**实验报告 Lab 4**

**一、按钮测试结果**

1、防止亚稳态

缺少该模块可能导致亚稳态，输出不稳定：触发器的输出可能会在0和1之间振荡，或者在很长时间内保持在一个中间电平，可能导致后续逻辑电路接收到错误的值，进而导致系统行为异常或崩溃。

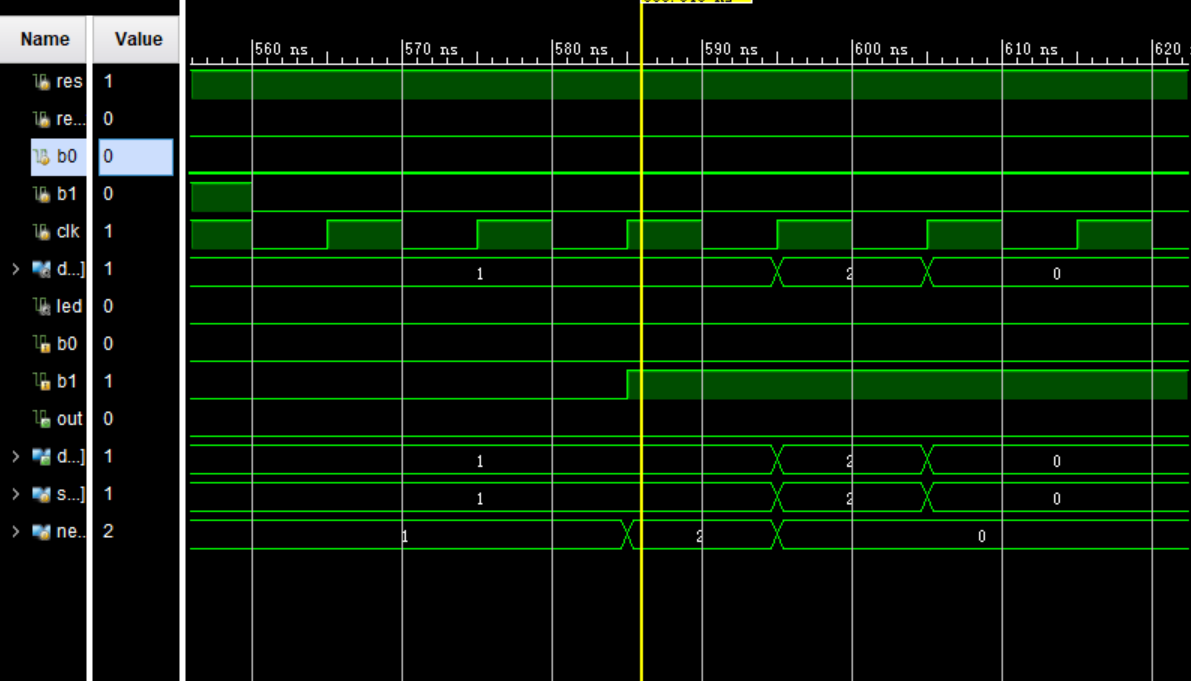
2、消抖模块

按钮在按下和释放时会产生机械抖动，导致多个上升沿和下降沿。若没有模块处理抖动，可能会导致输出状态在短时间内多次切换。

3、脉宽变换模块：

|  |
| --- |
|  |
| 无模块 |
|  |
| 有模块 |

脉宽变换模块负责将按钮按下的脉冲控制在一个时钟周期，避免后续导入FSM时出现意外的状态转换。

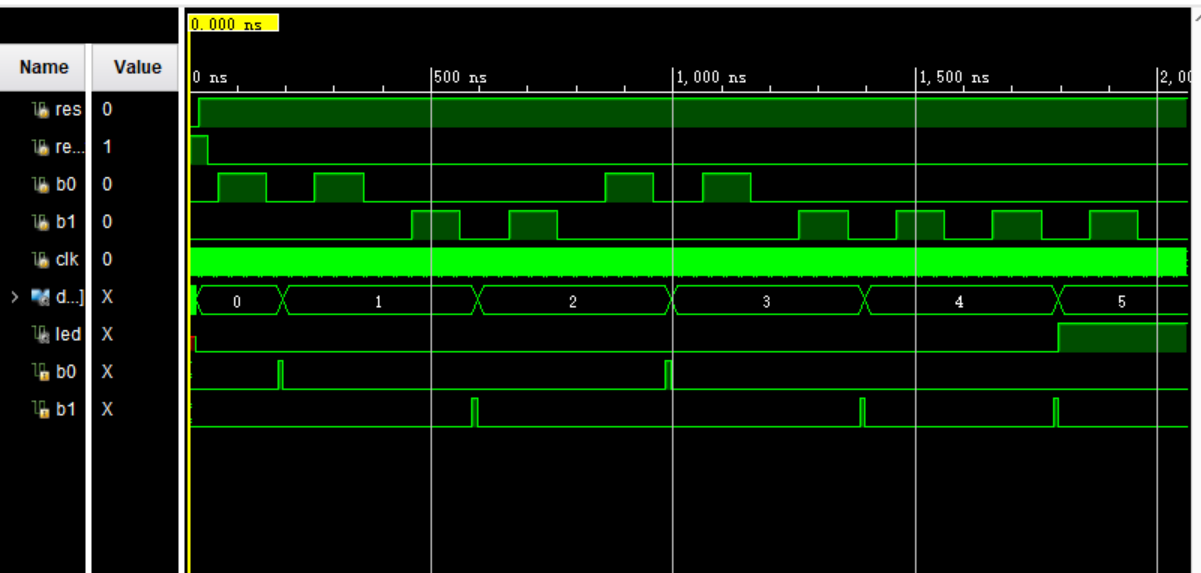


如图，在b1由0转1时，next\_state由1转2，如果不把开关脉冲控制在一个周期内，就会误认为按钮一直被按下，从而发生意想不到的状态转变，如图中next\_state从2意外跳变到0。

**二、行为级代码**

代码（见附录）

仿真（见下图）

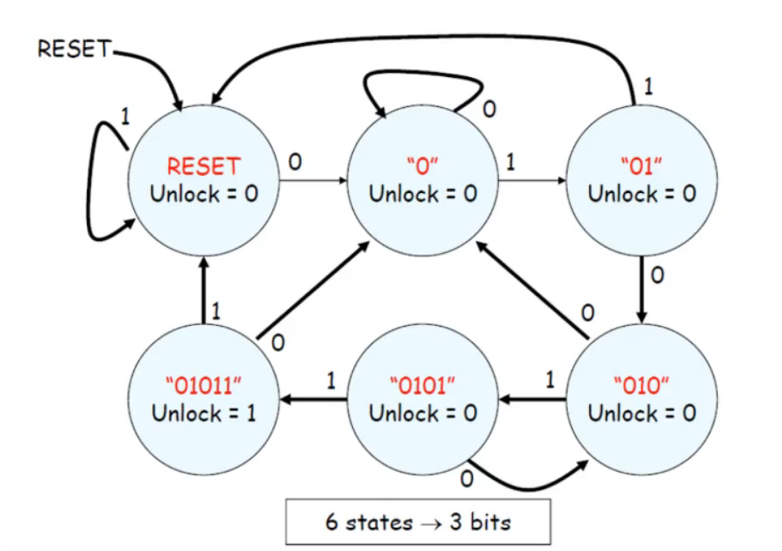


FPGA板上结果（见下图）

|  |  |
| --- | --- |
|  |  |
| 1）初始/按下RESET | 2）按下“0” |
|  |  |
| 3）按下“1” | 4）按下“0” |
|  |  |
| 5）按下“1” | 6）按下“1” |

**三、状态图 状态方程 驱动方程 状态机电路**

手写状态图（见下图）：



由状态图可列出如下的状态转换表：

|  |  |  |
| --- | --- | --- |
| 现态S2S1S0 | 次态S2nextS1nextS0next | |
| Y=0 | Y=1 |
| (RESET) 000 | 001 | 000 |
| (0) 001 | 001 | 010 |
| (01) 010 | 011 | 000 |
| (010) 011 | 001 | 100 |
| (0101) 100 | 011 | 101 |
| (01011) 101 | 001 | 000 |

由以上状态转换表可推导出如下的状态方程和驱动方程：

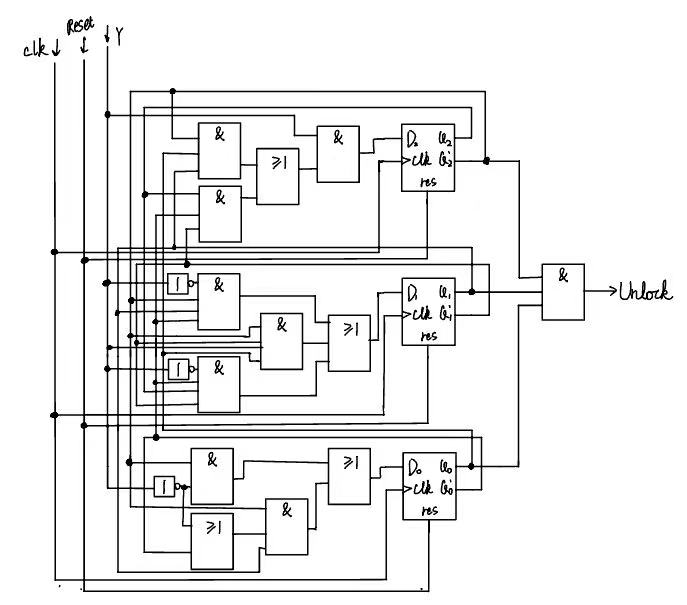
S2next= S2’S1S0Y + S2S1’S0’Y

S1next= S2’S1’S0Y+ S2’S1S0’Y’ + S2S1’S0’Y’

S0next= S2’Y’+ S2’S1 (Y’ + S0’)

Unlock= S2S1’S0

状态机电路图如下：



**四、电路级代码**

电路级代码（见附录）

**附录 代码**

1 行为级代码

**module lock(**

**input reset,**

**input b0,**

**input b1,**

**input clk,**

**output an,**

**output reg a,**

**output reg b,**

**output reg c,**

**output reg d,**

**output reg e,**

**output reg f,**

**output reg g,**

**output led**

**);**

**wire b0\_out;**

**wire b1\_out;**

**onoff button\_0(.clk(clk),.reset1(1),.button\_in(b0),.out(b0\_out));**

**onoff button\_1(.clk(clk),.reset1(1),.button\_in(b1),.out(b1\_out));**

**wire[3:0]display;**

**FSM fsm1(.clk(clk),.reset(reset),.b0(b0\_out),.b1(b1\_out),.out(led),.display(display)**

**);**

**assign an = 1;**

**always @(\*) begin**

**case(display)**

**3'b000 : {a,b,c,d,e,f,g}=7'b1111110;**

**3'b001 : {a,b,c,d,e,f,g}=7'b0110000;**

**3'b010 : {a,b,c,d,e,f,g}=7'b1101101;**

**3'b011 : {a,b,c,d,e,f,g}=7'b1111001;**

**3'b100 : {a,b,c,d,e,f,g}=7'b0110011;**

**3'b101 : {a,b,c,d,e,f,g}=7'b1011011;**

**default : {a,b,c,d,e,f,g}=7'b1111110;**

**endcase**

**end**

**endmodule**

**module FSM(**

**input clk,reset,b0,b1,**

**output out,**

**output [3:0] display**

**);**

**parameter S\_RESET = 0; parameter S\_0 = 1;**

**parameter S\_01 = 2; parameter S\_010 = 3;**

**parameter S\_0101 = 4; parameter S\_01011 = 5;**

**reg [2:0] state, next\_state;**

**always@(\*) begin**

**if(reset==0) next\_state = S\_RESET;**

**else case (state)**

**S\_RESET: next\_state = b0 ? S\_0 :b1 ? S\_RESET:state;**

**S\_0: next\_state = b0 ? S\_0 :b1 ? S\_01:state;**

**S\_01: next\_state = b0 ? S\_010 :b1 ? S\_RESET:state;**

**S\_010: next\_state = b0 ? S\_0 :b1 ? S\_0101:state;**

**S\_0101: next\_state = b0 ? S\_010 :b1 ? S\_01011:state;**

**S\_01011: next\_state = b0 ? S\_0 :b1 ? S\_RESET:state;**

**default: next\_state = S\_RESET;**

**endcase**

**end**

**always@(posedge clk) state<= next\_state;**

**assign out = (state == S\_01011);**

**assign display = {1'b0,state};**

**endmodule**

**module display(**

**input [2:0]din,**

**output an,**

**output reg a,**

**output reg b,**

**output reg c,**

**output reg d,**

**output reg e,**

**output reg f,**

**output reg g**

**);**

**assign an = 1;**

**always @(\*) begin**

**case(din)**

**3'b000 : {a,b,c,d,e,f,g}=7'b1111110;**

**3'b001 : {a,b,c,d,e,f,g}=7'b0110000;**

**3'b010 : {a,b,c,d,e,f,g}=7'b1101101;**

**3'b011 : {a,b,c,d,e,f,g}=7'b1111001;**

**3'b100 : {a,b,c,d,e,f,g}=7'b0110011;**

**3'b101 : {a,b,c,d,e,f,g}=7'b1011011;**

**default : {a,b,c,d,e,f,g}=7'b1111110;**

**endcase**

**end**

**endmodule**

**module onoff(input clk, reset1, button\_in, output wire out);**

**wire reset;**

**reg light\_reg;**

**assign reset = ~reset1;**

**//synchronizer**

**reg button,btemp;**

**always @(posedge clk)**

**{button,btemp} <= {btemp,button\_in};**

**//debounce push button**

**wire bpressed;**

**debounce db1(.clk(clk),.reset(reset),.noisy(button),.clean(bpressed));**

**reg old\_bpressed; //state last clk cycle**

**reg out0;**

**always @(posedge clk) begin**

**if (reset)**

**begin out0 <= 0; old\_bpressed <=0;end**

**else if (old\_bpressed==0 && bpressed==1)begin**

**//button changed from 0 to 1**

**out0 <= ~out0;**

**old\_bpressed <= bpressed;**

**end**

**else if (old\_bpressed==1 && bpressed==0)begin**

**//button changed from 1 to 0**

**out0 <= ~out0;**

**old\_bpressed <= bpressed;**

**end**

**end**

**pwm pwm1(.clk(clk),.in(out0),.out(out));**

**endmodule**

**module debounce (**

**input clk,**

**input reset,**

**input noisy,**

**output reg clean**

**);**

**parameter COUNTER\_WIDTH = 20;**

**parameter MAX\_COUNT = 650000;**

**reg [COUNTER\_WIDTH-1:0] count;**

**reg noisy\_reg;**

**ys @(posedge clk or posedge reset) begin**

**if (reset) begin**

**count <= 0;**

**noisy\_reg <= 0;**

**clean <= 0;**

**end else begin**

**noisy\_reg <= noisy;**

**if (noisy\_reg != noisy) begin**

**count <= 0;**

**end else if (count < MAX\_COUNT) begin**

**count <= count + 1;**

**end else begin**

**clean <= noisy\_reg;**

**end**

**end**

**end**

**endmodule**

**module pwm(**

**input wire in,**

**input wire clk,**

**output wire out**

**);**

**reg q;**

**always @(posedge clk) begin**

**q <= in;**

**end**

**assign out = in & ~q;**

**endmodule**

2 电路级代码

**module FSM(**

**input clk,reset,b0,b1,**

**output out,**

**output [3:0] display**

**);**

**parameter S\_RESET = 3'b000;**

**parameter S\_0 = 3'b001;**

**parameter S\_01 = 3'b010;**

**parameter S\_010 = 3'b011;**

**parameter S\_0101 = 3'b100;**

**parameter S\_01011 = 3'b101;**

**reg [2:0] state;**

**wire [2:0] next\_state;**

**assign next\_state[2] = (b0 == 0 && b1 == 0) ? state[2] : ((~state[2] & state[1] & state[0]) | (state[2] & ~state[1] & ~state[0])) & ~b0 & b1 & ~reset;**

**assign next\_state[1] = (b0 == 0 && b1 == 0) ? state[1] : ((~state[2] & ~state[1] & state[0] & ~b0 & b1) |**

**(state[2] & ~state[1] & ~state[0] & b0 & ~b1) |**

**(~state[2] & state[1] & ~state[0] & b0 & ~b1)) & ~reset;**

**assign next\_state[0] = (b0 == 0 && b1 == 0) ? state[0] : ((state[2] & ~state[1] & (~state[0] | b0 & ~b1)) |**

**(~state[2] & b0 & ~b1)) & ~reset;**

**always @(posedge clk or posedge reset) begin**

**if (reset) state <= S\_RESET;**

**else state <= next\_state;**

**end**

**assign out = state[2] & ~state[1] & state[0];**

**assign display = {1'b0,state};**

**endmodule**