ELC 2137 Lab 05: Intro to Verilog

Kyra Rose

October 20, 2020

Summary

In this lab, we are introduced to coding verilog in Vivado. We built the three circuits, half adder, full adder, and 2 bit adder, in Vivado and observed their behavioral simulations. After viewing their simulations, I can conclude that the results were as I expected in my truth table.

Q&A

What is one thing that you still don't understand about Verilog?

I still don't understand what exactly a test bench is a how to make an instance.

Code

Listing 1: Half Adder code

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

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

module halfadder_test();
reg a;
reg b;
wire c;
wire s;

halfadder dut(
.a(a),
.b(b),
.c(c),
.s(s)
);

initial begin
a=0;b=0;#10;
a=0;b=1;#10;
```

```
a=1;b=0;#10;
a=1;b=1;#10;
$finish;
end
```

Listing 2: Full Adder code

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

wire c1;
wire c2;
wire s1;

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

halfadder ha1 (
.a(ain),
.b(bin),
.c(c2),
.s(sout)
);

assign cout= c1 | c2;

endmodule

module fulladder_test();

reg cin;
reg a;
reg b;
wire cout;
wire s;

fulladder dut(
.cin(cin),
.ain(a),
.bin(b),
.cout(cout),
.sout(s)
);

initial begin
```

```
   cin=0; a=0; b=0; #10;
   cin=0; a=0; b=1; #10;
   cin=0; a=1; b=0; #10;
   cin=0; a=1; b=1; #10;
   cin=1; a=0; b=0; #10;
   cin=1; a=0; b=1; #10;
   cin=1; a=1; b=0; #10;
   cin=1; a=1; b=1; #10;

   $finish;
   end
```

Listing 3: Adder Subtractor code

```
module addsub(
input [1:0] a,
input [1:0] b,
input mode,
output [1:0] sum,
output cbout
);

wire c1;
wire c2;
wire [1:0] b_n;
wire y1;
wire y0;
wire cout;

assign b_n = ~b;
assign y0 = b_n[0] ^ mode;
assign y1 = b_n[1] ^ mode;

fulladder fa0 (
.cin(mode),
.ain(a[0]),
.bin(y0),
.cout(c1),
.sout(sum[0])
);

fulladder fa1 (
.cin(mode),
.ain(a[1]),
.bin (y1),
.cout(cbout),
.sout(sum[1])
);

assign cbout = mode ^ cout;


endmodule
```

```
module addsub_test();

reg [1:0] a;
reg [1:0] b;
reg mode;
wire [1:0] sum;
wire cbout;

addsub dut(
.a(a),
.b(b),
.mode(mode),
.sum(sum),
.cbout(cbout)
);

initial begin

a=2'b00; b=2'b01; mode=0; #10; mode=1; #10;
a=2'b00; b=2'b10; mode=0; #10; mode=1; #10;
a=2'b00; b=2'b11; mode=0; #10; mode=1; #10;
a=2'b01; b=2'b01; mode=0; #10; mode=1; #10;
a=2'b10; b=2'b01; mode=0; #10; mode=1; #10;
a=2'b10; b=2'b00; mode=0; #10; mode=1; #10;

$finish;
end

endmodule
```

Results

Table 1: Half Adder Truth Table

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

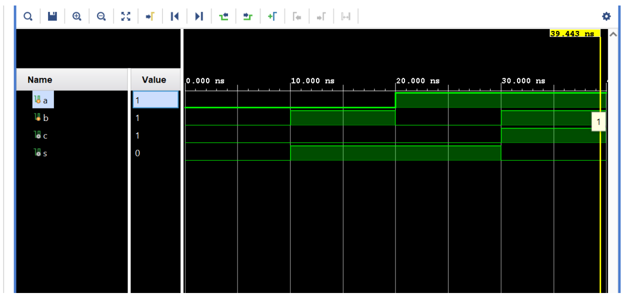


Figure 1: Half Adder Behavioral Simulation

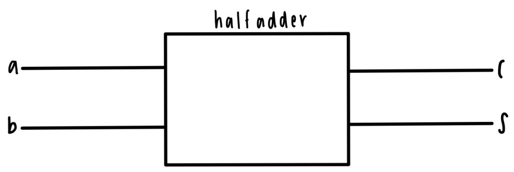
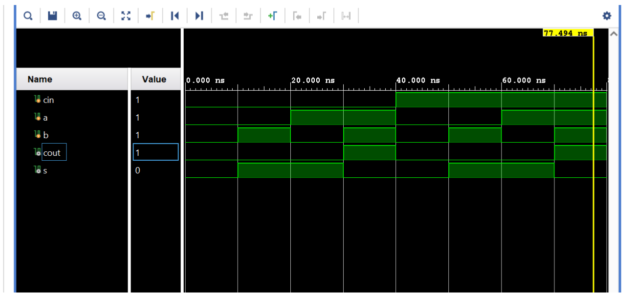


Figure 2: Half Adder Block Diagram

Table 2: Full Adder Truth Table

Cin	A	B	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Name	Value
cin	1
a	1
b	1
cout	1
s	0

Figure 3: Full Adder Behavioral Simulation

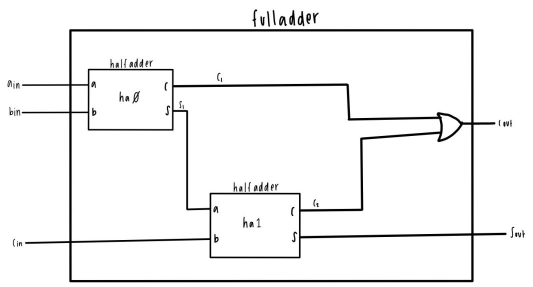


Figure 4: Full Adder Block Diagram

Table 3: Adder Subtracter Truth Table

A	B	B 2's comp	Sub (expected)	Dec	Sub (actual)
00	01	111	111	-7	011
00	10	110	110	-6	010
00	11	101	101	-5	001
01	01	111	000	0	100
10	01	111	001	1	101
10	00	000	010	2	110

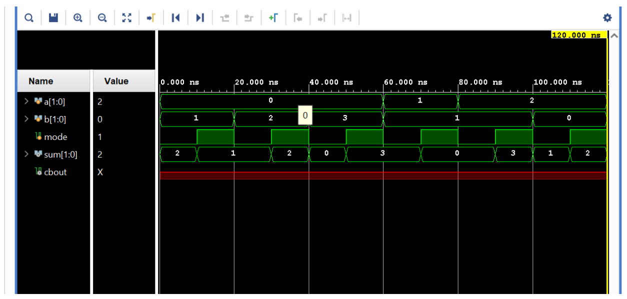


Figure 5: Adder Subtractor Behavioral Simulation

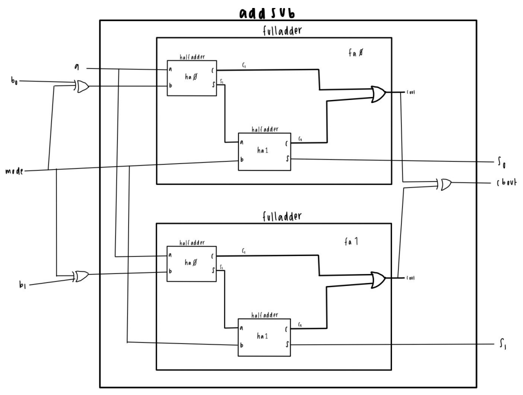


Figure 6: Adder Subtractor Block Diagram