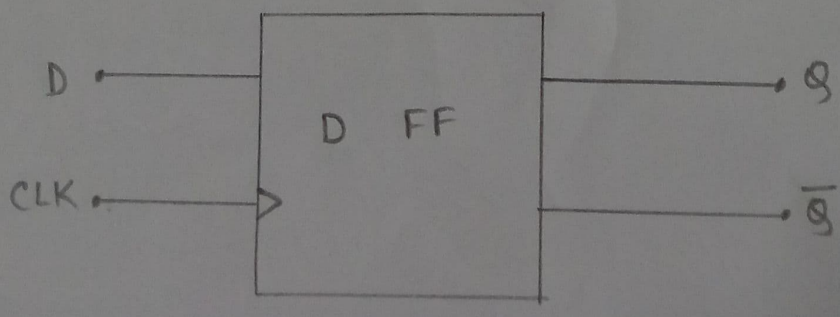
Name - Moitrish Maity Roll-48 CSE Sem-4

EXPERIMENT-9

- Title: Sequential logic
- Objective: Implementation of sequential logic
- Course Outcome: CO3
- Bloom's Level: Analysis
- Problem Statement: Write a verilog program to design, simulate and test circuits of following flipflops —
  - A) D Flipflop
  - B) S-R Flipflop
  - C) J-K Flipflop
  - D) T Flipflop.

[A] D Flipflop

- Theory: A D (or delay) flipflop is a sequential circuit used to delay the change of state of its output signal (Q) until the next rising or falling edge of a clock timing input signal occurs. Here, output remains constant unless changed by altering the state of D input followed by a clock signal.
- Logic Diagram:



Truth Table:

CLK	D	Q(t)	$\bar{Q}(t)$	REMARKS
L	X	Q(t-1)	$\bar{Q}(t-1)$	PREVIOUS STATE
H	L	L	H	RESET
H	H	H	L	SET

Design Code:

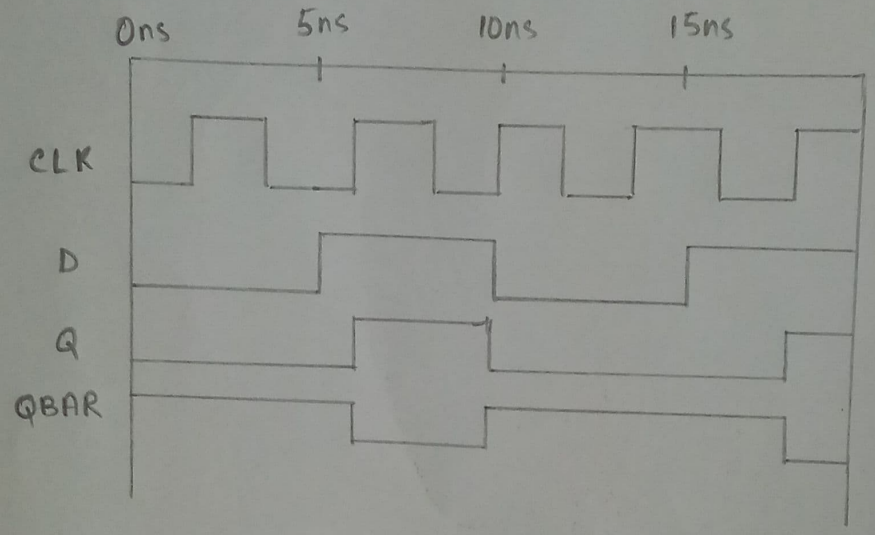
```
module d_ff(clk, d, q, qbar);
    input clk;
    input d;
    output reg q=0;   output qbar;
  assign qbar = ~q;
  always @(posedge clk)
  begin
        q <= d;
  end
endmodule
```

Testbench Code:

```
module d_ff_tb();
  reg CLK;
  reg D;
  wire Q, QBAR;
  d_ff uut(.clk(CLK), .d(D), .q(Q), .qbar(QBAR));
      initial begin
          $dumpfile("dump.vcd"); $dumpvars(1);
```

```
        CLK=0;
        D=0; #5;
        D=1; #5;
        D=0; #5;
        D=1; #5;
      $finish;
      end
        always #2 CLK=~CLK;
endmodule
```

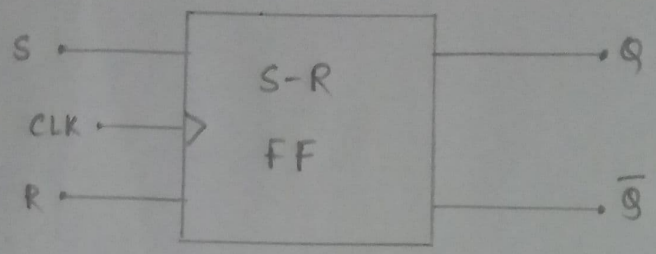
□ Timing Diagram:



[B] S-R Flip Flop

□ Theory: SR FF is a 1 bit memory bistable device having two inputs i.e. SET and RESET. The SET input 'S' set the device or produce the output '1' in 'Q' terminal whereas RESET input 'R' reset the device and produce output '0' in 'Q' terminal. If both 'S' and 'R' are active simultaneously then the S-R ff goes to a invalid state where $Q = \overline{Q}$ happens.

Logic Diagram:



Truth Table:

CLK	S	R	$Q(t)$	$\bar{Q}(t)$	REMARKS
L	X	X	$Q(t-1)$	$\bar{Q}(t-1)$	PREVIOUS STATE
H	L	L	$Q(t-1)$	$\bar{Q}(t-1)$	NO CHANGE
H	L	H	L	H	RESET
H	H	L	H	L	SET
H	H	H	-	-	INVALID

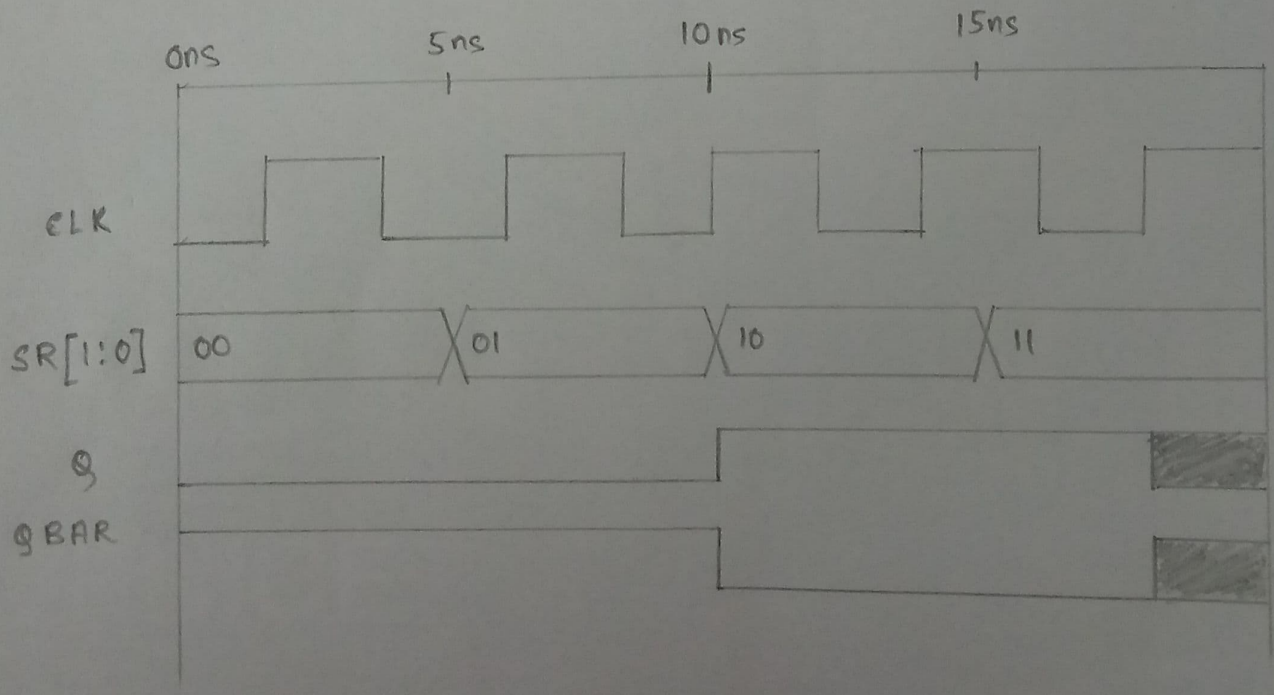
Design Code:

```
module sr_ff (sr, clk, q, qbar);
input [1:0] sr;
input clk;
output reg q=0; output qbar;
assign qbar = ~q;
   always @ (posedge clk)
   begin
        case (sr)
            2'b00 : begin q<=q ; end
            2'b01 : begin q<=0 ; end
            2'b10 : begin q<=1 ; end
            2'b11 : begin q<=1'bx ; end

        endcase
   end
endmodule
```

Testbench Code:

```
module sr_ff_tb();
    reg [1:0]SR ;
    reg CLK;
    wire Q, QBAR;
    sr_ff uut (.sr(SR),.clk(CLK),.q(Q),.qbar(QBAR));
    initial begin
        $dumpfile("dump.vcd"); $dumpvars(1);
        CLK=0;
        SR=2'b00 ; #5 ;
        SR=2'b01 ; #5 ;
        SR=2'b10 ; #5 ;
        SR=2'b11 ; #5 ;
      $ finish;
      end
       always  #2  CLK=~CLK;
endmodule
```

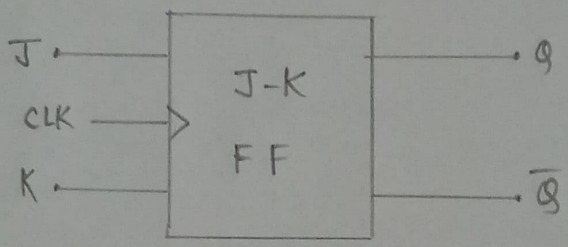
Timing Diagram:



C J-K Flipflop

Theory: JK flipflop is basically a gated SR flipflop with addition of clock circuit input. When both the inputs 'S' and 'R' are equal to logic '1', the invalid condition takes place. Thus to prevent this condition, a clock circuit is introduced. The J-K flipflop has four possible input combinations "logic 1", "logic 0", "No change" and "toggle". Here, cross-coupling of S-R flipflop is used to generate the 'Toggle' action.

Logic Diagram:



Truth Table:

CLK	J	K	$Q(t)$	$\bar{Q}(t)$	REMARKS
L	X	X	$Q(t-1)$	$\bar{Q}(t-1)$	PREVIOUS STATE
H	L	L	$Q(t-1)$	$\bar{Q}(t-1)$	NO CHANGE
H	L	H	L	H	RESET
H	H	L	H	L	SET
H	H	H	$\bar{Q}(t-1)$	$Q(t-1)$	TOGGLE

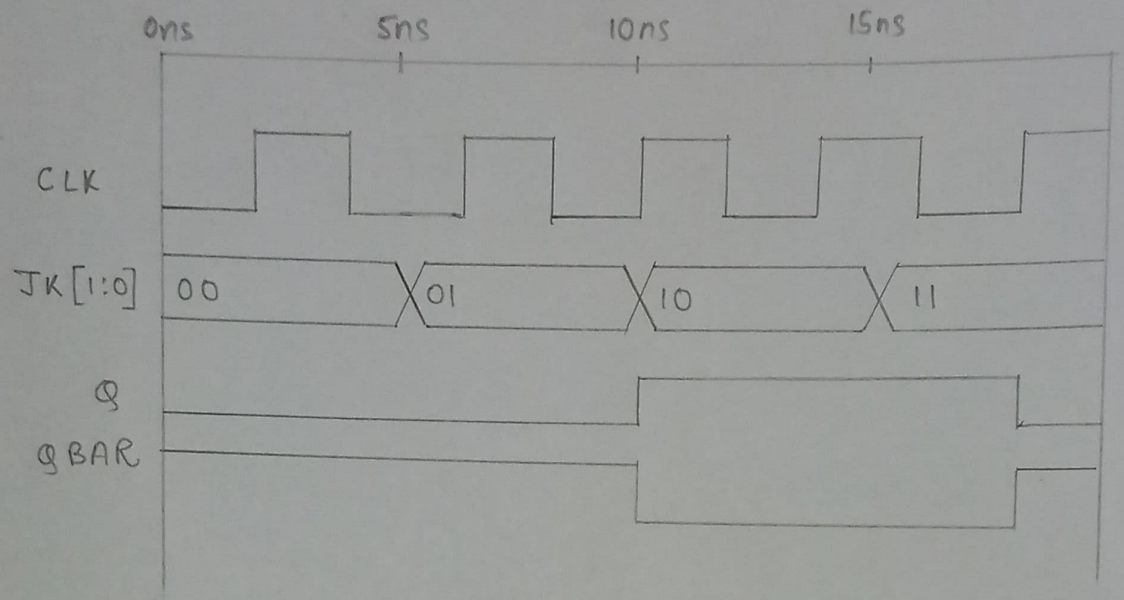
Design Code:

```
module jk-ff (jk, clk, q, qbar);
    input [1:0]jk;
    input clk;
    output reg q=0; output qbar;
    assign qbar = ~q;
    always @(posedge clk)
      begin
          case (jk)
            2'b00 : begin q<=q; end
            2'b01 : begin q<=0; end
            2'b10 : begin q<=1; end
            2'b11 : begin q<=~q; end
          endcase
    end
endmodule
```

Testbench Code:

```
module jk-ff-tb();
    reg [1:0] JK; reg CLK;
    wire Q, QBAR;
    jk-ff uut(.jk(JK),.clk(CLK),.q(Q),.qbar(QBAR));
    initial begin
        $dumpfile("dump.vcd"); $dumpvars(1);
        CLK=0;
        JK = 2'b00; #5;
        JK = 2'b01; #5;
        JK=2'b10; #5;
        JK=2'b11; #5;
        $finish;
    end
endmodule
```

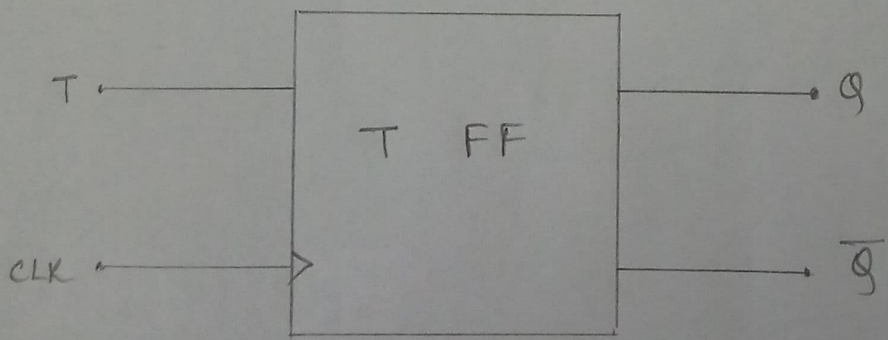
□ Timing Diagram:



D T Flipflop

□ Theory: T flipflop is also known as 'Toggle' Flipflop. Toggling means changing the next state output to compliment of present state output. We can design T ff by making simple modifications to J-K flipflop by connecting J and K inputs together and giving them with single input called 'T'.

□ Logic Diagram:



Truth Table

CLK	T	$q(t)$	$\overline{q}(t)$	REMARKS
L	X	$q(t-1)$	$\overline{q}(t-1)$	NO CHANGE
H	L	$q(t-1)$	$\overline{q}(t-1)$	NO CHANGE
H	H	$\overline{q}(t-1)$	$q(t-1)$	TOGGLE

Design Code:

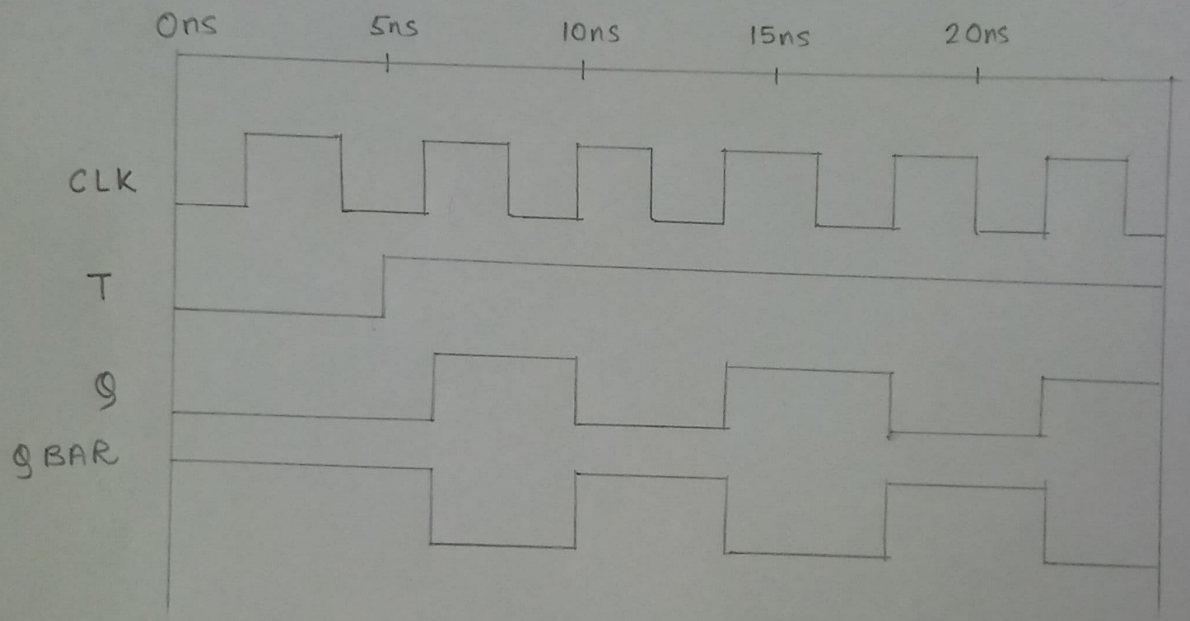
```
module t_ff (clk, t, q, qbar);
  input clk, t;
  output reg q=0;
  output qbar;
  assign qbar = ~q;
  always @ (posedge clk)
    begin
        if (t==1) begin
            q <= ~q; end

    end
endmodule
```

Testbench Code:

```
module t_ff_tb();
    reg CLK;
    reg T;
    wire Q, QBAR;
    t_ff uut (.clk(CLK), .t(T), .q(Q), .qbar(QBAR));
        initial begin
            $ dumpfile ("dump.vcd");
            $ dumpvars (1);
```

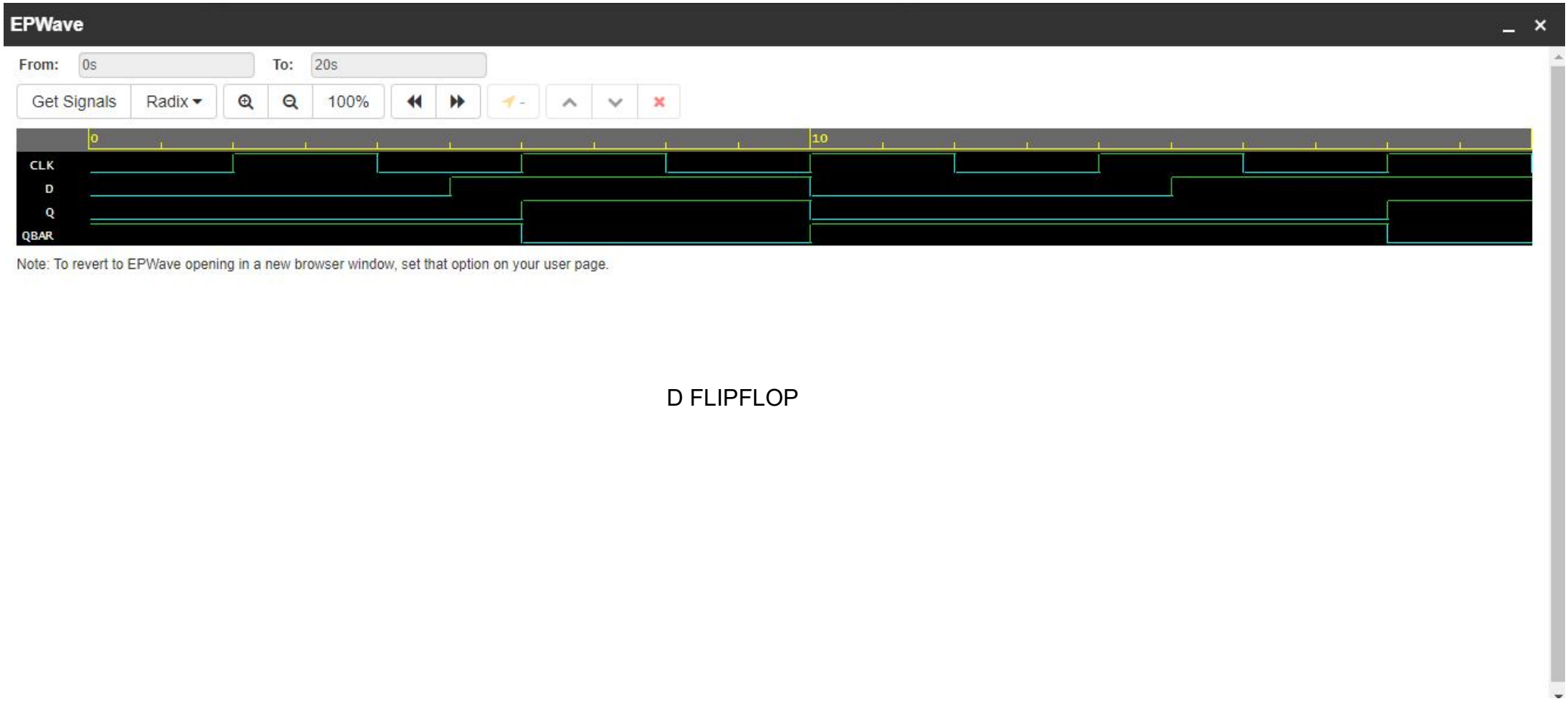
```
        CLK=0 ;
        T=0 ; #5 ;
        T=1 ; #5 ;
        T=1 ; #5 ;
        T=1 ; #5 ;
        T=1 ; #5 ;
    end
endmodule
```

- Timing Diagram:

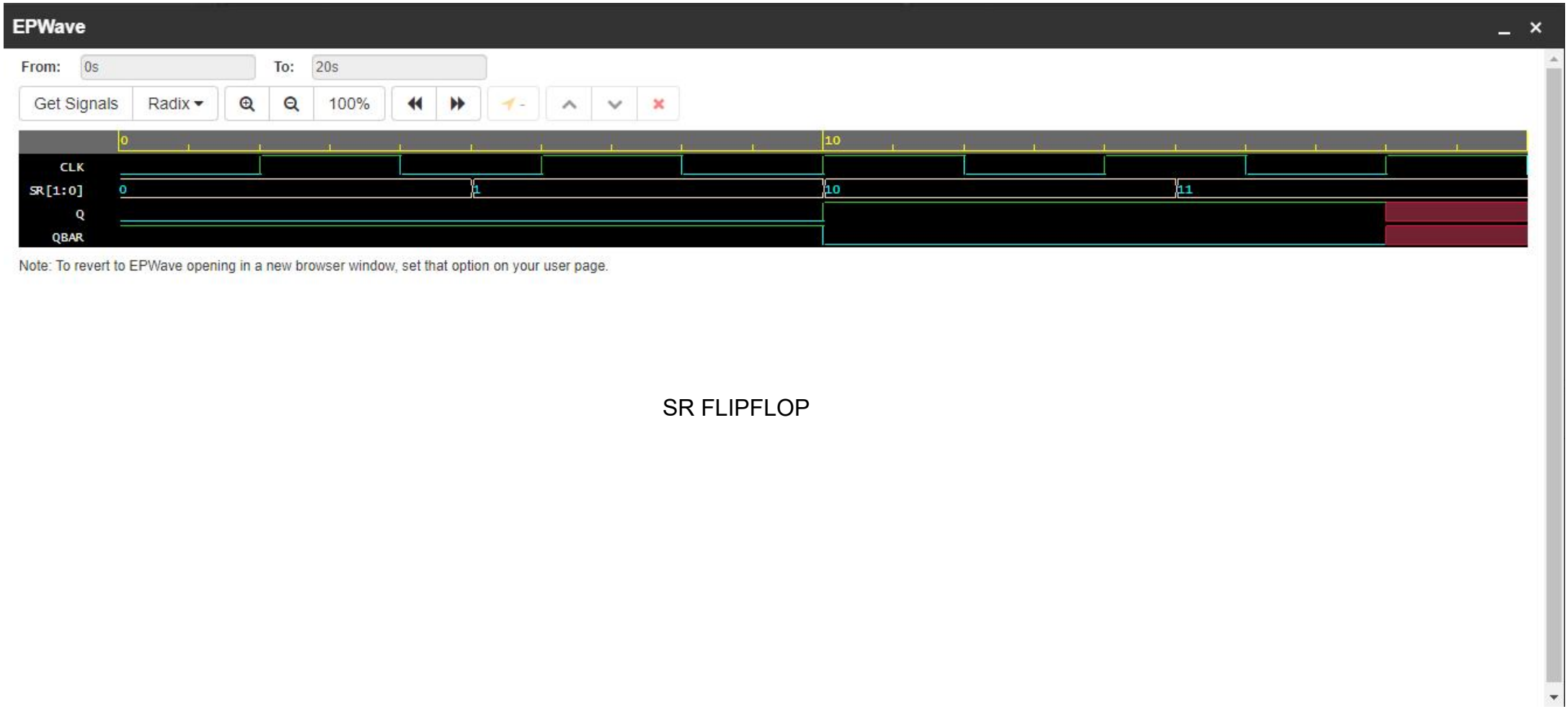


- Discussion: In this experiment, we have implemented various sequential circuits like flipflops. We have learned about 'always @' keyword and many other keywords in verilog.

- Justification of CO: In this experiment, we have implemented various sequential circuits which fulfils the conditions of CO3. Hence, CO3 is justified.


EPWave
From:
0s
To:
20s
Get Signals
Radix
100%
0
10
CLK
D
Q
QBAR
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

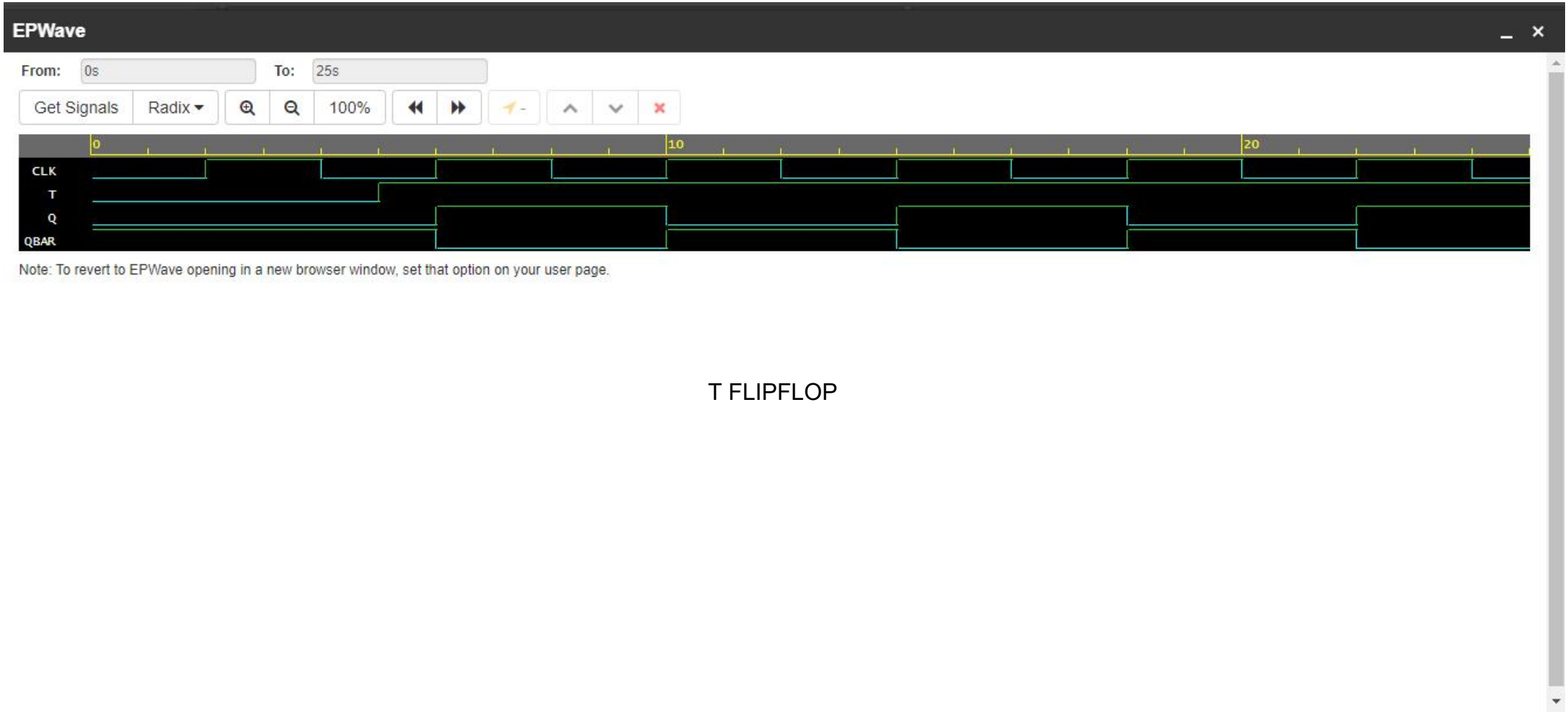
D FLIPFLOP


EPWave
From:
0s
To:
20s
Get Signals
Radix
100%
0
10
CLK
SR[1:0]
0
1
10
11
Q
QBAR
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

SR FLIPFLOP


EPWave
From:
0s
To:
20s
Get Signals
Radix
100%
0
10
CLK
JK[1:0]
0
1
10
11
Q
QBAR
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

JK FLIPFLOP


EPWave
From:
0s
To:
25s
Get Signals
Radix
100%
0
10
20
CLK
T
Q
QBAR
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

T FLIPFLOP