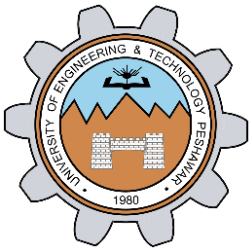
FLIP FLOP AND LATCHES IN VERILOG

LAB # 09



Fall 2023

CSE-304L Computer Organization and Architecture Lab

Submitted by: **Ali Asghar**

Registration No.: **21PWCSE2059**

Class Section: **C**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Submitted to:

Dr. Bilal Habib

Date:

29th December 2023

Department of Computer Systems Engineering

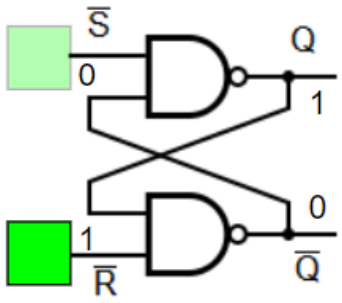
University of Engineering and Technology, Peshawar

ASSESSMENT RUBRICS COA LABS

LAB REPORT ASSESSMENT				
Criteria	**Excellent**	**Average**	**Nill**	**Marks Obtained**
1. Objectives of Lab	All objectives of lab are properly covered [Marks 10]	Objectives of lab are partially covered [Marks 5]	Objectives of lab are not shown [Marks 0]	
2. MIPS instructions with Comments and proper indentations.	All the instructions are well written with comments explaining the code and properly indented [Marks 20]	Some instructions are missing are poorly commented code [Marks 10]	The instructions are not properly written [Marks 0]	
3. Simulation run without error and warnings	The code is running in the simulator without any error and warnings [Marks 10]	The code is running but with some warnings or errors. [Marks 5]	The code is written but not running due to errors [Marks 0]	
4. Procedure	All the instructions are written with proper procedure [Marks 20]	Some steps are missing [Marks 10]	steps are totally missing [Marks 0]	
5. OUTPUT	Proper output of the code written in assembly [Marks 20]	Some of the outputs are missing [Marks 10]	No or wrong output [Marks 0]	
6. Conclusion	Conclusion about the lab is shown and written [Marks 20]	Conclusion about the lab is partially shown [Marks 10]	Conclusion about the lab is not shown[Marks0]	
7. Cheating			Any kind of cheating will lead to 0 Marks	
Total Marks Obtained:_____ Instructor Signature: _____				

Task 1:

Write a Verilog code to implement S R latch.



TRUTH TABLE

INPUTS		OUTPUTS	
$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$
0	0	X	X
0	1	1	0
1	0	0	1
1	1	Q_0	$\overline{Q}_0$

Code:

DUT Code:

SR_Latch.v | tb_SR_Latch.v | SR_FlipFlop.v | tb_SR_FlipFlop.v | JK_FlipFlop.v | JK_FF.v

```
1   module SR_Latch(S, R, Q, QBar);
2
3       input S;
4       input R;
5
6       output Q;
7       output QBar;
8
9       //reg Q,QBar;
10
11      assign Q = ~(S & QBar);
12      assign QBar = ~(R & Q);
13
14
15  endmodule
16
```

Test bench Code:

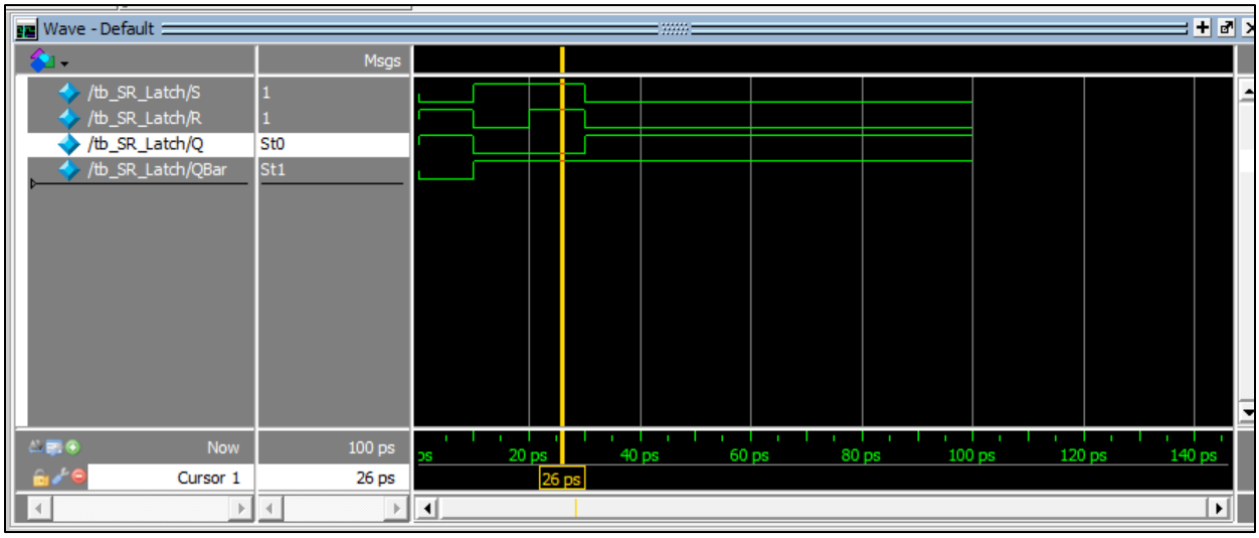
SR_Latch.v | tb_SR_Latch.v | SR_FlipFlop.v | tb_SR_FlipFlop.v | JK_FlipFlop.v | JK_FF.v | tb_JK_FlipFlop.v | D_

```
1   module tb_SR_Latch();
2
3       reg S;
4       reg R;
5       wire Q;
6       wire QBar;
7
8       SR_Latch my_SR_Latch(S, R, Q, QBar);
9
10      initial begin
11      $display("S R Q QBar");
12
13      S = 0;
14      R = 1;
15      #10
16      $display("%b %b %b %b", S , R, Q,QBar);
17
18
19      S = 1;
20      R = 0;
21      #10
22      $display("%b %b %b %b", S , R, Q,QBar);
```

SR_Latch.v | tb_SR_Latch.v | SR_FlipFlop.v | tb_SR_FlipFlop.v | JK_FlipFlop.v | JK_FF.v | tb_JK_Flip

```
16      $display("%b %b %b %b", S , R, Q,QBar);
17
18
19      S = 1;
20      R = 0;
21      #10
22      $display("%b %b %b %b", S , R, Q,QBar);
23
24      S = 1;
25      R = 1;
26      #10
27      $display("%b %b %b %b", S , R, Q,QBar);
28
29
30      S = 0;
31      R = 0;
32      #10
33      $display("%b %b %b %b", S , R, Q,QBar);
34
35      end
36
37  endmodule
```

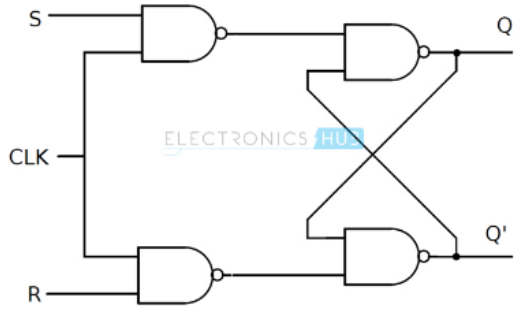
Output:



```
# S R Q QBar
# 0 1 1 0
# 1 0 0 1
# 1 1 0 1
# 0 0 1 1
```

Task 2:

Implement SR flip flop in Verilog.



Code:

DUT Code:

SR_Latch.v | tb_SR_Latch.v | SR_FlipFlop.v | tb_SR_FlipFlop.v | JK_FlipFlop.v | JK_FF.v | tb_JK_FlipFlop.v

```
1  module SR_FlipFlop(S, R,Clk, Q, QBar);
2
3      input S;
4      input R;
5      input Clk;
6
7      output Q;
8      output QBar;
9
10     wire nA;
11     wire nB;
12
13     nand n1(nA, S, Clk);
14     nand n2(nB, R, Clk);
15
16     nand n3(Q, nA, QBar);
17     nand n4(QBar, nB, Q);
18
19
20
21 endmodule
```

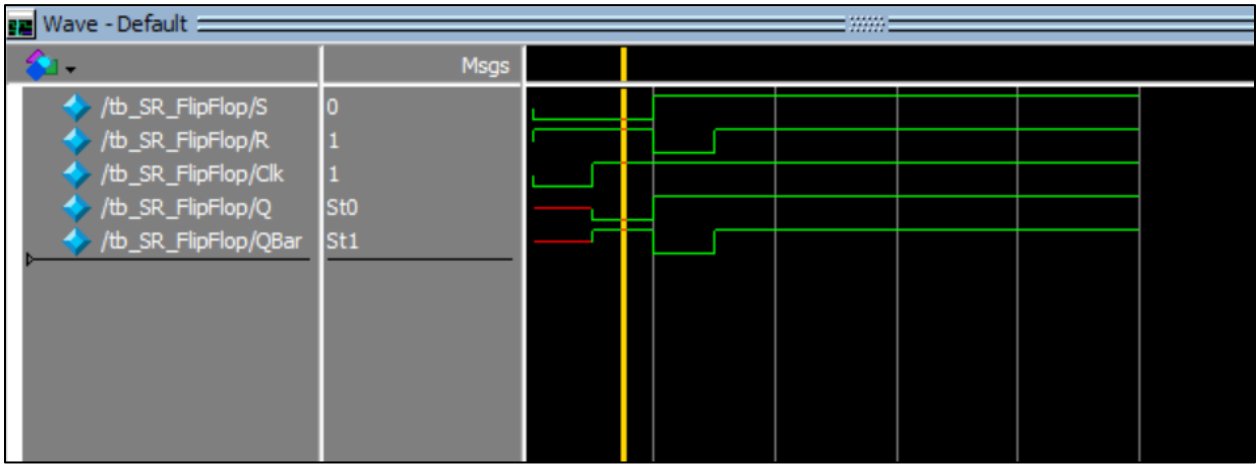
Test bench Code:

SR_Latch.v | tb_SR_Latch.v | SR_FlipFlop.v | tb_SR_FlipFlop.v | JK_FlipFlop.v | JK_FF.v | tb_JK_FlipFlop.v | D_FlipFlop.v

```
1  module tb_SR_FlipFlop();
2
3      reg S;
4      reg R;
5      reg Clk;
6
7      wire Q;
8      wire QBar;
9
10     SR_FlipFlop my_SR_FlipFlop(S, R, Clk, Q, QBar);
11
12     initial begin
13
14     $display("S R C Q Q'");
15
16     S = 0;
17     R = 1;
18     Clk = 0;
19     #10
20     $display("%b %b %b %b %b", S , R,Clk, Q,QBar);
21
22
```

```
23      S = 0;
24      R = 1;
25      Clk = 1;
26      #10
27      $display("%b %b %b %b %b", S , R,Clk, Q,QBar);
28
29      S = 1;
30      R = 0;
31      Clk = 1;
32      #10
33      $display("%b %b %b %b %b", S , R,Clk, Q,QBar);
34
35
36      S = 1;
37      R = 1;
38      Clk = 1;
39      #10
40      $display("%b %b %b %b %b", S , R,Clk, Q,QBar);
41
42      end
43  endmodule
```

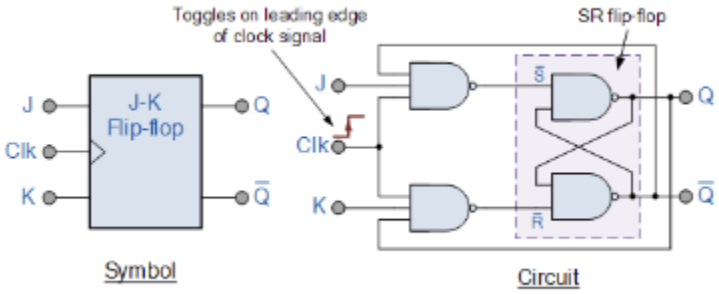
Output:




```
# S R C Q Q'
# 0 1 0 x x
# 0 1 1 0 1
# 1 0 1 1 0
# 1 1 1 1 1
```

Task 3:

Implement JK Flip Flop in Verilog



Code:

DUT Code:

```
1   module jkff_behavior(J, K, rst, Clk, Q, QBar);
2
3       input J;
4       input K;
5       input Clk;
6       input rst;
7
8       output reg Q, QBar;
9
10      always @(posedge Clk or posedge rst)
11      begin
12          if (rst)
13          begin
14              Q <= 1'b0;
15          end
16
17          else
18          begin
19              case ({J, K})
20                  2'b00: Q <= Q;
21                  2'b01: Q <= 1'b0;
22                  2'b10: Q <= 1'b1;
```

```
        if (rst)
        begin
            Q <= 1'b0;
        end

        else
        begin
            case ({J, K})
                2'b00: Q <= Q;
                2'b01: Q <= 1'b0;
                2'b10: Q <= 1'b1;
                2'b11: Q <= ~Q;
            endcase
        end
    end

    always @(Q)
    begin
        QBar <= ~Q;
    end

endmodule
```

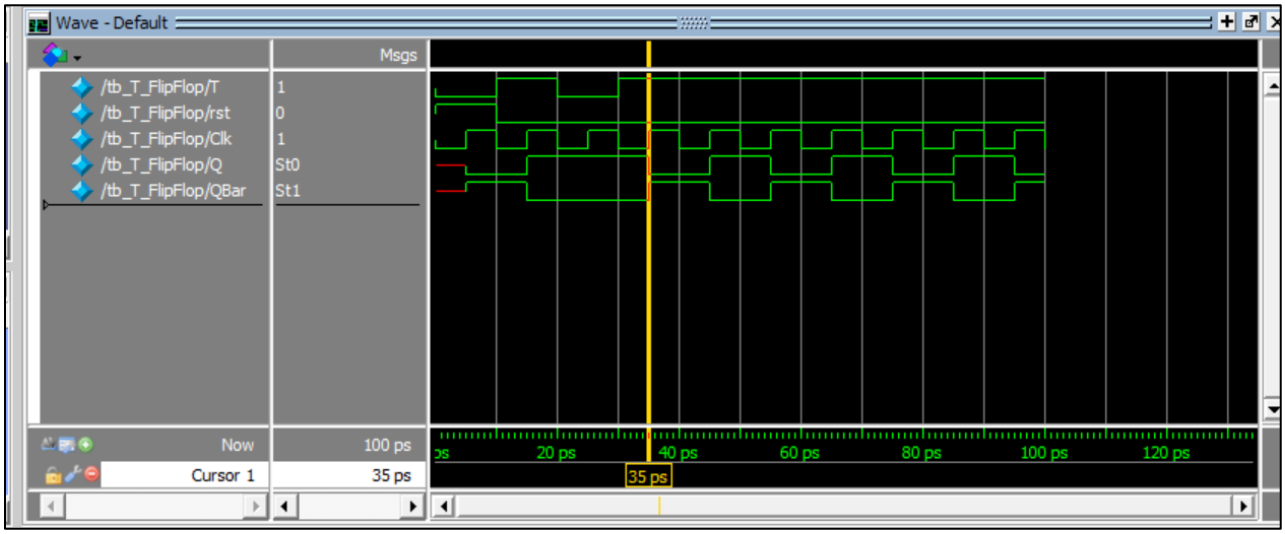
Test bench Code:

```
module tb_jkff_behavior();
    reg J, K;
    reg rst;
    reg Clk;

    wire Q, QBar;

    jkff_behavior my_jkff_behavior(.J(J), .K(K), .rst(rst),
    .Clk(Clk), .Q(Q), .QBar(QBar));

    // Clock generation
    always #5 Clk = ~Clk;

    initial begin

    $display("J K Q Q'");
    rst = 1;
    Clk = 1;
    #10
    $display("%b %b %b %b", J , K, Q,QBar);

    J = 0;
```

```
22      J = 0;
23      K = 0;
24      rst = 0;
25      #10
26      $display("%b %b %b %b", J , K, Q,QBar);
27
28      J = 1;
29      K = 0;
30      #10
31      $display("%b %b %b %b", J , K, Q,QBar);
32
33      J = 0;
34      K = 1;
35      #10
36      $display("%b %b %b %b", J , K, Q,QBar);
37
38      J = 1;
39      K = 1;
40      #10
41      $display("%b %b %b %b", J , K, Q,QBar);
42
```

```
42
43      J = 0;
44      K = 0;
45      #10
46      $display("%b %b %b %b", J , K, Q,QBar);
47      end
48
49  endmodule
```

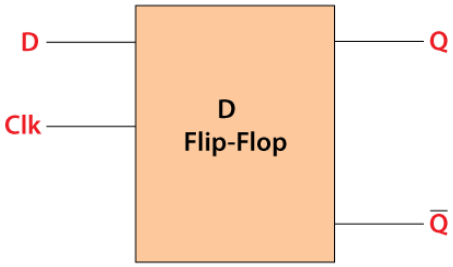
Output:



```
# J K Q Q'
# x x 0 1
# 0 0 0 1
# 1 0 1 0
# 0 1 0 1
# 1 1 1 0
# 0 0 1 0

VSIM 21>
```

Task 4:

Design D Flip Flop in Verilog.



Code:

DUT Code:

D_FlipFlop.v | tb_D_FlipFlop.v | tb_JK_FF.v | T_FlipFlop.v | tb_T_FlipFlop.v | four_bit_counter_behaviour.v | tb_four

```
1  module D_FlipFlop(D,rst, Clk, Q, QBar);
2      input D, Clk, rst;
3
4      output Q, QBar;
5
6      reg Q,QBar;
7
8      always@(posedge Clk)
9
10         if (rst)
11             begin
12                 Q <= 1'b0;
13                 QBar <= 1'b1;
14             end
15
16         else
17             begin
18                 Q <= D;
19                 QBar <= ~D;
20             end
21 endmodule
```

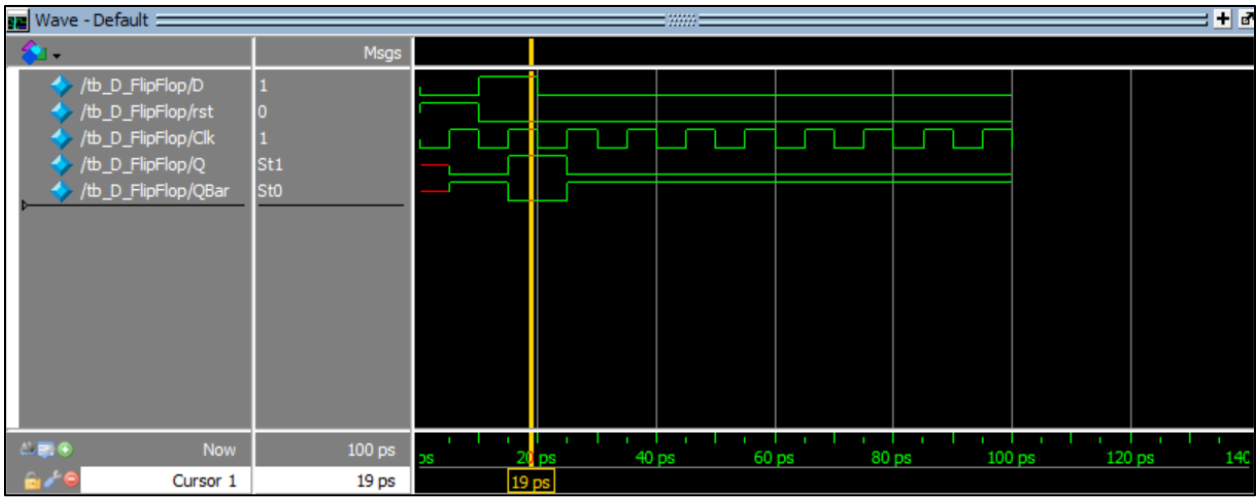
Test bench Code:

D_FlipFlop.v | tb_D_FlipFlop.v | tb_JK_FF.v | T_FlipFlop.v | tb_T_FlipFlop.v | four_bit_counter_behaviour.v | tb_four_bit_counter

```
module tb_D_FlipFlop();

    reg D, rst, Clk;

    wire Q;
    wire QBar;

    D_FlipFlop my_D_FlipFlop(D, rst, Clk, Q, QBar);

    initial begin
    Clk = 0;

    $display("D R C Q Q'");

    D = 0;
    rst = 1;
    #10
    $display("%b %b %b %b %b", D , rst, Clk, Q,QBar);

    D = 1;
    rst = 0;
    #10
```

```
    D = 1;
    rst = 0;
    #10
    $display("%b %b %b %b %b", D , rst, Clk, Q,QBar);

    D = 0;
    rst = 0;
    #10
    $display("%b %b %b %b %b", D , rst, Clk, Q,QBar);

    D = 0;
    rst = 0;
    #10
    $display("%b %b %b %b %b", D , rst, Clk, Q,QBar);

    end

    // Clock generation
    always #5 Clk = ~Clk;

endmodule
```

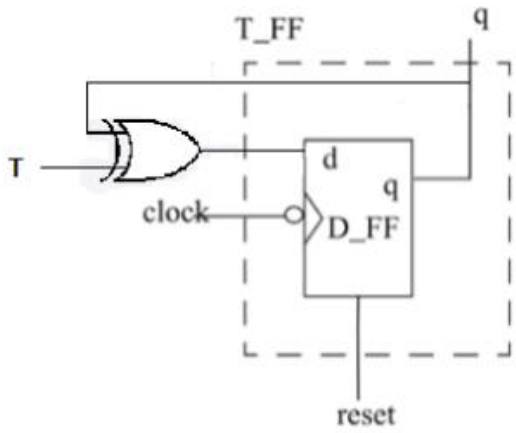
Output:




```
# D R C Q Q'
# 0 1 1 0 1
# 1 0 1 1 0
# 0 0 1 0 1
# 0 0 1 0 1
```

Task 5:

Design T Flip Flop using D Flip Flop in Verilog.



Code:

DUT Code:

```
module T_FlipFlop(T, rst, Clk,  Q, Qbar);

    input T;
    input rst;
    input Clk;
    output Q;
    output Qbar;

    wire O;

    xor(O, T, Q);
    D_FlipFlop my_D_FlipFlop(O, rst, Clk, Q, Qbar);


endmodule
```

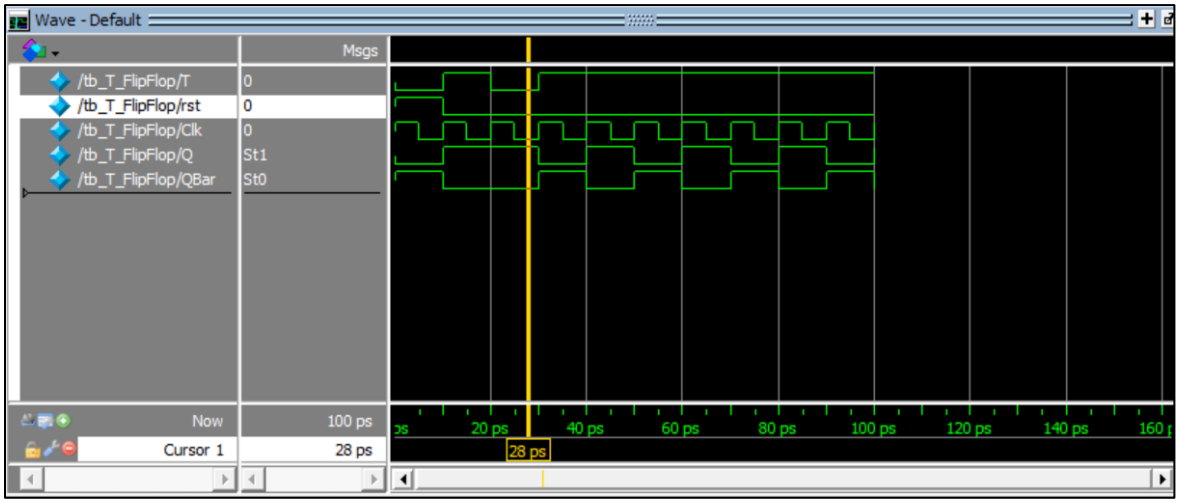
Test bench Code:

D_FlipFlop.v | tb_D_FlipFlop.v | tb_JK_FF.v | T_FlipFlop.v | tb_T_FlipFlop.v | four_bit_counter_behaviour.v | tb_four_b

```
module tb_T_FlipFlop();

    reg T, rst, Clk;
    wire Q, QBar;

    T_FlipFlop my_T_FlipFlop(T, rst, Clk, Q, QBar);

    initial begin
    Clk = 1;

    $display("T R Q Q'");

    T = 0;
    rst = 1;
    #10
    $display("%b %b %b %b ", T , rst, Q,QBar);

    T = 1;
    rst = 0;
    #10
    $display("%b %b %b %b ", T , rst, Q,QBar);

```

```
22
23      T = 0;
24      rst = 0;
25      #10
26      $display("%b %b %b %b", T , rst, Q,QBar);
27
28      T = 1;
29      rst = 0;
30      #10
31      $display("%b %b %b %b", T , rst, Q,QBar);
32      end
33
34      // Clock generation
35      always #5 Clk = ~Clk;
36
37  endmodule
```

Output:




```
# T R Q Q'
# 0 1 0 1
# 1 0 1 0
# 0 0 1 0
# 1 0 0 1
```

Conclusion:

In this lab, I learned how to implement Latches and Flip Flops in Verilog.