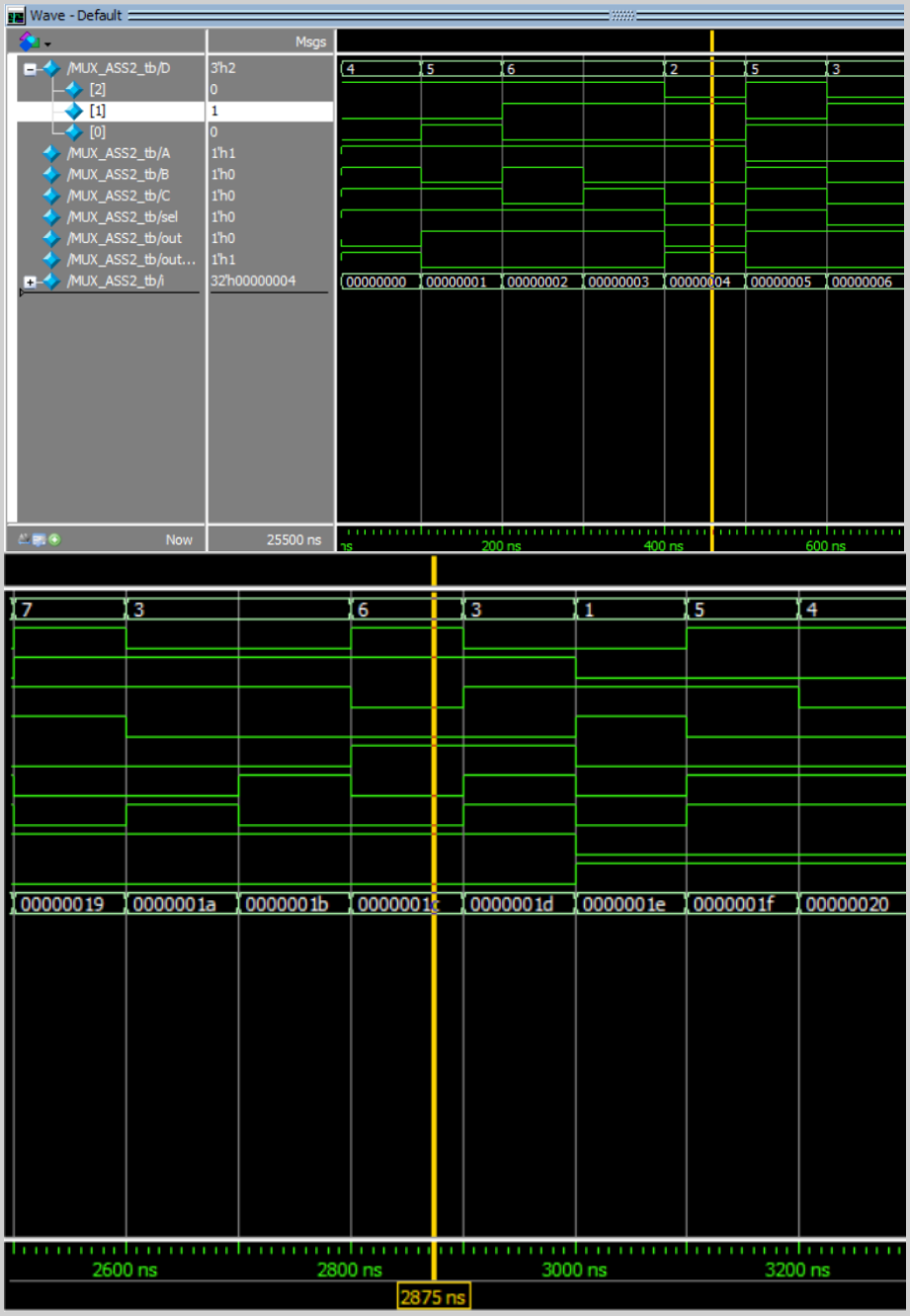
ASSINMENT 2

GROUP	NAME
1	Ahmed Khalaf Mohamed Ali

1)

```
Ln#
 1  module MUX_ASS2(
 2  input [2:0]D,
 3  input A,B,C,sel,
 4  output out,out_bar
 5      );
 6      wire AND1,OR1,XNOR1;
 7      and(AND1,D[0],D[1]);
 8      or(OR1,AND1,D[2]);
 9      xnor(XNOR1,A,B,C);
10      assign out=(sel)?XNOR1:OR1;
11      assign out_bar=~out;
12  endmodule
13
```

```
Ln#
 1  module MUX_ASS2_tb();
 2  /*wire and reg*/
 3  reg [2:0]D;
 4  reg A,B,C,sel;
 5  wire out,out_bar;
 6  /*instantiation*/
 7  MUX_ASS2 DUT (D,A,B,C,sel,out,out_bar);
 8  /*values*/
 9  integer i;
10  initial begin
11  for(i=0 ; i<255; i=i+1) begin
12  D=$random;
13  A=$random;
14  B=$random;
15  C=$random;
16  sel=$random;
17  #100;
18  end
19  $stop;
20  end
21  initial begin
22      $monitor("D=%b, A=%b, B=%b, C=%b, sel=%b, out=%b, out_bar=%b", D, A, B, C, sel, out, out_bar);
23  end
24
25  endmodule
26
```


Wave - Default
Msgs
/MUX_ASS2_tb/D
[2]
[1]
[0]
/MUX_ASS2_tb/A
/MUX_ASS2_tb/B
/MUX_ASS2_tb/C
/MUX_ASS2_tb/sel
/MUX_ASS2_tb/out
/MUX_ASS2_tb/out...
/MUX_ASS2_tb/i
3'h2
0
1
0
1'h1
1'h0
1'h0
1'h0
1'h1
32'h00000004
4
5
6
2
5
3
00000000
00000001
00000002
00000003
00000004
00000005
00000006
Now
25500 ns
200 ns
400 ns
600 ns
7
3
6
3
1
5
4
00000019
0000001a
0000001b
0000001c
0000001d
0000001e
0000001f
00000020
2600 ns
2800 ns
3000 ns
3200 ns
2875 ns

```
# D=011, A=1, B=0, C=0, sel=0, out=1, out_bar=0
# D=100, A=0, B=1, C=1, sel=1, out=1, out_bar=0
# D=110, A=0, B=0, C=1, sel=0, out=1, out_bar=0
# D=011, A=0, B=1, C=1, sel=1, out=1, out_bar=0
# D=001, A=0, B=1, C=1, sel=0, out=0, out_bar=1
# D=010, A=1, B=0, C=0, sel=0, out=0, out_bar=1
# D=101, A=0, B=0, C=1, sel=0, out=1, out_bar=0
# D=101, A=1, B=1, C=0, sel=1, out=1, out_bar=0
# D=100, A=1, B=1, C=1, sel=0, out=1, out_bar=0
# D=100, A=0, B=0, C=0, sel=0, out=1, out_bar=0
# D=010, A=1, B=1, C=1, sel=1, out=0, out_bar=1
# D=011, A=1, B=1, C=0, sel=0, out=1, out_bar=0
# D=111, A=1, B=1, C=0, sel=1, out=1, out_bar=0
# D=001, A=0, B=1, C=0, sel=1, out=0, out_bar=1
# D=101, A=1, B=0, C=1, sel=0, out=1, out_bar=0
# D=101, A=1, B=0, C=0, sel=0, out=1, out_bar=0
# D=111, A=0, B=0, C=0, sel=0, out=1, out_bar=0
# D=000, A=1, B=0, C=0, sel=0, out=0, out_bar=1
# D=111, A=0, B=1, C=1, sel=0, out=1, out_bar=0
# D=101, A=0, B=0, C=0, sel=0, out=1, out_bar=0
# D=101, A=1, B=1, C=1, sel=0, out=1, out_bar=0
# D=110, A=1, B=0, C=1, sel=1, out=1, out_bar=0
# D=111, A=0, B=1, C=1, sel=0, out=1, out_bar=0
# D=001, A=1, B=1, C=1, sel=0, out=0, out_bar=1
# D=110, A=0, B=1, C=1, sel=1, out=1, out_bar=0
# D=100, A=1, B=0, C=0, sel=1, out=0, out_bar=1
# D=001, A=0, B=0, C=1, sel=0, out=0, out_bar=1
# D=011, A=0, B=1, C=1, sel=0, out=1, out_bar=0
# D=001, A=1, B=1, C=0, sel=0, out=0, out_bar=1
# D=010, A=1, B=0, C=0, sel=1, out=0, out_bar=1
# D=000, A=0, B=1, C=0, sel=0, out=0, out_bar=1
# ** Note: $stop    : C:/questasim64_2021.1/examples/MUX_ASS2_tb.v(19)
#    Time: 25500 ns  Iteration: 0  Instance: /MUX_ASS2_tb
# Break in Module MUX_ASS2_tb at C:/questasim64_2021.1/examples/MUX_ASS2_tb.v line 19
```

2)

C:/questasim64_2021.1/examples/priority_encoder.v (/PRIORITY_ENCODER_tb/DUT) - Default

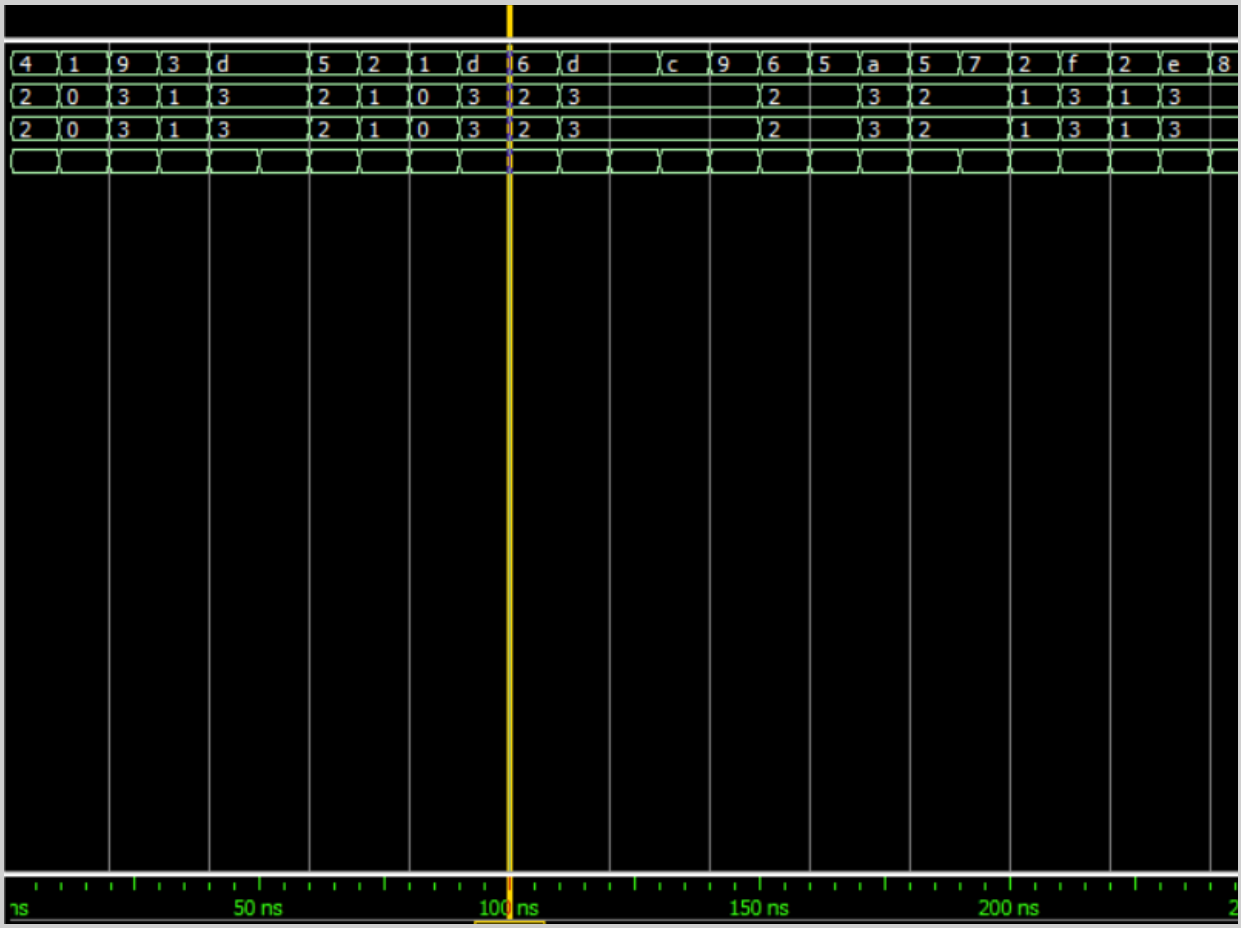
```
Ln#
 1  module PRIORITY_ENCODER(X,Y);
 2  input [3:0] X;
 3  output reg [1:0] Y;
 4  always @(X) begin
 5  casex(X)
 6  4'b1xxx : Y=2'b11;
 7  4'b01xx : Y=2'b10;
 8  4'b001x : Y=2'b01;
 9  4'b000x : Y=2'b00;
10  endcase
11  end
12  endmodule
13
```

C:/questasim64_2021.1/examples/priority_encoder_tb.v (/PRIORITY_ENCODER_tb) - Default

```
Ln#
 3  reg [1:0] Y_EXPECTED;
 4  wire [1:0] Y_tb;
 5  PRIORITY_ENCODER DUT(X_tb,Y_tb);
 6  /*initial */
 7  integer i;
 8  initial begin
 9  for(i=0 ; i<99; i=i+1) begin
10  X_tb=$random;
11  #10;
12  casex(X_tb)
13  4'b1xxx : Y_EXPECTED=2'b11;
14  4'b01xx : Y_EXPECTED=2'b10;
15  4'b001x : Y_EXPECTED=2'b01;
16  4'b000x : Y_EXPECTED=2'b00;
17  endcase
18  if(Y_EXPECTED != Y_tb) begin
19  $display("error ,there are some thing not correct1");
20  end
21
22  end
23  $stop;
24  end
25  initial begin
26
27      $monitor("X_tb=%b,Y_EXPECTED=%b,Y_tb=%b",X_tb,Y_EXPECTED,Y_tb);
28
29  end
30  endmodule
31
```

C:/questasim64_2021.1/examples/priority_encoder_tb.v (/PRIORITY_ENCODER_tb) - Default

```
Ln#
 4   wire [1:0] Y_tb;
 5   PRIORITY_ENCODER DUT(X_tb,Y_tb);
 6   /*initial */
 7   integer i;
 8   initial begin
 9   for(i=0 ; i<99; i=i+1) begin
10   X_tb=$random;
11   #0;
12   casex(X_tb)
13   4'b1xxx : Y_EXPECTED=2'b11;
14   4'b01xx : Y_EXPECTED=2'b10;
15   4'b001x : Y_EXPECTED=2'b01;
16   4'b000x : Y_EXPECTED=2'b00;
17   endcase
18   if(Y_EXPECTED != Y_tb) begin
19   $display("error ,there are some thing not correctl");
20   end
21   #10;
22   end
23   X_tb=4'b1111;  Y_EXPECTED=2'b00;
24   if(Y_EXPECTED != Y_tb) begin
25   $display("error ,there are some thing not correctl");
26   end
27   $stop;
28   end
29   initial begin
30
31       $monitor("X_tb=%b,Y_EXPECTED=%b,Y_tb=%b",X_tb,Y_EXPECTED,Y_tb);
32
33   end
34   endmodule
35
```


4 1 9 3 d 5 2 1 d 6 d c 9 6 5 a 5 7 2 f 2 e 8
2 0 3 1 3 2 1 0 3 2 3 2 3 2 1 3 1 3
2 0 3 1 3 2 1 0 3 2 3 2 3 2 1 3 1 3
ns
50 ns
100 ns
150 ns
200 ns

```
# X_tb=1010,Y_EXPECTED=11,Y_tb=11
# X_tb=1110,Y_EXPECTED=11,Y_tb=11
# X_tb=0101,Y_EXPECTED=10,Y_tb=10
# X_tb=0001,Y_EXPECTED=00,Y_tb=00
# X_tb=1001,Y_EXPECTED=11,Y_tb=11
# X_tb=0010,Y_EXPECTED=01,Y_tb=01
# X_tb=1100,Y_EXPECTED=11,Y_tb=11
# X_tb=1111,Y_EXPECTED=11,Y_tb=11
# X_tb=1000,Y_EXPECTED=11,Y_tb=11
# X_tb=0111,Y_EXPECTED=10,Y_tb=10
# X_tb=1111,Y_EXPECTED=11,Y_tb=11
# X_tb=1100,Y_EXPECTED=11,Y_tb=11
# X_tb=1011,Y_EXPECTED=11,Y_tb=11
# X_tb=1001,Y_EXPECTED=11,Y_tb=11
# X_tb=0000,Y_EXPECTED=00,Y_tb=00
# X_tb=0111,Y_EXPECTED=10,Y_tb=10
# X_tb=0001,Y_EXPECTED=00,Y_tb=00
# X_tb=0110,Y_EXPECTED=10,Y_tb=10
# X_tb=1100,Y_EXPECTED=11,Y_tb=11
# X_tb=0010,Y_EXPECTED=01,Y_tb=01
# X_tb=1000,Y_EXPECTED=11,Y_tb=11
# X_tb=0111,Y_EXPECTED=10,Y_tb=10
# X_tb=1101,Y_EXPECTED=11,Y_tb=11
# X_tb=0010,Y_EXPECTED=01,Y_tb=01
# X_tb=1110,Y_EXPECTED=11,Y_tb=11
# X_tb=1101,Y_EXPECTED=11,Y_tb=11
# X_tb=1001,Y_EXPECTED=11,Y_tb=11
# X_tb=1111,Y_EXPECTED=11,Y_tb=11
# X_tb=0011,Y_EXPECTED=01,Y_tb=01
# X_tb=0101,Y_EXPECTED=10,Y_tb=10
# error ,there are some thing not correctl
```

3)

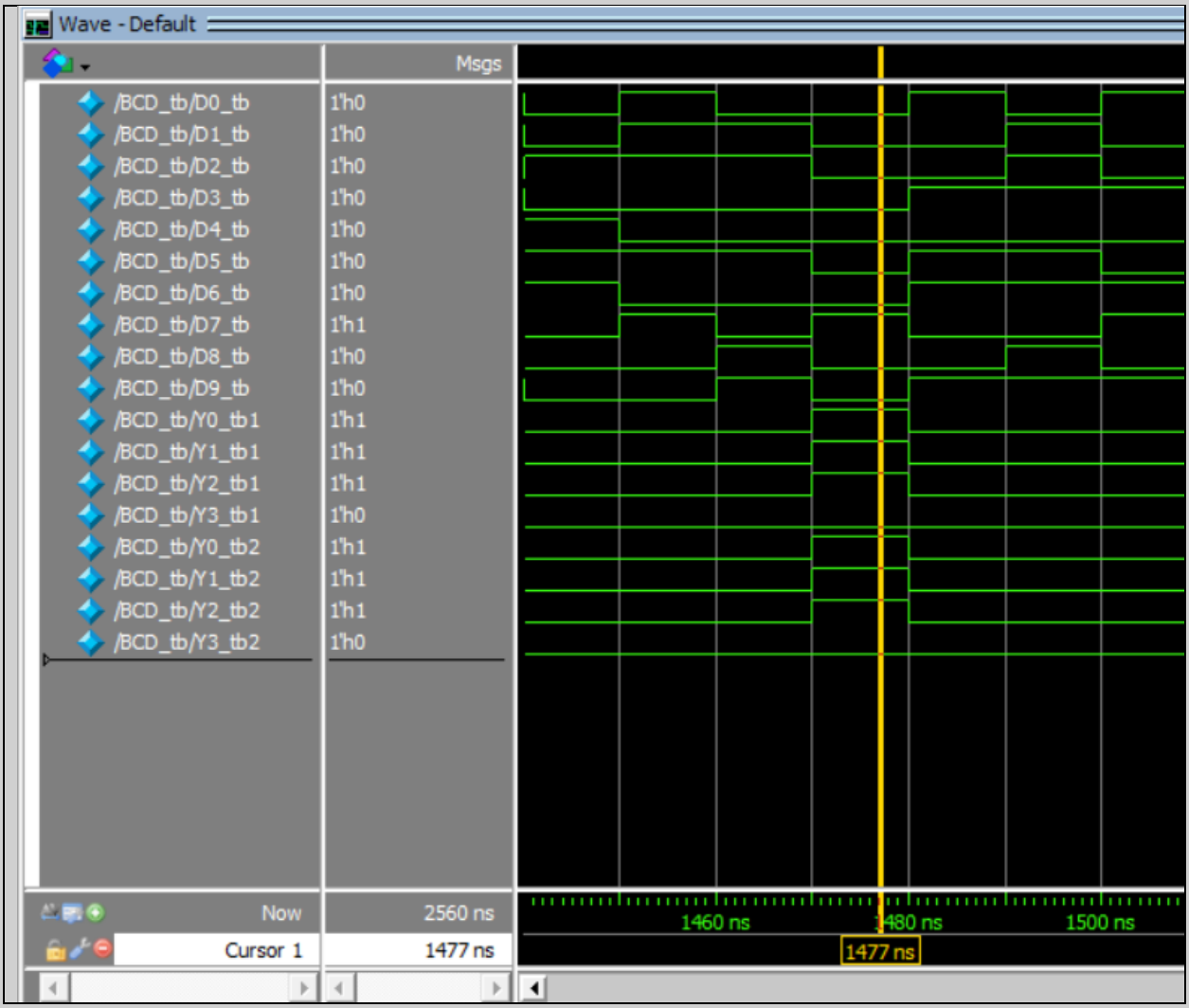
C:/questasim64_2021.1/examples/BCD.v (/BCD_tb/DUT1) - Default

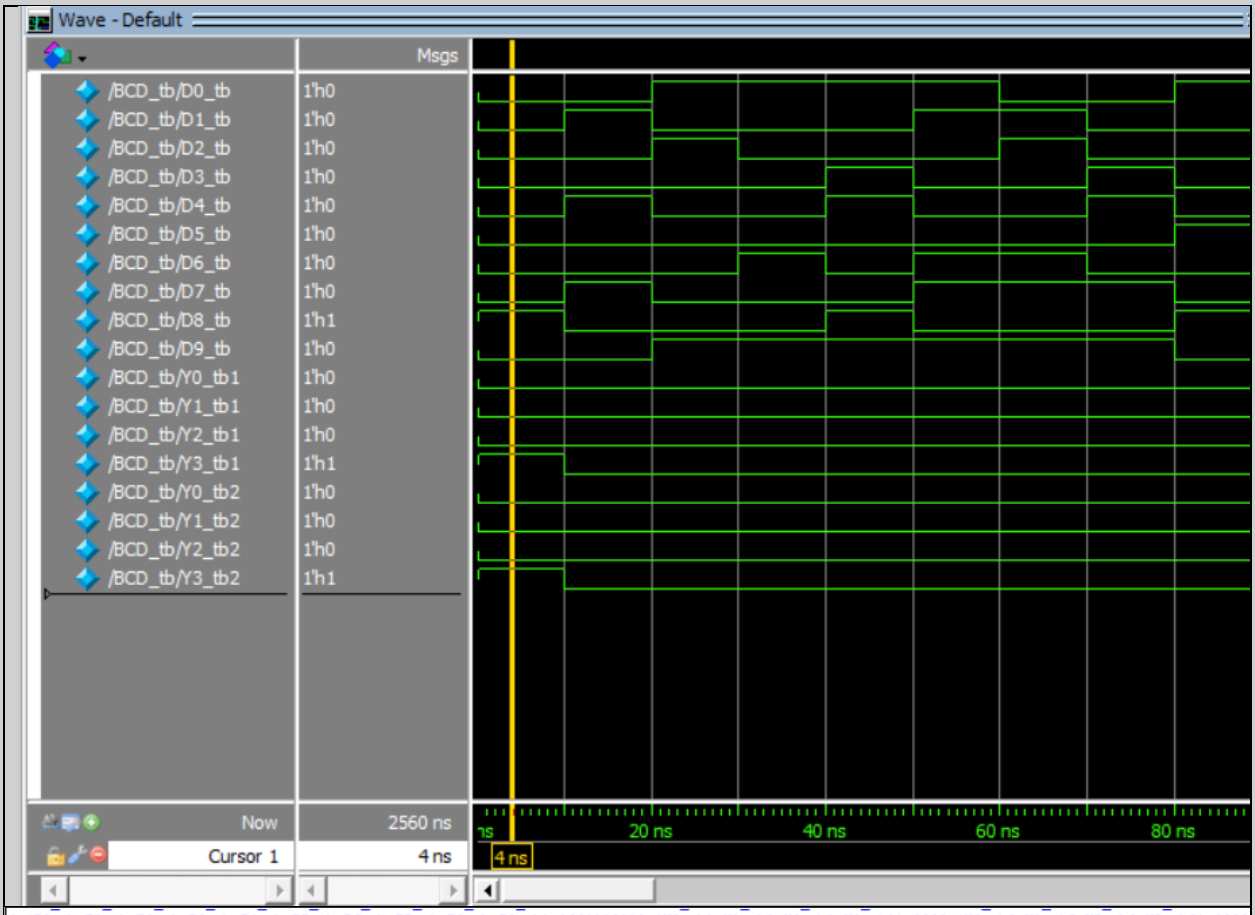
```
Ln#
 1  module BCD(D0,D1,D2,D3,D4,D5,D6,D7,D8,D9,Y0,Y1,Y2,Y3);
 2  input D0,D1,D2,D3,D4,D5,D6,D7,D8,D9;
 3  output reg Y0,Y1,Y2,Y3;
 4  always @(*) begin
 5  case({D9,D8,D7,D6,D5,D4,D3,D2,D1,D0})
 6  1: {Y3,Y2,Y1,Y0}=0;
 7  2: {Y3,Y2,Y1,Y0}=1;
 8  4: {Y3,Y2,Y1,Y0}=2;
 9  8: {Y3,Y2,Y1,Y0}=3;
10  16: {Y3,Y2,Y1,Y0}=4;
11  32: {Y3,Y2,Y1,Y0}=5;
12  64: {Y3,Y2,Y1,Y0}=6;
13  128: {Y3,Y2,Y1,Y0}=7;
14  256: {Y3,Y2,Y1,Y0}=8;
15  512: {Y3,Y2,Y1,Y0}=9;
16  default : {Y3,Y2,Y1,Y0}=0;
17  endcase
18  end
19  endmodule
20
```

C:/questasim64_2021.1/examples/BCD2.v (/BCD_tb/DUT2) - Default

```
Ln#
 1  module BCD2(D0,D1,D2,D3,D4,D5,D6,D7,D8,D9,Y0,Y1,Y2,Y3);
 2  input D0,D1,D2,D3,D4,D5,D6,D7,D8,D9;
 3  output reg Y0,Y1,Y2,Y3;
 4
 5  always @(*) begin
 6      if ({D9, D8, D7, D6, D5, D4, D3, D2, D1, D0} == 10'b0000000001)
 7          {Y3, Y2, Y1, Y0} = 4'b0000;
 8      else if ({D9, D8, D7, D6, D5, D4, D3, D2, D1, D0} == 10'b0000000010)
 9          {Y3, Y2, Y1, Y0} = 4'b0001;
10      else if ({D9, D8, D7, D6, D5, D4, D3, D2, D1, D0} == 10'b0000000100)
11          {Y3, Y2, Y1, Y0} = 4'b0010;
12      else if ({D9, D8, D7, D6, D5, D4, D3, D2, D1, D0} == 10'b0000001000)
13          {Y3, Y2, Y1, Y0} = 4'b0011;
14      else if ({D9, D8, D7, D6, D5, D4, D3, D2, D1, D0} == 10'b0000010000)
15          {Y3, Y2, Y1, Y0} = 4'b0100;
16      else if ({D9, D8, D7, D6, D5, D4, D3, D2, D1, D0} == 10'b0000100000)
17          {Y3, Y2, Y1, Y0} = 4'b0101;
18      else if ({D9, D8, D7, D6, D5, D4, D3, D2, D1, D0} == 10'b0001000000)
19          {Y3, Y2, Y1, Y0} = 4'b0110;
20      else if ({D9, D8, D7, D6, D5, D4, D3, D2, D1, D0} == 10'b0010000000)
21          {Y3, Y2, Y1, Y0} = 4'b0111;
22      else if ({D9, D8, D7, D6, D5, D4, D3, D2, D1, D0} == 10'b0100000000)
23          {Y3, Y2, Y1, Y0} = 4'b1000;
24      else if ({D9, D8, D7, D6, D5, D4, D3, D2, D1, D0} == 10'b1000000000)
25          {Y3, Y2, Y1, Y0} = 4'b1001;
26      else
27          {Y3, Y2, Y1, Y0} = 4'b0000;
28  end
29
30  endmodule
31
```

```
Ln#
 1  module BCD_tb();
 2  reg D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb;
 3  wire Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1;
 4  wire Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2;
 5  BCD  DUT1(D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb,Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1);
 6  BCD2 DUT2(D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb,Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2);
 7  integer i;
 8  initial begin
 9  #0 {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=2;
10  if({Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1} != {Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}) begin
11  $display("error ,there are some thing not correct1");
12  end
13  #10;
14  for(i=0 ; i<255; i=i+1) begin
15  {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=$random;
16  if({Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1} != {Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}) begin
17  $display("error ,there are some thing not correct1");
18  end
19  #10;
20  end
21  $stop;
22  end
23  initial begin
24
25      $monitor("{D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=%b,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=%b,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=%b",{D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb
26
27  end
28  endmodule
29
```


Wave - Default
Msgs
/BCD_tb/D0_tb
/BCD_tb/D1_tb
/BCD_tb/D2_tb
/BCD_tb/D3_tb
/BCD_tb/D4_tb
/BCD_tb/D5_tb
/BCD_tb/D6_tb
/BCD_tb/D7_tb
/BCD_tb/D8_tb
/BCD_tb/D9_tb
/BCD_tb/Y0_tb1
/BCD_tb/Y1_tb1
/BCD_tb/Y2_tb1
/BCD_tb/Y3_tb1
/BCD_tb/Y0_tb2
/BCD_tb/Y1_tb2
/BCD_tb/Y2_tb2
/BCD_tb/Y3_tb2
1'h0
1'h0
1'h0
1'h0
1'h0
1'h0
1'h0
1'h1
1'h0
1'h0
1'h1
1'h1
1'h1
1'h0
1'h1
1'h1
1'h1
1'h0
Now
2560 ns
Cursor 1
1477 ns
1460 ns
1480 ns
1500 ns
1477 ns




```
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=1001110110,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=0111001010,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=0110110110,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=0110010101,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=1001000110,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=1100000100,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=0111110111,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=1101101001,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=1010110100,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=0110001000,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=0100101000,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=0000101101,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=0011000111,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=0000101110,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=0000001000,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0110,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0110
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=0100011100,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=1011111101,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=1100101001,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=0000011100,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=1110000110,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=0011011010,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=1000111101,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=1101100110,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=0001110000,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=0001110011,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=0110111010,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=1101011110,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
# {D0_tb,D1_tb,D2_tb,D3_tb,D4_tb,D5_tb,D6_tb,D7_tb,D8_tb,D9_tb}=0111111010,{Y0_tb1,Y1_tb1,Y2_tb1,Y3_tb1}=0000,{Y0_tb2,Y1_tb2,Y2_tb2,Y3_tb2}=0000
```

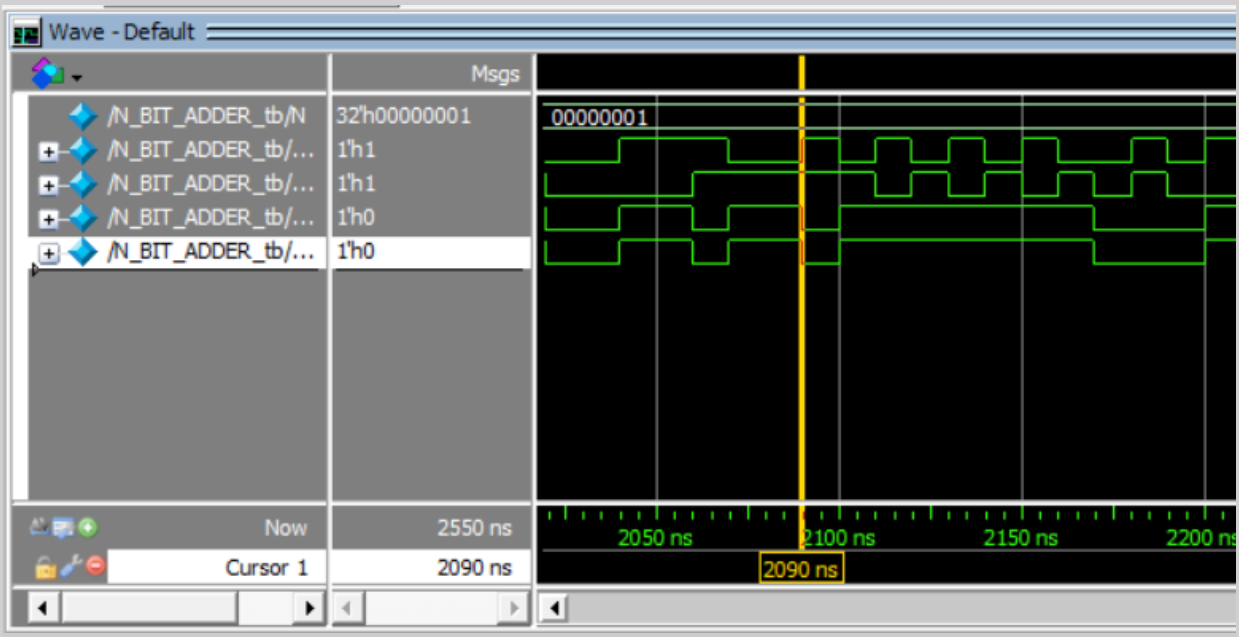
4)

C:/questasim64_2021.1/examples/N_BIT_ADDER.v (/N_BIT_ADDER_tb/DUT)

```
Ln#
1   module N_BIT_ADDER (A,B,C);
2   parameter N=1;
3   input [N-1:0] A;
4   input [N-1:0] B;
5   input [N-1:0] C;
6   assign C=A+B;
7   endmodule
8
```

C:/questasim64_2021.1/examples/N_BIT_ADDER_tb.v (/N_BIT_ADDER_tb) - Default *

```
Ln#
1   module N_BIT_ADDER_tb();
2   parameter N=1;
3   reg [N-1:0] A_tb;
4   reg [N-1:0] B_tb;
5   reg [N-1:0] Y_expected;
6   wire [N-1:0] C_tb;
7   N_BIT_ADDER #(.N(N)) DUT (A_tb,B_tb,C_tb);
8   integer i;
9   initial begin
10
11  for(i=0 ; i<4; i=i+1) begin
12  A_tb=$random;
13  B_tb=$random;
14  Y_expected=A_tb+B_tb;
15  if( C_tb != Y_expected) begin
16  $display("error ,there are some thing not correctl");
17  end
18  #10;
19  end
20  $stop;
21  end
22  initial begin
23  $monitor ("A_tb=%b,B_tb=%b,Y_expected=%b,C_tb=%b",A_tb,B_tb,Y_expected,C_tb);
24  end
25  endmodule
26
```



Transcript

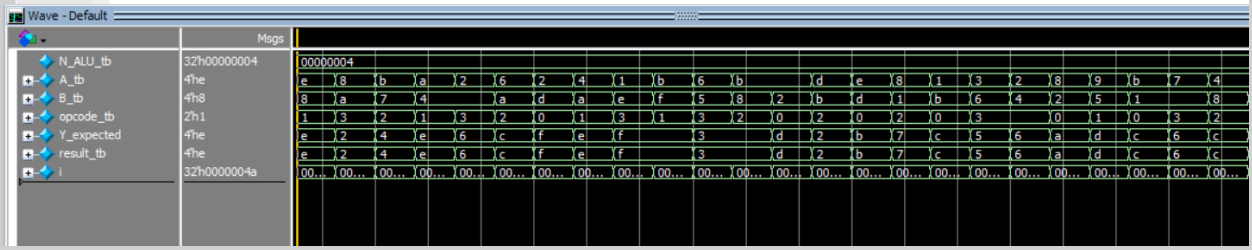
```
# A_tb=1,B_tb=0,Y_expected=1,C_tb=1
# A_tb=0,B_tb=1,Y_expected=1,C_tb=1
# A_tb=1,B_tb=0,Y_expected=1,C_tb=1
# A_tb=1,B_tb=1,Y_expected=0,C_tb=0
# A_tb=0,B_tb=1,Y_expected=1,C_tb=1
# A_tb=1,B_tb=1,Y_expected=0,C_tb=0
# A_tb=0,B_tb=1,Y_expected=1,C_tb=1
# A_tb=1,B_tb=1,Y_expected=0,C_tb=0
# A_tb=0,B_tb=0,Y_expected=0,C_tb=0
# A_tb=1,B_tb=0,Y_expected=1,C_tb=1
# A_tb=0,B_tb=1,Y_expected=1,C_tb=1
# A_tb=0,B_tb=0,Y_expected=0,C_tb=0
# A_tb=1,B_tb=1,Y_expected=0,C_tb=0
# A_tb=0,B_tb=1,Y_expected=1,C_tb=1
# A_tb=1,B_tb=1,Y_expected=0,C_tb=0
# A_tb=1,B_tb=0,Y_expected=1,C_tb=1
# A_tb=0,B_tb=1,Y_expected=1,C_tb=1
# A_tb=1,B_tb=0,Y_expected=1,C_tb=1
# A_tb=0,B_tb=1,Y_expected=1,C_tb=1
# A_tb=1,B_tb=1,Y_expected=0,C_tb=0
# A_tb=0,B_tb=0,Y_expected=0,C_tb=0
# A_tb=1,B_tb=0,Y_expected=1,C_tb=1
# A_tb=0,B_tb=1,Y_expected=1,C_tb=1
# A_tb=1,B_tb=1,Y_expected=0,C_tb=0
# A_tb=0,B_tb=1,Y_expected=1,C_tb=1
# A_tb=1,B_tb=0,Y_expected=1,C_tb=1
# A_tb=0,B_tb=0,Y_expected=0,C_tb=0
# A_tb=1,B_tb=1,Y_expected=0,C_tb=0
# A_tb=1,B_tb=0,Y_expected=1,C_tb=1
# A_tb=0,B_tb=0,Y_expected=0,C_tb=0
# A_tb=1,B_tb=1,Y_expected=0,C_tb=0
# ** Note: $stop    : C:/questasim64_2021.1/examples/N_BIT_ADDER_tb.v(18)
#    Time: 2550 ns  Iteration: 0  Instance: /N_BIT_ADDER_tb
# Break in Module N_BIT_ADDER_tb at C:/questasim64_2021.1/examples/N_BIT_ADDER_tb.v line 18
```

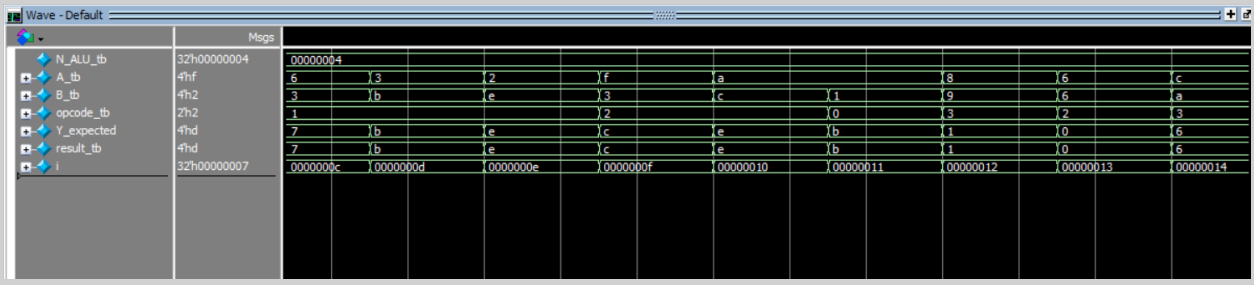
5)

C:/questasim64_2021.1/examples/N_bit_ALU.v (/N_BIT_ALU_tb/DUT) - Default

```
Ln#
 1  module N_BIT_ALU(A,B,opcode,result);
 2  parameter N_ALU=4;
 3  input [N_ALU-1:0] A;
 4  input [N_ALU-1:0] B;
 5  input [1:0] opcode;
 6  output reg [N_ALU-1:0] result;
 7  wire [N_ALU-1:0]ADD_signal;
 8   N_BIT_ADDER #(.N(N_ALU)) ADDER0(.A(A),.B(B),.C(ADD_signal));
 9  always @(*) begin
10  case (opcode)
11  2'b00: result = ADD_signal;
12  2'b01: result = A | B;
13  2'b10: result = A-B;
14  2'b11: result = A^B;
15  endcase
16  end
17  endmodule
18
```

```
Ln#
 1  module N_BIT_ALU_tb();
 2  parameter N_ALU_tb=4;
 3  reg [N_ALU_tb-1:0] A_tb;
 4  reg [N_ALU_tb-1:0] B_tb;
 5  reg [1:0] opcode_tb;
 6  reg [N_ALU_tb-1:0] Y_expected;
 7  wire [N_ALU_tb-1:0] result_tb;
 8  N_BIT_ALU #(.N_ALU(N_ALU_tb)) DUT (A_tb,B_tb,opcode_tb,result_tb);
 9  integer i;
10  initial begin
11  #0;
12  for(i=0 ; i<99; i=i+1) begin
13  A_tb=$random;
14  B_tb=$random;
15  opcode_tb=$random % 4;
16  case (opcode_tb)
17  2'b00: Y_expected=A_tb+B_tb;
18  2'b01: Y_expected= A_tb|B_tb;
19  2'b10: Y_expected=A_tb-B_tb;
20  2'b11: Y_expected=A_tb^B_tb;
21  endcase
22  #5
23  if( result_tb != Y_expected) begin
24  $display("error ,there are some thing not correctl");
25  $stop;
26  end
27  #10;
28  end
29  $stop;
30  end
31  initial begin
32  $monitor ("A_tb=%b,B_tb=%b,opcode_tb=%b,Y_expected=%b,result_tb=%b",A_tb,B_tb,opcode_tb,Y_expected,result_tb);
33  end
34  endmodule
```





```
# A_tb=1100,B_tb=1010,opcode_tb=10,Y_expected=0010,result_tb=0010
# A_tb=1000,B_tb=1001,opcode_tb=01,Y_expected=1001,result_tb=1001
# A_tb=1110,B_tb=0110,opcode_tb=11,Y_expected=1000,result_tb=1000
# A_tb=1010,B_tb=1010,opcode_tb=01,Y_expected=1010,result_tb=1010
# A_tb=1110,B_tb=1000,opcode_tb=01,Y_expected=1110,result_tb=1110
# A_tb=1000,B_tb=1010,opcode_tb=11,Y_expected=0010,result_tb=0010
# A_tb=1011,B_tb=0111,opcode_tb=10,Y_expected=0100,result_tb=0100
# A_tb=1010,B_tb=0100,opcode_tb=01,Y_expected=1110,result_tb=1110
# A_tb=0010,B_tb=0100,opcode_tb=11,Y_expected=0110,result_tb=0110
# A_tb=0110,B_tb=1010,opcode_tb=10,Y_expected=1100,result_tb=1100
# A_tb=0010,B_tb=1101,opcode_tb=00,Y_expected=1111,result_tb=1111
# A_tb=0100,B_tb=1010,opcode_tb=01,Y_expected=1110,result_tb=1110
# A_tb=0001,B_tb=1110,opcode_tb=11,Y_expected=1111,result_tb=1111
# A_tb=1011,B_tb=1111,opcode_tb=01,Y_expected=1111,result_tb=1111
# A_tb=0110,B_tb=0101,opcode_tb=11,Y_expected=0011,result_tb=0011
# A_tb=1011,B_tb=1000,opcode_tb=10,Y_expected=0011,result_tb=0011
# A_tb=1011,B_tb=0010,opcode_tb=00,Y_expected=1101,result_tb=1101
# A_tb=1101,B_tb=1011,opcode_tb=10,Y_expected=0010,result_tb=0010
# A_tb=1110,B_tb=1101,opcode_tb=00,Y_expected=1011,result_tb=1011
# A_tb=1000,B_tb=0001,opcode_tb=10,Y_expected=0111,result_tb=0111
# A_tb=0001,B_tb=1011,opcode_tb=00,Y_expected=1100,result_tb=1100
# A_tb=0011,B_tb=0110,opcode_tb=11,Y_expected=0101,result_tb=0101
# A_tb=0010,B_tb=0100,opcode_tb=11,Y_expected=0110,result_tb=0110
# A_tb=1000,B_tb=0010,opcode_tb=00,Y_expected=1010,result_tb=1010
# A_tb=1001,B_tb=0101,opcode_tb=01,Y_expected=1101,result_tb=1101
# A_tb=1011,B_tb=0001,opcode_tb=00,Y_expected=1100,result_tb=1100
# A_tb=0111,B_tb=0001,opcode_tb=11,Y_expected=0110,result_tb=0110
# A_tb=0100,B_tb=1000,opcode_tb=10,Y_expected=1100,result_tb=1100
# A_tb=0100,B_tb=0010,opcode_tb=10,Y_expected=0010,result_tb=0010
# ** Note: $stop    : C:/questasim64_2021.1/examples/N_BIT_ALU_tb.v(29)
#    Time: 1485 ns  Iteration: 0  Instance: /N_BIT_ALU_tb
# Break in Module N_BIT_ALU_tb at C:/questasim64_2021.1/examples/N_BIT_ALU_tb.v line 29
```

6)

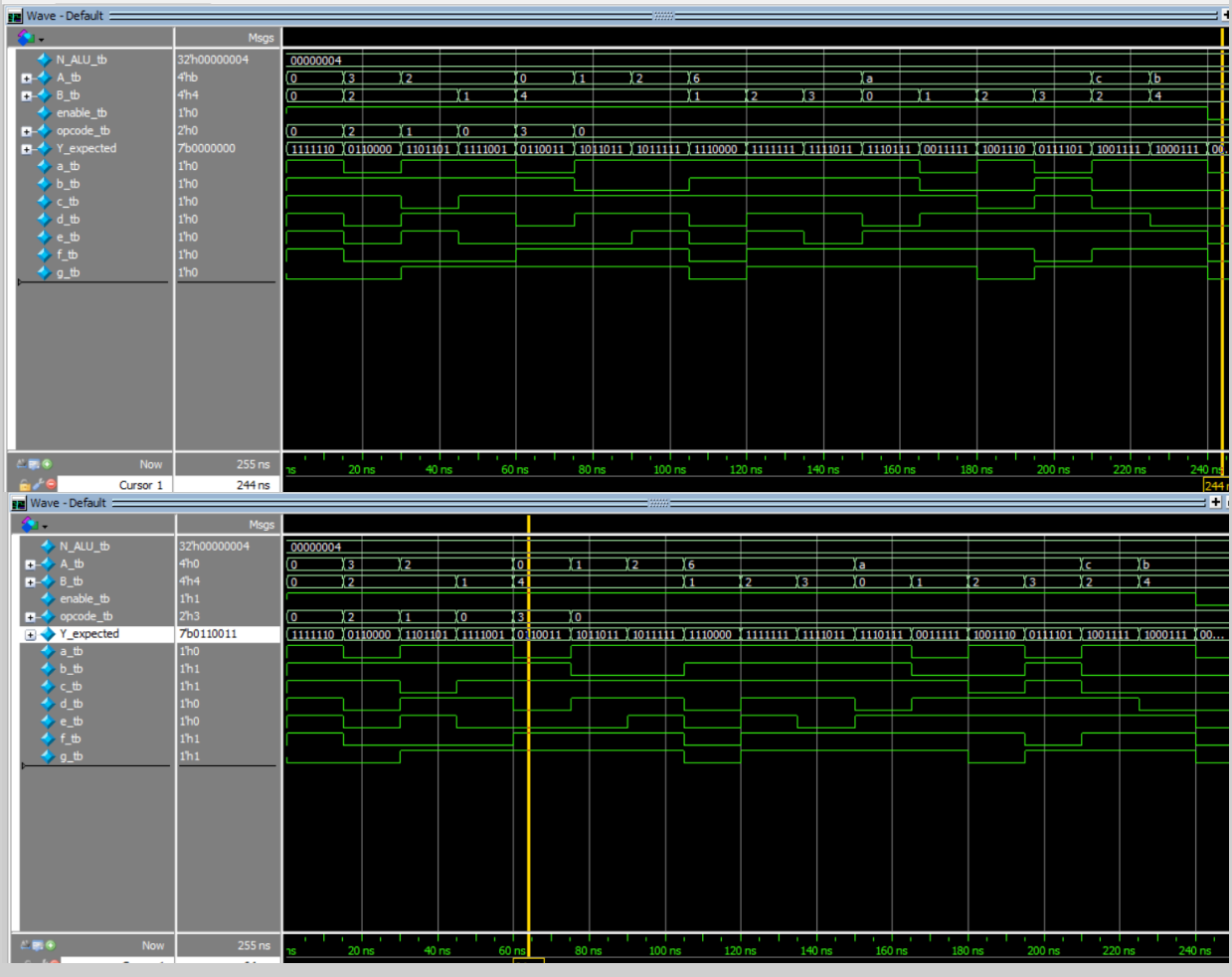
```
Ln#
 1  module SEVEN_SEGMENT(A,B,opcode,enable,a,b,c,d,e,f,g);
 2  parameter N_ALU=4;
 3  input [N_ALU-1:0] A;
 4  input [N_ALU-1:0] B;
 5  input [1:0] opcode;
 6  input enable;
 7  output reg a,b,c,d,e,f,g;
 8  wire [N_ALU-1:0]ADD_signal;
 9   N_BIT_ALU #(.N_ALU(N_ALU)) ALU1(.A(A),.B(B),.opcode(opcode),.result(ADD_signal));
10  always @(*) begin
11  if(enable)
12          case(ADD_signal)
13                  0: {a,b,c,d,e,f,g}=7'b1111110;
14                  1: {a,b,c,d,e,f,g}=7'b0110000;
15                  2: {a,b,c,d,e,f,g}=7'b1101101;
16                  3: {a,b,c,d,e,f,g}=7'b1111001;//
17                  4: {a,b,c,d,e,f,g}=7'b0110011;//
18                  5: {a,b,c,d,e,f,g}=7'b1011011;//
19                  6: {a,b,c,d,e,f,g}=7'b1011111;//
20                  7: {a,b,c,d,e,f,g}=7'b1110000;//
21                  8: {a,b,c,d,e,f,g}=7'b1111111;//
22                  9: {a,b,c,d,e,f,g}=7'b1111011;//
23                  4'hA: {a,b,c,d,e,f,g}=7'b1110111;
24                  4'hB: {a,b,c,d,e,f,g}=7'b0011111;
25                  4'hC: {a,b,c,d,e,f,g}=7'b1001110;//
26                  4'hD: {a,b,c,d,e,f,g}=7'b0111101;//
27                  4'hE: {a,b,c,d,e,f,g}=7'b1001111;
28                  4'hF: {a,b,c,d,e,f,g}=7'b1000111;
29                  default : {a,b,c,d,e,f,g}=7'b0000000;
30          endcase
31  else
32  {a,b,c,d,e,f,g}=7'b0000000;
33  end
34  endmodule
35
```

```
1  module SEVEN_SEGMENT_tb();
2  parameter N_ALU_tb=4;
3  reg [N_ALU_tb-1:0] A_tb;
4  reg [N_ALU_tb-1:0] B_tb;
5  reg enable_tb;
6  reg [1:0] opcode_tb;
7  reg [6:0] Y_expected;
8  wire a_tb,b_tb,c_tb,d_tb,e_tb,f_tb,g_tb;
9  //wire [N_ALU_tb-1:0] result_tb;
10 SEVEN_SEGMENT #(.N_ALU(N_ALU_tb)) DUT (.A(A_tb),
11     .B(B_tb),
12     .opcode(opcode_tb),
13     .enable(enable_tb),
14     .a(a_tb),
15     .b(b_tb),
16     .c(c_tb),
17     .d(d_tb),
18     .e(e_tb),
19     .f(f_tb),
20     .g(g_tb));
21 initial begin
22 #0 A_tb=0;B_tb=0;enable_tb=1;opcode_tb=0;Y_expected=7'b1111110;
23 #5;
24 if( {a_tb,b_tb,c_tb,d_tb,e_tb,f_tb,g_tb} != Y_expected) begin
25 $display("error ,there are some thing not correct1");
26 $stop;
27 end
```

```
28      #10;
29       A_tb=3;B_tb=2;enable_tb=1;opcode_tb=2;Y_expected=7'b0110000;
30      #5;
31      if( {a_tb,b_tb,c_tb,d_tb,e_tb,f_tb,g_tb} != Y_expected) begin
32      $display("error ,there are some thing not correctl");
33      $stop;
34      end
35      #10;
36       A_tb=2;B_tb=2;enable_tb=1;opcode_tb=1;Y_expected=7'b1101101;
37      #5;
38      if( {a_tb,b_tb,c_tb,d_tb,e_tb,f_tb,g_tb} != Y_expected) begin
39      $display("error ,there are some thing not correctl");
40      $stop;
41      end
42      #10;
43       A_tb=2;B_tb=1;opcode_tb=0;Y_expected=7'b1111001;
44      #5;
45      if( {a_tb,b_tb,c_tb,d_tb,e_tb,f_tb,g_tb} != Y_expected) begin
46      $display("error ,there are some thing not correctl");
47      $stop;
48      end
49      #10;
50       A_tb=0;B_tb=4;opcode_tb=11;Y_expected=7'b0110011;
51      #5;
52      if( {a_tb,b_tb,c_tb,d_tb,e_tb,f_tb,g_tb} != Y_expected) begin
53      $display("error ,there are some thing not correctl");
54      $stop;
55      end
56      #10;
57       A_tb=1;B_tb=4;opcode_tb=00;Y_expected=7'b1011011;
58      #5;
59      if( {a_tb,b_tb,c_tb,d_tb,e_tb,f_tb,g_tb} != Y_expected) begin
60      $display("error ,there are some thing not correctl");
61      $stop;
62      end
63      #10;
64       A_tb=2;B_tb=4;Y_expected=7'b1011111;
65      #5;
66      if( {a_tb,b_tb,c_tb,d_tb,e_tb,f_tb,g_tb} != Y_expected) begin
67      $display("error ,there are some thing not correctl");
68      $stop;
69      end
```

```
70    #10;
71     A_tb=6;B_tb=1;Y_expected=7'b1110000;
72    #5;
73    if( {a_tb,b_tb,c_tb,d_tb,e_tb,f_tb,g_tb} != Y_expected) begin
74    $display("error ,there are some thing not correctl");
75    $stop;
76    end
77    #10;
78     A_tb=6;B_tb=2;Y_expected=7'b1111111;
79    #5;
80    if( {a_tb,b_tb,c_tb,d_tb,e_tb,f_tb,g_tb} != Y_expected) begin
81    $display("error ,there are some thing not correctl");
82    $stop;
83    end
84    #10;
85     A_tb=6;B_tb=3;Y_expected=7'b1111011;
86    #5;
87    if( {a_tb,b_tb,c_tb,d_tb,e_tb,f_tb,g_tb} != Y_expected) begin
88    $display("error ,there are some thing not correctl");
89    $stop;
90    end
91    #10;
92     A_tb=4'hA;B_tb=0;Y_expected=7'b1110111;
93    #5;
94    if( {a_tb,b_tb,c_tb,d_tb,e_tb,f_tb,g_tb} != Y_expected) begin
95    $display("error ,there are some thing not correctl");
96    $stop;
97    end
98    #10;
99     A_tb=4'hA;B_tb=1;Y_expected=7'b0011111;
100   #5;
101   if( {a_tb,b_tb,c_tb,d_tb,e_tb,f_tb,g_tb} != Y_expected) begin
102   $display("error ,there are some thing not correctl");
103   $stop;
104   end
105   #10;
106    A_tb=4'hA;B_tb=2;Y_expected=7'b1001110;
107   #5;
108   if( {a_tb,b_tb,c_tb,d_tb,e_tb,f_tb,g_tb} != Y_expected) begin
109   $display("error ,there are some thing not correctl");
110   $stop;
111   end
```

```
112     #10;
113      A_tb=4'hA;B_tb=3;Y_expected=7'b0111101;
114     #5;
115     if( {a_tb,b_tb,c_tb,d_tb,e_tb,f_tb,g_tb} != Y_expected) begin
116     $display("error ,there are some thing not correct1");
117     $stop;
118     end
119     #10;
120      A_tb=4'hC;B_tb=2;Y_expected=7'b1001111;
121     #5;
122     if( {a_tb,b_tb,c_tb,d_tb,e_tb,f_tb,g_tb} != Y_expected) begin
123     $display("error ,there are some thing not correct1");
124     $stop;
125     end
126     #10;
127      A_tb=4'hB;B_tb=4;Y_expected=7'b1000111;
128     #5;
129     if( {a_tb,b_tb,c_tb,d_tb,e_tb,f_tb,g_tb} != Y_expected) begin
130     $display("error ,there are some thing not correct1");
131     $stop;
132     end
133     #10;
134     enable_tb=0;Y_expected=7'b0000000;
135     #5;
136     if( {a_tb,b_tb,c_tb,d_tb,e_tb,f_tb,g_tb} != Y_expected) begin
137     $display("error ,there are some thing not correct1");
138     $stop;
139     end
140     #10;
141     $stop;
142     end
143     initial begin
144         $monitor ("A_tb=%b, B_tb=%b, opcode_tb=%b, enable_tb=%b, Y_expected=%b, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb}=%b
145     A_tb, B_tb, opcode_tb, enable_tb, Y_expected, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb});
146     end
147     endmodule
```



```
VSIM 11> run -all
# A_tb=0000, B_tb=0000, opcode_tb=00, enable_tb=1, Y_expected=1111110, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb}=1111110
# A_tb=0011, B_tb=0010, opcode_tb=10, enable_tb=1, Y_expected=0110000, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb}=0110000
# A_tb=0010, B_tb=0010, opcode_tb=01, enable_tb=1, Y_expected=1101101, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb}=1101101
# A_tb=0010, B_tb=0001, opcode_tb=00, enable_tb=1, Y_expected=1111001, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb}=1111001
# A_tb=0000, B_tb=0100, opcode_tb=11, enable_tb=1, Y_expected=0110011, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb}=0110011
# A_tb=0001, B_tb=0100, opcode_tb=00, enable_tb=1, Y_expected=1011011, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb}=1011011
# A_tb=0010, B_tb=0100, opcode_tb=00, enable_tb=1, Y_expected=1011111, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb}=1011111
# A_tb=0110, B_tb=0001, opcode_tb=00, enable_tb=1, Y_expected=1110000, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb}=1110000
# A_tb=0110, B_tb=0010, opcode_tb=00, enable_tb=1, Y_expected=1111111, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb}=1111111
# A_tb=0110, B_tb=0011, opcode_tb=00, enable_tb=1, Y_expected=1111011, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb}=1111011
# A_tb=1010, B_tb=0000, opcode_tb=00, enable_tb=1, Y_expected=1110111, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb}=1110111
# A_tb=1010, B_tb=0001, opcode_tb=00, enable_tb=1, Y_expected=0011111, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb}=0011111
# A_tb=1010, B_tb=0010, opcode_tb=00, enable_tb=1, Y_expected=1001110, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb}=1001110
# A_tb=1010, B_tb=0011, opcode_tb=00, enable_tb=1, Y_expected=0111101, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb}=0111101
# A_tb=1100, B_tb=0010, opcode_tb=00, enable_tb=1, Y_expected=1001111, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb}=1001111
# A_tb=1011, B_tb=0100, opcode_tb=00, enable_tb=1, Y_expected=1000111, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb}=1000111
# A_tb=1011, B_tb=0100, opcode_tb=00, enable_tb=0, Y_expected=0000000, {a_tb, b_tb, c_tb, d_tb, e_tb, f_tb, g_tb}=0000000
# ** Note: $stop    : C:/questasim64_2021.1/examples/SEVEN_SEGMNET_tb.v(141)
#    Time: 255 ns  Iteration: 0  Instance: /SEVEN_SEGMENT_tb
# Break in Module SEVEN_SEGMENT_tb at C:/questasim64_2021.1/examples/SEVEN_SEGMNET_tb.v line 141
```