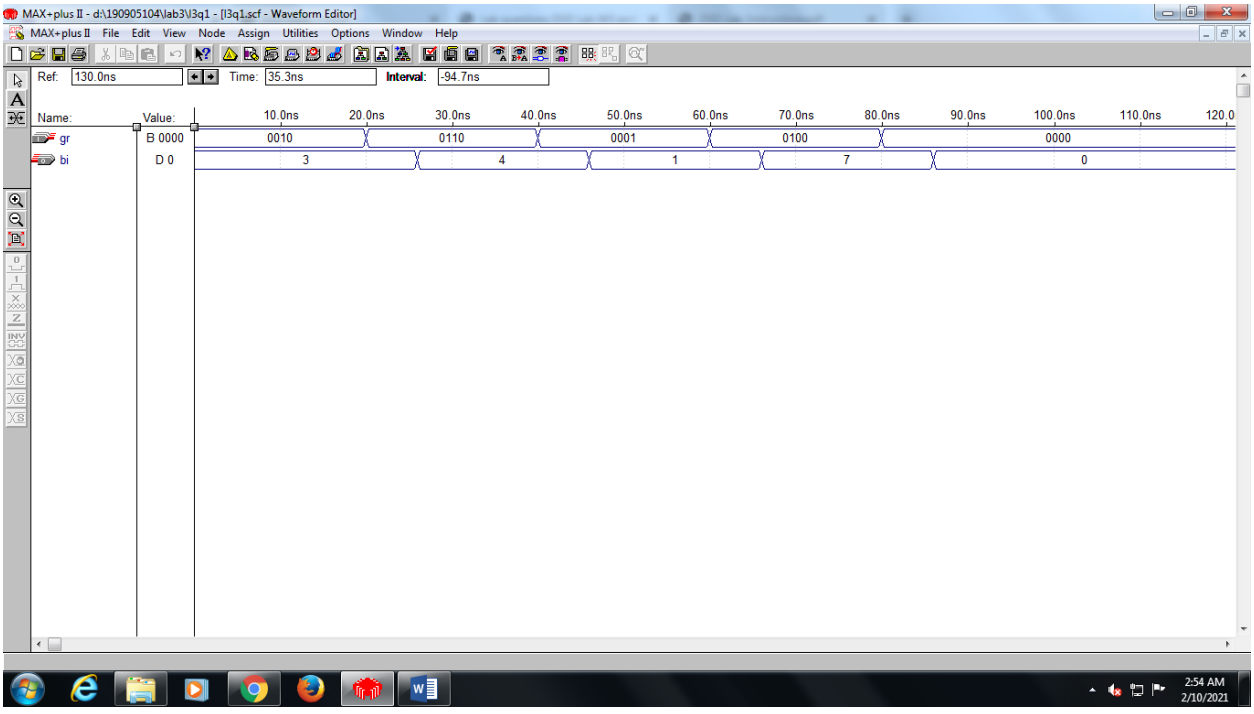
Lab 3

1) Using for loop, write behavioral Verilog code to convert an N bit grey code into equivalent binary code

```
// Grey to binary
module l3q1(gr, bi);
parameter n = 4;
input [n-1:0]gr;
output [n-1:0]bi;
reg [n-1:0]bi;
integer i;

always@(gr)
begin
bi[n-1] = gr[n-1];
for(i=n-2;i>=0;i=i-1)
begin
bi[i] = bi[i+1]^gr[i];
end
end
endmodule
```

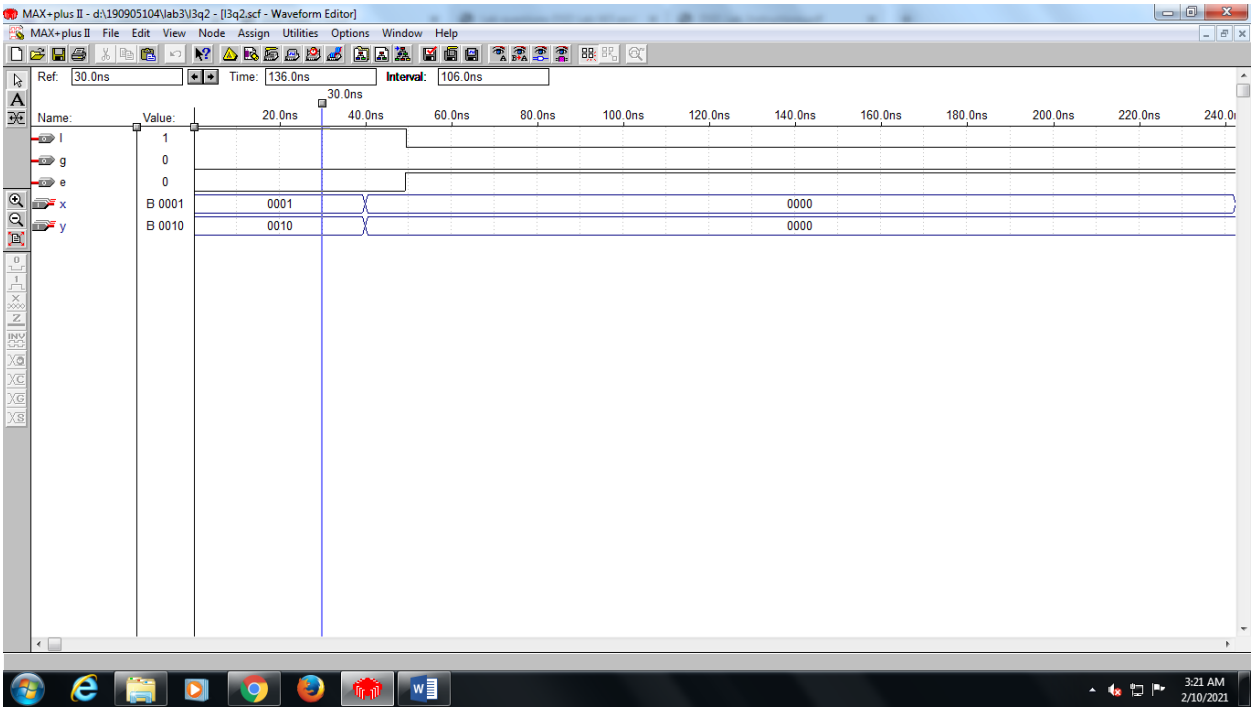


2)

Write and simulate the Verilog code for a 4-bit comparator using 2-bit comparators.

// Write and simulate the Verilog code for a 4-bit comparator using 2-bit comparators.

```
module comp2bit(x0, y0, x1, y1, l, g, e);
input x0, y0, x1, y1;
output l, g, e;
wire i, j;
assign j = ~(x1 ^ y1);
assign i = ~(x0 ^ y0);
assign e = i & j;
assign g = (x1 & ~y1)|(j & x0 & y0);
assign l = ~(g | e);
endmodule

// 4 bit comp
module l3q2(x, y, l, g, e);
input [3:0]x;
input [3:0]y;
output l, g, e;
wire l1, l2, g1, g2, e1, e2;
comp2bit c1(x[0], y[0], x[1], y[1], l1, g1, e1);
comp2bit c2(x[2], y[2], x[3], y[3], l2, g2, e2);
assign e = e1 & e2;
assign g = g2 | (e2 & g1);
assign l = l2 | (e2 & l1);
endmodule
```



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.

i)

```
module mux8to1_case(I, sel, Y);

input [7:0]I;

input [2:0]sel;

output Y;

reg Y;

always@(I or sel or Y)

begin

case(sel)

3'b000: Y=I[0];
```

3'b001: Y=I[1];

3'b010: Y=I[2];

3'b011: Y=I[3];

3'b100: Y=I[4];

3'b101: Y=I[5];

3'b110: Y=I[6];

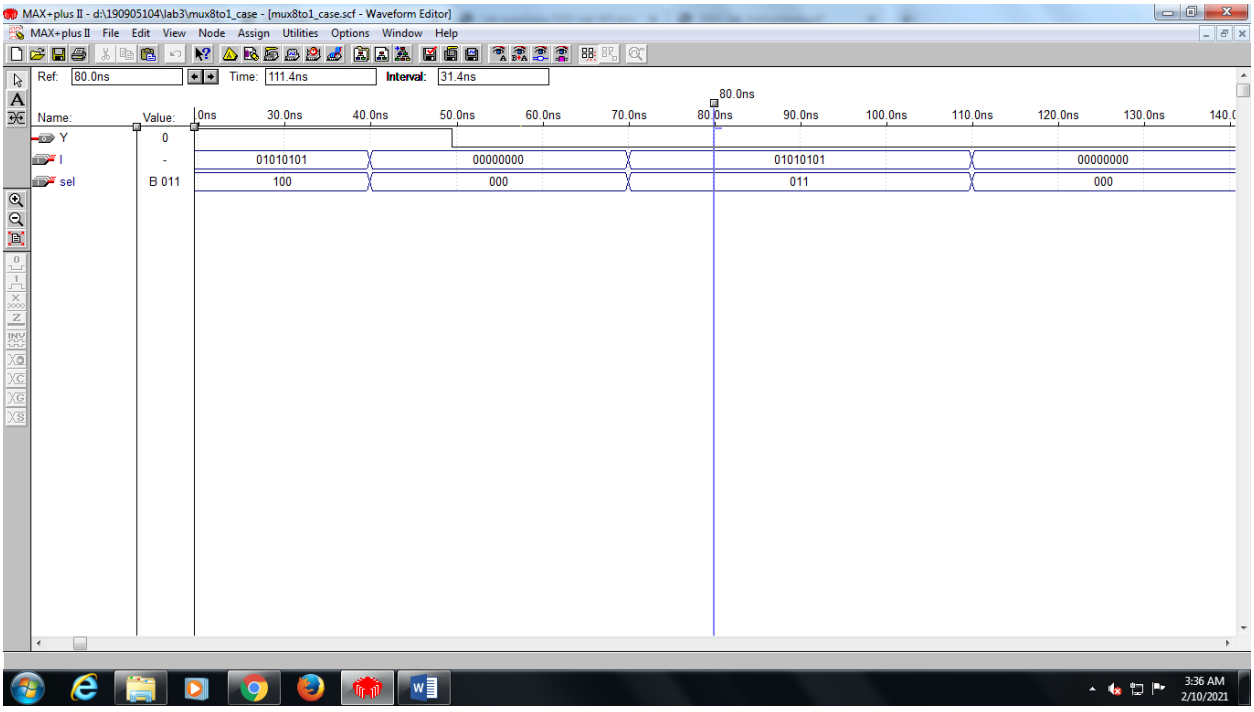
3'b111: Y=I[7];

default: Y=1'b0;

endcase

end

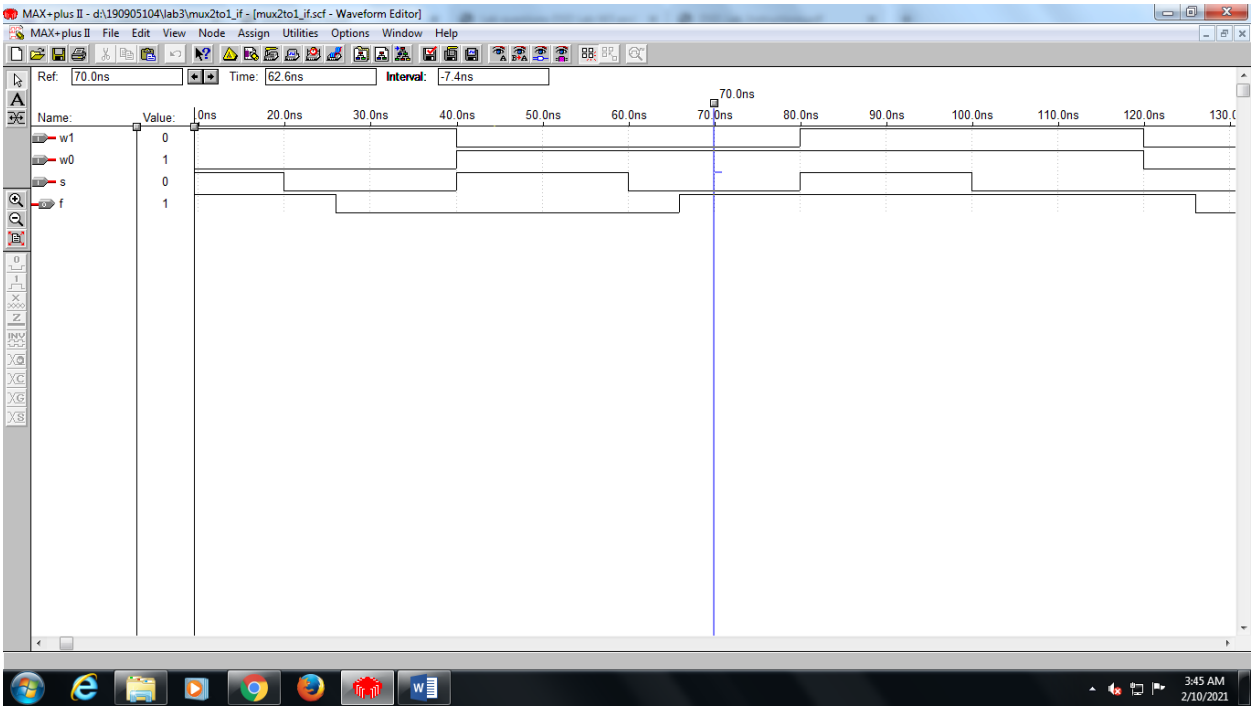
endmodule



ii)

module mux2to1_if(w0, w1, s, f);

```
input w0, w1, s;

output f;

reg f;

always @(w0 or w1 or s)

begin

if(s==0)

assign f = w0;

else

assign f = w1;

end

endmodule
```

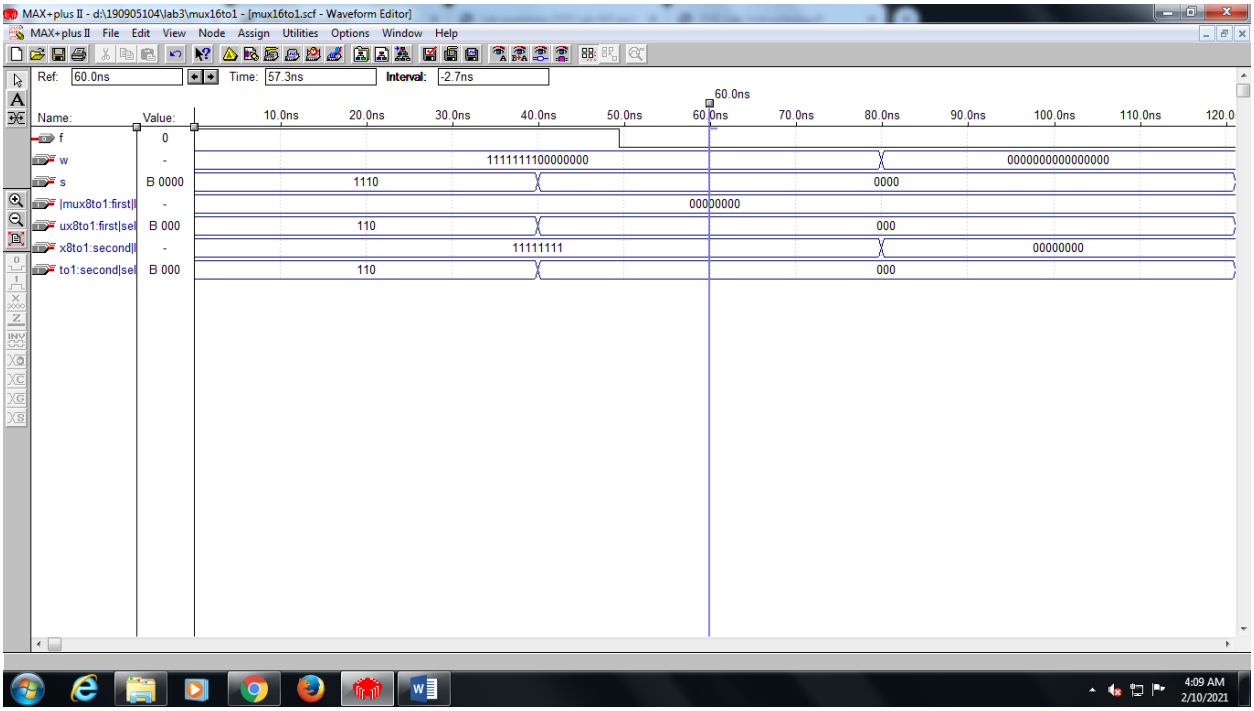


Using the above modules write the hierarchical code for a 16 to 1 multiplexer

```
module mux2to1(w0, w1, s, f);
input w0, w1, s;
output f;
reg f;
always @(w0 or w1 or s)
begin
if(s==0)
assign f = w0;
else
assign f = w1;
end
endmodule
```

```
module mux8to1(I, sel, Y);
input [7:0]I;
input [2:0]sel;
output Y;
reg Y;
always@(I or sel or Y)
begin
case(sel)
3'b000: Y=I[0];
3'b001: Y=I[1];
```

```
3'b010: Y=I[2];

3'b011: Y=I[3];

3'b100: Y=I[4];

3'b101: Y=I[5];

3'b110: Y=I[6];

3'b111: Y=I[7];

default: Y=1'b0;

endcase

end

endmodule


module mux16to1(w, s, f);

input [15:0]w;

input [3:0]s;

output f;

reg f;

reg s1, s2;

//always @(w or s)

//begin

mux8to1 first(w[7:0],s[2:0], s1);

mux8to1 second(w[15:8], s[2:0], s2);

mux2to1 final(s1, s2, s[3], f);

//end

endmodule
```


MAX+plus II - d:\190905104\lab3\mux16to1 - [mux16to1.scf - Waveform Editor]
MAX+plus II File Edit View Node Assign Utilities Options Window Help
Ref: 60.0ns
Time: 57.3ns
Interval: -2.7ns
60.0ns
Name:
Value:
10.0ns 20.0ns 30.0ns 40.0ns 50.0ns 60.0ns 70.0ns 80.0ns 90.0ns 100.0ns 110.0ns 120.0
f 0
w -
1111111100000000
0000000000000000
s B 0000
1110
0000
|mux8to1:first|l -
00000000
ux8to1:first|sel B 000
110
000
x8to1:second|l -
11111111
00000000
to1:second|sel B 000
110
000
4:09 AM
2/10/2021