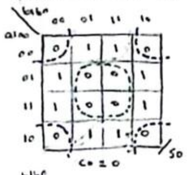
1.

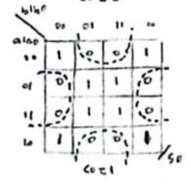
Truth table:

a1	a0	b1	b0	c0	c1	s1	s0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	1
0	0	1	0	0	0	1	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	1	0	1	0	0
0	1	1	0	0	0	1	1
1	1	0	0	0	0	1	1
1	1	0	1	0	1	0	0
1	1	1	1	0	1	1	0
1	1	1	0	0	1	0	1
1	0	0	0	0	0	1	0
1	0	0	1	0	0	1	1
1	0	1	1	0	1	0	1
1	0	1	0	0	1	0	0

a0	a1	b1	b0	c0	c1	s1	s0
0	0	0	0	1	0	0	1
0	0	0	1	1	0	1	0
0	0	1	1	1	1	0	0
0	0	1	0	1	0	1	1
0	1	0	0	1	0	1	0
0	1	0	1	1	0	1	1
0	1	1	1	1	1	0	1
0	1	1	0	1	1	0	0
1	1	0	0	1	1	0	0
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	1	1	1	1	0
1	0	1	0	1	1	0	1

• since we want to build a circuit using NOR gates, we solve the k-maps using maxterms

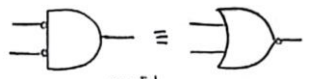




$\rightarrow \overline{s0} = \overline{a0}\,\overline{b0}\,\overline{c0} + a0\,b0\,\overline{c0} + a0\,\overline{b0}\,c0 + \overline{a0}\,b0\,c0$

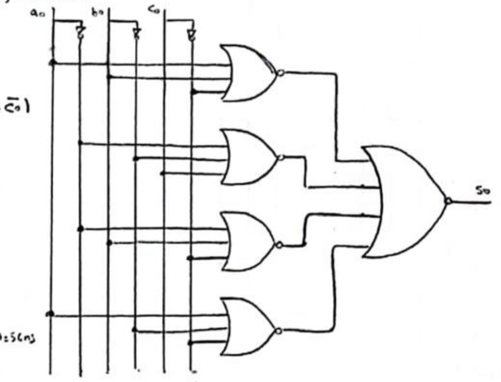
$\rightarrow s0 = (a0 + b0 + c0).(\overline{a0} + \overline{b0} + c0).(\overline{a0} + b0 + \overline{c0}).(a0 + \overline{b0} + \overline{c0})$

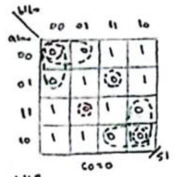
• we know:

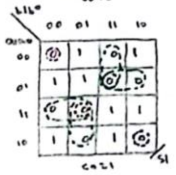


delays: T=1: { 4-input nor: 5×4 = 20 ns ; 3-input nor: 7×3 = 21 ns } → 41 ns
(pass T=1)
(pass 1→2)

T=0: { 4-input nor: 4×7 = 28 ns ; 3-input nor (1→1): 5×3 = 15 ns ; 3-input nor (4→0): 7×3 = 28 ns } 28>15 → 28+28 = 56 ns

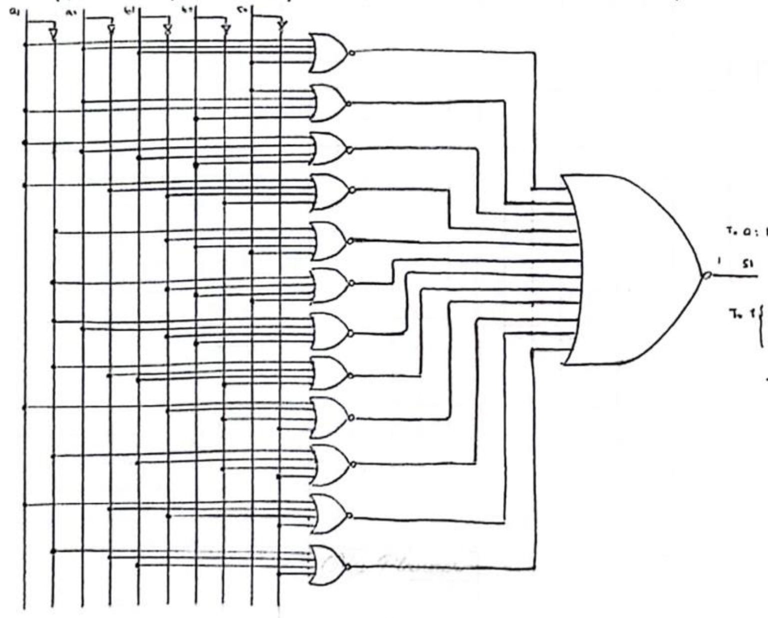






$\rightarrow \overline{s1} = \overline{c0}\,\overline{a1}\,\overline{a0}\,\overline{b1} + \overline{c0}\,\overline{a1}\,\overline{a0}\,\overline{b0} + \overline{a1}\,\overline{a0}\,\overline{b1}\,\overline{b0} + \overline{a1}\,a0\,b1\,b0 + \overline{c0}\,b1\,\overline{b0}\,a1 + \overline{c0}\,a1\,b1\,\overline{b0} + a1\,\overline{a0}\,b1\,\overline{b0} + a1\,a0\,\overline{b1}\,b0 + c0\,\overline{a1}\,b1\,b0 + c0\,\overline{a1}\,a0\,b1 + c0\,a1\,\overline{b1}\,b0 + c0\,a1\,a0\,\overline{b1}$

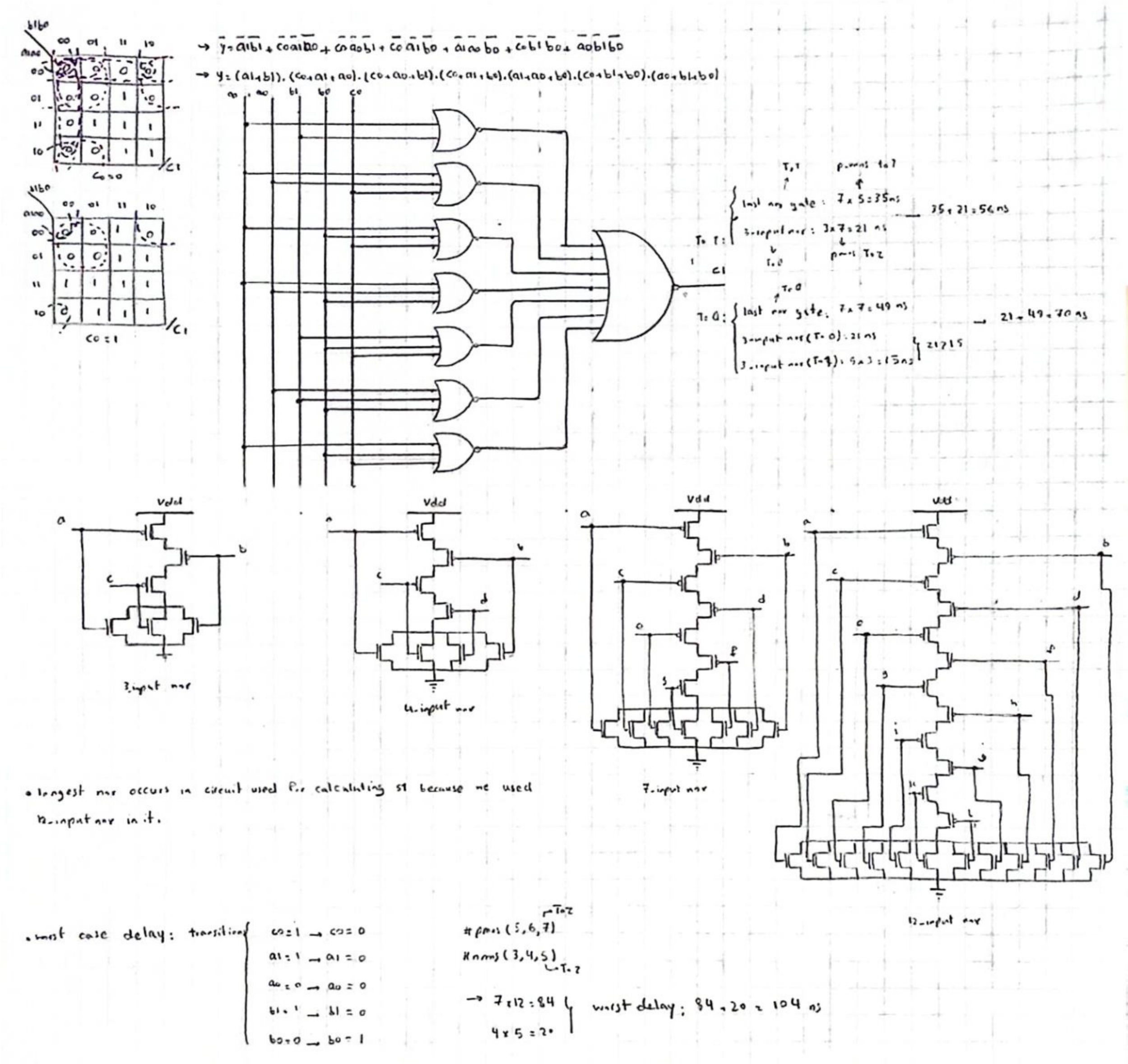
$\rightarrow s1 = (c0 + a1 + a0 + b1).(c0 + a1 + a0 + b0).(a1 + a0 + b1 + b0).(a1 + \overline{a0} + \overline{b1} + \overline{b0}).(c0 + \overline{b1} + b0 + \overline{a1}).(c0 + \overline{a1} + \overline{b1} + b0).(\overline{a1} + a0 + \overline{b1} + b0).$
$(\overline{a1} + \overline{a0} + b1 + \overline{b0}).(\overline{c0} + a1 + \overline{b1} + \overline{b0}).(\overline{c0} + a1 + \overline{b1} + \overline{a0}).(\overline{c0} + \overline{a1} + b1 + \overline{b0}).(\overline{c0} + \overline{a1} + \overline{a0} + b1)$



T=0: 104 ns

T=1 { 12-input nor: 12×5 = 60 ns ; 4-input nor: 4×7 = 28 ns }
(pass T=1)

→ 28 + 60 = 88 ns



→ $\bar{y} = \bar{a_1}\bar{b_1} + \bar{c_0}\bar{a_1}\bar{b_0} + \bar{c_0}\bar{a_0}\bar{b_1} + \bar{c_0}\bar{a_1}\bar{b_0} + \bar{a_1}\bar{a_0}\bar{b_0} + \bar{c_0}\bar{b_1}\bar{b_0} + \bar{a_0}\bar{b_1}\bar{b_0}$

→ $y = (a_1+b_1).(c_0+a_1+a_0).(c_0+a_0+b_1).(c_0+a_1+b_0).(a_1+a_0+b_0).(c_0+b_1+b_0).(a_0+b_1+b_0)$

T=1: last nor gate : 7 x 5 = 35 ns (T=1, pmos 1 to 7); 3-input nor : 3 x 7 = 21 ns (T=0, pmos T=2) → 35 + 21 = 56 ns

T=0: last nor gate : 7 x 7 = 49 ns (T=0); 3-input nor (T=0) : 21 ns; 3-input nor (T=1) : 5 x 3 = 15 ns } 21 > 15 → 21 + 49 = 70 ns

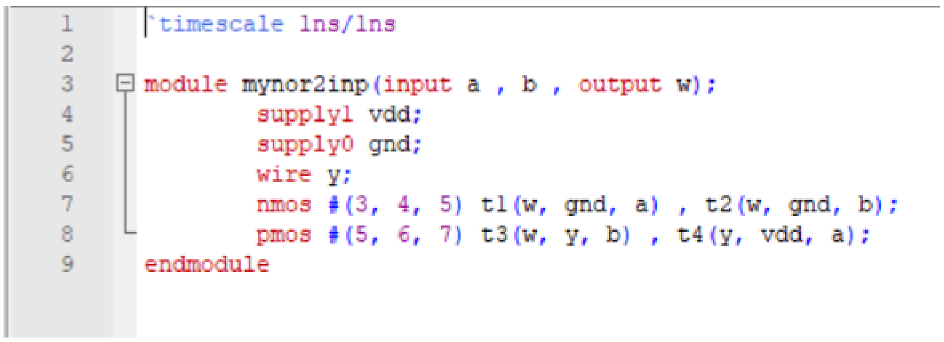
• longest nor occurs in circuit used for calculating s1 because we used 12-input nor in it.

• worst case delay: transition { c0=1 → c0=0; a1=1 → a1=0; a0=0 → a0=0; b1=1 → b1=0; b0=0 → b0=1 }

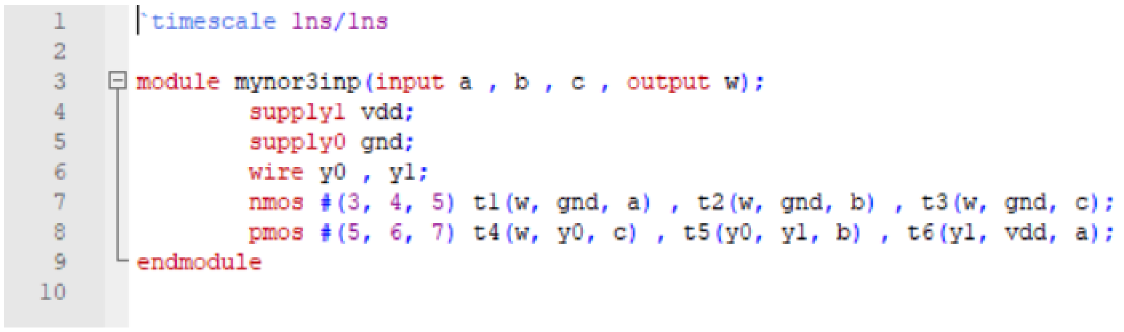
pmos (5,6,7) ← T=2

nmos (3,4,5) ← T=2

→ 7 x 12 = 84 ; 4 x 5 = 20 } worst delay: 84 + 20 = 104 ns

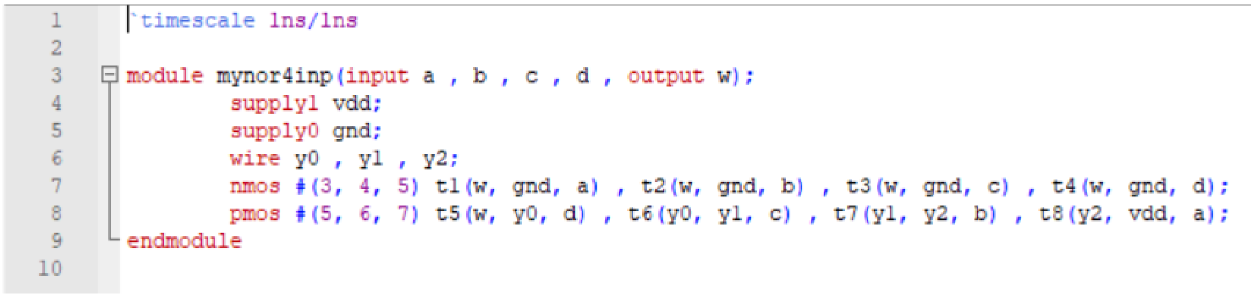

```
`timescale 1ns/1ns

module mynor2inp(input a , b , output w);
        supply1 vdd;
        supply0 gnd;
        wire y;
        nmos #(3, 4, 5) t1(w, gnd, a) , t2(w, gnd, b);
        pmos #(5, 6, 7) t3(w, y, b) , t4(y, vdd, a);
endmodule
```

2-input nor



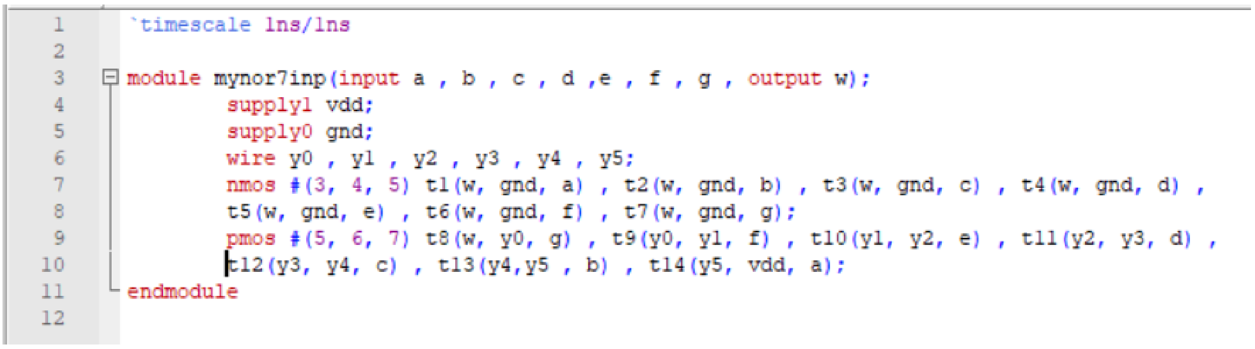
```verilog
`timescale 1ns/1ns

module mynor3inp(input a , b , c , output w);
        supply1 vdd;
        supply0 gnd;
        wire y0 , y1;
        nmos #(3, 4, 5) t1(w, gnd, a) , t2(w, gnd, b) , t3(w, gnd, c);
        pmos #(5, 6, 7) t4(w, y0, c) , t5(y0, y1, b) , t6(y1, vdd, a);
endmodule
```

3-input nor



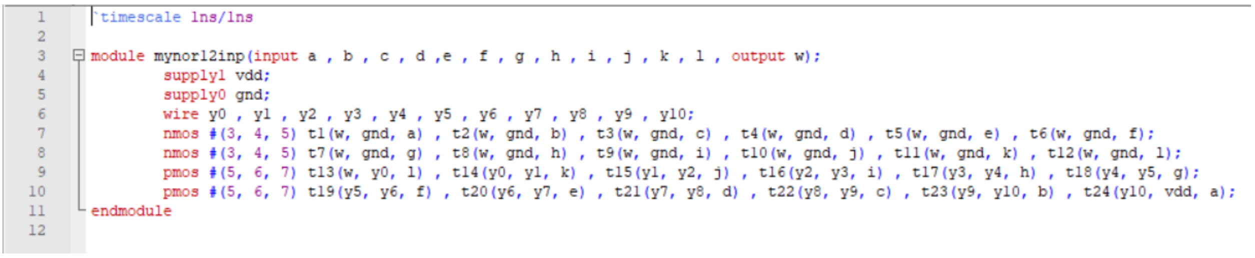
```verilog
`timescale 1ns/1ns

module mynor4inp(input a , b , c , d , output w);
        supply1 vdd;
        supply0 gnd;
        wire y0 , y1 , y2;
        nmos #(3, 4, 5) t1(w, gnd, a) , t2(w, gnd, b) , t3(w, gnd, c) , t4(w, gnd, d);
        pmos #(5, 6, 7) t5(w, y0, d) , t6(y0, y1, c) , t7(y1, y2, b) , t8(y2, vdd, a);
endmodule
```

4-input nor



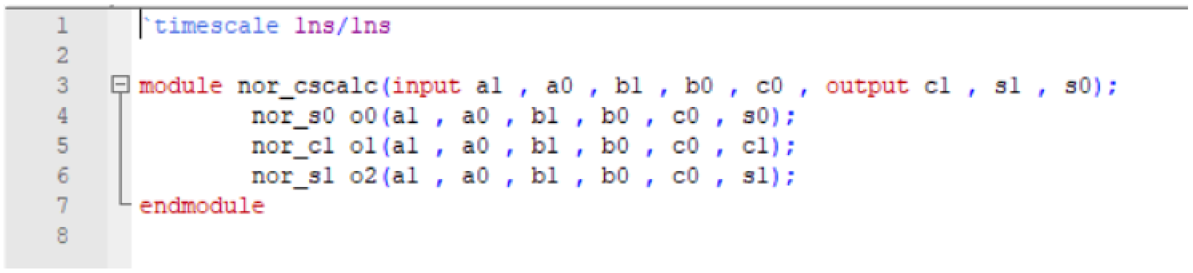
```verilog
`timescale 1ns/1ns

module mynor7inp(input a , b , c , d ,e , f , g , output w);
        supply1 vdd;
        supply0 gnd;
        wire y0 , y1 , y2 , y3 , y4 , y5;
        nmos #(3, 4, 5) t1(w, gnd, a) , t2(w, gnd, b) , t3(w, gnd, c) , t4(w, gnd, d) ,
        t5(w, gnd, e) , t6(w, gnd, f) , t7(w, gnd, g);
        pmos #(5, 6, 7) t8(w, y0, g) , t9(y0, y1, f) , t10(y1, y2, e) , t11(y2, y3, d) ,
        t12(y3, y4, c) , t13(y4,y5 , b) , t14(y5, vdd, a);
endmodule
```

7-input nor

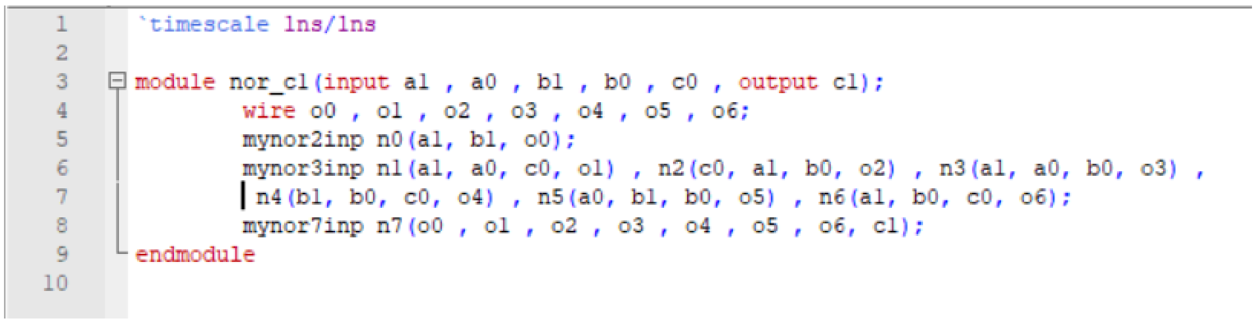


```verilog
`timescale 1ns/1ns

module mynor12inp(input a , b , c , d ,e , f , g , h , i , j , k , l , output w);
        supply1 vdd;
        supply0 gnd;
        wire y0 , y1 , y2 , y3 , y4 , y5 , y6 , y7 , y8 , y9 , y10;
        nmos #(3, 4, 5) t1(w, gnd, a) , t2(w, gnd, b) , t3(w, gnd, c) , t4(w, gnd, d) , t5(w, gnd, e) , t6(w, gnd, f);
        nmos #(3, 4, 5) t7(w, gnd, g) , t8(w, gnd, h) , t9(w, gnd, i) , t10(w, gnd, j) , t11(w, gnd, k) , t12(w, gnd, l);
        pmos #(5, 6, 7) t13(w, y0, l) , t14(y0, y1, k) , t15(y1, y2, j) , t16(y2, y3, i) , t17(y3, y4, h) , t18(y4, y5, g);
        pmos #(5, 6, 7) t19(y5, y6, f) , t20(y6, y7, e) , t21(y7, y8, d) , t22(y8, y9, c) , t23(y9, y10, b) , t24(y10, vdd, a);
endmodule
```

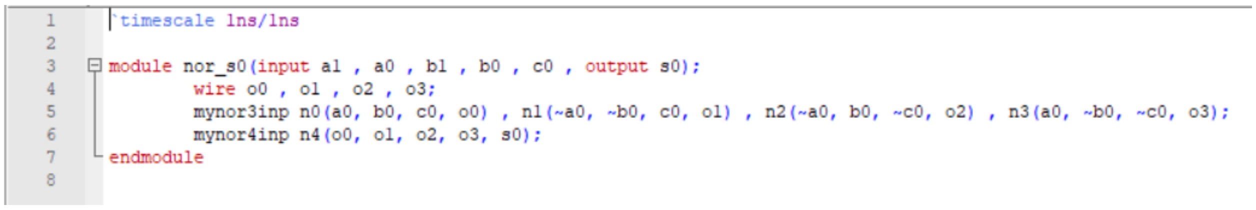
12-input nor



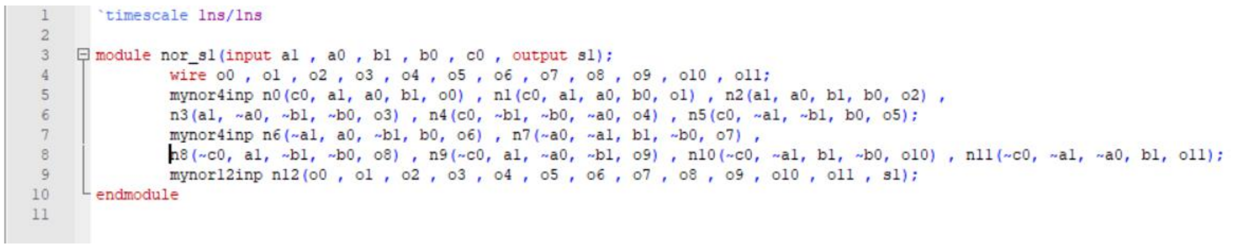
```verilog
`timescale 1ns/1ns

module nor_cscalc(input a1 , a0 , b1 , b0 , c0 , output c1 , s1 , s0);
        nor_s0 o0(a1 , a0 , b1 , b0 , c0 , s0);
        nor_c1 o1(a1 , a0 , b1 , b0 , c0 , c1);
        nor_s1 o2(a1 , a0 , b1 , b0 , c0 , s1);
endmodule
```



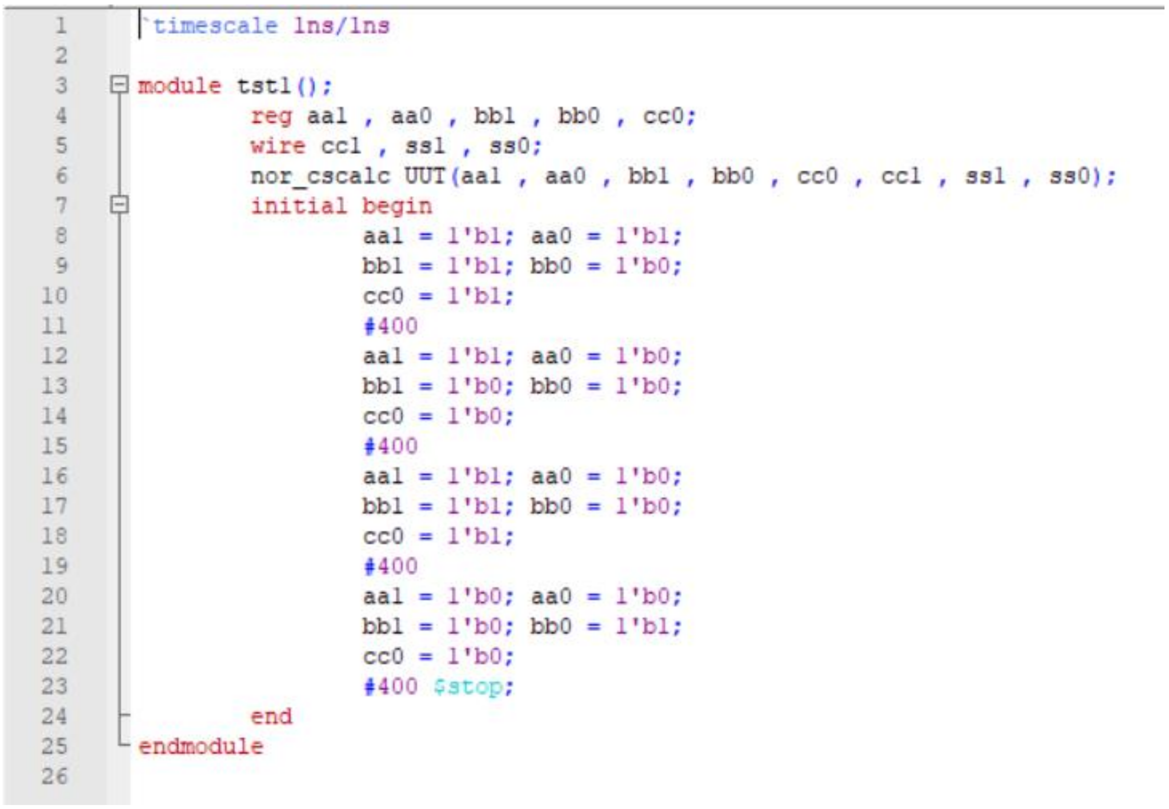
```verilog
`timescale 1ns/1ns

module nor_c1(input a1 , a0 , b1 , b0 , c0 , output c1);
        wire o0 , o1 , o2 , o3 , o4 , o5 , o6;
        mynor2inp n0(a1, b1, o0);
        mynor3inp n1(a1, a0, c0, o1) , n2(c0, a1, b0, o2) , n3(a1, a0, b0, o3) ,
        n4(b1, b0, c0, o4) , n5(a0, b1, b0, o5) , n6(a1, b0, c0, o6);
        mynor7inp n7(o0 , o1 , o2 , o3 , o4 , o5 , o6, c1);
endmodule
```



```verilog
`timescale 1ns/1ns

module nor_s0(input a1 , a0 , b1 , b0 , c0 , output s0);
        wire o0 , o1 , o2 , o3;
        mynor3inp n0(a0, b0, c0, o0) , n1(~a0, ~b0, c0, o1) , n2(~a0, b0, ~c0, o2) , n3(a0, ~b0, ~c0, o3);
        mynor4inp n4(o0, o1, o2, o3, s0);
endmodule
```

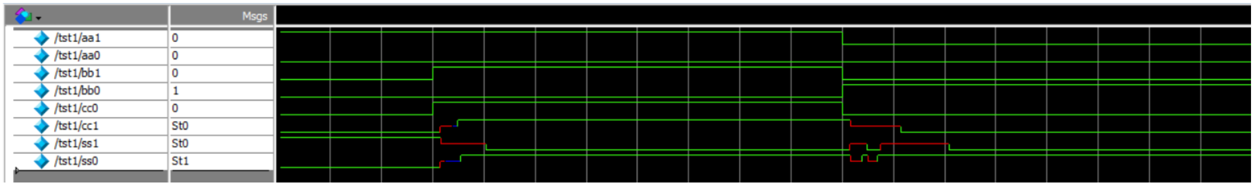


```verilog
`timescale 1ns/1ns

module nor_s1(input a1 , a0 , b1 , b0 , c0 , output s1);
        wire o0 , o1 , o2 , o3 , o4 , o5 , o6 , o7 , o8 , o9 , o10 , o11;
        mynor4inp n0(c0, a1, a0, b1, o0) , n1(c0, a1, a0, b0, o1) , n2(a1, a0, b1, b0, o2) ,
        n3(a1, ~a0, ~b1, ~b0, o3) , n4(c0, ~b1, ~b0, ~a0, o4) , n5(c0, ~a1, ~b1, b0, o5);
        mynor4inp n6(~a1, a0, ~b1, b0, o6) , n7(~a0, ~a1, b1, ~b0, o7) ,
        n8(~c0, a1, ~b1, ~b0, o8) , n9(~c0, a1, ~a0, ~b1, o9) , n10(~c0, ~a1, b1, ~b0, o10) , n11(~c0, ~a1, ~a0, b1, o11);
        mynor12inp n12(o0 , o1 , o2 , o3 , o4 , o5 , o6 , o7 , o8 , o9 , o10 , o11 , s1);
endmodule
```



```
`timescale 1ns/1ns

module tst1();
        reg aa1 , aa0 , bb1 , bb0 , cc0;
        wire cc1 , ss1 , ss0;
        nor_cscalc UUT(aa1 , aa0 , bb1 , bb0 , cc0 , cc1 , ss1 , ss0);
        initial begin
                aa1 = 1'b1; aa0 = 1'b1;
                bb1 = 1'b1; bb0 = 1'b0;
                cc0 = 1'b1;
                #400
                aa1 = 1'b1; aa0 = 1'b0;
                bb1 = 1'b0; bb0 = 1'b0;
                cc0 = 1'b0;
                #400
                aa1 = 1'b1; aa0 = 1'b0;
                bb1 = 1'b1; bb0 = 1'b0;
                cc0 = 1'b1;
                #400
                aa1 = 1'b0; aa0 = 1'b0;
                bb1 = 1'b0; bb0 = 1'b1;
                cc0 = 1'b0;
                #400 $stop;
        end
endmodule
```

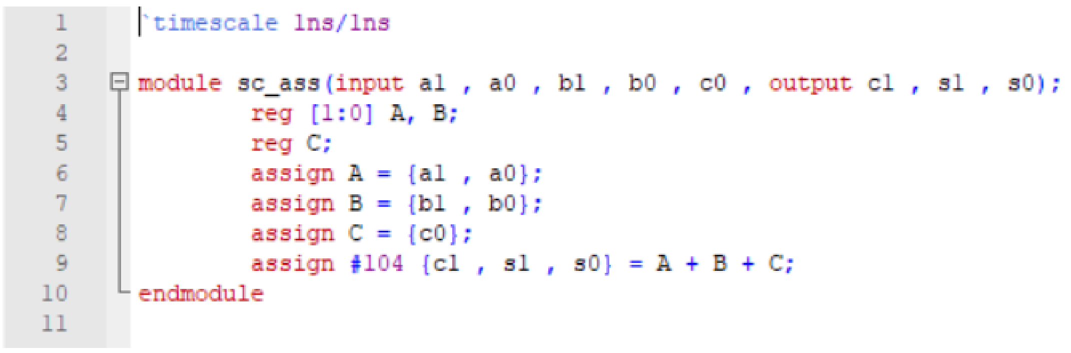
Test bench



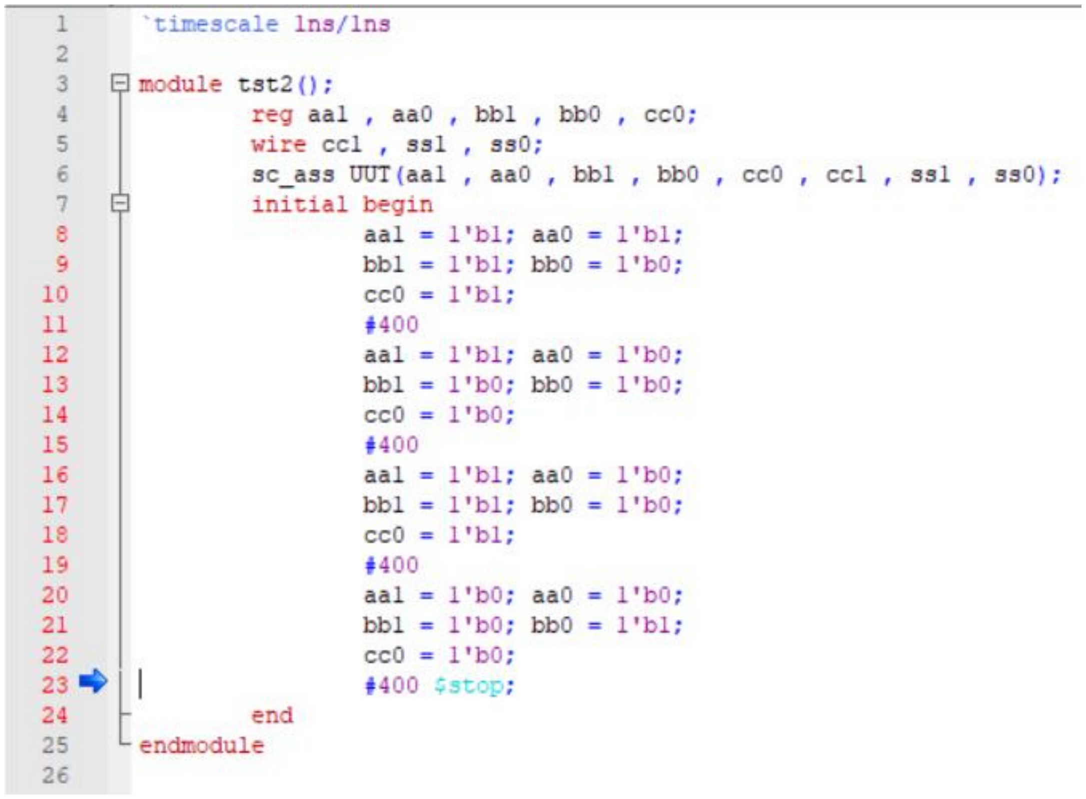
```
                Msgs
/tst1/aa1       0
/tst1/aa0       0
/tst1/bb1       0
/tst1/bb0       1
/tst1/cc0       0
/tst1/cc1       St0
/tst1/ss1       St0
/tst1/ss0       St1
```

2.

2. assign statement : worst case delay ; 104 ns

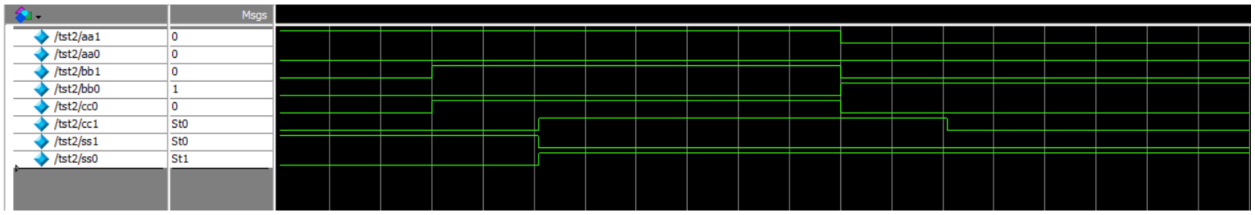


```
`timescale 1ns/1ns

module sc_ass(input a1 , a0 , b1 , b0 , c0 , output c1 , s1 , s0);
        reg [1:0] A, B;
        reg C;
        assign A = {a1 , a0};
        assign B = {b1 , b0};
        assign C = {c0};
        assign #104 {c1 , s1 , s0} = A + B + C;
endmodule
```



```verilog
`timescale 1ns/1ns

module tst2();
        reg aa1 , aa0 , bb1 , bb0 , cc0;
        wire cc1 , ss1 , ss0;
        sc_ass UUT(aa1 , aa0 , bb1 , bb0 , cc0 , cc1 , ss1 , ss0);
        initial begin
                aa1 = 1'b1; aa0 = 1'b1;
                bb1 = 1'b1; bb0 = 1'b0;
                cc0 = 1'b1;
                #400
                aa1 = 1'b1; aa0 = 1'b0;
                bb1 = 1'b0; bb0 = 1'b0;
                cc0 = 1'b0;
                #400
                aa1 = 1'b1; aa0 = 1'b0;
                bb1 = 1'b1; bb0 = 1'b0;
                cc0 = 1'b1;
                #400
                aa1 = 1'b0; aa0 = 1'b0;
                bb1 = 1'b0; bb0 = 1'b1;
                cc0 = 1'b0;
                #400 $stop;
        end
endmodule
```

Testbench



```
                Msgs
/tst2/aa1       0
/tst2/aa0       0
/tst2/bb1       0
/tst2/bb0       1
/tst2/cc0       0
/tst2/cc1       St0
/tst2/ss1       St0
/tst2/ss0       St1
```

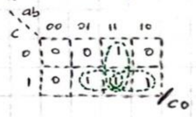
3.

3. For this part I built a 3-input adder with 2-bits as output.

Truth table:

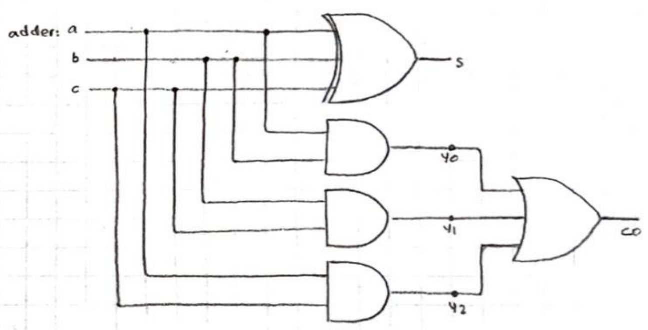
a	b	c	co	S
0	0	0	0	0
0	1	0	0	1
1	1	0	1	0
1	0	0	0	1
0	0	1	0	1
0	1	1	1	0
1	1	1	1	1
1	0	1	1	0

↓
The same as 3-input XOR truth table



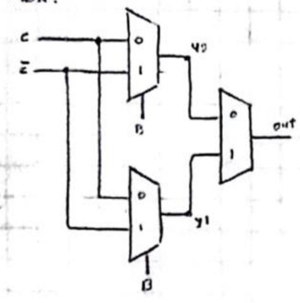
↳ $co = ab + ac + bc$

⇒

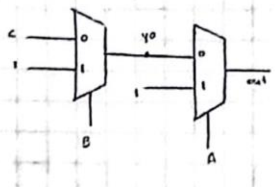


- now we must build XOR, AND and OR with MUX:

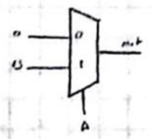
1. XOR:



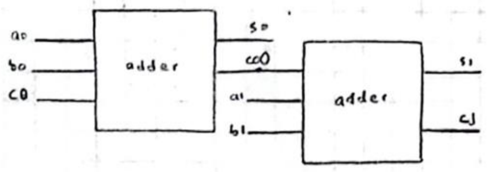
2. OR:

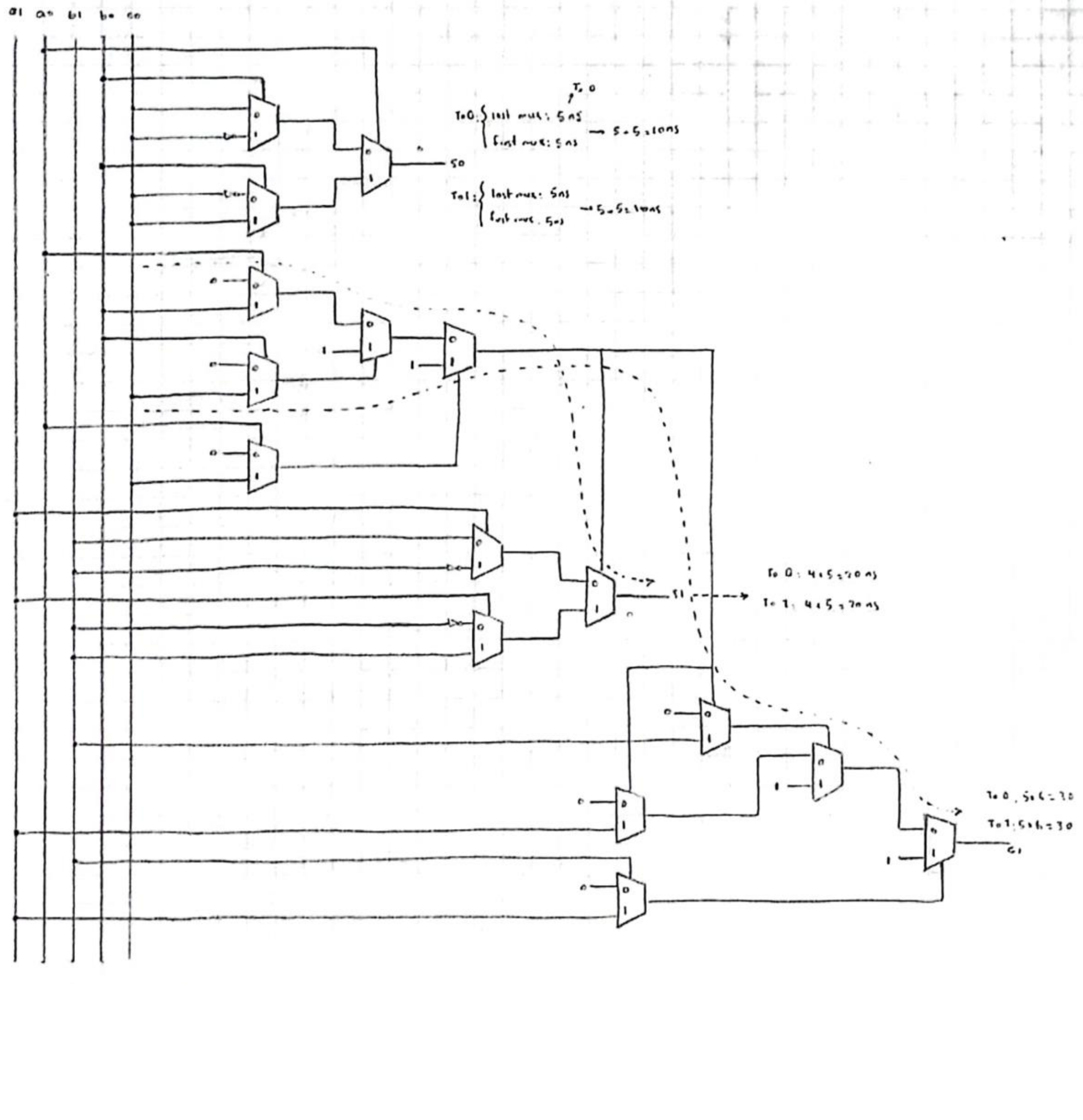


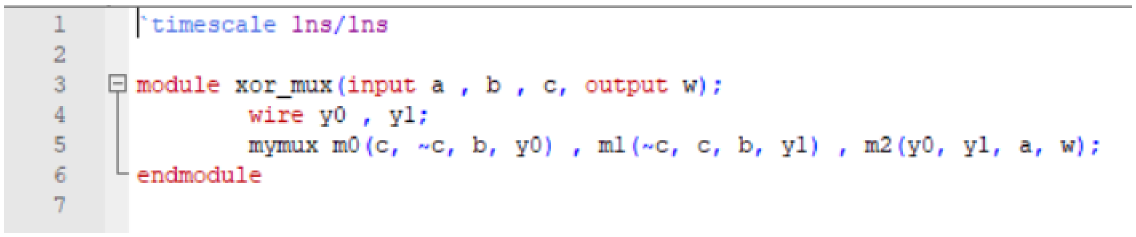
3. AND:



• SC calculator:

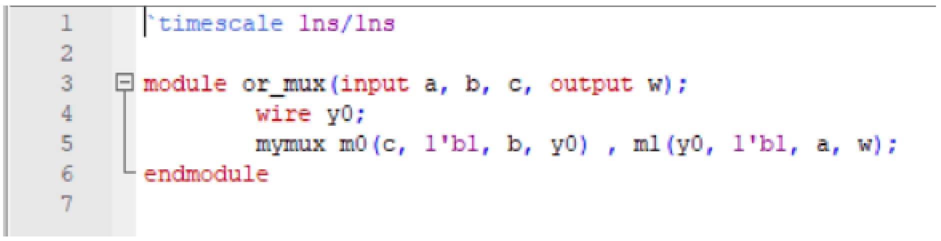



a1 a0 b1 b0 c0
T=0
T=0: { last mux: 5ns / first mux: 5ns → 5+5=10ns
S0
T=1: { last mux: 5ns / first mux: 5ns → 5+5=10ns
S1
T=0: 4×5=20 ns
T=1: 4×5=20 ns
T=0: 5×6=30
T=1: 5×6=30
C1



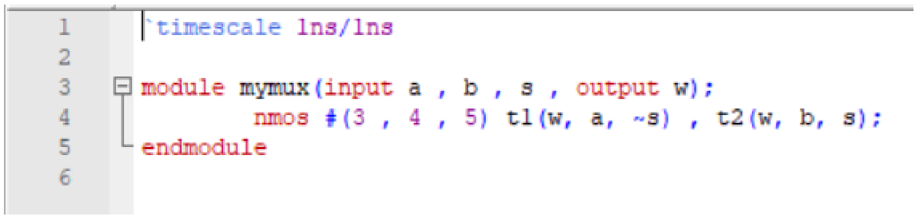
```
`timescale 1ns/1ns

module xor_mux(input a , b , c, output w);
        wire y0 , y1;
        mymux m0(c, ~c, b, y0) , m1(~c, c, b, y1) , m2(y0, y1, a, w);
endmodule

```

Xor



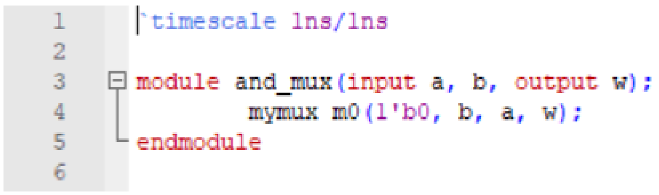
```
`timescale 1ns/1ns

module or_mux(input a, b, c, output w);
        wire y0;
        mymux m0(c, 1'b1, b, y0) , m1(y0, 1'b1, a, w);
endmodule

```

Or



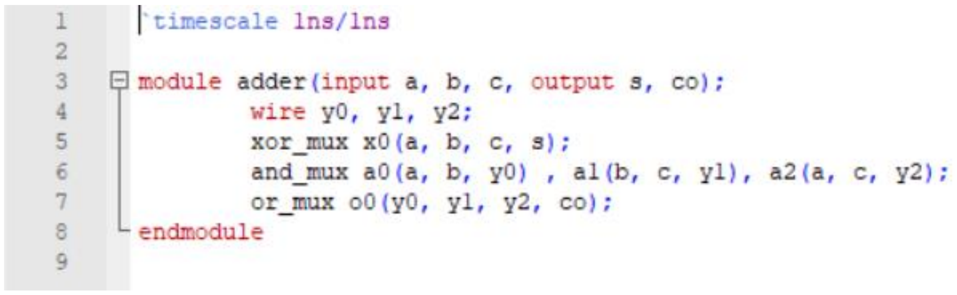
```
`timescale 1ns/1ns

module mymux(input a , b , s , output w);
        nmos #(3 , 4 , 5) t1(w, a, ~s) , t2(w, b, s);
endmodule

```

Mux

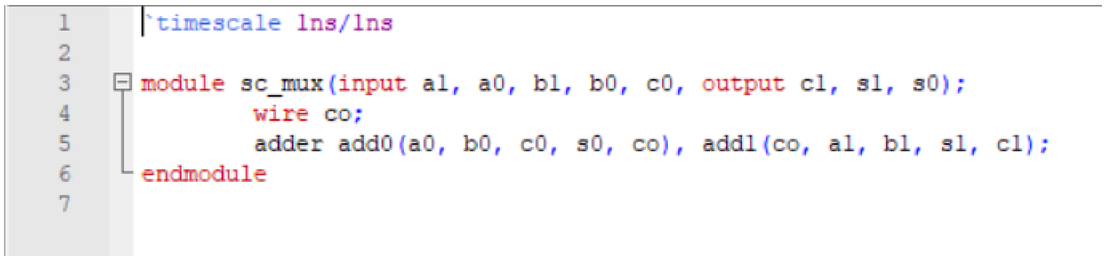


```
`timescale 1ns/1ns

module and_mux(input a, b, output w);
        mymux m0(1'b0, b, a, w);
endmodule

```

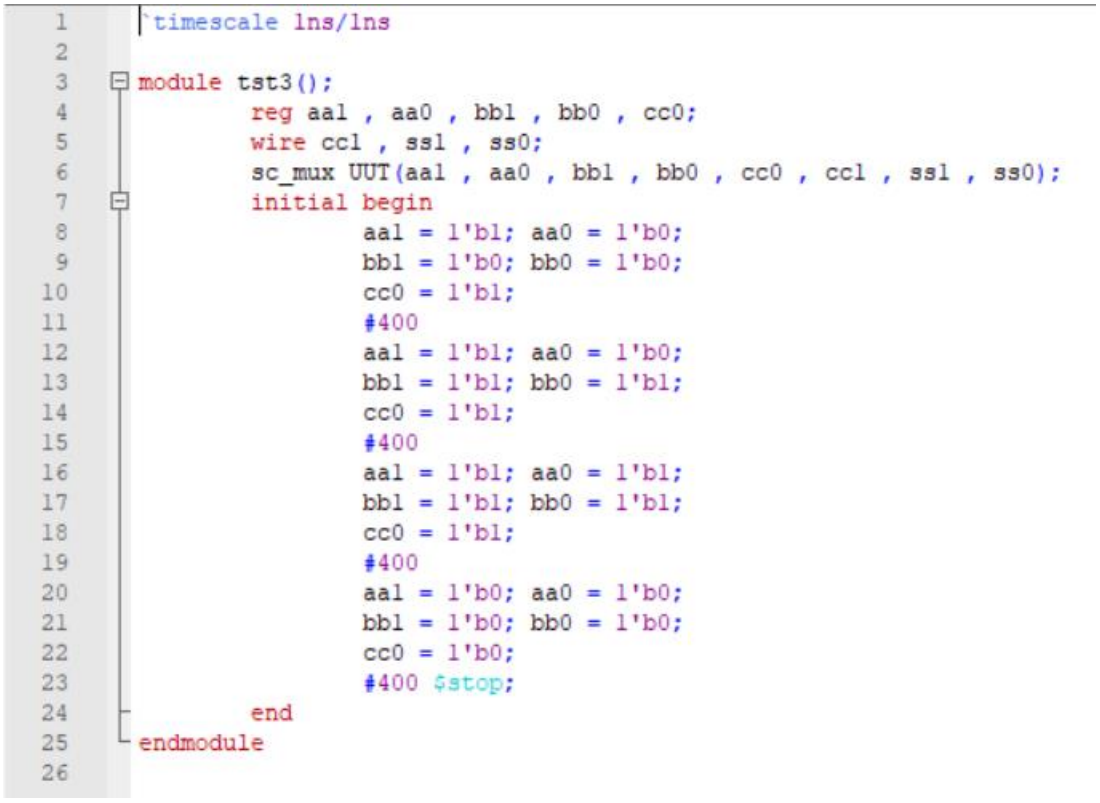
And



```
`timescale 1ns/1ns

module adder(input a, b, c, output s, co);
        wire y0, y1, y2;
        xor_mux x0(a, b, c, s);
        and_mux a0(a, b, y0) , a1(b, c, y1), a2(a, c, y2);
        or_mux o0(y0, y1, y2, co);
endmodule

```

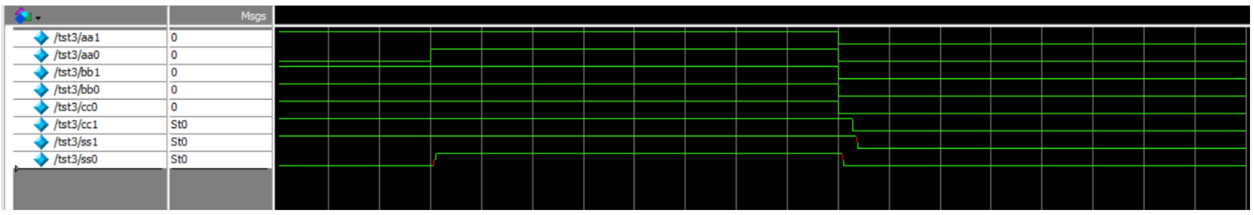


```verilog
`timescale 1ns/1ns

module sc_mux(input a1, a0, b1, b0, c0, output c1, s1, s0);
        wire co;
        adder add0(a0, b0, c0, s0, co), add1(co, a1, b1, s1, c1);
endmodule

```



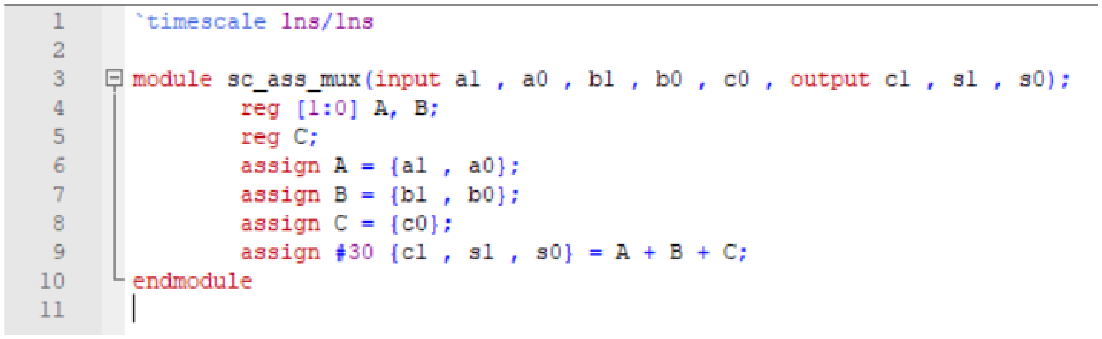
```verilog
`timescale 1ns/1ns

module tst3();
        reg aa1 , aa0 , bb1 , bb0 , cc0;
        wire cc1 , ss1 , ss0;
        sc_mux UUT(aa1 , aa0 , bb1 , bb0 , cc0 , cc1 , ss1 , ss0);
        initial begin
                aa1 = 1'b1; aa0 = 1'b0;
                bb1 = 1'b0; bb0 = 1'b0;
                cc0 = 1'b1;
                #400
                aa1 = 1'b1; aa0 = 1'b0;
                bb1 = 1'b1; bb0 = 1'b1;
                cc0 = 1'b1;
                #400
                aa1 = 1'b1; aa0 = 1'b1;
                bb1 = 1'b1; bb0 = 1'b1;
                cc0 = 1'b1;
                #400
                aa1 = 1'b0; aa0 = 1'b0;
                bb1 = 1'b0; bb0 = 1'b0;
                cc0 = 1'b0;
                #400 $stop;
        end
endmodule

```

Testbench

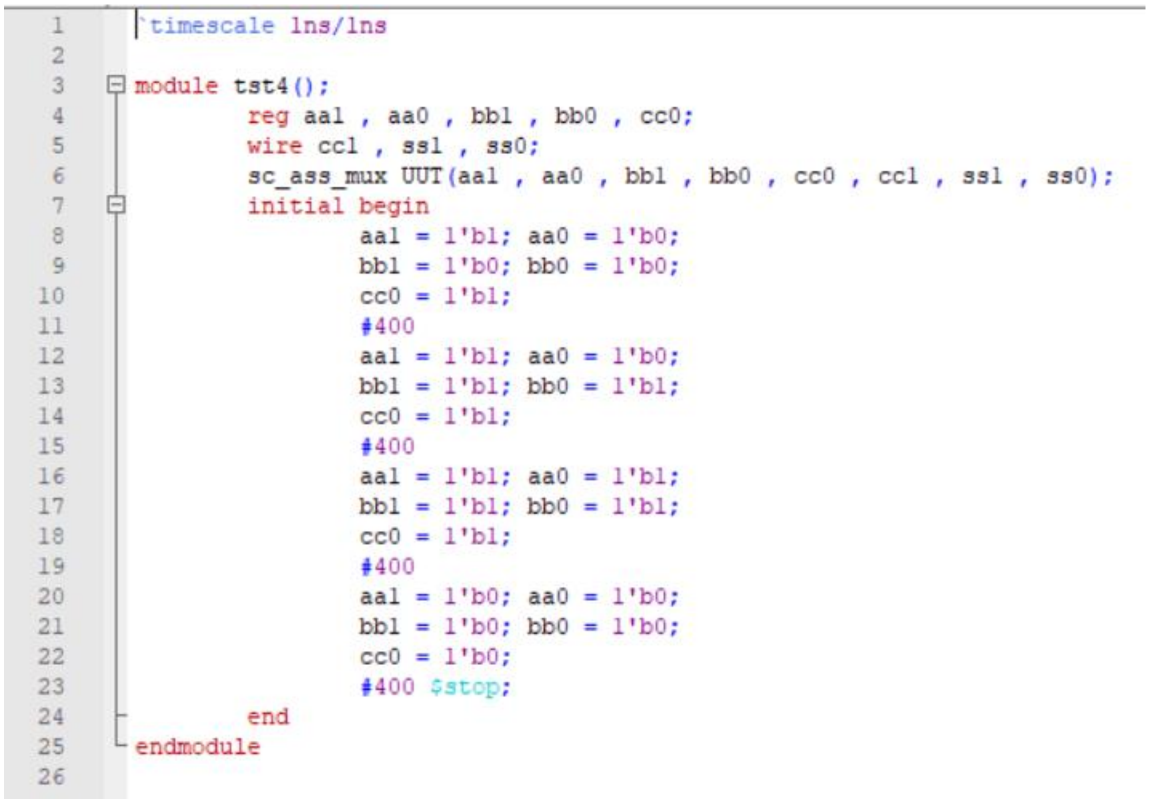


```
             Msgs
/tst3/aa1    0
/tst3/aa0    0
/tst3/bb1    0
/tst3/bb0    0
/tst3/cc0    0
/tst3/cc1    St0
/tst3/ss1    St0
/tst3/ss0    St0
```

4.

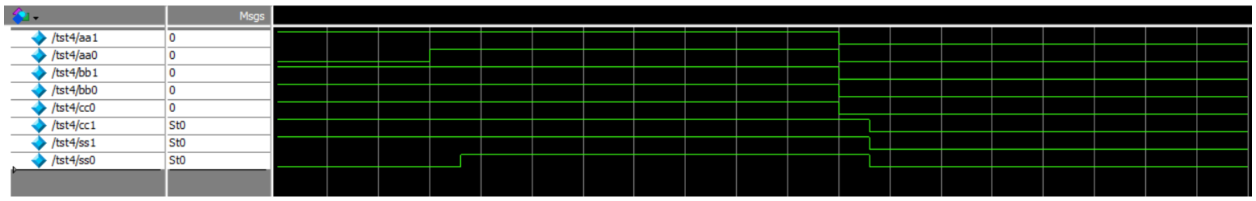


```
`timescale 1ns/1ns

module sc_ass_mux(input a1 , a0 , b1 , b0 , c0 , output c1 , s1 , s0);
        reg [1:0] A, B;
        reg C;
        assign A = {a1 , a0};
        assign B = {b1 , b0};
        assign C = {c0};
        assign #30 {c1 , s1 , s0} = A + B + C;
endmodule
```



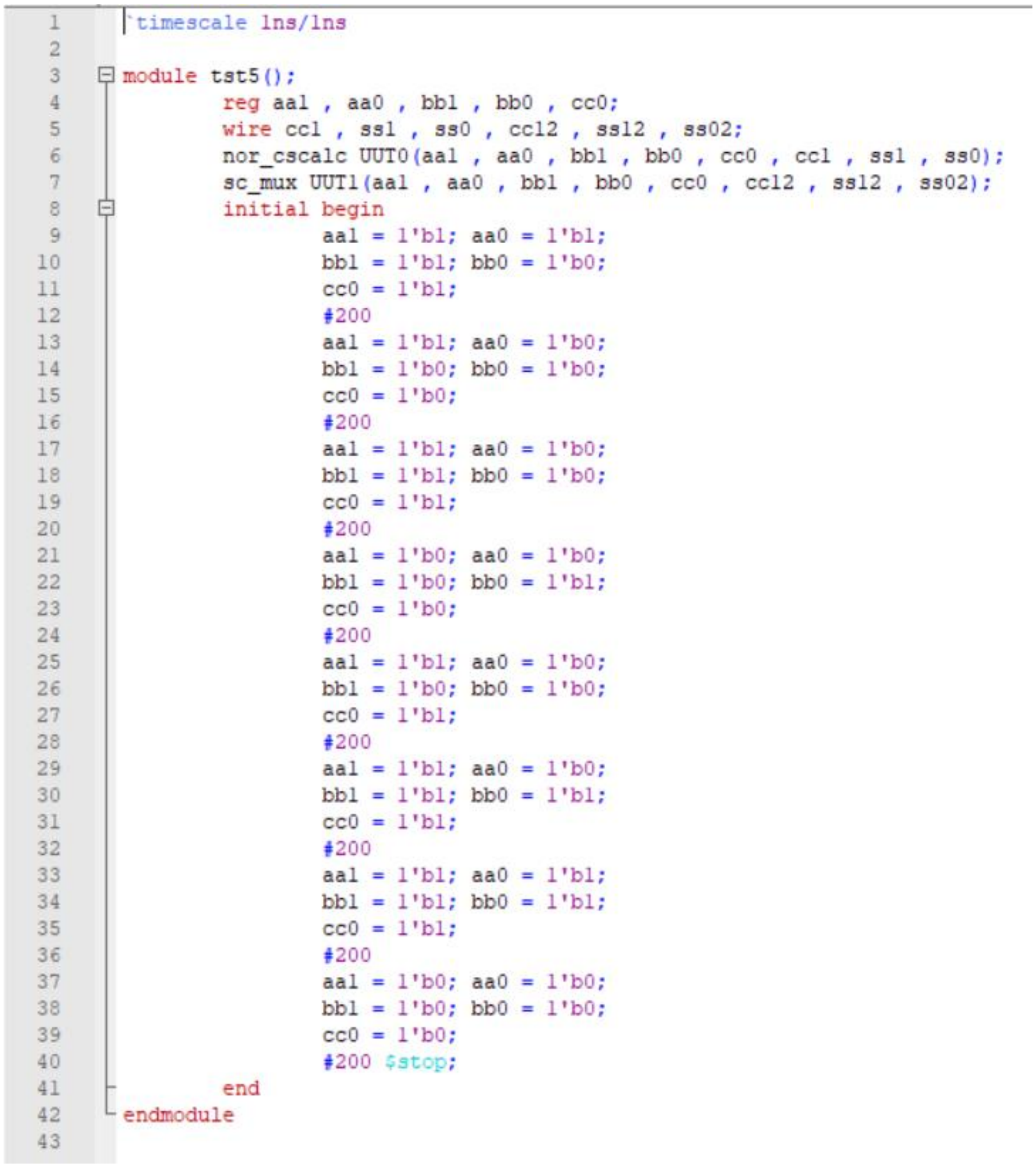
```
`timescale 1ns/1ns

module tst4();
        reg aa1 , aa0 , bb1 , bb0 , cc0;
        wire cc1 , ss1 , ss0;
        sc_ass_mux UUT(aa1 , aa0 , bb1 , bb0 , cc0 , cc1 , ss1 , ss0);
        initial begin
                aa1 = 1'b1; aa0 = 1'b0;
                bb1 = 1'b0; bb0 = 1'b0;
                cc0 = 1'b1;
                #400
                aa1 = 1'b1; aa0 = 1'b0;
                bb1 = 1'b1; bb0 = 1'b1;
                cc0 = 1'b1;
                #400
                aa1 = 1'b1; aa0 = 1'b1;
                bb1 = 1'b1; bb0 = 1'b1;
                cc0 = 1'b1;
                #400
                aa1 = 1'b0; aa0 = 1'b0;
                bb1 = 1'b0; bb0 = 1'b0;
                cc0 = 1'b0;
                #400 $stop;
        end
endmodule

```

Testbench



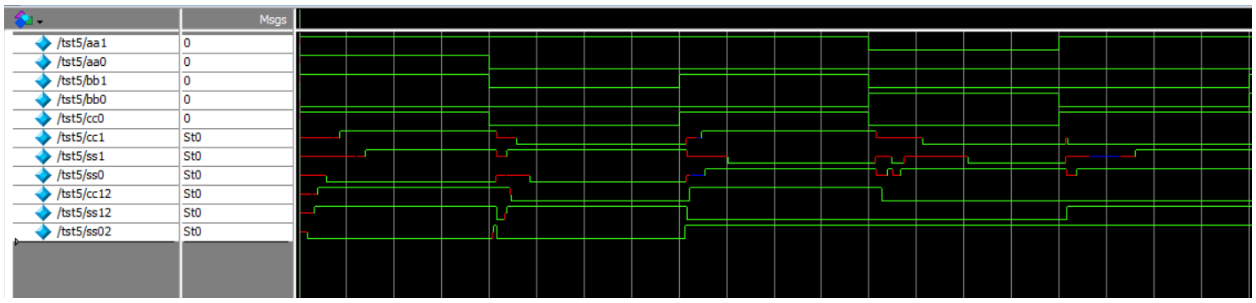
```
                Msgs
/tst4/aa1       0
/tst4/aa0       0
/tst4/bb1       0
/tst4/bb0       0
/tst4/cc0       0
/tst4/cc1       St0
/tst4/ss1       St0
/tst4/ss0       St0
```

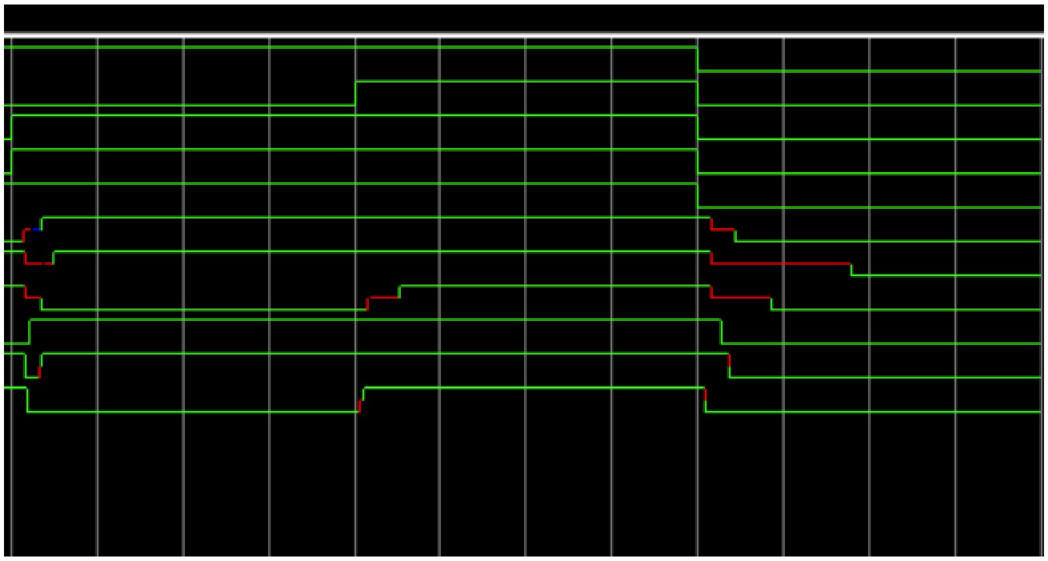
5.



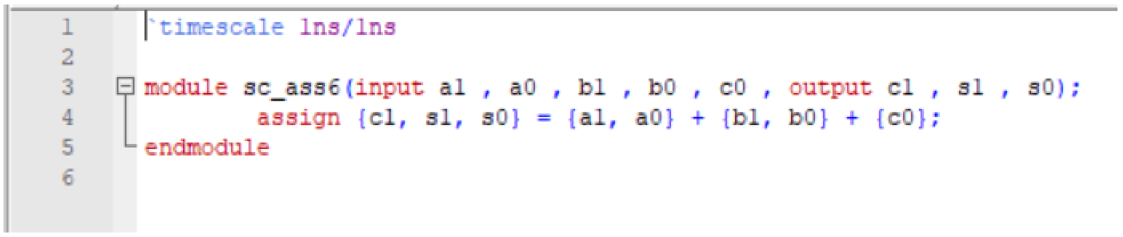
```
`timescale 1ns/1ns

module tst5();
        reg aa1 , aa0 , bb1 , bb0 , cc0;
        wire cc1 , ss1 , ss0 , cc12 , ss12 , ss02;
        nor_cscalc UUT0(aa1 , aa0 , bb1 , bb0 , cc0 , cc1 , ss1 , ss0);
        sc_mux UUT1(aa1 , aa0 , bb1 , bb0 , cc0 , cc12 , ss12 , ss02);
        initial begin
                aa1 = 1'b1; aa0 = 1'b1;
                bb1 = 1'b1; bb0 = 1'b0;
                cc0 = 1'b1;
                #200
                aa1 = 1'b1; aa0 = 1'b0;
                bb1 = 1'b0; bb0 = 1'b0;
                cc0 = 1'b0;
                #200
                aa1 = 1'b1; aa0 = 1'b0;
                bb1 = 1'b1; bb0 = 1'b0;
                cc0 = 1'b1;
                #200
                aa1 = 1'b0; aa0 = 1'b0;
                bb1 = 1'b0; bb0 = 1'b1;
                cc0 = 1'b0;
                #200
                aa1 = 1'b1; aa0 = 1'b0;
                bb1 = 1'b0; bb0 = 1'b0;
                cc0 = 1'b1;
                #200
                aa1 = 1'b1; aa0 = 1'b0;
                bb1 = 1'b1; bb0 = 1'b1;
                cc0 = 1'b1;
                #200
                aa1 = 1'b1; aa0 = 1'b1;
                bb1 = 1'b1; bb0 = 1'b1;
                cc0 = 1'b1;
                #200
                aa1 = 1'b0; aa0 = 1'b0;
                bb1 = 1'b0; bb0 = 1'b0;
                cc0 = 1'b0;
                #200 $stop;
        end
endmodule
```

Testbench

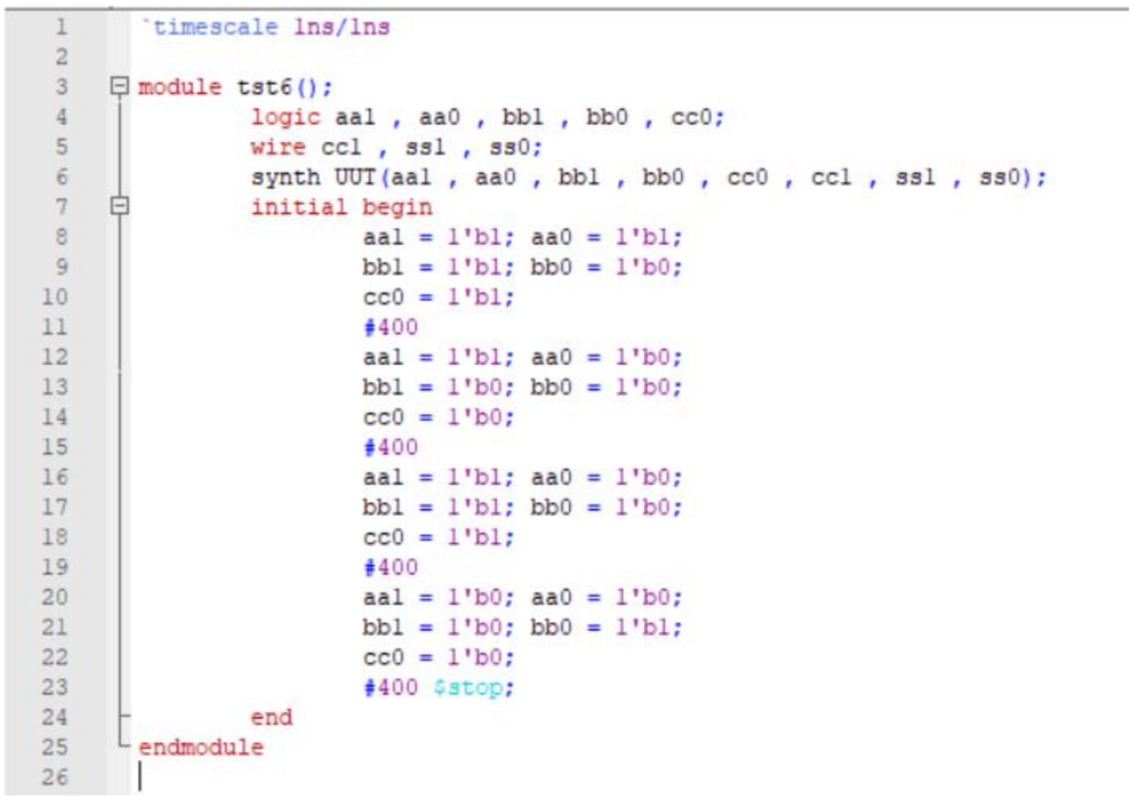




6.



```
`timescale 1ns/1ns

module sc_ass6(input a1 , a0 , b1 , b0 , c0 , output c1 , s1 , s0);
        assign {c1, s1, s0} = {a1, a0} + {b1, b0} + {c0};
endmodule

```



```
`timescale 1ns/1ns

module tst6();
        logic aa1 , aa0 , bb1 , bb0 , cc0;
        wire cc1 , ss1 , ss0;
        synth UUT(aa1 , aa0 , bb1 , bb0 , cc0 , cc1 , ss1 , ss0);
        initial begin
                aa1 = 1'b1; aa0 = 1'b1;
                bb1 = 1'b1; bb0 = 1'b0;
                cc0 = 1'b1;
                #400
                aa1 = 1'b1; aa0 = 1'b0;
                bb1 = 1'b0; bb0 = 1'b0;
                cc0 = 1'b0;
                #400
                aa1 = 1'b1; aa0 = 1'b0;
                bb1 = 1'b1; bb0 = 1'b0;
                cc0 = 1'b1;
                #400
                aa1 = 1'b0; aa0 = 1'b0;
                bb1 = 1'b0; bb0 = 1'b1;
                cc0 = 1'b0;
                #400 $stop;
        end
endmodule
```

Testbench