Lab4\_有限状态机

金泽文 pb15111604

实验目的：

* 进一步学习时序逻辑电路
* 了解有限状态机的工作原理
* 学会使用“三段式”有限状态机设计电路
* 掌握按键去抖动、信号取边沿等处理技巧

实验内容：

* 用三段式有限状态机实现序列检测功能电路
  + 按从高位到低位逐位串行输入一个序列
  + 每当检测到序列“1101”（不重叠）时，LED指示灯亮，否则灭，例如
    - 输入： 1 1 0 1 1 0 1 1 0 1
    - 输出： 0 0 0 1 0 0 0 0 0 1
  + 用拨动开关输入检测序列
  + 按键按下的瞬间将拨动开关状态锁存
    - 注意防抖动（按键按下瞬间可能会有多次的电平跳变）

具体实现：

用FSM模块实现状态机，共5个状态。

用key\_jitter模块实现去抖动功能，(以保持10ms以上为有效)

用Output实现数码管显示

实验结果：

成功实现。

实验分析：

再次因为语法不清楚，以及阻塞与非阻塞赋值之间的不清楚浪费 了很长的时间，下次不能因为这些基本概念的混淆而浪费实现 了。

意见建议：

暂无

附录：

代码：

顶层模块

`timescale 1ns / 1ps

module top(

input clk,

input rst\_n,

input btn,

input In,

output led,

output [3:0] sel,

output [7:0] data

);

wire Op;

wire out;

key\_jitter u\_key\_jitter(

.clk (clk),

.rst\_n (rst\_n),

.btn (btn),

.Op (Op)

);

FSM u\_FSM(

.clk (clk),

.rst\_n (rst\_n),

.In (In),

.Op (Op),

.led (led)

);

Output u\_Output(

.clk (clk),

.rst\_n (rst\_n),

.Op (Op),

.In (In),

.sel (sel),

.data (data)

);

endmodule

消除抖动模块

`timescale 1ns / 1ps

module key\_jitter(

input clk,

input rst\_n,

input btn,

output reg Op

);

reg Ol;

reg [18:0] cnt;

initial Ol = 0;

always@(posedge clk or negedge rst\_n)

begin

if(~rst\_n)

cnt <= 19'h0;

else

begin

if(Ol == btn)

cnt <= 19'h0;

else

cnt <= cnt + 19'h1;

if(cnt == 19'h511\_999)

begin

Ol = btn;

cnt <= 19'h0;

end

end

end

always@(posedge clk)

begin

if((cnt == 19'h511\_999) && (Ol == 1))

Op <= 1;

if(Op == 1)

Op <= 0;

end

endmodule、

状态机模块

`timescale 1ns / 1ps

module FSM(

input clk,

input rst\_n,

input In,

input Op,

output reg led

);

parameter S0 = 0, S1 = 1, S2 = 2, S3 = 3, S4 = 4;

reg [2:0] next\_state;

reg [2:0] curr\_state;

initial next\_state = S0;

always@(posedge clk or negedge rst\_n)

begin

if(~rst\_n)

curr\_state <= S0;

else if(Op)

curr\_state <= next\_state;

end

always@(\*) begin

case(curr\_state)

S0: if(In) next\_state <= S1; else next\_state <= S0;

S1: if(In) next\_state <= S2; else next\_state <= S0;

S2: if(In) next\_state <= S2; else next\_state <= S3;

S3: if(In) next\_state <= S4; else next\_state <= S0;

S4: if(In) next\_state <= S1; else next\_state <= S0;

default: next\_state <= S0;

endcase

end

always@(posedge clk or negedge rst\_n) begin

if(~rst\_n)

led <= 0;

else if(Op) begin

case(curr\_state)

S0: if(In) led <= 0; else led <= 0;

S1: if(In) led <= 0; else led <= 0;

S2: if(In) led <= 0; else led <= 0;

S3: if(In) led <= 1; else led <= 0;

S4: if(In) led <= 0; else led <= 0;

default: led <= 0;

endcase

end

end

endmodule

数码管显示模块

module Output(

input clk,

input rst\_n,

input Op,

input In,

output reg [3:0] sel,

output reg [7:0] data

);

reg [16:0] sel\_ctrl;

reg [7:0] data1;

reg [7:0] data2;

reg [7:0] data3;

reg [7:0] data4;

initial sel\_ctrl = 17'h0;

initial sel = 4'b1110;

initial data = 8'b1111\_1111;

initial data1 = 8'b1111\_1111;

initial data2 = 8'b1111\_1111;

initial data3 = 8'b1111\_1111;

initial data4 = 8'b1111\_1111;

always@(posedge clk)

begin

if(sel\_ctrl == 17'd100\_000)

sel\_ctrl <= 17'd0;

else

sel\_ctrl <= sel\_ctrl + 17'h1;

end

always@(posedge clk)

begin

if(sel\_ctrl == 17'd25000)

sel <= 4'b1110;

else if(sel\_ctrl == 17'd50000)

sel <= 4'b1101;

else if(sel\_ctrl == 17'd75000)

sel <= 4'b1011;

else if(sel\_ctrl == 17'd100\_000)

sel <= 4'b0111;

end

always@(posedge clk or negedge rst\_n)

begin

if(~rst\_n)

begin

data1 <= 8'b1111\_1111;

data2 <= 8'b1111\_1111;

data3 <= 8'b1111\_1111;

data4 <= 8'b1111\_1111;

end

else if(Op)

begin

data4 <= data3;

data3 <= data2;

data2 <= data1;

if(In)

data1 <= 8'b1001\_1111;

else

data1 <= 8'b0000\_0011;

end

end

always@(\*)

begin

if(sel == 4'b1110)

data = data1;

else if(sel == 4'b1101)

data = data2;

else if(sel == 4'b1011)

data = data3;

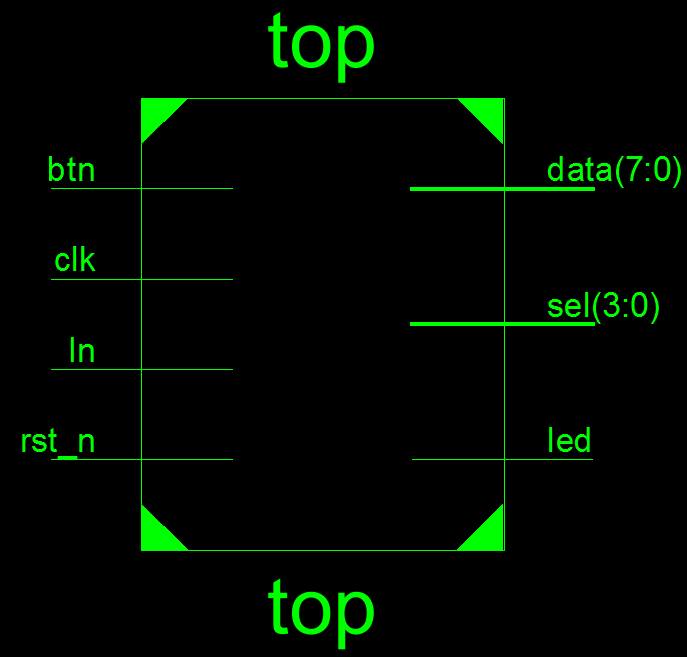
else

data = data4;

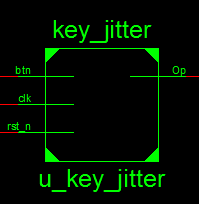
end

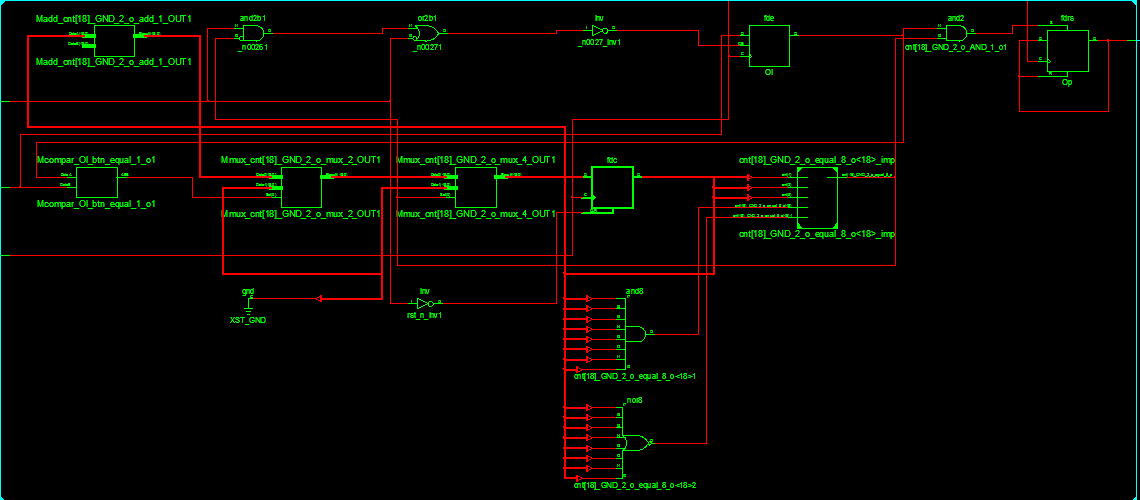
endmodule

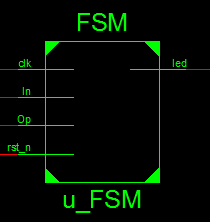
原理图：

top模块

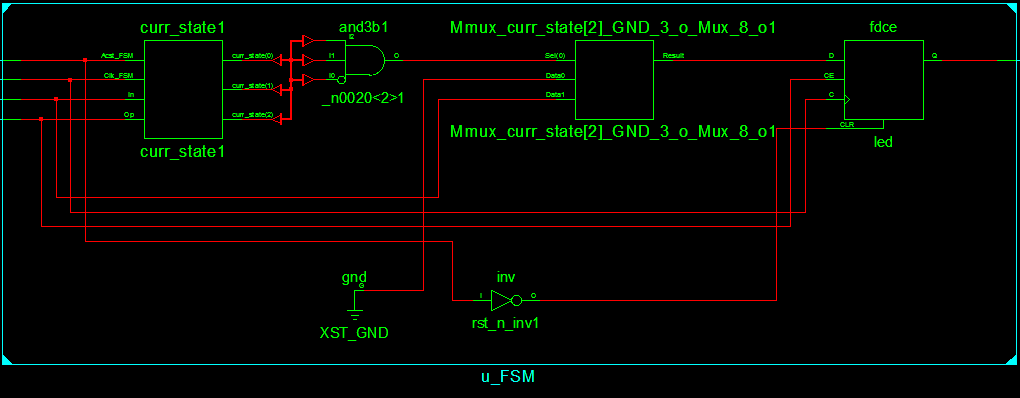
key\_jitter模块

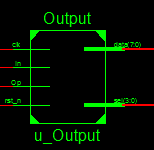






FSM模块





Output模块

