ELC 2137 Lab #5: Intro to Verilog

Sebastian Lopez, Megan Gordon, and Jane Ross

February 20, 2020

Summary

In this lab we organized the files and adhered to the given folder structure. We had to learn the basic Verilog syntax and used it to create simple components and systems. We then created a full adder using both gate-level and functional/behavioral styles of coding. Following that we combined full adder components to create a 2-bit and 4-bit adder/subtractor. Throughout the process of the lab we also had to create test benches in order verify our designs.

Q&A

1. **Comment on whether the simulations match the expected output values.**

   In last week's lab we found the expected output values using variables a, b, and mode. This week after testing and running the code, our simulations did match the results of the expected outcome.

2. **What is the one thing you still don't understand about Verilog?**

   We don't fully understand the conditions of the if statements. We are also not fully understanding the application of a proper 'always' statement.

Results

1. All of the Verilog Code

Listing 1: Half-adder Source Code

```
`timescale 1ns / 1ps
// Sebastian Lopez, Megan Gordon , Jane Ross ELC 2137, 2020-02-19


module halfadder (input a1, b1,
output c1, s1);

xor(s1, a1, b1);
and(c1, a1, b1);


endmodule //halfadder
```

Listing 2: Half-adder Test Bench Code

```
`timescale 1ns / 1ps
// Sebastian Lopez, Megan Gordon , Jane Ross  ELC 2137, 2020-02-19
```

```
module halfadder_test ();

reg a1_in, b1_in;
wire c1_out, s1_out;

halfadder ha0(
.a1(a1_in), .b1(b1_in),
.c1(c1_out), .s1(s1_out));

initial
begin
a1_in = 0;
b1_in = 0;
#10;

//Test Case #2
a1_in = 0;
b1_in = 1;
#10
$finish;
end

endmodule //halfadder_test
```

Listing 3: Full-adder Source Code

```
`timescale 1ns / 1ps
// Jane Ross, Megan Gordon and Sebastian Lopez ELC 2137, 2020-02-19

module fulladder(
input a1_in, b1_in, cin,
output s2,cout, c2
);

wire s1, c1, c2;

xor(cout, c1, c2);

halfadder ha1(
.a1(a1_in), .b1(b1_in),
.c1(c1), .s1(s1));
halfadder ha2(
.a1(s1), .b1(cin),
.c1(c2), .s1(s2));
endmodule//fulladder
```

Listing 4: Full-adder Test Bench Code

```
`timescale 1ns / 1ps
// Sebastian Lopez, Megan Gordon , Jane Ross ELC 2137, 2020-02-19

module fulladder_test();
```

```
reg a1_in, b1_in, cin;
wire s1, c1, c2, s2, cout;

halfadder ha1(
.a1(a1_in), .b1(b1_in),
.c1(c1), .s1(s1));
halfadder ha2(
.a1(s1), .b1(cin),
.c1(c2), .s1(s2));

assign cout = c2 ^ c1;


initial
begin
//Test case #1
a1_in = 0;
b1_in = 0;
cin = 0;
#10;


//Test Case #2
a1_in = 1;
b1_in = 0;
cin = 0;
#10

//Test Case #3
a1_in = 1;
b1_in = 1;
cin = 0;
#10

//Test case #4
a1_in = 0;
b1_in = 0;
cin = 1;
#10

//Test case #5
a1_in = 1;
b1_in = 0;
cin = 1;
#10

//Test case #6
a1_in = 1;
b1_in = 1;
cin = 1;
#10

$finish;
end
```

```
endmodule//fulladder_test
```

Listing 5: 2-bit adder Source Code

```
‘timescale 1ns / 1ps
// Jane Ross, Megan Gordon and Sebastian Lopez  ELC 2137, 2020-02-19

module addsub(
input [1:0] a1, b1, a2, b2, mode,
output s2,cout2, s4
);

wire x1,x2, c2, c3, s3, c4, x3;

xor(x1,b1,mode);
xor(x2,b2,mode);
xor(x3,c4,c3);
xor(cout2,x3,mode);

fulladder fa1(
.a1(a1), .x1(x1),
.s2(s2), .c2(c2));

halfadder ha1(
.a2(a2), .x2(x2),
.s3(s3), .c3(c3));
halfadder ha2(
.s3(s3), .c2(c2),
.s4(s4), .c4(c4));

endmodule//addsub
```

Listing 6: 2-bit adder Test Bench Code

```
‘timescale 1ns / 1ps
// Jane Ross, Megan Gordon and Sebastian Lopez  ELC 2137, 2020-02-19

module addsub_test();

reg a1, b1, a2, b2, mode;
wire x1,x2, c2, c3, s3, c4, x3, s2,cout2, s4;

fulladder fa1(
.a1(a1), .x1(x1),
.s2(s2), .c2(c2));

halfadder ha1(
.a2(a2), .x2(x2),
.s3(s3), .c3(c3));
halfadder ha2(
.s3(s3), .c2(c2),
.s4(s4), .c4(c4));
```

```
assign x1 = mode ^ b1;
assign x2 = mode ^ b2;
assign x3 = c4 ^ c3;
assign cout2 = x3 ^ mode;



initial
begin

//Test #1
a1 = 0;
a2 = 0;
b1 = 1;
b2 = 0;
mode = 1;
#10

//Test #2
a1 = 0;
a2 = 0;
b1 = 0;
b2 = 1;
mode = 1;
#10

//Test #3
a1 = 0;
a2 = 0;
b1 = 1;
b2 = 1;
mode = 1;
#10

//Test #4
a1 = 1;
a2 = 0;
b1 = 1;
b2 = 0;
mode = 1;
#10

//Test #5
a1 = 0;
a2 = 1;
b1 = 1;
b2 = 0;
mode = 1;
#10

//Test #6
a1 = 0;
a2 = 1;
```

```
b1 = 0;
b2 = 0;
mode = 1;
#10

$finish;
end

endmodule//addsub_test
```

2. Block-diagrams for each Module

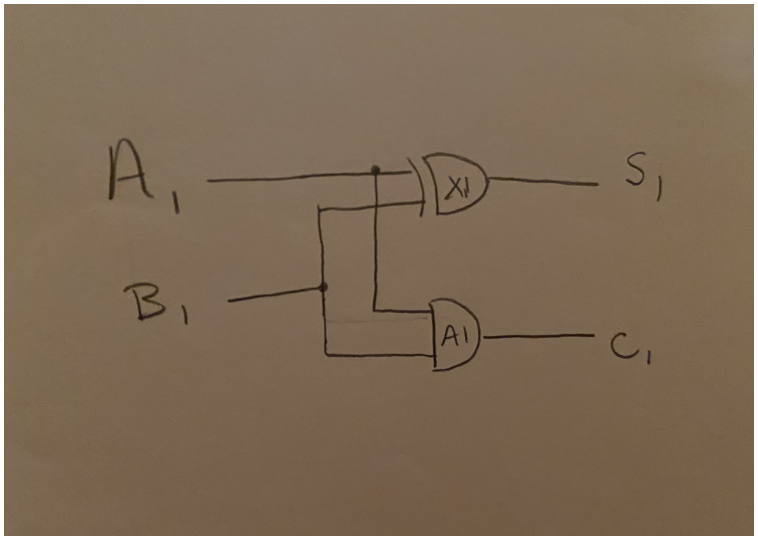

Figure 1: This is the diagram for the Half Adder

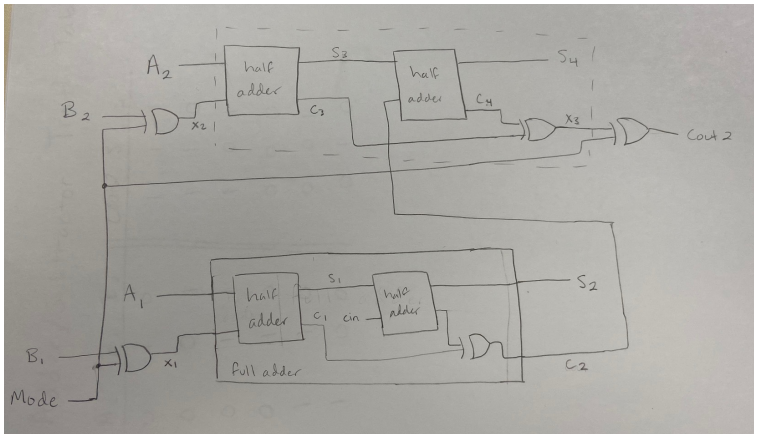

Figure 2: This is the diagram for the Full and 2-bit Adder

3. ERTs and screenshots for Behavioral Simulations

Table 1: Half Adder ERT

Time (ns)	A1	B1	C1	S1
0	0	1	0	1
10	0	0	1	1
20	0	0	0	1
30	0	1	1	0

Table 2: Full Adder ERT

Time (ns)	A1	B1	Cin	Cout	S
0	0	0	0	0	0
10	1	0	0	0	1
20	1	1	0	1	0
30	0	0	1	0	1
40	1	0	1	1	0
50	1	1	1	1	1

Table 3: Full Adder ERT

Time (ns)	A1	B1	A2	B2	Mode	Cout	S2	S4
0	0	0	0	0	0	0	0	0
10	0	1	0	0	1	1	1	1
20	0	0	0	1	1	1	1	0
30	0	1	0	1	1	1	0	1
40	1	1	0	0	0	0	0	0
50	0	1	1	0	0	0	0	1
60	0	0	1	0	0	0	1	0

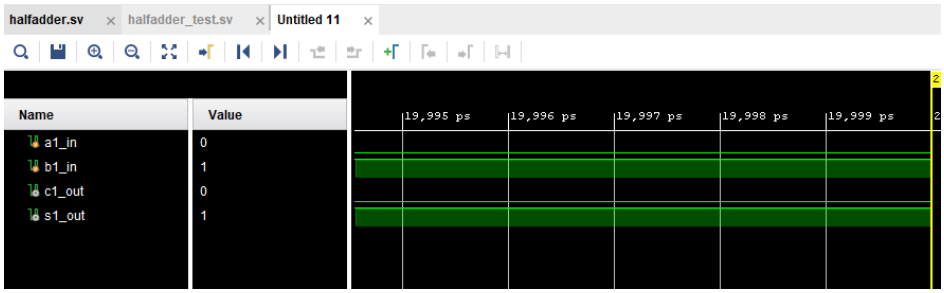

Figure 3: This is the Half Adder Simulation Screenshot.

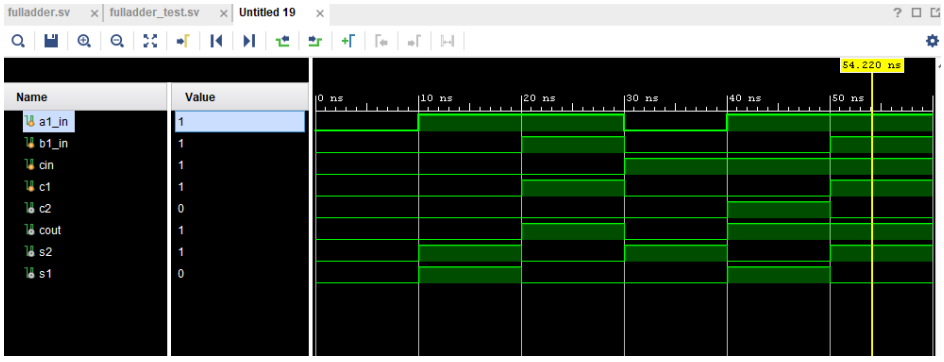

Figure 4: This is the Full Adder Simulation Screenshot.