AN EFFICIENT IMPLEMENTATION OF FAST FOURIER TRANSFORMATION ON XILINX FPGAS

Submitted by:

Nighat Jamil

Supervisor

Dr. Arshad Aziz

Thesis

Submitted to the Department of Electronic and Power Engineering

Pakistan Navy Engineering College (PNEC), Karachi

National University of Sciences and Technology, H-12, Islamabad

In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

With Specialization in Communication

March 2012

AN EFFICIENT IMPLEMENTATION OF FAST FOURIER TRANSFORMATION ON XILINX FPGAS

Submitted by:
Nighat Jamil

Supervised by:
Dr. Arshad Aziz

THESIS

Submitted to:

Department of Electronics and Power Engineering,

Pakistan Navy Engineering College Karachi,

National University of Sciences and Technology, Islamabad

In fulfillment of requirements for the award of the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

With Specialization in Communication

March 2012

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ

I dedicate this thesis to my family especially my parents Jamil Ahmad and Nuzhat Jamil. Without their love and support, I would not have achieved this goal.

ACKNOWLEDGEMENTS

I am really thankful to Almighty Allah for giving me the strength and courage to complete the thesis and successfully achieving the results. I am heartily thankful to my supervisor, Dr. Arshad Aziz, whose encouragement, guidance and support from the initial to the final level enabled me to develop an understanding of the subject.

I thank all the Guidance Committee members Dr. Pervez Akhtar, Dr. Valiuddin Abbas and Dr. Athar Mahboob. I am also thankful to Cmdr (R) Riaz Mahmud, for always encouraging me in difficult phases of my research work.

I put forward my consents and regard to my family who supported me throughout my research phase especially my father-in-law, Dr. S.K. Hasnain whose guidance ad wisdom helped me a lot throughout the MS course.

A special expression of gratitude goes to my husband, Syed Furrukh Hasnain who has been very patient and supportive. During my daily work, I was blessed with the encouragement and love of my mother-in-law, Farzana Khursheed and my daughter, Amal Hasnain.

TABLE OF CONTENTS

List of Figures

List of Tables

ABSTRACT

Nighat Jamil

Under the supervision of Professor Dr. Arshad Aziz at the National University of Sciences and Technology.

Efficient algorithms for computing the Discrete Fourier Transform have enabled widespread access to Fourier transformation in numerous fields. One of those efficient algorithms is the fast Fourier Transform for computing the transformation of time-domain signals. Fast Fourier transform covers a wide span of diverse applications such as applied mechanics and structural modeling to bio medical engineering. In the signal processing arena, FFT has been widely used for signal recognition, estimation and spectral analysis. Similarly in the field of communication, FFT is used for echo cancellation, filtering, coding and compression. The ability to compute FFT in real time and with minimal hardware is the key to successful implementation of these complex systems.

The aim of this thesis is to optimize the area and throughput of FFT core for high performance. For high throughput, we have designed our architecture in such a way that it exploits the main features of Virtex-5 family that are high-tech XtremeDSP slices. Further the FPGA's dedicated embedded memories BRAM are ideal in implementing the FFT algorithm efficiently on FPGA.

Finally we have compared the performance of our design with other architectures. Our results show the optimized results.

Dr. Arshad Aziz

Chapter 1

Introduction

Fourier related transforms have central importance in the field of science, engineering and technology. It is signal processing tool that converts the time domain signal into frequency domain. The discrete Fourier transform enjoys extremely important position in the area of frequency analysis. The Fast Fourier transform takes the discrete signal in the time domain as an input and after processing it, the time domain signal is being transformed into discrete frequency domain signal. This conversion is important to perform the Fourier transform on Digital Signal Processor based system.

The Fast Fourier Transform is considered to be faster and efficient version of the Discrete Fourier Transform (DFT). The FFT utilizes algorithm to perform the same transformation of DFT, but in much less time. The event of progression of this algorithm accelerated the development of DSP.

Fast Fourier transform (FFT) was first given by Cooley and Tukey in 1965 which is a discrete Fourier transform algorithm which reduces the number of computations needed for N points from $2N^2$ to 2NlogN, where log is the base-2 logarithm [15]. As Fast Fourier is considered to be an efficient algorithm to compute discrete Fourier transforms, it reduces the computational complexity to O (N log N) from O (N^2). FFT is also used in the real world scenarios to process any given set of data from its time domain to frequency domain components. A simple modification in the algorithm can be used to evaluate the inverse Fourier transform i.e. reconstructing a signal from frequency to time domain. The importance of FFT lies in various applications that include digital signal processing, digital image processing, filter analysis, solving partial differential equations and quick multiplication of large integers.

Apart from this, the Fourier transform imposes its application in the world of physical sciences as well. With the passage of time, Fourier series and transform are making remarkable achievements in engineering, physics, applied mathematics, and chemistry.

Field Programmable Gate Arrays (FPGAs) are integrated circuits containing programmable logic components and interconnectors that used to create complex logic functions. The reconfiguration ability of FPGA enables the user to use it as an effective solution as compared to ASIC. They are characterized by the unique capability to be updated or changed depending on current requirements.

In this thesis we have tried to implement the FFT using Xilinx FPGAs: Spartan 3E and Virtex 5. We have utilized the main features of FPGA to implement FFT efficiently regarding the speed and resources. The algorithm which is being used for the fast Fourier transform is given by Cooley and Tukey. This is considered to be computationally efficient method and is widely used. We have compared the results of various devices of FPGAs with recent published papers. FPGAs are found to be ideal for the fast Fourier Transform due to their property of reconfiguration and whenever its performance is compared with ASICs, FPGA always provide time and cost effective solutions.

The thesis has been organized as follows: Chapter 2 gives the introduction of Fast Fourier Transform. In this chapter the number of calculations required to compute the fast Fourier Transform has been explained. Number of additions and multiplications required for the manipulation of various points FFT has also been given.

Chapter 3 presents the details of architectural view of those hardware devices that are used in our thesis to implement the Fourier Transform effectively. The structural description with build-in features of Spartan 3E and Virtex 5 are explained in this chapter highlighting the high-tech XtremeDSP slices structure of Viretx-5.

Chapter 4 outlines our approach to implement the FFT in Xilinx FPGAs. Our design has been discussed in this chapter and highlights the better throughput and low area implementations by utilizing the special feature of FPGA. Different parameters of optimization, that formulated the efficient implementation of FFT on FPGA, have also been discussed.

Chapter 5 presents all of our implementation results and its comparison with other FPGA implementations of FFT are given in the Chapter 5.The results have also been compared within two families of Xilinx FPGAs. Conclusion with further future discussion is briefed in Chapter 6.

Chapter 2

Fast Fourier Transform

2.1 Introduction

The world of signal processing widely utilizes Discrete Fourier transforms which is used to process the information stored in computers as well as from the real world applications. From solving partial differential equations to performing convolutions, they have covered a wide area of application. Several ways has been suggested to calculate the Discrete Fourier Transform (DFT) such as solving simultaneous linear equations or the correlation method. Many algorithms have been developed so far to implement the discrete Fourier algorithm efficiently on the software. The Fast Fourier Transform (FFT) is another method for calculating the DFT. While it produces the same result as the other approaches, it is incredibly more efficient, often reducing the computation time by hundreds.

FFT eliminates certain redundancies in the DFT by multiplying the input vector by fixed complex terms and thus reducing the number of computations (additions and multiplications) from the order of N^2 to the order of NlogN, N being the number of data points. Whenever there is larger value of N, it shows remarkable improvements.

2.2 Family of FFT Algorithms

The FFT algorithm family is very large encompassing different types of algorithms for different data formats:

- Algorithm for different radices such as radix-2, radix-4 and mixed radices.
- Decimation-In-Time and Decimation-In-Frequency algorithms.

- Real and complex algorithms.

Every algorithm follows the same basic principle of FFT of divides and conquers that is decomposing the large point FFT into smaller DFTs. We have implemented the same logic of divide and conquer rule in our algorithm. The FFT becomes faster not only due to the preceding dividing process but also due to the symmetry property of twiddle factor. The main interesting property of some twiddle factors is having its real or imaginary parts equal to ± 1or 0 and these factors don't require any multiplication [5].

2.3 Decimation in time Radix-2 FFT Algorithm

This section of the chapter describes how the decimation in time makes the DFT algorithm efficient when the sequence is of the length of power of 2(i.e. N=2^M where M is a positive integer) decimation in time follows the same technique of splitting x (n) into smaller sequence.

The N-point DFT of an N point sequence x (n) is given by

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}$$

Because x (n) may be either real or complex, X (k) is evaluated on the order of N complex multiplications and additions for each value of k. Hence computing N point DFT requires N^2 complex multiplications and additions.

In 1965, an algorithm was developed by Cooley and Tukey for the Radix-2 decimation in time FFT. The basic strategy that is used in FFT algorithm is one of the 'divide and conquer' method that decomposes the N point DFT into successively smaller

DFTs. Suppose that the length of x(n) is even (i.e. N is divisible by 2) and is split into two sequences, each of length N/2,one is odd-indexed and one is even-indexed [10].

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

Since $W_N^{2mk} = W_{N/2}^{mk}$; the above equation can be written as

$$X(k) = \sum_{n=0}^{N-1} x(n)W_N^{nk} = \sum_{m=0}^{(N/2)-1} x_e(2m)W_{N/2}^{mk} + W_N^k \sum_{m=0}^{(N/2)-1} x_o(2m)W_N^{(2m+1)k}$$

The computation of each term requires 2x $(N/2)^2$ multiply and add operations. In addition to that, the computation of $W_N^k \sum_{m=0}^{(N/2)-1} x_o(2m)$ requires another (N/2) multiply and add operations. The result is $(N^2+N)/2$ operations, which is almost a 50% reduction over the N- point DFT if N is large. This process is known to be decimation in time because in this time samples are alternately rearranged and a Radix-2 algorithm mainly because it has samples which are power of two and there are two groups in the decomposition [6].

Figure 2.1 illustrates the basic computation of an N point FFT through a signal flow graph which is known to be a butterfly diagram because of its shape.

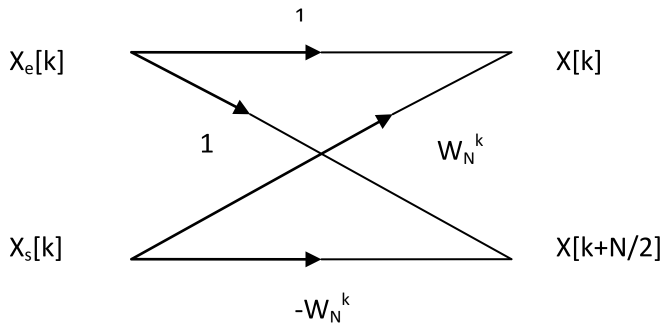

Figure 2.1 Basic Butterfly Diagram

The Radix-2 decimation in time saves computational time by recursively performing the two lengths $\frac{N}{2}$ DFTs. The algorithm divides the length into halves and performs DFT until a shorter DFTs is reached which is of length 2. [6].

The Table 2.1 shows the number of operations with the Radix-2 FFT as compared with direct computation of DFT.

Table 2.1 Computations of DFT versus FFT

Number of points	**Direct computation of DFT**		**Computations using FFT**	
	Complex multiplications	**Complex additions**	**Complex multiplications**	**Complex additions**
N	N^2	$N^2 - N$	$(N/2)\log N$	$N \log N$
4	16	12	4	8
16	256	240	32	64
64	4096	4032	192	384
256	65536	65280	1024	2048
1024	1048576	1047552	5120	10240

2.4 Decimation in time Radix-4 FFT Algorithm

When the number of data points *N* in the DFT is a power of 4, Radix-2 algorithm can readily be used for the computation. However when the number of FFT points is large, employing the Radix-4 FFT algorithm can be more efficient in terms of computation.

Like Radix-2 algorithm, Radix-4 decimation-in-time and decimation-in-frequency fast Fourier transforms are computationally fast by dividing the large number of samples into smaller one and performing the transform on these groups. The results of smaller and intermediate computations are then reused to perform multiple DFT frequency outputs. The Radix-4 decimation-in-time algorithm rearranges the discrete Fourier transform (DFT) equation into four parts: sums over all groups of every fourth discrete-time index n. If x (n) is the given input sequence then the four subsequences are x(4n), x(4n+1), x(4n+2), x(4n+3), n = 0, 1, ... , N/4-1. Thus due to the division of input vectors in a group of four and rearranging of samples in an alternate fashion the algorithm is known as decimation in time Radix-4 FFT algorithm. [7]

The outputs of the shorter FFTs are reused to compute many outputs; this operation greatly reduces the total computational cost. The Radix-4 decimation-in-frequency FFT groups every fourth output sample into shorter-length DFTs to save computations. The Radix-4 FFTs require only 75% of the total complex multiplication as the Radix-2 FFTs i.e. number of complex multiplication is given by $\frac{3}{8}N\log_2 N$ where as the number of complex additions remains same as Radix-2 i.e $N\log_2 N$.

Chapter 3
Field Programmable Gate Array

3.1 Introduction

Fast Fourier Transform algorithm can be implemented in various software like MATLAB and C languages as well its hardware implementations utilize hardware description languages such as Verilog HDL and VHDL using two major application environments: Application Specific Integrated Circuits (ASICs) and Field Programmable Gate Arrays (FPGA).

Most digital signal processing algorithms are based mainly on multiply and accumulate (MAC) operations. Field Programmable Gate Arrays can easily be used to implement MAC cells. As FPGA technology can provide more bandwidth through multiple MAC cells on one chip, it is highly effective to implement in various high-bandwidth signal-processing applications such as wireless, multimedia, or satellite transmission [12].

Field Programmable Gate Arrays (FPGAs) are a member of devices called field-programmable logic (FPL). These logics are defined as programmable devices containing repeated fields of small logic blocks and elements.

A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by the customer or designer after manufacturing. The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC).Thus the ability of FPGA to update the functionality after shipping, partial re-configuration of the portion of the design and the low non-recurring engineering costs relative to an ASIC design , offer advantages for many applications. Latest versions of FPGA have added special features which accelerate the performance of digital system.

Almost all of the FPGAs in Xilinx family consist of configurable logic blocks (CLBs), the I/O pads, LUTs, RAMs, embedded multipliers, DCM etc. The configurable logic block is the basic logic unit in FPGA having a switch matrix with either 4 or 6 inputs.

3.2 Basic FPGA Features

Modern day FPGAs have evolved many basic features that advanced the capabilities of FPGA. They have incorporated basic functionality such as Random Access Memory, clock management and other signal processing aspects. Next section explains the basic components in an FPGA.

3.2.1 Configurable Logic Block (CLBs)

The CLB is considered to be the basic logic unit in an FPGA. There are different numbers of configurable logic blocks in every device of FPGA featuring different properties. However, there are some characteristics like configurable switch matrix having 4 or 6 inputs, flip-flops and some selection circuitry (MUX, etc) are available in every device. The switch matrix is highly flexible and configurable.

3.2.2 Interconnect

The flexible interconnect are another main feature FPGA whose basic purpose is to routes the signals between Configurable logic blocks and among I/Os. The user designs the routing task in such a way that it remains hidden to him. Once designed efficiently it reduces the design complexity of the architecture. Thus significantly reducing design complexity.

3.2.3 SelectIO (IOBs)

Modern FPGAs supports several I/O standards that provide the ideal interface bridge in a system.

3.2.4 Memory

Embedded Block RAM memory is available in most FPGAs, which allows on-chip memory. In Xilinx FPGAs there is a provision of on-chip memory up to 10 Mbits in 36 kbit blocks. These memory chips can also be used to support true dual-port operation. The 16 x 1 bit distributed RAM and ROM configuration can also be achieved by using slice LUTs.

3.2.5 Complete Clock Management

All Xilinx FPGA provides the feature of Digital clock management. Xilinx FPGAs have built in digital clock management system. It also offers phase-looped locking that reduces the jitter and filtering with precise clock signals. [11]

3.3 Overview of Spartan 3E Family

3.3.1 Introduction

The Spartan-3E family of Field-Programmable Gate Arrays (FPGAs) is specific family which fulfills the demand of high volume requirement in the market. Electronic applications with cost-sensitive consumer ship have higher demand of this reconfigurable platform whose system gates range from 100,000 to 1.6 million.

The Spartan-3E family is considered to an alternative to ASICs. The major drawback of conventional ASIC is high initial cost, the lengthy development cycles and the inherent inflexibility which is avoided in FPGA. It is impossible to upgrade the program design in ASIC without replacing any hardware component, whereas this facility is available in FPGA.

3.3.2 Spartan-3E Architectural Overview

The Spartan-3E family architecture consists of five fundamental programmable functional elements that are very useful in implementing the FFT in Xilinx:

• **Configurable Logic Blocks (CLBs)**: This element is responsible for performing different variety of logical function as well as to store the data. It contains various look up tables which implements logic plus various storage elements used as flip-flops or latches.

• **Input/output Blocks (IOBs)**: This element is responsible to control the basic flow of data between internal logic of the device and the Input and output pins. With each IOB data flow can be directed in both the directions.

• **Block RAM**: The fundamental block of the device stores the data in the form of 18-Kbit dual-port blocks.

• **Multiplier Blocks:** This basic block performs the main operation that is involved in every signal processing i.e. it accepts the two 18-bit binary numbers as inputs and calculate the product of those two numbers.

• **Digital Clock Manager (DCM) Blocks**: This block provides the self-calibrating digital clock management system. The clock signals are required to be distributed, delayed, divided, multiplied and phase-shifted, all of these solutions are provided by this block. [3]

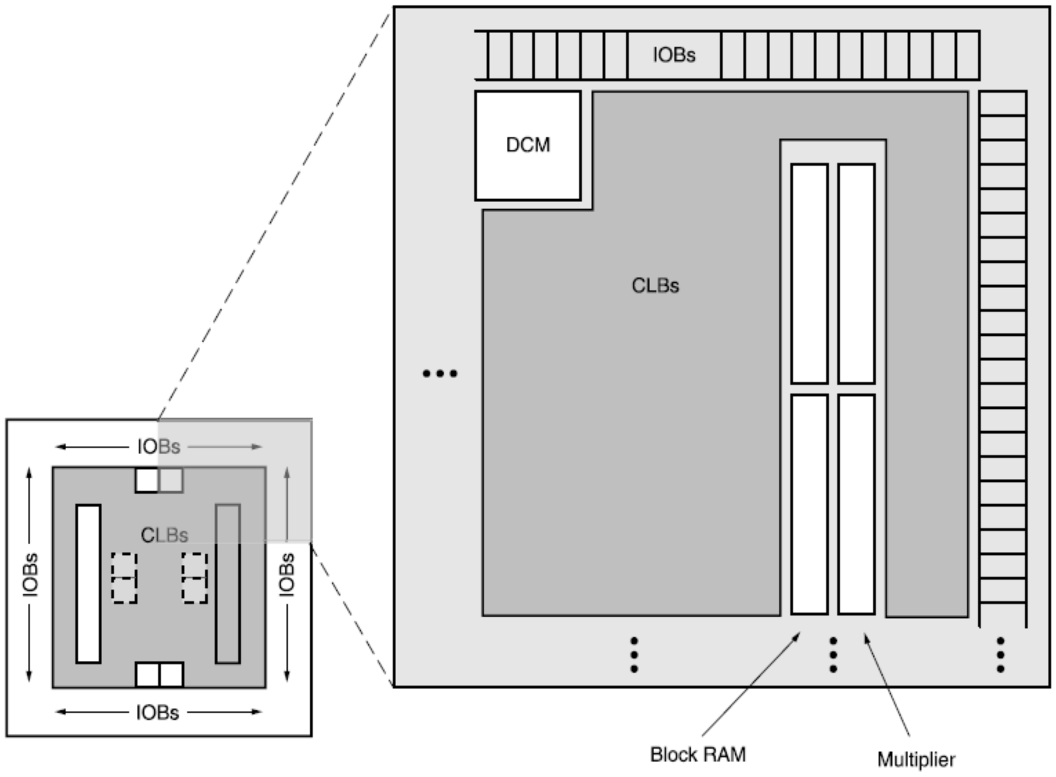

Figure 3.1 Spartan 3E Family Architecture [3].

3.4 Overview of Virtex-5 family:

3.4.1 Introduction

The Virtex-5 family provides the newest features which makes it one of the most powerful in the FPGA market. The Virtex-5 family comprises of five distinct subfamilies. Each family has different features providing specific and advanced logic design required for different application. The logic fabric of Virtex 5 is considered to be advanced giving high performance. Virtex 5 family encompasses several hard-IP system level blocks [4].

3.4.2 Virtex-5 Architectural overview

Following are the basic blocks of Virtex-5 architecture.

- **Input/output blocks:**

Virtex-5 have Input/output blocks which are the main building blocks and a source to provide the interface between package pins and the internal configurable logic. There are programmable input/output blocks which supports many important and leading-edge i/o standards.

- **Configurable Logic Blocks (CLBs)**

Like every Xilinx FPGA, Configurable Logic Blocks (CLBs) are considered to be the basic logic elements for Xilinx FPGAs. These CLBs have the ability to be configured as SRL32 shift register or distributed memory. These CLBs are based on the look up table with real 6-input. The performance and capabilities are incomparable with the previous versions of programmable logic.

- **Block RAM modules**

BRAM modules are main blocks of Virtex-5 architecture that can be cascaded together to form large memory blocks. They are flexible RAMs with 36 Kbit true dual ports. The built in option for programmable FIFO logic, that increases the utilization of the device, is due to these block RAMs. There are several designs in which there is a requirement of smaller RAM blocks; BRAM can be configured as two independent 18 Kbit true dual-port RAM blocks.

- **Embedded DSP48E slices**

The high performance ability of Virtex-5 is met by the cascadable embedded DSP48E slices. This ability is due to the presence of 25 x 18 two's complement multipliers and 48-bit adders. The same sized subtracter or accumulator is also present to perform many operations of signal processing. Performance ability is increased by the provision of parallel support of various DSP algorithms. Moreover, each DSP48E slice can be configured to perform bitwise logical functions.

- **Clock Management Tile (CMT)**

Every Virtex-5 FPGA has high-performance and flexible clocking system provided by the block known as Clock management tile. This tile comprises of two different types of self calibrating blocks i.e. two Digital Clock Manager (DCM) blocks which are fully configured as digital and one PLL block which is analog. The main purpose of these blocks is to provide delay compensation for clock distribution. [4]

Chapter 4

Our work: Design Implementation Details of FFT on Xilinx FPGA

4.1 Introduction

This chapter focuses on the design implementation of the FFT on Xilinx FPGA which is the main aim of our thesis. The key to successful implementation of Fast Fourier Transform is the ability to compute it in the real time with minimal hardware. Fast Fourier Transform has become almost ubiquitous and most important in high speed signal processing for recognition of signals and spectral analysis. When considering the various implementations, the FFT architecture should be chosen according to the specification and application which takes into account the execution speed hardware complexity and flexibility.

In our thesis, we have implemented the FFT through two different architectures on two different families of Xilinx i.e. Spartan 3E and Virtex5 FPGA and then compared the results within these two families and among previous work in terms of speed and resources. These two architectures are:

1. Radix- 4 FFT Architecture: this option loads and processes the input data vector separately and using Radix-4 algorithm, the FFT transformation is carried out through an iterative approach.
2. Radix-2 FFT Architecture: this option loads and processes data separately but uses a smaller butterfly of Radix-2. The transformation time for Radix-2 is longer than the Radix-4.

We have used Spartan 3E and Virtex-5 FPGA to analyze the FFT implementing operation. The results are verified at the MATLAB platform. However, we have utilized the high-tech feature available in Virtex-5 FPGAs which are the XtremeDSP slices to optimize the FFT in terms of speed and resources. It has been used to implement the Adders, subtractors, control unit, and complex multipliers. We have noticed that the provision of embedded DSP48E slices reduces overall power consumption and increases maximum frequency. It can implement 25x18-bit multipliers, with add, subtract, accumulate, and bit-wise logic.

4.2 Architectural Considerations

There are several keys to consider regarding the architecture before we had designed our structure for FPGA. The basic step is to select the radix according to the memory system available. There are many ways to structure the computation while implementing FFT. The radix of the operation determines the number of data to be combined at any stage.

In this section, we have discussed the two main architectures that are employed to implement the FFT algorithm i.e. Radix-2 and Radix-4 decompositions. For these architectures, the decimation-in-time (DIT) method is used which is elaborated in Chapter 2. Implementing the Radix-4 decimation in time algorithm for the N-point FFT, the process involves $\log_4$ (N) stages. In each stage there are N/4 Radix-4 butterflies. Similarly for N-point FFT using Radix-2 decimation in time algorithm, the transformation is being completed in $\log_2$ (N) stages, with every stage containing N/2 Radix-2 butterflies. Radix-2 architecture is being chosen because of its simplicity and smallest butterfly unit while radix-4 increases the complexity of operation but at the same time it reduces the total number of operations. [14]

4.2.1 Radix-4 FFT Architecture:

With the radix -4 algorithms, the designed architecture uses one Radix-4 butterfly processing engine. During the transformation process, the input is being loaded separately. It implies that the data input and the processing is not simultaneous.

Our designed algorithm has used Radix-4 architecture which has lower resources as only one butterfly is used but due to this low resource a longer transform is required to process the data. When the transformation begins, the data enters into the BRAM and get loaded. The data can only be unloaded once the computation has finished. [1].

The advantage of our algorithm is using the Radix-4 architecture is that it utilizes lower resources than the pipelined architecture which have several butterfly engines instead of one. We have used BRAM in our design to store the data and phase factors.

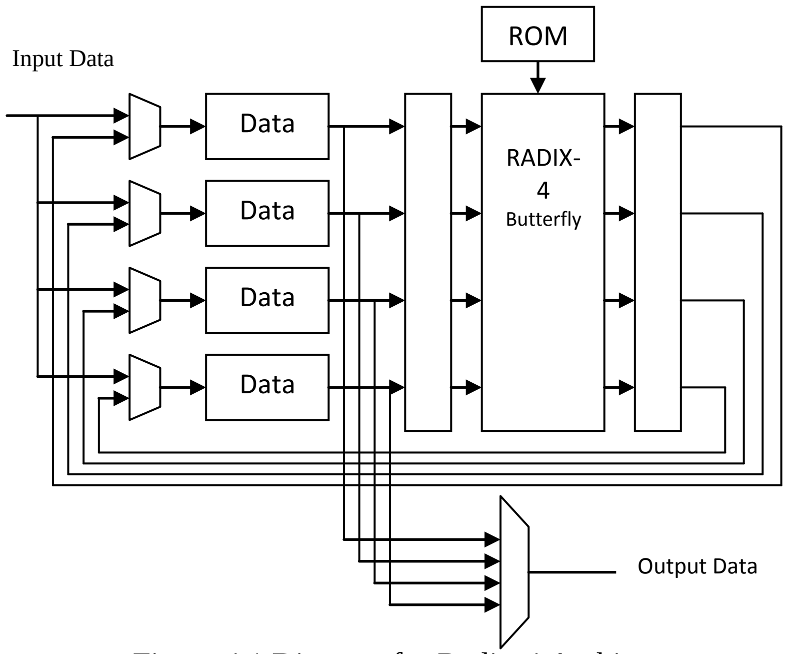

Figure 4.1 Diagram for Radix-4 Architecture.

4.2.2 Radix-2 FFT Architecture

With the Radix-2, our designed architecture utilizes one Radix-2 butterfly processing engine. Due to this hardware constraint the loading or unloading of the data is separate from executing the Fast Fourier transform similar to Radix-4 architecture. It implies that the data input and the processing are not simultaneous [1].

The architecture has lower resources as only one butterfly is used but due to this low resource a longer transform is required to process the data. But still the architecture is smaller than the Radix-4.

The basic radix-2 butterfly stage operates on a pair of results from the previous stage. Each basic butterfly computation consists of a single complex multiplication and two complex additions [14] as shown in the butterfly diagram in Figure 2.1.

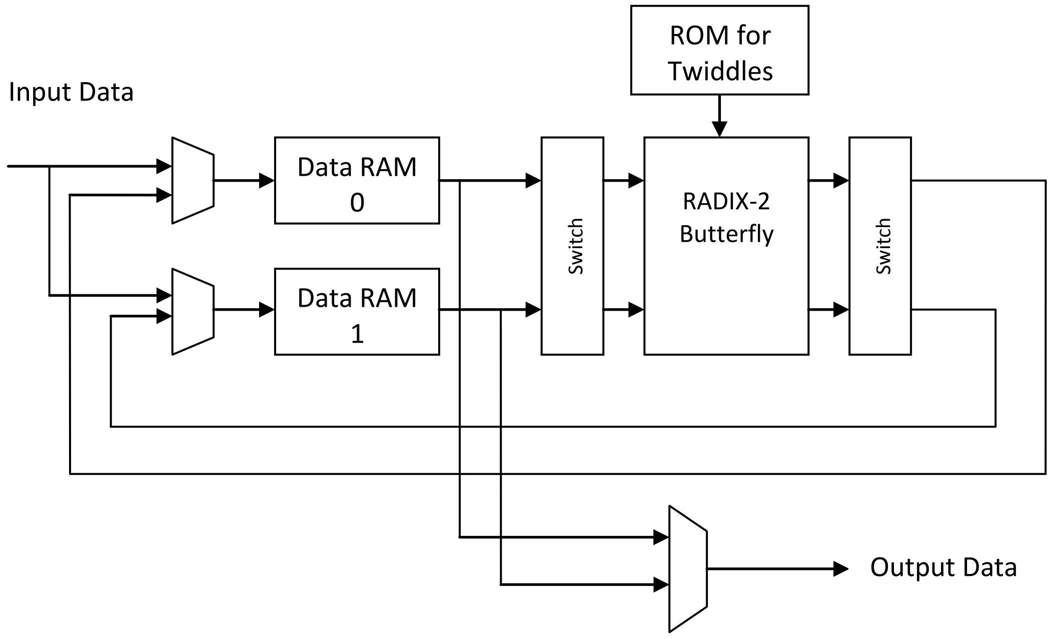

Figure 4.2 Diagram for Radix-2 Architecture

4.3 Optimization Parameters

We have exploited following properties of FFT and available features of Xilinx FPGAs to implement the FFT algorithm efficiently. Using these parameters effectively reduced the number of resources and increased the maximum frequency. These parameters are:

- Bit/digit reversal order
- Scheme for number representation
- RAM for storing the data inputs
- Phase factor width
- Complex Multipliers
    - CLB logic
    - 3-multiplier Structure
    - 4-multiplier Structure
- Butterfly arithmetic
    - CLB logic
    - Xtreme DSP slices

4.3.1 Bit /Digit reversal Order

The inherent nature of the decimation in time algorithm of FFT, the order of the output is in bit reversed order. In case of Radix-4 architecture, this reversal order is referred to as Digit reversal order. We have optimized our architecture by ordering the output data in the reversed order. The algorithm is designed in such a way that whenever the input data array is being in natural order the output is in the bit digit reversal order. Output data in the natural order imposes extra cost in terms of architecture i.e. additional numbers of BRAM. As the output data index is along with the output data so there is no requirement to place the output in the natural order.

4.3.2 Scheme for number representation:

The basic factor in optimizing the FFT in Xilinx FPGA is selecting the appropriate scheme for number representation. The storage requirement of Xilinx depends on the number representation. As the numbers are stored in the on chip memory therefore even if a few bits are saved in storing these numbers, minimal hardware resources will be required [13].

As the FPGA hardware is designed to store the input, intermediate values and final results therefore fixed point implementation is selected to keep the level of precision high. The speed of operation is observed to be high with fixed point number representation with utilization of few resources.

4.3.4 RAM for storing the data inputs:

The data inputs in the Radix-4 and Radix-2 architecture can be stored in RAM. As we have two options available therefore either block RAM or distributed RAM can be used for this purpose. We have stored our input data in the BRAM.

4.3.5 Complex Multipliers:

FFT requires complex multiplications in their operation. We have observed a tradeoff between resources and performance optimization.

- **CLB logic**: using these CLB logic slices all complex multipliers can be constructed. Applications that doesn't require high performance can easily make use of the CLB logic to execute the complex multiplication. Similarly, it can be used in those devices in which there are few or no XtremeDSP slices or 18 x 18 multiplier.

- **3-multiplier structure for optimization of the resources**: If the optimization of the resource is of prime importance then the three real multiply and five add or subtract structure is used. Multiplier structure is constructed using XtremeDSP slices or 18x18 multiplier. Although this structure reduces the number of DSP slices used but some slice logics are also used for this purpose. Some devices can utilize the XtremeDSP slice pre-adder which eliminates the requirement of extra logic cells.
- **4-multiplier structure for optimization of the performance:** in order to optimize the performance , the complex multiplier are designed in such a way that it utilizes the complex multipliers utilizing four real multiply and two add or subtract structure. It uses XtremeDSP slices or Multiplier 18x18. Highest performance of clock can be achieved using this structure but the resources are increased as dedicated multipliers are increased. Virtex-5 devices are equipped with XtremeDSP slices which are used to execute the addition and subtraction operations. Spartan 3E have Mult18x18s so the addition and subtraction operations are carried out through slice logic.

4.3.6 Butterfly Arithmetic: Implementing two different types of logic to our designs resulted in dynamic utilization of the resources.

- **CLB logic**: the arithmetic which is carried out inside the butterfly stages utilizes slice logic.

- **XtremeDSP Slices:** Butterfly arithmetic is carried out with XtremeDSP slices, the addition in all the butterfly stages of FFT is executed with these DSP slices inside Virtex-5.

4.4 FFT on Spartan-3E

We have started implemented our design specification in the Spartan-3E.It is one of those family of Field-Programmable Gate Arrays which provides high volume application in comparative low cost. We have selected this family as it reduces the cost per logic cell. The implementation in this device was easy and their results provided a benchmark for comparsion with other previous work.

4.4.1 Various point FFT on Spartan 3E:

Our designed architecture have been used to implement the various lengths FFT (i.e. 16, 64, 256, 512 and 1024 points FFT) on Spartan 3E, using Verilog and analyze the performance of this FPGA on speed and frequency and number of resources used. We have executed different number of points through different architectures. We have used the platform of Xilinx ISE project navigator 12.4 for the efficient implementation of FFT.

Starting with the relatively smaller number of FFT point, device xc3s1600e-5fg320 is selected for its execution. The summary after place and route shows that this device utilizes only 2% of the total flips flops available and 1% of the total LUTS available. This resource consumption is very low.Radix-2 architecture is being utilized for this assignment.

For the implementation of 16 point FFT, the Spartan device xc3s1600-4fg400 has been selected. When 16 point FFT with radix-2 FFT is being executed, it has been found that consuming approximately 181 MHz frequency, this utilizes 3% of the total available flip flops and 2% of the total available input LUTS.

Similarly increasing the number of points of FFT to 256 and using Radix-2 architecture, it has been observed that there was a slight decrease in the frequency but the utilization of the resources was still very low. Approximately 7-8% of the available flips flops and LUTS are being used in the computation.

FFT is found to be more efficient when the value of N is large. Now for the 512 and 1024 points FFT, xc3s1600e-4fg400 is being used.

We have summarized our results in the next chapter. The result explicitly shows that with Radix-4 FFT architecture, the frequency required to perform the implementing operation is 173.34 MHz with the utilization of almost 10% of the available slices only which clearly shows that Radix-4 provides faster computation even with the larger N.

4.5 FFT on Virtex-5

Another targeted family to implement various sizes FFT is Virtex-5. Virtex-5 FPGAs are equipped with most advance and high performance logics that efficiently meet the requirement to optimize the FFT algorithm.

4.5.1 XtremeDSP 48 Slice

The main advantage of using the high-tech Virtex-5 is using their embedded high performance XtremeDSP DSP48 slice that uses time-multiplexing method. Due to this time multiplexing it is convenient for the designer to implement multiple slower operations. Following are the advantages provided by the XtremeDSP slices:

- FPGAs have become more flexible and full device utilization.
- Different applications are being implemented with improved efficiency.
- Performance consumes less power.
- Maximum frequency is being increased with these slices.
- The main operation in DSP operation is multiply accumulate (MACC), multiply add, this feature is readily supported with XtremeDSP slices.
- Many DSP filters and complex arithmetic can be performed by using the slices alone. It supports multiple DSP48s slices which when cascaded can implement wide math functions.

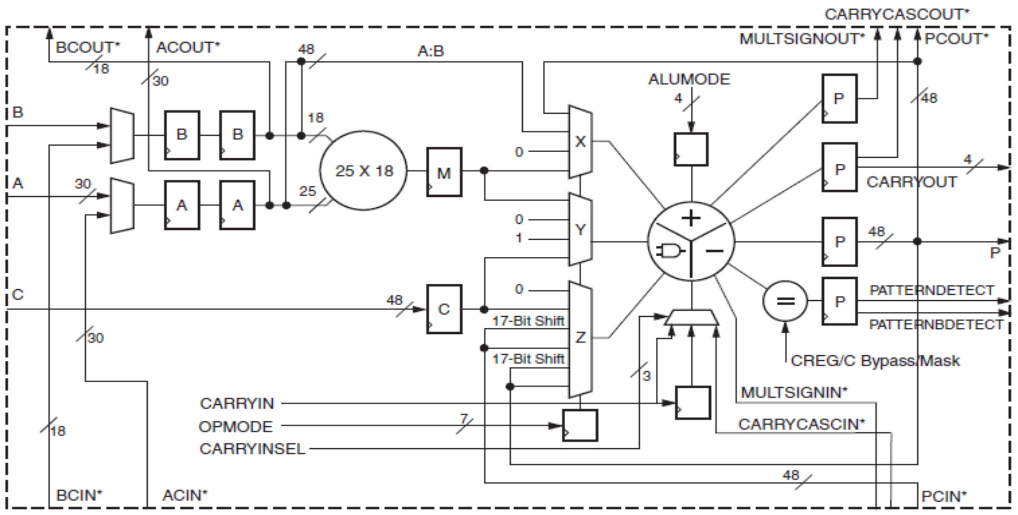

Figure 4.3 Structure of XtremeDSP Slice

With the invention of DSP48E slices and its availability in all Virtex-5 devices, many algorithms are accelerated. This enabled the integration of DSP at higher level with low power consumption. This efficiency was absent in the previous generation of Virtex devices.

This family supports several operating modes which are controlled dynamically. The different modes are required for different applications like multiplier, multiplier-accumulator, multiplier-adder/subtractor, three input adder, barrel shifter, wide bus multiplexers, wide counters, and comparators.

Virtex-5 family also has the ability to perform complex math efficiently and to implement high-performance filters with efficient adder-chain architecture. [8]

4.5.2 Various points FFT on Virtex-5

We have executed different points of FFT on Virtex-5 devices and compared the results with that of the Spartan 3E.with the availability of the XtremeDSP slices there was a drastic increase in the maximum frequency which was obtained after place and route step.

Using the device Xc5vlx110-3ff676, 64 points FFT is executed with Radix-2 architecture which processes the given size of input data array. Utilizing only 5 out of 64 available DSP48slices the maximum frequency increased from 181 MHz in Spartan 3E to 408MHz in Virtex 5. The same device of Virtex-5 is being used and Radix-2 architecture 256 and 512 points FFT is also being calculated.

Radix-4 architecture is used to calculate another vector of 256 points data with the Xc5vsx35t-3ff65. The absence of DSP48E slices in Spartan-3E was a basic drawback in the performance and hence resulted in low frequency operation although the numbers of points for Fourier transform were only 64, whereas using Virtex 5 with only 13.5% utilization of DSP slices gives the frequency of 409.165 MHz.

In next chapter, the results are tabulated giving analysis of different points FFT (i.e. 64,256,512, 256, and 1024) on Virtex-5 FPGA architectures.

Hence we can summarize that the 256-point FFT is implemented through both the architecture i.e. Radix-2 FFT and Radix-4 FFT. The results clearly show that the performance of Radix-4 FFT is better than the Radix-2 architecture in terms of frequency. But as there is a tradeoff between the resource and transformation time, the Radix-4 architecture required more slices that were used as LUTS and registers and hence more embedded DSP48 slices.

Chapter 5

Result Comparisons

In this thesis, we have presented the optimized approach to implement the FFT algorithm. The results have been tabulated and compared with previous conventional FPGA based implementation of FFT. Our design has successfully been implemented using Xilinx ISE 12.4 tool. All executions are first coded in Verilog and then synthesized using the Xilinx XST synthesis tool. Moreover Synthesis and place and route were achieved successfully for each design. Various sizes of input test vectors are given and the behavior is verified by ISE Simulator.

5.1 Comparison between Spartan 3E and Virtex-5

Different points FFT operations were implemented in Spartan3E and Virtex 5 FPGA. Table 5.1 shows the analysis of FFT of different points on Spartan 3E. The first architecture is Radix-2 in Spartan 3E. The absence of DSP48E slices resulted in low frequency operation in every experiment. We have tried to optimize the resources by using 3 multiplier structures. But the resource optimization had a constraint on the performance. We have observed that increasing the number of points decreases the maximum frequency.

Table 5.1 Implementation of various point FFT on Spartan-3E

	Device	FFT size	Architecture	No. of slices registers/ available	No. of slices LUTS/ available	Frequency
1	Xc3s1600e -5fq320	16	Radix-2	797/29504	565/29504	185.87MHz
2	Xc3s1600e -4fg400	64	Radix-2	758/29504	847/29504	181.15 MHz
3	Xc3s1600e -4fg400	256	Radix-4	2543/29504	2331/29504	166.97 MHz
4	Xc3s1600e -4fg400	512	Radix-4	2635/29504	2440/29504	169.86 MHz
5	Xc3s1600e -4fg400	1024	Radix-4	2353/21760	1659/21760	173.34MHz

Similarly, different points of FFT are implemented on the Virtex-5 device with different implementation option i.e. Radix-2 and Radix-4 and then analysis is carried out regarding the number of slices utilized as registers and LUTS. As devices of Virtex-5 are equipped with XtremeDSP slices then additional information of their utilization are given in the Table 5.2. 256 points FFT is being implemented by using Radix-4 architecture as well as radix-2 architecture. If we compare the resource utilization then Radix-4 has taken more slices and DSP at the cost of the increased frequency.

Table 5.2 Implementation of various point FFT on Virtex-5

	Device	**Size of FFT**	**Impleme ntation option**	**No. of slices registers/ available**	**No. of slices LUTS/avail able**	**No. of DSP 48Es**	**Frequen cy**
1	Xc5vlx110 -3ff676	64	Radix-2	921/69120	645/69120	5/64	408.16 MHz
2	Xc5vlx110 -3ff1760	256	Radix-2	1868/69210	1430/69120	12/64	444.44 MHz
3	Xc5vlx110 -3ff676	512	Radix-2	1844/69120	1256/69120	16/64	451.875 MHz
4	Xc5vsx35t-3ff665	256	Radix-2	1007/21760	712/21760	8/192	347.947 MHz
5	Xc5vsx35t-3ff665	256	Radix-4	2353/21760	1659/21760	26/192	409.165 MHz
6	Xc5vsx35t-3ff665	1024	Radix-4	2163/21760	1797/21760	9/192	449.035 MHz

5.2 Comparison with other related work

There are three relevant papers that discuss the design and implementation improvements of FFT algorithm in a FPGA. Each of them provides important advice concerning the structure of the design considered in this thesis. In the first paper, the device used by Bin Zhou and David Hwang [2] is Spartan 3 and Virtex E. They had used the pipelined FFT architectures. The results comparison with their thesis is tabulated in the Table 5.3.

Table 5.3 Performance Comparison Results on Virtex-5 devices

Point Size	**Input data width**	**DSP 48**	**Slices**	**BRAM**	**Max speed (MHz)**	**Throughput/ area**
R4SDC [2]						
16	16	4	530	1	236.7	0.447
64	16	8	803	2	236.4	0.294
256	16	12	1370	3	218.9	0.160
1024	16	16	3064	8	219.2	0.072
R2^{2}SDF [2]						
16	16	4	517	1	237.9	0.460
64	16	8	779	2	236.7	0.304
256	16	12	1234	3	236.7	0.192
1024	16	16	2256	8	235.6	0.104
Our design in Xilinx Virtex-5						
16	16	5	832	2	523.1	0.629
64	16	5	921	6	408.2	0.443
256	16	12	1868	5	444.4	0.237
1024	16	9	2163	5	449.0	0.207

Our designed architecture that implements 1024-point FFT utilizes 9 DSP48 slices that gives a throughput of 0.207 which is better than the architecture suggested in [2]. Our design has also provided the smaller area than R4SDC and R2^{2}SDF. The better throughput and smaller area can be observed in every implementation.

Similarly, another comparison is carried out with Spartan 3 devices. The results in Table 5.4 shows that although the absence of DSP48 slices in Spartan devices resulted in lower frequency but still a better throughput is achieved with lower consumption of slices.

Table 5.4 Performance Comparison Results on Spartan 3 devices.

Point Size	**Input data width**	**Twiddle factor width**	**Slices**	**BRAM**	**Max speed (MHz)**	**Throughput/ area**
R4SDC [2]						
16	16	16	468	2	108.20	0.231
64	16	16	952	2	107.23	0.113
256	16	16	1990	3	111.98	0.056
1024	16	16	4409	8	123.84	0.028
R2^{2}SDF [2]						
16	16	16	427	2	121.4	0.284
64	16	16	810	2	98.14	0.121
256	16	16	1303	3	98.73	0.076
1024	16	16	2802	8	95.25	0.034
Our design in Xilinx Spartan 3						
16	16	16	797	2	185.87	0.233
64	16	16	758	6	181.15	0.239
256	16	16	2543	5	166.97	0.065
1024	16	16	2353	5	173.34	0.074

5.3 Comparison with FFT on other FPGAs

A comparison is done between FFT processor in [9] and our design to implement the FFT. Our design using the Virtex family achieves the highest operating frequency while implementing the FFT on FPGA. In [9] FPGA implementation of a 256 complex data point in pipeline Split Radix FFT processor. They had used the Altera Stratix-II family. Following Table 5.5 highlights the results.

Table 5.5 Performance Comparison of FFT on other FPGAs

	No. of points	**Data Width**	**FPGA family**	**Frequency(MHz)**
SRFFT processor [9]	256	16	Altera Stratix-II	350
Our design	**256**	**16**	**Xilinx Virtex-5**	**444.44**

Chapter 6

Conclusion and Future Work

This chapter discusses the concluding remarks and further future direction for the advancement in this field.

6.1 Conclusion

While implementing the FFT on the Xilinx FPGA, we have tried to present the better design that give better throughput while utilizing smaller area as we have observed a tradeoff between the area and throughput. To achieve our goal we have studied the FFT operation in detail and its implementation in two different families of Xilinx FPGAs i.e. Spartan 3E and Virtex-5.

Our design was based on Radix-4 and Radix-2 architectures. The primary goal of optimizing our design is achieved by exploiting the special feature of Virtex 5 which is XtremeDSP slices. These slices increased the operational frequency in Virtex-5.

We have experimented with various points of FFT and analysed their results. Various sizes of FFT architectures were implemented through synthesizable verilog code and verified with simulation through ISIM simulator against Matlab. Xilinx Synthesis tool is used to synthesize the code.

6.2 Future Work

The design which is being implemented by us gives an easy way to increase the size of FFT. Also a little modification in the design can easily make it to use it for the inverse Fourier Transform. The future work includes the exploration of new embedded features in modern FPGAs so throughput improvement and area utilization can be improved.

List of References

[1] Data Sheet.2010, April, “Xilinx, Logicore IP Fast Fourier Transform”, Version 7.1.Xilinx Inc. Available at www.xilinx.com/support/documentation/ip.../xfft_ds260.pdf

[2] B. Zhou, Y. Peng, “Pipeline FFT Architectures Optimized for FPGAs”. *International Journal of Reconfigurable Computing*, vol. 2009, Article ID 219140, 9 pages, 2009.

[3] Data sheet. 2009, “Spartan 3E FPGA Family”. Xilinx Inc. Available at www.xilinx.com/support/documentation/data_sheets/ds312.pdf.

[4] Data sheet. 2009, “Virtex-5 Family Overview” .DS100. Xilinx Inc. Available at www.xilinx.com/support/documentation/data_sheets/ds100.pdf.

[5] S.M.Kuo and W.S. Gan, “Digital Signal Processors : Architectures, Implementations and Applications.” Chapter 8, pp. 448–455, 2005.

[6] Douglas L. Jones, “Decimation-in-time (DIT) Radix-2 FFT”. Available at

http://cnx.org/content/m12016/1.7/

[7] Douglas L. Jones, “Radix-4 FFT Algorithms” .Available at

http://cnx.org/content/m12027/1.4/

[8] Data Sheet. XtremeDSP slices. Available at

http://www.xilinx.com/technology/dsp/xtremedsp.html

[9] J. García1, J. A. Michell, G.Ruiz, and A.M. Burón, "FPGA realization of a Split Radix FFT processor." Proc. of *SPIE.Microtechnologies for the New Millennium*, vol. 6590, May 2007.

[10] J.G. Proakis and D Manolakis, "Digital Signal Processing. Principles, Algorithms and Applications.' 4th edition, 2006.

[11] Xilinx, available at http://www.xilinx.com/company/gettingstarted/

[12] U.Mayer and Baese, "Digital Signal Processing with Field Programmable Gate Arrays", *Springer*, 3rd edition, pp 12-15.2003.

[13] J. A. C. Bingham, "ADSL, VDSL, and Multi Carrier Modulation." Appendix C, pp 259-273, October 2001.

[14] K.S.Hemmert and K.D. Underwood, "An Analysis of the Double-Precision Floating-Point FFT on FPGAs." Proc. of *13th Annual IEEE symposium on Field-Programmable Custom Computing Machines*, pp 171-180, April 2005.

[15] J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series.", *Mathematics of Computation*, vol. 19 No.90, pp. 297-301, April 1965.