ASSIGNMENT – 8
NAME- HIMANSHU KUMAR SINGH
ENROLMENT NO – S24CSEU0721
BATCH-43

SOLUTION 1-
A flip-flop is a memory element used in digital electronics to store one bit of data (0 or 1).
It is edge-triggered, meaning it updates its output only on a specific edge of the clock signal (usually the rising edge ↑ or falling edge ↓).
DIFFERENCE BETWEEN THEM-

Flip-Flop	Latch
Edge-triggered (on clock edge ↑ or ↓)	**Level-sensitive** (when Enable = 1)
Requires **clock signal**	Controlled by **enable or gate signal**
Changes output only on edge of clock	Changes output **any time enable = 1**
Synchronous circuits	**Asynchronous** or gated circuits
D Flip-Flop, JK Flip-Flop	SR Latch, D Latch
More complex (needs clocking logic)	Simpler design

SOLUTION 2-

An SR (Set-Reset) flip-flop is a basic memory element that stores one bit (0 or 1) of data. It has:

1. S (Set) input: makes the output Q = 1
2. R (Reset) input: makes the output Q = 0
3. Clock input (in edge-triggered versions): tells the flip-flop when to update

Role of SR Flip-Flop in Digital Circuits

1. Data Storage

Stores one bit of information (either 0 or 1)

Output remains constant until a new input is triggered

2. Control Logic / State Machines

Used to build sequential circuits that move between states

Example: in traffic lights, vending machines, etc.

3. Edge Detection / Event Capture

Can detect events like button presses or signal changes

Captures a signal only when triggered by a clock

4. Debouncing Mechanical Switches

SR flip-flop can help stabilize the input from bouncy mechanical switches

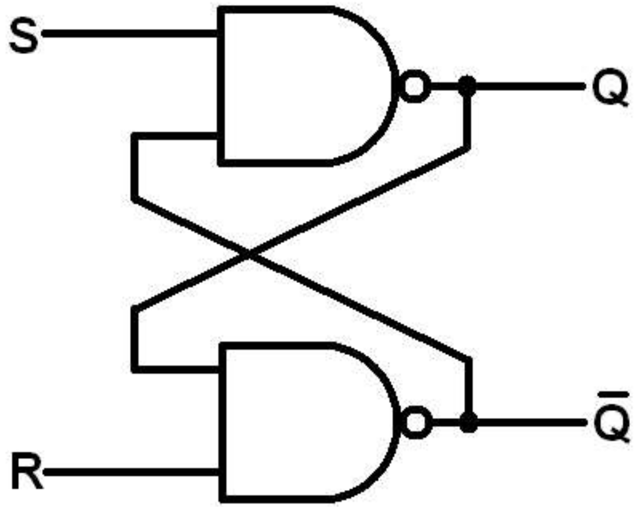
Only one clean ON/OFF signal is registered

5.Part of Larger Memory Structures

SR flip-flops are building blocks for registers, RAM, counters, and more

Combine multiple flip-flops to store bytes or word

SR Flip Flop



Sno	S	R	Q	Q’	State
1	1	0	1	0	Q is set to 1
2	1	1	1	0	No change
3	0	1	0	1	Q’ is set to 1
4	1	1	0	1	No change
5	0	0	1	1	Invalid

TEST BENCH AND DESIGN CODE-

testbench.v.txt design.v.txt

File Edit View

```
module A8Q2Gate(input A, S, R, output reg F, output reg valid);
    always @(*) begin
        if (S && R) begin
            F = 1'bx;
            valid = 1;
        end
        else if (S) begin
            F = 1;
            valid = 0;
        end
        else if (R) begin
            F = 0;
            valid = 0;
        end
        else begin
            F = A;
            valid = 0;
        end
    end
endmodule
```

testbench.v.txt design.v.txt

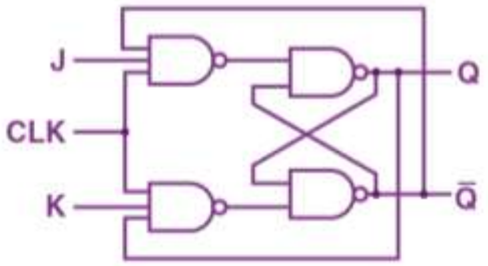
File Edit View

```
 module A8Q2_gate;

reg A, S, R;
wire F, valid;

A8Q2Gate uut(.A(A), .S(S), .R(R), .F(F), .valid(valid));

initial begin
        $dumpfile("A8Q2Test.vcd");
        $dumpvars(1);
        end

initial begin
        $monitor("A=%b Set=%b Reset=%b | F=%b valid=%b", A, S, R, F, valid);
                A = 0; S = 0; R = 0;
                #10
                A = 0; S = 0; R = 1;
                #10
                A = 0; S = 1; R = 0;
                #10
                A = 0; S = 1; R = 1;
                #10
                A = 1; S = 0; R = 0;
                #10
                A = 1; S = 0; R = 1;
                #10
                A = 1; S = 1; R = 0;
                #10
                A = 1; S = 1; R = 1;
                #10
                $finish();
                end
endmodule
```

OUTPUT AND GTK WAVE-

```
C:\iverilog\bin>.\vvp testmodule
VCD info: dumpfile A8Q2Test.vcd opened for output.
A=0 Set=0 Reset=0 | F=0 valid=0
A=0 Set=0 Reset=1 | F=0 valid=0
A=0 Set=1 Reset=0 | F=1 valid=0
A=0 Set=1 Reset=1 | F=x valid=1
A=1 Set=0 Reset=0 | F=1 valid=0
A=1 Set=0 Reset=1 | F=0 valid=0
A=1 Set=1 Reset=0 | F=1 valid=0
A=1 Set=1 Reset=1 | F=x valid=1
```

SOLTUION 3-A JK flip-flop is an edge-triggered flip-flop that solves the invalid state problem of an SR flip-flop.
It has two inputs:
J (Set input)
K (Reset input)
and one Clock input.



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

J = 0, K = 0 → No change
J = 0, K = 1 → Reset Q to 0
J = 1, K = 0 → Set Q to 1
J = 1, K = 1 → Toggle output: if Q = 1 → Q = 0, and vice versa
The key feature: toggle on J = K = 1, which solves the invalid state in SR flip-flop.

DESIGN AND TESTBENCH CODE-

testbench.v.txt | design.v.txt

File Edit View

```
module jkflipflop (
    input wire clk,
    input wire J,
    input wire K,
    output reg Q
);
    always @(posedge clk) begin
        case ({J, K})
            2'b00: Q <= Q;
            2'b01: Q <= 0;
            2'b10: Q <= 1;
            2'b11: Q <= ~Q;
        endcase
    end
endmodule
```

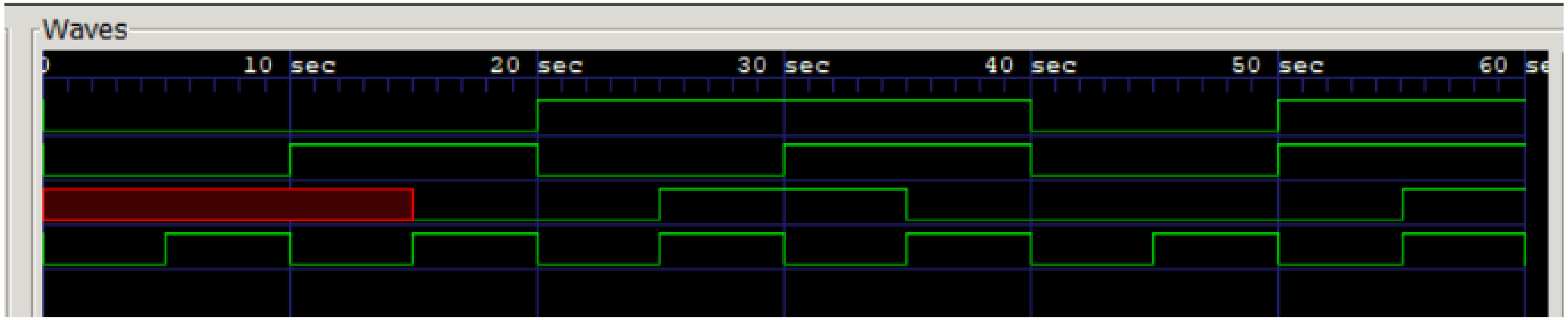
testbench.v.txt | design.v.txt

File Edit View

```
module tb_jk_flipflop;
    reg clk, J, K;
    wire Q;


    jkflipflop uut (
        .clk(clk),
        .J(J),
        .K(K),
        .Q(Q)
    );

    initial begin
    $dumpfile("jk_ff.vcd");
    $dumpvars(0, tb_jk_flipflop);
end

    initial begin
        clk = 0;
        forever #5 clk = ~clk;
    end


    initial begin
        $display("Time\tJ K Q");
        $monitor("%0dns\t%b %b %b", $time, J, K, Q);


        J = 0; K = 0;


        #10 J = 0; K = 1;
        #10 J = 1; K = 0;
        #10 J = 1; K = 1;
        #10 J = 0; K = 0;
        #10 J = 1; K = 1;
        #10 $finish;
    end
endmodule
```

Output and gtkwave-

```
C:\iverilog\bin>.\vvp testmodule
Time     J K Q
0ns      0 0 x
10ns     0 1 x
15ns     0 1 0
20ns     1 0 0
25ns     1 0 1
30ns     1 1 1
35ns     1 1 0
40ns     0 0 0
50ns     1 1 0
55ns     1 1 1
```



SOLUTION 4-

A D (Data or Delay) Flip-Flop is a clocked memory device that captures the input D at the rising edge (or falling edge) of the clock and holds it until the next clock edge

To make JK behave like D:

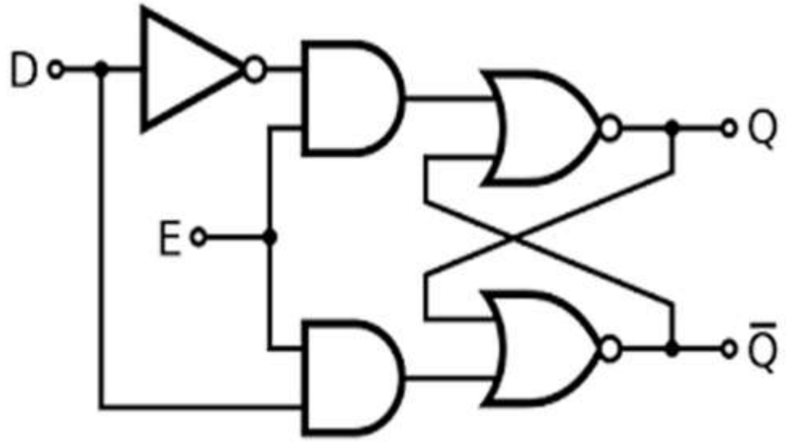
J = D

K = $\overline{D}$ (NOT D)

This way, the JK flip-flop sets (Q = 1) when D = 1 and resets (Q = 0) when D = 0 — just like a D flip-flop.

TRUTH TABLE-

D Flip Flop Circuit



D	S	R	Q	State
	0	0	Previous State	No Change
0	0	1	0	Reset
1	1	0	1	Set
	1	1	?	Forbidden

SR & D Flip Flop TruthTable

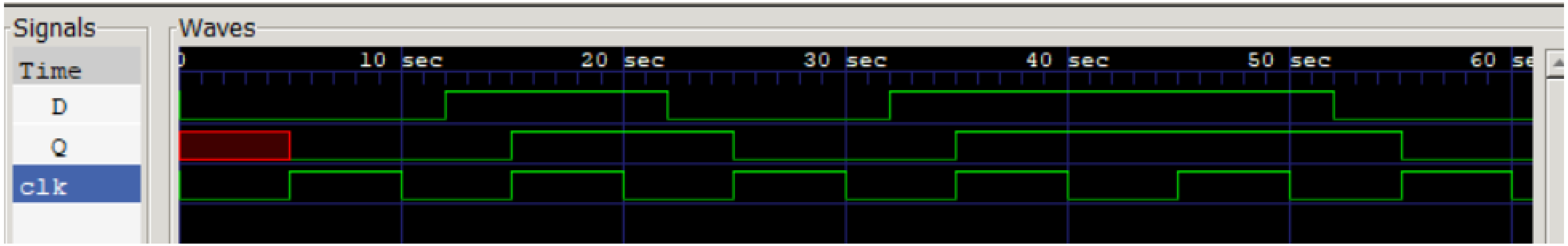
TESTBENCH AND DESIGN CODE-

```
module tb_d_from_jk;
    reg clk, D;
    wire Q;

    d_from_jk uut (
        .clk(clk),
        .D(D),
        .Q(Q)
    );

    initial begin
        $dumpfile("d_from_jk.vcd");
        $dumpvars(0, tb_d_from_jk);

        clk = 0;
        forever #5 clk = ~clk;
    end

    initial begin
        $display("Time\tD Q");
        $monitor("%0dns\t%b %b", $time, D, Q);

        D = 0; #10;
        D = 1; #10;
        D = 1; #10;
        D = 0; #10;
        D = 1; #10;
        D = 0; #10;
        $finish;
    end
endmodule
```

```
module d_from_jk (
    input wire clk,
    input wire D,
    output wire Q
);
    wire J, K;

    assign J = D;
    assign K = ~D;

    jk_flipflop jk (
        .clk(clk),
        .J(J),
        .K(K),
        .Q(Q)
    );
endmodule
```

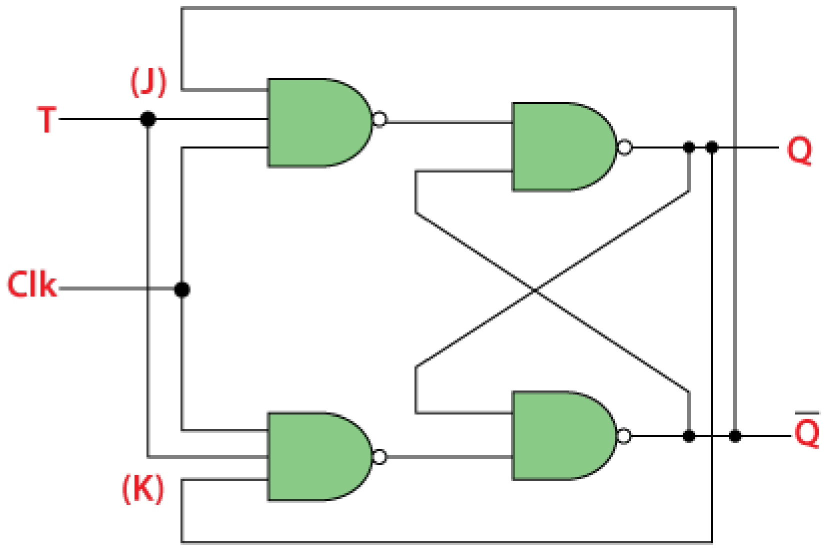
Output and gtkwave-

```
C:\iverilog\bin>.\vvp
VCD info: dumpfile d_
Time     D Q
0ns      0 x
5ns      0 0
12ns     1 0
15ns     1 1
22ns     0 1
25ns     0 0
32ns     1 0
35ns     1 1
52ns     0 1
55ns     0 0
```



SOLUTION 5-

A T (Toggle) flip-flop changes its output state (Q) on the rising edge of the clock if T = 1.
If T = 0, it holds the previous state.



TESTBENCH AND DESIGN CODE-

testbench.v.txt design.v.txt

File Edit View

```
module tb_t_flipflop;
    reg clk;
    reg T;
    wire Q;

    t_flipflop uut (
        .clk(clk),
        .T(T),
        .Q(Q)
    );

    initial begin
        clk = 0;
        forever #5 clk = ~clk;
    end

    initial begin
        $dumpfile("t_flipflop.vcd");
        $dumpvars(0, tb_t_flipflop);

        $display("Time\tT Q");
        $monitor("%0dns\t%b %b", $time, T, Q);

        T = 0; #12;
        T = 1; #10;
        T = 1; #10;
        T = 0; #10;
        T = 1; #10;
        T = 0; #10;
        T = 1; #10;
        $finish;
    end
endmodule
```

testbench.v.txt design.v.txt

File Edit View

```
module t_flipflop (
    input wire clk,
    input wire T,
    output reg Q
);
    always @(posedge clk) begin
        if (T)
            Q <= ~Q;
        else
            Q <= Q;
    end
endmodule
```

Output and gtkwave-

```
C:\iverilog\bin>.\vvp testmodule
VCD info: dumpfile t_flipflop.vcd opened for output.
Time    T Q
0ns     0 x
12ns    1 x
32ns    0 x
42ns    1 x
52ns    0 x
62ns    1 x
```