北京郵電大學

数字电路综合实验报告

实验名称：	选号机的设计与实现
学　　院：	信息与通信工程学院
班　　级：	2015211109
姓　　名：	戚婧晨
班内序号：	24
学　　号：	2015210260
日　　期：	2017 年 11 月 7 日

选号机的设计与实现

一、设计课题的任务要求

设计一个选号机，可以选择以一位字母和五位阿拉伯数字组成一串号码

1. 基本要求

1.用 SW7 作为选号机开关，打开开关 SW7 后选号机自检：8*8 点阵和数码管 DISP7~DISP0 全亮 0.5s 熄灭 0.5s 重复三次，进入待机状态；
2.使用按键 BTN7 进入选号状态，按以下顺序进行选号，当前面的号码未选定时，后面的按键无效。具体要求如下：
a)8*8 点阵轮流显示“A”“B”“C”“D”“E”“F”六个大写字母，每个字母显示停留时间 0.5s，按动 BTN5 选中当前显示的字母，该字母稳定显示；
b)数码管 DISP4 上轮流显示“0~9”十个数字，每个数字显示停留时间 0.3s，按动 BTN4 选中当前显示的数字，该数字稳定显示；
c)数码管 DISP3 上轮流显示“0~9”十个数字，每个数字显示停留时间 0.2s，按动 BTN4 选中当前显示的数字，该数字稳定显示；
d)数码管 DISP2 上轮流显示“0~9”十个数字，每个数字显示停留时间 0.1s，按动 BTN4 选中当前显示的数字，该数字稳定显示；
e)数码管 DISP1 上轮流显示“0~9”十个数字，每个数字显示停留时间 0.08s，按动 BTN4 选中当前显示的数字，该数字稳定显示；
f)数码管 DISP0 上轮流显示“0~9”十个数字，每个数字显示停留时间 0.05s，按动 BTN4 选中当前显示的数字，该数字稳定显示；
3.DISP0 内容选定后表示所有内容选择完毕，所有内容整体以 2Hz 闪烁三次以示提醒，然后稳定显示；
4.使用按键 BTN7 可以重新进入选号状态，再一次进行选号。

2. 提高要求

1.自检功能、各项内容滚动时、内容选定后进行闪烁提醒时伴有适当的音乐，各个按键按下时伴有按键音；
2.点阵显示字母时切换方式改为滚入滚出，滚动速度为 0.05S/行，只有字母完整显示时才能被选中；
3. 各个数码管显示数字的方式改为随机显示“0~9”十个数字中的一个。

二、系统设计

1. 设计思路

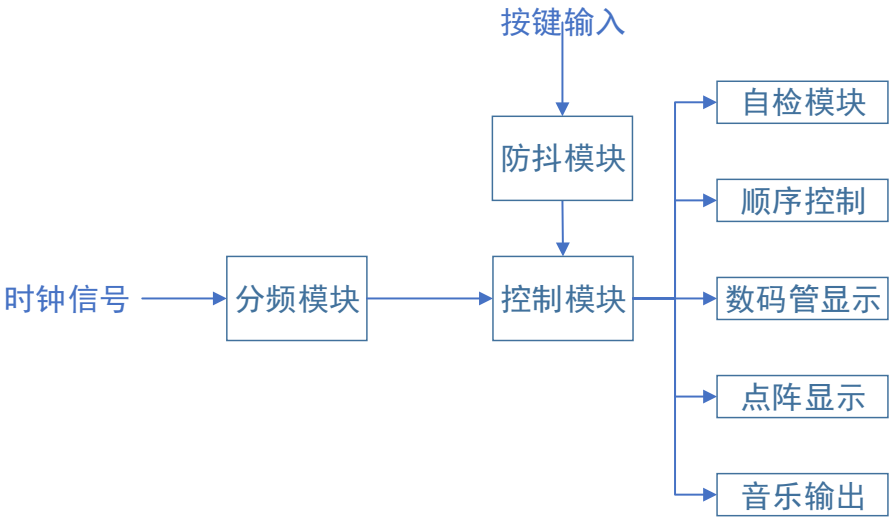
根据实验要求，本实验通过分频模块、控制模块、点阵模块、数码管模块、蜂鸣器模块实现了选号机的基本和提高功能。首先，通过分频器模块，将输入的50MHz 信号分频，得到实验中需要的 8 个不同频率的信号；其次，通过控制模块(test_control)实现自检、待机、选号、选号后闪烁、稳定显示状态的选择和状态间的转换；最后通过数码管模块、点阵模块和蜂鸣器模块实现各项功能的显示及输出。

本系统可以采用模块化设计和有限状态机的设计方案来实现。采用模块化设计方法即从整个系统的功能出发，将系统的整体逐步分解为若干个子系统和模块，然后用 VHDL 语言对各个模块进行 component 编程，最后形成顶层文件，在 Quartus Ⅱ环境下进行编译与仿真，检查所编程序是否运行正确。如果出现错误，需要进行修改，直到完全通过为止。最后在电路板上下载进行调试。

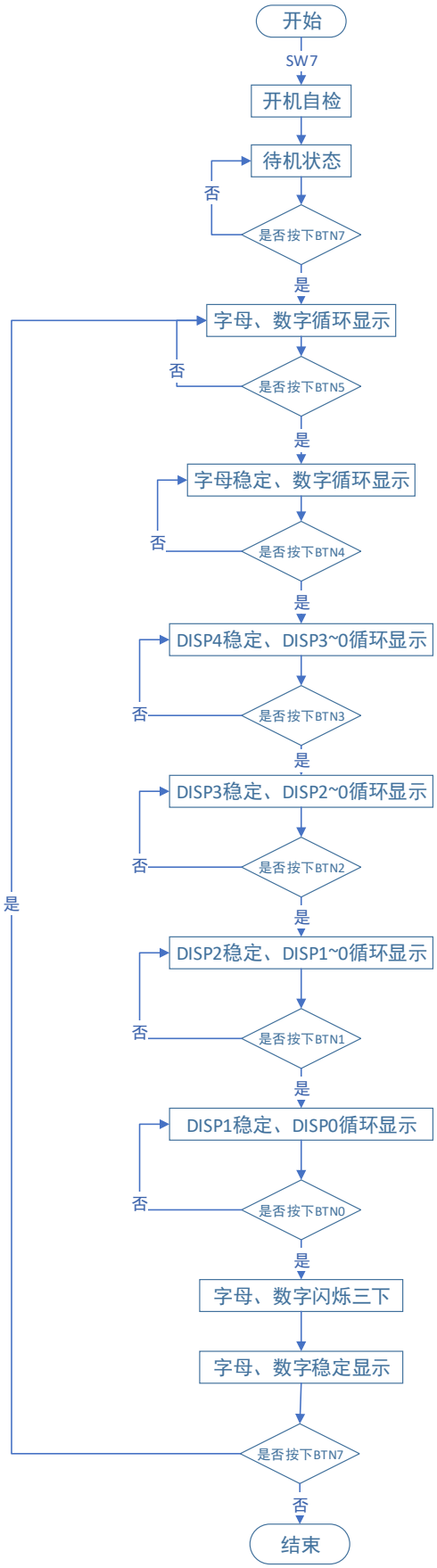
采用模块化设计的优点在于：对设计的描述从上到下逐步由粗略到详细，符合常规的逻辑思维习惯。便于由多个设计者同时进行设计从而加速整个项目的开发进度；每个子模块都能够灵活使用综合和实现工具独立进行优化，从而达到更好的优化结果；调试、更改某个子模块时，不会影响其他模块的实现结果，保证了整个设计的稳定性。

2. 总体框图

模块划分



系统结构图（ASM 图）


开始
SW7
开机自检
待机状态
否
是否按下BTN7
是
字母、数字循环显示
否
是否按下BTN5
是
字母稳定、数字循环显示
否
是否按下BTN4
是
DISP4稳定、DISP3~0循环显示
否
是否按下BTN3
是
DISP3稳定、DISP2~0循环显示
否
是否按下BTN2
是
DISP2稳定、DISP1~0循环显示
否
是否按下BTN1
是
DISP1稳定、DISP0循环显示
否
是否按下BTN0
是
字母、数字闪烁三下
字母、数字稳定显示
是否按下BTN7
是
否
结束

3. 分块设计

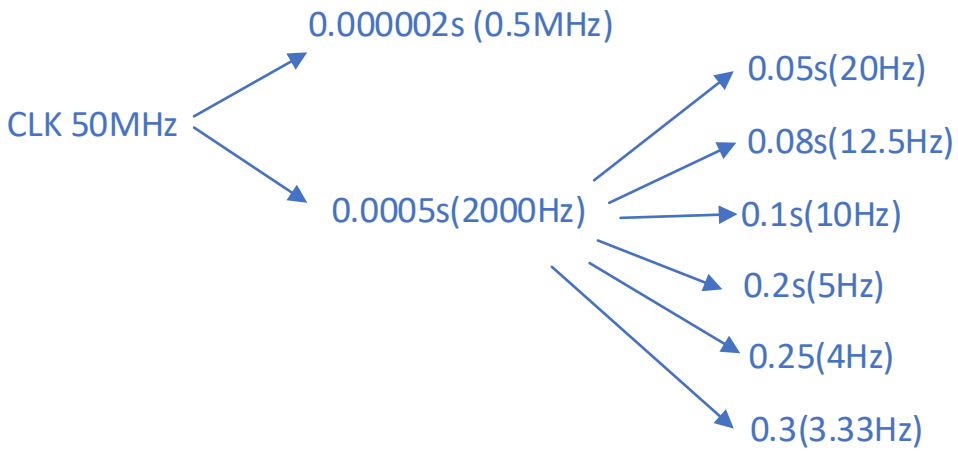
分频模块

各个部分所需频率如下表：

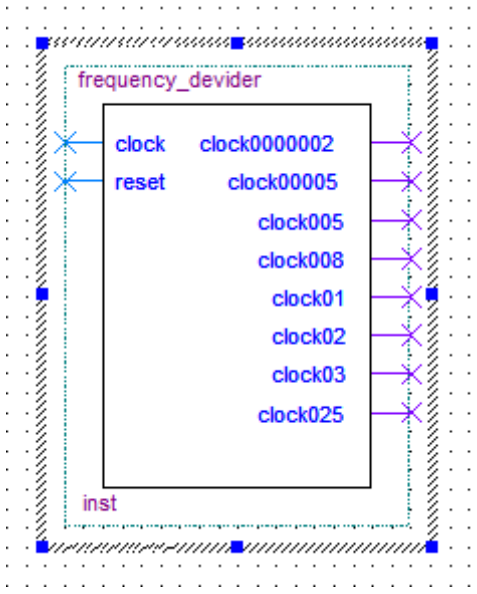
实现功能	所需频率
开机自检	1Hz
字母选择	2Hz
第一个数字选择	3.33Hz
第二个数字选择	5Hz
第三个数字选择	10Hz
第四个数字选择	12.5Hz
第五个数字选择	20Hz
闪烁三次以示提醒	2Hz

通过已确定的频率计算分频系数，设置一个计数变量，数到第（分频系数/2）个高频时钟时，使输出时钟翻转，计数变量归零，以此循环往复，得到所需的低频时钟。

分频模块流程图如图所示：



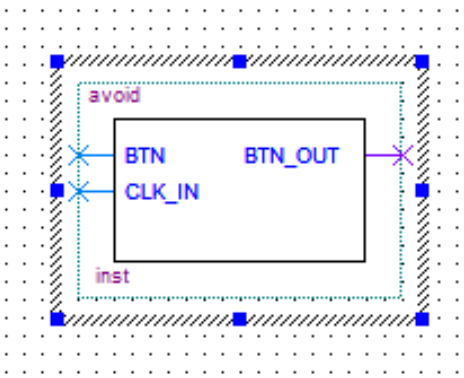
分频模块的模块符号图形如下图：



防抖模块

此防抖模块需要加入到所有按键输入的后一级，以使得每一次从按键输入的电平都能转换成持续时间稳定的高电平，从而使得元件内部的信号频率稳定，有利于整个程序的良好运行。由于一共使用了 7 个按键，BTN7 控制选号状态的开启与重新开始游戏，BTN5 控制点阵中字母的选择，BTN4~0 控制数码管中数字的选择，所以需要多个防抖模块。

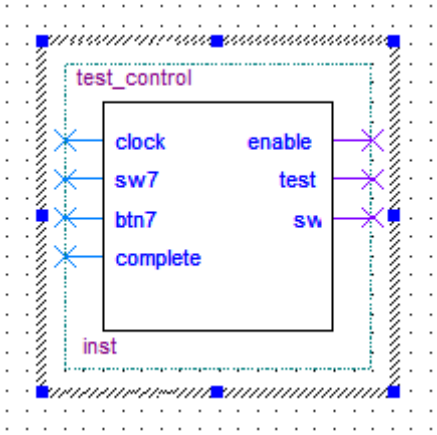
avoid 模块符号图如下：



控制模块

本程序通过 test_crontrl 控制模块来实现选号机各个状态及状态间的转移。选号机共有关机状态、开机自检状态、待机状态、选号状态、选号后闪烁提醒状态和稳定显示状态。

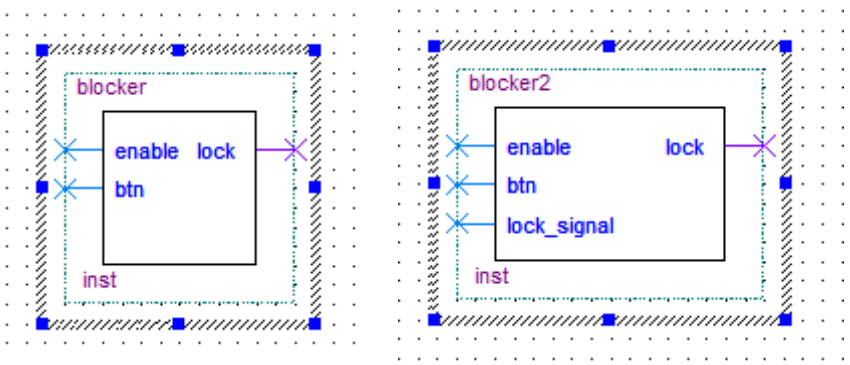
模块符号图如下：



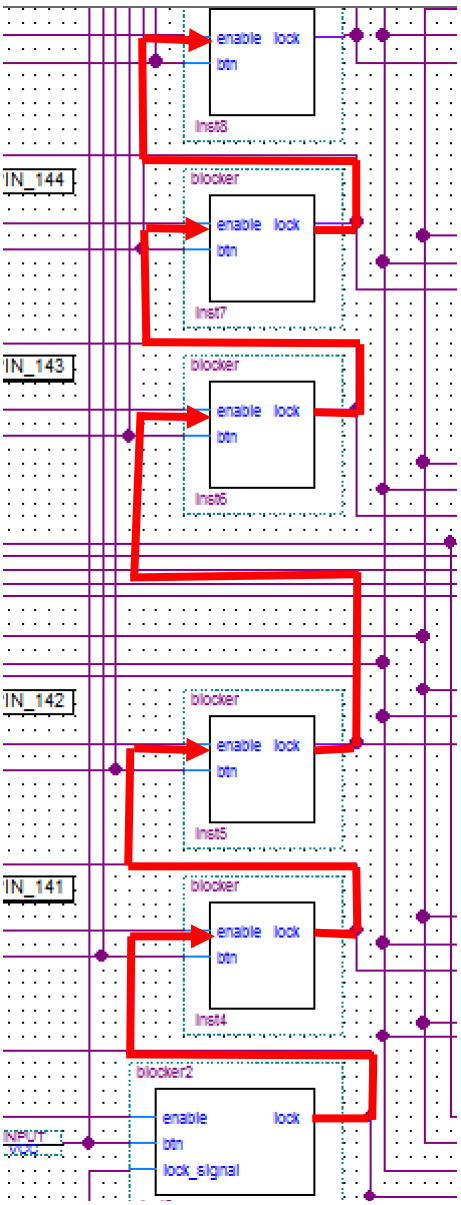
顺序控制模块

根据实验要求，使用 BTN7 进入选号状态后，应按照顺序选号，当前面号码未选定时，后面的无效。本程序通过 blocker&blocker2 模块来实现此功能。将前一个模块的 lock 输出作为下一个 blocker 模块的使能端输入(enable)，当前一个字母或数字未选定时，lock=0，下一模块的使能端输入为 0，信号被锁住，无法继续传播；当前一个字母或数字已选定，lock=1，使能端输入为 1，在按键同时为 1 的情况下实现字母或数字的选择。

blocker&blocker2 模块符号图如下：



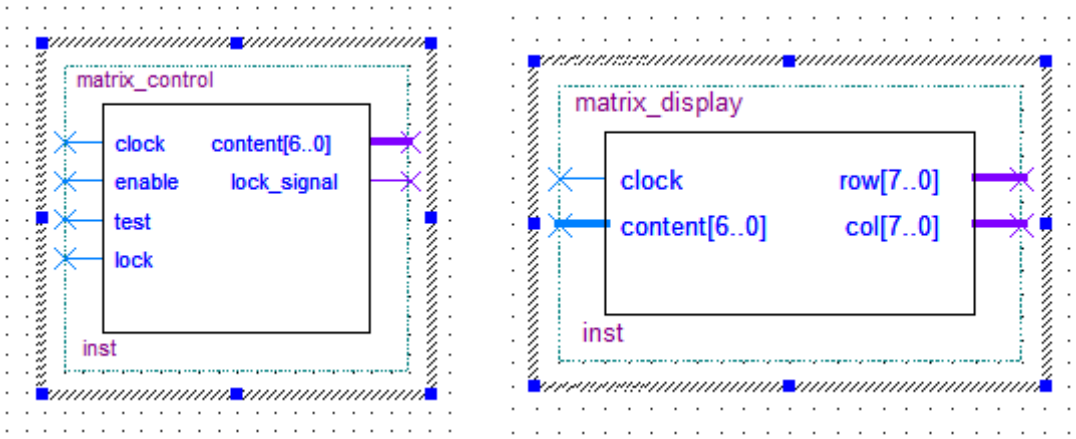
信号的传输路径示意图如下：



点阵模块

点阵模块包括 matrix_control 和 matrix_display 两个子模块。matrix_display 模块定义了矩阵的图形显示，即全亮、全暗和字母 A-F 之间的循环,通过高频率扫描每一行,利用人眼的辨识误差通过动态扫描显示静态内容。matrix_control 模块定义了各个状态之间的转换，并采用矩阵下标索引的方式来查找显示区域，将相互分离的字母选择和显示模块关联起来，成功实现了字母自上而下的滚动功能。

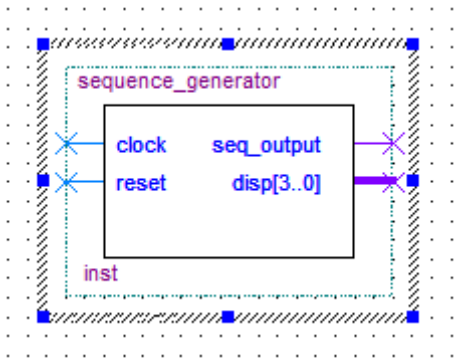
点阵模块符号图如下：



数码管模块

数码管模块包括 sequence_generator、disp_control、disp_control2、disp_display 四个子模块。sequence_generator 为随机序列发生器，产生 100010000 的序列，并以脉冲上升沿为新的时钟进行 0-9 的转换。因为序列时钟频率为 50MHz，远高于各个数码管数字切换的频率，故无法判断各个数码管每个时刻得到的数字，从而实现生成随机数的效果。

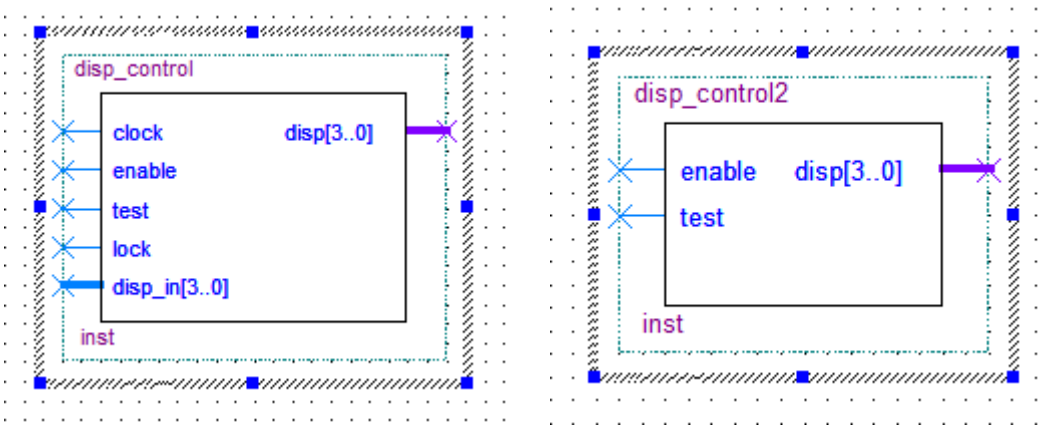
sequence_generator 模块符号图如下：



disp_control 和 disp_control2 用于控制 8 个数码管，其中 disp_control 用于控制选择数字的 5 位数码管，disp_control2 用于控制不需要选择数字，仅在开机自检和完成选号的提醒时闪烁功能的 3 个数码管。disp_control2 通过 enable 和 test 两个输入确定选号机所处状态，在自检和选定号码后提醒时闪烁三下，其余时间保持熄灭。disp_control 通过 enable, test, lock 输入控制数码管的状态。当 enable=0 时数码管熄灭；enable=1, test=1 时闪烁；

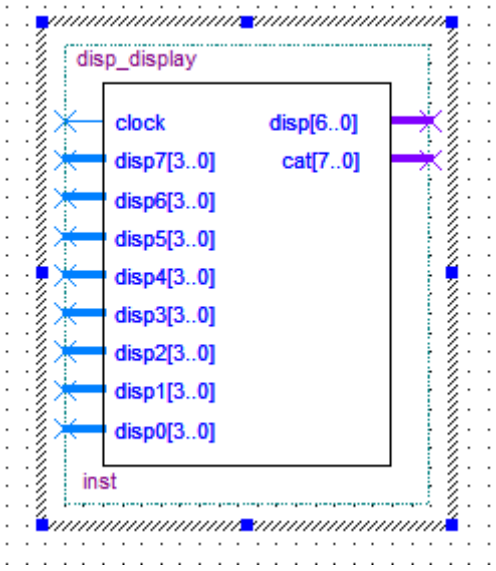
enable=1,test=0,lock=0 时保持选号状态，控制数码管按照频率随机显示号码；enable=1,test=0,lock=1 时表示为已选定状态，数字保持不变。

disp_control&disp_control2 模块符号图如下:



disp_display 模块用于数码管的译码显示。其定义了全部点亮、全部熄灭、显示数字 1-9 的显示控制，通过 test_control 和 disp_control 实现输出转换。在 disp_display 模块中，首先进行位选，使用 2000Hz 的频率扫描数码管，利用视觉暂留达到选号的效果。再进行段选，译码输出数码管显示的数字，实现随机显示下一位数字的功能。

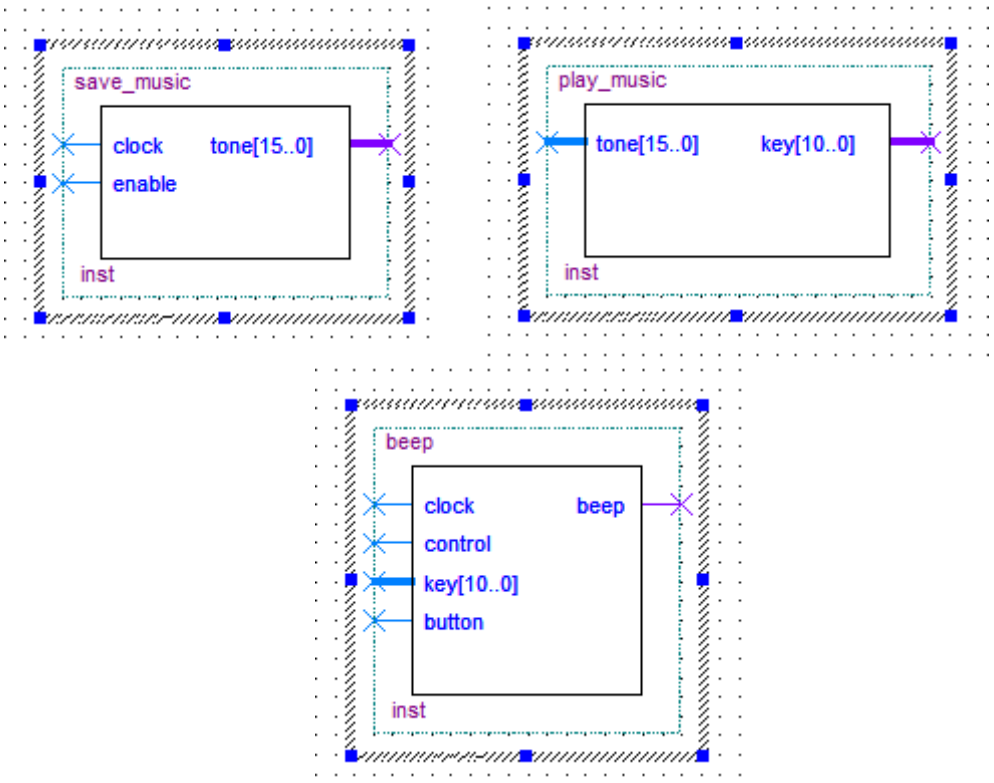
disp_display 模块符号图如下:



音乐模块

音乐模块包括 save_music、play_music 和 beep 三个子模块。save_music 模块定义了 60 个状态及每个状态对应的音调(tune)，音乐《蓝精灵》共有 17 个音调，play_music 定义每个音调对应的频率，这两个子模块实现了音乐的存入及播放。Beep 模块控制按键音和蜂鸣器。按键音的输入信号为各个按键输入信号的或运算，及只要有按键输入就有按键音提醒。按键音对应的频率为 214Hz。在 beep 模块中通过条件判断语句实现按键音和音乐的交错播放。如果按下按键，则蜂鸣器发出按键音，否则仍按照 play_music 文件中对应的音调频率播放音乐。通过计数器的状态循环实现音乐和按键音的循环往复播放，增强了选号机的交互性和趣味性。

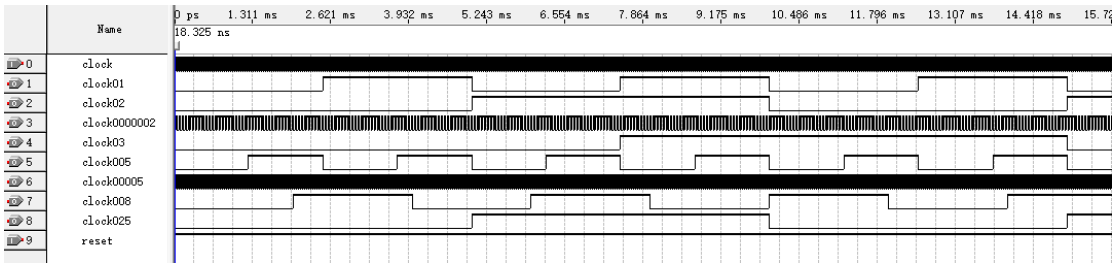
save_music,play_music 和 beep 模块的符号示意图如下



三、仿真波形及波形分析

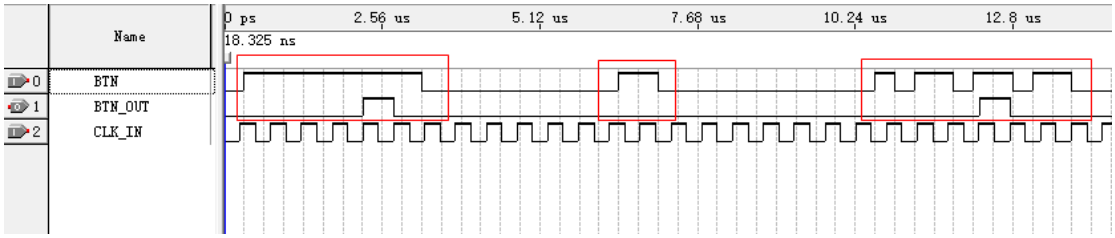
1. 分频模块仿真

（为了减少仿真时间，用小分频系数验证分频部分代码的正确性，下载时再调整到需要频率的分频系数）



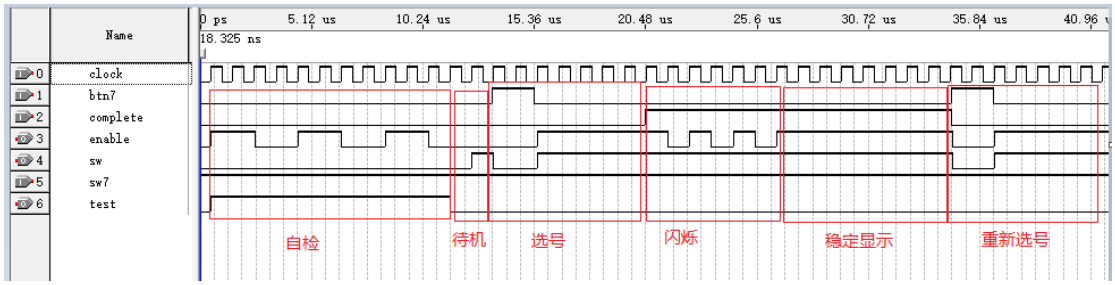
波形分析： clk0000002（0.5MHz）和 clk00005（2kHz）由系统时钟分别 100 分频、25000 分频（仿真时改为 50 分频）而来，clk005(20Hz)、clk008(12.5Hz)、clk01(10Hz)、clk02(5Hz)、clk025(4Hz)、clk03(3.33Hz)则由 clk00005(2kHz)分别 100 分频、160 分频、200 分频、400 分频、500 分频、600 分频而来。

2. 防抖模块仿真



仿真分析：消抖的原理就是把一个按键周期内所输入的所有有效信号，包括那些毛刺，处理成一个脉冲输出。由上图可以看出，当有按键按下时，该模块会根据时钟信号以及按下低电平的持续时间来判断是否有按键按下，并且输出稳定的方波作为其他模块的输入。

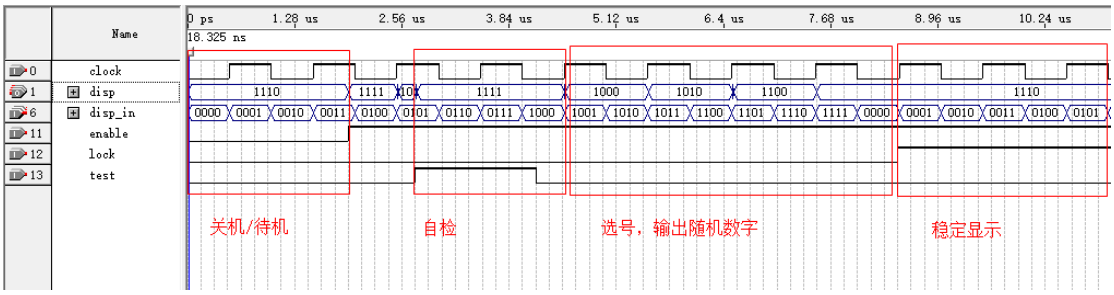
3. 控制模块仿真



波形分析：控制模块的时钟信号是 4Hz。打开开关 SW7（sw7=1），系统进入自检状态，以 1Hz 的频率闪烁三次，然后进入待机状态，按下 BTN77 后（btn7=1）进入选号状态，选号完成后（complete=1），以 2Hz 的频率闪烁三次，然后进入稳定显示状态，此时再次按下 BTN7 重新进入选号状态。

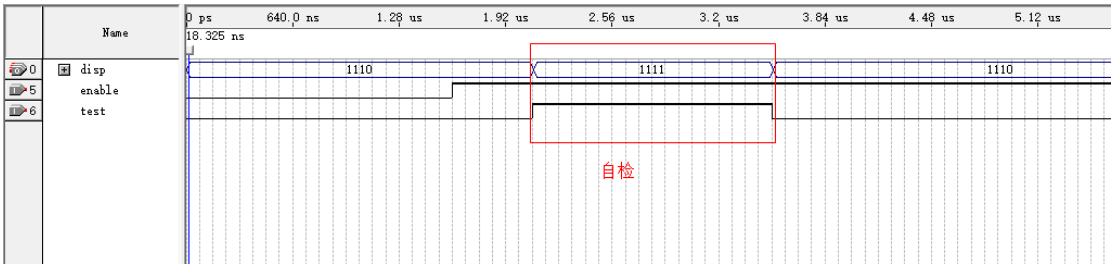
4. 数码管模块仿真

disp_control(控制 disp0~disp4):



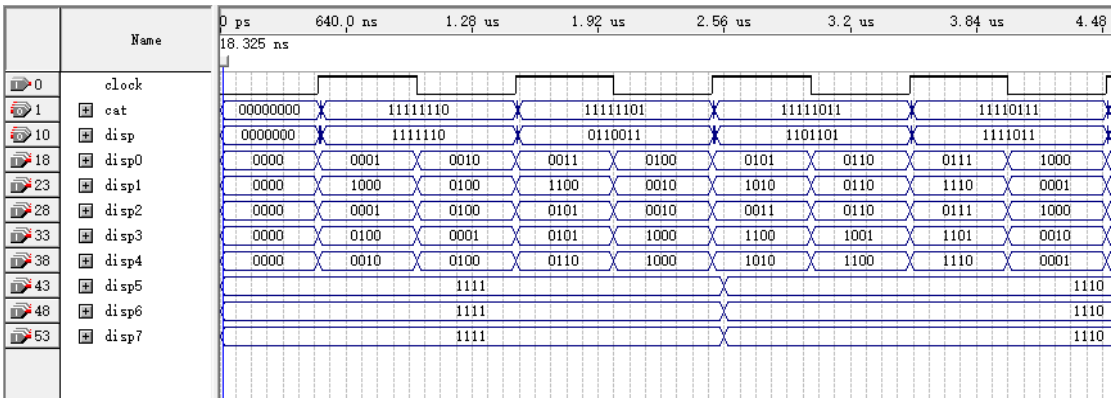
波形分析：当使能信号失效（enable=0）时，输出 1110，使数码管熄灭；当 enable=1&test=1 时，进入自检状态，数码管输出 1111；当 enable=1&test=0 时，进入选号状态，输出为随机数字；当 lock=1 时，数码管稳定显示最后一个随机数字。

disp_control2(控制 disp5~disp7):



波形分析:当且仅当 enable=1&test=1 时,进入自检状态,所以数码管输出 1111;其他情况下该数码管恒输出 1110，使数码管熄灭。

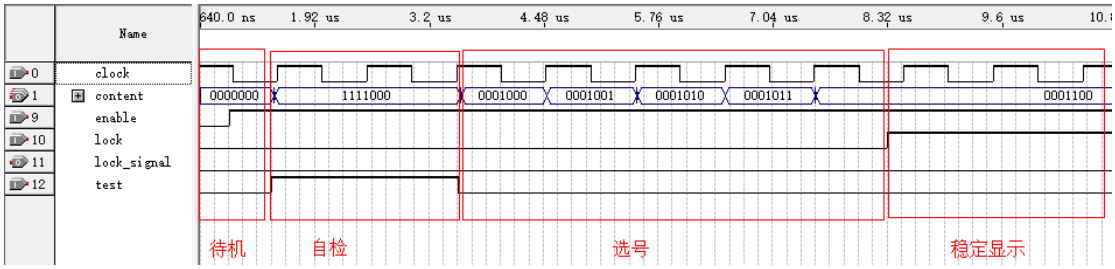
disp_display:



波形分析：每次扫描时只位选一位数码管输出，但由于不同数码管频率不同，依据人眼视觉停留效果可看到五位数码管同时点亮。

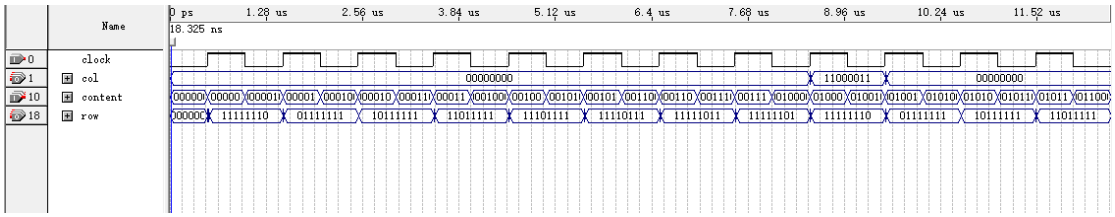
5. 点阵模块仿真

matrix_control:



波形分析：由上图可看出，当使能信号无效（enable=0）时，点阵熄灭；当 enable=1&test=1 时进入自检状态，点阵全亮；当 enable=1&test=0 时进入选号状态，循环显示字母；当 lock=1 时，点阵稳定显示最后一个字母。

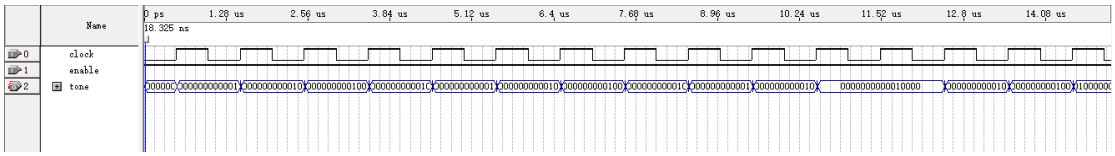
matrix_display:



波形分析：每次扫描行时只有一行有效，但扫描频率（2kHz）大于人眼视觉感知频率，所以可看到点亮的字母。

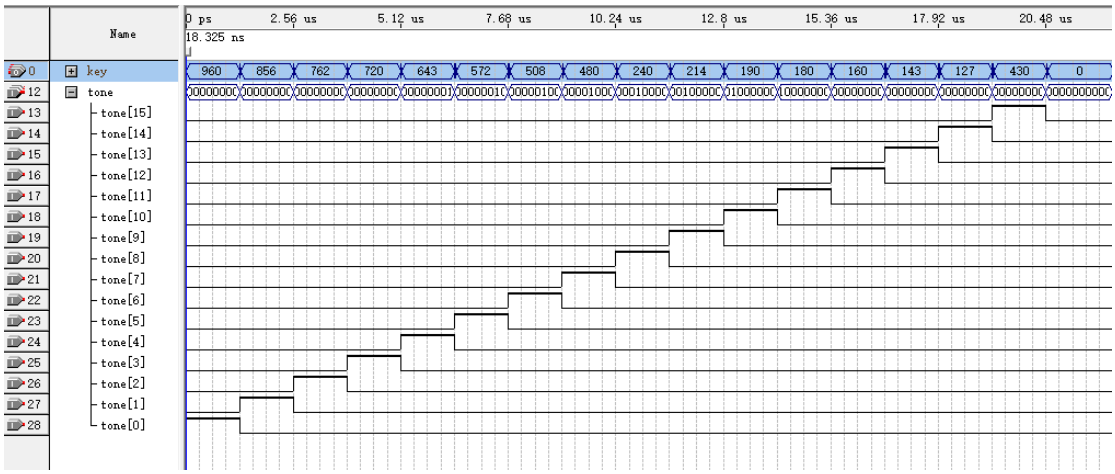
6. 音乐模块仿真

save_music:



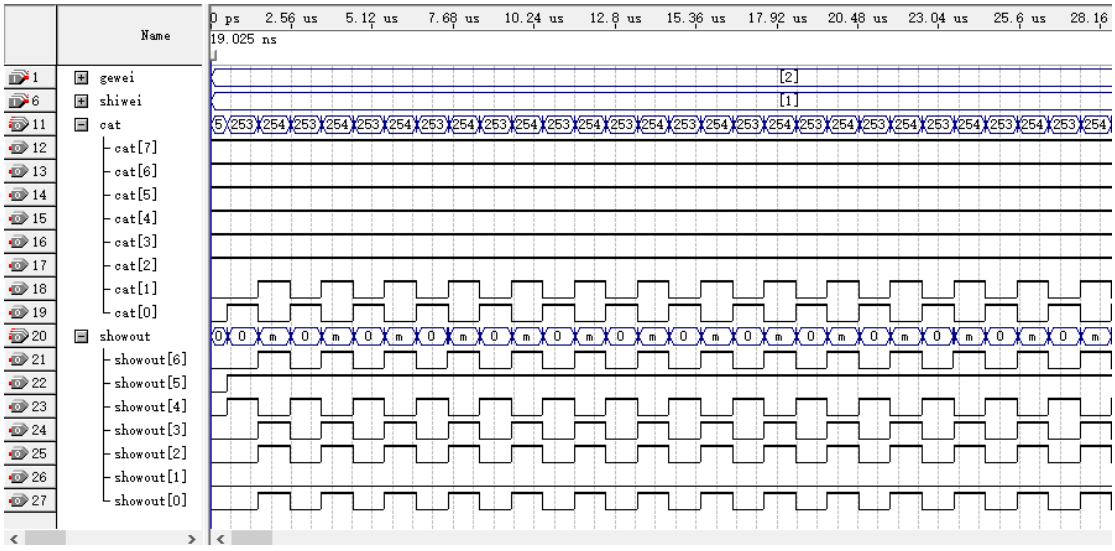
波形分析：save_music 模块输入时钟为 3.33Hz，控制音乐节拍，保存乐曲《蓝精灵》的 60s 乐谱，输出乐谱调子对应的二进制表示，作为下一个模块 play_music 的输入。

play_music:



波形分析： play_music 将 save_music 的输出作为输入，将其译码成相应的频率，一个有 17 个频率，包括按键音的频率。

8. 数码管模块仿真



波形分析：由波形图可知，设定显示输入控制十位为 1，个位为 2，选通信号 cat 和 showout 七段码信号都输出正常。

四、源程序

1. 分频模块

```
--------------------------------------------------------------------------------
--frequency devider
--input     50MHz
--output T=     0.0005s
--              0.05s
--              0.08s
--              0.1s
--              0.2s
--              0.3s
--              0.5s
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
--head

library ieee;
use ieee.std_logic_1164.all;

--head
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
--entity

entity frequency_devider is
   port (
      --in
      clock:in std_logic;         --clock    50MHz
      reset:in std_logic;         --sw79

      --out

      clock0000002:out std_logic; --0.000002s 0.5MHz
      clock00005:out std_logic;      --0.0005s    2000Hz
      clock005:out std_logic;         --0.05s    20Hz
      clock008:out std_logic;         --0.08s    12.5Hz
      clock01:out std_logic;          --0.1s    10Hz
      clock02:out std_logic;          --0.2s    5Hz
      clock03:out std_logic;          --0.3s    3.3Hz
      clock025:out std_logic          --0.25s     4Hz
   ) ;
end entity ; -- frequency_devider
```

```
--entity
-------------------------------------------------------------------------


-------------------------------------------------------------------------
--architecture

architecture arch of frequency_devider is
    -------------------------------------------------------------------------
    --signal
    --0.000002s    0.5MHz
    signal counter0000002:integer range 0 to 49;
    signal clk0000002:std_logic;
    --0.0005s    2000Hz
    signal counter00005:integer range 0 to 12499;
    signal clk00005:std_logic;
    --0.05s    20Hz
    signal counter005:integer range 0 to 49;
    signal clk005:std_logic;
    --0.08s    12.5Hz
    signal counter008:integer range 0 to 79;
    signal clk008:std_logic;
    --0.1s    10Hz
    signal counter01:integer range 0 to 99;
    signal clk01:std_logic;
    --0.2s    5Hz
    signal counter02:integer range 0 to 199;
    signal clk02:std_logic;
    --0.25s    4Hz
    signal counter025:integer range 0 to 249;
    signal clk025:std_logic;
    --0.3s    3.3Hz
    signal counter03:integer range 0 to 299;
    signal clk03:std_logic;
    --signal
    -------------------------------------------------------------------------
begin

    --frequency devider 0.000002s 0.5MHz
    devider000002 : process( reset,clock )
    begin
        if reset='0' then
            counter0000002<=0;
        elsif clock'event and clock='1' then
            if counter0000002=49 then
                counter0000002<=0;
                clk0000002<=not clk0000002;
            else
```

```
                counter0000002<=counter0000002+1;
            end if ;
        end if ;
        clock0000002<=clk0000002;
    end process ; -- devider000002


    --frequency devider 0.0005s 2000Hz
    devider0005 : process( reset,clock )
    begin
        --if reset='0' then
            --counter00005<=0;
        if clock'event and clock='1' then
            if counter00005=12499 then
                counter00005<=0;
                clk00005<=not clk00005;
            else
                counter00005<=counter00005+1;
            end if ;
        end if ;
        clock00005<=clk00005;
    end process ; -- devider0005

    --frequency devider 0.05 20Hz
    devider005 : process( reset,clk00005 )
    begin
        if reset='0' then
            counter005<=0;
        elsif clk00005'event and clk00005='1' then
            if counter005=49 then
                counter005<=0;
                clk005<=not clk005;
            else
                counter005<=counter005+1;
            end if ;
        end if ;
        clock005<=clk005;
    end process ; -- devider005

    --frequency devider 0.08 12.5Hz
    divider008 : process( reset,clk00005 )
    begin
        if reset='0' then
            counter008<=0;
        elsif clk00005'event and clk00005='1' then
            if counter008=79 then
                counter008<=0;
                clk008<=not clk008;
```

```
            else
                counter008<=counter008+1;
            end if ;
        end if ;
        clock008<=clk008;
    end process ; -- devider008

    --frequency devider 0.1s    10Hz
    divider01 : process( reset,clk00005 )
    begin
        if reset='0' then
            counter01<=0;
        elsif clk00005'event and clk00005='1' then
            if counter01=99 then
                counter01<=0;
                clk01<=not clk01;
            else
                counter01<=counter01+1;
            end if ;
        end if ;
        clock01<=clk01;
    end process ; -- devider01

    --frequency devider 0.2s    5Hz
    divider02 : process( reset,clk00005 )
    begin
        if reset='0' then
            counter02<=0;
        elsif clk00005'event and clk00005='1' then
            if counter02=199 then
                counter02<=0;
                clk02<=not clk02;
            else
                counter02<=counter02+1;
            end if ;
        end if ;
        clock02<=clk02;
    end process ; -- devider02

    --frequency devider 0.25s    4Hz
    divider025 : process( reset,clk00005 )
    begin
        if reset='0' then
            counter025<=0;
        elsif clk00005'event and clk00005='1' then
            if counter025=199 then
                counter025<=0;
                clk025<=not clk025;
```

```
                else
                    counter025<=counter025+1;
                end if ;
            end if ;
            clock025<=clk025;
        end process ; -- devider025

        --frequency devider 0.3s    3.3Hz
        divider03 : process( reset,clk00005 )
        begin
            if reset='0' then
                counter03<=0;
            elsif clk00005'event and clk00005='1' then
                if counter03=299 then
                    counter03<=0;
                    clk03<=not clk03;
                else
                    counter03<=counter03+1;
                end if ;
            end if ;
            clock03<=clk03;
        end process ; -- devider03

end architecture arch;

--architecture
-----------------------------------------------------------------
```

2. 控制模块

```
--------------------------------------------------------------------------------
--test_control
--input sw7 btn7

--output test enable
--sw7 for switch
--btn7 for reset

--complete for finish twinkle
--test: test=1 & send 3 pulses enable
--then keep enable=1

--complete:send 3 pulses enable
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
--head
library ieee;
```

```
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
--head
----------------------------------------------------------------------------


----------------------------------------------------------------------------
--entity
	entity test_control is
	   port (
		--in
		clock:in std_logic;		--0.25s    4Hz
		sw7:in std_logic;		--switch
		btn7:in std_logic;		--reset
		complete:in std_logic;	--connect to the lase lock

		--out
		enable:out std_logic;
		test:out std_logic;		--test enable
		sw:out std_logic
	   ) ;
	end entity ; -- test_control
--entity
----------------------------------------------------------------------------


----------------------------------------------------------------------------
--architecture
	architecture arch of test_control is
	-----------------------------------------------------------------------
		--signal
		signal btn7_re:std_logic;
		signal self_test_re:std_logic;

		--counter
		signal count:integer range 0 to 12;
		signal count_for_complete:integer range 0 to 6;

		--complete_re
		signal complete_re:std_logic;

		--signal
	-----------------------------------------------------------------------
	begin
	-----------------------------------------------------------------------
		--processer
		processer : process( complete,clock,btn7,sw7 )
		begin
			if sw7='0' then
				enable<='0';
```

```
                count<=0;
                count_for_complete<=0;
                test<='0';
                btn7_re<='0';
                self_test_re<='0';
                sw<='0';
            elsif btn7='1' then
                enable<='0';
                count_for_complete<=0;
                btn7_re<='1';
                sw<='0';
            elsif clock'event and clock='1' then
                if self_test_re='0' then
                    case (count) is
                        when 0 => count<=1; enable<='1'; test<='1';
                        when 1 => count<=2; enable<='1';
                        when 2 => count<=3; enable<='0';
                        when 3 => count<=4; enable<='0';
                        when 4 => count<=5; enable<='1';
                        when 5 => count<=6; enable<='1';
                        when 6 => count<=7; enable<='0';
                        when 7 => count<=8; enable<='0';
                        when 8 => count<=9; enable<='1';
                        when 9 => count<=10; enable<='1';
                        when 10 => count<=11; enable<='0';
                        when 11 => count<=12; enable<='0';test<='0';
                        when others => self_test_re<='1'; sw<='1';
                    end case;
                elsif btn7_re='0' then
                    enable<='0';
                elsif complete='1' then
                    case (count_for_complete) is
                        when 0 => count_for_complete<=1; enable<='1';
                        when 1 => count_for_complete<=2; enable<='0';
                        when 2 => count_for_complete<=3; enable<='1';
                        when 3 => count_for_complete<=4; enable<='0';
                        when 4 => count_for_complete<=5; enable<='1';
                        when 5 => count_for_complete<=6; enable<='0';
                        when others => enable<='1';
                    end case;
                else
                    enable<='1';
                    sw<='1';
                end if ;
            end if ;
        end process ; -- processer
        --processer
-----------------------------------------------------------------------
```

```
    end architecture ; -- arch
--architecture
-------------------------------------------------------------------------------
```

3. 顺序控制模块

blocker

```
-------------------------------------------------------------------------------
--blocker
--input enable btn
--          set lock=1 when enable=1 and btn=1

--output lock
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
--head
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
--head
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
--entity

entity blocker is
  port (
    --in
    enable:in std_logic;
    btn:in std_logic;

    --out
    lock:out std_logic
  ) ;
end entity ; -- blocker

--entity
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
--architecture

architecture arch of blocker is
```

```
begin

    ---------------------------------------------------------------------------
    --processer

    processer : process( enable,btn )
    begin
        if enable='0' then
            lock<='0';
        elsif btn='1' then
            lock<='1';
        end if ;
    end process ; -- processer

    --processer
    ---------------------------------------------------------------------------
end architecture ; -- arch
--architecture
-------------------------------------------------------------------------------
```

blocker2

```
-------------------------------------------------------------------------------
--blocker2

--input enable btn
--          set lock=1 when enable=1 and btn=1
--          keep lock=1 when enable=1 and lock_signal=1

--output lock
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
--head
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
--head
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
--entity
entity blocker2 is
  port (
    --in
```

```
        enable:in std_logic;
        btn:in std_logic;
        lock_signal:in std_logic;

        --out
        lock:out std_logic
    ) ;
end entity ; -- blocker2

--entity
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
--architecture
architecture arch of blocker2 is
        --------------------------------------------------------------------------
        --signal
        signal lock_flag:std_logic;
        --signal
        --------------------------------------------------------------------------

begin
        --------------------------------------------------------------------------
        --processer
        processer : process( enable,btn,lock_signal )
        begin
                if enable='0' then
                        lock<='0';
                        lock_flag<='0';
                elsif lock_signal='1' then
                        lock<='1';
                        lock_flag<='1';
                elsif lock_flag='1' then
                        lock<='1';
                elsif btn='1' then
                        lock<='1';
                else
                        lock<='0';
                end if ;
        end process ; -- processer
        --processer
        --------------------------------------------------------------------------
end architecture ; -- arch
--architecture
--------------------------------------------------------------------------------
```

4. 点阵模块

matrix_control

```
----------------------------------------------------------------------------
--matrix_control
--input enable test lock
--          enable    0     dis
--                    test  1     test signal
--                                lock 1     keep signal
--                                           0     circulation

--output content
----------------------------------------------------------------------------


----------------------------------------------------------------------------
--head
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
--head
----------------------------------------------------------------------------

----------------------------------------------------------------------------
--entity
entity matrix_control is
   port (
      --in
      clock:in std_logic;         --clock 0.5s 2Hz
      enable:in std_logic;
      test:in std_logic;
      lock:in std_logic;

      --out
      content:out std_logic_vector(6 downto 0);
      lock_signal:out std_logic       --lock_feedback
   ) ;
end entity ; -- matrix_control
--entity
----------------------------------------------------------------------------

----------------------------------------------------------------------------
--architecture
      architecture arch of matrix_control is
      ------------------------------------------------------------------
            --signal
```

```
        --counter
        signal count:integer range 0 to 127;       --for content
        --signal
    ------------------------------------------------------------------------
    begin
        ------------------------------------------------------------------------
        --processer
        processer : process( enable,test,lock,clock )
        begin
            if enable='0' then
                content<="0000000";
                lock_signal<='0';
            else
                if test='1' then
                    count<=120;
                else
                    if lock='0' then
                        if count=120 then
                            count<=8;
                        elsif count=104 then
                            count<=8;
                        elsif clock'event and clock='1' then
                            count<=count+1;
                        end if ;
                    elsif count rem 16=0 then
                        lock_signal<='1';
                    end if ;
                end if ;
                content<=conv_std_logic_vector(count,7);
            end if ;
        end process ; -- processer
        --processer
    ------------------------------------------------------------------------
    end architecture ; -- arch
--architecture
```

matrix_display

```
--------------------------------------------------------------------------------
--matrix_display

--input     clock & content
--content: all    1111000
--                dis   0000000
--                A-F 0001000-1101000

--output matrix
--------------------------------------------------------------------------------
```

```
----------------------------------------------------------------------------
--head
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
--head
----------------------------------------------------------------------------
----------------------------------------------------------------------------
--entity
entity matrix_display is
   port (
      --in
      clock:in std_logic;          --clock 0.005s 200Hz
      content:in std_logic_vector(6 downto 0);      --content
      --out
      --row       low level effective
      row:out std_logic_vector(7 downto 0);  --up downto down
                                                    --pin1 to pin8
      --col        high level effective
      col:out std_logic_vector(7 downto 0)    --left downto right
                                                    --pin22 to pin11
   ) ;
end entity ; -- matrix_display
--entity
----------------------------------------------------------------------------


----------------------------------------------------------------------------
--architecture
architecture arch of matrix_display is
      ----------------------------------------------------------------------------
      --signal
      --matrix    all downto dis
      type table_type is array (127 downto 0) of std_logic_vector(7 downto 0);
      constant table : table_type :=
      (
            x"FF",    x"FF",    x"FF",    x"FF",    x"FF",    x"FF",    x"FF",    x"FF",    --all
            x"00",    x"00",    x"00",    x"00",    x"00",    x"00",    x"00",    x"00",    --empty
            x"00",    x"00",    x"00",    x"00",    x"00",    x"00",    x"00",    x"00",    --empty
            ------------------circulation(down to up)----------------------------
            x"7E",    x"7E",    x"60",    x"7E",    x"7E",    x"60",    x"60",    x"60",    --F
            x"00",    x"00",    x"00",    x"00",    x"00",    x"00",    x"00",    x"00",    --empty
            x"7E",    x"7E",    x"60",    x"7E",    x"7E",    x"60",    x"7E",    x"7E",    --E
            x"00",    x"00",    x"00",    x"00",    x"00",    x"00",    x"00",    x"00",    --empty
            x"FC",    x"FE",    x"C6",    x"C3",    x"C3",    x"C6",    x"FE",    x"FC",    --D
            x"00",    x"00",    x"00",    x"00",    x"00",    x"00",    x"00",    x"00",    --empty
            x"1E",    x"3E",    x"70",    x"60",    x"60",    x"70",    x"3E",    x"1E",    --C
```

```
        x"00",   x"00",   x"00",   x"00",   x"00",   x"00",   x"00",   x"00",   --empty
        x"7C",   x"66",   x"66",   x"7C",   x"7C",   x"66",   x"66",   x"7C",   --B
        x"00",   x"00",   x"00",   x"00",   x"00",   x"00",   x"00",   x"00",   --empty
        x"18",   x"3C",   x"66",   x"66",   x"7E",   x"FF",   x"C3",   x"C3",   --A
        x"00",   x"00",   x"00",   x"00",   x"00",   x"00",   x"00",   x"00",   --empty
        ------------------------------------------------------------------------
        x"00",   x"00",   x"00",   x"00",   x"00",   x"00",   x"00",   x"00"    --dis
    );

    --counter for row
    signal count:integer range 0 to 7;   --down to up
    --signal
    ---------------------------------------------------------------------------
begin
    ---------------------------------------------------------------------------
    --counter
    counter : process( clock )
    begin
        if clock'event and clock='1' then
            if count=0 then
                count<=7;
            else
                count<=count-1;
            end if ;
        end if ;
    end process ; -- counter
    --counter
    ---------------------------------------------------------------------------
    ---------------------------------------------------------------------------
    --display_col
    display_col : process( clock )
    begin
        if clock'event and clock='1' then
            col<=table(conv_integer(conv_integer(count+conv_integer(content))));
            case (count) is
                when 7 => row<="01111111";
                when 6 => row<="10111111";
                when 5 => row<="11011111";
                when 4 => row<="11101111";
                when 3 => row<="11110111";
                when 2 => row<="11111011";
                when 1 => row<="11111101";
                when 0 => row<="11111110";

                when others => row<="11111111";
            end case;
        end if ;
    end process ; -- display_col
```

```
    --display_col
    ---------------------------------------------------------------------
end architecture ; -- arch
--architecture
-------------------------------------------------------------------------------
```

5. 数码管模块

sequence_generator

```
-------------------------------------------------------------------------------
--m sequence generator
--decoder
--input clock

--output random disp
--disp:    all   1111
--         dis   1110
--         9     1001
--         .     .
--         .     .
--         0     0000
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
--head
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
--head
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
--entity
entity sequence_generator is
  port (
    clock:in std_logic;  --50M
    reset:in std_logic;

    --debug
    seq_output:out std_logic;

    disp:out std_logic_vector(3 downto 0)

  ) ;
end entity ; -- sequence_generator
--entity
```

```
----------------------------------------------------------------------------


----------------------------------------------------------------------------
--architecture
architecture arch of sequence_generator is
    ----------------------------------------------------------------------------
    --signal
    --m sequence
    signal seq:std_logic_vector(8 downto 0);

    --sequence output signal
    signal seq_out:std_logic;

    --disp_re
    signal disp_re:std_logic_vector(3 downto 0);

    --signal
    ----------------------------------------------------------------------------

begin
    ----------------------------------------------------------------------------
    --m sequence generator

    m_sequence_generator : process( clock )
    begin
        if reset='0' then
            seq<="000000001";
        elsif clock'event and clock='1' then
            seq(0)<=seq(3) xor seq(8);
            seq(1)<=seq(0);
            seq(2)<=seq(1);
            seq(3)<=seq(2);
            seq(4)<=seq(3);
            seq(5)<=seq(4);
            seq(6)<=seq(5);
            seq(7)<=seq(6);
            seq(8)<=seq(7);
        end if ;
        seq_out<=seq(8);
        seq_output<=seq_out;
    end process ; -- m_sequence_generator
    --m sequence generator
    ----------------------------------------------------------------------------


    ----------------------------------------------------------------------------
    --decoder
    decoder : process( seq_out )
    begin
```

```
        if seq_out'event and seq_out='1' then
            case (disp_re) is
                when "0000" => disp_re<="0001";
                when "0001" => disp_re<="0010";
                when "0010" => disp_re<="0011";
                when "0011" => disp_re<="0100";
                when "0100" => disp_re<="0101";
                when "0101" => disp_re<="0110";
                when "0110" => disp_re<="0111";
                when "0111" => disp_re<="1000";
                when "1000" => disp_re<="1001";
                when "1001" => disp_re<="0000";
                when others => disp_re<="0000";
            end case;
        end if ;
        disp<=disp_re;
    end process ; -- decoder
    --decoder
    ----------------------------------------------------------------------------
end architecture ; -- arch
--architecture
----------------------------------------------------------------------------
```

disp_control

```
----------------------------------------------------------------------------
--disp_control
--
--input --input enable test lock
--          enable    0     dis
--                    test  1     test signal
--                                lock 1     keep signal
--                                           0     circulation
--output disp[7 => 0]
----------------------------------------------------------------------------

----------------------------------------------------------------------------
--head
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
--head
----------------------------------------------------------------------------

----------------------------------------------------------------------------
--entity
```

```
entity disp_control is
   port (
      --in
      clock:in std_logic;  --change with disp
      enable:in std_logic;
      test:in std_logic;
      lock:in std_logic;
      disp_in:in std_logic_vector(3 downto 0);
      --out
      disp:out std_logic_vector(3 downto 0)          --disp[7 => 0]
   ) ;
end entity ; -- disp_control
--entity
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
--architecture
      architecture arch of disp_control is
            ------------------------------------------------------------------------
            --signal
            signal count:std_logic_vector(3 downto 0);   --for disp
            --signal
            ------------------------------------------------------------------------

      begin
            ------------------------------------------------------------------------
            --processer
            processer : process( enable,test,lock,clock )
            begin
                  if enable='0' then
                        disp<="1110";
                  else
                        if test='1' then
                              count<="1111";
                        else
                              if lock='0' then
                                    if clock'event and clock='1' then
                                          count<=disp_in;
                                    end if ;
                              end if ;
                        end if ;
                        disp<=count;
                  end if ;
            end process ; -- processer
            --processer
            ------------------------------------------------------------------------
      end architecture ;
-- architecture
```

```
--------------------------------------------------------------------------------
```

disp_control2

```
--------------------------------------------------------------------------------
--disp_control2
--for disp[7 => 5]
----input --input enable test lock
--            enable     0     dis
--                       test  1     test signal
--                                   0     dis

--output disp[7 => 0]
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
--head
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
--head
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
--entity
entity disp_control2 is
   port (
      --in
      enable:in std_logic;
      test:in std_logic;

      --out
      disp:out std_logic_vector(3 downto 0)          --disp[7 => 0]
   ) ;
end entity ; -- disp_control2
--entity
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
--architecture
architecture arch of disp_control2 is
begin
      --------------------------------------------------------------------------
      --processer
      processer : process( enable,test )
      begin
```

```
            if enable='0' then
                disp<="1110";
            else
                if test='1' then
                    disp<="1111";
                else
                    disp<="1110";
                end if ;
            end if ;
        end process ; -- processer
        --processer
        ----------------------------------------------------------------------
end architecture ; -- arch
--architecture
------------------------------------------------------------------------------
```

disp_display

```
------------------------------------------------------------------------------
--disp_display
--input clock disp[7]
--disp:     all   1111
--          dis   1110
--          9     1001
--          .     .
--          .     .
--          0     0000
--output disp cat
------------------------------------------------------------------------------


------------------------------------------------------------------------------
--head
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
--head
------------------------------------------------------------------------------


------------------------------------------------------------------------------
--entity
entity disp_display is
    port (
        --in
        clock:in std_logic;          --0.005s    200Hz
        disp7:in std_logic_vector(3 downto 0);  --disp
        disp6:in std_logic_vector(3 downto 0);  --disp
        disp5:in std_logic_vector(3 downto 0);  --disp
        disp4:in std_logic_vector(3 downto 0);  --disp
```

```
        disp3:in std_logic_vector(3 downto 0);  --disp
        disp2:in std_logic_vector(3 downto 0);  --disp
        disp1:in std_logic_vector(3 downto 0);  --disp
        disp0:in std_logic_vector(3 downto 0);  --disp

        --out
        disp:out std_logic_vector(6 downto 0);    --a downto g
        --cat low level effective
        cat:out std_logic_vector(7 downto 0)     --cat7 downto cat0
    ) ;
end entity ; -- disp_display
--entity
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
-- architecture
        architecture arch of disp_display is
            ------------------------------------------------------------------
            --signal
            --table    all downto 0
            type table_type is array(15 downto 0) of std_logic_vector(6 downto 0);
            constant table : table_type :=
            (
                "1111111",      --all 1111
                "0000000",      --dis       1110
                "0000000",      --dis       1101
                "0000000",      --dis       1100
                "0000000",      --dis       1011
                "0000000",      --dis       1010
                "1111011",      --9   1001
                "1111111",      --8   1000
                "1110000",      --7   0111
                "1011111",      --6         0110
                "1011011",      --5         0101
                "0110011",      --4         0100
                "1111001",      --3         0011
                "1101101",      --2         0010
                "0110000",      --1         0001
                "1111110"       --0         0000
            );
            --counter for cat
            signal count:integer range 0 to 7;
            --signal
            ------------------------------------------------------------------
        begin
            ------------------------------------------------------------------
            --counter
            counter : process( clock )
```

```
        begin
            if clock'event and clock='1' then
                if count=7 then
                    count<=0;
                else
                    count<=count+1;
                end if ;
            end if ;
        end process ; -- counter
        --counter
        ---------------------------------------------------------------------


        ---------------------------------------------------------------------
        --disp_display

        disp_display : process( clock )
        begin

            if clock'event and clock='1' then
                case (count) is
                    when 0 => cat<="11111110"; disp<=table(conv_integer(disp0));
                    when 1 => cat<="11111101"; disp<=table(conv_integer(disp1));
                    when 2 => cat<="11111011"; disp<=table(conv_integer(disp2));
                    when 3 => cat<="11110111"; disp<=table(conv_integer(disp3));
                    when 4 => cat<="11101111"; disp<=table(conv_integer(disp4));
                    when 5 => cat<="11011111"; disp<=table(conv_integer(disp5));
                    when 6 => cat<="10111111"; disp<=table(conv_integer(disp6));
                    when 7 => cat<="01111111"; disp<=table(conv_integer(disp7));

                    when others => cat<="11111111";
                end case;
            end if ;
        end process ; -- disp_display
        --disp_display
        ---------------------------------------------------------------------

    end architecture ; -- arch
--architecture
-------------------------------------------------------------------------------
```

6. 蜂鸣器模块

save_music

```
-------------------------------------------------------------------------------
--save music
--input clock enable
```

```
--output tone
---------------------------------------------------------------------------


---------------------------------------------------------------------------
--head
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
--head
---------------------------------------------------------------------------


---------------------------------------------------------------------------
--entity
entity save_music is
   port (
      --in
      clock:in std_logic;
      enable:in std_logic;

      --out
      tone:out std_logic_vector(15 downto 0)
   ) ;
end entity ; -- save_music
--entity
---------------------------------------------------------------------------


---------------------------------------------------------------------------
--architecture
architecture arch of save_music is
      ---------------------------------------------------------------------------
      --signal
      signal count: integer range 0 to 64 :=0;
      --signal
      ---------------------------------------------------------------------------
begin
      ---------------------------------------------------------------------------
      --play
      play : process( clock )
      begin
            if enable = '1' then
                  if clock'event and clock='1' then
                        if count < 61 then
                              case count is
                                    when 0 => tone <=b"0000000000000100";
                                    when 1 => tone <=b"0000000000001000";
                                    when 2 => tone <=b"0000000000010000";
```

```
when 3 => tone <=b"0000000000001000";
when 4 => tone <=b"0000000000000100";
when 5 => tone <=b"0000000000001000";
when 6 => tone <=b"0000000000010000";
when 7 => tone <=b"0000000000001000";
when 8 => tone <=b"0000000000000100";
when 9 => tone <=b"0000000000001000";
when 10 => tone <=b"0000000000010000";
when 11 => tone <=b"0000000000010000";
when 12 => tone <=b"0000000000001000";
when 13 => tone <=b"0000000000010000";
when 14 => tone <=b"0000010000000000";
when 15 => tone <=b"0000000100000000";
when 16 => tone <=b"0000000100000000";
when 17 => tone <=b"0000000000000010";
when 18 => tone <=b"0000000000000100";
when 19 => tone <=b"0000000000001000";
when 20 => tone <=b"0000000000000100";
when 21 => tone <=b"0000000000001000";
when 22 => tone <=b"0000001000000000";
when 23 => tone <=b"0000000001000000";
when 24 => tone <=b"0000000001000000";
when 25 => tone <=b"0000000000000010";
when 26 => tone <=b"0000000000000100";
when 27 => tone <=b"0000000000001000";
when 28 => tone <=b"0000000000000100";
when 29 => tone <=b"0000000000001000";
when 30 => tone <=b"0000000000100000";
when 31 => tone <=b"0000000000010000";
when 32 => tone <=b"0000000000010000";
when 33 => tone <=b"0000000000000100";
when 34 => tone <=b"0000000000001000";
when 35 => tone <=b"0000000000010000";
when 36 => tone <=b"0000000000001000";
when 37 => tone <=b"0000000000000100";
when 38 => tone <=b"0000000000001000";
when 39 => tone <=b"0000000000010000";
when 40 => tone <=b"0000000000001000";
when 41 => tone <=b"0000000000000100";
when 42 => tone <=b"0000000000001000";
when 43 => tone <=b"0000000000010000";
when 44 => tone <=b"0000000000010000";
when 45 => tone <=b"0000000000001000";
when 46 => tone <=b"0000000000010000";
when 47 => tone <=b"0000010000000000";
when 48 => tone <=b"0000001000000000";
when 49 => tone <=b"0000000000000010";
when 50 => tone <=b"0000000000000100";
```

```
                                when 51 => tone <=b"0000000000001000";
                                when 52 => tone <=b"0000000000000100";
                                when 53 => tone <=b"0000000000001000";
                                when 54 => tone <=b"0000001000000000";
                                when 55 => tone <=b"0000000100000000";
                                when 56 => tone <=b"0000000001000000";
                                when 57 => tone <=b"0000000000100000";
                                when 58 => tone <=b"0000000001000000";
                                when 59 => tone <=b"0000000010000000";
                                when others => tone <=b"0000000000000000";
                            end case;
                            count <= count+1;
                        else
                            count<=0;
                        end if;
                    end if;
                end if;
            end process ; -- play
            --play
            ---------------------------------------------------------------------
end architecture ; -- arch
--architecture
---------------------------------------------------------------------------
```

play_music

```
---------------------------------------------------------------------------
--play music
--input tone
--output key
--decoder
---------------------------------------------------------------------------

---------------------------------------------------------------------------
--head
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
--head
---------------------------------------------------------------------------

---------------------------------------------------------------------------
--entity
entity play_music is
   port(
      --in
```

```
        tone : in std_logic_vector(15 downto 0);

        --out
        key: out integer range 0 to 2047
    ) ;
end entity ; -- play_music
--entity
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
--architecture
architecture arch of play_music is
begin
        --------------------------------------------------------------------------
        --decoder
        process(tone)
            begin
                case tone is
                    when "0000000000000000"   =>   key <=0;
                    when "0000000000000001"   =>   key <=960;
                    when "0000000000000010"   =>   key <=856;
                    when "0000000000000100"   =>   key <=762;
                    when "0000000000001000"   =>   key <=720;
                    when "0000000000010000"   =>   key <=643;
                    when "0000000000100000"   =>   key <=572;
                    when "0000000001000000"   =>   key <=508;
                    when "0000000010000000"   =>   key <=480;
                    when "0000000100000000"   =>   key <=240;
                    when "0000001000000000"   =>   key <=214;
                    when "0000010000000000"   =>   key <=190;
                    when "0000100000000000"   =>   key <=180;
                    when "0001000000000000"   =>   key <=160;
                    when "0010000000000000"   =>   key <=143;
                    when "0100000000000000"   =>   key <=127;
                    when "1000000000000000"   =>   key <=430;
                    when others => key <=0;
                    end case;
        end process ;
        --decoder
        --------------------------------------------------------------------------
end architecture ; -- arch
--architecture
--------------------------------------------------------------------------------
```

beep

```
--------------------------------------------------------------------------------
--beep
```

```
--input clock control keep button
--output beep
--play music or button music
----------------------------------------------------------------------------


----------------------------------------------------------------------------
--head
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
--head
----------------------------------------------------------------------------


----------------------------------------------------------------------------
--entity
entity beep is
   port (
      --in
      clock : in std_logic;
      control :in std_logic;
      key: in integer range 0 to 2047;
      button:in std_logic;

      --out
      beep : out std_logic
   );
end beep;
--entity
----------------------------------------------------------------------------


----------------------------------------------------------------------------
--architecture
architecture behave of beep is
      signal st2: std_logic;
      signal key_tmp:integer range 0 to 2047;
begin
      p1:process(clock,key)
            variable count1: integer range 0 to 2047;
      begin
            if control='1' then
                  if button='1' then
                        key_tmp<=214;
                  else
                        key_tmp<=key;
                  end if ;
                  if clock'event and clock='1' then
                        if count1<key_tmp then
```

```
                        count1:= count1+1;
                        st2 <= '1';
                    else
                        count1:=0;
                        st2<='0';
                    end if;
                end if;
            else
                st2<='0';
            end if;
        end process p1;

        p2: process(st2)
            variable count2:std_logic:= '0';
        begin
            if control='1' then
                if st2'event and st2 = '1' then
                    count2 := not count2;
                    if count2 = '1'then
                        beep<='1';
                    else
                        beep<='0';
                    end if;
                end if;
            else
                beep<='0';
            end if;
        end process p2;
end behave;

--architecture
---------------------------------------------------------------------------
```

7. 系统整体顶层级联

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY work;
ENTITY final_1 IS
    PORT
    (
        reset_sw7_125 :   IN   STD_LOGIC;
        clock_18 :   IN   STD_LOGIC;
        btn7_124 :   IN   STD_LOGIC;
        btn5_122 :   IN   STD_LOGIC;
        btn4_121 :   IN   STD_LOGIC;
        btn3_91 :   IN   STD_LOGIC;
```

```
        btn2_89 :   IN   STD_LOGIC;
        btn1_20 :   IN   STD_LOGIC;
        btn0_61 :   IN   STD_LOGIC;
        reset_led15_137 :   OUT   STD_LOGIC;
        enable_led14_138 :   OUT   STD_LOGIC;
        test_led13_139 :   OUT   STD_LOGIC;
        lock1_led12_140 :   OUT   STD_LOGIC;
        lock2_led11_141 :   OUT   STD_LOGIC;
        lock4_led9_143 :   OUT   STD_LOGIC;
        lock3_led10_142 :   OUT   STD_LOGIC;
        lock5_led8_144 :   OUT   STD_LOGIC;
        lock6_led7_73 :   OUT   STD_LOGIC;
        beep :   OUT   STD_LOGIC;
        cat :   OUT   STD_LOGIC_VECTOR(7 DOWNTO 0);
        col :   OUT   STD_LOGIC_VECTOR(7 DOWNTO 0);
        content_led6_0 :   OUT   STD_LOGIC_VECTOR(6 DOWNTO 0);
        disp :   OUT   STD_LOGIC_VECTOR(6 DOWNTO 0);
        row :   OUT   STD_LOGIC_VECTOR(7 DOWNTO 0)
    );
END final_1;
ARCHITECTURE bdf_type OF final_1 IS

------------------------------matrix_control---------------------------------
COMPONENT matrix_control
    PORT(clock : IN STD_LOGIC;
         enable : IN STD_LOGIC;
         test : IN STD_LOGIC;
         lock : IN STD_LOGIC;
         lock_signal : OUT STD_LOGIC;
         content : OUT STD_LOGIC_VECTOR(6 DOWNTO 0)
    );
END COMPONENT;
-------------------------------------------------------------------------------

--------------------------------beep-------------------------------------------
COMPONENT beep
    PORT(clock : IN STD_LOGIC;
         control : IN STD_LOGIC;
         button : IN STD_LOGIC;
         key : IN STD_LOGIC_VECTOR(10 DOWNTO 0);
         beep : OUT STD_LOGIC
    );
END COMPONENT;
-------------------------------------------------------------------------------


------------------------------save_music-------------------------------------
COMPONENT save_music
```

```
    PORT(clock : IN STD_LOGIC;
         enable : IN STD_LOGIC;
         tone : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
    );
END COMPONENT;
-------------------------------------------------------------------------------

--------------------frequency_devider-----------------------------------------
COMPONENT frequency_devider
    PORT(clock : IN STD_LOGIC;
         reset : IN STD_LOGIC;
         clock0000002 : OUT STD_LOGIC;
         clock00005 : OUT STD_LOGIC;
         clock005 : OUT STD_LOGIC;
         clock008 : OUT STD_LOGIC;
         clock01 : OUT STD_LOGIC;
         clock02 : OUT STD_LOGIC;
         clock03 : OUT STD_LOGIC;
         clock025 : OUT STD_LOGIC
    );
END COMPONENT;
-------------------------------------------------------------------------------

--------------------------------play_music-------------------------------------
COMPONENT play_music
    PORT(tone : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
         key : OUT STD_LOGIC_VECTOR(10 DOWNTO 0)
    );
END COMPONENT;
-------------------------------------------------------------------------------

--------------------------------disp_display-----------------------------------
COMPONENT disp_display
    PORT(clock : IN STD_LOGIC;
         disp0 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
         disp1 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
         disp2 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
         disp3 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
         disp4 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
         disp5 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
         disp6 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
         disp7 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
         cat : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
         disp : OUT STD_LOGIC_VECTOR(6 DOWNTO 0)
    );
END COMPONENT;
-------------------------------------------------------------------------------
```

```
------------------------disp_control2---------------------------------------
COMPONENT disp_control2
    PORT(enable : IN STD_LOGIC;
         test : IN STD_LOGIC;
         disp : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
    );
END COMPONENT;
-----------------------------------------------------------------------------

---------------------------------blocker2------------------------------------
COMPONENT blocker2
    PORT(enable : IN STD_LOGIC;
         btn : IN STD_LOGIC;
         lock_signal : IN STD_LOGIC;
         lock : OUT STD_LOGIC
    );
END COMPONENT;
-----------------------------------------------------------------------------

-------------------------disp_control----------------------------------------
COMPONENT disp_control
    PORT(clock : IN STD_LOGIC;
         enable : IN STD_LOGIC;
         test : IN STD_LOGIC;
         lock : IN STD_LOGIC;
         disp_in : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
         disp : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
    );
END COMPONENT;
-----------------------------------------------------------------------------

------------------------sequency_generator---------------------------------
COMPONENT sequence_generator
    PORT(clock : IN STD_LOGIC;
         reset : IN STD_LOGIC;
         seq_output : OUT STD_LOGIC;
         disp : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
    );
END COMPONENT;
-----------------------------------------------------------------------------

----------------------------test_control-------------------------------------
COMPONENT test_control
    PORT(clock : IN STD_LOGIC;
         sw7 : IN STD_LOGIC;
         btn7 : IN STD_LOGIC;
         complete : IN STD_LOGIC;
         enable : OUT STD_LOGIC;
```

```
        test : OUT STD_LOGIC;
        sw : OUT STD_LOGIC
    );
END COMPONENT;
-------------------------------------------------------------------------------

----------------------------------blocker---------------------------------------
COMPONENT blocker
    PORT(enable : IN STD_LOGIC;
        btn : IN STD_LOGIC;
        lock : OUT STD_LOGIC
    );
END COMPONENT;
-------------------------------------------------------------------------------

----------------------------matrix_display-------------------------------------
COMPONENT matrix_display
    PORT(clock : IN STD_LOGIC;
        content : IN STD_LOGIC_VECTOR(6 DOWNTO 0);
        col : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
        row : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
    );
END COMPONENT;
-------------------------------------------------------------------------------

SIGNAL SYNTHESIZED_WIRE_0 :  STD_LOGIC;
SIGNAL SYNTHESIZED_WIRE_57 :  STD_LOGIC;
SIGNAL SYNTHESIZED_WIRE_58 :  STD_LOGIC;
SIGNAL SYNTHESIZED_WIRE_59 :  STD_LOGIC;
SIGNAL SYNTHESIZED_WIRE_60 :  STD_LOGIC;
SIGNAL SYNTHESIZED_WIRE_5 :  STD_LOGIC;
SIGNAL SYNTHESIZED_WIRE_6 :  STD_LOGIC;
SIGNAL SYNTHESIZED_WIRE_7 :  STD_LOGIC_VECTOR(10 DOWNTO 0);
SIGNAL SYNTHESIZED_WIRE_61 :  STD_LOGIC;
SIGNAL SYNTHESIZED_WIRE_9 :  STD_LOGIC_VECTOR(15 DOWNTO 0);
SIGNAL SYNTHESIZED_WIRE_62 :  STD_LOGIC;
SIGNAL SYNTHESIZED_WIRE_11 :  STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL SYNTHESIZED_WIRE_12 :  STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL SYNTHESIZED_WIRE_13 :  STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL SYNTHESIZED_WIRE_14 :  STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL SYNTHESIZED_WIRE_15 :  STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL SYNTHESIZED_WIRE_63 :  STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL SYNTHESIZED_WIRE_21 :  STD_LOGIC;
SIGNAL SYNTHESIZED_WIRE_22 :  STD_LOGIC;
SIGNAL SYNTHESIZED_WIRE_64 :  STD_LOGIC;
SIGNAL SYNTHESIZED_WIRE_65 :  STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL SYNTHESIZED_WIRE_28 :  STD_LOGIC;
SIGNAL SYNTHESIZED_WIRE_66 :  STD_LOGIC;
```

```
SIGNAL SYNTHESIZED_WIRE_33 :    STD_LOGIC;
SIGNAL SYNTHESIZED_WIRE_67 :    STD_LOGIC;
SIGNAL SYNTHESIZED_WIRE_38 :    STD_LOGIC;
SIGNAL SYNTHESIZED_WIRE_68 :    STD_LOGIC;
SIGNAL SYNTHESIZED_WIRE_69 :    STD_LOGIC;
SIGNAL SYNTHESIZED_WIRE_48 :    STD_LOGIC;
SIGNAL SYNTHESIZED_WIRE_56 :    STD_LOGIC_VECTOR(6 DOWNTO 0);

BEGIN
reset_led15_137 <= SYNTHESIZED_WIRE_21;
enable_led14_138 <= SYNTHESIZED_WIRE_58;
test_led13_139 <= SYNTHESIZED_WIRE_59;
lock1_led12_140 <= SYNTHESIZED_WIRE_60;
lock2_led11_141 <= SYNTHESIZED_WIRE_64;
lock4_led9_143 <= SYNTHESIZED_WIRE_67;
lock3_led10_142 <= SYNTHESIZED_WIRE_66;
lock5_led8_144 <= SYNTHESIZED_WIRE_68;
lock6_led7_73 <= SYNTHESIZED_WIRE_69;
content_led6_0 <= SYNTHESIZED_WIRE_56;
SYNTHESIZED_WIRE_0 <= '0';


SYNTHESIZED_WIRE_6 <= btn7_124 OR btn4_121 OR btn5_122 OR btn3_91 OR btn1_20 OR
btn2_89 OR btn0_61 OR SYNTHESIZED_WIRE_0;

b2v_inst1 : matrix_control
PORT MAP(clock => SYNTHESIZED_WIRE_57,
         enable => SYNTHESIZED_WIRE_58,
         test => SYNTHESIZED_WIRE_59,
         lock => SYNTHESIZED_WIRE_60,
         lock_signal => SYNTHESIZED_WIRE_22,
         content => SYNTHESIZED_WIRE_56);

b2v_inst11 : beep
PORT MAP(clock => SYNTHESIZED_WIRE_5,
         control => reset_sw7_125,
         button => SYNTHESIZED_WIRE_6,
         key => SYNTHESIZED_WIRE_7,
         beep => beep);

b2v_inst12 : save_music
PORT MAP(clock => SYNTHESIZED_WIRE_61,
         enable => reset_sw7_125,
         tone => SYNTHESIZED_WIRE_9);

b2v_inst13 : frequency_devider
PORT MAP(clock => clock_18,
         reset => reset_sw7_125,
```

```
        clock0000002 => SYNTHESIZED_WIRE_5,
        clock00005 => SYNTHESIZED_WIRE_62,
        clock005 => SYNTHESIZED_WIRE_57,
        clock008 => SYNTHESIZED_WIRE_38,
        clock01 => SYNTHESIZED_WIRE_33,
        clock02 => SYNTHESIZED_WIRE_28,
        clock03 => SYNTHESIZED_WIRE_61,
        clock025 => SYNTHESIZED_WIRE_48);

b2v_inst14 : play_music
PORT MAP(tone => SYNTHESIZED_WIRE_9,
        key => SYNTHESIZED_WIRE_7);

b2v_inst15 : disp_display
PORT MAP(clock => SYNTHESIZED_WIRE_62,
        disp0 => SYNTHESIZED_WIRE_11,
        disp1 => SYNTHESIZED_WIRE_12,
        disp2 => SYNTHESIZED_WIRE_13,
        disp3 => SYNTHESIZED_WIRE_14,
        disp4 => SYNTHESIZED_WIRE_15,
        disp5 => SYNTHESIZED_WIRE_63,
        disp6 => SYNTHESIZED_WIRE_63,
        disp7 => SYNTHESIZED_WIRE_63,
        cat => cat,
        disp => disp);

b2v_inst17 : disp_control2
PORT MAP(enable => SYNTHESIZED_WIRE_58,
        test => SYNTHESIZED_WIRE_59,
        disp => SYNTHESIZED_WIRE_63);

b2v_inst2 : blocker2
PORT MAP(enable => SYNTHESIZED_WIRE_21,
        btn => btn5_122,
        lock_signal => SYNTHESIZED_WIRE_22,
        lock => SYNTHESIZED_WIRE_60);

b2v_inst21 : disp_control
PORT MAP(clock => SYNTHESIZED_WIRE_61,
        enable => SYNTHESIZED_WIRE_58,
        test => SYNTHESIZED_WIRE_59,
        lock => SYNTHESIZED_WIRE_64,
        disp_in => SYNTHESIZED_WIRE_65,
        disp => SYNTHESIZED_WIRE_15);

b2v_inst22 : disp_control
PORT MAP(clock => SYNTHESIZED_WIRE_28,
        enable => SYNTHESIZED_WIRE_58,
```

```
            test => SYNTHESIZED_WIRE_59,
            lock => SYNTHESIZED_WIRE_66,
            disp_in => SYNTHESIZED_WIRE_65,
            disp => SYNTHESIZED_WIRE_14);

b2v_inst23 : disp_control
PORT MAP(clock => SYNTHESIZED_WIRE_33,
            enable => SYNTHESIZED_WIRE_58,
            test => SYNTHESIZED_WIRE_59,
            lock => SYNTHESIZED_WIRE_67,
            disp_in => SYNTHESIZED_WIRE_65,
            disp => SYNTHESIZED_WIRE_13);

b2v_inst24 : disp_control
PORT MAP(clock => SYNTHESIZED_WIRE_38,
            enable => SYNTHESIZED_WIRE_58,
            test => SYNTHESIZED_WIRE_59,
            lock => SYNTHESIZED_WIRE_68,
            disp_in => SYNTHESIZED_WIRE_65,
            disp => SYNTHESIZED_WIRE_12);

b2v_inst25 : disp_control
PORT MAP(clock => SYNTHESIZED_WIRE_57,
            enable => SYNTHESIZED_WIRE_58,
            test => SYNTHESIZED_WIRE_59,
            lock => SYNTHESIZED_WIRE_69,
            disp_in => SYNTHESIZED_WIRE_65,
            disp => SYNTHESIZED_WIRE_11);

b2v_inst26 : sequence_generator
PORT MAP(clock => clock_18,
            reset => reset_sw7_125,
            disp => SYNTHESIZED_WIRE_65);

b2v_inst3 : test_control
PORT MAP(clock => SYNTHESIZED_WIRE_48,
            sw7 => reset_sw7_125,
            btn7 => btn7_124,
            complete => SYNTHESIZED_WIRE_69,
            enable => SYNTHESIZED_WIRE_58,
            test => SYNTHESIZED_WIRE_59,
            sw => SYNTHESIZED_WIRE_21);

b2v_inst4 : blocker
PORT MAP(enable => SYNTHESIZED_WIRE_60,
            btn => btn4_121,
            lock => SYNTHESIZED_WIRE_64);
```

```
b2v_inst5 : blocker
PORT MAP(enable => SYNTHESIZED_WIRE_64,
         btn => btn3_91,
         lock => SYNTHESIZED_WIRE_66);

b2v_inst6 : blocker
PORT MAP(enable => SYNTHESIZED_WIRE_66,
         btn => btn2_89,
         lock => SYNTHESIZED_WIRE_67);

b2v_inst7 : blocker
PORT MAP(enable => SYNTHESIZED_WIRE_67,
         btn => btn1_20,
         lock => SYNTHESIZED_WIRE_68);

b2v_inst8 : blocker
PORT MAP(enable => SYNTHESIZED_WIRE_68,
         btn => btn0_61,
         lock => SYNTHESIZED_WIRE_69);

b2v_inst9 : matrix_display
PORT MAP(clock => SYNTHESIZED_WIRE_62,
         content => SYNTHESIZED_WIRE_56,
         col => col,
         row => row);

END bdf_type;
```

五、功能说明及资源利用情况

1.功能说明

按键说明

SW7：实现开机关机功能
BTN7：按下即可开始选号，完成选号后再次按下即可再次进入选号状态重新选号
BTN5：字母选择按键，按下即可在点阵中选择字母
BTN4：数字选择按键，按下即可在数码管中选择 DISP4 对应的数字
BTN3：数字选择按键，按下即可在数码管中选择 DISP3 对应的数字
BTN2：数字选择按键，按下即可在数码管中选择 DISP2 对应的数字
BTN1：数字选择按键，按下即可在数码管中选择 DISP1 对应的数字
BTN0：数字选择按键，按下即可在数码管中选择 DISP0 对应的数字

显示说明

（1） 按下 SW7 开机，开始播放音乐，点阵和全部数码管全亮全暗闪烁三次，实现开机自检功能；
（2） 进入待机状态，点阵和数码管全部熄灭
（3） 按下 BTN7 进入选号状态，点阵滚动显示 A-F 六个大写字母，滚动速度为 0.01S/行，五位数码管以不同的速度随机显示 0-9 十个数字；
（4） 按下 BTN5 选择字母，只有当字母完成显示时才可被选中；
（5） 按下 BTN4 选择 DISP4 对应的数字
（6） 按下 BTN3 选择 DISP3 对应的数字
（7） 按下 BTN2 选择 DISP2 对应的数字
（8） 按下 BTN1 选择 DISP1 对应的数字
（9） 按下 BTN0 选择 DISP0 对应的数字
（10） 全部完成后闪烁三下以示提醒
（11） 所选号码稳定显示；
（12） 若按下 BTN7 则再次进入选号状态。

2. 资源利用情况

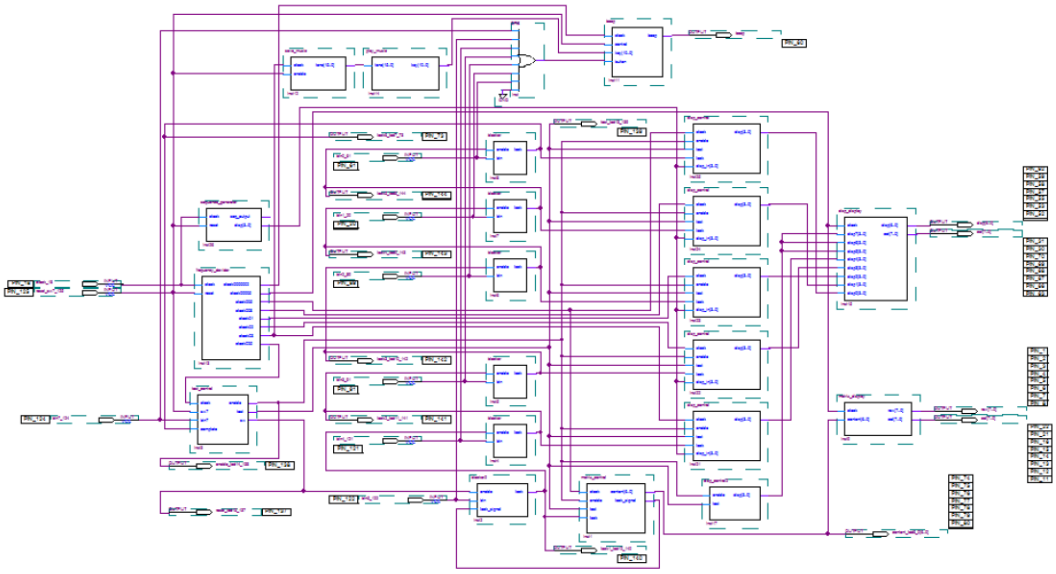
Flow Status	Successful - Tue Nov 07 19:23:43 2017
Quartus II Version	9.0 Build 235 06/17/2009 SP 2 SJ Web Edition
Revision Name	final_1
Top-level Entity Name	final_1
Family	MAX II
Device	EPM1270T144C5
Timing Models	Final
Met timing requirements	No
Total logic elements	481 / 1,270 (38 %)
Total pins	57 / 116 (49 %)
Total virtual pins	0
UFM blocks	0 / 1 (0 %)

由编译结果可得：本程序一共使用逻辑单元 481 个 ， 占用率为 38%。使用逻辑管脚 57 个，占用率是 49%。

管脚分配图

	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Group
1	beep	Output	PIN_60	4		3.3-V LVTTL (default)		
2	btn0_61	Input	PIN_61	4		3.3-V LVTTL (default)		
3	btn1_20	Input	PIN_20	1		3.3-V LVTTL (default)		
4	btn2_89	Input	PIN_89	3		3.3-V LVTTL (default)		
5	btn3_91	Input	PIN_91	3		3.3-V LVTTL (default)		
6	btn4_121	Input	PIN_121	2		3.3-V LVTTL (default)		
7	btn5_122	Input	PIN_122	2		3.3-V LVTTL (default)		
8	btn7_124	Input	PIN_124	2		3.3-V LVTTL (default)		
9	cat[7]	Output	PIN_31	1		3.3-V LVTTL (default)		cat[7..0]
10	cat[6]	Output	PIN_30	1		3.3-V LVTTL (default)		cat[7..0]
11	cat[5]	Output	PIN_70	4		3.3-V LVTTL (default)		cat[7..0]
12	cat[4]	Output	PIN_69	4		3.3-V LVTTL (default)		cat[7..0]
13	cat[3]	Output	PIN_68	4		3.3-V LVTTL (default)		cat[7..0]
14	cat[2]	Output	PIN_67	4		3.3-V LVTTL (default)		cat[7..0]
15	cat[1]	Output	PIN_66	4		3.3-V LVTTL (default)		cat[7..0]
16	cat[0]	Output	PIN_63	4		3.3-V LVTTL (default)		cat[7..0]
17	clock_18	Input	PIN_18	1		3.3-V LVTTL (default)		
18	col[7]	Output	PIN_22	1		3.3-V LVTTL (default)		col[7..0]
19	col[6]	Output	PIN_21	1		3.3-V LVTTL (default)		col[7..0]
20	col[5]	Output	PIN_16	1		3.3-V LVTTL (default)		col[7..0]
21	col[4]	Output	PIN_15	1		3.3-V LVTTL (default)		col[7..0]
22	col[3]	Output	PIN_14	1		3.3-V LVTTL (default)		col[7..0]
23	col[2]	Output	PIN_13	1		3.3-V LVTTL (default)		col[7..0]
24	col[1]	Output	PIN_12	1		3.3-V LVTTL (default)		col[7..0]
25	col[0]	Output	PIN_11	1		3.3-V LVTTL (default)		col[7..0]
26	content_led6_0[6]	Output	PIN_74	3		3.3-V LVTTL (default)		content_led6_0[6..0]
27	content_led6_0[5]	Output	PIN_75	3		3.3-V LVTTL (default)		content_led6_0[6..0]
28	content_led6_0[4]	Output	PIN_76	3		3.3-V LVTTL (default)		content_led6_0[6..0]
29	content_led6_0[3]	Output	PIN_77	3		3.3-V LVTTL (default)		content_led6_0[6..0]
30	content_led6_0[2]	Output	PIN_78	3		3.3-V LVTTL (default)		content_led6_0[6..0]
31	content_led6_0[1]	Output	PIN_79	3		3.3-V LVTTL (default)		content_led6_0[6..0]
32	content_led6_0[0]	Output	PIN_80	3		3.3-V LVTTL (default)		content_led6_0[6..0]
33	disp[6]	Output	PIN_62	4		3.3-V LVTTL (default)		disp[6..0]
34	disp[5]	Output	PIN_59	4		3.3-V LVTTL (default)		disp[6..0]
35	disp[4]	Output	PIN_58	4		3.3-V LVTTL (default)		disp[6..0]
36	disp[3]	Output	PIN_57	4		3.3-V LVTTL (default)		disp[6..0]
37	disp[2]	Output	PIN_55	4		3.3-V LVTTL (default)		disp[6..0]
38	disp[1]	Output	PIN_53	4		3.3-V LVTTL (default)		disp[6..0]
39	disp[0]	Output	PIN_52	4		3.3-V LVTTL (default)		disp[6..0]
40	enable_led14_138	Output	PIN_138	2		3.3-V LVTTL (default)		
41	lock1_led12_140	Output	PIN_140	2		3.3-V LVTTL (default)		
42	lock2_led11_141	Output	PIN_141	2		3.3-V LVTTL (default)		
43	lock3_led10_142	Output	PIN_142	2		3.3-V LVTTL (default)		
44	lock4_led9_143	Output	PIN_143	2		3.3-V LVTTL (default)		
45	lock5_led8_144	Output	PIN_144	2		3.3-V LVTTL (default)		
46	lock6_led7_73	Output	PIN_73	3		3.3-V LVTTL (default)		
47	reset_led15_137	Output	PIN_137	2		3.3-V LVTTL (default)		
48	reset_sw7_125	Input	PIN_125	2		3.3-V LVTTL (default)		
49	row[7]	Output	PIN_1	1		3.3-V LVTTL (default)		row[7..0]
50	row[6]	Output	PIN_2	1		3.3-V LVTTL (default)		row[7..0]
51	row[5]	Output	PIN_3	1		3.3-V LVTTL (default)		row[7..0]
52	row[4]	Output	PIN_4	1		3.3-V LVTTL (default)		row[7..0]
53	row[3]	Output	PIN_5	1		3.3-V LVTTL (default)		row[7..0]
54	row[2]	Output	PIN_6	1		3.3-V LVTTL (default)		row[7..0]
55	row[1]	Output	PIN_7	1		3.3-V LVTTL (default)		row[7..0]
56	row[0]	Output	PIN_8	1		3.3-V LVTTL (default)		row[7..0]
57	test_led13_139	Output	PIN_139	2		3.3-V LVTTL (default)		

整体电路原理图



六、故障及问题分析

问题一：没有找到合适的随机数种子，因而在数码管随机数滚动上遇到了困难。
解决方法：采用待机时间作为随机数种子，经过 m 序列发生器，循环计数器来实现随机数的生成。在两边时钟差距较大的情况下生成伪随机数。

问题二：在加上字母滚动的功能前，字母的选择显示是相互分离的，因而很难达到滚动的效果
解决方法：在思考后采用了矩阵下标索引的方式来查找显示区域，成功实现了字母自上而下的滚动。

问题三：在顺序控制部分遇到了问题，之前准备通过条件判断语句设置对应的状态，在前面状态完成的条件下才能开始下一个，但方案可行性低，不易实现。
解决方法：通过设置 blocker 的顺序控制模块，以其输出 lock 信号作为下一状态的 enable 信号，当上一状态未结束时通过 blocker 模块将下一模块锁住，使其无法工作，从而达到在上一部分选号未完成前，无法进行下一数字的选择的功能。

问题四：蜂鸣器发声问题。一开始测试蜂鸣器模块的时候，我以为只要有高电平输出，蜂鸣器就会鸣响，但下载到板子上后发现，结束工作时蜂鸣器只是发出很小很小的声音，不靠近仔细听几乎听不到。
解决方法：查阅实验板的说明书，了解到蜂鸣器也是需要方波来驱动的。于是我重新设计蜂鸣器模块，增加一个从系统时钟分频的模块以驱动蜂鸣器发出人耳可识别的响声。

七、总结和结论

本实验实现了一个较为完整的选号机程序，其中含有开机自检、号码选择、点阵滚动等全部功能，并实现了良好的用户交互，完成了实验要求。

通过本次实验，我对数字电路与逻辑设计和 VHDL 的开发编写过程有了更深刻、直观的认识。从最初的学习，到不断纠错，再到拓展、创新、进阶，我领会到了硬件语言 VHDL 的思路与逻辑。在不断查找资料、添加功能的过程中，我逐渐积累了许多相关的知识，并对数字电路、硬件语言 VHDL 方面产生了浓厚的兴趣。在我小的时候，我也玩过类似选号机的游戏，所以我在选题时毫不犹豫的选择了这个题目。这次选号机程序的编写，让我对其中的原理有了非常深刻的认识，也让我深刻地感受到了硬件语言的强大与开发人员的心血。

前两周，我主要进行了代码的编译与仿真；第三周时，我几乎一没课就泡在了主楼实验室里，主楼 6 楼成了那段时间我最喜欢的地方。在为期三周的实验过程中，我遇到了各种各样的问题，无论是在代码的编译过程中，仿真的模拟过程中，还是实验下载到电路板的过程中，都没有想象的顺利。但我始终没有放弃，一点点地进行分析与修正，终于在最后的实验中得到了想要的结果。

在具体的 VHDL 代码编写问题上，与其他软件编程语言不同，VHDL 能够很清晰地分辨出并行语法和串行语法对于输出信号(因变量)带来的影响，这一点对于时序电路的实际控制影响巨大。而且对 VHDL 语言来讲，有的时候逻辑上可行，但实际电路却不可实现，例如在编 VHDL 代码过程中在一个进程里同时检测两个时钟的上升沿，编译报错，通过查阅资料才发现实际的硬件电路是不可能做到同时检测两个时钟的上升沿，所以一定要清楚地明白硬件编程与软件编程的区别，掌握一些基本的电路知识，才能更好地用 VHDL 语言进行系统设计。

通过这次数电实验，我才终于真正深入理解了上学期数字电路与逻辑设计课程所学的知识，也对 VHDL 的知识掌握有了更好的理解，对于 QuartuesⅡ软件也有了更熟练的掌握。更重要的是，在一次次地尝试过程中，我对于这种软件与硬件相结合的工程，有了更多的喜爱与热情，也明白了只要用心去做，只要用心去钻研，终会发现更广阔的一片天空。