LAB – 3

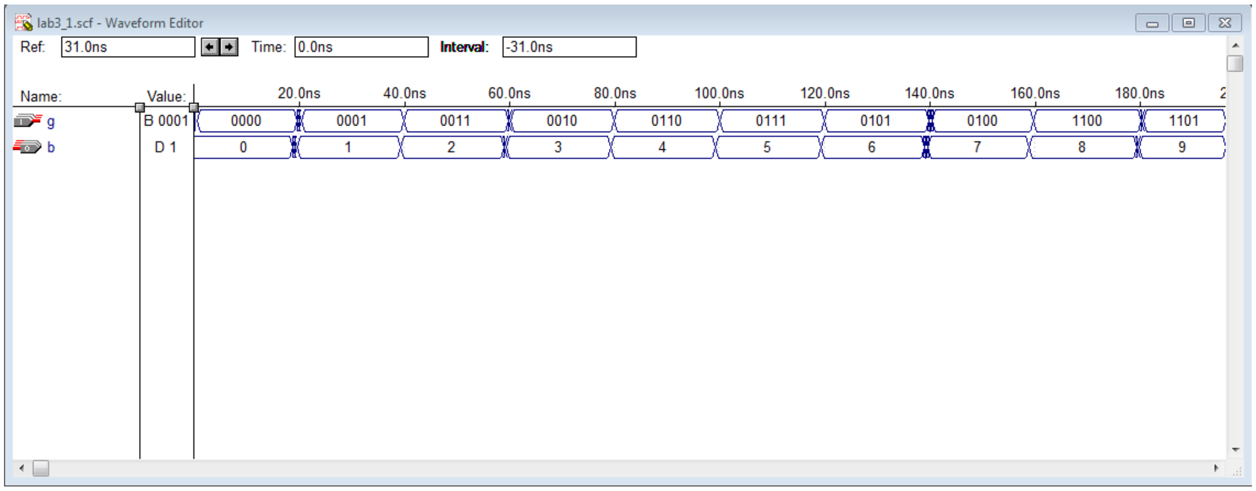
Exercise:

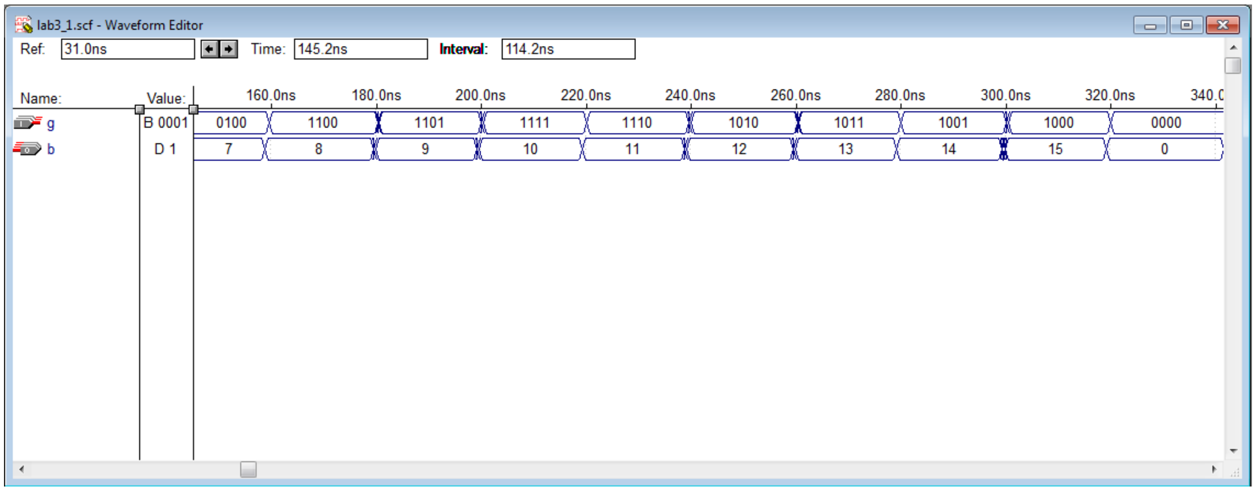
1. Using **for** loop, write behavioral Verilog code to convert an N bit grey code into equivalent binary code.

Verilog code :

```
module lab3_1(g,b);
parameter n=4;
input [n-1:0]g;
output [n-1:0]b;
reg [n-1:0]b;
integer i;
always @(g)
begin
b[n-1]=g[n-1];
for(i=n-2;i>=0;i=i-1)
begin
b[i]=b[i+1]^g[i];
end
end
endmodule
```

Output :



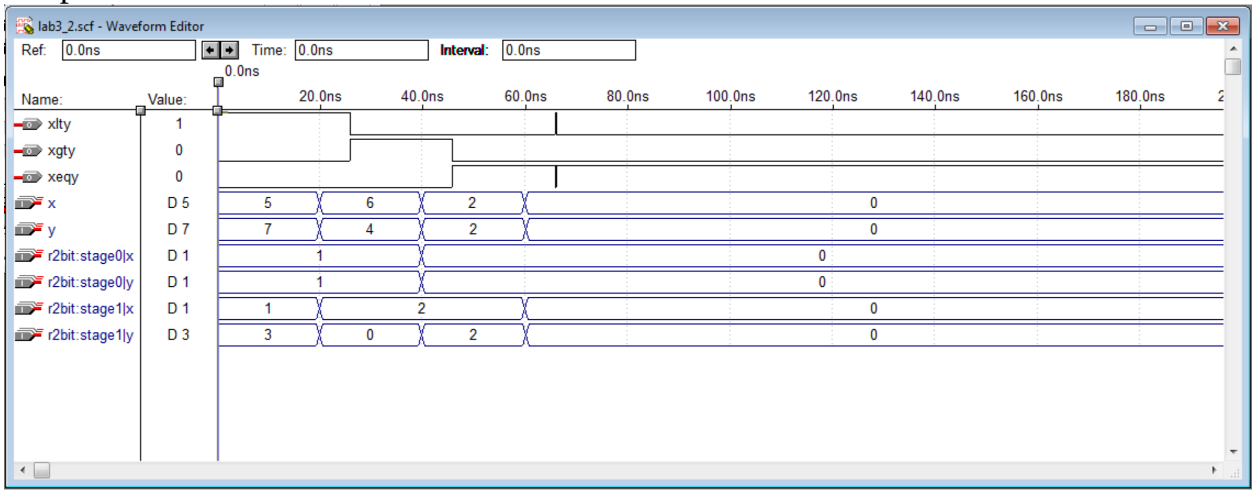


2. Write and simulate the Verilog code for a 4-bit comparator using 2-bit comparators.

Verilog code :

```
module comparator2bit(x,y,xlty,xeqy,xgty);
input [1:0]x,y;
output xeqy,xlty,xgty;
wire i1,i0;
assign i1 =~(x[1]^y[1]);
assign i0 =~(x[0]^y[0]);
assign xeqy =i1&i0;
assign xgty =(x[1] & ~y[1])|(i1 & x[0] & ~y[0]);
assign xlty =~(xeqy | xgty);
endmodule

module lab3_2(x,y,xlty,xeqy,xgty);
input [3:0]x,y;
output xeqy,xlty,xgty;
wire e1,e2,g1,g2,l1,l2;
comparator2bit stage0(x[3:2],y[3:2],l1,e1,g1);
comparator2bit stage1(x[1:0],y[1:0],l2,e2,g2);
assign xeqy =e1&e2;
assign xgty =g1 | (e1 & g2);
assign xlty =l1 | (e1 & l2);
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.

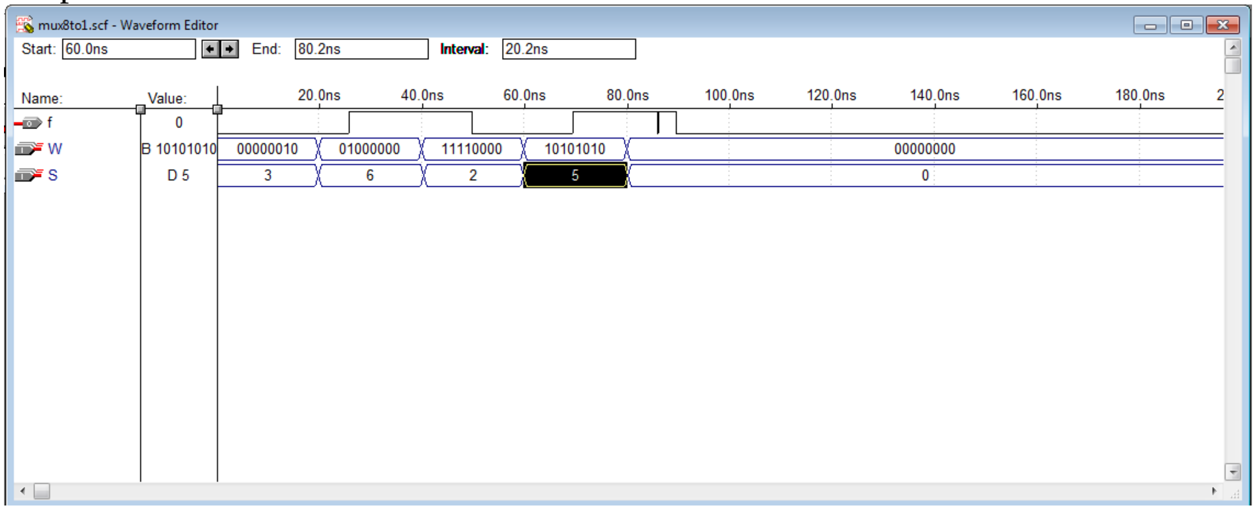
Using the above modules write the hierarchical code for a 16 to 1 multiplexer.

i) Verilog code :

```
module mux8to1(W,S,f);
input [7:0]W;
input [2:0]S;
output f;
reg f;
always @(W or S)
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
endmodule
```

Output :

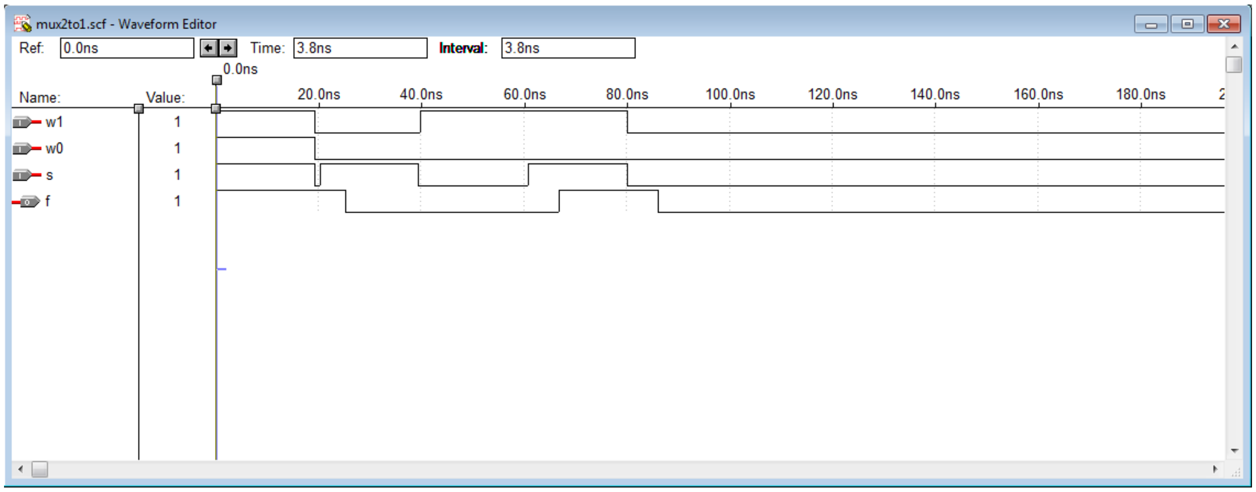


```
mux8to1.scf - Waveform Editor
Start: 60.0ns    End: 80.2ns    Interval: 20.2ns

Name:   Value:
f       0
W       B 10101010   00000010  01000000  11110000  10101010  00000000
S       D 5          3         6         2         5         0
```

ii) Verilog code :

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

Output :



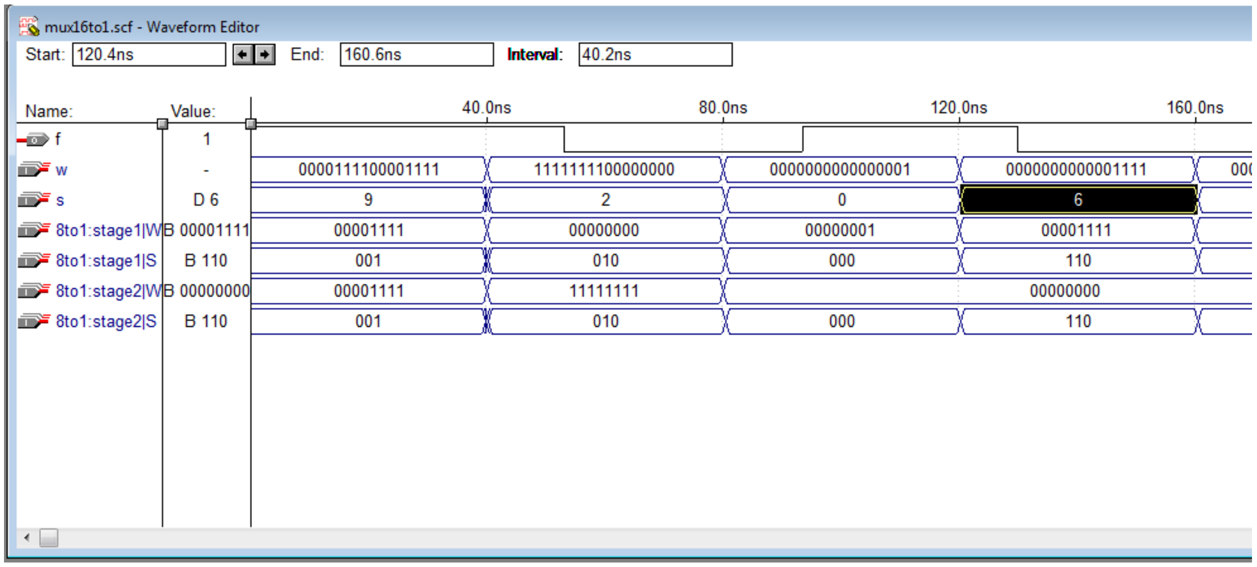
iii)

Verilog code :

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

module mux8to1(W,S,f);
input [7:0]W;
input [2:0]S;
output f;
reg f;
always @(W or S)
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
endmodule

module mux16to1(w,s,f);
input [15:0]w;
input [3:0]s;
output f;
wire [1:0]i;
mux8to1 stage1(w[7:0],s[2:0],i[0]);
mux8to1 stage2(w[15:8],s[2:0],i[1]);
mux2to1 stage_last(i[0],i[1],s[3],f);
endmodule
```

Output :



```
mux16to1.scf - Waveform Editor
Start: 120.4ns   End: 160.6ns   Interval: 40.2ns

Name:              Value:          40.0ns              80.0ns              120.0ns             160.0ns
f                  1
w                  -         0000111100001111 | 1111111100000000 | 0000000000000001 | 0000000000001111 | 000
s                  D 6       9                | 2                | 0                | 6                |
8to1:stage1|W      B 00001111 00001111        | 00000000         | 00000001         | 00001111         |
8to1:stage1|S      B 110     001              | 010              | 000              | 110              |
8to1:stage2|W      B 00000000 00001111        | 11111111         | 00000000                            |
8to1:stage2|S      B 110     001              | 010              | 000              | 110              |
```

****

LAB – 4

Exercise:

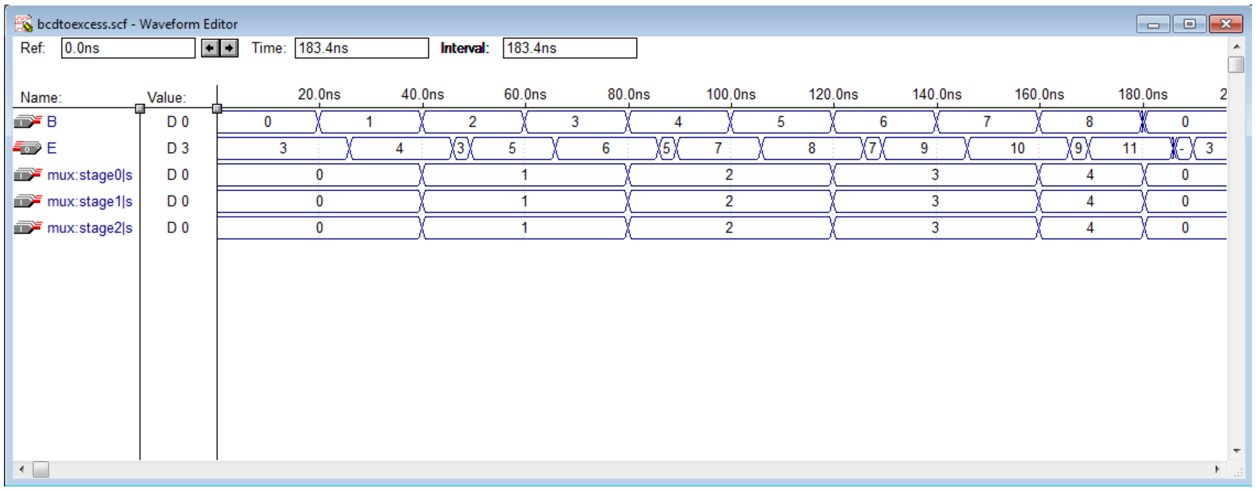
1. Write and simulate the Verilog code for a BCD to Excess 3 code converter using 8 to 1multiplexers and other necessary gates.

Verilog code :

```
module bcdtoexcess(b,e);
input [3:0]b;
output [3:0]e;
wire z,one;
wire [0:7] w3,w2,w1,w0;
assign z=0;
assign one=1;
assign w3={z,z,b[0],one,one,z,z,z};
assign w2={b[0],one,~b[0],z,b[0],z,z,z};
assign w1={~b[0],b[0],~b[0],b[0],~b[0],z,z,z};
assign w0={~b[0],~b[0],~b[0],~b[0],~b[0],z,z,z};
mux8to1 stage0(w0,b[3:1],e[0]);
mux8to1 stage1(w1,b[3:1],e[1]);
mux8to1 stage2(w2,b[3:1],e[2]);
mux8to1 stage3(w3,b[3:1],e[3]);
endmodule

module mux8to1(w,s,f);
input [0:7]w;
input [2:0]s;
output f;
function mux;
input [0:7]x;
input [2:0]ss;
case(ss)
0:mux=x[0];
1:mux=x[1];
2:mux=x[2];
3:mux=x[3];
4:mux=x[4];
5:mux=x[5];
6:mux=x[6];
```

```
7:mux=x[7];
endcase
endfunction
assign f=mux(w,s);
endmodule
```

Output :


bcdtoexcess.scf - Waveform Editor

Ref: 0.0ns Time: 183.4ns Interval: 183.4ns

```
Name:          Value:
B              D 0
E              D 3
mux:stage0|s   D 0
mux:stage1|s   D 0
mux:stage2|s   D 0
```

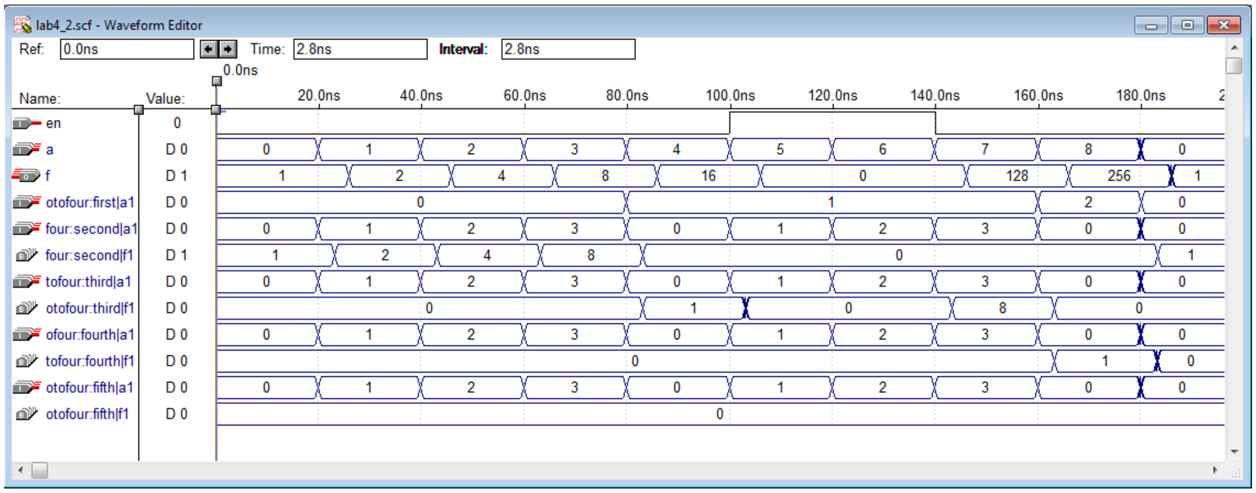
2. Write behavioral Verilog code for a 2 to 4 decoder with active low enable input and active high output using **case** statement. Using this, design a 4 to 16 decoder with active low enable input and active high output and write the Verilog code for the same.

Verilog code :

```
module twotofour(a1, en1, f1);
input en1;
input [1:0]a1;
output [3:0]f1;
reg [3:0]f1;
always @(a1 or en1)
begin
case({en1, a1})
3'b000: f1 = 4'b1000;
3'b001: f1 = 4'b0100;
3'b010: f1 = 4'b0010;
3'b011: f1 = 4'b0001;
default: f1= 4'b0000;
endcase
```

```
end
endmodule
module lab4_2(a, en, f);
input en;
input [3:0]a;
output [15:0]f;
wire [3:0]w;
twotofour first(a[3:2], en, w);
twotofour second(a[1:0], ~w[0], f[3:0]);
twotofour third(a[1:0], ~w[1], f[7:4]);
twotofour fourth(a[1:0], ~w[2], f[11:8]);
twotofour fifth(a[1:0], ~w[3], f[15:12]);
endmodule
```

Output :



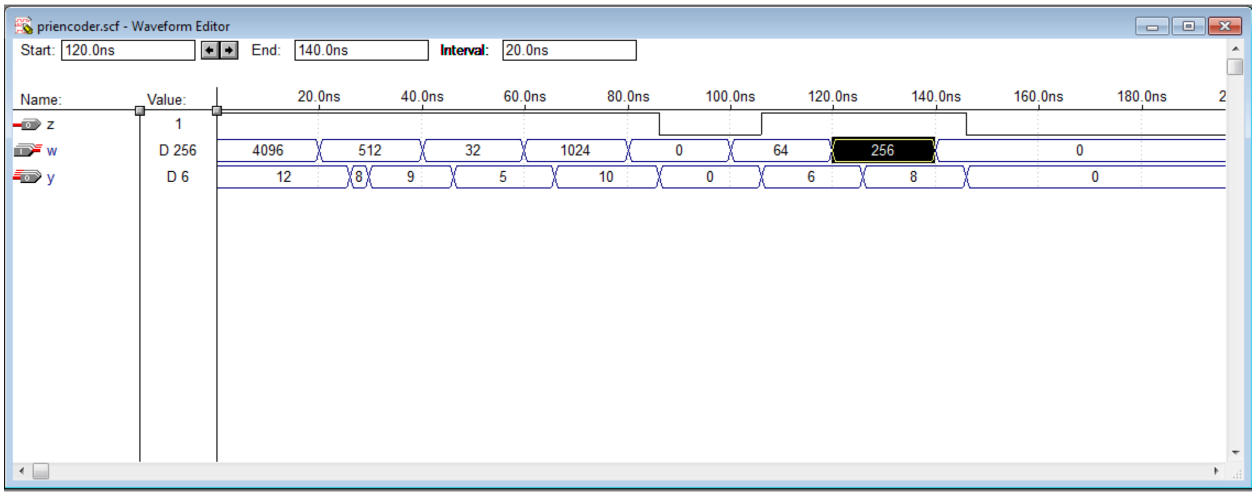
3. Write behavioral Verilog code for 16 to 4 priority encoder using **for** loop.

Verilog code :

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

input [15:0]w;

output z;

output [3:0]y;
```

```
reg [3:0]y;
reg z;
integer k;
always @(w)
begin
z=0;
y=4'bx;
for(k=0;k<16;k=k+1)
begin
if(w[k])
begin
y=k;
z=1;
end
end
end
endmodule
```

Output :


priencoder.scf - Waveform Editor
Start: 120.0ns
End: 140.0ns
Interval: 20.0ns
Name:
Value:
20.0ns
40.0ns
60.0ns
80.0ns
100.0ns
120.0ns
140.0ns
160.0ns
180.0ns
z
1
w
D 256
4096
512
32
1024
0
64
256
0
y
D 6
12
8
9
5
10
0
6
8
0

****