A

MINI PORJECT REPORT

ON

“8 - Bit ALU using Verilog”

FOR PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE

MINI PROJECT

OF

T.E. E&TC – 2019 Course, SPPU, Pune.

By

Aarti R. Jagtap	**-**	**32359**
Niveda J. Pagdhare	**-**	**32364**
Prachi K. Vitekar	**-**	**32365**

GUIDE

Prof. H. B. Mali

Department Of

ELECTRONICS AND TELECOMMUNICATION ENGINEERING

PUNE INSTITUTE OF COMPUTER TECHNOLOGY

PUNE – 43

ACADEMIC YEAR: 2022 - 2023

Department of Electronics and Telecommunication Engineering

Pune Institute of Computer Technology, Pune – 43

CERTIFICATE

This is to certify that the Mini Project report entitled

"8 BIT ALU USING VERILOG"

has been successfully completed by

Aarti R. Jagtap	**-**	**32359**
Niveda J. Pagdhare	**-**	**32364**
Prachi K. Vitekar	**-**	**32365**

Is a bonafide work carried out by them under the supervision of Prof. H. B. Mali and it is approved for the partial fulfilment of the requirements for the Mini Project of T.E. E&TC – 2019 Course of the Savitribai Phule Pune University, Pune.

Prof. H. B. Mali — Project Guide

Dr. Mousami V. Munot — HOD, E&TC Dept. of E&TC

Place: Pune

Date:

ACKNOWLEDGEMENT

We would like to express my sincere gratitude to all the people who have supported us in the completion of this project. Firstly, we would like to thank our supervisor Prof. H. B. Mali, for their guidance, support and valuable feedback throughout the project. Their expertise and insights have been invaluable to us, and we are grateful for their encouragement and advice.

We would like to express our gratitude towards our H.O.D. Dr. Mousami V. Munot and faculty of E&TC department for their kind cooperation and encouragement which helped us in completion of this project.

Furthermore, we would like to thank Pune Institute of Computer Technology for providing us with the necessary resources and equipment to carry out the project. Their support and generosity have been crucial to the completion of the project.

We find no words to acknowledge the sacrifice, help and inspiration rendered by our parents to take up this study. Our thanks and appreciation also go to our colleagues in developing the project and people who have willingly helped us out with their abilities. With thanks to all,

Aarti R. Jagtap - 32359
Niveda J. Pagdhare - 32364
Prachi K. Vitekar - 32365

ABSTRACT

We have implemented the 8-bit Arithmetic Logic Unit (ALU) using Verilog project aims to design and implement an ALU on a Field Programmable Gate Array (FPGA) board. The ALU will be capable of performing arithmetic and logical operations on two 8-bit binary numbers. The project will be implemented using Verilog, a hardware description language used to model digital systems. The ALU will be designed to support the following operations: addition, subtraction, AND, OR, XOR, XNOR, NAND and NOR. The ALU will also include a flag register to indicate overflow, carry, and negative results. The project will be implemented on an FPGA board. The completed project will demonstrate the ability to design and implement an ALU using Verilog and FPGA technology. The ALU will be tested and verified to ensure that it performs the expected operations correctly. This project will provide a solid foundation for further exploration of digital logic design using Verilog and FPGA technology.

List of Figures:

List of Tables:

Contents

FEASIBILITY STUDY REPORT

Tools required	Testing possibility	Controller	Any Cost
Hardware Spartan3 FPGA board	Yes		No
Software Xilinx ISE 14	Software Yes		No
Tools available within campus or outside	Sensors required	Signal conditioning if any	
Within the Campus VLSI Laboratory	None	No	
Applications if any	PCB design and fabrication	Datasheets/ application notes available	
CPU Processing, Digital Signal Processing, Cryptography, Control systems.	No	https://www.xilinx.com/products/silicon-devices/fpga/xa-spartan-3e.html	

Table 1 Feasibility Study

CHAPTER 1
INTRODUCTION

BACKGROUND STUDY

The 8-bit Arithmetic Logic Unit (ALU) is a fundamental component of digital systems that performs arithmetic and logical operations on binary numbers. An ALU is typically used in microprocessors, digital signal processors, and other digital systems.

Verilog is a hardware description language used to model digital systems. It allows designers to describe the behaviour and structure of digital circuits at a high level of abstraction. Verilog code can be compiled and synthesised to create a hardware implementation of the digital system.

FPGAs are integrated circuits that can be programmed and reprogrammed to perform a specific function. They contain programmable logic blocks and interconnects that can be configured to create custom digital circuits.

The implementation of an 8-bit ALU using Verilog and an FPGA involves designing the ALU using Verilog, synthesising the Verilog code into a netlist, and then programming the FPGA to implement the netlist. The Verilog code must be tested and verified using a test bench to ensure that it performs the expected operations correctly.

The project involves exploring the fundamentals of digital logic design, Verilog syntax, and FPGA programming. It provides an opportunity to gain hands-on experience in designing and implementing a digital system using Verilog and an FPGA. The completed project can be used as a building block for more complex digital systems and can be extended to support additional operations and functionality.

RELEVANCE

The design and implementation of an 8-bit Arithmetic Logic Unit (ALU) using Verilog and an FPGA is a relevant project in the field of digital systems design. The project has several applications in the computing industry, including microprocessors, digital signal processors, and other digital systems.

The project provides hands-on experience in digital logic design, Verilog syntax, and FPGA programming. It helps to develop skills in circuit design, simulation, and verification. Additionally, it provides a strong foundation for further exploration of digital systems and FPGA programming.

In the current era of technology, the use of programmable devices, such as FPGAs, has increased significantly due to their flexibility and reusability. By implementing an 8-bit ALU using Verilog and an FPGA, the project demonstrates the use of programmable devices to create custom digital circuits.

In conclusion, the 8-bit ALU using Verilog and FPGA project is relevant and valuable in the field of digital systems design. It provides practical knowledge and skills in circuit design, simulation, and verification, and demonstrates the use of programmable devices to create custom digital circuits.

LITERATURE SURVEY

In the literature, there are several studies related to the design and implementation of ALUs using Verilog and FPGAs. Some of these studies are discussed below:

1) In a study published in the International Journal of Advanced Research in Computer Science and Software Engineering, the authors designed an 8-bit ALU using Verilog and implemented it on an FPGA. The ALU supported arithmetic and logical operations, and the authors verified its functionality using simulation and testing. The study demonstrated the effectiveness of Verilog and FPGAs in digital circuit design.

2) In a study published in the Journal of Low Power Electronics and Applications, the authors proposed an energy-efficient design of an 8-bit ALU using Verilog and FPGA technology. The authors used a low-power adder and a pipelined architecture to reduce power consumption. The design was implemented on an FPGA, and the authors demonstrated its functionality using simulation and testing.

3) In a study published in the Journal of Electronic Testing: Theory and Applications, the authors designed a high-speed 8-bit ALU using Verilog and FPGA technology. The authors used parallel processing techniques and a pipelined architecture to improve performance. The design was implemented on an FPGA, and the authors demonstrated its functionality using simulation and testing.

4) In a study published in the International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, the authors designed an 8-bit ALU using Verilog and FPGA technology. The ALU supported arithmetic and logical operations and included a flag register to indicate overflow, carry, and negative results. The authors demonstrated the functionality of the design using simulation and testing.

These studies demonstrate the versatility and effectiveness of Verilog and FPGA technology in the design and implementation of ALUs. They provide valuable insights into the design and optimization of ALUs for different applications, including low-power and high-speed operations. The literature survey highlights the importance of simulation and testing to verify the functionality of the designs.

MOTIVATION

Designing an 8-bit ALU using Verilog can be an excellent project for those interested in digital design, computer architecture, and computer engineering. Verilog is a hardware description language used to design and simulate digital circuits. An 8-bit ALU is a crucial building block of a CPU, so designing and implementing one using Verilog will help you gain a deeper understanding of computer architecture. The motivation behind an 8-bit ALU project using Verilog is to learn and implement a fundamental building block of a computer system using a hardware description language.

AIM AND OBJECTIVES

The aim of an 8-bit ALU project using Verilog is to design and implement an 8-bit Arithmetic Logic Unit (ALU) using Verilog hardware description language. An ALU is a fundamental component of a computer's central processing unit (CPU) that performs arithmetic and logical operations on binary numbers.

The main **objectives** of the project include:

1) Understanding the basic principles of digital logic design and computer architecture.
2) Familiarising yourself with the Verilog hardware description language.
3) Designing an 8-bit ALU that can perform various arithmetic and logical operations on two 8-bit binary numbers.
4) Implementing the design using Verilog and simulating it to verify its functionality.
5) Testing and debugging the implemented design to ensure that it meets the project's requirements.
6) Synthesising the Verilog code and generating a bitstream file that can be used to program a Field Programmable Gate Array (FPGA) or Application Specific Integrated Circuit (ASIC).
7) Demonstrating the functionality of the implemented 8-bit ALU through various test cases.

Overall, the project aims to provide hands-on experience in designing and implementing a fundamental building block of a computer system using Verilog, which will help in understanding the principles of digital logic design and computer architecture.

TECHNICAL APPROACH

The technical approach for an 8-bit ALU project using Verilog can be divided into several steps:

1) **Defining the project requirements**: The first step is to define the project requirements, such as the types of arithmetic and logical operations the ALU should support and the input and output formats.

1) **Designing the ALU:** The next step is to design the ALU based on the project requirements. The ALU design will include a block diagram of the ALU's various components, such as adders, logic gates, and multiplexers.

2) **Creating a testbench:** Once the design is complete, a testbench should be created to simulate the ALU's functionality. The testbench will include a series of test cases to verify that the ALU meets the project requirements.

3) **Writing Verilog code:** After the testbench is created, the Verilog code for the ALU should be written. The Verilog code will include modules for each of the ALU's components and a top-level module that connects them together.

4) **Simulating the design:** The Verilog code should be simulated using the testbench to ensure that the ALU functions correctly. Any errors or bugs should be identified and corrected at this stage.

5) **Synthesising the design**: After verifying the design's functionality, the Verilog code should be synthesised to generate a gate-level netlist that can be used to program an FPGA or ASIC.

6) **Testing the implemented design:** The implemented design should be tested using various test cases to ensure that it meets the project requirements.

   Overall, the technical approach for an 8-bit ALU project using Verilog involves defining the project requirements, designing the ALU, creating a testbench, writing Verilog code, simulating the design, synthesising the code, and testing the implemented design.

CHAPTER 2
DESCRIPTION OF PROJECT

2.1 INTRODUCTION

An 8-bit ALU (Arithmetic Logic Unit) is a digital circuit that performs arithmetic and logical operations on 8-bit binary numbers. In this Verilog project, we have designed an 8-bit ALU that can perform addition, subtraction, AND, OR, XOR, NAND, NOR and XNOR operations.

The design will use behavioural Verilog modelling, which means we will describe the functionality of the ALU using a high-level description of its operation rather than specifying its structure.

The ALU will have two 8-bit input ports A and B, and a 3-bit select line i.e ALU_Sel that will select the operation to be performed. The output port ALU_Out will be a 8-bit result, and we have one port for indicating carry or borrow.

2.2 BLOCK DIAGRAM

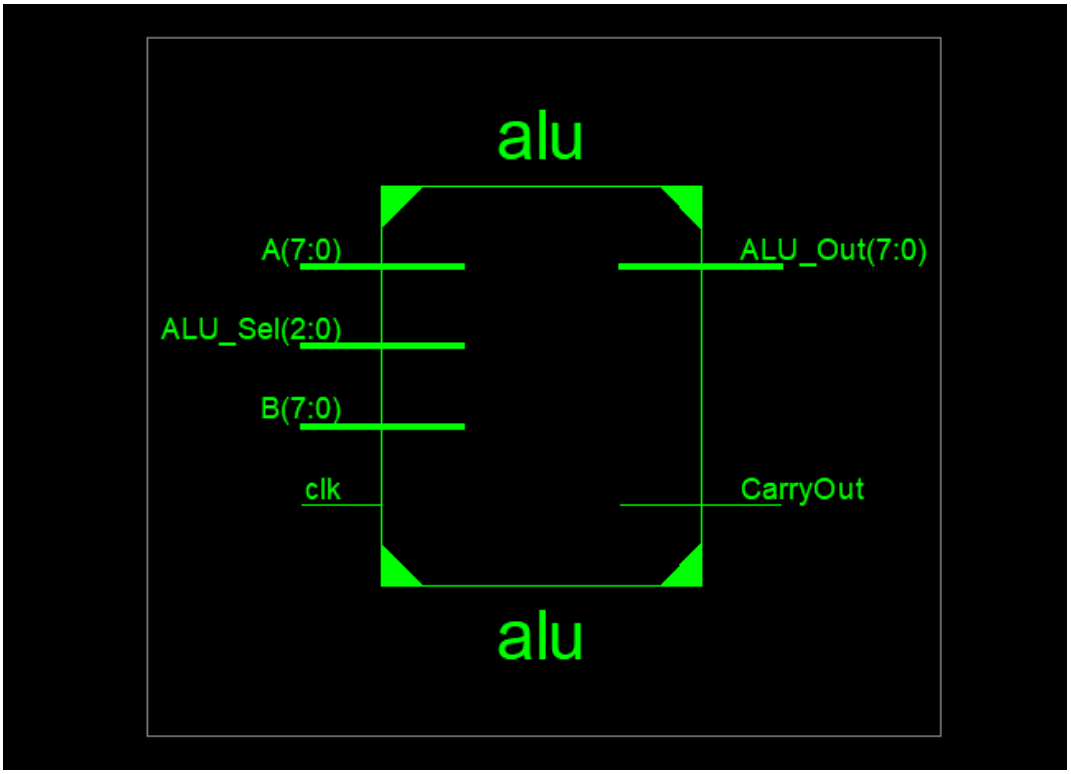

Figure 1 Block Diagram

2.2 REQUIREMENTS

- HARDWARE.
  - ✓ A windows 8 or 10 computer with sufficient processing power and memory to run Verilog software.
  - ✓ Field Programmable Gate Array (FPGA) board.
  - ✓ Downloading cables.
- SOFTWARE.
  - ✓ A Verilog simulator or synthesis tool- Xilinx ISE is needed to synthesize the Verilog code and program it onto an FPGA or development board.
  - ✓ A text editor or IDE is needed to write and edit the Verilog code.

CHAPTER 3
SYSTEM DESIGN

3.1 DETAILED STRUCTURE.

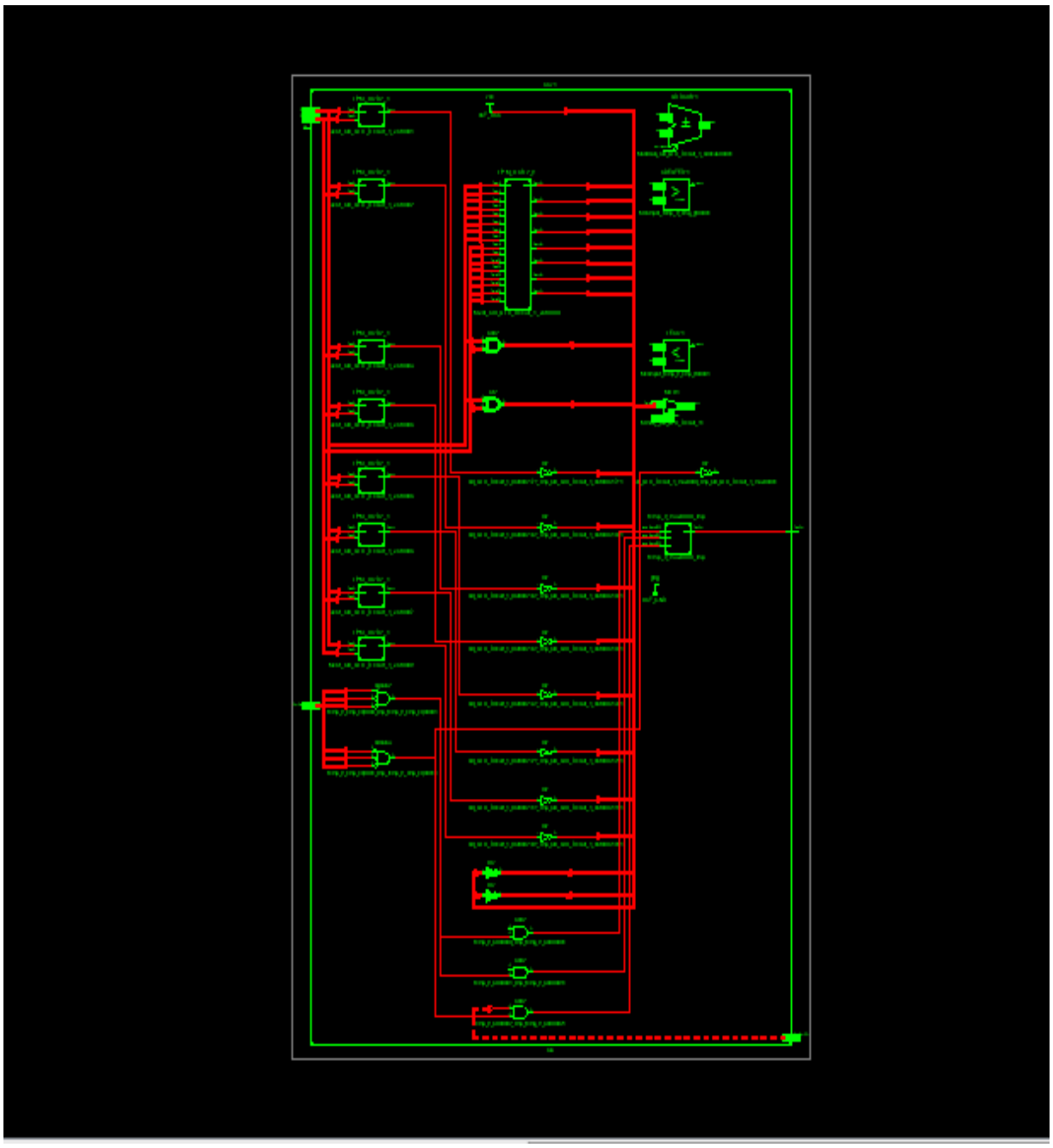

Figure 2 RTL Schematic Structure

3.2 SIGNIFICANCE OF PROJECT

The design and implementation of an 8-bit ALU using Verilog is a significant project that provides valuable learning experience for students or engineers interested in digital systems design. An ALU is a fundamental component of digital systems that performs arithmetic and logical operations on binary data. An 8-bit ALU is capable of processing 8 bits of data at a time and can be used in a variety of applications, such as microprocessors, calculators, and other digital devices. The design of an 8-bit ALU requires a deep understanding of digital circuits, including adders, subtractors, logical gates, and multiplexers. By designing an 8-bit ALU using Verilog, students or engineers can gain practical experience in digital design and Verilog programming, which are important skills in the field. Furthermore, the project involves solving complex problems and overcoming challenges, which helps to enhance problem-solving skills and develop an analytical approach to design. The practical demonstration of skills in digital system design and Verilog programming provided by the project can enhance career prospects for students or engineers interested in the field. Overall, the design and implementation of an 8-bit ALU using Verilog is a significant project that provides valuable learning experience, enhances problem-solving skills, and improves future career prospects.

3.3 PROGRAM

Main code

```
module alu
(       input [7:0] A,B,
            input [2:0] ALU_Sel,
            output [7:0] ALU_Out,
            output CarryOut,
                        input clk
                        //input reset,
                        //output
);
        reg[8:0] ALU_Result;
        reg[8:0] temp;
        reg[7:0] b_c;
        reg[8:0] sub_r;
         reg[2:0] count;

        assign ALU_Out=ALU_Result[8:0];
        assign CarryOut = temp[8];


        always @(*) begin
              case(ALU_Sel)
                 3'b000:
                    ALU_Result={1'b0,A}+{1'b0,B}; // Addition
                 3'b001:
                          ALU_Result={1'b0, A} - {1'b0,B}; // Subtraction
                 3'b010:
                    ALU_Result=A&B;  // AND
                 3'b011:
                    ALU_Result=A|B;  // OR
                 3'b100:
                    ALU_Result=A^B;  // XOR
                 3'b101:
                    ALU_Result=A^~B; //XNOR
                 3'b110:
                    ALU_Result=~(A&B); //nand
```

```
            3'b111:
               ALU_Result=~(A|B); //nor
            //default: ALU_Result=A+B;
          endcase


                  //Addition Carry
                  if (ALU_Sel==3'b000 && ALU_Result[8] == 1)
                        temp[8] = 1'h1;
                  else
                        temp[8] = 1'h0;

                  //subtraction borrow
                  if (ALU_Sel==3'b001 && A>B)
                  begin
               //b_c=~B+1;
                     //sub_r={1'b0,A}+{1'b0,b_c};
                     temp[8] = 1'h0;
                  end
     else if (ALU_Sel == 3'b001 && A < B)
               begin
                     b_c=~B+1;
                      sub_r={1'b0,A}+{1'b0,b_c};
     temp[8] = 1'h1;
     end

               end

          always @(posedge clk) begin
           count<=ALU_Sel;
    //count <= 0;
           count <= count + 1;
    end


endmodule
```

Test Bench Code

```
`timescale 1ns/1ps
module alu_tb();
    reg[7:0] A,B;
    reg[2:0] ALU_Sel;
          reg clk;

          wire[7:0] ALU_Out;
    wire CarryOut;

     alu test_unit(A,B,ALU_Sel,ALU_Out,CarryOut,clk);
     initial begin
         A=8'H80;
         B=8'H90;
         ALU_Sel=3'b000;
         clk=1'b1;

                    repeat (8) begin
    for (ALU_Sel = 3'b000; ALU_Sel <= 3'b111; ALU_Sel = ALU_Sel + 1) begin
       #10;
     end
   end

   #10;
   $finish;

     end
                always #5 clk = ~clk;
endmodule
```

UCF Code

```
NET "A[7]" LOC = P205;
NET "A[6]" LOC = P206;
NET "A[5]" LOC = P203;
NET "A[4]" LOC = P200;
NET "A[3]" LOC = P202;
NET "A[2]" LOC = P197;
NET "A[1]" LOC = P199;
NET "A[0]" LOC = P196;
NET "B[7]" LOC = P192;
NET "B[6]" LOC = P193;
NET "B[5]" LOC = P189;
NET "B[4]" LOC = P190;
NET "B[3]" LOC = P186;
NET "B[2]" LOC = P187;
NET "B[1]" LOC = P185;
NET "B[0]" LOC = P181;
NET "ALU_Out[7]" LOC = P179;
NET "ALU_Out[6]" LOC = P180;
NET "ALU_Out[5]" LOC = P177;
NET "ALU_Out[4]" LOC = P178;
NET "ALU_Out[3]" LOC = P152;
NET "ALU_Out[2]" LOC = P168;
NET "ALU_Out[1]" LOC = P171;
NET "ALU_Out[0]" LOC = P172;
NET "ALU_Sel[1]" LOC = P167;
NET "ALU_Sel[0]" LOC = P163;
NET "CarryOut" LOC = P153;
NET "clk" LOC = P132;
NET "ALU_Sel[2]" LOC = P165;
```

CHAPTER 4
IMPLEMENTTATION, TESTING AND DEBUGGING.

4.1 SIMULATION RESULTS.

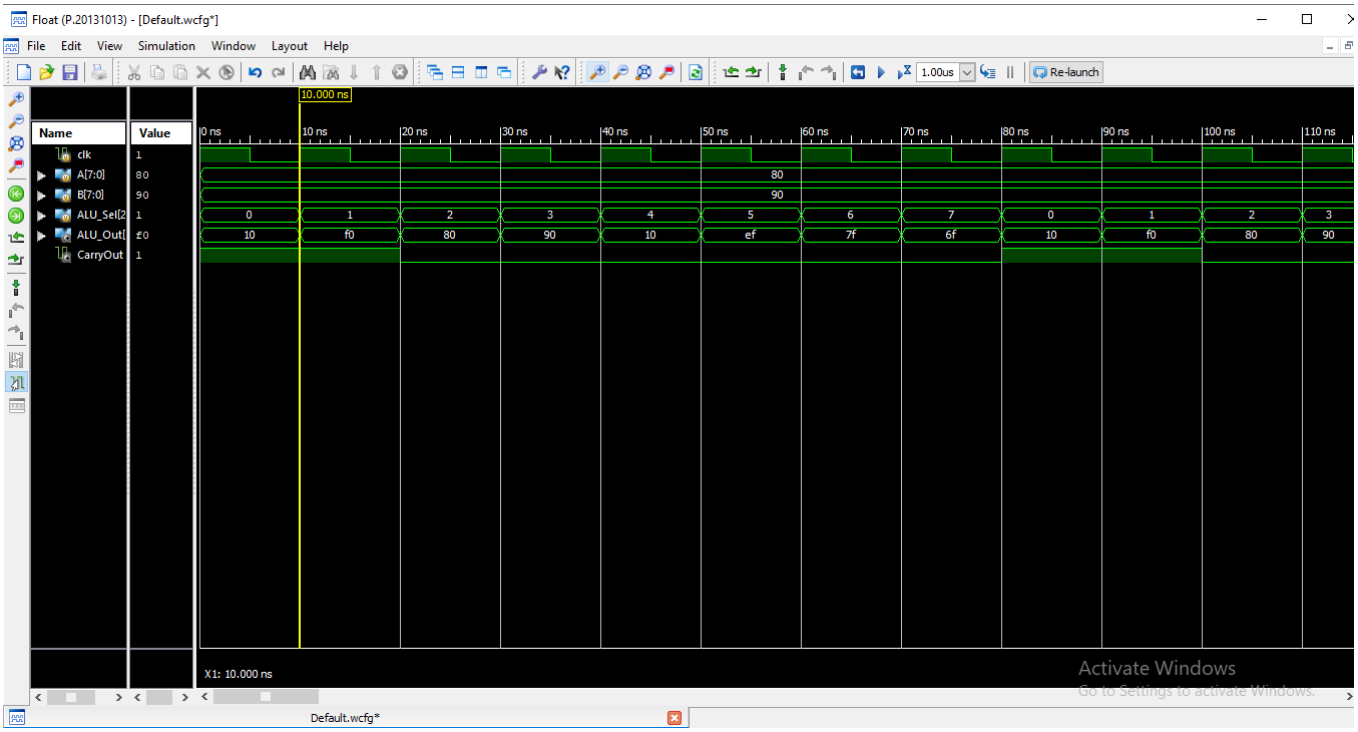

Figure 3 Simulation Result

- The above simulation results are obtained through the execution of testbench code.
- A and B are the two inputs given to the ALU to check the output of the system.
- A is given value 80H and B is given the value 90H. (H denotes hexadecimal)
- The selection of inputs is made in such a way that the output will generate CarryOut signal in both addition and subtraction operation.
- The ALU select line is incremented every positive edge of clock cycle in order to display all the 8 operations of ALU.
- The outputs can be cross checked with the observation table.
- The carry out which is being generated during addition operation, is denoted by CarryOut signal as observed in waveform.
- The output of subtraction operation is a negative output and it is denoted by the high vale of CarryOut signal.

4.2 OBSERVATION TABLE

OBSERVATION TABLE				
Operands: A= 80H B=90H				
Sr. No.	ALU_Sel	Operation	ALU_Out	CarryOut
1)	000	Addition	10	1
2)	001	Subtraction	F0	1
3)	010	AND	80	0
4)	011	OR	90	0
5)	100	XOR	10	0
6)	101	XNOR	EF	0
7)	110	NAND	7F	0
8)	111	NOR	6F	0

Table 2 Observation Table

- The Inputs are selected in order to make sure we can observe the output of every operation and ensure the proper working of logic implemented in the code.
- The addition and subtraction operations generate the CarryOut signal as expected.
- The output of subtraction is in 2’s complement form.
- All the outputs were cross checked by carrying out the calculations on paper.

4.3 FINAL PROJECT PHOTOGRAPH AND WORKING.

Figure 4 Testing on FPGA Kit

4.4 TESTING AND DEBUGGING

Testing

The testing was done on FPGA Board - Spartan-3E XC3S250E. We generated the UCF file, in which we allotted the FPGA pins to input and output. After generating the bit file, we connected the FPGA kit to the laptop through the downloading cable having one end of USB connector and other as a JTAG connector.
In order to dump the code, we just have to edit the windows FPGA batch file and run it. Once the FPGA was loaded with the code the led next to the IC will be started blinking indicating the status of the chip - programmed and we observed the output on the FPGA Board. The input is provided through the DIP switches and the clock can be varied through a trim pot available on the kit. The output is observed through the LEDs which are interfaced with the DIP switches.

Debugging

While testing the ALU module, we performed some troubleshooting operations whenever the output wasn't as expected or when encountered with errors. The main problem faced was during the implementation of the carry and borrow/ signed logic which was being denoted by CarryOut. The addition of two numbers wasn't producing the CarryOut signal as expected due to some logical errors in the equation being implemented.
While the implementation of borrow logic, the Output wasn't matching the calculations made on the paper. We solved the above errors by cross checking the logic of equations and ensured that the output is remaining correct by implementing various test cases with different values of inputs.
We also implemented the counter in the code, the results of code we observed were as expected in the simulation waveforms, but due to some warnings that are being generated in the code, the operation of counter on FPGA kit is not generated, i.e., the select lines aren't incrementing on every clock edge pulse. Therefore, we have to provide the input of ALU select lines manually through the DIP switches available on the FPGA kit.

CHAPTER 5
RESULTS AND DISCUSSION.

5.1 CONCLUSION

An 8-bit ALU project using Verilog is a complex digital design that involves designing a hardware circuit that can perform arithmetic and logical operations on two 8-bit operands.

Once the circuit is designed, it is simulated using a Verilog simulator to test its functionality and ensure that it behaves as expected. Simulation results are analysed to ensure that the circuit meets the design specifications and performs the required operations correctly. FPGA design offers greater design flexibility and additional features can be added to the existing design without any change in equipment.

In conclusion, an 8-bit ALU project using Verilog is an excellent opportunity to learn digital design principles, Verilog syntax, and simulation tools.

5.2 FUTURE SCOPE

The future scope of an 8-bit ALU project using Verilog includes expanding the functionality of the circuit to perform more complex operations and increasing the bit width to support larger operands. Some potential directions for future development of the project could include:

1. Increasing the bit width: Expanding the ALU to support larger operands, such as 16-bit or 32-bit, would allow it to perform more complex calculations.

2. Adding more operations: The ALU could be expanded to include additional arithmetic and logic operations, such as multiplication, division, and bitwise shifting.

3. Optimizing for performance: The design of the ALU could be optimized to improve its speed and reduce its power consumption.

4. Integration with a larger system: The ALU could be integrated into a larger digital system, such as a processor or a microcontroller, to perform arithmetic and logical operations as part of a larger computing system.

5. Testing and verification: The ALU project could be further developed to include comprehensive testing and verification procedures to ensure its correct functionality in a variety of scenarios.

5.3 REFERENCES

1. "Digital Design and Computer Architecture" by David Harris and Sarah Harris - This textbook provides an excellent introduction to digital design principles and Verilog programming.

2. "Verilog HDL: A Guide to Digital Design and Synthesis" by Samir Palnitkar - This book provides an in-depth tutorial on Verilog HDL and its use in digital design.

3. "Verilog by Example: A Concise Introduction for FPGA Design" by Blaine Readler - This book offers practical examples and case studies of Verilog programming for FPGA design.

4. "FPGA Prototyping by Verilog Examples: Xilinx Spartan-3 Version" by Pong P. Chu - This book provides examples of digital circuits designed using Verilog for FPGA prototyping.

5. "Digital System Design with FPGA: Implementation Using Verilog and VHDL" by Cem Unsalan, Bora Tar and A. Enis Cetin - This book offers comprehensive coverage of digital system design using both Verilog and VHDL.

6. Online resources like Verilog tutorial websites, Verilog reference manuals, and Verilog code repositories can also be helpful for designing and implementing the 8-bit ALU project.

7. Manit Kantawala, “Design and Implementation Of 8 Bit And 16 Bit Alu Using Verilog Language.” June 2018 – IJEAST-VOL3, Jaipur, Rajasthan, India.

5.4 DATASHEET

Chapter 1: Introduction

Thank you for purchasing the Xilinx Spartan™-3E based VLSI Development Board (**SLT-UNI-VLSI-V1**). You will find it useful in developing your Spartan-3E FPGA application. The board provides easy to use development platform for implementing digital designs.

Key Components and Features:

The key features of the VLSI Development board are:

- Spartan-3E XC3S250E FPGA
  - Up to 172 user-I/O pins
  - 208-pin FBGA package
  - Over 5,000 logic cells
- 2-line, 16-character LCD screen
- Two 9-pin RS-232 ports (DTE- and DCE-style)
- 32 discrete LEDs and DIP switches
- PS/2 mouse or keyboard port
- Separate VGA display port
- Expansion connectors (32 free user I/O arranged in 10 pin FRC)
- Dual, 8 bit DAC
- 8 bit, 8 channel ADC
- USB interface(type B)
- 8 pushbuttons for trigger, input
- 6 common anode 7-segment LED display
- Xilinx 2 Mbit Platform Flash configuration PROM

Chapter 4: LED's and DIP Switches Interface

The **SLT-UNI-VLSI-V1** board has 32 individual bidirectional I/O's. Each I/O is connected with a surface-mount LED and a DIP switch. A LED is assigned to each I/O to indicate its data status when I/O is configured as input. DIP switch is used to provide digital input (i.e. logic 0 and logic 1) to the FPGA.

The LED can display the output data value of I/O by configuring it as output and keeping its corresponding DIP switch at 0 positions.

The LEDs are divided in four groups of 8 LED based on 8-pin DIP switches I/O's are labeled as TL1 to TL32. (SW1:TL1-TL8, SW2:TL9-TL16, SW3:TL17-TL-24, SW4:TL25-TL32).

Pin Assignment (UCF Location) for IOs:

DIP Switch	Signal Name	XC3S250E-PQ208	XCS6LX9-TQG144	DIP Switch	Signal Name	XC3S250E-PQ208	XCS6LX9-TQG144
SW1	TL1	P205	P29	SW3	TL17	P179	NC
	TL2	P206	P30		TL18	P180	NC
	TL3	P203	P26		TL19	P177	NC
	TL4	P200	P23		TL20	P178	NC
	TL5	P202	P24		TL21	P152	NC
	TL6	P197	P21		TL22	P168	NC
	TL7	P199	P22		TL23	P171	NC
	TL8	P196	P17		TL24	P172	NC
SW2	TL9	P192	P11	SW4	TL25	P165	NC
	TL10	P193	P12		TL26	P167	NC
	TL11	P189	P9		TL27	P163	NC
	TL12	P190	P10		TL28	P164	NC
	TL13	P186	P144		TL29	P161	NC
	TL14	P187	P2		TL30	P162	NC
	TL15	P185	P5		TL31	P160	NC
	TL16	P181	P6		TL32	P153	NC

Table 4.1: Pin Assignment (UCF) for LED's and DIP switches

Chapter 12: Clock Sources

The **SLT-UNI-VLSI-V1** board supports multiple clock input sources which are listed below.

- The board includes an on-board 12 MHz clock oscillator
- Three Clock outputs form USB to UART IC FT232HL.
  - 7.5 MHz, 15 MHz, 30MHz
- 32 KHz Clock output from RTC IC3231.
- Variable Clock using IC555. The Clock is varied using potentiometer PR1.

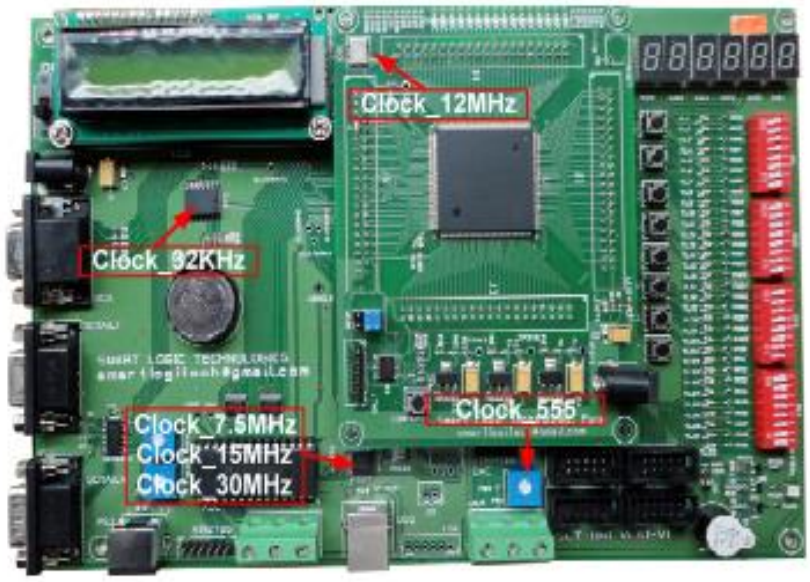

Figure 12.1: Clock sources for the FPGA

Pin Assignment (UCF Location) for Clock Sources:

Signal Name	XC3S250E-PQ208	XC6SLX9-TQG144
RESET	--	P32
Clock_12MHz	P80	P55
Clock_7.5MHz	P78	NC
Clock_15MHz	P77	NC
Clock_30MHz	P83	NC
Clock_32kHz	P75	NC
Clock_555	P132	P120

Table 12.1: Pin Assignment (UCF) for Clock sources