EECE 2323 Digital Logic Design Lab Report

Lab 3 ALU

Student Name:Ousmane Toure

Section #: 1

Instructor Name: Dr. Aboelela

Date: 07/19/22

Lab TA Name: Keshav Bharadwaj Vaidyanathan, Srinidi Somasundaram Subbulakshmi, Yuchao Su

1. Background & Purpose:

In this experiment, we completed the partial Arithmetic and Logic Unit (ALU) from Lab 3. Our ALU is responsible for doing addition, bitwise NOT, AND, and OR, Arithmetic Shift left and logical shift left, and branch if equal or branch if not equal. The results were then displayed utilizing the VIO Dashboard on Vivado and the LEDs on the PYNQ. ALU's are essential because they give central processing units the ability to perform mathematical operations. This is important because it allows the CPU to do functions like addition, division, subtraction etc. The goal of this lab was to utilize verilog code in vivado to complete our partial ALU and virtually test it before physically implementing it. Furthermore enhancing confidence and proficiency in vivado and computational architecture.

2. Pre-Lab Response:

Table 1: 2's Complement Truth Table

a	b	**s**	f	take_branch	ovf
8'd0 00000000	8'd0 00000000	**3'd0** **000**	8'd0 00000000	0	0
-8'd12 11110100	-8'd34 11011110	**3'd0** **000**	-8'd46 11010010	0	0
8'd100 01100100	8'd100 01100100	**3'd0** **000**	-8'd56 11001000	0	1
8'd1 00000001	8'd1 00000001	**3'd1** **001**	8'd4 00000100	0	0
-8'd12 11110100	-8'd12 11110100	**3'd1** **001**	8'd11 00001011	0	0
8'd100 01100100	8'd100 01100100	**3'd1** **001**	-8'd101 10011011	0	0
8'd1 00000001	8'd1 00000001	**3'd2** **010**	8'd1 00000001	0	0
8'd1	8'd3	**3'd2**	8'd1	0	0

00000001	00000011	**010**	00000001		
-8’d4 11111100	-8’d5 11111011	**3'd2 010**	-8’d8 11111000	0	0
8’d1 00000001	8’d1 00000001	**3'd3 011**	8’d1 00000001	0	0
8’d12 00001100	-8’d16 11110000	**3’d3 011**	-8’d4 11111100	0	0
-8’d2 11111110	-8’d6 11111010	**3'd3 011**	-8’d2 11111110	0	0
8’d3 00000011	8’d0 00000000	**3’d4 100**	8’d1 00000001	0	0
8’d100 01100100	8’d0 00000000	**3’d4 100**	8’d50 00110010	0	0
8’d5 00000101	8’d0 00000000	**3’d4 100**	8’d5 00000010	0	0
8’d3 00000011	8’d1 00000001	**3’d5 101**	8’d6 00000110	0	0
8’d5 00000101	8’d2 00000010	**3’d5 101**	8’d20 00010100	0	0
-8’d11 11110101	8’d3 00000011	**3’d5 101**	-8’d88 10101000	0	0
8’d5 00000101	8’d5 00000101	**3’d6 110**	0	1	0
8’d5 00000101	8’d2 00000010	**3’d6 110**	0	0	0
-8’d11 11110101	-8’d11 11110101	**3’d6 110**	0	1	0
-8’d11 11110101	8’d2 00000010	**3’d7 111**	0	1	0
8’d5 00000101	8’d5 00000101	**3’d7 111**	0	0	0
-8’d12 11110100	-8’d34 11011110	**3’d7 111**	0	1	0
					0

3. Summary of Design Implementation

3.1.Results and Analysis:

When conducting this experiment, the partial ALU from Lab 2 was copied over and updated to add our new elements. We first added more select ports and created new test values s and ran a simulation to verify that our operations were in fact working. The table is displayed in the prelab. Our results, as stated in Appendix B Figure 1. The results of the test bench verified that our code in fact does work and gives us the green light to program straight into the PYNQ-Z2 board. After programming our board and connecting the add on board, we opened the virtual input output (VIO) dashboard which allowed us to test our values as our PYNQ board did not have enough physical inputs. Shift Left, Branch if Equal and AND operation were tested using the VIO dashboard and the resulting screenshots were placed in Appendix B. All tests resulted in values that were consistent with our pre lab test bench truth table highlighting that our circuit was in fact correct. You could face many errors when conducting this lab. For example, when creating your virtual input output, if you didn't correctly instantiate your values in your top file, Vivado will leave you with constants with no values.

3.2.Conclusion & Recommendations:

Based on our results, we can conclude that Lab 2 consisted of completing our Partial Arithmetic Logical Unit from Lab 2 utilizing verilog code in Vivado. Testing its implementation virtually with a test bench confirmed that our verilog code was correct which gave us the green light to program the PYNQ Board using Virtual Input Output (VIO) ports. . The lab resulted in successfully being able to perform different arithmetic and logical functions like addition and bit shifting as well as checking if values were equal. These tools are crucial for a Central Processing Unit, and gives it the ability to perform varying tasks of different degrees. Recommendations going forward would be to make the partial ALU the prelab to completing the full ALU. Overall, the lab can be seen as a success.

Appendix A: Design Program Files (Verilog modules, testbenches, etc)

Verilog Code:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 07/14/2022 10:11:34 AM
// Design Name:
// Module Name: eightbit_alu
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module eightbit_alu(
```

```
input signed [7:0] a,
input signed[7:0] b,
input [2:0] sel,
output reg [7:0] f,
output reg ovf,
output reg take_branch
);

always@(a or b or sel)
begin
   case(sel)
      3'b000: //add
      begin
         f = a + b;
         ovf  =((a[7] & b[7]) &!f[7]) | ((!a[7] & !b[7])&f[7]);
         take_branch = 0;
      end

      3'b001: //b inversion
      begin
         f= !b;
         ovf = 0;
         take_branch = 0;
      end

      3'b010: //AND
      begin
         f = a * b;
         ovf = 0;
         take_branch = 0;
      end

      3'b011: //OR
      begin
         f = a | b;
         ovf = 0;
         take_branch = 0;
      end

      3'b100: //Shift right
      begin
         f = a >>> 1;
         ovf = 0;
         take_branch = 0;
      end

      3'b101: //shift left
      begin
         f = a << b;
         ovf = 0;
         take_branch = 0;
      end

      3'b110: //branch if equal
```

```
        begin
            f = 0;
            ovf = 0;
            take_branch = (a == b);
        end

        3'b111: //Branch if not equal
        begin
            f = 0;
            ovf = 0;
            take_branch = (a != b);
        end

        default:
         begin
            f = 0;
            take_branch=0;
            ovf  =0;
        end

      endcase
   end


endmodule
```

TestBench Code:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 07/14/2022 10:22:52 AM
// Design Name:
// Module Name: alu_unit_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module alu_unit_tb;
```

```
reg [7:0] a =8'b0;
reg [7:0] b = 8'b0;
reg [2:0] sel = 3'd0;
wire [7:0] f;
wire ovf;
wire take_branch;

    eightbit_alu UUT(
    .a(a),
    .b(b),
    .sel(sel),
    .f(f),
    .ovf(ovf),
    .take_branch(take_branch)
    );

    initial begin
    sel=3'd0; a=8'd0;b=8'd0; //add
    #100;
    a=-8'd12; b=-8'd34;
    #100;
    a=8'd100; b=8'd100;

    #100; //inver
    sel=3'd1; a=8'd1; b=8'd1;
    #100;
    a=-8'd12; b=-8'd12;
    #100;
    a=8'd100; b=8'd100;

    #100;//and
    sel=3'd2; a=8'd1; b=8'd1;
    #100;
    a=8'd1; b=8'd3;
    #100;
    a=-8'd4; b=-8'd5;

    #100;//or
    sel=3'd3; a=8'd1; b=8'd1;
    #100;
    a=8'd12; b=-8'd16;
    #100;
    a=-8'd2; b=-8'd6;

    #100;//shift right
    sel=3'd4; a=8'd3;
    #100;
    a=8'd100;
    #100;
    a=8'd5;

    #100;//shift left
    sel=3'd5; a=8'd3; b=8'd1;
```

```
    #100;//shift left
    a=8'd5; b=8'd2;
    #100;//shift left
    a= -8'd11; b=8'd3;

    #100;//branch if equal
    sel=3'd6; a=8'd5; b=8'd5;
    #100;//branch if equal
    a=8'd5; b=8'd2;
    #100;//branch if equal
    a= -8'd11; b=-8'd11;

    #100;//branch if  NOT equal
    sel=3'd7; a=-8'd11; b=8'd2;
    #100;//branch if  NOT equal
    a=8'd5; b=8'd5;
    #100;//branch if  NOT equal
    a=-8'd12; b=-8'd34;
    #100;//branch if  NOT equal
    a=8'd0; b=8'd0;




    end
endmodule
```

Appendix B: Captures of the output screens and simulation waveforms.

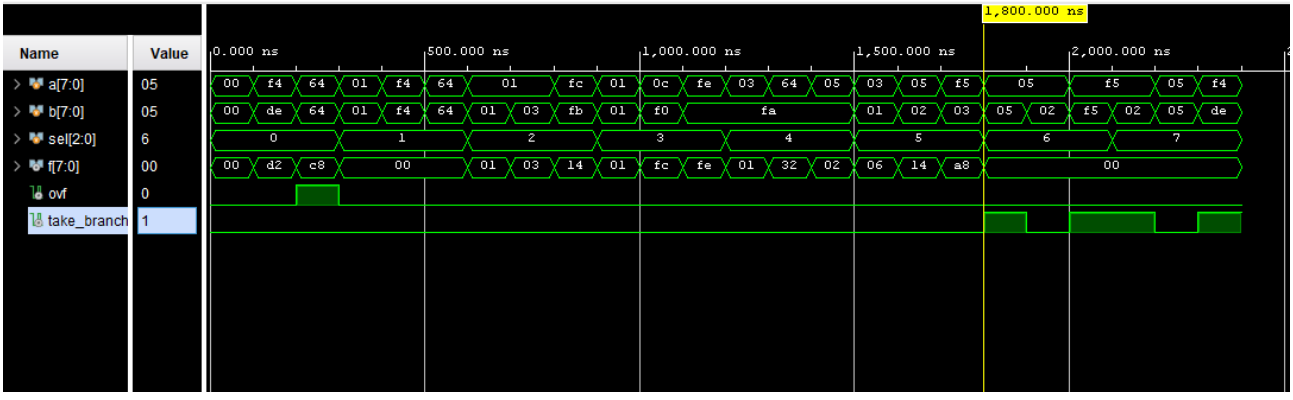

Figure 1: Test Bench WaveForm Simulation

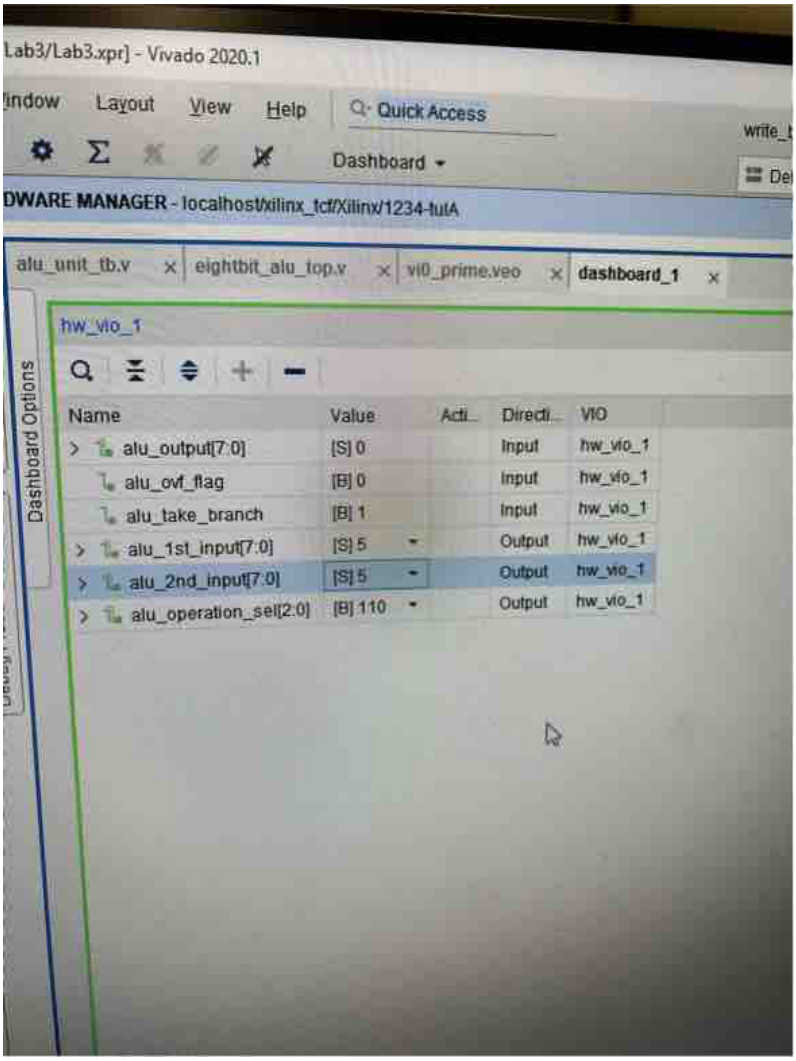

Figure 2: VIO Dashboard Select 6

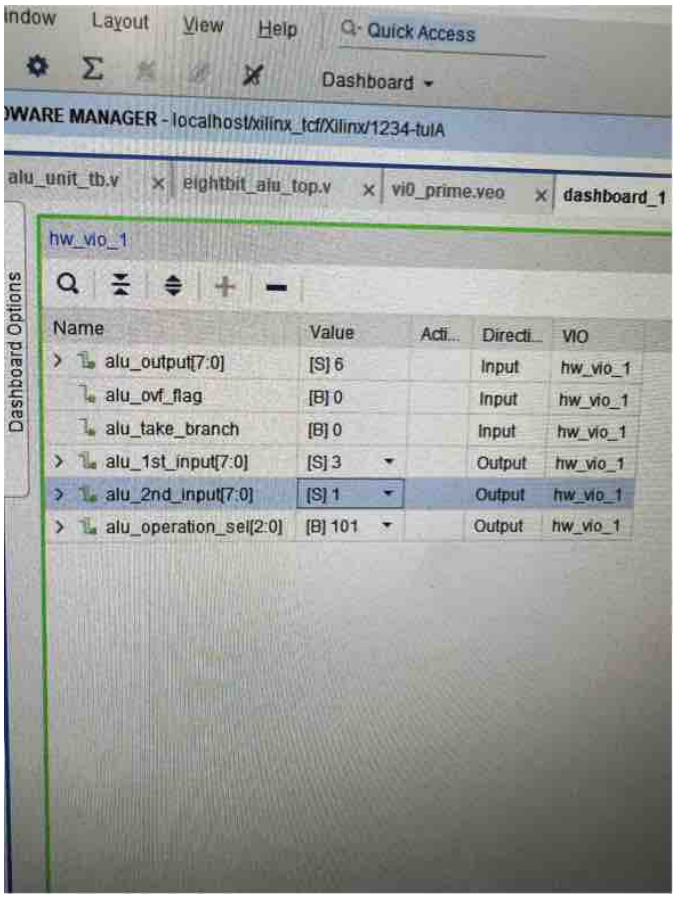

Figure 2: VIO Dashboard Select 5

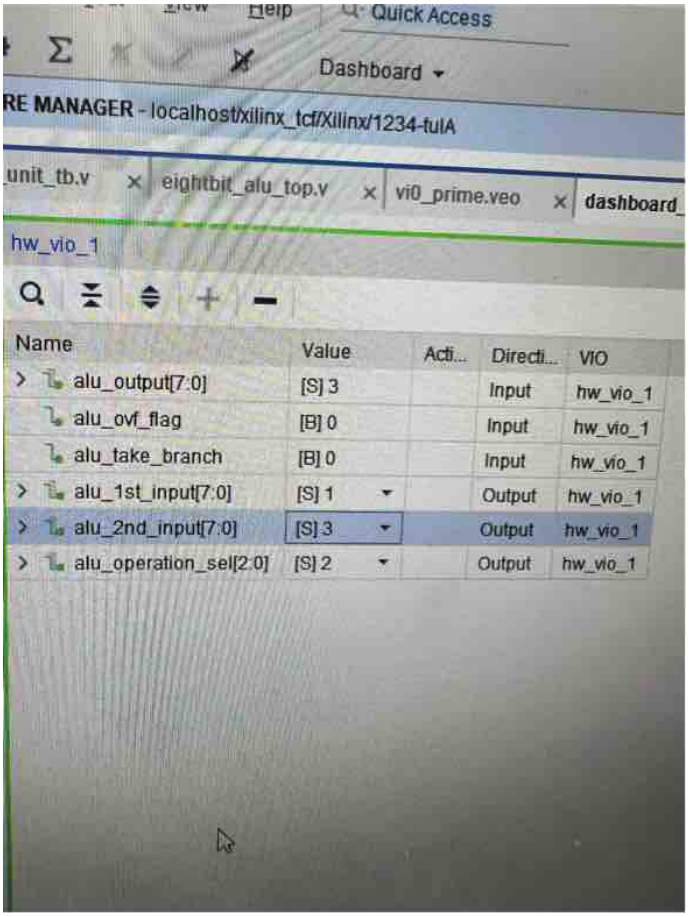

Figure 2: VIO Dashboard Select 2