ELC 2137 Lab 5: Intro to Verilog

Celaine Hornsby

September 29, 2020

Summary

In this lab we learned how to code the circuits that we built in previous labs. We were able to code the half adder, full adder, and 2-bit adder subtractor that we built in lab. We gained a better understanding of how Verilog and Vivado works from this lab. through the use of design files and simulation files that acted as a test bench for our design files.

Q&A

1. What is one thing that you still don't understand about Verilog?
   I don't understand how to run simulations of design files. Mine constantly have confusing errors that make my simulations not be able to run.

Code

Listing 1: halfadder.sv

```
‘timescale 1ns / 1ps
//
   //////////////////////////////////////////////////////////////////////////////////

// Company: ELC 2137
// Engineer: Celaine Hornsby
//
// Create Date: 09/24/2020 04:47:49 PM
// Design Name:
// Module Name: halfadder
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
```

```
//
   //////////////////////////////////////////////////////////////////////////////////



module halfadder(
    input a,
    input b,
    output c,
    output s
    );

    assign c = a & b;
    assign s = a ^ b;
endmodule
```

Listing 2: halfadder test.sv

```
`timescale 1ns / 1ps
//
   //////////////////////////////////////////////////////////////////////////////////

// Company: ELC 2137
// Engineer: Celaine Hornsby
//
// Create Date: 09/24/2020 04:54:54 PM
// Design Name:
// Module Name: halfadder_test
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module halfadder_test();
    reg a1, b1;
    wire c1, s1;


    halfadder ha0(
    .a(a1), .b(b1),
    .c(c1), .s(s1)
    );
```

```
    initial begin
        a1 = 0; b1 = 0; #10;
        a1 = 1; b1 = 0; #10;
        a1 = 0; b1 = 1; #10;
        a1 = 1; b1 = 1; #10;
        $finish ;
        end
endmodule
```

Listing 3: fulladder.sv

```
`timescale 1ns / 1ps
//
   //////////////////////////////////////////////////////////////////////////////////

// Company: ELC 2137
// Engineer: Celaine Hornsby
//
// Create Date: 09/24/2020 05:07:41 PM
// Design Name:
// Module Name: fulladder
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module fulladder(
    input ain,
    input bin,
    input cin,
    output cout,
    output sout
    );

    wire c1, c2, s1;

    halfadder ha0(
    .a(ain), .b(bin),
    .c(c1), .s(s1)
    );

    halfadder ha1(
    .a(s1), .b(cin),
```

```
    .c(c2), .s(sout)
    );

    assign cout = c1 ^ c2;
endmodule
```

Listing 4: full adder test.sv

```
`timescale 1ns / 1ps
//
    //////////////////////////////////////////////////////////////////////////////////

// Company: ELC 2137
// Engineer: Celaine Hornsby
//
// Create Date: 09/24/2020 05:07:41 PM
// Design Name:
// Module Name: fulladder
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
    //////////////////////////////////////////////////////////////////////////////////



module FA_test();

    reg cin, ain, bin;
    wire cout, sout;

    fulladder fa0(
    .ain(ain), .bin(bin), .cin(cin),
    .cout(cout), .sout(sout)
    );

    initial begin
        cin = 0; ain = 0; bin = 0; #10;
        cin = 0; ain = 0; bin = 1; #10;
        cin = 0; ain = 1; bin = 0; #10;
        cin = 0; ain = 1; bin = 1; #10;
        cin = 1; ain = 0; bin = 0; #10;
        cin = 1; ain = 0; bin = 1; #10;
        cin = 1; ain = 1; bin = 0; #10;
        cin = 1; ain = 1; bin = 1; #10;
       $finish ;
```

```
        end

endmodule
```

Listing 5: 2bit

```
‘timescale 1ns / 1ps
//
   //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 09/28/2020 10:38:37 PM
// Design Name:
// Module Name: addsub3
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////////



module addsub3(
    input [1:0] a, b,
    input mode,
    output cbout,
    output [1:0] sout
    );

    wire c1, c2;
    wire [1:0] b_n;

    assign b_n[0] = b[0] ^ mode;
    assign b_n[1] = b[1] ^ mode;

    fulladder fa0(
    .ain(a[0]) , .bin(b_n[0]), .cin(mode),
    .cout(c1), .sout(sout[0])
    );

    fulladder fa1(
    .ain(a[1]) , .bin(b_n[1]), .cin(c1),
    .cout(c2), .sout(sout[1])
    );
```

```
    assign cbout = c2 ^ mode;
endmodule
```

Listing 6: 2bit test

```
`timescale 1ns / 1ps
//
    //////////////////////////////////////////////////////////////////////////////////

// Company: ELC 2137
// Engineer: Celaine Hornsby
//
// Create Date: 09/24/2020 06:14:47 PM
// Design Name:
// Module Name: addsub2_test
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
    //////////////////////////////////////////////////////////////////////////////////



module as_test();

    reg [1:0] a;
    reg [1:0] b;
    reg m;
    wire [1:0] s;
    wire cb;

    addsub3 as(
    .a(a), .b(b), .mode(m),
    .sout(s), .cbout(cb)
    );

    initial begin

        a[0] = 0; b[0] = 1; a[1] = 0; b[1] = 0; m = 0; #10;
        a[0] = 0; b[0] = 0; a[1] = 0; b[1] = 1; m = 0; #10;
        a[0] = 0; b[0] = 1; a[1] = 0; b[1] = 1; m = 0; #10;
        a[0] = 1; b[0] = 1; a[1] = 0; b[1] = 0; m = 0; #10;
        a[0] = 0; b[0] = 1; a[1] = 1; b[1] = 0; m = 0; #10;
        a[0] = 0; b[0] = 0; a[1] = 1; b[1] = 0; m = 0; #10;
        a[0] = 0; b[0] = 1; a[1] = 0; b[1] = 0; m = 1; #10;
```

```
        a[0] = 0; b[0] = 0; a[1] = 0; b[1] = 1; m = 1; #10;
        a[0] = 0; b[0] = 1; a[1] = 0; b[1] = 1; m = 1; #10;
        a[0] = 1; b[0] = 1; a[1] = 0; b[1] = 0; m = 1; #10;
        a[0] = 0; b[0] = 1; a[1] = 1; b[1] = 0; m = 1; #10;
        a[0] = 0; b[0] = 0; a[1] = 1; b[1] = 0; m = 1; #10;

       $finish ;
        end
endmodule
```

Results

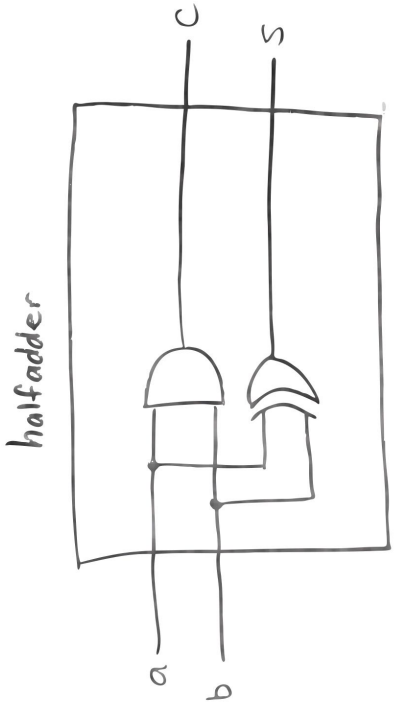

Figure 1: Half Adder Block Diagram

Time (ns):	0	10	20	30
a	0	1	0	1
b	0	0	1	1
c	0	0	0	1
s	0	1	1	0

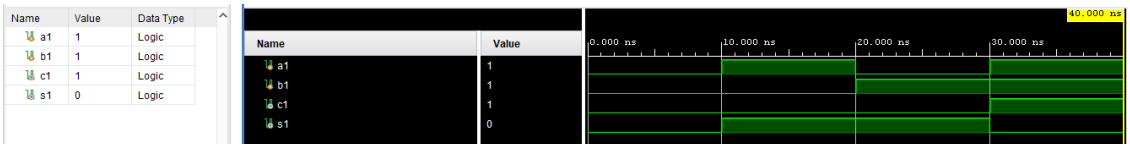

Figure 2: Half Adder waveform and ERT

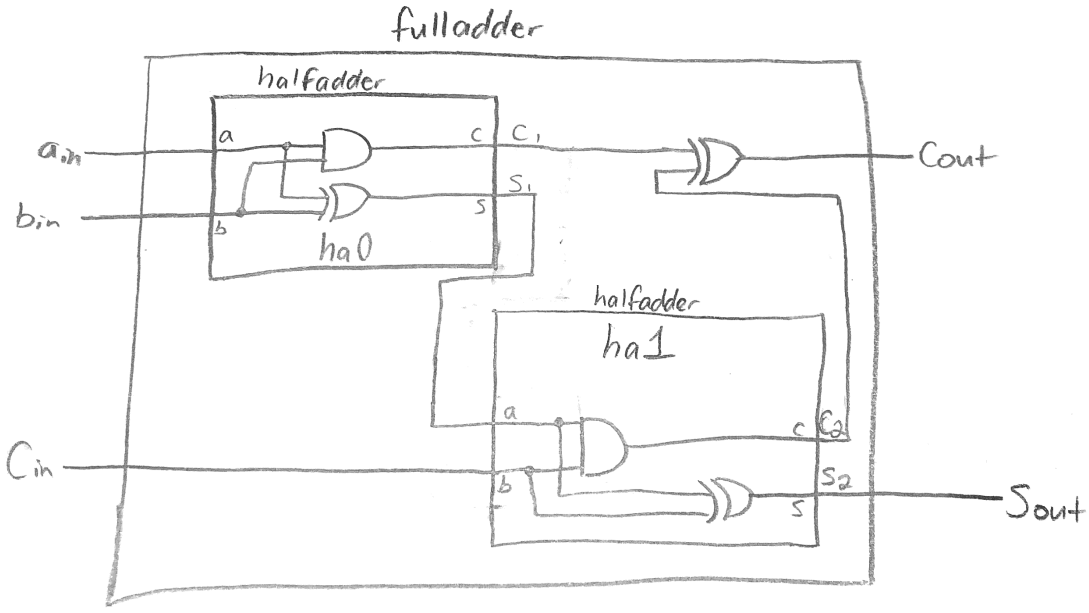

Figure 3: Full Adder Block Diagram

Time (ns):	0	10	20	30	40	50	60	70
a	0	1	0	1	0	1	0	1
b	0	0	1	1	0	0	1	1
cin	0	0	0	0	1	1	1	1
cout	0	0	0	1	0	1	1	1
sout	0	1	1	0	1	0	0	1

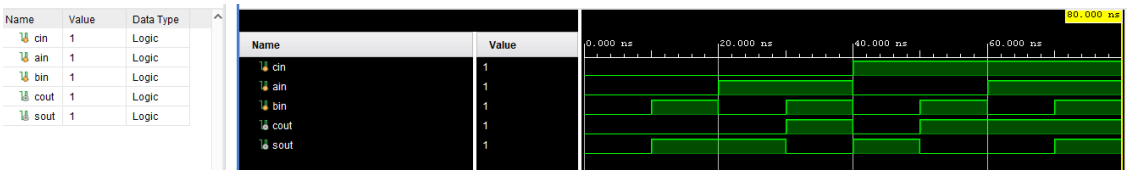

Figure 4: Full Adder Waveform and ERT

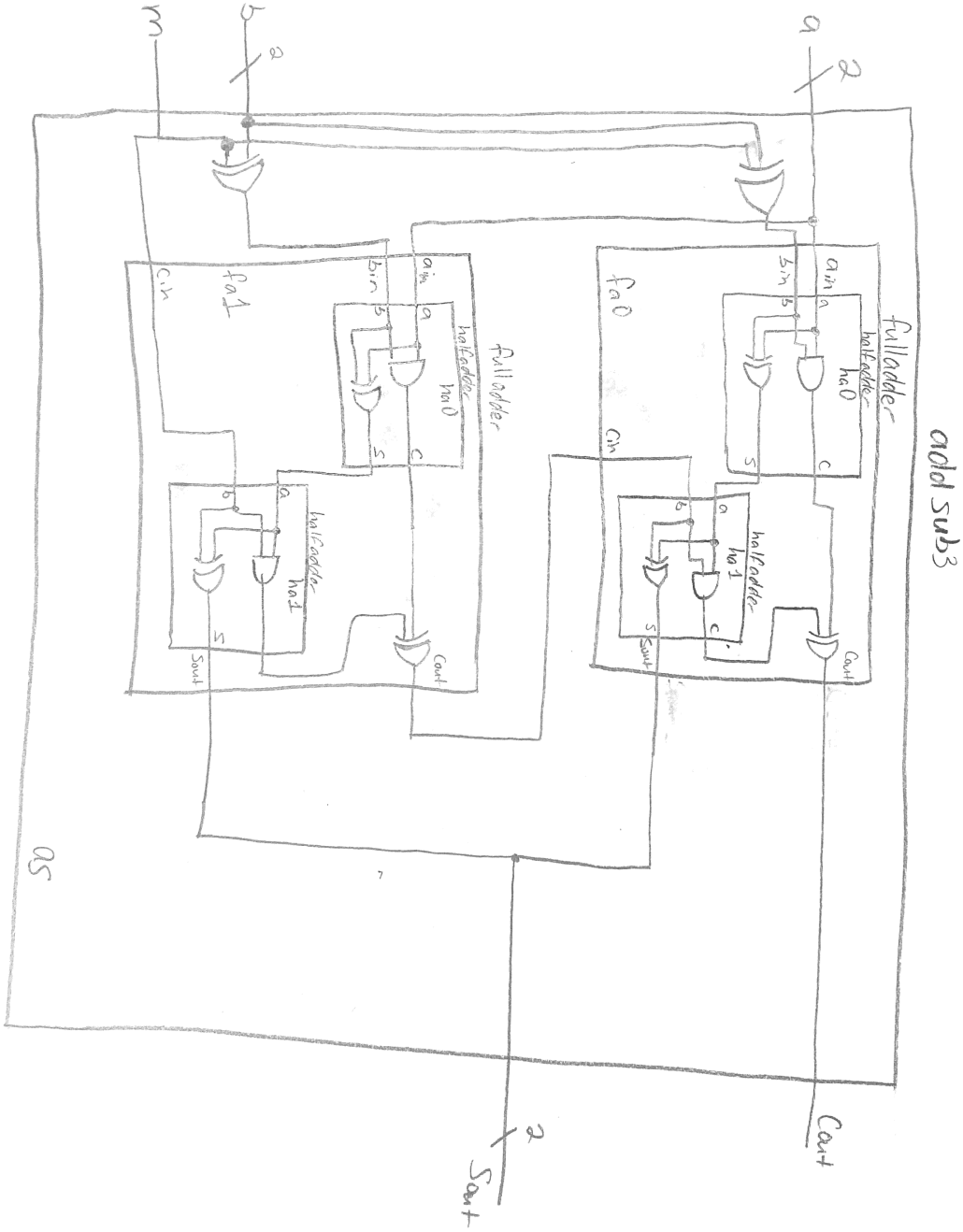

Figure 5: 2Bit Adder/Subtractor Block Diagram

Time (ns):	0	10	20	30	40	50	60	70	80	90	100	110
a1	0	0	0	1	0	0	0	0	0	1	0	0
b1	1	0	1	1	1	0	1	0	1	1	1	0
a2	0	0	0	0	1	1	0	0	0	0	1	1
b2	0	1	1	0	0	0	1	1	0	0	0	0
mode	0	0	0	0	0	0	1	1	1	1	1	1
c	0	0	0	0	0	0	1	1	1	0	0	0
s1	0	1	1	1	1	1	1	1	0	0	0	1
s2	1	0	1	0	1	0	1	0	1	0	1	0

Figure 6: 2Bit Adder SubtractorWaveform and ERT