# 《数字逻辑与数字系统》实验报告　　　　天津大学本科生实验报告专用纸

| | | | | | |
|---|---|---|---|---|---|
| 学部/院 | 智能与计算学部 | 年级 | 2021 | 班级 | 软工 3 班 |
| 姓名 | 陈昊昆 | 学号 | 3021001196 | 实验日期 | 5.17 |

实验项目名称＿自动贩售机的设计与实现＿＿＿＿＿＿＿＿＿

## 一. 实验目的

1. 掌握有限状态机的设计方法。;
2. 能够使用 SystemVerilog 进行三段式状态机的建模。

## 二. 实验内容

采用有限状态机，基于 SystemVerilog HDL 设计并实现一个报纸自动贩售机。 整个工程的顶层模块如图 4-3 所示，输入/输出端口如表 4-1 所示。使用 4 个七段数码管实时显示已付款和找零情况。其中，两个数码管对应“已付款”，另两个 数码管对应“找零”，单位为分。通过 1 个拨动开关对数字钟进行复位控制。使用 两个按键模拟投币，其中一个按键对应 5 分，另一个按键对应 1 角。使用 1 个 LED 灯标识出售是否成功，灯亮表示出售成功，否则表示已付款不够，出售失败。

假设报纸价格为 15 分，合法的投币组合包括：

● 1 个 5 分的硬币和一个 1 角的硬币，不找零

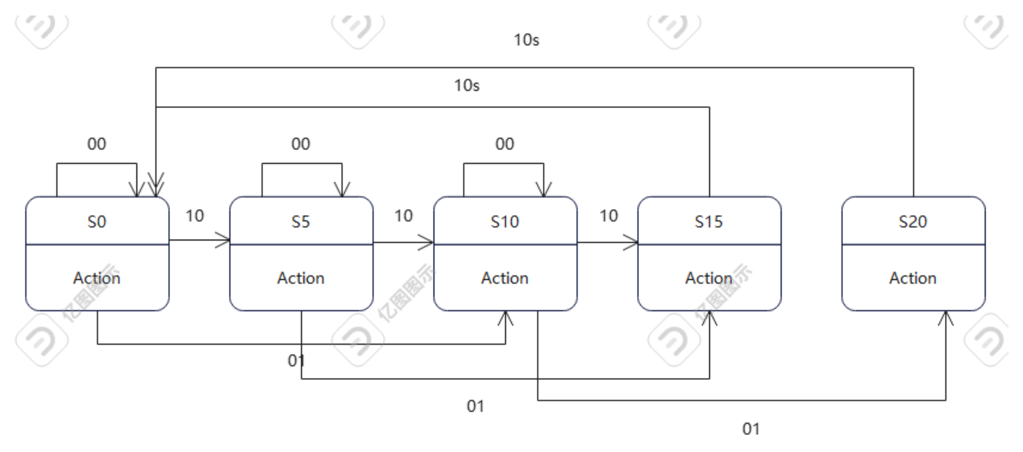
● 3 个五分的硬币，不找零

● 1 个 1 角的硬币和一个 5 分的硬币，不找零

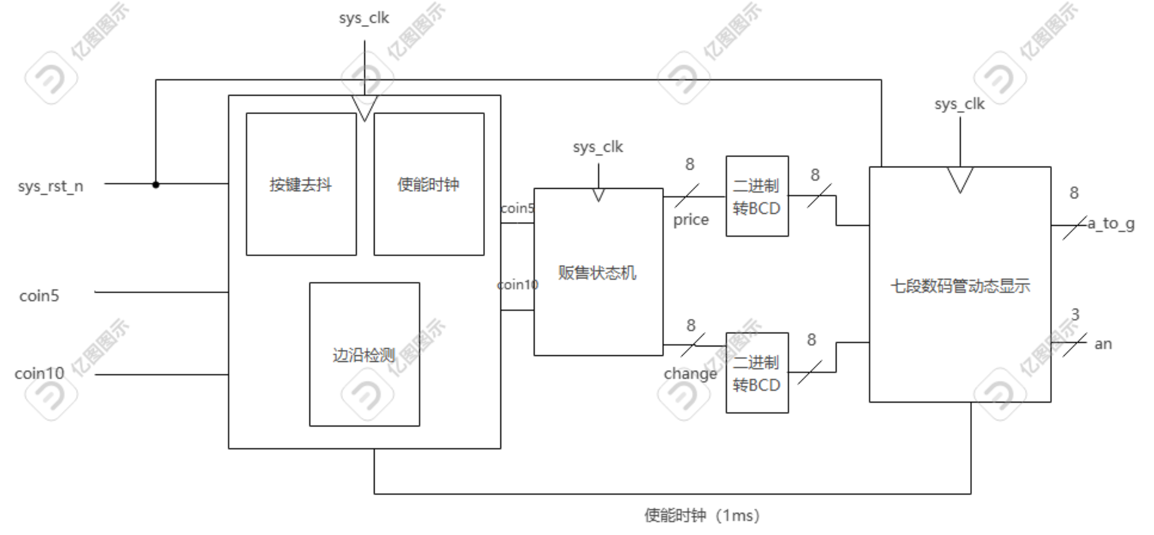
● 两个 1 角的硬币是合法的，找零 5 分。 当投入硬币的组合为上面 4 种之一时，则购买成功，LED 灯亮。购买成功后，LED 灯持续亮 10 秒，然后自动熄灭，4 个数码管也恢复为 0。

## 三．实验原理与步骤（注：步骤不用写工具的操作步骤，而是设计步骤）

1. 画出自动贩售机的状态转换图。



2. 画出 自动贩售机 电路 的 原理图 （模块级别即可，如 使能时钟模块 、 边沿检测模块等



3. 报纸自动贩售机的 SystemVerilog 代码。

顶层模块

```
module vend(
        input sys_clk, sys_rst_n,
        input coin5, coin10,
        output [3 : 0] an,
        output [7 : 0] a_to_g,
        output open
        );

        logic coin5de, coin10de, coin5ed, coin10ed, clkflag;
        logic [7:0] change, price, changed, priced;

        debounce debouncemux5 (.sys_clk(sys_clk), .sys_rst_n(sys_rst_n), .i_btn(coin5), .o_btn(coin5de));
        debounce                                                    debouncemux10
(.sys_clk(sys_clk), .sys_rst_n(sys_rst_n), .i_btn(coin10), .o_btn(coin10de));
        edge_detect
edge_detectmux5(.sys_clk(sys_clk),   .sys_rst_n(sys_rst_n), .i_btn(coin5de), .posedge_flag(coin5ed));
        edge_detect
edge_detectmux10(.sys_clk(sys_clk),   .sys_rst_n(sys_rst_n), .i_btn(coin10de), .posedge_flag(coin10e
d));
        clken clkenmux(.clk(sys_clk), .rst_n(sys_rst_n), .clk_flag(clkflag));
        fsm
fsmmux(.sys_clk(sys_clk),   .sys_rst_n(sys_rst_n),  .coin5(coin5ed),  .coin10(coin10ed),  .light(open),  .c
hange(change), .price(price));
        bin2bcd_0 changebin (.bin(change), .bcd(changed));
        bin2bcd_0 pricebin (.bin(price), .bcd(priced));
        seven_seg_disp
ssdmux(.sys_clk(sys_clk), .clk(clkflag), .rst_n(sys_rst_n), .num0(changed[7:4]), .num1(changed[3:0]), .
num2(priced[7:4]), .num3(priced[3:0]), .a_to_g(a_to_g), .an(an));

endmodule
```

按键去抖

```
module debounce(
        input    logic   sys_clk,
        input    logic   sys_rst_n,
        input    logic   i_btn,
        output   logic   o_btn
);

logic [21:0] count;
logic clk_flag;

logic [2:0]i_btn_prev;


always_ff @(posedge sys_clk) begin

    if(~sys_rst_n) count <= 0;


    else if (count == 22'd249999)

        count <= 22'd0;

    else

        count <= count + 22'd1;

end


always_ff @(posedge sys_clk) begin

    if (count == 22'd249999) clk_flag <= 1'b1;

    else clk_flag <= 1'b0;

end



    always_ff @(posedge sys_clk) begin

        if (~sys_rst_n) i_btn_prev <= 0;

        else if (clk_flag)begin

            i_btn_prev[0] <= i_btn;

            i_btn_prev[1] <= i_btn_prev[0];

            i_btn_prev[2] <= i_btn_prev[1];

            end

    end


    assign o_btn = i_btn_prev[0] && i_btn_prev[1] && i_btn_prev[2];


endmodule
```

边沿检测

```
module edge_detect(

    input   logic   sys_clk,

    input   logic   sys_rst_n,

    input   logic   i_btn,

    output logic   posedge_flag

);



    logic [1:0]i_btn_prev;
    always_ff @(posedge sys_clk) begin
        if (~sys_rst_n) i_btn_prev <= 0;
        else begin
            i_btn_prev[0] <= i_btn;
            i_btn_prev[1] <= i_btn_prev[0];
            end
    end

    assign posedge_flag = i_btn_prev[0] & ~i_btn_prev[1];


endmodule
```

使能时钟

```
module clken #(
parameter SYS_CLK_FREQ = 25_000_000,
parameter TARGET_CLK_FREQ = 1_000
)
(
input logic clk,
input logic rst_n,
output logic clk_flag
);

localparam CNT_MAX = SYS_CLK_FREQ / TARGET_CLK_FREQ;
logic [14 : 0] count;
always_ff @(posedge clk) begin
    if (~rst_n) count <= 0;
    else if (count == CNT_MAX - 1) count <= 0;
    else count <= count + 1;
    end

always_ff @(posedge clk) begin
    if (~rst_n) clk_flag <= 1'b0;
    else if (count == CNT_MAX - 1) clk_flag <= 1'b1;
    else clk_flag <= 1'b0;
end
endmodule
```

有限状态机

```
module fsm(
    input   logic   sys_clk,
    input   logic   sys_rst_n,
    input   logic   coin5,
    input   logic   coin10,
    output logic   light,
    output logic [7 : 0] change,
    output logic [7 : 0] price
);

   parameter S0    = 3'b000,
             S5       = 3'b001,
             S10       = 3'b010,
             S15       = 3'b011,
             S20       = 3'b100;

logic   [2:0] cstate, nstate;
logic [27:0] count;
logic stop;

    // 第一个 always 块为时序逻辑，用于建模状态寄存器，实现从次态到现态的转移
    always_ff @(posedge sys_clk) begin
        if (~sys_rst_n) begin
            cstate <= S0;
        end
        else begin
            cstate <= nstate; // 采用非阻塞赋值
        end
    end

    // 第二个 always 模块为组合逻辑，用于描述状态转移条件的判断
    always @(*) begin
        case (cstate)
            S0: if ({coin5, coin10} == 2'b00) nstate = S0;
                else if({coin5, coin10} == 2'b10) nstate = S5;
                else if({coin5, coin10} == 2'b01) nstate = S10;
                else nstate = S0;
            S5: if ({coin5, coin10} == 2'b00) nstate = S5;
                else if({coin5, coin10} == 2'b10) nstate = S10;
                else if({coin5, coin10} == 2'b01) nstate = S15;
                else nstate = S5;
            S10: if ({coin5, coin10} == 2'b00) nstate = S10;
                 else if({coin5, coin10} == 2'b10) nstate = S15;
                 else if({coin5, coin10} == 2'b01) nstate = S20;
```

```
                        else nstate = S10;
                S15: if(~stop) nstate = S15;
                        else nstate = S0;
                 S20: if(~stop) nstate = S20;
                         else nstate = S0;
        endcase

end

// 第三个 always 模块可以是组合逻辑，也可以是时序逻辑，用于描述输出逻辑
always_ff @(posedge sys_clk) begin
        case (cstate)
                S0: begin
                        light <= 0;
                        change <= 0;
                        price <= 0;
                end
                S5: begin
                        light <= 0;
                        change <= 0;
                        price <= 5;
                end
                S10: begin
                        light <= 0;
                        change <= 0;
                        price <= 10;
                end
                S15: begin
                        light <= 1;
                        change <= 0;
                        price <= 15;
                end
                S20: begin
                        light <= 1;
                        change <= 5;
                        price <= 20;
                end
        endcase
end

always_ff @(posedge sys_clk) begin
        if (~sys_rst_n) begin
```

```
            stop <= 0;
            count <= 0;
        end
        if(light) begin
             if(count == 28'd24_999_9999) begin
             count <= 28'd0;
             stop <= 1;
         end
         else
             count <= count + 28'd1;
        end
    end
endmodule
```

七段数码管动态显示

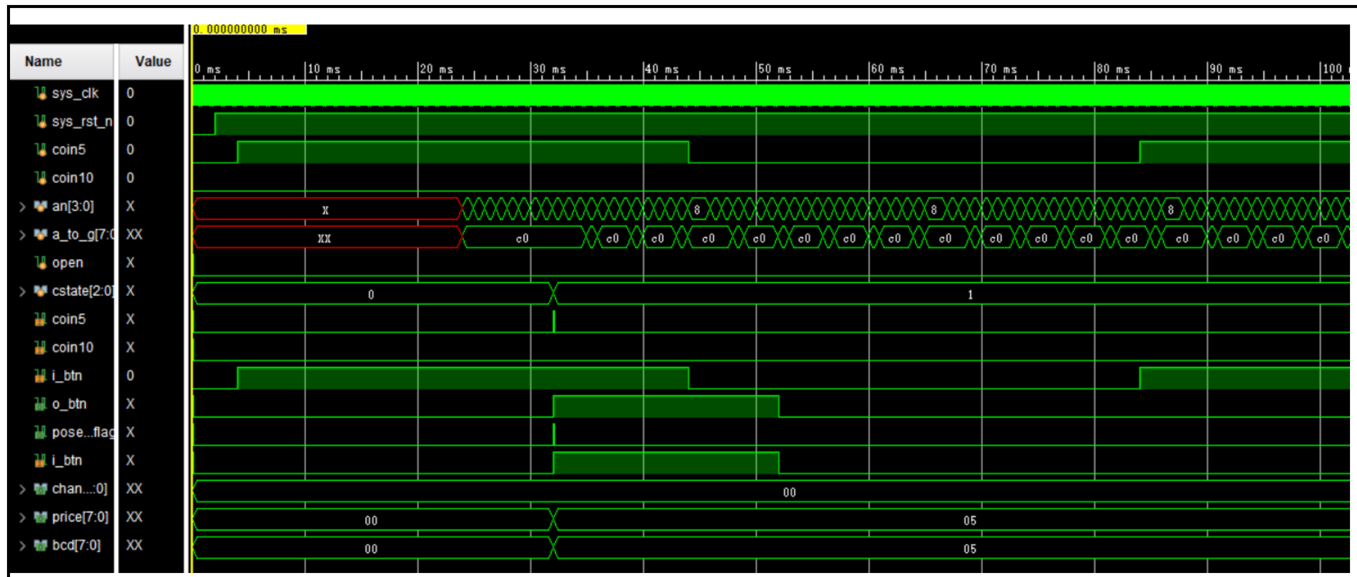
```
module seven_seg_disp(
    input logic sys_clk,
    input logic clk,
    input logic rst_n,
    input logic [3:0] num0 ,
    input logic [3:0] num1 ,
    input logic [3:0] num2 ,
    input logic [3:0] num3 ,
    output logic [7:0] a_to_g,
    output logic [3:0] an
);

logic [1:0] count;
logic [5:0] cnt;

parameter [7:0] enc_0 = 8'b11000000;
parameter [7:0] enc_1 = 8'b11111001;
parameter [7:0] enc_2 = 8'b10100100;
parameter [7:0] enc_3 = 8'b10110000;
parameter [7:0] enc_4 = 8'b10011001;
parameter [7:0] enc_5 = 8'b10010010;
parameter [7:0] enc_6 = 8'b10000010;
parameter [7:0] enc_7 = 8'b11111000;
parameter [7:0] enc_8 = 8'b10000000;
parameter [7:0] enc_9 = 8'b10010000;

always_ff @(posedge sys_clk) begin
```

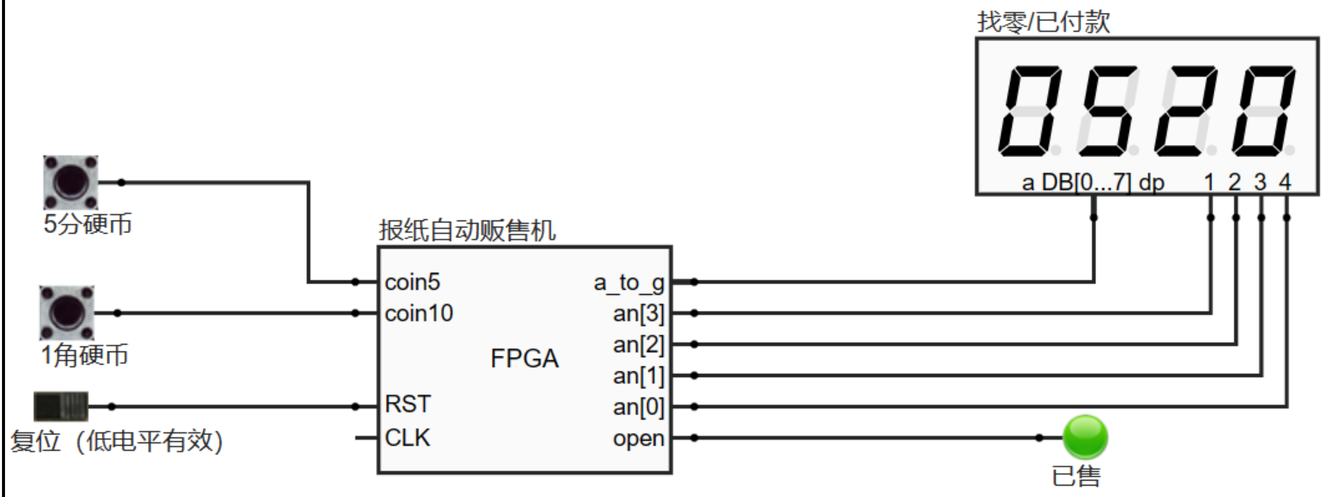
```
	if(~rst_n) cnt <= 0;
	else if (cnt == 20) cnt <= 0;
	else cnt <= cnt + 1;
end

always_ff @(posedge clk) begin
	if (count == 2'd3 || cnt == 0) count <= 0;
	else count <= count + 1;
end

always_ff @(posedge clk) begin
	if(count == 0) begin
		an <= 4'b1000;
		case (num0)
			4'd0: a_to_g <= enc_0;
			4'd1: a_to_g <= enc_1;
			4'd2: a_to_g <= enc_2;
			4'd3: a_to_g <= enc_3;
			4'd4: a_to_g <= enc_4;
			4'd5: a_to_g <= enc_5;
			4'd6: a_to_g <= enc_6;
			4'd7: a_to_g <= enc_7;
			4'd8: a_to_g <= enc_8;
			4'd9: a_to_g <= enc_9;
		endcase
	end
	if(count == 1) begin
	an <= 4'b0100;
		case (num1)
			4'd0: a_to_g <= enc_0;
			4'd1: a_to_g <= enc_1;
			4'd2: a_to_g <= enc_2;
			4'd3: a_to_g <= enc_3;
			4'd4: a_to_g <= enc_4;
			4'd5: a_to_g <= enc_5;
			4'd6: a_to_g <= enc_6;
			4'd7: a_to_g <= enc_7;
			4'd8: a_to_g <= enc_8;
			4'd9: a_to_g <= enc_9;
		endcase
	end
	if(count == 2) begin
```

```
an <= 4'b0010;
        case (num2)
                4'd0: a_to_g <= enc_0;
                4'd1: a_to_g <= enc_1;
                4'd2: a_to_g <= enc_2;
                4'd3: a_to_g <= enc_3;
                4'd4: a_to_g <= enc_4;
                4'd5: a_to_g <= enc_5;
                4'd6: a_to_g <= enc_6;
                4'd7: a_to_g <= enc_7;
                4'd8: a_to_g <= enc_8;
                4'd9: a_to_g <= enc_9;
        endcase
end
if(count == 3) begin
an <= 4'b0001;
        case (num3)
                4'd0: a_to_g <= enc_0;
                4'd1: a_to_g <= enc_1;
                4'd2: a_to_g <= enc_2;
                4'd3: a_to_g <= enc_3;
                4'd4: a_to_g <= enc_4;
                4'd5: a_to_g <= enc_5;
                4'd6: a_to_g <= enc_6;
                4'd7: a_to_g <= enc_7;
                4'd8: a_to_g <= enc_8;
                4'd9: a_to_g <= enc_9;
        endcase
    end
end



endmodule
```

**四．仿真与实验结果（注：仿真需要给出波形图截图，截图要清晰，如果波形过长，可以分段截取；实验结果为远程 FPGA 硬件云平台的截图）**

注：远程 FPGA 硬件云平台截图只需要一个测试激励即可





## 五．实验中遇到的问题和解决办法

问题：如何使得购买成功后，灯持续亮 10s，然后熄灭，数字归零。

解决：在贩售状态机内部加入计数器充当计时器，在达到 S15 或 S20 后开始计时，达到十秒后回到状态 S0

**教师签字：**

**年　月　日**