Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

3C7 DIGITAL SYSTEMS DESIGN LABORATORY

Department of Electronic and Electrical Engineering

Assignment 1: MINI ALU

NAME: SHRUTI KATHURIA

TCD ID: 21355061

Question:

MINI ARITHMETIC LOGIC UNIT (MINI-ALU)

You need to **design, write/modify** the Verilog modules for the following functions, and test a "mini" arithmetic logic unit. An **arithmetic logic unit (ALU)** uses combinatorial logic to implement common arithmetic and logical functions. A typical ALU has a wide range of functionality from **addition to bit shifting**. Your ALU will provide a **narrow range of functions** performed on **two 6-bit inputs A and B**. A and B are **in 2's complement format**. The **output of the ALU is a 6-bit number** X, also **in 2's complement** form as appropriate, and the input **fxn** controls the output as follows:

fxn	X[5:0]
000	A
001	B
010	-A
011	-B
100	A<B (is A less than B)
101	(A *nxor* B) (Bitwise XNOR)
110	A+B
111	A-B

IMPLEMENTATION:

The aim of the assignment was to design a Mini Athematic Logical Unit. The ALU takes in two 6-bit numbers as inputs, a 3-bit input for functionality (add, subtract, comparison, etc.) and outputs a 6-bit number depending upon the functionality. We were supposed to re-use our blocks of code from the previous labs such as the ripple carry adder and the comparator and use them as building blocks for the ALU.

The implementation of the mini ALU is done on the basys 3 board by using different modules in this assignment.

DESIGN STRATEGY :

1. **Fxn = 000 (A)**
   For the above function, the expected output was the first input. I made use of a ripple carry adder to implement this. I passed in the first input and I passed the second input as 0. The functionality selected was addition. (A + 0 = A).

2. **Fxn = 001 (B)**
   For the above function, the expected output was the second input. I made use of a ripple carry adder to implement this. I passed in the first input as 0 and the second input B. The functionality selected was addition. (0 + B = B).

3. **Fxn = 010 (-A)**
   For the above function, the expected output was negation of the first input. I made use of a 2's compliment module to implement this. It inverts all bits of the input and then adds 1 to the LSB of the inverted number and hence, gives the expected output.

4. **Fxn = 011 (-B)**
   For the above function, the expected output was negation of the second input. I made use of a 2's compliment module to implement this. It inverts all bits of the input and then adds 1 to the LSB of the inverted number and hence, gives the expected output.

5. **Fxn = 100 (A<B) (This function does not run and gives an error)**
   For the above function, the expected output was supposed to be high when the condition A<B was true. I made use of a 1-bit comparator as a building block for the 6-bit comparator. The 1-bit comparator asserts 1 if the first input is less than the second input.

6. **Fxn = 101 (A xnor B)**
   For the above function, the expected output was the bitwise xnor of the inputs. I made use of a 6-bit bitwise xnor module for this.

7. **Fxn = 110 (A+B)**
   For the above function, the expected output was the sum of the inputs. I made use of the modules full adder and ripple carry adder/subtractor for this to implement this. The inputs were passed, and the function selected was 0 (sel=0, addition) when the ripple carry adder/subtractor module was instantiated.

8. **Fxn = 111 (A-B)**
   For the above function, the expected output was the difference of the inputs. I made use of the modules full adder and ripple carry adder/subtractor for this to implement this. The inputs were passed, and the function selected was 1(sel=1, subtraction) when the ripple carry adder/subtractor module was instantiated.

SIMULATIONS:

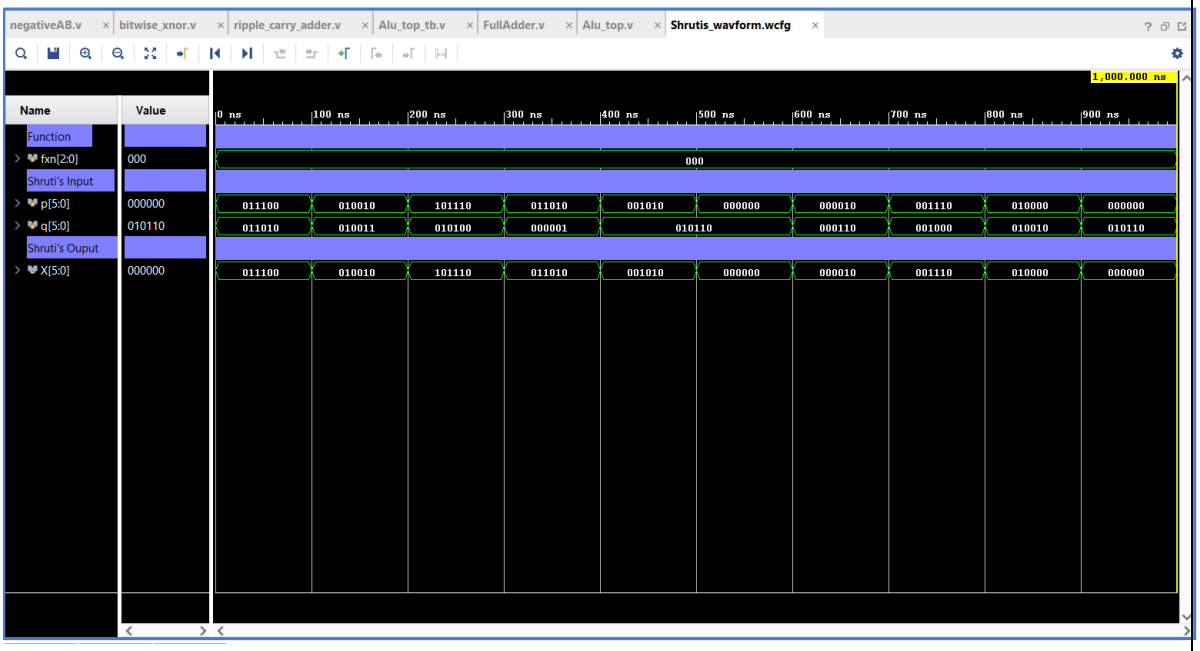


Fig: Fxn = b'000(Waveform for p=x)

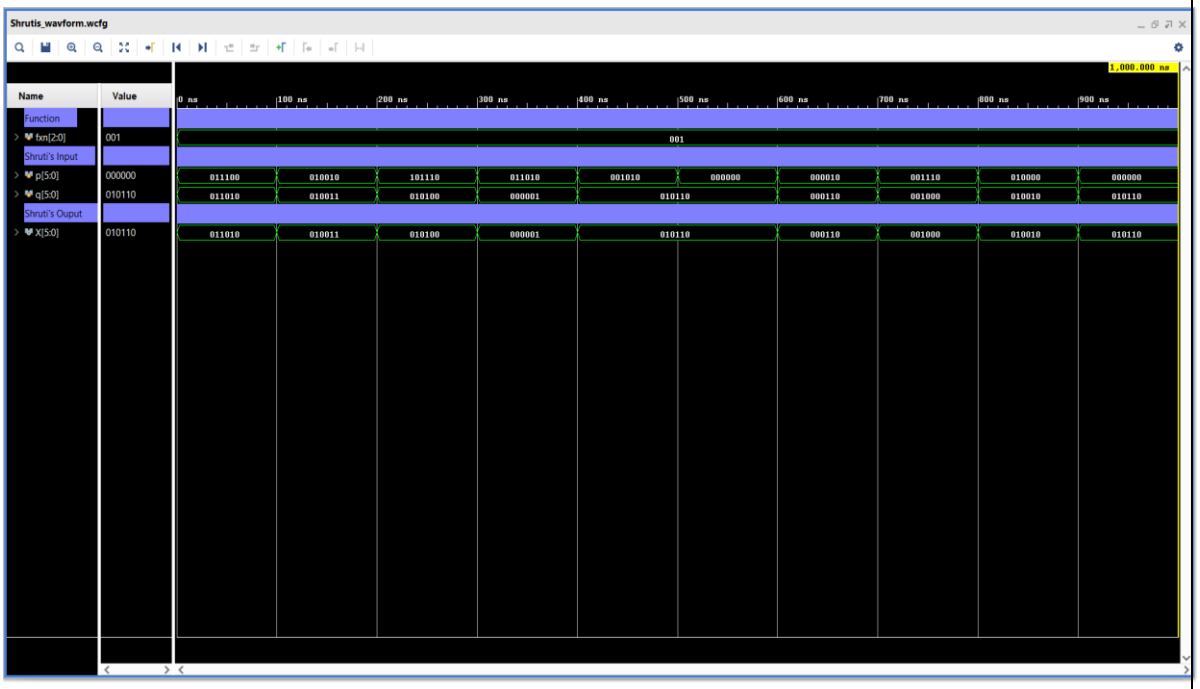


Fig: Fxn = b'001 (Waveform for q=x)

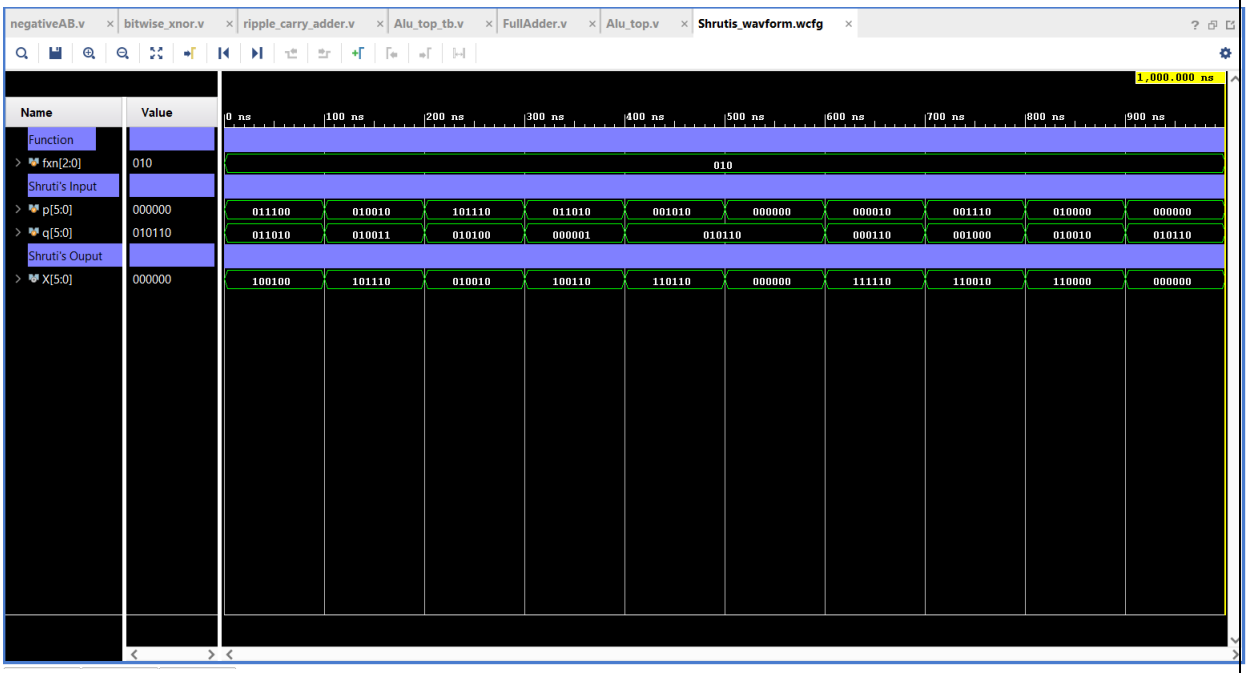


Fig: Fxn = b’010 (Waveform for -p=x)

Fig: Fxn = b’011(Waveform for -q=x)

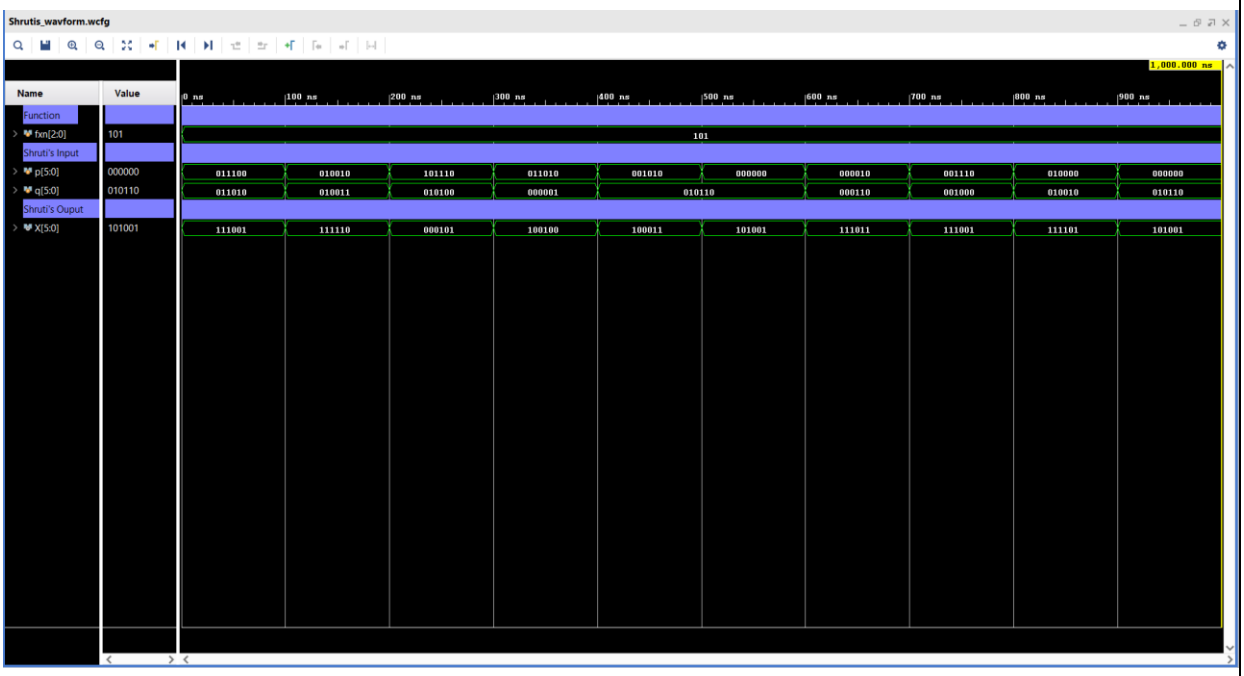


Fig: Fxn = b’101(Waveform for pxnorq)

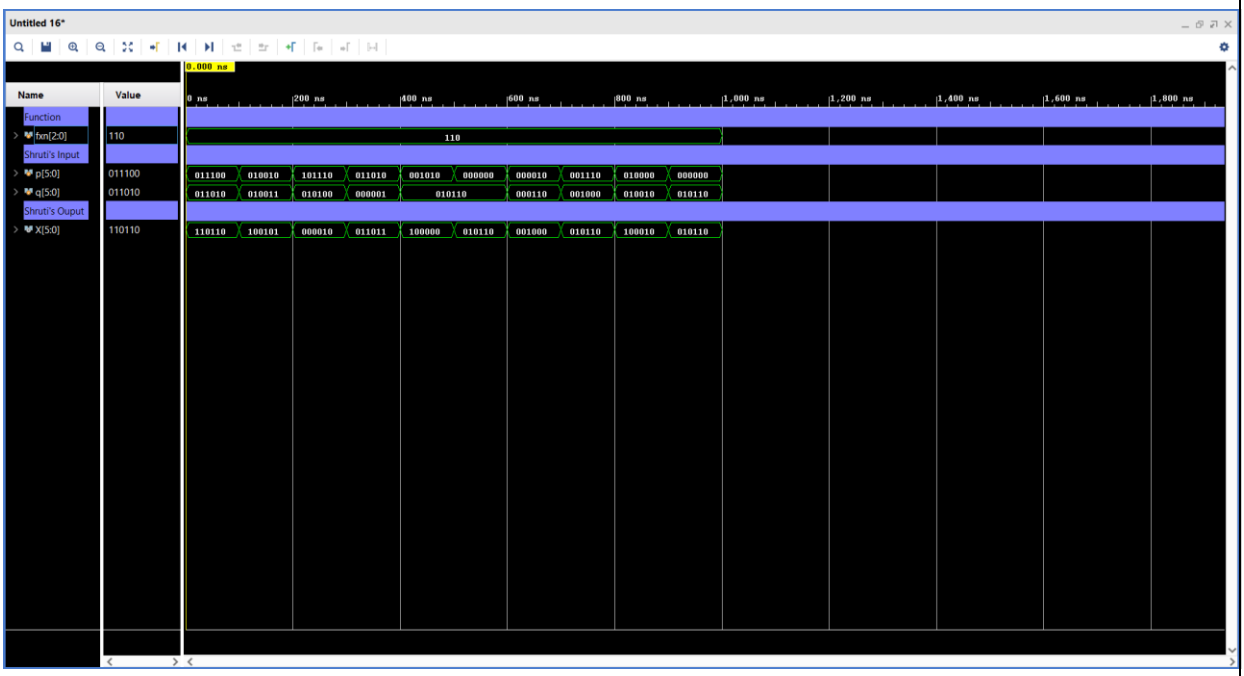


Fig: Fxn = b’110(Waveform for Addition)

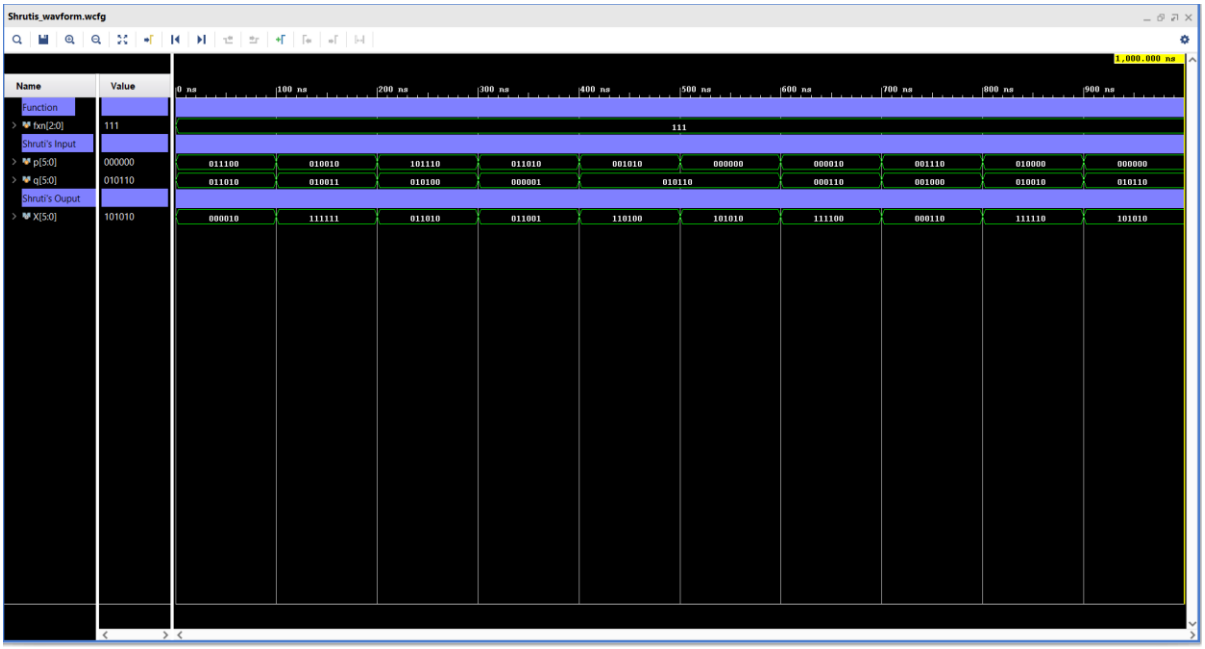


Fig: Fxn = b'111(Waveform for Subtraction)

SOURCES:

The modules that I have used for the implementation of the project are as follows:

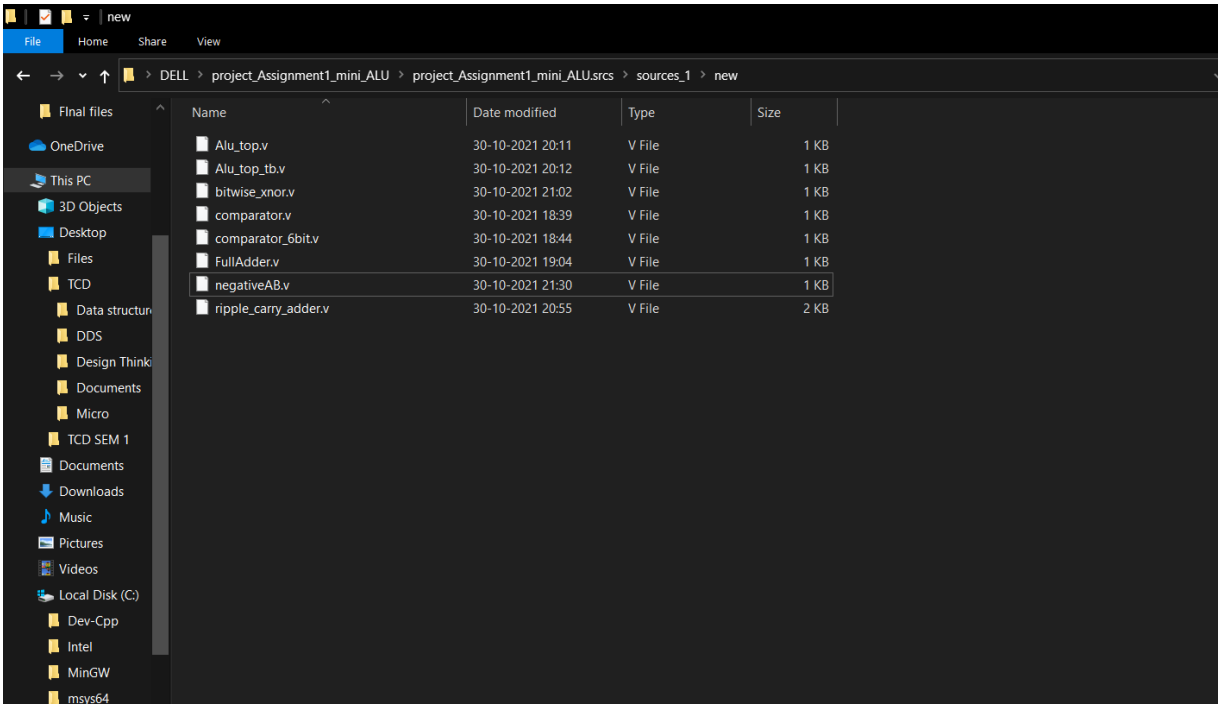


Fig: Directory with the sources in it

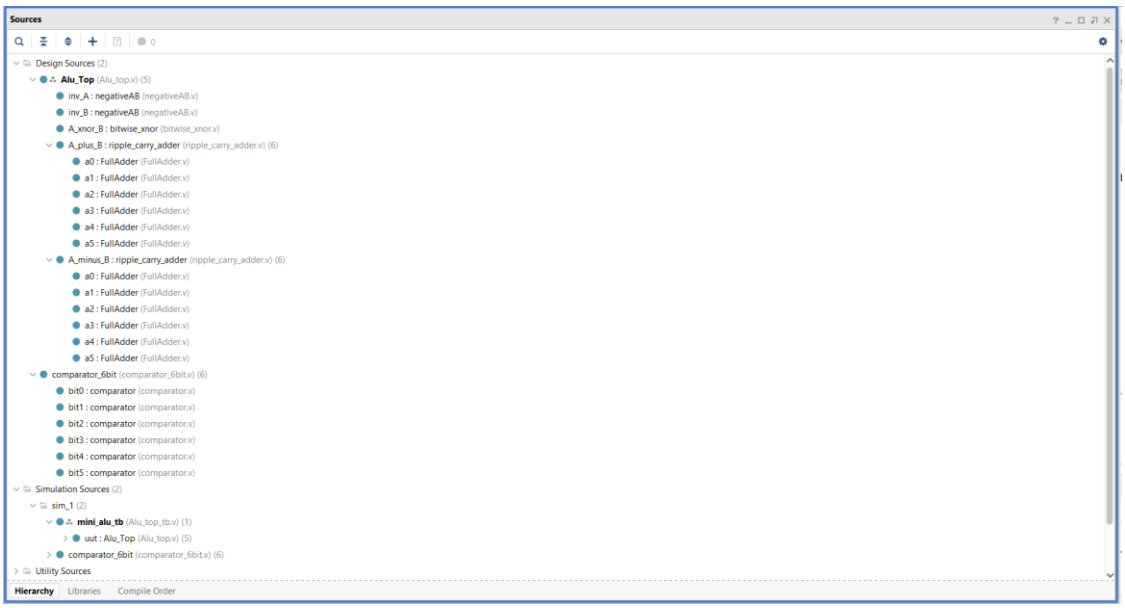


Fig: Design Sources(Hierarchy of the design)

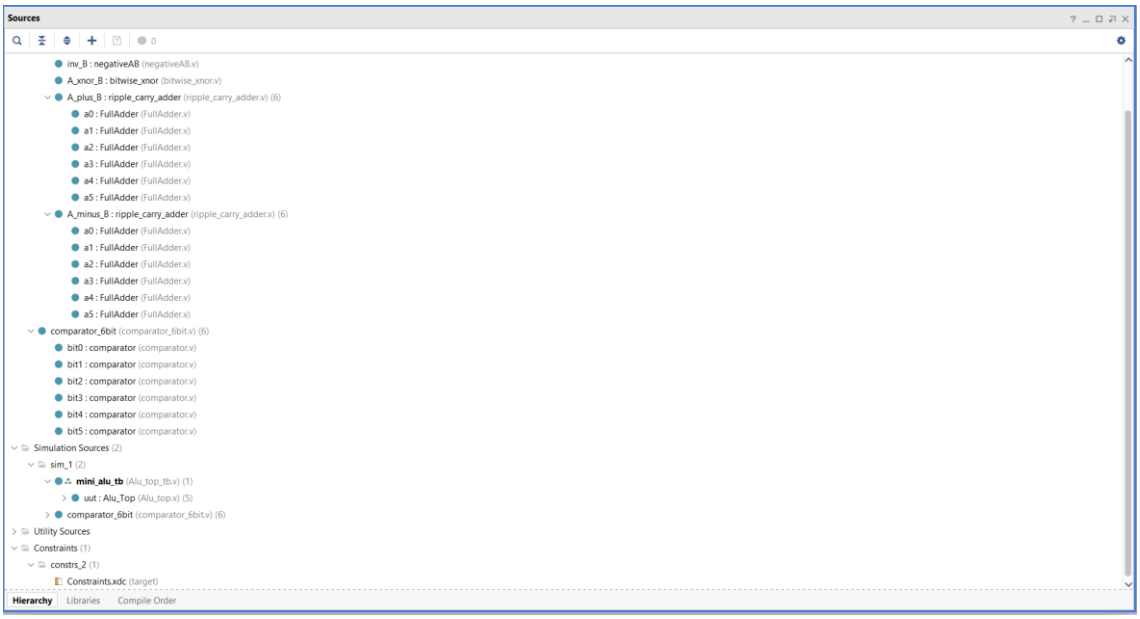


Fig: Design Sources

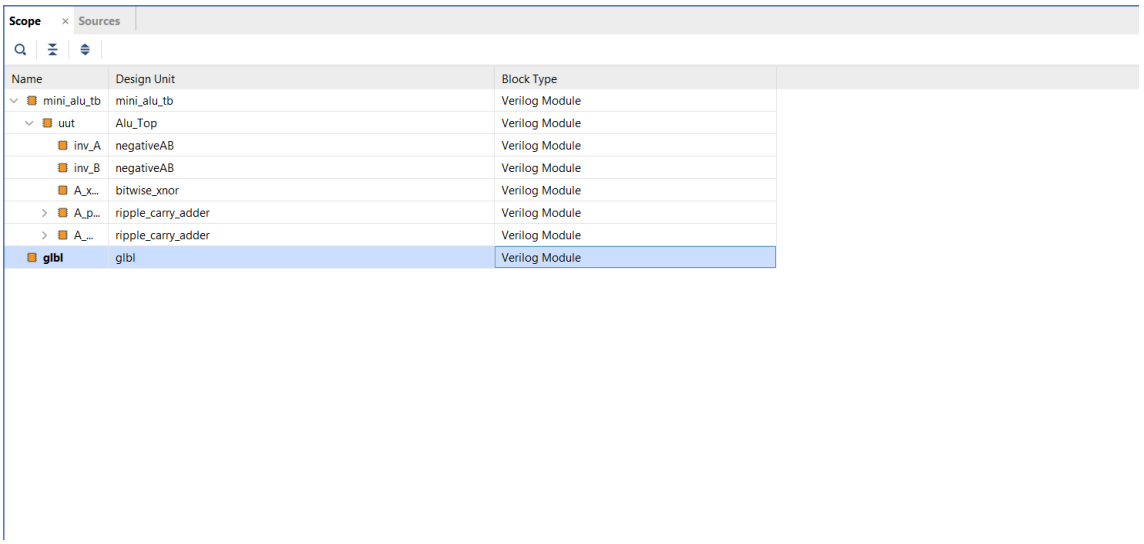


Fig:Sources

SCHEMATICS:

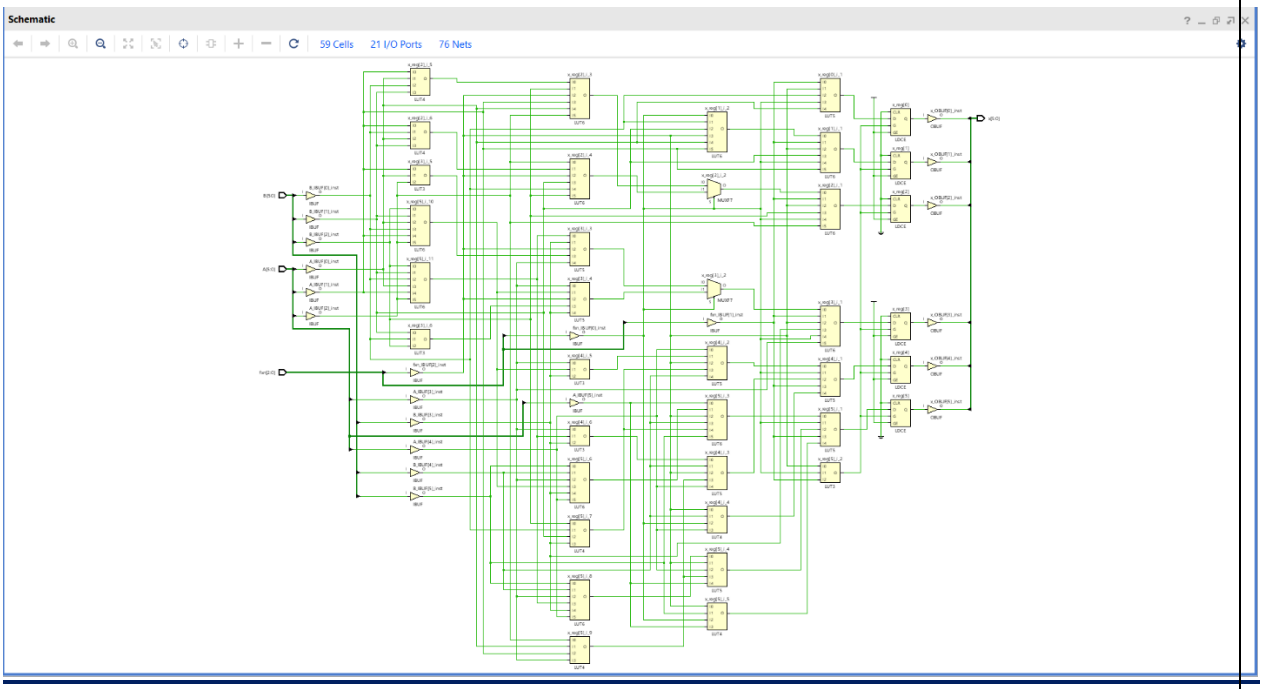


Fig: Elaborated design

Fig: Package

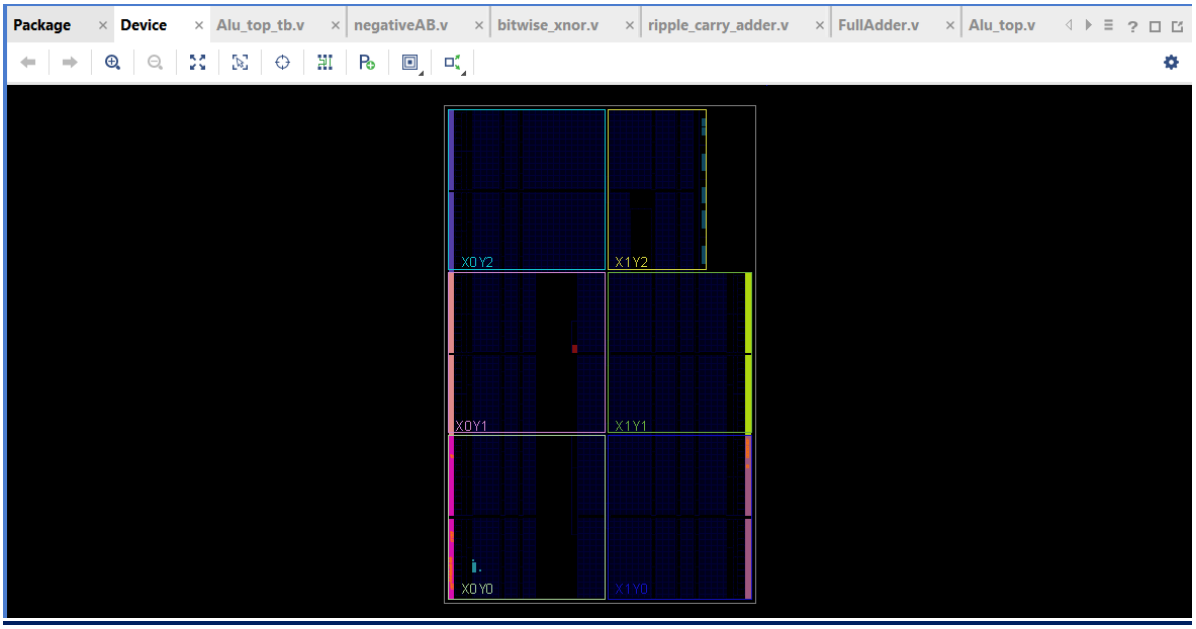
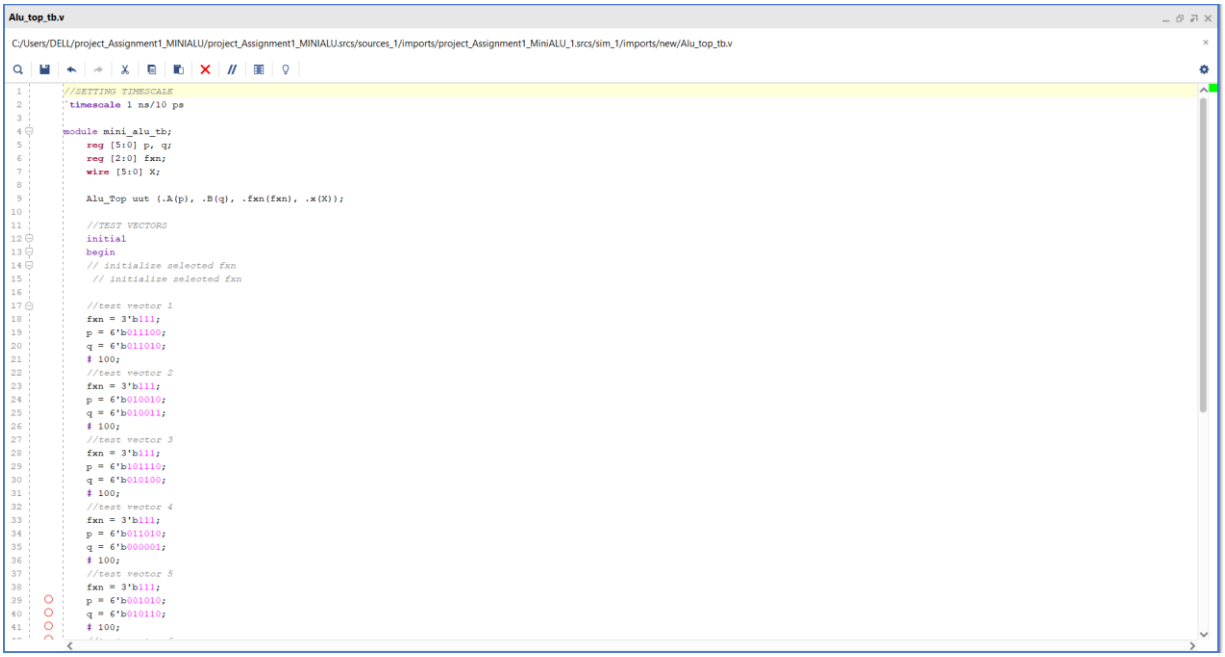


Fig: Implemented Device

CODE SNAPSHOTS:

1. Alu_top_tb testbench code:



Alu_top_tb.v

C:/Users/DELL/project_Assignment1_MINIALU/project_Assignment1_MINIALU.srcs/sources_1/imports/project_Assignment1_MiniALU_1.srcs/sim_1/imports/new/Alu_top_tb.v

```
    p = 6'b101110;
    q = 6'b010100;
    # 100;
    //test vector 4
    fxn = 3'b111;
    p = 6'b011010;
    q = 6'b000001;
    # 100;
    //test vector 5
    fxn = 3'b111;
    p = 6'b001010;
    q = 6'b010110;
    # 100;
    //test vector 6
    fxn = 3'b111;
    p = 6'b000000;
    q = 6'b010110;
    # 100;
     //test vector 7
     fxn = 3'b111;
    p = 6'b000010;
    q = 6'b000110;
    # 100;
     //test vector 8
     fxn = 3'b111;
    p = 6'b001110;
    q = 6'b001000;
    # 100;
     //test vector 9
     fxn = 3'b111;
    p = 6'b010000;
    q = 6'b010010;
    # 100;
     //test vector 10
     fxn = 3'b111;
    p = 6'b000000;
    q = 6'b010110;
    # 100;

    end//STOP
endmodule
```

2.Alu_Top code

Alu_top.v

C:/Users/DELL/project_Assignment1_MINIALU/project_Assignment1_MINIALU.srcs/sources_1/imports/project_Assignment1_MiniALU_1.srcs/sources_1/imports/new/Alu_top.v

```
// THIS MODULE IMPLEMENTS THE ALU BY CALLING OTHER MODULES FOR EACH FUNCTION
module Alu_Top
    (
    input wire [5:0] A, B,  // 6-bit - inputs a,b
    input wire [2:0] fxn,   // 3-bit - input fxn
    output reg [5:0] x      // 6-bit - output x
    );

    wire [5:0] invA, invB, AxorB, AplusB, AminusB;  //TEMPORARY 6-BIT WIRES FOR THE OUTPUT OF EACH MODULE
    wire AlessthanB;    //TEMPORARY 1-BIT VARIABLE FOR THE 1-BIT OUTPUT OF THE >= CIRCUIT

    // instantiation to modules
    negativeAB inv_A (.a(A), .s(invA));    // X = -A
    negativeAB inv_B (.a(B), .s(invB));    // X = -B

    bitwise_xnor A_xor_B(.a(A), .b(B), .xor_(AxorB)); // X = A ^ B
    ripple_carry_adder A_plus_B (.x(A), .y(B), .sel(0), .sum(AplusB)); // X = A + B
    ripple_carry_adder A_minus_B (.x(A), .y(B), .sel(1), .sum(AminusB));   // X = A - B

    // control for fxn call
    always @(*)
    begin
        if (fxn == 3'b000)  {x} = {A};                  // X = A  This is according to the question given.
        if (fxn == 3'b001)  {x} = {B};                  // X = B
        if (fxn == 3'b010)  {x} = {invA};               // X = -A
        if (fxn == 3'b011)  {x} = {invB};               // X = -B

        if (fxn == 3'b101)  {x} = {AxorB};              // X = A ^ B
        if (fxn == 3'b110)  {x} = {AplusB};             // X = A + B
        if (fxn == 3'b111)  {x} = {AminusB};            // X = A - B
    end
endmodule
```

3.bitwise xnor code

bitwise_xnor.v

C:/Users/DELL/project_Assignment1_MINIALU/project_Assignment1_MINIALU.srcs/sources_1/imports/project_Assignment1_MiniALU_1.srcs/sources_1/imports/new/bitwise_xnor.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 30.10.2021 18:35:48
// Design Name:
// Module Name: bitwise_xnor
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


`timescale 1ns / 1ps
module bitwise_xnor(
input [5:0] a,
input [5:0] b,
output [5:0] xor_
);

assign xor_ = a ~^b;
endmodule
```

4. Full Adder code

FullAdder.v

C:/Users/DELL/project_Assignment1_MINIALU/project_Assignment1_MINIALU.srcs/sources_1/imports/project_Assignment1_MiniALU_1.srcs/sources_1/imports/new/FullAdder.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 30.10.2021 19:04:51
// Design Name:
// Module Name: FullAdder
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module FullAdder(a, b, cin, s, cout);
  // a and b are the bits to add
  // cin is carry in
  input wire a, b, cin;

  // s is the sum of a and b. cout is any carry out bit
  // wires since just using assign here
  output wire s, cout;

  // logic for sum and carry
  assign s = cin ^ a ^ b;
  assign cout = (b & cin) | (a & cin) | (a & b);

endmodule
```

5. Six bit Comparator code

comparator_6bit.v

C:/Users/DELL/project_Assignment1_MINIALU/project_Assignment1_MINIALU.srcs/sources_1/imports/project_Assignment1_MiniALU_1.srcs/sources_1/imports/new/comparator_6bit.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 30.10.2021 18:40:42
// Design Name:
// Module Name: comparator_6bit
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps
module comparator_6bit (
input [5:0] in1,
input [5:0] in2,
output [5:0] y
);

comparator bit0 (.i1(in1[0]), .i2(in2[0]), .less_than(y[0]));
comparator bit1 (.i1(in1[1]), .i2(in2[1]), .less_than(y[1]));
comparator bit2 (.i1(in1[2]), .i2(in2[2]), .less_than(y[2]));
comparator bit3 (.i1(in1[3]), .i2(in2[3]), .less_than(y[3]));
comparator bit4 (.i1(in1[4]), .i2(in2[4]), .less_than(y[4]));
comparator bit5 (.i1(in1[5]), .i2(in2[5]), .less_than(y[5]));

endmodule
```

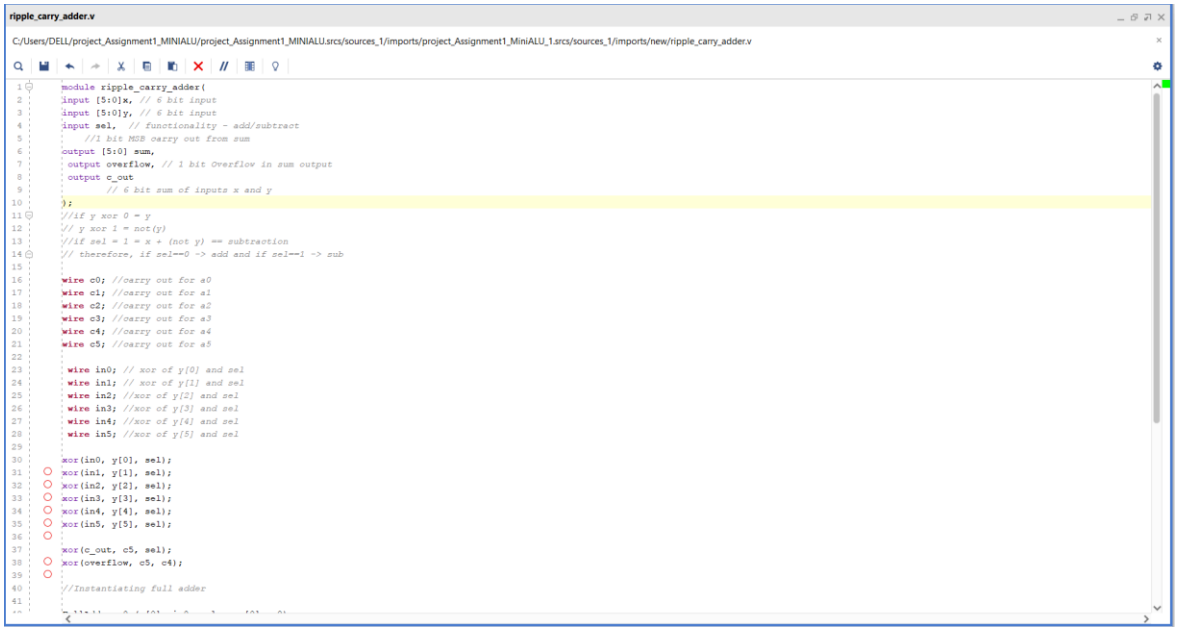
6. One bit comparator code

comparator.v

C:/Users/DELL/project_Assignment1_MINIALU/project_Assignment1_MINIALU.srcs/sources_1/imports/project_Assignment1_MiniALU_1.srcs/sources_1/imports/new/comparator.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 30.10.2021 18:38:49
// Design Name:
// Module Name: comparator
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module comparator(

input i1, input i2,
output less_than
);
assign less_than = ~i1 & i2;
endmodule
```

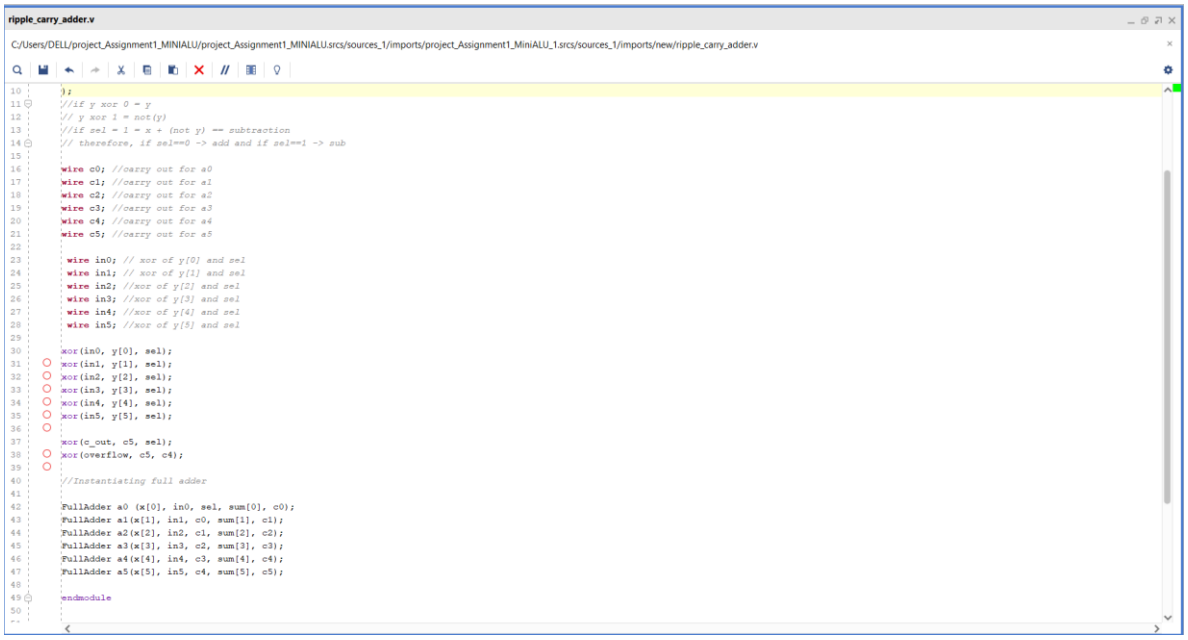
7. Ripple carry adder code



ripple_carry_adder.v

C:/Users/DELL/project_Assignment1_MINIALU/project_Assignment1_MINIALU.srcs/sources_1/imports/project_Assignment1_MiniALU_1.srcs/sources_1/imports/new/ripple_carry_adder.v

```
module ripple_carry_adder(
input [5:0]x, // 6 bit input
input [5:0]y, // 6 bit input
input sel,  // functionality - add/subtract
    //1 bit MSB carry out from sum
output [5:0] sum,
 output overflow, // 1 bit Overflow in sum output
 output c_out
        // 6 bit sum of inputs x and y
);
//if y xor 0 = y
// y xor 1 = not(y)
//if sel = 1 = x + (not y) == subtraction
// therefore, if sel==0 -> add and if sel==1 -> sub

wire c0; //carry out for a0
wire c1; //carry out for a1
wire c2; //carry out for a2
wire c3; //carry out for a3
wire c4; //carry out for a4
wire c5; //carry out for a5

 wire in0; // xor of y[0] and sel
 wire in1; // xor of y[1] and sel
 wire in2; //xor of y[2] and sel
 wire in3; //xor of y[3] and sel
 wire in4; //xor of y[4] and sel
 wire in5; //xor of y[5] and sel

xor(in0, y[0], sel);
xor(in1, y[1], sel);
xor(in2, y[2], sel);
xor(in3, y[3], sel);
xor(in4, y[4], sel);
xor(in5, y[5], sel);

xor(c_out, c5, sel);
xor(overflow, c5, c4);

//Instantiating full adder
```

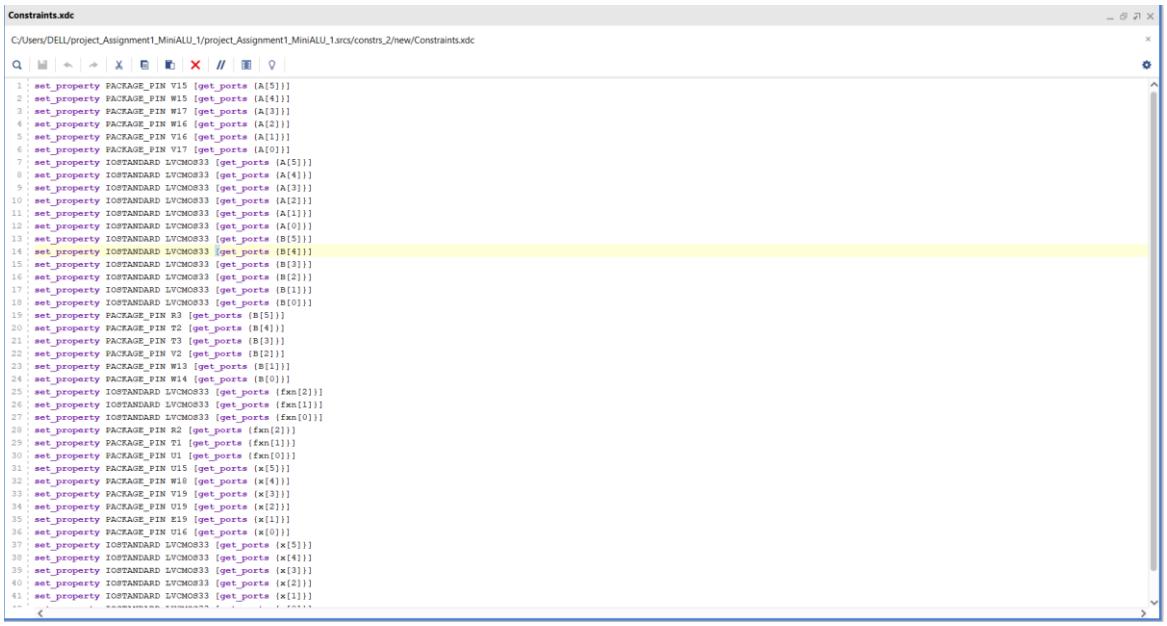


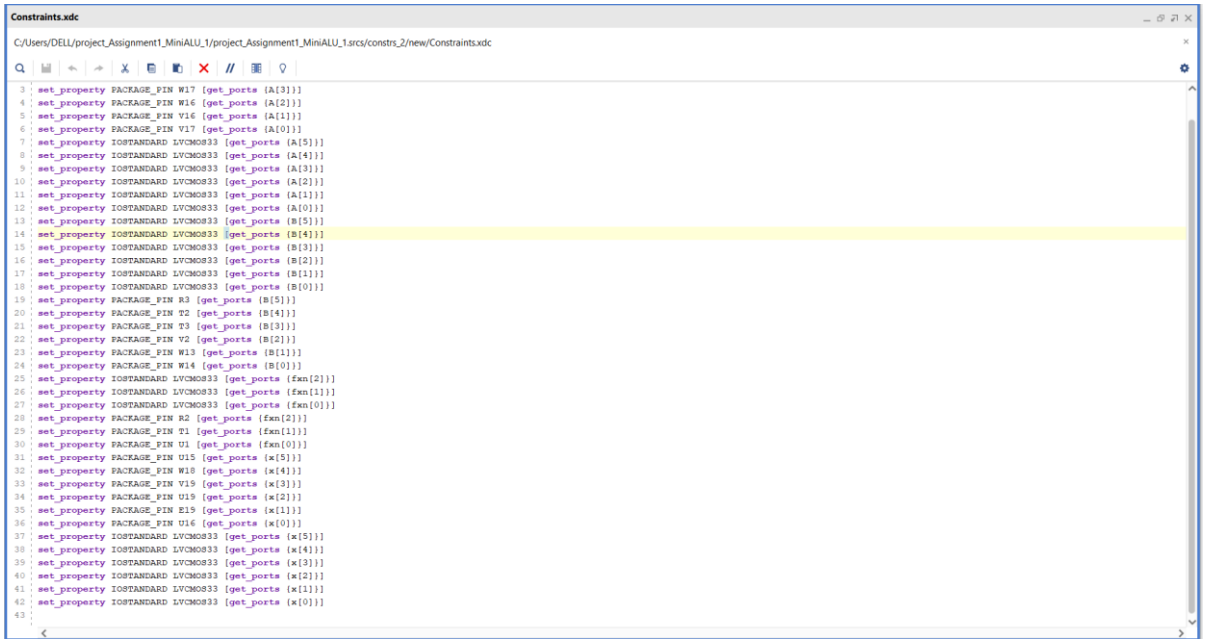
ripple_carry_adder.v

C:/Users/DELL/project_Assignment1_MINIALU/project_Assignment1_MINIALU.srcs/sources_1/imports/project_Assignment1_MiniALU_1.srcs/sources_1/imports/new/ripple_carry_adder.v

```
);
//if y xor 0 = y
// y xor 1 = not(y)
//if sel = 1 = x + (not y) == subtraction
// therefore, if sel==0 -> add and if sel==1 -> sub

wire c0; //carry out for a0
wire c1; //carry out for a1
wire c2; //carry out for a2
wire c3; //carry out for a3
wire c4; //carry out for a4
wire c5; //carry out for a5

 wire in0; // xor of y[0] and sel
 wire in1; // xor of y[1] and sel
 wire in2; //xor of y[2] and sel
 wire in3; //xor of y[3] and sel
 wire in4; //xor of y[4] and sel
 wire in5; //xor of y[5] and sel

xor(in0, y[0], sel);
xor(in1, y[1], sel);
xor(in2, y[2], sel);
xor(in3, y[3], sel);
xor(in4, y[4], sel);
xor(in5, y[5], sel);

xor(c_out, c5, sel);
xor(overflow, c5, c4);

//Instantiating full adder

FullAdder a0 (x[0], in0, sel, sum[0], c0);
FullAdder a1(x[1], in1, c0, sum[1], c1);
FullAdder a2(x[2], in2, c1, sum[2], c2);
FullAdder a3(x[3], in3, c2, sum[3], c3);
FullAdder a4(x[4], in4, c3, sum[4], c4);
FullAdder a5(x[5], in5, c4, sum[5], c5);

endmodule
```

While adding the input and output ports before the bitstream a constraint's.xdc file is generated





DEMO SECTION:

Demo on board:

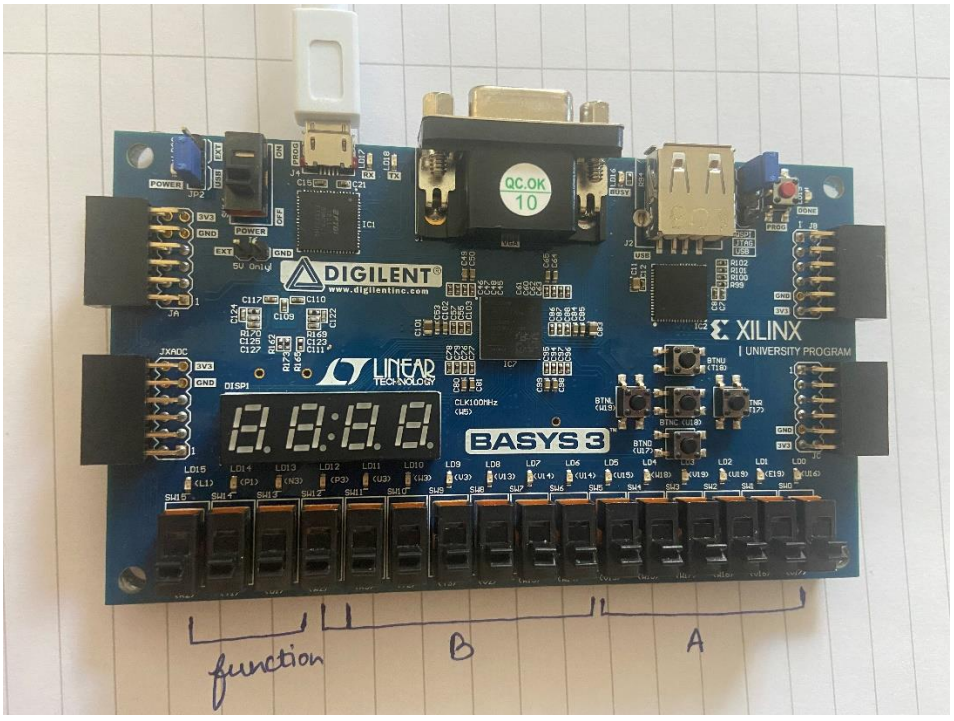
Fxn	Switch
Fxn[0]	U1
Fxn [1]	T1
Fxn [2]	R2

A	Switch
A[0]	V17
A[1]	V16
A[2]	W16
A[3]	W17
A[4]	W15
A[5]	V15

B	Switch
B[0]	W14
B[1]	W13
B[2]	V2
B[3]	T3
B[4]	T2
B[5]	R3

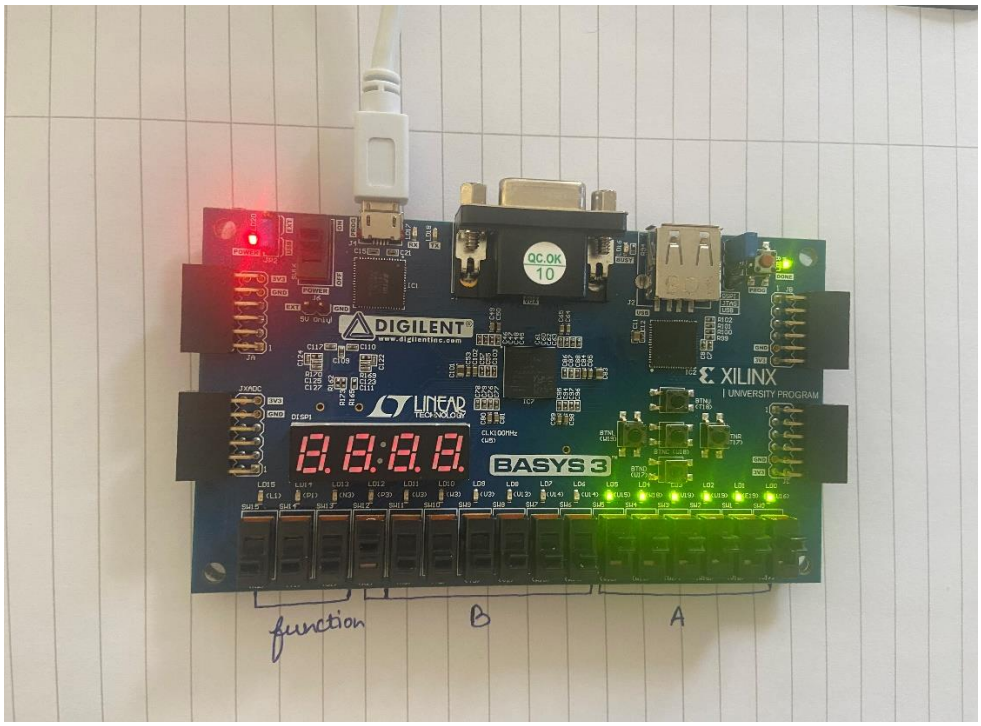
X	LED
X[0]	U16
X[1]	E19
X[2]	U19
X[3]	V19
X[4]	W18
X[5]	U15

BASYS 3 BOARD:

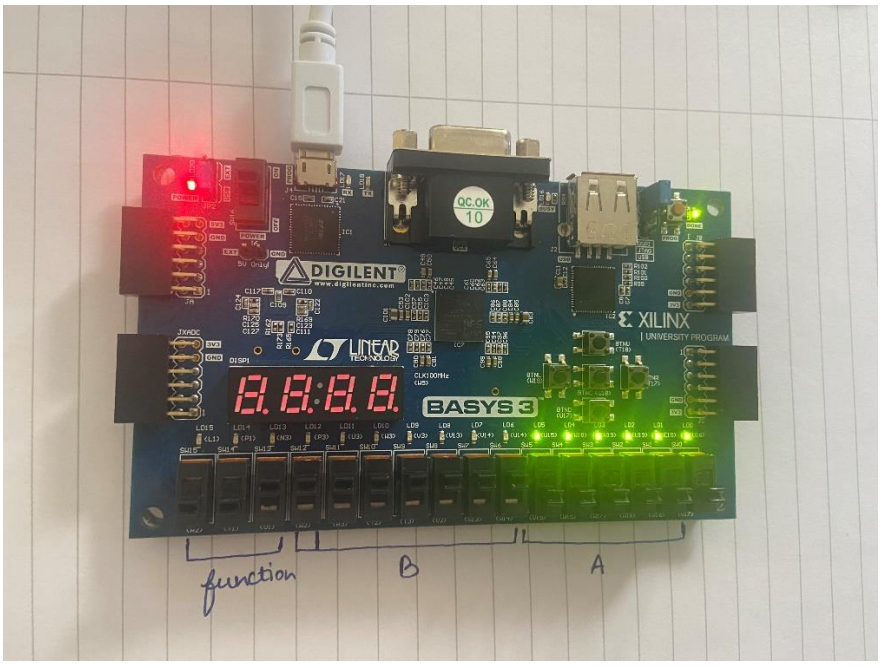


TEST CASES:

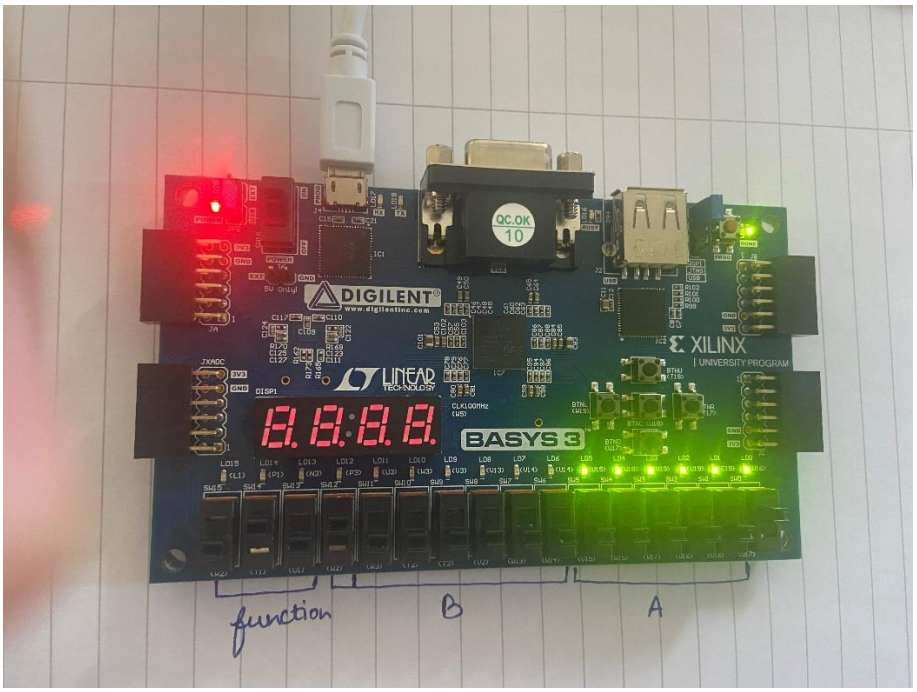
CASE1: fxn=b’000



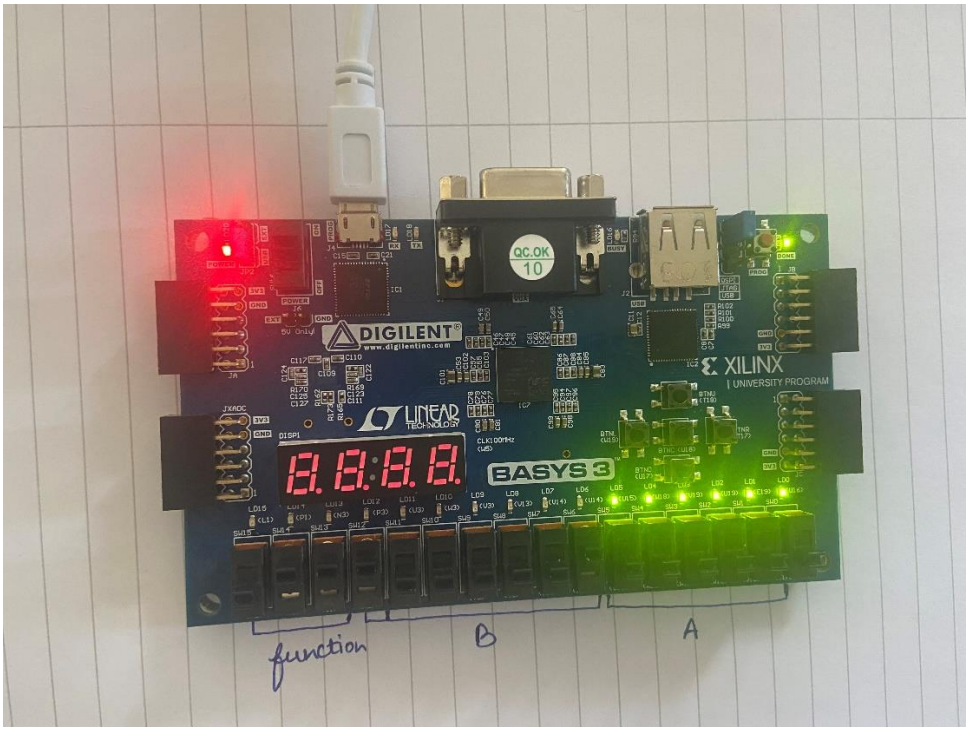
CASE 2: fxn=b’001



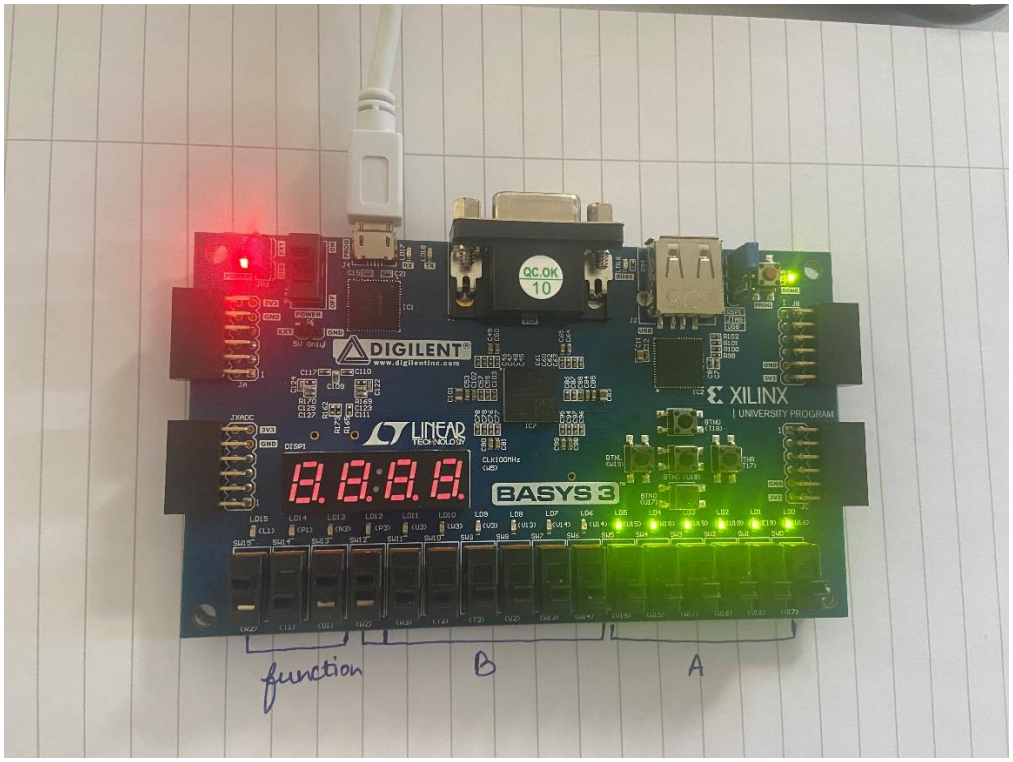
CASE 3: fxn=b’010



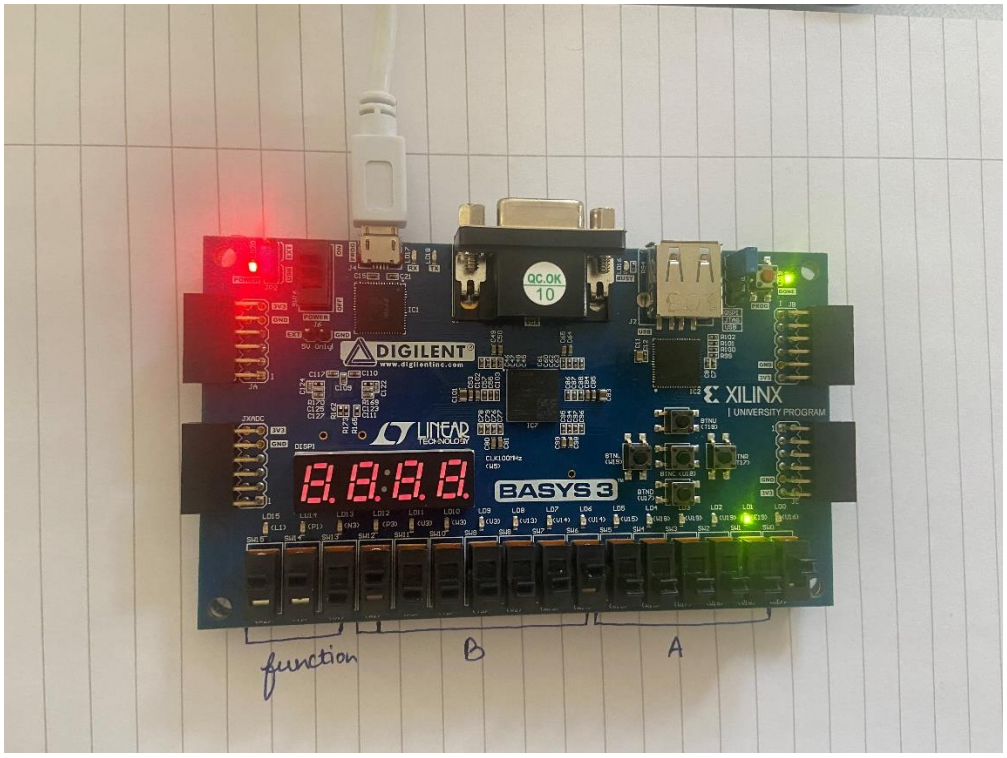
CASE 4 : fxn=b’011



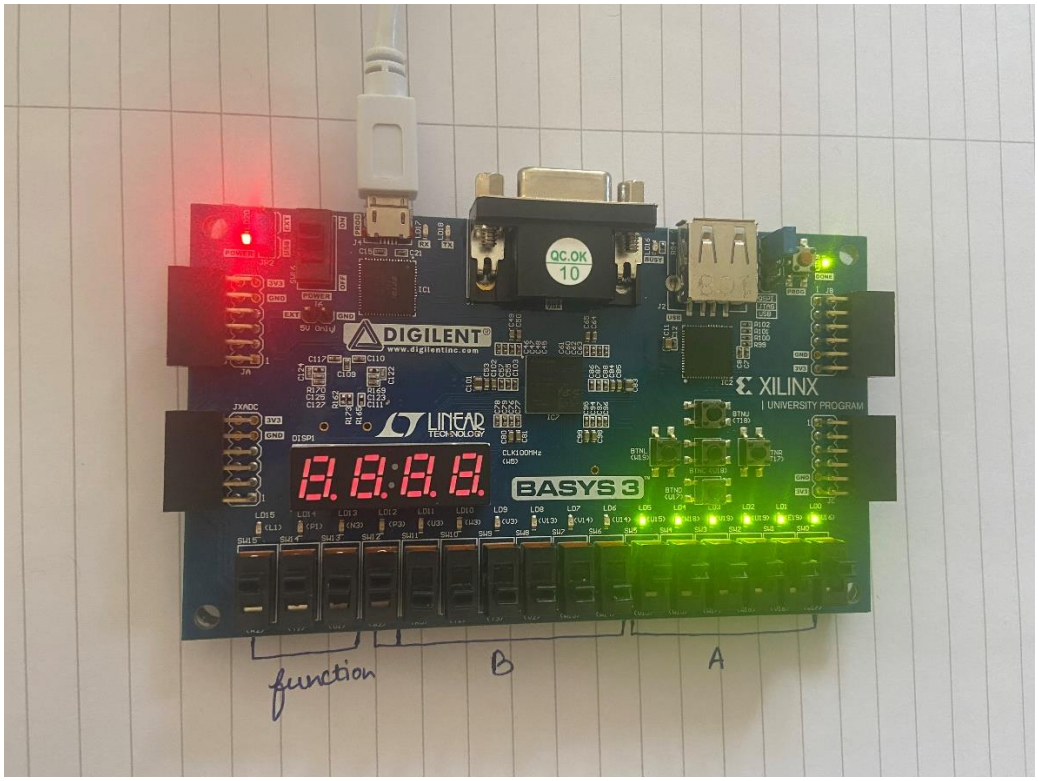
CASE 5: fxn=b'101



CASE 6: fxn=b'110



CASE 7 : fxn=b'111



CONCLUSIONS:

In this assignment I learned how to make modules and add constraints and then form different cases and upload the program upto the basys 3 board and check different functions on it.

APPENDIX:

LabB_ShrutiKathuria, ripple_carry_adder, full_adder, and LabC_ShrutiKathuria which were submitted in the last labs.

SHRUTI KATHURIA 101903325 DIGITAL SYSTEM DESIGNS