23302010034 傅子澈 实验报告

实验目的

- 实现一个自动售货机系统，支持六种商品（价格分别为0.5,1,1.5,2,6,13）
- 并有以下操作流程： 首先通过货物种类开关(选用3个swich开关)选择所需要的商品编号，同时将商品编号显示到7段数码管上。然后用四个按钮输入相应的货币, 当输入货币总和正确以后，点亮购买成功标志的led。此处假定输入的货币总和都是正好的，只要等于所选商品价格即可, 不必找零。

设计思路

考虑1ab的设计目的，与以前相比，需要保存付款途中的状态，可以使用寄存器进行储存
此lab的七段数码管显示需要三位（元 角 商品类型），可以在先前七段数码管显示模块的基础上进行细微调整来实现
根据以上要求，进行实现

实现方式

Controller模块

controller包含了所有主要的设计逻辑包含所有输入输出信号

```
`define SIMULATION

module Controller(
    input wire CLK100MHZ,
    input wire rst,
    input wire [2:0] item_type,
    input wire [3:0] money_input,
    input wire reset_money_input,

    output reg purchase_success,
    output [6:0] seg,
    output [7:0] AN,
    output DP
    `ifdef SIMULATION
    ,
    output reg [7:0] amount_paid,
    output reg [7:0] item_price,
    output wire [3:0] debounced_money_input,
    output reg button_pressed
    `endif
    );

    `ifndef SIMULATION
    reg [7:0] amount_paid;
    reg [7:0] item_price;
    wire [3:0] debounced_money_input;
    reg button_pressed;
```

```
    `endif


    Debounce_money debounce_inst (
        .CLK100MHZ(CLK100MHZ),              //Clock signal
        .rst(rst),                      // reset signal
        .money_input(money_input),    // original button input for money
        .stable_money_input(debounced_money_input) // money input with bounce
removed
    );

    /*
    1=0.5CNY
    */


        always @(posedge CLK100MHZ or posedge rst) begin
        if (rst) begin

            amount_paid <= 0;
            item_price <= 0;
            purchase_success <= 0;
            button_pressed <= 0;
        end else begin

            case (item_type)
                3'b001:begin
                  item_price <= 1;
                end
                3'b010: begin
                    item_price <= 2;
                end
                3'b011: begin
                    item_price <= 3;
                end
                3'b100: begin
                    item_price <= 4;
                end
                3'b101: begin
                    item_price <= 13;
                end
                3'b110: begin
                    item_price <= 26;
                end
                default: begin
                item_price <= 0;
                purchase_success <= 0;
                amount_paid<=0;
                end
            endcase


            if (debounced_money_input != 0 && !button_pressed) begin
```

```
                case (debounced_money_input)
                    4'b0001: amount_paid <= amount_paid + 1;
                    4'b0010: amount_paid <= amount_paid + 2;
                    4'b0100: amount_paid <= amount_paid + 4;
                    4'b1000: amount_paid <= amount_paid + 10;
                endcase
                button_pressed <= 1;
            end else if (debounced_money_input == 0) begin

                button_pressed <= 0;
            end


            if (amount_paid >= item_price&&item_price>0) begin
                purchase_success <= 1;
            end
            else begin
                purchase_success<=0;
            end
        end
    end


    HexTo7Segment display (
        .CLK100MHZ(CLK100MHZ),
        .item_type(item_type),
        .amount_paid(amount_paid),
        .AN(AN),
        .seg(seg),
        .DP(DP)
    );

endmodule

endmodule
```

Controller主要由两个重要部分组成，信号定义和always部分
always部分负责处理核心的两个逻辑
当用户拨动开关时，根据开关类型切换商品类型，修改对应的价格
当用户按下按钮时，根据按钮的预设值增加amount paid(四个按钮对应要求种的0.5,1,2,5),一个按钮对应reset,清零输入
当已付钱大于价格时，设置purchase_success点亮led，表示购买成功
购买成功后，按下reset或将所有开关复位可以重置已付金额

使用define ifndef ifdef语句进行条件编译，使得在simulation时可以方便的看到中间变量的值，同时，不需要在实际产生比特流时为中间变量分配引脚和IO标准，保证了仿真和实现的准确度。

防抖动模块

```
module Debounce_money(
    input wire CLK100MHZ,
    input wire rst,
    input wire [3:0] money_input,
    output reg [3:0] stable_money_input
    );
    reg [19:0] debounce_counter;
    reg [3:0] money_input_prev;

    always @(posedge CLK100MHZ or posedge rst) begin
        if (rst) begin
            debounce_counter <= 0;
            stable_money_input <= 0;
            money_input_prev <= 0;
        end else begin
            if (money_input == money_input_prev) begin
                // if the input is not changed,update debounce_counter util it
reaches edge
                debounce_counter <= debounce_counter + 1;
                if (debounce_counter == 20'hFFFFF) begin
                    stable_money_input <= money_input;   //update stable input
                end
            end else begin
                // if input changed reset counter
                debounce_counter <= 0;
                money_input_prev <= money_input;
            end
        end
    end
endmodule
```

这个模块的功能是防止抖动，防止信号短时间多次变化
它的原理如下

- 当接受到money输入时，进行一个计数，当计数期间输入金额信号保持不变时，更新计数
- 当计数达到预设值时，更新stable输出用于Controller进行付款逻辑
- 如果money变化，则重置计数

这个方法可以有效防止短时间多次变化的信号影响输出

显示模块

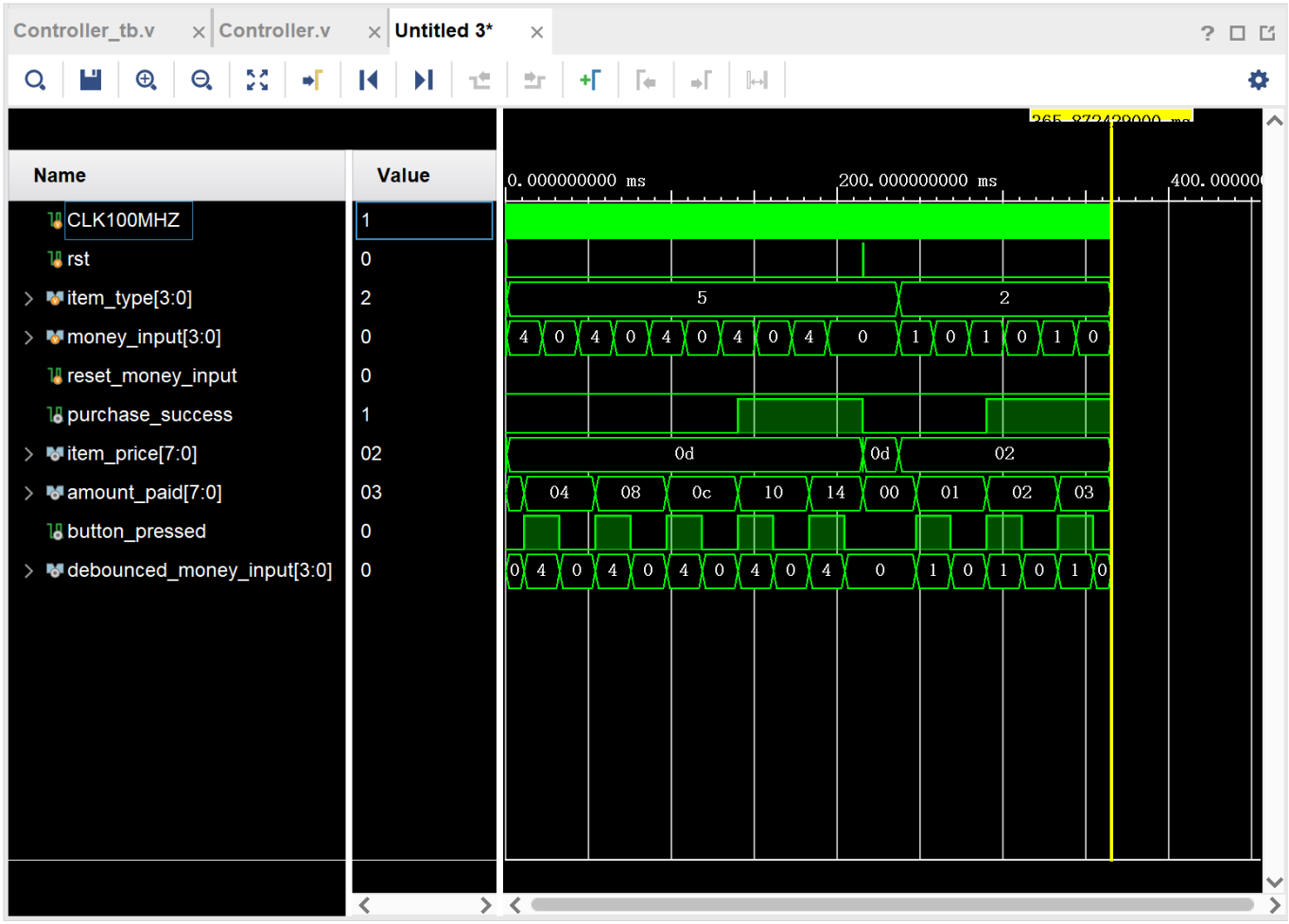
该模块大多数逻辑和上一次实验相同，不同的是，为了将金额转化为元-角形式，进行了转化

```
always @(*) begin
    total_jiao = amount_paid * 5;
    yuan_tens = (total_jiao / 10) / 10;
    yuan_units = (total_jiao / 10) % 10;
    jiao_units = total_jiao % 10;
```

```
end
```

这是一个经典的转化逻辑，不多赘述，将其转化后，显示在数码管上的是可读性较好的金额

仿真效果



仿真效果如图，进行了两轮测试。第一次选用商品5（6.5CNY，对应13）作为测试对象
每次付钱2元。每次付钱一段时间后，都进行同样时间的停顿，防止被防抖动误伤。经过4个循环后，可以看到Purchase success信号变为1，说明购买成功。

然后短暂设置reset信号，将付钱总额调回0并切换商品类型为2（1CNY,对应2），每次付钱一元，2个循环后，购买成功

仿真证明了代码的准确性，包括其付钱和累计的逻辑，判断购买成功的逻辑以及重置的逻辑，均满足要求

遇到的问题和解决

更新已付金额

选择采取了计数器的方法，一开始计数器没有正确设置，导致储存后的money_input会被多次加到已付金额。
解决方法

添加button_pressed信号位，保证在button被press一次后只被添加一次

防抖动处理

我选用计数器而不是助教给出方法是因为个人更熟悉目前使用的方法。他们都能完成防抖动的作用，且作为一个单独模块也保持了耦合性不会过高。

购买成功的显示

一开始没有仔细考虑这一点，直接在已付款金额大于价格时将其设置为1，导致没有选择商品时也会使其发亮这是因为没有考虑到电路不是顺序执行的，最后解决方法是进行额外的逻辑判断，只有当价格是大于0时才进一步判断是否应当将灯亮起

处理较为复杂的仿真

防抖动处理导致仿真中随意的赋值无法达到预期的效果。因此，需要更谨慎的调整wait的时间，以保证输入的money_input可以准确的被识别并加入到总的已付金额中。
在这里，我使用了条件编译来决定是否将中间值放入最后的输出。 这为之后的pj也有了一定帮助作用

实验结论

本次实验实现了一个售货机系统，支持了要求的所有功能且做了防抖动处理，完成了所有任务