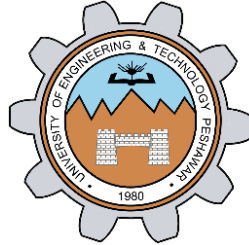


**FLIP FLOPS AND
LATCHES IN
VERILOG
LAB # 09**



Fall 2023

CSE-304L

Computer Organization & Architecture Lab

Submitted by: **AIMAL KHAN**

Registration No.: **21PWCSE1996**

Class Section: **A**

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

Student Signature: _____

A handwritten signature in black ink, appearing to be "Aimal Khan", written over a horizontal line.

Submitted to:

Dr. Bilal Habib

Friday, December 29, 2023

Department of Computer Systems Engineering
University of Engineering and Technology, Peshawar

ASSESSMENT RUBRICS COA LABS

LAB REPORT ASSESSMENT				
Criteria	Excellent	Average	Nil	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] [Marks 0]	
7. Cheating			Any kind of cheating will lead to 0 Marks	
Total Marks Obtained: _____ Instructor Signature: _____				

Flip Flop and Latches in Verilog

Objectives:

- Write and implement latches in Verilog
- Write and implement flipflops in Verilog
- Understand working and design of sequential circuits

Tasks:

Task 1: Write a Verilog code to implement S R latch.

DUT Code:

```
module SR_latch(S, R, Q, Qbar);
    input S;
    input R;
    output Q;
    output Qbar;

    assign Q = ~(S & Qbar);
    assign Qbar = ~(R & Q);
endmodule
```

Test Code:

```
module SR_latch_tb();
    reg S;
    reg R;
    wire Q;
    wire Qbar;

    SR_latch latch(S, R, Q, Qbar);

    initial begin
        S = 0;
        R = 1;
        #10
        $display("%b %b %b %b", S, R, Q, Qbar);

        S = 1;
        R = 1;
        #10
        $display("%b %b %b %b", S, R, Q, Qbar);

        S = 1;
        R = 0;
        #10
        $display("%b %b %b %b", S, R, Q, Qbar);

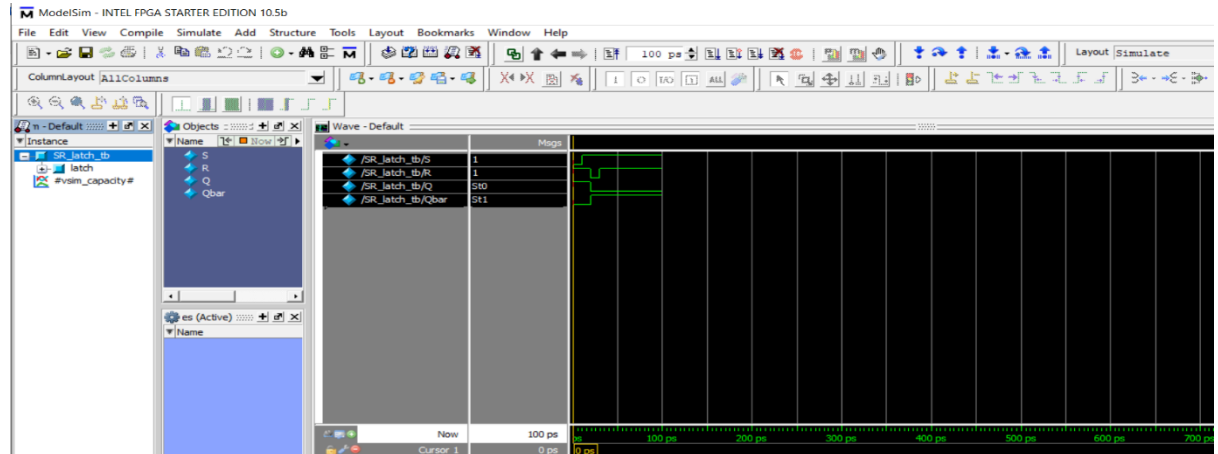
        S = 1;
```

```

        R = 1;
        #10
        $display("%b %b %b %b", S, R, Q, Qbar);
    end
endmodule;

```

Output:



Task 2: Implement SR flip flop in Verilog

DUT Code:

```

module SR_flipflop(S, R, clk, Q, Qbar);
    input S;
    input R;
    input clk;
    output Q;
    output Qbar;

    wire nan1;
    wire nan2;

    assign nan1 = ~(S & clk);
    assign nan2 = ~(R & clk);

    assign Q = ~(nan1 & Qbar);

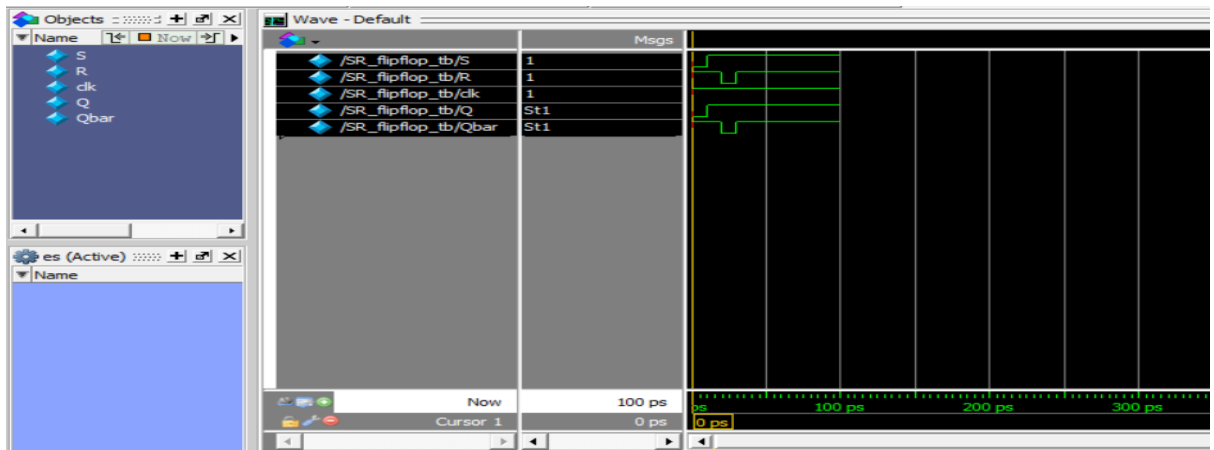
```

```
    assign Qbar = ~(nan2 & Q);  
endmodule;
```

Test Code:

```
module SR_flipflop_tb();  
    reg S;  
    reg R;  
    reg clk;  
    wire Q;  
    wire Qbar;  
  
    SR_flipflop flipflop(S, R, clk, Q, Qbar);  
  
    initial begin  
        clk =1;  
        S = 0;  
        R = 1;  
        #10  
        $display("%b %b %b %b %b", S, R, clk, Q, Qbar);  
  
        clk =1;  
        S = 1;  
        R = 1;  
        #10  
        $display("%b %b %b %b %b", S, R, clk, Q, Qbar);  
  
        clk =1;  
        S = 1;  
        R = 0;  
        #10  
        $display("%b %b %b %b %b", S, R, clk, Q, Qbar);  
  
        clk =1;  
        S = 1;  
        R = 1;  
        #10  
        $display("%b %b %b %b %b", S, R, clk, Q, Qbar);  
    end  
endmodule
```

Output:



```
Transcript
* Start time: 21:39:19 On Dec 20, 2023
# Loading work.SR_flipflop_tb
# Loading work.SR_flipflop
add wave -position insertpoint sim:/SR_flipflop_tb/*
VSIM 7> run
# 0 1 1 0 1
# 1 1 1 1 1
# 1 0 1 1 0
# 1 1 1 1 1
VSIM 8>
```

Task 3: Implement JK Flip Flop in verilog

DUT Code:

```
module JK_flipflop(J, K, clk, Q, Qbar);
    input J;
    input K;
    input clk;
    output Q;
    output Qbar;

    wire nan1;
    wire nan2;

    assign nan1 = ~(J & clk & Qbar);
    assign nan2 = ~(K & clk & Q);

    assign Q = ~(nan1 & Qbar);
    assign Qbar = ~(nan2 & Q);
endmodule;
```

Test Code:

```
module JK_flipflop_tb();
    reg J;
    reg K;
    reg clk;
```

```

wire Q;
wire Qbar;

JK_flipflop flipflop(J, K, clk, Q, Qbar);

initial begin
    clk =1;
    J = 0;
    K = 1;
    #10
    $display("%b %b %b %b %b", J, K, clk, Q, Qbar);

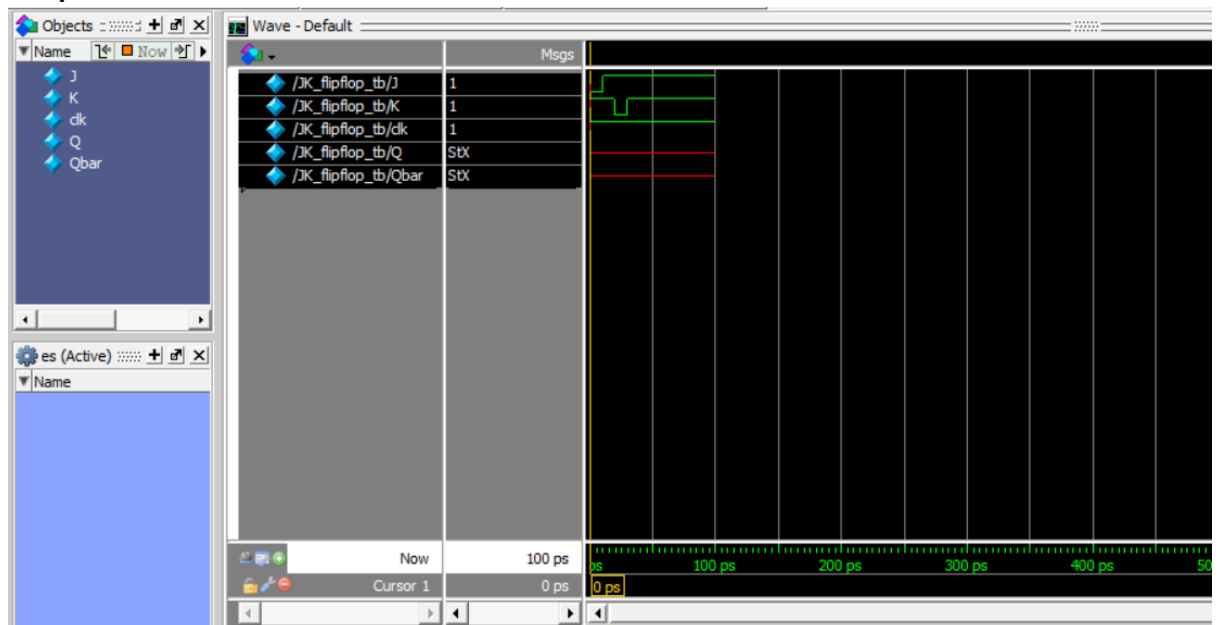
    clk =1;
    J = 1;
    K = 1;
    #10
    $display("%b %b %b %b %b", J, K, clk, Q, Qbar);

    clk =1;
    J = 1;
    K = 0;
    #10
    $display("%b %b %b %b %b", J, K, clk, Q, Qbar);

    clk =1;
    J = 1;
    K = 1;
    #10
    $display("%b %b %b %b %b", J, K, clk, Q, Qbar);
end
endmodule;

```

Output:



```
Transcript
# Start time: 22.01.42 On Dec 20, 2023
# Loading work.JK_flipflop_tb
# Loading work.JK_flipflop
add wave -position insertpoint sim:/JK_flipflop_tb/*
VSIM 11> run
# 0 1 1 x x
# 1 1 1 x x
# 1 0 1 x x
# 1 1 1 x x
VSIM 12>
```

Task 4: Design D Flip Flop in Verilog

DUT Code:

```
module D_flipflop(D, clk, Q, Qbar);
    input D;
    input clk;
    output Q;
    output Qbar;

    wire nan1;
    wire nan2;
    wire nD;

    not n1(nD, D);

    assign nan1 = ~(D & clk);
    assign nan2 = ~(nD & clk);

    assign Q = ~(nan1 & Qbar);
    assign Qbar = ~(nan2 & Q);
endmodule;
```

Test Code:

```
module D_flipflop_tb();
    reg D;
    reg clk;
    wire Q;
    wire Qbar;

    D_flipflop flipflop(D, clk, Q, Qbar);

    initial begin
        clk = 0;
        D = 0;
        #10
        $display("%b %b %b %b ", D, clk, Q, Qbar);

        clk = 1;
```



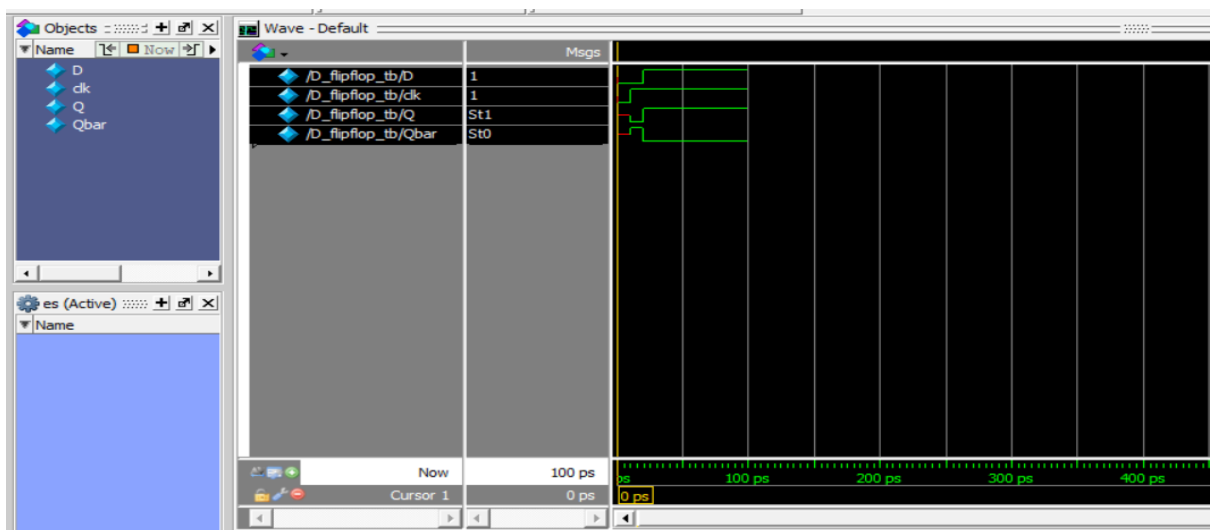
```

        D = 0;
        #10
        $display("%b %b %b %b ", D, clk, Q, Qbar);

        clk =1;
        D = 1;
        #10
        $display("%b %b %b %b", D, clk, Q, Qbar);
    end
endmodule;

```

Output:



```

Transcript
# vsim -gui work.D_flipflop_tb
# Start time: 22:04:17 on Dec 28, 2023
# Loading work.D_flipflop_tb
# Loading work.D_flipflop
add wave -position insertpoint sim:/D_flipflop_tb/*
VSIM 19> run
# 0 0 x x
# 0 1 0 1
# 1 1 1 0
VSIM 20>

```

Task 5: Design T Flip Flop using D Flip Flop in Verilog.

DUT Code:

```

module T_flipflop(T, clk, Q, Qbar);
    input T;
    input clk;
    output Q;
    output Qbar;

```

```

wire D;
wire Q;
xor x1(D, T, Q);
D_flipflop d1(D, clk, Q, Qbar);
endmodule;

```

Test Code:

```

module T_flipflop_tb();
    reg T;
    reg clk;
    wire Q;
    wire Qbar;

    T_flipflop flipflop(T, clk, Q, Qbar);

    initial begin
        $display( "clk  T  Q  Qbar ");

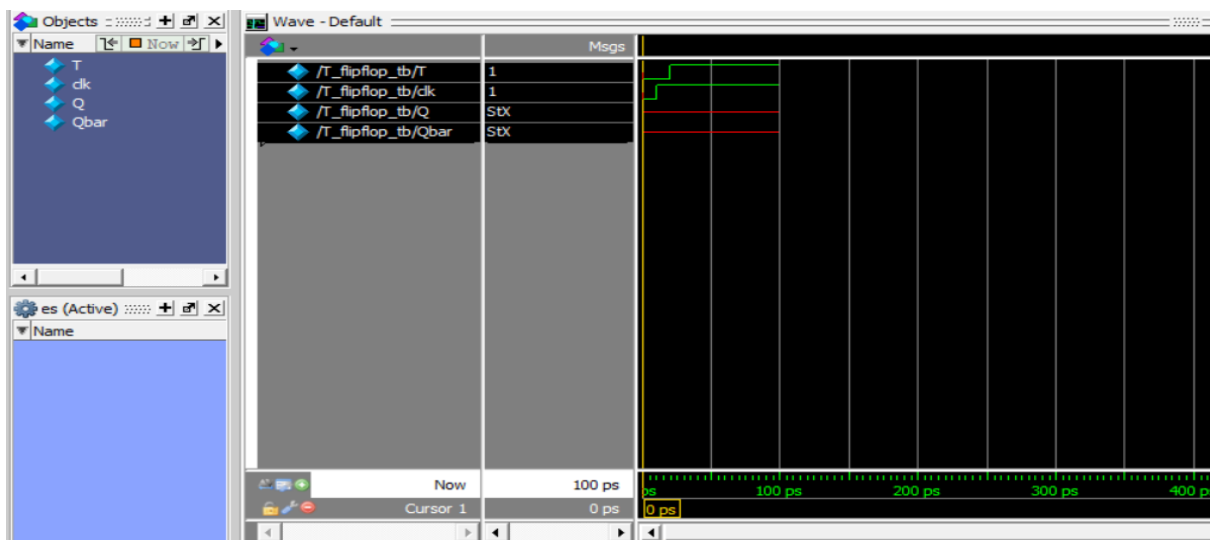
        clk =0;
        T = 0;
        #10
        $display("%b  %b  %b  %b ", clk , T,  Q, Qbar);

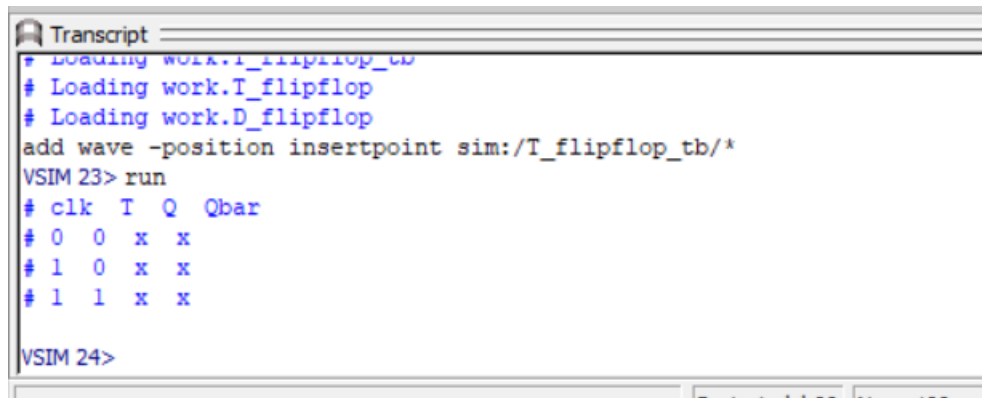
        clk =1;
        T = 0;
        #10
        $display("%b  %b  %b  %b ", clk , T,  Q, Qbar);

        clk =1;
        T = 1;
        #10
        $display("%b  %b  %b  %b", clk , T,  Q, Qbar);
    End
endmodule;

```

Output:





```
Transcript
# Loading work.T_flipflop_tb
# Loading work.T_flipflop
# Loading work.D_flipflop
add wave -position insertpoint sim:/T_flipflop_tb/*
VSIM 23> run
# clk T Q Qbar
# 0 0 x x
# 1 0 x x
# 1 1 x x
VSIM 24>
```

Reference:

To view my codes, please refer to my [GitHub Account](#).

Conclusion:

In conclusion, I have learned how to work in Verilog to implement flip flops and latches of various types. Now I am able to use my founding's in real world problems solving strategies.

The End.