A Mini Project Report

On

**8-BIT ALU DESIGNING AND ITS IMPLANTATION ON FPGA**

**Department of Electronics &Telecommunication Engineering**

Submitted By

**Pranay Ramchandra Bedekar**

**Vipul Sudhakar Bhoge**

**Neha Santosh Chavan**

**Shraddha Nilesh Dhayde**

Supervisor

**Prof. Seema Mishra**



**Department of Electronics & Telecommunication Engineering**

**PILLAI COLLEGE OF ENGINEERING**

**New Panvel– 410 206**

**UNIVERSITY OF MUMBAI**

**Academic Year 2021 – 22**



**Department of Electronics & Telecommunication Engineering**

**PILLAI COLLEGE OF ENGINEERING**

**New Panvel– 410 206**

CERTIFICATE

This is to certify that the following students:

**Name Roll No.**

Pranay Bedekar A07

Vipul Bhoge A10

Neha Chavan A16

Shraddha Dhayde A19

have satisfactorily carried out the Mini Project 2Bwork entitled “8-BIT ALU AND ITS IMPLEMENTATION ON FPGA “ for the semester VI of Bachelor of Engineering in Electronics & Telecommunication Engineering of Mumbai University during the Academic Year 2021 – 2022.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Guide** |  | **Head of Department** |
| **(Prof. Seema Mishra)** |  | **Dr. Avinash Vaidya** |
|  |  |  |
|  |  |  |
| **Internal Examiner** |  | **External Examiner** |

**Declaration**

We declare that this written submission for Mini Project 2B entitled “**8-BIT ALU AND ITS IMPLEMENTATION ON FPGA**” represent our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any ideas / data / fact / source in our submission. We understand that any violation of the above will cause for disciplinary action by institute and also evoke penal action from the sources which have thus not been properly cited or from whom paper permission have not been taken when needed.

**Project Group Members:**

Pranay Bedekar & Sign: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Vipul Bhoge & Sign: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Neha Chavan & Sign: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Shraddha Dhayde Sign: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date:

Place:

iii

**Table of Contents**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Abstract................................................................................................................................ | | | | i |
| List of Figures...................................................................................................................... | | | | ii |
| List of Tables....................................................................................................................... | | | | iii |
| **1.** | Introduction................................................................................................................. | | | 1 |
|  | **1.1** | Fundamentals................................................................................................... | | 2 |
|  | **1.2** | Objectives........................................................................................................ | | 2 |
|  | **1.3** | Scope............................................................................................................... | | 2 |
|  | **1.4** | Outline……………………………….............................................................. | | 2 |
| **2.** | Literature Survey......................................................................................................... | | | 3 |
|  | **2.1** | Literature Review ………………………………………................................ | | 3 |
|  | **2.2** | Summary of Literature Survey.…………………………................................ | | 4 |
| **3.** | Proposed System ……………………………............................................................ | | | 5 |
|  | **3.1** | Overview…………………….......................................................................... | | 5 |
| 3.2.1 Hardware and Software Specification……………………………………. 7  **3.3** Implementation…............................................................................................ 8 | **3.2** | Design of 8-bit ALU………............................................................................ | | 6 |
|  |  | **3.3** | Implementation….................................................................. | 10 |
|  |  | 3.3.1 | 8  2-bit ALU …....................................................................................... | 10 |
|  |  | 3.3.2 | 4-bit ALU…...................................................................................... | 10 |
|  |  | 3.3.3 | 8-bit ALU…....................................................................................... | 13 |
| **4.** | Applications……………………................................................................................. | | | 17 |
|  | **4.1** | Complex Operation …………….................................................................... | | 17 |
| **4.3** Future Application………………………………………………………………… 17 | **4.2** | Fast & efficient Sectors…..……..................................................................... | | 17 |
| **5.** | Summary….................................................................................................................. | | | 18 |
| References............................................................................................................................ | | | | 19 |
| Acknowledgement……………………………………………………………………......... | | | | 20 |

## **Abstract**

The main objective of project is to design and verify different operations of Arithmetic and Logical Unit (ALU). We have designed an 8-bit ALU which accepts two 8 bits numbers and the code corresponding to the operation which it has to perform from the user. The ALU performs the desired operation and generates the result accordingly. The different operations that we dealt with are arithmetical, logical and relational. Arithmetic operations include arithmetic addition, subtraction, multiplication and division. Logical operations include AND, OR, NAND, XOR, NOT and NOR. These take two binary inputs and result in output logically operated. The operations like the greater than, less than, equal to, exponential etc. are also included. To implement ALU, the coding was written in VHDL and verified in Modalism. The waveforms were obtained successfully. After the coding was done, the synthesis of the code was performed using Xilinx-ISE. Synthesis translates VHDL code into net list (a textual description). Thereafter, the simulation was done to verify the synthesized code. And it was, then converted into binary format. Components and connections are mapped to CLB design and is placed and routed to fit onto FPGA. User constraint file is generated and also bit file to load design on FPGA when the later was connected to the laptop. Then device was configured and using FPGA verification, debugging was done. Thus, we successfully verified our code using FPGA.

vi

**List of Figures**

|  |  |  |
| --- | --- | --- |
| Figure 3.1 | ALU Architecture | 6 |
| Figure 3.2 | Block Diagram of ALU | 6 |
| Figure 3.3 | Output waveform of 2-bit ALU | 10 |
| Figure 3.4 | Output waveform of 4-bit ALU | 13 |
| Figure 3.5 | Output waveform of 8-bit ALU | 17 |

vii

**List of Tables**

|  |  |  |
| --- | --- | --- |
| Table 2.1 | Summary of Literature survey | 4 |
| Table 3.1 | Hardware Details | 7 |
| Table 3.2 | Software Details | 7 |
| Table 3.3 | Truth Table (2-bit) | 8 |
| Table 3.4   |  |  |  | | --- | --- | --- | | Table 3.5 | Truth Table (8-bit) | 14 | | Truth Table (4-bit) | 10 |

**Chapter 1**

**Introduction**

**1.1 Fundamentals**

In this project, our scope was to design an 8-bit ALU and its implement it on FPGA.

An arithmetic logic unit (ALU) is a digital circuit that performs arithmetic and logical operations. The ALU is a fundamental building block of the central processing unit (CPU) of a computer, and even the simplest microprocessors contain one for purposes such as maintaining timers. The processors found inside modern CPUs and graphics processing units (GPUs) accommodate very powerful and very complex ALUs; a single component may contain a number of ALUs.

Most of a processor's operations are performed by one or more ALUs. An ALU loads data from input registers, an external Control Unit then tells the ALU what operation to perform on that data, and then the ALU stores its result into an output register. The inputs to the ALU are the data to be operated on (called operands) and a code from the control unit indicating which operation to perform. Its output is the result of the computation.

The coding of the ALU has been done in VHDL.VHDL (Very High-Speed Integrated Circuits Hardware Description Language) is a hardware description language used in electronic design automation to describe digital and mixed-signal systems such as field-programmable gate arrays and integrated circuits. It is commonly used to write text models that describe a logic circuit. Such a model is processed by a synthesis program, only if it is part of the logic design. A simulation program is used to test the logic design using simulation models to represent the logic circuits that interface to the design.

After the coding had been completed in VHDL, the synthesis part was done using Xilinx ISE. Xilinx ISE is a software tool for synthesis and analysis of HDL designs, which enables the developers to synthesize ("compile") their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer.

**1.2 Objectives**

Because ALUs can be built in so many ways with wide specifications and since the objective of the class project is to learn the basic of Verilog design, the specifications of the ALU were relaxed. The main objective of the project is to have a working ALU that performs different arithmetic and logic functions for all possible combinations of the inputs. The speed of ALU was not an issue and we wanted it to run at low power.

**1.3 Scope**

The main objective throughout this project was to become familiar with VLSI design. Thus, there were no high goals set for this project to attain. Simply designing a fully functioning 8­bit ALU was considered sufficient for the scope of this course. The goal was to implement the full functionality of the FPGA kit and test it at various speeds to determine the efficiency of the unit.

**1.4 Outline**

We make and study a number of different logic gates were custom designed for this chip. Gates with multiple numbers of inputs were needed. The different logic gates used in the design will be discussed first, followed by the schematic for the ALU.

**Chapter 2**

**Literature Survey**

In this chapter the relevant techniques in literature is reviewed. It describes various techniques used in the work. Identify the current literature on related domain problem. Identify the techniques that have been developed and present the various advantages and limitation of these methods used extensively in literature.

Write advantage and disadvantage of this method. The summary of the literature presented at the end of this Chapter.

**Bennett [1973]:** proved that incorporation of reversible gates keeps a circuit from dissipating any power. In paper “16-bit ALU using Reversible Logic Gates”. The, the number of inputs for a reversible is same as the number of outputs. Not only that, there is one to one mapping between the inputs and outputs of a reversible circuit. Arithmetic logic unit is the vital part of CPU since it allows computer to perform arithmetic and logic operations. In simple sense arithmetic logic unit is a combinational logic circuit having one or more inputs and a single output. Which implies that the present value of output depends only on the present input values only. The complexity of Arithmetic logic unit depends on the processor variations. It can be a simple one or complex structure. Thomson [2010] designed an arithmetic logic unit made up of reversible gates for performing modular operation [1]

**Y Syamala et.al [2011]:** “32-bit ALU using Logic Gates” designed a reversible ALU for performing logic and arithmetic operations here in thx is paper we propose a new design of ALU made up of reversible gates with better power saving property. Venu gopal G: In this work reversible ALU for a computing device is designed for performing one arithmetic operation and three logic operation. Our design shows better quality in the design parameters like number of gates number of garbage outputs and quantum cost. Earlier, the ALU and full adder circuit was implemented for area and delay, each with their distinct features that bring about best area and delay. Out of them some was briefly described to give us an idea of earlier work and different optimization techniques. Past work gives us a suggestion about its operation, performance, design and simulation issues. In G. Karthik Reddy, the 1was design using the Pass Transistor Logic (PTL) technology and 1-bit full adder is used based on low power 6Transistors. ALU circuit was designed using 6Tfull adder cell and Pass Transistor Logic which was basedon logic blocks such as AND, OR, XOR. The simulation was done on 65nm single n-well CMOS bulk technology, in virtuoso platform of cadence tool. As per the nature of work, they have applied the technique which may have faces the problem in designing the ALU with such technology and it was also not easy to reduce the power. In Geetanjali1 and Nishant Tripathi2 , the was implemented using VHDL and simulation was carried out by using modalism 5.4a tool. The ALU was designed to perform the arithmetic operation such as addition, subtraction, increment, decrement, transfer and logical operation such as AND, NOT, OR, NAND, NOR, EXNOR. How the operation take place is shown by behavioral modeling style, the behavioral capabilities of VHDL was more powerful and more convenient for this design.[2]

**2.1 Literature Summary**

A literature review is an objective, critical summary of published research literature relevant to a topic under consideration for research. The summary is presented here.

Table 2.1 Summary of literature survey

|  |  |  |
| --- | --- | --- |
| **SN** | **Paper** | **Advantages and Disadvantages** |
| 1. | **Bennett [1973]**  [1]  16-bit ALU using Reversible Logic Gates | Advantages: Learning of profile is made easy.  Disadvantages: making this in very complicated way. |
| 2. | **Y Syamala et.al [2011]** [2]  32-bit ALU using Logic Gates | Advantages: The web site based on this has shown great performance.  Disadvantages: cost is more in this project. |

**Chapter 3**

**Proposed System**

# 3.1 Overview

An arithmetic logic unit (ALU) is a digital circuit used to perform arithmetic and logical operations. It represents the fundamental component of a computer's CPU. Modern processors contain very powerful and complex ALUs. In addition to ALUs, modern CPUs contain a control unit (CU). Most CPU operations are performed by one or more ALUs, which load data from the input registers. A record is a small amount of storage available as part of a CPU. The control unit tells the ALU the operation that will be performed on this data and the ALU stores the result in an output record. The control unit moves the data between these registers, the ALU and the memory. An ALU performs basic arithmetic and logic operations. Examples of arithmetic operations are addition, subtraction, multiplication, and division. Examples of logical operations are comparisons of values such as NOT, AND, and OR.

The entries of an ALU are the data to be operated, called operands, and a code that indicates the operation to be performed; the output of the ALU is the result of the operation performed. In many designs, the ALU also has state inputs or outputs, or both, that transmit information about a previous operation or a current operation, respectively, between the ALU and the status registers External. An ALU is a combinational logic circuit, which means that its outputs will change asynchronously in response to input changes. In normal operation, stable signals are applied to all ALU inputs and when enough time is passed (called "propagation delay") for the signals to propagate through the ALUs, the result of the ALU operation appears. in the ALU the exits. The external circuits connected to the ALU are responsible for ensuring the stability of the ALU's input signals throughout the operation and for allowing sufficient time for the signals to propagate through the ALU beforehand. Shows the result of the ALU. In general, external circuits control an ALU by applying signals to its inputs. Typically, the external circuits use sequential logic to control the operation of the ALU, which is reinforced by a clock signal of a sufficiently low frequency to guarantee sufficient time for the ALU outputs to stabilize under the conditions of the most worst case also.

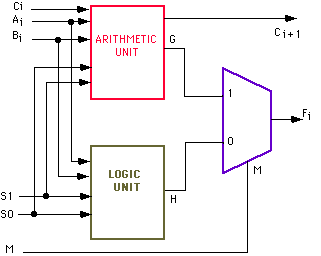


Fig.3.1 ALU Architecture

ALU is getting smaller and more complex nowadays to allow the development of a more powerful system but a smaller computer. However, some limiting factors slow down the development of IC chip more complex and are IC manufacturing technology, designer productivity and design costs. The growing demand for large scale high speed integration (VLSI) can be addressed at different levels of design, like the architecture, the circuit, the design and the level of process technology. At the level of the design of the circuit, there is considerable potential for speed improvement through the appropriate choice of a logical style for implementation combinatorial circuits. This is because all the important parameters that govern the speed switching capability, Transition activity and short-circuit currents are strongly influenced by the chosen logical style. That depends on application, the type of circuit to be implemented and the design technique used, different aspects of the performance be important In the past, such parameters as low power dissipation, small area and low cost issues of concern, while speed considerations attract the attention of the scientific community associated with the VLSI design.

**3.2 DESIGN OF 8 BIT ALU**

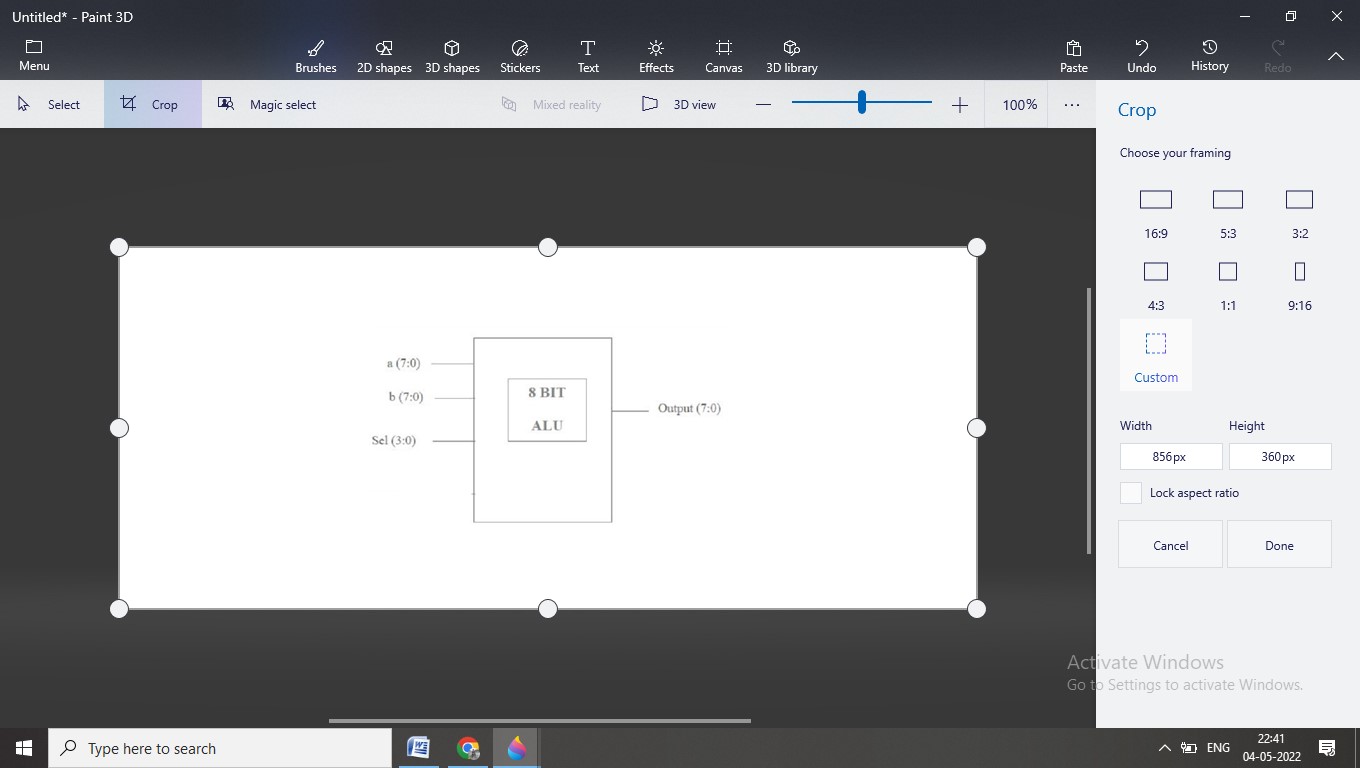


Fig 3.2 Block diagram of 8-bit ALU

This ALU operate on 8 bit input. It performs arethematic and logical operations. This gives appropriate output. 8 bit ALU performs 8 operations. There are 2 data inputs and one select line. According to select line input appropriate operation is performed between 2 inputs. Output of this 8 bit ALU is connected between ROM and RAM. A number of basic arethematic and bitwise logic functions are performed in the ALU. ALU can be used in complex operations.

**3.2.1 Hardware and Software Specifications**

The experiment setup is carried out on a computer system which has the different hardware and software specifications as given in Table 3.1 and Table 3.2 respectively.

Table 3.1 Hardware details

|  |  |
| --- | --- |
| Hardware | Model |
| FPGA kit | XILINX FPGA |
| Cables | - |

Table 3.2 Software details

|  |  |
| --- | --- |
| Software | Operating System |
| XILINX | XILINX 14.7 |

**3.3 Implementations**

**3.3.1 2-bit ALU**

Table 3.3 Truth Table (2-bit)

|  |  |  |  |
| --- | --- | --- | --- |
| Sr.no | Opcode | Operation | Description |
| 1 | 00 | A+B | Addition |
| 2 | 01 | A-B | Subtraction |
| 3 | 10 | A\*B | Multiplication |
| 4 | 11 | A/B | Division |

//VERILOG PROGRAMME OF 2-BIT ALU

module 2-bit(A, B, op, y);  
// initailize input and output  
    input [1:0] A;  
    input [1:0] B;  
    input [1:0] op;  
    output [1:0] y;  
  
reg [1:0] y;  
always @(\*)  
begin  
case(op)  
2'b00:y=A+B;  //addition  
2'b01:y=A-B;  //substration  
2'b10:y=A\*B;  //multiplication  
2'b11:y=A/B;  //division  
  
endcase  
end  
endmodule

// TESTBENCH PROGRAMME OF 2-BIT ALU

module 2-bit\_tb\_v;  
  
// Inputs  
reg [1:0] A;  
reg [1:0] B;  
reg [1:0] op;  
  
// Outputs  
wire [1:0] y;  
  
// Instantiate the Unit Under Test (UUT)  
2-bit uut (A,B,op,y);  
  
  
initial begin  
op=2'b00; A=2'b01; B=2'b01;  
#10;  
op=2'b01; A=2'b01; B=2'b01;  
#10;  
op=2'b10; A=2'b01; B=2'b01;  
#10;  
op=2'b11; A=2'b01; B=2'b01;  
#10;  
end  
endmodule

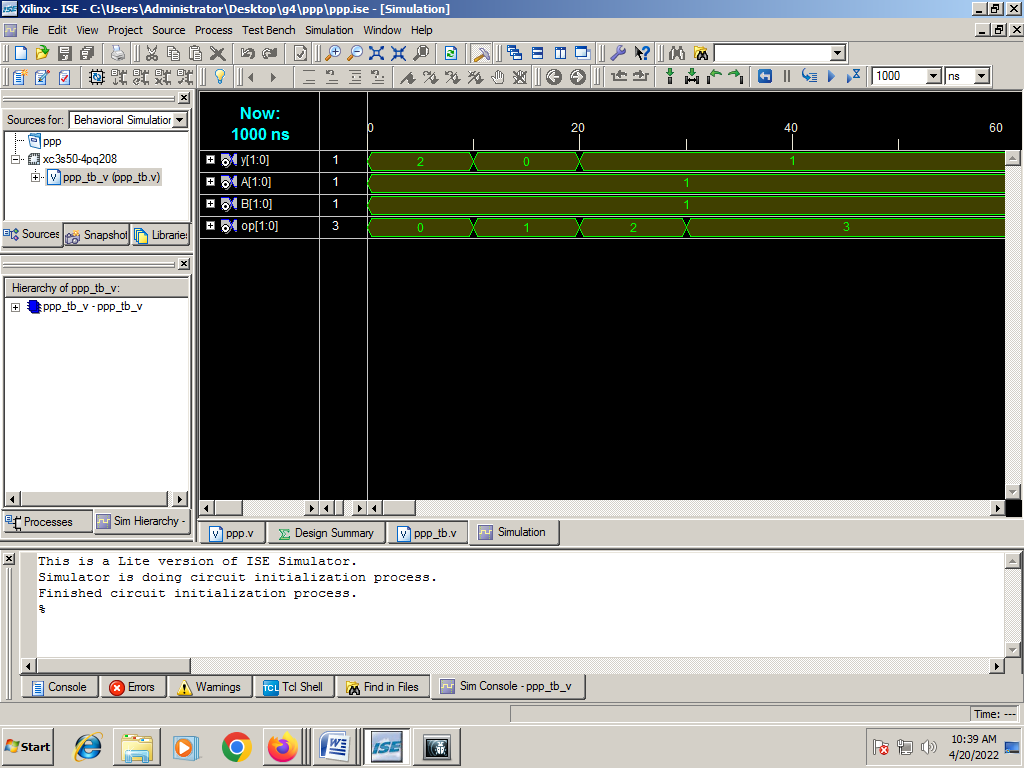


Fig 3.3 Output Waveform of 2-bit ALU

**3.3.2 4-bit ALU**

Table 3.4 Truth Table (4-bit)

|  |  |  |  |
| --- | --- | --- | --- |
| Sr.no | Opcode | Operation | Description |
| 1 | 000 | A+B | Addition |
| 2 | 001 | A-B | Subtraction |
| 3 | 010 | A\*B | Multiplication |
| 4 | 011 | A/B | Division |
| 5 | 100 | A&B | Bitwise AND |
| 6 | 101 | A|B | Bitwise OR |
| 7 | 110 | A&&B | Logical AND |
| 8 | 111 | A||B | Logical OR |

//VERILOG PROGRAMME OF 4-BIT ALU

module 4-bit(A,B,op,y);  
    input [3:0] A;  
    input [3:0] B;  
    input [3:0] op;  
    output [3:0] y;  
     
reg [3:0] y;  
always @(\*)  
begin  
case(op)  
3'b000:y=A+B;  //addition  
3'b001:y=A-B;  //substration  
3'b010:y=A\*B;  //multiplication  
3'b011:y=A/B;  //division  
3'b100:y=A&B;  //bitwise AND  
3'b101:y=A|B;  //bitwise OR  
3'b110:y=A&&B;  //logical AND

3'b111:y=A||B;  //logical OR  
endcase  
end  
endmodule

// TESTBENCH PROGRAMME OF 4-BIT ALU

module 4-bit\_tb;  
// Inputs  
reg [3:0] A;  
reg [3:0] B;  
reg [3:0] op;  
  
// Outputs  
wire [3:0] y;  
  
// Instantiate the Unit Under Test (UUT)  
4-bit uut (  
.A(A),  
.B(B),  
.op(op),  
.y(y)  
);  
  
initial begin  
op=3'b000; A=3'b001; B=3'b001;  
#10;  
op=3'b001; A=3'b001; B=3'b001;  
#10;  
op=3'b010; A=3'b001; B=3'b001;  
#10;  
op=3'b011; A=3'b110; B=3'b011;  
#10;  
op=3'b100; A=3'b001; B=3'b001;  
#10;  
op=3'b101; A=3'b001; B=3'b001;  
#10;  
op=3'b110; A=3'b111; B=3'b001;  
#10;  
op=3'b110; A=3'b001; B=3'b001;  
#10;

op=3'b111; A=3'b001; B=3'b001;  
#10;  
  
  
end  
endmodule

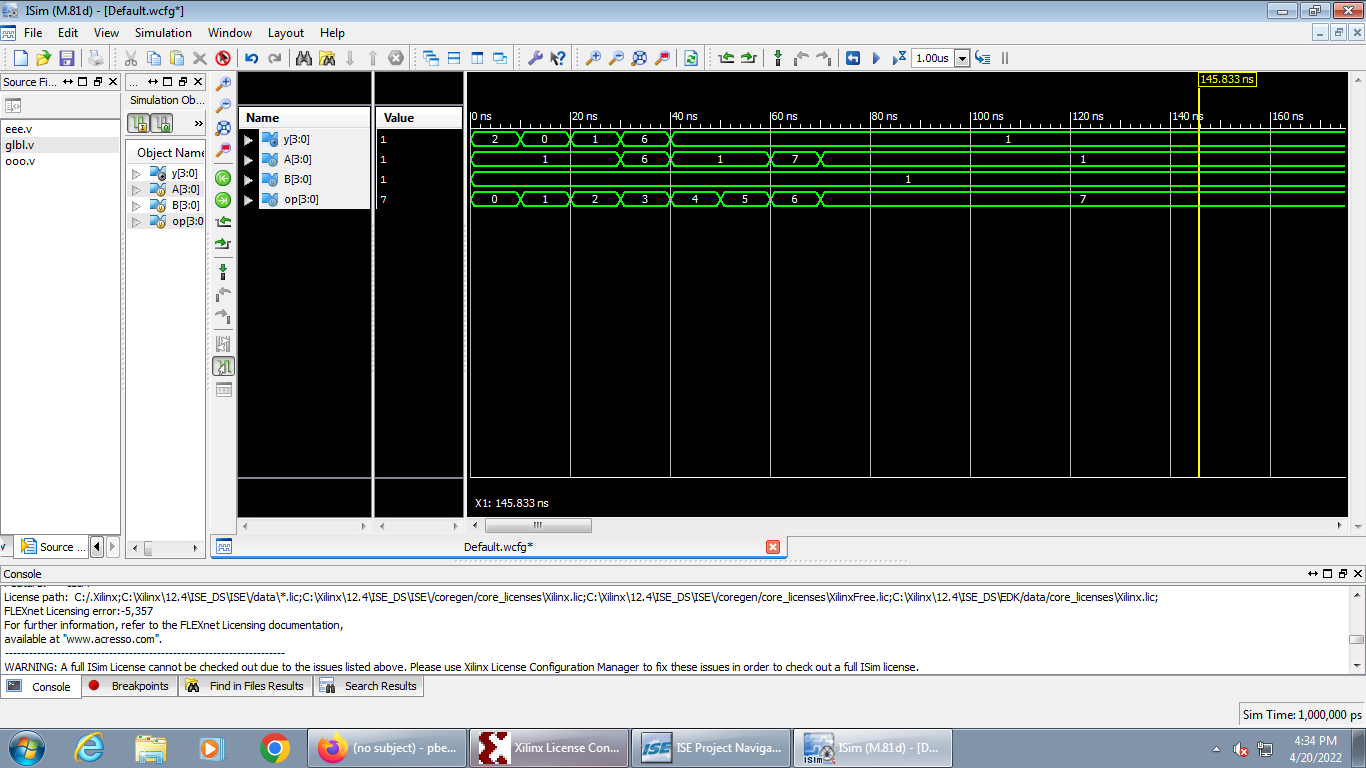


Fig 3.4 Output Waveform of 4-bit ALU

**3.3.3 8-bit ALU**

Table 3.5 Truth Table (8-bit)

|  |  |  |  |
| --- | --- | --- | --- |
| Sr.no | Opcode | Operation | Description |
| 1 | 0000 | A+B | Addition |
| 2 | 0001 | A-B | Subtraction |
| 3 | 0010 | A\*B | Multiplication |
| 4 | 0011 | A/B | Division |
| 5 | 0100 | A%B | Modulus |
| 6 | 0101 | A&B | Bitwise AND |
| 7 | 0110 | A|B | Bitwise OR |
| 8 | 0111 | A&&B | Logical AND |
| 9 | 1000 | A||B | Logical OR |
| 10 | 1001 | A^B | Bitwise X-OR |
| 11 | 1010 | A>>B | Right Shift |
| 12 | 1011 | A<<B | Left Shift |
| 13 | 1100 | A>B | Greater Than |
| 14 | 1101 | A<B | Lower Than |
| 15 | 1110 | A+1 | Increment |
| 16 | 1111 | A-1 | Decrement |

//VERILOG PROGRAMME OF 8-BIT ALU

module design (A,B,op,alu\_out);  
  input [7:0] A,B;  
  input [3:0] op;  
  output [7:0] alu\_out;  
  reg [7:0] alu\_out;  
  always @(\*)  
    begin  
      case (op)  
        4'b0000: alu\_out=A+B;  
        4'b0001: alu\_out=A-B;  
        4'b0010: alu\_out=A\*B;  
        4'b0011: alu\_out=A/B;  
        4'b0100: alu\_out=A%B;  
        4'b0101: alu\_out=A&B;  
        4'b0110: alu\_out=A|B;  
        4'b0111: alu\_out=A&&B;  
        4'b1000: alu\_out=A||B;  
        4'b1001: alu\_out=A^B;  
        4'b1010: alu\_out=A>>B;  
        4'b1011: alu\_out=A<<B;  
        4'b1100: alu\_out=A>B;  
        4'b1101: alu\_out=A<B;  
        4'b1110: alu\_out=A+1;  
        4'b1111: alu\_out=A-1;  
        default: alu\_out=0;  
      endcase  
    end  
endmodule

// TESTBENCH PROGRAMME OF 8-BIT ALU

module testbench();  
  reg [7:0] A,B;  
  reg [3:0] op;  
  wire [7:0] alu\_out;  
   
  design testbench (A,B,op,alu\_out);  
  initial  
    begin  
      op=4'b0000; A=7'b00000001; B=7'b00001101;  
      #10;  
      op=4'b0001; A=7'b00001111; B=7'b00001010;  
      #10;  
      op=4'b0010; A=7'b00000001; B=7'b00001101;  
      #10;  
      op=4'b0011; A=7'b00001111; B=7'b00000011;  
      #10;  
      op=4'b0100; A=7'b00000001; B=7'b00001101;  
      #10;  
      op=4'b0101; A=7'b00000001; B=7'b00001101;  
      #10;  
      op=4'b0110; A=7'b00000001; B=7'b00001101;  
      #10;  
      op=4'b0111; A=7'b00000001; B=7'b00001101;  
      #10;  
      op=4'b1000; A=7'b00000001; B=7'b00001101;  
      #10;  
      op=4'b1001; A=7'b00000001; B=7'b00001101;  
      #10;  
      op=4'b1010; A=7'b00000001; B=7'b00001101;  
      #10;  
      op=4'b1011; A=7'b00001111; B=7'b00001101;  
      #10;  
      op=4'b1100; A=7'b00000001; B=7'b00001101;  
      #10;  
      op=4'b1101; A=7'b00000001; B=7'b00001101;  
      #10;  
      op=4'b1110; A=7'b00000001; B=7'b00001101;  
      #10;  
      op=4'b1111; A=7'b00000001; B=7'b00001101;  
      #10;  
    end

endmodule

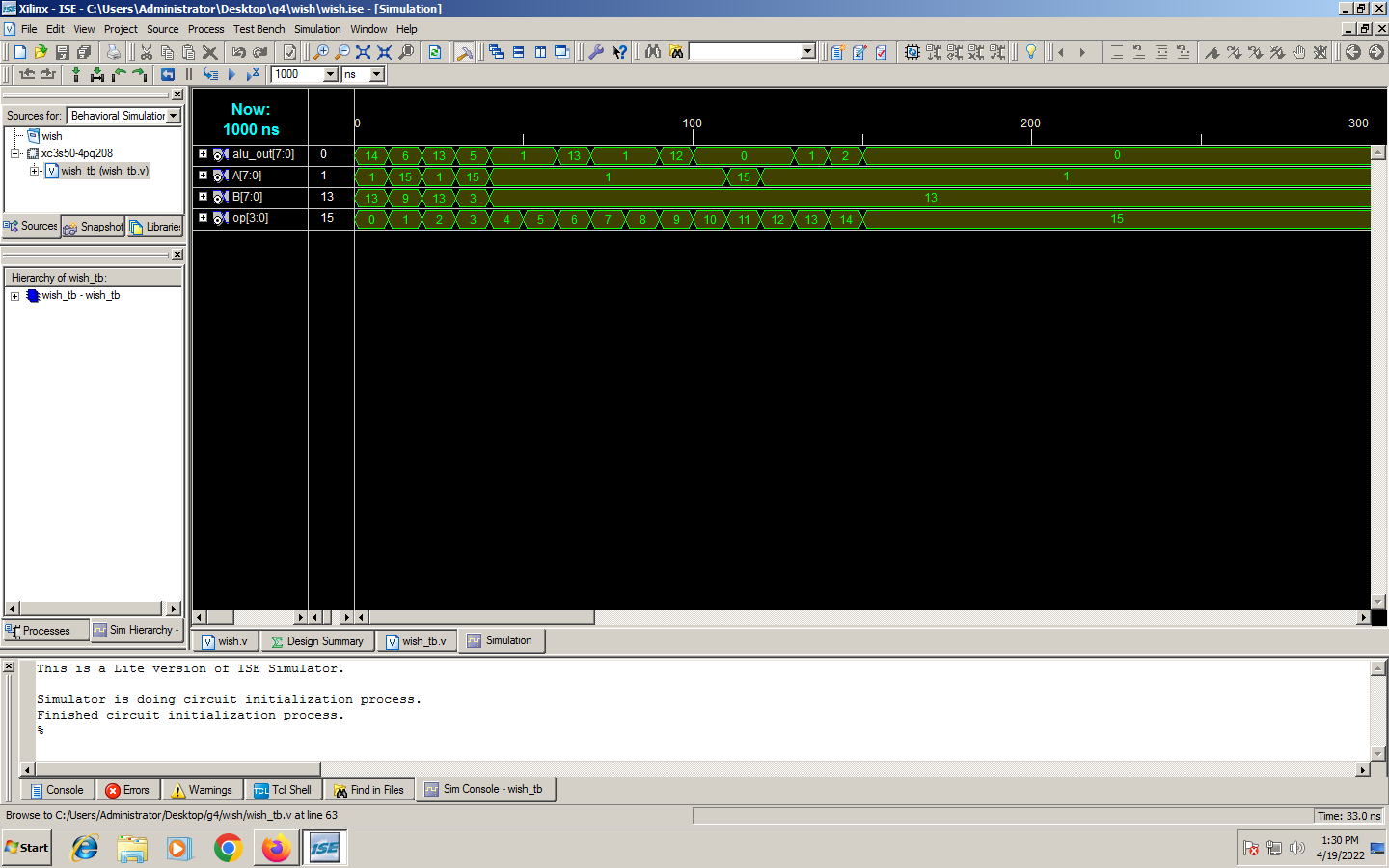


Fig 3.5 Output Waveform of 8-bit ALU

**Chapter 4**

**Applications**

There are various applications of this domain system. The application is listed here.

**4.1 complex operations**

The ALU performs simple addition, subtraction, multiplication, division, and logic operations, such as OR and AND, etc.

**4.2 Fast and efficient sectors**

ALU plays a vital role in CPU’s, Microprocessors and Microcontroller so, ALU is the heart of the processor, the central processing unit speed is greatly depends upon the ALU so we need to have fast and efficient ALU.

### 4.3 Future applications

The ALU described here can be further refined to make more advanced-level projects with some more complex logics in order to handle complex calculations in the CPUs.

**Chapter 5**

**Summary**

In this report, the study of different bits of ALU is presented. We successfully design 2-bit ALU, and 4-bit ALU as well as successfully design 8-bit ALU.

**References**

[1]. Ravindran N. and Lourde R. Mary (2015).An Optimum VLSI Design for 16Bit ALU. IEEE International Conference on Information and Communication Technology Research, (pp52-55).

[2]. Chouhan Ankit (2014).16-Bit Arithmetic and Logic Unit Design Using Mixed Type of Modelling In VHDL. IJERA International Journal of Engineering Research and Applications, (pp – 213- 215).

[3]. B S Premananda, Ravindranath Y M (2013) Design and Synthesis of 16-bit ALU using Reversible Logic Gates. IJARCCE International Journal of Advanced Research in Computer and Communication Engineering, (pp-4137- 4141).

[4]. N. Naik Chandni and M. Velvani Vaishnavi. VLSI Based 16 Bit ALU with Interfacing Circuit. International Journal of Innovative and Emerging Research in Engineering Volume 2, (pp-65-69).

[5]. Zhou Yu and Guo Hui (2008).Application Specific Low Power ALU Design. IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, (pp – 214-220).

[6]. Sobotka Jiří and Zeman Václav (2011).Application of FMEA Procedure for ALU Unit Testing. IEEE, (pp- 428-432).

**Acknowledgement**

We sincerely thank our faculty guide Prof. SEEMA MISHRA, Professor, Pillai’s of Engineering & Technology for guiding us in every way she could. In our project “DESIGN OF ALU AND ITS DESIGNING OF FPGA”. We would also like to express our gratitude to Mr. Ashutosh Gupta and Mrs Nidhi Gaur for their magnificent help and support throughout the project. Last but not the least, we wish to avail this opportunity to express our gratitude and love to our friends and our beloved parents for their moral support, strength, help and for everything

Pranay Bedekar

Vipul Bhoge

Neha Chavan

Shraddha Dhayde