

数字电路与逻辑设计

课程设计报告

报 告 人：____刘本嵩____

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 12 日

数字逻辑课程设计学生工作表

[illegible]

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立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。

华中科技大学课程设计报告

目 录

1 综合实验设计概述.....	5
1.1 实验目的.....	5
1.2 实验要求.....	5
1.3实验任务.....	5
1.4 实验环境.....	5
2 自动售货机设计方案.....	7
2.1 内容.....	7
2.2 设计思路.....	8
2.2.1 shopping模块设计.....	9
2.2.2 numshow模块设计.....	9
2.2.3 cterToAn模块设计.....	10
2.2.4 digitToPhy(WithPt)模块设计.....	10
2.3代码实现.....	11
2.4 仿真过程.....	19
2.5 主要问题及解决方法.....	24
2.6 功能测试.....	25
3 总结与心得.....	36
3.1课设总结.....	36
3.2课设心得.....	36
4 参考文献.....	37
附录1 课程设计报告的格式要求补充说明.....	38
附录2 参考文献格式补充说明.....	39
附录3 仓库地址 二进制文件地址和备份.....	40

华中科技大学课程设计报告

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

华中科技大学课程设计报告

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，扬声器放大器以及大量的 I/O 设备，让 Nexys4-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**”。

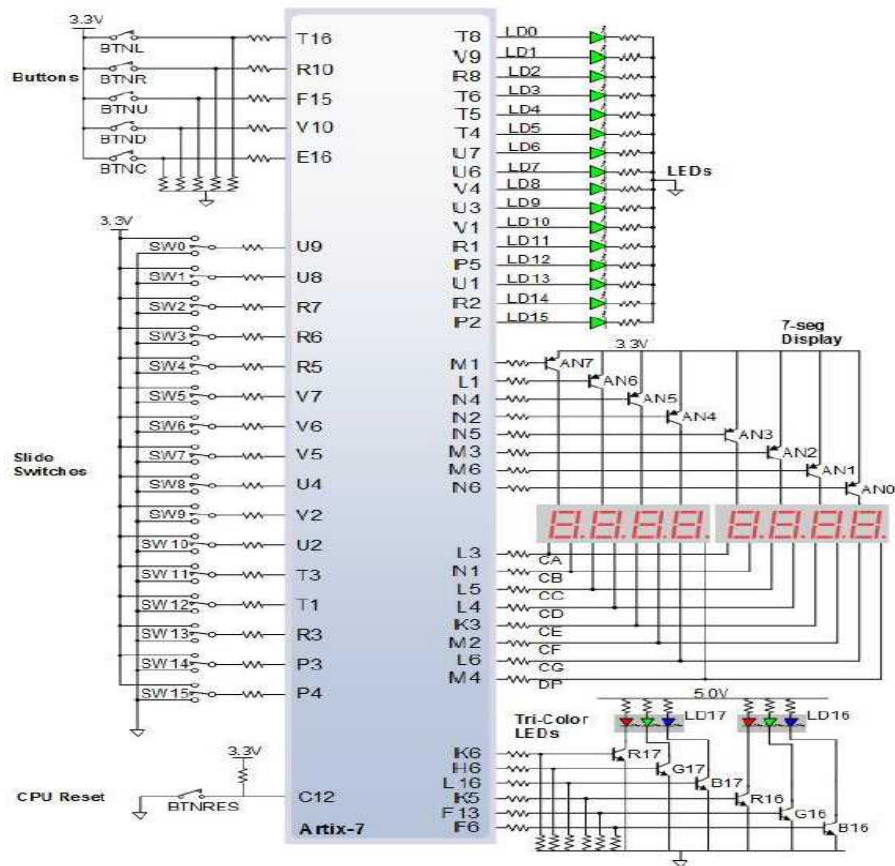


图 1-1 Nexys4 通用 I/O 设备

2 自动售货机设计方案

2.1 内容

(1) 基本内容

(1) 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”；

(2) 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；

(3) 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

(4) “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”。

饮料自动投币售卖机控制器系统框图参见图 2-1 所示。

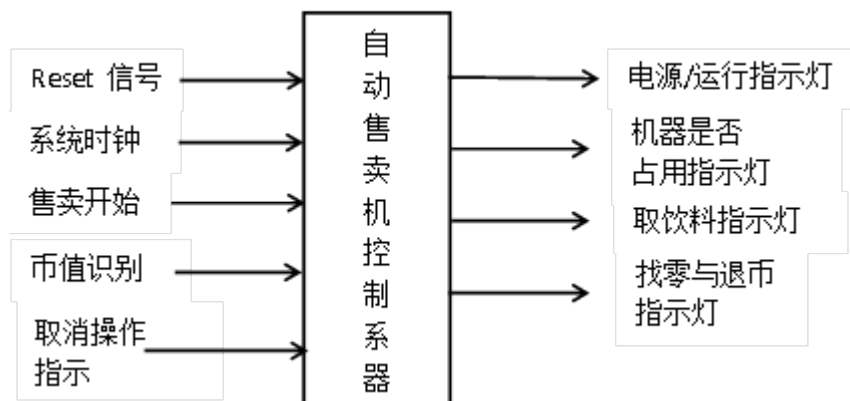


图 2-1 自动投币售卖机控制器框图

- (a) 系统时钟 (**clk**) 作为系统的同步信号；
- (b) **Reset** 作为电路总清零信号；
- (c) 售卖开始信号：表示客户开始操作时的启动信号；
- (d) 币值识别信号：由人民币识别器给出具体的币值；

华中科技大学课程设计报告

- (e) 取消操作信号：表示客户取消操作；
- (f) 机器是否占用指示灯：表示该机是未被占用；
- (g) 取饮料指示灯：表示客户可以取走饮料；
- (h) 找零与退币标志指示灯：表示找零提示；
- (i) 找零与退币币值：退给客户的币值。

2.2 设计思路

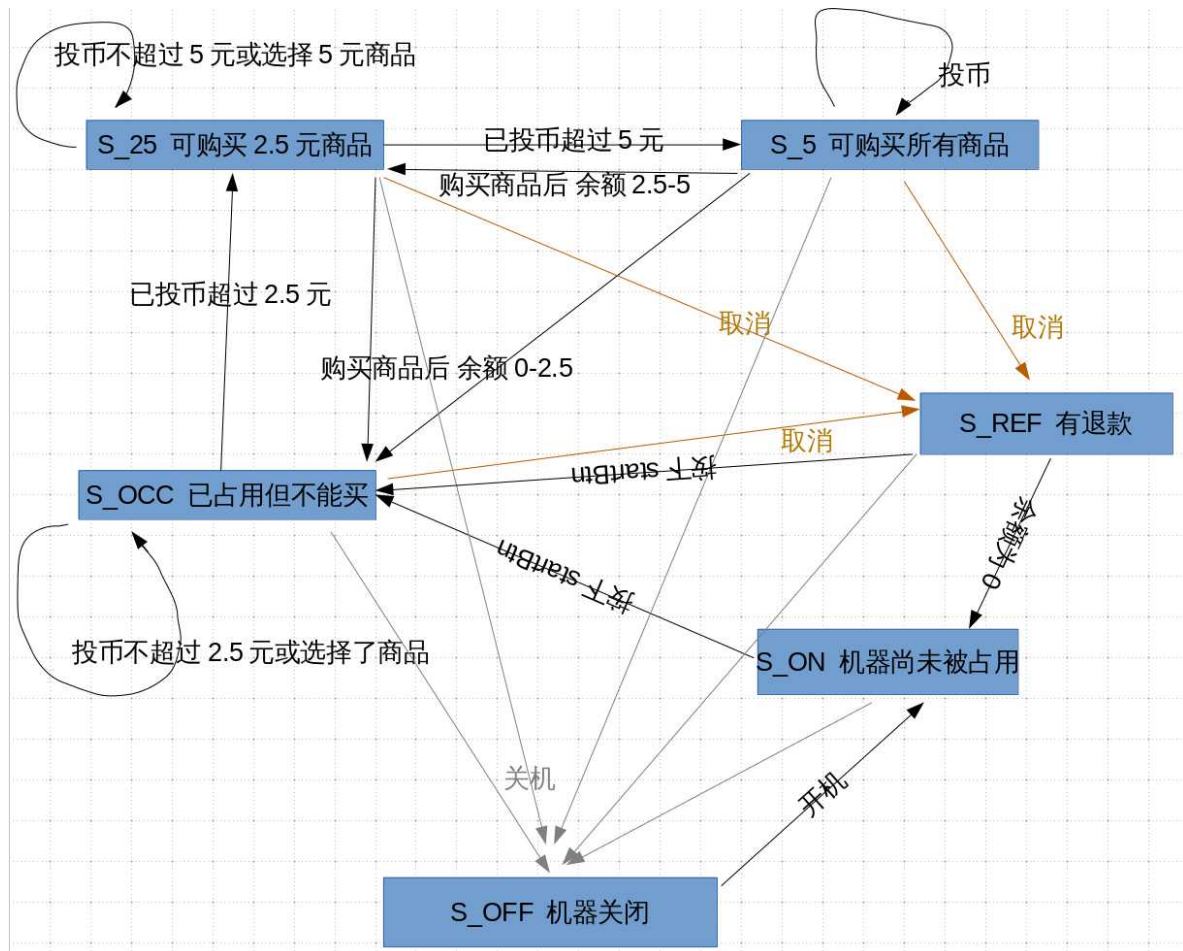
本设计要求顶层采用原理图设计，各底层均采用 SystemVerilog 设计，其模块图参见图 2-2 所示：



图 2-2 自动售货机的模块图

华中科技大学课程设计报告

2.2.1 shopping 模块设计



2.2.2 numshow 模块设计

numshow 先使用以下语句得到即将显示的数字。由于 vivado 的 IPbug，在 _xilinx_bug_433316_workaround 中的实现稍有不同。此 bug 已向 xilinx 报告。

```
wire [31:0] digits [3:0] = num > 9999 ? '{20, 20, 20, 20} :
```

```
num == 9999 ? '{16, 17, 18, 19} :
```

```
'{num/1000%10, num/100%10, num/10%10, num/1%10};
```

然后将 digits 数组中的二进制数字由 digitToPhy(WithPt)转换为适合数码管的数字表示。代码如图。

```
cterToAn_2(cter, an);
```

华中科技大学课程设计报告

```
digitToPhy _3(digits[3], digits_7s[2]);
digitToPhyWithPt _4(digits[2], digits_7s[1]);
digitToPhy _5(digits[1], digits_7s[0]);
digitToPhy _6(digits[0], digits_7s[3]);
```

然后其使用如下所示的时钟循环。其中分频器被 inline 在代码中。

```
always @(posedge clk) begin
    cterShowNextDigit <= cterShowNextDigit + 1;
    if(cterShowNextDigit == 32000) fork
        cter <= cter + 1;
        phyPad <= digits_7s[cter];
        cterShowNextDigit <= 0;
    join
end
```

经过测试，其工作正常。

2.2.3 cterToAn 模块设计

其使用纯组合电路将 0-8 的 cter 转换为 AN 信号。

2.2.4 digitToPhy(WithPt)模块设计

其使用纯组合电路将二进制数字转换为对应的数码管信号。其中 hello-都有对应的特殊值。

```
// 0 1 2 3 4 5 6 7 8 9 a b c d e f h e l o -
// 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 Error
assign phy =
    (digit == 0) ? 8'b11000000 :
    (digit == 1) ? 8'b11111001 :
    ...
```

特别的，digitToPhyWithPt 会加上小数点。其实现很简单。

```
assign phy[7] = 0;
wire [7:0] joined;
assign phy[6:0] = joined[6:0];
```

华中科技大学课程设计报告

digitToPhy_i1(digit, joined);

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。

自动售货机顶层原理图，参见图 2-2 所示。（RTL Analysis 下“Schematic”截图）

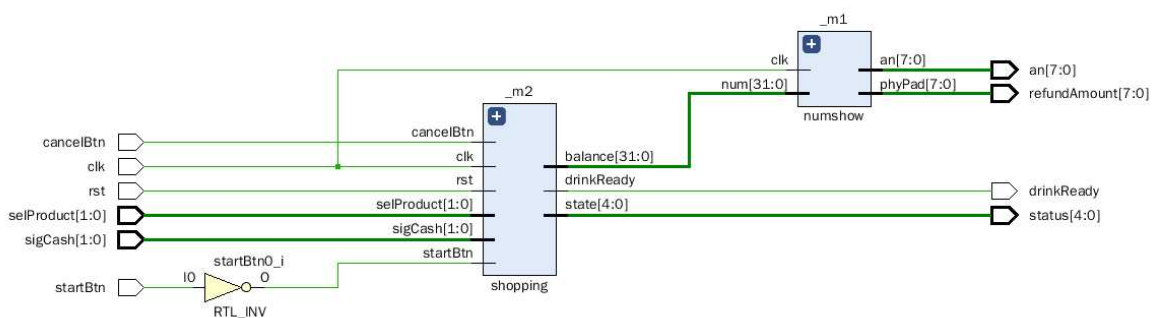


图 2-2 自动售货机顶层原理图

(1) **main** 顶层模块（例如：**main** 用于整合整个系统）

程序 2-1：**main** 顶层模块（例如：**main** 用于整合整个售卖机系统）**Verilog** 代码

```
`include "global.svh"

module main (

    input rst,

    input clk,

    input startBtn,

    input money_t sigCash,

    input item_t selProduct,

    input cancelBtn,

    output state_t status,

    output wire drinkReady,

    output wire [7:0] refundAmount,

    output wire [7:0] an
```

华中科技大学课程设计报告

```
);  
  
    wire [31:0] balance;  
  
    numshow _m1(clk, balance, refundAmount, an);  
  
    // startBtn is CPURESET btn so it must be reversed  
  
    shopping _m2(rst, clk, ~startBtn, sigCash, selProduct, cancelBtn, status, drinkReady, balance);  
  
    // Fucking vivado simulator forbits to cast any other type to sv enum.  
  
    // I have to use its register type everywhere!!!!  
  
endmodule
```

(2) shopping 底层模块 Verilog 代码

(本模块给出底层模块中最主要的一个模块即可，比如状态机模块等)

程序 2-2 : shopping 底层模块 Verilog 代码

```
`include "global.svh"  
  
module shopping (  
    input logic rst,  
    input logic clk,  
    input logic startBtn,  
    input money_t sigCash,  
    input item_t selProduct,  
    input logic cancelBtn,  
    output state_t state,  
    output logic drinkReady,  
    // state is {running, occupied, drinkReady, refundReady}  
    output int balance  
    // connected to number output. Unit: cent  
);
```

华中科技大学课程设计报告

```
int cash; // tmpVar
bit moneyEnough; // tmpVar
money_t sigCashOld = CNY_NULL;
item_t selProductOld = ITEM_NULL;

bit [7:0] clk_fucker = 0;
always @(posedge clk) begin
`ifndef XILINX_ISIM
// disable divider on simulation
    clk_fucker <= clk_fucker + 1;
`endif
    if(clk_fucker == 0) begin
        if(rst) begin
            state <= STATE_OFF;
            balance <= 99999; // shows ----
        end else
            case(state) inside
                STATE_OFF: fork
                    state <= STATE_ON;
                    balance <= 9999; // shows helo
                    drinkReady <= 0;
                join
                STATE_ON, STATE_REFUND:
                    if(startBtn) fork
                        state <= STATE_OCCUPIED;
                        balance <= 0;
                        drinkReady <= 0;
```

华中科技大学课程设计报告

```
join
STATE_OCCUPIED:
if(cancelBtn) fork
    state <= STATE_TEMP; // tip: no latency
join else begin
    // Not canceled
    if(balance > 499) fork
        state <= STATE_MONEY_GE_5;
    join else if(balance > 249) fork
        state <= STATE_MONEY_GE_3;
    join else if(sigCash != CNY_NULL && sigCashOld == CNY_NULL) fork
        // Inserting cash
        cashToPrice(sigCash, cash);
        balance <= balance + cash;
    join
end
STATE_MONEY_GE_3:
if(cancelBtn) fork
    state <= STATE_TEMP; // tip: no latency
join else begin
    // Not canceled
    if(balance < 249) fork
        state <= STATE_OCCUPIED;
    join else if (balance > 499) fork
        state <= STATE_MONEY_GE_5;
    join else if(sigCash != CNY_NULL && sigCashOld == CNY_NULL) fork
        // Inserting cash
        cashToPrice(sigCash, cash);
```

华中科技大学课程设计报告

```
        balance <= balance + cash;

    join else if(selProduct != ITEM_NULL && selProductOld == ITEM_NULL) fork

        // purchasing

        fuck_balance(selProduct, balance, drinkReady);

    join

end

STATE_MONEY_GE_5:

if(cancelBtn) fork

    state <= STATE_TEMP; // tip: no latency

join else begin

    // Not canceled. Deal with QiongBi first.

    if(balance < 249) fork

        state <= STATE_OCCUPIED;

    join else if(balance < 499) fork

        state <= STATE_MONEY_GE_3;

    join else if(sigCash != CNY_NULL && sigCashOld == CNY_NULL) fork

        // Inserting cash

        cashToPrice(sigCash, cash);

        balance <= balance + cash;

    join else if(selProduct != ITEM_NULL && selProductOld == ITEM_NULL) fork

        // purchasing

        fuck_balance(selProduct, balance, drinkReady);

    join

end

STATE_TEMP:

if(balance != 0)

    state <= STATE_REFUND;

else
```

华中科技大学课程设计报告

```
        state <= STATE_ON;

    default: begin end

endcase

sigCashOld <= sigCash;

selProductOld <= selProduct;

end

end

task automatic cashToPrice(input money_t sig, output int cash);

    case(sig)

        CNY_NULL:

            cash = 0;

        CNY_1:

            cash = 100;

        CNY_10:

            cash = 1000;

        default:

            cash = 9999;

    endcase

endtask //automatic

task automatic fuck_balance(input item_t purchased, ref int balance, ref logic ok);

    int price;

    itemToPrice(purchased, price);

    if(balance < price)

        ok = 0;

    else begin
```

华中科技大学课程设计报告

```
        ok = 1;

        balance = balance - price;

    end

endtask //automatic
```

```
task automatic itemToPrice(input item_t item, output int price);
```

```
    // if the price is 12.34, it must return price=1234
```

```
    case(item)
```

```
        ITEM_PRICED_2p5:
```

```
            price = 250;
```

```
        ITEM_PRICED_5:
```

```
            price = 500;
```

```
        ITEM_NULL:
```

```
            price = 0;
```

```
        default:
```

```
            price = 7777;
```

```
    endcase
```

```
endtask //automatic
```

```
/* Fucking vivado failed to simulate these code
```

```
function cashToPrice(input money_t sig);
```

```
    case(sig)
```

```
        CNY_NULL:
```

```
            return 0;
```

```
        CNY_1:
```

```
            return 100;
```

```
        CNY_10:
```

```
            return 1000;
```

```
        default:
```

华中科技大学课程设计报告

```
        return 9999;

    endcase

endfunction

task automatic fuck_balance(input item_t purchased, ref int balance, ref bit ok);

    int price;

    price = itemToPrice(purchased);

    if(balance < price)

        ok = 0;

    else begin

        ok = 1;

        balance = balance - price;

    end

endtask //automatic

function itemToPrice(input item_t item);

    // if the price is 12.34, it must return price=1234

    case(item)

        ITEM_PRICED_2p5:

            return 250;

        ITEM_PRICED_5:

            return 500;

        ITEM_NULL:

            return 0;

        default:

            return 7777;

    endcase

endfunction
```

华中科技大学课程设计报告

```
*/  
  
endmodule
```

2.4 仿真过程

为了验证设计的正确性，对 shopping、numshow 等模块进行了仿真，具体过程如下：

测试者注意：仿真时必须全局定义 **verilog** 宏 **XILINX_ISIM**。否则作为警告，时钟将不会工作。如果强行令时钟工作，那么模块内部不会正常工作。

(1) 顶层模块仿真

目的：验证系统的基本功能，包括启动机器，插入钞票，购买商品，退币，再次启动机器等操作

输入：共 6 个，分别为：rst,clk,startBtn,sigCash,selProduct,cancelBtn；

输出：共 4 个，分别为：,status,drinkReady,refundAmount,an

程序 2-3：顶层模块仿真文件

测试者注意：仿真时必须全局定义 **verilog** 宏 **XILINX_ISIM**。否则作为警告，时钟将不会工作。如果强行令时钟工作，那么模块内部不会正常工作。

```
`include "global.svh"  
  
module main_tb (  
  
  
);  
  
    logic rst=0;  
  
    logic clk=0;  
  
    logic startBtn=0;  
  
    money_t sigCash=CNY_NULL;  
  
    item_t selProduct=ITEM_NULL;  
  
    logic cancelBtn=0;  
  
    logic drinkReady=0;  
  
    state_t status=STATE_OFF;  
  
    bit [7:0] refundAmount=0;
```

华中科技大学课程设计报告

```
bit [7:0] an=0;

`SIMU_SET_CLK(clk);

initial begin

    `SIMU_SET_BTN_ON(startBtn);

    `SIMU_PUSH_MONEY(CNY_1);

    `SIMU_PUSH_MONEY(CNY_1);

    `SIMU_PUSH_ITEM(ITEM_PRICED_2p5);

    `SIMU_PUSH_MONEY(CNY_1);

    `SIMU_PUSH_MONEY(CNY_1);

    `SIMU_PUSH_MONEY(CNY_10);

    `SIMU_PUSH_MONEY(CNY_1);

    `SIMU_PUSH_ITEM(ITEM_PRICED_5);

    `SIMU_PUSH_ITEM(ITEM_PRICED_2p5);

    `SIMU_PUSH_MONEY(CNY_1);

    `SIMU_PUSH_MONEY(CNY_1);

    `SIMU_SET_BTN_ON(cancelBtn);

    `SIMU_PUSH_MONEY(CNY_1);

    `SIMU_SET_BTN_ON(startBtn);

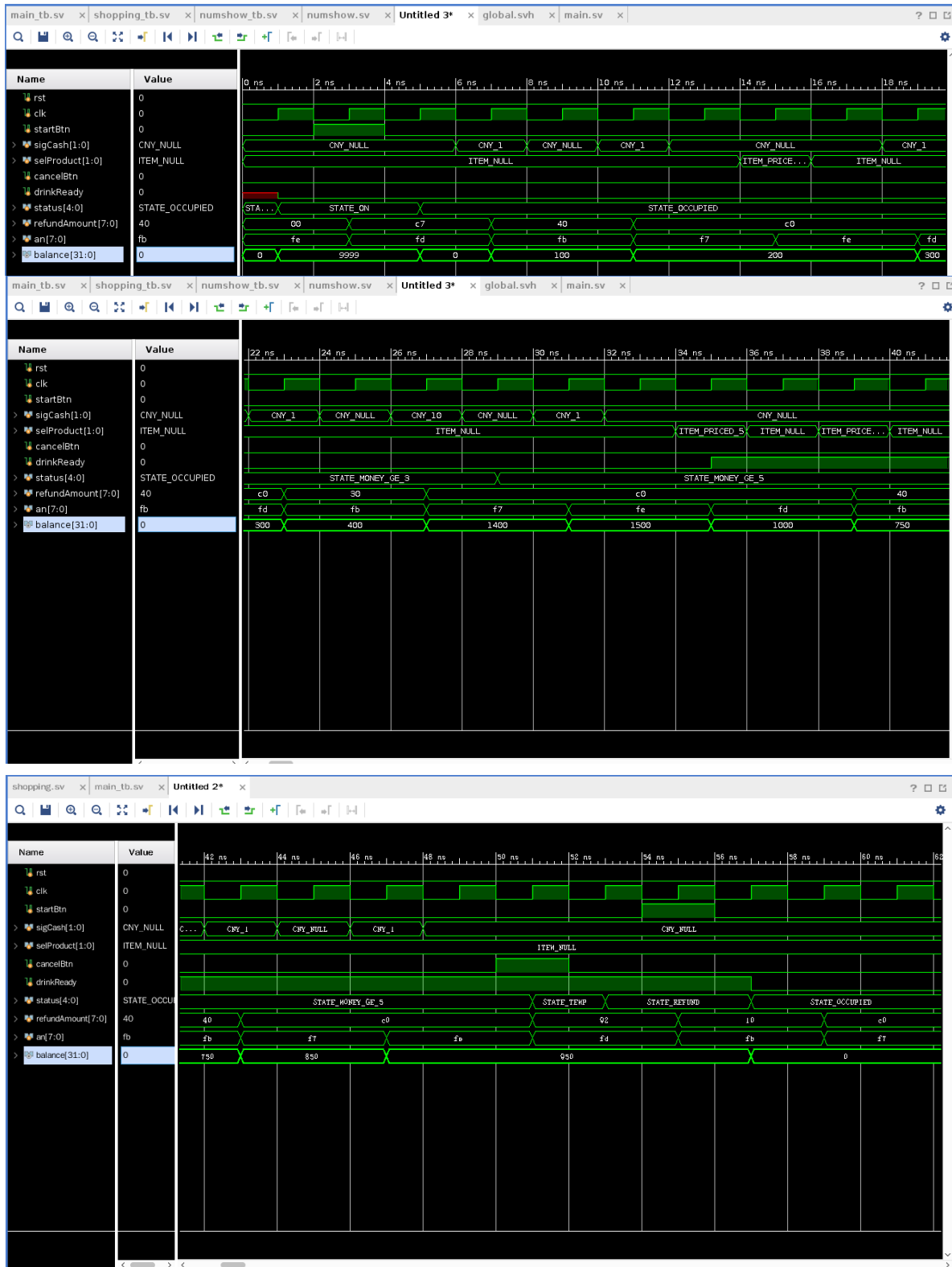
end

main _7(rst,clk,startBtn,sigCash,selProduct,cancelBtn,status,drinkReady,refundAmount,an);

endmodule
```

主模块仿真图如图 101 所示。

华中科技大学课程设计报告



华中科技大学课程设计报告

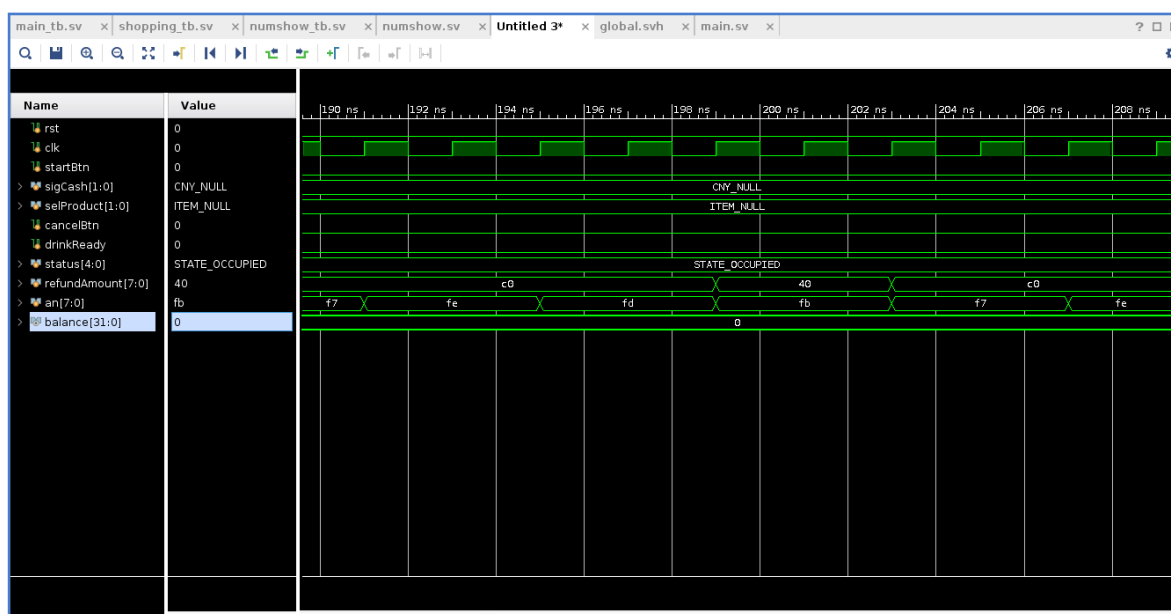


图 101 (a) main 主模块仿真测试图(为便于查看 分段截图)

仿真结果说明：

注意，由于测试时操作极其紧凑，numshow 模块会来不及刷新所有数码管。在下面的单模块测试会验证此模块的正确性。因此余额请以 balance 为准，这本质上是对 shopping 模块的仿真。

status 的某些位被直接作为对应灯的状态，因此在仿真中见不到这些灯对应的变量。请查阅 global.svh 中声明的 state_t - value 映射和 state_t 的位 - LED 灯的映射。

- 1ns: balance 为 9999 时 numshow 会显示 hello。
- 5ns: startBtn 生效。hello 变为当前投币数 0.00 元，状态变为 OCCUPIED。
- 7ns 9ns: 投了两张 1 元人民币。余额对应发生变化。
- 11ns: 尝试购买了一个 2.5 元商品。金钱不足被拒绝。
- 19 23 27 31ns: 投入 2 张 1 元，1 张 10 元,1 张 1 元。余额正确。可以注意到状态在随着余额一起变化。
- 35ns 买了个 5 元的饮料。drinkReady 为 1 说明饮料出来了。余额正确。
- 39 43 47ns 又买了个 2.5 元饮料并继续投了两块钱。余额正确，状态正确。
- 51ns 按下取消。经过一个临时状态(持续两个时钟周期)后，状态变为 REFUND，并且显示了退出的钱数。行为正确。拿走退出的钱，人即离开。
- 55ns 下一个人按下 startBtn 启动机器。启动后的状态正常。测试流程结束。

(2) numshow 底层模块仿真

华中科技大学课程设计报告

由于 numshow 模块的仿真是不可缺少的，所以没有机会对 shopping 模块进行仿真，它和 main 几乎完全相同。请谅解。

目的：验证 numshow 的基本功能

输入：clk,num

输出：phyPad,an

程序 2-4：numshow 底层模块仿真文件

```
`include "global.svh"

module numshow_tb (

);

    bit clk = 0;

    int num = 5301;

    bit [7:0] phyPad;

    bit [7:0] an;

    `SIMU_SET_CLK(clk);

    numshow _1(clk, num, phyPad, an);

endmodule
```

numshow 底层模块仿真图如图 2-3 所示。

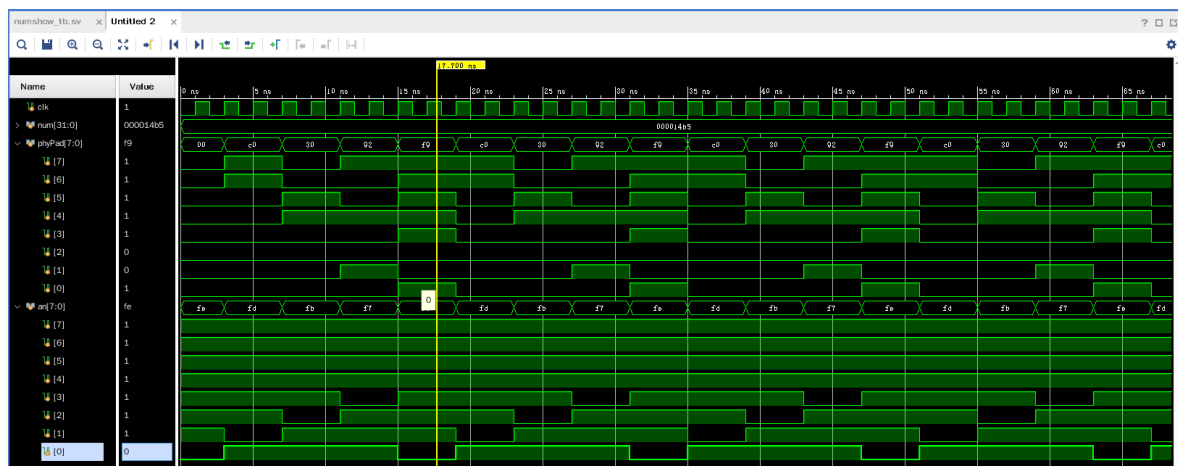


图 2-3 numshow 底层模块仿真测试图

仿真结果说明：被显示的数字是 53.01

A. 17ns AN 表示只显示第 0 位，phyPad 对应最低位的值 1

华中科技大学课程设计报告

- B. 17ns AN 表示只显示第 1 位，phyPad 对应第 1 位的值 0
- C. 17ns AN 表示只显示第 2 位，phyPad 对应第 2 位的值 3，有小数点。
- D. 17ns AN 表示只显示第 3 位，phyPad 对应第 3 位的值 5

2.5 主要问题及解决方法

(1) 故障 1 (给出错误实例)

问题描述：vivado IP Segmentation Fault

ERROR: [XSIM 43-3316] Signal SIGSEGV received.

Printing stacktrace...

```
[0] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x4de468]
[1] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x4ccd6b]
[2] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x4cadbe]
[3] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x4cca06]
[4] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x554c56]
[5] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x65ac63]
[6] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x660e93]
[7] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x4930da]
[8] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x4850ab]
[9] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x7d2775]
[10] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x7d1bd5]
[11] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x7d236f]
[12] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x7d2775]
[13] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x7d1bd5]
[14] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x7d236f]
[15] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x7d2775]
[16] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x7a6832]
[17] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x46bb94]
[18] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x47e165]
[19] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x44fb9f]
```


华中科技大学课程设计报告

[20] /usr/lib/libc.so.6(__libc_start_main+0xf3) [0x7f68c8d9c223]

[21] /home/recolic/extraDisk/xilinx/Vivado/2018.1/bin/unwrapped/lnx64.o/xelab() [0x46a101]

问题复现：

```
assign digits = num > 9999 ? '{20, 20, 20, 20} :
```

```
num== 9999 ? '{16, 17, 18, 19} :
```

```
'{num/1000%10, num/100%10, num/10%10, num/1%10};
```

问题分析：xilinx vivado bug。只能绕过

解决方法：使用更暴力的赋值方式。

```
assign digits = '{
```

```
num>9999?20: num==9999?16: num/1000%10,
```

```
num>9999?20: num==9999?17: num/100%10,
```

```
num>9999?20: num==9999?18: num/10%10,
```

```
num>9999?20: num==9999?19: num/1%10
```

```
};
```

(2) 故障 2 (给出错误实例)

问题描述：未知的内存竞争

问题分析：推测为电路过长，门电路延迟的积累超过一个时钟周期导致。但是检查设计的电路图却没有支持这种推测。

解决方法：硬件运行时时钟周期减慢 256 倍。仿真时自动使用宏进行屏蔽即可。

```
always @(posedge clk) begin
```

```
`ifndef XILINX_ISIM
```

```
// disable divider on simulation
```

```
clk_fucker <= clk_fucker + 1;
```

```
`endif
```

```
if(clk_fucker == 0) begin
```

```
。 。 。
```

2.6 功能测试

共进行了 2 项功能测试，它们分别为：投币和购买人工功能测试，开启与退款人工功能测试。

(0) 开发板输入输出安排

华中科技大学课程设计报告

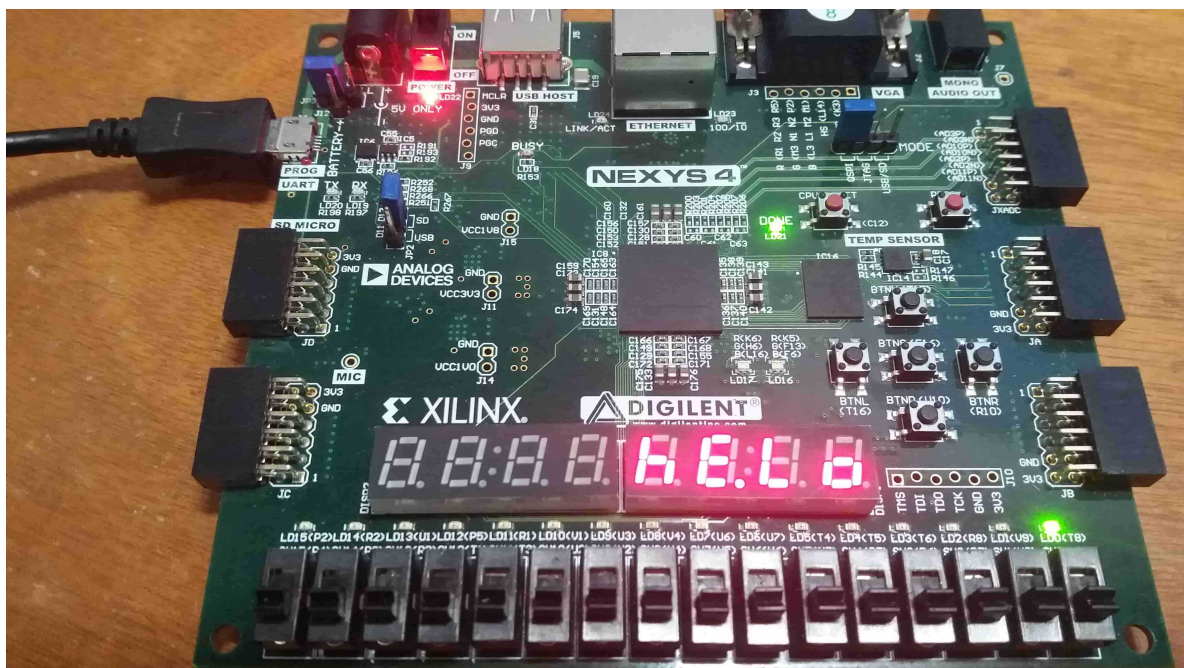
指示灯: 左边 [2.5 可购买] [5 可购买] 中间 [退币] [有饮料] [占用] [开机] 右边

开关：最右边为 rst

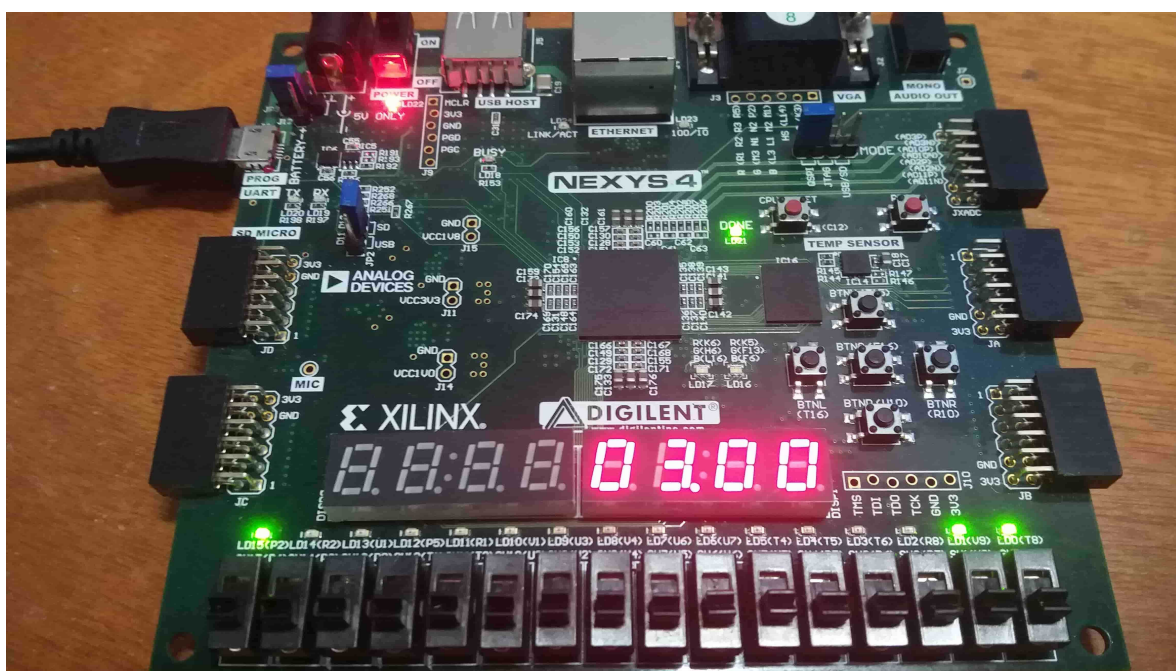
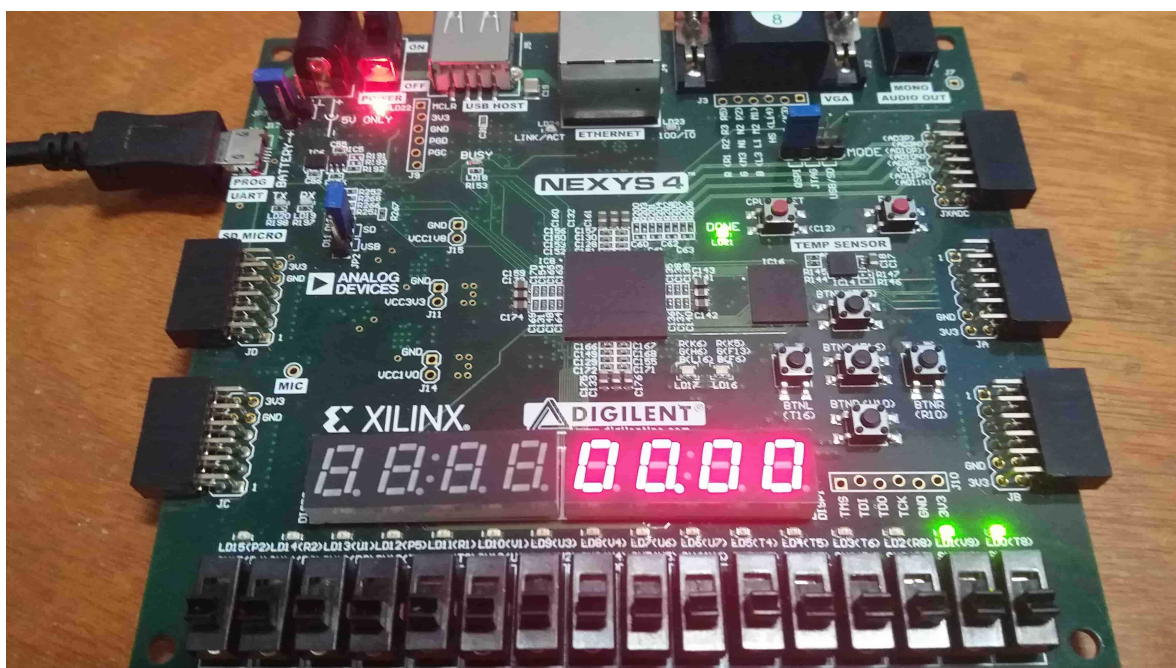
按钮：CPU RESET 为 startBtn，左[投 1 元] 右[投 10 元] 上[买 5] 下[买 2.5] 中间[cancelBtn]

(1) 投币和购买人工功能测试

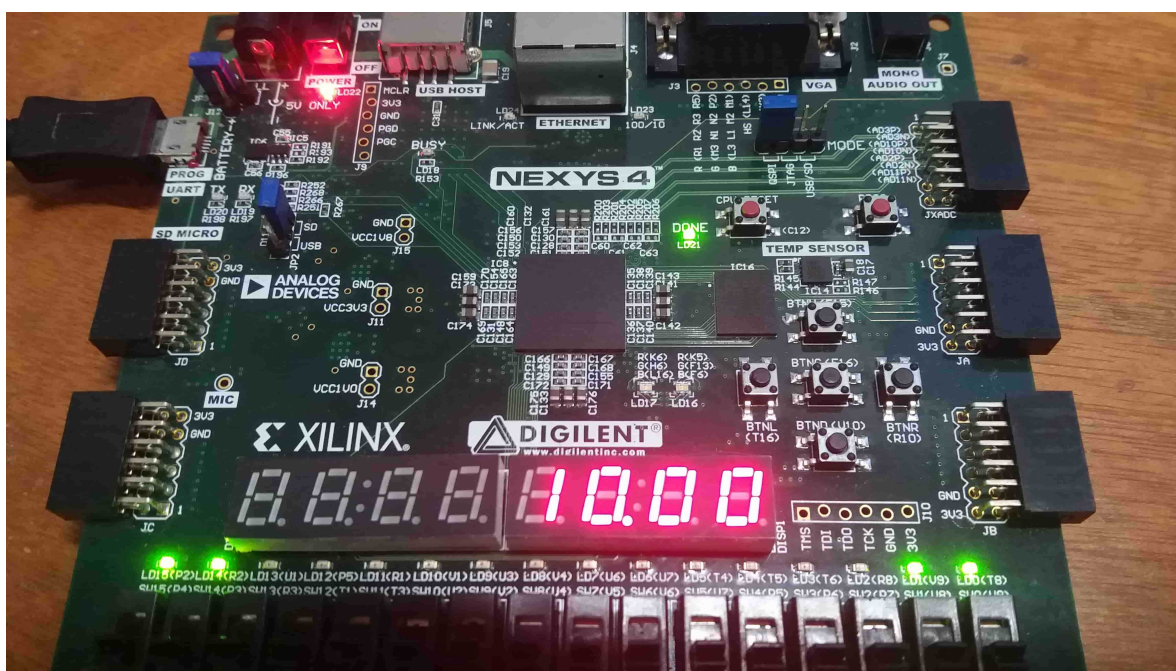
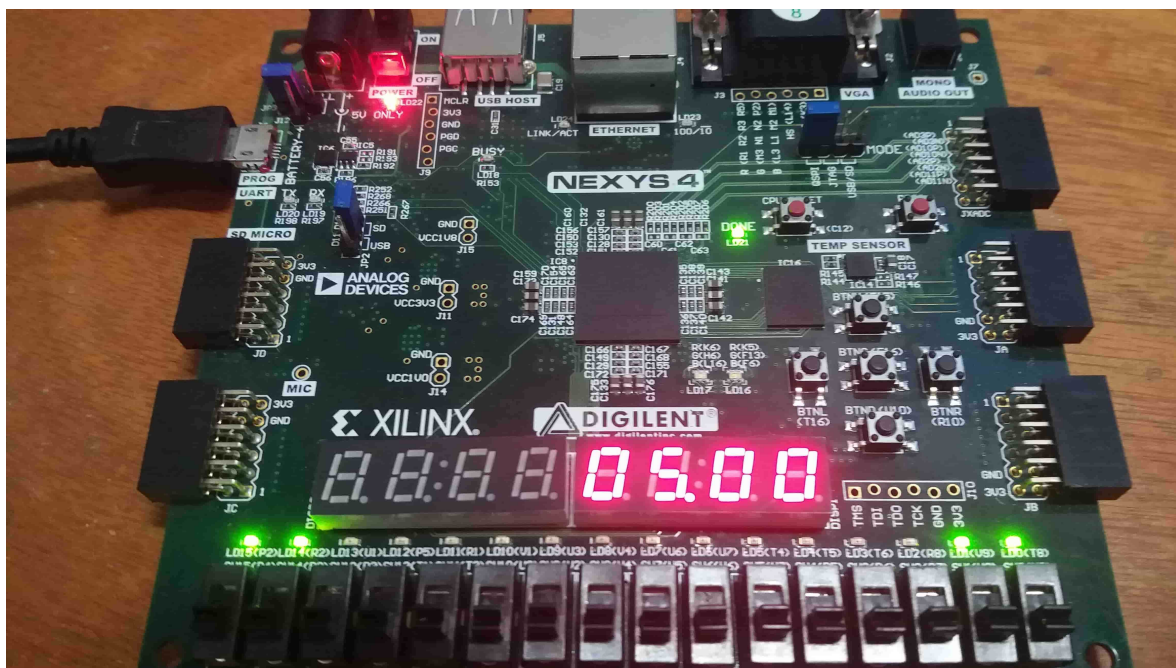
动作	结果
用 vivadow burn 写入程序并开机	显示 hello
按一下 startBtn	显示当前余额为 0
连续按 10 下投 1 元的按钮	2.5 元 5 元可购买灯依次亮起 余额正确
交替按 6 下投 10 元和 20 下投 1 元	余额正确的增加至 90.00 元
按下购买 2.5 元商品的按钮	余额扣除 2.5 元 取饮料灯亮
交替按 14 下买 2.5 元和 10 下买 5 元	余额正确扣除到 2.5 元 可购买 5 元灯熄灭
按 5 下投 1 元按钮	余额增加到 7.5 元 可购买 5 元灯亮
买 5 元商品	余额扣到 2.5 元 取饮料灯亮 可买 5 元灯灭
按下 cancelBtn	显示退钱 2.5 元 退币灯亮 可买 2.5 元灯灭
按下购买 2.5 元商品按钮和充钱按钮	已经退钱，所以没有反应
将 rst 拉上去	关机 所有灯灭 显示 ----



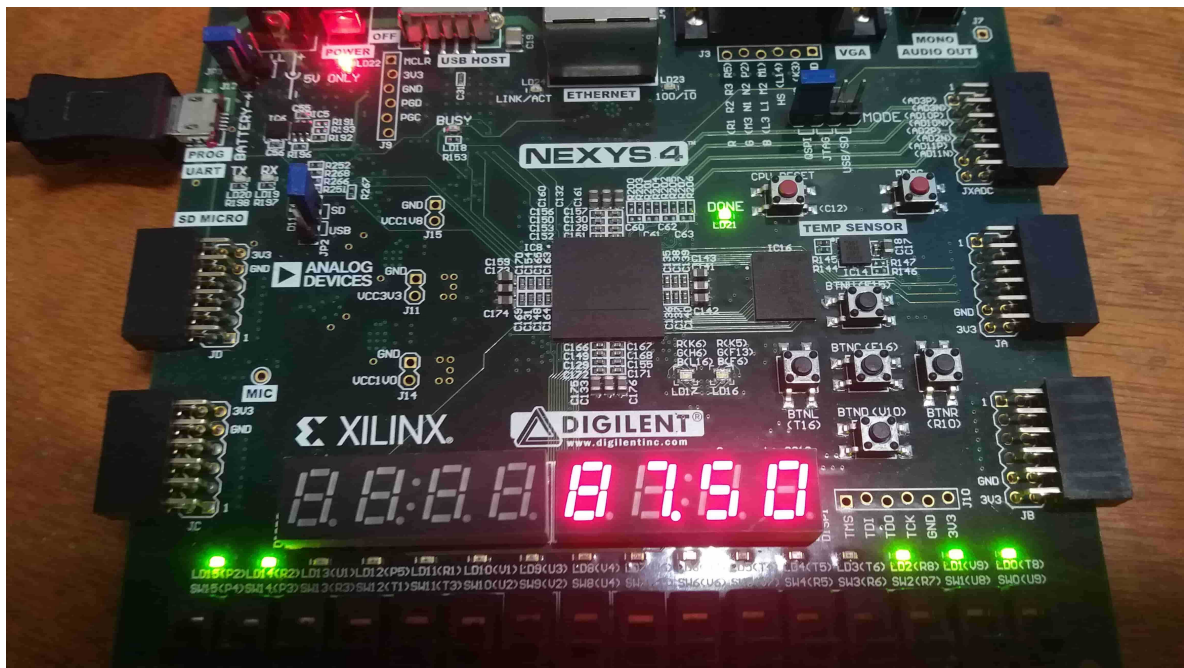
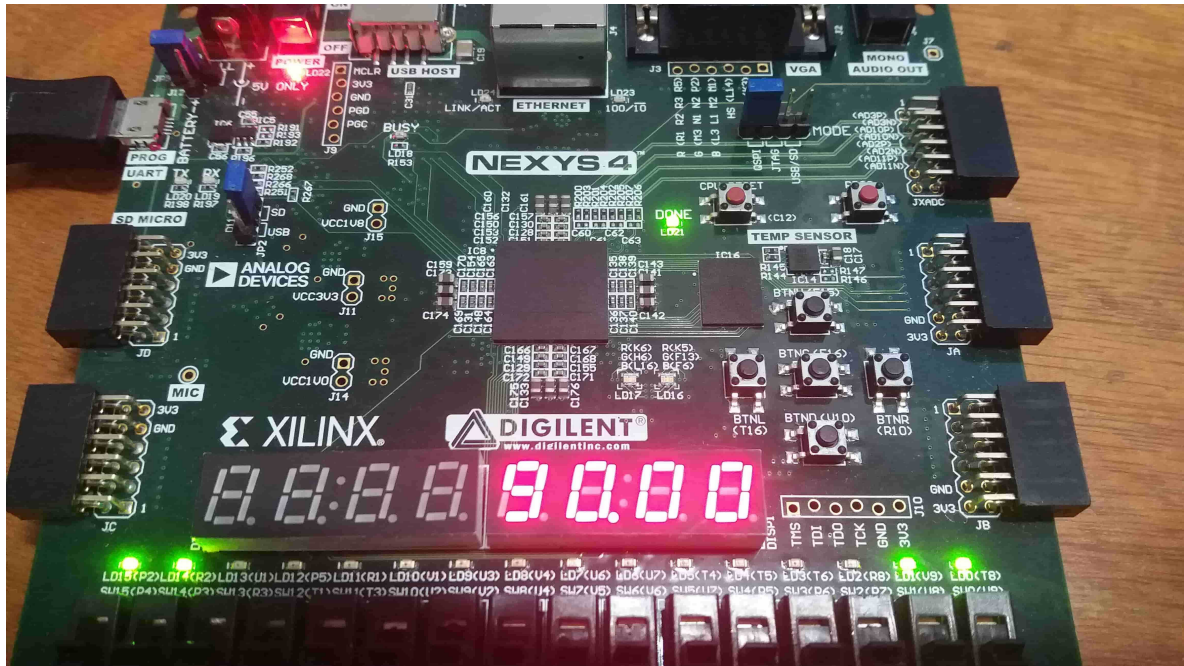
华中科技大学课程设计报告



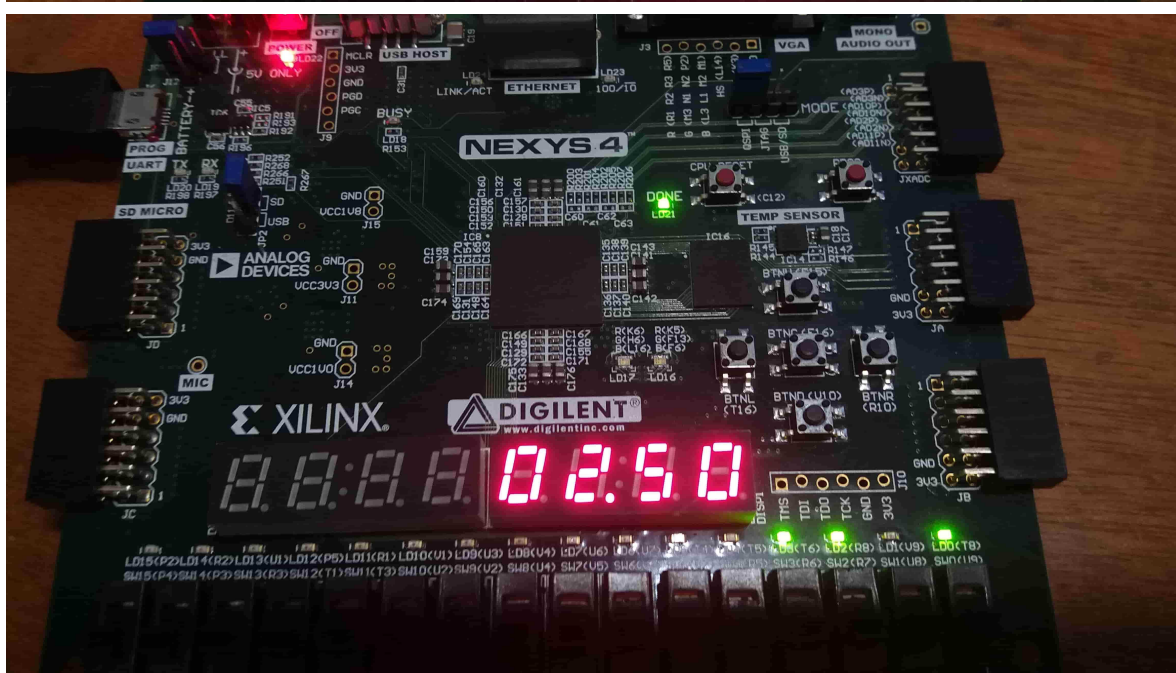
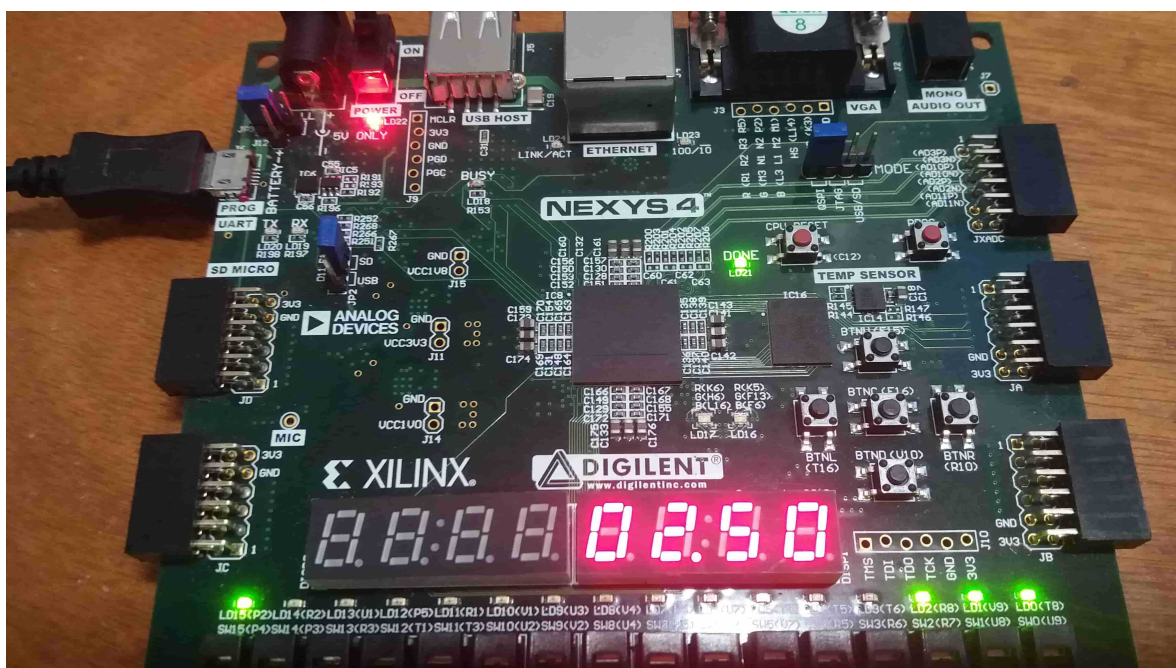
华中科技大学课程设计报告



华中科技大学课程设计报告



华中科技大学课程设计报告



华中科技大学课程设计报告

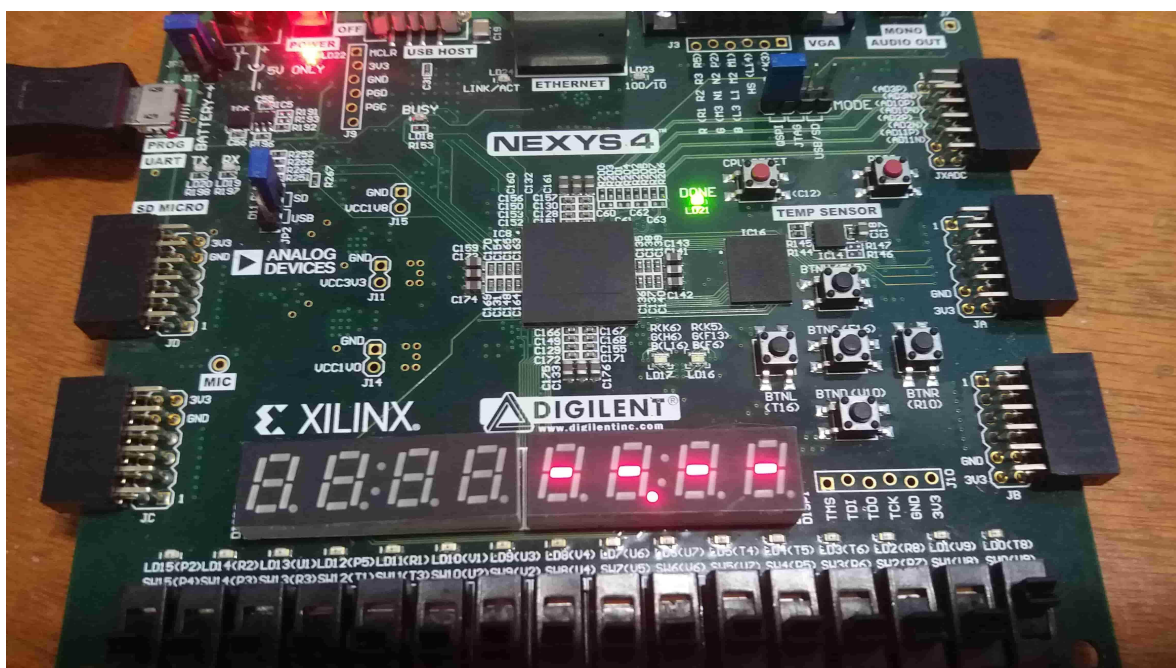
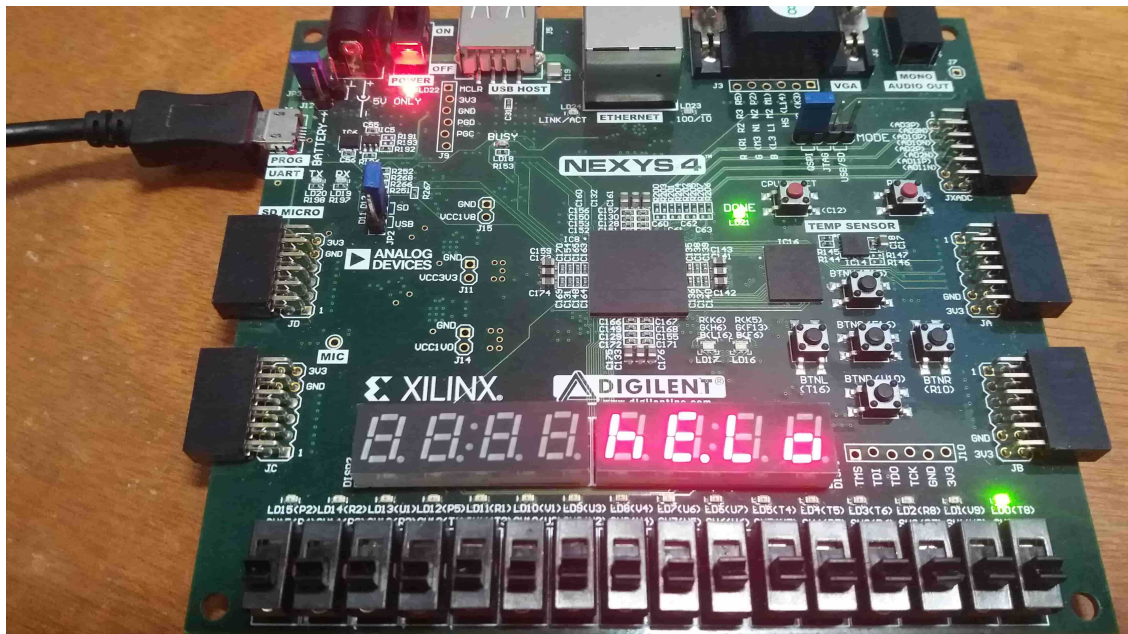


图 2-4 投币和购买人工功能测试

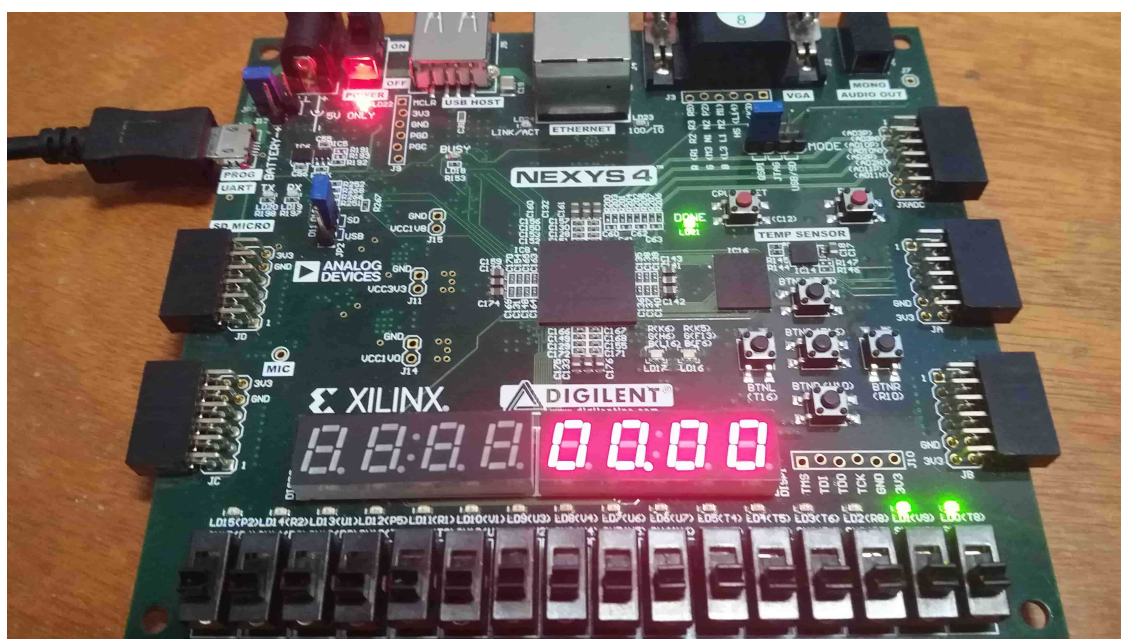
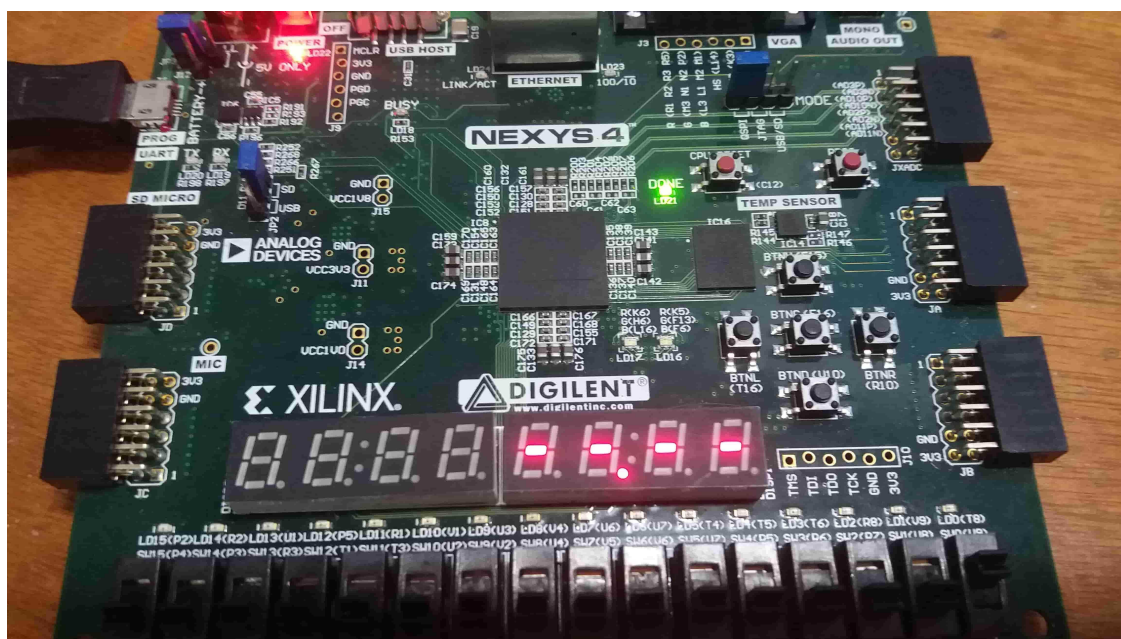
华中科技大学课程设计报告

(2) 开启与退款人工功能测试

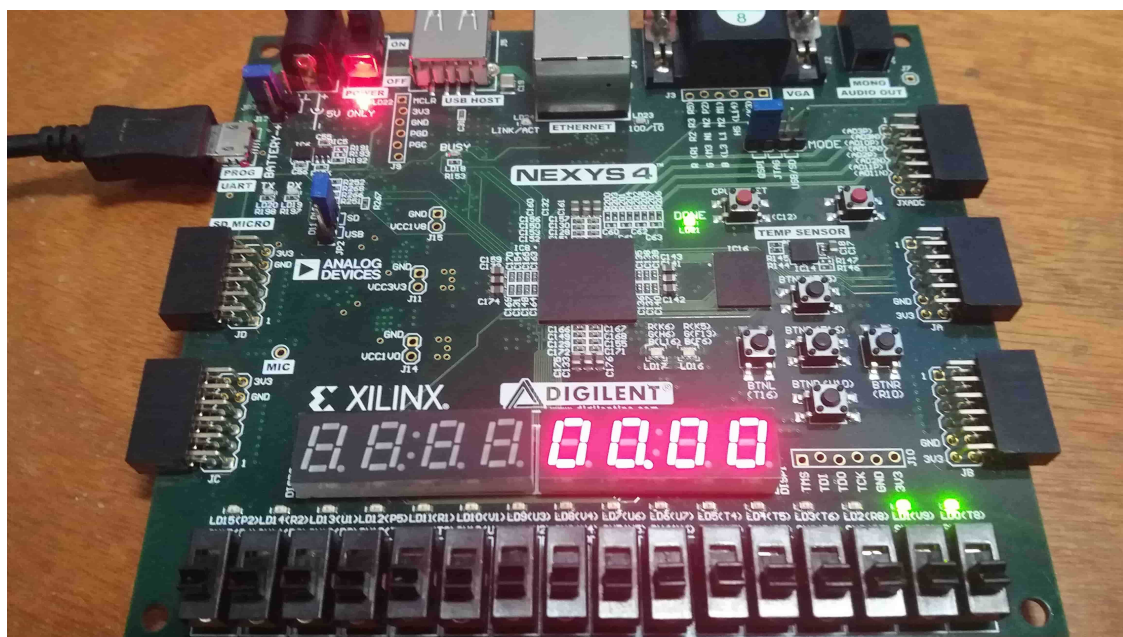
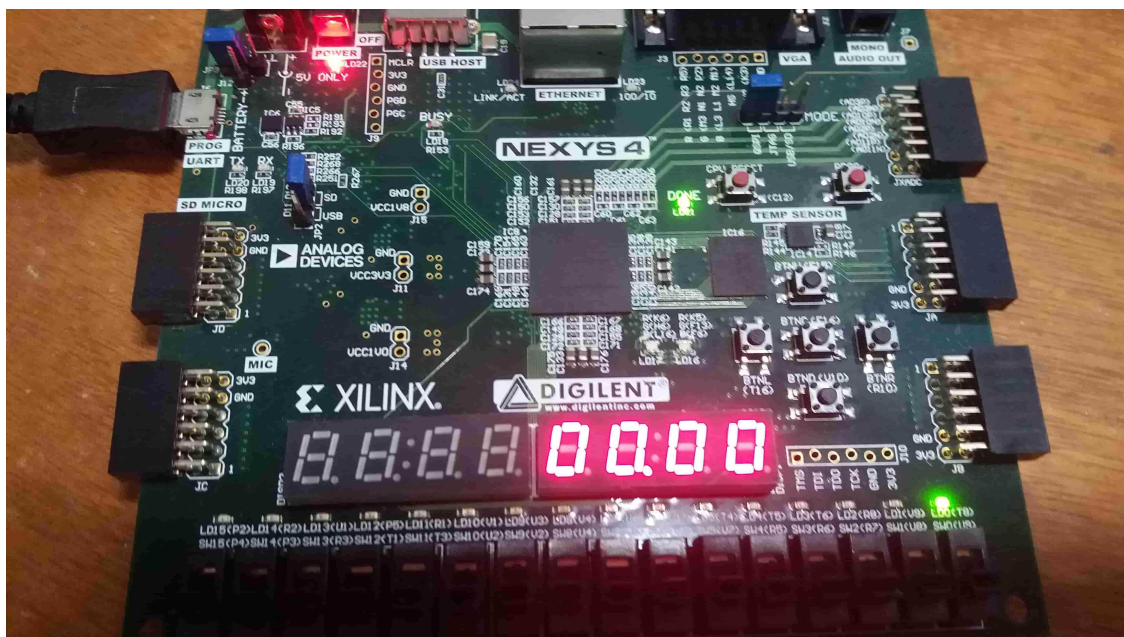
动作	结果
用 vivadow burn 写入程序并开机	0
拉起 rst 关机	显示----
放下 rst 开机 按下 startBtn	显示 00.00
按下 cancelBtn	购买被取消 没有饮料和退款灯 显示 00.00
按下 startBtn，依次购买 2.5 和 5 元商品	因为余额不足 指示灯没有反应
投入 1 元，再次购买两种商品并退币	仍然无法购买商品 退币时退币指示灯亮
拉起 rst 关机 此时依次按下所有其他按钮	没有反应 因为已经关机
放下 rst 开机 尝试购买商品和投币和退币	没有反应 因为没有 start
保持多数按钮同时按下 随机松开约 100 次	指示灯和余额没有进入未定义状态



华中科技大学课程设计报告



华中科技大学课程设计报告



华中科技大学课程设计报告

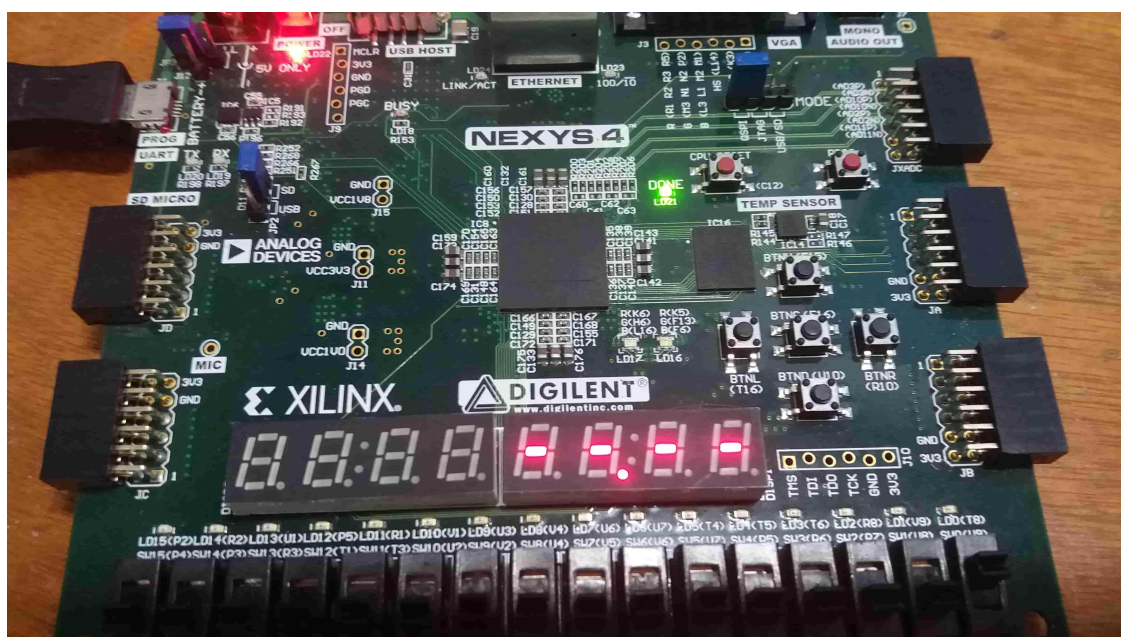
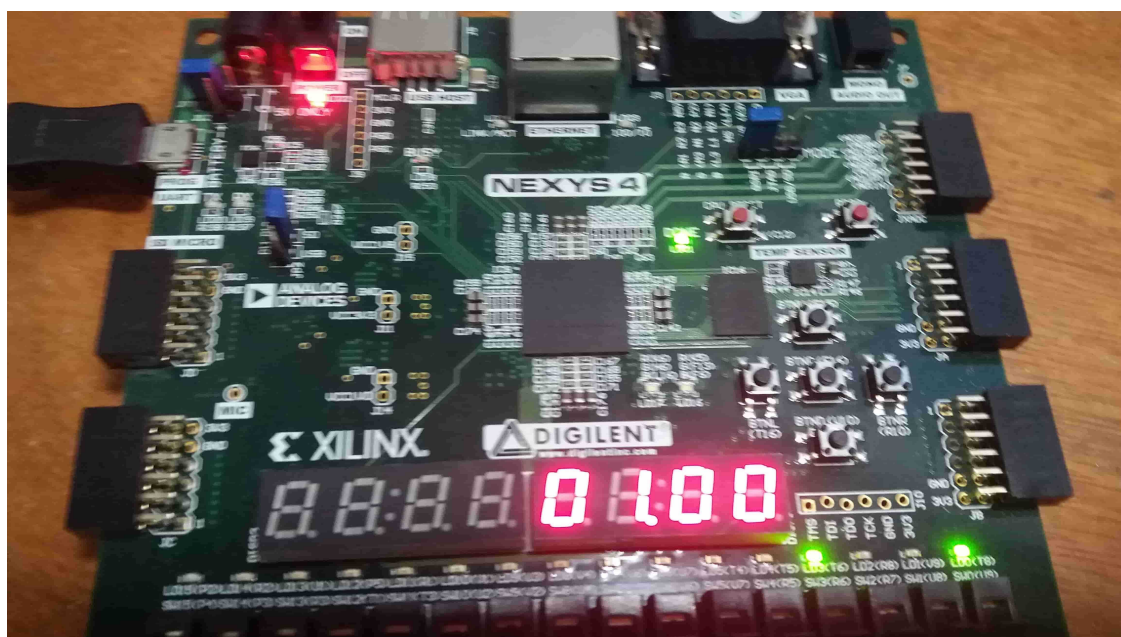


图 2-5 开启与退款人工功能测试

3 总结与心得

3.1 课设总结

为完成自动售卖机控制系统，共完成了如下几方面工作。

(1) 分析题目，对题目要求仔细阅读，摘录要点，构思整个系统的设计框架，并进行状态划分和模块划分

(2) 将大模块划分为小模块并进行整合

(3) 将各个小模块整合到顶层模块中

(4) 对各个模块进行仿真测试，通过后在 **NEXYS4** 开发板上进行测试实现，调试至与题目吻合。

3.2 课设心得

在整个课程设计的过程中，我获得了仅通过日常授课老师们无法传授给我们的宝贵经验，与此同时，老师在课程设计中传授给我们的许多设计方面的想法对我产生了潜移默化的影响。

在分析题目时，我们应该明确题目要求，仔细看清题目的每个细节，这样的成果不仅方便自身调试，也方便他人验收。在报告撰写中，我们应严格按照格式要求，便于他人验收。并且，在课程设计中我们也不应拘泥于任务书，要有自己的看法，是课设变得更加丰富。

总的来说，本次课程设计让我获得了更好的设计思路，并让我对一些基本功能模块的使用有了更深一步的掌握。

4 参考文献

- [1]Wikipedia contributors. "SystemVerilog." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 23 Jul. 2018. Web. 12 Sep. 2018.
- [2]Wikipedia contributors. "Verilog." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 20 Aug. 2018. Web. 12 Sep. 2018.
- [3]Tala, D. K.. "ASIC World Verilog Tutorial". ASIC World. ASIC World, 01 Feb. 1998. Web. 12 Sep. 2018.
- [4] Pieper, K. and 1800_WG SystemVerilog Language Working Group. "IEEE 1800-2017 - IEEE Standard for SystemVerilog--Unified Hardware Design, Specification, and VerificationLanguage". IEEE Standards Association. IEEE Standards Association, 22 Feb. 2018. Web 12 Sep. 2018.

华中科技大学课程设计报告

附录 1 课程设计报告的格式要求补充说明

(1) 正文部分使用五号字，行间距为 1.25 倍行距，其中中文用宋体，英文和数字使用 Times New Roman 字体。

(2) 程序清单，包括仿真程序和后面的附录，使用五号字，行间距为单倍行距，中文注释用宋体，英文和数字使用 Times New Roman 字体。

(3) 所有的章节下面的项目编号使用 (1) (2) (3) 式样，如果还有下一级编号则用 a. b. c. (注意后面有点)。

(4) 所有的图、表、仿真程序要按照章编号，图的编号和名称放在图的下面，表的编号和名称放在上面，程序清单的内容放在双线表格中，且图和图名，表和表名不能分页显示。

(5) 图表的编号和名称均使用小五号字，中文用宋体，英文和数字使用 Times New Roman 字体。

(6) 所有的段落（除章节标题外）均在首行空两个中文字符。所有的章必须另起一页，即按章分页。

(7) 报告完成后，应删除掉报告中红字部分的说明，以及多余的说明。

(8) 参考文献的格式样例如下（注意顺序和标点符号）：

[1]王静康,张凤宝,夏淑倩等.论化工本科专业国际认证与国内认证的“实质性”.高等工程教育研究,2014,5:1-4

[2]Stone J A, Howard L P. A simple technique for observing periodic nonlinearities in Michelson interferometers. Precision Engineering,1998,22(4):220-232

[3]朱印红,袁衍明.Dreamweaver 完美网页设计——技术入门篇.(第一版).北京:中国电力出版社,2006:19~20

[4]Lewis S L. Physics and chemistry of the solar system.北京:北京大学出版社,2014.1~2

[5]陈剑.上博简《民之父母》“而得既塞於四海矣”句解释[EB/OL].简帛研究网站, <http://www.bamboosilk.org/Wssf/2003/chenjian03.htm> . 2003-01-18

附录 2 参考文献格式补充说明

由于模板给出的参考文献格式看起来不符合任何一种广为使用的参考文献格式，其与 *MLA Style Manual* 最相似。因此参考文献使用了 *MLA Style Manual* 格式进行引用。

附录 3 仓库地址 二进制文件地址和备份

可以在此处访问 Git 仓库 <https://github.com/recolic/hust-digital-electronics-exp>

可以在此处下载已编译好的二进制文件 <https://dl.recolic.net/res/main.bit>
<https://dl.recolic.net/res/numshow.bit>

在此将 main.bit 二进制文件先 xz 压缩，再 base64 可打印编码，再 GnuPG 签名，然后书写在此处，以备查验。GnuPG 公钥 ID 为 CA5C947F。

--begin--

此部分文字打印时省略。共计 1400 行

--end--