奋斗的小孩之 altera 系列

第二十七篇 呼吸灯

对于每一个的小实验，我们都可以把它看作是一个小项目，逐步的去分析，设计，调试，最后完成功能。下面我们就开始我们的“小项目”。

项目名称：呼吸灯

具体要求：led 灯在由亮到暗以及由暗到亮的逐渐变化。

项目分析：

1. 要求分析

灯光在微电脑的控制之下完成由亮到暗的逐渐变化，感觉好像是人在呼吸。根据不同人群以及不用情况下的呼吸频率不同，笔者这里采取一分钟呼吸 15 次，呼气 2 秒钟，吸气 2 秒钟。

2. 实现原理

由于 LED 的亮度与流过的电流成正比，如果能够去控制流经 LED 的电流，使电流在 2 秒钟内从 i=0A(0A：0 安培)逐渐递增到 i=XA（XA：一定的电路强度），就实现了 LED 的由暗到亮的逐渐变化；使电流在 2 秒钟内从 i=XA 逐渐递增到 i=0A 就实现了 LED 的由亮到暗的逐渐变化。以此为周期，不断地变化下

去，就实现了呼吸灯的功能。

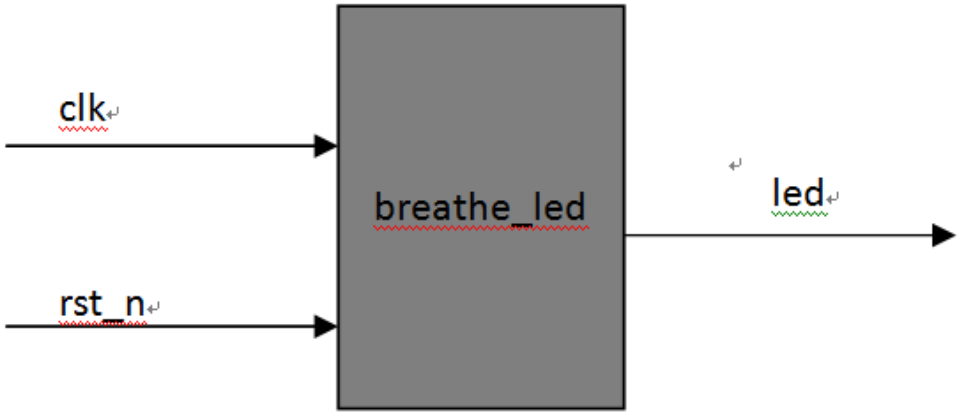
在数字电路中，控制电流的大小几乎是不能实现的。如果将 LED 在 T/2 内点亮，将 LED 在 T/2 内熄灭，我们会看到什么呢？人的眼睛有视觉暂留（人眼在观察景物时，光信号传入大脑神经，需经过一段短暂的时间，光的作用结束后，视觉形象并不立即消失，这种残留的视觉称“后像”，视觉的这一现象则被称为“视觉暂留”），若 T 的时间足够短，那么将会看到一直亮着的 LED，亮度由点亮的时间所占 T 的比例决定的。

3. 实现方案

一分钟呼吸 15 次，呼气 2 秒钟，吸气 2 秒钟。我们先做由暗到亮，将 2 秒钟分成 1000 个时间段，每个时间段 2ms。第一个时间段全部熄灭；第二个时间段 2us 的时间点亮，其余时间熄灭；第三个时间段 4us 的时间点亮，其余时间熄灭 ••••••

每个时间段为 2ms，相邻两个时间段增加 2us，可以理解为点亮的比例每个时间段增加千分之一。这样的话，就可以完成由暗到亮。将上述的输出波形取反，就是点亮的比例每个时间段减少千分之一，完成由亮到灭。

架构图如下：



系统设计：

1. 工程的名称：breathe_led
2. 实现方法：用计数器 1 实现 2us 的计时，用计数器 1 驱动计数器 2，实现 2ms 的计数。用计数器 1 和计数器 2 驱动计数器 3，实现 2s 的计数。用计数器 2 和计数器 3 进行比较，来完成占空比的调节。计数器 3 每增加 1，计数器 2 都会从 0 增加到 999，如果将计数器 3 数值大于计数器 2 数值的时间段，当作点亮的时间，那么就可以实现由暗到亮的过程了。

设计代码如下：

```
/*
模块名称：breathe_led
模块功能：呼吸灯
编写时间：2016-09-01
```

```
    作者：至芯科技----奋斗的小孩
    邮箱：zxopenhxs@126.com
    */
module breathe_led (clk, rst_n, led);

    input clk;//50MHz，周期为20ns
    input rst_n;

    output led;

    parameter T2us = 100;
    parameter T2ms = 1000;
    parameter T2s = 1000;

    reg [6:0] cnt1;//2us的计数器

  always @ (posedge clk or negedge rst_n)
     begin
        if (!rst_n)
           begin
              cnt1 <= 7'd0;
```

```
            end
        else
            begin
                if (cnt1 < T2us - 1)
                    begin
                        cnt1 <= cnt1 + 1'b1;
                    end
                else
                    begin
                        cnt1 <= 7'd0;
                    end
            end
    end

wire flag;//计时到 2us 时，产生一个同步脉冲

assign flag = (cnt1 == T2us - 1) ? 1'b1 : 1'b0;

reg [9:0] cnt2;//2ms 计数器

always @ (posedge clk or negedge rst_n)
```

```
begin
    if (!rst_n)
        begin
            cnt2 <= 10'd0;
        end
    else
        begin
            if ((cnt2 < T2ms - 1)&&(flag))
                begin
                    cnt2 <= cnt2 + 1'b1;
                end
            else
                begin
                    if (flag)
                        begin
                            cnt2 <= 10'd0;
                        end
                    else
                        begin
                            cnt2 <= cnt2;
                        end
```

```
                end
            end
        end

reg [9:0] cnt3;//2s 的计数器
reg change;//控制占空比由大变小或者由小变大

always @ (posedge clk or negedge rst_n)
    begin
        if (!rst_n)
            begin
                cnt3 <= 10'd0;
                change <= 1'b0;
            end
        else
            begin
            if ((cnt3 < T2s - 1)&&(cnt2 == T2ms - 1) && (flag))
                    begin
                        cnt3 <= cnt3 + 1'b1;
                    end
                else
```

```
                begin
                    if ((cnt2 == T2ms - 1) && (flag))
                        begin
                            change <= ~change;
                            cnt3 <= 10'd0;
                        end
                    else
                        begin
                            cnt3 <= cnt3;
                        end
                end
            end
        end

reg led_n;

always @ (posedge clk or negedge rst_n)
    begin
        if (!rst_n)
            begin
                led_n <= 1'b0;
```

```
            end
        else
            begin
                if (cnt3 > cnt2)
                    begin
                        led_n <= 1'b0;
                    end
                else
                    begin
                        led_n <= 1'b1;
                    end
            end
    end

  assign led = (change == 1) ? ~led_n : led_n;

endmodule
```

代码解析：

1. assign led = (change == 1) ? ~led_n : led_n;

代表：当 change=1 时，led=~led_n，当 change=0 时，led=led_n。

2. cnt2 清零时，必须等待 flag。如果没有 flag，那么 cnt2 的最后一个数值将只会存在一个周期。cnt3 同理。
3. 当 cnt3 计时到最后一刻时，将 change 取反，输出的占空比变化情况发生改变。

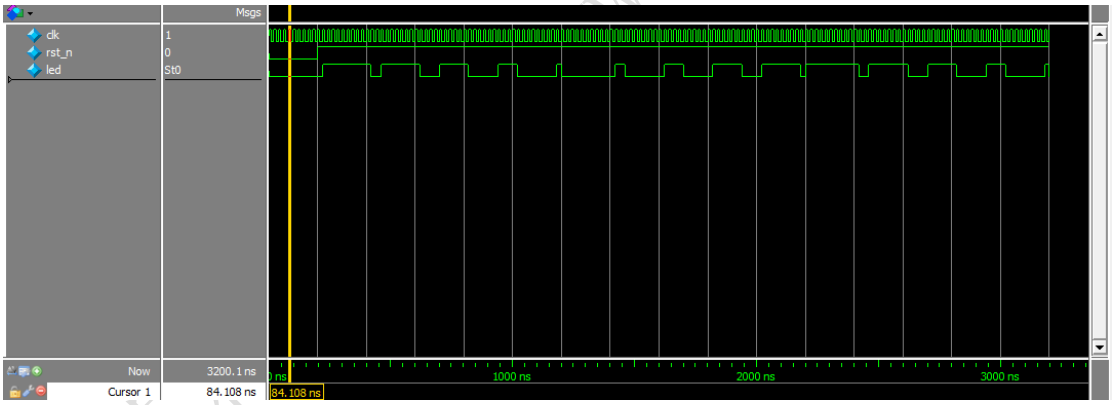
激励代码如下：

```
/*
模块名称：breathe_led_tb
模块功能：为 breathe_led 模块提供激励信号
编写时间：2016-09-01
作者：至芯科技----奋斗的小孩
邮箱：zxopenhxs@126.com
*/
`timescale 1ns/1ps

module breathe_led_tb;

    reg clk;
    reg rst_n;

    wire led;
```

```
parameter T2us = 2;//仿真时，时间缩短
parameter T2ms = 5;
parameter T2s = 5;

initial begin
   clk = 1'b1;
   rst_n = 1'b0;
   # 200.1
   rst_n = 1'b1;
   # 3000
   $stop;
end

always # 10 clk = ~clk;//50MHz

breathe_led
   #(
      .T2us(T2us),//传递仿真参数
      .T2ms(T2ms),
      .T2s(T2s)
```

```
    )
    breathe_led_dut(
        .clk(clk),
        .rst_n(rst_n),
        .led(led)
    );

endmodule
```

仿真波形如下：



led的占空比从大到小，再从小到大，一直不断地变化下去。

如果设计要求或者本地晶振与笔者的设计不同，请自行更改设计，以保证设计的正确性。如果还是有不明白的读者可以发邮件到我邮箱或者加群询问。

制作人:奋斗的小孩 fpga 交流群:282124839