



**Design of 8-Point Radix-2 DIF FFT
Algorithm Using a Modified Multiplier and
Adder Unit**



A PROJECT REPORT

Submitted by

SIDHARTH PRABUKUMAR

Reg. No.: 1110107092

VINEETH. J

Reg. No.: 1010107118

HARIHARAN. S

Reg. No.: 1110107306

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION

ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE-641049

(An Autonomous Institution Affiliated to Anna University, Chennai)

APRIL 2015

**KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE-641049**

(An Autonomous Institution Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report titled “**DESIGN OF 8-POINT RADIX-2 DIF
FFT ALGORITHM USING A MODIFIED AND IMPROVED MULTIPLIER
AND ADDER UNIT**” is the bonafide work of “**SIDHARTH PRABUKUMAR,
VINEETH. J AND HARIHARAN. S**” who carried out the project work under
my supervision.

SIGNATURE

Ms. G. Amirtha Gowri

Supervisor
Professor, Department of ECE
Kumaraguru College of Technology,
Coimbatore 641049

SIGNATURE

Dr. Rajeswari Mariappan

Head Of The Department
Department of ECE
Kumaraguru College of Technology,
Coimbatore 641049

The candidates with Register numbers 1110107092, 1010107118 and
1110107306 are examined by us in the project viva-voce examination held on
.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First we would like to express our praise and gratitude to the Lord, who has showered his grace and blessing enabling us to complete this project in an excellent manner. He has made all things in beautiful in his time.

We express our sincere thanks to our beloved Joint Correspondent, **Shri. Shankar Vanavarayar M.B.A., PGD.**, for his kind support and for providing necessary facilities to carry out the project work.

We would like to express our sincere thanks to our beloved Principal **Dr. R. S. Kumar BE (Hons), M.Tech., Ph.D.**, who encouraged us with his valuable thoughts.

We would like to express our sincere thanks and deep sense of gratitude to our HOD, **Dr. Rajeswari Mariappan M.E., B.Tech Ed., Ph.D.**, for her valuable suggestions and encouragement which paved way for the successful completion of the project.

We are greatly privileged to express our deep sense of gratitude to the Project Coordinator **Ms. A. Kalaiselvi M.E.** Assistant Professor, for her continuous support throughout the course.

In particular, We wish to thank and express our everlasting gratitude to the Supervisor **Dr. G. Amirtha Gowri M.E., Ph.D.**, Associate Professor for her expert counseling in each and every step of our project work and we wish to convey our deep sense of gratitude to all teaching and non-teaching staff members of ECE Department for their help and cooperation.

Finally, we thank our parents and our family members for giving us the moral support in all of our activities and our dear friends who helped us to endure our difficult times with their unfailing support and warm wishes.

CHAPTER NO	TITLE	PAGE NO
	LIST OF FIGURES	v.
	ABSTRACT	vii.
1	INTRODUCTION	1
	1.1 Fast Fourier Transform	1
	1.2 Multiplier Design	3
	1.3 Adder Design	9
	1.4 Fast Fourier Transform Implementation	12
2	HARDWARE DESCRIPTION	16
	2.1 ALTERA de0 Board	16
3	SOFTWARE DESCRIPTION	20
	3.1 ALTERA Quartus II 64 Bit	20
	3.2 ModelSim ALTERA Starter Edition 6.4a	22
4	SIMULATION AND RESULTS	26
	4.1 Design Simulation	26
	4.2 Power Analysis and Fitter Report	27
	4.3 Result	29
5	CONCLUSION	30
6	PUBLICATION	31
7	REFERENCES	32

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
1	Comparison of real and complex DFT	1
2	Signal flow graph for 8 point DIT-FFT with input scrambling	3
3	Signal flow graph for 8 point DIF-FFT with output scrambling	4
4	RTL Schematic of Full Adder	5
5	CSA Architecture Multiplier	6
6	Shannon Based Full Adder Cell	7
7	RTL schematic of the proposed full adder cell	9
8	The Carry Select Adder Construction by Sharing the Common Boolean Logic Term	10
9	RTL Schematic of 8-Point DIF FFT Algorithm	12
10	Butterfly Unit	12
11	Block Diagram of FFT Algorithm	15
12	The de0 Board	16
13	Block Diagram of EP3C16F484 FPGA	18
14	LCD Module on the de0 Board	19
15	Connections between the LCD module and Cyclone III FPGA	29
16	Typical CAD Flow	20
17	Tool Structure and Flow	22
18	Test Bench Input for FFT	26
19	Test Bench Output for FFT	26
20	Fitter Report of our Proposed Design	28

21	Fitter Report of the Conventional Design	28
22	Power Report for our Proposed Design	29
23	Power Report of the Present Design	29

ABSTRACT

The Fast Fourier Transform (FFT) is an efficient algorithm for computing the Discrete Fourier Transform (DFT) and requires less number of computations than that of direct evaluation of DFT. It has several applications in signal processing.

However, there are two problems. One related to the algorithmic point of view and the other based on ASIC architecture. The last one was pushed by VLSI technology evolution.

In this project, we implement 4 point Radix 2 DIF-FFT algorithm using a modified multiplier and adder unit. Multiplier and adder units are vital in digital signal processors, microprocessors or any device which involves predominant and continuous use of multiplication and addition schemes.

Around 70% of the operations in RISC processors make use of addition and multiplication in its data path. This brings about the importance of the MAC unit in processors and the need for optimization with regards to power, area and delay. In this project, we focus on reducing the power consumption of the multiplier and adder without compromising too much on the delay.

CHAPTER-1

INTRODUCTION:

1.1 The Fast Fourier Transform

J.W. Cooley and J.W. Tukey are given credit for bringing the FFT to the world in their paper: "An algorithm for the machine calculation of complex Fourier Series," *Mathematics Computation*. The FFT is based on the *complex DFT*, a more sophisticated version of the *real DFT*. These transforms are named for the way each represents data, that is, using complex numbers or using real numbers.

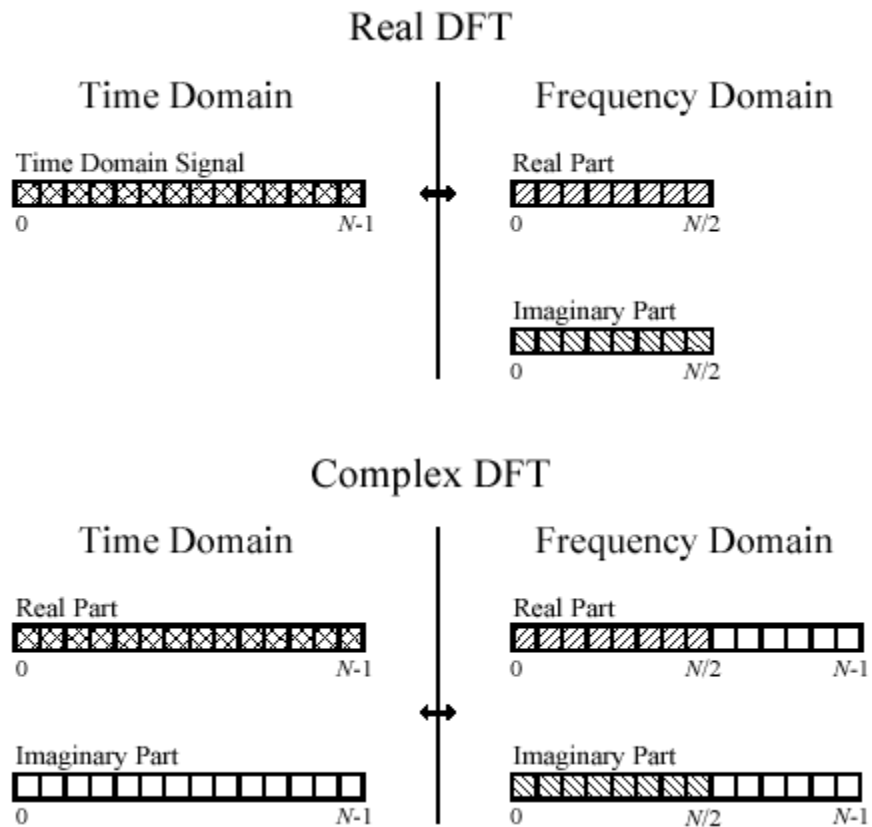


Figure-1. Comparison of real and complex DFT

1.1.1 Comparison of Real DFT and Complex DFT

Since the FFT is an algorithm for calculating the complex DFT, it is important to understand how to transfer *real DFT* data into and out of the *complex DFT* format. The real DFT transforms an N point time domain signal into two point frequency domain signals. The time domain $N/2 + 1$ signal is called just that: the *time domain signal*. The two signals in the frequency domain are called the *real part* and the *imaginary part*, holding the amplitudes of the cosine waves and sine waves, respectively. In comparison, the complex DFT transforms two N point time domain signals into two N point frequency domain signals. The two time domain signals are called the *real part* and the *imaginary part*, just as are the frequency domain signals. In spite of their names, all of the values in these arrays are just ordinary numbers. Suppose there is an N point signal, and we need to calculate the *real DFT* by using the FFT, then set all of the samples in the imaginary part to *zero*. Then, move the N point signal into the real part of the complex DFT's time domain, and compute DFT using the FFT. The result is a real and an imaginary signal in the frequency domain, each composed of N points. Samples 0 through $N/2$ of these signals correspond to the real DFT's spectrum.

1.1.2 FFT

The FFT is a complicated algorithm, and its details are usually left to those that specialize in such things. This section describes the general operation of the FFT. The FFT operates by decomposing an N point time domain signal into N time domain signals each composed of a single point. The second step is to calculate the N frequency spectra corresponding to these N time domain signals. Lastly, the N spectra are synthesized into a single frequency spectrum. There are basically two

algorithms in FFT. One is called DIT(Decimation in time) and the other DIF(Decimation in frequency). In the DIT approach, the initial DFT is divided into two transforms, one consisting of a transform of even samples and the other consisting of a transform of odd samples. This process is carried out until the initial transform is reduced to a set of two-point transforms of the initial data. An in-place FFT implementation allows the results of each FFT butterfly to replace its inputs. In order to use an in place algorithm it is necessary either to re-order the input data array or re-order the output array. This re-ordering is simply arranged by reversing the address bits. Before starting to calculate the DFT, the input data is ordered such that its address is bit-reversed, that is if the binary address of the required sequence of data is 110 then the bit reversed version on that becomes 011. Given below is the signal flow graph for the DIT (figure2)

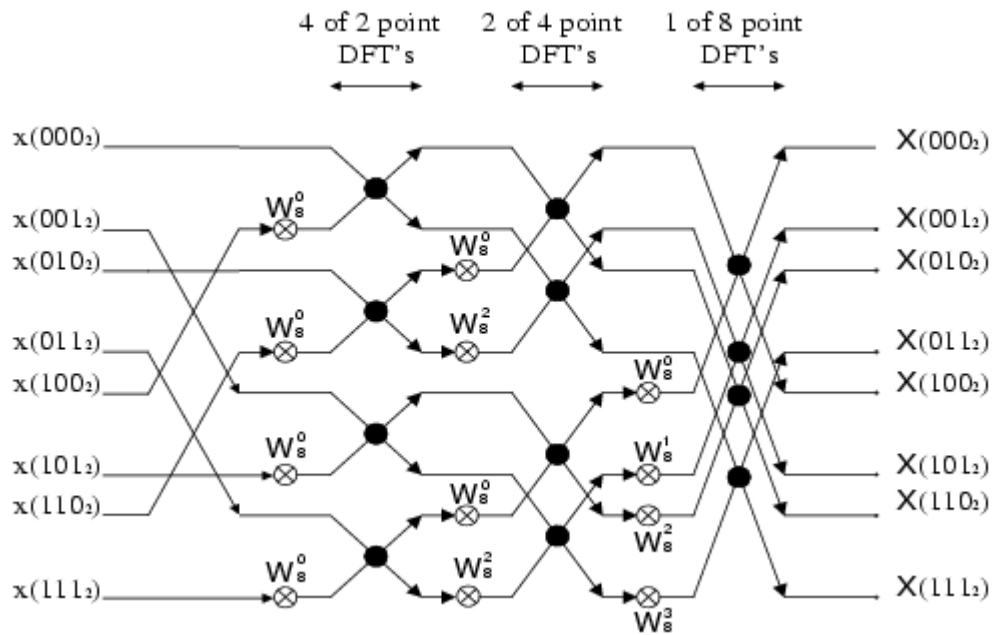


Figure-2. Signal flow graph for 8 point DIT-FFT with input Scrambling

This signal flow graph consists of a number of butterflies. Each butterfly takes a pair of input data values A and B and outputs A1 and B1 as shown below. The

input data is multiplied by the twiddle factor W_N^k . The solid dots represent addition\subtraction shown in figure3.

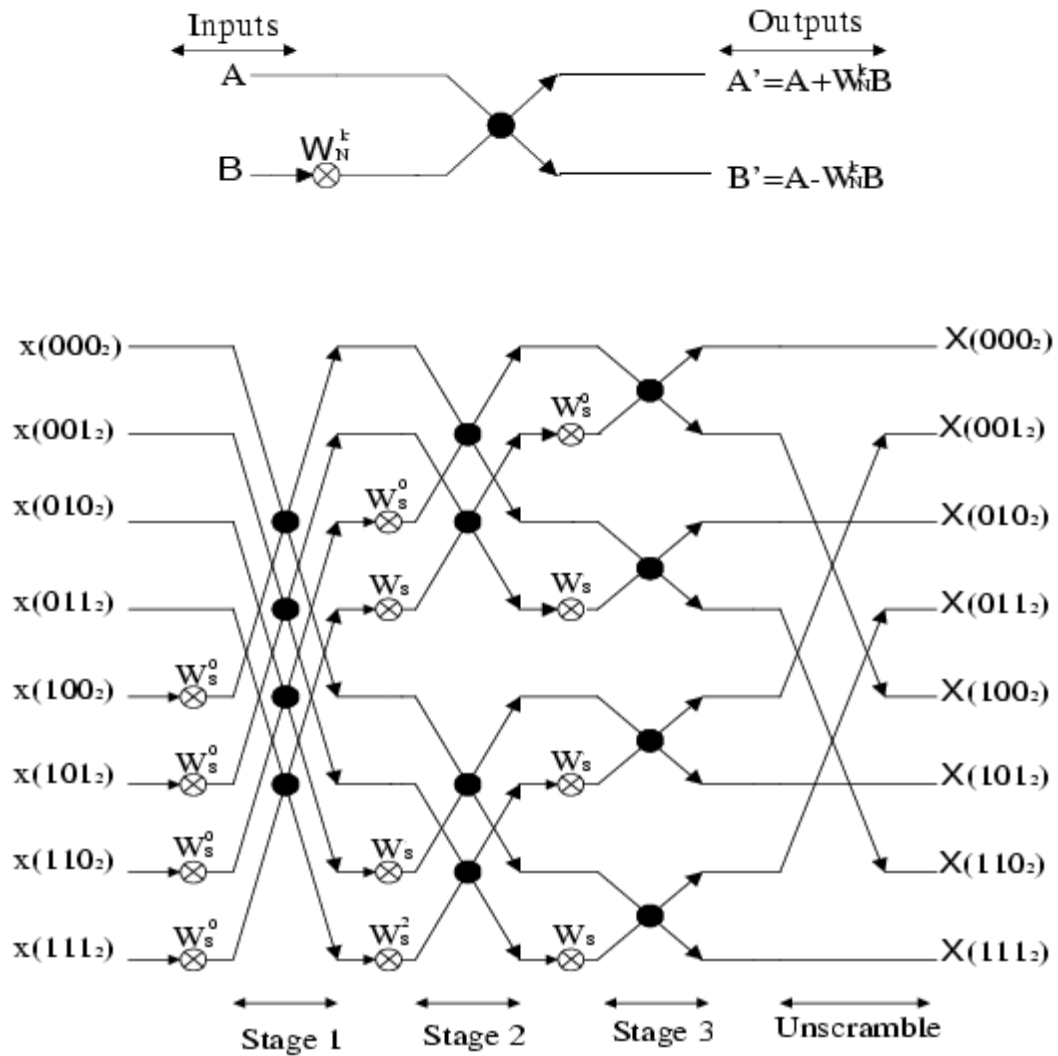


Figure-3. Signal flow graph for 8 point DIF-FFT with output scrambling

1.2 Multiplier Design:

The multiplier used in this project is a power optimized multiplier using Shannon based multiplexing logic which was first introduced by P. Karunakaran et al. Here, a novel design for a full adder is done based on Shannon's multiplexing logic.

Our multiplier is a Carry Save Array multiplier where the individual full adder and half adder cells are implemented using the proposed adder design. From the results depicted, the proposed adder has minimum area and power consumption when compared to existing adders. The proposed full adder cell has the following characteristic equation:

$$\text{Sum} = ((A \text{ xor } B).C') + ((A \text{ xor } B)'.C) \quad \text{Carry} = ((A \text{ xor } B).C) + ((A \text{ xor } B)'.A)$$

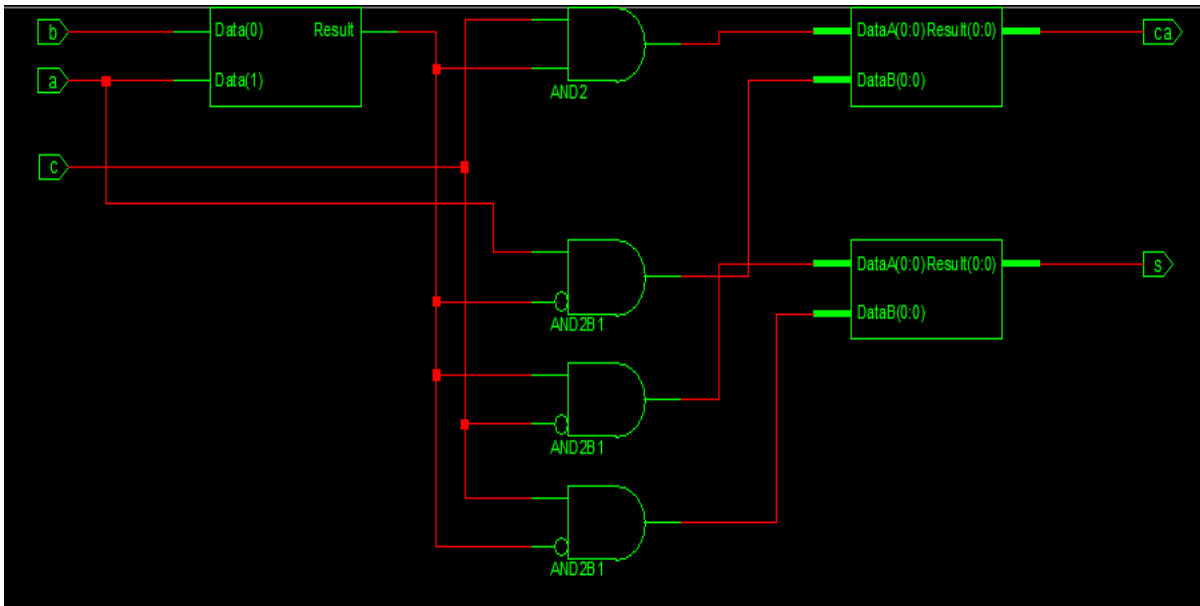


Figure-4. The RTL schematic of the proposed full adder cell is shown.

The half adder cells are also designed based on the Shannon multiplexing logic. The 'Sum' bit of the cell remains the same, while the equation for the carry bit is changed. It is given as, $\text{Carry} = (A.B) + (B.B')$. This model of the full adder and

half adder cells are used in the design of the carry save array multiplier whose architecture is shown.

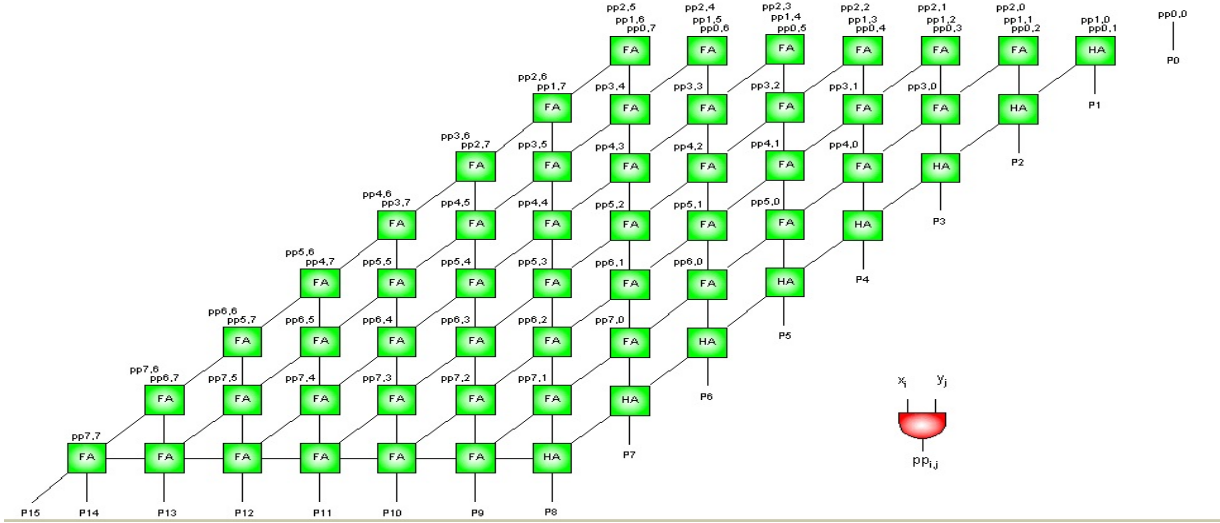


Figure-5. Multiplier design

In this project, an 8-bit multiplier is used. The Carry save Array (CSA) multiplier is a linear array multiplier. The linear multiplier propagates data down through the array cell. Each row of the CSAs adds one additional partial-product to the partial sum. As the operand size increases, linear arrays grow at a rate equal to the square of the operand size because the number of rows in the array is equal to the length of the multiplier, and the width of each row is equal to the width of multiplicand.

1.1.4 Shannon Based Multiplexing Logic:

The proposed Shannon full adder circuit as shown combines the multiplexing operation for the sum operation and the Shannon Theorem for the carry operation; the sum and carry circuits are designed based on Standard full adder equations. An input C and its complement are used as the control signal of the sum circuit. The two-input X-OR gate is developed using the multiplexer method. The output node

of the two-input multiplexer circuit is the differential node. According to standard full adder equation, the sum circuits need three inputs. In order to avoid increasing the number of transistors due to the addition of a third input, the following arrangement is made, the CPL X-OR gate multiplying with C's complement input and EX-NOR gate is multiplied with input C, and thereby reducing the number of transistors in the sum circuit.

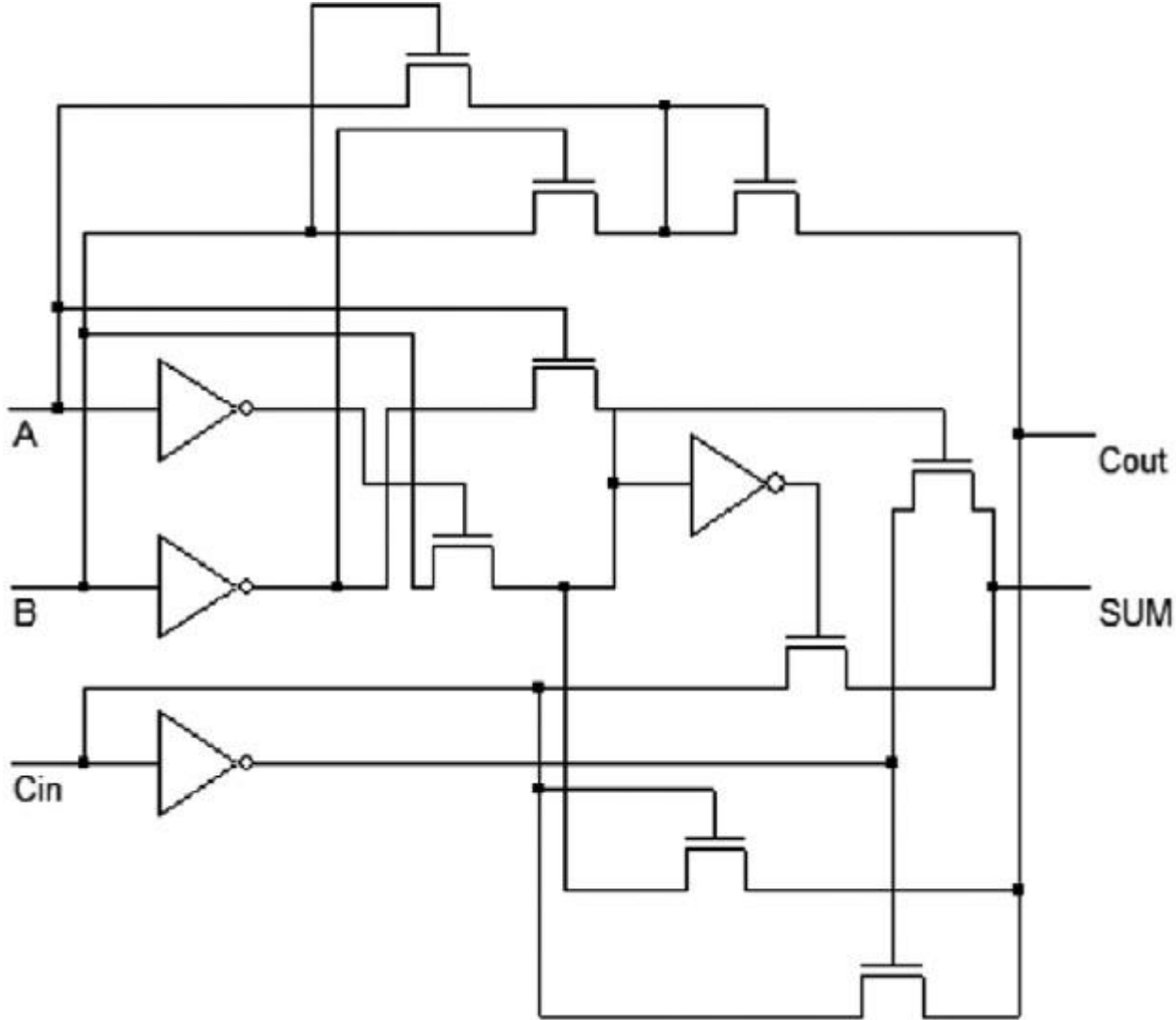


Figure-6. Shannon Based Full Adder Design

The carry for the half adder is given by,

Half Adder

$$\text{Carry} = A.B$$

Shannon's Theorem

$$\text{Carry} = (A.B) + (B.B')$$

Full Adder

$$\text{Sum} = A \text{ xor } B \text{ xor } C$$

$$\text{Sum} = ((A \text{ xor } B).C') + ((A \text{ xor } B)'.C)$$

$$\text{Carry} = (A.B) + (B.C) + (C.A)$$

$$\text{Carry} = (A+B) C + (A.B)$$

Existing adder

$$\text{Carry} = (A+B) C + (A.B) + (B'.C')$$

1.3 Adder Design:

For the full adder design, we made use of an area efficient carry select adder by sharing a common Boolean logic term which was first introduced by I-Chyn Wey et al. According to them, by utilizing the multiplexer to select the correct output according to its previous carry-out signal, we can still preserve the original characteristics of the parallel architecture in the conventional carry select adder. By sharing the common Boolean logic term, the duplicated adder cells can be removed in the conventional carry select adder. In this way, the circuit area and transistor count can be greatly reduced and power delay product of the adder circuit can be also greatly lowered. Thus, this design will optimize power and delay. The RTL schematic of the proposed full adder cell of the carry select adder design is shown in figure7.

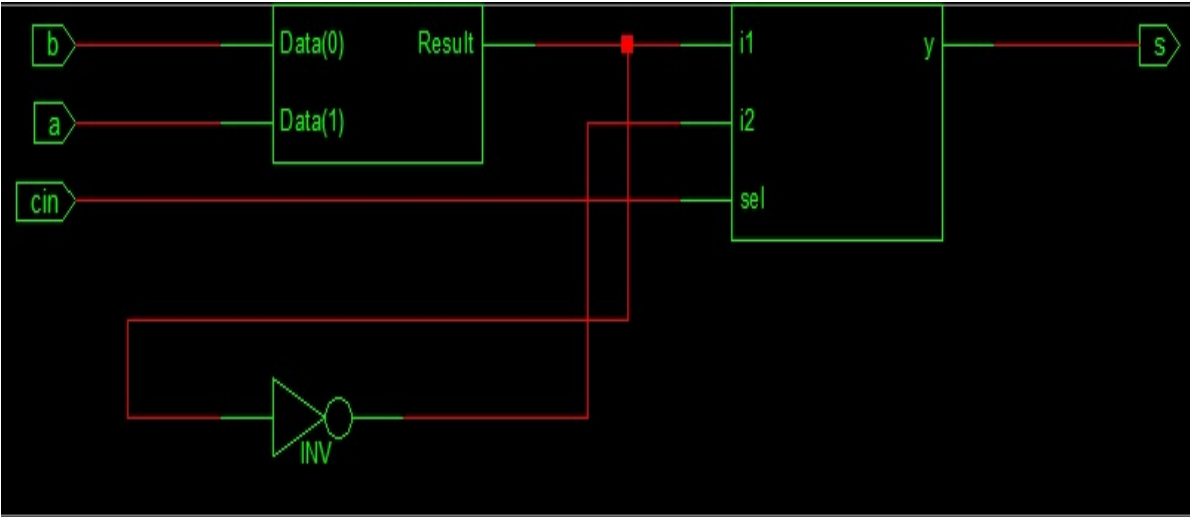


Figure-7. CSA Full Adder Cell

This implementation of the full adder cell figure 8, when extended to n-bits, will give rise to a structure similar to the one below. In our implementation, we made

use of a 16-bit Carry Save adder.

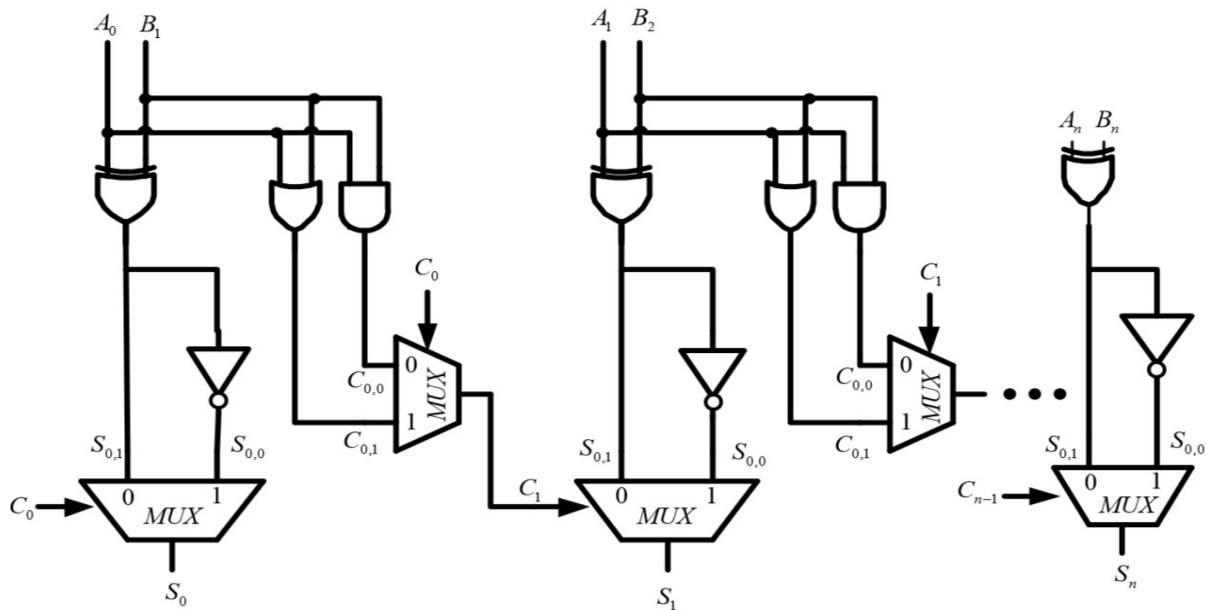


Figure-8. The Carry Select Adder Construction by Sharing the Common Boolean Logic Term

To share the common Boolean logic term, we only need to implement one XOR gate with one INV gate to generate the summation signal pair. As the carry-in signal is ready, we can select the correct summation output according to the logic state of carry-in signal. As for the carry propagation path, we construct one OR gate and one AND gate to anticipate possible carry input values in advance. Once the carry-in signal is ready, we can select the correct carry-out output according to the logic state of carry-in signal. In this way, we can keep both the summation generation circuit of XOR gate and INV gate and the carry-out generation circuit of OR gate and AND gate in parallel. Since we still retain part of parallel architecture of conventional carry select adder, we can still maintain some competitiveness in speed. On the other hand, we needn't to prepare the duplicated

adder cells in the conventional carry select adder, which can greatly reduce the transistor count and lower the power consumption.

In the proposed carry select adder, we trade-off transistor count with speed to achieve a lower power-delay product. In the N-bit carry ripple adder, the delay time can be expressed as:

$$TCRA = (N-1)T_{carry} + T_{sum} \quad (1)$$

In the N-bit carry select adder, the delay time is:

$$TCSA = T_{setup} + (N/M)T_{carry} + MT_{mux} + T_{sum} \quad (2)$$

In this carry select adder, the delay time is:

$$T_{new} = T_{setup} + (N-1)T_{mux} + T_{sum} \quad (3)$$

As compared with the conventional carry select adder, our speed is a little slower since the parallel path in our design is shorter. However, we can achieve lower area, lower power consumption, and lower PDP. As compared with the carry ripple adder, our speed can be faster because some of the parallel architecture in the conventional carry select adder is retained. The delay time in our proposed adder design is also proportional to the bit number N; however, the delay time of multiplexer is shorter than that of full adder. Consequently, our area-efficient adder can perform with nearly the same transistor count, nearly the same power consumption, but with faster speed and lower PDP as compared with the carry ripple adder.

1.4 Fast Fourier Transform Implementation:

The RTL schematic of the 8 point FFT algorithm is shown below.

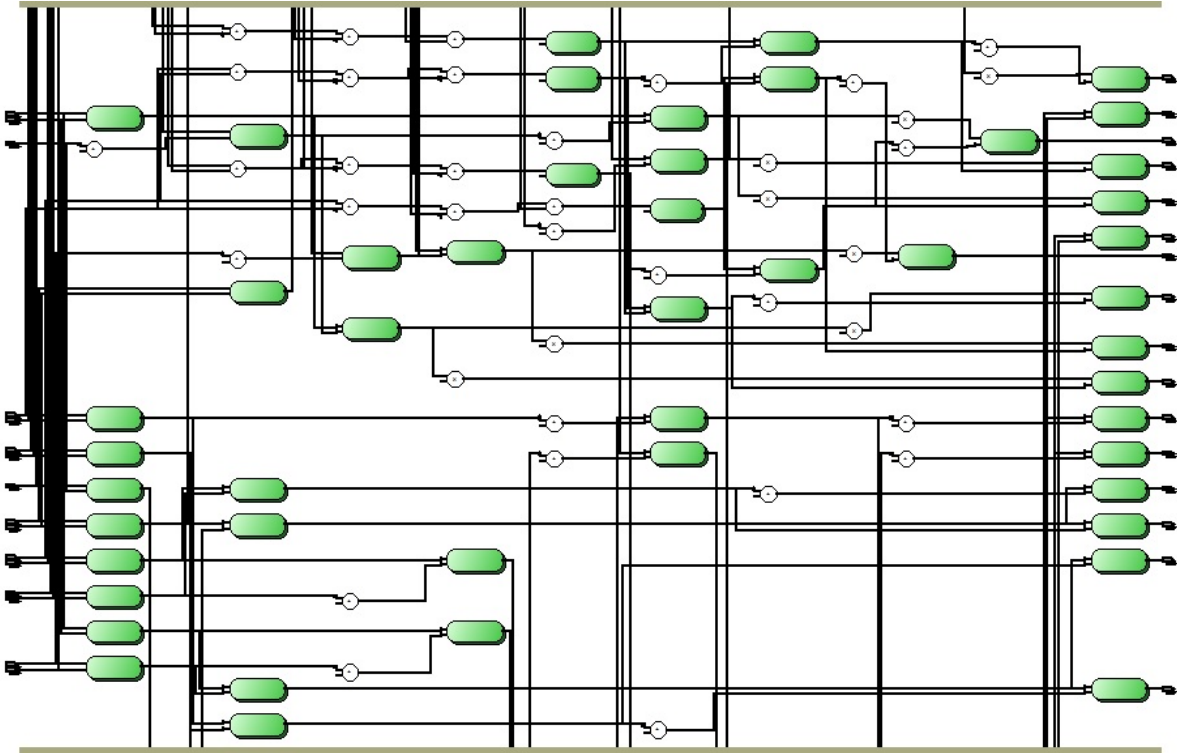


Figure-9. RTL Schematic of 8-Point DIF FFT Algorithm

1.4.1. Butterfly Unit:

The basic module for implementation is butterfly module which is shown

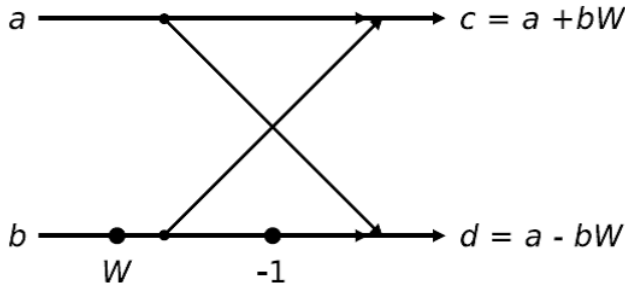


Figure-10. Butterfly Unit

There are two inputs called a, b and two outputs c, d and twiddle factor W. Output is as shown as follows:

$$C = a + bW$$

$$D = a - bW$$

With these butterfly units we can build the whole FFT structure. If N is the input for FFT, each stage requires $N/2$ butterflies. As we can see from the figure that one butterfly unit requires 1 complex multiplier and 2 adders for executing single butterfly. For every DIF-FFT radix -2 algorithm with N input sequence, there is a requirement of $N/2$ multipliers and N adders.

1.4.2. Number Representation:

For number representation of both real as well as imaginary, fixed point scheme is followed so that we can reduce the complexity of using floating point arithmetic. The twiddle factors used are in complex form real and imaginary. To represent this number we multiply these numbers by scaling factor which is 100 in our case. So that twiddle factor is rounded off to an integer number. For complex multiplication we require twiddle factor magnitude and sign bit so $s+1$ bits are required to represent twiddle factor. As the input given to the design can also be in floating form then we can apply the same scheme of rounding off input. As we scale the input and twiddle factor, we have to scale down the signals at the output which leads to rounding off errors. Simple way for scaling down is by multiplying or using shifting operation.

1.4.5. Complex Multiplier:

Most tedious part in FFT is the complex multiplication. Complex numbers are divided into two parts real and imaginary. Say $a_r + jb$ is a complex number which is again multiplied by complex number $c_r + jd$.

$$a_r + jb$$

$$c_r + jd$$

Multiplying these equations we will get

$$(ac - bd) + j(bc + ad)$$

Another method for complex multiplication is shift and add for non-trivial twiddle factor multiplication. In radix-2 8 point FFT algorithm, the twiddle factor multiplication with $W_2^8 = -j$ and factors is trivial, multiplication with easily can be done by exchanging real to imaginary part and vice versa, by changing the sign of real and imaginary numbers. For other twiddle factors, we require complex multiplication. In the case of 8 point FFT non trivial twiddle factors $W_1^8 = 0.707 - j0.707$, $W_3^8 = -0.707 - j0.707$, both these twiddle factors have 0.707 number common in it. Because of this, we can easily reduce the multiplicative complexity. Our implementation is done with only two complex multiplications.

The block diagram of our FFT algorithm is shown in figure 11.

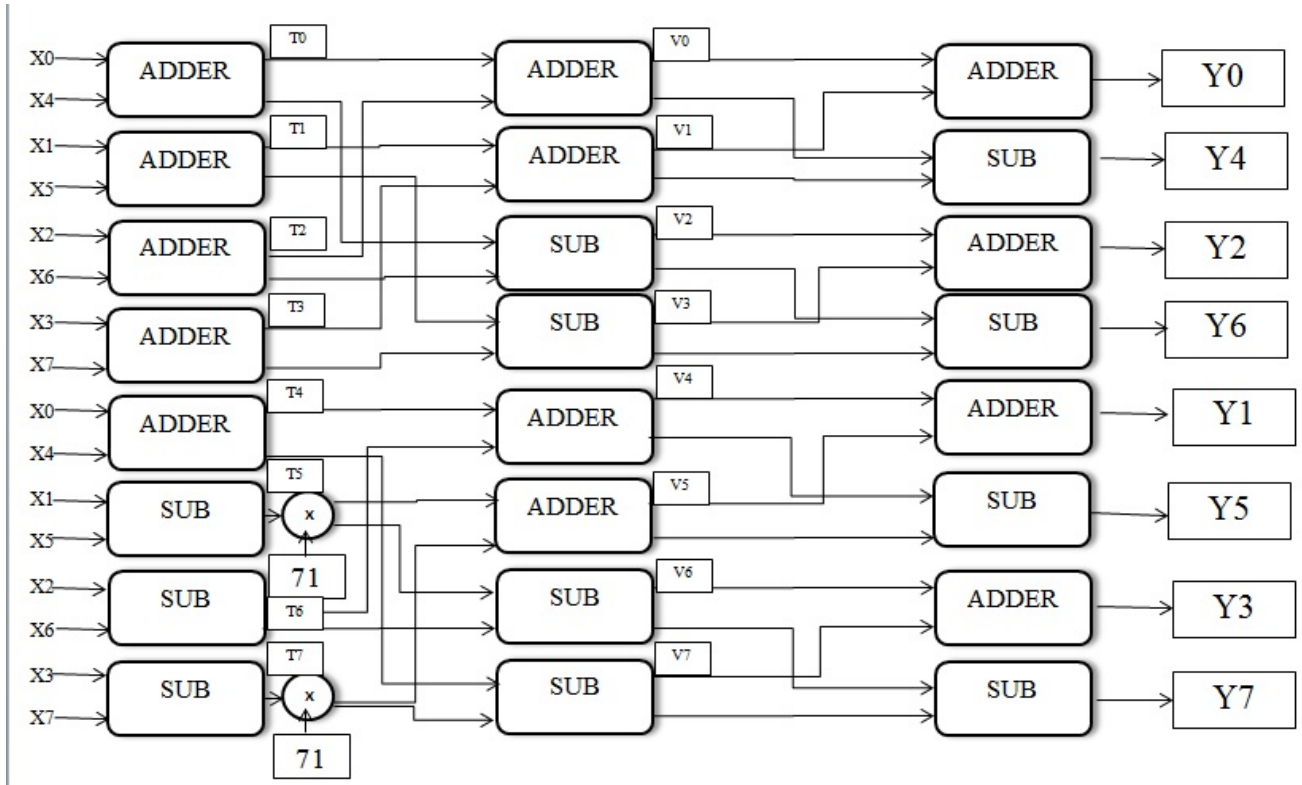


Figure-11. Block diagram of our FFT Algorithm

CHAPTER 2

HARDWARE DESCRIPTION:

2.1 Altera DE0 Board:

2.1.1. Layout and Components:

A photograph of the DE0 board is shown in Figure12. It depicts the layout of the board and indicates the location of the connectors and key components.

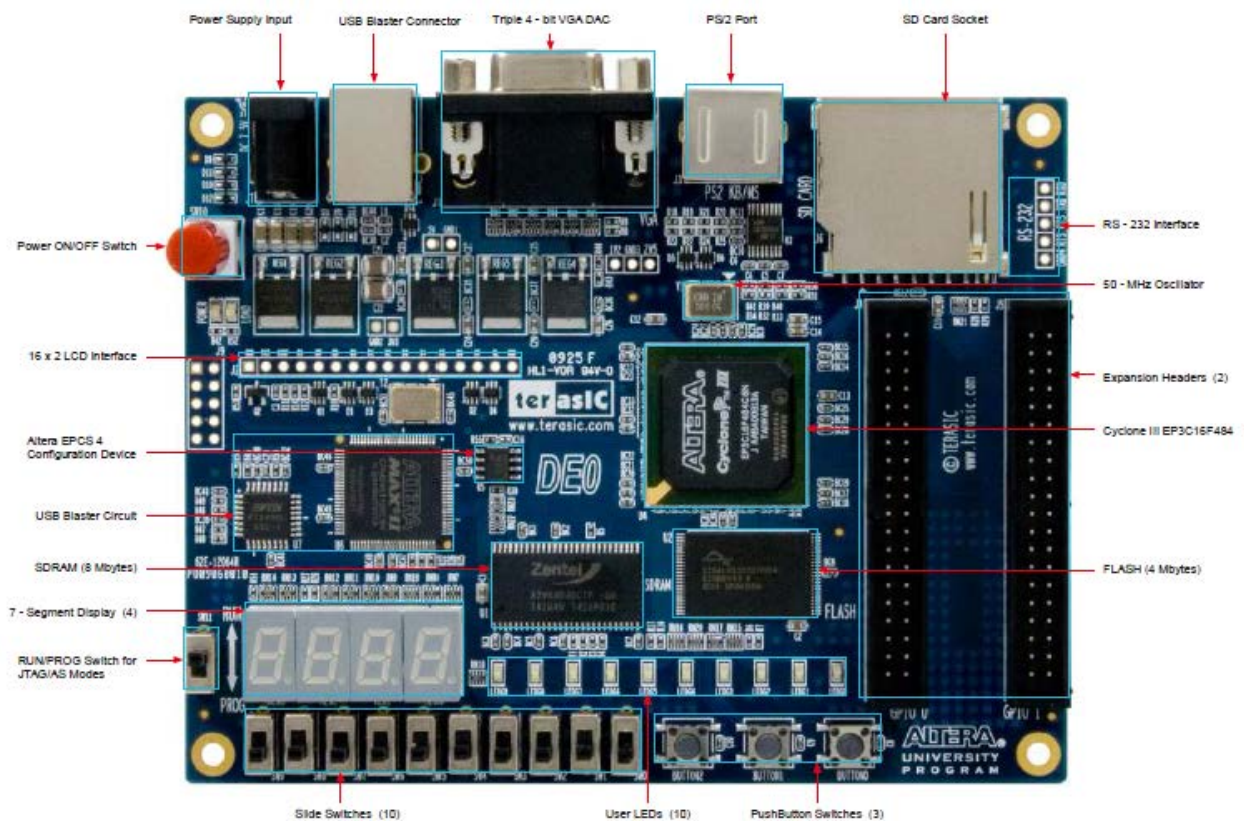


Figure-12. The DE0 board.

The DE0 board has many features that allow the user to implement a wide range of designed circuits, from simple circuits to various multimedia projects. The following hardware is provided on the DE0 board:

Altera Cyclone® III 3C16 FPGA device

Altera Serial Configuration device – EPCS4

USB Blaster (on board) for programming and user API control; both JTAG and Active Serial(AS) programming modes are supported

8-Mbyte SDRAM

4-Mbyte Flash memory

SD Card socket

3 pushbutton switches

10 toggle switches

10 green user LEDs

50-MHz oscillator for clock sources

VGA DAC (4bit resistor network) with VGA-out connector

RS-232 transceiver

PS/2 mouse/keyboard connector

Two 40-pin Expansion Headers

2.1.2 Block Diagram of the Altera Cyclone III FPGA:

Figure13 gives the block diagram of the DE0 board. To provide maximum flexibility for the user, all connections are made through the Cyclone III FPGA device. Thus, the user can configure the FPGA to implement any system design.

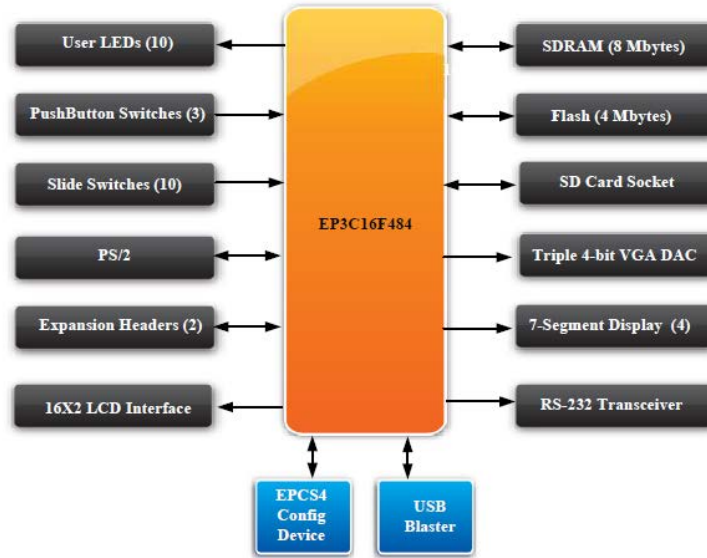


Figure-13. Block Diagram of EP3C16F484 FPGA

2.1.3. Interfacing the LCD Module:

The DE0 board provides a 2x16 LCD interface. In order to use the LCD interface, users are required to solder a LCD module onto the DE0 board shown in figure. The LCD module has built-in fonts and can be used to display text by sending appropriate commands to the display controller, which is called HD44780. A schematic diagram of the LCD module showing connections to the Cyclone III FPGA is given in Figure14.

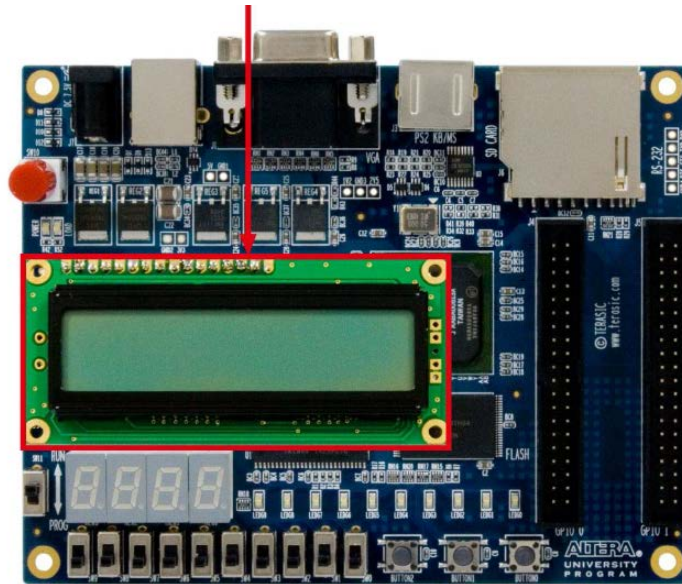


Figure-14. LCD module on DE0 board

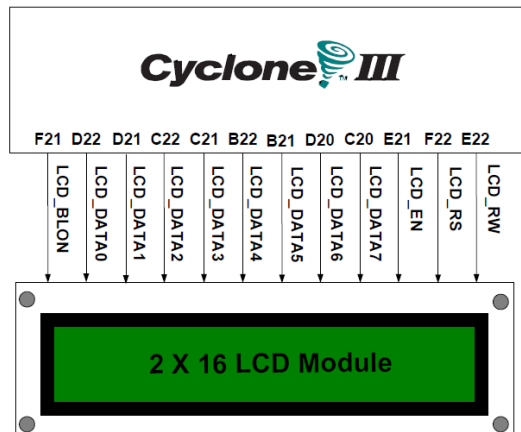


Figure-15. Connections between the LCD module and Cyclone III FPGA

CHAPTER 3

SOFTWARE DESCRIPTION:

3.1. Altera Quartus-II 64-Bit:

Computer Aided Design (CAD) software makes it easy to implement a desired logic circuit by using a programmable logic device, such as a field-programmable gate array (FPGA) chip. A typical FPGA CAD flow is illustrated in Figure 16

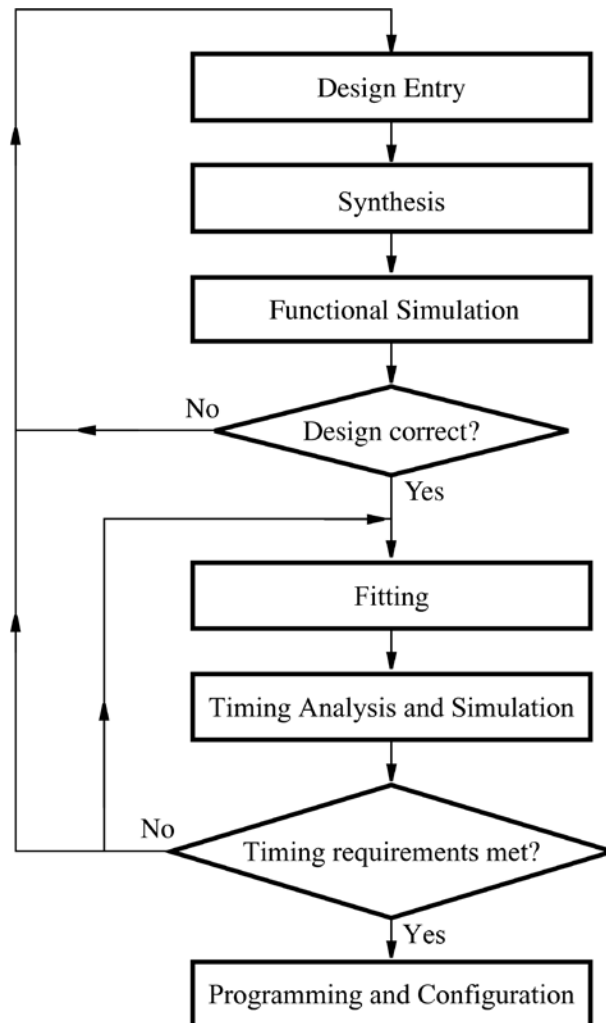


Figure-16. Typical CAD Flow

It involves the following basic steps:

- **Design Entry** – the desired circuit is specified either by using a hardware description language, such as Verilog or VHDL, or by means of a schematic diagram.
- **Synthesis** – the CAD Synthesis tool synthesizes the circuit into a netlist that gives the logic elements (LEs) needed to realize the circuit and the connections between the LEs.
- **Functional Simulation** – the synthesized circuit is tested to verify its functional correctness; the simulation does not take into account any timing issues.
- **Fitting** – the CAD Fitter tool determines the placement of the LEs defined in the netlist into the LEs in an actual FPGA chip; it also chooses routing wires in the chip to make the required connections between specific LEs.
- **Timing Analysis** – propagation delays along the various paths in the fitted circuit are analysed to provide an indication of the expected performance of the circuit

3.2 ModelSim ALTERA STARTER EDITION 6.4a:

3.2.1 Tool Structure and Flow:

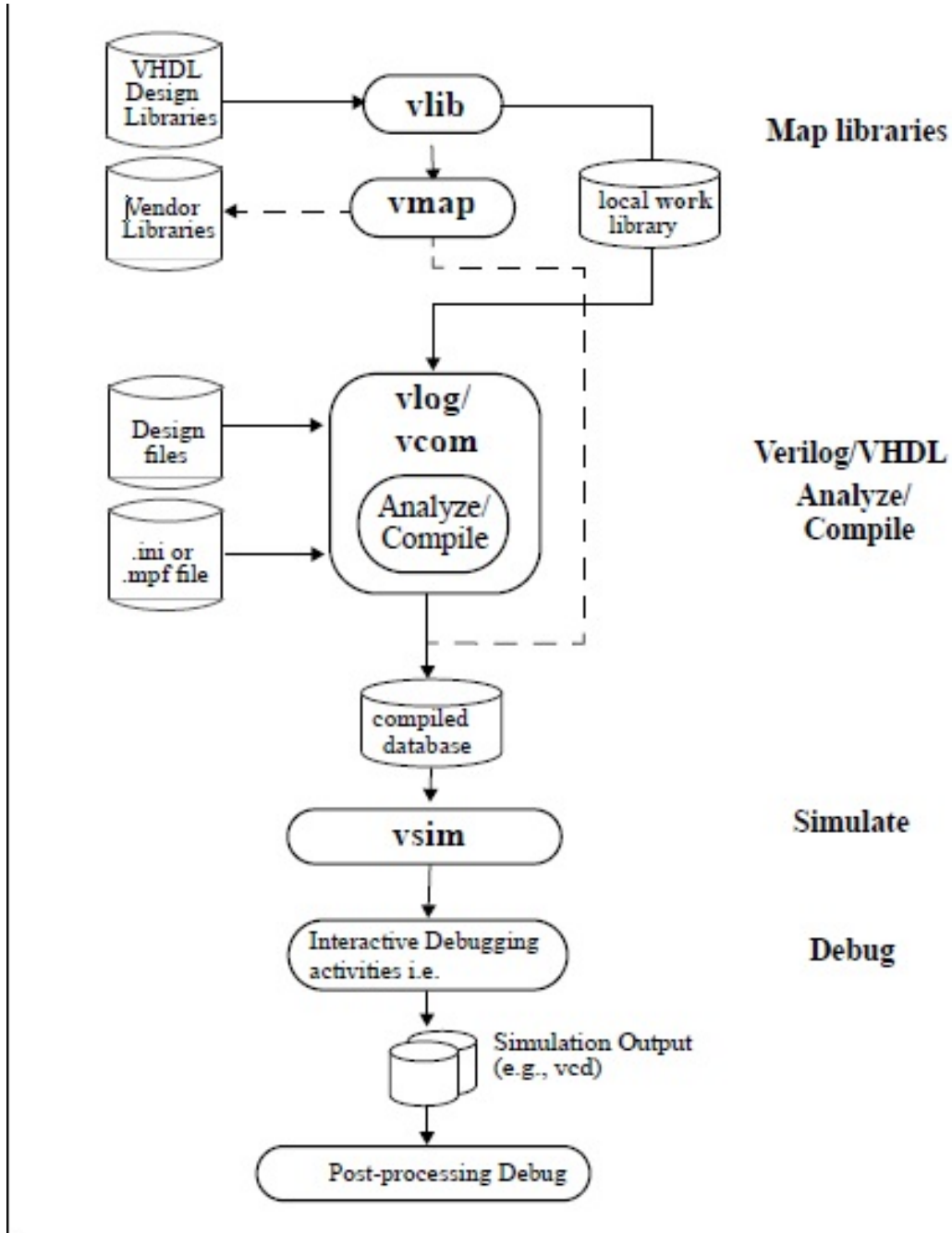








Figure-17. Tool Structure and Flow

3.2.2 Simulation Task Overview

The following table provides a reference for the tasks required for compiling, loading, and simulating a design in ModelSim.

Task	Example Command Line Entry	GUI Menu Pull-down	GUI Icons
Step 1: Map libraries	<code>vlib <library_name></code> <code>vmap work <library_name></code>	a. File > New > Project b. Enter library name c. Add design files to project	N/A
Step 2: Compile the design	<code>vlog file1.v file2.v ...</code> (Verilog) <code>vcom file1.vhd file2.vhd ...</code> (VHDL)	a. Compile > Compile or Compile > Compile All	Compile or Compile All icons:  
Step 3: Load the design into the simulator	<code>vsim <top></code> or <code>vsim <opt_name></code>	a. Simulate > Start Simulation b. Click on top design module or optimized design unit name c. Click OK This action loads the design for simulation	Simulate icon: 
Step 4: Run the simulation	<code>run</code> <code>step</code>	Simulate > Run	Run, or Run continue, or Run -all icons:   
Step 5: Debug the design	Common debugging commands: <code>bp</code> <code>describe</code> <code>drivers</code> <code>examine</code> <code>force</code> <code>log</code> <code>show</code>	N/A	N/A

Simulation Tasks-ModelSim

3.2.3 Basic Steps for Simulation:

Here, a detailed description related to each step in the process of simulating the design using ModelSim is provided.

Step-1- Collecting Files and Mapping Libraries:

Files needed to run ModelSim on your design:

Design files (VHDL and/or Verilog), including stimulus for the design libraries, both working and resource
modelsim.ini (automatically created by the library mapping command

Providing Stimulus to the Design

You can provide stimulus to your design in several ways:

Language based testbench

Tcl-based ModelSim interactive command, force

VCD files / commands

3rd party testbench generation tools

Step-2- Compiling the Design:

Designs are compiled with one of the three language compilers.

Compiling Verilog (vlog)

ModelSim's compiler for the Verilog modules in your design is vlog. Verilog files may be compiled in any order, as they are not order dependent. See Compiling Verilog Files for details.

Compiling VHDL (vcom)

ModelSim's compiler for VHDL design units is vcom. VHDL files must be compiled according to the design requirements of the design. Projects may assist you in determining the compile order

Step-3- Loading the Design for Simulation

vsim top Level Module:

The design is ready for simulation after it has been compiled. Vsim may then be invoked with the names of the top-level modules (many designs contain only one top-level module). For example, if your top-level modules are "testbench" and "globals", then invoke the simulator as follows:

```
vsim testbench globals
```

After the simulator loads the top-level modules, it iteratively loads the instantiated modules and UDPs in the design hierarchy, linking the design together by connecting the ports and resolving hierarchical references.

Using SDF:

You can incorporate actual delay values to the simulation by applying SDF back-annotation files to the design.

Step-4- Simulating the Design

Once the design has been successfully loaded, the simulation time is set to zero, and **run** command must be entered to begin simulation.

The basic simulator commands are:

- add wave
- bp
- run
- force
- step

Step-5- Debugging the Design

Numerous tools and windows useful in debugging the design are available from the ModelSim GUI. In addition, several basic simulation commands are available from the command line to assist in debugging the design:

- describe
- show
- drivers
- examine
- force
- log

CHAPTER 4

SIMULATION AND RESULTS:

4.1 Design Simulation:

The Verilog code has been successfully simulated on ModelSim ALTERA(version 6.3) and synthesized using Quartus II(64 bit) version 13.0. There are total 8 inputs all are in the real and imaginary part represented as xreal and ximg i.e. real input and imaginary input as shown in figure and outputs are shown with names yreal and yimg as real and imaginary parts of output. It is shown below.

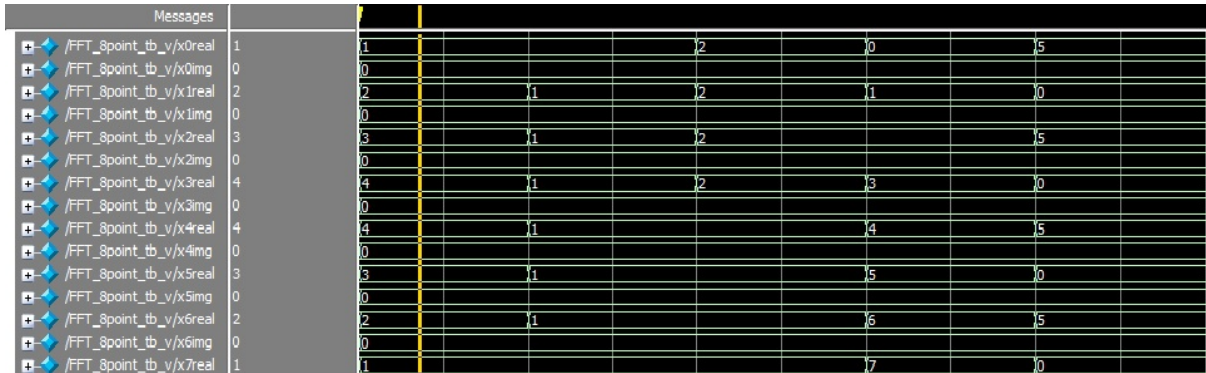


Figure-18. Test Bench Input for FFT



Figure-19. Test Bench Output for FFT

It can be observed that the output has been scaled by a factor of 100. So, in order to obtain the original value, all values from the output have to be multiplied by a factor of 100 to overcome the rounding off error.

4.2 Power Analysis and Fitter Report:

The power analysis was done using PowerPlay Power Analyzer by Altera's Quartus II version 13.0.

The details of the device used for synthesis is provided as follows.

- Target Device: EP3C16F484C6
- Device Family: Cyclone III
- LEs: 15408
- User IOs: 347
- Memory Bits: 516096
- Embedded Multiplier 9-Bit Elements: 112
- PLLs: 4
- Global Clocks: 20

Fitter Summary	
Fitter Status	Successful - Wed Apr 08 18:13:53 2015
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	fft_8point_modified_1
Top-level Entity Name	fft_8point_modified_1
Family	Cyclone III
Device	EP3C16F484C6
Timing Models	Final
Total logic elements	287 / 15,408 (2 %)
Total combinational functions	287 / 15,408 (2 %)
Dedicated logic registers	0 / 15,408 (0 %)
Total registers	0
Total pins	192 / 347 (55 %)
Total virtual pins	0
Total memory bits	0 / 516,096 (0 %)
Embedded Multiplier 9-bit elements	0 / 112 (0 %)
Total PLLs	0 / 4 (0 %)

Figure-20. Fitter Report of our Proposed Design

Fitter Summary	
Fitter Status	Successful - Mon Apr 20 14:17:31 2015
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	fft_8pt_original_1
Top-level Entity Name	FFT_8piont
Family	Cyclone III
Device	EP3C16F484C6
Timing Models	Final
Total logic elements	463 / 15,408 (3 %)
Total combinational functions	463 / 15,408 (3 %)
Dedicated logic registers	0 / 15,408 (0 %)
Total registers	0
Total pins	256 / 347 (74 %)
Total virtual pins	0
Total memory bits	0 / 516,096 (0 %)
Embedded Multiplier 9-bit elements	0 / 112 (0 %)
Total PLLs	0 / 4 (0 %)

Figure-21. Fitter Report of the Conventional Design

From analyzing the fitter report of our modified design and the conventional design, it can be observed that the total logic elements have been reduced for our design along with the combinational functions and the total pins.

Upon using the PowerPlay Power analyzer tool, our modified FFT algorithm design yielded the following report.

PowerPlay Power Analyzer Summary	
PowerPlay Power Analyzer Status	Successful - Wed Apr 08 18:19:06 2015
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	fft_8point_modified_1
Top-level Entity Name	fft_8piont_modified_1
Family	Cyclone III
Device	EP3C16F484C6
Power Models	Final
Total Thermal Power Dissipation	82.00 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	51.77 mW
I/O Thermal Power Dissipation	30.23 mW

Figure-22. Power Report for our Proposed Design

We also implemented the power analyzer tool for the existing design and the report which we obtained is given below.

PowerPlay Power Analyzer Summary	
PowerPlay Power Analyzer Status	Successful - Wed Apr 08 18:40:47 2015
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	fft_8pt_original_1
Top-level Entity Name	FFT_8piont
Family	Cyclone III
Device	EP3C16F484C6
Power Models	Final
Total Thermal Power Dissipation	163.50 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	99.21 mW
I/O Thermal Power Dissipation	64.29 mW

Figure-23. Power Report of the Conventional Design

4.3 Result:

It can be seen that the power consumption is 163.50mW for the present design and 82.00mW for our proposed design. Thus, power reduction of around 50% is observed.

CHAPTER 5

CONCLUSION:

In this project, we have designed and implemented the Radix-2 8-point DIF FFT algorithm using power and area efficient adder and multiplier. The simulation was done using Altera Cyclone III FPGA. The results of the simulation were seen to match with the results of Matlab. Implementation of 8- point FFT algorithm 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. The algorithm can be easily upgraded for a 128 point or 256 point FFT.

PUBLICATION

Sidharth P., Hari Haran S., Amirtha Gowri G., “**Design and Implementation of a Highly Efficient MAC Unit in FPGA**”, *Proceedings International Conference on Innovations in Electrical, Electronics, Information & Communication Technology (ICIEEICT)*, October 2014.

ISBN- 978-93-84209-57-5

REFERENCES

- [1] Koc, C.K., "RSA Hardware Implementation", RSA Laboratories, RSA Data Security, Inc. 1996.
- [2] Ranjan Kumar Barik, Sourav Kumar Dwibedi, Shasanka Sekhar Rout, Satya Ranjan Sahu, "An Improved and Efficient MAC Unit and its Implementation in FPGA", International Journal of Review in Electronics & Communication Engineering (IJRECE) Volume 2 - Issue 2 April 2014
- [3] KarthikKadir.V, C.VishnuKarthik, Sriram, Aravindh, C.B.Rajesh, "Design and Implementation of High Speed, Area Efficient Carry Select Adders", Proceedings of SARCITR International Conference, 04th May-2014
- [4] P.Karunakaran, S.Venkatraman, I.HameemShanavas, T.Kapilachander," Power Optimized Multiplier Using Shannon Based Multiplexing Logic", International Journal of Intelligent Systems and Applications, 2012, 6, 39-45
- [5] I-Chyn Wey, Cheng-Chen Ho, Yi-Sheng Lin, Chien-Chang Peng, "An Area-Efficient Carry Select Adder Design by Sharing the Common Boolean Logic Term", Proceedings of the International MultiConference of Engineers and Computer Scientists, 2012 Vol II, 1091-1094
- [6] ShuchiVerma, Sampath Kumar V., "Design & Analysis of Low Power, Area-Efficient Carry Select Adder", International Journal of Engineering Research and Applications, March 2014, pp.53-55
- [7] Naveen Kumar, Manu Bansal, Navnish Kumar, "VLSI Architecture of Pipelined Booth Wallace MAC Unit", International Journal of Computer Applications, November 2012
- [8] Anju S, M Saravanan, "HIGH PERFORMANCE DADDA MULTIPLIER IMPLEMENTATION USING HIGH SPEED CARRY SELECT ADDER", International Journal of Advanced Research in Computer and Communication Engineering, March 2013



IRAJ RESEARCH FORUM

In Association with

INSTITUTE OF RESEARCH AND JOURNALS



International Conference on

Innovations In Electrical, Electronics, Information & Communication Technology

ICIEEICT-GOA

Certificate

This is to certify that *Sidharth P.* has presented a paper entitled "*Design and Implementation of a Highly Efficient MAC Unit in FPGA*" at the International Conference on Innovations In Electrical, Electronics, Information & Communication Technology (ICIEEICT) held at Goa on

12th October 2014.

IR-IEEICTCOIM-12104-101

Paper ID




Chairman

INSTITUTE OF RESEARCH AND JOURNALS