz PWM harmonic spectrum. Figure 5.15 also displays the simulation of the carrier and reference signals involved in the PWM production on the upper section of the scope window.

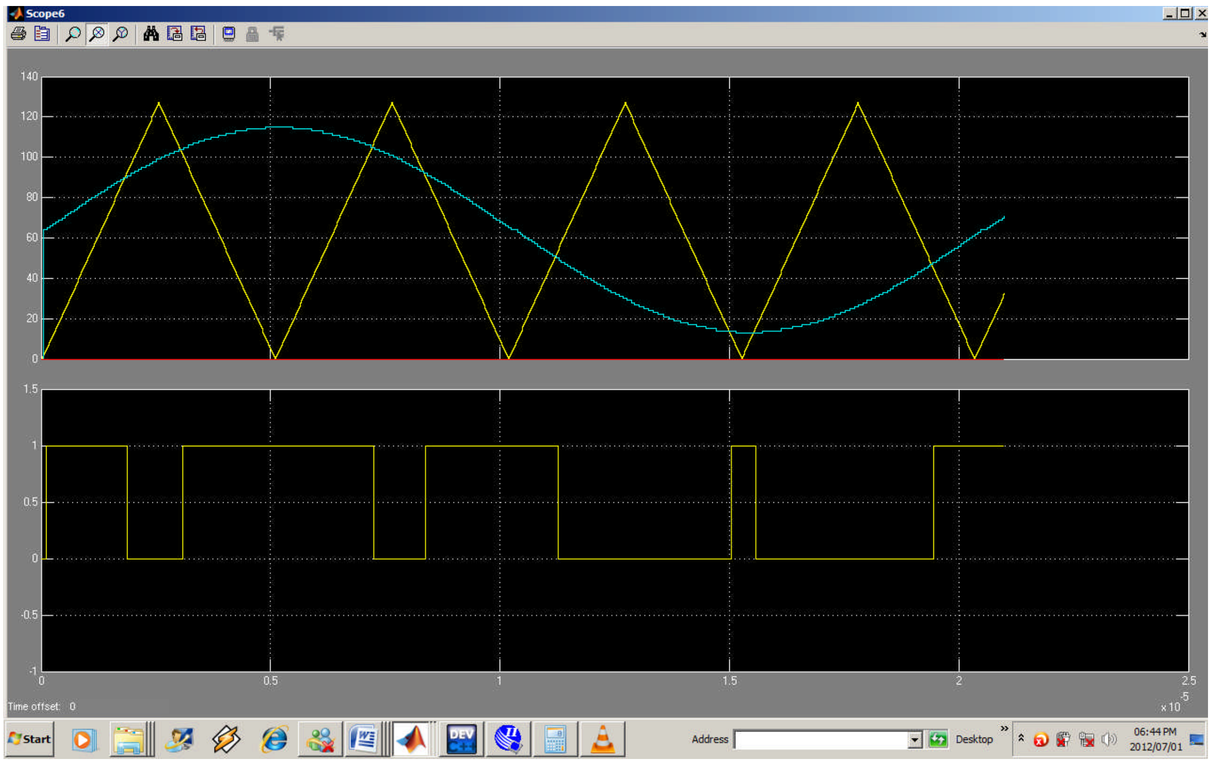


Figure 5.14: Three phase inverter - interleaved switching and PWM signals for a switching frequency of 200 kHz and reference signal of 50 kHz.

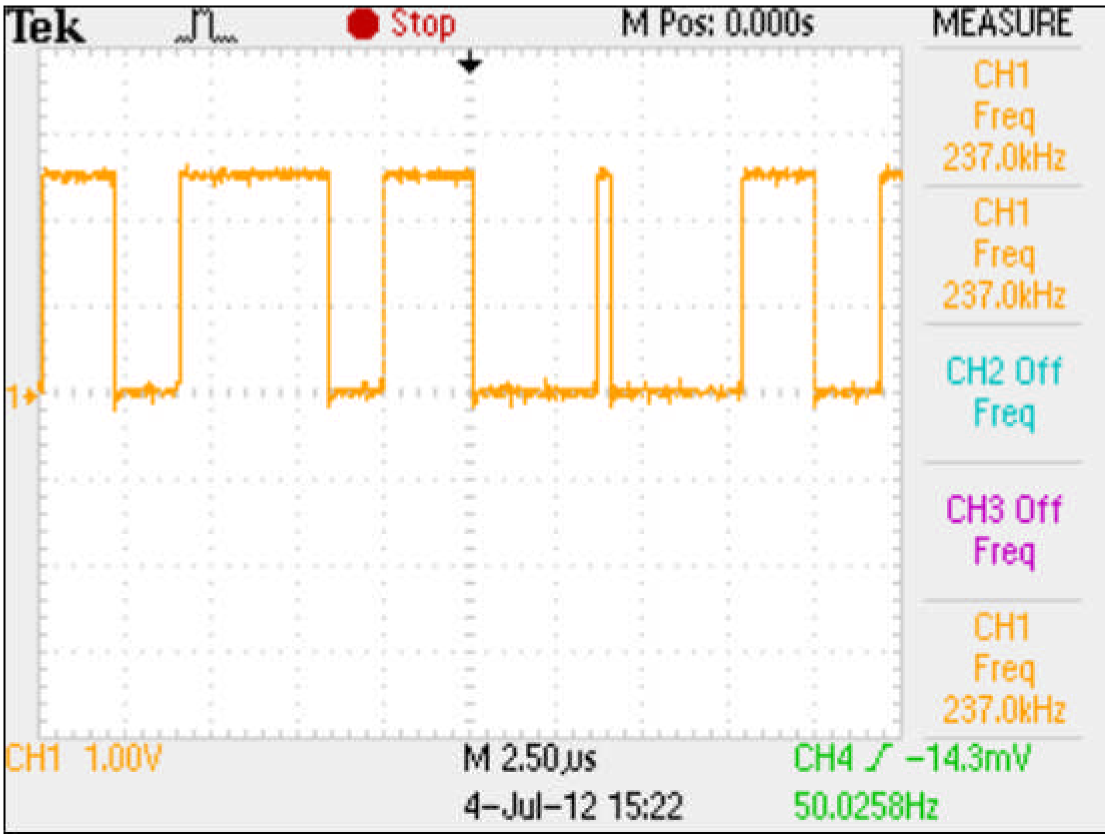


Figure 5.15: Measured 200 kHz high –side PWM signal

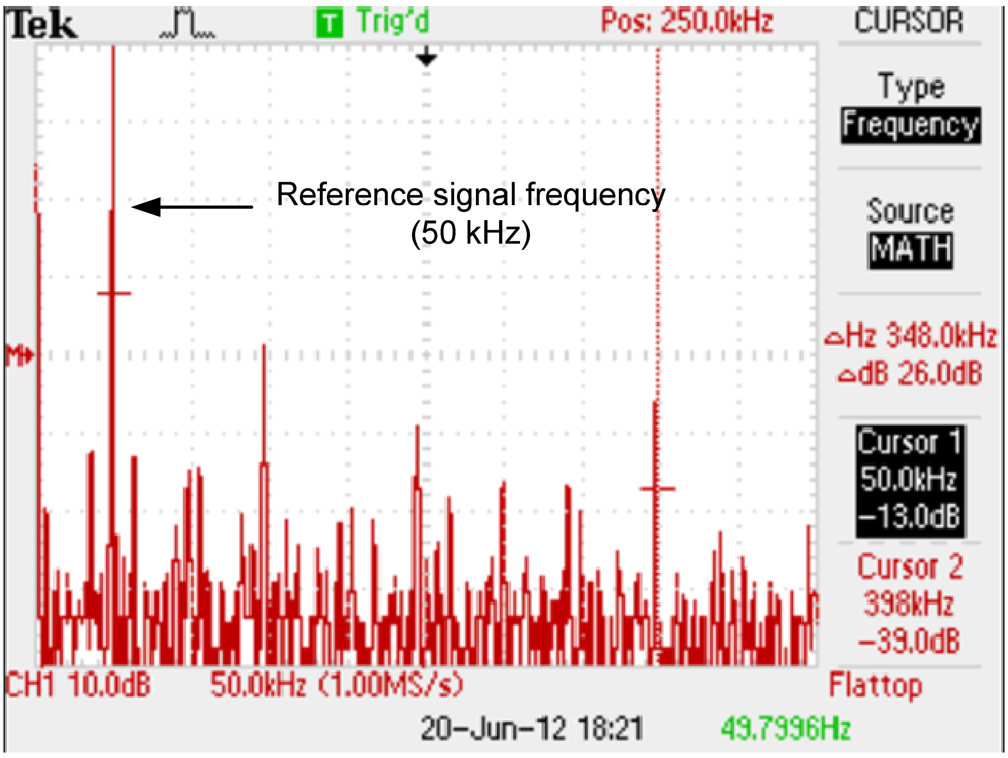


Figure 5.16: Measured FFT of a single 200 kHz PWM gating signal displaying 50 kHz reference frequency

- For the next instances, the "peripheral_clk" signal was changed to 833 Hz in order to produce another frequency range. Figure 5.18 displays one cycle of the 1.66 kHz PWM signal, while Figure 5.19 displays the 100 Hz reference signal component in the 833 Hz PWM harmonic spectrum. Figure 5.20 displays one cycle of the 833 Hz PWM signal and Figure 5.21 displays the 50 Hz reference signal component in the PWM harmonic spectrum.

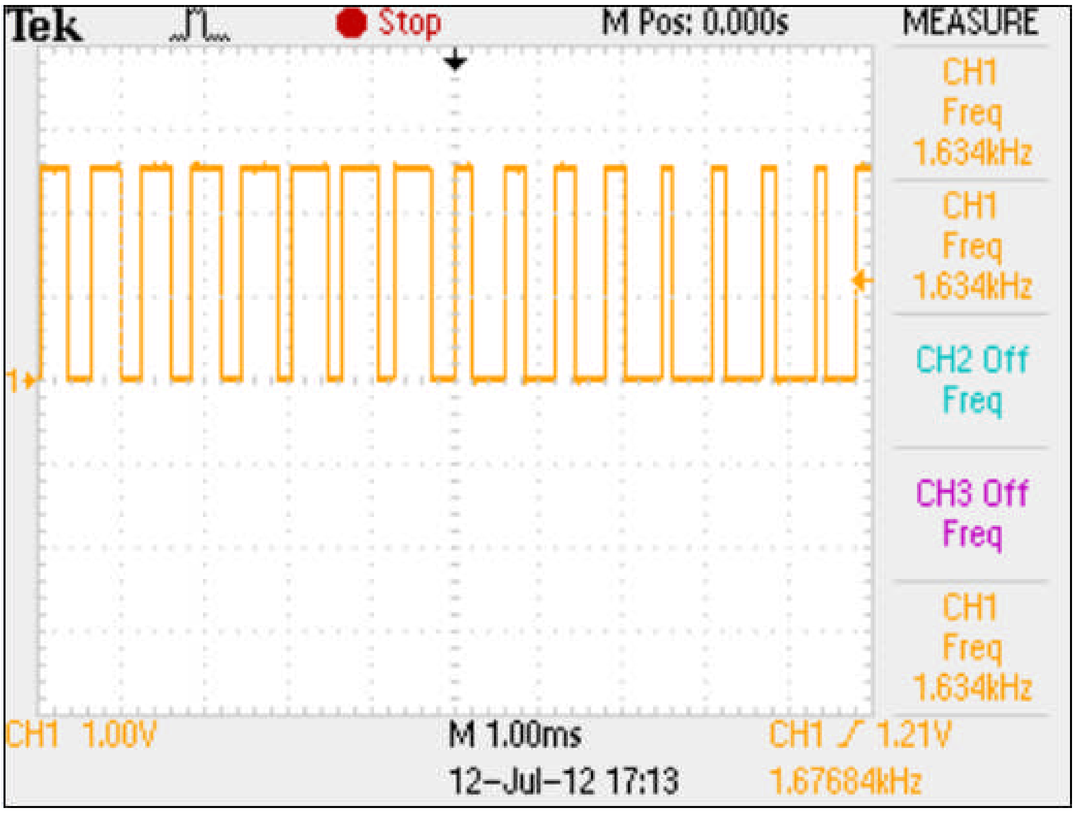


Figure 5.17: Measured 1.6 kHz high –side PWM signal

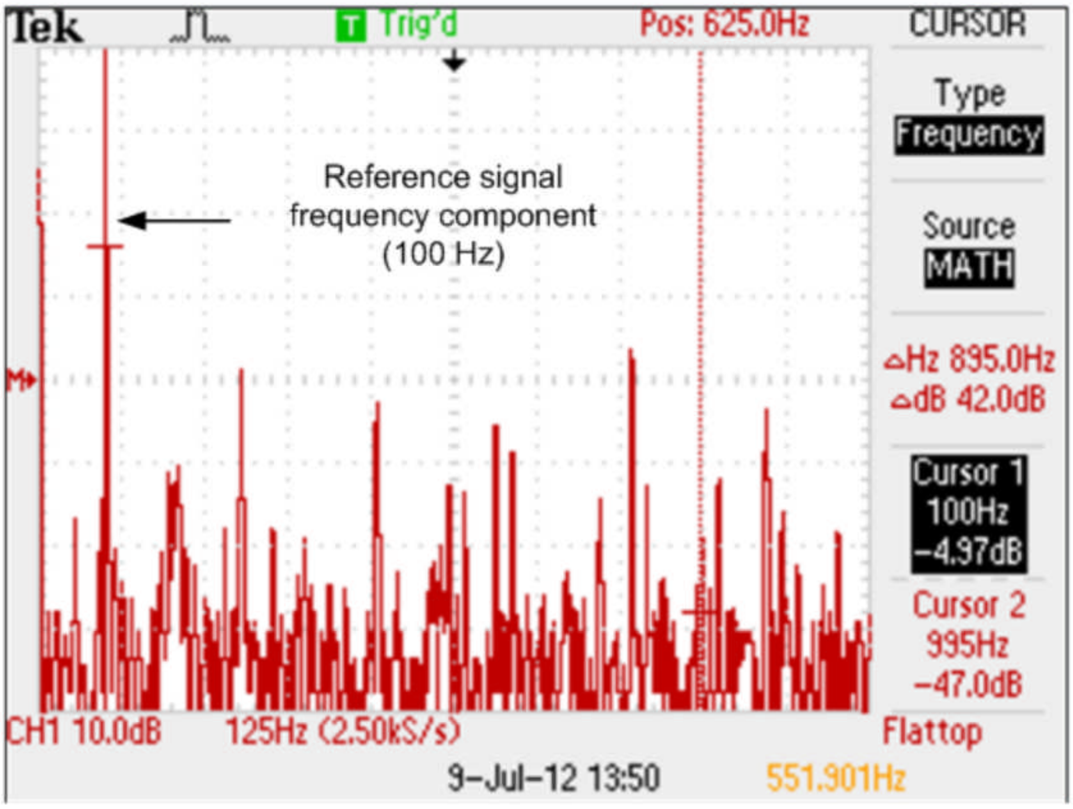


Figure 5.18: Measured FFT of a single 1.6 kHz PWM gating signal displaying 100 Hz reference frequency.

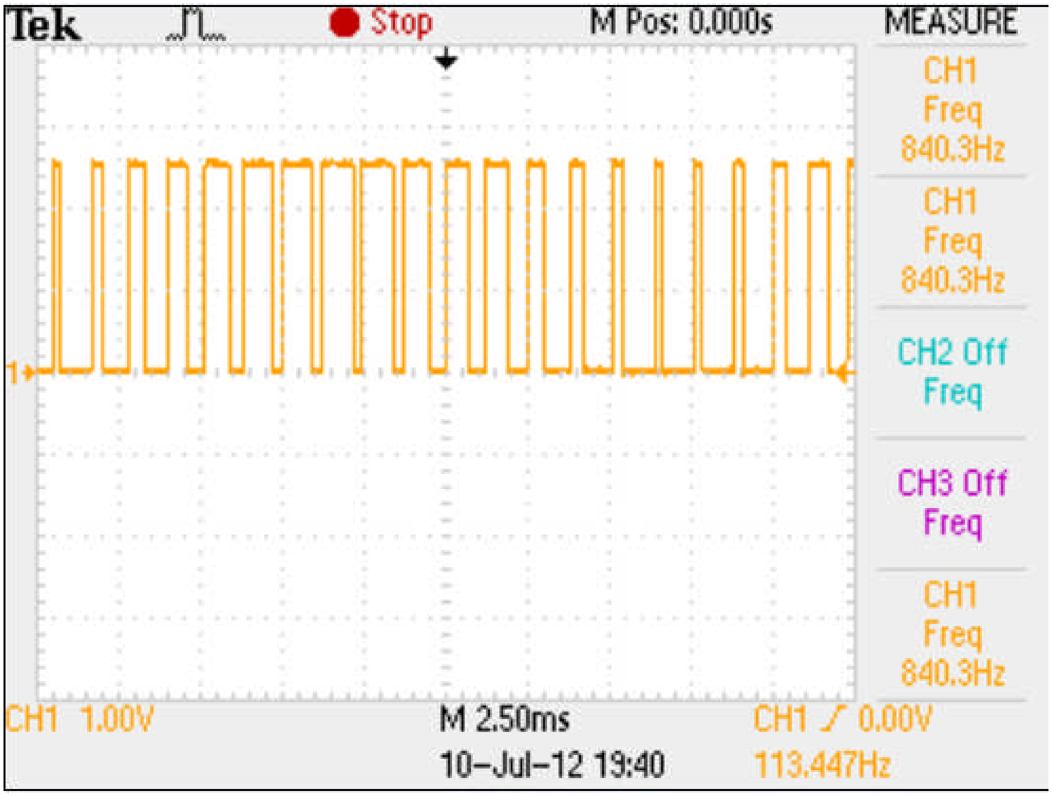


Figure 5.19: Measured 833 Hz high –side PWM signal

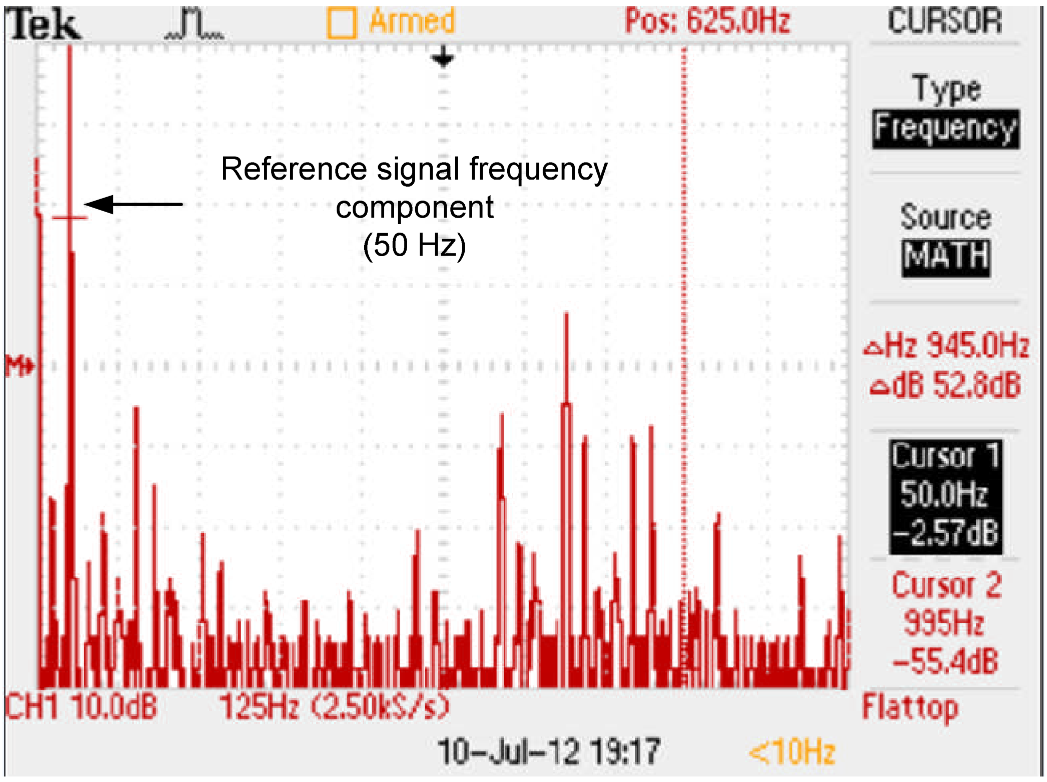


Figure 5.20: Measured FFT of a single 1.6 kHz PWM gating signal displaying 100 Hz reference frequency.

The collected results above prove the successful operation and flexibility of the SCBC in terms of signal frequency control, as all this section's figures exhibit adequate PWM signals, which operate at some of the possible frequencies.

5.5 Application control results

In this section the SCBC's reconfigurability in terms of PE application selection was tested and verified in simulation and on the hardware level. The SCBC can only operate on one PE application at a time though. Adequate HSMC pin assignments were carried out through the Quartus® II software to ensure the right selection, and the number of PWM signals to be produced accordingly. The number of PWM output signals is dependent on the number of power semiconductor devices that are present in the switching converter. For each P.E. application, required signal shifting was implemented and tested. In the case of three phase inverters, reference signals are phase-shifted by 120°. In the case of a 5-cell multilevel inverter, carrier signals are phase-shifted by 72°. To accomplish this, a time delay system was implemented in the DSP Builder design. The GUI from Figure 5.5 was involved in the PE application control.

- **Five-cell inverter**

As both Figures 5.22 and 5.23 show, the ten required PWM signals were observed in both simulation and experimental results. The 72° phase shifting can also be observed on both figures as they run the five complimentary PWM signal pairs. In Figure 5.22, the complimentary PWM signal pairs are shown to run after 72° of one another, creating a noticeable increased time delay.

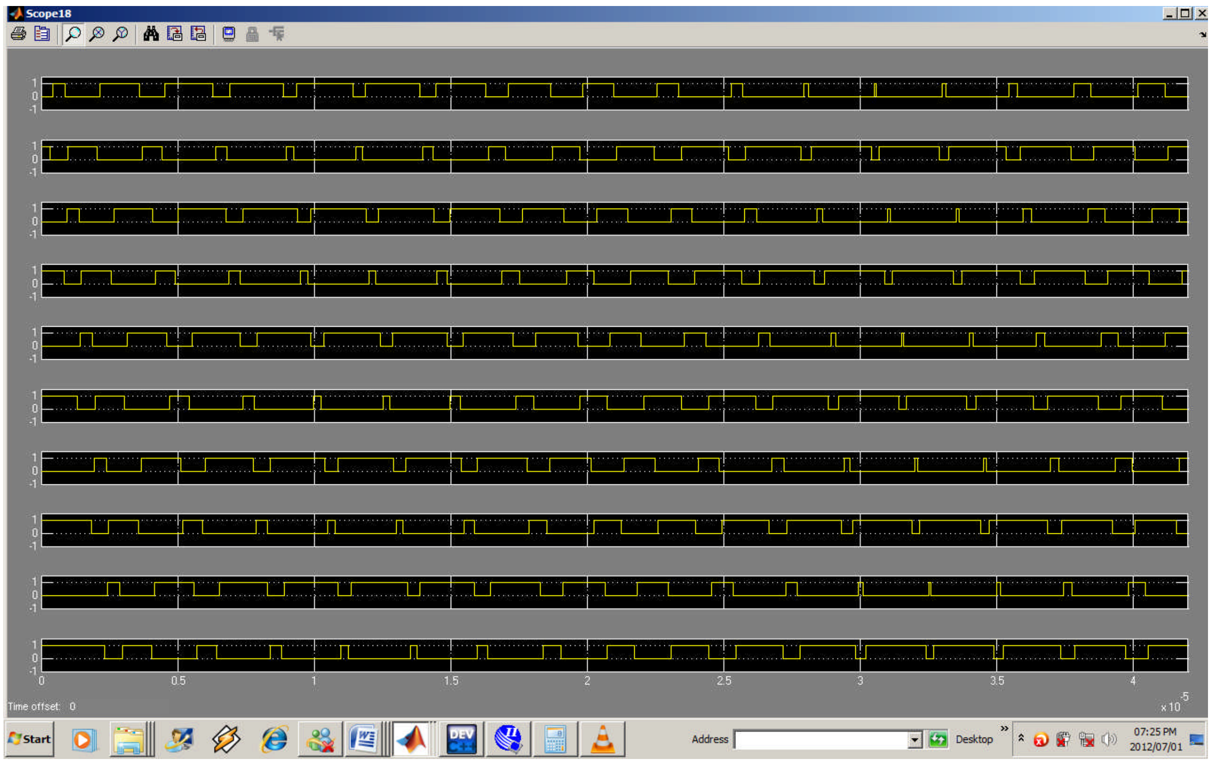


Figure 5.21: 5-cell multilevel inverter PWM signals (simulation – DSP Builder)

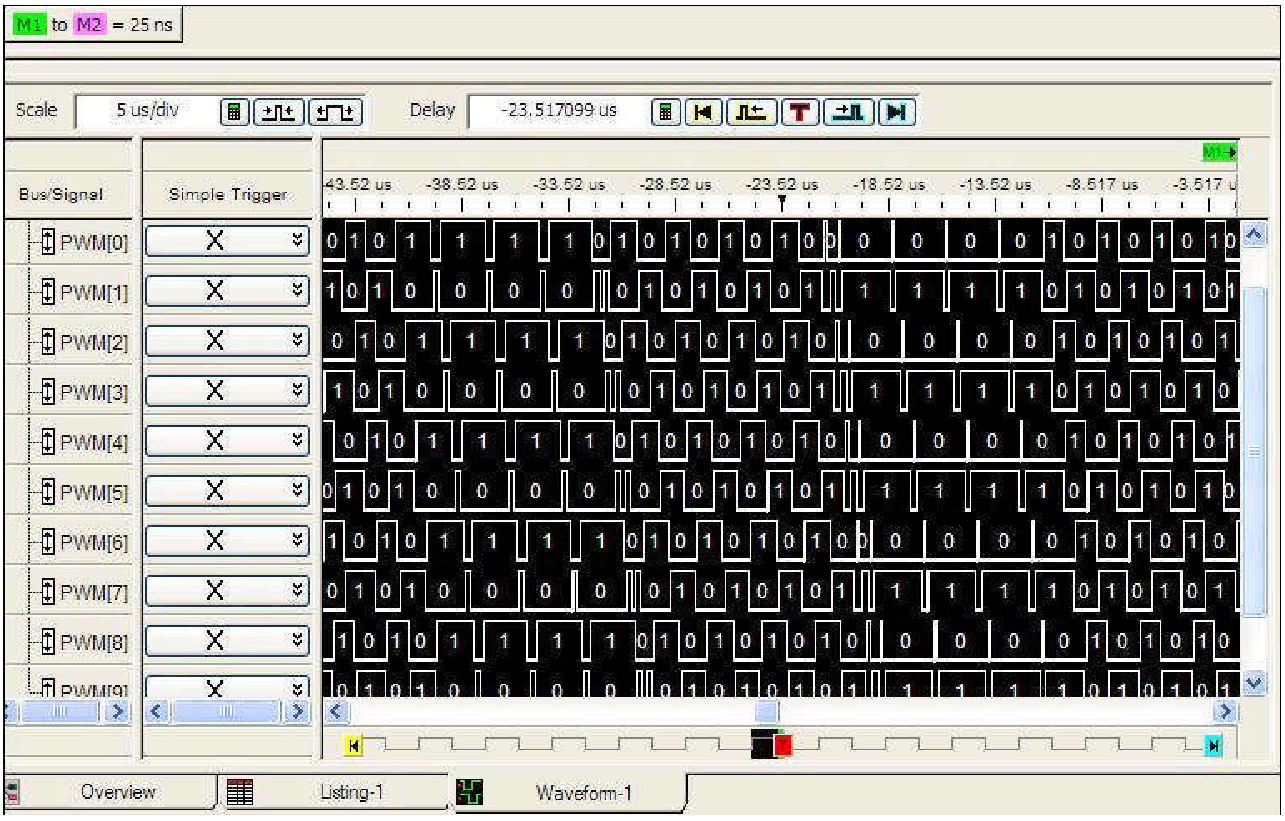


Figure 5.22: 5-cell multilevel inverter PWM signals (hardware level)

- **Three phase inverters**

The 120° phase shifting and the three complimentary PWM signal pairs are clearly apparent in Figures 5.24 – 5.27. Figures 5.24 – 5.26 display simulation results, which show a noticeable increased time delay after each PWM signal namely the 5-cell inverter. In this instance, it occurs after 120° of one PWM signal pairs.

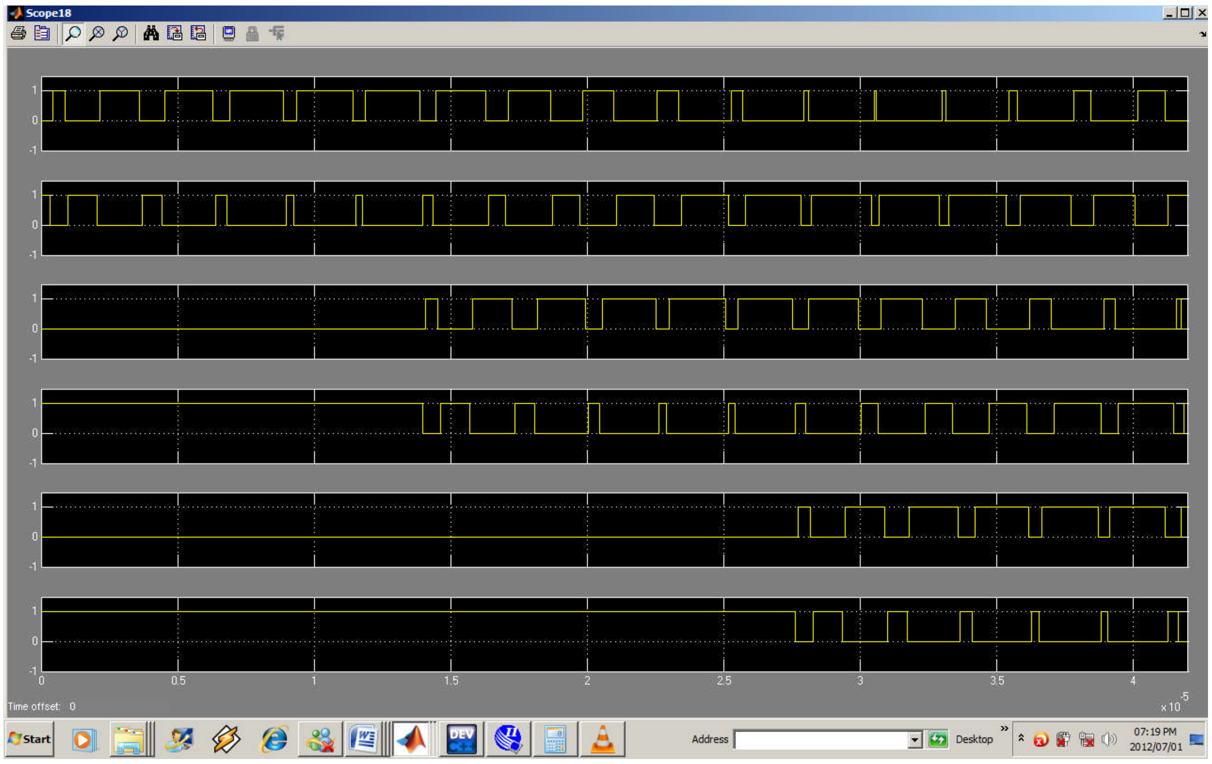


Figure 5.23: Three phase inverter PWM signals (simulation – DSP Builder)

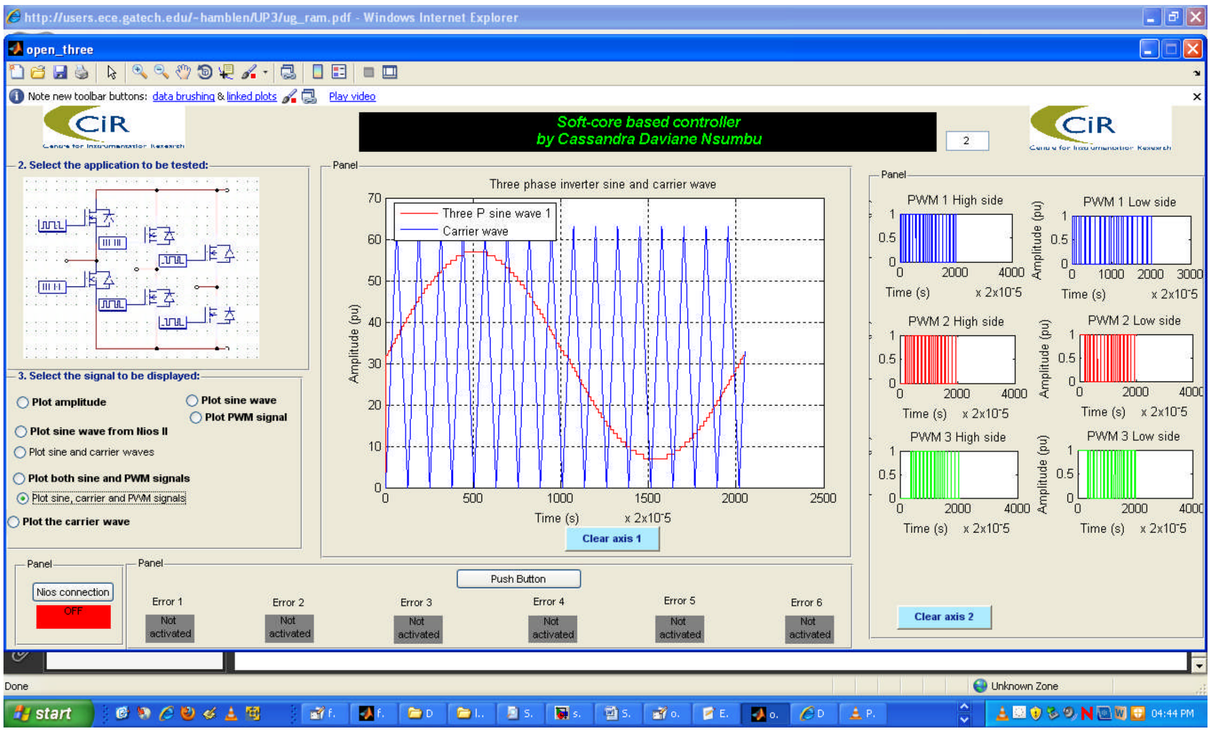

http://users.ece.gatech.edu/~hamblen/UP3/ug_ram.pdf - Windows Internet Explorer
open_three
Note new toolbar buttons: data brushing & linked plots Play video
Soft-core based controller
by Cassandra Daviane Nsumbu
2. Select the application to be tested:
3. Select the signal to be displayed:
Plot amplitude
Plot sine wave
Plot PWM signal
Plot sine wave from Nios II
Plot sine and carrier waves
Plot both sine and PWM signals
Plot sine, carrier and PWM signals
Plot the carrier wave
Three phase inverter sine and carrier wave
Three P sine wave 1
Carrier wave
Amplitude (pu)
Time (s) x 2x10^-5
Clear axis 1
PWM 1 High side
PWM 1 Low side
PWM 2 High side
PWM 2 Low side
PWM 3 High side
PWM 3 Low side
Clear axis 2
Nios connection
OFF
Push Button
Error 1
Error 2
Error 3
Error 4
Error 5
Error 6
Not activated
Done
Unknown Zone
start
04:44 PM

Figure 5.24: Three phase inverter inverter PWM signals (simulation - MATLAB)

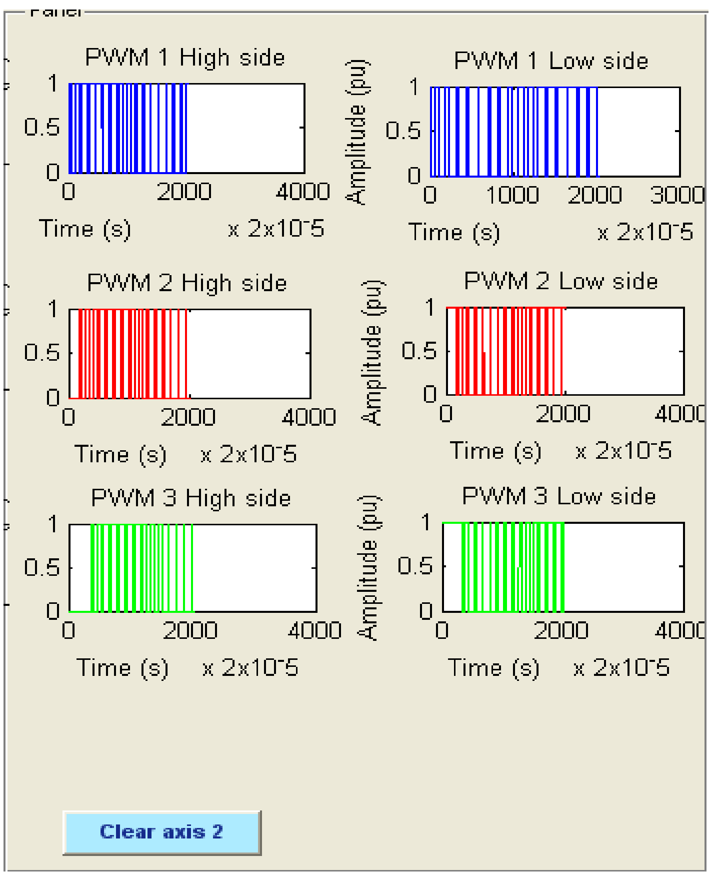


Figure 5.25: Enlarged screenshot of Three phase inverter inverter PWM signals (simulation - MATLAB®)

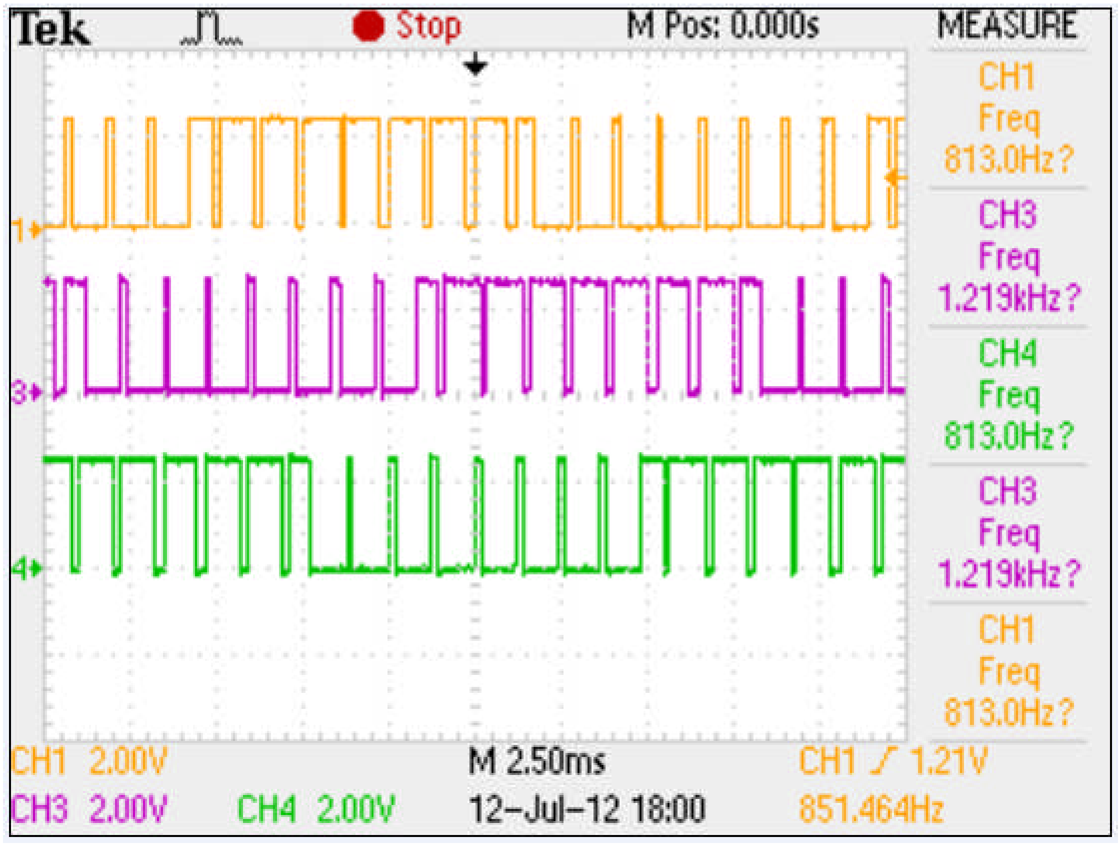


Figure 5.26: Three High –side PWM signals running at 833 Hz (hardware level)

- **Half bridge converter**

Figures 5.28 and 5.29 display the two required PWM signals.

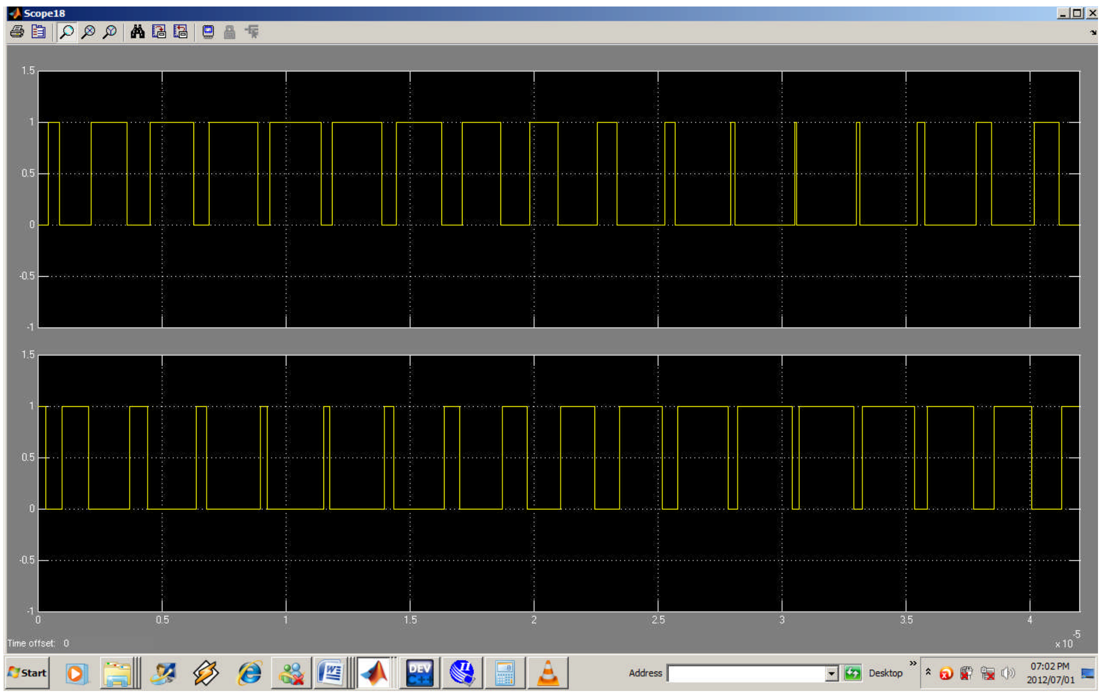


Figure 5.27: Half bridge converter PWM signals (simulation – DSP Builder)

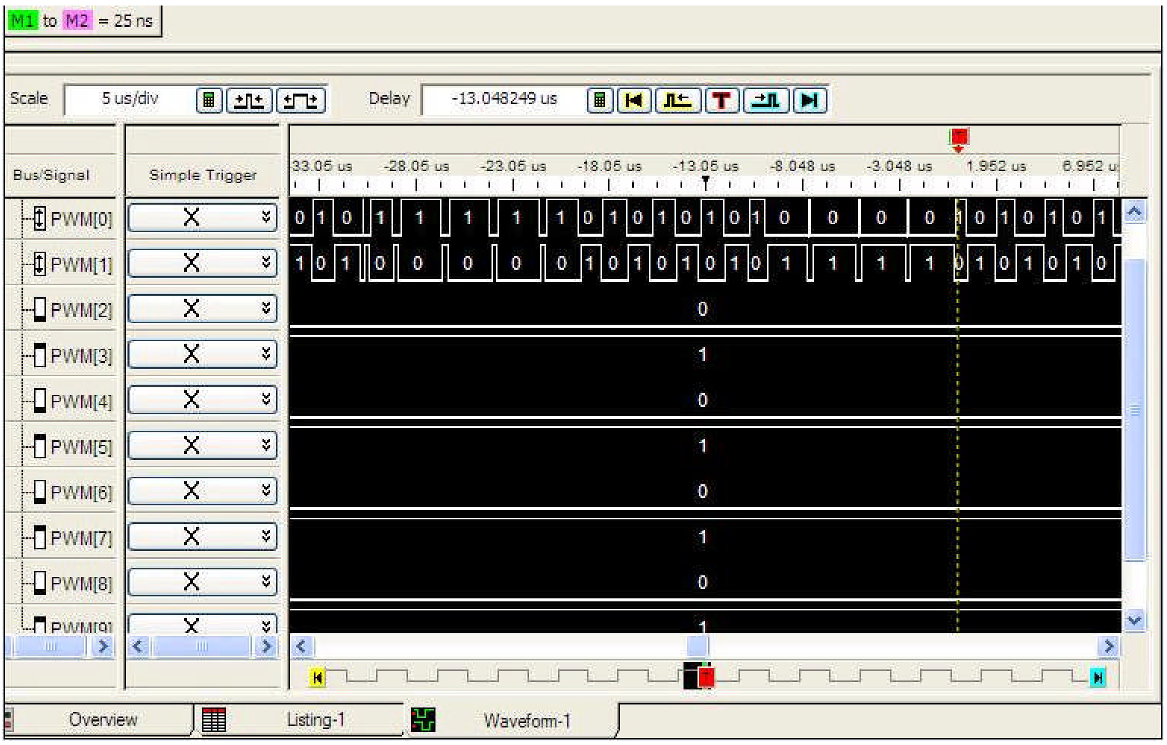


Figure 5.28: Half bridge converter PWM signals (hardware level)

- **Full bridge converter**

Figures 5.30 and 5.31 show the four required PWM signals and the 180° phase shift between the two PWM signal pairs.

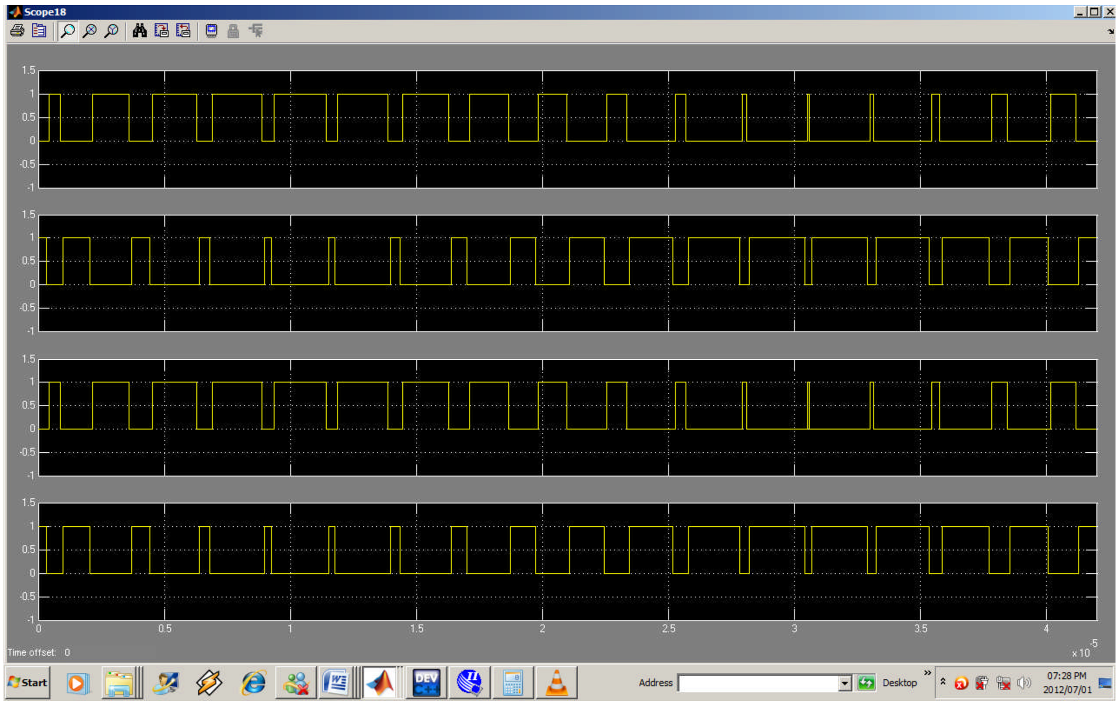


Figure 5.29: Full bridge converter PWM signals (simulation – DSP Builder)

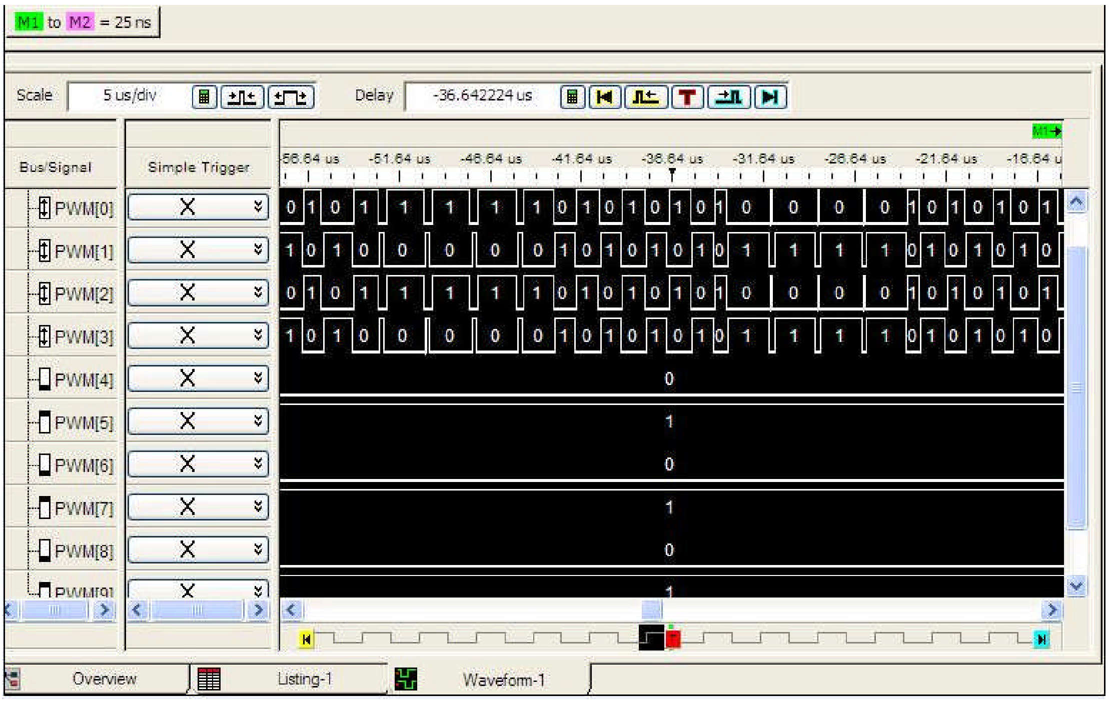
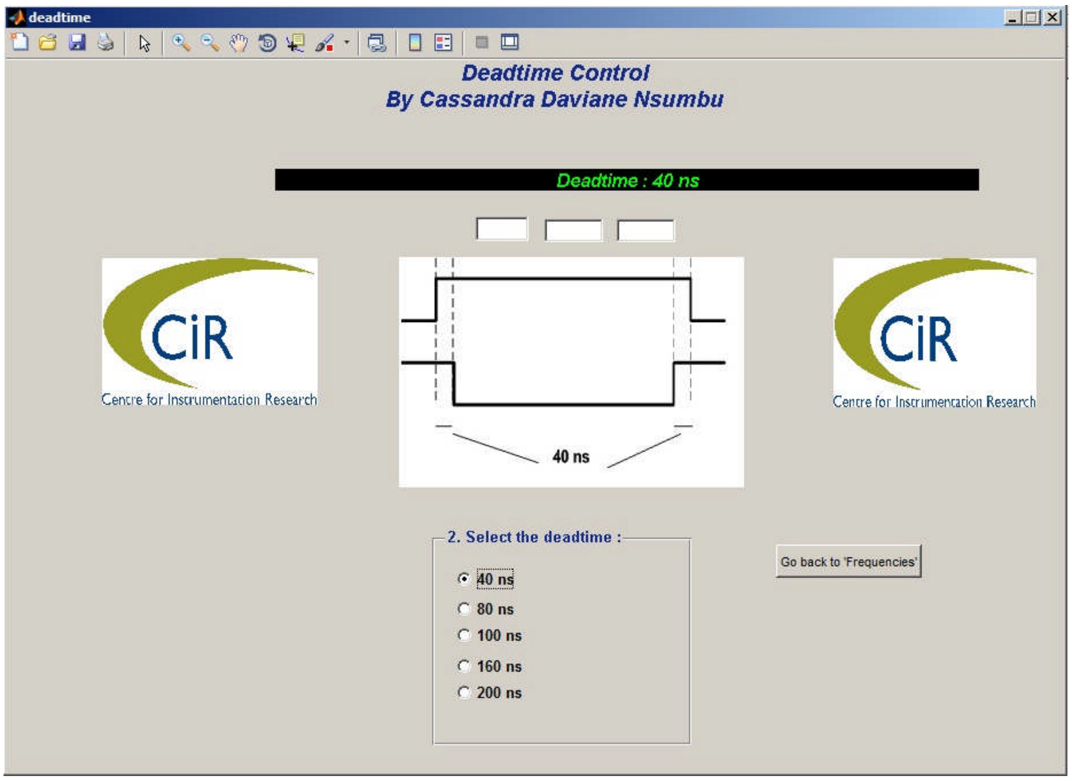


Figure 5.30: Full bridge converter PWM signals (hardware level)

As Figures 5.22 – 5.31 show above, signal shifting is visible and is successfully implemented in the DSP Builder design owing to the implemented delays. The same was observed for application selection control. Nevertheless, the amount of shifting implemented in hardware and I/O pins selection was verified, and proved to be correctly implemented, even when other reference signals were used.

5.6 Deadtime control results

In this section the SCBC’s reconfigurability in terms of deadtime value selection was tested and verified in simulation, and on the hardware level. As mentioned in Chapter Two, deadtime is essential for adequate switching, and must be implemented to avoid short circuits during switching periods of complimentary signals. The following GUI in Figure 5.32 was involved in the deadtime value control. The controller user had the option of choosing between 40, 80, 100, 160 and 200 ns deadtime. Only some of these deadtime values are displayed below though. Different deadtime values are necessary in order to accommodate several power switching devices, which possess different characteristics in terms of rise and fall time requirements. An excess of deadtime could damage some PWM signals, which operate at low frequencies.


deadtime
CiR
Centre for Instrumentation Research
40 ns
CiR
Centre for Instrumentation Research
Go back to 'Frequencies'

Deadtime Control
By Cassandra Daviane Nsumbu

Deadtime : 40 ns

2. Select the deadtime :
- [✓ 40 ns]
- 80 ns
- 100 ns
- 160 ns
- 200 ns

Figure 5.31: Deadtime control GUI

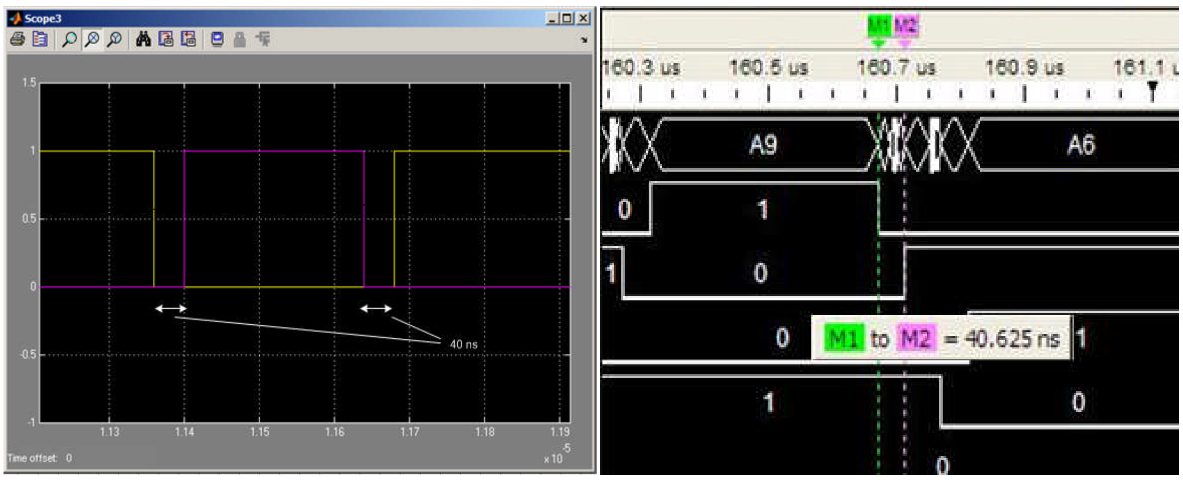


Figure 5.32: 40 ns deadtime implementation results, in simulation and hardware level

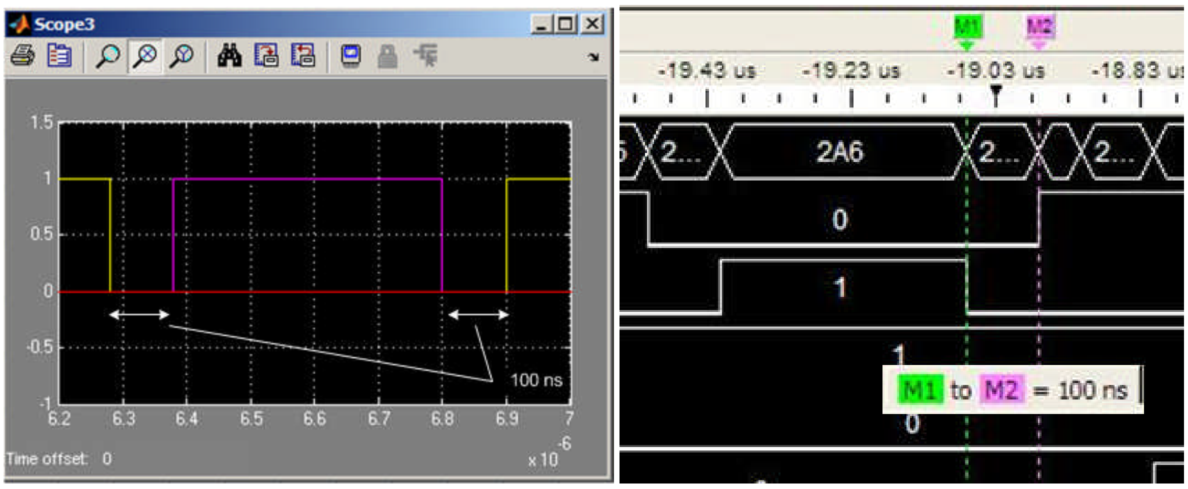


Figure 5.33: 100 ns deadtime implementation results, in simulation and hardware level

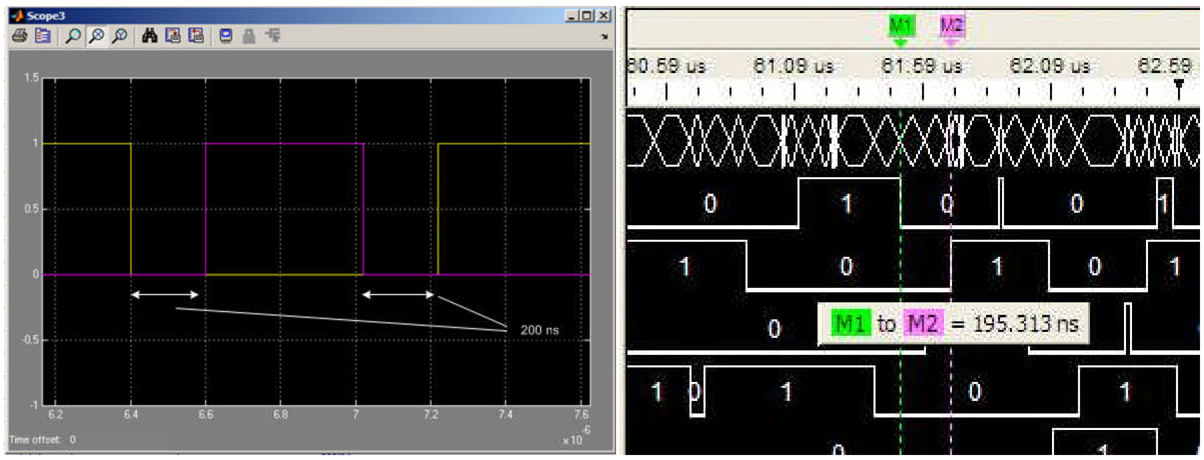


Figure 5.34: 200 ns deadtime implementation results, in simulation and hardware level

Figures 5.33 – 5.35 showcase the successful implementation both in simulation and experimental testing of the developed deadtimes.

5.7 PWM duty cycle control

In this section the SCBC's reconfigurability in terms of duty cycle control was tested and verified in simulation and on the hardware level. In DC-DC converters, output signal regulation is determined by the duty cycle control. The controller user has the option to choose between 15, 25, 50, 75 and 90 % duty cycle. All these values were successfully implemented, simulated, experimentally tested and are displayed below. The GUI, which is presented in Figure 5.36 was involved in the PWM duty cycle control.

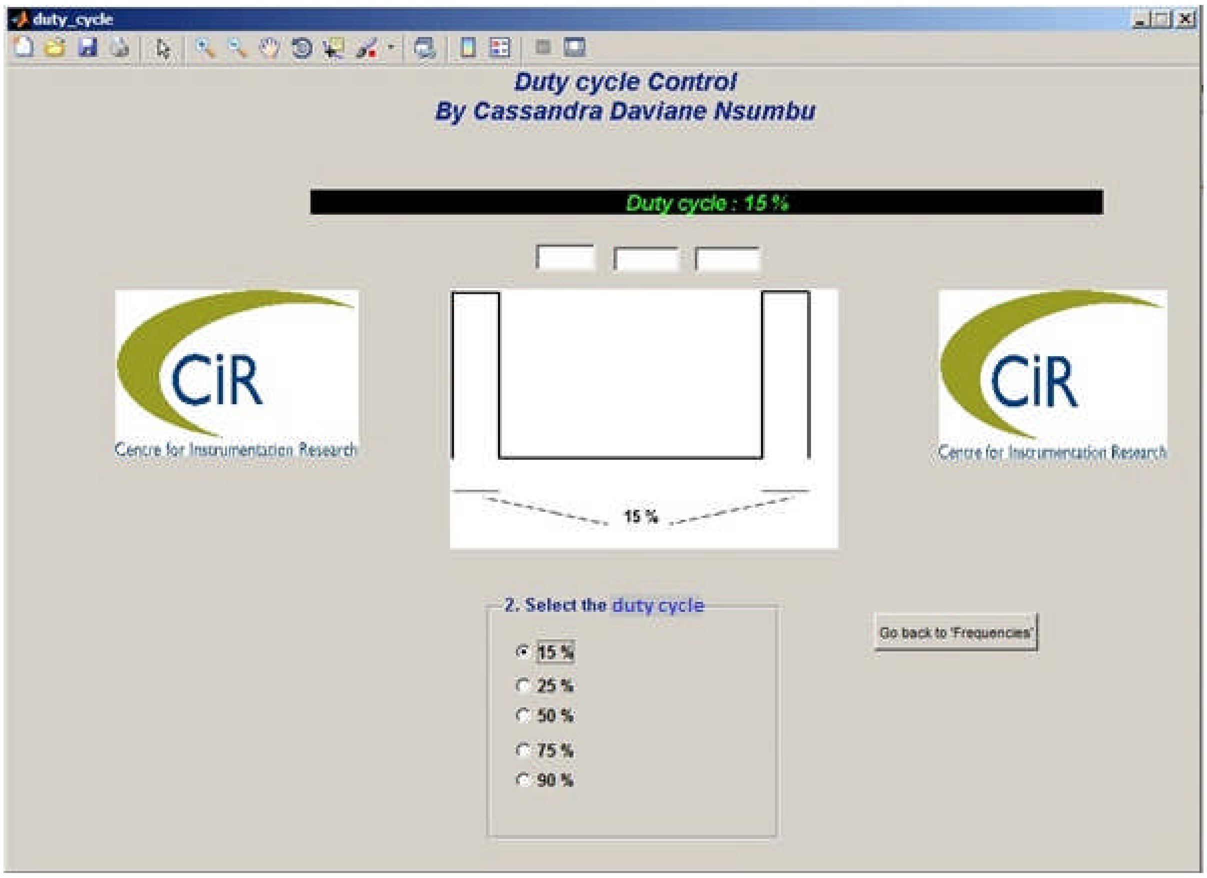


Figure 5.35: PWM duty cycle control GUI

- 15 % duty cycle

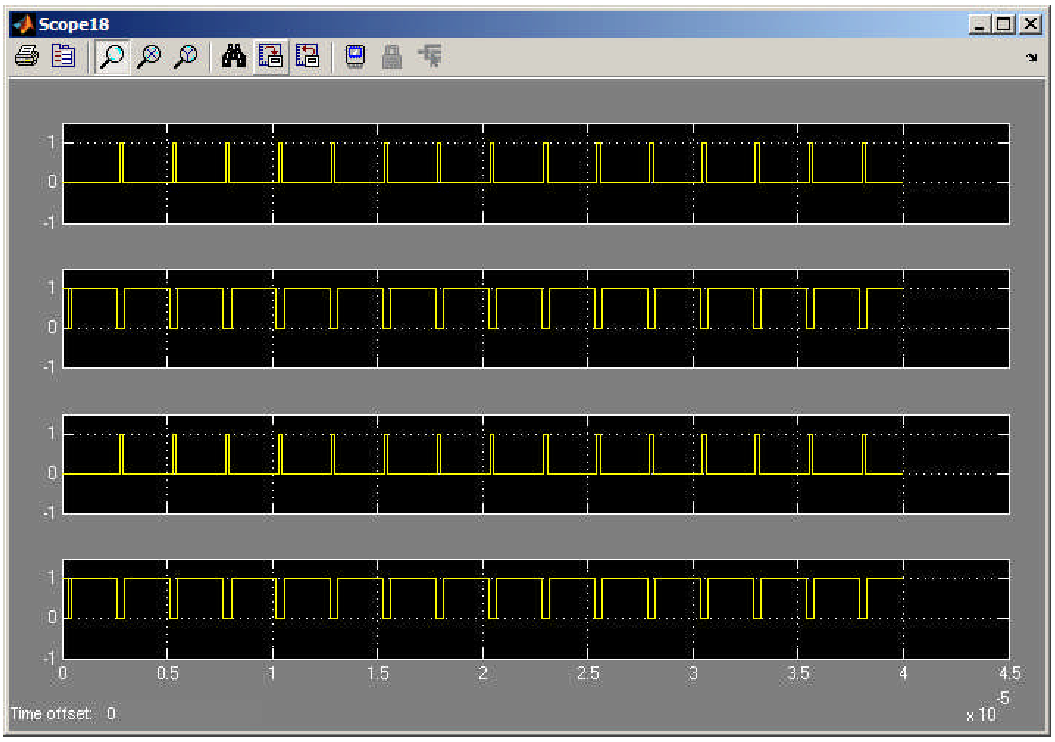


Figure 5.36: 15 % duty cycle (simulation)

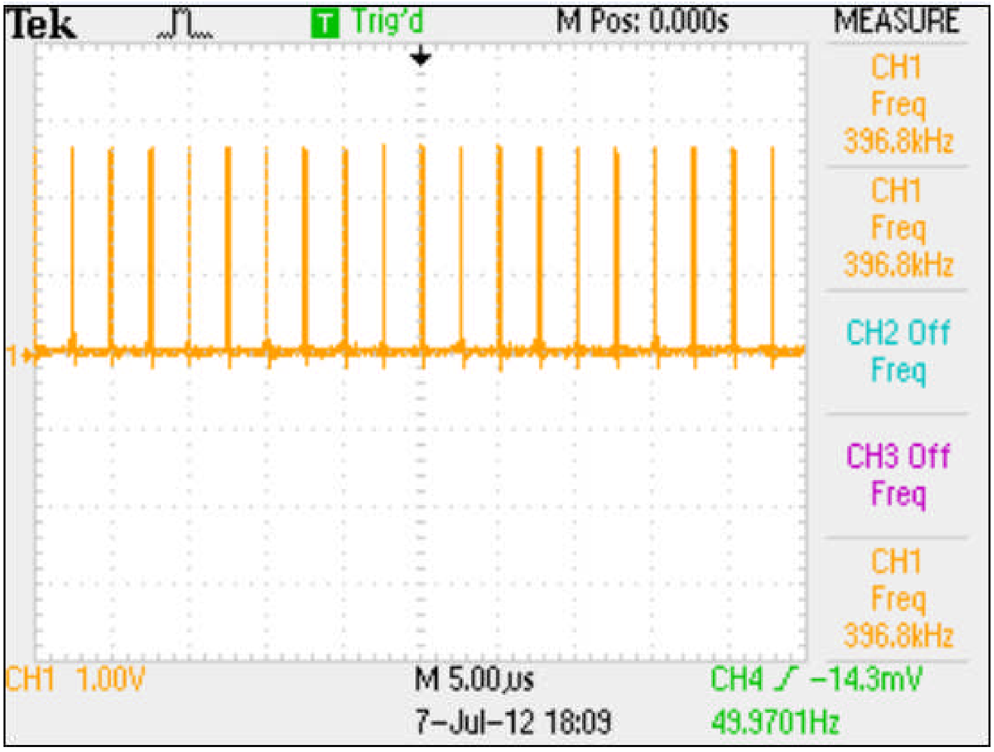


Figure 5.37: 15 % duty cycle (hardware level)

- 25 % duty cycle

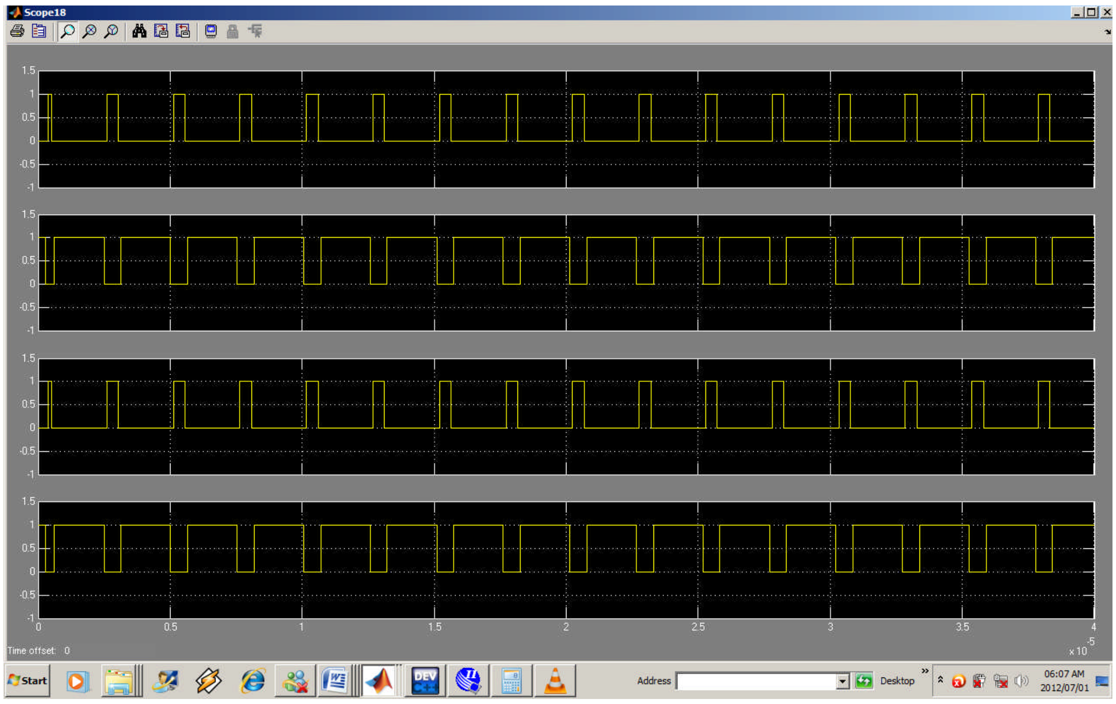


Figure 5.38: 25 % duty cycle (simulation)

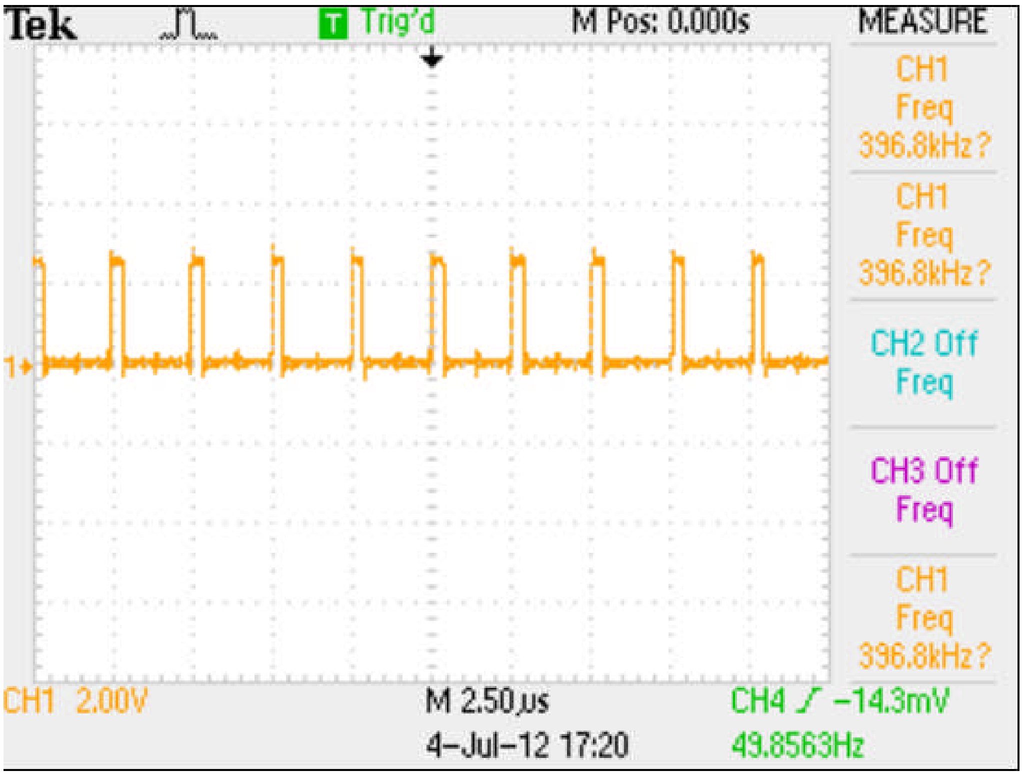


Figure 5.39: 25 % duty cycle (hardware level)

- 50 % duty cycle

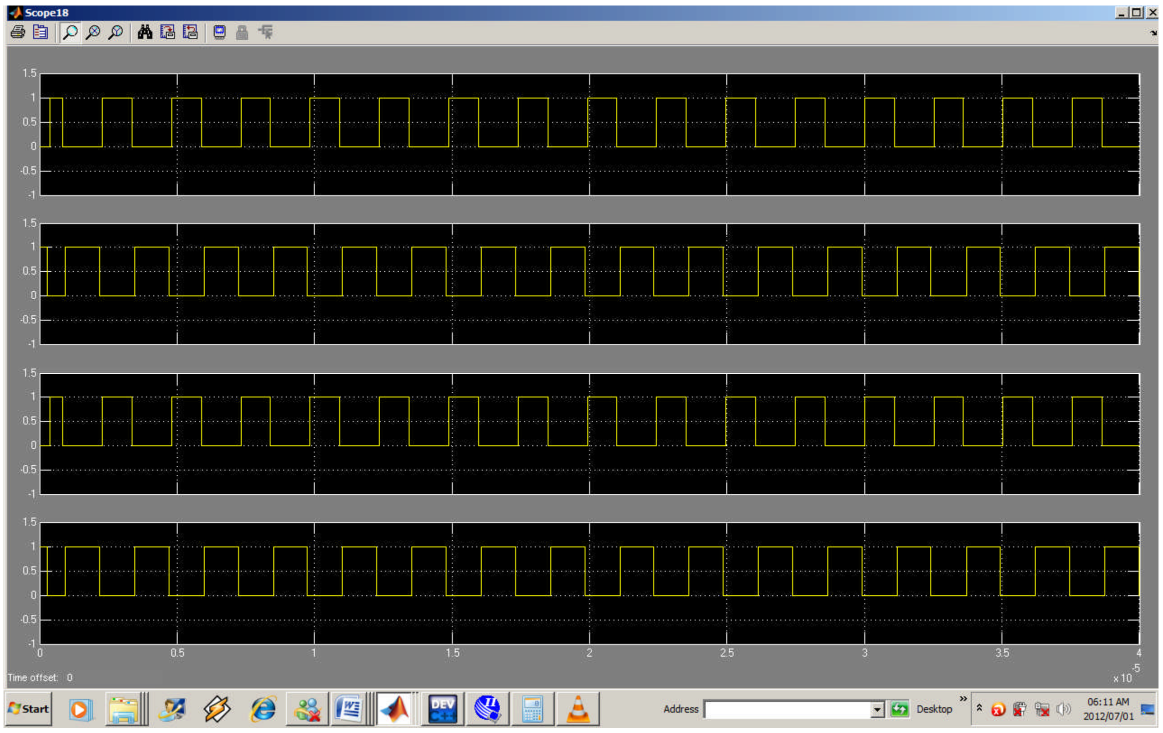


Figure 5.40: 50 % duty cycle (simulation)

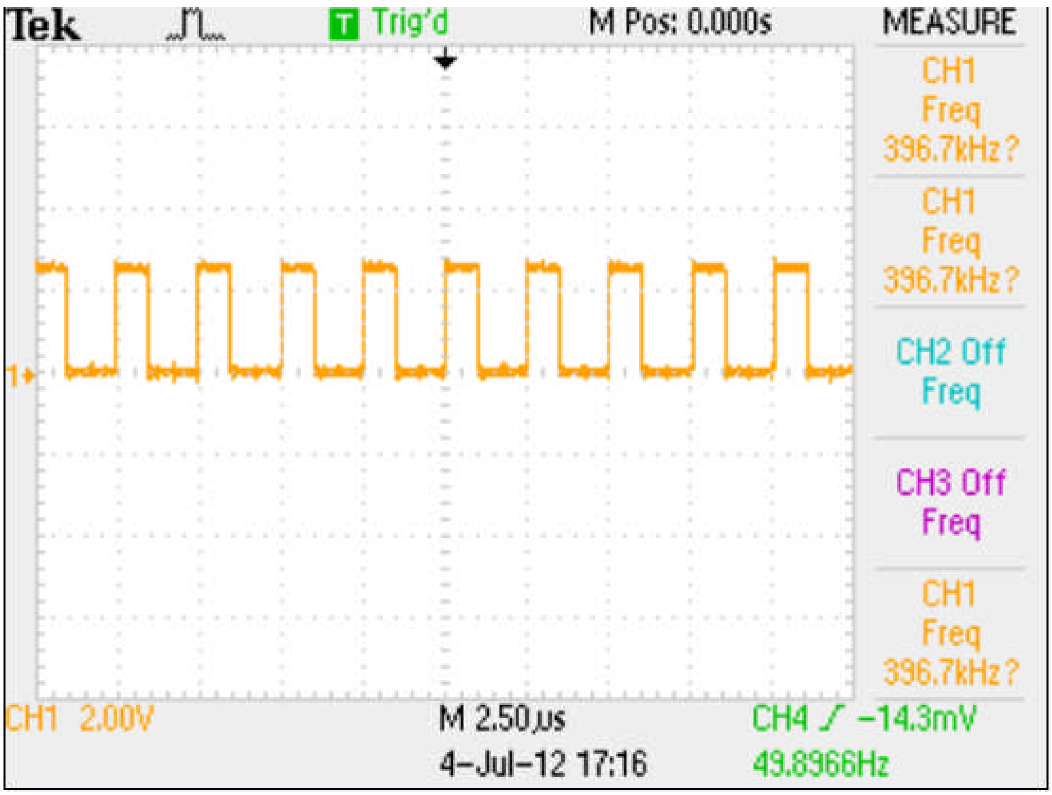


Figure 5.41: 50 % duty cycle (hardware level)

- 75 % duty cycle

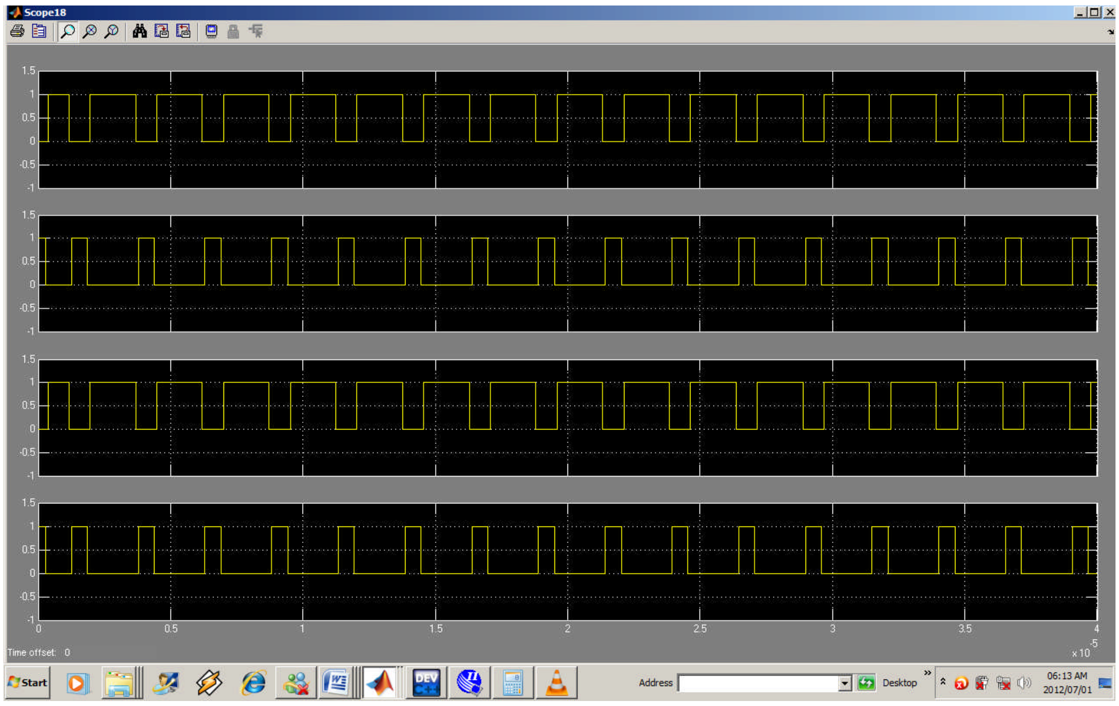


Figure 5.42: 75 % duty cycle (simulation)

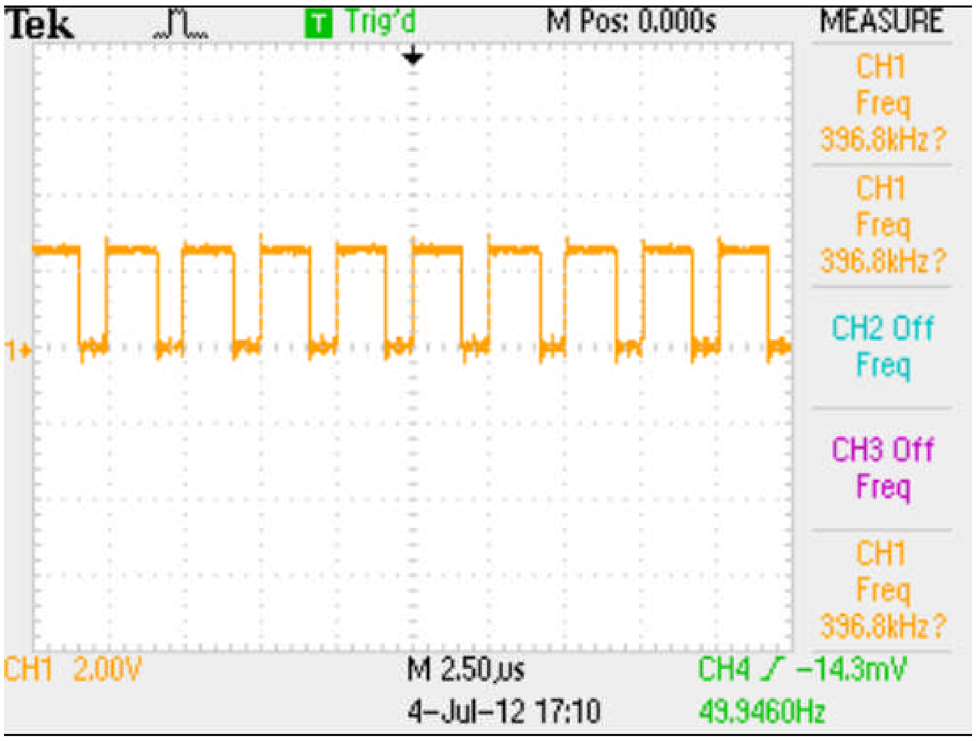


Figure 5.43: 75 % duty cycle (hardware level)

- 90 % duty cycle

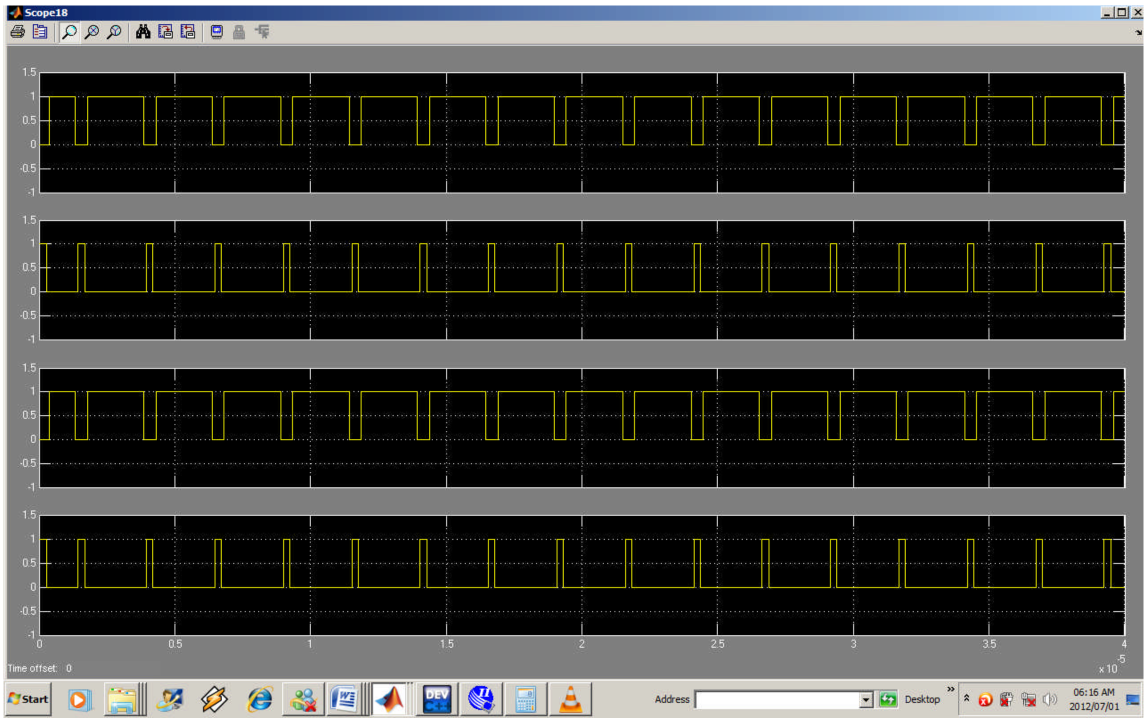


Figure 5.44: 90 % duty cycle (simulation)

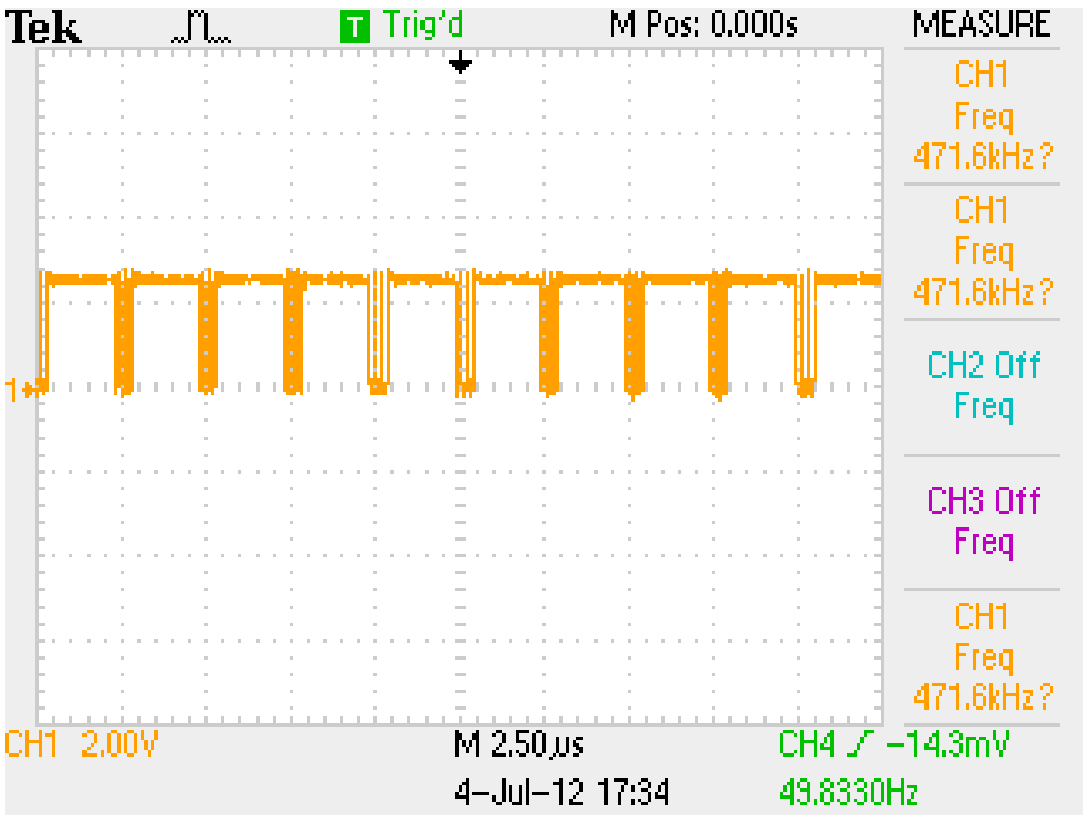


Figure 5.45: 90 % duty cycle (hardware level)

5.8 Summary

This chapter presented results from both the simulation and experimental testing. The simulation testing platform was the DSP Builder/Simulink environment, while experimental testing was performed by analyzing the PWM output signals from the SCBC, and then capturing them on an oscilloscope and logic analyser. Both simulation and experimental results were compared, proved to be similar and correlated to the theory presented in Chapters Two and Four. This also proved the Nios® II‘s overall system design, including sine wave and carrier signal generation and interleaved switching, which were all successfully implemented in the hardware of the FPGA architecture.

CHAPTER 6

Conclusions and future work

6.1 Introduction

This thesis focused on the development of a digital controller for PE applications by implementing a Nios® II processor embedded onto a FPGA platform. The simulation and experimental tests in Chapter Five were performed to analyse and prove the capabilities of the SCBC's individual units, which were designed and developed during Chapter Four. These test results prove that the soft-core based controller worked as planned, and according to this thesis' objectives and given specifications.

6.2 Conclusions

From the presented outcomes in Chapter Five, the established objectives from Chapter One were fulfilled, namely:

- The investigation of different embedded system based controllers and FPGA/soft-core technology;
- The design and implementation of a digital controller, which incorporates a Nios® II processor both in hardware and software aspects. The controller can operate on the mentioned P.E. above; and

- The use of Pulse Width Modulation (PWM) as the control technique, with the internal generation of carrier signals and reference signals with adjustable frequencies, as inputs from the Nios® II processor.

Intensive research on power electronics, embedded systems and, especially FPGA/soft-core systems, was conducted with the help of numerous sources such as theses, journals and conference papers, websites and books in order to gain as much information as possible. These sources were useful in providing insight on these reconfigurable technologies, as well as knowledge on how to implement them successfully.

The embedded Nios® II soft-core processor was utilized to set up and implement an entire control system for PE applications. It proved to be an efficient device and provided a high amount of flexibility and reconfigurability. It was showcased as it creates systems of almost any size by selecting and implementing a specific set of peripherals to meet the designer's needs. This is achieved in less time than expected if done manually with analogue elements. The SCBC design consisted of four units, which were each implemented successfully, as shown in Chapter Five. Reconfigurability is again showcased by easily modifying parameters in the Nios® II C code.

The first unit of the overall design is the DSP Builder design, which consists of a sine generator, carrier signal generator, an adjustable comparator block, application selection and PWM control blocks. This module proved to operate successfully, and allowed the controller to generate switching and reference signals at adjustable frequencies, and to work for a variety of PE applications.

The second unit was the Quartus® II/SOPC system, which incorporates the Nios® II processor as the "brain" of the whole system and the DSP Builder design as a custom peripheral interface. Interaction between the Nios® II processor, SOPC components and the custom peripheral was successfully established. Adequate PWM signals were sent to specific I/O FPGA pins on the HSMC card once they were correctly configured with the Quartus® II Assignment Editor. It can be concluded that DSP Builder and Quartus® II SOPC Builder tools are efficient for the development of reconfigurable, large and complex PE systems onto a single chip without a need to create long and extremely time-consuming HDL codes.

The third unit involved the development of the C code, which operates on the processor via the Nios® II IDE environment. The use of one fourth of a sine wave and LUT methods were introduced and were useful in reducing the usage of FPGA memory resources significantly. As the DSP Builder simulation results from Chapter Five show, the sine wave samples generated by the Nios® II processor proved to be valid and compatible with the other developed portion of hardware to generate the adequate PWM signals as required. Direct interaction of the C code with the DSP Builder was also successfully established to have the processor directly control the controller's hardware.

The fourth unit consisted of the creation of MATLAB® GUIs whose function is to provide a communication interface between the SCBC user and the Nios® II processor and graphic and monitoring tools. They enable the user to enter data as input parameters in the system, and to display simulation results that are also provided. They were all successful in their operations.

The final outcome of this work relates to the successful operation of the SCBC in open loop control mode for half and full bridge, three phase and 5-cell multilevel inverters, as required by the specifications. Its level of reconfigurability was impressive, as the control calculations implemented in the C code running on the processor may be altered, and the same applies to the digital logic in the DSP Builder design as well. This work has increased the understanding of the FPGA embedded Nios® II processor, SOPC systems and their application in power electronics, from theory to hands-on experience. An important lesson was learnt in considering how a small change in software can affect the entire system by ensuring the successful interaction of each module of the system.

6.4 Encountered problems

The biggest challenge in the design and implementation was fitting all the necessary design parts together, while still respecting the limit amount of LE resources and memory space available from the Cyclone® 3C25F324C8 FPGA. Part of this challenge was to develop a wide enough frequency range without altering the system's PLL clock signal, "peripheral_clk". A lot of work and time was spent to write import VHDL codes in a convenient way in order to minimize the size of the overall design, to implement adequate data formatting for all the individual design units, and still obtain the correct results after synthesis. The same was applied when selecting only necessary and relevant SOPC modules to be inserted in the design.

A lack of time and available Nios® II custom instructions for trigonometric functions led to a disadvantage, and a procedure in generating sine wave values through sine tables. A better option would have been the values generation in higher resolutions directly from the soft-core processor without using sine tables.

6.5 Future work

Although a large amount of work was completed, several more ideas were excluded owing to time constraints, and to remain focussed on the main part of the work. These same ideas would be interesting to see implemented in future studies, and are shown below.

- Implementation of closed-loop control mode for PE applications (ADC implementation). Closed-loop control in digital controllers provides several advantages such as disturbance rejection, guaranteed performance even with design uncertainties, high system stability and reduced sensitivity to parameter variations (Geethanjali *et al.*, 2010). The SPI protocol can be used as the communication protocol between the ADC chips and the rest of the Nios® II system.
- Finer resolution for direct reference signal computation and generation from the Nios® II core without the use of sine tables.
- Introduction of a UART module in the SOPC system to provide a more flexible communication interface, with or without MATLAB® GUIs, and with the help of a keypad or LCD module.
- Turning the MATLAB® GUIs into stand-alone applications. A way of eliminating a need for repetitive running of the software in the Nios® II IDE whenever new input data is entered into the system would be of great interest. This will render the Nios® II system mobile and more independent.

List of References

Ababu, T. 2007, Design of Pulse Width Modulation (PWM) and its implementation in Xilinx Field Programmable Gate Array (FPGA), MSc Thesis, Addis Ababa University, Addis Ababa, Ethiopia.

Albiach, J. I. M. 2006. Interfacing a Processor Core in FPGA to an Audio System, MSc Thesis, Linköping Institute of Technology, Linköping, Sweden.

Alcalde, A. L. P., Mohr H.B., Borgonovo D., Mussa S. A. 2009. Experimental validation of the Nios II processor-FPGA on the digital control of PFC converter, Brazilian Power Electronics Conference (COBEP), Florianopolis, Brazil, pp. 895 – 900.

Altera Corporation, 2002. Altera SOPC Solution, 2002 SOPC World Conference, USA. [Online] Available at: http://www.pldworld.com/_exhibit/2002/A_1st_Session.pdf.

Altera Corporation 2004. The World's Most Versatile Processor Nios II.

[Online] Available at: http://www.ee.mut.ac.th/FPGA/SupportFile/Achieva/Nios_II_6.pdf.

Altera Corporation, 2010a. Avalon Interface Specifications, version 1.3. [Online] Available at: http://www.altera.com/literature/manual/mnl_avalon_spec.pdf.

Altera Corporation, 2010b. Cyclone III FPGA Starter Board Reference Manual, version 1.3 [pdf].

[Online] Available at: http://www.altera.com/literature/manual/rm_ciii_starter_board.pdf.

Altera Corporation, 2010c. DSP Builder Handbook Volume 2: Blockset, version 1.0.

[Online] Available at: http://www.altera.com/literature/hb/dspb/hb_dspb_std.pdf.

Altera Corporation, 2010d. Quartus II Handbook, Version 1.3 Volume 5: Embedded Peripherals [pdf].
[Online] Available at: www.altera.com/literature/ug/ug_embedded_ip.pdf.

Altera Corporation, 2010e. Nios II Processor Reference Handbook, version 10.1. [Online] Available at: http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf.

Altera Corporation, 2010f. Nios II Software Developer's Handbook. [Online] Available at: www.altera.com/literature/hb/nios2/n2sw_nii5v2.pdf.

Altera Corporation, 2011. Embedded Design Handbook, version 11.0. [Online] Available at: http://www.altera.com/literature/hb/nios2/edh_ed51008.pdf.

Altera Corporation, 2012, DSP Builder (picture). [Online] Available at: http://www.altera.com/products/software/products/dsp/dsp-builder.html.

Altera Forums, 2010. How the avalon mm slave "address alignment" works? [Forum]. [Online] Available at: http://www.alteraforum.com/forum/showthread.php?t=21324

Anemaet, P., van As, T. 2008. Microprocessor Soft-Cores: An Evaluation of Design Methods and Concepts on FPGAs, Computer Architecture (Special Topics), Publication View. 42624821, ET4 078, The Netherlands.

Arbinger D. & Erdmann J. 2006. Designing with an embedded soft-core processor, the Plexus Technology Group**.** [Online] Available from http://www.embedded.com/design/mcus-processors-and-socs/4006632/Designing-with-an-embedded-soft-core-processor. Accessed on 25/11/12.

Bakar, A. A., Abdullah, F. S. & Ahmad M. Z. 2009. Design of FPGA-based SPWM Single Phase Full Bridge Inverter, *Proceedings of Malaysian Technical Universities Conference on Engineering and Technology (MUCEET) 2009*, Johor, Malaysia.

Barr, M. 1999. *Programming Embedded Systems in C and C++*, first edition, O'Reilly Media, pp.18.

Bengtsson, H. & Jonssson, M. 2009. Control of Switch-mode Power Supplies: A digital approach for half-bridge converters, MSc Thesis. Chalmers University of Technology, Göteborg, Sweden.

Chamberlain, R., Lockwood, J., Gayen, S., Hough, H. & Jones, P. 2005. Use of a Soft-Core Processor in a Hardware/Software Co-design Laboratory, *Proceedings of 2005 IEEE International Conference on Microelectronic Systems Education (MSE'05)*, California, USA, pp. 97-98.

Chapman, S. J. 2002. *MATLAB programming for engineers*, Second Edition, Brooks/Cole-Thomson Learning, pp. 14.

Daya, B. 2009. Rapid Prototyping of Embedded Systems Using Field Programmable Gate Arrays, BSc thesis, University of Florida, Florida, USA.

De Castro, A., Zumel, P., García, O., Riesgo, T. & Uceda, J. 2003. Concurrent and Simple Digital Controller of an AC/DC Converter with Power Factor Correction Based on FPGA, *IEEE Transaction on Power Electronics*, Volume 18, no. 1, Texas, USA, pp. 334-343.

EE TimesAsia, 1997. New MATLAB software generates C code (article), EE TimesAsia DSP DesignLine. [Online] Available at: http://www.eetasia.com/ART_8800482220_499495_NP_6a075c56.HTM. Accessed on 03/09/11.

Erickson, R.W. & Maksimovic, D. 2004. *Fundamentals of Power Electronics,* Second Edition, Springer, pp. 1-2, pp. 7.

Francis, J. & Boroyevich, D. 2001. Design of a Universal Controller for Distributed Control and Power Electronics Applications, *Proceedings of Center for Power Electronics Systems (CPES) and National Science Foundation (NSF)/Industry Annual Review*, PEBB Publications, Virginia, USA.

[Online] Available from http://web-cat.cs.vt.edu/PEBB/R94_Francis.pdf [12/10/2010].

Geethanjali, P., Vijaya, P. P., Kowsalya, M. & Raju J. 2010. Design and Simulation of Digital PID Controller for Open loop and Closed Loop Control of Buck Converter, *International Journal of Computer Science and Technology (IJCST),*Volume 1, Issue 2, Vellore, India, pp 202 – 206.

Guan, T. G. 2010. Wireless Channel Model Development using FPGA, BSc thesis, University of Technology Malaysia (UTM), Johor, Malaysia.

Grout, I. 2008. *Digital systems design with FPGAs and CPLDs*, Newnes / Elsevier publishers, Oxford, UK, pp. 28, pp. 63 – 64, pp. 195.

Hamblen, J.O. & Hall, T.S. 2006. Using System-on-a-Programmable-Chip Technology to Design Embedded Systems, *Proceedings of International Journal of Computers and Their Applications (IJCA)*, Volume 13, No. 3**,** pp. 1-11.

Hamblem, J. O. & Furman, M. D. 2002. *Rapid Prototyping of Digital Systems,* SOPC edition, Kluwer Academic Publishers, pp. 326.

Van Heerden, G. J. 2003. Design and Implementation of a DSP Based Controller for Power Electronic Applications, MSc Thesis, University of Stellenbosch, Stellenbosch, South Africa.

Jooste, C. & Wilkinson, R.H. 2009. Development of a Generic Digital Controller for Power Electronic Applications, *Proceedings of South African Universities Power Engineering Conference (SAUPEC),* Stellenbosch, South Africa, pp. 163-166.

Joshi, N. N., Dakhole, P. K. & Zode, P. P. 2009. Embedded Web Server on Nios II Embedded FPGA Platform, *Second International Conference on Emerging Trends in Engineering and Technology, ICETET-09,* Nagpur, India, pp. 372 – 377.

Kariyappa, B. & Uttara Kumari, U. 2008. FPGA Based Speed Control of AC Servomotor Using Sinusoidal PWM, *International Journal of Computer Science and Network Security (IJCSNS)*, Volume 8 No.10, Bangalore-59, India, pp. 346-350.

Kimmo, R. 2006. FPGA-Based control design for power electronic applications, PhD thesis, Lappeenranta University of Technology, Lappeenranta, Finland. [Online]Available from https://oa.doria.fi/bitstream/handle/10024/31138/TMP.objres.472.pdf.

Klumpner, C. 2004a. Inverters (pdf), Power Electronics course, University of Nottingham.

[Online] Available at:
http://hermes.eee.nott.ac.uk/teaching/h5cpe2/Inverters%20%28with%20pics%29.pdf.

Klumpner, C. 2004b. Inverters - Triangular PWM (Photograph), Power Electronics course, University of Nottingham.
[Online] Available at: http://hermes.eee.nott.ac.uk/teaching/h5cpe2/triangular_PWM.gif.

Koskinen, T. J. 2009. Metadata-based Automated Configuration of System-on-Chip, MSc thesis, Information Technology Department Council meeting, Tampere University of Technology, Tampere, Finland.

Koutroulis, E., Dollas, A. & Kalaitzakis, K. 2006. High-Frequency Pulse Width Modulation Implementation using FPGA and CPLD ICs, *Journal of Systems Architecture*, Volume 52, Technical University of Crete, Crete, Greece, pp. 332–344.

Krah, J., Höltgen, M., Rath, A. & Ritcher, R. 2011. FPGA-based Control of Three-Level Inverters, PCIM Europe 2011 International Exhibition & Conference for Power Electronics, Intelligent Motion, Power Quality, Nuremberg, Germany, pp. 66.

Leonov, M. 2009. Method and Implementation of Multi-Channel Correlation in the Hybrid CPU+FPGA System, MSc thesis, Auckland University of Technology (AUT), New Zealand.

Luo, F.L., Ye, H. & Rashid, M.H. 2005. *Digital Power Electronics and Applications*. Academic Press, pp. 32, 88 – 89.

Maksimovic, D., Zane, R. & Erickson, R. 2004. Impact of digital control in power electronics. *Proceedings of the 16th International Symposium on Power Semiconductor Devices and ICs (ISPSD)*, Kitakyushu, Japan, pp. 2 – 22.

Malik, J. & P, Ratna S. 2010. Design of MP3 Player Application with Nios II Embedded Evaluation Kit, BTech thesis, National Institute of Technology (NIT), Rourkela, India.

Martin, T.W. & Ang, S.S. 1995. Digital Control for Switching Converters. *Proceedings of the IEEE International Symposium on Industrial Electronics*, Arkansas, USA, pp. 480-484.

Mastascusa, E. J. 2002.Control Systems lessons, Bucknell University. [Online] Available from http://www.facstaff.bucknell.edu/mastascu/econtrolhtml/Intro/Intro1.html.

Mohan, N., Undeland, T. M., Robbins, W. B. 2003. *Power Electronics - Converters, Applications and Design*, Second Edition. John Wiley & Sons, Inc.

Molepo, S. A. 2003. A Multilevel Inverter for DC Reticulation, MSc thesis, University of Stellenbosch, Stellenbosch, South Africa.

Noman, M. 2004. Low-Power Embedded Processor, ECE 290 Final Report, University of Connecticut, Connecticut, USA.

Nurmi, J. 2007. *Processor Design: System-On-Chip computing for ASICs and FPGAs*. Springer publishers, pp. 230.

Pedroni, A. V. 2004. *Circuit Design with VHDL*, first edition, The MIT Press, London, England, pp.3.

Prodic, A., Maksimovic, D. & Erickson, R. W. 2001. Design and Implementation of a Digital PWM Controller for a High-Frequency Switching DC-DC Power Converter, *27th Annual Conference of the IEEE Industrial Electronics Society (IECON)*, University of Colorado, Boulder, USA, pp. 893 - 898.

Quasim, S. M., Telba, A. A. & AlMazroo A. Y. 2010. FPGA Design and Implementation of Matrix Multiplier, Architectures for Image and Signal Processing Applications, *International Journal of Computer Science and Network Security (IJCSNS)*, Volume 10 No.2, pp. 168 - 176.

Sicking, J. 2005. Implementation of Asynchronous Communication for ForSyDe in Hardware and Software, MSc Thesis, Institute of Microelectronics and Information Technology, Royal Institute of Technology (KTH), Stockholm, Sweden.

Simka, M. 2002. Conception of Connection of Embedded Processor to Arithmetic Co-processor in SOPC Altera, MSc thesis, Technical University of Kosice, Kosice, Slovakia.

Schietekat, L. M. 2011. Design and Implementation of the Main Controller of a Solid-State Transformer, MSc thesis, University of Stellenbosch, Stellenbosch, South Africa.

Singh, K. B. K. 2008. *Power Electronics*. Second Edition, McGraw – Hill, pp. 11 – 12.

Speaking Technology – Electrical and Electronic Engg, 2012. Open & Closed Loop Control System - Advantages & Disadvantages. [Online] Available from http://amitbiswal.blogspot.com/2012/01/open-closed-loop-system-advantages.html. Accessed on 15/11/2012.

Tong, J.G., Anderson, I.D.L. & Khalid, M.A.S. 2006. Soft-Core Processors for Embedded Systems, *Proceedings of the 18th International Conference on Microelectronics (ICM)*, Ontario, Canada, pp. 170 - 173.

Urriza, I., Barragán, L. A., Navarro, D., Artigas, J. I., Lucía, O. & Jiménez, O. 2009. FPGA Embedded Soft-Core Processor Implementation of a Digital Controller for a DC-DC

Converter, *Proceedings of the 14th Conference on Design of Circuits and Integrated Systems*, Zaragoza, Spain, pp. 72-77.

[Online] Available from http://dcis2009.unizar.es/FILES/CR2/p72.pdf . Accessed on 12/10/2010.

Venkatesh, S. C. 1994. The Development of a Digital Controller for a Three Phase Induction Motor, MSc. Thesis, Massachusetts Institute of Technology, Massachusetts, USA.

Wilkinson, R.H. 1997. Topology, Control and Development of High Power Multilevel Converters, MSc thesis, University of Stellenbosch, Stellenbosch, South Africa.

Wilkinson, R.H. 2004. Natural Balancing of Multicell Converters, PhD thesis, University of Stellenbosch, Stellenbosch, South Africa.

Williams, B. W. Power inverters [pdf].
[Online] Available at: http://www.eee.strath.ac.uk/~bwwilliams/Book/Chapter 14.pdf. Accessed on 19/10/2010.

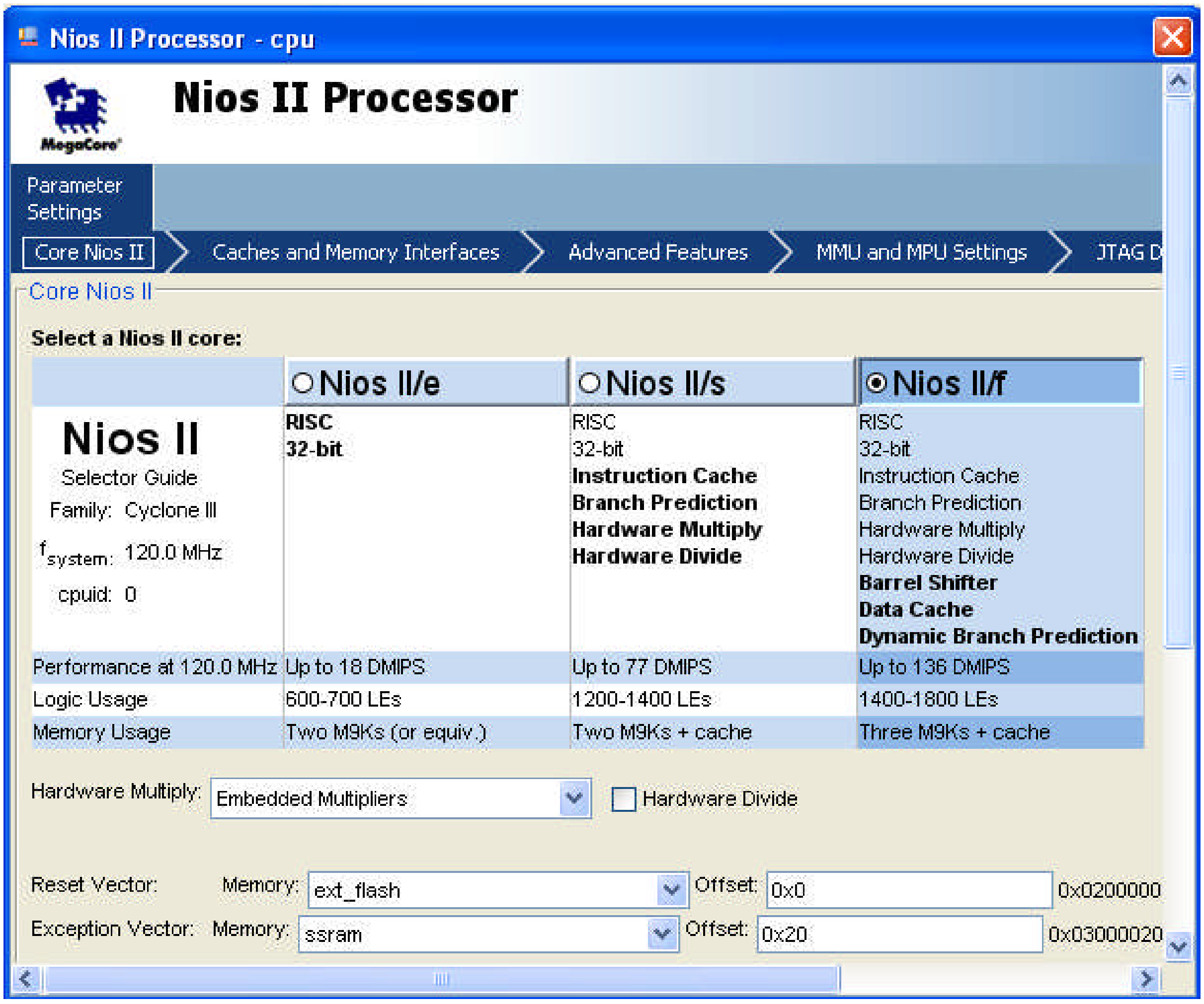
Zeidman, C. 2004. Zeidman Technologies [article].

[Online] Available at: http://chipdesignmag.com/display.php?articleId=386. Accessed on 03/05/2010.

Zhang, D. 2006. A Stochastic Approach to Digital Control Design and Implementation in Power Electronics, PhD Thesis, the Florida State University College of Engineering, Florida, USA.

Zhang, Z. & Dong, F. 2010. A Harmonic Signal Generator Based on DDS and SOPC, Chinese Control and Decision Conference (CCDC), Tianjin University, Tianjin, China, pp. 1542 – 1547.

Appendix A – SOPC Builder configuration settings



Select a Nios II core:

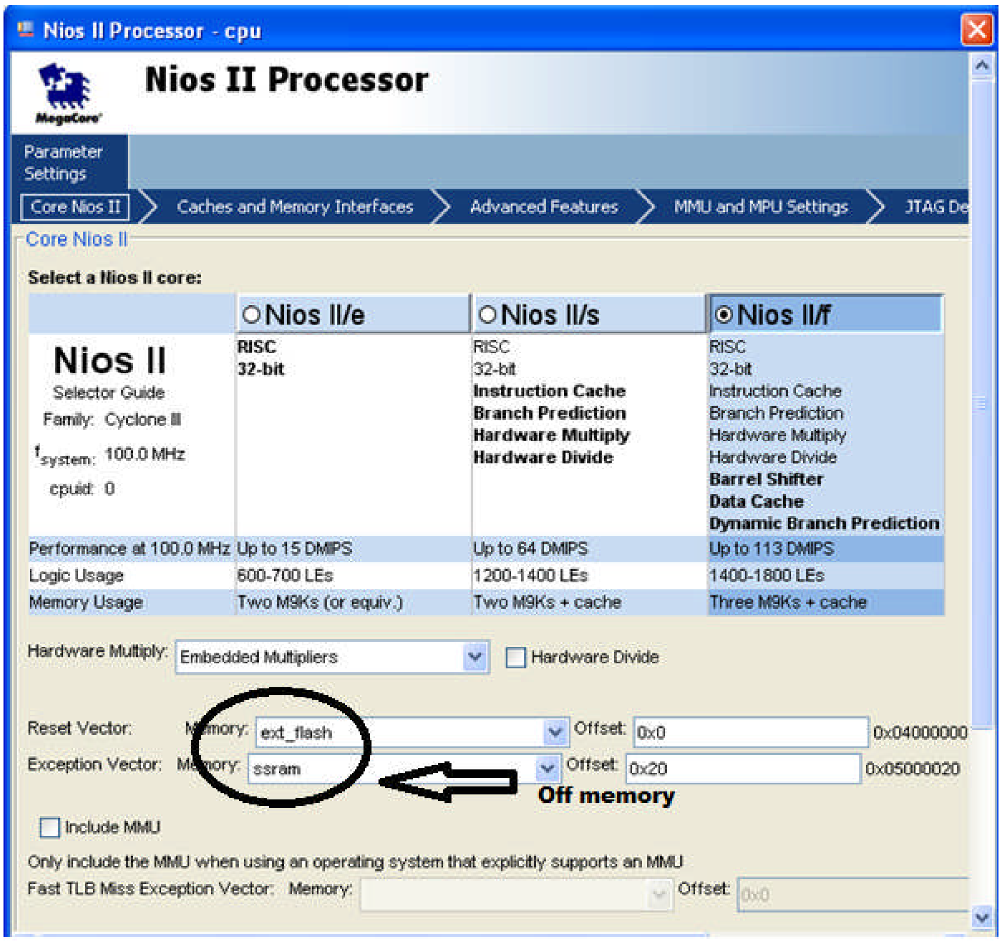
	○ Nios II/e	○ Nios II/s	● Nios II/f
Nios II Selector Guide Family: Cyclone III f_{system}: 120.0 MHz cpuid: 0	**RISC** **32-bit**	RISC 32-bit **Instruction Cache** **Branch Prediction** **Hardware Multiply** **Hardware Divide**	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide **Barrel Shifter** **Data Cache** **Dynamic Branch Prediction**
Performance at 120.0 MHz	Up to 18 DMIPS	Up to 77 DMIPS	Up to 136 DMIPS
Logic Usage	600-700 LEs	1200-1400 LEs	1400-1800 LEs
Memory Usage	Two M9Ks (or equiv.)	Two M9Ks + cache	Three M9Ks + cache

Hardware Multiply: Embedded Multipliers ☐ Hardware Divide

Reset Vector: Memory: ext_flash Offset: 0x0 0x0200000

Exception Vector: Memory: ssram Offset: 0x20 0x03000020

Figure A.1: Nios® II processor core configuration


Nios II Processor - cpu

Nios II Processor

MegaCore

Parameter Settings

Core Nios II > Caches and Memory Interfaces > Advanced Features > MMU and MPU Settings > JTAG De

Core Nios II

Select a Nios II core:

	○ Nios II/e	○ Nios II/s	[✓ Nios II/f]
Nios II Selector Guide Family: Cyclone II f_{system}: 100.0 MHz cpuid: 0	**RISC** **32-bit**	RISC 32-bit **Instruction Cache** **Branch Prediction** **Hardware Multiply** **Hardware Divide**	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide **Barrel Shifter** **Data Cache** **Dynamic Branch Prediction**
Performance at 100.0 MHz	Up to 15 DMIPS	Up to 64 DMIPS	Up to 113 DMIPS
Logic Usage	600-700 LEs	1200-1400 LEs	1400-1800 LEs
Memory Usage	Two M9Ks (or equiv.)	Two M9Ks + cache	Three M9Ks + cache

Hardware Multiply: Embedded Multipliers ☐ Hardware Divide

Reset Vector: Memory: ext_flash Offset: 0x0 0x04000000

Exception Vector: Memory: ssram Offset: 0x20 0x05000020

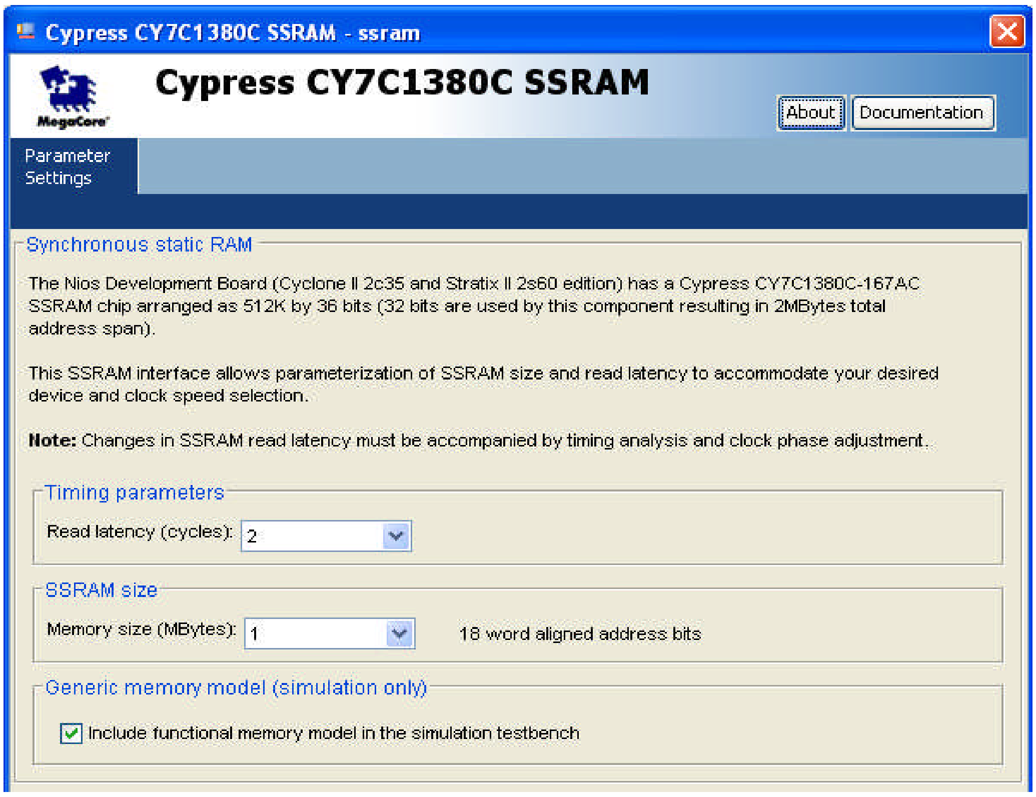
Off memory

☐ Include MMU

Only include the MMU when using an operating system that explicitly supports an MMU

Fast TLB Miss Exception Vector: Memory: Offset: 0x0

Figure A.2: Off-chip memory cores utilized by the Nios® II core CPU



Synchronous static RAM

The Nios Development Board (Cyclone II 2c35 and Stratix II 2s60 edition) has a Cypress CY7C1380C-167AC SSRAM chip arranged as 512K by 36 bits (32 bits are used by this component resulting in 2MBytes total address span).

This SSRAM interface allows parameterization of SSRAM size and read latency to accommodate your desired device and clock speed selection.

Note: Changes in SSRAM read latency must be accompanied by timing analysis and clock phase adjustment.

Timing parameters

Read latency (cycles): 2

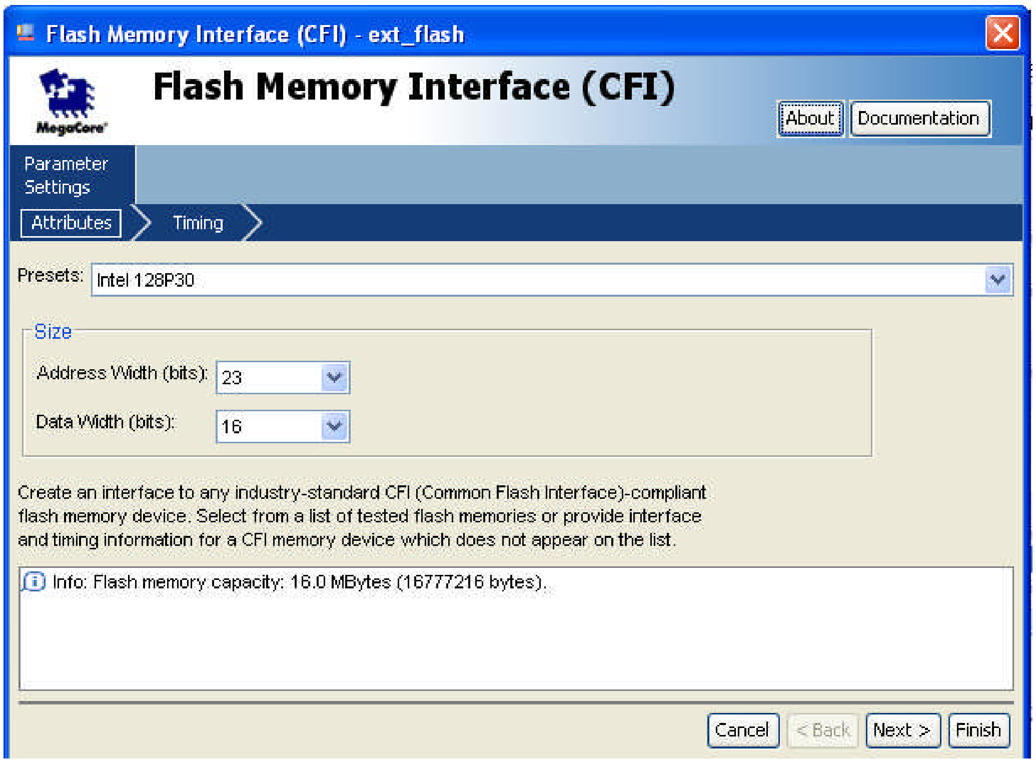
SSRAM size

Memory size (MBytes): 1   18 word aligned address bits

Generic memory model (simulation only)

✓ Include functional memory model in the simulation testbench

Figure A.3: SSRAM off-chip memory configuration



Presets: Intel 128P30

Size

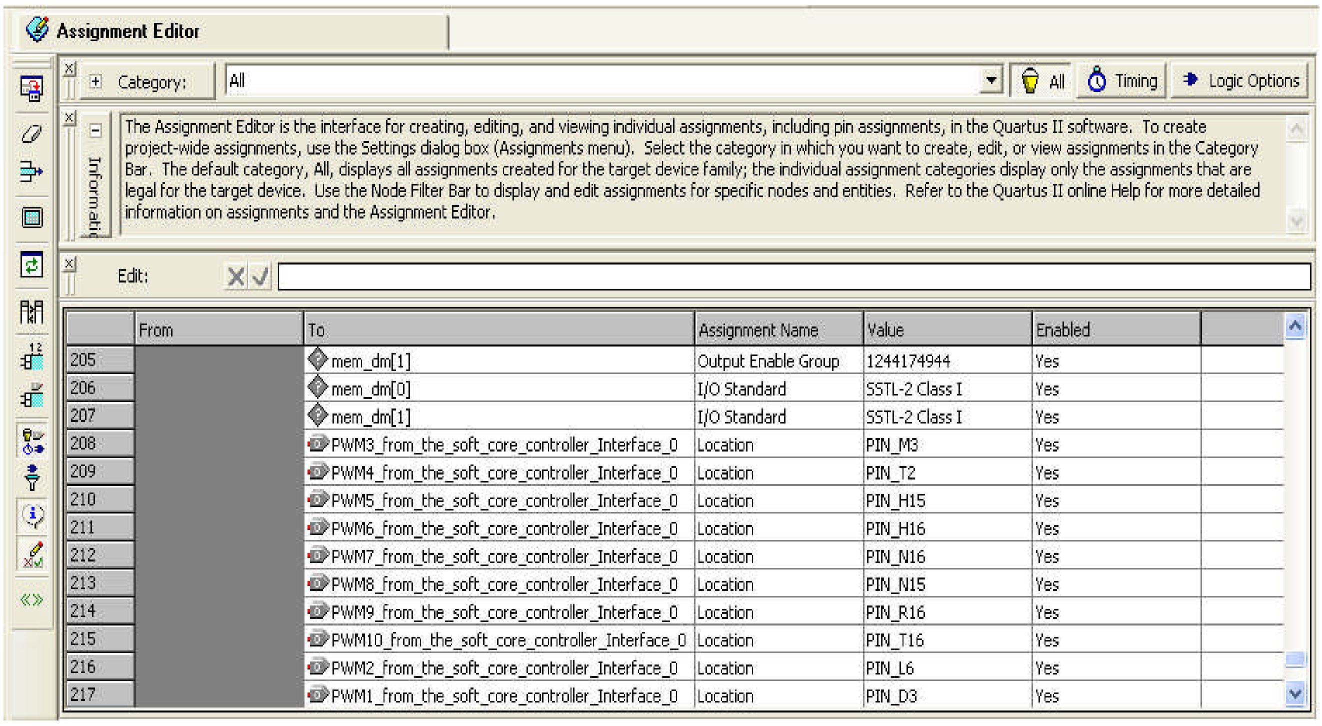
Address Width (bits): 23

Data Width (bits): 16

Create an interface to any industry-standard CFI (Common Flash Interface)-compliant flash memory device. Select from a list of tested flash memories or provide interface and timing information for a CFI memory device which does not appear on the list.

Info: Flash memory capacity: 16.0 MBytes (16777216 bytes).

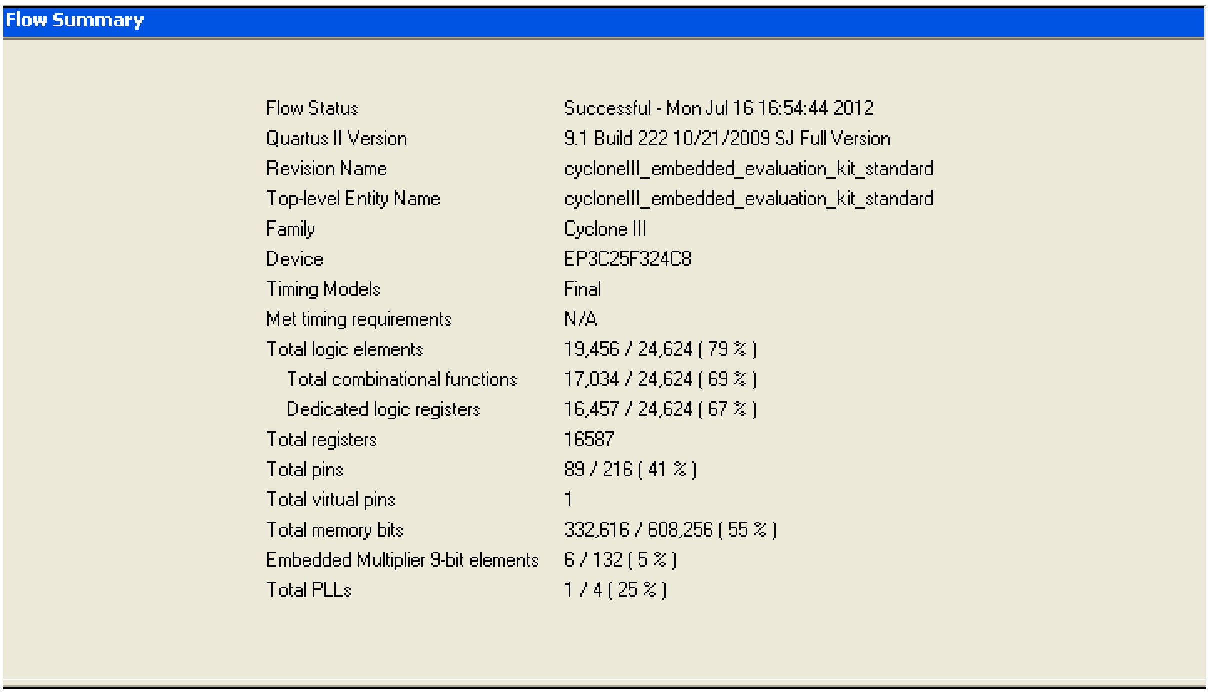
Figure A.4: FLASH off-chip memory configuration



The Assignment Editor is the interface for creating, editing, and viewing individual assignments, including pin assignments, in the Quartus II software. To create project-wide assignments, use the Settings dialog box (Assignments menu). Select the category in which you want to create, edit, or view assignments in the Category Bar. The default category, All, displays all assignments created for the target device family; the individual assignment categories display only the assignments that are legal for the target device. Use the Node Filter Bar to display and edit assignments for specific nodes and entities. Refer to the Quartus II online Help for more detailed information on assignments and the Assignment Editor.

	From	To	Assignment Name	Value	Enabled
205		mem_dm[1]	Output Enable Group	1244174944	Yes
206		mem_dm[0]	I/O Standard	SSTL-2 Class I	Yes
207		mem_dm[1]	I/O Standard	SSTL-2 Class I	Yes
208		PWM3_from_the_soft_core_controller_Interface_0	Location	PIN_M3	Yes
209		PWM4_from_the_soft_core_controller_Interface_0	Location	PIN_T2	Yes
210		PWM5_from_the_soft_core_controller_Interface_0	Location	PIN_H15	Yes
211		PWM6_from_the_soft_core_controller_Interface_0	Location	PIN_H16	Yes
212		PWM7_from_the_soft_core_controller_Interface_0	Location	PIN_N16	Yes
213		PWM8_from_the_soft_core_controller_Interface_0	Location	PIN_N15	Yes
214		PWM9_from_the_soft_core_controller_Interface_0	Location	PIN_R16	Yes
215		PWM10_from_the_soft_core_controller_Interface_0	Location	PIN_T16	Yes
216		PWM2_from_the_soft_core_controller_Interface_0	Location	PIN_L6	Yes
217		PWM1_from_the_soft_core_controller_Interface_0	Location	PIN_D3	Yes

Figure A.5: Output pins assignments on Quartus® II Assignment Editor



Flow Status	Successful - Mon Jul 16 16:54:44 2012
Quartus II Version	9.1 Build 222 10/21/2009 SJ Full Version
Revision Name	cycloneIII_embedded_evaluation_kit_standard
Top-level Entity Name	cycloneIII_embedded_evaluation_kit_standard
Family	Cyclone III
Device	EP3C25F324C8
Timing Models	Final
Met timing requirements	N/A
Total logic elements	19,456 / 24,624 (79 %)
Total combinational functions	17,034 / 24,624 (69 %)
Dedicated logic registers	16,457 / 24,624 (67 %)
Total registers	16587
Total pins	89 / 216 (41 %)
Total virtual pins	1
Total memory bits	332,616 / 608,256 (55 %)
Embedded Multiplier 9-bit elements	6 / 132 (5 %)
Total PLLs	1 / 4 (25 %)

Figure A.6: Hardware resource usage displayed by the Quartus® II Compilation Report