EDA实验2实验报告

一、实验目的

- 实践基于 HDL 设计和实现时序逻辑电路的流程和方法；
- 掌握用 HDL 实现有限状态机的方法；
- 实践利用 FPGA 解决实际问题的方法。

二、实验内容

- 利用实验板上的拨码开关和按键开关模拟投币、购物和退币输入，用发光二极管模拟各种提示信息，用数码管显示余额，实现一个自动售货机。

三、实验设计思路

- 通过拨码开关，获得当前的实验模式，包括：投币模式、购物模式、退币模式、错误模式；
- 对于当前模式，对于按键开关，进行处理，使钱数改变；
- 对当前的钱数进行显示。

四、各模块实现方法

- 获得当前实验模式：并通过一个发光二极管判断当前模式是否为错误模式。

```
module mode_change(sw, mode, led_wrongmode);
input [3:0] sw;              // 拨码开关
output [1:0] mode;
output led_wrongmode;        // 指示灯

reg [1:0] mode_r;
reg led_r;
always @ (sw)
begin
    case (sw)
        4'b1000:
        begin
            mode_r = 2'b01;     // 投币模式
            led_r = 0;
        end
        4'b0100:
        begin
            mode_r = 2'b10;     // 购物模式
            led_r = 0;
        end
        4'b0010:
        begin
            mode_r = 2'b11;     // 退币模式
            led_r = 0;
        end
        default:
```

```
        begin
            mode_r = 2'b00;     // 错误模式 (wrong mode)
            led_r = 1;
        end
    endcase
end
assign led_wrongmode = led_r;
assign mode = mode_r;
endmodule
```

- 对于当前模式，对于按键开关，进行处理，使钱数改变；

```
module work(clk,mode,led_fail,led_get,led_withdraw,key,money);
input clk;                  //时间
input [1:0] mode;           //模式
input [3:0] key;            //按键开关
output led_fail, led_get, led_withdraw; //分别表示购买失败，取货，退款的发光二极管
output [7:0] money;         // money为二进制数，转为十进制除以2即为钱数

reg [7:0] money_r;
reg formal1,formal2,formal3;
reg r_ledfail,r_ledget,r_ledwithdraw;
reg [10:0] cur;

always @ (posedge clk)
begin
    cur <= cur+1'b1;
end

always@(posedge cur[9:9])   // 每过2^10次周期（一周期为1/40mHz）检测一次
begin
    formal1 <= key[0:0];
    formal2 <= key[1:1];
    formal3 <= key[2:2];
end

always @ (posedge cur[9:9])
begin
    if (mode == 2'b11)      //退币模式
    begin
        money_r <= 0;
        r_ledwithdraw = 1;
    end
    if (mode == 2'b00)      //错误模式，令购买失败，取货，退款的发光二极管关闭
    begin
        r_ledfail = 0;
        r_ledget = 0;
        r_ledwithdraw = 0;
    end
    if (formal1 == 1 && key[0:0] == 0) // 按键开关1被按了
    begin
    case (mode)
        2'b01: begin        // 投币模式
```

```
            money_r <= money + 1;
            r_ledfail = 0;
            r_ledget = 0;
        end
        2'b10: begin        // 购物模式
            if (money > 2)
            begin
                money_r <= money - 3;
                r_ledfail = 0;
                r_ledget = 1;
            end
            else begin
                money_r <= money;
                r_ledfail = 1;
                r_ledget = 0;
            end
        end
        2'b11: begin    // 退币模式
            money_r <= 0;
            r_ledfail = 0;
            r_ledget = 0;
        end
        default:begin   // 错误模式 (wrong mode)
            money_r <= 0;
            r_ledfail = 0;
            r_ledget = 0;
        end
    endcase
    end
    if (formal2 == 1 && key[1:1] == 0) // 按键开关2被按了
    begin
    case (mode)
        2'b01: begin        // 投币模式
            money_r <= money + 2;
            r_ledfail = 0;
            r_ledget = 0;
        end
        2'b10: begin        // 购物模式
            if (money > 4)
            begin
                money_r <= money - 5;
                r_ledfail = 0;
                r_ledget = 1;
            end
            else begin
                money_r <= money;
                r_ledfail = 1;
                r_ledget = 0;
            end
        end
        2'b11: begin    // 退币模式
            money_r <= 0;
            r_ledfail = 0;
```

```
            r_ledget = 0;
        end
        default:begin   // 错误模式 (wrong mode)
            money_r <= 0;
            r_ledfail = 0;
            r_ledget = 0;
        end
    endcase
    end
    if (formal3 == 1 && key[2:2] == 0) // 按键开关3被按了
    begin
    case (mode)
        2'b01: begin        // 投币模式
            money_r <= money + 10;
            r_ledfail = 0;
            r_ledget = 0;
        end
        2'b10: begin        // 购物模式
            money_r <= money;
            r_ledfail = 0;
            r_ledget = 0;
        end
        2'b11: begin    // 退币模式
            money_r <= 0;
            r_ledfail = 0;
            r_ledget = 0;
        end
        default:begin   // 错误模式 (wrong mode)
            money_r <= 0;
            r_ledfail = 0;
            r_ledget = 0;
        end
    endcase
    end
end
assign led_fail = r_ledfail;
assign led_get = r_ledget;
assign led_withdraw = r_ledwithdraw;
assign money = money_r;
endmodule
```

- 对当前的钱数进行显示。

```
module show_money10(clk,num,seg,dig);
input clk;
input [7:0] num;
output [6:0] seg;
output [2:0] dig;

reg [3:0] num_r1,num_r2,num_r3; // 三位，num_r1为小数位，num_r2为个位，num_r3为十位

reg [6:0] num_r;
reg [11:0] cur;
```

```
reg [2:0] dig_r;
reg [6:0] seg_r;

assign dig = dig_r;
assign seg = ~seg_r;

always @ (num)
begin
    num_r1 = (num % 2) * 5;
    num_r2 = (num / 2)% 10;
    num_r3 = num / 20;

end

always @ (num_r1 or num_r2 or num_r3 or cur[11:10])
begin
    case (cur[11:10])
        2'b00: num_r <= num_r3;
        2'b01: num_r <= num_r2;
        2'b10: num_r <= num_r1;
        2'b11: num_r <= 4'b1111;
    endcase
end

always @ (cur[11:10])
begin
    case (cur[11:10])
        2'b00: dig_r <= 3'b011;
        2'b01: dig_r <= 3'b101;
        2'b10: dig_r <= 3'b110;
        2'b11: dig_r <= 3'b111;
    endcase
end

always @ (num_r)
begin
    case (num_r)
        4'b0000: seg_r <= 7'b0111111;   //0
        4'b0001: seg_r <= 7'b0000110;   //1
        4'b0010: seg_r <= 7'b1011011;   //2
        4'b0011: seg_r <= 7'b1001111;   //3
        4'b0100: seg_r <= 7'b1100110;   //4
        4'b0101: seg_r <= 7'b1101101;   //5
        4'b0110: seg_r <= 7'b1111101;   //6
        4'b0111: seg_r <= 7'b0100111;   //7
        4'b1000: seg_r <= 7'b1111111;   //8
        4'b1001: seg_r <= 7'b1101111;   //9
        default: seg_r <= 7'b0000000;       //其余则不显示
    endcase
end

always @ (posedge clk)
begin
```

```
    cur <= cur+1'b1;
end

endmodule
```

五、实验中遇到的问题与解决方法

一开始在每次clk的下降沿都进行一次按键开关的工作判断，但发现这样无法理想工作。

解决方法：改变clk，对其进行分频处理，使其工作正常。