An-Najah National University
Faculty of Engineering and IT

جامعة النجاح الوطنية
كلية الهندسة وتكنولوجيا المعلومات

Electrical and Computer Engineering Department
Digital Circuit Design II Lab : (10636391)

Report Grading Sheet

Instructor Name: Dr. Ashraf Armoush	**Experiment #: 3**
Academic Year: 2024/2025	**Performed on: 22/2/2024**
Semester: 2st semester	**Submitted on:**

Student Names:

1-Ahmad Ashayer	2-Asem Diab
3-	4-
5-	6-

Evaluation Criterion	CLO	Grade	Points
Abstract and Aims Aims and idea of the experiment are clearly stated in simple words		10	
Introduction, Apparatus and Procedures Introduction is complete and well-written, all grammar/spelling correct, Appropriate background information related to the principles of the experiment is provided. The list of apparatus and procedures are also provided		15	
Experimental Results, Calculations and Discussion Results analyzed correctly. Experimental findings adequately and specifically summarized, in graphical, tabular, and/or written form. Comparison of theoretical predictions to experimental results, including discussion of accuracy and error analysis as needed.		50	
Conclusions Conclusions summarize the major findings from the experimental results with adequate specificity. Highlighting the most important results		15	
Appearance Title page is complete, page numbers applied, content is well organized, correct spelling, fonts are consistent, good visual appeal. You have also to use reference for the information you provide		10	
Total		100	

-Abstract and Aims-

This experiment has four parts, all of them implemented in verlog, the first one is designing 4-bit Asynchronous counter by designing four T-Flip Flops, second one is designing 4-bit Synchronous updown counter, additionally for these two parts we need to design clock divider, for the third part designing button debouncing to use it as a trigger for updown counter, finally designing two digit BCD counter and show the digits by using two 7-segments display, for this part also it needed decoder and time multiplexer.

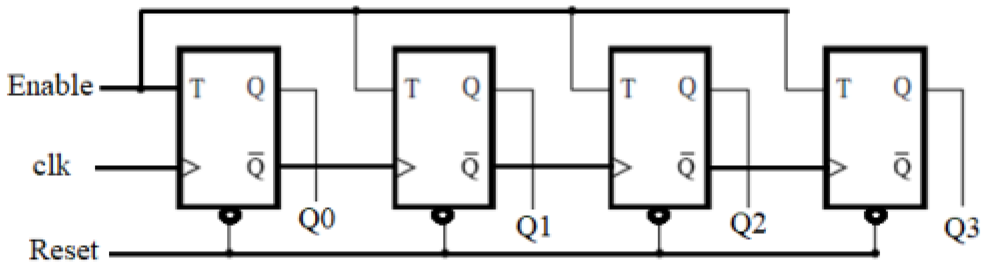
-Introduction-

4-bit Asynchronous Up Counter

First for this part designing this counter needs four T-Flip Flops and this is the truth table for it:-

T	Clear	Q
X	0	0
0	1	Q
1	1	~Q

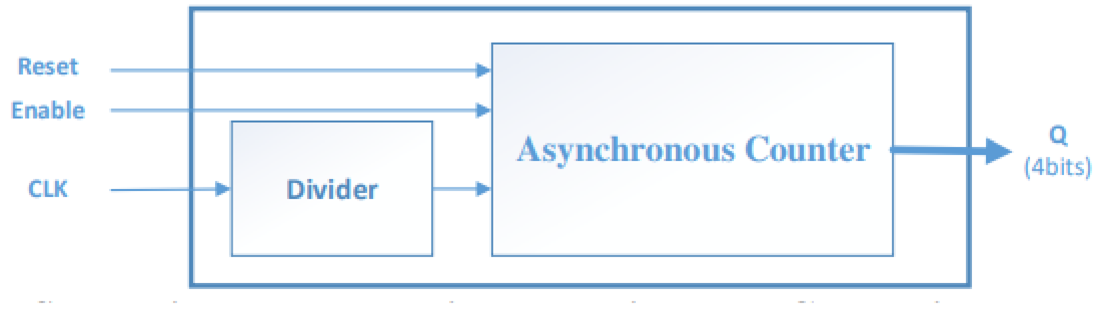
And this is the figure for the Asynchronous counter:-



Clock Divider

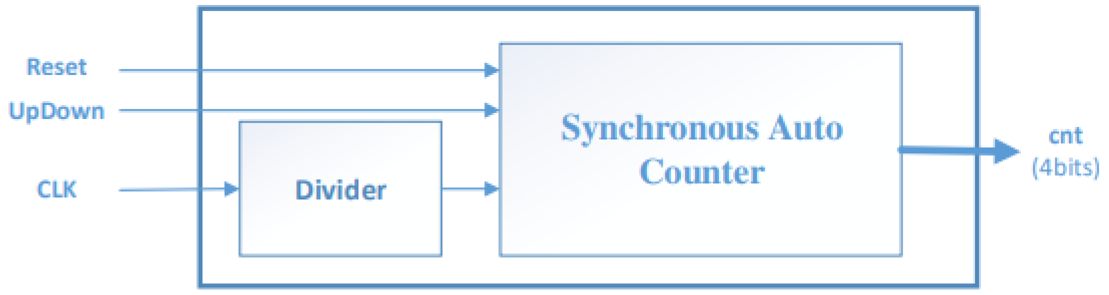
This circuit for the design clock, it take the clock as input and decrease it’s frequency as output, and for this part the clock frequency is 100MHz and the output clock is 1Hz, so for some calculations the output of the counter which is maximum value will be 50M

Top-Level Module



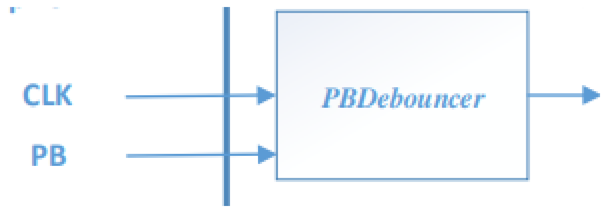
4-bit Synchronous Auto Updown Counter

For this second part, the implemented Synchronous counter has up and down situations, so it implemented by using arithmetic operators like + and – , and for this part it designed in behavioral description, and also as before it used the clock divider module for decreasing the frequency for the input clock as shown in the **top level module**:-



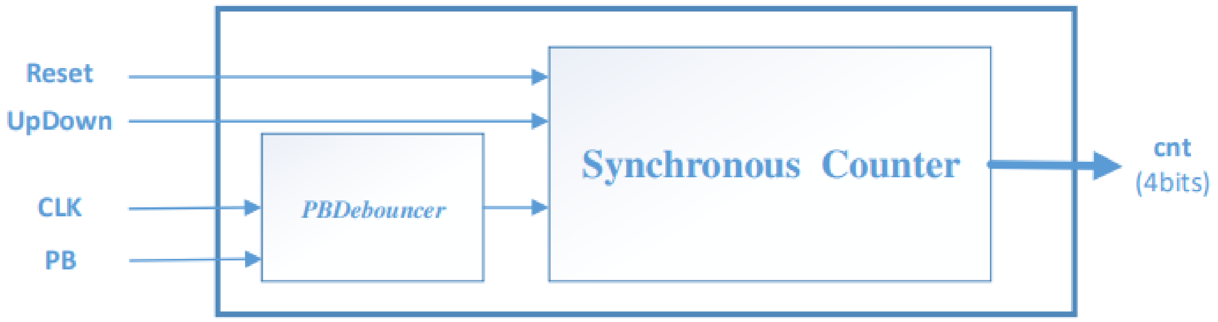
Push Button UpDown Counter

For this part we implemented the counter but with additional part which is push button for controlling the counter, but before that there is one single problem with this circuit which is **Bouncing,** this appears when clicking on button there is a noise that making an effect to the design, and to solve this issue we make a software debouncing solution to avoid this noise, and this is the module for this circuit:-



It takes two input and gives one output, for this solution, inside this block we determine the interval for that noise and simple making some delay for solving the issue.

Finally this is the top-level module for that counter:-



Two-Digit BCD Counter

For this final part, the BCD counter includes many blocks, some of them like Clock-Divider and this block has been implemented both in Asynchronous and Syncron us counters, and also it needs some additional blocks each one has it's objective.

BCD Counter

This digital circuit has counter from 0 to 9, but in this part we need two show the counter between 00 and 99, for the up situation, when the first digit reaches 9, the second digit incremented by 1 and the first will be 0.

And for the down situation when the first digit reach's 0, the second one will decremented by one and the first one will be 9.

Time Multiplexer

This digital circuit for displaying each digit in the same time, it takes the two digits as inputs and gives two output, one of them is the BCD digit itself, the other is Digit selector.

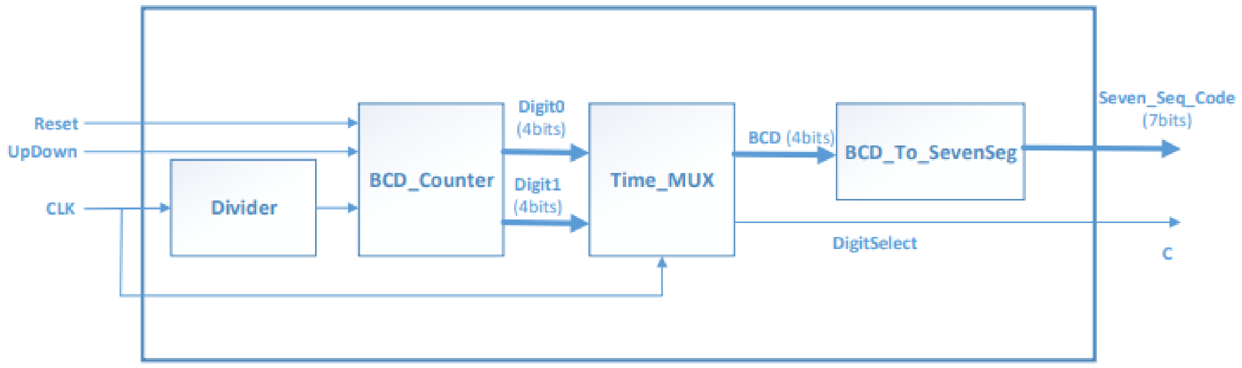
BCD to 7-segment Decoder

This circuit for decode the 4 bit digit counter to 7 outputs each one is related to the 7-segment display.

And this is the truth table that needed for this part of experiment:-

	A	B	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	0	0	1	1
defalut	0	0	0	0	0	0	0

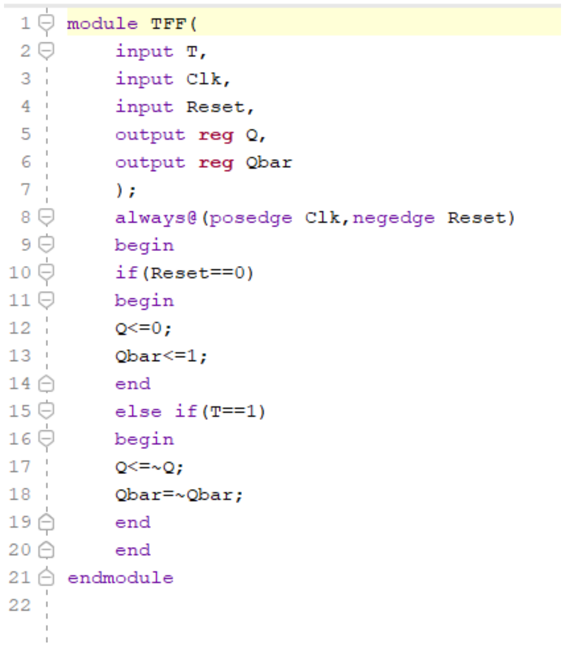
Top-Level Module



- Experimental Results –

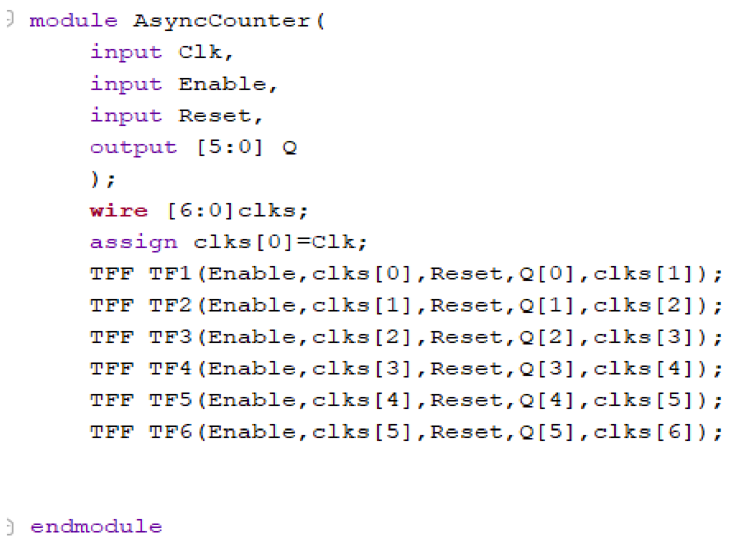
Part one : 4-bit Asynchronous Up Counter

Code:



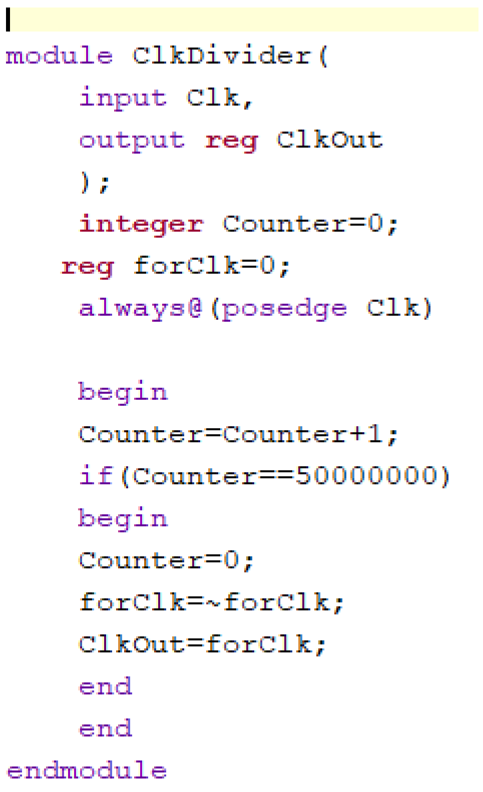
```
module TFF(
    input T,
    input Clk,
    input Reset,
    output reg Q,
    output reg Qbar
    );
    always@(posedge Clk,negedge Reset)
    begin
    if(Reset==0)
    begin
    Q<=0;
    Qbar<=1;
    end
    else if(T==1)
    begin
    Q<=~Q;
    Qbar=~Qbar;
    end
    end
endmodule
```

Figure 1: T Flip Flop



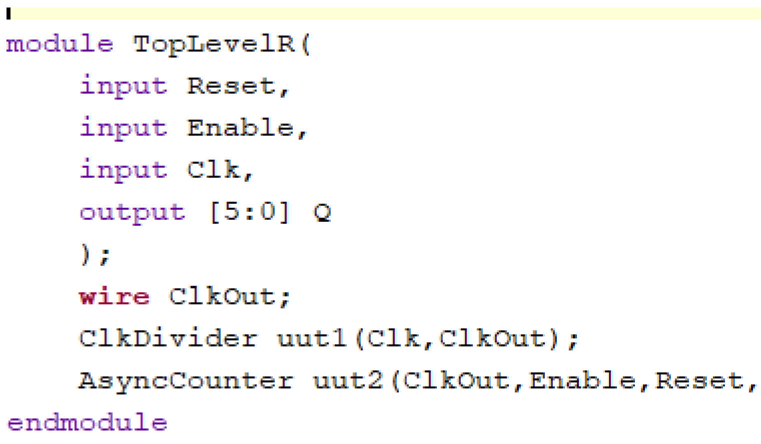
```
module AsyncCounter(
    input Clk,
    input Enable,
    input Reset,
    output [5:0] Q
    );
    wire [6:0]clks;
    assign clks[0]=Clk;
    TFF TF1(Enable,clks[0],Reset,Q[0],clks[1]);
    TFF TF2(Enable,clks[1],Reset,Q[1],clks[2]);
    TFF TF3(Enable,clks[2],Reset,Q[2],clks[3]);
    TFF TF4(Enable,clks[3],Reset,Q[3],clks[4]);
    TFF TF5(Enable,clks[4],Reset,Q[4],clks[5]);
    TFF TF6(Enable,clks[5],Reset,Q[5],clks[6]);


endmodule
```

Figure 2 : Asyncous Counter



```
module ClkDivider(
    input Clk,
    output reg ClkOut
    );
    integer Counter=0;
   reg forClk=0;
    always@(posedge Clk)

    begin
    Counter=Counter+1;
    if(Counter==50000000)
    begin
    Counter=0;
    forClk=~forClk;
    ClkOut=forClk;
    end
    end
endmodule
```

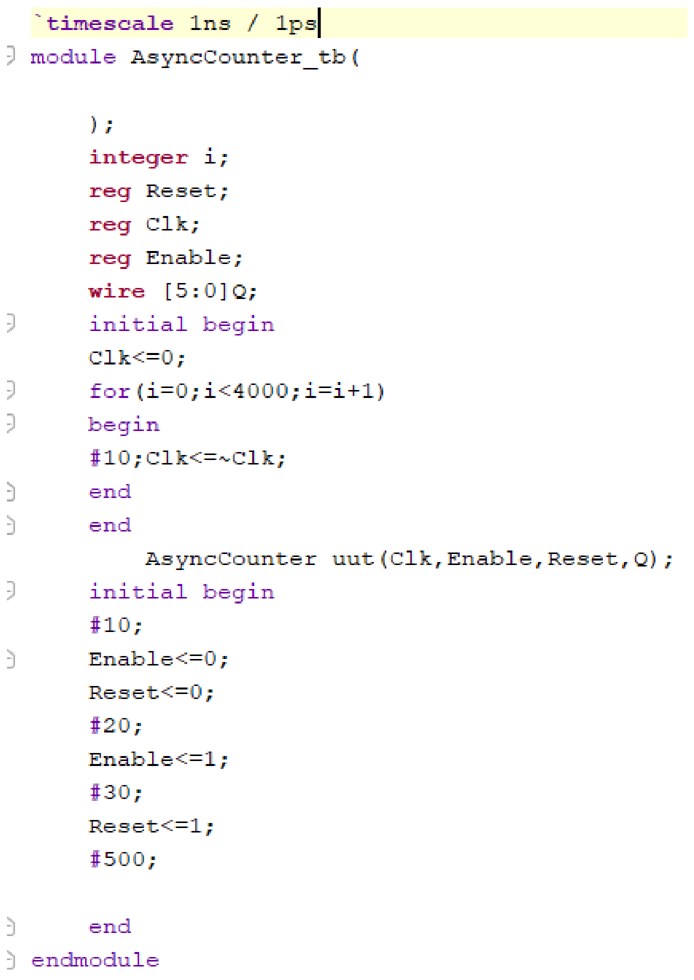
Figure 3 : Clock Divider



```
module TopLevelR(
    input Reset,
    input Enable,
    input Clk,
    output [5:0] Q
    );
    wire ClkOut;
    ClkDivider uut1(Clk,ClkOut);
    AsyncCounter uut2(ClkOut,Enable,Reset,
endmodule
```

Figure 4 : Top Level

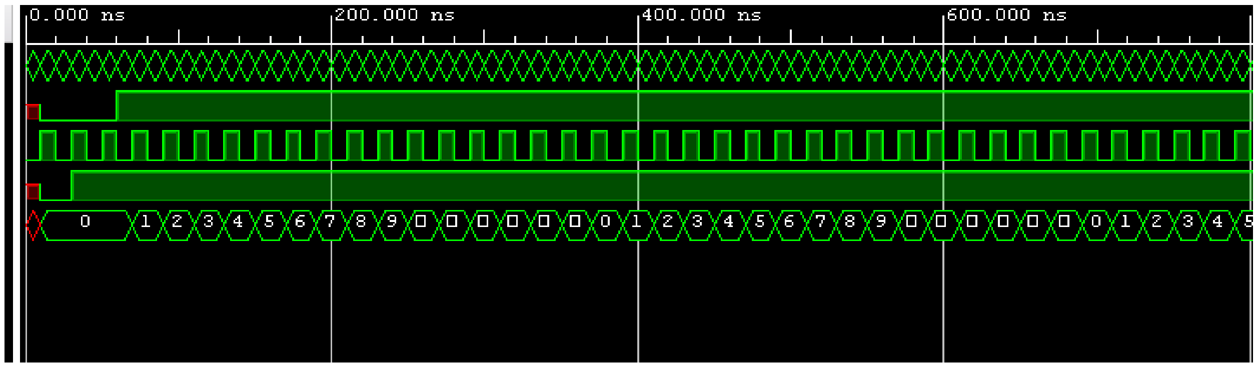
Testbench:



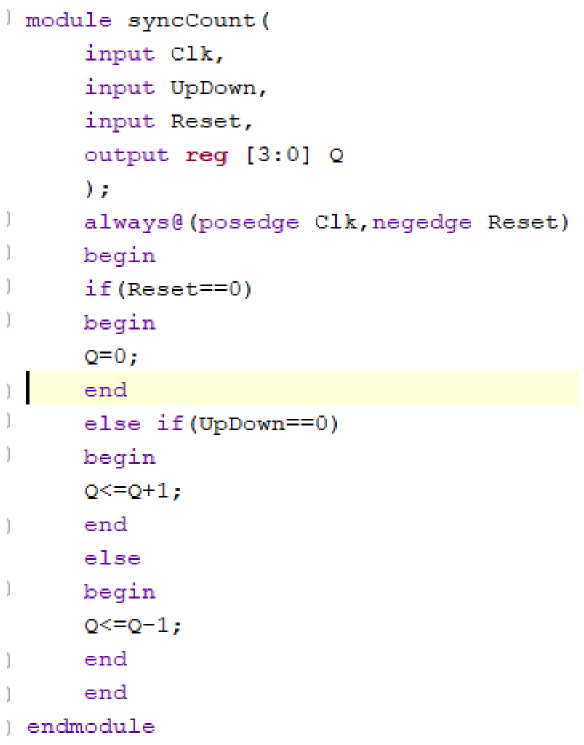
```
`timescale 1ns / 1ps
module AsyncCounter_tb(

    );
    integer i;
    reg Reset;
    reg Clk;
    reg Enable;
    wire [5:0]Q;
    initial begin
    Clk<=0;
    for(i=0;i<4000;i=i+1)
    begin
    #10;Clk<=~Clk;
    end
    end
        AsyncCounter uut(Clk,Enable,Reset,Q);
    initial begin
    #10;
    Enable<=0;
    Reset<=0;
    #20;
    Enable<=1;
    #30;
    Reset<=1;
    #500;

    end
endmodule
```

Figure 5 : testbench of asyc counter

Wave :

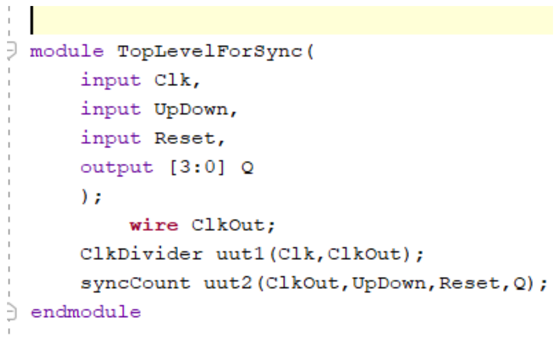


Part two : 4-bit synchronous Up/down Counter



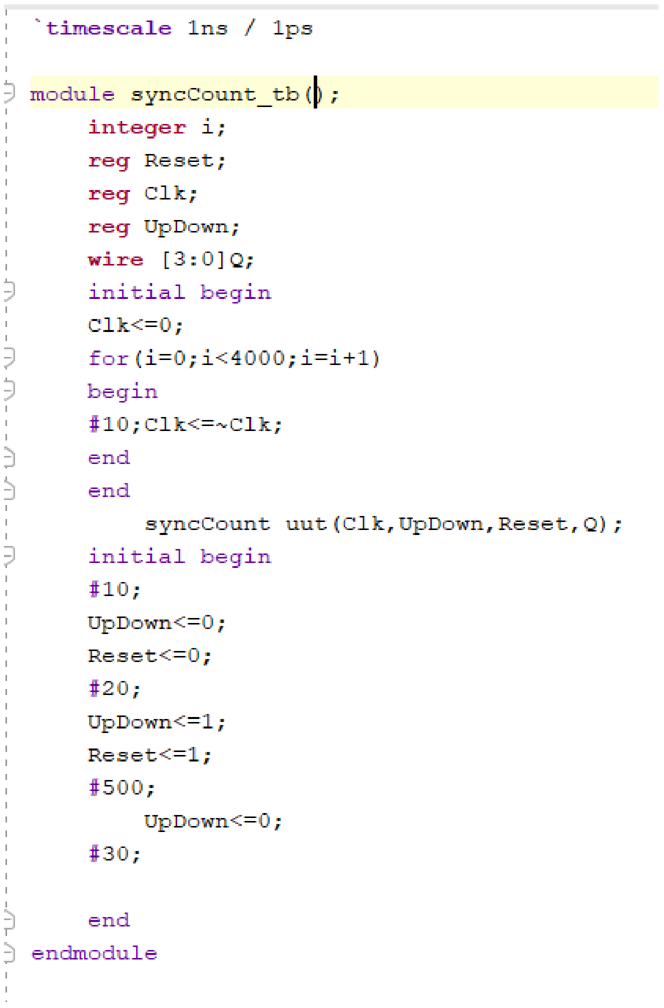
```
module syncCount(
     input Clk,
     input UpDown,
     input Reset,
     output reg [3:0] Q
     );
     always@(posedge Clk,negedge Reset)
     begin
     if(Reset==0)
     begin
     Q=0;
     end
     else if(UpDown==0)
     begin
     Q<=Q+1;
     end
     else
     begin
     Q<=Q-1;
     end
     end
endmodule
```

Figure 6 : syncrous up/down counter



```
module TopLevelForSync(
     input Clk,
     input UpDown,
     input Reset,
     output [3:0] Q
     );
         wire ClkOut;
     ClkDivider uut1(Clk,ClkOut);
     syncCount uut2(ClkOut,UpDown,Reset,Q);
endmodule
```

Figure 7: top level module



```
`timescale 1ns / 1ps

module syncCount_tb();
     integer i;
     reg Reset;
     reg Clk;
     reg UpDown;
     wire [3:0]Q;
     initial begin
     Clk<=0;
     for(i=0;i<4000;i=i+1)
     begin
     #10;Clk<=~Clk;
     end
     end
         syncCount uut(Clk,UpDown,Reset,Q);
     initial begin
     #10;
     UpDown<=0;
     Reset<=0;
     #20;
     UpDown<=1;
     Reset<=1;
     #500;
         UpDown<=0;
     #30;

     end
endmodule
```

Figure 8: testbench

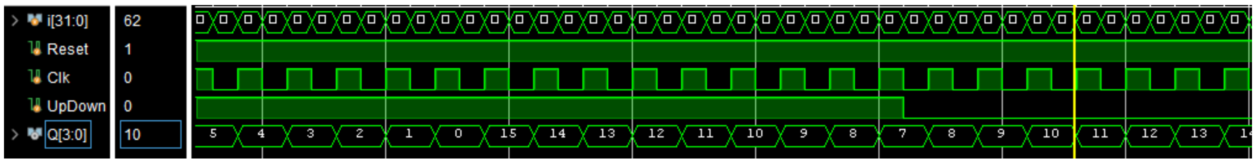
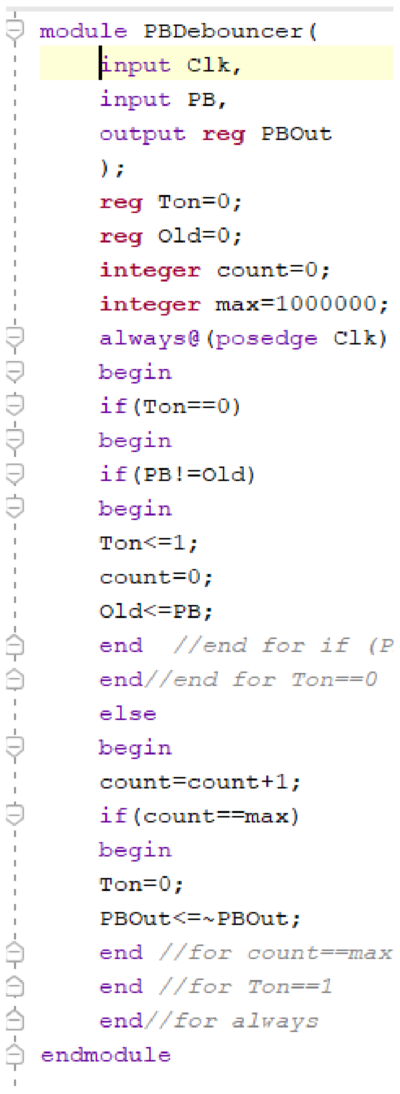


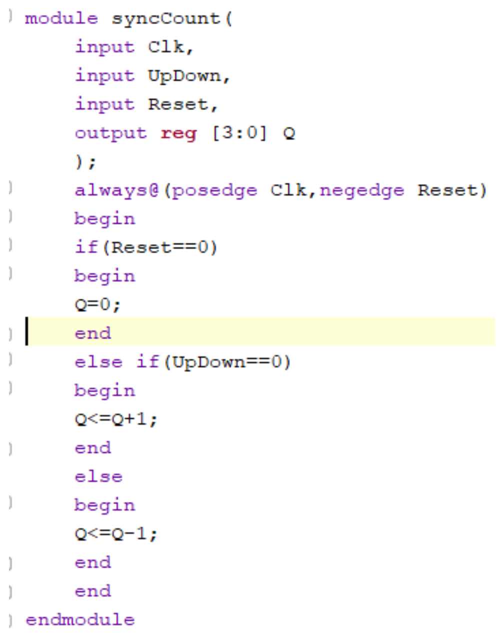
Figure 9: wave of sumlation

Part three : Push Button UpDown Counter



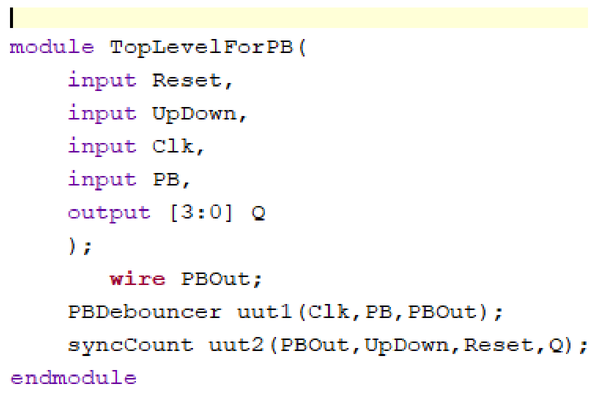
```
module PBDebouncer(
     input Clk,
     input PB,
     output reg PBOut
     );
     reg Ton=0;
     reg Old=0;
     integer count=0;
     integer max=1000000;
     always@(posedge Clk)
     begin
     if(Ton==0)
     begin
     if(PB!=Old)
     begin
     Ton<=1;
     count=0;
     Old<=PB;
     end  //end for if (P
     end//end for Ton==0
     else
     begin
     count=count+1;
     if(count==max)
     begin
     Ton=0;
     PBOut<=~PBOut;
     end //for count==max
     end //for Ton==1
     end//for always
endmodule
```

Figure 10 : push button debouncer



```
module syncCount(
     input Clk,
     input UpDown,
     input Reset,
     output reg [3:0] Q
     );
     always@(posedge Clk,negedge Reset)
     begin
     if(Reset==0)
     begin
     Q=0;
     end
     else if(UpDown==0)
     begin
     Q<=Q+1;
     end
     else
     begin
     Q<=Q-1;
     end
     end
endmodule
```

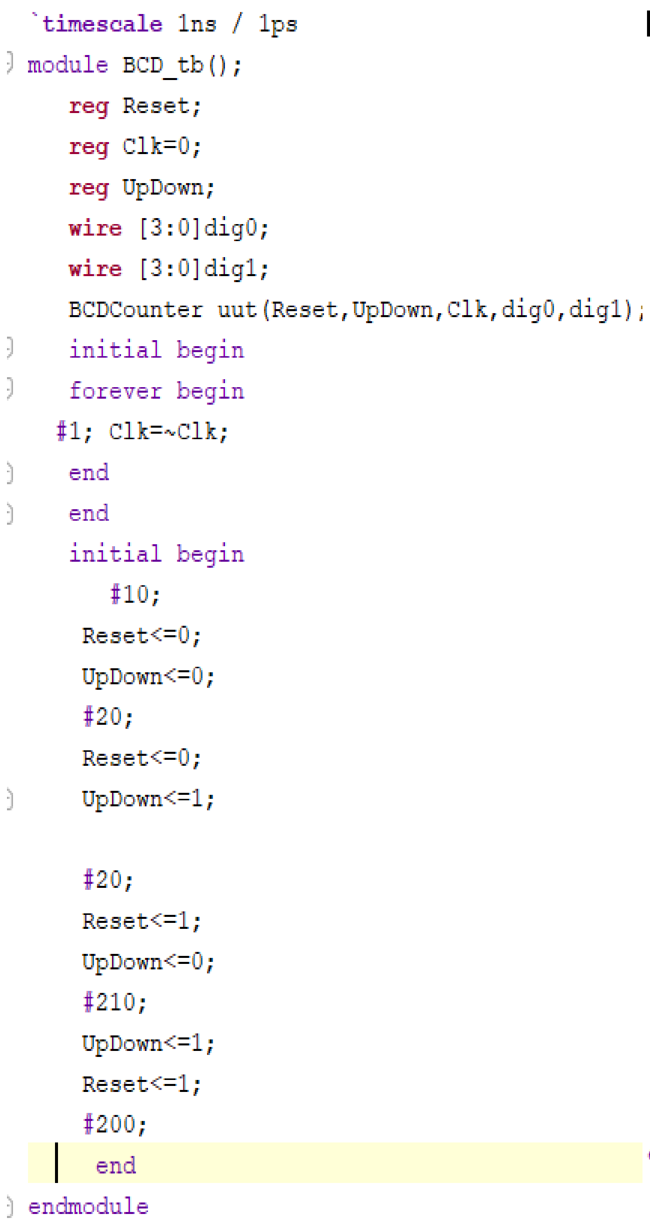
Figure 11:syncrous counter implemention



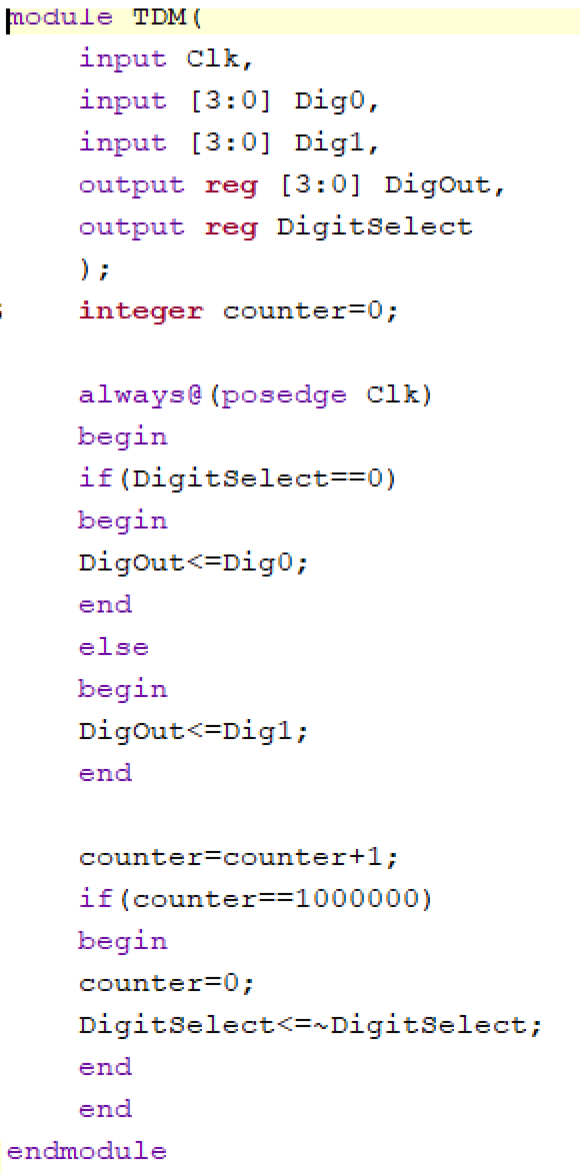
```
module TopLevelForPB(
     input Reset,
     input UpDown,
     input Clk,
     input PB,
     output [3:0] Q
     );
        wire PBOut;
     PBDebouncer uut1(Clk,PB,PBOut);
     syncCount uut2(PBOut,UpDown,Reset,Q);
endmodule
```

Figure 12 : top level of part 3

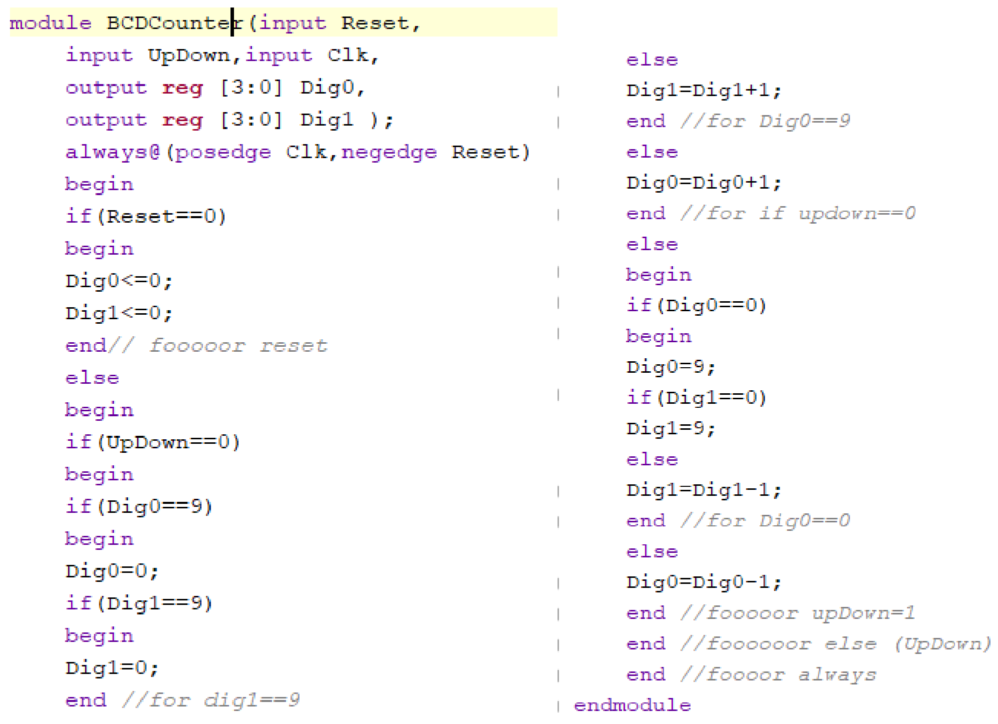
Part 4: Two-Digit BCD Counter



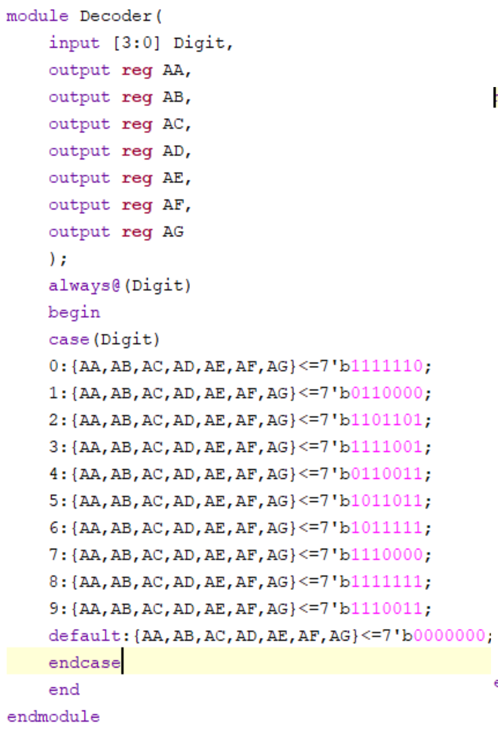
```
 `timescale 1ns / 1ps
module BCD_tb();
   reg Reset;
   reg Clk=0;
   reg UpDown;
   wire [3:0]dig0;
   wire [3:0]dig1;
   BCDCounter uut(Reset,UpDown,Clk,dig0,dig1);
   initial begin
   forever begin
  #1; Clk=~Clk;
   end
   end
   initial begin
      #10;
    Reset<=0;
    UpDown<=0;
    #20;
    Reset<=0;
    UpDown<=1;

    #20;
    Reset<=1;
    UpDown<=0;
    #210;
    UpDown<=1;
    Reset<=1;
    #200;
     end
endmodule
```



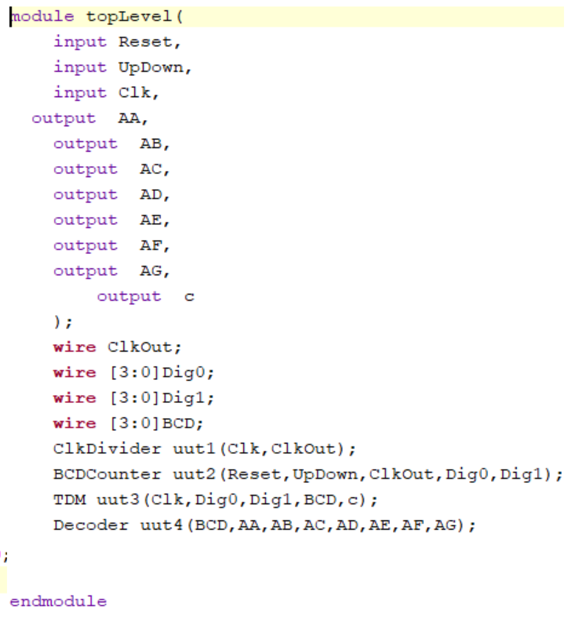
```
module TDM(
    input Clk,
    input [3:0] Dig0,
    input [3:0] Dig1,
    output reg [3:0] DigOut,
    output reg DigitSelect
    );
    integer counter=0;

    always@(posedge Clk)
    begin
    if(DigitSelect==0)
    begin
    DigOut<=Dig0;
    end
    else
    begin
    DigOut<=Dig1;
    end

    counter=counter+1;
    if(counter==1000000)
    begin
    counter=0;
    DigitSelect<=~DigitSelect;
    end
    end
endmodule
```



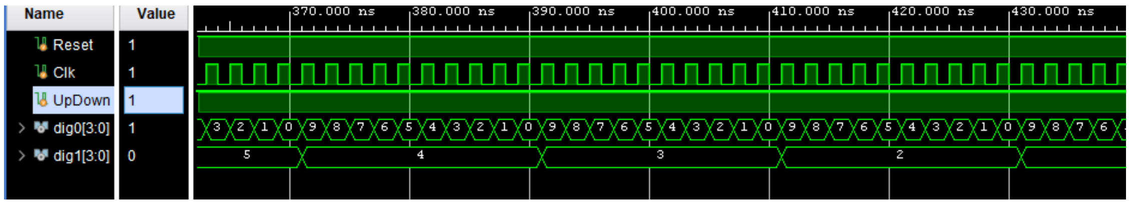
```
module BCDCounter(input Reset,
    input UpDown,input Clk,
    output reg [3:0] Dig0,
    output reg [3:0] Dig1 );
    always@(posedge Clk,negedge Reset)
    begin
    if(Reset==0)
    begin
    Dig0<=0;
    Dig1<=0;
    end// fooooor reset
    else
    begin
    if(UpDown==0)
    begin
    if(Dig0==9)
    begin
    Dig0=0;
    if(Dig1==9)
    begin
    Dig1=0;
    end //for dig1==9
    else
    Dig1=Dig1+1;
    end //for Dig0==9
    else
    Dig0=Dig0+1;
    end //for if updown==0
    else
    begin
    if(Dig0==0)
    begin
    Dig0=9;
    if(Dig1==0)
    Dig1=9;
    else
    Dig1=Dig1-1;
    end //for Dig0==0
    else
    Dig0=Dig0-1;
    end //foooooor upDown=1
    end //fooooooor else (UpDown)
    end //foooor always
endmodule
```

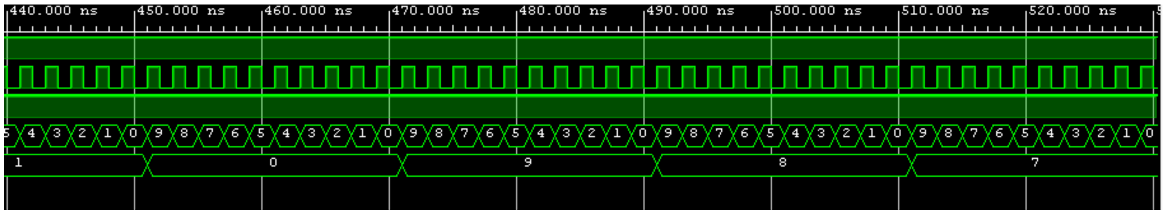


```
module Decoder(
    input [3:0] Digit,
    output reg AA,
    output reg AB,
    output reg AC,
    output reg AD,
    output reg AE,
    output reg AF,
    output reg AG
    );
    always@(Digit)
    begin
    case(Digit)
    0:{AA,AB,AC,AD,AE,AF,AG}<=7'b1111110;
    1:{AA,AB,AC,AD,AE,AF,AG}<=7'b0110000;
    2:{AA,AB,AC,AD,AE,AF,AG}<=7'b1101101;
    3:{AA,AB,AC,AD,AE,AF,AG}<=7'b1111001;
    4:{AA,AB,AC,AD,AE,AF,AG}<=7'b0110011;
    5:{AA,AB,AC,AD,AE,AF,AG}<=7'b1011011;
    6:{AA,AB,AC,AD,AE,AF,AG}<=7'b1011111;
    7:{AA,AB,AC,AD,AE,AF,AG}<=7'b1110000;
    8:{AA,AB,AC,AD,AE,AF,AG}<=7'b1111111;
    9:{AA,AB,AC,AD,AE,AF,AG}<=7'b1110011;
    default:{AA,AB,AC,AD,AE,AF,AG}<=7'b0000000;
    endcase
    end
endmodule
```



```
module topLevel(
    input Reset,
    input UpDown,
    input Clk,
  output  AA,
    output  AB,
    output  AC,
    output  AD,
    output  AE,
    output  AF,
    output  AG,
        output  c
    );
    wire ClkOut;
    wire [3:0]Dig0;
    wire [3:0]Dig1;
    wire [3:0]BCD;
    ClkDivider uut1(Clk,ClkOut);
    BCDCounter uut2(Reset,UpDown,ClkOut,Dig0,Dig1);
    TDM uut3(Clk,Dig0,Dig1,BCD,c);
    Decoder uut4(BCD,AA,AB,AC,AD,AE,AF,AG);

endmodule
```





Conclusions -

In this experiment we learned how to implement asynchronous counter by using flips-flops circuit, and also implementing synchronous counter and for both we implemented clock divider for the system design clock,

Also we learned how to deal with bouncing problem in the lap and solving the noise issue when we use push button,

Finally we learned how to make BCD counter and how to show it in multiple 7-segments display.