University of Miami

EEN 414 Computer Organization and Design

EXPERIMENT #. EXPERIMENT NAME

Submitted by: Brandon Rubio, Isabela Bandrich, Rainier Young, Nikeem Dunkelly-Allen
On: 9/30/22
to : Dr. Onur Tigli

Objective

In this experiment, you are required to design and verify the correct operation of an adder/subtractor unit using structural and dataflow modeling techniques in Verilog. Coding, verification, simulation, synthesis and timing analysis will be carried out using Xilinx ISE tools. Verified designs will be implemented on Digilent's NEXSYS-4 FPGA boards for demonstration of correct operation.

Verilog HDL and Testbench Code

Half Adder:

```
module Half_Adder(
    input in_0,
    input in_1,
    output sum,
    output c_out
    );

    xor(sum, in_0, in_1);
    and(c_out, in_0, in_1);

endmodule
```

```
module tb_Half_Adder;

   // Inputs
   reg in_0;
   reg in_1;

   // Outputs
   wire sum;
   wire c_out;

   // Instantiate the Unit Under Test (UUT)
   Half_Adder uut (
      .in_0(in_0),
      .in_1(in_1),
      .sum(sum),
      .c_out(c_out)
   );

   initial begin
      // Initialize Inputs
      in_0 = 0;
      in_1 = 0;
      // Wait 100 ns for global reset to finish
      #100;

      // Add stimulus here
     in_0 = 0;
     in_1 = 0;
     #100;

     in_0 = 1;
     in_1 = 0;
     #100;

     in_0 = 0;
     in_1 = 1;
     #100;

     in_0 = 1;
     in_1 = 1;
     #100;

   end

endmodule
```

Full Adder:

```
module FullAdder1(
    input a,
    input b,
    input cin,
    output cout,
    output sum
    );

    wire HA1sum,HA1cout,HA2cout;
    HalfAdder1 HA1(a,b,HA1sum,HA1cout);
    HalfAdder1 HA2(HA1sum,cin,sum,HA2cout);
    or(cout,HA2cout,HA1cout);


endmodule
```

```
24
25  module tb_FullAdder1;
26
27     // Inputs
28     reg a;
29     reg b;
30     reg cin;
31
32     // Outputs
33     wire cout;
34     wire sum;
35
36     // Instantiate the Unit Under Test (UUT)
37     FullAdder1 uut (
38        .a(a),
39        .b(b),
40        .cin(cin),
41        .cout(cout),
42        .sum(sum)
43     );
44     reg[1:0] i;
45     reg[1:0] j;
46     reg[1:0] n;
47     initial begin
48        // Initialize Inputs
49        a = 0;
50        b = 0;
51        cin = 0;
52
53        // Wait 100 ns for global reset to finish
54        #100;
55
56        // Add stimulus here
57        for (i = 0; i < 2; i = i + 1) begin
58            cin = i;
59            for (j = 0; j < 2; j = j + 1) begin
60                 a = j;
61                 for (n = 0; n < 2; n = n +1)begin
62                    b = n;
63                    #10;
64                 end
65            end
66        end
67     end
68
69  endmodule
```

4-bit Adder Subtractor:

```
module Adder_Subtractor_4bit(
    input [3:0] A,
    input [3:0] B,
    input E,
    output [3:0] S,
    output Cout
    );
    wire b0, b1, b2, b3;
    wire c0, c1, c2;

    xor(b0, B[0], E);
    xor(b1, B[1], E);
    xor(b2, B[2], E);
    xor(b3, B[3], E);

    Full_Adder fa1(A[0], b0, E, S[0], c0);
    Full_Adder fa2(A[1], b1, c0, S[1], c1);
    Full_Adder fa3(A[2], b2, c1, S[2], c2);
    Full_Adder fa4(A[3], b3, c2, S[3], Cout);

endmodule
```

```
module tb_Adder_Subtractor_4bit;

    // Inputs
    reg [3:0] A;
    reg [3:0] B;
    reg E;

    // Outputs
    wire [3:0] S;
    wire Cout;

    // Instantiate the Unit Under Test (UUT)
    Adder_Subtractor_4bit uut (
        .A(A),
        .B(B),
        .E(E),
        .S(S),
        .Cout(Cout)
    );
    reg [1:0] i;
    reg [4:0] j;
    reg [4:0] k;

    initial begin
        // Initialize Inputs
        A = 0;
        B = 0;
        E = 0;
        // Wait 100 ns for global reset to finish
        #100;

        for(i=0;i<2;i=i+1) begin
            E = i;
            for(j=0;j<16;j=j+1) begin
                A = j;
                for(k=0;k<16;k=k+1) begin
                    B = k;
                    #10;
                end
            end
        end
    end

endmodule
```

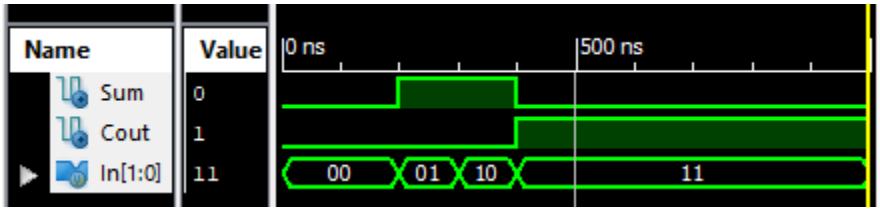
Seven Segment Display:

```
module Decoder(
    input [3:0] In,
    output [6:0] Out
    );
//seven segment display
   reg [6:0] temp;
   always @(In) begin
   case(In)
   0: temp = 7'b1111110;
   1: temp = 7'b0110000;
   2: temp = 7'b1101101;
   3: temp = 7'b1111001;
   4: temp = 7'b0110011;
   5: temp = 7'b1011011;
   6: temp = 7'b1011111;
   7: temp = 7'b1110000;
   8: temp = 7'b1111111;
   9: temp = 7'b1111011;
   10: temp = 7'b1110111;
   11: temp = 7'b0011111;
   12: temp = 7'b1001110;
   13: temp = 7'b0111101;
   14: temp = 7'b1001111;
   15: temp = 7'b1000111;
   endcase
   end
   assign Out = temp;

endmodule
```

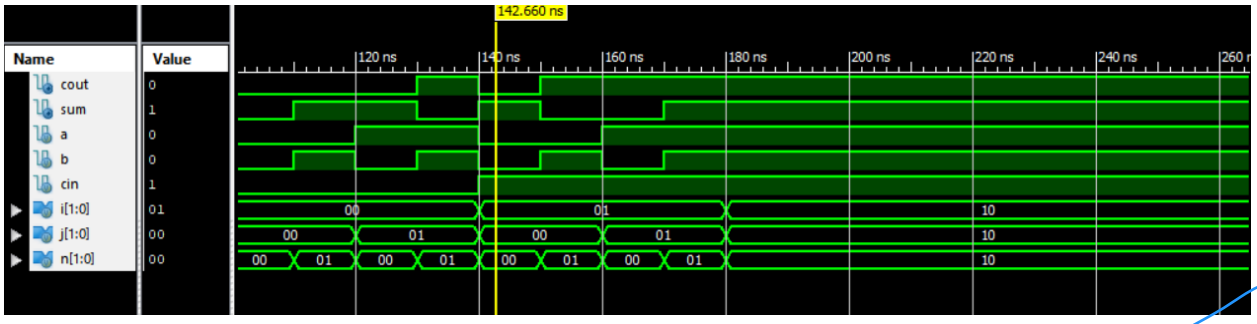
```
module tb_Decoder;

    // Inputs
    reg [3:0] In;

    // Outputs
    wire [6:0] Out;

    // Instantiate the Unit Under Test (UUT)
    Decoder uut (
        .In(In),
        .Out(Out)
    );
    reg [3:0] i;

    initial begin
        // Initialize Inputs
        In = 0;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
        for(i=0;i<16;i=i+1) begin
            In = i;
            #100;
        end

    end

endmodule
```

Simulation Results

Half Adder:

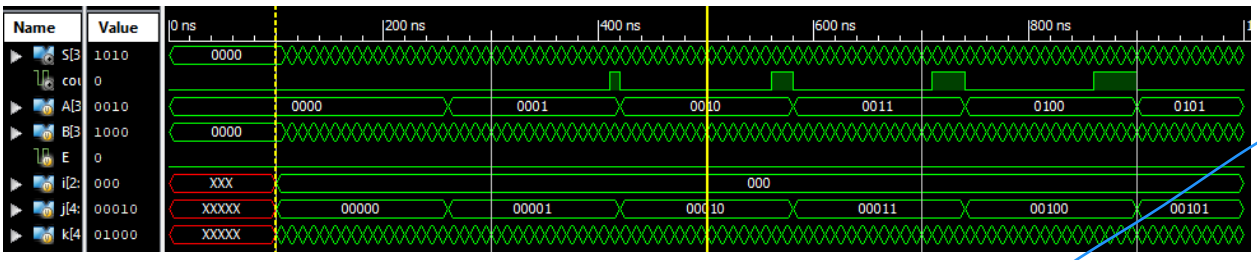


Full Adder:

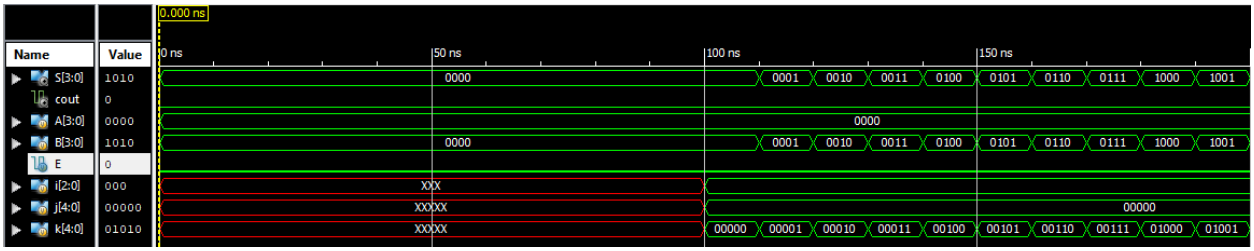


4-bit Adder Subtractor:

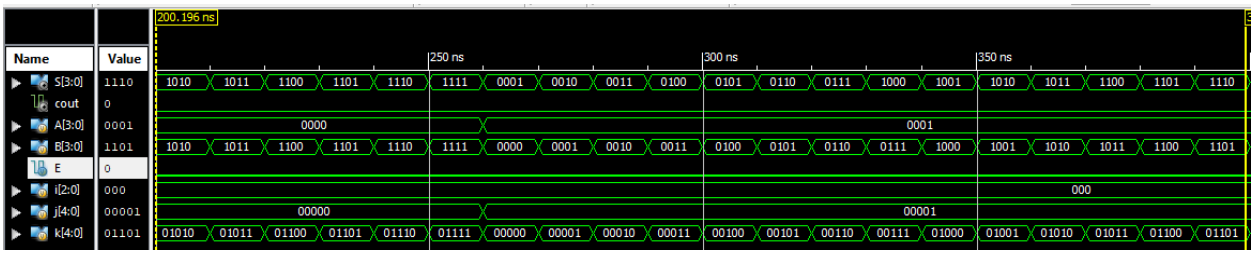
(Overview)



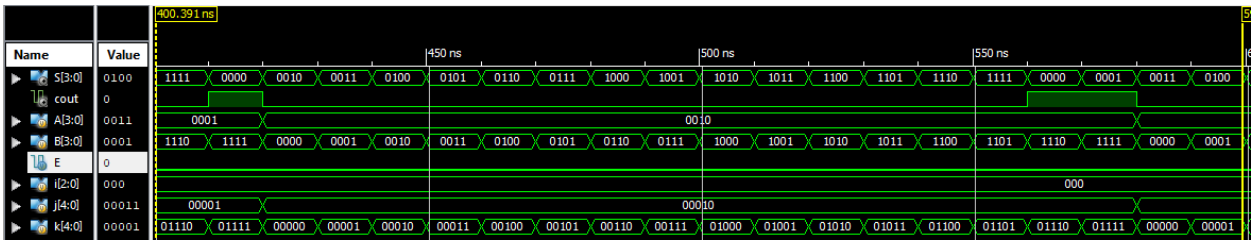
(0ns to 200ns)



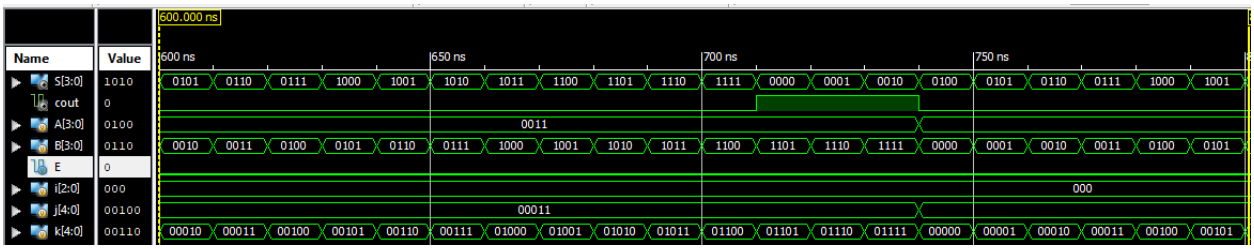
(200ns to 400ns)



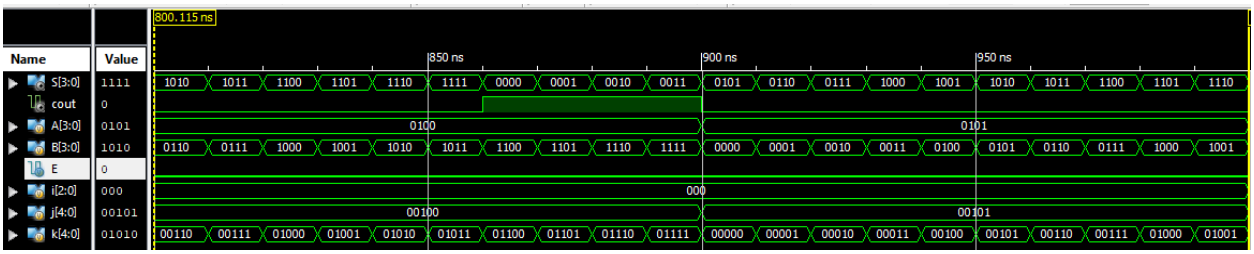
(400ns to 600ns)



(600ns to 800ns)

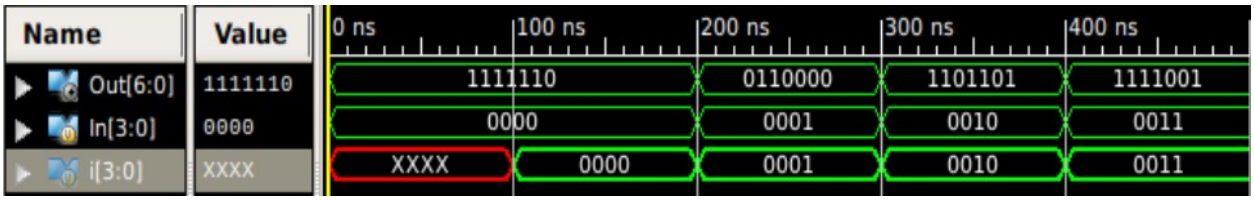


(800ns to 1000ns)

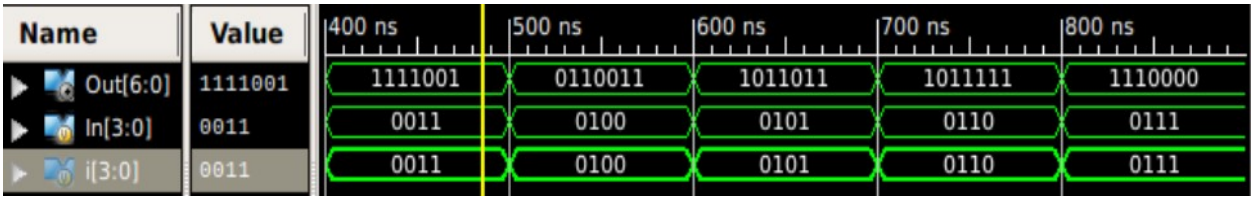


7-Segment Display:

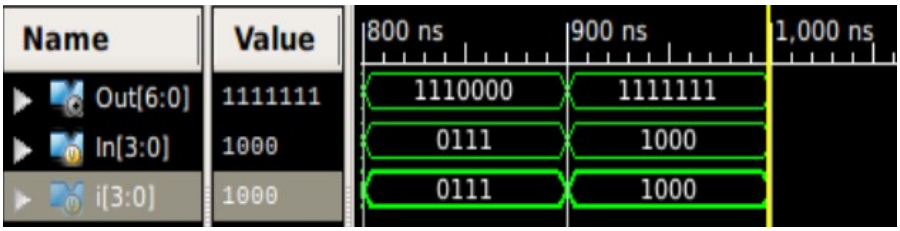
(0ns to 400ns)



(400s to 800ns)



(800ns to 1000ns)



Device Utilization Summary

Seven Segment Display:

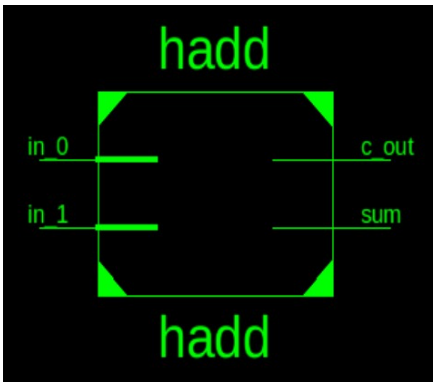
The complete DSU table is required

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	7	63400	0%
Number of fully used LUT-FF pairs	0	7	0%
Number of bonded IOBs	11	170	6%

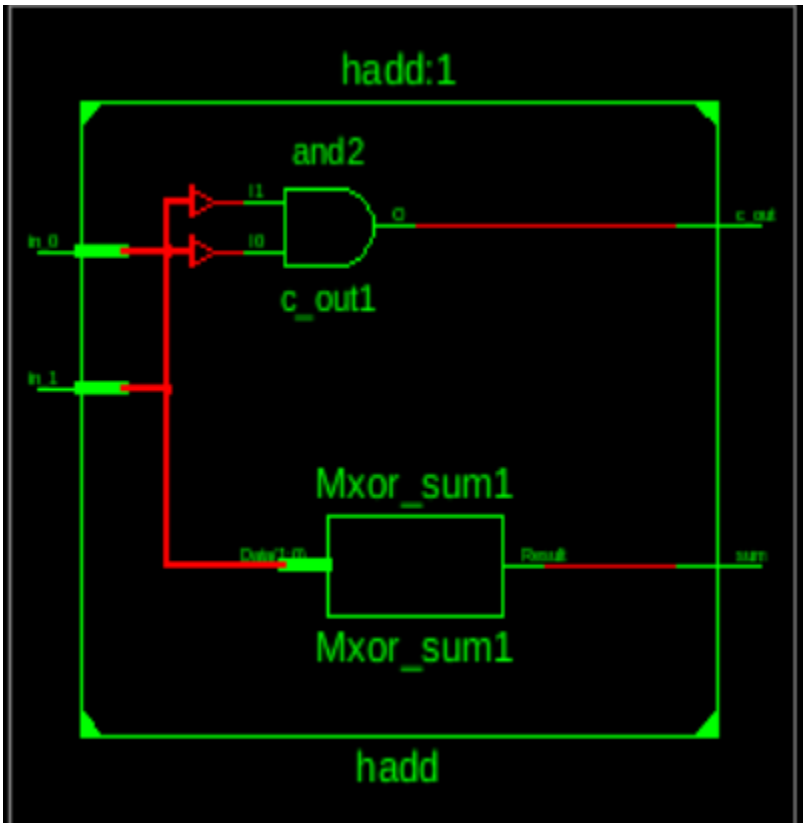
RTL Schematic and Technology Schematic

Half Adder:

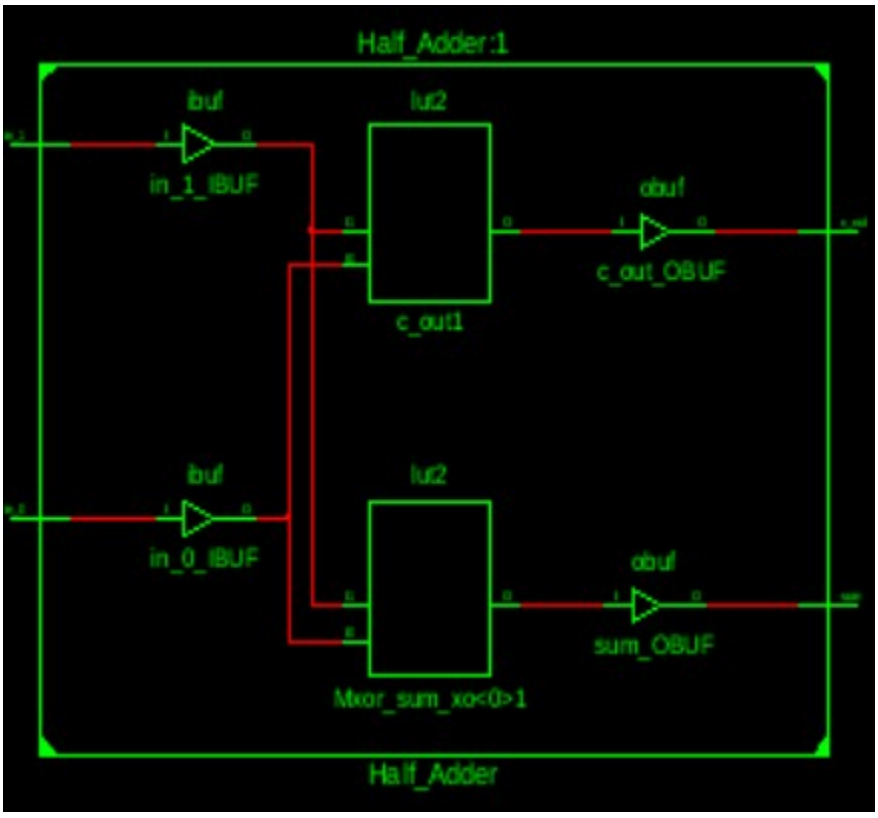
Top Level Schematic

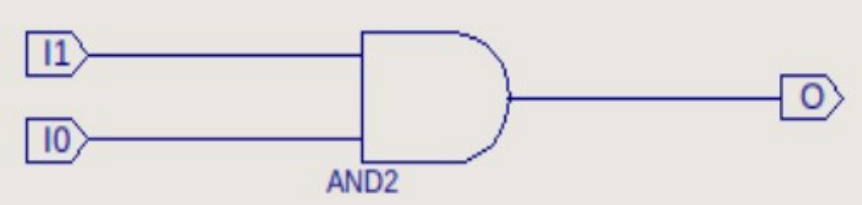


RTL Schematic Internal View



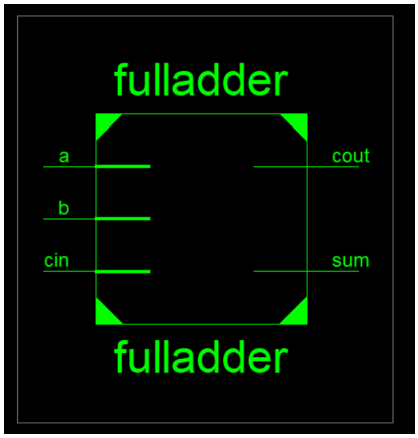
Technical Schematic Internal View



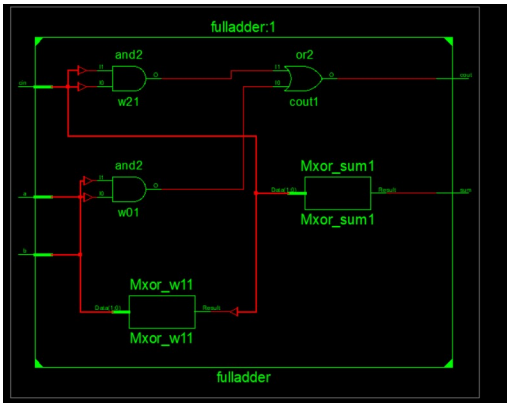


Full Adder:

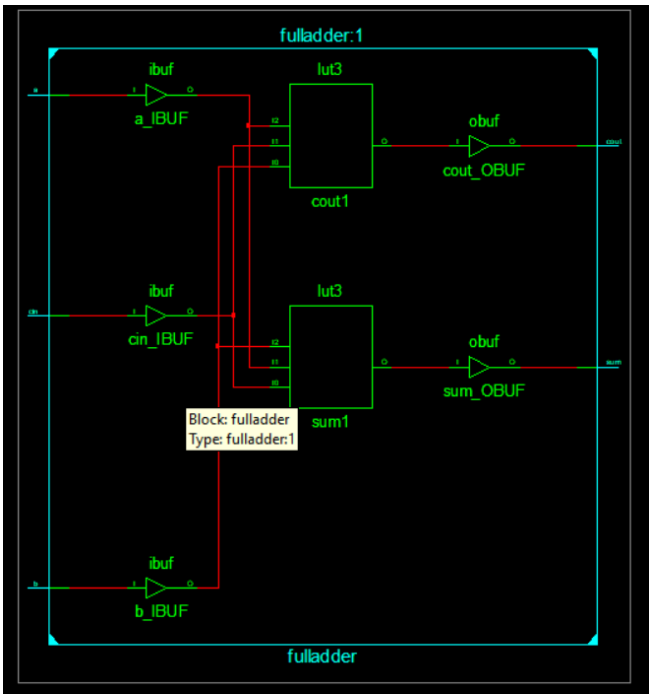
Top Level Schematic

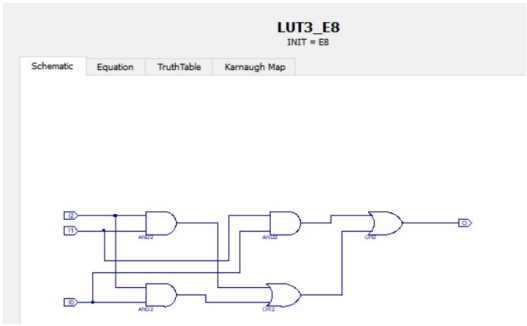


RTL Schematic Internal View



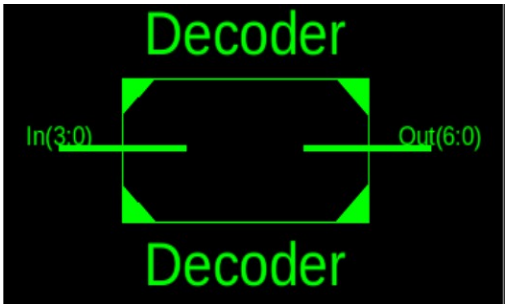
Technical Schematic Internal View



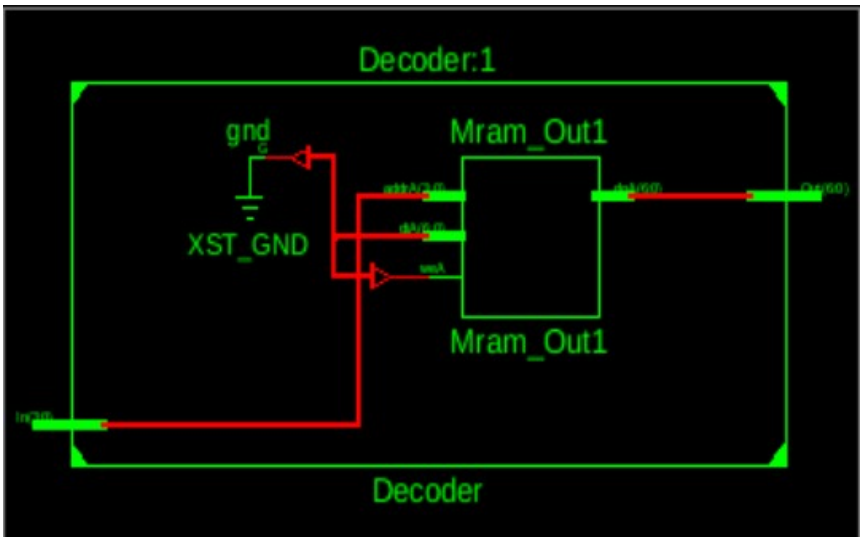


Seven Segment Display:

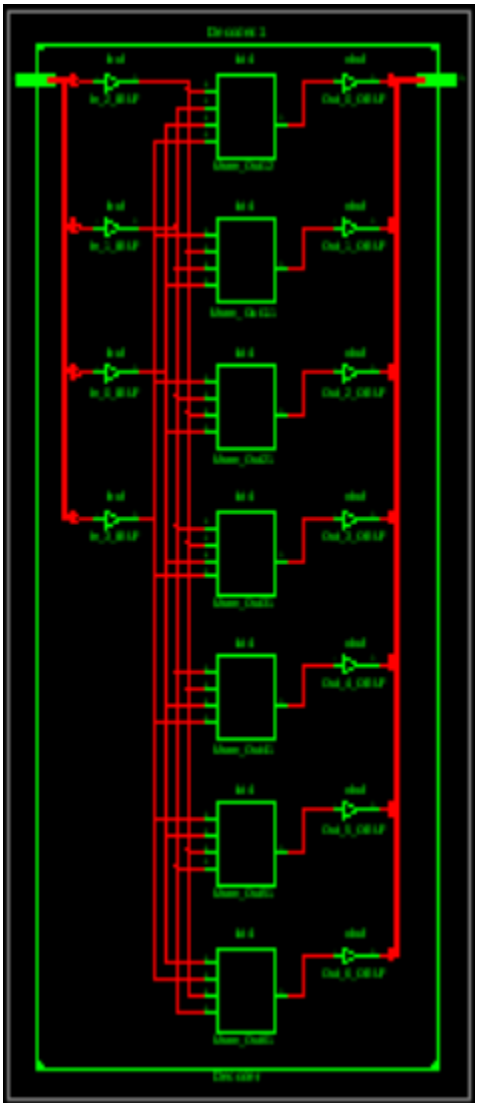
Top Level Schematic

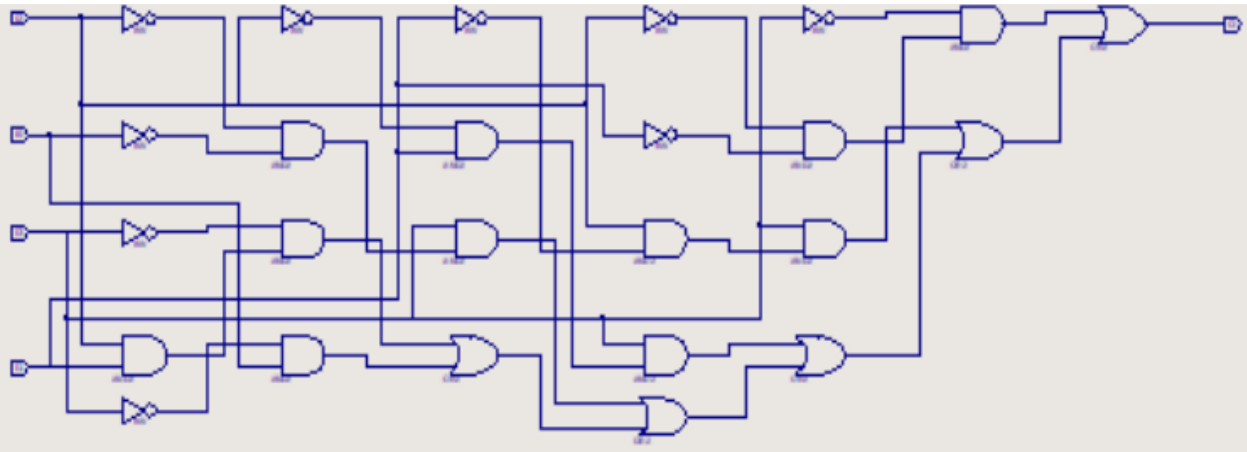


RTL Schematic Internal View



Technical Schematic Internal View





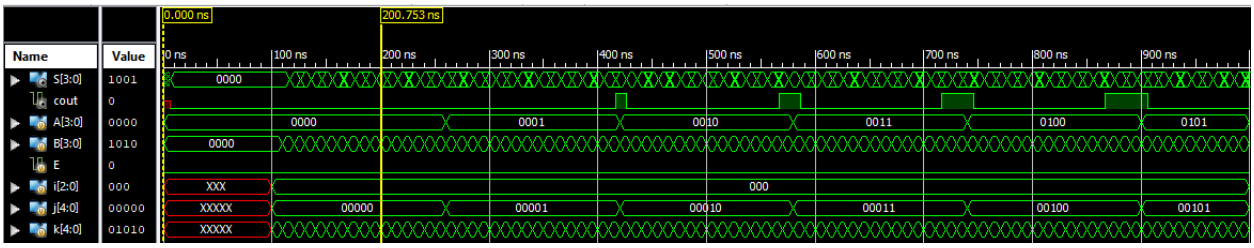
Post Place and Route Static Timing

We were unable to complete the seven segment display by the end of this lab. It will be completed for the following lab.

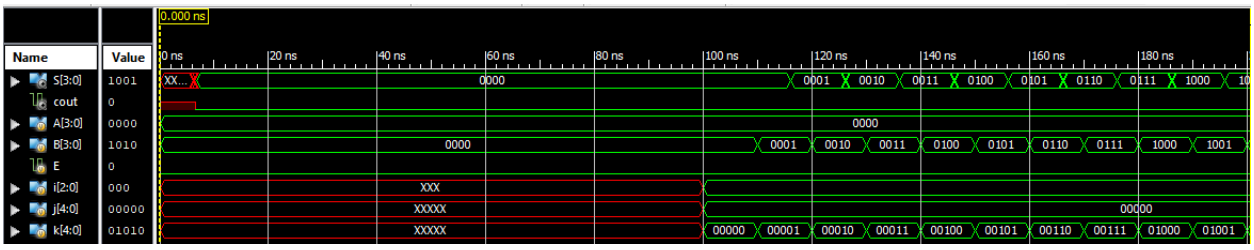
Post Place and Route Simulation Results

4-bit Adder Subtractor:

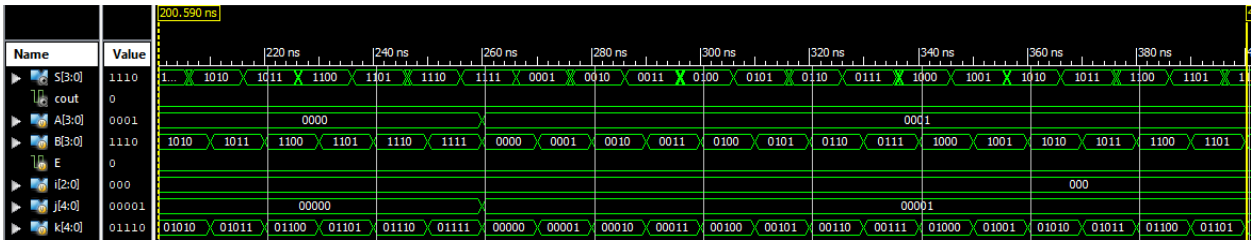
(Overview)



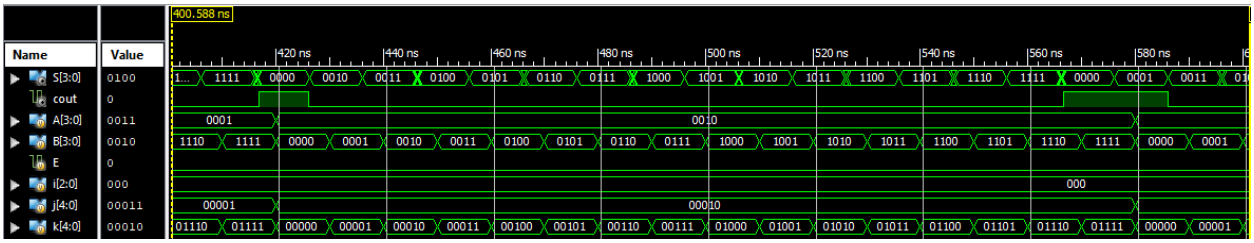
(0ns to 200ns)



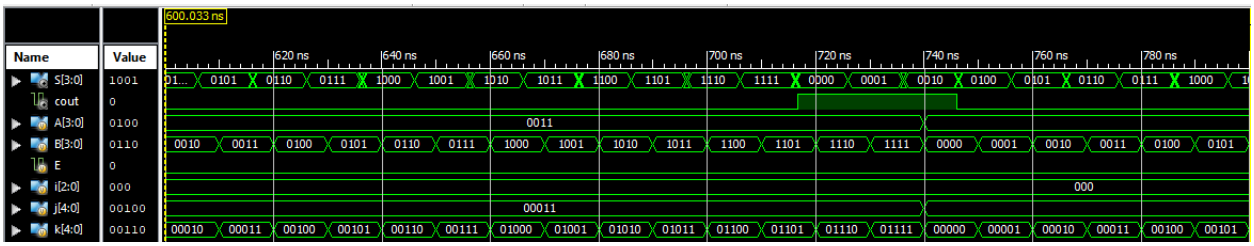
(200ns to 400ns)



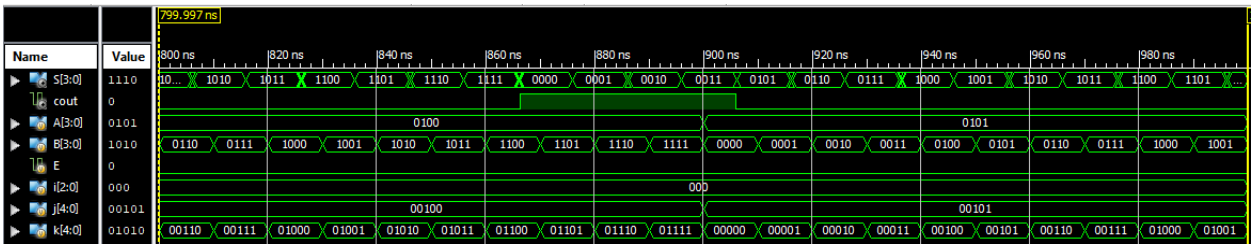
(400ns to 600ns)



(600ns to 800ns)



(800ns to 1000ns)



Power Data

We were unable to complete the seven segment display by the end of this lab. It will be completed for the following lab.

Conclusions

To start the lab, the half adder was created using gate primitives XOR and AND. The full adder was then made based on the half adder's instances. This was turned into a full four bit adder and subtractor. The four bit adder and subtractor were instantiated into the seven segment display which was given to us beforehand. We had a misunderstanding that the entire seven segment display Verilog code was to be made entirely from scratch by us, we later found out that it was already given and needed to be implemented into our code.

References

[1]: Nexys4™ FPGA Board Reference Manual, Nexys-4 rev. B; Revised November 19, 2013 by Diligent Beyond Theory

[2]: Introduction to FPGA ECE414 - Fall 2020 pdf by Randil Gajasinghe and Prof. Onur Tigli

[3]: ECE 414 Computer Organization and Design - Experiment 2. Adder/Subtractor with 7-Segment Display