

无源蜂鸣器驱动设计

小梅哥编写，可适用于芯航线 FPGA 学习套件，作者保留一切所有权

2016 年 7 月 16 日星期六

蜂鸣器是一种一体化结构的电子讯响器，采用直流电压供电，广泛应用于计算机、打印机、复印机、报警器、电子玩具、汽车电子设备、电话机、定时器等电子产品中作发声器件。蜂鸣器主要分为压电式蜂鸣器和电磁式蜂鸣器两种类型。

压电式蜂鸣器：压电式蜂鸣器主要由多谐振荡器、压电蜂鸣片、阻抗匹配器及共鸣箱、外壳等组成。有的压电式蜂鸣器外壳上还装有发光二极管。

多谐振荡器由晶体管或集成电路构成。当接通电源后（1.5~15V 直流工作电压），多谐振荡器起振，输出 1.5~2.5kHz 的音频信号，阻抗匹配器推动压电蜂鸣片发声。

电磁式蜂鸣器：电磁式蜂鸣器由振荡器、电磁线圈、磁铁、振动膜片及外壳等组成。

接通电源后，振荡器产生的音频信号电流通过电磁线圈，使电磁线圈产生磁场。振动膜片在电磁线圈和磁铁的相互作用下，周期性地振动发声。

根据蜂鸣器本身是否集成了震荡源，蜂鸣器可以分为有源蜂鸣器与无源蜂鸣器。

有源蜂鸣器直接接上额定电源(新的蜂鸣器在标签上都有注明)就可连续发声；而无源蜂鸣器则和电磁扬声器一样，需要接在音频输出电路中才能发声。

有源蜂鸣器与无源蜂鸣器的区别：

注意：这里的“源”不是指电源，而是指震荡源。

也就是说，有源蜂鸣器内部带震荡源，所以只要一通电就会叫；

而无源内部不带震荡源，所以如果用直流信号无法令其鸣叫。必须用 2K-5K 的方波去驱动它

有源蜂鸣器往往比无源的贵，就是因为里面多个震荡电路。

无源蜂鸣器的优点是：

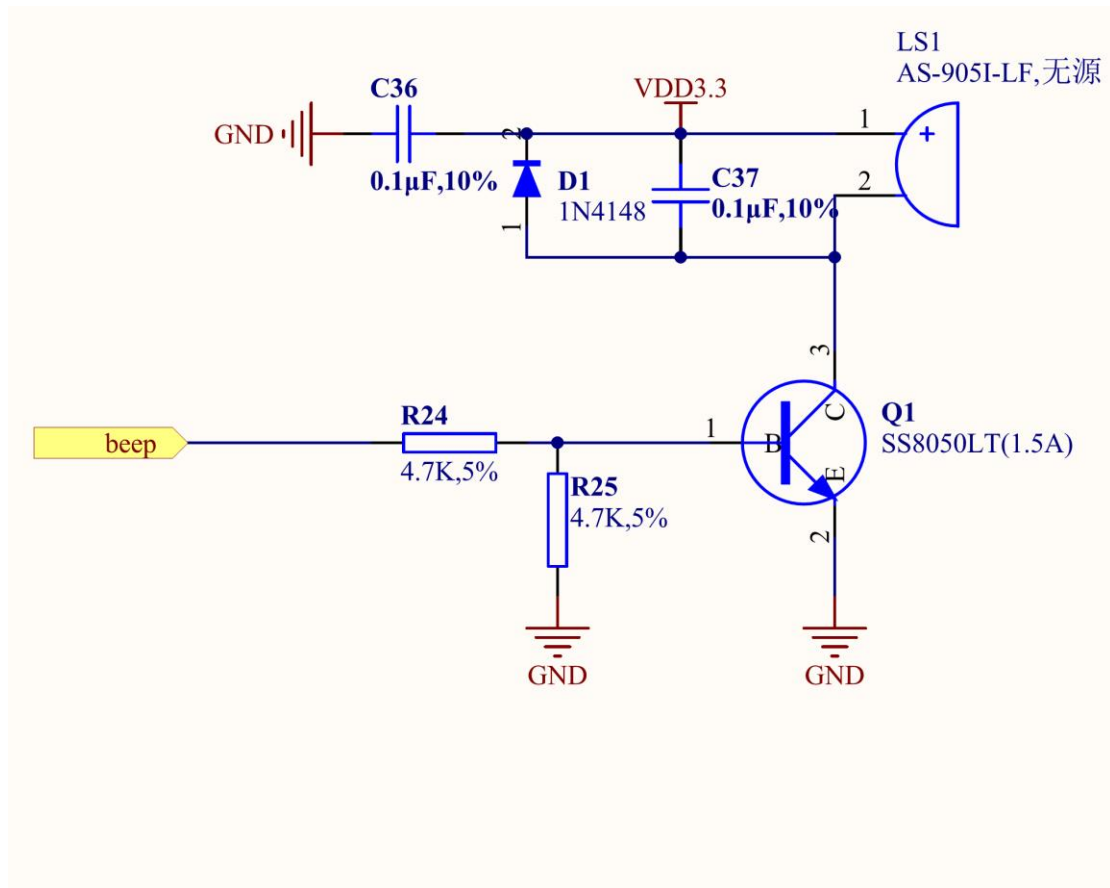
1. 便宜
2. 声音频率可控，可以做出“多来米发索拉西”的效果
3. 在一些特例中，可以和 LED 复用同一个控制口

有源蜂鸣器的优点是：程序控制方便。

以上介绍了蜂鸣器的种类以及有源蜂鸣器、无源蜂鸣器的特点。接下来，我们将介绍芯航线 FPGA 学习套件主板上使用的蜂鸣器电路，并使用 Verilog 设计一个蜂鸣器驱动电路，来驱动蜂鸣器发声。

蜂鸣器电路介绍

芯航线 FPGA 学习套件主板上使用了一枚 3.3V 驱动无源蜂鸣器，其电路如下所示：



电容 C37 用于提高电路抗干扰性能。D1 起保护三极管的作用，当三极管突然截止时，无源蜂鸣器两端产生的瞬时感应电动势可以通过 D1 迅速释放掉，避免叠加到三极管集电极上从而击穿三极管。

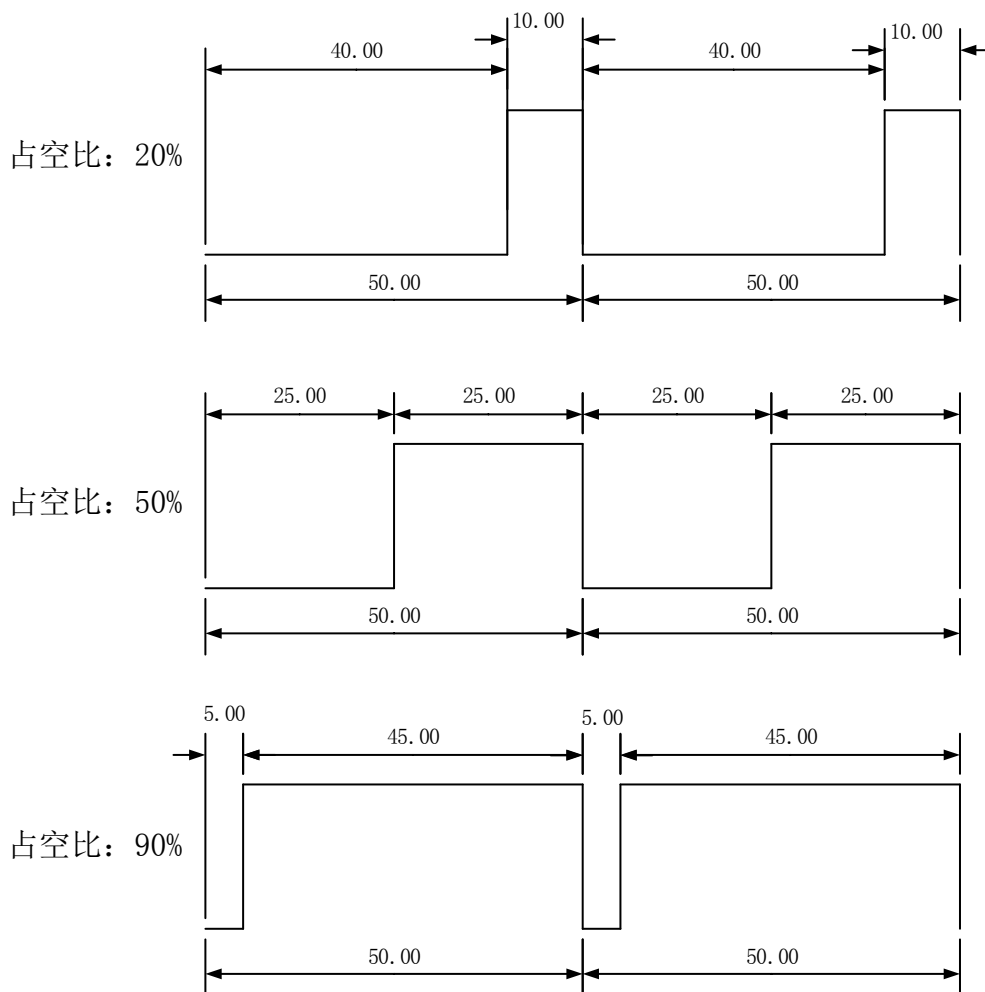
beep 端口接 FPGA 输出管脚，使用时，只需要在 beep 信号上输出 2~5KHz 的 pwm 波，就能驱动蜂鸣器发声。

无源蜂鸣器控制器设计

通过前面对无源蜂鸣器的特点介绍可知，要使无源蜂鸣器能够正常发声，需要在控制端 beep 给出相应频率的 PWM 波。因此，对于无源蜂鸣器的控制，就转化为了设计一个 PWM 波发生电路。因此，接下来我们将介绍 PWM 波发生电路的设计。

何为 PWM 波？PWM 的英文全名叫 Pulse Width Modulation，即脉冲宽度调制。通过对一系列脉冲的宽度进行调制，来等效地获得所需要波形(含形状和幅值)。PWM 控制技术在逆变电路中的应用最广，应用的逆变电路绝大部分是 PWM 型，广泛应用在从测量、通信到功率控制与变换的许多领域中。

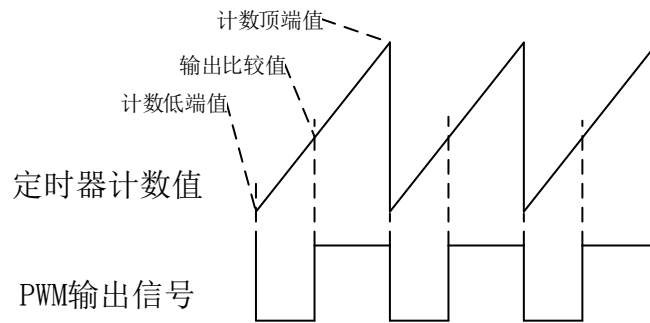
以下为周期为 1KHz，脉冲宽度（占空比）分别为 20%、50%、90%的波形图：



由图可知，当信号周期一定，信号高电平时间所占总时间的百分比不一样，即为不同占空比的 PWM 波。在逆变电路中，当使用这样的波形去驱动 MOS 管的导通时，因为一个周期内不同占空比的 PWM 信号其高电平持续长度不一样，因此使得 MOS 管的开通时间也不一样，从而使得电路中的平均电流也不一样，因此，通过调整驱动信号的占空比即可调整被控制电路中的平均电流。

而除了调整 PWM 信号的占空比，PWM 信号的周期也是可以调整的，例如，在逆变电路中，使用 IGBT 作为开关器件，常见开关频率为几 K 到几十 K，而使用 MOS 管作为开关器件，其开关频率则可高达几百 K。因此，对于不同的器件，对驱动信号的频率要求也不一样。所以，还需要能够对 PWM 波的频率进行调整。

通过以上分析，可以知道，要设计一个 PWM 发生电路，需要能够实现对信号的频率和占空比的调节。使用过单片机或者 DSP 产生 PWM 波的朋友应该知道，在单片机或者 DSP 中，产生 PWM 波的方法就是使用片上定时器进行循环计数，通过设定定时器的一个定时周期时长来确定对应输出 PWM 信号的频率，同时还有一个比较器，该比较器比较定时器的实时计数值与用户设定的比较值的大小，根据比较结果来控制输出信号的电平高低。通过设定不同的比较值，即可实现不同占空比的 PWM 信号输出。



对于 FPGA 来说，要产生 PWM 波，也可以借鉴单片机或 DSP 使用定时器产生 PWM 波的思路。

基于 FPGA 的 PWM 电路设计

根据 DSP 产生 PWM 波典型原理，在 FPGA 中设计 PWM 发生器时，也可提取出如下两个主要电路：

- 1、定时器/计数器电路
- 2、输出比较电路

定时器/计数器电路设计

定时器电路设计较为简单，在《小梅哥 FPGA 设计思想与验证方法视频教程》中，04 课“计数器设计与验证”介绍了最简单的计数器设计。参考各种 MCU 的计数器输出 PWM 波时的典型配置，可知该定时/计数器采用循环递减的计数方式，计数器循环从设定的计数初始值递减到 0，然后再回到计数初始值再次递减。这样，只需要设定一个计数初始值，并确定计数时钟源频率，即可确定计数一个完整周期的时间，也即 PWM 信号频率。

在本节中，我们设计定时/计数器的计数时钟源频率为芯航线 FPGA 学习套件主板上晶体振荡器的输出频率 50MHz，定时/计数器位宽为 32 位，则计数器代码如下所示：

```
reg [31:0] counter; //定义 32 位计数器
reg [31:0] counter_arr; //定义 32 位预重装寄存器
always@(posedge Clk50M or negedge Rst_n)
if(!Rst_n)
    counter <= 32'd0;
else if(cnt_en)begin
    if(counter == 0)
        counter <= counter_arr; //计数到 0，加载自动预重装寄存器值
    else
        counter <= counter - 1'b1; //计数器自减 1
end
else
    counter <= counter_arr; //没有使能时，计数器值等于预重装寄存器值
```

输出比较电路

输出比较电路通过比较计数器实时计数值与比较寄存器中的设定值，来确定最终 PWM 输出信号的电平状态。这里，我们可以定义，当计数器计数值大于等于比较值时，PWM 输出端输出低电平，当计数值小于比较值时，PWM 输出端输出高电平。因此输出比较电路设计代码如下：

```
reg o_pwm; //pwm 输出信号
reg [31:0] counter_ccr; //定义 32 位输出比较寄存器
always@(posedge Clk50M or negedge Rst_n)
if(!Rst_n) //让 PWM 输出信号复位时输出低电平
    o_pwm <= 1'b0;
else if(counter >= counter_ccr) //计数值大于比较值
    o_pwm <= 1'b0; //输出为 0
else //计数值小于比较值
    o_pwm <= 1'b1; //输出为 1
```

完整 PWM 发生电路设计

通过以上设计，一个最简单的 PWM 产生电路主要电路就设计完成了，以下为 PWM 产生电路的完整代码：

```
module pwm_generator(
    Clk50M,
    Rst_n,
    cnt_en,
    counter_arr,
    counter_ccr,
    o_pwm
);
input Clk50M; //50MHz 时钟输入
input Rst_n; //复位输入，低电平复位
input cnt_en; //计数使能信号
input [31:0] counter_arr; //输入 32 位预重装值
input [31:0] counter_ccr; //输入 32 位输出比较值
output reg o_pwm; //pwm 输出信号

reg [31:0] counter; //定义 32 位计数器
always@(posedge Clk50M or negedge Rst_n)
if(!Rst_n)
    counter <= 32'd0;
else if(cnt_en) begin
    if(counter == 0)
        counter <= counter_arr; //计数到 0，加载自动预重装寄存器值
    else
```

```

        counter <= counter - 1'b1; //计数器自减 1
    end
    else
        counter <= counter_arr; //没有使能时，计数器值等于预重装寄存器值

    always@(posedge Clk50M or negedge Rst_n)
    if(!Rst_n) //让 PWM 输出信号复位时输出低电平
        o_pwm <= 1'b0;
    else if(counter >= counter_ccr) //计数值大于比较值
        o_pwm <= 1'b0; //输出为 0
    else //计数值小于比较值
        o_pwm <= 1'b1; //输出为 1
    endmodule

```

PWM 发生电路仿真验证

对本 PWM 发生电路的验证思路比较简单，只需要产生 50MHz 基准计数时钟源（其他频率也可以，只需要修正频率和占空比计算公式中的相关参数），然后给出预重装值和输出比较值，然后使能计数，即可启动 PWM 输出。在运行过程中，修改预重装值可以设置输出 PWM 信号的频率，并将同时影响输出占空比，而在预重装值确定的情况下，修改输出比较值，则可以设置输出占空比。

最终输出 PWM 波的频率计算公式为：

$$f_{pwm} = \frac{f_{clk}}{counter_arr + 1}$$

因此，当输出频率确定时，可计算得到预重装值，计算公式为：

$$counter_arr = \frac{f_{clk}}{f_{pwm}} - 1$$

例如，当希望设置输出信号频率为 5KHz 时

$$counter_arr = \frac{f_{clk}}{f_{pwm}} - 1 = \frac{50000000}{5000} - 1 = 9999$$

因此，我们只需要设置 counter_arr 值为 9999 即可使得最终输出信号频率为 5KHz。
当输出 PWM 频率确定后，其输出占空比计算则为输出比较值与预重装值之商。计算公式为：

$$PW = \frac{counter_ccr}{counter_arr}$$

因此，当输出占空比确定时，可计算得到输出比较值，计算公式为：

$$counter_ccr = PW \times counter_arr$$

例如，当输出频率为 5KHz，输出占空比为 70%时

$$counter_ccr = PW \times counter_arr = 9999 \times 0.7 = 6999$$

PWM 发生电路 testbench 设计

根据上述计算公式，可以设计 pwm_generator 模块的仿真文件如下所示：

```
`timescale 1ns/1ns
`define clk_period 20

module pwm_generator_tb;

    reg Clk50M; //50MHz 时钟输入
    reg Rst_n; //复位输入，低电平复位
    reg cnt_en; //计数使能信号
    reg [31:0]counter_arr;//输入 32 位预重装值
    reg [31:0]counter_ccr;//输入 32 位输出比较值
    wire o_pwm; //pwm 输出信号

    pwm_generator pwm_generator(
        .Clk50M(Clk50M),
        .Rst_n(Rst_n),
        .cnt_en(cnt_en),
        .counter_arr(counter_arr),
        .counter_ccr(counter_ccr),
        .o_pwm(o_pwm)
    );

    initial Clk50M = 0;
    always #(`clk_period/2) Clk50M = ~Clk50M;

    initial begin
        Rst_n = 0;
        cnt_en = 0;
        counter_arr = 0;
        counter_ccr = 0;
        #(`clk_period*20 +1);
        Rst_n = 1;
        #(`clk_period*10 +1);
        counter_arr = 999;//设置输出信号频率为 50KHz
        counter_ccr = 400;//设置输出 PWM 波占空比为 40%
        #(`clk_period*10);
        cnt_en = 1; //启动计数以产生 PWM 输出
        #100050;
        counter_ccr = 700;//设置输出 PWM 波占空比为 70%
        #100050;
        cnt_en = 0; //停止计数以关闭 PWM 输出
    end
endmodule
```

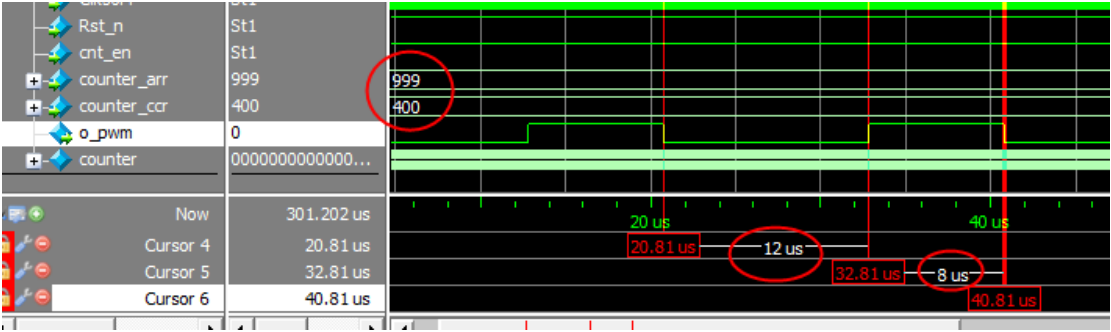
```
counter_arr = 499; //设置输出信号频率为 100KHz
counter_ccr = 250; //设置输出 PWM 波占空比为 50%
#(`clk_period*10);
cnt_en = 1; //启动计数以产生 PWM 输出
#50050;
counter_ccr = 100; //设置输出 PWM 波占空比为 20%
#50050;
$stop;

end

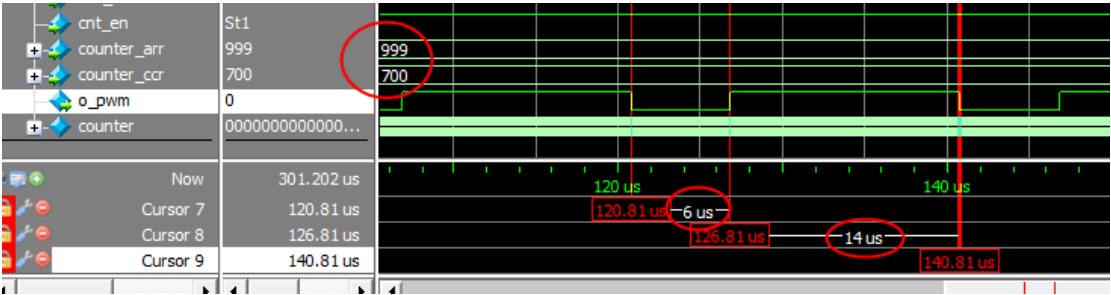
endmodule
```

仿真结果分析

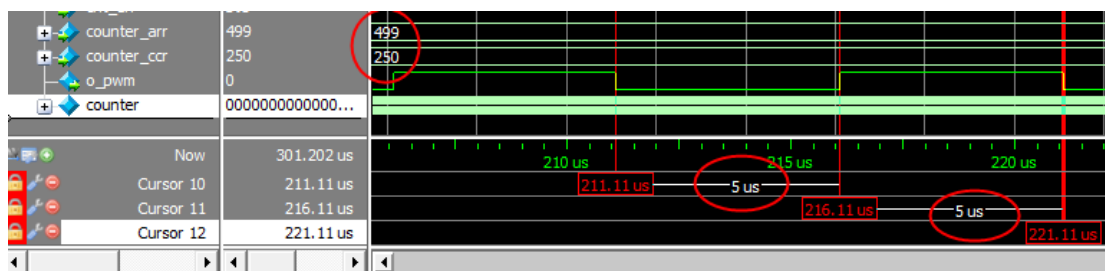
下图为设置输出 PWM 波频率为 50KHz (counter_arr 为 999)、占空比为 40% (counter_ccr 为 400) 时的仿真波形，由图可知，低电平周期为 12us，高电平周期为 8us，整个信号周期为 20us，即频率为 50KHz。占空比为 $8/20 = 0.4$ 。



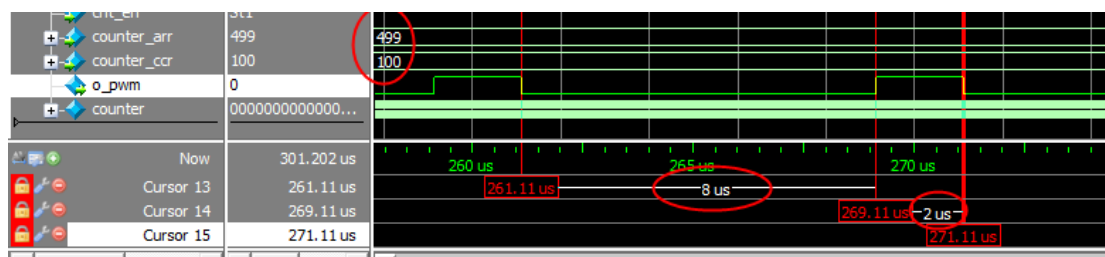
下图为设置输出 PWM 波频率为 50KHz (counter_arr 为 999)、占空比为 70% (counter_ccr 为 700) 时的仿真波形，由图可知，低电平周期为 6us，高电平周期为 14us，整个信号周期为 20us，即频率为 50KHz。占空比为 $14/20 = 0.7$ 。



下图为设置输出 PWM 波频率为 100KHz (counter_arr 为 499)、占空比为 50% (counter_ccr 为 250) 时的仿真波形，由图可知，低电平周期为 5us，高电平周期为 5us，整个信号周期为 10us，即频率为 100KHz。占空比为 $5/10 = 0.5$ 。



下图为设置输出 PWM 波频率为 100KHz(counter_arr 为 499)、占空比为 20%(counter_ccr 为 100) 时的仿真波形，由图可知，低电平周期为 8us，高电平周期为 2us，整个信号周期为 10us，即频率为 100KHz。占空比为 $2/10 = 0.2$ 。



由此可知，该 PWM 生成电路能够正确的产生 PWM 输出信号。

PWM 驱动蜂鸣器板级验证

通过仿真验证，我们确认了该 PWM 发生电路理论设计正确，接下来，我们将使用该 PWM 发生模块来驱动芯航线 FPGA 开发板上的无源蜂鸣器，让无源蜂鸣器能够循环依次发出“哆来咪发梭拉西”的音调。（本想让蜂鸣器能够演奏一曲的，可是无奈本人音乐天赋为负数，学不会谱曲，因此只能把最基本的“哆来咪发梭拉西”放出来了，希望有音乐天赋的朋友能在此基础上谱写演奏出美丽的乐章）。

以下为查资料得知的每个乐调对应的频率。

音名	频率/Hz	音名	频率/Hz	音名	频率/Hz
低音 1	261.6	中音 1	523.3	高音 1	1045.5
低音 2	293.7	中音 2	587.3	高音 2	1174.7
低音 3	329.6	中音 3	659.3	高音 3	1318.5
低音 4	349.2	中音 4	698.5	高音 4	1396.9
低音 5	392	中音 5	784	高音 5	1568
低音 6	440	中音 6	880	高音 6	1760
低音 7	493.9	中音 7	987.8	高音 7	1975.5

根据每个音调的频率值，可以计算得出 PWM 发送模块的预重装值，以下为计算得出的音调频率与对应 PWM 发送模块输出相应频率的预重装值。

频率/Hz	预重装值	频率/Hz	预重装值	频率/Hz	预重装值
261.6	191130	523.3	95546	1045.5	47823
293.7	170241	587.3	85134	1174.7	42563

329.6	151698	659.3	75837	1318.5	37921
349.2	143183	698.5	71581	1396.9	35793
392	127550	784	63775	1568	31887
440	113635	880	56817	1760	28408
493.9	101234	987.8	50617	1975.5	25309

本例中，保持 PWM 波的占空比始终为 50%即可，而通过前面仿真验证可知，占空比为 50%时，输出比较值刚好为预重装值的一半，因此，我们只需要将预重装值除以 2（右移一位）的结果直接赋值给输出比较值即可，这样可以避免再重复计算输出比较值。

另外，为了保证音调的切换能够让我们容易分辨，因此设计一个 500ms 的定时器，每 500ms 切换一次音调。该部分电路非常简单，因此本板级验证部分将不再讲解代码的详细设计思路，只给出具体代码。

音调播放电路的代码如下所示：

```

module pwm_generator_test(
    Clk50M,
    Rst_n,
    beep
);

input Clk50M;
input Rst_n;
output beep;

reg [31:0]counter_arr; //预重装值寄存器
wire [31:0]counter_ccr; //输出比较值

reg [24:0]delay_cnt;    //500ms 延时计数器
reg [4:0]Pitch_num; //音调编号

localparam
    L1 = 191130, //低音 1
    L2 = 170241, //低音 2
    L3 = 151698, //低音 3
    L4 = 143183, //低音 4
    L5 = 127550, //低音 5
    L6 = 113635, //低音 6
    L7 = 101234, //低音 7

    M1 = 95546, //中音 1
    M2 = 85134, //中音 2
    M3 = 75837, //中音 3
    M4 = 71581, //中音 4
    M5 = 63775, //中音 5

```

```

M6 = 56817, //中音 6
M7 = 50617, //中音 7

H1 = 47823, //高音 1
H2 = 42563, //高音 2
H3 = 37921, //高音 3
H4 = 35793, //高音 4
H5 = 31887, //高音 5
H6 = 28408, //高音 6
H7 = 25309; //高音 7

//输出比较值为预重装值一半
assign counter_ccr = counter_arr >> 1;

pwm_generator pwm_generator(
    .Clk50M(Clk50M),
    .Rst_n(Rst_n),
    .cnt_en(1'b1),
    .counter_arr(counter_arr),
    .counter_ccr(counter_ccr),
    .o_pwm(beep)
);

//500ms 延时计数器计数
always@(posedge Clk50M or negedge Rst_n)
if(!Rst_n)
    delay_cnt <= 25'd0;
else if(delay_cnt == 0)
    delay_cnt <= 25'd24999999;
else
    delay_cnt <= delay_cnt - 1'b1;

//每 500ms 切换一次音调
always@(posedge Clk50M or negedge Rst_n)
if(!Rst_n)
    Pitch_num <= 5'd0;
else if(delay_cnt == 0)begin
    if(Pitch_num == 5'd20)
        Pitch_num <= 5'd0;
    else
        Pitch_num <= Pitch_num + 5'd1;
end
else
    Pitch_num <= Pitch_num;

```

```
//根据音调编号给预重装值给相应的值
always@(*)
    case(Pitch_num)
        0 :counter_arr = L1;
        1 :counter_arr = L2;
        2 :counter_arr = L3;
        3 :counter_arr = L4;
        4 :counter_arr = L5;
        5 :counter_arr = L6;
        6 :counter_arr = L7;
        7 :counter_arr = M1;
        8 :counter_arr = M2;
        9 :counter_arr = M3;
        10:counter_arr = M4;
        11:counter_arr = M5;
        12:counter_arr = M6;
        13:counter_arr = M7;
        14:counter_arr = H1;
        15:counter_arr = H2;
        16:counter_arr = H3;
        17:counter_arr = H4;
        18:counter_arr = H5;
        19:counter_arr = H6;
        20:counter_arr = H7;
        default:counter_arr = L1;
    endcase
endmodule
```

蜂鸣器音调播放电路的引脚分配如下表所示:

无源蜂鸣器	对应 FPGA 管脚
beep	PIN_N8
Clk50M	PIN_E1
Rst_n	PIN_M1

引脚分配完成后对工程全编译,然后下载到芯航线 FPGA 开发板上,下载完成后蜂鸣器即开始循环从低音 1 播放到高音 7。