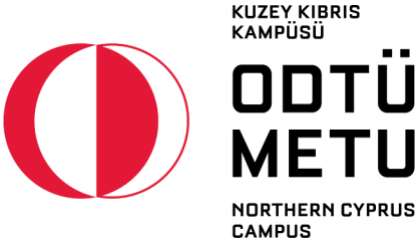

Lab 3 Report

Name: Mert Can Bilgin

Student ID: 2453025

EXPERIMENTAL RESULTS

2.2.7 LOGIC UNIT WITH MULTIPLEXERS

QUESTION 2.2.7.1

Using the basic gates in LAB 1, design the logic unit by combining with 4-to-1 and 2-to-1 Multiplexers as to fulfil the correct operation as described in Table 4.

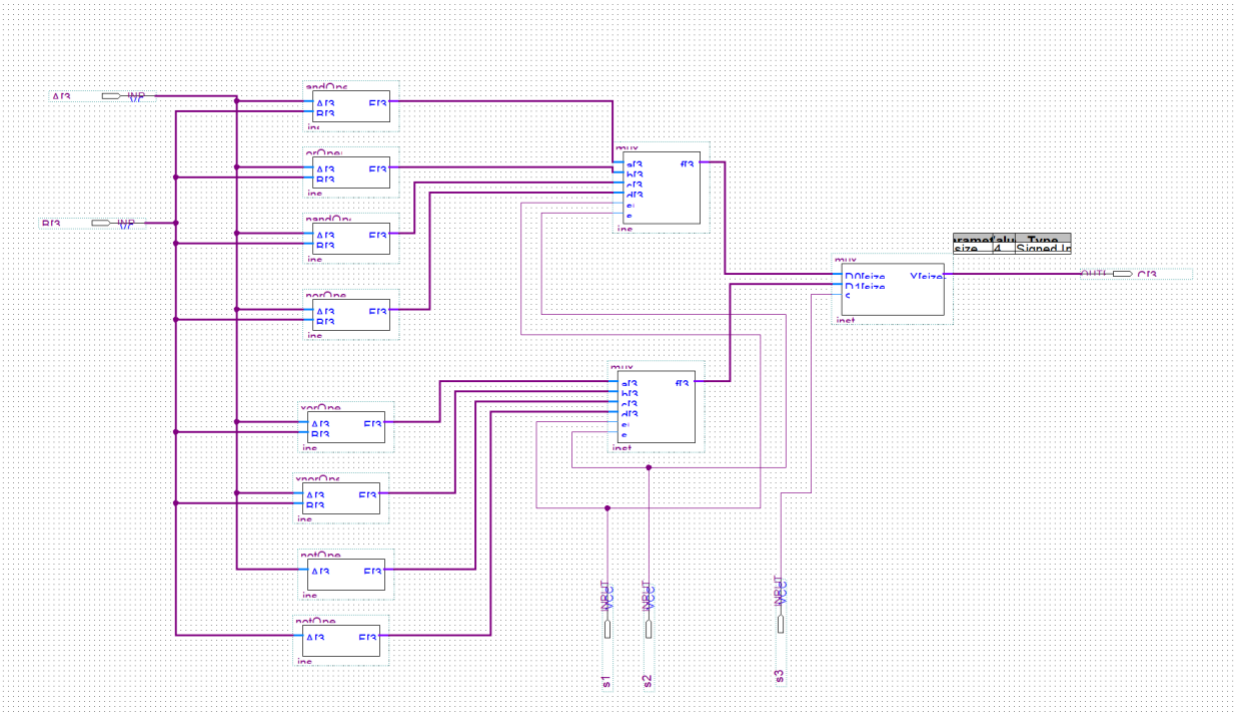

Figure 1: Schematic of Logic Unit

QUESTION 2.2.7.2

Write a parametrized and structural verilog HDL code named as `LOGIC_UNIT` using the Verilog code that have been implemented in LAB 1 for basic gates and implement the logic unit designed in 2.2.7.1.

C:/Users/mertc/Desktop/LOGIC_UNIT/Modelsim/LOGIC_UNIT.v (/LOGIC_UNIT_tb/DUT) - Default

Ln#

```
1   module LOGIC_UNIT(A,B,s1,s2,s3,C);
2
3   parameter size = 4;
4
5   input [size - 1:0] A,B;
6   input s1,s2,s3;
7   output [size - 1:0] C;
8   wire [size - 1:0] And, Or, Xor, Nand, Nor, Xnor, Not, Not2, Mux1, Mux2;
9
10  andOperation U1(And,A,B);
11  orOperation U2(Or,A,B);
12  nandOperation U3(Nand,A,B);
13  norOperation U4(Nor,A,B);
14  xorOperation U5(Xor,A,B);
15  xnorOperation U6(Xnor,A,B);
16  notOperation U7(Not,A);
17  notOperation U8(Not2,B);
18
19
20  mux_41 U9(Mux1, And, Or, Nand, Nor, s1,s2);
21  mux_41 U10(Mux2,Xor,Xnor,Not,Not2,s1,s2);
22  mux21 U11(C,Mux1,Mux2,s3);
23
24  endmodule
25
```

Wave | LOGIC_UNIT.v | LOGIC_UNIT_tb.v

Figure 2: Verilog Code of Logic Unit

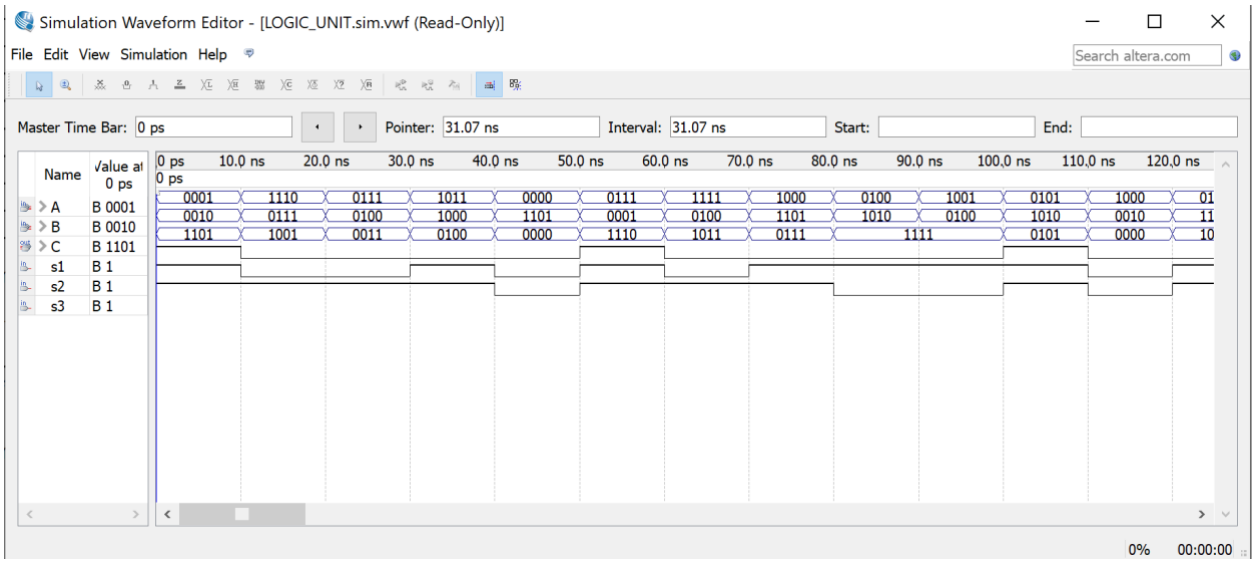

Figure 3: Waveform of Logic Unit

QUESTION 2.2.7.3

Write a testbench for your design using Modelsim® and simulate your Logic Unit.

```
1  module LOGIC_UNIT_tb();
2
3    reg [3:0] A, B;
4    reg s1,s2,s3;
5    wire [3:0] C;
6
7  LOGIC_UNIT DUT(A,B,s1,s2,s3,C);
8
9  always
10 begin
11   A = 4'b0100; B = 4'b0110; s1 = 1'b0; s2 = 1'b0; s3 = 1'b0; #100;
12   A = 4'b1000; B = 4'b0010; s1 = 1'b0; s2 = 1'b0; s3 = 1'b1; #100;
13   A = 4'b0011; B = 4'b0011; s1 = 1'b0; s2 = 1'b1; s3 = 1'b0; #100;
14   A = 4'b0101; B = 4'b1001; s1 = 1'b1; s2 = 1'b0; s3 = 1'b0; #100;
15   A = 4'b1111; B = 4'b0100; s1 = 1'b1; s2 = 1'b0; s3 = 1'b1; #100;
16   A = 4'b0001; B = 4'b0001; s1 = 1'b1; s2 = 1'b1; s3 = 1'b0; #100;
17   A = 4'b0100; B = 4'b0010; s1 = 1'b1; s2 = 1'b1; s3 = 1'b1; #100;
18 end
19 endmodule
20
```

Figure 4: TestBench of Logic Unit

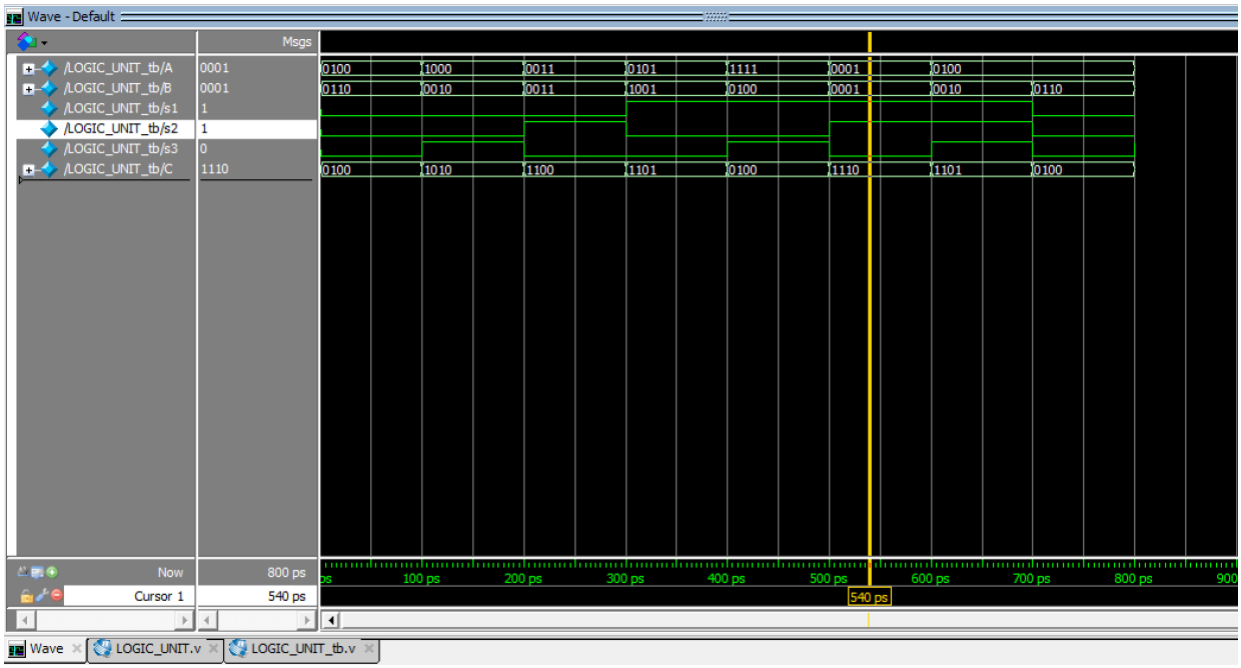

Figure 5: Simulation of Logic Unit

Comments:

- I have learned that Logic Unit gives a chance to computers perform basic operations in this experiment. The decision is dependent on selections. The selections come from multiplexers. The selections tells to logic unit what operation to perform.
- Table 5 shows us which operations shoul be performed in the manuel. For example, when selections are 000, AND operation should be performed.
- Firstly, I have modified my verilog code for basic gates which I coded in Lab 1. I implement the new code for 4-bit operations. Then, I added two 4-1 multiplexer, and I connected them to a one 2-to-1 multiplexer.
- After completing the verilog codes for basic gates, I created symbol for each them in order to use them in schematic. Then, I completed my design.
- I have used parameterized structural method in my code. After the basic gate operations, I called the multiplexers in my code.
- After that, I created a waveform to be ensure that my design is correct according to timing and functional simulation.
- After getting correct results from waveform, I wrote testbench and completed simulation.
- I am sure that my results are correct by comparing with truth tables. For instance, between 500ps and 600ps, the selection is 110, so NOR operation should be performed. A is 0001 and B is 0001. Therefore, we can easily see that (A NOR B) is 1110. Hence, it is correct.
- Also, I have implemented my pin assignments to be able to see the least significant bit at rightmost. In addition, the first four switch represents the A, the other four represents the B. I have chosen push buttons for selections.

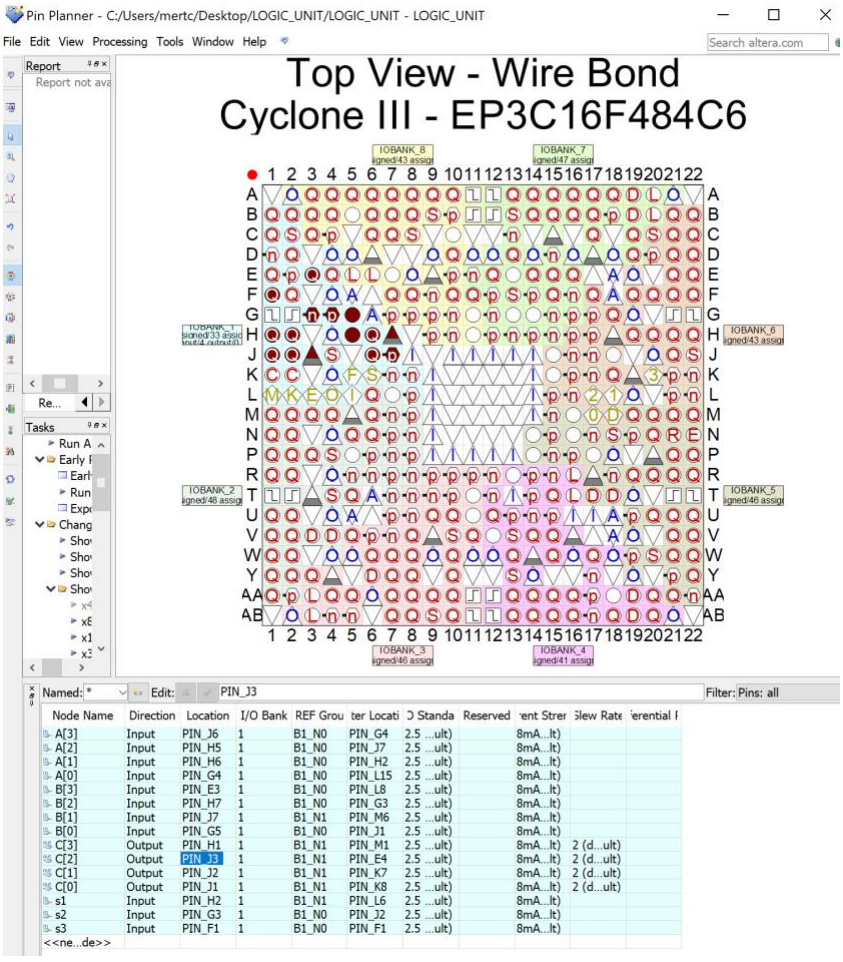

Figure 6: Pin Assignment of Logic Unit

2.2.8 ARITHMETIC LOGIC UNIT (ALU) TOP LEVEL DESIGN

QUESTION 2.2.8.1

After all components required for ALU is designed, implemented and tested. Create a top-level structural (hierarchical) Verilog HDL file called **ALU** and instantiate and connect all the components as depicted in Figure 2.

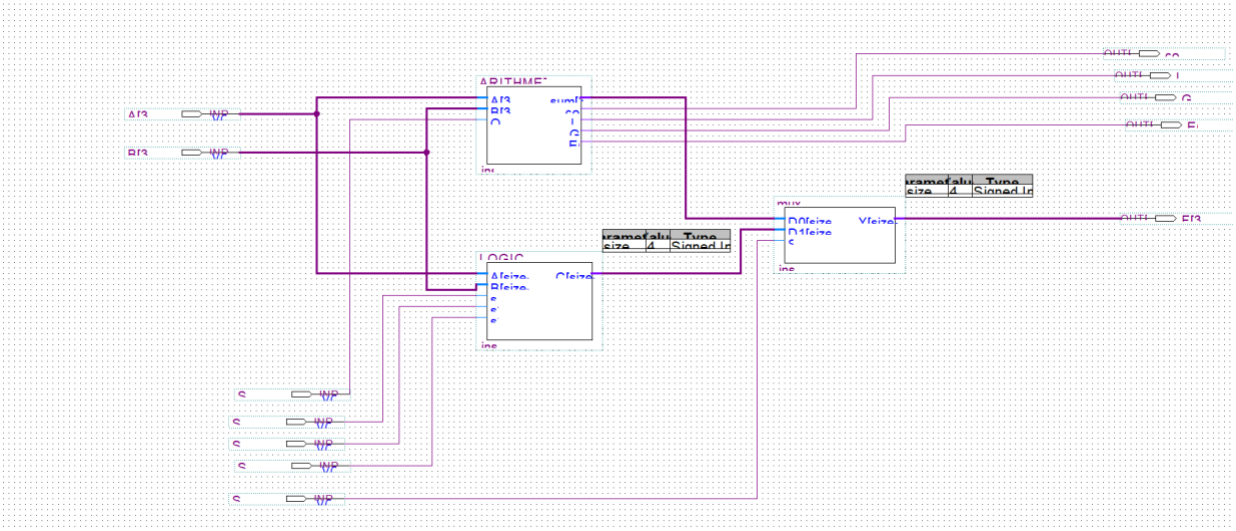

Figure 7: Schematic of ALU

```
1  module ALU(A,B,S0,S1,S2,S3,S4,EQ,GT,LT,F);
2
3  input [3:0] A,B;
4  input S0,S1,S2,S3,S4;
5  output [3:0] F;
6  output EQ,GT,LT;
7  wire [3:0] AU,LU,cout;
8
9  ARITHMETIC_UNIT U1(A,B,S0,AU,cout,LT,GT,EQ);
10 LOGIC_UNIT U2(A,B,S1,S2,S3,LU);
11 mux21 U3(F,AU,LU,S4);
12 endmodule
13
```

Figure 8: Verilog Code of ALU

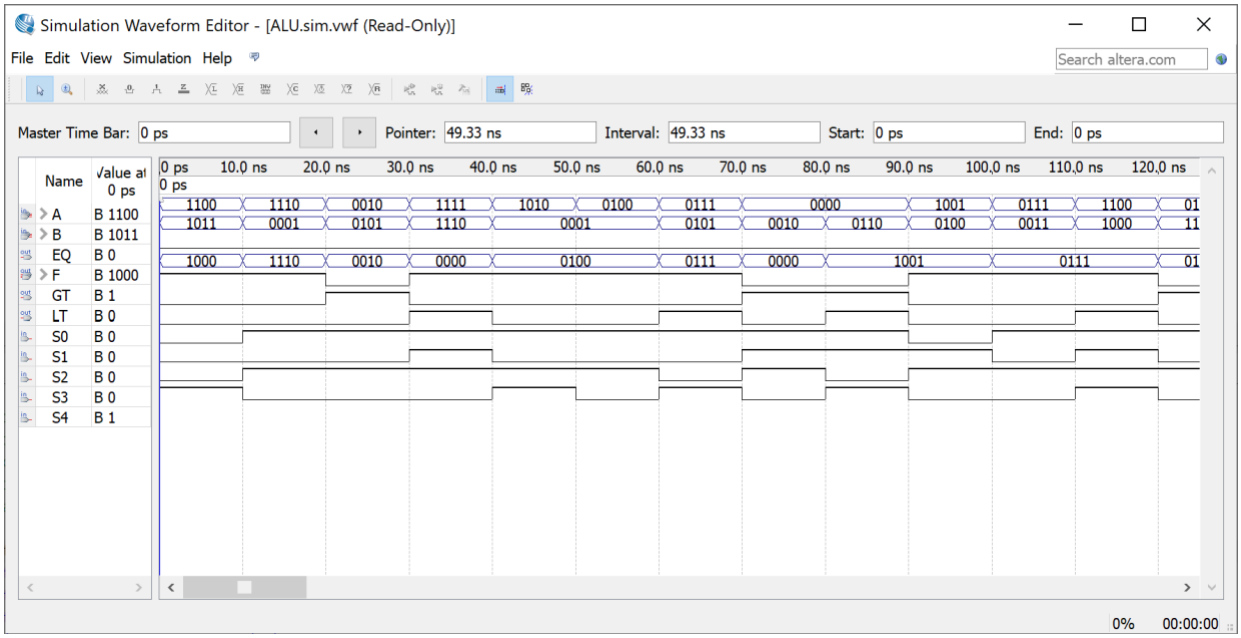

Figure 9: Waveform of ALU

QUESTION 2.2.8.2

Write a testbench for your design using Modelsim® and simulate your ALU.

C:/Users/mertc/Desktop/ALU/Modelsim/ALU_tb.v (/ALU_tb) - Default

```
Ln#
 1  module ALU_tb();
 2
 3  reg [3:0] A,B;
 4  reg S0,S1,S2,S3,S4;
 5  wire [3:0] F;
 6  wire EQ,GT,LT;
 7
 8  ALU DUT(A,B,S0,S1,S2,S3,S4,EQ,GT,LT,F);
 9
10  always
11  begin
12    A = 4'b1000; B = 4'b0100; S0 = 1'b0; S1 = 1'b0; S2 = 1'b0; S3 = 1'b0; S4 = 1'b0; #100;
13    A = 4'b1010; B = 4'b0101; S0 = 1'b0; S1 = 1'b0; S2 = 1'b0; S3 = 1'b0; S4 = 1'b1; #100;
14    A = 4'b1111; B = 4'b0110; S0 = 1'b0; S1 = 1'b0; S2 = 1'b0; S3 = 1'b1; S4 = 1'b0; #100;
15    A = 4'b0001; B = 4'b0100; S0 = 1'b0; S1 = 1'b0; S2 = 1'b0; S3 = 1'b1; S4 = 1'b1; #100;
16    A = 4'b1000; B = 4'b1000; S0 = 1'b0; S1 = 1'b0; S2 = 1'b1; S3 = 1'b0; S4 = 1'b0; #100;
17    A = 4'b1001; B = 4'b0110; S0 = 1'b0; S1 = 1'b0; S2 = 1'b1; S3 = 1'b0; S4 = 1'b1; #100;
18    A = 4'b1001; B = 4'b1001; S0 = 1'b0; S1 = 1'b0; S2 = 1'b1; S3 = 1'b1; S4 = 1'b0; #100;
19    A = 4'b0100; B = 4'b0010; S0 = 1'b0; S1 = 1'b0; S2 = 1'b1; S3 = 1'b1; S4 = 1'b1; #100;
20    A = 4'b0111; B = 4'b1000; S0 = 1'b0; S1 = 1'b1; S2 = 1'b0; S3 = 1'b0; S4 = 1'b0; #100;
21    A = 4'b0010; B = 4'b1100; S0 = 1'b0; S1 = 1'b1; S2 = 1'b0; S3 = 1'b0; S4 = 1'b1; #100;
22    A = 4'b0110; B = 4'b1000; S0 = 1'b0; S1 = 1'b1; S2 = 1'b0; S3 = 1'b1; S4 = 1'b0; #100;
23    A = 4'b0010; B = 4'b1001; S0 = 1'b0; S1 = 1'b1; S2 = 1'b0; S3 = 1'b1; S4 = 1'b1; #100;
24    A = 4'b0001; B = 4'b0001; S0 = 1'b0; S1 = 1'b1; S2 = 1'b1; S3 = 1'b0; S4 = 1'b0; #100;
25    A = 4'b1110; B = 4'b0100; S0 = 1'b0; S1 = 1'b1; S2 = 1'b1; S3 = 1'b0; S4 = 1'b1; #100;
26    A = 4'b0100; B = 4'b1111; S0 = 1'b0; S1 = 1'b1; S2 = 1'b1; S3 = 1'b1; S4 = 1'b0; #100;
27    A = 4'b0101; B = 4'b1110; S0 = 1'b0; S1 = 1'b1; S2 = 1'b1; S3 = 1'b1; S4 = 1'b1; #100;
28    A = 4'b0110; B = 4'b1101; S0 = 1'b1; S1 = 1'b0; S2 = 1'b0; S3 = 1'b0; S4 = 1'b0; #100;
29    A = 4'b1000; B = 4'b1100; S0 = 1'b1; S1 = 1'b0; S2 = 1'b0; S3 = 1'b0; S4 = 1'b1; #100;
30    A = 4'b1001; B = 4'b0111; S0 = 1'b1; S1 = 1'b0; S2 = 1'b0; S3 = 1'b1; S4 = 1'b0; #100;
31    A = 4'b1010; B = 4'b0110; S0 = 1'b1; S1 = 1'b0; S2 = 1'b0; S3 = 1'b1; S4 = 1'b1; #100;
32    A = 4'b1100; B = 4'b0101; S0 = 1'b1; S1 = 1'b0; S2 = 1'b1; S3 = 1'b0; S4 = 1'b0; #100;
33    A = 4'b1101; B = 4'b0100; S0 = 1'b1; S1 = 1'b0; S2 = 1'b1; S3 = 1'b0; S4 = 1'b1; #100;
34    A = 4'b1110; B = 4'b0010; S0 = 1'b1; S1 = 1'b0; S2 = 1'b0; S3 = 1'b0; S4 = 1'b0; #100;
35    A = 4'b1111; B = 4'b0001; S0 = 1'b1; S1 = 1'b1; S2 = 1'b1; S3 = 1'b1; S4 = 1'b1; #100;
36  end
37  endmodule
38
```

Figure 10: TestBench of ALU

Figure 11: Simulation of ALU

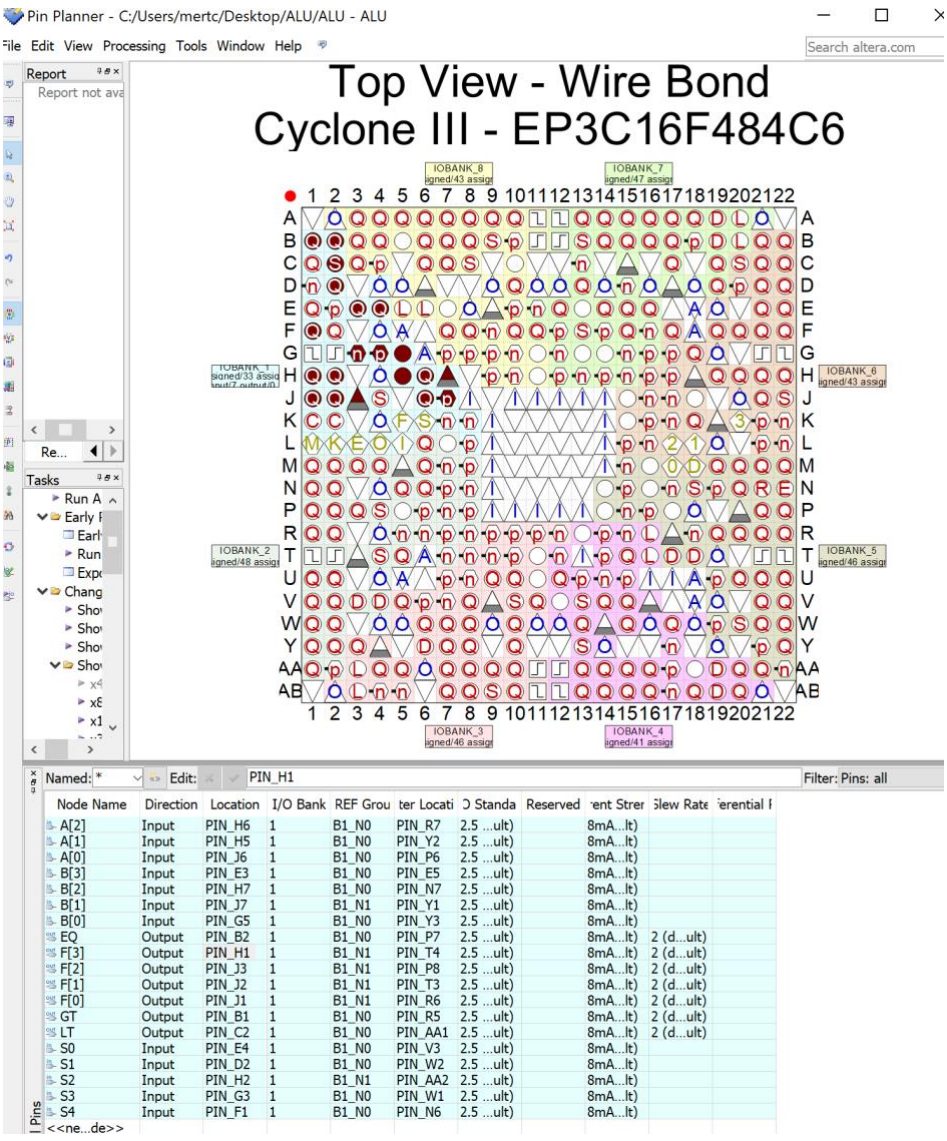

Figure 12: Pin Assignments of ALU

Comments:

- I have learned that ALU is a top level design of what we do in the labs until now. It includes the Logic Unit and Arithmetic Unit. That means, it performs addition, subtraction, comparison, and basic operations.
- I have used an extra selection input for the 2-to-1 multiplexer which connects the ALU and Logic Unit.
- Firstly, I copied all the verilog files in LAB-2, and I created a new project. Then, I assigned the ALU as top level entity. After that, I created symbols of arithmetic unit and logic unit in order to use them in schematic. At the end of this part, I completed my schematic.
- After completing the schematic, I wrote the verilog code. Before calling the modules which are arithmetic and logic units, I declared the inputs and outputs.
- When the Verilog code finished, I compiled a Waveform to be sure that my expectations are correct in both functional and timing.
- After being sure, I switched to Modelsim, and wrote the testbench. I am sure that my results are correct by comparing with truth table. For example, between 300ps and 400ps, S4 is 1. That means, Logic Operations will be performed. To learn which operation will be performed, we need to look at S1,S2 and S3. It is 001, and it means XOR operation. We know that A is 0001 and B is 0100. If we look at F, we see that (0001 XOR 0100) = 0101 as we expected. Also, comparison results can be seen. A is less than B, so LT is ‘1’.
- Also, I have implemented my pin assignments to be able to see the least significant bit at rightmost. In addition, the first four switch represents the A, the other four represents the B. I have used some of the push buttons for selections because the number of switchs are not enough in the board.