A FPGA Test Platform for Data Converters

Venkatesh Karra

Submitted for the award of Master of Engineering

University of Limerick

Supervised by Dr. Brendan Mullane
Joint Supervisor Dr. Ciaran MacNamee

Submitted to the University of Limerick, November, 2017
Abstract

A FPGA Test Platform for Data Converters

Venkatesh Karra

Data converters are used in mixed-signal designs and are important electronic devices with applications in many areas like consumer electronics, communications, computing, control and instrumentation. Design of these data converters to high performance is a challenging task whereas testing of these data converters for correct functionality has also got equal importance.

The aim of this thesis is to develop a method for testing data converters using Verilog HDL code and to implement a platform design on a FPGA board. This platform uses the Fast Fourier Transforms as a spectral analysis method for determining the dynamic performance of data converters and associated logic.

A focus of attention is to understand the fundamentals of digital signal processing and digital design using good Verilog HDL coding practice. This is achieved through the design and application of a 7-tap FIR filter both in MATLAB and Verilog RTL.

This design system is achieved by developing an FPGA test platform using a Xilinx Zynq board FPGA which supports a 1M Sample ADC along with an Analog Mixed Signal card containing external Digital to Analog converters for signal generation. The FPGA supports a sine wave generator that uses sine wave signals as test vectors that are fed to DAC so that an analogue signal is provided to the ADC. The output samples are sampled by the ADC and stored in memory that is then accessed through a serial to parallel interface for spectral analysis in MATLAB.

This design is furthered with development of a potential application where the ADC outputs are applied to Finite Impulse Response filters for further analysis. The final results of this thesis include FFT plots and also simulated timing diagrams.
Declaration

I hereby declare that this thesis is entirely my own work and that it has not been submitted for a degree in part or in full to any other University.

__________________________
Venkatesh Karra
Acknowledgements

I would like to thank my supervisor and mentor Dr. Brendan Mullane for guiding me through the Master’s degree.

I would also like to thank Dr. Ciaran MacNamee for his help and advice throughout.

I want to thank University of Limerick and especially Circuits and Systems Research Center (CSRC) with all its team Mark, Tony, Danny, Vincent, Shantanu, Petar, James, and Vincent.

Last, but not least, I wish to thank my brother Anand Karra for his moral support and also my parents Nageswara Rao Karra, Sudha Kumari Karra. Without whom my studies in India and Limerick would not have been possible.
Dedication

To,
Anand
# Table of Contents

Abstract ................................................................................................................................. i

Declaration .............................................................................................................................. iii

Acknowledgements .................................................................................................................. v

Dedication ............................................................................................................................... vii

Table of Contents ..................................................................................................................... ix

List of Figures .......................................................................................................................... xiii

List of Tables ........................................................................................................................... xvi

List of Acronyms ..................................................................................................................... xvii

1 Thesis Overview ..................................................................................................................... 1

1.1 Introduction ........................................................................................................................ 1

1.2 Thesis layout ....................................................................................................................... 1

1.3 Learning outcomes during thesis ..................................................................................... 3

1.4 Summary ............................................................................................................................ 4

2 Literature Review ................................................................................................................... 5

2.1 Introduction ........................................................................................................................ 5

2.2 Dynamic Testing of ADC .................................................................................................. 5

2.2.1 Manual back to back Dynamic ADC testing ................................................................. 6

2.2.2 DSP-based ADC testing ............................................................................................... 7

2.3 FFT basics .......................................................................................................................... 8

2.4 Standardization of ADC test methods .............................................................................. 9

2.4.1 Frequency domain testing of ADCs ............................................................................. 9

2.4.2 ADC modelling and testing [9] ..................................................................................... 10

2.4.3 Harmonization of standard procedures for ADC time domain dynamic testing ... 10

2.5 Conclusion ......................................................................................................................... 11

3 Basic Design of FIR Filters .................................................................................................. 13

3.1 Introduction ....................................................................................................................... 13

3.2 Basic filter theory .............................................................................................................. 13

3.3 Finite Impulse Response filter (FIR) overview and basic principles ......................... 14

3.4 Causal and non-causal systems ....................................................................................... 15

3.5 Advantages of FIR filters ............................................................................................... 15
5.2.2 Operation overview ................................................................. 42
5.3 Clock Reset Control (CRCTL) ................................................. 43
  5.3.1 Clock Control ................................................................. 43
  5.3.2 Reset Control .............................................................. 44
  5.3.3 Timing analysis for the CRCTL ....................................... 44
5.4 SPI_Master & SPI_Slave (SPIS) used in this system ...................... 44
5.5 Function of Master and Slave .................................................. 46
5.6 Direct Digital Synthesizer (DDS) generator .................................. 47
5.7 System Control Unit (SCU) .................................................... 48
  5.7.1 Finite State Machine (FSM) controller ............................... 48
5.8 Digital to Analogue Converter (DAC) ........................................ 50
5.9 Xilinx Analogue to Digital Converter (XADC) ............................. 51
  5.9.1 Unipolar mode .......................................................... 51
  5.9.2 Bipolar mode ........................................................... 52
5.10 SPI DAC interface .............................................................. 53
  5.10.1 Timing analysis of SPI DAC interface ............................... 54
5.11 Block memory .................................................................. 54
5.12 Arbitration Block (ARB) and PC Controlled Register (PCR) ............. 55
5.13 Key specifications and overall timing diagram ............................. 57
5.14 Verilog coding style ............................................................ 59
5.15 Conclusion ....................................................................... 59

6 Simulation and Testing of FPGA Test Platform for Data Converters .......... 61
  6.1 Introduction ........................................................................ 61
  6.2 Top level test bench ............................................................ 62
    6.2.1 Clock & Reset ............................................................ 63
    6.2.2 SPI_Master ................................................................. 63
    6.2.3 Sys_top ................................................................ 63
    6.2.4 spi_dac_model .......................................................... 63
    6.2.5 XADC output and sine wave input ................................. 63
  6.3 Verilog test bench RTL aspects .............................................. 64
  6.4 Test Flow Procedure .......................................................... 66
  6.5 Xilinx ISIM simulation ......................................................... 67
  6.6 MATLAB for FFT spectral analysis .......................................... 70
    6.6.1 Results and FFT plots using MATLAB ......................... 70
  6.7 Analysis of dynamic parameters ............................................. 71
  6.8 Potential application of FPGA embedded test platform .................. 73
    6.8.1 Verilog implementation of scenario 1 .............................. 75
    6.8.2 Verilog implementation of scenario 2 .............................. 78
Appendix 3: Verilog RTL Code List .......................................................... 113

Appendix 4: MATLAB Code List ............................................................ 115

Appendix 5: Sample XADC Input waveform format ................................. 117
List of Figures

Figure 1-1: Overview of proposed architecture and platform design .............................................. 2
Figure 2-1: Back to back dynamic ADC testing [1] .......................................................................... 6
Figure 2-2: Simplified block diagram of DSP based testing [2] ......................................................... 7
Figure 2-3: Fourier analysis between time/frequency domains ......................................................... 8
Figure 2-4: ADC test setup ............................................................................................................... 9
Figure 2-5: An ADC with state space compensation [9] ................................................................. 10
Figure 3-1: Filter response highlighting the transition band .............................................................. 14
Figure 3-2: FIR filter structure ...................................................................................................... 14
Figure 3-3: Ideal brick wall low pass filter ..................................................................................... 15
Figure 3-4: Impulse response for the ideal LPF ............................................................................. 16
Figure 3-5: Ideal filter with cut off $\Omega_c$ ..................................................................................... 16
Figure 3-6: Magnitude and phase response of 7-tap filter ............................................................. 19
Figure 3-7: DC gain of the filter ..................................................................................................... 20
Figure 3-8: Unity gain FIR low pass filter of 7-taps ..................................................................... 20
Figure 3-9: Pole-Zero plot in Z plane ............................................................................................ 21
Figure 3-10: Pole-Zero plot for an FIR filter showing system is unconditionally stable ............... 22
Figure 3-11: Implementation of multiplication function using shift and add .................................. 24
Figure 4-1: Time domain view of 32Hz sine wave ....................................................................... 29
Figure 4-2: Frequency domain showing single amplitude peak at 32 Hz .................................... 29
Figure 4-3: Block diagram of a DFT ............................................................................................ 30
Figure 4-4: FFT output for ideal 12-bit ADC, Input = 2.111 MHz, $f_s = 82$MSPS, $M = 8192$ [14] ... 31
Figure 4-5: Signals and their spectra without leakage and with leakage [7] ................................. 32
Figure 4-6: Hanning window in time domain ............................................................................... 33
Figure 4-7: Spurious Free Dynamic Range (SFDR) .................................................................... 35
Figure 4-8: Top level procedure using Verilog / MATLAB to display spectral performance .......... 36
Figure 5-1: Hardware setup ......................................................................................................... 40
Figure 5-2: Top level block diagram ............................................................................................ 41
Figure 5-3: Clock Reset Control .................................................................................................. 43
Figure 5-4: CRCTL timing ........................................................................................................... 44
Figure 5-5: SPI timing diagram used in this design .................................................................... 45
Figure 5-6: SPI Master-Slave interface ....................................................................................... 46
Figure 5-7: DDS1 and DDS2 ...................................................................................................... 48
Figure 5-8: SCU state diagram .................................................................................................... 49
Table of Contents

Figure 5-9: DAC setup [26]...........................................................................................................50
Figure 5-10: XADC unipolar mode ..........................................................................................51
Figure 5-11: XADC bipolar mode ...........................................................................................52
Figure 5-12: SPI DAC interface ...............................................................................................53
Figure 5-13: Timing diagram for the SPI DAC interface ..........................................................54
Figure 5-14: Block memory [28].............................................................................................55
Figure 5-15: PCR and ARB integration ......................................................................................56
Figure 5-16: Timing diagram of overall system .......................................................................58
Figure 6-1: Basic test cycle .......................................................................................................62
Figure 6-2: Top level test bench ...............................................................................................62
Figure 6-3: Sample code showing the instantiation of SPI Master .............................................64
Figure 6-4: Sample code showing the use of `ifdef ................................................................64
Figure 6-5: Sample code showing the clock generation in Verilog test bench .......................65
Figure 6-6: Sample code showing the reset generation in Verilog test bench .......................65
Figure 6-7: Test flow procedure ...............................................................................................66
Figure 6-8: Simulation of the design where SCU writes data into the memory .......................68
Figure 6-9: Simulation results where the data is read from memory through MISO ...............69
Figure 6-10: FFT plots for different output tones of different frequencies .................................71
Figure 6-11: SAR ADC producing output in two different times ............................................74
Figure 6-12: Input to the filter where the first LSB is initially set to zero ....................................75
Figure 6-13: FIR structure of (a) standard FIR filter and (B) structure used for scenario 1 ....75
Figure 6-14: Output of both reference and research 3 tap filters (scenario 1) .........................76
Figure 6-15: FFT plot of input and output of 3-tap FIR filters (scenario 1) .................................77
Figure 6-16: Input to the filter where the first coefficient is fed with previous LSB .................79
Figure 6-17: FIR structure of (a) standard FIR filter and (b) structure used for scenario 2 ....79
Figure 6-18: Output of both Golden and Research 3-tap filters (scenario 2) ............................80
Figure 6-19: FFT plot of input and output of 3-tap FIR filters (scenario 2) ...............................81
Figure 6-20: LFSR FFT output .................................................................................................83
Figure 6-21: Histogram of a 32-bit LFSR .................................................................................84
Figure 6-22: FIR structure of (a)Reference FIR filter and (b) Research Filter used for scenario 3...84
Figure 6-23: Input to the filter where the first coefficient is fed LFSR output ............................84
Figure 7-1: Adaptive filter approach for the reduction of error at the output of ADC ...............91
Figure 7-2: Equalization-based adaptive background digital calibration architecture of SAR-ADC [31].92
Figure A1-1: Sample revision control header ...........................................................................97
Figure A1-2: Sample Verilog code showing standard indentation and commenting style used throughout this thesis .........................................................................................................................99
Figure A1-3: Standard Verilog signal naming convention used in the thesis ............................99
Figure A1-4: Sample code of blocking assignments ..................................................................100
Figure A1-5: Sample code of non-blocking assignments....................................................... 100
Figure A1-6: Sample code for synchronous reset .................................................................... 101
Figure A1-7: Sample code for asynchronous reset................................................................. 101
Figure A1-8: Sample code showing the usage of params.v..................................................... 102
Figure A1-9: Sample code illustrating the latch formation...................................................... 102
Figure A1-10: Latch formation due to incomplete assignment ................................................ 103
Figure A1-11: Sample code to resolve the problem of latch formation .................................. 103
Figure A1-12: Finite State Machine (FSM) ............................................................................. 104
Figure A1-13: Example code for the structure of state machine used in this thesis ............ 105
Figure A1-14: Sample code showing instantiation of Design Under Test (DUT) .............. 106
Figure A1-15: MATLAB code for the design and analysis of FIR filter ................................ 107
Figure A1-16: Verilog code for exporting data into a text file ............................................. 107
Figure A1-17: MATLAB code for importing data from a text file ....................................... 108
Figure A1-18: Spectral analysis of digital signals using MATLAB ....................................... 108
Figure A1-19: MATLAB code for function that calculates dynamic parameters of ADC .... 109
Figure A2-1: Q 15 format for representing fixed point ......................................................... 111
Figure A2-2: General format or representation of binary number in fixed point ............... 112
List of Tables

Table 3-1: Filter coefficients for 7-tap filter ................................................................. 18
Table 3-2: Filter coefficients of 7-tap filter ................................................................ 18
Table 3-3: Values of zeros in Equation 19..................................................................... 22
Table 3-4: Implementation 7 multiplied by 0.75 using shift and add .................................. 25
Table 3-5: Restoration of precision by increasing the bit precision of the register ............. 25
Table 4-1: Critical ADC parameters [10] ......................................................................... 28
Table 4-2: Comparison of dynamic test parameters of a 7-tap filter for different input frequencies .......................... 37
Table 5-1: Bus arbitration based on PCR ........................................................................ 55
Table 5-2: Values of PCR register ................................................................................ 55
Table 5-3: Key specifications ......................................................................................... 57
Table 6-1: Coherent frequency tones generated by the DDS based on phase increment values ....... 71
Table 6-2: Key specifications of the signals DDS, XADC, Block memory ....................... 72
Table 6-3: Comparison of SNR and SINAD for DDS, XADC, Block memory ................. 72
Table 6-4: Expected loss in the amplitude of the signal in frequency domain .................... 73
Table 6-5: Comparison of 3-tap FIR filter performance – scenario 1 ................................. 77
Table 6-6: Comparison of 7-tap FIR filter performance – scenario 1 ................................. 78
Table 6-7: Comparison of 11-tap FIR filter performance – scenario 1 ............................... 78
Table 6-8: Comparison of 21-tap FIR filter performance – scenario 1 ............................... 78
Table 6-9: Comparison of 3-tap FIR filter performance – scenario 2 ................................. 82
Table 6-10: Comparison of 7-tap FIR filter performance – scenario 2 ............................... 82
Table 6-11: Comparison of 11-tap FIR filter performance - scenario 2 ............................. 82
Table 6-12: Comparison of 21-tap FIR filter performance – scenario 2 ............................. 82
Table 6-13: Comparison of 3-tap FIR filter performance – scenario 3 (LFSR) .................... 85
Table 6-14: Comparison of 7-tap FIR filter performance – scenario 3 (LFSR) .................... 85
Table 6-15: Comparison of 11-tap FIR filter performance – scenario 3 (LFSR) ................... 85
Table 6-16: Comparison of 21-tap FIR filter performance – scenario 3 (LFSR) ................... 85
Table 6-17: Comparison of 11-tap FIR filter performance – (2x sampling and zero LSB) .... 86
Table 6-18: Comparison of 11-tap FIR filter performance – (4x sampling and zero LSB) ...... 86
Table 6-19: Comparison of 11-tap FIR filter performance – (2x sampling and previous LSB) ... 86
Table 6-20: Comparison of 11-tap FIR filter performance – (4x sampling and previous LSB) .... 86
Table A1-1: Inbuilt MATLAB functions for the computation of dynamic test parameters of ADC ...... 108
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>Analogue-to-Digital Converter</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application Specific Integration Circuit</td>
</tr>
<tr>
<td>DAC</td>
<td>Digital-to-Analogue Converter</td>
</tr>
<tr>
<td>DDS</td>
<td>Direct Digital Synthesizer</td>
</tr>
<tr>
<td>DFT</td>
<td>Discrete Fourier Transform</td>
</tr>
<tr>
<td>DNL</td>
<td>Differential Nonlinearity</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processing</td>
</tr>
<tr>
<td>ENOB</td>
<td>Effective Number of Bits</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>FIR</td>
<td>Finite Impulse Response</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>FSM</td>
<td>Finite State Machine</td>
</tr>
<tr>
<td>HDL</td>
<td>Hardware Description Language</td>
</tr>
<tr>
<td>IC</td>
<td>Integrated Circuit</td>
</tr>
<tr>
<td>IIR</td>
<td>Infinite Impulse Response</td>
</tr>
<tr>
<td>INL</td>
<td>Integral Nonlinearity</td>
</tr>
<tr>
<td>IP</td>
<td>Intellectual Property</td>
</tr>
<tr>
<td>LMS</td>
<td>Least Mean Square</td>
</tr>
<tr>
<td>LSB</td>
<td>Least Significant Bit</td>
</tr>
<tr>
<td>MISO</td>
<td>Master In Slave Out</td>
</tr>
<tr>
<td>MOSI</td>
<td>Master Out Slave In</td>
</tr>
<tr>
<td>MSB</td>
<td>Most Significant Bit</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>rms</td>
<td>Root Mean Square</td>
</tr>
<tr>
<td>RTL</td>
<td>Register Transfer Language</td>
</tr>
<tr>
<td>SAR</td>
<td>Successive Approximation Register</td>
</tr>
<tr>
<td>SFDR</td>
<td>Spurious Free Dynamic Range</td>
</tr>
<tr>
<td>SINAD</td>
<td>Signal-to-Noise and Distortion</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal-to-Noise Ratio</td>
</tr>
<tr>
<td>THD</td>
<td>Total Harmonic Distortion</td>
</tr>
<tr>
<td>VLSI</td>
<td>Very Large Scale Integration</td>
</tr>
<tr>
<td>XADC</td>
<td>Xilinx Analogue-to-Digital Converter</td>
</tr>
</tbody>
</table>
1 Thesis Overview

1.1 Introduction

The aim of this thesis is to develop an embedded digital test platform specifically targeted at data converters and digital filters. The primary application of this platform involves dynamic testing the outputs of Analogue-to-Digital Converters (ADC) in real time and their further analysis using Finite Impulse Response (FIR) filters. A secondary objective allows for the future research on adaptive filters that are used for post correction of ADC outputs. A complementary objective was to develop an understanding of digital Finite Impulse Response (FIR) filter design fundamentals and then using this study as a mechanism for furthering know-how into good digital design practice. This includes understanding of signal processing techniques, MATLAB code application and robust Verilog coding techniques which are useful for design into system on chips applications.

1.2 Thesis layout

This thesis is sectioned into seven chapters including this chapter. Chapter 2 addresses historical developments and state of art in the field. This literature review helps to put the work presented into context and explains the motivation for the research undertaken. This chapter focuses on an introduction to real time testing methods of ADCs including Fast Fourier Transform (FFT) testing.

Chapter 3 involves the learning and basic understanding of digital signal processing through basic design of FIR filters using MATLAB and Verilog code.

Chapter 4 emphasizes the importance of Fast Fourier Transforms (FFT) and its computation using MATLAB code. The FFT computation is the important tool used in this thesis to precisely validate the filter and test platform output performance. This chapter also discusses important dynamic ADC parameters that are used to judge the quality of the test signal.
Chapter 5 presents the design of the data converter test platform and SPI interface using Verilog RTL coding style and discusses the selection of hardware for this prototype platform. Figure 1-1 shows the overview of proposed architecture and has four main components, which are the Zed board FPGA, AMS101 card, SPI interface and PC acting as a SPI Master. The main focus of attention is the design & test platform implementation on the FPGA device that is then connected to a PC using an SPI interface through virtex4 SPI interface board. The system is validated using FFTs to measure spectral performance of the design under test.

![Diagram](image)

Figure 1-1: Overview of proposed architecture and platform design

Chapter 6 concentrates on the testing and simulation of this platform, where a dedicated Verilog test bench is developed that generates all the test signals required to validate the functionality of the test platform. This wrapper includes a SPI_Master that mimics the real time PC and also instantiates the designed test platform and outputs the results in a text file which is used for analysis of FFT using MATLAB. A potential application for the test platform is introduced where the outputs of the ADC are applied to FIR filter examples to enable further analysis. This application is developed in consideration of Successive Approximation Analogue-to-Digital Converters (SAR ADC).
Chapter 7 discusses the work and the outcomes of the thesis. Furthermore, this section highlights future research activities, which include adaptive filter design for post correction of ADC outputs.

The remaining part of this thesis includes appendices for the various MATLAB and Verilog code used in the design implementation. The appendices are divided into five sections. Appendix 1 discusses practical Verilog Register Transfer Language (RTL) coding style and recommended design practices, which illustrate the importance of a structured approach to applying RTL code in VLSI design. The formats introduced in this Appendix are used throughout this thesis. Appendix 2 provides an introduction to Q point representation which is used to translate the real value filter coefficients to the two’s complement formation used in digital design implementations. Appendix 3 includes all the Verilog RTL programs implemented throughout the project and is attached as a CD. Appendix 4 lists all the MATLAB codes for the thesis and Appendix 5 shows the file format used for applying a sinewave to the Xilinx ADC (XADC) input.

The learning outcomes are explained in the following section.

1.3 Learning outcomes during thesis

The key focus of interest for this thesis is to develop an embedded test platform for data converters based on a FPGA design implementation. During this process, the following outcomes were achieved;

- Knowledge of implementing the digital design principles in Verilog HDL with good coding style and naming conventions where the source of each signal can be identified by its signal as shown by the code contained in Appendix 2. Various design tips using Verilog HDL were learned that include Clock/Reset control.

- Good knowledge in digital design principles, which includes the fundamentals of digital design (combinatorial and sequential logic), followed by specifications of sequential logic (rise time and fall time, delay). Another important aspect is learning aspects of digital design such as synchronisers (explained in Chapter 4), Serial Parallel Interface (SPI) bus, etc.

- Fundamental knowledge gained in digital signal processing aspects of FIR filter design followed by the practical analysis using MATLAB.
MATLAB is a major tool used to analyse DSP techniques and for performing FFTs. Robust code was developed in MATLAB for the input/output analysis of the designs under test using FFT.

Experience in System on Chip (SoC) design using FPGA and Xilinx Intellectual Property cores (IP).

Good knowledge in design of test benches for the verification of SoC for correct functionality.

Knowledge in design of digital filters for the analysis of ADC outputs.

The essential work from this thesis was to learn the functionality of each component in the design, thoroughly accessing data sheets where necessary to ensure the design met the specified requirements.

Good writing and presentation skills were also developed during the study of this project.

This project focuses on the digital front end design flow for FPGA design which includes specification, architecture development, digital design using Verilog coding, FPGA implementation and design verification. The traditional ASIC digital back-end flow consists of layout floor planning, pin placement, place and route, and timing analysis which couldn’t be implemented as this project did not use a digital IC design flow to map the design onto a chip integrated circuit. Instead, the Xilinx ISE tool was used to apply the synthesized Verilog code to an Artix 7 series FPGA device integrated onto the Xilinx Zed board. This platform allows for future work in development of an adaptive filter design highlighted in Chapter 7, which could be used for the post correction of ADC outputs.

1.4 Summary

This chapter provides an overview of the thesis objectives, including the layout structure, briefly describing the contents of each chapter. The outputs achieved during the course of this work were also listed including a short section on what was not met during the project timeframe. This material becomes potential for future work associated with this project. The next chapter provides a literature review that outlines the historical developments and state of art in the field of real time ADC testing methods including FFT testing.
2 Literature Review

2.1 Introduction

The history of data converters finds their place in the 1940s continuing until the present day. Data converters are normally classified into two categories depending on their sampling frequency: (1) low speed or precision data converters and (2) high speed or low precision data converters. The interest in ADCs and DACs increased in the 1960s, as solid-state data converters as well as main frame computers became available. Some examples of early data converter uses include data analysis, instrumentation, Pulse Code Modulation (PCM), and radar applications. ADCs and Digital to Analogue Converters (DACs) find their applications in digital video during 1970s and therefore this made application specific testing a primary requirement. Standardization of ac specifications such as SNR, SINAD, ENOB, THD, etc., took place in 1980s and widespread growth in ac testing took place in this period [1]. These specifications are very important in emerging applications such as communications where dynamic range plays an important role. By the 1990s, frequency-domain testing of ADCs and DACs came to the fore based on using FFTs for signals analysis [1].

2.2 Dynamic Testing of ADC

A focus for this thesis is to perform dynamic testing of an ADC to measure ac performance characteristics such as SNR, SINAD, SFDR, etc. The classical method of ac testing in early 1970’s was performed using a back to back test method where a high-performance DAC was used to reconstruct the ADC output [1]. This method allows the use of traditional analogue equipment. In mid 1970’s, Digital Signal Processing (DSP) based testing of ADC’s started to evolve [1]. In today’s world, practically all ADC testing is carried out using some type of digital analysis of the output data. The latest practice
from data converter vendors is to have manufacturer supplied evaluation boards, accompanied with PC-based software packages [1].

2.2.1 Manual back to back Dynamic ADC testing

A typical back-to-back ADC test setup is shown in Figure 2-1.

![Back to back dynamic ADC testing](image)

This test method is most suitable for 8 and 10-bit high speed ADCs and the evaluation setup contains a reconstruction DAC which is sufficient to perform basic tests. Nowadays, the reconstruction DAC is typically included on the manufacturer’s ADC evaluation board. The real ac evaluation of the ADC should be done using DSP techniques. Back to back testing still find its place in applications like audio codecs where back to back performance of ADC/DAC combination metrics is used to determines the overall system performance. This testing mainly measures parameters like SNR, SINAD, THD, SFDR [1]. The test setup includes an analogue signal generator that can generates a single, multitone sine wave, pulse or specific audio test signal, etc while the analogue signal measurement equipment is typically an oscilloscope, spectrum analyzer, network analyzer or audio measurement equipment, etc. The reconstruction DAC produces a series of rectangular pulses whose widths are equal to the reciprocal of the sampling frequency. The frequency response of a signal follows a Sinc function and hence this needs a compensation low pass filter.
2.2.2 DSP-based ADC testing

DSP based ADC testing is always more accurate than the analog testing. This test equipment has the following components;

- A very fast and accurate Digital Signal Processor (DSP)
- Processing based on double precision floating point arithmetic

A sample system is shown in Figure 2-2.

![Simplified block diagram of DSP based testing](image)

Figure 2-2: Simplified block diagram of DSP based testing [2]

DSP based testing [3] in this class are limited to discrete (i.e. sampled) waveforms of finite length with features such as

1. **Coherent DSP based testing**

   DSP based testing helps to reduce test time since we can create signals with multiple frequencies at the same time. Once the output response of the DUT is captured using a digitizer or captured into memory, DSP allows the separation of test tones to give individual gain and phase measurements. Also, by removing the input test tones, we can measure noise and distortion without running many separate tests.

2. **Separation of Signal Components**

   By using coherent test tones, we are guaranteed that the harmonic distortion components will fall neatly into separate Fourier spectral bins rather than being smeared across many
bins. DSP based testing also has the major advantage in the elimination of errors and poor repeatability.

3. Advanced Signal Manipulations

DSP allows us to manipulate digitized output waveforms to achieve a variety of results. We can apply mathematical filters to remove noise thereby achieving better accuracy. The references [4, 5] give detailed explanation of DSP testing and the Automatic Waveform Generator (AWG).

2.3 FFT basics

This section explains about the basics of FFT and their application for testing of ADCs. Fourier analysis [1, 6, 7] forms the very basics of digital signal processing. Fourier analysis can be used to find the equivalent representation of a time domain signal in the frequency domain. Conversely, once the frequency response of a signal is known, the inverse Fourier transform can be used to determine the time domain signal. The Discrete Fourier Transform (DFT) is the family of Fourier transforms that used with digital periodic signals. A detailed explanation of DFT is given in chapter 4.

![Diagram of Discrete Fourier Transform (DFT)](image)

Figure 2-3: Fourier analysis between time/frequency domains

Applications of Fourier Transform include:

- Filter design
  - Calculating impulse response from frequency response
  - Calculating frequency response from impulse response
- Digital spectral analysis
  - ADC testing
  - Pattern recognition
- Speech processing
  - The FFT is an efficient calculation for the DFT. An in-depth analysis of FFTs is carried out in chapter 4.

2.4 Standardization of ADC test methods

2.4.1 Frequency domain testing of ADCs

Traditional off the shelf ADC device implementations consist of high-performance analog designs combined with some minimal digital functions for control or interface. In contrast, digital/mixed-signal integrated circuit devices combine complex state-of-art digital designs with added analog input or output functionality. The standard ADC test set-up is documented in [8] where this method of dynamic testing of ADC requires digital and analog stimuli followed by digital capture as shown in Figure 2-4.

The analog input signal stimulates the device in a known manner for analysis. The digital stimulus selects the device and provides control signals to select device ranges or to start conversion. Digital capture acquires the digital representation of analog signals. The controller or host computer connects all this hardware together.

![Figure 2-4: ADC test setup](image-url)
2.4.2 ADC modelling and testing [9]

Dynamic tests are based on the use of time varying input signals. The choice of the input stimulus is mostly dictated by the availability of well characterised sources: sine-waves are preferred, but voltage steps and noise are also of interest. Dynamic tests provide a measure of the degradation of the signal collected at the output. In the limit case of slow variations, the same performances are expected, which might be predicted on the basis of the static transfer characteristic, and the static ADC model is effective. ADCs in real world show plenty of dynamic effects and it would be highly desirable to be able to describe them by a high level behavioural model there by helping the designer to predict the desirable performances. A block diagram representing the state space compensation is shown below in Figure 2-5. In this case an estimate of input signal is obtained by subtracting the error from the output.

![Figure 2-5: An ADC with state space compensation][9]

2.4.3 Harmonization of standard procedures for ADC time domain dynamic testing

This method deals with the ADC dynamic parameters and test methods in the time domain by means of sine wave fitting.

**Sine wave fitting**

The IEEE standard [10] defines ADC testing methods which makes use of least squares sine wave fitting algorithms. After sine fitting, an error sequence is obtained and if the
quantization noise dominates this sequence the ADC error analysis may be misleading. A new method is proposed in [11] which addresses this problem. One of the reason of this problem could be, the amplitude of the test signal is small enough or the amplitude is out of range of ADC.

Another method is proposed by Giaquinto in [12] for the fast and accurate ADC testing. In this method a new algorithm is proposed where there is the possibility of using test signals with amplitude greater than the full scale range of the device under test.

An analytical comparison of these methods, according to the internationally available standards and the research in the field, is described in [13]. The results of an experimental investigation carried out on actual ADCs by applying the sine wave fitting methods are also reported and compared to other test methods as a step towards the harmonization of the existing standards. The increasing availability and the growing number of applications of high speed ADCs led most manufacturers to include specifications for ADC dynamic performance in their datasheets [14].

2.5 Conclusion

Testing of ADCs is the primary concern of this thesis. An ADC test environment for evaluating the dynamic performance of an ADC is proposed and developed using an FPGA based evaluation board as part of this project. Dynamic ADC testing is performed on the input and output signals of ADC by analysing the ac parameters using FFT analysis [1, 14] [6, 7, 15]. Dynamic testing can also be used to evaluate designs such as digital filters that are often used as post-processing elements attached to the ADC output. The following chapters in this thesis outline the design approaches used to apply these methods.
3 Basic Design of FIR Filters

3.1 Introduction

The aim of this chapter is to present the fundamentals of digital signal processing by explaining the basic theory of digital filters with the design of a 7-tap FIR filter both in MATLAB and Verilog RTL code. The outcomes include a better understanding and knowledge of digital design processing techniques attached to FIR filter design leading to an effective implementation in Verilog RTL, which aids in the development of an embedded test platform for data converters using FPGA.

3.2 Basic filter theory

Electronic filters [7] perform signal processing functions specifically to select or pass-desired frequencies and suppress unwanted frequency components. Filters mostly find their application in data convertors, RF, video processing and in telecommunications. In general, both passive and active components can be used to design analogue filter circuits. The required frequency cannot often be selected due to inaccuracy in passive components such as resistor, capacitor and inductor and op-amp drift. Translating the signal into the discrete sampled digital domain solves this problem, where filter coefficients and accumulators replace the passive components and multipliers perform the computation. Despite the advantages of analogue filters such as speed and large dynamic range, digital filters help us to generate smaller transition band as depicted in Figure 3-1. Digital filters are normally classified into two types - Finite Impulse Response (FIR) and Infinite Impulse Response (IIR). FIR filters are systems where the output depends upon the present and past inputs whereas in IIR filters, the output is a combination of inputs as well.
as the past outputs. This chapter introduces the basic structure and advantages of FIR filters while also proposing a standard methodology for the design approach.

![Filter response highlighting the transition band](image)

**Figure 3-1: Filter response highlighting the transition band**

### 3.3 Finite Impulse Response filter (FIR) overview and basic principles

The diagram in Figure 3-2 shows the basic structure of a FIR filter where $x[n]$ represents the input sequence and $y[n]$ represents the output sequence. The output $y[n]$ is the result of $x[n]$ after passing through all the delay elements and multipliers and adders.

Equation (3-1) is the mathematical representation of FIR filters.

![FIR filter structure](image)

**Figure 3-2: FIR filter structure**
Basic FIR Filter Design

\[ H(z) = \sum_{n=0}^{N} h(n)Z^{-n}. \]  
(3-1)

3.4 Causal and non-causal systems

A causal system is the one in which the output \( y[n] \) at time \( n \) depends only on present input \( x[n] \) and past input samples \( x[n-1], x[n-2] \ldots \) at time \( n \). If the output \( y[n] \) depends on future inputs such as \( x[n+1], x[n+2] \ldots \) then the system is called non-causal.

3.5 Advantages of FIR filters

- All FIR filters are linear phase filters [6].
- FIR filters are unconditionally stable because the poles cannot be placed outside the unit circle [6].

3.6 Reasons for non-realization of an ideal filter

In practice, an ideal filter cannot be realized due to the following reasons.

- The impulse response of the ideal filter is non-causal.
- Since the impulse response for an ideal filter such as brick wall extends from \((-\infty, +\infty)\), the sum of all components are not realizable.

3.6.1 Mathematical analysis

This is best explained by design of an ideal LPF with a cut off frequency of 25 kHz at a sampling rate of 10MHz. The ideal response is shown in Figure 3-3.

![Figure 3-3: Ideal brick wall low pass filter](image-url)
The time domain response of this filter or the inverse filtering gives an impulse response which extends from \((-\infty, +\infty)\) as shown in the Figure 3-4.

The impulse response in Figure 3-4 is non-causal where the sum of all components is not realizable.

Since an ideal filter cannot be realized in real time an alternative method is introduced in the next section for the design of the digital filters.

### 3.7 Design of FIR filter using Fourier Transform method

The Fourier Transform method detailed by Li Tan in [7] uses the basic inverse Fourier Transform for the design of a FIR filter of desired specification.

#### 3.7.1 Mathematical analysis

Let us assume a brick wall filter of cut off frequency \(\Omega_c\) as shown in Figure 3-5.

![Figure 3-5: Ideal filter with cut off \(\Omega_c\)]
The above filter can be characterized as follows in Equation (3-2)

\[ H(e^{j\Omega}) = \begin{cases} 
1, & 0 \leq |\Omega| \leq \Omega_c \\
0, & \Omega_c \leq |\Omega| \leq \pi 
\end{cases} \]  

(3-2)

The Fourier Transform is given in Equation (3-3)

\[ h(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{j\Omega}) \ast e^{j\Omega \ast n} d\Omega, \quad \text{where } -\infty < n < \infty. \]  

(3-3)

This is a non-causal FIR filter and is solved as in Equation (3-4) and (3-5)

\[ h(0) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{j\Omega}) \ast e^{j\Omega \ast 0} d\Omega, \quad \text{with } n = 0. \]  

(3-4)

\[ h(0) = \frac{1}{2\pi} \int_{-\Omega_c}^{\Omega_c} 1 d\Omega = \frac{\Omega_c}{\pi}. \]  

(3-5)

For \( n \neq 0 \), \( h(n) \) is defined as Equation (3-6) and (3-7)

\[ h(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{j\Omega}) \ast e^{j\Omega \ast n} d\Omega. \]  

(3-6)

\[ h(n) = \frac{1}{2\pi} \int_{-\Omega_c}^{\Omega_c} e^{j\Omega \ast n} d\Omega = \frac{1}{\pi n} \left( e^{j\Omega_c \ast n} - (e^{j\Omega_c \ast n}) \right) = \frac{\sin(n\Omega_c)}{\pi}. \]  

(3-7)

This concludes the mathematical approach for the design of digital filter and the next section shows the MATLAB implementation of the same filter.

### 3.8 Practical implementation with MATLAB

Let us design a 7-tap FIR filter with a cut off frequency of 25 KHz at a sampling rate of 10MHz using the Fourier Transform method.
Step 1: In order to design a filter in MATLAB, we require the coefficients of the numerator and denominator of the transfer function.

Step 2: First, the normalized frequency highlighted in Figure 3-5 is calculated as follows as in Equation (3-8)

$$\Omega_c = 2\pi f_c T_s = 2\pi \times \frac{25e3}{10e6} = 0.01571 \text{ radians.}$$  \hspace{1cm} (3-8)

Step 3: It is a 7-tap filter, using Equation (3-7) and (3-8) we have h(0), h(1), h(2) and h(3) using symmetry

<table>
<thead>
<tr>
<th>Filter taps</th>
<th>Using symmetry</th>
</tr>
</thead>
<tbody>
<tr>
<td>h(0) = \frac{0.01571}{\pi} = 0.00500</td>
<td>h(-1) = h(1) = 0.0050</td>
</tr>
<tr>
<td>h(1) = \frac{\sin(1 \times 0.01571)}{1 \times \pi} = 0.0050</td>
<td>h(-2) = h(2) = 0.0050</td>
</tr>
<tr>
<td>h(2) = \frac{\sin(2 \times 0.01571)}{2 \times \pi} = 0.0050</td>
<td>h(-3) = h(3) = 0.0050</td>
</tr>
<tr>
<td>h(3) = \frac{\sin(3 \times 0.01571)}{3 \times \pi} = 0.0050</td>
<td></td>
</tr>
</tbody>
</table>

Table 3-1: Filter coefficients for 7-tap filter

Finally, the coefficients are presented in Table 3-2.

<table>
<thead>
<tr>
<th>Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>b0 = h (-3) = 0.0050</td>
</tr>
<tr>
<td>b1 = h (-2) = 0.0050</td>
</tr>
<tr>
<td>b2 = h (-1) = 0.0050</td>
</tr>
<tr>
<td>b3 = h (0) = 0.0050</td>
</tr>
<tr>
<td>b4 = h (1) = 0.0050</td>
</tr>
<tr>
<td>b5 = h (2) = 0.0050</td>
</tr>
<tr>
<td>b6 = h (3) = 0.0050</td>
</tr>
</tbody>
</table>

Table 3-2: Filter coefficients of 7-tap filter

Step 4: Finally, the transfer function is defined as in Equation (3-9)
\[ H(z) = 0.00500 + 0.00500z^{-1} + 0.00500z^{-2} + 0.00500z^{-3} + 0.00500z^{-4} + 0.00500z^{-5} + 0.00500z^{-6}. \] (3-9)

**Step 5:** This is implemented in MATLAB code which is listed in Appendix 3.

**Step 6:** The magnitude and the phase response is shown in the plot of Figure 3-6.

![Figure 3-6: Magnitude and phase response of 7-tap filter](image)

### 3.9 Design of unity gain filter of 7-taps

The 7-tap filter designed in the previous section has a loss of 21.92dB in the pass band as shown in Figure 3-6. In order to design a unity, gain filter, the transfer function of the filter is scaled such that the sum of the coefficients is unity. This is done by dividing the filter coefficients with the DC gain of the filter.

#### 3.9.1 DC gain of the filter

When a linear time invariant filter (LTI) (A filter that obeys both linearity and time invariance properties) is fed with a unit impulse, the output of the filter is known as the DC gain of the filter represented by Figure 3-7.
The DC gain is directly obtained by placing $z = 1$ in Equation (3-9), which gives $H(1) = 0.0350$. On dividing Equation (3-9) with the DC gain, we get the new transfer function defined as follows in Equation (3-10)

$$H(z) = 0.1428 + 0.1429z^{-1} + 0.1429z^{-2} + 0.1429z^{-3} + 0.1429z^{-4} + 0.1429z^{-5} + 0.1428z^{-6}.$$ (3-10)

The magnitude and phase response of the transfer function in Equation (3-10) is shown in Figure 3-8.
3.10 Pole-Zero plots

The Z-Plane pole-zero plot as explained in [7] is a very useful tool for the study of digital systems. This is a graphical technique, which helps us to investigate the characteristics of a digital system including stability and frequency response. The transfer function of a system in pole-zero form is shown in Equation (3-11) for which poles and zeros can be plotted on the Z plane. The Z-plane is depicted in Figure 3-9

\[
H(z) = \frac{b_0(z-z_1)(z-z_2)\ldots(z-z_M)}{(z-p_1)(z-p_2)\ldots(z-p_N)}. \tag{3-11}
\]

Figure 3-9: Pole-Zero plot in Z plane

**Pole-Zero plot and analysis of stability**

A filter is said to be stable if the poles of the filter lie inside the unit circle. Zeros do not affect stability [6, 7]. From Equation (3-11), poles can be obtained by finding the roots of the denominator. The poles and zeros of the filter from Equation (3-10) in pole-zero format is shown in Equation (3-12)

\[
H(z) = (z - z1)(z - z2)(z - z3)(z - Z4)(z - z5)(z - z6). \tag{3-12}
\]
### 3.11 MATLAB analysis on the stability of FIR filters

Figure 3-10 depicts the pole-zero plot of the FIR filter designed in Equation (3-12) where it can be seen that all the poles are at the origin, inside the unit circle. Hence the system is stable.

![Pole-Zero plot for an FIR filter showing system is unconditionally stable](image)

#### Table 3-3: Values of zeros in Equation (3-12)

<table>
<thead>
<tr>
<th>Zeros</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>z1</td>
<td>0.623423023971260 + 0.781884731391098i</td>
</tr>
<tr>
<td>z2</td>
<td>0.623423023971260 - 0.781884731391098i</td>
</tr>
<tr>
<td>z3</td>
<td>-0.900972941174769 + 0.433875280778807i</td>
</tr>
<tr>
<td>z4</td>
<td>-0.900972941174769 - 0.433875280778807i</td>
</tr>
<tr>
<td>z5</td>
<td>-0.222552912742327 + 0.974920612680800i</td>
</tr>
<tr>
<td>z6</td>
<td>-0.222552912742327 - 0.974920612680800i</td>
</tr>
</tbody>
</table>

### 3.12 Verilog implementation of a 7-tap filter

Digital filters can be implemented in Verilog RTL once we know the coefficients and the difference equation of the filter. The implementation in Verilog RTL and the results after simulation showcase practical issues associated with the digital filter design. The transfer
function of the 7-tap filter with a cut off frequency of 25 kHz is shown in Equation (13), from which the coefficients can be written as;

\[ b_0 = 0.1428, \quad b_1 = 0.1429, \quad b_2 = 0.1429, \quad b_3 = 0.1429, \quad b_4 = 0.1429, \quad b_5 = 0.1429, \quad b_6 = 0.1428, \]

**Step 1**: Firstly, these coefficients are represented in binary format using the Q format representation and 2’s complement method that is explained in Appendix 1.

**Step 2**: The Q.15 format representation of the coefficients are shown below;

\[ b_0 = 0001_0010_0100_0111, \]
\[ b_1 = 0001_0010_0100_1010, \]
\[ b_3 = 0001_0010_0100_1010, \]
\[ b_4 = 0001_0010_0100_1010, \]
\[ b_5 = 0001_0010_0100_1010, \]
\[ b_6 = 0001_0010_0100_0111. \]

**Step 3**: Figure 3-2 shows the structural realization of a 7-tap filter which requires three basic components to realize the filter; the first one is \(Z^{-1}\) which represents the delay element and is implemented using a flip-flop (register). A multiplier is used to compute the multiplication of input data with the coefficients of the filter. Finally, an adder is used to add the output from the multipliers. The Verilog RTL code for this 7-tap FIR filter is included in Appendix 2.1. The next section explains a method for coding filters using non-multiplier logic in Verilog RTL.

### 3.13 Shift and add method for the implementation of filter in Verilog

The realization of filters using multipliers as in Section 2.12 consumes more power in practice due to complex logic computations. This problem can be optimized by shifting the values of the registers where each shift towards the left implies multiplication by an increasing power of 2 and right shift represents a decreasing power of two as shown in Figure 3-11.
### Shift operation vs Mathematical analogy

<table>
<thead>
<tr>
<th>Shift operation</th>
<th>Mathematical analogy</th>
<th>Sum of the divisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>x[n]&gt;&gt;2</td>
<td>x[n]/2^2</td>
<td>1/2^2 = 0.25</td>
</tr>
<tr>
<td>x[n]&gt;&gt;4</td>
<td>x[n]/2^4</td>
<td>1/2^4 = 0.0625</td>
</tr>
<tr>
<td>x[n]&gt;&gt;6</td>
<td>x[n]/2^6</td>
<td>1/2^6 = 0.015625</td>
</tr>
<tr>
<td>x[n]&gt;&gt;8</td>
<td>x[n]/2^8</td>
<td>1/2^8 = 0.00390625</td>
</tr>
<tr>
<td>x[n]&gt;&gt;11</td>
<td>x[n]/2^11</td>
<td>1/2^11 = 0.00048828125</td>
</tr>
<tr>
<td>x[n]&gt;&gt;12</td>
<td>x[n]/2^12</td>
<td>1/2^12 = 0.000244140625</td>
</tr>
<tr>
<td>x[n]&gt;&gt;13</td>
<td>x[n]/2^13</td>
<td>1/2^13 = 0.0001220703125</td>
</tr>
<tr>
<td>Total Sum of the divisions</td>
<td>0.33288574 = 0.3329(Approximately)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3-11: Implementation of multiplication function using shift and add

A version of the 7-tap filter Verilog RTL that uses shift and add functionality is included in Appendix2.1.

## 3.14 Limitations of shift and add and restoration of the precision

Consider the following example where 7 is represented using 4-bit binary notation. Multiply 7 by 0.75, which is simply the sum of 7 right shifted by 1 and 7 right shifted by 2, which should give the result of 5.25. The shift operation in Table 3-4 gives the result of 4 instead of 5.25. This loss of data results in loss of precision. Increasing the length of the register by additional bits solves this problem. In Table 3-5, a 7-bit register is used
where the fractional precision is maintained up to three bits and this gives the result of 5.25 instead of 4.

<table>
<thead>
<tr>
<th>7</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>7&gt;&gt;1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>7&gt;&gt;2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(7&gt;&gt;1)+(7&gt;&gt;2)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 3-4: Implementation 7 multiplied by 0.75 using shift and add

<table>
<thead>
<tr>
<th>7&gt;&gt;1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>3.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>7&gt;&gt;2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1.75</td>
</tr>
<tr>
<td>7&gt;&gt;1+7&gt;&gt;2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>5.25</td>
</tr>
</tbody>
</table>

Table 3-5: Restoration of precision by increasing the bit precision of the register

Hence shift and add functions are a convenient method for multiplication of input data with coefficients in digital filters when implemented in Verilog HDL with sufficient precision of bits.

### 3.15 Summary

This chapter introduces a short study of digital design fundamentals and digital signal processing through practical approach in Verilog and MATLAB using the development of a basic 7-tap digital filter. A useful digital design technique includes implementation of coefficient multiplication through shift and adds operations that help in reducing the complexity of design. The Q point method is also introduced which converts the real coefficients in MATLAB to binary representation in Verilog RTL. The study aspects of digital signal processing include a description of the characteristics of FIR filters such as gain-magnitude response and pole-zero analysis for stability. Chapter 3 deals with digital design principles and coding practices using Verilog and MATLAB. The next chapter provides an introduction to FFTs and dynamic ADC parameters. Additionally, FFTs can be applied to study the FIR filter dynamic performance as designed in this chapter 3.
4 FFT Analysis and Dynamic Test Parameters

4.1 Introduction

This chapter provides an overview of ADC test techniques that are used in the analysis of input and output signals of the test platform designed in Chapter 5. These test parameters are also used for verifying the dynamic performance of digital filters. These important ADC dynamic parameters and the recommended IEEE standards covering these test parameters are introduced. This chapter also covers Fast Fourier Transforms (FFT), which is a mathematical tool used to calculate the dynamic parameters for the determined outputs and inputs of the test system.

4.2 Standard test parameters

Some important ADC parameters and techniques which are used to measure performance are described in the IEEE standard 1241-2000 [10]. The critical ADC test parameters that are specific to the application of ADCs are listed in Table 4-1.
<table>
<thead>
<tr>
<th>Typical applications</th>
<th>Critical ADC parameters</th>
<th>Performance issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio</td>
<td>SINAD, THD</td>
<td>Power consumption. Crosstalk and gain matching.</td>
</tr>
<tr>
<td>Digital oscilloscope/waveform</td>
<td>SINAD, ENOB Bandwidth</td>
<td>SINAD for wide bandwidth Amplitude resolution. Low thermal noise for repeatability.</td>
</tr>
<tr>
<td>recorder</td>
<td>Out-of-range recovery</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Word error rate</td>
<td></td>
</tr>
<tr>
<td>Geophysical</td>
<td>THD, SINAD, long-term</td>
<td>Mill hertz response.</td>
</tr>
<tr>
<td></td>
<td>stability</td>
<td></td>
</tr>
<tr>
<td>Image processing</td>
<td>DNL, INL, SINAD, ENOB.</td>
<td>DNL for sharp-edge detection. High-resolution at switching rate. Recovery for blooming.</td>
</tr>
<tr>
<td></td>
<td>Out-of-range recovery</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Full-scale step response</td>
<td></td>
</tr>
<tr>
<td>Typical applications</td>
<td>Critical ADC parameters</td>
<td>Performance issues</td>
</tr>
<tr>
<td>Radar and sonar</td>
<td>SINAD, IMD, ENOB</td>
<td>SINAD and IMD for clutter Cancellation and Doppler processing.</td>
</tr>
<tr>
<td></td>
<td>SFDR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Out-of-range recovery</td>
<td></td>
</tr>
<tr>
<td>Spectrum analysis</td>
<td>SINAD, ENOB SFDR</td>
<td>SINAD and SFDR for high linear dynamic range measurements.</td>
</tr>
<tr>
<td>Spread spectrum</td>
<td>SINAD, IMD, ENOB</td>
<td>IMD for quantization of small signals in a strong interference Environment. SFDR for spatial filtering. NPR for inter channel crosstalk.</td>
</tr>
<tr>
<td>communication</td>
<td>SFDR, NPR Noise-to-distortion ratio</td>
<td></td>
</tr>
<tr>
<td>Telecommunication</td>
<td>SINAD, NPR, SFDR, IMD</td>
<td>Wide input bandwidth channel bank. Inter channel crosstalk. Compression. Power consumption.</td>
</tr>
<tr>
<td>personal communications</td>
<td>Bit error rate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Word error rate</td>
<td></td>
</tr>
</tbody>
</table>

Table 4-1: Critical ADC parameters [10]

This thesis focuses specifically on dynamic test parameters such as Signal-to-Noise ratio (SNR), Signal-to-Noise and Distortion (SINAD) and Spurious Free Dynamic Range (SFDR). These are the important parameters to be tested in most applications. In addition, these parameters are based on using a single test tone signal. The following sections deal with the Discrete Fourier Transform (DFT) and FFT spectrum analysis that is then followed by more discussion about dynamic parameters in detail.
4.3 Discrete Fourier Transform

As explained in [7], the time domain analysis of a signal allows us to study the amplitude and sample characteristics of a signal. However, in some practical applications such as acoustics, video and communications, knowledge of the signal frequency characteristics is very important. The Discrete Fourier Transform (DFT) is an algorithm that converts the time domain representation of a signal into the frequency domain. Figure 4-1 shows a signal with a frequency of 32 Hz in the time domain and Figure 4-2 shows the same signal in frequency domain where there is a single amplitude peak at 32 Hz and $f_s = 1$kHz. The algorithm is explained in the following sections.

![Figure 4-1: Time domain view of 32Hz sine wave](image1.png)

![Figure 4-2: Frequency domain showing single amplitude peak at 32 Hz](image2.png)
4.4 Discrete Fourier Transform overview

The DFT can be applied to both periodic and non-periodic signals and the formula for DFT is given as in Equation (4-1)[7]

\[ X(k) = \sum_{n=0}^{M-1} x(n)e^{-j\frac{2\pi kn}{N}}. \]  (4-1)

Where \( x(n) \) represents the input sequence in the time domain and \( X(k) \) represents the output sequence which is in frequency domain as shown in Figure 4-3.

![Figure 4-3: Block diagram of a DFT](image)

The DFT helps us to find the frequency component and to investigate the characteristics and performance of the system. Section 4.5 deals with the faster implementation of the DFT known as the Fast Fourier Transform (FFT).

4.5 FFT characteristics

DFT is a transform operation between discrete N-length signal from time to frequency domain and FFT is an efficient algorithm to compute it. The FFT is a robust tool which is used for the spectral analysis of signals. FFTs help in analysing dynamic parameters. However, FFT’s should be calculated with care to achieve a perfect output with number of points in FFT typically assigned in powers of 2.
4.5.1 Spectrum of FFT

The output spectrum of FFT [14] in the frequency domain is a series of M/2 points, where M is the size of FFT. The frequency spacing between each point is $f_s/M$, and the total frequency range covered is DC to $f_s/2$. Here $f_s$ denotes the sampling rate.

![FFT Analysis and Dynamic ADC Test Parameters](image)

Figure 4-4: FFT output for ideal 12-bit ADC, Input = 2.111 MHz, $f_s = 82$MSPS, $M = 8192$ [14]

The width of each frequency bin is $f_s/M$. Figure 4-4 shows a FFT output for an ideal 12-bit ADC. This Figure 4-4 is taken from [14]. It is noted that the theoretical noise floor of the FFT is equal to the sum of the theoretical SNR and FFT process gain which is $10\times\log (M/2)$. Here the noise used in SNR calculations is noise extended over the Nyquist band width DC to $f_s/2$ (Sampling frequency is double the highest sampling frequency). The FFT can be used as analogue spectrum analyser to measure the amplitude of various harmonics and noise components of digitized components.

4.5.2 Spectral leakage

When FFTs are computed theoretically, two assumptions are made:

- Firstly, the sampled data is periodic.
- Secondly, the data is continuous and band limited to a folding frequency.
Often in practice, the second assumption is violated and this produces undesired harmonics as shown in Figure 4-5.

![Plot 1](image1.png)

**Window size N = 18 is used which is not an integral multiple of 32**

![Plot 2](image2.png)

**Window size N = 16 is used which is an integral multiple of 32**

**Figure 4-5: Signals and their spectra without leakage and with leakage [7]**

The second spectral plot has a single tone as expected but the first plot demonstrates leakage. This is because an FFT is conventionally the multiplication between the window of N points and a sine wave in the frequency domain where the zero crossings of the window cancel the harmonics of the sine wave resulting in a pure tone as shown in Figure 4-5 plot 2. In this first plot, the window size is not an integer multiple of sample points which results in spectral leakage. The next section introduces a method to reduce the spectral leakage.

### 4.6 Hanning window

A window is a particular function which starts at zero and ends at zero and has a special shape between the two null points. A Hanning window is used in this application because it reduces spectral leakage and has frequency resolution that suits this application. The time domain representation of Hanning window is shown in Figure 4-6 and mathematically defined in Equation (4-2).
\[ w(n) = 0.5 - 0.5(\cos\left(\frac{2\pi n}{N - 1}\right)) \]  

4.6.2 Signal coherence for FFT tests

Coherent sampling explains the sampling of the periodic signal, where an integer number of its cycles fit into a predefined sample window [1]. This is mathematically defined as in Equation (4-3)

\[ f_{in} = \text{integer} \times \frac{f_s}{N} . \]  

Here \( f_{in} \) is the input frequency and \( f_s \) represents the Sampling frequency, \( N \) is number of points in the FFT. In practice, it is hard to achieve coherent sampling because the input signal is unknown. In such a case, the input signal is multiplied with a non-rectangular window in the frequency domain that reduces spectral leakage. In this design, a Hanning window is used to convolve the input signal in the time domain. In addition, the number of periods of the input waveform should not be a non-prime submultiple of the FFT length. This is mathematically described in Equation (4-4) [16]

\[ \frac{f_{in}}{f_s} = \frac{\text{Number of cycles (prime number)}}{\text{Number of points in FFT}} . \]
The next section explains dynamic test parameters for ADCs and their derivation using FFTs [17].

4.7 Dynamic test parameters

Dynamic test parameters [14], [18], [10] specify the behaviour of an ADC when a sine wave is applied.

4.7.1 Signal-to-Noise Ratio (SNR)

The SNR is the ratio of root mean square (rms) signal amplitude to the rms value of the sum of all spectral components excluding harmonics and the DC component. This is the result when a single tone pure sine wave is applied to the ADC. Equation (4-5) mathematically defines SNR as follows, where n is resolution of ADC.

\[ SNR\ (dB) = 6.02(n) + 1.76\ dB. \tag{4-5} \]

4.7.2 Signal-to-Noise and Distortion (SINAD/SNDR)

Signal-to-Noise and Distortion (SINAD) is the ratio of the rms value of the signal to the rms value of sum of all spectral components excluding DC. The difference between the SNR and SINAD is the difference between the energy in first six harmonics. The following equation is taken from [10].

\[ SINAD\ (dB) = 20 \log\left(\frac{\text{rms}(\text{Signal})}{\text{rms}(\text{Noise}+\text{Distortion})}\right). \tag{4-6} \]

Here, the input signal should be a filtered low noise signal with large amplitude, but not greater than the ADC maximum input value. In this thesis, we use the frequency domain analysis to determine the dynamic test parameter performance.

Noise in the frequency domain is determined as follows:

- In the spectrum \( X(f_m) \), the DC component is reflected where \( f = 0 \).
- The fundamental component where \( f = f_i \) or the difference between the sampling frequency and the input frequency \( (f_s-f_i) \).
- The above two parameters have to be made zero.
- The residual noise spectrum without DC and the tone is computed by Equations (4-7) and (4-8) below, where \( E(f_m) \) is the energy in the signal \( f_m \)
The Spurious Free Dynamic Range (SFDR) is depicted in Figure 4-7; this test parameter is the ratio of the rms value of the signal to the rms value of the peak spurious spectral component of the signal regardless of where it falls in the frequency spectrum. The worst spur may not be the harmonic of the original signal. The following section explains how the FFT is used to analyse output of the digital filters using these principles of dynamic testing.

4.7.4 Total Harmonic Distortion (THD)

THD is the ratio of the root-sum-square value of all the harmonics (2×, 3×, 4×, etc.) to the rms signal level. Generally speaking, only the first five or six harmonics are significant in the THD measurement. In many practical situations, there is negligible error if only the second and third harmonics are included, since the higher order terms most often are greatly reduced in amplitude [14].
4.7.5 Test method for validating the HDL code of the 7-tap filter

A 7-tap filter with a cut off of 25 kHz was designed both in MATLAB and Verilog RTL in chapter 3. Now the objective is to test the designed filter to confirm valid operation. As stated in section 4.7, a single tone sinusoidal signal is used to test the digital filter.

4.7.6 Generation of test pattern

- The sinusoidal signal is generated in MATLAB [17] and is written to a text file.
- This file is imported and provided as a test signal input for filters designed in both MATLAB and Verilog RTL. The filtered output response is in turn saved to a text file.

This procedure is shown as a test block diagram in Figure 4-8.

![Figure 4-8: Top level procedure using Verilog / MATLAB to display spectral performance](image)

4.7.8 Top level method for testing FIR filter

The top level testing procedure contains two parts for verifying the filter design - MATLAB and Verilog. MATLAB code generates the test stimulus which is written to a text file, while a FFT of the input and output of the Verilog filter is also computed. Now the sine wave data that is stored in the input and output of the Verilog filter is also saved. The output signal from the HDL filter is written to a text file and read back into MATLAB for FFT analysis.
4.7.9 FFT computation

The following frequency tones are used for testing the filters. The dynamic parameter Signal to Noise Ratio (SNR) is recorded in Table 4-2.

<table>
<thead>
<tr>
<th>F(kHz)</th>
<th>I/P (dB)</th>
<th>O/P (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>SNR</td>
</tr>
<tr>
<td>186</td>
<td>0</td>
<td>98.2</td>
</tr>
<tr>
<td>430</td>
<td>0</td>
<td>98.5</td>
</tr>
<tr>
<td>1389</td>
<td>0</td>
<td>98.1</td>
</tr>
</tbody>
</table>

Table 4-2: Comparison of dynamic test parameters of a 7-tap filter for different input frequencies

Table 4-2 shows the comparison of dynamic test parameters for the 7-tap filter. The filter is designed for a 25kHz cut-off and sampling frequency of 10MHz. Since the filter uses only 7-taps, a large transition band exists between the start and stop bands. Hence a sharp cut-off is not seen at 25kHz. Another important point to consider is the precision of the output signal from the filter, which is less accurate. This is because the filter is implemented using a finite number of bits and also the multiplication is done using shift and add technique. It can be seen that there is a decrease in the dynamic performance of the output signal as the input signal frequency increases above the cut-off showing the characteristics of a low pass filter as shown in Figure 3-8.

4.8 Conclusion

This chapter provides a discussion of FFTs and how they are used in the analysis of ADC dynamic parameters followed by an example of testing a 7-tap FIR filter that was designed in chapter 3 both in MATLAB and Verilog RTL code. A discussion of dynamic parameters for the output and input signals of the filter is achieved and also the important results are tabulated. The next chapter introduces a design for an ADC and filter test platform based on a mixed-signal FPGA device and board. This test platform can be used in real time to test the ADC using sine waves. For the purpose of understanding digital design and digital signal processing features attached to filters, Chapter 2 and 3 focussed on Verilog coding principles and FIR filter design in MATLAB along with a Verilog RTL code implementation and a testing method. Now the focus is to develop a test platform in Verilog RTL code that is suitable for digital system-on-chip (SoC) designs and in parallel to test the output of an ADC in real time to analyse its dynamic performance.
5 FPGA Test Platform for Data Converters

5.1 Introduction

The aim of this chapter is to develop a FPGA test platform for data converters using the Xilinx ZED board FPGA [19] and Xilinx AMS101 evaluation card [20]. This design can further be used for analysing the dynamic test performance of the Xilinx-based ADC (XADC) using sine wave samples stored into memory and then externally analysed by FFT computation. A key motivation for building this platform was to develop a test platform as detailed in [1] for testing of ADCs and to further analysis of ADC outputs using FIR filters developed in Chapter 3 of this thesis. The system required implementation of a Finite State Machine (FSM) to control the complete design and also to develop a serial-to-parallel communication interface using the SPI protocol. This platform design was very useful for building up expertise and capability in design and prototyping of embedded digital systems using FPGA. The next section provides details on the hardware requirements for the design.

5.2 Selection of hardware

To design this test platform, the hardware must consist of a signal generation block, a data acquisition capture block using memory for storing the dynamic test results and finally a digital controller implemented on FPGA as a programmable design block. To meet these requirements, the Xilinx ZED board FPGA and AMS 101 card were selected. This Zynq FPGA ZED board and add-on AMS101 card has the following specific features.

- Zynq 7000 Series programmable logic (for digital control and FPGA programmability).
- On-chip 12-bit 1MSPS Xilinx ADC (XADC) which is an analogue-to-digital converter.
- Block RAM of 256KB to store the results.
- The AMS101 card has two SPI supported 16-bit Nano DAC’s for the digital-to-analogue conversion of signals generated on the ZED board along with a 2.5V reference device.

The complete board set up is shown in Figure 5-1. As this setup contains both an on-board XADC along with the appropriate high quality external DACs for controlled signal generation; this makes the Zynq board and AMS101 card an appropriate choice to develop a reference test platform. The next section illustrates the top level block diagram for the proposed architecture.
5.2.1 Top level block diagram

This top-level test platform FPGA design diagram is shown in Figure 5-2 and is a closed loop design that has the following sections.

- **SPI_Master** that controls the operation of the system.
- Sine wave generator (DDS1) for which the output acts as test vector.
- Clock reset control block (CRCTL) used to generate the required clock frequencies and reset signals for the rest of the system.
- Finite state machine acting as the System Control Unit (SCU) for the whole design.
- DAC serial peripheral interface (SPI_DAC) for the conversion of parallel data into a serial SPI supported stream.
- An AMS101 board that has two 16-bit Nano DACs for the conversion of digital data into analogue format.
An in-built Xilinx ADC core (XADC) in the FPGA for analogue-to-digital conversion and a block memory core for storage of the samples.

A serial-to-parallel interface slave (SPI) that receives commands from the SPI master and controls access to internal block functionality.

An arbitration block (ARB) - this block arbitrates the memory bus (address and data lines of the memory) between the SCU and the SPI_Master.

PC Control Register (PCR) is a flag register whose values are set by SPI_Master and SCU. Depending on these values, the ARB block arbitrates the memory bus operation.

5.2.2 Operation overview

The design is controlled externally using the SPI master, which starts the generation of sine wave by setting a phase increment value to the DDS block via the SPI_slave and then SPI_Master triggers the SCU by making MasterCntrl high. This ensures the ARB gives memory access to the SCU. Now the SCU is activated and sends digital sine wave samples from the DDS block to the external SPI DACs. The XADC then samples the input waveform and writes the required number of XADC output samples to the memory core. The SCU then flags this event by changing the value of the PCR register. The SPI_master polls the PCR register flag bits and once the SCU has stopped sending samples to memory, the ARB block hands over memory bus access to the SPI_Slave. Now the SPI_Master starts reading the memory data contents via the SPI Interface. The next sections explain each block of Figure 5-2 in more detail.
5.3 Clock Reset Control (CRCTL)

The Clock Reset Control (CRCTL) block diagram is shown in Figure 5-3 and is responsible for the generation of all synchronous clocks and reset operations for the complete system. The CRCTL has two sections Clock Control and Reset Control.

5.3.1 Clock Control

The Clock Control has an input signal MasterCLKp originating from the FPGA board, which is 100MHz and generates three different output clock signals:

1) CRCTL_SysCLKp that is 50MHz and acts as the system clock.

2) CRCTL_DCLKp whose frequency is the sampling frequency for the Direct Digital Synthesizer (DDS) and can be controlled by external switches.

Note: The DACs on the AMS101 evaluation card has a settling time of 8µ seconds and hence the Sampling rate cannot be more than 125 kHz for reliable performance of the DACs.

3) CRCTL_NDacSmp that acts as trigger for the SCU indicating that a new sine wave sample is available for transfer. The frequency of this signal is same as CRCTL_DCLKp.
5.3.2 Reset Control

The Reset Control adheres to the following principle “**assert the reset asynchronously and de-assert synchronously**”. Asserting the reset does not require any extra logic, however de-asserting uses a 3-state synchroniser. This reduces the problem of de-bouncing and metastability.

5.3.3 Timing analysis for the CRCTL

![Clock timing](image)

![RESET timing](image)

Figure 5-4: CRCTL timing

5.3.3.1 Clock timing

The timing waveforms for CRCTL are shown Figure 5-4 where it can be seen that the clock timing section has four key signals (1) the *MasterCLKp* which is 100MHz and (2) the *SysCLKp* which is 50MHz. The other two signals, (3) *CRCTL_DCLKp* and (4) *CRCTL_NDacSmp*, are of same frequency but have different pulse widths.

5.3.3.1 Reset timing

The Reset timing waveforms are also shown in the bottom section of Figure 5-4 where it can be seen that the asynchronous reset signal is immediately asserted whereas de-assertion takes place after three clock cycles of the System clock. The next section illustrates the function of the SPI_Master and Slave sections of the design.

5.4 SPI_Master & SPI_Slave (SPIS) used in this system

The Serial Peripheral Interface (SPI) [21] is a synchronous serial communication interface specification, which is a 4 -wire interface as shown in Figure 5-5. This design
is a communication peripheral between two devices, one that is called SPI_Master and the other is called the SPI_Slave. The SPI_Master commands the SPI_Slave via the Master Out Slave In (MOSI) signal and the slave responds to the Master via the Master In Slave Out (MISO) signal line; the description of these communication lines is as follows;

- **SCLK**: Serial clock signal sent by SPI_Master to slave and all the signals are synchronous to this clock.
- **MOSI**: Master sends data to the slave through this line synchronous to SCLK.
- **MISO**: Slave sends data to the master through this line synchronous to SCLK.
- **SS**: This is the slave select signal and the transmission is valid only when this signal is low.

This is a full duplex communication where the master and slave can communicate simultaneously as shown in Figure 5-5.

**Clock timing**

![Clock timing diagram](image)

Figure 5-5: SPI timing diagram used in this design
Protocol for MOSI and MISO

Command bits Address Bits Data Bits
C7 C6 C0 A13 A12 A0 D11 D10 D9

Protocol for MOSI and MISO

5.5 Function of Master and Slave

This design uses a 34-bit sub frame to transmit data from SPI_Master to SPI_Slave where the first 8 bits represent the action to be performed by the SPI_Slave and the next 14 bits represents the address information, and the final 12 bits is the data packet. The timing diagram of SPI_Interface is shown in Figure 5-5 and the function of the SPI_Master is to command the slave peripheral as shown in Figure 5-6. The response of the SPI Slave for different commands is shown below.

- Command DDS Write: Initially the Master sends the 34-bit frame with command field set as 8’h20. The Slave reads the data and separates the sub fields; depending on the command data it sets the phase increment value SPIS_pi and also T_valid value to the DDS (explained in Section 5.6).
- Command PCR Write: If the command sub field received by the slave matches 8’b01 it makes PCR[1:0] = 2’b01 (explained in Section 5.12), which triggers the state machine and arbitrates the bus access to SCU.

- Command PCR Read: The master continuously polls the PCR_DataRd register for a specific value that indicates all the data is written into the memory.

- Command Mem Read: Master initiates this command only when the PCR Read command reads the required value. Mem Read command sets PCR to 2’b11 and the master now starts reading the data from the memory via the SPI Slave.

### 5.6 Direct Digital Synthesizer (DDS) generator

The DDS generator is an in-built IP core [22] supported by the Xilinx FPGA design tools, which is used to generate a digital sine wave of the desired frequency based on an integer which is known as the phase increment. The DDS generator (named DDS1) is shown in Figure 5-7 which has three inputs –

1. aclk, the sampling frequency of the digital sine wave which comes from the CRCTL block,

2. Phase increment

3. T_valid signal which are set by SPI_Master through the SPI Interface.

The DDS1 block has two output signals

1. m_axis_data_tdata and

2. m_axis_data_tvalid.

In this design a 16-bit length digital sine wave is generated by the DDS generator which is the m_axis_data_tdata. The m_axix_data_tvalid signal states that the sine wave is valid only during high period of this signal.

The DAC on the AMS101 card (explained in Section 5.8) has an input voltage range from 0 to +VDD. The DDS generator generates the digital sine wave, which varies from −VDD to +VDD. The DDS2 block introduces a DC offset so that the signal varies from 0 to +2VDD. The ADC on the FPGA is then fed back with the output of DAC on the AMS101 card and has an input voltage range of a maximum 1 Volt. The sine wave after DC offset has a maximum 16-bit digital value of 65535, which corresponds to 2.5 volts DC. (The
DAC contained on the AMS101 card utilises a 2.5V reference. In order to meet the specification of XADC the sine wave is further scaled down to 0.8 volts DC.

Figure 5-7: DDS1 and DDS2

5.7 System Control Unit (SCU)

The system control unit [23] is the heart of the system, which controls all the platform operations between each sub-design block. The main aim of the SCU is to transmit the required number of samples through the SPI to the DAC and from the DAC to the XADC and then finally to store ADC samples in the block memory. The SCU is triggered by an external signal initiated by the PC or Laptop.

5.7.1 Finite State Machine (FSM) controller

This is a Moore-style state machine [24] with eleven states. The state diagram for the FSM is shown in Figure 5-8 and a short description for each supported state is provided.
**FPGA Test Platform for Data Converters**

**Figure 5-8: SCU state diagram**

**IDLE:** This is the first state of the FSM and the SCU stays in this state until the reset is low; once the Master_Cntrl and reset signals high the FSM moves to the DAC_A state.

**DAC_A and DAC_A_UPDATE:** In this state, the SCU_DAC_SEL signal and the SCU_DAC_UPDATE signal are made high such that DAC_A on the AMS101 board is selected. The DAC_SPI interface transmits a 32-bit data format, which includes the 16-bit data length along with a command and address length of 8 bits in total (the rest of the bits are don’t cares). The flag_w signal goes high once all these bits are transmitted and then the FSM moves to the DAC_B state.

**DAC_B and DAC_B_UPDATE:** These FSM states operate similarly to DAC_A. In this state, the only difference is that it makes SCU_DAC_SEL low so that the DAC_B is selected and SCU_DAC_UPDATE is made high for the update of a new sample. Now
the FSM moves into DAC_B_UPDATE where DAC_SPI interface transmits the sine wave samples generated by DDS and then moves to stCONV_START state.

**CONV_START and CONV_WAIT:** In CONV_START, the FSM triggers the XADC for analogue-to-digital conversion and moves to CONV_WAIT, which is an unconditional transition - now the SCU waits for the XADC_EOC (End of Conversion) signal back from the XADC IP core to signal a data conversion complete once this signal is high, the FSM moves to stDRP_1.

**DRP_1 and DRP_2 and XADCrd:** The dynamic reconfiguration port (DRP) [25] supported by the XADC and collects all the analogue to digital converted samples.

**Mem_Write:** This FSM state stores the converted samples in the Block Memory core and then moves back to the state DAC_B. This loop between DAC_B and Mem_Write continues until the required number of samples is written into the memory. In this design, an 8192 size FFT is computed so the numbers of samples stored are 8192.

### 5.8 Digital to Analogue Converter (DAC)

The AMS101[20] board has two 16-bit Nano AD5065 DACs [26] named DAC A and DAC B. The DAC A generates an offset voltage and DAC B converts the 16-bit serial input sine wave from the SPI (explained in Section 5.10) into an analogue sine wave with a range of 0 to +VDD (0.8V). The DAC setup is shown in Figure 5-9.

![Figure 5-9: DAC setup](image-url)
The AC characteristics of the DAC states that the maximum output settling time of the DAC is 8µs and hence the maximum update rate from the SPI is set to 125 KHz.

5.9 Xilinx Analogue to Digital Converter (XADC)

The ZED board FPGA has an on-chip dual input 12 bit XADC [25, 27] whose maximum sampling rate is 1MSPS and can be operated in two different modes.

1) Unipolar mode
2) Bipolar mode

5.9.1 Unipolar mode

The XADC has two inputs Vp and Vn and in this mode as per data sheet [25], Vp has a range of 0 to 1 volt and Vn has a range of 0 to 0.5V. Hence the full scale input voltage of the XADC would be in the range of -0.5 to 0.5 V as shown in Figure 5-10. In this design, VP is set to operate from 0 to 0.9V while Vn is chosen as 0 to 0.4V.
5.9.2 Bipolar mode

Bipolar mode operation has two inputs $V_p$ and $V_n$ and the input voltage $XADC_{in}$ swings around the common mode voltage $V_n$; thereby the output voltage is $V_p - V_n$ as shown in Figure 5-11. The maximum input range of $V_p$ is 0 to 1 V and $V_n$ is 0 to 0.5V and hence the maximum range for $XADC_{in}$ varies from $-0.5$V to $0.5$V. In this design $V_p$ and $V_n$ is selected in the range of 0 to 0.8V and 0 to 0.4V respectively which is below the XADC full scale voltage to avoid clipping. Clipping causes unwanted distortion components to appear at the output of the ADC.

Figure 5-11: XADC bipolar mode
5.10 SPI DAC interface

The DAC on the AMS101 card is a serial input DAC and a parallel to serial interface is required to convert the 16-bit parallel word to a serial bit stream. This is done by SPI_DAC interface which has four prominent outputs as shown in Figure 5-12. (1) Serial clock (SPI_clk), (2) LDAC (SPI_ldac), (3) Data in (SPI_mosi) and (4) Chip select (SPI_cs).

- The SPI_mosi is a serial data input to the DAC and the input rate is controlled by SPI_clk.
- The SPI_cs signal is an active low signal – when active low, the DAC is updated with the values of SPI_mosi on the falling edge of the SPI_clk.
- The LDAC in this design is kept high continuously.
- The DAC SPI interface has 5 inputs –
  (1) SCU_DACSel
  (2) SCU_DacUpdate which are generated by the SCU.
  (3) Clock
  (4) Reset signal from the CRCTL block
  (5) Sine wave data sample from the DDS generator.
5.10.1 Timing analysis of SPI DAC interface

![Timing diagram for the SPI DAC interface](image)

The timing diagram in Figure 5-13 illustrates the operation of the SPI DAC interface. Initially the SCU_DacUpdate and SCU_DacSel signal are asserted by the SCU and a new sample is made ready for transmission while the DAC is selected on the AMS101 board.

5.11 Block memory

The FPGA device has an in-built block memory supporting 256 kilobytes [28] which can be accessed using the block memory generator which is a Xilinx IP (Intellectual property) core. Figure 5-14 shows the IP core from the block memory generator, which is a single port memory set to a depth of 8190 words and a data width of 12 bits. This design supports a clock (CLKA), reset (RSTA) and write enable (WEA) signals, in addition to an address (ADDRA) and data lines (DINA), which are also inputs. The read and write operations are controlled using the WEA signal. When WEA is high, the DINA is written to the memory location defined on the address line and when WEA is low the DOUTA is read from the memory location specified on the address line.
5.12 Arbitration Block (ARB) and PC Controlled Register (PCR)

Generally, in digital design architectures, multiple devices try to communicate with each other using single address and data bus lanes. In this design, both the SPI_Master and SCU need access to the memory, where the SCU writes data into the memory and the SPI_master reads data from the memory. A technique called arbitration is used to multiplex the memory bus between the SCU and SPI_Master blocks. This arbitration uses a multiplexer whose select line is controlled by a two-bit value from the PCR. Table 5-1 lists the arbitration bus operation between SCU and SPI Slave blocks. The PCR is a 12-bit register whose input values are controlled by both SCU and SPI_Master.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Select line[1:0]</th>
<th>Arbitration Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00</td>
<td>SCU with trigger signal as 0</td>
</tr>
<tr>
<td>2</td>
<td>01</td>
<td>SCU with trigger signal as 1</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>SCU with trigger signal 0 and flags the SPI_master</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>SPI (Master through Slave)</td>
</tr>
</tbody>
</table>

Table 5-1: Bus arbitration based on PCR

<table>
<thead>
<tr>
<th>Operation</th>
<th>Signal</th>
<th>PCR[1:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SPIS_Twr</td>
<td>01</td>
</tr>
<tr>
<td>2</td>
<td>SPIS_MemRd</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>SCU_PCRzero</td>
<td>00</td>
</tr>
<tr>
<td>4</td>
<td>SCU_RdStart</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 5-2: Values of PCR register
Table 5-2 lists the PCR values that are assigned based upon the input signals. The ARB block as shown in Table 5-1 arbitrates the memory bus access between the SCU and PCR based upon this PCR values.

Figure 5-15: PCR and ARB integration

Figure 5-15 shows the integration of the ARB and PCR blocks - it can be seen that depending on the signal settings from the PCR, the ARB block switches the memory access.
5.13 Key specifications and overall timing diagram

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Conditions/Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MasterCLKp</td>
<td>-</td>
<td>100</td>
<td>-</td>
<td>MHz</td>
<td>On board FPGA Clock</td>
</tr>
<tr>
<td>CRCTL_SysCLKp</td>
<td>-</td>
<td>50</td>
<td>-</td>
<td>MHz</td>
<td>Digital System Clock</td>
</tr>
<tr>
<td>CRCTL_DCLKp</td>
<td>-</td>
<td>-</td>
<td>125</td>
<td>KHz</td>
<td>Sampling Frequency for the sine wave generator</td>
</tr>
<tr>
<td>CRCTL_NDacsmp</td>
<td>-</td>
<td>-</td>
<td>125</td>
<td>KHz</td>
<td>Goes high only for one clock cycle of Master Clock</td>
</tr>
<tr>
<td>Dynamic Input Range for ADC</td>
<td>-0.5</td>
<td>-</td>
<td>+0.5</td>
<td>Volts</td>
<td>Maximum input range for the ADC</td>
</tr>
<tr>
<td>DAC VREF</td>
<td>-</td>
<td>-</td>
<td>2.5</td>
<td>Volts</td>
<td>Reference voltage of the DAC</td>
</tr>
<tr>
<td>SPI_clk</td>
<td>-</td>
<td>25</td>
<td>-</td>
<td>MHz</td>
<td>SPI serial clock to DAC</td>
</tr>
<tr>
<td>ADC conversion rate</td>
<td>-</td>
<td>-</td>
<td>1M</td>
<td>Samples/Second</td>
<td>XADC Conversion rate</td>
</tr>
<tr>
<td>DAC input range</td>
<td>-</td>
<td>-</td>
<td>2.5</td>
<td>V</td>
<td>DAC input range</td>
</tr>
<tr>
<td>DAC-Output Range</td>
<td>0</td>
<td>-</td>
<td>2.5</td>
<td>V</td>
<td>DAC output range</td>
</tr>
<tr>
<td>XADC</td>
<td>-0.5</td>
<td>-</td>
<td>0.5</td>
<td>Volts</td>
<td>XADC input range</td>
</tr>
</tbody>
</table>

Table 5-3: Key specifications

Table 5-3 outlines the key specification criteria for the top level system design. Figure 5-16 also illustrates the timing diagram of the overall design. The functionality of the total system is explained below.

- Initially after the active low reset is made high, the clock reset control block generates the system clock for the complete design and the sampling clock for the DDS generator.
- The SPI_Master sets the value of phase increment in the DDS and, depending on the sampling clock frequency, the DDS block generates the 16-bit coherent sine wave output to the DAC.
Now the SPI_Master triggers the SCU block by making master control high and sets the PCR value as 2'b01. Depending upon the value of PCR, the ARB block gives the memory access to the SCU. The SCU FSM also generates DAC_Update and DAC_sel signal to the SPI DACs on the external AMS101 board. The main operation of the SPI DAC interface is to convert the parallel 16-bit stream to serial and also to select either DAC A or DAC B, depending on the control form SCU.

Depending on the trigger signals from the SCU, the SPI selects DAC A firstly to set an offset voltage by feeding it with a serial bit stream of data. Now the SCU resets the value of PCR.

Then SPI DAC interface then selects DAC B and feeds it with a serially converted digital sine wave from the DDS generator. During this process the SPI master continuously polls the value of the PCR register.

Figure 5-16: Timing diagram of overall system
The DACs on the AMS101 card converts the input digital bit stream into an analogue voltage for input to the XADC. Then the FSM controller waits for 10 system clock cycles.

The analogue signal from the DAC is fed to XADC through the XADC header and ADC conversion starts taking 9 clock cycles to complete. Now the data is stored in the XADC DRP registers.

Finally, the XADC digital sample is stored in block memory. This loop continues until the required number of samples (which is the FFT size) is stored in the block memory.

### 5.14 Verilog coding style

A key focus of this thesis was to develop a Verilog coding philosophy that includes good indentation and alignment to make the code readable. Secondly, a standard naming convention is also utilised where the signal name starts with the name of the source block from which it is generated. This helps in identifying the source and destination of the signals. Thirdly, the Verilog code is written in such a way so that the design is easy to understand and portable to both FPGA and ASIC IC designs. Further details of the coding style and methodology adopted is included in Appendix 5. The Verilog RTL code for the complete top-level design and hierarchical sub-blocks is contained on the CD in Appendix 2.

### 5.15 Conclusion

This chapter emphasizes the design of an embedded test platform for ADC using FPGA that uses the methodology and structural approaches introduced in Chapter 2. The design is implemented to ensure that the test platform works properly to meet the specification with minimal number of re-iterations and code fixing. A state machine diagram and description of the controller was provided in addition to a detailed description of all the sub-blocks within this top-level design. The Verilog code for each block and the total system is contained in Appendix 2. The next chapter explains the test bench that was developed to validate this platform design and illustrates the simulated results along with some FIR application scenarios.
6 Simulation and Testing of FPGA Test Platform for Data Converters

6.1 Introduction

Chapter 5 introduced the FPGA test platform for data converters based on a digital design implementation that is coded in Verilog RTL. Once the prototype is designed, the next phase is to simulate and validate that prototype for expected functionality. Testing is a key phase in the design flow and this chapter develops a test bench for the prototype to investigate and fix flaws in the functionality. The basic test cycle [24] followed in this chapter has three phases as shown in Figure 6-1. Initially, the stimulus is generated and applied to the device under test (DUT). Secondly, the DUT is allowed time to respond, and finally the results are checked for the expected functionality. Xilinx ISIM, which is a RTL simulator integrated with the Xilinx FPGA Tool suite [29], is used to run the simulations. There are three main outputs in this design, (1) output from the DDS generator, (2) output from the XADC and (3) output of the memory read by the SPI_Master. All these signals are exported to three separate text files during simulation and are then imported into MATLAB to perform FFT analysis and compute the dynamic parameters. An application of this embedded test platform is developed in the later section, which involves the design and implementation of 3, 7, 11 and 21-tap FIR filters in Verilog HDL to post correct the output data of an ADC. The next section deals with the top level block for testing the platform and its implementation.
6.2 Top level test bench

Figure 6-2 illustrates the top level test bench for the global test platform which has the following blocks:

- Clock & Reset
- SPI_Master
- Sys_top
- spi_dac_model
- XADC output and sine wave input

The Verilog RTL code for the top-level test bench is contained in Appendix 3.
6.2.1 Clock & Reset

The Clock block generates a clock signal of 100 MHz using the Verilog ‘forever’ command which mimics the on board FPGA clock. This signal is fed to two blocks - one is the sys_top and the other is the SPI_Master. The Reset block is used to generate an asynchronous reset signal.

6.2.2 SPI_Master

This block represents the SPI_Master which is designed using behavioural modelling. This block has two inputs - Clkp and MRstn which are generated by the Verilog test bench. This SPI_Master reads the data from the memory through the SPI interface and that signal is exported to text file Mem_out.txt.

6.2.3 Sys_top

This block instantiates all the components of FPGA test platform excluding the AMS 101 board and SPI_master, which represents the complete design developed in Chapter 4. This block has three signals as input fed by the test bench. The MasterCLKp, which mimics the on board FPGA clock, the system reset signal and then a 3-bit Master switch bus.

6.2.4 spi_dac_model

This block mimics the operation of the serial DAC on AMS 101 board. A text file called XADC_in.txt is generated, which has analogue inputs to the XADC block instantiated in SYS_top during simulation.

- XADC_in.txt: This holds the analogue inputs Vp and Vn (explained in Chapter 4) and also the simulation time in three column formats shown in Appendix 4. This text file path is written in the Verilog functional model of XADC, which reads this file during simulation.

6.2.5 XADC output and sine wave input

The test bench export two text files that hold the output generated by the XADC (XADC_out.txt), which is a 12-bit signal and the Sine wave generated by DDS generator which is a 16-bit signal (DDS_Usine.txt). The next section explains the test flow procedure.
6.3 Verilog test bench RTL aspects

Verilog RTL test bench code aspects such as module instantiation, clock and reset generation, and ifdef directives are discussed in this section. The test bench must instantiate the module or unit under test for which the test bench is designed. The module instantiation style is shown below;

```
// Instantiating SPI MASTER
// --------------------------------------------------------------------
SPI_bfm_12 u_SPI_bfm_12 (  
  .CLKp      (CLKp),  
  .MISO      (MISO),  
  .MOSI      (MOSI),  
  .SCLK      (SCLK),  
  .SSn       (SSn)  
);
```

Figure 6-3: Sample code showing the instantiation of SPI Master

In some cases the module needs to be instantiated only during simulation but not during synthesis; in that case the module instantiation is done using a `ifdef compiler directive as shown below.

```
`ifdef SPI_DAC_MODEL
    Spi_dac_model  Spi_dac_model_u (  
        .CRCTL_SysCLKp  (CRCTL_SysCLKp),  
        .MasterCLKp    (MasterCLKp),  
        .MasterSwitch  (MasterSwitch),  
        .sclk          (SPI_clk),  
        .resetn        (MasterRSTn),  
        .cs            (SPI_cs),  
        .ldac          (SPI_ldac),  
        .mosi          (SPI_mosi),  
        .SCU_RAM_read  (SCU_RAM_read),  
        .SCU_Wen       (SCU_Wen),  
        .SCU_FFT_points (SCU_FFT_points),  
        .CRCTL_SynRSTn (CRCTL_SynRSTn),  
        .SCU_Address   (SCU_Address)  
    );
`endif
```

Figure 6-4: Sample code showing the use of `ifdef

Here the Spi_dac_model is instantiated only when SPI_DAC_MODEL directive is set to 1 in the start of the code using `define SPI_DAC_MODEL 1. Signal generation is also an important aspect of test bench design. The following style was used to generate the clock and reset signal generation for Verilog RTL verification.
Figure 6-5: Sample code showing the clock generation in Verilog test bench

The parameter pPeriod2 defines the clock period to generate a 6.25ns clock signal. The Master reset signal is activated at the start of the test and then deactivated after 768ns for this simulation. The next section introduces the Test flow procedure flow chart.

Figure 6-6: Sample code showing the reset generation in Verilog test bench
6.4 Test Flow Procedure

Figure 6-7: Test flow procedure

Figure 6-7 shows the flow chart for the testing procedure and basic test cycle. As explained in chapter 5, the primary task of this prototype is to write a predetermined number of XADC output samples into block memory through the State machine, and read the samples written in the same memory through the SPI_Master. Finally, this data is monitored to check the signal quality using FFT analysis. Since there is only a single memory used, the memory bus is arbitrated between the SPI_Master and the SCU.

- Initially the clock, reset, switch signals are generated.
- The task is initiated, which feeds the DDS block with the phase increment value. Simultaneously, this signal is exported to a text file.
- Following this task, PCR_Trigger is initiated, which places the state machine in a loop and sets the PCR bits [1:0] = 2'b01.
Now the state machine continues until all the predetermined number of samples are sent to the external DAC, captured by the XADC and then written into memory. Once all the FFT samples are written, the SCU makes the PCR [1:0] = 2'b10. In parallel, the output of the XADC is exported to the text file.

During the previous step, the SPI_Master continuously polls the PCR [1:0]. Once the PCR [1:0] = 2'b10, the SPI_Master makes the PCR [1:0] = 2'b11 which gives the SPI master access to the memory bus and thereby reads the data from the memory.

The data that is read from the memory through the SPI interface is also written to a text file.

### 6.5 Xilinx ISIM simulation

This section covers the testing of the DUT in Xilinx ISIM. Initially the test stimulus is applied to the DUT and the response of the DUT is simulated and shown below. Figure 6-8 shows the simulation results of the system where the SCU writes data into the memory.
Figure 6-8: Simulation of the design where SCU writes data into the memory
Figure 6-9: Simulation results where the data is read from memory through MISO
Figure 6-9 shows the ISIM simulation results where the data is read from the SPI_Master and this can be clearly seen on the MISO line. Once this simulation is complete, the data is written into the text files for verification and analysis. The next section explains about the analysis using FFT.

### 6.6 MATLAB for FFT spectral analysis

The DDS block generates a sine wave of required frequency, which is serially output to the DAC. This wave is input to the XADC then is sampled to give a 12-bit digital output. This data is stored in the memory and read back via the SPI Master. To validate the results, three text files are imported in MATLAB and FFTs are performed. This should give three single tones of the same frequency.

#### 6.6.1 Results and FFT plots using MATLAB

This section explains the frequency computation using DDS and its frequency domain plots for validation. From Chapter 5, the DDS generator is used to generate a sine wave that needs two parameters as inputs - one is the sampling frequency and second is the phase increment value. The sine wave from the DDS generator and the output signal of the XADC must adhere to the coherent sine wave generation rule which holds the following relationship [16]

\[
    f_{in} = \text{integer (prime)} \times \frac{f_s}{N}. \tag{6-1}
\]

The DDS Phase increment is calculated using the following rule as in Equation (6-2)

\[
    \Delta \theta(\text{phase increment}) = \frac{f_{in} \times 2^{16}}{f_s}. \tag{6-2}
\]

Five different frequency tones are generated which satisfies the above conditions and are listed in the Table 6-1.
Table 6-1: Coherent frequency tones generated by the DDS based on phase increment values

<table>
<thead>
<tr>
<th>Tone Number</th>
<th>Fin (kHz)</th>
<th>Phase Increment</th>
<th>phase Increment</th>
<th>periods</th>
<th>Fs (kHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.05887</td>
<td>3448</td>
<td>110101111000' (12 bits)</td>
<td>431</td>
<td>96.154</td>
</tr>
<tr>
<td>2</td>
<td>11.702</td>
<td>7976</td>
<td>1111100101000' (13 bits)</td>
<td>997</td>
<td>96.154</td>
</tr>
<tr>
<td>3</td>
<td>24.9892</td>
<td>17032</td>
<td>100001010001000' (15 bits)</td>
<td>2129</td>
<td>96.154</td>
</tr>
<tr>
<td>4</td>
<td>35.224</td>
<td>24008</td>
<td>101110111001000' (15 bits)</td>
<td>3001</td>
<td>96.154</td>
</tr>
<tr>
<td>5</td>
<td>39.99</td>
<td>27256</td>
<td>110101001111000' (15 bits)</td>
<td>3407</td>
<td>96.154</td>
</tr>
</tbody>
</table>

Figure 6-10 shows the FFTs computed on the data written to the XADC_out.txt which is red in colour and the DDS_Usine.txt which is blue in colour. The data captured from memory - RAM_read.txt is green in colour. It can be seen that the output of XADC and block memory are identical, so they overlap. This validates that the data generated by DDS and processed by XADC and then stored and extracted from block memory are identical having the same frequency. Here the frequency of the sine wave generated is 5.05887 kHz.

Figure 6-10: FFT plots for different output tones of different frequencies

6.7 Analysis of dynamic parameters

The FFT is a fantastic tool that helps to analyse the dynamic parameters of the signals and also confirms the full scale range of the single through the single frequency plot.
The key specifications of the input signals are given in Table 6-2 where the DDS signal can have a range of 0 to 2.5 volts, however only 0.8 volts is generated to meet the input requirements of XADC. The XADC input can go from \([-0.5, 0.5]\) volts bipolar mode (see chapter 5) with 1V declared as full-scale. In order to prevent signal clipping, the XADC input nominal value is chosen as \([-0.4, 0.4]\) volts, which is 80% of full-scale.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Bit length</th>
<th>Min(nominal) (Volts)</th>
<th>Max(nominal) (Volts)</th>
<th>LSB</th>
<th>Full scale (Volts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDS</td>
<td>16</td>
<td>0</td>
<td>0.8</td>
<td>2.5/2^16</td>
<td>2.5</td>
</tr>
<tr>
<td>XADC &amp; Memory</td>
<td>12</td>
<td>-0.4</td>
<td>0.4</td>
<td>1.0/2^12</td>
<td>[-0.5, 0.5] 1 volt</td>
</tr>
</tbody>
</table>

Table 6-2: Key specifications of the signals DDS, XADC, Block memory

Table 6-3: Comparison of SNR and SINAD for DDS, XADC, Block memory

Table 6-3 shows the analysis of the dynamic parameters where one column shows the expected values and the other shows the obtained values. It can be seen that SNR for DDS (16-bit signal) is 82.9 dB whereas the expected value is 98 dB. This is because the signal only achieves ~32% of full scale and there is a loss of approximately 3-bits. For the XADC and memory blocks which are 12 bit signals, the expected value is 74dB and the practical result is 71.59 dB; here the signal is considered 80% of full scale giving a loss of approximately 0.4 bits. In summary, the test platform is working according to the aim of the design and the results validate the functionality of the test platform. Hence this platform achieves its purpose in testing the A/D converters using the FFT spectral analysis and creating a roadmap into development of an adaptive test platform.
Table 6-4: Expected loss in the amplitude of the signal in frequency domain

<table>
<thead>
<tr>
<th>Input Signal</th>
<th>dBFs = 20*log10(nominal/Full scale)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDS</td>
<td>= 20*log10(0.8/2.5) = -10 dB</td>
</tr>
<tr>
<td>XADC &amp; Block Memory</td>
<td>= 20*log10(0.8/1) = -2 dB</td>
</tr>
</tbody>
</table>

The FFT results also help us to look at the loss in the amplitude of the signal. The Y-axis of the FFT is normalized to 0dBFS, it means the signal is generated to its full scale in the time domain. In this design, all of the signals are seen to be below full scale. Table 6-4 illustrates the attenuation of the signal in decibels. These results show how FFTs can be used for the spectral analysis of important signals in the development of this test platform.

### 6.8 Potential application of FPGA embedded test platform

Up to this point, an embedded test platform is developed in Verilog HDL for Zynq board FPGA and is tested and validated for correct functionality using FFT spectral analysis. The theme of this test platform is to generate a dynamic test signal (Sine wave) that is serially converted and externally input to a DAC on an AMS-101 card. The analogue output of this DAC is then fed directly to XADC as input, which is sampled at the same DAC update frequency. The XADC samples the DAC waveform and the digital data is captured and stored in block memory, which is controlled by SCU. Using a SPI_Master, the block memory is accessed through the SPI interface and the captured samples are then used to analyse the performance of the XADC through computation of FFT, which is used to test the dynamic parameters for signal quality.

This section introduces a potential application of the FPGA test platform where the outputs of ADC are applied to FIR filters to enable further analysis. Multiple FIR filters of 3,7,11 and 21-taps are developed in Verilog HDL to connect to the outputs of ADC. This application is developed in consideration of the case of a SAR ADC converter. Generally, FIR filter functionality assumes that the complete data word is available at the input of the filter. In real-time analogue-to-digital sampling systems such as in Successive Approximation Analogue to Digital Converters (SAR ADC) [17], typically the MSB data is available first before the LSB data bits are determined for each output sample. In this case optimal filtering using the output of the ADC bits attached to a FIR filter is difficult because of the length of delay in the output word formation from the SAR. Figure 6-11 shows the pictorial representation of the output from the SAR ADC assuming an equal time allocation for MSB and LSB data formation.
Figure 6-11 illustrates an example of a 10-bit SAR converter producing the MSB (5bits) before the LSB (5bit) data. Hence at the start of the clock update cycle only the current MSB data and the previous LSB data is available to the input of the filter followed by successive LSB and MSB data. The motivation for this chapter is to determine if we can optimise the cycle length so that filtering is carried out at every falling edge of the clock, helping to reduce the latency between the SAR and FIR output. There are four scenarios that are considered;

**Scenario 1**: Filtering is always carried out on current MSB with initial LSB data set to zero for the first cycle update into FIR filter. Subsequent stages of the FIR filter will use the correct MSB & LSB data.

**Scenario 2**: Filtering is always carried out on current MSB with previous LSB data applied to 1st stage of filter. Subsequent stages of the FIR filter will use the correct MSB & LSB data.

**Scenario 3**: Filtering is always carried out on current MSB with random data output from linear feedback shift register (LFSR) applied to 1st stage of filter. Subsequent stages of the FIR filter will use the correct MSB & LSB data.

**Scenario 4**: The sampling frequency is increased, which reduces the change in the LSB of successive samples of the sine wave.

The remainder of this chapter project outlines the key stages of implementation and the results analysis.
6.8.1 Verilog implementation of scenario 1

Figure 6-12: Input to the filter where the first LSB is initially set to zero

Figure 6-12 shows the timing diagram of the 10-bit input to the FIR filter where the first LSB is initially set to zero and the following stages are fed with the correct MSB and LSB data.

Figure 6-13: FIR structure of (a) standard FIR filter and (B) structure used for scenario 1

Figure 6-13 shows the structure used for the realization of FIR filters. A reference filter was designed as outlined in chapter 3 where the complete data word is available at the input for filtering. The Research filter is the design where stage 1 is directly fed with MSB
data appended with 5 zeros. The following stages are fed through a buffer that combines the LSB and MSB data thereby forming the complete data word. The following section shows the time domain and frequency domain analysis of research filter compared to the Golden filter. The Verilog RTL code for the implementation of research filter is shown in Appendix 2.

Figure 6-14: Output of both reference and research 3 tap filters (scenario 1)

Figure 6-14 depicts the output of the low pass FIR filter of 3 taps when the input is fed with a 58 kHz sine wave. It can be seen that there is a loss of data between the reference filter and the research filter because the first coefficient is fed with MSB appended with zeros instead of the actual LSB data.
Figure 6-15 shows the FFT plot for the input and output of the reference and research FIR 3 tap filters for a tone of 58 kHz and it can be seen that the harmonics and noise floor is high in the research filter when compared to reference filter. The next section illustrates the dynamic performance for different tap filters in Scenario 1 where the LSB of the first coefficient is set to zero.

6.8.1.1 Comparison of SNR, SINAD, SFDR, THD performance for different input frequencies for the Reference and Research filters

This section compares the dynamic parameters such as SNR, SINAD, SFDR etc., for 3,7,11,21-tap filters which is shown in Table 6-5, Table 6-6, Table 6-7, and Table 6-8 respectively. It can be seen that the dynamic parameters (SNR,SINAD,THD,SFDR) of the research filter increases as the number of taps increase and is approximating the SNR of the reference filter.

<table>
<thead>
<tr>
<th>F (KHz)</th>
<th>I/P Amplitude (dB)</th>
<th>O/P Amplitude (dB) Reference</th>
<th>O/P Amplitude (dB) Research</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>A</td>
<td>SNR</td>
</tr>
<tr>
<td>58</td>
<td>-6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
<tr>
<td>123</td>
<td>-6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
<tr>
<td>305</td>
<td>-6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
</tbody>
</table>

Table 6-5: Comparison of 3-tap FIR filter performance – scenario 1
Simulation and Testing of FPGA Test Platform for Data Converters

Table 6-6: Comparison of 7-tap FIR filter performance – scenario 1

<table>
<thead>
<tr>
<th>F (kHz)</th>
<th>I/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reference</td>
<td>Research</td>
<td>Reference</td>
</tr>
<tr>
<td>A</td>
<td>SNR</td>
<td>SINAD</td>
<td>THD</td>
</tr>
<tr>
<td>32</td>
<td>6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
<tr>
<td>52</td>
<td>6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
<tr>
<td>137</td>
<td>6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
</tbody>
</table>

Table 6-7: Comparison of 11-tap FIR filter performance – scenario 1

<table>
<thead>
<tr>
<th>F (kHz)</th>
<th>I/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reference</td>
<td>Research</td>
<td>Reference</td>
</tr>
<tr>
<td>A</td>
<td>SNR</td>
<td>SINAD</td>
<td>THD</td>
</tr>
<tr>
<td>16</td>
<td>6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
<tr>
<td>32</td>
<td>6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
<tr>
<td>64</td>
<td>6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
</tbody>
</table>

Table 6-8: Comparison of 21-tap FIR filter performance – scenario 1

<table>
<thead>
<tr>
<th>F (kHz)</th>
<th>I/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reference</td>
<td>Research</td>
<td>Reference</td>
</tr>
<tr>
<td>A</td>
<td>SNR</td>
<td>SINAD</td>
<td>THD</td>
</tr>
<tr>
<td>16</td>
<td>6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
<tr>
<td>25</td>
<td>6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
<tr>
<td>32</td>
<td>6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
</tbody>
</table>

The next section explains scenario 2 where the first coefficient of the filter is fed with present MSB and previous available LSB, whereas the subsequent coefficients receive perfect data.

6.8.2 Verilog implementation of scenario 2

Figure 6-16 shows the timing diagram for Scenario 2 and it can be seen that the first coefficient of the filter is fed with MSB of the present sample and the LSB of the previous sample. The design structure for Scenario 2 is shown in Figure 6-17. The idea behind
Scenario 2 is to test the assumption that only a small change is expected in LSB of two successive samples hence the past sample LSB can better approximate the value of the present sample LSB data.

Figure 6-16: Input to the filter where the first coefficient is fed with previous LSB

Figure 6-17: FIR structure of (a) standard FIR filter and (b) structure used for scenario 2

This approach is similarly tested for different filter orders and at various frequency tones over the following sections.
6.8.2.1 Time and frequency domain analysis of Scenario 2 research filter compared with reference filter

Figure 6-18: Output of both Golden and Research 3-tap filters (scenario 2)

Figure 6-18 shows the output of the Low pass FIR filter of 3-taps when the input is fed with a 58 kHz sine wave. It can be seen that there is a loss of data between the reference filter and the research filter because the first coefficient is fed with MSB data appended with previous LSB data.
Figure 6-19 shows the FFT plot of the input and output of Reference and Research FIR 3-tap filters for a tone of 58 kHz and it can be seen that the harmonics and noise floor is high in Research filter of Scenario 2 when compared to Scenario 1. One of the possible reasons for this abrupt behaviour is due to the previous LSB data which is considered as noise by the filter. The next section illustrates the dynamic parameters for different tap filters in Scenario 2.

6.8.2.2 Comparison of SNR, SINAD, SFDR, THD for different input frequencies for reference and research filter

This section compares the dynamic parameters such as SNR, SINAD, SFDR etc., for 3,7,11,21-tap filters which is shown in Table 6-9, Table 6-10, Table 6-11, and Table 6-12 respectively. It can be seen that the SNR of the Research filter increases as the number of taps increase and is approximating the SNR of the Gold filter. The test frequencies for the filters are selected as per their pass band and stop band frequencies.

<table>
<thead>
<tr>
<th>F (kHz)</th>
<th>I/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A  SNR  SINAD  THD</td>
<td>Reference</td>
<td>Research</td>
</tr>
<tr>
<td></td>
<td>A  SNR  SINAD  THD</td>
<td>SFDR</td>
<td>SFDR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A  SNR  SINAD  THD</td>
<td>A  SNR  SINAD  THD</td>
</tr>
<tr>
<td>58</td>
<td>-6.0 55.9 55.9 -82.6</td>
<td>-6.4 54.3 54.3 77.3</td>
<td>-6.4 32.1 31.7 42.3</td>
</tr>
<tr>
<td>123</td>
<td>-6.0 55.9 55.9 -83.3</td>
<td>-7.9 52.9 52.8 67.5</td>
<td>-7.8 30.5 30.1 40.5</td>
</tr>
<tr>
<td>305</td>
<td>-6.0 55.9 55.9 -inf</td>
<td>-25.3 35.2 35.2 -inf</td>
<td>-24.8 13.9 13.9 -inf</td>
</tr>
</tbody>
</table>
Table 6-9: Comparison of 3-tap FIR filter performance – scenario 2

<table>
<thead>
<tr>
<th>F (kHz)</th>
<th>I/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Reference</td>
<td>Research</td>
</tr>
<tr>
<td>A</td>
<td>SNR</td>
<td>SINAD</td>
<td>THD</td>
</tr>
<tr>
<td>32</td>
<td>6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
<tr>
<td>52</td>
<td>6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
<tr>
<td>137</td>
<td>6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
</tbody>
</table>

Table 6-10: Comparison of 7-tap FIR filter performance – scenario 2

<table>
<thead>
<tr>
<th>F (kHz)</th>
<th>I/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Reference</td>
<td>Research</td>
</tr>
<tr>
<td>A</td>
<td>SNR</td>
<td>SINAD</td>
<td>THD</td>
</tr>
<tr>
<td>16</td>
<td>6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
<tr>
<td>32</td>
<td>6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
<tr>
<td>64</td>
<td>6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
</tbody>
</table>

Table 6-11: Comparison of 11-tap FIR filter performance - scenario 2

<table>
<thead>
<tr>
<th>F (kHz)</th>
<th>I/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Reference</td>
<td>Research</td>
</tr>
<tr>
<td>A</td>
<td>SNR</td>
<td>SINAD</td>
<td>THD</td>
</tr>
<tr>
<td>16</td>
<td>6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
<tr>
<td>25</td>
<td>6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
<tr>
<td>32</td>
<td>6.0</td>
<td>55.9</td>
<td>55.9</td>
</tr>
</tbody>
</table>

Table 6-12: Comparison of 21-tap FIR filter performance – scenario 2

The tables above show the comparison of same filter with different taps. As the number of taps are increased, the filter finds its roll off approximating its cut off frequency. The next section explains scenario 3 where the first coefficient of the filter is fed with present MSB and random 5-bit LSB data that is produced from a Linear Feedback Shift Registers (LFSR). Subsequent coefficients in the filter path receive perfect data.

6.8.3 Verilog implementation of scenario 3

Generally, the output of a standard LFSR has uniform noise floor in the frequency domain. To achieve this, the length of the LFSR must be sufficient enough to produce a large sequence of random data. A 32-bit LFSR is used in this design where the least 5 bits
are used as the LSB input for the first coefficient of the filter and is tested for the analysis of the dynamic parameters. The efficiency of the LFSR is defined by the repetition of each code and the LFSR is said to be efficient if all the codes or patterns are repeated at a uniform rate. This can be determined by plotting the histogram of the LFSR output which shows the count of each code or pattern. Figure 6-20 and Figure 6-21 shows the FFT and histogram of a 32-bit LFSR and it can be seen that noise floor is almost uniform and the histogram demonstrates that each code is repeated equally. Figure 6-22 and Figure 6-23 shows the structured realization of the FIR filter and timing diagram when the filter is fed with the 5-bit LFSR output. The next section deals with the dynamic performance of different tap filters when tested with this LFSR output.

![Frequency domain output of a 32 bit LFSR](image1)

**Figure 6-20 : LFSR FFT output**

![LFSR Histogram Data](image2)

**Figure 6-21 : LFSR Histogram Data**
Figure 6-21: Histogram of a 32-bit LFSR

Figure 6-22: FIR structure of (a)Reference FIR filter and (b) Research Filter used for scenario 3

Figure 6-23: Input to the filter where the first coefficient is fed LFSR output
6.8.3.1 Comparison of SNR, SINAD, SFDR, and THD for different input frequencies for Reference and Research filter (scenario 3)

This section compares the dynamic parameter performance such as SNR, SINAD, SFDR, THD for 3, 7, 11, 21-tap filters which is shown in Table 6-13, Table 6-14, Table 6-15, and Table 6-16 respectively. It can be seen that the SNR, SINAD, SFDR, THD of the Research filter increases as the number of taps increase and is approximating the SNR of the Gold filter. The test frequencies for the filters are selected as per their pass band and stop band frequencies.

It can be seen that the SNR, SINAD, SFDR, THD performance of 11-tap filter is better than 21-tap - this is because as the number of taps increase, the transition band reduces which means the filter is approximating the cut-off frequency of ideal filter. Now the test frequency 16kHz falls near the stop band of the 21-tap filter since the filter has small transition band when compared to 11-tap filter and hence a loss in SNR observed.
6.8.4 Scenario 4 – Oversampling

The change in LSB of successive samples of a discretized continuous time signal reduces as the sampling frequency is increased. In this section, the sampling rate of the signal and the filter is doubled and quadrupled. The aim of this method is to check whether the least change in LSB improves the dynamic parameters of the signal which are listed below.

<table>
<thead>
<tr>
<th>F (kHz)</th>
<th>I/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Reference</td>
<td>Research</td>
</tr>
<tr>
<td>32</td>
<td>-6.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>55.9</td>
<td>55.9</td>
<td>-83.5</td>
</tr>
</tbody>
</table>

Table 6-17: Comparison of 11-tap FIR filter performance – (2x sampling and zero LSB)

<table>
<thead>
<tr>
<th>F (kHz)</th>
<th>I/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Reference</td>
<td>Research</td>
</tr>
<tr>
<td>32</td>
<td>-6.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>55.9</td>
<td>55.9</td>
<td>-84.5</td>
</tr>
</tbody>
</table>

Table 6-18: Comparison of 11-tap FIR filter performance – (4x sampling and zero LSB)

<table>
<thead>
<tr>
<th>F (kHz)</th>
<th>I/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Reference</td>
<td>Research</td>
</tr>
<tr>
<td>32</td>
<td>-6.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>55.9</td>
<td>55.9</td>
<td>-83.5</td>
</tr>
</tbody>
</table>

Table 6-19: Comparison of 11-tap FIR filter performance – (2x sampling and previous LSB)

<table>
<thead>
<tr>
<th>F (kHz)</th>
<th>I/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
<th>O/P Amplitude (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Reference</td>
<td>Research</td>
</tr>
<tr>
<td>32</td>
<td>-6.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>55.9</td>
<td>55.9</td>
<td>-84.5</td>
</tr>
</tbody>
</table>

Table 6-20: Comparison of 11-tap FIR filter performance – (4x sampling and previous LSB)

These results show that oversampling doesn’t bring any appreciable performance improvement in SNR, SINAD, THD, SFDR performance to overcome the incorrect LSB data into the first FIR filter coefficient multiplication. A further solution that offers better potential is to use adaptive filtering [7] so as to train the filter coefficients to remove the unwanted noise from the incorrect LSB data operation. This is the subject of future work that is discussed in chapter 7.

6.9 Conclusion

A Verilog RTL test bench is developed to validate the functionality of FPGA test platform designed in Chapter 5. The RTL code uses a SPI master to mimic the action of an external SPI device controlling and accessing the system design so that FFT data from the ADC
is generated and captured from memory. FFT spectral analysis is used to validate the determined input and output signals of XADC and memory. An application of using FIR filters to try and improve the performance of an ADC design is also introduced. An example was provided where a SAR ADC converter produces MSB and LSB data at different sampling time instants. Typically, a SAR generates MSB data first before generating the LSB data word. The idea is to see if this feature can be used to generate valid SAR data output at an improved latency of ½ clock sampling cycle. This feature could be potentially useful when deploying ADCs in feedback loops for which latency is a critical concern. This objective was to check if the LSB data for the first coefficient operation can be replaced with imprecise data such as zero data or previous sample data. Dynamic performance results were recorded and analysed for various research filters scenarios. The aim was to develop a filter that could be used in real time for SAR ADC to post correct its output providing for ½ sample latency improvement. The results were tabulated and discussed in detail. Chapter 7 discusses the achievements and outputs of this thesis and also include a section that describes future work.
7 Conclusions and Future Work

7.1 Overview

In this thesis an investigation was performed into the study of fundamentals for digital design and digital signal processing through the development of a 7-tap FIR filter, in both MATLAB and Verilog. A study of FFTs was also carried out in the thesis and this knowledge is used to analyse dynamic performance of an ADC through the frequency spectrum analysis of sine waves. These parameters were used to analyse the input and output sine waves of the 7-tap filter. Using the knowledge of digital design and digital signal processing, a dynamic ADC test FPGA test platform was developed in Verilog on a Zynq board FPGA that was combined with an AMS-101 card.

The development of this test platform involved study and understanding of data sheets of different mixed signal devices on FPGA board, which includes digital synthesizers for generation of sine waves, the Xilinx ADC and DAC, and block memory components. Smart digital techniques, such as the careful generation of clock and reset signals to avoid potential hazards like metastability, were also covered.

Overall, the test platform was used for the generation of a 16-bit dynamic test signal (sine wave) from a DDS block, which was serially converted and externally input to a 16-bit Nano DAC. The analogue output of this DAC was then fed directly to 12-bit 1MSPS XADC input which was sampled at the same DAC update frequency. The XADC sampled the DAC waveform and the digital data is captured and stored in block memory. Using a SPI_Master, the block memory was accessed through the SPI interface and the captured samples are used to analyse the performance of XADC through computation of FFT, which tested the dynamic parameters showing the signal quality.
This project was developed by building a SPI interface that connected to the digital design platform that was coded in Verilog RTL and implemented onto a FPGA. A potential application was developed for the FPGA test platform where the outputs of ADC are applied to FIR filters to enable further analysis. Multiple FIR filters of 3, 7, 11 and 21-taps were developed in Verilog HDL to connect to the outputs of ADC. This application was developed in consideration of the case of a SAR ADC converter. Future work includes interfacing this FPGA board to a PC through a real time SPI interface where the PC acts as SPI _Master controlling the FPGA board through SPI interface. Once this is built, adaptive FIR filters can be developed to analyse the outputs of XADC and indeed the output of adaptive filter can also be viewed on the PC in real time.

7.2 Achievements

The following achievements gained during the development of this project include;

- Good knowledge in digital design principles, which includes the fundamentals of combinatorial and sequential logic design.
- Fundamental knowledge gained in digital signal processing aspects of FIR filter.
- Robust code was developed in MATLAB for the input/output analysis of the designs under test using FFT.
- Full System on Chip (SoC) design was developed using FPGA and Xilinx Intellectual Property (IP) cores such as DDS and XADC components.
- Good knowledge in design of test benches for the verification of the SoC for correct functionality.
- An application with different input scenarios using digital FIR filters was developed to analyse the output of the Xilinx Analog to Digital Converter (ADC).
- Comprehended the functionality of each component in the design accessing data sheets where necessary to ensure the design met the specified requirements.
- Practical aspects of digital system design using effective Verilog coding style and FPGA digital implementation was enabled with this project.
- Good writing and presentation skills were also developed during the study of this project.
- Knowledge in development of digital FIR filters for the further analysis of ADC outputs.
A study of an adaptive filter approach was also carried to post correct the ADC output which will be the focus of future work and now summarized in the following section.

7.3 Future work

7.3.1 Adaptive Filtering

A further solution that offers better potential is to use adaptive filtering [6, 7] so as to train the filter coefficients to remove the unwanted noise from the incorrect LSB data operation. Such a system is shown in Figure 7-1. Here a Reference Filter Transform $H_{ref}(Z)$ is used to provide a reference signal so that the error difference between the Reference and Research filter outputs produce an error. This error difference is minimised using a Least Mean Square (LMS) or Recursive Least-Squares (RLS) algorithm [7] so that the adaptive filter coefficients in $H_{adf}(Z)$ are trained to remove the unwanted noise from the imprecise LSB data appearing at the first coefficient location.

The overhead in this system is the addition of a Reference filter and the application of an adaptive algorithm such that filter coefficients can be modified. The filter coefficient must be programmable which also increases the design complexity. This solution approach was not implemented and tested, however it is noted that this adaptive filtering method has been shown to post-correct non-linear ADC errors [30].
7.3.2 Equalization-Based Calibration

Pelgrom [30] states that large integral and differential non-linearities (INL/INL) errors in data converters result from component errors due to capacitor mismatch, comparator offset, and switch-induced offset. In the case of SAR ADC converters, Liu and Chiu [31] give an example where these non-linearities can be corrected in the digital domain using an Least Mean Square FIR adaptive filter (LMS-ADF), which performs linear equalisation (LE) to align digital codes of the SAR-ADC to those of a slow but accurate reference ADC.

7.3.2.1 Proposed architecture

![Diagram of equalization-based adaptive background digital calibration architecture of SAR-ADC](image)

Figure 7-2: Equalization-based adaptive background digital calibration architecture of SAR-ADC [31]

The equalization-based SAR-ADC calibration scheme is shown in Figure 7-2 where the output bits from the SAR-ADC are fed into adaptive filter LMS-ADF, which employs least mean square approximation [6, 7]. Based on the error signal $e$, the filter coefficients are adjusted to produce accurate binary output code. This same algorithm can also be used to post correct the output of pipelined ADC which is explained by C. Yun, C. W. Tsang, B. Nikolic, and P. R. Gray in [32].

7.4 Conclusion

This chapter overviews the work developed as part of this thesis and lists the key outcomes and achievements delivered. An application of using FIR filters to try and improve the performance of an ADC design was introduced in Chapter 6. It was shown that such a filter is difficult to design. These approaches can be extended by designing an
adaptive FIR filter system as shown in Figure 7-1. In this case, the filter coefficients in
an adaptive filter could be tuned for error minimisation using a reference filter providing
the signal comparison. The disadvantage of this approach is the design overhead, but the
signal quality would be expected to improve. Another application of adaptive filters is
also introduced where the non-linearities of the SAR ADC due to capacitance mismatch
and offset errors are corrected by digital means. Here the filter uses the LMS algorithm
in the adaptive filter for the post correction of binary codes of SAR-ADC using the
reference of an accurate ADC.

[2] DSP based Analog and Mixed-signal Test - Slides Available:


Appendix 1: Verilog and MATLAB Implementation Style

A1.1 Introduction

Verilog HDL and MATLAB are the two coding languages used throughout this thesis. This chapter briefly explains the essential RTL coding practices and guidelines for good Verilog style and VLSI implementation aspects that were used during the project. A short MATLAB code example is also introduced to show how dynamic parameters such as SNR, SINAD are extracted using the Fast Fourier Transform. This code was used to analyse data that was written as file I/O from the Verilog RTL code.

A1.2 Revision control

The Verilog RTL code developed during this thesis was designed from scratch. When starting out on a design, revision control that keeps a record of all the changes made during the development should be used. A standard revision control using the RCS (revision control system) utility\(^1\) was implemented which keeps track of the changes made in the code. The sample revision control header used is shown in Figure A1-1 and is attached to the top of each module code.

```
//  ________________________________________________________________
//  Copyright (c) 2015 by CSRC
//  All Rights Reserved
//  University of Limerick CONFIDENTIAL
//  ________________________________________________________________
//  The information contained herein is proprietary of CSRC
//  ________________________________________________________________
//  File Name   : $RCSfile: SPI_Slave,v $ 
//  Author      : $name: Venkatesh Karra $ 
//  Revision    : $Revision: 1.1 $ 
//  Date        : $Date: 2015/18/11 16:27:32 $ 
//  Description : Dummy SPI slave module 
//                 : $Log: SPI_Slave,v $ 
//                 : Revision 1.1 2015/18/11 16:27:32 karrav 
//                 : Initial revision 
//                 : Revision 1.2 2015/20/11 16:27:32 karrav 
//                 : Fix bug in output signal generation 
//  ________________________________________________________________
```

Figure A1-1: Sample revision control header

---

\(^1\) https://www.gnu.org/software/rcs/
The RCS revision control header keeps a log of all the changes made in the Verilog code. Each time code is checked out, edited and checked in to RCS, a new revision for the code is generated. The $Log maintains a description for each modification that is checked into RCS. Simple commands to checkin code –‘ci –u SPI_slave.v’ and to checkout code ‘co –l SPI_slave.v’ are used. The notation ‘–u’ and ‘–l’ - unlock and lock the code so that code can only be edited when it is checked out.

### A1.3 Verilog HDL and standard indentation style

Verilog HDL is a VLSI hardware description language based on the IEEE standard 1364 [33] used to model electronic systems. The main reason for choosing Verilog HDL over VHDL is for its similarity to C coding style and that it is easier for designers to learn and implement. Basic RTL code is shown below where it can be seen that the code is well spaced and indented for readability. Also, important functionality within the code should be well commented so that a new user can understand clearly. A timescale directive is inserted at the top of each Verilog module such that the simulator understands the time units. Timescale specifies the time unit and time precision of a module that follows it. The simulation time and delay values are measured using time unit. The precision factor is needed to measure the degree of accuracy of the time unit, in other words how delay values are rounded before being used in simulation. This code example in Figure A1-2 shows how input and outputs signals are declared and passed though the hierarchy. Simple descriptive module names such as ‘DDS2’ should be used and standard parameter notation using pOFFSET as an example is maintained throughout.

```verilog
// Overview: Direct Digital Synthesis Adjust
// This module designs logic that converts signed Sinewave to unsigned logic.
// timescale 1ns / 1ps

module DDSA
(input CRCTL_DCLKp, // DAC sampling clock
     input wire CRCTL_SynDRSTn, // Synchronous reset signal from CRCTL to DDS2
     input signed [15:0] DDS1_Sine, // Signed sine waveform from DDS1
     // Outputs
     output reg [15:0] DDS2_USine //Unsigned sine wave to SPI
);
```

// Parameters

parameter pOFFSET = 16'd32768;

// Main Code

//Converting 2s complement to unsigned representation
//Reducing the voltage from 2.5V(DAC_ref) to 0.9V(90% of XADC_in) using
//shift and add technique

always @(posedge CRCTL_DCLKp or negedge CRCTL_SynDRSTn)
begin
    if(!CRCTL_SynDRSTn)
        DDS2_USine <= 0;
    else
        begin
            DDS2_USine <= ((DDS1_Sine + pOFFSET)>>1) - ((DDS1_Sine +
            pOFFSET)>>3) - ((DDS1_Sine + pOFFSET)>>4);
        end
end
endmodule

Figure A1-2: Sample Verilog code showing standard indentation and commenting style used throughout this thesis

A1.4 Standard naming conventions

A standard method of signal naming convention is followed throughout the thesis which is shown in Figure A1-3. The input signal name starts with the name of the block it is coming from followed by name of the signal itself. This makes the code very easy to understand. Input1 and Input2 are signal sources from A Block and the output signals Output1 and Output2 are sent out from B Block. This convention makes the design more structured and keeping this signal name consistent all the way through the hierarchy makes the code easier to follow. Another standard naming convention is used for wire and registers where each register name is followed by “_r” and the name of the wire is followed by “_w”. All the Verilog codes used in this thesis practice this same methodology as can be seen in Appendix 2. This is an in-house coding style used in this thesis.

Figure A1-3: Standard Verilog signal naming convention used in the thesis
A1.4.1 Blocking and non-blocking assignments

Verilog HDL supports two types of assignments within always blocks that feature two different behaviours.

A1.4.1.1 Blocking assignments

In this type of assignment each and every statement within the always block is evaluated and assigned immediately.

```verilog
always @ (a or b or c)
begin
    x = a | b;  //Evaluate a | b, assign result to x
    y = a ^ b ^ c;  //Evaluate a ^ b ^ c, assign result to y
    z = b & ~c;  //Evaluate b & (~c), assign result to z
end
```

Figure A1-4: Sample code of blocking assignments

This style is used for generating combinatorial logic.

A1.4.1.2 Non-blocking assignments

In the case of non-blocking assignment, each and every statement within the always block is deferred until all the statements are evaluated.

```verilog
always @ (posedge Clk)
begin
    x <= a | b;  // Evaluate a | b but defer assignment of x
    y <= a ^ b ^ c;  // Evaluate a ^ b ^ c but defer assignment of y
    z <= b & ~c;  // Evaluate b & (~c) but defer assignment of z
end
```

Figure A1-5: Sample code of non-blocking assignments

Non-blocking assignment is used for the generation of synchronous logic (registers/flip-flops) where every statement should be executed either at the rising edge or falling edge of clock signal.

A1.5 Synchronous design with asynchronous reset

The sensitivity list in the always block controls the execution of each statement in it. When it comes to synchronous design which is made of flip-flops and combinational
logic, the execution depends on the rising or following edge of clock as shown in Figure A1-6

```verilog
always @ (posedge CLK)
begin
    // The XADC_r is controlled by RST only when posedge of clock happens
    if (!RSTn)
        XADC_r <= 1'b0;
    // The XADC_r is controlled by sel only when posedge of clock happens
    elsif sel = 1
        XADC_r <= ADC_in;
    else
        XADC_r <= Test_in;
end
```

Figure A1-6: Sample code for synchronous reset

In the above example, the sensitivity list has a clock signal, which controls the procedure. This design supports synchronous reset behaviour where if the reset signal goes high, the XADC_r is not set to zero until the rising edge of the clock occurs. This type of reset behaviour causes undefined don’t cares during simulation which can ripple through a design. This is solved by coding all synchronous logic with asynchronous reset behaviour as shown in Figure A1-7

```verilog
always @ (posedge CLK or negedge RSTn)
begin
    if (!RSTn)
        XADC_r <= 1'b0;
    elsif sel = 1
        XADC_r <= ADC_in;
    else
        XADC_r <= Test_in;
end
```

Figure A1-7: Sample code for asynchronous reset

Here the reset signal – RSTn is asynchronously de-asserted and is not dependent on the clock signal. This style is recommended to ensure undefined signals (don’t cares) ‘X’’s do not appear in simulation.

**A1.6 Parameterize designs**

Use parameters where appropriate as this makes the RTL code portable and helps to upgrade existing code. In the example below a local “params.v” is used to contain all the
parameters within the design and is called with the syntax `include “params.v”`. All the parameters are contained within this “params.v” file as shown in Figure A1-8.

```vhdl
module DDSA(
  // Inputs
  input  CRCTL_DCLKp, // DAC sampling clock
  input wire CRCTL_SynDRSTn, // Synchronous reset signal from CRCTL to DDS2
  input signed [pWIDTH-1:0] DDS1_Sine, // Signed sine waveform from DDS1
  //Outputs
  output reg [pWIDTH-1:0] DDS2_USine //Unsigned sine wave to SPI
);

'include "params.v"

parameter pWIDTH = 16

endmodule
```

Figure A1-8: Sample code showing the usage of params.v

### A1.7 Avoid latch interference

Latch generation is a very subtle point that has the potential to cause frustration while working with digital design in FPGA or ASICs. Consider the program below Figure A1-9

```vhdl
//wire Trigger, Pass;
reg A, C;
always @ (*) begin
  A = 1'b0; // Assigning A with default value this avoids latches
  if ( Trigger ) begin
    A = Pass;
    C = Pass; // Since C has no default value a latch is formed
  end
end
```

Figure A1-9: Sample code illustrating the latch formation
Here A is always assigned. This is because the first line specifies the default value of A. However what happens to C if trigger is not valid; Here C does not have any value which infers a latch for output C. This is shown in Figure A1-10. This problem is resolved by ensuring a default case for every variable.

![Figure A1-10: Latch formation due to incomplete assignment](image)

The program code below resolves the issue of latch formation

```verilog
code
wire Trigger, Pass;
reg A, C;

always @ (*) begin
    A = 1'b0; // Assigning A with default value this avoids latches
    C = 1'b1; // Assigning C with default value this avoids latches
    if (Trigger) begin
        A = Pass;
        C = Pass;
    end
end
```

![Figure A1-11: Sample code to resolve the problem of latch formation](image)

Here both variables are assigned a default value this prevents the problem of latch inference.

**A1.8 State machine design**

A Finite State Machine(FSM) controller is designed in this thesis that acts as a control unit for the testing platform. The state machine is based on Moore style where the outputs only depend upon the present state as illustrated in Figure A1-12
Example code for the structure of state machine used in this thesis is shown Figure A1-13. The code also contains state register logic followed by next state logic. The states are also syntactically defined using ‘stXXX’ notation. In this state machine, the output logic is also registered and so the outputs are guaranteed to be glitch free.

```verbatim
// State Parameters
//
parameter stIDLE = 2'b00,
    stDAC_A = 2'b01;

// State register logic or Synchronous state transitions
always @(posedge CRCTL_SysCLKp or negedge CRCTL_SynRSTn)
    begin
    if(!CRCTL_SynRSTn)
        rCurrentState <= stIDLE;
    else
        rCurrentState <= rNextState;
    end

// rNextState Logic or Conditional State-Transition always(*) block
always @(*)
    begin
        SCU_DacUpdate = 0;
        SCU_DacSel = 0;
        rNextState = rCurrentState;
        rWaitCntRst = 0;
        rDEN = 0;
        rCONVST = 0;
        rSample = 0;
        case (rCurrentState)
            stIDLE: begin
                if(CRCTL_SynRSTn == 1)
                    begin
                        if(MasterCntrl == 1)
                            rNextState = stDAC_A;
                    end
            end
            ...  // Other cases
    end
```
else
    rNextState = stIDLE;
end

stDAC_A: begin
    rNextState = stDAC_A_UPDATE;
end

...default: rNextState = stIDLE;
endcase
end

// Registering the output logic
always @(posedge CRCTL_SysCLKp or negedge CRCTL_SynRSTn)
begin
    if(!CRCTL_SynRSTn)
        begin
            XadcData <= 0;
        end
    else
        begin
            SCU_CONVST <= rCONVST;
            if (rCurrentState == stXADCrd)
                XadcData <= XADC_DO[15:4];
        end
end

Figure A1-13: Example code for the structure of state machine used in this thesis

A1.9 Development of test bench RTL

Another important skill mastered during this thesis is the development of test bench RTL code that feeds the system or unit under test with the required test vectors and writes the output to a text file, which is then accessed through MATLAB for analysis. An aspect for the test bench is instantiation of the module or unit under test. The module instantiation style is shown in Figure A1-14. More details of the test bench code style in Verilog RTL is further explained in Chapter 6. The next section introduces about MATLAB implementation style used in this thesis.
// Instantiate SPI MASTER

SPI_bfm_12 u_SPI_bfm_12 (  
  .CLKp      (CLKp),  
  .MISO      (MISO),  
  .MOSI      (MOSI),  
  .SCLK      (SCLK),  
  .SSn       (SSn)  
);  

Figure A1-14: Sample code showing instantiation of Design Under Test (DUT)

**A1.10 MATLAB implementation style**

MATLAB [34] is a high performance language for technical computing, it integrates computation, visualization and programming in an easy to use environment where code is expressed in familiar mathematical notation. MATLAB code scripting is used in this thesis for the design of FIR filters and for the spectral analysis of signals using FFT, where the generation, quantization and computation of FFT for sine waves can be based on both coherent and non-coherent signals. The MATLAB code for this thesis is contained in Appendix 3. The next section explains the combination of MATLAB and Verilog HDL for file I/O.

**A1.10.1 Filter design using MATLAB**

The digital FIR filter design is one of the objectives of this thesis. MATLAB is used to design this FIR filter for required number of taps and to analyse its magnitude and phase plots. The MATLAB code for the design and analysis of FIR filter is shown in Figure A1-15 and the full discussion is given in Chapter 3 Sections 3.7,3.8,3.9.

```matlab
%*********************************************************************
% Defining initial requirements
%*********************************************************************
fc = 25e3;  % cut off frequency
fs = 10e6;  % sampling frequency
Ts = 1/fs;  % Sampling time period
M = 3;      % Since total length of filter L = 2*M+1
L = 2*M+1;  % total number of taps in the filter
oc = 2*pi*fc*Ts;  % Calculation of normalized cut off frequency
h0 = oc/pi;  % For the coefficient n = 0;
%*********************************************************************
%For coefficients n = 1,2,..., and n = -1,-2,...,
%*********************************************************************
for n = 1:M    
    h(n) = sin(oc*n)/(n*pi);  
end  
%*********************************************************************
%Arranging coefficients in standard form as
```
[h(-n-1),..., h(-1), h(0), h(1),...h(n-1)]
%*********************************************************************
coefficients = horzcat(fliplr(h),h0,h)
%*********************************************************************
%Calculating unity gain filter coefficients which means the sum of coefficients is unity.
%*********************************************************************
coefficients unity =(coefficients)/sum(coefficients)
%***********************************************************
%Filter visualisation tool
%*********************************************************************
fvtool(coefficients_unity,1)

Figure A1-15: MATLAB code for the design and analysis of FIR filter

Here the function fvtool (denominator, numerator) launches a filter visualisation tool which allows the user to analyse the magnitude and phase plots.

A1.10.2 File input and output (I/O) using both Verilog and MATLAB

File I/O is another extensive feature used in this thesis so that test signals generated from the digital design can be externally analysed for dynamic performance. The verification format has got one input sine wave and two output sine waves. One output is from the XADC which is a 12-bit digital sine wave and the second one is the data collected from the block memory. These signals are generated on the Xilinx Verilog platform and are tested using FFTs in the MATLAB platform. The sine waves generated from the Xilinx Verilog platform are exported to a text file using the following format as shown in Figure A1-16

.Alpha text file format is imported into MATLAB as shown in Figure A1-17

This test file is imported into MATLAB as shown in Figure A1-17
MATLAB reads this variable as a vector and is then used for the FFT analysis.

**A1.10.3 FFT using MATLAB**

The FFT is a mathematical tool used in this thesis for the spectral analysis of the test signals. MATLAB is used as the platform to perform and plot the FFT of input digital signals as shown in Figure A1-18. Initially the digital signal is imported from text file and an FFT is performed on this signal. Once the FFT is calculated, then this data is used to compute dynamic test parameters which specify the dynamic behaviour of ADC and finally all the results are plotted.

![Spectral analysis of digital signals using MATLAB](image)

In this thesis, the dynamic performance is calculated in two ways

- The first method uses the in-built MATLAB subroutines listed in table below.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>snr(x)</td>
<td>Computes the signal to noise ratio on input signal x.</td>
</tr>
<tr>
<td>sfdr(x)</td>
<td>Computes the spurious free dynamic range on input signal x.</td>
</tr>
<tr>
<td>thd(x)</td>
<td>Computes the total harmonic distortion on input signal x.</td>
</tr>
<tr>
<td>sinad(x)</td>
<td>Computes the signal to noise and distortion on input signal x.</td>
</tr>
</tbody>
</table>

Table A1-1: Inbuilt MATLAB functions for the computation of dynamic test parameters of ADC
The second method uses a user defined subroutine. The MATLAB code shown in Figure A1-19.

```matlab
function [SNR,SINAD,SFDR]= Dynamic_parameters( input_sig, N_fft, fs)
    s       = input_sig;  %Assigning input signal to variable s
    s(1:5)  = 1e-100;    %Neutralizing the dc component

%*********************************************************************
%x and y represents the position & amplitude fundamental tone
%respectively
%*********************************************************************
[y,x]   = max(s)    %Calculating maximum of the input signal

%*********************************************************************
N_fft   = 8192       %Number of FFT points
fs      = 9.6154e+04; %Sampling frequency
fs_norm = 1;         %Normalized frequency
fin     = ((x)/N_fft)*fs; %Input frequency
fh      = (2:8) * fin; %First six harmonics

%*********************************************************************
%Finding the harmonic frequencies
%*******************************************************************
while max(fh) > 0.5
    fh = abs(fh -(fh>0.5));
end

%*********************************************************************
%harmonic bin locations
%*********************************************************************
bh          = round ( N_fft * fh / fs_norm + 1);

%*********************************************************************
Asignal     = s(x);   %fundamental frequency
Aharm       = sqrt(sum(s(bh).^2));    %rms value of harmonics
sn          = s;

sn(x-3:x+3) = 1e-100;   %Removing signal spreading
SFDR        = 20*log10(Asignal/max(sn)); %Spurious free dynamic range

%*********************************************************************
%Neutralizing the harmonic bin locations
%*********************************************************************

sn(bh)      = 1e-100;

%*********************************************************************
Anoise      = sqrt(sum(sn.^2));    %Noise in the signal
SNR         = 20*log10(Asignal/Anoise) %Signal to noise ratio

%*********************************************************************
%Signal to noise and distortion
%*********************************************************************
SINAD       = 20*log10(Asignal/(Anoise+Aharm))
end
```

Figure A1-19: MATLAB code for function that calculates dynamic parameters of ADC.

Prior to calling the subroutine ‘Dynamic_parameters’, the FFT is calculated using the ‘abs(fft(x))’ command which generates the absolute value of the discrete Fourier transform of vector x. The output of the FFT is used as input signal (input_sig) for this subroutine and includes the length of the signal (N_fft) and sampling frequency (fs)
information. The subroutine produces the outputs, that are the dynamic parameters SFDR, SNR and SINAD.

A1.11 Conclusion

This Appendix briefly summarizes the key Verilog RTL coding principles used for the development of this thesis work. The important areas of revision control, signal Verilog coding style, naming conventions, file I/O and state machine design formats were discussed. Aspects of the MATLAB coding style and dynamic parameter estimation were also introduced.
Appendix 2: Q-format method for binary representation of decimal number

In real time systems, Digital Signal Processors (DSP) typically uses fixed point arithmetic data processing. The Q format method converts a decimal number having both integer and fraction parts into a binary number using the 2’s complement method.

A2.1 General format for Q-point representation

Representation of numbers in Q format is the most common method for the fixed-point DSP implementation for numbers from -1 to +1. This particular format is demonstrated in Figure A2-1.

As indicated in Error! Reference source not found., Q-15 means the data to be processed is in sign magnitude form where there are 15 bits for magnitude and one bit for the sign. After the sign bit, the dot represents the implied binary point.

A2.1.1 Observations

- The range is divided into $2^{16}$ intervals each with a size of $2^{-15}$.
- Hence the number is normalized between [-1, 1] and any multiplication result in a fraction of same range.
- The minimum value is -1 and the maximum value is 1-$2^{-15}$.

A2.2 Mathematical explanation of Q-15 representation of a signed number

A2.2.1 Example for computing the signed Q point representation of a decimal number

Decimal number D = +0.1871.
**Step 1**: Multiply the number with $2^N$ where $N = \text{number of bits for magnitude representation}$. Here $N=15$.

**Step 2**: $D = 0.1871 \times 2^{15} = 6130.9$. Here we need to truncate the value resulting in a loss of data which is a drawback for this method.

**Step 3**: Now convert $D$ into binary form using 2’s complement method.

$B = 0001_0111\_1111\_0010$. The sign bit is 0 and hence indicates a positive number.

**A2.2.2 Computation of Negative signed decimal -0.1820 using Q point method.**

Decimal number $D = -0.160123$.

**Step 1**: Multiply the number with $2^N$ where $N = \text{number of bits for magnitude representation}$. Here $N = 15$.

**Step 2**: $D = 0.160123 \times 2^{15} = -5246.9$. On truncation gives $D = -5246$.

**Step 3**: Now convert $D$ into binary form using 2’s complement method hence

$B = 1110\_1011\_1000\_0010$. The sign bit is 1 indicating a negative number.

**A2.3 Generalized method for representing the fixed-point data in binary form:**

In general, the coefficients can be greater than 1 which consists of both integer and fraction. In this case, the format depends on the precision required and based on the number of bits that are assigned to the fractional and integer parts.

The generalized form is shown in Figure A2-2.

![Figure A2-2](image-url)

Figure A2-2: General format or representation of binary number in fixed point
# Appendix 3: Verilog RTL Code List

This appendix includes the Verilog RTL code modules used throughout this thesis and referenced in the chapters. The following table lists the Verilog code contained on the DVD.

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A3.1</td>
<td>FIR_7TAP_MUL.v</td>
<td>7-tap FIR Filter using multiplication</td>
</tr>
<tr>
<td>A3.2</td>
<td>CRCTL.v</td>
<td>Clock Reset Controller module</td>
</tr>
<tr>
<td>A3.3</td>
<td>DDS2.v</td>
<td>Direct Digital Synthesis (sinewave generator)</td>
</tr>
<tr>
<td>A3.4</td>
<td>SCU.v</td>
<td>System Control Unit module</td>
</tr>
<tr>
<td>A3.5</td>
<td>ARB.v</td>
<td>Arbitration unit module</td>
</tr>
<tr>
<td>A3.6</td>
<td>PCR.v</td>
<td>Program Control Register module</td>
</tr>
<tr>
<td>A3.7</td>
<td>SPIS.v</td>
<td>SPI Slave module</td>
</tr>
<tr>
<td>A3.8</td>
<td>XADC.v</td>
<td>Xilinx Analog to Digital Converter wrapper module</td>
</tr>
<tr>
<td>A3.9</td>
<td>SPI_DAC.v</td>
<td>SPI DAC interface for AMS101 DAC parts</td>
</tr>
<tr>
<td>A3.10</td>
<td>SPI_MASTER.v</td>
<td>Behavioural SPI Master for test bench verification</td>
</tr>
<tr>
<td>A3.11</td>
<td>Test_Sys_top.v</td>
<td>Top-level system test bench</td>
</tr>
<tr>
<td>A3.12</td>
<td>SPI_dac_model.v</td>
<td>SPI DAC model</td>
</tr>
<tr>
<td>A3.13</td>
<td>Fir_lpf_3tap_g_1.v</td>
<td>3-tap Golden FIR filter</td>
</tr>
<tr>
<td>A3.14</td>
<td>Fir_lpf_3tap_r_1.v</td>
<td>3-tap Research FIR filter</td>
</tr>
<tr>
<td>A3.15</td>
<td>Fir_lpf_11tap_r_sc2_1.v</td>
<td>11-tap Research scenario 2 FIR filter</td>
</tr>
<tr>
<td>A3.16</td>
<td>Fir_lpf_21tap_lfsr_1.v</td>
<td>21-tap Research FIR filter</td>
</tr>
<tr>
<td>A3.17</td>
<td>Fir_lpf_7tap_g_1.v</td>
<td>7 tap Research FIR filter</td>
</tr>
<tr>
<td>A3.18</td>
<td>Fir_7tap_lfsr_1.v</td>
<td>7-tap FIR filter with LFSR</td>
</tr>
<tr>
<td>A3.19</td>
<td>SYS_top.v</td>
<td>Top level module of test platform</td>
</tr>
<tr>
<td>A3.20</td>
<td>Fir_lpf_7tap_r_1.v</td>
<td>7-tap Research FIR filter</td>
</tr>
<tr>
<td>A3.21</td>
<td>Fir_lpf_11tap_r_1.v</td>
<td>11-tap Research filter</td>
</tr>
<tr>
<td>A3.22</td>
<td>Fir_lpf_11tap_g_1.v</td>
<td>11-tap Gold filter</td>
</tr>
<tr>
<td>A3.23</td>
<td>Fir_lpf_21tap_r_sc2_1.v</td>
<td>21-tap Research filter scenario 2</td>
</tr>
<tr>
<td>A3.24</td>
<td>Fir_lpf_21tap_g_1.v</td>
<td>21-tap Gold filter</td>
</tr>
<tr>
<td>A3.25</td>
<td>Fir_lpf_21tap_r_1.v</td>
<td>21-tap Research filter</td>
</tr>
<tr>
<td>A3.26</td>
<td>Fir_lpf_11tap_lfsr_1.v</td>
<td>11-tap filter with LFSR</td>
</tr>
<tr>
<td>A3.27</td>
<td>Fir_lpf_3tap_lfsr_1.v</td>
<td>3-tap filter with LFSR</td>
</tr>
<tr>
<td>A3.28</td>
<td>FIR_7TAP_SA.v</td>
<td>7-tap FIR Filter implementation using shift and add</td>
</tr>
</tbody>
</table>
Appendix 4: MATLAB Code List

This appendix includes the MATLAB codes used in this thesis. The table listing the code below is included on the DVD.

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A4.1</td>
<td>FIR_n_tap_filter.m</td>
<td>7-tap FIR filter MATLAB code</td>
</tr>
<tr>
<td>A4.2</td>
<td>SineFFT_ntap.m</td>
<td>FFT analysis code for 7-tap FIR filter</td>
</tr>
<tr>
<td>A4.3</td>
<td>Sine_DAC_ADC_DDS.m</td>
<td>FFT analysis of XADC and DDS extracting dynamic parameters</td>
</tr>
<tr>
<td>A4.4</td>
<td>Dynamic_parameters.m</td>
<td>Subroutine for the calculation of dynamic signal parameters</td>
</tr>
</tbody>
</table>
Appendix 5: Sample XADC Input waveform format

This appendix material shows the sample input waveform format that is used to apply an input waveform to the Xilinx XADC so as to validate the full design in Verilog simulation. In this case, a DAC sinewave frequency of 96.153 KHz is applied to the positive terminal VP of the XADC, while a steady state 0.400009V is applied to the negative VN terminal input. This sets up the XADC in bi-polar mode as was discussed in chapter 5 - Section 5.9.2

//DACFs = 96153.846154 Hz
//FFT_POINTS = 8192

<table>
<thead>
<tr>
<th>TIME</th>
<th>VP</th>
<th>VN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000</td>
<td>0.000000</td>
<td>0.400009</td>
</tr>
<tr>
<td>0001865</td>
<td>0.000000</td>
<td>0.400009</td>
</tr>
<tr>
<td>0032985</td>
<td>0.449219</td>
<td>0.400009</td>
</tr>
<tr>
<td>0043385</td>
<td>0.449219</td>
<td>0.400009</td>
</tr>
<tr>
<td>0053785</td>
<td>0.449219</td>
<td>0.400009</td>
</tr>
<tr>
<td>0064185</td>
<td>0.449219</td>
<td>0.400009</td>
</tr>
<tr>
<td>0074585</td>
<td>0.449219</td>
<td>0.400009</td>
</tr>
<tr>
<td>0084985</td>
<td>0.449257</td>
<td>0.400009</td>
</tr>
<tr>
<td>0095385</td>
<td>0.898476</td>
<td>0.400009</td>
</tr>
<tr>
<td>0105785</td>
<td>0.892449</td>
<td>0.400009</td>
</tr>
<tr>
<td>0116185</td>
<td>0.874711</td>
<td>0.400009</td>
</tr>
<tr>
<td>0126585</td>
<td>0.845605</td>
<td>0.400009</td>
</tr>
<tr>
<td>0136985</td>
<td>0.806008</td>
<td>0.400009</td>
</tr>
<tr>
<td>0147385</td>
<td>0.756875</td>
<td>0.400009</td>
</tr>
<tr>
<td>0157785</td>
<td>0.699578</td>
<td>0.400009</td>
</tr>
<tr>
<td>0168185</td>
<td>0.635605</td>
<td>0.400009</td>
</tr>
<tr>
<td>0178585</td>
<td>0.566712</td>
<td>0.400009</td>
</tr>
<tr>
<td>0188985</td>
<td>0.494652</td>
<td>0.400009</td>
</tr>
<tr>
<td>0199385</td>
<td>0.421410</td>
<td>0.400009</td>
</tr>
<tr>
<td>0209785</td>
<td>0.348931</td>
<td>0.400009</td>
</tr>
<tr>
<td>0220185</td>
<td>0.279045</td>
<td>0.400009</td>
</tr>
<tr>
<td>0220185</td>
<td>0.279045</td>
<td>0.400009</td>
</tr>
<tr>
<td>0230585</td>
<td>0.213776</td>
<td>0.400009</td>
</tr>
<tr>
<td>0240985</td>
<td>0.154724</td>
<td>0.400009</td>
</tr>
</tbody>
</table>