FACULTY OF ENGINEERING AND TECHNOLOGY

ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT

ADVANCED DIGITAL DESIGN

ENCS3310

Prepared by: Shahd Ali

ID Number: 1200183

Instructor: Dr. Abdellatif Abu-Issa

Section: 1

Date: 4/7/2023

Table Of Contacts

Brief introduction and background

1) Brief introduction

In this project we build a structural circuit using T-Flip Flops and combination logic, which can count either Prime numbers or Fibonacci depends on a value of an input. Also, it can count them either up or down depends on the value of another input. The counter should count the first 11 numbers of any sequence.

e.g. (first 11 numbers) - Prime Numbers 2,3,5,7,11,13,17,19,23,29,31

-Fibonacci Numbers 0,1,1,2,3,5,8,13,21,34,55

We should have some inputs to controls on our counter:

- Mode: this input will let me know any type of numbers I count so if (mode == 0) then we count prime numbers and if (mode ==1) then we count Fibonacci numbers.
- Reset: this input will let me return to the first value that mean if I 'am on prime numbers and click on reset then I will back to value 2.
- Enable: this input if (en == 0) then it will count the same number that I stop on it and if (en ==1) then we will continue the counting normally.
- Up-down: this input will count up when (up-down==0) and will count down when (up-down ==1)

We only need to build on T-flip flop and other gates (and, or, Nand…), there is on figure (1) what we have do.

Figure 1

2) Background about how I think

I divide my work for 4 parts every part have the same 6 inputs and different by 2 inputs which are mode and up-down.

Why 6 inputs? Because the highest number I have is 55 and in binary is (110111) so it's 6 bits.

In general it will be 8 inputs for every part and 6 output, so I did truth tables and k-map for every part and take their equation and use it in my coding.

That is it very easy and simple!

If we have logically error then we need to do verification to fix it.

Design philosophy

1) T Flip-flop

I write T flip flop in structural to suit the main code (all system), its simple code with 2 inputs T and reset with clk and 1 output called Q and the output will be equation, and I make a testbench for it to let me use it in the main code and verification, here is the code with the testbench.

```
//T-FlipFlop Code
//Shahd Ali - 1200183

module t_flipflop (
  input wire clk,
  input wire T,
  input wire reset,
  output reg Q
);

  wire t_and, t_not, t_nand, t_xor;

  // AND gate for T and Q
  assign t_and = T & Q;

  // NOT gate for T
  assign t_not = ~T;

  // NAND gate for T and Q
  assign t_nand = t_and & clk;

  // XOR gate for T, Q, and CLK
  assign t_xor = t_nand ^ clk;

  // D flip-flop
  always @(posedge clk, negedge reset)
  begin
    if (!reset)
      Q <= 1'b0;
    else
      Q <= t_xor;
  end

endmodule
```

```
//Testbench Code
module t_flipflop_tb;

  reg clk;
  reg T;
  reg reset;
  wire Q;

  t_flipflop dut (
    .clk(clk),
    .T(T),
    .reset(reset),
    .Q(Q)
  );

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

  initial begin
    // Initialize inputs
    clk = 0;
    T = 0;
    reset = 0;

    // Apply reset
    reset = 1;
    #10 reset = 0;

    // Test case 1: Toggle Q
    T = 1;
    #20 T = 0;

    // Test case 2: Hold Q
    T = 0;
    #20 T = 0;

    // Test case 3: Toggle Q
    T = 1;
    #20 T = 0;

    // Test case 4: Reset
    reset = 1;
    #10 reset = 0;
    T = 0;
    #20 reset = 1;
    #10 reset = 0;

    // Finish simulation
    $finish;
  end
```

2) All system

Most my works were here in this section, so I divided for 4 parts depends on mode and up-down inputs.

At first I made truth table for the system by 6 inputs for the numbers and 2 inputs will be for the most important inputs mode and up-down and these all will been present states and the output will be 6 for the T flip flop.

Then by the output for the truth tabels from 4 parts I made k-map for every output and that mean for every t flip flop outs. That make for every t flip flop from the 6 outputs will have 4 equations from every part and these equations which I used in coding.

e.g. The inputs are (A,B,C,D,E,F) and (mode,up-down)

The outputs are (tA,tB,tC,tD,tE,tF) and these are the t flip flop outs.

I hope you got it doctor.

Now lets get example → we have this number (A,B,C,D,E,F) (000011) (3)and let mode =0 and up-down =0 so that mean its prim and up so the ouput will be (000101)(5) and that happen by the equations that we got it.

Now lets talk about the code…

Like we see I used the reset and clk on the flipflop when I caled it.

```
1  //All System Code
2  //Shahd Ali-1200183
3
4  module System (A, B, C, D, E, F, mode, up_down, en, rest, clk, TA, TB, TC, TD, TE, TF);
5
6    input wire A, B, C, D, E, F;
7    input wire mode, up_down, en, rest, clk;
8    output  TA, TB, TC, TD, TE, TF;
9
10   wire tA, tB, tC, tD, tE, tF;
11
12    t_flipflop tA_inst (.clk(clk), .T(A), .reset(~A), .Q(tA));
13    t_flipflop tB_inst (.clk(clk), .T(B), .reset(~B), .Q(tB));
14    t_flipflop tC_inst (.clk(clk), .T(C), .reset(~C), .Q(tC));
15    t_flipflop tD_inst (.clk(clk), .T(D), .reset(~D), .Q(tD));
16    t_flipflop tE_inst (.clk(clk), .T(E), .reset(~E), .Q(tE));
17    t_flipflop tF_inst (.clk(clk), .T(F), .reset(~F), .Q(tF));
18
```

Like we see here in this code I called the inputs and outpus of the system and called the t_flipflops and like we see its 6.

And I called the wire because I use it to put the answer from the t flip flop to the systems outs, like we see in the code below.

```
 9
10    wire tA, tB, tC, tD, tE, tF;
11
12     t_flipflop tA_inst (.clk(clk), .T(A), .reset(~A), .Q(tA));
13     t_flipflop tB_inst (.clk(clk), .T(B), .reset(~B), .Q(tB));
14     t_flipflop tC_inst (.clk(clk), .T(C), .reset(~C), .Q(tC));
15     t_flipflop tD_inst (.clk(clk), .T(D), .reset(~D), .Q(tD));
16     t_flipflop tE_inst (.clk(clk), .T(E), .reset(~E), .Q(tE));
17     t_flipflop tF_inst (.clk(clk), .T(F), .reset(~F), .Q(tF));
18
19
20    // Assign TB, TC, TD, TE, TF based on the mode and up_down inputs
21    assign TA = tA;
22    assign TB = tB;
23    assign TC = tC;
24    assign TD = tD;
25    assign TE = tE;
26    assign TF = tF;
```

After all of that I start did the main thing and its called the equations which I take it.

```
27
28    //CASE '1' mode == 0 && up_down == 0
29    // prime and up
30    assign TB = (mode == 0 && up_down == 0) ? (~A & C & D & E & F) : 1'b0;
31    assign TC = (mode == 0 && up_down == 0) ? ((~A & B & E & F) | (~A & ~B &
32    assign TD = (mode == 0 && up_down == 0) ? ((~A & C & E & F)|(~A & ~B & C
33    assign TE = (mode == 0 && up_down == 0) ? ((~A & ~B & ~C & ~E & F)|(~A &
34    assign TF = (mode == 0 && up_down == 0) ? ((~A & ~B & ~C & ~D & E & ~F)|
35
```

Here is the case 1 and its mean when mode=0 and up-down=0 its will start count prime numbers and up.

And here I use assign to put the equation because we have to work stuctural.

```
35
36    //CASE '2 'mode == 0 && up_down == 1
37    // prime and down
38    assign TB = (mode == 0 && up_down == 1) ? ((~A & ~B & ~C & ~D
39    assign TC = (mode == 0 && up_down == 1) ? ((A & B & ~C & E & F
40    assign TD = (mode == 0 && up_down == 1) ? ((~A & ~C & ~E & F)|
41    assign TE = (mode == 0 && up_down == 1) ? ((~A & D & ~E & F)|(
42    assign TF = (mode == 0 && up_down == 1) ? (~A & ~B & ~C & ~D)
43
```

Here is the case2 and its mean when mode=0 and up-down=1 its will start count prime numbers and down.

```
44    //CASE '3' mode == 1 && up_down == 0
45    // fib. and up
46    assign TA = (mode == 1 && up_down == 0) ? ((A & B & ~C & D & E & F)|(~A & B &
47    assign TB = (mode == 1 && up_down == 0) ? ((A & B & ~C & D & E & F)|(~A & B &
48    assign TC = (mode == 1 && up_down == 0) ? (~A & ~B & D & E & F) : 1'b0;
49    assign TD = (mode == 1 && up_down == 0) ? ((A & B & ~C & D & E & F)|(~A & B &
50    assign TE = (mode == 1 && up_down == 0) ? ((A & B & ~C & D & E & F)|(~A & B &
51    assign TF = (mode == 1 && up_down == 0) ? ((A & B & ~C & D & E & F)|(~A & B &
52
```

Here is the case3 and its mean when mode=1 and up-down=0 its will start count Fibonacci numbers and up.

```
52
53    //CASE '4' mode == 1 && up_down == 1
54    // fib. and down
55    assign TA = (mode == 1 && up_down == 1) ? ((~A & ~B & ~C & ~D & ~E & 
56    assign TB = (mode == 1 && up_down == 1) ? ((A & B & ~C & E & F)|(~A &
57    assign TC = (mode == 1 && up_down == 1) ? ((~A & B & D & F)|(~A & ~B &
58    assign TD = (mode == 1 && up_down == 1) ? ((~A & ~C & ~E & F)|(~A & ~E
59    assign TE = (mode == 1 && up_down == 1) ? ((~A & ~B & ~C & ~D & ~E & 
60    assign TF = (mode == 1 && up_down == 1) ? ((A & B & ~C & D & E & F)|(
61
62
```

And finally here in case 4 mean when mode =1 and up-down=1 will start count Fibonacci numbers and down.

Finally I made testbench for the all code its in the next page.

```
66
67  //Testbench Code
68  module System_TB;
69    reg A, B, C, D, E, F;
70    reg mode, up_down, en, rest, clk;
71    wire TA, TB, TC, TD, TE, TF;
72
73    // Instantiate the System module
74    System dut (
75      .A(A), .B(B), .C(C), .D(D), .E(E), .F(F),
76      .mode(mode), .up_down(up_down), .en(en), .rest(rest), .clk(clk),
77      .TA(TA), .TB(TB), .TC(TC), .TD(TD), .TE(TE), .TF(TF)
78    );
79
80    // Clock generation
81    always begin
82      #5 clk = ~clk;
83    end
84
85    // Stimulus
86    initial begin
87      // Initialize inputs
88      A = 0; B = 0; C = 0; D = 0; E = 0; F = 0;
89      mode = 0; up_down = 0; en = 0; rest = 0; clk = 0;
90
91      // Apply test vectors
92      #10 A = 1; B = 0; C = 1; D = 1; E = 1; F = 1;
93      #10 A = 0; B = 1; C = 0; D = 0; E = 0; F = 0;
94      #10 A = 1; B = 1; C = 1; D = 1; E = 0; F = 1;
95      #10 A = 0; B = 0; C = 0; D = 1; E = 0; F = 1;
96      #10 A = 1; B = 0; C = 1; D = 1; E = 0; F = 1;
97      #10 A = 0; B = 1; C = 1; D = 0; E = 1; F = 0;
98      #10 A = 1; B = 1; C = 0; D = 0; E = 1; F = 0;
99      #10 A = 0; B = 0; C = 0; D = 0; E = 0; F = 0;
100     #10 A = 1; B = 1; C = 1; D = 1; E = 1; F = 1;
101     #10 A = 1; B = 0; C = 0; D = 0; E = 0; F = 1;
102     #10 A = 0; B = 1; C = 0; D = 1; E = 1; F = 0;
103     #10 A = 1; B = 1; C = 0; D = 1; E = 0; F = 1;
104     #10 A = 0; B = 0; C = 1; D = 0; E = 1; F = 0;
105
106
107     // End simulation
108     #10 $finish;
109   end
110 endmodule
111
```

Here is the testbench and I made a lot of test vectors like we see and we can put how much we want.

Now like we saw I used all of 6 inputs (number bits), I used the clk and reset too when I called the flipflop and did it exacly like what we want, and I used mode and up-down on the equation to let the counter work .

Note: now there an input called enable and this input if (en == 0) then it will count the same number that I stop on it and if (en ==1) then we will continue the counting normally. I don't know how exactly I have to do it, so this is the code that I did it at the end.

Like we see in the code is behavioral but I didn't know other way ….

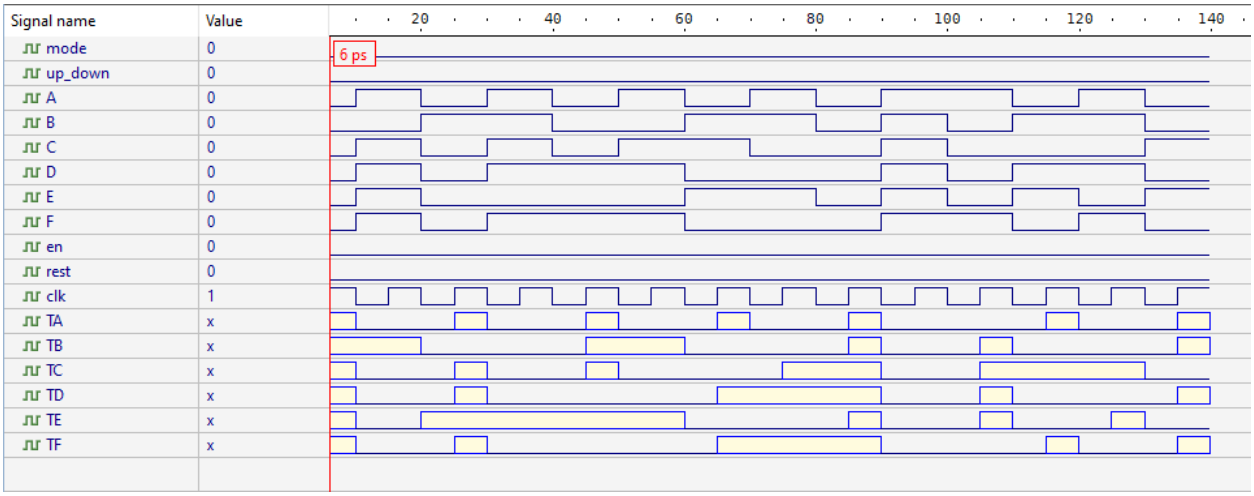
```
112      always @*
113    begin
114      if (en == 1'b0)
115          begin
116            assign TA = tA;
117            assign TB = tB;
118            assign TC = tC;
119            assign TD = tD;
120            assign TE = tE;
121            assign TF = tF;
122          end
123
124      else
125          begin
126            assign TA = A;
127            assign TB = B;
128            assign TC = C;
129            assign TD = D;
130            assign TE = E;
131            assign TF = F;
132          end
133    end
134
```

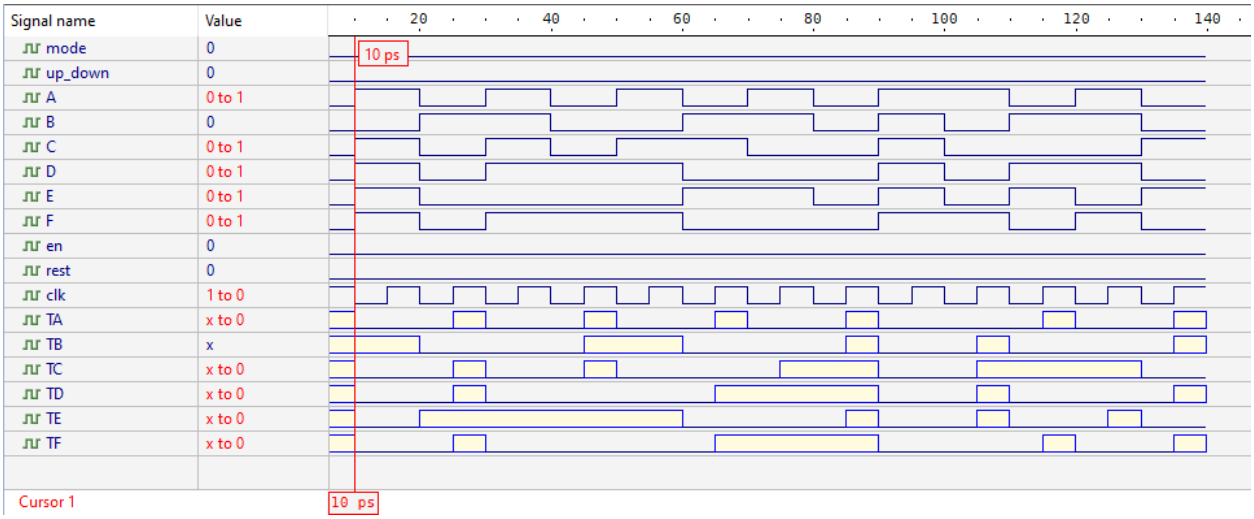
3) Simulation of the system

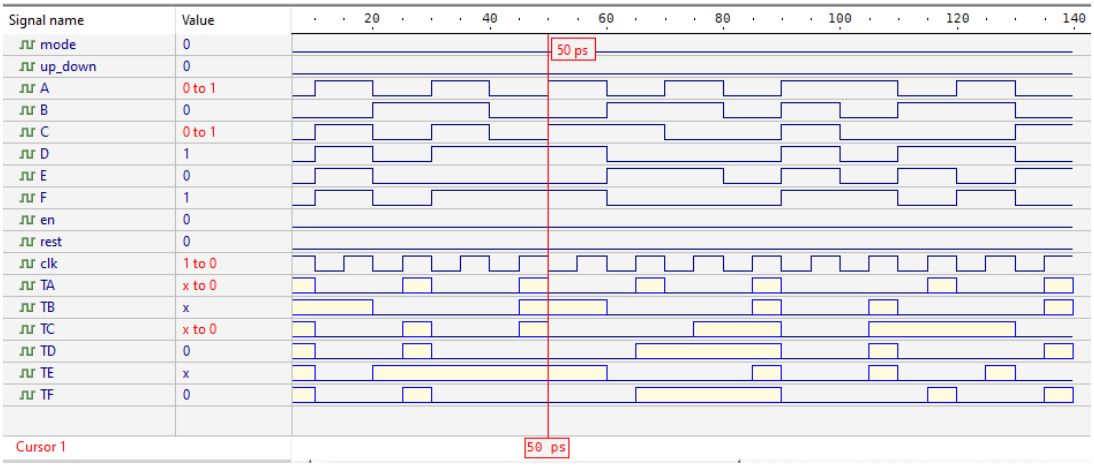
I made a simulation for the testbench of the system and it's made for the test vectors that I put it

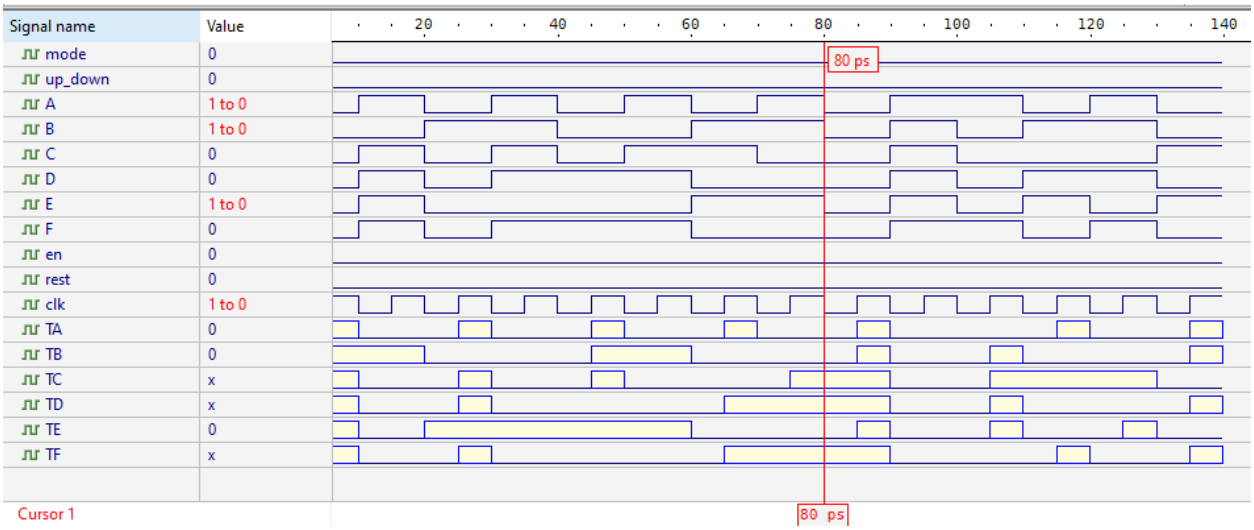
```
91      // Apply test vectors
92      #10 A = 1; B = 0; C = 1; D = 1; E = 1; F = 1;
93      #10 A = 0; B = 1; C = 0; D = 0; E = 0; F = 0;
94      #10 A = 1; B = 1; C = 1; D = 1; E = 0; F = 1;
95      #10 A = 0; B = 0; C = 0; D = 1; E = 0; F = 1;
96      #10 A = 1; B = 0; C = 1; D = 1; E = 0; F = 1;
97      #10 A = 0; B = 1; C = 1; D = 0; E = 1; F = 0;
98      #10 A = 1; B = 1; C = 0; D = 0; E = 1; F = 0;
99      #10 A = 0; B = 0; C = 0; D = 0; E = 0; F = 0;
100     #10 A = 1; B = 1; C = 1; D = 1; E = 1; F = 1;
101     #10 A = 1; B = 0; C = 0; D = 0; E = 0; F = 1;
102     #10 A = 0; B = 1; C = 0; D = 1; E = 1; F = 0;
103     #10 A = 1; B = 1; C = 0; D = 1; E = 0; F = 1;
104     #10 A = 0; B = 0; C = 1; D = 0; E = 1; F = 0;
105
106
107     // End simulation
108     #10 $finish;
```

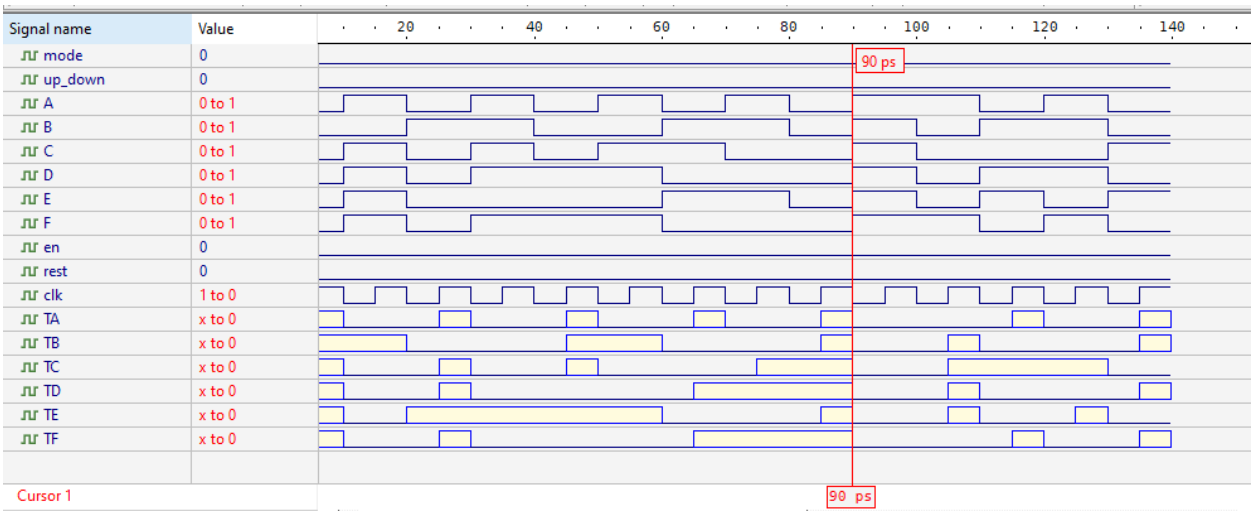
And here is the simulation that we got it

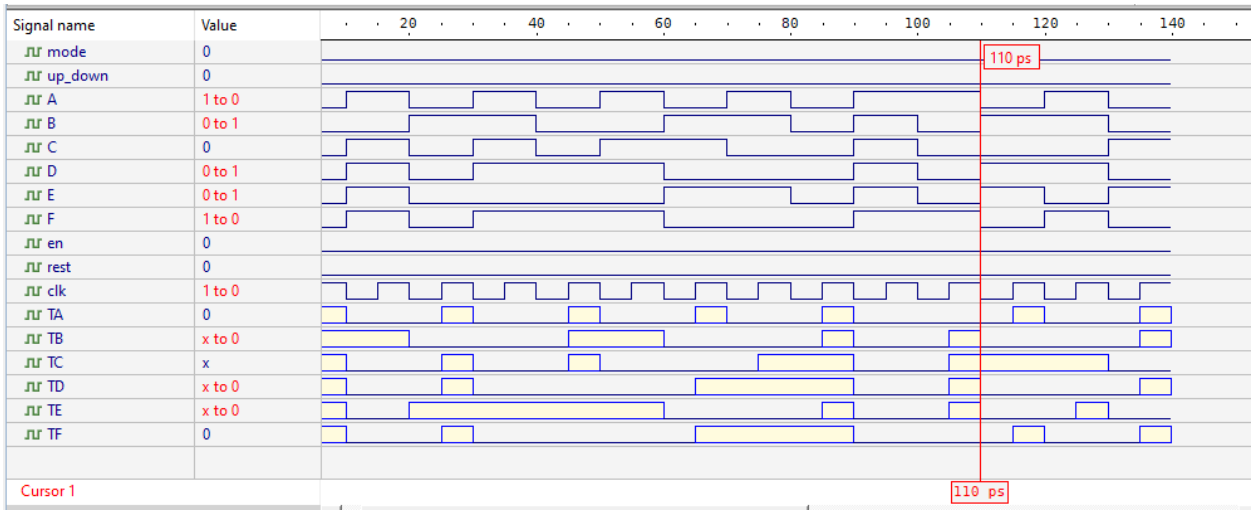


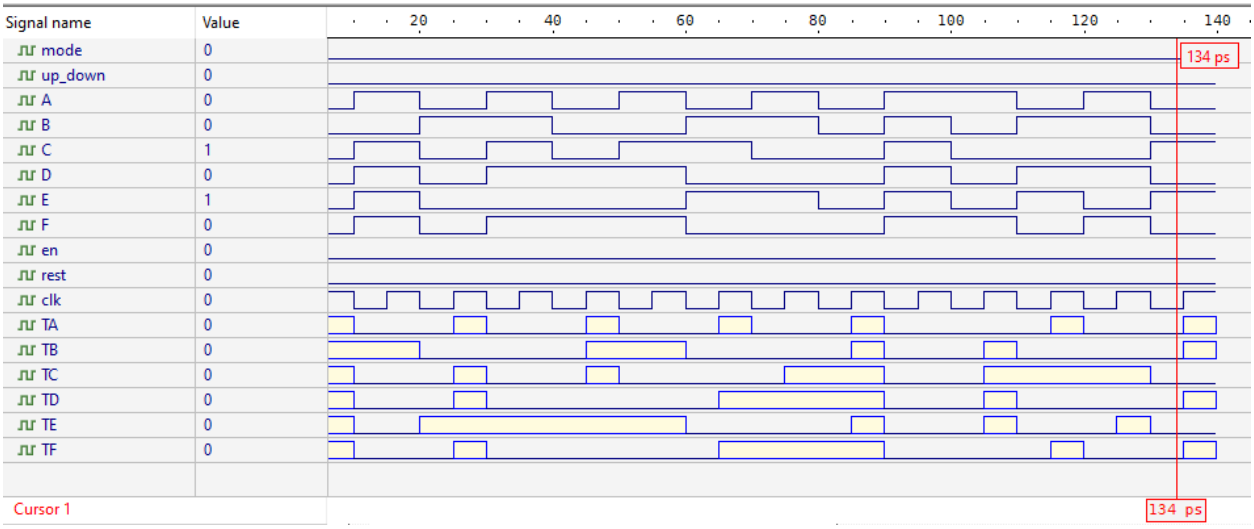

Signal name
Value
20
40
60
80
100
120
140
mode
0
10 ps
up_down
0
A
0 to 1
B
0
C
0 to 1
D
0 to 1
E
0 to 1
F
0 to 1
en
0
rest
0
clk
1 to 0
TA
x to 0
TB
x
TC
x to 0
TD
x to 0
TE
x to 0
TF
x to 0
Cursor 1
10 ps


Signal name
Value
20
40
60
80
100
120
140
mode
0
50 ps
up_down
0
A
0 to 1
B
0
C
0 to 1
D
1
E
0
F
1
en
0
rest
0
clk
1 to 0
TA
x to 0
TB
x
TC
x to 0
TD
0
TE
x
TF
0
Cursor 1
50 ps


Signal name
Value
20
40
60
80
100
120
140
mode
0
80 ps
up_down
0
A
1 to 0
B
1 to 0
C
0
D
0
E
1 to 0
F
0
en
0
rest
0
clk
1 to 0
TA
0
TB
0
TC
x
TD
x
TE
0
TF
x
Cursor 1
80 ps


Signal name
Value
20
40
60
80
100
120
140
mode
0
up_down
0
A
0 to 1
B
0 to 1
C
0 to 1
D
0 to 1
E
0 to 1
F
0 to 1
en
0
rest
0
clk
1 to 0
TA
x to 0
TB
x to 0
TC
x to 0
TD
x to 0
TE
x to 0
TF
x to 0
90 ps
Cursor 1
90 ps


Signal name
Value
20
40
60
80
100
120
140
mode
0
up_down
0
A
1 to 0
B
0 to 1
C
0
D
0 to 1
E
0 to 1
F
1 to 0
en
0
rest
0
clk
1 to 0
TA
0
TB
x to 0
TC
x
TD
x to 0
TE
x to 0
TF
0
110 ps
Cursor 1
110 ps


Signal name
Value
20
40
60
80
100
120
140
mode
0
up_down
0
A
0
B
0
C
1
D
0
E
1
F
0
en
0
rest
0
clk
0
TA
0
TB
0
TC
0
TD
0
TE
0
TF
0
134 ps
Cursor 1
134 ps

Results, Conclusion and Future works

Results

I tried my best to get the perfect results from the test bench and simulation you will that the result its right

Conclusion and Future works

In this project I build a structural circuit using T-Flip Flops and combination logic, which can count either Prime numbers or Fibonacci depends on a value of an input. Also, it can count them either up or down depends on the value of another input.

I try all of my best to do this project on simplest and easiest way, the project actually is not hard at all is just take time to do it.

I learned a lot from this project about T flip flop and how we can use it alone without anything but still helpful a lot .

After gaining a deeper understanding of Verilog, I successfully enhanced my coding skills and developed greater familiarity with utilizing its functionalities. I delved into the intricacies of writing codes, incorporating delays, and testing systems by leveraging the HDL tool to simulate my project and evaluate the outcomes.