Digital Design and Computer Organization Laboratory

3rd Semester, Academic Year 2024

Date: 23/08/2024

Name: Rithvik Rajesh Matta	SRN: PES2UG23CS485	Section: H

Week#____3_______

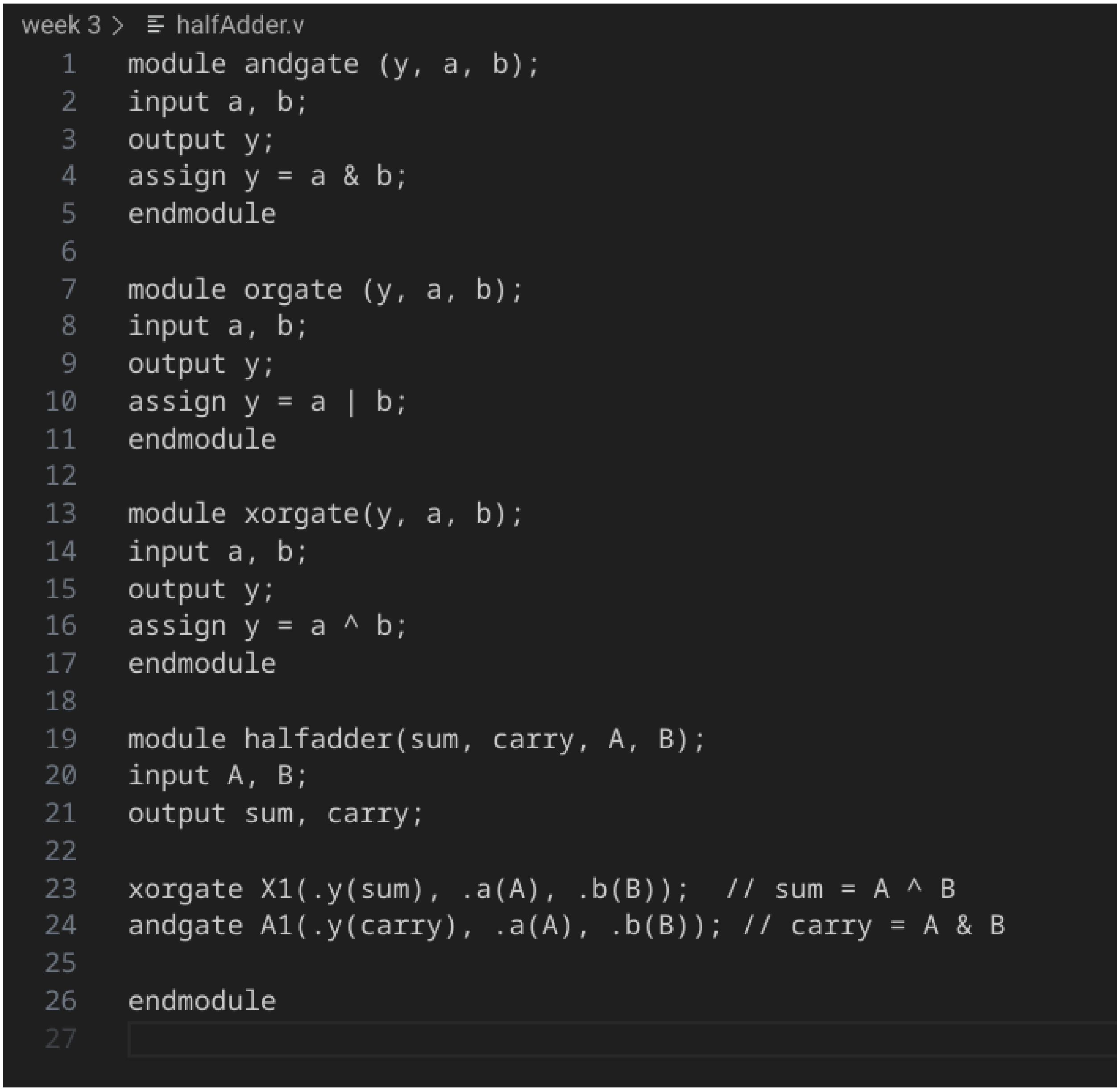
TITLE:

implement half adder, full adder and ripple carry order for 4 bit number.

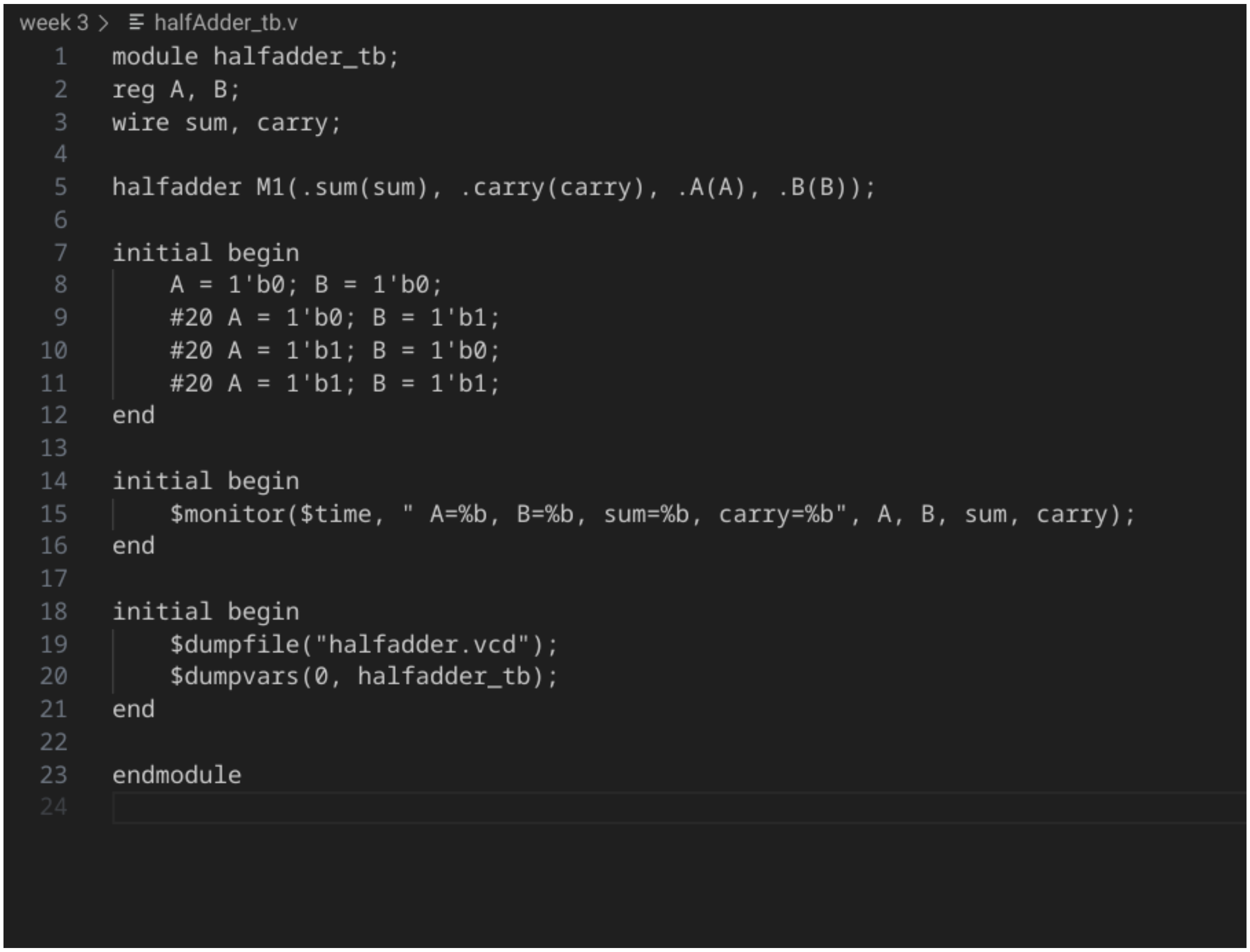
GENERATE THE VVP OUTPUT AND SIMULATION WAVEFORM USING GTKWAVE. VERIFY THE OUTPUT AND WAVEFORM WITH THE AND GATE TRUTH TABLE

1)HALF ADDER

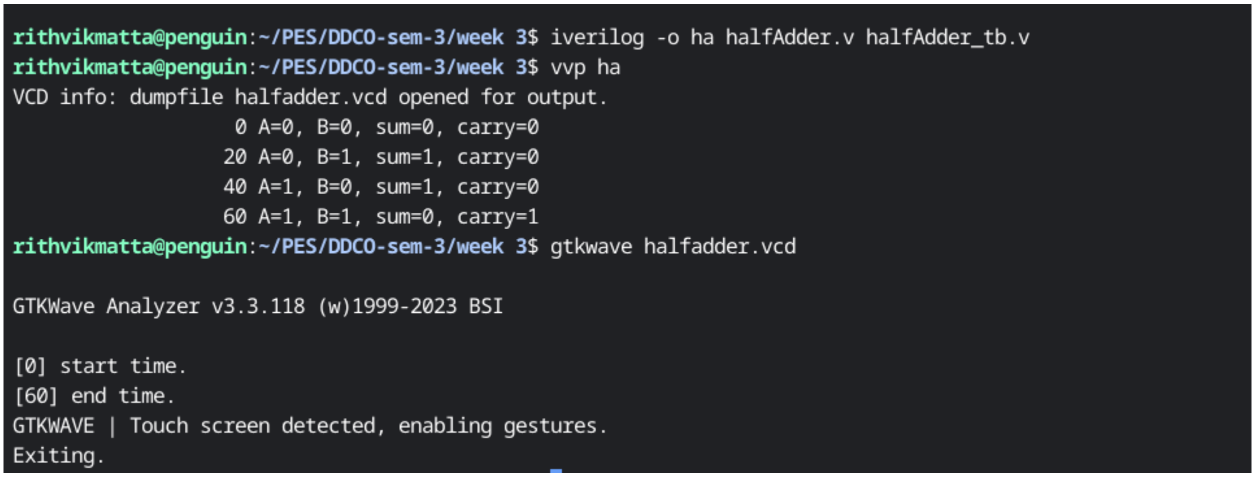
A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



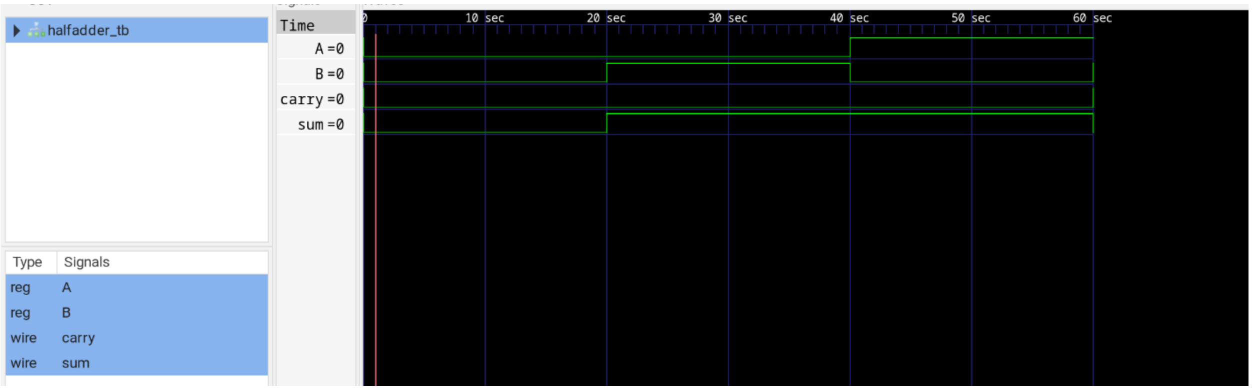
```verilog
module andgate (y, a, b);
input a, b;
output y;
assign y = a & b;
endmodule

module orgate (y, a, b);
input a, b;
output y;
assign y = a | b;
endmodule

module xorgate(y, a, b);
input a, b;
output y;
assign y = a ^ b;
endmodule

module halfadder(sum, carry, A, B);
input A, B;
output sum, carry;

xorgate X1(.y(sum), .a(A), .b(B));  // sum = A ^ B
andgate A1(.y(carry), .a(A), .b(B)); // carry = A & B

endmodule

```



```
module halfadder_tb;
reg A, B;
wire sum, carry;

halfadder M1(.sum(sum), .carry(carry), .A(A), .B(B));

initial begin
    A = 1'b0; B = 1'b0;
    #20 A = 1'b0; B = 1'b1;
    #20 A = 1'b1; B = 1'b0;
    #20 A = 1'b1; B = 1'b1;
end

initial begin
    $monitor($time, " A=%b, B=%b, sum=%b, carry=%b", A, B, sum, carry);
end

initial begin
    $dumpfile("halfadder.vcd");
    $dumpvars(0, halfadder_tb);
end

endmodule
```



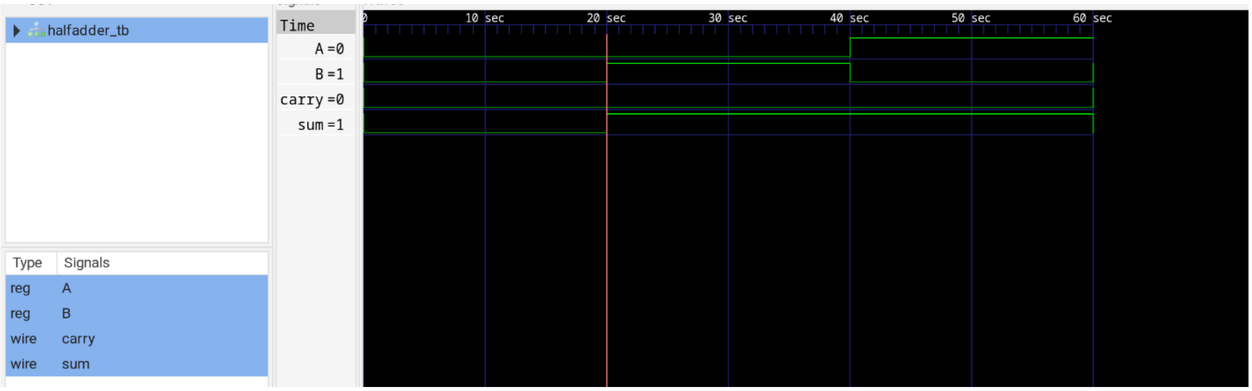
```
rithvikmatta@penguin:~/PES/DDCO-sem-3/week 3$ iverilog -o ha halfAdder.v halfAdder_tb.v
rithvikmatta@penguin:~/PES/DDCO-sem-3/week 3$ vvp ha
VCD info: dumpfile halfadder.vcd opened for output.
                   0 A=0, B=0, sum=0, carry=0
                  20 A=0, B=1, sum=1, carry=0
                  40 A=1, B=0, sum=1, carry=0
                  60 A=1, B=1, sum=0, carry=1
rithvikmatta@penguin:~/PES/DDCO-sem-3/week 3$ gtkwave halfadder.vcd

GTKWave Analyzer v3.3.118 (w)1999-2023 BSI

[0] start time.
[60] end time.
GTKWAVE | Touch screen detected, enabling gestures.
Exiting.
```



halfadder_tb

Type	Signals
reg	A
reg	B
wire	carry
wire	sum

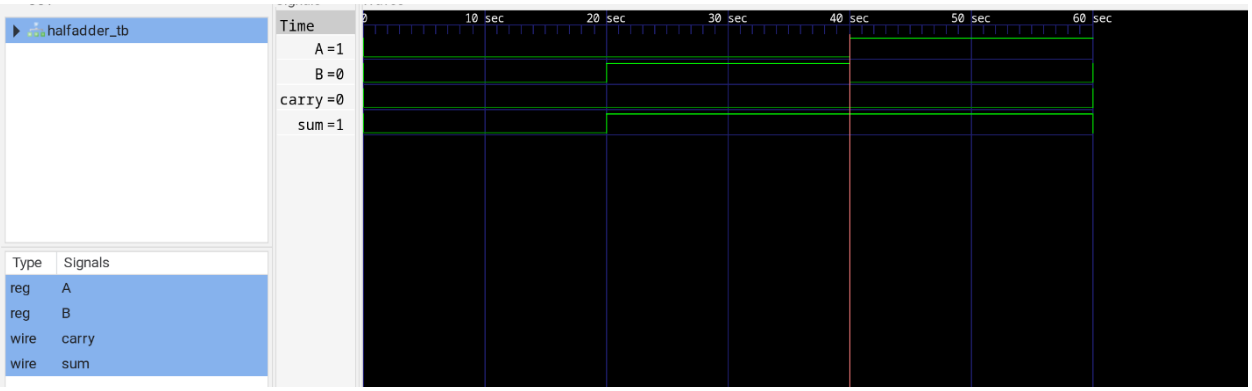
```
Time
   A =0
   B =0
carry =0
  sum =0
```



halfadder_tb

Type	Signals
reg	A
reg	B
wire	carry
wire	sum

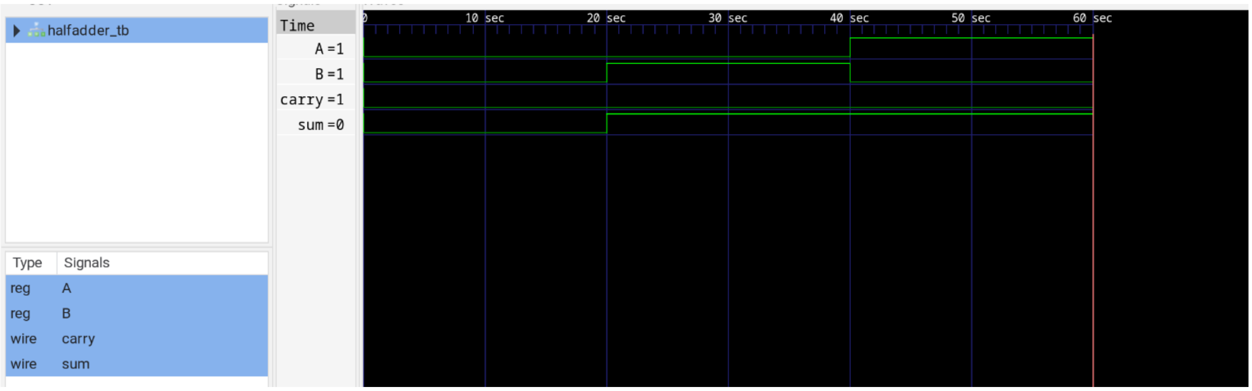
```
Time
   A =0
   B =1
carry =0
  sum =1
```



halfadder_tb

Type	Signals
reg	A
reg	B
wire	carry
wire	sum

```
Time
   A =1
   B =0
carry =0
  sum =1
```



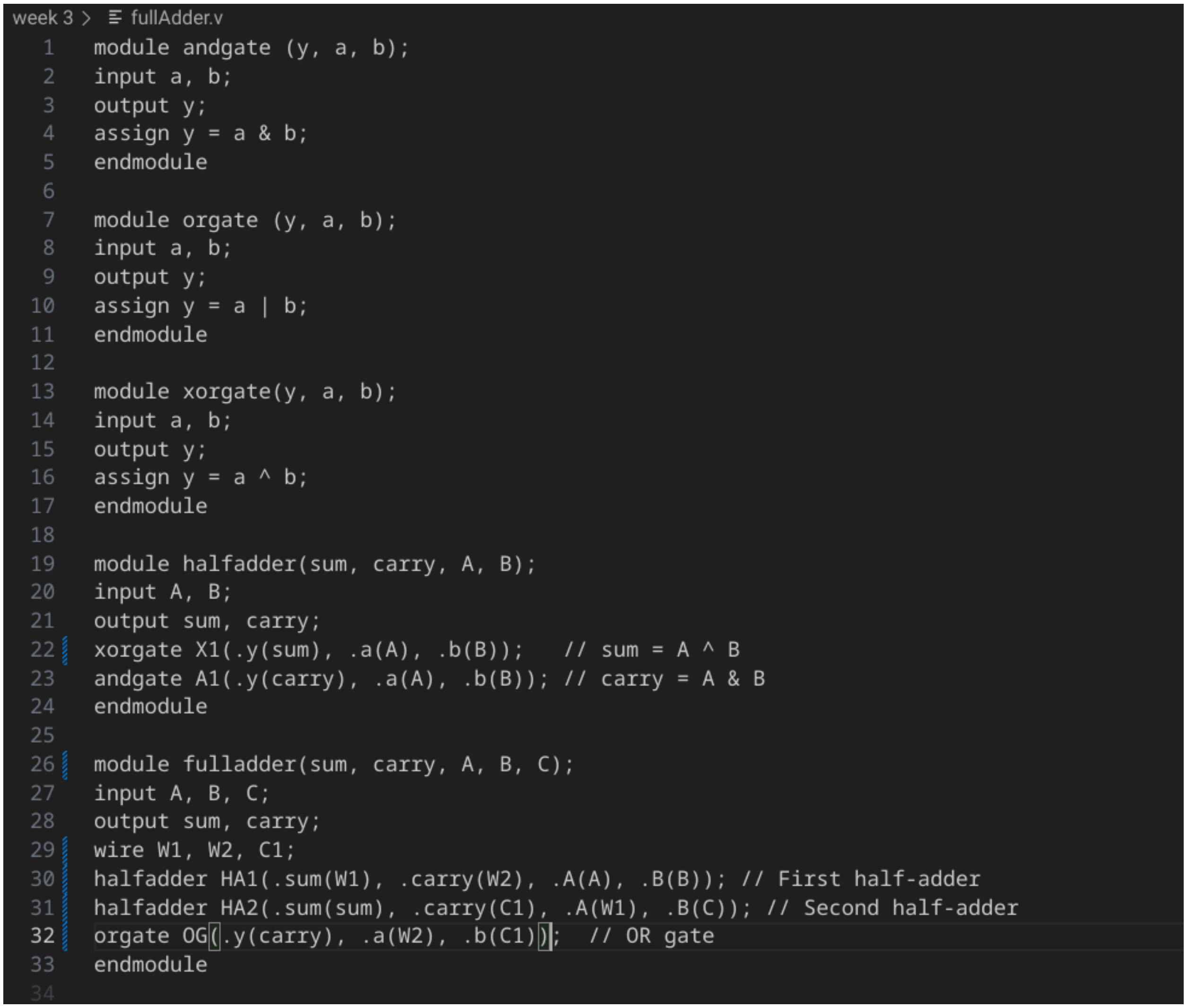
halfadder_tb

Type	Signals
reg	A
reg	B
wire	carry
wire	sum

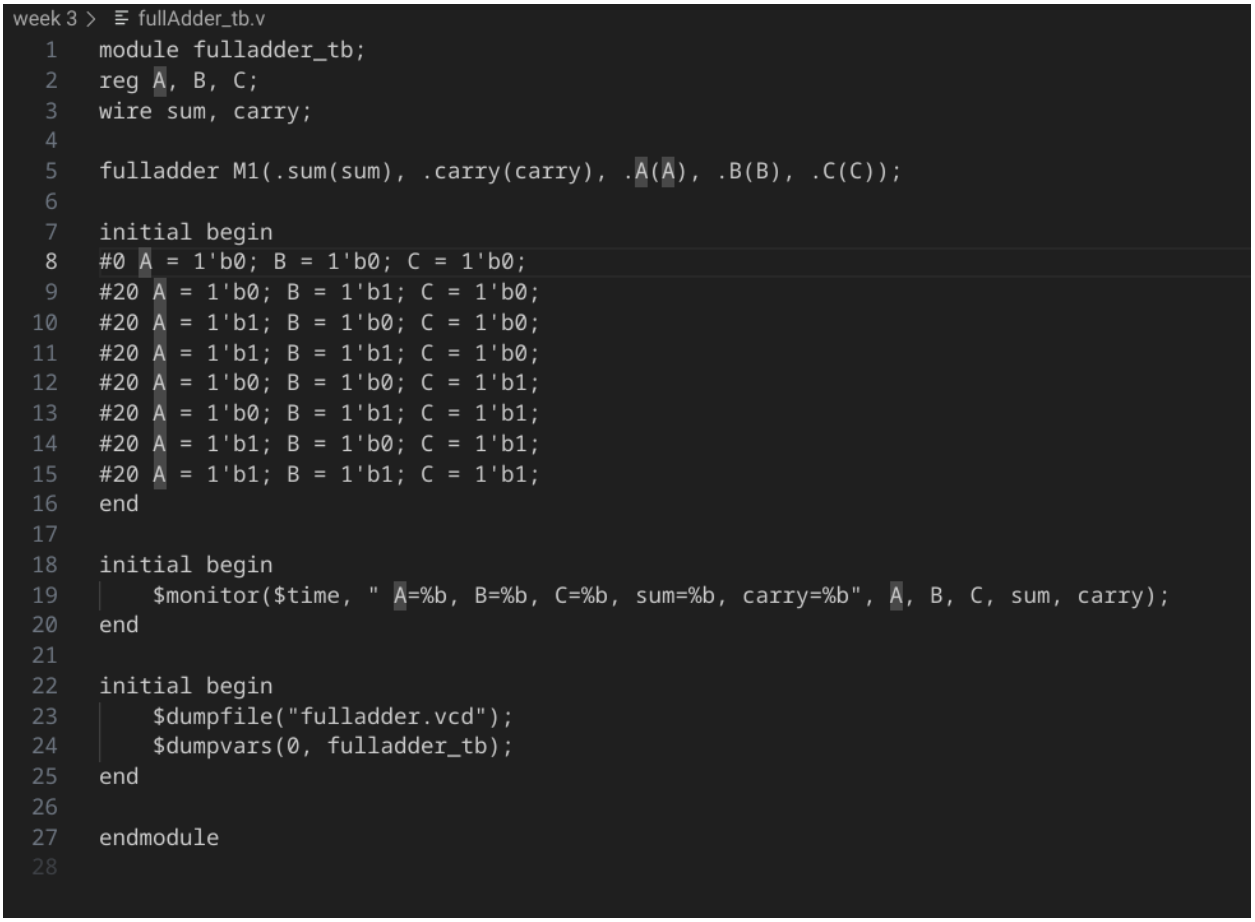
```
Time
   A =1
   B =1
carry =1
  sum =0
```

2)FULL ADDER

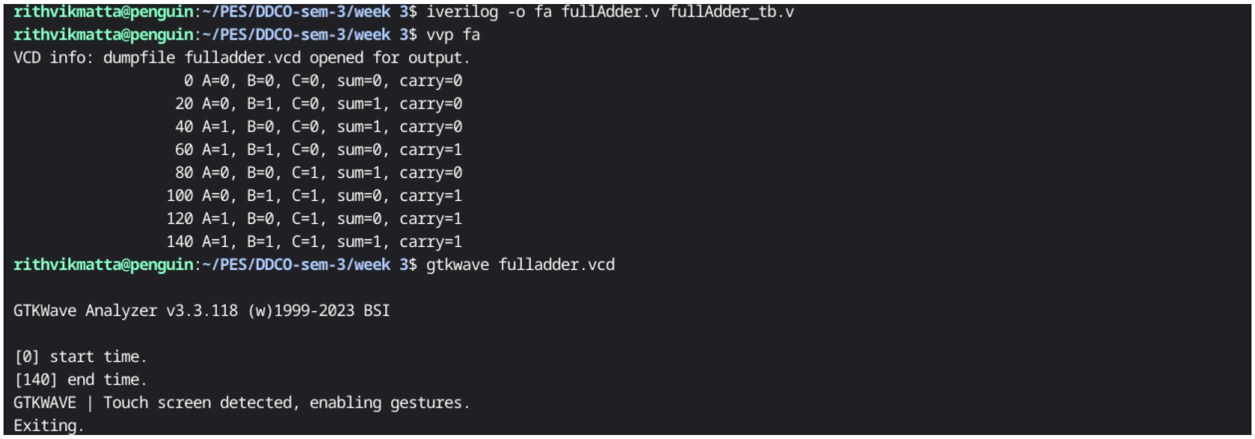
A	B	C	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



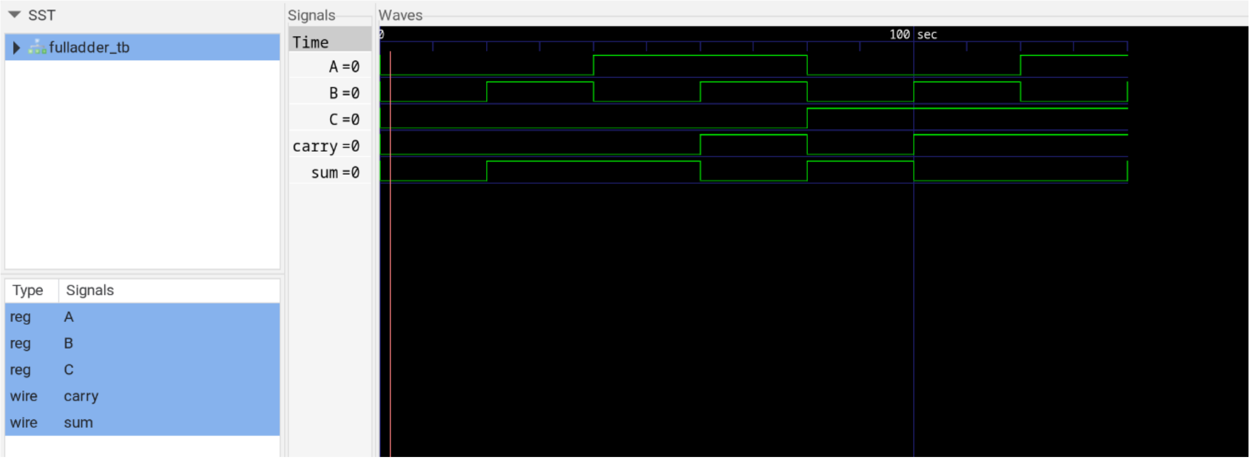
```verilog
module andgate (y, a, b);
input a, b;
output y;
assign y = a & b;
endmodule

module orgate (y, a, b);
input a, b;
output y;
assign y = a | b;
endmodule

module xorgate(y, a, b);
input a, b;
output y;
assign y = a ^ b;
endmodule

module halfadder(sum, carry, A, B);
input A, B;
output sum, carry;
xorgate X1(.y(sum), .a(A), .b(B));   // sum = A ^ B
andgate A1(.y(carry), .a(A), .b(B)); // carry = A & B
endmodule

module fulladder(sum, carry, A, B, C);
input A, B, C;
output sum, carry;
wire W1, W2, C1;
halfadder HA1(.sum(W1), .carry(W2), .A(A), .B(B)); // First half-adder
halfadder HA2(.sum(sum), .carry(C1), .A(W1), .B(C)); // Second half-adder
orgate OG(.y(carry), .a(W2), .b(C1));  // OR gate
endmodule
```

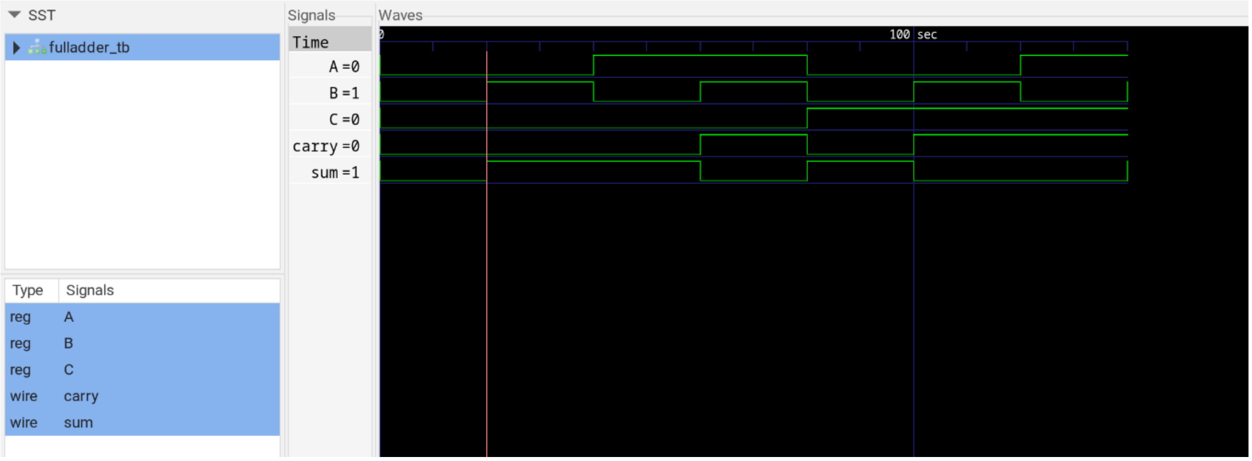


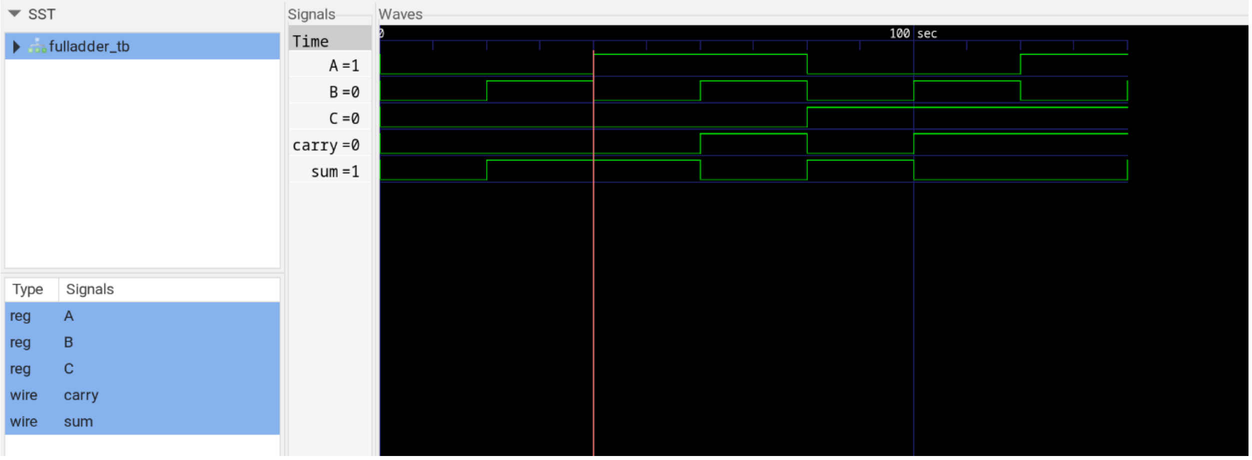
```
module fulladder_tb;
reg A, B, C;
wire sum, carry;

fulladder M1(.sum(sum), .carry(carry), .A(A), .B(B), .C(C));

initial begin
#0 A = 1'b0; B = 1'b0; C = 1'b0;
#20 A = 1'b0; B = 1'b1; C = 1'b0;
#20 A = 1'b1; B = 1'b0; C = 1'b0;
#20 A = 1'b1; B = 1'b1; C = 1'b0;
#20 A = 1'b0; B = 1'b0; C = 1'b1;
#20 A = 1'b0; B = 1'b1; C = 1'b1;
#20 A = 1'b1; B = 1'b0; C = 1'b1;
#20 A = 1'b1; B = 1'b1; C = 1'b1;
end

initial begin
    $monitor($time, " A=%b, B=%b, C=%b, sum=%b, carry=%b", A, B, C, sum, carry);
end

initial begin
    $dumpfile("fulladder.vcd");
    $dumpvars(0, fulladder_tb);
end

endmodule
```

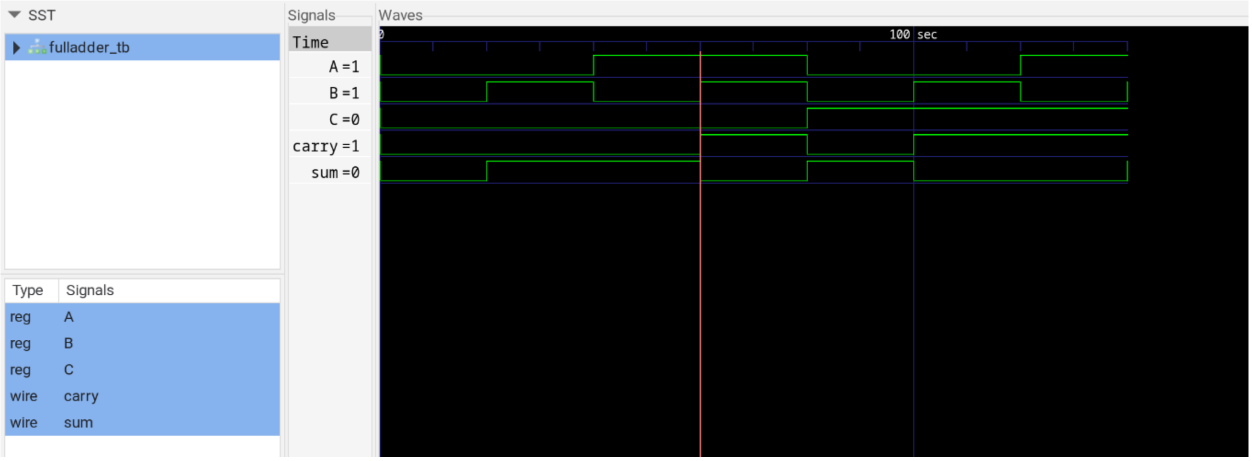


```
rithvikmatta@penguin:~/PES/DDCO-sem-3/week 3$ iverilog -o fa fullAdder.v fullAdder_tb.v
rithvikmatta@penguin:~/PES/DDCO-sem-3/week 3$ vvp fa
VCD info: dumpfile fulladder.vcd opened for output.
                   0 A=0, B=0, C=0, sum=0, carry=0
                  20 A=0, B=1, C=0, sum=1, carry=0
                  40 A=1, B=0, C=0, sum=1, carry=0
                  60 A=1, B=1, C=0, sum=0, carry=1
                  80 A=0, B=0, C=1, sum=1, carry=0
                 100 A=0, B=1, C=1, sum=0, carry=1
                 120 A=1, B=0, C=1, sum=0, carry=1
                 140 A=1, B=1, C=1, sum=1, carry=1
rithvikmatta@penguin:~/PES/DDCO-sem-3/week 3$ gtkwave fulladder.vcd

GTKWave Analyzer v3.3.118 (w)1999-2023 BSI

[0] start time.
[140] end time.
GTKWAVE | Touch screen detected, enabling gestures.
Exiting.
```


SST
fulladder_tb
Type Signals
reg A
reg B
reg C
wire carry
wire sum
Signals
Waves
Time
A =0
B =0
C =0
carry =0
sum =0
0
100 sec

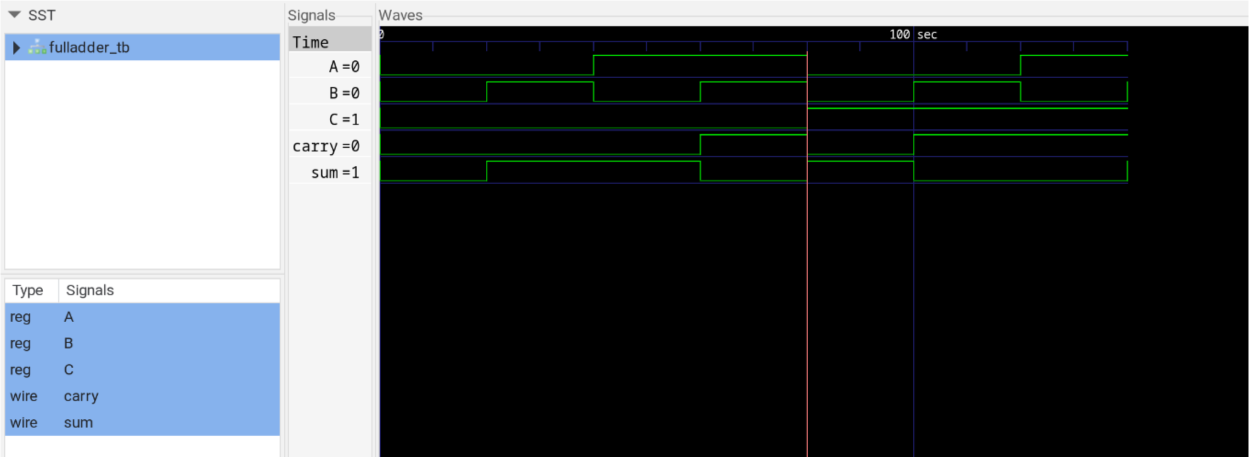

SST
fulladder_tb
Type Signals
reg A
reg B
reg C
wire carry
wire sum
Signals
Waves
Time
A =0
B =1
C =0
carry =0
sum =1
0
100 sec


SST
fulladder_tb
Type Signals
reg A
reg B
reg C
wire carry
wire sum
Signals
Waves
Time
A =1
B =0
C =0
carry =0
sum =1
0
100 sec



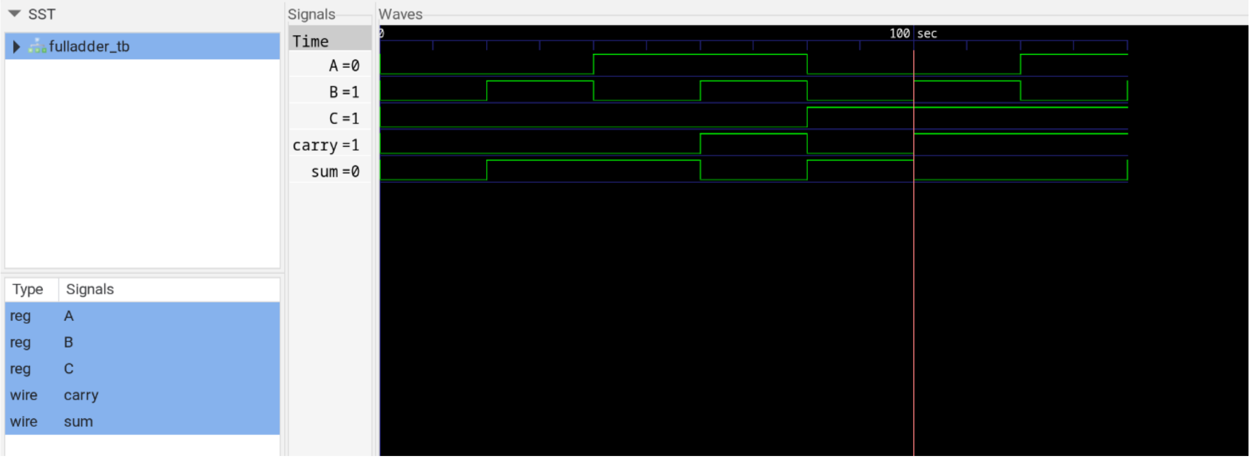
Time
A =1
B =1
C =0
carry =1
sum =0

Type	Signals
reg	A
reg	B
reg	C
wire	carry
wire	sum



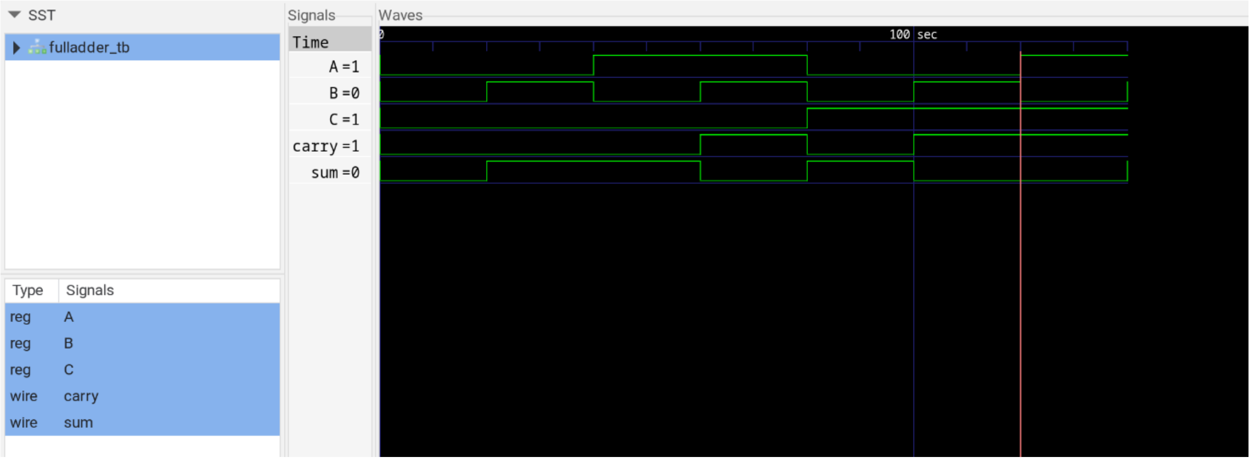
Time
A =0
B =0
C =1
carry =0
sum =1

Type	Signals
reg	A
reg	B
reg	C
wire	carry
wire	sum



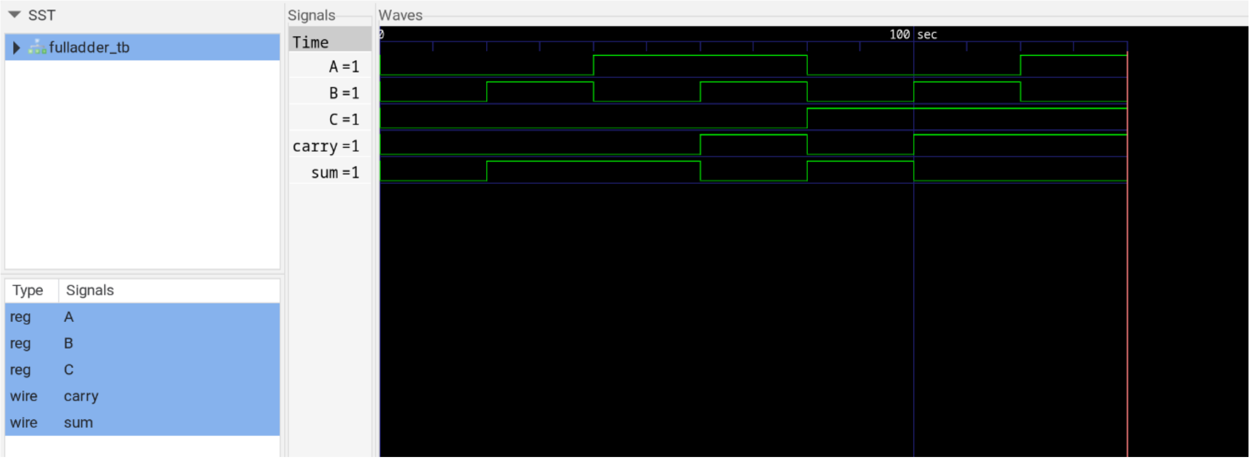
Time
A =0
B =1
C =1
carry =1
sum =0

Type	Signals
reg	A
reg	B
reg	C
wire	carry
wire	sum



Type	Signals
reg	A
reg	B
reg	C
wire	carry
wire	sum

```
Time
A =1
B =0
C =1
carry =1
sum =0
```

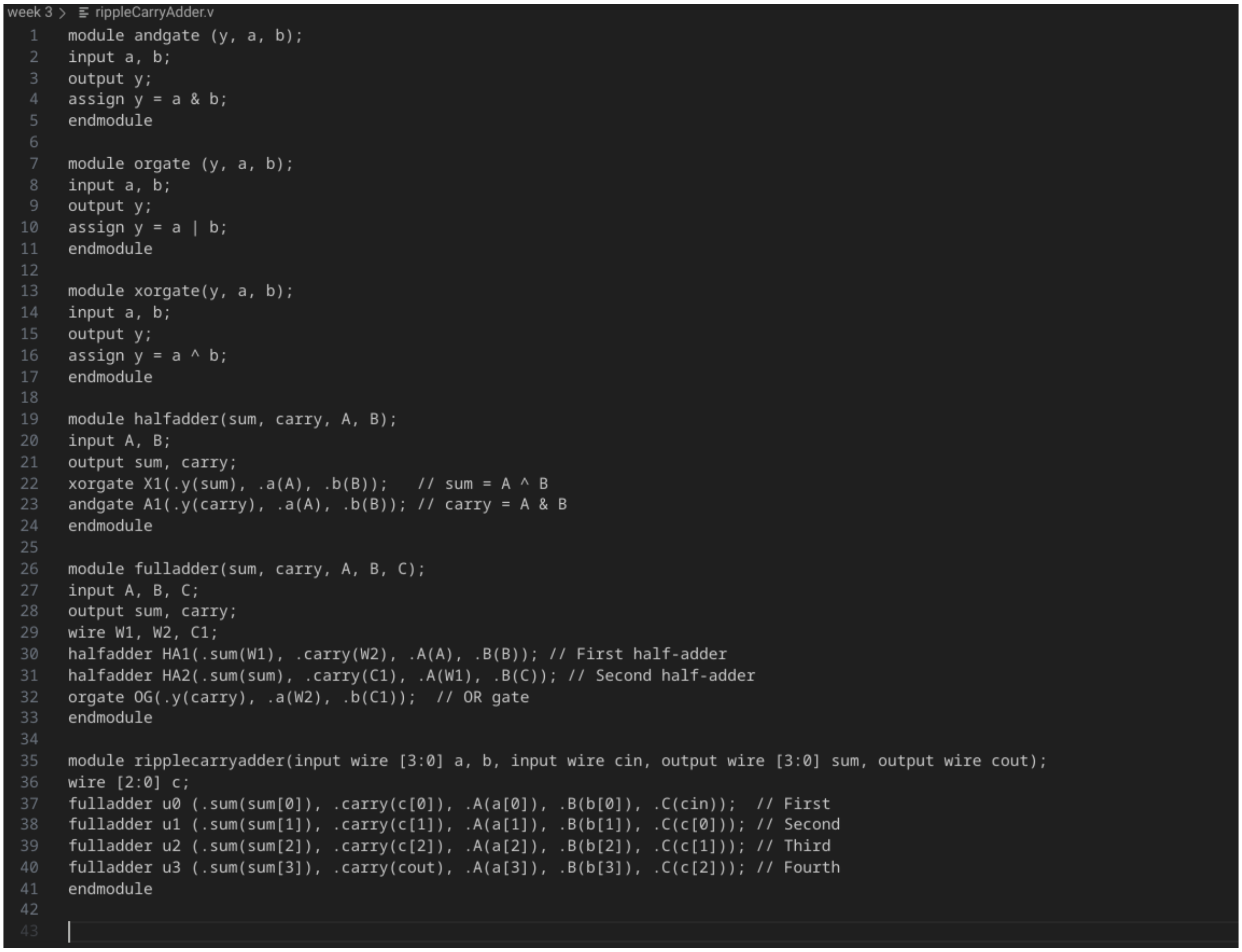


Type	Signals
reg	A
reg	B
reg	C
wire	carry
wire	sum

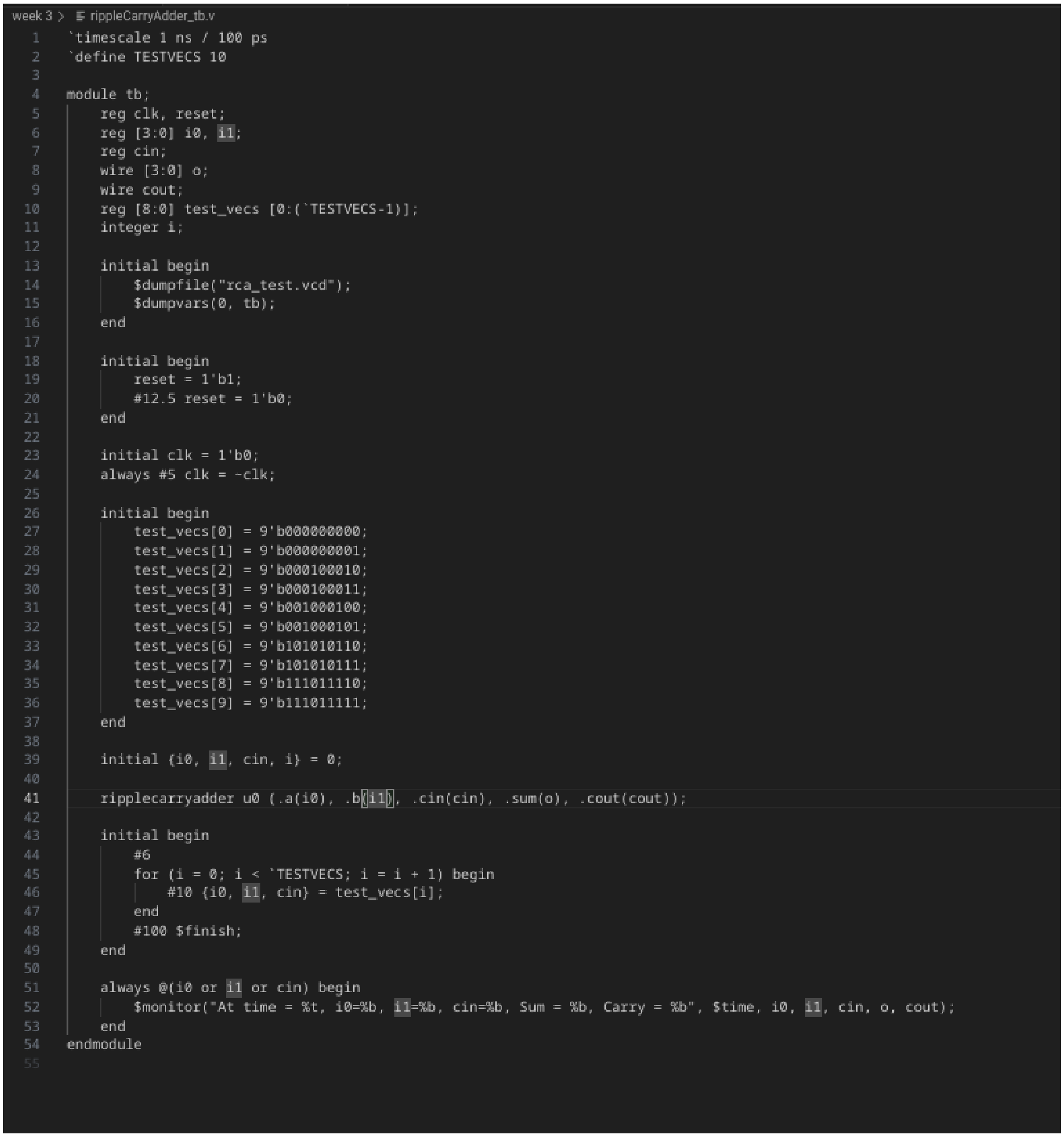
```
Time
A =1
B =1
C =1
carry =1
sum =1
```

3)Ripple carry adder

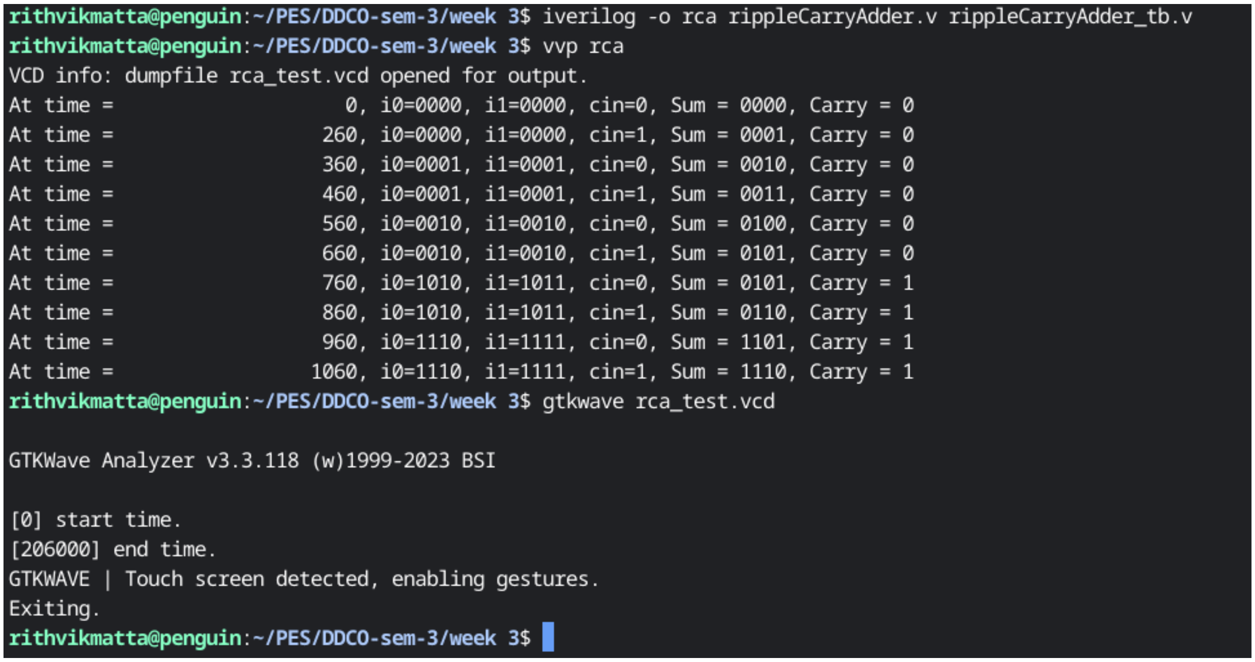
	a3	a2	a1	a0	b3	b2	b1	b0	cin		
	I0[3]	I0[2]	I0[1]	I0[0]	I1[3]	I1[2]	I1[1]	I1[0]	cin	Sum[3:0]	cout
TESTVECTOR[0]	0	0	0	0	0	0	0	0	0	0+0+0=0000	0
TESTVECTOR[1]	0	0	0	0	0	0	0	0	1	0+0+1=0001	0
TESTVECTOR[2]	0	0	0	1	0	0	0	1	0	1+1+0=0010	0
TESTVECTOR[3]	0	0	0	1	0	0	0	1	1	1+1+1=0011	0
TESTVECTOR[4]	0	0	1	0	0	0	1	1	0	2+3+0=0010	1
TESTVECTOR[5]	0	0	1	0	0	0	1	1	1	2+3+1=0101	0
TESTVECTOR[6]	1	0	1	0	1	0	1	1	0	A+B+0=1010	0
TESTVECTOR[7]	1	0	1	0	1	0	1	1	1	A+B+1=1011	1
TESTVECTOR[8]	1	1	1	0	1	1	1	1	0	E+F+0=1100	0
TESTVECTOR[9]	1	1	1	0	1	1	1	1	1	E+F+1=1111	1



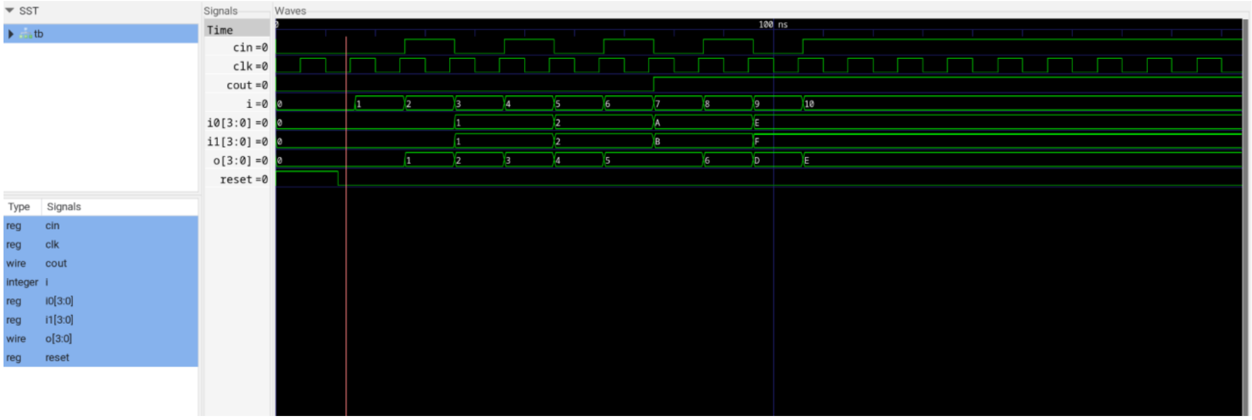
```verilog
module andgate (y, a, b);
input a, b;
output y;
assign y = a & b;
endmodule

module orgate (y, a, b);
input a, b;
output y;
assign y = a | b;
endmodule

module xorgate(y, a, b);
input a, b;
output y;
assign y = a ^ b;
endmodule

module halfadder(sum, carry, A, B);
input A, B;
output sum, carry;
xorgate X1(.y(sum), .a(A), .b(B));   // sum = A ^ B
andgate A1(.y(carry), .a(A), .b(B)); // carry = A & B
endmodule

module fulladder(sum, carry, A, B, C);
input A, B, C;
output sum, carry;
wire W1, W2, C1;
halfadder HA1(.sum(W1), .carry(W2), .A(A), .B(B)); // First half-adder
halfadder HA2(.sum(sum), .carry(C1), .A(W1), .B(C)); // Second half-adder
orgate OG(.y(carry), .a(W2), .b(C1));  // OR gate
endmodule

module ripplecarryadder(input wire [3:0] a, b, input wire cin, output wire [3:0] sum, output wire cout);
wire [2:0] c;
fulladder u0 (.sum(sum[0]), .carry(c[0]), .A(a[0]), .B(b[0]), .C(cin));  // First
fulladder u1 (.sum(sum[1]), .carry(c[1]), .A(a[1]), .B(b[1]), .C(c[0])); // Second
fulladder u2 (.sum(sum[2]), .carry(c[2]), .A(a[2]), .B(b[2]), .C(c[1])); // Third
fulladder u3 (.sum(sum[3]), .carry(cout), .A(a[3]), .B(b[3]), .C(c[2])); // Fourth
endmodule

```

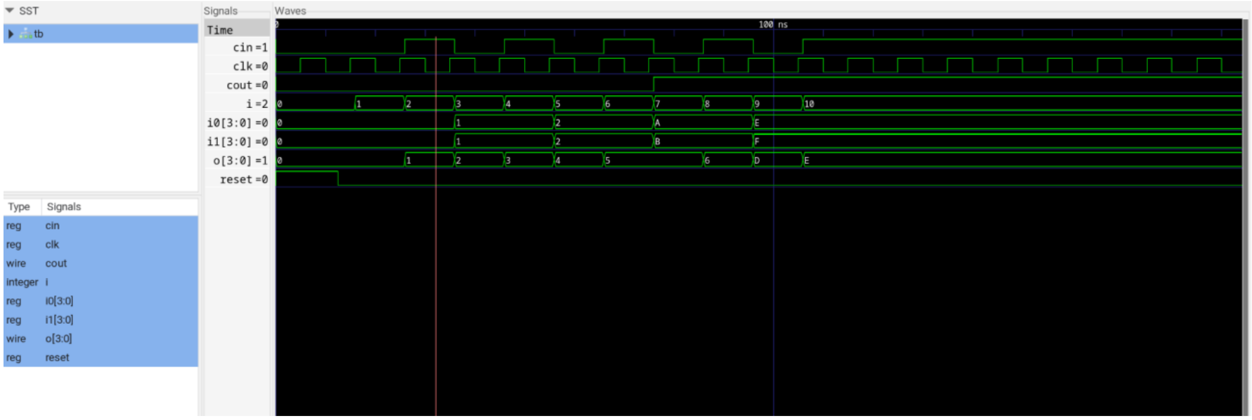


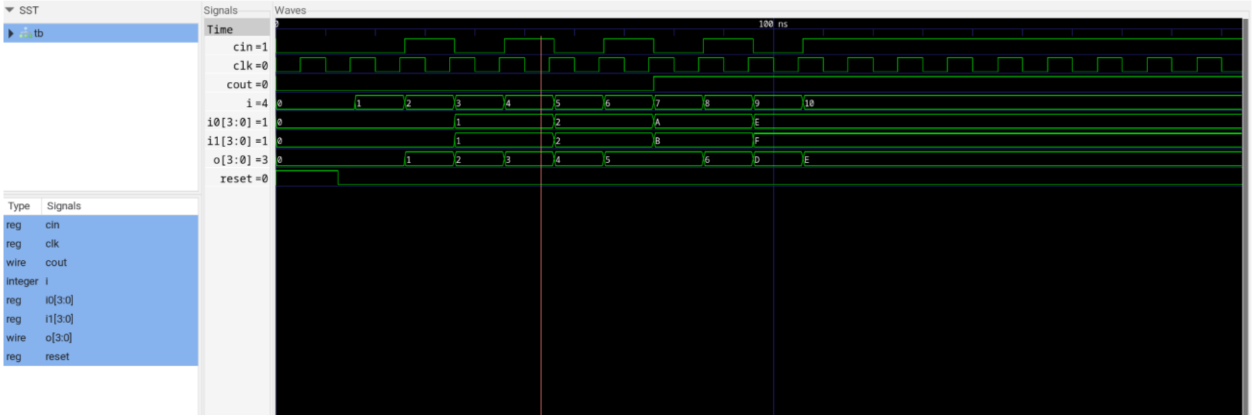
```
`timescale 1 ns / 100 ps
`define TESTVECS 10

module tb;
    reg clk, reset;
    reg [3:0] i0, i1;
    reg cin;
    wire [3:0] o;
    wire cout;
    reg [8:0] test_vecs [0:(`TESTVECS-1)];
    integer i;

    initial begin
        $dumpfile("rca_test.vcd");
        $dumpvars(0, tb);
    end

    initial begin
        reset = 1'b1;
        #12.5 reset = 1'b0;
    end

    initial clk = 1'b0;
    always #5 clk = ~clk;

    initial begin
        test_vecs[0] = 9'b000000000;
        test_vecs[1] = 9'b000000001;
        test_vecs[2] = 9'b000100010;
        test_vecs[3] = 9'b000100011;
        test_vecs[4] = 9'b001000100;
        test_vecs[5] = 9'b001000101;
        test_vecs[6] = 9'b101010110;
        test_vecs[7] = 9'b101010111;
        test_vecs[8] = 9'b111011110;
        test_vecs[9] = 9'b111011111;
    end

    initial {i0, i1, cin, i} = 0;

    ripplecarryadder u0 (.a(i0), .b(i1), .cin(cin), .sum(o), .cout(cout));

    initial begin
        #6
        for (i = 0; i < `TESTVECS; i = i + 1) begin
            #10 {i0, i1, cin} = test_vecs[i];
        end
        #100 $finish;
    end

    always @(i0 or i1 or cin) begin
        $monitor("At time = %t, i0=%b, i1=%b, cin=%b, Sum = %b, Carry = %b", $time, i0, i1, cin, o, cout);
    end
endmodule
```



```
rithvikmatta@penguin:~/PES/DDCO-sem-3/week 3$ iverilog -o rca rippleCarryAdder.v rippleCarryAdder_tb.v
rithvikmatta@penguin:~/PES/DDCO-sem-3/week 3$ vvp rca
VCD info: dumpfile rca_test.vcd opened for output.
At time =                    0, i0=0000, i1=0000, cin=0, Sum = 0000, Carry = 0
At time =                  260, i0=0000, i1=0000, cin=1, Sum = 0001, Carry = 0
At time =                  360, i0=0001, i1=0001, cin=0, Sum = 0010, Carry = 0
At time =                  460, i0=0001, i1=0001, cin=1, Sum = 0011, Carry = 0
At time =                  560, i0=0010, i1=0010, cin=0, Sum = 0100, Carry = 0
At time =                  660, i0=0010, i1=0010, cin=1, Sum = 0101, Carry = 0
At time =                  760, i0=1010, i1=1011, cin=0, Sum = 0101, Carry = 1
At time =                  860, i0=1010, i1=1011, cin=1, Sum = 0110, Carry = 1
At time =                  960, i0=1110, i1=1111, cin=0, Sum = 1101, Carry = 1
At time =                 1060, i0=1110, i1=1111, cin=1, Sum = 1110, Carry = 1
rithvikmatta@penguin:~/PES/DDCO-sem-3/week 3$ gtkwave rca_test.vcd

GTKWave Analyzer v3.3.118 (w)1999-2023 BSI

[0] start time.
[206000] end time.
GTKWAVE | Touch screen detected, enabling gestures.
Exiting.
rithvikmatta@penguin:~/PES/DDCO-sem-3/week 3$
```

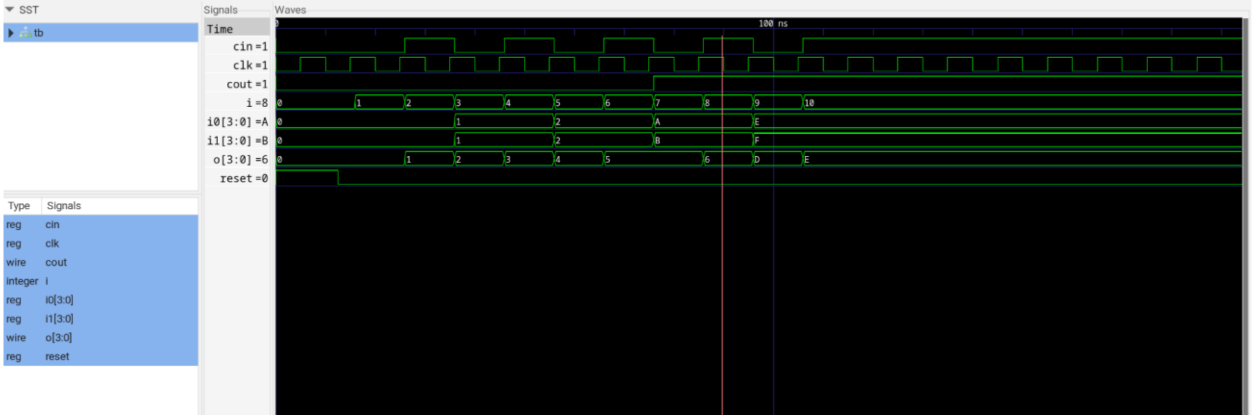






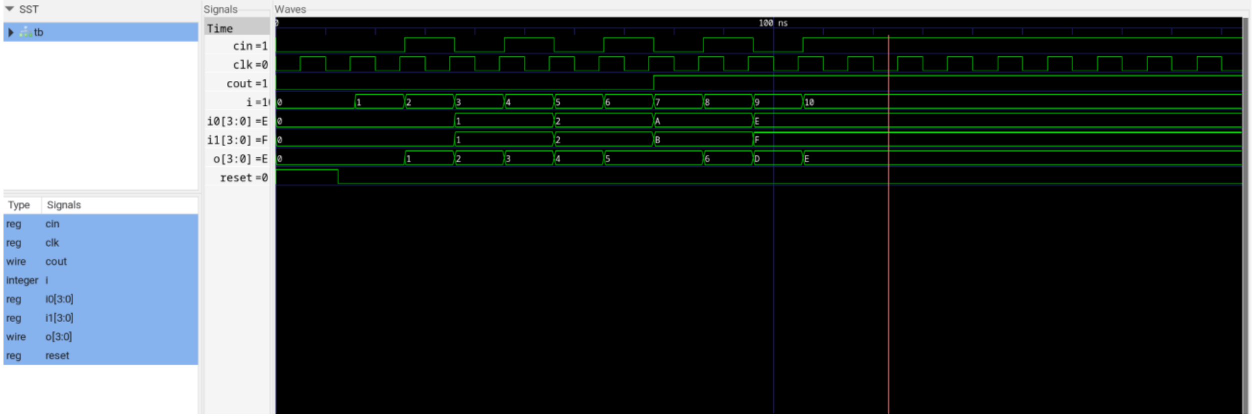
Type	Signals
reg	cin
reg	clk
wire	cout
integer	i
reg	i0[3:0]
reg	i1[3:0]
wire	o[3:0]
reg	reset

```
Time
cin =1
clk =0
cout =0
i =4
i0[3:0] =1
i1[3:0] =1
o[3:0] =3
reset =0
```



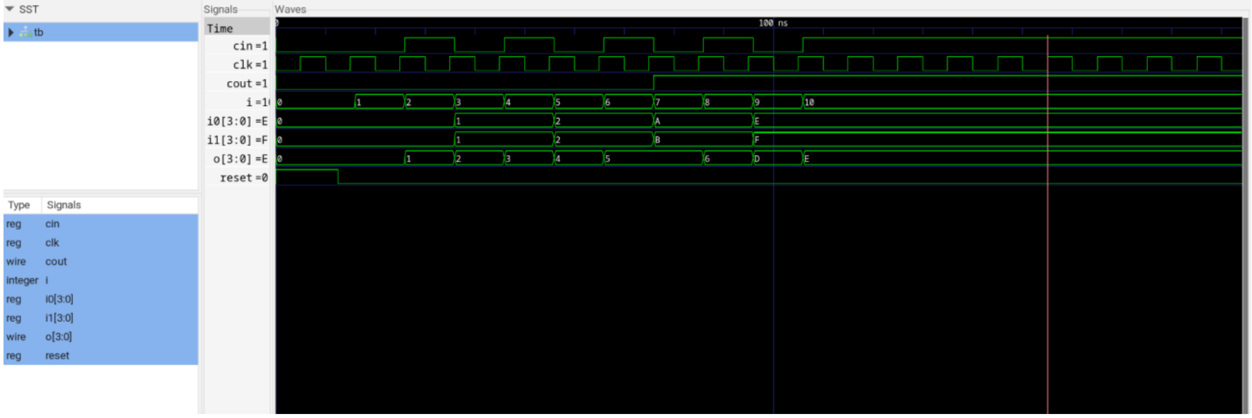
Type	Signals
reg	cin
reg	clk
wire	cout
integer	i
reg	i0[3:0]
reg	i1[3:0]
wire	o[3:0]
reg	reset

```
Time
cin =1
clk =1
cout =1
i =8
i0[3:0] =A
i1[3:0] =B
o[3:0] =6
reset =0
```



Type	Signals
reg	cin
reg	clk
wire	cout
integer	i
reg	i0[3:0]
reg	i1[3:0]
wire	o[3:0]
reg	reset

```
Time
cin =1
clk =0
cout =1
i =1(
i0[3:0] =E
i1[3:0] =F
o[3:0] =E
reset =0
```



Type	Signals
reg	cin
reg	clk
wire	cout
integer	i
reg	i0[3:0]
reg	i1[3:0]
wire	o[3:0]
reg	reset

```
Time
cin =1
clk =1
cout =1
i =1(
i0[3:0] =E
i1[3:0] =F
o[3:0] =E
reset =0
```

If found plagiarized, I will abide with the disciplinary action of the University.

Signature: Rithvik

Name: Rithvik Rajesh Matta

SRN: PES2UG23CS485

Section: H

Date: 23/08/2024