Implementation of FFT algorithm using FPGA technique

Dr.Thamer M. Jamel
Department of electrical & electronic engineering/ University of Technology Baghdad – Iraq
Email:thamerjamel@yahoo.com

Keyword: FFT, FPGA

Abstract:-

The fast Fourier transform (FFT) algorithm now play in important role in the design of digital signal processing system for communications, measurement and control applications. Although the algorithm was developed for use on general- purpose computers, or special purpose microprocessor (DSP µPs). Semiconductor technology now make it economically feasible to implement FFT using field programming grids array (FPGA) technique. In this paper 8- point FFT algorithm was implemented using FPGA technique type (Xilinx) with the aid of schematic software package .

1.Introduction

The Fourier transform converts information from the time domain into the frequency domain. It is an important analytical tool in such diverse fields as acoustics, optics, seismology, telecommunications, speech, signal processing, and image processing. The FFT is the generic name for a class of computationally efficient algorithms and are widely used in the field of digital signal processing [1,2]. FFT can be implemented by using different techniques. One of the modern and new techniques is FPGA approach. FPGA as it is more widely called is a type of programmable device. Programmable device are a class of general – purpose chips that can be configured for a wide variety of applications.

There are two main basic hardware design methodology currently available for FPGA technique: Schematic and language based designs. For the purpose of this paper the first one was chosen due to this design gives better optimization implementation in terms of both area and speed. In this paper, first, basic concept of FPGA technique will be described in details which will cover the necessary subjects that needed in the application design, the design procedures for FPGA technique will be presented then an application of implementation of 8-points FFT algorithm will be presented with its results.

2. Basic Concept of FPGA

Field Programmable Gate Arrays (FPGAs) are one of the fastest growing segments of the semiconductor industry. They were first introduced in 1985, and since then they have quickly gained widespread acceptance as an excellent technology for implementing moderately large digital circuits in low production volumes [3, 4, and 5]. FPGAs are

programmable devices that can be directly configured by the end user without the use of an integrated circuit fabrication facility. They offer the designer the benefits of custom hardware, eliminating high development costs and manufacturing time. Figure (1) shows a conceptual diagram of a typical FPGA [5].

Field Programmable Gate Arrays are called this because rather than having a structure similar to a PAL or other programmable device, they are structured very much like a gate array ASIC (Application Specific Integrated Circuit) [3].

The first programmable device was the programmable array logic (PAL). One of the PAL devices is PLD. Programmable Logic Devices (PLDs) are programmable devices that can be configured for a wide variety of applications. They enable faster implementation and emulation of circuit designs on hardware. The flexibility provided by these devices through the presence of reconfigurable elements has increased their popularity [6].

There are two major types of PLDs: Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs). Among the various possible FPGA architectures, lookup-table (LUT) based FPGA architectures have been the most popular ones. A LUT-based FPGA consists of an array of programmable logic blocks (PLBs) together with programmable interconnections. [7]

The maximum numbers of gates in an FPGA are as high as 500,000 and doubling every 18 months while the prices dropped dramatically [8].

3. Field Programmable Devices

FPDs, or Field Programmable Devices, is a general term for all devices that can programmed (and possibly reprogrammed) after fabrication. Several standard approaches to programmability are used in the industry in the form of PROMs, PLAs, PALs, CPLDs, and FPGAs. These approaches vary significantly in their complexity (and subsequently their cost) and the applications for which they are best suited. Amongst the largest and fastest growing FPDs are Field -Programmable Gate Arrays (FPGAs).

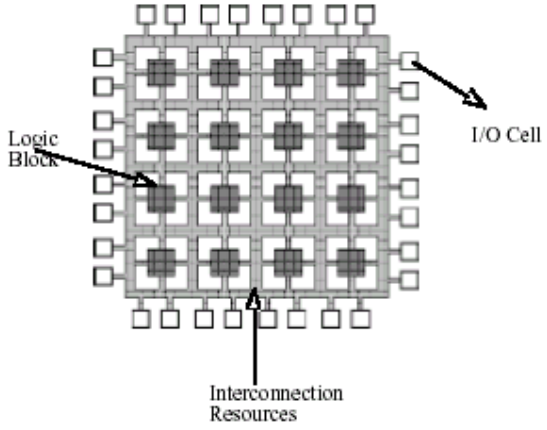

Figure (1): conceptual diagram of FPGA.

Although there are many types of FPGAs, all architectures include logic blocks, I/O blocks, and programmable routing, which are arranged in a regular pattern. FPGA provide narrow logic resources; in other words, their logic blocks are generally small and uncommitted. One advantage of an FPGA over other types of FPDs is that they generally have much higher logic capacities than other FPDs and offer a higher ratio of flip-flops to logic [9]. A higher ratio of flip-flops to logic is important because flip-flops are often the limiting factor in designs.

FPGAs are the most common form of FPD offered by programmable logic vendors. One such vendor, Xilinx, offers several different "families" of FPGAs that target different design sizes, design speeds, and cost requirements. Some of the more popular devices include the XC4000, the Spartan series, and the Virtex II series.

Connection blocks facilitate connectivity between logic block pins and the routing channels. Each input pin can be programmed to connect to one or more of the tracks in a channel using either a multiplexer or multiple transistors (see Figure 2). Output pins, on the other hand, are connected to tracks using tri-state buffers. The number of tracks that a pin can connect to is called its *connection block flexibility*.

Switch blocks reside at the intersections of horizontal and vertical routing channels. They provide programmable switches used to connect tracks from both the vertical and horizontal channels incident to the switch. The number of outgoing tracks that each ingoing track can connect to is called its *switch block flexibility*. [10]

An FPGA generally consists of a two-dimensional array of *logic blocks* that can be connected by general interconnection resources. The

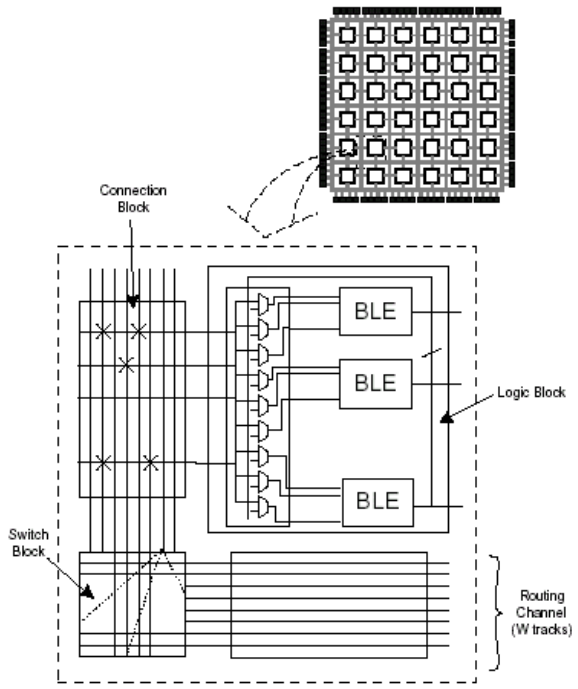

Figure (2): An FPGA consists of a logic block, vertical & horizontal channels, switch block and connection blocks.

interconnect comprises segments of wire, where the segments may be of various lengths. The interconnect resources include programmable switches that serve to connect the logic blocks to one another or one wire segment to another. Logic circuits are implemented in the FPGA by partitioning the logic into individual logic blocks and then interconnecting the blocks as required via the switches.

The structure and content of a logic block are called its *architecture.* There are different kinds of logic block architecture available, and logic blocks can be built using look-up tables (Xilinx), multiplexers (Actel) or even PALs (Altera).

The structure and content of the interconnect resources in an FPGA is called its *Routing architecture.* The routing architecture consists of wire segments and programmable switches. There exists many different ways to design the structure of a routing architecture, some FPGAs offer simple connection between blocks, and others provide fewer, but more complex routes [5]. Each manufacturer has a distinct name for their basic block, but the fundamental unit is the LUT. Altera call theirs a Logic Element (LE) while Xilinx's FPGAs have configurable logic blocks (CLBs) organized in an array [11].

4: Basic Building Blocks

Xilinx user-programmable gate arrays include two major configurable elements: configurable logic blocks (CLBs) and input/output blocks (IOBs).

- CLBs provide the functional elements for constructing the user's logic.
- IOBs provide the interface between the package pins and internal signal lines. Programmable interconnect resources provide routing paths to connect the inputs and outputs of these configurable elements to the appropriate networks. Customized configuration is established by programming internal static memory cells that determine the logic functions and internal connections implemented in the FPGA.

Configurable Logic Blocks implement most of the logic in an FPGA.
The principal CLB elements are shown in figure (3).

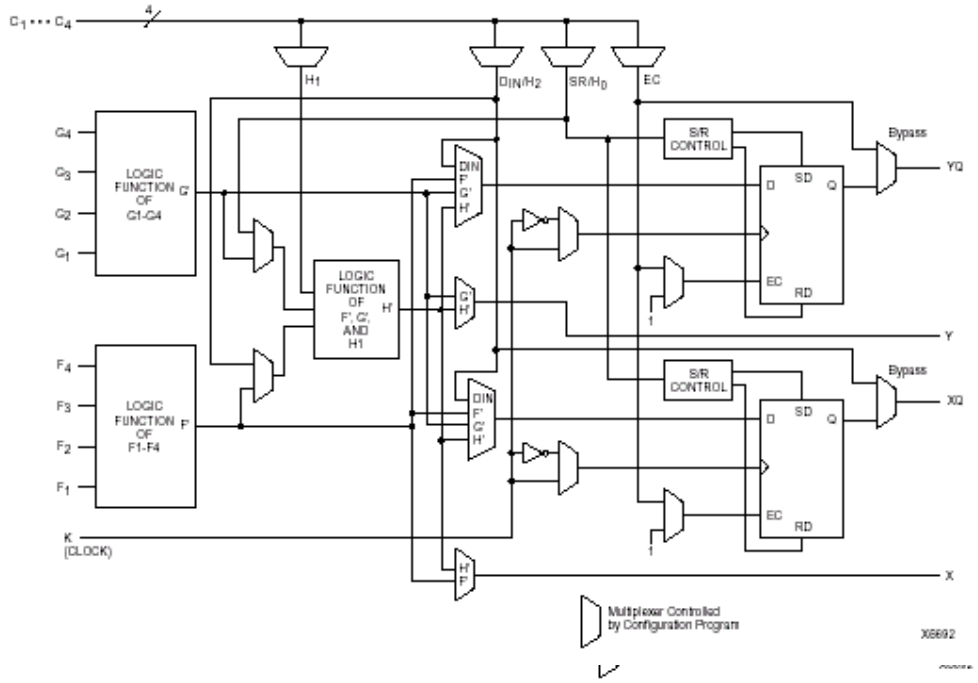

Figure (3): Simplified Block Diagram of XC4000 Series CLB

Two 4-input function generators (F and G) offer unrestricted versatility. Most combinatorial logic functions need four or fewer inputs. However, a third function generator (H) is provided. The H function generator has three inputs. Either zero, one, or two of these inputs can be the outputs of F and G; the other input(s) are from outside the CLB. The CLB can, therefore, implement certain functions of up to nine variables, like comparison of two sets of four inputs.

Each CLB contains two storage elements that can be used to store the function generator outputs. However, the storage elements and function

generators can also be used independently. H1 can drive the other through the H function generator. Function generator outputs can also drive two outputs independent of the storage element outputs. This versatility increases logic capacity and simplifies routing. [12]

5. CAD for FPGAs

Computer-aided design (CAD) is a very important aspect of FPGA technology. It allows users to convert a circuit description represented in a hardware description language (HDL) or as a schematic to a bit stream that can be uploaded to an FPGA for programming.

Examples of CAD software for FPGAs are Xilinx Alliance and Foundation, Altera Quartus II and Max+Plus II, and Actel Libero.

Implementing a logic design with an FPGA usually consists of the following steps (depicted in the figure (6), which follows):

1. You enter a description of your logic circuit using a *hardware description language* (HDL) such as VHDL or Verilog. You can also draw your design using a schematic editor.
2. You use a *logic synthesizer* program to transform the HDL or schematic into a *netlist*. The netlist is just a description of the various logic gates in your design and how they are interconnected.
3. You use the *implementation tools* to map the logic gates and interconnections into the FPGA. The FPGA consists of many *configurable logic blocks* which can be further decomposed into *look-up tables* that perform logic operations. The mapping tool collects your netlist gates into groups that fit into the LUTs and then the place & route tool assigns the gate collections to specific CLBs while opening or closing the switches in the routing matrices to connect the gates together.
4. Once the implementation phase is complete, a program extracts the state of the switches in the routing matrices and generates a *bitstream* where the ones and zeroes correspond to open or closed switches.

The bitstream is *downloaded* into a physical FPGA chip. The electronic switches in the FPGA open or close in response to the binary bits in the bitstream. Upon completion of the downloading, the FPGA will perform the operations specified by your HDL code or schematic. [13] .The following figure (Fig(6)) shows the design flow of the FPGA.

6: Device Configuration: -

Configuring these programmable devices can be achieved in several ways, such as schematic design entry, the use of hardware description languages (HDLs), and the use of high-level language compilers [11].

Schematic design practices entails selecting standard logic gates from a library to create a graphic description of the circuit to be realized, and manually wiring them together. The schematic design library typically includes standard Boolean logic gates, multiplexers, I/O buffers, and macros for device specific functions, such as counter. Custom components can be constructed from the smaller blocks to create user macros for use in large designs. Once the logic equations are determined the circuit can be easily assembled [11].

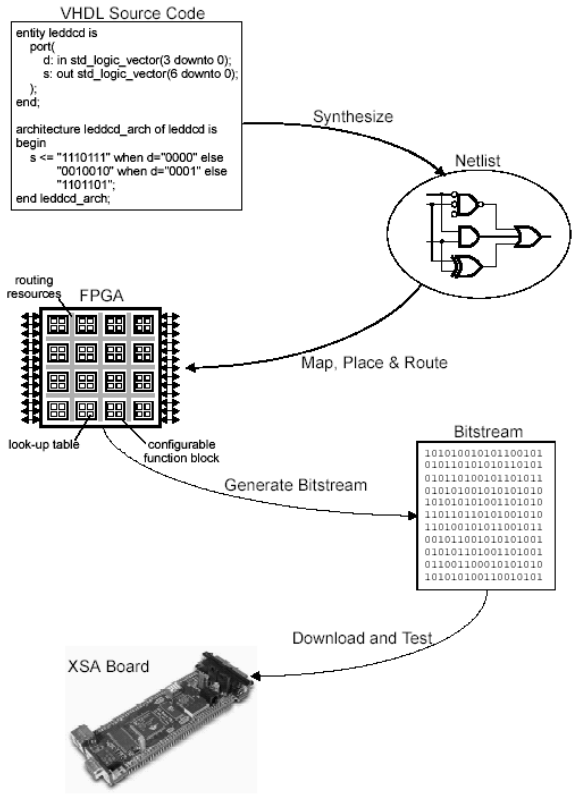

Figure (4): Design follow of FPGA

7. Decimation-In-Time (DIT) FFT Butterfly

In the radix-2 DIT FFT algorithm, the time decimation process passes through a total of M stages where $N=2^M$ with N/2 2-point FFT or butterflies per stage ,giving a total of $(N/2)\log_2 N$ butterflies per N-point FFT. For the case of 8-point FFT implemented using radix-2 DIT FFT algorithm, that shown in the figure (5), each input samples are processed through three stages. Four butterflies are required per stage, giving a total of twelve butterflies in the radix-2 implementation. Each butterfly is 2-point DFT of the form depicted in figure (6). P and Q are the inputs to the radix-2 DIT FFT butterfly. As shown in this figure, the outputs P' and Q' of the radix-2 butterfly are given by [14]:-

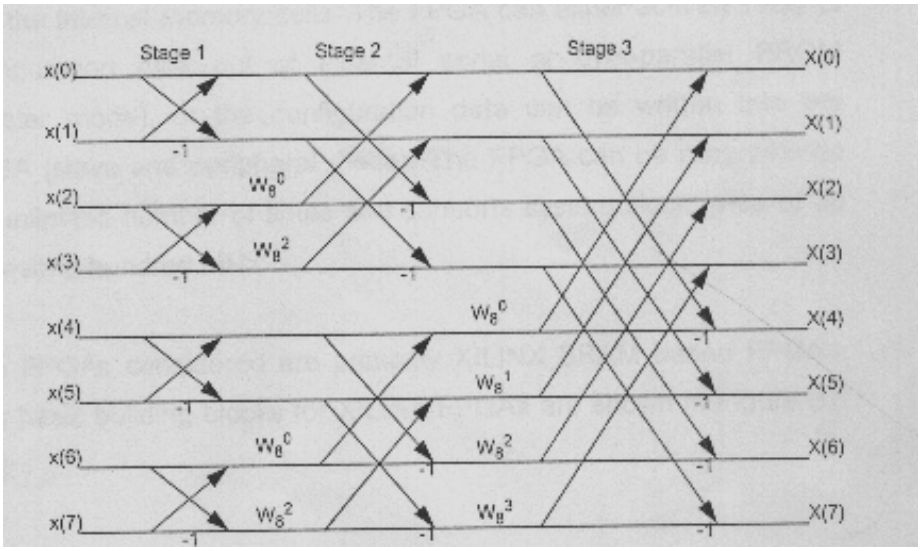

Figure (5) Radix – 2 DFT structure for 8- point FFT

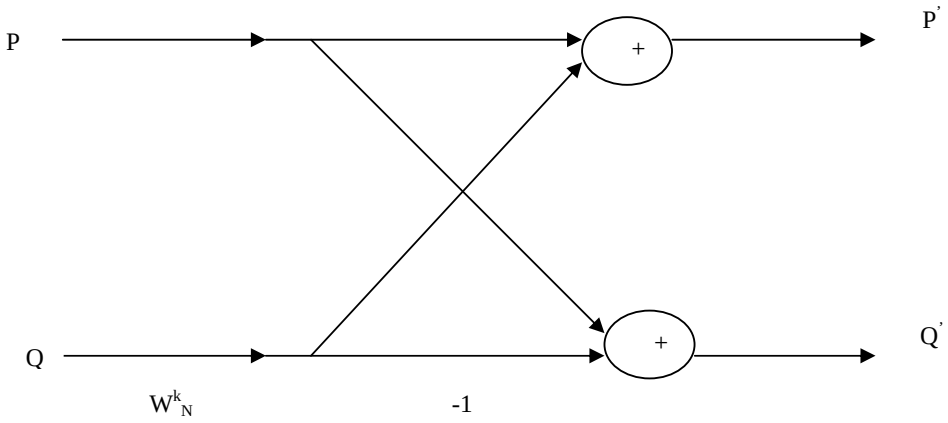

Figure (6) Radix-2 DIT FFT butterfly Flowgragh

$$P' = P + QW_N^k \qquad \text{……(1)}$$

$$Q' = P - QW_N^k \qquad \text{……(2)}$$

Where W_N^k is twiddle factor and it is equal to

$$e^{-j(2\pi/N)k} = \cos(X) - j\sin(X) \qquad \text{……(3)}$$

Here $X = (2\pi/N)k$, $j = \sqrt{-1}$, and k=0,1,2,…, N-1

The inputs P an Q can be expressed in terms of their real (R) and imaginary (I) parts as follows [14]:-

$$\begin{aligned} P &= PR + jPI \\ Q &= QR + jQI \end{aligned} \qquad \text{…….. (4)}$$

Now by substituting (3) and rearranged according to (4), then equations (1) and (2) becomes:-

$$\begin{aligned} P' &= \left(PR + QR\cos(X) + QI\sin(X)\right) + j\left(PI + QI\cos(X) - QR\sin(X)\right) \\ Q'' &= \left(PR - QR\cos(X) + QI\sin(X)\right) + j\left(PI - QI\cos(X) - QR\sin(X)\right) \end{aligned} \qquad \text{…….(5)}$$

8- FFT Design with FPGA

The Radix – 2, eight – point FFT- DIT that shown in figure (5) was built by using schematic technique according to the steps that has been explained previously in section (5).

1- The first and second stage of FFT requires adder, subtraction and register to implement equation (5) which is shown in the figure (7). The symbolic representation of the adder/subtraction that used in the figure (7) is shown in figure (8).

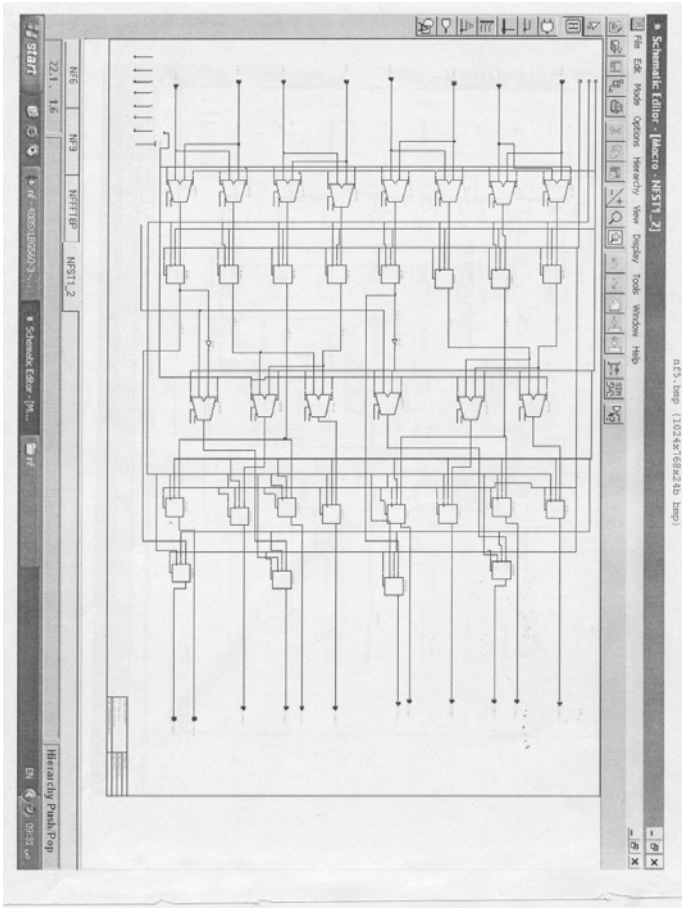

Fig (7) Schematic diagram of first & Second Stage of FFT

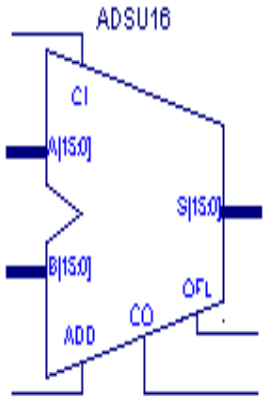

Fig (8): Symbolic representation of the adder/subtraction.

Where:

CI: Is 1-bit input carry-in.
A [15:0]: Are the 16-bit input pins of the first number.
B [15:0]: Are the 16-bit input pins of the second number.
ADD: Is the addition operation selected pin, it is active state is1.
If it is 0 then the operation selected is subtraction.
OFL: Is the overflow pin.
CO: Is the output carry pin.
S [15:0]: Is the 16-bits output data vector pins.
The adder/subtraction tested by using simulation tools. The arbitrary input vectors were applied to the input and the resultant output was recorded.
Also ,the 16-bit data register with clock enable and Asynchronous clear is shown in the figure (9) below:-

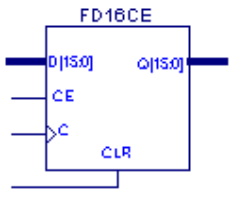

Fig (9) 16-bits Data Register

Data register is used to store the input values such as (X_O, X_1,….,etc) and also the values of sin(X) and cos(X) are calculated manually and then the calculated values are saved in these data registers.

2- The Third stage using multiplier of core generator was shown in figure (10). Figure (11) shows merge step 1 and step 2 and The complete FFT algorithm design is shown in the figure (12).
In this step we used the multiplier that shown in the figure (13) to implement this stage , and it was put in the schematic editor.

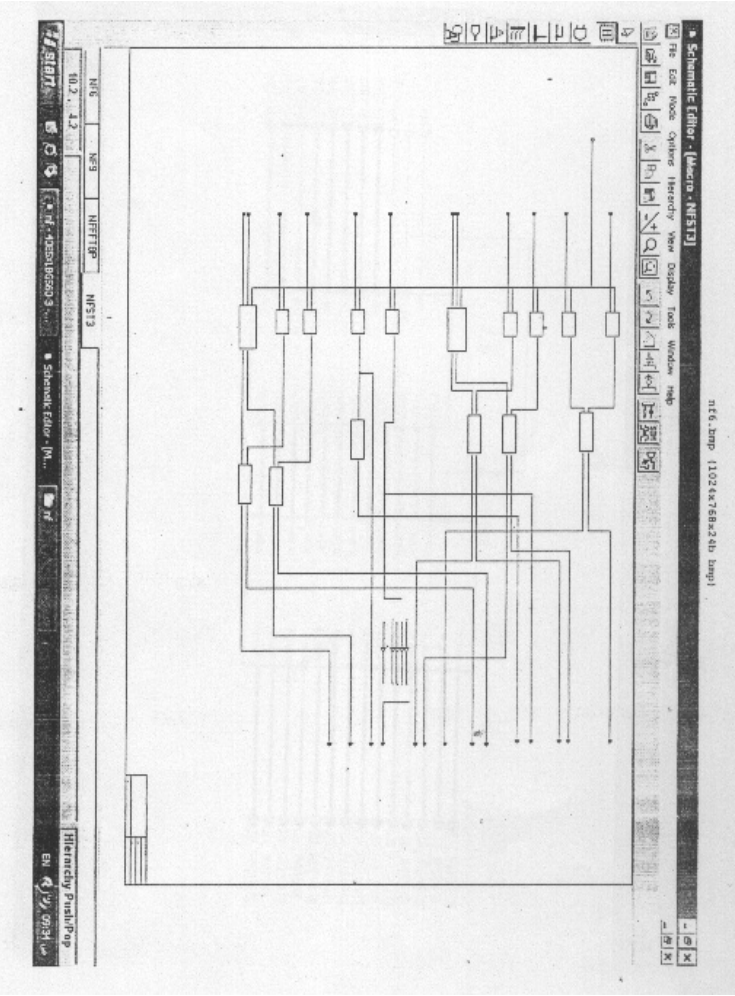

Fig (10) Schematic diagram of third stage of FFT

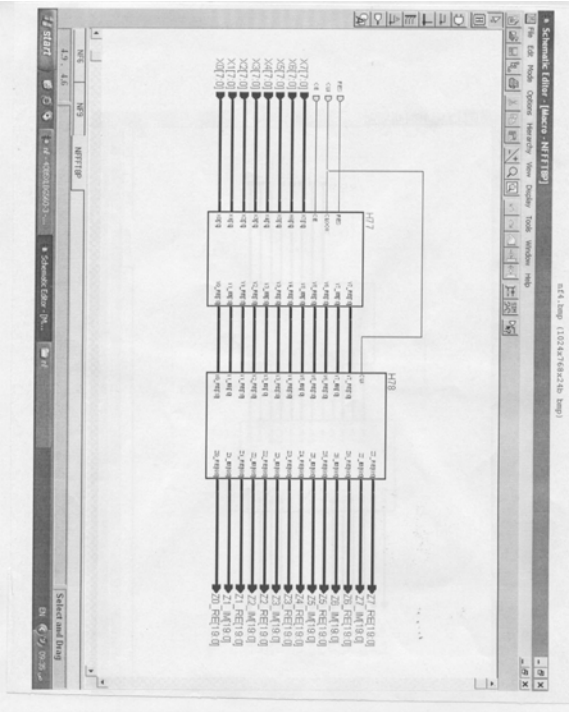

Fig (11) Merge first & Second Steps of FFT

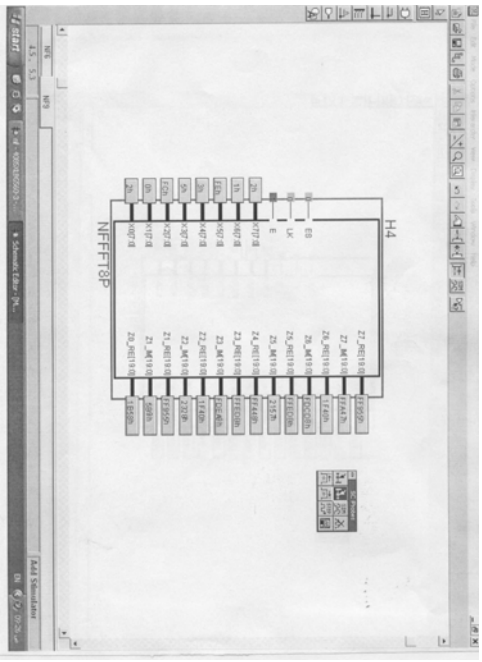

Fig (12) Complete FFT Design

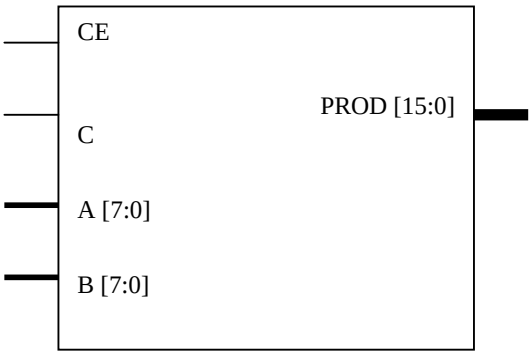

Fig. (13): Symbolic representation of the multiplier.

Where: -

CE: Is the clock enable signal pin.

C: Is the master clock applied to the system.

A [7:0]: Is the 8-bit input vector pins.

B [7:0]: Is the 8-bit input vector pins.

PROD [15:0]: Is the 16-bit output of the multiplier.

The multiplier was tested by using simulation tools. The arbitrary input vectors were applied to the input and the resultant output was recorded.

3- The FFT algorithm tested by insert numbers in HEXA format to the input and take the result from the output as shown in the figures (14,15).In these two figures two different frequencies are used and the input numbers represents 8 bits (0-7) and the outputs (real and imaginary) will displayed as 20 bits (0-19) as shown in these two figures. In the figure (14) the frequency was taken from B6 and it is equal to 1.5MHz, and in the second figure (15) it is taken from B7 and it is equal to 0.78 MHz.

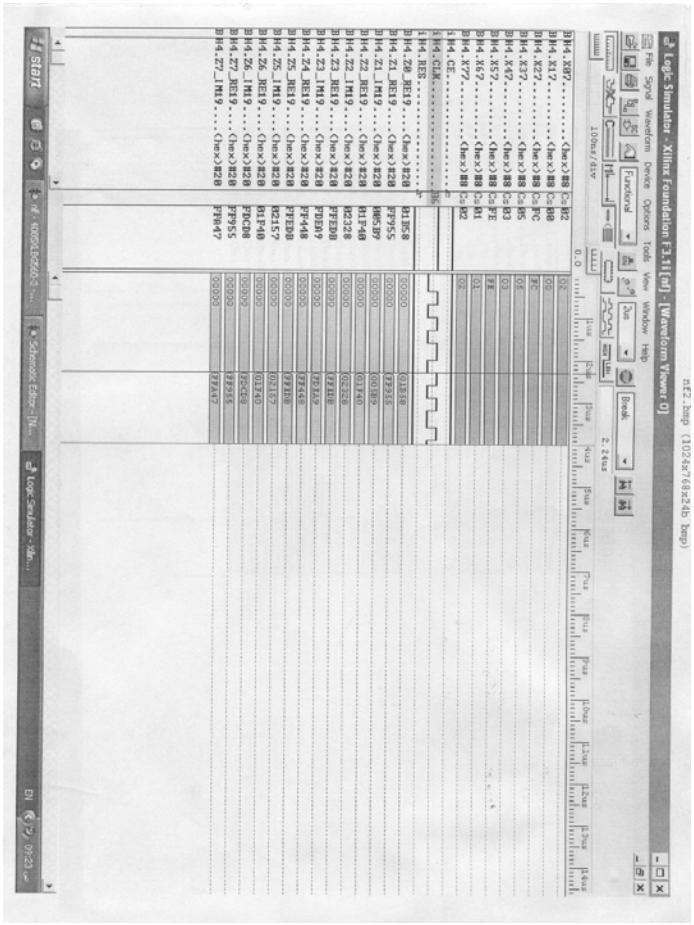

Fig (14) Results of FFT algorithm When B6 equal to 1.5MHz

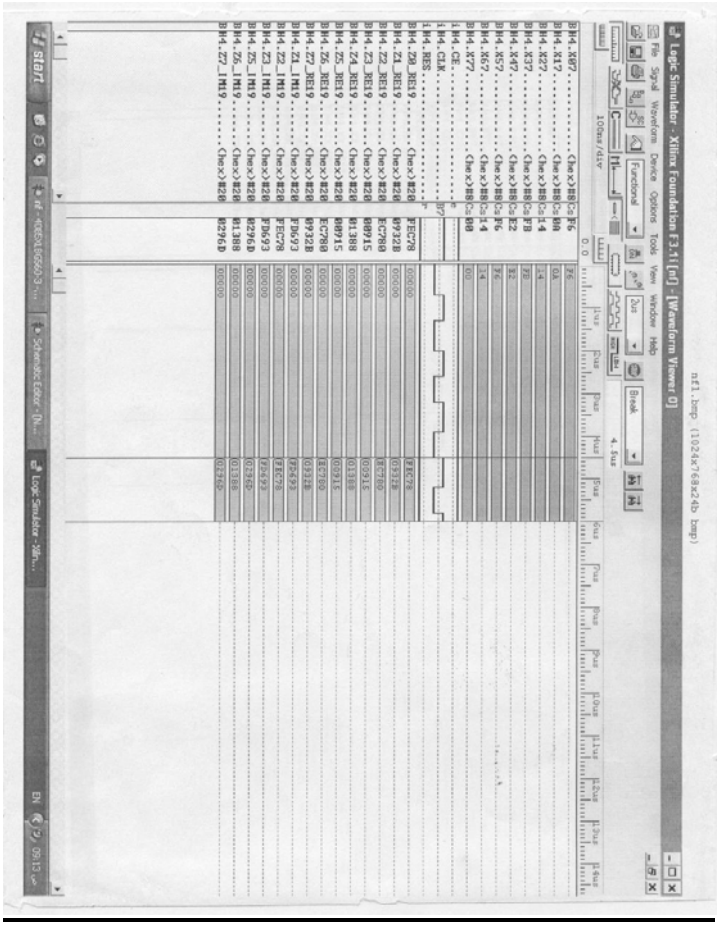

Fig (15) Results of FFT algorithm When B7 equal to 0.78MHz

9- Conclusions:

The purpose of this paper has been to develop an understanding of the underlying principles in FFT implementation with the FPGA technique as an alternate approach to the general or special purpose microprocessors. Implementation of 8- point FFT algorithm using FBGA technique seems to be easy and simple compared with other techniques. Also it is possible to implement another applications or algorithms using this approach in the field of single or image processing. Communications systems, and electronic circuit design… etc.

10- References

[1] Bracewell, R. N." The Fourier Transform and its applications", the McGraw- hill Companies, Inc, 2000, ISBN: 0-07-303938-1.

[2] Cartwright, M," 'Fourier Methods for mathematicians, scientists and engineers" , Ellis Hoi- wood Limtied, 1990, ISBN: 0-13-327016.

[3] Bob Zeidman," Introduction to CPLD & FPGA Design", www.chalknet.com.

[4] J.L. Durka & J.B.Givet," Innovative Technologies and Designs, ASIC development and validation in safety components" In INERIS &JAY Electronique, 2000.

[5] R. Venkatesan ," FPGA Implementation of RNS Structures", Ms.C thesis ,Dec.1994.

[6] J.Petrone," Adaptive Filter Architctures for FPGA Implementation," Ms.C thesis 2004.

[7] M.Kumar, A.Jayram, and B.Kamakoti," SHAPER: Synthesis for Hybrid FPGA Architectures Containing PLA Elements Using Reconvergence Analysis", India, Email: kama@iitm.ernet.in

[8] Andraka Consulting Group Inc., "FPGA Basics", article at www.andraka.com, 2002.

[9] Ch.K. J an ," VHDL Implementation of Systolic Modular Multiplications and Cryptosystem" ,Ms.C thesis 2001.

[10] P. Hallschmid," Embeded Programmable IP Cores", Ms.C thesis ,Jan 2003.

[11] J.Ortiz," Hardware Filter Architectures for FPGA Implementation", Ms.C thesis ,May 2004.

[12] Xilinx Inc.: 'Xilinx Virtex- E Data book', http:// www. Xilinx.com, 2000-2001, (Ace 2001-02-05).

[13] Elanix Inc., "FPGA Architect- Xilinx", Elanix product manual, www.elanix.com, 2002.

[14] P.Papamichalls and John So," Implementation of Fast Transform Algorithim with the TMS32020", Application of TMS320 Family handbook, Texas Instruments, 1988.