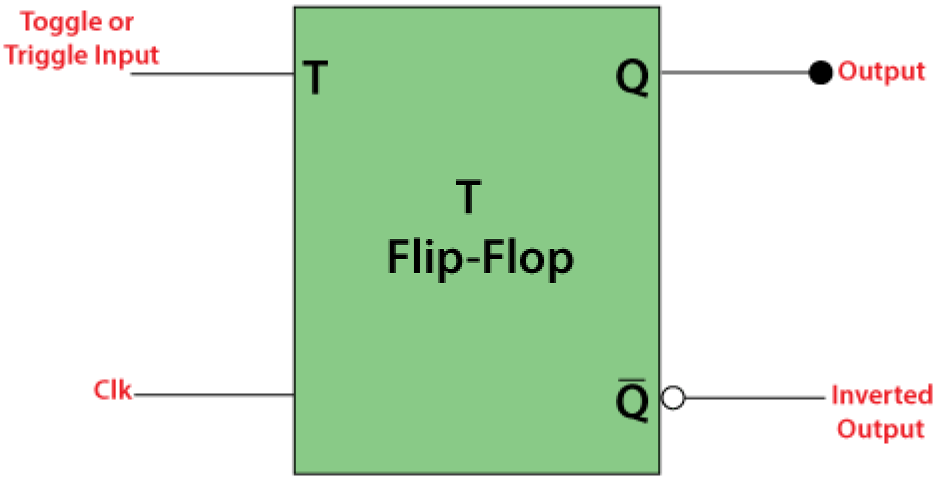
DAY 24:

T FlipFlop

In T flip flop, "T" defines the term "Toggle". In SR Flip Flop, we provide only a single input called "Toggle" or "Trigger" input to avoid an intermediate state occurrence. Now, this flip-flop work as a Toggle switch. The next output state is changed with the complement of the present state output. This process is known as "Toggling"'.

We can construct the "T Flip Flop" by making changes in the "JK Flip Flop". The "T Flip Flop" has only one input, which is constructed by connecting the input of JK flip flop. This single input is called T. In simple words, we can construct the "T Flip Flop" by converting a "JK Flip Flop". Sometimes the "T Flip Flop" is referred to as single input "JK Flip Flop".

Block diagram of the "T-Flip Flop" is given where T defines the "Toggle input", and CLK defines the clock signal input.

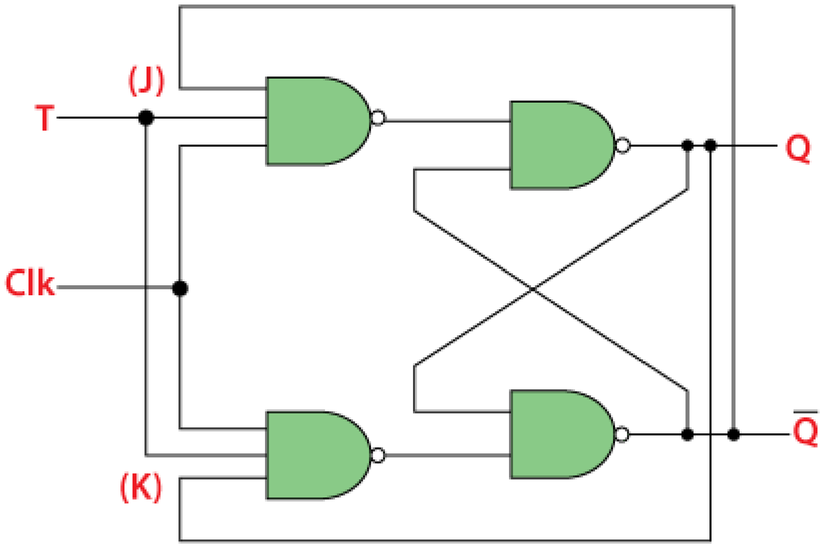


T Flip Flop Circuit

There are the following two methods which are used to form the "T Flip Flop":

- By connecting the output feedback to the input in "SR Flips Flop".
- We pass the output that we get after performing the XOR operation of T and Q_{PREV} output as the D input in D Flip Flop.

Construction



Truth Table of T Flip Flop

	Previous		Next	
T	Q	Q'	Q	Q'
0	0	1	0	1
0	1	0	1	0
1	0	1	1	0
1	1	0	0	1

The upper NAND gate is enabled, and the lower NAND gate is disabled when the output Q To is set to 0. make the flip flop in "set state(Q=1)", the trigger passes the S input in the flip flop.

The upper NAND gate is disabled, and the lower NAND gate is enabled when the output Q is set to 1. The trigger passes the R input in the flip flop to make the flip flop in the reset state(Q=0).

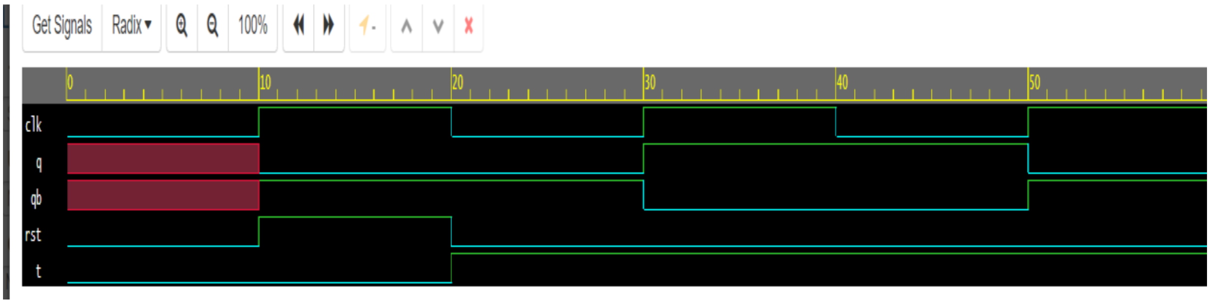
RTL CODE:

```
module T_flipflop(input T, clk, output reg q);
  always@(posedge clk)
    begin
      case({T})
```

```
        1'b0: q<= q;
        1'b1: q<= ~q;
      endcase
    end
endmodule
```

TEST BENCH:

```
module testbench;
  reg T,clk;
  wire q;
  T_flipflop a1 (T, clk, q);
  initial
    begin
      $dumpfile(".vcd");
      $dumpvars(1);
    end
  initial
    begin
      clk = 1;
      forever #15 clk= ~clk;
    end
  initial
    begin
      repeat(5)
    begin
     T=$random;
    end
  end
 initial
  begin
   #50 $finish();
 end
endmodule
```


Get Signals
Radix ▾
100%
0
10
20
30
40
50
clk
q
qb
rst
t