

**DIGITAL DESIGN**

**ASSIGNMENTREPORT**

**ASSIGNMENT ID : 04**

**Student Name: Weibao Fu**

**Student ID: 11812202**

PART 1: DIGITAL design THEORY

1.

**JK filp-flop**

|  |  |  |  |
| --- | --- | --- | --- |
| J | K | Q | Qt+1 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Characteristic equation: Qt+1 = JQ’ + K’Q

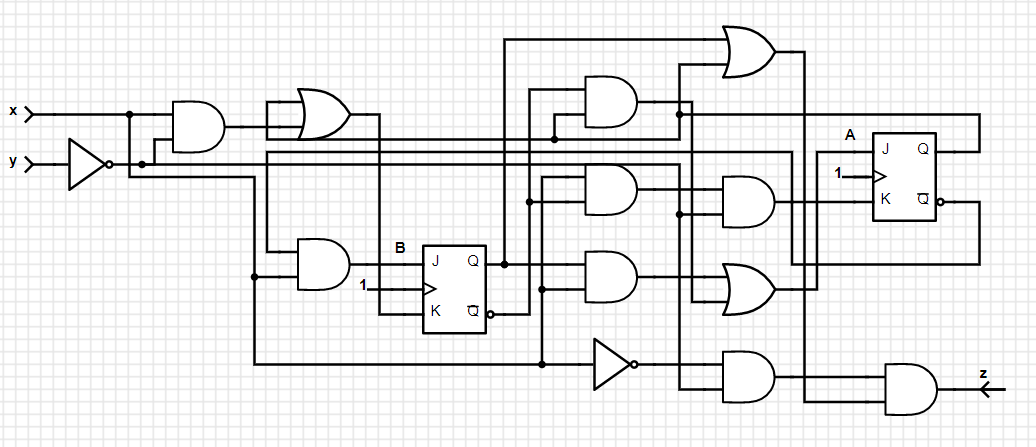
**T flip-flop**

|  |  |  |
| --- | --- | --- |
| T | Q | Qt+1 |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Characteristic equation: Qt+1 = TQ’ + T’Q

2.

**Circuit diagram**



**State table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | y | A | B | JA | KA | JB | KB | z |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |

**State equation**

From the state table, we can get At+1 = JAAt’ + KA’At

= (Btx + Bt’y’)At’ + (Bt’xy’)’At

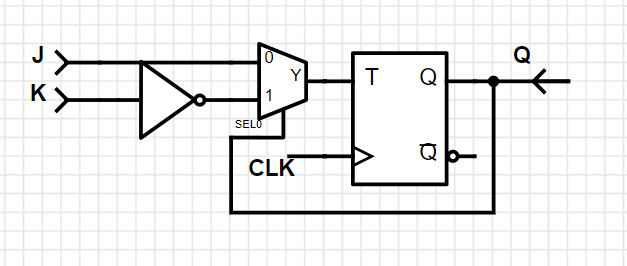
= At’Bt’y’ + Btx + Atx’ + Aty

Bt+1 = JBBt’ + KB’Bt

= (At’x)Bt’ + (At + xy’)’Bt

= At’Bt’x + At’Bt(x’+y)

3.



**D**

Since the state equation of JK flip-flop is Q = JQ’ + K’Q

Then we can use multiplexer, we can set Q0 = J, Q1 = K to implement

Y = JQ0’ + K’Q1 , and we use a D flip-flop to finish the JK flip-flop.

4.

**State table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| At | Bt | TA | TB | At+1 | Bt+1 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 |

**State equation**

TA = A + B

TB = A’ + B

At+1 = TAA’ + TA’A

= (A + B)A’ + (A+B)’A

= A’B

Bt+1 = TBB’ + TB’B

= (A’ + B)B’ + (A’ + B)’B

= A’B’

**State transfer graph**

11

From the state transfer graph and state table, we can get that the circuit is a ternary counter. It will loop in every 3 periods.

PART 2: DIGITAL design LAB (Task1)

##### Design

***Verilog Code***

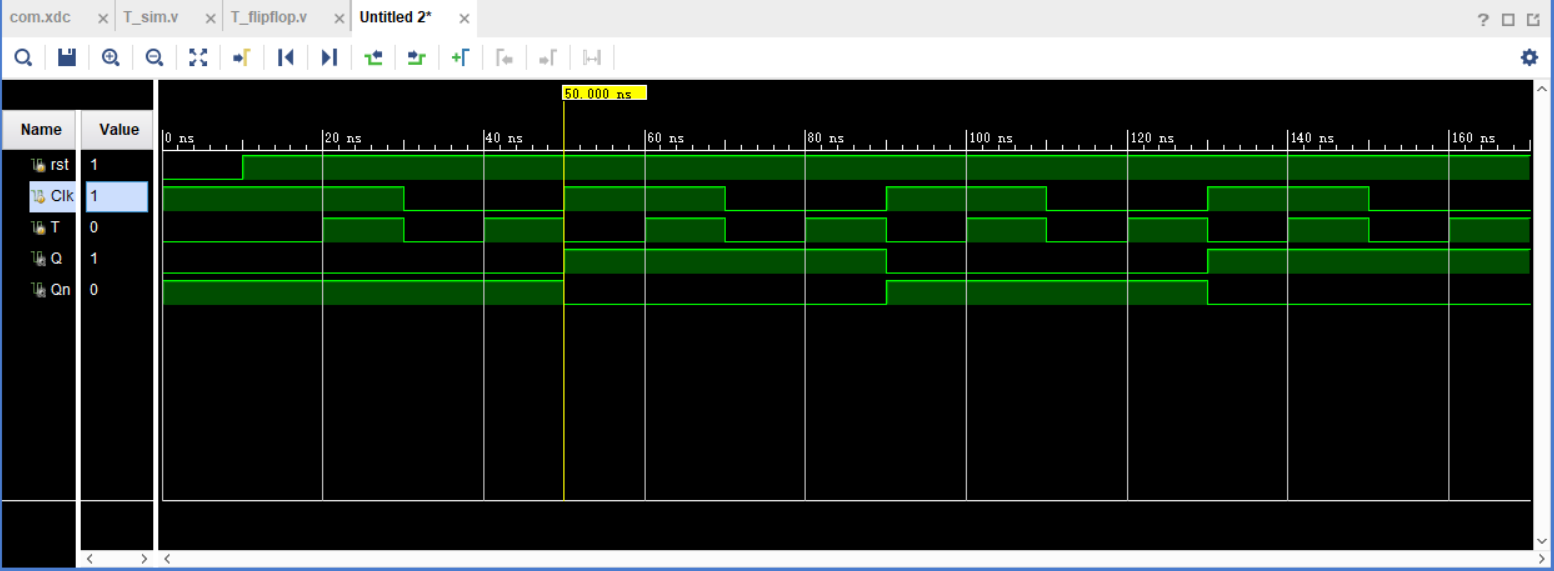
`timescale 1ns / 1ps  
module T\_flipflop(T,Clk,rst,Q,Qn);  
input  T,Clk,rst;  
output Q,Qn;  
wire q1,qn,temp1,temp2,temp3,temp4;  
nand G7(temp1,~Clk,T,Qn,rst);  
nand G8(temp2,~Clk,T,Q);  
nand G5(q1,temp1,qn);  
nand G6(qn,temp2,q1,rst);  
nand G3(temp3,q1,Clk);  
nand G4(temp4,qn,Clk);  
nand G1(Q,temp3,Qn);  
nand G2(Qn,temp4,Q,rst);  
endmodule

##### simulation

***Verilog Code (Simulation)***

`timescale 1ns / 1ps  
module T\_sim( );  
reg Clk,T,rst;  
wire Q,Qn;  
T\_flipflop usim(T,Clk,rst,Q,Qn);  
initial  
begin  
   {rst,Clk,T} = 3'b010;  
   #10 rst = 1'b1;  
   repeat(16)  
   #10 {Clk,T} = {Clk,T}+1;  
   $finish;  
end  
endmodule

***Wave Form (rst is low level effective)***



*When 0-10ns, rst = 0, then Q=0, Qn=1*

*When 10-170ns, rst =1:*

*When 50ns, T is 1, and Clk is in posedge, Qt = 0, so Qt+1 = 1 and Qt = 0*

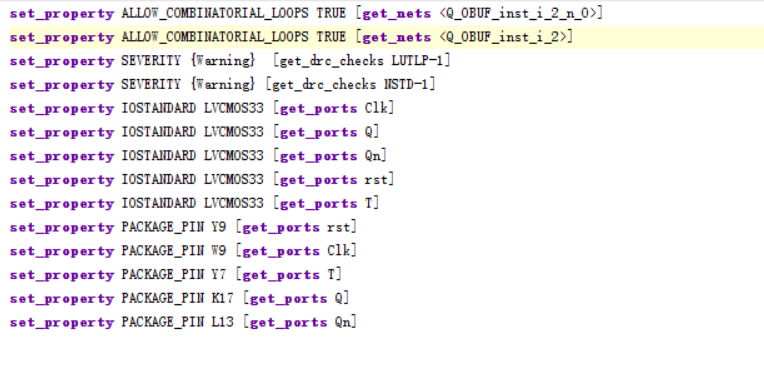
*When 90ns, T is 1, and Clk is in posedge, Qt = 1, so Qt+1 = 0 and Qt = 1*

*When 130ns, T is 1, and Clk is in posedge, Qt = 0, so Qt+1 = 1 and Qt = 0*

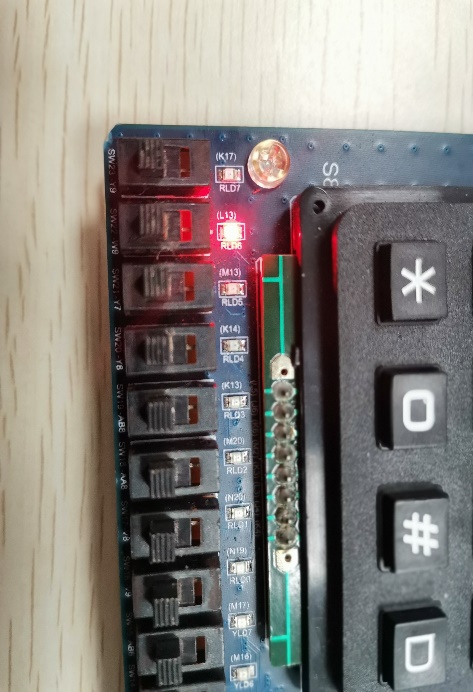
*From the wave form, we can get when Clk is in posedge and T=1, the state of Q will change, else the state of Q will not change. The result is satisfied T filp-flop.*

##### Constraint file and the testing

***Constraint File***



***Analysis***

**

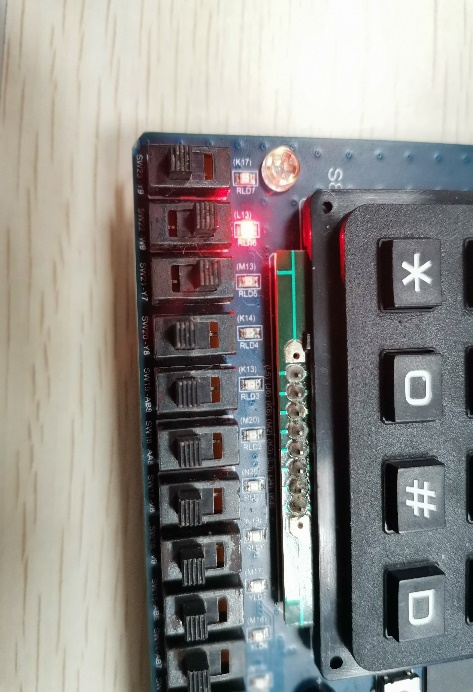
*Testcase #1*

*When rst is 0,*

*Then Q = 0, Qn = 1*

Q Qn

rst Clk T

**

*Testcase #2*

*Although Clk and T are 1,*

*Q = 0, Qn = 1 since rst = 0*

Q Qn

rst Clk T

**

*Testcase #3*

*When rst is 1, Clk and T are 1*

*Then the state will change,*

*Q = 1, Qn = 0*

Q Qn

rst Clk T

**

*Testcase #4*

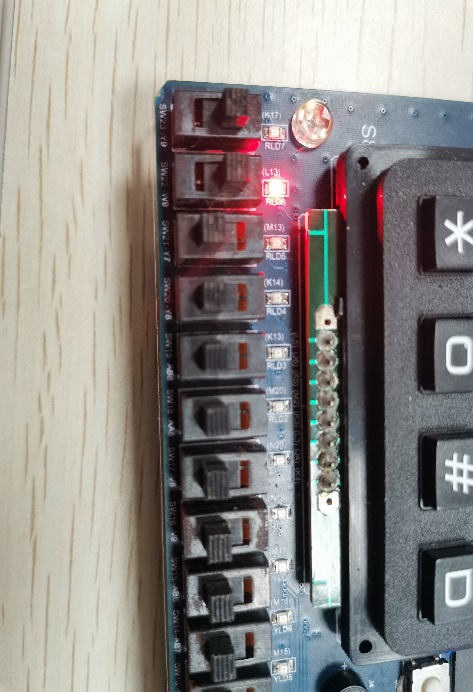
*When rst is 1, Clk and T are 1*

*Then the state will change,*

*Q = 0, Qn = 1*

Q Qn

rst Clk T

**

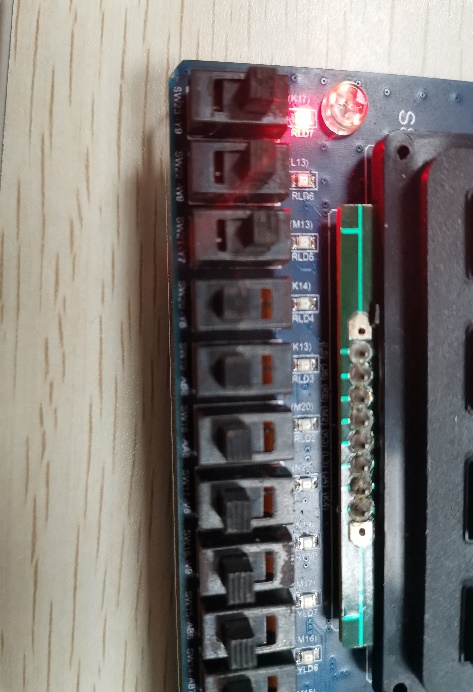
*Testcase #5*

*When rst is 1, Clk is 1,*

*Then the state will not change since T=0, Q = 0, Qn = 1*

Q Qn

rst Clk T

**

*Testcase #6*

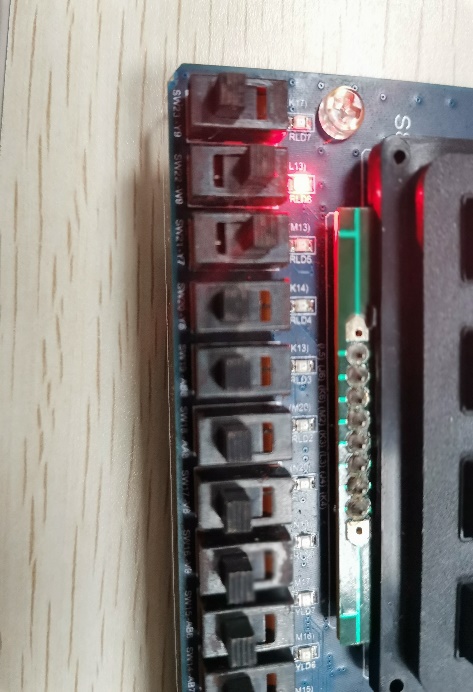
*When rst is 1, Clk and T are 1*

*Then the state will change,*

*Q = 1, Qn = 0*

Q Qn

rst Clk T

**

*Testcase #7*

*Although Clk and T are 1,*

*the state will reach original state since rst = 0, Q = 0, Qn = 1*

Q Qn

rst Clk T

PART 2: DIGITAL design LAB (Task2)

##### Design

***Verilog Code***

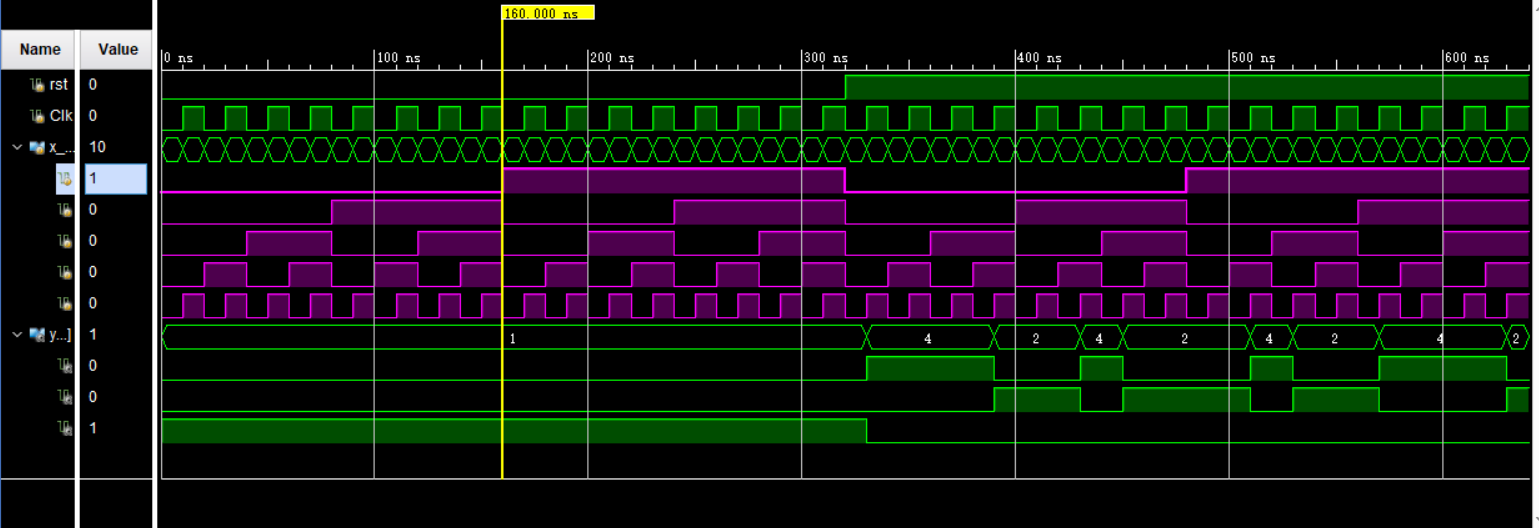
`timescale 1ns / 1ps  
module task2(Clk,rst,x\_in,y\_out);  
input Clk,rst;  
input [4:0]x\_in;  
output reg[2:0]y\_out;  
reg [2:0]y\_next;  
reg sum=0;  
parameter A=3'b001,B=3'b010,C=3'b100;  
always @(posedge Clk,negedge rst) begin  
   if(~rst) y\_next <= A;  
   else begin  
      sum = sum^x\_in[0]^x\_in[1]^x\_in[2]^x\_in[3]^x\_in[4];  
      if(sum)  y\_next <= C;  
      else y\_next <= B;  
   end  
end  
always @(y\_next) begin  
   y\_out <= y\_next;  
end  
endmodule

##### simulation

***Verilog Code (Simulation)***

`timescale 1ns / 1ps  
module task2\_sim( );  
reg Clk,rst;  
reg [4:0]x\_in;  
wire [2:0]y\_out;  
task2 usim(Clk,rst,x\_in,y\_out);  
initial begin     
   {Clk,rst,x\_in}=7'b0000000;  
   while({rst,x\_in}<6'b111111)  
   #10 {rst,x\_in} = {rst,x\_in}+1;  
   #10 $finish;  
end  
always #10 Clk=~Clk;  
endmodule

***Wave Form (rst is low level effective)***



*From the wave form:*

*When 0-320ns, rst = 0, so y\_out is state A(001)*

*When 320-640ns, rst=1:*

*When 330ns, the clock is in posedge, x\_in is (00001), the sum of 1 is 1, so the state is C(100)*

*When 350ns, the clock is in posedge, x\_in is (00011), the sum of 1 is 3, so the state is C(100)*

*When 370ns, the clock is in posedge, x\_in is (00101), the sum of 1 is 5, so the state is C(100)*

*When 390ns, the clock is in posedge, x\_in is (00111), the sum of 1 is 8, so the state is B(010)*

*When 410ns, the clock is in posedge, x\_in is (01001), the sum of 1 is 10, so the state is B(010)*

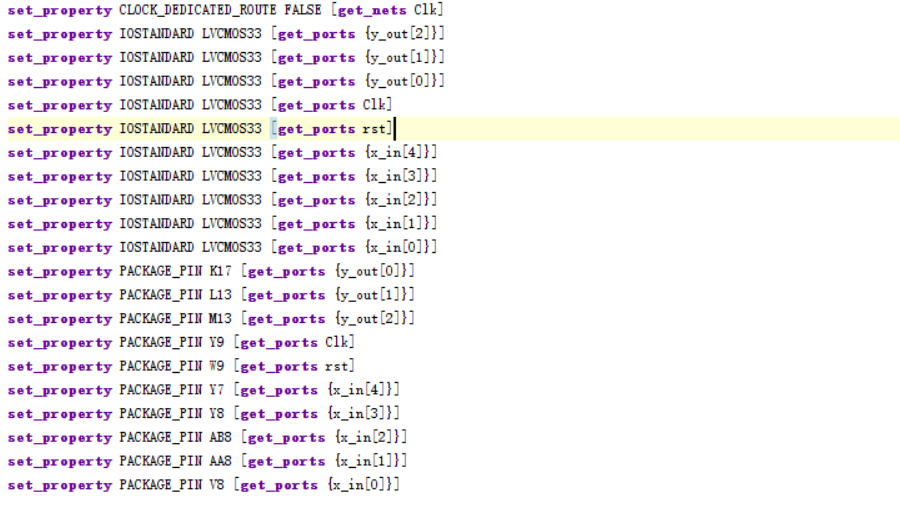
*When 430ns, the clock is in posedge, x\_in is (01011), the sum of 1 is 13, so the state is C(100)*

*……(Others are ignored)*

*From the wave form, we can conclude that if rst is effective, then the output state is A. Else we read the information of in\_x, and count the total number of 1, when the count is odd, the output state is C, else the output state is B. Such that, the result is satisfied.*

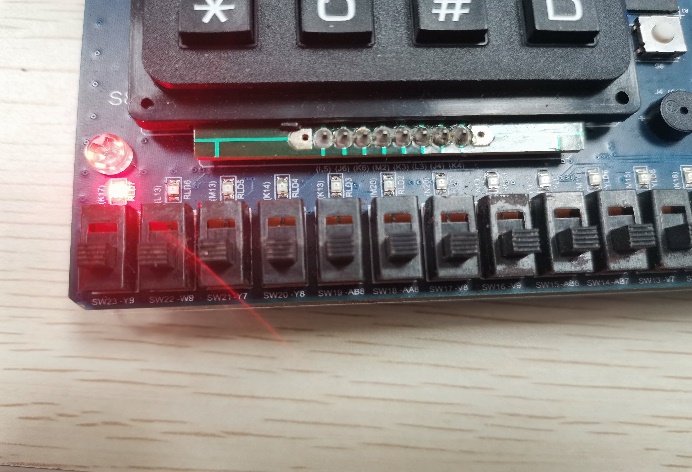
##### Constraint file and the testing

***Constraint File***



*Analysis*

*(In order to make the result looks more comfortable, I swap the order of (A,B,C) as below. Which means I link the higher level port with “M13”, lower with “K17”)*

**

Clk rst

*Testcase #1:*

*When rst=0, the sum of 1 is 0*

*The state is A (001)*

A B C

**

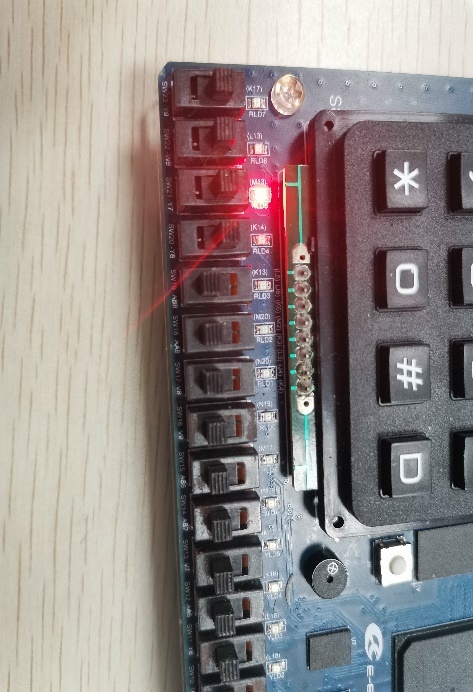
Clk rst

*Testcase #2:*

*When rst=1, and Clk is in posedge, x\_in(10000), the sum of 1 is 1*

*The state is C (100)*

A B C

**

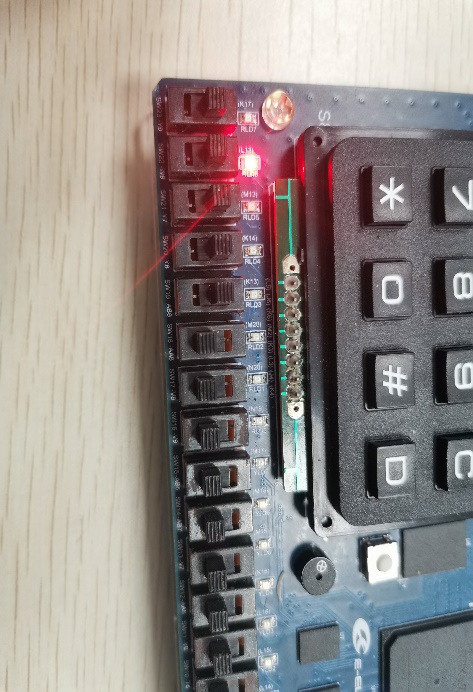
Clk rst

*Testcase #3:*

*When rst=1, and Clk is in posedge, x\_in(11000), the sum of 1 is 3*

*The state is C (100)*

A B C

**

Clk rst

*Testcase #4:*

*When rst=1, and Clk is in posedge, x\_in(11100), the sum of 1 is 6*

*The state is B (010)*

A B C

**

Clk rst

*Testcase #5:*

*When rst=1, and Clk is in posedge, x\_in(11111), the sum of 1 is 11*

*The state is C (100)*

A B C

**

Clk rst

*Testcase #6:*

*When rst=0, and Clk is in posedge, Although x\_in(11111), the sum of 1 is still 11 since rst = 0*

*The state is A (001)*

A B C

**

Clk rst

*Testcase #7:*

*When rst=1, and Clk is in posedge, x\_in(11111), the sum of 1 is 16*

*The state is B (010)*

A B C

##### the description of operation

*No problem*