《数字逻辑与数字系统》实验报告

学院 智算 年级 2019 级 班级 软工一班 姓名 俞林昊 学号 3019207450

课程名称 数字逻辑和数字系统 实验日期 2021.5.20 成绩

同组实验者

实验项目名称 分秒数字钟的设计和实现

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。

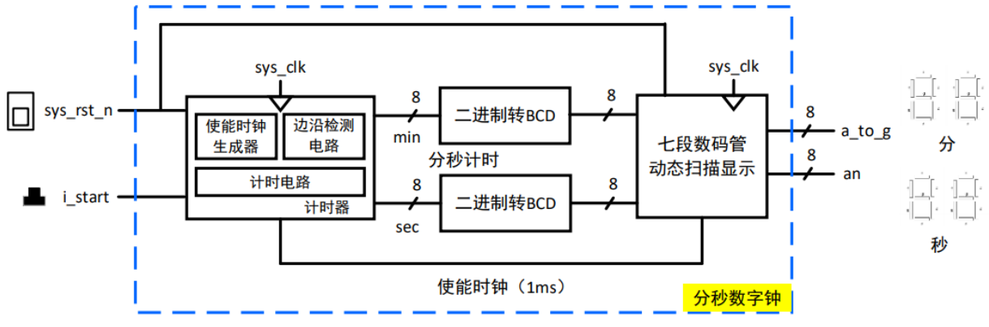


图 3-6 分秒数字钟顶层模块

天津大学本科生实验报告专用纸

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。

表 3-1 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。复位信号低电平有效。
i_start	输入	1	连接按键，用于控制暂停/计时。每按下按键进行暂停和计时切换。按键按下为高电平。

a_to_g	输出	8	连接七段数码管的数据输入端 CA~CG 和 DP，用于显示秒表的分和秒。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 显示秒，AN3 和 AN4 显示分。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：

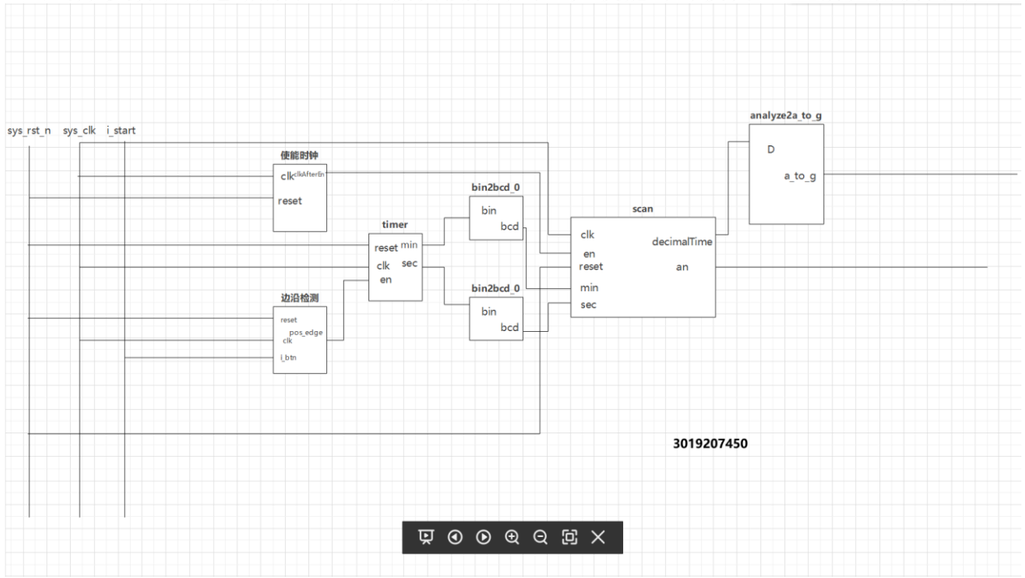
1. 7 段数码管动态扫描必须采用**使能时钟**实现，扫描频率为 **1KHz**（1ms）。
2. 必须通过边沿检测电路识别**“暂停/计时”**按键按下产生的上升沿，以边进行后续处理。
3. 用于计时的时钟频率为 **25MHz**（40ns）。
4. 本实验在 xdc 约束文件中添加**时钟约束**，所对应的语句如下：

```
set_property -dict { PACKAGE_PIN F5    IOSTANDARD LVCMOS33 } [get_ports { sys_clk }];
create_clock -add -name sys_clk_pin -period 40.00 [get_ports {sys_clk}];
```

5. 由于分秒数字钟计时单位为 1 秒，7 段数码管扫描周期是 1ms，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？

一共有三个计数器。在计时电路中的计数器根据系统时钟来输出分秒信号。系统时钟周期为 40ns，每个周期计数器加 1，所以每增加一秒，计数器都要记到 25000000。在使能时钟生成器中的计数器根据系统时钟进行分频操作。系统钟周期为 40ns，输出周期为 1ms，所以每有一个上升沿，计数器都要记到 25000。在七段数码管动态扫面显示模块的两位计数器每过 1ms（每个上升沿）加 1，轮流产生四个不同的数，每个数对应一个数码管，代表此时只有该数码管有效。

3. 给出分秒数字钟的 SystemVerilog 代码。

Clk_en

```
module clk_en #(parameter N = 25000) (
        input clk,
        input reset,
        output logic clkAfterEn
        );

        logic [31 : 0] r_cnt;

        always_ff@(posedge clk) begin
                if(!reset) r_cnt <= 0;
```

```
            else if(r_cnt == N-1) r_cnt <= 0;
            else r_cnt <= r_cnt + 1;
        end

        always_ff@(posedge clk) begin
            if(!reset) clkAfterEn <= 1'b0;
            else if(r_cnt == N-1) clkAfterEn <= 1'b1;
            else clkAfterEn <= 1'b0;
        end

endmodule
```

Edge_detection

```
module Edge_detection (
        input clk,
        input reset,
        input i_btn,
        output logic pos_edge
        );

        logic q1;
        logic q2;
        always_ff@(posedge clk)
            if(reset) begin
                q1 <= i_btn;
                q2 <= q1;
            end
            else if(!reset) begin
                q1 <= 0;
                q2 <= 0;
            end

        assign start_flag = (~q2) & q1;
endmodule
```

Timer

```
module timer #(parameter MAX_SEC = 59, MAX_MIN = 59, MAX_COUNT = 25000000) (
        input clk,
        input reset,
        input en,
        output logic [7:0] min,
        output logic [7:0] sec
        );

        logic [31 : 0] r_cnt;
        logic state;

        always_ff@(posedge clk) begin
            if(!reset) state = 1'b0;
            else if(en) state = ~state;
        end

        always_ff@(posedge clk) begin
            // reset
            if(!reset) begin
                min = 8'd0;
                sec = 8'd0;
            end

            // timing
            else if(state) begin
                if (r_cnt < MAX_COUNT - 1) r_cnt <= r_cnt + 1;
                else if(r_cnt == MAX_COUNT - 1) begin
                    r_cnt <= 0;
                    if(sec == MAX_SEC) begin
                        sec <= 0;
                        min <= min + 1;
                    end
                    else if(sec < MAX_SEC) sec <= sec + 1;
                    else if(min == MAX_MIN) min <= 0;
                end
            end

            else if(!state) begin
                min <= min;
                sec <= sec;
```

```
            end
        end

endmodule
```

Scan

```
module scan(
    input clk,
    input reset,
    input en,
    input [7 : 0] min,
    input [7 : 0] sec,
    output logic [3 : 0] decimalTime,
    output logic [3 : 0] an
    );

    always_ff@(posedge clk)
        if(!reset)
          an <= 4'd1;
        else if(reset && en)
          an <= {an[2 : 0], an[3]};


    always@(*) begin
        if(an == 1) decimalTime = sec[3 : 0];
        else if(an == 2)
            decimalTime = sec[7 : 4];
        else if(an == 4)
            decimalTime = min[3 : 0];
        else if(an == 8)
            decimalTime = min[7 : 4];
    end
endmodule
```

analyze2a_to_g

```
module analyze2a_to_g(
    input [3 : 0] D,
    output logic [7 : 0] a_to_g
    );
    always@(*) begin
        unique case(D)
            4'd0: a_to_g = 8'b11000000;
            4'd1: a_to_g = 8'b11111001;
            4'd2: a_to_g = 8'b10100100;
            4'd3: a_to_g = 8'b10110000;
            4'd4: a_to_g = 8'b10011001;
            4'd5: a_to_g = 8'b10010010;
            4'd6: a_to_g = 8'b10000010;
            4'd7: a_to_g = 8'b11111000;
            4'd8: a_to_g = 8'b10000000;
            4'd9: a_to_g = 8'b10010000;
        endcase
    end

endmodule
```

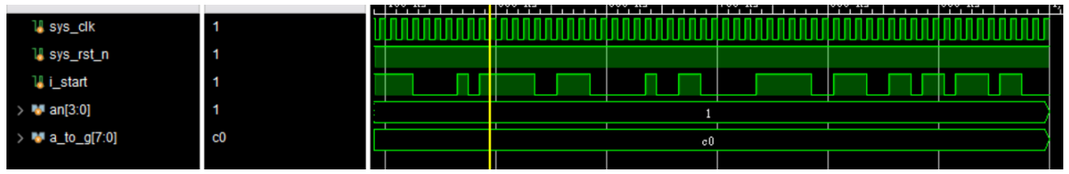
dig_clock

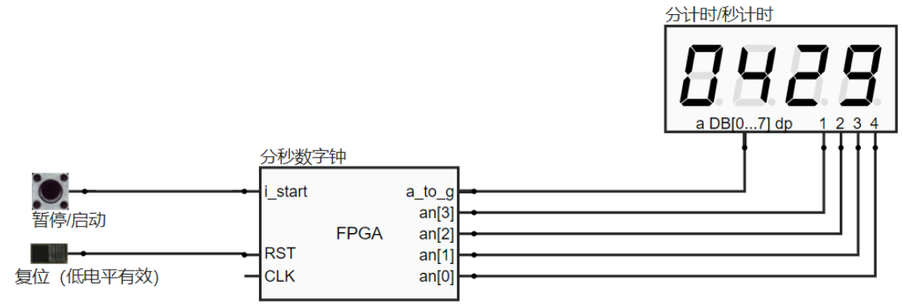
```
`timescale 1ns / 1ps

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 clkAfterEn;
    logic work;
    logic [7 : 0]   min, sec, bcd_min, bcd_sec;
    logic [3 : 0]   decimalTime;

    clk_en U0(
        .clk (sys_clk),
        .reset (sys_rst_n),
```

```
.clkAfterEn (clkAfterEn)
);

Edge_detection U1(
        .clk (sys_clk),
        .reset (sys_rst_n),
        .i_btn (i_start),
        .pos_edge (work)
);

timer U2(
        .clk (sys_clk),
        .reset (sys_rst_n),
        .en (work),
        .min (min),
        .sec (sec)
);
bin2bcd_0 U3(
        .bin (min),
        .bcd (bcd_min)
);

        bin2bcd_0 U4(
                .bin (sec),
                .bcd (bcd_sec)
        );

        scan U5(
                .clk (sys_clk),
                .reset (sys_rst_n),
                .en (clkAfterEn),
                .min (bcd_min),
                .sec (bcd_sec),
                .decimalTime (decimalTime),
                .an (an)
        );

        analyze2a_to_g U6(
                .D (decimalTime),
                .a_to_g (a_to_g)
        );
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

实验中，对于多个模块进行分层设计。在书写代码的时候，不能在定义的时候就把初始值赋给变量。

在设计的时候，在动态扫描之后，又加上了一个将动态扫描的结果作为 analyze2a_to_g 这个模块的输入。这是因为发现 scan 模块输出的是一个十进制数字，要将其转化为一个能够点亮七段数码管的二进制数字。

具体的实验模块之间的逻辑如原理图所示。先是使能时钟将系统的时间分频，边沿检测电路来检测出时钟的上升沿，之后这两个信号作为计时器模块的输入。计时器模块的输出作为两个 bin_bcd 模块的输入，输出十进制的时间。

教师签字：

年　月　日