数字电路与逻辑设计
课程设计报告

报　　告　　人：　洪志远　
实验指导教师：　石宣化　
报告批阅教师：　

计算机科学与技术学院

2018 年 **10** 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）
CS1607	洪志远	U201613570	

（学生填写）

（1）主要工作的描述

a. 选题和分析题目要求。分析状态机状态，绘制状态图；分析模块划分，绘制模块图（草图）

b. 在实现分频模块的编码时，输出三种频率：1Hz、10Hz、1000Hz，分别用于倒计时器、同步时钟、数码管刷新

c. 在实现数码管显示模块时，输入为币值，输出为数码管 SEG、AN。使用分频器输出的 1000Hz 的频率，轮流刷新 8 个数码管。8 个数码管都可显示内容。

d. 在实现状态机时，将 3 个状态纳入状态机，分别是是否显示 HELLO、退币指示、取饮料指示。后两者同时和定时器相连，亮 20 个时钟周期（约 2s）后熄灭

e. 顶层模块调用上述三个底层模块，并用线网类型的变量将其相连。与相应的引脚绑定。

（2）难点、亮点

a. 由于是转专业学生，之前只用过 ISE，对 Vivado 很不熟练。

b. 状态机中修改状态经常遇到 Multi-Driver 的错误；显示模块有字母、数字、带小数点的数字，较为复杂；状态机的设计较难。在设计中考虑了拓展性和亮点。

课设进度记录（学生填写）

日期	进度
10 月 8 日	确定选题为“自动销售机控制器设计”
10 月 8 日	分析题目要求
10 月 8 日	绘制状态图、流程图
10 月 8 日	完成分频器和数码管的 HELLO 及其仿真
10 月 9 日	上午完成数码管数字、小数的显示及其仿真
10 月 9 日	下午完成币值加法和减法及其仿真
10 月 9 日	由于增加亮点，修改状态图的细节
10 月 9 日	完成状态机及其仿真
10 月 9 日	模块组装，完成设计

实验平台故障记录（学生填写，请注明实验平台的编号

c. 亮点： 1. 退币独立显示 2. RGB 灯表示可否购买某饮料 3. 可连续购买 4. 关机自动退币 5. 退币时仍可投币，系统继续进入等待购买状态 6. 购物出货时仍可投币	

华 中 科 技 大 学 课 程 设 计 报 告

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_____）

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”。

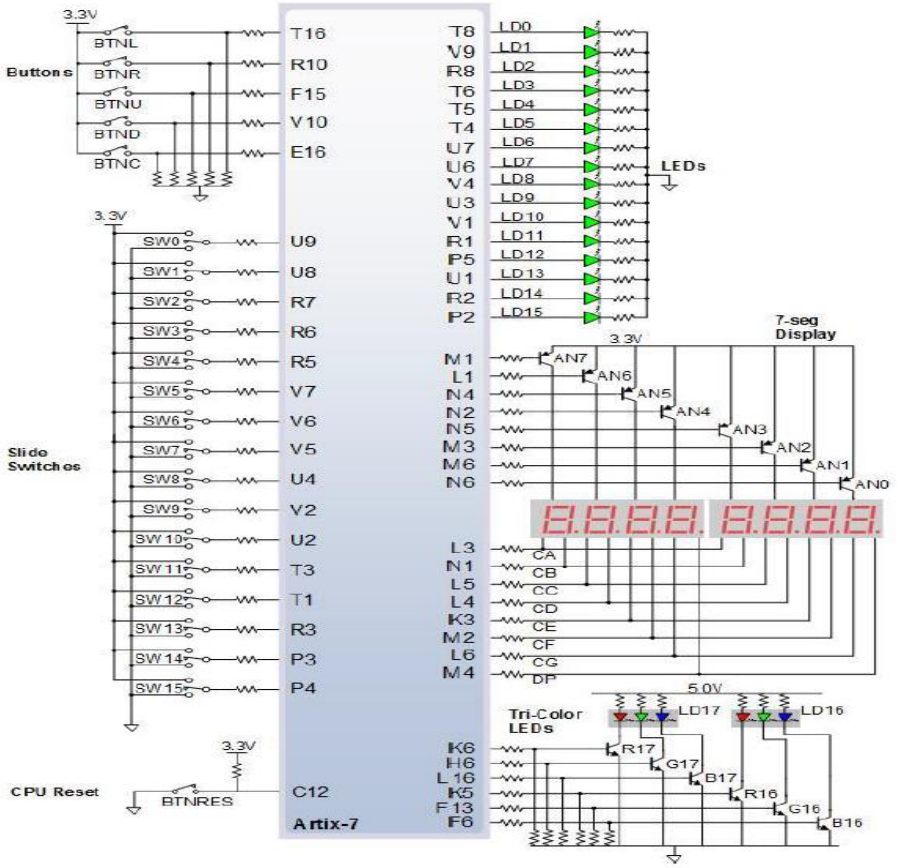

图 1-1 Nexys4 通用 I / O 设备

2. 自动销售机控制器设计方案

2.1 内容

2.1.1 基本内容

（1） **采用 FSM 设计并利用 Nexys4 开发板实现一个饮料自动售货机，它具有下述功能：**

a. 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；最多投入 99 元人民币，售货机最多显示 99.5 元人民币。

b. 销售机处于售卖状态时显示“Hello” 状态， 当投入 1 元时， 机器会进入余额不足的状态， 相应的饮料指示灯为红色，直到投入的金额大于 2. 5 元为止，此时相应的饮料指示灯为绿色。 如果一次投入 10 元， 则可以选择所有的产品，所有的饮料指示灯都为绿色， 否则就只能选择 2. 5 元的产品；

c. 完成选择后， 将会卖出商品并且找回剩余的零钱， 随后（约）， 机器又将返回售卖状态。

（2） **根据功能描述画出系统结构框图如下：**

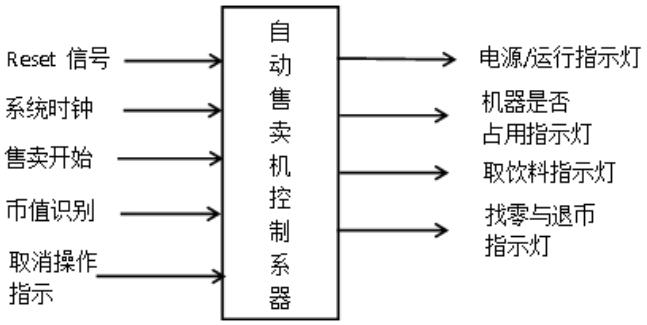

图 2-1 系统结构图

2.1.2 拓展功能

（1） **可以连续购买**：用户在投币数量到达某饮料的价格时即可购买，用户需要主动点击退币或者取消按钮才能退币

（2） **关机自动退币**：点击关机时，会自动退出已经投入的钱币

（3） **退币金额独立显示**：退币金额显示在左边 4 个数码管上，同时剩余金额置零

2.2 设计思路

本设计要求顶层采用原理图设计，各底层均采用 Verilog HDL 设计，其模块图参见图 2-1 所示。自动售货机的电源开关 Power 作为总开关，当按下 Power 时，切换开关机状态。当重新开机时，会 Reset 整个系统的所有状态，进入 hello 界面。

（1） 顶层模块的输入：

a. clk 为硬件时钟，Nexys4 开发板上是 100Mhz。

b. coin_type 为 switch 控制的比值识别输入，当该值为 0 时为 1 元，当该值为 1 时为 10 元。

c. drink_1、drink_2 分别为 2.5 元饮料和 5 元饮料的购买按钮。在金额足够的情况下按下该按扭即立刻购买相应的饮料。

d. insert_coin 为投币脉冲，每投一次币，便会有一个上升沿。

e. power 为电源键，电源键可以开关整个设备。当存在余额时投币，即会自动退出金额。

f. cancel 取消操作，当用户投币后不再希望继续购物，按此按钮可以退出余额。退币时左边四个数码管显示退币金额（例如：15.50），右边显示当前余额，固定为 0.00。

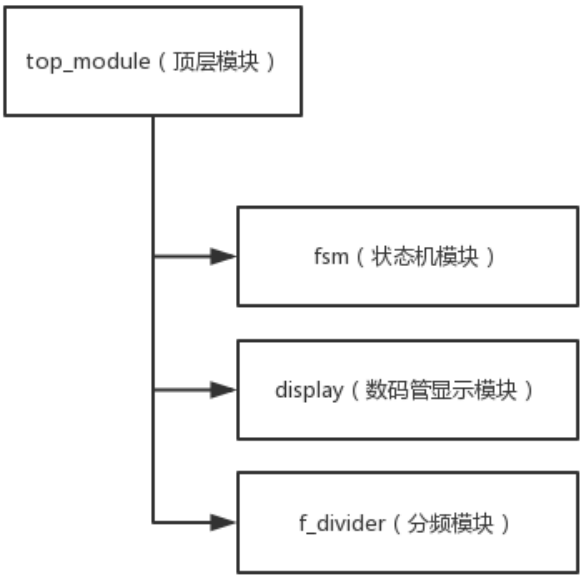

图 2-1 自动销售机控制器的模块图

（2） 顶层输出：

a. drink_1_red、drink_2_red、drink_1_green、drink_2_green 分别驱动 RGB1、RGB2 用于显示是否可以购买相应的饮料：红色代表不能购买，绿色代表可以购买。

b. error 为错误指示灯，当余额不足而尝试购买时，此输出驱动的 LED 灯会亮起。

c. power_led 为电源指示灯，当系统处于开机状态时，此灯亮起。

d. in_use 为占用指示灯，此灯亮起时机器被占用。

e. take_drink、take_momey 为取饮料和取币指示灯，当饮料或余额被被退出，此灯亮 3s 左右，之后熄灭。用于提示用户及时取走饮料和钱币。

f. AN 和 SEG 控制数码管的显示。

（3） FSM 的状态图：

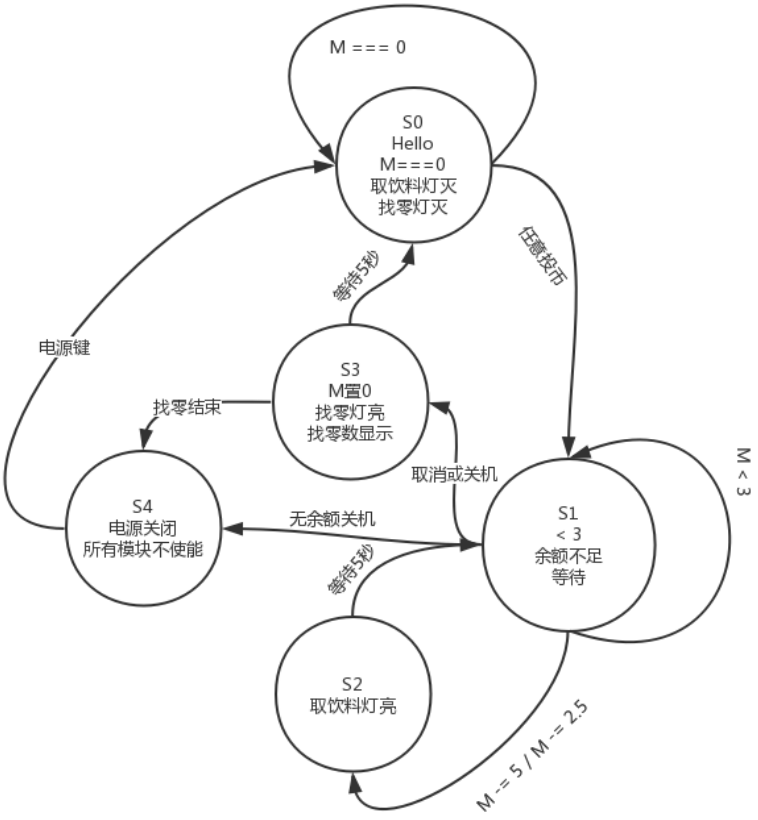

图 2-2 FSM 状态机状态转移图

在上述状态图中，存在 5 种状态：S0 待机状态，S1 等待操作状态，S2 取饮料状态，S3 找零状态，S4 关机状态。

状态的流转由按钮控制和余额的数量控制。

2.2.1 顶层模块设计

顶层模块使用线网类型的变量用于连接各个底层模块。同时将底层模块的输出进行逻辑运算得到控制各个引脚的输出。

具体地，顶层模块（top_module）实例化底层模块 f_divider、display、FSM，并将 f_divider 的三个时钟分别用线网型变量连接到 display、FSM。

2.2.2 分频模块设计

分频模块的作用是将系统硬件时钟分频得到频率较低的三种频率：1. 倒计时 1Hz 的频率。2. 用于同步按钮监测和防抖的时钟 100Hz。3. 用于驱动数码管显示的时钟 1000Hz。

2.2.3 显示模块设计

显示模块有三种状态：

(1) “HELLO”显示模块

(2) 正常余额显示

(3) 特殊余额显示，即带有退款金额的显示

这三种状态由模块的 input 控制。

除此之外，display 模块接受来自分频模块的 clk_display 时钟信号，同时维护 [3:0]scan_count 变量作为当前显示位，译码为 AN 的值，以低电平有效的方式，控制当前需要显示的数码管灯。

2.2.4 状态机模块设计

由图 2-3 可知，该系统有 5 种状态。使用一个[2:0]state 变量作为状态标志。

该模块可接受各个按键的值，在同步时钟的上升沿，判断哪个键被按下（带有按键防抖）。根据按下的按键，修改 state 的值，以此控制状态流转。余额、退款金额的计算在该同步时钟驱动的 always@ 块内。

同时，该模块内部集成了倒计时控制器，同样为同步时钟控制。维护一个计数器，当该计数器的值达到某值时（即计时结束），进行状态流转。

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。

自动售货机控制器顶层原理图，参见图 2-4 所示。

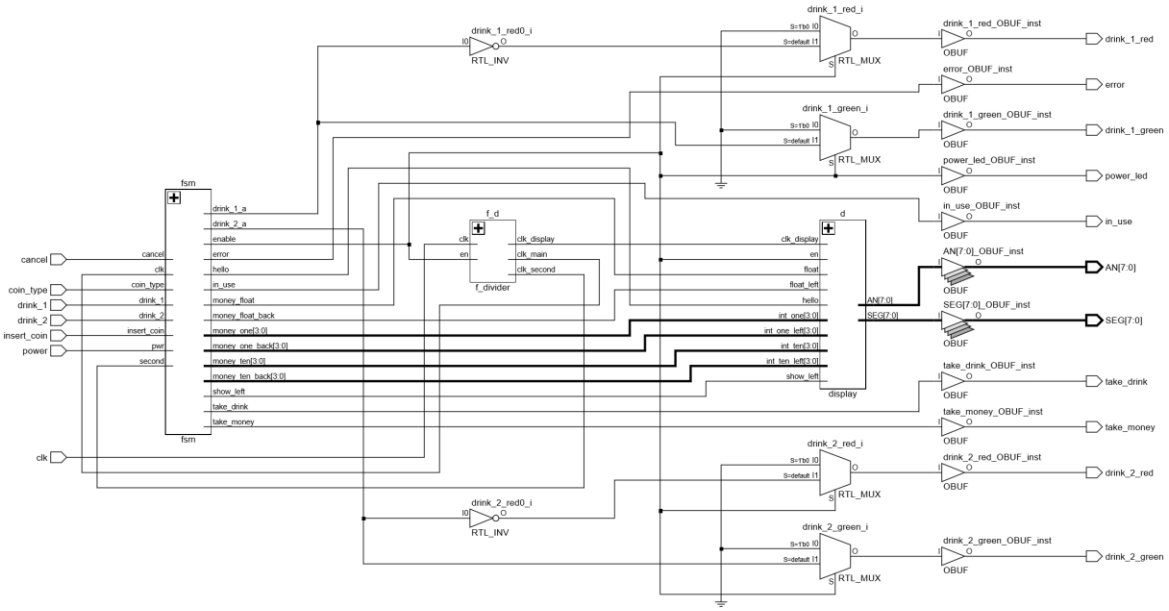

图 2-3 自动售货机控制器顶层原理图

（1）自动售货机控制器顶层模块（例如：main 用于整合整个系统）

程序 2-1：自动售货机控制器顶层模块 Verilog 代码

```
`timescale 1ns / 1ps
module top_module(
    input  power,
    input  coin_type,
    input  insert_coin,
    input  drink_1,
    input  drink_2,
    input  cancel,
    input  clk,
    output reg power_led,
```

```
    output error,
    output take_drink,
    output take_money,
    output in_use,
    output reg drink_2_green, // drink 2 可购买
    output reg drink_2_red, // drink 2 不可购买
    output reg drink_1_green, // drink 1 可购买
    output reg drink_1_red,  // drink 1 不可购买
    output [7:0] AN,
    output [7:0] SEG
    );
    wire second_clk;
    wire display_clk;
    wire main_clk;
    wire hello;
    wire [3:0]money_ten;
    wire [3:0]money_one;
    wire money_float;
    wire show_left;
    wire [3:0]money_ten_back;
    wire [3:0]money_one_back;
    wire money_float_back;
    wire drink_2_a;
    wire drink_1_a;
    wire enable;
    fsm fsm(
        power,
        main_clk,
        second_clk,
        coin_type,
        insert_coin,
        drink_1,
        drink_2,
        cancel,
        enable,
        take_drink,
        take_money,
        in_use,
        error,
        drink_2_a,
        drink_1_a,
        hello,
        money_ten,
        money_one,
        money_float,
        show_left,
        money_ten_back,
        money_one_back,
        money_float_back
    );
    f_divider f_d(
        enable,
        clk,
        display_clk,
        second_clk,
        main_clk
    );
    display d(
        enable,
```

```
        hello,
        money_ten,
        money_one,
        money_float,
        show_left,
        money_ten_back,
        money_one_back,
        money_float_back,
        display_clk,
        AN,
        SEG
);
    always @(drink_1_a) begin
        drink_1_green = drink_1_a;
        drink_1_red = ~drink_1_a;
        if (!enable) begin
            drink_1_green = 0;
            drink_1_red = 0;
        end
    end
    always @(drink_2_a) begin
        drink_2_green = drink_2_a;
        drink_2_red = ~drink_2_a;
        if (!enable) begin
            drink_2_green = 0;
            drink_2_red = 0;
        end
    end
    always @(enable)
        power_led = enable;
endmodule
```

（2）自动售货机控制器底层模块 Verilog 代码

程序 2-2：自动售货机控制器底层模块 Verilog 代码

```
`timescale 1ns / 1ps
module fsm(
    input pwr,
    input clk,
    input second,
    input coin_type,
    input insert_coin,
    input drink_1,
    input drink_2,
    input cancel,
    output reg enable = 1,
    output reg take_drink = 0,
    output reg take_money = 0,
    output reg in_use = 0,
    output reg error = 0,
    output reg drink_2_a = 0,
    output reg drink_1_a = 0,
    output reg hello = 1,
    output reg [3:0]money_ten = 0,
    output reg [3:0]money_one = 0,
```

```
    output reg money_float = 0,
    output reg show_left = 0,
    output reg [3:0]money_ten_back = 0,
    output reg [3:0]money_one_back = 0,
    output reg money_float_back = 0
);
    reg [2:0]state = 4;
    reg [5:0]time_to_show_money = 6'b111111;
    reg [5:0]time_to_show_drink = 6'b111111;
    reg power_on = 0;
    reg [3:0]throttle_reg = 12;
    always @(state) begin
        case (state)
            0: begin
                hello = 1;
                enable = 1;
                show_left = 0;
                take_drink = 0;
                take_money = 0;
            end
            1: begin
                hello = 0;
                enable = 1;
                show_left = 0;
                take_drink = 0;
                take_money = 0;
            end
            2: begin
                hello = 0;
                enable = 1;
                take_drink = 1;
                show_left = 0;
                take_money = 0;
            end
            3: begin
                hello = 0;
                enable = 1;
                show_left = 1;
                take_money = 1;
                take_drink = 0;
            end
            4: begin
                enable = 0;
                hello = 0;
                show_left = 0;
                take_money = 0;
                take_drink = 0;
            end
        endcase
    end
    always @(posedge clk) begin
        error = 0;
```

```
        if (state == 3) begin
            time_to_show_money = time_to_show_money - 1;
            if (time_to_show_money == 0) begin
                if (money_ten != 0 || money_one != 0 || money_float != 0)
                    state = 1;
                else begin
                    if (power_on)
                        state = 0;
                    else
                        state = 4;
                end
                time_to_show_money = 63;
            end
        end
        if (state == 2) begin
            time_to_show_drink = time_to_show_drink - 1;
            if (time_to_show_drink == 0) begin
                if (money_ten != 0 || money_one != 0 || money_float != 0)
                    state = 1;
                else
                    state = 0;
                time_to_show_drink = 63;
            end
        end
        if (insert_coin && power_on && throttle_reg == 0) begin
            if (state != 3 && state != 2)
                state = 1;
            if (money_ten == 9 && money_one == 9) begin
                // retrun;
            end else begin
                if (coin_type == 0) begin
                    if (money_one == 9) begin
                        money_one = 0;
                        money_ten = money_ten + 1;
                    end else begin
                        money_one = money_one + 1;
                    end
                end else begin
                    if (money_ten != 9)
                        money_ten = money_ten + 1;
                end
            end
            throttle_reg = 12;
        end else if (cancel && power_on && throttle_reg == 0) begin
            if (state == 1) begin
                state = 3;
                money_ten_back   = money_ten;
                money_one_back   = money_one;
                money_float_back = money_float;
                money_ten = 0;
                money_one = 0;
                money_float = 0;
```

```
            end
            throttle_reg = 12;
        end else if (drink_1 && power_on && throttle_reg == 0) begin
            if (money_one >= 3) begin
                if (money_float) begin
                    money_float = 0;
                    money_one = money_one - 2;
                end else begin
                    money_float = 1;
                    money_one = money_one - 3;
                end
                state = 2;
            end else if (money_one == 2 && money_float == 1) begin
                money_one = 0;
                money_float = 0;
                state = 2;
            end else if (money_ten >= 1) begin
                money_ten = money_ten - 1;
                if (money_float) begin
                    money_one = money_one + 10 - 2;
                    money_float = 0;
                end else begin
                    money_one = money_one + 10 - 3;
                    money_float = 1;
                end
                state = 2;
            end else
                error = 1;
            throttle_reg = 12;
        end else if (drink_2 && power_on && throttle_reg == 0) begin
            if (money_one >= 5) begin
                money_one = money_one - 5;
                state = 2;
            end else if (money_ten >= 1) begin
                money_ten = money_ten - 1;
                money_one = money_one + 10 - 5;
                state = 2;
            end else
                error = 1;
            throttle_reg = 12;
        end else if (pwr && throttle_reg == 0) begin
            power_on = ~power_on;
            if (power_on) begin
                if (state == 4)
                    state = 0;
            end else begin
                if (money_ten != 0 || money_one != 0 || money_float != 0) begin
                    state = 3;
                    money_ten_back   = money_ten;
                    money_one_back   = money_one;
                    money_float_back = money_float;
                    money_ten = 0;
```

```
                money_one = 0;
                money_float = 0;
            end else begin
                state = 4;
                money_ten = 0;
                money_one = 0;
                money_float = 0;
            end
        end

        throttle_reg = 12;
    end
    if (throttle_reg > 0) begin
        throttle_reg = throttle_reg - 1;
    end
end

always @(money_ten or money_one or money_float) begin
    if (money_ten != 0 || money_one != 0 || money_float != 0)
        if (money_ten > 0 || money_one >= 5) begin
            drink_2_a = 1;
            drink_1_a = 1;
        end else if (money_one >= 3 || (money_one == 2 && money_float == 1)) begin
            drink_1_a = 1;
            drink_2_a = 0;
        end else begin
            drink_2_a = 0;
            drink_1_a = 0;
        end
    else begin
        drink_2_a = 0;
        drink_1_a = 0;
    end
end
always @(hello) begin
    in_use = ~hello;
    if (!enable)
        in_use = 0;
end
endmodule
```

2.4 仿真过程

为了验证设计的正确性，对顶层模块、状态机模块等模块进行了仿真，具体过程如下：

（1）顶层模块仿真

目的：验证系统的基本功能，例如是否能够正确的将状态机模块的输出值运算并输出、主要逻辑是否正确、LED 灯的显示是否正确等

输入：共 7 个，分别为：power（电源按钮），coin_type（比值类型），insert_coin（投币脉冲），drink_1（买饮料一脉冲），drink_2（买饮料二脉冲），clk（系统基础时钟）；

输出：共 11 个，分别为：power_led（电源灯），error（错误），take_drink（取饮料指示灯），take_money（取币指示灯），in_use（占用指示灯），drink_1_green（饮料一可以购买指示灯 green 部分），drink_2_green（饮料二可以购买指示灯 green 部分），drink_1_red（饮料一可以购买指示灯 red 部分），drink_2_red（饮料二可以购买指示灯 red 部分），AN、SEG（数码管输出）

程序 2-3：顶层模块仿真文件

```
`timescale 1ns / 1ps
module top_module_sim();
    reg clk = 0;
    reg coin_type = 0;
    reg insert_coin = 0;
    reg power = 0;
    reg drink_1 = 0;
    reg drink_2 = 0;
    reg cancel = 0;

    initial begin

        #100 power = 1;
        #100 power = 0; // push power button.

        #2000; // wait, for checking HELLO

        #100 insert_coin = 1;
        #100 insert_coin = 0; // push insert_coin button

        #100 drink_1 = 1;
        #100 drink_1 = 0; // buy drink_1

        #100 drink_2 = 1;
        #100 drink_2 = 0; // by drink_2

        #2000;

        coin_type = 1; // change the coin_type to "10 yuan"

        #100 insert_coin = 1;
        #100 insert_coin = 0; // push insert_coin button

        #100 drink_1 = 1;
        #100 drink_1 = 0; // buy drink_1

        #100 drink_2 = 1;
```

```
        #100 drink_2 = 0; // by drink_2
    end
    always #2 clk=~clk;
    wire power_led;
    wire error_led;
    wire take_drink_led;
    wire take_money_led;
    wire in_use_led;
    wire drink_2_green;
    wire drink_2_red;
    wire drink_1_green;
    wire drink_1_red;
    wire [7:0]AN;
    wire [7:0]SEG;
    top_module t_m(
        power,
        coin_type,
        insert_coin,
        drink_1,
        drink_2,
        cancel,
        clk,
        power_led,
        error_led,
        take_drink_led,
        take_money_led,
        in_use_led,
        drink_2_green, // drink 2
        drink_2_red, // drink 2
        drink_1_green, // drink 1
        drink_1_red,  // drink 1
        AN,
        SEG
    );
endmodule
```

主模块仿真图如图 2-5 所示。

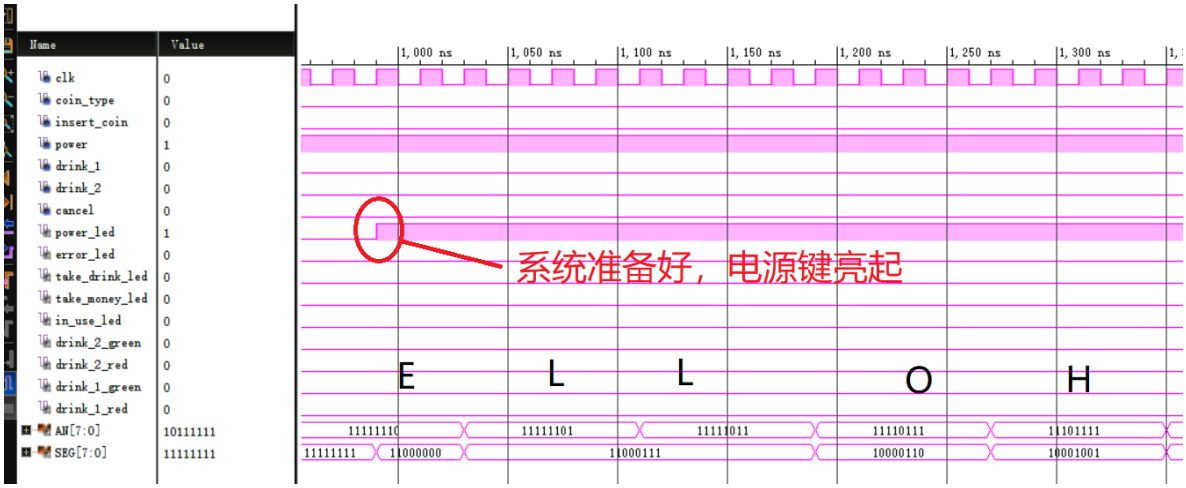

图 2-4（a）自动售货机控制器主模块仿真测试图

图 2-4（b）自动销售机控制器主模块仿真测试图

图 2-4（c） 自动销售机控制器主模块仿真测试图

仿真结果说明：

A. 电源键可以正常工作；HELLO 正常显示

B. 投币正常；购买饮料时的错误判断正常；电源等显示正常；能否购买某饮料显示正常

C. 取饮料灯可以显示；能否购买某饮料显示正常

（2）自动销售机控制器底层模块仿真

目的：验证 FSM 状态机的状态流转是否正常；验证状态机是否正确计算金额；验证电源键、退币键的功能。

输入：pwr（电源按钮），clk（主同步时钟），second（秒时钟），coin_type（币值类型），insert_coin（投币脉冲），drink_1、drink_2（购买饮料一、二按钮），cancel（取消按钮）

输出：enable（全局使能），take_drink、take_money（取饮料、取币），in_use（在占用），error（错误），drink_1_a、drink_2_a（drink 1、2 可购买），hello（显示 HELLO），money_ten、money_one、money_float（余额显示：十位、个位、是否有 0.5），show_left（是否显示退币金额），money_ten_back、money_one_back、money_float_back（退币显示：十位、个位、是否有 0.5）

程序 2-4：自动销售机控制器底层模块仿真文件

```
`timescale 1ns / 1ps
module fsm_sim();
    reg pwr = 0;         // power btn
    reg clk = 0;         // system clk
    reg second = 0;      // second clk
    reg coin_type = 0;   // coin_type switch
    reg insert_coin = 0; // insert_coin btn
    reg drink_1 = 0;     // drink_1 btn
    reg drink_2 = 0;     // drink_2 btn
    reg cancel = 0;      // cancel btn
    wire enable;     // enable all other module: f_divider, display
    wire take_drink; // drink is ok to take
    wire take_money; // money is ok to take
    wire in_use;     // this machine is in use
    wire error;      // operation is ERROR
    wire drink_2_a;  // can buy drink 2
    wire drink_1_a;  // can buy drink 1
    wire hello;      // display in hello mode
```

```
    wire [3:0]money_ten;
    wire [3:0]money_one;
    wire money_float;
    wire show_left;
    wire [3:0]money_ten_back;
    wire [3:0]money_one_back;
    wire money_float_back;

    initial begin
        #100 pwr = 1;
        #10 pwr = 0;      // push power button.

        #200;

        #100 insert_coin = 1;
        #10 insert_coin = 0; // push insert_coin button, 1 yuan now
        #100 insert_coin = 1;
        #10 insert_coin = 0; // push insert_coin button, 2 yuan now
        #100 insert_coin = 1;
        #10 insert_coin = 0; // push insert_coin button, 3 yuan now

        #100 drink_1 = 1;
        #10 drink_1 = 0;     // buy drink_1, 0.5 yuan now

        coin_type = 1;        // change the coin_type to "10 yuan"

        #100 insert_coin = 1;
        #10 insert_coin = 0; // 10.5 yuan now

        #100 drink_2 = 1;
        #10 drink_2 = 0;     // by drink_2, 5.5 yuan now

        #300;

        #100 cancel = 1;
        #10 cancel = 0; // push cancel button

        #300;
        #100 insert_coin = 1;
        #10 insert_coin = 0; // 10 yuan now

        #100 pwr = 1;
        #10 pwr = 0;
    end
    always #1 clk = ~clk;

    fsm fsm(
        pwr,         // power btn
        clk,         // system clk
        second,      // second clk
        coin_type,   // coin_type switch
        insert_coin, // insert_coin btn
```

```
        drink_1,     // drink_1 btn
        drink_2,     // drink_2 btn
        cancel,      // cancel btn
        enable,      // enable all other module: f_divider, display
        take_drink,  // drink is ok to take
        take_money,  // money is ok to take
        in_use,      // this machine is in use
        error,       // operation is ERROR
        drink_2_a,   // can buy drink 2
        drink_1_a,   // can buy drink 1
        hello,       // display in hello mode
        money_ten,
        money_one,
        money_float,
        show_left,
        money_ten_back,
        money_one_back,
        money_float_back
    );
endmodule
```

自动销售机控制器底层模块仿真图如图 2-6 所示。

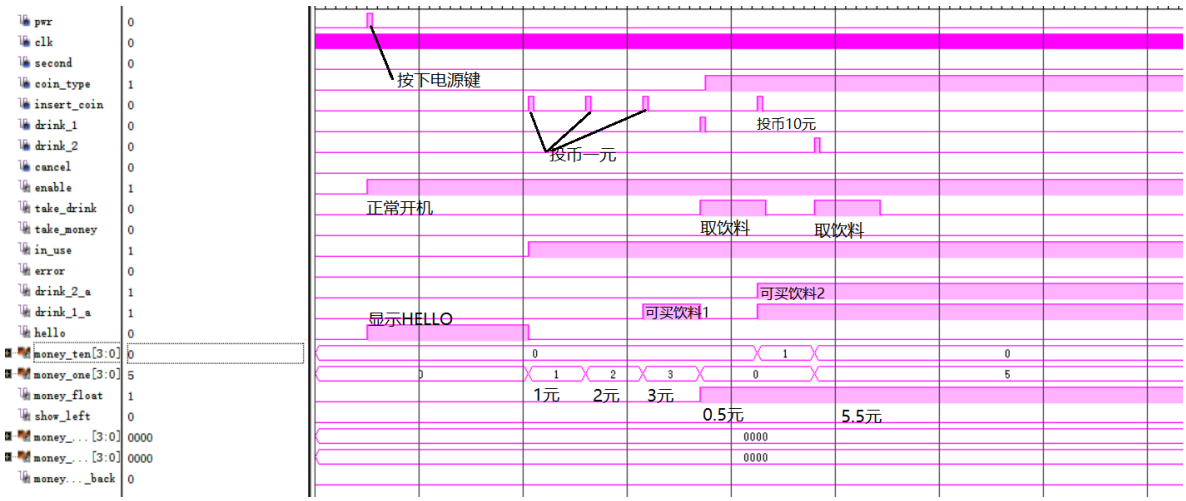

图 2-5（a）自动售货机控制器底层模块仿真图

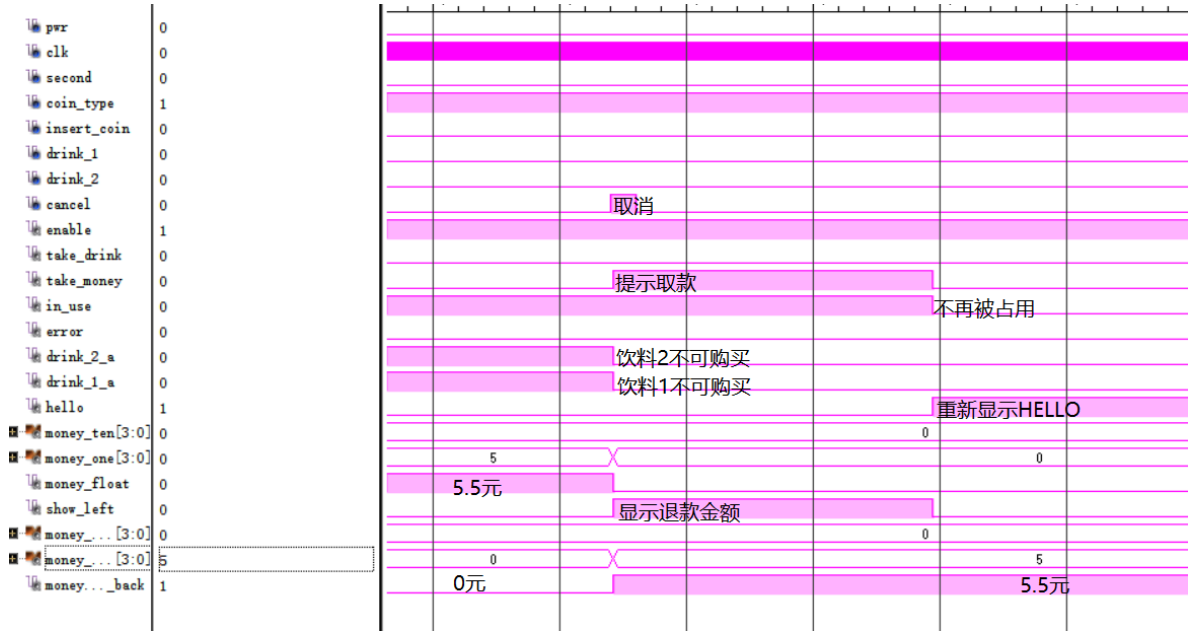

图 2-5（b）自动售货机控制器底层模块仿真图

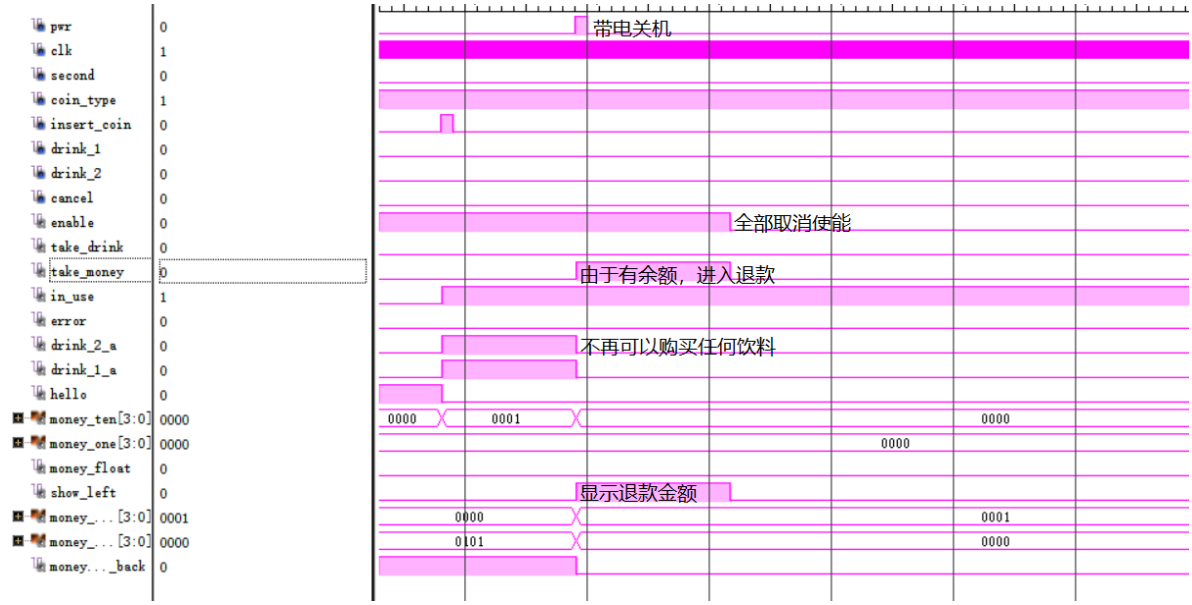

图 2-5（c）自动售货机控制器底层模块仿真图

仿真结果说明：

A. 状态机正常处理开机、投币、币值切换等输入；可以正确地输出能否取饮料、能否取币、可否买饮料；可以正确计算币值。

B．状态机可以处理取消购买的输入；可以正确的倒计时，并及时流转到下一状态。

C．状态机对于带电关机可以正常处理。

2.5 主要问题及解决方法

（1） 故障 1

```
always @(posedge clk or posedge insert_coin) begin
    time_to_show_error = time_to_show_error - 1;
    if (time_to_show_error == 0) begin
        error = 0;
```

图 2-6 故障 1 源代码

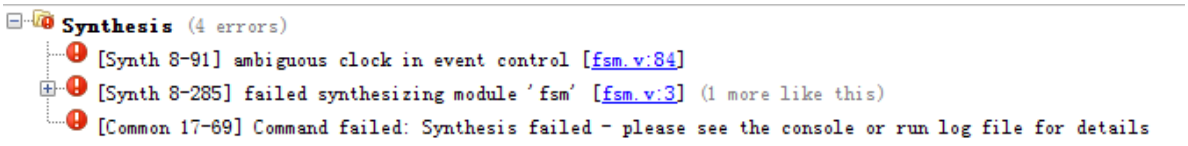

图 2-7 故障 1 提示

问题描述：由于 always@() 敏感变量出错导致不能综合

问题分析：always@()块内没有用于区分哪个变量触发的逻辑语句

解决方法：考虑到不需要异步设计，移除 posedge insert_coin 是最佳的解决方案。但是解决此问题也可以通过在 always 块内增加 if … else…来解决。

给出修改后的实例

```
always @(posedge clk) begin
    time_to_show_error = time_to_show_error - 1;
    if (time_to_show_error == 0) begin
        error = 0;
```

图 2-8 故障 1 解决代码

（2） 故障 2

```
// Main part of state machine, this FSM maintains 5 vars
always @(posedge insert_coin) begin
    case (state)
        0: begin
            hello = 1;
            enable = 1;
```

图 2-9 故障 2 源代码

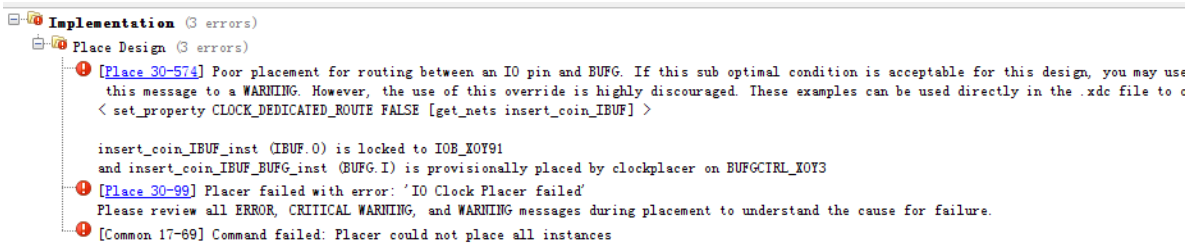

图 2-10 故障 2 错误提示

问题描述：由于缺少 xdc 约束导致问题

问题分析：由于非时钟变量在敏感列表内，会导致实现时的警告

解决方法：添加如图 2-7 所示的一条约束

给出修改后的实例

```
90
91 set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets insert_coin_IBUF];
92 set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets drink_1_IBUF];
93 set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets drink_2_IBUF];
```

图 2-11 故障 2 添加一条约束

2.6 功能测试

共进行了 4 项功能测试，它们分别为：开机和初始状态测试，投币功能测试，购买功能测试功能测试，退币、关机功能测试。

下图是设计的管脚分配图：

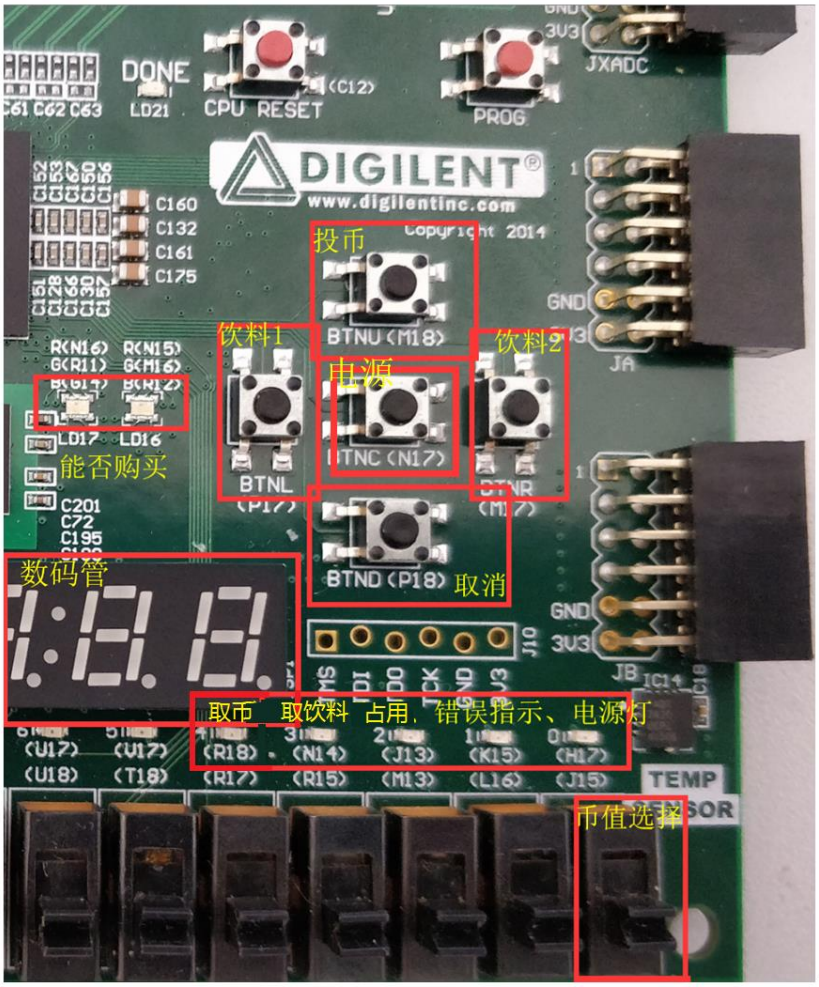

图 2-12 引脚分配

（1） 开机和初始状态测试

预期结果为开机正常，可以显示 HELLO，测试结果如图 2-13（a）。

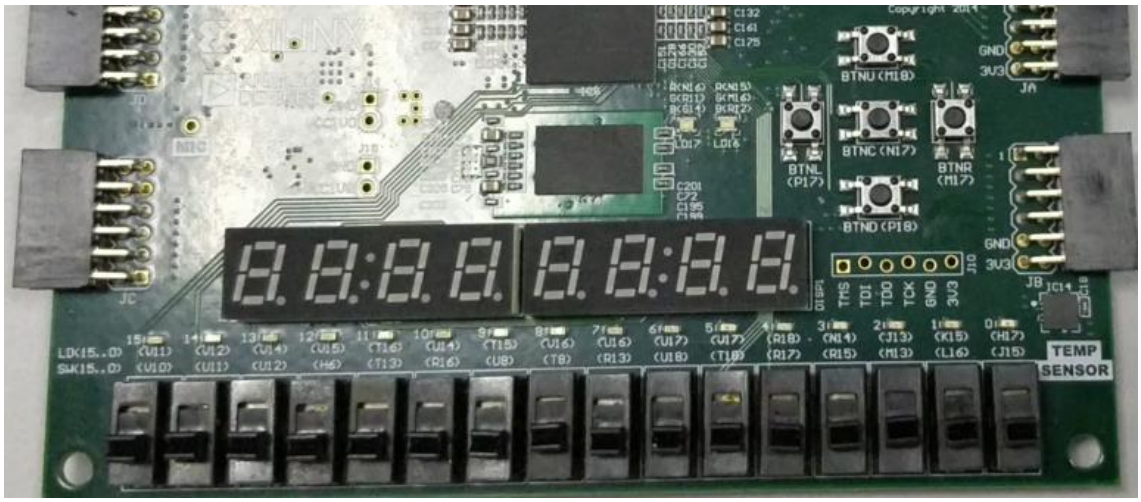

图 2-13（a） 开机和初始状态测试

当按下电源键时，可以正常显示 HELLO，同时，由于当前余额为 0，两种饮料都不能购买。如图 2-13（b）所示

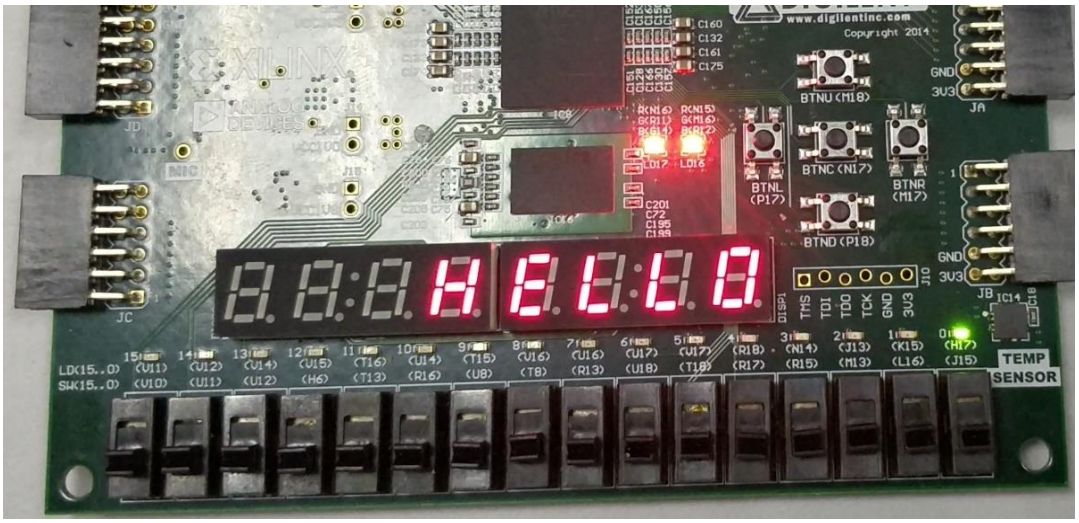

图 2-13（b） 开机和初始状态测试

（2） 投币功能测试

按下投币键若干次，币值显示正常。预期结果为每按一次增加 1，当右下角的 Switch 调为上时，每按一次加 10。当有币值时，不再显示 HELLO，而是显示币值，同时占用状态启用。测试结果如图 2-9（a）。

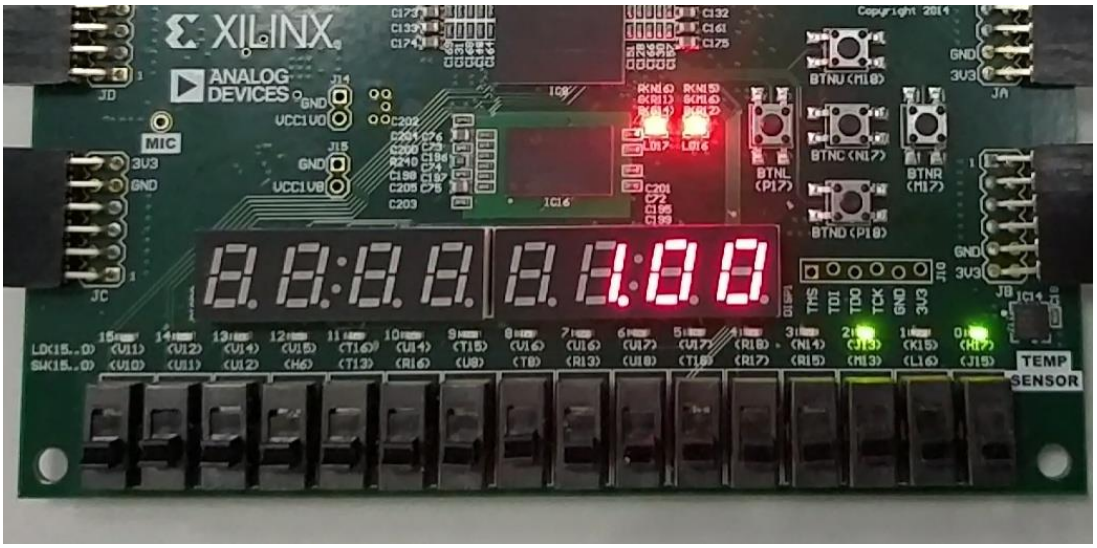

图 2-14（a） 投币功能测试

再按四次调节 Switch 至“上”，再次按投币键，预期结果为显示 15.00，所有饮料都可购买同时占用仍亮，测试结果如图 2-9（b）所示。

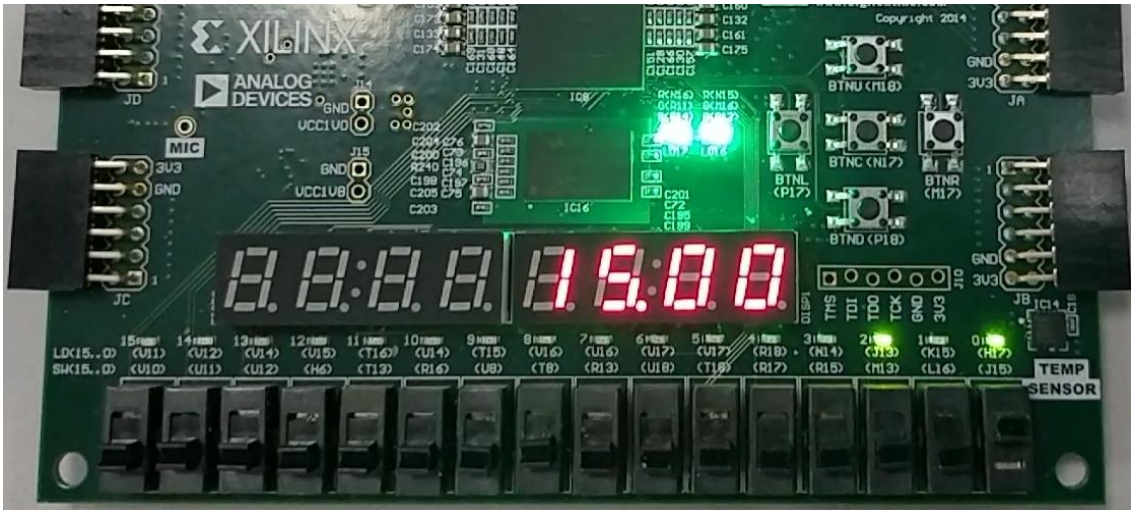

图 2-14（b） 投币功能测试

（3） 购买功能测试

当币值为 20.00 时按购买 2.5 元饮料按键，此时余额减少 2.5 元，取饮料灯亮起 5 秒。如图 2-10（a）所示。

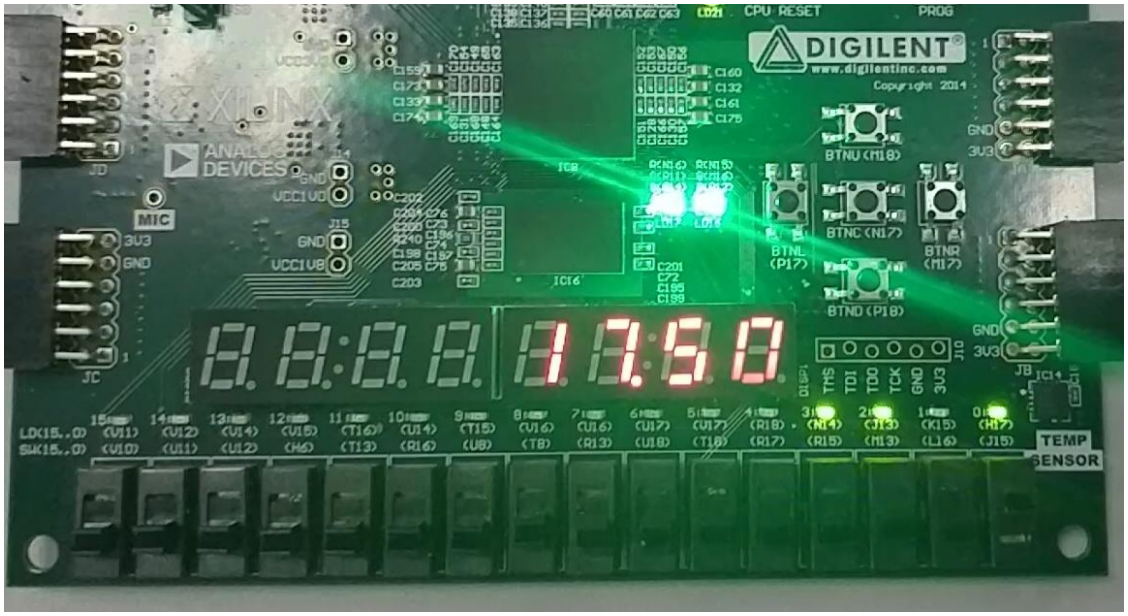

图 2-15（a） 购买功能测试

连续购买 5 元饮料至 2.5 元，再次尝试购买，会显示错误灯，同时只能能购买 2.5 元饮料，不能买 5 元饮料。如图 2-10（b）所示。（图中 2.5 未显示 0 不是 BUG，而是实际拍摄效果导致）

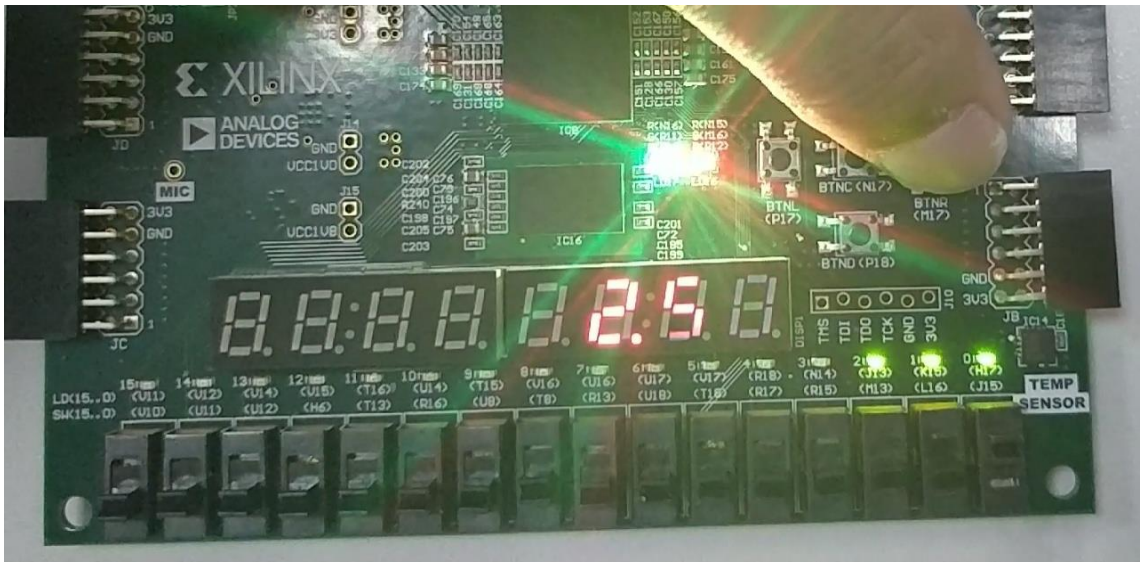

图 2-15（b） 购买功能测试

当余额恰好为 5 元时，购买 5 元饮料会使余额为 0.00。此时，取饮料灯亮起 5 秒后熄灭。等待过程中如果继续投币，则会增加余额。如果没有余额，之后进入 HELLO 状态。如下图所示。

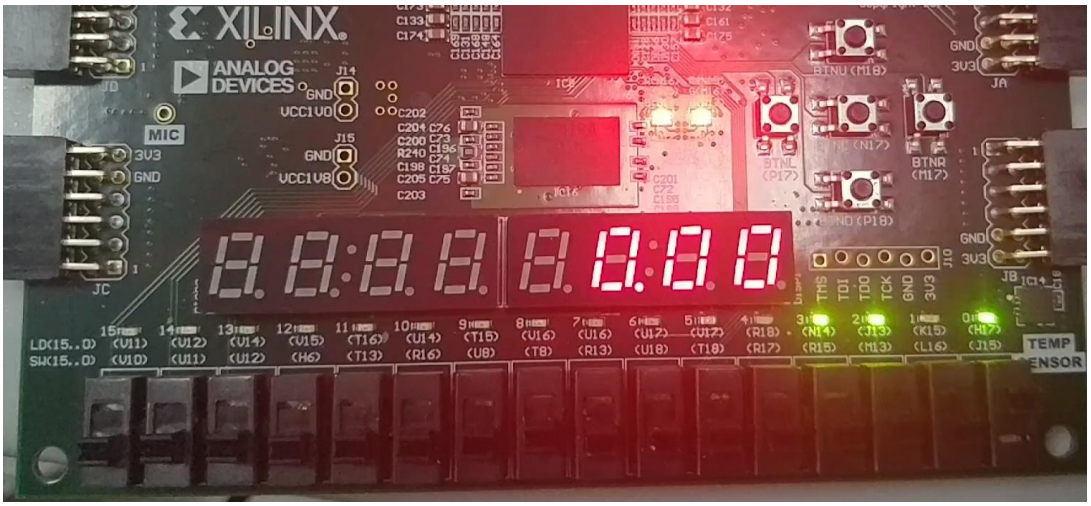

图 2-15（c） 购买功能测试

（4） 退币、关机功能测试

当余额为 2.5 元时按下取消键，进入退币流程：退币金额显示在左侧，余额显示为 0,；退币灯亮起 5 秒左右。然后系统进入待机 HELLO 状态。此时如果按下投币键，仍可投币，结果为退币之后进入等待购物状态。如图 2-11（a）所示。（图中最后一个零未显示为数码管扫描频率导致，不是 BUG）

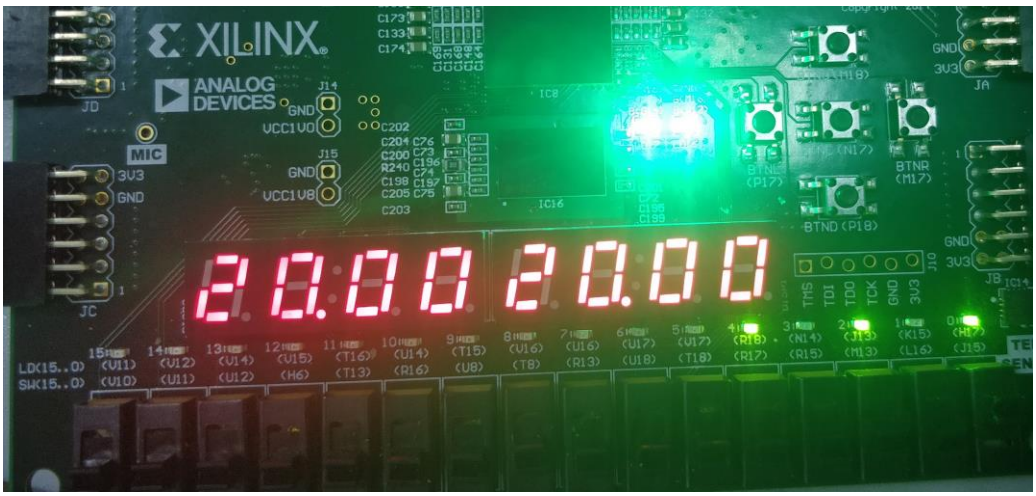

图 2-16（a） 退币、关机功能测试

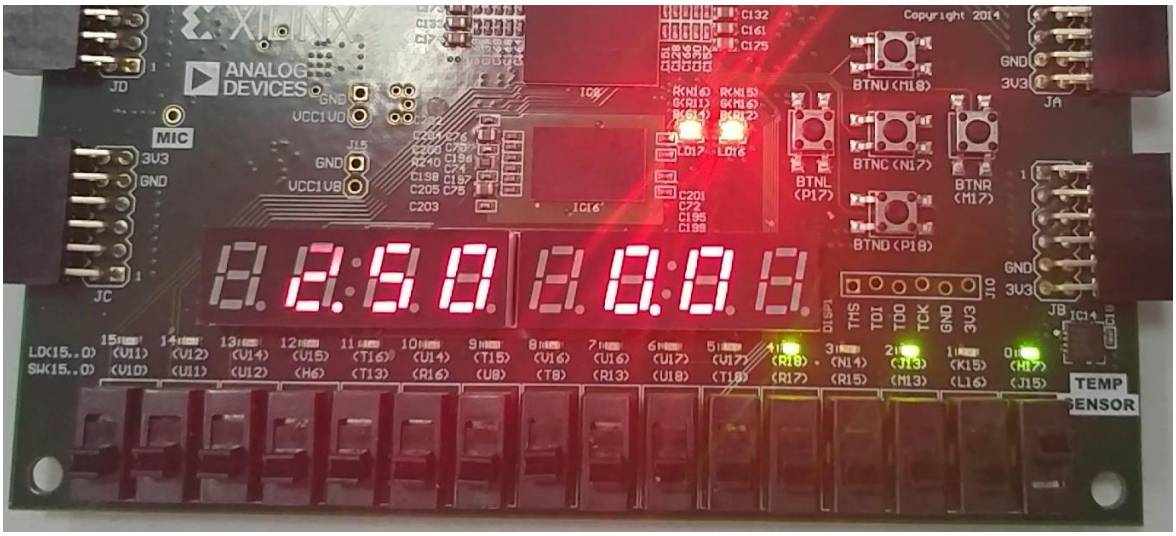

图 2-16（b） 退币、关机功能测试

当存在余额点击电源键时，会进入退币流程，此时余额为零，所有饮料不能购买。退币灯亮 5 秒左右熄灭，然后完全关机。如图 2-11（c）和 2-11（d）所示。

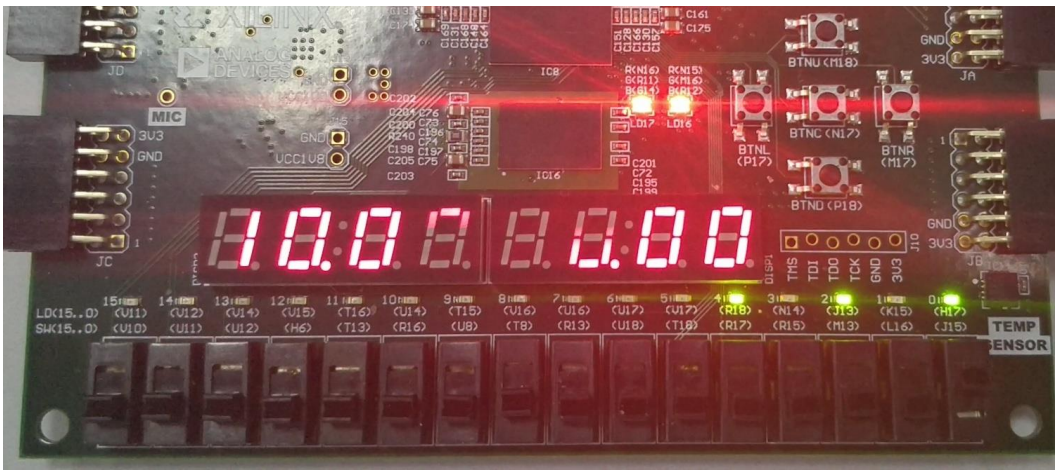

图 2-16（c） 退币、关机功能测试

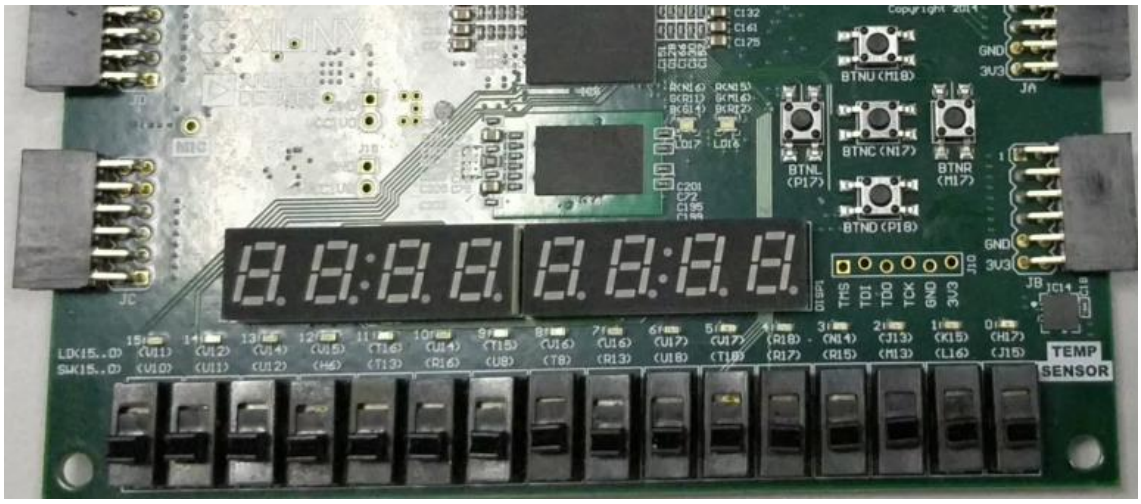

图 2-16（d） 退币、关机功能测试

3. 总结与心得

3.1 课设总结

为了做好这个实验，我做了如下准备：

（1） 复习 Verilog HDL

（2） 对课设要求认真研读，构思如何入手设计

（3） 一开始先以状态机的思路来实现，我思考了状态机的设计，在网络上参考了其他人关于状态机的理解。

（4） 在编码期间，我全身心的投入到 Verilog 中。仔细设计每一行代码。对于每一个输入输出，都进行了充足的思考

（5） 对于模块的拆分和组合，我也花费了很多心思，如何设计结构使他们通信的成本最低，我考虑了很多。

3.2 课设心得

此次课程设计使我更加熟练了 Verilog 和 Vivado，对于硬件设计也更加熟练了。

我认为我在课设中采取的先构思，在下手为我省下了很多时间和成本。这样的流程让我更难犯错。

通过对数码管显示数字和字母的设计，我对此类设计有了举一反三的能力。我明白了 RGB 灯也是通过设置每个颜色亮的时间来改变颜色的。

我也更加熟练了如何复用和解耦模块，这样，修改一个模块而不必要改动其他地方的代码，使得我的开发效率高了很多。

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]白中英，谢松云. 数字逻辑. 北京：科学出版社，2013

[3]徐光辉，程东旭，黄如. 基于 FPGA 的嵌入式开发与应用. 北京:电子工业出版社，2006.

[4]Stephen Brown. 数字逻辑基础与 Verilog 设计, 机械工业出版社, 2009

课外文献阅读 ：

[1]Analysis and Design of Digital Integrated Circuits

http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-374-analysis-and-design-of-digital-integrated-circuits-fall-2003/