Laboratory Report #2

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

Laboratory Exercise Title: Basic Constructs in Verilog HDL **Date Completed: 09-27-2020**

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 2A:

In exercise 2A, that task was to setup the testbench of the halfadder circuit. First, the ModelSim-Altera feature of the Quartus Software was setup, then the design entry was coded to program how the simulation will run, it has 2 inputs that explains the variables x and y which were treated as regs, and 2 outputs, C and S which were treated as wire types. The design entry/code can be found on Figure 1.1 After writing the testbench code, it was added to the group of testbench for the ModelSim-Altera.

After setting-up the testbench, the project was analyzed and synthesized, flow summary can be found in Figure 1.23. Then after, the RTL Simulation with framework ModelSim-Altera was run. In the RTL Simulation, the timing diagram of the outputs were displayed as seen on Figure 1.3.

Figure 1.1. Design Entry of Test Bench

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

   //all inputs from UUT are reg type
   reg x,y;
   //all outputs from UUT are wire type
   wire C,S;
   //instatntiate UUT with implicit port mapping
   HalfAdder UUT(x, y, C,S);

   //generate stimuli
   initial
   begin
      x = 0; y = 0; #10
      x = 0; y = 1; #10
      x = 1; y = 0; #10
      x = 1; y = 1; #50

         $stop;
   end

   endmodule
```

Figure 1.2. Flow Summary

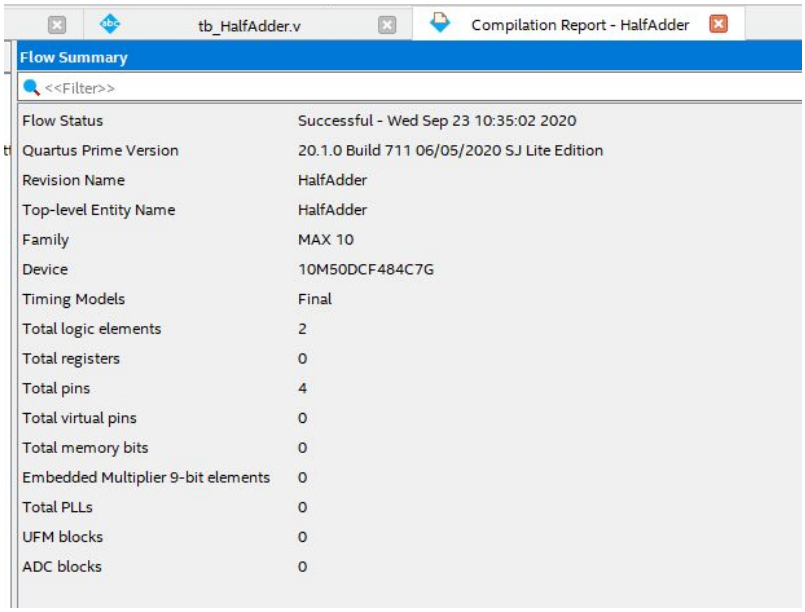
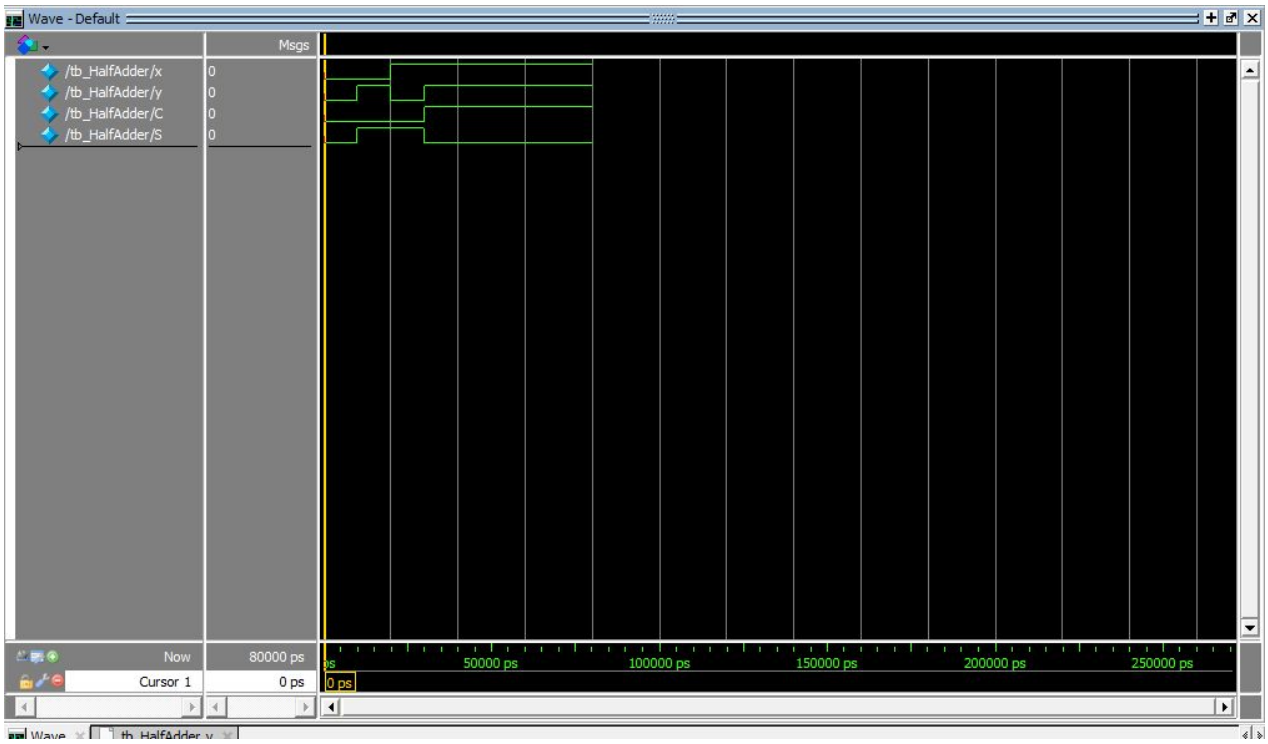


Figure 1.3. Timing Diagram in ModelSim-Altera



Exercise 2B:

In exercise 2A, that task was to create a full adder circuit and make a testbench for it. First, the truth table and output functions were determined. Second, the Verilog Entry was created based on the functions determined. Third, the test bench file was created, following the steps in Exercise 2B. Fourth, the RTL Simulation was run.

Table 2.1 Truth Table for Full Adder Circuit

INPUT			OUTPUT	
A	B	C_in	S	C_out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figure 2.1 K-Maps

S K-Map				
A/B C_in	00	01	11	10
0		1		1
1	1		1	
S = A'B'C_in + A'BC_in' + AB'C_in' + ABC_in **S = C_in XOR (A XOR B)**				

C_out K-Map				
A/B C_in	00	01	11	10
0			1	
1		1	1	1
C_out = BC_in + AC_in + AB				

Figure 2.1. Design Entry of Full Adder

FullAdder.v | tb_FullAdder.v | Compil

```
1  //German E Felisarta III   16101002 Group 3 M 07:30 AM - 10:30 AM
2
3  module FullAdder(A, B, C_in, S, C_out);
4
5     input A, B, C_in;
6     output S, C_out;
7     wire W1;          // S Wires
8     wire W2, W3, W4;     // C_out Wires
9
10
11    //S Function
12    xor X1 (W1, A, B);
13    xor X2 (S, C_in, W1);
14
15    //C_out Function
16    and A1 (W2, B, C_in);
17    and A2 (W3, A, C_in);
18    and A3 (W4, A, B);
19
20    //joining gates for C_out
21    or (C_out, W2, W3, W4);
22
23
24
25 endmodule
26
27
28 /*
29
30 FULL ADDER FUNCTIONS
31
32 S = C_in XOR (A XOR B)
33 C_out = BC_in + AC_in + AB
34
35
36 */
37
```

Figure 2.2. RTL View of Full Adder

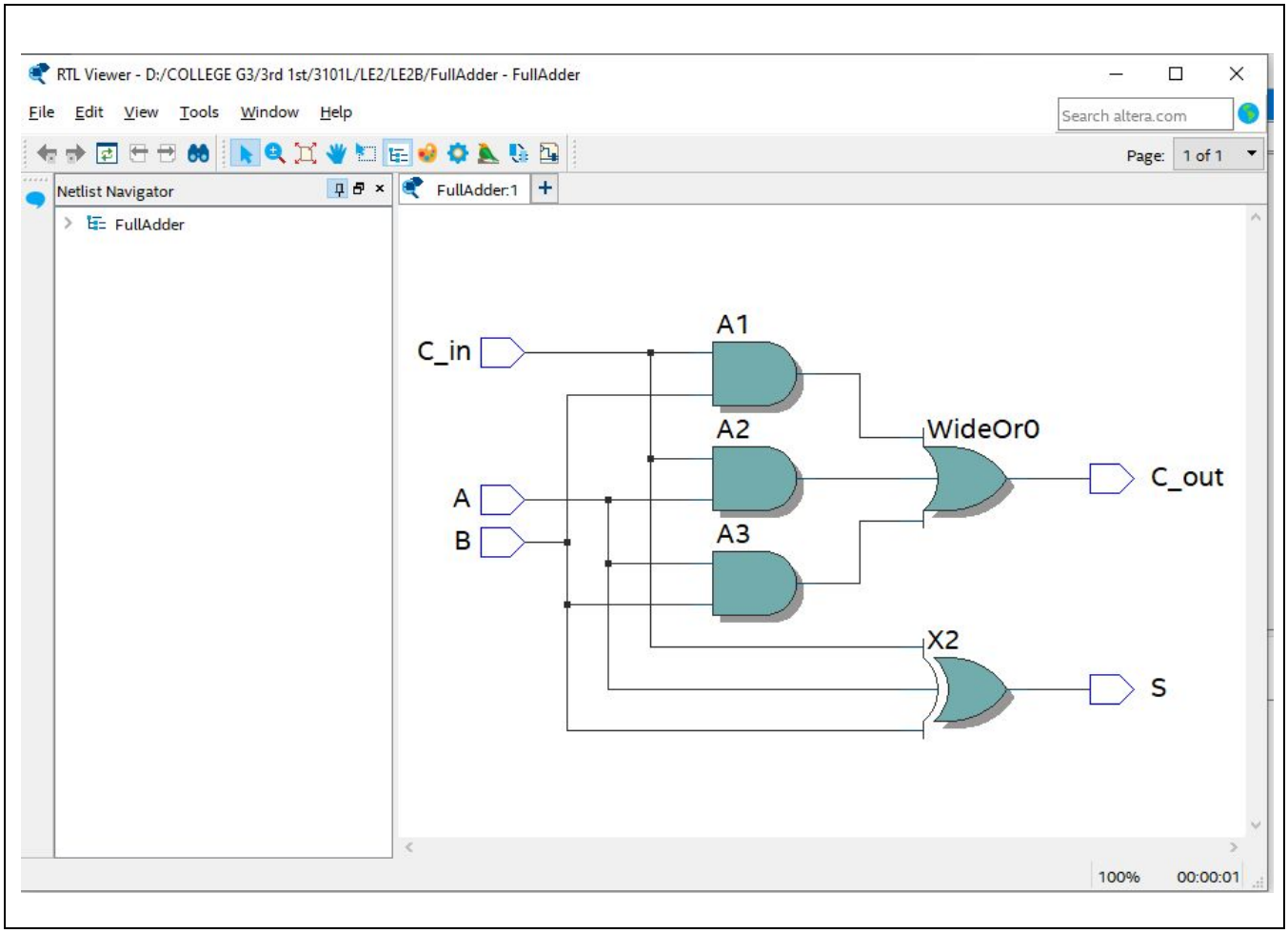


Figure 2.3. Design Entry of Full Adder Testbench

FullAdder.v tb_FullAdder.v

```
1   /*
2      FILE:          tb_FullAdder.v
3      AUTHOR:        German E Felisarta III
4      CLASS:         CpE 3101L
5      GROUP/SCHED:   GROUP#3, M 07:30 AM - 10:30 AM
6      DESCRIPTION:   Testbench file for FullAdder.v
7
8   */
9
10  `timescale 1 ns / 1 ps
11  module tb_FullAdder();
12
13     //all inputs to UUT are declared as reg type
14     reg A, B, C_in;
15     //all outputs from UUT are declares as wire type
16
17     //instantiate UUT with implicit port mapping
18     FullAdder UUT (A, B, C_in, S, C_out);
19
20     //generate stimuli
21     initial
22     begin
23        A = 0; B = 0; C_in = 0; #10
24        A = 0; B = 0; C_in = 1; #10
25        A = 0; B = 1; C_in = 0; #10
26        A = 0; B = 1; C_in = 1; #10
27        A = 1; B = 0; C_in = 0; #10
28        A = 1; B = 0; C_in = 1; #10
29        A = 1; B = 1; C_in = 0; #10
30        A = 1; B = 1; C_in = 1; #30
31
32           $stop;
33     end
34
35  endmodule
36
```

Figure 2.4. Flow Summary

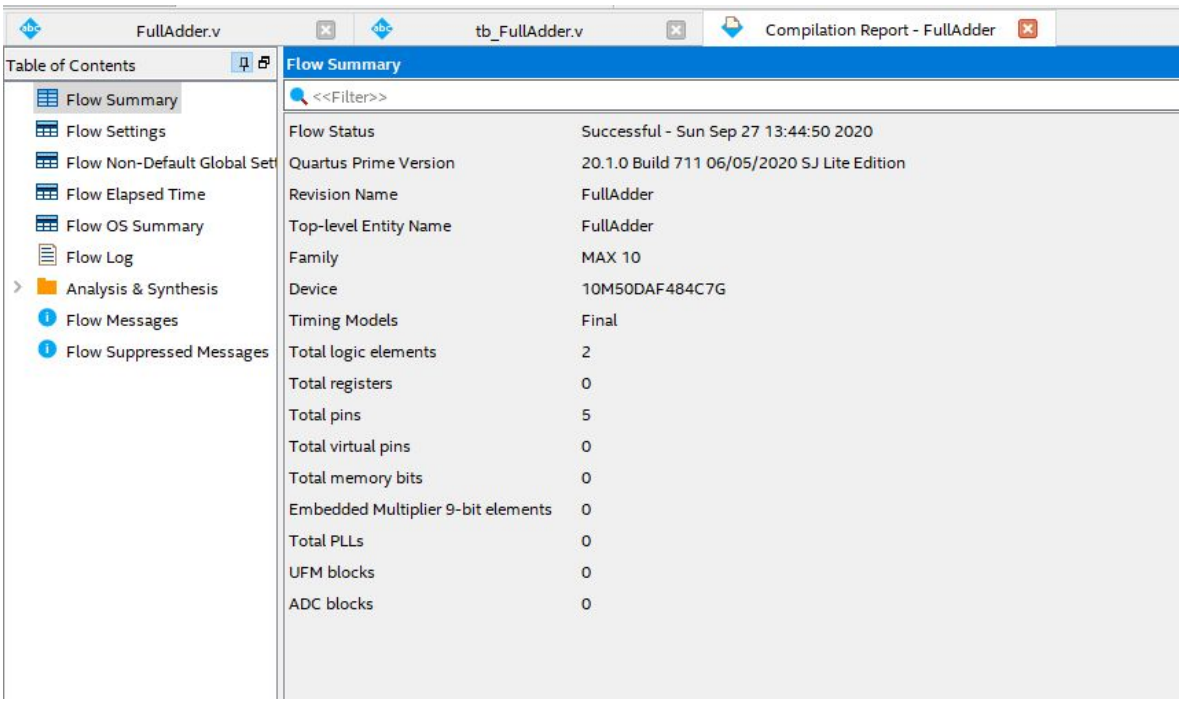
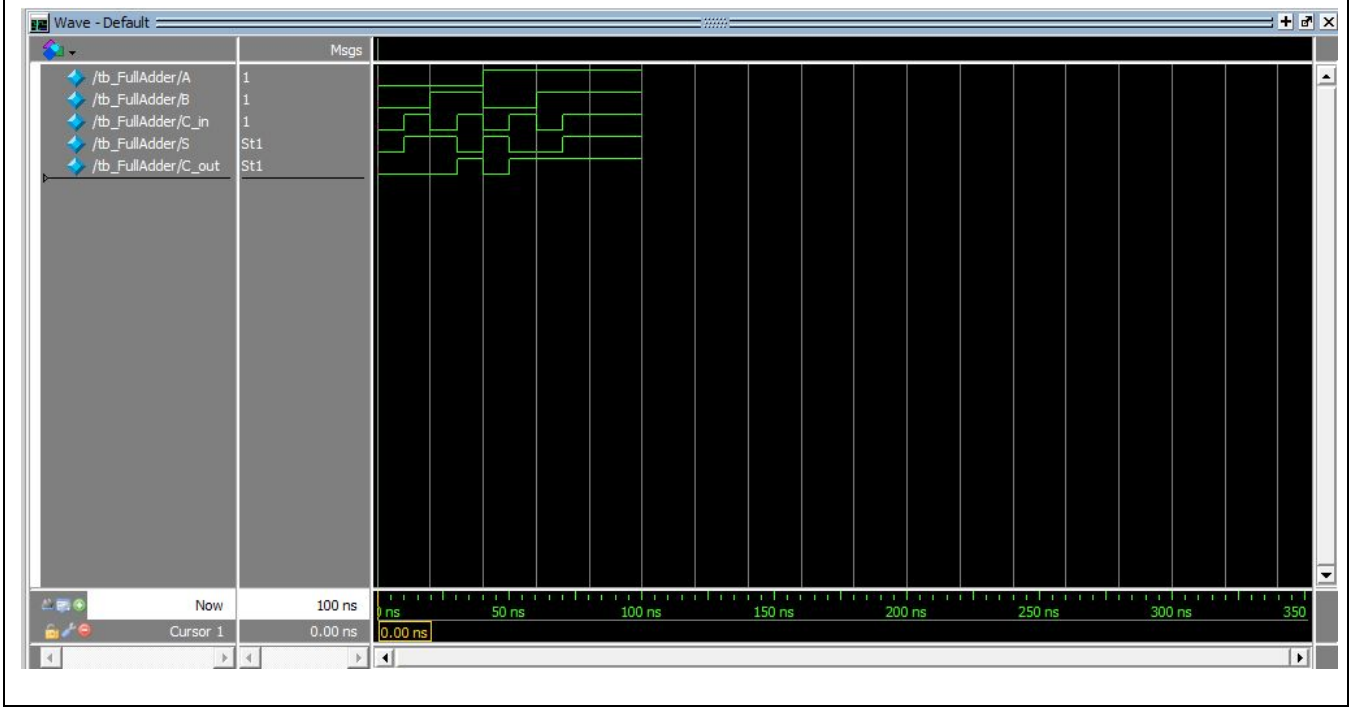


Figure 2.5. Timing Diagram in ModelSim-Altera



Exercise 2C:

In exercise 2C, that task was to create a 4 Bit adder circuit using the Full Adder Circuit created in Exercise 2B and make a testbench for it. First, another module called Adder_4Bit was made. Second, the Verilog Entry was based on the schematic diagram in the lab exercise sheet where 4 full adders are connected with the C_Out and C_In ports. Third, the test bench file was created, following the steps in Exercise 2B. Fourth, the RTL Simulation was run.

Figure 3.1. Design Entry of Adder_4Bit

```
1   //German E Felisarta III   16101002 Group 3 M 07:30 AM - 10:30 AM
2
3   module Adder_4Bit(X, Y, CIn, SUM, COut);
4
5      input        [3:0]X;
6      input        [3:0]Y;
7      input          CIn;
8      output     [3:0]SUM;
9      output         COut;
10     wire     W1, W2, W3;
11
12     FullAdder FA1 (X[0], Y[0], CIn, SUM[0], W1);
13     FullAdder FA2 (X[1], Y[1], W1, SUM[1], W2);
14     FullAdder FA3 (X[2], Y[2], W2, SUM[2], W3);
15     FullAdder FA4 (X[3], Y[3], W3, SUM[3], COut);
16
17
18  endmodule
19
20
21  module FullAdder(A, B, C_in, S, C_out);
22
23     input A, B, C_in;
24     output S, C_out;
25     wire W1, W2, W3, W4;
26
27     //S Function
28     xor X1 (W1, A, B);
29     xor X2 (S, C_in, W1);
30
31     //C_out Function
32     and A1 (W2, B, C_in);
33     and A2 (W3, A, C_in);
34     and A3 (W4, A, B);
35
36     //joining gates for C_out
37     or (C_out, W2, W3, W4);
38
39
40
41  endmodule
```

Figure 3.2. RTL View of Adder_4Bit

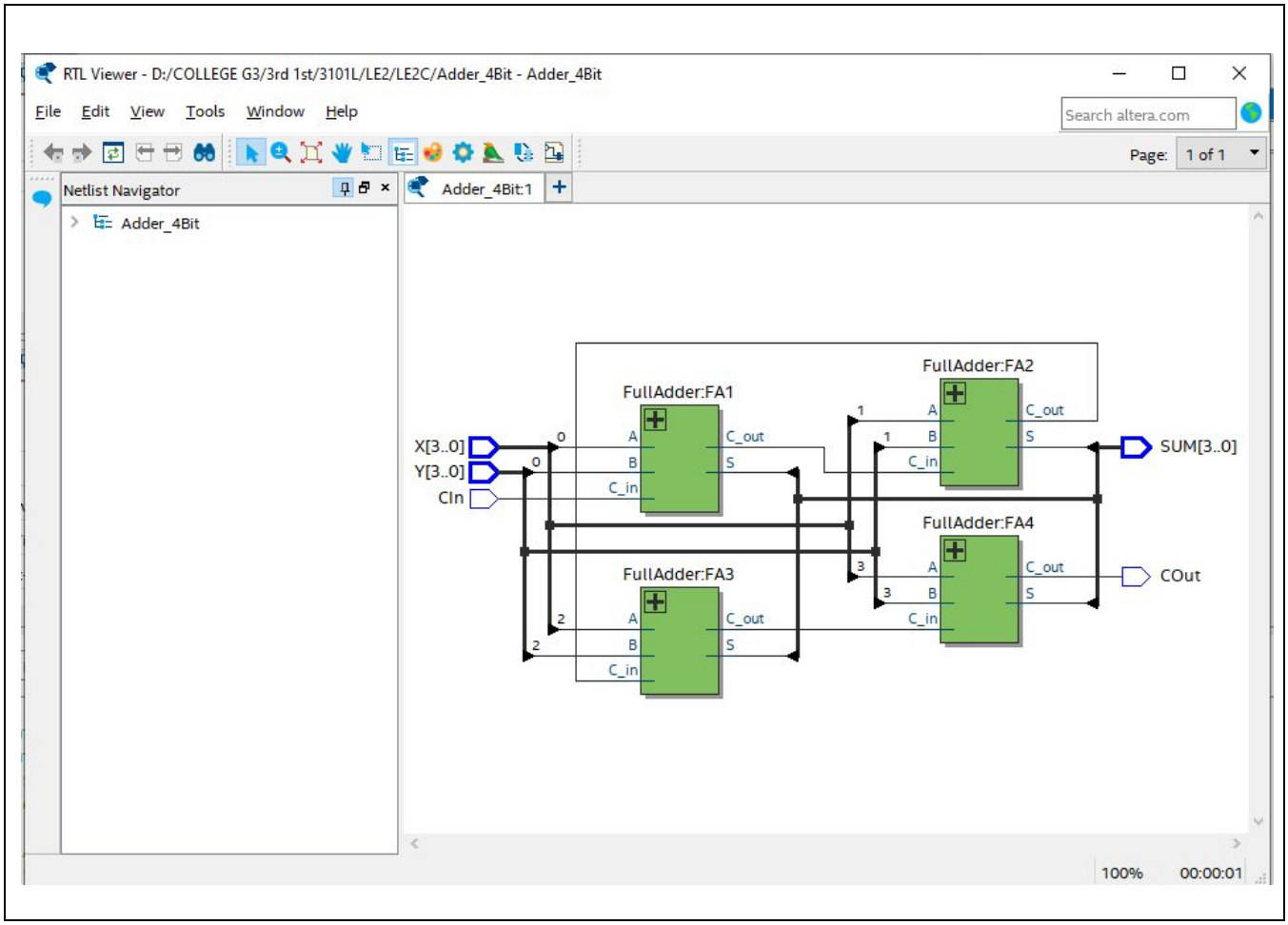


Figure 3.3. Design Entry of Adder_4Bit Testbench

Adder_4Bit.v | Compilation Report - Adder_4Bit | tb_Adder_4Bit.v*

```
/*
   FILE:          tb_Adder_4Bit.v
   AUTHOR:        German E Felisarta III
   CLASS:         CpE 3101L
   GROUP/SCHED:   M 7:30 - 10:30 AM
   DESC:          Testbench file for Adder_4Bit.v

*/
`timescale 1 ns /  1 ps
module tb_Adder_4Bit();

   //all inputs to UUT are declared as reg type
   reg [3:0] A, B;
   reg       C_in;

   //all outputs from UUT are declared as wire type
   wire [3:0] S;
   wire   C_out;

   //instantiate UUT with explicit mapping
   Adder_4Bit UUT (A, B, C_in, S, C_out);

   //generate stimuli
   initial
   begin
      A = 4'd0;   B = 4'd0;   C_in = 0; #10
      A = 4'd3;   B = 4'd8;   C_in = 1; #10
      A = 4'd11;  B = 4'd3;   C_in = 0; #10
      A = 4'd12;  B = 4'd6;   C_in = 0; #10
      A = 4'd5;   B = 4'd4;   C_in = 1; #10
      A = 4'd1;   B = 4'd9;   C_in = 0; #10
      A = 4'd15;  B = 4'd15;  C_in = 0; #10
      A = 4'd15;  B = 4'd15;  C_in = 1; #10

         $stop;
   end
endmodule
```

Figure 3.4. Flow Summary

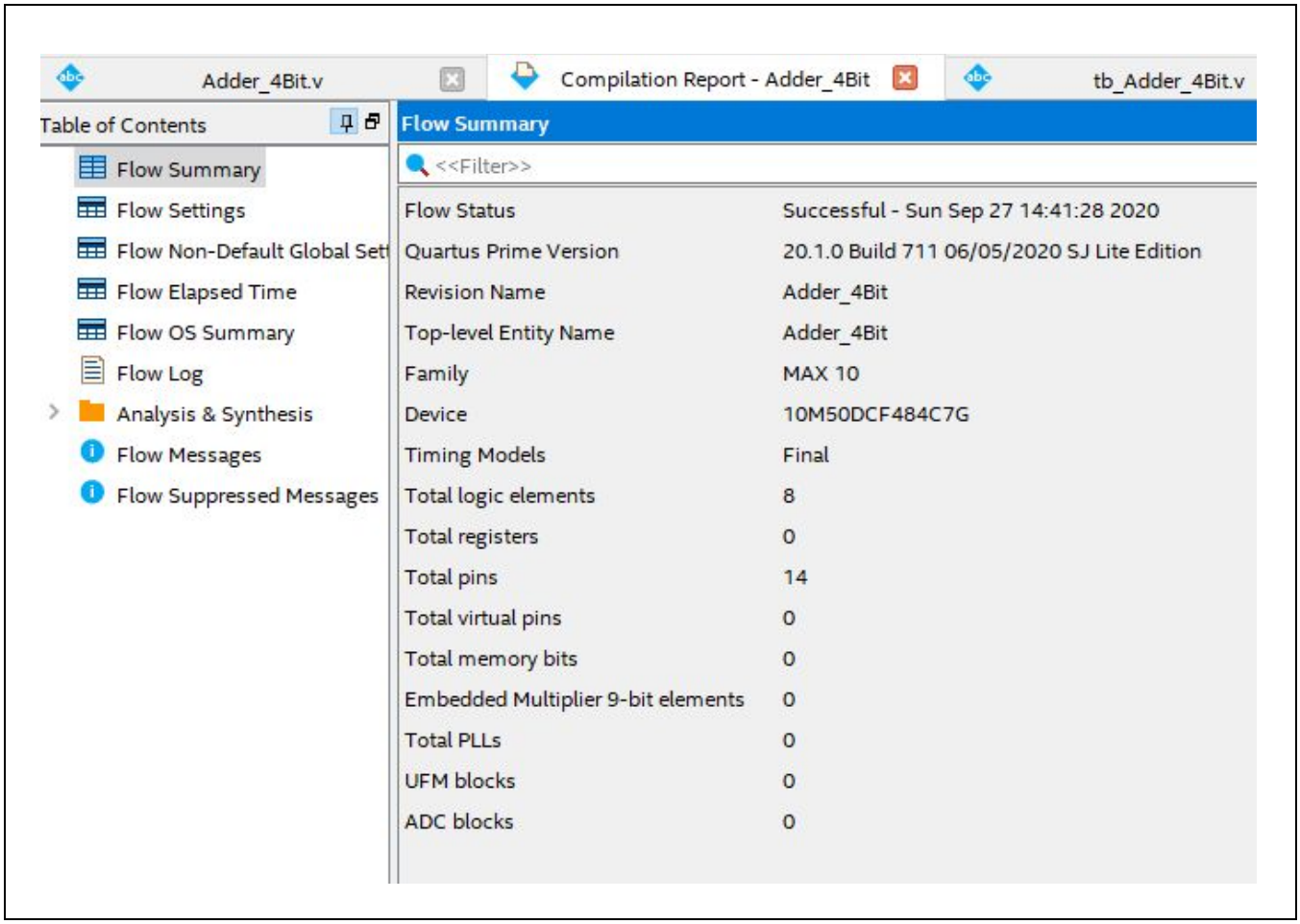


Figure 3.5. Timing Diagram in ModelSim-Altera