Novel Architecture of Smart FFT Processor

Rozita Teymourzadeh

NOVEL ARCHITECTURE OF 1024-POINT HIGH EFFICIENT FFT PROCESSOR

Author: Asst. Prof. Dr. Rozita Teymourzadeh, CEng.
Formatted by: Kok Wai Chan

Dedicated To My Mother “Nasrin Teymourzadeh”

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

e	Number of exponent bit
$f_{s.\max}$	Maximum clock frequency
μs	Micro second
Ms/sec	Mega sample per second
N	Number of input samples
W_N^{kn}	Twiddle factor
$x(n)$	Even input data in discrete-time
$x_e(n)$	Odd input data in discrete-time
$x_o(n)$	Input data in discrete-time
$X(\omega)$	Input data in frequency domain
$\delta(t-t_n)$	Shifted impulse signal
τ	Signal domain

LIST OF ABBREVIATIONS

ADC	Analog to Digital Converter
ASIC	Application Specific Integrated Circuit
ATV	Advanced Televisions
CDMA	Code Division Multiple Access
CFMR	Continuous Flow Mixed-Radix
CMOS	Complementary Metal Oxide Semiconductor
CORDIC	Coordinate Rotational Design Computer
DA	Design Analyzer
DAB	Digital Audio Broadcasting
DAC	Digital to Analog Converter
DFT	Discrete Fourier Transform
DIF	Decimation In Frequency
DIT	Decimation In Time
DVB	Digital Video Broadcasting
DS	Direct Sequence
DSL	Digital Subscriber Loop
DSP	Digital Signal Processing
DTFT	Discrete-Time Fourier Transform
FDMA	Frequency Division Multiple Access
FIR	Finite Impulse Response
FFT	Fast Fourier Transform
FM	Frequency Modulation
FPGA	Field Programmable Gate Array
FPP	Floating-point Parallel Pipeline
HDTV	High-Definition Television
HSPA	High-Speed Packet Access
HVAC	Heating Ventilation and Air-Conditioning
IC	Integrated Circuit
IFFT	Inverse Fast Fourier Transform
I/O	Input/Output
ISE	Integrated Synthesis Environment
LSB	Least Significant Bit
MC	Multi Carrier
MCM	Multi Carrier Modulation
MDC	Multi-path Delay Commentator
MOS	Metal-Oxide Silicon
MR	Mix Radix
MSA	Multiply-Sub-tract-Add

MSB	Most Significant Bit
MSE	Mean Squared Error
NSR	Noise to Signal Ratio
OFCDM	Orthogonal Frequency and Code Division Multiplexing
OFDM	Orthogonal Frequency-Division Multiplexing
PAR	Place And Route
PAVR	Peak-to-Average Ratio
PE	Processing Element
RAM	Random Access Memory
RNS	Residue Number System
ROM	Read Only Memory
SDF	Single-path Delay Feedback
SNR	Signal to Noise Ratio
SQNR	Signal Quantization Noise Ratio
SOC	System On Chip
TCP	Transmission Control Protocol
THD	Total Harmonic Distortion
VHDL	VHSIC hardware description language
VHSIC	Very High Speed Integrated Circuit
VLSI	Very Large Scale Integration
WLAN	Wireless Local Area Network
2D	Two Dimensional
4G	Fourth Generation

CHAPTER I

INTRODUCTION

1.1 BACKGROUND

The need for new generation of digital processor identified as floating point fast Fourier transform (FFT) that is capable of handling new requirement in signal processing has mobilised the world of high performance digital signal processing (DSP). The FFT processor (Cooley & Tukey 1965) is the heart of signal processing which is widely used in multi-media applications (Bever et al. 1990, Jeong & Choon 1995), telecommunication systems (Le et al. 1995, Nee & Prasad 2000, Zhiqiang & Nassar 2005) and DSP processor. In the field of DSP processor, there are specific applications such as fast finite impulse response (FIR) filtering (Eniscetin et al. 1997), spectral analysis, synthesis and correlation (Jont et al. 1977). Among DSP processors, FFT algorithm is most practical processor. High performance FFT is required in this particular study due its efficient algorithm.

The current prevalent practice, there is the digital hardware with finite word-length to present and analyze DSP system. Realistic FFT implementation requires special attention due to potential quantization and arithmetic errors as well as the possibility of overflow and round off. These effects must constantly be taken into consideration in DSP system design and the implementation for sensible applications. A FFT Processor's data format determines its ability to handle signals of different precisions, dynamic ranges and signal-to-quantization-noise ratio (SQNR).

The data is represented in digital systems with different formats known as fixed point and floating-point arithmetic. In fixed-point FFT processors, a number is represented with a series of binary which the left most bit is named the most significant bit (MSB). This MSB represents sign of the number. In addition fixed point arithmetic, IEEE 754 (IEEE 1985) standard introduced a new format identified as floating-point arithmetic.

The floating-point arithmetic automatically scales the number to obtain the full range representation of the mantissa, which is done by increasing or decreasing the exponent value for small or large numbers respectively. In other words, the floating-point algorithm tracks the number and adjusts the value of the exponent. Due to the FFT processor being involved with huge calculations, its fixed-point implementation limits the dynamic range, which jeopardizes the precision. Hence, this book describes the designing of high-speed floating point FFT processor for very high-resolution applications.

1.2 PROBLEM STATEMENT

It is common knowledge in the research community that the implementation of a fixed point is preferred due to its efficiency: it requires less computational complexity and less silicon area when the hardware is implemented. However, a problem arises on most current work on fixed point as it has limited accuracy, which is inherent in many FFT calculations.

In order to execute fixed-point calculation in DSP processors, normalization, round off and quantization are required, which results in low precision and low dynamic range. Using fixed-point processors would cause overflow if the incoming numbers are in full scale. In addition, quantization effects also decreases SNR of the signal. Therefore, floating-point structure is rise up to function.

Floating-point arithmetic such as IEEE 754 standard promises more numerical accuracy than fixed point. Due to the fact that floating point offers high resolution it is thus adopted more as DSP applications which require high performance for signal processing.

The trade-off between fixed and floating points is the achieved resolution, speed, power consumption and core size. In order to implement in floating point architecture, there are the undesirable issues as follow:

i) Although many different fixed point FFT algorithms have been developed, there are still not enough floating-point literatures as a resource for this research work.

ii) It is a norm that the floating-point arithmetic is customarily ignored due to its complex in nature.

iii) In the past in order to obtain high resolution, the throughput has to be sacrificed due to complex architecture.

iv) The system latency will be raised due to the application of more components to benefit high resolution.

v) Its implementation is not acceptable because of the high cost of implementation and its instability.

vi) Most of the chip implementations reported for the implementation of fixed point and floating point FFT processors (Fox & Surace 1987, Long et. al. 2007, Morton et al. 1994, Plessey 1990) have been involved with multiple chips, which are to perform FFT calculations that resulted in high cost and low efficiency.

Hence based on the issues above, the design was proposed with the use of a novel floating point FFT architecture to achieve high resolution ($\leq$ 0.01% accuracy) and high speed ($\geq$200 MHz), low area (single chip) and low power consumption ($\leq$1 *W*) with reduction of latency.

1.3 MOTIVATION

The idea of this book is a floating point FFT processor with high speed and high resolution. The advancement of VLSI processing (60 nm) enables implementation of capable DSP architecture. Hence based on the explanations given prior to this and the

problems faced in terms of low resolution and overflow in fixed point FFT structure, it motivates us to design and investigate the floating-point FFT processor and implement it in order to provide higher precision and larger dynamic range. The designed floating-point FFT architecture claims closed parameter characteristics as well as fixed-point architecture in terms of speed and latency and single-chip implementation.

Therefore, the floating-point operation supports more accurate FFT calculations at the speed of processing integers while preventing difficult problems such as overflow, operand alignments, and signal scaling that commonly occur in fixed-point operations. Furthermore, floating point FFT processors support addressing modes and a higher degree of parallel execution. A high level hardware language is more efficient on a floating point processor with register based architectures, floating point instruction, a large address space, powerful addressing mode and a faster floating-point computing engine. These methods were utilized to achieve an intelligent controller for proposed floating-point FFT.

Additionally, with the upcoming requirement of signal processing systems and convergence of high performance DSP applications, DSP processors are needed to configure for the computation of various algorithms used in high-resolution applications. Based on the previous section, the significant incentive in this project is the design of high performance floating-point single chip FFT algorithm for signal processing. This motivation leads to design on-chip implementation of floating-point components to bring down the cost and optimize the system for the smallest size possible area to fit in the field-programmable gate array (FPGA) board and application specific integrated circuit (ASIC) implementation. Furthermore, FPGA implementation of the floating-point FFT is well suited in requirement for portable applications and low power consumption has long been the main constraint.

Several other factors were taken into consideration such as more functionality, higher workload, and longer operation time that contribute towards making the power consumption and energy efficiency even more critical and desirable for DSP application. However, in order to have low power, the implementation of FFT

algorithm is still a challenging task. (Cummings & Haruyama 1999, Dick & Harris 1999).

1.4 OBJECTIVES

The overall objective of this book is to propose the high performance floating point-based FFT that are suitable for harmonic analysis applications. Specifically the proposed FFT processor must have a 32-bit resolution, a SNR of more than 100dB and a maximum clock frequency of more than 100 MHz in order to meet harmonic monitoring applications. In achieving the above objectives, it is accomplished:

i) To develop a novel 1024-point radix-2 FFT algorithm DSP processor suitable for high resolution motor monitoring application.

ii) To optimize the architecture of a novel algorithm for high-speed and high resolution, low power consumption as well as small silicon area for VLSI implementation.

iii) To design and test the proposed novel architecture 1024-point radix-2 FFT processor implemented on the FPGA board.

iv) ASIC implementation of active core area for novel FFT algorithm based on 0.18 μm SILTERRA and 0.35 μm MIMOS technology libraries.

v) To conduct the performance analysis of the FPGA 1024-point FFT chip.

1.5 SCOPE OF WORK

Our research and development scope is the designing, implementation and testing of the high-resolution novel 1024-point radix-2 floating-point parallel pipeline FFT processor in signal processing system. The proposed FFT processor must meet high-resolution requirement for accurate DSP applications. The system had to be

implemented on FPGA board and optimised on ASIC. The research and development will not focus on the designing of other corporation in the complete equipment and devices.

1.6 PROJECT WORKFLOW

The workflow of this project is based on research, simulation, implementation and optimization. First, the problem statement and objectives of the research are formulated. Then the scope of work is determined. Once the algorithm for the proposed and conventional FFT is identified, a comprehensive MATLAB simulation was carried out to determine the functionality of the algorithm. Later, the hardware coding (using VHDL) will be derived then followed by implementation using ASIC or FPGA board to fulfill system requirement.

Figure 1.1 detailed the overall workflow carried out in this project. It shows the structure used to validate the simulation of the conventional 1024-point radix 2 FFT then followed by the implementation of the proposed 1024-point radix-2 floating-point parallel pipeline (FPP) FFT processor. The proposed FPP-FFT is also optimized using Xilinx, Synopsys and CAD tools. With the scope of work successfully narrowed down to realistic goals, the design and implementation of the novel floating point FFT architecture has been identified. This overall research work involves the effort of simulation and VLSI implementation of high-speed high-resolution floating point FFT structure to meet the required system functionalities.

The output result in terms of amplitude, phase, spectrum and specifications will be evaluated and the comparison will be performed using MATLAB and fixed point radix-2 FFT processor hardware implementation.

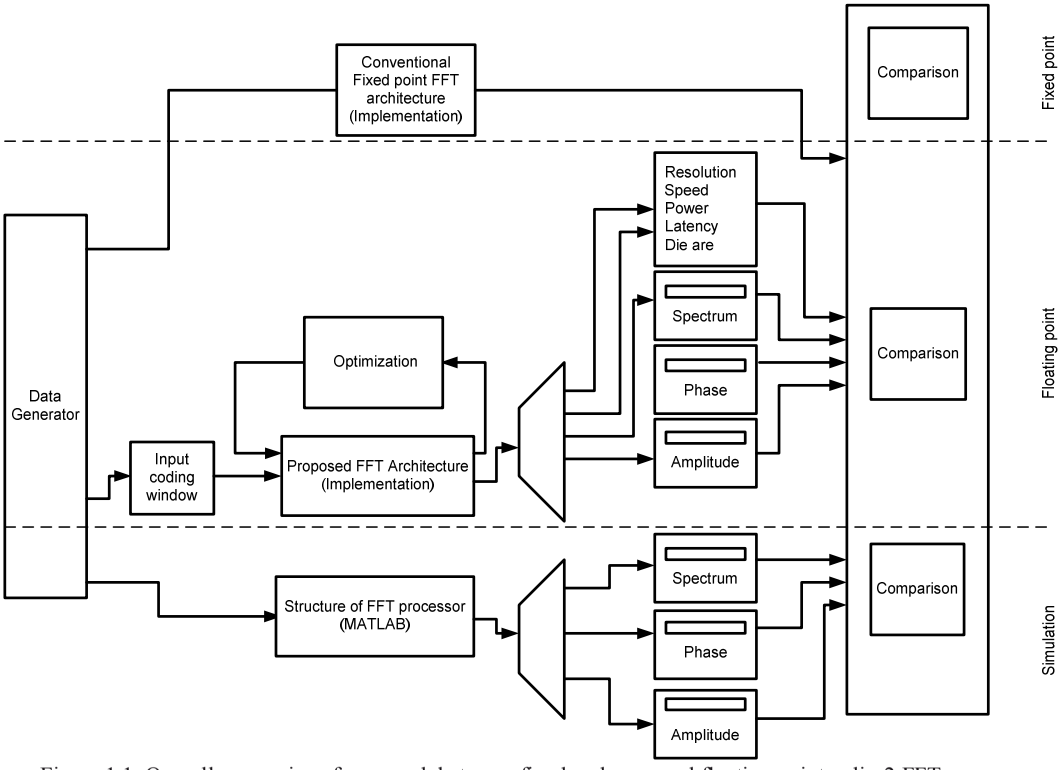


Figure 1.1 Overall comparison framework between fixed and proposed floating point radix-2 FFT processor

1.7 METHODOLOGY

Based on the discussion in the previous sections, the methodology presented in this book is as follow:

i) To present the novel architecture of the high-speed FFT processor. The research also continues to achieve high efficiency in proposed algorithm.

ii) To perform MATLAB simulation of the 3rd stage decimation in time (DIT) FFT structure to obtain the needed frequency response.

iii) To conduct VHDL implementation of proposed novel 1024-point radix-2 FPP-FFT algorithm by synthesising and defining constrained file for the design.

iv) To implement the design and download the design to Xilinx Virtex II FPGA board and test the FFT with a logic analyser. The FPGA board output is compared with MATLAB simulation result.

v) To ASIC optimization of proposed radix 2 FFT architecture in SILTERRA and MIMOS technology libraries.

1.8 CHAPTER ORGANIZATION

The work in this book organized into seven chapters. The first chapter gives a brief introduction of the FFT processor.

The second chapter provides the review of FFT and DFT processor and its application. The comparison between floating-point and fixed-point processor was made in this part. This chapter also provides brief summaries of the literature review

prior to engaging the mentioned scope of work. Several topics related to this research are reviewed to give an overall picture of the background knowledge involved. The summary of the literature review is given to clarify the research rationale.

Chapter Three presents the principle of the FFT processor in detail. The different architecture of FFT based on decimation in time (DIT) and decimation in frequency (DIF) and its comparison are discussed. In addition, different architecture of FFT processor based on radix structure is considered in detail.

Chapter Four introduces the architecture of the proposed FFT algorithm followed by the introduction of floating-point data structure in detail. The proposed floating-point parallel FFT processor architecture is designed. All components of this processor are evaluated separately. New algorithm are explained and performed to increase the resolution and throughput of the FFT architecture. The structural advantage of the proposed system is discussed in detail.

In chapter Five, simulation result of the standard radix-2 FFT processor using MATLAB tools is presented. In order to implement the FFT processor the individual block diagram is designed and simulated and a particular model was considered to perform floating-point data testing.

Chapter Six focuses on realization of the proposed FFT architecture and its implementation. The 1024-point radix-2 FPP-FFT processor are structured and implemented to increase the throughput of the processor. The implementation of advanced components such as floating-point adder/subtractor, address generator and butterfly radix-2 are investigated and performed accordingly. The system is synthesized by Xilinx ISE synthesis software then the floating point pipeline adder/subtractor and multiplier architecture are introduced to perform arithmetic calculation. Additionally, the proposed processor is also synthesized in SILTERRA 0.18 µm technology and the MIMOS 0.35 µm technology.

Chapter Seven extracts the book conclusions and proposes future work.

CHAPTER II

AN OVERVIEW OF FFT ALGORITHM

The prevalent subject of Fourier analysis encompasses a vast spectrum of mathematics with parts that may appear quite different at first glance. In Fourier analysis, the term Fourier transforms often refers to the process that decomposes a given function into the harmonics domain. This process results in another function that describes what frequencies are in the original function. However, the transformation is often given a more specific name depending upon the domain and other properties of the function being transformed. Moreover, the original concept of Fourier analysis has been extended over time to apply to general situations and the general field often known as harmonic analysis. In this project, the common approach in obtaining Fourier analysis will be discussed.

2.1 DISCRETE FOURIER TRANSFORM (DFT)

The main concepts of discrete Fourier transform (DFT) (Bergland 1969, Duhamel et. al. 1988) are central of most DSP processor. The DFT is a Fourier representation of a finite-length sequence which is the most important fundamental operation in digital signal processing and communication system (Gold & Radar 1969, Smith 2007). However, the computation complexity of the direct evaluation of an N-point DFT involves a long phase computational time and large power consumption. As result of these problems, it is important to develop fast algorithm. There are numerous viewpoints that can be taken toward the derivation and interpretation of the DFT representation of a finite-duration sequence. The sequence of $\tilde{x}(n)$ that is periodic with period N so that $\tilde{x}(n) = \tilde{x}(n+kN)$ functions for any integer value of k.

It is possible to represent $\tilde{x}(n)$ in terms of Fourier series that is by the sum of *sine* and *cosine* values or equivalently complex exponentially sequences with frequencies that are integer multiplies of the fundamental frequencies $2\pi/N$ associated with the periodic sequence. The same representation can be applied to finite-duration sequence.

The resulting Fourier representation for finite duration sequences will be referred to as the DFT. Sequence of length *N* by a periodic sequence can be represented by a periodic sequence with period *N*, one period of which is identical to the finite-duration sequence. The sampled sequence signal in frequency is defined as:

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\omega n} \tag{2.1}$$

The DFT $X(\omega)$ is a function of continuous-frequency variable ω, and the summation in equation (2.1) extends towards positive and negative infinitively. Therefore, the DFT is a theoretical Fourier transform of a digital signal. However, it cannot be implemented for real applications. It is the sample of the signal in time domain at a particular time and can be expressed as:

$$x(n) = \int_0^{\infty} x(t)\delta(t - t_n) \tag{2.2}$$

The frequency analysis of a finite-length sequence is equal to the sample of continuous frequency variable ω at *N* equally spaced frequencies $\omega_k = 2\pi k/N$ for $k = 0, 1, 2, \ldots N-1$ on the unit circle. These frequency samples are expressed as:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi kn}{N}} = \sum_{n=0}^{N-1} x(n) W_N^{kn} \quad , k = 0,1, \ldots, N-1 \tag{2.3}$$

where the twiddle factors are defined as:

$$W_N^{kn} = e^{-j\left(\frac{2\pi}{N}\right)kn} = cos\left(\frac{2\pi kn}{N}\right) - j\, sin\left(\frac{2\pi kn}{N}\right) \tag{2.4}$$

The DFT is based on the assumption that the signal $x(n)$ is periodic. Therefore, $X(k)$ for $k = 0,1, \ldots, N-1$ can uniquely represent a periodic sequence $x(n)$ of period *N*. the

inverse DFT is the reversed process of the DFT. It converts the frequency spectrum *X(k)* back to the time domain signal *x(n)*. (Kuo & Gan 2005).

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\, e^{j2\pi kn/N} = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\, W_N^{-kn}, n = 0,1,\dots,N-1 \tag{2.5}$$

Direct computation of an *N*-point DFT according to Equation (2.3) requires *N(N-1)* complex additions and *N(N-1)* complex multiplications. The complexity for computing an *N*-point DFT is therefore $O(N^2)$. With the contribution from Cooley and Tukey (1965) the complexity for computation of an *N*-point DFT can be reduced to $O(N\log_2(N))$.

Based on the explanations given, the algorithm was developed to achieve high efficiency of DFT calculations, hence resulting in the introduction of a fast Fourier transform (FFT).

2.2 FAST FOURIER TRANSFORM (FFT)

Although the DFT method is very clear and straight forward, it is quite inefficient. As the number of input data in the DFT increases, the amount of necessary mathematical calculations such as additions, multiplications or data manipulations become excessive which is undesirable in DSP processor. Hence much effort has gone into developing alternative and more efficient ways of implementing DFT.

In 1965 Cooley and Tukey developed the use of FFT in order to save time and avoid unnecessary complex calculations. Nowadays, FFT is an important tool in DSP applications. The FFT is an efficient DFT algorithm (Oppenheim & Schafer 1989). As discussed, computing a DFT of *N* points in the obvious way is reduced to *O(N log N)* operations when the *log* is to the base-2 by applying FFT technique.

If the function to be transformed is not harmonically related to the sampling frequency, the response of an FFT looks like a *sinc* function. Aliasing (leakage) also reduces the resolution and negatively affects the frequency domain. However, aliasing reduction is at the expense of broadening the spectral response. There are many distinct FFT algorithms involving a wide range of mathematics, from simple complex-number arithmetic to group theory and number theory. The next section will describe the history of FFT processor followed by expressing the other research work on different FFT structure in detail.

2.3 THE FLOATING POINT FFT ALGORITHM

As discussed in Chapter I, in order to implement high performance and high efficiency DSP processor, it is imperative to increase the precision and dynamic range accordingly. Some issues make the frequency spectrum inaccurate.

The first is that the measured frequency will be subject to quantization error with respect to the real frequency. This is caused by the fact that the FFT only computes the spectrum at discrete frequencies. This error is said to affect the accuracy. The second effect is that of spectral leakage. This effect becomes very important when small amplitude harmonics are close to large amplitude ones since they become hidden by the energy distribution of the larger harmonic. Furthermore, the fixed internal arithmetic calculation of the FFT processor injects noise in the frequency spectrum. To reduce the effects, the floating-point technique was applied.

The floating-point techniques allow numbers to be represented with a large dynamic range. Therefore, floating-point arithmetic enables the reduction of overflow problems that occur in fixed-point arithmetic. Although it is at the expense of throughput and chip area size, the new architecture is designed and investigated to avoid undesired effects in floating-point FFT algorithm.

Based on IEEE 754 standards, floating-point arithmetic provides higher precision and a much larger dynamic range. Therefore, floating-point operations

support more accurate DSP operations. Table 2.1 compares the efficiency between fixed-point and the floating-point FFT processor:

Table 2.1 Fixed-point and floating point FFT processor efficiency

Fixed-point FFT	Floating-point FFT
16-bit or 24-bit	32-bit
Limited dynamic range	Large dynamic range
Overflow and quantization errors	Less error
Higher frequency	Low frequency
Less silicon area	More silicon area
Cheaper	More expensive
Low power consumption	High power consumption

As illustrated in Table 2.1, implementing floating point FFT architecture scarifies the high frequency and the chip core size, which resulted in high power consumption.

2.4 THE FLOATING POINT FFT APPLICATION

In particular, the FFT processor is widely employed in harmonic measurement (Biswas et al. 2009, Farhang-Broujerdy & Gazor 1994), signal processing (Jain 1989, Schafer & Rabiner 1973, Jain et. al. 1979) communication system (Adams 1987, Dovel 1989, Smith et. al. 1990, Stearn & David 1988) and many more.

Generally, there are differences in application between fixed-point and floating point FFT processors. Fixed-point processors are mostly used for large-volume products such as modems and wireless phones. It is also applied in consumer audio applications such as MP3 players, multimedia gaming, digital cameras and speech coding. Among these different applications, floating point 1024-point radix-2 FFT processor is applied in image processing radar, high resolution motor monitoring, high–end audio applications such as ambient acoustics simulators, professional audio

encoding and audio mixing. It is also appropriate to be used in sound synthesis and prototyping. In this book, the implementation of high precision FFT processor was applied in the harmonic spectrum for motor monitoring applications. However, the design can be utilized in ambient acoustics, speech and signal processing systems (Schimmel et al. 2009) as well.

In general, effective performance of harmonic analysis operating in practical environments may require suppression of noise from the wave form. The design of advanced system control monitoring involves careful considerations of the rotation of the motor, transducer locations and digital signal processing. The high dynamic range in the motor vibration signals is very helpful but hard to obtain (Rabenstein & Zayati 1999).

Furthermore, many signal processing applications require algorithms that are robust to reverberation such as noise cancellation algorithms in the system aids and must function in a wide variety of reverberant conditions.

Similarly, it is often desirable that the performance of motor movement recognition systems do not suffer in presence of reverberation. Hence, it is necessary to design high precision FFT processor to recognize the signal spectrum in the authentication systems and this processor significantly affects monitoring applications in terms of high efficiency.

To conclude, Table 2.2 shows the FFT algorithm application in different fixed-point and floating-point architectures.

Table 2.2 Fixed-point and floating-point FFT processor applications

Fixed-point FFT	Floating-point FFT
Low resolution disk drive	Radar , Image processing
Consumer audio application	High-end audio application, ambient acoustics simulators
Channel coding	Professional audio encoding /decoding and audio mixing
Communication device	Sound synthesis in professional audio and video coding /decoding
	Prototyping
	4G OFDM Transceiver
	High resolution motor monitoring

The high performance FFT processor can find application in instrumentation and communication system, which will be discussed. However, our concentration in this research work is based on instrumentation.

2.4.1 Instrumentation

High resolution motor monitoring and instrumentation is the major application, which is focused. Modern monitoring techniques commonly use high performance FFT processor. Vibration or current spectra are often unique to a particular series of motors or even particular motors. When a motor is commissioned or when it is in a healthy state, a reference spectrum is monitored which can be compared later with the new status of the motor spectrum (Gieras & Wing 2002).

In addition, white goods appliance with blowers and compressors, heating ventilation and air-conditioning (HVAC) systems, industrial servo drives, automotive control systems and variable speed control of AC electrical machines make use of high-resolution FFT processor (Mathwork 2010). Each motor can be used either in generator or in motor mode. Combined with linear and nonlinear elements such as transformers, lines, loads, breakers, etc., they can be used to simulate electromechanical transients in an electrical network. They can also be combined with power electronic devices to simulate drives. However, it is significantly imperative to

design a high performance motor control system while they are running. Furthermore, the harmonic study instruments are a challenging task.

Hence based on above explanation, it is needful to apply the high resolution floating point FFT processor in performing a harmonic analysis of the motors and vibration monitoring (Biswas et al. 2009). The proposed FFT unit allows computation of the fundamental component of voltage and current while the motor is active in the network. The advantages of the proposed high performance system are to:

- Enable the use of real-time algorithms for more precise and accurate control.
- Yield better power efficiency and small die core size.
- Enable spectrum analysis to be performed by using a single chip.
- Reduce system complexity and low latency

As an example, Figure 2.1 shows the fundamental component and total harmonic distortion (THD) of certain motor-based appliances on its harmonic versus the magnitude.

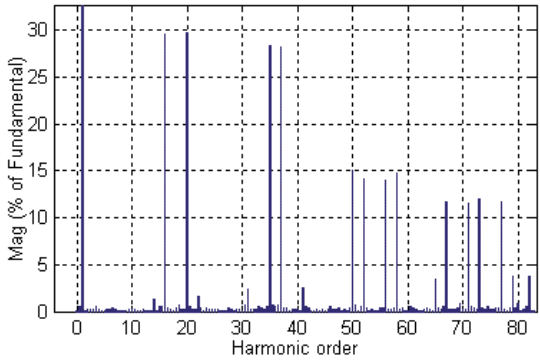


Figure 2.1 Harmonic analysis of the motor using FFT processor

2.4.2 Communication Systems

Beside the instrumentation applications, the updated version of FFT that is the floating-point FFT processor (Enis et al. 1997, Jont & Rabinner 1977) can be used in four-generation (4G) standard communication systems. The 4G refers to the cellular wireless telecommunication standard is a successor to 3Gs and 2Gs.

The 4G mobile communication systems are expected to be standardized and commercialized sometimes between 2010 and 2020. Currently 4G utilizes orthogonal frequency division multiplexing (OFDM) (Nassar 2002, Zhiqiang & Nassar 2005) as transceiver in higher data rate transmission.

In order to grasp a better understanding of the differences between 3G and 4G systems, some criteria will have to be looked at. Amongst which, speed is one that is accepted by the standards bodies with regards to the maximum data rates supported. One of the main advantages of upgrading from a 3G system to a 4G is speed. The 3G systems, which possess a high-speed packet, access (HSPA) supplies up to approximately 15-20 Mbits/s downlink and about 5-10 Mbits/s uplink. Meanwhile, the 4G systems are designed to be an improved rate that supports 5 to 10 times the existing rates of 3G, with greater than 100 Mbits/s or more in the downlink and over 50Mbits/s in the uplink (Friedmann 2007).

With the advent of fourth generation (4G) (Lefevre & Okrah 2001) communication systems, it has become increasingly important for electrical engineers to develop high standard transceiver to cover 4G requirement. Cellular phone systems have never experienced more than 30-Mbps data transmission even in the current 3G (Ojanpera and Prasad 1998) standard; therefore, as a totally new scheme, the OFDM and orthogonal frequency and code division multiplexing (OFCDM) scheme (Atarashi & Sawahashi 2001, Flock et al. C.1995) is proposed for 4G. The techniques are widely employed in data delivery systems over the phone line, digital radio and television and wireless networking systems (Wu & Zou 1995, Paiement 1994).

On the other hand, wireless LANs are already able to provide up to 54-Mbps data transmission even in the current standards, such as IEEE802.11a (IEEE 1999) and HIPERLAN/2 (ETSI TR 2000) , although the service provision is still limited for stationary or low mobility users. Therefore, if a future wireless LAN standard, which might be similar to the current OFDM-based standards (Couasnon & Monnier 1994, Nee and Prasad 2000) with a bit higher transmission rate, and is able to cope with high mobility of users it can then be termed as 4G systems. The expectations from 4G are high in terms of data rates, spectral efficiency, mobility and integration. The future 4G wireless network infrastructures will consist of a set of various networks that use transmission control protocol (TCP) as a common communication protocol. Besides that, in order to satisfy the high data rate requirement efficiency in supporting multimedia services, OFDM is considered as one of the most compromising candidates for the physical layer standard of future generation mobile communications. For these reasons, a mobile system based on OFDM is being intensively considered as the next generation standard by 4G committees such as IEEE 802.16e and IEEE 802.20 (Kim & Yoon 2005), and is proving to be a possible multiple access technology to be used in 4G in terms of high performance and efficient components.

However, OFDM itself comes with its own challenges such as linearity concerns, efficient internal FFT and phase noise (Qaddour 2006). Hence, the proposed floating-point FFT is a viable choice to function in 4G OFDM transceiver due to its high efficiency. The OFDM offers several advantages over other types of modulation schemes in the improvement of transceiver reception and communication in 4G communication system as well as saving bandwidth. However, implementation of an efficient high performance IFFT/FFT is an important key issue in OFDM hardware design. In the OFDM system, the sub-carriers are created using IFFT in the transmitter, and FFT is used in the receiver to recover the data. The IFFT and the FFT complement each other in function and the most appropriate term depends on whether the signal is being received or generated. In cases where the signal is independent of this distinction then the term FFT and IFFT is used interchangeably. Hence, a high-speed high-resolution FFT/IFFT processor is required for parsing and processing the data.

Furthermore, in real world applications, the resolution of ADC has continued to increase in order to obtain clear signal industry. The resolution rises from 16-bit to 32-bit to ensure signal accuracy. It is required for DSP processors such as floating-point FFT, to provide higher precision to handle signals with larger word length.

Figure 2.2 shows the conventional MCM transceiver whilst Figure 2.3 illustrates the OFDM transceiver with utilizing FFT/IFFT algorithm as modulator.

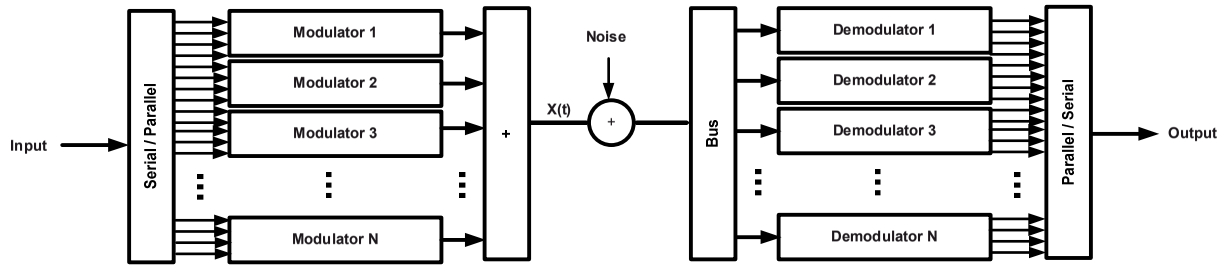


Figure 2.2 Conventional MCM transceiver

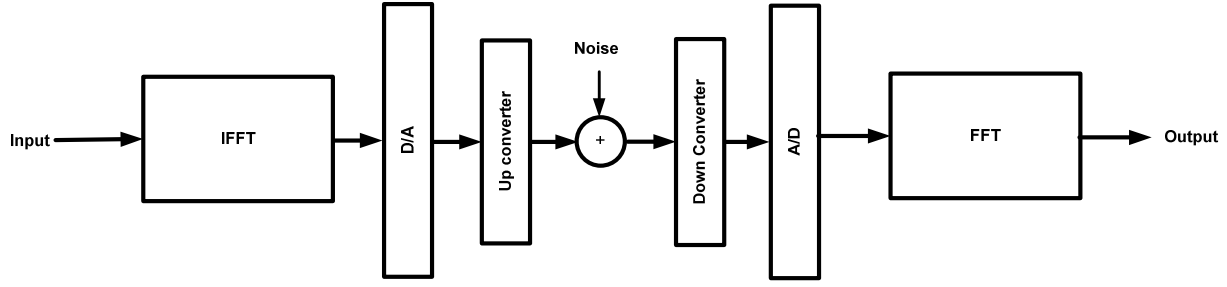


Figure 2.3 The 4G OFDM transceiver with FFT/IFFT modulator

2.5 LITERATURE REVIEW

In 1965, Cooley and Tukey introduced a novel FFT algorithm. They not only reduced the computation complexity in DFT calculation, but also increased the regularity for implementation; they are very popular in modern VLSI implementation. They demonstrated the simplicity and efficiency of the 'divide-and-conquer' approach for DFT computation and made the FFT algorithms widely accepted. They applied the new technique to break down a problem into two sub-problems of the related type. The sub-problems are then independently solved and their solutions are combined to give a solution to the original problem. The conventional Cooley-Tukey radix-2 FFT algorithm requires 192 complex butterfly operations for a 64-point FFT computation. Considering that, one fixed-point FFT unit has to be computed within 4 μs, one butterfly operation has to be completed within 20.8 ns, which leads to 96 MHz operation frequency for a single butterfly FFT unit when each butterfly operates on 2 clock cycles. The system requires a higher clock rate.

Later, in 1977, Thong and Bede investigated accumulation of round off error floating point FFT. They discussed the statistical model for round off error to predict the output noise to signal ratio (NSR) of the two common FFT algorithms, decimation in time (DIT) and decimation in frequency (DIF) algorithms. They introduced a new approach to unify error in FFT algorithm. This method was tested for radix 2 and arbitrary radix.

Tseng et al. (1979) considered the implementation of FFT structure using arrays of ROM. The arithmetic operations were based entirely on residue number system (RNS). Their research work developed optimum procedures for choosing both scaling factors and the position of scaling arrays in the structure. This design was applied for high-speed convolution filter implementation with the RNS. The filters were experimentally verified in a speech processing application.

In 1996, Hui et al. introduced a new fixed-point FFT architecture and chip design for motion compensation based on phase correlation. They designed a low

power FFT processor for use in digital television applications. The proposed 64-point FFT was fabricated using 0.6 µm CMOS technology which comprises 0.5 million transistors in a die area 7.8 × 8 mm^2 with 90-96 dB dynamic range at the FFT output. This processor used 24-bit internal precision at 16.1 ms/sec. The Multiplication br/sec was calculated 62.2 billion in 1 W power consumption. The maximum clock frequency at which the system operated on was 36 MHz. Figure 2.4 shows the proposed Radix-4 FFT architecture introduced by Hui (1996):

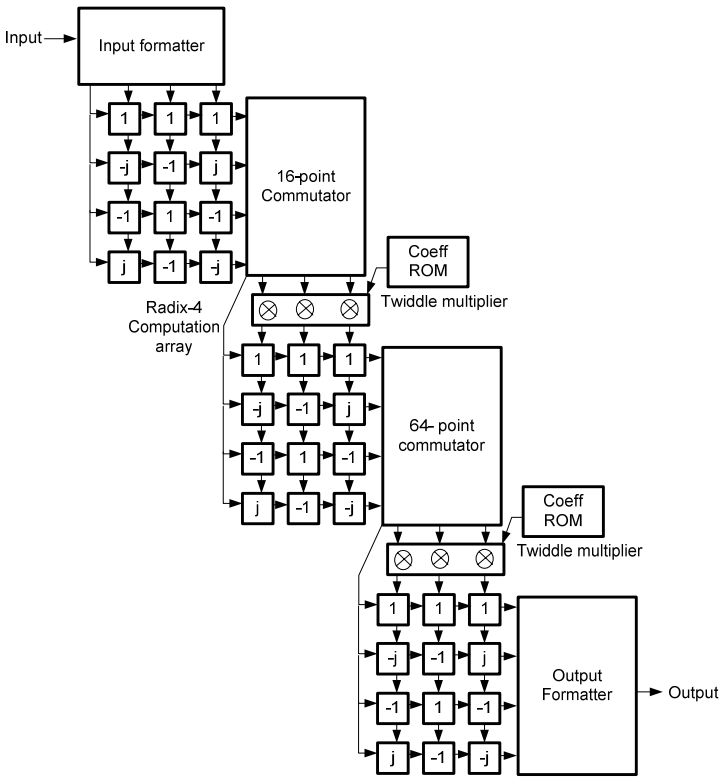


Figure 2.4 Radix-4 FFT architecture introduced
Source: Hui 1996

Melander et al. (1996) presented an efficient design of 128-point radix-2 FFT processor for OFDM applications. This fixed-point algorithm implemented successfully in 0.8 µm CMOS technology library. It was able to calculate the FFT procedure within 80 µs. The core size was measured as 27 mm^2 and the total area including I/O pad 37 mm^2 respectively. The proposed system could operate with maximum clock frequency of 120 MHz with supply voltage of 3.0 V.

Later, the implementation of 1024 pipeline FFT processor was presented by He and Torkelson in 1998. The proposed FFT was based on Radix-2^2 algorithm. The chip was fabricated under 0.5 µm CMOS technology and took an area of 40 mm^2 with a 3.3 V power supply. The sampling frequency was 30 MHz with signal quantization noise ratio (SQNR) of 30 dB for white noise input.

Frigo and Johnson (1998) discussed adaptive software architecture for the FFT. They proposed an adaptive FFT program that tunes the computation automatically for any particular hardware. They compared their program known as FFTW, with others implementations of FFT on seven platforms. They claimed that development of an FFTW-like system requires knowledge of programming languages and compilers.

Later, Lihong et al. (1998) implemented the 8000 fixed-point 64-point FFT algorithm base on radix-2/4/8 in 0.6 µm CMOS with power supply of 3.3 V for DVB applications. The chip was capable of computing an 8K FFT for every 200 µs. I/O bit were 20 to 24 bits. The chip was fabricated with 1.3 M transistors with a core area of 107 mm^2 and chip size of 140 mm^2. The power consumption was 650 mW when the input frequency is 20 MHz.

In 1999, Li and Wanhammar designed and implemented 0.35 µm CMOS high-speed low power 1024-point pipeline FFT processor with flexible internal data length and novel processing element. The proposed FFT completed the calculation within 40 µs.

They introduced their design for wide coverage mobile radio modem with 10 users of 2 Mb/sec each. The Radix 4 single path delay prototype commutator architecture FFT was selected to optimise the power consumption. The implemented core area was measured as 3.1mm×3.4mm and dissipates less than 200 mW at 1.5 V with throughput of 25 MHz.

Beukelman and Bierens presented the design of the fast floating-point FFT processor in 1999. The FFT performed 1024-points floating-point complex FFT with 4 memory banks in DSP applications. The internal hybrid floating point data format (2×24 mantissa+9 bits exponents) resulted more than a single floating-point level of accuracy and dynamic range applications. The SNR for 1K complex FFT was 130 dB with the maximum clock frequency of 100 MHz. The data rate was determined as 2×50 ms/s for 2 FFT processors. The FFT processor was based on the design and there was no implementation result reported. Figure 2.5 shows the FFT processor structure introduced by Beukelman and Bierens.

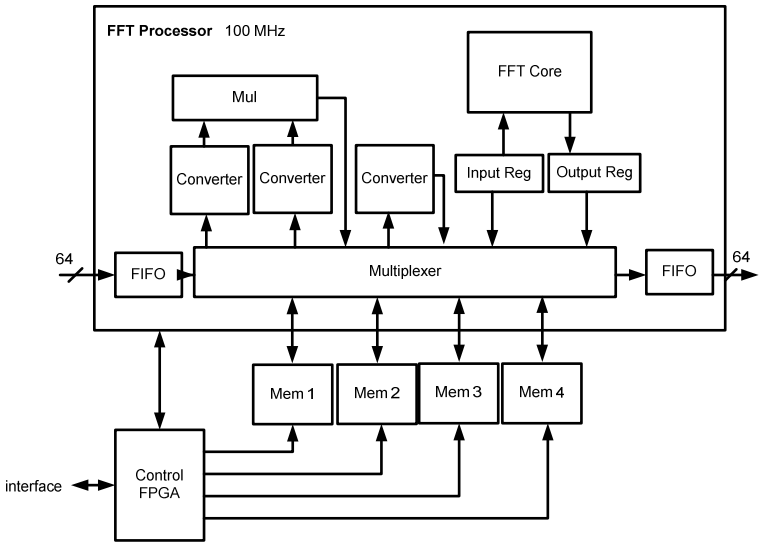


Figure 2.5 Floating-point FFT processor
Source: Beukelman & Bierens 1999

In 1999, Baas introduced a low power high performance 1024-point FFT processor that was fabricated by 460000 transistors under 0.7µm CMOS standard for the DSP application. The proposed 20 to 24-bit fixed-point radix-2 FFT system operated with 3.3 V power supply in order to calculate its function within 30 µs while consuming 845 mW. The maximum clock frequency was 173 MHz at 3.3 V power supply. The FFT processor occupied 5.985×8.204 mm^2 and it was fully functional on first-pass silicon.

Yeo et al. (2002) demonstrated an efficient scheme using reduced precision and word length optimization to reduce power and increase the performance of FFT in the transmitter for IEEE 802.11a WLAN applications. They claimed that with the input reduction bits during transmission, the power consumption was reduced considerably. They designed a 64-point radix-4 DIF-FFT processor. They selected the word length of input data as 10 bit. Although the power consumption measurement has not been shown in the paper, they illustrated that the number of calculation was significantly reduced.

Son et al. (2002) proposed a high-speed 256-pint FFT processor for OFDM based on Radix-4 algorithm. The 20-bit pipeline fixed-point FFT architecture was modelled by VHDL and logic synthesis and was performed using the Samsung 0.5 µm SOG cell library. The total gate count was 98326 excluding the RAM. The processor could operate with 42 MHz clock frequency. The proposed processor could calculate 256-point complex FFT in 260 clock cycles (6 µs).

Yong in 2003, proposed a method of high speed 512 complex fixed-point radix-8 FFT processor. Although there was no report of his implementation, he estimated the maximum clock frequency of 160 MHz and the latency of 4096 (12.8 µs) clock cycles in parallel FFT architecture for DSP application. The internal bit width was considered as 20 bits.

Later on, Dabbagh and Eshghi (2003) constructed the self-timed 8-point pipeline floating point FFT processor. The self-timed technique was used to overcome

a global clock overhead and distribution problem in synchronous FFT processors due to a large area size of floating-point arithmetic units. They suggested using the single shift registers instead of barrel switches in floating-point adder to reduce the required area size. Although they did not mention how much this technique could improve the area and the throughput, MATLAB simulation and VHDL implementation result comparison were given accordingly.

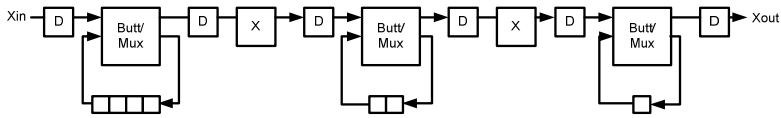


Figure 2.6 Pipelined FFT processor architecture
Source: Dabbagh & Eshghi 2003

In recent decade Jen et al. (2003) designed and implemented a programmable 64-2048-point FFT/IFFT processor to cover the different specifications of OFDM applications. They implemented the processing element (PE) by using coordinate rotational design computer (CORDIC) algorithm to replace the multiplier based PE. Additionally, they proposed $\pi/4$-pre-rotation and modified EEAS-Cordic VLSI architecture to reduce the iteration number and quantization noise. Finally, the FFT processor was implemented with TSMC 0.35 µm CMOS technology. The die area of the FFT was 12.25 mm^2 including 2048×32 bits memory. The chip operated under 80 MHz clock frequency and met most of the standard requirement.

In 2004, Miyamoto et al. designed and built a 36-bit FFT Processor based on the two-stage cached-memory architecture, which integrates 552000 transistors within an area of 2.8×2.8 mm^2 with CMOS 0.35 µm triple–layer-metal process. The Proposed processor could execute a 512-point fixed-point data format, one-dimensional FFT in 23.2 µs (3063 Clock cycle). The maximum clock frequency was 133 MHz with the power consumption of 439.6 mW at 3.3 V power supply.

A new continuous–flow mixed-radix (CFMR) FFT processor was proposed by Byung & Myung (2005). They used the MR (radix-4/2) algorithm and a novel in-place strategy. The proposed algorithm supported fixed-point data. The pipeline CFMR FFT processor using the 0.18 μm SEC cell library consisted of 37000 gates excluding memories, required only 640 clock cycles for 512 point FFT and run at 100 MHz. Figure 2.7 shows the proposed CFMR FFT processor.

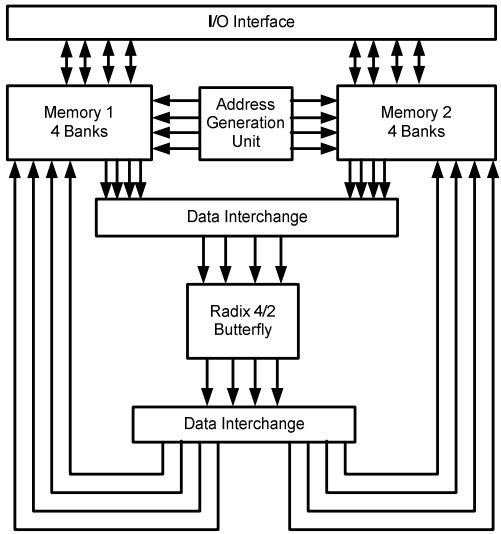


Figure 2.7 CFMR FFT processor
Source: Byung and Myung 2005

Shyue et al. (2005) designed a method for testability and fault tolerant for FFT processors based on M-testability conditions for module level systolic FFT arrays. Their M-testability conditions guaranteed one hundred percent (100%) single-module fault testability with a minimum number of test patterns, according to the proposed design fault tolerant approached at the bit level and the multiply-sub-tract-add (MSA) module level were proposed. This proved that the resulting architecture is simpler as compared with previous work and the reliability of the FFT system increases significantly.

Chin et al. (2006) constructed a low power 64-point FFT/IFFT architecture and chip adopting the retrenched 8-point FFT/IFFT (R8-FFT) and an efficient data-swapping method based output buffer unit. Their concerns were the area, power, latency and pending cycles for the application of IEEE 802.11a WLAN standard. Consequently, the design based on the new method was completed and the synthesis report illustrated the chip power consumption of 22.36 mW under 1.2 V at 20 MHz in CMOS process. The WLAN standards require the FFT calculation to be within 3.2 μs. However, the proposed design needs 72-clock cycle when the timing budget is 44.4 ns. For the post layout, the core area was 1.66 mm^2 and the whole chip size was 2.58 mm^2.

In 2006, Chin et al. proposed low multiplication complexitiy 256-point FFT architecture for WiMAX 802.16a systems. The proposed FFT architecture utilizes cascaded simplified radix-16 single-path delay feedback (SDF) structures. The proposed structure required one complex multiplier and 56 complex adders for supporting the 256-point computations. As a result of this, the output throughput rate of FFT was estimated up to 35.5 Ms/sec with 0.18 μm standard cell technology. The system maximum clock frequency was 32 MHz in FPGA and 51.5 MHz. The total gate count and power consumption were 173875 gates and 162.7 mW respectively in 1.8 V supply voltage.

A novel 128/64 point FFT/IFFT processor for 'multiple-input multiple-output' (MIMO) OFDM based IEEE 802.11n WLAN baseband processor was constructed by Yu & Chen (2007). The unfolding mixed radix multipath delay feedback FFT architecture was applied to deal with multiple data sequences efficiency. The proposed FFT/IFFT processor was designed in a 0.13 μm CMOS process. The core area was 660×2142 μm^2. At the operation clock rate of 40 MHz, the proposed processor could calculate 128-point FFT within 3.2 μs. The maximum clock frequency was 100 MHz with the power consumption of 5.2 mW from the prime power simulator.

Ochi (2008) proposed 64-point floating-point FFT with parallel architecture for wireless application. He focused to improve the area size and dynamic range sequentially and designed a 16-bit Radix-4 FFT processor with a 64-serial data. Both

the butterfly circuit and weight multiplications were carried out in parallel in the various stages. In addition, the parallel architecture reduced the latency by 89 clock cycles whilst the conventional FFT processor has 136 clock cycles latency. In terms of speed, he implemented a 16-bit FFT with maximum clock frequency of 120 MHz and 32-bit FFT with 97.7 MHz clock rate for floating point arithmetic. Its ALUTs is 8261 and 30815 respectively. The architecture of floating-point arithmetic for his proposed design was seen in Figure 2.8:

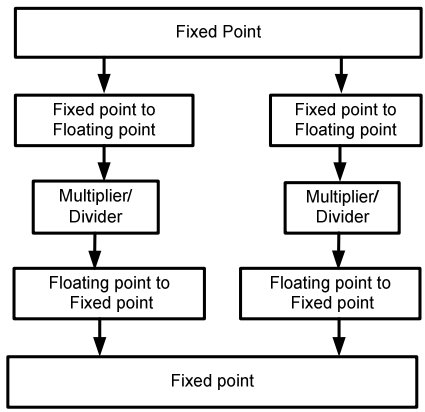


Figure 2.8 Block floating point arithmetic
Source: Ochi 2008

In 2008, Zainal et al. presented a new study over logics circuit operation in sub threshold and threshold region. They developed an FFT processor as an example of digital wireless circuit. The low power consumption of a radix-2 pipeline FFT with low voltage power supply was evaluated. The design was simulated using HSpice level 49 models in 0.18 µm CMOS technologies. The maximum throughput was measured as 100 MHz with the power consumption of 320 mW when power supply was 1 V.

Hojin et al. (2009) discussed 2-dimentional (2D) FFT. They developed a systematic method for improving the throughput of 2D-FFT implementations on

FPGAs. They applied unrolling technique to deploy multiple processing units within a single 1D-FFT. The 2D-FFT design was implemented and evaluated for two different sizes of images 256×256 and 2048×2048 respectively. The maximum available memory for their design was 128 MB operates on 40 MHz frequency.

In 2009, Xin et al. designed and implemented fast memory addressing scheme for radix-4 FFT Implementation. They proposed methods by utilising extra registers to buffer and re-order the data inputs of the butterfly unit. It avoided the module in the address generation hence the critical path was significantly shorter than the conventional radix 4 FFT implementations. The proposed architecture was implemented on an FPGA board and is also synthesized by CMOS 0.18 µm technology. They used pipeline 1024-point FFT in their design. Each real and imaginary part used a 16-bit data which resulted in the maximum throughput of 182 MHz with the total gate count of 176746 cells.

Zhang and Meng (2009) presented a memory-based architecture of 1024-point FFT processor customized for OFDM-based communication systems. They designed a DIF pipeline FFT algorithm for WLAN, DVB-T and ADSL and adopted an efficient memory access scheme to achieve considerable power consumption and cost reduction results. Finally, the VLSI implementation of radix-2 FFT processor was carried out in a 0.13 µm CMOS technology. The estimated area and power consumption of the proposed FFT processor were 2.96 mm^2 and 268 mW respectively with the maximum clock frequency of 100 MHz and operating voltage of 1.08 V.

Finally Gijung and Yunho (2010) designed the scalable FFT processor for MIMO-OFDM-based SDR system which can support the variable length of 64,128, 512, 1024 and 2048 point data.

By reducing the required number of nontrivial multipliers with mixed radix (MR) and multi-path delay commentator (MDC) architecture, where they found that the complexity of their systems was dramatically decreased. The implementation was done in hardware and synthesized to the gate-level circuits using 0.13 µm CMOS

standard cell library. The proposed architecture had a 46k gate count and it could be operated with the maximum clock frequency of 40 MHz.

To review the history of the floating-point arithmetic unit, the previous research work was investigated. Narasimhan et al. (1993) proposed an 8-stage floating-point adder for the FPGA. The specifications for the adder stated that it should work at 62.5 MHz in 2004; Thompson et al. presented a decimal floating-point adder with 5 stages that is compiled with the current draft revision of IEEE-754 standard. The adder supported operations on 64-bit (16- digit) decimal floating-point operands. Initial synthesis testing and evaluation was done and performed using the Synopsys design compiler and LSI Logic's Gflxp 0.11 micron CMOS standard cell library.

Later, Chi (2005) presented a high-speed double-precision floating-point adder using custom-designed macro modules and other advanced optimization technology. Based on SMIC six-layer-metal CMOS process, he achieved a 4-stage pipelined double precision floating-point adder, which could complete a floating-point addition in 7.72ns. The total gate count in this system is 37977 gates. All the results show that the effort is taken to achieve low latency and area, high resolution and speed by introducing the new algorithms. Similar works were also represented within year 2005 to 2009 (Chi et al. 2008, Shi et al. 2008, Xenoulis et al. 2005), to demonstrate engineers effort in increasing the capability of floating-point calculation. However, there are insufficient available resources pertaining to floating-point arithmetic.

2.6 SUMMARY

This chapter describes the DFT algorithm and its applications. It then narrows down to the introduction on the development of DFT by Cooley and Tukey (1965). They called their design as FFT processor. It has engaged in the description of the application of the FFT processor which is the subject of this research and has also made a comparison between the floating-point and fixed-point FFT processor. The advantages and disadvantages of implementing FFT in floating-point architecture were also considered. In this book, an effort is taken to implement a novel algorithm of floating-

point 1024-point radix-2 FFT processor to overcome the disadvantages and achieve high resolution and high dynamic range in DSP applications. Consequently, the literature study presented gives an outline of the relevant fields that are required and used in the duration of this research. In the next chapter, the study will focus on the research methodology employed and a detailed explanation of the FFT processor structure.

CHAPTER III

FAST FOURIER TRANSFORM ARCHITECTURE

INTRODUCTION

As discussed earlier in the previous chapter, FFT algorithm computes an *N*-point forward DFT or inverse DFT (IDFT) where *N* is 2^m. The FFT is a family of algorithms that efficiently implements the DFT. The basic principle of FFT algorithms is the "divide-and-conquer" approach, which decomposes *N*-point DFT into progressively smaller DFTs, thus reducing the computational load as compared with that of the original *N*-point DFT.

Dividing an *N*-point data sequence into two *N*/2-point sequences and performing the DFT on these two sequences individually results in the order of $2(N/2)^2 = N^2/2$ complexity as compared with the original N^2 operations in an *N*-point DFT. Further computational reduction can be achieved by dividing these two sequences with *N*/2 samples into four sub-sequences with *N*/4 samples and performing an independent *N*/4-point DFT on these four shorter sequences (Gentleman & Sande 1966).

The process of dividing can be continued until a 2-point DFT is reached. In addition to the dividing process, the periodic and symmetric properties of twiddle factors can be exploited to reduce the computational load further. Table 3.1 shows the comparison between the calculation of direct DFT and FFT when a different number of *N* is applied.

Table 3.1 Comparison of calculation in DFT and FFT algorithm

Number of points	DFT		Radix – 2 FFT	
	Complex Addition	Complex Multiplication	Complex Addition	Complex Multiplication
N	$N(N-1)$	N^2	$N\log_2 N$	$(N/2)\log_2 N$
4	12	16	8	4
8	56	64	24	12
16	240	256	64	32
32	992	1024	160	80
64	4032	4096	384	192
127	16256	16384	896	448

3.2 FFT TYPE STRUCTURE

To calculate FFT algorithm, there are two well-known methods identified as DIT-FFT and DIF-FFT calculations (Hemmert & Underwood 2005, Shiqun zheng & Dunshan Yu 2004, Thulasiram & Thulasiraman 2003). Both algorithms introduced perform the same functioning. In general, FFT processor has many types in terms of Fourier calculation. Taking into account different types of FFT algorithms are:

- Different radixes, such as radix-2, radix-4, etc. and mixed-radix algorithms.
- DIT and DIF.
- Real and complex algorithm.

3.2.1 DIT Radix-2 Butterfly FFT Processor

As it discussed earlier, the FFT structure will divide the input sequence into odd and even sequence. The number of stream in FFT is $N = 2^m$ when m is positive integer.

$$x_e(n) = x(2m),\ m = 0, 1, \ldots, (N/2) - 1 \tag{3.1}$$

$$x_o(n) = x(2m + 1),\ m = 0, 1, \ldots, (N/2) - 1 \tag{3.2}$$

Based on the DFT definition and combination of the FFT concept, $X(k)$ can be written as:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} = \sum_{m=0}^{\left(\frac{N}{2}\right)-1} x(2m) W_N^{2mk} + \sum_{m=0}^{\left(\frac{N}{2}\right)-1} x(2m+1) W_N^{(2m+1)k} \tag{3.3}$$

Since $W_N^{2mk} = W_{N/2}^{mk}$, the equation will be simplified as:

$$X(k) = \sum_{m=0}^{\left(\frac{N}{2}\right)-1} x_e(m) W_{N/2}^{mk} + W_N^k \sum_{m=0}^{\left(\frac{N}{2}\right)-1} x_o(m) W_{\frac{N}{2}}^{mk} \qquad k = 0,1,\dots,N-1 \tag{3.4}$$

W_N^k which are defined as twiddle factors, and are complex variables with unit amplitude and different phase angles. The 8-point FFT utilises the twiddle factors from W_N^0 to W_N^7. The first twiddle factor is $W_N^0 = 1$. All twiddle factors are distributed around the unit circle. Figure 3.1 shows the twiddle factor for 8-point Fourier transform. The twiddle factor W_N^k repeats itself after every multiple of *N*. The twiddle factors are periodic and for 8-point FFT twiddle factor 0 and 8 are equal.

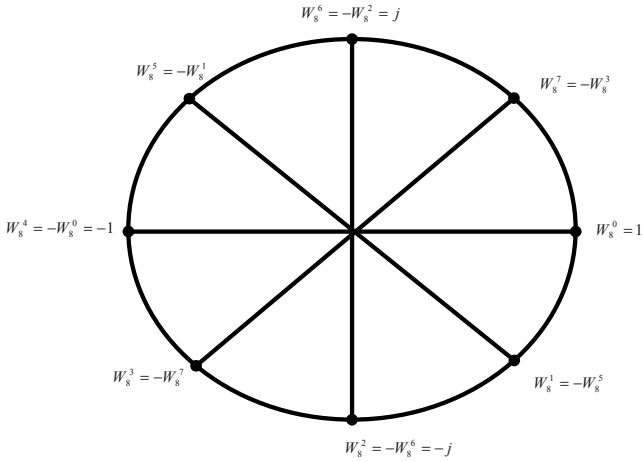


Figure 3.1 8-point FFT twiddle factor

By assuming $X_e(k) = \sum_{m=0}^{\left(\frac{N}{2}\right)-1} x_e(m) W_{N/2}^{mk}$ and $X_o(k) = \sum_{m=0}^{\left(\frac{N}{2}\right)-1} x_o(m) W_{\frac{N}{2}}^{mk}$, this symmetric property provides a reduction in calculations as:

$$X(k) = \begin{cases} X_e(k) + W_N^k X_o(k), & k = 0,1,\dots,\left(\frac{N}{2}\right) - 1 \\ X_e(k) - W_N^k X_o(k), & k = \left(\frac{N}{2}\right),\dots,N-1 \end{cases} \tag{3.5}$$

In general, a radix r butterfly FFT will be created when $N = r^m$. In this book, radix-2 structure as it is the most important radix butterfly that can be concentrated on. The concept of radix-2 will be expanded to cover other radix type structure.

The Butterfly calculation is the fundamental concept of the FFT algorithm and its structure is illustrated by Figure 3.2 when the 2-point data is applied.

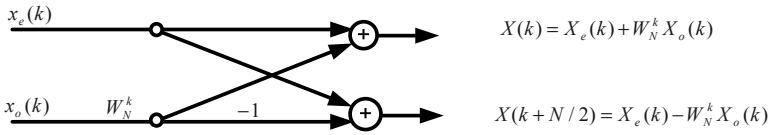


Figure 3.2 2-point butterfly in DIT FFT algorithm

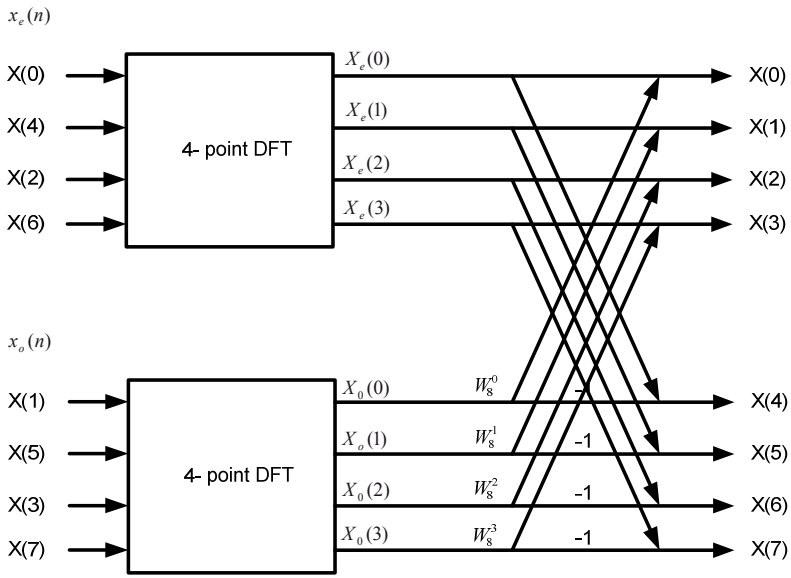


Figure 3.3 Final decomposition of 8-point DIT FFT

To achieve better concept of the radix-2 butterfly FFT calculation, it is assumed $N = 4$. Hence FFT algorithm can be expressed as:

$$X_k = \sum_{n=0}^{3} x_n W_4^{kn} \qquad k = 0,1,2,3 \tag{3.6}$$

$$X_k = x_0 W_4^0 + x_1 W_4^{1k} + x_2 W_4^{2k} + x_3 W_4^{3k} \tag{3.7}$$

Since $W_4^0 = 1$, it is substituted as

$$X_k = \left(x_0 + x_2 W_4^{2k}\right) + W_4^k \left(x_1 + x_3 W_4^{2k}\right) \tag{3.8}$$

Now P_k and Q_k will be defined as:

$$X_k = P_k + W_4^k Q_k \tag{3.9}$$

Hence for 4-point data X_k is expressed as:

$$X_0 = P_0 + W_4^0 Q_0 \tag{3.10}$$

$$X_1 = P_1 + W_4^1 Q_1 \tag{3.11}$$

$$X_2 = P_2 + W_4^2 Q_2 \tag{3.12}$$

$$X_3 = P_3 + W_4^3 Q_3 \tag{3.13}$$

Since P_k and Q_k are periodic with period of ($N/2$), the values for X_k are defined as below:

$$X_0 = P_0 + W_4^0 Q_0 \tag{3.14}$$

$$X_1 = P_1 + W_4^1 Q_1 \tag{3.15}$$

$$X_2 = P_0 + W_4^2 Q_0 \tag{3.16}$$

$$X_3 = P_1 + W_4^3 Q_1 \tag{3.17}$$

Based on the explanations given, a radix-2 butterfly FFT is decomposed into M stages, where $M = \log_2 N$ since $2^M = N$. In each stage, $N/2$ complex are multiplied by the twiddle factors and where N complex additions are required. Therefore, the total computational requirements are $(N\log_2 N)/2$ complex multiplications and $N\log_2 N$ complex additions. Figure 3.4 demonstrates the flow graph of the final decomposition of 4-point DIT-FFT.

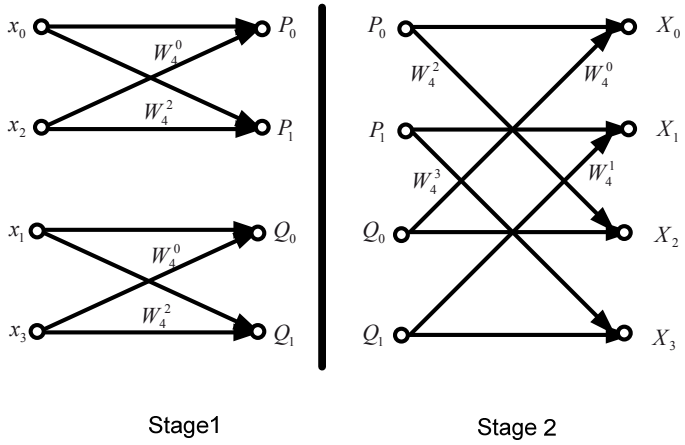


Figure 3.4 Flow graph of the final decomposition of 4-point DIT-FFT

As it is clear from the calculation of the 4-point FFT, two stages are required and each stage has 2 butterfly calculations.

$$P_k = x_0 + x_2 W_4^{2K} \tag{3.18}$$

$$P_0 = x_0 + x_2 W_4^0 \tag{3.19}$$

$$P_1 = x_0 + x_2 W_4^2 \tag{3.20}$$

$$Q_k = x_1 + x_3 W_4^{2K} \tag{3.21}$$

$$Q_0 = x_1 + x_3 W_4^0 \tag{3.22}$$

$$Q_1 = x_1 + x_3 W_4^2 \tag{3.23}$$

Consequently, expanding the radix-2 butterfly calculation for 8-bit data will result as Figure 3.5.

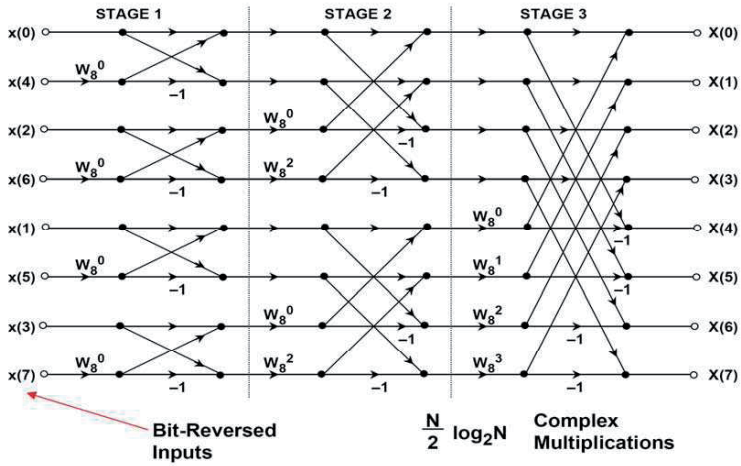


Figure 3.5 Flow graph of the final decomposition of 8-point DIT-FFT

Based on Figure 3.5, the input data must be stored in a non-sequential order for the computation to function appropriately. In fact, the order in which the input data are stored is in bit-reversed order. As an example for 8-point FFT, bit-reverse block changes the location of three binary digits as shown in Table 3.2.

Table 3.2 Bit-Reversal process for N=8

Binary Index	**Bit-reversed index**
000 (0)	000 (0)
001 (1)	100 (4)
010 (2)	010 (2)
011 (3)	110 (6)
100 (4)	001 (1)
101 (5)	101 (5)
110 (6)	011 (3)
111 (7)	111 (7)

If three binary digits is the representative of the index of the sequence $x(n)$, then the sequence value $x(n_2n_1n_0)$ is stored in the input array position $X(n_0n_1n_2)$. The advantage of this bit-reversing is the separation between the odd and even numbers. The even numbered samples are located at the top half of the module and the odd numbered samples are located at the bottom half. Figure 3.5 shows the data location clearly. Formerly, such a separation of the data can be carried out by examining the least significant bit (LSB) n_0, in the index n. Next the even and odd sequences are each sorted into their even and odd parts. Thus the necessity for bit reversed ordering of the sequence $x(n)$ is seen as a result of the manner of FFT calculation. To show the overall structure of DIT-FFT algorithm, Figures 3.6 and 3.7 provides the concept of DIT-FFT when bit-reversed is applied.

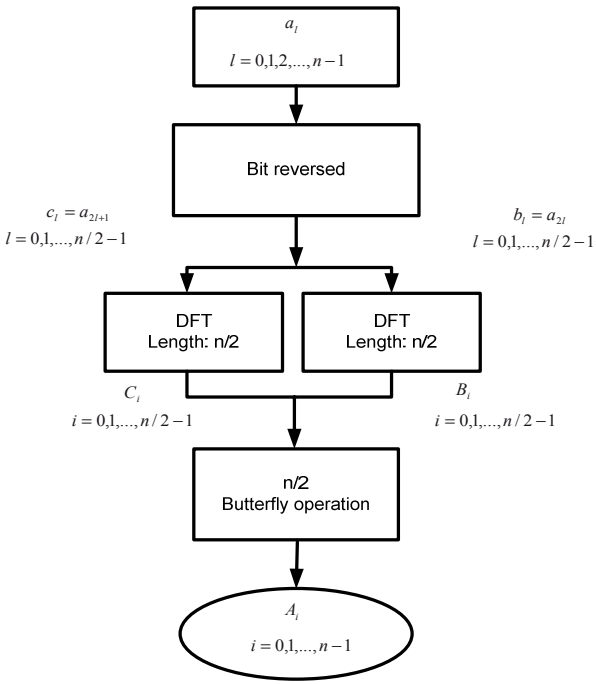


Figure 3.6 Flow chart of Radix 2 DIT-FFT Structure

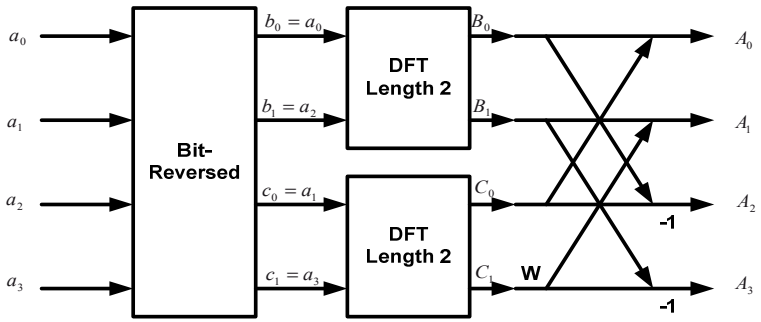


Figure 3.7 4-point Radix-2 DIT-FFT Structure

3.2.2 DIF Radix-2- Butterfly FFT Processor

DIF-FFT calculation is similar to the DIT-FFT algorithm. As far as FFT calculation is involved, the time domain sequence is divided into two sub-sequences with $N/2$ samples:

$$\{x(0), x(1), \ldots, x(N/2\text{-}1)\} \tag{3.24}$$

$$\{x(N/2), x(N/2+1), \ldots, x(N\text{-}1)\} \tag{3.25}$$

The DFT concept of $x(n)$ expressed as :

$$X(k) = \sum_{n=0}^{\left(\frac{N}{2}\right)-1} x(n) W_N^{nk} + \sum_{n=\frac{N}{2}}^{N-1} x(n)\, W_N^{nk}$$

$$= \sum_{n=0}^{\left(\frac{N}{2}\right)-1} x(n) W_N^{nk} + \sum_{n=0}^{(N/2)-1} x(n+N/2)\, W_N^{nk}\, W_N^{\left(\frac{N}{2}\right)k} \tag{3.26}$$

Given that $W_N^{\left(\frac{N}{2}\right)k} = (-1)^k$, equation (3.26) can be simplified to :

$$X(k) = \sum_{n=0}^{\left(\frac{N}{2}\right)-1} [x(n) + (-1)^k x(n + N/2)] W_N^{nk} \tag{3.27}$$

Later, Equation (3.27) will be expanded into two parts including even $X(2k)$ and odd $X(2k+1)$ samples. By using the twiddle factor characteristic, Equation (3.27) is simplified to:

$$W_n^{2kn} = W_{N/2}^{kn} \ , \ W_n^{(2k+1)n} = W_N^n W_{N/2}^{kn} \tag{3.28}$$

$$X(2k) = \sum_{n=0}^{\left(\frac{N}{2}\right)-1} \left[x(n) + x\left(n + \frac{N}{2}\right)\right] W_{N/2}^{nk} = \sum_{n=0}^{\left(\frac{N}{2}\right)-1} x_1(n) W_{N/2}^{nk} \tag{3.29}$$

$$X(2k+1) = \sum_{n=0}^{\left(\frac{N}{2}\right)-1} \left[x(n) - x\left(n + \frac{N}{2}\right)\right] W_N^{nk} W_{N/2}^{nk} = \sum_{n=0}^{\left(\frac{N}{2}\right)-1} x_2(n) W_N^n W_{N/2}^{nk}$$

$$k = 0,1, \ldots, \left(\frac{N}{2}\right) - 1 \tag{3.30}$$

Based on the explanation given, Figure 3.8 shows the heart of DIF FFT algorithm which is butterfly as well as Figure 3.9 which illustrates the concept of DIF FFT for 8-point FFT.

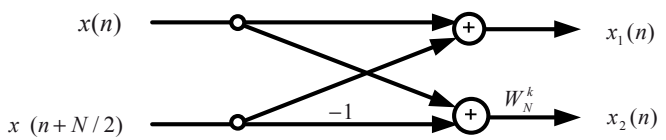


Figure 3.8 2-point butterfly in DIF-FFT algorithm

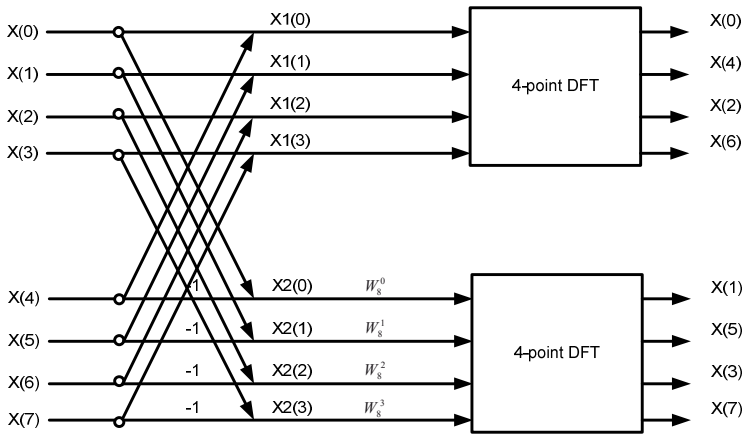


Figure 3.9 Final decomposition of 8-point DIF-FFT processor

The output sequence $X(k)$ of the DIF-FFT is bit-reversed, while the input sequence $x(n)$ of the DIT-FFT is bit-reversed. In addition, there is a slight difference in the butterfly computation. As shown in Figure 3.2, the complex multiplication is performed before the complex addition or subtraction in the DIT-FFT. In contrast, the complex subtraction is performed before the complex multiplication in the DIF-FFT as shown in Figure 3.10. The process of decomposition is continued until the last stage is reduced to the 2-point DFT. Since the frequency samples in the DIF-FFT are bit-reversed, the bit-reversal algorithm must be applied to these frequency samples in order to obtain the natural order of frequency samples. Similarly, the DIF-FFT algorithm also uses in-place computation.

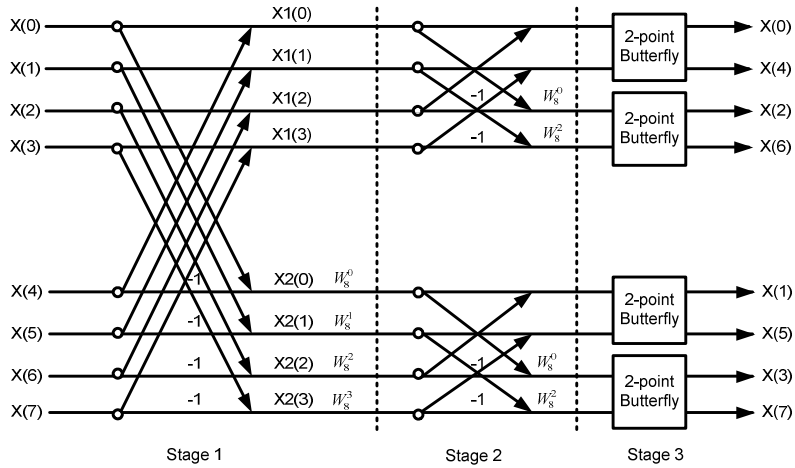


Figure 3.10 Internal calculation of 8-point DIF-FFT processor

To show the overall structure of DIF-FFT algorithm, Figures 3.11 and 3.12 provide the concept of DIF-FFT when bit-reversal is applied.

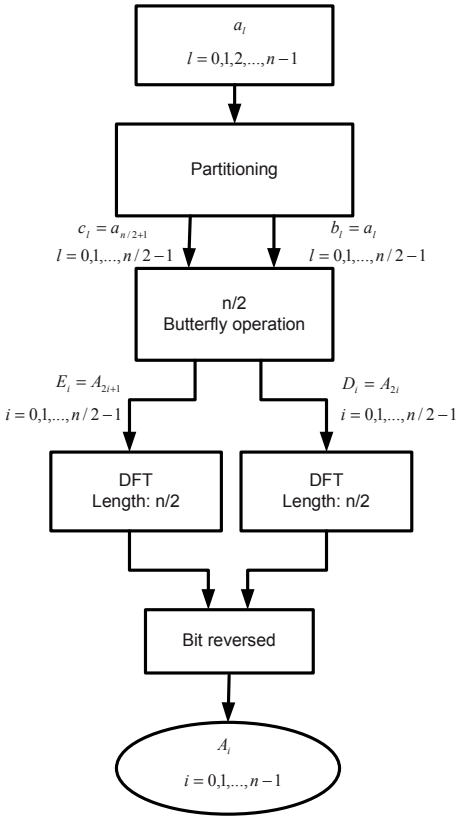


Figure 3.11 Flow chart of radix-2 DIF-FFT Structure

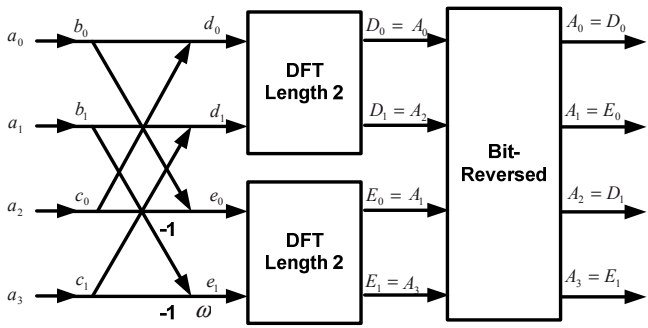


Figure 3.12 4-point radix-2 DIF-FFT structure

3.3 COMPARISON OF DIT-FFT AND DIF-FFT ARCHITECTURE

Based on the introduced DIT and DIF structures, the input data in DIT-FFT is bit-reversed while the output is in natural order. For the DIF structure, the input is in normal order while the output is reversed. However, both the DIT and DIF can go from normal to shuffled data or vice versa. Furthermore, considering the butterfly diagram, in the DIF architecture, the complex multiplication takes place after the adder/subtractor calculation.

To apply the radix-2 FFT structures, the conditions are that both the DIT and DIF algorithms require the same number of operations and bit-reversal to compute the FFT. The overall structure of the FFT processor is depending on the applications, the hardware implementation, and convenience. If the design is focused on high speed, the processor has to take the most efficient approach and algorithm to perform the FFT calculation accordingly. In this book based on design, hardware implementation and proposed algorithm, the DIT-FFT architecture is selected to function in achieving higher efficiency.

3.4 FFT PROCESSOR ARCHITECTURE

In 2009, Xilinx Logic core introduced the available FFT architecture processors. The proposed FFT processors were designed to offer a trade-off between core sizes and transform time. These architectures are classified as follow:

- FFT Processor with radix-2 pipelined serial I/O architecture
- FFT Processor with radix-4, parallel I/O (Burst) architecture
- FFT Processor with radix-2, parallel I/O (Burst) architecture
- FFT Processor Radix-2 lite, parallel I/O (Burst) architecture

The pipeline serial I/O allows continue data processing, whereas the burst parallel I/O loads and process data separately by using iterative approach. It is smaller in size than parallel but has longer transform time. In the case of radix-2 algorithm, it uses the

same iterative approach as radix-4 with the difference of smaller butterfly size that differentiates it. But the transformation time is longer. Finally for the last category, based on radix-2 architecture, this variant uses a time multiplexed approach to the butterfly for an even smaller core, at the expense of longer transformation time.

Figure 3.13 shows the throughput versus resource among the four architectures as follows: all the four architectures may be configured to utilise a fixed point interface with one of the three fixed-point arithmetic methods (un-scaled, scaled, or block floating-point).

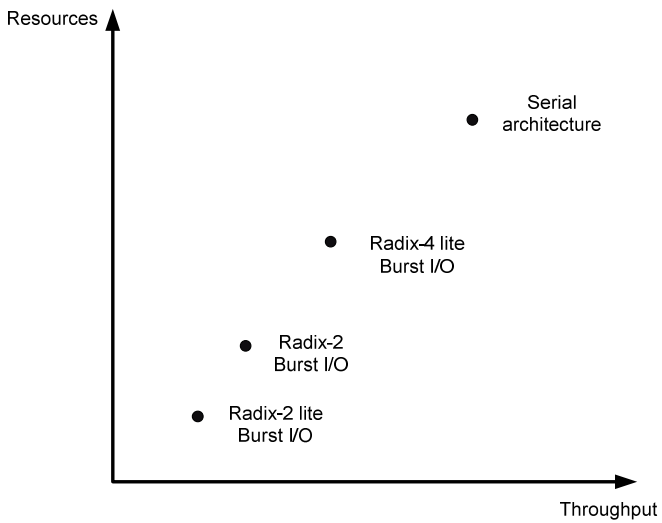


Figure 3.13 Comparison between available resources of FFT architecture

3.4.1 FFT Processor with Radix-2 Pipelined, Serial I/O

In this design the *n*-stage of radix-2 butterfly are connected as a serial structure. Each unit of radix-2 butterfly has its own RAM memory to upload and download data. The input data will be stored in the RAM while the processor simultaneously performs

transform calculations on the current frame of data and load input data for the next frame of data and unload the results of the previous frame of data. The processor has the ability of streaming data in for FFT calculations. In the scaled fixed-point mode, the data is scaled after every pair of radix-2 stages. The block floating-point mode may use significantly more resources than the scaled mode as it must maintain extra bits of precision to allow dynamic scaling. Therefore, if the input data is well understood and is unlikely to exhibit large amplitude fluctuation, using scaled arithmetic (with a suitable scaling schedule to avoid overflow in the known worst case) is sufficient and resources may be saved.

The input data is presented in natural order. The unloaded output data can either be in bit-reversed order or in natural order. When natural order output data is selected, an additional memory resource is utilized.

Figure 3.14 illustrates the architecture of the pipeline serial I/O with individual memory bank which connects in a serial structure.

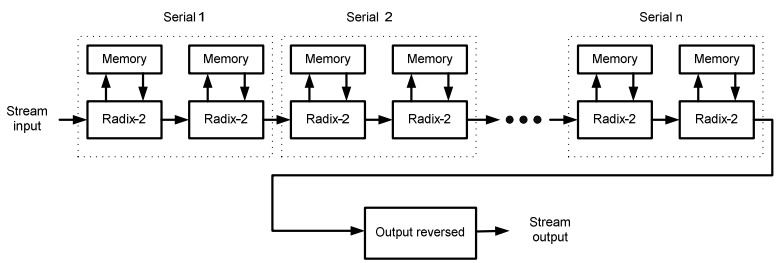


Figure 3.14 FFT processor with pipeline serial I/O Architecture
Source: Xilinx 2009

3.4.2 FFT Processor with Radix-4, Burst I/O

Radix-4 structure accepts 4 input data simultaneously whereas radix-2 takes 2 input data to perform FFT calculations. The 4 input data uploaded to the FFT processor, cannot be uploaded while the calculation is underway. When the FFT is started, the data is loaded. After a full frame has been loaded, the core computes the

transformation. The result can be downloaded after the full process is over. The data loading and unloading processes can be overlapped if the data is unloaded in digit- reversed order.

Figure 3.15 shows the radix-4 structure when 4 input data are loaded for FFT calculation.

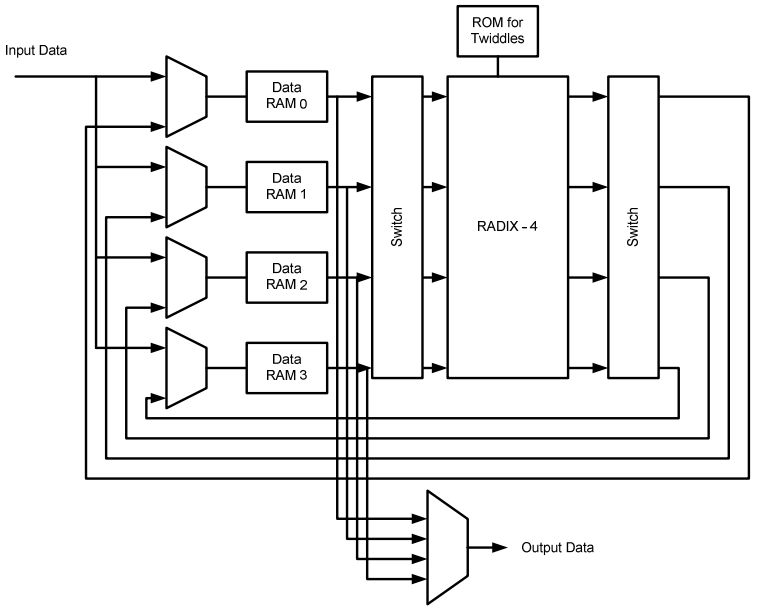


Figure 3.15 FFT Processor with radix-4 architecture
Source: Xilinx 2009

3.4.3 FFT Processor with Radix – 2, Burst I/O

The mentioned FFT processor utilises radix–2 butterfly calculation to execute FFT arithmetic structure. In spite of radix–4 with burst I/O processor, which the input data can not simultaneously load and unload, the radix-2 processor accepts the input data during the FFT processor and data can be simultaneously loaded and unloaded when

the output samples are in bit-reversed order. The twiddle factors are stored in the ROM blocks whilst the output and input data will be stored in separate or mixed RAM blocks.

Figure 3.16 shows the radix-2 structure when 2 input data are loaded for FFT calculation.

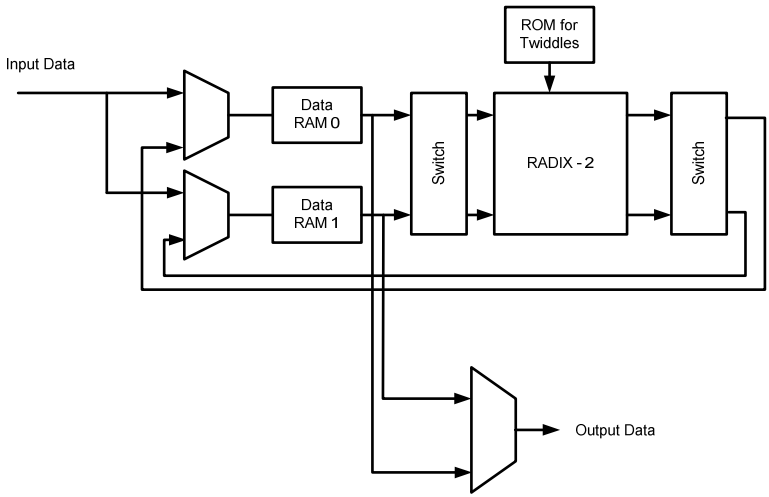


Figure 3.16 FFT Processor with radix-2 burst I/O architecture
Source: Xilinx 2009

3.4.4 FFT Processor with Radix – 2 Lite, Burst I/O

FFT processor with radix-2 Lite architecture uses one shared RAM, hence reducing resources at the expense of an additional delay per butterfly calculation. The multiplier in this structure multiplies the real part of complex number in one clock cycle and the imaginary in the next. In this architecture the data can be simultaneously loaded and unloaded if the output samples are in bit-reversed order. In this

architecture sine and cosine twiddle factor coefficient saved in the ROM and the output data will be saved in a single RAM. Although this proposed architecture saves the resources, but the throughput is significantly limited by the FFT structure due to the sequence calculations. Figure 3.17 shows the radix-2 Lite structure when two input data are loaded for FFT calculations.

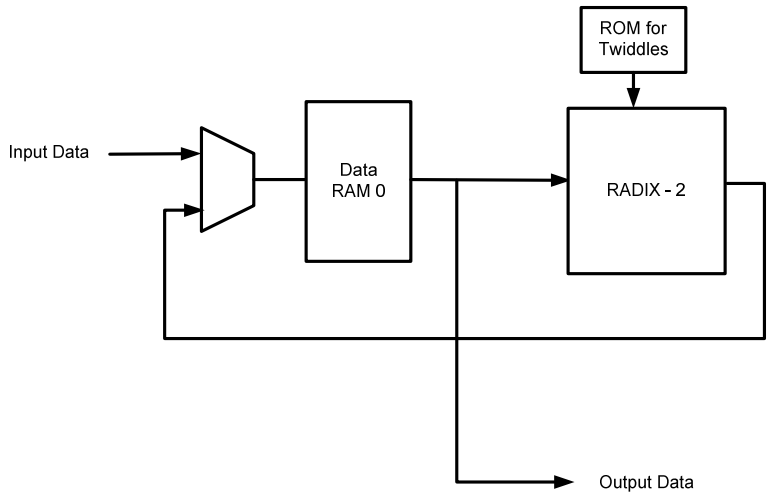


Figure 3.17 FFT Processor with radix-2 Lite burst I/O architecture
Source: Xilinx 2009

3.5 FFT PROCESSOR AND INPUT SIGNAL

In the previous section the fundamental concept of the FFT processor was introduced. For the linear shift invariant systems a representation of the input sequence as a weighted sum of delayed unit sample sequences is considered to lead to a representation of the output as a weighted sum of delayed unit sample responses. The fundamental property of linear shift-invariant system is steady–state response to a sinusoidal input and it is sinusoidal of the same frequency as the input, with amplitude and phase determined by the system. It is the property of linear shift-invariant systems

that makes representations of signals in terms of sinusoids or complex exponentials (Fourier representation) very useful in linear system theory. It is considered that the sampled rectangular signal as ideal low pass filter is entered to the FFT processor unit. Deriving the Fourier Transform of a single rectangular pulse of width τ (-*N*/2, *N*/2) and height A (Figure 3.18) produces sinusoids *sinc* function.

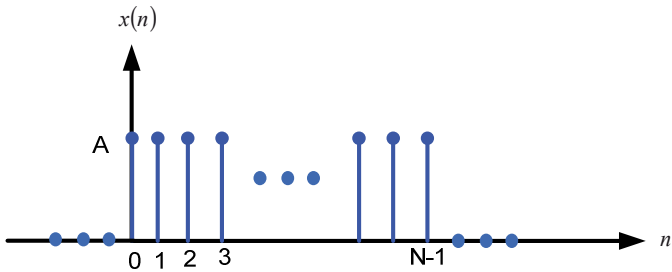


Figure 3.18 Single rectangular pulse

$$X(\omega) = \int_{-\tau/2}^{\tau/2} Ae^{-j\omega t}dt = -\frac{A}{j\omega}e^{-j\omega t}\Big|_{-\tau/2}^{\tau/2} \tag{3.31}$$

$$= \frac{2A}{\omega}\left(\frac{e^{j\omega\tau/2} - e^{-j\omega\tau/2}}{2j}\right) = A\tau\frac{\sin(\omega\tau/2)}{(\omega\tau/2)}$$

Hence the frequency response possesses the figure of *sinc* function which is shown in Figure 3.19. As *x*(*n*) is extended to a large number of periods, the *sincs* will begin to look more and more like impulses.

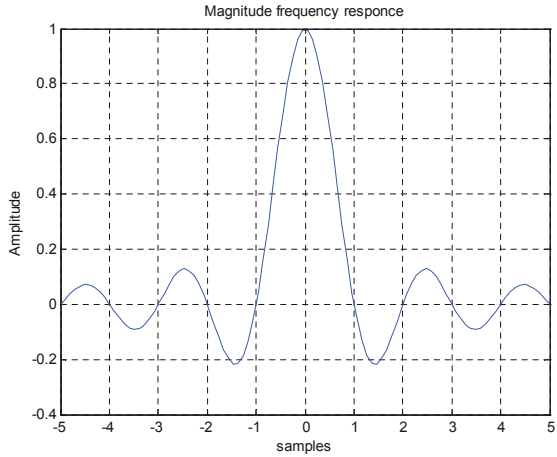


Figure 3.19 Rectangular frequency responses

3.6 SUMMARY

In this chapter the principle of the FFT was introduced. The conventional FFT as two types of DIT-FFT and DIF-FFT processor was described and a comparison was made. The chapter was described the performance of different type of FFT processors could be different based on the applications, the hardware implementation, and convenience of the particular research projects.

Furthermore, the schematic of different FFT processors were presented and its advantages were explained. Four FFT processors were introduced and identified as (i) radix-2 pipelined serial I/O architecture, (ii) radix-4, parallel I/O (Burst) architecture, (iii) radix-2 parallel I/O (Burst) architecture and finally (iv) radix-2 Lite, parallel I/O (Burst) architecture.

All the described FFT processor was based on fixed-point architecture and then, is trying to improve resolution, power consumption, speed as well as area. However moving to floating point architecture has obtained this requirement and this is the subject of the coming chapter on floating point architecture.

CHAPTER IV

THE FLOATING-POINT PARALLEL PIPELINE (FPP) RADIX-2 FFT PROCESSOR

As stated in Chapter I the objective is to design floating point 1024-point radix-2 FFT processor that has low power consumption, high-resolution and high-speed specification. This chapter elaborates and establishes the architecture in order to meet the stated requirement. This chapter is commenced with introduction to the floating-point number, and then followed by presenting the proposed architecture. Finally the performance of individual components is explained with the discussion of proposed system advantages.

The detailed hardware implementation framework of the project is indicated in Figure 4.1. The architecture of the proposed architecture is implemented by the utilisation of the VHDL and CAD Tools.

The project is continued in order to achieve the goals to enhance the previous FFT processor research work. The hardware code is transferred to the UNIX machine to check the Fourier transform properties in gate level by CAD tools (Synopsys and Cadence). The workflow will be executed to the next stage if the design meets the desirable specifications, which is the FPGA prototype and testing. Concurrently to that, the design is synthesized, the timing verified and finally simulated at gate level on the 0.18 SILTERRA and 0.35 MIMOS process technology.

The comparison was also done between these two technologies in terms of their performance (speed, power consumption, area). Based on the architecture and design specifications, the hardware design flowchart of the proposed radix-2 FFT processor is illustrated in Figure 4.1.

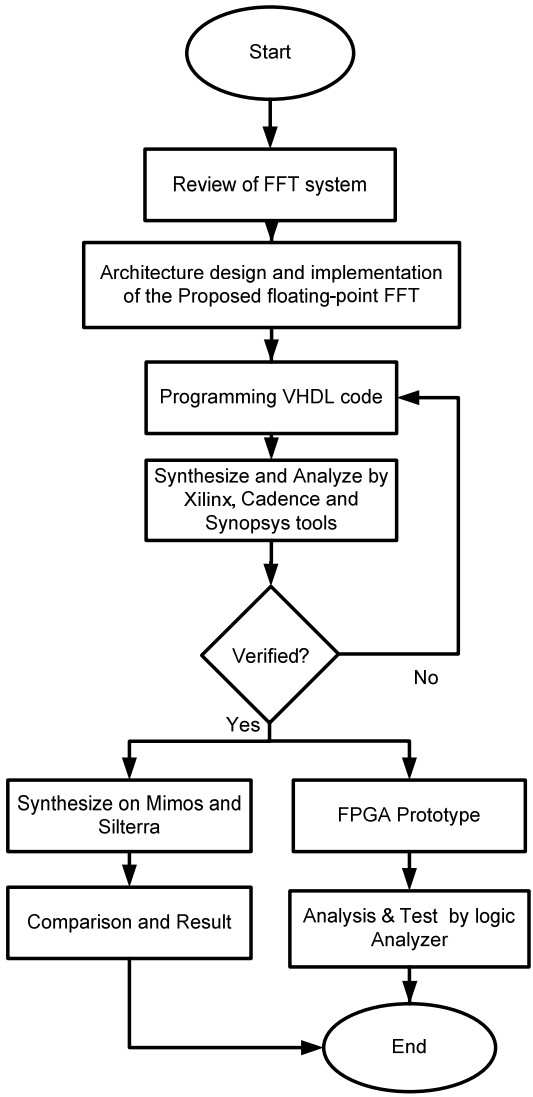


Figure 4.1 Hardware design methodology framework

Hence the focus is on achieving high performance floating-point FFT processor architecture to cover signal processing in harmonic analyzing applications. The

system is designed and implemented to achieve high resolution and high dynamic range and at the same time meet fixed-point specifications.

4.1 DATA STRUCTURE

In realising the floating-point-based arithmetic for the implementation on the 1024 point FFT processor, it is thus necessary for the data to be presented in floating-point format. Hence implementation of floating-point register and biased exponent are discussed in detail in this chapter. It is followed by design architecture of the proposed floating-point parallel pipeline 1024 point (FPP) FFT processor.

4.1.1 Floating – Point Data Format

In floating-point format, the data is translated based on power and mantissa in decimal system. This notation can be expanded into the binary system. Representing data in power and mantissa system gives the storing data capability a much greater range of numbers than if the binary points were fixed. Numbers are represented approximately to a fixed number of significant digits and scaled using an exponent. The base for the scaling is normally 2, 10 or 16. The typical number that can be represented exactly is of the form:

$$Significant\ digits \times base^{\,exponent}$$

Floating-point refers to the 'truth' that the radix point; which refers to the decimal point or in computers is known as the binary point, has the capability to float. This entails the event to occur anywhere that is relative to the significant digit of the number. Thus, a floating-point representation, with its position indicated separately in the internal representation is a computer's recognition of a scientific concept.

Although the benefit of floating-point representation over fixed-point (and integer) representation is much wider range of values, but the floating-point format needs slightly more storage (to encode the position of the radix point), hence when

stored in the same space, floating-point numbers achieve greater range at the expense of precision.

In addition, the speed of floating-point operations is an important factor of performance for computers in DSP application. Hence the implementation of high performance system requires applying efficient and fast floating-point processor which is competitive with the fixed-point processor. Various types of floating-point representation have been used in computers in the past. However in the last decade, the IEEE 754 standard has defined the representation.

According to the IEEE 754 standard, the single-precision is chosen to represent the floating-point data. The IEEE standard specifies a way in which the three values described can be represented in a 32-bit or a 64-bit binary number, referred to single and double precision, respectively. In this project single precision is selected to function.

For the 32-bit numbers, the first bit (MSB) specifies the sign, followed by 8 bits for the exponent, and the remaining 23 bits are used for the mantissa. This arrangement is illustrated in Figure 4.2. The sign bit is set to zero if the number is positive, and the bit is set to 1 if the number is negative. The mantissa bits are set to the fractional part of the mantissa in the original number in bits 22 to 0.

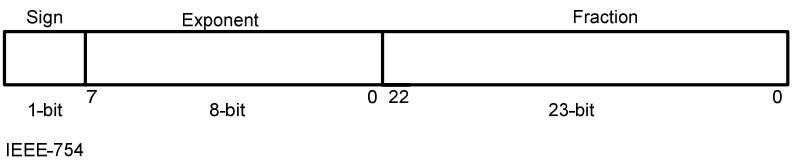


Figure 4.2 32-bit floating-point registers

4.1.2 Biased Exponent

The exponent stored in floating-point format had to be in biased structure. Biased format ensures no negative power is saved in the floating-point register. The exponent is biased by $(2^{e-1}) - 1$, where *e* is the number of bits used for the exponent in floating-point register. Biasing is done because exponents have to be signed values in order to be able to represent both tiny and huge values. It turns out that biasing the exponent results in simpler circuitry for computing the magnitude of the two floating-point numbers. The two's complement, the usual representation for signed values, would make comparison harder. Hence, the exponent is biased before being stored by adjusting its value to put it within an unsigned range suitable for comparison.

The mantissa is normalized to binary representation of the number which is multiplied by 2 raised to the power defined by the exponent. As an example to encode 118.625 as a float, the sign bit assign by zero. To find the exponent and mantissa, first the number is converted to the binary format which is 1110110.101. Next normalize the number to 1.110110101×2^6, which is the binary equivalent of scientific notation. The exponent is 6 and the mantissa is 1.110110101.

The exponent must be biased, which is 6+127 = 133. The binary representation of 133 is 10000101. Thus the floating-point encoded value of 118.625 is 0100 0010 1111 0110 1010 0000 0000 0000. Binary values are often referred to in their hexadecimal equivalent which is 42F6A000.

In this project, it is assumed the mantissas are shifted after the point and the exponents have been increased. It means that the integer value of the fraction number is equal to zero. (e.g. $[9.5]_{10} = [0.10011 \times 2^4]_2$)

4.2 STAGE REALISATION OF 1024 POINT FPP- FFT PROCESSOR

Since the main purpose of this project is to design and implement the high performance 1024-point floating based radix-2 FFT processor algorithm that

conserves energy that most satisfies the constraints, selecting the efficient algorithm (Figures 3.14-3.17) has large impact on the performance of the proposed FFT processors.

Since the introduced algorithm can be used for computation of fixed-point architecture, in this project, the effort is taken to present a new algorithm that supports the floating-point architecture.

In spite of existing fixed point FFT architecture, the proposed radix-2 FPP-FFT is freed of having a local memory for each individual part (Figure 3.14) since it has a shared memory application. A fine-grained distribution of loops that will implement the FFT functions at each level will result in effective parallelization. The proposed method proves high performance of the 1024 point FPP-FFT algorithm and its architecture will be discussed in detail. The principle architecture is based on using a radix-2 algorithm in floating-point format to calculate 1024 point FFT structure. The proposed processor takes the advantages of (i) shared memory to store the input and output data and makes the system as single chip. Hence, it reduces hardware complexity. Furthermore (ii) the entire individual arithmetic unit are designed to operate within 1 clock cycle to increase the maximum clock frequency. Additionally (iii) the butterfly structures is in parallel and pipelined architecture to minimize delay caused by the FFT calculations and finally (iv) the strong controller with collaboration of address generator unit ignores the need of using N numbers of butterfly unit, since radix-2 calculation is carried out within one butterfly unit that results reduction of power consumption, area and avoid system complexity. The proposed design is implemented with optimizing the previous processor architecture (Figures 3.14-3.17) to enable the system in maintaining a reasonable clock rate. The throughput of the operation is limited by the amount of available logic in the target device. Figure 4.3 illustrates the main block diagram of the 1024-point radix 2 FPP-FFT processor in detail.

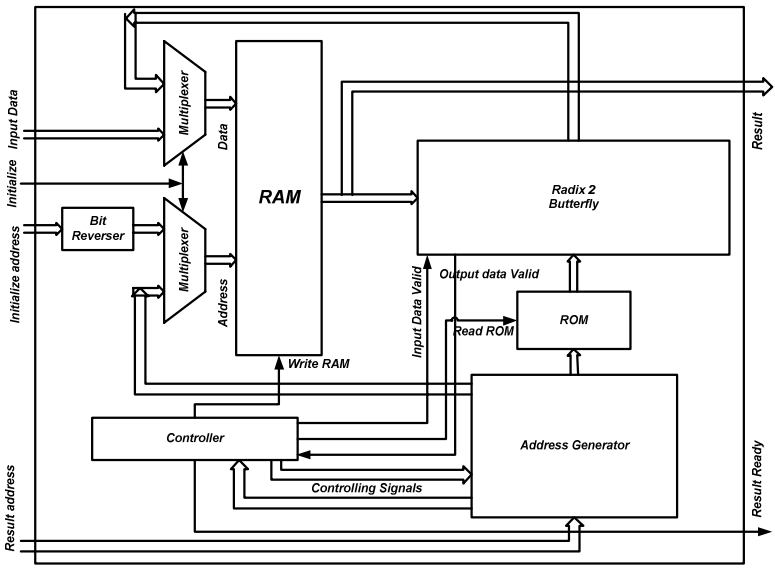


Figure 4.3 1024 point radix-2 FPP-FFT block diagram

As shown in Figure 4.3, there are six major building units in the proposed 1024 point radix-2 FPP-FFT architecture. These units are shared memory unit, bit reverse, butterfly arithmetic unit, smart controller, ROM stage and finally address generator unit. The floating-point input data acts as a variable streaming architecture. The variable streaming architecture allows continuous streaming of input data and produces continuous stream of output data. In the next section, each unit will be investigated separately. Detailed design of the critical components is presented in next section.

4.2.1 Bit Reverse

The bit-reverse block parses the interleaved data sample into reverse set of serial data before entering into the FFT processor. Figure 4.4 shows the re-arrangement pattern required for the *n*-bit bit reverse block. The input data is the sampled original signal

and in the output data is reversal of the input data. The FFT time domain decomposition is carried out by a bit reversal sorting algorithm. This involves re-arranging the order of the *N* time domain samples by counting the bits in binary.

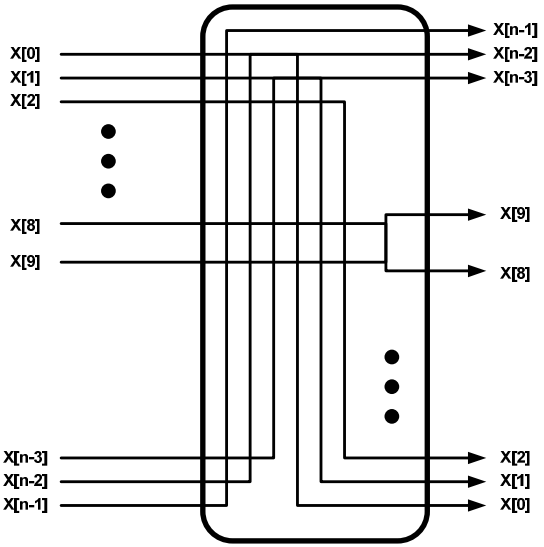


Figure 4.4 Bit-reverse block

4.2.2 Radix-2 Butterfly Architecture

The difference between proposed design and the conventional type relies on the architecture of the data path and the technique of butterfly calculations. The proposed design achieved high efficiency and high resolution FFT calculation within a limited time by utilizing the unique parallel and pipeline techniques and employing the floating-point approach. With the new architecture, the area was reduced due to having only one butterfly structure in the processor system. An intelligent DSP controller with collaboration of radix-2 butterfly makes the FFT calculation possible due to the supervision of data path. Furthermore, the FFT calculation process requires

that the twiddle factor terms (of the $W_N^K = s^{j2\pi\frac{k}{N}}$ type) be available in some time, where *k* ranges from 0 to *(N/2)-1*.

Figure 4.5 shows the internal schematic of the pipeline butterfly algorithm with the parallel architecture whereas Figure 4.6 shows the implementation view of the butterfly architecture. To improve speed calculations in the radix-2 butterfly algorithm, the pipeline registers are located after each addition, subtraction and multiplication blocks. Hence, the pipeline butterfly algorithm keeps the final result in the register to be transferred to the RAM by the next clock cycle. In addition, the parallel architecture splits the data in real and imaginary format and increase the speed of FFT calculation by 50%. This radix-2 FPP-FFT algorithm is exactly executed after $(N/2\,Log_2 N)+11$ clock a pulse which proves the improvement in compare with conventional radix-2 FFT processor. The 11 clock pulses delay is created by 11 pipeline registers in adder, subtractor and multiplier in a serial butterfly block. Additionally, parallel design of the FFT algorithm decreases the calculation time significantly.

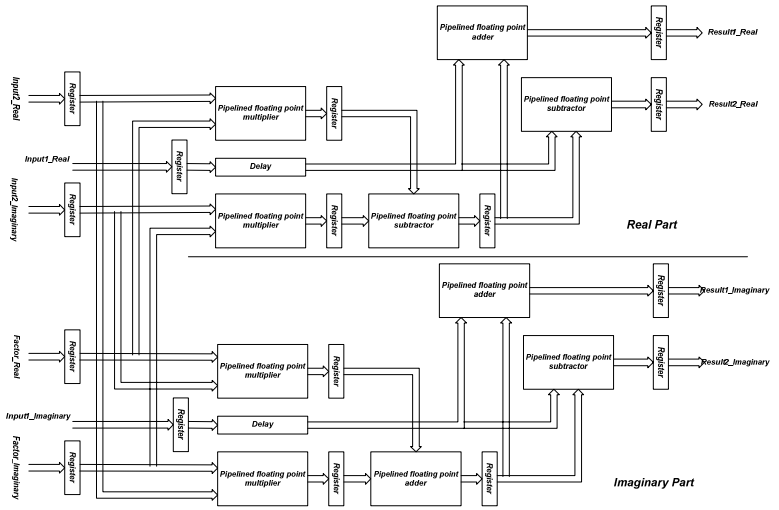


Figure 4.5 Proposed Radix-2 butterfly structure

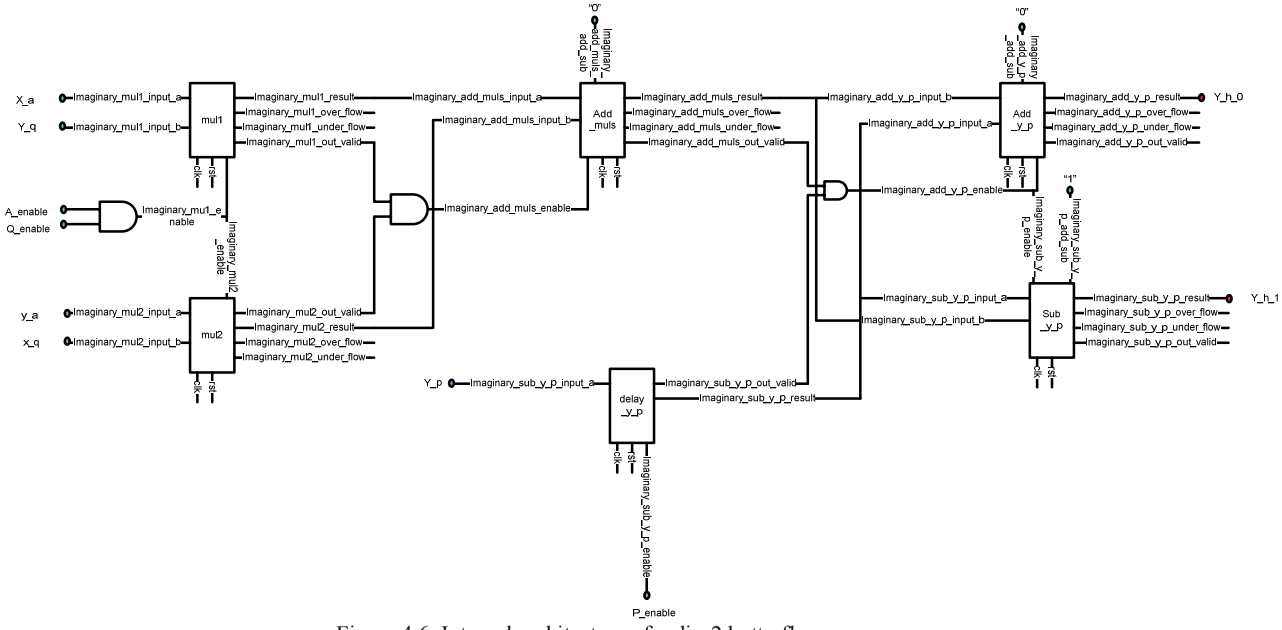


Figure 4.6 Internal architecture of radix-2 butterfly

The radix-2 butterfly unit is responsible for calculating the complex butterfly equations.

$$output1 = input1 + W^k \times input2 \tag{4.1}$$

$$output2 = input1 - W^k \times input2 \tag{4.2}$$

To calculate the butterfly equations, it is necessary to initiate the RAM. An external processor loads the data in the RAM in bit-reverse format. As shown in the Figure 4.3, the radix-2 butterfly unit is designed based on DIT-FFT architecture. If $output1 = X_{01} + iY_{o1}$, $output2 = X_{o2} + iY_{o2}$ and $W^k = X_{w^k} + iY_{W^k}$ inputting into Equations (4.1) and (4.2),we have:

$$X_{o1} + iY_{o1} = \{X_{i1} + (X_{W^k} \times X_{i2} - Y_{W^k} \times Y_{i2})\} + i\{Y_{i1} + (X_{W^k} \times Y_{i2} + Y_{W^k} \times X_{i2})\} \tag{4.3}$$

$$X_{o2} + iY_{o2} = \{X_{i1} - (X_{W^k} \times X_{i2} - Y_{W^k} \times Y_{i2})\} + i\{Y_{i1} - (X_{W^k} \times Y_{i2} + Y_{W^k} \times X_{i2})\} \tag{4.4}$$

Thus, each butterfly requires four multiplication units (two for the real and two for the imaginary) and six addition units (three for the real and three for the imaginary part). If the above equations are implemented using fixed-point calculations, the error is expected to be high due to round-off, overflow and coefficient quantization errors (Ifeachor & Jervis 2002). Thus in order to reduce the error as well as to achieve high-resolution output, the floating-point adder and subtractor are used to replace all the fixed-point adder and subtractor.

4.2.3 Proposed Floating Point Adder/Subtractor

The performance of the butterfly is greatly depending on its arithmetic units. Hence, this section will emphasize on the structures and focus on the design and implementation of efficient high performance floating-point adder/subtractor for the FFT algorithm.

By taking into consideration the previous research for the floating-point adder/subtractor, new architecture is proposed to function in arithmetic unit. Based on IEEE-754 standard (1985) for floating-point arithmetic, 32-bit data register is considered to allocate mantissa, exponent and sign bit in a portion of 23, 8 and 1 bit respectively. The advantages of this adder are that (i) the bias power is applied to complete the calculations and avoid using unsigned values.

Additionally (ii) the floating-point adder unit performs the addition and subtraction using substantially the same hardware as used for the floating-point operations. This minimizes the core area by minimizing the number of elements. Furthermore, (iii) each block of the proposed floating-point adder/subtractor operate the arithmetic calculation within only one clock cycle that results high throughput and low latency for entire proposed FFT processor.

Figure 4.7 shows the new structure of the floating-point adder when it is divided into four separate blocks. The algorithm of the proposed floating-point adder is presented in Figures 4.8 and 4.9.

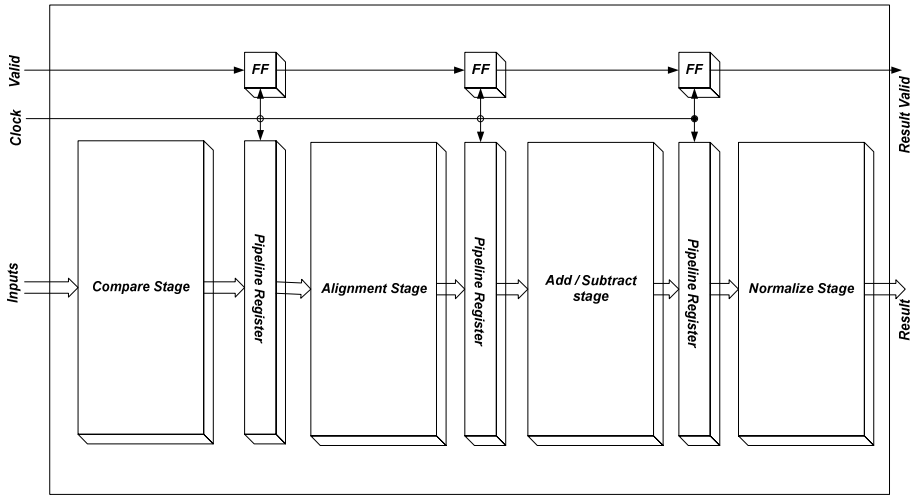


Figure 4.7 The schematic diagram of floating-point adder

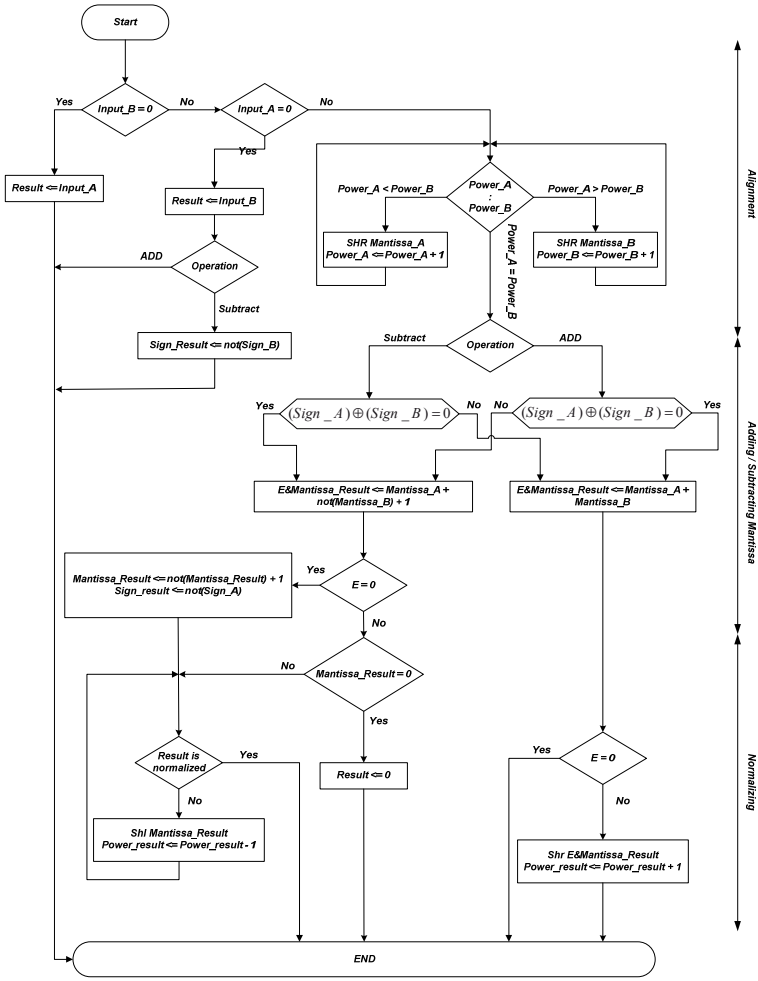


Figure 4.8 Floating-point adder algorithm

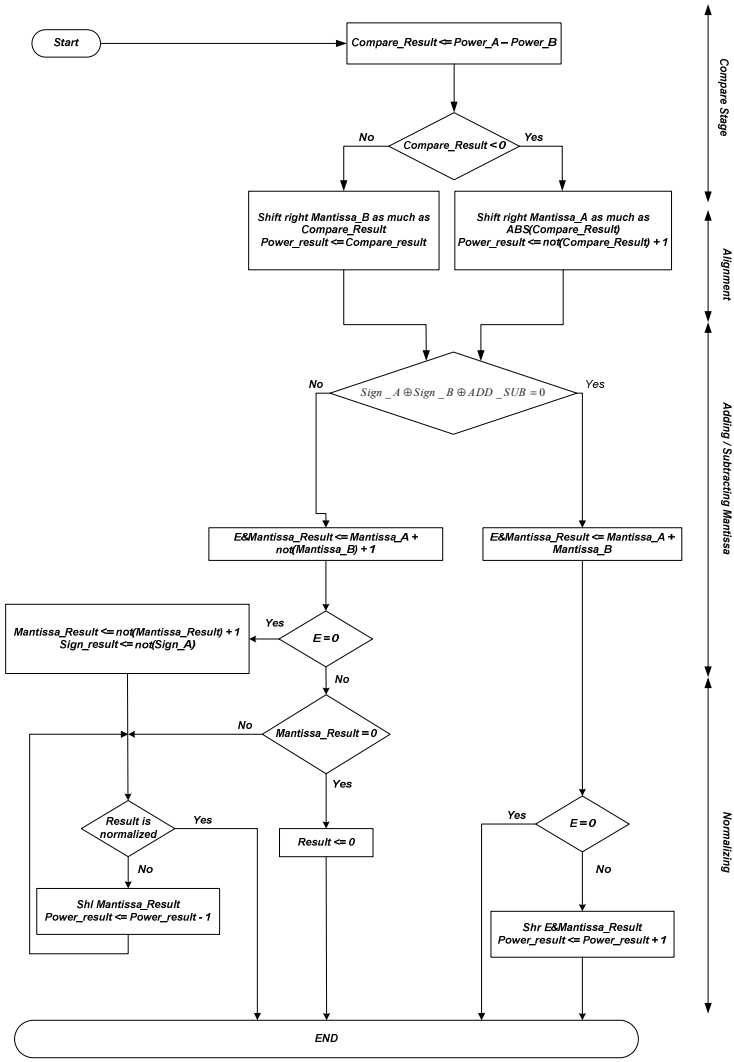


Figure 4.9 Optimized floating-point adder algorithm

The purpose of having separate blocks is to share the total critical path delay into three equal blocks. As discussed before, these blocks calculate the arithmetic function within one clock cycle. However the propagation delay can be associated with continuous assignment (Ciletti 2003) to increase the overall critical path delay and for the slowing down of the throughput. Hence in this design, the effort is taken to reduce the delay for the arithmetic calculations.

Based on combinational circuit design, the output of each stage depends on its input value at the time. The unique structure of this adder enables feeding of the output result in the pipeline registers after every clock cycles. Hence, the sequential structure is applied for the overall pipelined add/subtractor algorithm to combine the stages. The processing flow of the alternative floating-point addition/subtraction operation consists of comparison, alignment, addition/subtraction, and normalization stages.

a) Comparison Stage

The comparison stage, which compares two input exponents, is shown in Figure 4.10. This unit compares the exponent *A* and *B* and gives the result of the next stage. This comparison is performed by two subtractors and the result is declared by *compare_sign* bit.

The internal architecture of the comparison stage with the consideration of the minimum gate cell used is illustrated in Figure 4.11.

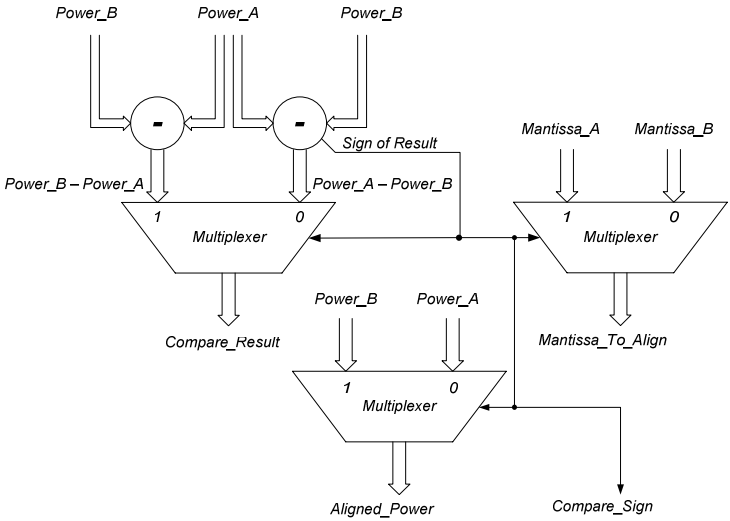


Figure 4.10 The schematic of comparison stage structure

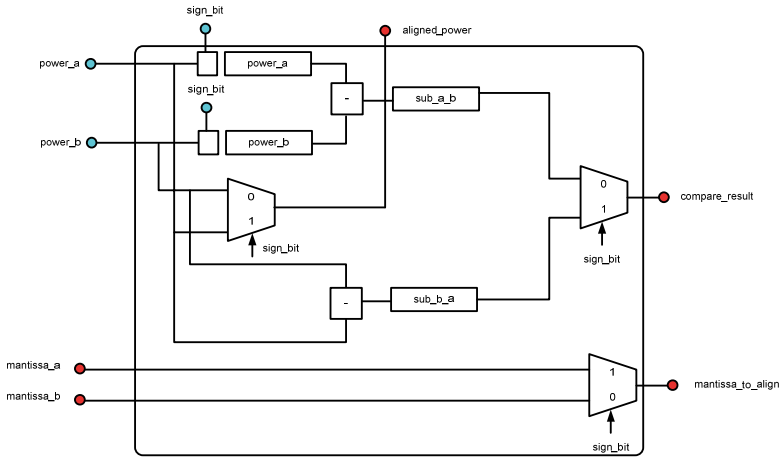


Figure 4.11 The internal architecture of comparison stage

b) Alignment Stage

While the effective operation is determined by the components, the comparison stage and alignment stage lead us to align the mantissa and exponent accordingly to warrant having the same exponent in two operands. The significant point is to design the function so that all calculation performs within one clock cycle. The basic operation of the aligned mantissa and normalized block is 'shifting'. Every shifting needs one clock cycle which causes huge propagation delay to align 32-bit operand. Hence, the advanced algorithm is designed to avoid having high latency on aligning stages.

According to the data result by the comparison stage, the alignment stage shifts the mantissa and transfers it to the adder/subtractor stage. The number of shifting will be selected by the comparison stage output. However every stage of the proposed floating-point adder algorithm is executed within one clock cycle. Figure 4.12 illustrates the alignment stage structure whilst the internal architecture is shown in Figure 4.13.

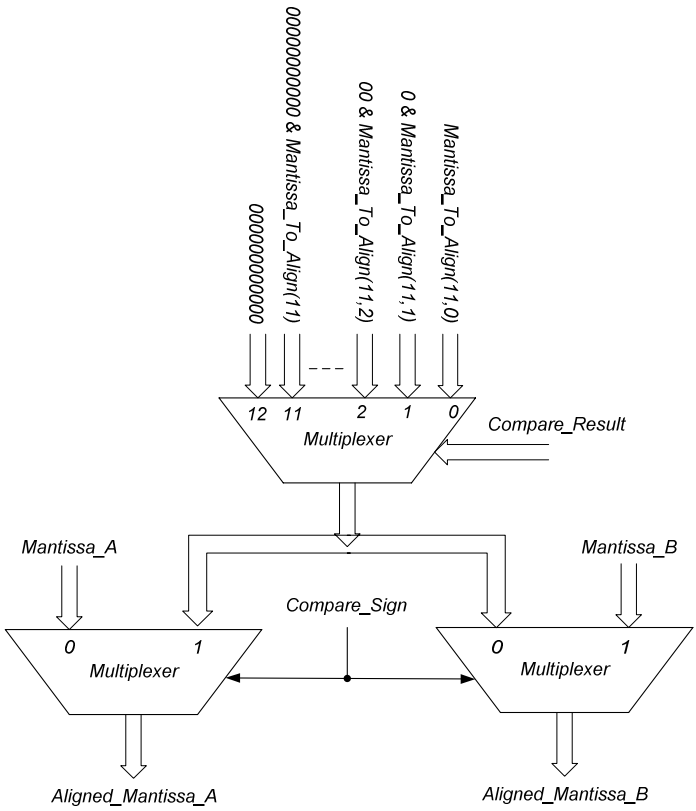


Figure 4.12 The schematic diagram of the alignment unit

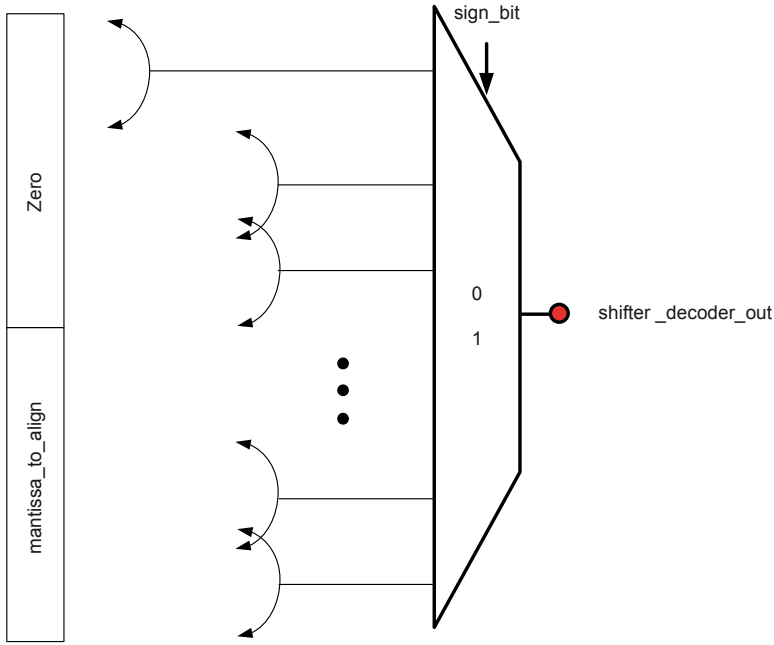


Figure 4.13 The multiplexer architecture of the alignment stage

c) Addition Subtraction Stage

A model for the proposed adder/subtractor unit which can perform in the floating-point adder/subtractor structure is presented in this section. The major requirements and structure to achieve this aim are described and algebraically verified. In general, the adder/subtractor unit sacrifices a large space of the wafer and power consumption to obtain high speed especially in the floating-point data structure. Furthermore, it is necessary to use additional execution time and hardware logics for the renormalization in the next stage. Thus the performance improvement and cost-effective design is achieved by optimizing the component accordingly.

The proposed design is implemented to calculate both the positive and negative mantissa by taking complement format. As shown in Figure 4.14 and 4.15, there are logic gates involved with the stages which cause higher delay propagation through the circuit.

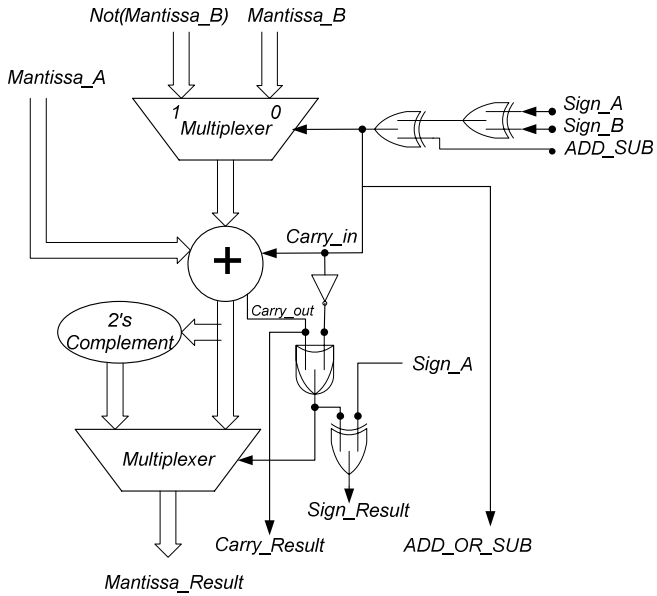


Figure 4.14 The schematic of addition/subtraction unit

As shown in Figures 4.14 and 4.15, the smart algorithm is designed for add/sub unit to utilize two's complement for the operand if it is required and combined addition and subtraction in the same hardware block diagram. In addition to save the power consumption, the proposed architecture offers power savings due to the simplification of the data paths. By ignoring the shifting blocks and combining the adder and subtractor the power-effectiveness in the overall system is significantly reduced.

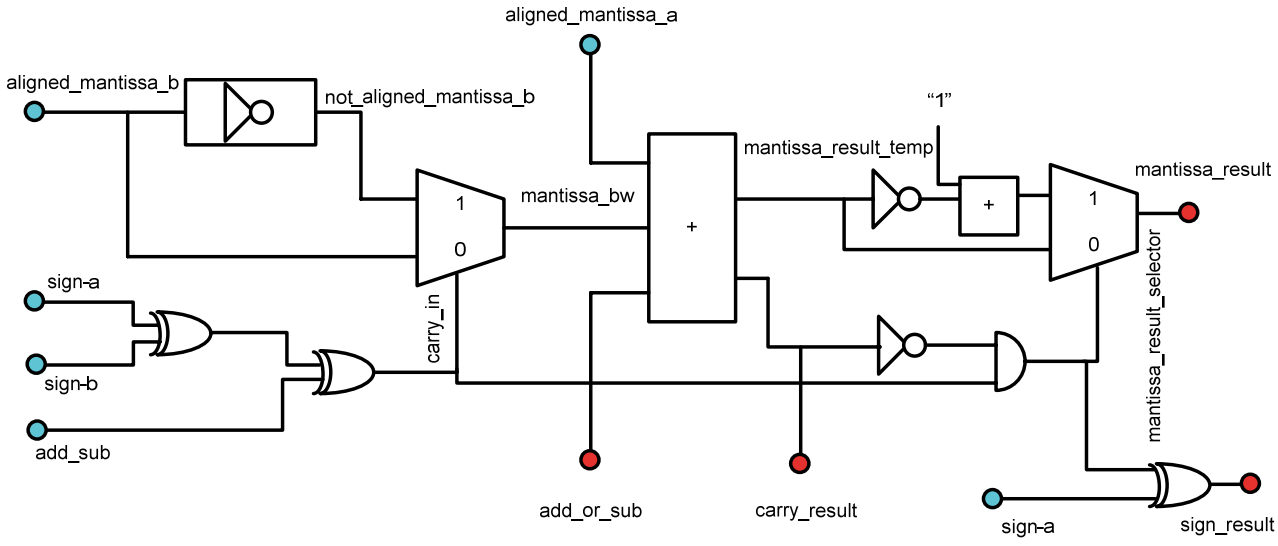


Figure 4.15 The addition/subtraction unit architecture

In Figure 4.15, if *carry_result* is equal to "0", it means the *mantissa_result_selector* output is dependent on *carry_in*. The *mantissa_result_temp* will be selected if the carry_in is equal to "0" where the *mantissa_result_temp* 2'complement will be selected in vice versa situation. The output will be *mantissa_result_temp* automatically if the *carry_result* is equal to "1".

The *Sign_result* output is dependent on the *sign_a* and *sign_b*. This means that if the *carry_result* turns to "0", the *sign_result* will be selected by the *carry_in*. In addition if the *carry_in* is equal to "0", this means that the *sign_b* is positive and the *sign_result* will be chosen by *sign_a*. The *carry_in* value is "1" with *sign_b* ~~is~~ negative and *sign_a* which will hence clarify the final sign.

d) Normalized Stage

Floating-point numbers are generally stored in registers as normalized numbers. This means that the most significant bit of the mantissa has a non-zero value. Employing this method allows the most accurate value of a number to be stored in a register. For this purpose, the normalized stage is required. This unit is located after the add/sub stage. The output signal representing the add/sub block leads to zero digits of an un-normalized result of the calculation operation. The normalized block ignores the digital value of zero from the MSB of the mantissa and shifts the mantissa to imply value of one in digital as MSB in mantissa.

Figure 4.16 shows the internal structure of the normalized stage. In this architecture the multiplexer with different figures is replaced with the shifter to avoid propagating delay all over the system due to the shifting section. However this structure still creates certain delay but it compensates the significant number of shifting delay in 23 bit mantissa and speed up the system calculations.

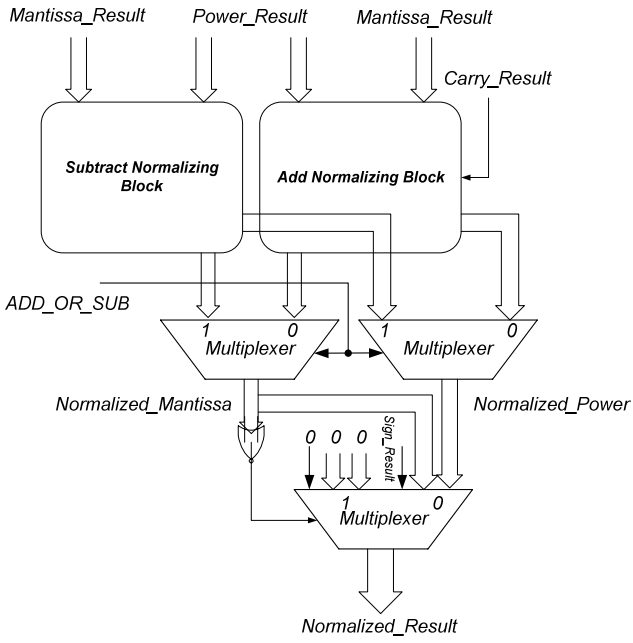


Figure 4.16 The schematic of normalized unit structure

In this structure *sub_normalized_mantissa, sub_normalized_power, add_normalized_mantissa* and *add_normalize_ power* are calculated separately and they are transferred to the last multiplexer. The result will be scaled depending on the *add_or_sub* control pin and outputted to the next stage. Figure 4.17 illustrates the internal architecture of normalized unit in detail.

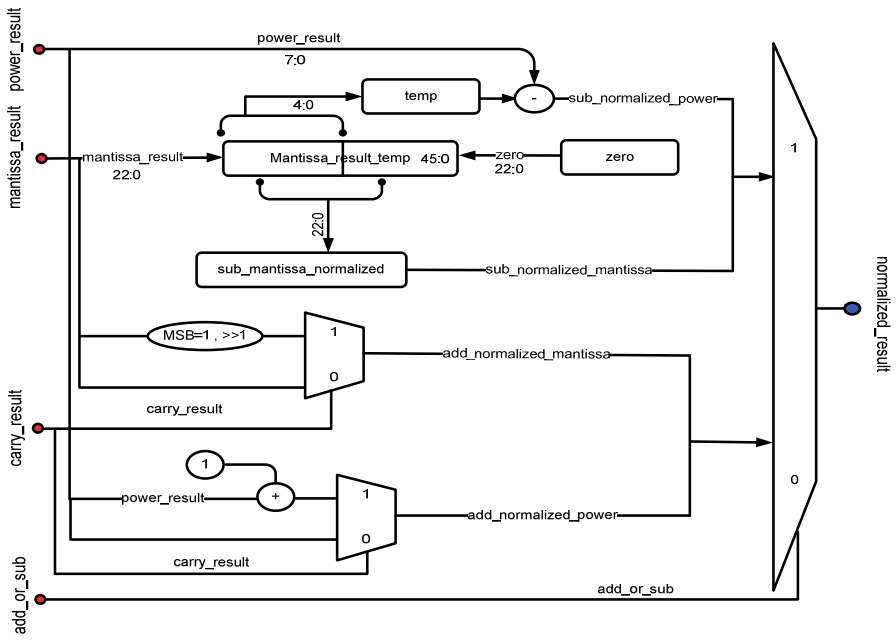


Figure 4.17 The internal architecture of normalized unit

e) Combination of Floating point Adder/Sub Unit

The combinations of the comparison stage, alignment stage, addition/subtraction stage and normalized stage as sub-units of the proposed pipeline floating-point adder/subtractor provides the fast efficient pipelined arithmetic unit to achieve the minimum latency for arithmetic calculation. The adder architecture is converted to pipeline architecture by locating the I/O pipeline register in the circuit architecture.

Figure 4.18 shows the input/output (I/O) structure of the proposed pipeline floating-point adder/subtractor.

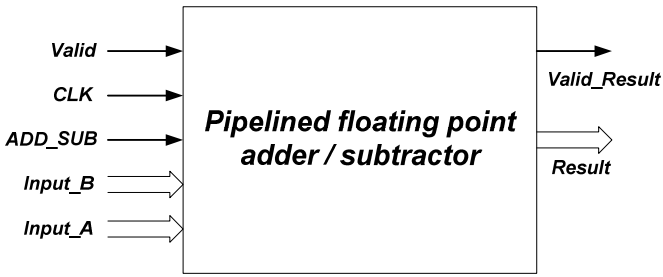


Figure 4.18 I/O structure of the proposed floating-point adder

Figure 4.19 indicates the architecture of the proposed combination pipeline floating-point adder/subtraction when the single precision is applied. The output of the sequential adder will appear within four clock cycles when the input operand and clock are asserted to the system.

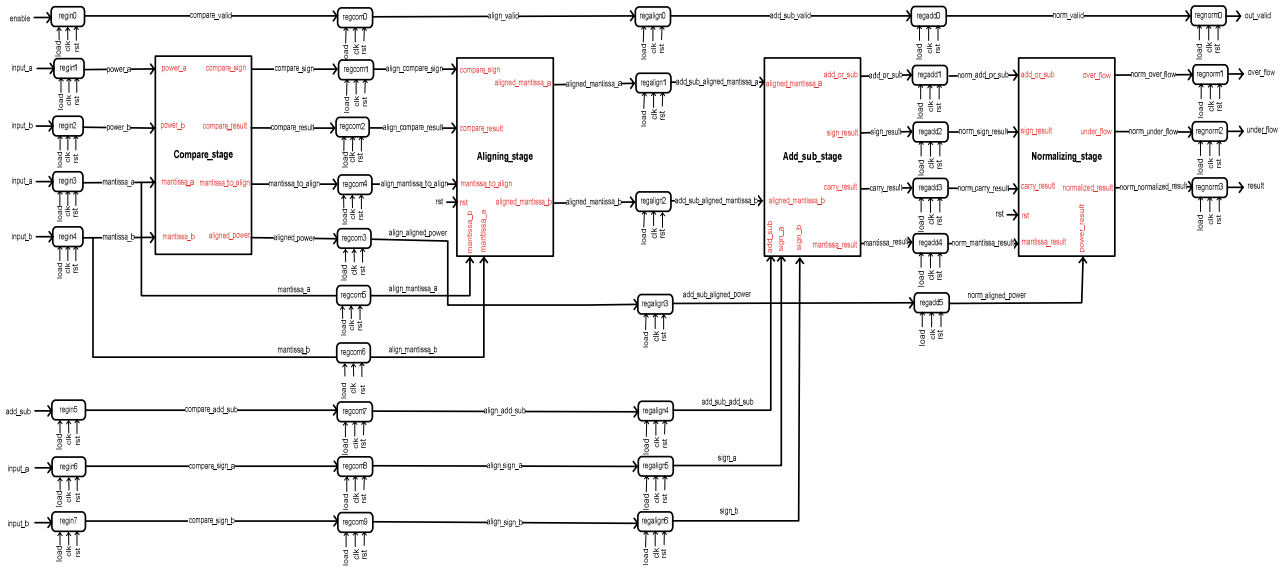


Figure 4.19 The proposed pipeline floating-point adder/subtractor architecture

4.2.4 Proposed Floating –Point Multiplier

The number representation is based on the same principles as the single-precision and double-precision formats defined by the IEEE 754 standard. However, the multiplier shown here uses a single precision normalized mantissa and eight-bit exponent only. The project in hand has developed the architecture for partial-product reduction for the IEEE standard floating-point multiplication, leading to a structured high speed floating-point multiplier. The shortening of the data path is desirable because they require shorter wires and therefore support faster operation. The former approach uses a reduction scheme based on combination unit and connects it as parallel architecture. Implementing floating-point multiplier is easier than floating-point adder since it does not require alignment stage.

The processing flow of the alternative floating-point multiplication operation consists of multiply stage and normalized stage. Figure 4.20 shows the overall block diagram of the proposed multiplier whist the flowchart of the multiplier structure constructed is shown in Figure 4.21.

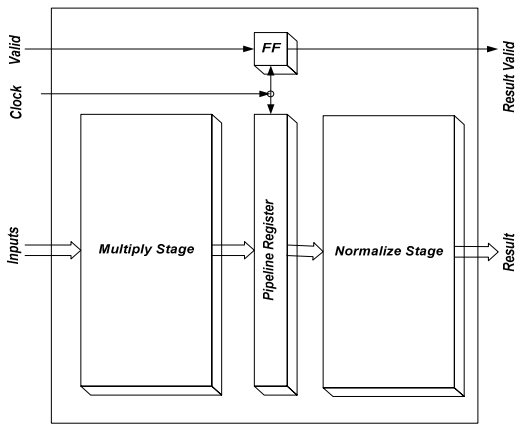


Figure 4.20 The schematic diagram of floating-point multiplier

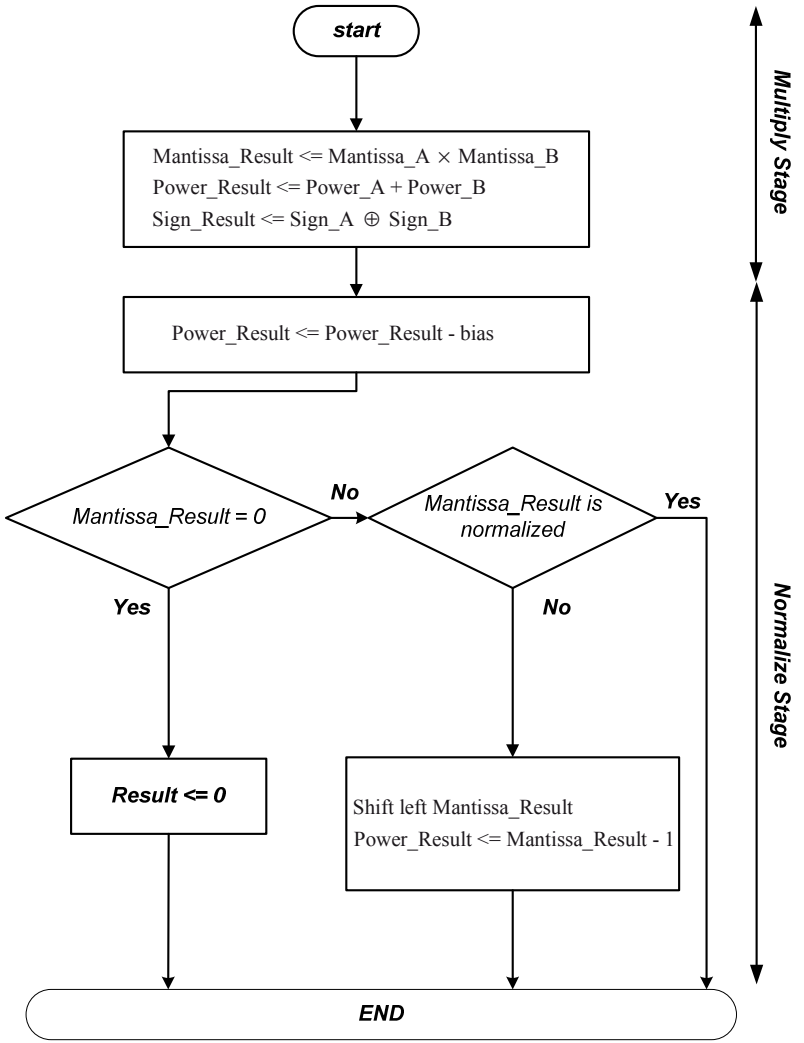


Figure 4.21 The flow chart of floating-point multiplier

As it is clear in the Figure 4.21, the bias power format is applied to avoid having negative exponent in the data structure. Additionally, the multiplier is pipelined, so that the initial result appears after the latency period where the result can then be obtained after every clock cycle. Figure 4.22 shows the schematic symbol of the proposed floating-point multiplier. This multiplier offers low latency and high throughput and is IEEE 754 compliant. This design allows a trade-off between the clock frequency and the overall latency by adding the pipeline stage.

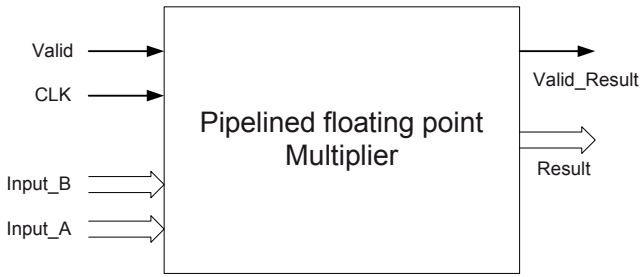


Figure 4.22 The schematic symbol of the proposed floating-point multiplier

a) Multiply Stage

The floating-point multiply stage is designed to perform single precision multiplication represented in the IEEE 754 standard. The multiply stage consists of the three units as sign, exponent and mantissa which work in parallel. In the multiply stage while the exponent of input operand accumulates, the mantissa will multiply consequently. The output sign is the XOR of two sign bit input.

The internal architecture of pipeline floating-point multiplier is shown in Figure 4.23. Note that the multiplier uses the bias exponent when encoding the result of multiplication, as long as this does not lead to a loss of precision.

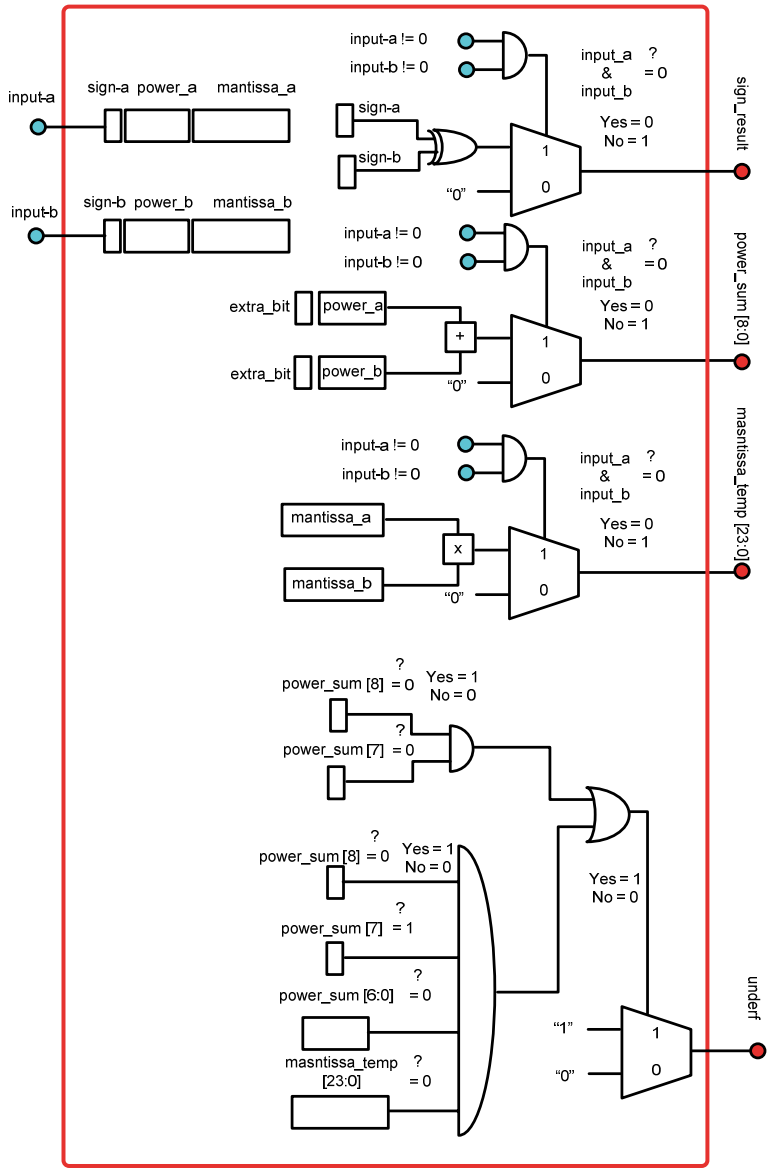


Figure 4.23 The schematic architecture of the proposed floating-point multiplier

b) Normalized Stage

The second stage of the floating-point multiplier performs the normalization of the output obtained from the first stage. The normalization is the last and most complicated part of all the stages. This unit initially calculates the amount that the mantissa needed to shift to the left. In order to avoid the delay propagation in the circuit, the multiplexer is designed to perform the shifting within one clock cycle. Figure 4.24 illustrates the structure of the normalized stage in detail. Furthermore, the architecture of the normalized stage and the overall pipelined floating-point multiplier are given in Figures 4.25 and 4.26 respectively.

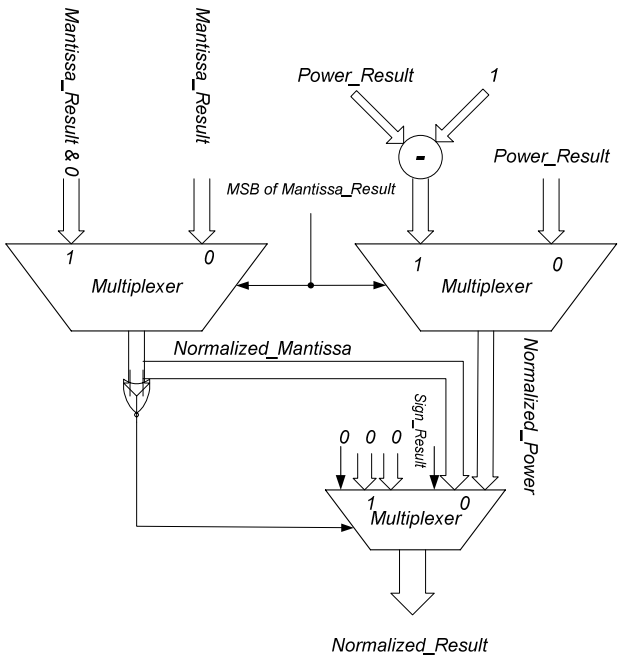


Figure 4.24 The structure of the normalized stage in the proposed multiplier

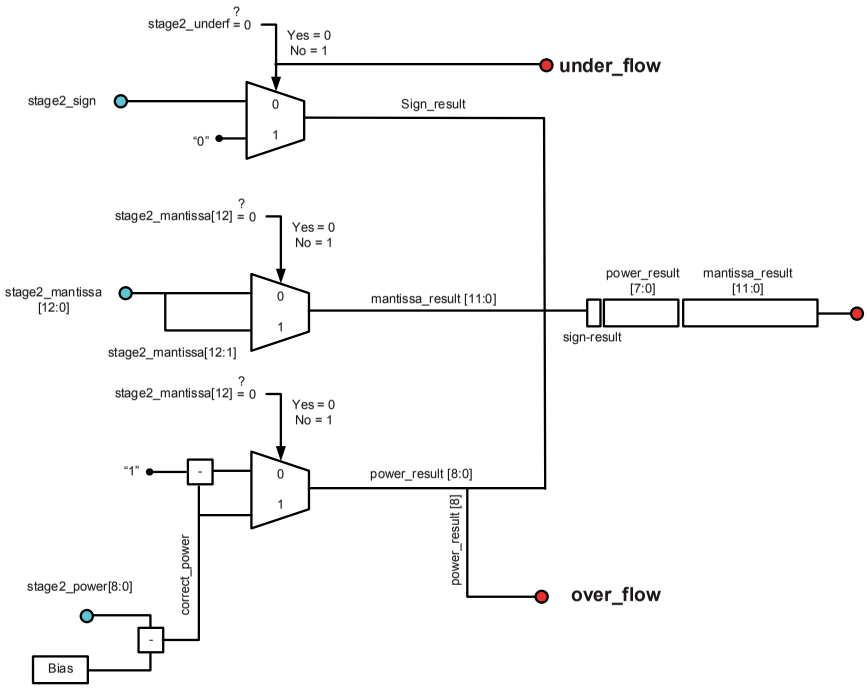


Figure 4.25 The architecture of the normalized stage in the proposed multiplier

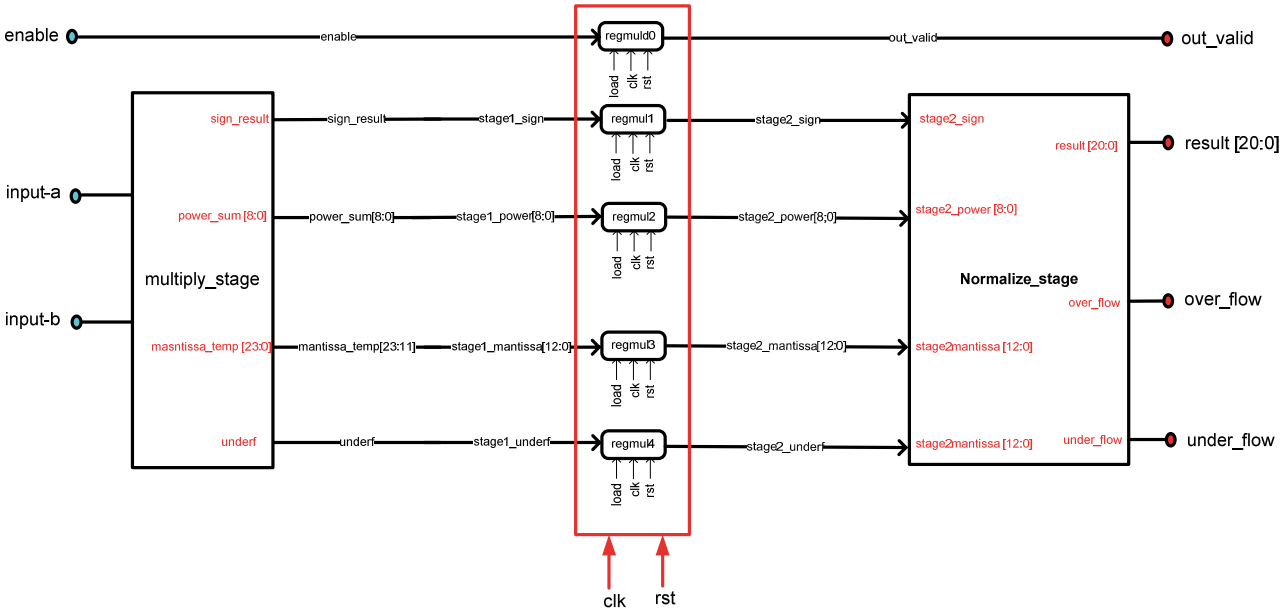


Figure 4.26 The intermediate architecture of the proposed multiplier

4.2.5 Controller Architecture

The smart controller unit significantly affects the efficiency of the 1024 radix-2 FPP-FFT processor. Furthermore, small die area can be achieved by designing high performance controller for the FFT processor. In this architecture the controller needs is designed with the pipeline capability. The global control unit provides the control signals to the different parts of the FFT processor. Additionally, several paths are switched between the data input and data output in architecture design and the data path is controlled. Figure 4.27 shows the block diagram of controller and its connections with others FFT unit.

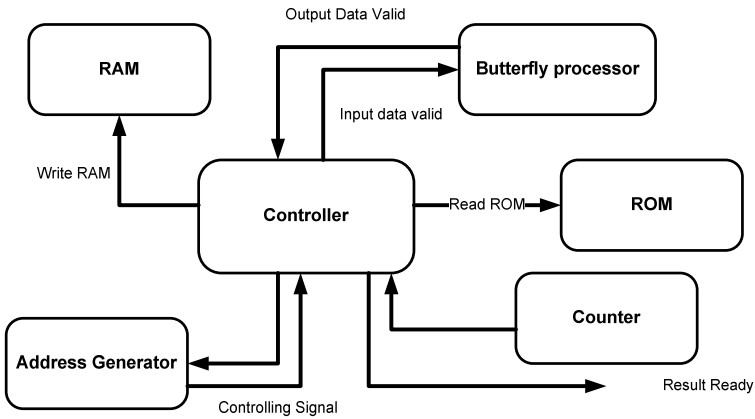


Figure 4.27 Block diagram of controller unit

To calculate 1024 point radix-2 FFT processor, it is necessary to have $log_2 N$ stages which are 10 stages for 1024 point data. Furthermore, each stage calculates $\frac{N}{2}$ butterfly that is 512 butterfly calculations in this design (Figure 3.5). Hence there are two counter in corporation with the controller to count the stage number of the processor and the number of butterfly calculation. The proposed intelligent controller

with the processor architecture of only one butterfly to reduce power consumption and area, combine the counter and address generator to produce the required address and locate it in relevant position for further radix-2 calculation. Figure 4.28 shows the smart controller state machine which controls the proposed radix-2 FFT processor.

There are several control signal pins in the smart controller to clarify the presence of correct output after finishing the current cycle of FFT calculation. The control signal pin transfers the information through the RAM, ROM, butterfly and address generator (Figure 4.27). The proposed design controller is operating using state machine. The controller unit was detaied into sub-blocks including the sequential and combination units. Figures 4.29 and 4.30 illustrate the sequential and combination unit separately. Sequential unit is responsible of updating the state of the processor whereas the combinational unit detailed the states individually. The state machine controller waits for processor core to complete FFT calculations to write data points to the memory.

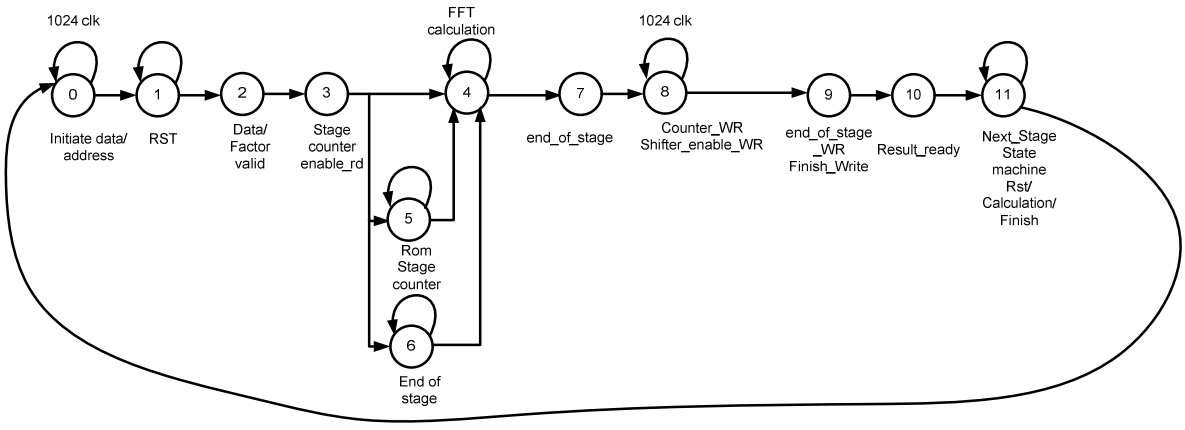


Figure 4.28 State machine block diagram for controller unit

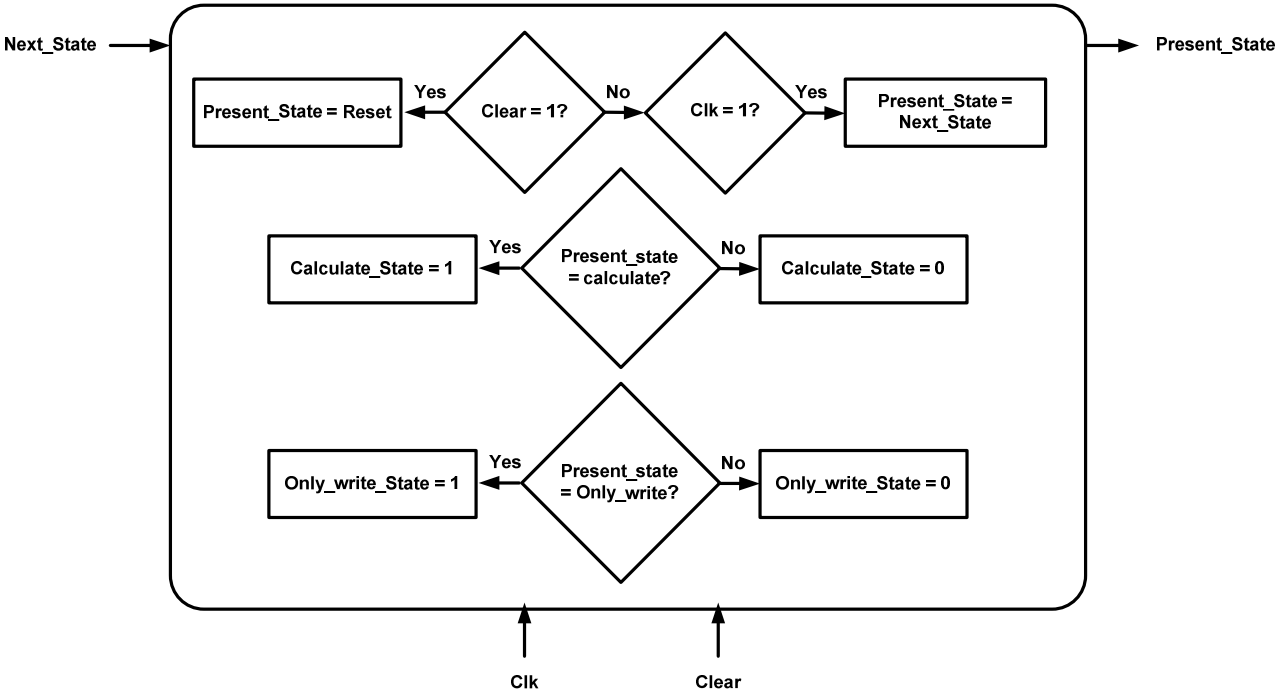


Figure 4.29 Sequential algorithm of controller unit

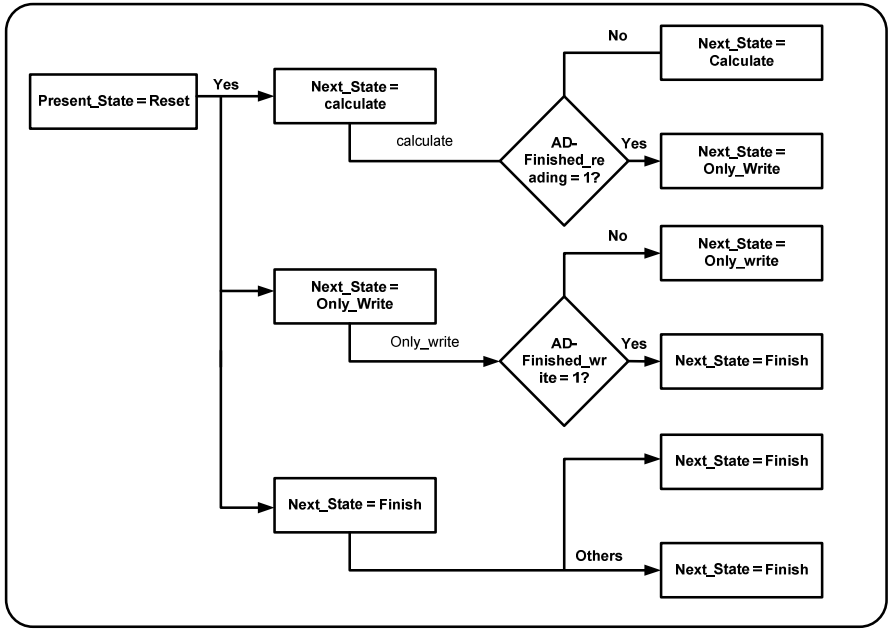


Figure 4.30 Combination algorithm of controller unit

Based on Figure 4.29 and 4.30, the *reset* state is rich every time the reset input is asserted. The following *calculate* state is reached after the *reset* input signal is removed. At the calculate state, the following actions can be requested: initialization of memory, result reading of memory or calculation of a new FFT. The memory initialization is done by putting the data and address into an appropriate bus and asserting the proper input read signal.

The input data and address are latched into a processor and the data is stored into an internal memory at the bit reversal address. The results can be read through the request for the read data procedure. This is done by putting the address to be read and asserting the read input signal. The requested data is read from data bus. The FFT process is calculated by entering to the *calculate* state. The controller changes the state to the finish state when the processor finishes the calculation to indicate the processing is over.

4.2.6 Address Generator

The address generator has an important role in the radix-2 FFT computation, since it delivers for each computation stage, the addresses of the input/output data in an appropriate way. The main objective of the read/write address generator, which is treated as part of the I/O system, is to provide a block of memory addresses in or from which the introduced butterfly's input data or the processed butterfly's output data is collected from or stored into the specific provided memory address locations. This unit grants memory address to the processor, so that it can access the right positions. To explore the symmetry of the time-decimation process, the calculation flow requires that the data access do not occur in a consecutive way. Therefore, the actual address depends on the current stage of computation which implies that many counters are needed. Address generator architecture consists of three sub-blocks:

- ROM address generator
- Read address generator
- Write address generator

The ROM address generator produces the reading address for W^k in the ROM module. The reading address represents the address of the twiddle factor which must be taken to feed the butterfly structure. This address generator is designed to select the specific twiddle factor for the butterfly calculations.

Meanwhile, the Write address generator is designed to save the result of the butterfly calculation in the proper location in the complex RAM. The proposed smart address generator is designed to provide the correct result for the next stage of the butterfly in 1024-point radix-2 FFT calculations. The architecture of the Read address generator is similar to the Write address generator. The butterfly will save the data result after reading from the certain address and input it to the butterfly, in the previous address line. The *reading_RAM_select* control signal ensures the correct location of data in the complex RAM. Figure 4.31 illustrates the architecture of the address generator in detail.

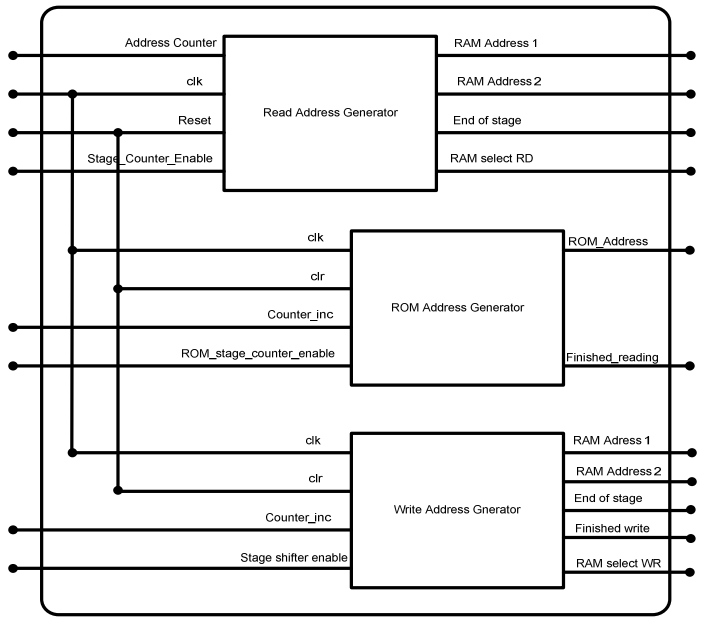


Figure 4.31 Internal structure of address generator unit

4.2.7 Memory Modules

The memory modules are used for the storing input and output results with 1024 complex long-words of 32-bits registers. The implemented architecture for the memory is shown in Figure 4.32. The capacity of the memory is 1024-point data for real and imaginary data. In our implementation, we only use single shared RAM architecture and this implemented as a single-chip FFT processor. The proposed design makes the radix-2 FFT architecture entirely independent of the type of FPGA board since it has on board memory system. Furthermore, each complex RAM has the capability of saving real and imaginary input data separately. Figure 4.33 shows the detailed RAM unit architecture as the component of the wide RAM modules.

As illustrated in the Figure 4.33, the memory module is programmed with a dual-in-line header to provide the appropriate location for storing input and output result in each stage consequently. It is composed of two delay memories and multiplexer which allows straight through or crossed input-output connection as required in the pipeline algorithm. The memory unit also contains the controller trig. The controller which is connected directly to the memory modules takes the responsibilities of transferring data through the memory and arithmetic blocks ensuring that no data conflict occurs within the complete process of FFT calculations. This is another advantage of proposed smart memory modules, by which data can be read and write in the memory simultaneously without sending bubble data in the FFT processor.

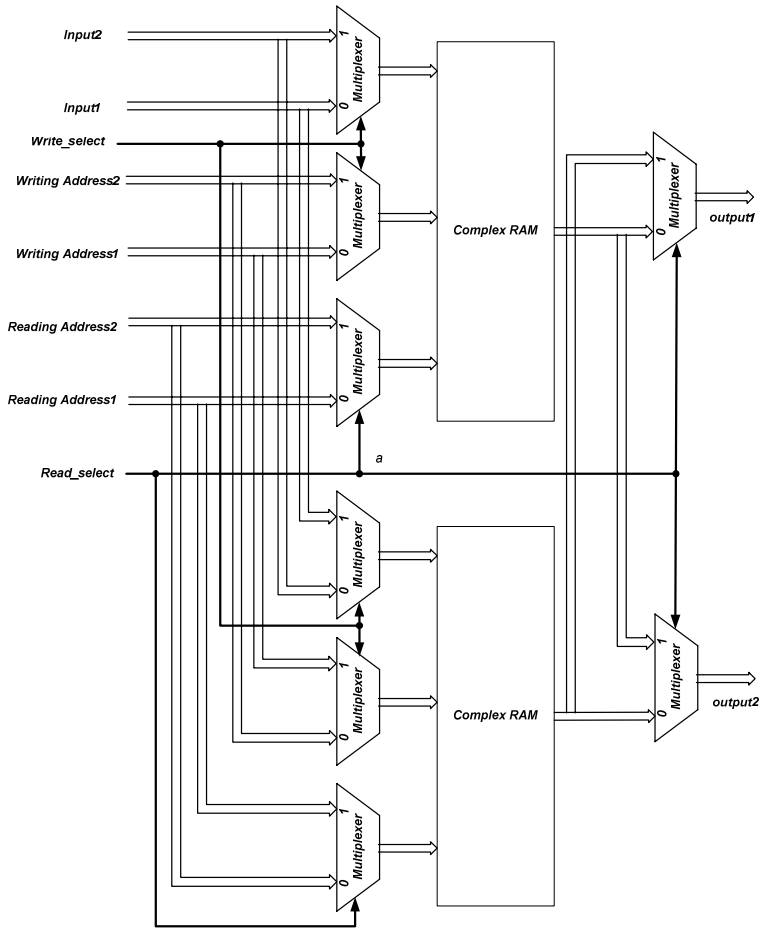


Figure 4.32 Internal structure of RAM unit

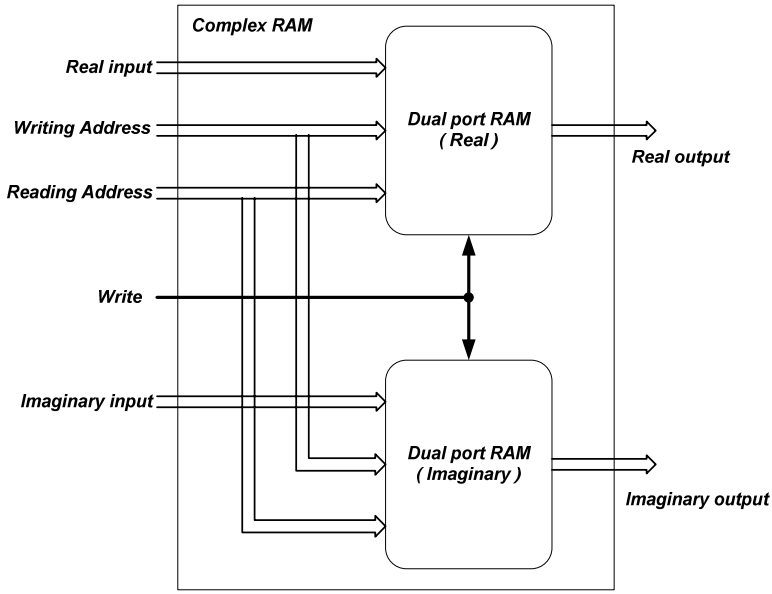


Figure 4.33 Internal structure of complex RAM unit

Beside the RAM modules, there is an on-chip ROM memory, which stores *sine* and *cosine* coefficients. Since in trigonometric circle, amplitude of *sine* and *cosine* in each quarter circle is the same and they are different only in sign, hence the intelligent designed controller change the sign of twiddle factors to the relevant coefficients in order to decreased system complexity. By storing only half of the coefficients from 0 to *N*/2, the hardware complexity of the proposed FFT processor will decrease as well as power consumption and area. Therefore only *N/2* position memory is required. The ROM used in proposed 1024-point FPP-FFT architecture applies look-up table architecture. The content of the ROM block is W^k coefficient. Each W^k coefficient is a complex number and it has two parts comprising of real and imaginary side. Hence the ROM block has two individual outputs to split complex coefficient into two parts. One output has been allocated for real and the other for the imaginary part.

4.3 ADVANTAGES OF THE PROPOSED PROCESSOR

In this section, the advantages of the architecture of the 1024-point rdix-2 FPP-FFT processor will be discussed. The technical advantages will be analyzed in the next chapter. The design architecture of the proposed 1024 point radix-2 FPP-FFT processor made use the standard components in the library where the result was optimised accordingly. However in order to improve the processor, further speed and power optimization by using the modified architecture of the processor were implemented. The new architecture is introduced by the following methods:

- Speed and power improvement by design and implementation of novel fast floating-point adder and multiplier.
- Speed and power improvement by combining the parallel pipeline architecture in the proposed radix-2 butterfly design.
- Obtain high resolution by applying the floating-point architecture with the competitive factor such as speed, power consumption and die area for FFT processor in compare with fixed-point FFT processor.
- Design complex dual RAM for single-chip FFT to make the design relaxed depending on the different types of FPGA board.
- Storing the input data and output data in a single complex RAM to reduce the hardware complexity for the conservation of the power consumption.
- Storing only half the size of the twiddle factor ($N/2$ factor of W^k) to reduce the hardware complexity in the ROM stage.
- The implementation of the combination of each individual component with the same latency and by connecting it to the sequential stage as the pipeline architecture improved the speed significantly.
- Designing the new equation for the latency in the proposed FFT processor. The data will execute storage in the RAM exactly after $(N/2\log_2^N)+11$ clock cycles.
- Designing a smart controller to arrange the overall FFT processor within a single parallel butterfly to reduce the power consumption and die size.

4.4 SUMMARY

In this chapter, the proposed 1024 floating-point parallel pipeline radix-2 FFT processor architectures were designed and implemented. All the components of this processor were evaluated separately and the algorithm has been designed. The new algorithms were explained and performed to increase the resolution and throughput of the FFT architecture. The overall system was called single-chip processor with high accuracy. The proposed design was implemented by the utilisation of the single precision data (32-bit) where it is compatible to switch to a lower bit number. The structural advantage of the proposed system was discussed in detail. The advantages of the proposed 1024-point radix 2 FPP-FFT processor are sorted accordingly.

CHAPTER V

FUNCTIONAL VERIFICATION OF FFT SPECTRUM

This chapter starts with the conventional MATLAB simulation of 8-point radix-2 FFT which will form as reference for comparison. The simulation will be expanded to 1024-point radix-2 FFT calculations. The next chapter will be followed by a detail implementation of the 1024-point floating-point radix-2 FFT processor in HDL language. In order to see resolution improvement, implementation of FFT floating point based will be completed.

5.1 8-POINT FFT SIMULATION MODULE

An 8-point radix-2 double precision FFT processor system was modelled using the MATLAB to allow various parameters of the system to be varied and tested. To ascertain that the FFT processor functions correctly, it was simulated using complete radix-2 arithmetic for 8-point FFT structure. By using MATLAB software, a model of FFT processor including adders, multipliers and multiplexers was designed and simulated. In practice, FFT computations are performed a one-dimensional array with new values overwriting old values. The data is in bit- reversed format. Alternatively, if the input sequence is in bit-revised order, the output sequence is in normal order. The output will be screened out accordingly. Figure 5.1 illustrates the random input sample given to the structural FFT blocks as real and imaginary.

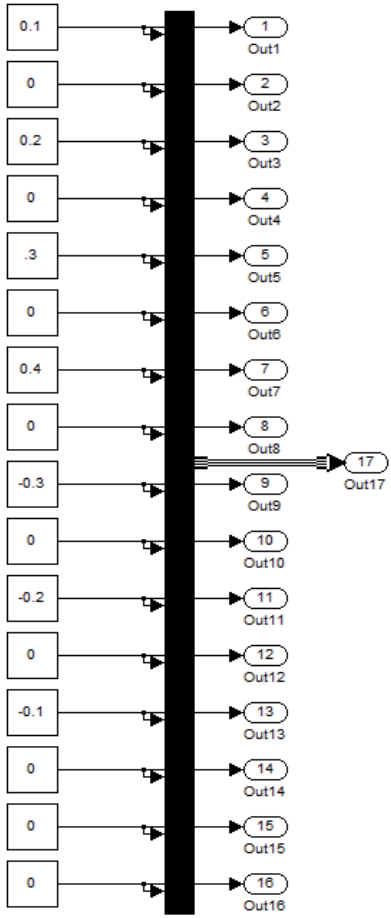


Figure 5.1 The input sampled data for 8-point FFT calculations

The data is injected into the 8-point double precision radix-2 FFT processor. There are 16 data (8-real and 8-imaginary) which are entered to the bus and will be changed as bit reverse format. Figures 5.2-5.5 show the MATLAB simulation of the 8-point radix-2 FFT processor using the butterfly structure. The internal architecture of the

processor is designed in detail (Figure 5.4). As shown in the figure 5.4, there are 3 stages for 8-point radix-2 FFT and each stage consists of 4 butterfly architectures. To calculate 8-point FFT, 4 twiddle factors are used to complete the calculation. Each individual butterfly unit by taking the relevant twiddle factor and two complexes input calculates the radix-2 FFT equation respectively. The result will transfer to the next stage for further FFT calculation.

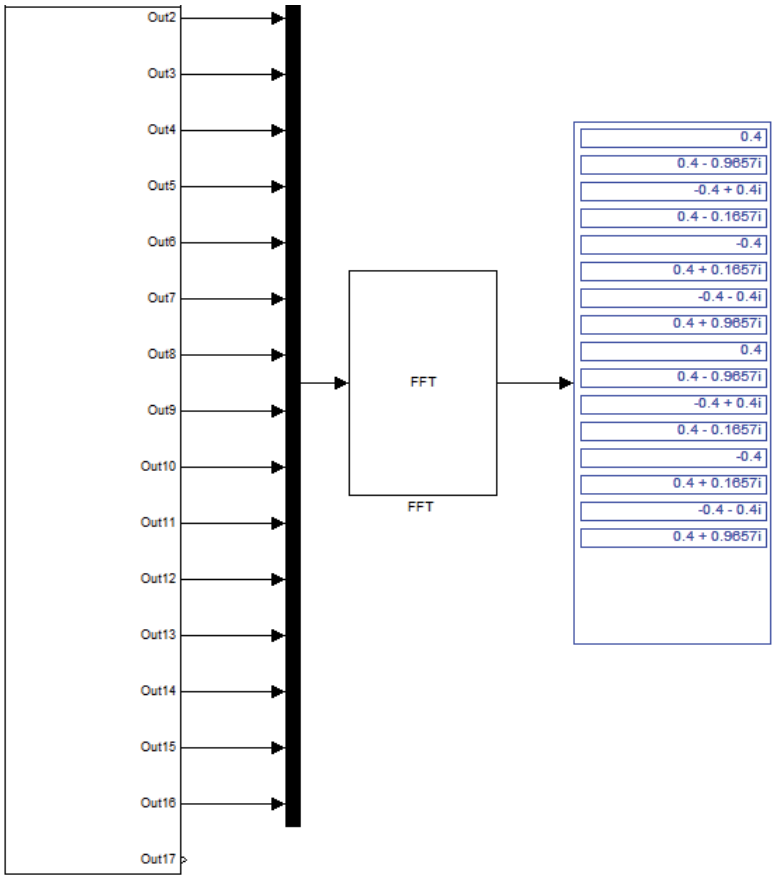


Figure 5.2 Simulation of 8-point FFT processor (MATLAB Toolbox)

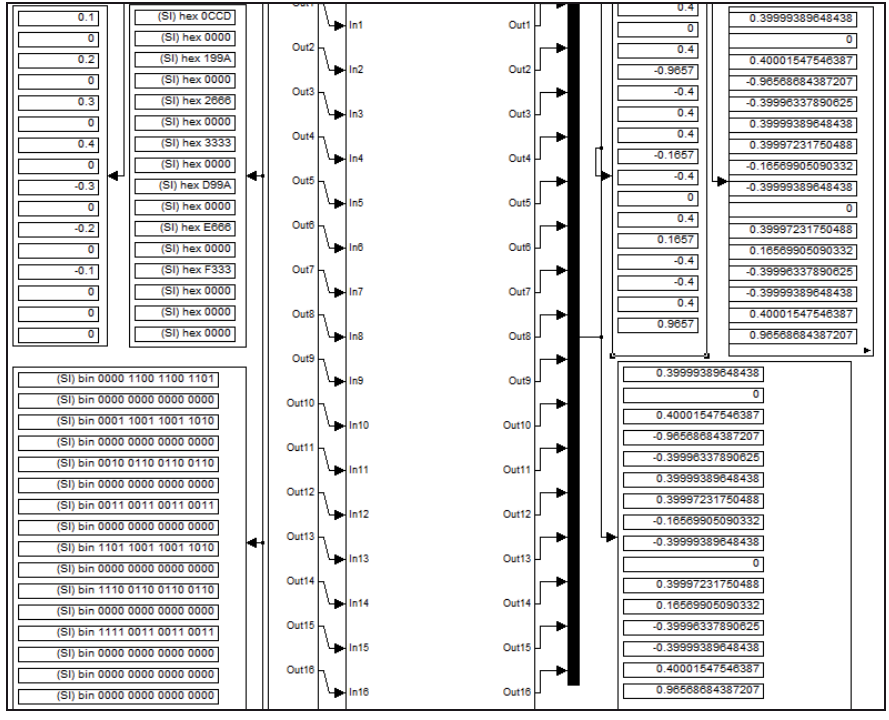


Figure 5.3 MATLAB Simulation of 8-point radix-2 FFT processor

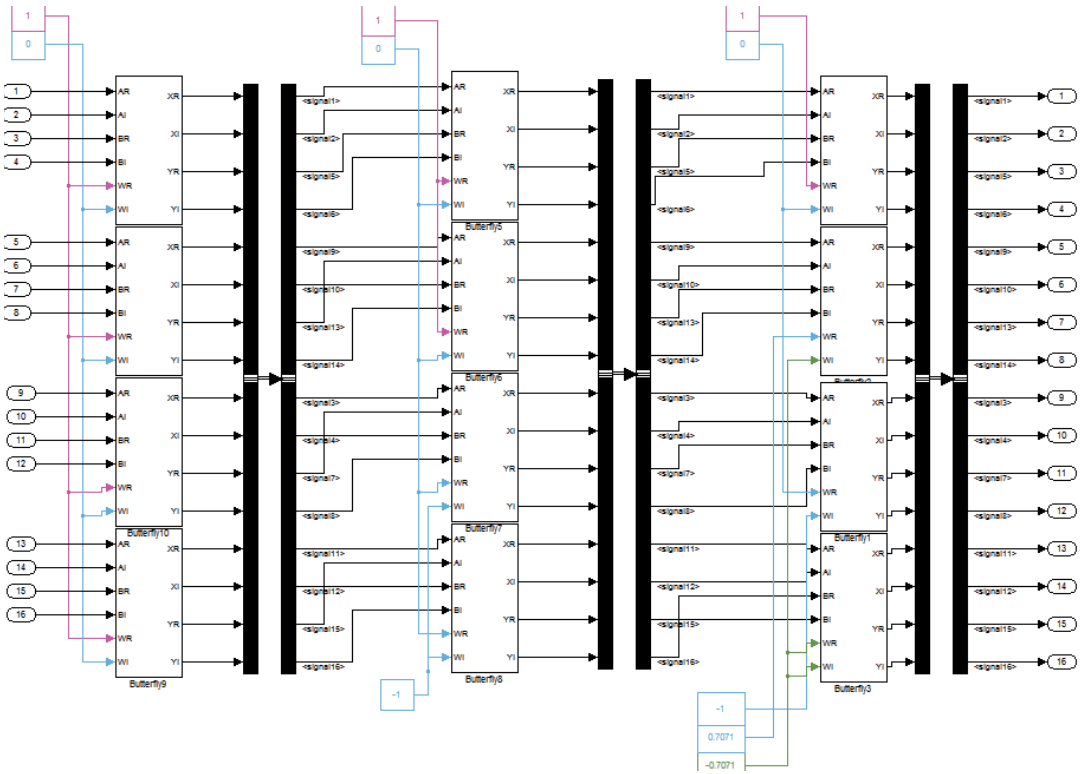


Figure 5.4 Internal structural of 8-point radix-2 FFT processor

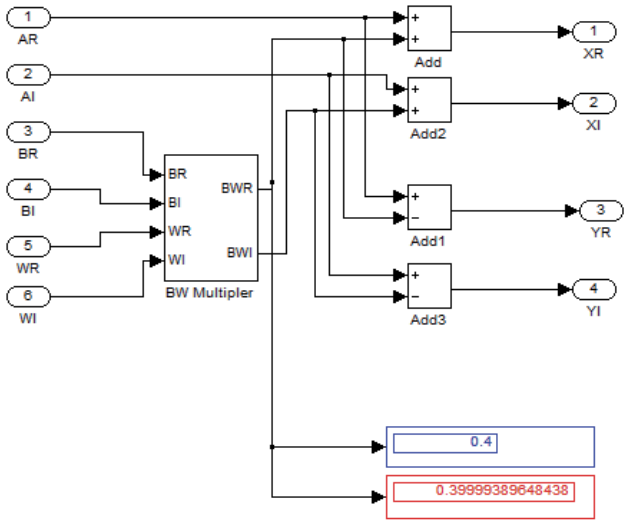


Figure 5.5 MATLAB simulation of butterfly unit in Radix-2 processor

Due to complete radix-2 butterfly calculation, the twiddle factors are injected to the FFT processor. As shown in Figure 5.6, it is required to store twiddle factor in the ROM. The next section explains how twiddle factors are calculated to store into the ROM. As illustrated in Figure 5.4, there are only four different twiddle factor values in the 8-point FFT processor. To calculate twiddle factors, $W = e^{-\frac{2\pi nj}{N}}$ for N = 1024 MATLAB software can provide the calculation.

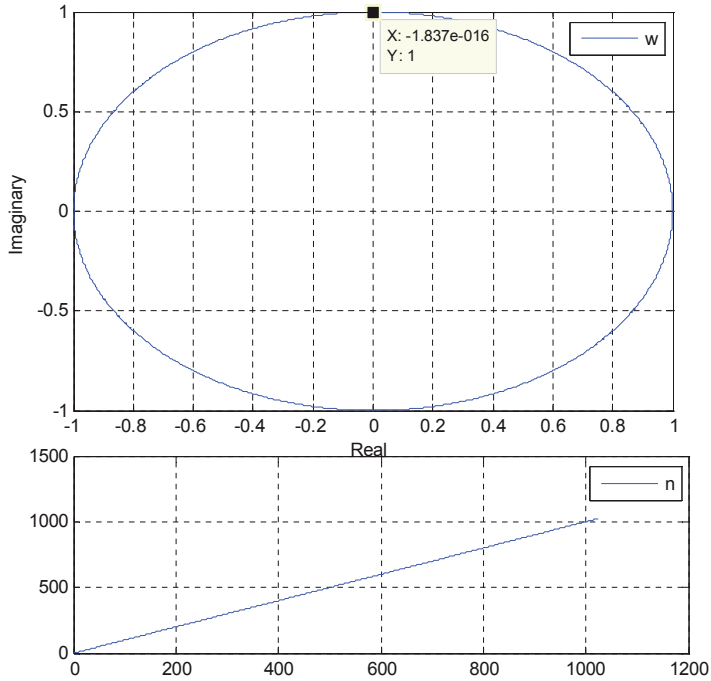


Figure 5.6 Twiddle factor when $N = 1024$

Based on the computed twiddle factors, they are loaded into the ROM which is inside the proposed radix-2 FFT architecture.

Although this representation of twiddle factor produces an additional latency at the beginning of the FFT computation, at this particular instance, it could be ignored during the analysis of latencies. After computing the twiddle factors and inputted it into the FFT simulation structure, the 1024-point FFT processor will calculate the Fourier transform and the final result will be stored in a separate file to compare it with the implementation of the Fourier transformation result.

Presented is the 8-point double precision architecture. The project has been moved onto simulation of 1024 radix-2 FFT processor. But due to the complex structure, it will not be detailed the simulation structure. It is only showed the simulation result of overall processor followed by VHDL implementation of proposed processor for 1024-point radix-2 FFT in next chapter.

5.2 1024-POINT FFT SIMULATION RESULT

The MATLB software provides the output simulation of double precision 1024-point FFT processor and the twiddle factors. To find the system resolution, the implementation results are compared with the simulation output. The random input data was tested for the 8-point FFT processor whereas the rectangular signal with amplitude of 2 coded as a single precision floating point data for testing the proposed 1024- point FPP-FFT. Table 5.1 shows the MATLAB simulation output of the conventional 1024-point FFT processor when N = 1024. Each number is represented by real and imaginary parts.

Table 5.1 Expected MATLAB simulation results for radix-2 FFT processor

MATLAB FFT (x)	
Input x(n) = 1024 samples	
Real	**Imaginary**
244	0
222.1080845	-86.46028488
163.4574705	-149.9863166
86.44938802	-175.7548947
13.94525101	-161.9378169
-35.05249811	-119.5011318
-51.77981999	-67.19575906
-39.7275488	-23.97775135
-11.78182712	-2.044340937
15.98247349	-3.230116609
31.04140023	-19.92463871
28.87413771	-39.69060396
13.51218305	-51.26607916
-5.614881197	-49.25132592
-18.75209068	-35.60650715

Figures 5.7 and 5.8 show the amplitude and the phase of the rectangular sampled signal respectively when the 1024-point FFT processor is applied. As shown in the figures, the *sinc* signal for the amplitude and linear phase are created to prove the linearity of the processor and its stability.

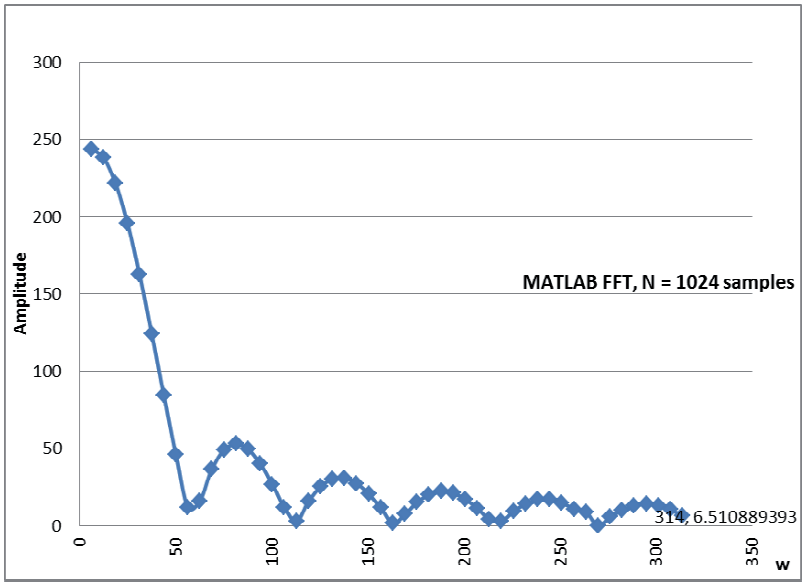


Figure 5.7 MATLAB simulation of amplitude frequency response of the rectangular input signal

The representation of a sequence by the transformation of the FFT processor is not restricted to the given–sample response of a system but can be applied to any sequence provided that the series converges.

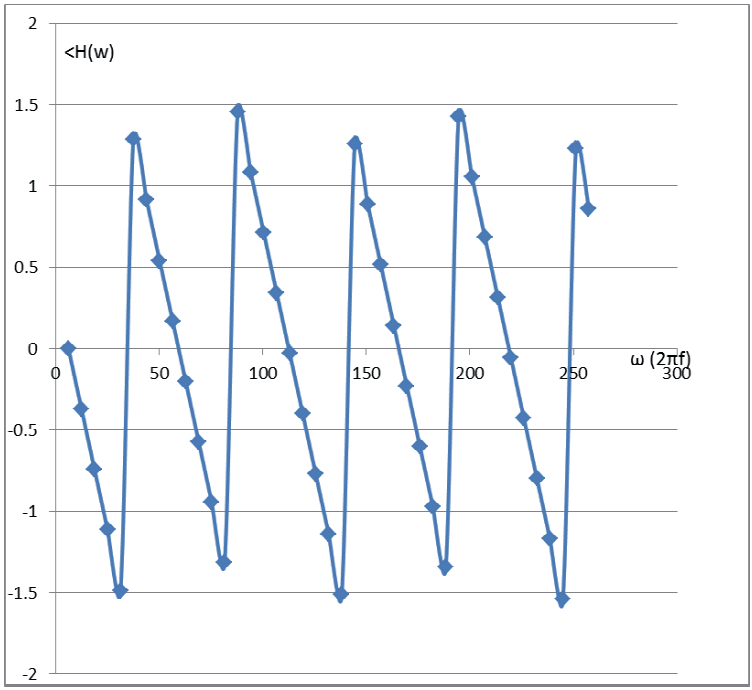


Figure 5.8 MATLAB simulation of phase frequency response of the rectangular input signal

To control the internal calculation of floating-point radix-2 FFT processor, the MATLAB simulation module was designed (Figure 5.9). However the final result obtained by the proposed FPP-FFT processor needs to be compared with the MATLAB simulation as de facto and fixed-point radix-2 FFT processor implementation result.

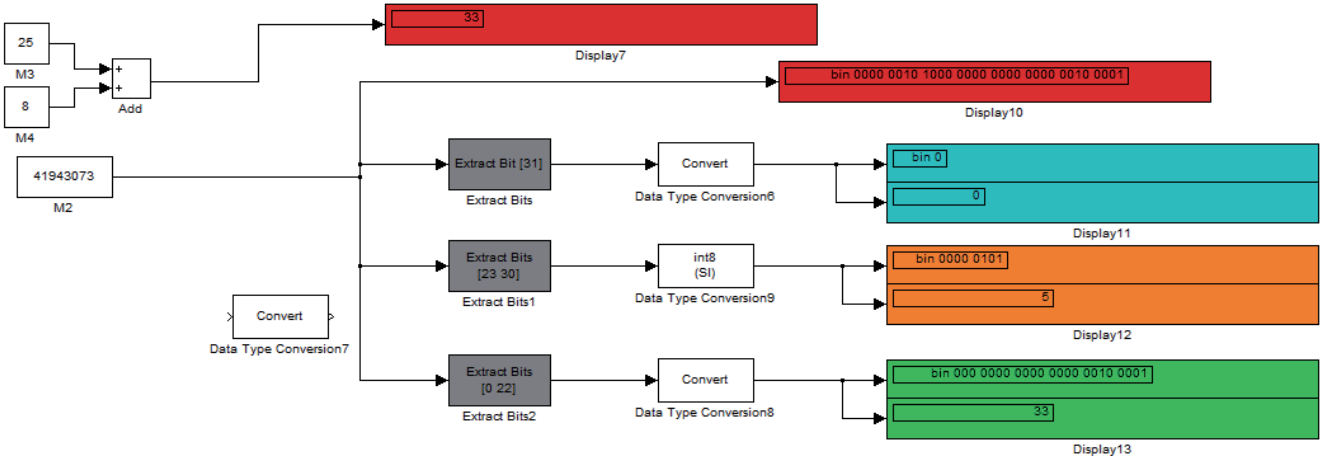


Figure 5.9 MATLAB simulation of floating point data structure

5.3 PROPOSED FLOATING POINT FFT APPLICATION

As stated in Chapter II, we can apply the processor for harmonic measurement or power spectrum measurement due to its high performance.

a) Harmonic Measurement

Figure 5.10 shows the noisy input signal in time domain while Figure 5.11 illustrates the frequency spectrum of input signal when floating-point FFT applied versus the frequency response when fixed-point FFT utilized (Figure 5.12). As shown in the Figures 5.11 and 5.12 noise floor appears in high resolution floating-point FFT whereas this noise appears as nulls in fixed-point FFT processor.

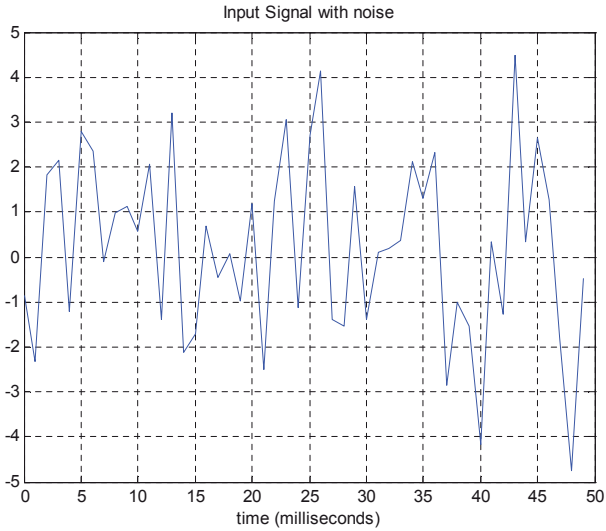


Figure 5.10 The noisy input signal in time domain

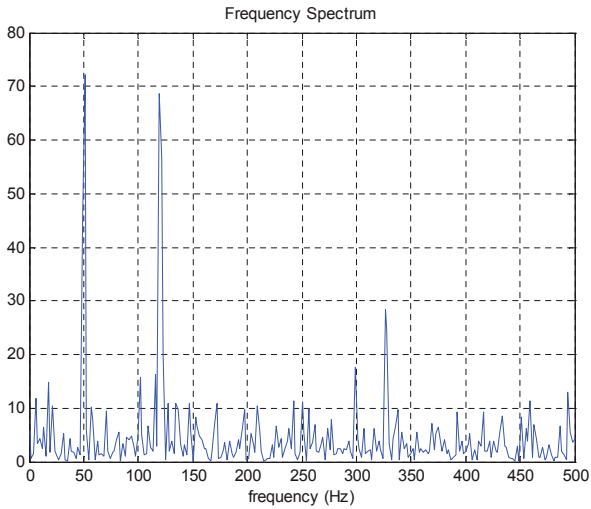


Figure 5.11 The harmonic measurement of noisy signal with floating-point FFT

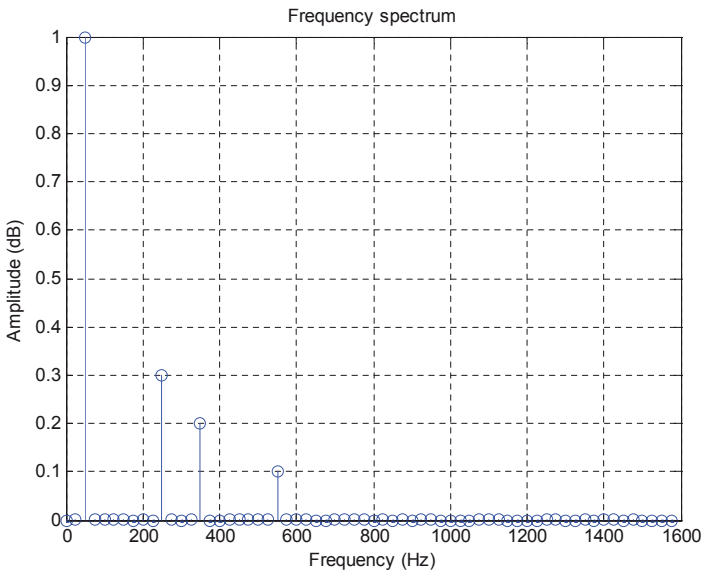


Figure 5.12 The harmonic measurement of noisy signal with fixed-point FFT

b) Power Spectrum

At the same time, proposed 1024-point radix-2 FFT processor can be applied for power spectrum measurement. Since the energy of the signal are expressed as:

$$E = \sum_{n=0}^{N-1}|x(n)|^2 = \frac{1}{N}\sum_{k=0}^{N-1}|X(k)|^2 \quad (5.1)$$

where $X(k)$ is in frequency domain and $x(n)$ in time domain.

It is important to note that $|X(k)|^2$ is defined either as the energy spectrum when the signal $x(n)$ is of finite duration or the power spectrum if the signal is a periodic sequence (Sen & Woon 2005).

$$P(k) = \frac{1}{N}|X(k)|^2 \quad (5.2)$$

The power spectrum is created by floating-point radix-2 FFT calculation to analyse the frequency components of signals. The spectrum is adequate for spectral analysis on the condition that the signal does not change its characteristics. For time-varying signals, single block fixed-point FFT-based power–spectrum estimation is not suitable for extracting or displaying time events since it is not accurate. Hence high resolution floating-point FFT power spectrum block rises up to function. Figure 5.13 shows the power spectrum of the signal $x(n)$ when the high resolution radix-2 FFT processor is applied.

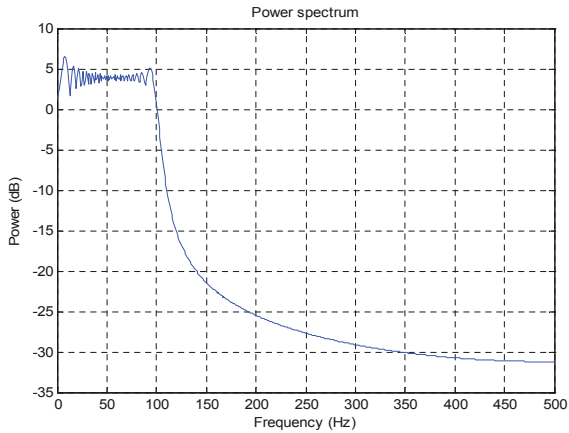


Figure 5.13 The MATLAB simulation of power spectrum using floating-point FFT

5.4 SUMMARY

In this chapter, the simulation result of the 8-point radix-2 FFT processor which utilises MATLAB tools is presented, and then followed by the extension of 1024-point FFT MATLAB simulation. The simulated FFT processor was examined by rectangular signal as input signal. The frequency response (amplitude and phase) of the sampled rectangular signal by using the FFT processor are given to prove the stability of the processor due to the phase linearity. To stage realisation of the FFT processor the individual block diagram was designed and simulated and a particular model to perform floating-point data testing was considered. In order to find the system resolution, the result of the FFT processor was stored. Later on, the actual output data produced by the proposed FPP-FFT processor architecture will be compared with the simulation result.

CHAPTER VI

IMPLEMENTATION RESULTS

Chapter V detailed the proposed algorithm as 1024-point floating-point parallel pipeline radix-2 FFT processor. The hardware implementation of the algorithm will be presented in this chapter so the FPGA and ASIC implementation of FFT can be achieved. Figures 6.1 and 6.2 illustrate the framework of proposed FFT implementation in detail.

As stated in the figures, first the RTL behavior description of design is programmed and verified to proceed for FPGA implementation. The procedure is continued by attaching the library cell and constraint file for ASIC implementation. Later, the proposed designed is transferred to the gate level synthesis and elaboration to complete post-simulation stage. The software tools such as MODELSIM and NC-launch in Synopsys post-simulate the behavioral and netlist of the processor respectively.

After post-simulation in front end VLSI implementation, we moved forward to the back end implementation by 0.18 μm SILTERRA technology and 0.35 MIMOS technology library. The netlist design with attaching the SDC constraint and IO pad is transferred to floor planning and place and route process to optimize the design and achieve the specification. The implementation detail is shown in Figures 6.1 and 6.2.

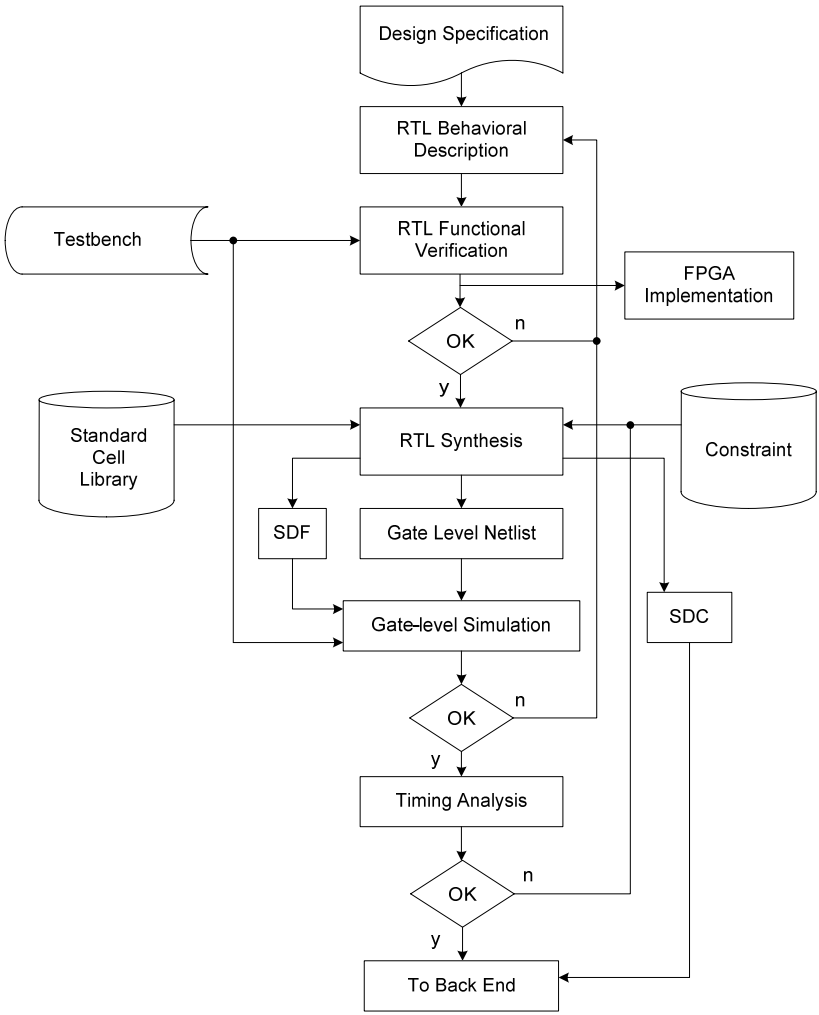


Figure 6.1 VLSI Front end design flow of the project

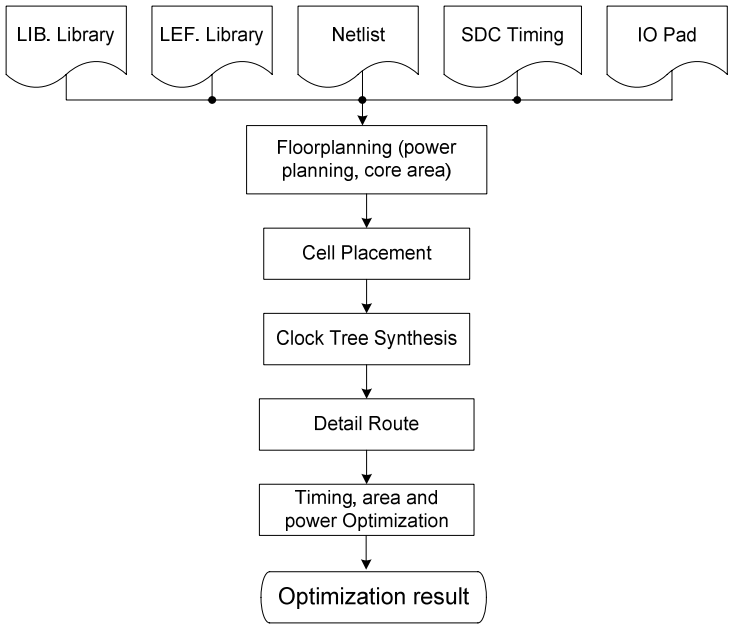


Figure 6.2 VLSI back end design flow of the project

6.1 HARDWARE IMPLEMENTATION OF 1024 POINT FPP-FFT

In order to verify the functionality of the 1024-point FPP-FFT processor, the VHDL code for the overall system was developed and downloaded to the FPGA board. To realize the hardware for algorithm, Xilinx and MODELSIM- EDA tools were used to synthesize and to simulate the design. In summary, the FPGA prototype processes are listed in (i) and (ii):

1) VHDL coding for the processor and simulating the design by MODELSIM simulator.

MODELSIM results are shown after implementing individual units. The MODELSIM simulation results illustrate the output and the intermediate input signal of the processor top/sub-modules in FPGA board.

There are six sub-modules in the design which are bit-reverse, ROM, RAM, controller, butterfly and the address generator. In the next section the implementation of the proposed architecture will be discussed.

6.1.1 Top-Module of the Radix- 2 FPP- FFT Processor

The implementation of the proposed processor was done for a range of design parameters. The top-design of the processor with the I/O pins is shown in Figure 6.3 whilst Figure 6.4 illustrates the chip design of the FPP-FFT.

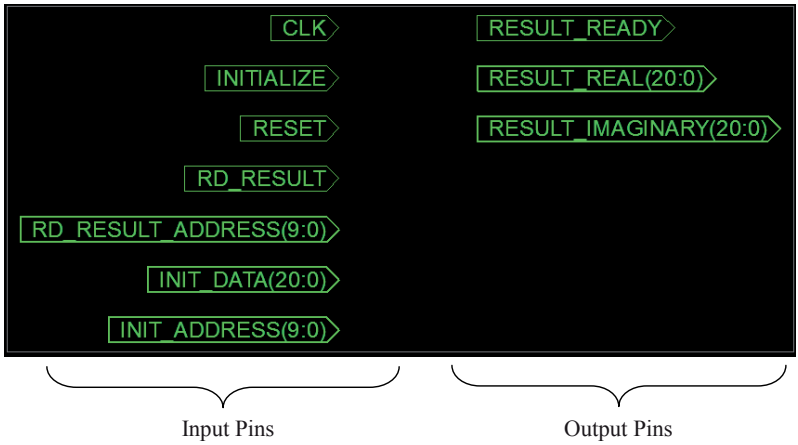


Figure 6.3 Input/Output pins of the FPP-FFT processor

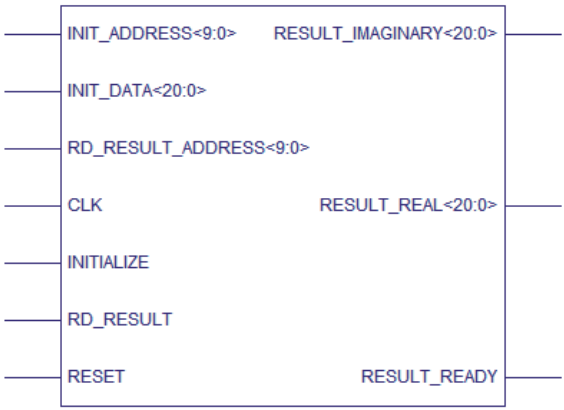


Figure 6.4 FPP-FTT processor top level

When the FFT has completed the transformation of the input block, the core asserts the *RD_result* and outputs the complex FFT transform of real and imaginary results. The *RD_result* signal indicates the first output sample. The input and output of the FFT are both in natural order that is 1...*N*. The output data is in single precision format. The single precision format could be expanded to the double precision format; however, this precision provided enough information to conclude about their impact over the latency, throughput, and resources consumed and maximum frequency.

Figures 6.5 – 6.7 show the overall implementation of the proposed FFT processor which consists of the ROM and RAM unit, the butterfly Radix-II, the controller and the address generator.

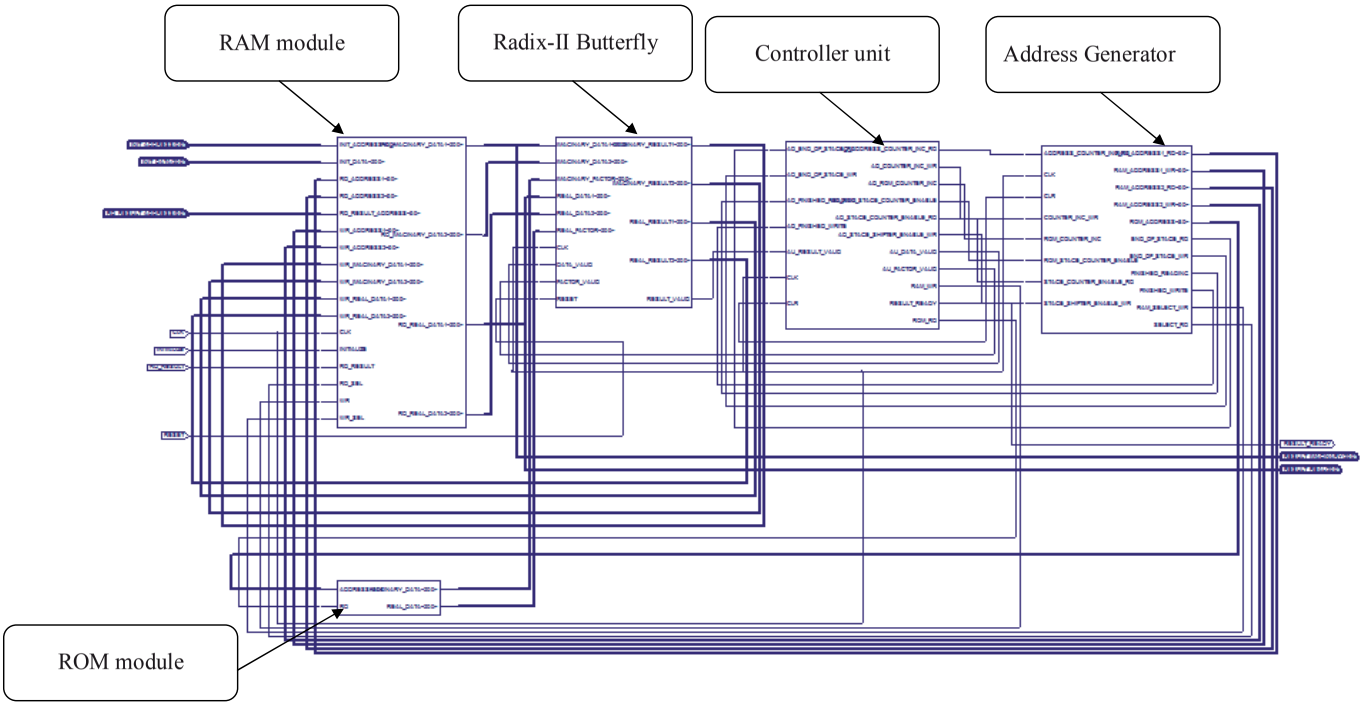


Figure 6.5 Proposed 1024-point pipelined floating point FPP-FTT processor

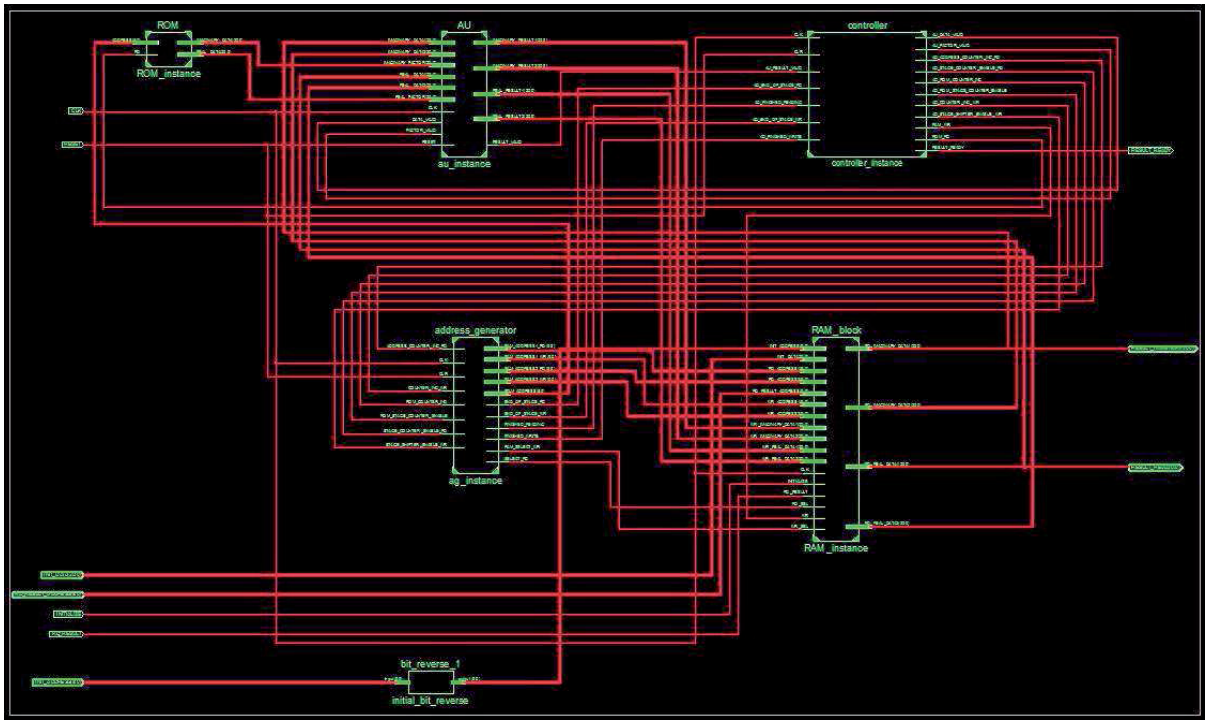


Figure 6.6 Behavioral layout of FPP-FTT processor

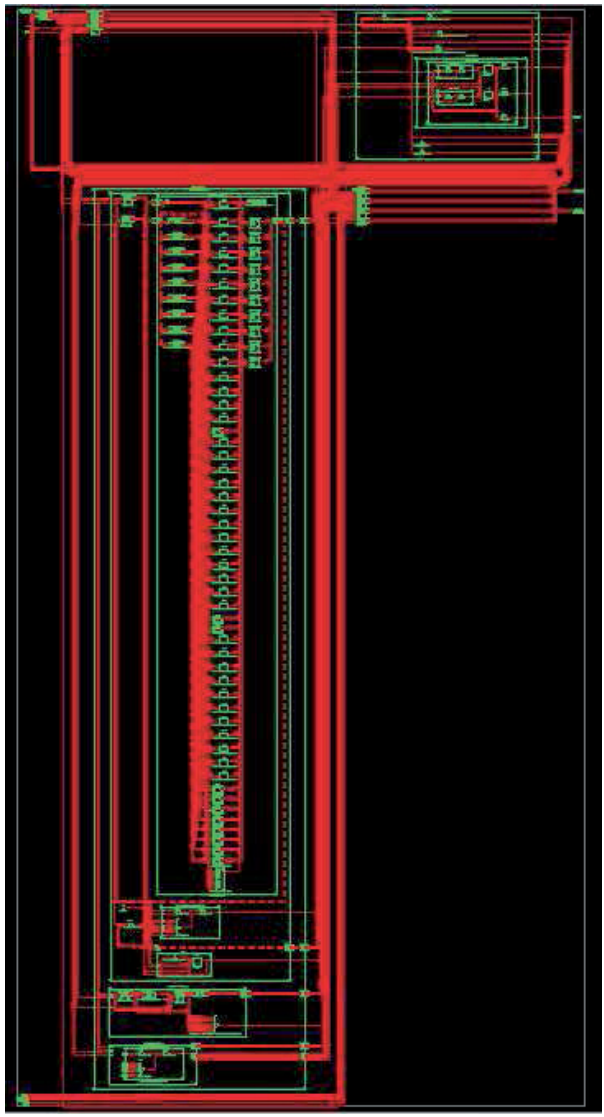


Figure 6.7 Internal behavioral layout of FPP-FTT processor

Table 6.1 shows the summary of Xilinx ISE synthesis for the overall proposed novel architecture of FPP-FFT processor.

Table 6.1 Proposed FPP-FFT Processor specification

HDL Synthesis Report		Timing Summary	
Registers Flip-Flops	1175	Minimum period (ns)	4.391
Shift Registers	43 (6%)	Maximum Frequency (MHz)	227.747
LUTs Slice	4419 (23%)	Min. input arrival time (ns)	3.788
Logic Slice	2584 (13%)	Max. output required time (ns)	6.774
RAM Cells	1835 (35%)	Total equivalent gate count	998678
IOs	88 (40%)	Total Number of Path	220310
Memory usage (MB)	254 (40%)	Total Number of Destinations	5926
Multiplexers	77		
Tri-states	98		

As shown in Table 6.1 the new architecture of 1024 point radix-2 FPP-FFT processor is able to operate with the maximum clock frequency of 227.7 MHz.

6.1.2 Bit Reverse Implementation

The overview of the bit-reverse in 7-bit module is given in Figure 6.8. Although the functionality of this block is relatively simple, it is imperative to figure out the output as Fourier transform. However the system specification is given in Table 6.2.

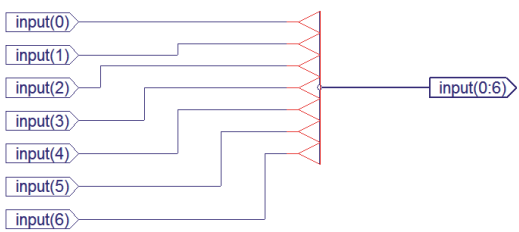


Figure 6.8 Bit-reverse implementation

The internal structure layout of the bit-reverse is shown in Figure 6.9. The interval gate in the bit-reverse structure leads bit change location from the input and injects it into the output. Hence the output of the processor is in reverse style.

Table 6.2 Bit-reverse specifications

HDL Synthesis Report		Timing Summary	
LUTs Slice	22	Max. combinational path delay (ns)	4.96
Logic Slice	22	Total JTAG gate count for IO	672
IOs	14	Total Number of Path	105
Memory usage (MB)	133	Total Number of Destinations	7
Tri-states	7		

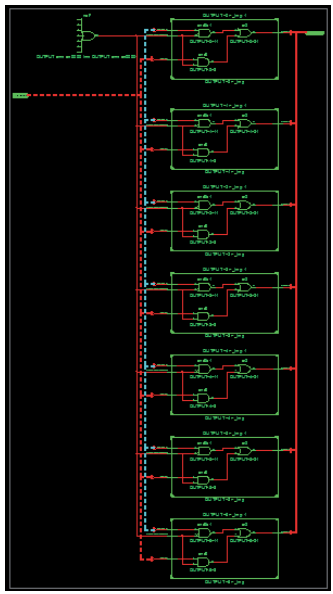


Figure 6.9 Internal structural layout of bit-reverse

To test the bit-reverse implementation, MODELSIM tools provides the interval signal as the input and output to prove the system functionality as shown in Figure 6.10.

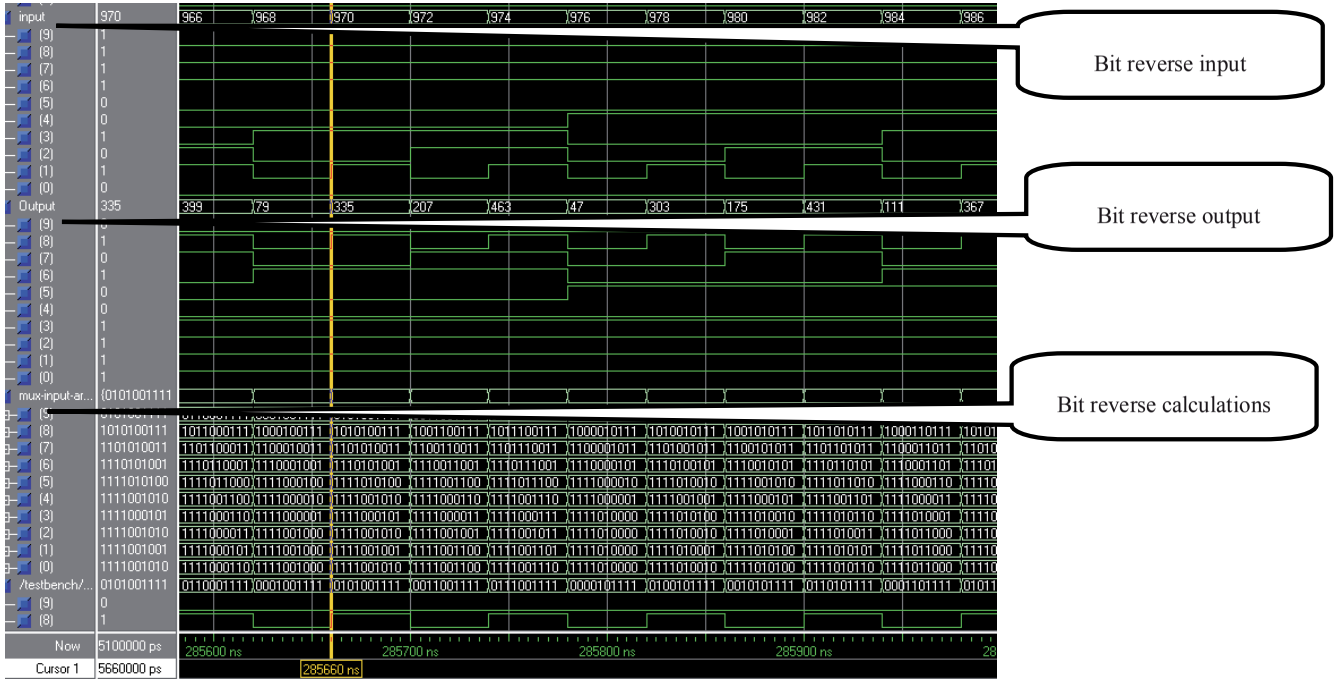


Figure 6.10 Bit-reverse input/output signal

6.1.3 Radix-2 Butterfly Implementation

As previously stated, radix-2 butterfly architecture is the heart of FFT processor. Therefore, high performance implementation of the radix-2 butterfly will directly affect the system efficiency. Figures 6.11-6.13 show the implementation of the butterfly architecture as discussed in Chapter V. The control pins such as *enable*, *reset* and *valid* open the gate for data to transfer among the stages and these are a part of radix-2 controller. The system was synthesized using Xilinx ISE synthesis tools where Table 6.3 clarifies the system specification. The layout architecture of the radix-2 butterfly is given in Figure 6.14. The novel architecture of radix-2 butterfly is achieved by the floating-point parallel pipeline structure with the cooperation of the advance arithmetic units (adder, subtractor and multiplier).

Table 6.3 Proposed butterfly specification

HDL Synthesis Report		**Timing Summary**	
Registers Flip-Flops	977	Minimum period (ns)	3.563
LUTs Slice	1714	Maximum Frequency (MHz)	280.674
Logic Slice	977	Min. input arrival time (ns)	4.221
RAM Cells	-	Max. output required time (ns)	8.055
IOs	216	Total equivalent gate count	40714
Memory usage (MB)	161	Total Number of Destinations	970
		Total Number of Path	5926

In enabling the achievement of high speed processor, the design located pipeline registers in the radix-2 butterfly processor architecture. The pipeline architecture keeps the data for one clock cycle for synchronization in order to increase the system efficiency. Figure 6.15 shows the input and output signal of complete proposed radix-2 architecture using MODELSIM simulation tools. The input in appearance of the positive clock edge is entered to the proposed FFT processor and later, the output which is the Fourier transform form is appeared in the output port.

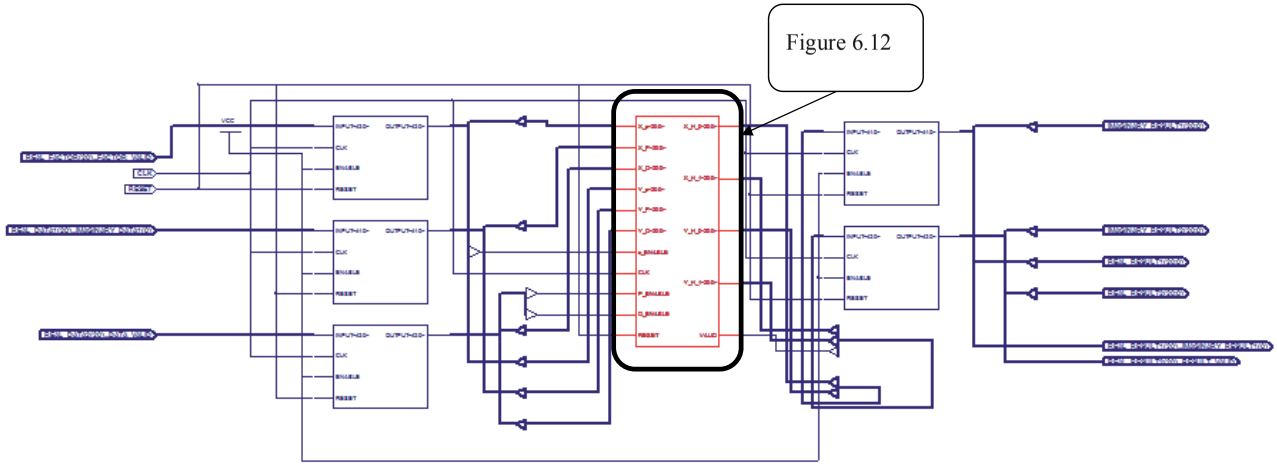


Figure 6.11 Radix-2 butterfly architecture with pipeline registers

Figure 6.11

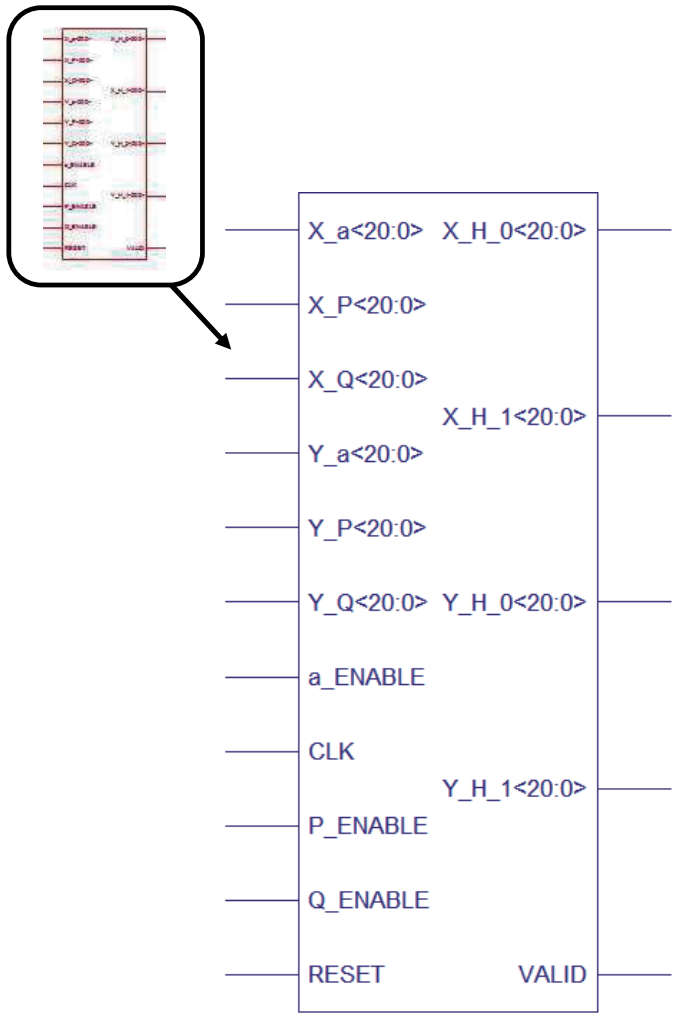


Figure 6.12 Top- module of Radix-II butterfly architecture

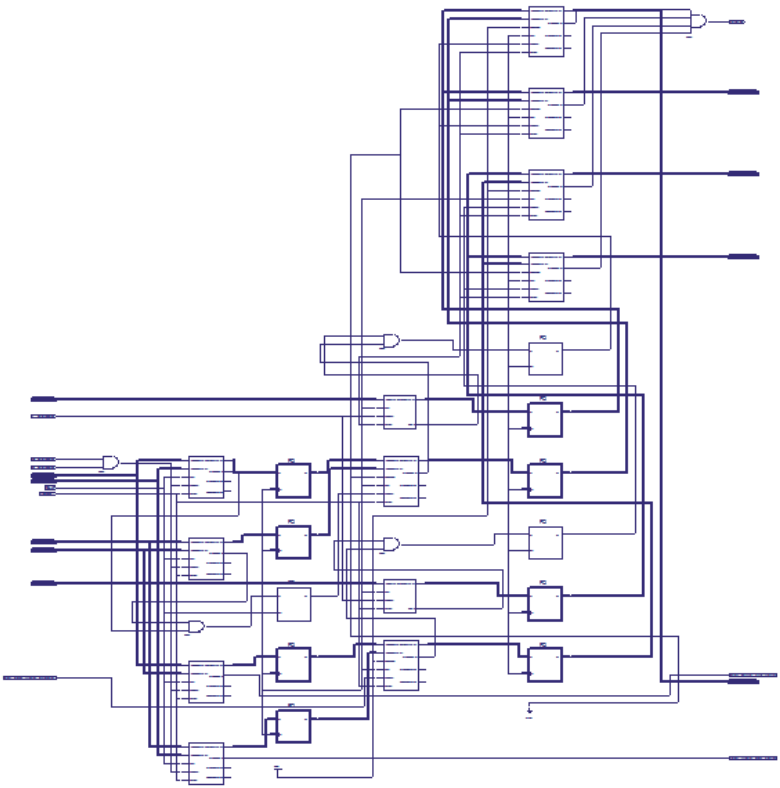


Figure 6.13 Internal butterfly architecture

Figure 6.14 Internal behavioral layout of Radix-2 butterfly

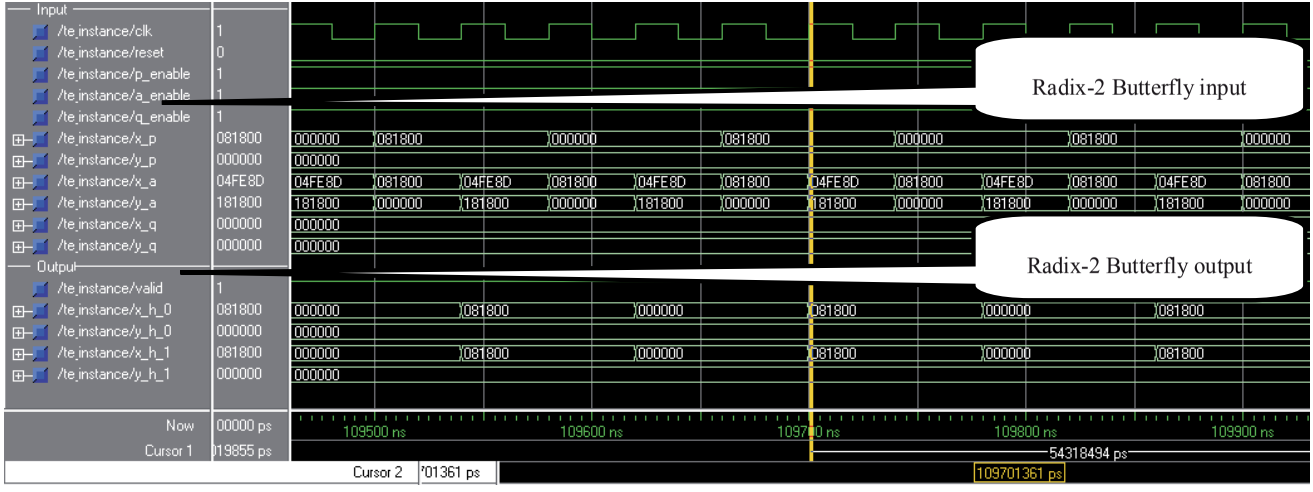


Figure 6.15 Input and output signal of Radix-2 butterfly

a) Floating Point Adder/Subtractor Implementation

As discussed in Chapter IV, the floating-point arithmetic units in radix-2 butterfly comprises of four (4) stages called comparison stage, alignment, add/sub and normalized stage.

The implementation of the pipeline floating-point adder/subtractor was modeled in VHDL code and simulated by MODELSIM software. The design was synthesized by Xilinx-ISE software and downloaded to the FPGA Virtex II. The synthesis result shows the system specification of each stage of the pipelined floating - point adder/subtractor separately. Finally the overall system synthesis results from the combination of the mentioned stages are given in Tables 6.4 – 6.8.

From the Xilinx ISE synthesize report; it was found that the minimum clock period is 3.592 ns (the Maximum Frequency is 278.428 MHz) for the floating-point adder/subtractor. Furthermore, the minimum clock periods increased sharply after add/sub stage was applied.

This issue can be explained by utilizing the FPGA adder to calculate the fixed-point arithmetic. However the speed result is high enough to cover the subject. To enhance the maximum clock frequency, high speed prefix adder (Burgess 2004) can be applied to calculate the fixed- point arithmetic for future work.

The proposed adder implementation results are shown in Figures 6.16-6.21. The estimated latency of the design for 32-bit resolution is 4 clock cycle due to its pipeline structures. The specification tables of the proposed floating-point adder/subtractor are given in Tables 6.4 – 6.8.

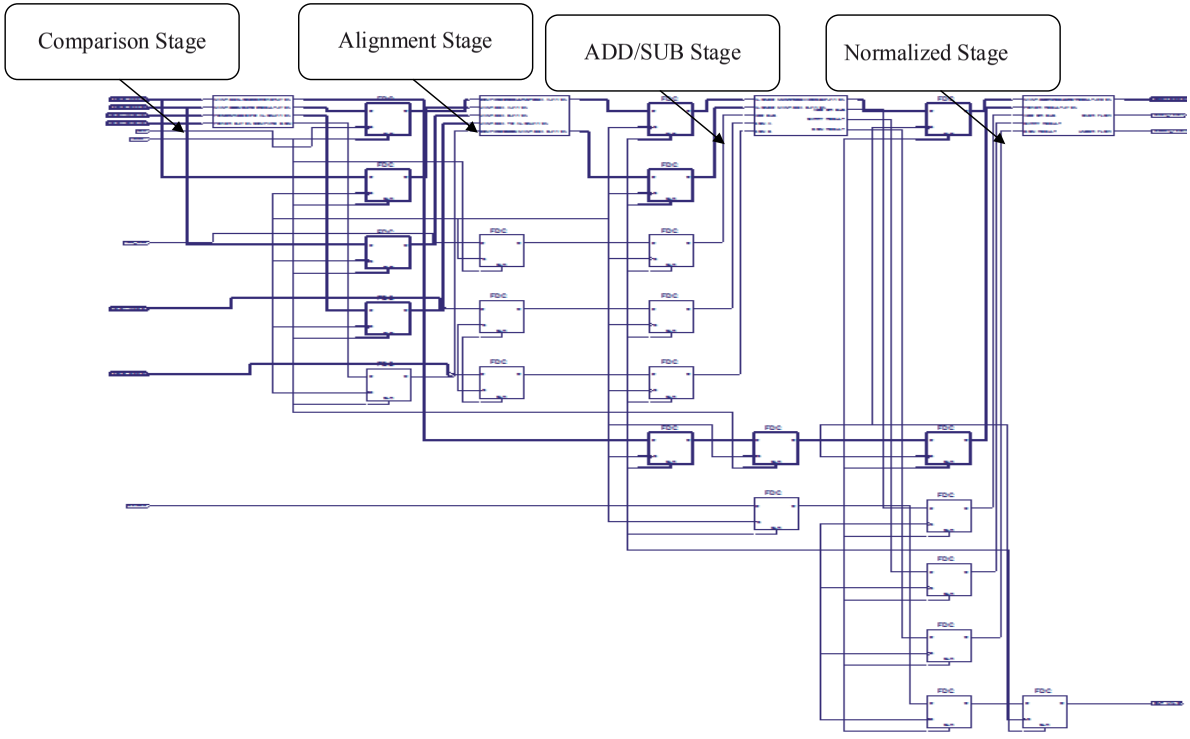


Figure 6.16 Fast floating-point adder/subtractor internal architecture

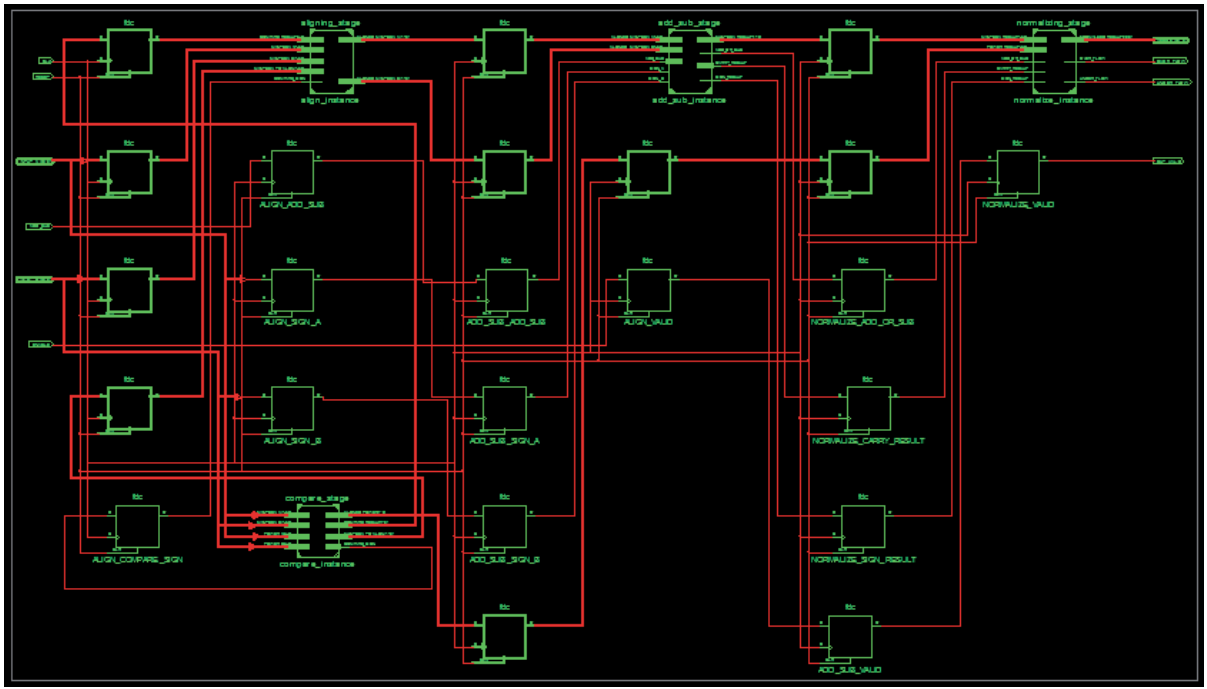


Figure 6.17 Fast floating-point adder/subtractor layout

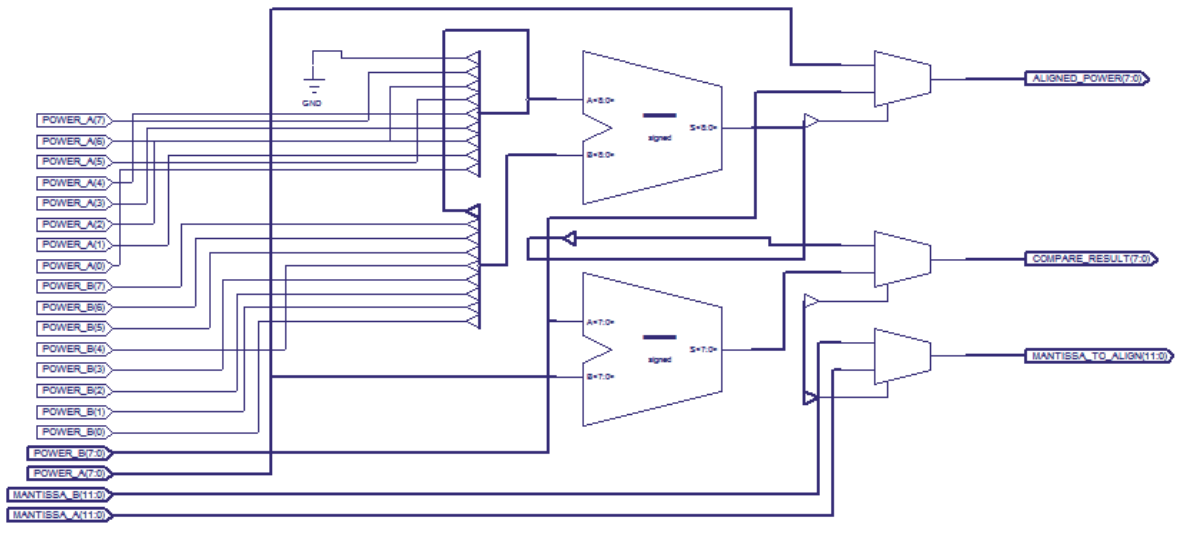


Figure 6.18 Comparison stage internal architecture

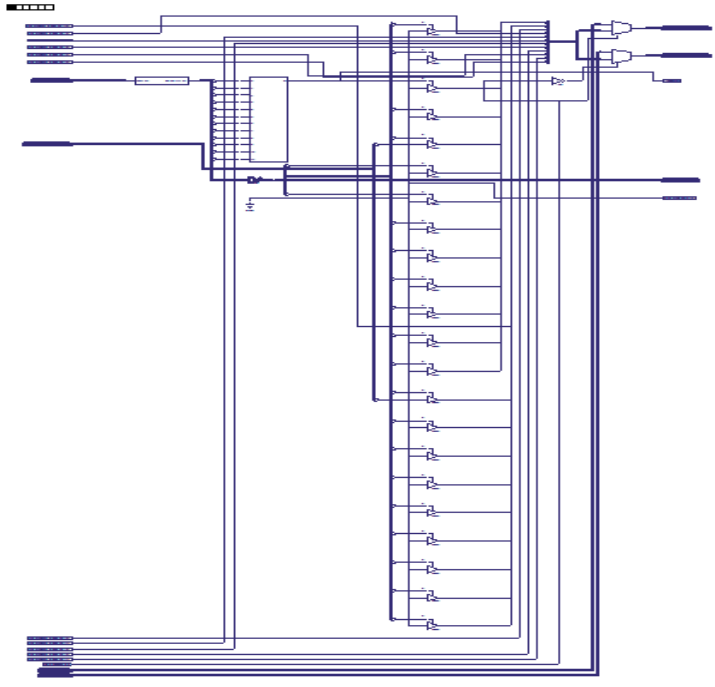


Figure 6.19 Alignment stage internal architecture

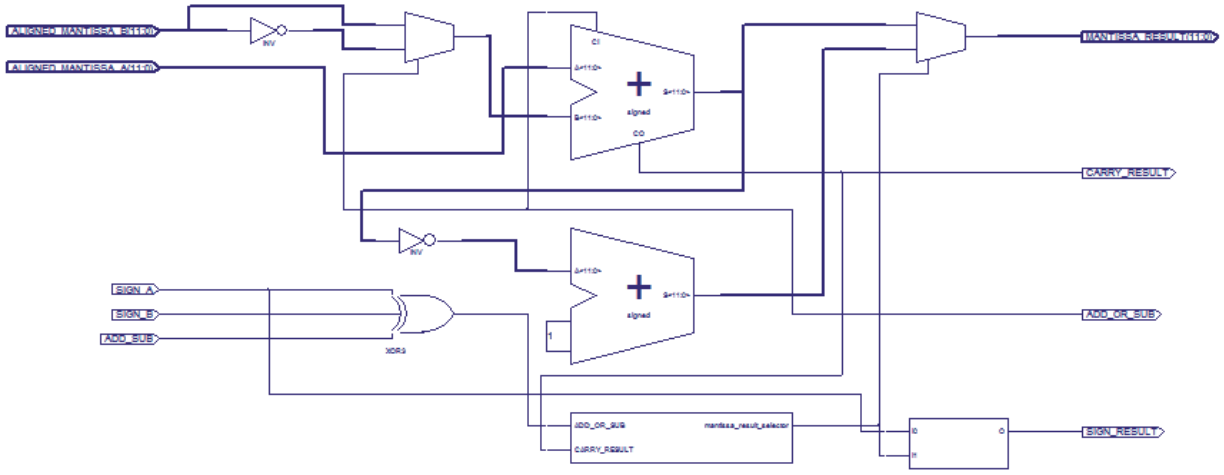


Figure 6.20 Adder/subtractor stage internal architecture

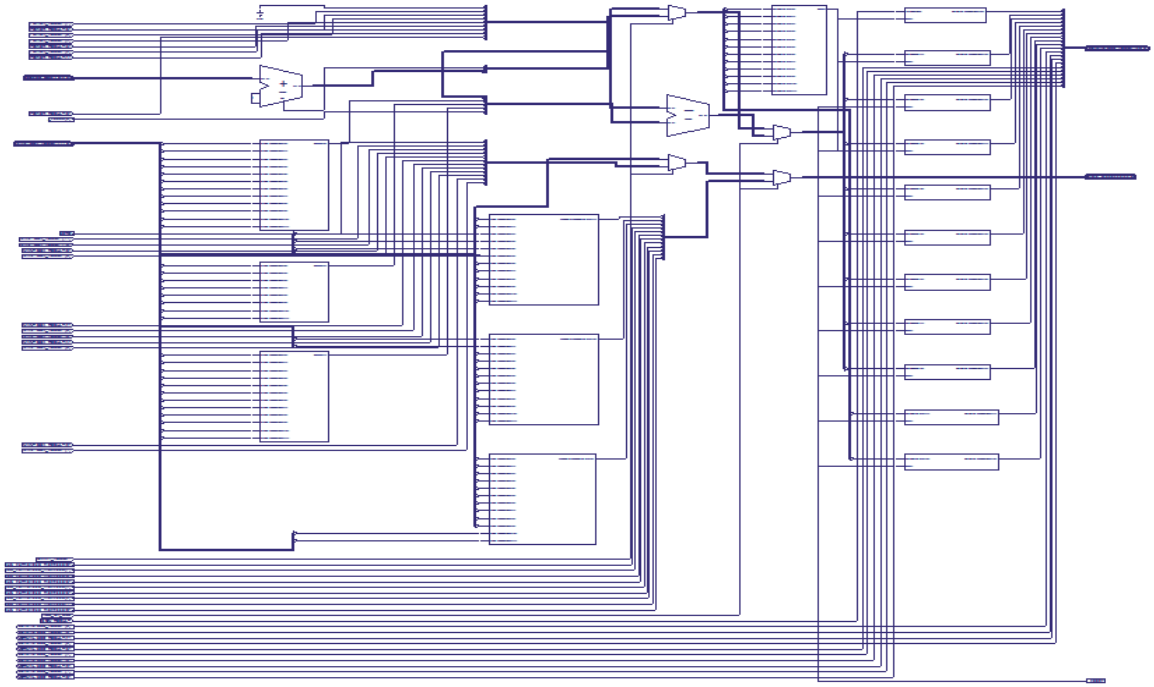


Figure 6.21 Normalized stage internal architecture

Efficient algorithm of high resolution high speed low area floating-point adder/subtractor with reducing mean latency for FFT in high resolution applications was designed and investigated. Each stage of the floating point adder/subtractor unit, calculates the arithmetic operation within 1 clock cycle. The result will be entered to the pipeline register which in turn will result in a reduction of the latency. The unique structure of this adder ignores the shifter cell and replaces them with multiplexer to reduce the delay propagation through each cells. The evaluation indicates that the proposed pipelined adder is attractive due to its high resolution (32-bit floating point), low area (6691 gate count) and low latency of 4-clock cycle. The maximum frequency for this adder is 278.428 MHz.

Table 6.4 Comparison stage specifications

HDL synthesis report	
QTY	
Design statistics	
No. of IOs	103
No. of multiplexer	4
No. of Xor	0
Slice number	42
	(2%)
Timing report	
Minimum period (ns)	1.031
Maximum frequency (MHz)	969.744
Minimum input arrival time before clock (ns)	1.988
Maximum output required time after clock (ns)	2.775
Total equivalent gate count for design	893
Total memory usage (MB)	59

Table 6.5 Alignment stage specifications

HDL synthesis report	
Design statistics	
No. of IOs	126
No. of multiplexer	2
No. of Xor	0
Slice number	15 (10%)
Timing report	
Minimum period (ns)	0.847
Maximum frequency (MHz)	1181
Minimum input arrival time before clock (ns)	4.032
Maximum output required time after clock (ns)	2.775
Total equivalent gate count for design	2211
Total memory usage (MB)	65

Table 6.6 Add/subtractor stage specifications

HDL synthesis report	
Design statistics	
No. of IOs	77
No. of multiplexer	1
No. of Xor	1
Slice number	68 (4%)
Timing report	
Minimum period (ns)	2.377
Maximum frequency (MHz)	420.76
Minimum input arrival time before clock (ns)	2.299
Maximum output required time after clock (ns)	2.779
Total equivalent gate count for design	1615
Total memory usage (MB)	69

Table 6.7 Normalized stage specifications

HDL synthesis report	
Design statistics	
No. of IOs	70
No. of multiplexer	4
No. of Xor	0
Slice number	201 (13%)
Timing report	
Minimum period (ns)	2.062
Maximum frequency (MHz)	484.919
Minimum input arrival time before clock (ns)	5.522
Maximum output required time after clock (ns)	2.775
Total equivalent gate count for design	3687
Total memory usage (MB)	61

Table 6.8 Overall Floating point adder/subtractor specifications

HDL synthesis report	
Design statistics	
No. of IOs	104
No. of multiplexer	11
No. of Xor	1
Slice number	349 (22%)
Timing report	
Minimum period (ns)	3.592
Maximum frequency (MHz)	278.428
Minimum input arrival time before clock (ns)	2.534
Maximum output required time after clock (ns)	2.779
Total equivalent gate count for design	6691
Total memory usage (MB)	70

As shown in the results, the total slice number in the overall pipeline floating-point adder is less than the sum of the each cell block. It is due to the advance configurations which avoid repetition of similar slice and the reuse of the similar slice again.

The advance design algorithm leads to have the minimum equivalence of gate counts (6691 gates) results in area saving and power conservation reduction that satisfies low area and low power on the chip core. This value is 37977 gates as found in previous research work (Huang et al. 2005).

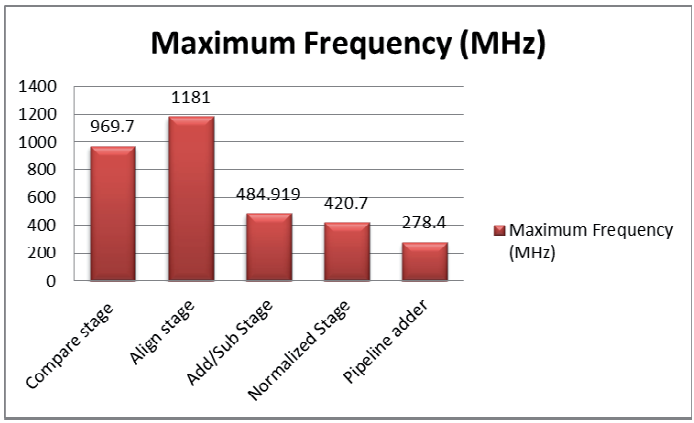


Figure 6.22 Adder stage frequency comparison

Beside the maximum clock frequency, the total equivalent gate was counted for each individual stage of the floating point adder/subtractor to give estimation for the area calculation in the silicon.

Figure 6.23 shows the total equivalent gate count whilst Figure 6.24 shows the speed comparison of the proposed floating-point adder/subtractor versus previous similar research works.

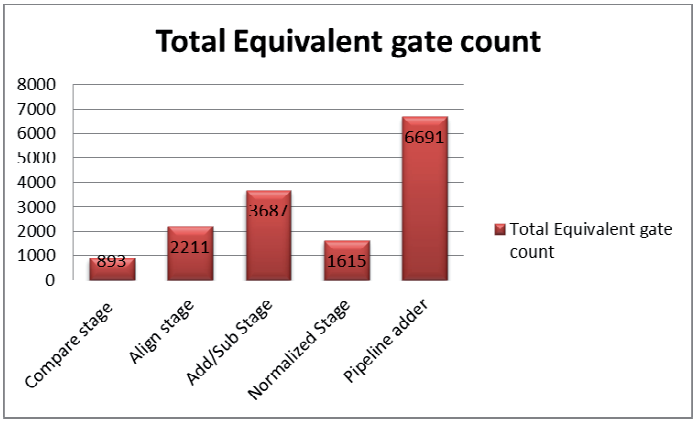


Figure 6.23 Adder stage frequency comparison

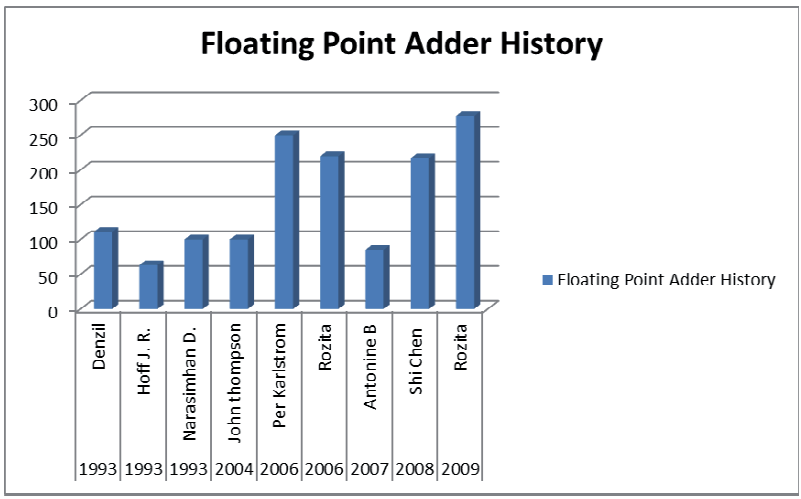


Figure 6.24 Floating-point adder speed comparison

The floating-point adder/subtractor produces the output of the arithmetic calculation after the 4-clock cycles. This delay is considered as low latency for the floating-point adder/subtractor unit when it is compared with previous research works for floating-

point architecture in recent years and is most suitable for high efficiency architecture. Figure 6.25 compares the latency of the adder for different research work whilst Figure 6.26 shows the appearance of the output after 4-clock cycle in the proposed adder/subtractor.

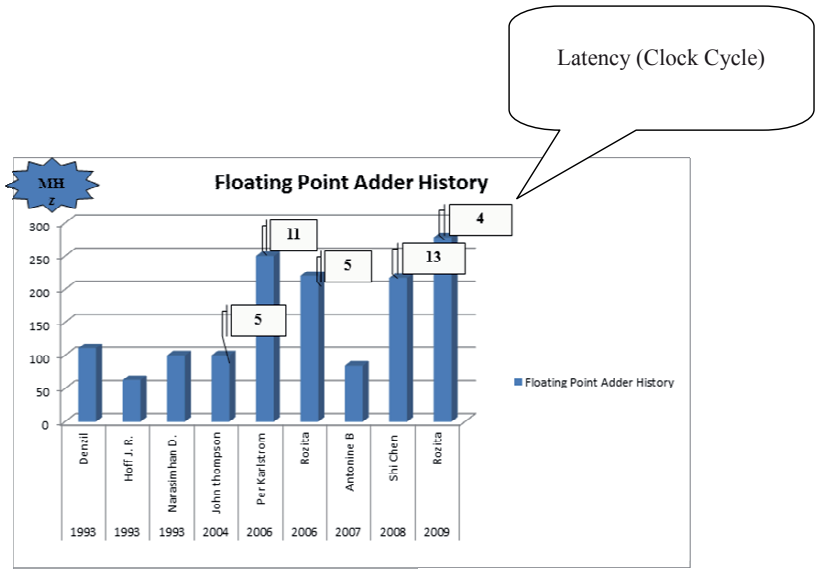
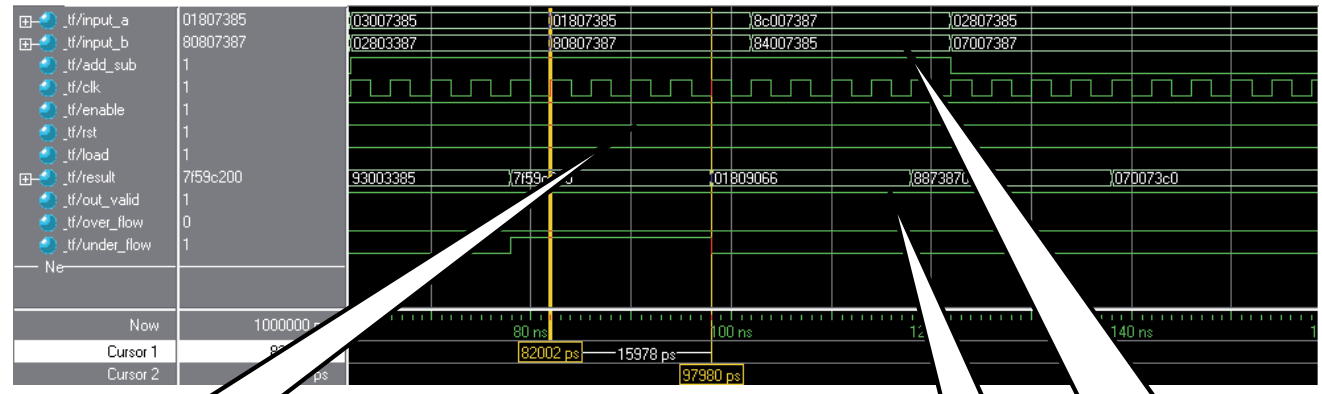


Figure 6.25 Floating point adder latency comparison



4 clock cycle

A and B as input data

Result as output data

Figure 6.26 Floating point adder/subtractor output signal

b) Floating-Point Multiplier Implementation

As discussed, the floating-point multiplier consists of two stages known as the multiplier stage and normalized stage. To design the floating-point multiplier, the propagation delay for each unit is only one clock cycle. Figure 6.27 illustrates the internal architecture of the proposed multiplier whilst Figure 6.28 shows a part of the 32-bit multiplier layout in detail.

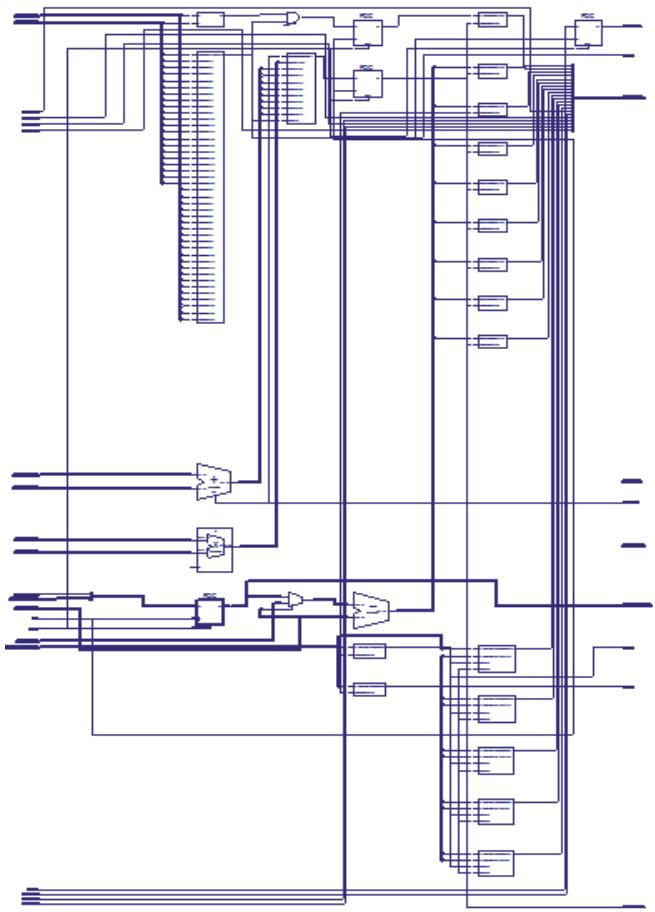


Figure 6.27 Multiplier internal architecture

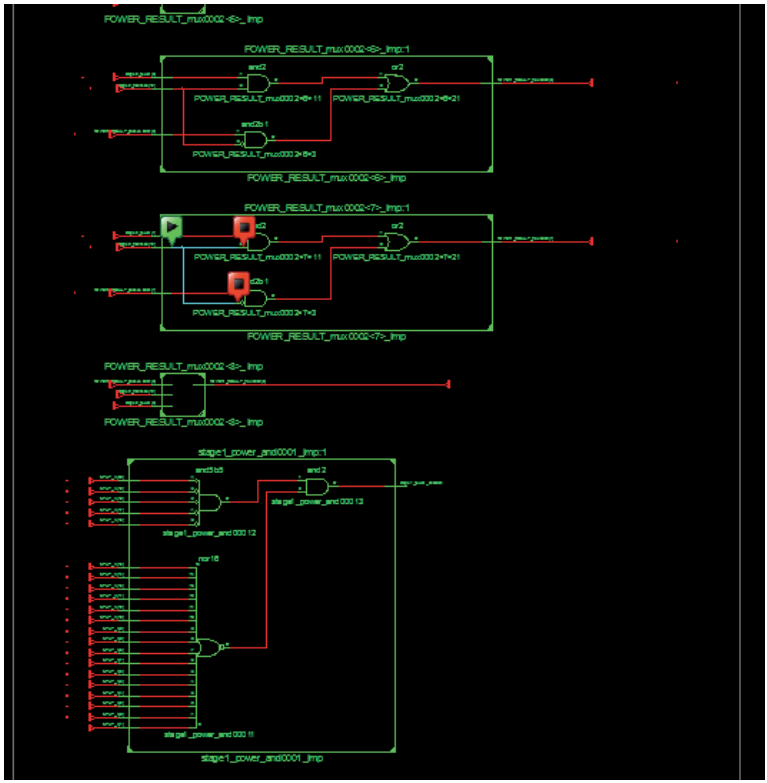


Figure 6.28 Multiplier internal architecture layout

The Xilinx ISE synthesize software produces the specification of the proposed floating-point multiplier which is given in Table 6.9. As shown in the table, the maximum clock frequency of this multiplier is 322 MHz and the system is more relaxed in gate count in compare with adder/subtractor unit due to multiplier does not require the alignment stage. The overall gate count available in proposed floating-point multiplier is 4878 gates. The floating-point multiplier unit performs the multiplication within two clock cycle; hence the latency of the multiplier is very low and is suitable for high performance architecture.

Table 6.9 Proposed floating-point multiplier specifications

HDL Synthesis Report		Timing Summary	
Registers Flip-Flops	202	Minimum period (ns)	3.1
LUTs Slice	78	Maximum Frequency (MHz)	322
Logic Slice	24	Min. input arrival time (ns)	4.216
RAM Cells	-	Max. output required time (ns)	4.705
IOs	101	Total equivalent gate count	4878
Memory usage (MB)	134	Total Number of Path	1785
Multiplexers	3	Total Number of Destinations	26
Tri-states	53		

The MODELSIM simulation result of the multiplier is shown in Figure 6.29 .This figure proves the two clock cycle latency for the proposed floating-point multiplier.

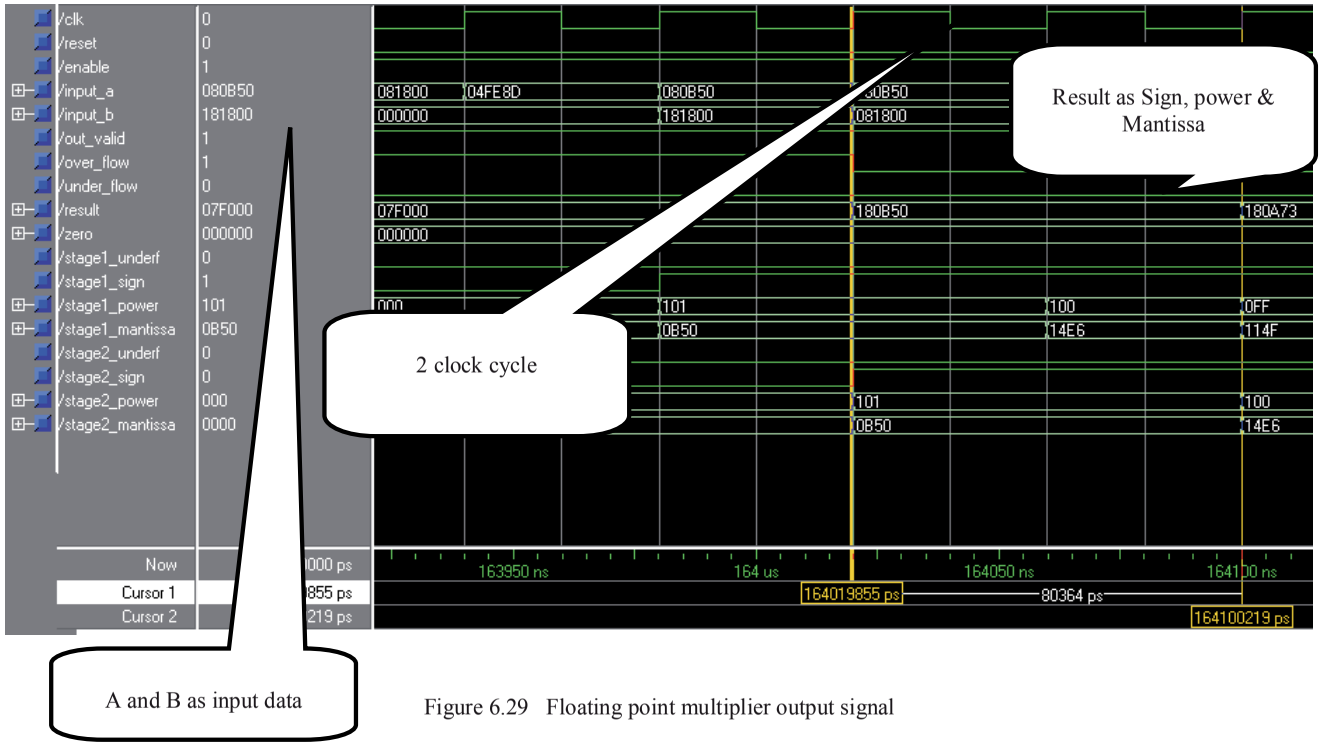


Figure 6.29 Floating point multiplier output signal

6.1.4 CONTROLLER UNIT ARCHITECTURE

As discussed, the data path is controlled by a controller unit which is a state machine with all necessary control signals to synchronize the movement of data through the different blocks of the data path. Figures 6.30 and 6.31 illustrate the architecture of the controller unit. The states consist of storage and control of the twiddle factor and its movement through the stage and reading the proper input data. It also consists of the storing of the result after the FFT calculation was executed accordingly. The multiplexer sends the input vectors to the permutation which permutes the data and sends them to the butterfly calculation.

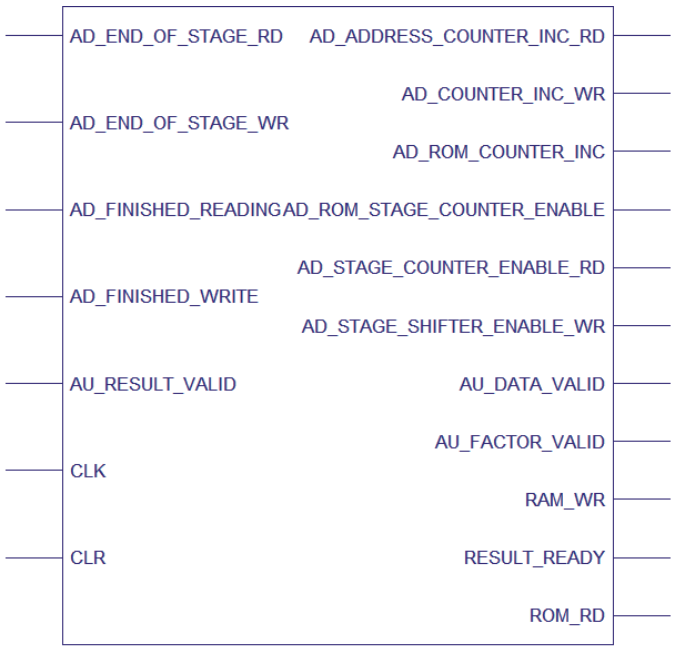


Figure 6.30 Top- module of controller unit

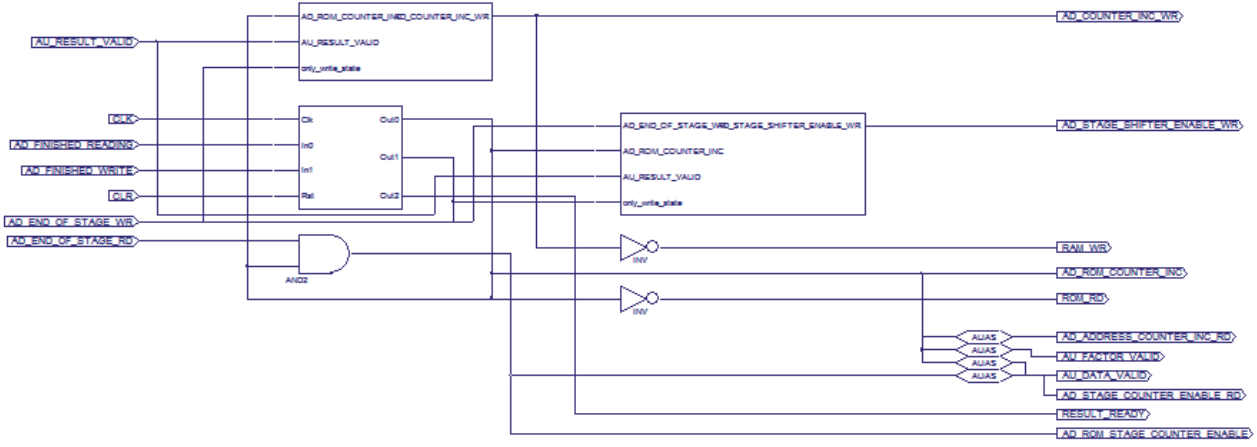


Figure 6.31 Controller internal architecture

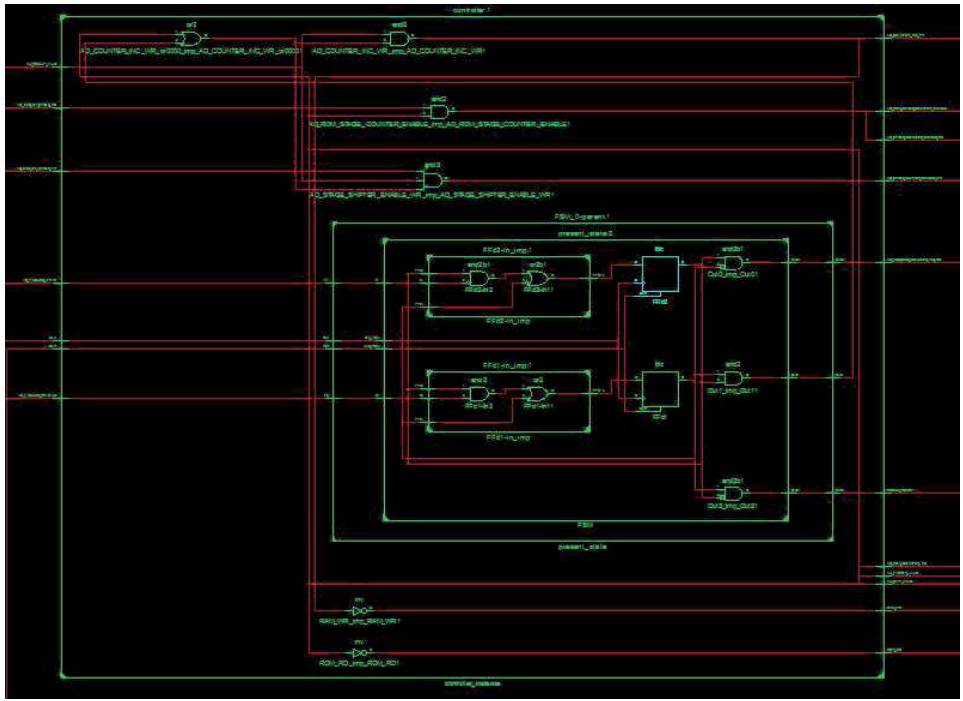


Figure 6.32 Controller architecture layout

The Xilinx ISE synthesize result shows the total equivalent gate of 312 with the maximum clock frequency as 1 GHz for controller architecture. The specification of high performance controller is given in Table 6.10.

Table 6.10 Proposed controller specifications

HDL Synthesis Report		Timing Summary	
Registers Flip-Flops	2	Minimum period (ns)	0.937
LUTs Slice	9	Maximum Frequency (MHz)	1067
Logic Slice	2	Min. input arrival time (ns)	1.329
RAM Cells	-	Max. output required time (ns)	3.445
IOs	18	Total equivalent gate count	312
Memory usage (MB)	120	Total Number of Path	43
Multiplexers	-	Total Number of Destinations	22
Tri-states	31		

6.1.5 RAM and ROM Architecture

In order to obtain the FFT core with the required flexibility, it is necessary to identify an efficient strategy to integrate the butterfly processor and the communications scheme with memory. The proposed single-chip implementation of FFT processor in this book is to provide flexibility with an internal ROM and RAM; which are to store the twiddle factor and the input/output data respectively.

Although this architecture increases the active core area, it however leads the processor to operate independently regardless of the different external devices in the scene of memories.

Figures 6.33 and 6.34 show the architecture of the entire memory systems. The RAM with the dual complex RAM structure occupied larger active core area as compared with the twiddle factor ROM. The specification results provided by the Xilinx ISE synthesis software shows the particular of each individual memory cell. (Tables 6.11-6.12)

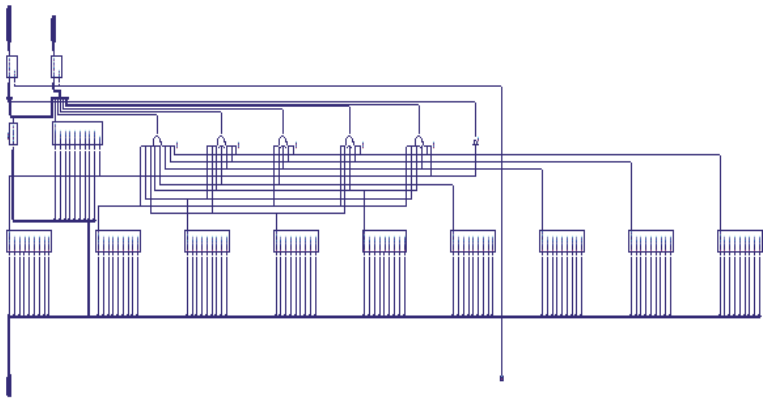


Figure 6.33 ROM internal architecture

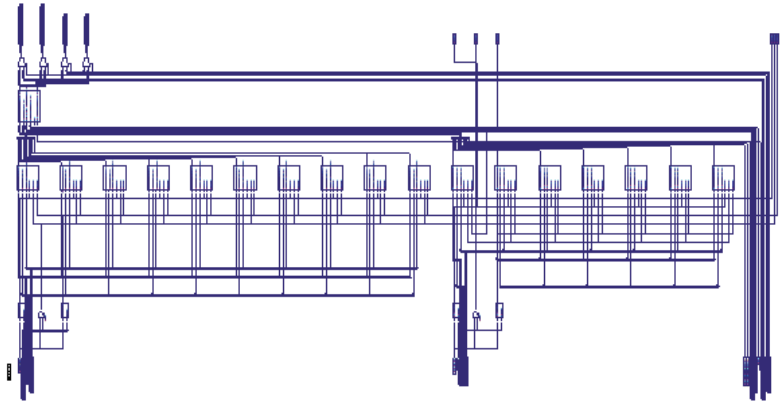


Figure 6.34 RAM internal architecture

The Xilinx ISE synthesis software results of the internal layout architecture of RAM and ROM are shown in Figures 6.35 and 6.36 respectively (illustrate the synthesis software results).

Table 6.11 Proposed RAM specifications

HDL Synthesis Report		Timing Summary	
LUTs Slice	734	Min. input arrival time (ns)	3.443
Logic Slice	734	Max. output required time (ns)	5.085
		Max. Combinational path (ns)	5.739
Dual port RAM	4	Total equivalent gate count	333641
IOs	234	Total Number of Path	8672
Memory usage (MB)	237	Total Number of Destinations	1408
Multiplexers	-		
Tri-states	31		

Table 6.12 Proposed ROM specifications

HDL Synthesis Report		Timing Summary	
LUTs Slice	254	Min. input arrival time (ns)	3.685
Logic Slice	254	Max. output required time (ns)	2.989
Statistic ROM	1	Total equivalent gate count	6915
IOs	52	Total Number of Path	2092
Memory usage (MB)	139		
Multiplexers	-		
Tri-states	-		

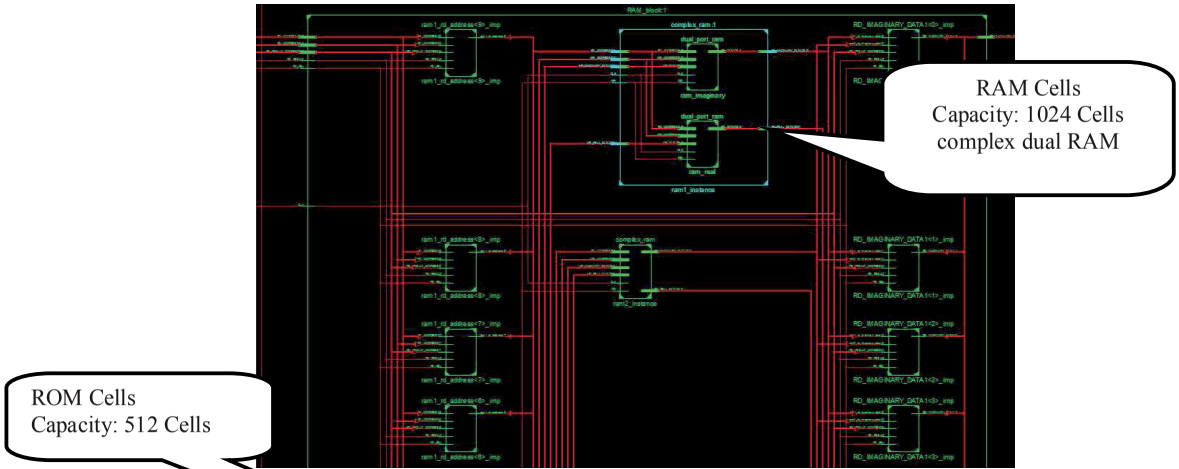


Figure 6.35 RAM architecture layout

Figure 6.36 ROM architecture layout

6.1.6 Address Generator Architecture

In order to enable parallel access to the memory units and in-place calculation, it is necessary to have the 1024 point radix-2 FPP-FFT processor address schemes to utilize by the address generator. This address generator architecture requires substantial logic unit including read address generator, write address generator and ROM address generator to generate the address and register, in order to buffer the outputs of each butterfly operation. The function of each state was discussed in Chapter IV. The specification table (Table 6.13) and the internal architecture (Figures 6.37-6.41) and the logic layout (Figure 6.42) of the address generator are given respectively.

Table 6.13 Proposed Address Generator specification

HDL Synthesis Report		**Timing Summary**	
Registers Flip-Flops	62	Minimum period (ns)	2.257
LUTs Slice	65	Maximum Frequency (MHz)	443
Logic Slice	120	Min. input arrival time (ns)	2.249
RAM Cells	-	Max. output required time (ns)	5.328
IOs	53	Total equivalent gate count	1519
Memory usage (MB)	135	Total Number of Path	198
Multiplexers	4	Total Number of Destinations	29
Tri-states	8		

As seen in the table, the address generator is able to operate with the maximum clock frequency of 443 MHz which is preferable for high performance the DSP processor. To verify the I/O port of the address generator, Figure 6.37 shows the top-module of this unit in detail.

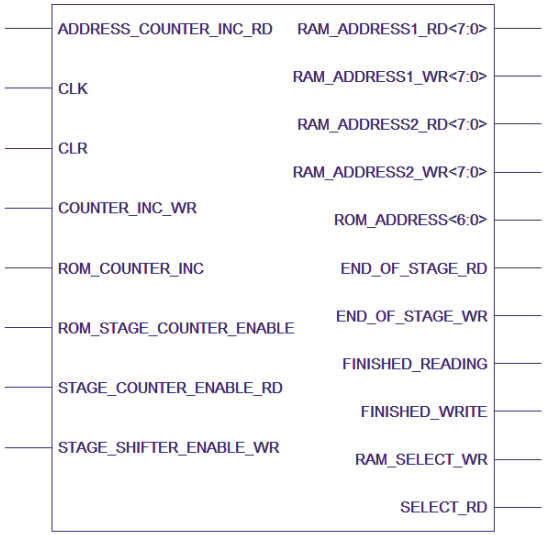


Figure 6.37 Top- module of address generator unit

As shown in the Figure 6.37, the address generator by functioning the control pins such as *counters*, *read* and *write* pins, *enables* and *end of the stage* pins, and with mutual aid of intelligent controller provides the required address for RAM, ROM and radix-2 butterfly unit.

In addition, there is a selective bit reverse sub-block in address generator architecture to sort the output address accordingly. The generated address will allocate for further calculations. The smart function of the address generator decreases hardware complexity significantly and result shows low-power consumption in entire proposed radix-2 FPP-FFT processor.

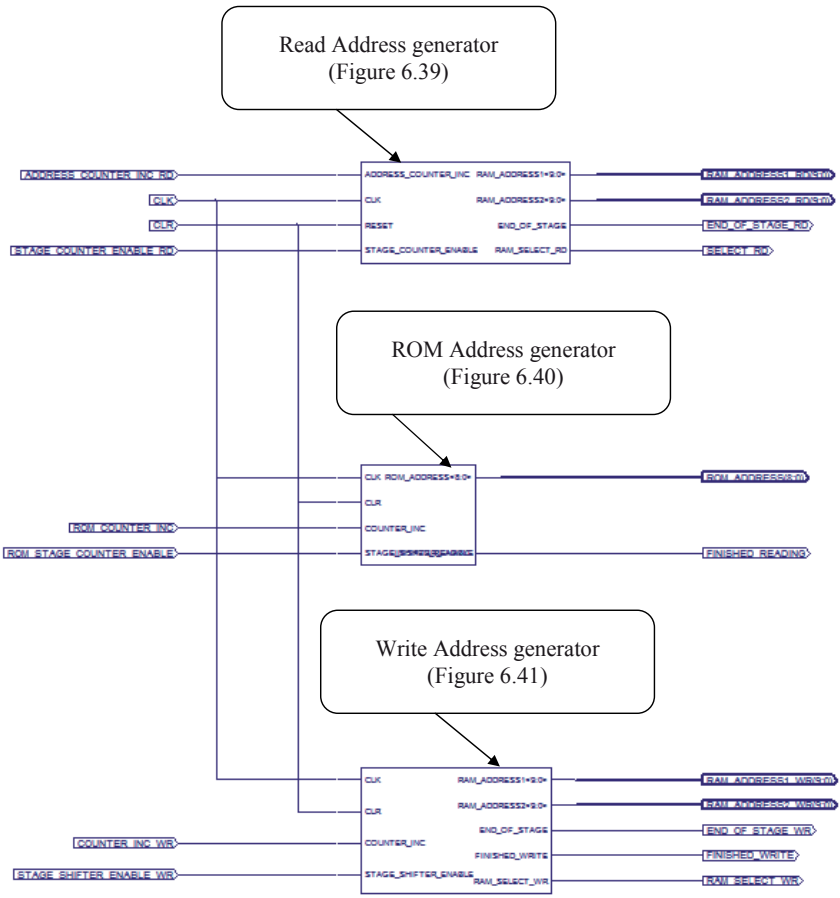


Figure 6.38 Address generator internal architecture

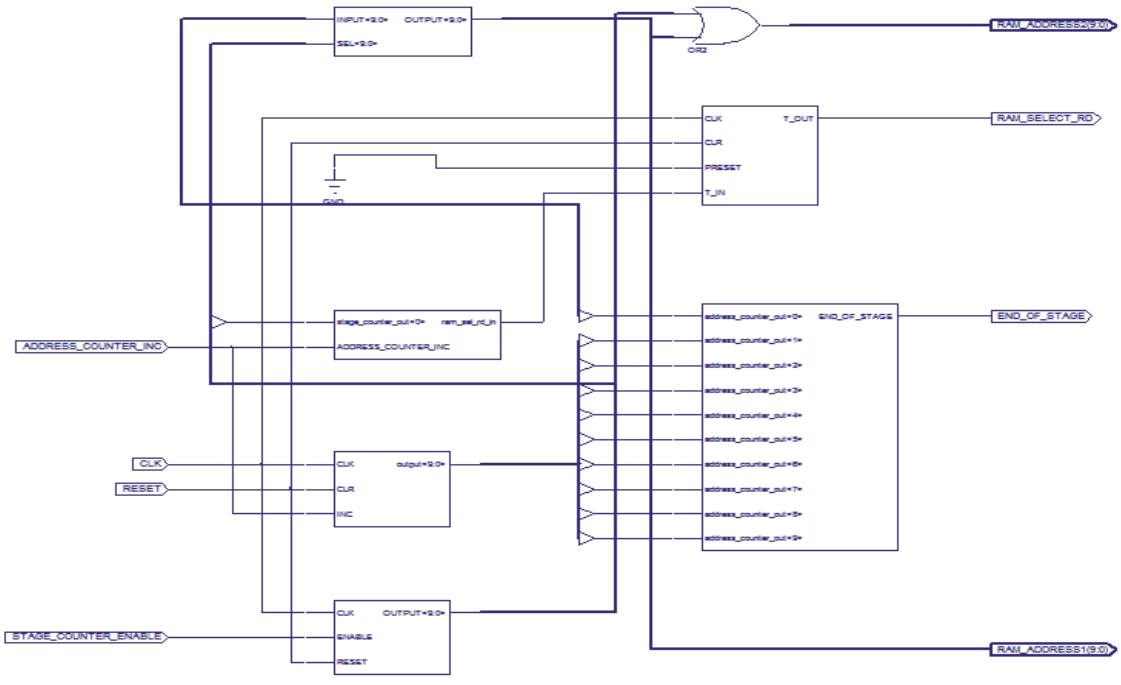


Figure 6.39 Read address generator internal architecture (Schematic)

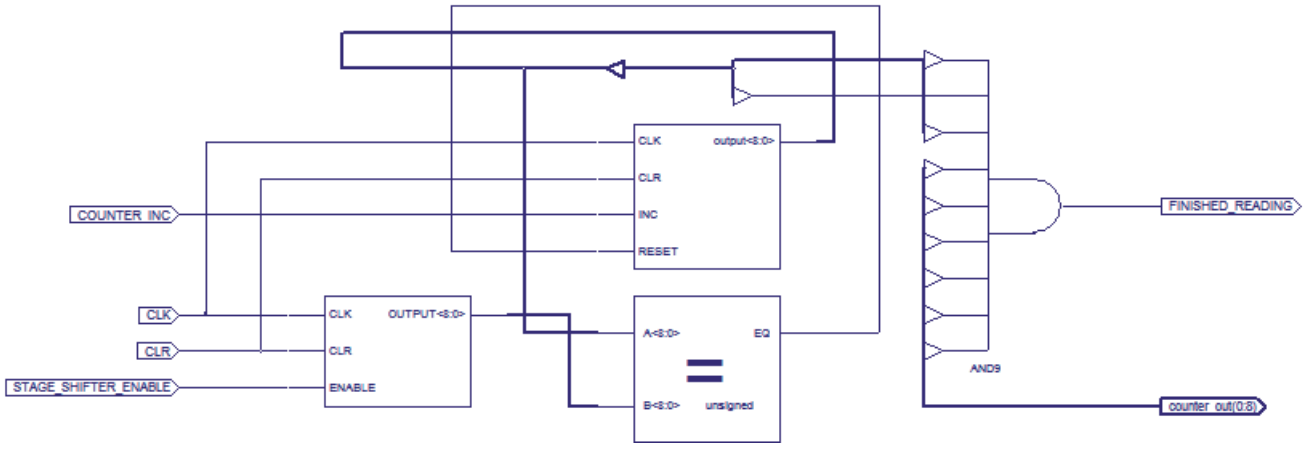


Figure 6.40 ROM address generator internal architecture (Schematic)

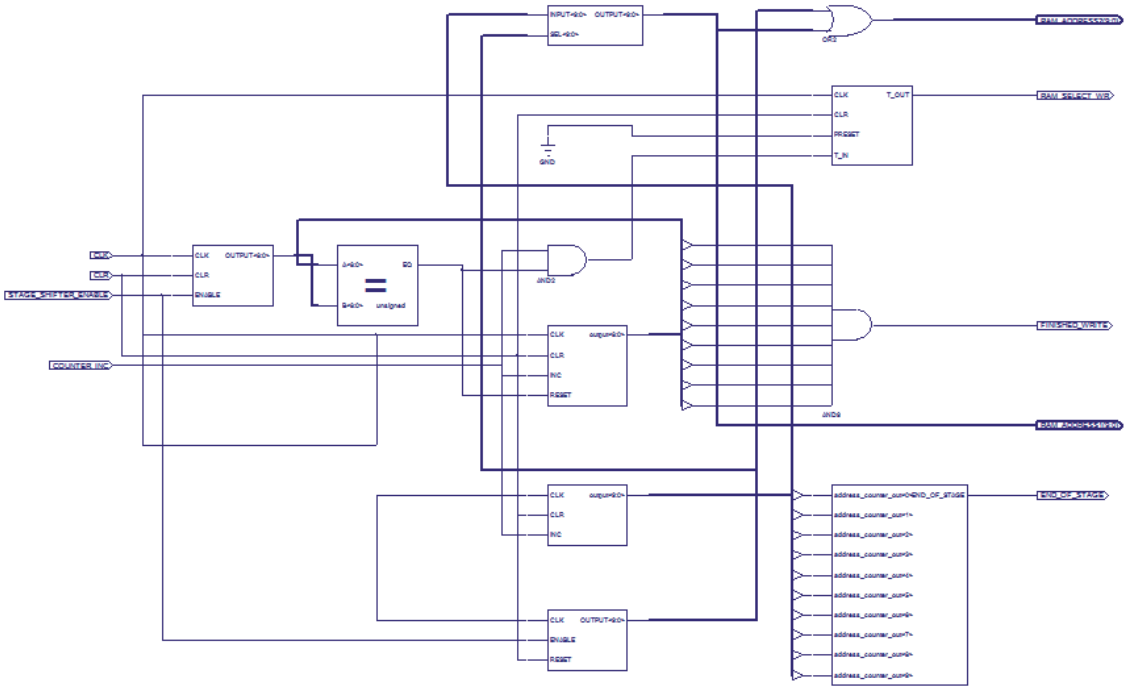


Figure 6.41 Write address generator internal architecture (Schematic)

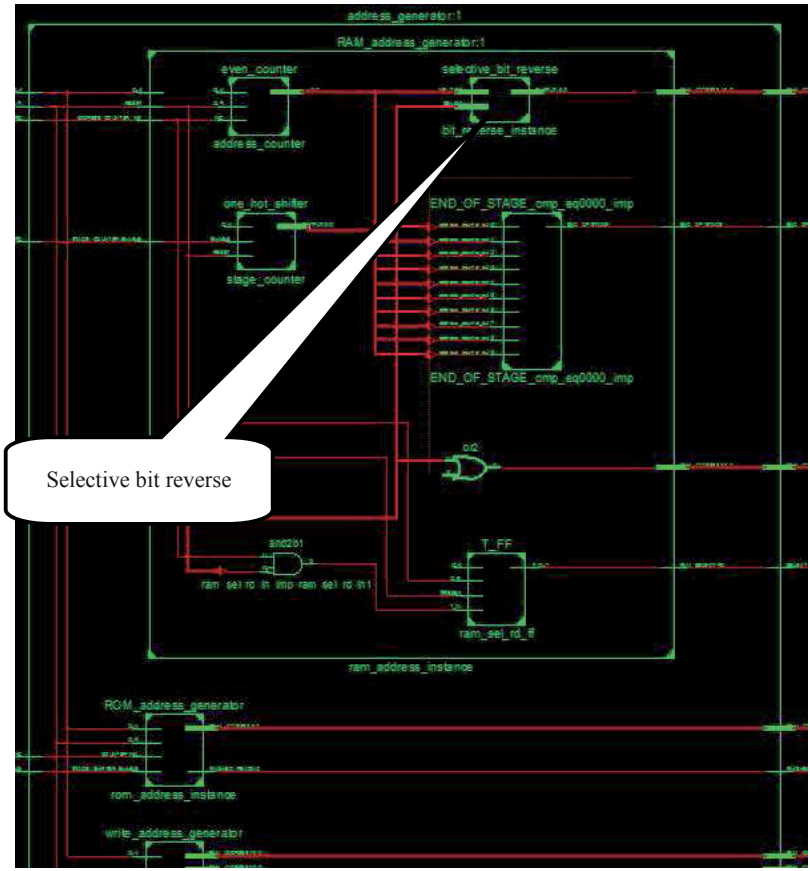


Figure 6.42 Address generator internal layout

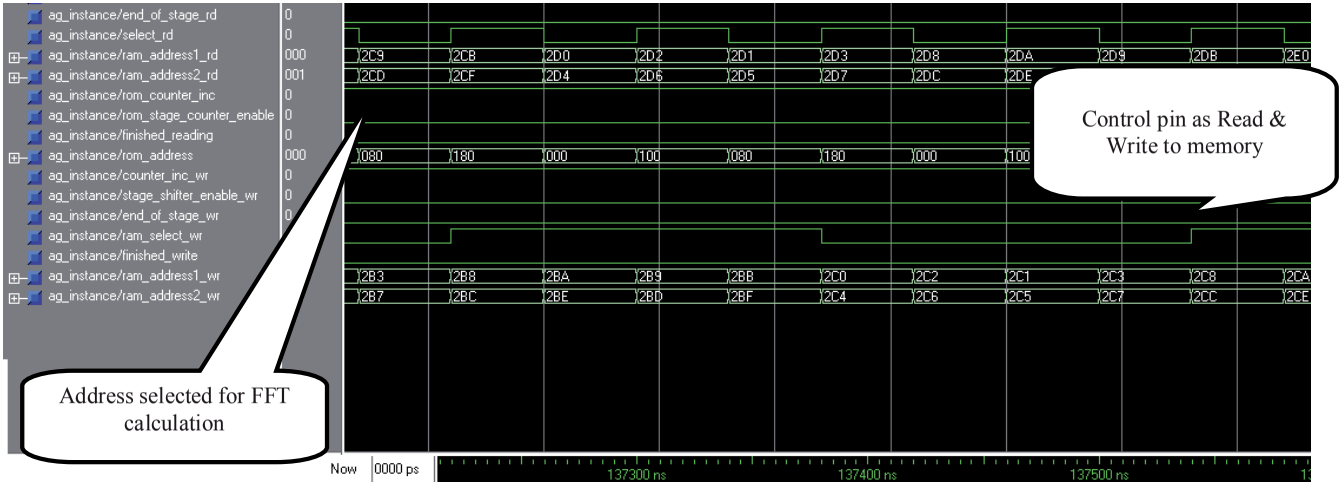


Figure 6.43 Address generator output signal

6.2 DOWNLOADING TO FPGA (Xilinx ISE software)

The overall architecture was synthesized by Xilinx ISE software and the result is shown in Figure 6.44. In order to verify the functionality of the 1024-point radix-2 FPP-FFT processor, the synthesized code is downloaded to the Virtex II- XC2V1500-6-FG676 FPGA board. Xilinx and MODELSIM Next, EDA tools were used to synthesize and to simulate the design. The output signal of the proposed processor is shown in Figures 6.45-6.47. From Figure 6.46, the total clock cycle required to implement the FFT computation for our proposed architecture is 5131 cycles and this cycles number is in agreement with the computation complexity derived from $(N/2\log_2 N)+11$ which also gives 5131 clock cycle if $N = 1024$.

The result of the proposed 1024-point radix-2 FPP-FFT processor after a successful completion of synthesis place and route (PAR) process is shown in Figure 6.48. These results are tabulated in Table 6.1. In addition, Table 6.14 shows the accuracy of the 1024-point radix-2 FPP-FTT processor in contrast with 1024-point radix-2 FFT simulation result and fixed point conventional radix-2 FFT implementation result. It compares the values obtained from the simulation and implementation of fixed-point and floating-point when the same input signals are applied for the processors. Hence, the proposed single precision floating-point FFT processor proves the resolution less than ±0.01 % error whereas the resolution in fixed-point FFT processor is less ±0.1 % error. As it is clear from Table 6.14, the proposed 1024 point floating-point FFT indicates the performance improvement which is strongly preferable for high performance application.

Table 6.14 Comparison of the simulation and implementation results in FFT processor

MATLAB FFT (Simulation)		Proposed Floating-point FFT (Implementation)		Conventional Fixed point FFT (Implementation)	
Input x(n) = 14 samples		Input x(n) = 14 samples		Input x(n) = 14 samples	
Real	Imaginary	Real	Imaginary	Real	Imaginary
244	0	244	0.0000	243.9100	0.0000
222.1080845	-86.46028488	222.1080	-86.4601	221.9900	-86.4501
163.4574705	-149.9863166	163.4570	-149.9860	163.4560	-149.9770
86.44938802	-175.7548947	86.4490	-175.7540	86.4390	-175.7240
13.94525101	-161.9378169	13.9460	-161.9378	13.9440	-161.9375
-35.05249811	-119.5011318	-35.0525	-119.5012	-35.0514	-119.3999
-51.77981999	-67.19575906	-51.7789	-67.1958	-51.7399	-67.1936
-39.7275488	-23.97775135	-39.7275	-23.9778	-39.7266	-23.9763
-11.78182712	-2.044340937	-11.7811	-2.0440	-11.7731	-2.0333
15.98247349	-3.230116609	15.9833	-3.2301	15.9833	-3.2280
31.04140023	-19.92463871	31.0414	-19.9242	31.0393	-19.9237
28.87413771	-39.69060396	28.8741	-39.6906	28.8757	-39.6810
13.51218305	-51.26607916	13.5122	-51.2663	13.5002	-51.2332
-5.614881197	-49.25132592	-5.6149	-49.2511	-5.6101	-49.2524

2) The verified proposed 1024-point radix-2 FPP-FFT processor VHDL codes were downloaded to the FPGA using the Xilinx ISE soft tools and JTAG hardware. The downloading was successfully done and the routing for the FFT processor is shown in Figure 6.48.

Figure 6.50 illustrates the chip layout of the proposed FFT processor in the FPGA board after the PAR stage.

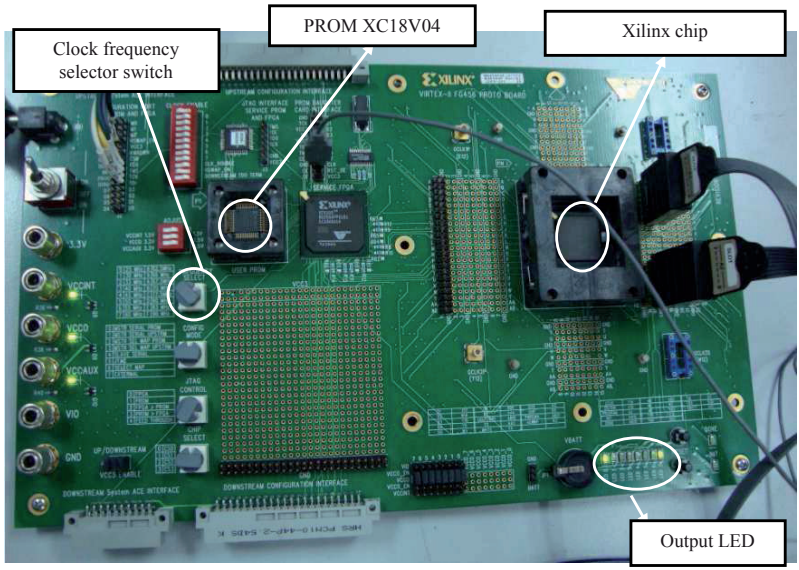


Figure 6.44 Implementation of the FPP-FFT processor on the FPGA board

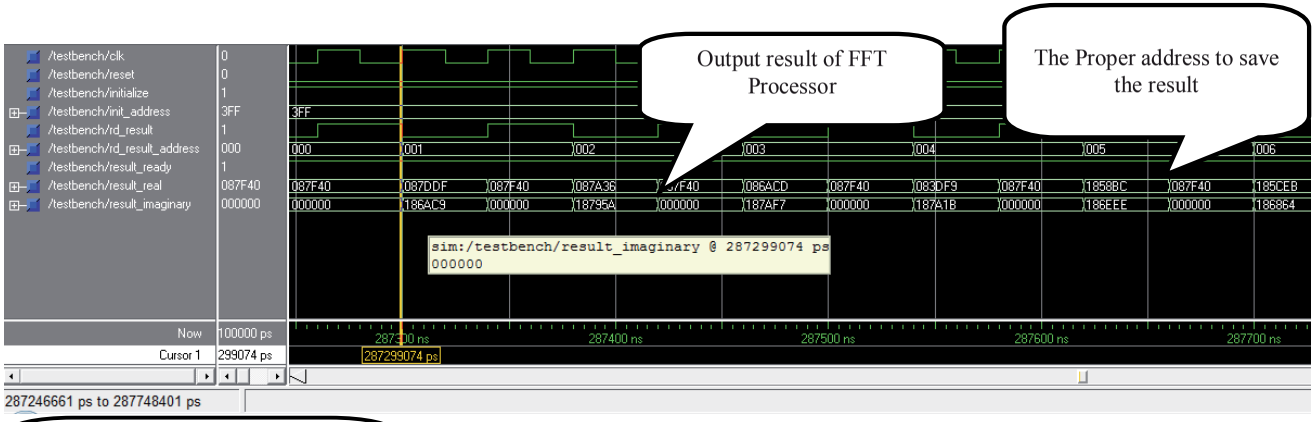


Figure 6.45 Overall FPP_FFT processor output signal

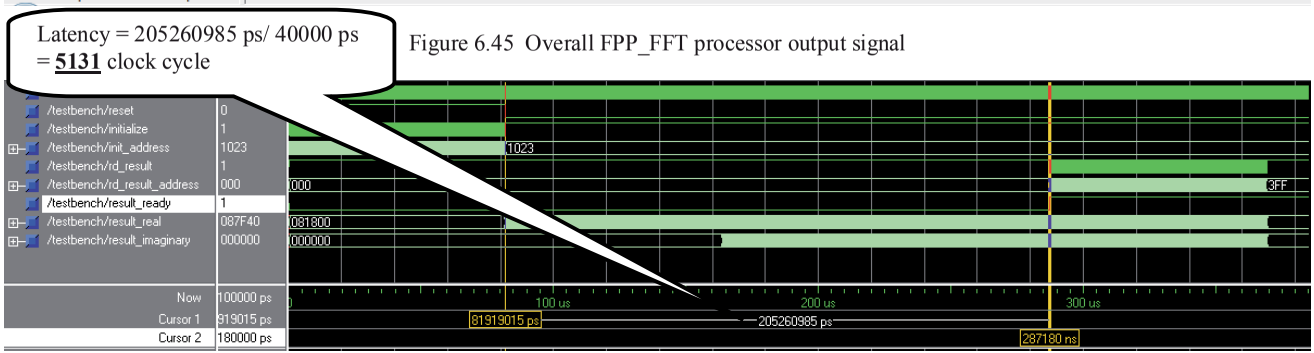


Figure 6.46 Overall FPP_FFT processor latency

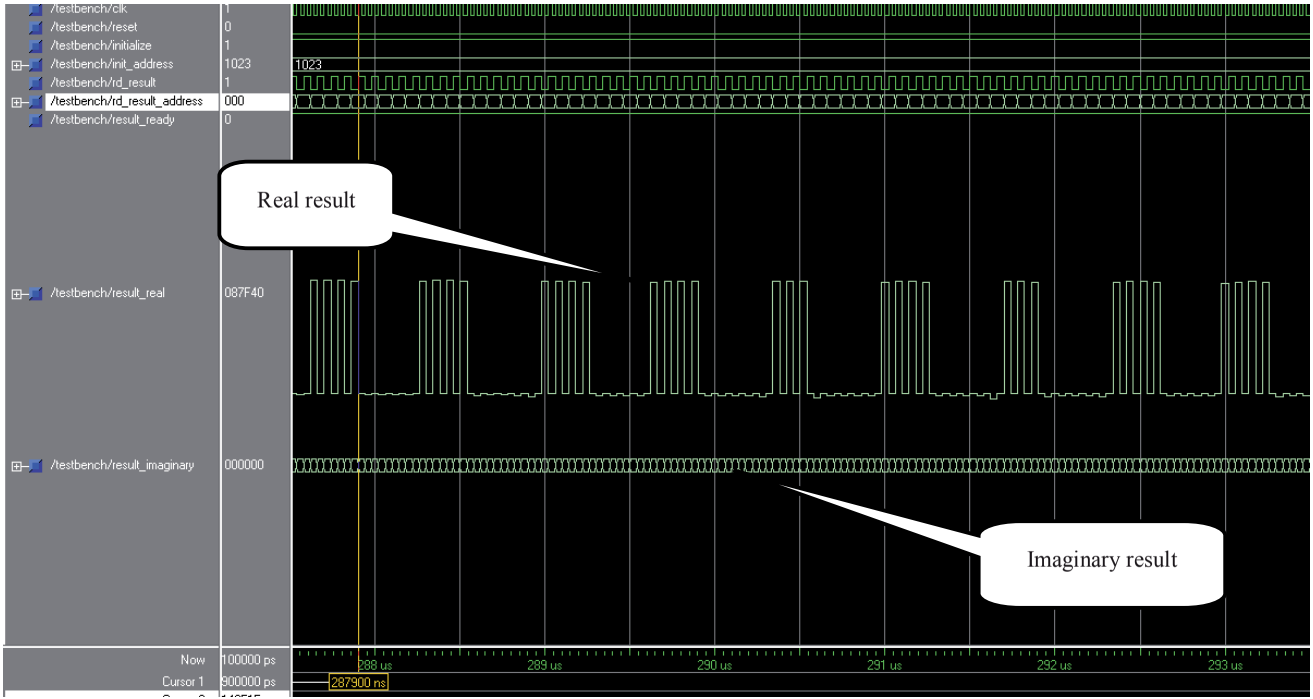


Figure 6.47 1024-point FPP-FFT processor output signal as floating point coding

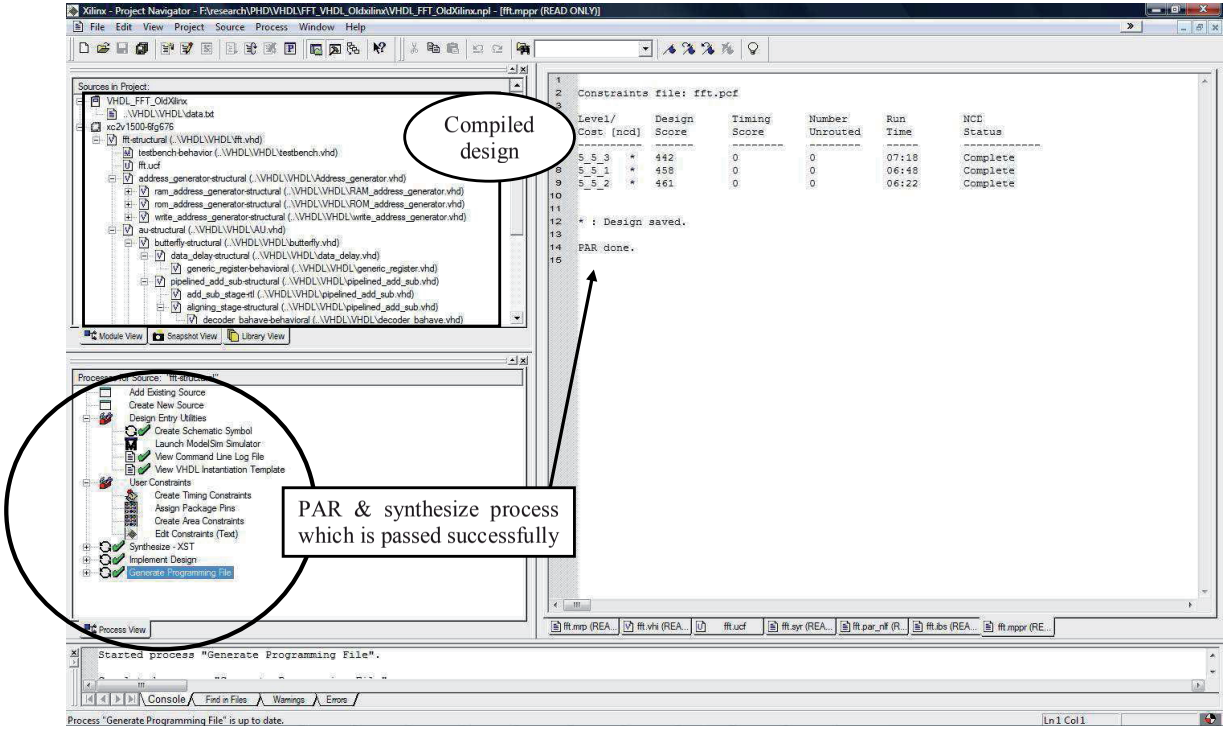


Figure 6.48 Successful implementation of place and route (PAR) for FPP-FFT in Xilinx ISE

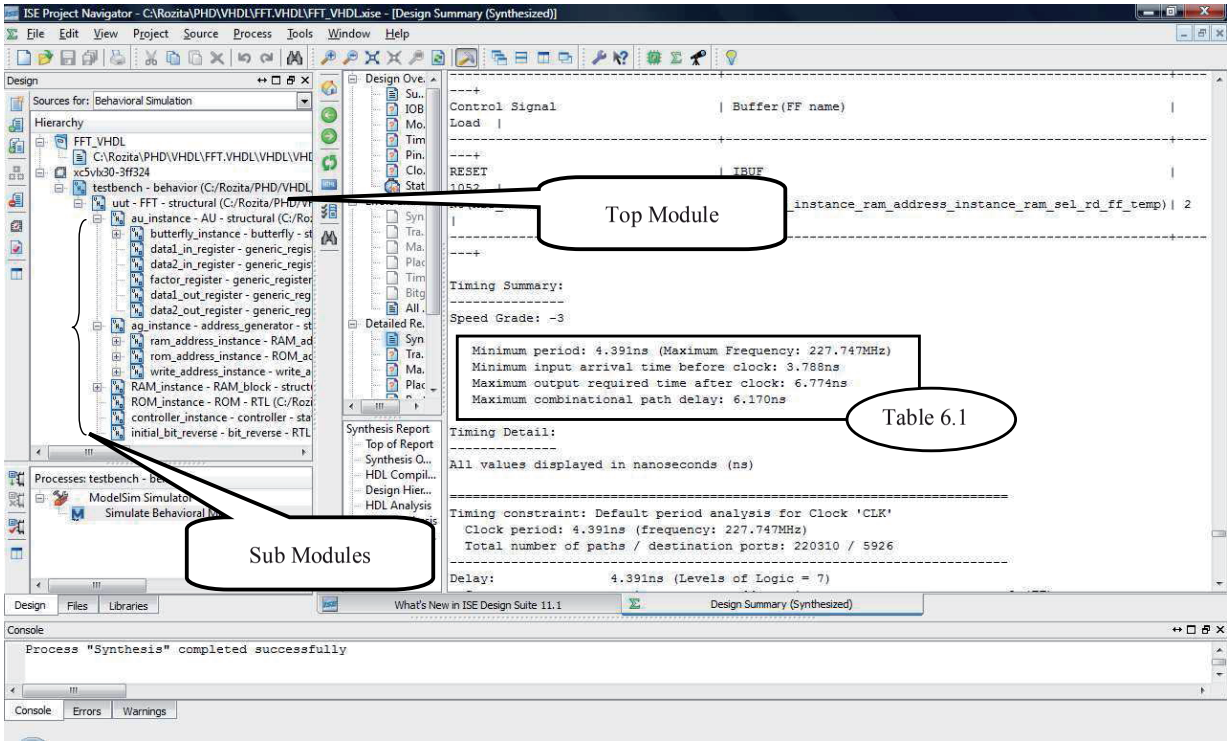


Figure 6.49 Successful synthesis process of FPP-FFT in Xilinx

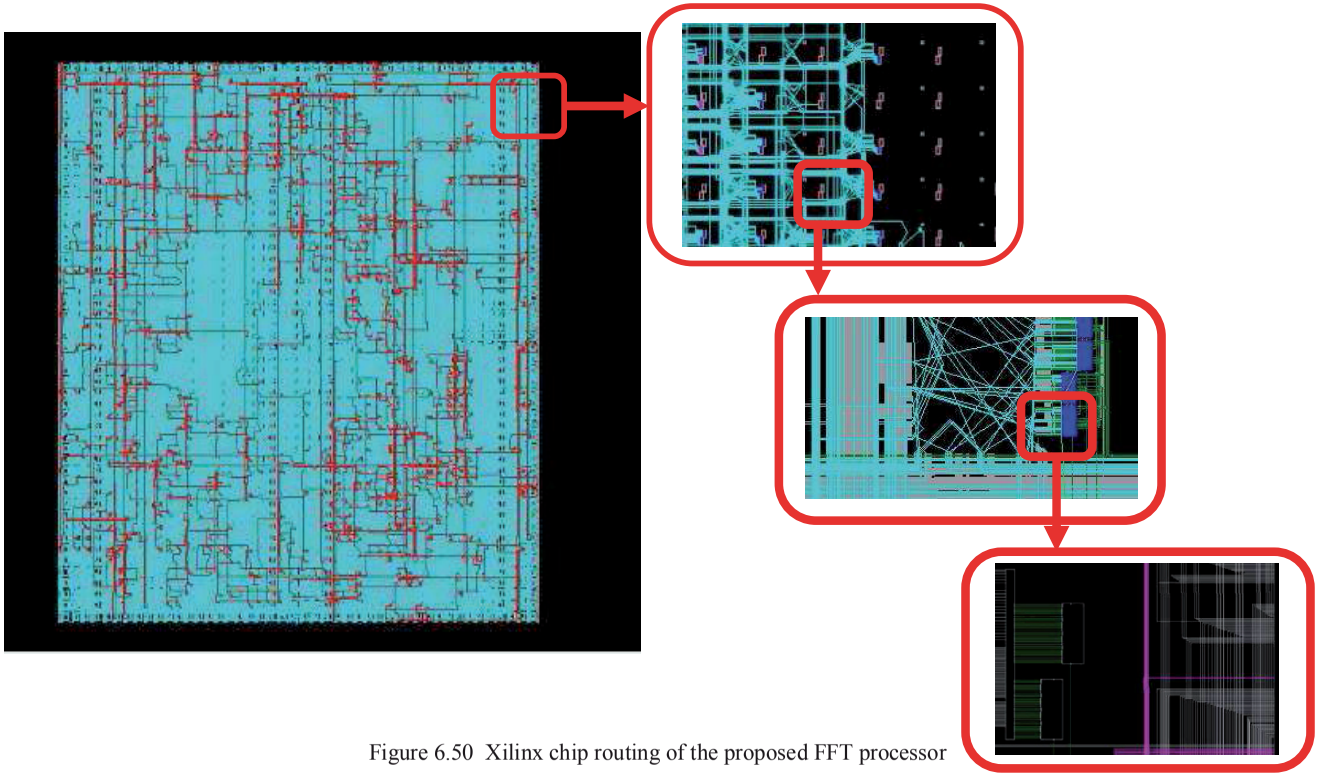


Figure 6.50 Xilinx chip routing of the proposed FFT processor

Figure 6.51 shows the testing result of the FPGA board and equipment setup for the on logic analyzer (LA) when the FFT processor is downloaded. The clock frequency is set on the maximum rate (180 MHz). The LA result shows that the processor can function correctly as shown in Figure 6.52.

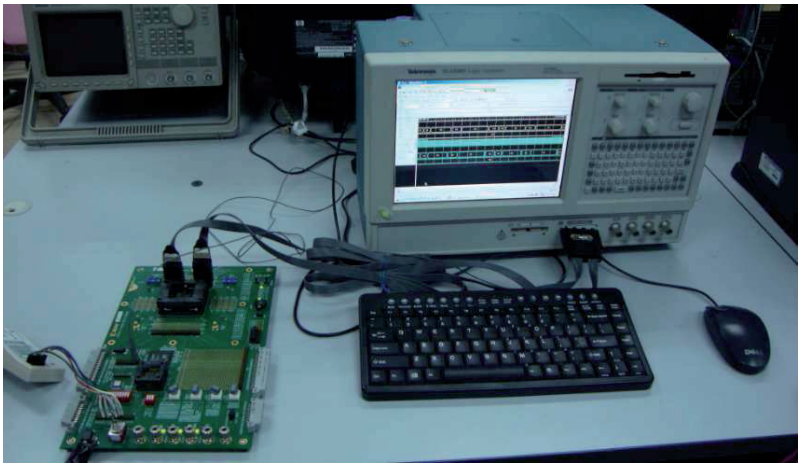


Figure 6.51 FFT processor test circuit using Logic Analyzer

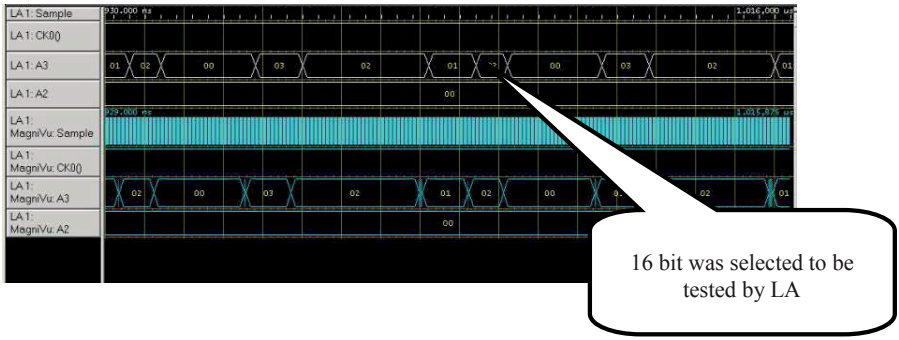


Figure 6.52 FFT processor output result displayed on Logic Analyzer

The FPGA power consumption depends on the amount of configurable blocks, which are used for the implementation of the processor. This processor was implemented by utilizing 0.18 µm and 0.35 µm technologies which are expected to have better performance compared to the FPGA chip. The results are explained in the next sections.

6.3 ASIC IMPLEMENTATION (GATE LEVEL SYNTHESIS)

An application specific integrated circuit (ASIC) is an integrated circuit (IC) modified for particular use, rather than intended for general purpose use. ASIC is often termed as a system on a chip (SOC) which will bring about greater cost savings and lower power consumption. The minimal propagation delays can be achieved in ASICs versus the FPGA. Hardware language such as Verilog and VHDL are used to describe the system functionality on ASIC.

From the explanations given, the proposed 1024 point radix-2 FPP-FFT processor is synthesized on ASIC to optimize the area and power consumption by using Synopsys and Cadence tools. The proposed processor was optimized in Synopsys tools by the passed to the PC (in the MODELSIM) in order to obtain the gate level implementation. In this level the processor is post-simulated by Cadence CAD design tools for implant and the layout is produced.

The technology libraries to execute optimization are SILTERRA 0.18 µm and MIMOS 0.35 µm. In this section the design is synthesized to convert RTL code into the gate level (Figure 6.1). The timing constraints are defined to meet the final chip result. The RTL design code is divided into two parts; the core and chip. The chip contains the top wrapper module that instantiates this core and puts I/O buffer pads along the ports. In fact, analysis and simulation are required to ensure that the power supply is adequate. In this section the functionality of the VHDL behavioral proposed FFT processor is verified by Cadence VHDL simulation. Furthermore, the RTL model of the proposed FPP FFT design is converted into the netlist (gate level).

Using design compiler (Synopsys tool), the design processor is compiled and executed. Firstly, all the module and sub-module in the radix-2 FPP FFT design are analyzed to check the VHDL syntax of the design. It is converted into intermediate format and stored in the specified library. Secondly, the top design is elaborated to convert the processor into generic gates and logic blocks. The elaboration reports the memory elements in the FFT architecture. After the elaboration process the overall system is checked for debugging the error.

Figure 6.53 shows the layout of the FFT processor active core using SILTERRA 0.18 µm technology. This layout was routed by Encounter tools. Figure 6.54 demonstrates the gate level output result of the processor produced by the NC-Launch using CAD tools. The output signals are provided from netlist of the system.

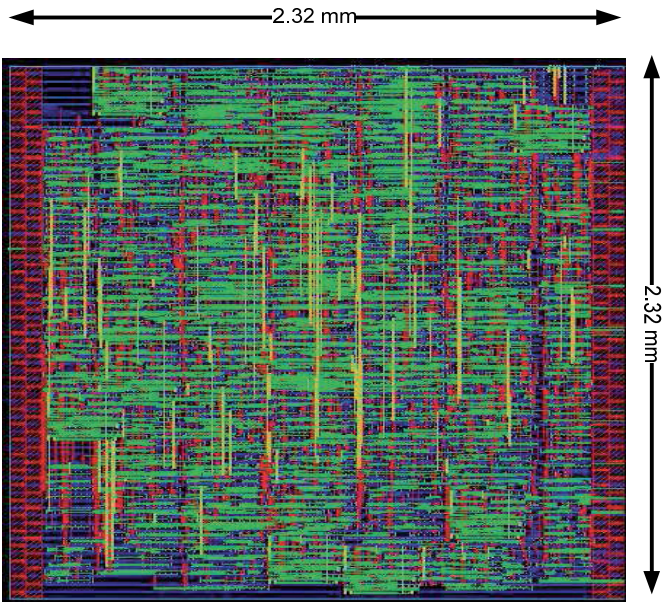


Figure 6.53 Active core die of the proposed FFT processor in SILTERRA technology

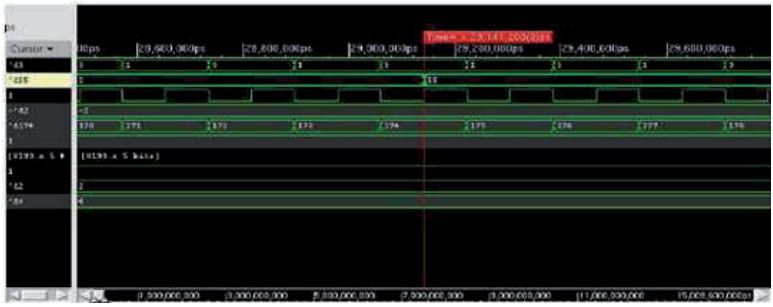


Figure 6.54 NC-Launch result of the FPP_FTT processor in the gate level

The amplitude signal achieved from the gate level design is represented in Figure 6.55 for the 1024-point input data. The plotted signal illustrates the *sinc* function produced by the rectangular input signal. The amplitude output signal functions as Fourier Transform of the rectangular signal.

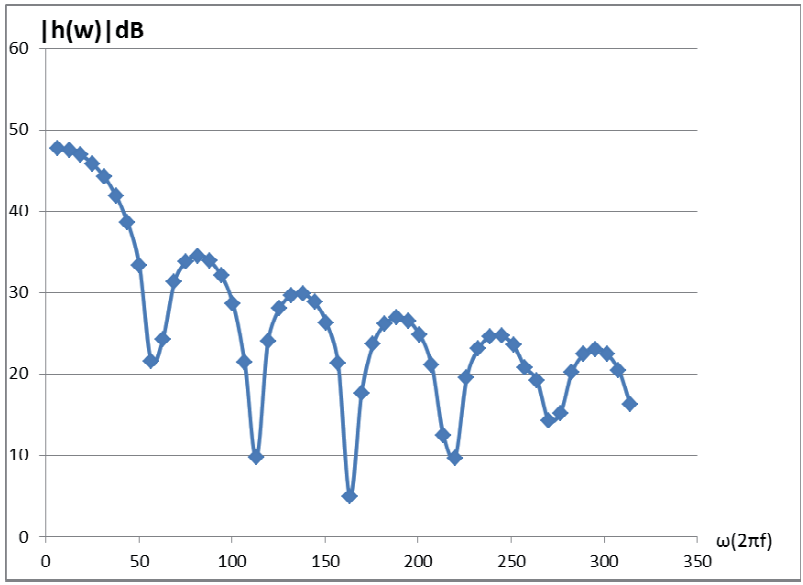


Figure 6.55 The output plot of gate level processor (Silicon)

Next, the 1024-point FPP-FFT core is compiled. After compiling, the netlist is created. The estimated area and power consumption result of the high-speed floating-point FFT processor in SILTERRA 0.18 µm and MIMOS 0.35 µm technology in maximum clock frequency is shown in Table 6.15.

Table 6.15 Estimated power/ area of proposed FFT in different technology libraries

The FPP-FFT specification	SILTERRA 0.18 µm technology	MIMOS 0.35 µm technology
Active core area (mm^2)	2.32×2.32	4.256×4.256
Power consumption (mW)	640	1198

Table 6.16 compares the proposed design with conventional type of fixed-point 1024 point FFT. The system was synthesized in the design analyzer under the SILTERRA and the MIMOS technology library.

Table 6.16 Estimated power/area of conventional FFT in different technology

Conventional FFT specification	SILTERRA 0.18 µm technology	MIMOS 0.35 µm technology
Active core area (mm^2)	9.3426×9.342	15.778×15.778
Power consumption (mW)	3.4	9.5

Figures 6.56 and 6.57 show the symbol and schematic view of the FFT processor followed by internal architecture of the proposed FFT, respectively.

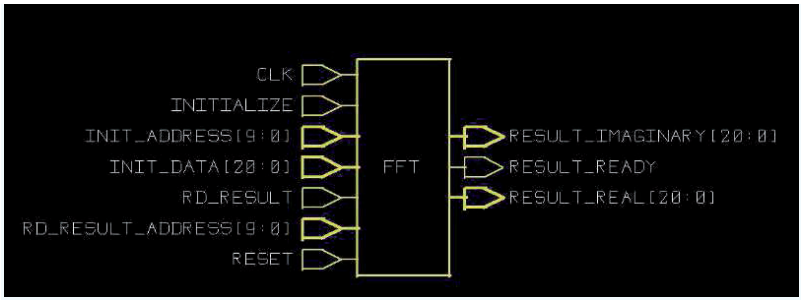


Figure 6.56 Symbol view of the FFT processor

The top-module with suffix .db is created which is the I/O pad wrapper. Some constraints need to be defined for the FPP-FFT design. These constraints are necessary conditions that have to be met for the proper functionality of the circuit. The most important constraint is related to the signal timing of the system. Constraints for clock include the period and skew.

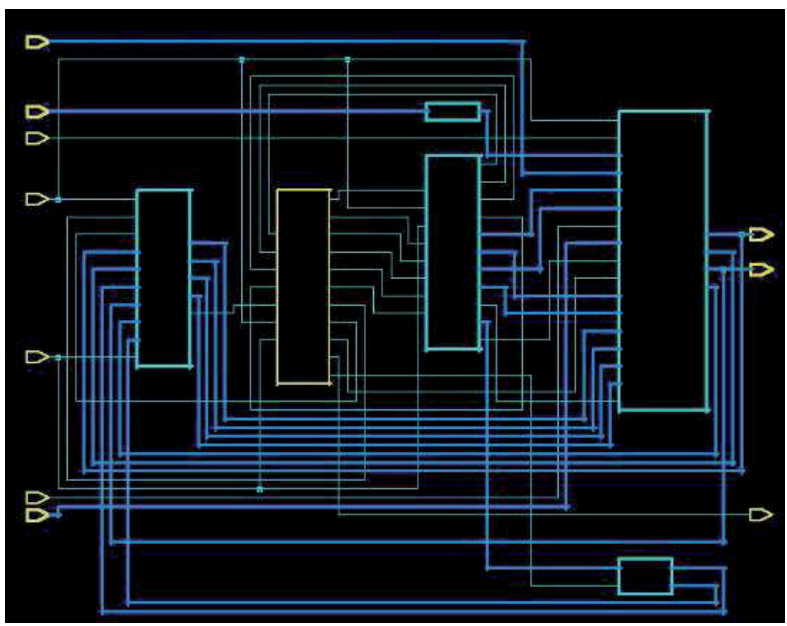


Figure 6.57 Schematic view of the FFT processor

It is necessary to specify the clock period, to show the maximum speed of the FFT design. All the constraints are fixed to the design prior compilation. These are typically applied to the top module and propagated down to the lower level module as separated step. For defining constraints, the input and output pins had to be set before compiling the FFT design. The maximum fan-out and timing are determined for driving pins to change logic values. Within Synopsys there is a power analysis tool which estimates power consumption under various conditions and configurations. After defining the constraint, the design is checked to ensure that there is no driving of multiple ports by a single net and that all constraints are defined at the top module.

The schematic gate level of the different components in the FFT processor is shown in Figures 6.58 – 6.66.

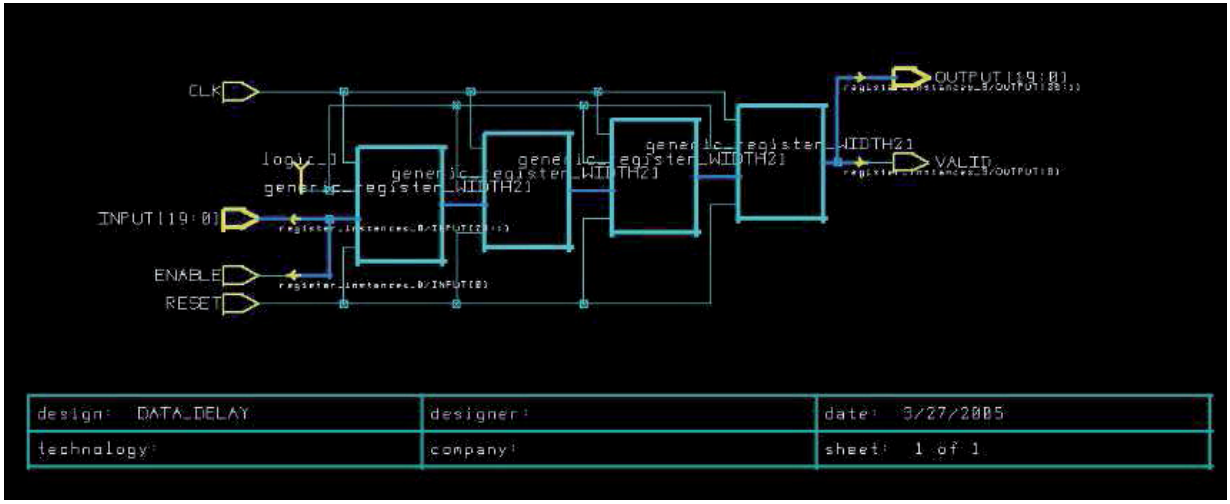


Figure 6.58 Internal implementation of the data delay

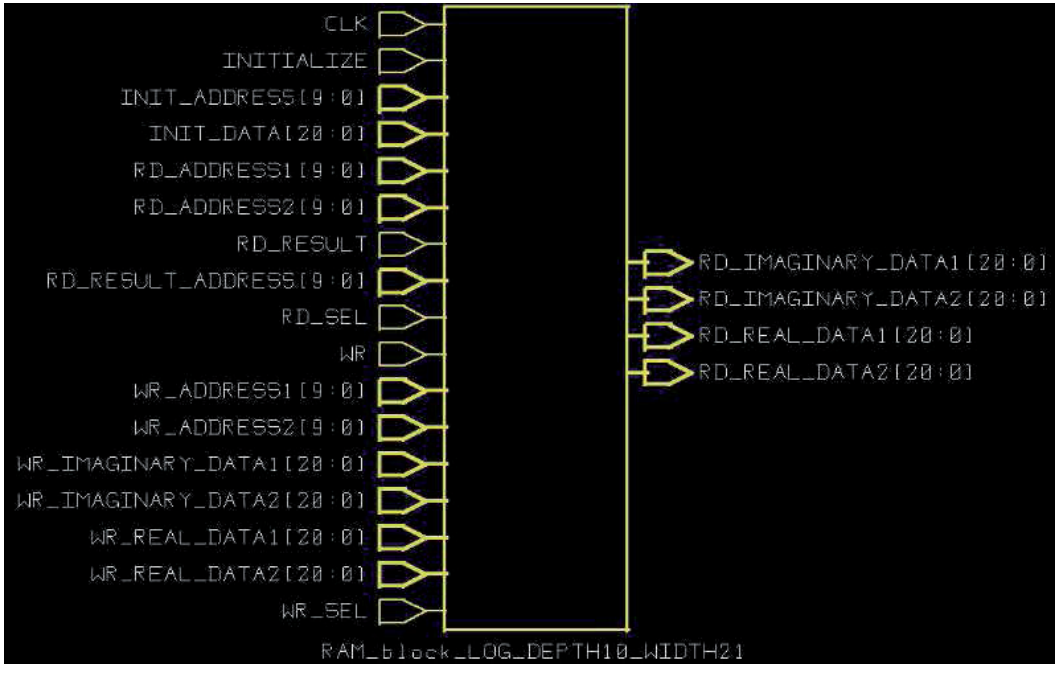


Figure 6.59 Top level implementation of the RAM modules

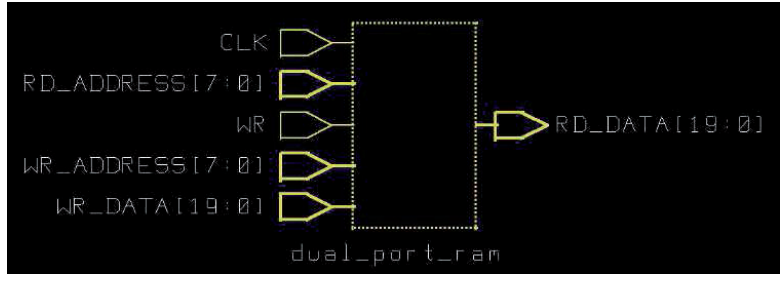


Figure 6.60 Top level implementation of the dual port RAM modules

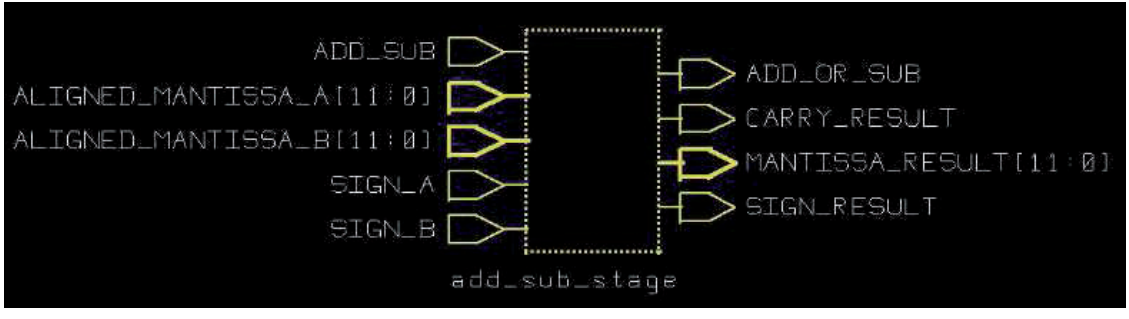


Figure 6.61 Top level implementation of the floating point adder/subtractor stage

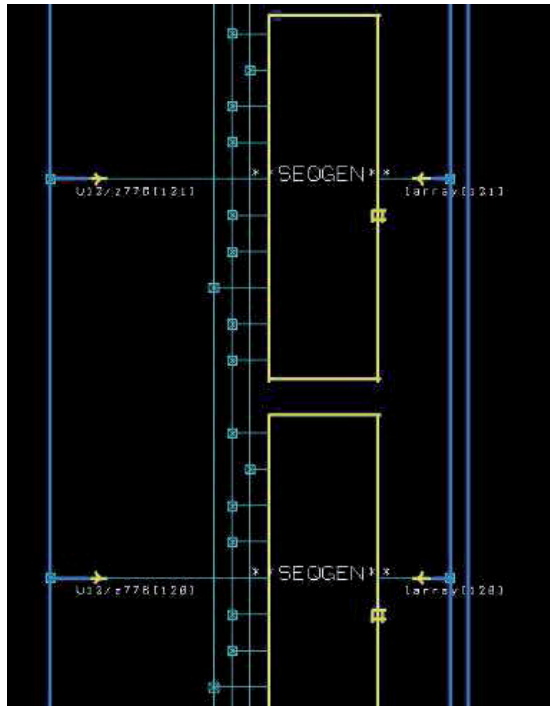


Figure 6.62 Internal implementation of the dual port RAM modules

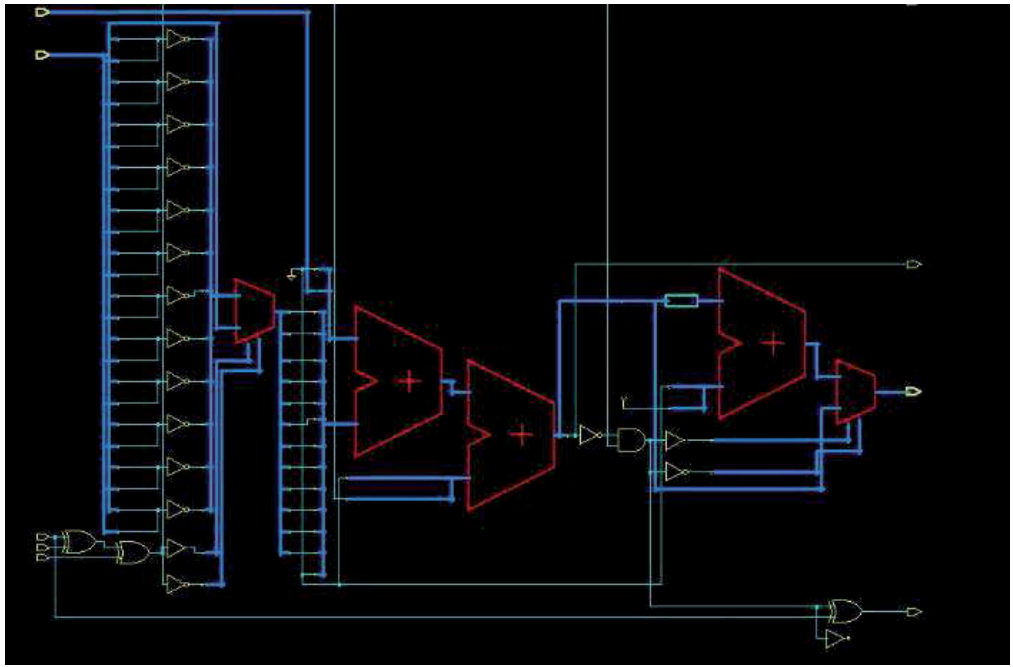


Figure 6.63 Internal implementation of the add_sub_stage

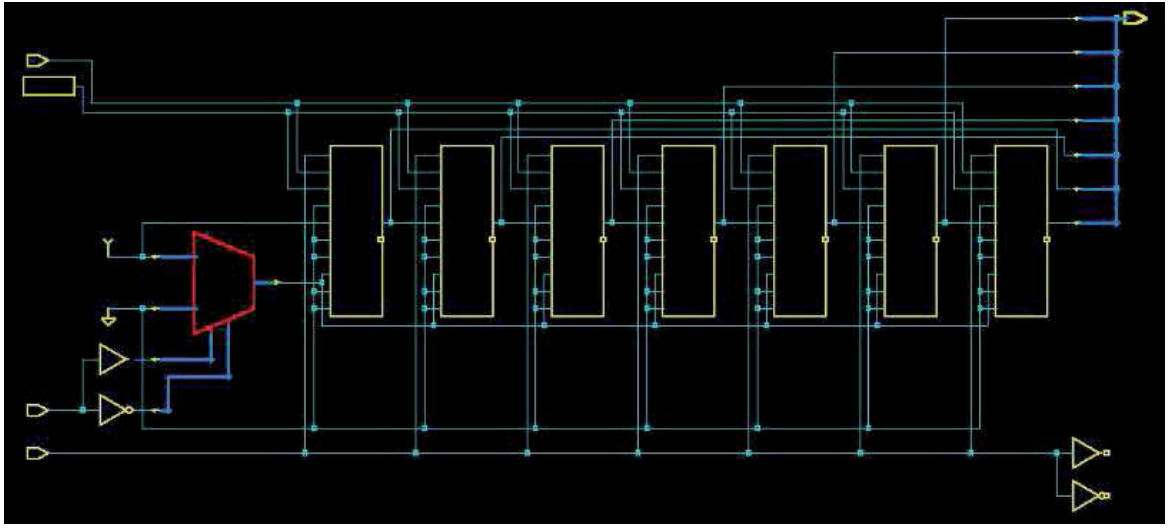


Figure 6.64 Internal implementation of the controller

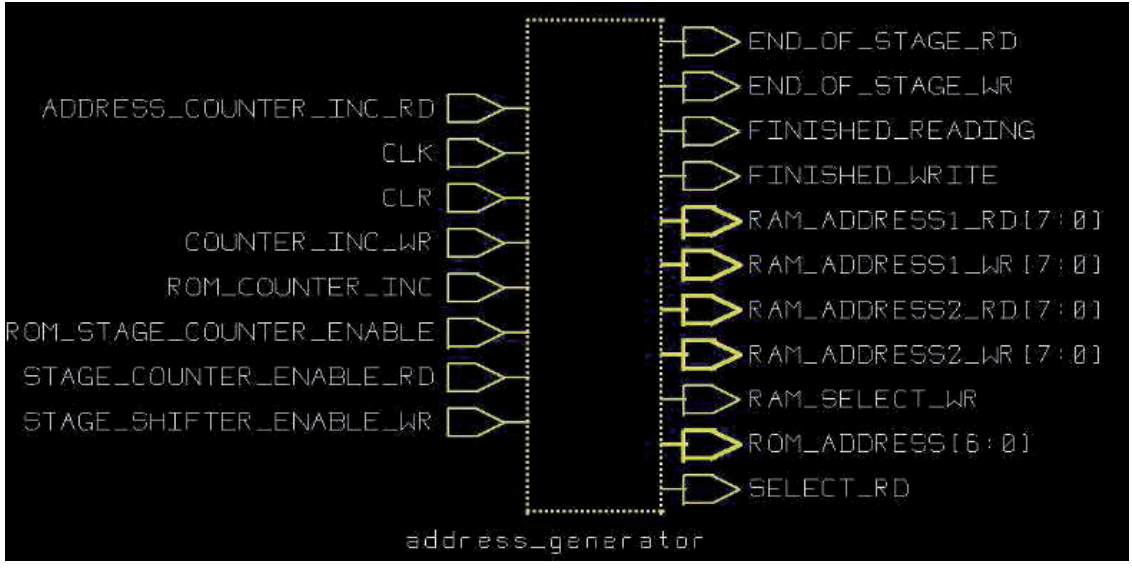


Figure 6.65 Top level implementation of the address generator

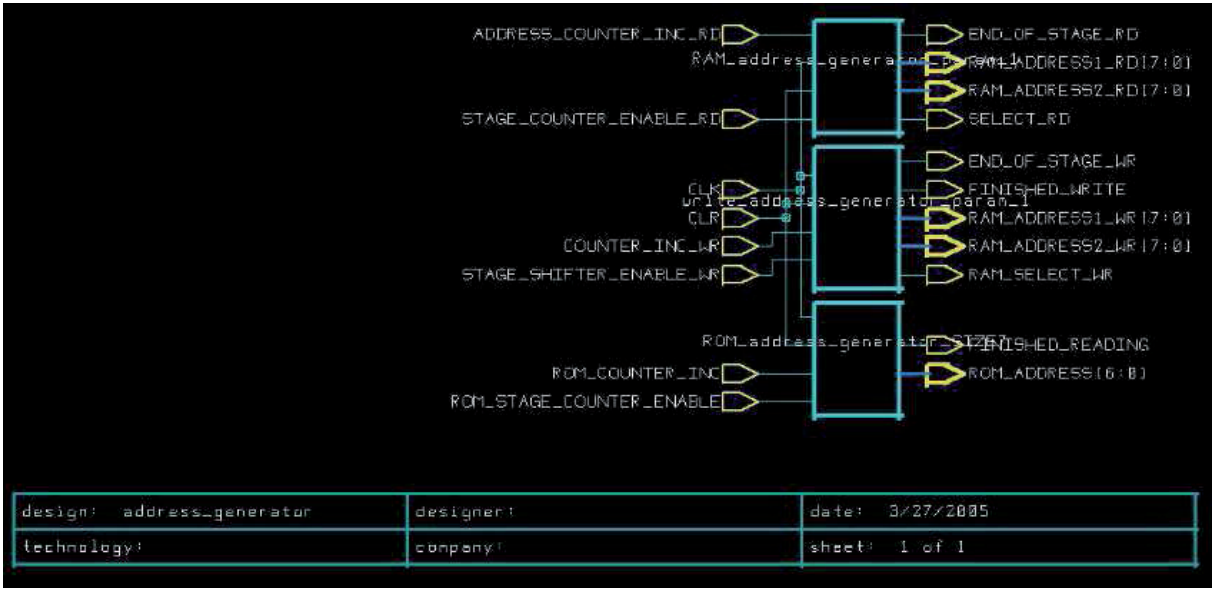


Figure 6.66 Internal implementation of the address generator

To conclude, after FPGA implementation and ASIC optimization and with considering available software and hardware resources, the proposed 1024- point radix-2 FPP-FFT processor was implemented and tested. Xilinx ISE software and CAD tools in synopsis provide the table of specifications shown in Table 6.17.

Table 6.17 Summary performance of the FPP-FFT processor

Parameters	**unit**	**specification**
Processor machine		Radix - 2
Calculation Type		Floating-point
Latency (μs)		22
Maximum Precision		32-bit
No. of input data		1024
Data rate (*Ms/s*)		25
Max. clock frequency	$f_{s\_\max}$	227 MHz
Signal to noise ratio	SNR	192 dB
Power consumption (Silrerra 0.18 μm library)	P_o	640 mW
Active core Area (Silrerra 0.18 μm library)	-	$2.32mm \times 2.32mm$
Resolution		$\leq 0.01\%$

6.4 RESEARCH CONTRIBUTION

To conclude, the research contribution of this research work is classified in three main categories of algorithm, high performance and technical aspect which are detailed as follow:

i) Algorithm:

A new algorithm for implementing FFT-based floating point that is capable of utilising high resolution signal processing, was designed and implemented. The standard floating-point FFT requires several chips for operation purposes whereas the proposed floating point FFT design requires only a single chip.

This particular characteristic makes the floating-point radix-2 FFT compatible to be downloaded in several version type of the FPGA board regardless of the memory size. In addition, with an extension of independently accessible memory bank it finds its application in high performance digital signal processing prototyping. Furthermore, to reduce hardware complexity, smart controller in cooperation with address generator unit is designed and implemented accordingly.

ii) High Performance

Its high performance could be seen in high speed and high resolution as well as low power and low latency which are achieved by decreasing the number of calculation (novel architecture), using fast floating-point adder/subtractor (270 MHz in 32-bit), eliminating unnecessary elements (achieved by parallel architecture) and designing smart controller in processor architecture. Additionally the floating point FFT also deserved to know as low latency parallel FFT processor. Furthermore the pipeline structure of the proposed FFT is presented and implemented to improve the speed in de-modulator system. The total estimated power consumption for the design after defining the constraints was less than 640 mW in 0.18 µm SILTERRA technologies and 1198 mW in 0.35 µm MIMOS technology. Moreover the number of adders and multipliers used in the proposed butterfly algorithm are 6 and 4 respectively for radix 2 calculation, which shows the reduction on the used elements considerably.

iii) Technical Aspects

The proposed FFT algorithm is integrated in parallel and pipeline architecture. This structure with the use of advanced floating-point adder/subtractor as the summation blocks will provide a speed of up to 227.74 MHz with resolution of less than 0.01%.

The computation complexity time of the proposed FFT follows the new equation of $(N/2\log_2 N)+11$, (N is the number of input data) which is much faster than the complex computation time for conventional N-point FFT structure $O(N\log_2 N)$ (Losing et al. 2005). The estimated chip die area has been optimized. The result shows that the active core area is 2.32×2.32 mm^2 in SILTERRA 0.18 µm technology library and 4.256×4.256 mm^2 in MIMOS 0.35 μm technology library.

Table 6.18 summerized the system improvement in terms of area, power , latency, resolution and clock frequency. Whilst Figure 6.67 shows the resolution improvement made by proposed architectre in compare with fixed point architecture.

Table 6.18 Persentage of system improvement

Description	Improvement (%)
Max. Clock frequency	20
Max. Clock frequency in floating point arithmetic unit	11
Power consumption	4
Resolution	10
Latency	50

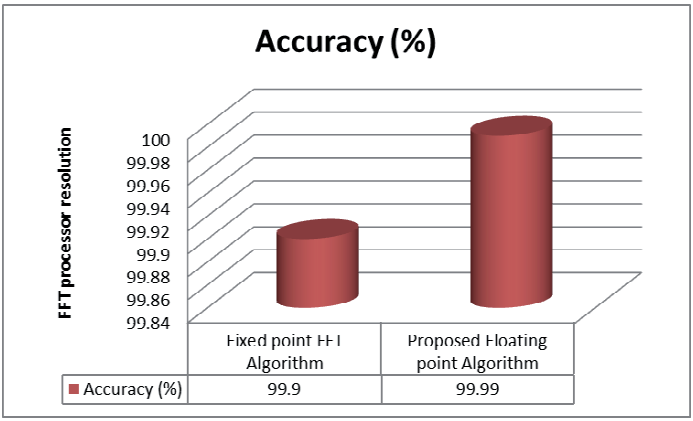


Figure 6.67 Fixed-point and proposed floating-point FFT resolution comparison

6.5 SUMMARY

In this chapter the proposed 1024-point radix-2 FFT processor was implemented. The design was launched with introducing 32-bit data single precision floating-point parallel pipeline architecture. Then it was followed by implementing the sub-components such as radix-2 butterfly and smart

controller. The implementation result of the proposed 1024-point radix-2 FPP FFT processor was provided accordingly. The advantages of this system were stated in Chapter V. Designing high speed floating-point arithmetic unit such as adder/subtractor (278 MHz), and multiplier (322 MHz), implementing smart controller to save area and increase system efficiency, design processor as single chip by implementing complex dual memory and providing pipeline and parallel architecture leads to present a high performance 1024-point radix-2 FPP FFT processor. In addition, the comparison was made in simulation of the FFT module and fixed-point conventional FFT implementation in order to obtain the processor resolution. From the comparison, Xilinx ISE synthesis report and Synopsis CAD tools, it was found that the proposed processor shows the accuracy less than 0.01% error in maximum clock frequency of 227 MHz. The latency for calculating 1024-point FFT is 22 μs. After FPGA implementation, the proposed processor was optimized in ASIC under SILTERRA 0.18 μm and MIMOS 0.35 μm technology libraries. The estimation power consumption was reported 640 mW in SILTERRA and 1.198 *W* in MIMOS technology library with sample rate of 25 ms/sec. The procedure was followed by defining the constraints and the netlist (gate level) to produce ASIC layout. The design compiler result shows the die size of 2.32×2.32 mm^2 in SILTERRA 0.18 μm technology and 4.256×4.256 mm^2 in MIMOS 0.35 μm technology. From the given specification it was found that the proposed 1024-point radix-2 FPP FFT processor is suitable for DSP application in order to achieve high-speed and high-resolution performance.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

7.1 CONCLUSIONS

The 1024-point radix-2 FPP-FFT processor is designed and implemented as a stand-alone core. This core satisfies the FFT size requirement of the 1024-points for a maximum clock frequency of 227 MHz and the resolution of less than ±0.01% error for the DSP application. The overall floating-point parallel FFT architecture is executed with radix-2 engine. This reference design has the following features:

i) VHDL-based design.

ii) MATLAB bit-accurate simulation with implementation precision of less than ±0.01% error.

iii) Target clock rate at 227 MHz.

iv) Streaming 1024-point FFT as serial input/output data.

v) Floating-point I/O representation to maintain precision.

vi) 32-bit wide I/O data capable of switching to 21-bit I/O.

vii) FFT specifiable on a per-block basis.

viii) Single butterfly architecture that communicate with a high-performance controller.

The research contribution of the proposed radix-2 FFT processor was discussed. The advantages of the 1024 point radix-2 FPP-FFT processor in order to improve speed, resolution, area, power consumption and latency are:

- A novel algorithm for implementation 1024 point radix-2 FFT-based floating-point that is capable of utilising high-resolution signal processing ($\leq$ 0.01%). This improvement obtained 10 times higher resolution in comparison with fixed-point algorithm. (Table 6.14; Figure 6.68).

- The unique combination of the parallel and pipeline structure in the internal architecture of proposed radix-2 butterfly engine processor. In the proposed engine processor, there is only a single butterfly (6 adders and 4 multipliers) that it is resulted the reduction of the components and speed improvement by 20% (227 MHz). (Figure 6.53)

- Design and implementation of intelligent controller to compile the overall FFT calculations within a single parallel pipeline butterfly to reduce the hardware complexity and result shows low power consumption (640 mW, 4% improvement) and die size.

- Design and implementation of novel high-speed floating-point arithmetic unit such as adder/subtractor (278 MHz), and multiplier (322 MHz). This floating-point arithmetic unit shows on improvement of 11% in maximum clock frequency.

- The memory relaxed design due to utilization of the internal complex dual RAM for stand-alone single-chip radix-2 FFT processor, regardless of the different type of FPGA board.

- The reduction of the hardware complexity and power consumption by the storing of input data and output data in a single complex RAM only.

- The reduction of the hardware complexity in the ROM stage which stores only half the size of the twiddle factor ($N/2$ factor of W^{kn}).

- The significant improvement of the speed through a combined individual component with the same latency and its connection to the sequential stage as pipeline architecture.

- Through the fact that the new equation of proposed algorithm was derived and designed for latency in the proposed radix-2 FFT processor which produce the data exactly after $(N/2\log_2^N)+11$ clock cycles. This equation shows 50% latency improvement compared with the previous conventional fixed point FFT type calculation.

Based-on above sorted advantages, the proposed 1024-point radix-2 FPP-FFT processor specifications from the Xilinx ISE software and CAD tools were summarised (Table 6.18). To conclude, table of improvement for proposed 1024 point FPP-FFT processor is given in Table 6.

7.2 FUTURE WORK

It is suggested to fabricate this research project in order to design floating-point FFT processor and integrate this chip to apply in condition monitoring (DSP application) and communication system. Although the floating-point FFT processor is a high-speed processor in comparison with similar research works, the maximum operation clock frequency can be improved by the application of the advance fast adder such as the prefix adder in floating-point style for future research and development. However, with the advent of Nano-technology, the design also can be fabricated to Nano scaled in order to achieve the higher throughput for improving the system efficiency. Along that, the wire engineering in Nano scaling will lead to provide better fabrication in Nano technology.

REFERENCES

Adams, J.W. 1987. A subsequence approach to interpolation using the FFT. *IEEE Trans CAS* 37: 623-625.

Ai, B., Jian Hua Ge, Yong Wang, Shi Yong Yang, Pei Liu & Gang Liu. 2004. Frequency offset estimation for OFDM in wireless communications. *IEEE Transaction on Consumer Electronics* 50(1):73-77. DOI: 10.1109/TCE.2004.1277843.

Atarashi, H., & Sawahashi, M. 2001. Variable spreading factor orthogonal frequency and code division multiplexing (VSF-OFCDM). *Proceeding of International workshop on Multicarrier Spread-Spectrum and related Topics (MC-SS2001),* pp. 112-122.

Antoine B., Abche, Aldo Maalouf, Rafic Ayoubi, Elie Karam & A. M. Alameddine. 2007. An FPGA implementation of a high resolution phase shift beam former. *IEEE conference on signal and communications (ICSPC 2007),* pp 1319-1322.

Baas, B. M. 1999. A low power high performance 1024 Point FFT processor. *IEEE Journal of Solid-State Circuits* 34(3):380-387.

Bergland, G. D. 1969. A guided tour of the fast fourier transform. *IEEE Spectrum* conference, pp. 41-52.

Beukelman, P. C. R. & Bierens, L. H. J. 1999. Fastest floating-point single-chip FFT processor. *IEEE/ProRISC99, STW/SAFE99*, pp: 649-654.

Bever, M., Feder, Ch., Hehmann, D., Schottmuller, C & Stuttgen, H. 1990. Communication support for multimedia applications. *IET Conference on Integrated Broadband Services and Network,* pp. 115-120.

Biswas, B., Das, S., Purkait, P., Mandal, M. S. & Mitra, D. 2009. Current harmonics analysis of inverter-Fed induction motor drive system under fault conditions. *International Multi-Conference of Engineers and Computer Scientist IMECS*, 2: 1-5.

Burgess, N., 2004. Prenormalization rounding in IEEE floating-point operations using a flagged prefix adder. *IEEE J. Very Large Scale Integrat*. Syst. 13: 266-277. DOI: 10.1109/TVLSI.2004.840764

Byung G. J. & Myung H. S.2005. New continues-flow mixed-radix (CFMR) FFT processor using novel in-place strategy. *IEEE Transactions on Circuits and Systems* 52(5):911-919.

Chi Huang, Xinyu Wu, Jinmei Lai, Chengshou Sun and Gang Li. 2005. A design of high speed double precision floating point adder using macro modules. *Design Automation Conference, Proceedings of the ASP-DAC*, Asia and South Pacific 2:D11-D12. DOI: 10.1109/ASPDAC.

Chi Wai Yu, Smith, A.M., Luk, W. Leong, P.H.W. & Wilton, S. J. E. 2008. Optimizing coarse-grained units in floating point hybrid FPGA. *IEEE International*

Conference on Field-Programmable Technology. *FPT*, pp. 57-64. DOI: 10.1109/FPT.2008.4762366.

Chin-Peng Fan, Mau-Shih Lee & Guo-An Su. 2006. A low multiplier and multiplication costs 256-point FFT implementation with simplified Radix-16 SDF architecture. *IEEE Conference on Circuits and Systems APCCAS,* pp. 1935-1938. DOI: 10.1109/APCCAS.2006.342239.

Chin-Teng Lin, Yuan-Chu Yu & Lan-Da Van. 2006. A low-power 64-point FFT/IFFT design for IEEE 802.11a WLAN agpplication. *IEEE Conference on Circuits and Systems ISCAS,* pp. 4523-4526. DOI: 10.1109/ISCAS.2006.1693635.

Ciletti, D. M. 2003. *Advanced Digital Design with the Verilog HDL*. Department of Electrical and Computer Engineering University of Colorado at Colorado Springs: Prentice Hall.

Cooley, J. W. & Tukey, J. W. 1965. An algorithm for the machine computation of complex fourier Series. *Mathematics of computation Journal* 19:297-301.

Couasnon T. D., Monnier R. 1994. OFDM for Digital TV Broadcasting. *Elsevier Signal Processing* 39: 0165-1684. DOI: 1-32. ISSN.

Cummings, M. & Haruyama, S. 1999. FPGA in the software Radio. *IEEE Communication Magazine* 37(2):108-112.

Dabbagh-Sadeghpour, K. & Eshghi, M. 2003. A self-timed, pipelined floating point FFT processor architecture. *IEEE Conference on Signals, Circuits and Systems*, 1: 33-36. ISBN: 0-7803-7979-9.

Denzil, F., Vijai Raj & Narsimhan D. 1993. Asynthesis tool based design of a 111 MHz CMOS floating point adder with built in test ability. *IEEE ASIC Conference and Exhibit*, pp. 412 – 415.

Dick, C. & Harris, F. J. 1999. Configurable logic for digital communications: some signal processing perspectives. *IEEE Communication Magazine* 37(8):112-117.

Dovel, G. 1989. *FFT analyzers make spectrum analysis a snap*. Spectrum analysis. EDN. pp. 149-155.

Duhamel, P. & Hollmann, H. 1984. Split Radix FFT algorithm. *IEEE Electronics Letters* 20:14-16. DOI: 10.1049/el:19840012.

Duhamel, P., Piron, B. & Etcheto, J. M. 1988.On computing the inverse DFT. *IEEE Trans. ACoust, Speech on Signal Processing* 36(2):285-286. DOI: 10.1109./29.1519.

Enis C. A., Omer N. G. & Yardimci, Y. 1997. Equiripple FIR filter design by the FFT algorithm. *IEEE magazine of Signal processing 1053-5888/97*, pp. 60-64.

ETSI TR 101 475. 2000. *Broadband radio access networks (BRAN); HIPERLAN Type 2; Physical (PHY) Layer*. ETSI BRAN.

Farhang-Boroujeny, B. & Gazor, S. 1994. Generalized sliding FFT and its applications to implementation of block LMS adaptive filters. *IEEE Trans. SP* 42: 532-538.

Friedmann, A. 2007. *Understanding OFDMA, the interface for 4G wireless.* Courtesy of Mobile Handset Design Line: *Texas Instruments (TI) Software Manager, Communications Infrastructure Group*.

Frigo, M. & Johnson, S. G. 1998. FFTW: an adaptive software architecture for the FFT. *IEEE Conference on Acoustics, speech and signal processing,* 3: 1381-1384.

Gentleman, W. M. & Sande, G.1996. Fast Fourier Transform for Fun and Profit. *In Proceedings 1966 Fall Joint Computer Conference AFIPS 29,* pp 563-578.

Gieras, J. F. & Wing, M. 2002. *Permanent magnet motor technology design and application*. Second Edition, Revised and Expanded: Marcel Dekker Inc. ISBN: 0-8247-0739-7.

Gijung Yang & Yunho Jung. 2010. Scalable FFT processor for MIMO-OFDM based SDR systems. *IEEE International Symposium on Wireless Pervasive Computing (ISWPC)*, pp. 517-521.

Gold, B. & Radar, C. 1969. *Digital processing of Signals*", McGraw-Hill, New York.

He, S. & Torkelson, M. 1998.Design and Implementation of 1024-point Pipeline FFT Processor.IEEE Conference on Custom Integrated Circuits. pp. 131-134. DOI: 10.1109/CICC.1998.694922.

Hemmert, K. S. & Underwood, K. D. 2005. An Analysis of The Double-Precision Floating-point FFT on FPGAs, *13th IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 171-180.

Hoff J., 1993. A Full Custom High Speed Floating Point Adder. *Fermi National Accelerator Lab.*

Hojin Kee, Bhattacharyya, S. S., Petersen, N. & Kornerup, J. 2009.Resources Efficient Acceleraion of 2-Dimentional FFT Computation on FPGA.*IEEE Conference on Distributed Smart Cameras*.ICDSC. pp. 1-8. DOI: 10.1109/ICDSC.2009.5289356.

Huang, C., X. Wu, J. Lai, C. Sun and G. Li. 2005. A design of high-speed double precision floating point-adder using macro modules. *Proceedings of the Asia and South Pacific Design Automation Conference*, pp. 18-21.

Hui C. W., Tiong Jiu Ding, McCanny J. V. & Woods F. R. 1996. A New FFT architecture and chip design for motion compensation based on phase correlation. *IEEE Conference Application Specific Systems, Architectures and Processor,* pp. 83-92.

IEEE Std. 802.11a. 1999. *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-Speed Physycal Layer Extension in the 5-GHz Band*. New York: IEEE Syandard Publication.

IEEE Std. 1985. *For binary Floating-Point Arithmetic*. New York: IEEE Standard 754-1985. IEEE inc, pp1-17.

Ifeachor, E C. & Jervis, B. V. 2002. *Digital Digital Signal Processing: A Practical Approch*. Second Editiond: Prantice Hall.

Jaehee Cho, Namshin Cho, Keukjoon Bang, Myunghee Park, Heeyoung Jun, Hyncheol Park&Daesik Hong. 2001. PC-based receiver for Eureka-147 digital audio broadcasting. *IEEE transactions on Broadcasting* 47(2): 95-102. *DOI: 10.1109/11.948262.*

Jain, A.K. 1989. *Fundamentals of digital image processing*. Filtering LPF, BPF, HPF. Generalized spectrum and homomorphic filtering. Englewood Cliffs, NJ: Prentice Hall.

Jain, V.K., Collins, W.L. & Davis, D.C. 1979. High accuracy analog measurements viainterpolated FFT. *IEEE Trans. IAM*, IM-28: 113-122.

Jen-Chin Kuo, Ching-Hua Wen & An-Yeu-Wu. 2003. Implementation of programmable 64-2048-point FFT/IFFT processor for OFDM-based communication system. *IEEE Conference on Circuit and Systems, ISCAS*, 2:121-124.

Jeong Ho Kim, ChoonSikYim. 1995. ADSL/HDSL technologies and applications in VOD and multimedia services. *IET Conference on broadcasting convention,* pp. 346-350.

Jihad Qaddour. 2006. High peak to average ratio solution in OFDM of 4G mobile systems. *International Conference on Communications and Mobile Computing,* pp. 337-342. ISBN:1-59593-306-9

Jont B. A. & Rabinner, R. 1977. A unified approach to short time Fourier analysis and synthesis. *IEEE Proceedings of Analysis and Synthesis* 65(11):1558-1564.

Ke Liu, HuarongZheng, Jianing Su, Bo Shen & Hao Min. 2005. A novel synchronization scheme in HDTV system with adaptive detection and low implementation complexity. *IEEE International Conference on ASIC, ASICON,* 1: 270-273.

Kim, N. & Yoon H. 2005. Handoff procedure for seamless service in IP and OFDM based 4G mobile systems. *IEICE Transactions Electron*. E88:C(12).

Kuo, S. M. & Woon-SengGan. 2005. *Digital Signal Processors, Architecture, Implementations and Applications*. Pearson Education International: prentice Hall. ISBN: 0131277669.

Le Flock, B., Alard, M. & Berrou, C.1995. Coded orthogonal frequency division multiplex. *Proc. IEEE*. 83(6): 982-996.

Lefevre, M. & Okrah, P. 2001. Fundamental changes required in modulation and signal processing for 4G. *Communications Systems Design Magazine.*

Li, W & Wanhammar, L. 1999. A pipeline FFT processor. *IEEE Conference on Signal Processing Systems*, pp. 654-662. DOI: 10.1109/SIPS.1999.822372.

Lihong Jia, Yong Hong Gao, Jouni Isoaho & Hannu Tenhunen. 1998. A new VLSI-oriented FFT algorithm and Implementation. *IEEE Conference on ASIC,* pp. 337-341. DOI: 10.1109/ASIC.1998.723029.

Lo Sing Cheng, Miri, A. & Tet Hinb Yeap. 2005. Efficient FPGA implementation of FFT based multipliers. *IEEE Conference Electrical and Computer Engineering,* pp. 1300-1303.

Long Chen, Ziang Hu, Jumin Lin, Gao G. R. 2007. Optimizing the fast Fourier transform on a multi-core architecture. *IEEE International Parallel and Distributed Processing Symposium,* pp. 1-8.

Mathwork. 2010. *Sim Power System, Simulating Variable Speed Motor Control*. The Math works, accelerating the pace of engineering and science: Mathwork Inc.

Mazlaini, Y. 2006. Practical packet detection and symbol timing synchronization scheme for packet OFDM system. *IEEE Conference on RF and Microwave.* pp. 421-425. DOI:10.1109/RFM.2006.331118.

Melander, J., Widhe, T. & Wanhammar, L. 1996. Design of an 128-Point FFT processor for OFDM applications. *IEEE Conference on Electronics, Circuits and Systems*, *ICECS96*. 2: 828-831. DOI: 10.1109/ICECS.1996.584499.

Miyamoto, N., Karnan, L., Kazuyuki Maruo, Koji Kotani & Tadahiro Ohmi. 2004. A small area high performance 512-point 2-dimensional FFT single-chip processor. *IEEE Conference on solid-state circuits conference,* pp. 603-606. DOI: 10.1109/ESSCIRC.2003.1257207

Morton, S. V., Appleton, S.S. & Liebelt, M. J. 1994. An event controlled reconfigurable multi-chip FFT. *IEEE Conference on Advanced research in Asynchronous Circuits and Systems*. pp. 144-153. DOI: 10.1109/ASYNC. 1994.656304.

Narasimhan, D., Fernandes, D., Raj, V. K., Dorenbosch, J., Bowden, M., & Kapoor., V. S. 1993. A 100 MHz FPGA based floating point adder. *IEEE custom integrated circuits conference,* pp. 3.1.1-3.1.4. DOI: 10.1109/CICC.1993.590348.

Nee, R. V. & Prasad, R. 2000. *OFDM for Wireless Multimedia Communications.* First Edition. Boston: MA Artech House. ISBN:0890065306.

Ochi, H. 2008. RTL design of parallel FFT with block floating point arithmetic. *IEEE Conference on soft computing in industrial applications.* Japan, pp. 273- 276.

Ojanpera, T., & Prasad. 1998. *Wideband CDMA for Third Generation Mobile Communications*. Norwood, MA: Artech House.

Oppenheim, A. & Schafer R. 1989. *Discreet-Time Signal Processing*. Englewood Cliffs, NJ: Prentice Hall.

Paiement, R. V. 1994. Evaluation of single carrier and multicarrier modulation techniques for digital ATV terrestrial broadcasting. *CRC Report No. CRC-RP-004.Canada.*

Rabenstein, R. & Zayati, A. 1999. A direct method to computational acoustics. *IEEE International Conference on Acoustics, Speech and Signal Processing,* 1:69-72. DOI: 10.1109/ICASSP.1996.540292.

Rainbolt, B. J. & Miller, S. L. 1998. The necessity for and use of CDMA transmitter filtering in overlay systems. *IEEE Journal on Selected areas in Communications,* pp: 1756-1764. DOI: 10.1109/49.737644.

Schafer, R.W. & Rabiner, L.R. 1973. A digital signal processing approach to interpolation. *IEEE Journal* 61(6):692-702.

Schimmel, S. M., Muller, M. F. & Dillier, N. 2009. A fast and accurate "shoebox" room acoustics simulator. *IEEE Conference on Acoustics, Speech and Signal Processing,* pp. 241-244. DOI: 10.1109/ICASSP.2009.4959565.

Sen M. Kuo & Woon-Seng Gan. 2005. *Digital Signal Processors, Architecture, Implementations, and applications*. International Edition, New York: Pearson prentice Hall. ISBN 0-13-127766-9.

Shi Chen, Venkatatesan R. & Gillard, P. 2008. Implementation of vector floating-point processing unit on FPGAs for high performance computing. *IEEE Conferrence on Electrical and Computer Engineering*. CCECE 2008, pp. 881-886. Canada. DOI: 10.1109/CCECE.2008.4564662.

Shiqun Zheng & Dunshan Yu. 2004. Design and implementation of a parallel real-time FFT processor. *7th IEEE conference on Solid-State and Integrated Circuits Technology*, 3:65-168.

Shyue-Kung Lu, Jen-Sheng Shih & Shih-Chang Hhuang. 2005. Design-for-testability and fault-tolerant techniques for FFT processors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 13(6):732-741.

Smith, J.O. 2007. *Mathematices of the Discrete Fourier Transform (DFT) with Audio Applications*. Second Edition:W3K Publishing. ISBN 978-0-9745607-4-8.

Smith, T., Smith, M.R. Nichols, S.T. 1990. Efficient sinc function interpolation techniques for center padded data. *IEEE Trans. ASSP* 38:1512-1517.

Soliman, S. & Wheatley, C. 1995.Frequency coordination between CDMA and non-CDMA systems. *IEEE Conference on Technologies for wireless Applications Digest,* pp.79-87. DOI: 10.1109/MTTTWA.1995.512331.

Son, B. S., Jo, B. G., Sunwoo, M. H. & Yong Serk Kim. 2002. A high speed FFT processor for OFDM systems. *IEEE International Conference on Circuits and System,* 3:281-284. DOI: 10.1109/ISCAS.2002.1010215.

Stearn, S.D. & David, R.A. 1988. *Signal Processing Algorithms*. Decimation and Interpolation Routines. Convolution and correlation using FFT. FFT convolution and correlation. Englewood Cliffs, NJ: Prentice-Hall.

Teymourzadeh, R, BurhanuddinYeop Majlis, Jimmy Mok vee Hong, Masuri Bin Othman. 2009. VLSI implementation of high resolution high speed low latency pipeline floating point adder/subtractor for FFT application. *Nano Technology Conference Energy Health & Environment*, pp. 327-331.

Thompson, J., Nandini Karra & Schulte, M. J. 2004. A 64-bit Decimal floating-point adder. *IEEE Computer Society Annual on VLSI Systes Design*. DOI: 0-7695-2097-9/04. 2004.

Thulasiram R. K. & Thulasiraman P. 2003. Performance evaluation of a multithreaded fast Fourier transform algorithm for derivative pricing. *The Journal of Supercomputing* 26(1):43-58.

Thong, T. & Liu, B. 1977.Accumulation of round off errors in floating point FFT. *IEEE Journal on Circuits and Systems* 24(3):132-143.

Tseng, B. D., Jullien G. A. & Miller W. C. 1979. Implementation of FFT structures using residue number system. *IEEE Journal on Computers* C-28(11):831-845.

Wiegandt, D. A. & Nassar, C. R. 2002. High-throughput, high-performance OFDM via pseudo-orthogonal carrier interferometer type 2. *IEEE International Symposium on Wireless Personal Multimedia Communications,* 2:729-733. DOI: 10.1109/TCOMM.2003.814196.

Wu Y. & Zou, Y. W. 1995. Orthogonal frequency division multiplexing: a multicarrier modulation scheme. *IEEE Transaction on Consumer Electronics* 41(3):392-399.

Xenoulis, G., Psarakis, M., Gizopoulos D. & Paschalis, A. 2005. Test generation methodology for high-speed floating point adders. *IEEE International Conference on On-line testing symposium*, *IOLTS2005,* pp. 227-232. DOI: 10.1109/IOLTS.2005.67.

Xilinx Logic core, 2009. Fast Fourier transform. *Versian 7.0.DS260 product specification*. pp. 1-64.

Xin Xiao, Oruklu, E. & Saniie, J. 2009.Fast memory addressing scheme for Radix-4 FFT implementation. *IEEE International Conference on Electro/Information Technolog,* pp. 437-440. DOI: 10.1109/EIT.2009.5189656.

Yeo, D. Zhongjun Wang, Bin Zhao & Yajuan He. 2002. Low Power Implementation of FFT/IFFT processor for IEEE 802.11a Wireless LAN Transceiver. *IEEE Conference on Communication System* . *ICCS2002*, 1:250-254.

Yong Jun Peng. 2003. A parallel architecture for VLSI implementation of FFT processor. *IEEE Conference on ASIC Proceedings,* 2:748-751.

Yu-Wei Lin, & Chen-Yi Lee. 2007. Design of an FFT/IFFT processor for MIMO OFDM systems. *IEEE Transaction on Circuit and Systems* 54(4):807-815. DOI: 10.1109/TCSI.2006.888664.

Zainal, M. S., Yoshizawa, S. & Miyanaga, Y. 2008. Low power FFT design for wireless communication systems. *IEEE Conference on Intelligent Signal Processing and Communications Systems*, pp. 1-4. DOI: 10.1109/ISPACS.2009.4806724.

Zhang, Q. & Meng, N. 2009. A Low area pipelined FFT processor for OFDM-Based systems. *IEEE Conference on Wireless Communications, Networking and Mobile Computing, WiCom*, pp. 1-4. DOI: 10.1109/WICOM.2009.5303332.

Zhiqiang Wu & Nassar C. R. 2005. Narrowband interface rejection in OFDM via carrier interferometry spreading codes. *IEEE Transaction on wireless communications* 4(4):1491-1505.