数字电路与逻辑设计
课程设计报告

报　告　人：吕鹏泽
实验指导教师：赵贻竹
报告批阅教师：赵贻竹

计算机科学与技术学院

2018 年　**9** 月　**10** 日

评　分　表

评 分 项		分 数	备　注	合 计
实验	设计过程（结构、状态机/流程图）			
	仿真			
	下板验证结果（Bug）			
	考勤			
	附加（亮点）功能			
	名次			
设计报告	设计思路（含系统结构框图）			
	实现代码			
	仿真过程			
	主要问题及解决方法			
	功能测试			
	总结与心得			
	格式			
	参考文献			
	工作表			
合　　计				
教　师　签　名				

华 中 科 技 大 学 课 程 设 计 报 告

数字逻辑课程设计学生工作表

班 级	姓 名	学 号	验收时间（教师填写）
CS1601	吕鹏泽	U201614532	

（学生填写）

（1）主要工作的描述

（2）难点、亮点

课设进度记录（学生填写）

日期	进度
9 月 3 日	确定选题，拟定设计思路和状态图
9 月 4 日	确定设计思路，作出状态图和模块图
9 月 5 日	重新设计状态机，完成设计思路
9 月 6 日	完成设计思路，显示模块、状态机模块代码
9 月 7 日	完成自动销售机的大体框架
9 月 8 日	完善自动销售机的细节处理
9 月 9 日	撰写自动销售机报告

实验平台故障记录（学生填写，请注明实验平台的编号

实验编号 022，未遇到实验故障

重要说明

1、 **时间安排：课内** 2 周。

2、 验收准备：

1） 完成本表学生应该填写部分；

2） 每位学生必须都能**以独自完成的方式**应对任何形式的验收；

3） 完成课程设计报告书（**格式参见模板**）；

4） 将源程序和报告的电子文档交班长。

3、 检查过程：

1） 提交验收准备材料，请求老师验收，之后按验收老师的要求做；

2） 在开发平台上根据验收老师的要求进行演示；

3） 检查过程中独立回答老师提出的相关问题；

4） 验收老师有权根据具体情况调整验收的内容与方式；

5） 验收完成后关闭电源，整理好设备。

4、 评分标准：

1) 在完成控制器基本要求外，有**亮点**为加分项；

2) 在规定时间内完成控制器基本要求；

3) 在规定时间内完成控制器**部分**基本要求；

4) 检查时间。

5、 课程设计判定为不合格的一些情形： （本人已阅读此条款 1-5 项：签名 吕鹏泽 ）

1） 请人代做或冒名顶替者；

2） 替人做且不听劝告者；

3） 课程设计报告内容抄袭或雷同者；

4） 课程设计报告内容与实际实验内容不一致者；

5） 课程设计代码抄袭者。

目 录

1 综合实验设计概述

1.1 实验目的

（1）掌握 Vivado 软件的使用方法；

（2）熟悉 FPGA 器件的使用方法；

（3）用 Verilog HDL 进行较复杂逻辑电路的设计和调试；

（4）学习数字系统的设计方法；

（5）通过规范化的实验报告，培养学生良好的文档习惯以及撰写规范文档的能力。

1.2 实验要求

（1）能够全面地应用课程中所学的基本理论和基本方法，完成从设计逻辑电路到设计简单数字系统的过渡；

（2）能力独立思考、独立查阅资料，独立设计规定的系统；

（3）能够独立地完成实施过程，包括电路设计、调试、排除故障、仿真和下载验证。

1.3 实验任务

本次课程设计每人要完成一个设计任务，具体参见数字逻辑课程综合实验设计题目。

（1）制定出详细设计方案，认真记载毕业设计工作日记；

（2）通过 Verilog HDL 完成规定的设计任务，采取模块化、层次化的设计方法设计电路，然后进行编译和仿真，认真记录实施过程中遇到的各自故障以及解决方法，保证设计的正确性；

（3）生成 bit 文件，下载到开发板上，通过实际线路进行验证设计的正确性；

（4）撰写设计报告，并对存在的问题进行分析、提出改进意见。

1.4 实验环境

开发环境为 Vivado 2015.2 软件和开发板 NEXYS 4（芯片为 XC7A100TCSG324-1，封装为 CSG3242）。 Vivado 2015.2 是使用 Xilinx FPGA 必备的设计工具。它可以完成 FPGA 开发的全部流程，包括设计输入、仿真、 综合、布局布线、生成 bit 文件、配置以及在线调试等功能。

Nexys4 开发板简介：参见图 1-1 所示，它是一款简单易用的数字电路开发平台，可以支持在课堂环境中来设计一些行业应用。大规模、高容量的 FPGA，海量的外部存储，各种 USB、以太网、以及其它接口、这些让 Nexys4-DDR 能够满足从入门级组合逻辑电路到强大的嵌入式系统的设计。同时，板上集成的加速度、温度传感器，MEMs 数字麦克风，扬声器放大器以及人量的 I/O 设备，让 Vexys4-DDR 不需要增添额外组件而用于各种各样的设计。

注意：开发板提供的时钟信号频率为 100Mhz，对应的引脚封装编号为“E3”。

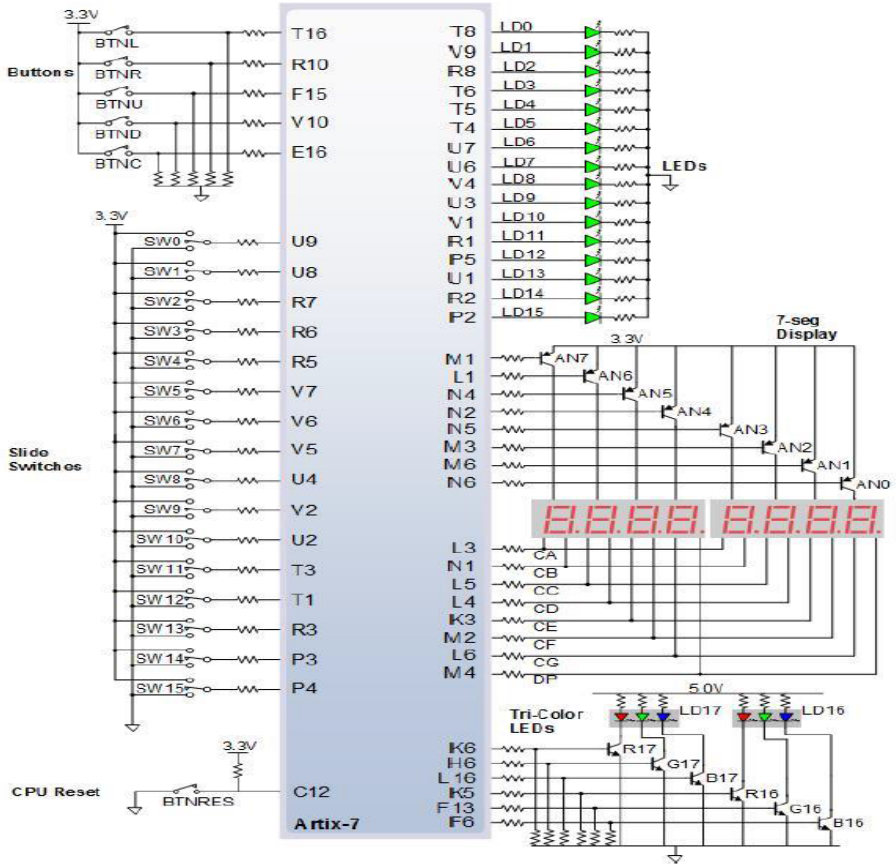

图 1-1 Nexys4 通用 I / O 设备

2 自动销售机控制器设计方案

2.1 内容

（1）基本内容

（A）设计并利用 Nexys4 开发板实现自动销售机控制器，它具有下述功能：

a.电源开关同时作为电路总清零信号（Reset），当 Reset = Off 不工作、为 On 时电路进入初始状态，且电源指示灯亮（又称售卖机运行指示灯），销售机处于售卖状态同时显示“Hello”；

b.自动销售机销售两类商品，一类售价 2.5 元，另一类售价 5 元，该贩卖机只能辨识 1 元，10 元两种人民币；

c.当投入 1 元时，机器会进入余额不足的状态，直到投入的金额大于 2.5 元为止。如果一次投入 10 元，则可以选择所有的产品，否则就只能选择 2.5 元的产品

d.“选择”完成后，售卖机卖出商品并且找回剩余的零钱，随后，机器又将返回售卖状态同时显示“Hello”。

（B）根据功能描述画出系统结构框图

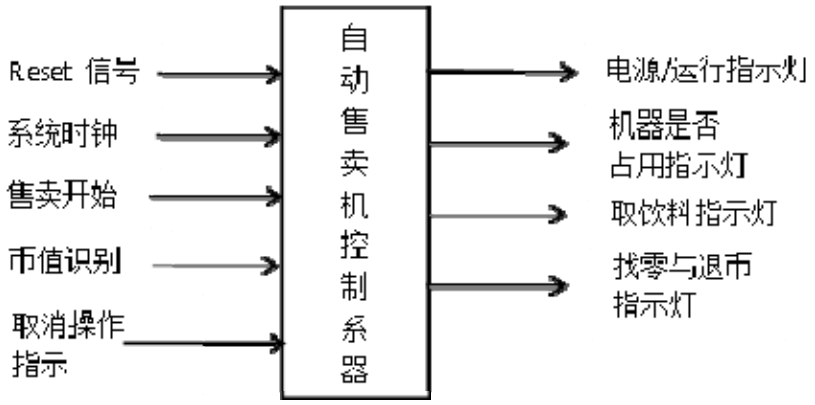

图 2-1 自动投币售卖机控制器框图

（2）扩展功能

a.购买完商品后如果余额不为 0，用户可继续投币并继续购买。如果余额为 0，售货机则回到 hello 状态。

b.当余额小于购买商品价格时，售货机余额不足指示灯会亮 2s，数码管会显示 nob（no_buy）

c.当余额大于 63 元时，此时售货机不再接受投币，若用户继续投币，禁止投币指示灯会亮 2s，数码管会显示 nop(no_pay)，余额不再增加。

2.2 设计思路

（1）自动销售机模块图

自动销售机的模块图如图 2-2 所示，各个模块的作用如下：

a.自动销售机要有一个 top 模块，负责总体信号的输入与输入以及子模块之间的信号传递

b.系统时钟为 100MHz，divider 模块负责分频提供给子模块

c. FSM 为负责状态转移的状态机模块

d. cur_money 模块负责当前余额，当有有效投币信号时，余额增加，当有有效支付信号时，余额减少。

e. led_inc 模块负责输出指定时间的电平信号，当输入信号出现上升沿时，输出信号就会输出指定时间的电平，用于控制取货和找零指示灯的亮。

f. fangdou 模块消除按键的抖动现象

g. display 模块负责数码管的显示内容

h. pattern 模块将输入的 8421 码转换为对应数码管数字的编码

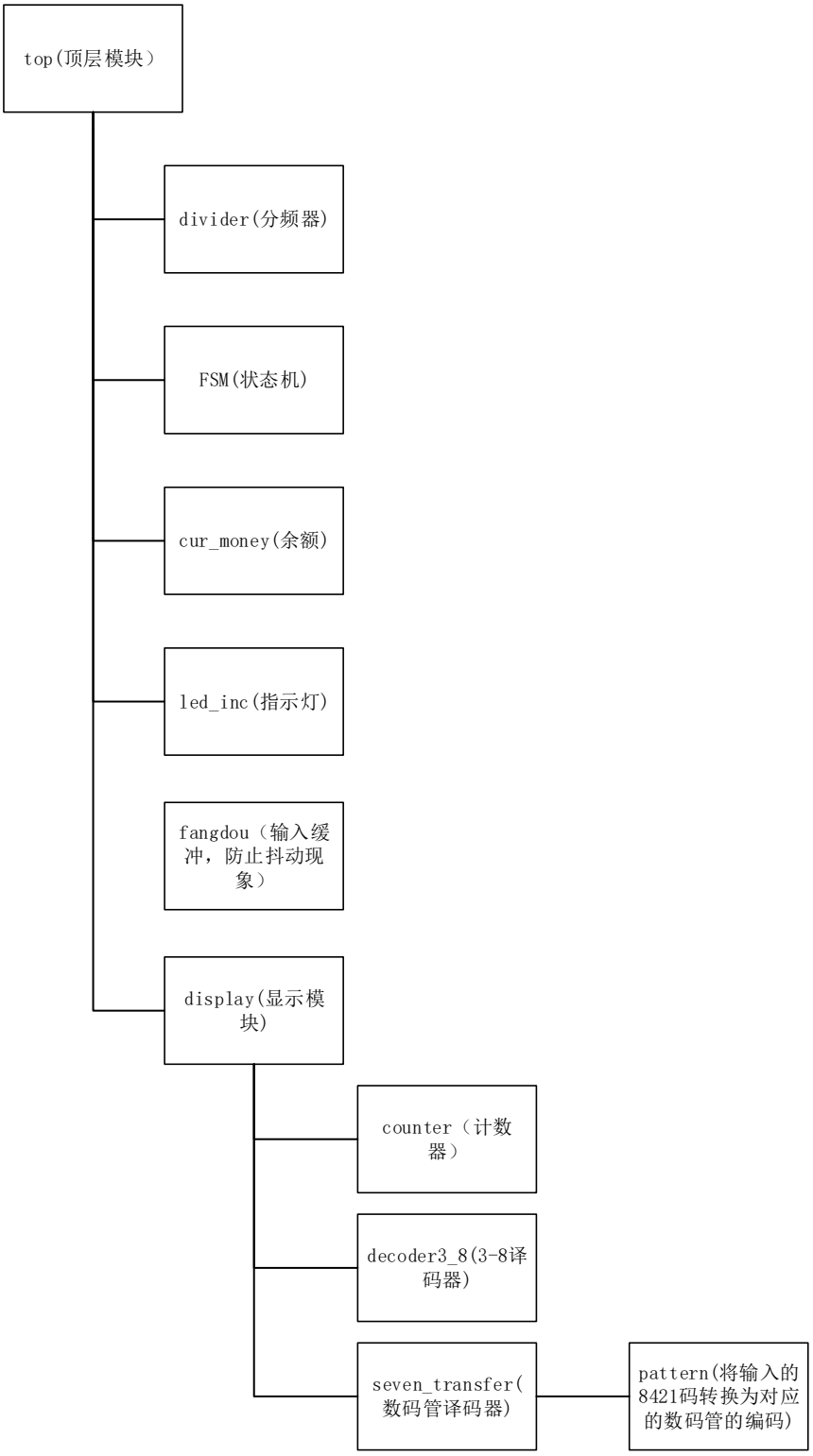

图 2-2　自动销售机控制器的模块图

（2）自动销售机流程图

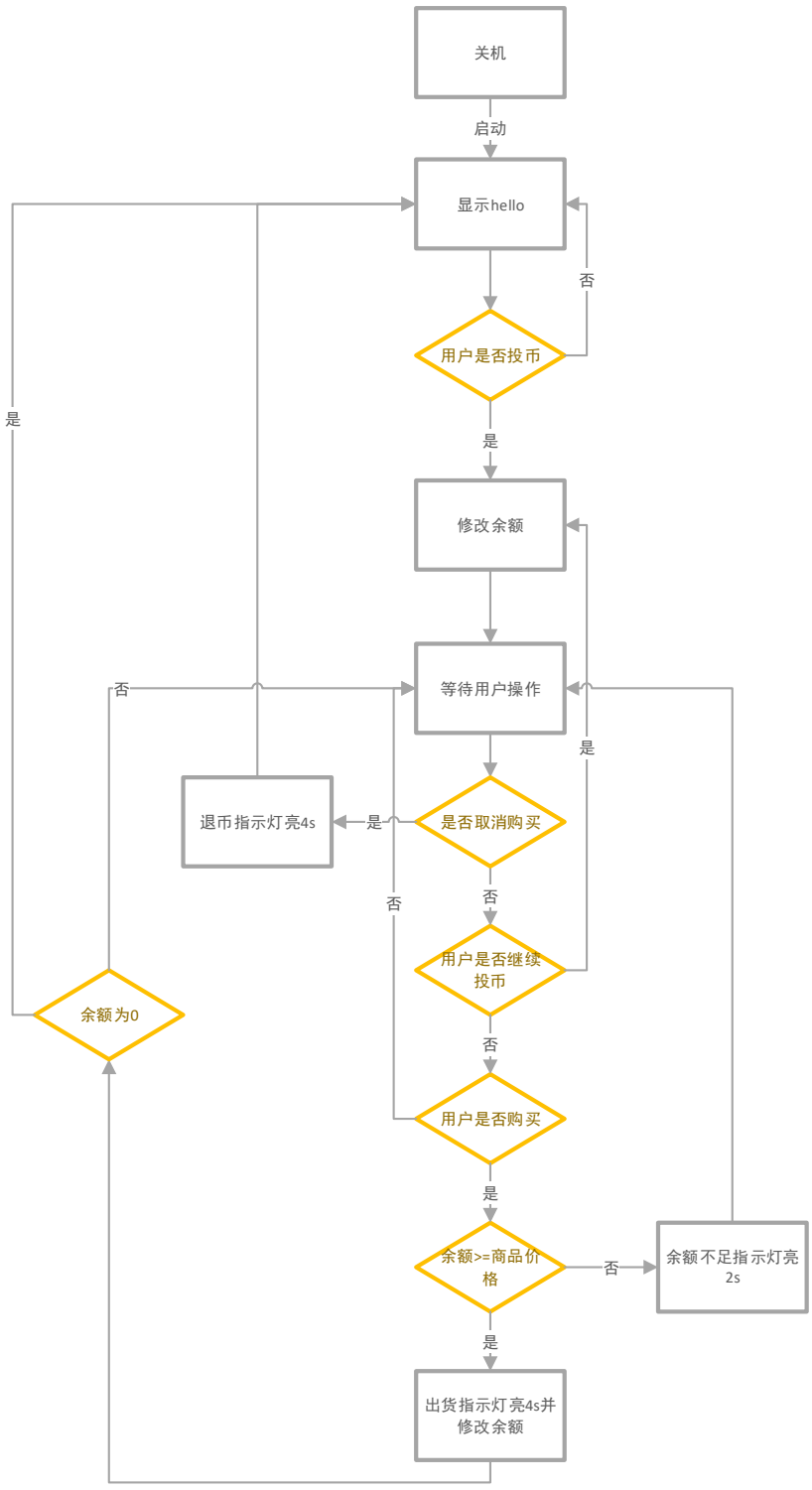

图 2-3 自动销售机销售流程图

a.检测到用户投币后销售机即被占用，此时用户可选择继续投币、购买商品、退币取消购买。

b.若用户选择继续投币，销售机中的余额会增加。

c.若用户选择购买商品，若余额大于商品价格，取货指示灯亮 4s，等待用户下一步操作。若余额小于商品价格，余额不足指示灯亮 2s，提示用户余额不足，等待用户下一步操作。

d.若用户选择退币取消购买，退币指示灯亮 4s，状态机回到 hello 状态，数码管显示 hello，等待下一位用户的操作。

2.2.1 top 模块设计

1)实现输入与输出信号的交互

2)实现子模块之间的信号传递

3)将币值转为二进制数传递给子模块

2.2.2 FSM 模块设计

（1）状态机信号说明：

输入信号：

a. 系统时钟（clk）作为系统的同步信号；

b. rst 系统重置信号

c. cancel_sig:用户取消购买信号

d. pay_sig:用户投币信号

e. coin_type:用户投币类型

f. cur_val:当前余额

输出信号：

g. hold_led：占用指示灯

h. charge_sig:找零信号

i. hello_sig：状态机位于 hello 状态时 hello_sig=1，否则为 0，控制数码管的显示内容

j. can_buy:标识用户当前可购买的商品

（2）自动投币售卖机的状态确定:

共 5 种状态：

HELLO 状态 余额不足状态 可购买 2.5 元商品状态 可购买 5 元商品状态 找零状态

（3）自动投币售卖机的状态转移图，参见图 2-4 所示；

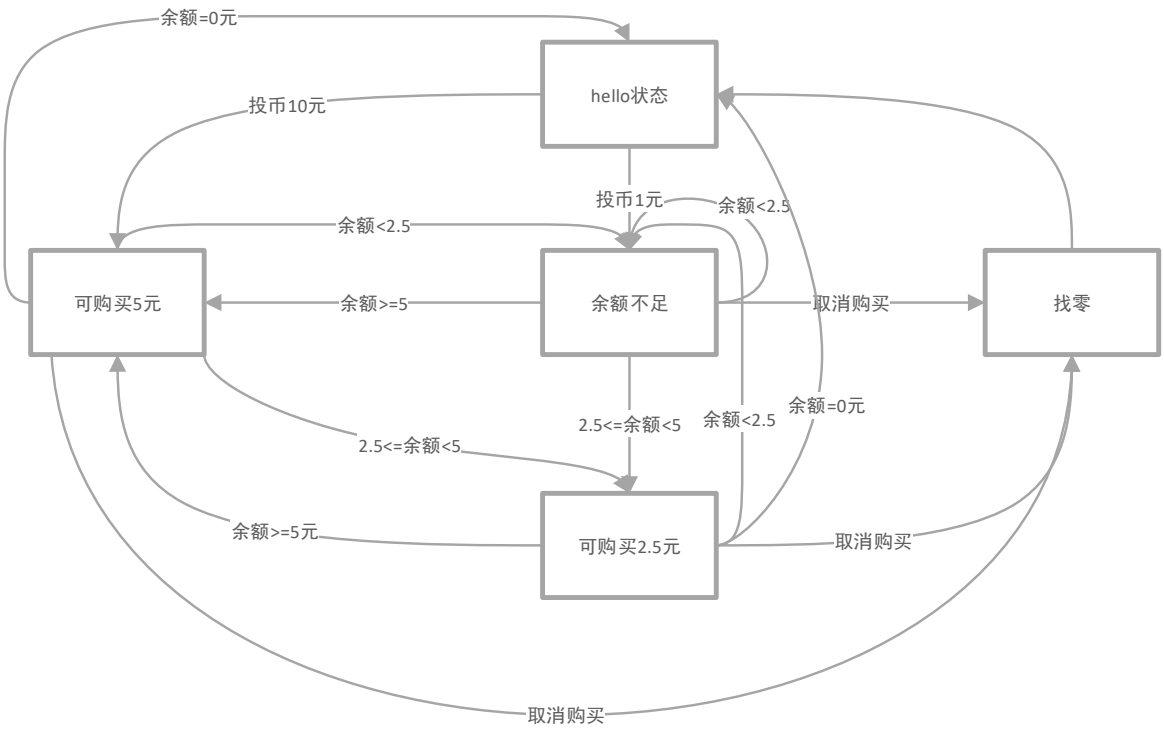

图 2-4 自动售卖机状态转移图

（4）自动售卖机状态说明：

a. hello 状态，当收到投币信号时进行状态转移

hold_led=0;

charge_sig=0;

hello_sig=1;

can_buy=2'b00;

b. 余额不足状态，当余额发生变化时进行状态转移

hold_led=1;

can_buy=2'b00;

hello_sig=0;

c. 可购买 2.5 元商品状态，当余额发生变化时进行状态转移

hold_led=1;

can_buy=2'b01

hello_sig=0;

d. 可购买 5 元商品状态，当余额发生变化时进行状态转移

hold_led=1;

can_buy=2'b10;

hello_sig=0;

e. 找零状态，当进入找零状态后状态转移到 hello 状态

charge_sig=1;

hello_sig=0;

2.2.3 display 模块设计

（1）使用计数器控制数码管的刷新(数码管一次亮一个并显示内容，然后切换到下一个数码管并显示下一个内容，当频率足够高时轮流点亮数码管人眼会认为它们是一起亮的)

（2）使用 7 段译码器根据 hello_sig，计数器的值选择数码管亮的内容

（3）使用 3-8 译码器将计数器的值对应点亮的数码管

2.2.4 cur_money 模块设计

cur_money 包含两个 reg 类型的寄存器，分别记录投币总值和购买总值，当收到 clear 信号（即

找零信号）和 rst 信号（系统重启信号）时清空为 0，一个线网类型 cur_val，其值等于投币总值减去购买总值。当收到投币信号时，投币总值加上投币额。当收到购买信号时，如果余额大于饮料价格，购买总值加上饮料价格，否则保持不变。

2.2.5 led_inc 模块设计

led_inc 输入为 sig 信号和频率为 1Hz 的时钟，当 sig 信号出现上升沿时，计时器重置为 0，在 clk 信号的作用下进行累加，然后计时器数值大于 N 后停止累加，将计时器数值<N 的真值作为结果输出即可实现指定时间的电平信号。

2.3 代码实现

顶层模块首先要设计系统的原理图，然后用 Verilog HDL 实现它，底层各模块均采用 Verilog HDL 语言设计。

自动销售机顶层原理图，参见图 2-5、图 2-6 和图 2-7 所示。

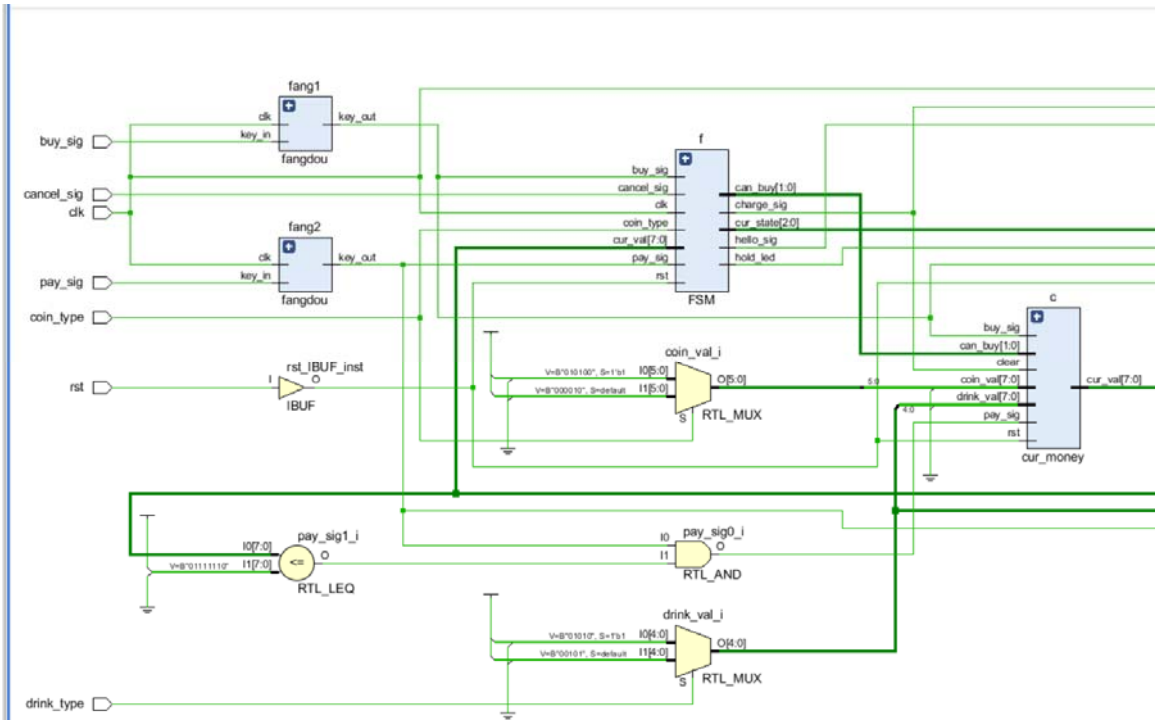

图 2-5 自动售卖机局部放大图 a

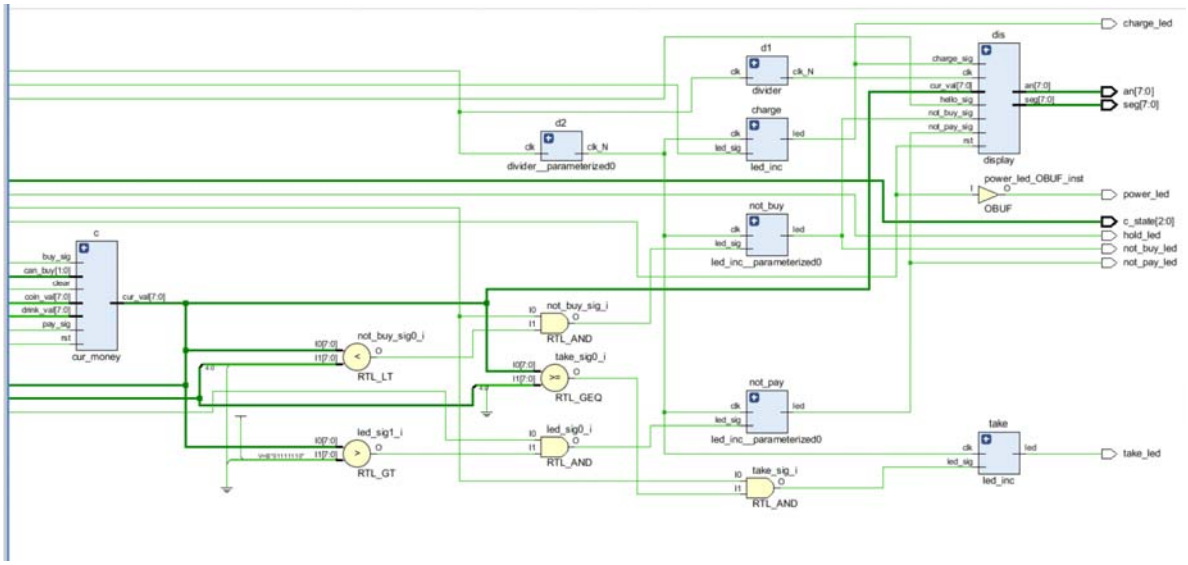

图 2-6 自动售卖机局部放大图 b

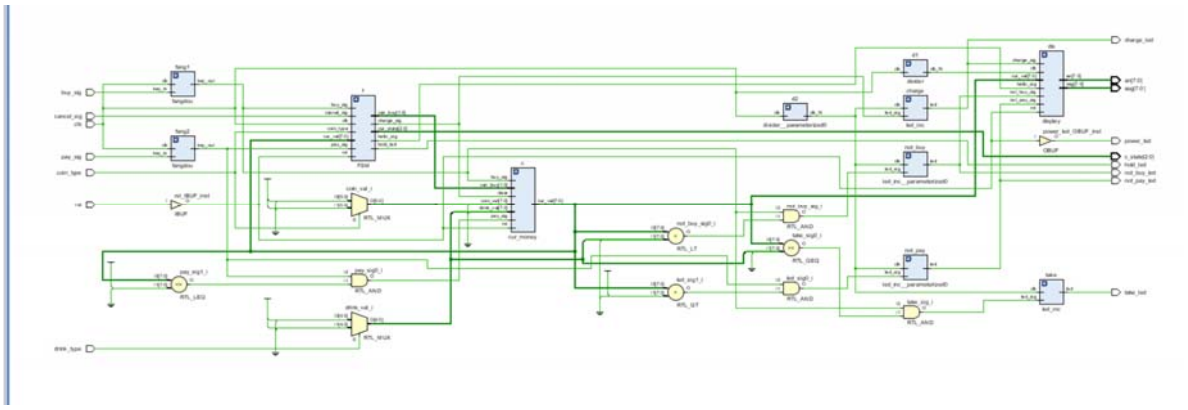

图 2-7 自动售卖机顶层原理图

（1）top 顶层模块

程序 2-1：top 顶层模块 Verilog 代码

```
`timescale 1ns / 1ps
module
top(clk,rst,buy_sig,pay_sig,coin_type,drink_type,cancel_sig,power_led,hold_led,take_led,charge_led,an,
seg,c_state,not_buy_led,not_pay_led);
input clk,rst,buy_sig,pay_sig,coin_type,drink_type,cancel_sig;//时钟，重置，购买信号，支付信
```

号，投币类型，饮料类型，取消信号

output power_led,hold_led,take_led,charge_led;//电源灯，占用灯，取货灯，退币灯

output [7:0]an,seg;//数码管相关

output not_buy_led,not_pay_led;

output [2:0]c_state;//现态

wire not_buy_dis,not_pay_dis;//持续 2s 的余额不足信号和禁止投币信号，用于数码管显示 nob（not buy）和 nop（not pay）

assign not_buy_dis=not_buy_led;//余额不足指示灯，在余额不足时进行购买会亮 2s

assign not_pay_dis=not_pay_led;//投币阈值指示灯，当余额大于 63 元时进行投币会亮 2s，同时余额不会增加

wire [7:0] drink_val,coin_val;//饮料价格，投币值

assign drink_val=drink_type?8'b00001010:8'b00000101;//drink_type=0,drink_val=2.5 元,drink_type=1,drink_val=5 元

assign coin_val=coin_type?8'b00010100:8'b00000010;//coin_type=1,coin_val=10 元,coin_type=0,coin_val=1 元

assign power_led=rst;//rst=1，电源灯亮，否则电源灯灭

wire buy_sig_f;

wire pay_sig_f;

fangdou fang1(clk,buy_sig,buy_sig_f);//购买按键的防抖动

fangdou fang2(clk,pay_sig,pay_sig_f);//投币按键的防抖动

wire hello_sig;//hello_sig=1,输出 hello，否则输出余额

wire [7:0] cur_val;//当前余额

wire charge_sig;

wire [1:0] can_buy;//可购买信号

FSM

```
f(clk,rst,cancel_sig,pay_sig_f,buy_sig_f,coin_type,cur_val,hold_led,charge_sig,hello_sig,can_buy,c_stat
e);
    cur_money
c(rst,charge_sig,pay_sig_f&cur_val<=8'b01111110,buy_sig_f,coin_val,drink_val,can_buy,cur_val);

    wire clk_dis;//数码管显示的刷新时钟
    divider #(50000)d1(clk,clk_dis);
    display dis(rst,clk_dis,hello_sig,charge_led,not_buy_dis,not_pay_dis,cur_val,an,seg);

    wire take_sig;
    assign take_sig=buy_sig_f&cur_val>=drink_val;//当收到购买信号并且余额大于饮料价格时可
以取饮料
    wire not_buy_sig;
    assign not_buy_sig=buy_sig_f&cur_val<drink_val;
    wire clk_led;
    divider #(100_000_000)d2(clk,clk_led);//clk_led 频率为 1Hz，控制灯亮
    led_inc take(clk_led,take_sig,take_led);//取饮料灯
    led_inc charge(clk_led,charge_sig,charge_led);//找零灯
    led_inc #(2)not_pay(clk_led,pay_sig_f&&cur_val>8'b01111110,not_pay_led);
    led_inc #(2)not_buy(clk_led,not_buy_sig,not_buy_led);//余额不足指示灯
    endmodule
```

（2）cur_money 底层模块 Verilog 代码

程序 2-2：cur_money 底层模块 Verilog 代码

```
`timescale 1ns / 1ps
module cur_money(rst,clear,pay_sig,buy_sig,coin_val,drink_val,can_buy,cur_val);
input pay_sig,buy_sig,rst,clear;
```

```
input [1:0] can_buy;
input [7:0] coin_val,drink_val;
output [7:0] cur_val;
reg [15:0] all_val,buy_val;//总投币值，总购买值
wire[15:0] sub_val;
assign sub_val=all_val-buy_val;
assign cur_val=sub_val[7:0];
initial begin
all_val=0;buy_val=0;
end
always @(negedge pay_sig,posedge clear,negedge rst) begin
    if(!rst||clear) all_val=0;//清空信号
    else all_val=all_val+{8'b00000000,coin_val};//付款信号
end
always @(negedge buy_sig,posedge clear,negedge rst) begin
    if(!rst||clear) buy_val=0;//清空信号
    else begin//购买信号
        if(can_buy==2'b10)//可购买 5 元饮料
          buy_val=buy_val+{8'b00000000,drink_val};
        else if(can_buy==2'b01&&drink_val==8'b00000101)//可购买 2.5 元饮料
          buy_val=buy_val+16'b0000000000000101;
        else buy_val=buy_val;
        end

end
endmodule
```

（3）seven_transfer 底层模块 Verilog 代码

程序 2-3：seven_transfer 底层模块 Verilog 代码

```
`timescale 1ns / 1ps
```

```
module seven_transfer(hello_sig,charge_sig,not_buy_sig,not_pay_sig,cur_val,num,out);
input hello_sig,charge_sig,not_buy_sig,not_pay_sig;
input [7:0] cur_val;
input [2:0] num;
output reg [7:0] out;
wire[3:0] one,ten;//个位，十位
assign one=cur_val[7:1]%10;
assign ten=cur_val[7:1]/10;
wire[7:0] pat_one,pat_ten;
pattern p1(one,pat_one);//余额的个位对应的数码管编码
pattern p2(ten,pat_ten);//余额的十位对应的数码管编码
always @(hello_sig,cur_val,num,pat_one,pat_ten) begin
    if(hello_sig==1) begin
        if(charge_sig==1) begin
        case(num)
        3'b000:out=8'b11000000;//0
        3'b001:out=8'b01000000;//0
        3'b010:out=8'b11000000;//0
        default:out=8'b11111111;
        endcase
    end
    else begin
         case(num)
         3'b000:out=8'b11000000;//O
         3'b001:out=8'b11000111;//L
         3'b010:out=8'b11000111;//L
         3'b011:out=8'b10000110;//E
         3'b100:out=8'b10001001;//H
         default:out=8'b11111111;//不显示
```

```
            endcase
        end
        end
        else begin//hello_sig==0
            if(not_buy_sig==1) begin
                case(num)
                3'b000:out=8'b10000011;//b
                3'b001:out=8'b11000000;//o
                3'b010:out=8'b11001000;//n
                default:out=8'b11111111;
                endcase
            end
            else if(not_pay_sig==1) begin
                case(num)
                3'b000:out=8'b10001100;//p
                3'b001:out=8'b11000000;//o
                3'b010:out=8'b11001000;//n
                default:out=8'b11111111;
                endcase
            end

            else begin
                case(num)
                3'b000:out=cur_val[0]?8'b10010010:8'b11000000;//有小数位输出 5，无小数位输
出 0
                3'b001:out=pat_one;//out=个位
                3'b010:out={1'b1,pat_ten[6:0]};//out=十位并去除小数点
                default:out=8'b11111111;
                endcase
```

```
            end
        end
    end
    endmodule
```

2.4 仿真过程

为了验证设计的正确性，对 top 顶层模块、cur_money 底层模块，dis 底层模块进行了仿真，具体过程如下：

（1）顶层模块仿真

目的：验证系统的基本功能，包括投币、购买商品、退币、数码管显示、指示灯部分。

输入：共 7 个，分别为：clk（时钟）,rst（重置）,buy_sig（购买信号）,pay_sig（投币信号）,coin_type（投币类型）,drink_type（购买商品类型）,cancel_sig（取消信号）

输出：共 9 个，分别为：power_led（电源指示灯）,hold_led（占用指示灯）,take_led（取货之指示灯）,charge_led（找零指示灯）,an（数码管选择）,seg（数码显示管）,c_state（售货机当前状态）,not_buy_led（余额不足指示灯）,not_pay_led（禁止投币指示灯）

注：为了仿真便于观察，子模块使用的时钟不再分频，与主模块时钟相同。

程序 2-4：顶层模块仿真文件

```
`timescale 1ns / 1ps
module top_test();
reg clk,rst,buy_sig,pay_sig,coin_type,drink_type,cancel_sig;
wire power_led,hold_led,take_led,charge_led;
wire [7:0]an,seg;
wire [2:0]c_state;
wire not_buy_led,not_pay_led;
top
t(clk,rst,buy_sig,pay_sig,coin_type,drink_type,cancel_sig,power_led,hold_led,take_led,charge_led,an,se
```

```
g,c_state,not_buy_led,not_pay_led);
    initial begin
    clk<=0;
    rst<=1;
    buy_sig<=0;
    pay_sig<=0;
    coin_type<=0;
    drink_type<=0;
    cancel_sig<=0;
    #5 rst=0;
    #5 rst=1;
    coin_type=0;drink_type=0;
    #50 pay_sig=1;#5 pay_sig=0;//投币 1 元*3 次
    #5 pay_sig=1;#5 pay_sig=0;
    #5 pay_sig=1;#5 pay_sig=0;

    #10coin_type=1;
    #10 pay_sig=1;#5 pay_sig=0;//投币 10 元
    //此时余额 23 元
    #5 buy_sig=1;#5 buy_sig=0;//购买 2.5 元商品
    #10drink_type=1;
    #20 buy_sig=1;#5 buy_sig=0;//购买 5 元商品
    #20 buy_sig=1;#5 buy_sig=0;
    #20 buy_sig=1;#5 buy_sig=0;//此时余额不足
    #20 cancel_sig=1;//退币
    #5 cancel_sig=0;

    #50 pay_sig=1;#5 pay_sig=0;//重启系统
    #10 rst=0;
```

```
#10 rst=1;
end
always begin
    #1.3 clk=!clk;
end
endmodule
```

主模块仿真图如图 2-8、图 2-9、图 2-10 所示。

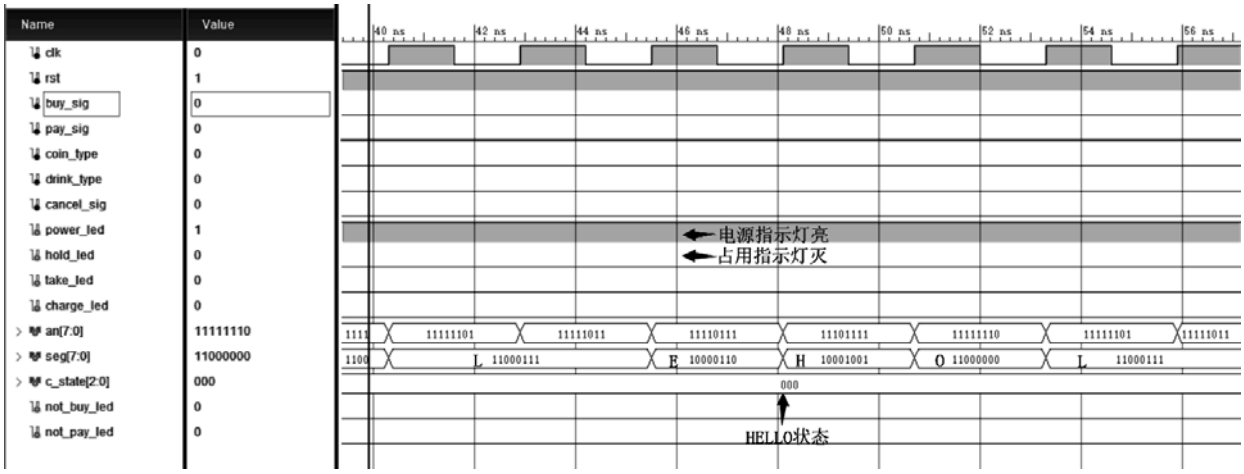

图 2-8　top 主模块仿真测试图

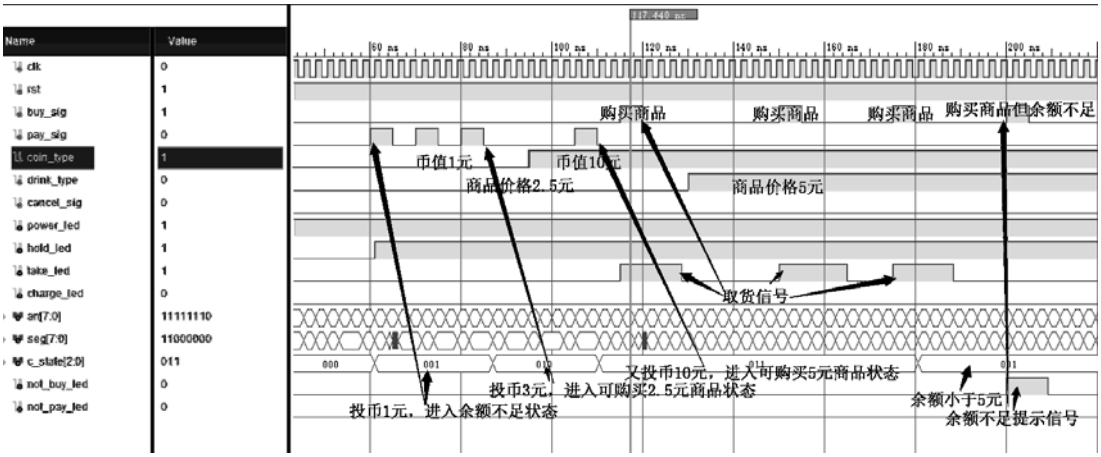

图 2-9　top 主模块仿真测试图

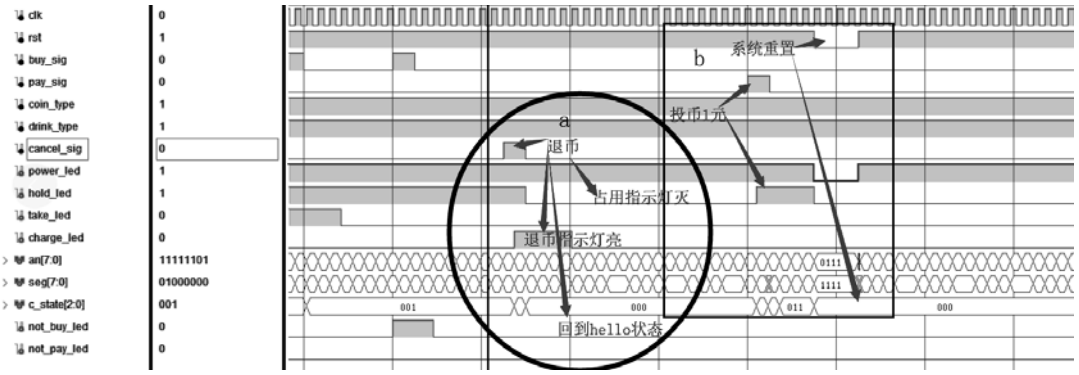

图 2-10　top 主模块仿真测试图

仿真结果说明：

A.如图 2-8 所示，首先观察 hello 状态下的数码管显示(40ns-56ns)，预期 seg 的内容为 HELLO 的数码管编码，电源指示灯亮，占用指示灯灭，仿真结果发现对应 HELLO 的数码管编码，符合预期

B.如图 2-9 所示，在 hello 状态下投币 3 个 1 元（60ns-84ns），1 个 10 元(104ns-108ns)，预期结果状态机在在投币 1 元后进入 001 状态且占用指示灯亮，在投币 3 元后进入 010 状态，在投币 10 元后进入 011 状态。观察仿真图像，可知状态机的状态正确改变，以及在第一次投币后占用指示灯亮，与预期结果符合。

C.如图 2-9，购买一件 2.5 元商品（112ns-128ns），3 件 5 元商品(140ns-210ns)，预期结果在前 3 件商品购买后出货指示灯亮，在购买第四件商品时余额不足，由仿真图像观察到出货指示灯亮 3 次（114ns-128ns，148ns-162ns，174ns-188ns），最后一次购买时余额不足导致余额不足指示灯亮，出货指示灯灭（200ns-208ns）。

D.如图 2-10，如图中 a 部分，执行退币操作后预期占用指示灯灭，退币指示灯亮，状态机回到 hello 状态，观察仿真结果，与预期符合。如图中 b 部分，预期在 rst 信号的下降沿自动售货机回到初始状态，仿真图像符合预期。

（2）cur_money 底层模块仿真

目的：验证在有投币信号、购买信号、清空信号钱币系统的变化。

输入：rst（重置信号），clear（清空信号），pay_sig（投币信号），buy_sig（购买信号），coin_val（投币类型），drink_val（商品类型），can_buy（可购买的商品类型）

输出：cur_val（当前余额）

程序 2-5：cur_money 底层模块仿真文件

```
`timescale 1ns / 1ps
module cur_money_test();
reg rst,clear,pay_sig,buy_sig;
reg [1:0]can_buy;
reg [7:0]drink_val,coin_val;
wire [7:0] cur_val;
cur_money test(rst,clear,pay_sig,buy_sig,coin_val,drink_val,can_buy,cur_val);
initial begin
rst=1;
pay_sig=0;buy_sig=0;coin_val=0;drink_val=8'b00000101;clear=0;can_buy=00;
#5 rst=0;//重置系统
#5 rst=1;
#5 coin_val=8'b00000010;
#5 pay_sig=1;#5 pay_sig=0;//投币 1 元
#5 pay_sig=1;#5 pay_sig=0;//投币 1 元
#5 pay_sig=1;#5 pay_sig=0;//投币 1 元
#5 coin_val=8'b00010100;#5 pay_sig=1;#5 pay_sig=0;//投币 10 元
#5 pay_sig=1;#5 pay_sig=0;//投币 10 元
#20 can_buy=2'b10;//可购买 5 元商品
#5 buy_sig=1;#5 buy_sig=0;//购买 2.5 元饮料
#5 drink_val=8'b00001010;#5 buy_sig=1;#5 buy_sig=0;//购买 5 元饮料
#5 can_buy=2'b00;//不可购买商品
#5 buy_sig=1;#5 buy_sig=0;//购买 5 元饮料不成功
```

```
#10 clear=1;#5 clear=0;//清空
end
endmodule
```

cur_money 底层模块仿真图如图 2-11 所示。

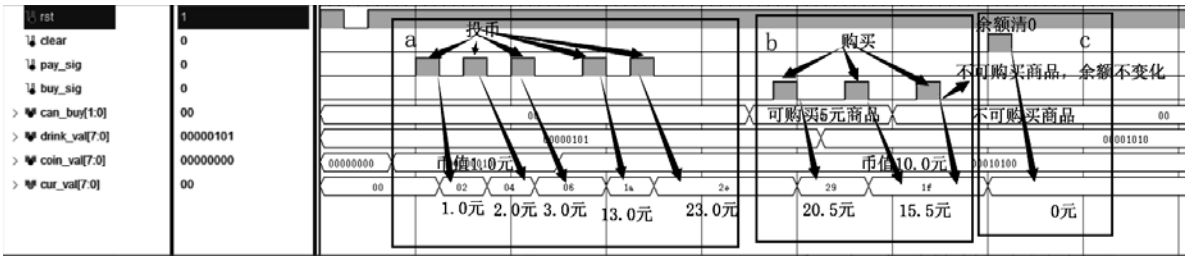

图 2-11　cur_money 底层模块仿真测试图

仿真结果说明：

A.首先测试投币功能，如图 2-11a 部分所示，进行 3 次 1 元投币和两次 10 元投币，最终结果为 2e，即二进制数 00101110B,将最后一位视为小数位，对应 10 进制数 23.0 元

B.如图 2-11b 部分所示，预期结果在 can_buy=10（可购买 5 元商品）情况下，购买一件 2.5 元商品与一件 5 元商品，余额变为 15.5 元。在 can_buy=00(不可购买商品)情况下，购买 5 元商品余额不变化，仿真结果与预期相符。

C.如图 2-11c 部分，预期清空信号 clear 出现上升沿时，余额清零，仿真结果与预期相符。

2.5 主要问题及解决方法

（1）故障 1

图 2-12 综合报错

问题描述：综合报错，如图 2-12 所示

问题分析：同一个 reg 类型不能再两个 always 语句种被赋值

解决方法：同一个 reg 只在一个 always 语句中赋值

给出修改后的实例

```
always @(negedge pay_sig,posedge clear,negedge rst) begin
    if(!rst||clear) all_val=0;//清空信号
    else all_val=all_val+{8'b00000000,coin_val};//付款信号
end
always @(negedge buy_sig,posedge clear,negedge rst) begin
    if(!rst||clear) buy_val=0;//清空信号
    else begin//购买信号
         if(can_buy==2'b10)//可购买5元饮料
           buy_val=buy_val+{8'b00000000,drink_val};
         else if(can_buy==2'b01&&drink_val==8'b00000101)//可购买
           buy_val=buy_val+16'b0000000000000101;
         else buy_val=buy_val;
         end
```

图 2-13 修改后实例

（2）故障 2

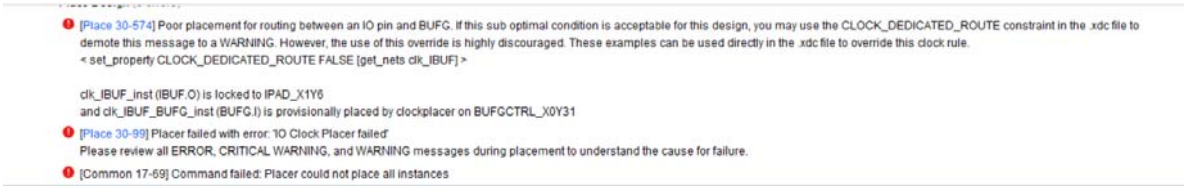

图 2-14 生成比特流报错

问题描述：生成比特流失败，报错提示如图 2-14 所示

问题分析：这个错误是由于将某一变量命名为“clk”，在 Nexys4 开发板上，“clk”为时钟信号，默认是和“E3”这个引脚连接在一起的，如果将其他引脚和“clk”相连，就会报这个错误

解决方法：在约束文件中增加<set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets buy_sig_IBUF]>”

给出修改后的实例

```
set_property PACKAGE_PIN T16 [get_ports buy_sig]
    set_property IOSTANDARD LVCMOS33 [get_ports buy_sig]
    set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets buy_sig_IBUF]
```

图 2-14 修改后实例

（3）故障 3

问题描述：开发板按下一次投币按键有时会增加多次余额

问题分析：按动按键过程中会产生抖动

解决方法：增加防抖动模块，使用计时器判断如果抖动时间在一定范围内，系统不认为是一个正确的按键。

（4）故障 4

问题描述：投币一次余额会增加多次

问题分析：投币信号为一个电平信号，在电平信号为 1 期间每经过一个时钟余额就会增加一次

解决方法：使用时序逻辑电路设计，使余额增加只在投币信号的边沿触发。

给出修改后的实例

```
always @(negedge pay_sig,posedge clear,negedge rst) begin
    if(!rst||clear) all_val=0;//清空信号
    else all_val=all_val+{8'b00000000,coin_val};//付款信号
end
always @(negedge buy_sig,posedge clear,negedge rst) begin
    if(!rst||clear) buy_val=0;//清空信号
    else begin//购买信号
         if(can_buy==2'b10)//可购买5元饮料
           buy_val=buy_val+{8'b00000000,drink_val};
         else if(can_buy==2'b01&&drink_val==8'b00000101)//可购买2.5元饮料
           buy_val=buy_val+16'b0000000000000101;
         else buy_val=buy_val;
         end
```

图 2-15 修改后实例

2.6 功能测试

开发板各个按键功能如图 2-16 所示：

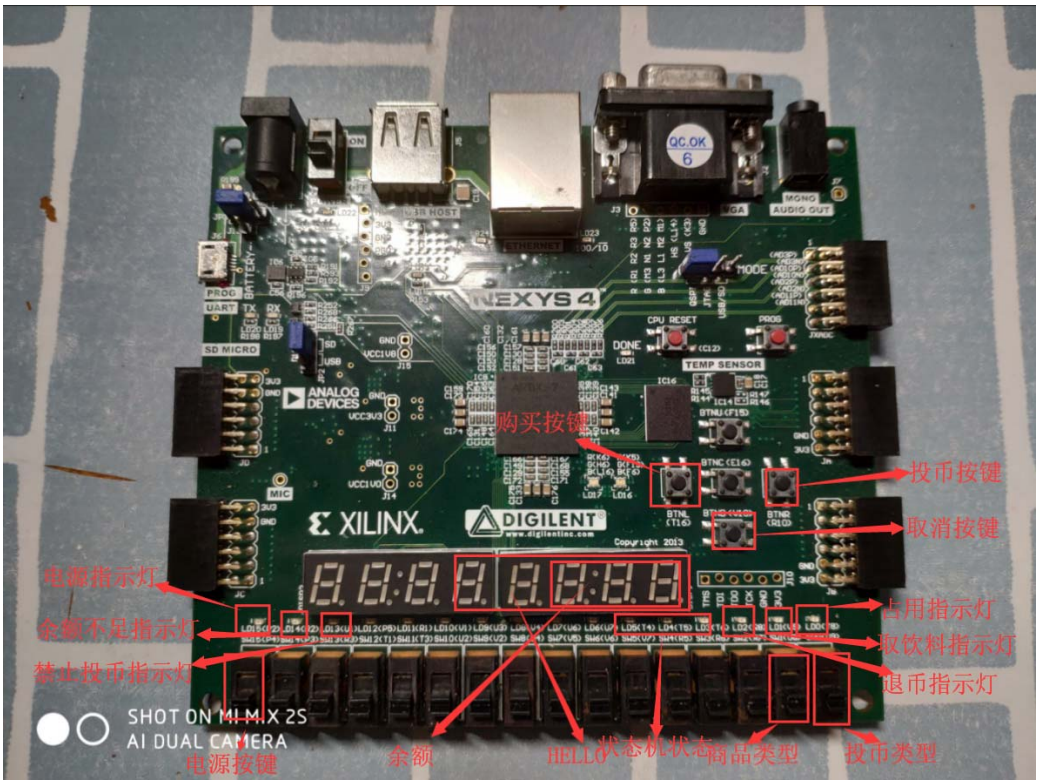

图 2-16 开发板按键功能

共进行了 4 项功能测试，它们分别为：投币功能测试，购买功能测试，退币功能测试，电源开关测试。

（1）投币功能测试

开机进入 Hello 状态，选择投币类型为 1 元，按下投币按键，预期结果为数码管显示由 HELLO 变为 1.0，占用指示灯亮，状态机状态变为 01.测试结果如图 2-17 和图 2-18 所示。与预期结果相符。

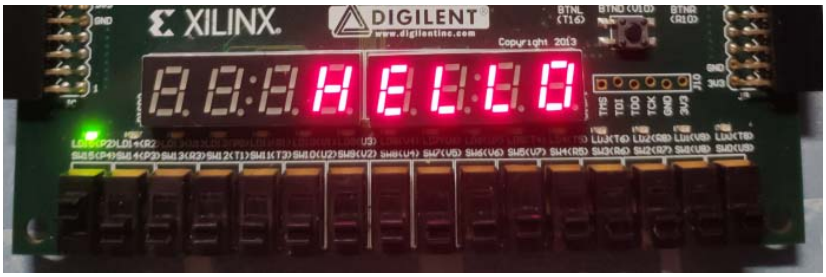

图 2-17 初始状态

图 2-18 投币 1 元

将币值设置为 10 元，按下投币按键，预期结果余额变为 11 元，状态机状态变为 11，测试结果如图 2-19 所示，与预期结果相符。

图 2-19 再投币 10 元

当当前余额大于 63 元时不再接受投币并给出提示，预期结果禁止投币指示灯亮 2s，数码管显示 nop(no pay),测试结果如图 2-20 和图 2-21 所示，与预期结果相符。

图 2-20 当前余额

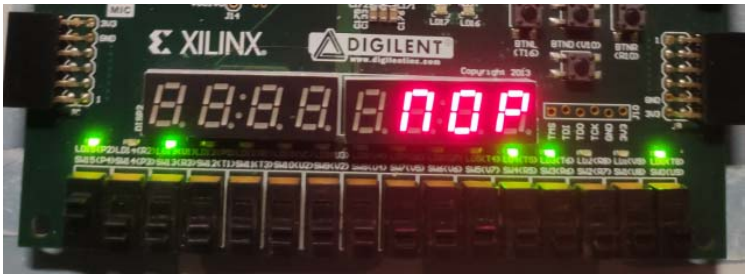

图 2-21 禁止投币

（2）购买功能测试

在投币 11 元的情况下，购买 2.5 元商品，预期结果余额变为 8.5 元，取饮料指示灯亮 5s，测试结果如图 2-22 所示，与预期符合。

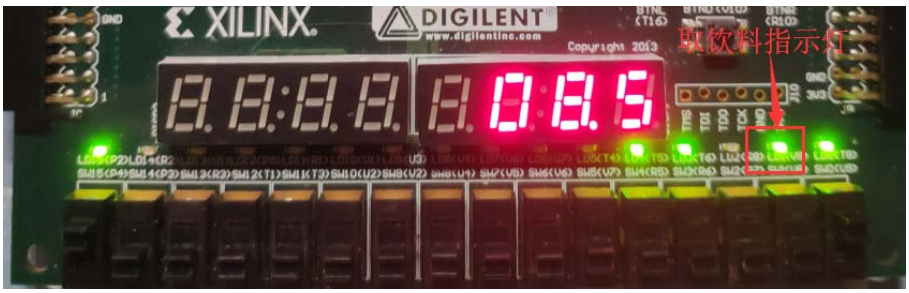

图 2-22 购买 2.5 元商品功能测试

在余额为 3 元的情况下，购买 5 元商品，余额不足指示灯亮 2s，数码管显示 nob（no buy），测试结果如图 2-23，图 2-24 所示，与预期符合。

图 2-23 余额不足

图 2-24 余额不足

（3）退币功能测试

在投币 3 元的情况下，按下取消按键，预期结果余额变为 0 元，退币指示灯亮 5s，测试结果

如图 2-25 所示，与预期符合。

图 2-25 退币功能测试

（4）电源开关测试

关闭电源开关，预期结果所有指示灯和数码管灭掉，测试结果如图 2-26 所示，与预期结果符合。

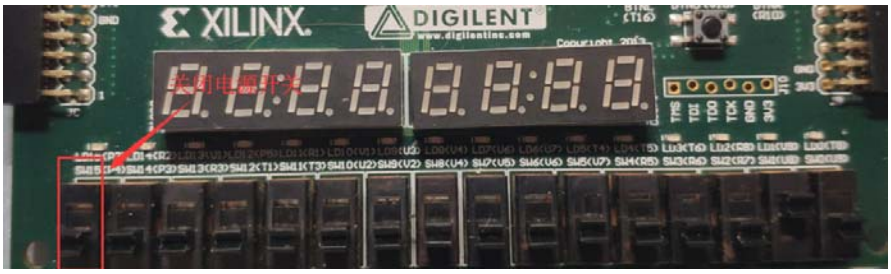

图 2-26 电源开关测试

3 总结与心得

3.1 课设总结

完成了自动售货机的设计，我主要做了以下工作：

（1）首先设计出自动售货机的状态转移图，在状态转移图的基础上设计出 FSM 模块。

（2）完善自动售货机的总体设计，增加余额模块、显示模块、分频器模块的设计。

（3）分别对状态机模块、余额模块进行仿真测试。

（4）对显示模块进行下板子验证。

（5）各个子模块基本实现，在 top 主模块进行连接，修改错误，调试 bug，下板子验证，基本实现自动售货机主要功能。

（6）在测试过程中发现按键有抖动现象，增加防抖动输入缓冲模块，并增加余额不足提示、禁止投币提示等功能。

（7）将修改后的程序下板子验证。

（8）撰写课程设计报告。

3.2 课设心得

（1）比写程序更重要的时程序的设计，在本次实验中，对状态机的设计、对模块的划分都必须提前设计好，而在看到题目后直接上手写程序的方法不可取，否则在写到一半发现行不通，从头开始就该就会极大增加工作量乃至要推翻重写。

（2）模块化的优点是便于调试，在本次实验中我深有感触，每写完一个模块对齐进行调试，这样很容易就定位到错误出现在哪里，极大的提高了编程效率。

（3）在写程序时要有良好的编程风格，比如变量的命名、always、if 语句下的缩进以及对程

序关键处进行注释。

4 参考文献

教学参考书：

[1]欧阳星明，于俊清. 数字逻辑. 武汉：华中科技大学出版社，2012

[2]巴斯克（Bhasker，J）著，夏宇闻，甘伟译.北京:北京航空航天大学出版社,2008.9

[3]CSDN 博客: https://blog.csdn.net/wanruoqingkong/article/details/40261379