**华中科技大学**

**《电子线路设计、测试与实验》实验报告**

|  |  |
| --- | --- |
| 实验名称： | FPGA综合实验 |
| 院（系）： | 机械科学与工程学院 |
| 专业班级： |  |
| 姓名： | Wyt |
| 学号： |  |
| 时间： | 2019.12.18 |
| 地点： | 华中科技大学南一楼中213 |
| 实验成绩： |  |
| 指导教师： |  |

2019 年12 月 18 日

# 一、实验目的

1.熟练掌握Basys2的开发流程；

2.提高FPGA综合调试的能力；

# 二、实验选题及设计目标

实验选题：

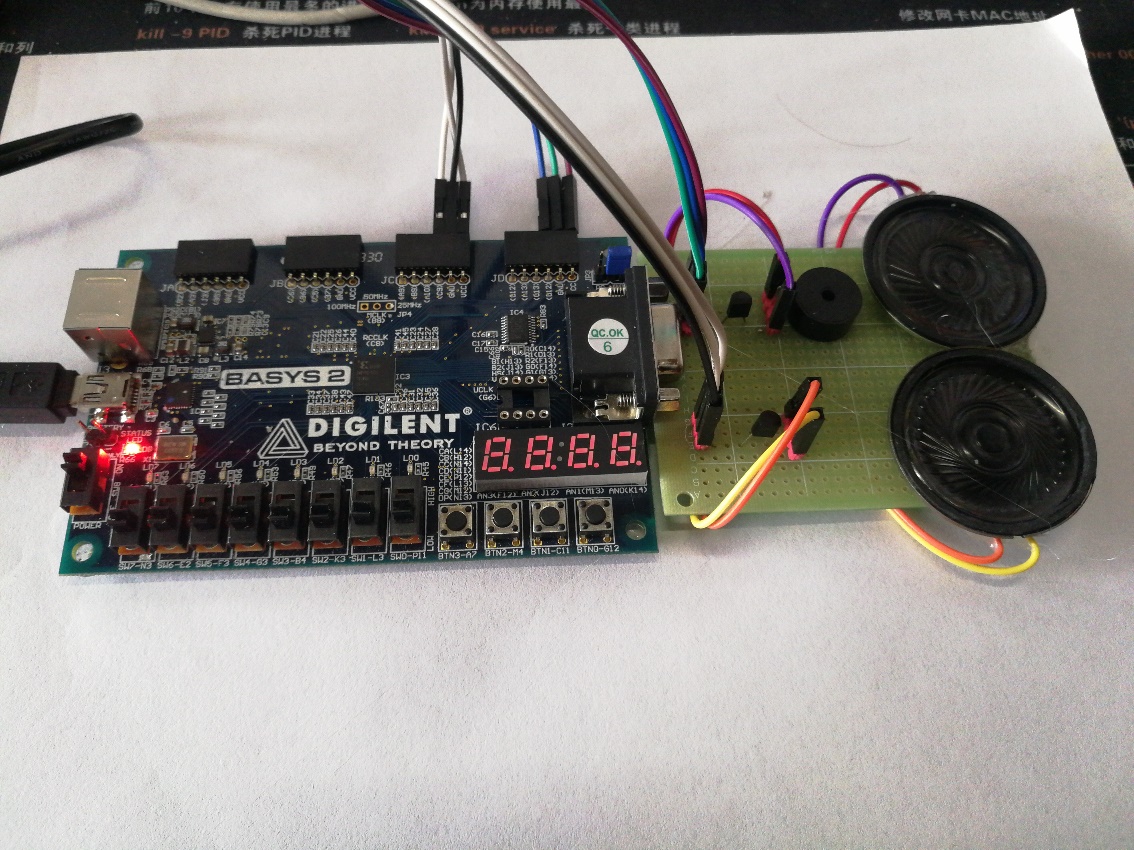
音乐播放器+电子琴（双声道）

项目目标：

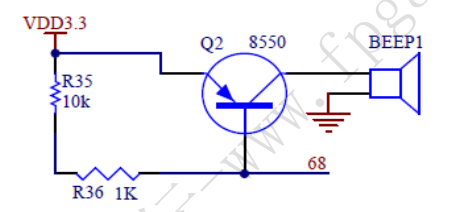
本项目要设计实现一个基于BASYS2的多功能电子琴，音乐播放和电子琴各占用一个声道的双声道电子琴。计划采用两个扬声器作为双声道的输出设备，利用拨码开关实现音乐播放器的控制，利用轻触开关实现电子琴的琴键（这里由于BASYS2开发板上只有4个轻触开关，因此电子琴部分仅选用了四个音，仅用来演示电子琴的实现原理，也可以外接按键，或者矩阵键盘拓展）。另外在程序中写定一个音乐的乐谱，实现音乐播放器的演示功能，如果时间允许，会在程序中存储多段音乐，使用拨码开关选择播放曲目。

# 三、硬件部分

## 实验整体电路



## 扬声器简易驱动电路



# 四、软件部分

## 系统主频获取

系统主频由开发板主频经过10分频得到：50MHz分频为5MHz

代码（digital\_piano.v）:

/\* 10分频：50MHz到5Mhz分频 \*/

always@(posedge inclk)

begin

if(cnt<3'd5)

cnt <= cnt + 3'b1;

else

begin

cnt <= 3'b0;

clk\_5MHz <= ~clk\_5MHz;

end

end

## 电子琴子模块

读取输入的轻触按键的状态，决定声音频率，再通过计数器实现定频率方波的输出，作为扬声器的驱动信号。

其中origin的值为根据各个音调对应的声音频率换算确定的。

代码（piano.v）:

/\* piano子模块 \*/

module piano(inclk, clk\_5MHz, key\_in, beep);

input inclk; //开发板主频

input clk\_5MHz; //系统时钟

input [3:0] key\_in; //轻触按键输入

output beep; //扬声器输出

wire carry;

reg beep\_r;

assign beep = beep\_r; //输出音乐

/\* 按键转换音调输出 \*/

always @ (posedge clk\_5MHz)

begin

case (key\_in[3:0])

/\* 消除不按时的噪声 \*/

4'b0000: origin<=0000;

4'b0001: origin<=6826; // Hdo

4'b0010: origin<=7871; // Hre

4'b0100: origin<=8798; // Hmi

4'b1000: origin<=9224; // Hfa

default: origin<=0000;

endcase

end

assign carry=(count == 16383);

/\* 获取驱动信号 \*/

always @(posedge clk\_5MHz)

begin

if(key\_in[3:0] == 4'b0000)

begin

end

else

if(carry)

count = origin;

else

count = count + 1;

end

/\* 扬声器驱动 \*/

always @(posedge carry)

begin

beep\_r<=~beep\_r;

end

endmodule

## 音乐播放器子模块

音乐播放器子模块首先通过分频实现了一个低频时钟(这里采用5Hz),作为音乐的节拍器。然后调用按照程序翻译出来的曲谱函数(这里由于时间原因仅翻译了天空之城一首音乐的一部分)。

代码（song.v）：

module song(clk\_5MHz, select, beep);

input clk\_5MHz;

input select; //音乐播放暂停选项

output beep; //蜂鸣器输出

reg [25:0] cnt2; //计数器

reg clk\_5Hz;

wire beep\_r;

wire out3; //曲谱的输出

wire clk;

assign beep = beep\_r;

/\* 调用曲谱模块——模拟有限状态机实现 \*/

/\* 《天空之城》 \*/

song3 m3(.clk\_5MHz(clk\_5MHz),.clk\_4Hz(clk),.select(select),.beep(out3));

assign beep\_r = out3;

assign clk = clk\_5Hz;

/\* 5hz分频 \*/

always@(posedge clk\_5MHz)

begin

// if(select)

// begin

if(cnt2<25'd400000)

cnt2 <= cnt2+25'b1;

else

begin

cnt2<=25'b0;

clk\_5Hz <= ~clk\_5Hz;

end

// end

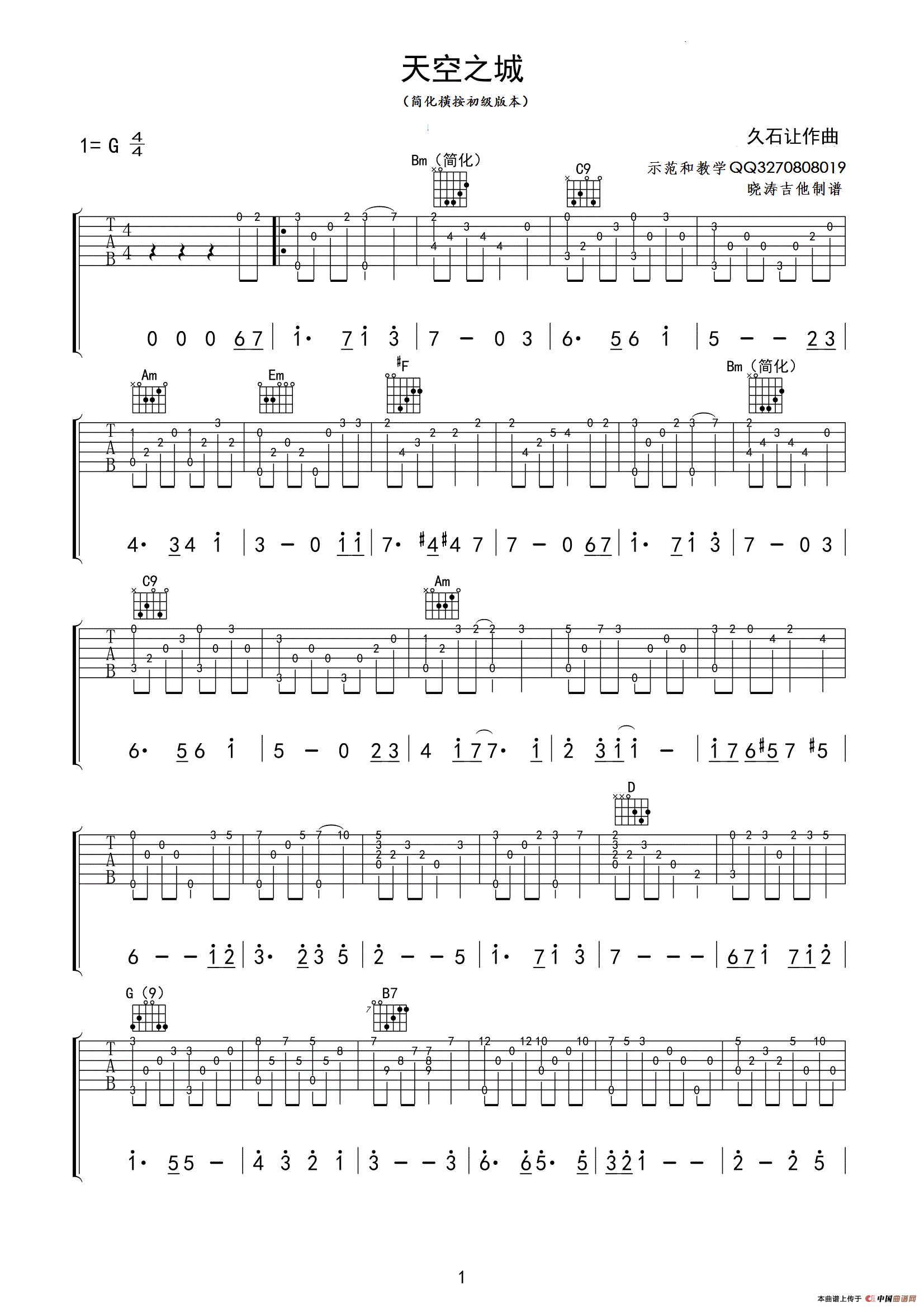
end

endmodule

## 曲谱子模块

这里对天空之城曲谱的一部分进行了翻译,由于钢琴谱比较复杂,且一般为双手谱,这里选择了一个较为简单的吉他谱,按照吉他谱中的简谱进行逐拍翻译。

曲谱如下：



代码（song3.v）:

/\* 《天空之城》 \*/

module song3(clk\_5MHz,clk\_4Hz,select,beep);

input clk\_5MHz,clk\_4Hz,select; //系统时钟，节拍时钟，播放暂停选项

output beep; //扬声器输出

reg [3:0] high,med,low;

reg [15:0] origin;

reg beep\_r;

reg [7:0] state;

reg [15:0] count;

assign beep = beep\_r;

/\* 扬声器基本驱动 \*/

always @(posedge clk\_5MHz)

begin

/\* 计数器 \*/

count <= count + 1'b1;

If(count == origin)

begin

/\* 计数器清零 \*/

count <= 16'h0;

/\* 输出取反 \*/

beep\_r <= !beep\_r;

end

end

/\* 音调输出转换 \*/

always@(posedge clk\_4Hz)

begin

if(select)

begin

case({high,med,low})

/\* 24音转换为方波频率 \*/

'b000000000001:origin=22900; //低1

'b000000000010:origin=20408; //低2

'b000000000011:origin=18181; //低3

'b000000000100:origin=17142; //低4

'b000000000101:origin=15267; //低5

'b000000000110:origin=13605; //低6

'b000000000111:origin=12121; //低7

'b000000010000:origin=11472; //中1

'b000000100000:origin=10216; //中2

'b000000110000:origin=9101; //中3

'b000000111000:origin=8571; //中4

'b000001010000:origin=7653; //中5

'b000001100000:origin=6818; //中6

'b000010000000:origin=6060; //中7

'b000100000000:origin=5733; //高1

'b001000000000:origin=5108; //高2

'b001100000000:origin=4551; //高3

'b001010000000:origin=4294; //高4

'b010000000000:origin=3826; //高5

'b011000000000:origin=3409; //高6

'b010100000000:origin=3050; //高7

endcase

end

else

/\* 消除杂音 \*/

origin=0000;

end

/\* 《天空之城》乐谱翻译 \*/

always @(posedge clk\_4Hz)

begin

if(select)

begin

/\* 全曲总节拍数 \*/

if(state ==193)

/\* 自动重放 \*/

state = 0;

else

/\* 节拍计数 \*/

state = state + 1'b1;

case(state)

/\* 按小节间隔 \*/

/\* 1 \*/

0: {high,med,low}='b000001100000;//中6

1: {high,med,low}='b000010000000;//中7

/\* 2 \*/

2,3,4: {high,med,low}='b000100000000;//高1

5: {high,med,low}='b000010000000;//中7

6,7: {high,med,low}='b000100000000;//高1

8,9: {high,med,low}='b001100000000;//高3

/\* 3 \*/

10,11,12,13,14,15: {high,med,low}='b000010000000;//中7

16,17: high,med,low}='b000000110000;//中3

/\* 4 \*/

18,19,20: {high,med,low}='b000001100000;//中6

21: {high,med,low}='b000001010000;//中5

22,23: {high,med,low}='b000001100000;//中6

24,25: {high,med,low}='b000100000000;//高1

………………………………..

/\* 25 \*/

186,187,188,189,190,191,192,193: {high,med,low}='b001100000000;//高3

endcase

end

end

endmodule

## 管脚定义

代码（digital\_piano\_ucf.ucf）:

NET "inclk" LOC = "B8"; //开发板主频

NET "select" LOC = "N3"; //拨码开关作为播放器暂停选项

/\* 双声道输出:电子琴&音乐播放器,互不干扰

NET "out\_r" LOC = "D12"; //右声道输出

NET "out\_l" LOC = "C9"; //左声道输出

/\* 电子琴演示琴键:轻触开关 \*/

NET "key\_in[3]" LOC = "G12";

NET "key\_in[2]" LOC = "C11";

NET "key\_in[1]" LOC = "M4";

NET "key\_in[0]" LOC = "A7";

NET "select" CLOCK\_DEDICATED\_ROUTE = FALSE;

# 五、实验结果及分析

# 1、电子琴功能

### 结果：

经过测试，电子琴功能声音输出正常。

但是和音乐播放器对比声音较小，不清楚具体原因，可能是两个扬声器的硬件差别。

### 实验结果分析结论：

实验结果满足实验要求，实验完成。

# 2、音乐播放器功能

### 结果：

经过测试，音乐播放器功能声音输出正常。

有个别音不准，可能是人工翻译过程中的疏忽。

### 实验结果分析结论：

实验结果满足实验要求，实验完成。

# 六、小结

这次实验的综合性比较强，代码的规模比较大。

在完成本次实验的过程中，我首先在网上搜索了相关的类似项目设计，在设计思路上进行了一定程度的参考和借鉴，这使得我的实验完成进度还是比较快的。我体会到，当第一次面对一个陌生的比较复杂的项目问题时，查找参考资料其实是一项必备的技能，往往能起到事半功倍的效果，而且可以学习优秀的工程设计思路。

另一方面也对我们自己的创新性提出了挑战，在实现有参考的项目时，可能我们遇到的问题，大部分都可以在网上找到答案，但是当我们真的面对一个全新的项目，作为一个开拓者或者探索者的时候，就是对于我们分析问题解决问题的能力的巨大考验。

另外，这次的项目设计在发声器件的设计方面其实耽误了一定的时间，是不应该的。最后是采用了一个扬声器，直接用方波驱动，声音效果比较好。起初想要使用无源蜂鸣器的，但是在测试过程中，不知道什么原因，无法驱动，输入的方波会被直接拉高，后来怀疑可能是蜂鸣器型号问题。之前做一些东西的时候，都是图方便直接选用有源蜂鸣器，这次想要尝试一下无源却遇到问题，有待后面自己研究。