1.

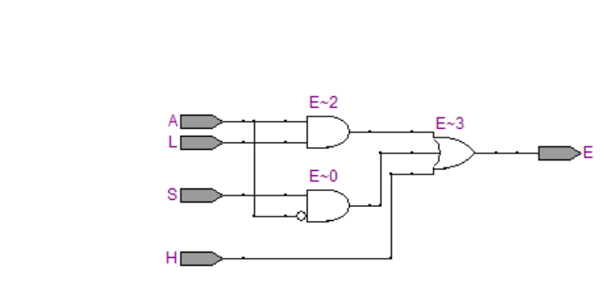
module ben(S,A,H,L,E);

input S,A,H,L;

output E;

assign E = (S&(~A))|H|(A&L) ;

endmodule



2.

module aa(A,D,P);

input [3:0]A;

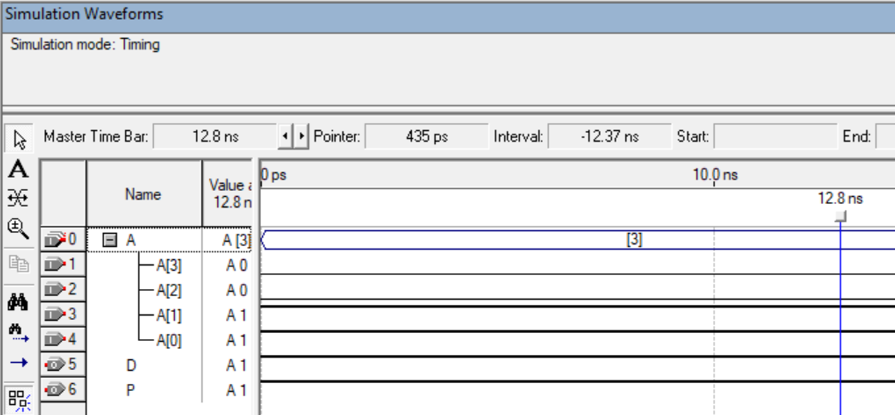
output D,P;

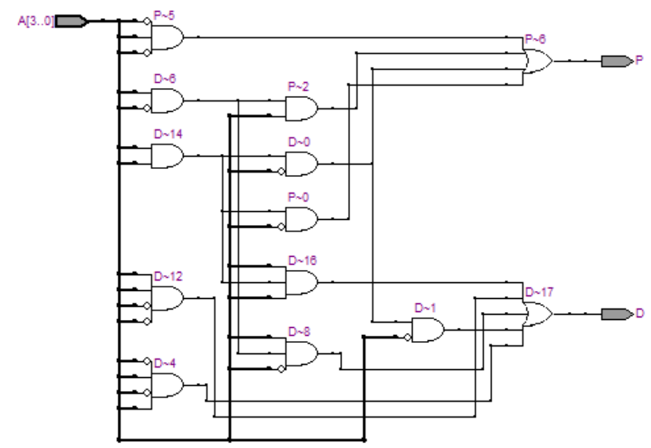
//AFTER SIMPLIFICATION EQUATIONS USING K-MAP

assign D = A[0]&A[1]&(~A[2])&(~A[3]) | (~A[0])&A[1]&A[2]&(~A[3]) | A[0]&(~A[1])&(~A[2])&A[3] | (~A[0])&(~A[1])&A[2]&A[3] | A[0]&A[1]&A[2]&A[3];

assign P = (A[0])&A[1]&(~A[3]) | (A[0])&(A[1])&(~A[2]) | (A[0])&(~A[1])&A[2] | (A[1])&(~A[2])&(~A[3]);

endmodule





3.

module aa(A,Y,NONE);

input [7:0]A;

output [2:0]Y;

output NONE;

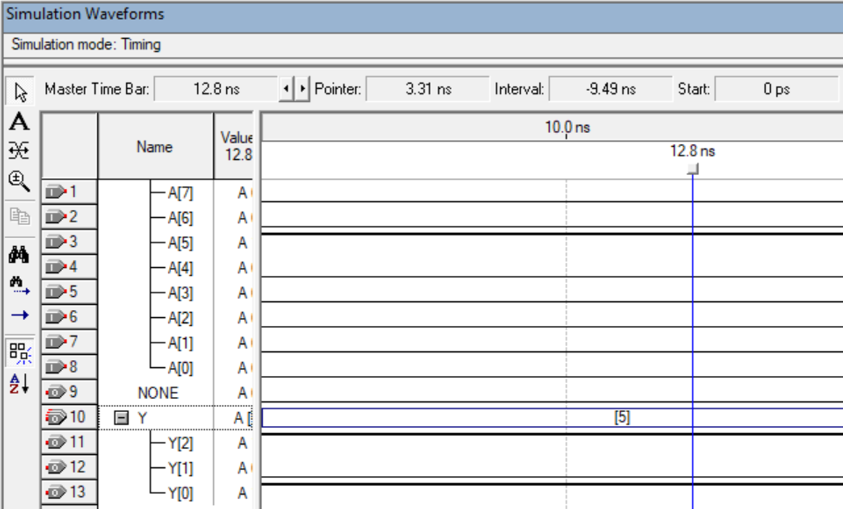
assign Y[2] = A[7]|A[6]|A[5]|A[4];

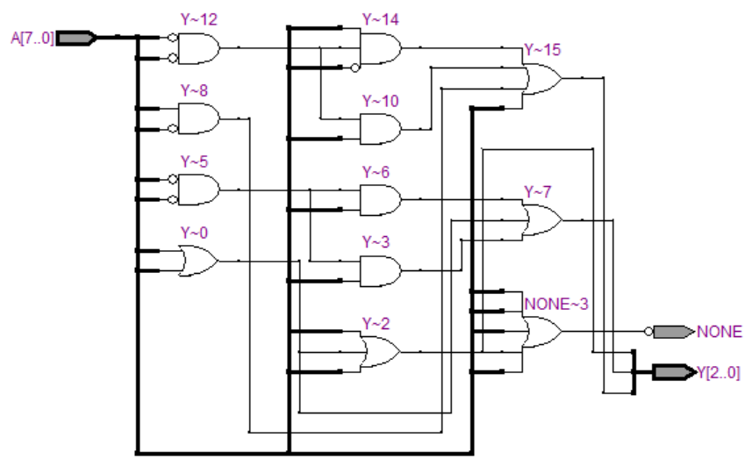
assign Y[1] = A[7]|A[6]|(~A[5]&~A[4]&A[3])|(~A[5]&~A[4]&A[2]);

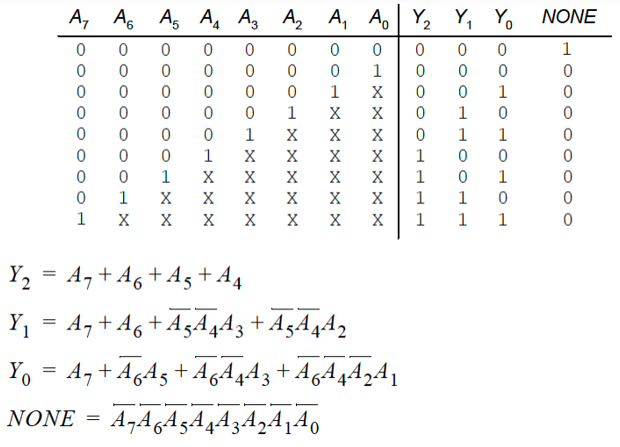
assign Y[0] = A[7]|(~A[6]&A[5])|(~A[6]&~A[4]&A[3])|(~A[6]&~A[4]&~A[2]&A[1]);

assign NONE = ~(A[7]|A[6]|A[5]|A[4]|A[3]|A[2]|A[1]|A[0]);

endmodule







6.

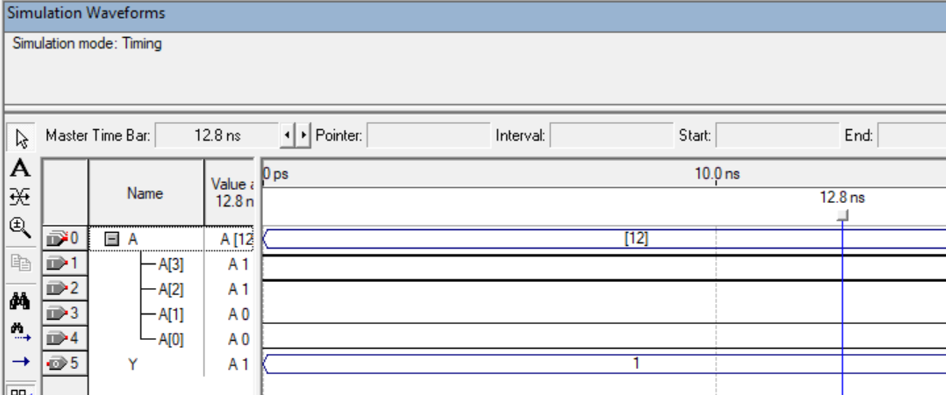
module aa(A,Y);

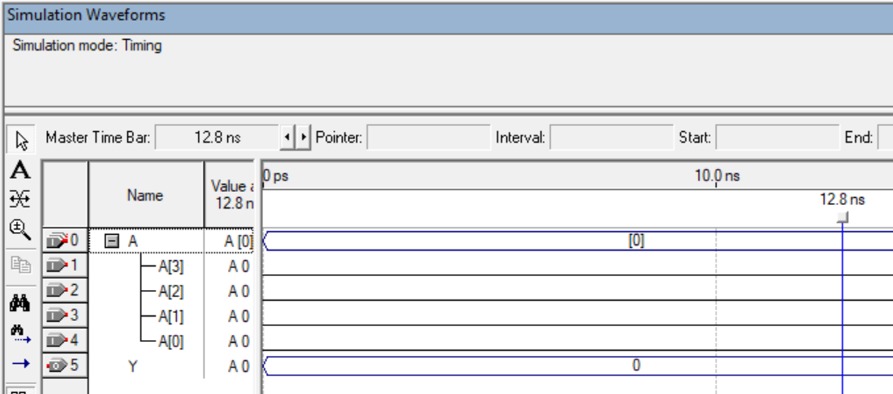
input [3:0]A;

output Y;

assign Y= A[3]^A[0];

endmodule





7.

module aa(s,clk,out);

input s,clk;

output out;

reg r1,r2,r3;

always @ (posedge clk)

begin

r1 <= s;

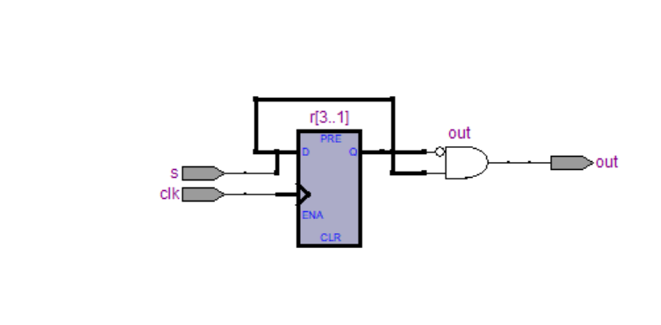
r2 <= r1;

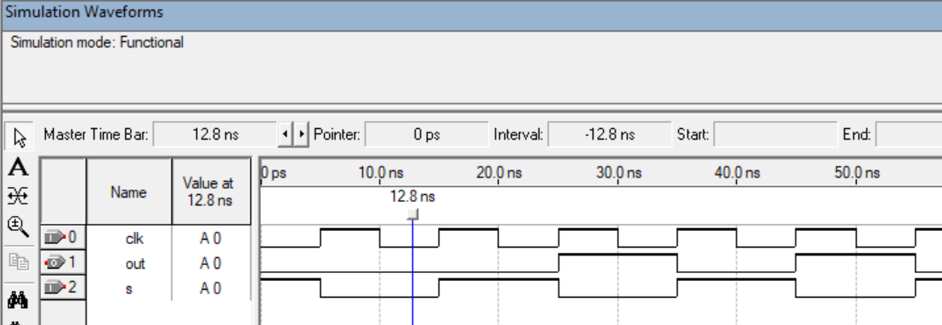
r3 <= r2;

end

assign out = ~r3 & r2;

endmodule





8.

module aa(clk, reset, Up, count);

input clk, reset;

input Up ;

output [2:0] count;

reg [2:0] count = 0;

always @ (posedge clk )

begin

if (reset)

count <= 0;

else if (Up == 1 && count >= 0 && count <= 7)

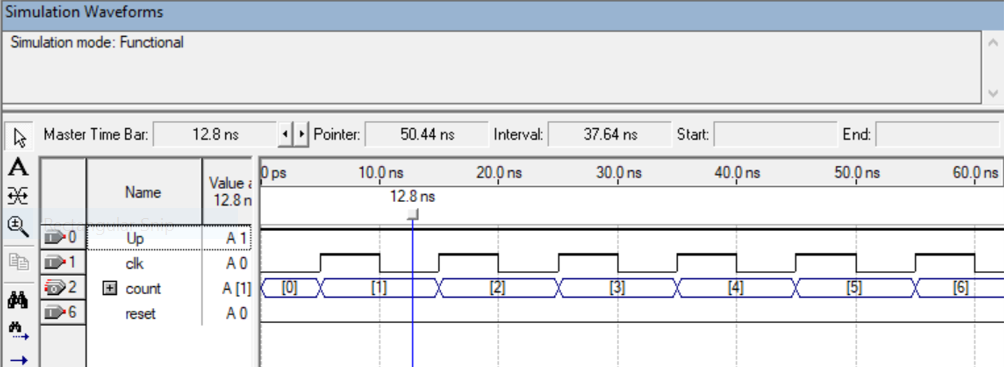
count = count + 1;

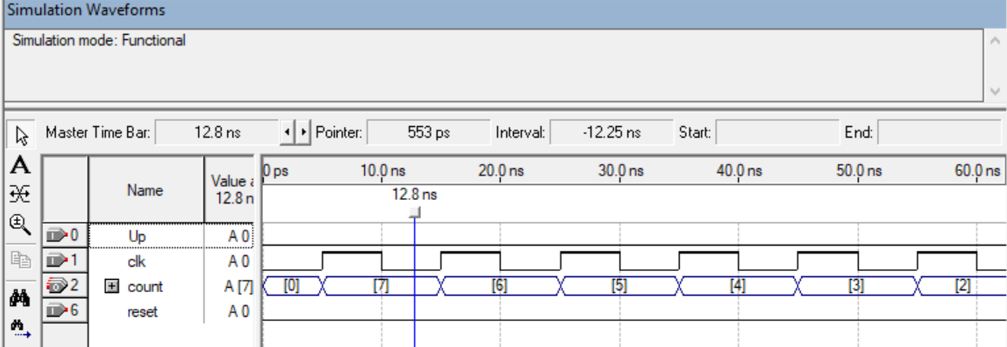
else if ( Up == 0 && count >= 0 && count <=7)

count = count - 1;

end

endmodule





9.

module aa( clk, rst, inp, outp);

input clk, rst, inp;

output outp;

reg [2:0] state;

reg outp;

always @( posedge clk, posedge rst )

begin

if( rst )

state <= 3'b000;

else

begin

case( state )

3'b000:

begin

if( inp ) state <= 3'b000;

else state <= 3'b001;

end

3'b001:

begin

if( inp ) state <= 3'b010;

else state <= 3'b001;

end

3'b010:

begin

if( inp ) state <= 3'b000;

else state <= 3'b011;

end

3'b011:

begin

if( inp ) state <= 3'b100;

else state <= 3'b001;

end

3'b100:

begin

if( inp ) state <= 3'b000;

else state <= 3'b101;

end

3'b101:

begin

if( inp ) state <= 3'b100;

else state <= 3'b001;

end

endcase

end

end

always @(posedge clk, posedge rst)

begin

if( rst )

outp <= 0;

else if( state == 3'b101 )

outp <= 1;

else outp <= 0;

end

endmodule

