Laboratory Report #2

Name: Josh Ratificar **Date Completed:** 09-09-2023

Laboratory Exercise Title: Basic Constructs in Verilog HDL

Part B. *Full Adder*

```
1  /*************************************************************
2  *  File:              FullAdder.v
3  *  Author:            Josh Ratificar
4  *  Class:             Gr.3 CpE 3101L Introduction to HDL
5  *  Group/Schedule:    Friday, 7:30 AM to 10:30 AM
6  *  Description:       FullAdder.v module.
7  *************************************************************/
8
9  module HalfAdder (x, y, C, S);
10    input    x, y;
11    output   C, S;
12    xor      X1 (S, x, y);
13    and      A1 (C, x, y);
14 endmodule
15 module FullAdder (A, B, C_in, S, C_out);
16    input       A, B, C_in;
17    output      C_out, S;
18    wire        w1, w2, w3;
19    HalfAdder   Ha1(A, B, w1, w2);
20    HalfAdder   Ha2(w2, C_in, w3, S);
21    xor         xorGate(C_out, w1, w3);
22 endmodule
23
```

Figure 1.0 – *Full Adder Module (with Half Adders)*

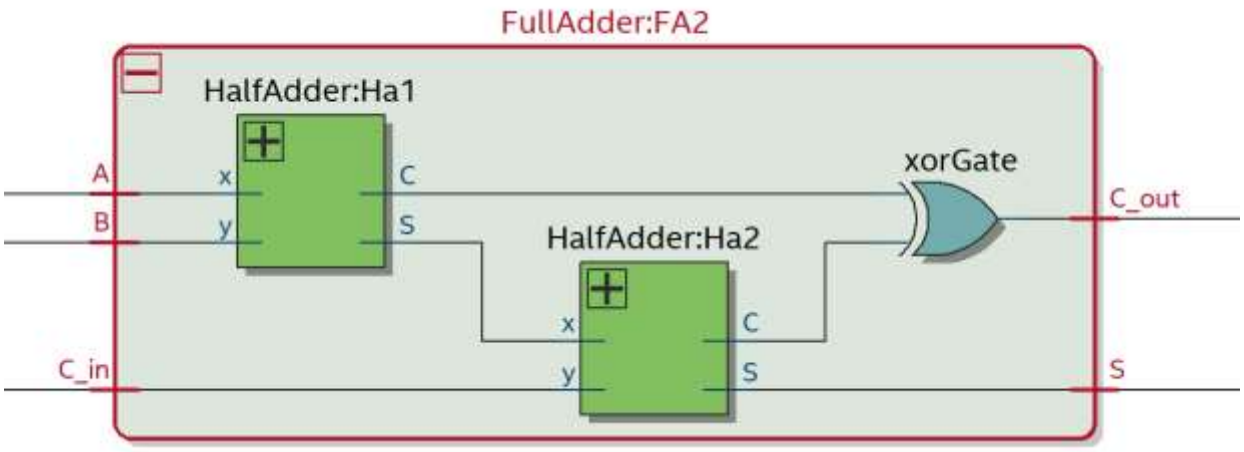

Figure 1.1 – *Full Adder Block Diagram (RTL Viewer)*

Part C. *Four-Bit Adder*

Table. 2.0 – *Compilation Report for Flow Summary*

# of Logic Elements Used:	**8**
# of Logic Pins Used:	**14**

```
1  /*************************************************************
2  *  File:              FourBitAdder.v
3  *  Author:            Josh Ratificar
4  *  Class:             CpE 3101L Introduction to HDL
5  *  Group/Schedule:    Gr. 3 Friday, 7:30 AM to 10:30 AM
6  *  Description:       FourBitAdder.v Module to add 4 bits
7  *************************************************************/
8  module FourBitAdder(A[3:0], B[3:0], C_in, S[3:0], C_out);
9     input    [3:0] A, B;
10    input    C_in;
11    output   [3:0] S;
12    output   C_out;
13    wire     w1, w2, w3;
14
15    // FullAdder (A, B, C_in, S, C_out);
16    FullAdder   FA1(A[0], B[0], C_in, S[0], w1);
17    FullAdder   FA2(A[1], B[1], w1, S[1], w2);
18    FullAdder   FA3(A[2], B[2], w2, S[2], w3);
19    FullAdder   FA4(A[3], B[3], w3, S[3], C_out);
20 endmodule
21
```

Figure 2.0 – *FourBitAdder.v Module*

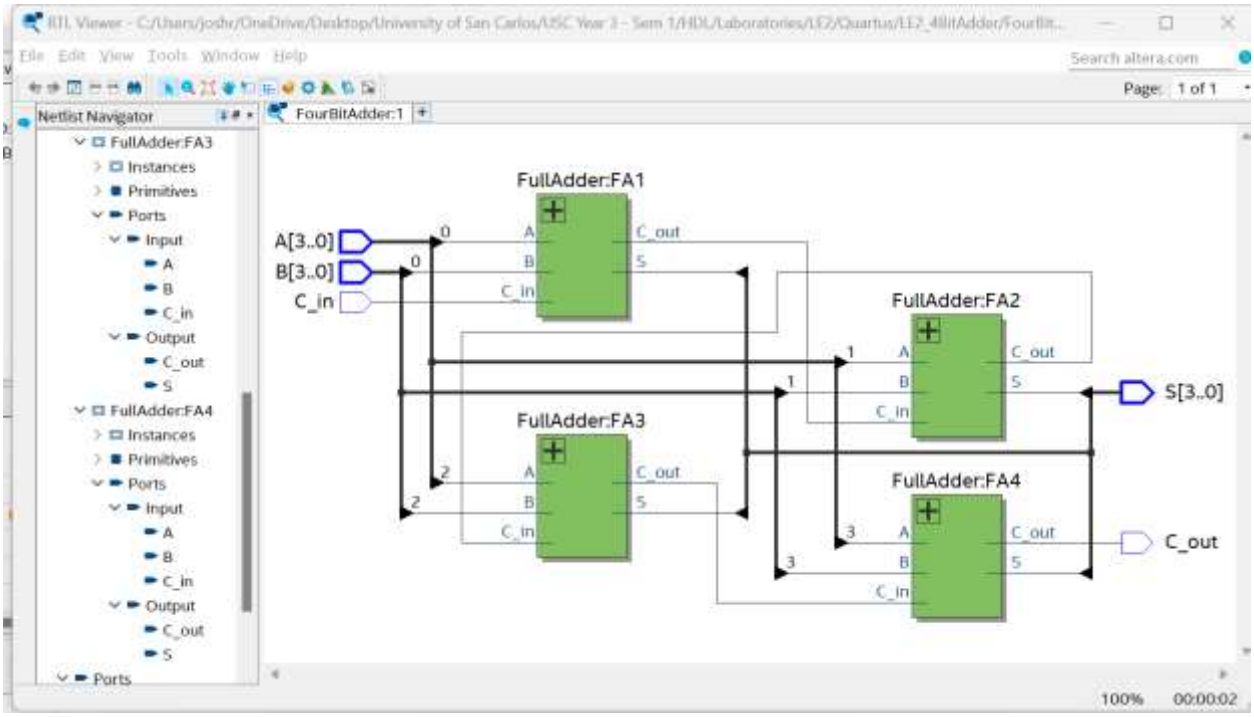

Figure 2.0 – *FourBitAdder.v RTL Viewer*

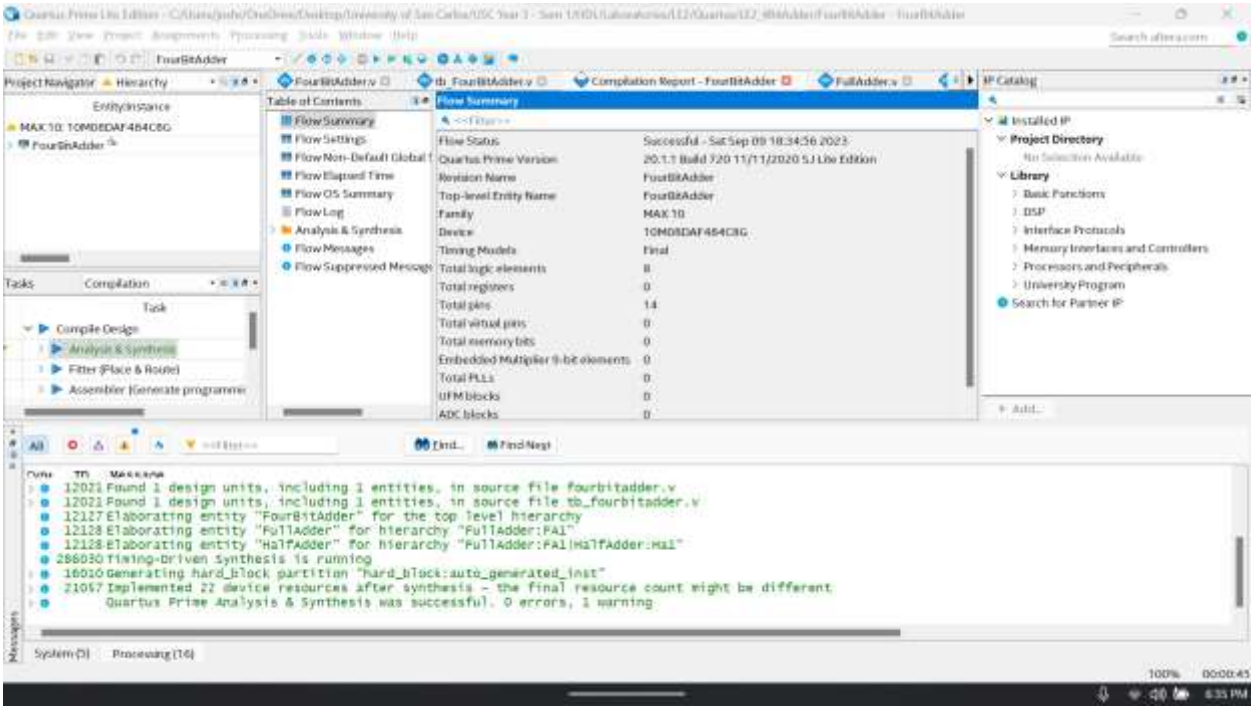

Figure 2.2 – *FourBitAdder.v Analysis and Synthesis*

```
1  /*************************************************************
2  *  File:              tb_4-BitAdder.v
3  *  Author:            Josh Ratificar
4  *  Class:             CpE 3101L Introduction to HDL
5  *  Group/Schedule:    Friday, 7:30 AM to 10:30 AM
6  *  Description:       Testbench file for 4-BitAdder.v
7  *************************************************************/
8  `timescale 1 ns/ 1 ps
9  module tb_FourBitAdder();
10    // all inputs to UUT are declared as reg type
11    reg      [3:0] A, B;
12    reg      C_in;
13    wire     [3:0] S;
14    wire     C_out;
15    // instantiate UUT with implicit oprt mapping
16    FourBitAdder UUT (A[3:0], B[3:0], C_in, S[3:0], C_out);
17
18    // generate stimuli
19    initial
20    begin
21       A =  4'd0;  B = 4'd0;   C_in = 0;   #10
22       A =  4'd3;  B = 4'd8;   C_in = 1;   #10
23       A =  4'd11; B = 4'd3;   C_in = 0;   #10
24       A =  4'd12; B = 4'd6;   C_in = 0;   #10
25       A =  4'd5;  B = 4'd4;   C_in = 1;   #10
26       A =  4'd1;  B = 4'd9;   C_in = 0;   #10
27       A =  4'd15; B = 4'd15;  C_in = 0;   #10
28       A =  4'd15; B = 4'd15;  C_in = 1;   #10
29       $stop;
30    end
31 endmodule
```

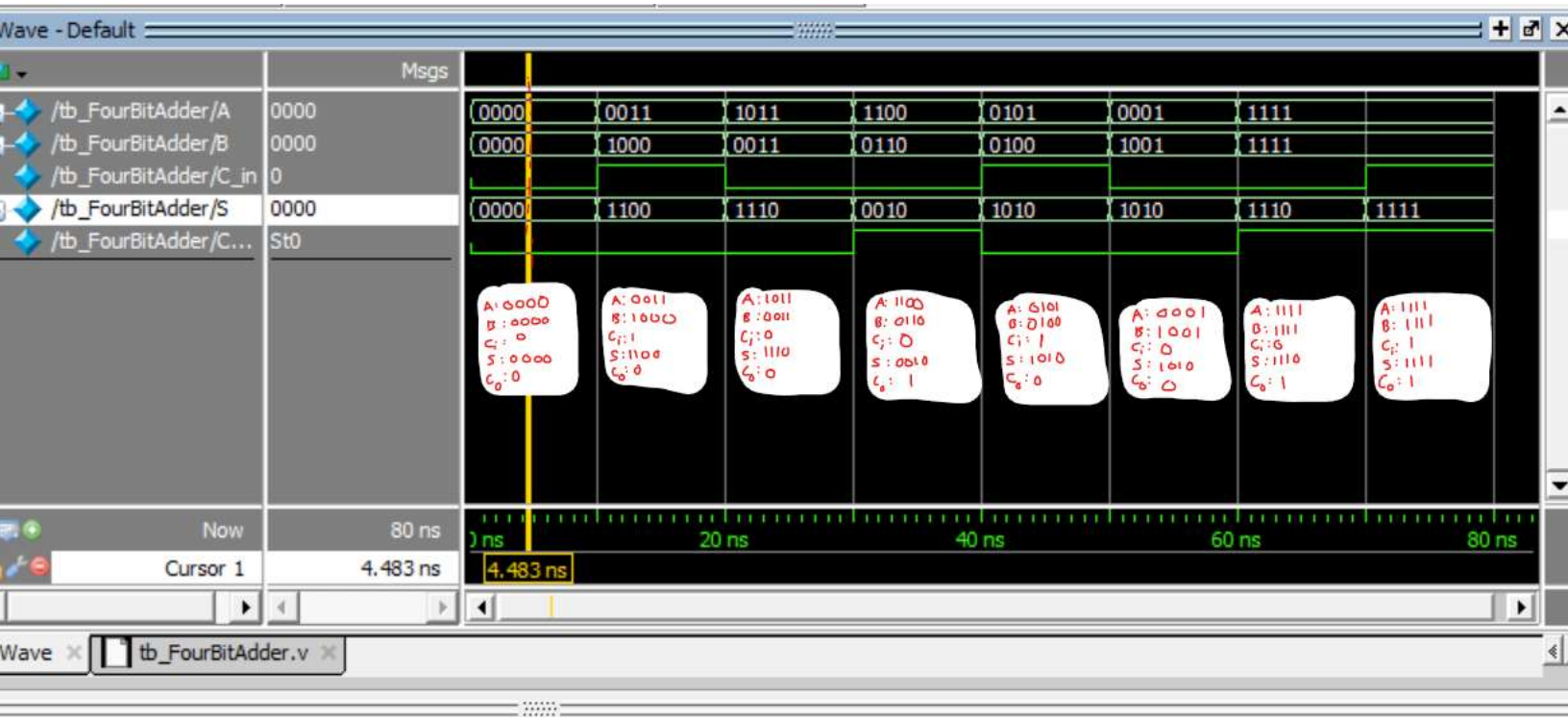

Figure 2.4 – *tb_FourBitAdder.v Test Bench Script*