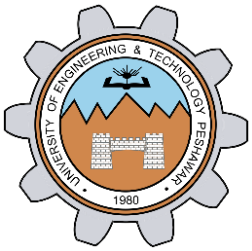
COUNTERS IN VERILOG

LAB # 10



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:

30th 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 4 BIT counter.

Code:

DUT Code:

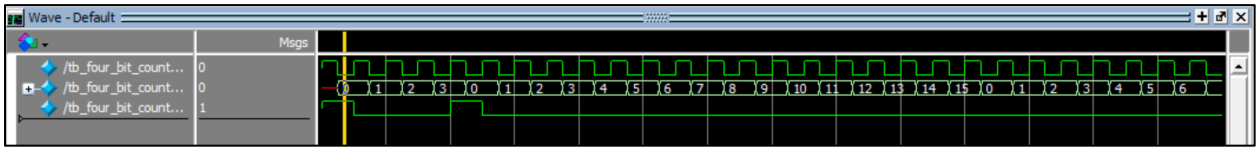
```
module four_bit_counter_behaviour(rst, Clk, out);

    input rst;
    input Clk;

    output reg [3:0]out;


    always@(negedge Clk)

        if (rst)
            out <= 1'b0;

        else
            out <= out + 1;


endmodule
```

Test bench Code:

```
module tb_four_bit_counter_behaviour();

    reg rst;
    reg Clk;

    wire [3:0]out;

    four_bit_counter_behaviour my_four_bit_counter_behaviour(.rst(rst), .Clk(Clk), .out(out));

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

    initial begin

    rst = 1;
    Clk = 1;
    #10

    rst = 0;
    //Clk = 0;
    #10

```

```
22
23      rst = 0;
24      //Clk = 1;
25      #10
26
27      rst = 0;
28      //Clk = 0;
29      #10
30
31      rst = 1;
32      //Clk = 1;
33      #10
34
35      rst = 0;
36
37      end
38
39
40  endmodule
```

Output:



Task 2:

Write a Verilog code to implement 8 BIT counter.

Code:

DUT Code:

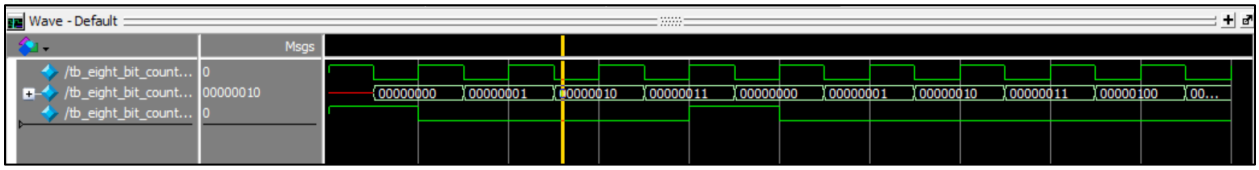
```
1   module eight_bit_counter_behaviour(rst, Clk, out);
2
3       input rst;
4       input Clk;
5
6       output reg [7:0]out;
7
8
9       always@(negedge Clk)
10
11      if (rst)
12          out <= 1'b0;
13
14      else
15          out <= out + 1;
16
17
18  endmodule
```

Test bench Code:

```
module tb_eight_bit_counter_behaviour();

    reg rst;
    reg Clk;

    wire [7:0]out;

    eight_bit_counter_behaviour my_eight_bit_counter_behaviour(.rst(rst), .Clk(Clk), .out(o

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

    initial begin

    rst = 1;
    Clk = 1;
    #10

    rst = 0;
    //Clk = 0;
    #10
```

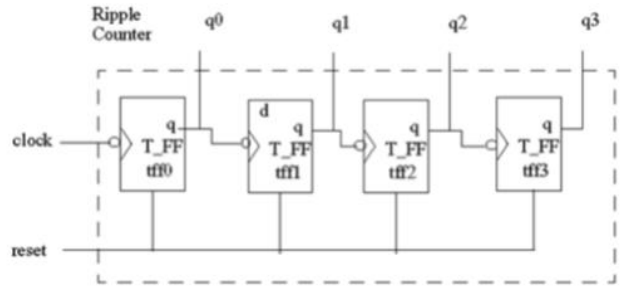
```
    rst = 0;
    //Clk = 1;
    #10

    rst = 0;
    //Clk = 0;
    #10

    rst = 1;
    //Clk = 1;
    #10

    rst = 0;

    end
endmodule
```

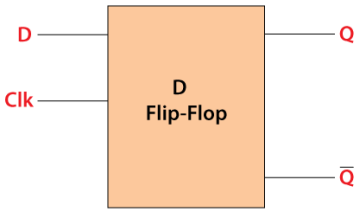
Output:

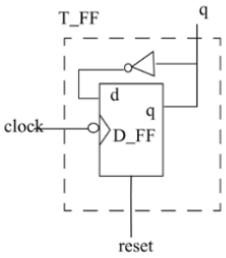


Task 3:

Implement 4 BIT Uncontrolled Asynchronous UP COUNTER in Verilog using T FLIP FLOP:







Code:

DUT Code:

T_FF.v | tb_T_FF.v | task3.v | tb_task3.v | tb_jkff_behavior.v | pc.v | mar.v | ram.v

```
1   module task3(rst, Clk, out, QBar);
2
3       input rst;
4       input Clk;
5
6       output [3:0]out;
7       output [3:0]QBar;
8
9
10      U_T_FF f1(rst, Clk, out[0], QBar[0]);
11      U_T_FF f2(rst, out[0] , out[1], QBar[1]);
12      U_T_FF f3(rst, out[1], out[2], QBar[2]);
13      U_T_FF f4(rst, out[2], out[3], QBar[3]);
14
15      //U_T_FF f4(rst, n[2], n[3]);
16      //U_T_FF f3(rst, n[1], n[2]);
17      //U_T_FF f2(rst, n[0] , n[1]);
18      //U_T_FF f1(rst, Clk, n[0]);
19
20      //assign out = n;
21
22  endmodule
```

T_FF.v | tb_T_FF.v | task3.v | tb_task3.v | tb_jkff_behavior.v | pc.v | mar.v | ram.v

```
1   module U_T_FF(rst, Clk, Q, QBar);
2
3           input rst;
4           input Clk;
5
6           output Q;
7           output QBar;
8
9           wire n1;
10
11          not n(n1, Q);
12          D_FF my_D_FF(n1, rst, Clk, Q, QBar);
13
14  endmodule
```

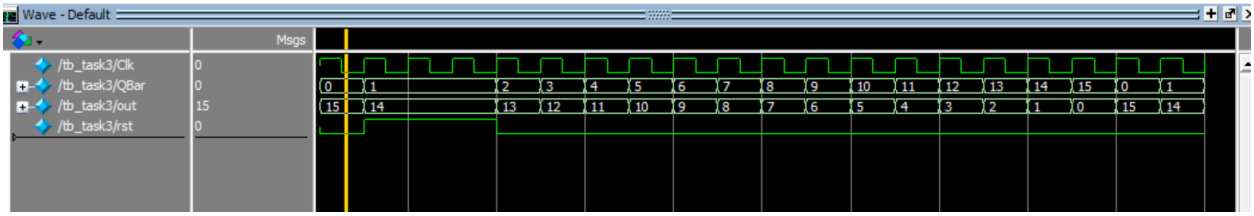
D_FF.v | tb_D_FF.v | T_FF.v | tb_T_FF.v | task3.v | tb_task3.v | tb

```
2
3       input D;
4       input Clk;
5       input rst;
6       output reg Q = 0;
7       output reg QBar = 1;
8
9
10      always@(posedge Clk)
11
12          if (rst) begin
13                  Q <= 1'b0;
14                  QBar <= 1'b1;
15              end
16
17          else begin
18                  Q <= D;
19                  QBar <= ~D;
20              end
21  endmodule
```

Test bench Code:

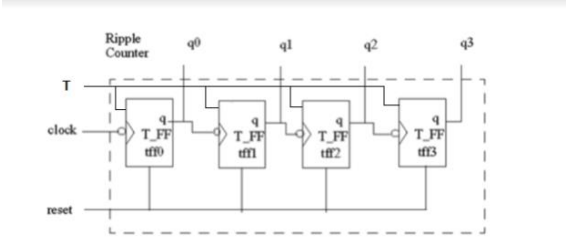
```
module tb_task3();

    reg rst;
    reg Clk;
    wire [3:0]out;
    wire [3:0]QBar;
    task3 my_task3(.rst(rst), .Clk(Clk), .out(out), .QBar(QBar));

    always #5 Clk = ~Clk;

    initial begin
    rst = 0;
    Clk = 1;
    #10

    rst = 1;
    #30

    rst = 0;
    end

endmodule
```

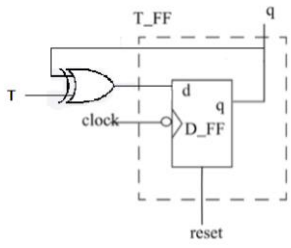
Output:



Task 4:

Implement 4 BIT Controlled Asynchronous UP COUNTER in Verilog using T FLIP FLOP:





Code:

DUT Code:

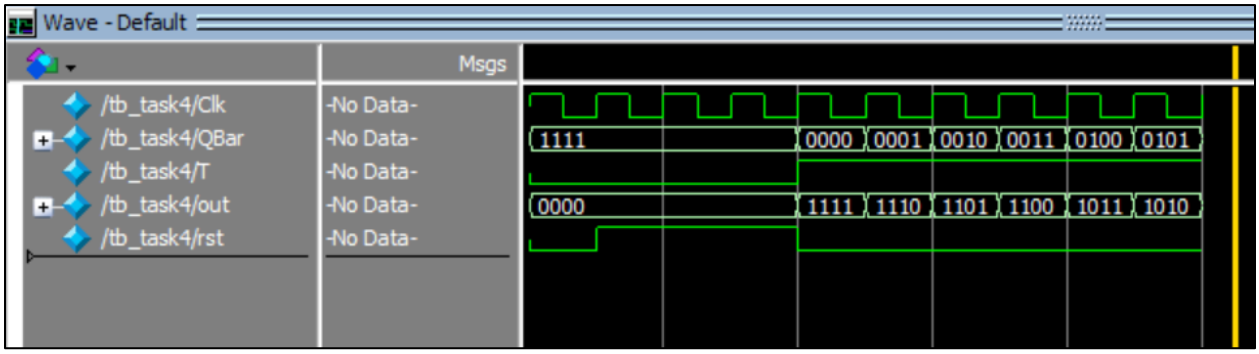
```
module task4(T, rst, Clk, out, QBar);

    input rst;
    input Clk;
    input T;

    output [3:0]out;
    output [3:0]QBar;


    C_T_FF f1(T, rst, Clk, out[0], QBar[0]);
    C_T_FF f2(T, rst, out[0] , out[1], QBar[1]);
    C_T_FF f3(T, rst, out[1], out[2], QBar[2]);
    C_T_FF f4(T, rst, out[2], out[3], QBar[3]);

endmodule
```

```
module C_T_FF(T, rst, Clk, Q, QBar);

        input rst;
        input Clk;
        input T;
        output Q;
        output QBar;

        wire n1;

        xor n(n1, Q, T);
        D_FF my_D_FF(n1, rst, Clk, Q, QBar);

endmodule
```

Test bench Code:

```
module tb_task4();
    reg rst, Clk, T;
    wire [3:0]out;
    wire [3:0]QBar;
    task4 my_task4(.T(T), .rst(rst), .Clk(Clk), .out(out), .QBar(QBar));

    always #5 Clk = ~Clk;

    initial begin
    T = 0;
    rst = 0;
    Clk = 1;
    #10

    T = 0;
    rst = 1;
    #30

    T = 1;
    rst = 0;
    end
endmodule
```

Output:



Task 5:

Implement 16*8 RAM in Verilog.

Code:

DUT Code:

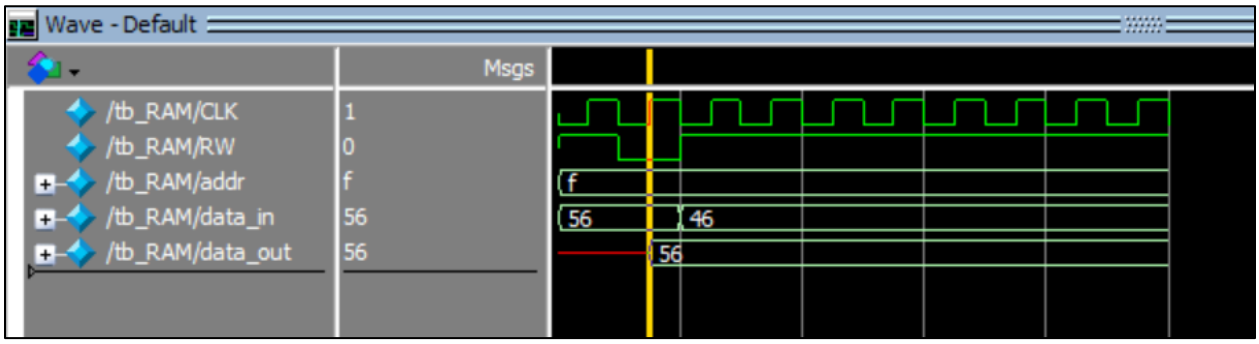
```
module RAM(CLK, RW, data_in, addr, data_out);

    input CLK, RW;
    input [7:0]data_in;
    input [3:0]addr ;
    output reg [7:0]data_out;
    reg[7:0] mem[15:0];

    always @(posedge CLK)
        if(RW)
            mem[addr] = data_in;
        else
            data_out = mem[addr];
endmodule

```

```
module RAM_TwoPorts(CLK, R, W, data_in, addr, data_out);

    input CLK, R, W;
    input [7:0]data_in;
    input [3:0]addr ;
    output reg [7:0]data_out;
    reg[7:0] mem[15:0];

    always @(posedge CLK)
        if(W)
            mem[addr] = data_in;
        else if(R)
            data_out = mem[addr];
endmodule
```

Test bench Code:

```
module tb_RAM();
    reg CLK = 0;
    reg RW;
    reg [7:0]data_in;
    reg [3:0]addr ;
    wire [7:0]data_out;

    RAM ram(.CLK(CLK), .RW(RW), .data_in(data_in), .addr(addr), .data_out(data_out));

    always #5 CLK = ~CLK;

    initial begin
        data_in = 8'h56;
        RW = 1;
        addr = 4'hF;
        #10


        RW = 0;
        addr = 4'hF;
        #10

        data_in = 8'h46;
        RW = 1;
        addr = 4'hF;
        #10

    end

endmodule
```

Output:



Conclusion:

In this lab, I learned how to implement Counters and RAM in Verilog.