Lab 6 – RISC V

Name: Breeha Qasim	**ID:** 08283

Task 1

Code

Codes should contain meaningful variable naming and comments

**Add snip of code or copy-paste the code written in the Editor window. Make sure the irrelevant area of snip is cropped.*

Design Modules

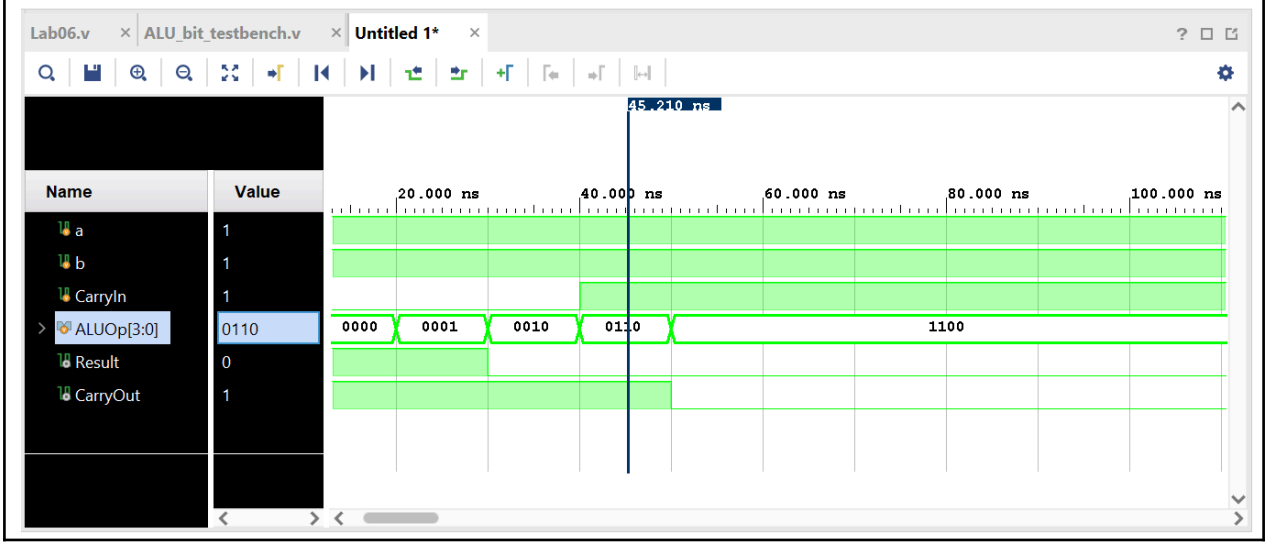
```
module ALU_1_bit(
    input a,
    input b,
    input CarryIn,
    input [3:0] ALUOp,
    output Result,
    output CarryOut
    );
    wire mux1out;
    wire mux2out;
    assign mux1out = ALUOp[3] ? ~a : a;
    assign mux2out = ALUOp[2] ? ~b : b;
    //calculating result based on [1:0] ALUOp bits
    assign Result = ALUOp[0] == 1 ?
(mux1out|mux2out) : ALUOp[1] == 0 ? (mux1out
& mux2out) : (mux1out + mux2out + CarryIn);
    //carryout calculation CarryOut= (a . CarryIn) +
(b . CarryIn) + (a . b)
    assign CarryOut = (mux1out & CarryIn) |
(mux2out & CarryIn) |(mux1out & mux2out);

endmodule
```

Testbench

```
module ALU_bit_testbench();
    reg a;
    reg b;
    reg CarryIn;
    reg [3:0] ALUOp;
    wire Result;
    wire CarryOut;

    ALU_1_bit ALU(a, b, CarryIn, ALUOp, Result,
CarryOut);
    initial
    begin
        a = 1'b1;
        b = 1'b1;
        CarryIn = 1'b0;

        #10
        ALUOp = 4'b0000; //test for AND

        #10
        ALUOp = 4'b0001; //test for OR

        #10
        ALUOp = 4'b0010; //test for ADDITION

        #10
        CarryIn = 1'b1;
                    ALUOp = 4'b0110; //test for
SUBTRACTION

        #10
        ALUOp = 4'b1100; //test for NOR
    end

endmodule
```

Results (Waveforms)

**Add snip of relevant signals' waveforms. Results in Log Window are optional. Make sure the irrelevant area of snip is cropped.*



Comments

**Comment on the obtained results/working of code*

It performs different operations on two 1-bit inputs (a and b) based on a 4-bit operation code (ALUOp). The operations include logical operations (AND, OR, NOT) and arithmetic operations (Addition, Subtraction), with the specific operation determined by the ALUOp code. The module outputs a 1-bit result (Result) of the operation and a carry-out bit (CarryOut) which is useful for multi-bit arithmetic operations involving multiple ALU instances. The module uses internal wires (mux1out, mux2out) to optionally invert inputs a and b based on ALUOp, and combines these processed inputs to produce the final result and carry-out according to the operation specified by ALUOp.

Task 2

Code

Codes should contain meaningful variable naming and comments

**Add snip of code or copy-paste the code written in the Editor window. Make sure the irrelevant area of snip is cropped.*

Design Modules

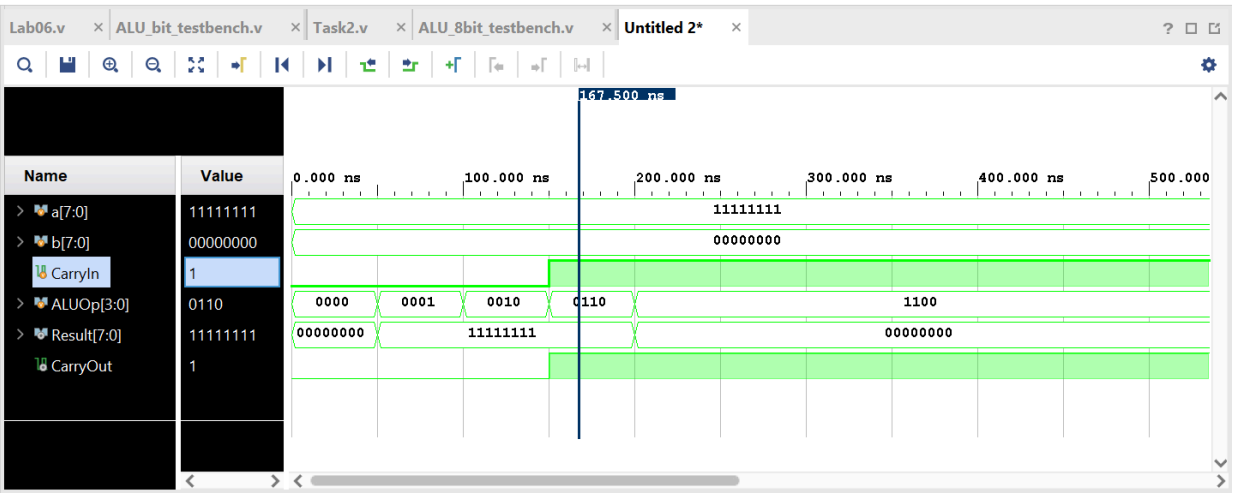
```
module ALU_8_bit(
    input wire [7:0] a,
    input wire [7:0] b,
    input wire CarryIn,
    input [3:0] ALUOp,
    output wire [7:0] Result,
    output wire CarryOut
    );
    wire [6:0] Carry;
    //instantiating ALU 1 bit module 8 times with
different carry outs
    ALU_1_bit ALU1(a[0], b[0], CarryIn, ALUOp,
Result[0], Carry[0]);
    ALU_1_bit ALU2(a[1], b[1], Carry[0],
ALUOp, Result[1], Carry[1]);
    ALU_1_bit ALU3(a[2], b[2], Carry[1],
ALUOp, Result[2], Carry[2]);
    ALU_1_bit ALU4(a[3], b[3], Carry[2],
ALUOp, Result[3], Carry[3]);
    ALU_1_bit ALU5(a[4], b[4], Carry[3],
ALUOp, Result[4], Carry[4]);
    ALU_1_bit ALU6(a[5], b[5], Carry[4],
ALUOp, Result[5], Carry[5]);
    ALU_1_bit ALU7(a[6], b[6], Carry[5],
ALUOp, Result[6], Carry[6]);
    ALU_1_bit ALU8(a[7], b[7], Carry[6],
ALUOp, Result[7], CarryOut);

endmodule
```

Testbench

```
module ALU_8bit_testbench;
    reg [7:0] a;
    reg [7:0] b;
    reg CarryIn;
    reg [3:0] ALUOp;
    wire [7:0] Result;
    wire CarryOut;

        ALU_8_bit(a, b, CarryIn, ALUOp, Result,
CarryOut);

    initial
    begin
        a=8'b11111111;
        b=8'b00000000;
        CarryIn=1'b0;

        //testing AND
        ALUOp=4'b0000;

        #10
        // testing OR
        ALUOp=4'b0001;

        #10
        // testing Addition
        ALUOp=4'b0010;

        #10
        // testing Subtraction
        CarryIn = 1'b1;
        ALUOp = 4'b0110;

        #10
        // testing NOR
        ALUOp=4'b1100;

        end
endmodule
```

Results (Waveforms)

**Add snip of relevant signals' waveforms. Results in Log Window are optional. Make sure the irrelevant area of snip is cropped.*



Comments

**Comment on the obtained results/working of code*

This module is instantiating ALU_1_bit 8 times. Each 1-bit ALU handles a single bit of the 8-bit inputs a and b, and they are connected in a way that allows for carry bits to propagate between them, from the least significant bit to the most significant bit. The ALU supports five operations (Addition, Subtraction, AND, OR and NOR) , determined by the 4-bit ALUOp input, and provides an 8-bit result along with a carry out signal indicating overflow from the most significant bit.

Task 3

Code

Codes should contain meaningful variable naming and comments

**Add snip of code or copy-paste the code written in the Editor window. Make sure the irrelevant area of snip is cropped.*

Design Modules

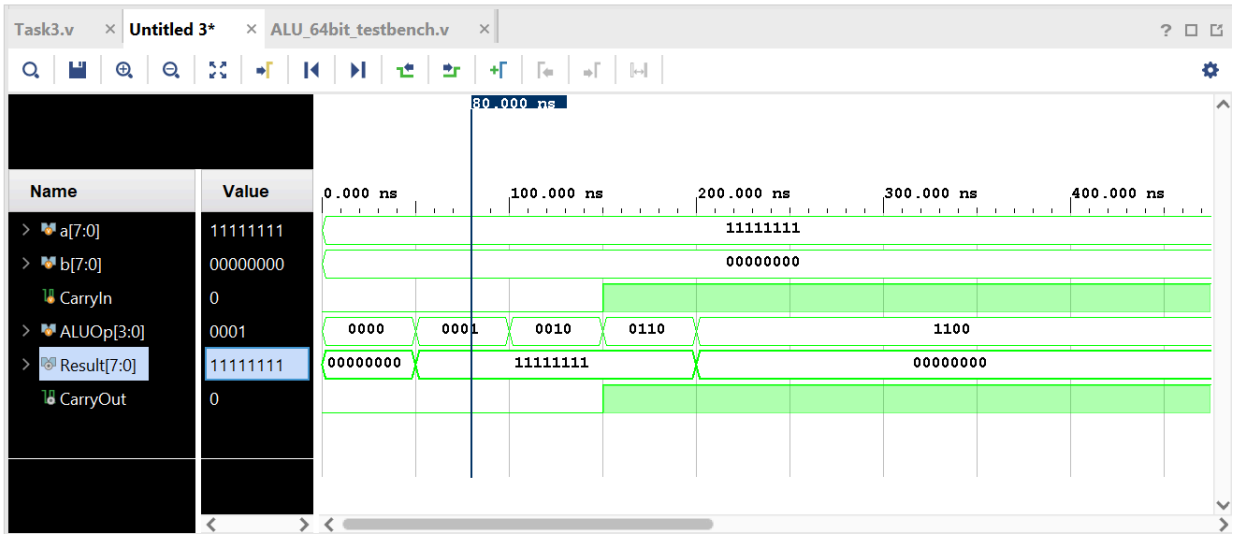
```
module ALU_64_bit(
    input [63:0] a,
    input [63:0] b,
    input [3:0] ALUOp,
    output reg [63:0] Result,
    output ZERO
    );
    always @(*)
    begin

     case (ALUOp)

     4'b0000:
     Result = (a&b);     // when AND

     4'b0001:
     Result = (a|b);     // when OR

     4'b0010:
     Result = (a+b);     // when Addition

     4'b0110:
     Result = (a-b);     // when Subtraction

     4'b1100:
     Result = ~(a|b);     // when NOR

     default:
     Result = 64'h0000;

     endcase
     end
     assign ZERO = Result? 0:1;
endmodule
```

Testbench

```
`timescale 1ns / 1ps
module test_64bit_ALU();
    reg [63:0] a;
    reg [63:0] b;
    reg [3:0] ALUOp;
    wire [63:0] Result;
    wire ZERO;

    ALU_64_bit a64(a, b, ALUOp, Result, ZERO);

    initial
    begin

       a=64'h1111;
       b=64'h0000;

       ALUOp=4'b0000;         //testing AND

       #50
       ALUOp=4'b0001;         // testing OR

       #50
       ALUOp=4'b0010;         // testing Addition

       #50
       ALUOp=4'b1100;         // testing NOR

    end

endmodule
```

Results (Waveforms)

**Add snip of relevant signals' waveforms. Results in Log Window are optional. Make sure the irrelevant area of snip is cropped.*



Comments

**Comment on the obtained results/working of code*

It performs various arithmetic and logic operations based on the input signals. The module has two 64-bit input operands a and b, a 4-bit input ALUOp to select the operation, a 64-bit output Result that holds the outcome of the operation, and a ZERO flag that indicates if the result is zero.

Operations supported by this ALU include:

- AND (ALUOp = 4'b0000): Bitwise AND of a and b which in my code will give output 00000000 when a and b are AND
- OR (ALUOp = 4'b0001): Bitwise OR of a and b which in my code will give 11111111
- Addition (ALUOp = 4'b0010): Adds a and b which in my code will give 11111111
- Subtraction (ALUOp = 4'b0110): Subtracts b from a which in my code will give 11111111
- NOR (ALUOp = 4'b1100): Bitwise NOR of a and b which in my code will give 00000000
- Default: If none of the specified ALUOp codes are matched, Result is set to 0.

Lab 06 – RISC ALU

Assessment Rubric

Name:	**Student ID:**

Points Distribution

Task No.	**LR 2** **Code**	**LR 5** **Results**	**AR 7** **Report Submission**
Task 1	/20	/10	/20
Task 2	/15	/10	
Task 3	/15	/10	
Total Points	/100 Points		
CLO Mapped	CLO 1		

For description of different levels of the mapped rubrics, please refer the provided Lab Evaluation Assessment Rubrics and Affective Domain Assessment Rubrics.

#	Assessment Elements	Level 1: Unsatisfactory	Level 2: Developing	Level 3: Good	Level 4: Exemplary
LR2	Program/Code/ Simulation Model/ Network Model	Program/code/simulation model/network model does not implement the required functionality and has several errors. The student is not able to utilize even the basic tools of the software.	Program/code/simulation model/network model has some errors and does not produce completely accurate results. Student has limited command on the basic tools of the software.	Program/code/simulation model/network model gives correct output but not efficiently implemented or implemented by computationally complex routine.	Program/code/simulation /network model is efficiently implemented and gives correct output. Student has full command on the basic tools of the software.
LR5	Results & Plots	Figures/ graphs / tables are not developed or are poorly constructed with erroneous results. Titles, captions, units are not mentioned. Data is presented in an obscure manner.	Figures, graphs and tables are drawn but contain errors. Titles, captions, units are not accurate. Data presentation is not too clear.	All figures, graphs, tables are correctly drawn but contain minor errors or some of the details are missing.	Figures / graphs / tables are correctly drawn and appropriate titles/captions and proper units are mentioned. Data presentation is systematic.

AR9	Report Content/Code Comments	No summary provided. The number/amount of tasks completed below the level of satisfaction and/or submitted late	Couldn't provide good summary of in-lab tasks. Some major tasks were completed but not explained well. Submission on time. Some major plots and figures provided	Good summary of In-lab tasks. All major tasks completed except few minor ones. The work is supported by some decent explanations, Submission on time, All necessary plots, and figures provided	Outstanding Summary of In-Lab tasks. All task completed and explained well, submitted on time, good presentation of plots and figure with proper label, titles and captions