30 Days of RTL Coding

-By T Tushar Shenoy

Day 2

Problem Statement: Implementing Adders and Subtractors

Theory:

1. **Half Adder**: Half Adder is a combinational logic circuit which is designed by connecting one EX-OR gate and one AND gate. The half adder circuit has two inputs: A and B, which add two input digits and generates a carry and a sum. The output obtained from the EX-OR gate is the sum of the two numbers while that obtained by AND gate is the carry. There will be no forwarding of carry addition because there is no logic gate to process that. Thus, this is called the Half Adder circuit.

Truth Table			
Input		Output	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Fig: Half Adder Truth Table

2. **Full Adder**: Full Adder is the circuit that consists of two EX-OR gates, two AND gates, and one OR gate. Full Adder is the adder that adds three inputs and produces two outputs which consist of two EX-OR gates, two AND gates, and one OR gate. The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C-OUT and the normal output is designated as S which is SUM.

Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Fig: Full Adder Truth Table

3. **Half subtractor**: A half subtractor is a digital logic circuit that performs binary subtraction of two single-bit binary numbers. It has two inputs, A and B, and two outputs, DIFFERENCE and BORROW. The DIFFERENCE output is the difference between the two input bits, while the BORROW output indicates whether borrowing was necessary during the subtraction. The half subtractor can be implemented using basic gates such as XOR and NOT gates. The DIFFERENCE output is the XOR of the two inputs A and B, while the BORROW output is the NOT of input A and the AND of inputs A and B.

A	B	Diff	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Fig: Half Subtractor Truth Table

4. **Full subtractor**: A full subtractor is a combinational circuit that performs subtraction of two bits, one is minuend and other is subtrahend, taking into account borrow of the previous adjacent lower minuend bit. This circuit has three inputs and two outputs. The three inputs A, B and Bin, denote the minuend, subtrahend, and previous borrow, respectively. The two outputs, D and Bout represent the difference and output borrow, respectively.

INPUT			OUTPUT	
A	B	Bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Fig: Full Subtractor Truth Table

1. Half-Adder

Verilog Code:

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

//Gate Level Implementation
xor(s,a,b);
and(c,a,b);

endmodule
```

Testbench Code:

```
module half_adder_tb();

reg a,b;
wire s,c;

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

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

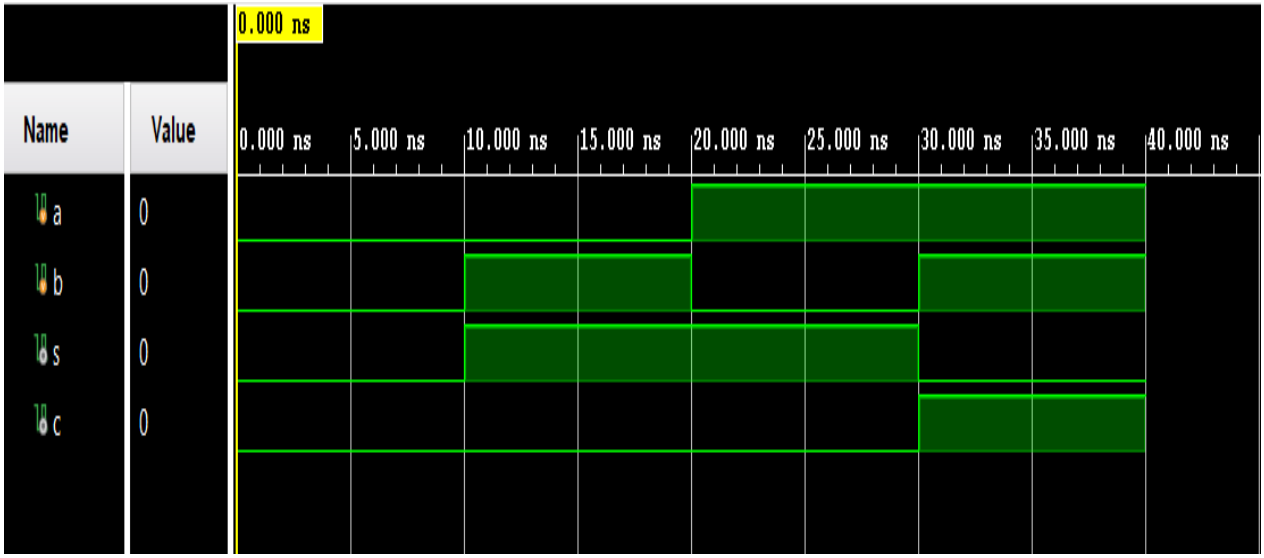
#10 a=1; b=1;

#10 $finish;

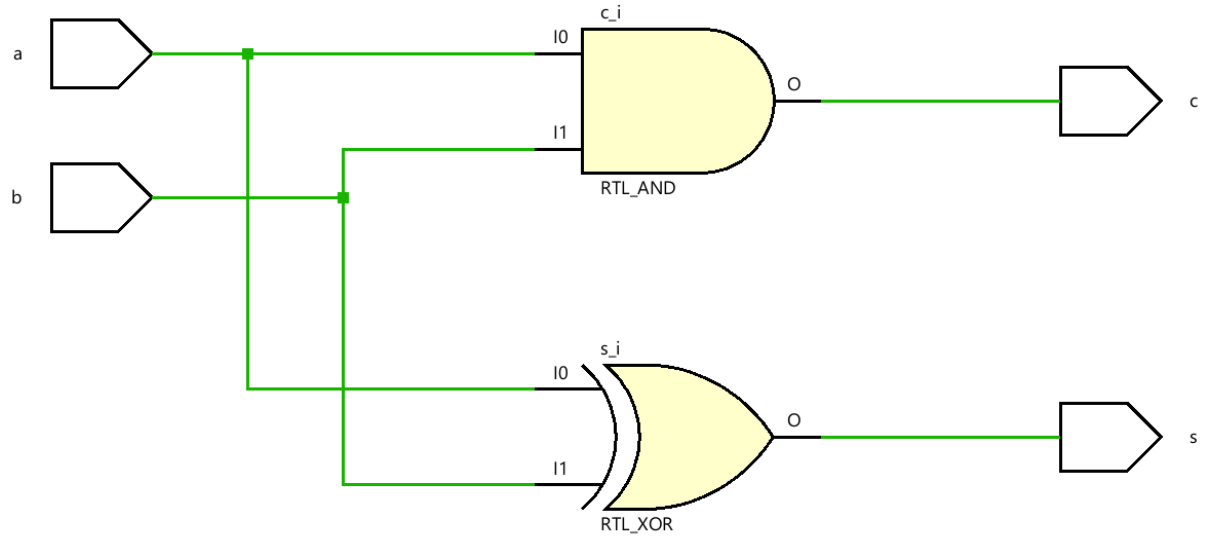
end

endmodule

Simulation Output:



Schematic:



2. Full-Adder

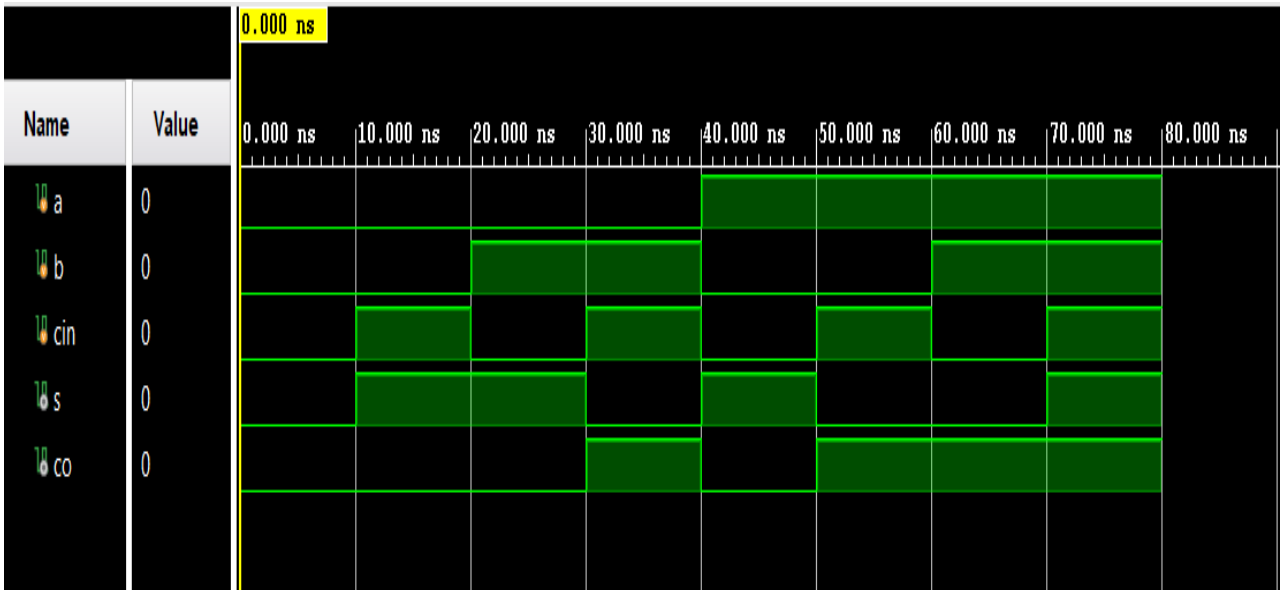
Verilog Code:

```
module full_adder(a,b,cin,s,co);
input a,b,cin;
output s,co;

//Gate Level Implementation
xor(s,a,b,cin);

wire w1,w2,w3;
and(w1,a,b);
and(w2,b,cin);
and(w3,a,cin);

or(co,w1,w2,w3);

endmodule
```

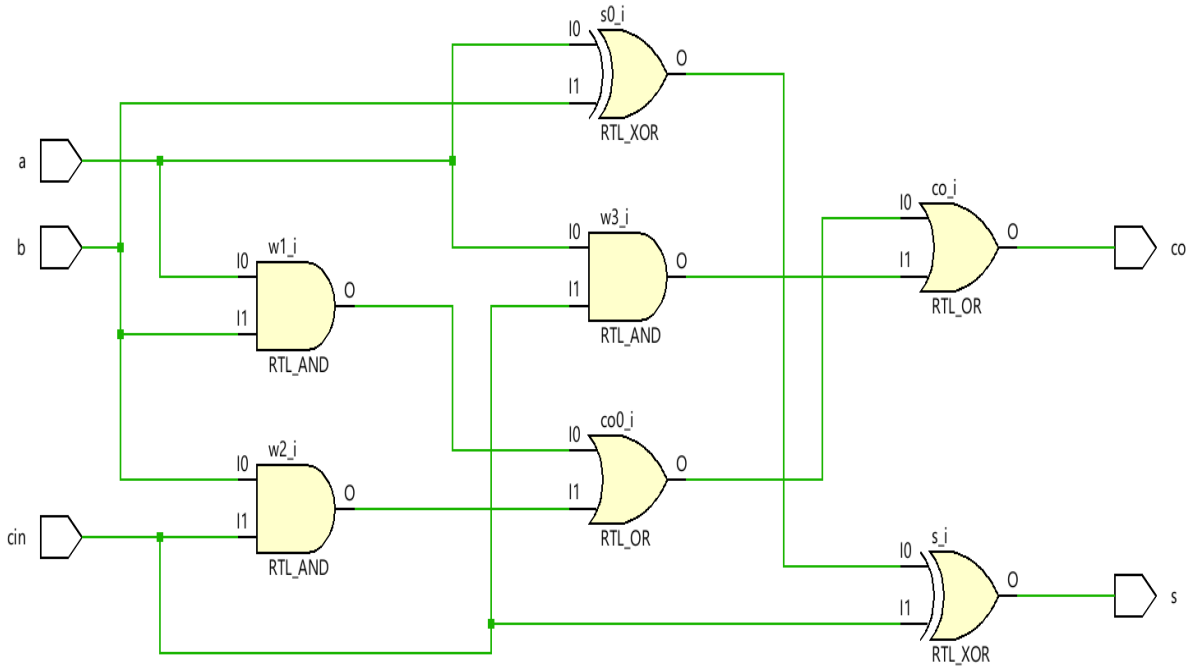
Testbench Code:

```
module full_adder_tb();

reg a,b,cin;
wire s,co;

full_adder dut(.a(a),.b(b),.cin(cin),.s(s),.co(co));

initial begin
//stimulus
    a=0; b=0;cin=0;
#10 a=0; b=0;cin=1;
#10 a=0; b=1;cin=0;
#10 a=0; b=1;cin=1;
#10 a=1; b=0;cin=0;
#10 a=1; b=0;cin=1;
#10 a=1; b=1;cin=0;
#10 a=1; b=1;cin=1;
#10 $finish;

end

endmodule
```

Simulation Output:



Schematic:



3. Half-Subtractor

Verilog Code:

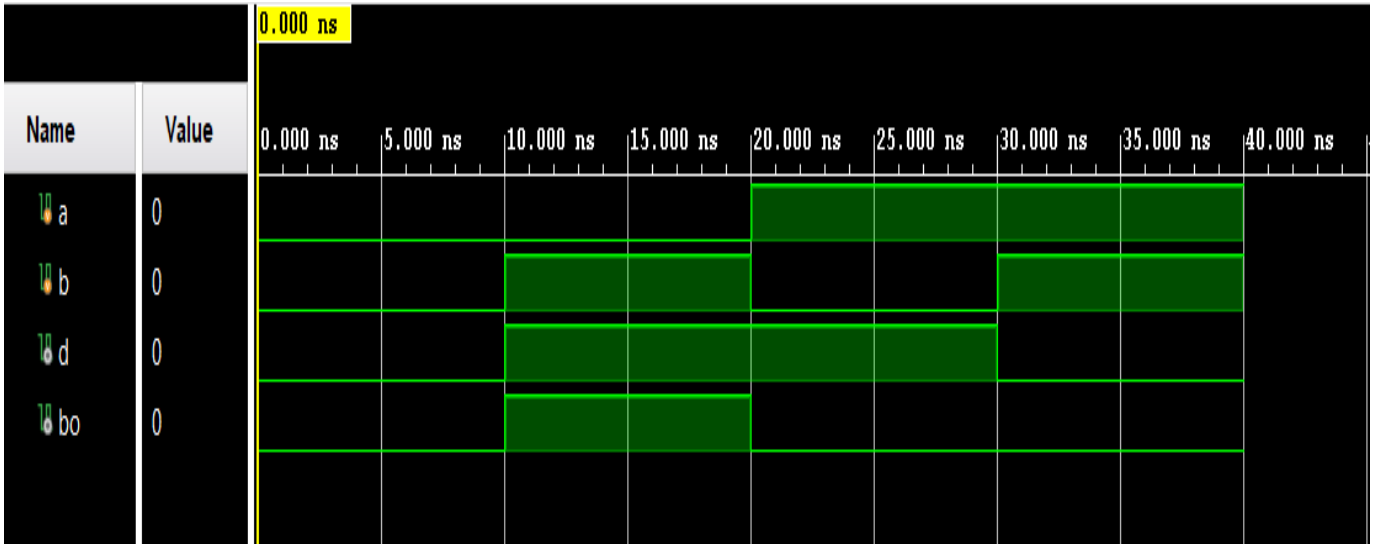
```
module half_subtractor(a,b,d,bo);
input a,b;
output d,bo;

//Gate Level Implementation
xor(d,a,b);
and(bo,~a,b);

endmodule
```

Testbench Code:

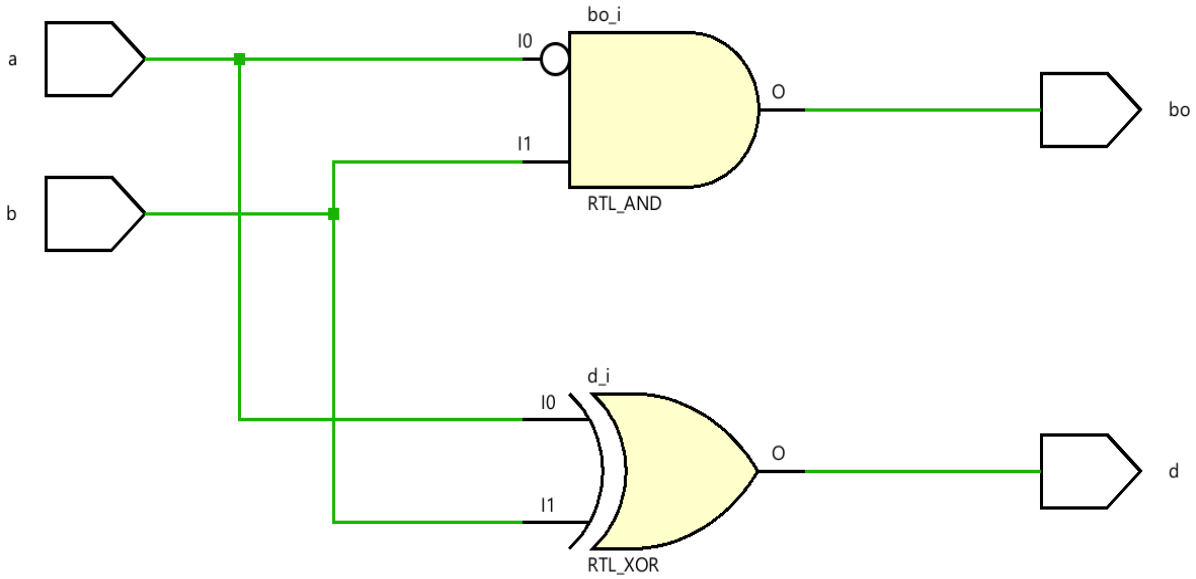
```
module half_subtractor_tb();

reg a,b;
wire d,bo;

half_subtractor dut(.a(a),.b(b),.d(d),.bo(bo));

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

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

Simulation Output:



Schematic:



4. Full-Subtractor

Verilog Code:

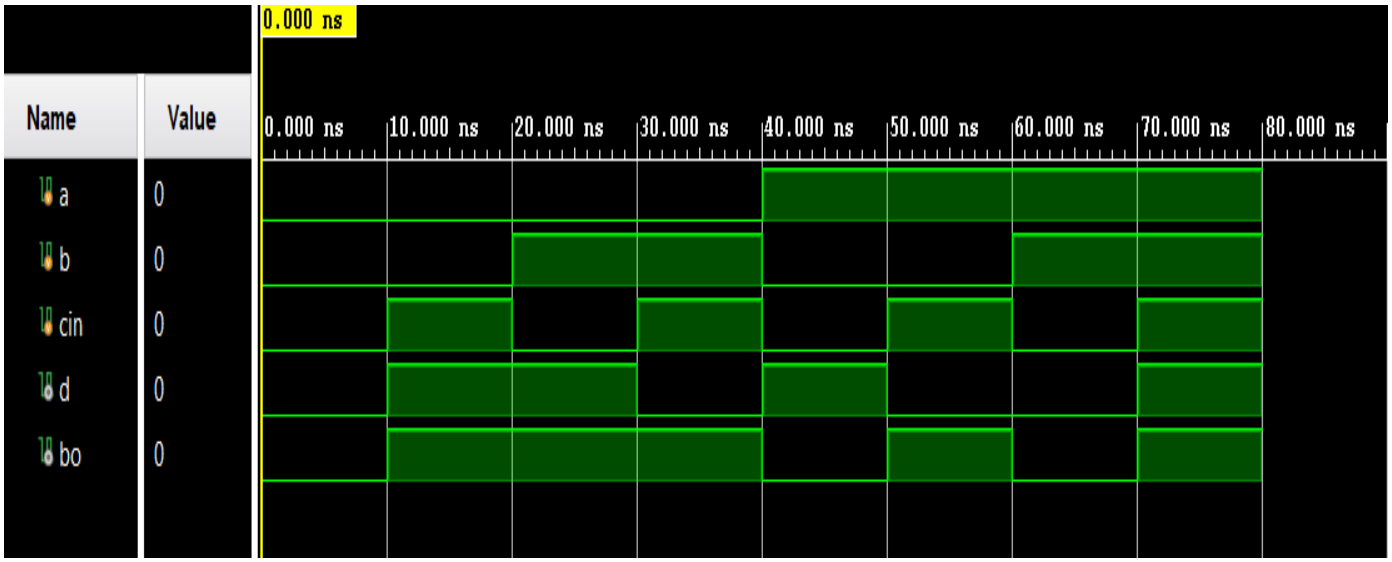
```
module full_subtractor(a,b,cin,d,bo);
input a,b,cin;
output d,bo;

//Gate Level Implementation
xor(d,a,b,cin);

wire w1,w2,w3;
and(w1,~a,b);
and(w2,~b,cin);
and(w3,b,cin);

or(bo,w1,w2,w3);

endmodule
```

Testbench Code:

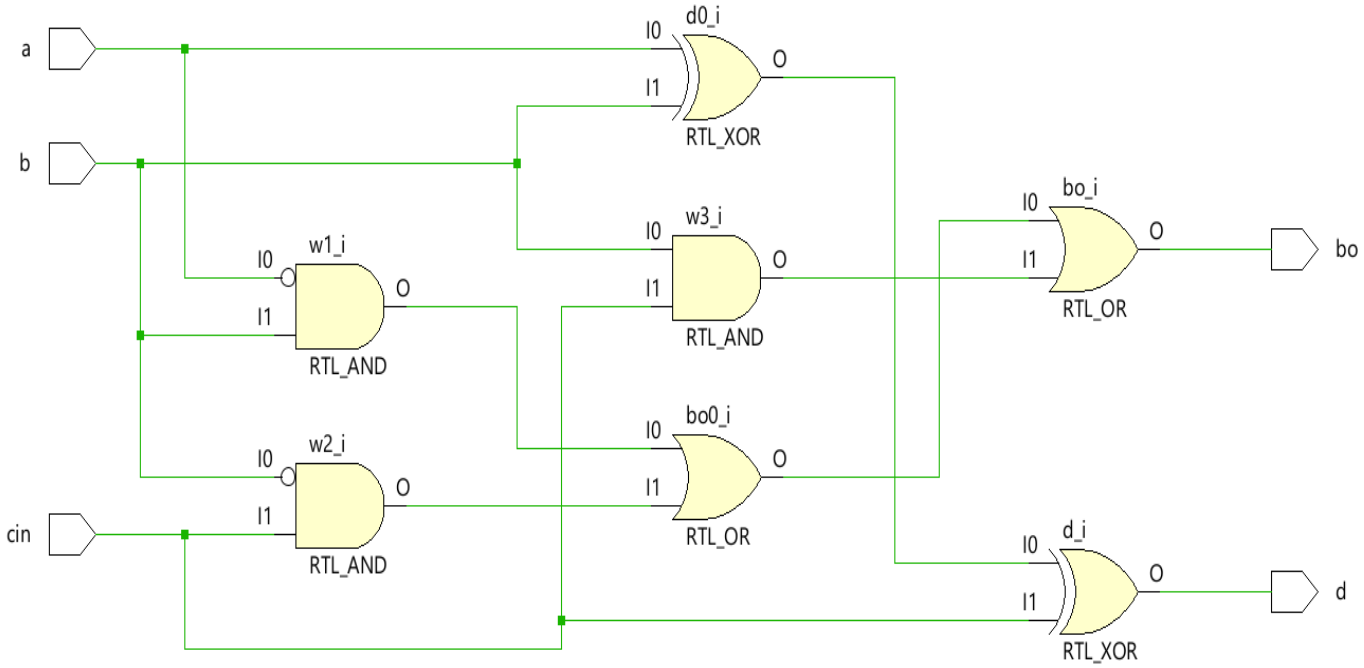
```
module full_subtractor_tb();

reg a,b,cin;
wire d,bo;

full_subtractor dut(.a(a),.b(b),.cin(cin),.s(s),.co(co));
```

```
initial begin
//stimulus
    a=0; b=0;cin=0;
#10 a=0; b=0;cin=1;
#10 a=0; b=1;cin=0;
#10 a=0; b=1;cin=1;
#10 a=1; b=0;cin=0;
#10 a=1; b=0;cin=1;
#10 a=1; b=1;cin=0;
#10 a=1; b=1;cin=1;
#10 $finish;

end
endmodule
```

Simulation Output:



Schematic:



GitHub Repository Url :-https://github.com/tusharshenoy/RTL-Day-2-Adder-Subtractor