DSD LAB

LAB 6

Q1)

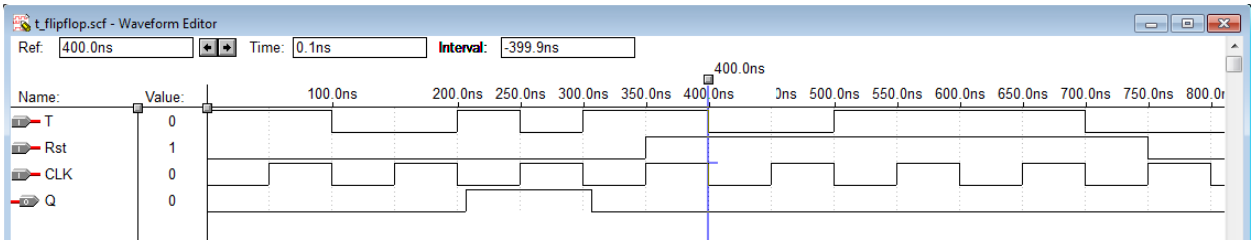
Source Code:

```
module t_flipflop(T, Rst, CLK, Q);
input T, CLK, Rst;
output Q;
reg Q;

always @(negedge CLK)
begin
if(T == 1)
Q <= ~Q;
if(Rst == 1)
Q <= 0;
end

endmodule
```

Output Waveform:



Q2)

Source Code:

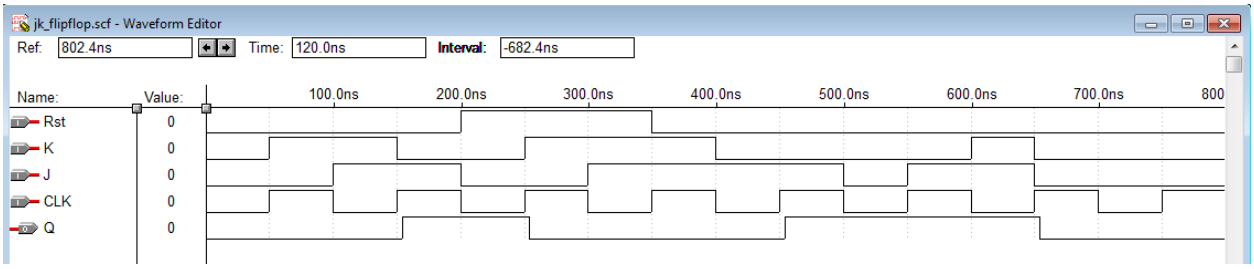
```
module jk_flipflop(J, K, CLK, Rst, Q);
input J, K, CLK, Rst;
output Q;
reg Q;

always @(posedge CLK)
begin
case({J, K})
0: Q <= Q;
1: Q <= 0;
2: Q <= 1;
3: Q <= ~Q;
```

endcase

if(Rst == 1)
Q <= 0;
end
endmodule

Output Waveform:



Q3)

a)

Source Code:

module bit4ring(clk, Rst, Q);
input clk;
input Rst;
output [3:0] Q;
wire [3:0] Q;
wire [1:0] Qt;

r2bitc r1(clk, Rst, Qt);
decoder2to4 d1(Qt, 1'b1, Q);

endmodule

module decoder2to4(W, En, Y);
input [1:0] W;
input En;
output [0:3] Y;
reg [0:3] Y;

always @(W | En)
begin
if(En == 1)
case(W)
0: Y = 4'b1000;

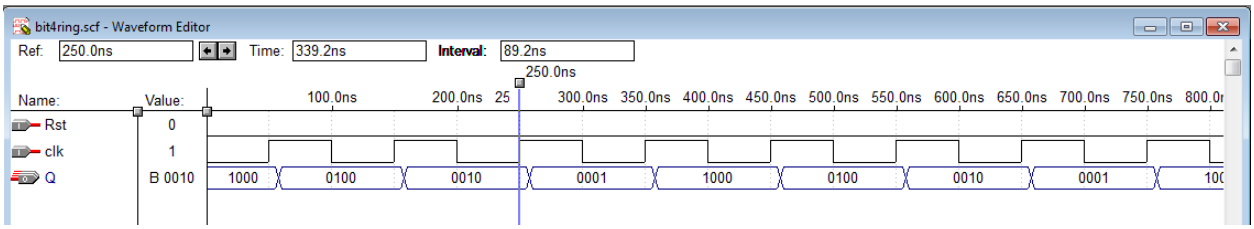
```
1: Y = 4'b0100;
2: Y = 4'b0010;
3: Y = 4'b0001;
endcase
else
Y = 4'b0000;
end
endmodule

module r2bitc(CLK,Rst,Q);
input CLK, Rst;
output [0:1] Q;
reg [0:1]Q;

always @(posedge CLK)
begin
Q <= Q + 1;
end

endmodule
```

Output Waveform:



b)

Source Code:

```
module johnson_ctr(CLK, Rst, Q);
parameter N = 5;
input CLK, Rst;
output [N-1:0] Q;
reg [N-1:0] Q;
integer i;

always @(posedge CLK)
begin
if(!Rst)
Q <= 1;
else
begin
Q[N-1] <= ~Q[0];
for(i = 0; i<N-1; i = i+1)
```

```
begin
Q[i] <= Q[i+1];
end
end
end
endmodule
```

Output Waveform: