**190905514 MOHAMMAD TOFIK**

**WEEK 3 LAB 4**

**1.**using for loop write Verilog code to convert an n bit gray code into equivalent binary code.

module greytobin (bin, grey);

parameter n = 4;

input [n-1: 0] grey;

output [n-1: 0] bin;

reg [n-1: 0] bin;

integer i;

always @(grey)

begin bin[n-1] = grey[n-1];

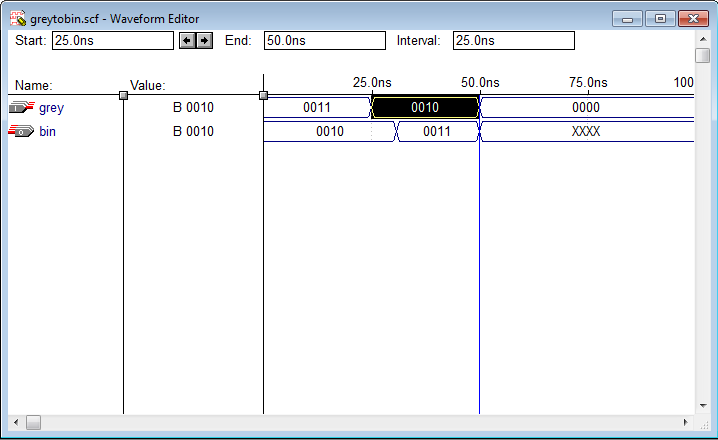
for(i = n-2; i >= 0; i = i-1)

bin[i] = bin[i+1]^grey[i];

end

endmodule

**OUTPUT :**



**2.**write and simulate the Verilog code for 4-bit comparator using 2-bit comparators.

module bit4cmp(A,B,g,l,e);

input[3:0] A,B;

output g,l,e;

wire g1,e1,g2,e2;

bit2cmp one(A[1:0],B[1:0],g1,e1);

bit2cmp two(A[3:2],B[3:2],g2,e2);

assign e = e1&e2;

assign g = g2|(e2&g1);

assign l = ~(e|g);

endmodule

module comparator2(A,B,g,e);

input[1:0] A,B;

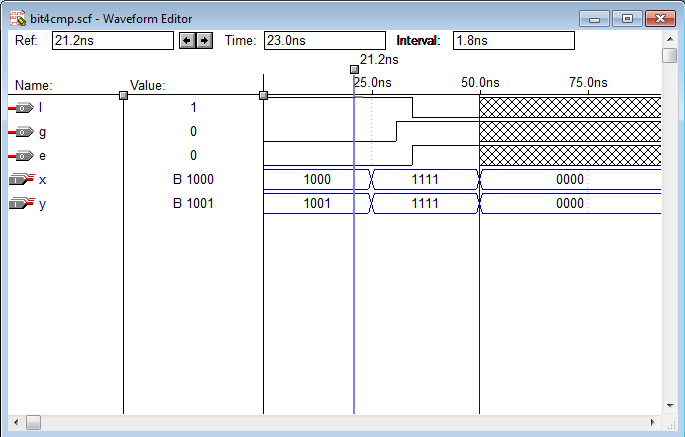
output g,e;

assign e = (~(A[1]^B[1])&~(A[0]^B[0]));

assign g = (A[1]&~B[1])|(~(A[1]^B[1])&(A[0]&~B[0]));

endmodule

**OUTPUT :**



**3.** Write behavioral Verilog code for

 an 8 to 1 multiplexer using **case** statement

 a 2 to 1 multiplexer using the **if-else** statement.

Using the above modules write the hierarchical code for a 16 to 1 multiplexer

1.

module MUX(d,selection,z);

input[8:0]d;

input[2:0]selection;

output z;

reg z ;

always @(d or selection)

begin

case (selection)

3'b000:z=d[0];

3'b001:z=d[1];

3'b010:z=d[2];

3'b011:z=d[3];

3'b100:z=d[4];

3'b101:z=d[5];

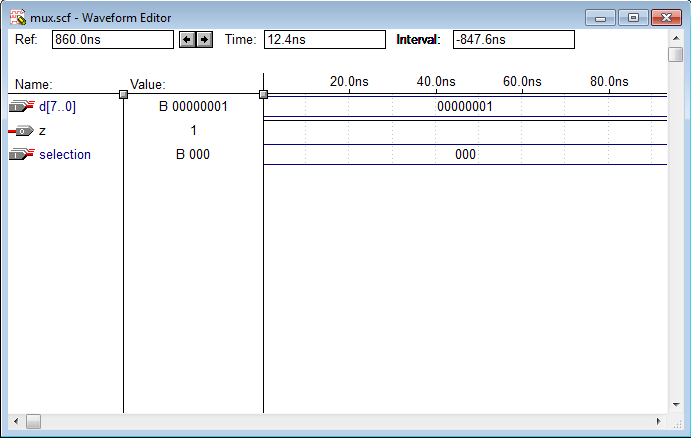
3'b110:z=d[6];

3'b111:z=d[7];

endcase

end

endmodule



**2.**  a 2 to 1 multiplexer using the **if-else** statement.

module mux2( D0, D1, S, Y);

input D0, D1, S;

wire D0, D1;

output Y;

reg Y;

always @(D0 or D1 or S)

begin

if(S)

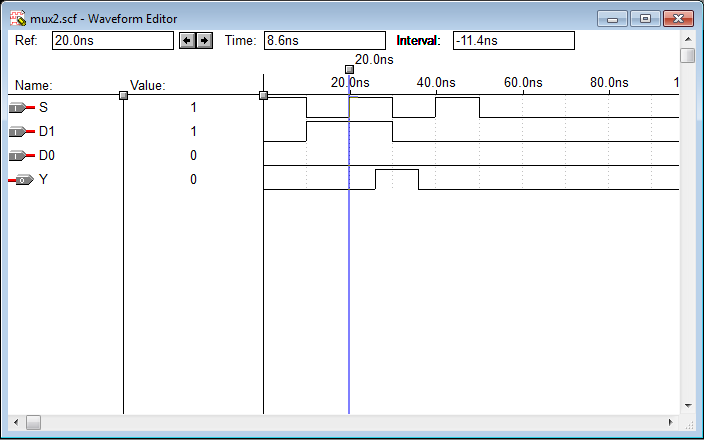
Y= D1;

else

Y=D0;

end

endmodule



**3.** Using the above modules write the hierarchical code for a 16 to 1 multiplexer

module mux16to(p, D, S);

input [15:0] D;

input [3:0] S;

output p;

wire [15:0] D;

wire [3:0] S;

wire [1:0] x;

MUX m1(D[7:0], S[2:0], x[0]);

MUX m2(D[15:8], S[2:0], x[1]);

mux2 m3(x[1], x[0], S[3], p);

endmodule

module MUX(d,selection,z);

input[7:0]d;

input[2:0]selection;

output z;

reg z ;

always @(d or selection)

begin

case (selection)

3'b000:z=d[0];

3'b001:z=d[1];

3'b010:z=d[2];

3'b011:z=d[3];

3'b100:z=d[4];

3'b101:z=d[5];

3'b110:z=d[6];

3'b111:z=d[7];

endcase

end

endmodule

module mux2( D0, D1, S, Y);

input D0, D1, S;

wire D0, D1;

output Y;

reg Y;

always @(D0 or D1 or S)

begin

if(S)

Y= D1;

else

Y=D0;

end

endmodule

**OUTPUT :**

