ELC 2137 Lab 5: Intro to Verilog

Forrest Knee

March 3, 2021

Summary

This lab involved creating a half adder, full adder, and 2-bit adder/subtractor and simulating them in verilog.

Q&A

4. Do the simulations match the expected output values?

Yes, my simulations do perform as expected.

5. What is one thing that you still don't understand about verilog?

There are many many options that we were told to skip through and select certain settings when creating our project and file. I did not understand many of those choices.

Results

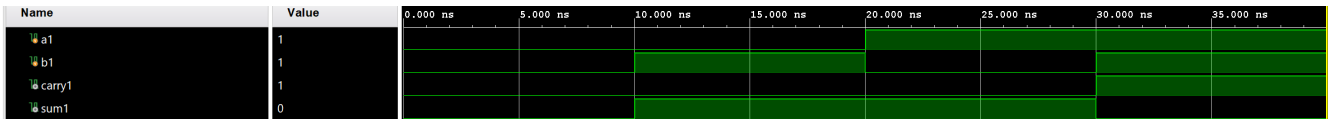


Figure 1: Half Adder

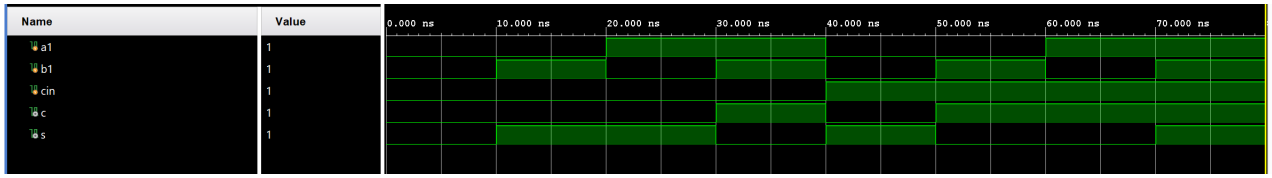


Figure 2: Full Adder

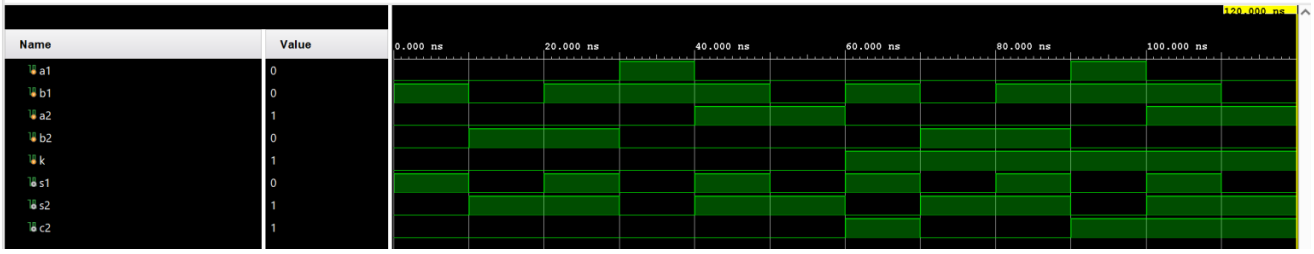
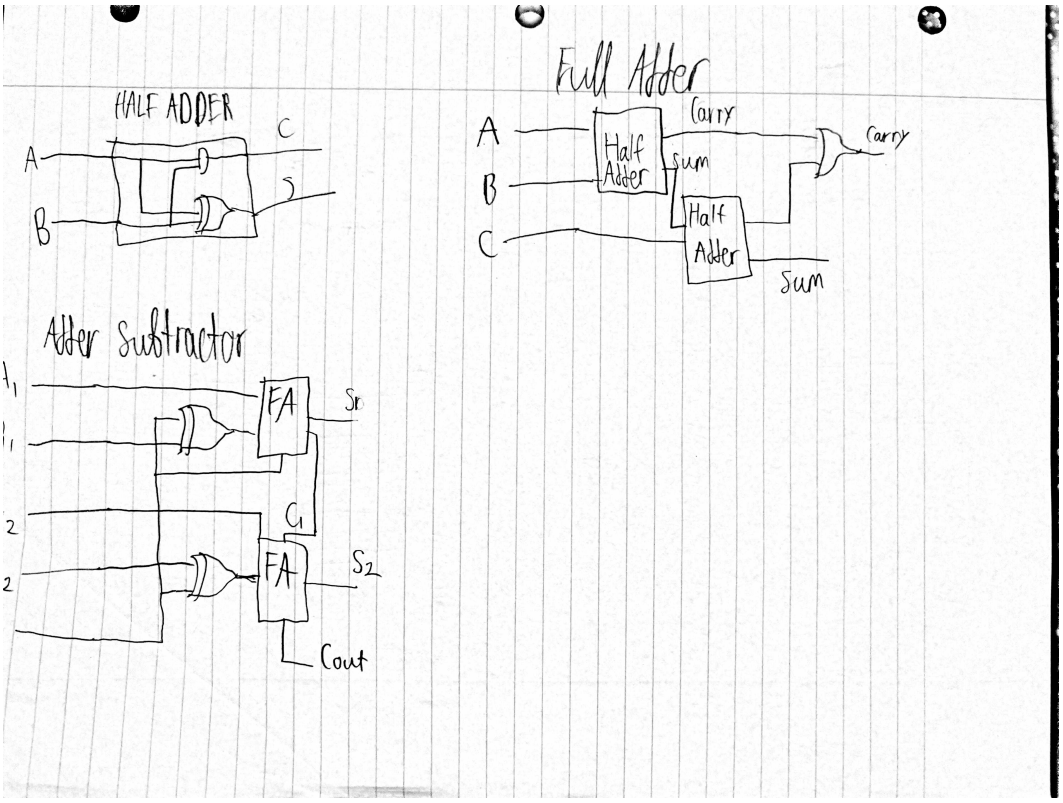


Figure 3: 2 Bit Adder Subtractor



Code

Listing 1: Half Adder

```
`timescale 1ns / 1ps
//Forrest Knee


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

    assign c=(a&b);
    assign s=(a^b);

endmodule
```

Listing 2: Half Adder Test

```
`timescale 1ns / 1ps

//Forrest Knee


module halfadder_SIM();

    reg a1,b1;
    wire carry1,sum1;

    halfadder ha1(

    .a(a1), .b(b1), .c(carry1), .s(sum1));

    initial begin

    //case 1

    a1=0; b1=0;
    #10;

    //case 2

    a1=0; b1=1;
    #10

    //case 3

    a1=1; b1=0;
    #10

    //case 4

    a1=1; b1=1;
    #10
```

```
    $finish;
    end


endmodule
```

Listing 3: Full Adder

```
`timescale 1ns / 1ps

// Forrest Knee


module fulladder(

    input a1,b1,cin,
    output s,c
    );

    wire s1,c1,c2;

    halfadder ha1(
    .a(a1), .b(b1), .c(c1), .s(s1));

    halfadder ha2(
    .a(cin), .b(s1), .c(c2), .s(s));

    assign c=(c1 | c2);

endmodule
```

Listing 4: Full Adder Test

```
`timescale 1ns / 1ps

//Forrest Knee


module fulladder_SIM();

    reg a1,b1,cin;
    wire c,s;

    fulladder fa1(

    .a1(a1), .b1(b1), .cin(cin), .c(c), .s(s));

    initial begin

    //case 1
   a1 = 0;
   b1 = 0;
   cin = 0;
   #10
```

```
    //case 2
   a1 = 0;
   b1 = 1;
   cin = 0;
   #10

    //case 3
   a1 = 1;
   b1 = 0;
   cin = 0;
   #10

    //case 4
   a1 = 1;
   b1 = 1;
   cin = 0;
   #10

    //case 5
   a1 = 0;
   b1 = 0;
   cin = 1;
   #10

    //case 6
   a1 = 0;
   b1 = 1;
   cin = 1;
   #10

    //case 7
   a1 = 1;
   b1 = 0;
   cin = 1;
   #10

    //case 8
   a1 = 1;
   b1 = 1;
   cin = 1;
   #10
  $finish;
  end


endmodule
```

Listing 5: 2 Bit Adder Subtractor

```
`timescale 1ns / 1ps

//Forrest Knee
```

```
module addersubtractor(
    input a1,b1,a2,b2,k,
    output s1,s2,c2);

    wire cb1, cb2, c1;

    assign cb1=(b1^k);
    assign cb2=(b2^k);

    fulladder fa1(
    .a1(a1), .b1(cb1), .cin(k), .c(c1), .s(s1));

    fulladder fa2(
    .a1(a2), .b1(cb2), .cin(k), .c(c2), .s(s2));




endmodule
```

Listing 6: 2 Bit Adder Subtractor Test

```
`timescale 1ns / 1ps
//Forrest Knee


module addsub_SIM();

    reg a1,b1,a2,b2,k;
    wire s1,s2,c2;

    addersubtractor as1(

    .a1(a1), .b1(b1), .a2(a2), .b2(b2), .k(k), .s1(s1), .s2(s2), .c2(c2)
    );

    initial begin

    // Case 1
    a1= 0;
    b1= 1;
    a2= 0;
    b2= 0;
    k= 0;
    #10;
    // Case 2
    a1= 0;
    b1= 0;
    a2= 0;
    b2= 1;
    k= 0;
    #10
```

```
    // Case 3
    a1= 0;
    b1= 1;
    a2= 0;
    b2= 1;
    k= 0;
    #10
    // Case 4
    a1= 1;
    b1= 1;
    a2= 0;
    b2= 0;
    k= 0;
    #10
    // Case 5
    a1= 0;
    b1= 1;
    a2= 1;
    b2= 0;
    k= 0;
    #10
    // Case 6
    a1= 0;
    b1= 0;
    a2= 1;
    b2= 0;
    k= 0;
    #10
    // Case 7
    a1= 0;
    b1= 1;
    a2= 0;
    b2= 0;
    k= 1;
    #10
    // Case 8
    a1= 0;
    b1= 0;
    a2= 0;
    b2= 1;
    k= 1;
    #10
    // Case 9
    a1= 0;
    b1= 1;
    a2= 0;
    b2= 1;
    k= 1;
    #10
    // Case 10
    a1= 1;
    b1= 1;
    a2= 0;
    b2= 0;
```

```
    k= 1;
    #10
    // Case 11
    a1= 0;
    b1= 1;
    a2= 1;
    b2= 0;
    k= 1;
    #10
    // Case 12
    a1= 0;
    b1= 0;
    a2= 1;
    b2= 0;
    k= 1;
    #10

$finish;

end

endmodule
```