《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	5.15

实验项目名称 分秒数字钟的设计与实现

一. 实验目的

1. 掌握基于 SystemVerilog HDL 的时序逻辑电路建模方法;
2. 掌握计数器设计方法，并能够使用计数器设计使能时钟（用于时钟分频);
3. 掌握移位寄存器设计方法，并能够利用移位寄存器设计边沿检测电路;
4. 掌握 7 段数码管的动态显示。

二. 实验内容

基于 SystemVerilog HDL 设计并实现一个分秒数字钟。整个工程的顶层模块 如图 3-6 所示，输入/输出端口如表 3-1 所示。使用 4 个七段数码管显示当前的 计时。其中，两个数码管对应“分”，另两个数码管对应“秒”。通过 1 个拨动开关 对数字钟进行复位控制。使用 1 个按键对数字中进行“暂停/计时”控制，按键每按 下一次，进行暂停和计时的切换，即暂停时，按下按键启动计时；计时过程中， 按下按键暂停计时。

分秒数字钟由 3 部分构成。

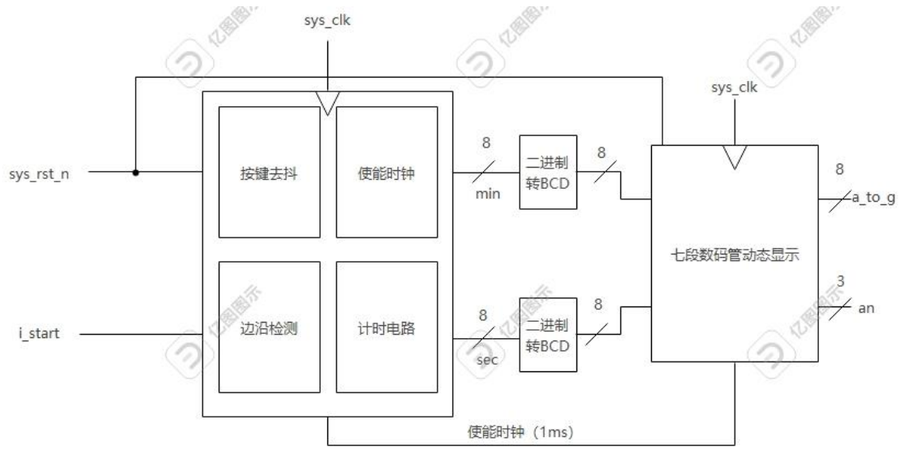
● 第一部分是数字钟的核心——计时器模块，该模块又由 3 个子模块构成，分别是计时电路、使能时钟生成电路和边沿检测电路。 ″ 计时电路通过计数器实现计时功能，产生二进制的“分”（min）和“秒” （sec）输出。 ″ 使能时钟生成电路用于产生控制七段数码管动态显示的使能时钟，使能时钟高电平出现的周期为 1ms。 ″ 边沿检测电路模块对按键输入进行上升沿检测，产生控制计时器暂 停和启动的信号。

● 第二部分是 8 位二进制转 BCD 模块，它将二进制的“分”、“秒”值转化为 BCD 编码，即 10 进制数。其中，分、秒各使用一个。本实验中，该模 块不需要实现，由教师直接提供 IP 使用。

● 第三部分是 7 段数码管动态扫描显示模块，它实现“分”、“秒”值的最终 显示。“分”、“秒”值各使用两个 7 段数码管进行显示。

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. 画出分秒数字钟电路的原理图（模块级别即可，如使能时钟模块、边沿检测模块等）



2. 分秒数字钟电路中一共使用了几个计数器，作用分别是什么？

一共用了三个计数器

第一个在计时电路中，用于计时

第二个在按键去抖电路中，用于时钟分频

第三个在使能时钟中，用于使能低频的时钟

3. 给出分秒数字钟的 SystemVerilog 代码。

顶层模块

```
`timescale 1ns / 1ps


module dig_clock(
    input                       sys_clk,
    input                       sys_rst_n,
    input                       i_start,
    output logic [3 : 0]        an,
    output logic [7 : 0]        a_to_g
    );


    logic press1, press2;
    logic clkflag;
    logic [7:0] min, sec, mind, secd;

    debounce debouncemux(.sys_clk(sys_clk), .sys_rst_n(sys_rst_n), .i_btn(i_start), .o_btn(press1));
    edge_detect
edge_detectmux(.sys_clk(sys_clk),    .sys_rst_n(sys_rst_n), .i_btn(press1), .posedge_flag(press2));
    timer timermux (.clk(sys_clk), .rst_n(sys_rst_n), .i_start(press2), .min(min), .sec(sec));
    clken clkenmux(.clk(sys_clk), .rst_n(sys_rst_n), .clk_flag(clkflag));
    bin2bcd_0 minbin (.bin(min), .bcd(mind));
    bin2bcd_0 secbin (.bin(sec), .bcd(secd));
    seven_seg_disp
ssdmux(.sys_clk(sys_clk), .clk(clkflag), .rst_n(sys_rst_n), .num0(mind[7:4]), .num1(mind[3:0]), .num2(s
ecd[7:4]), .num3(secd[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
```

计时电路

```
module timer
(
  input logic clk,
  input logic rst_n,
  input logic i_start,

  output logic [7:0] min,
  output logic [7:0] sec
);

    logic [25:0] count;
    logic stop;


    always_ff @(posedge clk) begin
    always_ff @(posedge clk) begin
        if (~rst_n) begin
            count <= 26'd0;
            min <= 6'd0;
            sec <= 6'd0;
            stop <= 1'd1;
        end

        if(i_start) begin
        stop <= ~stop;
        end
```

```
if(rst_n && ~stop) begin
    if(count == 26'd24_999_999) begin
        sec <= sec + 6'd1;
        count <= 26'd0;
    end
    else
        count <= count + 26'd1;

    if(sec == 6'd59) begin
        min <= min + 6'd1;
        sec <= 6'd0;
    end
end
end


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
```

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
```

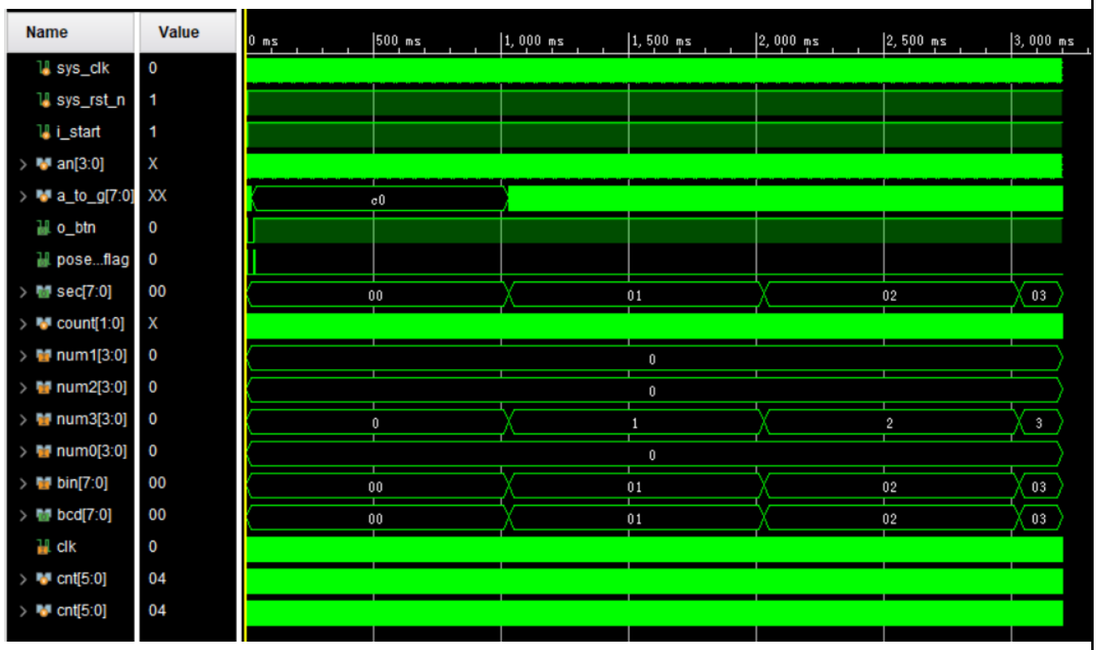
```
if(count == 0) begin
    an <= 4'b1000;
    case (num0)
        4'd0: a_to_g <= enc_0;
        4'd1: a_to_g <= enc_1;
        4'd2: a_to_g <= enc_2;
        4'd3: a_to_g <= enc_3;
        4'd4: a_to_g <= enc_4;
        4'd5: a_to_g <= enc_5;
        4'd6: a_to_g <= enc_6;
        4'd7: a_to_g <= 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;
```

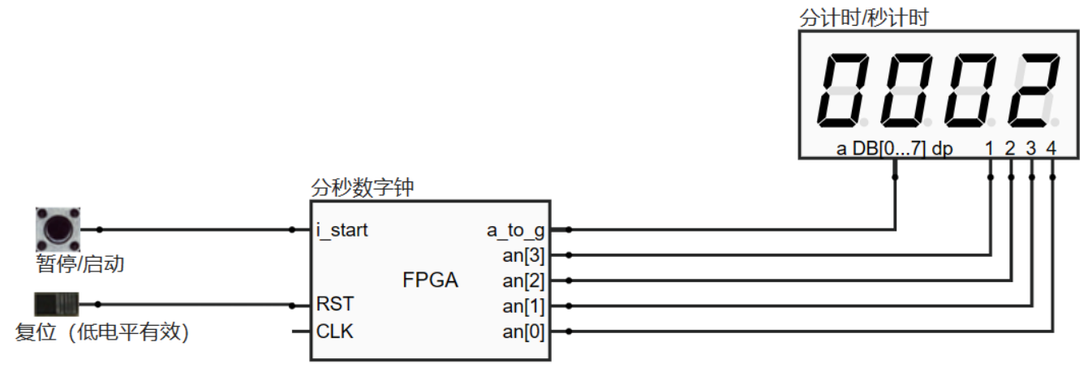
```
            4'd9: a_to_g <= enc_9;
        endcase
end
if(count == 3) begin
an <= 4'b0001;
        case (num3)
            4'd0: a_to_g <= enc_0;
            4'd1: a_to_g <= enc_1;
            4'd2: a_to_g <= enc_2;
            4'd3: a_to_g <= enc_3;
            4'd4: a_to_g <= enc_4;
            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 1：动态显示七段数码管 无法在复位时初始化

解决：利用 sys_clk 增加一个中间量，用于其中 an 的初始化

问题 2：仿真时间过长

解决：降低仿真时 sys_clk 的频率，加快仿真速度

教师签字：

年　月　日