《数字逻辑与数字系统》实验报告

学院 智能与计算学部 年级 2019 级 班级 4 班 姓名 董跃 学号 3019205149

课程名称 数字逻辑与数字系统 实验日期 2021.5.20 成绩_____

同组实验者 无

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。

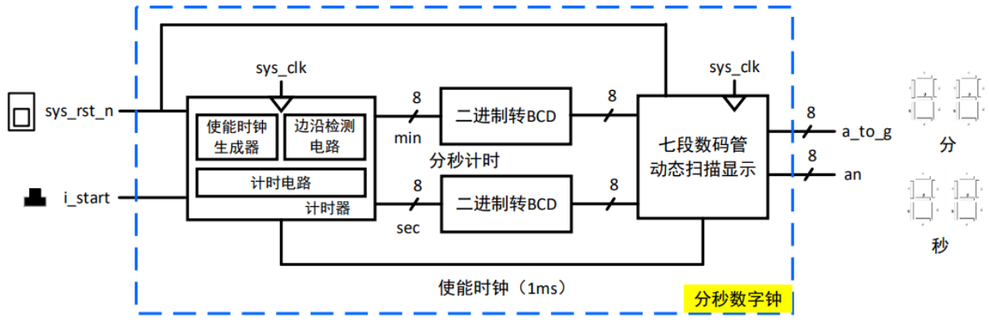


图 3-6 分秒数字钟顶层模块

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：

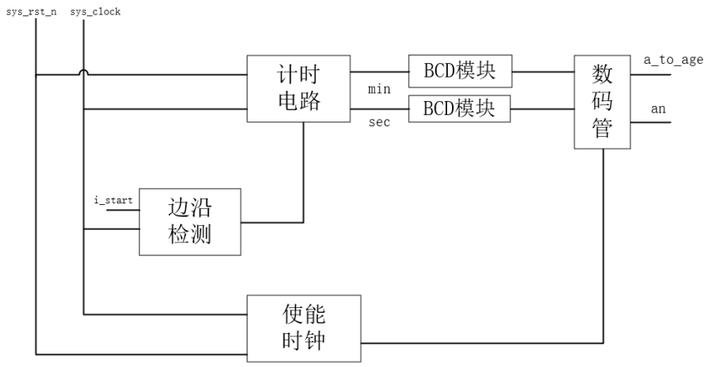
1. 7 段数码管动态扫描必须采用**使能时钟**实现，扫描频率为 **1KHz** (1ms)。
2. 必须通过边沿检测电路识别**"暂停/计时"**按键按下产生的上升沿，以边进行后续处理。
3. 用于计时的时钟频率为 **25MHz** (40ns)。
4. 本实验在 xdc 约束文件中添加**时钟约束**，所对应的语句如下：

```
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，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？

答：时钟一共用了3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

(1)计时电路

```
module clk_timeCircuit(
    input logic sys_clock,
    input logic sys_reset_n,
    input logic i_start,
    output logic [7:0]sec,
    output logic [7:0]min,
    output logic [31:0] count=0
    );
    logic[7:0]tempsec=0,tempmin=0;
    logic flag=0;
    always_ff@(posedge sys_clock)begin
            if(i_start)
                    flag<=~flag;
            if(!sys_reset_n) begin
                count<=0;
                tempsec<=0;
                tempmin<=0;
            end
            else if(!flag&&count==32'd24999999)begin
                    count<=32'd0;
                    if(tempsec==8'd59)begin
                        tempsec<=8'd0;
                        if(tempmin==8'd59)
                            tempmin<=8'd0;
                        else
                            tempmin<=tempmin+1;
                    end
                  else tempsec<=tempsec+1;
            end
            else if(!flag) count<=count+1;
    end
            assign min=tempmin;
            assign sec=tempsec;
endmodule
```

(2)使能时钟

```
module Enable_clock(
    input logic sys_clock,
    input logic sys_reset_n,
    output logic clk_flag
    );
    logic [15:0] count=0;
    always_ff@(posedge sys_clock)begin
        if(!sys_reset_n) begin
            clk_flag<=0;count<=0;
        end
        else
            if(count==16'd24999)begin
                count<=16'd0;
                clk_flag<=1;
            end
            else begin
                count<=count+1;
                clk_flag<=0;
            end
    end
endmodule
```

(3)边沿检测

```
module posedge_check(
    input logic i_start,
    input logic sys_clk,
    output logic pos_edge
    );
    logic t1;
    logic t2;
    always_ff@(posedge sys_clk)begin
    t1 <= i_start;
    t2 <= t1;
    end
    assign pos_edge=t1&(~t2);
endmodule
```

(4)七段数码管动态扫描显示

```
module dig_7seg(
    input logic [7:0] sec,
    input logic [7:0] min,
    input logic sys_clock,
    output logic [7:0] a_to_g,
    output logic[3:0] an
    );
    logic [1:0] count=2'b00;
    always_ff@(posedge sys_clock)begin
        if(count==2'b00)begin
            case (sec[3:0])
                4'b0000:a_to_g<=~8'h3f;
                4'b0001:a_to_g<=~8'h06;
                4'b0010:a_to_g<=~8'h5b;
                4'b0011:a_to_g<=~8'h4f;
                4'b0100:a_to_g<=~8'h66;
                4'b0101:a_to_g<=~8'h6d;
                4'b0110:a_to_g<=~8'h7d;
                4'b0111:a_to_g<=~8'h07;
                4'b1000:a_to_g<=~8'h7f;
                4'b1001:a_to_g<=~8'h6f;
            endcase
            count<=count+1;
            an<=4'b0001;
        end
        else if(count==2'b01)begin
            case (sec[7:4])
                4'b0000:a_to_g<=~8'h3f;
                4'b0001:a_to_g<=~8'h06;
                4'b0010:a_to_g<=~8'h5b;
                4'b0011:a_to_g<=~8'h4f;
                4'b0100:a_to_g<=~8'h66;
                4'b0101:a_to_g<=~8'h6d;
                4'b0110:a_to_g<=~8'h7d;
                4'b0111:a_to_g<=~8'h07;
                4'b1000:a_to_g<=~8'h7f;
                4'b1001:a_to_g<=~8'h6f;
            endcase
```

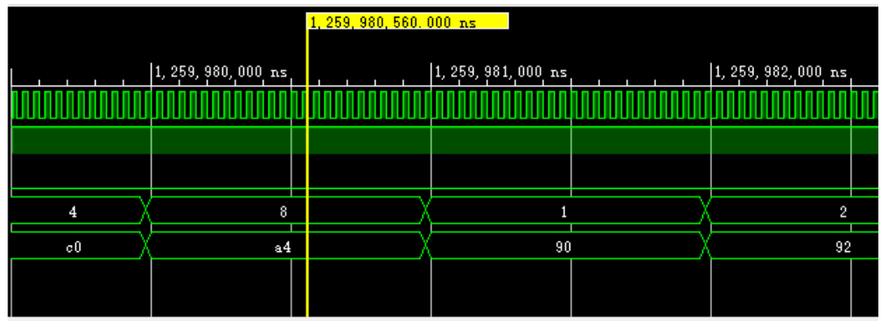
```
            count<=count+1;
            an<=4'b0010;
        end
        else if(count==2'b10)begin
            case (min[3:0])
                4'b0000:a_to_g<=~8'h3f;
                4'b0001:a_to_g<=~8'h06;
                4'b0010:a_to_g<=~8'h5b;
                4'b0011:a_to_g<=~8'h4f;
                4'b0100:a_to_g<=~8'h66;
                4'b0101:a_to_g<=~8'h6d;
                4'b0110:a_to_g<=~8'h7d;
                4'b0111:a_to_g<=~8'h07;
                4'b1000:a_to_g<=~8'h7f;
                4'b1001:a_to_g<=~8'h6f;
            endcase
            count<=count+1;
            an<=4'b0100;
        end
        else begin
            case (min[7:4])
                4'b0000:a_to_g<=~8'h3f;
                4'b0001:a_to_g<=~8'h06;
                4'b0010:a_to_g<=~8'h5b;
                4'b0011:a_to_g<=~8'h4f;
                4'b0100:a_to_g<=~8'h66;
                4'b0101:a_to_g<=~8'h6d;
                4'b0110:a_to_g<=~8'h7d;
                4'b0111:a_to_g<=~8'h07;
                4'b1000:a_to_g<=~8'h7f;
                4'b1001:a_to_g<=~8'h6f;
            endcase
            count<=2'b00;
            an<=4'b1000;
        end
    end
endmodule
```

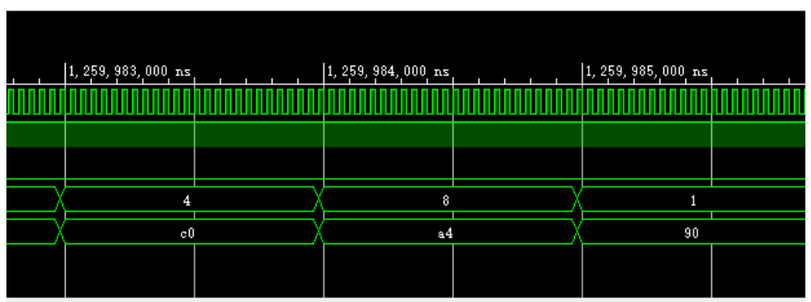
(5)测试

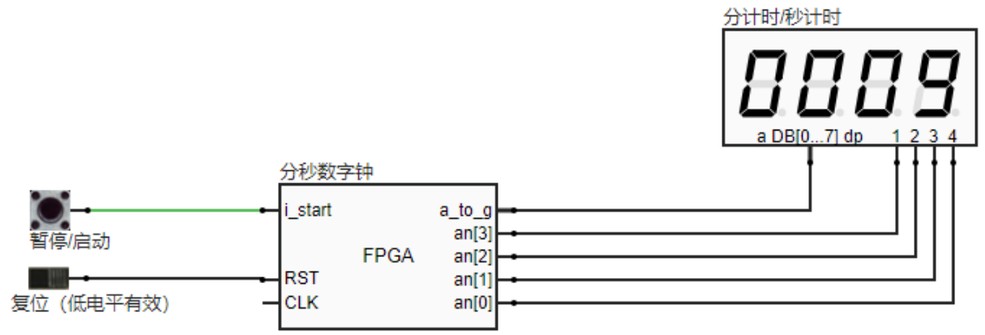
```
module dig_clock_tb(

    );
    logic sys_clk,sys_rst_n,i_start;
        logic[3:0]an;
        logic[7:0]a_to_g;
        dig_clock module1(.sys_clk(sys_clk),.sys_rst_n(sys_rst_n),
                          .i_start(i_start),.an(an),.a_to_g(a_to_g));
        initial begin
              sys_clk = 0;
              sys_rst_n = 1;
              forever #20 sys_clk = ~sys_clk;
        end
        initial begin
          i_start = 0;
        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

在测试时，因为用 assign 对变量赋初值 0，导致无法通过，错误是变量可能被同时赋予多个值，通过修改之后，测试能够正常通过。

教师签字：

年　月　日