数字逻辑电路

课程设计

**2023. 9.4~2023.9.8（第1周）**

**课题： 模拟自动售货机**

**学号： 19210231 19210230**

**姓名： 徐洋 袁睿**

**指导老师： 黄为民老师**

# 目录

[一.实验目的——————————————————3](#_一、实验目的)

[二.实验原理——————————————————4](#_二、实验原理)

[三.功能实现——————————————————6](#_三、功能实现)

[四.分析与设计思路———————————————7](#_四、分析与设计思路)

[五.设计代码——————————————————8](#_五、设计代码)

[五（修改）.代码修改增添————————————18](#_五(修)、代码修改)

[六.管教分配——————————————————25](#_六、管脚分配)

[七.仿真结果——————————————————26](#_七、仿真结果)

[八.实验结果展示————————————————27](#_八、实验结果展示)

[九.总结与感悟—————————————————32](#_九、总结与感悟)

[十.参考资料——————————————————33](#_十、参考资料)

## 一、实验目的

1. 设计一个模拟自动售货机，此机能出售1元、2元、5元、10元的四种商品。出售哪种商品可有顾客按动相应的一个按键即可，并同时用数码管显示出此商品的价格。

2. 顾客投入硬币的钱数也是有1元、2元、5元、10元四种，但每次只能投入其中的一种硬币，此操作通过按动相应的一个按键来模拟，并同时用数码管将投币额显示出来。

3．顾客投币后，按一次确认键，如果投币额不足时则报警，报警时间3秒（可用LED灯模拟报警）。如果投币额足够时自动送出货物（送出的货物用相应不同的LED灯显示来模拟），同时多余的钱应找回，找回的钱数用数码管显示出来。

4．顾客一旦按动确认键3秒后，自动售货机即可自动恢复到初始状态，此时才允许顾客进行下一次购货操作。

5．售货机还应具有供商家使用的累加卖货额的功能，累加的钱数要用数码管显示，显示2位即可。此累加器只有商家可以控制清零。

6. 此售货机要设有一个由商家控制的整体复位控制。

## 二、实验原理

整个数字系统的核心是控制模块（Control Module），其思想基于有限状态机，通过在不同状态的转换中完成选择、投币、找零计算、投币不足时报警、交易成功状态的转换等操作。在控制模块输出的需要显示在七段数码管上的数据需要有一个将二进制转换为BCD码的识别转换模块（BCDconvertModule），最后这些处理好的数据通过显示模块（DisplayModule）进行在七段数码管上显示。对于时钟信号clk，其频率是50MHZ，需要一个分频模块（clk\_1hz）来将频率降为1hz。整个系统需要一个计时模块（集成在控制模块，作为一个等待状态实现）来实现按下确认键后三秒复位/警报三秒的功能。

**我完成的是BCDconvertModule和DisplayModule，以及后续对于该程序增加商品，增加货币面额的修改。**

**BCDconvertModule原理：**

因为我只需要考虑转换两位数的码，所以7位二进制编码长度即可，BCD码长度8位**，低四位存储个位**，**高四位存储十位**。

对于提取二进制的**十**位，只需将其**/**10，得到的商存入高四位。

对于提取二进制的**个**位，则将其**%10**,得到的余数存入低四位。

图片包含 文本

描述已自动生成

**DisplayModule原理：**

如图所示七段数码管包含七根数码管和一个小点h。

每组数码管由a~g的7个独立阴极组成，7个独立阴极的亮熄状态决定了数码管显示的状态。每个BCD码对应数码管的一种显示，故考虑用case语句完成对一组数码管a~g的赋值。a到h从右往左由低到高

我设计低电平有效，不需要点h输出，所以设置“seg[7] = 1'b1;”

对于考虑到高低电平有效的改变，如果直接赋值，下次更改高低电平有效，就要重新计算真值表，所以我在赋值前取反，若更改则只需把取反删去，方便修改

“4'h0: seg[6:0] = ~7'h3F;”。

## 三、功能实现

1. 功能按键

图片包含 游戏机, 电路

描述已自动生成

1. 功能描述

将复位rst键（或者称为使能端）先调至1（低电平使能，调至低电平时会清空当前选择商品的价格，投入硬币的总量，但是不会清空总销售额，此时整个数字系统暂停运作），此时选择一个价格的商品（对应开关上拨后下拨记为一次选择，后续开关同理），该商品的价格会在对应的数码管上显示，此时系统等待我们投币，我们选择需要投币的数额，上拨开关再下拨记为一次投币，每次只能投入一种硬币，数码管会实时显示投币总额。按下确认按钮后，会出现两种情况：1）当投币数额大于等于商品价格时，对应商品的LED灯点亮表示出货，数码管显示需要找客户的钱，同时更新累加总销售额。三秒后自动售货机恢复初始状态，可以进行下一次选择商品并购买的操作。2）当投币额小于商品价格时，警报LED灯亮起，找钱的数额即为投入硬币的数额，代表售货机将钱退给客户，本次购买失败。三秒后售货机恢复初始状态。

对于总销售额，在自动售货机正常运行时会不断累加，只有当商家拨动整体复位按钮时才会清零，代表商家将赚到的钱从售货机中拿出。

## 四、分析与设计思路

1. 电路图

图形用户界面

描述已自动生成

1. 状态转换图

图示

描述已自动生成

## 五、设计代码

（1）BCD码转换：

module BCDconvertModule(input[6:0] Num, output reg[7:0] BCD);

/\*

reg[3:0] tens;

reg[3:0] ones;

integer i;

always@(hexNum)

begin

for( i = 6; i >= 0; i = i - 1)

begin

if(ones >= 4'd5) ones = ones + 4'd3;

if(tens >= 4'd5) tens = tens + 4'd3;

tens = {tens[2:0], ones[3]};

ones = {ones[2:0], hexNum[i]};

end

hexBCD = {tens, ones};

end

\*/

always@(Num)

begin

BCD[7:4] = Num / 10;

BCD[3:0] = Num % 10;

end

endmodule

（2）显示模块：

module DisplayModule(input [3:0]displayNum, output reg[7:0]seg);

// hex digit to 7 seg

always @(displayNum)

begin

seg[7] = 1'b1;

case(displayNum)

4'h0: seg[6:0] = ~7'h3F;

4'h1: seg[6:0] = ~7'h06;

4'h2: seg[6:0] = ~7'h5B;

4'h3: seg[6:0] = ~7'h4F;

4'h4: seg[6:0] = ~7'h66;

4'h5: seg[6:0] = ~7'h6D;

4'h6: seg[6:0] = ~7'h7D;

4'h7: seg[6:0] = ~7'h07;

4'h8: seg[6:0] = ~7'h7F;

4'h9: seg[6:0] = ~7'h6F;

4'ha: seg[6:0] = ~7'h77;

4'hb: seg[6:0] = ~7'h7C;

4'hc: seg[6:0] = ~7'h39;

4'hd: seg[6:0] = ~7'h5E;

4'he: seg[6:0] = ~7'h79;

4'hf: seg[6:0] = ~7'h71;

endcase

end

endmodule

（3）分频模块：

module clk\_1hz(CLK, RST, clk\_1Hz);

input CLK, RST;

output clk\_1Hz;

wire clk\_1MHz, clk\_100KHz, clk\_10KHz, clk\_1KHz, clk\_100Hz, clk\_10Hz;

//assign clock = {clk\_1MHz, clk\_100KHz, clk\_10KHz, clk\_1KHz, clk\_100Hz, clk\_10Hz, clk\_1Hz};

divide\_by\_50 d6 (clk\_1MHz, CLK, RST);

divide\_by\_10 d5 (clk\_100KHz, clk\_1MHz, RST);

divide\_by\_10 d4 (clk\_10KHz, clk\_100KHz, RST);

divide\_by\_10 d3 (clk\_1KHz, clk\_10KHz, RST);

divide\_by\_10 d2 (clk\_100Hz, clk\_1KHz, RST);

divide\_by\_10 d1 (clk\_10Hz, clk\_100Hz, RST);

divide\_by\_10 d0 (clk\_1Hz, clk\_10Hz, RST);

endmodule

module divide\_by\_10 (Q, CLK, RST,);

input CLK, RST;

output Q;

reg Q;

reg [2:0] count;

always @(posedge CLK or negedge RST)

begin

if (~RST)

begin

Q <= 1'b0;

count <= 3'b000;

end

else if (count < 4)

begin

count <= count + 1'b1;

end

else

begin

count <= 3'b000;

Q <= ~Q;

end

end

endmodule

module divide\_by\_50 (Q, CLK, RST);

input CLK, RST;

output Q;

reg Q;

reg [4:0] count;

always @(posedge CLK or negedge RST)

begin

if (~RST)

begin

Q <= 1'b0;

count <= 5'b00000;

end

else if (count < 24)

begin

count <= count + 1'b1;

end

else

begin

count <= 5'b00000;

Q <= ~Q;

end

end

endmodule

（4）控制模块：

module ControlModule(

input wire clk,

input wire rst,

//input wire [1:0] state,

//input wire [3:0] price,

//input wire [3:0] coinBalance,

// input wire [1:0] selectedProduct,

input wire coin1,

input wire coin2,

input wire coin5,

input wire coin10,

input wire select1,

input wire select2,

input wire select5,

input wire select10,

input wire confirm,

input wire resetTotal,

output reg[6:0] price,

output reg[6:0] coinBalance,

output reg[3:0] selectItem,

output reg alarm,

// output reg vendSuccess,

output reg[6:0] change,

output reg[6:0] total

);

parameter UNSELECTED = 2'b00, SELECTED= 2'b01, WAIT\_3SEC = 2'b10;

// temp var

reg[1:0] ITEM; // 0 -> 1 1 -> 2 2 -> 5 3 -> 10

//reg [6:0] coinBalance\_ ;

//reg[6:0] change\_;

//reg[6:0] total\_;

reg[1:0] state = UNSELECTED;

reg[1:0] counter = 0;

always @(posedge clk or negedge rst)

begin

if(~rst)

begin

price <= 7'h0;

//coinBalance\_ <= 7'd0;

alarm <= 1'b0;

state <= UNSELECTED;

coinBalance <= 7'b0;

selectItem <= 4'b0;

change <= 7'b0;

//total <= 0;

end

else if(resetTotal)

begin

//total\_ <= 7'b0;

total <= 7'b0;

end

else

begin

case(state)

UNSELECTED:

begin

// init

price <= 7'b0;

//coinBalance\_ <= 7'b0;

alarm <= 1'b0;

state <= UNSELECTED;

coinBalance <= 7'b0;

selectItem <= 4'b0;

change <= 7'b0;

counter <= 0;

// select

if(select1)

begin

price <= 7'b0000001;

state <= SELECTED;

ITEM <= 2'd0;

end

else if(select2)

begin

price <= 7'b0000010;

state <= SELECTED;

ITEM <= 2'd1;

end

else if(select5)

begin

price <= 7'b0000101;

state <= SELECTED;

ITEM <= 2'd2;

end

else if(select10)

begin

price <= 7'b0001010;

state <= SELECTED;

ITEM <= 2'd3;

end

end

SELECTED:

begin

if(coin1)

begin

//coinBalance\_ <= coinBalance\_ + 1;

coinBalance <= coinBalance + 1;

//hexBCDconvertModule converter1(.hexNum(coinBalance), .hexBCD(coinBalance\_BCD));

end

else if(coin2)

begin

//coinBalance\_ <= coinBalance\_ + 2;

coinBalance <= coinBalance + 2;

//hexBCDconvertModule converter2(.hexNum(coinBalance), .hexBCD(coinBalance\_BCD));

end

else if(coin5)

begin

//coinBalance\_ <= coinBalance\_ + 5;

coinBalance <= coinBalance + 5;

//hexBCDconvertModule converter3(.hexNum(coinBalance), .hexBCD(coinBalance\_BCD));

end

else if(coin10)

begin

//coinBalance\_ <= coinBalance\_ + 10;

coinBalance <= coinBalance + 10;

//hexBCDconvertModule converter4(.hexNum(coinBalance), .hexBCD(coinBalance\_BCD));

end

if(confirm)

begin

if(price <= coinBalance)

begin

//change\_ <= coinBalance - price;

//hexBCDconvertModule converter5(.hexNum(change), .hexBCD(change\_BCD));

total <= total + price;

change <= coinBalance - price;

//hexBCDconvertModule converter6(.hexNum(total), .hexBCD(total\_BCD));

//total <= total\_;

//change <= change\_;

case(ITEM)

0: selectItem <= 4'b0001;

1: selectItem <= 4'b0010;

2: selectItem <= 4'b0100;

3: selectItem <= 4'b1000;

endcase

//total <= total\_;

state <= WAIT\_3SEC;

end

else

begin

change <= coinBalance;

state <= WAIT\_3SEC;

alarm <= 1'b1;

//selectItem <= selectItem & 4'b0000;

end

end

end

// time counter 3 sec

WAIT\_3SEC:

begin

if(~rst) counter <= 0;

if(counter == 3)

begin

counter <= 0;

if(~alarm)

state <= UNSELECTED;

else

begin

state <= UNSELECTED;

end

end

else

counter <= counter + 1;

end

default: state <= UNSELECTED;

endcase

end

end

endmodule

（5）顶层模块

module VendingMachine(

input wire clk,  **// 时钟信号**

input wire rst, **// 复位信号**

input wire coin1,  **// 1元硬币输入信号**

input wire coin2, **// 2元硬币输入信号**

input wire coin5,  **// 5元硬币输入信号**

input wire coin10,  **// 10元硬币输入信号**

input wire select1, **// 选择商品1信号**

input wire select2,  **// 选择商品2信号**

input wire select5,  **// 选择商品5信号**

input wire select10, **// 选择商品10信号**

input wire confirm,  **// 确认购买信号**

input wire resetTotal,  **// 清零销售总额信号**

output[15:0] price\_7seg, **// the price of item, printed in 7seg**

output[15:0] change\_7seg,

output[15:0] coinBalance\_7seg,

output[15:0] total\_7seg,

output[3:0] selectItem, // LED reveal the successfully bought item

output alarm **// 报警信号**

**// output vendSuccess, // 购买成功信号**

**// output [3:0] change, // 找零信号**

**// output [3:0] total // 销售总额**

);

wire clk\_1hz;

wire[7:0] price\_BCD;

wire[7:0] coinBalance\_BCD;

wire[7:0] change\_BCD;

wire[7:0] total\_BCD;

wire[6:0] price;

wire[6:0] coinBalance;

wire[6:0] change;

wire[6:0] total;

clk\_1hz clk\_1hz\_generator(.CLK(clk), .RST(rst), .clk\_1Hz(clk\_1hz));

ControlModule controlModule(

.clk(clk\_1hz),

**//.clk(clk),**

.rst(rst),

**//.state(state),**

**//.price(price),**

**//.coinBalance(coinBalance),**

.coin1(coin1),

.coin2(coin2),

.coin5(coin5),

.coin10(coin10),

.select1(select1),

.select2(select2),

.select5(select5),

.select10(select10),

.confirm(confirm),

.resetTotal(resetTotal),

.price(price),

.coinBalance(coinBalance),

.selectItem(selectItem),

.alarm(alarm),

**//.vendSuccess(vendSuccess),**

.change(change),

.total(total));

BCDconvertModule convert1(.Num(price),

.BCD(price\_BCD));

BCDconvertModule convert2(.Num(coinBalance),

.BCD(coinBalance\_BCD));

BCDconvertModule convert3(.Num(change),

.BCD(change\_BCD));

BCDconvertModule convert4(.Num(total),

.BCD(total\_BCD));

**// 1 -> shi wei 2 -> ge wei**

DisplayModule displayPrice1(.displayNum(price\_BCD[7:4]),

.seg(price\_7seg[15:8]));

DisplayModule displayPrice2(.displayNum(price\_BCD[3:0]),

.seg(price\_7seg[7:0]));

DisplayModule displayBalance1(.displayNum(coinBalance\_BCD[7:4]),

.seg(coinBalance\_7seg[15:8]));

DisplayModule displayBalance2(.displayNum(coinBalance\_BCD[3:0]),

.seg(coinBalance\_7seg[7:0]));

DisplayModule displayChange1(.displayNum(change\_BCD[7:4]),

.seg(change\_7seg[15:8]));

DisplayModule displayChange2(.displayNum(change\_BCD[3:0]),

.seg(change\_7seg[7:0]));

DisplayModule displayTotal1(.displayNum(total\_BCD[7:4]),

.seg(total\_7seg[15:8]));

DisplayModule displayTotal2(.displayNum(total\_BCD[3:0]),

.seg(total\_7seg[7:0]));

endmodule

## 五(修)、代码修改

对于商品或者货币的增添修改，在此给出修改增加20元价值商品的修改代码。

只需要在顶层模块和控制模块增加20元商品。

包括（20元商品选择，20元商品信号灯显示，售卖成功信号，选择后的指令增添）

顶层模块：

module VendingMachine(

input wire clk, // 时钟信号

input wire rst, // 复位信号

input wire coin1, // 1元硬币输入信号

input wire coin2, // 2元硬币输入信号

input wire coin5, // 5元硬币输入信号

input wire coin10, // 10元硬币输入信号

input wire select1, // 选择商品1信号

input wire select2, // 选择商品2信号

input wire select5, // 选择商品5信号

input wire select10, // 选择商品10信号

**input wire select20, // 选择商品20信号**

input wire confirm, // 确认购买信号

input wire resetTotal, // 清零销售总额信号

output[15:0] price\_7seg, // the price of item, printed in 7seg

output[15:0] change\_7seg,

output[15:0] coinBalance\_7seg,

output[15:0] total\_7seg,

output[**4**:0] select Item, **//五个商品，所以增加一位**

output alarm // 报警信号

// output vendSuccess, // 购买成功信号

// output [3:0] change, // 找零信号

// output [3:0] total // 销售总额

);

wire clk\_1hz;

wire[7:0] price\_BCD;

wire[7:0] coinBalance\_BCD;

wire[7:0] change\_BCD;

wire[7:0] total\_BCD;

wire[6:0] price;

wire[6:0] coinBalance;

wire[6:0] change;

wire[6:0] total;

clk\_1hz clk\_1hz\_generator(.CLK(clk), .RST(rst), .clk\_1Hz(clk\_1hz));

ControlModule controlModule(

.clk(clk\_1hz),

//.clk(clk),

.rst(rst),

//.state(state),

//.price(price),

//.coinBalance(coinBalance),

.coin1(coin1),

.coin2(coin2),

.coin5(coin5),

.coin10(coin10),

.select1(select1),

.select2(select2),

.select5(select5),

.select10(select10),

**.select20(select20), //增加20元商品选择**

.confirm(confirm),

.resetTotal(resetTotal),

.price(price),

.coinBalance(coinBalance),

.selectItem(selectItem),

.alarm(alarm),

//.vendSuccess(vendSuccess),

.change(change),

.total(total));

BCDconvertModule convert1(.Num(price),

.BCD(price\_BCD));

BCDconvertModule convert2(.Num(coinBalance),

.BCD(coinBalance\_BCD));

BCDconvertModule convert3(.Num(change),

.BCD(change\_BCD));

BCDconvertModule convert4(.Num(total),

.BCD(total\_BCD));

// 1 -> shi wei 2 -> ge wei

DisplayModule displayPrice1(.displayNum(price\_BCD[7:4]),

.seg(price\_7seg[15:8]));

DisplayModule displayPrice2(.displayNum(price\_BCD[3:0]),

.seg(price\_7seg[7:0]));

DisplayModule displayBalance1(.displayNum(coinBalance\_BCD[7:4]),

.seg(coinBalance\_7seg[15:8]));

DisplayModule displayBalance2(.displayNum(coinBalance\_BCD[3:0]),

.seg(coinBalance\_7seg[7:0]));

DisplayModule displayChange1(.displayNum(change\_BCD[7:4]),

.seg(change\_7seg[15:8]));

DisplayModule displayChange2(.displayNum(change\_BCD[3:0]),

.seg(change\_7seg[7:0]));

DisplayModule displayTotal1(.displayNum(total\_BCD[7:4]),

.seg(total\_7seg[15:8]));

DisplayModule displayTotal2(.displayNum(total\_BCD[3:0]),

.seg(total\_7seg[7:0]));

endmodule

**控制模块：**

module ControlModule(

input wire clk,

input wire rst,

//input wire [1:0] state,

//input wire [3:0] price,

//input wire [3:0] coinBalance,

// input wire [1:0] selectedProduct,

input wire coin1,

input wire coin2,

input wire coin5,

input wire coin10,

input wire select1,

input wire select2,

input wire select5,

input wire select10,

input wire select20,

input wire confirm,

input wire resetTotal,

output reg[6:0] price,

output reg[6:0] coinBalance,

output reg[**4**:0] selectItem, **//多一位**

output reg alarm,

// output reg vendSuccess,

output reg[6:0] change,

output reg[6:0] total

);

parameter UNSELECTED = 2'b00, SELECTED= 2'b01, WAIT\_3SEC = 2'b10;

// temp var

reg[**2**:0] ITEM; **// 000 -> 1 001 -> 2 010 -> 5 011 -> 10 100 - 20**

//reg [6:0] coinBalance\_ ;

//reg[6:0] change\_;

//reg[6:0] total\_;

reg[1:0] state = UNSELECTED;

reg[1:0] counter = 0;

always @(posedge clk or negedge rst)

begin

if(~rst)

begin

price <= 7'h0;

//coinBalance\_ <= 7'd0;

alarm <= 1'b0;

state <= UNSELECTED;

coinBalance <= 7'b0;

selectItem <= 5'b0;

change <= 7'b0;

//total <= 0;

end

else if(resetTotal)

begin

//total\_ <= 7'b0;

total <= 7'b0;

end

else

begin

case(state)

UNSELECTED:

begin

// init

price <= 7'b0;

//coinBalance\_ <= 7'b0;

alarm <= 1'b0;

state <= UNSELECTED;

coinBalance <= 7'b0;

selectItem <= 5'b0;

change <= 7'b0;

counter <= 0;

// select

if(select1)

begin

price <= 7'b0000001;

state <= SELECTED;

ITEM <= **3**'d0;

end

else if(select2)

begin

price <= 7'b0000010;

state <= SELECTED;

ITEM <= **3**'d1;

end

else if(select5)

begin

price <= 7'b0000101;

state <= SELECTED;

ITEM <= **3**'d2;

end

else if(select10)

begin

price <= 7'b0001010;

state <= SELECTED;

ITEM <= **3**'d3;

end

**else if(select20) //增加20元商品选择**

**begin**

**price <= 7'b0010100;**

**state <= SELECTED;**

**ITEM <= 3'd4;**

**end**

**end**

SELECTED:

begin

if(coin1)

begin

//coinBalance\_ <= coinBalance\_ + 1;

coinBalance <= coinBalance + 1;

//hexBCDconvertModule converter1(.hexNum(coinBalance), .hexBCD(coinBalance\_BCD));

end

else if(coin2)

begin

//coinBalance\_ <= coinBalance\_ + 2;

coinBalance <= coinBalance + 2;

//hexBCDconvertModule converter2(.hexNum(coinBalance), .hexBCD(coinBalance\_BCD));

end

else if(coin5)

begin

//coinBalance\_ <= coinBalance\_ + 5;

coinBalance <= coinBalance + 5;

//hexBCDconvertModule converter3(.hexNum(coinBalance), .hexBCD(coinBalance\_BCD));

end

else if(coin10)

begin

//coinBalance\_ <= coinBalance\_ + 10;

coinBalance <= coinBalance + 10;

//hexBCDconvertModule converter4(.hexNum(coinBalance), .hexBCD(coinBalance\_BCD));

end

if(confirm)

begin

if(price <= coinBalance)

begin

//change\_ <= coinBalance - price;

//hexBCDconvertModule converter5(.hexNum(change), .hexBCD(change\_BCD));

total <= total + price;

change <= coinBalance - price;

//hexBCDconvertModule converter6(.hexNum(total), .hexBCD(total\_BCD));

//total <= total\_;

//change <= change\_;

case(ITEM)

0: selectItem <= **5**'b00001;

1: selectItem <= **5**'b00010;

2: selectItem <= **5**'b00100;

3: selectItem <= **5**'b01000;

4: selectItem <= **5**'b10000;

endcase

//total <= total\_;

state <= WAIT\_3SEC;

end

else

begin

change <= coinBalance;

state <= WAIT\_3SEC;

alarm <= 1'b1;

//selectItem <= selectItem & 4'b0000;

end

end

end

// time counter 3 sec

WAIT\_3SEC:

begin

if(~rst) counter <= 0;

if(counter == 3)

begin

counter <= 0;

if(~alarm)

state <= UNSELECTED;

else

begin

state <= UNSELECTED;

end

end

else

counter <= counter + 1;

end

default: state <= UNSELECTED;

endcase

end

end

endmodule

## 六、管脚分配

表格

描述已自动生成

表格

描述已自动生成

## 七、仿真结果

图形用户界面, 应用程序, 表格, Excel

描述已自动生成

## 八、实验结果展示

1）选择10元商品后投币7元（尚未按下确认键）

图片包含 游戏机, 电子, 电路

描述已自动生成

2）选择10元商品投币7元后按下确认，发出警报，此时找钱为7元，意为投币不足，把钱退给客户

图片包含 电子, 电路, 游戏机

描述已自动生成

3）报警三秒后恢复初始状态

电子设备的屏幕

中度可信度描述已自动生成

4）选择五元商品投币25元，尚未按下确认按键

电子设备的屏幕

描述已自动生成

5）选择五元商品投币25元，按下确认按键，此时对应五元商品的LED灯亮起，找钱显示20元，总销售额从零元变为5元

图片包含 电子, 游戏机, 电路

描述已自动生成

6）三秒归零后，选择10元商品投币27元，尚未按下确认按键，此时销售额还是五元

图片包含 电子, 游戏机, 电路

描述已自动生成

7）按下确认键，此时找钱为17元，总销售额从5元变为15元，10元商品对应的LED灯亮起

电子设备的屏幕

中度可信度描述已自动生成

8）三秒后归零，此时总销售额为15元

图片包含 游戏机, 电子, 电路

描述已自动生成

9）此时，按下商家的总体复位键清空总销售额，意为商家从售货机中取出赚到的钱，此时售货机回到最初始状态。

电子机器

低可信度描述已自动生成

## 九、总结与感悟

这是我第一次和同学合作完成一门课程设计。在数字逻辑电路课设中，我们小组设计并实现了一个模拟自动售货机。我在小组中担任**组员**的角色，另一名成员是组长，他在整个过程中起到了指导和主导的作用，特别是在代码书写方面。以下是我对整个课设过程的总结与感悟。

首先，通过这个课设项目，我对数字逻辑电路的原理和应用有了更深入的理解。我们需要设计一个能够处理输入信号并产生特定输出的逻辑电路，以实现自动售货机的各项功能。在这个过程中，我学习了寄存器、数码管以及其它数字逻辑电路的概念和工作原理。

其次，团队合作在项目中起到了至关重要的作用。组长通过指导和引导，帮助我理解与代码编写有关的概念和技术，使我能够更好地参与到项目中。我们共同克服了遇到的问题，最终完成了一个功能完善的模拟自动售货机。

同时，这个课设项目也锻炼了我的问题解决能力和编程技巧。在编写代码的过程中，我需要根据给定的要求，使用适当的逻辑门和寄存器来处理输入信号，并生成正确的输出。通过不断调试和测试，我学会了如何排查和修复代码中可能存在的错误，并最终使自动售货机的各项功能正常运行。

最后，通过这个课设项目，我对团队合作和项目管理的重要性有了更深刻的认识。良好的沟通和协作是成功的关键，而且及时的项目管理和任务分配有助于确保项目的进展和质量。

总的来说，参与数字逻辑电路课设的过程让我受益匪浅。我不仅学到了有关数字逻辑电路的理论知识和实际应用，还培养了解决问题和团队合作的能力。我相信这些经验和技能将对我未来的学习和职业发展产生积极的影响。

## 十、参考资料

[1] 康华光. 电子技术基础(数字部分) [M ]. 第五版

[2]Verilog HDL程序设计教程 （第一版） 王金明主编

[3] 夏宇闻. 复杂数字电路与系统的Verilog HDL设计技术 [M ].北京: 北京航空航天大学出版社, 1998

[4] 张明. V erilog HDL 实用教程[M ]. 成都: 电子科技大学出版社, 1999

[5] FPGA设计及应用（第二版）西安电子科技大学出版社