数字设计实验课大作业设计报告

数据科学与计算机**学院**　　移动信息工程**专业**　　**15 级 18 班**　　张镓伟　　**学号**：15352408

设计题目：数字时钟设计。

一、设计目的

1. 熟悉并掌握 Vivado 的基本使用方法。
2. 熟悉 Verilog HDL 硬件描述语言，能够使用它编写不太复杂的程序。
3. 提高自己进行组合逻辑设计和时序逻辑设计的能力。
4. 帮助自己理解并巩固理论课所学知识。

二、设计要求

1. 能够正确的进行时间的计时。
2. 能够切换显示时分或者分秒。
3. 能够人工调整时间。
4. 能够进行闹钟的调整设定。
5. 自由发挥功能。

三、设计所需工具

1.BASYS3 Xilinx Artix-7™ FPGA(XC7A35T-ICPG236C) 开发板
2.Vivado 集成设计环境。

四、预期设计效果

1. FPGA 的四位数码管默认显示当前时和分，通过 SW15 开关可控制切换显示时分/分秒。秒同时以二进制的形式亮灯（LED6~LED0）

2. 打开开关 SW14 可以开启设置时间功能。按钮 BTNL 和 BTNR 分别对应小时+1，小时-1；按钮 BTNU 和 BTND 对应分钟+1，小时-1；按钮 BTNC 可以复位清零。(注,此模式的调整需要关闭开关 SW12 才能成功进行)。

3. 打开开关 SW12 开启闹钟设置功能，此时数码管从显示当前时间切换至显示设置闹钟的时间(但不影响当前时间继续计时)。按钮 BTNL 和 BTNR 分别对应小时+1，小时-1；按钮 BTNU 和 BTND 对应分钟+1，小时-1；按钮 BTNC 可以复位清零。(注，此模式的调整需要关闭开关 SW14 才能成功进行)。

4. 打开开关 SW13 可以开启闹钟功能。但时间与闹钟设置时间相同时 LED15~LED9 这 7 个灯会亮。

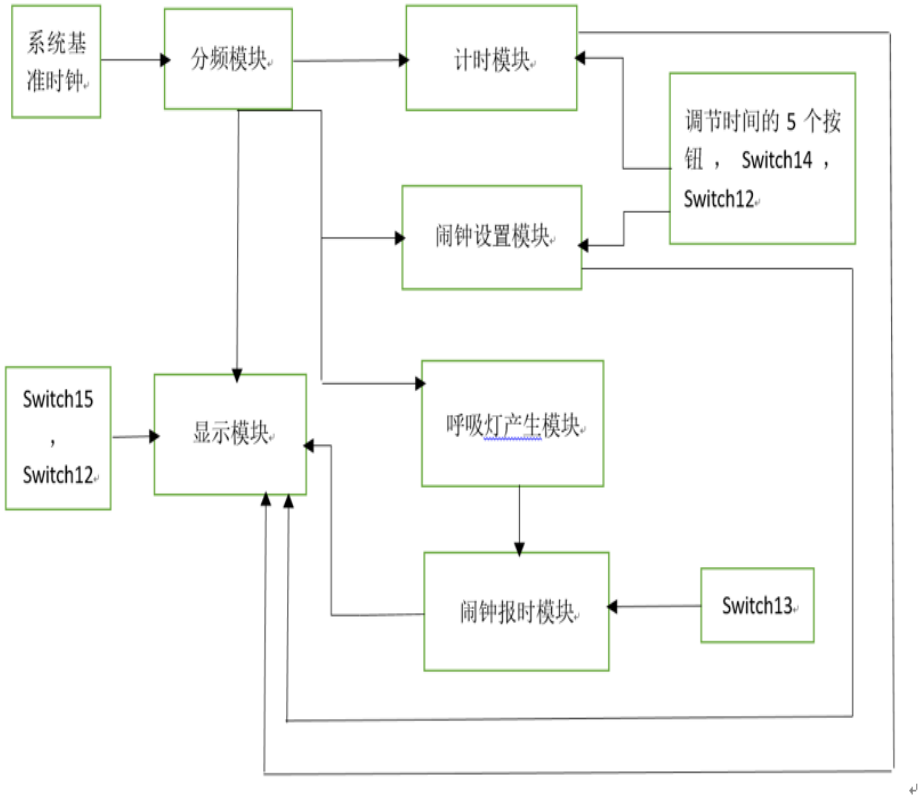
5. 创新内容我做了呼吸灯，并将呼吸灯用到了闹钟的报时亮灯上。

五、总体设计

1. **外部输入：**100MHZ 的时钟信号，高电平有效复位按钮*1，高电平有效调整时间按钮*4（分别为小时+1，小时-1，分钟+1，分钟-1），时分/分秒切换显示开关*1，调整时间开关*1。，闹钟设置开关*1，开/关闹钟功能开关*1.

2. **外部输出：**4 联装 7 段码显示数码管(用来显示时间)，led*6(将秒以二进制显示)，led*7(闹钟报时闪烁，呼吸灯)。

3. 此次设计的逻辑结构由分频模块，显示模块，计时模块，设置闹钟模块，闹钟报时模块及呼吸灯模块组成。分频模块将 FPGA 100MHZ 系统时钟进行分频成三路时钟信号，一路用来计时，一路用来扫描按钮，一路是数码管动态显示的扫描频率。显示模块采用动态扫描的方式显示 4 个需要显示的数字以及判断当前状态需要显示的 led 灯。设置计时模块完成计时功能和调整时间功能。设置闹钟模块通过一定频率扫描按钮并进行闹钟时间调整操作。呼吸灯模块用来进行产生呼吸灯的相关操作。闹钟报时模块判断闹钟设置值与当前时间是否相符，符合将调用产生的呼吸灯。



系统总体框图

六、具体设计

外部输入输出以及内部变量一览：

```
//外部输入输出
 input              clk100MHZ,//系统的时钟
 input       [15:0] SW,//开关
 input       [4:0]  BNT,//按钮
 output reg [15:0] LED,//led灯
 output reg [6:0]  SEG,//数码管七段码
 output reg [3:0]  AN//数码管选通

//内部变量
reg [7:0]clk=8'b0;//clk[7]为2HZ,clk[0]为256HZ，用于时间计数及设置
 reg [5:0]  clk2 = 6'b0;//clk[5]为8HZ，用于闹钟设置按钮扫描
 reg [31:0] clk_cnt = 32'b0;//100MHZ计数器
 reg [7:0]  second = 0;//存储秒
 reg [7:0]  minute = 0;//存储分
 reg [7:0]  hour = 0;//存储时
 reg [3:0]  digit;//存储显示的数字
 wire [1:0]  scan;//数字的扫描显示频率
 reg [7:0]   alerh = 0;//存储闹钟的时
 reg [7:0]   alerm = 0;//存储闹钟的分
 assign scan = clk_cnt[15:14];
```

1.分频模块

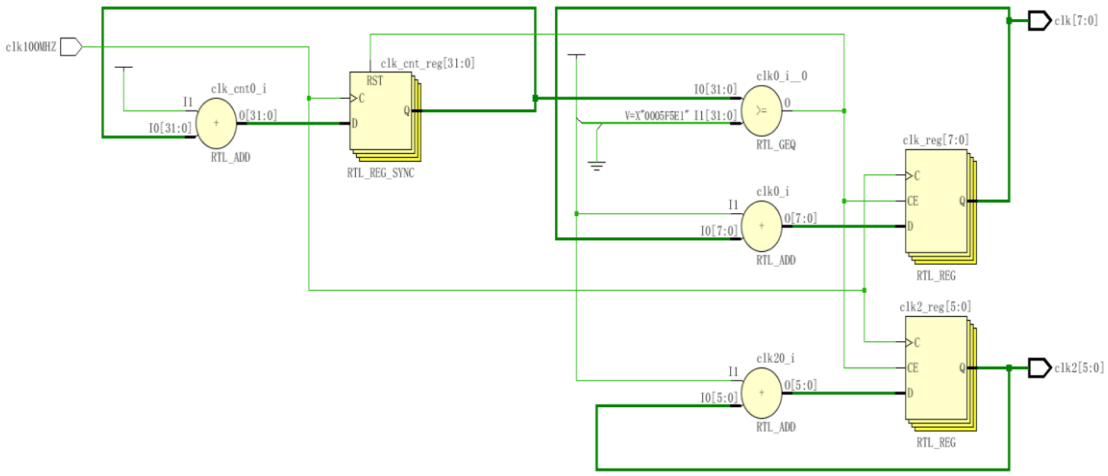
本时钟电路的设计中未使用外接时钟芯片，故需对 FPGA 提供的 100MHZ 的时钟频率进行分频，得到计时所需的时钟频率；时钟计数需要 1HZ 频率，扫描按钮我使用的是 4HZ 的频率。这里我使用了一个 8 位寄存器，最高位是 1HZ，最低位是 128HZ，第 6 位就是 4HZ。要从 100MHZ 得到 128HZ，通过计算 10^8/128/2=390625 知 100MHZ 的时钟计数 390625 次就是 128HZ 的频率。由于还要考虑系统的执行时间问题，我将当前与当前时间变化的触发沿和闹钟设置按钮的触发沿分开两个 clock。

源程序：

```
always @(posedge clk100MHZ)
 begin
     if (clk_cnt >= 390_625)
     begin
         clk <= clk + 1;
         clk2 <= clk2 + 1;
         clk_cnt <= 32'b0;
```

```
    end else begin
        clk_cnt <= clk_cnt + 1;
    end
end
```

Vivado 综合得电路原理图



2.显示模块

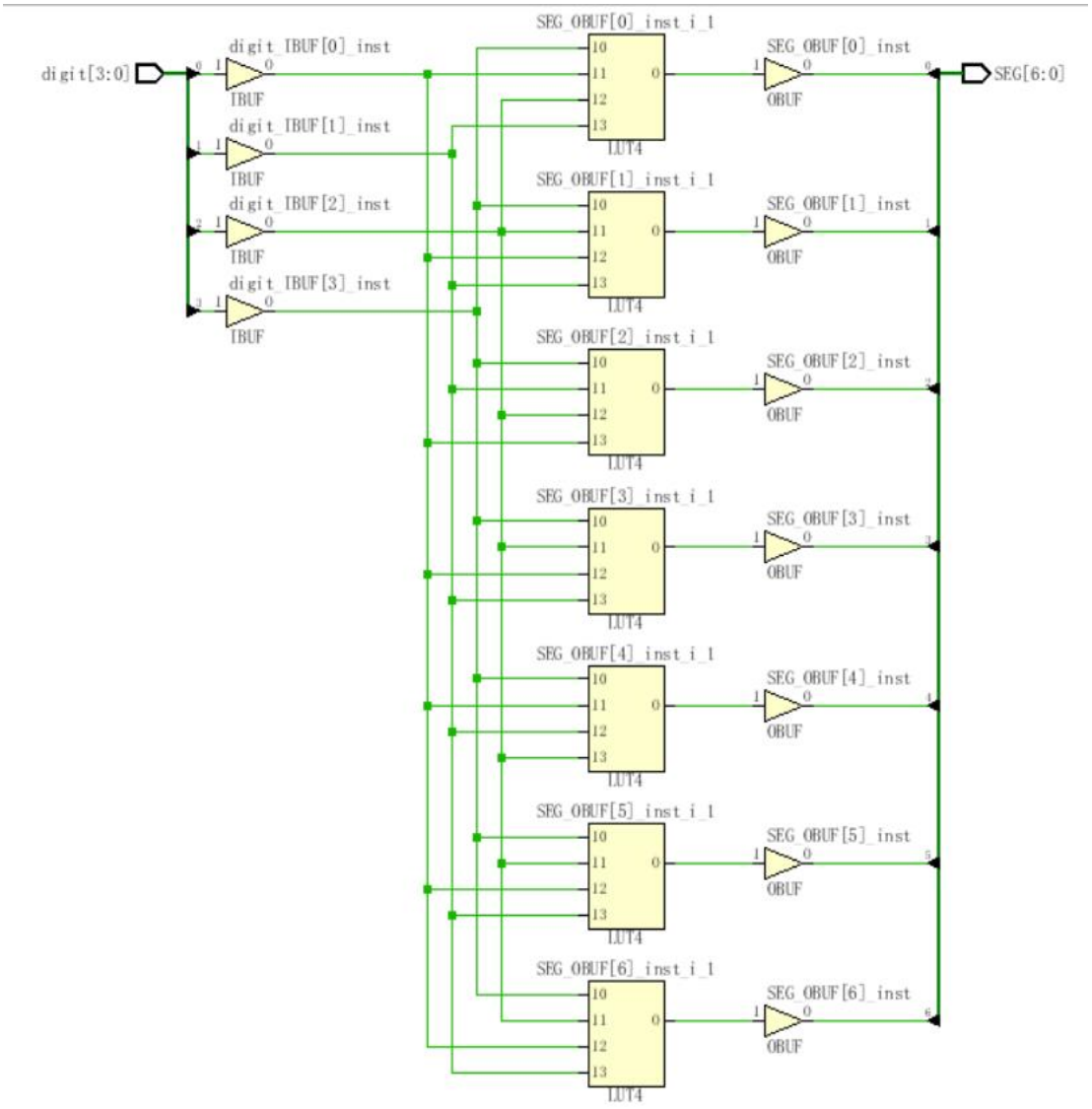
显示模块是将时间的数值转换成数码管段码值，并通过动态扫描的方式在 4 个数码管上实时显示时间。定义变量 digit 来控制数码管的显示数字，定义变量 AN 来控制数码管的显示位置。通过变量 scan 来给数码管的各个位选位送出低电平信号。通过 SW12 控制显示的是当前时间还是设置闹钟时间，通过 SW15 控制显示的当前时间的时分/时秒。将存放秒的状态的值赋给 7 个 LED，使得其能根据秒的二进制亮灯。Basys3 板子的数码管选通信号和控制数字某一段亮的信号都是低电平。

源程序:

```
//显示数字
always @(*)
begin
    case (digit)
            //                  abcd_efg
        0:      SEG = 7'b0000_001;          //0
```

```
        1:      SEG = 7'b1001_111;          //1
        2:      SEG = 7'b0010_010;          //2
        3:      SEG = 7'b0000_110;          //3
        4:      SEG = 7'b1001_100;          //4
        5:      SEG = 7'b0100_100;          //5
        6:      SEG = 7'b0100_000;          //6
        7:      SEG = 7'b0001_111;          //7
        8:      SEG = 7'b0000_000;          //8
        9:      SEG = 7'b0000_100;          //9
        default: SEG = 7'b1111_111;
    endcase
end
```

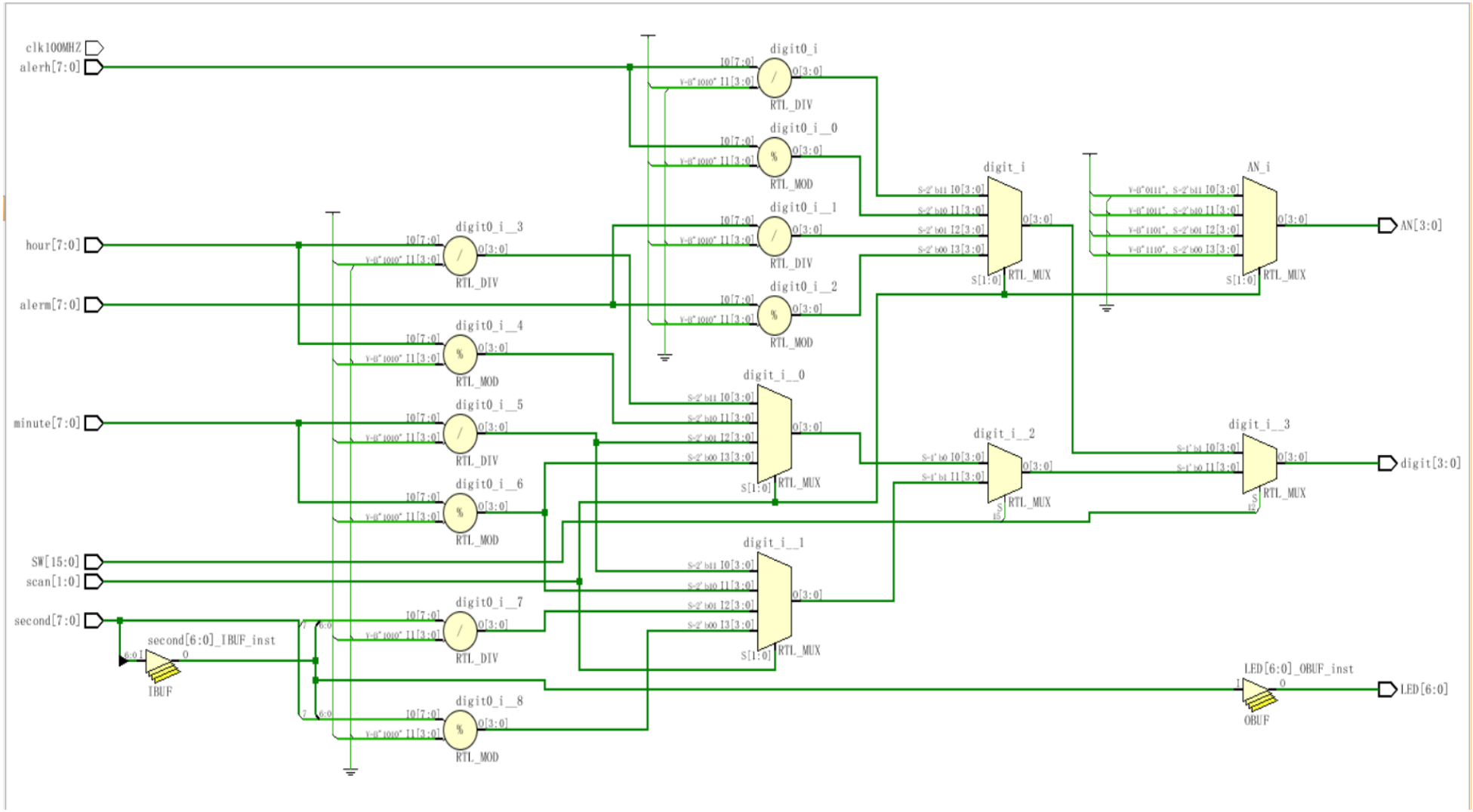
Vivado 综合得电路原理图



```
//选择显示的位置和根据不同模式选择相应的数字
always @(*)
begin

    case (SW[12])
        1:begin//设置闹钟模式
            case (scan)
                3: begin
                    AN = 4'b0111;
                    digit = alerh / 10;
                end

                2: begin
                    AN = 4'b1011;
                    digit = alerh % 10;
                end

                1: begin
                    AN = 4'b1101;
                    digit = alerm / 10;
                end

                0: begin
                    AN = 4'b1110;
                    digit = alerm % 10;
                end
                default: AN = 4'b1111;
             endcase

        end

        0:begin
            case (SW[15])
                0:begin//显示时分的模式
                    case (scan)
                        3: begin
                            AN = 4'b0111;
                            digit = hour / 10;
                        end
                        2: begin
                            AN = 4'b1011;
                            digit = hour % 10;
                        end
```

```
                    1: begin
                        AN = 4'b1101;
                        digit = minute / 10;
                    end
                    0: begin
                        AN = 4'b1110;
                        digit = minute % 10;
                    end
                    default: AN = 4'b1111;
                endcase

            end

            1:begin//显示分秒的模式
                case (scan)
                    3: begin
                        AN = 4'b0111;
                        digit = minute / 10;
                    end
                    2: begin
                        AN = 4'b1011;
                        digit = minute % 10;
                    end
                    1: begin
                        AN = 4'b1101;
                        digit = second / 10;
                    end
                    0: begin
                        AN = 4'b1110;
                        digit = second % 10;
                    end
                    default: AN = 4'b1111;
                endcase
            end
        endcase
    end
endcase
LED[6:0] = second[6:0]; //秒还以二进制的形式亮led灯
end
```

Vivado综合得电路原理图


clk100MHZ
alerh[7:0]
hour[7:0]
alerm[7:0]
minute[7:0]
SW[15:0]
scan[1:0]
second[7:0]
second[6:0]_IBUF_inst
IBUF
digit0_i
digit0_i__0
digit0_i__1
digit0_i__2
digit0_i__3
digit0_i__4
digit0_i__5
digit0_i__6
digit0_i__7
digit0_i__8
RTL_DIV
RTL_MOD
digit_i
digit_i__0
digit_i__1
digit_i__2
digit_i__3
RTL_MUX
AN_i
AN[3:0]
digit[3:0]
LED[6:0]_OBUF_inst
OBUF
LED[6:0]

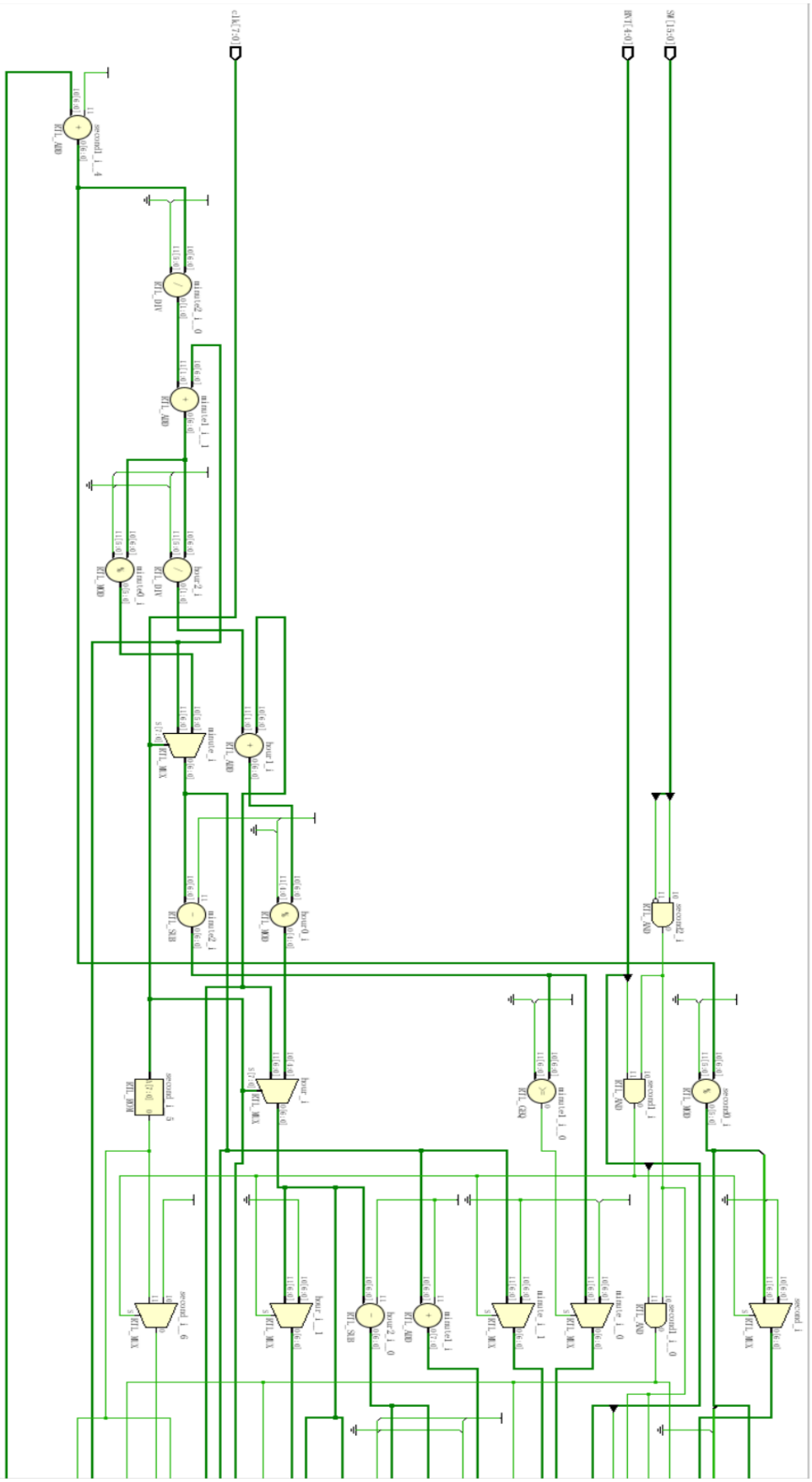
3.计时模块

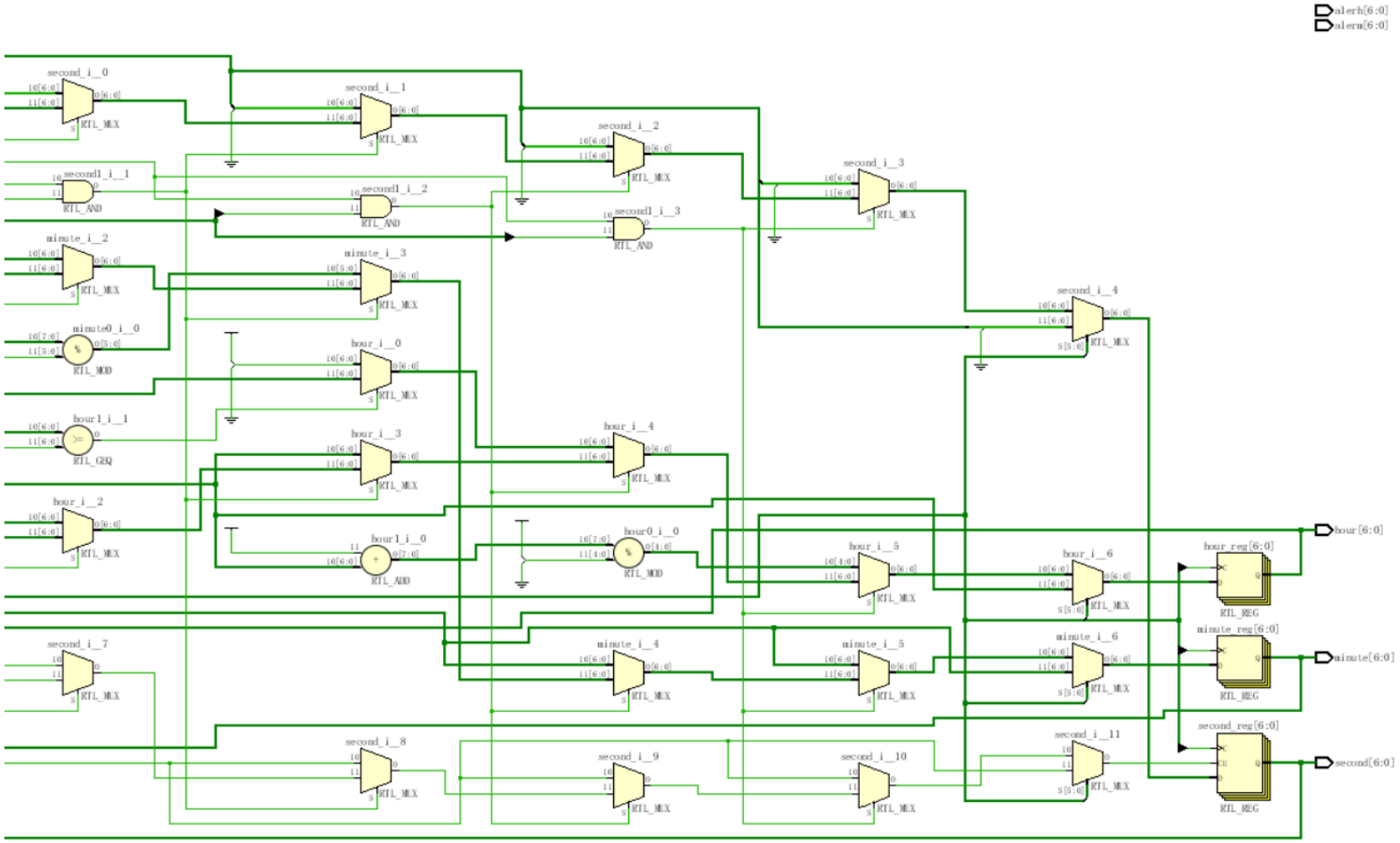
计时模块同时包含了时间的正常计数以及按按钮调整时间两个功能。由于正常计数和扫描按钮所需的处罚沿频率不一样，所以我以一个比他们所需频率更快的频率作为触发沿，再模块里再判断计时频率和扫描频率。这里我的触发沿用的频率是128HZ，计时频率为1HZ，扫描频率为4HZ。计时采用对1秒的信号源进行计数并向高位进位的思想。在调时间功能中需要SW14开关开启和SW12开关关闭。

源程序：

```
//时间变化和设置当前时间
always @(posedge clk[0])//128HZ信号的上升沿作触发
begin
    if (clk[7:0] == 8'b1111_1110)begin//时间变化，1s作触发
            second = second + 1;
            minute = minute + second / 60;
            second = second % 60;
            hour = hour + minute / 60;
            minute = minute % 60;
            hour = hour % 24;
    end
    if (clk[5:0] == 6'b11_1110) begin//按0.25s的速度扫描按钮，设置时间
            if (SW[14] & ~SW[12] & BNT[0]) begin//时+1
                    hour = (hour + 1) % 24;
            end
            else if (SW[14] & ~SW[12] & BNT[2]) begin//时-1
                    hour = (hour - 1);
                    if (hour >= 24) hour = 23;
            end
           else if (SW[14] & ~SW[12] & BNT[1]) begin//分+1
                    minute = (minute + 1) % 60;
            end
            else if (SW[14] & ~SW[12] & BNT[3]) begin//分-1
                    minute = (minute - 1);
                    if (minute >= 60) minute = 59;
            end
            else if (SW[14] & ~SW[12] & BNT[4]) begin//复位
                    second=0;
                    hour=0;
                    minute=0;
            end
    end
end
```

Vivado综合得电路原理图




alerh[6:0]
alern[6:0]
second_i__0
second_i__1
second_i__2
second_i__3
second_i__4
second1_i__1
second1_i__2
second1_i__3
minute_i__2
minute_i__3
minute0_i__0
hour_i__0
hour1_i__1
hour_i__3
hour_i__4
hour_i__2
hour1_i__0
hour0_i__0
hour_i__5
hour_i__6
second_i__7
minute_i__4
minute_i__5
minute_i__6
second_i__8
second_i__9
second_i__10
second_i__11
RTL_MUX
RTL_AND
RTL_MOD
RTL_GEQ
RTL_ADD
RTL_REG
hour_reg[6:0]
minute_reg[6:0]
second_reg[6:0]
hour[6:0]
minute[6:0]
second[6:0]

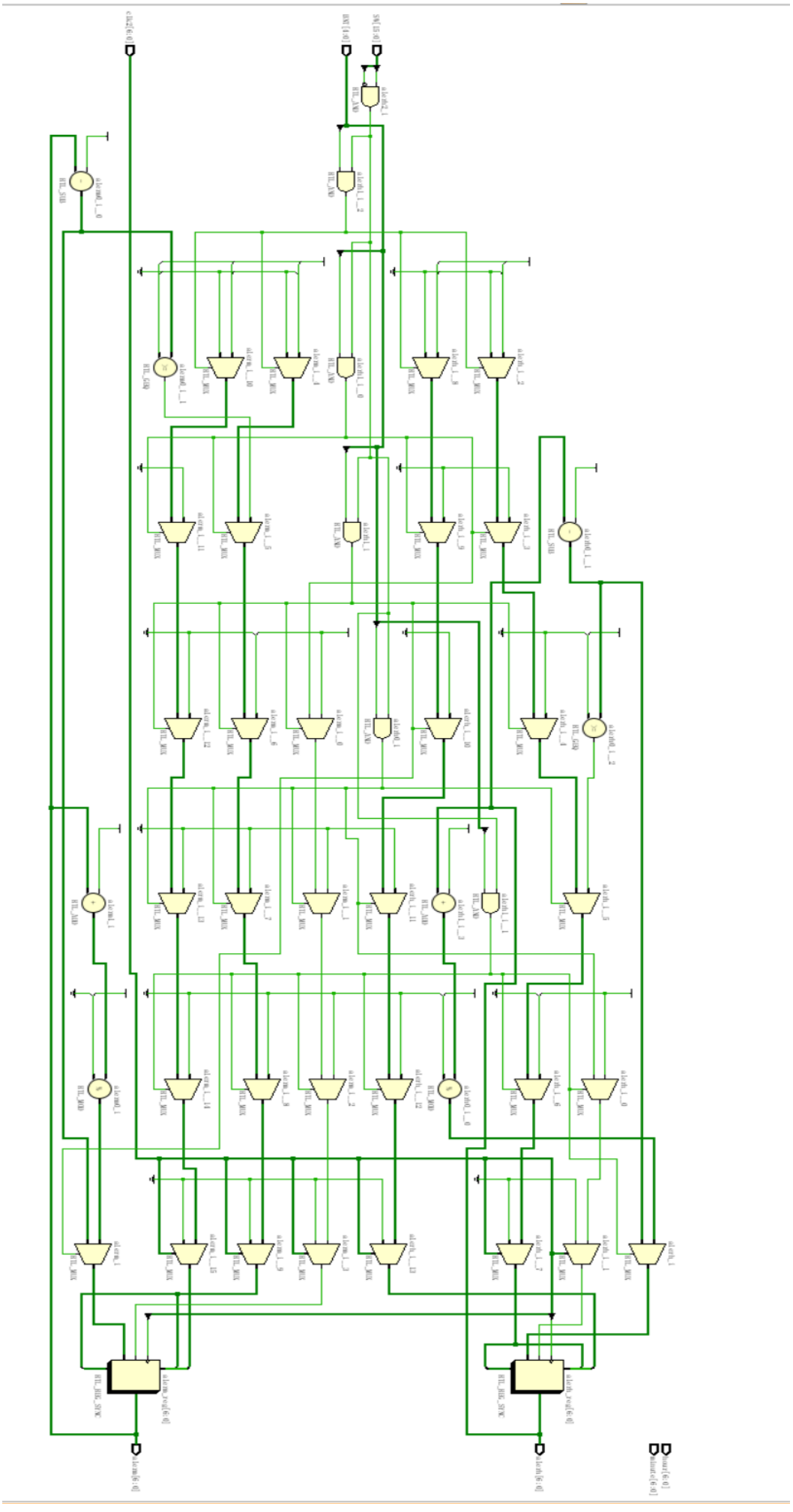
4.闹钟设置模块

此模块就是通过按钮来调整闹钟时间。需要关闭 SW14 开关和打开 SW12 开关才能成功进行。原理与给正常时间按钮调时的原理一样。

源程序：

```
always @(posedge clk2[0])//128HZ触发
begin

    if (clk2 == 6'b11_1110)begin//0.25s扫描

        if (SW[12] & ~SW[14] & BNT[0])begin//闹钟时+1
            alerh = (alerh +1) % 24;
        end

        else if (SW[12] & ~SW[14] & BNT[2])begin//闹钟时-1
            alerh = (alerh - 1);
            if (alerh >=24 )alerh = 23;
        end

        else if (SW[12] & ~SW[14] & BNT[1])begin//闹钟分+1
            alerm = (alerm + 1) % 60;
        end

        else if (SW[12] & ~SW[14] & BNT[3])begin//闹钟分-1
            alerm = (alerm - 1 );
            if (alerm >=60 ) alerm = 59;
        end

        else if (SW[12] & ~SW[14] & BNT[4])begin//复位清0
            alerm = 0;
            alerh = 0;
        end

    end

end
```

Vivado综合得电路原理图



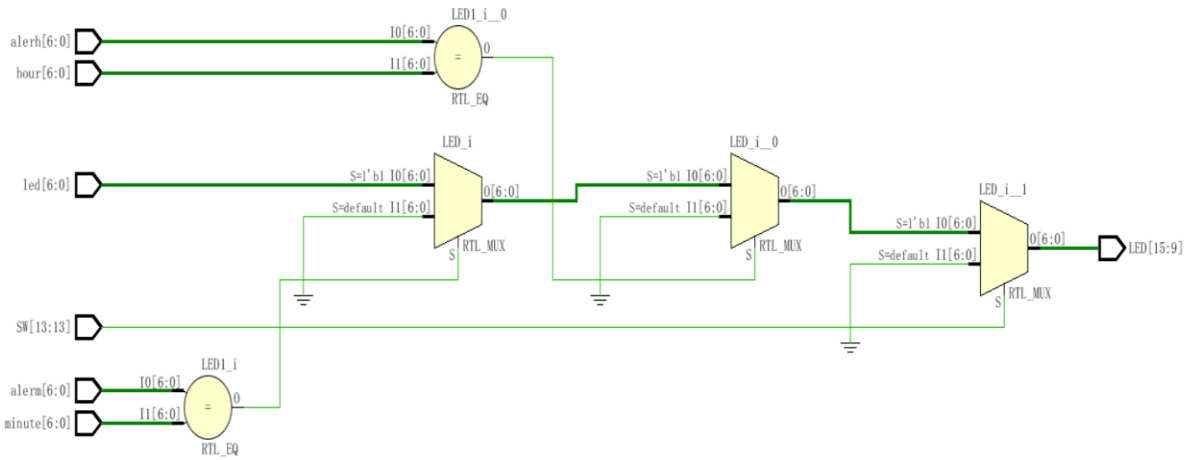
5.闹钟报时模块

若打开 SW13 开关，则会开启闹钟功能。当达到闹钟设置时间时，LED15-LED9 这 7 盏灯会以呼吸灯的形式闪烁，其余情况都是灭灯状态。这一部分是组合逻辑电路。

源程序：

```
//当前时间达到闹钟设定的时间则亮呼吸灯
always @(*)
begin
    if (SW[13])begin//若开启了闹钟功能
        if (alerh==hour)begin
            if (alerm==minute)begin
                LED[15:9]=led[6:0];//led是存放呼吸灯状态的寄存器
            end
            else  LED[15:9]=7'b000_0000;//其余情况不亮灯，下同
        end
        else  LED[15:9]=7'b000_0000;
    end
    else LED[15:9]=7'b000_0000;
end
```

Vivado综合得电路原理图



6.创新内容：呼吸灯产生模块

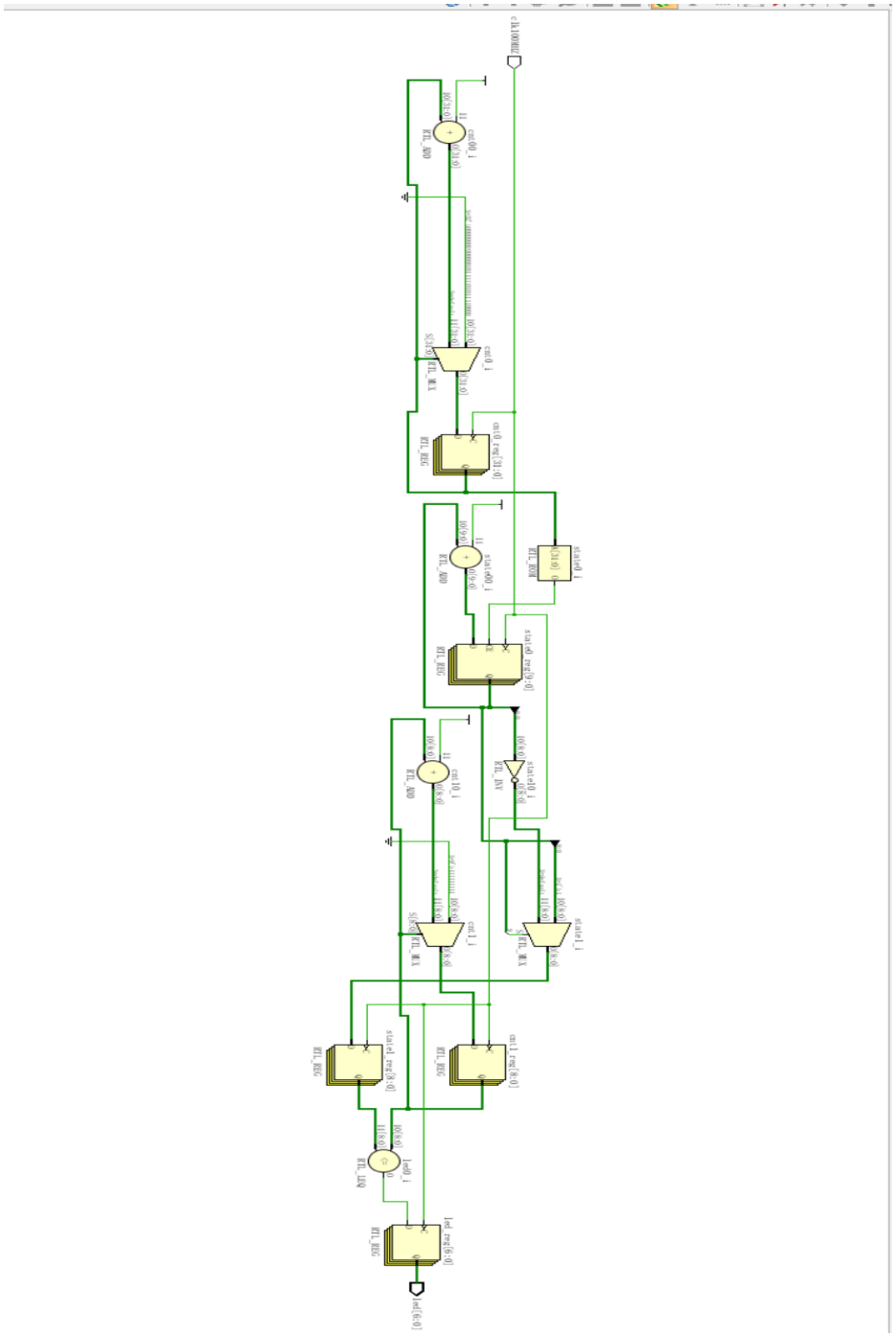
呼吸灯分为两个部分，一部分是由亮变暗，一部分是由暗变亮。我们知道，如果单纯地变亮变暗是不会有渐变的效果。我们把变化时间分成 n 等份在我的程序里 n 为 2^10。在这 n 等份中，一半时间是渐亮，一半时间渐暗。即在我的程序中有 2^9 份时间是由亮变暗，有 2^9 份时间是由暗变亮。在每一份时间中，把通过高电平的时间（即占空比）设置成亮灯，通过低电平的时间设置为灭灯，并且占空比能够每过一定的时间为就增大或减小，由于变化时间较快，在肉眼看来就是渐亮和渐暗的效果。这个过程实际上就是一个有限状态机。有 2^10 个状态循环变化，每个状态对应一定时间的占空比。

源程序：

```
//////////////////////////////////呼吸灯
parameter FREQUENCE=100_000_000;
parameter WIDTH=9;
reg [WIDTH:0] state0;//最高位控制占空比增大还是减小，后8位为占空比状态
reg [WIDTH-1:0] state1;//当前占空比状态
reg[31:0]cnt0;//时间计数器
reg [6:0] led;
//控制每个占空比时间
always@(posedge clk100MHZ)
begin
    if (cnt0==(FREQUENCE/(2**WIDTH)))
    begin
        cnt0<=0;
        state0 <= state0 + 1'b1;
    end
    else begin
        cnt0 <= cnt0 + 1'b1;
    end
end
//控制占空比增大与减小
always@(posedge clk100MHZ)
begin
    if(state0[WIDTH])state1<=state0[WIDTH-1:0];//增大
    else state1<=~state0[WIDTH-1:0];//减小

end
//生成与state1进行大小比较的计数器cnt1
wire [WIDTH-1:0] time_over;
assign time_over={WIDTH{1'b1}};//总时间
reg[WIDTH-1:0] cnt1;
always@(posedge clk100MHZ)
begin
    if (cnt1==time_over)cnt1<=0;
```

```
    else cnt1<=cnt1+1'b1;
end
//计数器cnt1与state1进行大小比较，以使led脉冲的占空比实现渐变
always @(posedge clk100MHZ)
begin
    if (cnt1<=state1)
        led[6:0]<= 7'b111_1111;//占空比时间内亮灯
     else led[6:0]<=7'b000_0000;//非占空比时间内灭灯
end
```

Vivado综合得电路原理图



七、感想与感悟

通过这次对时钟的设计，我对理论课上模糊不清的 verilog 语言有了一个比较清晰的了解。除了熟悉了其基本语法之外，我也了解到它和我们平时使用的 C++语言在编程思想上的不同之处。比如，在 verilog 中，寄存器只能在一个 always 块中作为赋值语句的左值。而我们在 C++中却可以在多个不同的函数中对同一个变量进行赋值。究其原因，是因为 verilog 是硬件描述语言，我们要错电路的角度去思考问题，一个寄存器的输入端只能有一个信号来源，但是其输出端的值可以输送到多个地方，这就是导致这个不同的地方。所以我觉得对我来说思考方式上的改变，是这次硬件编程较大的挑战。我在实际编码中也的确遇到了许多问题。比如在改变时间的值上这块，既要让他按照正常时间 1s 变化一次，又要根据按钮变化一次。如果我们使用 1s 的 clk 作为触发沿，那么就会导致按钮功能失灵，因为我们需要一个较快的时间频率去扫描按钮。通过不断改写代码和试验，我终于找到了解决方法，就是，将时钟存在一个多位的寄存器里，令最高位为 1HZ，这样，最低位就是一个比较快的频率，以这个较快的频率作为触发沿，再在 always 块中判断是否达到 1s 来控制时间变化，至于按钮，则可以用最高位和最低位之间的某一位来控制。在这个不断遇到问题，不断解决问题的过程中，我觉得自己的技能得到了很好的提升，这个过程也让我觉得很有趣。总之，我从这次设计中学到很多东西，也巩固了我的理论学习。