David Nguyen

Lab 2

Section A

10/19/2018

Lab 1 Writeup

Purpose:

The purpose of this lab was to learn how to use hierarchy in design and how to simulate the design with Vivado simulator using a segment adder and 7 segment display. I learned how to build a ripple adder out of full adders and learned how to display numbers onto a seven segment display physically and through a simulator.

Methods:

Adder:

1. The first part to building the adder as a module is creating a full adder module with inputs a, b, cin, and outputs s and cout. We would have to implement this design based on the truth table created by a, b, and cin.
2. The next step to create the segment adder is to connect the cout of the first bit to the cin of the second bit of the 2 respected numbers and continue until the n bit.
3. The answer of the addition will be the s output of the respected adder for the bit number

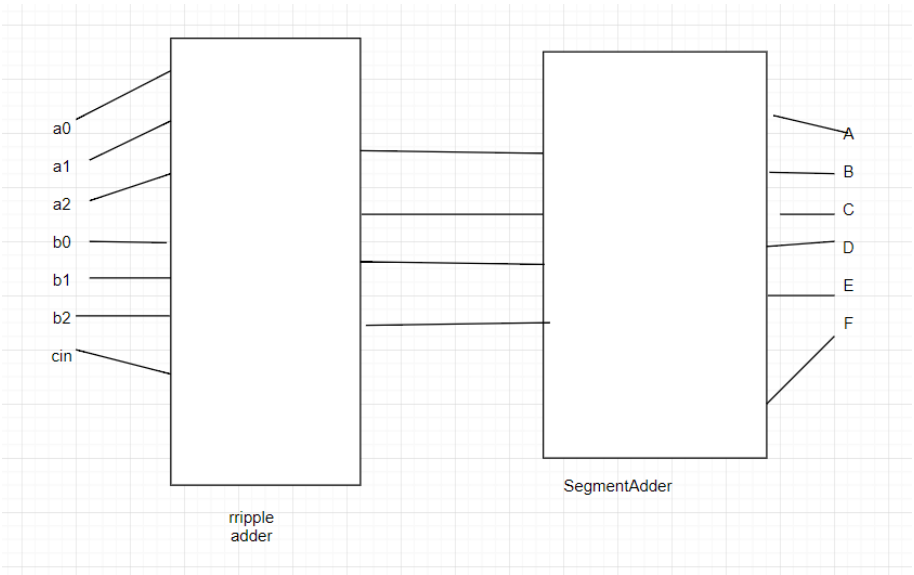
7 Segment Display:

1. The first part of 7 segment display is to make a truth table with inputs being a 4 bit number and the outputs as each of the logic equations for the 12 outputs of the 7 segment display. (7/12 are non-trivial).
2. The next step was to download and configure lab2_test.v to generate the display for 0-F advancing by 100ns each time.
3. The is also a step to connecting the 4 wires to the waveform in the simulation which then can be turned into a bus and can be used to check if the adder is working.

Results

1. In this lab, cin was connected to sw0, a0-2 was connected to sw1-3,and b0-2 was connected to sw4-6. A-G was connected to C# with the respected letter replacing the number sign. Finally, for the anodes for the display, an0-3 were connected to the same element.
2. The longest path in this 3-bit adder was input a0/b0 to cOut of the 3rd full adder going through 7 gates.
3. The Longest length from any input to output of N size is ((2*N)+1).

4. There are 128 possible input values for the adder. Because there are 7 bits, the max number of possibilities would be (2^7). We tested for 12.5% of test 16/128, this means there are more possible was to get a certain output that we did not test for even through we tested to make sure all outputs for the 7 segment display work.



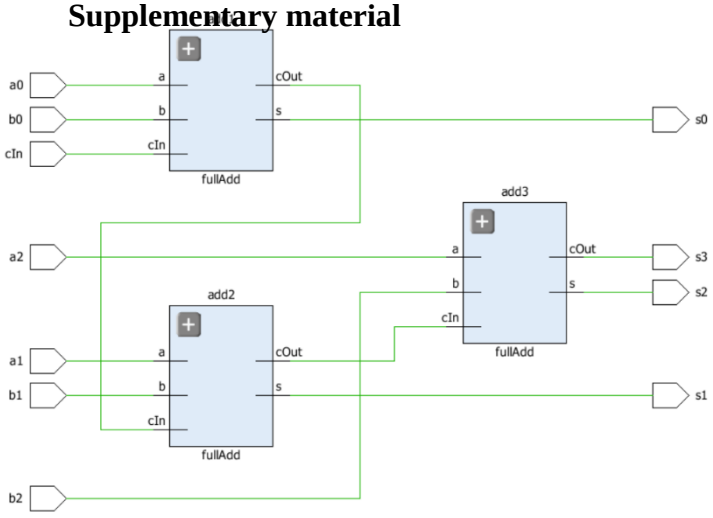
In this design there are 7 inputs for the top module which take in a0-2, b0-2, and cin based on the switches. This then goes trough ripple adder which adds the 2 3bit numbers together and outputs 4 different signals to segadder which uses the 4 signals to know which segment to display on A-G. The segAdder has logic based on the 4 signals to know if outputs A-G are supposed to light up based on the 4-bit number of the addition from ripple adder.

To test and simulate the design of topMod, in the test file, we would increment time by 100ns each test. In between the incremented time we would have to change the switches based on the input to get the output for 0-E. for example to get D to output, you can set the switch sw[0:6] to {0,1,1,1,1,1,1,}. But we can also set the switches to be {1,0,1,1,1,1,1} and the output of this will also be D on the display. These are the cases that we didn't test for in the simulation, because the output is still the same for the display.

Conclusion

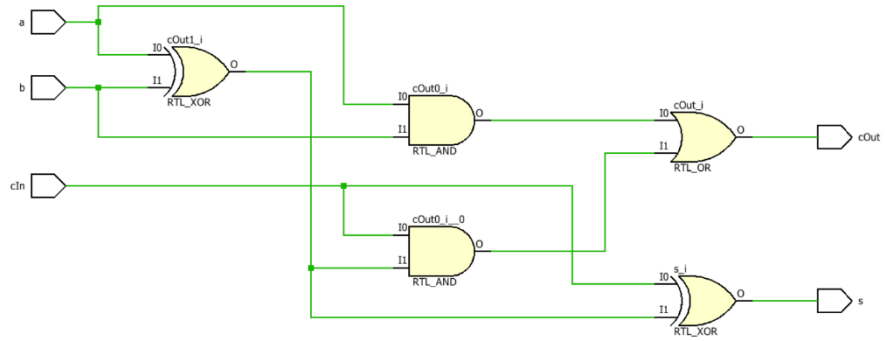
In conclusion, this lab was an introduction to hierarchy and how to use the simulation tool in Vivado to help debug the design. In part 1, we used hierarchy to complete a segmented adder out of full adders. In part 2, we used the simulation tool to check the outputs of our design while testing the design physically to debug any errors or mistakes. If I were to do this lab again I would write down which switches needed to be turned off an on for testing as it wok longer than necessary. I wouldn't change anything about this lab as it was clear with the instructions given and was a good learning lab.

Supplementary material




```
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/09/2018 10:21:34 AM
// Design Name:
// Module Name: ripAdd
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
```

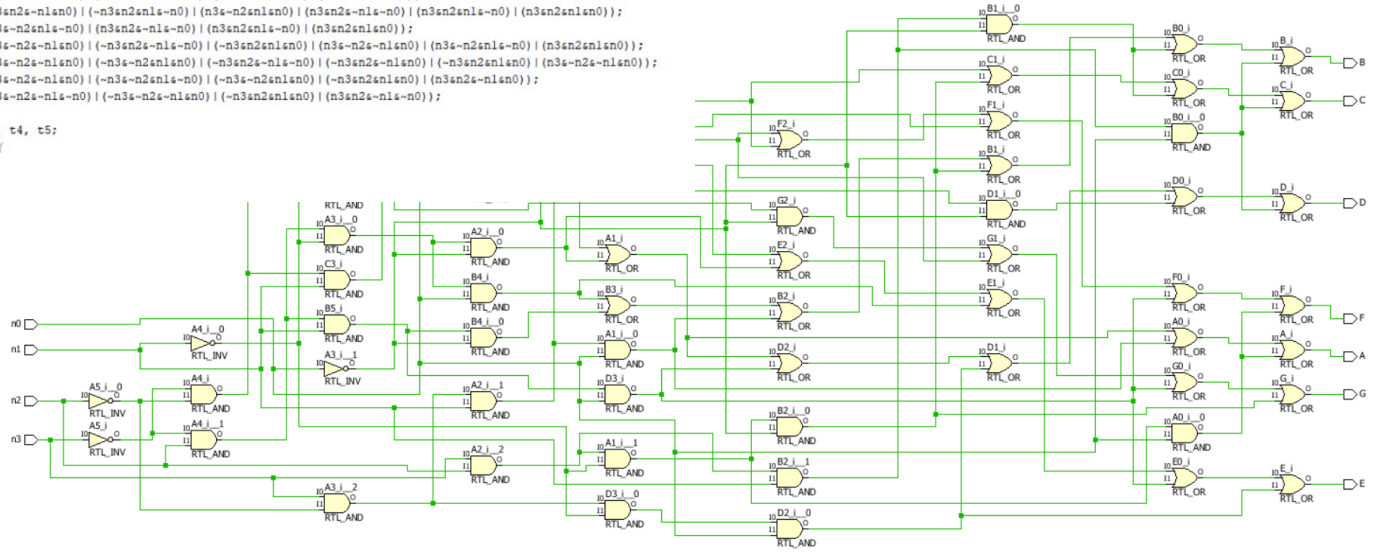
```
module ripAdd(
    input a0,
    input b0,
    input a1,
    input b1,
    input a2,
    input b2,
    input cIn,
    output s0,
    output s1,
    output s2,
    output s3
    );

    wire t1, t2;
    fullAdd add1(.a(a0), .b(b0), .cIn(cIn) , .cOut(t1), .s(s0));
    fullAdd add2(.a(a1), .b(b1), .cIn(t1) , .cOut(t2), .s(s1));
    fullAdd add3(.a(a2), .b(b2), .cIn(t2) , .cOut(s3), .s(s2));
endmodule
```

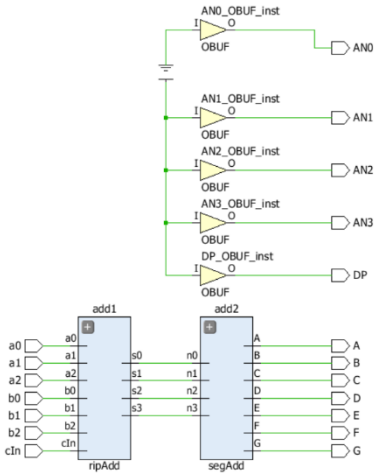



```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/09/2018 10:04:39 AM
// Design Name:
// Module Name: fullAdd
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
```

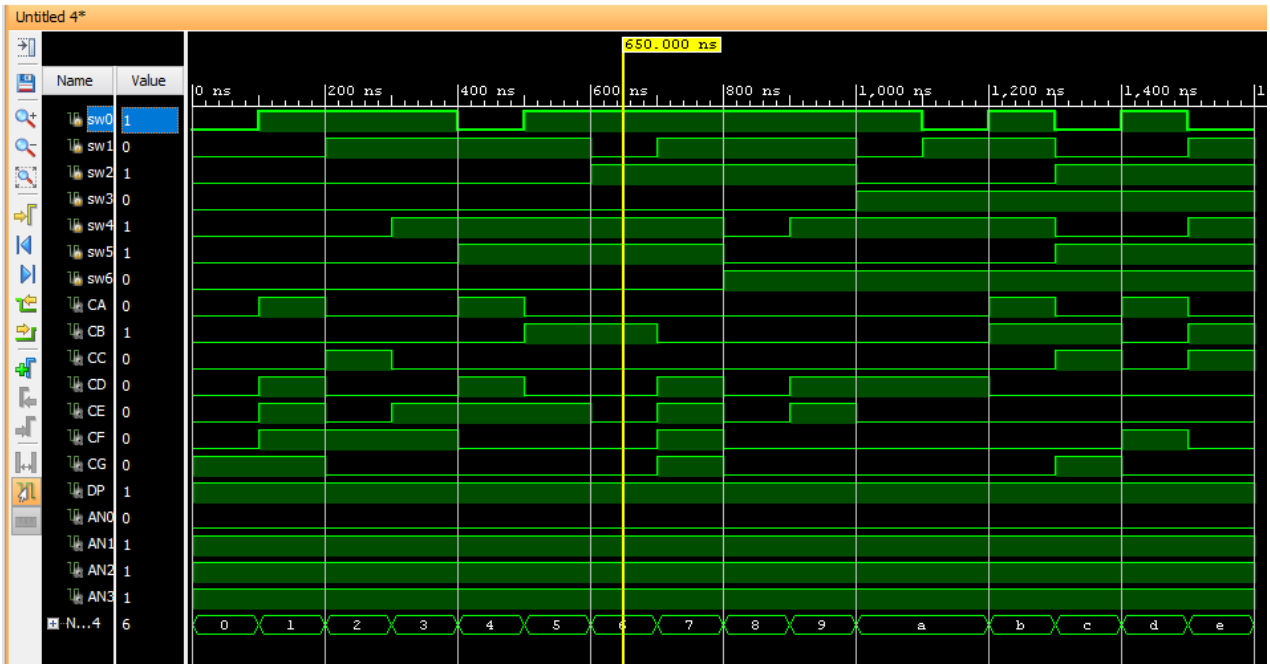
```
module segAdd(
    input n0,
    input n1,
    input n2,
    input n3,
    output A,
    output B,
    output C,
    output D,
    output E,
    output F,
    output G
    );
    assign A = ((~n3&~n2&~n1&n0)|(~n3&n2&~n1&~n0)|(n3&~n2&n1&n0)|(n3&n2&~n1&n0));
    assign B = ((~n3&n2&~n1&n0)|(~n3&n2&n1&~n0)|(n3&~n2&n1&n0)|(n3&n2&~n1&~n0)|(n3&n2&n1&~n0)|(n3&n2&n1&n0));
    assign C = ((~n3&~n2&n1&~n0)|(n3&n2&~n1&~n0)|(n3&n2&n1&~n0)|(n3&n2&n1&n0));
    assign D = ((~n3&~n2&~n1&n0)|(~n3&n2&~n1&~n0)|(~n3&n2&n1&n0)|(n3&~n2&~n1&n0)|(n3&~n2&n1&~n0)|(n3&n2&n1&n0));
    assign E = ((~n3&~n2&~n1&n0)|(~n3&~n2&n1&n0)|(~n3&n2&~n1&~n0)|(~n3&n2&~n1&n0)|(~n3&n2&n1&n0)|(n3&~n2&~n1&n0));
    assign F = ((~n3&~n2&~n1&n0)|(~n3&~n2&n1&~n0)|(~n3&~n2&n1&n0)|(~n3&n2&n1&n0)|(n3&n2&~n1&n0));
    assign G = ((~n3&~n2&~n1&~n0)|(~n3&~n2&~n1&n0)|(~n3&n2&n1&n0)|(n3&n2&~n1&~n0));

    wire t1, t2, t3, t4, t5;
    //ripAdd seven0(

endmodule
```

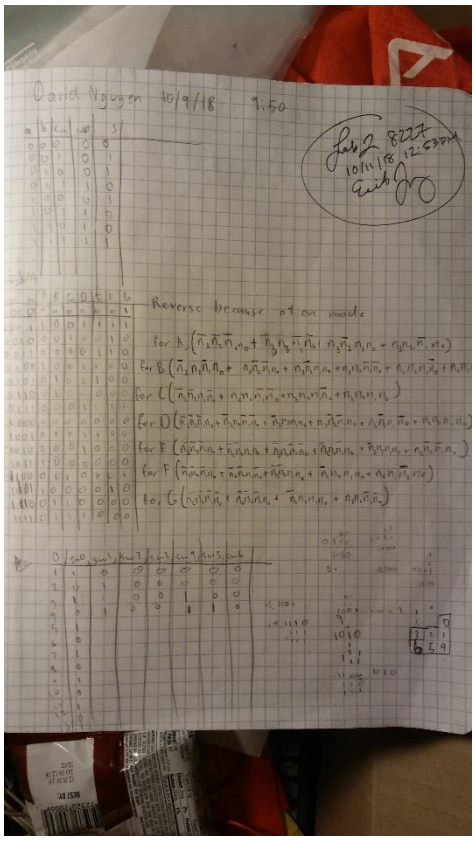
```
module fullAdd(
    input a,
    input b,
    input cIn,
    output cOut,
    output s
    );
    assign s = cIn^(a^b);
    assign cOut = (a&b)|(cIn&(a^b));
endmodule
```






```
module topMod(
    input a0,
    input b0,
    input a1,
    input b1,
    input a2,
    input b2,
    input cIn,
    output A,
    output B,
    output C,
    output D,
    output E,
    output F,
    output G,
    output AN0,
    output AN1,
    output AN2,
    output AN3,
    output DP
    );
    wire t0, t1,t2,t3,t4;
    ripAdd add1(.a0(a0), .b0(b0), .a1(a1), .b1(b1), .a2(a2), .b2(b2),
.cIn(cIn),.s0(t0),.s1(t1),.s2(t2),.s3(t3));
    segAdd add2(.n0(t0), .n1(t1), .n2(t2), .n3(t3),
.A(A),.B(B),.C(C),.D(D),.E(E),.F(F),.G(G));
    assign AN0 = 0;
    assign AN1 = 1;
    assign AN2 = 1;
    assign AN3 = 1;
    assign DP = 1;
endmodule
```




David Nguyen 10/9/18 9:50
Lab 2 822A
10/11/18 12:53PM
Reverse because of an inside