VLSI System Design

ELE301P

LAB - 5 - Report

Praveen B R

COE19B007

Submission Date: 12/10/2021

INDEX

Q1) Flip-Flops

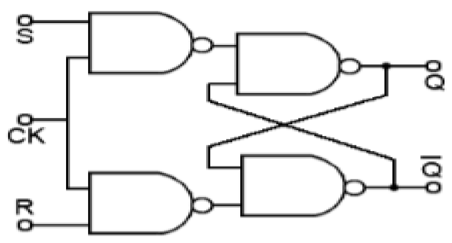
Objective:

To implement a SR, JK, D, T Flip-flops in Verilog using Behavioural Modelling

Theory:
Flip-flop is a circuit that maintains a state until directed by input to change the state. A basic flip-flop can be constructed using four-NAND or four-NOR gates.
Types of flip-flops:

- SR Flip Flop
- JK Flip Flop
- D Flip Flop
- T Flip Flop

SR Flip-flop:



TRUTH TABLE

S	R	Q_N	Q_{N+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	-
1	1	1	-

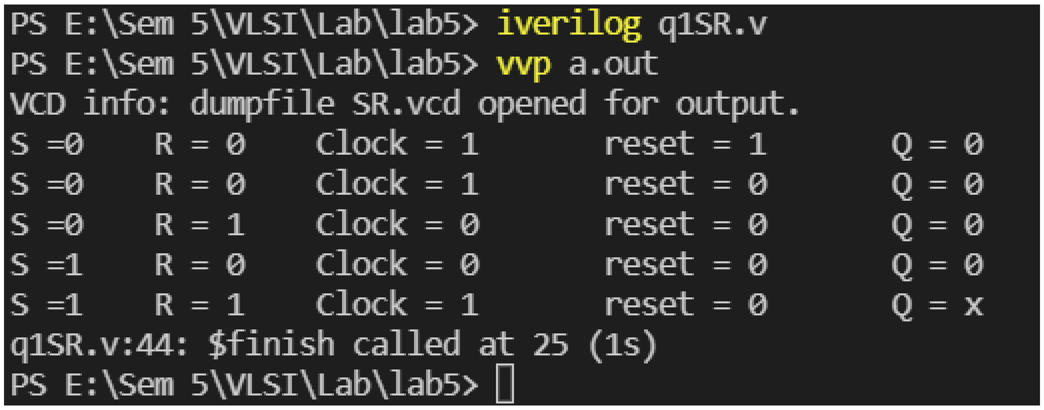
Code

```
module sr_flipflop(S,R,clock,reset,Q);
    input wire S,R,clock,reset;
    output reg Q;

    always @(posedge clock) begin
        if(reset) begin
            Q<= 1'b0;
        end
        else begin
```

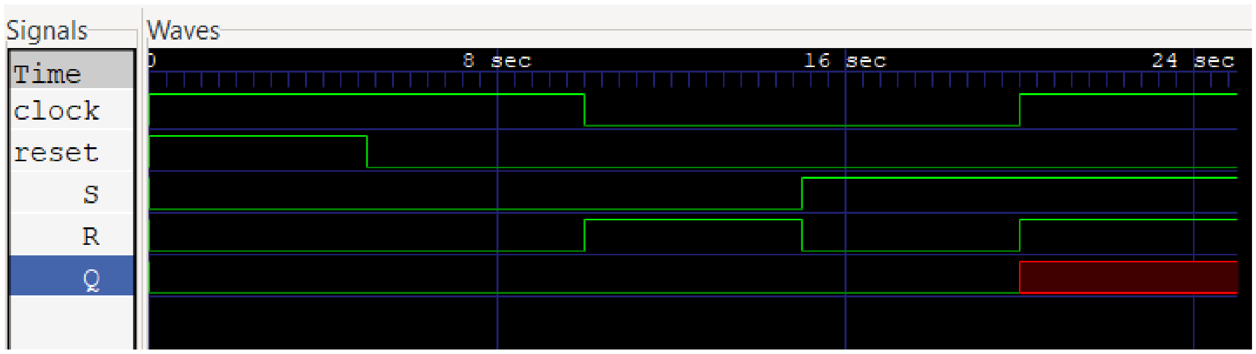
```
            case({S,R})
                2'b00: begin
                    Q<=Q;
                end
                2'b01: begin
                    Q<=1'b0;
                end
                2'b10: begin
                    Q<=1'b1;
                end
                default: begin
                    Q<=1'bx;
                end
            endcase
        end
    end
endmodule

module SR_ff_tb();

    reg S,R,clock,reset;
    wire Q;

    always begin
        #10 clock=~clock;
    end

    initial begin
        S=0;R=0;reset=1; clock=1;
        #20 S=0;R=0;reset=0;
        #20 S=0;R=1;reset=0;
        #20 S=1;R=0;reset=0;
        #20 S=1;R=1;reset=0;
        #20
    $finish;
    end

    initial begin
        $monitor("S =%b R = %b Clock = %b reset = %b Q =
%b",S,R,clock,reset,Q);
```

```
    end

    sr_flipflop SRFF1 (S,R,clock,reset,Q);

    initial begin
        $dumpfile("SRFF1.vcd");
        $dumpvars(0,SRFF1);
    end
endmodule;
```

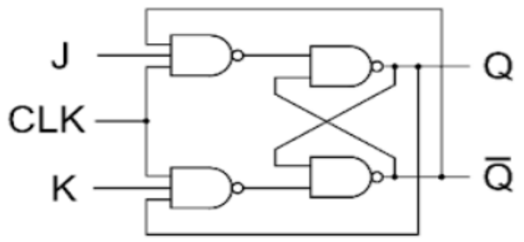
Output/Waveform



```
PS E:\Sem 5\VLSI\Lab\lab5> iverilog q1SR.v
PS E:\Sem 5\VLSI\Lab\lab5> vvp a.out
VCD info: dumpfile SR.vcd opened for output.
S =0    R = 0    Clock = 1       reset = 1       Q = 0
S =0    R = 0    Clock = 1       reset = 0       Q = 0
S =0    R = 1    Clock = 0       reset = 0       Q = 0
S =1    R = 0    Clock = 0       reset = 0       Q = 0
S =1    R = 1    Clock = 1       reset = 0       Q = x
q1SR.v:44: $finish called at 25 (1s)
PS E:\Sem 5\VLSI\Lab\lab5>
```



JK Flip-flop:



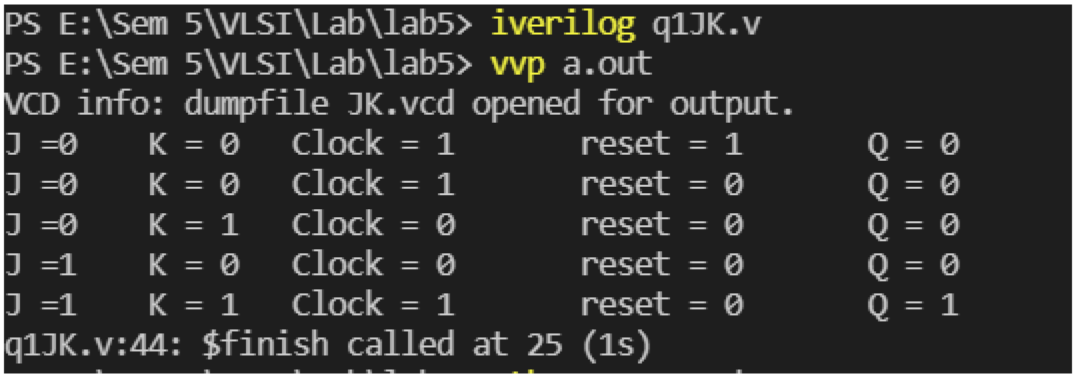
TRUTH TABLE

J	K	Q_N	Q_{N+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

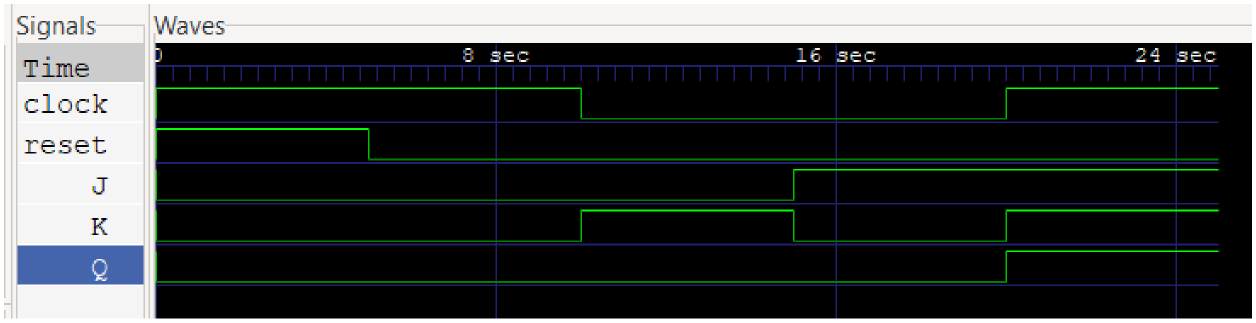
Code

```
module jk_flipflop(J,K,clock,reset,Q);
    input wire J,K,clock,reset;
    output reg Q;

    always @(posedge clock) begin
        if(reset) begin
            Q<= 1'b0;
        end
        else begin
            case({J,K})
                2'b00: begin
                    Q<=Q;
                end
                2'b01: begin
                    Q<=1'b0;
                end
                2'b10: begin
                    Q<=1'b1;
                end
                2'b11: begin
                    Q<=~Q;
                end
            endcase
        end
    end
endmodule

module JK_ff_tb();

    reg J,K,clock,reset;
    wire Q;

    always begin
        #10 clock=~clock;
    end

    initial begin
        J=0;K=0;reset=1; clock=1;
        #5 J=0;K=0;reset=0;
        #5 J=0;K=1;reset=0;
        #5 J=1;K=0;reset=0;
        #5 J=1;K=1;reset=0;
        #5
    $finish;
    end
    initial begin
        $monitor("J =%b \tK = %b \tClock = %b \treset = %b \tQ =
%b",J,K,clock,reset,Q);
    end

    jk_flipflop JKFF (J,K,clock,reset,Q);

    initial begin
        $dumpfile("JK.vcd");
        $dumpvars(0,JKFF);
    end
endmodule;
```

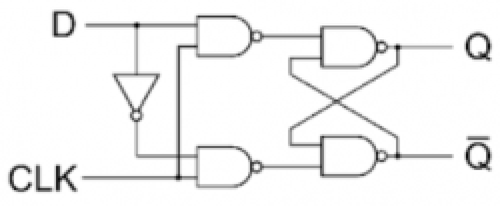
Output/Waveform



```
PS E:\Sem 5\VLSI\Lab\lab5> iverilog q1JK.v
PS E:\Sem 5\VLSI\Lab\lab5> vvp a.out
VCD info: dumpfile JK.vcd opened for output.
J =0    K = 0   Clock = 1       reset = 1       Q = 0
J =0    K = 0   Clock = 1       reset = 0       Q = 0
J =0    K = 1   Clock = 0       reset = 0       Q = 0
J =1    K = 0   Clock = 0       reset = 0       Q = 0
J =1    K = 1   Clock = 1       reset = 0       Q = 1
q1JK.v:44: $finish called at 25 (1s)
```



D Flip-flop:



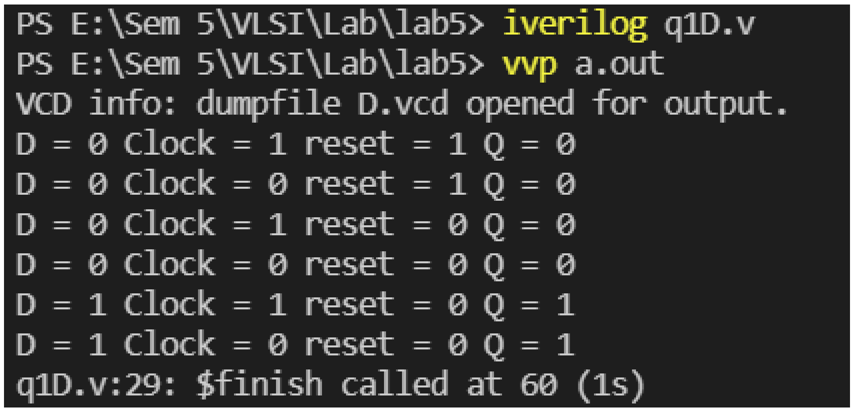
Q	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

Code

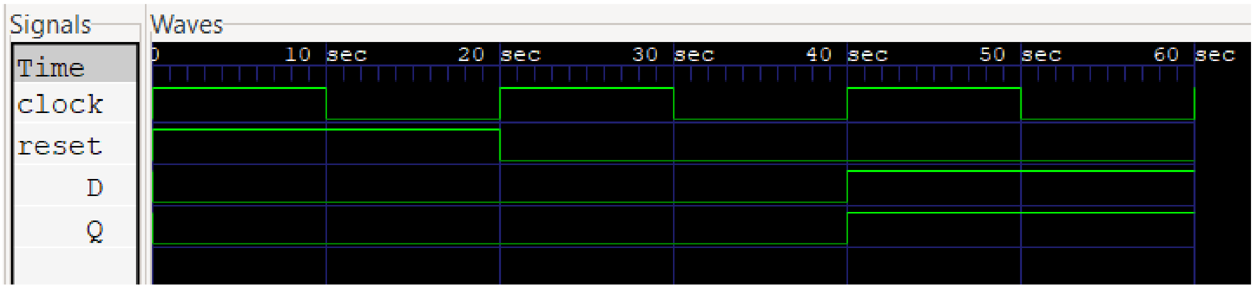
```
module d_flipflop(D,clock,reset,Q);
    input wire D,clock,reset;
    output reg Q;

    always @(posedge clock) begin
```

```
        if(reset) begin
            Q<=1'b0;
        end
        else begin
            Q<=D;
        end
    end
endmodule

module D_ff_tb();

    reg D,clock,reset;
    wire Q;

    always begin
        #10 clock=~clock;
    end

    initial begin
        D=0;reset=1; clock=1;
        #20 D=0;reset=0;
        #20 D=1;reset=0;
        #20
    $finish;
    end

    initial begin
        $monitor("D = %b Clock = %b reset = %b Q = %b",D,clock,reset,Q);
    end

    d_flipflop DFF1 (D,clock,reset,Q);

    initial begin
        $dumpfile("DFF1.vcd");
        $dumpvars(0,DFF1);
    end
endmodule;
```

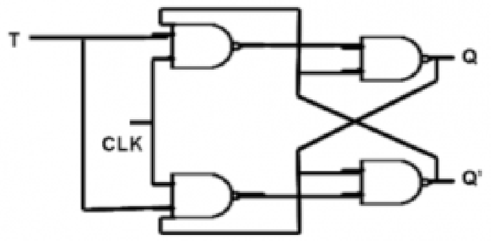
Output/Waveform

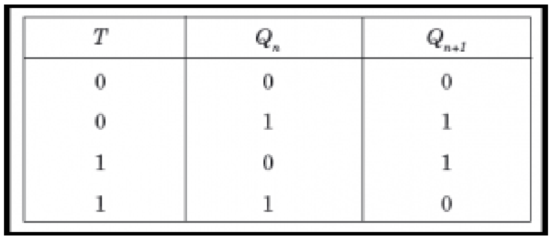


```
PS E:\Sem 5\VLSI\Lab\lab5> iverilog q1D.v
PS E:\Sem 5\VLSI\Lab\lab5> vvp a.out
VCD info: dumpfile D.vcd opened for output.
D = 0 Clock = 1 reset = 1 Q = 0
D = 0 Clock = 0 reset = 1 Q = 0
D = 0 Clock = 1 reset = 0 Q = 0
D = 0 Clock = 0 reset = 0 Q = 0
D = 1 Clock = 1 reset = 0 Q = 1
D = 1 Clock = 0 reset = 0 Q = 1
q1D.v:29: $finish called at 60 (1s)
```



T Flip-flop:



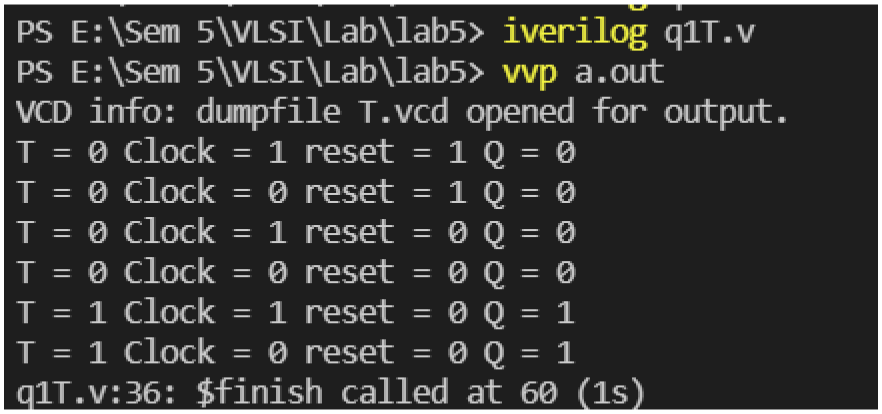


T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

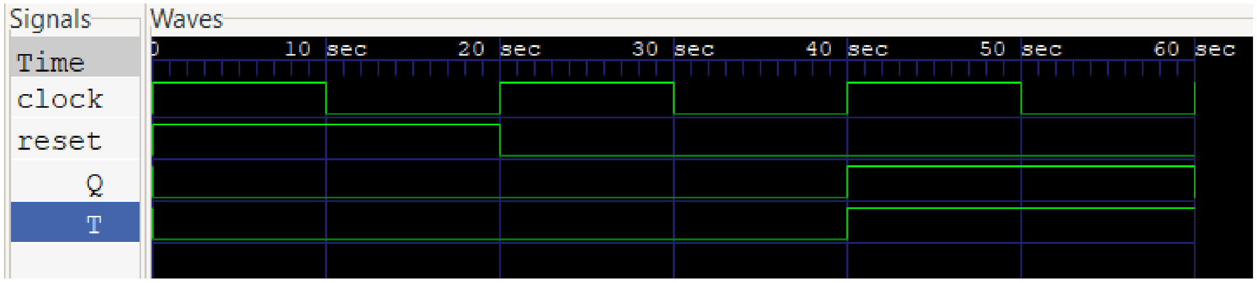
Code

```
module t_flipflop(T,clock,reset,Q);
    input wire T,clock,reset;
    output reg Q;
    always @(posedge clock) ebeegin
        if(rst) begin
            Q<=1'b0;
        end
        else begin
            case(T)
                1'b0: begin
                    Q<=Q;
                end
                1'b1: begin
                    Q<=~Q;
                end
            endcase
        end
    end
endmodule

module T_ff_tb();

    reg T,clock,reset;
    wire Q;

    always begin
        #10 clock=~ecelock;
e   eend
    initial begin
        T=0;rst=1; clock=1e;e
        #20 T=0;rst=0;
        #20 T=1;rst=0;
        #20
    $finish;
    end

    initial begin
        $monitor("T = %b Clock = %b rst = %b Q = %b",T,clock,reset,Q);
    end

    t_flipflop TFF (T,clock,reset,Q);

    initial begin
        $dumpfile("T.vcd");
        $dumpvars(0,TFF);
    end
endmodule;
```

Output/Waveform



```
PS E:\Sem 5\VLSI\Lab\lab5> iverilog q1T.v
PS E:\Sem 5\VLSI\Lab\lab5> vvp a.out
VCD info: dumpfile T.vcd opened for output.
T = 0 Clock = 1 reset = 1 Q = 0
T = 0 Clock = 0 reset = 1 Q = 0
T = 0 Clock = 1 reset = 0 Q = 0
T = 0 Clock = 0 reset = 0 Q = 0
T = 1 Clock = 1 reset = 0 Q = 1
T = 1 Clock = 0 reset = 0 Q = 1
q1T.v:36: $finish called at 60 (1s)
```



<u>Conclusion:</u>
Thus, flip-flops have been implemented successfully in verilog

<u>Application:</u>
These are the various types of flip-flops being used in digital electronic circuits and the applications of Flip-flops are as specified below.
Counters

- Frequency Dividers
- Shift Registers
- Storage Registers
- Bounce elimination switch
- Data storage
- Data transfer
- Latch
- Registers
- Memory

Q2) Mod 9 Counter

Objective:
To implement a Mod 9 counter in verilog using Structural Modelling

Theory:

Counter is a sequential circuit. A digital circuit which is used for a counting pulses is known counter. Counter is the widest application of flip-flops. It is a group of flip-flops with a clock signal applied. Counters are of two types.

- Asynchronous or ripple counters.
- Synchronous counters.

For designing Mod 9 Counter I chose to use T Flip Flop.
Excitation Table of T Flip Flop

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

State Input Table for mod 9 counter

Present State				Next State				Input			
P_4	P_3	P_2	P_1	N_4	N_3	N_2	N_1	T_4	T_3	T_2	T_1
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	1
0	1	0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	1	0	0	0

Now using the input values and draw k-maps,

	00	10	11	10
00				
10			1	
11	x	x	x	x
10	1	x	x	x

$T_4 = Q_4 + Q_3.Q_2.Q_1$

	00	10	11	10
00			1	
10			1	
11	x	x	x	x
10		x	x	x

$T_3 = Q_2.Q_1$

	00	10	11	10
00		1	1	
10		1	1	
11	x	x	x	x
10		x	x	x

$T_2 = Q_1$

	00	10	11	10
00	1	1	1	1
10	1	1	1	1
11	x	x	x	x
10		x	x	x

$T_1 = Q'_4$

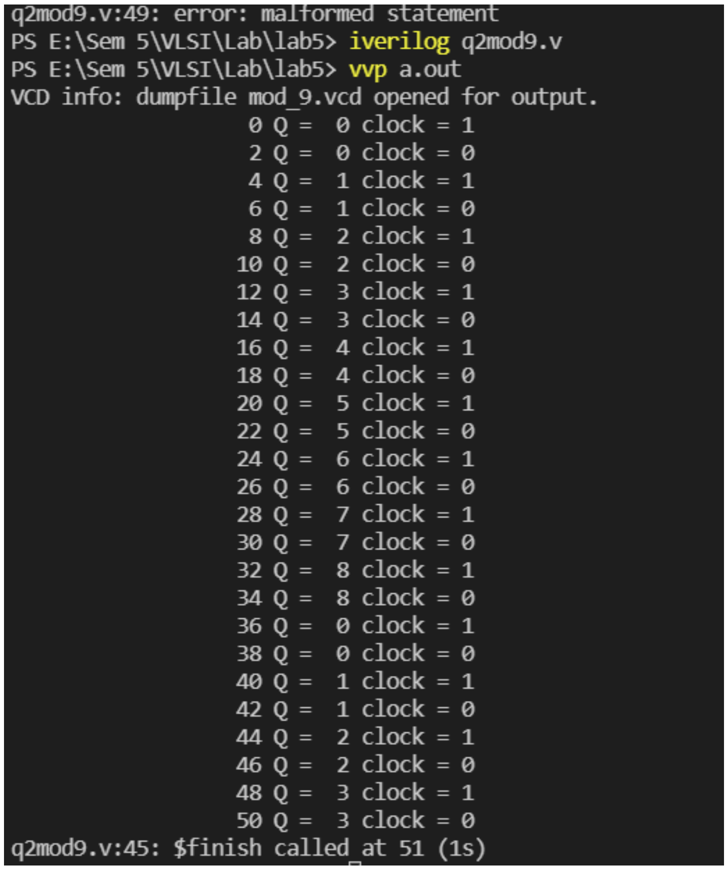
Then implement the obtained equations in verilog structural style

Code:

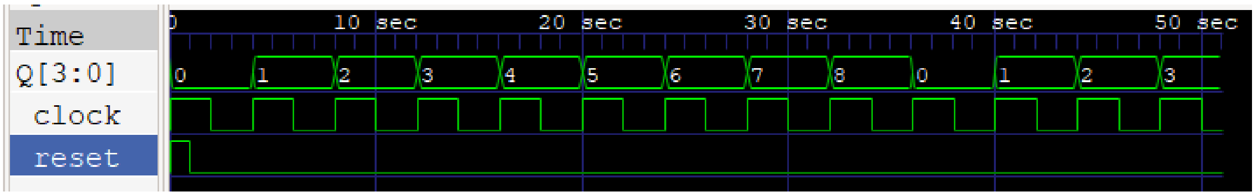
```
module t_flipflop(T,clock,reset,Q);
input wire T,clock,reset;
    output reg Q;

    always @(posedge clock) begin
        if(reset) begin
            Q<=1'b0;
        end
        else begin
            case(T)
                1'b0: begin
                    Q<=Q;
                end
                1'b1: begin
                    Q<=~Q;
                end
            endcase
        end
    end
endmodule
```

```
module mod9_Counter(reset,Q,clock);
input clock,reset;
output [3:0] Q;
wire t1,t2,t3;
and(t1,Q[0],Q[1]);
and(t2,t1,Q[2]);
or(t3,t2,Q[3]);

t_flipflop T_1(~Q[3],clock,reset,Q[0]);
t_flipflop T_2(Q[0],clock,reset,Q[1]);
t_flipflop T_3(t1,clock,reset,Q[2]);
t_flipflop T_4(t3,clock,reset,Q[3]);
endmodule

module mod9_tb;
reg clock,reset;
wire [3:0] Q;
mod9_Counter M1(reset,Q,clock);
initial
begin
    #0 clock = 1;reset = 1;
    #1 reset = 0;
    #50 $finish;
end
always
begin
    #2 clock = ~clock;
end
initial
begin
    $monitor($time," \tQ = %d \tclock = %b",Q,clock);
end
initial
begin
$dumpfile("mod_9.vcd");
$dumpvars(0,mod9_tb);
end
endmodule
```

Output/Waveform:



```
q2mod9.v:49: error: malformed statement
PS E:\Sem 5\VLSI\Lab\lab5> iverilog q2mod9.v
PS E:\Sem 5\VLSI\Lab\lab5> vvp a.out
VCD info: dumpfile mod_9.vcd opened for output.
                  0 Q =  0 clock = 1
                  2 Q =  0 clock = 0
                  4 Q =  1 clock = 1
                  6 Q =  1 clock = 0
                  8 Q =  2 clock = 1
                 10 Q =  2 clock = 0
                 12 Q =  3 clock = 1
                 14 Q =  3 clock = 0
                 16 Q =  4 clock = 1
                 18 Q =  4 clock = 0
                 20 Q =  5 clock = 1
                 22 Q =  5 clock = 0
                 24 Q =  6 clock = 1
                 26 Q =  6 clock = 0
                 28 Q =  7 clock = 1
                 30 Q =  7 clock = 0
                 32 Q =  8 clock = 1
                 34 Q =  8 clock = 0
                 36 Q =  0 clock = 1
                 38 Q =  0 clock = 0
                 40 Q =  1 clock = 1
                 42 Q =  1 clock = 0
                 44 Q =  2 clock = 1
                 46 Q =  2 clock = 0
                 48 Q =  3 clock = 1
                 50 Q =  3 clock = 0
q2mod9.v:45: $finish called at 51 (1s)
```



Conclusion:
Thus a mod 9 counter has been implemented successfully

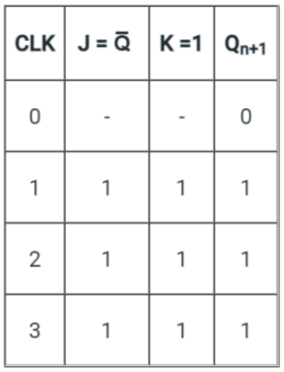
Applications:

- Alarm Clock
- Set AC Timer
- Set time in camera to take the picture
- Flashing light indicator in automobiles
- Car parking control

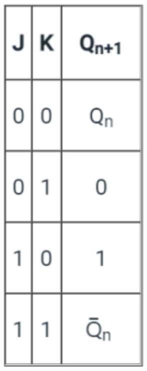
Other Questions

Question 1

J = Qn’ and K=1
Output table will be and characteristic table of JK flip flop will be:



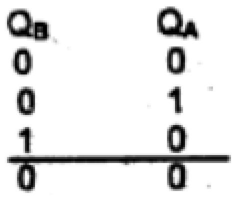
CLK	$J = \bar{Q}$	K =1	Q_{n+1}
0	-	-	0
1	1	1	1
2	1	1	1
3	1	1	1



J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\bar{Q}_n$

With this, we can say that the sequence will be..
1st J = 1 , K = 1 -> Q = 1,Q' = 0
2nd J = 0, K = 1 -> Q = 0,Q' = 1
3rd J = 1, K = 1 -> Q = 1,Q' = 0
4th J = 0, K = 1 -> Q = 0,Q' = 1
5th J = 1, K = 1 -> Q = 1,Q' = 0
6th J = 0, K = 1 -> Q = 0,Q' = 1

Question 2



Q_B	Q_A
0	0
0	1
1	0
0	0

So, it has to be a mod 3 counter

Question 3

Applications of Flip Flops
- Registers
- Memory
- Data storage
- Data transfer

Applications of Counters
- Frequency counters
- Digital clock
- Time measurement