WEEK 3

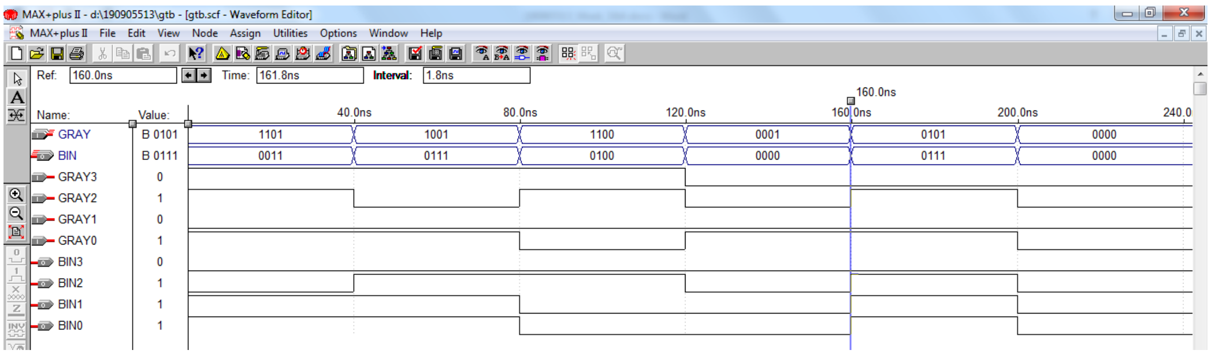
Lab Exercises:

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

Program:

```
module gtb (BIN, GRAY);
parameter N = 4;
input [N-1: 0] GRAY;
output [N-1: 0] BIN;
reg [N-1: 0] BIN;
integer i;
always @(GRAY)
begin BIN[N-1] = GRAY[N-1];
for(i = N-2; i >= 0; i = i-1)
BIN[i] = BIN[i+1]^GRAY[i];
end
endmodule
```

Waveform:

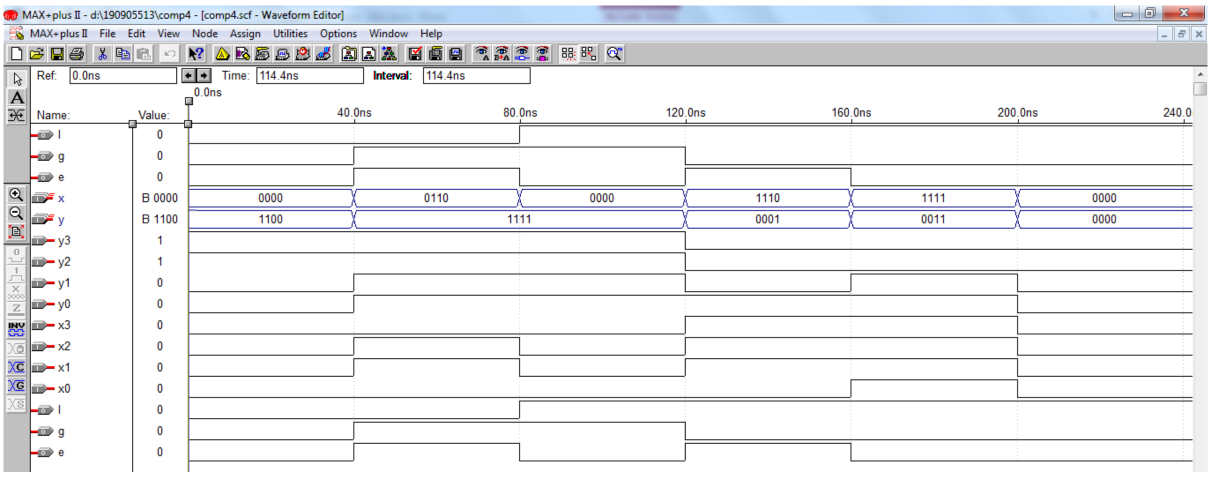

MAX+plus II - d:\190905513\gtb - [gtb.scf - Waveform Editor]
MAX+plus II File Edit View Node Assign Utilities Options Window Help
Ref: 160.0ns
Time: 161.8ns
Interval: 1.8ns
Name:
Value:
40.0ns
80.0ns
120.0ns
160.0ns
200.0ns
240.0
GRAY B 0101 1101 1001 1100 0001 0101 0000
BIN B 0111 0011 0111 0100 0000 0111 0000
GRAY3 0
GRAY2 1
GRAY1 0
GRAY0 1
BIN3 0
BIN2 1
BIN1 1
BIN0 1

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

Program:

```
module comp4(x,y,l,g,e);
input [3:0] x,y;
output l,g,e;
wire l1,g1,e1,l2,g2,e2;
com2bit stage0(x[0],y[0],x[1],y[1],l1,g1,e1);
com2bit stage1(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

module com2bit(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
```

Waveform:



3. Write behavioral Verilog code for

- **an 8 to 1 multiplexer using case statement**
- **a 2 to 1 multiplexer using the if-else statement.**

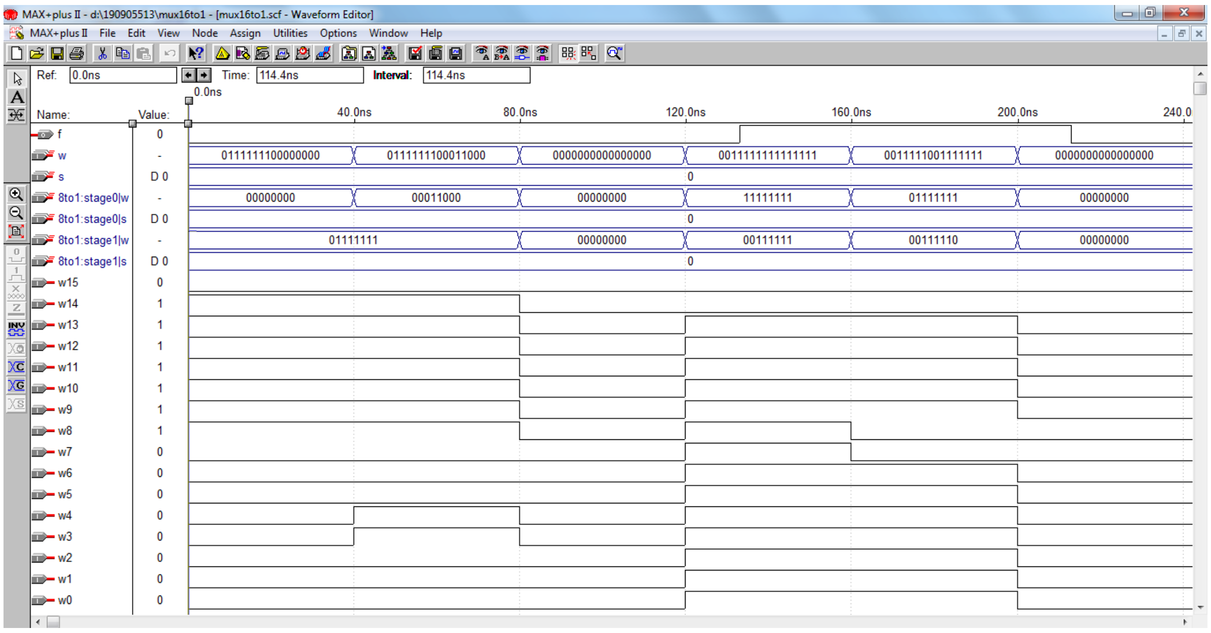
Using the above modules write the hierarchical code for a 16 to 1 multiplexer.

Program:

```
module mux16to1(w,s,f);

input [15:0]w;

input [3:0]s;

output f;

reg f;

mux8to1 stage0(w[7:0],s[2:0],s1);

mux8to1 stage1(w[15:8],s[2:0],s2);

mux2to1 stage2(s1,s2,s[3],f);

endmodule

module mux8to1(w,s,f);

input [7:0]w;

input [2:0]s;

output f;

reg f;

always @(w or s or f)

begin

case(s)

0:f=w[0];

1:f=w[1];

2:f=w[2];

3:f=w[3];

4:f=w[4];
```

```
5:f=w[5];

6:f=w[6];

7:f=w[7];

endcase

end

endmodule

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
```

Waveform:


MAX+plus II - d:\190905513\mux16to1 - [mux16to1.scf - Waveform Editor]
MAX+plus II File Edit View Node Assign Utilities Options Window Help
Ref: 0.0ns
Time: 114.4ns
Interval: 114.4ns
0.0ns
40.0ns
80.0ns
120.0ns
160.0ns
200.0ns
240.0
Name:
Value:
f 0
w -
s D 0
8to1:stage0|w -
8to1:stage0|s D 0
8to1:stage1|w -
8to1:stage1|s D 0
w15 0
w14 1
w13 1
w12 1
w11 1
w10 1
w9 1
w8 1
w7 0
w6 0
w5 0
w4 0
w3 0
w2 0
w1 0
w0 0
0111111100000000
0111111100011000
0000000000000000
0011111111111111
0011111001111111
0000000000000000
0
00000000
00011000
00000000
11111111
01111111
00000000
0
01111111
00000000
00111111
00111110
00000000
0

WEEK 4

Lab Exercises:

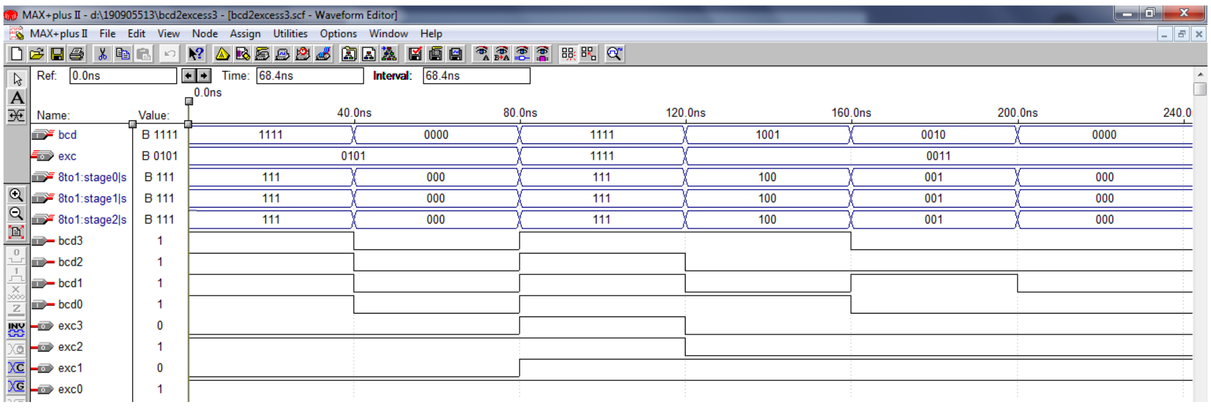
1: Write and simulate the Verilog code for a BCD to Excess 3 code converter using 8 to 1 multiplexers and other necessary gates.

Program:

```
module bcd2excess3(bcd,exc);

input [3:0]bcd;

output [3:0]exc;

mux8to1 stage0({1'b0,1'b0,bcd[0],1'b1,1'b1,X,X,X}, bcd[3:1], exc[3]);

mux8to1 stage1({bcd[0],1'b1,~bcd[0],1'b0,bcd[0],X,X,X}, bcd[3:1], exc[2]);

mux8to1 stage2({~bcd[0],bcd[0],~bcd[0],bcd[0],~bcd[0],X,X,X}, bcd[3:1], exc[1]);

assign exc[0]=~bcd[0];

endmodule

module mux8to1(w,s,f);

input [0:7]w;

input [2:0]s;

output f;

reg f;

always @(w or s or f)
```

```
begin

case(s)

0:f=w[0];

1:f=w[1];

2:f=w[2];

3:f=w[3];

4:f=w[4];

5:f=w[5];

6:f=w[6];

7:f=w[7];

endcase

end

endmodule
```

Waveform:


MAX+plus II - d:\190905513\bcd2excess3 - [bcd2excess3.scf - Waveform Editor]

Ref: 0.0ns Time: 68.4ns Interval: 68.4ns

```
Name:            Value:   0.0ns     40.0ns      80.0ns     120.0ns    160.0ns    200.0ns   240.0
bcd              B 1111   1111      0000        1111       1001       0010       0000
exc              B 0101   0101                  1111                  0011
8to1:stage0|s    B 111    111       000         111        100        001        000
8to1:stage1|s    B 111    111       000         111        100        001        000
8to1:stage2|s    B 111    111       000         111        100        001        000
bcd3             1
bcd2             1
bcd1             1
bcd0             1
exc3             0
exc2             1
exc1             0
exc0             1
```

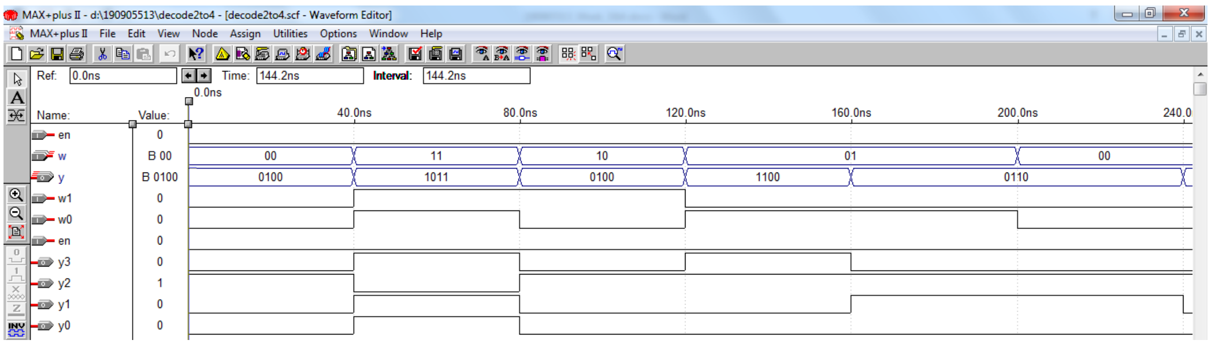
2: Write behavioral Verilog code for a 2 to 4 decoder with active low enable input and active high output using case statement. Using this, design a 4 to 16 decoder with active low enable input and active high output and write the Verilog code for the same.

Program:

```
module decode2to4(w,y,en);

input [1:0]w;

input en;

output [3:0]y;

reg [3:0]y;

always @(w or en)

begin

if(en == 0)

y = 15;

else

begin

case (w)

0:y = 14;

1:y = 13;

2:y = 11;

3:y = 7;

endcase

end

end
```

```
endmodule

module dec4to16(w,y,en);

input en;

input [3:0]w;

output [15:0]y;

reg [15:0]f;

wire [3:0]x;

decode2to4 stage0(w[3:2],x[3:0],en);

decode2to4 stage1(w[1:0],f[3:0],~x[0]);

decode2to4 stage2(w[1:0],f[7:4],~x[1]);

decode2to4 stage3(w[1:0],f[11:8],~x[2]);

decode2to4 stage4(w[1:0],f[15:12],~x[3]);

assign y=~f;

endmodule
```

Waveform:



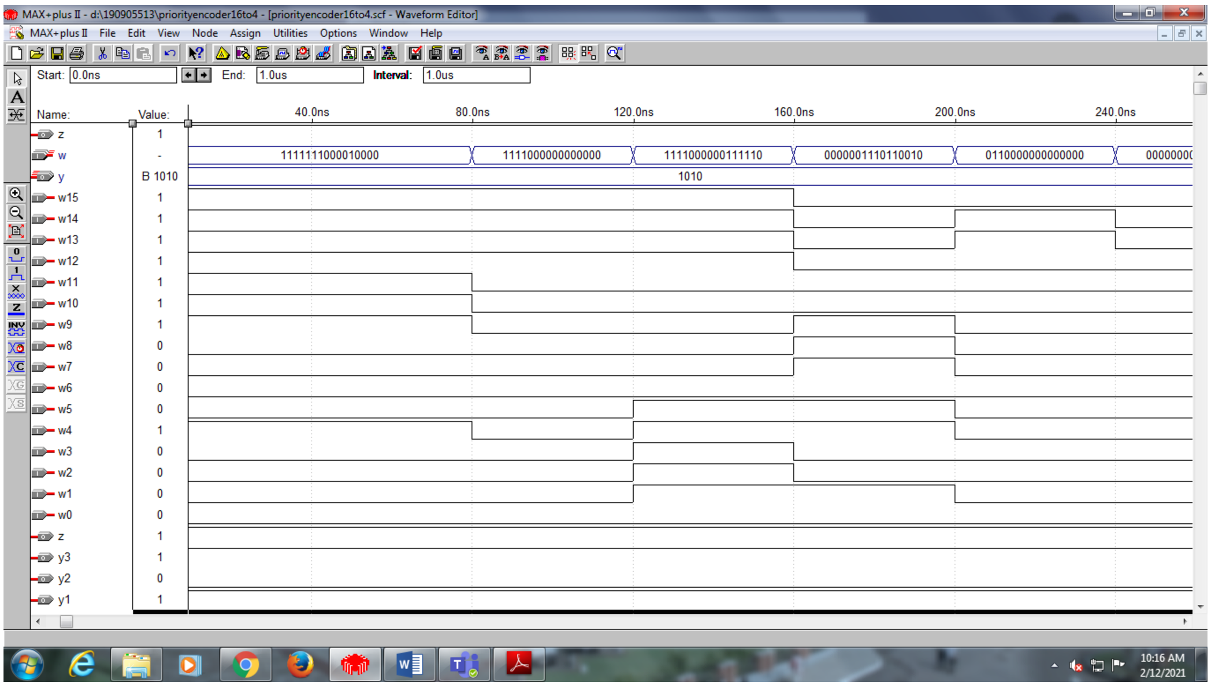
3: Write behavioral Verilog code for 16 to 4 priority encoder using for loop.

Program:

```
module priorityencoder16to4(w,y,z);

input [15:0]w;

output [3:0]y;

output z;

reg [3:0]y;

reg z;

integer k;

always @(w)

begin

z = 0;

if(w == 0)

y = 0;

else

begin

for(k = 0 ; k < 16 ; k = k + 1)
```

```
begin

if(w[k] == 1)

y = k;

end

z = 1;

end

end

endmodule
```

Waveform:



MAX+plus II - d:\190905513\priorityencoder16to4 - [priorityencoder16to4.scf - Waveform Editor]

Start: 0.0ns End: 1.0us Interval: 1.0us

Name:	Value:
z	1
w	-
y	B 1010
w15	1
w14	1
w13	1
w12	1
w11	1
w10	1
w9	1
w8	0
w7	0
w6	0
w5	0
w4	1
w3	0
w2	0
w1	0
w0	0
z	1
y3	1
y2	0
y1	1