Implementation of ALU using RTL to GDSII flow and on NEXYS 4 DDR FPGA board

by

Kachhadiya Radhika J.
201915011

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY
in
ELECTRONICS AND COMMUNICATION

with specialization in
Wireless Communication and Embedded Systems
to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

A program jointly offered with
C.R.RAO ADVANCED INSTITUTE OF MATHEMATICS, STATISTICS AND COMPUTER SCIENCE

July, 2021

Declaration

I hereby declare that

i) the thesis comprises of my original work towards the degree of Master of Technology in Electronics and Communications at Dhirubhai Ambani Institute of Information and Communication Technology & C.R.Rao Advanced Institute of Applied Mathematics, Statistics and Computer Science, and has not been submitted elsewhere for a degree,

ii) due acknowledgment has been made in the text to all the reference material used.

[illegible]

Radhika Kachhadiya

Certificate

This is to certify that the thesis work entitled Implementation of ALU using RTL to GDSII flow and on NEXYS 4 DDR FPGA board has been carried out by Kachhadiya Radhika for the degree of Master of Technology in Electronics and Communications at Dhirubhai Ambani Institute of Information and Communication Technology & C.R.Rao Advanced Institute of Applied Mathematics, Statistics and Computer Science under our supervision.

Prof. Rutu Parekh	Prof. Yash Agrawal
Thesis Supervisor	Thesis Co-Supervisor

Acknowledgments

Wisdom is not a product of schooling but of the lifelong attempt to acquire it. -Albert Einstein

Life is a continuous process of learning. My M.Tech journey is also an important part of this process. Completion of the thesis is one of the great achievements I have in my life. At this point, I would like to express gratitude to all the people without whom I cannot imagine traveling so far. First of all, I would like to thank almighty God for giving me the strength to complete the thesis when I have no clue for the next steps. I want to thank my parents for constantly supporting me in every situation and motivating me in life.

I would like to thank my thesis supervisors, Prof. Rutu Parekh and Prof. Yash Agrawal for constant guidance and support throughout my thesis journey. From giving such beautiful topic advice for the thesis to the last day of the thesis, they have supported me at each step. They were very patient when I was struggling in the experiments. Thank you, Mam and Sir, for having faith in me during this process. I feel exceptionally fortunate to have such a guide who has spent a lot of time with me discussing, understanding, and learning.

Lastly, I would like to thank all my friends on the campus to enrich my two years of Mtech journey beautiful. I will always be thankful to DAIICT for what it has given me.

Contents

Abstract

An ALU is the major part of the CPU which performs various arithmetic and logical operations. It is one of the most frequently used modules in the processor. This paper presents the implementation of 8-bit ALU using RTL to GDSII stream. The tools used for implementation are Cadence tools, Genus and Innovus. The technology node used for implementation is the 45nm technology node and 180nm technology node. The major focus of this thesis is the design optimization in terms of area, delay and power as the industry demands the chips with high speed and low power. Further, the results of both 45nm and 180nm has been compared. The improvement by using 45nm technology in area is 89.59%, in delay is 43.23% and in power is 4.56%. In addition to that, the implementation of 4-bit ALU is done on the FPGA board. The board used is the NEXYS 4 DDR FPGA board.

Keywords: *ALU, FPGA, Layout, Floor planning, Power planning, Routing, RTL, Simulations, Synthesis.*

List of Principal Symbols and Acronyms

ALU	Arithmetic-Logic Unit
ASIC	Application Specific Integrated Circuit
EDA	Electronic Design Automation
FPGA	Field Programmable Gate Array
GDSII	Graphic Design System II
RTL	Register Transfer Level

List of Tables

List of Figures

CHAPTER 1
Introduction

The expanding request of electronic gadgets which are solid, whose processing speed is high, that dissipates low power, and which are compact requests advancement in integrated circuit technology. To satisfy these needs for fast processors, it is important that every one of the functionalities ought to be present on a solitary chip, which is a test for a design engineer. To make this undertaking simpler, there are different apparatuses which can be utilized that contains ASIC libraries which comprises of essential structure blocks like multiplexer, flip flops, logic gates and so forth that depends on standard cell design strategies. Using these libraries, the time required for the implementation is reduced. Here, Cadence environment is used for the implementation of 8-bit ALU. These tools performs all the processes involved in RTL (Register Transfer Level) to GDS (Graphic Design System) flow and gives the final layout of the chip that is to be fabricated.

This design flow is a very mature and a silicon proven IC design process that includes different steps like design conceptualization, chip optimization, logical or physical implementation and design verification. According to the Moore's law, the number of transistors will continue to double in every 1.5 years. That means the same silicon area should accommodate more and more number of transistors. To achieve this the transistor size is gradually getting reduced. Here, the technology used for the implementation is 45nm and 180nm technology. The advantages of the 45nm technology is it uses less chip area that is to be fabricated. As the area is less the time delay also reduces and as a result the processing speed increases. Nowadays, these are the major requirements for any chip that is to be fabricated. RTL to GDSII flow is typical flow for IC designing and here it is used to implement 8-bit ALU by using Cadence tools. The flow starts with the RTL coding with is done in Verilog HDL. Further, the process of synthesis is done followed by the physical design process. In physical design process, various steps like floor plan, power plan, routing, etc. are involved. It is done at layout level.

Once the layout is made, its GDSII file is generated which is further sent to the industry for fabrication process.

1.1 Related Work

Most of the work in this area is limited to RTL level. The ALU has been implemented using reversible gates. The designed ALU performs 4 operations. The coding is done using Verilog and implemented in Cadence 180nm technology in [1]. However, in this paper the designed ALU performs 4 operations and is implemented in 180nm technology. The authors have given the idea of 16-bit Arithmetic Units with Input Gating, Power Gating and a combination of Input Gating and Power Gating. Its design, simulation and verification for their functionality has been done by using Cadence Virtuoso ADE spectre in [2]. Here, implementation is done at circuit level. In [3], the concept of Radix-2 Booth Multiplier is given and is implementation is done using Xilinx-Vivado tool. Here, in this implementation area, power and number of cells have been reduced. However, the implementation is done at RTL level. In [4], the authors proposed the design of 4-bit ALU and its implementation is done on BASYS 3 FPGA kit. The implementation is done at RTL level and for testing and verification of the implemented design simulated waveforms has been used. Further, it is synthesized using RTL analysis. In [5], comparative study between three arithmetic multipliers have been done in terms of delay and power. The implementation is done using Vering in Vivado platform at RTL level. Also, the implementation is done on NEXYS 4 FPGA board.

The authors have done the comparative study between seven 4-bit adders in terms of speed of operations and number of slices in FPGA in [6], comparisons have been done by using the spartan 3E FPGA kit. The authors have implemented ALU on the build microcontroller on FPGA with Xilinx ISE tool in [7]. However, the implementation is done at RTL level. The implementation 16-bit ALU which performs 12 operations. It is implemented in Xilinx Vivado 14.7 tool. Likewise it's implementation is carried out on BASYS 3 Artix 7 FPGA board in [8]. However, it is implemented at RTL level and for addition operation full adder is used while for multiplication operation array multiplier has been used. The authors have proposed an ALU design implemented in 180nm technology using Cadence Virtuoso tool in [9]. The architechture of 8-bit ALU and 16-bit ALU has been presented and its implemenation is done in Spartan 3E FPGA device using Verilog

HDL in [10]. Here, the implementation is done at RTL level.

Five different multipliers have been compared in terms of area, delay and power in [11]. A review is given about different types of implementation of Booth multiplier and its architechture is presented with its advantages in [12]. In [13], a review is given on various high speed multipliers with different techniques and by using different adders. The authors have implemented ALU which performs 8 operations in Xilinx 14.7 tool in [14]. The modules are written in Verilog HDL and for synthesis, physical design innovus tool from Cadence is used. The implemented ALU performs 8 different operations. In [15], the authors have implemented 4-bit Booth multiplier and 8-bit Modified Booth multiplier and have explained the complete algorithm. In [16], the 4-bit and 8-bit ALU has been implemented and it is synthesized using GDI standard cells. Further, it is compared with the CMOS standard cells. The implementation is done in 180nm technology using Cadence Virtuoso tool. In [17], the authors have presented the design of 8-bit ALU by cascading 1-bit ALUs using reversible logic. Here, the simulation and verification of the design is performed in 180nm technology using Cadence tool. The implemenatation is done at RTL level.

In [18], the idea of the complete physical design process has been given with the optimization in area and power by using smaller technology node. The authors have implemented 8-bit ALU that performs 8-operationa using Virtuoso tool by Cadence in [19]. The implementation is done by using 180nm technology. The aim of this article was to reduce the number of transistors. However, the implementation is done by using 180nm technology and the calculation of delay and power consumption is not done. The 4-bit ALU is designed by using full adder, 4xl and 2xl multiplexers and gate sub-modules such as ”XOR”, ”AND”, ”OR” and ”Inverter”. ALU executes a total of eight operations, four of which are arithmetic operations, the rest are logical operations. Three kinds of arithmetic operations are performed are adders, subtractors and comparators. The four logical operations are AND, OR, XOR and NOT in [20]. The existing multiplier designs and the proposed method were simulated by using Verilog HDL. The schematics for the 8 bit array multiplier was developed in 180nm technology by using Cadence tool in [21]. Also, the schematic for the full adder, full adder using 2x1 multiplexer and full adder using 4x1 multiplexer were implemented. Later, the performance parameters were compared by implementing array multipliers using different full adders. The step by step optimization approach for the ALU at the logic circuit level has been presented in [22].

The authors have implemented different logic gate cells in [23]. Here, the rectangular shapes for the cells are created and the area of the every cell is optimized. In [24], the authors have implemented a multiplier that reduces partial products. The authors have implemented 1-bit full subtractor in [25]. The simulations were performed by using Cadence Schematic editor and the layout is emulated by Cadence Virtuoso editor. In [26], the authors have presented the idea for creating graphical user interface for the opensource VLSI tool that is Qflow. The authors have given the idea of CMOS Comparator and implemented that is compact in size and uses low power with the supply voltage of 1.8V in [27]. It is designed and implemented in 180nm technology. In [28], the authors have presented the idea for implementation of ALU and focuses mainly on reducing the power consumption and reducing areaby using GDI technique. Here, 4x1 multiplexer, 2x1 multiplexer and a full adder is implemented in 180nm technology.

1.2 Motivation

Based on the above discussion, it is seen that most of the work done in this area is limited to RTL level or circuit level. Also, in most of the cases the implementation is done in 180nm technology node. The current industry demands chips with low power and area. So implementation in lower technology nodes is necessary as in that the area, delay and power will be lesser. So here the implementation is done in 45nm technology. Also, its area, delay and power has been optimized which is the requirement of the industry. In addition to that, the implementation is compared with the implementation in 180nm technology and the results shows 45nm technology node gives better results.

1.3 Thesis Outline

This Thesis presents work on RTL to GDSII flow and optimization of area, delay and power in the circuit. This is organized in 6 chapters.

- **Chapter 2** contains the brief introduction of Arithmetic-Logic Unit and its working. Also, its implementation is given with the explanation of adder and multiplier that are used in the circuit.

- **Chapter 3** contains the details of the complete RTL to GDSII flow. The application of this flow is presented here. Also, the explanation of each step of the flow is given.
- **Chapter 4** contains the brief introduction on Field Programmable Gate Array (FPGA). Also, the implementation of the circuit on the FPGA board is presented here.
- **Chapter 5** contains brief details about results and the comparison with the existing work.
- **Chapter 6** contains details about future works and conclusion of thesis.

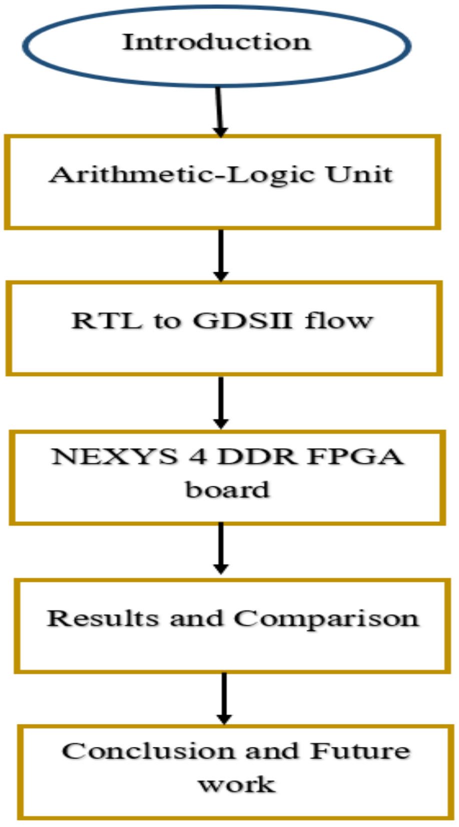

Figure 1.1: Thesis outline

CHAPTER 2
Arithmetic-Logic Unit

An Arithmetic logic unit (ALU) is a significant segment of the central processing unit of a PC framework. It does all cycles identified tasks which are related to logical and arithmetic operations. In some microchip structures, the ALU is isolated into the Arithmetic unit (AU) and the Logic unit (LU).

An ALU can be planned by specialists for computation of any operation. As the activities become more complex, the ALU likewise turns out to be more costly, occupies more area in the CPU and disperses more power. That is the reason engineers make the ALU sufficiently incredible to guarantee that the CPU is additionally amazing and quick, yet not really complex as to become restrictive in terms of cost and different inconveniences. The fundamental functions of the ALU are to do logic and arithmetic operations, including bit-wise shifting tasks. These are fundamental operations that should be done on practically any information that is being handled by the CPU.

In computing, an arithmetic-logic unit is a combinational digital circuit that is used to perform various arithmetic operations and bit-wise logical operations on integer binary numbers. This is in contrast to floating point unit that operates on floating point numbers. It is the fundamental building block of many of the computing circuits including central processing unit, floating point unit and graphics processing unit. An arithmetic-logic unit is an important part of Central Processing Unit in Computer. It carries out various arithmetic and logic operations. It performs different logical operations like AND, OR, NOT, etc. and arithmetic operations like Addition, Subtraction, etc. It also performs various shifting operations. The input to an ALU are the data that are to be operated and are called operands and one code that indicates the operation that is to be performed. The output of the ALU is the result of the operation that is performed.

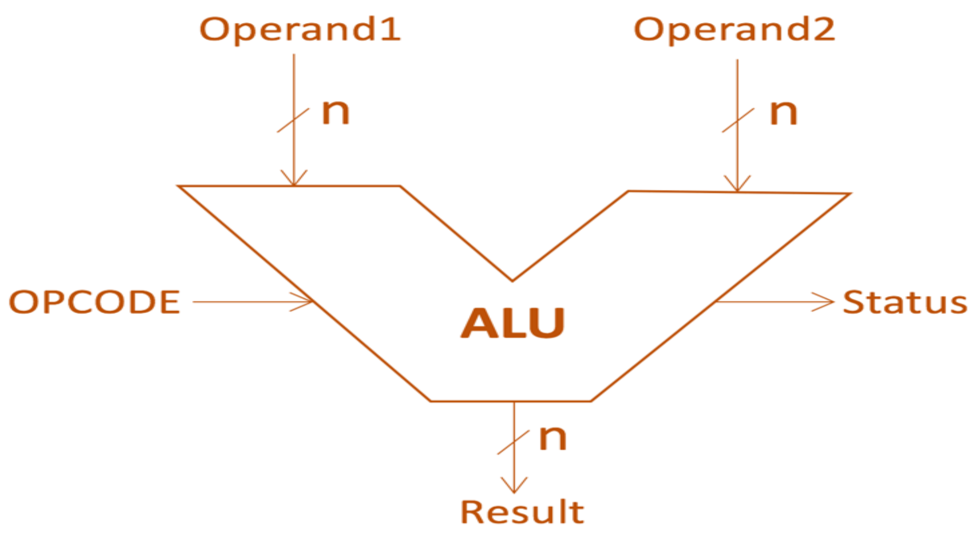

Figure 2.1: A symbolic representation of ALU

2.1 Arithmetic Operations

- Addition: The two inputs Op1 and Op2 are summed and the result appears at Out along with the carry-out.
- Subtraction: Op2 is being subtracted from Op1 (or vice versa) and the difference appears at Y and carry-out. For this operation, carry-out is successfully a "borrow" indicator.
- Multiplication: The two inputs are being multiplied and the result is shown at Out.
- Increment: The input Op1 is incremented by one and the result appears at Out.
- Decrement: The input Op1 is decremented by one and the result appears at Out.

2.2 Logical Operations

- AND: The bitwise AND operation of Op1 and Op2 is done and result is shown at Out.

- OR: The bitwise OR operation of Op1 and Op2 is done and result is shown at Out.
- NOT: The bits of Op1 is being inverted and the result appears at Out.
- Exclusive-OR: The bitwise XOR operation of Op1 and Op2 is done and result is shown at Out.
- Exclusive-NOR: The bitwise XNOR operation of Op1 and Op2 is done and result is shown at Out.
- NAND: The bitwise NAND operation is performed between Op1 and Op2 for which result appears at Out.
- NOR: The bitwise NOR operation is performed between Op1 and Op2 for which result appears at Out.

2.3 Shifting Operations

ALU shift activities cause operand Op1 or Op2 to move left or right (contingent upon the opcode) and the moved operand shows up at Out. In all single-cycle shift tasks, the bit moved out of the operand shows up on carry-out; the worth of the bit moved into the operand relies upon the kind of shift.

- Arithmetic shift: The operand is treated as a two's complement integer, meaning that the most significant bit is a "sign" bit and is preserved. The arithmetic shift operation can be used in multiplication as well as division algorithms.
- Logical shift: A logic zero is shifted into the operand. This is used to shift unsigned integers. The logical shift operation can be used only in multiplication algorithms.

2.4 8-bit ALU Implementation

The block diagram of implemented ALU is shown in figure 2.2. In this design, the Op1 and Op2 represents the 8-bit inputs which are first given to the registers along with the clock signal. Further, it output of the registers acts as the input to the ALU circuit which are given with the control signals that selects the operation of the ALU that is to be performed. The implemented ALU is of 8-bits and performs 16 operations, among which the arithmetic operations are addition, subtraction, multiplication, increment and decrement, and the bitwise logical operations are AND, OR, NOT, NAND, NOR, XOR, XNOR. Shifting operations performed are arithmetic and logical, left and right shift operations. For logical operations, different gates are implemented. For the addition operation, the Carry-Look Ahead adder is implemented and for the multiplication operation, the Modified Booth's algorithm is implemented because these are efficient. Control Unit implemented gives the instructions to the ALU and ALU performs the corresponding operation.

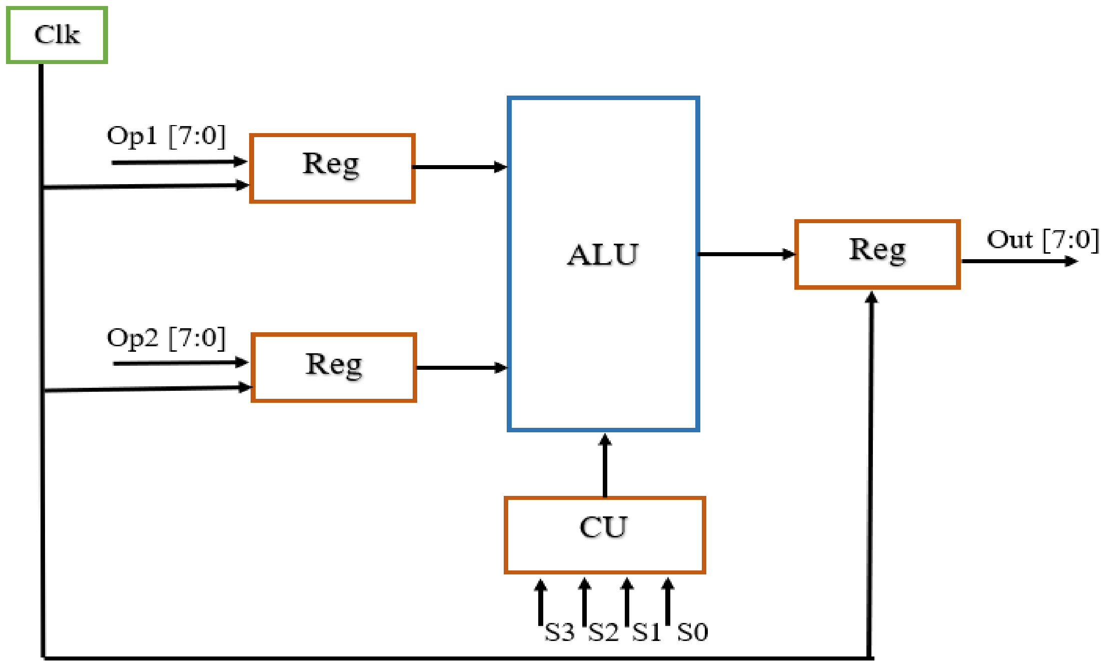

Figure 2.2: ALU Block Diagram

2.4.1 Carry Look-Ahead adder

Carry Look-Ahead adder speeds up by decreasing the measure of time needed to decide the carry. It can be compared with the simpler adder that is Ripple Carry Adder (RCA) which is usually slower. For this, the carry bit is determined along with the sum bit and each stage should stand by until the previous carry bit is determined. The Carry Look-Ahead adder computes at least one carry bit prior to adding, which reduces the waiting time for determining the result of the larger value of the adder. It reduces propagation delay by introducing more complex hardware. In this design, the ripple carry design is appropriately converted to simplify the carry logic on the fixed bit group of the adder to a two-level logic. In this design, two 4-bit Carry Look-Ahead adders are used to implement an 8-bit Carry Look-Ahead adder. By considering full adder circuit, two factors can be characterized as carry generate G_i and carry propogate P_i.

$$P_i = A_i \oplus B_i \tag{2.1}$$

$$G_i = A_i B_i \tag{2.2}$$

The sum output and the carry output are given as follows,

$$S_i = P_i \oplus C_i \tag{2.3}$$

$$C_{i+1} = G_i + P_i C_i \tag{2.4}$$

Where the carry generate is indicated as G_i which produces the carry when both the inputs i.e. A_i, B_i are one regardless of the input carry. The carry propagate is gives as P_i which is associated with the carry propagation from C_i to C_{i+1}. Figure 2 shows the circuit diagram of 4-bit carry look-ahead generator. The output of the carry in each stage can be given by the following equations.

$$C_1 = G_0 + P_0 C_i \tag{2.5}$$

$$C_2 = G_1 + P_1 G_0 + P_1 P_0 C_i \tag{2.6}$$

$$C_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_i \tag{2.7}$$

$$C_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_i \tag{2.8}$$

From the given equations (ref:geeksforgeeks.org), it can be seen that C_4 does not need to wait for C_2 and C_3 for its propagation. C_4 can be propagated at the same time when C_2 and C_3 propagates which eventually reduces delay of the circuit. Where G_i is a carry generate which produces the carry when both A_i, B_i are one regardless of the input carry. P_i is a carry propagate and it is associate with the propagation of carry from C_i to C_{i+1}. Figure 2.3 shows the circuit diagram of carry look-ahead generator.

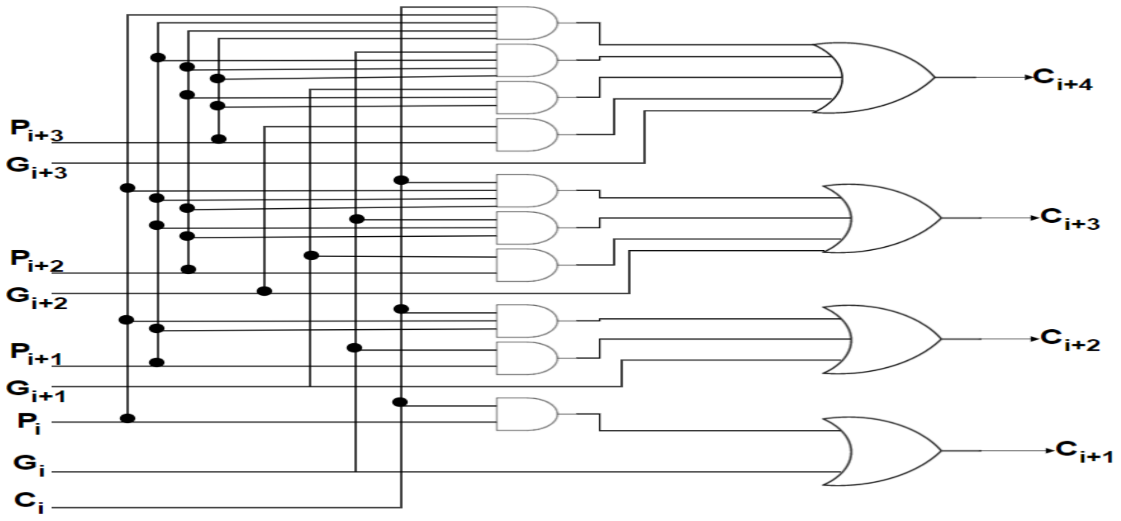

Figure 2.3: Carry Look-Ahead Generator

Some of the advantages of Carry look-ahead adder are, it gives the fastest logic for addition and hence the propagation delay can be reduced. It has some of the disadvantages like the number of variables increases and hence the circuit becomes complicated. Also it involves more amount of hardware which results in the increase of cost.

2.4.2 Modified Booth's algorithm

The algorithm for Booth multiplication comprises of three significant steps as displayed in the structure of the diagram for the algorithm. The algorithm generally incorporates reducing the partial products, generation of the partial product knows as recoding process and the addition that gives the final product of multiplication process. The given below figure 2.4 shows the flow diagram of Modified Booth's algorithm.

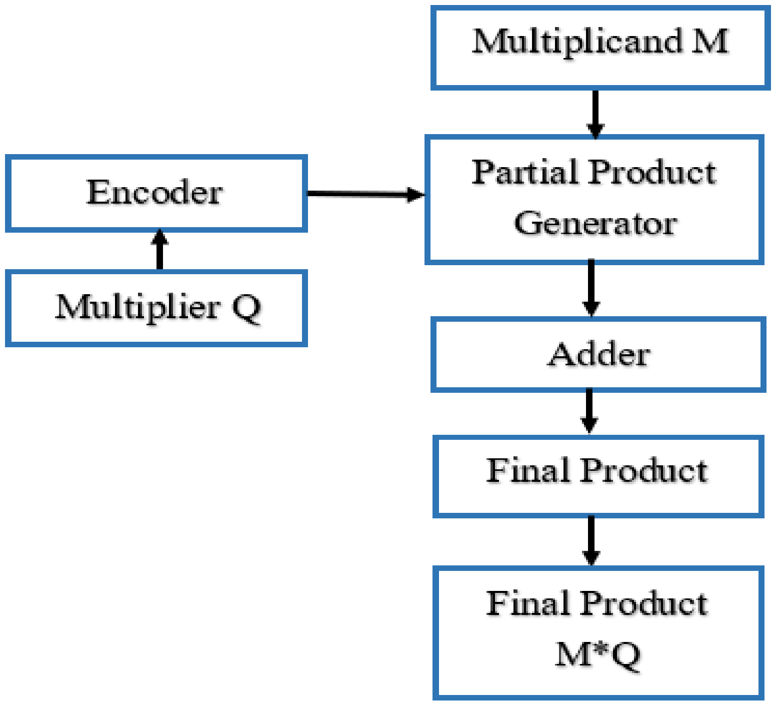

Figure 2.4: Modified Booth's Algorithm

The high-speed multiplication process can be carried out with Modified Booth's algorithm. We can reduce the quantity of products by half. This modified booth multiplier's computation time and the logarithm of the word length of operands are proportional to each other. The implemented multiplier is of 4-bits. The radix-4 Booth algorithm which is utilized here speeds up the multiplier and diminishes the space of the circuit. In this calculation, every other column is multiplied by 0 or +1 or +2 or - 1 or - 2, rather than moving and amassing every column of the Booth multiplier. Therefore, the use of this Booth algorithm can reduce number of the partial product by half. Depending on the multiplier bits, the encoding process of the multiplicand is performed by the base 4 Booth encoder. The overlapping process is used to compare three bits at a time. The grouping of the bits starts with the least significant bit. In the least significant bit, the first block uses only two bits of the Booth multiplier, and zero is assumed to be the third bit.

The table 2.1 shows functional operations of the radix-4 Booth encoder that comprises of eight distinct sorts of states. The results or multiplication of multiplicand with 0, 1, 2, - 1 and - 2 are successively acquired during these eight states.

Table 2.1: Booth Recoding Table

Multiplier bits block			**Recoded 1-bit pair**		**2 bit booth**	
i+1	***i***	***i-1***	***i+1***	***i***	***Multiplier Value***	***Partial Product***
0	0	0	0	0	0	Mx0
0	0	1	0	1	1	Mx1
0	1	0	1	-1	1	Mx1
0	1	1	1	0	2	Mx2
1	0	0	-1	0	-2	Mx-2
1	0	1	-1	1	-1	Mx-1
1	1	0	0	-1	-1	Mx-1
1	1	1	0	0	0	Mx0

The major steps for the radix-4 booth algorithm are given below:

- The first position sign bit is extended if required. It is done to ensure that the total number of bits is even.
- Then a zero is added to the right of the least significant bit of the multiplier.
- After the above steps, each partial product will 0, +1, -1, +2 or -2, according to the value of the vector.

CHAPTER 3
RTL to GDSII flow

Design streams are the express blend of electronic design automation tools to achieve the plan of an integrated circuit. Moore's law has driven the whole IC execution from RTL to GDSII configuration streams from which one basically utilizes independent routing, synthesis and placement algorithms to a coordinated development and examination streams for design conclusion. The RTL to GDSII stream went through critical changes from 1980 through 2005. The ongoing scaling of CMOS advancements fundamentally changed the targets of the different design steps. The absence of good indicators for delay has prompted huge changes in late design streams. New scaling difficulties, for example, reliability, variability and leakage power will keep on requiring critical changes to the design closure measure later on. Numerous elements depict what drove the design stream from a bunch of discrete design steps to a completely integrated methodology and what further changes are coming to address the most recent difficulties.

The design process can be divided into two parts, one is the front end design process, the other is the back-end design process. In VLSI, physical design is a process in which the structural level netlist moving from the front-end design process transforms to the back-end design process of the structural hierarchy netlist to a physical layout database, which consists of a Geometric design information for all physical layers used for interconnection. The front-end process defines the solution to a given problem and design of the circuit at the RTL level. The steps involved here are Block-level architecture design, RTL coding and its Functional Verification. The back-end process involves all other steps of integration to the format of the GDS file. Figure 3.1 shows the complete flowchart from RTL to GDSII. The following are the steps for RTL to GDSII flow:

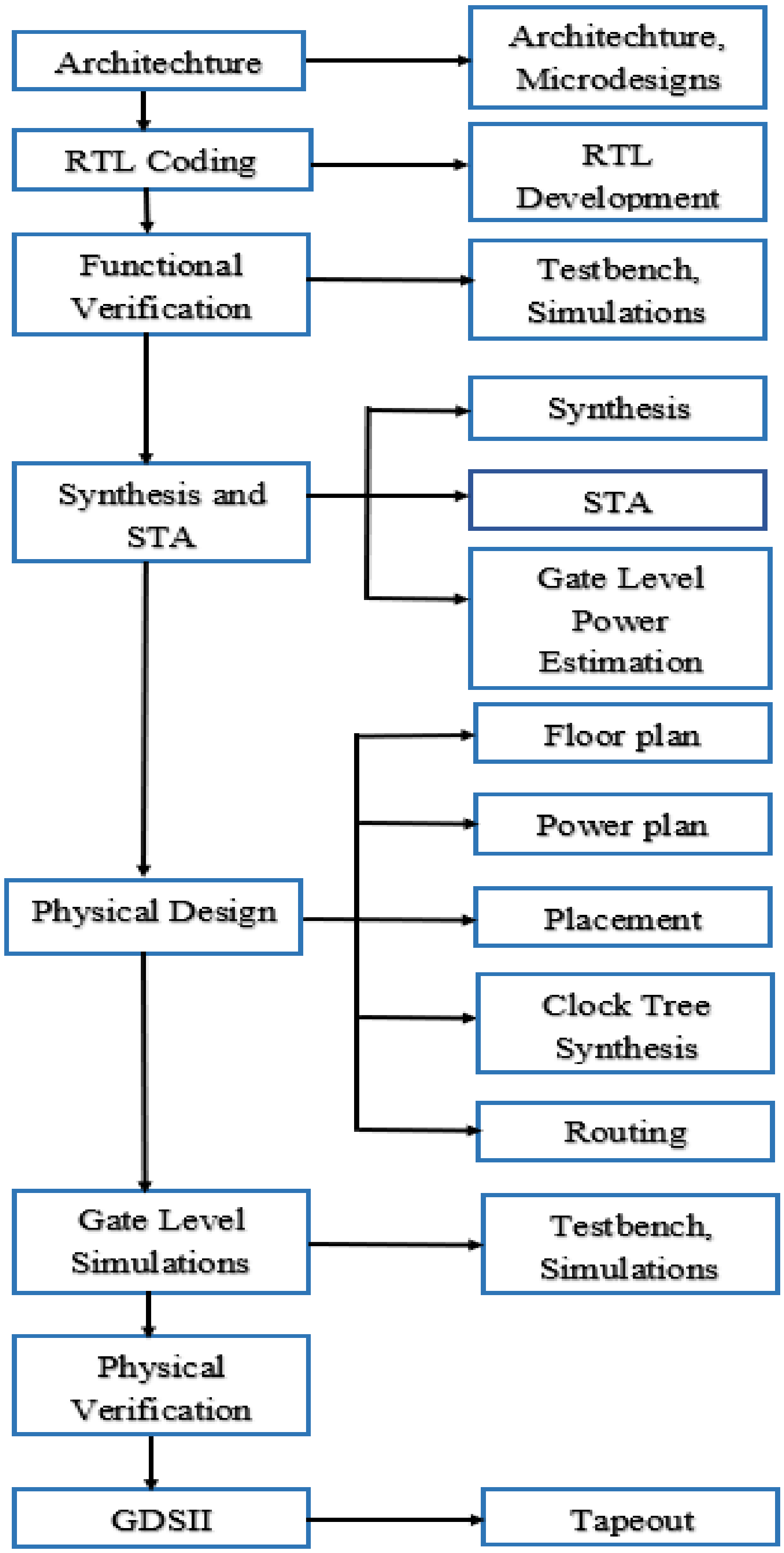

Figure 3.1: RTL to GDSII flow

3.1 Architechture

At this level, functions, features, and the specifications of the chip are in accordance with claim. Therefore, the customer's request plays an important role in determining how the chip works. So this is the first step to gather requirements, and accordingly it is first designed at the block level.

3.2 RTL level

According to chip specifications, RTL code of the ones used for block-level design is written in Verilog HDL and the complete verification is done. This is called Behavior simulation.

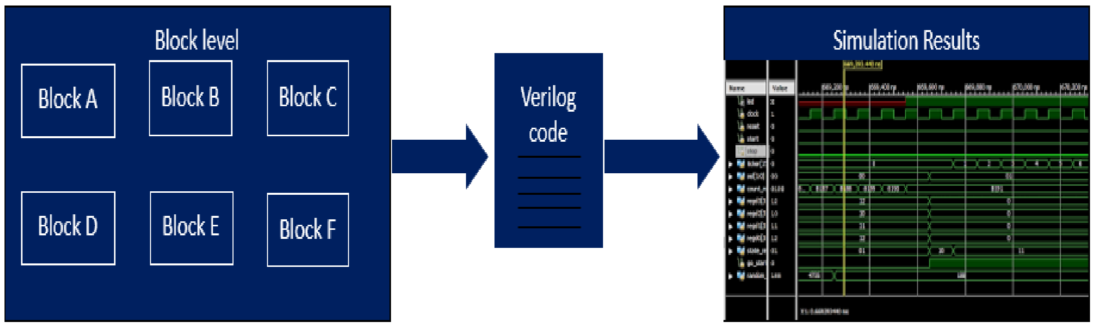

Figure 3.2: RTL level

3.3 Synthesis

In this process, once the RTL code and its testbench is generated it is converted into a gate-level netlist using a logical synthesis tool. It generally helps to produce the netlist that consists the description and interconnections of logic cells. Logic synthesis tools ensures that the netlist meets the specifications like timing, area

and power.
The goals of the synthesis process are:

- To get a gate level netlist.
- Inserting clock gates.
- Logic optimization.
- Inserting DFT logic.
- Logic equivalence between RTL and netlist should be maintained.

3.4 STA

Static timing analysis is one the techniques that is used to verify the timing constrains of a digital design. It verifies the digital design in terms of timing. It is a method which is used for the validation of timing performance of a design. It is done by checking all possible paths for timing violations. STA splits a design down into different timing paths, then calculates the signal propagation delay for each path and checks for violations of timing constraints inside the design and at the input/output interface.

3.5 Physical Design

Different steps are involved here, such as floor plan, power plan, placement, clock tree synthesis, routing. It is at the layout level. The floor plan involves arrangement of the different blocks on the chip. During the placement process, the position of the cell in the block is decided. The routing process makes the connection between the cell and the block. The concept behind the clock tree synthesis is to ensure that the clock arrives every flip-flop present in the chip. After this process, Physical verification is done to ensure that the designed layout work as it was intended or not. After the physical design process, the DEF (Design Exchange Format) file of the layout is generated. This DEF is used and it is read by the tool along with the LEF (Library Exchange Format) files. From this the optimization of the parameters has been done.

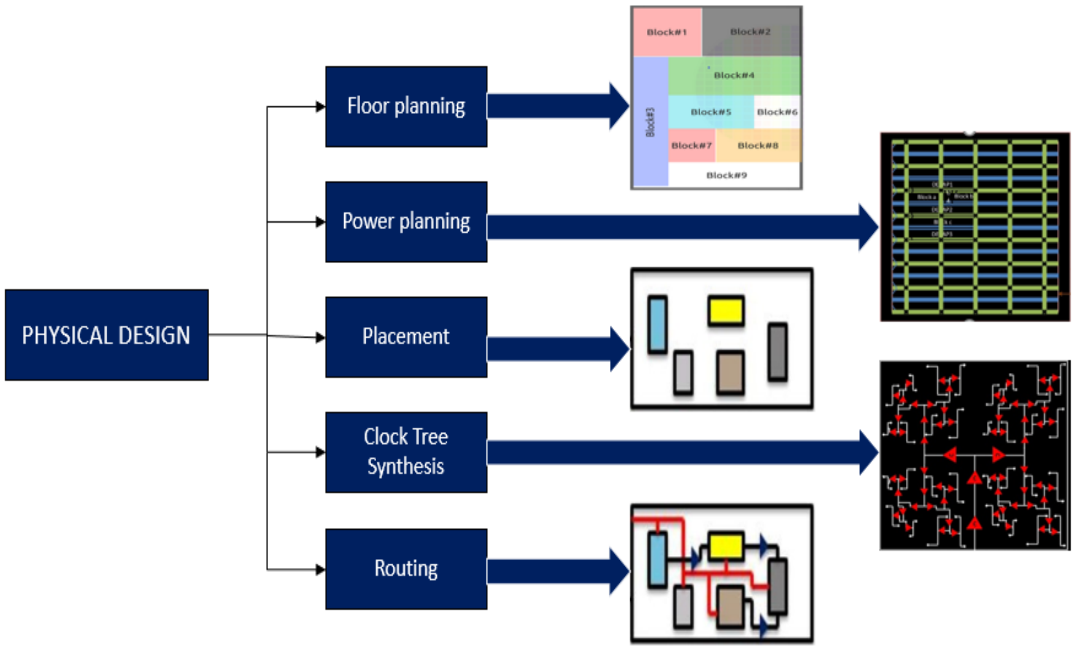

Figure 3.3: Physical design flow

3.6 GDSII

After verification, the GDS file of the design is formed. It is in binary file format that represents geometric shapes, text labels and others information about the layout of the chip that is to be manufactured.

CHAPTER 4

FPGA Implementation

Field Programmable Gate Arrays (FPGAs) are semiconductor gadgets that are based around a lattice of configurable logic blocks (CLBs) associated through programmable interconnects. FPGAs can be reprogrammed to wanted application or usefulness prerequisites subsequent to assembling. This component recognizes FPGAs from Application Specific Integrated Circuits (ASICs), which are specially made for specific tasks of design. Although once time programmable (OTP) FPGAs are accessible, the prevailing sorts are SRAM based which can be reprogrammed as the plan develops.

ASIC and FPGAs have different incentives, and they should be carefully assessed prior to picking any one over the other. Data proliferates that analyzes the two innovations. While FPGAs used to be chosen for lower speed/complexity/volume plans previously, the present FPGAs effectively push the 500 MHz execution boundary. With uncommon logic density increases and a large group of different highlights, like embedded processors, clocking, DSP blocks, etc. FPGAs are a convincing suggestion to design any sort of plan.

4.1 FPGA Architechture

A fundamental FPGA involves a huge number of key parts called configurable logic blocks (CLBs) enveloped by an arrangement of programmable interconnects, called a fabric, that courses hails between CLBs. Information or Output (I/O) blocks interface between the FPGA and outside devices. Given figure 4.1 shows the fundamental architechture of FPGA.

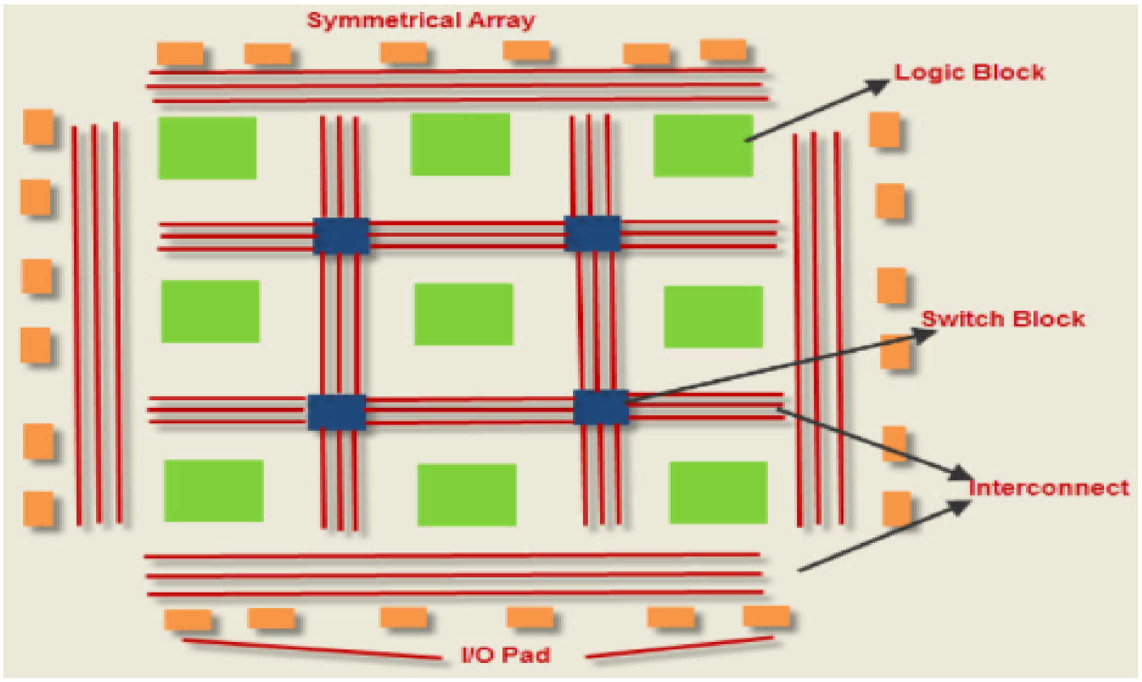

Figure 4.1: Basic FPGA Architechture (Image source:elprocus.com)

An individual CLB contains a couple of logic blocks. A lookup table (LUT) is a brand name feature of a FPGA. A LUT stores a predefined rundown of logic output for any blend of data sources: LUTs with four to six data bits generally used. Standard reasonable capacities like full adders (FAs), flip-flops and multiplexers (mux) are furthermore ordinary.

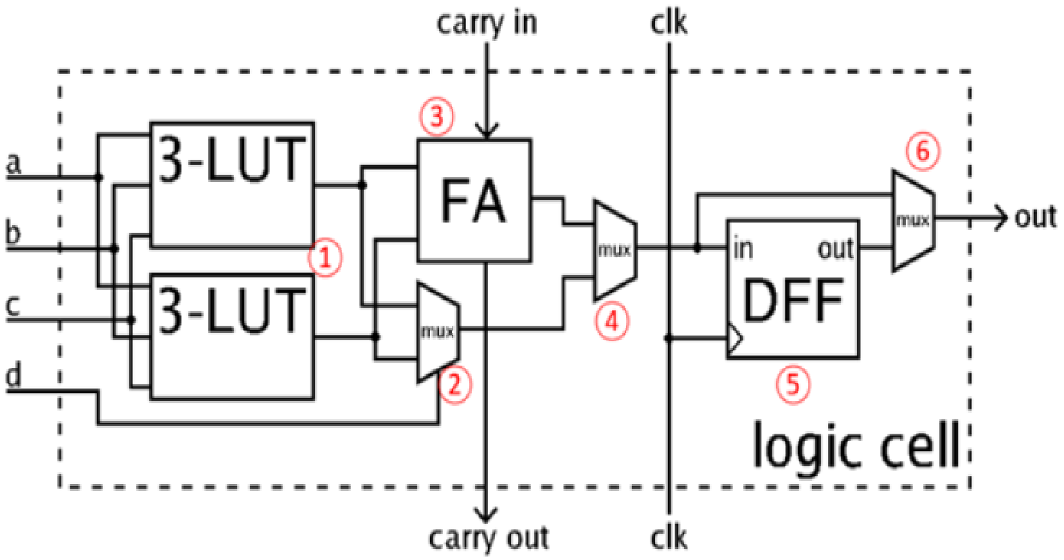

Figure 4.2: A Simplified CLB (Image source:Wikipedia)

The number and plan of segments in the CLB changes by gadget; the improved model is shown in figure 4.2. It contains two three-input LUTs as shown by (1), a full adder indicated by (3) and a D-type flip-flop as given as (5). In addition to a standard multiplexer indicated by (2) and two other multiplexers that are (4) and (6) are arranged during FPGA programming. This improved model of CLB has two methods of activity. In normal mode, the LUTs are joined with Multiplexer (2) to frame a four-input LUT; in arithmetic mode, the LUT yields are taken care of as contributions to the full adder along with a convey contribution from another CLB. Multiplexer (4) chooses between the full adder yield or the LUT yield. Multiplexer (6) decides if the activity is offbeat or synchronized to the FPGA clock by means of the D flip-flop. Current-age FPGAs incorporate more perplexing CLBs equipped for various activities with a single block; CLBs can consolidate for more mind boggling tasks like multipliers, registers, counters and surprisingly digital signal processing (DSP) capacities.

4.2 NEXYS 4 DDR FPGA board

The Nexys 4 DDR board is a completed, ready to-use modernized circuit improvement stage subject to the latest Artix-7 Field Programmable Gate Array (FPGA) from Xilinx. With its tremendous, high-limit FPGA (Xilinx part number XC7A100T-1CSG324C), liberal external memories, and grouping of USB, Ethernet, and various ports, the Nexys4 DDR can have plans going from the combinational circuits to fantastic introduced processors. A couple of innate peripherals, including an accelerometer, temperature sensor, MEMs mechanized mouthpiece, a speaker enhancer, and a couple of I/O devices grant the Nexys 4 DDR to be used for a wide extent of plans without requiring some different segments. The board used for implementation is shown in figure 4.3.

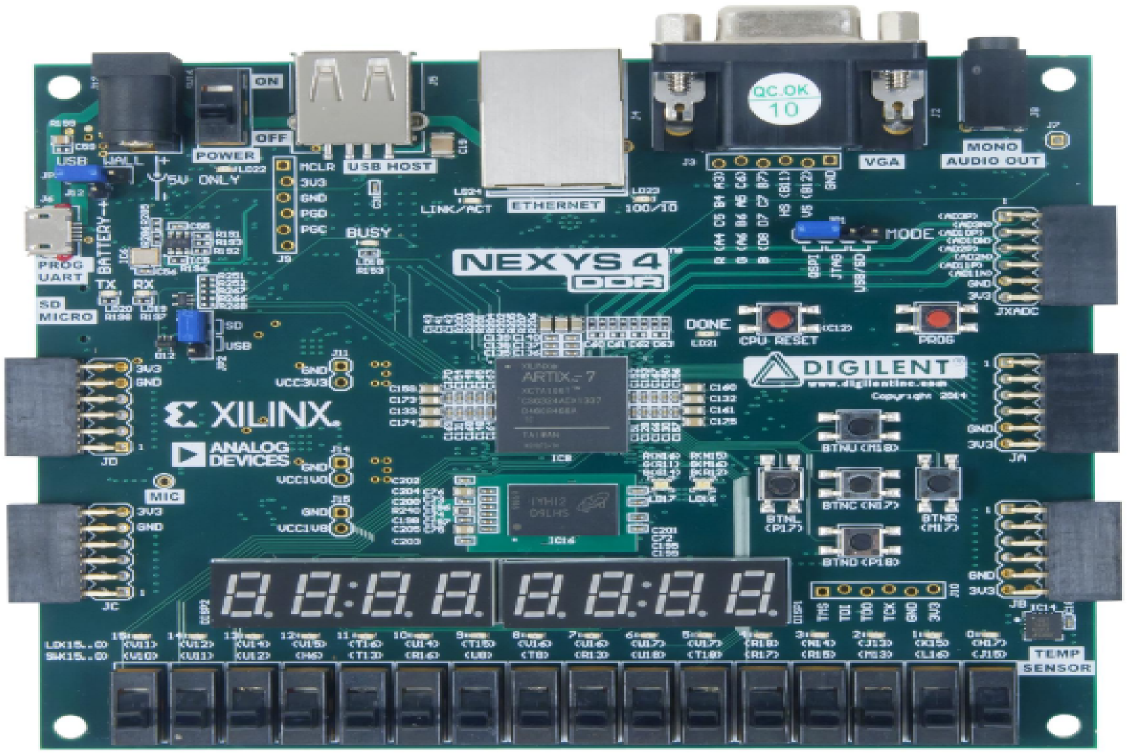

Figure 4.3: NEXYS 4 DDR FPGA board

Verilog HDL was utilized for coding of the design which was further compiled and simulated by utilizing Xilinx ISE 14.7. After that point, it was stacked into the FPGA board. The two 4-bit inputs, A and B were given to the FPGA to perform the 4-bit ALU operations. In FPGA implementation the inputs of the ALU were given to the switches of the FPGA board and the output was shown by LED's. According to the select lines that were given as input to the FPGA, the corresponding output can be observed by the LED's.

Steps for the implementation of ALU on NEXYS 4 DDR FPGA board are as follows:

- The first step was to compile the program, then the program was simulated and synthesized by running simulation.

- The next step was to load the program in the FPGA by using the cable and the inputs and outputs were mapped according to the code.

- In the final step, the operations were tested by giving inputs to the switches of FPGA board.

The hardware realization was carried out by using the FPGA kit. Further, the results were verified with the simulated waveforms.

CHAPTER 5
Results and Comparisons

In this chapter, we will discuss about the simulation results, results of area, delay and power in 45nm technology, 180nm trechnology and of FPGA implemenatation. Also, we will discuss about the synthesized RTL schematic which is obtained by using Genus tool from Cadence and about the layout of the design which is done by using Innovus tool from Cadence. At the end, the comparison is done between the results obtained from 45nm technology node and 180nm technology node. Also, the comparison is done with the existing work in terms of delay and power.

The simulations were performed using isim tool and Xilinx ISE tool to complete RTL level coding in Verilog HDL. The simulation results for ALU are shown in the following figures. Figure 5.1 shows the performed logical operations, figure 5.2 shows the results of shifting operations and figure 5.3 shows the results of arithmetic operations. Two inputs are given here, and an output is selected according to the instructions given by the control unit. The given two inputs are 15 and 3, the operation is performed according to the control signals given to the ALU. For multiplication operations, the inputs are -5 and 4. Once the RTL-level code is written, it is converted into a gate-level netlist through a synthesis process. The synthesized RTL schematic is shown in figure 5.4. Op1 and Op2 shown in figure are the 8-bit inputs that are given. Op3 and Op4 represents the 4-bit inputs for the multiplier. Op represents the opcode which is a 4-bit input that selects the operation for the ALU and clk is the clock given as the input. The 8-bit output is represented as Out in the figure. This process was done using the Genus tool and the area, power consumption and delay were optimized. After that, Innovus tool was used to complete the physical design process. The layout is shown in figure 5.5. In the figure Vdd and Vss lines are shown. Also, standard cells and connection lines has been represented. The technology node used for implementation is the 45nm technology node 180nm technology node. Further, results of both the implementation has been compared.

5.1 Logical Operation Results

The given figure 5.1 shows the results of the logical operations performed. The operations performed here are AND, OR, NOT, XOR, XNOR, NAND and NOR. It can be observed from the figure that two inputs are given and according to the opcode the corresponding operation is shown at the output.

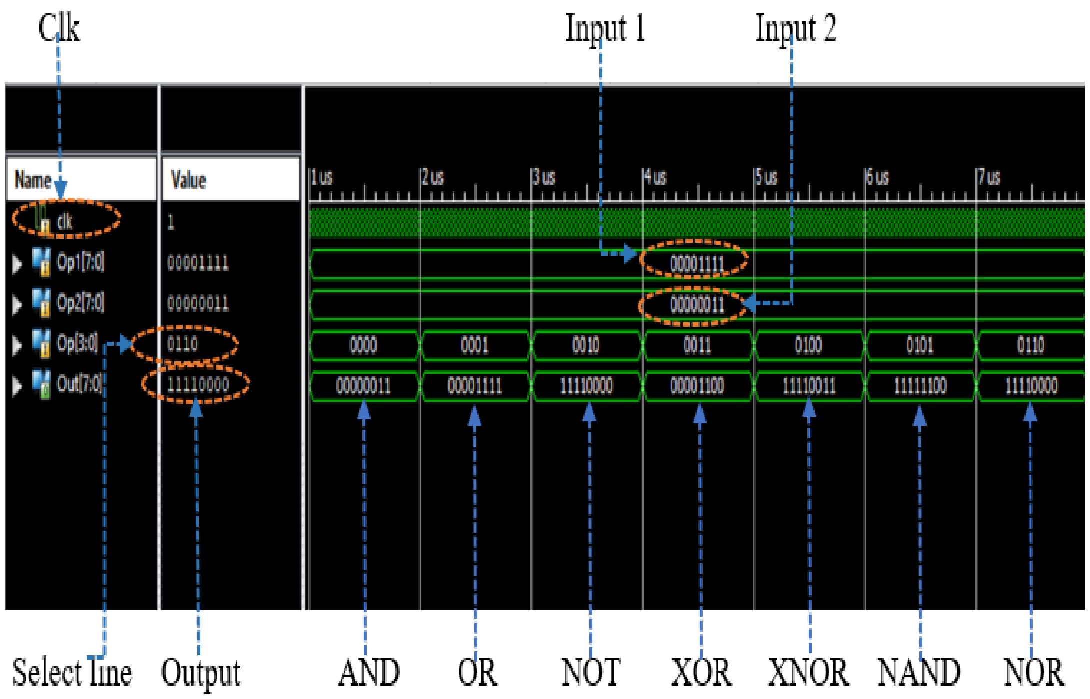

Figure 5.1: Logical Operations

5.2 Shifting Operation Results

The given figure 5.2 shows the results of the shifting operations performed. The operations performed for this are Logical Shift Left (LSL), Logical Shift Right (LSR), Arithmetic Shift Left (ASL), Arithmetic Shift Right (ASR). These operations are performed for input 1 and according to the opcode the corresponding operation is performed. The main difference between arithmetic shift and logical shift is arithmetic shift can perform multiplication as well as division whereas logical shift can only perform multiplication.

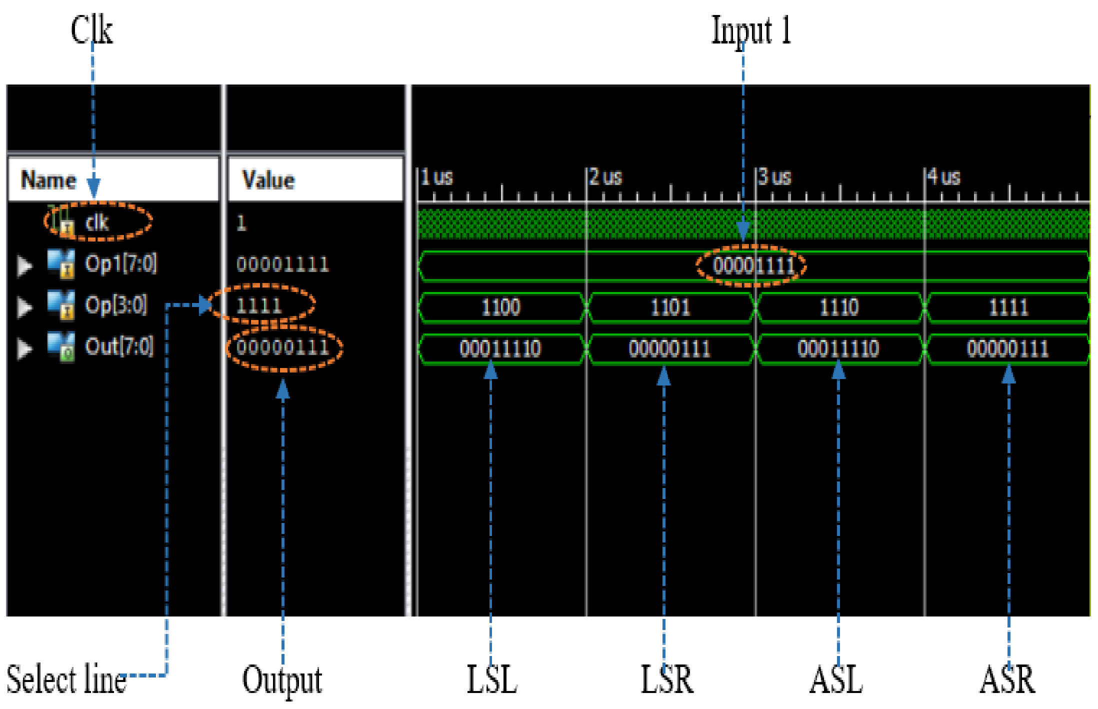

Figure 5.2: Shifting Operations

5.3 Arithmetic Operation Results

The given figure 5.3 shows the results of the arithmetic operations performed. The operations performed here are addition, subtraction, multiplication, increment and decrement. It can be observed that the inputs given here are 15 and 3 for which the results of addition, subtraction, increment and decrement are selected according to the opcode. For the multiplication operation the inputs given are -5 and 4 and the corresponding result is -20.

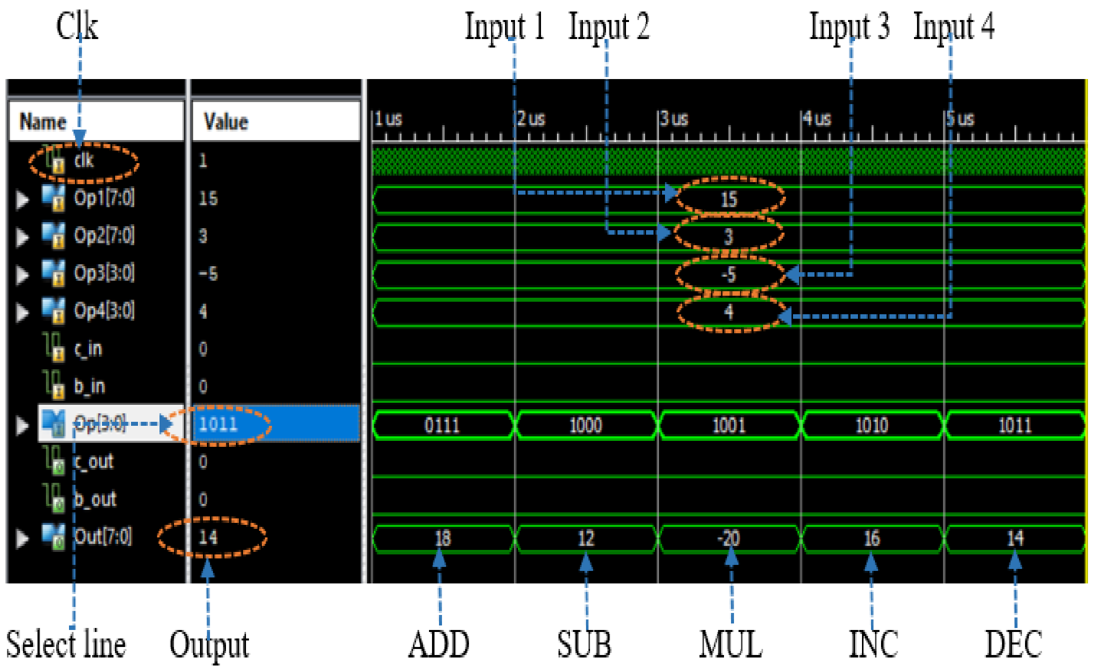

Figure 5.3: Arithmetic Operations

5.4 RTL schematic and Layout

The figure 5.4 shows the synthesized RTL schematic of the circuit. This is performed by using Genus tool. All the inputs and outputs are indicated and can be observed. The figure 5.5 shows the layout in 45nm technology node which is performed by using Innovus tool. The Vdd and Vss lines are shown as well as the standard cells and connection lines are indicated. Further, the optimization of area, delay and power is done by using Genus tool.

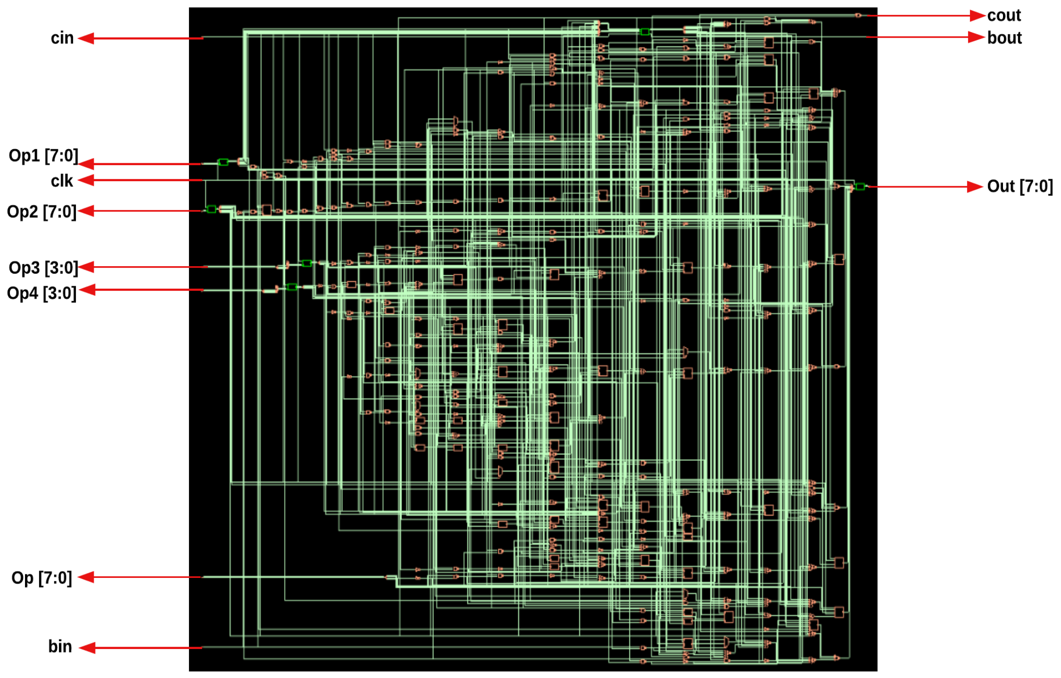

Figure 5.4: Synthesized RTL Schematic

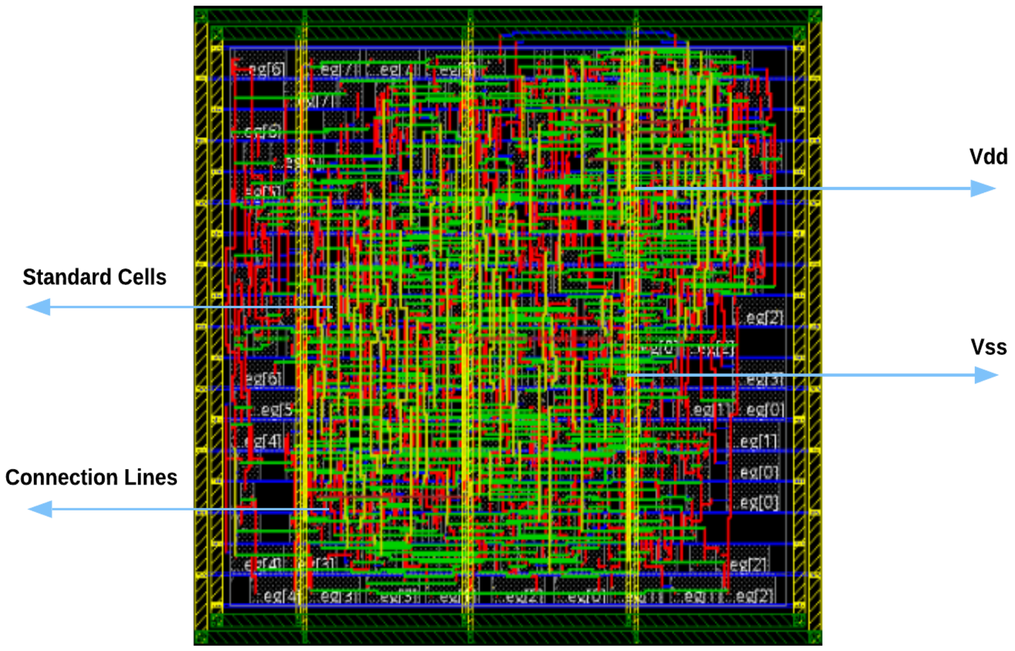

Figure 5.5: Layout in 45nm technology

5.5 Comparison between 45nm and 180nm

The below table 5.1 shows the comparison results in terms of area, delay and power and the improvement by using 45 nm technology. It can be observed from the results that the 45nm technology gives better results in terms of area, delay and power. Also, the comparison is done with the existing work in terms of delay and power. The comparison shows that the implemented design gives better results.

Table 5.1: Comparison Table

Technology Node	**Parameters measured**		
	Area (μm)	***Delay (ps)***	***Power (μW)***
ALU in 45nm	1493.31	478	54.75
ALU in 180nm	14356.54	842	57.37
Improvement (%)	89.59	43.23	4.56
FPGA Implementation	-	2146	88000
8-bit ALU [16]	-	-	560.99
8-bit ALU [17]	-	5520	-

CHAPTER 6
Conclusion and Future work

The complete implementation of 8-bit ALU has been successfully done from RTL level to layout level. The ALU was implemented in 45nm technology and 180nm technology by using Cadence tools. The optimization of design was done in terms of area, delay and power. Further, the design implemented in 45nm technology has been compared with the design implemented in 180nm technology and the observation shows that 45nm technology gives better results in terms of area, delay and power as compared to 180nm technology. The improved results shows that area is reduced by 89.59%, delay is reduced by 43.23% and power is reduced by 4.56%. Also, the 4-bit ALU is implemented on NEXYS 4 DDR FPGA board. The results were compared with simulated waveforms. The delay observed for this implementation was 2146ps and power consumption was 88000µW. The results of 8-bit ALU has also been compared with the results of the previous work in terms of delay and power and from the observation the current implemented design gives better output as shown in the table II.

The implementation of the design by using RTL to GDSII flow can be extended to 32-bits. Also, it can extended on the FPGA board.

References

[1] A. Vinotha and N. Radha, "Design And Implementation Of Reversible Logic Alu With 4 Operations", International Conference on Electrical, Information and Communication Technologies(ICEICT), 2017.

[2] K. Bikshalu and P. Soma, "Design and Simulation of 16 Bit Arithmetic Unit using Gating Techniques in Cadence 45nm Technology", International Journal of Engineering and Advanced Technology (IJEAT),. ISSN: 2249 – 8958, Volume-6 Issue-3, 2017.

[3] M. S. Kumar, S. Inthiyaz, M. A. Babu, B. Teja, S. J. Ahmed, T. S. Harika and K. Bhaskar, "Two speed Radix-2 Booth Multiplier using verilog", Journal of Critical Reviews, Vol 7, Issue 4, 2020.

[4] A.K. Panigrahi, S. Patra, M. Agarwal and S. Satapathy, "Design and Implementation of a high speed 4bit ALU using BASYS3 FPGA Board", Innovations in Power and Advanced Computing Technologies (i-PACT), 2019.

[5] U. Mandal, R. Banerjee and R. Mishra, "A comparative study of Arithmetic Multipliers", 2nd International Conference on Electronics, Materials Engineering and Nano-Technology (IEMENTech), 2018.

[6] B. Koyada, N. Meghana, M. O. Jaleel and P. R. Jeripotula, "A compartive study on adders", International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), 2017.

[7] U. Lakadiwala1, S. Hirapara, R. Ramani and N. Chaudhary, "Implementation of ALU on FPGA", International Research Journal of Engineering and Technology (IRJET), Vol 03 , Apr-2016.

[8] M. Kumar, S. K. Jha and R. Sharma, "A Case Study: Design of 16 bit Arithmetic and Logical unit using Xillinx 14.7 and Implementation on FPGA Board", International Journal of Engineering Technology Science and Research(IJETSR) Volume 4, Issue 9, Sep-2017.

[9] N. Telagam and N. Kandasami, "Low Power Delay Product 8-bit ALU Design using Decoder and Data Selector", Majlesi Journal of Electrical Engineering, 2018.

[10]] M. Kantawala, "Design and implementation of 8-bit and 16-bit ALU using Verilog language", International Journal of Engineering Applied Sciences and Technology, vol.2, pp. 30- 34, June-2018.

[11] B. Lambda and A. sharma, "A review paper on different multipliers based on their different performance parameters", 2nd International Conference on Inventive Systems and Control (ICISC), 2018.

[12] J. Kalia and V. Mittal, "Int. Journal of Engineering Research and Application", ISSN : 2248-9622, Vol. 7, Issue 5, pp.60-63(Part-4), May-2017.

[13] R. Garg, M. Kaur and D. Bansal, "Review Paper Of Modified Booth Multiplier With Different Methods", IJEDR, Volume 6, Issue 2, ISSN: 2321-9939, 2018.

[14] G. Madhukar, A. L. Kulkarni and J. S. Baligar, "ASIC Implementation of High Speed and Low Power ALU", International Journal of Engineering Research and Technology (IJERT), Vol 08 Issue 07, July-2019.

[15] B. Sakthivel, K. Maheshwari, J. Manojprabakar, S.Nandhini and A.Saravanapriya, "Implementation of Booth Multiplier and Modified Booth Multiplier", International Journal of Recent Trends in Engineering and Research (IJRTER) Conference on Electronics, Information and Communication Systems (CELICS), 2017.

[16] S. Hiremath and D. Koppad, "Alu design using low power GDI standard cells", International Journal of Electrical Engineering and Technology (IJEET), Volume 11, Issue 6, pp. 94-100, August-2020.

[17] A. Deeptha, D. Muthanna, M. Dhrithi, M. Pratiksha and B. S. Kariyappa, "Design and Optimization of 8 bit ALU using Reversible Logic", IEEE International Conference On Recent Trends In Electronics Information Communication Technology, May-2016.

[18] G. C. Ajay Kumar, K. N. Subrahmanyu, S. Ammikkallingal and S. A. Polisetti, "Physical Design, Power and Area Optimization of High Frequency Block at Smaller Technology Node ", 4th International Conference on Recent Trends on Electronics, Information, Communication and Technology (RTE-ICT), May-2019.

[19] G. S. Yuvashree and S. Sumanth, "A 8 Bit ALU Design using Cadence", International Journal for Research in Applied Science and Engineering Technology (IJRASET), ISSN: 2321-9653, Volume 8, Issue 8, Aug-2020.

[20] B. N. Vegha and V. Prakash, "Design and Implementation of 4-Bit ALU for Low-Power using Adiabatic Logic based on FINFET," International Journal of Engineering Research and Technology (IJERT), Vol. 9 Issue 07, July-2020.

[21] J. Kamatam and K. Gajula, "Design of Array Multiplier using Mux Based Full Adder," International Journal of Engineering Research and Technology (IJERT), Vol. 6 Issue 05, May - 2017.

[22] J. R. Shinde and S. J. Shinde, "An Optimization Design Strategy for Arithmetic Logic Unit," Universal Journal of Electrical and Electronic Engineering 6(1): 1-13, 2019.

[23] S. Puranmath, T.M. Manu, A. Kori, S, Meti, "Digital Library creation using standard cells implemented using GPDK 180nm technology," International Journal of Computer Applications, National Conference on Electronics and Communication, 2015.

[24] V. Sreehari, M.B. Srinivas, "Design of Optimized Arithmatic circuits for Multiplier Realization," IEEE Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics, December 2013.

[25] T.S. Monikashree, S. Usharani, Dr. J.S. Baligar, " Design and implementation of full subtractor using CMOS 180nm technology,"International Journal of Science, Engineering and Technology Research, vol.3, May 2014.

[26] K. S. Roy, K. Abhiram, M. A. Sumanth, J. Jaishankar, P. Abhishek, B. N. Prabhat, L.G.Teja. "Development of Graphical User Interface for Open Source VLSI Digital Synthesis Tool Qflow," International Journal of Engineering and Technology, 2018.

[27] A. Mishra, M. Kumar. "Design of a Low Power Dynamic Comparator in 180nm CMOS technology," International Conference on Advances in Computing, Communication Control and Computing, 2018.

[28] M. Mukhedkar, W.B. Pandurang. "A 180nm Efficient Low Power and Optimized Area ALU design usingGate Diffusion Input Technique," International Conference on Data Management, Analytics and Innovation, IEEE, 2017.