



**INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH
TECHNOLOGY**

IMPLEMENTATION OF A FLEXIBLE AND SYNTHESIZABLE FFT PROCESSOR

Vishnu Singh*

* Iv-Year, Ece Student, Vidya Jyothi Institute Of Technology, Hyderabad

ABSTRACT

The Fast Fourier Transform (FFT) is one of the rudimentary operations in field of digital signal and image processing. Some of the very vital applications of the fast Fourier transform include Signal analysis, Sound filtering, Data compression, Partial differential equations, Multiplication of large integers, Image filtering etc. Fast Fourier transform (FFT) is an efficient implementation of the discrete Fourier transform (DFT). This paper concentrates on the development of the Fast Fourier Transform (FFT), based on Decimation-In- Time (DIT) domain, Radix-2 algorithm, this paper uses VERILOG as a design entity. The input of Fast Fourier transform has been given by a keyboard using a test bench and output has been displayed using the waveforms on the Xilinx Design Suite 10.1 and Modelsim 6.4b and synthesis results in Xilinx show that the computation for calculating the 32-point Fast Fourier transform is efficient in terms of speed.

KEYWORDS: FFT, Modelsim 6.4b, VERILOG, DIT.

INTRODUCTION

INTRODUCTION TO TRANSFORM

A signal can be either continuous or discrete, and it can be either periodic or aperiodic. The combination of these two features generates the four categories, described below

- Aperiodic-Continuous

This includes, for example, decaying exponentials and the Gaussian curve. These signals extend to both positive and negative infinity without repeating in a periodic pattern. The Fourier Transform for this type of signal is simply called the Fourier Transform.

- Periodic-Continuous





Here the examples include: sine waves, square waves, and any waveform that repeats itself in a regular pattern from negative to positive infinity. This version of the Fourier transform is called the Fourier series.

- Aperiodic-Discrete

These signals are only defined at discrete points between positive and negative infinity, and do not repeat themselves in a periodic fashion. This type of Fourier transform is called the Discrete Time Fourier Transform.

- Periodic-Discrete

These are discrete signals that repeat themselves in a periodic fashion from negative to positive infinity. This class of Fourier Transform is sometimes called the Discrete Fourier Series, but is most often called the Discrete Fourier Transform.

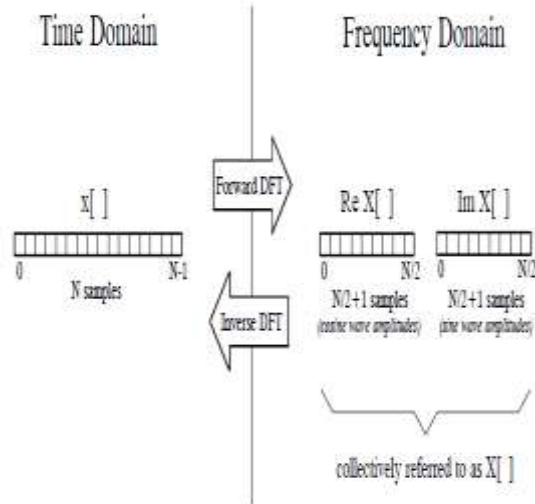
Type of Transform	Example Signal
Fourier Transform <i>signals that are continuous and aperiodic</i>	
Fourier Series <i>signals that are continuous and periodic</i>	
Discrete Time Fourier Transform <i>signals that are discrete and aperiodic</i>	
Discrete Fourier Transform <i>signals that are discrete and periodic</i>	

Each of the four Fourier Transforms can be subdivided into real and complex versions. The real version is the simplest, using ordinary numbers and algebra for the synthesis and decomposition. The complex versions of the four Fourier transforms are immensely more complicated, requiring the use of complex numbers. These are numbers such as: $3 + 4j$, where j is equal to $\sqrt{-1}$ (electrical engineers use the variable j , while mathematicians use the variable, i). Complex mathematics can quickly become overwhelming, even to those that specialize in DSP. In fact, a primary goal of this book is to present the fundamentals of DSP without the use of complex math, allowing the material to be understood by a wider range of scientists and engineers. The complex Fourier transforms are the realm of those that specialize in DSP, and are willing to sink to their necks in the swamp of mathematics.

The mathematical term: transform, is extensively used in Digital Signal Processing, such as: Fourier transform, Laplace transform, Z transform, Hilbert transform, Discrete Cosine transform, etc. A function is an algorithm or procedure that changes one value into another value. For example, $y = 2x + 1$ is a function. You pick some value for x , plug it into the equation, and out pops a value for y . Functions can also change several values into a single value, such as: $y = 2a + 3b + 4c$, where a , b , and c are changed into y .

Transforms are a direct extension of this, allowing both the input and output to have multiple values.

NOTATION AND FORMAT OF THE REAL DFT



As shown in Figure, the discrete Fourier transform changes an N point input signal into two $N/2+1$ point output signals. The input signal contains the signal being decomposed, while the two output signals contain the amplitudes of the component sine and cosine waves (scaled in a way we will discuss shortly). The input signal is said to be in the time domain. This is because the most common type of signal entering the DFT is composed of samples taken at regular intervals of time. Of course, any kind of sampled data can be fed into the DFT, regardless of how it was acquired. When you see the term "time domain" in Fourier analysis, it may actually refer to samples taken over time, or it might be a general reference to any discrete signal that is being decomposed. The term frequency domain is used to describe the amplitudes of the sine and cosine waves (including the special scaling we promised to explain).

The frequency domain contains exactly the same information as the time domain, just in a different form. If you know one domain, you can calculate the other. Given the time domain signal, the process of calculating the frequency domain is called decomposition, analysis, the forward DFT, or simply, the DFT. If you know the frequency domain, calculation of the time domain is called synthesis, or the inverse DFT. Both synthesis and analysis can be represented in equation form and computer algorithms.

The number of samples in the time domain is usually represented by the variable N . While N can be any positive integer, a power of two is usually chosen, i.e., 128, 256, 512, 1024, etc. There are two reasons for this. First, digital data storage uses binary addressing, making powers of two a natural signal length. Second, the most efficient algorithm for calculating the DFT, the Fast Fourier Transform (FFT), usually operates with N that is a power of two. Typically, N is selected between 32 and 4096. In most cases, the samples run from 0 to $N-1$, rather than 1 to N .

DFT BASIS FUNCTIONS

The sine and cosine waves used in the DFT are commonly called the DFT basis functions. In other words, the output of the DFT is a set of numbers that represent amplitudes. The basic functions are a set of sine and cosine waves with unity amplitude. If you assign each amplitude (the frequency domain) to the proper sine or cosine wave (the basic functions), the result is a set of scaled sine and cosine waves that can be added to form the time domain signal.

The DFT basis functions are generated from the equations:

$$c_k[i] = \cos(2\pi ki/N)$$

$$s_k[i] = \sin(2\pi ki/N)$$

where: c is the cosine wave for the amplitude held in $\text{Re}\bar{X}[k]$, and $s[k]$ is the sine wave for the amplitude held in $\text{Im}\bar{X}[k]$.

SYNTHESIS, CALCULATING THE INVERSE DFT

Pulling together everything said so far, we can write the synthesis equation:

$$x[i] = \sum_{k=0}^{N/2} \text{Re}\bar{X}[k] \cos(2\pi ki/N) + \sum_{k=0}^{N/2} \text{Im}\bar{X}[k] \sin(2\pi ki/N)$$

In words, any N point signal, $x[i]$, can be created by adding $N/2 + 1$ cosine waves and $N/2 - 1$ sine waves. The amplitudes of the cosine and sine waves are held in the arrays $\text{Re}\bar{X}[k]$ and $\text{Im}\bar{X}[k]$, respectively. The synthesis equation multiplies these amplitudes by the basis functions to create a set of scaled sine and cosine waves. Adding the scaled sine and cosine waves produces the time domain signal, $x[i]$. In above Eq 1, the arrays are called $\text{Im}\bar{X}[k]$ and $\text{Re}\bar{X}[k]$ rather than $\text{Im}X[k]$ and $\text{Re}X[k]$. This is because the amplitudes needed for synthesis are slightly different from the frequency domain of a signal (denoted by: $\text{Im}X[k]$ and $\text{Re}X[k]$). This is the scaling factor issue we referred to earlier. Although the conversion is only a simple normalization, it is a common bug in computer programs. In equation form, the conversion between the two is given by:

$$\text{Re}\bar{X}[k] = \frac{\text{Re}X[k]}{N/2}$$

$$\text{Im}\bar{X}[k] = -\frac{\text{Im}X[k]}{N/2}$$

except for two special cases:

$$\text{Re}\bar{X}[0] = \frac{\text{Re}X[0]}{N}$$

$$\text{Re}\bar{X}[N/2] = \frac{\text{Re}X[N/2]}{N}$$

There are two ways that the synthesis (Eq. 1) can be programmed, and both are shown. In the first method, each of the scaled sinusoids are generated one at a time and added to an accumulation array, which ends up becoming the time domain signal. In the second method, each sample in the time domain signal is calculated one at a time, as the sum of all the corresponding samples in the cosine and sine waves. Both methods produce the same result. The difference between these two programs is very minor; the inner and outer loops are swapped during the synthesis.

ANALYSIS, CALCULATING THE DFT

The DFT can be calculated in three completely different ways. First, the problem can be approached as a set of simultaneous equations. This method is useful for understanding the DFT, but it is too inefficient to be of practical use. The second method brings in an idea from correlation. This is based on detecting a known waveform in another signal. The third method, called the Fast Fourier Transform (FFT), is an ingenious algorithm that decomposes a DFT with N points, into N DFTs each with a single point. The FFT is typically hundreds of times faster than the other methods.

- **DFT BY SIMULTANEOUS EQUATIONS**

The DFT calculation in the following way. You are given N values from the time domain, and asked to calculate the N values of the frequency domain (ignoring the two frequency domain values that you know must be zero). To solve for N unknowns, you must be able to write N linearly independent equations. To do this, take the first sample from each sinusoid and add them together. The sum must be equal to the first sample in the time domain signal, thus providing the first equation.

Likewise, an equation can be written for each of the remaining points in the time domain signal, resulting in the required N equations. The solution can then be found by using established methods for solving simultaneous equations, such as Gauss Elimination. Unfortunately, this method requires a tremendous number of calculations, and is virtually never used in DSP. However, it is important for another reason, it shows why it is possible to decompose a signal into sinusoids, how many sinusoids are needed, and that the basic functions must be linearly independent.

- **DFT BY CORRELATION**

Suppose we are trying to calculate the DFT of a 64 point signal. This means we need to calculate the 33 points in the real part, and the 33 points in the imaginary part of the frequency domain. In this example we will only show how to calculate a single sample, $\text{Im } X[3]$, i.e., the amplitude of the sine wave that makes three complete cycles between point 0 and point 63. All of the other frequency domain values are calculated in a similar manner.

Below Figure illustrates using correlation to calculate $\text{Im}X[3]$. Figures (a) and (b) show two example time domain signals, called: $x1[n]$ and $x2[n]$, respectively. The first signal, $x1[n]$, is composed of nothing but a sine wave that makes three cycles between points 0 and 63. In contrast, $x2[n]$ is composed of several sine and cosine waves, none of which make three cycles between points 0 and 63. These two signals illustrate what the algorithm for calculating $\text{Im}X[3]$ must do. When fed $x1[n]$, the algorithm must produce a value of 32, the amplitude of the sine wave present in the signal. In comparison, when the algorithm is fed the other signal, $x2[n]$, a value of zero must be produced, indicating that this particular sine wave is not present in this signal.

The concept of correlation was to detect a known waveform contained in another signal, multiply the two and add the points in the resulting product. The single number that results from this procedure is a measure of how similar the two signals are. Figure above illustrates this approach. Figures (c) and (d) both display the signal we are looking for, a sine wave that makes 3 cycles between samples 0 and 63. Figure (e) shows the result of multiplying (a) and (c). Likewise, (f) shows the result of multiplying (b) and (d). The sum of all the points in (e) is 32, while the sum of all the points in (f) is zero, showing we have found the desired algorithm.

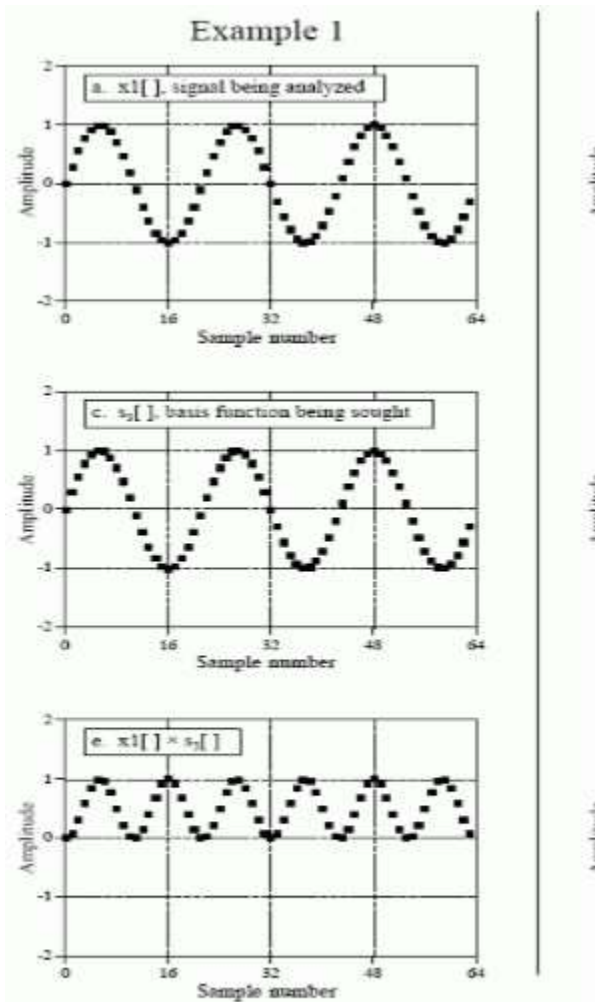


FIGURE 8-8
 Two example signals, (a) and (b), are analyzed for containin
 Figures (c) and (f) show the result of multiplying each exam
 average of 0.5, indicating that x1[] contains the basis functi
 a zero average, indicating that x2[] does not contain the ba

The other samples in the frequency domain are calculated in the same way. This procedure is formalized in the analysis equation, the mathematical way to calculate the frequency domain from the time domain:

EQUATION 8-4

The analysis equations for calculating the DFT. In these equations, $x[i]$ is the time domain signal being analyzed, and $Re X[k]$ & $Im X[k]$ are the frequency domain signals being calculated. The index i runs from 0 to $N-1$, while the index k runs from 0 to $N/2$.

$$Re X[k] = \sum_{i=0}^{N-1} x[i] \cos(2\pi k i / N)$$

$$Im X[k] = - \sum_{i=0}^{N-1} x[i] \sin(2\pi k i / N)$$

In order for this correlation algorithm to work, the basis functions must have an interesting property: each of them must be completely uncorrelated with all of the others. This means that if you multiply any two of the basis functions, the sum of the resulting points will be equal to zero. Basis functions that have this property are called orthogonal. Many other orthogonal basis functions exist, including: square waves, triangle waves, impulses, etc. Signals can be decomposed into these other orthogonal basis functions using correlation, just as done here with sinusoids.

- **DFT BY FAST FOURIER TRANSFORM**

Fourier Series

Any periodic waveform can be decomposed into a series of sine and cosine waves. The general expression is shown below.

$$g(t) = \sum_{n=0}^{\infty} A_n \cos\left(\frac{2\pi n t}{T}\right) + \sum_{n=0}^{\infty} B_n \sin\left(\frac{2\pi n t}{T}\right)$$

The Fast Fourier Transform (FFT) is a development of the DFT which removes duplicated terms in the mathematical algorithm to reduce the number of mathematical operations performed. In this way, it is possible to use large numbers of samples without compromising the speed of the transformation. The FFT reduces computation by a factor of $N/\log_2 N$. The relationship between frequency span, sampling frequency, resolution and display time in a multi-bit FFT analyzer, for a typical sampling frequency of 2.56 times the signal frequency, is summarized in table 1, for FFTs of different lengths N .

The Fourier Transform (FT) is a mathematical operation which converts time-domain signals to the frequency domain. Being able to visualize a signal in the frequency domain offers many benefits over time-domain representations. Frequency domain representations allow individual frequency components contained within a signal to be viewed including modulation sidebands, distortion effects and spurious frequency components.

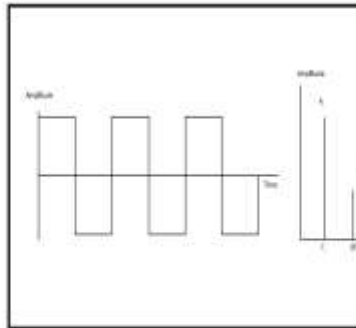


Figure 12 - Time- and frequency-domain representations of a square-wave signal

A good example of this is the square wave of figure. A square wave is composed of a fundamental frequency and all its odd harmonics. The level of each harmonic decreases in amplitude with harmonic number. Viewed in the time domain representation, the square wave gives no indication of its composition. Viewed in the frequency domain all frequencies components are displayed along with their relative amplitude.

In addition, the FT conserves the phase information contained in the signal, which can be used to measure the relative phase of different frequency components, or the phase difference between a number of signals acquired

simultaneously. The Discrete Fourier Transform (DFT) is a version of the FT which can be applied to sampled time-domain signals. The DFT produces a discrete frequency spectrum; i.e. amplitude levels at discrete frequencies, or 'frequency bins'.

N is the total number of samples taken from the original signal. Essentially this equation uses the values, x , of the N samples to calculate the amplitude, X , of the signal at the k th discrete frequency bin. The total frequency spectrum is constructed from all these values over the whole frequency range. Instruments use digital signal processing (DSP) hardware to apply this algorithm to convert the sampled time-domain signal into a frequency-domain one. However, when there are a large number of samples, as is often the case in order to obtain a good representation of the signal, the speed of the conversion process suffers.

FAST FOURIER TRANSFORM

The Fourier transform is the method of changing time representation to frequency representation. The discrete Fourier transform (DFT) is one of the Fourier transforms, used in Fourier analysis. It transforms one function that is time into another that is frequency, so as to get discrete signals, hence called the DFT, of the original function. The DFT of a given sequence $x[n]$ can be computed using the formula

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

$$W_N = e^{-j2\pi/N}$$

Where, W_N is twiddle factor. Twiddle factors referred to as the root of-unity complex multiplicative constants in the butterfly operations of the FFT algorithm, used to recursively combine smaller discrete Fourier transforms. Practically, at the input there is time domain so only real values should be present. But for our convenience we apply input data sequence $x(n)$ having both real and imaginary terms. We observe that for each value of k , direct computation of $X(k)$ involves N complex multiplications ($4N$ real multiplications) and $N-1$ complex additions ($4N-2$ real additions). Consequently, to compute all N values of the DFT requires N^2 complex multiplications and N^2-N complex additions.

The FFT uses a standard three-loop structure for the main FFT computation. A fast Fourier transform (FFT) is an efficient algorithm to compute the discrete Fourier transform (DFT) and its inverse. There are many distinct FFT algorithms involving a wide range of mathematics, from simple complex-number arithmetic to group theory and number theory. The Fast Fourier Transform is an optimized computational algorithm to implement the Discrete Fourier Transform to an array of 2^N samples where, N is the length of samples. It allows determining the frequency of a discrete signal, representing the signal in the frequency domain, convolution, etc. This algorithm has a complexity of $O(N \log_2(N))$. The ordering minimizes the number of fetches or computations of the twiddle-factor values. Since the bit-reverse of a bit-reversed index is the original index, bit-reversal can be performed fairly simply by swapping pairs of data.

HARDWARE REQUIREMENTS

GENERAL

VLSI stands for "Very Large Scale Integration". This is the field which involves packing more and more logic devices into smaller and smaller areas. VLSI circuits that would have taken boardfuls of space can now be put into a small space few millimeters across! This has opened up a big opportunity to do things that were not possible before. VLSI circuits are everywhere: your computer, your car, your brand new state-of-the-art digital camera, the cell-phones, and what have you. All this involves a lot of expertise on many fronts within the same field, which we will look at in later sections. VLSI has been around for a long time, but as a side effect of advances in the world of computers, there has been a dramatic proliferation of tools that can be used to design VLSI circuits. Alongside, obeying Moore's law, the

capability of an IC has increased exponentially over the years, in terms of computation power, utilisation of available area, yield. The combined effect of these two advances is that people can now put diverse functionality into the IC's, opening up new frontiers. Examples are embedded systems, where intelligent devices are put inside everyday objects, and ubiquitous computing where small computing devices proliferate to such an extent that even the shoes you wear may actually do something useful like monitoring your heartbeats. Integrated circuit (IC) technology is the enabling technology for a whole host of innovative devices and systems that have changed the way we live. Jack Kilby and Robert Noyce received the 2000 Nobel Prize in Physics for their invention of the integrated circuit; without the integrated circuit, neither transistors nor computers would be as important as they are today. VLSI systems are much smaller and consume less power than the discrete components used to build electronic systems before the 1960s. Integration allows us to build systems with many more transistors, allowing much more computing power to be applied to solving a problem. Integrated circuits are also much easier to design and manufacture and are more reliable than discrete systems; that makes it possible to develop special-purpose systems that are more efficient than general-purpose computers for the task at hand.

Three Categories

Analog:

Small transistor count precision circuits such as Amplifiers, Data converters, filters, hase Locked, sensors etc...

ASICs or Application Specific Integrated Circuits:

Progress in the fabrication of IC's has enabled us to create fast and powerful circuits in smaller and smaller devices. This also means that we can pack a lot more of functionality into the same area. The biggest application of this ability is found in the design of ASIC's. These are IC's that are created for specific purposes - each device is created to do a particular job, and do it well. The most common application area for this is DSP - signal filters, image compression, etc. To go to extremes, consider the fact that the digital wristwatch normally consists of a single IC doing all the time-keeping jobs as well as extra features like games,calendar, etc.

SOC OR SYSTEMS ON A CHIP:

These are highly complex mixed signal circuits (digital and analog all on the same chip). A network processor chip or a wireless radio chip is an example of an SoC.

APPLICATIONS OF VLSI

Electronic systems now perform a wide variety of tasks in daily life. Electronic systems in some cases have replaced mechanisms that operated mechanically, hydraulically, or by other means; electronics are usually smaller, more flexible, and easier to service. In other cases electronic systems have created totally new applications. Electronic systems perform a variety of tasks, some of them visible, some more hidden:

- Personal entertainment systems such as portable MP3 players and DVD players perform sophisticated algorithms with remarkably little energy.
- Electronic systems in cars operate stereo systems and displays; they also control fuel injection systems, adjust suspensions to varying terrain, and perform the control functions required for anti-lock braking (ABS) systems.
- Digital electronics compress and decompress video, even at high definition data rates, on-the-fly in consumer electronics.
- Low-cost terminals for Web browsing still require sophisticated electronics, despite their dedicated function.
- Personal computers and workstations provide word-processing, financial analysis, and games. Computers include both central processing units (CPUs) and special-purpose hardware for disk access, faster screen display, etc.

Medical electronic systems measure bodily functions and perform complex processing algorithms to warn about unusual conditions. The availability of these complex systems, far from overwhelming consumers, only creates demand for even more complex systems. The growing sophistication of applications continually pushes the design and manufacturing of integrated circuits and electronic systems to new levels of complexity. And perhaps the most

amazing characteristic of this collection of systems is its variety as systems become more complex, we build not a few general-purpose computers but an ever wider range of special-purpose systems. Our ability to do so is a testament to our growing mastery of both integrated circuit manufacturing and design, but the increasing demands of customers continue to test the limits of design and manufacturing.

VERIFICATION TOOL

MODELSIM

ModelSim is a useful tool that allows you to stimulate the inputs of your modules and view both outputs and internal signals. It allows you to do both behavioural and timing simulation, however, this document will focus on behavioural simulation. Keep in mind that these simulations are based on models and thus the results are only as accurate as the constituent models. ModelSim /VHDL, ModelSim /VLOG, ModelSim /LNL, and ModelSim /PLUS are produced by Model Technology™ Incorporated. Unauthorized copying, duplication, or other reproduction is prohibited without the written consent of Model Technology. The information in this manual is subject to change without notice and does not represent a commitment on the part of Model Technology. The program described in this manual is furnished under a license agreement and may not be used or copied except in accordance with the terms of the agreement. The online documentation provided with this product may be printed by the end-user. The number of copies that may be printed is limited to the number of licenses purchased. ModelSim is a registered trademark of Model Technology Incorporated. Model Technology is a trademark of Mentor Graphics Corporation. PostScript is a registered trademark of Adobe Systems Incorporated. UNIX is a registered trademark of AT&T in the USA and other countries. FLEXIm is a trademark of Globetrotter Software, Inc. IBM, AT, and PC are registered trademarks, AIX and RISC System/6000 are trademarks of International Business Machines Corporation. Windows, Microsoft, and MS-DOS are registered trademarks of Microsoft Corporation. OSF/Motif is a trademark of the Open Software Foundation, Inc. in the USA and other countries. SPARC is a registered trademark and SPARCstation is a trademark of SPARC International, Inc. Sun Microsystems is a registered trademark, and Sun, SunOS and Open Windows are trademarks of Sun Microsystems, Inc. All other trademarks and registered trademarks are the properties of their respective holders.

STANDARDS SUPPORTED

ModelSim VHDL supports both the IEEE 1076-1987 and 1076-1993 VHDL, the 1164-1993 Standard Multivalued Logic System for VHDL Interoperability, and the 1076.2-1996 Standard VHDL Mathematical Packages standards. Any design developed with ModelSim will be compatible with any other VHDL system that is compliant with either IEEE Standard 1076-1987 or 1076-1993. ModelSim Verilog is based on IEEE Std 1364-1995 and a partial implementation of 1364-2001, Standard Hardware Description Language Based on the Verilog Hardware Description Language. The Open Verilog International Verilog LRM version 2.0 is also applicable to a large extent. Both PLI (Programming Language Interface) and VCD (Value Change Dump) are supported for ModelSim PE and SE users.

MODELSIM - ADVANCED SIMULATION AND DEBUG

Mentor Graphics was the first to combine single kernel simulator (SKS) technology with a unified debug environment for Verilog, VHDL, and SystemC. The combination of industry-leading, native SKS performance with the best integrated debug and analysis environment make ModelSim the simulator of choice for both ASIC and FPGA designs. The best standards and platform support in the industry make it easy to adopt in the majority of process and tool flows.

MIXED HDL SIMULATION

ModelSim combines simulation performance and capacity with the code coverage and debugging capabilities required to simulate multiple blocks and systems and attain ASIC gate-level sign-off. Comprehensive support of Verilog, SystemVerilog for Design, VHDL, and SystemC provide a solid foundation for single and multi-language design verification environments. ModelSim's easy to use and unified debug and simulation environment provide today's FPGA designers both the advanced capabilities that they are growing to need and the environment that makes their work productive.

EFFECTIVE DEBUG ENVIRONMENT

The ModelSim debug environment's broad set of intuitive capabilities for Verilog, VHDL, and SystemC make it the choice for ASIC and FPGA design. ModelSim eases the process of finding design defects with an intelligently engineered debug environment. The ModelSim debug environment efficiently displays design data for analysis and

debug of all languages. ModelSim allows many debug and analysis capabilities to be employed post-simulation on saved results, as well as during live simulation runs. For example, the coverage viewer analyzes and annotates source code with code coverage results, including FSM state and transition, statement, expression, branch, and toggle coverage. Signal values can be annotated in the source window and viewed in the waveform viewer, easing debug navigation with hyperlinked navigation between objects and its declaration and between visited files. Race conditions, delta, and event activity can be analyzed in the list and wave windows. User-defined enumeration values can be easily defined for quicker understanding of simulation results. For improved debug productivity, ModelSim also has graphical and textual dataflow capabilities.

SYNTHESIS TOOL: [XILINX ISE (v10.1i)]

Xilinx ISE Simulator is a test bench and test fixture creation tool integrated in the Project Navigator framework. It constitutes of Waveform Editor which can be used to graphically enter stimuli and the expected response, and then generate a VHDL test bench or Verilog test fixture. ISE controls all aspects of the design flow. Through the Project Navigator interface, we can access all of the design entry and design implementation tools. We can also access the files and documents associated with your project.

STARTING THE ISE SOFTWARE

To start ISE: Double-click the ISE Project Navigator icon on desktop or select Start > All Programs > Xilinx ISE 10.1i > Project Navigator.

CREATING A NEW PROJECT

1. Create a Verilog source file for the project as follows:
2. Click the New Source button in the New Project Wizard.
3. Select verilog Module as the source type.
4. Type in the file name counter.
5. Verify that the Add to project checkbox is selected.
6. Click Next.
7. Declare the ports for the counter design by filling in the port information as shown below:
8. Click Next, and then Finish in the New Source Information dialog box to complete the new source file template..Click Next, then Next, then Finish.

SIMULATION

The design now is composed of Verilog elements and two cores. We now can synthesize the design using Xilinx ISE simulator. After the design is successfully defined, you will perform behavioral simulation, run implementation with the Xilinx Implementation Tools, perform timing simulation, and configure and download to the Spartan-3 FPGA board.

Simulation Report

Messages					
Rest1/00e	00000011	0000...	00000011		
Rest1/00f	00000000	00000000			
Rest1/010	00000001	0000...	00000001		
Rest1/011	00000000	00000000			
Rest1/012	00000010	0000...	00000010		
Rest1/013	00000000	00000000			
Rest1/014	00000001	0000...	00000001		
Rest1/015	00000000	00000000			
Rest1/016	00001000	0000...	00001000		
Rest1/017	00000000	00000000			
Rest1/018	00000010	0000...	00000010		
Rest1/019	00000000	00000000			
Rest1/01a	00000011	0000...	00000011		
Rest1/01b	00000000	00000000			
Rest1/01c	00000010	0000...	00000010		
Rest1/01d	00000000	00000000			
Rest1/01e	00000000	00000000			
Rest1/01f	00000001	0000...	00000001		
Rest1/020	00000000	00000000			
Rest1/021	00001001	0000...	00001001		
Rest1/022	00000000	00000000			
Rest1/023	00000111	0000...	00000111		
Rest1/024	00000000	00000000			
Rest1/025	00000001	0000...	00000001		
Rest1/026	00000000	00000000			
Rest1/027	00000110	0000...	00000110		
Rest1/028	00000000	00000000			

Messages					
Rest1/029	00001010	0000...	00001010		
Rest1/02a	00000000	00000000			
Rest1/02b	00000101	0000...	00000101		
Rest1/02c	00000000	00000000			
Rest1/02d	00000010	0000...	00000010		
Rest1/02e	00000000	00000000			
Rest1/02f	00000000	00000000			
Rest1/030	00000001	0000...	00000001		
Rest1/031	00000000	00000000			
Rest1/032	00000010	0000...	00000010		
Rest1/033	00000000	00000000			
Rest1/034	00000000	00000000			
Rest1/035	00000001	0000...	00000001		
Rest1/036	00000000	00000000			
Rest1/037	00000000	00000000			
Rest1/038	00000010	0000...	00000010		
Rest1/039	00000000	00000000			
Rest1/03a	00000000	00000000			
Rest1/03b	00000001	0000...	00000001		
Rest1/03c	00000000	00000000			
Rest1/03d	00000001	0000...	00000001		
Rest1/03e	00000000	00000000			
Rest1/03f	00000000	00000000			
Rest1/040	00000001	0000...	00000001		
Rest1/041	00000000	00000000			
Rest1/042	00000000	00000000			
Rest1/043	00000000	00000000			
Rest1/044	00000000	00000000			
Rest1/045	00000000	00000000			
Rest1/046	00000001	0000...	00000001		
Rest1/047	00000000	00000000			

test1/X28r	00000000	00000000		
test1/X28i	00000000	00000000		
test1/X29r	00000010	0000...00000010		
test1/X29i	00000000	00000000		
test1/X30r	00000001	0000...00000001		
test1/X30i	00000000	00000000		
test1/X31r	00000010	0000...00000010		
test1/X31i	00000000	00000000		
test1/Y0r	01001000	0000...01001000		
test1/Y0i	00000000	00000000		
test1/Y1r	01101000	0000...01101000		
test1/Y1i	10110110	0000...10110110		
test1/Y2r	00011101	0000...00011101		
test1/Y2i	10001101	0000...10001101		
test1/Y3r	00001111	0000...00001111		
test1/Y3i	10100010	0000...10100010		
test1/Y4r	10100111	0000...10100111		
test1/Y4i	01010010	0000...01010010		
test1/Y5r	00110111	0000...00110111		
test1/Y5i	10110000	0000...10110000		
test1/Y6r	10101110	0000...10101110		
test1/Y6i	01100101	0000...01100101		
test1/Y7r	01111011	0000...01111011		
test1/Y7i	00011101	0000...00011101		
test1/Y8r	11111001	0000...11111001		
test1/Y8i	11110101	0000...11110101		
test1/Y9r	01110111	0000...01110111		
test1/Y9i	10001101	0000...10001101		
test1/Y10r	00011100	0000...00011100		
test1/Y10i	11000110	0000...11000110		
test1/Y11r	00110001	0000...00110001		
test1/Y11i	10101100	0000...10101100		
test1/Y12r	01001111	0000...01001111		
test1/Y12i	01010110	0000...01010110		
test1/Y13r	00110001	0000...00110001		
test1/Y13i	10111110	0000...10111110		
test1/Y14r	00010101	0000...00010101		
test1/Y14i	00011001	0000...00011001		
test1/Y15r	00010010	0000...00010010		
test1/Y15i	01000110	0000...01000110		
test1/Y16r	00000010	0000...00000010		
test1/Y16i	00000000	00000000		
test1/Y17r	00010010	0000...00010010		
test1/Y17i	10111010	0000...10111010		
test1/Y18r	00010101	0000...00010101		
test1/Y18i	11100111	0000...11100111		
test1/Y19r	00000011	0000...00000011		
test1/Y19i	11011010	0000...11011010		
test1/Y20r	01001111	0000...01001111		
test1/Y20i	10101010	0000...10101010		
test1/Y21r	11111111	0000...11111111		
test1/Y21i	10111100	0000...10111100		
test1/Y22r	00100100	0000...00100100		
test1/Y22i	00000111	0000...00000111		
test1/Y23r	01110111	0000...01110111		
test1/Y23i	01110011	0000...01110011		
test1/Y24r	11111001	0000...11111001		
test1/Y24i	00001011	0000...00001011		
test1/Y25r	01111011	0000...01111011		
test1/Y25i	11100011	0000...11100011		
test1/Y26r	10110110	0000...10110110		
test1/Y26i	11001110	0000...11001110		
test1/Y27r	01101001	0000...01101001		
test1/Y27i	10111000	0000...10111000		
test1/Y28r	10100111	0000...10100111		
test1/Y28i	10101110	0000...10101110		
test1/Y29r	01000001	0000...01000001		
test1/Y29i	11110110	0000...11110110		
test1/Y30r	00011101	0000...00011101		
test1/Y30i	01110011	0000...01110011		
test1/Y31r	01101000	0000...01101000		
test1/Y31i	01001010	0000...01001010		

सम	10 म	15 म	20 म	25 म	30 म
सम 10:00:00	3000000			3000000	
सम 10:00:05	3000000			3111111	
सम 10:00:10	3000000			3222222	
सम 10:00:15	3000000			3333333	
सम 10:00:20	3000000			3444444	
सम 10:00:25	3000000			3555555	
सम 10:00:30	3000000			3666666	
सम 10:00:35	3000000			3777777	
सम 10:00:40	3000000			3888888	
सम 10:00:45	3000000			3999999	
सम 10:00:50	3000000			4111111	
सम 10:00:55	3000000			4222222	
सम 10:01:00	3000000			4333333	
सम 10:01:05	3000000			4444444	
सम 10:01:10	3000000			4555555	
सम 10:01:15	3000000			4666666	
सम 10:01:20	3000000			4777777	
सम 10:01:25	3000000			4888888	
सम 10:01:30	3000000			4999999	
सम 10:01:35	3000000			5111111	
सम 10:01:40	3000000			5222222	
सम 10:01:45	3000000			5333333	
सम 10:01:50	3000000			5444444	
सम 10:01:55	3000000			5555555	
सम 10:02:00	3000000			5666666	
सम 10:02:05	3000000			5777777	
सम 10:02:10	3000000			5888888	
सम 10:02:15	3000000			5999999	
सम 10:02:20	3000000			6111111	
सम 10:02:25	3000000			6222222	
सम 10:02:30	3000000			6333333	
सम 10:02:35	3000000			6444444	
सम 10:02:40	3000000			6555555	
सम 10:02:45	3000000			6666666	
सम 10:02:50	3000000			6777777	
सम 10:02:55	3000000			6888888	
सम 10:03:00	3000000			6999999	
सम 10:03:05	3000000			7111111	
सम 10:03:10	3000000			7222222	
सम 10:03:15	3000000			7333333	
सम 10:03:20	3000000			7444444	
सम 10:03:25	3000000			7555555	
सम 10:03:30	3000000			7666666	
सम 10:03:35	3000000			7777777	
सम 10:03:40	3000000			7888888	
सम 10:03:45	3000000			7999999	
सम 10:03:50	3000000			8111111	
सम 10:03:55	3000000			8222222	
सम 10:04:00	3000000			8333333	
सम 10:04:05	3000000			8444444	
सम 10:04:10	3000000			8555555	
सम 10:04:15	3000000			8666666	
सम 10:04:20	3000000			8777777	
सम 10:04:25	3000000			8888888	
सम 10:04:30	3000000			8999999	
सम 10:04:35	3000000			9111111	
सम 10:04:40	3000000			9222222	
सम 10:04:45	3000000			9333333	
सम 10:04:50	3000000			9444444	
सम 10:04:55	3000000			9555555	
सम 10:05:00	3000000			9666666	
सम 10:05:05	3000000			9777777	
सम 10:05:10	3000000			9888888	
सम 10:05:15	3000000			9999999	
सम 10:05:20	3000000			10000000	

सम	10 म	15 म	20 म	25 म	30 म
सम 10:00:00	3000000			3000000	
सम 10:00:05	3000000			3111111	
सम 10:00:10	3000000			3222222	
सम 10:00:15	3000000			3333333	
सम 10:00:20	3000000			3444444	
सम 10:00:25	3000000			3555555	
सम 10:00:30	3000000			3666666	
सम 10:00:35	3000000			3777777	
सम 10:00:40	3000000			3888888	
सम 10:00:45	3000000			3999999	
सम 10:00:50	3000000			4111111	
सम 10:00:55	3000000			4222222	
सम 10:01:00	3000000			4333333	
सम 10:01:05	3000000			4444444	
सम 10:01:10	3000000			4555555	
सम 10:01:15	3000000			4666666	
सम 10:01:20	3000000			4777777	
सम 10:01:25	3000000			4888888	
सम 10:01:30	3000000			4999999	
सम 10:01:35	3000000			5111111	
सम 10:01:40	3000000			5222222	
सम 10:01:45	3000000			5333333	
सम 10:01:50	3000000			5444444	
सम 10:01:55	3000000			5555555	
सम 10:02:00	3000000			5666666	
सम 10:02:05	3000000			5777777	
सम 10:02:10	3000000			5888888	
सम 10:02:15	3000000			5999999	
सम 10:02:20	3000000			6111111	
सम 10:02:25	3000000			6222222	
सम 10:02:30	3000000			6333333	
सम 10:02:35	3000000			6444444	
सम 10:02:40	3000000			6555555	
सम 10:02:45	3000000			6666666	
सम 10:02:50	3000000			6777777	
सम 10:02:55	3000000			6888888	
सम 10:03:00	3000000			6999999	
सम 10:03:05	3000000			7111111	
सम 10:03:10	3000000			7222222	
सम 10:03:15	3000000			7333333	
सम 10:03:20	3000000			7444444	
सम 10:03:25	3000000			7555555	
सम 10:03:30	3000000			7666666	
सम 10:03:35	3000000			7777777	
सम 10:03:40	3000000			7888888	
सम 10:03:45	3000000			7999999	
सम 10:03:50	3000000			8111111	
सम 10:03:55	3000000			8222222	
सम 10:04:00	3000000			8333333	
सम 10:04:05	3000000			8444444	
सम 10:04:10	3000000			8555555	
सम 10:04:15	3000000			8666666	
सम 10:04:20	3000000			8777777	
सम 10:04:25	3000000			8888888	
सम 10:04:30	3000000			8999999	
सम 10:04:35	3000000			9111111	
सम 10:04:40	3000000			9222222	
सम 10:04:45	3000000			9333333	
सम 10:04:50	3000000			9444444	
सम 10:04:55	3000000			9555555	
सम 10:05:00	3000000			9666666	
सम 10:05:05	3000000			9777777	
सम 10:05:10	3000000			9888888	
सम 10:05:15	3000000			9999999	
सम 10:05:20	3000000			10000000	

क्र.सं.	नाम	पता	सं.सं.	पता	सं.सं.
1	श्री. अ. अ. अ.				
2	श्री. अ. अ. अ.				
3	श्री. अ. अ. अ.				
4	श्री. अ. अ. अ.				
5	श्री. अ. अ. अ.				
6	श्री. अ. अ. अ.				
7	श्री. अ. अ. अ.				
8	श्री. अ. अ. अ.				
9	श्री. अ. अ. अ.				
10	श्री. अ. अ. अ.				
11	श्री. अ. अ. अ.				
12	श्री. अ. अ. अ.				
13	श्री. अ. अ. अ.				
14	श्री. अ. अ. अ.				
15	श्री. अ. अ. अ.				
16	श्री. अ. अ. अ.				
17	श्री. अ. अ. अ.				
18	श्री. अ. अ. अ.				
19	श्री. अ. अ. अ.				
20	श्री. अ. अ. अ.				
21	श्री. अ. अ. अ.				
22	श्री. अ. अ. अ.				
23	श्री. अ. अ. अ.				
24	श्री. अ. अ. अ.				
25	श्री. अ. अ. अ.				
26	श्री. अ. अ. अ.				
27	श्री. अ. अ. अ.				
28	श्री. अ. अ. अ.				
29	श्री. अ. अ. अ.				
30	श्री. अ. अ. अ.				
31	श्री. अ. अ. अ.				
32	श्री. अ. अ. अ.				
33	श्री. अ. अ. अ.				
34	श्री. अ. अ. अ.				
35	श्री. अ. अ. अ.				
36	श्री. अ. अ. अ.				
37	श्री. अ. अ. अ.				
38	श्री. अ. अ. अ.				
39	श्री. अ. अ. अ.				
40	श्री. अ. अ. अ.				
41	श्री. अ. अ. अ.				
42	श्री. अ. अ. अ.				
43	श्री. अ. अ. अ.				
44	श्री. अ. अ. अ.				
45	श्री. अ. अ. अ.				
46	श्री. अ. अ. अ.				
47	श्री. अ. अ. अ.				
48	श्री. अ. अ. अ.				
49	श्री. अ. अ. अ.				
50	श्री. अ. अ. अ.				
51	श्री. अ. अ. अ.				
52	श्री. अ. अ. अ.				
53	श्री. अ. अ. अ.				
54	श्री. अ. अ. अ.				
55	श्री. अ. अ. अ.				
56	श्री. अ. अ. अ.				
57	श्री. अ. अ. अ.				
58	श्री. अ. अ. अ.				
59	श्री. अ. अ. अ.				
60	श्री. अ. अ. अ.				
61	श्री. अ. अ. अ.				
62	श्री. अ. अ. अ.				
63	श्री. अ. अ. अ.				
64	श्री. अ. अ. अ.				
65	श्री. अ. अ. अ.				
66	श्री. अ. अ. अ.				
67	श्री. अ. अ. अ.				
68	श्री. अ. अ. अ.				
69	श्री. अ. अ. अ.				
70	श्री. अ. अ. अ.				
71	श्री. अ. अ. अ.				
72	श्री. अ. अ. अ.				
73	श्री. अ. अ. अ.				
74	श्री. अ. अ. अ.				
75	श्री. अ. अ. अ.				
76	श्री. अ. अ. अ.				
77	श्री. अ. अ. अ.				
78	श्री. अ. अ. अ.				
79	श्री. अ. अ. अ.				
80	श्री. अ. अ. अ.				
81	श्री. अ. अ. अ.				
82	श्री. अ. अ. अ.				
83	श्री. अ. अ. अ.				
84	श्री. अ. अ. अ.				
85	श्री. अ. अ. अ.				
86	श्री. अ. अ. अ.				
87	श्री. अ. अ. अ.				
88	श्री. अ. अ. अ.				
89	श्री. अ. अ. अ.				
90	श्री. अ. अ. अ.				
91	श्री. अ. अ. अ.				
92	श्री. अ. अ. अ.				
93	श्री. अ. अ. अ.				
94	श्री. अ. अ. अ.				
95	श्री. अ. अ. अ.				
96	श्री. अ. अ. अ.				
97	श्री. अ. अ. अ.				
98	श्री. अ. अ. अ.				
99	श्री. अ. अ. अ.				
100	श्री. अ. अ. अ.				

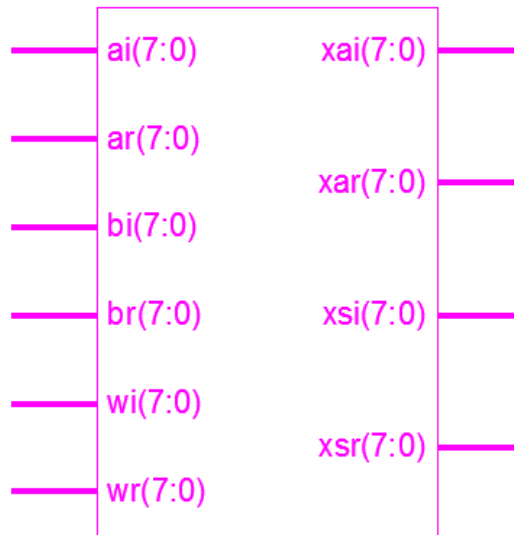
Synthesis Report

test1 Project Status (12/07/2018 - 161443)			
Project File:	shby.vise	Parser Errors:	No Errors
Module Name:	topmodule	Implementation State:	Synthesized
Target Device:	codyc760-2F(176)	*Errors:	No Errors
Product Version:	ISE 13.2	*Warnings:	4 Warnings (3 new)
Design Goal:	Balanced	*Routing Results:	
Design Strategy:	Wire Default (united)	*Timing Constraints:	
Environment:	Custom Settings	*Final Timing Score:	

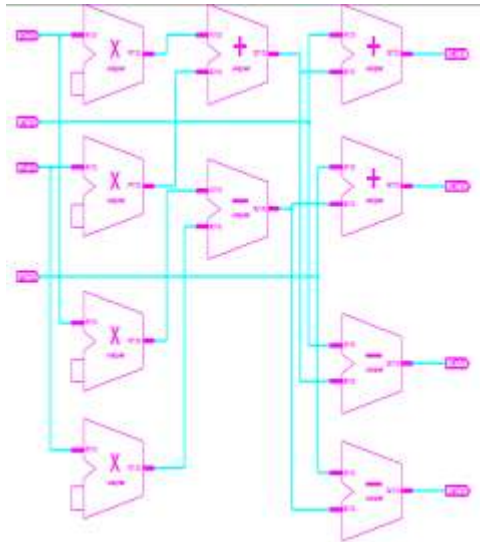
Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	768	474240	0%
Number of fully used LUT-FF pairs :	0	768	0%
Number of bonded IOBs	1024	1200	85%
Number of DSP48Es	544	884	62%

Detailed Reports					
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Fri Dec 7 05:14:42 2018	0	4 Warnings (3 new)	1 Info (0 new)
Translation Report	Out of Date	Fri Dec 7 04:55:38 2018	0	0	0
Place Report	Out of Date	Fri Dec 7 05:00:17 2018	0	0	6 Infos (5 new)

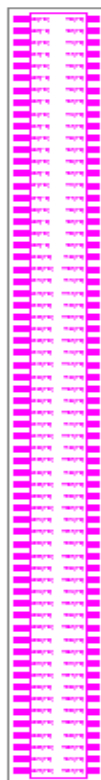
Block Diagram of 2- point DFT



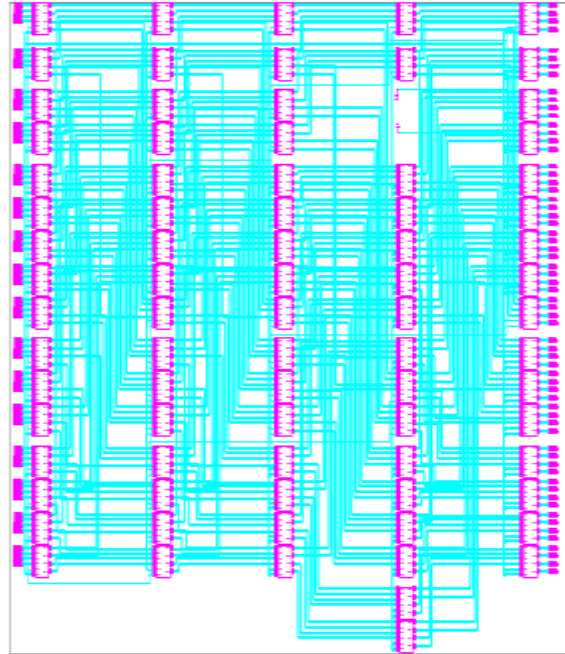
RTL Schematic of 2-point Butterfly computation



Block Diagram of 32-point FFT Algorithm



RTL Schematic of 32-point FFT Algorithm



REFERENCES

1. Sneha N.kherde, Meghana Hasamnis, "Efficient Design and Implementation of FFT", International Journal of Engineering Science and Technology (IJEST), ISSN : 0975-5462 NCICT Special Issue Feb 2011
2. Ahmed Saeed, M. Elbably, G. Abdelfadeel, and M. I. Eladawy, "Efficient FPGA implementation of FFT/IFFT Processor", INTERNATIONAL JOURNAL OF CIRCUITS, SYSTEMS AND SIGNAL PROCESSING, Issue 3, Volume 3, 2009
3. HardwareDescriptionLanguage.URL:http://en.wikipedia.org/wiki/Hardware_description_language.
4. Very High Speed Integrated Circuit Hardware Description Language. URL: <http://electrosofts.com/vhdl/>
5. Alan V. Oppenheim, Ronald W. Schaffer with John R. Buck, Discrete Time Signal Processing, Second Edition
6. B. Parhami, Computer Arithmetic, Algorithms and Hardware Designs, 1999
7. James W. Cooley and John W. Tukey, An Algorithm for the Machine Calculation of Complex Fourier Series
8. Peter J. Ashenden, The Designer's Guide to VHDL, Second Edition.
9. N. Weste, M. Bickerstaff, T. Arivoli, P.J. Ryan, J. W. Dalton, D.J. Skellern", and T.M. Percivalt A 50Mhz 16Point-FFT processor for WLAN applications.
10. Saad Bouguezel, M. Omair Ahmad, "IMPROVED RADIX-4 AND RADIX-8 FFT ALGORITHMS" IEEE. Department of Electrical and Computer Engineering Concordia University 1455 de Maisonneuve Blvd. West Montreal, P.Q., Canada.
11. Ali Saidi , " DECIMATION-IN-TIME-FREQUENCY FFT ALGORITHM" Motorola Applied Research, Paging and Wireless Data Group Boynton Beach.
12. Rizalafande Che Ismail and Razaidi Hussin "High Performance Complex Number Multiplier Using Booth-Wallace Algorithm" School of Microelectronic Engineering Kolej University Kejuruteraan Utara Malaysia.