WEEK 5

Lab Exercises:

1: Design and simulate a combinational circuit with external gates and a 4 to 16 decoder built using a decoder tree of 2 to 4 decoders to implement the functions below. F= ab'c + a'cd + bcd' , G=acd' + a'b'c and H=a'b'c' + abc + a'cd

Program:

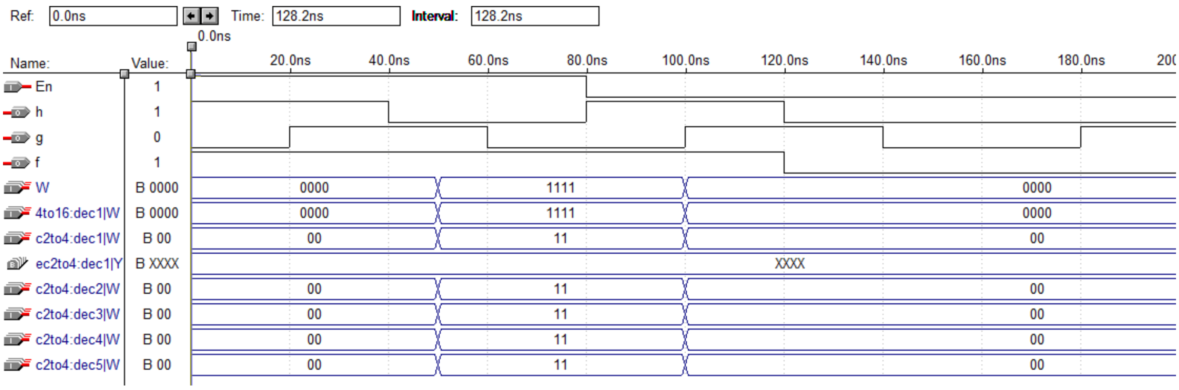
```
module dec2to4(W,En,Y);
input[1:0]W;
input En;
output [0:3]Y;
reg [0:3]Y;
always@(W or En)
begin
if(En==1)
case(W)
0: Y=4'b1000;
1: Y=4'b0100;
2: Y=4'b0010;
3: Y=4'b0001;
endcase
else
Y=4'b0000;
end
endmodule

module dec4to16(W,En,Y);
input [3:0]W;
input En;
output [0:15]Y;
wire[0:3]M;

dec2to4 dec1(W[3:2],En,M[0:3]);
dec2to4 dec2(W[1:0],M[0],Y[0:3]);
dec2to4 dec3(W[1:0],M[1],Y[4:7]);
dec2to4 dec4(W[1:0],M[2],Y[8:11]);
dec2to4 dec5(W[1:0],M[3],Y[12:15]);
endmodule
```

```
module lab5_1(W, En, f, g, h);
input [3:0] W;
input En;
output f, g, h;
wire [0:15]Y;
dec4to16 dec1(W, En, Y);
or(f, Y[3], Y[6], Y[7], Y[10], Y[11], Y[14]);
or(g, Y[2], Y[3], Y[10], Y[14]);
or(h, Y[0], Y[1], Y[3], Y[7], Y[14], Y[15]);
endmodule
```

Waveform:



2: Design and implement a full adder using 2 to 4 decoder(s)and other gates.

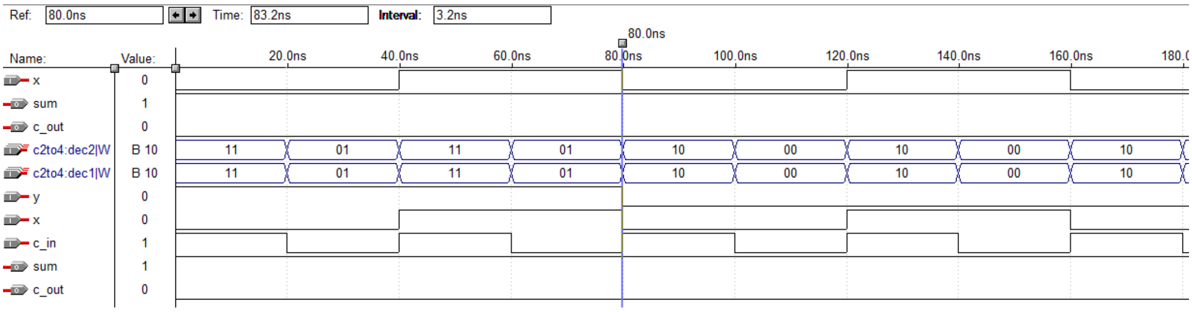
Program:

```
module dec2to4(W,En,Y);
input[1:0]W;
input En;
output [0:3]Y;
reg [0:3]Y;
always@(W or En)
begin
if(En==1)
case(W)
0: Y=4'b1000;
1: Y=4'b0100;
2: Y=4'b0010;
3: Y=4'b0001;
endcase
else
Y=4'b0000;
end
```

```
endmodule

module lab5_2(x, y, c_in, sum, c_out);
input x, y, c_in;
output sum, c_out;
wire [0:3] dec0w;
wire [0:3] dec1w;
wire [0:3] dec2w;
dec2to4 dec0({1'b0, x}, 1'b1, dec0w);
dec2to4 dec1({c_in, y}, dec0w[1], dec1w);
dec2to4 dec2({c_in, y}, dec0w[0], dec2w);
or(c_out, dec2w[3], dec1w[1], dec1w[2], dec1w[3]);
or(sum, dec2w[1], dec2w[2], dec1w[0], dec1w[3]);
endmodule
```

Waveform:



3: Design and simulate the circuit with 3 to 8 decoder(s) and external gates to implement thefunctions below.F(a, b, c, d)= Σm(2,4,7,9) G (a, b, c, d)= Σm (0,3,15) H(a, b, c, d)= Σm(0,2,10,12)

Program:

```
module dec2to4(W,En,Y);
input[1:0]W;
input En;
output [0:3]Y;
reg [0:3]Y;
always@(W or En)
begin
if(En==1)
case(W)
0: Y=4'b1000;
1: Y=4'b0100;
2: Y=4'b0010;
3: Y=4'b0001;
endcase
else
```

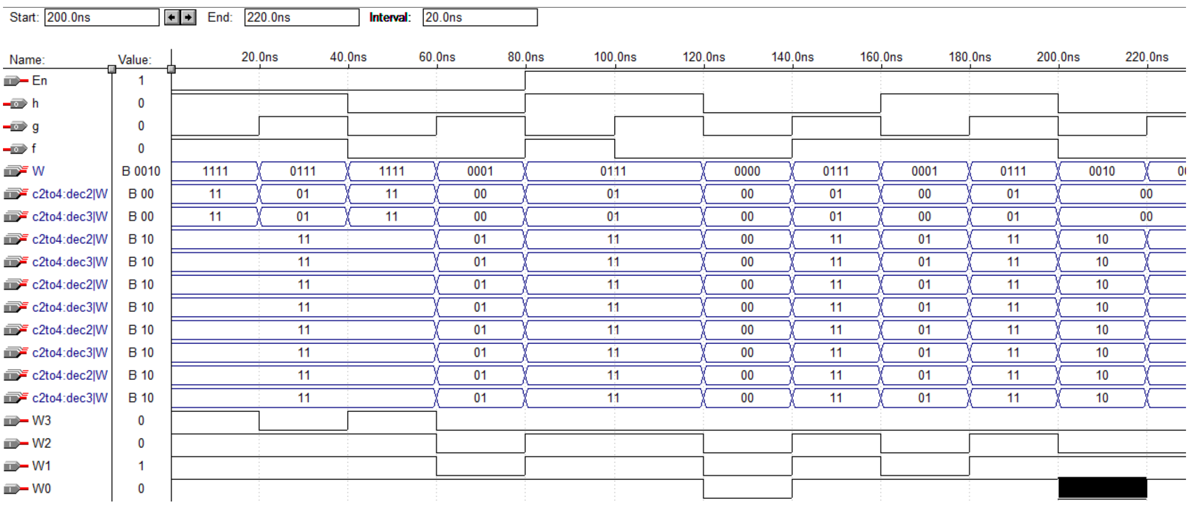
```
Y=4'b0000;
end
endmodule

module dec3to8(W,En,Y);
input [2:0]W;
input En;
output [0:7]Y;
wire[0:3]M;
dec2to4 dec1({1'b0, W[2]},En,M);
dec2to4 dec2(W[1:0],M[0],Y[0:3]);
dec2to4 dec3(W[1:0],M[1],Y[4:7]);
endmodule

module lab5_3(W, En, f, g, h);
input [3:0] W;
input En;
output f, g, h;
wire [0:7] temp0;
wire [0:7] temp1;
wire [0:7] temp2;
wire [0:7] temp3;
wire [0:7] temp4;
dec3to8 dec00({1'b0, W[3],W[2]}, En, temp0);
dec3to8 dec11({1'b0, W[1], W[0]}, temp0[0], temp1);
dec3to8 dec22({1'b0, W[1], W[0]}, temp0[1], temp2);
dec3to8 dec33({1'b0, W[1], W[0]}, temp0[2], temp3);
dec3to8 dec44({1'b0, W[1], W[0]}, temp0[3], temp4);

or(f, temp1[2], temp2[0], temp2[3], temp3[1]);
or(g, temp1[0], temp1[3], temp4[3]);
or(h, temp1[0], temp1[2], temp3[2], temp4[0]);
endmodule
```

Waveform:



WEEK 6

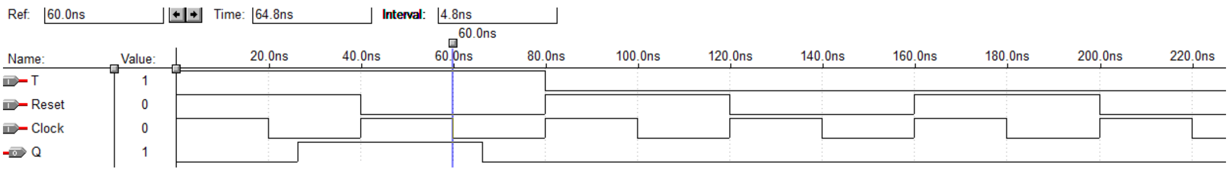
Lab Exercises:

1: Write behavioral Verilog code for a negative edge triggered TFF with asynchronous active low reset.

Program:

```
module lab6_1(T,Clock,Reset, Q);
input T,Clock, Reset;            output Q;
reg Q;
always @ (negedge Clock)
begin
if (T == 1)
Q <= ~Q;
if (Reset == 0)
Q <= 0;
end
endmodule
```

Waveform:



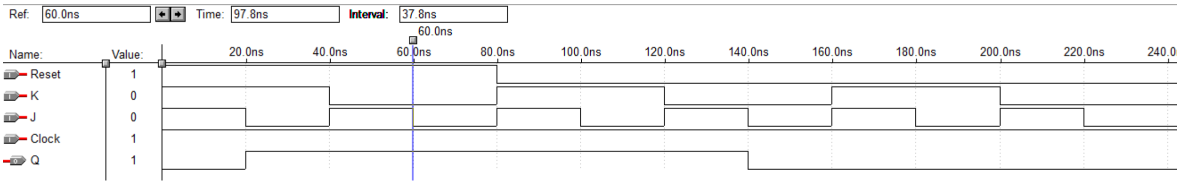
2: Write behavioural Verilog code for a positive edge-triggered JK FF with synchronous active highest.

Program:

```
module lab6_2(J, K, Clock, Reset, Q);
input J, K, Clock, Reset;
output Q;
reg Q;
always @ (posedge Clock or posedge Reset)
begin
```

```
case({J, K})
0: Q <= Q;
1: Q <= 0;
2: Q <= 1;
3: Q <= ~Q;
endcase
if(Reset == 1)
Q <= 0;
end
endmodule
```

Waveform:



3: Design and simulate the following counters:

a) 4-bit ring counter.

b) 5 bit Johnson counter.

Program a:

```
module DFlipFlop(D, clock,reset, q);
input D, clock, reset;
output q;
reg q;
always @ (posedge clock)
begin
if(reset)
q <= 0;
else if(D)
q <= 1;
else
q <= 0;
end
endmodule
```

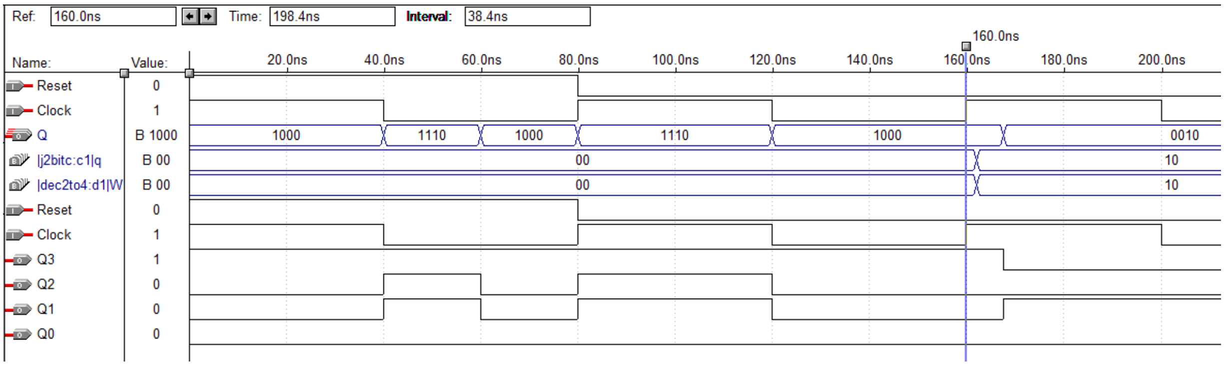
```
module j2bitc(clock, reset, q);
input clock, reset;
output [0:1]q;
wire [0:1]q;
DFlipFlop d1(~q[1], clock, reset, q[0]);
DFlipFlop d2(q[0], clock, reset, q[1]);
endmodule

module dec2to4(W,En,Y);
input[1:0]W;
input En;
output [0:3]Y; reg [0:3]Y;
always@(W or En)
begin
if(En==1)
case(W)
0: Y=4'b1000;
1: Y=4'b0100;
2:Y=4'b0010;
3:Y=4'b0001;
endcase
else
Y=4'b0000;
end
endmodule

module lab6_3_a(Clock, Reset, Q);
input Clock, Reset;
output [3:0] Q;
wire [3:0] Q;
wire [1:0] temp;
```

```
j2bitc c1(Clock, Reset, temp);
dec2to4 d1(temp, 1'b1, Q);
endmodule
```

Waveform a:

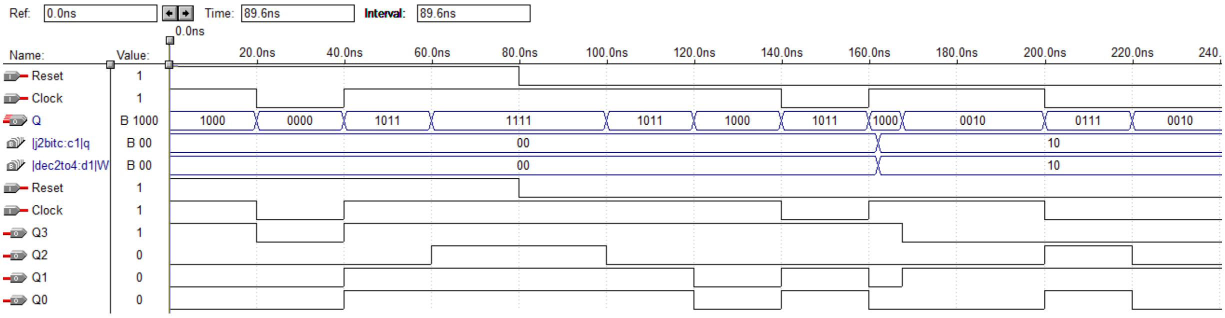


Program b:

```
module DFlipFlop(D, clock, reset, q);
input D, clock,reset;
output q;
reg q;
always @ (posedge clock)
begin
if(reset)
q <= 0;
else if(D)
q <= 1;
else
q <= 0;
end
endmodule

module lab6_3_b(clock, reset, q);
input clock, reset;
output [0:4]q;
wire [0:4]q;
DFlipFlop d1(~q[4], clock, reset, q[0]);
DFlipFlop d2(q[0], clock, reset, q[1]);
DFlipFlop d3(q[1], clock, reset, q[2]);
DFlipFlop d4(q[2], clock, reset, q[3]);
DFlipFlop d5(q[3], clock, reset, q[4]);
endmodule
```

Waveform b:


Ref: 0.0ns
Time: 89.6ns
Interval: 89.6ns
0.0ns
Name:
Value:
20.0ns
40.0ns
60.0ns
80.0ns
100.0ns
120.0ns
140.0ns
160.0ns
180.0ns
200.0ns
220.0ns
240.
Reset 1
Clock 1
Q B 1000
1000 0000 1011 1111 1011 1000 1011 1000 0010 0111 0010
|j2bitc:c1|q B 00
00 10
|dec2to4:d1|W B 00
00 10
Reset 1
Clock 1
Q3 1
Q2 0
Q1 0
Q0 0