多功能电子表的 FPGA 实现

61821326 李睿刚（报告撰写）
61821214 龙艺文

目的

- 掌握数字系统的 FPGA 实现，熟练使用 Verilog、Vivado、FPGA 等工具；
- 在实践中总结 FPGA 编程特点，积累硬件编程思路框架。

任务

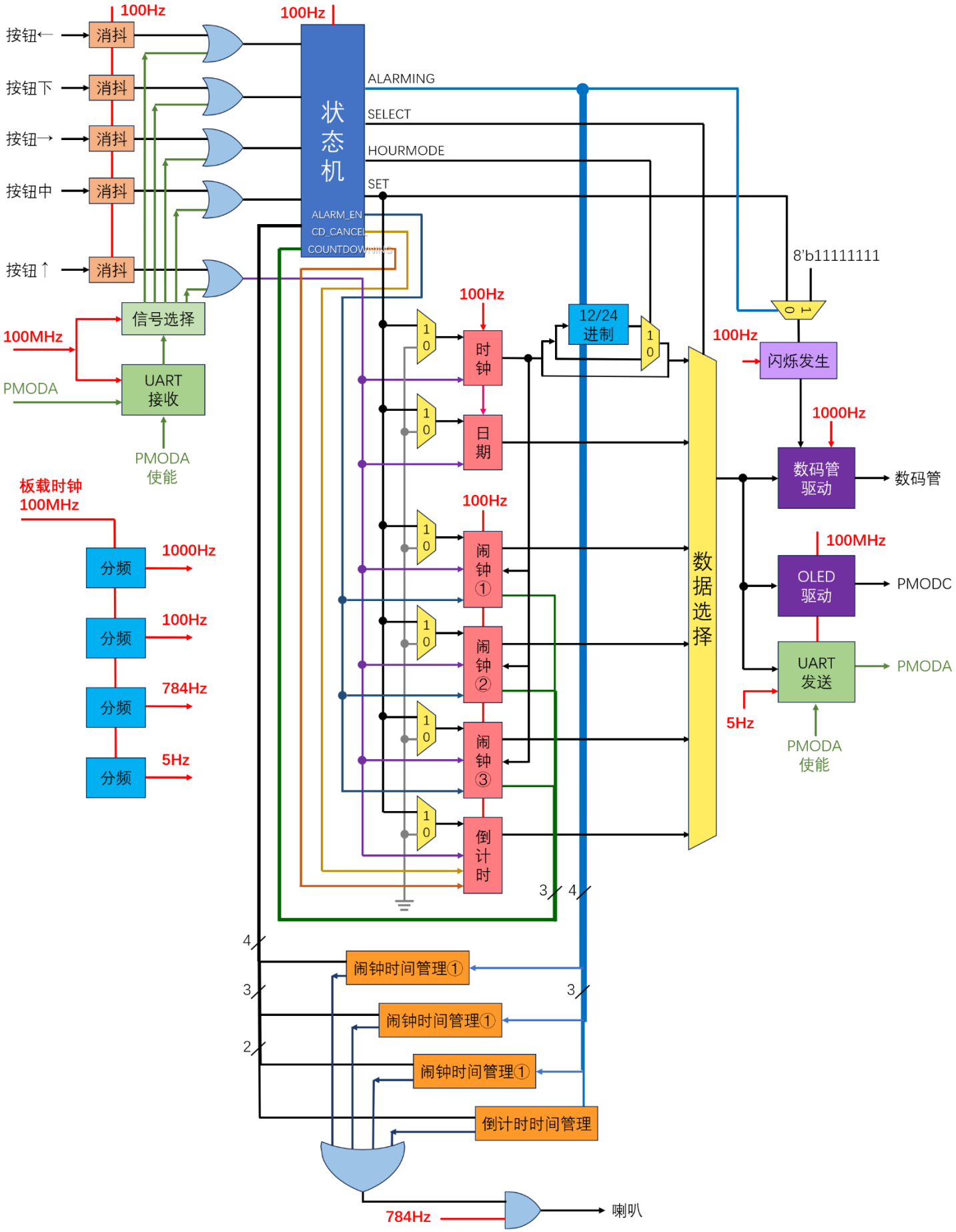
设计 FPGA 模块模拟多功能电子表的工作过程，具有多种功能，功能如下：

1．时间显示界面，要求从 00：00 点计到 23：59。
2．日期显示界面，要求显示当前日期，包含年、月、日。
3．调整时间界面，即可以设置或更改当前的时间（小时、分）。
4．日期设置界面。可以设置当前的日期，比如 2020 年 09 月 22 日。要求支持闰年与大小月的识别。
5．闹钟设置界面，可以设置 3 个闹钟，闹钟时间到了后会用 LED 闪烁提醒，提醒时间持续 5 秒，如果提醒时按解除键，则该闹钟解除提醒，如果闹钟响时没有按键或按其他按键，则响完 5 秒之后暂停，然后 10 秒钟后重新提醒一次后解除。
6．倒计时功能。可以设定倒计时的起始时间，比如 1 分钟，然后开始倒计时，从 01:00 倒计时到 00:00，然后 LED 灯闪烁 5 秒钟。倒计时中间可以暂停或重新开始。
7．电子表只有六个按键。请只使用六个按键来完成所有功能。

原理图

特色功能：小时 12/24 进制切换、闹钟启用控制、用声音提醒、UI 提示、OLED 显示、手机 APP 控制。

抽象后的系统框图如下，省略了部分细节（因此与实际 VIVAD 综合的 RTL 并不一致），只保留了设计的基本思路。其中，时钟用红线表示。

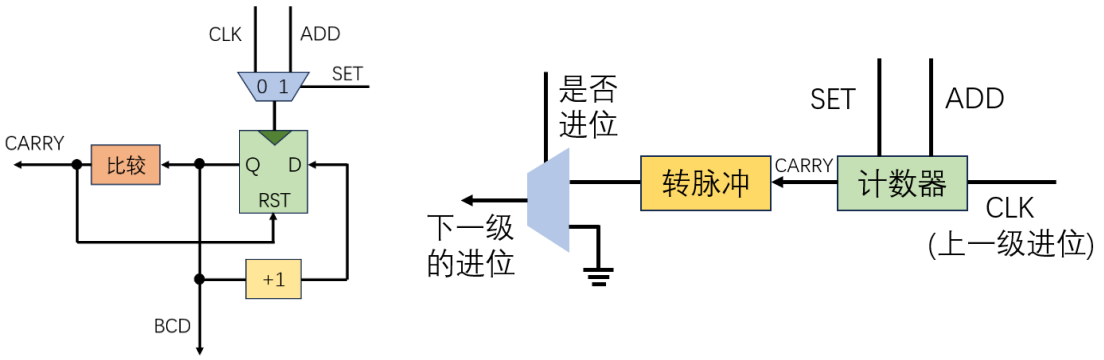

100Hz
100Hz
按钮←
消抖
按钮下
消抖
按钮→
消抖
按钮中
消抖
按钮↑
消抖
状态机
ALARMING
SELECT
HOURMODE
SET
ALARM_EN
CD_CANCEL
COUNTDOWNING
8'b11111111
信号选择
100MHz
PMODA
UART
接收
PMODA
使能
100Hz
时钟
日期
闹钟①
闹钟②
闹钟③
倒计时
12/24
进制
100Hz
闪烁发生
1000Hz
数码管
驱动
数码管
板载时钟
100MHz
分频
1000Hz
分频
100Hz
分频
784Hz
分频
5Hz
数据选择
100MHz
OLED
驱动
PMODC
UART
发送
PMODA
5Hz
PMODA
使能
闹钟时间管理①
闹钟时间管理①
闹钟时间管理①
倒计时时间管理
784Hz
喇叭

设计难点：

1. 计数器

计数器的设计决定了整个项目的结构。由于设计较早，因此事后觉得此设计拖累了项目。此处暂且抛开其优劣不谈。

设计时将一个数字分为了十位和个位两个计数器，以实现模块化。由上一级的计数器的进位充当下一位的时钟位。

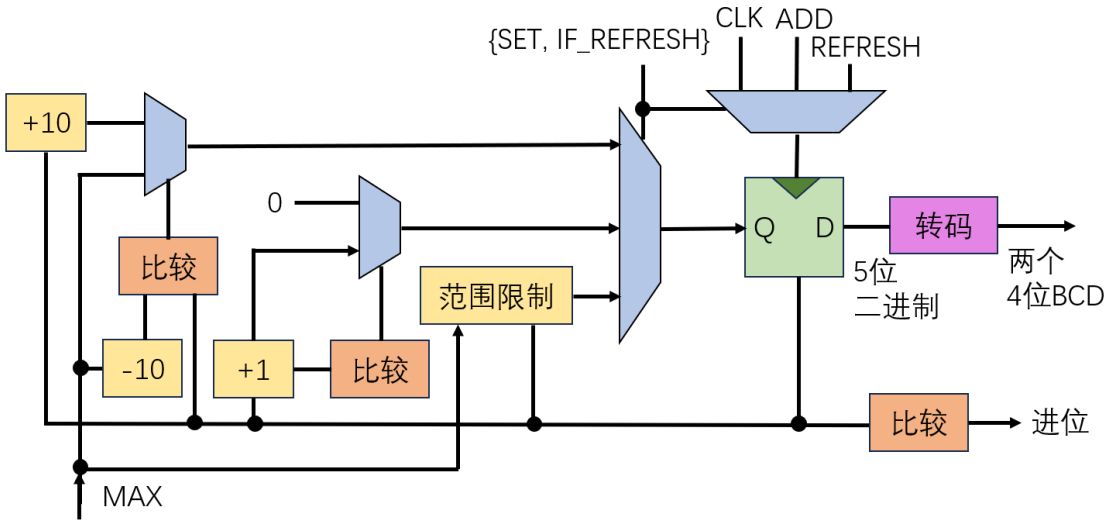


左：单个计数器　　右：计数器连接关系

a) 最值与清零

小时是满 24 清零，日期的最大值和月份、年份相关，月是满 12 清零。即个位计数器清零的清零条件会变。

解决方法：小时使用了特殊的计数器：传入一个 MAX 值和刷新时钟、刷新使能信号。在刷新使能位为 1 时，在刷新时钟的边沿将计数器的值按照 MAX 更新为范围内的值。而对于日期、月份，个位不仅有变化的最大值，还有最小值，比如日期从 01 号开始而不是 00 号。再用之前的方法会很复杂。于是将两位并成一个模块，加 10 或加 1，并判断范围。保留了刷新时钟和刷新使能，因为基本原理还是进位充当时钟。



“月”“日”使用的计数器设计（和实际有细微出入）

b) 脉冲化

根据画出的原理图，这样的设计(计数器的 CLK 由上一个计数器进位的上升沿充当)有一个缺陷：计数器的进位脉冲宽度和驱动其的 CLK 周期相同。如果该计数器表示小时，那么它的进位将 1 小时更新一次，即其进位的高电平将保持一个小时。

如果只是计时，这样可以满足要求。但是需要自己设置时间，意味着寄存器的更新还要能通过按钮的脉冲(称为 ADD)控制。由原理图知，实际更新寄存器信号是用复用器选择的 CLK 和

ADD 之一。问题来了，如果在 CLK 长达 1 小时的高电平期间，切换到 ADD 信号，再切换回去，就会多出一个上升沿，导致计数器无辜+1。因为由于 ADD 是按钮脉冲，故大多时候为低电平。如果在 ADD 为低电平的时候离开设置模式，即切换回长达 1 小时高电平的 CLK 信号，就会产生上升沿。因此，需要设计一个上升沿转脉冲的模块。“仿真结果”版块中可看到此模块的仿真。

关于长脉冲转短脉冲的实现写了篇文章：

https://blog.csdn.net/madderscientist/article/details/132613497

2. 报警

指来自闹 3 个闹钟和 1 个倒计时的提醒信号。

a) 并发

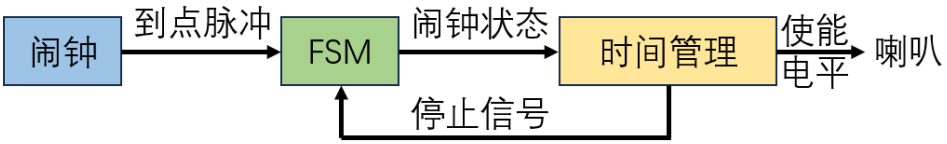
由于我们的闹钟精确到了毫秒，较之到仅分钟的闹钟，我们需要额外考虑多闹钟并发的情况。一个处理方法是，若多个闹钟/倒计时同时报警，则一次取消所有报警。但考虑到实际使用，最终选择四个报警源独立，即若同时报警，则需要每个报警解除一次。为了实现这个效果，考虑到一次只能显示一个界面，三个闹钟和倒计时有了优先级，并各自有一个标志位，同时报警模块分别例化而不是共用。

同时需要状态机的配合，根据报警优先级切换响应界面，当优先级高的报警被解除后，跳至正在报警的下一优先级界面，等待结束或解除。关于状态机的构造详见下一板块。

b) 取消

一段时间后自动停止：设计了一个负反馈回路。当闹钟响了/倒计时到了，会发出一个短脉冲，状态机进入报警状态，其输出的报警电平启动时间管理模块的计时，计时到后向状态机发送停止信号，使报警电平归零。而题目要求的比如“响 5 秒停 10 秒再响 5 秒”由时间管理模块输出的高低电平表示，再去控制喇叭。

而“按键取消”则在状态机中实现。



3. 闰年判断

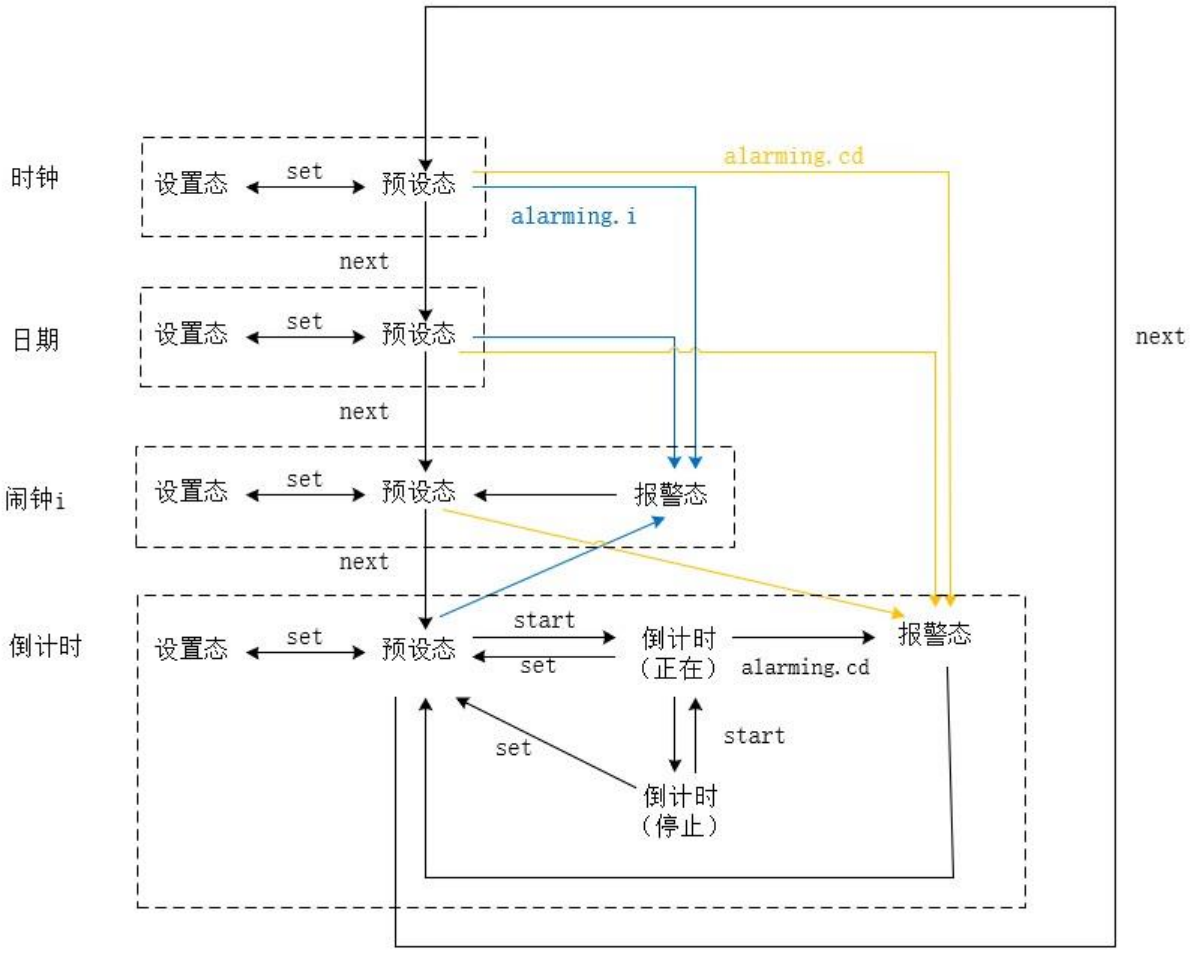
闰年的年份可以被 4 整除，但是不能被 100 整除，或者可以被 400 整除。如果以 ABCD 表示一个年份，每一位都是一个 4 位的 BCD 码的话：

解决问题的关键：一个两位数 AB 是否被 4 整除：即 A*10+B 能否被 4 整除，即[(A*10)%4+B%4]的低两位是否为零。根据观察，当 A 为奇数则 A*10%4=2，否则为 0；而 B%4 的结果为 B 的低两位。

利用以上原理，得到算法：

- 能被 400 整除：CD 全为 0，且 AB 能被 4 整除。
- 可以被 4 整除且不能被 100 整除：CD 不全为 0，且 CD 能被 4 整除，所以只要判断两位十进制数 AB 或 CD 能否被 4 整除。

状态机流程图



状态机功能概述：

1．分为 6 个界面，对应要求的 6 个功能。

2．每个界面有两个基本状态：**设置态**和**预设态**，设置态又有 **9 个子态**。用 8 位的独热码 SET 表示当前页面的设置情况。设置时 8 位仅有一位为 1，表示对应数正在被设置。有一个信号 set 控制 SET 是否为 0（进入/退出设置模式），有另一个信号 left 控制 SET 左移。

SET	含义
00000000	不在设置中(预设态)
00000001	正在设置第一位
00000010	正在设置第二位
00000100	正在设置第三位
……	以此类推

3．还有一个服务于 UI 的状态量 SELECT，表示当前正在显示/操作哪一个功能界面。有一个信号 next 控制此数+1。

SELECT	功能
000	时钟
001	日期

010	闹钟 1
011	闹钟 2
100	闹钟 3
101	倒计时

4．而闹钟和倒计时由于自身特殊的功能，有自己的独特的状态：报警态。有一个信号 alarming 控制进入报警态，其他所有信号控制退出报警态（按任意键解除，复用了按钮），同时有一个 stop 的输入信号同样控制退出报警态(见原理图-报警-取消第一段)。进入报警态优先级最高，可以跨功能跳转到此态（即不管当前处于什么状态什么界面，都直接进入该报警态）。

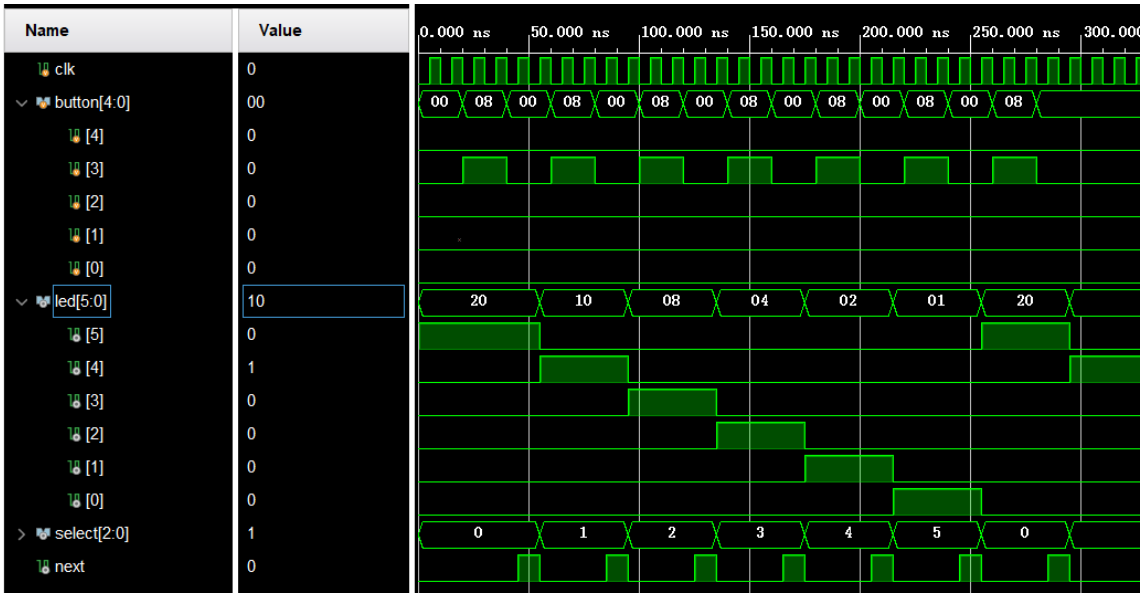
5．倒计时有“倒计时态(正在)”和“倒计时态(停止)”，统称“倒计时态”。从实际使用考虑，倒计时态不能直接到设置态，因此复用了设置模式的信号，如果在倒计时态，set 信号表示到预设态，而不是直接到设置态。此外，还有一个信号 start 控制进入倒计时态和倒计时子态的切换。

仿真结果

主要功能仿真：

1. 界面切换

功能描述：按【下】则切换界面，六次一循环，能从左边六个 LED 灯得知当前为何界面：



波形解释：

button[3]产生脉冲表示按键【下】按下，select[2:0]的值表示当前所处界面，0-5 的 6 个值分别表示不同的 6 个界面，led[5:0]的每位分别控制一个 led 灯的亮灭。如图可见，每当 button[3]产生脉冲后，select 的值发生改变，当达到 5 之后，又变回 0，六次一循环。同时，当 select[5:0]处于不同值时，led[5:0]的不同位为 1，用不同 led 灯的亮灭表征当前所处工作界面。

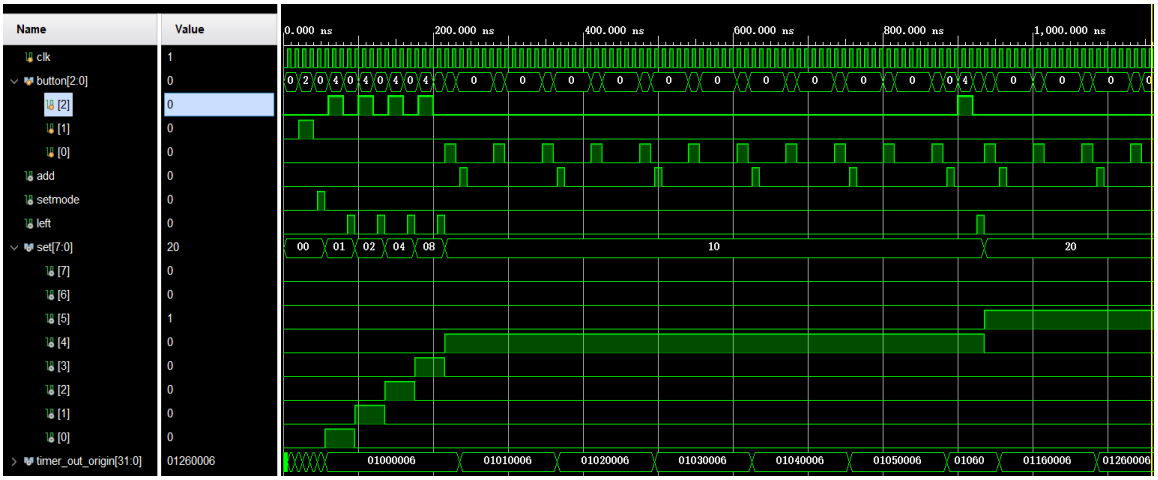
2. 设置模式

功能描述：

按【中】则进入设置模式，再按一次则退出。

设置模式下，闪烁位为当前设置位，按【上】则加一。此时，同单位间进位有效，但不向高一级进位。比如设置分钟时，个位能进位，但十位不会向小时进位。

设置模式下，按【左】改变当前设置位。



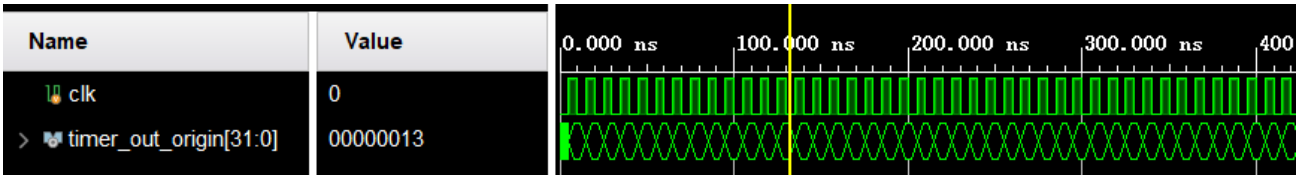
波形解释：

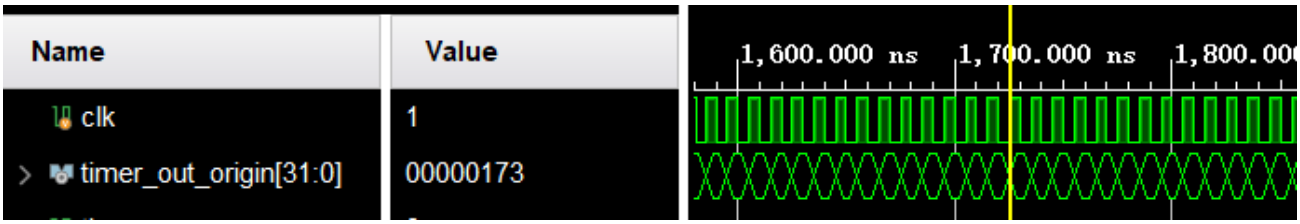
button[2]产生脉冲表示按键【左】按下，button[1]产生脉冲表示按键【中】按下，button[0]产生脉冲表示按键【上】按下，setmode 产生脉冲表示进入设置模式，set[7:0]的值表示当前正在设置的数位，其表示方法与上文中的 SET 变量相同，timer_out_origin[31:0]表示当前时钟各位的具体数值（这里以时钟为例，倒计时和闹钟同理）。

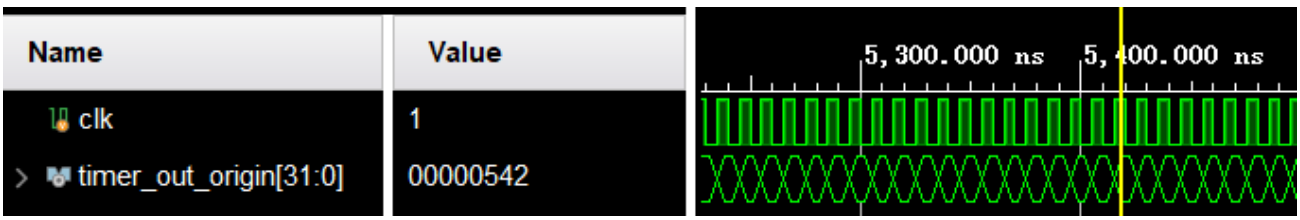
如图可见，一开始，button[1]产生脉冲，紧接着，setmode 也产生了脉冲，即按【中】则进入设置模式。然后，从 60nm 开始，button[2]产生脉冲，每个脉冲后，set[7:0]的值改变，即在设置模式下，按【左】改变当前设置位。在 200nm 处，button[2]的第四个脉冲结束后，button[0]开始产生脉冲，发现在每个脉冲后，timer_out_origin[19:16]的值加 1，且在达到 6 之后，使 timer_out_origin[23:20]的值加 1，实现了进位。即设置模式下，按【上】则当前设置位加一，且进位有效。

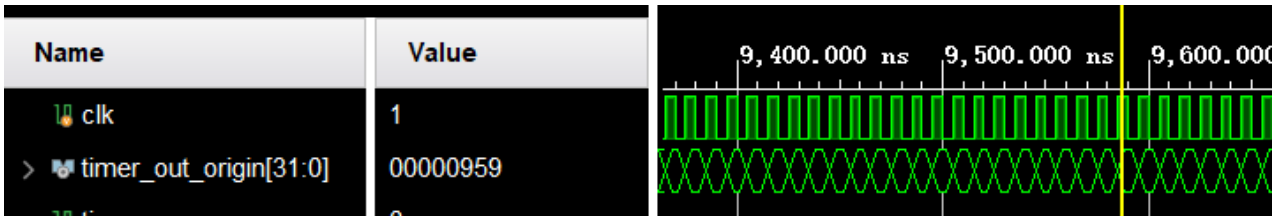
3. 时钟功能

- **模块宏观测试：**





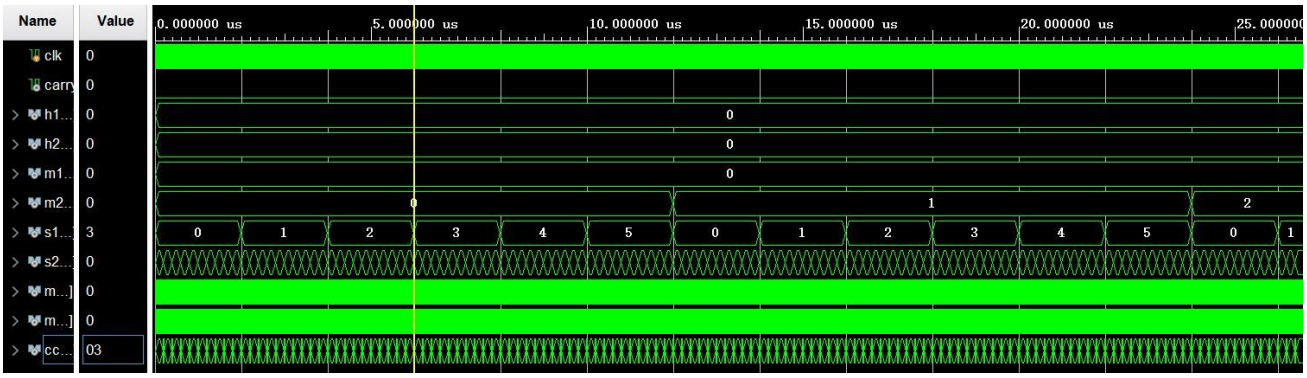




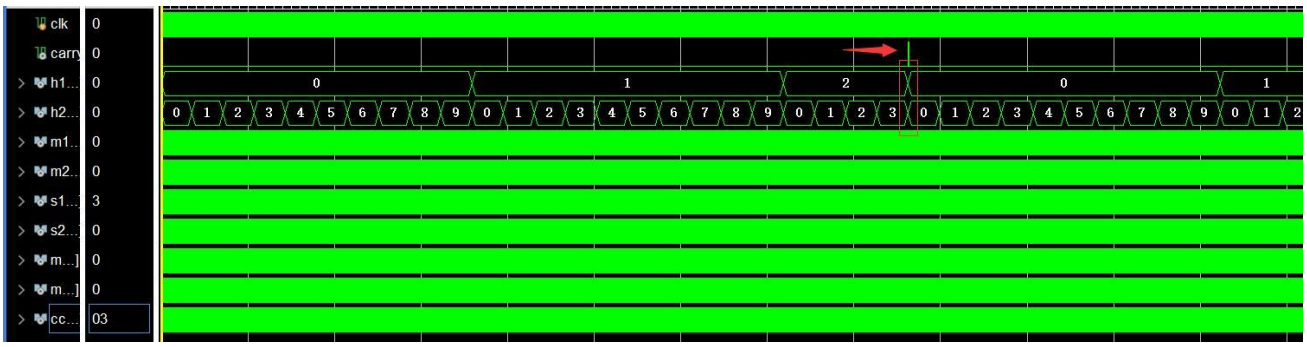
波形解释：

timer_out_origin[31:0]表示当前时钟各位的具体数值。如图，timer_out_origin[31:0]随着时钟周期稳定变化，每一周期后数值加 1，且各位进位正常有效。

- **时钟模块内测试：**



图一

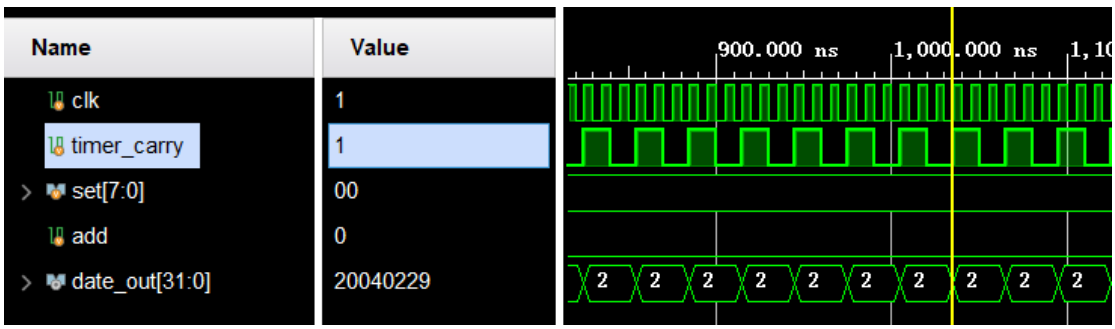


图二

波形解释：

h1~ms2 分别表示数码管最高位到最低位，每一位都是一个 4 位的 BCD 码。图一展示了 ms1 向 s2、s2 向 s1、s1 向 m2 的进位。而图二则展示了 h1=2，h2=3 下一时刻归零的过程 *(即 23 点到 0 点)*。

4. 日期功能



波形解释：

测试了日期模块。timer_carry 实际意义为来自时钟的进位。如图可见，随着时钟周期变化，日期按照日历规律增长，在闰年 2004 年，2 月能达到 29 日，且能正确进位到 3 月（图中未体现）。

5. 闹钟功能

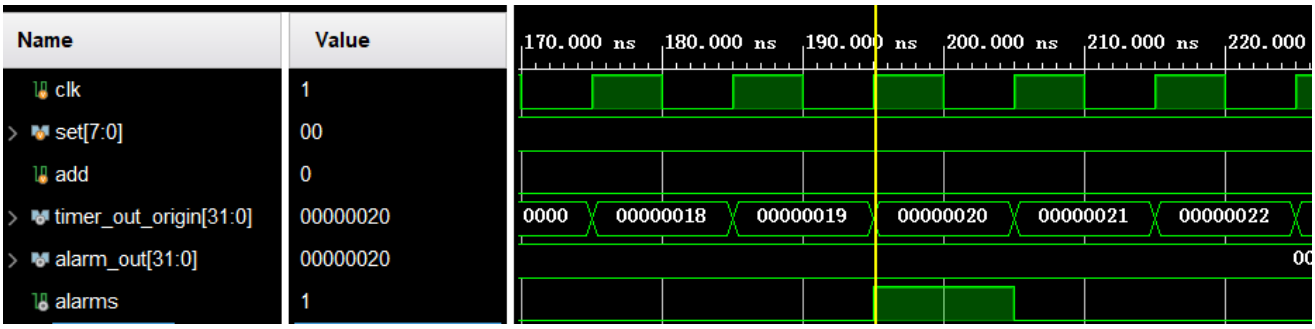
功能描述：

按【右】启用/禁用当前闹钟。

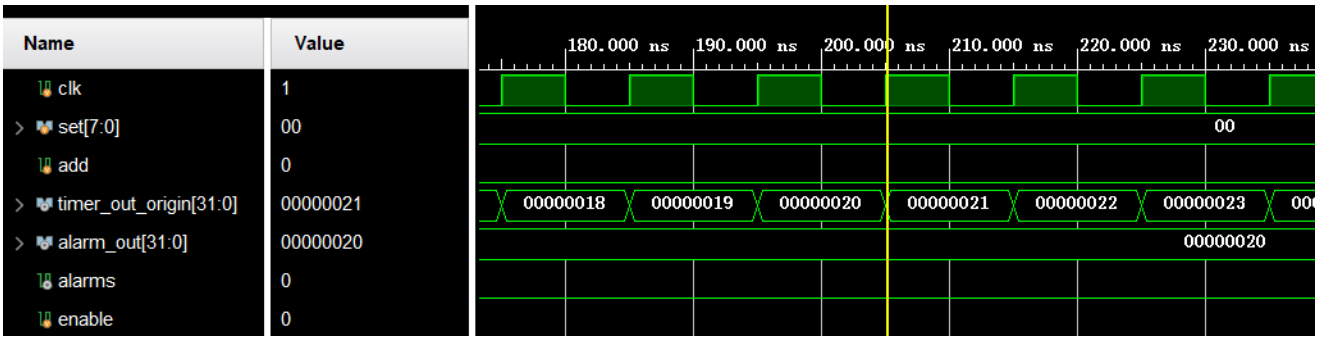
当前闹钟或时钟处于设置模式时，闹钟禁用，与是否启用无关。

为避免歧义，闹钟固定为 24 进制。

闹钟响后，界面将跳转对应闹钟界面，同时开始报警。如果没有按键解除，则响完 5 秒之后暂停，然后 10 秒钟后重新提醒一次后解除。注意，只是声音会停 10s，界面始终闪烁。



图一（使能，闹钟响，发出脉冲）



图二（不使能，没有脉冲）

波形解释：

timer_out_origin[31:0]代表当前时钟各位的具体数值，alarm_out[31：0]表示闹钟响时时钟各位具体数值，alarms 产生脉冲表示闹钟响，当按下一次【右】按键或当前闹钟或时钟处于设置模式时 enable 为 0。

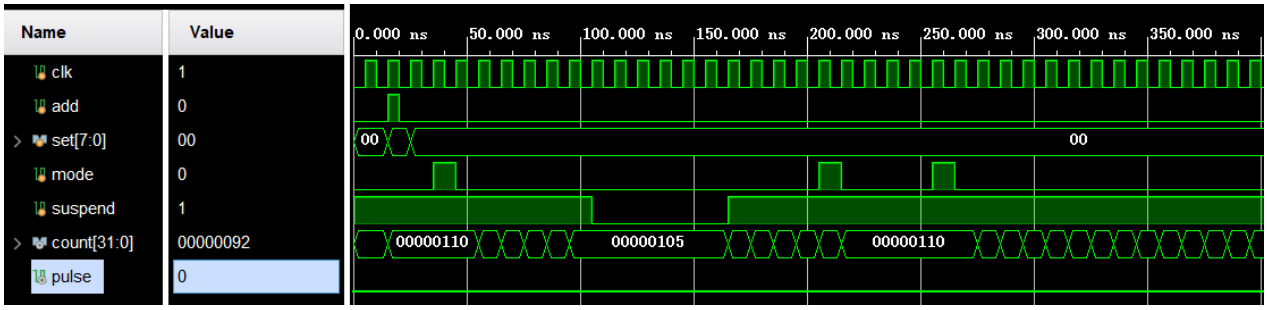
如上图一，仿真中将闹钟时间调为 200ms，当 timer_out_origin[31:0]变为 8'h00000020 时，alarms 立刻产生脉冲，闹钟响，即闹钟功能有效。如上图二， enable 为 0 时，当 timer_out_origin[31:0]变为 8'h00000020 时，alarms 不产生脉冲，闹钟不响。

6. 倒计时功能

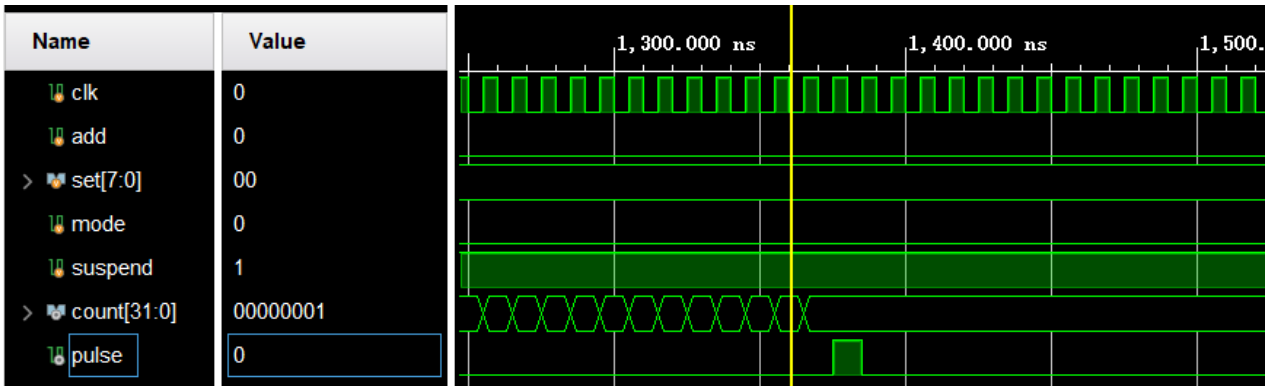
功能描述：

按【右】开始/停止倒计时；停止后可继续倒计时。

倒计时开始后，无论停止或在倒数，按【中】将退出倒计时。



图一



图二

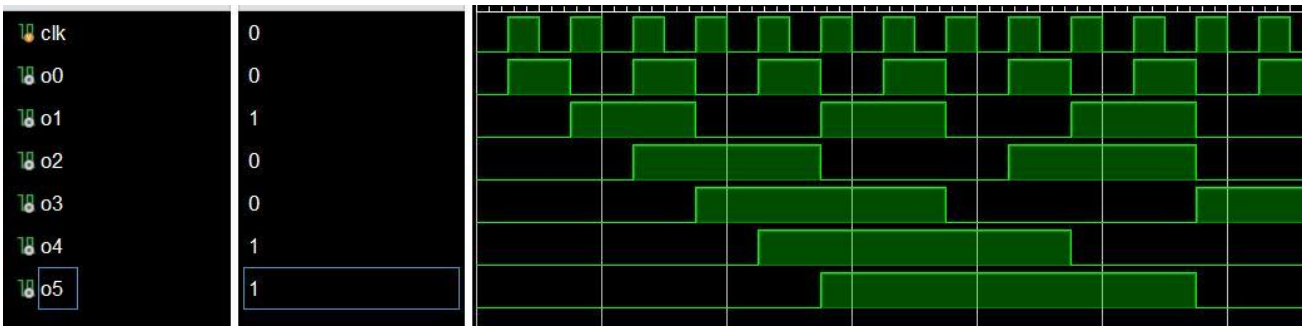
波形解释：

mode 产生脉冲表示按键【中】按下，按键【右】按下，会使 suspend 的值改变，从 0 变为 1 或从 1 变为 0，count[31:0]表示当前倒计时各位具体值，pulse 产生脉冲表示倒计时结束。

如图一所示，倒计时初始值设置为 8'b00000110，一开始，经过一个时钟周期的时间后，count[31:0]的值减一，即倒计时正常有效。105ns 后，suspend 的值由 1 变为 0（【右】按下），倒计时暂停，count[31:0]停留在 8'b00000105，然后在 165ns 处，suspend 的值又由 0 变为 1（【右】再次按下），倒计时继续进行，接着在 215ns 处，mode 产生脉冲（【中】按下），倒计时回到初始值，退出倒计时，接着在 275ns 处，mode 又产生脉冲（系统调整回倒计时界面），倒计时又重新开始，以上体现在操作上即：按【右】开始/停止倒计时，停止后可继续倒计时；倒计时开始后，无论停止或在倒数，按【中】将退出倒计时。最后，如图二，倒计时结束，pulse 产生脉冲。

重要模块仿真：

1. 分频器



```
`timescale 1ns/1ps
// 分频器 输出频率fo和输入频率fi的关系是：
// fo = fi / (2*MAX)
// 板子提供100M的CLK 要分为100Hz应该传参(16, 50000)
module DIVIDER #(parameter SIZE = 3, MAX = 5) (
    input CLK,
    output reg DIV
);
    reg [SIZE-1: 0] counter;
    always @(posedge CLK) begin
        if(counter == MAX-1) begin
            counter <= 0;
            DIV <= ~DIV;
        end
        else counter <= counter + 1;
    end

    initial begin
        DIV = 0;
        counter = 0;
    end
endmodule
```

```
module TEST();
    reg clk;
    wire o0;
    DIVIDER #(1,1) d0(clk, o0);
    wire o1;
    DIVIDER #(1,2) d1(clk, o1);
    wire o2;
    DIVIDER #(2,3) d2(clk, o2);
    wire o3;
    DIVIDER #(2,4) d3(clk, o3);
    wire o4;
    DIVIDER #(3,5) d4(clk, o4);
    wire o5;
    DIVIDER #(3,6) d5(clk, o5);
    always #5 clk = ~clk;
    initial begin
        clk <= 0;
    end
endmodule
```

波形解释：偶数次分频。

2. 闰年判断

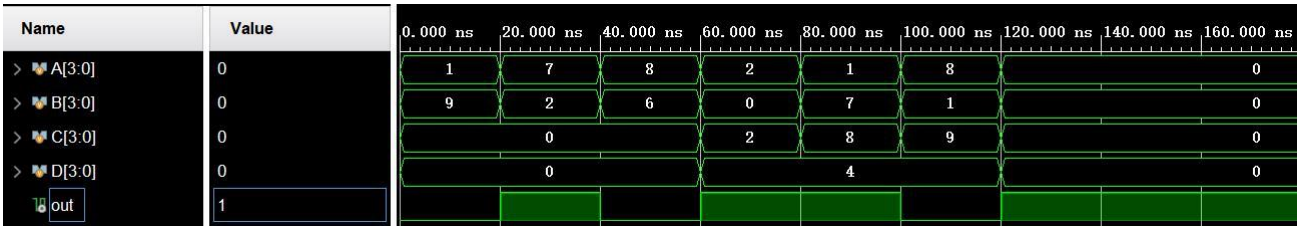
leapYear.sv

```
`timescale 1ns/1ps
// 根据4位BCD码判断是否为闰年
// 闰年的年份可以被4整除，但是不能被100整除，或者可以被400整除
// 输入十进制四位年份的BCD二进制码，共16位，表示为ABCD。
module LEAPYEAR(
    input [3:0] A,
    input [3:0] B,
    input [3:0] C,
    input [3:0] D,
    output reg leapyear
);
    // AB是否被4整除：即A*10+B能否被4整除，即[(A*10)%4+B%4]的低
    // 根据观察，当A为奇数则A*10%4=2，否则为0；而B%4的结果为B的
    wire[1:0] ABmod4_temp = ({A[0],1'b0} + B[1:0]);
    wire ABmod4 = ABmod4_temp == 2'b00;      // 为1则是4的倍数
    // CD是否被4整除同理
    wire[1:0] CDmod4_temp = ({C[0],1'b0} + D[1:0]);
    wire CDmod4 = CDmod4_temp == 2'b00;
    // 组合逻辑判断是否为闰年
    always @(*) begin
        // 能被400整除：CD全为0，且AB能被4整除。
        if ({C,D} == 8'b0) begin
            if(ABmod4) leapyear = 1'b1;
            else leapyear = 1'b0;
        // 可以被4整除且不能被100整除：CD不全为0，且CD能被4整除，
        end else if (CDmod4) leapyear = 1'b1;
        else leapyear = 1'b0;
    end
endmodule
```

TEST.sv

D: > PROGRAM > HDL > test > test.srcs > sim_1 > new > TEST.sv

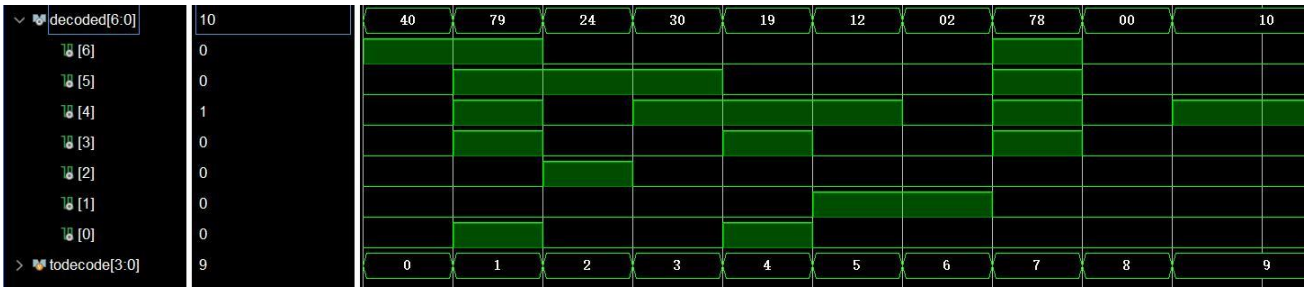
```
module TEST();
    reg [3:0] A;
    reg [3:0] B;
    reg [3:0] C;
    reg [3:0] D;
    wire out;
    LEAPYEAR EAR(A,B,C,D,out);
    initial begin
        A = 1;
        B = 9;
        C = 0;
        D = 0;
        #20;
        A = 7;
        B = 2;
        #20;
        A = 8;
        B = 6;
        #20;
        A = 2;
        B = 0;
        C = 2;
        D = 4;
        #20;
        A = 1;
        B = 7;
        C = 8;
        D = 4;
        #20;
        A = 8;
```



波形解释：年份为 ABCD，只有在闰年时 out 为高电平。

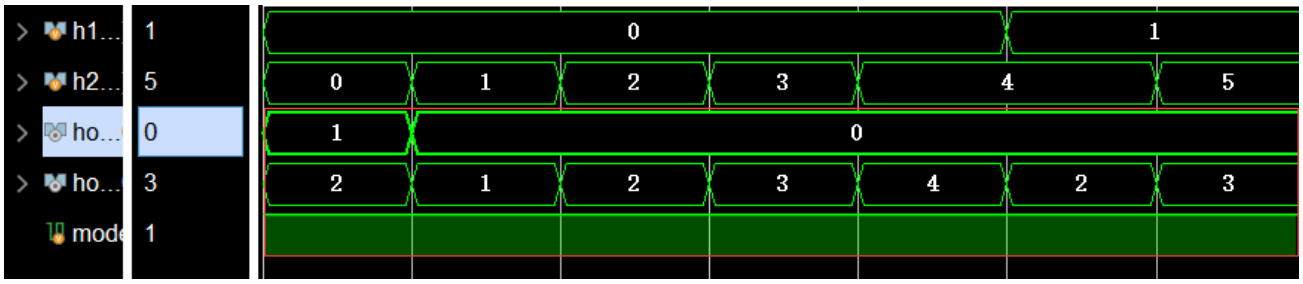
3. 译码器

功能描述：将 4 位 BCD 码转为 7 段数码管显示。需要注意数码管低电平才亮。



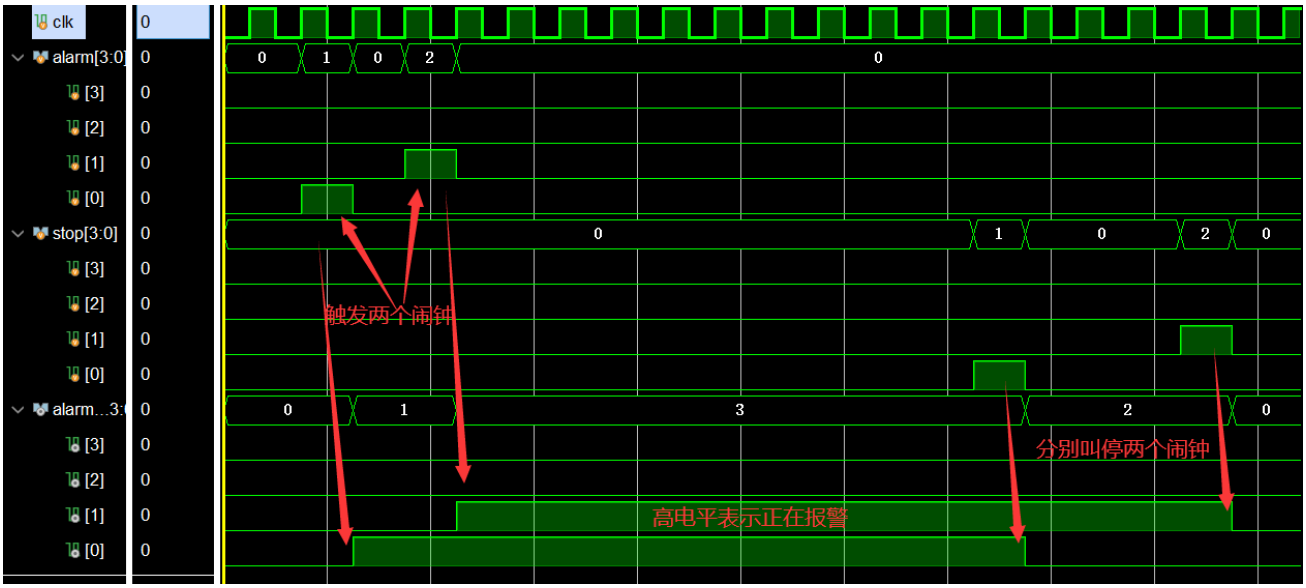
波形解释：todecode 为 BCD 码，而 decoded 中为解码内容，且 a 在最低位。

4. 小时的 12 进制切换



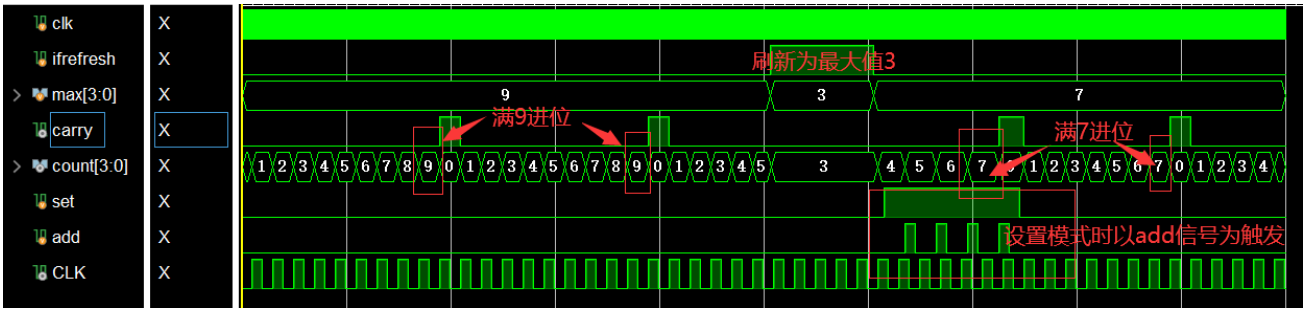
波形解释：上两行是 24 进制的十位-个位，下两行是 12 进制的十位-个位，15 对 03，正确。

5. 闹钟并发测试



波形解释：alarm[3:0]为闹钟到点后的脉冲，经过 FSM 使 alarming 对应位为 1，表示正在报警。而四个 stop 对应四个 alarming，用于分别消除。由图可见，闹钟的消除是独立的，所以对并发的处理是：报警独立，逐个消除。

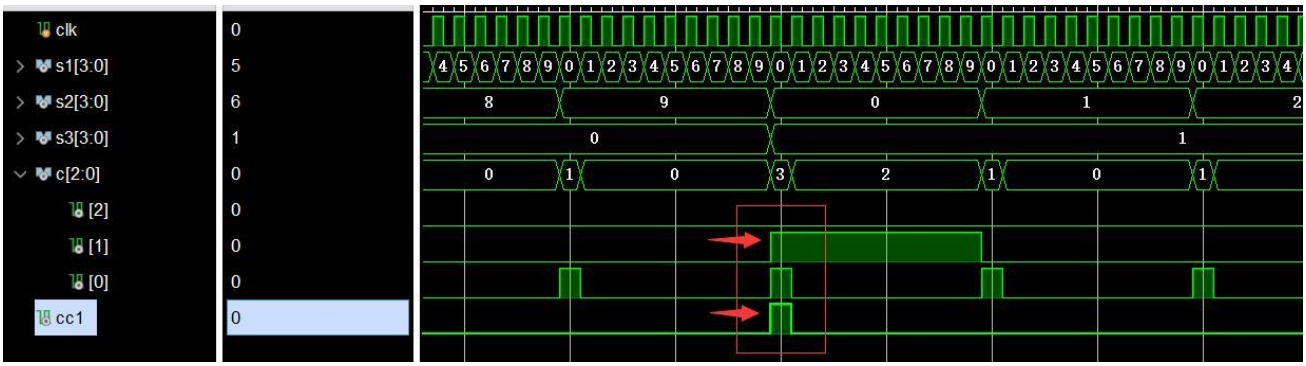
6. 自适应计数器测试



波形解释：max 为外界传入的最大值，四个进位体现了动态调整最大值。当 set=0 时，以 CLK 为触发而加一；set=1 时，以 add 为进位加一，体现了设置模式下计数器的加一动作触发方式。当 ifrefresh=1 时，刷新为当前的合法范围：此前为 5，而最大值设置位 3，故刷新为 3，体现了自适应。此计数器用在小时的个位，因 0~20 个位是满 9 进位，而 20 以上为满 4 进位。

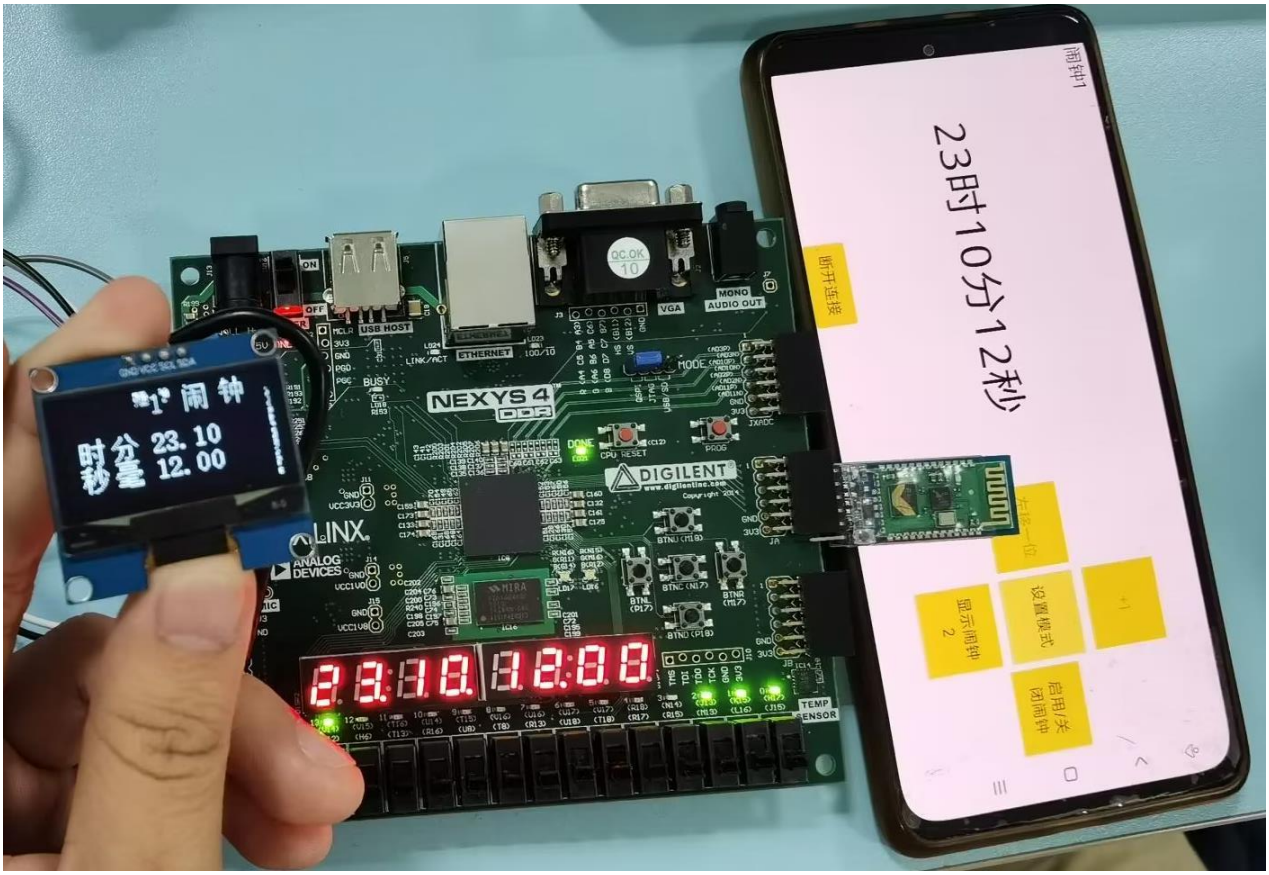
7. 上升沿转脉冲

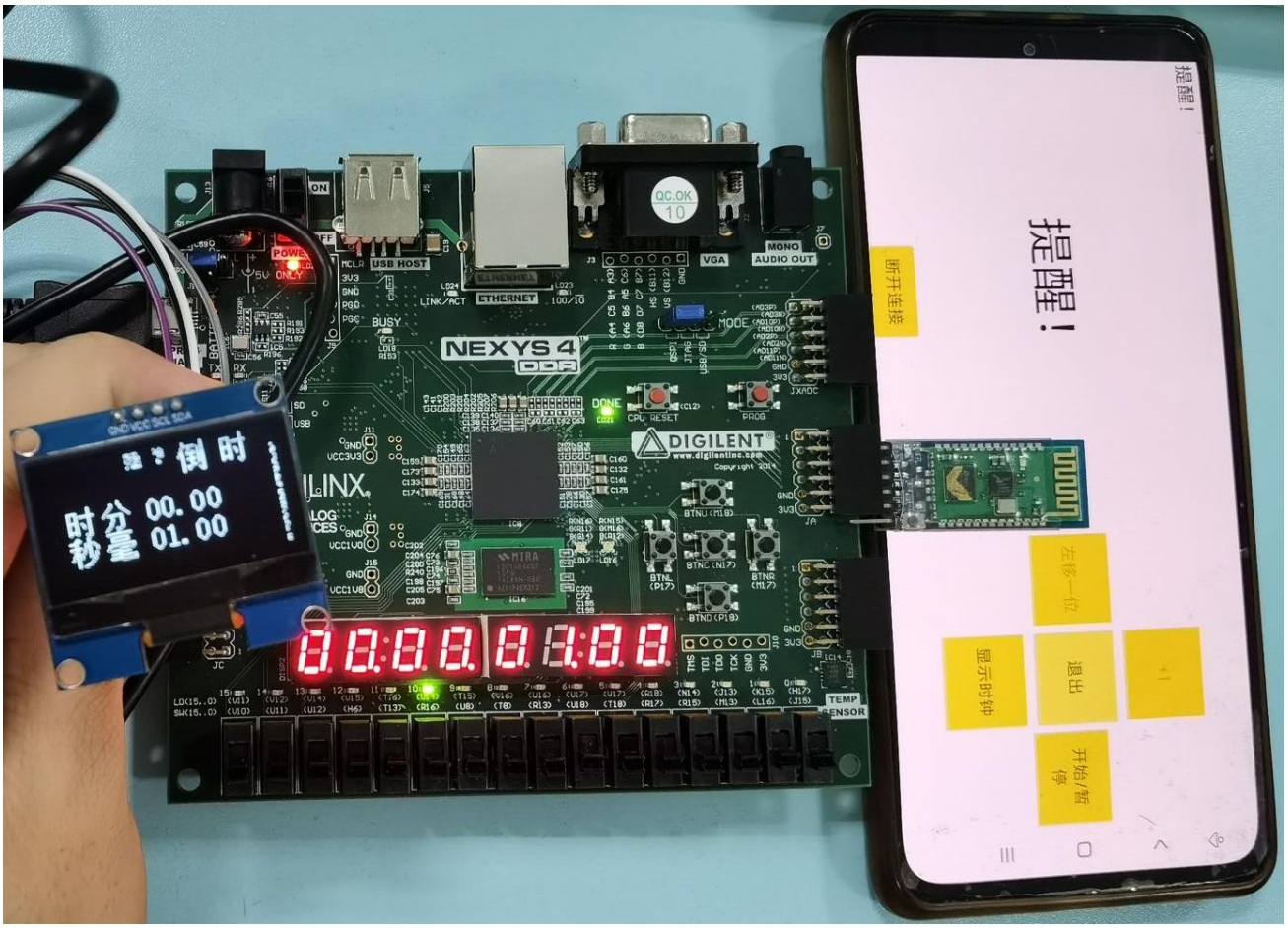
功能说明：如“计时器原理图”中所言，需要将宽电平转为短脉冲。

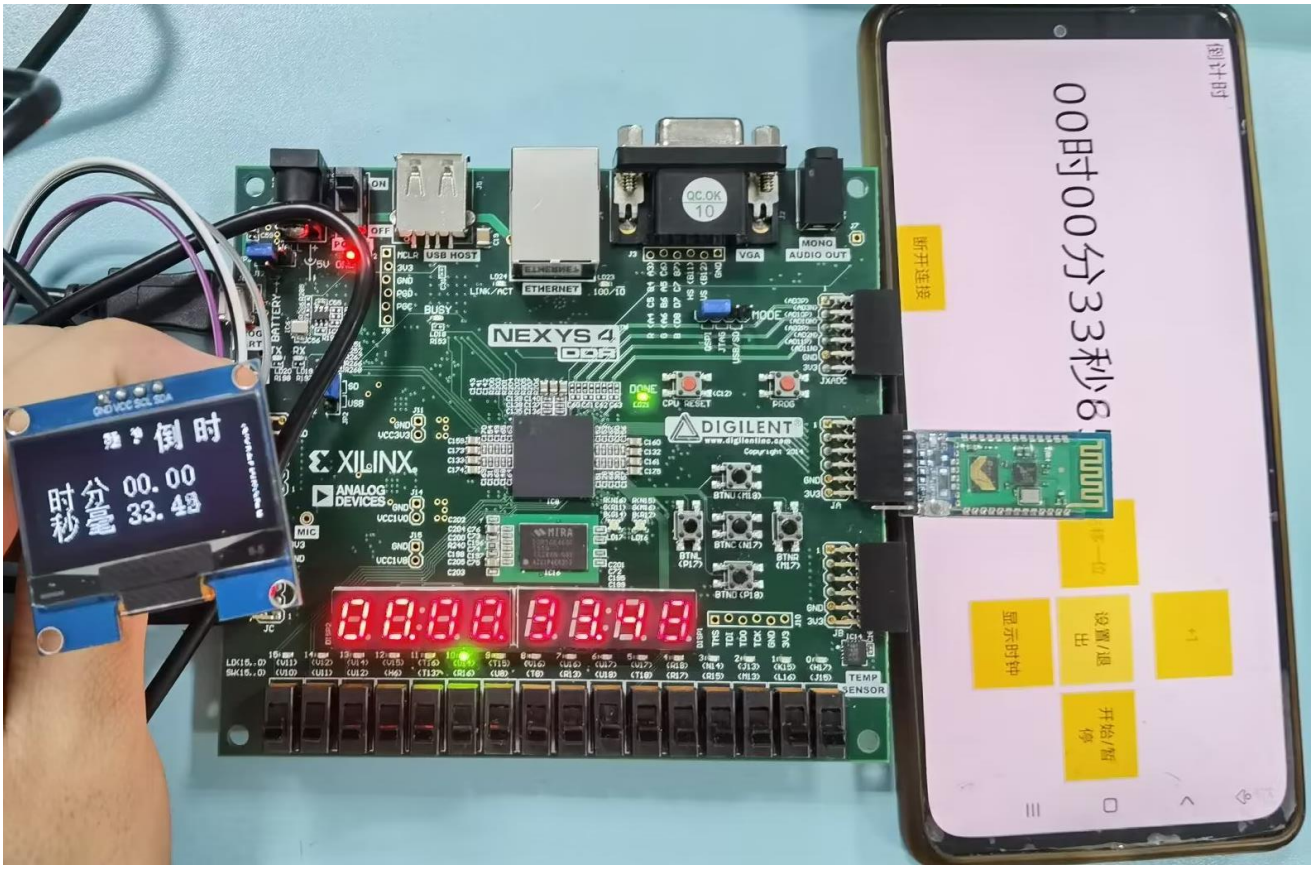


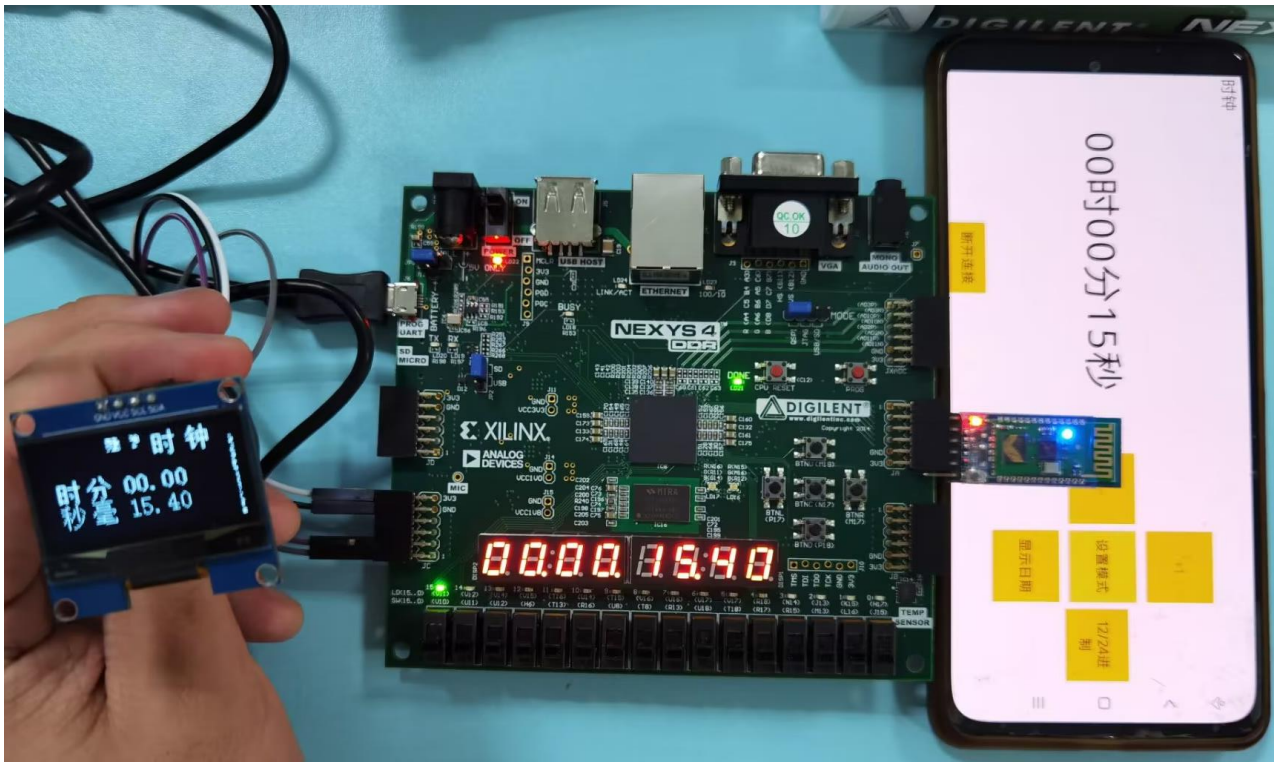
波形解释：c[2:0]表示 3 个计数器的进位。c[0]为最低位的进位，而 c[1]为第二位的进位。可见第二位的进位持续时间很长，在前面的也解释了长电平的后果。而 cc1 为 c[1]转脉冲之后的结果，可见其长度和 c[0]相同。

实拍


闹钟1
23时10分12秒
断开连接
设置模式
显示闹钟
2
启用/关闭闹钟
闹钟
时 23.10
分
秒 12.00
NEXYS 4
DDR
DIGILENT
ANALOG DEVICES
LINX
USB HOST
ETHERNET
VGA
MONO AUDIO OUT
TEMP SENSOR


倒时
时 00.00
秒 01.00
提醒！
断开连接
左移一位
显示时钟
退出
+1
开始/暂停


倒时
时 00.00
秒 33.43
倒计时
00时00分33秒6
断开连接
显示时钟
设置/退出
+1
开始/暂停



分工

龙艺文同学完成了 OLED 的显示和倒计时模块的编写，以及部分仿真（模块宏观仿真）；其余由李睿刚同学完成，包括整体框架设计、所有基本模块、时钟模块、日期模块、闹钟模块、状态机、蓝牙手法、安卓开发、报告撰写等等等。工作量分配为龙 35%，李 65%。

总结

不足之处：

哪个闹钟响了应该有个显示。修改方法：只要用报警标志位驱动 LED 即可。

Bug：

倒计时如果低 3 位都是 0，第三位会出现 0->F 的问题。修改方法：改变之前以秒为基准的对进位规则的判断，以每一微秒为基准计时，并以此为基础来判断进位。

龙艺文：

通过暑期学校，我从了解 verilog，到掌握其语法规则并能够进行简单的编程开发，将理论知识付诸于实践，我切实体会到了计时器、触发器、锁存器等基本模块的妙用，将各种简单模块通过数字逻辑进行组合，就能得到精巧的数字系统，从而实现很多功能。在编写倒计时模块的过程中，为实现倒计时暂停及重置，经过请教、查询资料，我学习到了检验上升沿的方法，从而通过脉冲以及电平变化来切换倒计时模块的模式，从而实现暂停和重置。通过研究 OLED 屏幕显示，我学习并掌握了 IIC 通信协议，OLED 屏幕底层显示原理，OLED 屏幕的底层硬件实现，以及实现显示的算法等。并且，经过总结，我最终熟练掌握了驱动 OLED 屏幕显示的方法。暑假学校虽短，但我确实通过编程，加深了对数字系统的理解，获益匪浅。

关于 OLED 模块，我按照 IIC 协议驱动 4 针 OLED 屏幕，简洁清楚地显示当前界面状态，数码管上显示的数值，以及这些数值代表的含义。为了使代码更加清楚简单，我先进行形象的类如刷新屏幕等的操作，再

将这些操作编译为符合 IIC 协议的输出，最终驱动屏幕。相比用单片机驱动，FPGA 的实现别有特点。

李睿刚：

虽然之前的课程中已经用 verilog 写了两次课程作业了，但是本次为第一次烧写 FPGA，对硬件的综合设计有了更深的理解。总结下来技术性的收获有以下几点：

1. 上升沿读取状态并执行一些操作，下降沿更新状态，能有效避免玄学问题。
2. 不要在多个 always 中对一个 reg 赋值。如果要实现一个影响一个再影响一个，可以用的框架是：某一个 always 中 CLK 每个上升沿读取相关信号状态，并对某个 reg 进行操作。下降沿来判断是否有沿或者更新状态。比如我在在 UART 里面用的。
3. 承上，最好在每个 CLK 边沿判断，而不是别的什么触发信号触发什么操作。当然后者能一定程度上简化代码，减少耦合，减轻硬件负担；但是前者适合大规模的项目，避免了复杂的连接关系和时序关系。
4. 沿判断电路：两个 reg，一个保存之前的，一个保存当前采样的，然后两者之间组合逻辑运算可以得到什么沿。

```
reg signal_prev, signal_now;
always @(negedge CLK) begin     // 下降沿更新状态
    signal_prev <= singal_now;
    signal_now <= signal;
end
wire signal_pos = ~signal_prev & signal_now;  // 上升沿
wire signal_neg = signal_prev & ~signal_now;  // 下降沿
```

5. 脉冲产生电路：原理同上，要产生脉冲只要翻转 reg 即可，注意延迟。

```
reg pulse_cache, pulse_generator;
always @(negedge CLK) begin     // 下降沿同步置零脉冲信号
    pulse_cache <= pulse_generator;
end
wire pulse = pulse_generator ^ pulse_cache;   // 脉冲信号
// 使用时：
pulse_generator <= ~pulse_generator;
```

还有些不足，关于计数器的设计。设计时思路被板子上仅有的 5 个按钮局限了，导致后续写 app 时无法拓展其功能，只能模拟板子上的按钮，相当于一个遥控器（不然可以加网络时钟校准等）。一开始未了解到上面的这些设计框架，核心部分的计数器用的还是之前数电课写钟的方法（即不依赖 CLK 更新，只依赖进位的上升沿），虽然硬件简单，但难以拓展，且需要额外考虑诸如长脉冲转短脉冲等细节。

如果现在重新设计，我不会再把十位个位分开，直接按照写日历的月日的方法用 case 将二进制映射为两位 BCD。这样能避免很多问题，而且功能易于拓展。

有一个遗憾：身边没有 VGA 的显示器，只有 HDMI 便携屏。写过 VGA 输出，但用 VGA 转 HDMI 但屏幕不亮。发现我的屏幕分辨率只能 1920*1080@60，用 PLL 倍频改参数还是无济于事。问题可能出在很多地方，比如显示器、转接口、程序。没有 VGA 屏幕，难以排查问题，故没有继续下去。

附录

文件结构

```
  |  alarm.sv          (李)
  |  alarmTime.sv      (李)
```

```
│  button.sv         (李)
│  countdown.sv      (龙)
│  counter.sv        (李)
│  decoder.sv        (李)
│  divider.sv        (李)
│  fsm.sv            (李)
│  hourMode.sv       (李)
│  main.sv           (李)
│  posedgeToPulse.sv(李)
│  selector.sv       (李)
│  sparkle.sv        (李)
│  timer.sv          (李)
│  tube.sv           (李)
│
├─CALENDAR
│      calender.sv   (李)
│      day.sv        (李)
│      leapYear.sv   (李)
│      month.sv      (李)
│
├─oled
│      iic.v            (龙)
│      OLED_FontData.v  (龙)
│      OLED_Init.v      (龙)
│      oled_main.v      (龙)
│      OLED_NumData.v   (龙)
│      OLED_Refresh.v   (龙)
│      OLED_SelData.v   (龙)
│      OLED_ShowData.v  (龙)
│      OLED_ShowFont.v  (龙)
│      OLED_Top.v       (龙)
│
└─USART
        receiver.sv (李)
        sender.sv   (李)
        usart.sv    (李)
```

本项目已上传 Github：https://github.com/madderscientist/codeRoad/tree/main/powerClock