not gate:

```
module not_gate(output out,input in);
not(out,in);
endmodule

module not_gate_tb;
reg in;
wire out;
not_gate obj (out,in);
initial begin
in=0;
#1 in=1;
end

initial begin
$monitor("T=%t in=%b out=%b",$time,in,out);
end
endmodule
```

---

and gate:

```
module and_gate(output out,input in1,input in2);
and(out,in1,in2);
endmodule

module and_gate_tb;
reg in1,in2;
wire out;
and_gate obj (out,in1,in2);
initial begin
in1=0; in2=0;
#1 in1=0; in2=1;
#1 in1=1; in2=0;
#1 in1=1; in2=1;
end

initial begin
$monitor("T=%t in1=%b in2=%b out=%b",$time,in1,in2,out);
end
endmodule
```

---

or gate:

```
module or_gate(output out,input in1,input in2);
or(out,in1,in2);
endmodule

module or_gate_tb;
reg in1,in2;
wire out;
or_gate obj (out,in1,in2);
initial begin
in1=0; in2=0;
#1 in1=0; in2=1;
#1 in1=1; in2=0;
#1 in1=1; in2=1;
end
```

```
initial begin
$monitor("T=%t in1=%b in2=%b out=%b",$time,in1,in2,out);
end
endmodule
----------------------------------------------------------------
nand gate:
module nand_gate(output out,input in1,input in2);
nand(out,in1,in2);
endmodule

module nand_gate_tb;
reg in1,in2;
wire out;
nand_gate obj (out,in1,in2);
initial begin
in1=0; in2=0;
#1 in1=0; in2=1;
#1 in1=1; in2=0;
#1 in1=1; in2=1;
end

initial begin
$monitor("T=%t in1=%b in2=%b out=%b",$time,in1,in2,out);
end
endmodule
----------------------------------------------------------------
nor gate:
module nor_gate(output out,input in1,input in2);
nor(out,in1,in2);
endmodule

module nor_gate_tb;
reg in1,in2;
wire out;
nor_gate obj (out,in1,in2);
initial begin
in1=0; in2=0;
#1 in1=0; in2=1;
#1 in1=1; in2=0;
#1 in1=1; in2=1;
end

initial begin
$monitor("T=%t in1=%b in2=%b out=%b",$time,in1,in2,out);
end
endmodule
----------------------------------------------------------------
module xor_gate(output out,input in1,input in2);
xor(out,in1,in2);
endmodule

module xor_gate_tb;
reg in1,in2;
wire out;
```

```
xor_gate obj (out,in1,in2);
initial begin
in1=0; in2=0;
#1 in1=0; in2=1;
#1 in1=1; in2=0;
#1 in1=1; in2=1;
end

initial begin
$monitor("T=%t in1=%b in2=%b out=%b",$time,in1,in2,out);
end
endmodule
```

---

xnor:

```
module xnor_gate(output out,input in1,input in2);
xnor(out,in1,in2);
endmodule

module xnor_gate_tb;
reg in1,in2;
wire out;
xnor_gate obj (out,in1,in2);
initial begin
in1=0; in2=0;
#1 in1=0; in2=1;
#1 in1=1; in2=0;
#1 in1=1; in2=1;
end

initial begin
$monitor("T=%t in1=%b in2=%b out=%b",$time,in1,in2,out);
end
endmodule
```

---

half adder:

```
module half_add(output sum1,output carry,input in1,input in2);
assign sum1=in1^in2;
assign carry=in1&in2;
endmodule

module half_add_tb;
reg in1,in2;
wire sum1,carry;
half_add obj (sum1,carry,in1,in2);
initial begin
in1=0; in2=0;
#1 in1=0; in2=1;
#1 in1=1; in2=0;
#1 in1=1; in2=1;
end

initial begin
$monitor("T=%t in1=%b in2=%b sum=%b carry=%b",$time,in1,in2,sum1,carry);
end
endmodule
```

```
--------------------------------------------------------------------------
full adder:
module full_add(output sum1,output carry_out,input in1,input in2,input carry_in);
assign sum1=carry_in^(in1^in2);
assign carry_out=in1&in2|carry_in&(in1^in2);
endmodule

module full_add_tb;
reg in1,in2,carry_in;
wire sum1,carry_out;
full_add obj (sum1,carry_out,in1,in2,carry_in);
initial begin
in1=0; in2=0; carry_in=0;
#1 in1=0; in2=0; carry_in=1;
#1 in1=0; in2=1; carry_in=0;
#1 in1=0; in2=1; carry_in=1;
#1 in1=1; in2=0; carry_in=0;
#1 in1=1; in2=0; carry_in=1;
#1 in1=1; in2=1; carry_in=0;
#1 in1=1; in2=1; carry_in=1;
end

initial begin
$monitor("T=%t in1=%b in2=%b carry_in=%b sum=%b carry=%b",$time,in1,in2,carry_in,sum1,carry_out);
end
endmodule
-----------------------------------------------------------------------------
half sub:
module half_sub(output diff,output borrow,input in1,input in2);
assign diff=(~in1&in2)|(in1&~in2);
assign borrow=~in1&in2;
endmodule

module half_sub_tb;
reg in1,in2;
wire diff,borrow;
half_sub obj (diff,borrow,in1,in2);
initial begin
in1=0; in2=0;
#1 in1=0; in2=1;
#1 in1=1; in2=0;
#1 in1=1; in2=1;
end

initial begin
$monitor("T=%t in1=%b in2=%b diff=%b borrow=%b",$time,in1,in2,diff,borrow);
end
endmodule
-----------------------------------------------------------------------------
full sub:
module full_sub(output diff,output borrow_out,input in1,input in2,input borrow_in);
assign diff=(in1^in2)^borrow_in;
assign borrow_out=(in1&~in2)&(in1&~borrow_in)&(in2&borrow_in);
endmodule
```

```
module full_sub_tb;
reg in1,in2,borrow_in;
wire diff,borrow_out;
full_sub obj (diff,borrow_out,in1,in2,borrow_in);
initial begin
in1=0; in2=0; borrow_in=0;
#1 in1=0; in2=0; borrow_in=1;
#1 in1=0; in2=1; borrow_in=0;
#1 in1=0; in2=1; borrow_in=1;
#1 in1=1; in2=0; borrow_in=0;
#1 in1=1; in2=0; borrow_in=1;
#1 in1=1; in2=1; borrow_in=0;
#1 in1=1; in2=1; borrow_in=1;
end

initial begin
$monitor("T=%t in1=%b in2=%b borrow_in=%b diff=%b borrow_out=%b",$time,in1,in2,borrow_in,diff,borrow_o
ut);
end
endmodule
```

BCD-EXCESS 3

```
module BCD2Ex3(a,b,c,d,w,x,y,z);
        input a,b,c,d;
        output w,x,y,z;
         assign w= a |(b&c) | (b&d);
         assign x=(~b&c) | (~b&d) |(b&~c&~d);
         assign y= ~(c^d);
         assign z=~d;
  endmodule
module test;
     reg a,b,c,d;
     wire w, x, y, z;
     BCD2Ex3 objt(a,b,c,d,w,x,y,z);
     initial
     begin
       $display("a\tb\tc\td\tw\tx\ty\tz");
       $monitor("%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b",a,b,c,d,w,x,y,z);
      a=0; b=0; c=0;d=0;
      #10 a=0; b=0; c=0;d=1;
      #10 a=0; b=0; c=1;d=0;
      #10 a=0; b=0; c=1;d=1;
      #10 a=0; b=1; c=0;d=0;
      #10 a=0; b=1; c=0;d=1;
      #10 a=0; b=1; c=1;d=0;
      #10 a=0; b=1; c=1;d=1;
      #10 a=1; b=0; c=0;d=0;
      #10 a=1; b=0; c=0;d=1;
       end
endmodule
```

Excess3 to BCD

```
module Ex3toBCD(a,b,c,d,w,x,y,z);
        input a,b,c,d;
        output w,x,y,z;
```

```
        assign w= (a&b)|(a&c&d);
        assign x=(~b&~c)|(~b&~d)|(b&c&d);
        assign y= (~c&d)|(c&~d);
        assign z=~d;
  endmodule
module test;
     reg a,b,c,d;
     wire w, x, y, z;
     Ex3toBCD obj(a,b,c,d,w,x,y,z);
     initial
     begin
       $display("a\tb\tc\td\tw\tx\ty\tz");
       $display("0\t0\t0\t0\tx\tx\tx\tx");
       $display("0\t0\t0\t1\tx\tx\tx\tx");
       $display("0\t0\t1\t0\tx\tx\tx\tx");
       $monitor("%b\t%b\t%b\t%b\t%b\t%b\t%b\t%b",a,b,c,d,w,x,y,z);

      a=0; b=0; c=1;d=1;
      #10 a=0; b=1; c=0;d=0;
      #10 a=0; b=1; c=0;d=1;
      #10 a=0; b=1; c=1;d=0;
      #10 a=0; b=1; c=1;d=1;
      #10 a=1; b=0; c=0;d=0;
      #10 a=1; b=0; c=0;d=1;
       end
endmodule
--------------------------------------------------------------------------------
4x1 mux
module mux1( select, d, q );

input[1:0] select;
input[3:0] d;
output     q;

wire      q;
wire[1:0] select;
wire[3:0] d;

assign q = d[select];

endmodule

module mux_tb;

reg[3:0] d;
reg[1:0] select;
wire     q;

integer  i;

mux1 my_mux( select, d, q );

initial
begin
  #1 $monitor("d = %b", d, "  |  select = ", select, "  |  q = ", q );
```

```
   for( i = 0; i <= 15; i = i + 1)
   begin
     d = i;
     select = 0;  #1;
     select = 1;  #1;
     select = 2;  #1;
     select = 3;  #1;
     $display("----------------------------------------");
    end

end
endmodule
--------------------------------------------------------------------------------
1x4 demux
module Demultiplexer_1_to_4_case (output reg [3:0] Y, input [1:0] A, input din);
always @(Y, A) begin
    case (A)
       2'b00 : begin Y[0] = din; Y[3:1] = 0; end
       2'b01 : begin Y[1] = din; Y[0] = 0;   end
       2'b10 : begin Y[2] = din; Y[1:0] = 0; end
       2'b11 : begin Y[3] = din; Y[2:0] = 0; end
    endcase
end
endmodule

module Demultiplexer_1_to_4_case_tb;
wire [3:0] Y;
reg [1:0] A;
reg din;
Demultiplexer_1_to_4_case I0 (Y, A, din);
initial begin
    din = 1;
    A = 2'b00;
#1  A = 2'b01;
#1  A = 2'b10;
#1  A = 2'b11;
end
initial begin
    $monitor("%t| Din = %d| A[1] = %d| A[0] = %d| Y[0] = %d| Y[1] = %d| Y[2] = %d| Y[3] = %d",
             $time, din, A[1], A[0], Y[0], Y[1], Y[2], Y[3]);
end
endmodule
--------------------------------------------------------------------------------
3x8 decoder
module decoder3_to_8( in,out, en);
input [2:0]  in;
input en;
output [7:0] out;
  reg [7:0] out;

 always @( in or en)
 begin

     if (en)
```

```
        begin
          out=8'd0;
          case (in)
             3'b000: out[0]=1'b1;
             3'b001: out[1]=1'b1;
             3'b010: out[2]=1'b1;
             3'b011: out[3]=1'b1;
             3'b100: out[4]=1'b1;
             3'b101: out[5]=1'b1;
             3'b110: out[6]=1'b1;
             3'b111: out[7]=1'b1;
             default: out=8'd0;
          endcase
      end
else
out=8'd0;
end
endmodule

module decoder_tb;
wire [7:0] out;
reg en;
reg [2:0] in;
integer i;

decoder3_to_8 dut(in,out,en);

initial begin
 $monitor( "en=%b, in=%d, out=%b ", en, in, out);
   for ( i=0; i<16; i=i+1)
        begin
           {en,in}  = i;
            #1;
        end
end
endmodule
--------------------------------------------------------------------------------
8x3 encoder
module encoder8x3(in,out);
input  [7:0] in;
output [2:0] out;
reg    [2:0] out;
reg    [2:0] i;
always @(in)
begin
  for(i=0;i<8;i=i+1)
   if(in[i])
     out=i;
end

endmodule
--------------------------------------------------------------------------------
2x4 decoder
module Decoder(s1, s0, o0, o1, o2, o3);
input s1, s0;
```

```
output o0, o1, o2, o3;
wire s1_inv, s0_inv;

not(s1_inv, s1);
not(s0_inv, s0);
and(o0, s1_inv, s0_inv);
and(o1, s1_inv, s0);
and(o2, s1, s0_inv);
and(o3, s1, s0);
endmodule

module TestMod;

reg s1, s0;
wire o0, o1, o2, o3;

Decoder dec(s1, s0, o0, o1, o2, o3);

initial begin
$display("Time \t s1 \t s0 \t o0 \t o1 \t o2 \t o3");
$display("-----------------------------------------------");
$monitor("%0d \t %b \t %b \t %b \t %b \t %b \t %b", $time, s1, s0, o0, o1, o2, o3);
end

initial #3 $finish;

initial begin
s1= 0; s0= 0;
#1;
s1= 0; s0= 1;
#1;
s1= 1; s0= 0;
#1;
s1= 1; s0= 1;
end
endmodule
```

4x2 encoder

```
`timescale 1ns / 1ps
module encoder42(
    input D0,
    input D1,
    input D2,
    input D3,
    output Q0,
    output Q1,
  output V
    );
wire a1, nD1;

not(nD1, D1);
and(a1, nD1, D2);

or(Q0, D0, D1);
or(Q1, a1, D0);
```

```
or(V, D0, D1, D2, D3);


endmodule
`timescale 1ns / 1ps

module encoder42_tb;

 // Inputs
 reg D0;
 reg D1;
 reg D2;
 reg D3;

 // Outputs
 wire Q0;
 wire Q1;
 wire V;

 // Instantiate the Unit Under Test (UUT)
 encoder42 uut (
  .D0(D0),
  .D1(D1),
  .D2(D2),
  .D3(D3),
  .Q0(Q0),
  .Q1(Q1),
  .V(V)
 );

 initial begin
  // Initialize Inputs
  D0 = 1;
  D1 = 0;
  D2 = 0;
  D3 = 0;

  // Wait 100 ns for global reset to finish
  #100;
  D0 = 0;
  D1 = 1;
  D2 = 0;
  D3 = 0;

  #100;
  D0 = 0;
  D1 = 0;
  D2 = 1;
  D3 = 0;

  #100;
  D0 = 0;
  D1 = 0;
  D2 = 0;
  D3 = 1;
```

```
  // Add stimulus here
 end
endmodule
```
