Laboratory Report #5

Name: **German E Felisarta III** **Group Number: 3**

Laboratory Exercise Title: **Behavioral Modeling of Combinational** **Date Completed:** **11/11/2020**

Target Course Outcomes:

CO1: Create descriptions of digital hardware components such as in combinational and sequential circuits using synthesizable Verilog HDL constructs.
CO2: Verify the functionality of HDL-based components through design verification tools.

Exercise 5A:

In this exercise, a 4-bit comparator was assembled, this exercise is similar to the last Lab Exercise made but just coded differently and with the use of the always @(*) syntax. All outputs are identical.

Figure 1a. Design Entry for 4 bit Comparator

fourBitComparator.v

```
1
2   //GERMAN E. FELISARTA III  16101002 CpE3101L
3
4   module fourBitComparator(R, A, B);
5
6      input wire [3:0]A;
7      input wire [3:0]B;
8      output reg [2:0]R;
9
10     always @ (*)
11        begin
12           if (A > B)
13              R[2] = 1;
14           else
15              R[2] = 0;
16
17           if (A == B)
18              R[1] = 1;
19           else
20              R[1] = 0;
21
22           if (A < B)
23              R[0] = 1;
24           else
25              R[0] = 0;
26        end
27
28  endmodule
29
30
```

Figure 1b. RTL view for 4 bit Comparator

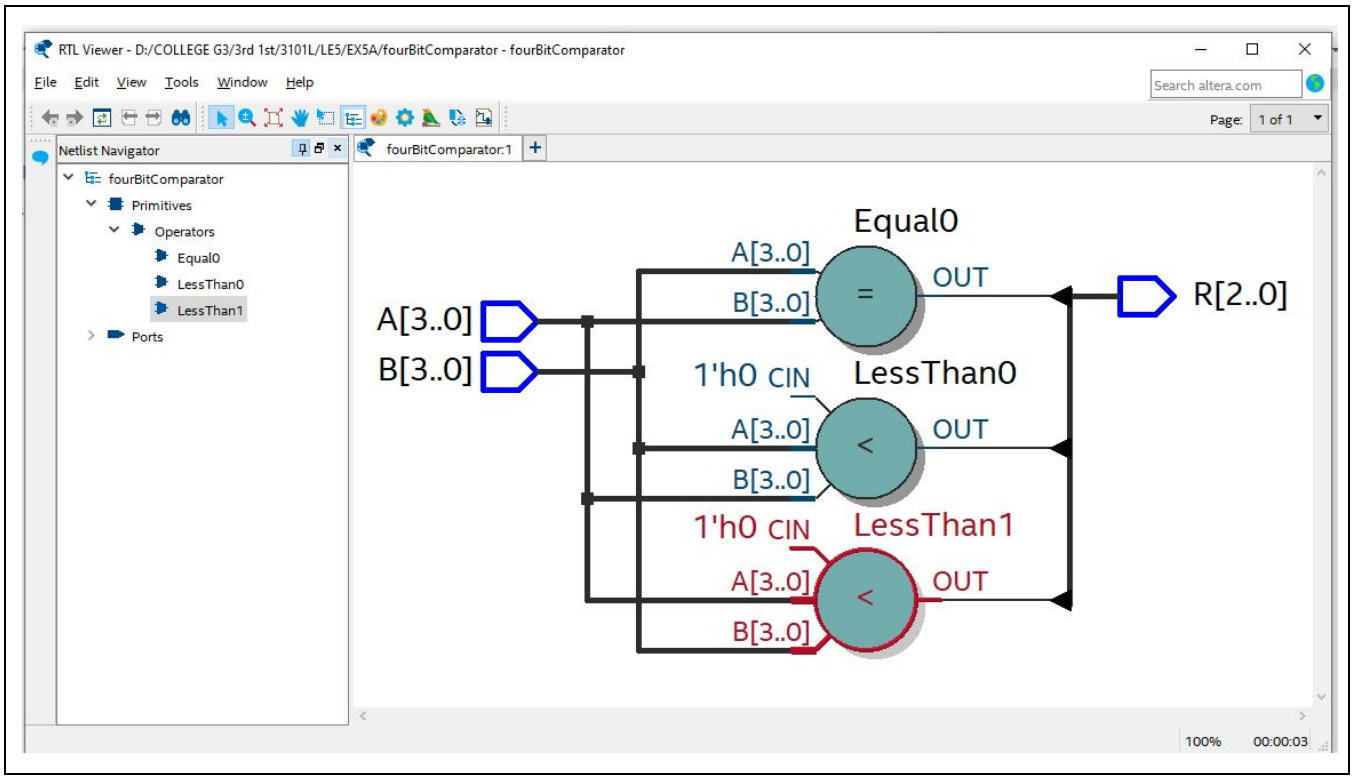


Figure 1c. Flow Summary for 4 bit Comparator

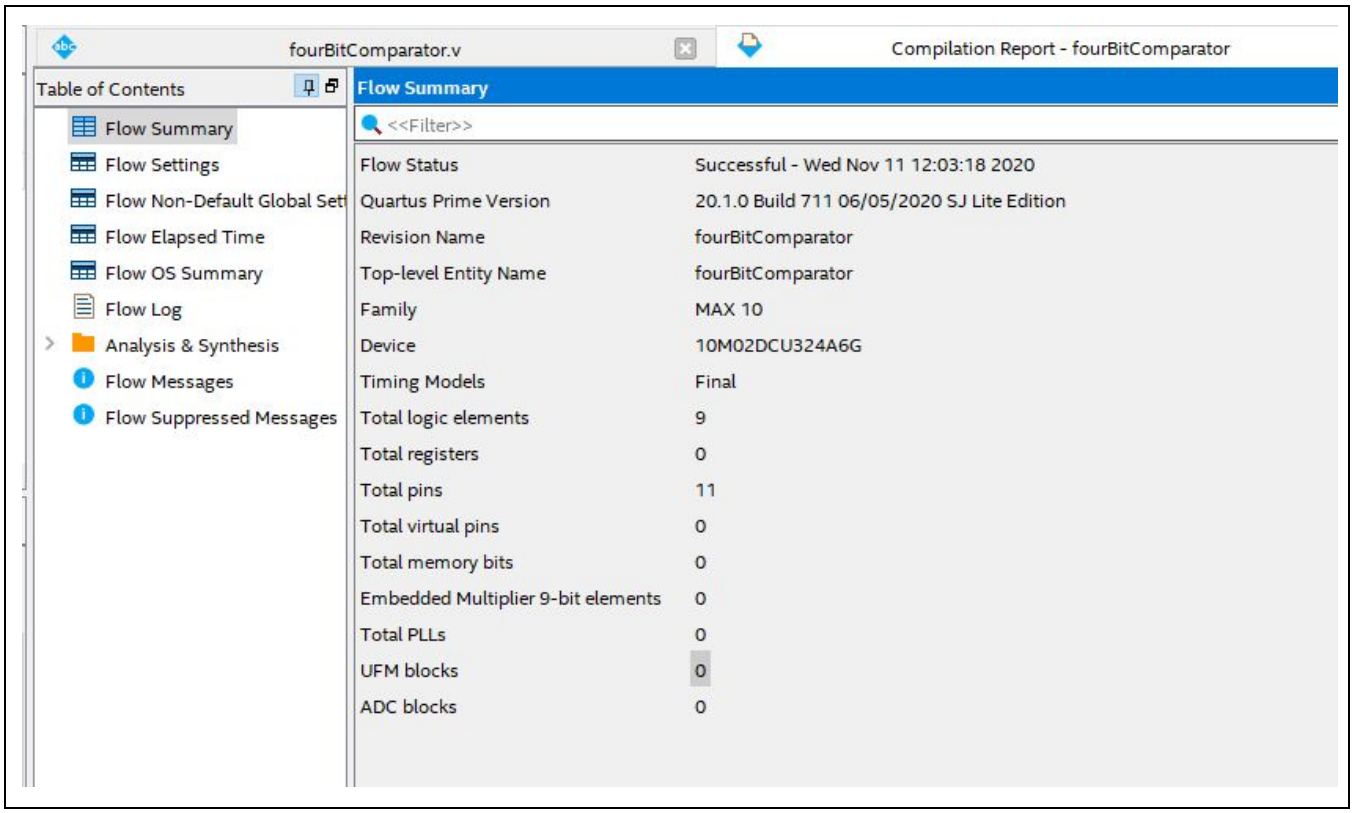
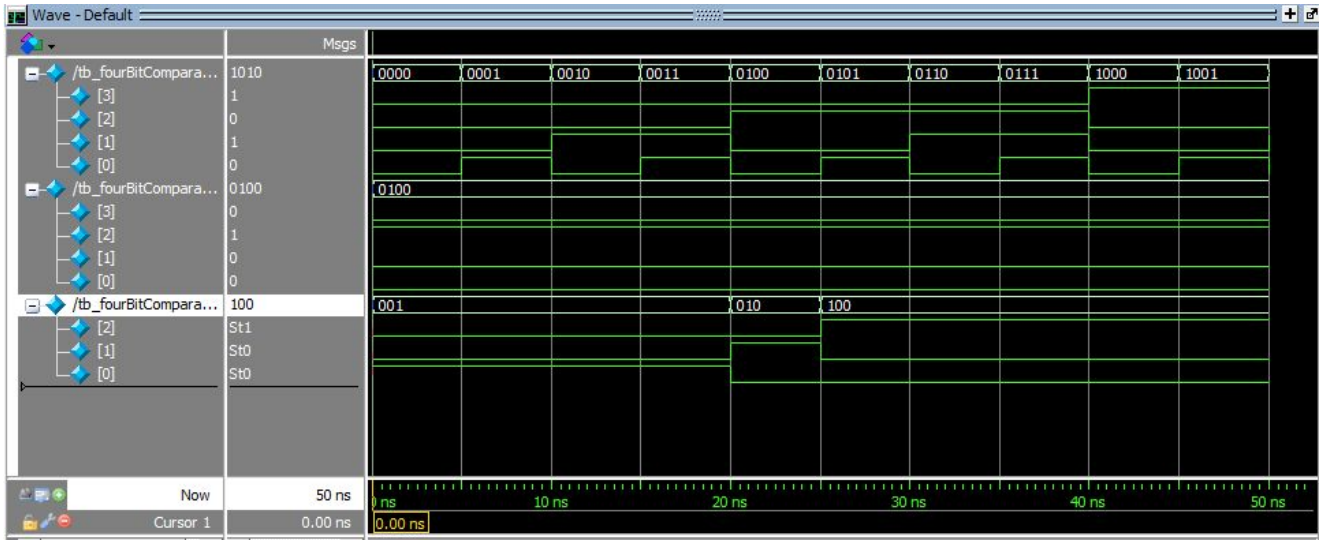


Figure 1d. Test Bench for 4 bit Comparator

fourBitComparator.v | Compilation Report - fourBitComparator | tb_fourBitComparator.v

```
1  //GERMAN E FELISARTA III 16101002   CpE3101L Grp 3
2
3  `timescale 1 ps / 1 ns
4  module tb_fourBitComparator();
5
6     reg [3:0]A;
7     reg [3:0]B;
8
9     wire [2:0]R;
10
11    fourBitComparator UUT (R, A, B);
12
13    initial begin
14       $display ("Starting simulation at %0d ns..", $time);
15
16       A = 4'b0000;
17       B = 4'b0001;
18
19       repeat(5)
20       #5 A = A + 4'b0001;
21
22       A = 4'b0001;
23       B = 4'b0000;
24
25       repeat(5)
26       #5 B = B + 4'b0001;
27
28
29
30       $display ("Finished simulation at &0d ns.", $time);
31
32       $stop;
33    end
34
35 endmodule
36
```

Figure 1e. Timing Diagram for 4 bit Comparator



Exercise 5B:

In this exercise, the objective is to create a hex to 7-segment decoder. Since it is a 3 digit hex, then all inputs will only be up to F_{16}. Referring to the diagram on the Lab Guide, the output SSeg is then determined where all the diodes that light up is equivalent to a digit place in the 7digit binary number.

Figure 2a. Design Entry for Hex to 7-Segment Decoder

hexaDigit.v*

```
1   //GERMAN E FELISARTA III 16101002   CpE 3101L
2
3   module hexaDigit(Hex, DP, SSeg);
4
5      input wire [3:0]Hex;
6      input wire DP;
7      output reg [7:0]SSeg;
8
9      always @ (*)
10        begin
11              case ({DP, Hex})
12                 5'b00000 : SSeg = 8'b0111111;
13                 5'b00001 : SSeg = 8'b0000110;
14                 5'b00010 : SSeg = 8'b1011011;
15                 5'b00011 : SSeg = 8'b1001111;
16                 5'b00100 : SSeg = 8'b1100110;
17                 5'b00101 : SSeg = 8'b1101101;
18                 5'b00110 : SSeg = 8'b1111101;
19                 5'b00111 : SSeg = 8'b0000111;
20                 5'b01000 : SSeg = 8'b1111111;
21                 5'b01001 : SSeg = 8'b1101111;
22                 5'b01010 : SSeg = 8'b1110111;
23                 5'b01011 : SSeg = 8'b1111100;
24                 5'b01100 : SSeg = 8'b0111001;
25                 5'b01101 : SSeg = 8'b1011110;
26                 5'b01110 : SSeg = 8'b1111001;
27                 5'b01111 : SSeg = 8'b1110001;
28                 default : SSeg = 8'b0000000;
29
30              endcase
31        end
32
33  endmodule
34
35
36  /*              GFEDCBA
37        0000 - 0 - 0111111
38        0001 - 1 - 0000110
39        0010 - 2 - 1011011
40        0011 - 3 - 1001111
```

Figure 2b. RTL view for Hex to 7-Segment Decoder

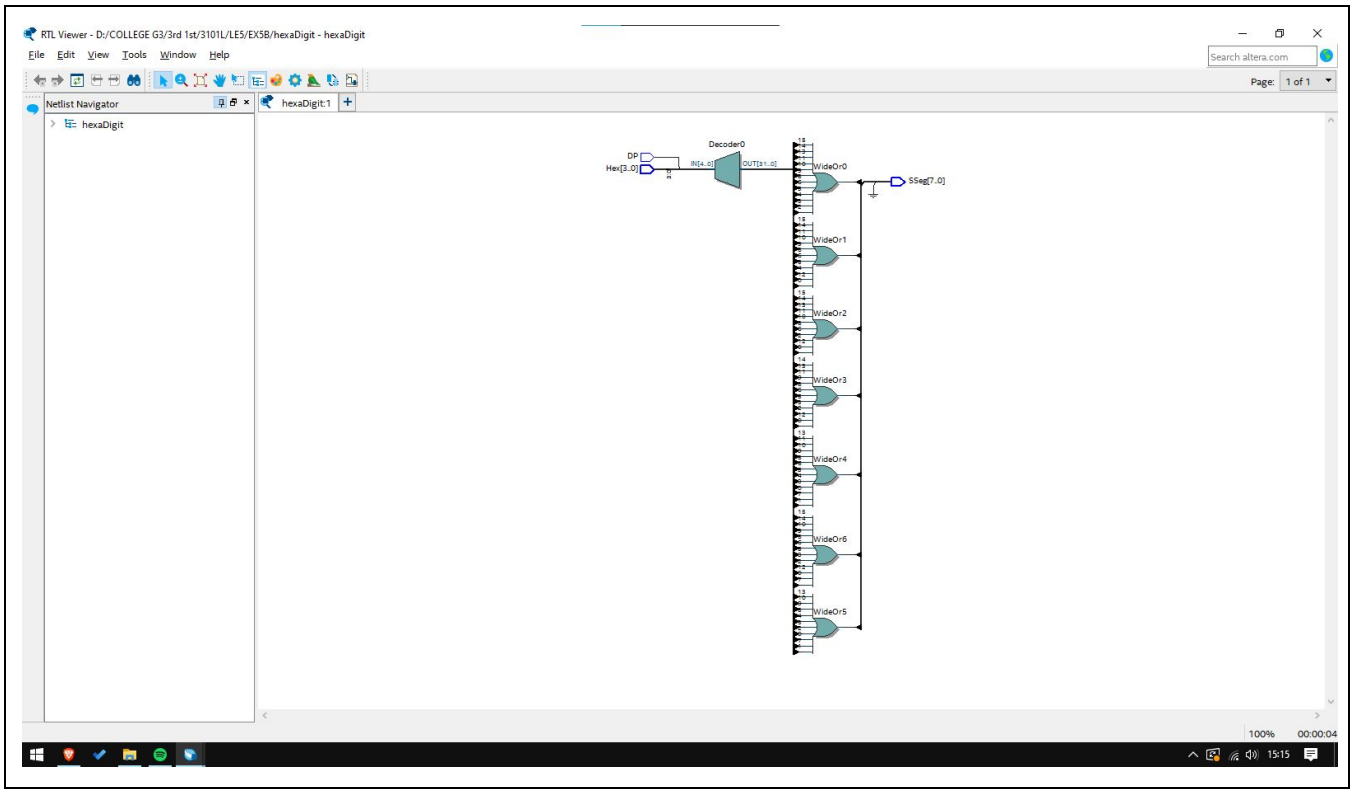


Figure 2c. Flow Summary for Hex to 7-Segment Decoder

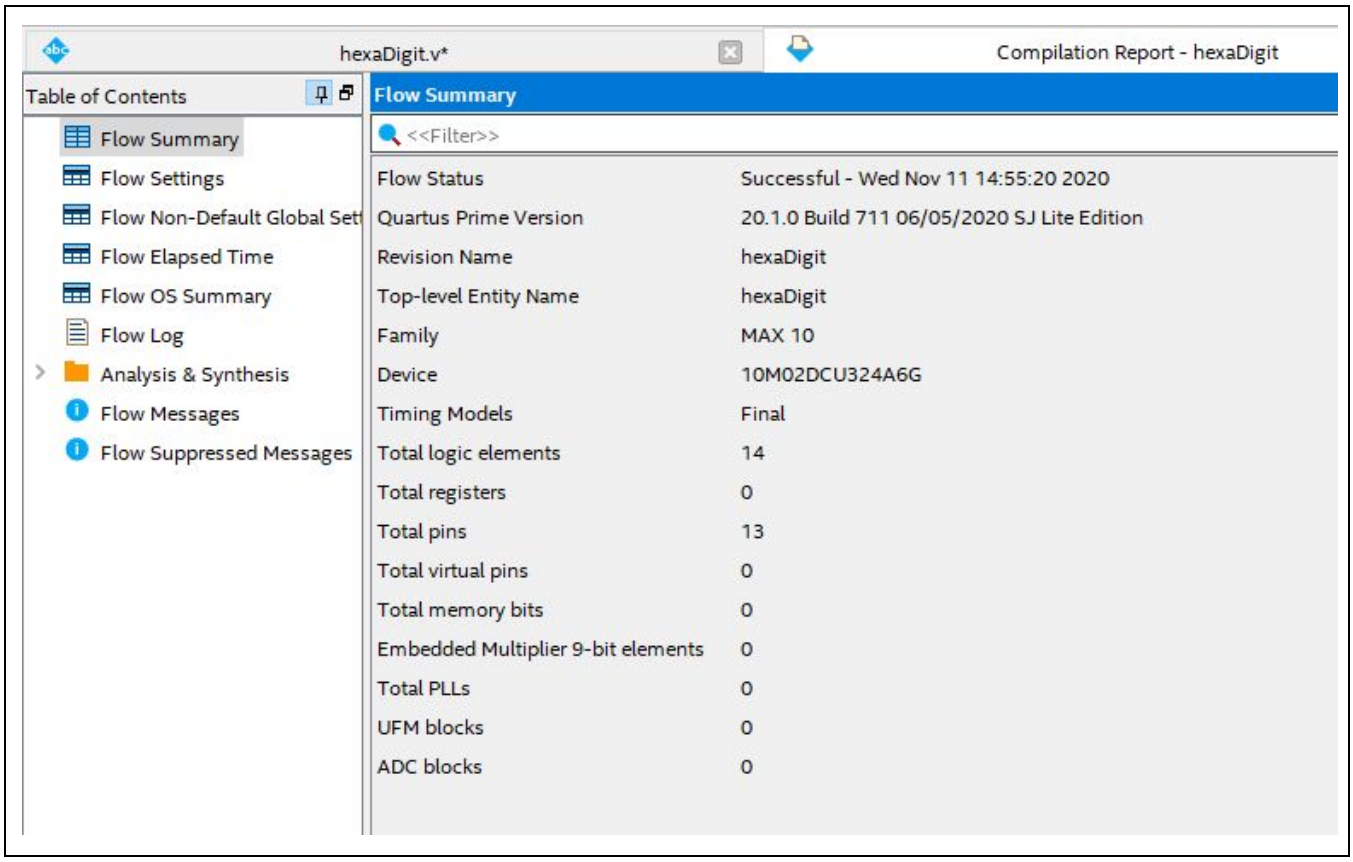
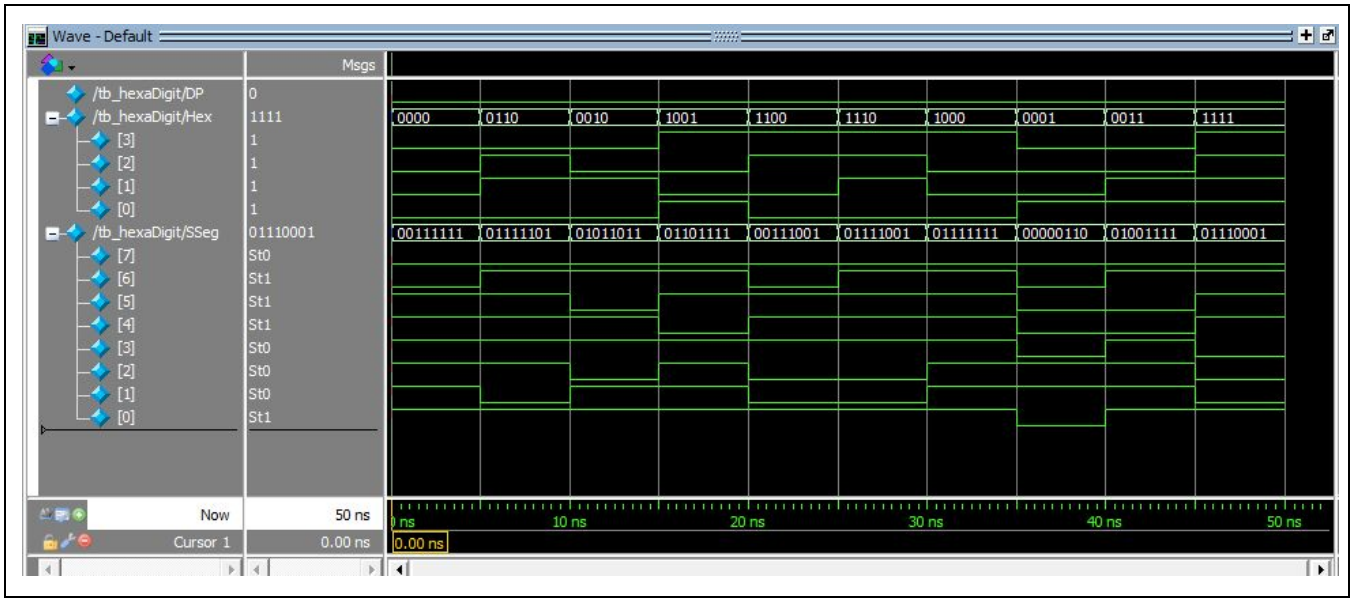


Figure 2d. Test Bench for Hex to 7-Segment Decoder

hexaDigit.v | Compilation Report - hexaDigit | tb_hexaDigit.v

```
1  //GERMAN E FELISARTA III   16101002 CpE 3101L Group 3
2
3  `timescale 1 ns / 1 ps
4  module tb_hexaDigit();
5
6     reg DP;
7     reg [3:0]Hex;
8     wire [7:0]SSeg;
9
10    hexaDigit UUT (Hex, DP, SSeg);
11
12    initial begin
13       $display ("Starting simulation at %0d ns...", $time);
14
15
16       {DP, Hex} = 5'b00000; #5
17       {DP, Hex} = 5'b00110; #5
18       {DP, Hex} = 5'b00010; #5
19       {DP, Hex} = 5'b01001; #5
20       {DP, Hex} = 5'b01100; #5
21       {DP, Hex} = 5'b01110; #5
22       {DP, Hex} = 5'b01000; #5
23       {DP, Hex} = 5'b00001; #5
24       {DP, Hex} = 5'b00011; #5
25       {DP, Hex} = 5'b01111; #5
26
27
28       $display ("Finished simulation at %0d ns...", $time);
29       $stop;
30
31    end
32
33 endmodule
```

Figure 2e. Timing Diagram for Hex to 7-Segment Decoder



Exercise 5B:

In this exercise, an ALU was created. Using a case syntax, the Mode is then checked for input and the corresponding output is assigned. All conditions are based-off of the table in the Laboratory Guide.

Figure 3a. Design Entry for n-Bit ALU

ALUnBit.v

```
1  //German E Felisarta III   16101002    CpE 3101L Grp 3
2
3  module ALUnBit(A, B, CB_in, Mode, Result, CB_out);
4
5     localparam n = 4;
6
7     input wire [(n-1):0]A;
8     input wire [(n-1):0]B;
9     input wire CB_in;
10    input wire [2:0]Mode;
11
12    output reg CB_out;
13    output reg [(n-1):0]Result;
14
15    always @(*)
16       begin
17          case (Mode)
18
19             3'b000 :
20                begin Result = (A + B); CB_out = 1; end
21             3'b001 :
22                begin Result = (A - B); CB_out = 1; end
23             3'b010 :
24                Result = (A & B);
25             3'b011 :
26                Result = (A | B);
27             3'b100 :
28                Result = (A ^ B);
29             3'b101 :
30                Result = (~A);
31             3'b110 :
32                Result = (A + 1'b1);
33             3'b111 :
34                Result = (A - 1'b1);
35
36          endcase
37       end
```

Figure 3b. RTL view for n-Bit ALU

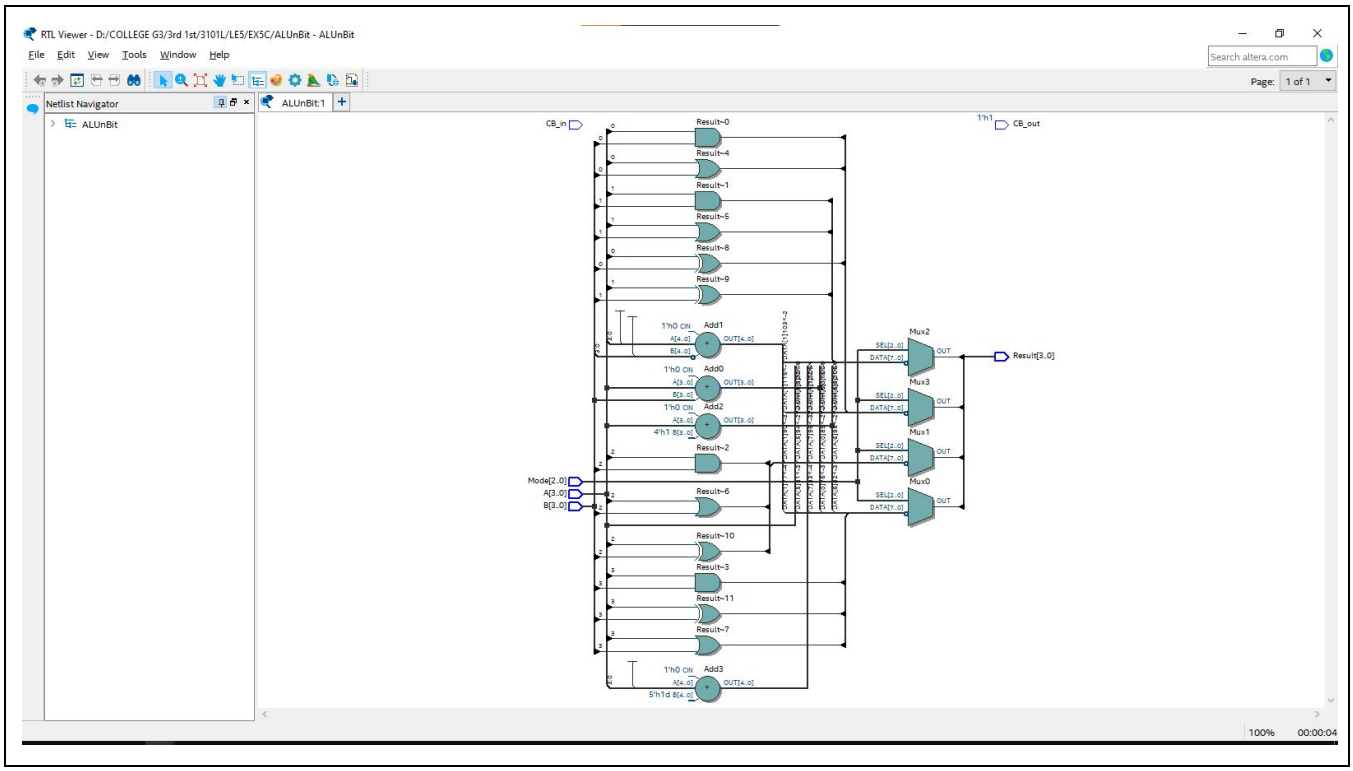


Figure 3c. Flow Summary for n-Bit ALU

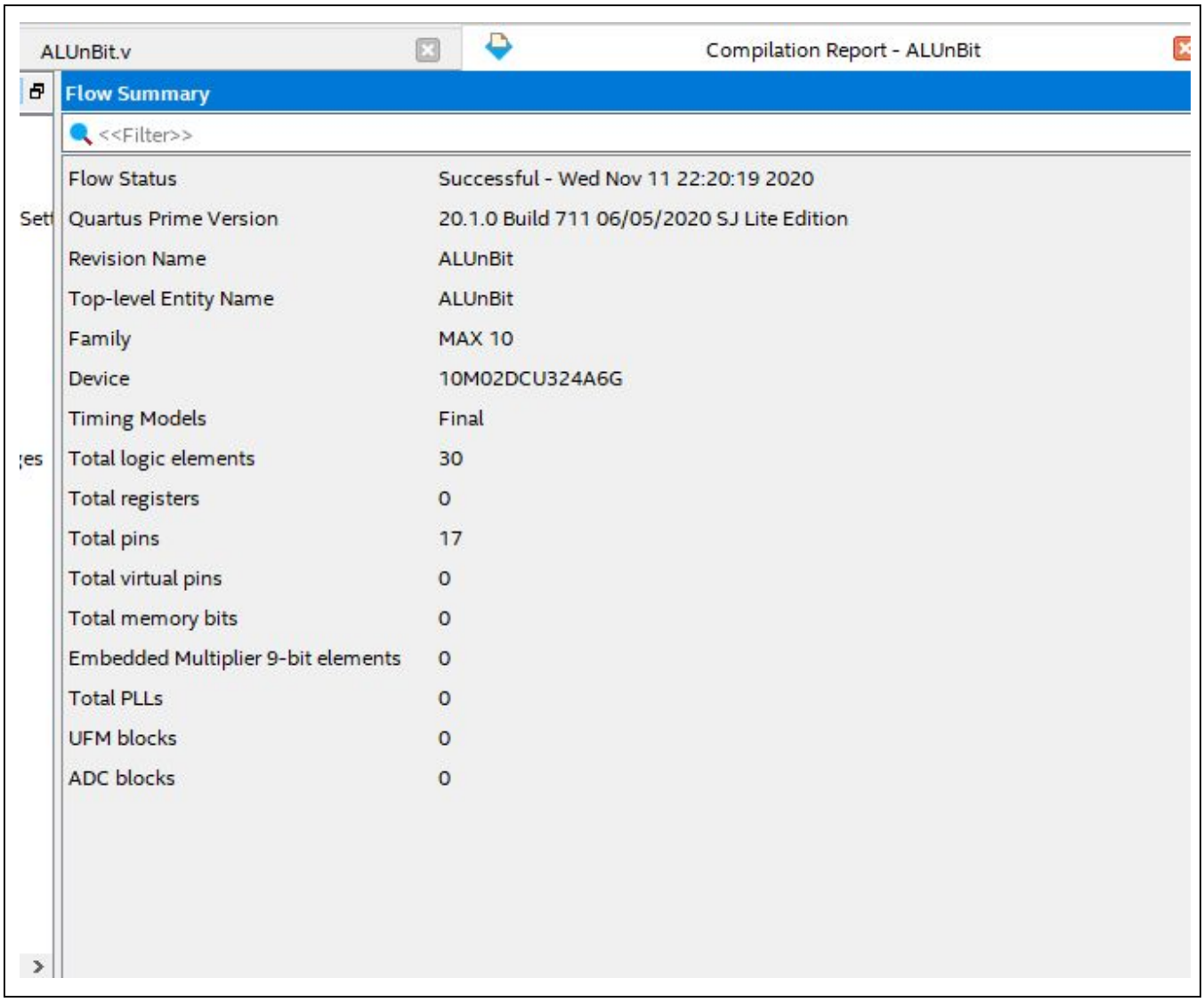


Figure 3d. Test Bench for n-Bit ALU

```
//German E Felisarta III  16101002        CpE 3101L Grp 3

`timescale 1 ns / 1 ps
module tb_ALUnBit();

        localparam n = 4;

        reg [(n-1):0]A;
        reg [(n-1):0]B;
        reg CB_in;
        reg [2:0]Mode;

        wire CB_out;
        wire [(n-1):0]Result;

        ALUnBit UUT (A, B, CB_in, Mode, Result, CB_out);

        initial begin
                $display("Starting simulation at %0d ns...", $time);

                //000

                A = 4'b0001;
                B = 4'b0010;
                CB_in = 1'b1;
                Mode = 3'b000;

                A = 4'b0011;
                B = 4'b0001;
                CB_in = 1'b1;
                Mode = 3'b000;

                A = 4'b0101;
                B = 4'b0010;
                CB_in = 1'b0;
                Mode = 3'b000;

                //001

                A = 4'b0011;
                B = 4'b0010;
                Mode = 3'b001;

                A = 4'b0111;
                B = 4'b0011;
                Mode = 3'b001;

                A = 4'b0001;
                B = 4'b0010;
                Mode = 3'b001;

                //010
```

```
A = 4'b0011;
B = 4'b0010;
Mode = 3'b010;

A = 4'b0111;
B = 4'b0111;
Mode = 3'b010;

A = 4'b0010;
B = 4'b0010;
Mode = 3'b010;

//011

A = 4'b0011;
B = 4'b0010;
Mode = 3'b011;

A = 4'b0111;
B = 4'b0111;
Mode = 3'b011;

A = 4'b0010;
B = 4'b0010;
Mode = 3'b011;

//100

A = 4'b0011;
B = 4'b0010;
Mode = 3'b100;

A = 4'b0111;
B = 4'b0111;
Mode = 3'b100;

A = 4'b0010;
B = 4'b0010;
Mode = 3'b100;

//101

A = 4'b0011;
Mode = 3'b101;

A = 4'b0111;
Mode = 3'b101;

A = 4'b0010;
Mode = 3'b101;

//110
```

```
                A = 4'b0011;
                Mode = 3'b110;

                A = 4'b0111;
                Mode = 3'b110;

                A = 4'b0010;
                Mode = 3'b110;

                //111

                A = 4'b0011;
                Mode = 3'b111;

                A = 4'b0111;
                Mode = 3'b111;

                A = 4'b0010;
                Mode = 3'b111;


                $display("Finished simulation at %0d ns.", $time);
                $stop;
        end

endmodule
```

Figure 3e. Timing Diagram for n-Bit ALU