Laboratory Report # 2

Name: Machacon, Zach Riane **Date Completed:** September 9, 2023

Laboratory Exercise Title: Basic Constructs in Verilog HDL

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 2C:

FullAdder.v

```
1  /*************************************************************
2  * File:              FullAdder.v
3  * Author:            Zach Riane I. Machacon
4  * Class:             CPE 3101L
5  * Group / Schedule:  Group 2 Fri 11:00AM - 2:00PM
6  * Description:       Design Entry for Full Adder
7  *
8  *************************************************************/
9
10 module HalfAdder (x, y, C, S);
11
12    input    x, y;
13    output   C, S;
14    xor      X1 (S, x, y);
15    and      A1 (C, x, y);
16
17 endmodule
18
19 module FullAdder (A, B, C_in, S, C_out);
20    input    A, B, C_in;
21    output   S, C_out;
22    wire     firstAdder_out, secondAdder_carry, firstAdder_carry;
23
24    HalfAdder firstAdder (  .x (A),
25                            .y (B),
26                            .C (firstAdder_carry),
27                            .S (firstAdder_out)
28                         );
29    HalfAdder secondAdder ( .x (C_in),
30                            .y (firstAdder_out),
31                            .C (secondAdder_carry),
32                            .S (S)
33                         );
34    or       (C_out, secondAdder_carry, firstAdder_carry);
35
36 endmodule
```

Figure 1: Design Entry for FullAdder in Verilog HDL

The FullAdder module makes use of structural modeling where it instantiates 2 HalfAdders to create a Full Adder. An OR gate primitive is also used to get the final carry out.

Adder_4bit.v

```
/*************************************************
* File:              Adder_4bit.v
* Author:            Zach Riane I. Machacon
* Class:             CPE 3101L
* Group / Schedule:  Group 2 Fri 11:00AM - 2:00PM
* Description:       Design Entry for 4-bit Adder
*
*************************************************/

module HalfAdder (x, y, C, S);

   input    x, y;
   output   C, S;
   xor      X1 (S, x, y);
   and      A1 (C, x, y);

endmodule

module FullAdder (A, B, C_in, S, C_out);
   input    A, B, C_in;
   output   S, C_out;
   wire     firstAdder_out, secondAdder_carry, firstAdder_carry;

   HalfAdder firstAdder (  .x (A),
                           .y (B),
                           .C (firstAdder_carry),
                           .S (firstAdder_out)
                        );
   HalfAdder secondAdder ( .x (C_in),
                           .y (firstAdder_out),
                           .C (secondAdder_carry),
                           .S (S)
                        );
   or       (C_out, secondAdder_carry, firstAdder_carry);

endmodule


module Adder_4bit(A, B, C_in, S, C_out);
   input    [3:0] A, B;
   input    C_in;
   output   [3:0] S;
   output   C_out;
   wire     FA1_out, FA2_out, FA3_out;

   FullAdder FA1  (  .A       (A[0]),
                     .B       (B[0]),
                     .C_in    (C_in),
                     .S       (S[0]),
                     .C_out   (FA1_out)
                  );
   FullAdder FA2  (  .A       (A[1]),
                     .B       (B[1]),
                     .C_in    (FA1_out),
                     .S       (S[1]),
                     .C_out   (FA2_out)
                  );
   FullAdder FA3  (  .A       (A[2]),
                     .B       (B[2]),
                     .C_in    (FA2_out),
                     .S       (S[2]),
                     .C_out   (FA3_out)
                  );
   FullAdder FA4  (  .A       (A[3]),
                     .B       (B[3]),
                     .C_in    (FA3_out),
                     .S       (S[3]),
                     .C_out   (C_out)
                  );
endmodule
```

Figure 2: Design Entry for 4-bit FullAdder in Verilog HDL

The Adder_4bit module also utilizes structural modeling. Building off the previous FullAdder and HalfAdder modules, the Adder_4bit module instantiates 4 FullAdders to create the 4-bit Full Adder.

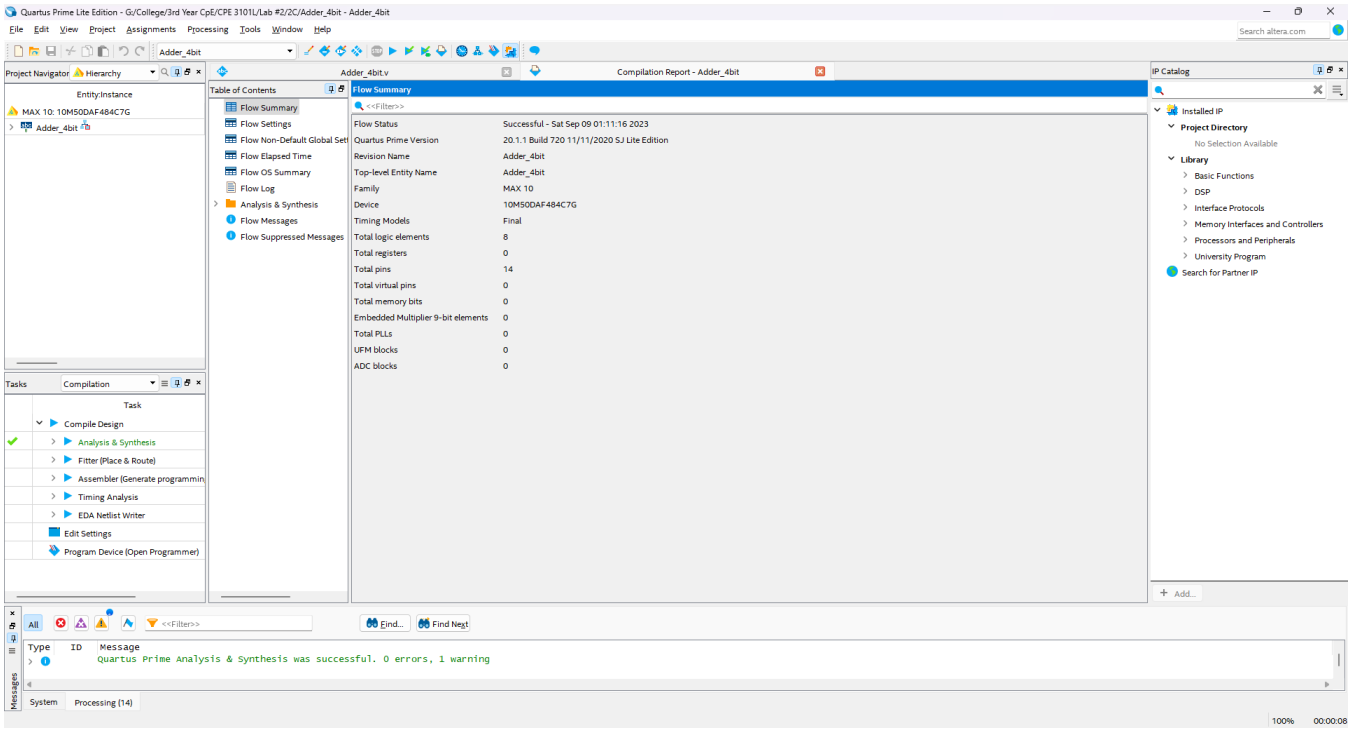

Figure 3: Compilation Report for the Flow Summary of Adder_4bit

According to the compilation report, 8 logic elements and 14 total pins are needed. Quartus also found 1 warning, however, it is negligible and does not interfere with the synthesis process.

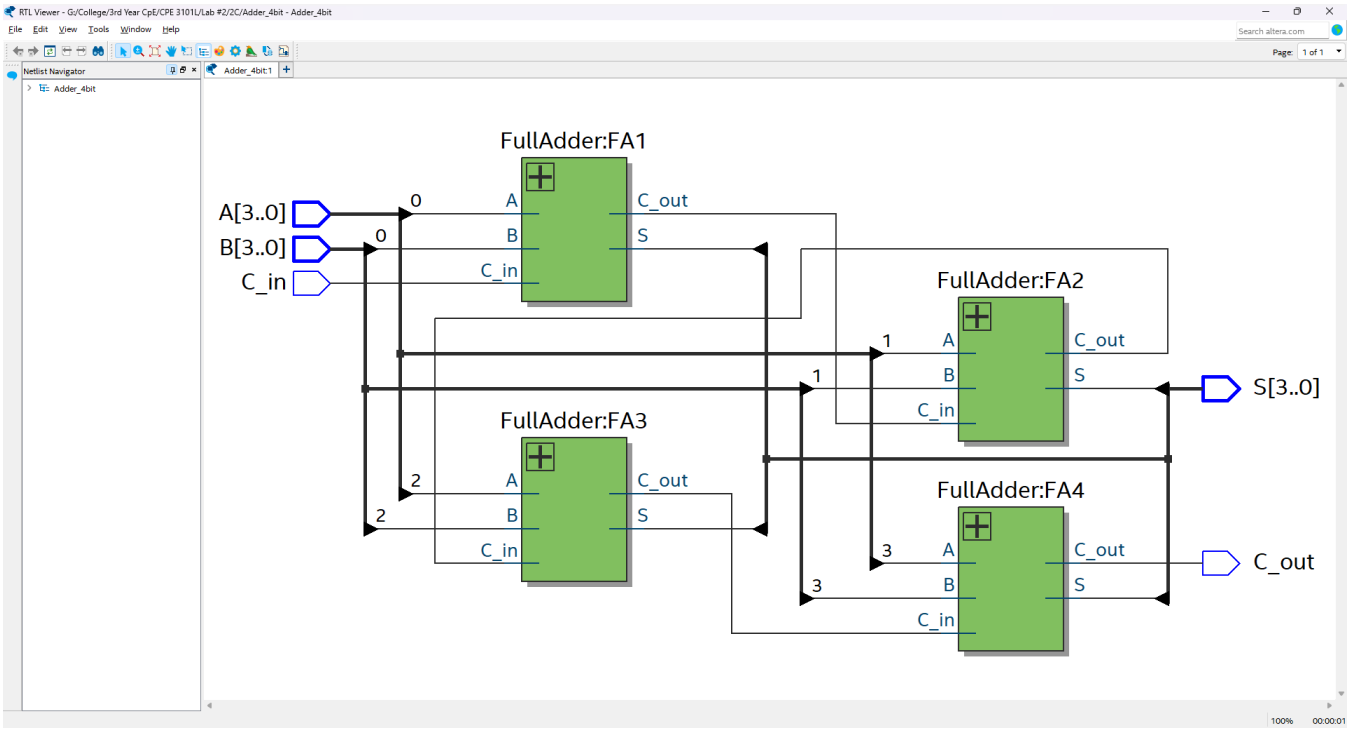

Figure 4: Schematic Diagram of Synthesized Circuit from Adder_4bit Project

To create the 4-bit Full Adder, 4 Full Adders are connected in sequence with each next adder's carry-in being the previous adder's carry-out. Each sum calculated from the Full Adders contribute the final 4-bit sum while the final carry-out is the carry-out of the final adder.

tb_Adder_4bit.v

```
/*************************************************************
* File:                 tb_Adder_4bit.v
* Author:               Zach Riane I. Machacon
* Class:                CPE 3101L
* Group / Schedule:     Group 2 Fri 11:00AM - 2:00PM
* Description:          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 port mapping (you must supply this part)
    Adder_4bit UUT (  .A        (A),
                      .B        (B),
                      .C_in     (C_in),
                      .S        (S),
                      .C_out    (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;                  // system task to end simulation
    end

endmodule
```

Figure 5: Verilog Testbench File for Adder_4bit Project

The testbench files contain several test inputs which are in decimal. This is to ensure that the 4-bit Full Adder is outputting the correct answers.

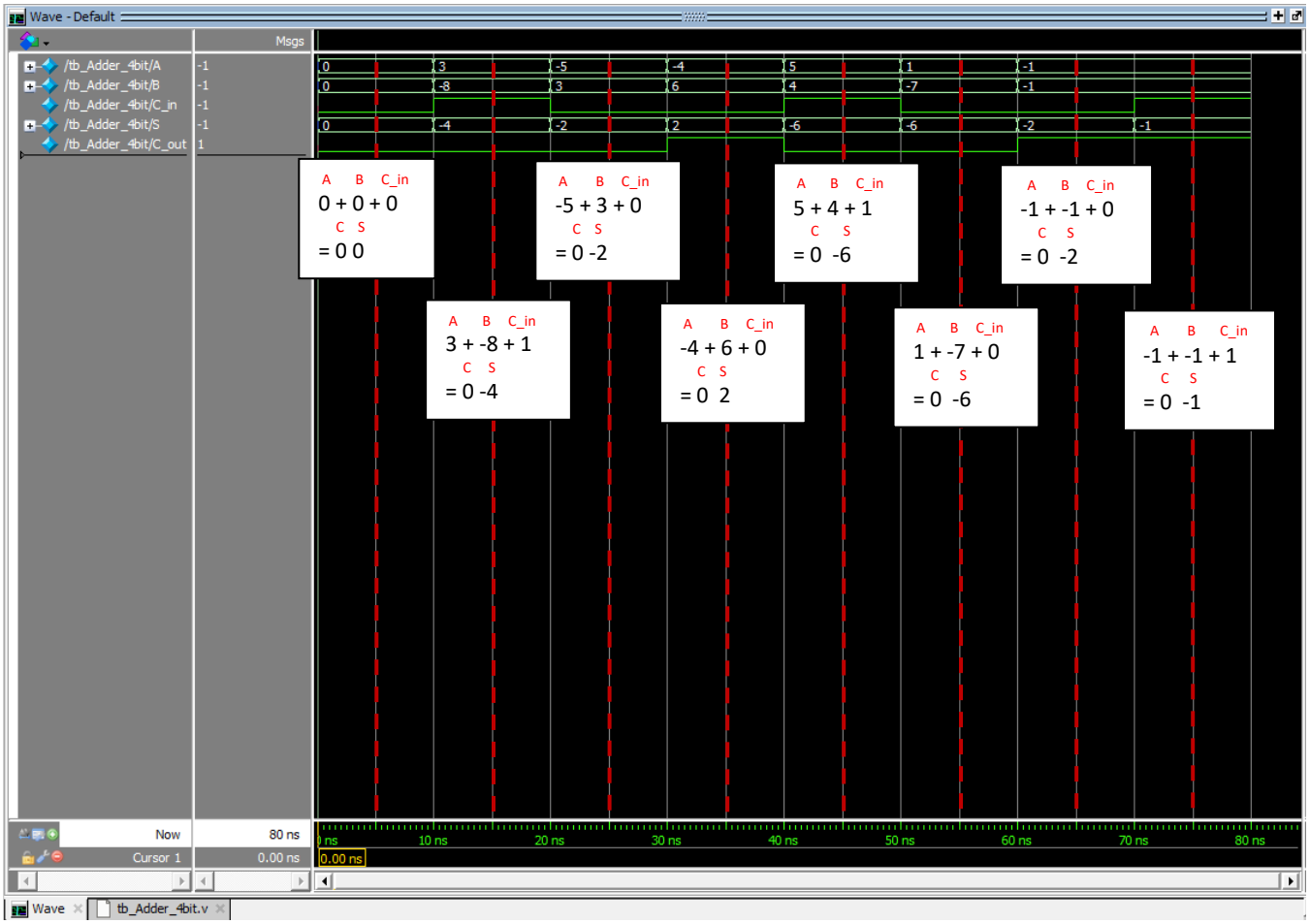

Figure 6: Verilog Testbench Waveform for Adder_4bit Project