**DECODER**

module decoder(a,b,en,y0,y1,y2,y3);

input a,b,en;

output y0,y1,y2,y3;

assign y0=(~a&~b&en);

assign y1=(~a&b&en);

assign y2=(a&~b&en);

assign y3=(a&b&en);

endmodule

**DECODER TB:**

module decoder\_tb();

reg a,b,en;

wire y0,y1,y2,y3;

decoder na(a,b,en,y0,y1,y2,y3);

initial

begin

en=1'b1;a=1'b0;b=1'b0;

end

always

begin

#10 en=1'b1;a=1'b0;b=1'b1;

#10 en=1'b1;a=1'b1;b=1'b0;

#10 en=1'b1;a=1'b1;b=1'b1;

end

initial #50 $stop;

endmodule

**ENCODER:**

module encoder(y0,y1,y2,y3,a,b,en);

input en,y0,y1,y2,y3;

output a,b;

assign a=y2+y3;

assign b=y1+y3;

endmodule

**ENCODER TB:**

module encoder\_tb();

reg en,y0,y1,y2,y3;

wire a,b;

encoder na(y0,y1,y2,y3,a,b,en);

initial

begin

en=1'b1;y0=1'b1;y1=1'b0;y2=1'b0;y3=1'b0;

#10 en=1'b1;y0=1'b0;y1=1'b1;y2=1'b0;y3=1'b0;

#10 en=1'b1;y0=1'b0;y1=1'b0;y2=1'b1;y3=1'b0;

#10 en=1'b1;y0=1'b0;y1=1'b0;y2=1'b0;y3=1'b1;

end

initial #50 $stop;

endmodule

**DEMUX:**

module demux(y,a,din);

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

**DEMUX TB:**

module demux\_tb();

wire[3:0]y;

reg[1:0]a;

reg din;

demux na(y,a,din);

initial

begin

din=1;a=2'b00;

#1 a=2'b01;

#1 a=2'b10;

#1 a=2'b11;

end

initial #10 $stop;

endmodule

**MUX:**

module mux\_data(a,b,c,d,s1,s0,y);

input a,b,c,d,s1,s0;

output y;

assign y = s0?(s1?d:b):(s1?c:a);

endmodule

**MUX TB:**

module mux\_tb();

reg a,b,c,d,s0,s1;

wire y;

mux\_data na(a,b,c,d,s1,s0,y);

initial

begin

a = 1'b0; b=1'b1; c=1'b1; d=1'b0; s0=1'b0; s1=1'b0;

end

always

begin

#10 a=1'b0;b=1'b1;c=1'b1; d=1'b0; s0=1'b0; s1=1'b1;

#10 a = 1'b0;b=1'b1;c=1'b1; d=1'b0; s0=1'b1; s1=1'b0;

#10 a = 1'b0;b=1'b1;c=1'b1; d=1'b0; s0=1'b1; s1=1'b1;

#10 a = 1'b0;b=1'b1;c=1'b1; d=1'b0; s0=1'b0; s1=1'b0;

end

initial #100 $stop;

endmodule

**LOGIC GATES:**

module logic(a,b,y0,y1,y2,y3,y4,y5,y6);

input a,b;

output y0,y1,y2,y3,y4,y5,y6;

assign y0=(a&b);

assign y1=(a/b);

assign y2=(a^b);

assign y3=~a;

assign y4=~(a&b);

assign y5=~(a/b);

assign y6=~(a^b);

endmodule

**LOGIC GATES TB:**

module logic\_tb();

reg a,b;

wire y0,y1,y2,y3,y4,y5,y6;

logic aa(a,b,y0,y1,y2,y3,y4,y5,y6);

initial

begin

a=1'b0; b=1'b0;

end

always

begin

#10 a=1'b0; b=1'b1;

#10 a=1'b1; b=1'b0;

#10 a=1'b1; b=1'b1;

end

initial #50 $stop;

endmodule

**HALF ADDER:**

module halfadder(a,b,sum,carry);

input a,b;

output sum,carry;

assign sum=a^b;

assign carry=a&b;

endmodule

**HALF ADDER TB:** module halfadder\_tb();

reg a,b;

wire sum,carry;

halfadder usc(a,b,sum,carry);

initial

begin

a=1'b0;b=1'b0;

#10 a=1'b0;b=1'b1;

#10 a=1'b1;b=1'b0;

#10 a=1'b1;b=1'b1;

end

initial #50 $stop;

endmodule

**FULL ADDER:**  
module fulladd(sum,carry,a,b,c);

input a,b,c;

output sum,carry;

assign sum=a^b^c;

assign carry=(a&b)+(b&c)+(c&a);

endmodule

**FULL ADDER TB:**  
 module fulladd\_tb();

reg a,b,c;

wire sum,carry;

fulladd usm(a,b,c,sum,carry);

initial

begin

a=0;b=0;c=0;

end

always

begin

#10 a=0;b=0;c=1;

#10 a=0;b=1;c=1;

#10 a=1;b=1;c=1;

end

initial #100 $stop;

endmodule