Laboratory Report # 4

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

Laboratory Exercise Title: **Dataflow Modeling of Combinational Cir.** **Date Completed:** **11/02/2020**

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 4A:

In this exercise, a 4-bit comparator was to be made. Where the outputs would vary depending on the values of A and B. The values on the test bench were randomized and based on the timing diagram, the exercise was successfully completed.

Figure 1a. Design Entry for 4 bit Comparator

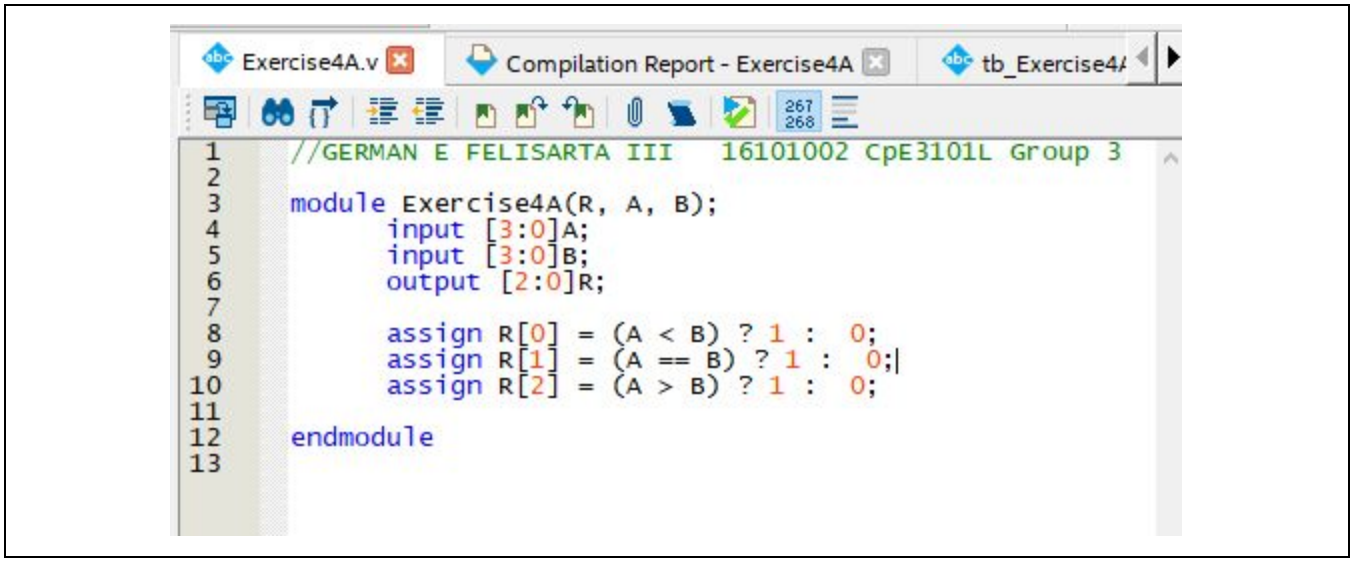


Figure 1b. Flow Summary for 4 bit Comparator

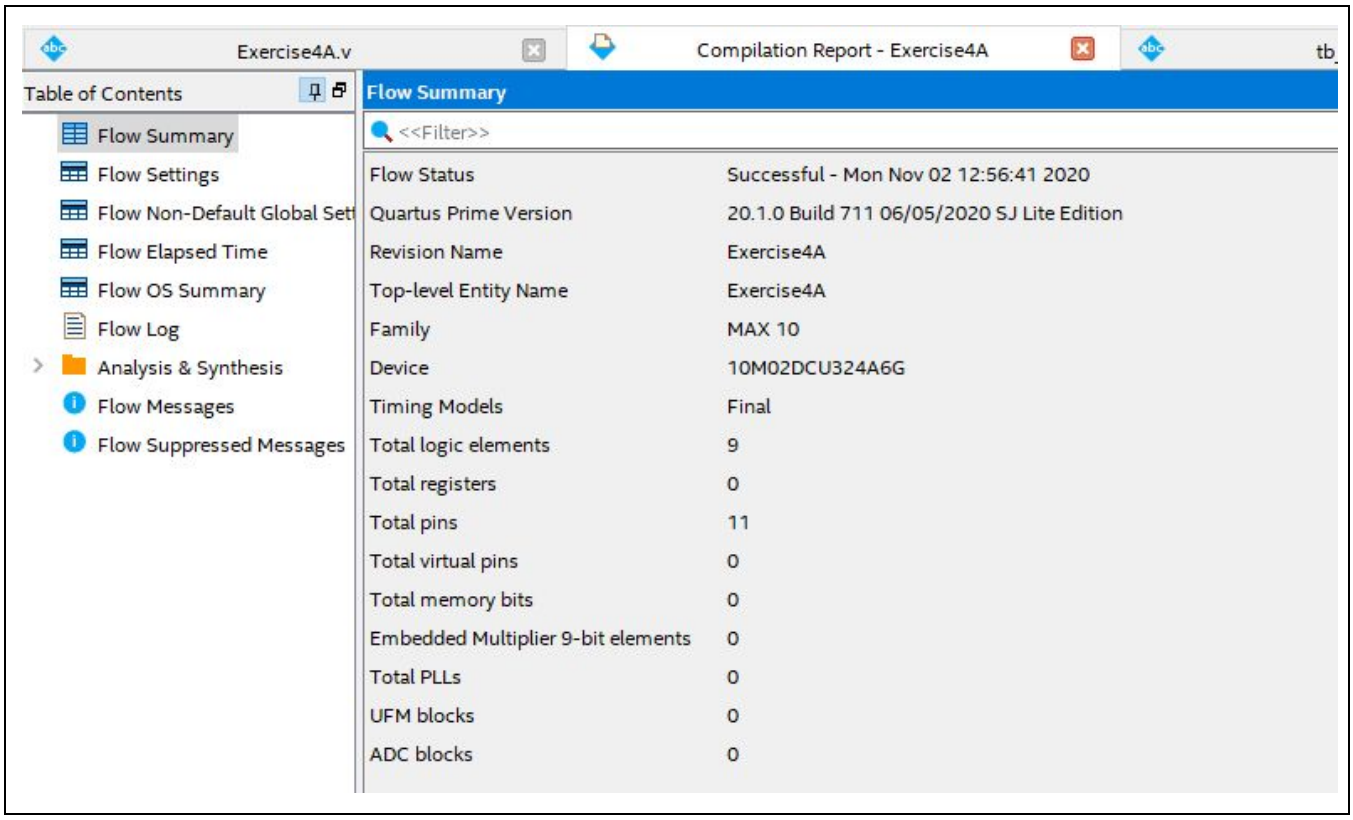


Figure 1c. Test Bench Design Entry for 4 bit Comparator

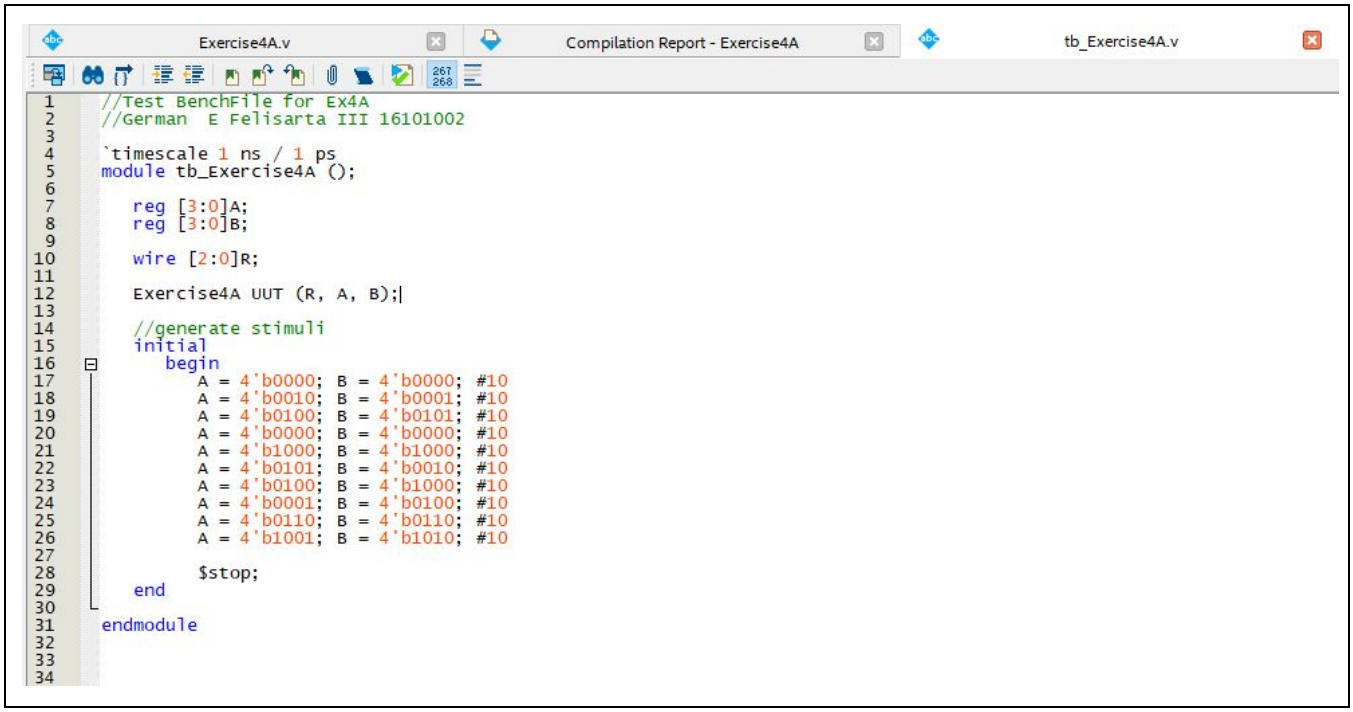
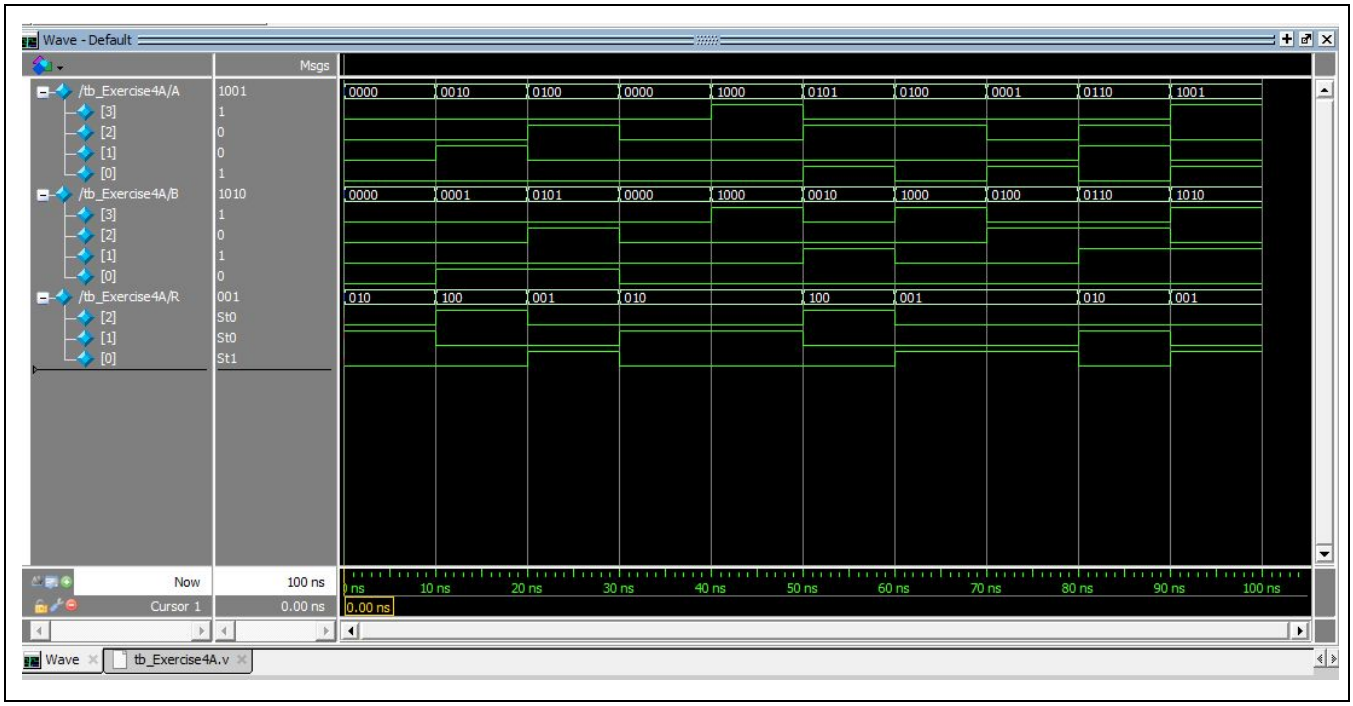


Figure 1d. Timing Diagram for 4 bit Comparator



Exercise 4B:

Before starting the coding phase of the exercise, a truth table was made based on the 3-bit parity generator. All outputs with the input E as low will be z or high impedance. After getting the truth table, the Kmap and boolean functions were generated. The process is much simpler now because of the if-else syntax making muxes easier to create. The outputs of the timing diagram were as expected.

Figure 2a. Truth Table for 3-bit Even Parity Generator

A[0]	A[1]	A[2]	E	P
0	0	0	1	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1
x	x	x	0	z

Figure 2b. Kmap for 3-bit Even Parity Generator

A[0]/A[1]A[2]	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$P = A[0] \oplus A[1] \oplus A[2]$$

Figure 2c. Design Entry for 3-bit Even Parity Generator

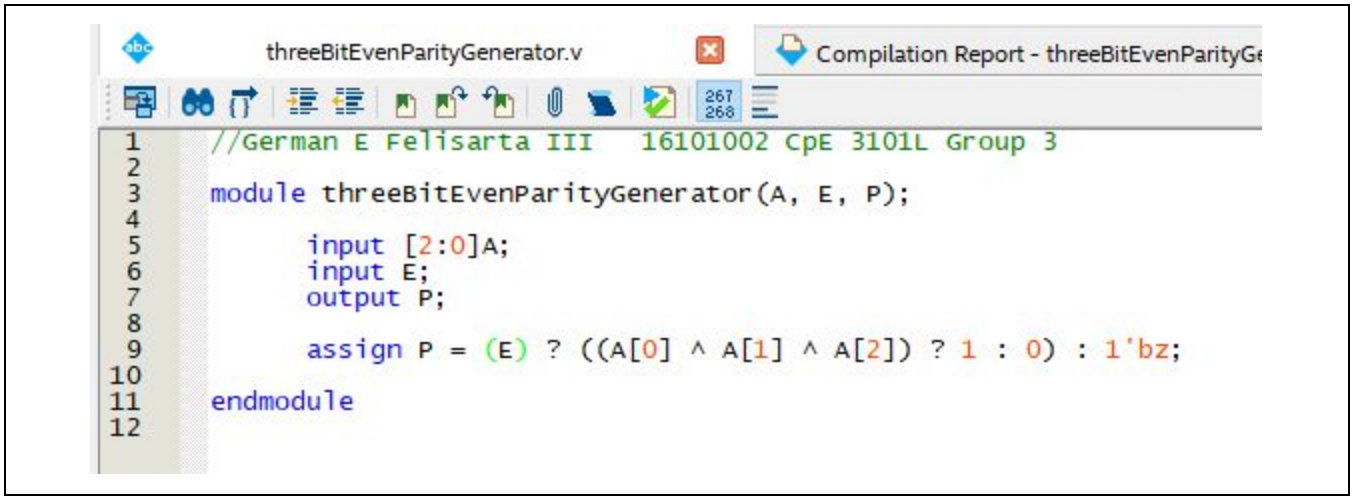


Figure 2d. Flow Summary for 3-bit Even Parity Generator

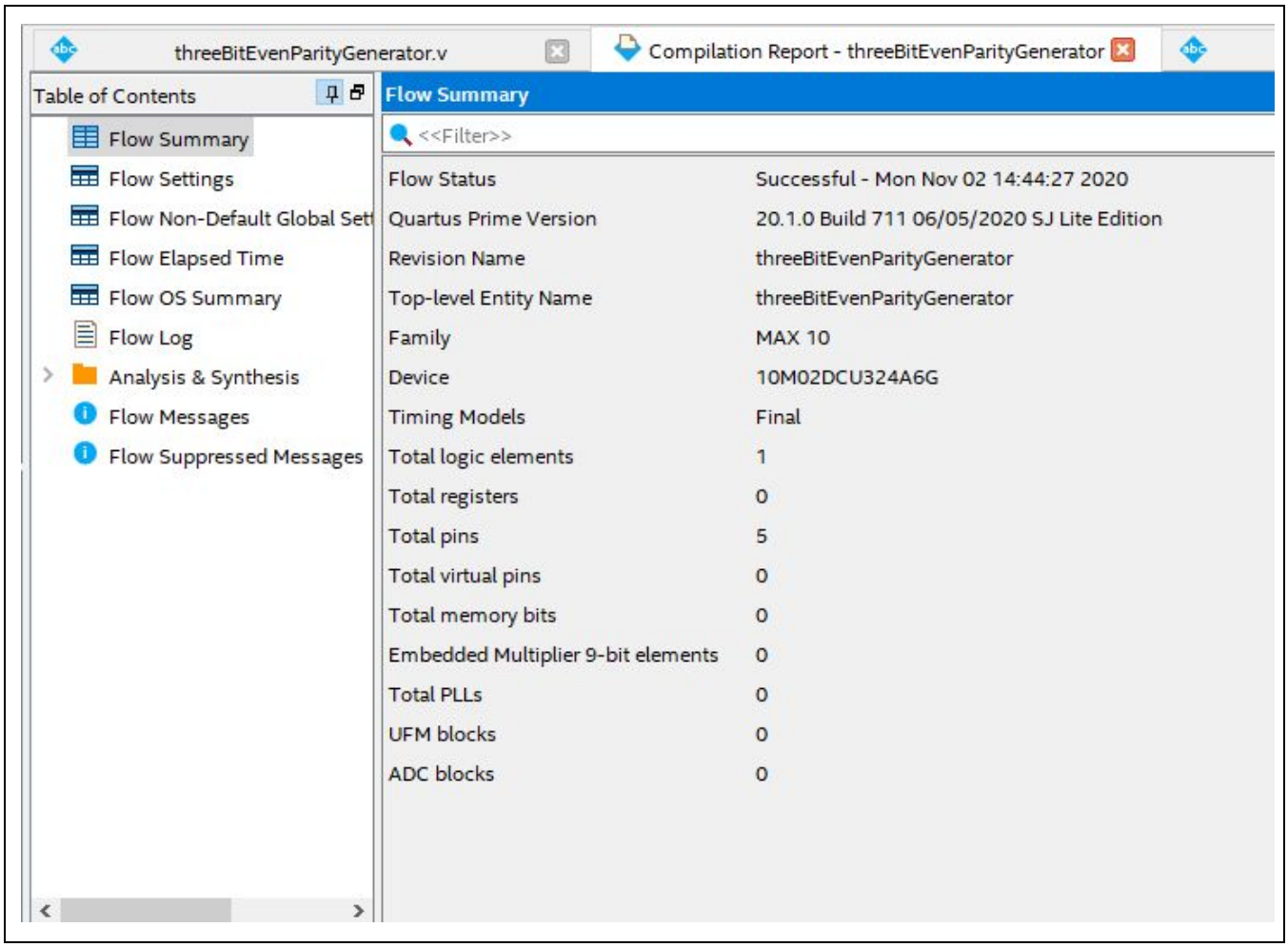
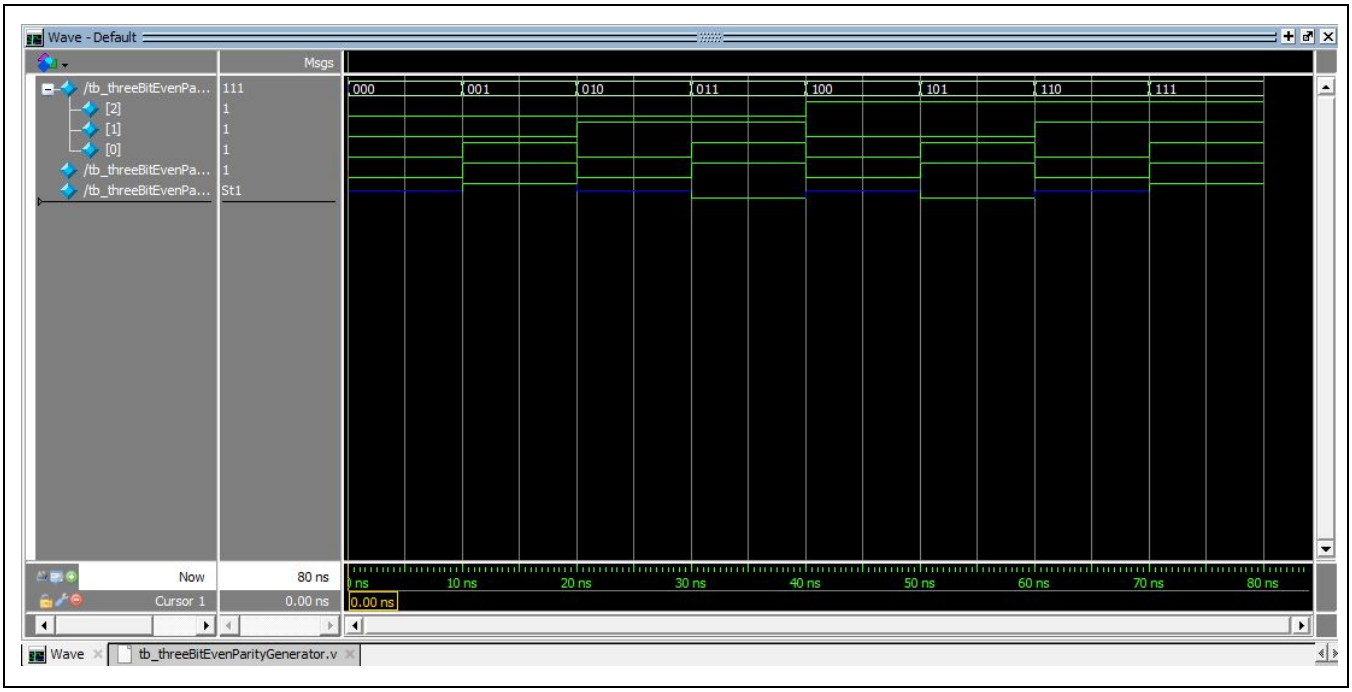


Figure 2e. Test Bench Design Entry for 3-bit Even Parity Generator

threeBitEvenParityGenerator.v | Compilation Report - threeBitEve

```
1   //TestBench file for 3Bit Parity Gen
2
3   `timescale 1 ns / 1 ps
4   module tb_threeBitEvenParityGenerator();
5
6      reg [2:0]A;
7      reg E;
8
9      wire P;
10
11     threeBitEvenParityGenerator UUT (A, E, P);
12
13     initial
14        begin
15           A = 3'b000; E = 0; #10
16           A = 3'b001; E = 1; #10
17           A = 3'b010; E = 0; #10
18           A = 3'b011; E = 1; #10
19           A = 3'b100; E = 0; #10
20           A = 3'b101; E = 1; #10
21           A = 3'b110; E = 0; #10
22           A = 3'b111; E = 1; #10
23
24        $stop;
25     end
26
27  endmodule
28
```

Figure 2f. Timing Diagram for 3-bit Even Parity Generator



Exercise 4C:

Part 1

In this exercise, a 4-bit 4x1 Mux is to be created. After many trials, the *#(parameter n = x)(...)* command does not work. To solve this problem, the *localparam* command was used and the value will be changed for part 2. The range for the input variables A, B, C, and D are up to n-1 because the constant value was already set. The circuit was then tested and yielded successful results. Where each selector input was different and would also have different outputs respectively.

Figure 3a. Design Entry for 4-bit Mux

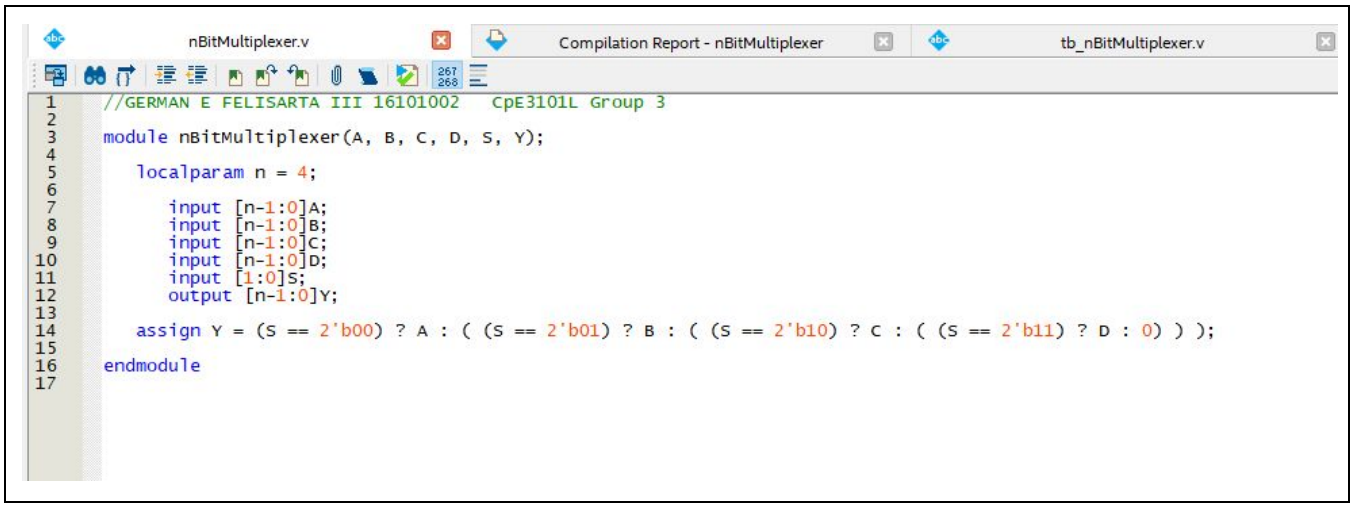


Figure 3b. Flow Summary for 4-bit Mux

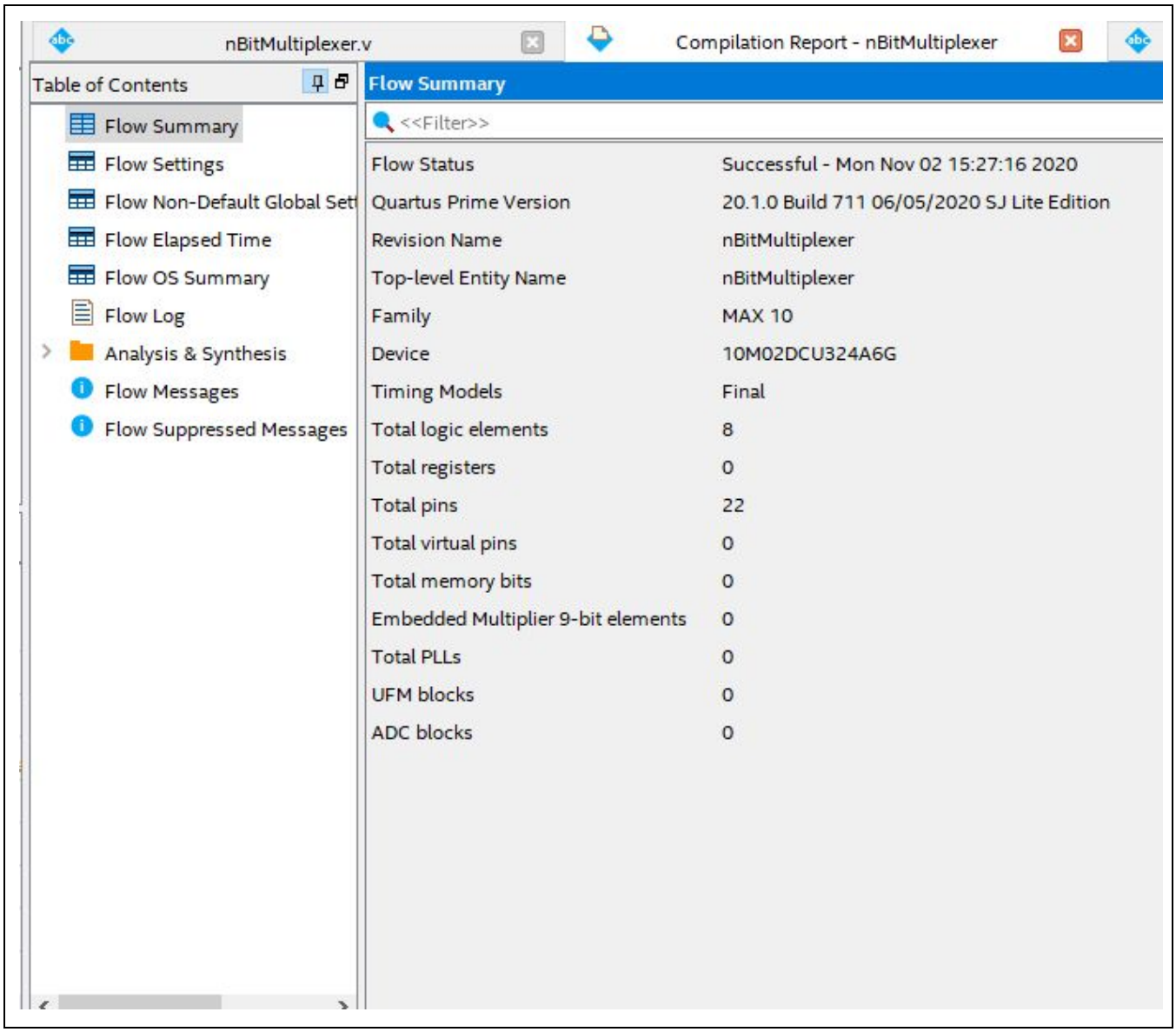
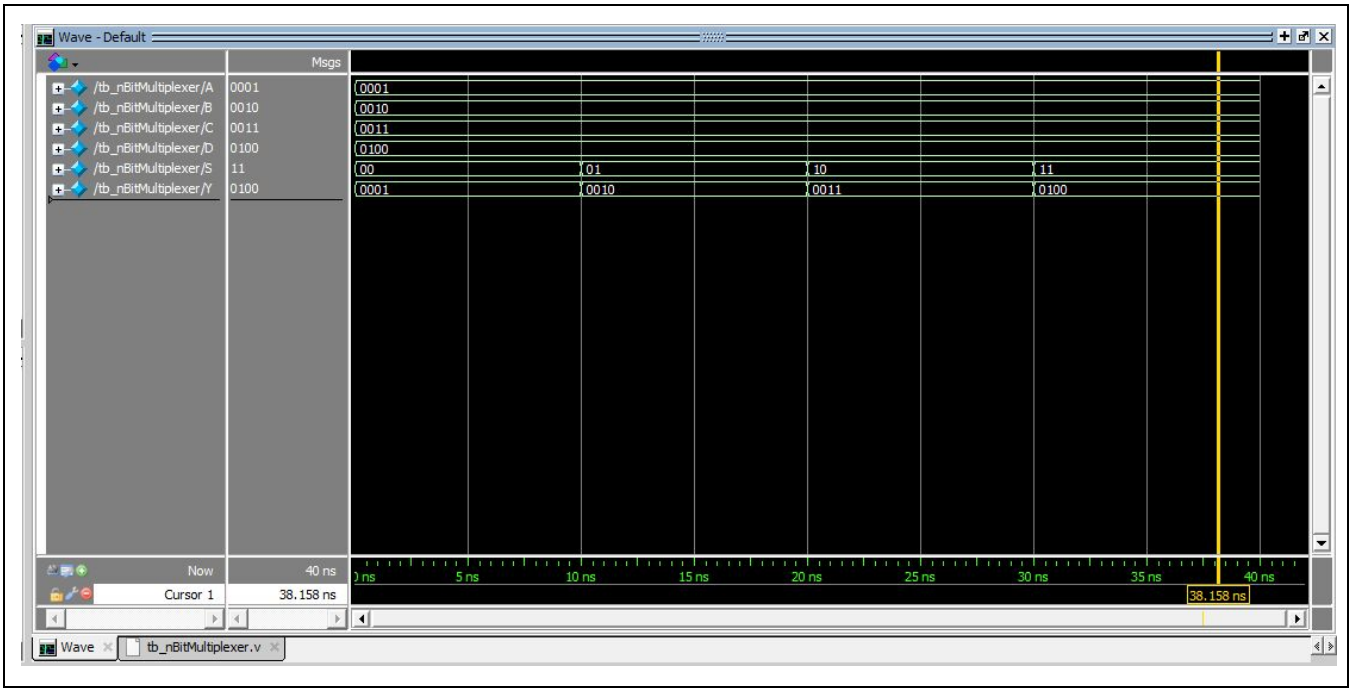


Figure 3c. Test Bench Design Entry for 4-bit Mux

nBitMultiplexer.v | Compilation Report - nBitMultiplexer | tb_nBitMultiplexer.v

```
1   //TestBench File for nBit Multiplexer
2
3   `timescale 1 ns / 1 ps
4   module tb_nBitMultiplexer();
5
6      reg [3:0]A;
7      reg [3:0]B;
8      reg [3:0]C;
9      reg [3:0]D;
10     reg [1:0]S;
11
12     wire [3:0]Y;
13
14     nBitMultiplexer UUT (A, B, C, D, S, Y);
15
16     initial
17        begin
18           A = 4'b0001; B = 4'b0010; C = 4'b0011; D = 4'b0100; S = 2'b00; #10
19           A = 4'b0001; B = 4'b0010; C = 4'b0011; D = 4'b0100; S = 2'b01; #10
20           A = 4'b0001; B = 4'b0010; C = 4'b0011; D = 4'b0100; S = 2'b10; #10
21           A = 4'b0001; B = 4'b0010; C = 4'b0011; D = 4'b0100; S = 2'b11; #10
22
23        $stop;
24     end
25
26  endmodule
27
```

Figure 3d. Timing Diagram for 4-bit Mux



Part 2

This exercise is just similar to part 1. The n value was changed to 8 so that it would be an 8-bit circuit. The Y and selector will still be the same since it still has the same 4 inputs and only the 2 bits of the S is important to pick what will output to Y. Since this is an 8-bit circuit, the values of A, B, C, and D will be changed to an 8 digit binary value. Those are all the changes and differences from part 1.

Figure 4a. Design Entry for 8-bit Mux

nBitMultiplexer.v | Compilation Report - nBitMultiplexer | tb_nBitMultiplexer.v

```
1  //GERMAN E FELISARTA III 16101002   CpE3101L Group 3
2
3  module nBitMultiplexer(A, B, C, D, S, Y);
4
5     localparam n = 8;
6
7        input [n-1:0]A;
8        input [n-1:0]B;
9        input [n-1:0]C;
10       input [n-1:0]D;
11       input [1:0]S;
12       output [n-1:0]Y;
13
14    assign Y = (S == 2'b00) ? A : ( (S == 2'b01) ? B : ( (S == 2'b10) ? C : ( (S == 2'b11) ? D : 0) ) );
15
16 endmodule
17
```

Figure 4b. Flow Summary for 8-bit Mux

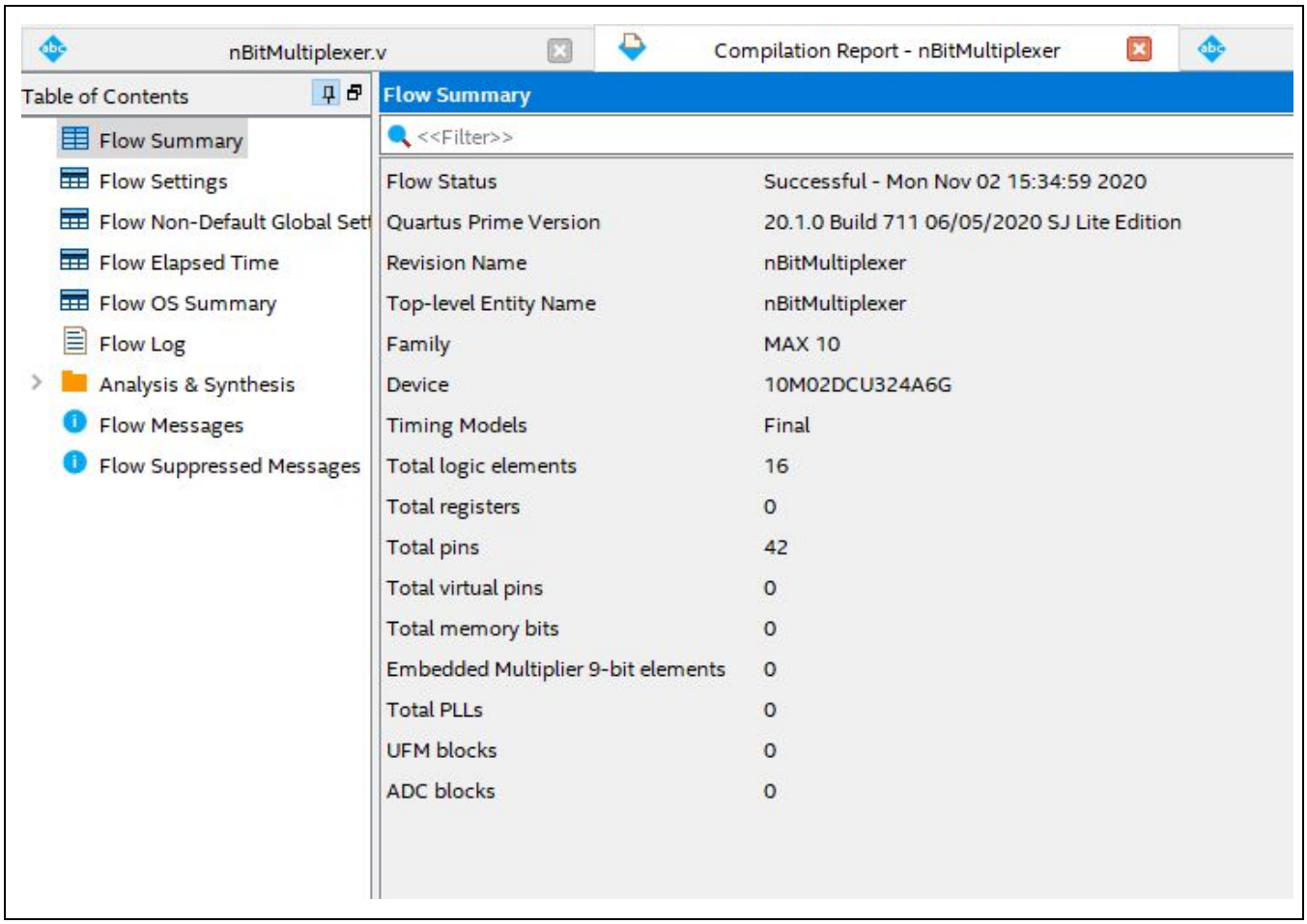


Figure 4c. Test Bench Design Entry for 8-bit Mux

nBitMultiplexer.v | Compilation Report - nBitMultiplexer | tb_nBitMultiplexer.v

```
1   //TestBench File for nBit Multiplexer
2
3   `timescale 1 ns / 1 ps
4   module tb_nBitMultiplexer();
5
6      reg [3:0]A;
7      reg [3:0]B;
8      reg [3:0]C;
9      reg [3:0]D;
10     reg [1:0]S;
11
12     wire [3:0]Y;
13
14     nBitMultiplexer UUT (A, B, C, D, S, Y);
15
16     initial
17        begin
18           A = 8'b0001; B = 8'b0010; C = 8'b0011; D = 8'b0100; S = 3'b000; #10
19           A = 8'b0001; B = 8'b0010; C = 8'b0011; D = 8'b0100; S = 3'b001; #10
20           A = 8'b0001; B = 8'b0010; C = 8'b0011; D = 8'b0100; S = 3'b010; #10
21           A = 8'b0001; B = 8'b0010; C = 8'b0011; D = 8'b0100; S = 3'b011; #10
22           A = 8'b0001; B = 8'b0010; C = 8'b0011; D = 8'b0100; S = 3'b100; #10
23           A = 8'b0001; B = 8'b0010; C = 8'b0011; D = 8'b0100; S = 3'b101; #10
24           A = 8'b0001; B = 8'b0010; C = 8'b0011; D = 8'b0100; S = 3'b110; #10
25           A = 8'b0001; B = 8'b0010; C = 8'b0011; D = 8'b0100; S = 3'b111; #10
26        $stop;
27     end
28
29  endmodule
30
```

Figure 4d. Timing Diagram for 8-bit Mux