CS207 Digital Design Project ——自动售货机

小组：lab3_4_麻麻我不想写 dsaa 了
小组成员：彭佳然_任艺伟_周民涛(lab4)

1.开发计划：

1.1 任务划分(分工比：1：1：1)：
任艺伟：蜂鸣器控制+查询测试+滚动
周民涛：管理员模式（补货+查询销售量，金额）+顶层设计
彭佳然：所有购买状况及返回
三人一起：七段数码管显示+按键防抖
1.2 进度安排 + 执行记录：·
第一周（2020-12-6 至 2020-12-12）：
安排：讨论自动售货机任务要求，并分配任务，决定将模块分为七位数码管显示信息模块，按键防抖模块，管理员模块，购买模块，查询模块，蜂鸣器模块。并研究决定，利用有限状态机进行购买，管理员，查询三个主要部分之间的转换，并将所有部分的数据规模。标准化，利于顶层设计时，方便代码共享。并决定每隔三天进行开会更新进度及代码。
执行人：任艺伟，彭佳然，周民涛
完成情况：完成√
第二周（2020-12-13 至 2020-12-17）：
安排：三人一起将项目中所涉及的输入输出信息确定，比特位数确定，时钟信号分频，确定相关端口及按键绑定。各自领取上周分配任务开始设计子模块，开始攻关：
具体完成情况：任艺伟：初始化后的商品信息滚动及游客查询货道及对应的商品。
周民涛：管理员模式下的显示最大补货量，补货，及查询补货结果。
彭佳然：对购买情况进行分类：未选择货物，货物剩余数量为 0，付款金额不足，付款超时。其中包括如何去记录已付款金额，并返回余额几部分。完成未选择货物，货物余量为 0，记录已付款金额部分。
第二周周末（2020-12-18 至 2020-12-20）：
安排：开会决定将各自剩余部分完成，并进行子模块检验调试。
具体完成情况：任艺伟：学习蜂鸣器相关原理，并应用到不同情况中去。
周民涛：查询已销售货量和金额。并开始尝试合并三部分代码。
彭佳然：剩余购买情况进行实现并与蜂鸣器进行代码对接。
第三周（2020-12-21 至 2020-12-24）：
安排：合并代码，将显示信息与七位数码显示管进行绑定，加入按键防抖，并进行调试，更改 bug，进行报告攥写。
完成情况：已完成√。

2、设计：

a、需求分析：
系统功能：项目实现自动售货机的类似操作，可以进行管理员补货，查询售量销售额，可

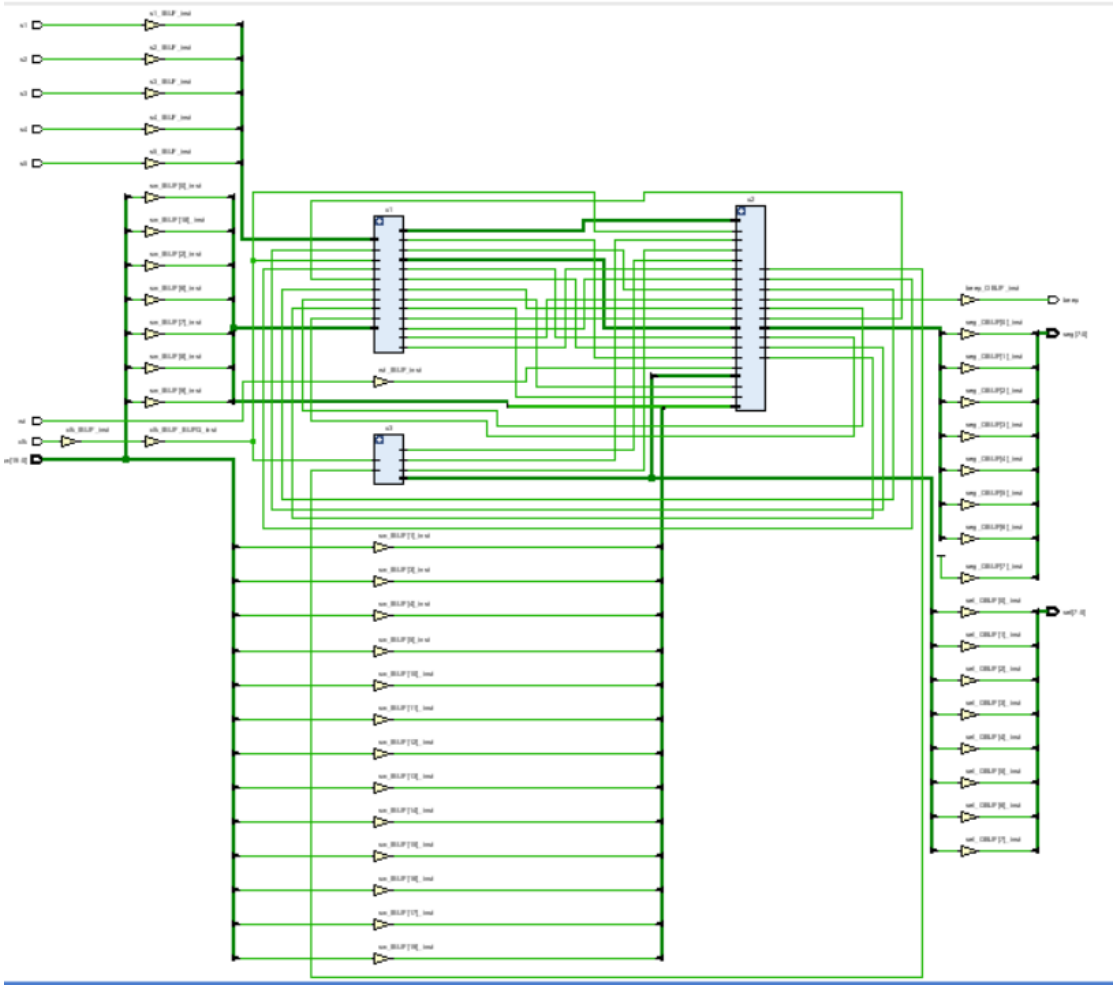
以支持正常买货，并针对不同情形利用蜂鸣器进行声音提示。
输入/输出设备及规格见下图。

```
module auto_machine
(
   input clk,     //100Mhz
   input rst,     //系统复位—s6
   input s1,//确认购买按键
   input s2,//回到查询阶段按键
   input s3,//确认补货按键
   input s4,//投币确认按键
   input s5,//补货确认按键
   input[19:0] sw,//sw[0]货道1，sw[1]:0代表货道1中的第一个商品;1代表货道1中的第二个商品
                  //sw[2]货道2，sw[3]:0代表货道2中的第一个商品;1代表货道2中的第二个商品
                  //sw[4]sw[5],代表选择商品的价格；sw[6]~sw[9],代表投币的金额
                  //sw[10]~sw[17],代表补充商品的数量;sw[18]查看各商品的售出数量;sw[19]查看总销售金额
  output[7:0] sel,seg,
  output  beep//蜂鸣器
);

wire s5_flag,s4_flag,s3_flag,s2_flag,s1_flag;//经过按键防抖处理后的
wire[31:0] dispay_num;//七位数码显示管显示信息
```

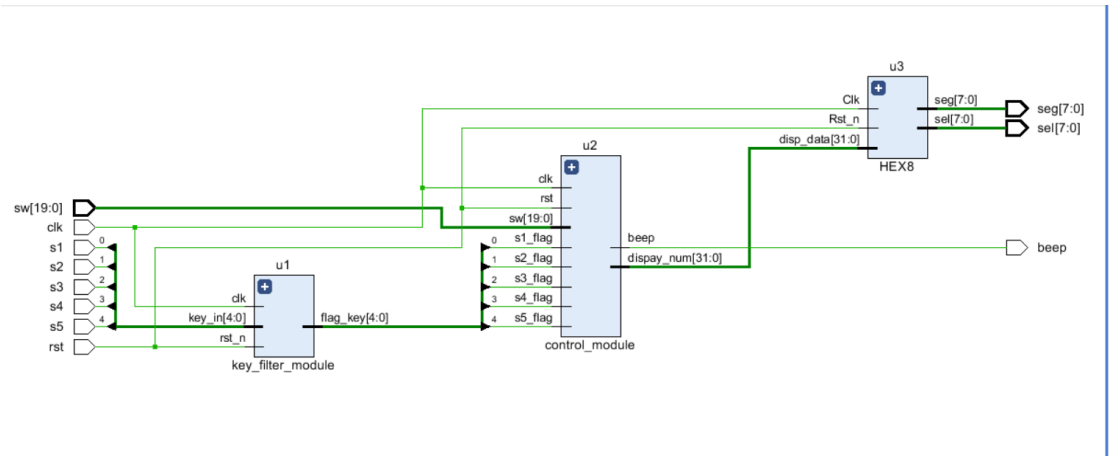
```
module control_module
(
    input clk,     //100Mhz
    input rst,     //系统复位
    input s1_flag,//确认购买按键按下标志
    input s2_flag,//回到查询阶段按键按下标志
    input s3_flag,//确认补货按键按下标志
    input s4_flag,//投币确认按键按下标志
    input s5_flag,//补货确认按键按下标志
    input[19:0] sw,//sw[0]货道1，sw[1]:0代表货道1中的第一个商品;1代表货道1中的第二个商品
                   //sw[2]货道2，sw[3]:0代表货道2中的第一个商品;1代表货道2中的第二个商品
                   //sw[4]sw[5],代表选择商品的价格；sw[6]~sw[9],代表投币的金额
                   //sw[10]~sw[17],代表补充商品的数量;sw[18]查看各商品的售出数量;sw[19]查看总销售金额
   output reg [31:0] dispay_num,
   output reg beep//蜂鸣器
);

reg buy_judge;

parameter S2=2'b00,S5=2'b01,S7=2'b10,S10=2'b11;
parameter M1=4'b0001,M5=4'b0010,M10=4'b0100,M20=4'b1000;
parameter query=2'b00,payment=2'b01,replenish=2'b10;
```

```
reg [3:0] price; //商品的价格
reg [7:0] price_all; //投币的总金额
reg [7:0] price_givechange; //找零的金额
reg [7:0] S2_num,S5_num,S7_num,S10_num;//各个商品的数量
reg [7:0] S2_sell,S5_sell,S7_sell,S10_sell;//各个商品售出的数量
reg move_flag;//滚动显示标志位
wire [23:0] display_move;//查询状态显示的当前滚动信息
reg [23:0] display_huodao;//显示货道的商品信息
reg [95:0] di wire [23:0] all;////查询状态显示的所有滚动信息
reg [29:0] Hz_cnt,Hz_cnt1; //1s计数
reg clk_1Hz,clk_2Hz,en_cnt;
reg [1:0] state;
reg [2:0] payment_flag,replenish_flag;
reg [7:0] time_payment,beep_cnt;
reg [15:0] price_sum;//销售总额
reg [16:0] time_cnt;

assign display_move = display_moveall[95:72];
```

b、系统结构图



RTL 分析:



子模块功能：

共 4 个 module，key_filiter_module 对应按键防抖模块；

control_module 对应有限状态机，三个部分之间的控制模块

HEX8 将控制模块传入的显示信息进行显示

auto_machine 顶层设计，将三个模块合在一起

模块间的接口关联：如下图：

auto_machine module:

```
//按键消抖模块
key_filter_module u1
(
    .clk(clk),              // 开发板上输入时钟：50Mhz
    .rst_n(rst),            // 开发板上输入复位按键
    .key_in({s5,s4,s3,s2,s1}),  // 输入按键信号
    .flag_key({s5_flag,s4_flag,s3_flag,s2_flag,s1_flag})// 输出一个时钟的高电平
);
//控制模块
control_module u2
(
    .clk(clk),    //100Mhz
    .rst(rst),    //系统复位
    .s1_flag(s1_flag),
    .s2_flag(s2_flag),
    .s3_flag(s3_flag),
    .s4_flag(s4_flag),
    .s5_flag(s5_flag),//s1-s5按键
    .sw(sw),//输入
    .dispay_num(dispay_num),//得到当前情况要显示的信息
    .beep(beep)//蜂鸣器控制端口
);
```

```
//数码管显示模块
HEX8 u3
(
    .Clk(clk),
    .Rst_n(rst),
    .disp_data(dispay_num),//控制模块对应当前情形下要显示的信息
    .sel(sel),
    .seg(seg)
);
```

c.系统执行流程:

1、数据流向:

由 RTL 分析可知，利用拨码开关和五个按键，按键经过防抖模块形成可以输出一个高电平的信号，传入 control module 中在这里面通过有限状态机，进入 query replenish payment 三个子模块（最后代码合并时合在了一起）得到要显示的七位数码管信息传入专门用于显示的 HEX8 module 中去进行显示，同时蜂鸣器根据不同状态发出声音。

2、状态转移图:

在 control_module 中，对应三个状态，query replenish payment,而 S1,S2,S3,S4,S5 共 5 个按键用于状态之间的改变，具体如之前的注释，S1 控制查询状态进入购买阶段，S2 各种状态下返回查询阶段，S4 投钱之后确认投币完成购买，仍为 payment 状态需要 S2 返回查询，S3 查询状态进入管理员模式，S5 管理员模式下进行补货确认。

转移图和部分伪代码均呈现在下图中

3、功能伪代码:

在一开始我们决定将代码规范化，均在各自状态下加入参量，对它们进行实时的更新，同时 always 模块敏感,实时实现对 display_num 的更新，这样的话我们不需要各自模块独立，再去绑定，大大减少了后续出错的可能。

要传入显示模块（HEX8）的信息 display_num[32:0] （我们里面的数据利用了 16 进制存储，可以保证货量够多，后续优化也可以增加金额，商品类，故 32 位每四位分别对应七位数码显示管上的一个位置，4*8 = 32）

payment 状态下设置对应参量 payment_flag[2:0] ：

payment_flag = 3'd1 对应商品数量为 0 -> display_num = 全为 2 货量不足

payment_flag = 3'd2 付款金额小于商品价格 ->

display_num = FF(代表金额不足)+已付金额+商品金额+剩余时间

payment_flag = 3'd3 付款成功 ->

display_num = 66(成功)+找零+商品金额+00（付款成功时间归 0）

payment_flag = 3'd4 超出付款时间 ->

display_num = FF(时间超时)+退回已付金额+商品金额+00(时间消耗完)

replenish 状态下设置对应参量 replenish_flag[2:0]

replenish_flag == 3'd1 货道 1 的信息 ->

display_num = 货道一的商品+其中每个商品的最大可补量

replenish_flag == 3'd2 补货后货道 1 的信息

display_num = 补货后后货道一的商品+其中每个商品的最大可补量

replenish_flag == 3'd3 货道 2 的信息

display_num = 货道二的商品+其中每个商品的最大可补量

replenish_flag == 3'd4 补货后货道 2 的信息

display_num = 补货后后货道一的商品+其中每个商品的最大可补量

replenish_flag == 3'd5 查看各商品的售出数量

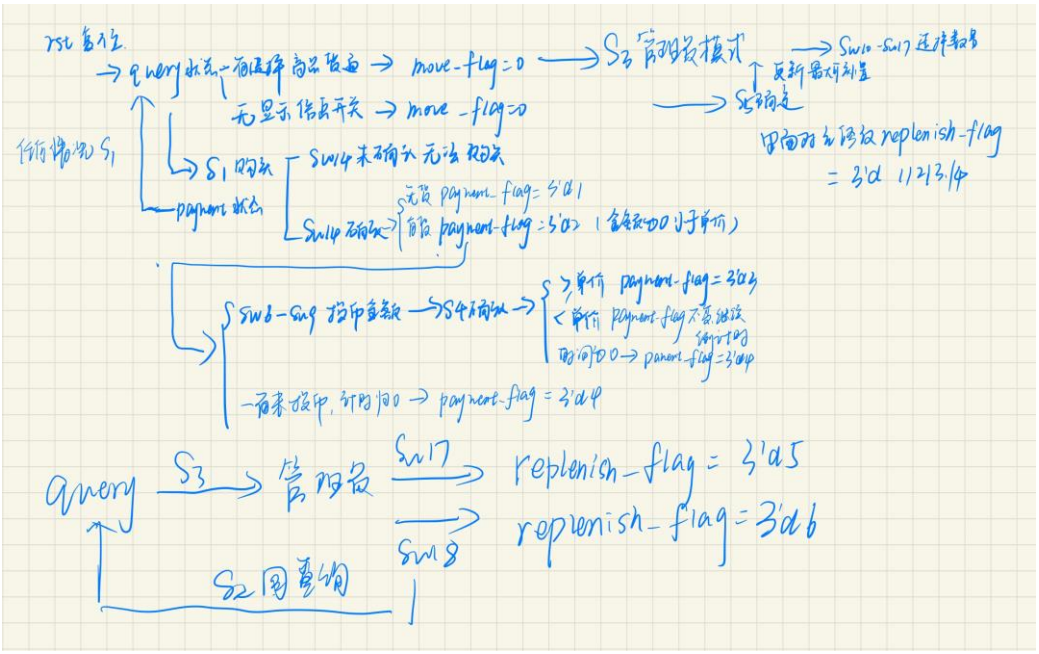
display_num = 每个商品对应的已售数量

replenish_flag == 3'd6 查看总销售金额

display_num = 销售总金额

query 状态下对应一个判断标志 move_flag 去判断是要滚动显示还是显示特定的货道商品 move_flag 为 1 display_num = 滚动显示的信息 不为 1 的话根据之前 sw 里面拨码情况显示当前是哪个商品，显示它的信息。

sw1-sw19 的意义在上面，不再做具体陈述。其余伪代码如下图流程图



d.

```
module auto_machine
(
    input clk,     //100Mhz
    input rst,     //系统复位—s6
    input s1,//确认购买按键
    input s2,//回到查询阶段按键
    input s3,//确认补货按键
    input s4,//投币确认按键
    input s5,//补货确认按键
    input[19:0] sw,//sw[0]货道1，sw[1]:0代表货道1中的第一个商品;1代表货道1中的第二个商品
                   //sw[2]货道2，sw[3]:0代表货道2中的第一个商品;1代表货道2中的第二个商品
                   //sw[4]sw[5],代表选择商品的价格；sw[6]~sw[9],代表投币的金额
                   //sw[10]~sw[17],代表补充商品的数量;sw[18]查看各商品的售出数量;sw[19]查看总销售金额
   output[7:0] sel,seg,
   output  beep//蜂鸣器
);

wire s5_flag,s4_flag,s3_flag,s2_flag,s1_flag;//经过按键防抖处理后的
wire[31:0] dispay_num;//七位数码显示管显示信息
```

```
//按键消抖模块
key_filter_module u1
(
    .clk(clk),                  // 开发板上输入时钟：50Mhz
    .rst_n(rst),                // 开发板上输入复位按键
    .key_in({s5,s4,s3,s2,s1}),  // 输入按键信号
    .flag_key({s5_flag,s4_flag,s3_flag,s2_flag,s1_flag})// 输出一个时钟的高电平
);
//控制模块
control_module u2
(
    .clk(clk),     //100Mhz
    .rst(rst),     //系统复位
    .s1_flag(s1_flag),
    .s2_flag(s2_flag),
    .s3_flag(s3_flag),
    .s4_flag(s4_flag),
    .s5_flag(s5_flag),//s1-s5按键
    .sw(sw),//输入
    .dispay_num(dispay_num),//得到当前情况要显示的信息
    .beep(beep)//蜂鸣器控制端口
);

//数码管显示模块
HEX8 u3
(
    .Clk(clk),
    .Rst_n(rst),
    .disp_data(dispay_num),//控制模块对应当前情形下要显示的信息
    .sel(sel),
    .seg(seg)
);
endmodule
```

control_module:

```
module control_module
(
    input clk,     //100Mhz
    input rst,     //系统复位
    input s1_flag,//确认购买按键按下标志
    input s2_flag,//回到查询阶段按键按下标志
    input s3_flag,//确认补货按键按下标志
    input s4_flag,//投币确认按键按下标志
    input s5_flag,//补货确认按键按下标志
    input[19:0] sw,//sw[0]货道1，sw[1]:0代表货道1中的第一个商品；1代表货道1中的第二个商品
                   //sw[2]货道2，sw[3]:0代表货道2中的第一个商品；1代表货道2中的第二个商品
                   //sw[4]sw[5],代表选择商品的价格；sw[6]~sw[9],代表投币的金额
                   //sw[10]~sw[17],代表补充商品的数量；sw[18]查看各商品的售出数量；sw[19]查看总销售金额
   output reg [31:0] dispay_num,
   output reg beep//蜂鸣器
);

reg buy_judge;
```

```
parameter S2=2'b00,S5=2'b01,S7=2'b10,S10=2'b11;
parameter M1=4'b0001,M5=4'b0010,M10=4'b0100,M20=4'b1000;
parameter query=2'b00,payment=2'b01,replenish=2'b10;

reg [3:0] price; //商品的价格
reg [7:0] price_all; //投币的总金额
reg [7:0] price_givechange; //找零的金额
reg [7:0] S2_num,S5_num,S7_num,S10_num;//各个商品的数量
reg [7:0] S2_sell,S5_sell,S7_sell,S10_sell;//各个商品售出的数量
reg move_flag;//滚动显示标志位
wire [23:0] display_move;//查询状态显示的当前滚动信息
reg [23:0] display_huodao;//显示货道的商品信息
reg [95:0] display_moveall;////查询状态显示的所有滚动信息
reg [29:0] Hz_cnt,Hz_cnt1; //1s计数
reg clk_1Hz,clk_2Hz,en_cnt;
reg [1:0] state;
reg [2:0] payment_flag,replenish_flag;
reg [7:0] time_payment,beep_cnt;
reg [15:0] price_sum;//销售总额
reg [16:0] time_cnt;


assign display_move = display_moveall[95:72];

//购买商品流程—状态机
always@(posedge clk or negedge rst)
begin
    if(!rst)
    begin
        buy_judge <= 0;
        price_all <= 8'd0;
        price_givechange <= 8'd0;
        move_flag <= 1;
        en_cnt <= 0;
        display_huodao <= 24'd0;
        //display_move <= {4'd1,4'd1,S2_num,8'd2};//初始化为第一个货道，第一个商品，剩余数量，商品价格
        display_moveall<={4'd1,4'd1,S2_num,8'h02,4'd1,4'd2,S5_num,8'h05,4'd2,4'd3,S7_num,8'h07,4'd2,4'd4,S10_num,8'h10};
        S2_num <= 8'd0;
        S5_num <= 8'd0;
        S7_num <= 8'd0;
        S10_num <= 8'd0;
        S2_sell <= 8'd0;
        S5_sell <= 8'd0;
        S7_sell <= 8'd0;
        S10_sell <= 8'd0;
        state <= query;
        payment_flag <= 3'b000;
        replenish_flag <= 3'b000;
        time_payment <= 8'h00;
        price_sum <= 16'h0000;
    end
    else begin
        case(state)
            query:
            begin
                display_moveall<={4'd1,4'd1,S2_num,8'h02,4'd1,4'd2,S5_num,8'h05,4'd2,4'd3,S7_num,8'h07,4'd2,4'd4,S10_num,
                if(move_flag)//显示滚动信息
                begin
                    en_cnt <= 1;
                    if(clk_1Hz)
                        display_moveall <= {display_moveall[72:0],display_moveall[95:72]};//循环左移滚动显示商品信息
                    else display_moveall <= display_moveall;
                end
                else en_cnt <= 0;
```

```
                display_huodao <= (!sw[1])?{4'd1,4'd1,S2_num,8'h02}:{4'd1,4'd2,S5_num,8'h05};
            end
            else if(sw[2])
            begin
                move_flag <= 0;
                display_huodao <= (!sw[3])?{4'd2,4'd3,S7_num,8'h07}:{4'd2,4'd4,S10_num,8'h10};
            end
            else move_flag <= 1;

            if(s1_flag)//进入付款阶段
            begin
                state <= payment;
                en_cnt <= 0;
                price_all <= 8'd0;
                buy_judge <= 1'b0;
                time_payment <= 8'h60;//付款设定时间60s
            end
            else if(s3_flag)//进入补货阶段
                state <= replenish;
            else state <= query;
```

```
    payment_flag <= 3'b000;
    replenish_flag <= 3'b000;
end
payment:
begin
    en_cnt <= 1;
    //阶段改变
    if(s2_flag||(((price==8'd2&&S2_num==8'd0)||(price==8'd5&&S5_num==8'd0)||
      (price==8'd7&&S7_num==8'd0)||(price==8'd10&&S10_num==8'd0))&&!buy_judge))//进入查询阶段
    begin
        en_cnt <= 0;
        price_all <= 8'd0;
        if(s2_flag)
        begin
            display_moveall<={4'd1,4'd1,S2_num,8'h02,4'd1,4'd2,S5_num,8'h05,4'd2,4'd3,S7_num,8'h07,4'd2,4'd4,S10_num,8'h10};
            state <= query;
        end
        else begin
            payment_flag <= 3'b001; //数码管显示22222222
            //if(beep_cnt>7)
              //  state <= query;
```

```
    end
end
else state <= payment;

if(price == 8'd0)
begin en_cnt <=1;
payment_flag <= 3'b111;
time_payment <= 8'h00;
end


if(!((price==8'd2&&S2_num==8'd0)||(price==8'd5&&S5_num==8'd0)||
                (price==8'd7&&S7_num==8'd0)||(price==8'd10&&S10_num==8'd0))&&price!=8'd0)begin
//付款倒计时
if(clk_1Hz)
    if(time_payment[3:0] != 0)
        time_payment[3:0] <= time_payment[3:0]-4'd1;
    else begin
        if(time_payment[7:4] != 0)
        begin
            time_payment[3:0] <= 4'h9;
            time_payment[7:4] <= time_payment[7:4]-4'd1;
        end
        else begin
            time_payment[7:4] <= 4'h0;
            if(beep_cnt>7)
               state <= query;
            en_cnt <= 0;
        end
    end
else time_payment <= time_payment;
//投币金额计算
if(s4_flag)
    case({sw[9:6]})
       M1:  //所投钱币+1
         price_all <= price_all + 7'd1;
       M5: //所投钱币+5
         price_all <= price_all + 7'd5;
       M10://所投钱币+10
         price_all <= price_all+ 7'd10;
       M20://所投钱币+20
         price_all <= price_all + 7'd20;
       default:  price_all <= price_all;
    endcase
 else price_all <= price_all;
//付款状态判断
if(time_payment > 0)
begin
     if(price_all < price)
        payment_flag <= 3'b010; //最左侧两个数码管显示AA
     else begin
         buy_judge = 1;
         time_payment <= 8'h00;
         payment_flag <= 3'b011; //最左侧两个数码管显示66
         price_givechange <= (price_all>price)?price_all-price:8'h00;
         case(price)
             8'd2:begin S2_num <= S2_num-8'd1;price_sum <= price_sum+16'd2;S2_sell<=S2_sell+8'd1;end
             8'd5:begin S5_num <= S5_num-8'd1;price_sum <= price_sum+16'd5;S5_sell<=S5_sell+8'd1;end
             8'd7:begin S7_num <= S7_num-8'd1;price_sum <= price_sum+16'd7;S7_sell<=S7_sell+8'd1;end
             8'd10:begin S10_num <= S10_num-8'd1;price_sum <= price_sum+16'd10;S10_sell<=S10_sell+8'd1;end
             default:price_sum <= price_sum;
         endcase
     end
```

```
 end
 else begin
      if(!buy_judge) begin
      payment_flag <= 3'b100;//最左侧两个数码管显示FF
      end
 end
nd
nd
eplenish:
egin
  if(s2_flag)
      state <= query;
  //选择货道进行补货
   else if(sw[0])//补充货道1的商品
   begin
       replenish_flag <= 3'b001;
       if(s5_flag)
       begin
          replenish_flag <= 3'b010;
          if(!sw[1])
              S2_num <= S2_num+sw[17:10];
                  else S5_num <= S5_num+sw[17:10];
             end
       end
       else if(sw[2])//补充货道2的商品
       begin
          replenish_flag <= 3'b011;
           if(s5_flag)
           begin
               replenish_flag <= 3'b100;
               if(!sw[3])
                   S7_num <= S7_num+sw[17:10];
               else S10_num <= S10_num+sw[17:10];
           end
       end
       else if(sw[18])//查看各商品的售出数量
          replenish_flag <= 3'b101;
       else if(sw[19])//查看总销售金额
          replenish_flag <= 3'b110;
       else replenish_flag <= replenish_flag;
   end
   default:begin state <= query;end
```

```
       endcase
   end
end
//选择买哪个商品
always@(posedge clk or negedge rst)
begin
   if(!rst)
       price <= 4'd0;
   else begin
       if(sw[14])//选择商品确认
           case({sw[5],sw[4]})
                S2:begin price <= 8'd2;end
                S5:begin price <= 8'd5;end
                S7:begin price <= 8'd7;end
                S10:begin price <= 8'd10;end
                default:begin price <= 8'd0;end
           endcase
       else price <= 8'd0;//仿真用—后续改成price <= 8'd0
   end
end
```

```
//各种状态蜂鸣器发出不同的提示音time_cnt
always@(posedge clk or negedge rst)
begin
    if(!rst)
    begin
        time_cnt <= 17'd0;
        beep <= 0;
        beep_cnt <= 8'd0;
    end
    else begin
        if(payment_flag == 3'd1)//商品剩余数量为 0
        begin
           if(clk_2Hz)
              beep_cnt <= beep_cnt+8'd1;
           if(time_cnt == 17'd95548)
           begin
              beep <= ~beep;
              time_cnt <= 17'd0;
           end
           else time_cnt <= time_cnt+17'd1;
        end
        else if(payment_flag == 3'd2)//付款金额小于商品金额
           if(time_cnt == 17'd75838)
           begin
              beep <= ~beep;
              time_cnt <= 17'd0;
           end
           else time_cnt <= time_cnt+17'd1;
        else if(payment_flag == 3'd3)//付款成功
           if(time_cnt == 17'd47778)
           begin
              beep <= ~beep;
              time_cnt <= 17'd0;
           end
           else time_cnt <= time_cnt+17'd1;
        else if(payment_flag == 3'd4)//超出付款时间
        begin
           if(clk_2Hz)
              beep_cnt <= beep_cnt+8'd1;
           if(time_cnt == 17'd85136)
           begin
              beep <= ~beep;

              time_cnt <= 17'd0;
           end
           else time_cnt <= time_cnt+17'd1;
        end
        else  begin
            beep_cnt <= 8'd0;
            beep <= 0;
            time_cnt <= 17'd0;
        end
    end
end

//数码管要显示的信息
always@(posedge clk or negedge rst)
begin
    if(!rst)
        dispay_num <= 32'd0;
    else begin
        if(state == query)//查看信息阶段数码管显示信息
            dispay_num <= (move_flag)?{display_move,8'h00}:{display_huodao,8'h00};
        else if(state == payment)
```

```
            if(payment_flag == 3'd1)//商品剩余数量为 0
               dispay_num <= 32'h22222222;
            else if(payment_flag == 3'd2)//付款金额小于商品金额
               dispay_num <= {8'hAA,price_all,price,time_payment};
            else if(payment_flag == 3'd3)//付款成功
               dispay_num <= {8'h66,price_givechange,price,time_payment};
            else if(payment_flag == 3'd4||payment_flag == 3'd7)//超出付款时间
               dispay_num <= {8'hFF,price_all,price,time_payment};
            else  dispay_num <= dispay_num;
        else if(state == replenish)
            if(replenish_flag == 3'd1)//货道1的信息
               dispay_num <= {8'h00,4'h1,8'hff-S2_num,4'h2,8'hff-S5_num};
            else if(replenish_flag == 3'd2)//补货后货道1的信息
               dispay_num <= {8'h33,4'h1,S2_num,4'h2,S5_num};
            else if(replenish_flag == 3'd3)//货道1的信息
               dispay_num <= {8'h00,4'h3,8'hff-S7_num,4'h4,8'hff-S10_num};
            else if(replenish_flag == 3'd4)//补货后货道2的信息
               dispay_num <= {8'h55,4'h3,S7_num,4'h4,S10_num};
            else if(replenish_flag == 3'd5)//查看各商品的售出数量
               dispay_num <= {S2_sell,S5_sell,S7_sell,S10_sell};
            else if(replenish_flag == 3'd6)//查看总销售金额

            begin
               dispay_num[31:16] <= 16'h0000;
               dispay_num[15:12] <= price_sum/1000;
               dispay_num[11:8] <= price_sum/100%10;
               dispay_num[7:4] <= price_sum/10%10;
               dispay_num[3:0] <= price_sum%10;
            end
            else  dispay_num <= dispay_num;
        else dispay_num <= dispay_num;
    end
end
```

```
//1s计时模块
always@(posedge clk or negedge rst)
begin
    if(!rst)
    begin
        clk_1Hz <= 0;
        Hz_cnt <= 0;
    end
    else begin
        if(en_cnt)
           if(Hz_cnt == 30'd99_999_999)
          // if(Hz_cnt == 30'd9)//仅用于仿真用
            begin
                 clk_1Hz <= 1;
                 Hz_cnt <= 0;
            end
            else begin
                 clk_1Hz <= 0;
                 Hz_cnt <= Hz_cnt+20'd1;
            end
        else begin
          Hz_cnt <= 0;
```

```
            clk_1Hz <= 0;
        end
    end
end

//0.5s计时模块
always@(posedge clk or negedge rst)
begin
    if(!rst)
    begin
        clk_2Hz <= 0;
        Hz_cnt1 <= 0;
    end
    else begin
        //if(Hz_cnt1 == 30'd49_999_999)
        if(Hz_cnt1 == 30'd4)//仅用于仿真用
        begin
            clk_2Hz <= 1;
            Hz_cnt1 <= 0;
        end
        else begin

          else begin
              clk_2Hz <= 0;
              Hz_cnt1 <= Hz_cnt1+20'd1;
          end
      end
end
endmodule
```

HEX8_module:

```
module HEX8(
        Clk,
        Rst_n,
        disp_data,
        sel,
        seg
    );

    input Clk;  //50M
    input Rst_n;

    input [31:0]disp_data;

    output [7:0] sel;
    output [7:0] seg;

    reg [16:0]divider_cnt;//25000-1

    reg [2:0]clk_1K;
    reg [7:0]sel_r;
    reg [7:0]seg_r;
    reg [3:0]data_tmp;
```

```
//1Khz分频
    always@(posedge Clk or negedge Rst_n)
    if(!Rst_n)
        divider_cnt <= 17'd0;
    else if(divider_cnt == 17'd99999)//17'd99999
        divider_cnt <= 17'd0;
    else
        divider_cnt <= divider_cnt + 1'b1;


    always@(posedge Clk or negedge Rst_n)
    if(!Rst_n)
        clk_1K <= 3'd0;
    else if(divider_cnt == 17'd99999)//17'd99999
        if(clk_1K == 3'd7)
            clk_1K <= 3'd0;
        else clk_1K <= clk_1K+3'd1;
    else
        clk_1K <= clk_1K;
```

```
    always@(*)
    case(clk_1K)
        3'd0:sel_r = 8'b11111110;
        3'd1:sel_r = 8'b11111101;
        3'd2:sel_r = 8'b11111011;
        3'd3:sel_r = 8'b11110111;
        3'd4:sel_r = 8'b11101111;
        3'd5:sel_r = 8'b11011111;
        3'd6:sel_r = 8'b10111111;
        3'd7:sel_r = 8'b01111111;
        default:sel_r = 8'b11111111;
    endcase
```

```
    always@(*)
        case(sel_r)
            8'b11111110:data_tmp = disp_data[3:0];
            8'b11111101:data_tmp = disp_data[7:4];
            8'b11111011:data_tmp = disp_data[11:8];
            8'b11110111:data_tmp = disp_data[15:12];
            8'b11101111:data_tmp = disp_data[19:16];
            8'b11011111:data_tmp = disp_data[23:20];
            8'b10111111:data_tmp = disp_data[27:24];
            8'b01111111:data_tmp = disp_data[31:28];
            default:data_tmp = 4'b0000;
        endcase
```

```
    always@(*)
       case(data_tmp)
        4'h0:seg_r = 8'b11000000;
        4'h1:seg_r = 8'b11111001;
        4'h2:seg_r = 8'b10100100;
        4'h3:seg_r = 8'b10110000;
        4'h4:seg_r = 8'b10011001;
        4'h5:seg_r = 8'b10010010;
        4'h6:seg_r = 8'b10000010;
        4'h7:seg_r = 8'b11111000;
        4'h8:seg_r = 8'b10000000;
        4'h9:seg_r = 8'b10010000;
        4'ha:seg_r = 8'b10001000;
        4'hb:seg_r = 8'b10000011;
        4'hc:seg_r = 8'b11000110;
        4'hd:seg_r = 8'b10100001;
        4'he:seg_r = 8'b10000110;
        4'hf:seg_r = 8'b10001110;
       endcase

    assign sel = sel_r;
    assign seg = seg_r;
endmodule
```

key_filter_module

```
module key_filter_module(
    clk,                // 开发板上输入时钟：50Mhz
    rst_n,              // 开发板上输入复位按键
    key_in,             // 输入按键信号
    flag_key             // 输出一个时钟的高电平
);

localparam n = 5;//按键的个数

//============================================================
// PORT declarations
//============================================================
input         clk;
input         rst_n;
input  [n-1:0] key_in;
output [n-1:0] flag_key;

//寄存器定义
reg [21:0] count;
reg [n-1:0] key_scan; //按键扫描值KEY


//============================================================
// 采样按键值，20ms扫描一次,采样频率小于按键毛刺频率，相当于滤除掉了高频毛刺信号。
//============================================================
always @(posedge clk or negedge rst_n)     //检测时钟的上升沿和复位的下降沿
begin
   if(!rst_n)                //复位信号低有效
    begin
      count <= 22'd0;        //计数器清0
        key_scan <= 0;
    end
   else
      begin
         if(count ==22'd1999_999)    //20ms扫描一次按键,20ms计数(100M/50-1=1999_999)
         //if(count ==22'd2)//仅用于仿真用
            begin
               count <= 22'd0;      //计数器计到20ms，计数器清零
               key_scan <= key_in; //采样按键输入电平
            end
         else
            count <= count + 22'd1; //计数器加1
     end
end
```

```
//=============================================================
// 按键信号锁存一个时钟节拍
//=============================================================
reg [n-1:0] key_scan_r = 0;
always @(posedge clk)
    key_scan_r <= key_scan;

assign flag_key = key_scan_r[n-1:0] & (~key_scan[n-1:0]);  //当检测到按键有上降沿变化时，代表该按键被按下，按键有效

endmodule
```

e.约束文件：

```
#////////////////////////////系统时钟和复位//////////////////////////////////
set_property -dict {PACKAGE_PIN Y18 IOSTANDARD LVCMOS33} [get_ports clk ]
set_property -dict {PACKAGE_PIN Y9 IOSTANDARD LVCMOS33} [get_ports rst ]
#////////////////////////////拨码开关sw0~sw19//////////////////////////////////
set_property -dict {PACKAGE_PIN W4 IOSTANDARD LVCMOS33} [get_ports {sw[0]}]
set_property -dict {PACKAGE_PIN R4 IOSTANDARD LVCMOS33} [get_ports {sw[1]}]
set_property -dict {PACKAGE_PIN T4 IOSTANDARD LVCMOS33} [get_ports {sw[2]}]
set_property -dict {PACKAGE_PIN T5 IOSTANDARD LVCMOS33} [get_ports {sw[3]}]
set_property -dict {PACKAGE_PIN U5 IOSTANDARD LVCMOS33} [get_ports {sw[4]}]
set_property -dict {PACKAGE_PIN W6 IOSTANDARD LVCMOS33} [get_ports {sw[5]}]
set_property -dict {PACKAGE_PIN W5 IOSTANDARD LVCMOS33} [get_ports {sw[6]}]
set_property -dict {PACKAGE_PIN U6 IOSTANDARD LVCMOS33} [get_ports {sw[7]}]
set_property -dict {PACKAGE_PIN V5 IOSTANDARD LVCMOS33} [get_ports {sw[8]}]
set_property -dict {PACKAGE_PIN R6 IOSTANDARD LVCMOS33} [get_ports {sw[9]}]
set_property -dict {PACKAGE_PIN T6 IOSTANDARD LVCMOS33} [get_ports {sw[10]}]
set_property -dict {PACKAGE_PIN Y6 IOSTANDARD LVCMOS33} [get_ports {sw[11]}]
set_property -dict {PACKAGE_PIN AA6 IOSTANDARD LVCMOS33} [get_ports {sw[12]}]
set_property -dict {PACKAGE_PIN V7 IOSTANDARD LVCMOS33} [get_ports {sw[13]}]
set_property -dict {PACKAGE_PIN AB7 IOSTANDARD LVCMOS33} [get_ports {sw[14]}]
set_property -dict {PACKAGE_PIN AB6 IOSTANDARD LVCMOS33} [get_ports {sw[15]}]
set_property -dict {PACKAGE_PIN V9 IOSTANDARD LVCMOS33} [get_ports {sw[16]}]
set_property -dict {PACKAGE_PIN V8 IOSTANDARD LVCMOS33} [get_ports {sw[17]}]
set_property -dict {PACKAGE_PIN AA8 IOSTANDARD LVCMOS33} [get_ports {sw[18]}]

set_property -dict {PACKAGE_PIN AB8 IOSTANDARD LVCMOS33} [get_ports {sw[19]}]
#////////////////////////////////////5个按键//////////////////////////////////////
set_property -dict {PACKAGE_PIN R1 IOSTANDARD LVCMOS33} [get_ports s1]
set_property -dict {PACKAGE_PIN P1 IOSTANDARD LVCMOS33} [get_ports s2]
set_property -dict {PACKAGE_PIN P5 IOSTANDARD LVCMOS33} [get_ports s3]
set_property -dict {PACKAGE_PIN P4  IOSTANDARD LVCMOS33} [get_ports s4]
set_property -dict {PACKAGE_PIN P2  IOSTANDARD LVCMOS33} [get_ports s5]

set_property -dict {PACKAGE_PIN A19 IOSTANDARD LVCMOS33} [get_ports beep ]

set_property -dict {PACKAGE_PIN C19 IOSTANDARD LVCMOS33} [get_ports {sel[0]}]
set_property -dict {PACKAGE_PIN E19 IOSTANDARD LVCMOS33} [get_ports {sel[1]}]
set_property -dict {PACKAGE_PIN D19 IOSTANDARD LVCMOS33} [get_ports {sel[2]}]
set_property -dict {PACKAGE_PIN F18 IOSTANDARD LVCMOS33} [get_ports {sel[3]}]
set_property -dict {PACKAGE_PIN E18 IOSTANDARD LVCMOS33} [get_ports {sel[4]}]
set_property -dict {PACKAGE_PIN B20 IOSTANDARD LVCMOS33} [get_ports {sel[5]}]
set_property -dict {PACKAGE_PIN A20 IOSTANDARD LVCMOS33} [get_ports {sel[6]}]
set_property -dict {PACKAGE_PIN A18 IOSTANDARD LVCMOS33} [get_ports {sel[7]}]
```

```
set_property -dict {PACKAGE_PIN F15 IOSTANDARD LVCMOS33} [get_ports {seg[0]}]
set_property -dict {PACKAGE_PIN F13 IOSTANDARD LVCMOS33} [get_ports {seg[1]}]
set_property -dict {PACKAGE_PIN F14 IOSTANDARD LVCMOS33} [get_ports {seg[2]}]
set_property -dict {PACKAGE_PIN F16 IOSTANDARD LVCMOS33} [get_ports {seg[3]}]
set_property -dict {PACKAGE_PIN E17 IOSTANDARD LVCMOS33} [get_ports {seg[4]}]
set_property -dict {PACKAGE_PIN C14 IOSTANDARD LVCMOS33} [get_ports {seg[5]}]
set_property -dict {PACKAGE_PIN C15 IOSTANDARD LVCMOS33} [get_ports {seg[6]}]
set_property -dict {PACKAGE_PIN E13 IOSTANDARD LVCMOS33} [get_ports {seg[7]}]
```

3.测试：

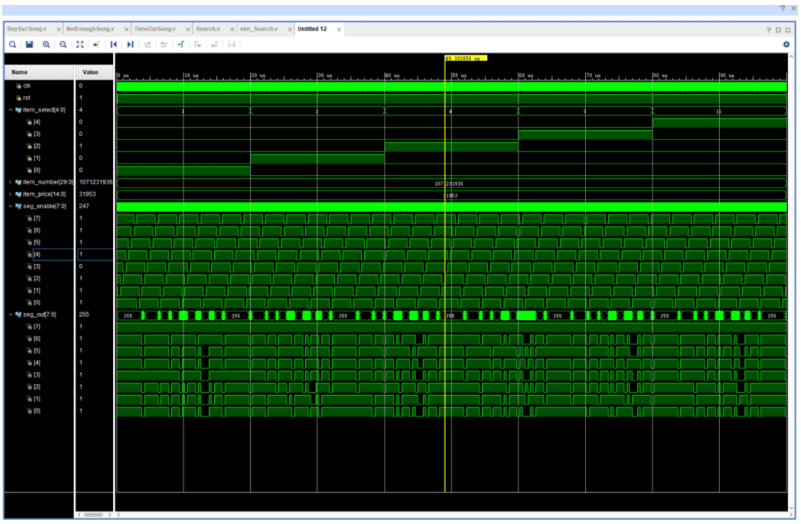
testbench 文件代码：

```
module auto_machine_tb();
reg clk,rst,s1,s2,s3,s4,s5;
reg[19:0]sw;
wire beep;
wire[7:0] sel,seg;

auto_machine uut
(
  .clk(clk),     //100Mhz
  .rst(rst),     //系统复位—s6
  .s1(s1),//确认购买按键
  .s2(s2),//回到查询阶段按键
  .s3(s3),//确认补货按键
  .s4(s4),//投币确认按键
  .s5(s5),//补货确认按键
  .sw(sw),//sw[0]货道1，sw[1]:0代表货道1中的第一个商品;1代表货道1中的第二个商品
          //sw[2]货道2，sw[3]:0代表货道2中的第一个商品;1代表货道2中的第二个商品
          //sw[4]sw[5],代表选择商品的价格；sw[6]~sw[9],代表投币的金额
          //sw[10]~sw[17],代表补充商品的数量;sw[18]查看各商品的售出数量;sw[19]查看总销售金额
  .sel(sel),
  .seg(seg),
  .beep(beep)//蜂鸣器
);

initial clk = 0;
always #5 clk = ~clk;
```

```
initial begin
    rst = 1;
    s1 = 1;
    s2 = 1;
    s3 = 1;
    s4 = 1;
    s5 = 1;
    sw = 20'd0;
     #40;
    rst = 0;
    #40;
    rst = 1;
    #3000;
    //补货
    s3 = 0;
    #200;
   s3 = 1;
   sw[0] =1;
   sw[1] = 0;
   sw[17:10] = 8'd6;
    s5 = 0;

     #200;
     s5 = 1;
     #200;
     sw[1] = 1;
    sw[17:10] = 8'd3;
     s5 = 0;
     #200;
     s5 = 1;
     #200;
     sw[0] =0;
      sw[2] =1;
       sw[3] = 0;
       sw[17:10] = 8'd9;
        s5 = 0;
        #200;
        s5 = 1;
         #200;
         s5 = 1;
         #200;
         sw[3] = 1;
        sw[17:10] = 8'd4;
         s5 = 0;
         #200;
         s5 = 1;
         #200;
         //回到查询阶段
          s2 = 0;
           #200;
           s2 = 1;
           #200;
    //付款
    sw[14] = 1;
    s1 = 0;
    #200;
    s1 = 1;
    #10000;
    $stop;
```

```
end
endmodule
```

仿真波形图:



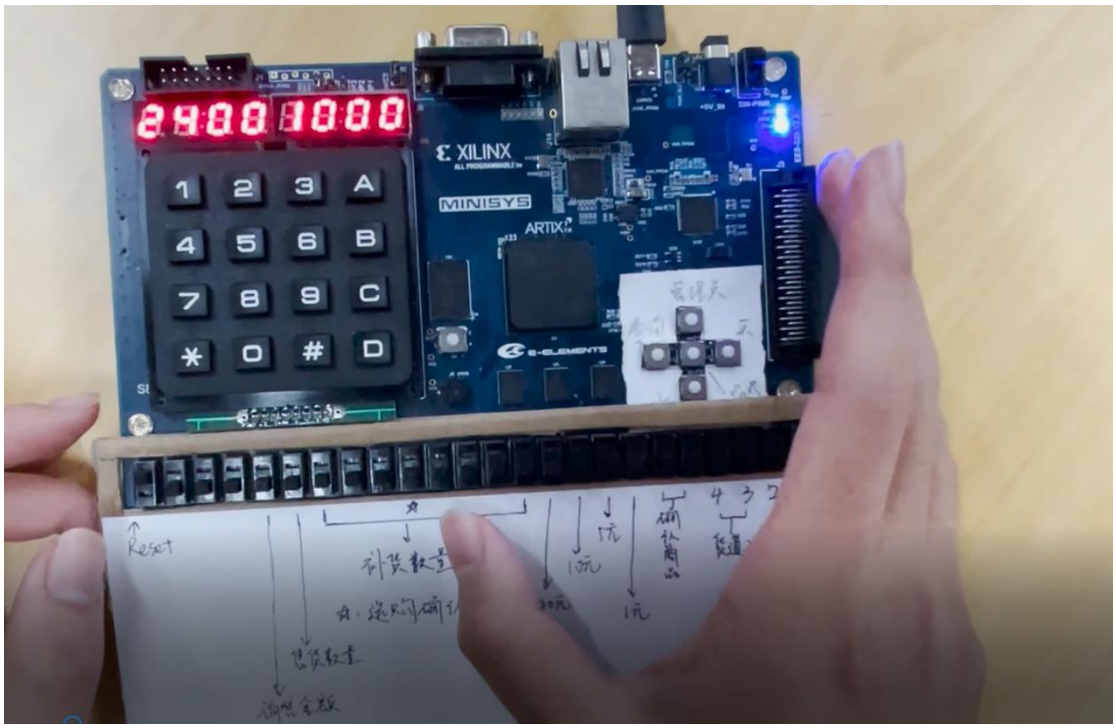
仿真波形图出现滚动显示，以及数字变化，仿真成功

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Design Name:
// Module Name: auto_machine_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module auto_machine_tb();

reg clk,rst,s1,s2,s3,s4,s5;
reg[19:0]sw;
```

```
wire beep;
wire[7:0] sel,seg;

auto_machine uut
(
   .clk(clk),    //100Mhz
   .rst(rst),    //系统复位--s6
   .s1(s1),//确认购买按键
   .s2(s2),//回到查询阶段按键
   .s3(s3),//确认补货按键
   .s4(s4),//投币确认按键
   .s5(s5),//补货确认按键
   .sw(sw),//sw[0]货道1，sw[1]:0代表货道1中的第一个商品;1代表货道1中的第二个商品
                  //sw[2]货道2，sw[3]:0代表货道2中的第一个商品;1代表货道2中的第二个商品
                  //sw[4]sw[5],代表选择商品的价格；sw[6]~sw[9],代表投币的金额
                  //sw[10]~sw[17],代表补充商品的数量;sw[18]查看各商品的售出数量;sw[19]查看总销售
金额
  .sel(sel),
  .seg(seg),
  .beep(beep)//蜂鸣器
);

initial clk = 0;
always #5 clk = ~clk;

initial begin
    rst = 1;
    s1 = 1;
    s2 = 1;
    s3 = 1;
    s4 = 1;
    s5 = 1;
    sw = 20'd0;
     #40;
    rst = 0;
    #40;
    rst = 1;
    #3000;
    //补货
    s3 = 0;
    #200;
   s3 = 1;
   sw[0] =1;
```

```
   sw[1] = 0;
   sw[17:10] = 8'd6;
    s5 = 0;
    #200;
    s5 = 1;
    #200;
    sw[1] = 1;
   sw[17:10] = 8'd3;
    s5 = 0;
    #200;
    s5 = 1;
    #200;
    sw[0] =0;
     sw[2] =1;
      sw[3] = 0;
      sw[17:10] = 8'd9;
       s5 = 0;
       #200;
       s5 = 1;
       #200;
       sw[3] = 1;
      sw[17:10] = 8'd4;
       s5 = 0;
       #200;
       s5 = 1;
       #200;
       //回到查询阶段
        s2 = 0;
         #200;
         s2 = 1;
         #200;
    //付款
    sw[14] = 1;
    s1 = 0;
    #200;
    s1 = 1;
    #10000;
    $stop;
end


endmodule
```

b.上板测试与解释：



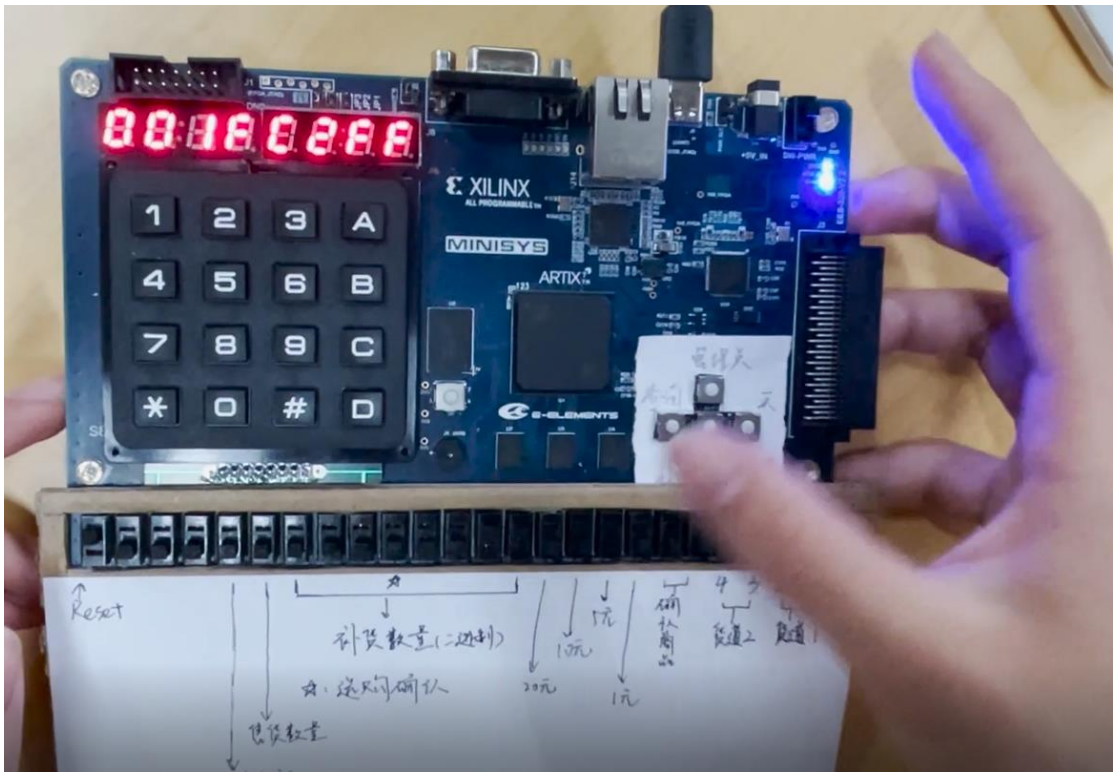
七段数码显示管此时显示 24001000；这里代表的含义分别是：

24：此时为第二的货道，编号为 4 的商品

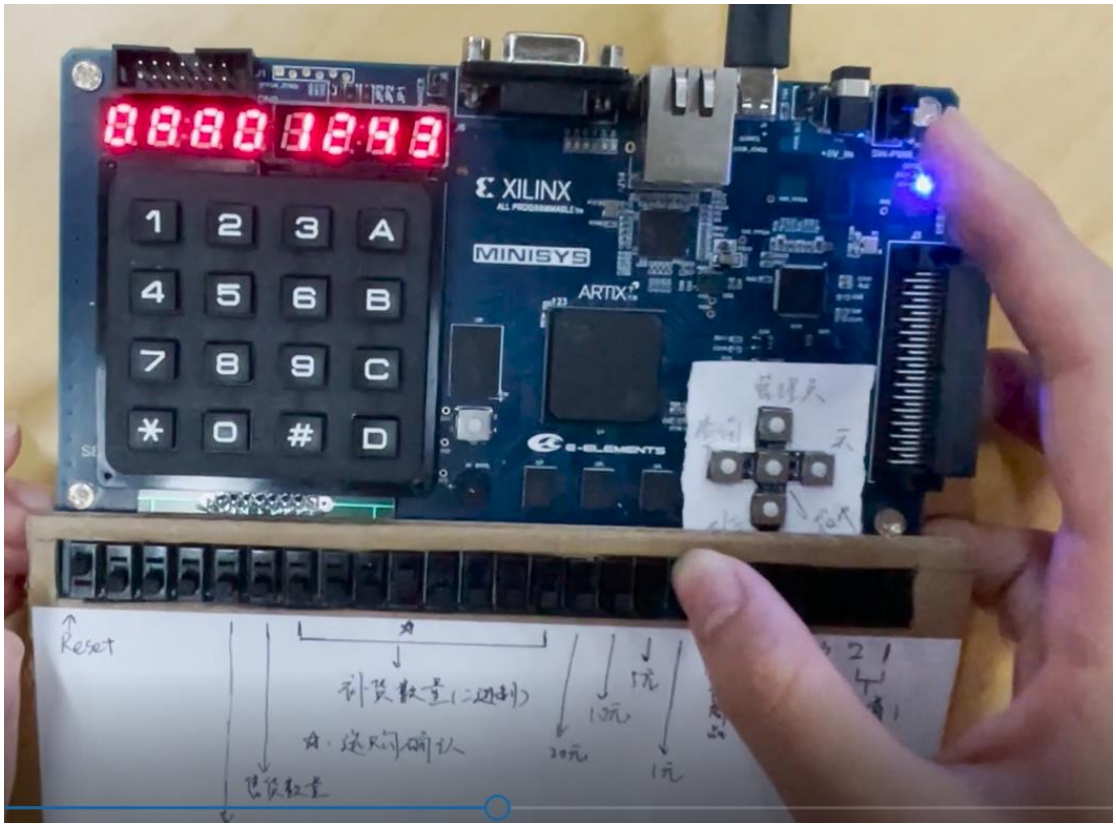
00：此时存货余量为 0；

10：商品售价为 10 元；

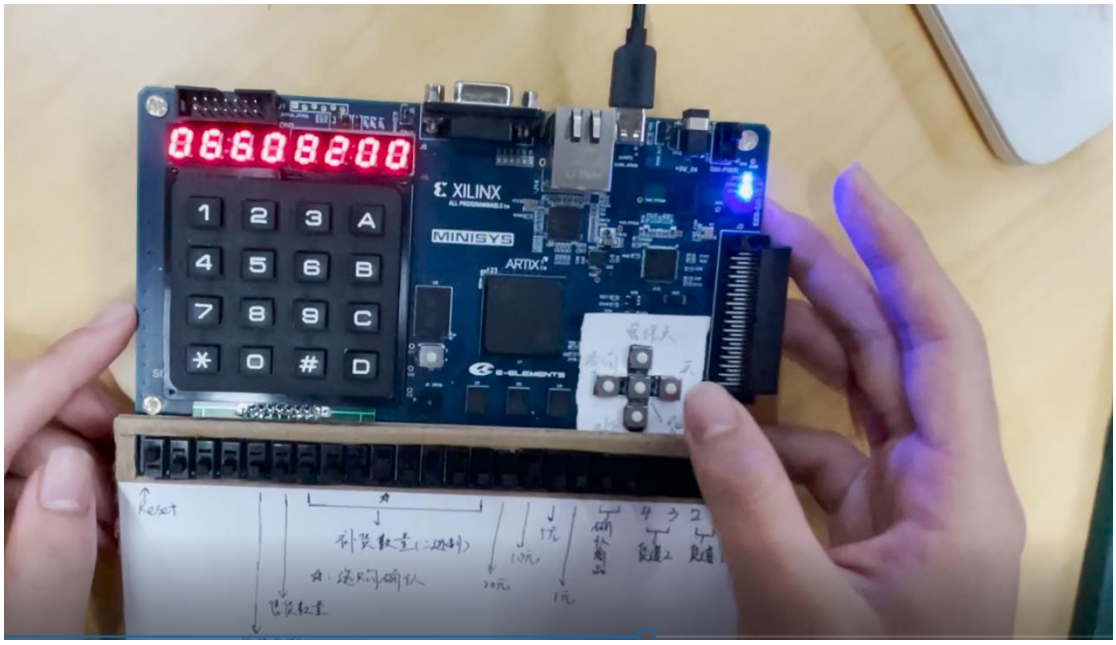
00：最后两位是付款时的倒计时阶段



此时属于管理员模式下的捕获阶段，我们可以看到，七段数码显示管上出现了了：
001FC2FF 的字样
1FC:表示 1 号商品剩余可补数量为 FC(16 进制)个
2FF:表示 2 号商品剩余可补数量为 FF(16 进制)个



此时为购买界面，七段数码显示管显示 0AA01243：
0AA0:表示正在购买
1：表示已经投币数为 1 元
2：表示商品单价为 2 元
43：付款倒计时还剩余 43 秒



此时为购买完成后界面，七段数码显示管显示 06608200：
0660：表示购买成功
8：表示退款为 8 元（因为之前投了 10 元的币）
2：表示商品价格为 2 元

4.总结：

a) 开发过程中遇到的问题及解决方案的总结
（1）滚动显示出现异常：对滚动模块的代码进行检查，发现是阻塞与非阻塞混用的原因导致异常
（2）查询金额显示异常：当我们按下查询金额的拨码开关时，无变化，发现是因为我们在代码中将其与查询数量的按键绑定在了一起，导致其行为与查询数量相同
（3）流程图设计与规范化位宽：讨论了许久，决定采用 16 进制，2 进制，及一些固定的位宽来规范。

b) 对当前系统的特色以及优化方向进行描述：
特色：符合现实中自动售货机的操作，
对于不同的购买情况有不同的蜂鸣器声响，
规范化位宽，便于代码共享

优化方向:
可以将蜂鸣器继续优化，比如说，用蜂鸣器制作 BGM,为每一个步骤都加上蜂鸣器
可以后续加上小键盘的使用，以方便用户直观的投币，
可以加入 LED 灯，付款时从左到右依次亮起，这样可以更为美观。