北京邮电大学



实 验 报 告

实验名称： 简易地铁自动售票系统

学 院： 信息与通信工程学院

班 级： 2017211128

姓 名：周磊

学 号： 2017210494

班内序号： 21

日 期： 2018-12-26

目录

一、设计课题的任务要求 ——————————3

二、系统设计（包括设计思路、总体框图、分块设计）—3

三、仿真波形及波形分析 ——————————11

四、代码 —————————18

五、功能说明及资源利用情况 —————————55

六、故障及问题分析 —————————56

七、总结和结论 —————————56

一、设计课题的任务要求

1.1任务要求

1、 用SW7作为售票机开关，打开开关SW7后售票机进行自检：8\*8点阵和数码管DISP7~DISP0全亮0.5S熄灭0.5S重复三次，进入待机状态；

2、 地铁票价5站（含）内3元；6~10站（含）4元；11~15站（含）5元；16~20站（含）6元；21站（含）以上部分，每增加1元可乘坐10站；

3、 通过按键BTN0输入购票张数，按一下加1，一次最多购买9张，加到9后返回1，并用数码管DISP0显示当前购票张数；

4、 通过按键BTN2、BTN1输入乘坐站数，2个按键分别对应乘坐站数的十位和个位，按一下加1，加到9后返回0，并用数码管DISP3和DISP2显示乘坐站数；

5、 用按键BTN5、BTN4、BTN3表示投入10元、5元和1元三种钱币，每个按键按一次表示该币种的钱币张数加1，并用数码管DISP7、DISP6和DISP5显示当前投入的总钱数；

6、 以上输入完成后，按BTN6表示确定购票，根据输入情况出票、找零或者提示错误，同时在点阵上显示出票、取找零或输入错误等提示图形；

7、 一次购票成功后系统能够恰当地转入下一次购票等待状态；

8、 系统能够复位。

提高要求：

1、 点阵上的各种提示信息用动画表示；

2、 不同的提示配合不同的音效；

3、 自拟其他功能。

1.2个人已完成功能

基本功能都已实现，提高功能实现了点阵动画显示，以及不同提示配合不同音效的功能。

二、系统设计

2.1设计思路

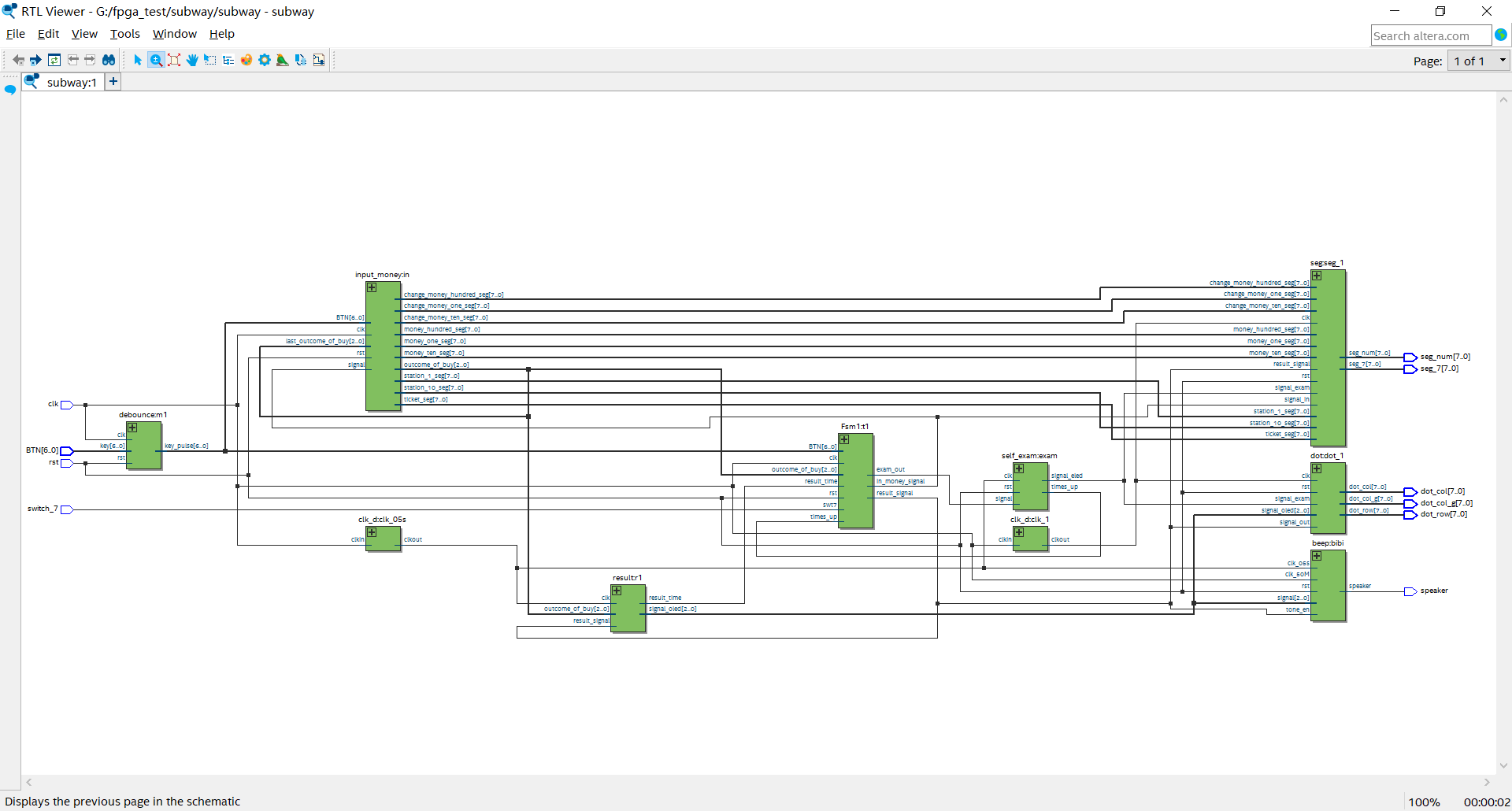
系统的核心为有限状态机，将售票系统的工作状态分为关机（state\_off）,自检(state\_exam),待机(state\_wait)，输入(state\_in),和输出(state\_out)五个状态，并将他们的状态转移条件写入到状态机的敏感列表，综合成有限状态机文件。

除了状态机模块，还有其他的子模块，如输入模块(input\_money)，自检模块（self\_exam）,结果模块(result),数码管模块(seg),点阵模块(dot)，蜂鸣器模块(beep),以及一些分频器模块(clk\_d)，消抖模块（debounce）。

首先，状态机会根据当前的状态产生各个子模块的使能信号，使子模块进行工作。而模块工作的结果（如购票的结果）或者一些时间延时结果（如自检模块工作三秒后停止，状态进入待机状态），会产生一些反馈信号，输入到状态机模块，来促使状态的转移。

2.2总体框图



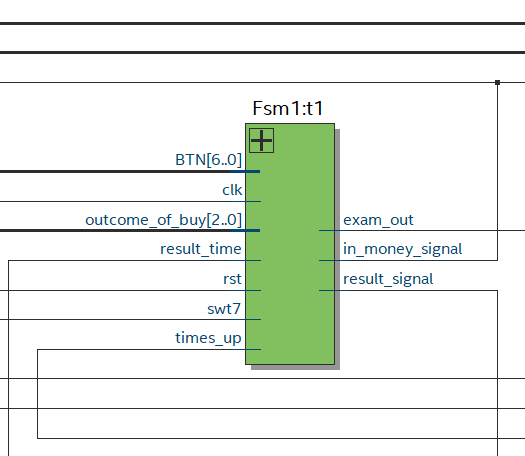




2.3分块设计

**1．状态机模块**

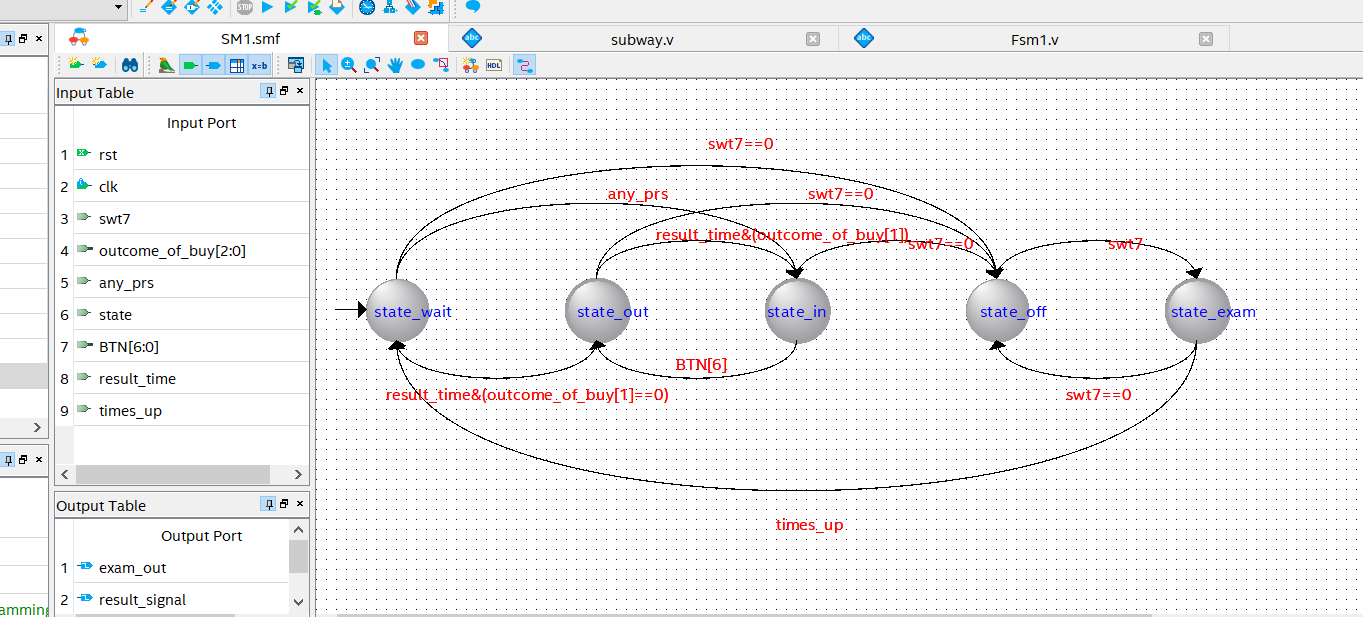
三段式，将外部输入信号与顶层模块内部反馈信号作为输入，进行状态转移，而输出信号为对其他模块的使能信号。



输入BTN[6:0]为消抖后按键,clk为50MHz系统时钟，outcome\_of\_buy为上次购票的结果（根据上次购票的结果来确定是进入待机状态，还是回到输入状态），result\_time为结果输出的时间结束（及点阵动画和提示音均结束），rst(复位)，swt7为拨码开关7,times\_up为自检3s时间结束，exam\_out为自检模块的使能信号，in\_money\_signal为输入模块的使能信号，result\_signal为结果的使能信号。

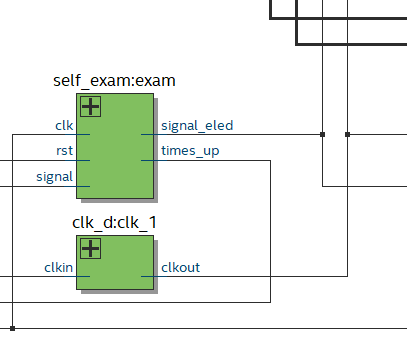
其中outcome\_of\_buy和result\_time以及times\_up为其他模块的反馈信号。

状态机模块中，预置state为关机状态(state\_off)。



BTN[6]为确定按键，any\_prs是检测除复位键以外是否有按键按下（按下为高电平），outcome\_of\_buy[2:0]为上次购票结果，其中010为购票失败，100购票成功且不需找零，101为购票成功需要找零，若购票失败则再次回到输入状态,成功则进入待机状态。

**2．自检模块**

****

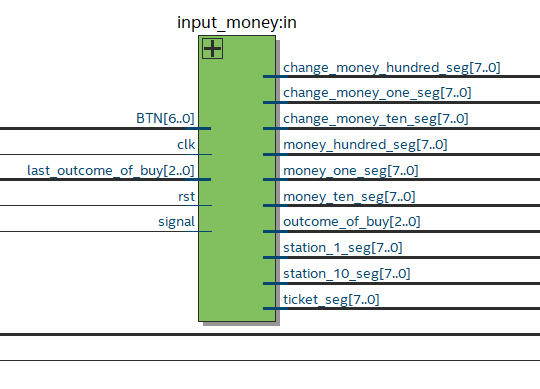
clk为2Hz时钟，rst复位，signal为exam\_out（使能信号），signal\_eled为数码管和点阵全亮状态的使能信号，times\_up为3s结束的反馈信号。

在self\_exam模块中有计数器，根据计数器的值来控制signal\_eled和times\_up的值。

**3．待机状态**

待机状态没有模块，但因为待机状态其他模块的使能信号全为0，故都不工作。

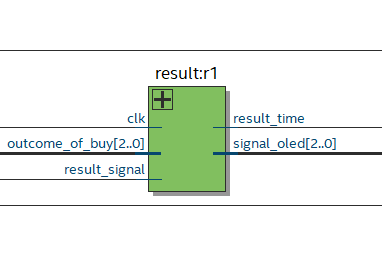
**4．输入模块**

****

BTN[6:0]位消抖后按键，clk为50MHz时钟，last\_outcome\_of\_buy[2:0]为上次购买结果（其中010为购票失败，100购票成功且不需找零，101为购票成功需要找零），rst为复位，signal为使能信号(对应in\_money\_signal)，输出为chang\_money\_hundred\_seg[7:0]……凡是结尾为seg均为对应数码管的八段信号，例如chang\_money\_ten为找零的钱的十位对应的数码管信号。而outcome\_of\_buy[2:0]为本次购买结果。

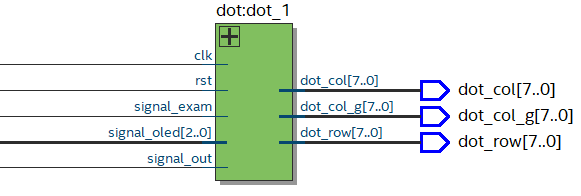
模块中，值得一提的是采用了加三移位法模块（find\_wei）来分离开输入钱数的百十个位，来减少资源占用。

**5.输出模块(结果模块)**

****

clk为2Hz时钟，outcome\_of\_buy[2:0]为本次购买结果，result\_signal为输出状态的使能信号，result\_time为结果结束的反馈信号，signal\_oled[2:0]是控制点阵和蜂鸣器的不同工作状态（因为不同提示音和不同动画相匹配，故可使用相同信号）。

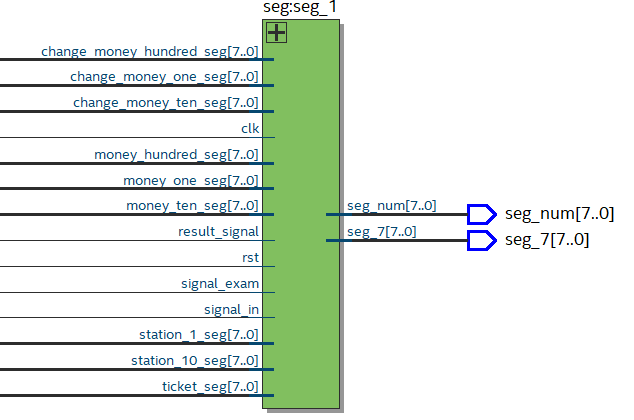
**6.点阵模块**

****

Clk为50MHz时钟，rst复位，signal\_exam为自检状态的使能信号，signal\_oled[2:0]为输出结果的三个状态间的选择信号，signal\_out为输出结果的使能信号。Dot\_col[7:0]为红色数码管，dot\_col\_g为绿色数码管，dot\_row[7:0]为数码管的行。

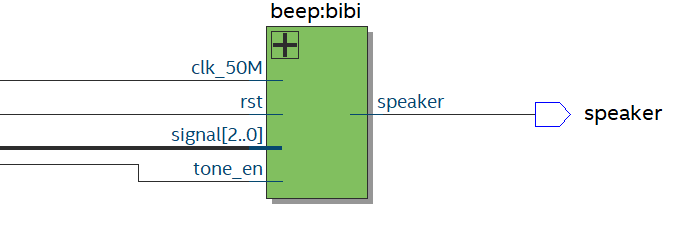
其中已经写了输出结果的动画显示，用频率小的计数器来控制图形的选择，用频率大的计数器来控制图形的扫描显示。

**7.数码管模块**

****

左侧的结果带seg均为译码后的信号，result\_signal为输出结果的使能信号（输出结果包括错误信号HHH，找零钱数），signal\_exam为自检状态的使能信号，signal\_in为输入状态的使能信号,seg\_7为8段数码管的abcdefg,dp,seg\_num为数码管的位。模块中因为各个使能信号同时只有一个为1，故使用case选择即可实现，不同状态输出不同图形的效果。

**8.蜂鸣器模块**



Clk\_05s为2Hz时钟，clk\_50Mhz时钟，rst复位，signal为根据购票结果的控制信号（这里用的仍是signal\_oled以使之与点阵动画匹配），tone\_en为使能信号（这里为resul\_signal）。

模块中，主要仍是运用分频器的思想，来使50Mhz分成各种音的频率，模块中还分了5Hz的基频来控制音符的转移。

**9.分频器及按键消抖等其他模块**

包括分频器，按键消抖，移位加3分离数字的位数等模块。

1.分频器是定义内部有计数器MAX，通过例化分频器后，用defparam来改变MAX来达到改变频率的目的。

2.按键消抖是相当于用内部计数到20ms后检测到按键仍为高电平，则确定按下一次。内部有parammeter N,表示需要消抖按键的数量，也可以例化后，通过外部defparam来改变。

3.BCD移位比较模块：

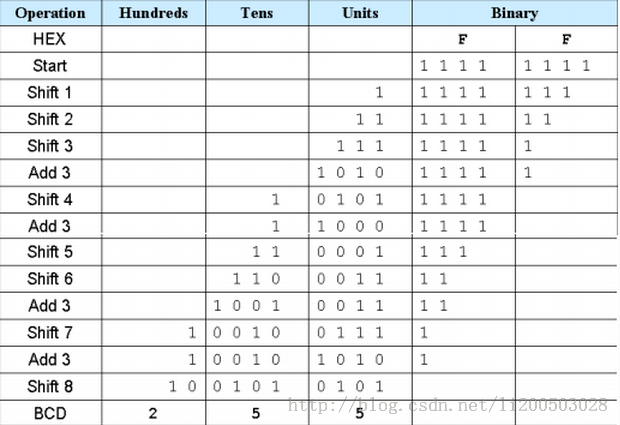
内部步骤: 1.将二进制码左移一位（或者乘2）。

               2.找到左移后的码所对应的个，十，百位。

               3.判断在个位，十位和百位的码是否大于（？等于）5，如果是则该段码加3。

               4.继续重复以上三步直到移位8次后停止。

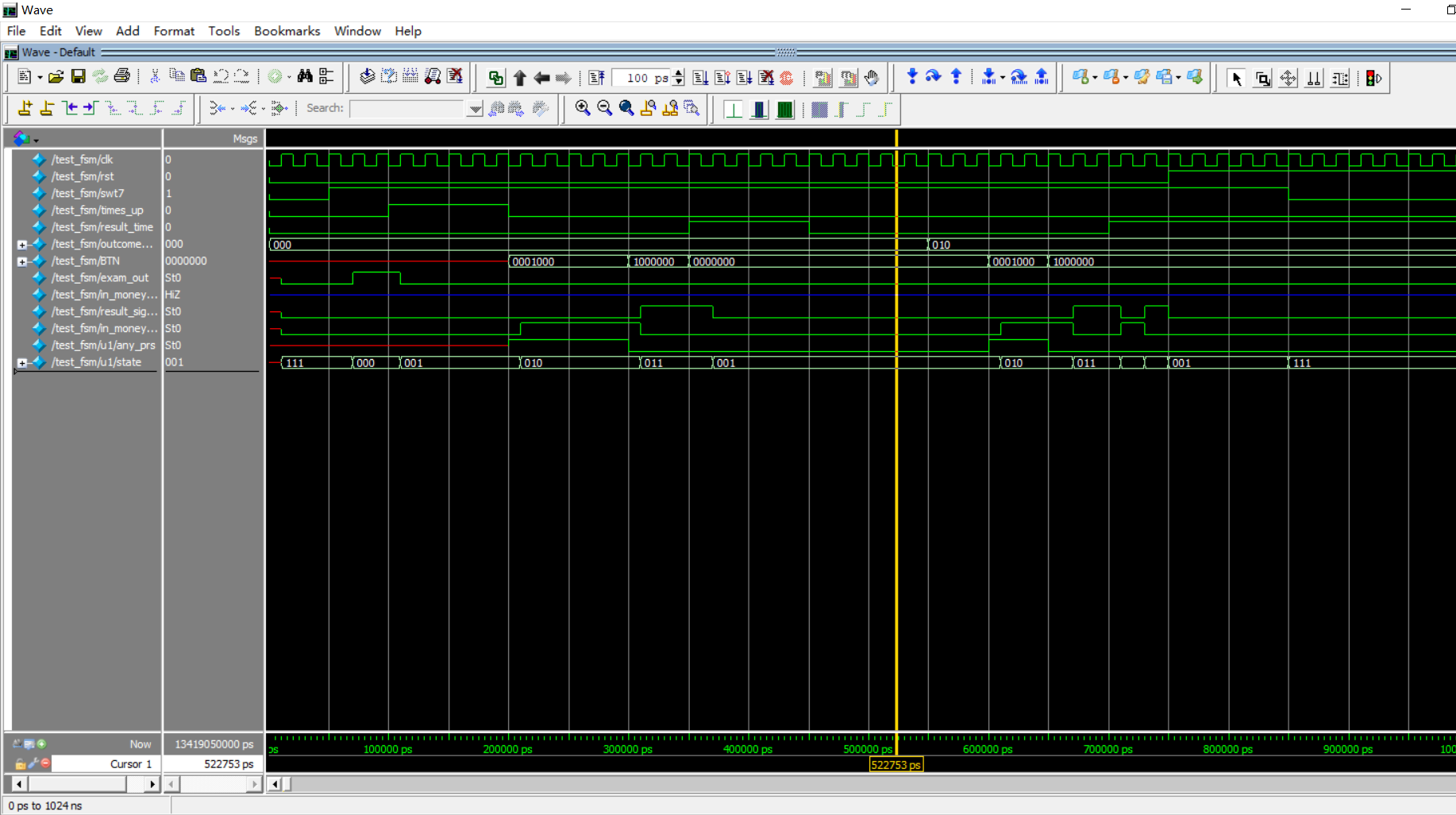
下面是一个例子 ，将 1111\_1111 转换为 BCD码 ，如果8bit数据最终移位得到18bit 数据 ，那么个位，十位，百位分别对应12~9，16~13，18~17位。



三、仿真波形及波形分析

因为整个顶端模块过于巨大，不便于整体仿真，故分块仿真。

**1.状态机仿真**

****

根据波形显示，swt7变为高后，state从111（关机）变为000（自检），并且exam\_out使能信号变为高，其他为低；紧接着times\_up变为高后，state变为001（待机）；再接着按钮变为0000100，any\_prs变为1，同时state变为010（输入），in\_money\_signal使能信号为1；

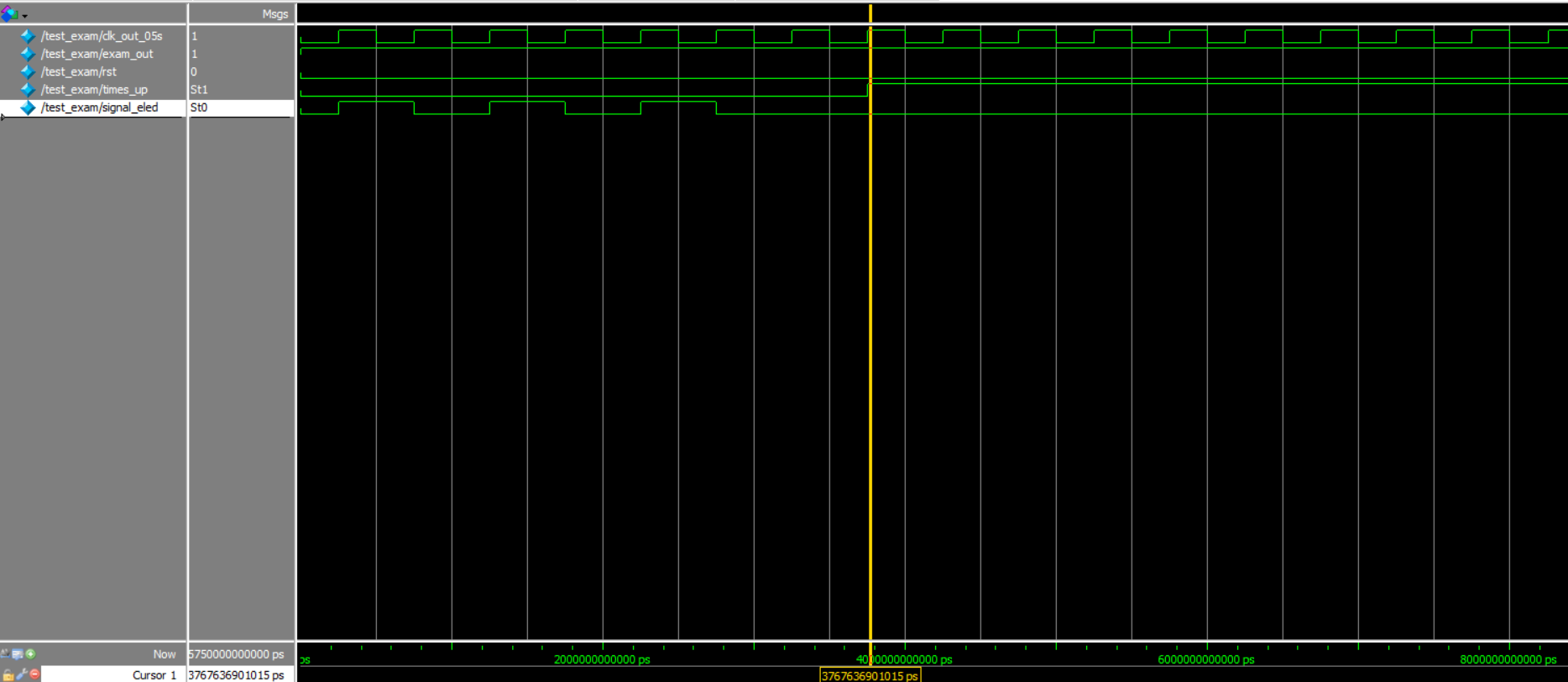
中间BTN[6]=1，state变为了011(输出)，并且result\_signal为1，当outcome\_of\_buy[1]=0时，result\_time=1信号来到后，就进入了待机态（001），而后面outcome\_of\_buy[1]=1,其他相同则再次回到了输入态（010）。最终rst=1，则state=001(待机)，swt7=0,则state=111（关机）。

仿真结果与预期相同。

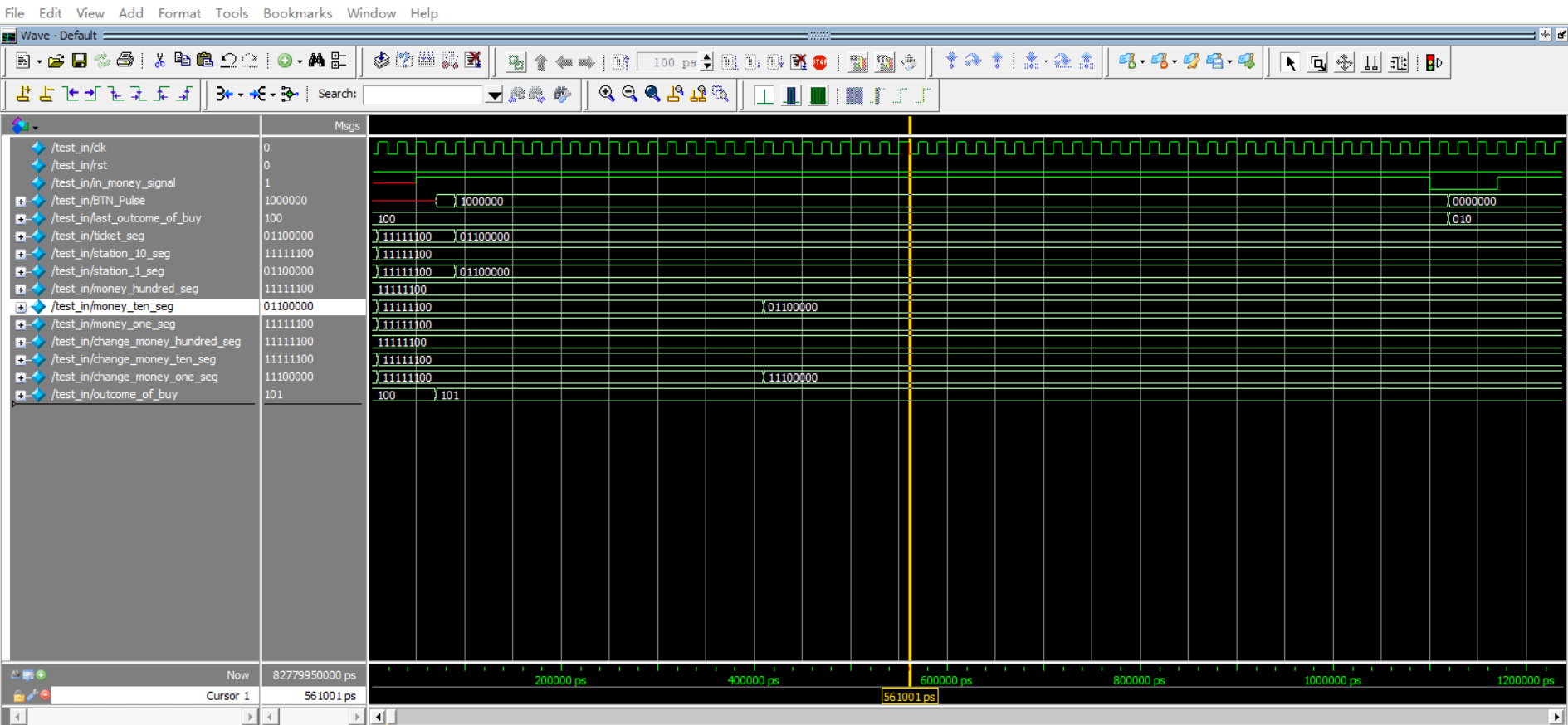
**2.自检模块仿真**

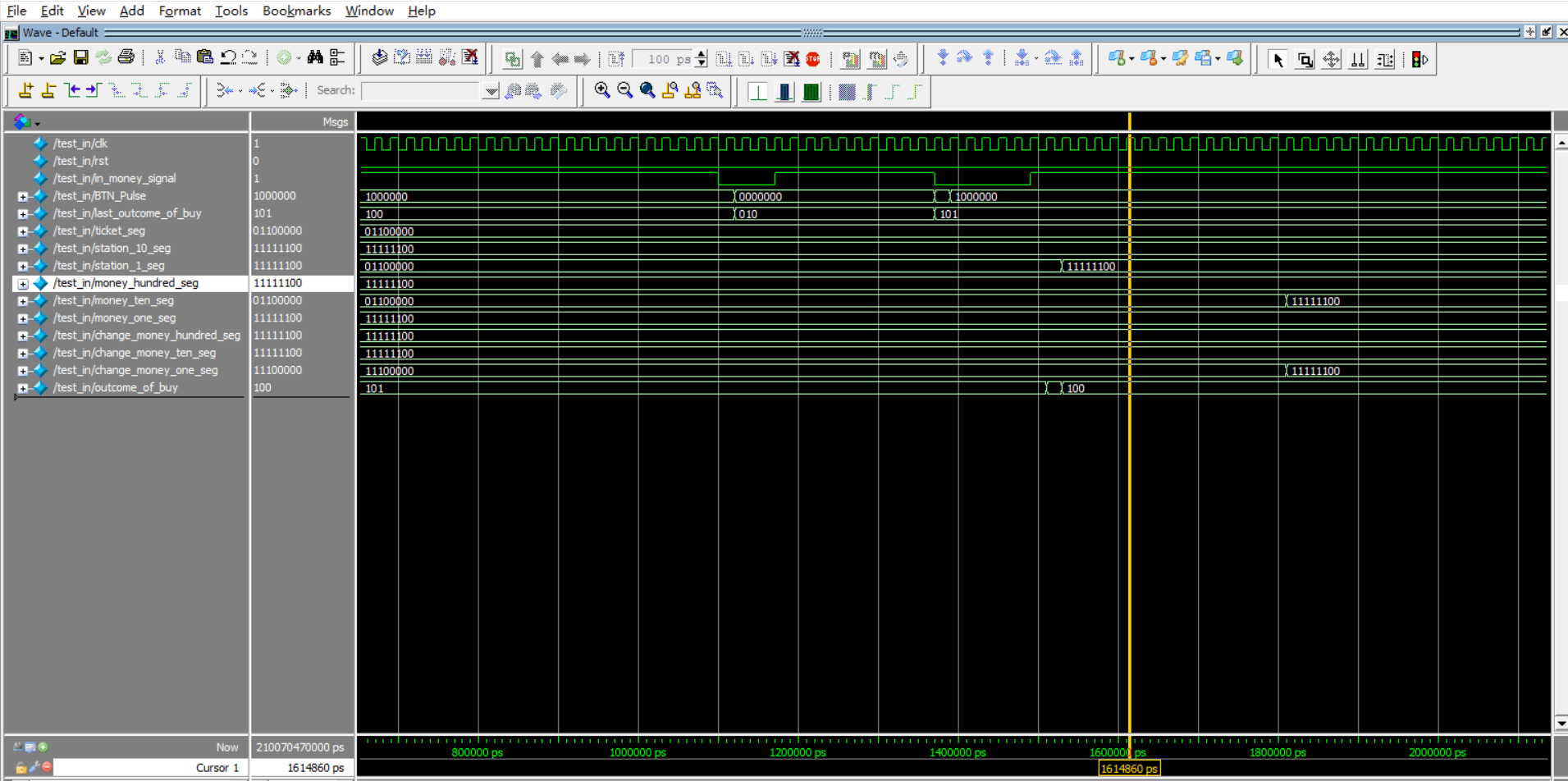
从波形图可以看出在exam\_out使能信号为1后，signal\_eled就出现了三个持续0.5s的高电位，对应着点阵和数码管开机自检时的3次0.5秒亮灭，再然后times\_up就变为1进行反馈。

该波形也与设想一致。

****

**3.输入模块仿真**

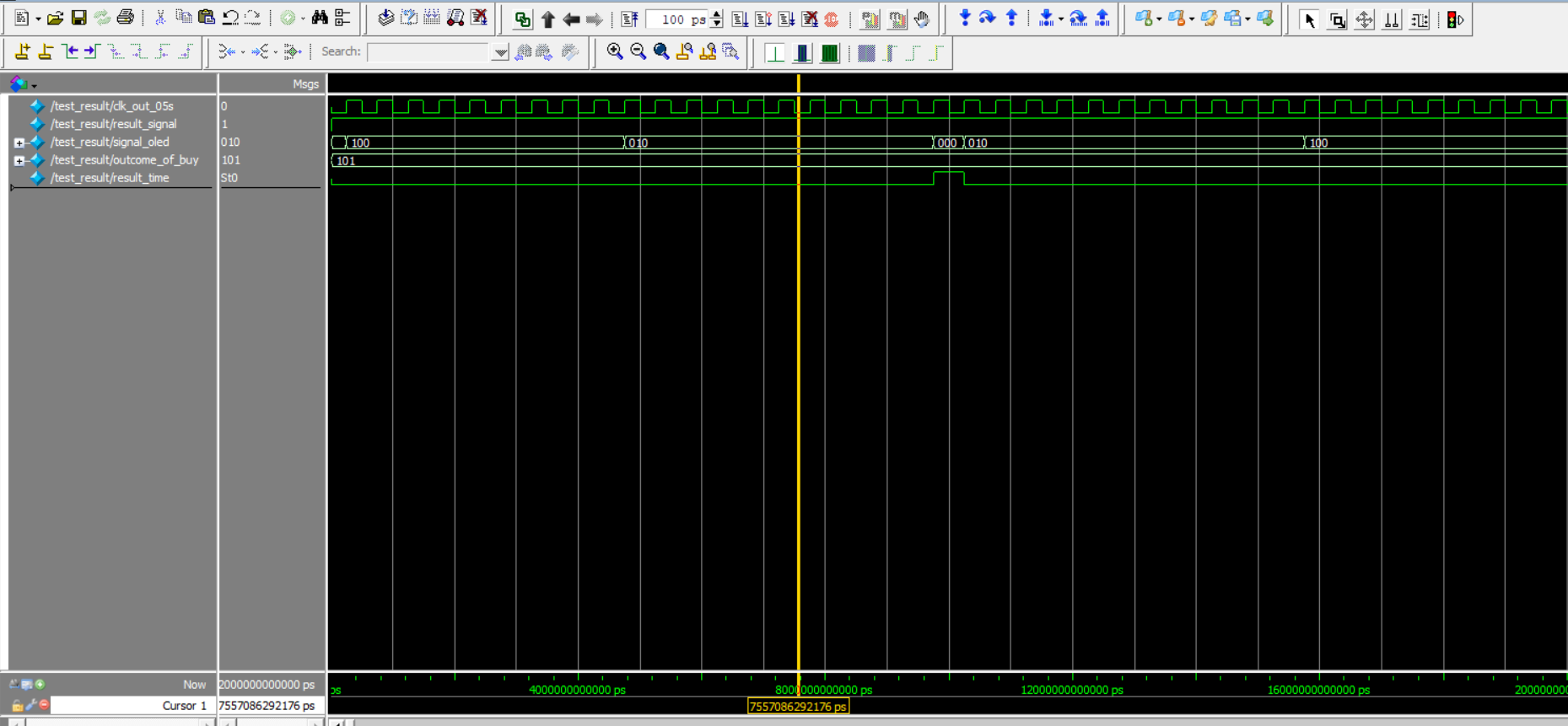
****



如第一张图所示，当in\_money\_signal使能信号接入且上一次购票成功(100)后，消抖后按键为01000011（输入十元，1站，1张票），则站数和票数的数码管输出信号为01100000（即为数字1），钱数输出money\_ten\_seg为01100000，即10元。而change\_money\_one\_seg为11100000，即代表数字7，即找零7元。当前购买结果outcome\_of\_buy为101，即购票成功且需要找零。而钱的位数的数码管信号输出较晚，是因为其中还有一个移三位分离百个十位的电路，所以延迟了一会才输出。而后，当上一次购买结果为010（失败），则输入模块内部的寄存器中所保存的站数票数等信息依旧保存，以便于转回输入模块进行修改操作，同时下方outcome\_of\_buy为101代表购票成功且需找零。

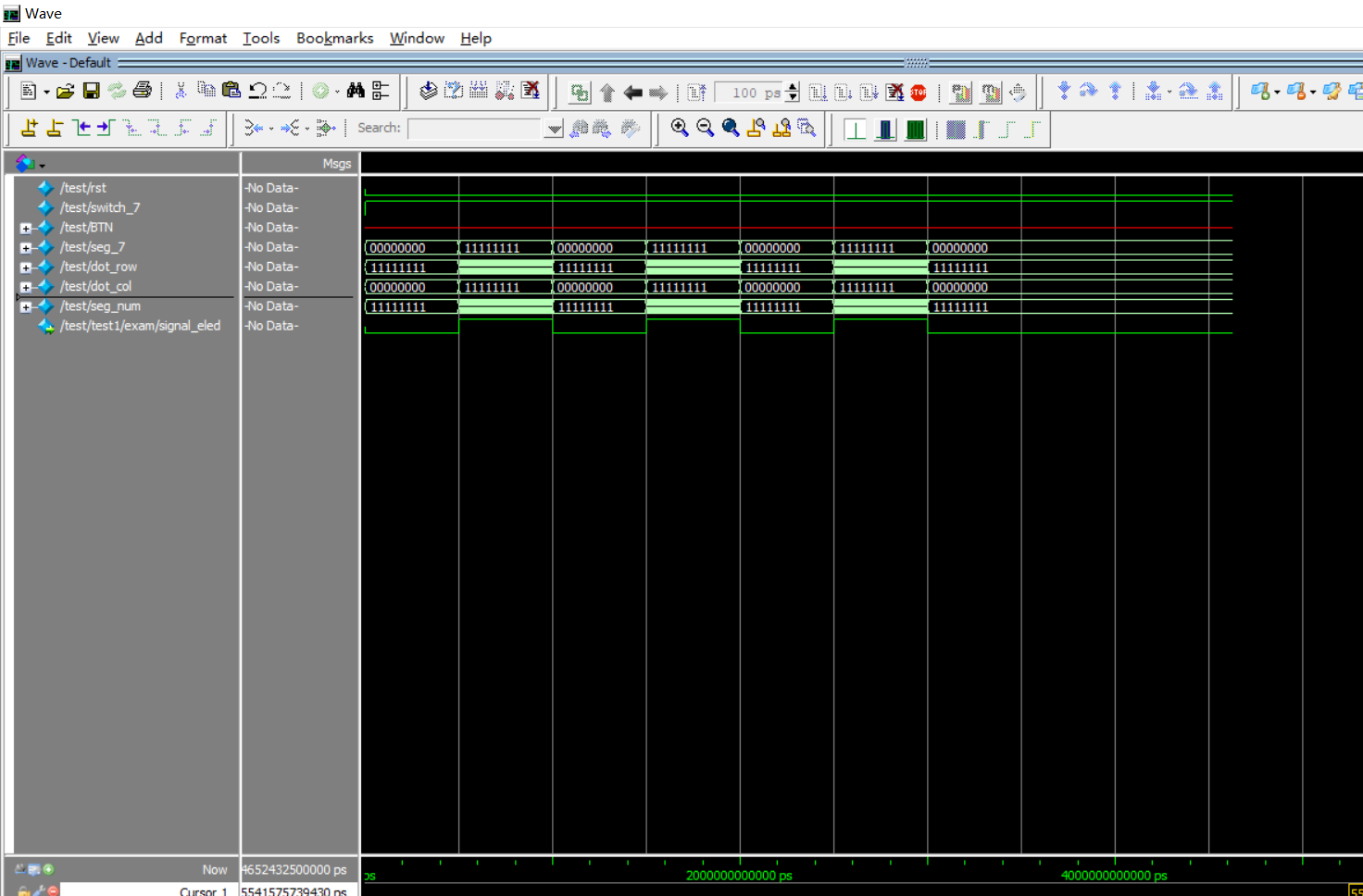
第二张图：接着输入使能信号为0，并按键，可以看到晶体管也并没有变化。接着将上次购票结果设为101（购票成功且需找零，可以看到输入状态下的晶体管值都变为11111100（晶体管0），即完成复位。故综上，输入信号仿真也与预期相同。

**4.输出模块仿真**

****

图中，clk\_out\_05s为2Hz时钟，当使能信号为高时，购买结果为3’b101（购买成功且需找零），可以看到10s内，前5s的signal\_oled（控制点阵动画和蜂鸣器的信号）为100，后五秒为010；10s后生成负反馈信号result\_time。这些都与预期一致。而之后signal\_oled就没有影响了，因为使能信号为0，即点阵数码管等都不显示这个状态。

**5.数码管模块仿真和点阵模块仿真**

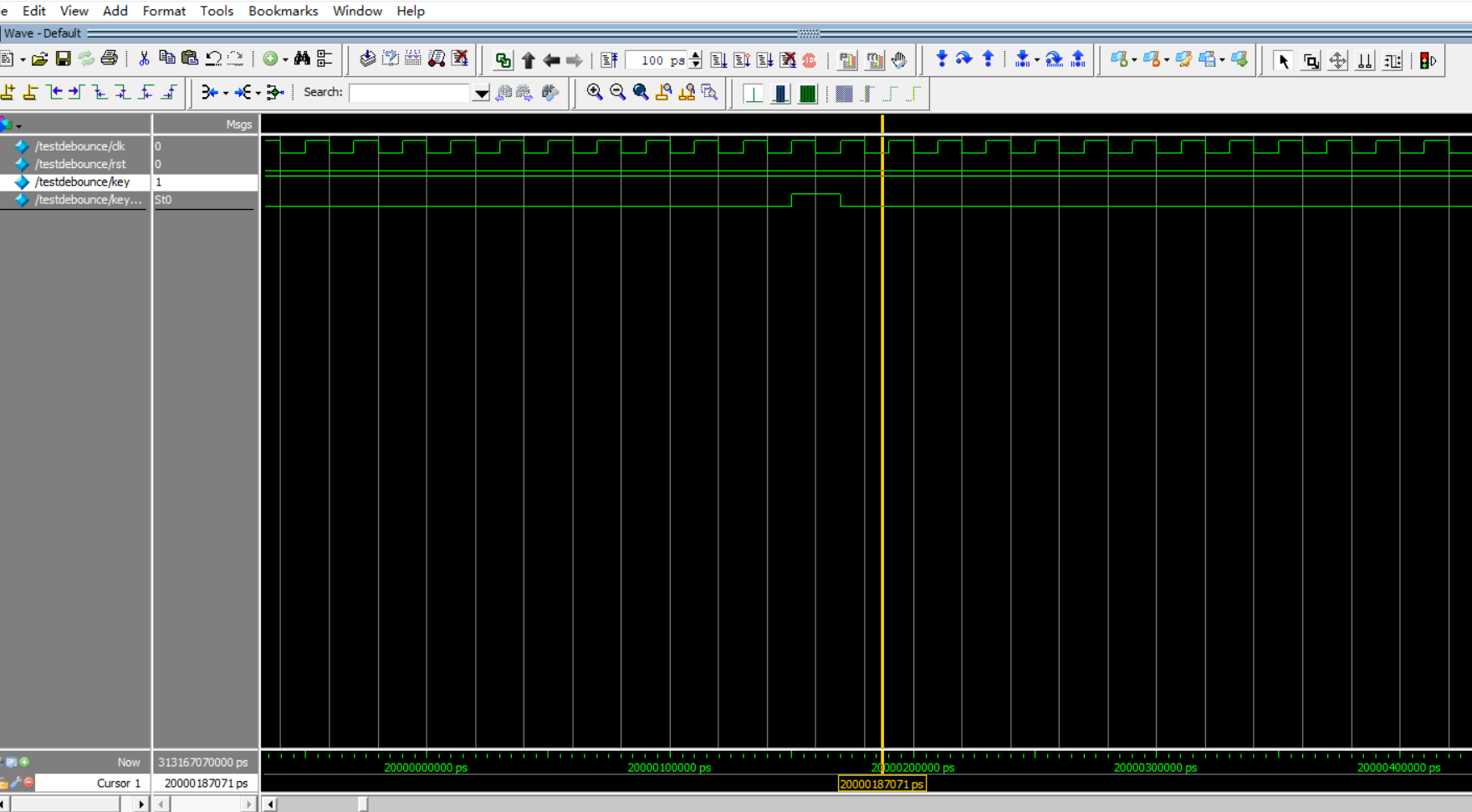
****

可以看到开机后点阵和数码管亮灭3次，每次0.5s，与理想情况相同。其他动画情况以及显示情况，只是在模块里更换了使能信号和case语句，故这里略过。

**6.蜂鸣器模块仿真**

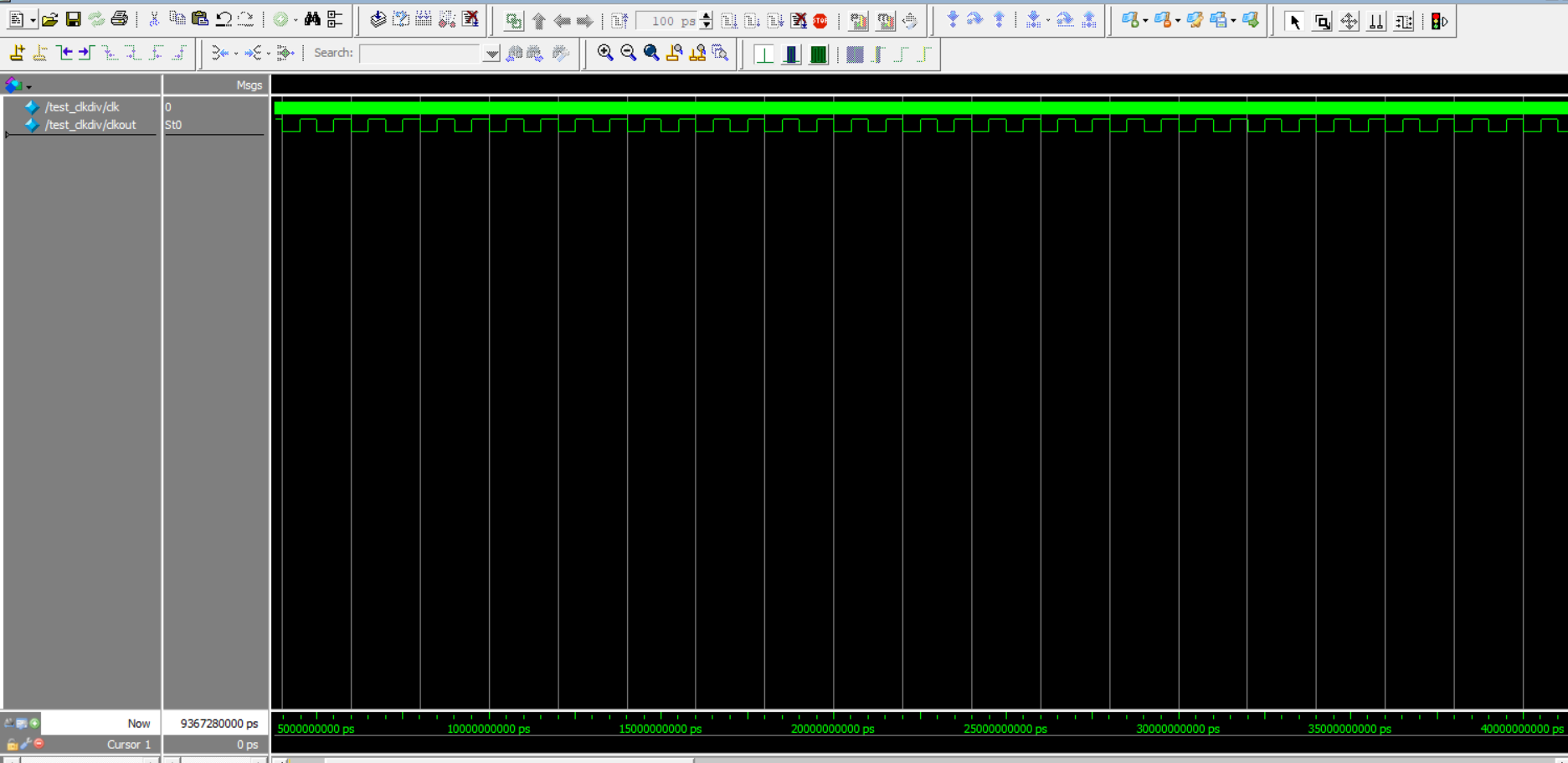
蜂鸣器模块和点阵模块，数码管模块功能类似，都是由相同的结果信号控制，故只仿真点阵和数码管模块。

**7.按键消抖仿真**

****

从波形中，可以看出一直按按键最后在20ms后产生了唯一一个脉冲，脉冲在50MHz两个上升沿之间，而实际中，因为门电路之间的延迟时间，所以脉冲为1时在别的模块中应只对应一个时钟的上升沿，故相当于按键一次。

**8.分频器仿真**



从图中可以看出50MHz分频出的1KHz时钟周期为1ms。代码中改变频率使用的是defparam的方法，故与预期相同。

四、代码

4.1顶端模块-subway

module subway(clk,rst,switch\_7,BTN,seg\_7,seg\_num,dot\_row,dot\_col,dot\_col\_g,speaker);

input clk,rst,switch\_7;

input [6:0]BTN;

output [7:0]seg\_7;

output [7:0]seg\_num;

output [7:0]dot\_row;

output [7:0]dot\_col; //红色点阵

output [7:0]dot\_col\_g; //绿色点阵

output speaker; //蜂鸣器

wire [6:0]BTN\_Pulse; //消抖后的按键

wire exam\_out; //自检状态 给dot 和seg 的信号。

wire times\_up; //自检的时间倒计时 为高时，代表过了3s 则进入待机状态

wire in\_money\_signal,result\_signal; //分别是表示当前状态为 输入钱，购票的信号 和 输出购票结果的使能信号。

wire signal\_eld; // exam\_led

wire [7:0]ticket\_seg,station\_10\_seg,station\_1\_seg,money\_hundred\_seg,money\_ten\_seg,money\_one\_seg; //不同的数码管显示

wire [7:0]change\_money\_hundred\_seg,change\_money\_ten\_seg,change\_money\_one\_seg; //找零的数码管显示

wire [2:0]signal\_oled; //控制不同动画显示的信号。

wire clk\_1k; //1khz时钟

wire clk\_out\_05s; //2Hz的时钟， 来控制0.5秒闪烁

wire result\_time; //结果输出 10s后 自动进入待机状 反馈信号

wire [2:0]outcome\_of\_buy; //购票的结果， 是否成功， 是否需要找零

Fsm1 t1(

.clk(clk),

.rst(rst),

.swt7(switch\_7),

.BTN(BTN\_Pulse[6:0]),

.exam\_out(exam\_out),

.outcome\_of\_buy(outcome\_of\_buy),

.times\_up(times\_up),

.in\_money\_signal(in\_money\_signal),

.result\_signal(result\_signal),

.result\_time(result\_time)

);

defparam m1.N=7;

debounce m1(

.clk(clk),

.rst(rst),

.key(BTN[6:0]),

.key\_pulse(BTN\_Pulse[6:0])

);

self\_exam exam(

.clk(clk\_out\_05s),

.signal(exam\_out),

.rst(rst),

.times\_up(times\_up),

.signal\_eled(signal\_eled)

);

input\_money in(

.clk(clk),

.rst(rst),

.BTN(BTN\_Pulse[6:0]),

.signal(in\_money\_signal),

.ticket\_seg(ticket\_seg),

.station\_10\_seg(station\_10\_seg),

.station\_1\_seg(station\_1\_seg),

.money\_hundred\_seg(money\_hundred\_seg),

.money\_ten\_seg(money\_ten\_seg),

.money\_one\_seg(money\_one\_seg),

.change\_money\_hundred\_seg(change\_money\_hundred\_seg),

.change\_money\_ten\_seg(change\_money\_ten\_seg),

.change\_money\_one\_seg(change\_money\_one\_seg),

.outcome\_of\_buy(outcome\_of\_buy),

.last\_outcome\_of\_buy(outcome\_of\_buy)

);

seg seg\_1(

.clk(clk\_1k),

.rst(rst),

.signal\_exam(signal\_eled),

.signal\_in(in\_money\_signal),

.result\_signal(result\_signal),

.ticket\_seg(ticket\_seg),

.station\_10\_seg(station\_10\_seg),

.station\_1\_seg(station\_1\_seg),

.money\_hundred\_seg(money\_hundred\_seg),

.money\_ten\_seg(money\_ten\_seg),

.money\_one\_seg(money\_one\_seg),

.change\_money\_hundred\_seg(change\_money\_hundred\_seg),

.change\_money\_ten\_seg(change\_money\_ten\_seg),

.change\_money\_one\_seg(change\_money\_one\_seg),

.seg\_7(seg\_7),

.seg\_num(seg\_num)

);

dot dot\_1(

.clk(clk\_1k),

.rst(rst),

.signal\_exam(signal\_eled),

.signal\_out(result\_signal),

.dot\_row(dot\_row),

.dot\_col(dot\_col),

.dot\_col\_g(dot\_col\_g),

.signal\_oled(signal\_oled)

);

result r1( //结果模块

.clk(clk\_out\_05s),

.result\_signal(result\_signal),

.signal\_oled(signal\_oled),

.result\_time(result\_time),

.outcome\_of\_buy(outcome\_of\_buy)

);

beep bibi( //蜂鸣器模块

.clk\_50M(clk), //50Mhz

.rst(rst),

.signal(signal\_oled),

.tone\_en(result\_signal), //使能端

.speaker(speaker)

);

clk\_d clk\_1( //1khz

.clkin(clk),

.clkout(clk\_1k));

defparam clk\_05s.MAX=12500000; //12500000

clk\_d clk\_05s //2Hz时钟

(.clkin(clk),

.clkout(clk\_out\_05s)

);

endmodule

4.2状态机

module Fsm1(clk,rst,swt7,outcome\_of\_buy,BTN,exam\_out,times\_up,in\_money\_signal,result\_signal,result\_time);

input clk;

input rst;

input swt7;

input [2:0]outcome\_of\_buy; //根据购票的结果来判定是进入什么状态

input times\_up; //自检状态结束的反馈信号

input [6:0]BTN;

input result\_time; //输出状态结束的使能信号

output reg exam\_out; //自检状态的使能信号

wire swt7\_edge; //开关的上升脉冲

output reg in\_money\_signal; //输入状态的使能信号

output reg result\_signal; //结果状态的使能信号

reg swt7\_last,swt7\_now;

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

reg outcome; //比较结果

parameter [2:0] state\_off=3'b111,

state\_exam=3'b000,

state\_wait=3'b001,

state\_in=3'b010,

state\_out=3'b011;

wire any\_prs; //有按键被按 则为高

always @(posedge clk)

begin

if(swt7==0) //关机

state<=state\_off;

else if(swt7\_edge&swt7) //开机上升沿

state<=state\_exam;

else if(rst) //复位

state<=state\_wait;

else

state<=next\_state;

end

always @(state or swt7 or times\_up or any\_prs or BTN[6] or result\_time)

begin

case(state)

state\_off:begin if(swt7) next\_state<=state\_exam; else next\_state<=state\_off; end

state\_exam:begin if(times\_up) next\_state<=state\_wait; else if(swt7==0) next\_state=state\_off; else next\_state<=state\_exam; end

state\_wait: begin if(any\_prs) next\_state<=state\_in;else if(swt7==0) next\_state=state\_off; else next\_state<=state\_wait ; end

state\_in:begin if(BTN[6]) next\_state<=state\_out;else if(swt7==0) next\_state=state\_off; else next\_state<=state\_in; end

state\_out:begin if(result\_time&(outcome\_of\_buy[1]==0)) next\_state<=state\_wait;else if(result\_time&(outcome\_of\_buy[1])) next\_state<=state\_in;else if(swt7==0) next\_state=state\_off; else next\_state<=state\_out;end

default: next\_state<=state\_wait;

endcase

end

always @(state)

begin //输出使能端信号

case(state)

state\_off: begin exam\_out=0; in\_money\_signal=0; result\_signal=0;end

state\_exam: begin exam\_out=1; in\_money\_signal=0; result\_signal=0;end

state\_wait: begin exam\_out=0; in\_money\_signal=0; result\_signal=0;end

state\_in: begin exam\_out=0; in\_money\_signal=1; result\_signal=0;end

state\_out: begin exam\_out=0; in\_money\_signal=0; result\_signal=1;end

endcase

end

/\*

判断sw7的上升沿

\*/

always @(posedge clk)

begin

swt7\_now<=swt7;

swt7\_last<=swt7\_now;

end

assign swt7\_edge=swt7\_now &(~swt7\_last);

press\_find find1( //检测按键状态

.BN(BTN[5:0]),

.any\_prs(any\_prs)

);

endmodule

4.3 自检模块

module self\_exam(signal,clk,rst,times\_up,signal\_eled);

input signal;

input clk;

input rst;

wire clk\_out\_3s; //计时

reg [2:0]cnt2=3'b000;

output reg times\_up=0;

output reg signal\_eled=0; //控制点阵和数码管的亮灭

//实现0.5秒计时

always@(posedge clk or posedge rst)

begin

if(rst)

cnt2<=0;

else

begin

if(signal)

if(cnt2<=3'b110)

cnt2<=cnt2+1'b1;

else if(cnt2==3'b111) //计时3s

times\_up<=1;

else

cnt2<=3'b0;

else //信号断开时 复位

begin

cnt2<=3'b0;

times\_up<=0;

end

end

end

always@(cnt2) //根据cnt2 来判断状态

begin

case(cnt2)

3'b000:signal\_eled<=0;

3'b001:signal\_eled<=1;

3'b010:signal\_eled<=0;

3'b011:signal\_eled<=1;

3'b100:signal\_eled<=0;

3'b101:signal\_eled<=1;

3'b110:signal\_eled<=0;

default:signal\_eled<=0;

endcase

end

//控制点阵显示

//always @(posedge clk) //当signal 从0 到1 即进入自检状态，则根据edge 对tme重置为0；

//begin

//signal\_now<=signal;

//signal\_last<=signal\_now;

//end

//assign signal\_edge=signal\_now &(~signal\_last); \*/

endmodule

4.4 输入模块

module input\_money(signal,clk,rst,BTN,ticket\_seg,station\_10\_seg,station\_1\_seg,money\_hundred\_seg,money\_ten\_seg,money\_one\_seg,change\_money\_hundred\_seg,change\_money\_ten\_seg,change\_money\_one\_seg,last\_outcome\_of\_buy,outcome\_of\_buy);

input signal; //使能信号

input clk;

input rst;

input [6:0]BTN;

input [2:0]last\_outcome\_of\_buy; //上一次购买结果

reg [3:0]station\_10=4'b0;

reg [3:0]station\_1=4'b0;

reg [3:0]ticket=4'b0;

reg [3:0]money\_10=4'b0; //输入

reg [3:0]money\_5=4'b0;

reg [3:0]money\_1=4'b0;

reg [7:0]one\_ticket\_money=8'b0;

reg [6:0]stations=7'b0;

reg [7:0]ticket\_money=8'b0;

reg [7:0]money\_in=8'b0;

wire [1:0]money\_hundred;

wire [3:0]money\_ten;

wire [3:0]money\_one;

output reg [7:0]ticket\_seg=8'b0;

output reg [7:0]station\_10\_seg=8'b0;

output reg [7:0]station\_1\_seg=8'b0;

output reg [7:0]money\_hundred\_seg=8'b11111100;

output reg [7:0]money\_ten\_seg=8'b0;

output reg [7:0]money\_one\_seg=8'b0;

output reg [2:0]outcome\_of\_buy=3'b0;

// succeed failed change\_money

// reg [7:0]change\_money;

reg [2:0]cnt=3'b0;

//reg signal\_last,signal\_now;

wire signal\_edge; //检测使能信号的上升沿

//票数

always@(posedge clk) //问题:按一次键持续20ms 而clk频率高，会不会一直加。

begin

if((signal\_edge&(last\_outcome\_of\_buy[1]==0))||rst) //检测到signal 上升沿 代表来到输入状态，并且上次购票成功则清零，否则保持

ticket<=4'b1;

else if(signal)

begin

if(BTN[0])

ticket<=ticket+1;

else if(ticket==10) // 当不按键 的时候，clk上升沿来到 开始检测

ticket<=4'b001;

end

else ;

end

//站数——10位

always @(posedge clk)

begin

if((signal\_edge&(last\_outcome\_of\_buy[1]==0))||rst) //检测到signal 上升沿 代表来到输入状态，并且上次购票成功则清零，否则保持

station\_10<=4'b0;

else if(signal)

begin

if(BTN[2])

station\_10<=station\_10+1;

else if(station\_10==10)

station\_10<=4'b000;

end

else

;

end

//站数--个位

always @(posedge clk)

begin

if((signal\_edge&(last\_outcome\_of\_buy[1]==0))||rst) //检测到signal 上升沿 代表来到输入状态，并且上次购票成功则清零，否则保持

station\_1<=4'b0;

else if(signal) begin

if(BTN[1])

station\_1<=station\_1+1;

else if(station\_1==10)

station\_1<=4'b000;

end

else

;

end

always @(station\_10 or station\_1)

begin

stations<=10\*station\_10+station\_1;

end

/\*

计算一张票的价钱。

\*/

always @(stations)

begin

if(stations==0)

one\_ticket\_money<=0;

else if(stations<=5)

one\_ticket\_money<=3;

else if(stations<=10)

one\_ticket\_money<=4;

else if(stations<=15)

one\_ticket\_money<=5;

else if(stations<=20)

one\_ticket\_money<=6;

else if(stations<=30)

one\_ticket\_money<=7; // (6+(stations-20)/10+1); 从85% 到 56%

else if(stations<=40)

one\_ticket\_money<=8;

else if(stations<=50)

one\_ticket\_money<=9;

else if(stations<=60)

one\_ticket\_money<=10;

else if(stations<=70)

one\_ticket\_money<=11;

else if(stations<=80)

one\_ticket\_money<=12;

else if(stations<=90)

one\_ticket\_money<=13;

else if(stations<=100)

one\_ticket\_money<=14;

else ;

end

/\*

计算所有的票的钱

\*/

always @(posedge clk) //一张票最多为14元钱 最多9张故 最多126元

begin

ticket\_money<=one\_ticket\_money\*ticket;

end

always @(posedge clk)

begin

if((signal\_edge&(last\_outcome\_of\_buy[1]==0))||rst) //检测到signal 上升沿 代表来到输入状态，并且上次购票成功则清零，否则保持

money\_in<=0;

else if(signal)

begin

if(BTN[5])

money\_in<=money\_in+10;

else if(BTN[4])

money\_in<=money\_in+5;

else if(BTN[3])

money\_in<=money\_in+1;

else ;

end

else

; //进入下一个状态后 money\_in清零 而 ticket\_money 是由函数决定ticket变为0 故也为0

end

/\* seg上显示钱数。

\*/

//always @(posedge clk)

//begin

//money\_hundred<=money\_in/100;

//money\_ten<=(money\_in-money\_hundred\*100)/10;

//money\_one<=money\_in-money\_hundred\*100-money\_ten\*10;

//end

/\*

输出结果信号 购票成功与否 结果 找零

\*/

reg [7:0]change\_money=8'b0;

wire [1:0]change\_money\_hun;

wire [3:0]change\_money\_ten;

wire [3:0]change\_money\_one;

output reg [7:0]change\_money\_hundred\_seg=8'b11111100;

output reg [7:0]change\_money\_ten\_seg=8'b0;

output reg [7:0]change\_money\_one\_seg=8'b0;

always@(posedge clk)

begin

if(rst) //复位清零 因为要传给输出 所以不用清零。

change\_money<=0;

else if(money\_in<ticket\_money||(money\_in==0)) //不输钱也是错误

change\_money<=8'hff; //error 错误 钱不够

else if(money\_in==ticket\_money)

change\_money<=8'b0; //钱正好 不需找零。

else

change\_money<=money\_in-ticket\_money;

end

//always @(posedge clk)

//begin

//change\_money\_hundred<=change\_money/100; //分离出 百十个位

//change\_money\_ten<=(change\_money-change\_money\_hundred\*100)/10;

//change\_money\_one<=change\_money-change\_money\_hundred\*100-change\_money\_ten\*10;

//end

always@(posedge clk)

begin

if(change\_money\_hun>=2) //因为找零大于二百 就意味着错误，因为上面已经定义了钱不够时change\_money为255 正常情况下是不会投这么多钱的。

begin //所以一切都在正常情况的前提下进行

change\_money\_hundred\_seg=8'b01101110;

change\_money\_ten\_seg=8'b01101110;

change\_money\_one\_seg=8'b01101110;

end

else

begin

case(change\_money\_hun)

2'b00: change\_money\_hundred\_seg=8'b11111100; //0

2'b01: change\_money\_hundred\_seg=8'b01100000; //1

2'b10: change\_money\_hundred\_seg=8'b11011010; //2

endcase

case(change\_money\_ten)

4'b0000: change\_money\_ten\_seg=8'b11111100;

4'b0001: change\_money\_ten\_seg=8'b01100000;

4'b0010: change\_money\_ten\_seg=8'b11011010;

4'b0011: change\_money\_ten\_seg=8'b11110010;

4'b0100: change\_money\_ten\_seg=8'b01100110;

4'b0101: change\_money\_ten\_seg=8'b10110110;

4'b0110: change\_money\_ten\_seg=8'b10111110;

4'b0111: change\_money\_ten\_seg=8'b11100000;

4'b1000: change\_money\_ten\_seg=8'b11111110;

4'b1001: change\_money\_ten\_seg=8'b11110110;

endcase

case(change\_money\_one)

4'b0000: change\_money\_one\_seg=8'b11111100;

4'b0001: change\_money\_one\_seg=8'b01100000;

4'b0010: change\_money\_one\_seg=8'b11011010;

4'b0011: change\_money\_one\_seg=8'b11110010;

4'b0100: change\_money\_one\_seg=8'b01100110;

4'b0101: change\_money\_one\_seg=8'b10110110;

4'b0110: change\_money\_one\_seg=8'b10111110;

4'b0111: change\_money\_one\_seg=8'b11100000;

4'b1000: change\_money\_one\_seg=8'b11111110;

4'b1001: change\_money\_one\_seg=8'b11110110;

endcase

end

end

///////////////////////////////////////////

always @(money\_in or ticket\_money)

begin

if(rst)

outcome\_of\_buy<=3'b0;

if(money\_in==0)

outcome\_of\_buy<=3'b010;

else if(money\_in<ticket\_money)

outcome\_of\_buy<=3'b010; //购票失败

else if(money\_in==ticket\_money)

outcome\_of\_buy<=3'b100;

else

outcome\_of\_buy<=3'b101; //购票成功且需要找零

end

always @(posedge clk)

begin

case(ticket)

4'b0000: ticket\_seg=8'b11111100;

4'b0001: ticket\_seg=8'b01100000;

4'b0010: ticket\_seg=8'b11011010;

4'b0011: ticket\_seg=8'b11110010;

4'b0100: ticket\_seg=8'b01100110;

4'b0101: ticket\_seg=8'b10110110;

4'b0110: ticket\_seg=8'b10111110;

4'b0111: ticket\_seg=8'b11100000;

4'b1000: ticket\_seg=8'b11111110;

4'b1001: ticket\_seg=8'b11110110;

endcase

end

always @(posedge clk)

begin

case(station\_10)

4'b0000: station\_10\_seg=8'b11111100;

4'b0001: station\_10\_seg=8'b01100000;

4'b0010: station\_10\_seg=8'b11011010;

4'b0011: station\_10\_seg=8'b11110010;

4'b0100: station\_10\_seg=8'b01100110;

4'b0101: station\_10\_seg=8'b10110110;

4'b0110: station\_10\_seg=8'b10111110;

4'b0111: station\_10\_seg=8'b11100000;

4'b1000: station\_10\_seg=8'b11111110;

4'b1001: station\_10\_seg=8'b11110110;

endcase

end

always @(posedge clk)

begin

case(station\_1)

4'b0000: station\_1\_seg=8'b11111100;

4'b0001: station\_1\_seg=8'b01100000;

4'b0010: station\_1\_seg=8'b11011010;

4'b0011: station\_1\_seg=8'b11110010;

4'b0100: station\_1\_seg=8'b01100110;

4'b0101: station\_1\_seg=8'b10110110;

4'b0110: station\_1\_seg=8'b10111110;

4'b0111: station\_1\_seg=8'b11100000;

4'b1000: station\_1\_seg=8'b11111110;

4'b1001: station\_1\_seg=8'b11110110;

endcase

end

always @(posedge clk)

begin

case(money\_hundred) //money\_hundred

2'b00: money\_hundred\_seg=8'b11111100;

2'b01: money\_hundred\_seg=8'b01100000;

2'b10: money\_hundred\_seg=8'b11011010;

2'b11: money\_hundred\_seg=8'b11110010;

endcase

end

always @(posedge clk)

begin

case(money\_ten) //money\_ten

4'b0000: money\_ten\_seg=8'b11111100;

4'b0001: money\_ten\_seg=8'b01100000;

4'b0010: money\_ten\_seg=8'b11011010;

4'b0011: money\_ten\_seg=8'b11110010;

4'b0100: money\_ten\_seg=8'b01100110;

4'b0101: money\_ten\_seg=8'b10110110;

4'b0110: money\_ten\_seg=8'b10111110;

4'b0111: money\_ten\_seg=8'b11100000;

4'b1000: money\_ten\_seg=8'b11111110;

4'b1001: money\_ten\_seg=8'b11110110;

endcase

end

always @(posedge clk)

begin

case(money\_one) //money\_one

4'b0000: money\_one\_seg=8'b11111100;

4'b0001: money\_one\_seg=8'b01100000;

4'b0010: money\_one\_seg=8'b11011010;

4'b0011: money\_one\_seg=8'b11110010;

4'b0100: money\_one\_seg=8'b01100110;

4'b0101: money\_one\_seg=8'b10110110;

4'b0110: money\_one\_seg=8'b10111110;

4'b0111: money\_one\_seg=8'b11100000;

4'b1000: money\_one\_seg=8'b11111110;

4'b1001: money\_one\_seg=8'b11110110;

endcase

end

//BCD移三位 分出百十个位

find\_wei money\_in\_wei

(

.clk(clk),

.rst\_n(rst),

.bin(money\_in),

.one(money\_one),

.ten(money\_ten),

.hun(money\_hundred)

);

find\_wei change\_m\_wei

(

.clk(clk),

.rst\_n(rst),

.bin(change\_money),

.one(change\_money\_one),

.ten(change\_money\_ten),

.hun(change\_money\_hun)

);

reg signal\_now;

reg signal\_last;

//wire singal\_edge

always @(posedge clk) //使一切在新的信号输入时为0；

begin

signal\_now<=signal;

signal\_last<=signal\_now;

end

assign signal\_edge=signal\_now &(~signal\_last);

endmodule

4.5结果模块

module result(clk,result\_signal,signal\_oled,result\_time,outcome\_of\_buy);

input clk;

input result\_signal;

input [2:0]outcome\_of\_buy;

output reg[2:0] signal\_oled=3'b000;

output reg result\_time; //输出的时间信号作为使能信号

parameter TIME=20; //输入clk 为2Hz 0.5s的计时器所以 TME=10即为 计到5s

parameter time1=10; //time1 用来将时间分一半给 出票和 找零

reg [4:0] cnt=0;

always@(posedge clk)

begin

if(result\_signal) //使能信号为1时开始计数

cnt<=cnt+1'b1;

else

cnt<=0;

end

always@(cnt)

begin

if(cnt==TIME) //计时到5s 后 输入信号给状态机 告诉输出结果状态结束

begin result\_time<=1; signal\_oled<=0; end

else

begin

result\_time<=0;

case(outcome\_of\_buy)

3'b101:

begin

if(cnt<time1)

begin signal\_oled<=3'b100; end//出票动画

else

begin signal\_oled<=3'b010; end

end

//cnt2<m 出票动画，>m 出钱动画倒计时 //需要找零

3'b100: begin signal\_oled<=3'b100; end//cnt2<m 出票动画 不需要找零 //声音

3'b010: begin signal\_oled<=3'b001; end// 滚动 错误 //声音

endcase

end

end

endmodule

4.6点阵模块

module dot(clk,rst,signal\_exam,signal\_out,dot\_row,dot\_col,signal\_oled,dot\_col\_g);

input clk;

input rst;

input signal\_exam; //控制开机自检

input [2:0]signal\_oled; // 不同的动画切换

input signal\_out; //确保结尾动画是在输出状态进行

output reg[7:0] dot\_row;

output reg[7:0] dot\_col;

output reg[7:0] dot\_col\_g;

//input outcome\_of\_money;

reg [2:0]cnt1=3'b0; //cnt1 用来控制刷新频率

reg [2:0]cnt2=3'b0; //cnt2 控制不同动画 之间切换。

parameter DTIME=40; //视觉暂留 ，25帧每秒 所以就是每秒cnt2 变化25次 如果为1khz 则一秒1k次变化，故1000/25=40

reg [7:0] CNT=8'b0;

always @(cnt1)

begin

if(rst)

begin dot\_row=8'hff; dot\_col=8'b0;end

else if(signal\_exam) //在自检状态

case(cnt1[2:0])

3'b000:begin dot\_row=8'b11111110; dot\_col=8'hff;end

3'b001:begin dot\_row=8'b11111101; dot\_col=8'hff;end

3'b010:begin dot\_row=8'b11111011; dot\_col=8'hff;end

3'b011:begin dot\_row=8'b11110111; dot\_col=8'hff;end

3'b100:begin dot\_row=8'b11101111; dot\_col=8'hff;end

3'b101:begin dot\_row=8'b11011111; dot\_col=8'hff;end

3'b110:begin dot\_row=8'b10111111; dot\_col=8'hff;end

3'b111:begin dot\_row=8'b01111111; dot\_col=8'hff;end

endcase

else if(signal\_out) //表示在 输出的状态

begin

case(signal\_oled) //根据信号选择不同的输出

3'b100:

begin dot\_row=8'hff; dot\_col=8'b0; //输出正确时 保证其他状态的点阵不亮

case(cnt2)

3'b000:

begin

case(cnt1)

3'b000:begin dot\_row=8'b01111111; dot\_col\_g=8'b01111110;end

3'b001:begin dot\_row=8'b10111111; dot\_col\_g=8'b01111110;end

3'b010:begin dot\_row=8'b11011111; dot\_col\_g=8'b01111110;end

3'b011:begin dot\_row=8'b11101111; dot\_col\_g=8'h0;end

3'b100:begin dot\_row=8'b11110111; dot\_col\_g=8'h0;end

3'b101:begin dot\_row=8'b11111011; dot\_col\_g=8'h0;end

3'b110:begin dot\_row=8'b11111101; dot\_col\_g=8'h0;end

3'b111:begin dot\_row=8'b11111110; dot\_col\_g=8'h0;end

endcase

end

3'b001:

begin

case(cnt1)

3'b000:begin dot\_row=8'b01111111; dot\_col\_g=8'b0;end

3'b001:begin dot\_row=8'b10111111; dot\_col\_g=8'b01111110;end

3'b010:begin dot\_row=8'b11011111; dot\_col\_g=8'b01111110;end

3'b011:begin dot\_row=8'b11101111; dot\_col\_g=8'b01111110;end

3'b100:begin dot\_row=8'b11110111; dot\_col\_g=8'h0;end

3'b101:begin dot\_row=8'b11111011; dot\_col\_g=8'h0;end

3'b110:begin dot\_row=8'b11111101; dot\_col\_g=8'h0;end

3'b111:begin dot\_row=8'b11111110; dot\_col\_g=8'h0;end

endcase

end

3'b010:

begin

case(cnt1)

3'b000:begin dot\_row=8'b01111111; dot\_col\_g=8'b0;end

3'b001:begin dot\_row=8'b10111111; dot\_col\_g=8'b0;end

3'b010:begin dot\_row=8'b11011111; dot\_col\_g=8'b01111110;end

3'b011:begin dot\_row=8'b11101111; dot\_col\_g=8'b01111110;end

3'b100:begin dot\_row=8'b11110111; dot\_col\_g=8'b01111110;end

3'b101:begin dot\_row=8'b11111011; dot\_col\_g=8'h0;end

3'b110:begin dot\_row=8'b11111101; dot\_col\_g=8'h0;end

3'b111:begin dot\_row=8'b11111110; dot\_col\_g=8'h0;end

endcase

end

3'b011:

begin

case(cnt1)

3'b000:begin dot\_row=8'b01111111; dot\_col\_g=8'b0;end

3'b001:begin dot\_row=8'b10111111; dot\_col\_g=8'b0;end

3'b010:begin dot\_row=8'b11011111; dot\_col\_g=8'b0;end

3'b011:begin dot\_row=8'b11101111; dot\_col\_g=8'b01111110;end

3'b100:begin dot\_row=8'b11110111; dot\_col\_g=8'b01111110;end

3'b101:begin dot\_row=8'b11111011; dot\_col\_g=8'b01111110;end

3'b110:begin dot\_row=8'b11111101; dot\_col\_g=8'h0;end

3'b111:begin dot\_row=8'b11111110; dot\_col\_g=8'h0;end

endcase

end

3'b100:

begin

case(cnt1)

3'b000:begin dot\_row=8'b01111111; dot\_col\_g=8'b0;end

3'b001:begin dot\_row=8'b10111111; dot\_col\_g=8'b0;end

3'b010:begin dot\_row=8'b11011111; dot\_col\_g=8'b0;end

3'b011:begin dot\_row=8'b11101111; dot\_col\_g=8'b0;end

3'b100:begin dot\_row=8'b11110111; dot\_col\_g=8'b01111110;end

3'b101:begin dot\_row=8'b11111011; dot\_col\_g=8'b01111110;end

3'b110:begin dot\_row=8'b11111101; dot\_col\_g=8'b01111110;end

3'b111:begin dot\_row=8'b11111110; dot\_col\_g=8'h0;end

endcase

end

3'b101:

begin

case(cnt1)

3'b000:begin dot\_row=8'b01111111; dot\_col\_g=8'b0;end

3'b001:begin dot\_row=8'b10111111; dot\_col\_g=8'b0;end

3'b010:begin dot\_row=8'b11011111; dot\_col\_g=8'b0;end

3'b011:begin dot\_row=8'b11101111; dot\_col\_g=8'b0;end

3'b100:begin dot\_row=8'b11110111; dot\_col\_g=8'b0;end

3'b101:begin dot\_row=8'b11111011; dot\_col\_g=8'b01111110;end

3'b110:begin dot\_row=8'b11111101; dot\_col\_g=8'b01111110;end

3'b111:begin dot\_row=8'b11111110; dot\_col\_g=8'b01111110;end

endcase

end

3'b110:

begin

case(cnt1)

3'b000:begin dot\_row=8'b01111111; dot\_col\_g=8'b01111110;end

3'b001:begin dot\_row=8'b10111111; dot\_col\_g=8'b0;end

3'b010:begin dot\_row=8'b11011111; dot\_col\_g=8'b0;end

3'b011:begin dot\_row=8'b11101111; dot\_col\_g=8'b0;end

3'b100:begin dot\_row=8'b11110111; dot\_col\_g=8'b0;end

3'b101:begin dot\_row=8'b11111011; dot\_col\_g=8'b0;end

3'b110:begin dot\_row=8'b11111101; dot\_col\_g=8'b01111110;end

3'b111:begin dot\_row=8'b11111110; dot\_col\_g=8'b01111110;end

endcase

end

3'b111:

begin

case(cnt1)

3'b000:begin dot\_row=8'b01111111; dot\_col\_g=8'b01111110;end

3'b001:begin dot\_row=8'b10111111; dot\_col\_g=8'b01111110;end

3'b010:begin dot\_row=8'b11011111; dot\_col\_g=8'b0;end

3'b011:begin dot\_row=8'b11101111; dot\_col\_g=8'b0;end

3'b100:begin dot\_row=8'b11110111; dot\_col\_g=8'b0;end

3'b101:begin dot\_row=8'b11111011; dot\_col\_g=8'b0;end

3'b110:begin dot\_row=8'b11111101; dot\_col\_g=8'b0;end

3'b111:begin dot\_row=8'b11111110; dot\_col\_g=8'b01111110;end

endcase

end

endcase

end

3'b010:

begin dot\_row=8'hff; dot\_col=8'b0;

case(cnt2[0])

0:

case(cnt1[2:0])

3'b000:begin dot\_row=8'b11111110; dot\_col\_g=8'b0011\_1100;end

3'b001:begin dot\_row=8'b11111101; dot\_col\_g=8'b0111\_1110;end

3'b010:begin dot\_row=8'b11111011; dot\_col\_g=8'b1111\_1111;end

3'b011:begin dot\_row=8'b11110111; dot\_col\_g=8'b1110\_0111;end

3'b100:begin dot\_row=8'b11101111; dot\_col\_g=8'b1110\_0111;end

3'b101:begin dot\_row=8'b11011111; dot\_col\_g=8'b1111\_1111;end

3'b110:begin dot\_row=8'b10111111; dot\_col\_g=8'b0111\_1110;end

3'b111:begin dot\_row=8'b01111111; dot\_col\_g=8'b0011\_1100;end

endcase

1:

case(cnt1[2:0])

3'b000:begin dot\_row=8'b11111110; dot\_col\_g=8'b0001\_1000;end

3'b001:begin dot\_row=8'b11111101; dot\_col\_g=8'b0001\_1000;end

3'b010:begin dot\_row=8'b11111011; dot\_col\_g=8'b0001\_1000;end

3'b011:begin dot\_row=8'b11110111; dot\_col\_g=8'b0001\_1000;end

3'b100:begin dot\_row=8'b11101111; dot\_col\_g=8'b0001\_1000;end

3'b101:begin dot\_row=8'b11011111; dot\_col\_g=8'b0001\_1000;end

3'b110:begin dot\_row=8'b10111111; dot\_col\_g=8'b0001\_1000;end

3'b111:begin dot\_row=8'b01111111; dot\_col\_g=8'b0001\_1000;end

endcase

endcase

end

3'b001:

begin dot\_col\_g=8'b0;dot\_col\_g=8'b0;

case(cnt2)

3'b000:

case(cnt1)

3'b000:begin dot\_row=8'b01111111; dot\_col=8'b0;end //E R

3'b001:begin dot\_row=8'b10111111; dot\_col=8'b1110\_1100;end //

3'b010:begin dot\_row=8'b11011111; dot\_col=8'b1000\_1010;end

3'b011:begin dot\_row=8'b11101111; dot\_col=8'b1110\_1110;end

3'b100:begin dot\_row=8'b11110111; dot\_col=8'b1000\_1100;end

3'b101:begin dot\_row=8'b11111011; dot\_col=8'b1110\_1010;end

3'b110:begin dot\_row=8'b11111101; dot\_col=8'b0;end

3'b111:begin dot\_row=8'b11111110; dot\_col=8'b0;end

endcase

3'b001:

case(cnt1)

3'b000:begin dot\_row=8'b01111111; dot\_col=8'b0;end //E R

3'b001:begin dot\_row=8'b10111111; dot\_col=8'b1110\_1100;end //

3'b010:begin dot\_row=8'b11011111; dot\_col=8'b1000\_1010;end

3'b011:begin dot\_row=8'b11101111; dot\_col=8'b1110\_1110;end

3'b100:begin dot\_row=8'b11110111; dot\_col=8'b1000\_1100;end

3'b101:begin dot\_row=8'b11111011; dot\_col=8'b1110\_1010;end

3'b110:begin dot\_row=8'b11111101; dot\_col=8'b0;end

3'b111:begin dot\_row=8'b11111110; dot\_col=8'b0;end

endcase

3'b010:

case(cnt1)

3'b000:begin dot\_row=8'b01111111; dot\_col=8'b0;end //R O

3'b001:begin dot\_row=8'b10111111; dot\_col=8'b1100\_0110;end //

3'b010:begin dot\_row=8'b11011111; dot\_col=8'b1010\_1001;end

3'b011:begin dot\_row=8'b11101111; dot\_col=8'b1110\_1001;end

3'b100:begin dot\_row=8'b11110111; dot\_col=8'b1100\_1001;end

3'b101:begin dot\_row=8'b11111011; dot\_col=8'b1010\_0110;end

3'b110:begin dot\_row=8'b11111101; dot\_col=8'b0;end

3'b111:begin dot\_row=8'b11111110; dot\_col=8'b0;end

endcase

3'b011:

case(cnt1)

3'b000:begin dot\_row=8'b01111111; dot\_col=8'b0;end //R O

3'b001:begin dot\_row=8'b10111111; dot\_col=8'b1100\_0110;end //

3'b010:begin dot\_row=8'b11011111; dot\_col=8'b1010\_1001;end

3'b011:begin dot\_row=8'b11101111; dot\_col=8'b1110\_1001;end

3'b100:begin dot\_row=8'b11110111; dot\_col=8'b1100\_1001;end

3'b101:begin dot\_row=8'b11111011; dot\_col=8'b1010\_0110;end

3'b110:begin dot\_row=8'b11111101; dot\_col=8'b0;end

3'b111:begin dot\_row=8'b11111110; dot\_col=8'b0;end

endcase

3'b100:

case(cnt1)

3'b000:begin dot\_row=8'b01111111; dot\_col=8'b0;end //R

3'b001:begin dot\_row=8'b10111111; dot\_col=8'b1100\_0000;end //

3'b010:begin dot\_row=8'b11011111; dot\_col=8'b1010\_0000;end

3'b011:begin dot\_row=8'b11101111; dot\_col=8'b1110\_0000;end

3'b100:begin dot\_row=8'b11110111; dot\_col=8'b1100\_0000;end

3'b101:begin dot\_row=8'b11111011; dot\_col=8'b1010\_0000;end

3'b110:begin dot\_row=8'b11111101; dot\_col=8'b0;end

3'b111:begin dot\_row=8'b11111110; dot\_col=8'b0;end

endcase

3'b101:

case(cnt1)

3'b000:begin dot\_row=8'b01111111; dot\_col=8'b0;end //R

3'b001:begin dot\_row=8'b10111111; dot\_col=8'b1100\_0000;end //

3'b010:begin dot\_row=8'b11011111; dot\_col=8'b1010\_0000;end

3'b011:begin dot\_row=8'b11101111; dot\_col=8'b1110\_0000;end

3'b100:begin dot\_row=8'b11110111; dot\_col=8'b1100\_0000;end

3'b101:begin dot\_row=8'b11111011; dot\_col=8'b1010\_0000;end

3'b110:begin dot\_row=8'b11111101; dot\_col=8'b0;end

3'b111:begin dot\_row=8'b11111110; dot\_col=8'b0;end

endcase

3'b110:

case(cnt1)

3'b000:begin dot\_row=8'b01111111; dot\_col=8'b0;end //空白不显示

3'b001:begin dot\_row=8'b10111111; dot\_col=8'b0;end //

3'b010:begin dot\_row=8'b11011111; dot\_col=8'b0;end

3'b011:begin dot\_row=8'b11101111; dot\_col=8'b0;end

3'b100:begin dot\_row=8'b11110111; dot\_col=8'b0;end

3'b101:begin dot\_row=8'b11111011; dot\_col=8'b0;end

3'b110:begin dot\_row=8'b11111101; dot\_col=8'b0;end

3'b111:begin dot\_row=8'b11111110; dot\_col=8'b0;end

endcase

3'b111:

case(cnt1)

3'b000:begin dot\_row=8'b01111111; dot\_col=8'b0;end //空白不显示

3'b001:begin dot\_row=8'b10111111; dot\_col=8'b0;end //

3'b010:begin dot\_row=8'b11011111; dot\_col=8'b0;end

3'b011:begin dot\_row=8'b11101111; dot\_col=8'b0;end

3'b100:begin dot\_row=8'b11110111; dot\_col=8'b0;end

3'b101:begin dot\_row=8'b11111011; dot\_col=8'b0;end

3'b110:begin dot\_row=8'b11111101; dot\_col=8'b0;end

3'b111:begin dot\_row=8'b11111110; dot\_col=8'b0;end

endcase

endcase

end

endcase

end

else begin dot\_row=8'hff; dot\_col=8'b0;dot\_col\_g=8'b0;dot\_col\_g=8'b0;end

end

always@(posedge clk) //控制动画的延时

begin

if(CNT==DTIME)

cnt2<=cnt2+1;

if(signal\_out)

CNT<=CNT+1;

else

begin

CNT<=0;

cnt2<=0;

end

end

always@(posedge clk or posedge rst) // dot输入频率1kHZ cnt1 八个状态控制刷新频率

begin

if(rst)

cnt1<=3'b000;

else

cnt1<=cnt1+1'b1;

end

endmodule

4.7 数码管模块

module seg(clk,rst,signal\_exam,signal\_in,result\_signal,ticket\_seg,station\_10\_seg,station\_1\_seg,money\_hundred\_seg,money\_ten\_seg,money\_one\_seg,change\_money\_hundred\_seg,change\_money\_ten\_seg,change\_money\_one\_seg,seg\_7,seg\_num);

input clk;

input rst;

input signal\_in; //输入状态使能信号

input signal\_exam; //自检使能

input result\_signal; //结果使能

input [7:0]ticket\_seg,station\_10\_seg,station\_1\_seg,money\_hundred\_seg,money\_ten\_seg,money\_one\_seg;

input [7:0]change\_money\_hundred\_seg,change\_money\_ten\_seg,change\_money\_one\_seg;

output reg[7:0] seg\_7;

output reg [7:0] seg\_num;

reg [2:0]cnt1=3'b0; //八个状态转移

always@(cnt1)

begin

if(rst)

begin

seg\_7<=8'b0; seg\_num<=8'hff;

end

else if(signal\_exam)

begin

case(cnt1)

3'b000:begin seg\_7<=8'hff; seg\_num<=8'b11111110;end

3'b001:begin seg\_7<=8'hff; seg\_num<=8'b11111101;end

3'b010:begin seg\_7<=8'hff; seg\_num<=8'b11111011;end

3'b011:begin seg\_7<=8'hff; seg\_num<=8'b11110111;end

3'b100:begin seg\_7<=8'hff; seg\_num<=8'b11101111;end

3'b101:begin seg\_7<=8'hff; seg\_num<=8'b11011111;end

3'b110:begin seg\_7<=8'hff; seg\_num<=8'b10111111;end

3'b111:begin seg\_7<=8'hff; seg\_num<=8'b01111111;end

default:;

endcase

end

else if(signal\_in)

case(cnt1)

3'b000:begin seg\_7<=ticket\_seg; seg\_num<=8'b11111110;end

3'b001:begin seg\_7<=8'b0; seg\_num<=8'b11111101;end

3'b010:begin seg\_7<=station\_1\_seg; seg\_num<=8'b11111011;end

3'b011:begin seg\_7<=station\_10\_seg; seg\_num<=8'b11110111;end

3'b100:begin seg\_7<=8'b0; seg\_num<=8'b11101111;end

3'b101:begin seg\_7<=money\_one\_seg; seg\_num<=8'b11011111;end

3'b110:begin seg\_7<=money\_ten\_seg; seg\_num<=8'b10111111;end

3'b111:begin seg\_7<=money\_hundred\_seg; seg\_num<=8'b01111111;end

default:;

endcase

else if(result\_signal)

case(cnt1)

3'b000:begin seg\_7<=change\_money\_one\_seg; seg\_num<=8'b11111110;end

3'b001:begin seg\_7<=change\_money\_ten\_seg; seg\_num<=8'b11111101;end

3'b010:begin seg\_7<=change\_money\_hundred\_seg; seg\_num<=8'b11111011;end

3'b011:begin seg\_7<=8'b0; seg\_num<=8'b11110111;end

3'b100:begin seg\_7<=8'b0; seg\_num<=8'b11101111;end

3'b101:begin seg\_7<=8'b0; seg\_num<=8'b11011111;end

3'b110:begin seg\_7<=8'b0; seg\_num<=8'b10111111;end

3'b111:begin seg\_7<=8'b0; seg\_num<=8'b01111111;end

default:;

endcase

else

begin seg\_7<=8'b0; seg\_num<=8'hff;end

end

always@(posedge clk or posedge rst) // 1/20ms 50hz 000-111

begin

if(rst)

cnt1<=3'b000;

else

cnt1<=cnt1+1;

end

endmodule

4.8 按键消抖模块

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// >>>>>>>>>>>>>>>>>>>>>>>>> COPYRIGHT NOTICE <<<<<<<<<<<<<<<<<<<<<<<<<

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// File name : debounce.v

// Module name : debounce

// Author : STEP

// Description :

// Web : www.stepfpga.com

//

// --------------------------------------------------------------------

// Code Revision History :

// --------------------------------------------------------------------

// Version: |Mod. Date: |Changes Made:

// V1.0 |2017/03/02 |Initial ver

// --------------------------------------------------------------------

// Module Function:按键消抖

module debounce (clk,rst,key,key\_pulse);

parameter N = 1; //要消除的按键的数量

reg [19:0]MAX=20'hf4240; // 50MHz

input clk;

input rst;

input [N-1:0] key; //输入的按键

output [N-1:0] key\_pulse; //按键动作产生的脉冲

reg [N-1:0] key\_rst\_pre= {N{1'b0}}; //定义一个寄存器型变量存储上一个触发时的按键值

reg [N-1:0] key\_rst= {N{1'b0}}; //定义一个寄存器变量储存储当前时刻触发的按键值

wire [N-1:0] key\_edge; //检测到按键由低到高变化是产生一个高脉冲

//利用非阻塞赋值特点，将两个时钟触发时按键状态存储在两个寄存器变量中

always @(posedge clk or posedge rst)

begin

if (rst) begin

key\_rst <= {N{1'b0}}; //初始化时给key\_rst赋值全为0，{}中表示N个0

key\_rst\_pre <= {N{1'b0}};

end

else begin

key\_rst <= key; //第一个时钟上升沿触发之后key的值赋给key\_rst,同时key\_rst的值赋给key\_rst\_pre

key\_rst\_pre <= key\_rst; //非阻塞赋值。相当于经过两个时钟触发，key\_rst存储的是当前时刻key的值，key\_rst\_pre存储的是前一个时钟的key的值

end

end

assign key\_edge = key\_rst & (~key\_rst\_pre);//脉冲边沿检测。当key检测到上升沿时，key\_edge产生一个时钟周期的高电平

reg [19:0] cnt; //产生延时所用的计数器，系统时钟50Mhz，要延时20ms左右时间，至少需要18位计数器

//产生20ms延时，当检测到key\_edge有效时计数器清零开始计数

always @(posedge clk or posedge rst)

begin

if(rst)

cnt <= 18'h0;

else if(key\_edge)

cnt <= 18'h0;

else

cnt <= cnt + 1'h1;

end

reg [N-1:0] key\_sec\_pre= {N{1'b0}}; //延时后检测电平寄存器变量

reg [N-1:0] key\_sec= {N{1'b0}};

//延时后检测key，如果按键状态变高产生一个时钟的高脉冲。如果按键状态是低的话说明按键无效

always @(posedge clk or posedge rst)

begin

if (rst)

key\_sec <= {N{1'b0}};

else if (cnt==20'hf4240) //f4240

key\_sec <= key;

end

always @(posedge clk or posedge rst)

begin

if (rst)

key\_sec\_pre <= {N{1'b0}};

else

key\_sec\_pre <= key\_sec;

end

assign key\_pulse = key\_sec & (~key\_sec\_pre);

endmodule

4.9 蜂鸣器模块

module beep(clk\_50M,rst,tone\_en,signal,speaker);

input clk\_50M,rst; //clk\_in为50Mhz

input [2:0]signal;// 不同的输出结果不同的音乐; 因为音乐与动画对应 所以可用同一个信号。

input tone\_en; //使能信号。 此为 signal\_out 代表进入了输出结果的状态

output reg speaker=0; //蜂鸣器输出。

reg [4:0]tone=5'd12; // 此处 的tone 是内置的 所以不需输入

reg [4:0]cnt\_05s=5'b0;

parameter wide=15;

reg[7:0] cnt=0; //音名数

reg[3:0] cnt1=4'b0; //5MHz基频

reg[23:0] cnt2=0;//节拍频率5Hz

reg[wide-1:0] origin=0;//预置数寄存器

reg[wide-1:0] drive=0;

reg[1:0] count=0;

reg carrier=0;

//分频产生5MHz和5Hz的频率

always @(posedge clk\_50M)

begin

if(rst)

begin

cnt1<=4'd0;

cnt2<=24'd0;

end

else if(!tone\_en)

begin

cnt1<=4'd0;

cnt2<=24'd0;

end

else

begin

cnt1<=cnt1+1'b1;

cnt2<=cnt2+1'b1;

if(cnt1==4'd9)

cnt1<=4'd0;

if(cnt2==24'h98967F)

cnt2<=24'd0;

end

end

always @(posedge clk\_50M)

begin

if(rst)

drive<=15'h0;

else if(cnt1==4'd9)

begin

if(drive==15'h7fff)

begin

drive<=origin;

carrier<=1'b1;

end

else begin drive<=drive+1'b1;carrier<=1'b0; end

end

end

//carrier的频率是每个音阶的频率

always @(posedge carrier)

begin

count<=count+1'b1;

if(count==4'd0)

speaker<=1'b1;

else speaker<=1'b0;

end

always @(posedge clk\_50M,posedge rst)

begin

if(rst)

begin

origin<=15'h0;

cnt<=8'd0;

end

else if(cnt2==24'h98967F)

begin

if(cnt==8'd139)

cnt<=8'd0;

else

cnt<=cnt+1'b1;

if(tone\_en)

begin

if(signal==3'b100)

case (cnt)

8'd0:origin<=15'h625F; //中音3，4个节拍

8'd1:origin<=15'h625F;

8'd2:origin<=15'h625F;

8'd3:origin<=15'h625F;

8'd4:origin<=15'h6715; //中音5,3个节拍

8'd5:origin<=15'h6715;

8'd6:origin<=15'h6715;

8'd7:origin<=15'h69cd;//中音6

8'd8:origin<=15'h6d55; //高音1，3个节拍

8'd9:origin<=15'h6d55;

8'd10:origin<=15'h6d55;

8'd11:origin<=15'h6f5f; //高音2

8'd12:origin<=15'h69cd; //中音6

8'd13:origin<=15'h6d55; //高音1

8'd14:origin<=15'h6715; //中音5

8'd15:origin<=15'h6715;

8'd16:origin<=15'h738a; //高音5

8'd17:origin<=15'h738a;

8'd18:origin<=15'h738a;

8'd19:origin<=15'h76aa; //倍高音1

8'd20:origin<=15'h69cd; //高音6

8'd21:origin<=15'h6715;//高音5

8'd22:origin<=15'h712f;//高音3

8'd23:origin<=15'h6715;//高音5

8'd24:origin<=15'h6f5f; //高音2

8'd25:origin<=15'h6f5f;

8'd26:origin<=15'h6f5f;

8'd27:origin<=15'h6f5f;

8'd28:origin<=15'h6f5f;

8'd29:origin<=15'h6f5f;

8'd30:origin<=15'h6f5f;

8'd31:origin<=15'h6f5f;

8'd32:origin<=15'h6f5f;

8'd33:origin<=15'h6f5f;

8'd34:origin<=15'h6f5f;

8'd35:origin<=15'h712f;//高音3

8'd36:origin<=15'h6c39; //中音7

8'd37:origin<=15'h6c39;

8'd38:origin<=15'h69cd;//中音6

8'd39:origin<=15'h69cd;

8'd40:origin<=15'h6715; //中音5

8'd41:origin<=15'h6715;

8'd42:origin<=15'h6715;

8'd43:origin<=15'h69cd;//中音6

8'd44:origin<=15'h6d55;//高音1

8'd45:origin<=15'h6d55;

8'd46:origin<=15'h6f5f;//高音2

8'd47:origin<=15'h6f5f;

8'd48:origin<=15'h625f;//中音3

8'd49:origin<=15'h625f;

8'd50:origin<=15'h6d55; //高音1

8'd51:origin<=15'h6d55;

8'd52:origin<=15'h69cd;//中音6

8'd53:origin<=15'h6715;//中音5

8'd54:origin<=15'h69cd; //中音6

8'd55:origin<=15'h6d55;//高音1

8'd56:origin<=15'h6715;//中音5

8'd57:origin<=15'h6715;

8'd58:origin<=15'h6715;

8'd59:origin<=15'h6715;

8'd60:origin<=15'h6715;

8'd61:origin<=15'h6715;

8'd62:origin<=15'h6715;

8'd63:origin<=15'h6715;

8'd64:origin<=15'h712f;//高音3

8'd65:origin<=15'h712f;

8'd66:origin<=15'h712f;

8'd67:origin<=15'h738a;//高音5

8'd68:origin<=15'h6c39;//中音7

8'd69:origin<=15'h6c39;

8'd70:origin<=15'h6f5f;//高音2

8'd71:origin<=15'h6f5f;

8'd72:origin<=15'h69cd; //中音6

8'd73:origin<=15'h6d55;//高音1

8'd74:origin<=15'h6715;//中音5

8'd75:origin<=15'h6715;

8'd76:origin<=15'h6715;

8'd77:origin<=15'h6715;

8'd78:origin<=15'h6715;

8'd79:origin<=15'h6715;

8'd80:origin<=15'h625f; //中音3

8'd81:origin<=15'h6715;//中音5

8'd82:origin<=15'h625f;//中音3

8'd83:origin<=15'h625f;

8'd84:origin<=15'h6715;//中音5

8'd85:origin<=15'h69cd;//中音6

8'd86:origin<=15'h6c39;//中音7

8'd87:origin<=15'h6f5f;//高音2

8'd88:origin<=15'h69cd;//中音6

8'd89:origin<=15'h69cd;

8'd90:origin<=15'h69cd;

8'd91:origin<=15'h69cd;

8'd92:origin<=15'h69cd;

8'd93:origin<=15'h69cd;

8'd94:origin<=15'h6715;//中音5

8'd95:origin<=15'h69cd;//中音6

8'd96:origin<=15'h6d55;//高音1

8'd97:origin<=15'h6d55;

8'd98:origin<=15'h6d55;

8'd99:origin<=15'h6f5f;////高音2

8'd100:origin<=15'h738a; //高音5

8'd101:origin<=15'h738a;

8'd102:origin<=15'h738a;

8'd103:origin<=15'h712f;//高音3

8'd104:origin<=15'h6f5f;//高音2

8'd105:origin<=15'h6f5f;

8'd106:origin<=15'h712f;//高音3

8'd107:origin<=15'h6f5f;//高音2

8'd108:origin<=15'h6d55;//高音1

8'd109:origin<=15'h6d55;

8'd110:origin<=15'h69cd;//中音6

8'd111:origin<=15'h6715;//中音5

8'd112:origin<=15'h625f;//中音3

8'd113:origin<=15'h625f;

8'd114:origin<=15'h625f;

8'd115:origin<=15'h625f;

8'd116:origin<=15'h6d55;//高音1

8'd117:origin<=15'h6d55;

8'd118:origin<=15'h69cd;//中音6

8'd119:origin<=15'h6d55;//高音1

8'd120:origin<=15'h69cd;//中音6

8'd121:origin<=15'h625f;//中音3

8'd122:origin<=15'h625f;

8'd123:origin<=15'h6f5f;//高音2

8'd124:origin<=15'h625f;//中音3

8'd125:origin<=15'h6715;//中音5

8'd126:origin<=15'h69cd;//中音6

8'd127:origin<=15'h6d55;//高音1

8'd128:origin<=15'h6715;//中音5

8'd129:origin<=15'h6715;

8'd130:origin<=15'h6715;

8'd131:origin<=15'h6715;

8'd132:origin<=15'h6715;

8'd133:origin<=15'h6715;

8'd134:origin<=15'h6715;

8'd135:origin<=15'h6715;

8'd136:origin<=15'h3fff;

8'd137:origin<=15'h3fff;

8'd138:origin<=15'h3fff;

8'd139:origin<=15'h3fff;

default:origin<=15'h0;

endcase

else if(signal==3'b010)

case(cnt[1:0])

2'b01:origin<=origin<=15'h69cd;//中音6

2'b00:origin<=15'h6f5f; //高2

2'b10:origin<=15'h625F; //中音3

2'b11:origin<=15'h625F; //中音3

endcase

else if(signal==3'b001)

case(cnt[1:0])

2'b01:origin<=15'h738a; //高音5

2'b00:origin<=15'h6715; //中 5

2'b10:origin<=15'h6715;

2'b11:origin<=15'h625F; //中音3

endcase

else

origin<=15'h3fff;

end

end

end

//

//always@(posedge clk\_05s)

//begin

//if(tone\_en) //进入结果状态

//cnt\_05s<=cnt\_05s+1;

//else

//cnt\_05s<=0;

//end

//根据输入信号，选择不同

//always@(posedge clk\_in) //动画的时间

//begin

// case(signal) //此处signal 和动画的signal\_oled 相同 100出票 010 找零 001 错误。

// 3'b100:

// case(cnt\_05s)

// 5'd1: tone<=5'd1;

// 5'd2: tone<=5'd1;

// 5'd3: tone<=5'd5;

// 5'd4: tone<=5'd5;

// 5'd5: tone<=5'd6;

// 5'd6: tone<=5'd6;

// 5'd7: tone<=5'd5;

// 5'd8: tone<=5'd4;

// 5'd9: tone<=5'd4;

// 5'd10:tone<=5'd3;

// default:tone<=5'd0;

// endcase

// 3'b010:

// case(cnt\_05s[0])

// 1: tone<=5'd12;

// 0: tone<=5'd4;

// default:tone<=5'd0;

// endcase

// 3'b001:

// case(cnt\_05s[1]) //相当于持续一秒 一个音高

// 1: tone<=5'd19;

// 0: tone<=5'd8;

// default:tone<=5'd0;

// endcase

// endcase

//

//end

endmodule

4.10 分频器模块

module clk\_d(clkin,clkout);

input clkin;

output reg clkout;

integer count=32'b0;//分频计数器，每计数到25000下clk\_1k时钟翻转 翻转2次 为一个周期 故频率为1khz

parameter MAX=25000;

initial

begin

clkout<=0;

end

always @(posedge clkin)//分频进程

begin

if(count==MAX)

begin

count=0;

clkout=~clkout;

end

else

count=count+1;

end

endmodule

五、功能说明及资源利用情况

1.基本功能全部实现，并完成了点阵上的各种提示信息用动画表示和不同的提示配合不同的音效的两个提高要求。

2.出票为绿色票向下滚动图形，找零为转动的硬币，错误为滚动的红色ERROR字母。不同的图形配有不同的音乐。其中若找零钱数为0，即不需要找零，则不会播放找零动画和对应的音乐。

注：在购票失败后，会返回上个输入界面，原先数据全部保留。因为每张票最多14元，所以9张票最多100多元。故默认用户投币不超过200元。

资源占用情况:

Total logic elements 954 / 1,270 ( 75 % )

Total pins 51 / 116 ( 44 % )

因为程序要保留很多预置图像，声音，以及进行百十个位的分离等，所以资源可能占用过高。

六、故障及问题分析

1. 在刚开始时，数码管和点阵无法显示，后来通过仿真发现是分频器的问题，通过改写分频器解决问题。
2. 后来蜂鸣器不工作，通过检查代码发现是外部连接线只定义了一位，故内部的case语句无法选择正确状态，故最终也成功解决。
3. 动画显示不流畅，主要分析延迟时间，通过调分频器参数，来使动画更加流畅。
4. 初期资源占用过大，分析主要是内部用的乘除法来分离钱的个十百位，后来写了新的模块移三位分离位数，使资源占用大幅减小。在代码中模块名称为find\_wei;
5. 状态机的敏感列表要全，并把所以可能情况考虑。
6. 写if条件语句时，尽量后面的用else if，同时注意优先级的问题，否则可能出现逻辑错误。

七、总结和结论

1.通过这次数电实验，个人对这个并行模块的理解更深了，主要是将状态机。将输入和反馈信号引回状态机，即可控制状态的转移。而不同状态有不同的使能信号，来控制不同模块的不同工作状态。这样，只要状态转移，就可以通过信号之间的来回传递，进行时序电路的实现。

2.分频的使用在数电中是最基础的，合适的分频器，可以使实验更加可靠。

3.在FPGA中，尽量不要用乘除运算，因为太占资源。对于分离数的百十个位，可以写模块（移位加五比较法模块）。本实验中就以用到。

4.在实验中，要根据情况，选择合适的复位条件。

5.对一些重复利用的代码，可以写成模块来例化以增加程序的可读性，减少资源占用。

6.在代码中能用case就不要用if语句，来减少资源占用。

7.实验还是要多仿真才可以快速发现问题。