Name : Ananya Garg Student ID: 21355233

(Note : This laptop belonged to my brother ''Parag Garg', which is why all directories shows his name instead of mine.)

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

3C7 Digital Systems Design
Assignment 1

Description :

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

Submitted by: Ananya Garg
Student ID: 21355233

Submission date : 30/10/2021

Name : Ananya Garg Student ID: 21355233

(Note : This laptop belonged to my brother ''Parag Garg', which is why all directories shows his name instead of mine.)

1. Implementations: The implementation of the mini-ALU onto the Basys3 board required initialization with modules. We have used some of the modules from our previous labs and have created some this time.

2. Simulations and code snippets: This section is divided into eight cases of functional selection of ALU to operate as defined in description taken from Assignment 1.

    1. **Equal.v :** This section contains the waveform,code and testbench for fxn=000 and fxn=001.

fxn	X[5:0]
000	A
001	B

Code for equal.v

C:/Users/PARAG GARG/Assignment1.xsim/Assignment1.xsim.srcs/sources_1/new/equal.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Create Date: 28.10.2021 20:17:14
4  // Module Name: equal
5  //////////////////////////////////////////////////////////////////////////////////
6  // starting module
7  module equal(
8      // I/O ports
9      input [5:0] I, // taking input as I
10     output [5:0] O // taking output as O
11     );
12     //body
13     assign O=I;     // assigning Output as I since it is an equal function
14                     // O will return A when fxn=000 and B when fxn=001
15
16 endmodule          // ending module
17
18
```

Figure 1.0

Testbench for equal.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Create Date: 28.10.2021 21:59:56
4  // Module Name: equal_testbench
5  //////////////////////////////////////////////////////////////////////////////////
6  module equal_testbench;
7      // signal declaration
8      reg[5:0] I;
9      wire [5:0] O;
10     // instantiate the circuit under test
11     equal uut(I,O);
12     // test vector generator
13     initial
14     begin
15             //test 1
16             I=6'b000001;
17             #200;
18             //test 2
19             I=6'b000010;
20             #200;
21             //test 3
22             I=6'b000100;
23             #200;
24             //test 4
25             I=6'b000110;
26             #200;
27             //test 5
28             I=6'b001000;
29             #200;
30       // stop simulation
31       $stop;
32     end
33 endmodule  // ending module
```

Figure1.1

Waveform for equal.v

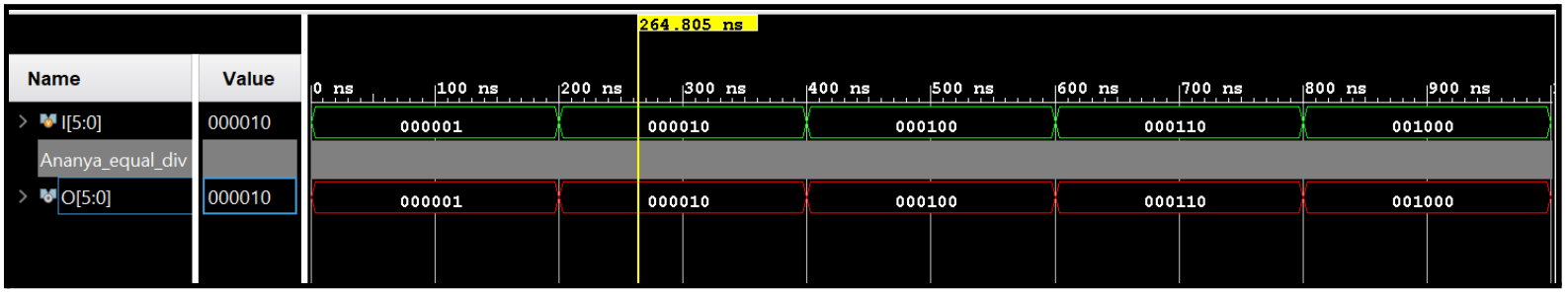

Figure 1.2 : equal.v waveform - radix set to binary

(Note : This laptop belonged to my brother ''Parag Garg', which is why all directories shows his name instead of mine.)

2. **Inverting.v :** This section contains the waveform,code and testbench for fxn=010 and fxn=011.

010	-A
011	-B

Code for inverting.v

C:/Users/PARAG GARG/Assignment1.xsim/Assignment1.xsim.srcs/sources_1/new/inverting.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Create Date: 28.10.2021 20:25:16
4  // Module Name: negate
5  //////////////////////////////////////////////////////////////////////////////////
6  // THIS MODULE CALCULATES THE 2'S COMPLEMENT INVERTION OF THE 6-BIT INPUT
7  // starting module
8  module inverting(
9      // I/O ports
10     input [5:0] I, // taking input as I
11     output [5:0] O // taking output as O
12     );
13     //body
14     assign O= ~I + 000001;    // assigning Output as ~I+000001 since it is an invertion function
15                               // O will return -A when fxn=010 and -B when fxn=011
16
17 endmodule           // ending module
```

Figure 1.3

Testbench for inverting.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Create Date: 28.10.2021 22:14:08
4  // Module Name: negate_testbench
5  module inverting_testbench;
6      // signal declaration
7      reg[5:0] I;
8      wire [5:0] O;
9      // instantiate the circuit under test
10     negate uut(I,O);
11     // test vector generator
12     initial
13     begin
14             //test 1
15             I=6'b000001;
16             #200;
17             //test 2
18             I=6'b000010;
19             #200;
20             //test 3
21             I=6'b000100;
22             #200;
23             //test 4
24             I=6'b000110;
25             #200;
26             //test 5
27             I=6'b001000;
28             #200;
29       // stop simulation
30       $stop;
31     end
32 endmodule  // ending module
```

Figure1.4

Waveform for inverting.v

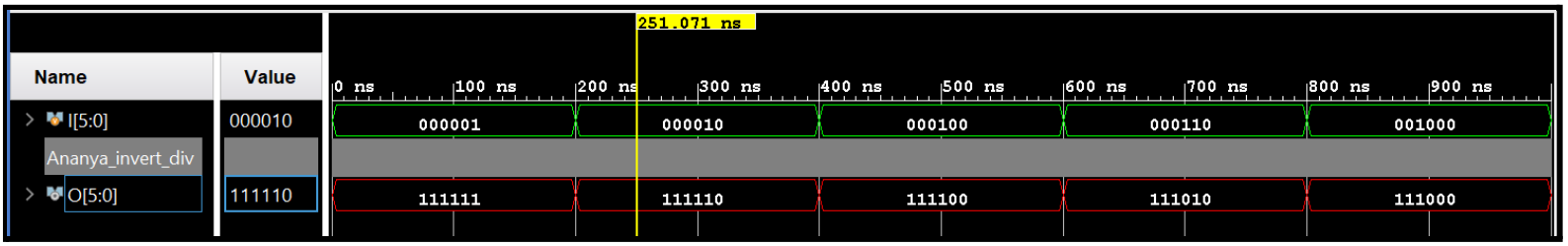

Figure 1.5 : inverting.v waveform - radix set to binary

3. **a<b :**

100	A<B (is A less than B)

I tried to run the code for A<B comparator and used the previous modules to do it but it did not work. I tried to negate the final output of my previous code and made some changes but the error still persists. I have attached a screen grab of the waveform and have submitted the module files as well.

(Note : This laptop belonged to my brother ''Parag Garg', which is why all directories shows his name instead of mine.)

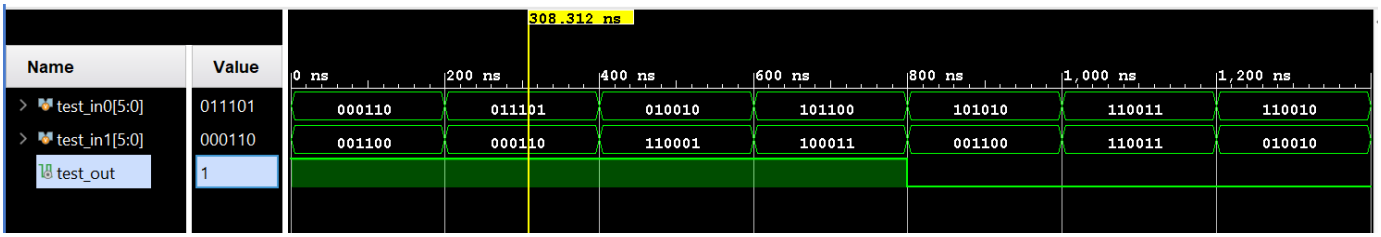

Figure 1.6

4. **XNOR.v :** This section contains the waveform,code and testbench for fxn=101.

101	(A *nxor* B) (Bitwise XNOR)

Code for XNOR.v

C:/Users/PARAG GARG/Assignment1.xsim/Assignment1.xsim.srcs/sources_1/new/xnor.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Create Date: 28.10.2021 20:38:27
4  // Module Name: xnor
5  //////////////////////////////////////////////////////////////////////////////////
6  // THIS MODULE PERFORMS THE BITWISE XOR ON TWO 6-BIT INPUTS
7  // starting module
8  module XNOR(
9       // I/O ports
10     input [5:0] I1, // taking inputs as I1 and I2
11     input [5:0] I2,
12     output [5:0] O  // taking output as O
13     );
14     assign O[0] = I1[0]*I2[0] + (~I1[0])*(~I2[0]);
15     assign O[1] = I1[1]*I2[1] + (~I1[1])*(~I2[1]);
16     assign O[2] = I1[2]*I2[2] + (~I1[2])*(~I2[2]);
17     assign O[3] = I1[3]*I2[3] + (~I1[3])*(~I2[3]);
18     assign O[4] = I1[4]*I2[4] + (~I1[4])*(~I2[4]);
19     assign O[5] = I1[5]*I2[5] + (~I1[5])*(~I2[5]);
20
21 endmodule  //ending module
```

Figure 1.7

Testbench for XNOR.v(XNOR_testbench.v)

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Create Date: 28.10.2021 22:21:21
4  // Module Name: XNOR_testbench
5  //////////////////////////////////////////////////////////////////////////////////
6  module XNOR_testbench;
7      // signal declaration
8      reg [5:0] I1;
9      reg [5:0] I2;
10     wire [5:0] O;
11     // instantiate the circuit under test
12     XNOR uut(I1,I2,O);
13     // test vector generator
14     initial
15     begin
16             //test 1
17             I1=6'b000001;
18             I2=6'b000010;
19             #200;
20             //test 2
21             I1=6'b000010;
22             I2=6'b000110;
23             #200;
24             //test 3
25             I1=6'b000100;
26             I2=6'b001001;
27             #200;
28             //test 4
29             I1=6'b000110;
30             I2=6'b110000;
31             #200;
32             //test 5
33             I1=6'b001000;
34             I2=6'b100100;
35             #200;
36              // stop simulation
37             $stop;
38     end
39 endmodule  // ending module
```

Figure1.8

Waveform for XNOR.v

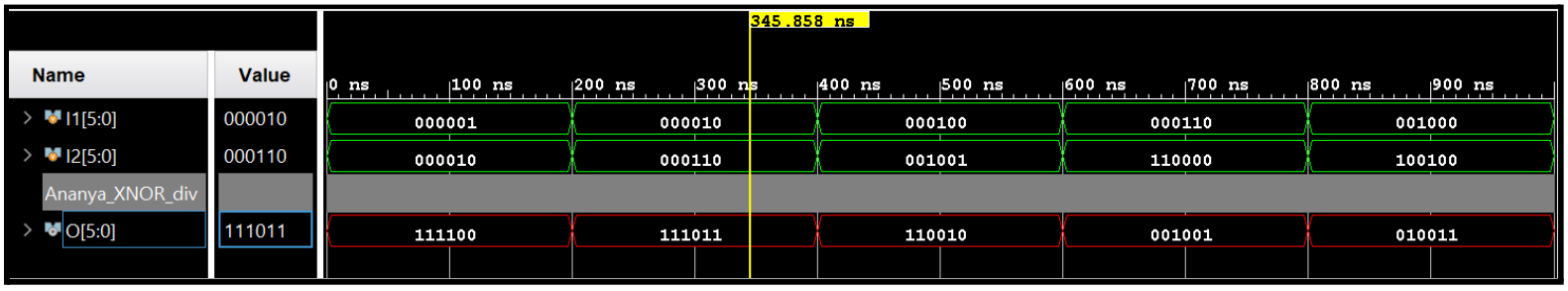

Figure 1.9 : XNOR.v waveform - radix set to binary

(Note : This laptop belonged to my brother ''Parag Garg', which is why all directories shows his name instead of mine.)

5. **adder6bit.v :** This section contains the waveform,code and testbench for fxn=110 and fxn=111.

110	A+B
111	A-B

Code for adder6bit.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Create Date: 29.10.2021 02:10:45
// Module Name: adder6bit
//////////////////////////////////////////////////////////////////////////////////
// THIS MODULE PERFORMS THE ADDITION/SUBTRACTION OF 2 6-BIT NUMBERS
// starting module
module adder6bit(
 // I/O ports
 input wire [5:0] x,
 input wire [5:0] y,
 input wire sel,
 output wire overflow,
 output wire c_out,
 output wire [5:0] sum
 );
 wire [5:0] sel_y;
 wire [5:0] c_outs;
 //body
 assign sel_y[0] = y[0] ^ sel;
 assign sel_y[1] = y[1] ^ sel;
 assign sel_y[2] = y[2] ^ sel;
 assign sel_y[3] = y[3] ^ sel;
 assign sel_y[4] = y[4] ^ sel;
 assign sel_y[5] = y[5] ^ sel;

 FullAdder fa0 (.a(x[0]), .b(sel_y[0]), .cin(sel), .s(sum[0]), .cout(c_outs[0]));
 FullAdder fa1 (.a(x[1]), .b(sel_y[1]), .cin(c_outs[0]), .s(sum[1]), .cout(c_outs[1]));
 FullAdder fa2 (.a(x[2]), .b(sel_y[2]), .cin(c_outs[1]), .s(sum[2]), .cout(c_outs[2]));
 FullAdder fa3 (.a(x[3]), .b(sel_y[3]), .cin(c_outs[2]), .s(sum[3]), .cout(c_outs[3]));
 FullAdder fa4 (.a(x[4]), .b(sel_y[4]), .cin(c_outs[3]), .s(sum[4]), .cout(c_outs[4]));
 FullAdder fa5 (.a(x[5]), .b(sel_y[5]), .cin(c_outs[4]), .s(sum[5]), .cout(c_outs[5]));

 assign c_out = c_outs[5];
 assign overflow = c_outs[5]^c_outs[4];

endmodule  // ending module
```

Figure 2.0

Testbench for adder6bit.v(adder6bit_testbench.v)

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Create Date: 10/19/2021 04:19:50 PM
// Module Name: add_sub_testbench_6bit
//////////////////////////////////////////////////////////////////////////////////
module add_sub_testbench_6bit;
    // signal declaration
    reg [5:0] xin, yin;
    reg selin;
    wire [5:0] test_out;
    wire overflowout, carryout;
    // instantiate the circuit under test
    add_sub_6_bit uut(.x(xin), .y(yin), .sel(selin),
    .overflow(overflowout), .c_out(carryout), .sum(test_out));
    // test vector generator
    initial
        begin
        // test 1
        xin = 6'b000001;
        yin = 6'b000000;
        selin = 1'b0;
        #200
        //test 2
        xin = 6'b111111;
        yin = 6'b000010;
        selin = 1'b1;
        #200
        // test 3
        xin = 6'b000100;
        yin = 6'b000001;
        selin = 1'b0;
        #200
        // test 4
        xin = 6'b000110;
        yin = 6'b000110;
        selin = 1'b1;
        #200
        //test 5
        xin = 6'b001000;
        yin = 6'b000111;
        selin = 1'b1;
        #200
        //test 6
        xin = 6'b000111;
        yin = 6'b000111;
        selin = 1'b1;
        #200
        // stop simulation
        $stop;
        end
endmodule // ending module
```

Figure2.1

Waveform for adder6bit.v

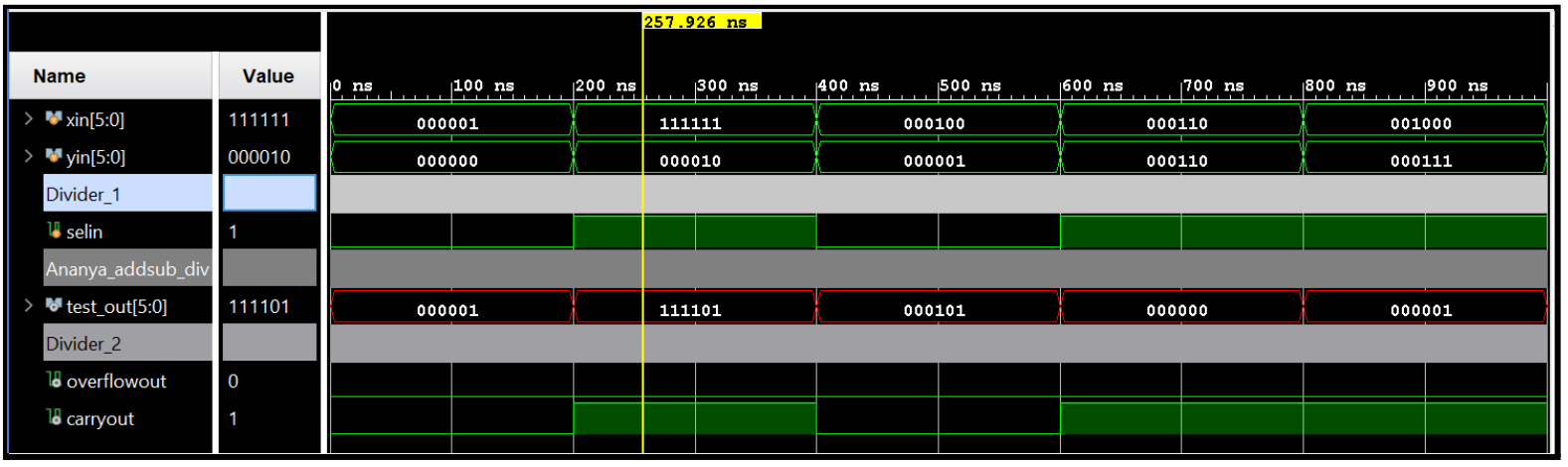

Figure 2.2 : adder6bit.v waveform - radix set to binary

(Note : This laptop belonged to my brother ''Parag Garg', which is why all directories shows his name instead of mine.)

6. **topALU.v :** This section contains all the combined functions mentioned above in our MINI-ALU as defined in the description of assignment-1. And this is the main module we have worked upon and in this module we are calling all our other modules created recently as well as from the previous labs.

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

Test vectors used are :

```
//test 1
fxn=000;
A=6'b000000;
B=6'b000010;

#200;

//test 2
fxn=001;
A=6'b000001;
B=6'b000110;
```

```
//test 3
fxn=010;
A=6'b000011;
B=6'b001001;

#200;

//test 4
fxn=011;
A=6'b000100;
B=6'b110000;

#200;
```

```
//test 5
fxn=101;
A=6'b011110;
B=6'b100100;

#200;

//test 6
fxn=110;
A=6'b001010;
B=6'b001100;

#200;
```

```
//test 7
fxn=111;
A=6'b000010;
B=6'b000110;

#200;
```

Code :

(Note : This laptop belonged to my brother ''Parag Garg', which is why all directories shows his name instead of mine.)

topALU.v

C:/Users/PARAG GARG/Assignment1.xsim/Assignment1.xsim.srcs/sources_1/new/topALU.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Create Date: 28.10.2021 20:57:20
4  // Module Name: topALU
5  //////////////////////////////////////////////////////////////////////////////////
6  //THIS MODULE IMPLEMENTS THE ALU BY CALLING OTHER MODULES FOR EACH FUNCTION
7  module topALU(
8      input [5:0] A,            // 6-bit input A
9      input [5:0] B,           // 6-bit input B
10     input [2:0] fxn,        // 3-bit input fxn
11     output reg [5:0] X,     // 6-bit output X
12     output reg O,
13     output reg c
14     );
15     wire overflow1, carryout1,overflow2, carryout2;
16     wire [5:0]w0,w1,w2,w3,w4,w5,w6,w7;
17     always @ *
18     if(fxn==3'b000)
19     begin
20        X=w0;O=0; c=0;
21        end
22     else if(fxn==3'b001)
23     begin
24        X=w1;O=0; c=0;
25        end
26     else if(fxn==3'b010)
27     begin
28        X=w2;O=0; c=0;
29        end
30     else if(fxn==3'b011)
31     begin
32        X=w3;O=0; c=0;
33        end
34     else if(fxn==3'b100)
35     begin
36        X=w4;O=0; c=0;
37        end
38     else if(fxn==3'b101)
39     begin
40        X=w5; O=0; c=0;
41        end
42     else if(fxn==3'b110)
43     begin
44        X=w6; O=overflow1;
45        c=carryout1;
46        end
47     else if(fxn==3'b111)
48      begin
49        X=w7;O=overflow2;
50        c=carryout2;
51     end
52 // instantiating the modules
53 equal eq1(A,w0);       // X = A
54 equal eq2 (B,w1);      // X = B
55 inverting n1(A,w2);     // X = -A
56 inverting n2 (B,w3);    // X = -B
57 eq4 compare(A,B,w4);    // X = A<B
58 XNOR x1 (A,B,w5);      // X = A xnor B
59 adder6bit a1 (A,B,0,overflow1,carryout1,w6);    // X = A+B
60 adder6bit a2 (A,B,1,overflow2,carryout2,w7);    // X = A-B
61 endmodule // ending module
```

Figure 2.3 : Code for topALU.v

Waveform:

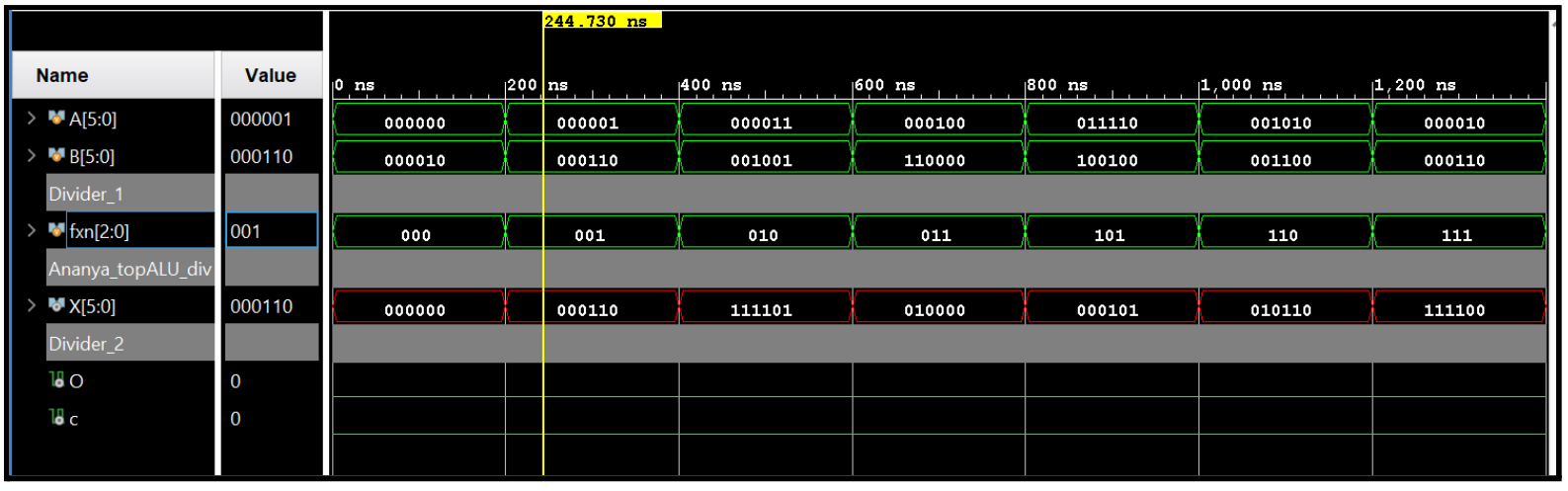

Figure 2.4 : Waveform for topALU.v - radix set to binary

(Note : This laptop belonged to my brother ''Parag Garg', which is why all directories shows his name instead of mine.)

3. Sources : The project created has the following components:

The design source consists of topALU (MINI-ALU) module which contains the properties of other instantiations of modules. The expanded design sources are the implemented modules used in the modules are given as under :

- eq1 (equal.v)
- eq2 (equal.v)
- n1 (negate.v)
- n2 (negate.v)
- eq4(acompareb.v)
- x1 (XNOR.v)
- a1 (adder6bit.v)
- a2 (adder6bit.v)

The simulation source consists of a topALU_testbench module which further contains the testbenches of other modules.

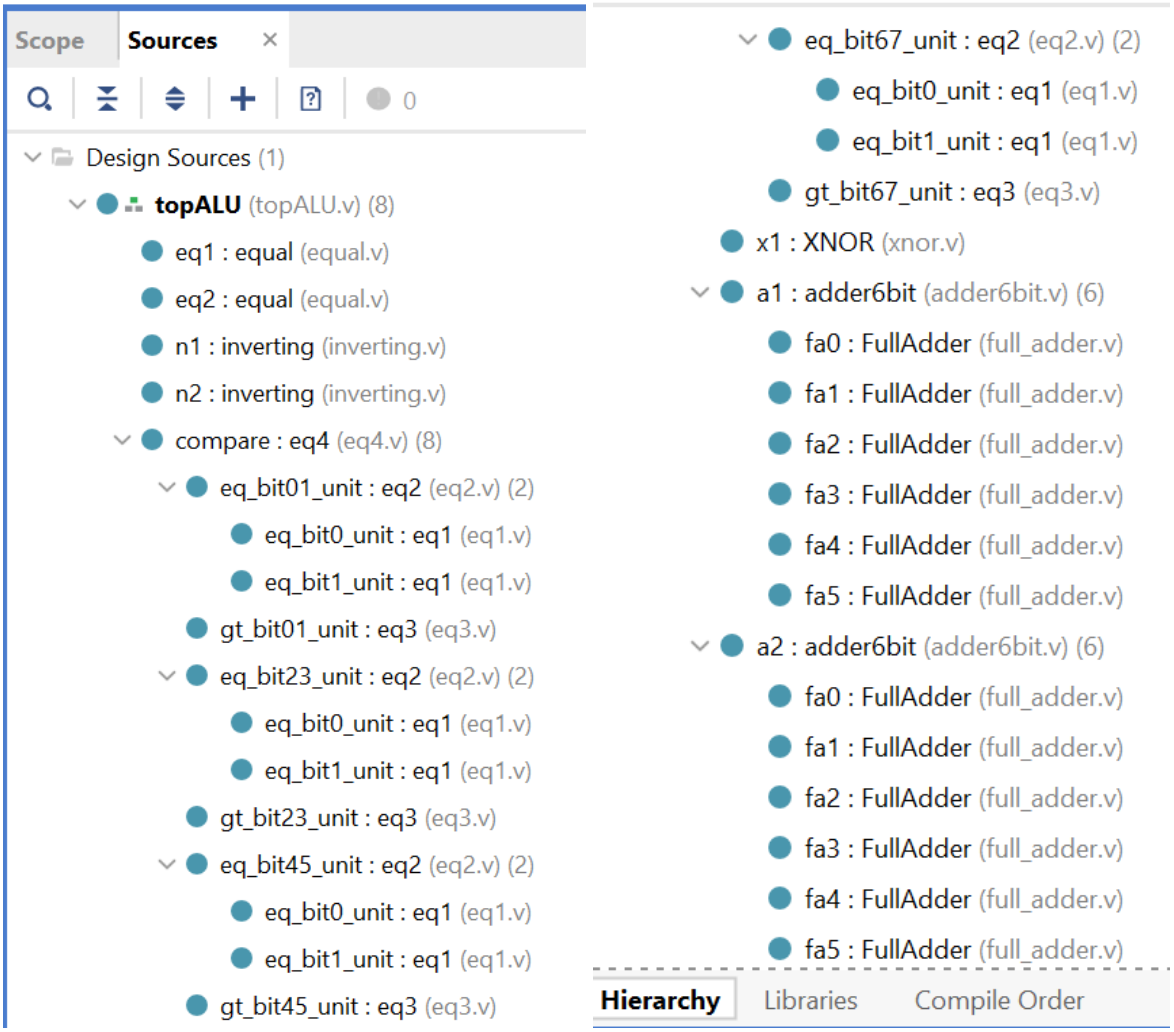

Figure 2.5 : Project - Design Sources

(Note : This laptop belonged to my brother ‘’Parag Garg’, which is why all directories shows his name instead of mine.)

Following is the file directory used :

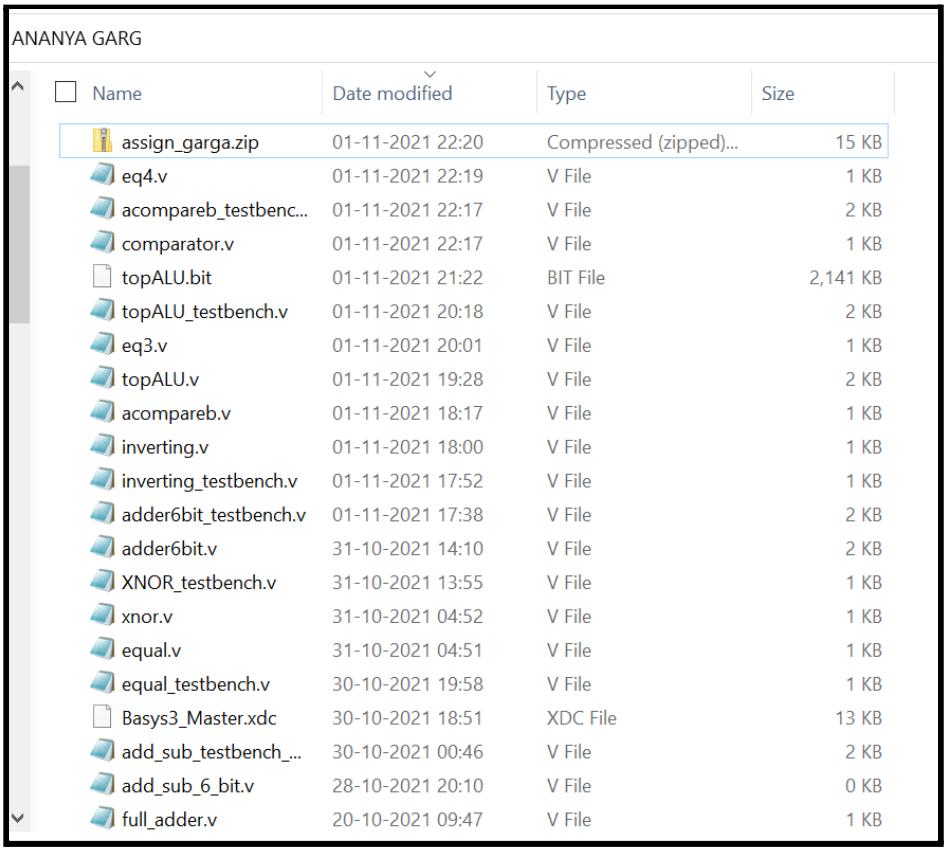

Figure 2.6 : File directory

(Note : This laptop belonged to my brother ''Parag Garg', which is why all directories shows his name instead of mine.)

The following images constitute the constraints for the Basys3 Board implementation. The implementation is further explained in the demo section.

```
## Switches
set_property PACKAGE_PIN V17 [get_ports {A[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A[0]}]
set_property PACKAGE_PIN V16 [get_ports {A[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A[1]}]
set_property PACKAGE_PIN W16 [get_ports {A[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A[2]}]
set_property PACKAGE_PIN W17 [get_ports {A[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A[3]}]
set_property PACKAGE_PIN W15 [get_ports {A[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A[4]}]
set_property PACKAGE_PIN V15 [get_ports {A[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A[5]}]
set_property PACKAGE_PIN W14 [get_ports {B[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B[0]}]
set_property PACKAGE_PIN W13 [get_ports {B[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B[1]}]
set_property PACKAGE_PIN V2 [get_ports {B[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B[2]}]
set_property PACKAGE_PIN T3 [get_ports {B[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B[3]}]
set_property PACKAGE_PIN T2 [get_ports {B[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B[4]}]
set_property PACKAGE_PIN R3 [get_ports {B[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B[5]}]
set_property PACKAGE_PIN W2 [get_ports {fxn[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {fxn[0]}]
set_property PACKAGE_PIN U1 [get_ports {fxn[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {fxn[1]}]
set_property PACKAGE_PIN T1 [get_ports {fxn[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {fxn[2]}]
#set_property PACKAGE_PIN R2 [get_ports {sw[15]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {sw[15]}]


## LEDs
set_property PACKAGE_PIN U16 [get_ports {X[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {X[0]}]
set_property PACKAGE_PIN E19 [get_ports {X[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {X[1]}]
set_property PACKAGE_PIN U19 [get_ports {X[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {X[2]}]
set_property PACKAGE_PIN V19 [get_ports {X[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {X[3]}]
set_property PACKAGE_PIN W18 [get_ports {X[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {X[4]}]
set_property PACKAGE_PIN U15 [get_ports {X[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {X[5]}]
set_property PACKAGE_PIN U14 [get_ports {O}]
    set_property IOSTANDARD LVCMOS33 [get_ports {O}]
set_property PACKAGE_PIN V14 [get_ports {c}]
    set_property IOSTANDARD LVCMOS33 [get_ports {c}]
```

Figure 2.7

(Note : This laptop belonged to my brother ''Parag Garg', which is why all directories shows his name instead of mine.)

4. Schematics Generated :

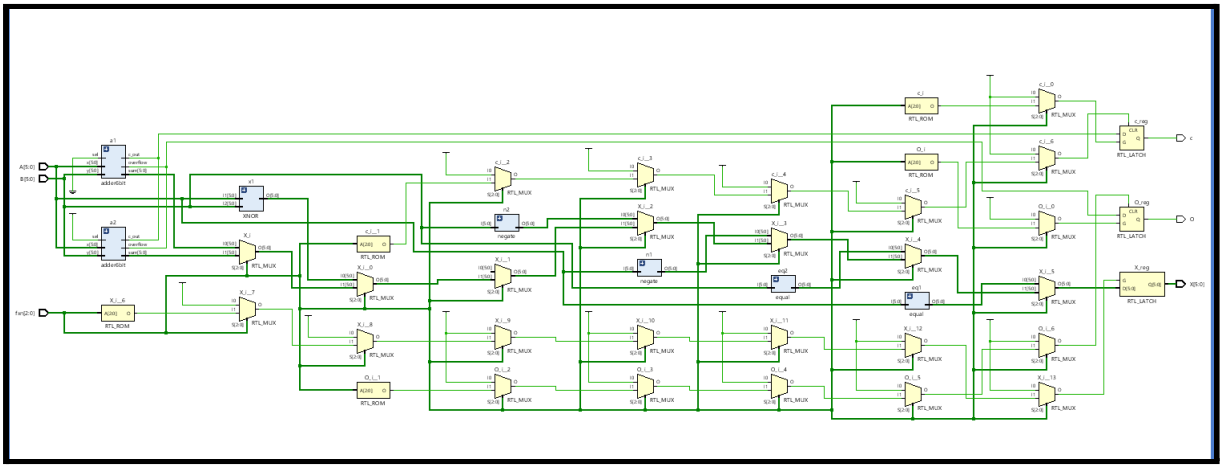

Figure 2.8 : Schematic representation

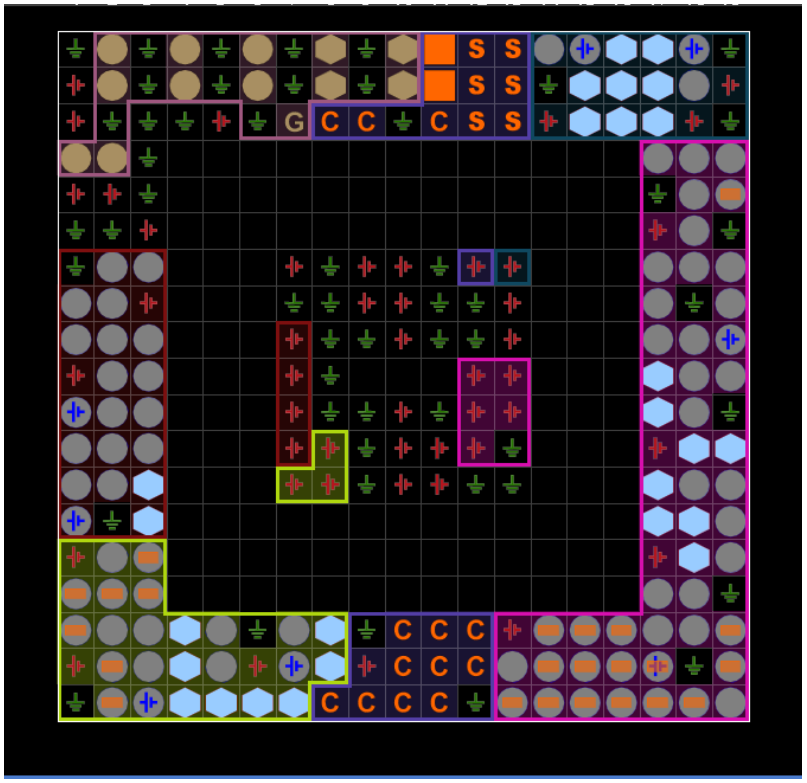

Figure 2.9 : Synthesised Design

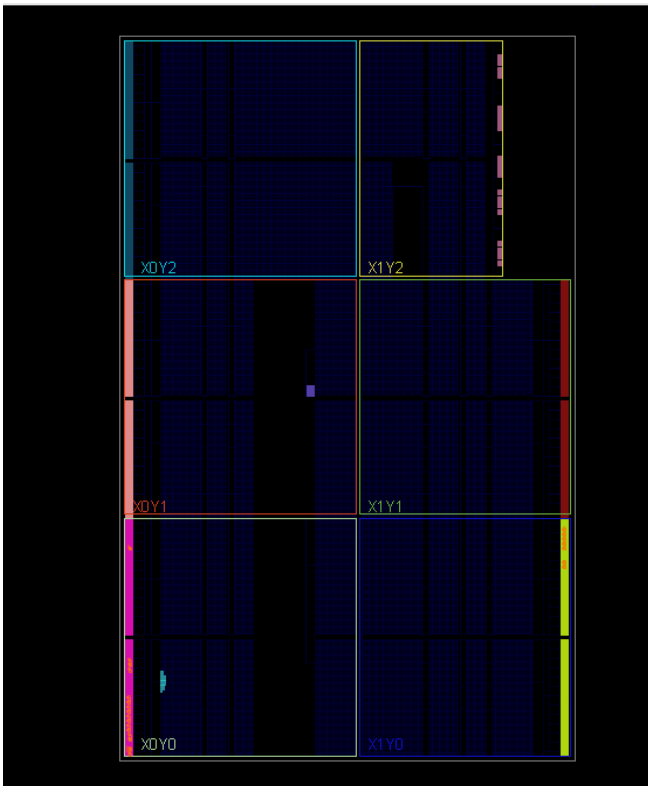

Figure 3.0 : Implemented Design

(Note : This laptop belonged to my brother ‘’Parag Garg’, which is why all directories shows his name instead of mine.)

5. Demonstration :

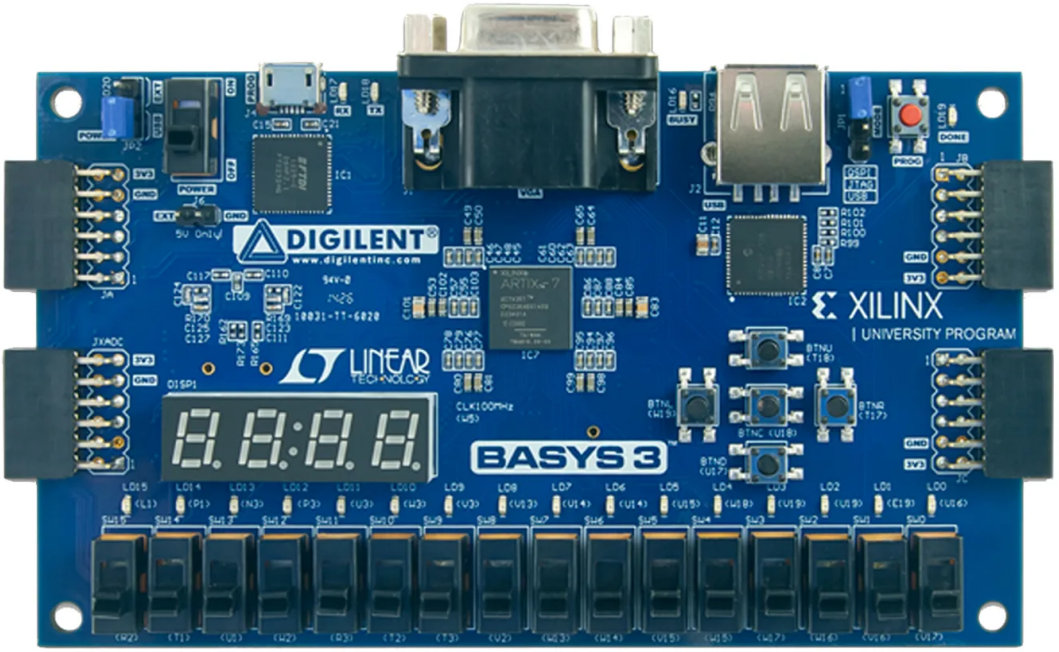

Figure 3.1 : Basys3 board

Target language - Verilog

The basys board implementation which was shown in the sources is further elaborated as follows:

#Switches

V17	A[0]	V2	B[2]
V16	A[1]	T3	B[3]
W16	A[2]	T2	B[4]
W17	A[3]	R3	B[5]
W15	A[4]	W2	fxn[0]
V14	A[5]	U1	fxn[1]
W14	B[0]	T1	fxn[2]
W13	B[1]		

#LED’s

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

OFF led = 0 bit whereas ON led = 1 bit.

(Note : This laptop belonged to my brother ‘’Parag Garg’, which is why all directories shows his name instead of mine.)

Basys3 board test cases:

Case 0 : fxn = b’000 Input A = 000001
B = 111111

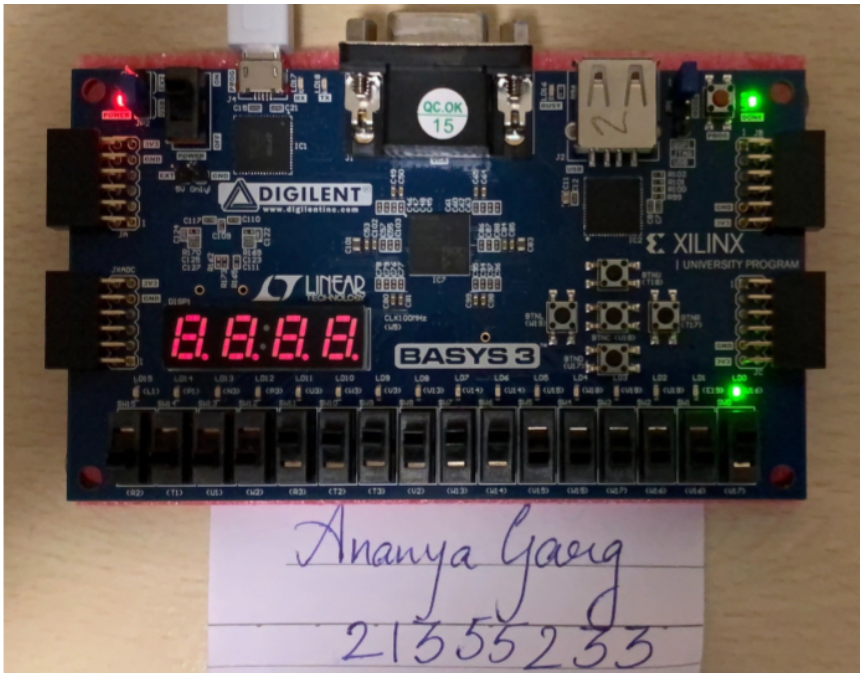

Figure 3.2 : Output A = 000001

Case 1 : fxn = b’001 Input A = 100001
B = 000110

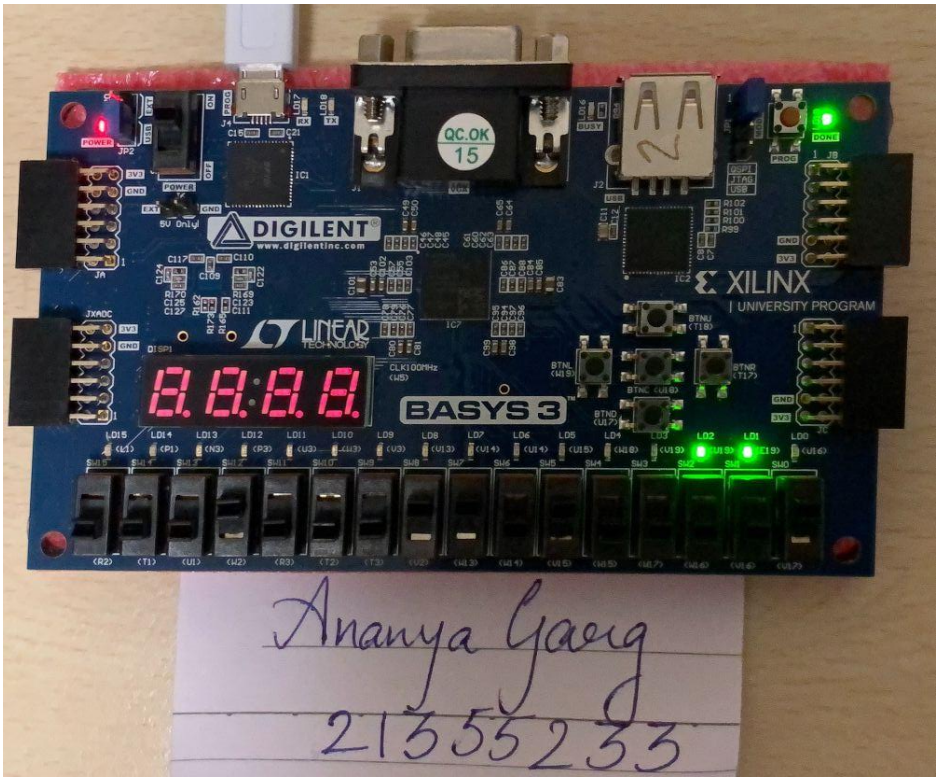

Figure 3.3 : Output B = 000110

(Note : This laptop belonged to my brother ‘’Parag Garg’, which is why all directories shows his name instead of mine.)

Case 2 : fxn = b’010 Input A = 000010
B = 000110

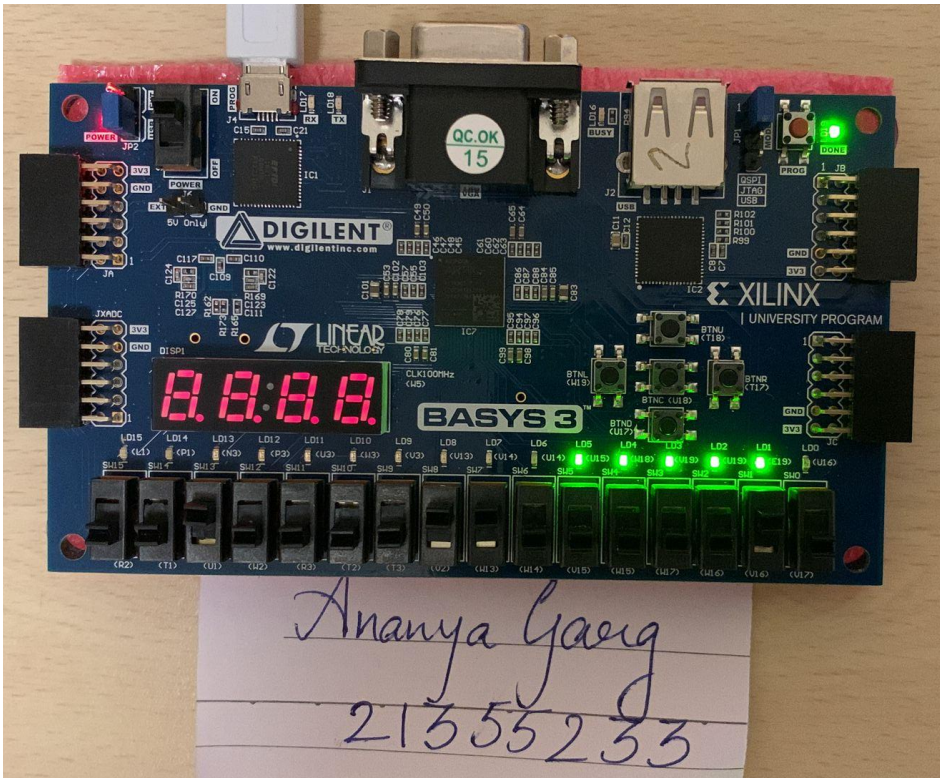

Figure 3.4 : Output -A = 111110

Case 3 : fxn = b’011 Input A = 000010
B = 101111

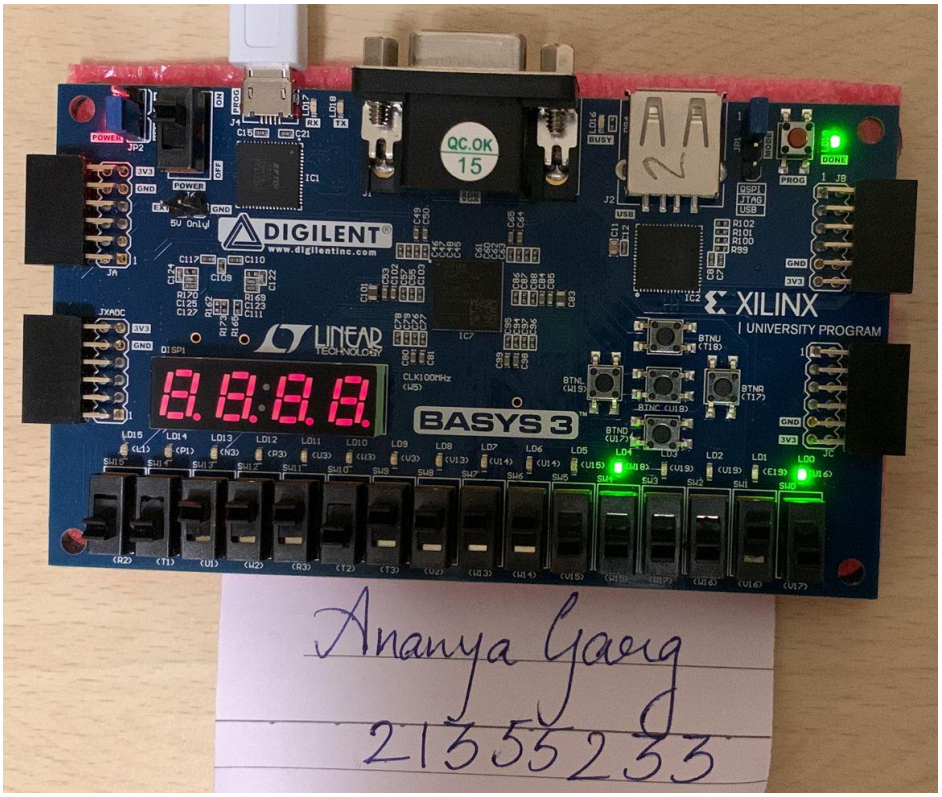

Figure 3.5 : Output -B = 010001

Case 4 : fxn = b’101 Input A= 011110
B = 100100

(Note : This laptop belonged to my brother ''Parag Garg', which is why all directories shows his name instead of mine.)

Figure 3.6 : Output A xnor B = 000101

Case 5 : fxn = b'110 Input A = 001010
B = 001100

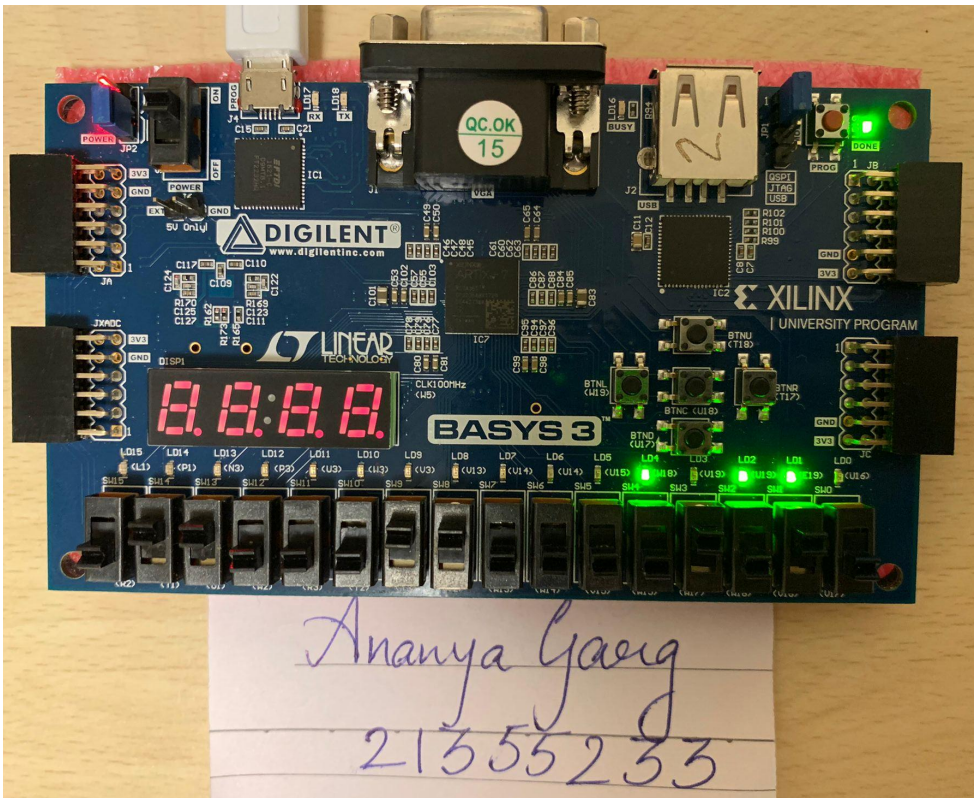

Figure 3.7 : Output A+B = 010110

Case 6 : fxn = b'111 Input A = 000010
B = 000110

(Note : This laptop belonged to my brother ‘’Parag Garg’, which is why all directories shows his name instead of mine.)

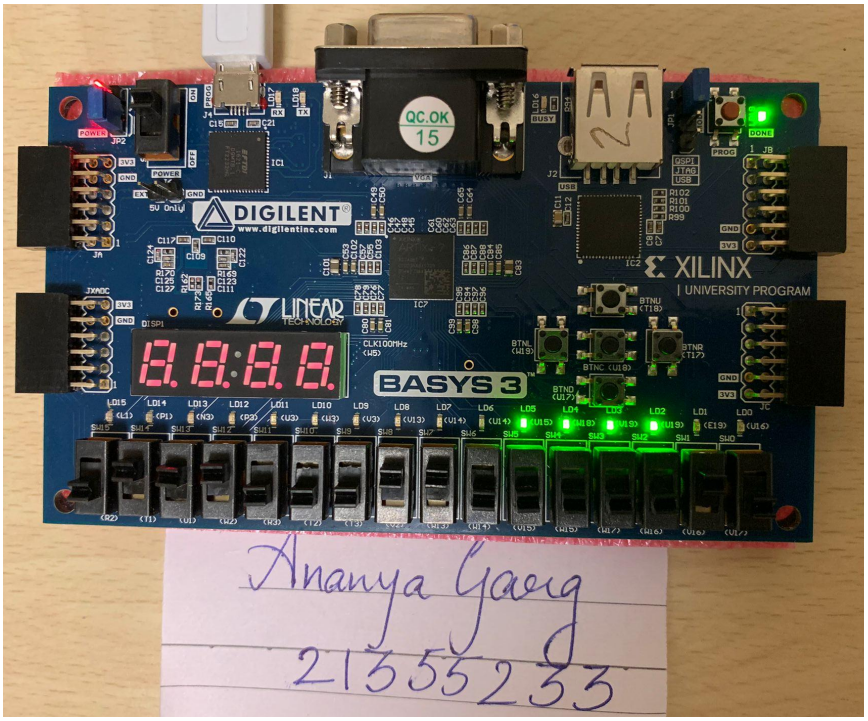

Figure 3.8 : Output A-B = 111100

6. Conclusion : I have learned Vivado to use and implement digital combinational circuits with the help of all the previous modules such as eq2,eq3 and recent modules such as adder6bit,xnor etc and mainly through the main module topALU (MINI-ALU).