[redacted]

DESIGN AND SIMULATE SR, JK, D & T FLIP-FLOPS

INTRODUCTION TO FLIP-FLOPS

- Flip-flops are fundamental building blocks of digital electronics systems used in computers, communications, and many other types of systems.
- Flip-flops and latches are used as data storage elements. It is the basic storage element in sequential logic circuits.
- A flip flop is an digital circuit with two stable states that can be used to store binary data. The stored data can be changed by applying varying inputs.
- Universal logic gate NAND and NOR can be used to implement flip-flop.

INTRODUCTION TO FLIP-FLOPS

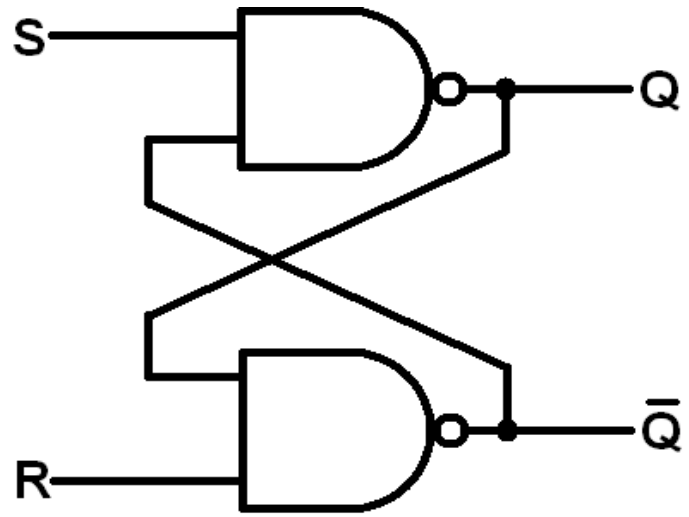
Applications of Flip-flops

- Data storage device
- Used as registers in Microprocessors/ microcontrollers
- For data transfer applications
- To design counters
- Used in frequency divider circuits

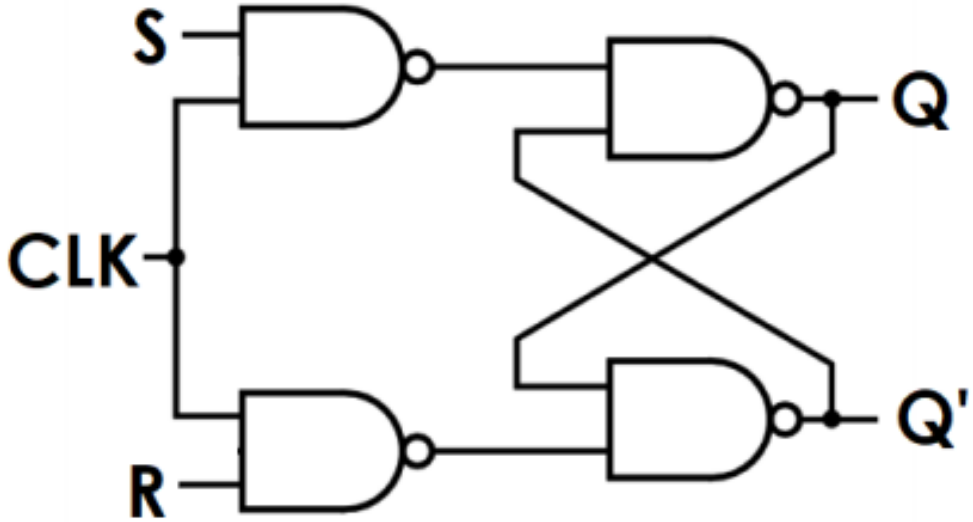
INTRODUCTION TO FLIP-FLOPS

- The basic difference between a latch and a flip-flop is a gating or clocking mechanism.
- A flip flop, on the other hand, is synchronous and is also known as gated or clocked SR latch.

SR LATCH



SR FLIP-FLOP

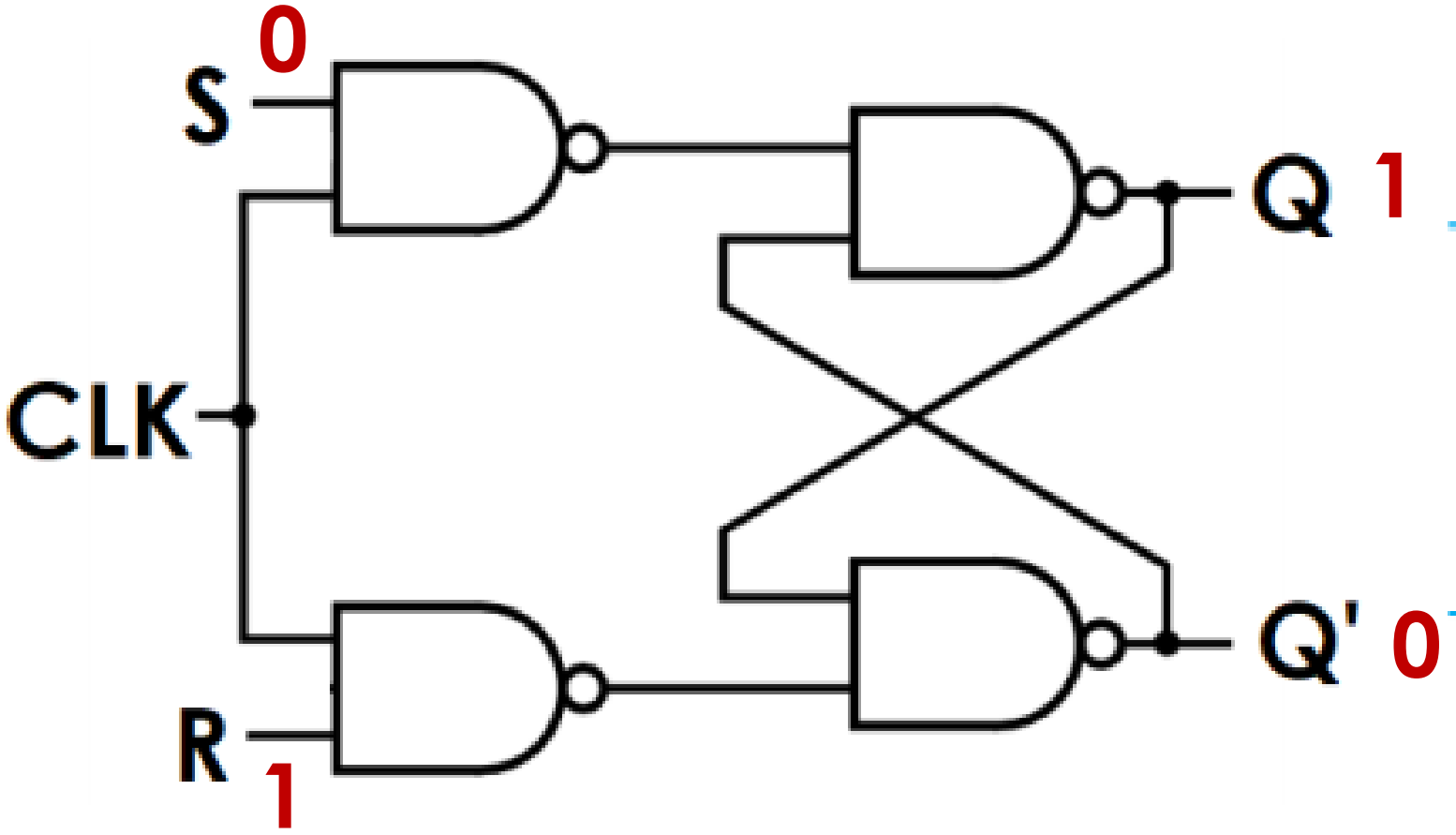


SR FLIP-FLOP

- The SR flip-flop, stands for “Set-Reset” flip-flop. This simple flip-flop is basically a one-bit memory bistable device.
- One which will “SET” the device (meaning the output Q = “1”), and is labelled S and one which will “RESET” the device (meaning the output Q = “0”), labelled R.
- A basic NAND gate SR flip-flop circuit provides feedback from both of its outputs back to its opposing inputs and is commonly used in memory circuits to store a single data bit.
- When S and R at HIGH state both outputs tries to get into HIGH state and not of them get into state output state. This state is called intermediate or invalid state.

SR FLIP-FLOP

LOGIC DIAGRAM



En	*S*	*R*	Next state of *Q*
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$; reset state
1	1	0	$Q = 1$; set state
1	1	1	Indeterminate

SR FLIP-FLOP

VERILOG HDL CODE (Behavioural)

```
module SR_FF(Q,QB,S,R,CLK);
        input S,R,CLK;
        output Q,QB;
        reg Q,QB;
        always @(posedge CLK)
        begin
                case({S,R})
                2'b00:Q=Q;
                2'b01:Q=0;
                2'b10:Q=1;
                2'b11:Q=1'bx;
                endcase
        QB=~Q;
        end
endmodule
```

SR FLIP-FLOP

TEST BENCH

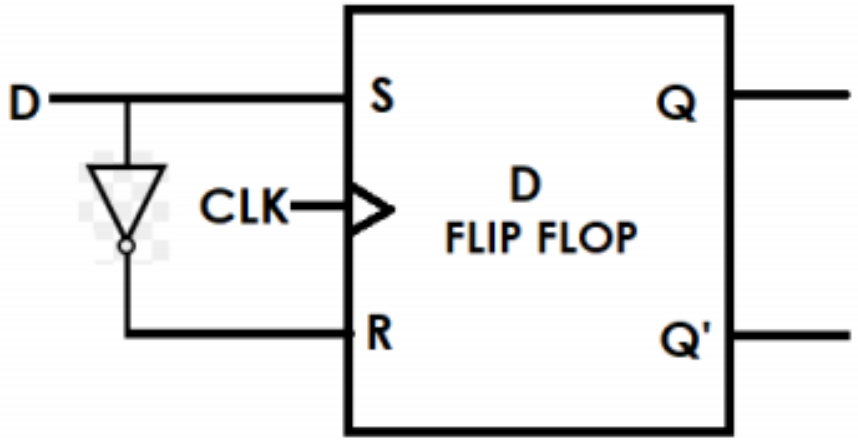
```
module SR_FF_TB;
        reg S,R,CLK;
        wire Q,QB;
        SR_FF uut (.Q(Q), .QB(QB), .S(S), .R(R), .CLK(CLK) );
        always #100 CLK=~CLK;
        initial begin
                CLK=1;
                #200 S=1; R=0;
                #200 S=0; R=0;
                #200 S=0; R=1;
                #200 S=1; R=1;
        end
endmodule
```

D FLIP-FLOP

- This flip-flop, called a Data flip-flop because of its ability to 'latch' and remember data, or a Delay flip-flop because latching and remembering data can be used to create a delay in the progress of that data through a circuit. This flip flop is also called transparent flip-flip due to output is same as input.
- A D flip–flop is constructed by modifying an SR flip – flop. The S input is given with D input and the R input is given with inverted D input. Hence, there will be no chance of intermediate state.
- If the clock signal is high (rising edge to be more precise) and if D input is high, then the output is also high and if D input is low, then the output will become low. Hence the output Q follows the input D in the presence of clock signal.

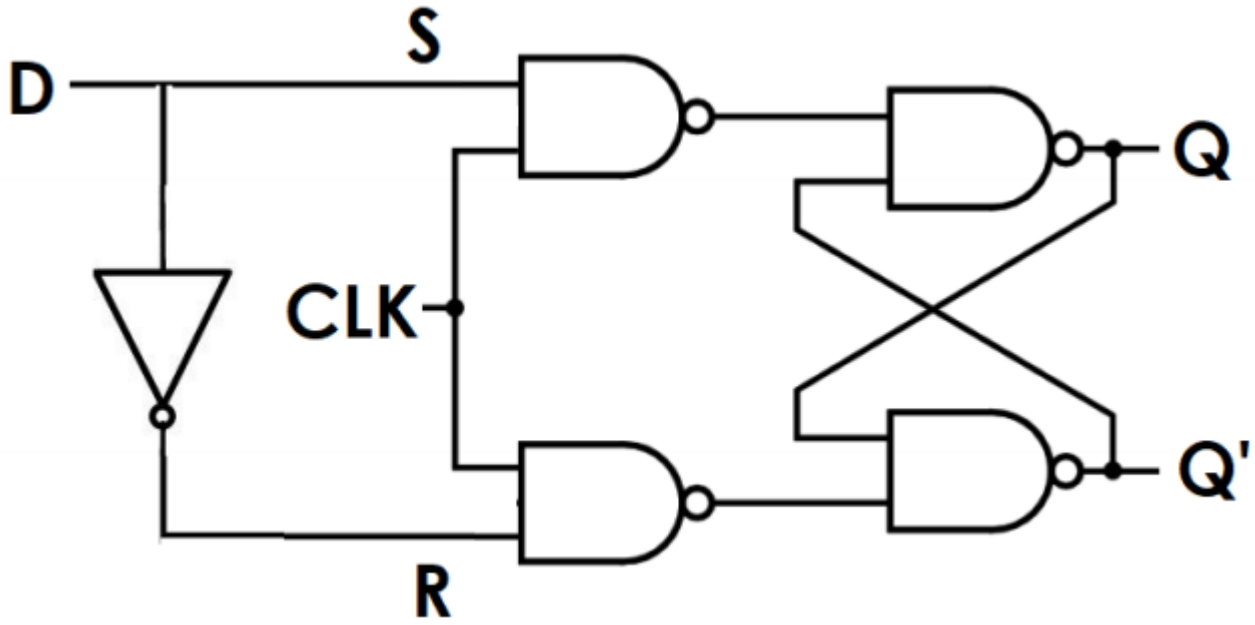
D FLIP-FLOP

BLOCK DIAGRAM



En	*D*	Next state of *Q*
0	X	No change
1	0	$Q = 0$; reset state
1	1	$Q = 1$; set state

LOGIC DIAGRAM



D FLIP-FLOP

VERILOG HDL CODE (Behavioural)

```
module D_FF(Q,QB,D,CLK);
input D,CLK;
output Q,QB;
reg Q,QB;
always @(posedge CLK)
begin
Q=D;
QB=~Q;
end
endmodule
```

D FLIP-FLOP

TEST BENCH

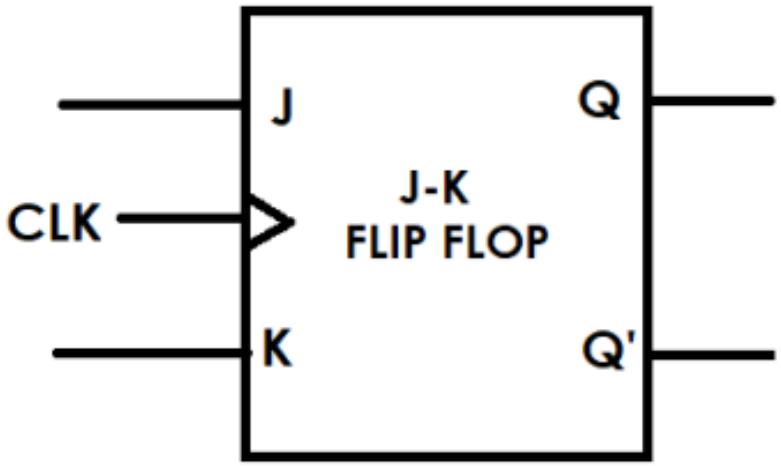
```
module D_FF_TB;
reg D;
reg CLK;
wire Q;
wire QB;
D_FF uut (.Q(Q), .QB(QB), .D(D), .CLK(CLK));
always #100 CLK=~CLK;
initial begin
CLK=1;
#200 D=1;
#200 D=0;
end
endmodule
```

JK FLIP-FLOP

- The JK(Jack-Kilby) Flip Flop is the most widely used flip flop. It is considered a **universal flip-flop circuit.**
- A JK Flip-Flop can be obtained from the clocked SR Flip-Flop by augmenting two AND gates.
- The sequential operation of the JK Flip Flop is same as SR flip-flop except JK Flip Flop does not invalid input states when S & R are 1.
- When both J and K are at logic “1”, the JK Flip Flop toggle.

JK FLIP-FLOP

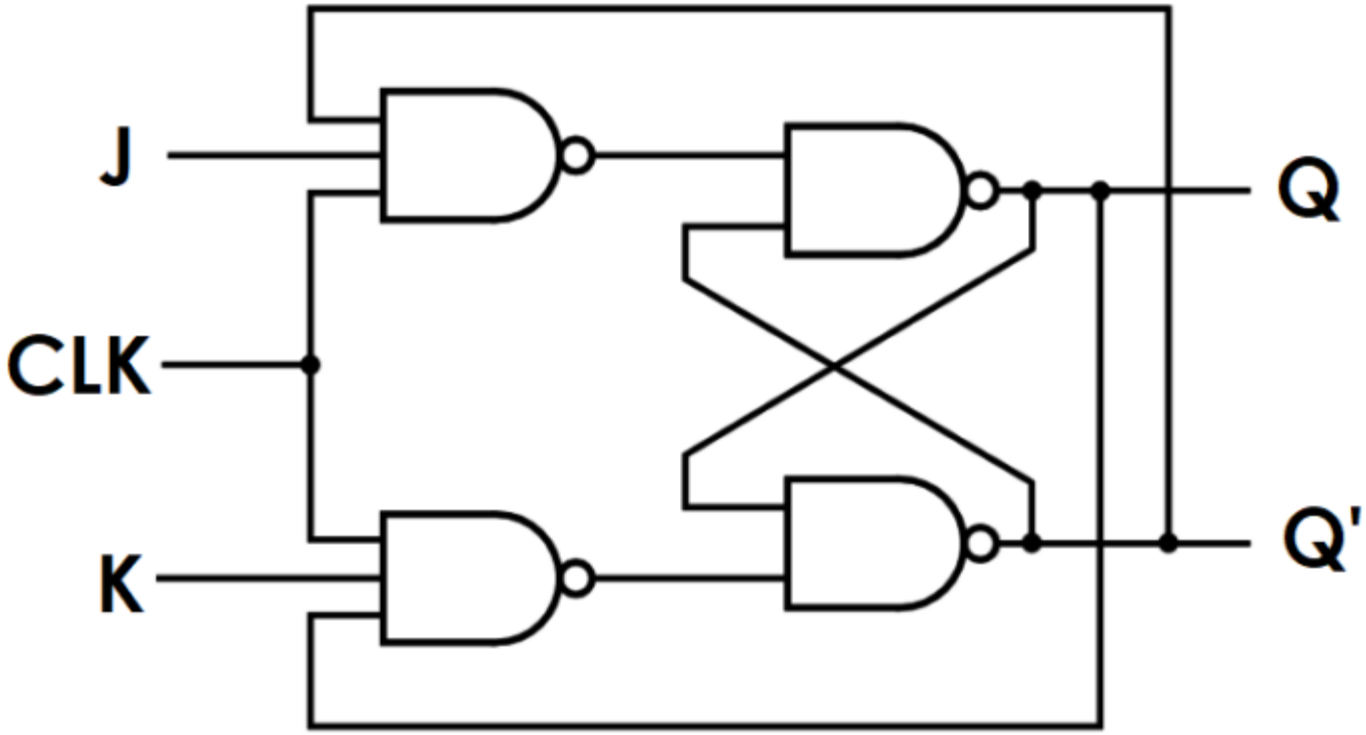
BLOCK DIAGRAM



JK Flip-Flop

J	K	Q(t + 1)	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

LOGIC DIAGRAM



JK FLIP-FLOP

VERILOG HDL CODE (Behavioural)

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

JK FLIP-FLOP

TEST BENCH

```
module JK_FF_TB;
reg J;
reg K;
reg CLK;
wire Q;
wire QB;
JK_FF uut (.Q(Q), .QB(QB), .J(J), .K(K), .CLK(CLK));
always #100 CLK=~CLK;
initial begin
        CLK=1;
        #200 J=1;K=0;
        #200 J=0; K=0;
        #200 J=0; K=1;
        #200 J=1; K=1;
end
endmodule
```

T FLIP-FLOP

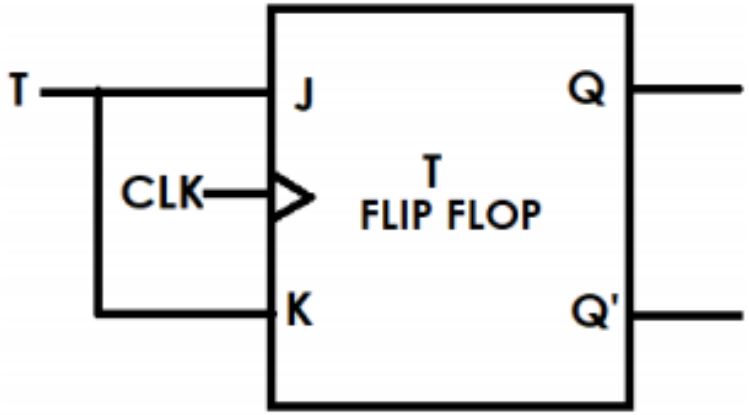
- T flip – flop is also known as “Toggle Flip – flop”. Toggling means ‘Changing the next state output to complement of the present state output’.

- we should provide only one input to the flip – flop called Trigger input or Toggle input (T). The T (Toggle) Flip-Flop is a modification of the JK Flip-Flop.

- It is obtained from JK Flip-Flop by connecting both inputs J and K together, i.e., single input. Regardless of the present state, the Flip-Flop complements its output when the clock pulse occurs while input T= 1.

T FLIP-FLOP

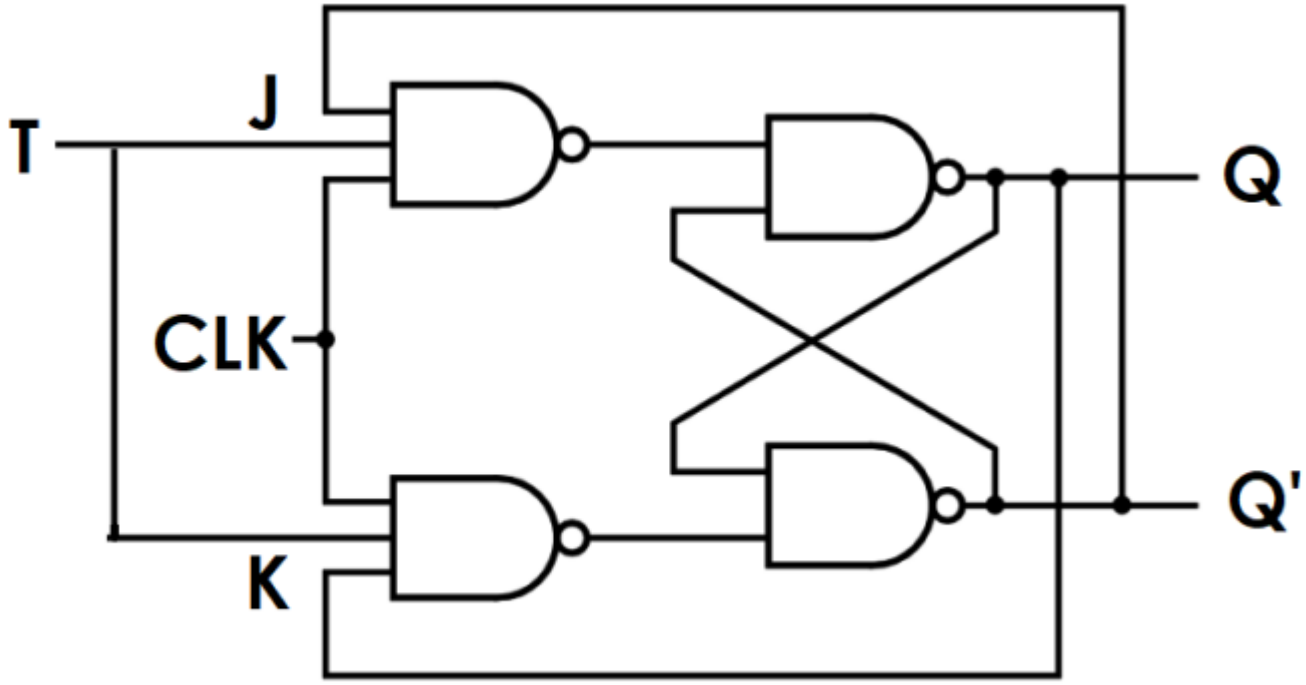
BLOCK DIAGRAM



T Flip-Flop

T	Q(t + 1)	
0	$Q(t)$	No change
1	$Q'(t)$	Complement

LOGIC DIAGRAM



T FLIP-FLOP

VERILOG HDL CODE (Behavioural)

```
module T_FF(Q,QB,T,CLK);
input T,CLK;
output Q,QB;
reg Q=0,QB;
always @(posedge CLK)
begin
case(T)
1'b0:Q=Q;
1'b1:Q=~Q;
endcase
QB=~Q;
end
endmodule
```

T FLIP-FLOP

TEST BENCH

```
module T_FF_TB;
reg T;
reg CLK;
wire Q;
wire QB;
T_FF uut (.Q(Q), .QB(QB), .T(T),.CLK(CLK));
always #100 CLK=~CLK;
initial begin
CLK=1;
#200 T=0;
#200 T=1;
#200 T=0;
#200 T=1;
end
endmodule
```