数电实验8

> ## 实验题目
>
> - **信号处理与有限状态机**
>
> ## 实验目的
>
> - 进一步熟悉 FPGA 开发的整体流程
> - 掌握几种常见的信号处理技巧
> - 掌握有限状态机的设计方法
- 能够使用有限状态机设计功能电路

> ## 实验环境
>
> - VLAB：vlab.ustc.eud.cn
> - FPGAOL：fpgaol.ustc.edu.cn
> - Logisim、Vivado

> ## 实验步骤
>
> Step1.信号整形及去毛刺
>
> Step2.取信号边沿技巧
>
> Step3. 有限状态机介绍

Step4.有限状态机 Verilog 实现

T1

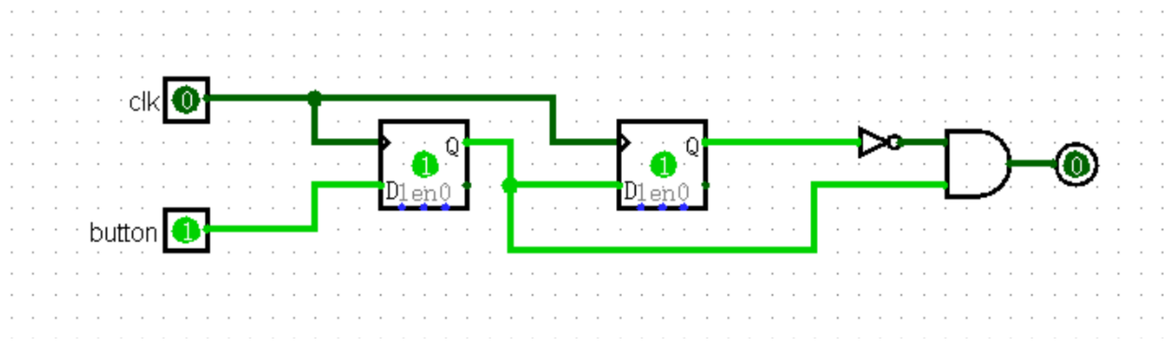
> 修改为三部分有限状态机

```
module test(input clk,rst, output led);
parameter a = 2'b00;
parameter b = 2'b01;
parameter c = 2'b10;
parameter d = 2'b11;
reg [1:0] cnt;
reg [1:0] next_cnt;
//有限状态机第一部分
always@(*)
    begin
        case(cnt)
            a:next_cnt <= 2'b01;
            b:next_cnt <= 2'b10;
            c:next_cnt <= 2'b11;
```

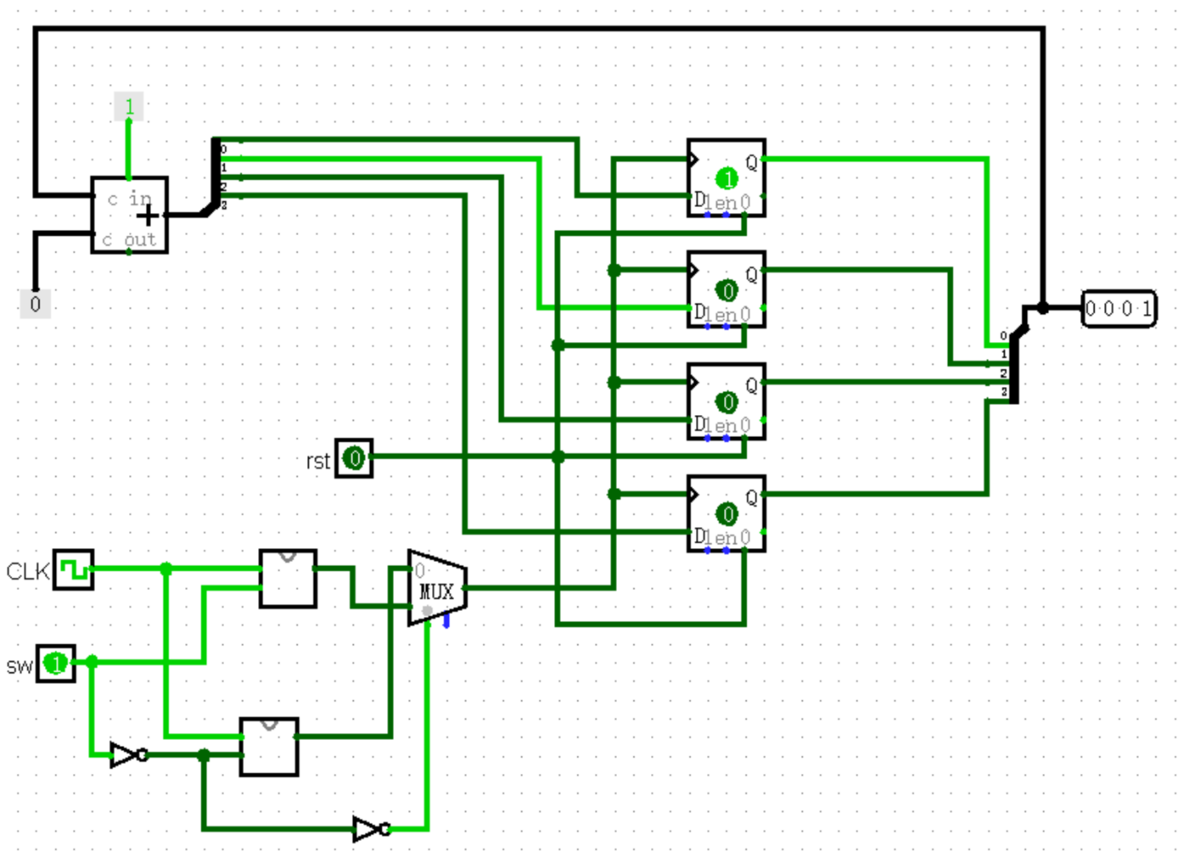
```
                d:next_cnt <= 2'b00;
            endcase
        end
//有限状态机第二部分
always@(posedge clk or posedge rst)
        begin
            if(rst)
                cnt <= 2'b00;
            else
                cnt<=next_cnt;
        end
//有限状态机第三部分
        assign led = (cnt==2'b11) ? 1'b1 : 1'b0;
endmodule
```

T2

> 实例化模块生成脉冲信号电路:



> 计数器电路:
>
> 通过四个D触发器和上述例化的模块完成计数，sw上升或下降就在右侧寄存器完成计数，rst表示置位信号，rst=1则寄存器置0。



T3

> 信号整形去毛刺：

```
module jitter_clr(
input clk,
input button,
output button_clean);
    reg [9 :0] cnt;
always@(posedge clk) begin
    if(button==1'b0) cnt <= 10'b00000_00000;
else if(cnt < 512) cnt <= cnt + 10'b00000_00001;
end
assign button_clean = cnt[9];
endmodule
```

> 取信号边沿：

```
module signal_edge(//取信号边沿
input clk,
input button,
output button_redge);
reg button_r1,button_r2;
    always@(posedge clk)
        button_r1 <= button;
    always@(posedge clk)
        button_r2 <= button_r1;
    assign button_redge = button_r1 &(~button_r2);
endmodule
```

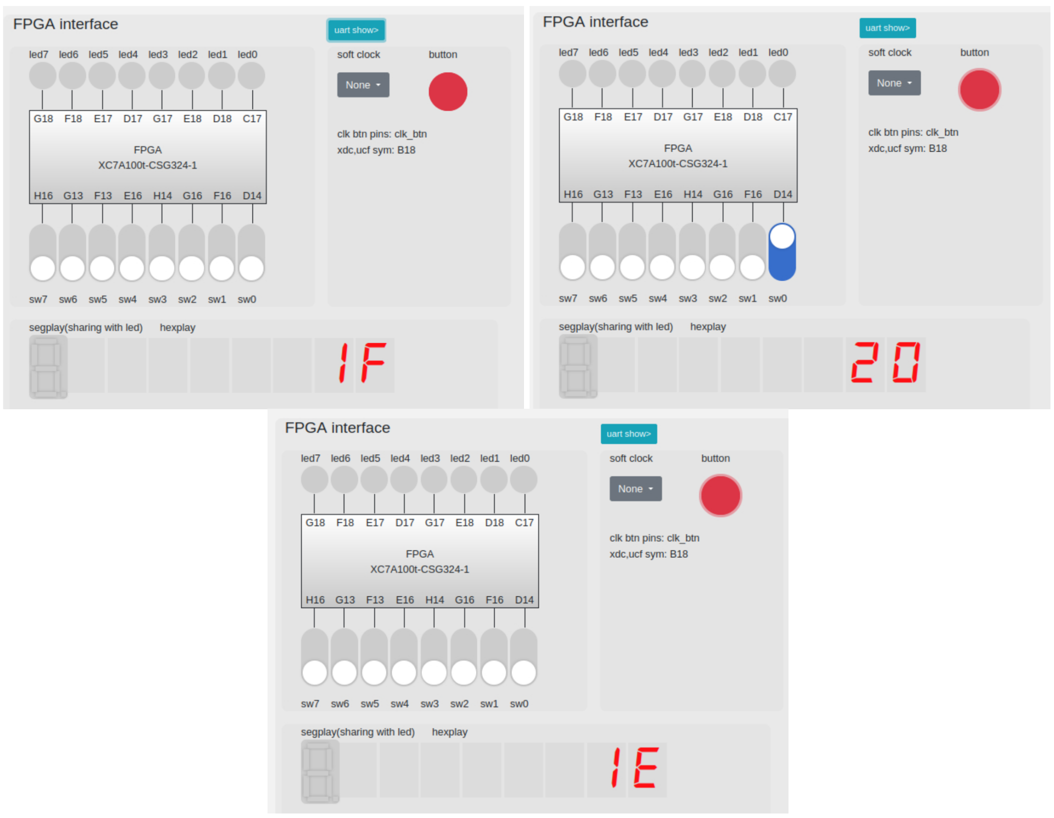
> 设计文件:

```verilog
module Counter(
input CLK,rst,button,sw,
output reg [3:0] d,
output reg an
);
wire add_cleant, sub_cleant, add_clean, sub_clean;
reg [7:0] counter;
reg [2:0] cnt;

jitter_clr j1(CLK, button, add_cleant);
signal_edge e1(CLK, add_cleant, add_clean);

always@(posedge CLK)
    begin
        if(rst)
            counter<=8'b0001_1111;
        else
            begin
                if(sw==1)
                    begin
                        if(add_clean)
                        counter <= counter + 8'b0000_0001;
                    end
                else
                    begin
                        if(add_clean)
                            counter <= counter - 8'b0000_0001;
                    end
            end
    end
always@(posedge CLK)
    begin
        cnt=cnt+1;
        an=cnt[2];
        if(an) d<=counter[7:4];//an不等于0, 表示cnt[2]不为0, 表示的就是前四个数码管
        else d<=counter[3:0];//后四个数码管
    end
endmodule
```

> 约束文件:

```
set_property -dict { PACKAGE_PIN E3   IOSTANDARD LVCMOS33 } [get_ports { CLK }];

set_property -dict { PACKAGE_PIN D14   IOSTANDARD LVCMOS33 } [get_ports { sw }];
set_property -dict { PACKAGE_PIN F16   IOSTANDARD LVCMOS33 } [get_ports { rst }];

set_property -dict { PACKAGE_PIN A14   IOSTANDARD LVCMOS33 } [get_ports { d[0] }];
set_property -dict { PACKAGE_PIN A13   IOSTANDARD LVCMOS33 } [get_ports { d[1] }];
set_property -dict { PACKAGE_PIN A16   IOSTANDARD LVCMOS33 } [get_ports { d[2] }];
set_property -dict { PACKAGE_PIN A15   IOSTANDARD LVCMOS33 } [get_ports { d[3] }];
set_property -dict { PACKAGE_PIN B17   IOSTANDARD LVCMOS33 } [get_ports { an }];

set_property -dict { PACKAGE_PIN B18   IOSTANDARD LVCMOS33 } [get_ports { button }];
```

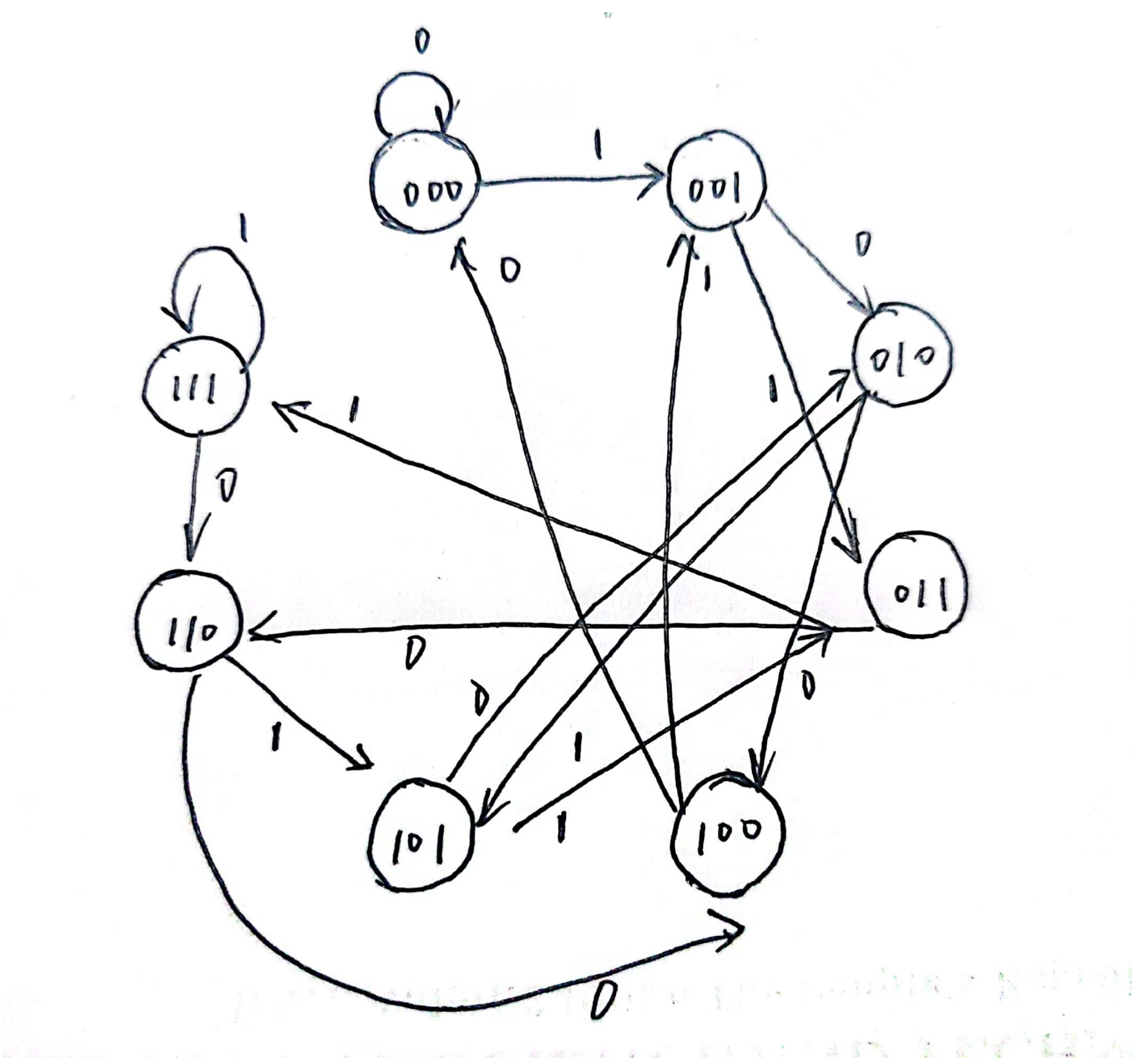
> 烧写显示结果


FPGA interface
uart show>
led7 led6 led5 led4 led3 led2 led1 led0
G18 F18 E17 D17 G17 E18 D18 C17
FPGA
XC7A100t-CSG324-1
H16 G13 F13 E16 H14 G16 F16 D14
sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0
soft clock
button
None
clk btn pins: clk_btn
xdc,ucf sym: B18
segplay(sharing with led)
hexplay
1F
20
1E

当sw[0]为0时递减，当sw[0]为1时递增。

T4

> 状态图：



> 信号整形去毛刺:

```
module jitter_clr(
input clk,
input button,
output button_clean);
reg [9:0] cnt;
always@(posedge clk)
    begin
        if(button==1'b0) cnt <= 10'b00000_00000;
        else if(cnt < 512) cnt <= cnt + 10'b00000_00001;
    end
assign button_clean = cnt[9];
endmodule
```

> 取信号边沿:

```
module signal_edge(
input clk,
input button,
output button_redge);
reg button_r1,button_r2;
    always@(posedge clk)
        button_r1<= button;
    always@(posedge clk)
        button_r2<= button_r1;
assign button_redge = button_r1 &(~button_r2);
endmodule
```

> 约束文件:

```
set_property -dict { PACKAGE_PIN E3   IOSTANDARD LVCMOS33 } [get_ports { CLK }];

set_property -dict { PACKAGE_PIN A14   IOSTANDARD LVCMOS33 } [get_ports { d[0] }];
set_property -dict { PACKAGE_PIN A13   IOSTANDARD LVCMOS33 } [get_ports { d[1] }];
set_property -dict { PACKAGE_PIN A16   IOSTANDARD LVCMOS33 } [get_ports { d[2] }];
set_property -dict { PACKAGE_PIN A15   IOSTANDARD LVCMOS33 } [get_ports { d[3] }];
set_property -dict { PACKAGE_PIN B17   IOSTANDARD LVCMOS33 } [get_ports { an[0] }];
set_property -dict { PACKAGE_PIN B16   IOSTANDARD LVCMOS33 } [get_ports { an[1] }];
set_property -dict { PACKAGE_PIN A18   IOSTANDARD LVCMOS33 } [get_ports { an[2] }];

set_property -dict { PACKAGE_PIN B18   IOSTANDARD LVCMOS33 } [get_ports { BTN }];

set_property -dict { PACKAGE_PIN D14   IOSTANDARD LVCMOS33 } [get_ports { SW }];

set_property -dict { PACKAGE_PIN F16   IOSTANDARD LVCMOS33 } [get_ports { rst }];
```

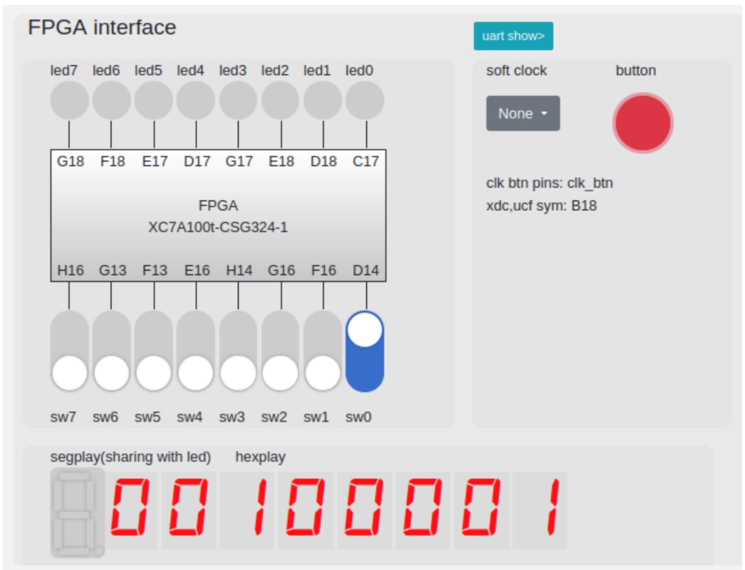
> 设计文件;

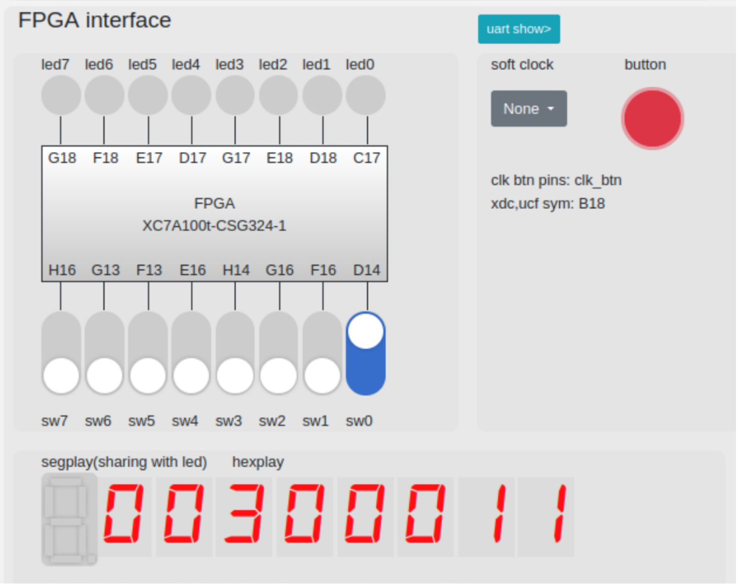
```
module T4(
input SW,
input CLK,
input BTN,
input rst,
output reg [3:0] d,
output reg [2:0] an);

parameter [2:0] s0=3'b000,s1=3'b001, s2=3'b010;
parameter [2:0] s3=3'b011,s4=3'b100,s5=3'b101,s6=3'b110,s7=3'b111;
reg [2:0] state,next_state,l_state;
```

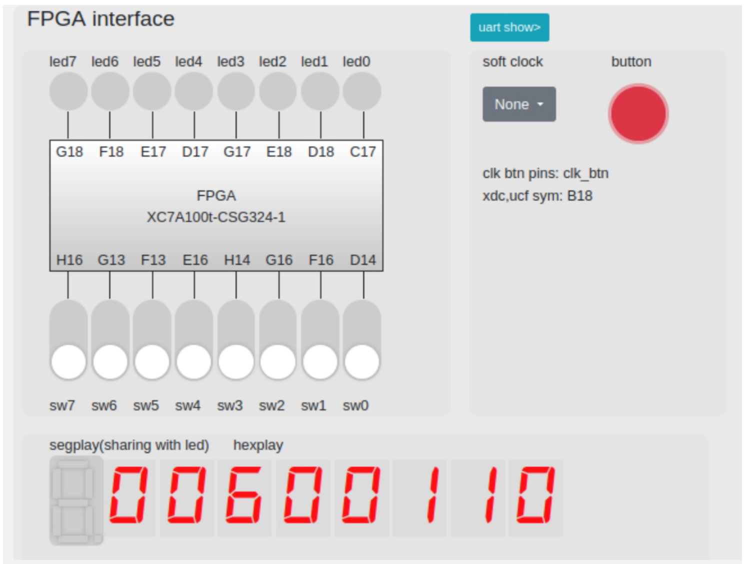
```verilog
reg [3:0] counter=4'b0000;
reg [6:0] cnt;
wire BTNt,BTN_edge;

jitter_clr c1(CLK,BTN,BTNt);
signal_edge e1(CLK,BTNt,BTN_edge);

always@(posedge CLK)
begin
    if(rst)
        begin
        l_state <= 0;
        state <= s0;
        counter<=0;
    end
    else if(BTN_edge)
        begin
            l_state<=state;
            state <=next_state;
            if(next_state==s4)
                if(state == s6)
                    counter=counter+4'b0001;
        end
end

always@(*) begin
case(state)
s0:
    begin
        if(!SW)//sw=0,000->000,s0->s0
            begin
            if(BTN_edge) next_state = s0;
            end
        else //sw=1,000->001,s0->s1
            begin
            if(BTN_edge) next_state = s1;
            end
    end
s1:
    begin
        if(!SW) //001->010
            begin
            if(BTN_edge)
                next_state = s2;
            end
        else
            begin //001->011
                if(BTN_edge) next_state = s3;
            end
    end
s2:
    begin
        if(!SW)
            begin//010->100
                if(BTN_edge)
                next_state = s4;
            end
        else
            begin //010->101
                if(BTN_edge) next_state = s5;
            end
    end
s3:
    begin
        if(!SW)//011->110
            begin
            if(BTN_edge) next_state = s6;
            end
        else
            begin //011->111
                if(BTN_edge) next_state = s7;
            end
    end
s4:
    begin
        if(!SW) //100->000
            begin
            if(BTN_edge) next_state = s0;
            end
        else //100->001
            begin
                if(BTN_edge) next_state = s1;
            end
    end
s5:
    begin
        if(!SW) //101->010
            begin
            if(BTN_edge)
                next_state=s2;
            end
        else
            begin //101->011
                if(BTN_edge) next_state =s3;
            end
    end
s6:
    begin
        if(!SW)
            begin //110->100
            if(BTN_edge)
                next_state=s4;
            end
        else //110->101
            begin
                if(BTN_edge) next_state=s5;
            end
    end
s7:
    begin
```

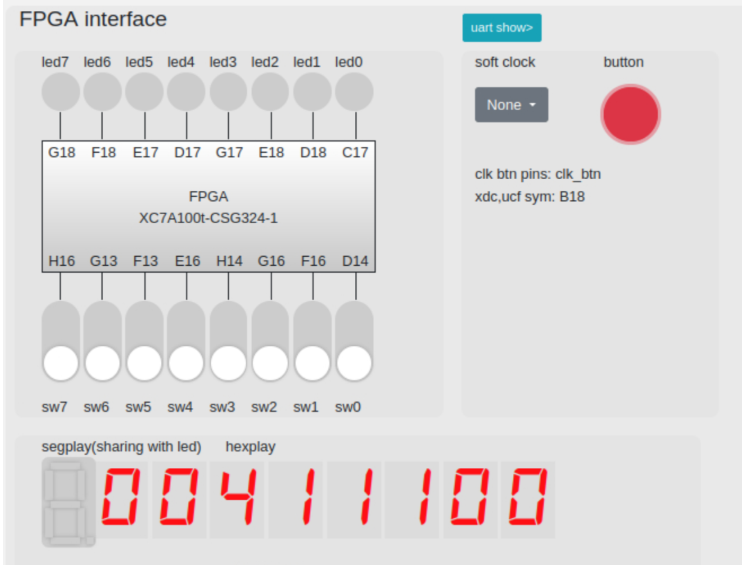
```
        if(!SW) //111->110
            begin
            if(BTN_edge) next_state = s6;
            end
        else
            begin //111->111
                if(BTN_edge) next_state=s7;
            end
    end
default:
    begin
        if(!SW)
            begin
            if(BTN_edge) next_state = s2;
            end
        else
            begin
                if(BTN_edge) next_state=s1;
            end
    end
endcase
end

always@(posedge CLK)
begin
cnt=cnt+1;
an<=cnt[6:4];
    if(an==3'b000) d<={3'b000,state[0]};
    if(an==3'b001) d<={3'b000,state[1]};
    if(an==3'b010) d<={3'b000,state[2]};
    if(an==3'b011) d<={3'b000,l_state[2]};
    if(an==3'b100) d<=counter;
    if(an==3'b101) d<={1'b0, state};
    if(an==3'b110) d<=0;
    if(an==3'b111) d<=0;
end
endmodule
```

> 实验结果显示
>
> 代码共有8个状态，后四位表示当前出现的序列，序列的后三位的二进制数转化为10进制后，对应的是相应的状态，比如后三位是011代表状态s3


FPGA interface
uart show>
led7 led6 led5 led4 led3 led2 led1 led0
G18 F18 E17 D17 G17 E18 D18 C17
FPGA
XC7A100t-CSG324-1
H16 G13 F13 E16 H14 G16 F16 D14
sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0
soft clock
button
None
clk btn pins: clk_btn
xdc,ucf sym: B18
segplay(sharing with led)
hexplay
00100001


FPGA interface
uart show>
led7 led6 led5 led4 led3 led2 led1 led0
G18 F18 E17 D17 G17 E18 D18 C17
FPGA
XC7A100t-CSG324-1
H16 G13 F13 E16 H14 G16 F16 D14
sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0
soft clock
button
None
clk btn pins: clk_btn
xdc,ucf sym: B18
segplay(sharing with led)
hexplay
00300011


FPGA interface
uart show>
led7 led6 led5 led4 led3 led2 led1 led0
G18 F18 E17 D17 G17 E18 D18 C17
FPGA
XC7A100t-CSG324-1
H16 G13 F13 E16 H14 G16 F16 D14
sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0
soft clock
button
None
clk btn pins: clk_btn
xdc,ucf sym: B18
segplay(sharing with led)
hexplay
00600110


FPGA interface
uart show>
led7 led6 led5 led4 led3 led2 led1 led0
G18 F18 E17 D17 G17 E18 D18 C17
FPGA
XC7A100t-CSG324-1
H16 G13 F13 E16 H14 G16 F16 D14
sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0
soft clock
button
None
clk btn pins: clk_btn
xdc,ucf sym: B18
segplay(sharing with led)
hexplay
00411100

> ## 总结
>
> 难度：本次实验难度较大
>
> 任务量：大
>
> 建议：手册可以再完善亿点点吗
>
> 收获：加深了对有限状态机的掌握和了解，更熟练掌握Vivado和Logisim