蜂鸣器实验论文

20 田鹏飞 赵阳 郑翔宇

邮箱：zy791027638@163.com

中国海洋大学 2012 级通信工程

摘要：给蜂鸣器输入相应的频率，可以使其发出表中所示的低音、中音、高音的do~xi的声音。将其按照音乐演奏的规律组合，便可以得到所需要的乐曲。

关键词：音调；频率；基准频率分频

选题背景：我们希望能够通过自己双手为大家做出一首美妙的乐曲，这既是一种乐趣，也一种锻炼自己的机会。乐曲是一种陶冶情操的美好的东西。所以我们选取能够用音乐愉悦大家的蜂鸣器实验。

意义：通过这次实验，我们认识到了自己的不足，也认识到了团队合作的重要性。我们这个团体通过这次实验经历的困难波折，认识到做事须认真，执着不放弃，只有勇敢的做下去就能拥有自己想要的结果。即是达不到想要的程度，也对的起自己的辛苦劳动。

相关工作：1、设置端口

1 输入端口

CLK：12MHZ 系统时钟输入端口。

2）输出端口

device：乐曲的声音输出端口，输出的是对应各音符频率的方波信号。

2、设置模块

1）自动演奏模块

自动演奏模块可以自动播放电子琴内置乐曲，按节拍读取内置乐谱。将键盘输入的音符信号输出。因此，本模块是向 Tone 模块提供音符信息。

首先，对 12MHz 系统时钟进行 3M 的分频，得到 4Hz 的信号，这样一秒中就可以按照四拍进行。然后依照此频率进行地址累计。

2）音频发生器模块

根据自动演奏模块的信号输出，不同的信号被翻译为不同的频率。

3）蜂鸣器驱动模块

根据音频发生器发出音频的不同，蜂鸣器得到的驱动也不同。首先，对系统时钟进行 16 分频，再对 0.75mhz 的脉冲再次分频，得到所需要的音符频率，然后再进行 2 分频。

实验结果及分析：

实验代码

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
```

```
use ieee.std_logic_unsigned.all;
entity tone is
    port(
            index: in std_logic_vector(15 downto 0);
--音符输入信号
            tone0: out integer range 0 to 2047
--音符的分频系数
        );
end tone;
architecture behavioral of tone is
begin
    search :process(index)  --此进程完成音符到音符的分频系数译码，音符的显示，高低音阶
    begin
        case index is
            when "0000000000000001" => tone0<=1433;
            when "0000000000000010" => tone0<=1277;
            when "0000000000000100" => tone0<=1138;
            when "0000000000001000" => tone0<=1074;
            when "0000000000010000" => tone0<=960;
            when "0000000000100000" => tone0<=853;
            when "0000000001000000" => tone0<=759;
            when "0000000010000000" => tone0<=716;
            when "0000000100000000" => tone0<=358;
            when "0000001000000000" => tone0<=319;
```

```
                when "0000010000000000" => tone0<=284;
                when "0000100000000000" => tone0<=268;
                when "0001000000000000" => tone0<=239;
                when "0010000000000000" => tone0<=213;
                when "0100000000000000" => tone0<=190;
                when "1000000000000000" => tone0<=638;
                when others => tone0<=0;
            end case;
        end process;
end behavioral;



library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity speaker is
    port(
                clk1: in std_logic;
--系统时钟 12mhz
                tone1: in integer range 0 to 2047;
--音符分频系数
                spks: out std_logic
--驱动扬声器的音频信号
            );
```

```
end speaker;
architecture behavioral of speaker is
signal  preclk, fullspks:std_logic;
begin
     p1:process(clk1)--此进程对系统时钟进行 16 分频
     variable count: integer range 0 to 16;
     begin
          if clk1'event and clk1='1' then
count:=count+1;
               if count=8 then
                    preclk<='1';
               elsif count=16 then
                    preclk<='0';
                    count:=0;
               end if;
          end if;
     end process p1;


   p2:process(preclk,tone1)--对 0.75mhz 的脉冲再次分频，
得到所需要的音符频率
     variable count11:integer range 0 to 2047;
     begin
          if preclk'event and preclk='1' then
             if count11<tone1 then
```

```
                count11:=count11+1;
                fullspks<='1';
            else
                count11:=0;
                fullspks<='0';
            end if;
        end if;
    end process p2;

    p3:process(fullspks)--此进程对 fullspks 进行 2 分频
    variable count2: std_logic:='0';
    begin
    if fullspks'event and fullspks='1' then
        count2:=not count2;
        if count2='1' then
            spks<='1';
        else
            spks<='0';
       end if;
    end if;
    end process p3;
end behavioral;

library ieee;
```

```
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity laohu is
    port(
            clk: in std_logic;--系统时钟；键盘输入/自动演奏
            tone_key_0: buffer std_logic_vector(15 downto 0)--音符信号输出
        );
end  laohu;
architecture behavioral of laohu is
signal count0:integer range 0 to 31;--change
signal clk2:std_logic;
begin
    p1:process(clk)  --对12mhz系统时钟进行3m的分频，得到4hz的信号clk2
    variable count:integer range 0 to 3000000;
    begin
        if clk'event and clk='1' then
            count:=count+1;
            if count=1500000 then
                clk2<='1';
            elsif count=3000000 then
                clk2<='0';
```

```
                count:=0;
            end if;
        end if;
    end process p1;

    p2:process(clk2)--此进程完成自动演奏部分乐曲的地址累加
    begin
        if clk2'event and clk2='1' then
            if count0=29 then
                count0<=0;
            else
                count0<=count0+1;
            end if;
        end if;
    end process p2;

    p3:process(count0,tone_key_0)
    begin
        case count0 is--此case语句：存储自动演奏部分的乐曲
            when 0 => tone_key_0<=b"00000001_00000000";  --1
            when 1 => tone_key_0<=b"00000010_00000000";  --2
```

```
                when 2 =>
tone_key_0<=b"00000100_00000000";  --3

                when 3 =>
tone_key_0<=b"00000001_00000000";  --1

                when 4 =>
tone_key_0<=b"00000001_00000000";  --1

                when 5 =>
tone_key_0<=b"00000010_00000000";  --2

                when 6 =>
tone_key_0<=b"00000100_00000000";  --3

                when 7 =>
tone_key_0<=b"00000001_00000000";  --1

                when 8 =>
tone_key_0<=b"00000100_00000000";  --3

                when 9 =>
tone_key_0<=b"00001000_00000000";  --4

                when 10 =>
tone_key_0<=b"00010000_00000000";  --5

                when 11 =>
tone_key_0<=b"00000100_00000000";  --3

                when 12 =>
tone_key_0<=b"00001000_00000000";  --4

                when 13 =>
tone_key_0<=b"00010000_00000000";  --5

                when 14 =>
tone_key_0<=b"00010000_00000000";  --5

                when 15 =>
tone_key_0<=b"00100000_00000000";  --6
```

```
                when 16 =>
tone_key_0<=b"00010000_00000000";  --5

                when 17 =>
tone_key_0<=b"00001000_00000000";  --4

                when 18 =>
tone_key_0<=b"00000100_00000000";  --3

                when 19 =>
tone_key_0<=b"00000001_00000000";  --1

                when 20 =>
tone_key_0<=b"00010000_00000000";  --5

                when 21 =>
tone_key_0<=b"00100000_00000000";  --6

                when 22 =>
tone_key_0<=b"00010000_00000000";  --5

                when 23 =>
tone_key_0<=b"00001000_00000000";  --4

                when 24 =>
tone_key_0<=b"00000100_00000000";  --3

                when 25 =>
tone_key_0<=b"00000001_00000000";  --1

                when 26 =>
tone_key_0<=b"00000100_00000000";  --3

                when 27 =>
tone_key_0<=b"00000000_00100000";  --di6

                when 28 =>
tone_key_0<=b"00000001_00000000";  --1

                when others => null;

                end case;
```

```
    end process p3;
end behavioral;


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity beep0 is
    port(
            clk:in std_logic;
            device:out std_logic
        );
end beep0;
architecture behavioral of beep0 is
component laohu is
    port(
            clk: in std_logic;--系统时钟；键盘输入/
自动演奏
            tone_key_0: out std_logic_vector(15
downto 0)--音符信号输出
        );
end  component;
component tone is
    port(
```

```
                index: in std_logic_vector(15 downto
0);--音符输入信号
                tone0: out integer range 0 to 2047--音符
的分频系数
            );
end component;
component speaker is
    port(
                clk1: in std_logic;--系统时钟 12mhz
                tone1: in integer range 0 to 2047; --音
符分频系数
                spks: out std_logic--驱动扬声器的音频信
号
            );
end component;
signal mid:std_logic_vector(15 downto 0);
signal tones:integer;
begin
    u0:laohu port map(clk,mid);
    u1:tone port map(mid,tones);
    u2:speaker port map(clk,tones,device);
end behavioral;
```

频率折算中，由于频率计数 3 不能是小数，采用了四舍五入的方法，所以得到的频率并不是十分精确的，但是不会影响结果。

乐曲按照每秒钟四拍进行，循环地从事先编好的代码中翻译出频率来，所以能听到完整的乐曲。

总结与展望：实验让我们这个团体更加的紧密，也让我们认识到自己在做动手操作方面的不足及相关知识学习的不扎实。实验的成功增加了我们学习未知知识的兴趣，增强了我们动手操作能力，认识到了团队协作的重要性。我们希望能够在以后的生活里拥有这样的提升自我的团队合作项目，让大家更好的充实自己。相信以后的实验会更加的精彩、神秘、动人心魄。

在此，我感谢老师给我们展示自我，充实自我的机会，让我们收益颇多。也感谢学校里拥有这样的平台来让我们完成自己想做的实验。更要感谢我的队友、伙伴。是这群队友的鼓励支持才让我们不放弃、坚持自己、完成实验。

贡献值：赵阳：自动演奏模块的编写、调试；

田鹏飞：音频发生模块的编写、调试；

郑翔宇： 蜂鸣器驱动模块的编写、调试；