DIGITAL DESIGN

ASSIGNMENTREPORT

ASSIGNMENT ID : 05

Student Name: Weibao Fu

Student ID: 11812202

PART 1: DIGITAL DESIGN THEORY

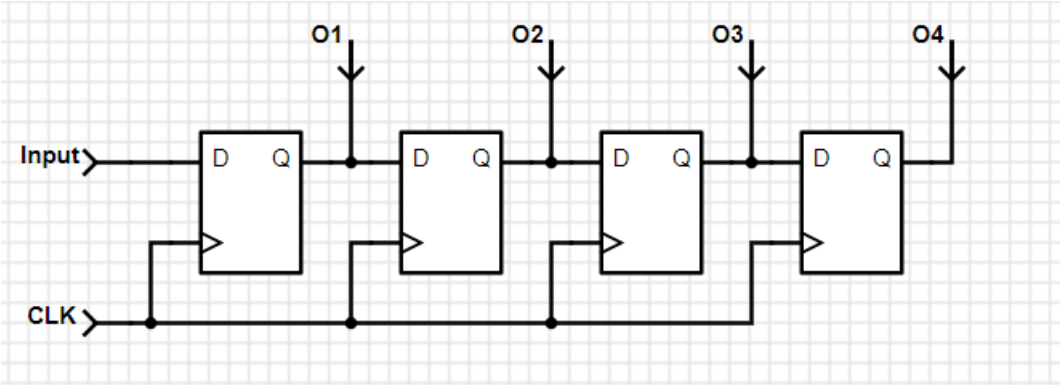
1.

Parallel transfer: Use multiple parallel data lines to transmit more than one bit at a time.

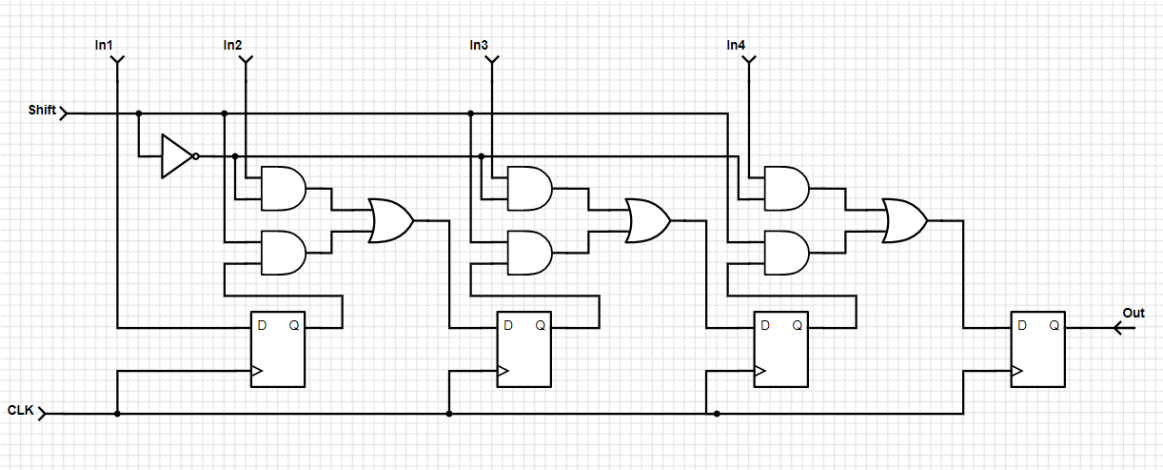
Serial transfer: Use a data line to transfer one bit at a time. If there exist more than one bit data, we should transfer them one after another.

Actually, we can use shift register to implement the transfer between serial and parallel. The way of implement is below;

Convert serial to parallel



Convert parallel to serial

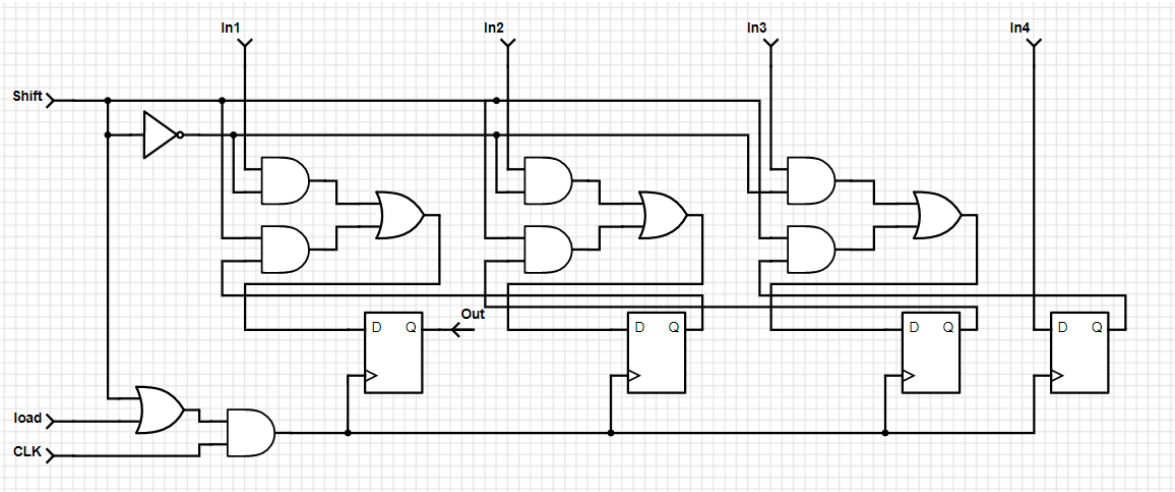


2.

We can get the following table

shift	load	Data
1	0	Left shift
1	1	Left shift
0	1	Load
0	0	Not change

We need to implement a shift left register, such that we only need to link the result of next D-flipflop to the previous input.

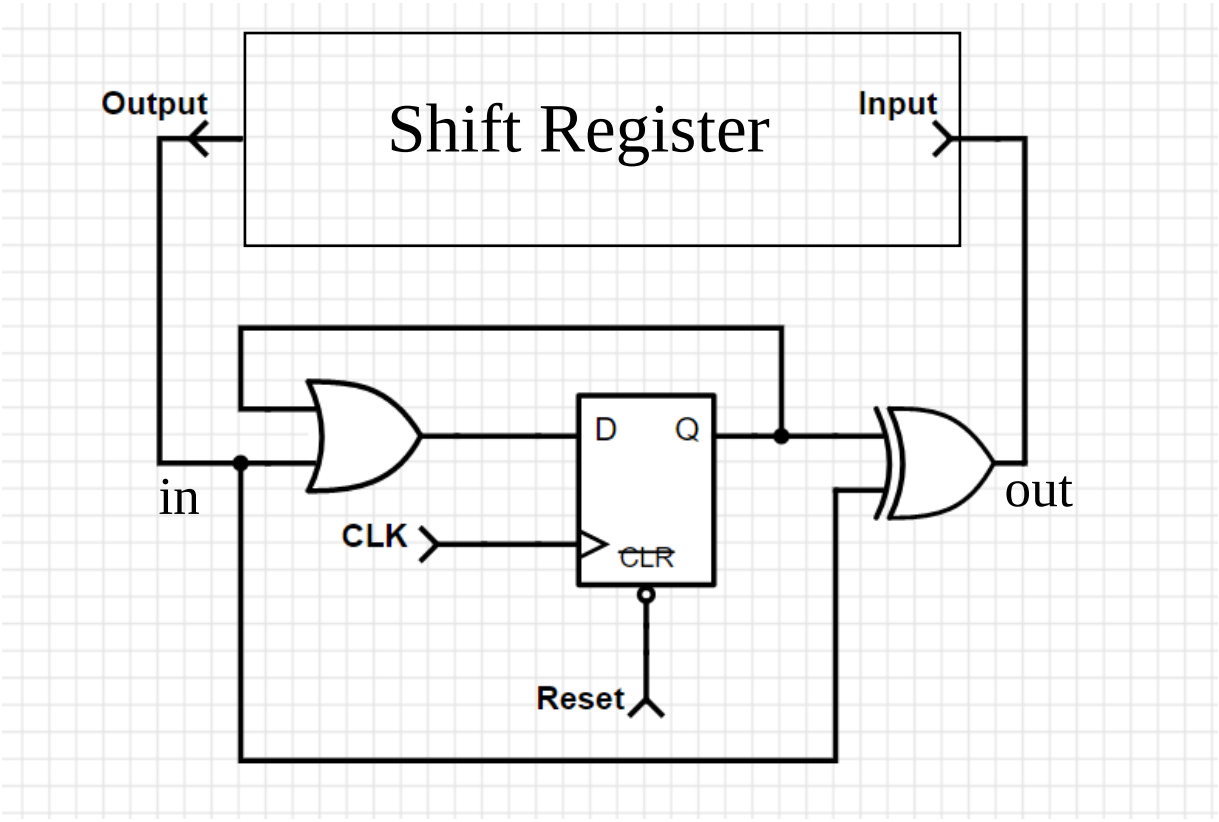


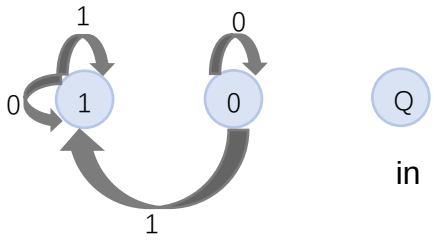
3.

Previous State		Next State	
Q_t	in	Q_{t+1}	out
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

We can conclude that in will be same to out until the first 1, so we can use the following diagram to implement it.

Shift Register





4.

Previous State				Next State			
A	B	C	E	A	B	C	E
0	0	1	0	1	0	0	1
1	0	0	1	0	1	0	0
0	1	0	0	1	0	1	0
1	0	1	0	1	1	0	1
1	1	0	1	0	1	1	0
0	1	1	0	1	0	1	1
1	0	1	1	0	1	0	1

0	1	0	1	0	0	1	0

We list all unused states in the table, we can notice that after 8 period, the state return to the initial state. Such that, when the counter is in an invalid state, it does not return to a valid state.

Previous State				Next State			
A	B	C	E	A	B	C	E
0	0	0	0	1	0	0	0
1	0	0	0	1	1	0	0
1	1	0	0	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	1	1	1
0	1	1	1	0	0	1	1
0	0	1	1	0	0	0	1
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
1	0	0	1	0	0	0	0
0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0
0	1	1	0	0	0	0	0
1	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0

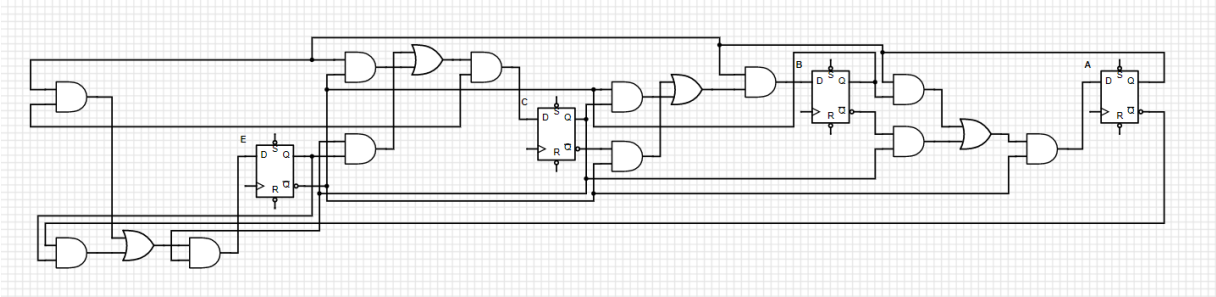
From the table, we can get $A_{t+1} = E'(A_tB+B'C)$

$$B_{t+1} = A(B_tC+C'E')$$

$$C_{t+1} = B(AE'+C_tE)$$

$$E_{t+1} = C(AB+A'E_t)$$

For every invalid state, the next state will be 0000, which return to the valid state. Such that the state equation can avoid the problem. The following is the implement of the state equation.



PART 2: DIGITAL DESIGN LAB (TASK1)

DESIGN

Design in Verilog (Shift Register 74194)

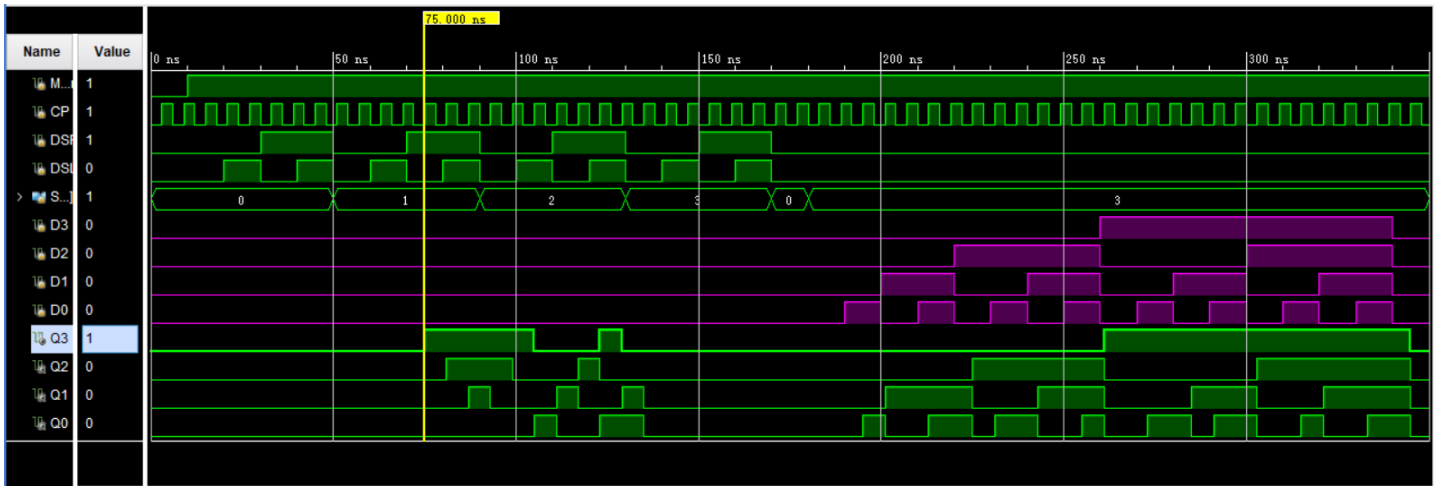
```
`timescale 1ns / 1ps
module SR_74194(
input MR_n, CP, DSR, DSL,
input [1:0]S,
input D3,D2,D1,D0,
output reg Q3,Q2,Q1,Q0
);
always @(posedge CP, negedge MR_n)
  if(!MR_n)
      {Q3,Q2,Q1,Q0} <= 4'b0000;
   else
      case(S)
      2'b00:{Q3,Q2,Q1,Q0} <= {Q3,Q2,Q1,Q0};
       2'b01:{Q3,Q2,Q1,Q0} <= {DSR,Q3,Q2,Q1};
```

```
        2'b10:{Q3,Q2,Q1,Q0} <= {Q2,Q1,Q0,DSL};
        2'b11:{Q3,Q2,Q1,Q0} <= {D3,D2,D1,D0};
        endcase
endmodule
```

Verilog Code (Simulation) *// To test the 74194*

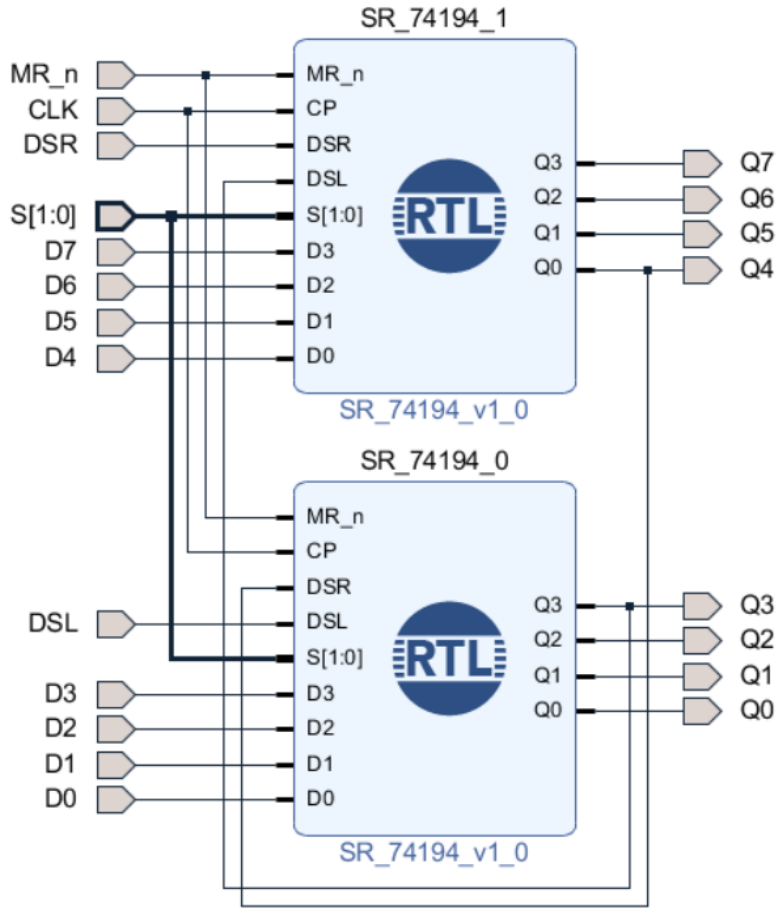
```
`timescale 1ns / 1ps
module sim_SR();
reg MR_n, CP, DSR, DSL;
reg [1:0]S;
reg D3,D2,D1,D0;
wire Q3,Q2,Q1,Q0;
SR_74194 simu(MR_n,CP,DSR,DSL,S,D3,D2,D1,D0,Q3,Q2,Q1,Q0);
initial
begin
   MR_n <= 0;
    CP <= 0;
   {S,DSR,DSL,D3,D2,D1,D0}=7'b00000000;
    #10 MR_n <= 1;
    while({S,DSR,DSL}<4'b1111)
   #10 {S,DSR,DSL} <= {S,DSR,DSL}+1;
    #10 S <= 2'b11;
    while({D3,D2,D1,D0}<4'b1111)
   #10 {D3,D2,D1,D0} <={D3,D2,D1,D0}+1;
    #10 $finish();
end
always #3 CP <= ~CP;
endmodule
```

Simulation of Shift Register 74194



From the simulation result, we can ensure that we implement the shift register 74194 successfully.

Block Diagram

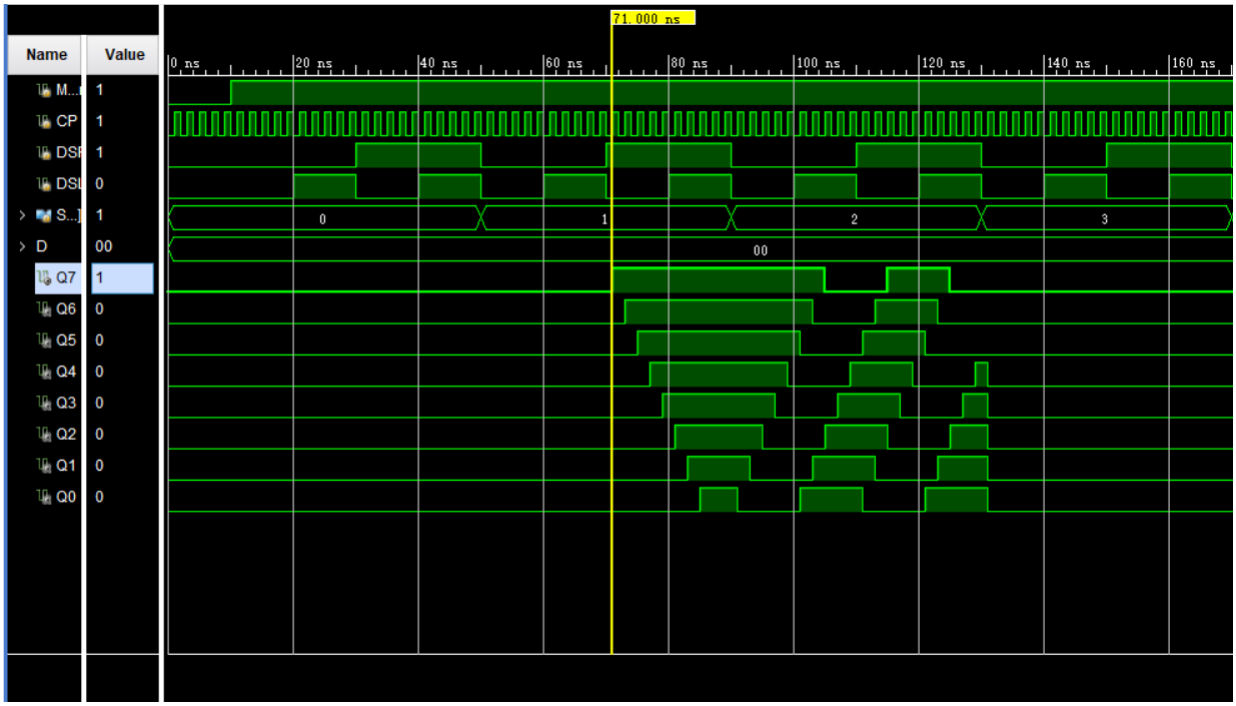


SIMULATION

Verilog Code (Simulation)

```
`timescale 1ns / 1ps
module sim_SR2();
reg MR_n, CP, DSR, DSL;
reg [1:0]S;
reg D7,D6,D5,D4,D3,D2,D1,D0;
wire Q7,Q6,Q5,Q4,Q3,Q2,Q1,Q0;
design_2_wrapper
simu(CP,D0,D1,D2,D3,D4,D5,D6,D7,DSL,DSR,MR_n,Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7,S);
initial
begin
   MR_n <= 0;
    CP <= 0;
    {S,DSR,DSL,D7,D6,D5,D4,D3,D2,D1,D0}=11'b00000000000;
    #10 MR_n <= 1;
    while({S,DSR,DSL}<4'b1111)
   #10 {S,DSR,DSL} <= {S,DSR,DSL}+1;
    #10 S <= 2'b11;
    repeat(20)
   #10 {D7,D6,D5,D4,D3,D2,D1,D0} = {D7,D6,D5,D4,D3,D2,D1,D0}+1;
    #10 $finish();
end
always #1 CP <= ~CP;
endmodule
```

Wave Form



From the wave form, we can get:

***When MR_n is 0**, then the output is 0000 0000*

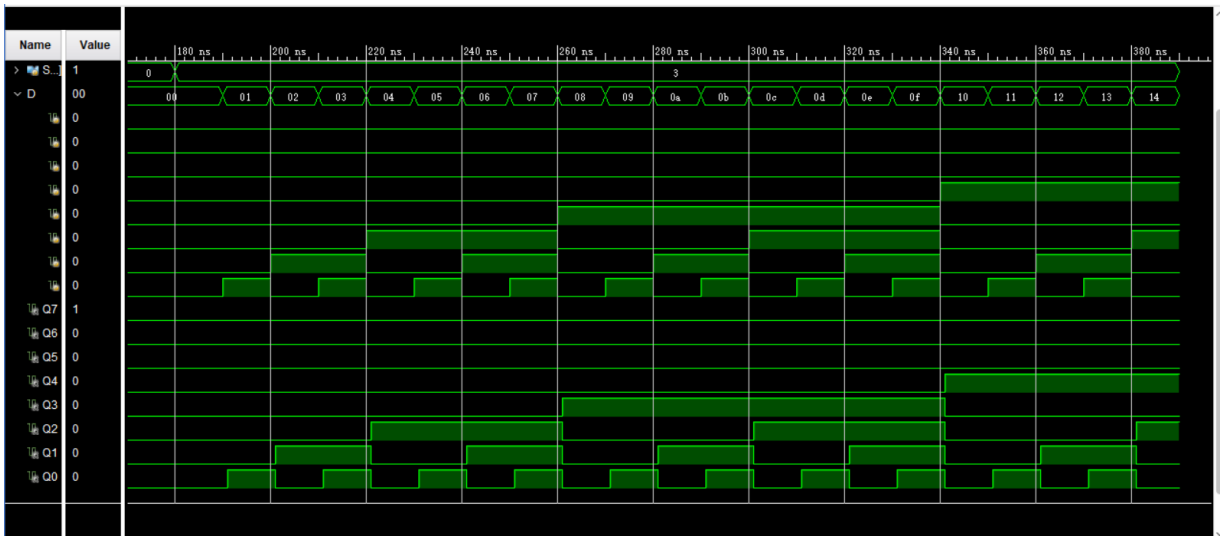
***When MR_n is 1**:*

When DSR is 1 and S = 2'b01, for every clock posedge, we can notice that the data in Q_i will transfer to Q_{i-1} ($1 \leq i \leq 7$), which mean the data will right shift.

When DSL is 1 and S = 2'b10, for every clock posedge, we can notice that the data in Q_i will transfer to Q_{i+1} ($0 \leq i \leq 6$), which mean the data will left shift.

When S = 2'b00, for every clock posedge, the data will not change.

When S = 2'b11, for every clock posedge, the output will be same as the data in D.



We can notice that, when S = 2'b11, and MR_n = 1, for every posedge of clock, the output Q will be same to the data of D.

From the analysis above, we can verify that the result of wave form is satisfied, which mean our implementation is correct.

CONSTRAINT FILE AND THE TESTING

Constraint File

```
set_property IOSTANDARD LVCMOS33 [get_ports Clk]
set_property IOSTANDARD LVCMOS33 [get_ports Q]
set_property IOSTANDARD LVCMOS33 [get_ports Qn]
set_property IOSTANDARD LVCMOS33 [get_ports rst]
set_property IOSTANDARD LVCMOS33 [get_ports T]
set_property PACKAGE_PIN Y9 [get_ports rst]
set_property PACKAGE_PIN W9 [get_ports Clk]
set_property PACKAGE_PIN Y7 [get_ports T]
set_property PACKAGE_PIN K17 [get_ports Q]
set_property PACKAGE_PIN L13 [get_ports Qn]
set_property IOSTANDARD LVCMOS33 [get_ports {S[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {S[1]}]
set_property PACKAGE_PIN W4 [get_ports {S[0]}]
set_property PACKAGE_PIN R4 [get_ports {S[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports CLK]
set_property IOSTANDARD LVCMOS33 [get_ports D0]
set_property IOSTANDARD LVCMOS33 [get_ports D1]
set_property IOSTANDARD LVCMOS33 [get_ports D2]
set_property IOSTANDARD LVCMOS33 [get_ports D3]
set_property IOSTANDARD LVCMOS33 [get_ports D4]
set_property IOSTANDARD LVCMOS33 [get_ports D5]
set_property IOSTANDARD LVCMOS33 [get_ports D6]
set_property PACKAGE_PIN Y9 [get_ports CLK]
set_property PACKAGE_PIN Y7 [get_ports DSL]
set_property PACKAGE_PIN W9 [get_ports MR_n]
set_property PACKAGE_PIN Y8 [get_ports DSR]
set_property PACKAGE_PIN AB8 [get_ports D7]
```

```
set_property PACKAGE_PIN AA8 [get_ports D6]
set_property PACKAGE_PIN V8 [get_ports D5]
set_property PACKAGE_PIN V9 [get_ports D4]
set_property PACKAGE_PIN AB6 [get_ports D3]
set_property PACKAGE_PIN AB7 [get_ports D2]
set_property PACKAGE_PIN V7 [get_ports D1]
set_property PACKAGE_PIN AA6 [get_ports D0]
set_property IOSTANDARD LVCMOS33 [get_ports D7]
set_property IOSTANDARD LVCMOS33 [get_ports DSL]
set_property IOSTANDARD LVCMOS33 [get_ports DSR]
set_property IOSTANDARD LVCMOS33 [get_ports MR_n]
set_property IOSTANDARD LVCMOS33 [get_ports Q0]
set_property IOSTANDARD LVCMOS33 [get_ports Q1]
set_property IOSTANDARD LVCMOS33 [get_ports Q2]
set_property IOSTANDARD LVCMOS33 [get_ports Q3]
set_property IOSTANDARD LVCMOS33 [get_ports Q4]
set_property IOSTANDARD LVCMOS33 [get_ports Q5]
set_property IOSTANDARD LVCMOS33 [get_ports Q6]
set_property IOSTANDARD LVCMOS33 [get_ports Q7]
set_property PACKAGE_PIN K17 [get_ports Q7]
set_property PACKAGE_PIN L13 [get_ports Q6]
set_property PACKAGE_PIN M13 [get_ports Q5]
set_property PACKAGE_PIN K14 [get_ports Q4]
set_property PACKAGE_PIN K13 [get_ports Q3]
set_property PACKAGE_PIN M20 [get_ports Q2]
set_property PACKAGE_PIN N20 [get_ports Q1]
set_property PACKAGE_PIN N19 [get_ports Q0]
```

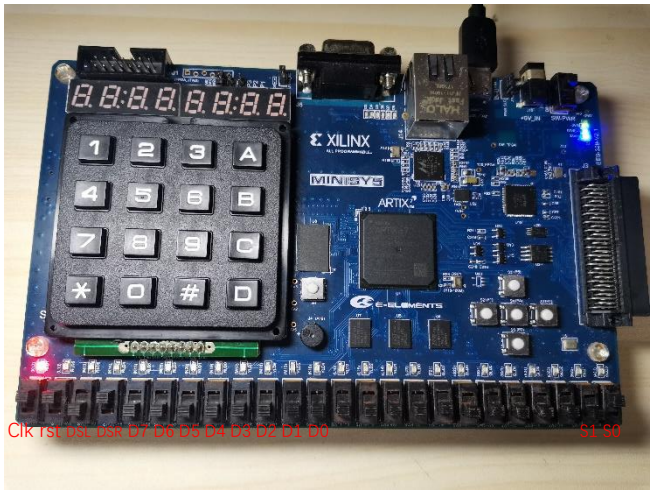
Analysis

Testcase #1

When rst is 0, and Clk is 0, then we notice that there will be the initial state. (No input)

Testcase #2

When rst is 0, and Clk is 1, then we notice that there will be the initial state. (No input)

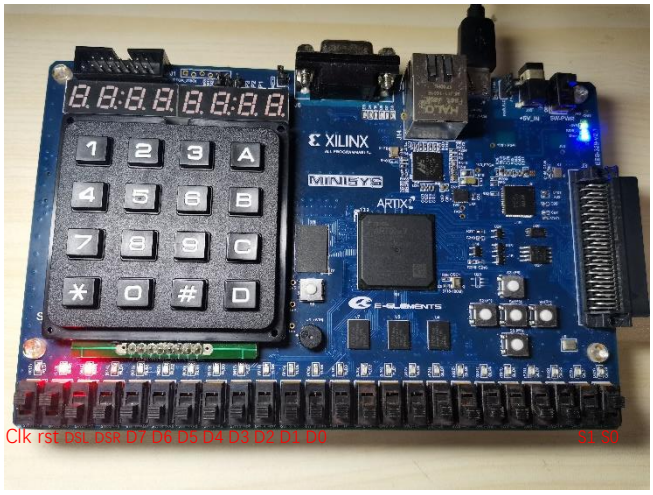


Testcase #3

*When rst is 1, and set the S = 01, which mean we input the data by serial. Set **DSR = 1**, We turn the Clk off and then turn it on, we notice the data is transferred.*

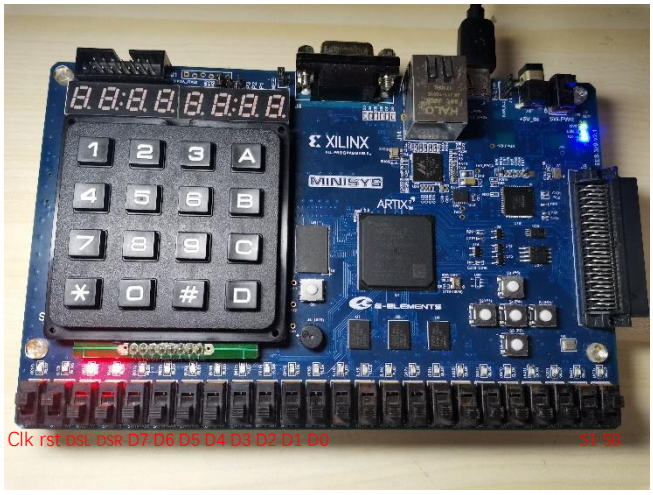
Testcase #4

*When rst is 1, and set the S = 01, which mean we input the data by serial. Set **DSR = 1**, We turn the Clk off and then turn it on, we notice the data is transferred. And the light is right shift.*



Testcase #5

*When rst is 1, and set the S = 01, which mean we input the data by serial. Set **DSR = 0**, We turn the Clk off and then turn it on, we notice the data is transferred. And the light is right shift.*



Testcase #6

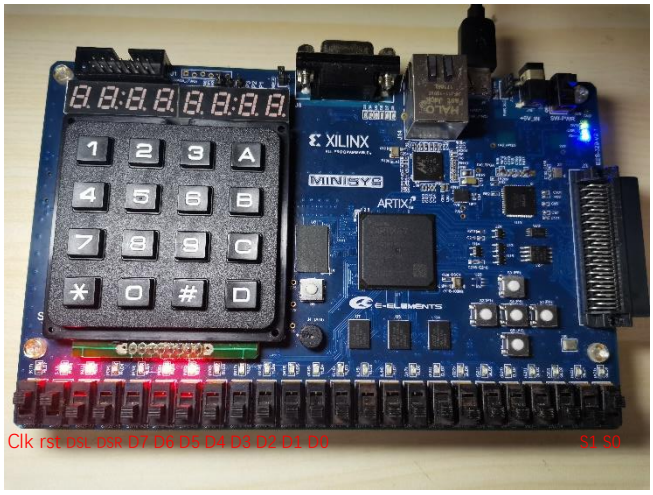
*When rst is 1, and set the S = 01, which mean we input the data by serial. Set **DSR = 0**, We turn the Clk off and then turn it on, we notice the data is transferred. And the light is right shift.*

Testcase #7

*When rst is 1, and set the S = 01, which mean we input the data by serial. Set **DSR = 1**, We turn the Clk off and then turn it on, we notice the data is transferred. And the light is right shift.*

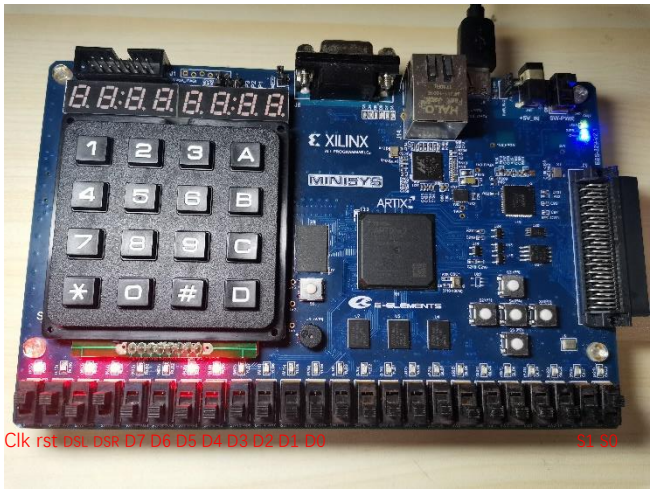
Testcase #8

*When rst is 1, and set the S = 01, which mean we input the data by serial. Set **DSR = 1**, We turn the Clk off and then turn it on, we notice the data is transferred. And the light is right shift.*



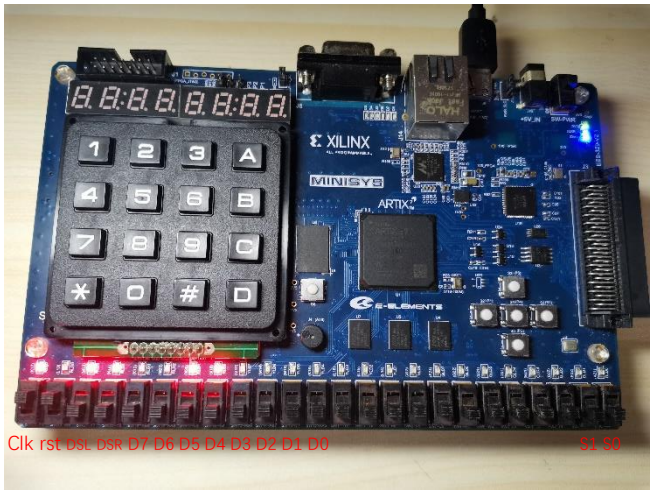
Testcase #9

*When rst is 1, and set the S = 01, which mean we input the data by serial. Set **DSR = 0**, We turn the Clk off and then turn it on, we notice the data is transferred. And the light is right shift.*



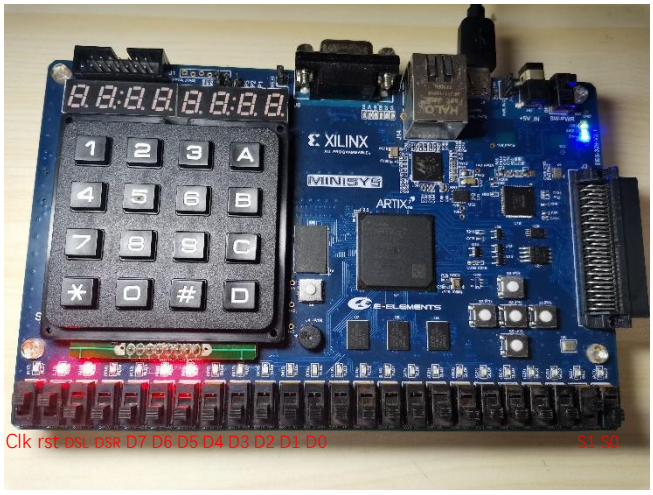
Testcase #10

*When rst is 1, and set the S = 01, which mean we input the data by serial. Set **DSR = 1**, We turn the Clk off and then turn it on, we notice the data is transferred. And the light is right shift.*



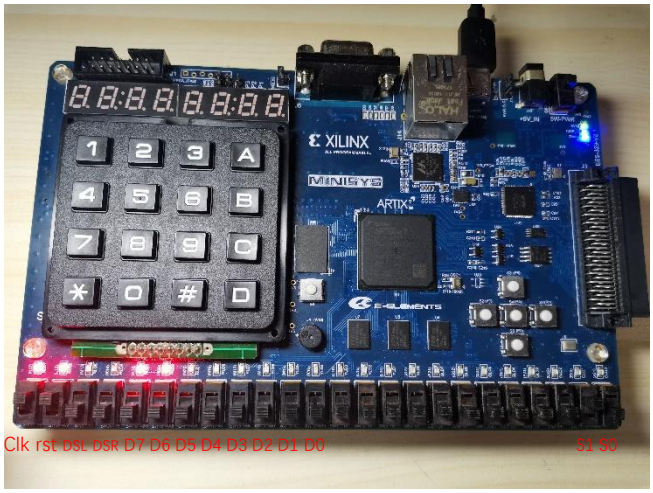
Testcase #11

*When rst is 1, and set the S = 00, which mean want the data maintain. Set **DSR = 0**, We turn the Clk off and then turn it on, we notice the light does not change, which mean the data maintain.*



Testcase #12

*When rst is 1, and set the S = 10, which mean we input the data by serial. Set **DSL = 0**, We turn the Clk off and then turn it on, we notice the data is transferred. And the light is left shift.*



Testcase #13

*When rst is 1, and set the S = 10, which mean we input the data by serial. Set **DSL = 0**, We turn the Clk off and then turn it on, we notice the data is transferred. And the light is left shift.*

Testcase #14

*When rst is 1, and set the S = 10, which mean we input the data by serial. Set **DSL = 0**, We turn the Clk off and then turn it on, we notice the data is transferred. And the light is left shift.*

Testcase #15

*When rst is 1, and set the S = 10, which mean we input the data by serial. Set **DSL = 0**, We turn the Clk off and then turn it on, we notice the data is transferred. And the light is left shift.*

Testcase #16

*When rst is 1, and set the S = 10, which mean we input the data by serial. Set **DSL = 0**, We turn the Clk off and then turn it on, we notice the data is transferred. And the light is left shift.*

Testcase #17

*When rst is 1, and set the S = 11, which mean we input the data by parallel. Set **D7~D0 to 10110110**, We turn the Clk off and then turn it on, we notice the data is transferred.*

Testcase #18

When rst is 0, and then we notice that all lights are off, which mean we return to the initial state.

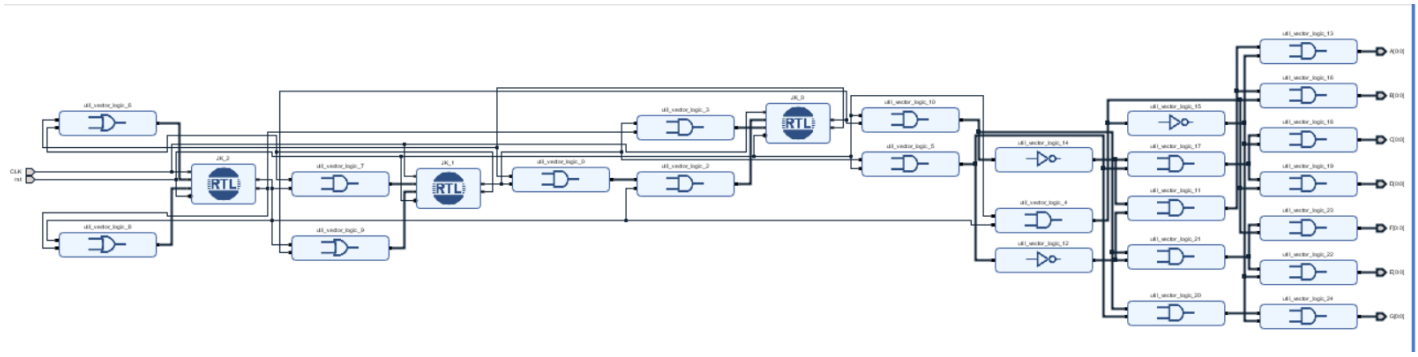
PART 2: DIGITAL DESIGN LAB (TASK2)

DESIGN

Design Code (JK flip_flop)

```
`timescale 1ns / 1ps
module JK(
input CLK,
input J,K,
input rst,
output reg Q,
output Qn
);
assign Qn=~Q;
always@(posedge CLK)
begin
   if(!rst) Q <= 0;
    else
   Q <= (J&~Q)||(~K&Q);
end
endmodule
```

Block Diagram

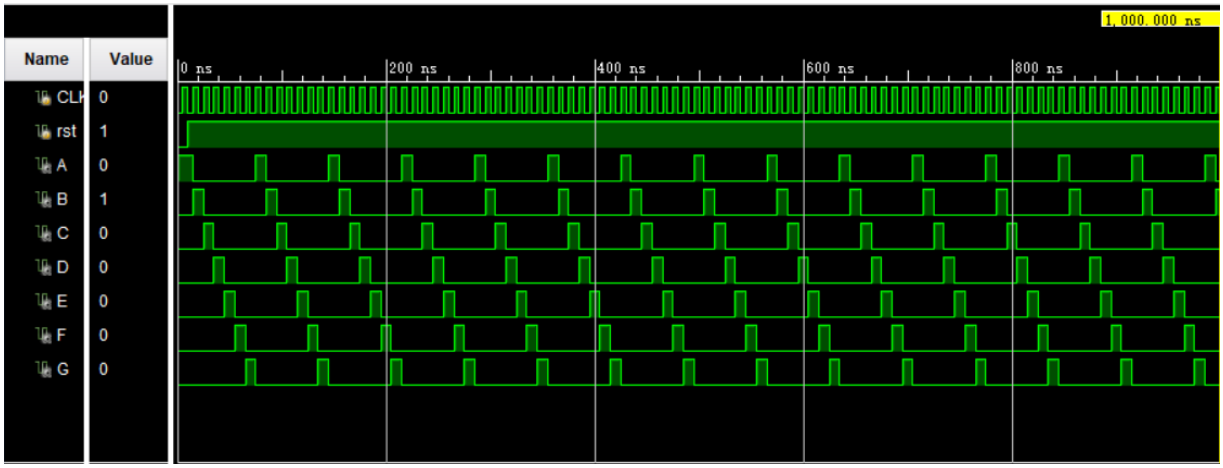


SIMULATION

Verilog Code (Simulation)

```
`timescale 1ns / 1ps
module sim1(
    );
reg CLK,rst;
wire A,B,C,D,E,F,G;
design_1_wrapper usim(A,B,C,CLK,D,E,F,G,rst);
initial
begin
    rst = 0;
     CLK = 0;
     #10 rst = 1;
end
always #5 CLK = ~CLK;
endmodule
```

Wave Form



From the Wave Form, we can get:

When rst = 0, *then we set counter to 0, such that output ABCDEFG is 1000000, which mean the number is 0.*

When rst = 1:

We can notice that for every posedge of clock, the counter will change, (1000000) -> (0100000) -> (0010000) -> (0001000) -> (0000100) -> (0000010) -> (0000001) -> (1000000), which mean it repeat the number 0,1,2,3,4,5,6.

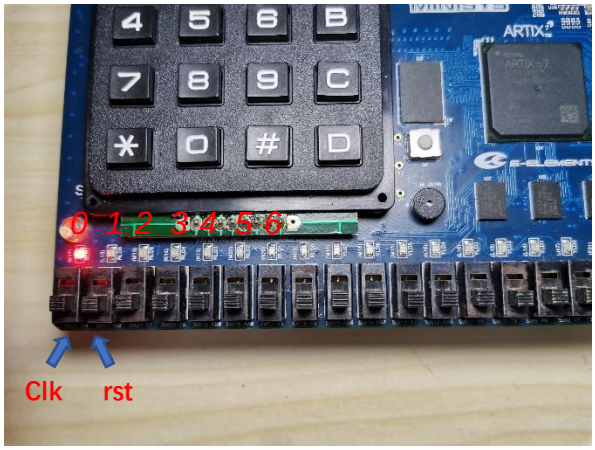
Such that, we can verify the result of wave form is satisfied and our implementation is correct.

CONSTRAINT FILE AND THE TESTING

Constraint File

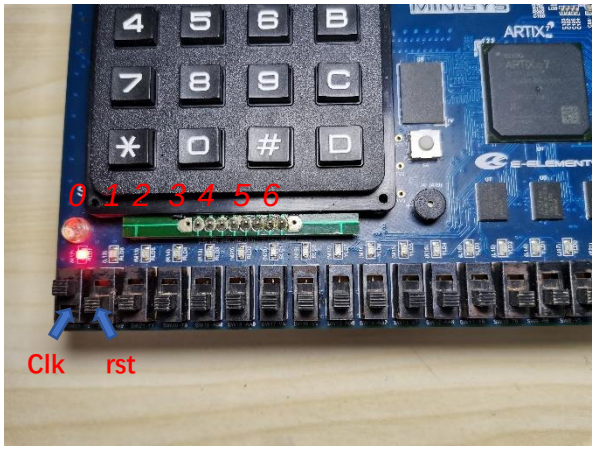
```
set_property IOSTANDARD LVCMOS33 [get_ports CLK]
set_property PACKAGE_PIN Y9 [get_ports CLK]
set_property IOSTANDARD LVCMOS33 [get_ports rst]
set_property PACKAGE_PIN W9 [get_ports rst]
set_property IOSTANDARD LVCMOS33 [get_ports {A[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {B[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {C[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {D[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {E[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {F[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {G[0]}]
set_property PACKAGE_PIN K17 [get_ports {A[0]}]
set_property PACKAGE_PIN L13 [get_ports {B[0]}]
set_property PACKAGE_PIN M13 [get_ports {C[0]}]
set_property PACKAGE_PIN K14 [get_ports {D[0]}]
set_property PACKAGE_PIN K13 [get_ports {E[0]}]
set_property PACKAGE_PIN M20 [get_ports {F[0]}]
set_property PACKAGE_PIN N20 [get_ports {G[0]}]

```

Analysis



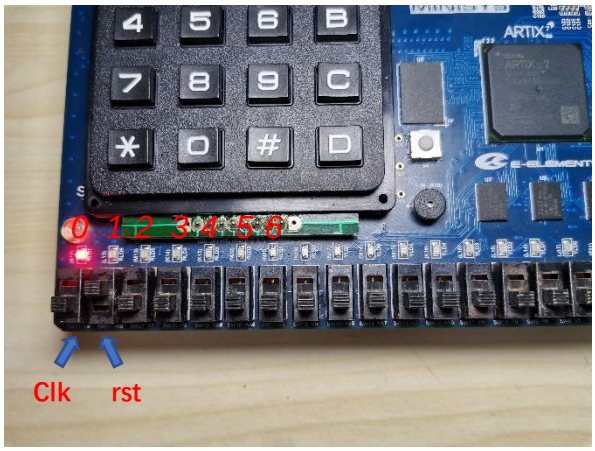
Testcase #1

When Clk and rst are both 0, then the number is 0



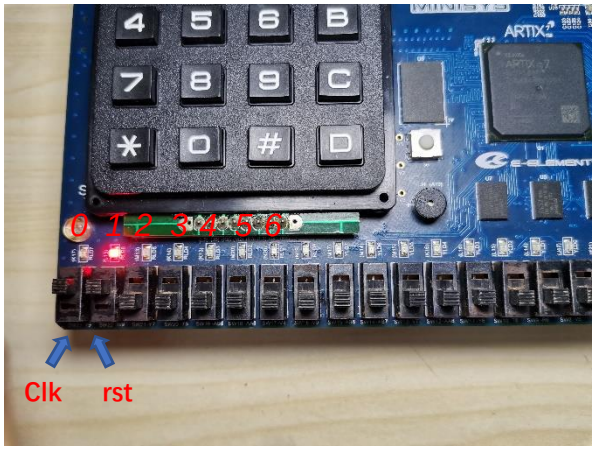
Testcase #2

When rst is 0, and Clk is 1, then the number is still 0



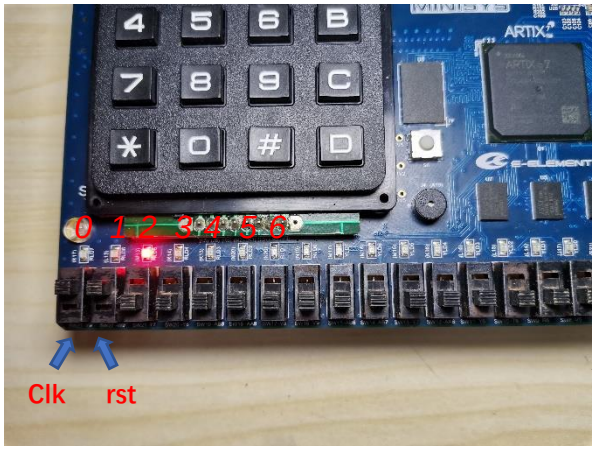
Testcase #3

When rst is 1, Clk is 0, then the number is 0



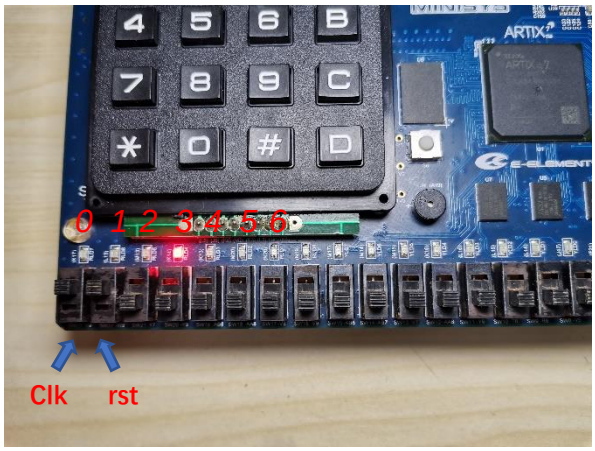
Testcase #4

When rst is 1, we turn the Clk off and then turn the Clk on, then the number is 1



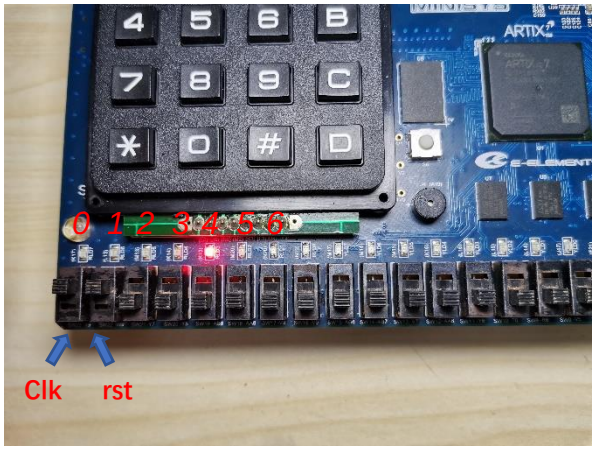
Testcase #5

When rst is 1, we turn the Clk off and then turn the Clk on, then the number is 2



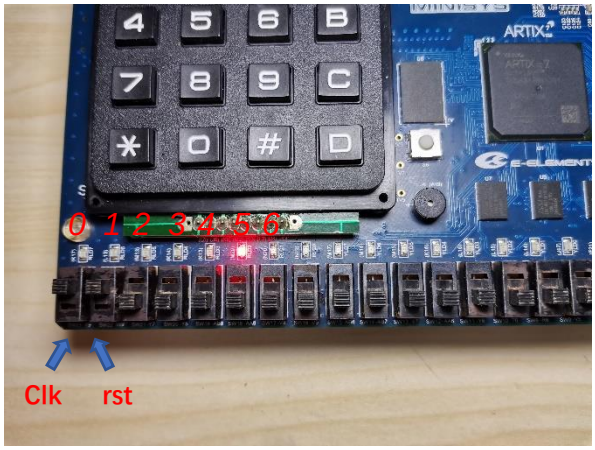
Testcase #6

When rst is 1, we turn the Clk off and then turn the Clk on, then the number is 3



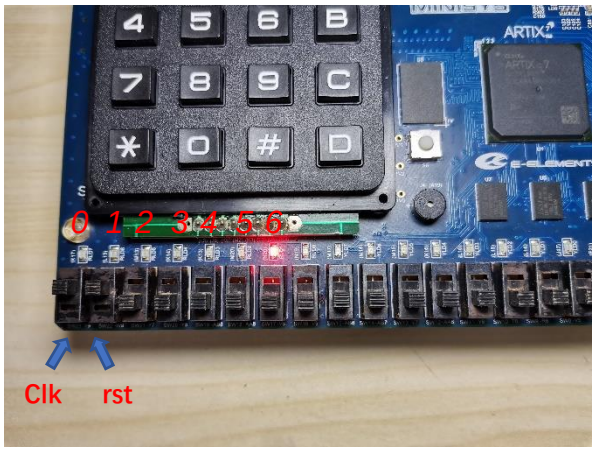
Testcase #7

When rst is 1, we turn the Clk off and then turn the Clk on, then the number is 4



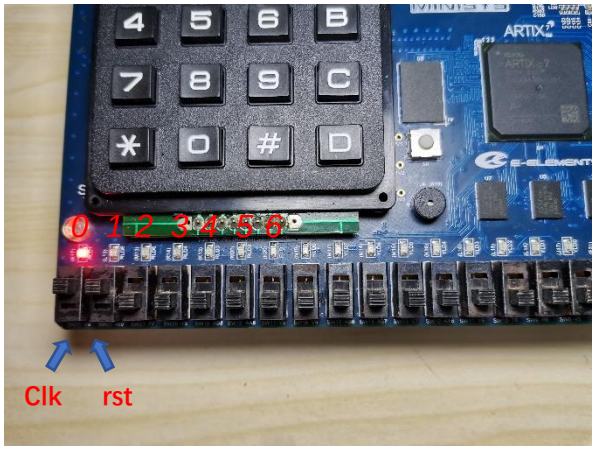
Testcase #7

When rst is 1, we turn the Clk off and then turn the Clk on, then the number is 5



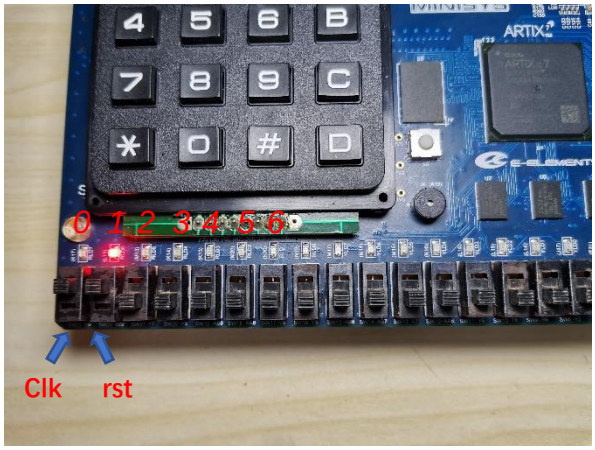
Testcase #8

When rst is 1, we turn the Clk off and then turn the Clk on, then the number is 6



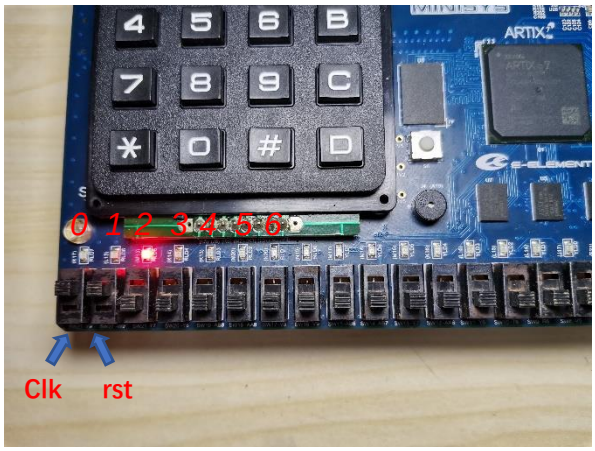
Testcase #9

When rst is 1, we turn the Clk off and then turn the Clk on, then the number is 0



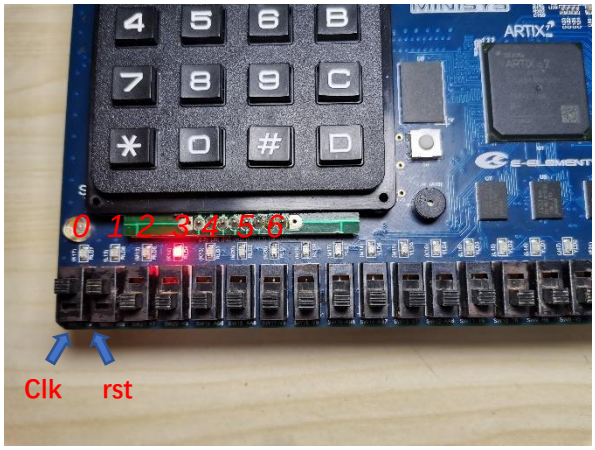
Testcase #10

When rst is 1, we turn the Clk off and then turn the Clk on, then the number is 1



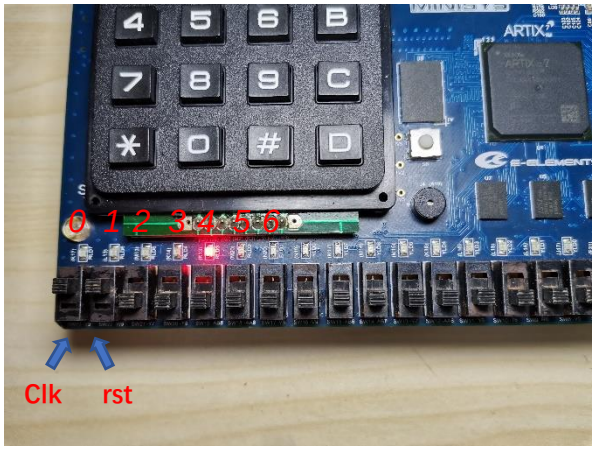
Testcase #11

When rst is 1, we turn the Clk off and then turn the Clk on, then the number is 2



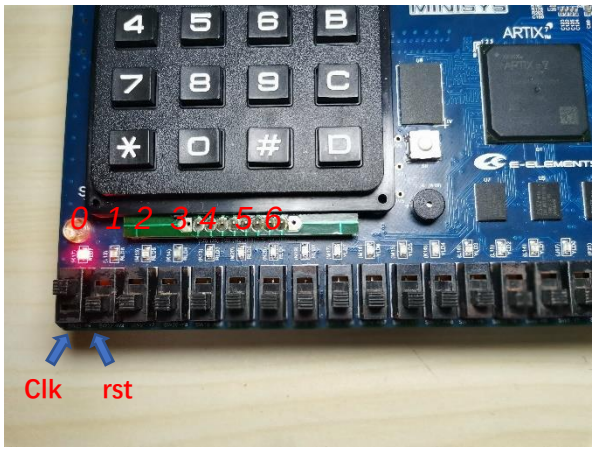
Testcase #12

When rst is 1, we turn the Clk off and then turn the Clk on, then the number is 3



Testcase #13

When rst is 1, we turn the Clk off and then turn the Clk on, then the number is 4



Testcase #14

When rst is 0, we turn the Clk off and then turn the Clk on, then the number is 0

THE DESCRIPTION OF OPERATION

No problem.