2008

FPGA Design of a Hardware Efficient Pipelined FFT Processor

Ryan T. Bone
Wright State University

FPGA DESIGN OF A HARDWARE EFFICIENT PIPELINED FFT PROCESSOR

A thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Engineering

By

RYAN THOMAS BONE

Bachelor of Science in Electrical Engineering

Wright State University, 2005

2008

Wright State University

WRIGHT STATE UNIVERSITY

SCHOOL OF GRADUATE STUDIES

September 19, 2008

I HERBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Ryan Thomas Bone ENTITLED FPGA Design of a Hardware Efficient Pipelined FFT Processor BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Engineering.

Chien-In Henry Chen, Ph.D.
Thesis Director

Kefu Xue, Ph.D.
Department Chair

Committee on Final Examination

Chien-In Henry Chen, Ph.D.

John M. Emmert, Ph.D.

Raymond E. Siferd, Ph.D.

Joseph F. Thomas, Jr., Ph.D.
Dean, School of Graduate Studies

Abstract

Bone, Ryan Thomas. M.S.Egr., Department of Electrical Engineering, Wright State University, 2008. *FPGA Design of a Hardware Efficient Pipelined FFT Processor.*

Digital receivers involve fast Fourier transform (FFT) computations that require a large amount of arithmetic operations. The implementation of a FFT processor is one of the most challenging parts in the realization of a wideband receiver and its hardware complexity is very high. Hence, kernel function FFT processors have been proposed to meet real-time processing requirements and to reduce hardware complexity by rounding the kernel function to predetermined kernel points so as to eliminate the multipliers and use only shifters and adders or subtractors. Because of the nonlinear nature of this approximation by the rounding errors, spurious responses are generated and reduce the two signal instantaneous dynamic range (IDR) of the receiver in comparison with ideal FFT. Furthermore, there is a need to increase the resolution bits of the analog-to-digital converter (ADC) for FFT to improve the receiver performance by reducing the false alarm and increasing the spur-free dynamic range (SFDR).

In this research, architecture for an FPGA-based 2.56 giga sample per second (GSPS) fixed kernel function FFT, using a truncated 10-bit ADC, is implemented. The FFT can produce an averaged single signal SFDR using the ideal ADC, of 22.8 dB with the ability to produce a two-signal IDR using the ideal ADC with a performance of 20.8 dB. With the ADC utilizing the eight most significant bit (MSB) values, the FPGA-based

FFT can detect a weak input signal at -17.6 dBm at a full scale amplitude of 3.6 dBm. The resulting spurious-free dynamic range (SFDR) has a performance of 21.2 dB, which is very close to the ideal realization. The eight least significant bit (LSB) values where evaluated as well, generating a low signal detection of -22.7 dBm for a full scale amplitude of -9.3 dBm. This truncation scheme resulted in an SFDR performance of 13.4 dB. There was also a reduction in the hardware utilization with the FPGA implementation. With the employment of a folding technique the available resources where reduces by over 50% in comparison with the unfolded models.

Table of Contents

List of Figures

Figure Page

List of Tables

List of Abbreviations

AOA	Angle of Arrival
ADC	Analog-to-Digital Converter
ADS	Acoustic Detection Systems
DFT	Discrete Fourier Transform
DIF	Decimation-in-Frequency
DIT	Decimation-in-Time
DSP	Digital Signal Processing
EM	Electromagnetic
ENOB	Effective Number of Bits
EP	Electronic Self-protect
EPM	Electronic Protective Measures
ES	Electrical Support
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
GSPS	Giga Sample per Second
HDL	Hardware Description Language
IDR	Instantaneous Dynamic Range
IF	Intermediate Frequency
IP	Intellectual Property
LSB	Least Significant Bit
LUT	Look-up Table
MSPS	Mega Sample per Second
PA	Pulse Amplitude
PDW	Pulse Descriptor Word
PLD	Programmable Logic Devices
PMC	PCI Mezzanine Card
PW	Pulse Width
RF	Radio Frequency
RTL	Register Transfer Language
SFDR	Spur-Free Dynamic Range
SNR	Signal-to-Noise Ratio
TOA	Time of Arrival
VHDL	Very-High-Speed Integrated Circuit Hardware Description Language
XSG	Xilinx System Generator

Acknowledgements

This thesis was conducted in conjunction with the NEWSTARS program, Wright-Patterson Air Force Base Research Laboratory, Dayton, Ohio.

The completion of this thesis could not have been possible without the encouragement of many. First and foremost I would like to thank my family and friends for their unwavering faith and support. Without their inspiration I would have been lost throughout this whole process. It must also be expressed that without the guidance of my advisor, Dr. Henry Chen, this would have never been possible. His constant encouragement and patience has been invaluable throughout this research. I must also express a special thanks to George Lee and Vivek Sarathy. Their feedback and proficient knowledge of the subject matter provided the direction I needed to accomplish this task. Finally, I would like to recognize and thank Dr. Marty Emmert and Dr. Raymond Siferd for their willingness to serve on my thesis committee.

Ryan Bone, MSEE 2008

This thesis research is dedicated to my parents, David and Rita, and my two brothers, Justin and Travis.

I. INTRODUCTION

1.1 Wideband Receiver Background

The wideband receiver is a crucial component in modern receiver systems. The receiver has been primarily used to expose and distinguish adverse radar signals, but can also be used to observe large bandwidths (on the order of 1 GHz or more) of the radio frequency (RF) spectrum to identify signals of interest. This basically means that any signal that is within the intended bandwidth will be detected without argument. For the wideband digital receiver, the spectrum estimator that is used for the identification of these signals is a highly important design element imperative to the fulfillment of mission requirements. As a result of the computational complexity of the spectrum estimator, or fast Fourier transform (FFT), this research exists.

The main architecture that will be explored specifically in this research is the radar receiver, which has the capability to manipulate a wide bandwidth of pulsed radar signals. It must also be expressed that the basic hardware components that comprise the wideband receivers are very similar to that of the narrowband communication receivers. With this in mind, it is important to distinguish the differences between these two systems.

1.1.1 Relative to Narrowband Receivers

The wideband and the narrowband communication receivers are two important types of systems employed to detect radar and communication signals. Due to the advancement in digital signal processing techniques, the basic components of these types of receivers are very similar. Despite this fact, there are two important aspects that distinguish one from the other besides the wide and narrow instantaneous bandwidth. In the design of a communication receiver, the frequency, types of modulation, and bandwidth of an incoming signal are all known. Thus, the input signal can be considered as a compliant type and the receiver can be designed very efficiently [7]. As for a radar receiver the transmitting signal may be specifically designed to avoid detection, which would cause the data from the input signal to be unknown [7].

The output pulse descriptor words (PDW) of the wideband receiver is another major difference between the narrowband and wideband receivers. The PDW is a set of digitized signal parameters that are taken from an individual radar pulse. The parameters may include the carrier frequency (also referred to as the RF), incident direction, pulse width (PW), pulse amplitude (PA), and time of arrival (TOA) on each recognized pulse [7]. Narrowband communication receivers, on the other hand, recoup information emitted by a transmitter and don't perform parameter encoding. In addition, a vast majority of narrowband receivers collect continuous wave (CW) signals, whereas the digital wideband receivers primarily accumulate pulsed radar signals, even though they can still exploit the CW signals.

1.1.2 Analog Wideband Receiver

Conventional wideband receivers are essentially made up of analog components. In addition, there are numerous architectures applied for wideband receivers based on desired requirements, such as sensitivity, input signal range, dynamic range, response time, and how many simultaneous signals can be detected [7]. Analog receivers are generally classified into six categories by their structure. A further discussion on the crystal video, superheterodyne, instantaneous frequency measurement (IFM), channelized, compressive (microscan), and Bragg cell classifications can be found in [1]. For the consideration of this research the topic will only involve a conventional receiver, which can be seen in Figure 1.1.

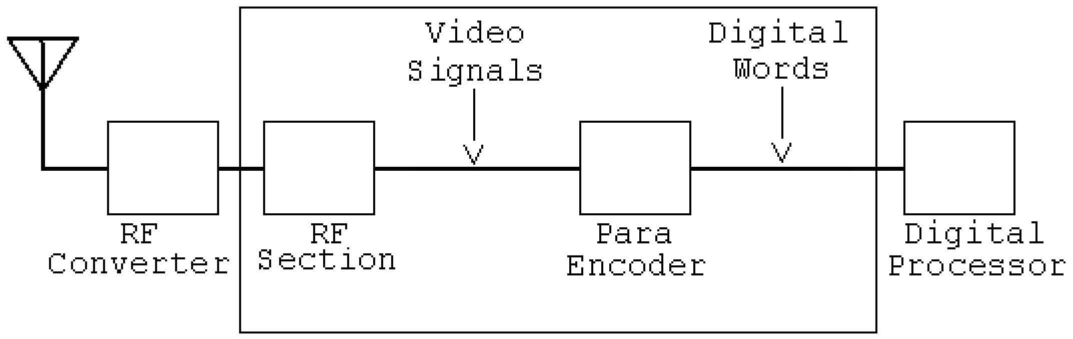


Figure 1.1: Conventional Wideband Receiver

The actual receiver itself is nothing more then a radio frequency section and a parameter encoder which are usually contained within one designed unit. The initial step in the functionality of the receiver is the antenna and RF converter. The antenna collects pulsed RF signals which range from 2 GHz to 100 GHz, but the more practical frequency range for radar is from 2 GHz to 18 GHz [7]. In order for the receiver device to process the elevated frequency impulse the RF converter is incorporated. The RF converter alters the high incoming frequency range from the antenna and down-converts it into a lower intermediate frequency (IF) signal that the receiver can process.

With the frequency range of interest altered to a more desirable structure it is delivered to the RF section for further signal conditioning. The radio frequency segment usually contains devices such as filters and amplifiers but also utilizes a diode envelope video detector which takes the RF pulses and converts them into a video, or DC signal. Once the signal has been adapted from RF to video it proceeds to a para (parameter) encoder that constructs a digital word depicting the parameters of the signal.

The digital word, also known as a pulse descriptor word (PDW), is a set of digitized signal parameters measured from the radar pulse signals that the receiver collects. The information that the PDW accumulates may contain pulse amplitude (PA), pulse width (PW), time of arrival (TOA), carrier frequency (also referred to as the RF), and the angle of arrival (AOA) [7]. In limited cases, the measurement of the electric polarization may occur.

The final phase in the flow of the conventional wideband receiver is the digital processor, which collects the generated PDW's to process them further with the intent to distinguish the signals of interest. In reality, it is assumed that a receiver will communicate with a few million pulses per second from multiple radar systems, whether friendly or not [7]. It is the function of the digital processor to exploit the measured parameters to distinguish these radar signals.

1.1.3 Digital Wideband Receiver

With the advancement in analog-to-digital converters (ADC) and the increase in digital signal processing speed, modern research has heightened in the development of digital wideband receivers [7]. The primary reason for trading analog functionality for digital domain processing is that digital signal processing (DSP) has performance advantages related to manufacturability, to insensitivity to the environment, and a greater ability to absorb design changes compared to their analog counterparts [10]. However, analog components are still essential because of the ADC's inability to sample the high frequency bandwidth of 2GHz to 18GHz usually associated with modern radar.

In contrast to the previously discussed structure of the conventional analog wideband receiver, the digital wideband receiver removes the components associated with the RF segment and incorporates an ADC and some type of spectrum estimator. The flow of a typical digital wideband receiver can be seen in Figure 1.2.

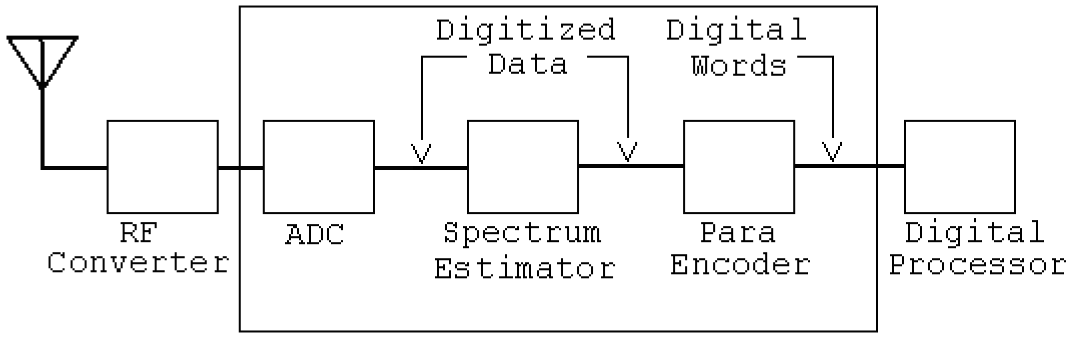


Figure 1.2: Typical Digital Wideband Receiver

The initial functionality of the antenna and RF converter has not been altered in comparison to the analog receiver. Pulsed RF signals are still collected by the antenna and then down-converted into a lower IF signal that can be easily processed. This lower analog frequency signal is then fed into an ADC, which will produce digitized data in the time domain. It is at this point that the remainder of the process is digital. This is significant because in digital signal processing there is no signal integrity losses associated with temperature drifting, gain variation, or DC level shifting as in analog circuits [7].

The next step is the conversion of the time domain digital data into the frequency domain with some type of spectrum estimator, which in the case of this research is a fast Fourier transform. The information available in the frequency domain is represented as spectral lines or spectrum density [7]. From here the parameter encoder accumulates the spectral lines for the purpose of converting them into the desired pulse descriptor words (PDW), which contain the measured parameter discussed previously. For the final phase of the process, a digital processor is once again employed to distinguish multiple signals of interest using the measured parameters in the PDW.

1.2 Motivation

FFT-based digital wideband receivers are essential elements of many modern wideband radar systems. The instrumental role in the functionality of the receiver is the capability of the FFT to take the spectral contents of a time-varying signal given by an ADC and determine the real and imaginary elements in the frequency domain. The fixed-point representation of this crucial component presented several challenges that needed to be overcome.

Currently, most modern FFT designs involve computations that require a large amount of arithmetic operations. The fast Fourier transform processor is constructed using discrete Fourier transform (DFT) butterfly modules that encompass a great majority of the overall unit. The complex multiplier employed by the butterfly networks contribute to the high complication in hardware utilization and power consumption. An approach that has been implemented to perform these complex operations was introduced by Alvin Despain [2]. In his work he established a technique to replace the complex multiplication arithmetic with simple shift operations. What has been proposed in this research is a method that utilizes this very technique.

In this thesis, the main focus is to reduce the power usage and minimize the total hardware consumption while improving the overall single signal spur-free and two-tone instantaneous dynamic range of a 256 and 128 fixed-point kernel FFT. Previous work done in the design of digital receivers and FFT’s [3-4] has shown a vast improvement of the dynamic range and hardware utilization. The FFT technique using 20 kernel points showed a marked improvement over the previous technique using 12 kernel points. Similar to early design schemes, a number of kernel functions will be used to represent the complex multiplication within a butterfly network.

Here we design a 256 and 128 fixed-point kernel FFT using 32 kernel functions. We further reduce the slices consumed by employing a folding technique that utilizes a butterfly network once and reuses it within the same stage of the FFT (folded and reused). A frequency detection block was also developed to take the sum of the squared real and imaginary units and approximate the signal frequencies.

1.3 Research Approach

The goal of this research is to design, implement, and test an FPGA-based fixed-point kernel function FFT specified for wideband receiver applications. In order to fulfill these goals a specific research approach was followed. The initial effort was taken to understand the mathematical theory and functionality of a discrete Fourier transform and the fast Fourier transform, which is an efficient algorithm for computing the DFT. The radix-2 butterfly design flow needed to be examined and a new variable truncation scheme needed to be investigated in the place of the twiddle factors. Finally, new FFT architecture techniques were investigated for portability requirements associated with the FPGA.

Once a solid foundation was produced, the theoretical design was constructed using MATLAB's Simulink environment and Xilinx's System Generator (XSG). With the design thoroughly tested and the behavioral simulations complete, synthesis and timing-based simulation was performed to verify the design would run as expected with the chosen parameters in the targeted FPGA [6].

1.3.1 FFT Design:

A typical digital wideband receiver takes in an input signal from an analog subsystem, which is then digitized by an analog-to-digital converter (ADC). The digitized data from the ADC is then fed into a spectrum estimator, which in this case is an FFT, and then a parameter encoder analyzes the spectrum and outputs pulse descriptor words that describe the characteristics of the input signal [7]. Figure 1.3 shows the flow of a typical digital wideband receiver.

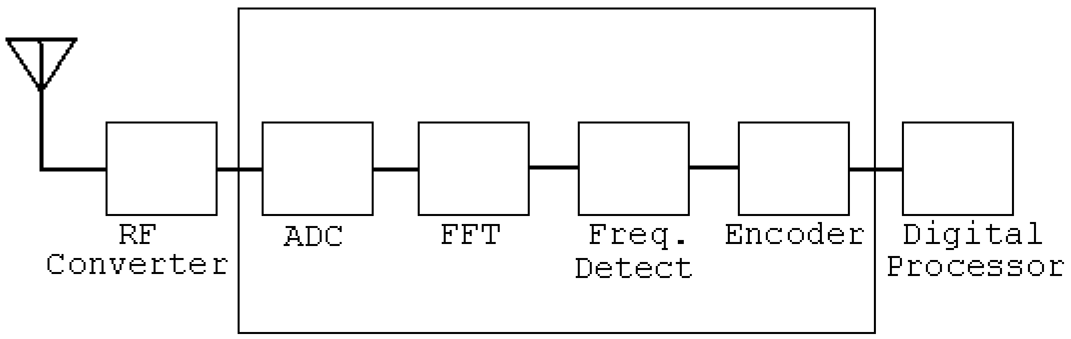


Figure 1.3: Digital Wideband Receiver

The block diagram shown in Figure 1.4 highlights the FFT and frequency detection blocks which have been developed for this research. The FFT which performs a spectral analysis of the samples supplied by the ADC was generated using MATLAB's Simulink environment and Xilinx's System Generator block sets. The frequency detection block which is no more then a sorting algorithm applied to the N outputs of the fast Fourier transform, is developed in a similar method as the FFT. Once complete the final product is generated and synthesized into a VHDL format and implemented onto an FPGA.

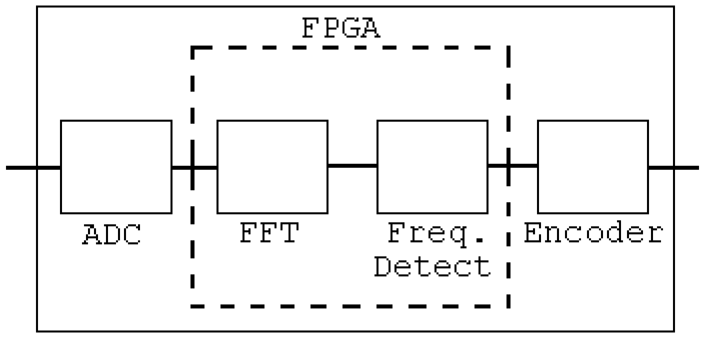


Figure 1.4: FFT and Frequency Detection

1.3.2 Testing and Analysis:

Initially, a 256 and 128 fixed-point kernel function FFT was developed and verified in an ideal environment using the Xilinx platform. Both the single signal SFDR and the two-tone IDR where performed and tested across a desired spectrum. With the validation complete the frequency detection was applied and the functionality of the two as a whole was confirmed using the same method. With the alliance of the two fulfilled, the complete design was synthesized and verified for area utilization, timing and power consumption using Xilinx ISE version 8.2i.

A complete design kit taken from Xilinx ISE was uploaded into ChipScope Pro version 8.2i for real-time verification on the FPGA. The testing of the FPGA was performed using a Xilinx Virtex-IV board model XC4VSX55, complete with an onboard Atmel 10-bit ADC. Using an RF signal generator each sample taken from the input spectrum is verified and then analyzed for low amplitude detection.

1.4 Document Organization

This thesis document is organized into six chapters. Chapter I contains a comprehensive background of the digital wideband receiver and its applications. Also, the motivation for this project and the research approach taken is discussed. The design environment in chapter II introduces a more comprehensive understanding of the field programmable gate array (FPGA) and analog-to-digital converter (ADC). The desirability of the Xilinx System Generator (XSG) software package is also considered. In chapter III the design considerations for the research is examined. The discussion of the digital wideband receiver becomes more in depth as we look at each segment more discretely. For chapter IV the global data flow and implementation of this research topic is reviewed, with a primary focus on the FFT and frequency detection architectures. The experimental and synthesis results disclosed in chapter V display the verification of the research in MATLAB, Xilinx System Generator, ISE and FPGA. Finally, in chapter VI conclusions and future work are discussed.

II. DESIGN ENVIORNMENT

2.1 Introduction

In this chapter we will be looking into the various design environments that where exploited in the completion of the research process. First of all a comprehensive view of the architecture and function of the Xilinx Virtex-IV field programmable gate array (FPGA) will be disclosed and discussed. The FPGA is onboard the Delphi ADC3255 and at the heart of the board is the Atmel 10-bit analog-to-digital converter (ADC) which will be given an overview as well. Finally, the Xilinx System Generator (XSG) design suite will be broken down to demonstrate its advantages and importance towards the completion of this research investigation.

2.2 Xilinx Virtex-IV FPGA

In recent years, field-programmable gate arrays (FPGA) have become fundamental elements in implementing high performance digital signal processing (DSP) operations, especially in the areas of medical imaging, digital communications, aerospace, and defense systems. The FPGA is a semiconductor device containing several programmable logic devices (PLD), which are nothing more then integrated circuits that can be programmed to perform complex functions. The makeup of modern day FPGA's consist of not only simple logic gates, registers, multiplexers, and look-up-tables (LUT), but also circuits that are devoted to fast adders, multipliers, and input/output processing.

The rate at which data can be read from or stored into the semiconductor device far exceeds that of a DSP processor running at clock rates two to ten times that of the FPGA. Coupled with a capability for implementing highly parallel arithmetic architectures, this makes the FPGA ideally suited for creating high-performance custom data path processors for tasks such as digital filtering, forward error correction, and of course fast Fourier transforms [8].

For the purposes of this research we will be utilizing the Xilinx Virtex-IV FPGA model XC4VSX55, which is a feasible alternative to previous FPGA's. There's usually an exceedingly large trade-off in power consumption, signal integrity, and cost to design a high-performance system such as an FPGA. For the Virtex-IV, the power consumption is cut in half in comparison to its predecessors. The lower power enables a higher clock frequency, higher reliability, better noise margins, and reduced operational costs [11]. The device also incorporates more than 200,000 logic cells [11], which are reconfigurable using hardware description languages (HDL) such as Verilog and VHDL. For the FFT design being discussed in this research, VHDL was utilized for the implementation on the board.

One of the main arguments for targeting the Virtex-IV for this implementation is the overall DSP performance. Ultra-high digital signal processing for the XC4VSX55 is being performed with up to 512 XtremeDSP slices operating at 500 MHz [11]. The configurable DSP blocks can also perform functions such as multipliers and counters without consuming logic fabric resources [11]. A photo of the XC4VSX55 model FPGA can be seen in Figure 2.1 [13].

Figure 2.1: Xilinx Virtex-IV FPGA model XC4VSX55 [13]

2.3 Delphi ADC

The Delphi ADC3255 is a PCI Mezzanine Card (PMC) on board the Xilinx Virtex-IV FPGA [12]. At the base of the board is an Atmel 10-bit analog-to-digital converter, which can function at a clock rate from 200 MSPS to 2 GSPS and higher [12]. For the functionality of the FFT in this research the ADC will be operating with a supplied input data rate of 2.56 GHz (fs) which will have an input sampling time (Ts) of 1/fs. This supplied clock of 2.56 GHz is buffered and used directly for the Atmel ADC where an analog input is sampled and converted from a real-world analog signal to digitized data.

The altered 10-bit digital data is then sent to a demultiplexer block which widens the data bus at 1/8 the input clock frequency [12]. At this point the digital data stream is supplied to the FPGA at a clock rate of 320 MHz. The modification of the digital signal begins at this point where the 10-bit data is reduced to the eight most significant bits (MSB), to produce the appropriate unit value required. The top level block diagram of the Delphi ADC3255 can be seen in Figure 2.2 [12].

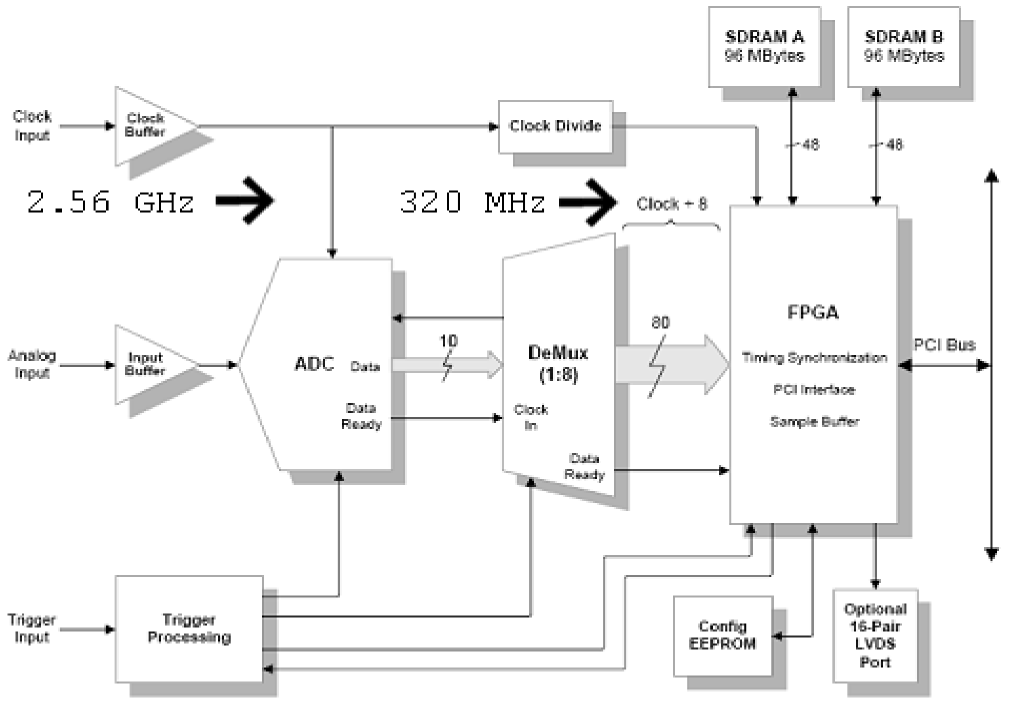


Figure 2.2: Delphi ADC3255 Top Level Block Diagram [12]

2.4 Xilinx System Generator

The Xilinx System Generator (XSG) is a software tool within MATLAB's Simulink environment that presents a high level abstract view of digital system processing (DSP) for FPGA-based designs [8]. The software utilizes hardware description languages (HDL), such as Verilog and VHDL, to construct high level abstraction for algorithm development and verification.

The Simulink design suite contains a traditional interrelated blockset, such as registers, basic logic operations, LUT's, and others, that allow for an ideal simulation environment for a designed model. The XSG is just an extension of Simulink that contains an additional blockset, called the Xilinx blockset, which contains FPGA-specific units for hardware realization. System Generator uses the variables within the Xilinx blockset to map Simulink system parameters into entities and architectures, ports, signals, and attributes in a hardware model [8]. The mapped parameters are then converted into a hierarchical VHDL netlist as well as the necessary command files to create an intellectual property (IP) block netlist, which creates project and script files for HDL simulation, synthesis, placement, routing, and bit stream generation [8].

For programming the Virtex-IV FPGA already mentioned, the XSG will transition a designed model into a synthesizable and efficient VHDL source code that is a faithful representation of the top level design. The implementation is faithful in the sense that it is bit and cycle-identical at the sample rates defined in Simulink [8]. A breakdown of the Xilinx System Generator design flow can be seen in Figure 2.3.

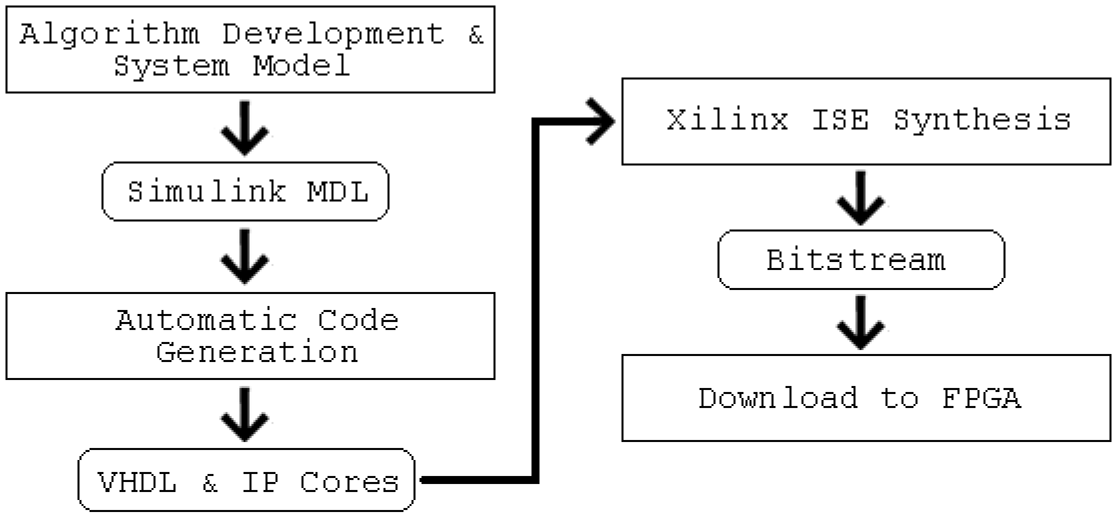


Figure 2.3: XSG Design Flow

From the very beginning, a DSP design revolves around the ability to understand the mathematical operations of a model in order to assemble a blueprint of the intended project. Once the concepts are understood, a mathematical description of the design is converted into a hardware structure by utilizing the building blocks in the XSG blocksets. The Xilinx blocksets are translated by System Generator to form subsystems and arbitrary hierarchies from the hardware description. From the newly formed Simulink model a register transfer language (RTL) can be generated to a user defined folder. The RTL consists of the HDL code files and IP cores for all the blocks within the design hierarchy.

The project files that have been created can now be imported to the Xilinx ISE Project Navigator tool. Within the ISE Project Navigator the design is combined with an FPGA design package and synthesized, simulated, and implemented in the software tool environment [8]. Finally, the represented stream of data from the previously synthesized design file is loaded to the FPGA where it is verified using Xilinx's ChipScope Pro logic analyzer.

III. DESIGN CONSIDERATION

3.1 Introduction

Within this chapter, the segments of the digital wideband receiver will be inspected a little further. The discussion of the elements leading up to the spectral estimator is very important in regards to the functionality of the design on the FPGA platform. The design consideration for the FFT, including the discrete Fourier transform and fast Fourier transform algorithms, will be investigated to provide a better understanding of their operation and purpose. Finally, the logic and importance of the spectral peak detection will be examined as well.

3.2 Digital Wideband Receiver

The functionality of the digital wideband receiver has been discussed in section 1.1. Pulsed RF signals are collected by an antenna and then down converted into a lower IF signal by an RF converter. The lower analog frequency signal is then fed into an ADC that can produce digitized data in the time domain that is then collected by a spectral estimator. The spectral estimator then converts the time domain representation into a frequency domain representation. Finally, the digital representation of the frequency spectrum is then passed to the parameter encoder, which generates a digital word that consists of several measured parameters lifted from a radar pulse.

The main focus of this research is the spectral estimator, which consists of a window function, demultiplexer, fast Fourier transform, and a frequency detector. The design flow for the FPGA-based FFT can be seen in Figure 3.1. Further discussion of each individual segment will be represented in greater detail within this chapter. Included within is an extensive look into the function of each individual segment with a more theoretical and mathematical portrayal.

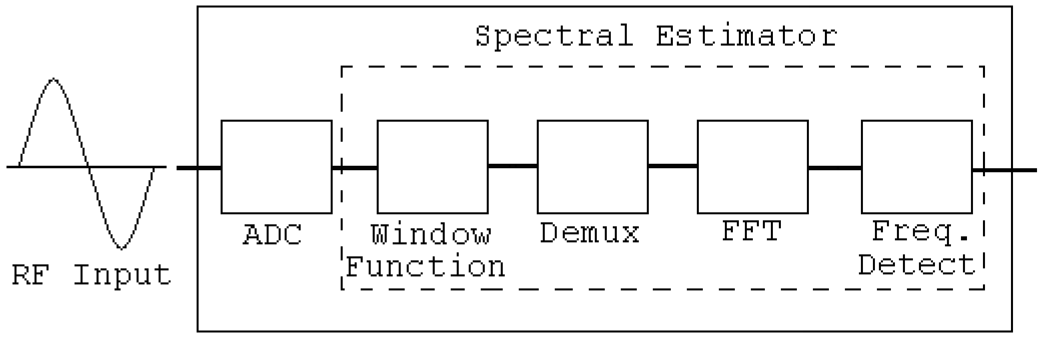


Figure 3.1: Design Flow of FPGA-Based FFT

The first point that will be considered is the analog-to-digital converter. The ideal and nonideal behavior as well as the discontinuities and errors associated will be discussed. The window functions utilized to reduce the discontinuities, such as the side lobes or leakage, in the ideal and real environments will follow. The demultiplexer, which is next in line, is used to collect the spectral signals from the window function and distribute them throughout the fast Fourier transform. An investigation into the operation and basic logic of this device will be pursued. To end the process a frequency selection block, which is used to collect and sort the outputs from the FFT, will be explored.

3.3 Translation of ADC Output

When you perform a fast Fourier transform (FFT) analysis at the output of an ADC with a pure sine wave applied, the resulting spectrum should ideally have one component at the sampled frequency of the input sinusoid with the rest represented as noise [14]. The remaining components depict nonlinearities in the transfer function of the ADC and reduce the signal-to-noise ratio (SNR) of the device [14]. The SNR is the measurement of the signal strength versus background noise of a pure sinusoidal signal. The reduction of the SNR will ultimately translate into the decline in accuracy, or effective number of bits (ENOB), of the ADC.

When an FFT attempts to translate the output of an ADC it assumes that all signals are repetitious and continuous, but data samples with a finite length tend to cause discontinuities because of the partial sinusoidal cycles. The inaccuracies resulting from the discontinuities are called side lobes or leakage. The resulting frequency spectrums can be seen in Figure 3.2(a) and (b).

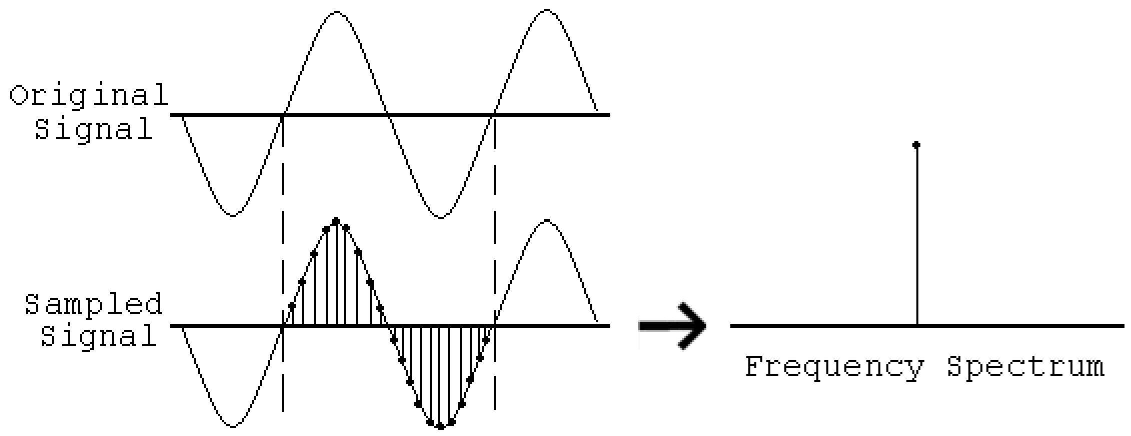


Figure 3.2(a): Ideal Sampling and Frequency Spectrum

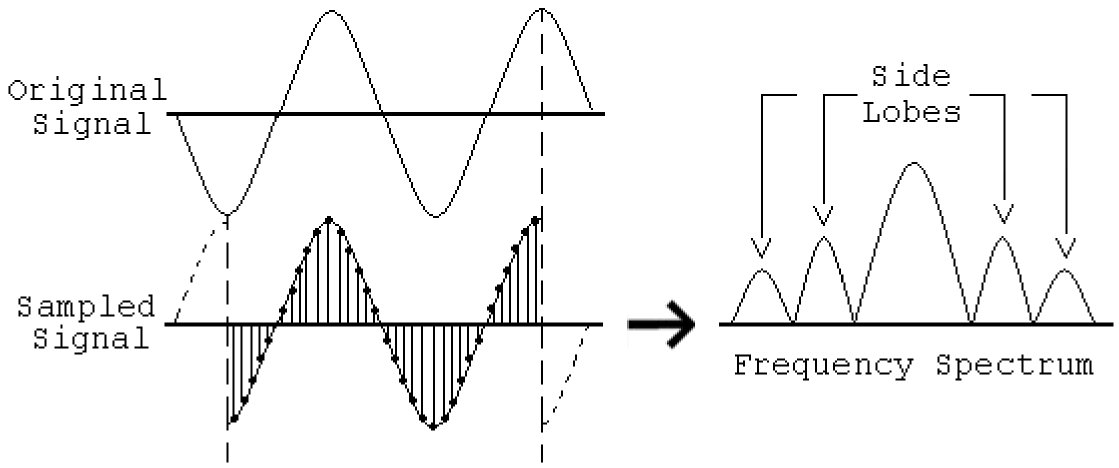


Figure 3.2(b): Partial Cycle Sampling and Frequency Spectrum

In order to reduce the leakage associated with the partial cycle sampling of the frequency spectrum a window function is utilized. The window function only samples a frequency spectrum within a specific interval while rendering the remaining samples outside the interval with a zero value.

3.4 Window Function

When an FFT performs an analysis on a frequency spectrum there are two major issues that develop. The first is the fact that a signal can only be measured for a limited amount of time and the fast Fourier transform assumes that the signal is repetitive and continuous. The second is that the FFT only calculates the results from a certain distinct spectral line, or frequency bin. The measured frequency within a certain calculated interval will be repeated uninterrupted. The issue presented with real signals is that there are discontinuities at the ends of the confined interval, and when the FFT accepts a repeated signal it will produce discontinuities that don’t exist. The inaccuracies that result from the disruption in the measured spectrum are called side lobes or leakage. The side lobes are the effect of a single spectral line taken from the frequency spectrum and spread out.

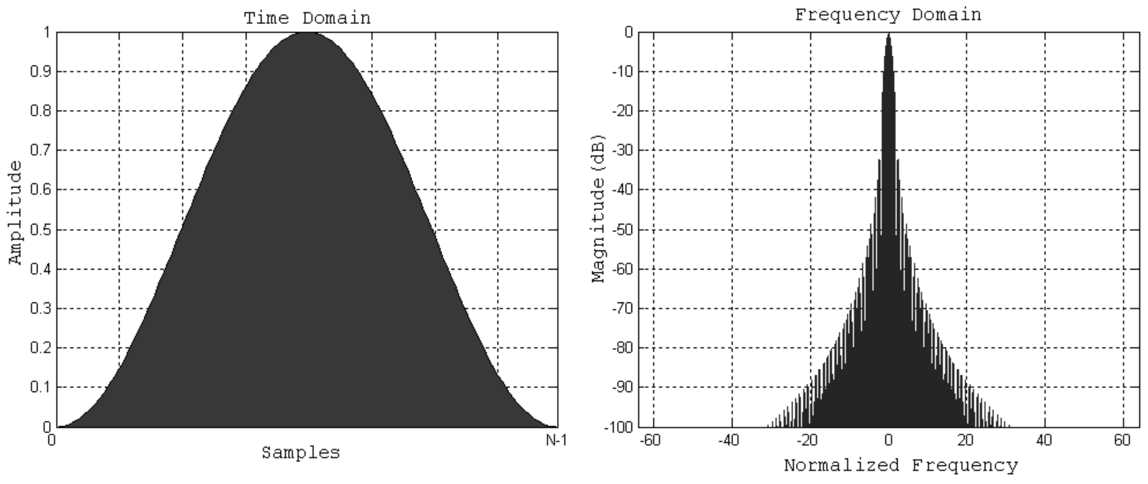
The spreading of the spectral line, or spectral leakage, could cause issues during the synthesis process of the FFT. The repercussion from the noise combined with the represented signal could degrade the signal-to-noise ratio (SNR), or the spectral spreading from a large frequency pulse could mask a smaller signal of a different frequency. In order to reduce the effects of spectral leakage significantly a window function was used. The two window functions utilized within the design process are represented in the next two sections.

3.4.1 Hanning Window

The Hann or Hanning window function was applied during the testing phase of the FFT and frequency detection in the ideal environment of Xilinx's System Generator. The Hanning window is an apodization or tapering function that is used to bring a sinusoidal signal down to zero at the edges of a sampled region in order to suppress spectral leakage. In spectral analysis it is used when a sine or cosine signal is extended beyond the length of the window. The Hanning function is shown in the Equation 3.1.

$$w(n) = 0.5\ \left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right),\ \ 0 \leq\ n \leq\ N-1 \qquad (3.1)$$

The value of *N* symbolizes the number of samples in the discrete-time window function where it is represented in the power of two. The Hanning window and the corresponding frequency response can be seen in Figures 3.3(a) and (b).



(a) (b)

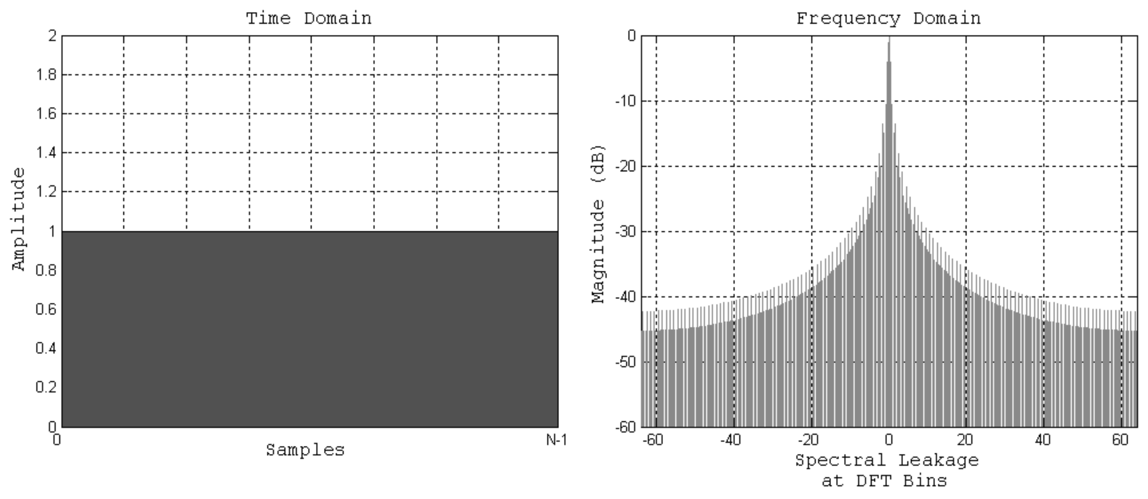
Figure 3.3: Hanning Window in Time Domain and Frequency Domain [15]

3.4.2 Rectangular Window

The Rectangular window (or no window) is the default window function used in the real spectral environment of the FPGA. The Rectangular window is a tapering function that is utilized in a similar manner as the Hann window. It is still employed to bring a sinusoidal signal down to zero at the edges of a sampled region, but for a smaller transient duration. The Rectangular function is shown in Equation 3.2.

$$w(n) = 1 \tag{3.2}$$

Where n is an integer with the value of $0 \le n \le N - 1$ and N symbolizes the number of samples represented by the power of two. The Rectangular window and the corresponding frequency response can be seen in Figures 3.4(a) and (b).



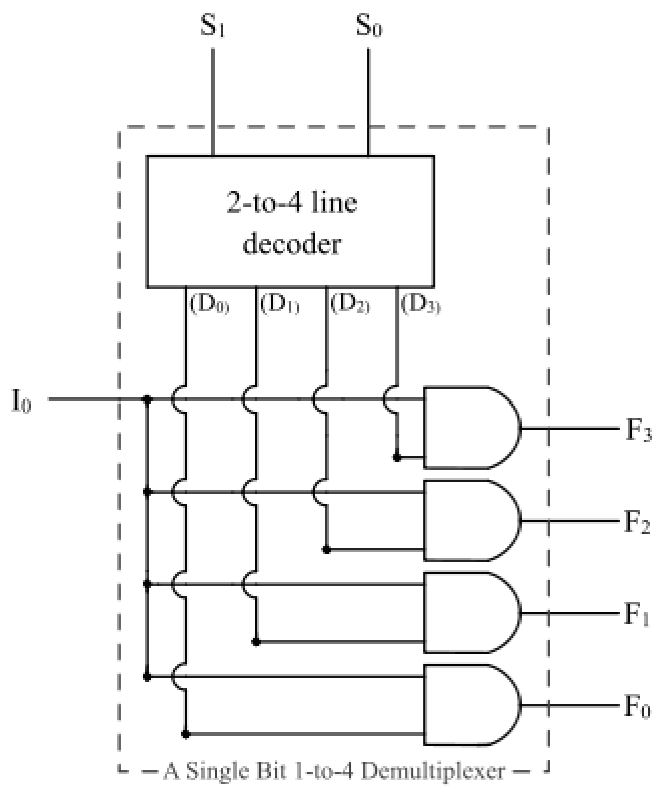
(a) (b)

Figure 3.4: Rectangular Window in Time Domain and Frequency Domain [15]

3.5 Demultiplexer Overview

The functionality of a demultiplexer is to separate a combined transmission signal, opposite of a multiplexer that combines the signals into one. The single line of information that it receives is divided by an input address used for output selection. The address will coincide with a particular output line in the demultiplexer. As an example, the architecture and truth table for a 1-to-4 demultiplexer can be seen in Figure 3.5.

The demultiplexer block for this application is applied after the window function. Its purpose is to collect the spectral data modified by the windowing operation and produce an output for every time interval of Ts, where the collected information will then be distributed through the ensuing block.



Truth Table

S_1	S_0	I_0	F_3	F_2	F_1	F_0
0	0	0	0	0	0	0
		1	0	0	0	1

S_1	S_0	I_0	F_3	F_2	F_1	F_0
0	1	0	0	0	0	0
		1	0	0	1	0

S_1	S_0	I_0	F_3	F_2	F_1	F_0
1	0	0	0	0	0	0
		1	0	1	0	0

S_1	S_0	I_0	F_3	F_2	F_1	F_0
1	1	0	0	0	0	0
		1	1	0	0	0

Figure 3.5: 1-to-4 Demultiplexer and Truth Table [15]

3.6 FFT Algorithm

The Fourier transform is one of the most commonly applied tools used for altering a function from the time domain to the frequency domain. For digital signal processing (DSP), the discrete Fourier transform (DFT) algorithm is used extensively for Fourier analysis, though never computed directly because of its complexity. Instead a collection of efficient algorithms where developed by Cooley and Tukey [16] to speed up the DFT computations considerably. The fast Fourier transform (FFT) provides a “divide and conquer” method to estimate the complex DFT algorithms.

3.6.1 The Discrete Fourier Transform

An *N*-point discrete Fourier transform (DFT) performs the conversion of time domain data into frequency domain data. The DFT operates using an *N*-point sequence of numbers, commonly referred to as *x(n)*, that are usually obtained by consistent sampling of a fixed period of some continuous function *f(x)* [17]. The DFT function of *X(k)*, which is an *N*-point sequence of *x(n)*, is defined in Equation 3.3.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j(2\pi/N)kn}, \qquad 0 \le k \le N-1 \tag{3.3}$$

The more common and simplified notation for the DFT can be seen in Equation 3.4,

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \qquad 0 \le k \le N-1 \tag{3.4}$$

where W_N represents the twiddle factor or "Nth root of unity" of a complex multiplier. The definition of the variable can be seen in Equations 3.5 and 3.6.

$$W_N = e^{-j(2\pi/N)} \tag{3.5}$$

$$= \cos\left(\frac{2\pi}{N}\right) - j\sin\left(\frac{2\pi}{N}\right) \tag{3.6}$$

The term "Nth root of unity" is frequently used to describe the twiddle factor because Equation 3.5 can be altered to give the following definition:

$$(W_N)^N = e^{-j2\pi} = 1$$

An important issue with the implementation of the DFT for an *N*-point sequence is the complex computations that cause problems for high-speed signal processing. The discrete Fourier transform requires $2N^2$ or $O(N^2)$ operations to calculate a sequence of length-*N* [17]. What this means is that an *N*-point DFT will need $N * (N - 1)$ complex additions and N^2 complex multiplications, which will demand an excessive amount hardware [17]. Because of the hardware resources needed and the complexity of the computations associated with the DFT, the fast Fourier transform (FFT) was developed to efficiently compute and reduce the number of operations involved with the discrete Fourier transform algorithm.

3.6.2 The Fast Fourier Transform

The fast Fourier transform (FFT) is able to effectively decompose and compute a DFT by utilizing the symmetry and periodic property of the complex sequence, W_N [18]. The properties are defined in Equations 3.7 and 3.8.

$$\text{Symmetry Property:} \quad W_N^{k+N/2} = -W_N^k \tag{3.7}$$

$$\text{Periodicity Property:} \quad W_N^{k+N} = W_N^k \tag{3.8}$$

By taking advantage of the DFT kernel, it is possible to obtain a much higher return in efficiency and lower the complexity below the $O(N^2)$ calculated operations.

To demonstrate the factorization of the FFT, lets consider the computation of a DFT to $N = 2^m$ points, where *m* is a positive integer value. The *N*-point sequence of *x(n)* can be separated into two entities of length *N*/2. The first grouping is comprised of even-numbered samples while the second consists of the odd-numbered samples. The resulting decimated *N*-point DFT is expressed in Equation 3.9.

$$X(k) = \sum_{\substack{n=0 \\ even}}^{N-2} x(n) W_N^{kn} + \sum_{\substack{n=1 \\ odd}}^{N-1} x(n) W_N^{kn} \tag{3.9}$$

The collection of the even and odd samples is shown as 2*m* and 2*m*+1 in Equation 3.10, where *m* = 0, 1, …, *N*/2 - 1.

$$X(k) = \sum_{m=0}^{N/2-1} x(2m)\left(W_N^2\right)^{mk} + \sum_{m=0}^{N/2-1} x(2m+1)\left(\left(W_N^2\right)^{mk}\right)W_N^k \qquad \textbf{(3.10)}$$

By exploiting Equation 3.5, W_N^2 can be simplified further.

$$W_N^2 = \left(e^{-j(2\pi/N)}\right)^2 \qquad \textbf{(3.11)}$$

$$= e^{-j2\pi/(N/2)}$$

$$= W_{(N/2)} \qquad \textbf{(3.12)}$$

Equation 3.10 can then be expressed as the following,

$$X(k) = \sum_{m=0}^{N/2-1} x(2m)\, W_{(N/2)}^{mk} + W_N^K \sum_{m=0}^{N/2-1} x(2m+1)\, W_{(N/2)}^{mk} \qquad \textbf{(3.13)}$$

where *x*(2*m*) is the sequence consisting of the even-numbered samples and *x*(2*m*+1) is the sequence consisting of the odd-numbered samples of *x(n)*. Since the DFT is periodic the odd and even segments only need to be calculated at 1/2 of the *N* values of *k*, or *N*/2 times. The result is the decimation of a high number of the required operations normally associated with the direct DFT computation in Equation 3.3.

Since each of the DFT stages are broken down into two smaller even and odd sequences it is considered to be in a class of FFT's called the radix-2 decimation-in-time (DIT) FFT [17]. The dataflow of the presented decimation-in-time FFT algorithm can be seen in Figure 3.6 for *N* = 8 input points. Where the decimation procedure is repeated for $\log_2(N)$ -1 times, yielding $\log_2(N)$ stages, until the sequence in the final stage is reduced to a *N*/4-point DFT [17]. The resulting *N*-point FFT will require *N*/2 complex multiplications per stage by some power of W_N, except for the final 2-point DFT stage where no multiplication is needed. The radix-2 DIT butterfly computation, seen in Figure 3.7, is used to simplify the calculation.

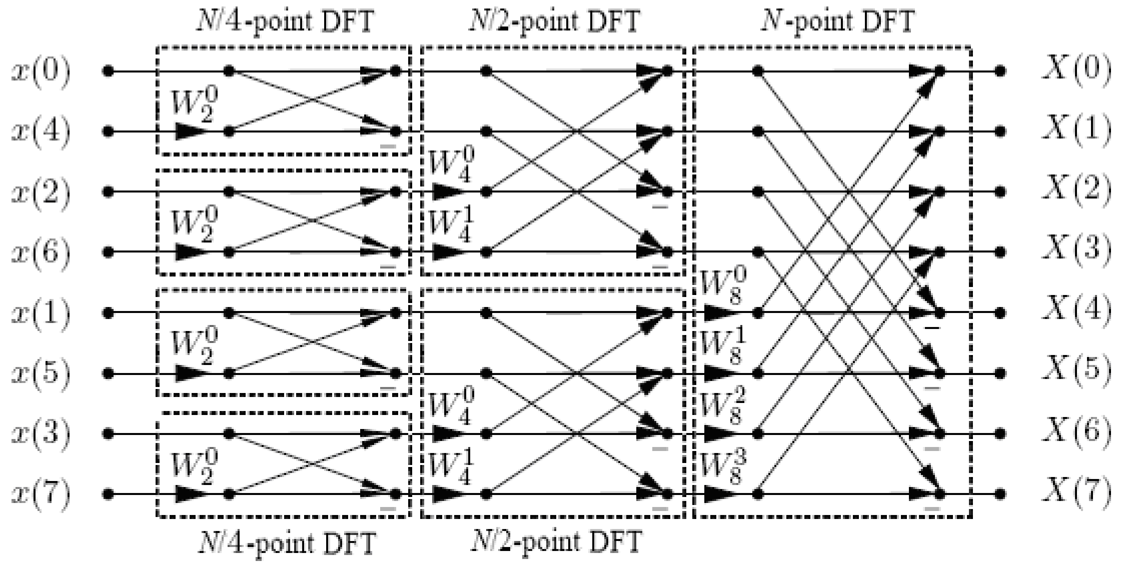


Figure 3.6: Flow Graph of an 8-Point Radix-2 DIT FFT

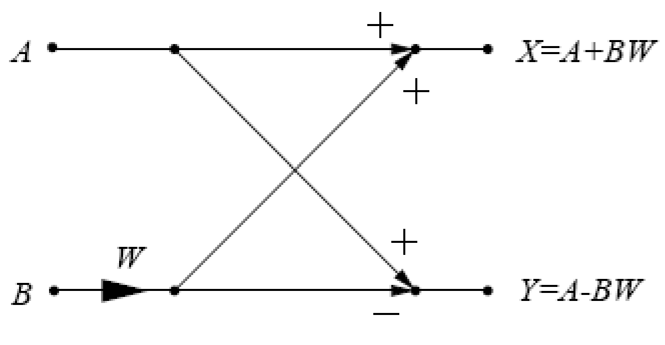


Figure 3.7: Radix-2 DIT Butterfly Signal Flow [17]

A different variation of the radix-2 FFT algorithm has been used in the course of this research. The decimation-in-frequency (DIF) FFT is a slight modification of the decimation-in-time algorithm. To obtain the similar “divide and conquer” approach the input sequence is separated into two arrays of *N*/2 data points instead of the even and odd numbered samples. The initial summation consists of the first *N*/2 data points while the other is comprised of the last *N*/2 data points. If Equation 3.9 is modified to coincide with the new DFT sequence, the following algorithm is produced.

$$X(k)=\sum_{n=0}^{N/2-1}x(n)W_N^{kn}+W_N^{Nk/2}\sum_{n=0}^{N/2-1}x\left(n+\frac{N}{2}\right)W_N^{kn} \qquad (3.14)$$

Using the symmetry property, $W_N^{Nk/2}$ can be expressed as $(-1)^k$ in the following equation.

$$X(k)=\sum_{n=0}^{N/2-1}\left(x(n)+(-1)^k x\left(n+\frac{N}{2}\right)\right)W_N^{kn} \qquad (3.15)$$

At this point *X(k)* can be decimated into the even and odd-numbered samples. The altered expressions can be seen in Equations 3.16 and 3.17, $W_N^2 = W_{(N/2)}$ still applies.

$$X(2k)=\sum_{n=0}^{N/2-1}\left(x(n)+x\left(n+\frac{N}{2}\right)\right), \qquad 0\le k\le\frac{N}{2}-1 \qquad (3.16)$$

$$X(2k+1)=\sum_{n=0}^{N/2-1}\left(x(n)-x\left(n+\frac{N}{2}\right)\right), \qquad 0\le k\le\frac{N}{2}-1 \qquad (3.17)$$

Using the *N*/2-point DFT’s, *X(2k)* and *X(2k+1)*, the process can be decimated for $\log_2(N)$ stages using *N*/2 radix-2 DIF butterfly computations. Same as the decimation-in-time algorithm, the operation will require $(N/2)\log_2(N)$ complex multiplications and $N\log_2(N)$ complex additions. The flow of an 8-point DIF FFT and the radix-2 DIF computation can be seen in Figure 3.8 and 3.9.

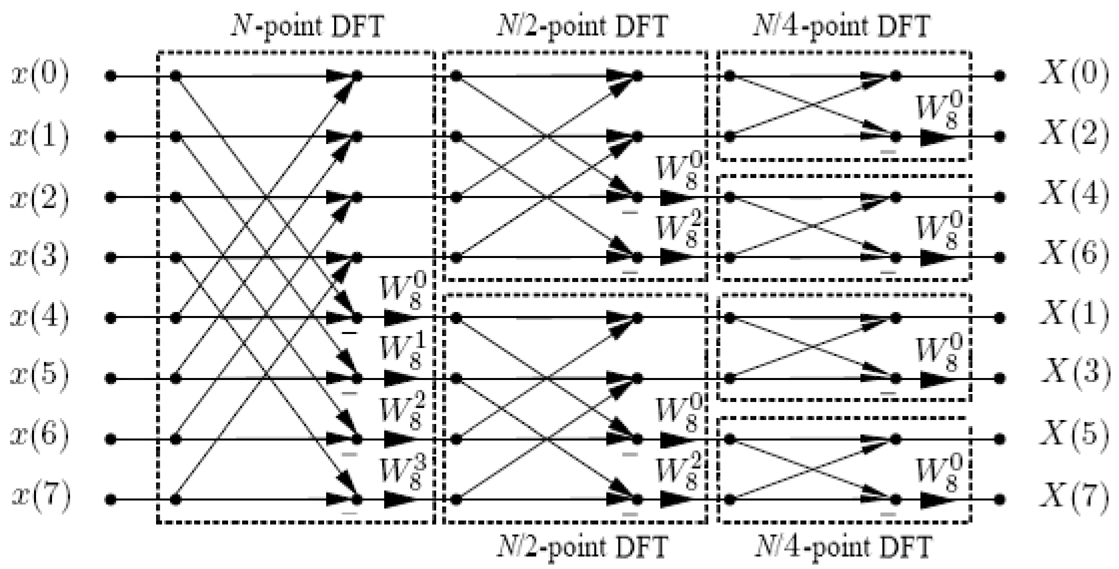


Figure 3.8: Flow Graph of an 8-Point Radix-2 DIF FFT

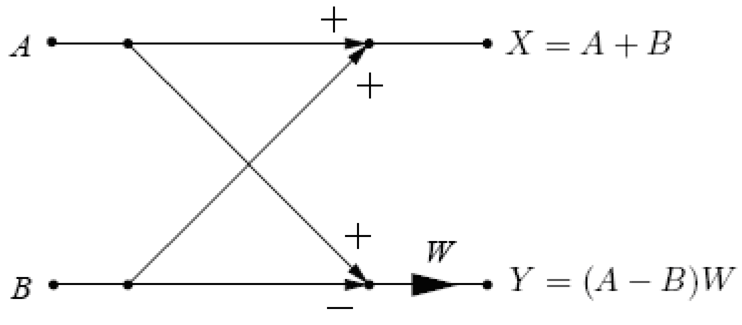


Figure 3.9: Radix-2 DIF Butterfly Signal Flow

It can be observed that the input sequence of *x(n)* occurs in natural order while the output of the decimation takes place in a bit-reversed order. The function of the algorithm is still the same as the DIT FFT. The input sequence of value *N* is decimated for log2(*N*)-1 times until it is reduced to a 2-point DFT in the final stage showing a gain in efficiency of *O*(*N*/log*N*) operations. Table 3.1 shows a comparison in the efficiency of the DFT computation and the demonstrated FFT algorithms of length-*N*.

Transform length (N)	DFT operations	FFT operations	DFT ops $\div$ FFT ops
16	256	64	4
128	16,400	896	18
1024	1.05×10^6	10,240	102
32,768	1.07×10^9	4.92×10^5	2185
1,048,576	1.10×10^{12}	2.10×10^7	52,429

Table 3.1: Comparison of DFT and FFT Efficiencies [17]

3.7 Frequency Detection

The purpose of frequency detection for a fast Fourier transform is to expose high signal peaks among a contaminated output spectrum. The signal detection displays the location and height of a frequency bin by determining the maximum amplitude of each spectral sample. The spectral samples are represented as a frequency interval of *fs*/*N*, where *fs* is the sampling rate and *N* is the input value of the FFT. Due to the sampled nature of the spectrum produced by the fast Fourier transform each peak is accurate within half a sample [19].

The frequency detection that is implemented within this research employs binary-tree based logic. The collected amplitude value for each spectral sample is compared with a neighboring frequency bin until a maximum amplitude value is determined. An example of a 4-point binary-tree representation of the peak detection can be seen in Figure 3.10. Each input in stage one is a binary expression of the frequency bins. For this example the first (00) and last (11) bins represent the higher two amplitude values.

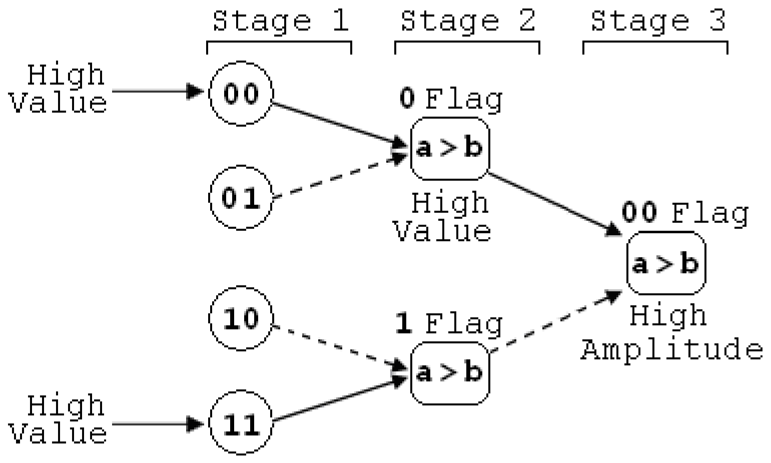


Figure 3.10: Binary-Tree Representation of Peak Detection

A comparison of the first and second set of frequency bins in stage two yields a flag that depicts the location of the higher bin. Since each comparator that is shown in the second stage is only considering two input bins the flag utilizes a single bit representation, (0) for the first input bin and (1) for the second. The comparison in stage three considers all four of the input frequency bins, hence the two bit representation for the flag. In the example, the first frequency bin that is collected has the maximum amplitude. The final flag depicts the location of the highest spectral sample.

IV. DESIGN METHODOLGY

4.1 Introduction

Within this chapter, the main focus will be on the design methodology and flow of the 256 and 128 fixed-point kernel FFT. A brief discussion of the data flow through the analog-to-digital converter and the window function will start the design approach. The implementation of the FFT architecture will follow, with a concentration on the fixed-point kernel approximation and component layouts in System Generator. An explanation of the folded decimation-in-frequency method used to construct the FFT's will also be addressed. The final point that will be examined will be the architecture and implementation of the peak detection block utilized in this research.

4.2 Global Data Flow

The design approach of an FPGA-based fixed-point kernel FFT and spectral peak detection will be analyzed in the majority of this chapter, but it is important to understand the elements leading up to the FFT design to obtain an understanding of the overall operation. An RF input provided by a signal generator will circulate through the Delphi ADC that was previously discussed in section 2.3. The operation of the ADC will be examined once again to clarify the function in regards to the proposed design. The output is then fed into a uniform window or rectangular window function. The deliberate use of this tapering function will be explored. The final segment that will be assessed is the implemented design of the demultiplexer for both the 256 and 128-point FFT layouts. The data flow of the complete process can be seen in Figure 4.1.

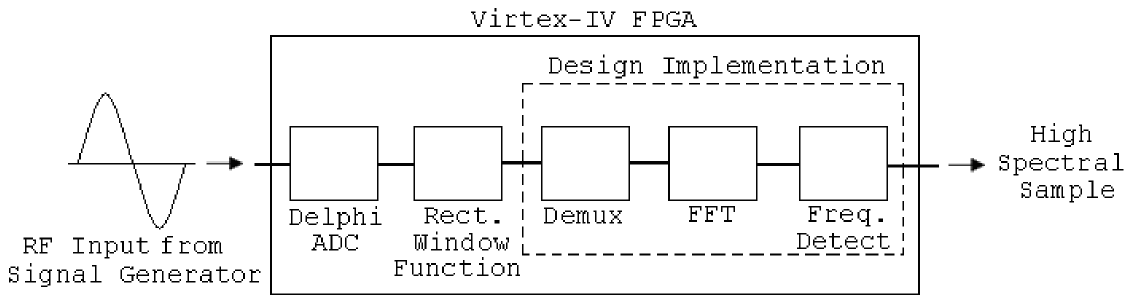


Figure 4.1: Data Flow of FPGA-Based FFT

4.2.1 ADC

At the base of the Delphi ADC3255 discussed in section 2.3, are an Atmel 10-bit analog-to-digital converter and a 1-to-8 demultiplexer. For the proposed application, the ADC received a supplied clock rate, or sampling frequency (*fs*), of 2.56 GHz with each analog signal provided at the input having a sampling interval (Ts) of 1/2.56 GHz. The altered 10-bit digitized data samples then advance to the 1-to-8 demultiplexer, which widens the provided data bus at 1/8 the supplied sample rate. From this point the digital data stream is supplied to the window function at a clock rate of 320 MHz.

4.2.2 Window Function

For each of the eight outputs of the preceding demultiplexer a uniform, or rectangular, window function is applied. For the real world analysis accomplished through this research, the data needs to be evaluated in a succession of short time intervals. The windowed sequence can be more accurately characterized as an infinite pulse which is zeroed outside a specific range, shown in Figure 4.2 [20]. The importance of this application is minimization of the discontinuities associated with the incoming measured samples. The rectangular window also provides a high frequency resolution, but with a high amount of spectral leakage. The mathematical expression for the rectangular window can be seen in Equation 4.1. A more extensive discussion regarding the spectral window functions can be found in previous sections.

$$w(n) = \begin{cases} 1, & \text{if } 0 \le n \le N-1 \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

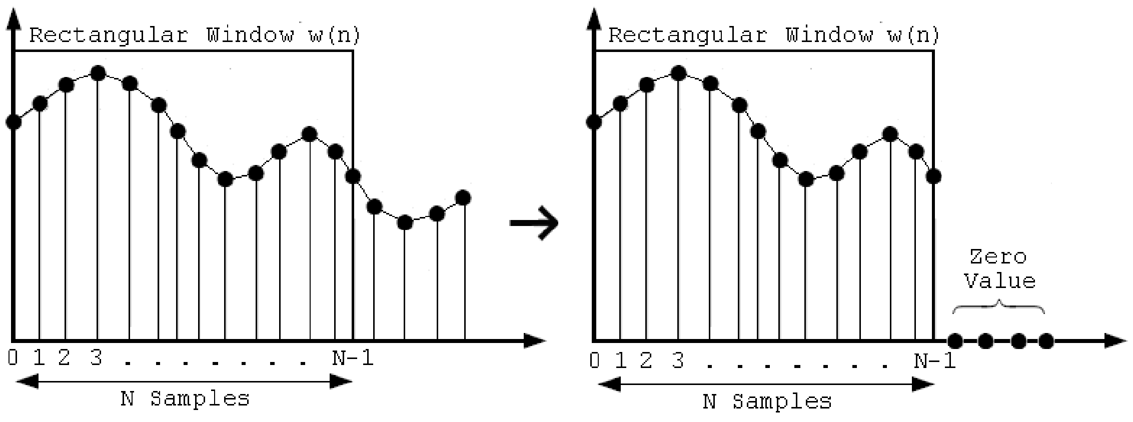


Figure 4.2: Windowing Samples of Length-N

4.2.3 Demultiplexer

The purpose of the demultiplexer block is to collect the sampled windowed spectrum from the ADC and distribute them throughout the implemented FFT architecture. As was discussed previously, the ADC samples eight 10-bit outputs at a sampling frequency of 320 MHz. Once the sequence is passed through the uniform window function it is truncated to the eight most significant bits (MSB) to produce the appropriate unit value required in this research.

In the case of the 128-point FFT architecture, 128 eight bit samples are collected every 50 ns by eight 1-to-16 point demultiplexer blocks. With the sampling interval of each demultiplexer coming in at 320 MHz, the FFT will receive an input data rate of 20 MHz (*fs*/*N*). The larger designed 256-point FFT will require the collection of 256 samples at a throughput rate of 100 ns. The necessary logic for this implementation is eight 1-to-32 point demultiplexer blocks that will distribute information at a data rate of 10 MHz. The diagrams of the two instances referred to above can be seen in Figures 4.3(a) and (b).

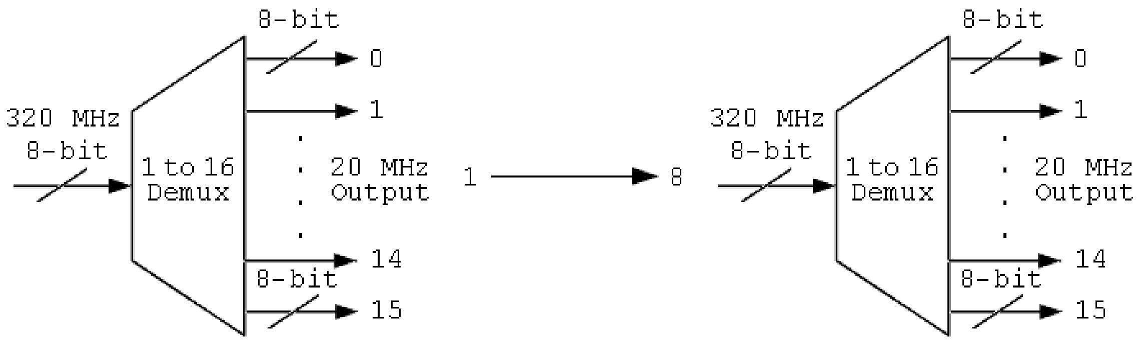


Figure 4.3(a): Diagram of Demultiplexer Block for 128-Point FFT

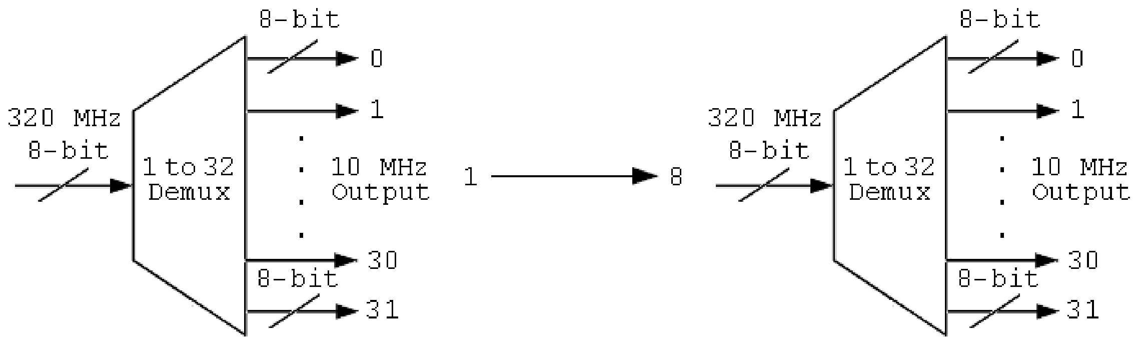


Figure 4.3(b): Diagram of Demultiplexer Block for 256-Point FFT

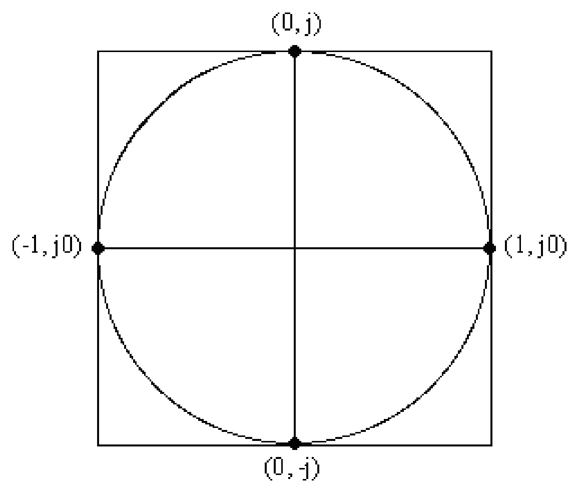
4.3 128 and 256 Fixed-Point Kernel FFT Implementation

In previous digital receiver designs [3, 20-21], the fixed-point kernel functions utilized for the FFT computations ranged from four to twenty with a gradual increase in performance for each increment. For this research, the proposed number of kernel functions where improved to thirty-two with the intent of conserving hardware area and providing a boost in system performance. The kernel points, or twiddle factors, are the weighing factors exploited throughout the FFT architecture. The execution and implementation of the butterfly networks utilizing the kernel functions will be deliberated further in this section.

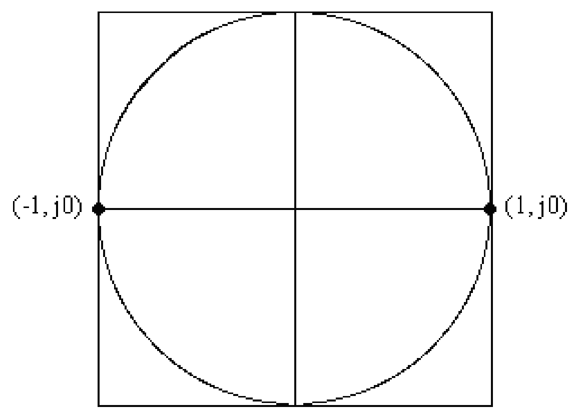
4.3.1 FFT Fixed-Point Kernel Function

As discussed in previous sections, the fast Fourier transform (FFT) was developed to efficiently decompose and compute the complex discrete Fourier transform (DFT) algorithm. The most weighted factor for this difficult computation is the complex roots of unity, or kernel functions, denoted by W_N in the FFT equations [3]. The biggest challenge in the implementation of the fixed-point Fourier transform is finding an appropriate representation for these floating point integers in the butterfly networks.

If we consider the final 2-point DFT stage of the radix-2 decimation in frequency FFT algorithm. The single butterfly computation involves no complex arithmetic and the kernel points can be easily realized in the real-imaginary coordinate axis of a unit circle. The same can be said of the subsequent 4-point DFT stage, which requires two complicated multipliers that are presented as four kernel points of ideal unit circle values. The kernel functions used in butterfly operations for the final two stages in the DIF FFT architecture can be seen in Figure 4.4(a) and (b).



(a) 4-Point DFT



(b) 2-Point DFT

Figure 4.4: Twiddle Factors for 2 and 4 Point DFT Stages

The complex computations of W_N for the two stages represented are placed within a unit circle of true or ideal lengths of unity. For the final 2-point DFT, it can be observed that the two kernel points can be denoted by the real-imaginary values of (1, j0) and (-1, j0), due to the property of symmetry only (1, j0) is needed for calculation. A similar representation can be seen with the subsequent stage, whose kernel values are expanded to four with the ideal values of (0, j) and (1, j0) used for the W_N computation. These variables are easily depicted in the hardware implementation of the multiplier, but as the input sequence of N for the FFT increases the twiddle factors become more complex.

If we consider the ideal fixed-point representation of the 8-point DFT, there are eight kernel points that can be distinguished from the four complex multipliers utilized by the butterfly network; because of symmetry only four are used in calculation. The result of the escalated number of twiddle factors is a fractional representation in the real-imaginary coordinate axis that is not easily realized in hardware. The eight kernel points that correspond to the 8-point DFT computation are presented in the unit circle seen in Figure 4.5.

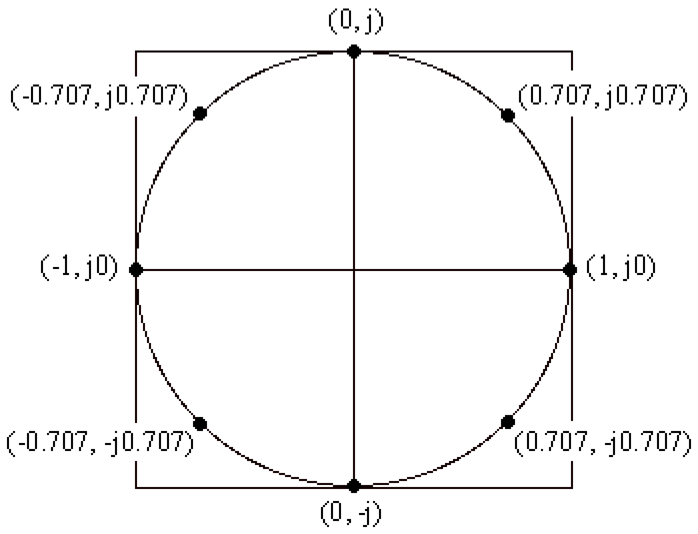
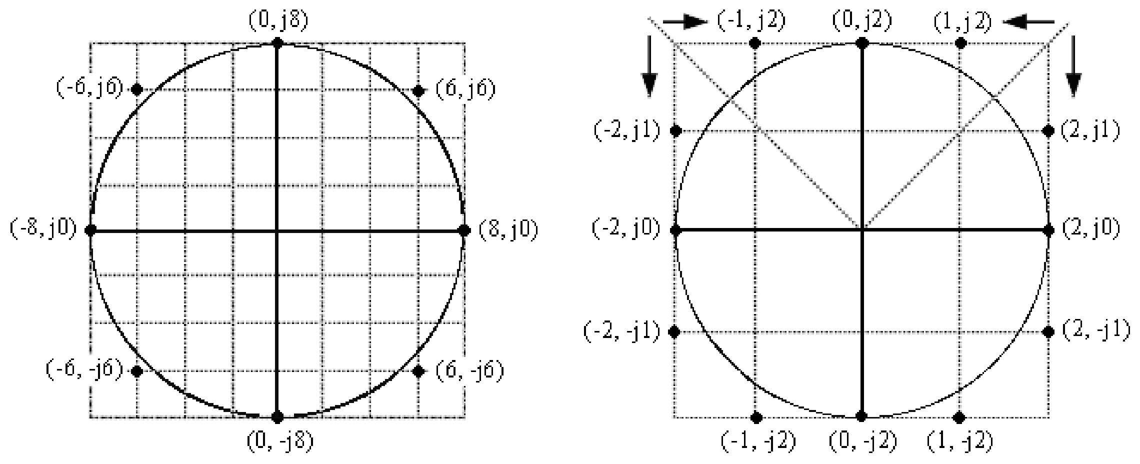


Figure 4.5: Ideal Kernel Functions for the 8-Point DFT

In order to simplify the complex calculations presented by the kernel functions (0.707, j0.707) and (-0.707, j0.707). The floating point values need to be reworked to correspond to the required fixed-point integer values. An approach to represent the difficult computations in a manner that is easier to implement was proposed in [22]. The method suggests that the unit circle be expanded by a power of two so the real and imaginary lengths of unity can be better represented in the coordinate axis.

For the kernel functions characterized in the ideal 8-point DFT above, the unit circle was scaled to an optimum value of eight, shown in Figure 4.6(a). The reason for using such a high scalar is to compensate for the rounding errors that would have otherwise been present with a smaller unit circle expansion. As an example, we could scale to a value of two, shown in Figure 4.6(b). By doing this the number of kernel points present would rise to twelve so that they can maintain a fixed-point integer value. In order to properly represent the eight kernel functions required, the fractional unit value from the ideal representation above would have to be rounded to the nearest integer. Regardless of using either (1, j2) or (2, j1) to depict one of the two fractional angles in the ideal unit circle above, the loss in accuracy due to the rounding error will be much higher in comparison to the (6, j6) unit value that is utilized in the unit circle scaled by eight.



(a) (b)

Figure 4.6: Unit Circle Scaled by 2 and 8

The design requirements for the FFT architectures implemented in this research called for the use of 32 kernel points, which required the unit circle to be scaled accordingly. In Tables 4.1 and 4.2, an optimal unit circle expansion and kernel function representation for each stage of the two FFT designs carried out in this thesis are shown.

Stage	1	2	3	4	5	6	7	8
FFT Bit Size	6	6	6	6	6	4	1	1
Circle Radian	32	32	32	32	32	8	1	1
Fixed Kernel Points	32	32	32	32	16	8	4	2

Table 4.1: Optimal Unit Circle Expansion for 256-Point FFT

Stage	1	2	3	4	5	6	7
FFT Bit Size	6	6	6	6	4	1	1
Circle Radian	32	32	32	32	8	1	1
Fixed Kernel Points	32	32	32	16	8	4	2

Table 4.2: Optimal Unit Circle Expansion for 128-Point FFT

4.3.2 Radix-2 Butterfly Architecture

The butterfly is the computational building block with which the proposed fast Fourier transforms are calculated. As a reminder, the algorithm for computing the radix-2 DIF FFT is represented in Equation 4.1,

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \qquad 0 \le k \le N-1$$

$$= \sum_{n=0}^{N/2-1} x(n) W_N^{kn} + W_N^{Nk/2} \sum_{n=0}^{N/2-1} x\left(n + \frac{N}{2}\right) W_N^{kn} \qquad \mathbf{(4.1)}$$

where the complex twiddle factor is represented in Equation 4.2.

$$W_N = e^{-j(2\pi/N)}$$

$$= \cos\left(\frac{2\pi}{N}\right) - j\sin\left(\frac{2\pi}{N}\right) \tag{4.2}$$

This equation is calculated by replicating the radix-2 butterfly for some length-*N* depending on the design requirements of the FFT in question. A simplified way to accomplish the computation in hardware was derived from the architecture for general digital signal processing (DSP) chips [23]; the implementation can be seen in Figure 4.7.

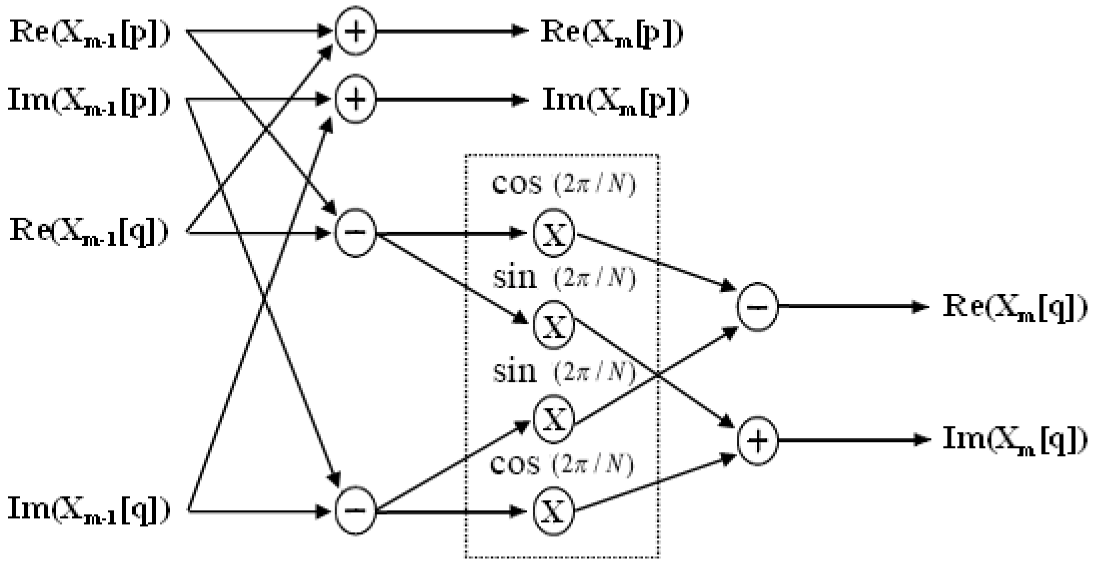


Figure 4.7: Radix-2 Butterfly Implementation

Following the signal flow of the represented DIF butterfly calculation, a sequence of real and imaginary values are received from a previous stage in the FFT and added or subtracted accordingly. The new variables are then involved with the complex twiddle factor multiplications before proceeding through more simplified arithmetic and continuing onto the subsequent stages.

The problem with the perceived calculations is the difficulty in producing an area efficient realization in hardware. If the twiddle factors where to be computed using the shown multipliers, it would consume a large amount of the hardware resources and the FFT design would become less efficient for high-speed calculations [5]. What has been done to limit the amount of hardware consumed by the complex multiplications was presented in [2, 22].

The fixed-point kernel functions that where produced by utilizing the method explained in the previous section, are used to replace the sine and cosine operations denoted in Figure 4.7. To represent the new twiddle factors, shift and add, or subtract, logic was implemented. The new butterfly computation can be seen in Figure 4.8, the presented architecture is taken from the XSG model used for a single butterfly operation in the completed FFT designs.

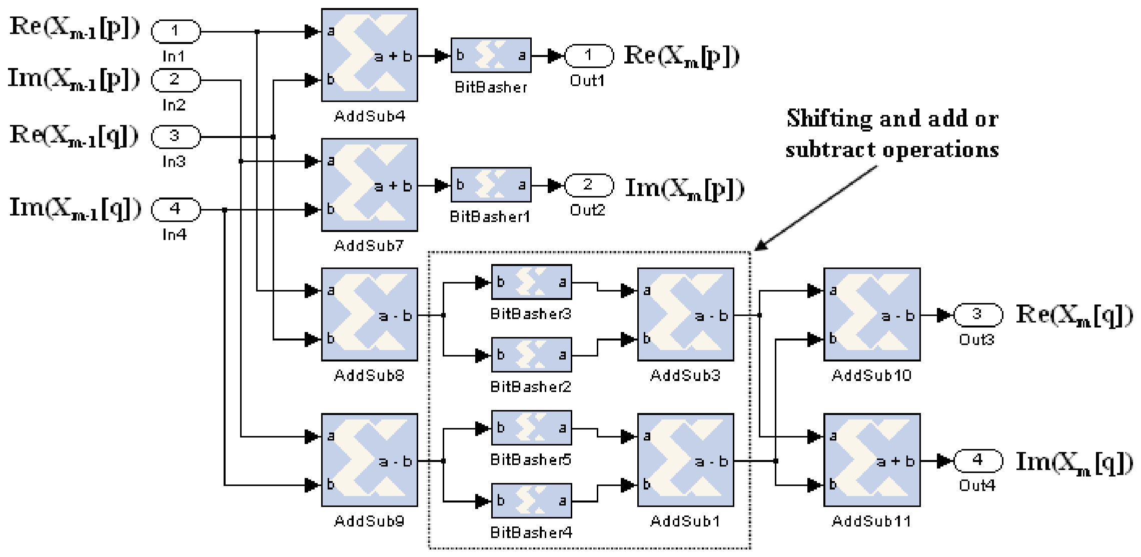


Figure 4.8: XSG Radix-2 Butterfly Implementation

To briefly explain how the new logic works, the XSG version of the butterfly calculation representing the previously discussed fractional kernel value in the 8-point DFT was presented. After the unit circle has been scaled to eight, the complex kernel point of (0.707, j0.707) is modified to (6, j6), which needs to be portrayed, in this case, using shifting and subtraction logic.

The butterfly is taking in 8-bit real and imaginary values that are then subtracted. This new 9-bit unsigned number is then truncated by the power of two and subtracted once more to obtain a desired value. The first shift function truncates the LSB bits by three, which represents a multiple of eight, and the second truncates the LSB bits by one, which represents a multiple of two. When the two values are subtracted from each other the desired twiddle factor is represented.

4.3.3 Fixed-Point Kernel DIF FFT Architecture

What is known about the decimation-in-frequency (DIF) FFT architecture thus far is that an input sequence of length-N is decimated over a $\log_2(N)$ amount of stages until the sequence is reduced to a final 2-point DFT stage. If we look back at Figure 3.8 in section 3.6.2, it can be seen that the stages are continuously divided by a value of $N/2$ as the data advances throughout. It is shown that as the algorithm progresses to the later stages, the DFT's are being reused to complete the computations.

This Fourier transform technique is very effective in minimizing the complexity of the DFT algorithm, but consumes a large amount of hardware and power by recycling the butterfly networks. In this research, a method of reusing a decimated DFT only once per stage has been implemented to help produce an effective architecture and consume a minimal amount of the hardware resources. This folding technique is represented in

Figure 4.9, for a 4-point DIF FFT. It is clear from the figure provided, that only one 2-point DFT is being employed to accomplish both cycles used in the previous technique.

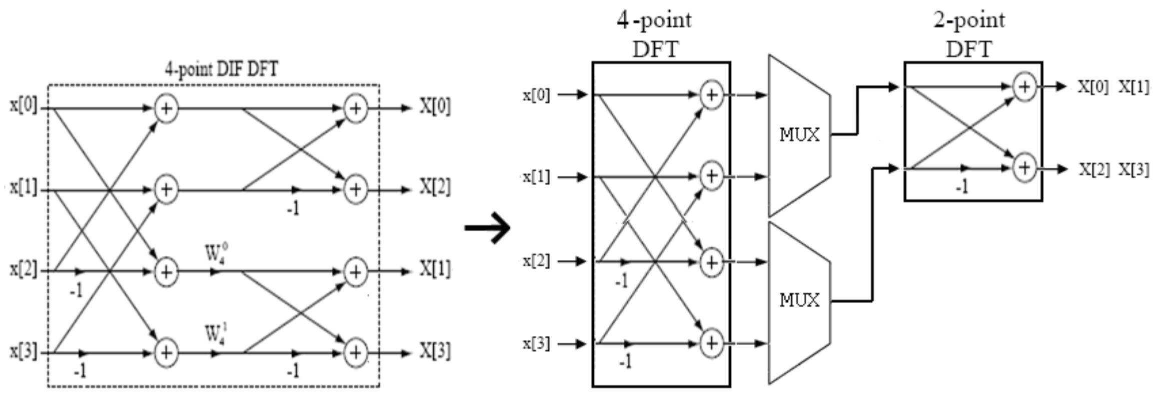


Figure 4.9: 4-Point DIF FFT with Folding

The folding of the butterfly networks can be extended over any length of *N*-points utilized by the FFT to create a more efficient architecture. The major challenge in the execution of this technique is the amplified data rates that take place in the folded stages. When folding is applied to a stage in the fast Fourier transform implementation, the data rate of that phase doubles in comparison with the previous phase. In order for the FFT to adapt to the routing delays at each folded junction, the technique is only applied a certain number of times.

For the 128 fixed-point FFT realized through this research, only the second, third, and fourth stages employed the folding technique. The architecture shown in Figure 4.10 is the final implementation of the 128 fixed-point DIF FFT. What is revealed through the presented architecture below is once the FFT receives the data samples from a 128-point demultiplexer block, at a sampling frequency of 20 MHz, the data rates continuously double through the course of the design. In order to accommodate the data rates required, a series of multiplexers where applied at the outputs of the folded stages. With the completion of the final 16-point DFT, the sampling frequency reached a value of 160 MHz, which is then passed through a 64-point demultiplexer and bit reversal logic that produces a sampling frequency equal to the first DFT stage.

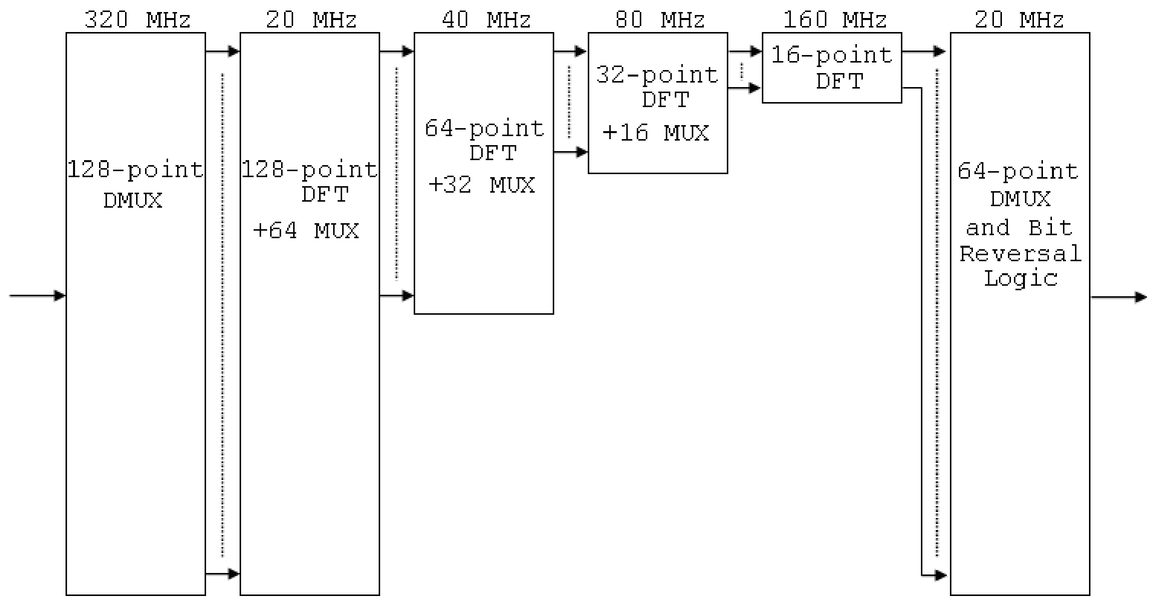


Figure 4.10: 128-Point DIF FFT with Three Stages of Folding

For the 256 fixed-point FFT implemented, the same rules explained above are relevant. With this design, the folding technique was spread over to include a fourth stage with an input from a 256-point demultiplexer block applying a sampling frequency of 10 MHz. The data rate of 10 MHz from the expanded 256-point DFT is then doubled through the completion of the 16-point DFT phase, which will be applied to a 128-point demultiplexer and bit reversal logic. The diagram for the 256 fixed-point DIF FFT architecture can be seen in Figure 4.11.

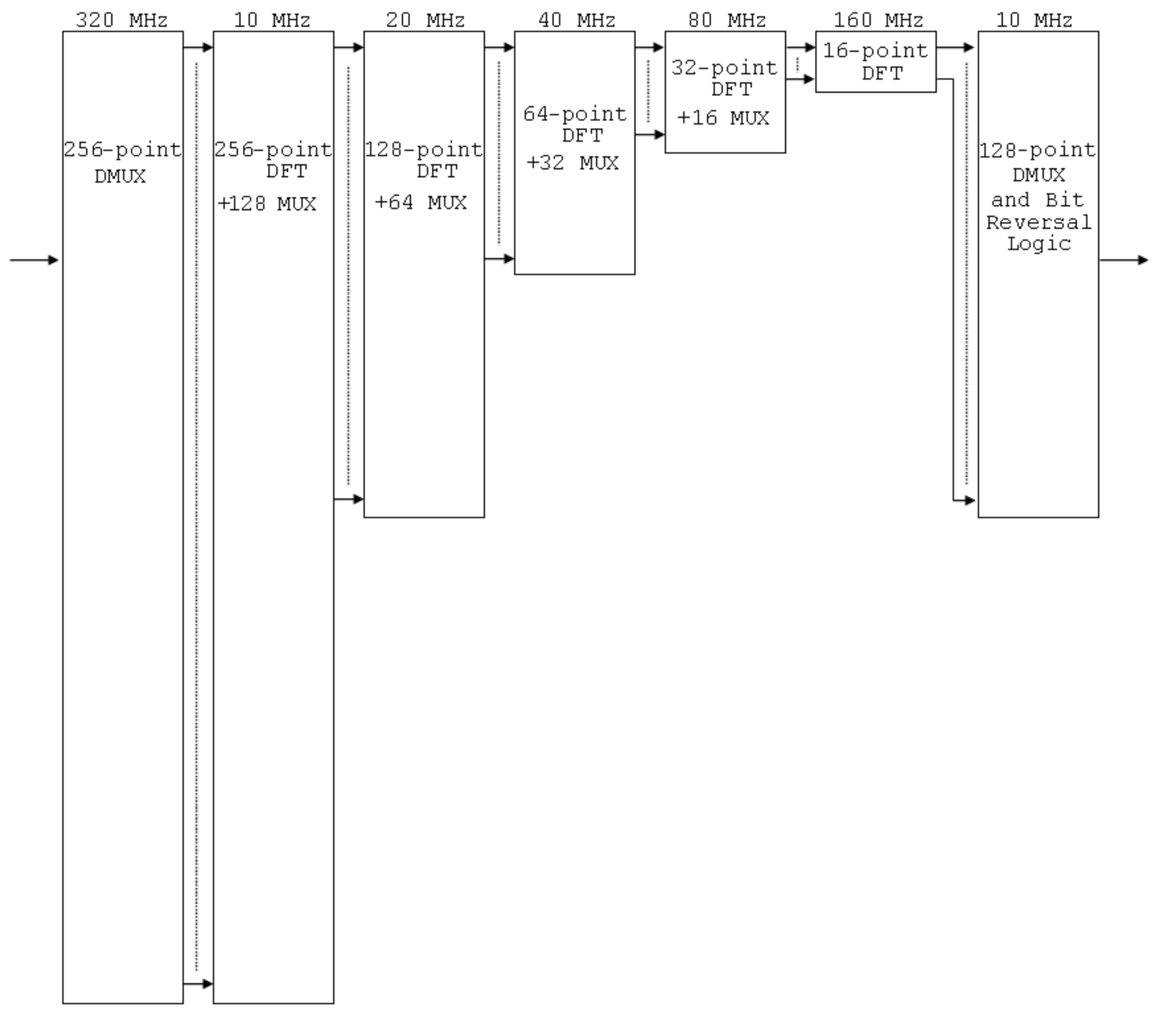
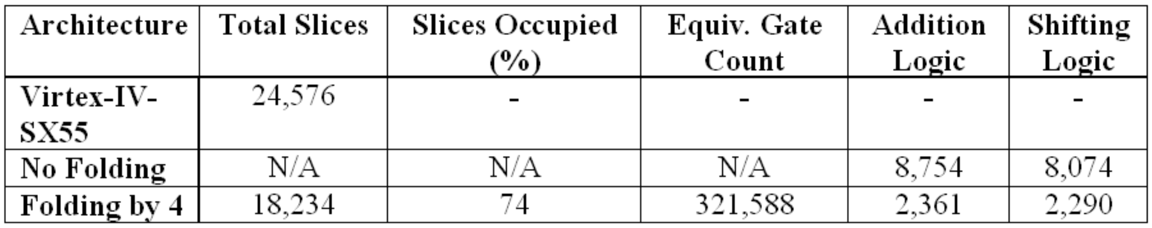


Figure 4.11: 256-Point DIF FFT with Four Stages of Folding

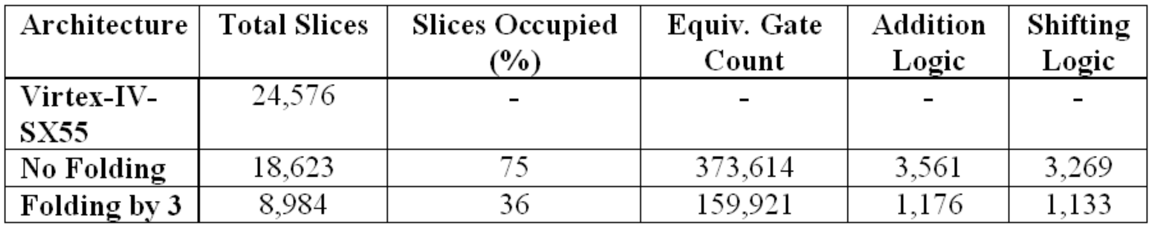
The results from the folding technique showed a significant reduction in the consumption of the FPGA slices. The XSG implementation of the 256-point DIF FFT using the conventional technique was impossible to synthesize, but with four folded stages the total slices consumed was reduced to 75% of the 24,576 available. It can also be seen that there was an extensive reduction in the shifting and addition or subtraction logic that was utilized in the design architecture. This includes the new representation of the complex twiddle factor computations, the approximate values showed a reduction of

72%. For the 128-point FFT, the folding of three stages was able to condense the already expended slices by about 48%, while also reducing the computational logic by 66% in comparison to the unfolded model. The two FFT architectures were synthesized using the Xilinx ISE environment, and the total hardware resources utilized are shown in Tables 4.3 and 4.4.



Architecture	Total Slices	Slices Occupied (%)	Equiv. Gate Count	Addition Logic	Shifting Logic
Virtex-IV-SX55	24,576	-	-	-	-
No Folding	N/A	N/A	N/A	8,754	8,074
Folding by 4	18,234	74	321,588	2,361	2,290

Table 4.3: 256-Point FFT Synthesis Results



Architecture	Total Slices	Slices Occupied (%)	Equiv. Gate Count	Addition Logic	Shifting Logic
Virtex-IV-SX55	24,576	-	-	-	-
No Folding	18,623	75	373,614	3,561	3,269
Folding by 3	8,984	36	159,921	1,176	1,133

Table 4.4: 128-Point FFT Synthesis Results

4.4 Frequency Detection Architecture

The results taken from the FFT architectures discussed above produce an N-point frequency response from the *N*-point spectral samples taken at the input. Each of these generated points in the frequency response are called the frequency bins. The purpose of the peak detection, if we recall from section 3.7, is to distinguish a specified response, or frequency bin, among an output spectrum that may be contaminated because of spectral spreading or leakage.

Each of the bins are represented as an interval of *fs*/*N*, where *fs* is the sampling rate of 2.56 GHz and *N* is the amount of samples taken at the input of the FFT. For the 256-point FFT architecture, the frequency bins coincide to samples that are taken from the input spectrum at the interval of 10 MHz, while the 128-point FFT receives samples for every interval of 20 MHz. At the completion of the fast Fourier transform computations each frequency bin requested is isolated using the peak detection architecture shown in Figure 4.12 for a length of *N*/2. The data rates by which the samples are processed are 10 and 20 MHz for the 256 and 128-point models respectively. A simplified explanation of the over all operation of the peak detection can be viewed in previous sections.

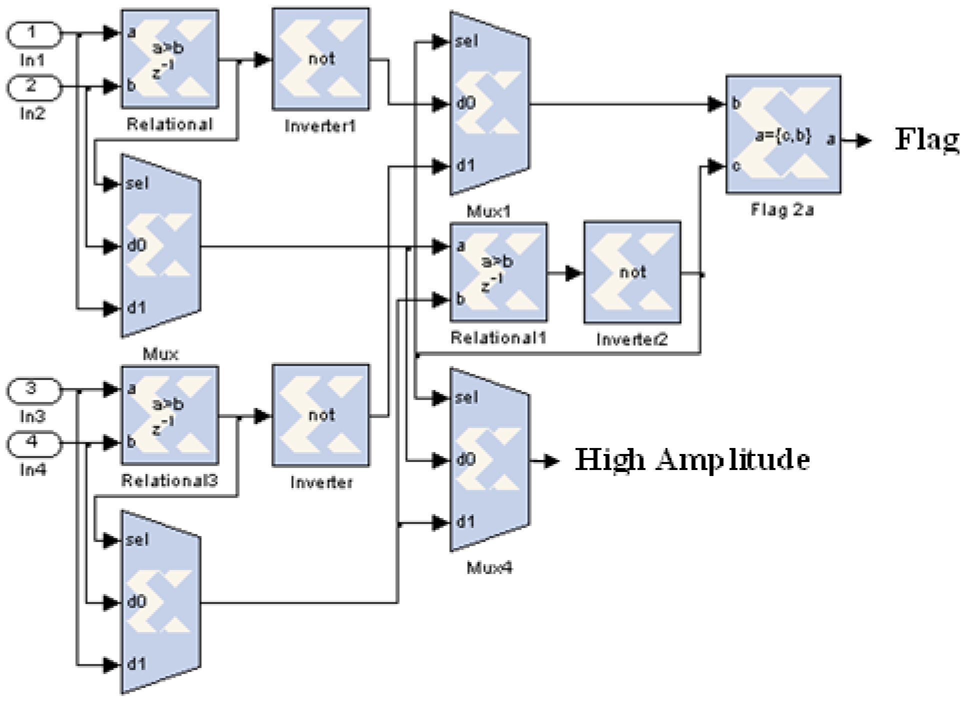


Figure 4.12: 4-Point Peak Detection Architecture

V. EXPERIMENTAL AND SYNTHESIS RESULTS

5.1 Introduction

Within this chapter the procedure implemented to perform a detailed analysis of the FPGA-based 128 fixed-point kernel DIF FFT and the investigation of the 256 fixed-point kernel DIF FFT in the ideal environment will be justified. At the opening of this section the outcome of the two FFT models using the Xilinx System Generator application will be examined. Both the single signal spur-free dynamic range (SFDR) and the dual tone instantaneous dynamic range (IDR) will be determined in the ideal ADC XSG environment. In the subsequent sections, the 128-point FFT will be evaluated using the Xilinx ISE tool for power and area consumed, and verified on the Xilinx Virtex-IV FPGA for low signal detection.

5.2 Xilinx System Generator Results

The Xilinx System Generator (XSG) is a software tool within the MATLAB Simulink environment that takes advantage of variables within the Xilinx blocksets in order to map the operating requirements into the architectures for the FFT models. Using this design suite the mathematical representation of the decimation-in-frequency FFT was implemented into an ideal working model for verification. In Figure 5.1 is the XSG representation of the 128 fixed-point kernel DIF FFT and the ideal input signal setup using the Xilinx blocksets.

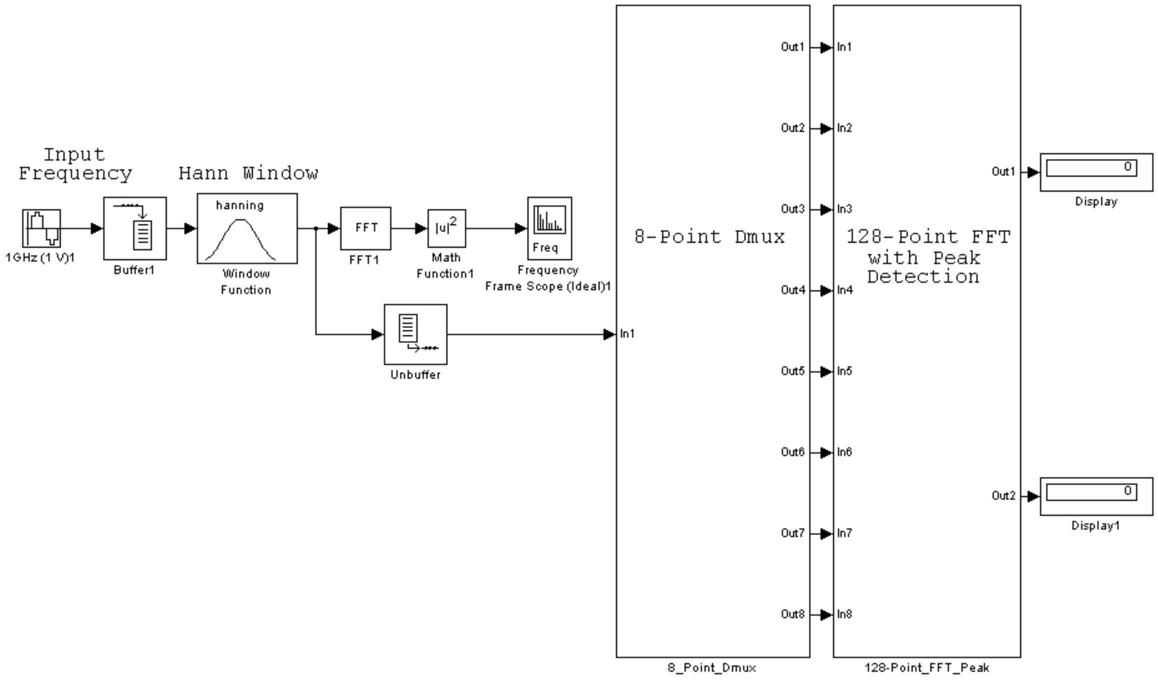


Figure 5.1: XSG 128-Point DIF FFT Model

To authenticate the operation of the design in question for an ideal environment, a thorough statistical analysis was conducted for the single signal spur-free dynamic range (SFDR) and the two-tone instantaneous dynamic range (IDR) utilizing the Xilinx System Generator for DSP version 8.2.02. The Xilinx model was confirmed using a 2.56 GHz sampling frequency *(fs)* that collects *fs*/*N* points from a bandwidth of 1.24 GHz (20 MHz to 1.26 GHz) at 20 MHz intervals with output throughput rates of 50 ns.

The verification of the 256 fixed-point DIF FFT was conducted in a similar fashion. The single signal SFDR and two-tone IDR where analyzed at the same clock rate of 2.56 GHz with the collected samples taken from a bandwidth of 1.25 GHz (10

MHz to 1.26 GHz) at 10 MHz intervals with an output throughput rate of 100 ns. The XSG model of the 256 fixed-point DIF FFT, with an ideal ADC input signal, can be seen in Figure 5.2.

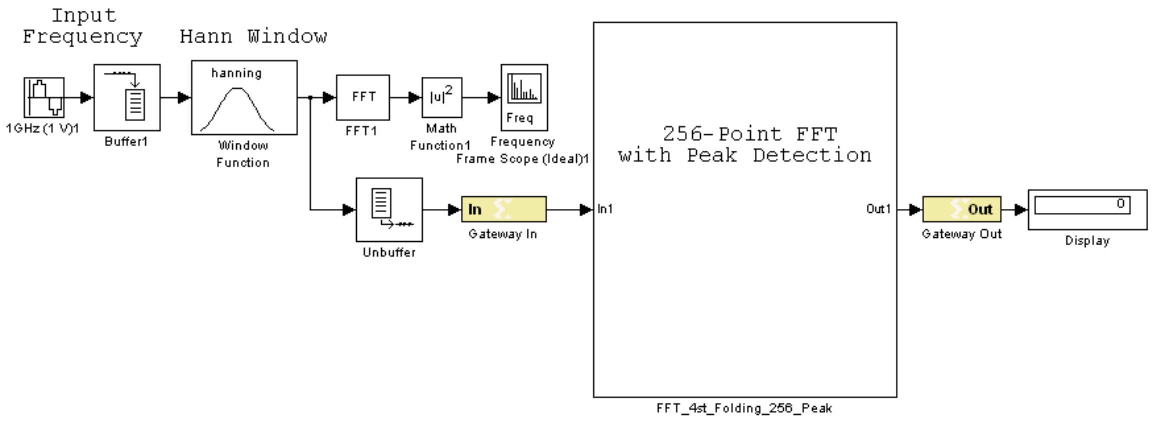


Figure 5.2: XSG 256-Point DIF FFT Model

5.2.1 Spurious-Free Dynamic Range (SFDR)

The principal definition of the spur-free dynamic range (SFDR) is the strength ratio of the fundamental signal to the strongest undesirable signal in the output, where the spur is classified as a nonsignal component within the spectrum that is confined to a single frequency. The frequency response that is below the spurious-free dynamic range is not credible and is especially difficult to distinguish as a true response. In Figure 5.3 and 5.4, is a demonstration of the SFDR for the sampled frequencies of 400 MHz and 800 MHz that where acquired during the evaluation of the 128 fixed-point decimation-in-frequency FFT model.

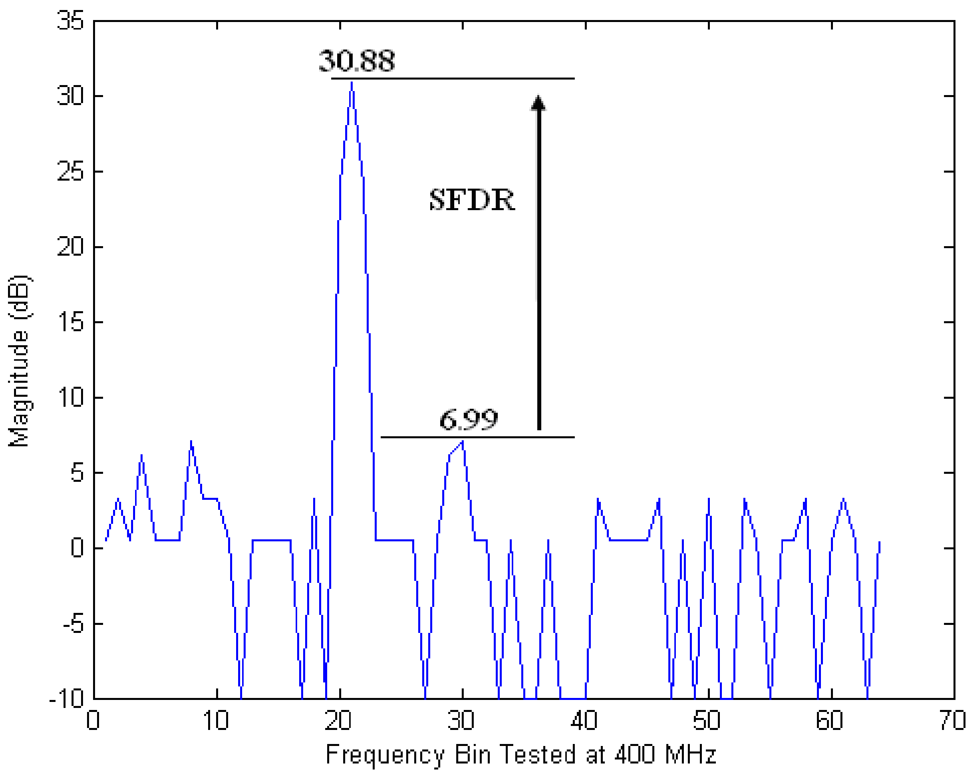


Figure 5.3: 128-Point FFT Model, SFDR at 400 MHz

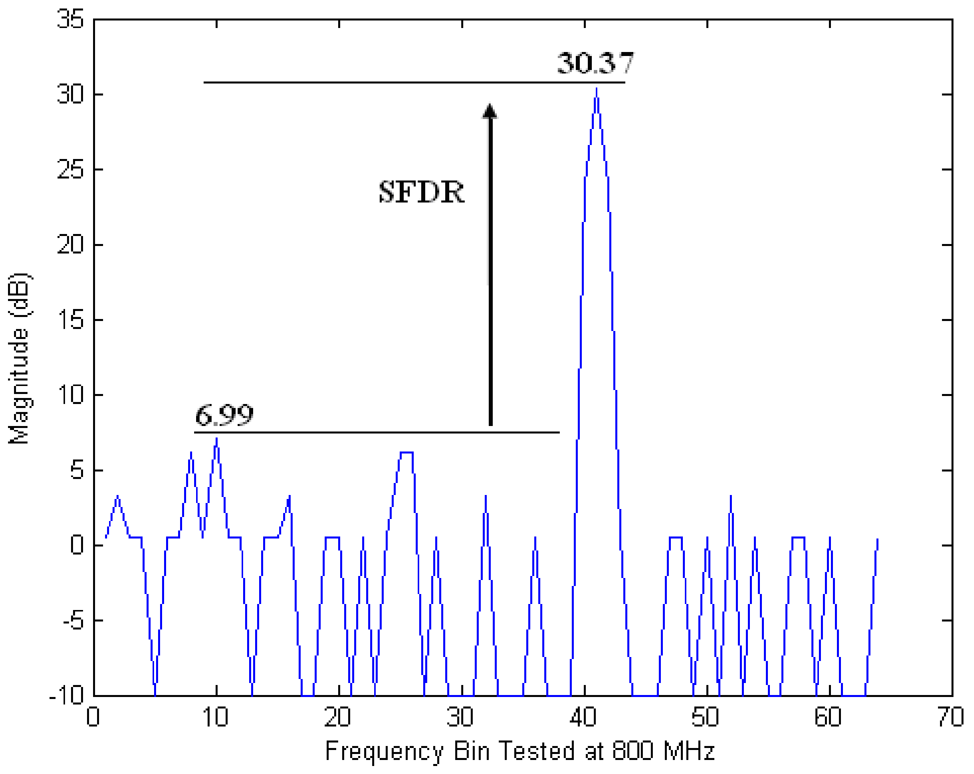


Figure 5.4: 128-Point FFT Model, SFDR at 800 MHz

In Figure 5.3, the magnitude at 400 MHz has a peak of 30.88 dB while the strongest spurious signal has a peak of 6.99 dB. The spurious-free dynamic range at the 400 MHz frequency bin is evaluated as the difference between the two values, for this case the SFDR is 23.89 dB. The same verification method is executed on the response in Figure 5.4, where the magnitude at 800 MHz, has a high peak of 30.37 dB and a strong spur at 6.99 dB, which results in a SFDR of 23.38 dB at the frequency bin. The dynamic range of the model was conducted for the frequency range of 20 MHz to 1.26 GHz. The spurious-free dynamic range of the 128 fixed-point FFT, with the folding technique applied to three stages, produced an average of 22.78 dB. The sample distribution can be seen in Figure 5.5.

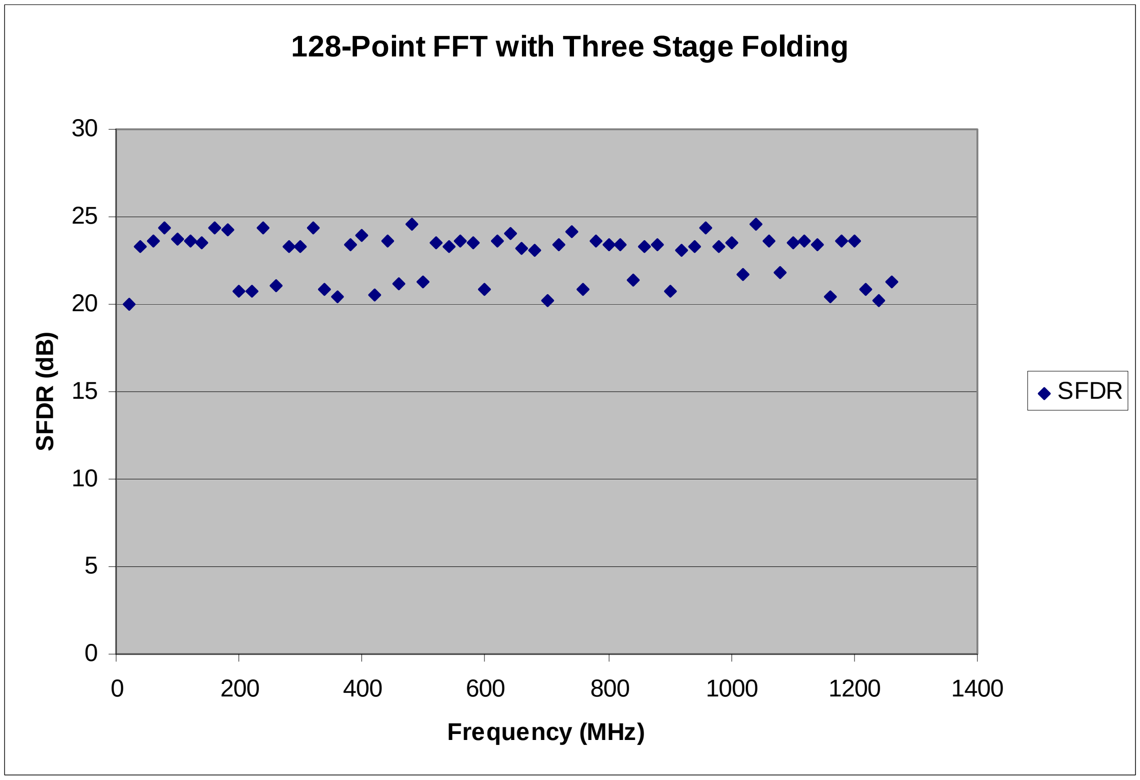


Figure 5.5: 128-Point FFT Model, SFDR Distribution

As a comparison, a demonstration of the SFDR for the sampled frequencies of 400 MHz and 800 MHz have also been included from the evaluation of the 256 fixed-point FFT model. In Figure 5.6, the frequency bin at 400 MHz showed a dynamic range of 21.09 dB, while the 800 MHz response in Figure 5.7 showed an SFDR of 20.83 dB. The average dynamic range for the FFT model was conducted for a frequency array from 10 MHz to 1.26 GHz. The average spurious-free dynamic range of the 256 fixed-point FFT, with the folding technique applied to four stages produced an average of 21.66 dB. The sample distribution can be seen in Figure 5.8.

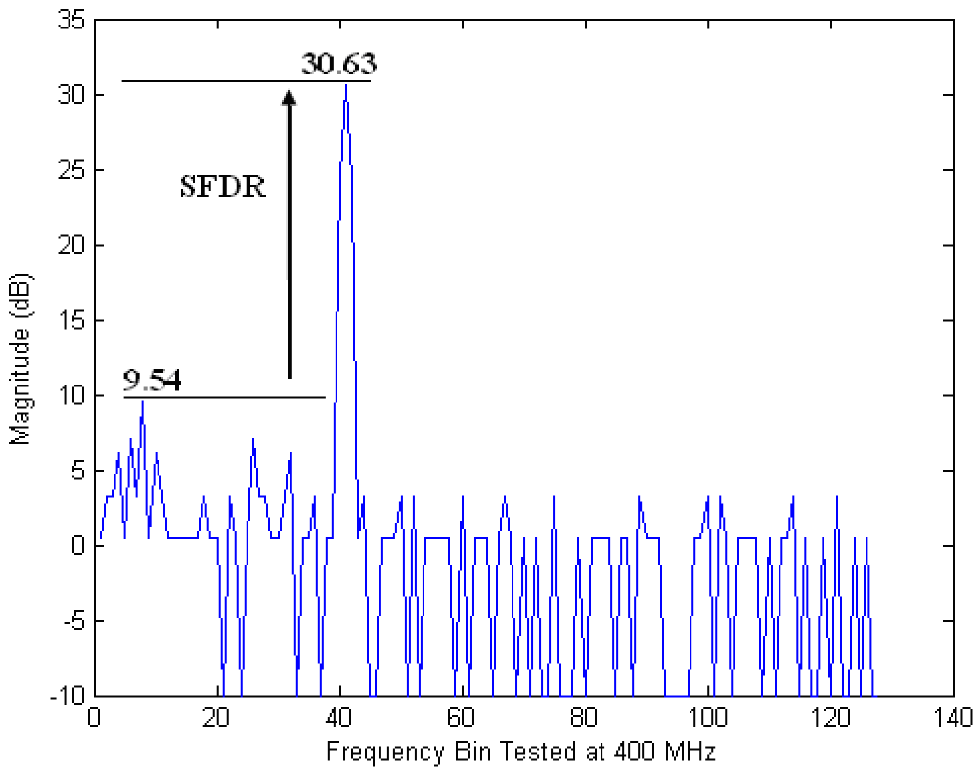


Figure 5.6: 256-Point FFT Model, SFDR at 400 MHz

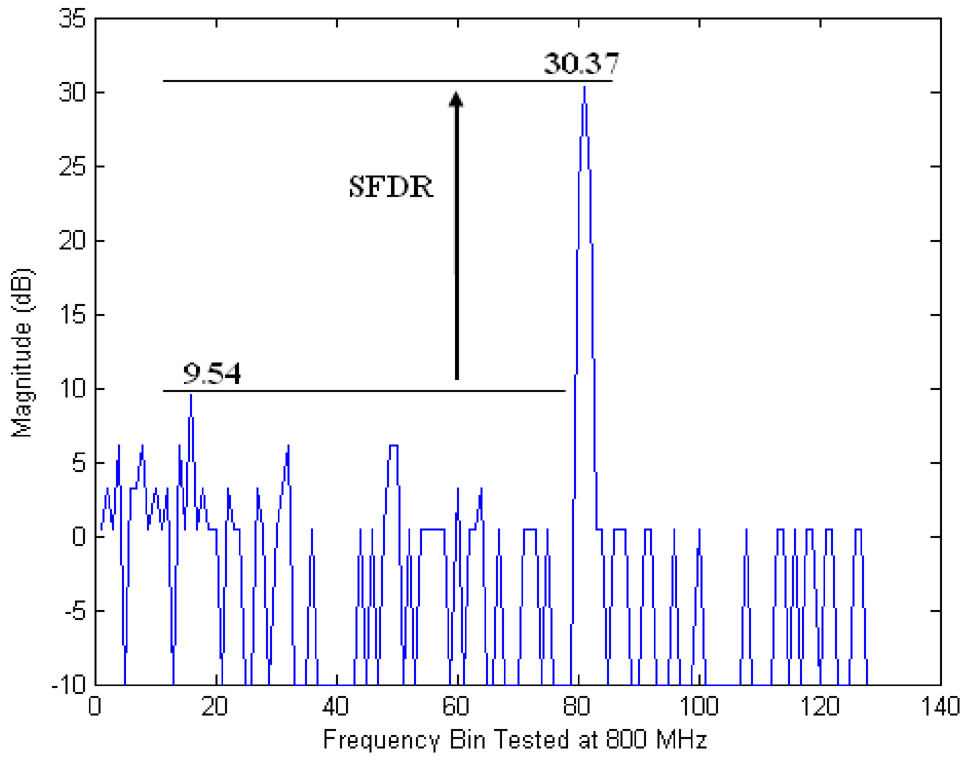


Figure 5.7: 256-Point FFT Model, SFDR at 800 MHz

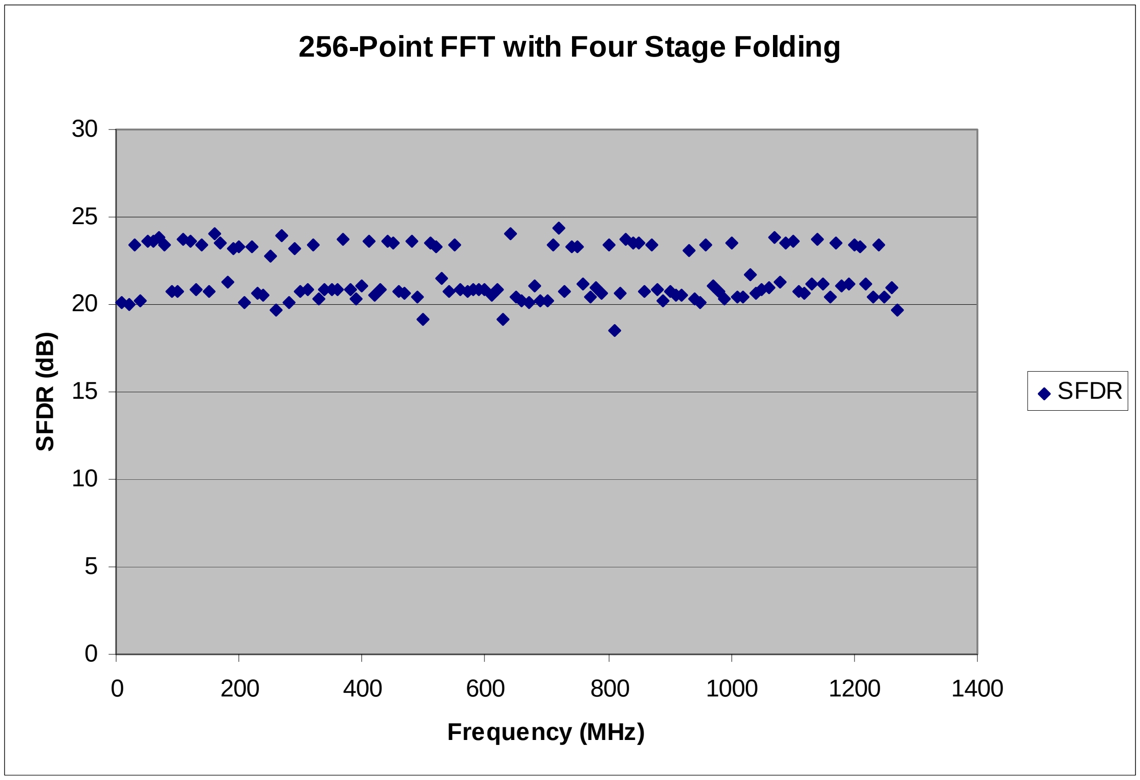


Figure 5.8: 256-Point FFT Model, SFDR Distribution

5.2.1 Instantaneous Dynamic Range (IDR)

An adequate definition of a two-tone instantaneous dynamic range would be the ratio of the smallest signal that will cause a change in the presence of a larger signal within the time domain. The IDR mainly relates to a receiver's capability to detect two simultaneous signals of different amplitudes. To accomplish the two signal IDR analysis, the collected samples from the 10 or 20 MHz to 1.26 GHz range where approximated to the lowest detectable amplitude accompanied by a second strong signal with a fixed amplitude. To assure an accurate analysis of the IDR, two signals, 500 MHz and 800 MHz, where used to represent the strong signal response. An example of a two-tone frequency response conducted during the 128-point FFT verification can be seen in Figure 5.9.

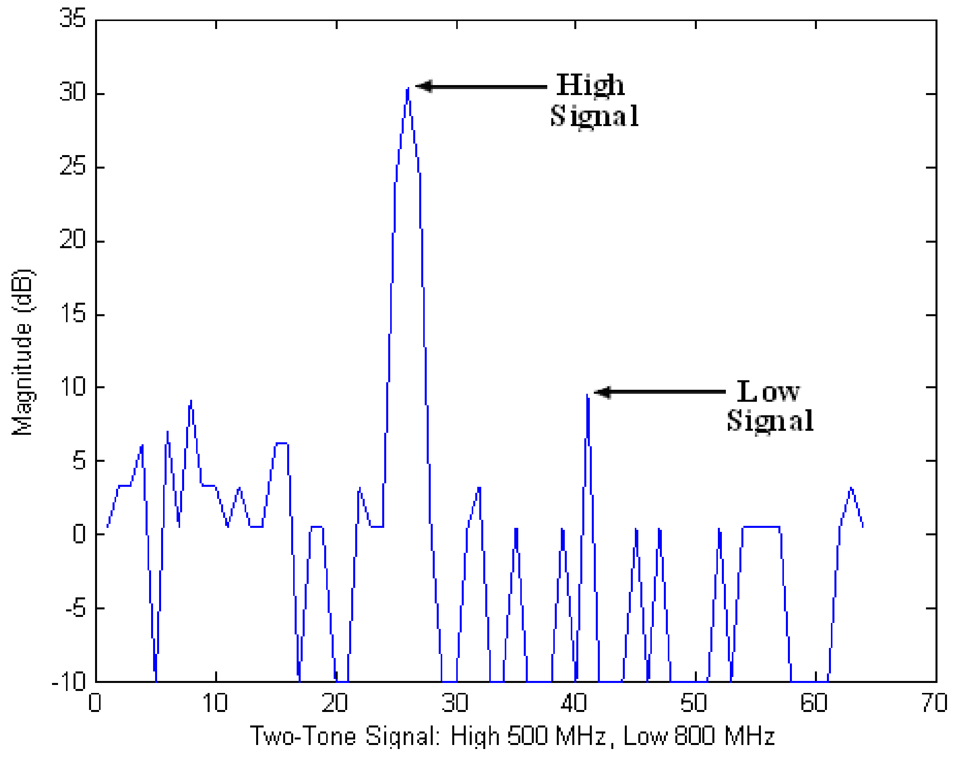


Figure 5.9: 128-Point FFT Model IDR, High Tone 500 MHz, Low Tone 800 MHz

This example shows the strong signal of 500 MHz, with a fixed amplitude of 128, and the lowest detectable signal of 800 MHz, with a verified low amplitude value of 7 in the XSG environment. The corresponding IDR was determined by taking the ratio in the time domain of the two simultaneous signals based off the acquired amplitude. For the two-tone frequency response shown, the instantaneous dynamic range was determined to be 24.08 dB.

In Figure 5.10, the same two signals where applied for the 256-point FFT model, which yields a verified low signal amplitude in XSG of 10 at 800 MHz and a dual-tone IDR of 22.14 dB. The distribution of the two-signal instantaneous dynamic range for both FFT models can be seen in Figures 5.11 and 5.12. There was an average IDR of 20.78 dB and 20.42 dB for the 128 and 256-point FFT's respectively.

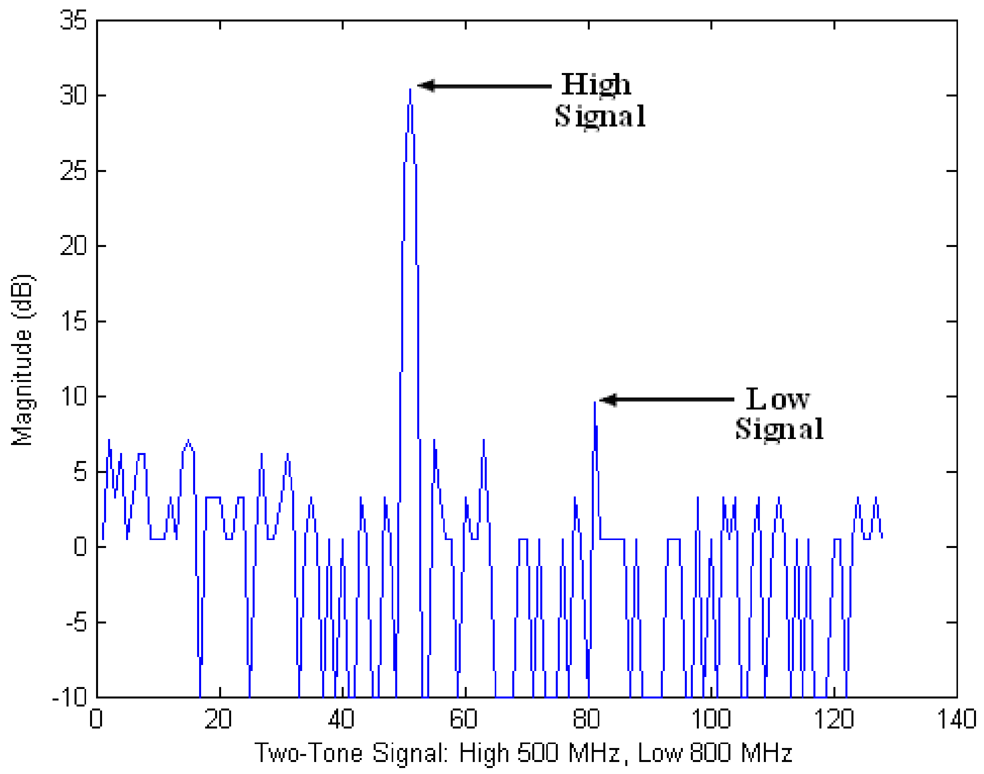


Figure 5.10: 256-Point FFT Model IDR, High Tone 500 MHz, Low Tone 800 MHz

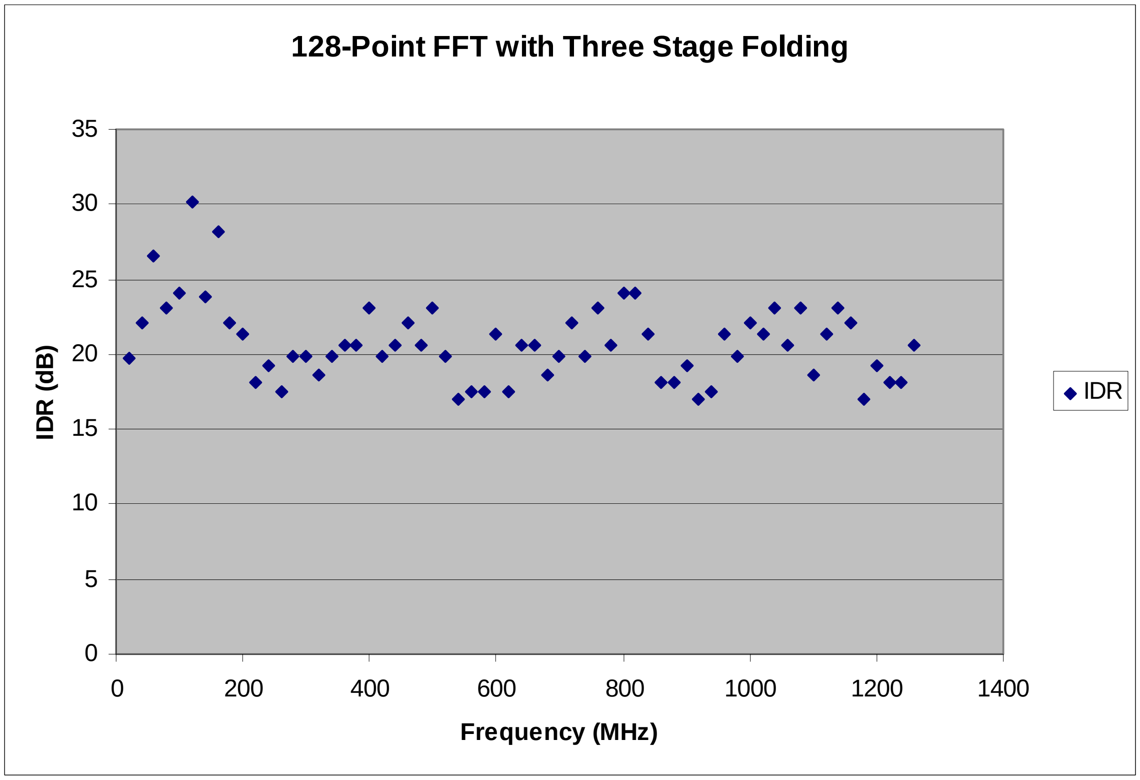


Figure 5.11: 128-Point FFT Model, IDR Distribution

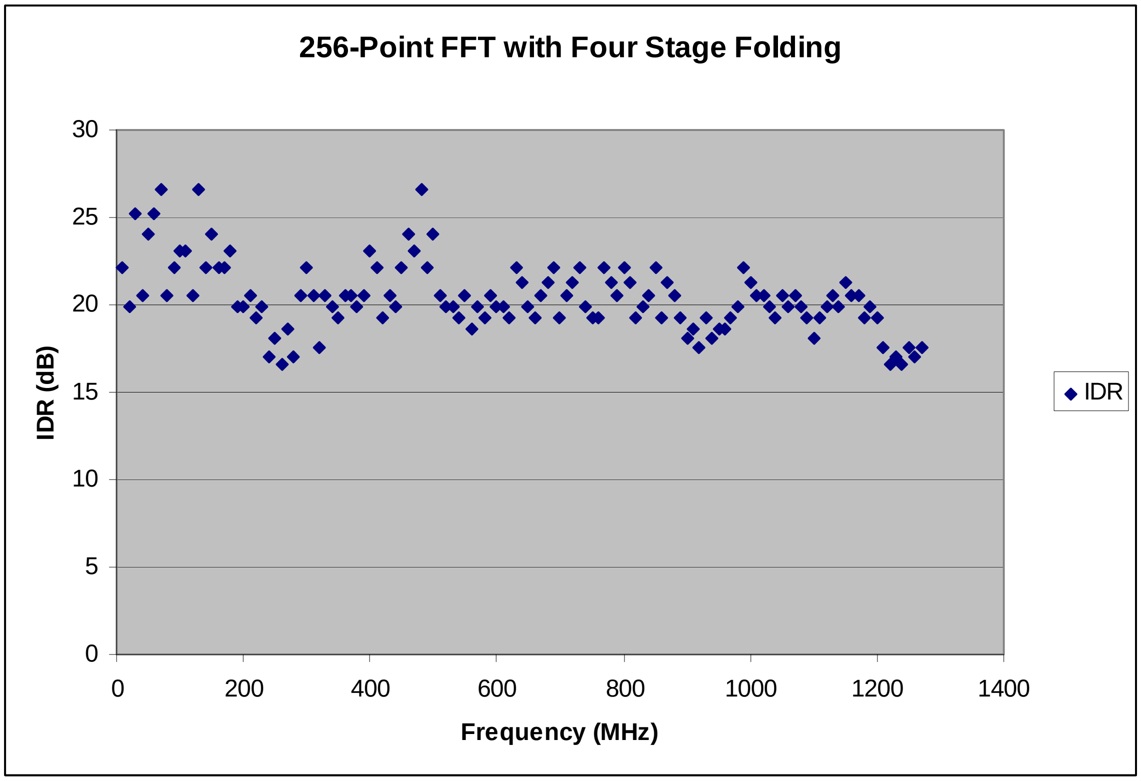


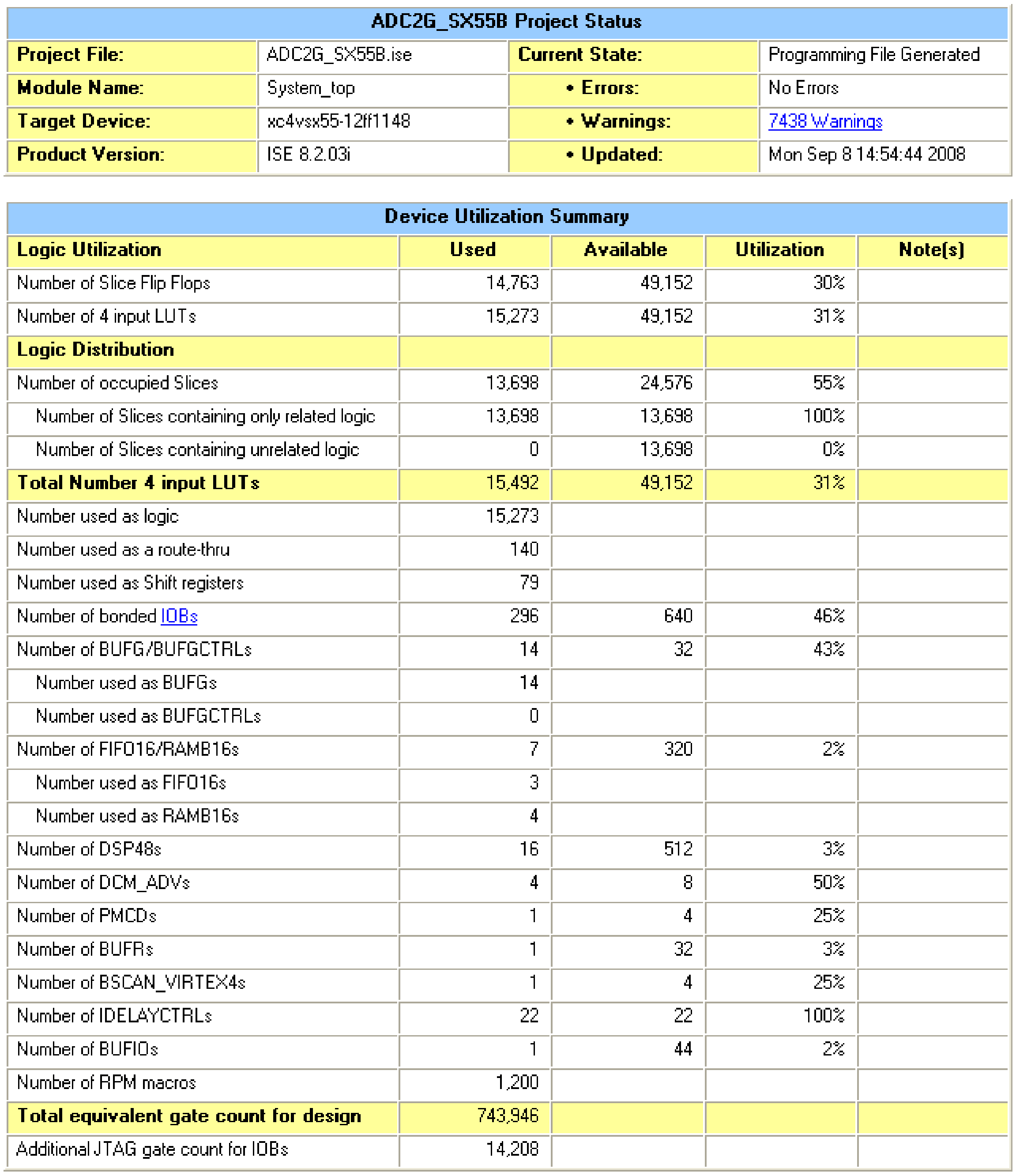
Figure 5.12: 256-Point FFT Model, IDR Distribution

5.3 Xilinx ISE Synthesis Results

With the verification of the hardware structure of the 128 fixed-point kernel DIF FFT, the finalized design is translated by System Generator to form subsystems and arbitrary hierarchies in a register transfer language (RTL), and placed in a user designated folder. The RTL consists of the HDL source code files and IP cores relevant to the completed design structure. These project files are then combined with an FPGA design package that was provided through Delphi, this design kit contains the basic codes used for demultiplexing the input ports of the FPGA implementation.

The combined profile is then synthesized, simulated, and implemented in the Xilinx ISE Project Navigator tool. The device utilization summary obtained through synthesis can be viewed in Figure 5.13. The logic distribution shows the percentage of the FPGA slices consumed by the 128-point FFT model. Of the 24,576 available slices on the Xilinx Virtex-IV FPGA, model XC4VSX55, the FFT consumes around 47%. The Delphi ADC3255 design package utilizes the other 8% of the 13,698 occupied slices. Also included at the end of the summary provided is the total equivalent gate count for the design, which came out around 846,181 with 22,114 of those being contributed by the four input look-up tables. The number of look-up tables available is 49,152 with the FFT using up around 41% and the design kit consuming the remaining 3%.

The device utilization layout of the complete floorplan on the Xilinx Virtex-IV FPGA can be viewed in Figure 5.14. The design hierarchy of the 128 fixed-point DIF FFT was incorporated into the USER_APP System_IF source code of the Delphi design package shown in the legend at the top right.



ADC2G_SX55B Project Status			
Project File:	ADC2G_SX55B.ise	**Current State:**	Programming File Generated
Module Name:	System_top	**• Errors:**	No Errors
Target Device:	xc4vsx55-12ff1148	**• Warnings:**	7438 Warnings
Product Version:	ISE 8.2.03i	**• Updated:**	Mon Sep 8 14:54:44 2008

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	14,763	49,152	30%	
Number of 4 input LUTs	15,273	49,152	31%	
Logic Distribution				
Number of occupied Slices	13,698	24,576	55%	
Number of Slices containing only related logic	13,698	13,698	100%	
Number of Slices containing unrelated logic	0	13,698	0%	
Total Number 4 input LUTs	15,492	49,152	31%	
Number used as logic	15,273			
Number used as a route-thru	140			
Number used as Shift registers	79			
Number of bonded IOBs	296	640	46%	
Number of BUFG/BUFGCTRLs	14	32	43%	
Number used as BUFGs	14			
Number used as BUFGCTRLs	0			
Number of FIFO16/RAMB16s	7	320	2%	
Number used as FIFO16s	3			
Number used as RAMB16s	4			
Number of DSP48s	16	512	3%	
Number of DCM_ADVs	4	8	50%	
Number of PMCDs	1	4	25%	
Number of BUFRs	1	32	3%	
Number of BSCAN_VIRTEX4s	1	4	25%	
Number of IDELAYCTRLs	22	22	100%	
Number of BUFIOs	1	44	2%	
Number of RPM macros	1,200			
Total equivalent gate count for design	743,946			
Additional JTAG gate count for IOBs	14,208			

Figure 5.13: Device Utilization Summary Generated by Xilinx ISE 8.2i

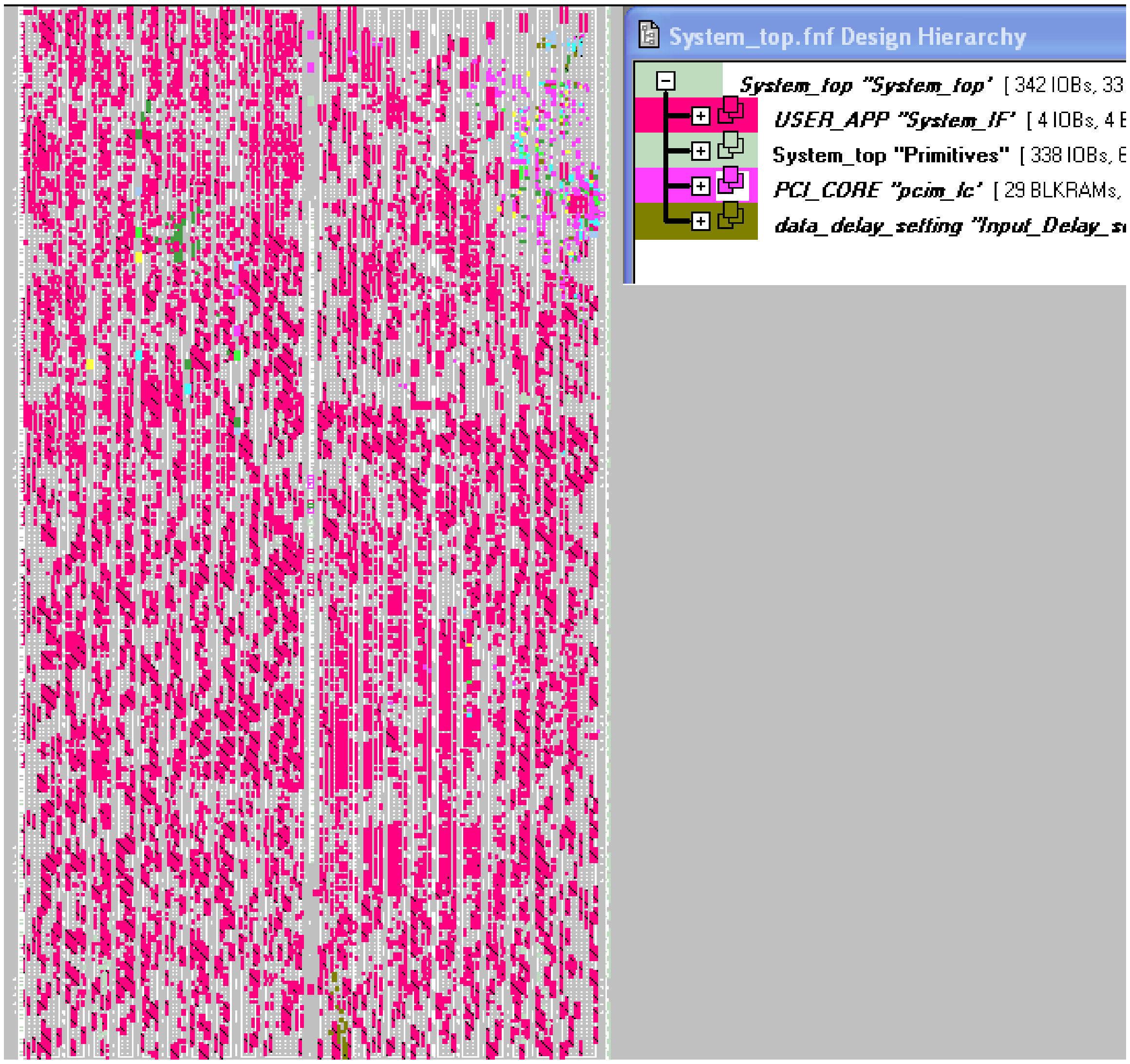

System_top.fnf Design Hierarchy
System_top "System_top" [342 IOBs, 33
USER_APP "System_IF" [4 IOBs, 4 E
System_top "Primitives" [338 IOBs, E
PCI_CORE "pcim_lc" [29 BLKRAMs,
data_delay_setting "Input_Delay_s

Figure 5.14: Xilinx Floorplanner of Implemented Design

5.4 FPGA Design Verification

The bitstream program file, which is the configuration data for the FPGA implementation, is generated through the synthesis of the completed FFT design. This stream of data is then introduced to the Xilinx ChipScope Pro 8.2i logic analyzer for verification. The diagram of the FPGA setup used to confirm the functionality and performance of the 128-point FFT is shown in Figure 5.15.

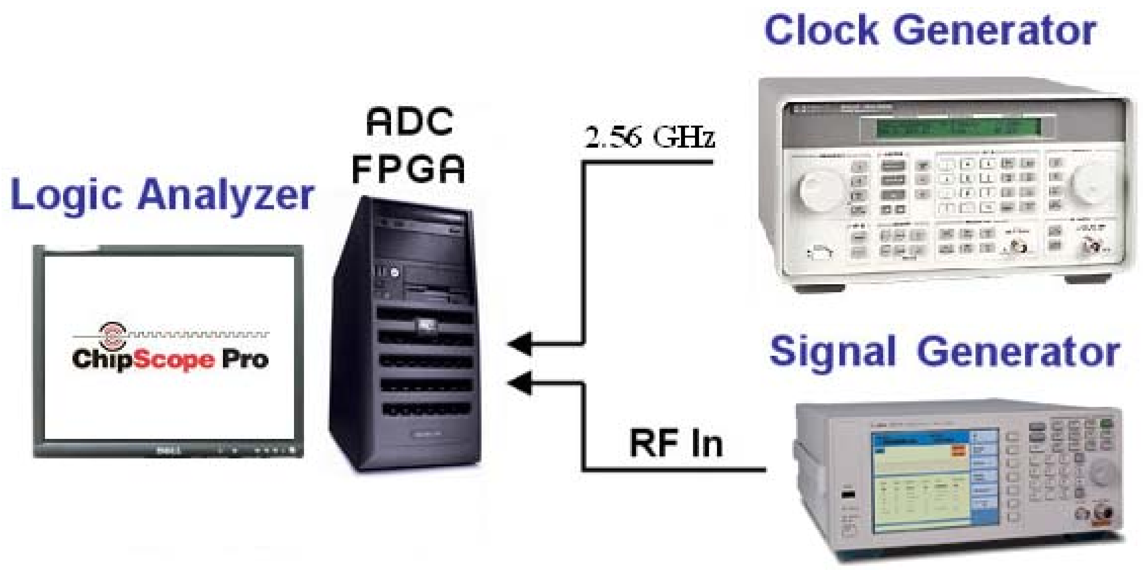


Figure 5.15: Xilinx Virtex-IV FPGA Test Setup

The Delphi ADC3255 and Xilinx Virtex-IV FPGA are clocked externally using an Agilent (Hewlett Packard) signal generator at an input sampling frequency of 2.56 GHz. The incoming RF input signal is provided using an Agilent RF signal generator at 20 MHz intervals for the desired range of 20 MHz to 1.26 GHz at a full scale amplitude value. The ChipScope Pro 8.2i logic analyzer was utilized to program the FPGA using the bitstream that was generated through synthesis on the Xilinx ISE 8.2i Project Navigator tool.

To demonstrate the functionality of the FFT design on the FPGA, several tests have been conducted and confirmed using the Xilinx ChipScope Pro verification tool. In Figures 5.16 and 5.17, the output waveforms where acquired for the RF signal generator input of 400 MHz and 800 MHz. Each of the tests shown below where validated using the clock generator at an input sampling frequency of 2.56 GHz with a full scale amplitude of -9.3 dBm.

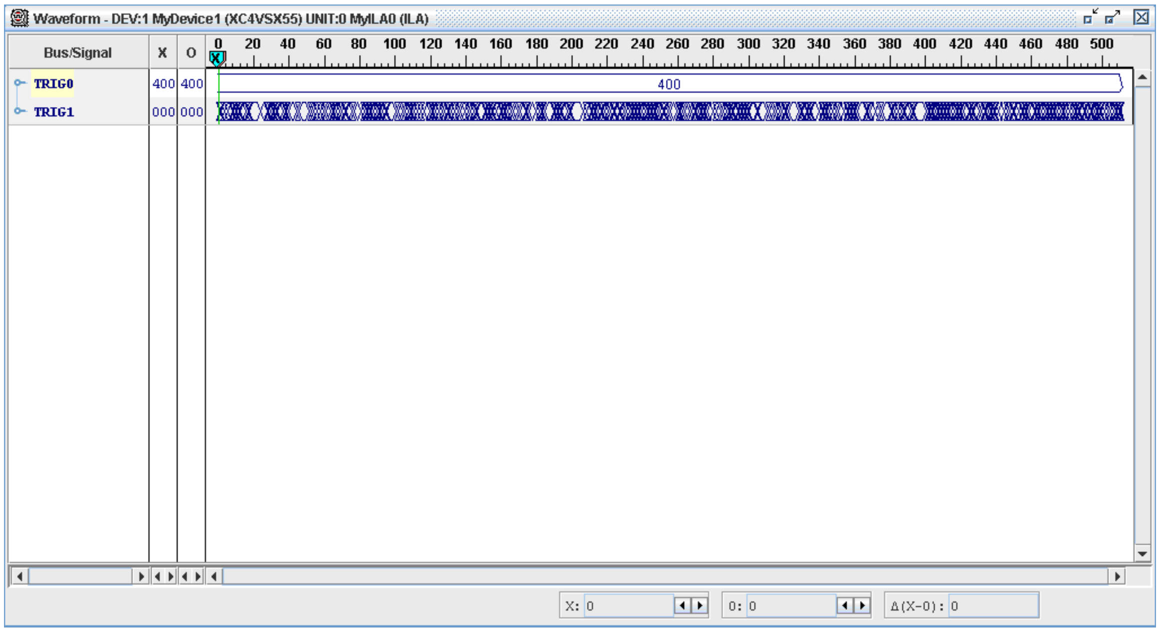


Figure 5.16: ChipScope Pro 8.2i Output for Test Frequency of 400 MHz

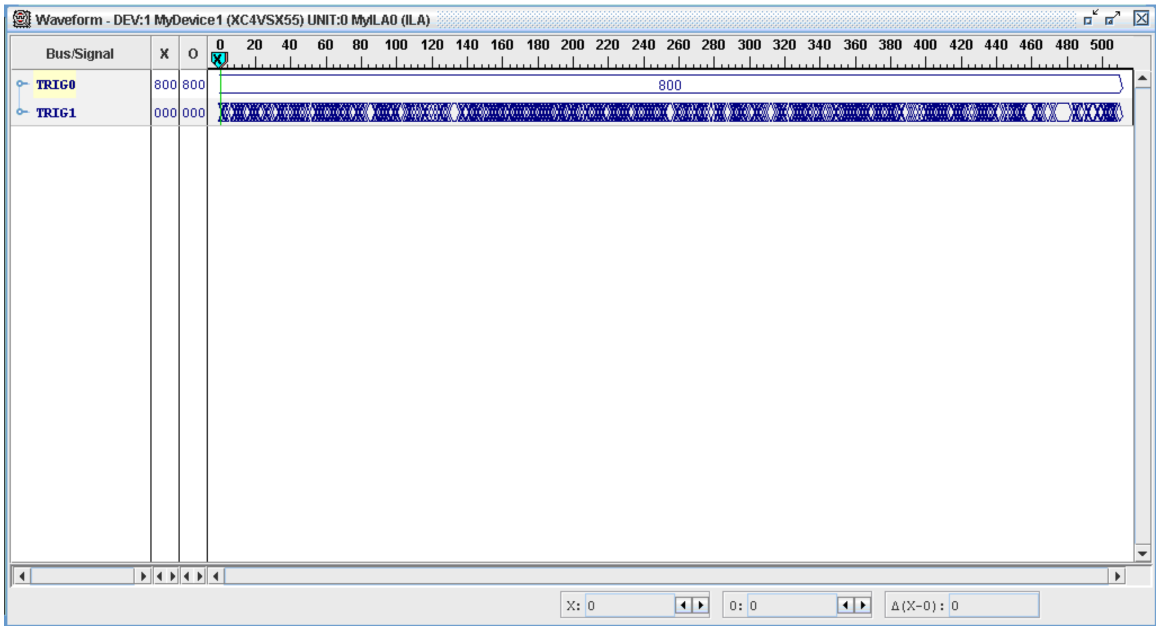


Figure 5.17: ChipScope Pro 8.2i Output for Test Frequency of 800 MHz

The testing for low signal detection was also conducted during the 128 fixed-point kernel FFT verification process. Utilizing the Agilent signal generator once more for an input sampling frequency of 2.56 GHz. The incoming RF signal is generated in 20 MHz intervals for the desired range of 20 MHz to 1.26 GHz at a fluctuating amplitude. The distribution of the lowest detectable signals obtained through the validation process can be seen in Figure 5.18 with the average low detection confirmed at around -17.6 dBm from a full scale amplitude of 3.6 dBm.

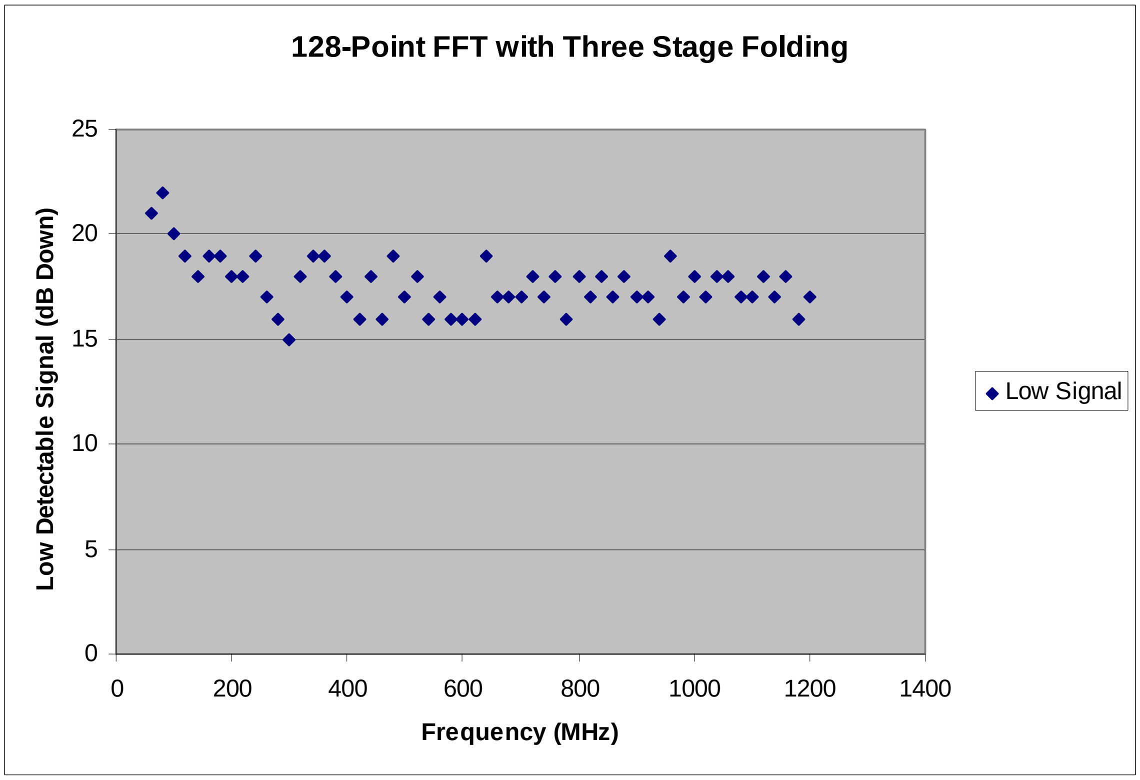


Figure 5.18: Lowest Detectable Signal Distribution

The resulting spur-free dynamic range (SFDR) was determined to be around 21.2 dB, which is very close to the SFDR using the ideal ADC which was found to be 22.78

dB for the 128-point FFT. These results were produced by applying the bit truncation to the eight most significant bit (MSB) values from the 10-bit ADC input. This showed a vast improvement over the previous bit truncation scheme which utilized the eight least significant bit (LSB) values from the ADC, producing a low signal detection of -22.7 dBm from a full scale amplitude of -9.3 dBm. The resulting SFDR was found to be around 13.4 dB, which is a difference of 7.8 dBm.

VI. CONCLUSION AND FUTURE WORK

6.1 Conclusion

The digital wideband receiver is a critical component in modern digital receivers. The receiver has the capability to expose and distinguish adverse signals contained within a large bandwidth (on the order of 1 GHz or more) of the radio frequency (RF) spectrum. The spectral estimator is a highly important design element imperative to the detection of the hostile signals within the intended bandwidth in order to fulfill mission requirements. This crucial element of an FFT-based digital receiver presented several challenges in the development of this research. The task at hand was to minimize the total hardware consumption contributed to the complex arithmetic operations while maintaining a high single signal spurious-free dynamic range (SFDR) and multi-tone instantaneous dynamic range (IDR).

In comparison with other work in the development of digital receiver's and FFT's [3-4]. The use of the kernel function approximation procedure showed a vast improvement in the dynamic range and hardware utilization, with the previously mentioned implementations using 20 and 12 kernel points. The 32 approximated functions contributed in this research, optimized the design further to produce less rounding error and contributed a more accurate result compared to the previous performance. The minimization of the hardware resources used where further reduced by employing a folding technique that reused a decimated DFT within the FFT structure.

This effective utilization of the hardware architecture showed a significant reduction in the consumed FPGA slices thus minimizing the total power consumed by the complete design.

For the 256 fixed-point kernel and FPGA-based 128 fixed-point kernel DIF FFT's completed with the methodology mentioned above, performed at an output throughput rate of 100 ns and 50 ns respectively. These high throughput rates performed better then the ideal Xilinx LogicCORE 256-point IP FFT's, which where around 4.83 to 5.12 us. With the folding technique applied, the deigns also showed a dramatic decrease in utilized slices on the Xilinx Virtex-IV FPGA, around 25% and 50% reduction in comparison to the previous unfolded models. In the ideal environment of the Xilinx System Generator application, the two designs showed a single signal SFDR of 22.78 dB, for the 128-point model, and 21.66 dB for the 256-point, with a multi-tone IDR of around 20.78 dB and 20.42 dB respectively. The verification of the FPGA-based 128-point FFT also showed a low detectable signal of around -17.60 dBm for the Xilinx Virtex-IV FPGA, utilizing an MSB 8-bit ADC the FFT had an SFDR performance of 21.2 dB at a full scale amplitude of 9.3 dBm.

6.2 Future Work

The goal of this research was to develop a computationally efficient hardware implementation and methodology for an FPGA-based fast Fourier transform (FFT), utilized in digital wideband receiver applications. At this point the contributed implementations have a great deal of potential to improve in performance and optimization. Several enhancements that could be applied would be to incorporate a method for optimal bit truncation, and an improved kernel approximation. Currently, the finalized FFT architectures employ a fixed 8-bit truncation scheme throughout that could contribute in a loss of information. Since the bit value continues to rise as the samples propagate through the FFT models, some data is truncated away. So a more optimal bit truncation scheme could help in the accuracy of the FFT computations.

An enhanced kernel function approximation would also be beneficial. An optimal number of kernel points would improve the hardware resources utilized and increase the performance of the spurious-free dynamic range (SFDR). Another improvement is contributed by the availability of resources pertaining to the FPGA. With the constant advancement of the field programmable gate array (FPGA) technology, the more complex FFT designs will have the opportunity to reach their full potential.

APPENDIX A.

The MATLAB source code shown within this appendix was written to aid in the design and implementation of the 128 and 256 fixed-point kernel fast Fourier transform architectures.

A.1 Fixed-Point Kernel Function Approximation

```
clc;
clear;
for i = 2;
  a = 1.4063;
  b(1) = 0;
  b(i) = b(i-1)+a;

  s_d(i) = sind(b(i));
  c_d(i) = cosd(b(i));

  s_r(i) = round(s_d(i));
  c_r(i) = round(c_d(i));

  x(i) = (s_d(i)/c_d(i));
  y(i) = atand(x(i));
end

  s_d(1) = sind(b(1));
  s_r(1) = round(s_d(1));
  c_d(1) = cosd(b(1));
  c_r(1) = round(c_d(1));
```

A.2 Unit Circle Approximation

```
function y = circle1(center,radius,nop,style)
% circle1 draws a circle with center defined as a
vector 'center'
```

```
% radius as a scalar 'radius'. 'nop' is the number of
points on the circle
% 'style' is the style of the point.
% Example to use: circle1([1 3],4,500, ':');
[m,n] = size(center);
if (~((m == 1) | (n == 1)) | (m == 1 & n == 1))
  error('Input must be a vector')
end
close all
x0=center(1);
y0=center(2);
t0=2*pi/nop;
axis equal
axis([x0-radius-1 x0+radius+1 y0-radius-1
y0+radius+1])
hold on
for i=1:nop+1
  pos1=radius*cos(t0*(i-1))+x0;
  pos2=radius*sin(t0*(i-1))+y0;
  plot(pos1,pos2,style);
end
```

REFERENCES

[1] Tsui, James Bao-Yen. *Microwave Receivers with Electronic Warfare Applications*, John Wiley & Sons, Inc., New York, NY, 1986.

[2] Despain, Alvin M. “Very Fast Fourier Transform Algorithms Hardware for Implementation,” *IEEE Trans. On Computers*, 28(5):333-341.1979.

[3] Patli, Sudhir. *Hardware Implementation of a Radix-4 Fast Fourier Transform Technique with Improved Two-Tone Resolution for Electronic Warfare Applications,* MS Thesis, Wright State University, Dayton, OH, August 2004.

[4] Tsui, James Bao-Yen. “Digital Techniques for Wideband Receivers,” *IEEE Tans. on Microwave Theory and Techniques*, Vol. 45, No. 12, December 1997.

[5] Sarathy, Vivek. *High Spurious-Free Dynamic Range Digital Wideband Receiver for Multiple Signal Detection and Tracking,* MS Thesis, Wright State University, Dayton, OH, December 2007.

[6] Buxa, Peter E. *Parameterizable Channelized Wideband Digital Receiver for High Update Rate*, MS Thesis, Wright State University, Dayton, OH, June 2007.

[7] Tsui, James Bao-Yen. *Digital Techniques for Wideband Receivers*, (Second Edition). Norwood, MA, Artech House, Inc., 2001.

[8] Xilinx System Generator v2.1 for Simulink User’s Guide. Online, www.mathworks.com/applications/dsp_comm/xilinx_ref/guide.pdf

[9] Spezio, Anthony E. “Electronic Warfare Systems,” *IEEE Transactions on Microwave Theory and Techniques*, 50, No. 3: 633-644, March 2002.

[10] Harris F. J., Dick C., and Rice M. “Digital Receivers and Transmitters Using Polyphase Filter Banks for Wireless Communications,” *IEEE Transactions on Microwave Theory and Techniques*, 51:1395-1409, April 2003.

[11] Xilinx Virtex-4 FPGA Overview. Online, www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex4/overview.htm

[12] Delphi Engineering Group, Inc., ADC3255 User Manual, v1.10, 2006.

[13] Delphi Engineering Group, Inc., Online, www.delphieng.com/adc3255.asp

[14] Fowler, Kim. “Analog-to-Digital Conversion in Real-Time Systems,” *IEEE Instrumentation and Measurement Magazine*, September 2003.

[15] Lathi, Bhagwandas Pannalai. *Signal Processing and Linear Systems*, Oxford University Press US, 1998.

[16] Cooley, James W. and John W. Tukey. “An Algorithm for the Machine Calculation of Complex Fourier Series,” *Mathematics Of Computation*, volume 19, pg. 297-301, April 1965.

[17] Baas, Bevan M. *An Approach to Low-Power, High-Performance, Fast Fourier Transform Processor Design*, PhD. Dissertation, Stanford University, February 1999.

[18] Oppenheim, A. V., R. W. Schafer, and J. R. Buck. *Discrete-Time Signal Processing*, (Second Edition), Upper Saddle River, NJ: Prentice Hall, Inc., 1999.

[19] Zolzer, Udo, Xavier Amatriain, and Daniel Arfib. *DAFX: Digital Audio Effects*, John Wiley and Sons, Inc., 2002.

[20] George, Kiranraj. *Design and Performance Evaluation of 1 Giga Hertz Wideband Digital Receiver*, PhD. Dissertation, Wright State University, June 2007.

[21] D. Pok, C.-I. H. Chen, J. Schamus, C. Montgomery and J. B. Y. Tsui, “Chip Design for Monobit Receiver,” *IEEE Trans. Microwave Theory and Techniques*, vol. 45, no. 12, pp. 2283-2295, December 1997.

[22] Tsui, B. Y., John J. Schamus, William S. McCormick, and John M. Emmert, “Quadbit Kernel Function Algorithm and Receiver,” *United States Patent #6,690,315*, February 2004.

[23] Pirsch, Peter. *Architectures for Digital Signal Processing*, New York, Wiley, 1998.