Lab 03

INTRODUCTION TO XILINX ISE, Spartan 6 BOARD AND IMPLEMENTATION OF RIPPLE CARRY ADDER AND FULL SUBTRACTOR

Spring 2025

Submitted by: **Mohsin Sajjad**

Registration No: **22pwsce2149**

Class Section: **A**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Student Signature: _____

Submitted to:

Engr. Faheem Jan

Month Day, Year (16 02, 2025)

Department of Computer Systems Engineering

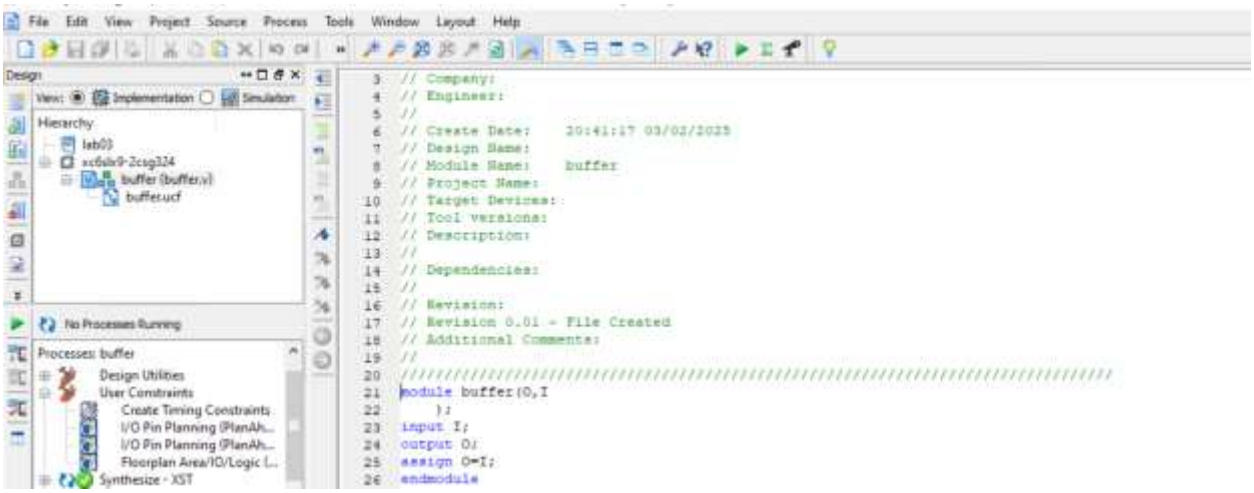
University of Engineering and Technology, Peshawar

Objectives:

- Introduction to FPGA
- Introuction Xilinx ISE

LAB TASKS:

1-Implement buffer ON the Kit and attach the snapshot.



Output on Xilinx:

2-Implement AND/OR/XOR gate on the Kit and attach the snapshot.

And Gate:

```
18  // Additional Comments:
19  //
20  //////////////////////////////////
21  module andgate(A, B, O);
22  input A, B;
23  output O;
24  assign O = A & B;
25  endmodule
```

Output:

OR Gate:

```
20  //////////////////////////////////////////////////////////////////
21  module orgate(A, B, O);
22  input A, B;
23  output O;
24  assign O = A | B;
25  endmodule
```

Output:

XOR Gate:

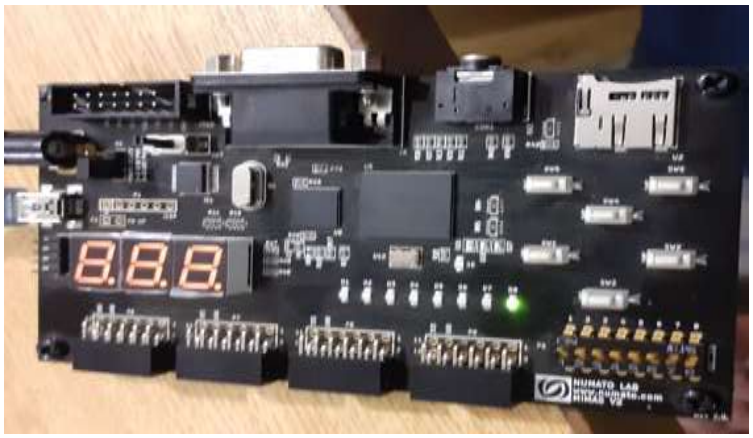
```
20  //////////////////////////////////////////////
21  module xorgate(A, B, O);
22  input A, B;
23  output O;
24  assign O = A ^ B;
25  endmodule
```

Output:

3-Impement OR gate using NAND gate

```
20  //////////////////////////////////////////////////////////////////
21  module orgate(A, B, O);
22  input A, B;
23  output O;
24  wire nandout1, nandout2;
25  nand n1(nandout1, A);
26  nand n2(nandout2, B);
27  nand n3(O, nandout1, nandout2);
28  endmodule
```

Output Xilinx:



4- Implement Lab1 and Lab1 ON the Kit and attach snapshot.

Equation:

```
module equ(Z,x1,x2,x3,x4,x5);
input x1,x2,x3,x4,x5;
output Z;
wire y1;
wire y2;
and n1(y1,x1,x2);
and n2(y2,x3,x4,x5);
nor n3(Z,y1,y2);
endmodule

module equ_tb();
reg x1,x2,x3,x4,x5;
wire Z;
equ ins(Z,x1,x2,x3,x4,x5);
initial begin
x1=0;
x2=0;
x3=0;
x4=0;
x5=0;
#20 $display("x1 = %b",x1, "x2 = %b",x2,"x3 = %b",x3,"x4 = %b",x4,"x5 = %b",x5,"Z = %b",Z);

x1=0;
x2=0;
x3=0;
```

```
x1=0;
x2=0;
x3=0;
x4=0;
x5=1;
#20 $display("x1 = %b",x1, "x2 = %b",x2,"x3 = %b",x3,"x4 = %b",x4,"x5 = %b",x5,"Z = %b",Z);

x1=0;
x2=0;
x3=0;
x4=1;
x5=0;
#20 $display("x1 = %b",x1, "x2 = %b",x2,"x3 = %b",x3,"x4 = %b",x4,"x5 = %b",x5,"Z = %b",Z);

x1=0;
x2=0;
x3=0;
x4=1;
x5=1;
#20 $display("x1 = %b",x1, "x2 = %b",x2,"x3 = %b",x3,"x4 = %b",x4,"x5 = %b",x5,"Z = %b",Z);
end
endmodule
```

Xilinx output:

Logic Circuit:

```
module logic_circuit(output Y, input A, B, C);
    wire A_not, X1, X2, X2_not, Y1, Y2, Y1_not;

    // Inverters
    not n1(A_not, A);
    not n2(X1_not, X1);

    // AND Gates
    nand a2(X1,B, C);
    and a3(X2,X1,B);
    and a1(Y1, A_not, X2);
    // XOR Gate
    xor x1(Y2, X2, X1_not);

    // Inverter for Y1
    not n3(Y1_not, Y1);

    // OR Gate
    or o1(Y, Y1_not, Y2);
endmodule

module logic_circuit_tb();
    reg A, B, C;
    wire Y;
    logic_circuit uut(Y, A, B, C);
```

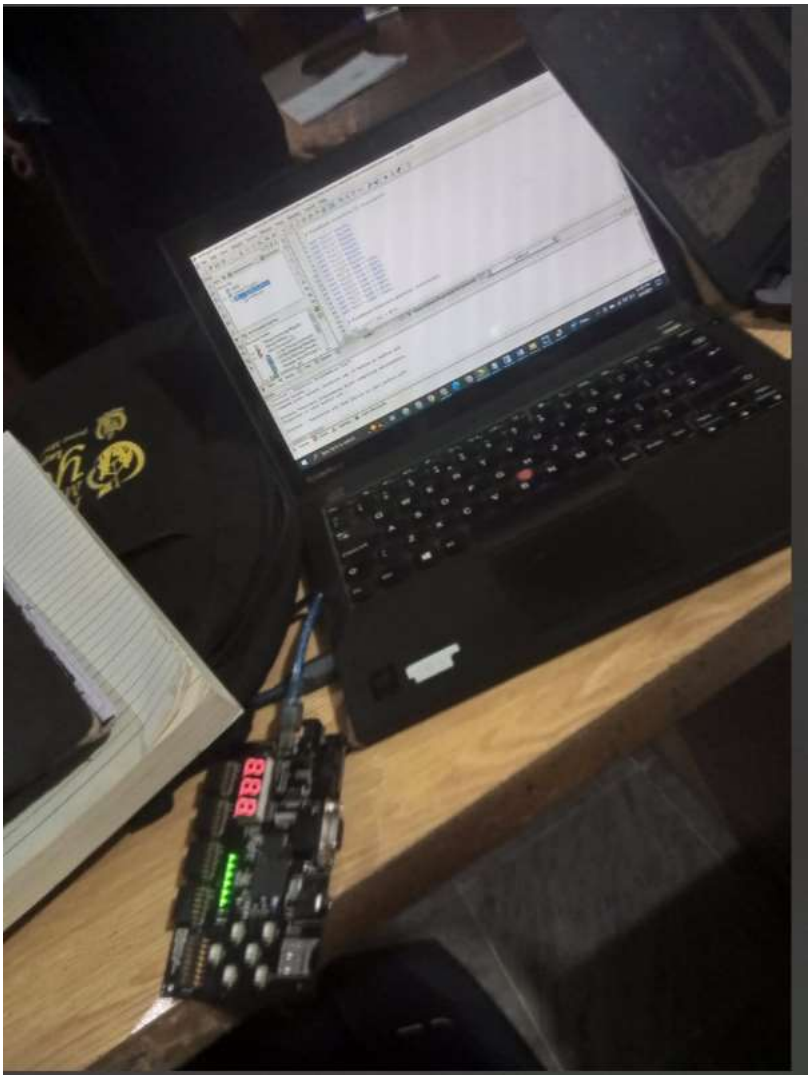
```
    logic_circuit uut(Y, A, B, C);

    initial begin
        // Display header
        $display("A  B  C  |  Y");
        $display("----------------");

        // Test all possible input combinations
        A = 0; B = 0; C = 0; #10; $display("%b  %b  %b  |  %b", A, B, C, Y);
        A = 0; B = 0; C = 1; #10; $display("%b  %b  %b  |  %b", A, B, C, Y);
        A = 0; B = 1; C = 0; #10; $display("%b  %b  %b  |  %b", A, B, C, Y);
        A = 0; B = 1; C = 1; #10; $display("%b  %b  %b  |  %b", A, B, C, Y);
        A = 1; B = 0; C = 0; #10; $display("%b  %b  %b  |  %b", A, B, C, Y);
        A = 1; B = 0; C = 1; #10; $display("%b  %b  %b  |  %b", A, B, C, Y);
        A = 1; B = 1; C = 0; #10; $display("%b  %b  %b  |  %b", A, B, C, Y);
        A = 1; B = 1; C = 1; #10; $display("%b  %b  %b  |  %b", A, B, C, Y);

        $stop;
    end
endmodule
```

Xilinx Output:



Ripple Carry Adder:

```
module Sum(S,A,B);
    input A,B;
    output S;
    xor x(S,A,B);
endmodule

module Carry(C,A,B);
    input A,B;
    output C;
    and a(C,A,B);
endmodule

module HA(S,C,A,B);
    input A,B;
    output S,C;
    Sum s1(S,A,B);
    Carry c1(C,A,B);
endmodule

module FA(S,C,A,B,Cin);
    input A,B,Cin;
    output S,C;
    wire S1,C1,C2;
```

```
25      HA h1(S1,C1,A,B);
26      HA h2(S,C2,Cin,S1);
27      or o(C,C1,C2);
28  endmodule
29
30  module RCA(S,Cout,A,B,Cin);
31      input [3:0] A,B;
32      input Cin;
33      output [3:0] S;
34      output Cout;
35      wire [2:0] C;
36
37      FA f1(S[0],C[0],A[0],B[0],Cin);
38      FA f2(S[1],C[1],A[1],B[1],C[0]);  // Unique instance name
39      FA f3(S[2],C[2],A[2],B[2],C[1]);  // Unique instance name
40      FA f4(S[3],Cout,A[3],B[3],C[2]);  // Unique instance name
41  endmodule
42
43  module RCA_tb();
44      reg [3:0] A,B;
45      reg Cin;
46      wire [3:0] S;
47      wire Cout;
48
```

```
48
49      RCA rr(S,Cout,A,B,Cin);
50
51      initial begin
52          A = 4'b0000; B = 4'b0001; Cin = 1'b0;
53          #20;
54          A = 4'b0101; B = 4'b0011; Cin = 1'b0;
55      end
56
57      initial
58          $monitor("A=%b B=%b Sum=%b Cout=%b", A, B, S, Cout);
59  endmodule
```

5- Implement Full Subtractor on the board and verify the truth table

```
21  module Difference (A, B, D);
22      input A, B;
23      output D;
24      xor x(D, A, B);
25  endmodule
26
27  module Borrow (A, B, Borr);
28      input A, B;
29      output Borr;
30      wire nA;
31      not n(nA, A);
32      and a(Borr, nA, B);
33  endmodule
34
35  module Hs (A, B, D, Borr);
36      input A, B;
37      output D, Borr;
38      Difference d1(A, B, D);
39      Borrow b1(A, B, Borr);
40  endmodule
41
42  module Fs (A, B, Bin, D, Borr);
43      input A, B, Bin;
44      output D, Borr;
45      wire b1, b2, d1;
```

```
45      wire b1, b2, d1;
46      Hs h1(A, B, d1, b1);
47      Hs h2(d1, Bin, D, b2);
48      or o(Borr, b1, b2);
49  endmodule
```

Output Xilinx: