PWM 流水灯

--作者：小黑同学

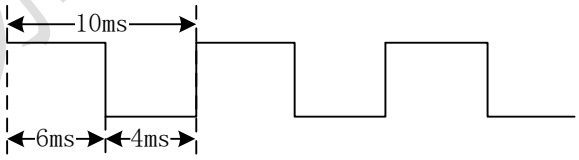
1.1 总体设计

1.1.1 概述

脉冲宽度调制技术（Pulse Width Modelation，PWM）是利用微处理器/FPGA 的数字输出对模拟电路进行控制的一种有效技术，其广泛应用于测量、通信、功率控制与变换等众多领域。PWM 数字信号从处理器到被控系统都采用数字形式，无需进行数模转换。航模中的控制信号大多是 PWM 信号，比如 FUTABA、JR 等舵机的控制都采用 PWM 方式，发射机给接收机输送脉冲后接收机就会控制舵机进行旋转。举个例子，假定基础脉宽是 100ms，当发射机的脉宽增大（如增加到 150ms）时接收机就控制舵机正向旋转；反之发射机的脉宽减小（如减小到 50ms）时，接收机就控制舵机逆向旋转。

PWM 是一种对模拟信号电平进行数字编码的方法。通过使用高分辨率计数器，对方波的占空比进行调制，从而对一个具体模拟信号的电平进行编码。由于在给定的任何时刻，满幅值的直流供电只存在有（ON）和无（OFF）两种状态，因此 PWM 信号仍然是数字信号。电压或电流源是以一种通（ON）或断（OFF）的重复脉冲序列被加到模拟负载上去的。直流供电被加到负载上的时候为“通”，负载供电被断开的时候为“断”。只要有足够的带宽，任何模拟值都可以使用 PWM 进行编码。

通俗来说，PWM 是连续的、具有一定比例占空比的脉冲信号，可以通过控制占空比来对其进行改变。简单来说，可以认为 PWM 就是一种方波，如图所示是一个周期为 10ms，高电平为 6ms，低电平时间为 4ms 的 PWM，其占空比（高电平时间占整个周期的比例）为 60%



PWM 波形图

发光二极管简称为 LED，是一种常用的发光器件，通过电子与空穴复合释放能量发光，它在照明领域应用广泛。发光二极管可高效的将电能转化为光能，在现代社会具有广泛的用途，如照明、平板显示、医疗器件等。可通过高低电平的变化来控制 LED 灯的明灭状态，当输出信号为低电平时，LED 灯亮，反之，当输出信号为高电平时，LED 灯灭。因此可通过改变 PWM 波形的占空比，来控制 LED 灯亮灭的时间，当有多个 LED 灯的时候，通过不同的 PWM 波形来控制不同的 LED 灯，就可以产生各种各样不同的效果。

1.1.2 设计目标

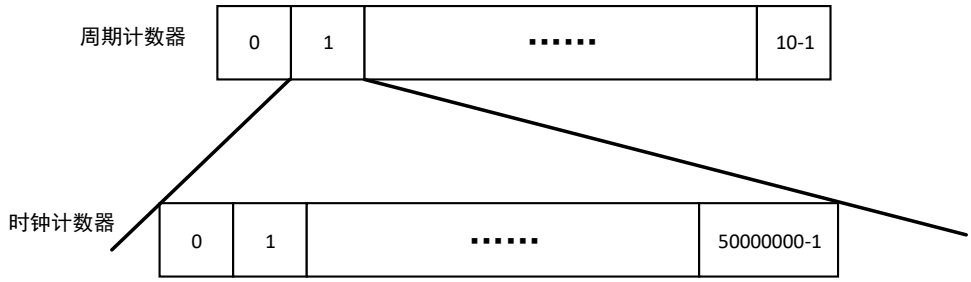
本模块产生 8 个不同的 PWM 脉冲，控制 8 个 LED 灯点亮不同的时间，从而达到流水灯的效果。每个脉冲周期为 10s，占空比从 10%~80%。上电后，led0 点亮 1s，熄灭 9s；再点亮 1s，熄灭 9s，……，依次不断循环。led1~led7 与 led0 类似，分别点亮 2s~8s，其他时候都是熄灭的。

1.1.3 信号列表

信号名	接口方向	定义
clk	输入	系统时钟，50Mhz
rst_n	输入	低电平复位信号
led	输出	8 位 led 灯输出信号

1.1.4 设计思路

根据题目功能要求可知，产生的 8 个不同的 PWM 脉冲的周期都是 10 秒（s），由此我们可以提出两个计数器的架构，如下图所示。



该架构由两个计数器组成：时钟计数器 cnt_1s 和周期计数器 cnt_10s。

时钟计数器 cnt_1s：用于计算 1 秒的时钟个数，加一条件为 1，表示一直计数；结束条件为 50 000000，表示数到 1 秒就清零。

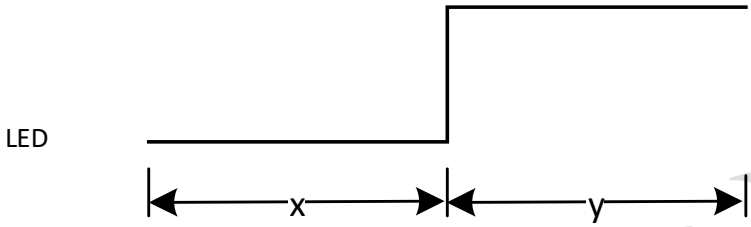
周期计数器 cnt_10s：用于对 1 秒进行计数，加一条件为 end_cnt_1s，表示数到 1 秒就加 1；结束条件为 10，表示数完 10 秒就清零。

至此，我们就将这个练习的计数器架构设计出来了。下面是计数器的代码。

```
always @(posedge clk or negedgerst_n)begin
    if(!rst_n)begin
        cnt_1s <= 0;
    end
    else if(add_cnt_1s)begin
        if(end_cnt_1s)
            cnt_1s <= 0;
        else
            cnt_1s <= cnt_1s + 1;
    end
end

assign add_cnt_1s = 1;
assign end_cnt_1s = add_cnt_1s && cnt_1s==50_000_000 -1 ;




always @(posedge clk or negedgerst_n)begin
    if(!rst_n)begin
        cnt_10s <= 0;
    end
    else if(add_cnt_10s)begin
        if(end_cnt_10s)
            cnt_10s <= 0;
        else
            cnt_10s <= cnt_10s + 1;
    end
```

```
end

assign add_cnt_10s = end_cnt_1s;
assign end_cnt_10s = add_cnt_10s && cnt_10s==10-1 ;
```

下面我们开始设计输出信号 led，在设计这些信号的时候，我们只需要关注他的变化点，也就是 0 变 1、1 变 0 的点。



以 led0 的变化为例，上电后，led[0]点亮 1s，熄灭 9s，再点亮 1s，熄灭 9s。也就是数到 1 秒时，即 add_cnt_10s && cnt_10s==1-1 时，led0 由 0 变 1。当数到 10 秒时，即 end_cnt_10s 时，led[0]由 1 变 0。

Led[1]~led[7]也是同样，即：

上电后，led[1]点亮 2s，熄灭 8s，再点亮 2s，熄灭 8s。也就是数到 2 秒时，即 add_cnt_10s && cnt_10s==2-1 时，led[1]由 0 变 1。当数到 10 秒时，即 end_cnt_10s 时，led[1]由 1 变 0。

上电后，led[2]点亮 3s，熄灭 7s，再点亮 3s，熄灭 7s。也就是数到 3 秒时，即 add_cnt_10s && cnt_10s==3-1 时，led[2]由 0 变 1。当数到 10 秒时，即 end_cnt_10s 时，led[2]由 1 变 0。

上电后，led[3]点亮 4s，熄灭 6s，再点亮 4s，熄灭 6s。也就是数到 4 秒时，即 add_cnt_10s && cnt_10s==4-1 时，led[3]由 0 变 1。当数到 10 秒时，即 end_cnt_10s 时，led[3]由 1 变 0。

上电后，led[4]点亮 5s，熄灭 5s，再点亮 5s，熄灭 5s。也就是数到 5 秒时，即 add_cnt_10s && cnt_10s==5-1 时，led[4]由 0 变 1。当数到 10 秒时，即 end_cnt_10s 时，led[4]由 1 变 0。

上电后，led[5]点亮 6s，熄灭 4s，再点亮 6s，熄灭 4s。也就是数到 6 秒时，即 add_cnt_10s && cnt_10s==6-1 时，led[5]由 0 变 1。当数到 10 秒时，即 end_cnt_10s 时，led[5]由 1 变 0。

上电后，led[6]点亮 7s，熄灭 3s，再点亮 7s，熄灭 3s。也就是数到 7 秒时，即 add_cnt_10s && cnt_10s==7-1 时，led[6]由 0 变 1。当数到 10 秒时，即 end_cnt_10s 时，led[6]由 1 变 0。

上电后，led[7]点亮 8s，熄灭 2s，再点亮 8s，熄灭 2s。也就是数到 8 秒时，即 add_cnt_10s && cnt_10s==8-1 时，led[7]由 0 变 1。当数到 10 秒时，即 end_cnt_10s 时，led[7]由 1 变 0。

由此便可以将所有的 led 信号设计出来。下面是 led 的代码。

```
always   @(posedge clk or negedgerst_n)begin
    if(rst_n==1'b0)begin
        led[0] <= 0;
    end
    else if(end_cnt_10s)begin
        led[0] <= 0;
    end
    else if(add_cnt_10s && cnt_10s==1-1)begin
```

```
            led[0] <= 1;
        end
end




always    @(posedge clk or negedgerst_n)begin
        if(rst_n==1'b0)begin
            led[1] <= 0;
        end
        else if(end_cnt_10s)begin
            led[1] <= 0;
        end
        else if(add_cnt_10s && cnt_10s==2-1)begin
            led[1] <= 1;
        end
end




always    @(posedge clk or negedgerst_n)begin
        if(rst_n==1'b0)begin
            led[2] <= 0;
        end
        else if(end_cnt_10s)begin
            led[2] <= 0;
        end
        else if(add_cnt_10s && cnt_10s==3-1)begin
            led[2] <= 1;
        end
end




always    @(posedge clk or negedgerst_n)begin
        if(rst_n==1'b0)begin
            led[3] <= 0;
        end
        else if(end_cnt_10s)begin
```

```
            led[3] <= 0;
        end
        else if(add_cnt_10s && cnt_10s==4-1)begin
            led[3] <= 1;
        end
end




always    @(posedge clk or negedgerst_n)begin
        if(rst_n==1'b0)begin
            led[4] <= 0;
        end
        else if(end_cnt_10s)begin
            led[4] <= 0;
        end
        else if(add_cnt_10s && cnt_10s==5-1)begin
            led[4] <= 1;
        end
end




always    @(posedge clk or negedgerst_n)begin
        if(rst_n==1'b0)begin
            led[5] <= 0;
        end
        else if(end_cnt_10s)begin
            led[5] <= 0;
        end
        else if(add_cnt_10s && cnt_10s==6-1)begin
            led[5] <= 1;
        end
end




always    @(posedge clk or negedgerst_n)begin
        if(rst_n==1'b0)begin
```

```
            led[6] <= 0;
        end
        else if(end_cnt_10s)begin
            led[6] <= 0;
        end
        else if(add_cnt_10s && cnt_10s==7-1)begin
            led[6] <= 1;
        end
end




always    @(posedge clk or negedgerst_n)begin
        if(rst_n==1'b0)begin
            led[7] <= 0;
        end
        else if(end_cnt_10s)begin
            led[7] <= 0;
        end
        else if(add_cnt_10s && cnt_10s==8-1)begin
            led[7] <= 1;
        end
end
```

1.1.5 参考设计代码

```
module pwmled(
        clk       ,
        rst_n     ,
        led
);

input                   clk           ;
input                   rst_n         ;
output     [ 7:0]       led           ;
reg        [25:0]       cnt_1s        ;
wire                    add_cnt_1s ;
```

```
wire                end_cnt_1s ;
reg       [ 7:0]    cnt_10s    ;
wire                add_cnt_10s;
wire                end_cnt_10s;
reg       [ 7:0]    led        ;




always @(posedge clk or negedgerst_n)begin
    if(!rst_n)begin
        cnt_1s <= 0;
    end
    else if(add_cnt_1s)begin
        if(end_cnt_1s)
            cnt_1s <= 0;
        else
            cnt_1s <= cnt_1s + 1;
    end
end

assign add_cnt_1s = 1;
assign end_cnt_1s = add_cnt_1s && cnt_1s==50_000_000 -1 ;




always @(posedge clk or negedgerst_n)begin
    if(!rst_n)begin
        cnt_10s <= 0;
    end
    else if(add_cnt_10s)begin
        if(end_cnt_10s)
            cnt_10s <= 0;
        else
            cnt_10s <= cnt_10s + 1;
    end
end

assign add_cnt_10s = end_cnt_1s;
assign end_cnt_10s = add_cnt_10s && cnt_10s==10-1 ;
```

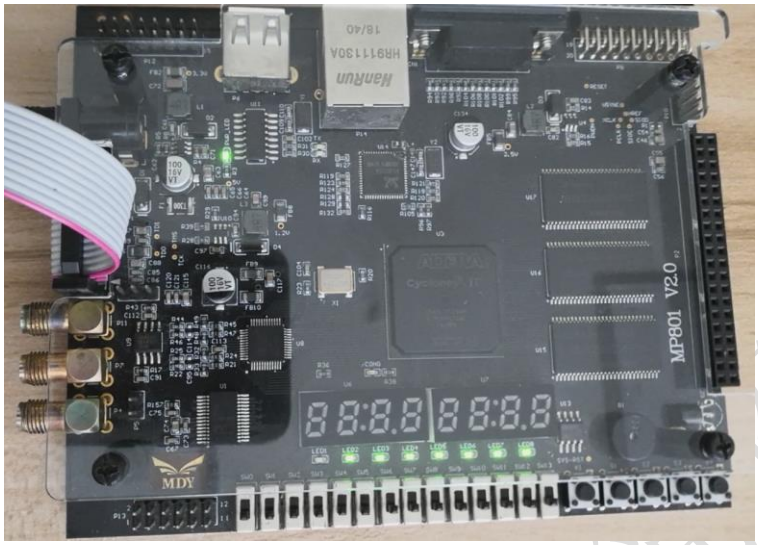
```
always   @(posedge clk or negedgerst_n)begin
      if(rst_n==1'b0)begin
            led[0] <= 0;
      end
      else if(end_cnt_10s)begin
            led[0] <= 0;
      end
      else if(add_cnt_10s && cnt_10s==1-1)begin
            led[0] <= 1;
      end
end




always   @(posedge clk or negedgerst_n)begin
      if(rst_n==1'b0)begin
            led[1] <= 0;
      end
      else if(end_cnt_10s)begin
            led[1] <= 0;
      end
      else if(add_cnt_10s && cnt_10s==2-1)begin
            led[1] <= 1;
      end
end




always   @(posedge clk or negedgerst_n)begin
      if(rst_n==1'b0)begin
            led[2] <= 0;
      end
      else if(end_cnt_10s)begin
            led[2] <= 0;
      end
      else if(add_cnt_10s && cnt_10s==3-1)begin
            led[2] <= 1;
```

```
    end
end




always   @(posedge clk or negedgerst_n)begin
    if(rst_n==1'b0)begin
        led[3] <= 0;
    end
    else if(end_cnt_10s)begin
        led[3] <= 0;
    end
    else if(add_cnt_10s && cnt_10s==4-1)begin
        led[3] <= 1;
    end
end




always   @(posedge clk or negedgerst_n)begin
    if(rst_n==1'b0)begin
        led[4] <= 0;
    end
    else if(end_cnt_10s)begin
        led[4] <= 0;
    end
    else if(add_cnt_10s && cnt_10s==5-1)begin
        led[4] <= 1;
    end
end




always   @(posedge clk or negedgerst_n)begin
    if(rst_n==1'b0)begin
        led[5] <= 0;
    end
    else if(end_cnt_10s)begin
        led[5] <= 0;
```

```
    end
    else if(add_cnt_10s && cnt_10s==6-1)begin
        led[5] <= 1;
    end
end




always   @(posedge clk or negedgerst_n)begin
    if(rst_n==1'b0)begin
        led[6] <= 0;
    end
    else if(end_cnt_10s)begin
        led[6] <= 0;
    end
    else if(add_cnt_10s && cnt_10s==7-1)begin
        led[6] <= 1;
    end
end




always   @(posedge clk or negedgerst_n)begin
    if(rst_n==1'b0)begin
        led[7] <= 0;
    end
    else if(end_cnt_10s)begin
        led[7] <= 0;
    end
    else if(add_cnt_10s && cnt_10s==8-1)begin
        led[7] <= 1;
    end
end
endmodule
```

1.2 效果和总结

- 时间经过 1 秒之后



- 时间经过 4 秒的时候

- **时间经过 8 秒的时候**

看上面的现象，可以发现，跟设计目标要求的都是一样的，成功完成设计目标。

我们的至简设计法和计数器模板在以上的设计中发挥了至关重要的作用，使设计能够快速准确完成。本设计中的 led 信号的代码为了方便理解，把 8 个 led 灯分开写了，希望有兴趣的同学可以将 led 的代码进行简化，合成一个 always，想不到也没关系，在讲解视频中也有讲述如何实现。

感兴趣的朋友也可以访问明德扬论坛（http://www.fpgabbs.cn/）进行 FPGA 相关工程设计学习，也欢迎大家在评论与我进行讨论！