《数字逻辑与数字系统》实验报告

学院 智算学部 年级 2019 级 班级 计科一班 姓名 李润泽 学号 3019244266

课程名称 数字逻辑与数字系统 实验日期 2021.6.1 成绩_____

同组实验者_____

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。

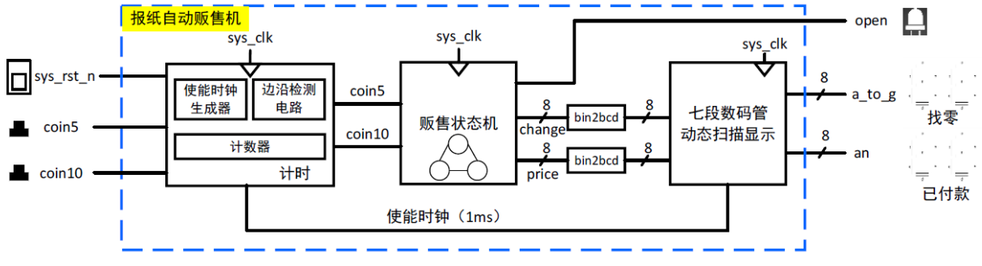
二. 实验内容

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下图所示，输入/输出端口如右表所示。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）1 个 5 分的硬币和 1 个 1 角的硬币，不找零；
（2）3 个 5 分的硬币，不找零；
（3）1 个 1 角的硬币和 1 个 5 分的硬币，不找零；
（4）2 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，同时 4 个数码管也恢复为 0.



报纸自动贩售机顶层模块

天津大学本科生实验报告专用纸

报纸自动贩售机由 4 部分构成。

第一部分是计时器模块，该模块又由 3 个子模块构成，分别是计数器电路、使能时钟生成电路和边沿检测电路。

第二部分是整个自动贩售机电路的核心——贩售机状态机。状态机根据投币情况产生“已付款”和“找零”输出。此外，如果已付款超过 15 分，则将 LED 灯点亮，表示出售成功。

第三部分是两个 8 位二进制转 BCD 模块，分别将二进制的“已付款”和“找零”值转化为 BCD 编码，即 10 进制数。本实验中，该模块不需要实现，由教师直接提供 IP 使用。

第四部分是 7 段数码管动态扫描显示模块，它实现“已付款”和“找零”值得最终显示。

输入/输出端口

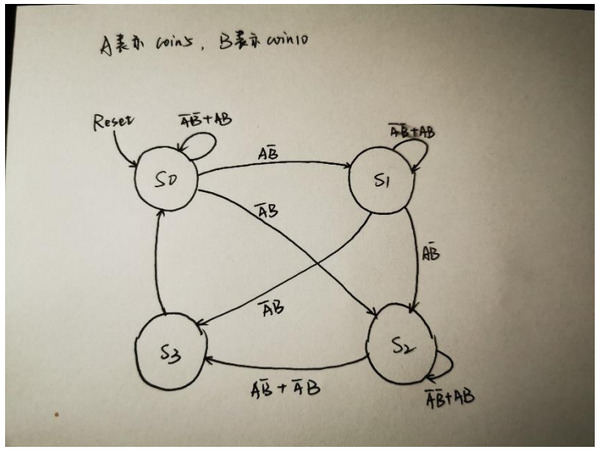
端口名	方向	宽度（位）	作用
sys_clk	输入	1	系统基准时钟，主频 25MHz。
sys_rst_n	输入	1	连接拨动开关，对数字中进行复位。低电平有效。
coin5	输入	1	连接按键，每按下一次，表示投入 1 个 5 分硬币，按键按下为高电平。
coin10	输入	1	连接按键，每按下一次，表示投入 1 个 1 角硬币，按键按下为高电平。
open	输出	1	连接 LED 灯，灯亮表示购买成功；灯灭表示支付费用不够，购买失败。购买成功后，灯持续亮 10s，然后熄灭。
a_to_g	输出	8	连接七段数码管的数据输入端 CA~CG 和 DP，用于实时显示“已支付费用”和“找零”。采用共阳极控制，数码管低电平点亮。DP 段永远不点亮。
an	输出	4	连接 4 个七段数码管的使能端 AN0~AN3，其中 AN0 和 AN1 显示“已支付费用”，AN2 和 AN3 显示“找零”。使能信号高电平有效。

完成上述分秒数字钟的设计，需要有以下几点需要注意：

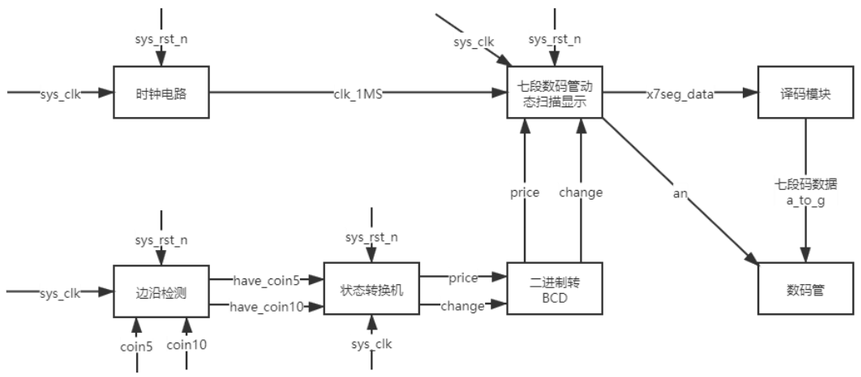
1、7 段数码管动态扫描必须采用使能时钟实现，扫描频率为 1KHz（1ms）。
2、必须通过边沿检测电路识别“5 分”和“1 角”按键按下产生的上升沿，以用于后续处理。
3、用于计时的时钟频率为 25MHz（40ns）。
4、由于 7 段数码管的扫描周期是 1ms，购买成功后需要等待 10s，从而造成仿真时间过长。为了加快仿真速度，可以在仿真的时候使用较大的计时单位和扫描速度。

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。

（1）vend.sv

```
module vend(
        input sys_clk, sys_rst_n,
        input coin5, coin10,
        output [3 : 0] an,
        output [7 : 0] a_to_g,
        output open
        );
        logic [7:0]price,change,bcd_0,bcd_1;
        logic clk_1MS;
        logic [3:0]x7seg_data;
        logic have_coin5,have_coin10;

        clken U0(
                .sys_clk        (sys_clk),
                .sys_rst_n      (sys_rst_n),
                .clk_flag       (clk_1MS)
        );
        edge_detect U1(
                .sys_clk        (sys_clk),
                .sys_rst_n      (sys_rst_n),
                .coin5          (coin5),
                .coin10         (coin10),
                .have_coin5 (have_coin5),
                .have_coin10(have_coin10)
    );
    state U2(
                .sys_clk        (sys_clk),
                .sys_rst_n      (sys_rst_n),
                .have_coin5 (have_coin5),
                .have_coin10(have_coin10),
                .price          (price),
                .change         (change),
                .open           (open)
    );
    bin2bcd_0 U3(
                .bin            (price),
                .bcd            (bcd_0)
    );
    bin2bcd_0 U4(
                .bin            (change),
                .bcd            (bcd_1)
    );
```

```
    x7seg_scan U5(
        .sys_clk       (sys_clk),
        .sys_rst_n     (sys_rst_n),
        .clk_flag      (clk_1MS),
        .price         (bcd_0),
        .change        (bcd_1),
        .x7seg_data (x7seg_data),
        .an            (an)
    );
    x7seg_dec U6(
        .D             (x7seg_data),
        .a_to_g        (a_to_g)
    );

endmodule
```

（2）clken.sv

```
module clken #(parameter SYS_CLK_FREQ = 25_000_000, TARGET_CLK_FREQ = 1000, N =
15)(
    input               sys_clk,
    input               sys_rst_n,
    output logic       clk_flag
    );

    localparam CNT_MAX = SYS_CLK_FREQ/TARGET_CLK_FREQ;
    logic [N-1:0]r_cnt;

    always_ff @(posedge sys_clk) begin
        if(!sys_rst_n) r_cnt <= 0;
        else if(r_cnt == CNT_MAX-1) r_cnt <= 0;
        else r_cnt <= r_cnt+1;
    end
    always_ff @(posedge sys_clk) begin
        if(!sys_rst_n) clk_flag = 1'b0;
        else if(r_cnt == CNT_MAX-1) clk_flag <= 1'b1;
        else clk_flag <= 1'b0;
    end

endmodule
```

（3）edge_detect.sv

```
module edge_detect(
    input               sys_clk,
    input               sys_rst_n,
    input               coin5,
    input               coin10,
    output logic       have_coin5,
    output logic       have_coin10
    );

    logic [1:0] coin5_start;
    logic [1:0] coin10_start;

    always_ff @(posedge sys_clk) begin
        if(sys_rst_n == 0) coin5_start <= 2'b00;
        else coin5_start <= {coin5_start[0], coin5};
    end
    always_ff @(posedge sys_clk) begin
        if(sys_rst_n == 0) coin10_start <= 2'b00;
        else coin10_start <= {coin10_start[0], coin10};
    end

    assign have_coin5 = (coin5_start[0] & (~coin5_start[1]));
    assign have_coin10 = (coin10_start[0] & (~coin10_start[1]));

endmodule
```

（4）state.sv

```
module state(
    input               sys_clk,
    input               sys_rst_n,
    input               have_coin5,
    input               have_coin10,
    output logic       [7:0]price,
    output logic       [7:0]change,
    output logic       open
    );

    logic [30:0]cnt;
    logic start_flag;
    logic [1:0]S0,S1,S2,S3;
    logic [1:0]current_state,next_state;
    logic [7:0]cnt5,cnt10;
```

```
    logic Is20;
    logic flag;

    assign S0=0;
    assign S1=1;
    assign S2=2;
    assign S3=3;

    always_ff @(posedge sys_clk) begin
        if(!sys_rst_n) begin
            cnt <= 0;
            flag <= 0;
        end
        else if(next_state == S3) begin
            if(cnt == 249_999_990) begin
                flag <= 1;
                cnt <= 0;
            end
        end
        else flag <= 0;
    end
    always_ff @(posedge sys_clk) begin
        if(!sys_rst_n) Is20 <= 0;
        else if(current_state == S2 && have_coin10) Is20 <= 1;
        else if(next_state == S0) Is20 <= 0;
        else Is20 <= Is20;
    end
    always_ff @(posedge sys_clk) begin
        if(!sys_rst_n) current_state <= S0;
        else current_state <= next_state;
    end

    always_comb begin
        case(current_state)
            S0: begin
                if(have_coin5) next_state = S1;
                else if(have_coin10) next_state = S2;
                else next_state = S0;
            end
            S1: begin
                if(have_coin5) next_state = S2;
                else if(have_coin10) next_state = S3;
                else next_state = S1;
            end
            S2: begin
                if(have_coin5) next_state = S3;
                else if(have_coin10) next_state = S3;
                else next_state = S2;
            end
            S3: begin
                if(flag) next_state = S0;
                else next_state = S3;
            end
        endcase
    end

    always_ff @(posedge sys_clk) begin
        case(next_state)
            S0: begin
                open <= 0;
                change <= 0;
                price <= 0;
            end
            S1: begin
                open <= 0;
                change <= 0;
                price <= 5;
            end
            S2: begin
                open <= 0;
                change <= 0;
                price <= 10;
            end
            S3: begin
                open <= 1;
                if(Is20) begin
                    change <= 5;
                    price <= 20;
                end
                else begin
                    change <= 0;
                    price <= 15;
                end
            end
        endcase
    end

endmodule
```

（5）x7seg_scan.sv

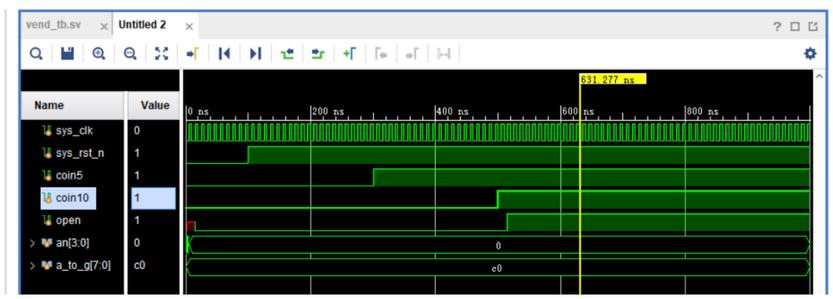
```
module x7seg_scan(
    input               sys_clk,
    input               sys_rst_n,
    input               clk_flag,
    input logic       [7:0]change,
    input logic       [7:0]price,
    output logic      [3:0]x7seg_data,
    output logic      [3:0]an
    );

    logic [3:0]cnt;

    always_ff @(posedge sys_clk) begin
        if(!sys_rst_n) cnt <= 0;
        else if(clk_flag) begin
            if(cnt == 4) cnt <= 1;
            else cnt <= cnt+1;
        end
    end
    always_ff @(posedge sys_clk) begin
        if(!sys_rst_n) x7seg_data <= 4'b0000;
        else if(cnt == 1) x7seg_data <= price[3:0];
        else if(cnt == 2) x7seg_data <= price[7:4];
        else if(cnt == 3) x7seg_data <= change[3:0];
        else if(cnt == 4) x7seg_data <= change[7:4];
    end
    always_ff @(posedge sys_clk) begin
        if(!sys_rst_n) an <= 4'b0000;
        else if(cnt == 1) an <= 4'b0001;
        else if(cnt == 2) an <= 4'b0010;
        else if(cnt == 3) an <= 4'b0100;
        else if(cnt == 4) an <= 4'b1000;
    end

endmodule
```

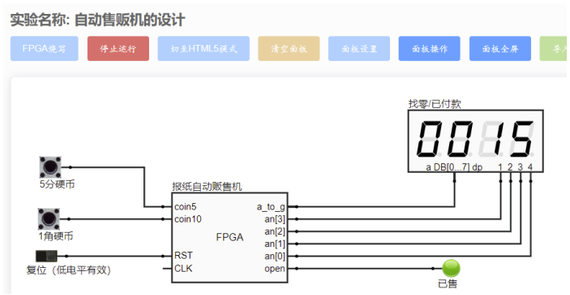
（6）x7seg_dec.sv

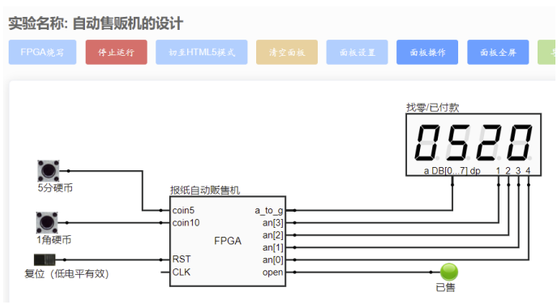
```
module x7seg_dec(
    input           [3:0]D,
    output logic[7:0]a_to_g
    );

    always_comb begin
        case(D)
            0 : a_to_g = 8'b11000000;
            1 : a_to_g = 8'b11111001;
            2 : a_to_g = 8'b10100100;
            3 : a_to_g = 8'b10110000;
            4 : a_to_g = 8'b10011001;
            5 : a_to_g = 8'b10010010;
            6 : a_to_g = 8'b10000010;
            7 : a_to_g = 8'b11111000;
            8 : a_to_g = 8'b10000000;
            9 : a_to_g = 8'b10010000;
            default a_to_g =8'b11000000;
        endcase
    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

六．附加题（若实验指导书无要求，则无需回答）

教师签字：

年　月　日