



Faculty of Engineering and Technology
Electrical and Computer Engineering Department
Advanced Digital Design ENCS3310
Project Report

Sondos Shahin 1200166

Instructor: Dr. Abdallatif Abuissa

Section 2

Table of Contents

Table of Contents.....	2
Table of figures	3
Introduction	4
Design philosophy.....	5
4x1 mux with enable.....	5
T- flip flop.....	6
Circuit design	7
Counting prime numbers up	8
Counting prime numbers down	9
Counting Fibonacci sequence up	10
Counting Fibonacci sequence down	11
Code and results	12
Mux & flip flop	12
System	12
Test bench	15
Waveform	15
Testbench 2	16
Conclusion	17

Table of figures

Figure 1- 4x1 mux block.....	5
Figure 2- 4x1 mux state table	6
Figure 3- 4x1 mux code.....	6
Figure 4- t flip flop block	6
Figure 5 - t flip flop state table	7
Figure 6- t flip flop code.....	7
Figure 7- circuit design.....	7
Figure 8- prime up state table	8
Figure 9- prime up equations	8
Figure 10- prime down state table	9
Figure 11 - prime numbers equations	9
Figure 12- Fibonacci up state table.....	10
Figure 13- Fibonacci up equations.....	10
Figure 14-Fibonacci down state table.....	11
Figure 15- Fibonacci down equations.....	11
Figure 16- mux & tff code	12
Figure 17- system code.....	14
Figure 18- test bench code	15
Figure 19-waveform	15
Figure 20-testbench2.....	16

Introduction

The aim of this project is to design and build a special counter structurally using T flip flops and other combinational logic. This counter counts the first 11 number of both the prime numbers and Fibonacci sequence, based on a specific input. It also counts the both sequences in up and down depending on another input.

The first 11 prime numbers are:

2,3,5,7,11,13,17,19,23,29,31

The first 11 number in Fibonacci sequence are:

0,1,1,2,3,5,8,13,21,34,55

Design philosophy

This circuit is basically controlled by the clock, reset and enable inputs. This counter has 4 tasks to do, count prime numbers up, prime numbers down, Fibonacci sequence up and Fibonacci sequence down. As a result, a 4x1 mux is needed in this design as it has 2 selection lines to determine the wanted count method.

The largest number we need to reach in this counter is 55, which require 6 binary bits to represent it. So, 6 4x1 mux and 6 t flip flops are needed to complete the design.

Every mux is controlled by the enable (synchronous input), whereas the flip flops are controlled by the edge of both the clock and reset inputs(asynchronous).

For each case, a state table was written and the equations of the flip flops were found.

4x1 mux with enable

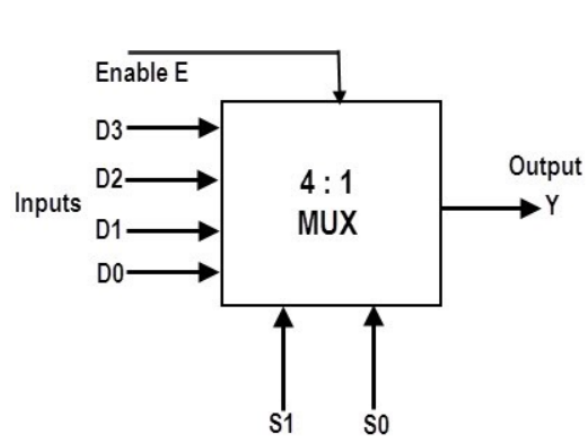


Figure 1- 4x1 mux block

Enable EN	Selection sequence	up/down	
0	x	x	0
1	0	0	Prime up
1	0	1	Prime down
1	1	0	Fibonacci up
1	1	1	Fibonacci down

Figure 2- 4x1 mux state table

```

module mux4x1 (in0, in1, in2, in3, en, s0, s1, out);
    input in0, in1, in2, in3, en, s0, s1;
    output reg out;
    always @ (in0 or in1 or in2 or in3 or en or s0 or s1 or out) begin
        if (en == 1) begin
            case ({s0,s1})
                2'b00: out = in0;
                2'b01: out = in1;
                2'b10: out = in2;
                2'b11: out = in3;
                default: out = 1'b0;
            endcase
        end
        else
            out = 1'b0;
        end
    end
endmodule

```

Figure 3- 4x1 mux code

T- flip flop

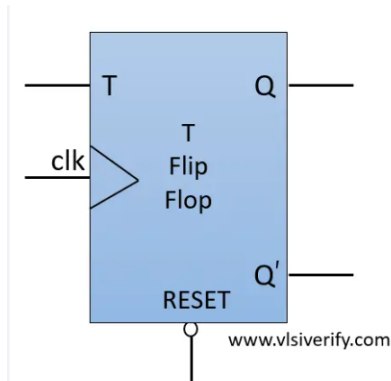


Figure 4- t flip flop block

T- Flip-Flop

present state	Next state	T
0	0	0
0	1	1
1	0	1
1	1	0

Figure 5 - t flip flop state table

```

module tff (Q, T, clk, rst);    //t flip flop
    output reg Q;
    input T, clk, rst;
    always @ (posedge clk or negedge rst)
        if (~rst)
            Q = 1'b0;
        else
            Q = Q ^ T;
endmodule

```

Figure 6- t flip flop code

Circuit design

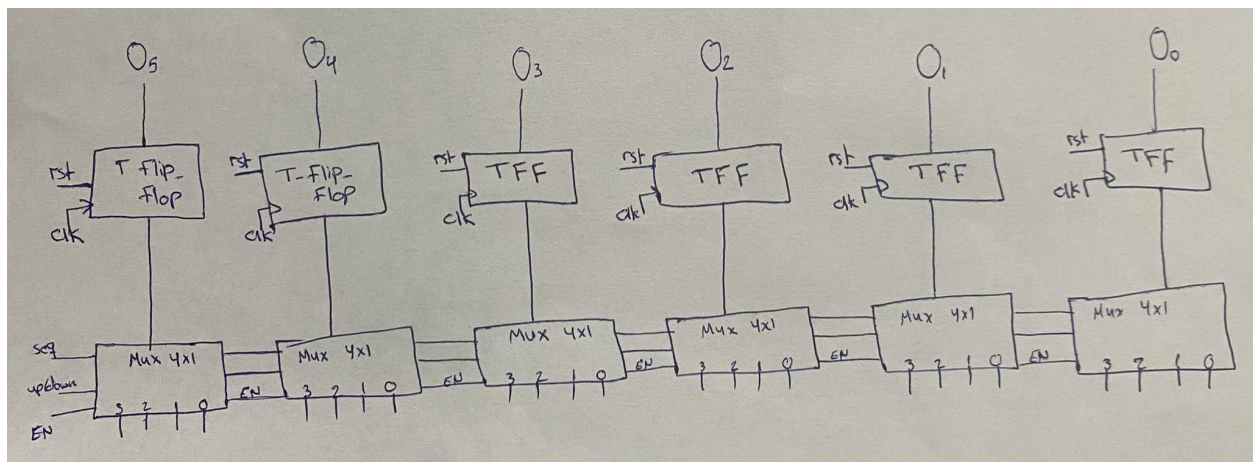


Figure 7- circuit design

Counting prime numbers up

Prime up		seq 0	up/down 0				
present state $Q_5 Q_4 Q_3 Q_2 Q_1 Q_0$	Next state $Q_5 Q_4 Q_3 Q_2 Q_1 Q_0$	T-Flip-Flops $T_5 T_4 T_3 T_2 T_1 T_0$					
0 0 0 0 1 0	0 0 0 0 1 1	0	0	0	0	0	1
0 0 0 0 1 1	0 0 0 1 0 1	0	0	0	1	1	0
0 0 0 1 0 1	0 0 0 1 1 1	0	0	0	0	1	0
0 0 0 1 1 1	0 0 1 0 1 1	0	0	1	1	0	0
0 0 1 0 1 1	0 0 1 1 0 1	0	0	0	1	1	0
0 0 1 1 0 1	0 1 0 0 0 1	0	1	1	1	0	0
0 1 0 0 0 1	0 1 0 0 1 1	0	0	0	0	1	0
0 1 0 0 1 1	0 1 0 1 1 1	0	0	0	1	0	0
0 1 0 1 1 1	0 1 1 1 0 1	0	0	1	0	1	0
0 1 1 1 0 1	0 1 1 1 1 1	0	0	0	0	1	0
0 1 1 1 1 1	0 0 0 0 1 0	0	1	1	1	0	1

Figure 8- prime up state table

$$T_5 = 0$$

$$T_4 = Q_5' Q_4' Q_3 Q_2 Q_1' Q_0 + Q_5' Q_4 Q_3 Q_2 Q_1 Q_0$$

$$T_3 = Q_5' Q_3' Q_2 Q_1 Q_0 + Q_5' Q_4' Q_3 Q_2 Q_1' Q_0 + Q_5' Q_4 Q_2 Q_1 Q_0$$

$$T_2 = Q_5' Q_4' Q_3' Q_1 Q_0 + Q_5' Q_4' Q_2' Q_1 Q_0 + Q_5' Q_4' Q_3 Q_2 Q_1' Q_0 + Q_5' Q_3' Q_2' Q_1 Q_0 + Q_5' Q_4 Q_3 Q_2 Q_1 Q_0$$

$$T_1 = Q_5' Q_4' Q_2' Q_1 Q_0 + Q_5' Q_4' Q_3' Q_2 Q_1' Q_0 + Q_5' Q_4 Q_3' Q_2 Q_1' Q_0 + Q_5' Q_4 Q_3' Q_2 Q_1 Q_0 + Q_5' Q_4 Q_3 Q_2 Q_1' Q_0$$

$$T_0 = Q_5' Q_4' Q_3' Q_2' Q_1 Q_0 + Q_5' Q_4 Q_3 Q_2 Q_1 Q_0$$

Figure 9- prime up equations

Counting prime numbers down

seq up/down
0 1

Prime down

Present state $Q_5 Q_4 Q_3 Q_2 Q_1 Q_0$	Next state $Q_5 Q_4 Q_3 Q_2 Q_1 Q_0$	T-Flip Flops					
		T_5	T_4	T_3	T_2	T_1	T_0
0 1 1 1 1 1	0 1 1 1 0 1	0	0	0	0	1	0
0 1 1 1 0 1	0 1 0 1 1 1	0	0	1	0	1	0
0 1 0 1 1 1	0 1 0 0 1 1	0	0	0	1	0	0
0 1 0 0 1 1	0 1 0 0 0 1	0	0	0	0	1	0
0 1 0 0 0 1	0 0 1 1 0 1	0	1	1	1	0	0
0 0 1 1 0 1	0 0 1 0 1 1	0	0	0	1	1	0
0 0 1 0 1 1	0 0 0 1 1 1	0	0	1	1	0	0
0 0 0 1 1 1	0 0 0 1 0 1	0	0	0	0	1	0
0 0 0 1 0 1	0 0 0 0 1 1	0	0	0	1	1	0
0 0 0 0 1 1	0 0 0 0 1 0	0	0	0	0	0	1
0 0 0 0 1 0	0 1 1 1 1 1	0	1	1	1	0	1

Figure 10- prime down state table

$$\begin{aligned}
 T_5 &= 0 \\
 T_4 &= Q_5' Q_4' Q_3' Q_2' Q_1 Q_0 + Q_5' Q_4 Q_3' Q_2' Q_1' Q_0 \\
 T_3 &= Q_5' Q_4' Q_3' Q_2' Q_1 Q_0 + Q_5' Q_4' Q_3' Q_2' Q_1' Q_0 + Q_5' Q_4 Q_3' Q_2' Q_1' Q_0 + Q_5' Q_4 Q_3 Q_2' Q_1' Q_0 \\
 T_2 &= Q_5' Q_4' Q_3' Q_2' Q_1 Q_0 + Q_5' Q_4' Q_3' Q_2' Q_1' Q_0 + Q_5' Q_4 Q_3' Q_2' Q_1' Q_0 + Q_5' Q_4 Q_3 Q_2' Q_1' Q_0 \\
 &\quad + Q_5' Q_4 Q_3 Q_2 Q_1' Q_0 \\
 T_1 &= Q_5' Q_4' Q_3' Q_2 Q_0 + Q_5' Q_4 Q_3' Q_2' Q_1 Q_0 + Q_5' Q_4 Q_3 Q_2 Q_0 + Q_5' Q_4' Q_2' Q_1' Q_0 \\
 T_0 &= Q_5' Q_4' Q_3' Q_2' Q_1
 \end{aligned}$$

Figure 11 - prime numbers equations

Counting Fibonacci sequence up

Fibonacci up						Seq 1	up/down 0										
Present state						Next state						T Flip Flops					
Q_5	Q_4	Q_3	Q_2	Q_1	Q_0	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0	T_5	T_4	T_3	T_2	T_1	T_0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	1
0	0	0	0	1	1	0	0	0	1	0	1	0	0	0	1	1	0
0	0	0	1	0	1	0	0	1	0	0	0	0	0	1	1	0	1
0	0	1	0	0	0	0	0	1	1	0	1	0	0	0	1	0	1
0	0	1	1	0	1	0	1	0	1	0	1	0	1	1	0	0	0
0	1	0	1	0	1	1	0	0	0	1	0	1	1	0	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	1
1	1	0	1	1	1	0	0	0	0	0	0	1	1	0	1	1	1

Figure 12- Fibonacci up state table

$$\begin{aligned}
 T_5 &= Q_5' Q_4 Q_3' Q_2 Q_1' Q_0 + Q_5 Q_4 Q_3' Q_2 Q_1 Q_0 \\
 T_4 &= Q_5' Q_4' Q_3 Q_2 Q_1' Q_0 + Q_5' Q_4 Q_3' Q_2 Q_1' Q_0 + Q_5' Q_4' Q_3' Q_2' Q_1 Q_0' + Q_5 Q_4 Q_3' Q_2 Q_1 Q_0 \\
 T_3 &= Q_5' Q_4' Q_2 Q_1' Q_0 \\
 T_2 &= Q_5' Q_4' Q_3' Q_2' Q_1 Q_0 + Q_5' Q_3' Q_2 Q_1' Q_0 + Q_5' Q_4' Q_3 Q_2' Q_1' Q_0' + Q_5 Q_4' Q_3' Q_2' Q_1 Q_0' \\
 &\quad + Q_5 Q_4 Q_3' Q_2 Q_1 Q_0 \\
 T_1 &= Q_5' Q_4' Q_3' Q_2' Q_0 + Q_5' Q_4 Q_3' Q_2 Q_1' Q_0 + Q_5 Q_4 Q_3' Q_2 Q_1 Q_0 \\
 T_0 &= Q_5' Q_4' Q_2' Q_1' Q_0' + Q_5' Q_3' Q_2 Q_1' Q_0 + Q_4' Q_3' Q_2' Q_1 Q_0' + Q_5 Q_4 Q_3' Q_2 Q_1 Q_0 \\
 &\quad + Q_5' Q_4' Q_3' Q_2' Q_1'
 \end{aligned}$$

Figure 13- Fibonacci up equations

Counting Fibonacci sequence down

Fibonacci down						seq	up/down										
Present state						Next state						T-Flip-Flops					
Q_5	Q_4	Q_3	Q_2	Q_1	Q_0	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0	T_5	T_4	T_3	T_2	T_1	T_0
1	1	0	1	1	1	1	0	0	0	1	0	0	1	0	1	0	1
1	0	0	0	1	0	0	1	0	1	0	1	1	1	0	1	1	1
0	1	0	1	0	1	0	0	1	1	0	1	0	1	1	0	0	0
0	0	1	1	0	1	0	0	1	0	0	0	0	0	0	1	0	1
0	0	1	0	0	0	0	0	0	1	0	1	0	0	1	1	0	1
0	0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1	0
0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	1	1	1	1	1	0	1	1	1

Figure 14-Fibonacci down state table

$$\begin{aligned}
 T_5 &= Q_5' Q_4' Q_3' Q_2' Q_1' Q_0' + Q_5 Q_4' Q_3' Q_2' Q_1' Q_0' \\
 T_4 &= Q_5' Q_4' Q_3' Q_2' Q_1' Q_0' + Q_5' Q_4' Q_3' Q_2' Q_1' Q_0 + Q_5 Q_4' Q_3' Q_2' Q_1' Q_0' + Q_5 Q_4' Q_3' Q_2' Q_1' Q_0 \\
 T_3 &= Q_5' Q_4' Q_3' Q_2' Q_1' Q_0' + Q_5' Q_4' Q_3' Q_2' Q_1' Q_0 \\
 T_2 &= Q_5' Q_4' Q_3' Q_2' Q_1' Q_0' + Q_5' Q_4' Q_3' Q_2' Q_1' Q_0 + Q_5' Q_4' Q_3' Q_2' Q_1' Q_0' + Q_5' Q_4' Q_3' Q_2' Q_1' Q_0 \\
 T_1 &= Q_5' Q_4' Q_3' Q_2' Q_1' Q_0' + Q_5' Q_4' Q_3' Q_2' Q_1' Q_0 + Q_4' Q_3' Q_2' Q_1' Q_0' \\
 T_0 &= Q_5' Q_4' Q_3' Q_2' + Q_5' Q_4' Q_3' Q_2' Q_1' Q_0' + Q_5' Q_4' Q_3' Q_2' Q_1' Q_0 + Q_4' Q_3' Q_2' Q_1' Q_0' \\
 &\quad + Q_5' Q_4' Q_3' Q_2' Q_1' Q_0
 \end{aligned}$$

Figure 15- Fibonacci down equations

Code and results

Mux & flip flop

```

1  module tff (Q, T, clk, rst);    //t flip flop
2      output reg Q;
3      input T, clk, rst;
4      always @ (posedge clk or negedge rst)
5          if (~rst)
6              Q = 1'b0;
7          else
8              Q = Q ^ T;
9  endmodule
10
11
12  module mux4x1 (in0, in1, in2, in3, en, s0, s1, out);
13      input in0, in1, in2, in3, en, s0, s1;
14      output reg out;
15      always @ (in0 or in1 or in2 or in3 or en or s0 or s1 or out) begin
16          if (en == 1) begin
17              case ({s0,s1})
18                  2'b00: out = in0;
19                  2'b01: out = in1;
20                  2'b10: out = in2;
21                  2'b11: out = in3;
22                  default: out = 1'b0;
23              endcase
24          end
25          else
26              out = 1'b0;
27          end
28      endmodule
29

```

Figure 16- mux & tff code

System

```

31  module sys (q5,q4,q3,q2,q1,q0,clk,rst,en,seq,ud,o5,o4,o3,o2,o1,o0);
32      input q5,q4,q3,q2,q1,q0,clk,rst,en,seq,ud;
33      output o5,o4,o3,o2,o1,o0;
34      wire t5,t4,t3,t2,t1,t0;
35
36      //pu -> prime up
37      //pd -> prime down
38      //fu -> fibonacci up
39      //fd -> fibonacci down
40
41      //mux5
42      //bit 5 - prime up & down = 0
43      wire outfu51, outfu52, outfu5, outfd5, outfd51, outfd52;
44      and fu51 (outfu51, ~q5, q4, ~q3, q2, ~q1, q0);
45      and fu52 (outfu52, q5, q4, ~q3, q2, q1, q0);
46      or fu5 (outfu5, outfu51, outfu52);    //bit 5 - fibonacci up
47
48      and fd51 (outfd1, ~q5, ~q4, ~q3, ~q2, ~q1, ~q0);
49      and fd52 (outfd2, q5, ~q4, ~q3, ~q2, q1, ~q0);
50      or fd5 (outfd5, outfd51, outfd52);    //bit 5 - fibonacci down
51
52      mux4x1 mux5 (1'b0, 1'b0, outfu5, outfd5, en, seq, ud, t5);
53

```

```

55 //mux4
56 wire outpu4,outpu41,outpu42,outpd4, outpd41, outpd42 ,outfu41, outfu42, outfu43, outfu44,outfu4 , outfd4, outfd41, outfd42, outfd43, outfd44;
57 and pu41 (outpu41,~q5, ~q4, q3, q2, ~q1, q0);
58 and pu42 (outpu42, ~q5, q4, q3, q2, q1, q0);
59 or pu4 (outpu4, outpu41, outpu42); //bit 4 - prime up
60
61 and pd41 (outpd41,~q5, ~q4, ~q3, ~q2, q1, ~q0);
62 and pd42 (outpd42, ~q5, q4, ~q3, ~q2, ~q1, q0);
63 or pd4 (outpd4, outpd41, outpd42); //bit 4 - prime down
64
65 and fu41 (outfu41,~q5, ~q4, q3, q2, ~q1, q0);
66 and fu42 (outfu42, ~q5, q4, ~q3, q2, ~q1, q0);
67 and fu43 (outfu43,q5, ~q4, ~q3, ~q2, q1, ~q0);
68 and fu44 (outfu44, q5, q4, ~q3, q2, q1, q0);
69 or fu4 (outfu4, outfu41, outfu42, outfu43, outfu44); //bit 4 - fibonacci up
70
71 and fd41 (outfd41,~q5, ~q4, ~q3, ~q2, ~q1, ~q0);
72 and fd42 (outfd42, ~q5, q4, ~q3, q2, ~q1, q0);
73 and fd43 (outfd43,q5, ~q4, ~q3, ~q2, q1, ~q0);
74 and fd44 (outfd44, q5, q4, ~q3, q2, q1, q0);
75 or fd4 (outfd4, outfd41, outfd42, outfd43, outfd44); //bit 4 - fibonacci down
76
77 mux4x1 mux4 (outpu4, outpd4, outfu4, outfd4, en, seq, ud, t4);
78
81 //mux3
82 wire outpu3,outpu31,outpu32, outpu33,outpd3, outpd31, outpd32, outpd33, outpd34 ,outfu3 , outfd3, outfd31, outfd32;
83 and pu31 (outpu31,~q5,~q3,q2,q1,q0);
84 and pu32 (outpu32,~q5,~q4,q3,q2,~q1,q0);
85 and pu33 (outpu33,~q5,q4,q2,q1,q0);
86 or pu3 (outpu3, outpu31, outpu32,outpu33); //bit 3 - prime up
87
88 and pd31 (outpd31,~q5, ~q4, ~q3, ~q2, q1, ~q0);
89 and pd32 (outpd32, ~q5, ~q4, q3, ~q2, q1, q0);
90 and pd33 (outpd33,~q5, q4, ~q3, ~q2, ~q1, q0);
91 and pd34 (outpd34,~q5, q4, q3, q2, ~q1, q0);
92 or pd3 (outpd3, outpd31, outpd32, outpd33, outpd34); //bit 3 - prime down
93
94 and fu3 (outfu3,~q5, ~q4, q2, ~q1, q0); //bit 3 - fibonacci up
95
96 and fd31 (outfd31,~q5, ~q4, q3, ~q2, ~q1, ~q0);
97 and fd32 (outfd32, ~q5, q4, ~q3, q2, ~q1, q0);
98 or fd3 (outfd3, outfd31, outfd32); //bit 3 - fibonacci down
99
100 mux4x1 mux3 (outpu3, outpd3, outfu3, outfd3, en, seq, ud, t3);
101
104 //mux2
105 wire outpu2, outpu21, outpu22, outpu23, outpu24,outpu25, outpd2, outpd21, outpd22,outpd23, outpd24, outpd25,
106 outfu2, outfu21, outfu22, outfu23, outfu24, outfu25,outfd2, outfd21, outfd22, outfd23, outfd24;
107
108 and pu21 (outpu21,~q5, ~q4, ~q3, q1, q0);
109 and pu22 (outpu22, ~q5, ~q4, ~q2, q1, q0);
110 and pu23 (outpu23,~q5, ~q4, q3, q2, ~q1, q0);
111 and pu24 (outpu24, ~q5, ~q3, ~q2, q1, q0);
112 and pu25 (outpu25, ~q5, q4, q3, q2, q1, q0);
113 or pu2 (outpu2, outpu21, outpu22, outpu23, outpu24,outpu25); //bit 2 - prime up
114
115 and pd21 (outpd21,~q5, ~q4, ~q3, ~q2, q1, ~q0);
116 and pd22 (outpd22, ~q5, ~q4, q2, ~q1, q0);
117 and pd23 (outpd23,~q5, ~q4, q3, ~q2, q1, ~q0);
118 and pd24 (outpd24,~q5, q4, ~q3, ~q2, ~q1, q0);
119 and pd25 (outpd25,~q5, q4, ~q3, q2, q1, q0);
120 or pd2 (outpd2, outpd21, outpd22, outpd23, outpd24,outpd25); //bit 2 - prime down
121
122 and fu21 (outfu21,~q5, ~q4, ~q3, ~q2, q1, q0);
123 and fu22 (outfu22, ~q5, ~q3, q2, ~q1, q0);
124 and fu23 (outfu23, ~q5, ~q4, q3, ~q2, ~q1, ~q0);
125 and fu24 (outfu24, q5, ~q4, ~q3, ~q2, q1, ~q0);
126 and fu25 (outfu25, q5, q4, ~q3, q2, q1, q0);
127 or fu2 (outfu2, outfu21, outfu22, outfu23, outfu24, outfu25); //bit 2 - fibonacci up
128
129 and fd21 (outfd21,~q5, ~q4, ~q2, ~q1, ~q0);
130 and fd22 (outfd22, ~q5, ~q4, q2, ~q1, q0);
131 and fd23 (outfd23,q5, ~q4, ~q3, ~q2, q1, ~q0);
132 and fd24 (outfd24, q5, q4, ~q3, q2, q1, q0);
133 or fd2 (outfd2, outfd21, outfd22, outfd23, outfd24); //bit 2 - fibonacci down
134
135 mux4x1 mux2 (outpu2, outpd2, outfu2, outfd2, en, seq, ud, t2);

```



```

138 //mux1
139 wire output1, output11, output12, output13, output14,output15, outpd1, outpd11, outpd12, outpd13, outpd14,
140 outfu1, outfu11, outfu12, outfu13, outfd1, outfd11, outfd12, outfd13;
141
142
143 and pu11 (output11,~q5, ~q4, ~q2, q1, q0);
144 and pu12 (output12, ~q5, ~q4, ~q3, q2, ~q1, q0);
145 and pu13 (output13,~q5, q4, ~q3, ~q2, ~q1, q0);
146 and pu14 (output14, ~q5, q4, ~q3, q2, q1, q0);
147 and pu15 (output15, ~q5, q4, q3, q2, ~q1, q0);
148 or pul (output1, output11, output12, output13, output14,output15); //bit 1 - prime up
149
150 and pd11 (outpd11,~q5, ~q4, ~q3, q2, q0);
151 and pd12 (outpd12, ~q5, q4, ~q3, ~q2, q1, q0);
152 and pd13 (outpd13,~q5, q4, q3, q2, q0);
153 and pd14 (outpd14, ~q5, ~q4, q2, ~q1, q0);
154 or pdl (outpd1, outpd11, outpd12, outpd13, outpd14); //bit 1 - prime down
155
156 and fu11 (outfu11,~q5, ~q4, ~q3, ~q2, q0);
157 and fu12 (outfu12, ~q5, q4, ~q3, q2, ~q1, q0);
158 and fu13 (outfu13, q5, q4, ~q3, q2, q1, q0);
159 or ful (output1, outfu11, outfu12, outfu13); //bit 2 - fibonacci up
160
161 and fd11 (outfd11,~q5, ~q4, ~q3, ~q2, ~q0);
162 and fd12 (outfd12, ~q5, ~q4, ~q3, q2, ~q1, q0);
163 and fd13 (outfd13, ~q4, ~q3, ~q2, q1, ~q0);
164 or fd1 (outfd1, outfd11, outfd12, outfd13); //bit 1 - fibonacci down
165
166 mux4x1 mux1 (output1, outpd1, outfu1, outfd1, en, seq, ud, t1);
167
168
169 //mux0
170 wire output0, output01, output02, outpd0, outfu0, outfu01, outfu02, outfu03, outfu04, outfu05,
171 outfd0, outfd01, outfd02, outfd03, outfd04, outfd05;
172
173
174
175 and pu01 (output01,~q5, ~q4, ~q3, ~q2, q1, ~q0);
176 and pu02 (output02, ~q5, q4, q3, q2, q1, q0);
177 or pu0 (output0, output01, output02); //bit 0 - prime up
178
179 and pd0 (outpd0,~q5, ~q4, ~q3, ~q2, q1); //bit 0 - prime down
180
181
182
183 and fu01 (outfu01,~q5, ~q4, ~q2, ~q1, ~q0);
184 and fu02 (outfu02, ~q5, ~q3, q2, ~q1, q0);
185 and fu03 (outfu03, ~q4, ~q3, ~q2, q1, ~q0);
186 and fu04 (outfu04, q5, q4, ~q3, q2, q1, q0);
187 and fu05 (outfu05, ~q5, ~q4, ~q3, ~q2, ~q1);
188 or fu0 (outfu0, outfu01, outfu02, outfu03, outfu04, outfu05); //bit 0 - fibonacci up
189
190 and fd01 (outfd01,~q5, ~q4, ~q3, ~q2);
191 and fd02 (outfd02, ~q5, ~q4, ~q2, ~q1, ~q0);
192 and fd03 (outfd03, ~q5, ~q4, q3, q2, ~q1, q0);
193 and fd04 (outfd04, ~q4, ~q3, ~q2, q1, ~q0);
194 and fd05 (outfd05, q5, q4, ~q3, q2, q1, q0);
195 or fd0 (outfd0, outfd01, outfd02, outfd03, outfd04, outfd05); //bit 0 - fibonacci down
196
197 mux4x1 mux0 (output0, outpd0, outfu0, outfd0, en, seq, ud, t0);
198
199
200 //connect the output of every mux with the input of a t flip flop
201 tff tf5 (o5,t5, clk, rst);
202 tff tf4 (o4,t4, clk, rst);
203 tff tf3 (o3,t3, clk, rst);
204 tff tf2 (o2,t2, clk, rst);
205 tff tf1 (o1,t1, clk, rst);
206 tff tf0 (o0,t0, clk, rst);
207
208 endmodule
209

```

Figure 17- system code

Test bench

```

214 //test bench
215 module test () ;
216 reg q5,q4,q3,q2,q1,q0,clk,rst,en,seq,ud;
217 wire o5,o4,o3,o2,o1,o0;
218 sys tsys (q5,q4,q3,q2,q1,q0,clk,rst,en,seq,ud,o5,o4,o3,o2,o1,o0);
219 initial begin
220     $monitor("time %0d  %b %b %b %b %b %b", $time, o5,o4,o3,o2,o1,o0);
221     rst = 0; clk = 0;
222     #10 rst = 1;
223     repeat (100)
224         #20 clk = ~clk;
225 end
226 initial begin
227     en = 1'b0;
228     #20 en = 1'b1;
229     //test prime up
230     seq= 1'b0; ud = 1'b0;
231     {q5,q4,q3,q2,q1,q0} = 6'b000000;
232     repeat (31)
233         #10 {q5,q4,q3,q2,q1,q0} = {q5,q4,q3,q2,q1,q0} + 6'b000001;
234     //test prime down
235     seq= 1'b0; ud = 1'b1;
236     {q5,q4,q3,q2,q1,q0} = 6'b000000;
237     repeat (31)
238         #10 {q5,q4,q3,q2,q1,q0} = {q5,q4,q3,q2,q1,q0} + 6'b000001;
239     //test fibonacci up
240     seq= 1'b1; ud = 1'b0;
241     {q5,q4,q3,q2,q1,q0} = 6'b000000;
242     repeat (63)
243         #10 {q5,q4,q3,q2,q1,q0} = {q5,q4,q3,q2,q1,q0} + 6'b000001;
244     //test fibonacci down
245     seq= 1'b1; ud = 1'b1;
246     {q5,q4,q3,q2,q1,q0} = 6'b000000;
247     repeat (63)
248         #10 {q5,q4,q3,q2,q1,q0} = {q5,q4,q3,q2,q1,q0} + 6'b000001;
249 end
250 endmodule

```

Figure 18- test bench code

Waveform

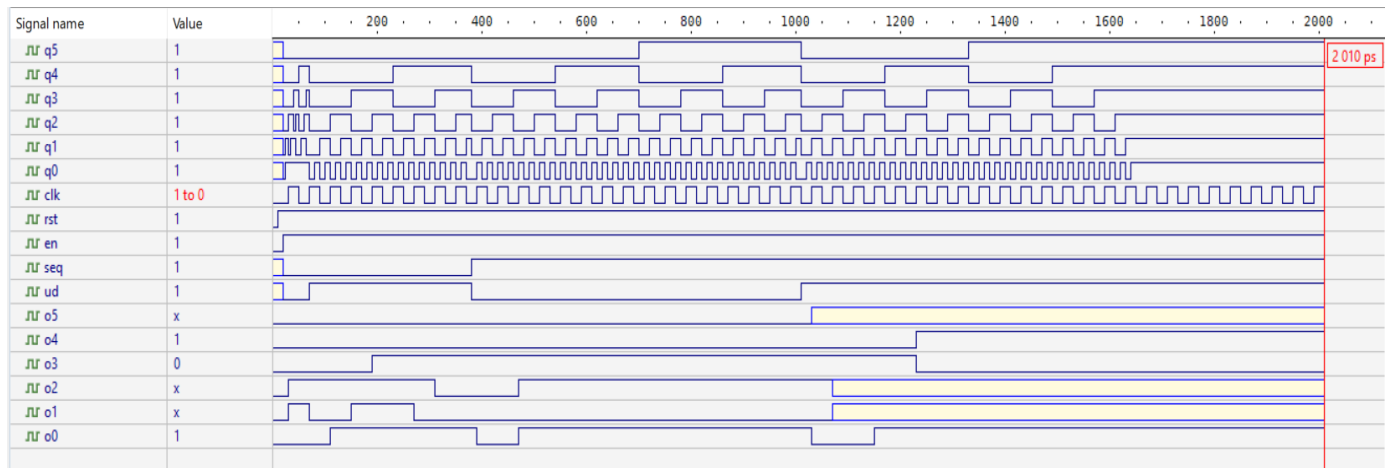


Figure 19-waveform

Testbench 2

```

214 //test bench
215 module test ();
216 reg q5,q4,q3,q2,q1,q0,clk,rst,en,seq,ud;
217 wire o5,o4,o3,o2,o1,o0;
218 sys tsys (q5,q4,q3,q2,q1,q0,clk,rst,en,seq,ud,o5,o4,o3,o2,o1,o0);
219 initial begin
220     rst = 0; clk = 0;
221     #10 rst = 1;
222     repeat (100)
223         #20 clk = ~clk;
224 end
225 initial begin
226     en = 1'b0;
227     #20 en = 1'b1;
228
229     //test prime up
230     seq= 1'b0; ud = 1'b0;
231     {q5,q4,q3,q2,q1,q0} = 6'b000000; //test generator
232     if ({o5,o4,o3,o2,o1,o0} != 6'b0000011) $display ("count failed"); //result analyzer
233     #5 {q5,q4,q3,q2,q1,q0} = 6'b000011; //test generator
234     if ({o5,o4,o3,o2,o1,o0} != 6'b000101) $display ("count failed"); //result analyzer
235     #5 {q5,q4,q3,q2,q1,q0} = 6'b000101; //test generator
236     if ({o5,o4,o3,o2,o1,o0} != 6'b000111) $display ("count failed"); //result analyzer
237     #5 {q5,q4,q3,q2,q1,q0} = 6'b000111; //test generator
238     if ({o5,o4,o3,o2,o1,o0} != 6'b001011) $display ("count failed"); //result analyzer
239     #5 {q5,q4,q3,q2,q1,q0} = 6'b001011; //test generator
240     if ({o5,o4,o3,o2,o1,o0} != 6'b001101) $display ("count failed"); //result analyzer
241     #5 {q5,q4,q3,q2,q1,q0} = 6'b001101; //test generator
242     if ({o5,o4,o3,o2,o1,o0} != 6'b010001) $display ("count failed"); //result analyzer
243     #5 {q5,q4,q3,q2,q1,q0} = 6'b010001; //test generator
244     if ({o5,o4,o3,o2,o1,o0} != 6'b010011) $display ("count failed"); //result analyzer
245     #5 {q5,q4,q3,q2,q1,q0} = 6'b010011; //test generator
246     if ({o5,o4,o3,o2,o1,o0} != 6'b010111) $display ("count failed"); //result analyzer
247     #5 {q5,q4,q3,q2,q1,q0} = 6'b010111; //test generator
248     if ({o5,o4,o3,o2,o1,o0} != 6'b011101) $display ("count failed"); //result analyzer
249     #5 {q5,q4,q3,q2,q1,q0} = 6'b011101; //test generator
250     if ({o5,o4,o3,o2,o1,o0} != 6'b011101) $display ("count failed"); //result analyzer
251     #5 {q5,q4,q3,q2,q1,q0} = 6'b011111; //test generator
252     if ({o5,o4,o3,o2,o1,o0} != 6'b011111) $display ("count failed"); //result analyzer
253 end
254 endmodule
255

```

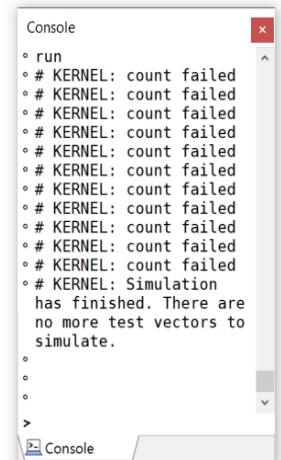


Figure 20-testbench2

Conclusion

As shown in the figures above, it is clear that there is an error in the functionality of the circuit, since the expected output is not the same as the actual output of the circuit.

The problem might be in the equations that I wrote from the state tables, or it might be in the design itself.