Laboratory Report #3

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

Laboratory Exercise Title: Structural Modeling of Combinational Circuits **Date Completed: 10/10/2020**

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 3A: 2x4 Decoder

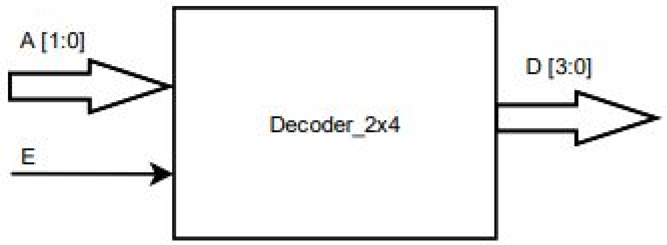


Figure 1.Entity Diagram of 2x4 Decoder

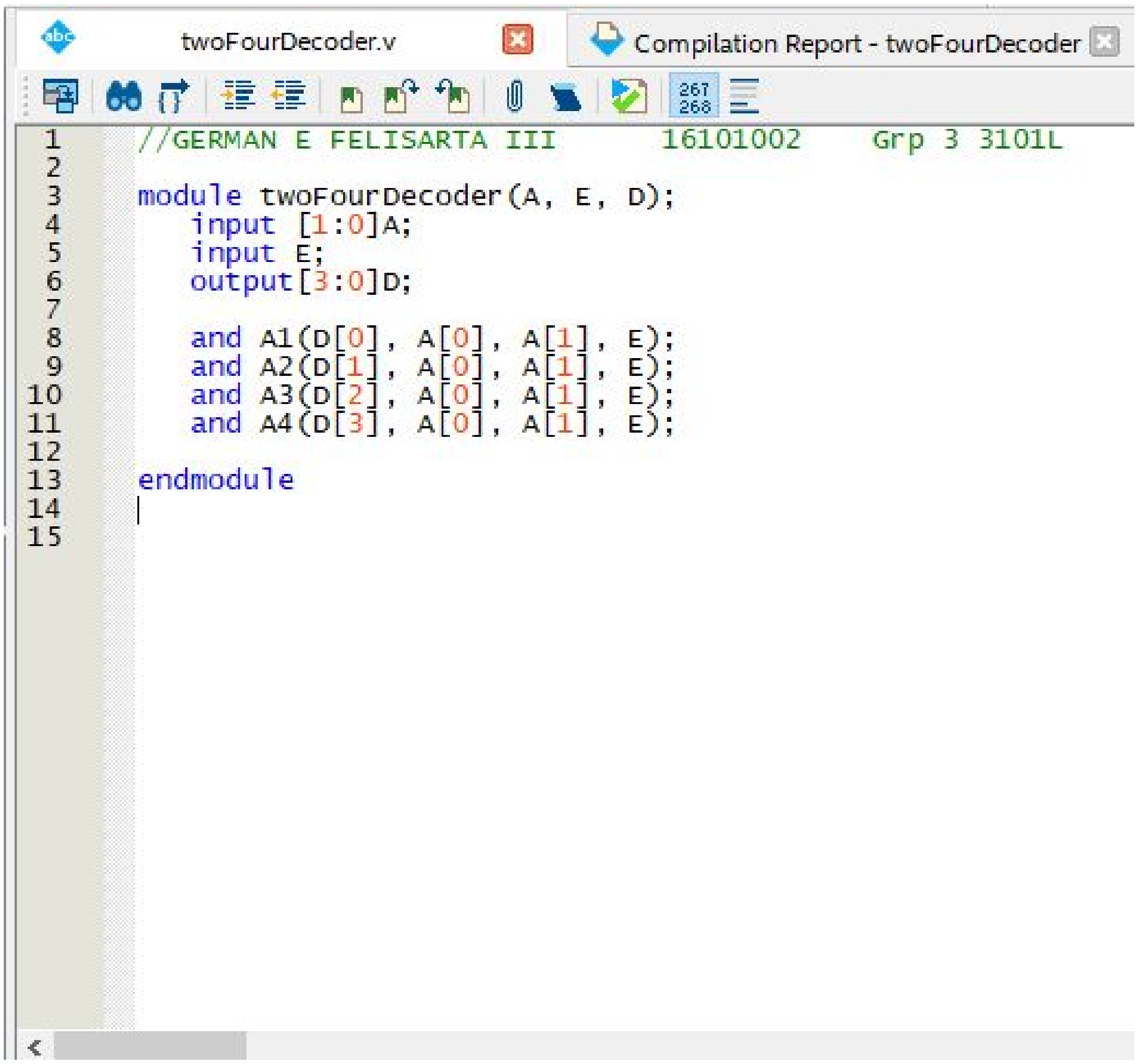
Truth Table:

Inputs		Output
E	**A**	**D**
0	xx	0000
1	00	0001
1	01	0010
1	10	0100
1	11	1000

In this exercise, a 2x4 decoder was remade based on the truth table given above. After deriving the boolean functions, the design entry was made. After testing the design entry, the test bench entry was then created to simulate and check if the truth table remains true.

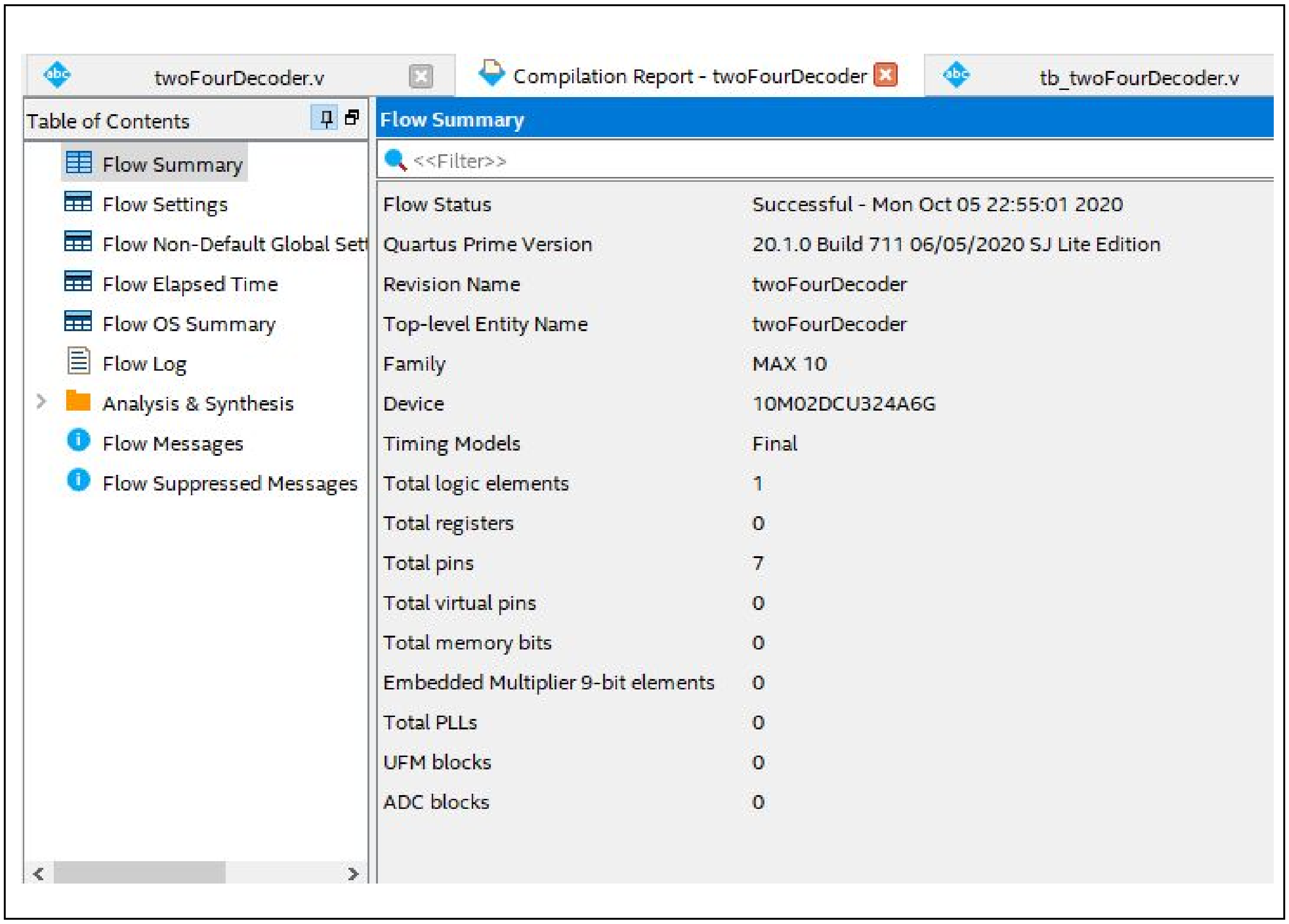
Fig 1.1 Boolean Functions
D_0 = xx $D_1 = E (A_0') A_1'$ $D_2 = E (A_0') A_1$ $D_3 = E (A_0) A_1'$ $D_4 = E (A_0) A_1$

Figure 1.2. Design Entry of 2x4 Decoder



```
//GERMAN E FELISARTA III      16101002    Grp 3 3101L

module twoFourDecoder(A, E, D);
   input [1:0]A;
   input E;
   output[3:0]D;

   and A1(D[0], A[0], A[1], E);
   and A2(D[1], A[0], A[1], E);
   and A3(D[2], A[0], A[1], E);
   and A4(D[3], A[0], A[1], E);

endmodule

```

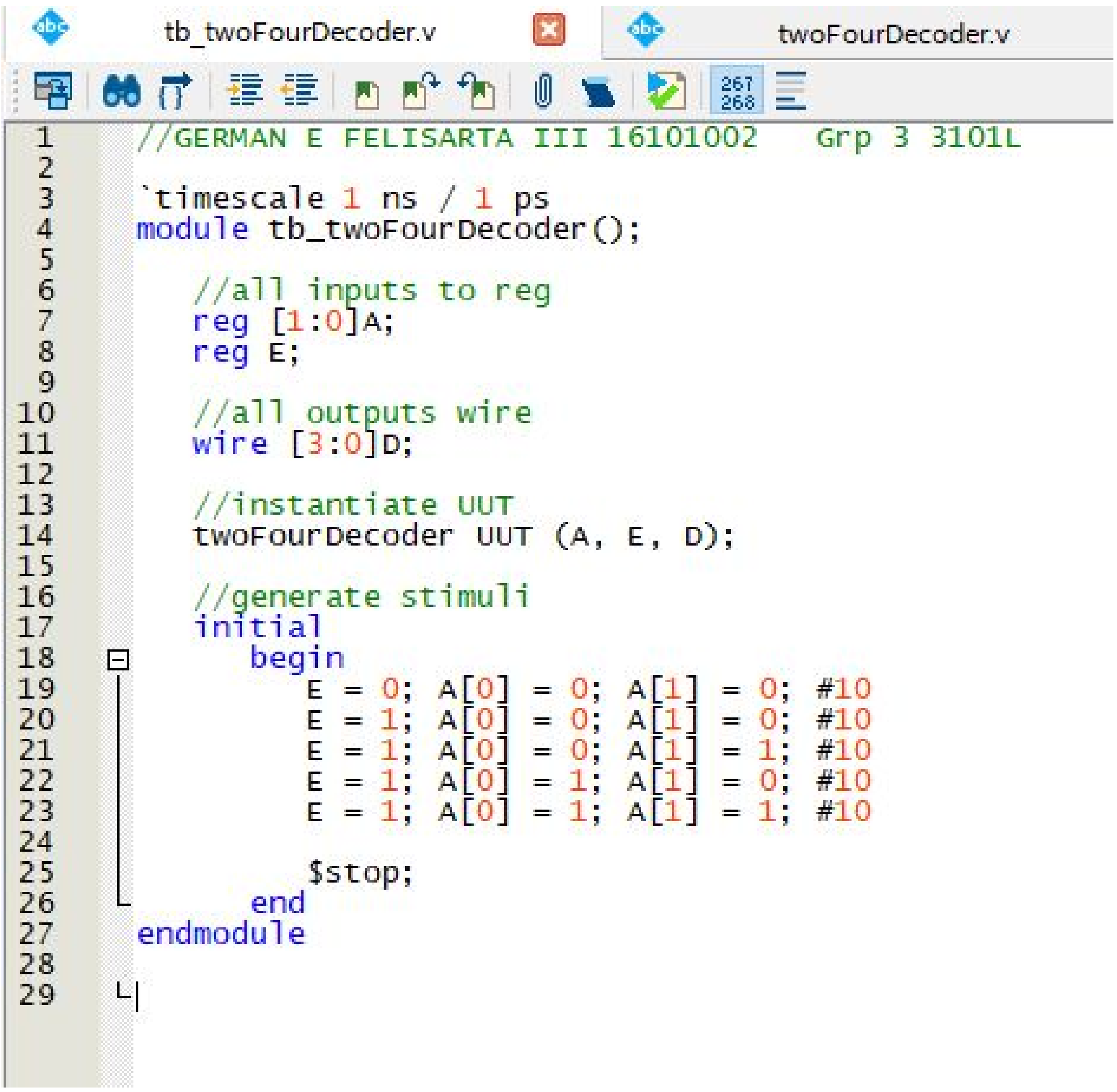
Figure 1.3. Flow Summary



Flow Summary

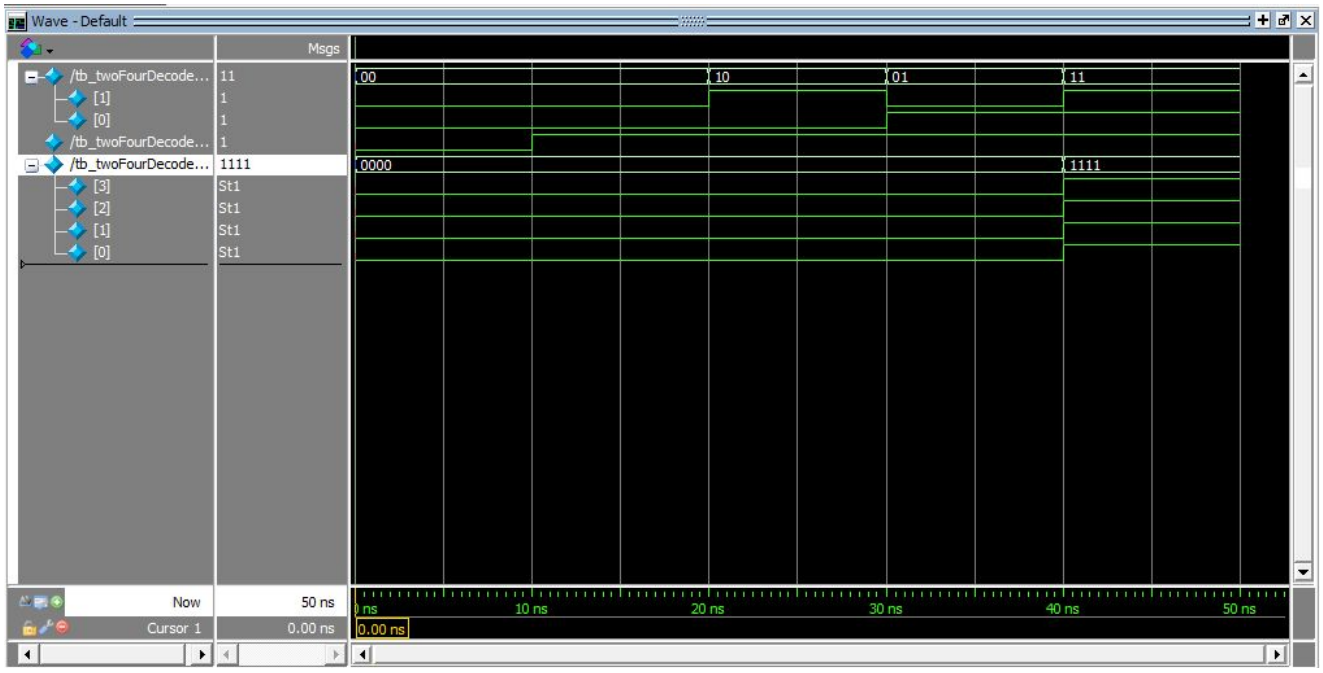
Flow Status	Successful - Mon Oct 05 22:55:01 2020
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	twoFourDecoder
Top-level Entity Name	twoFourDecoder
Family	MAX 10
Device	10M02DCU324A6G
Timing Models	Final
Total logic elements	1
Total registers	0
Total pins	7
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

Figure 1.4. Design Entry of testbench 2x4 Decoder



```
//GERMAN E FELISARTA III 16101002   Grp 3 3101L

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

   //all inputs to reg
   reg [1:0]A;
   reg E;

   //all outputs wire
   wire [3:0]D;

   //instantiate UUT
   twoFourDecoder UUT (A, E, D);

   //generate stimuli
   initial
      begin
         E = 0; A[0] = 0; A[1] = 0; #10
         E = 1; A[0] = 0; A[1] = 0; #10
         E = 1; A[0] = 0; A[1] = 1; #10
         E = 1; A[0] = 1; A[1] = 0; #10
         E = 1; A[0] = 1; A[1] = 1; #10

         $stop;
      end
endmodule

```

Figure 1.5. Timing Diagram of 2x4 Decoder in ModelSim-Altera



Exercise 3B: 3x8 Decoder

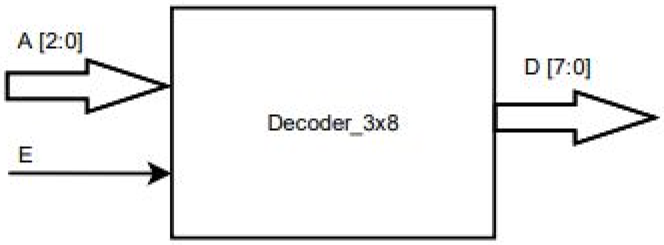


Figure 2. Entity Diagram of 3x8 Decoder

In this exercise, a 3x8 decoder was to be constructed. Based on design entry from Exercise 3A, the 2x8 decoder module was copied over to this exercise's project. A 3x8 decoder with enable can be made with 3 pcs of 2x4 decoders. Since the 1st decoder on the left(found in Fig2.1 and Fig 2.5), only uses two of the outputs, in the design entry was created to not include another 2x4 decoder, instead, it was manually created using AND gates.

Fig 2.1 2x4 equivalent of the 3x8 Decoder

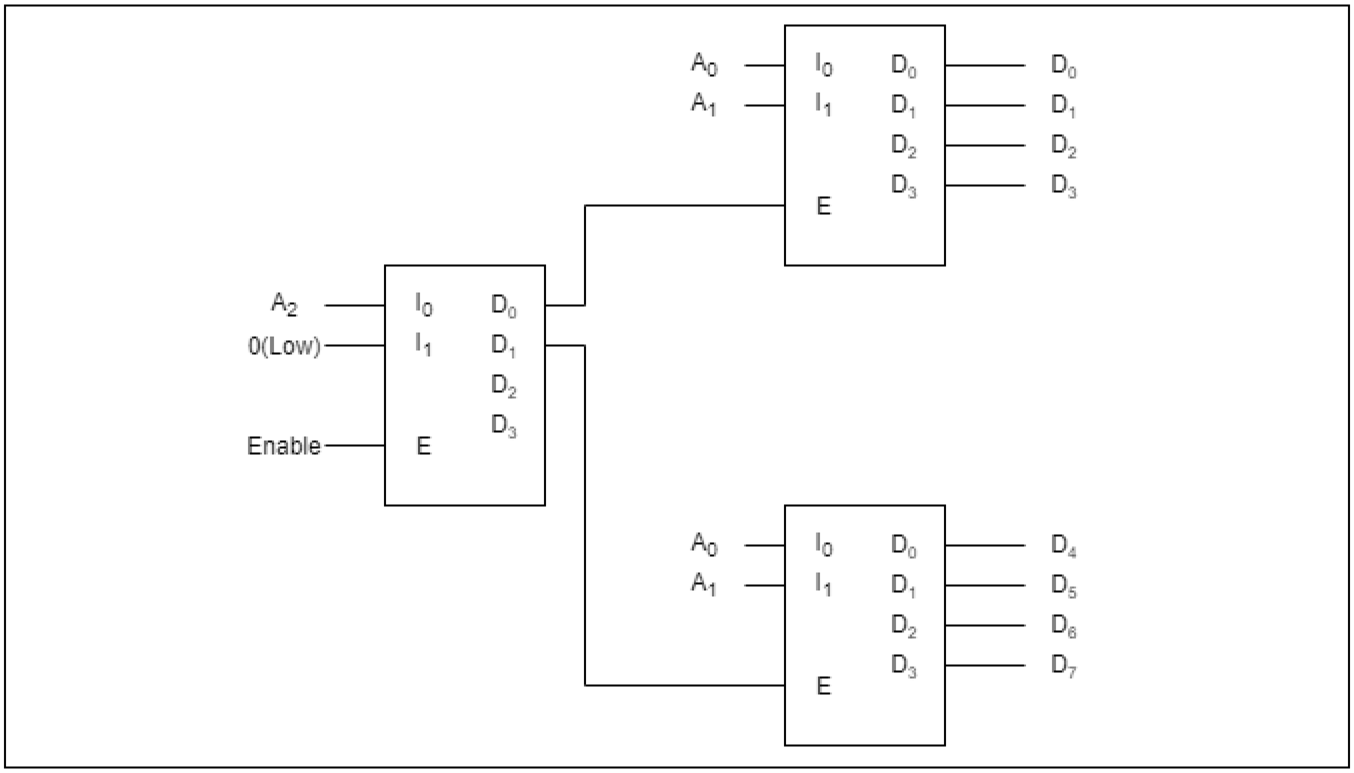
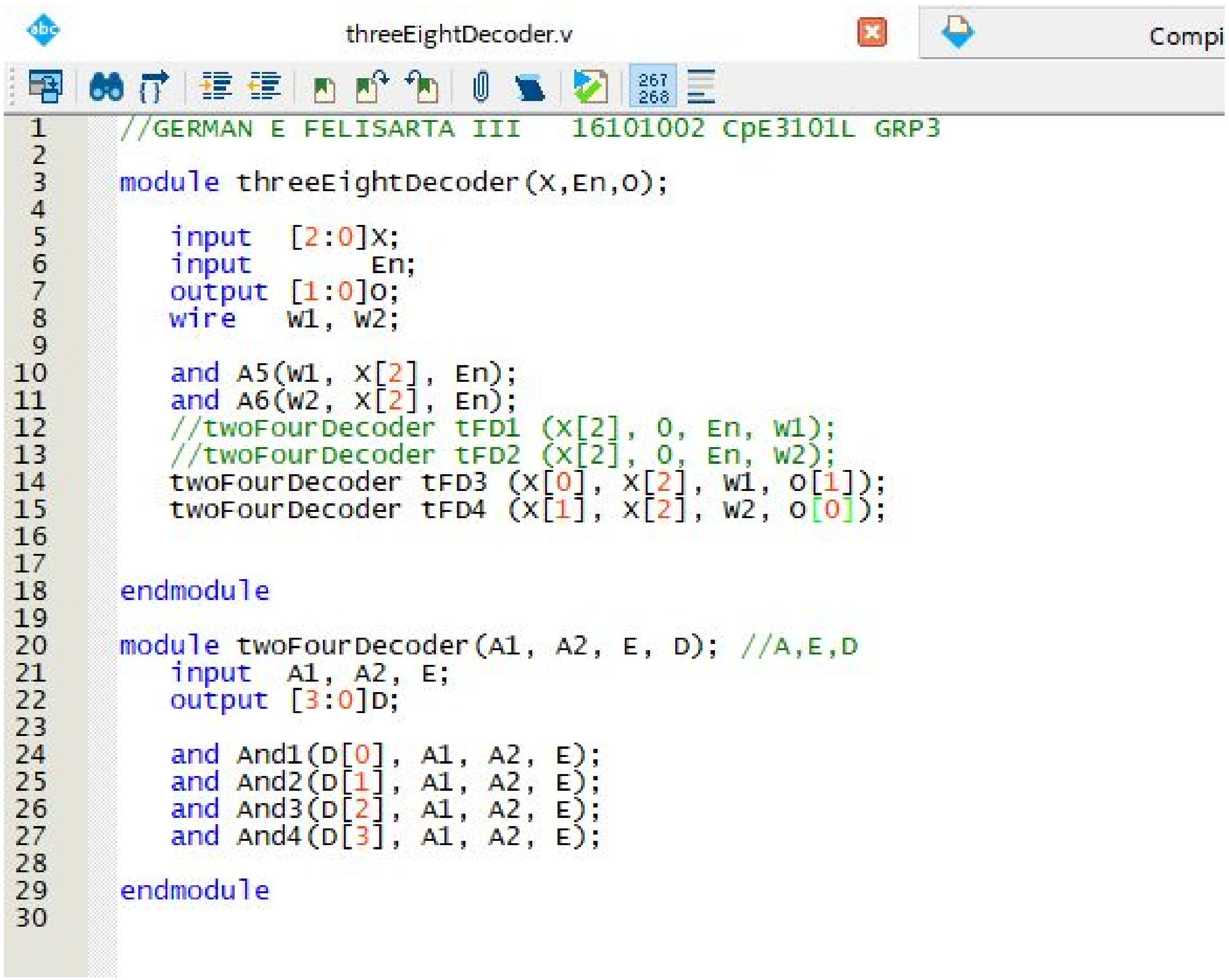


Figure 2.2. Boolean Functions

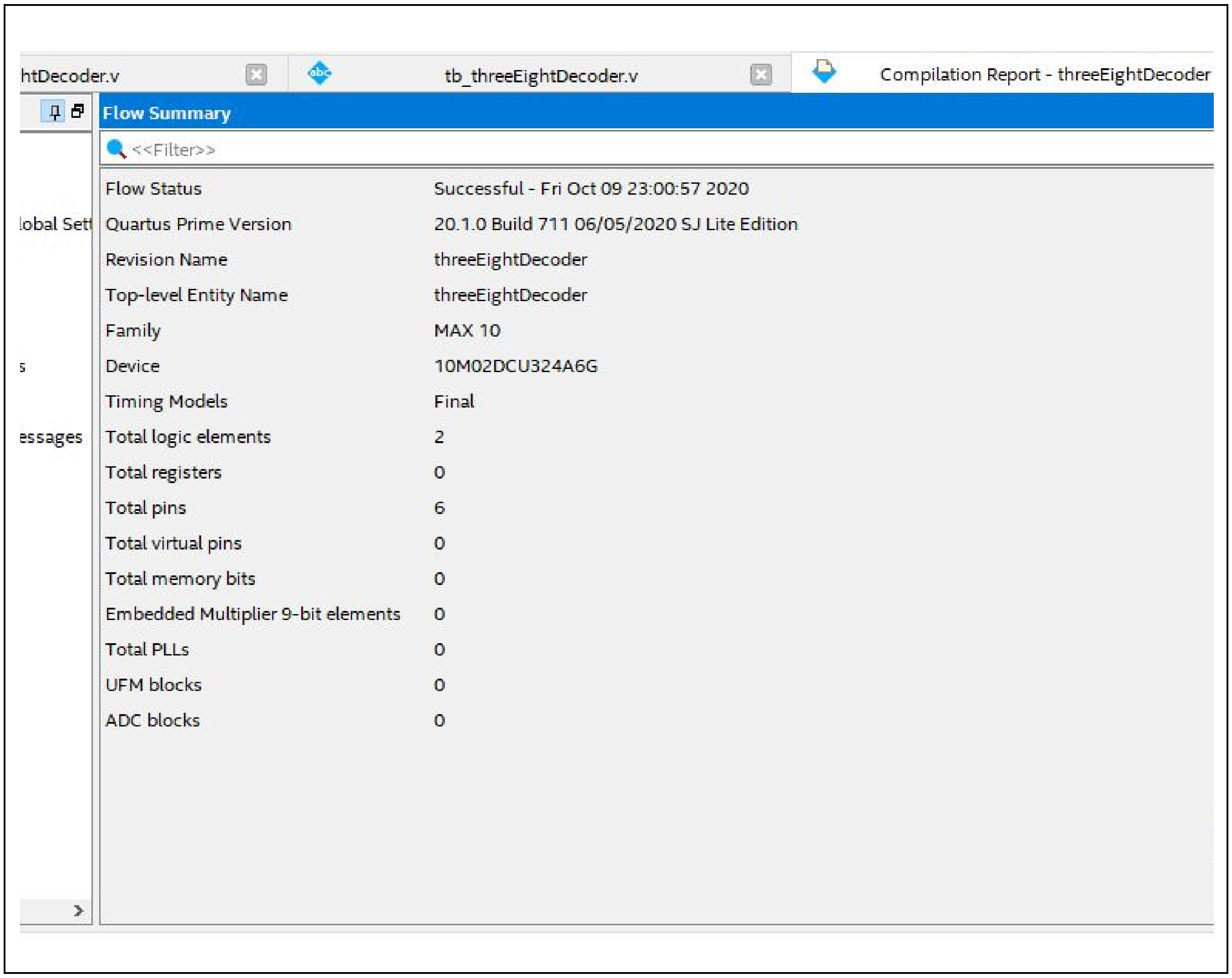
3x8 with A[2] and Enable	3x8 with A[1]	3x8 with A[0]
$D_a = E (A_0') A_1'$ $D_b = E (A_0') A_1$ ~~$D_c = E (A_0) A_1'$~~ ~~$D_d = E (A_0) A_1'$~~	$D_0 = E (A_0') A_1'$ $D_1 = E (A_0') A_1$ $D_2 = E (A_0) A_1'$ $D_3 = E (A_0) A_1'$	$D_4 = E (A_0') A_1'$ $D_5 = E (A_0') A_1$ $D_6 = E (A_0) A_1'$ $D_7 = E (A_0) A_1'$

Figure 2.3. Design Entry of 3x8 Decoder



```
//GERMAN E FELISARTA III   16101002 CpE3101L GRP3

module threeEightDecoder(X,En,O);

   input   [2:0]X;
   input        En;
   output [1:0]O;
   wire   W1, W2;

   and A5(W1, X[2], En);
   and A6(W2, X[2], En);
   //twoFourDecoder tFD1 (X[2], 0, En, W1);
   //twoFourDecoder tFD2 (X[2], 0, En, W2);
   twoFourDecoder tFD3 (X[0], X[2], W1, O[1]);
   twoFourDecoder tFD4 (X[1], X[2], W2, O[0]);


endmodule

module twoFourDecoder(A1, A2, E, D); //A,E,D
   input  A1, A2, E;
   output [3:0]D;

   and And1(D[0], A1, A2, E);
   and And2(D[1], A1, A2, E);
   and And3(D[2], A1, A2, E);
   and And4(D[3], A1, A2, E);

endmodule
```

Figure 2.4. Flow Summary



Flow Status	Successful - Fri Oct 09 23:00:57 2020
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	threeEightDecoder
Top-level Entity Name	threeEightDecoder
Family	MAX 10
Device	10M02DCU324A6G
Timing Models	Final
Total logic elements	2
Total registers	0
Total pins	6
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

Figure 2.5. RTL View

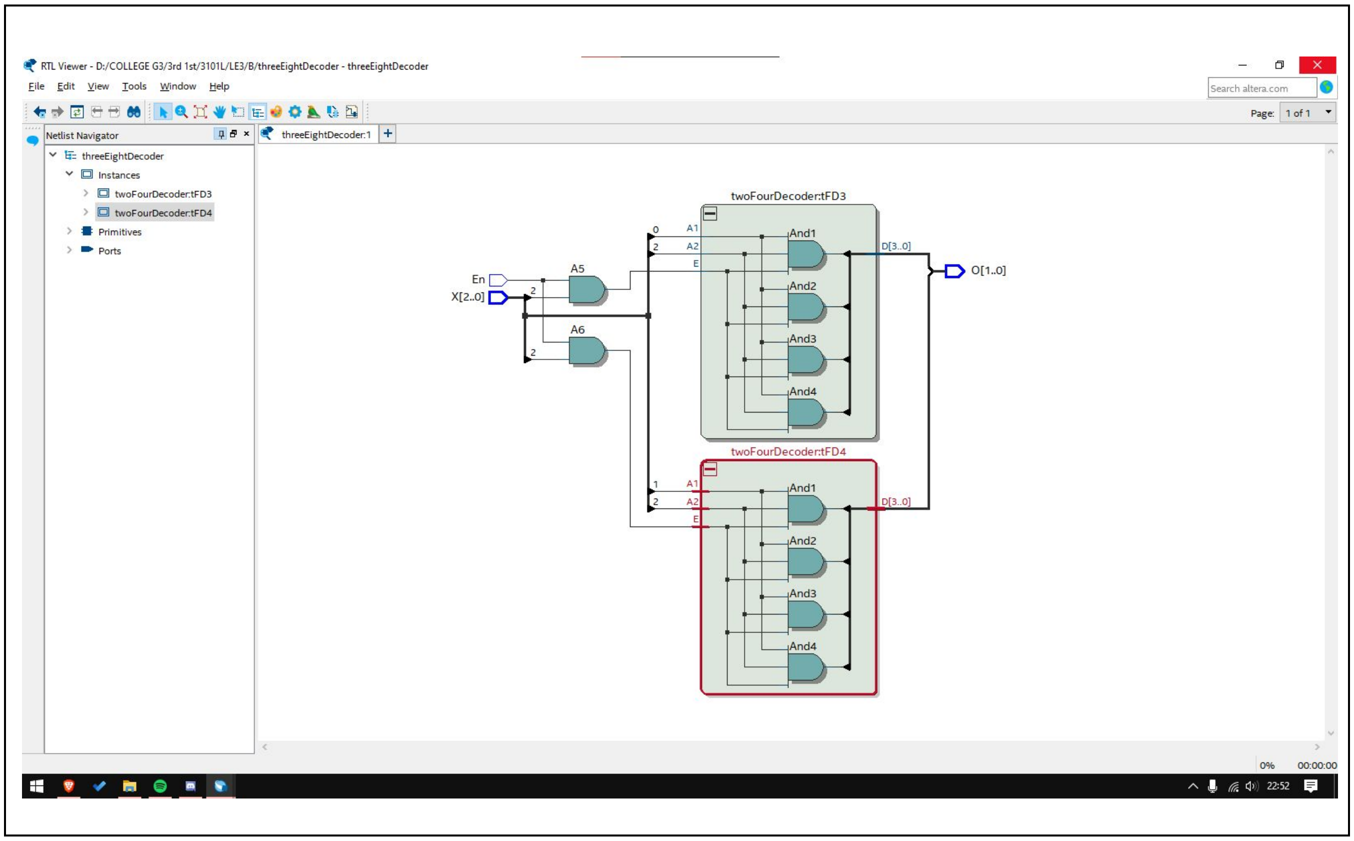
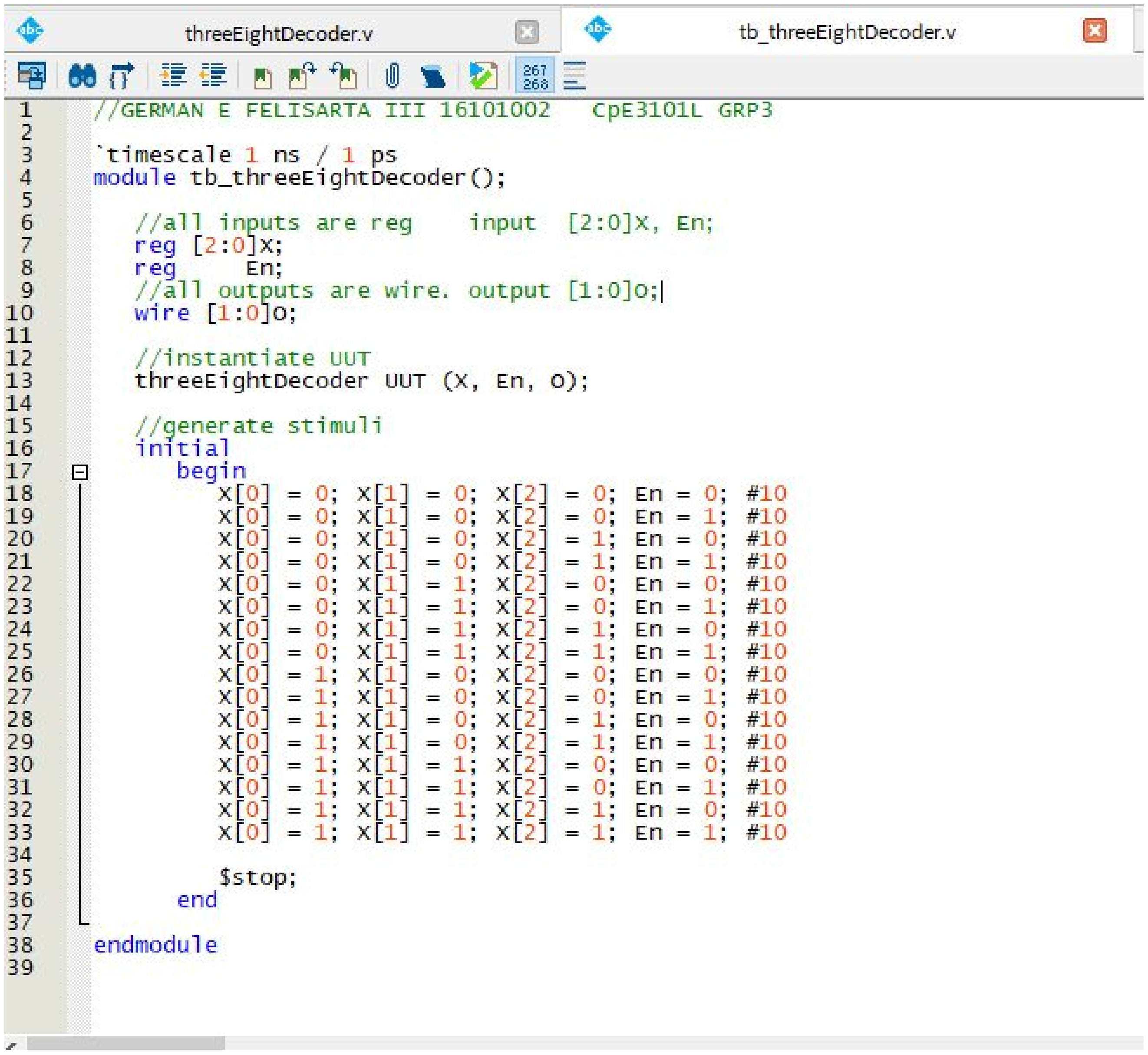
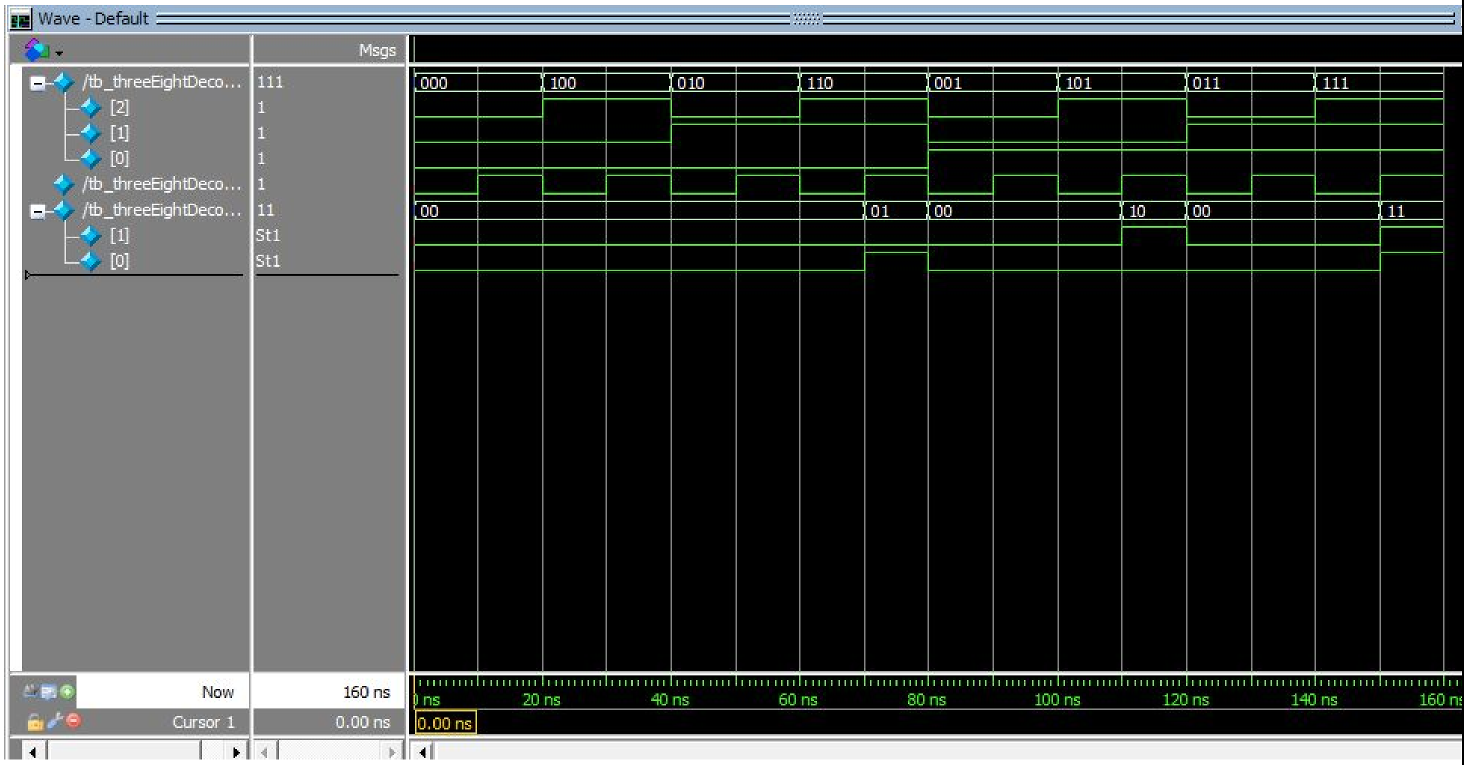


Figure 2.6. Design Entry of testbench 3x8 Decoder


threeEightDecoder.v tb_threeEightDecoder.v

```
//GERMAN E FELISARTA III 16101002   CpE3101L GRP3

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

   //all inputs are reg    input  [2:0]X, En;
   reg [2:0]X;
   reg     En;
   //all outputs are wire. output [1:0]O;
   wire [1:0]O;

   //instantiate UUT
   threeEightDecoder UUT (X, En, O);

   //generate stimuli
   initial
      begin
         X[0] = 0; X[1] = 0; X[2] = 0; En = 0; #10
         X[0] = 0; X[1] = 0; X[2] = 0; En = 1; #10
         X[0] = 0; X[1] = 0; X[2] = 1; En = 0; #10
         X[0] = 0; X[1] = 0; X[2] = 1; En = 1; #10
         X[0] = 0; X[1] = 1; X[2] = 0; En = 0; #10
         X[0] = 0; X[1] = 1; X[2] = 0; En = 1; #10
         X[0] = 0; X[1] = 1; X[2] = 1; En = 0; #10
         X[0] = 0; X[1] = 1; X[2] = 1; En = 1; #10
         X[0] = 1; X[1] = 0; X[2] = 0; En = 0; #10
         X[0] = 1; X[1] = 0; X[2] = 0; En = 1; #10
         X[0] = 1; X[1] = 0; X[2] = 1; En = 0; #10
         X[0] = 1; X[1] = 0; X[2] = 1; En = 1; #10
         X[0] = 1; X[1] = 1; X[2] = 0; En = 0; #10
         X[0] = 1; X[1] = 1; X[2] = 0; En = 1; #10
         X[0] = 1; X[1] = 1; X[2] = 1; En = 0; #10
         X[0] = 1; X[1] = 1; X[2] = 1; En = 1; #10

         $stop;
      end

endmodule
```

Figure 2.7. Timing Diagram of 3x8 Decoder in ModelSim-Altera



Exercise 3C: 4-Bit Majority Function using a 4x16 Decoder

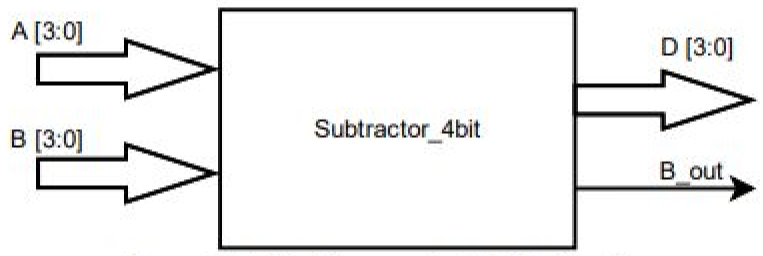


Figure 3. Entity Diagram of a 4-Bit Subtractor

Fig 3.1 4-bit Subtractor using Full Adders

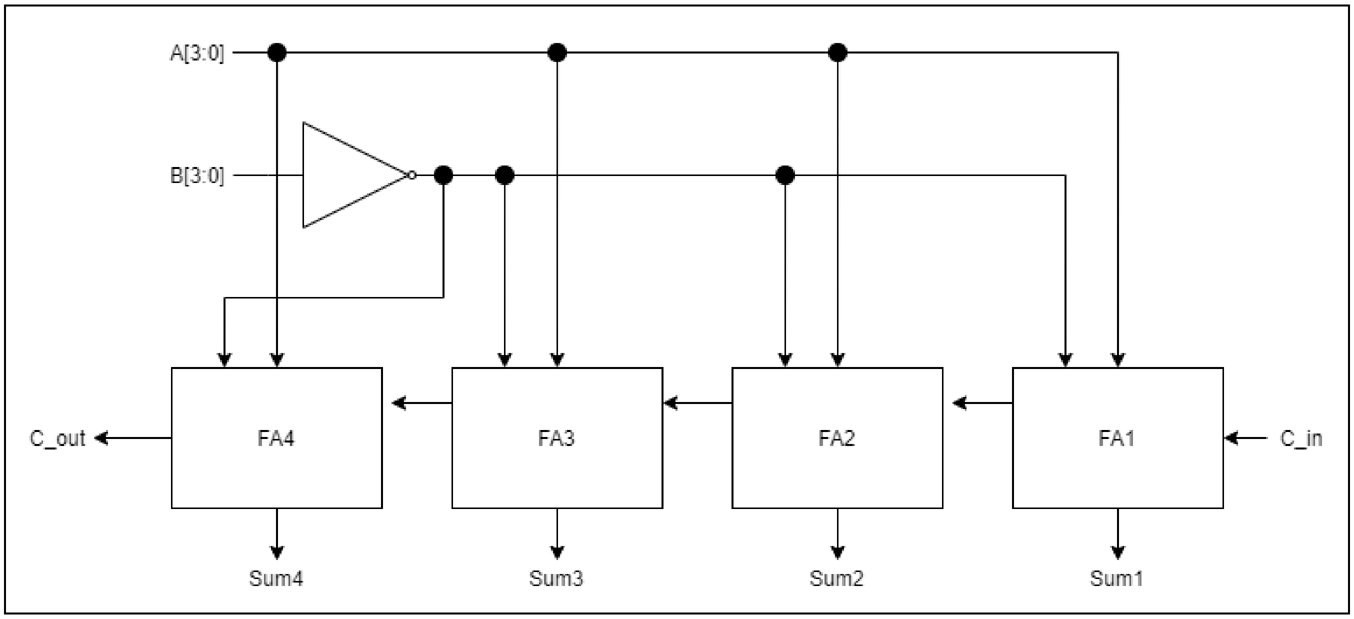
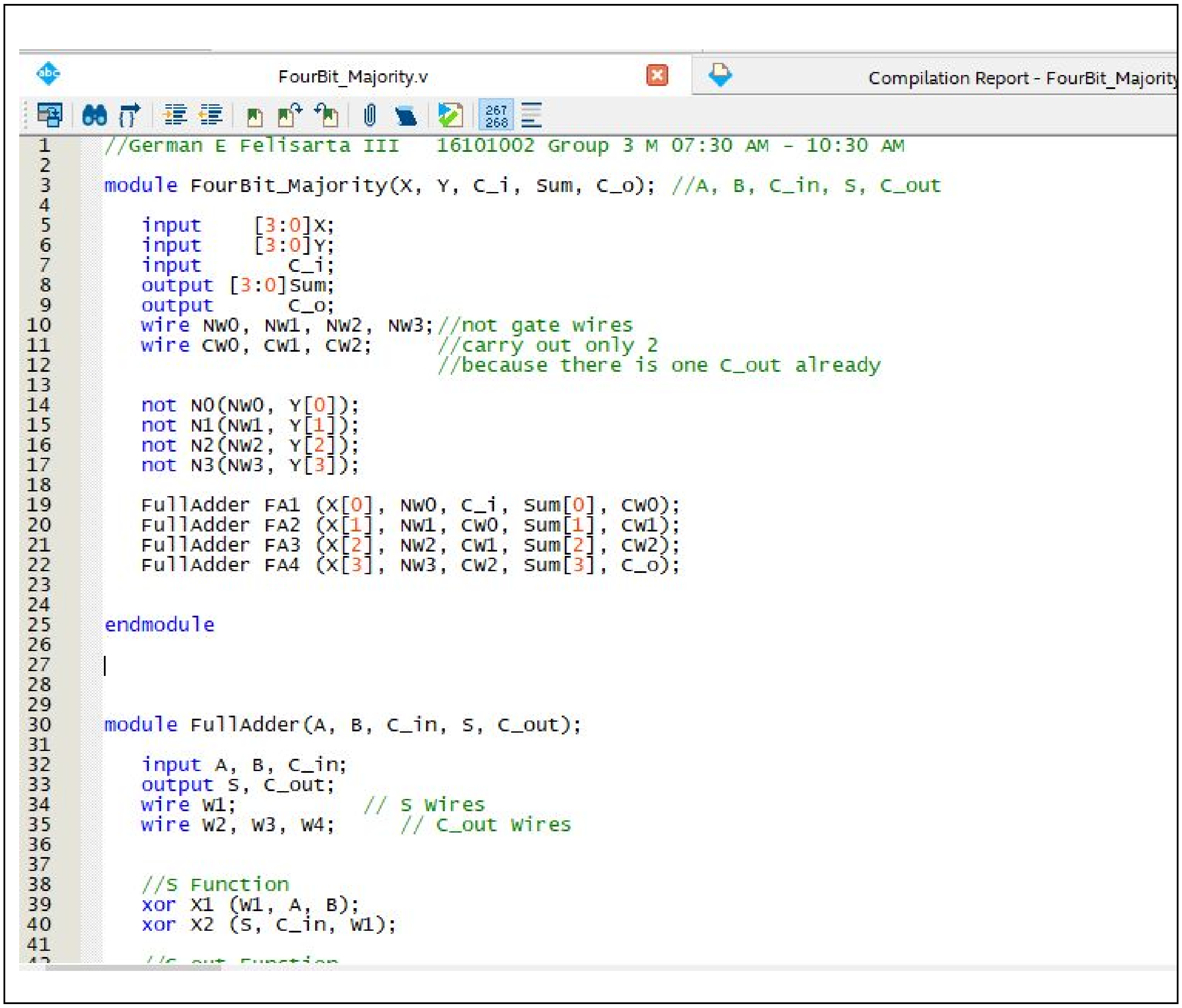


Figure 3.2. Design Entry of Subtractor

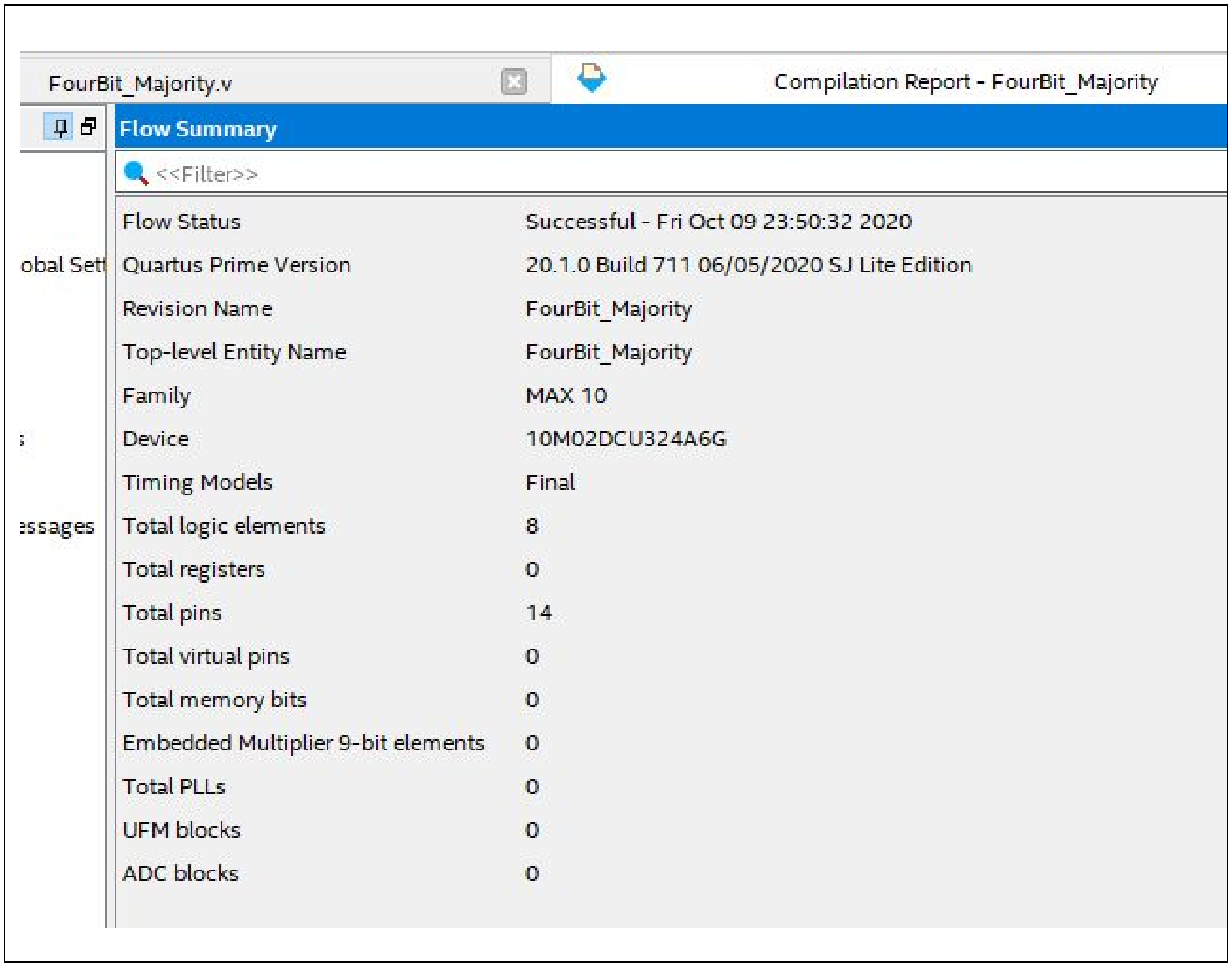


FourBit_Majority.v

Compilation Report - FourBit_Majority

```
//German E Felisarta III   16101002 Group 3 M 07:30 AM - 10:30 AM

module FourBit_Majority(X, Y, C_i, Sum, C_o); //A, B, C_in, S, C_out

   input     [3:0]X;
   input     [3:0]Y;
   input         C_i;
   output [3:0]Sum;
   output        C_o;
   wire NW0, NW1, NW2, NW3;//not gate wires
   wire CW0, CW1, CW2;     //carry out only 2
                           //because there is one C_out already

   not N0(NW0, Y[0]);
   not N1(NW1, Y[1]);
   not N2(NW2, Y[2]);
   not N3(NW3, Y[3]);

   FullAdder FA1 (X[0], NW0, C_i, Sum[0], CW0);
   FullAdder FA2 (X[1], NW1, CW0, Sum[1], CW1);
   FullAdder FA3 (X[2], NW2, CW1, Sum[2], CW2);
   FullAdder FA4 (X[3], NW3, CW2, Sum[3], C_o);


endmodule




module FullAdder(A, B, C_in, S, C_out);

   input A, B, C_in;
   output S, C_out;
   wire W1;            // S Wires
   wire W2, W3, W4;       // C_out Wires


   //S Function
   xor X1 (W1, A, B);
   xor X2 (S, C_in, W1);

   //C_out Function
```

Figure 3.3. Flow Summary



Flow Status	Successful - Fri Oct 09 23:50:32 2020
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	FourBit_Majority
Top-level Entity Name	FourBit_Majority
Family	MAX 10
Device	10M02DCU324A6G
Timing Models	Final
Total logic elements	8
Total registers	0
Total pins	14
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0
UFM blocks	0
ADC blocks	0

Figure 3.4. RTL View

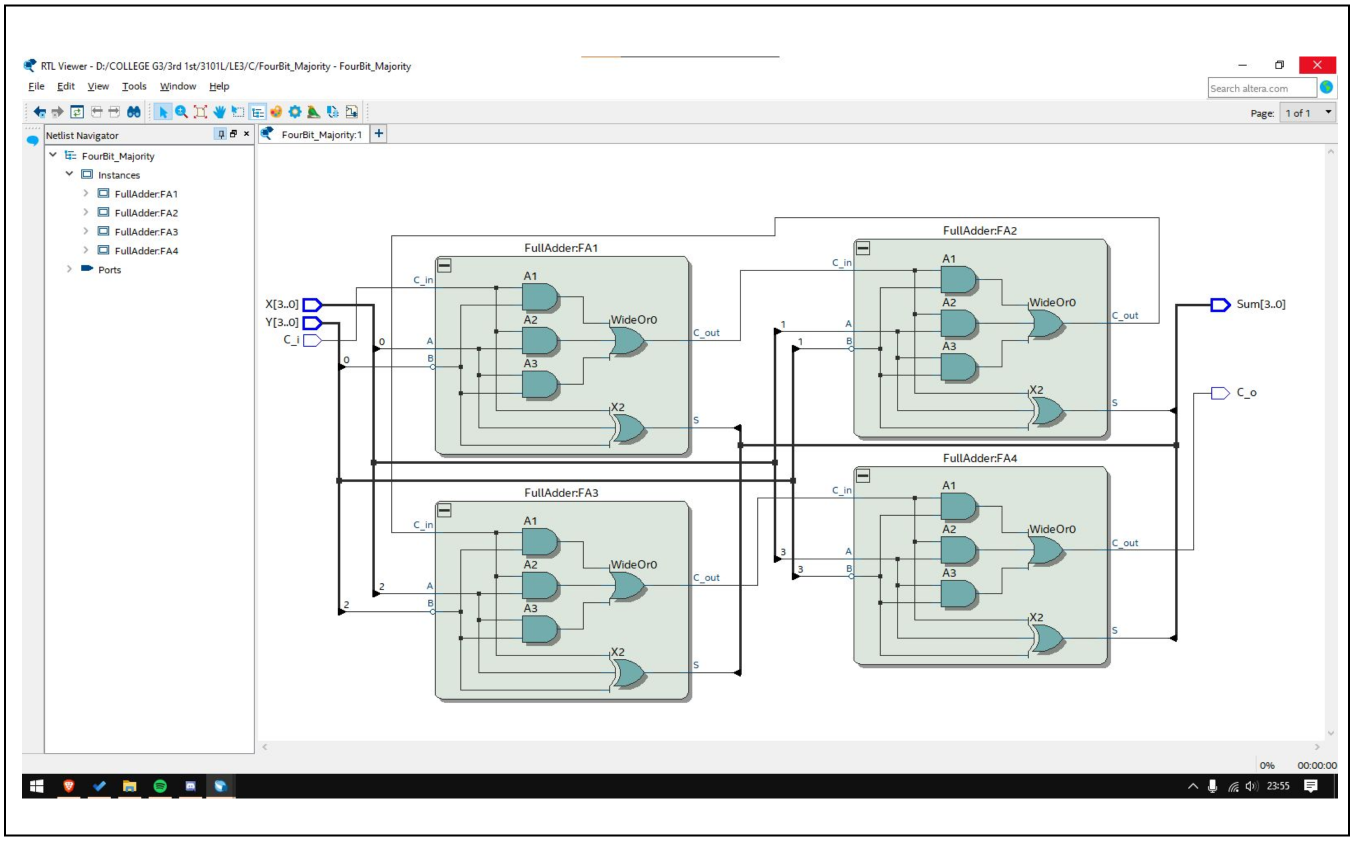
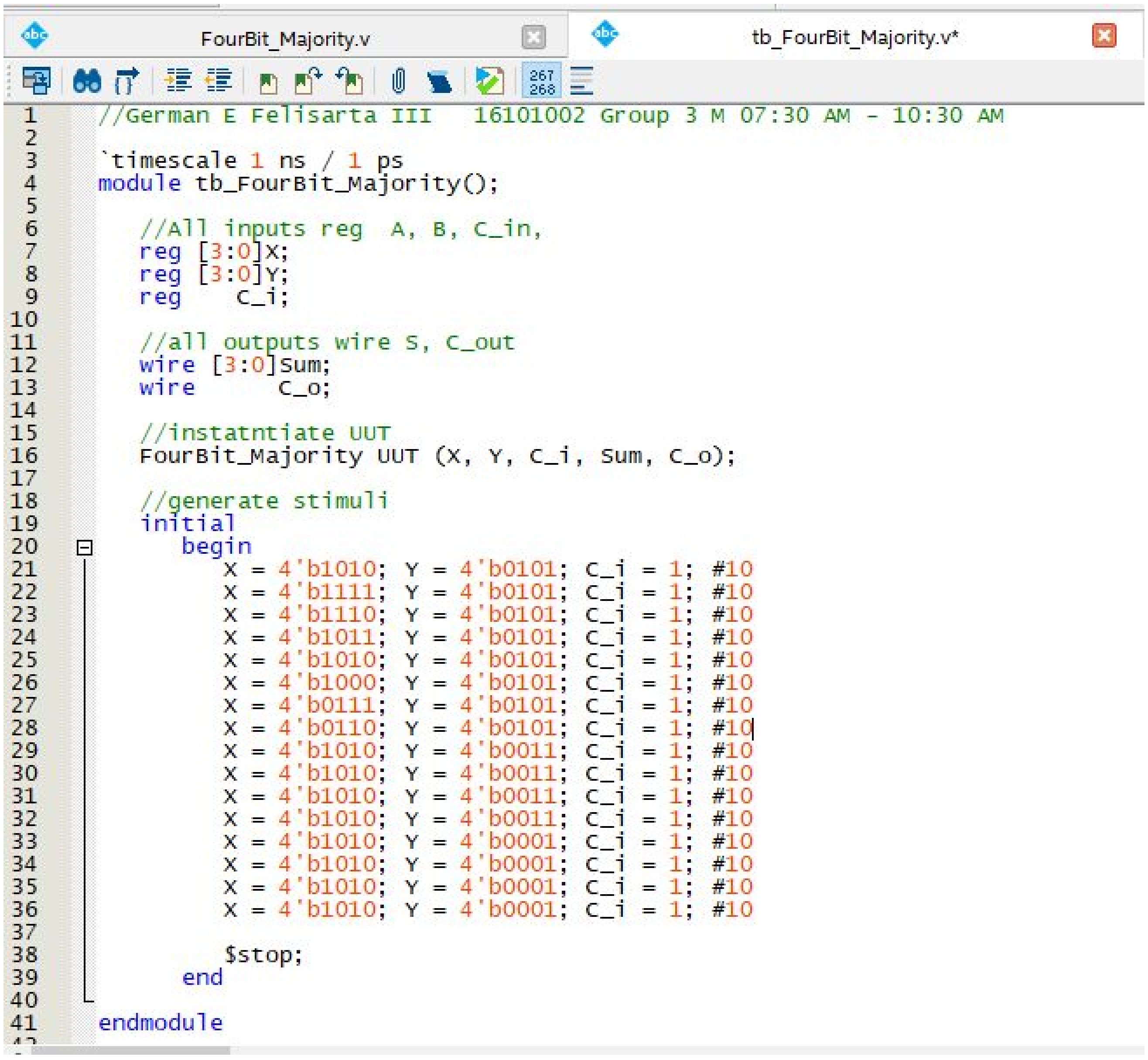
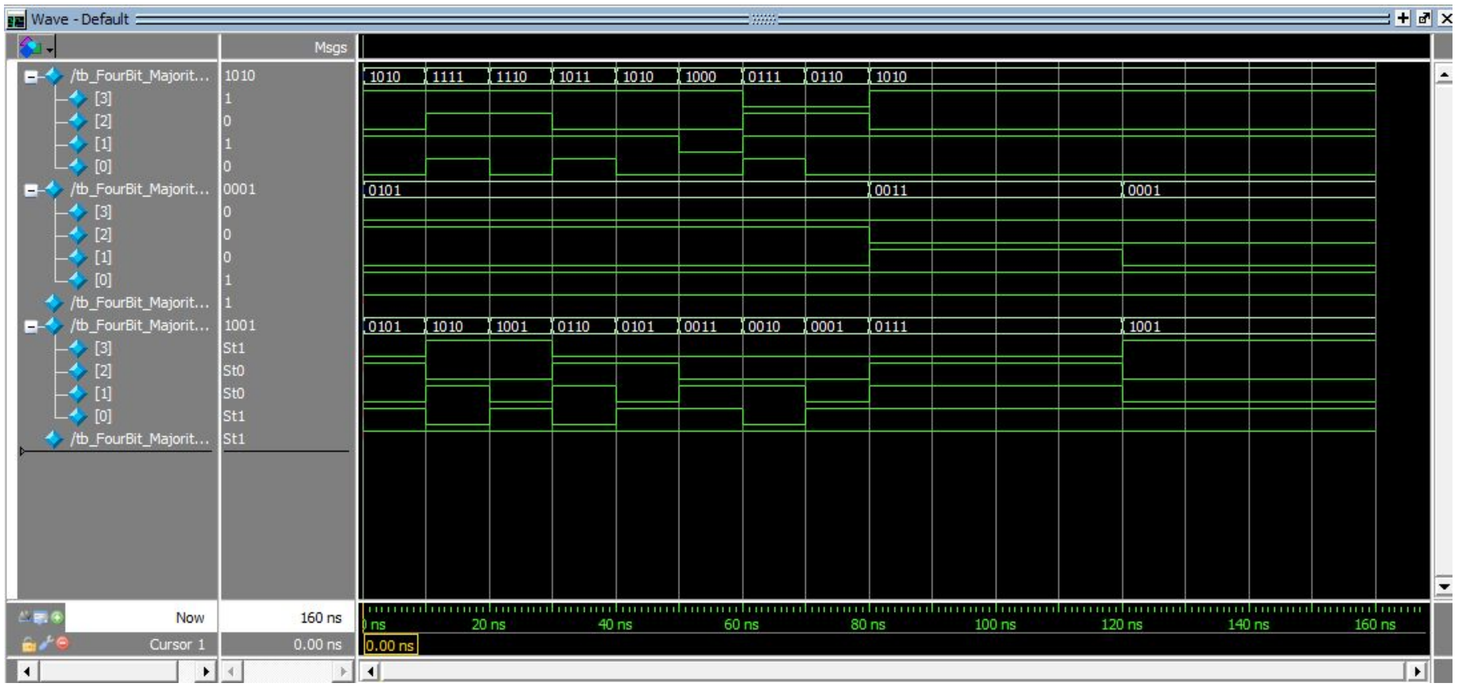


Figure 3.5. Design Entry of testbench 4-bit Subtractor



```verilog
//German E Felisarta III   16101002 Group 3 M 07:30 AM - 10:30 AM

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

   //All inputs reg  A, B, C_in,
   reg [3:0]X;
   reg [3:0]Y;
   reg    C_i;

   //all outputs wire S, C_out
   wire [3:0]Sum;
   wire      C_o;

   //instatntiate UUT
   FourBit_Majority UUT (X, Y, C_i, Sum, C_o);

   //generate stimuli
   initial
      begin
         X = 4'b1010; Y = 4'b0101; C_i = 1; #10
         X = 4'b1111; Y = 4'b0101; C_i = 1; #10
         X = 4'b1110; Y = 4'b0101; C_i = 1; #10
         X = 4'b1011; Y = 4'b0101; C_i = 1; #10
         X = 4'b1010; Y = 4'b0101; C_i = 1; #10
         X = 4'b1000; Y = 4'b0101; C_i = 1; #10
         X = 4'b0111; Y = 4'b0101; C_i = 1; #10
         X = 4'b0110; Y = 4'b0101; C_i = 1; #10
         X = 4'b1010; Y = 4'b0011; C_i = 1; #10
         X = 4'b1010; Y = 4'b0011; C_i = 1; #10
         X = 4'b1010; Y = 4'b0011; C_i = 1; #10
         X = 4'b1010; Y = 4'b0011; C_i = 1; #10
         X = 4'b1010; Y = 4'b0001; C_i = 1; #10
         X = 4'b1010; Y = 4'b0001; C_i = 1; #10
         X = 4'b1010; Y = 4'b0001; C_i = 1; #10
         X = 4'b1010; Y = 4'b0001; C_i = 1; #10

         $stop;
      end

endmodule
```

Figure 3.6. Timing Diagram 4-bit Subtractor in ModelSim-Altera


Wave - Default
Msgs
/tb_FourBit_Majorit... 1010
[3] 1
[2] 0
[1] 1
[0] 0
/tb_FourBit_Majorit... 0001
[3] 0
[2] 0
[1] 0
[0] 1
/tb_FourBit_Majorit... 1
/tb_FourBit_Majorit... 1001
[3] St1
[2] St0
[1] St0
[0] St1
/tb_FourBit_Majorit... St1
1010 1111 1110 1011 1010 1000 0111 0110 1010
0101 0011 0001
0101 1010 1001 0110 0101 0011 0010 0001 0111 1001
Now 160 ns
Cursor 1 0.00 ns
0 ns 20 ns 40 ns 60 ns 80 ns 100 ns 120 ns 140 ns 160 ns
0.00 ns