



EDA设计 (II)

多功能数字时钟

作者 : 许晓明 学号 : 9161040G0734

学院 : 电子工程与光电技术学院

专业 : 电子信息工程

班级 : 9161042103

题目 : EDA设计(II):

多功能数字钟设计

指导者 : 谭雪琴

2018 年 12月

摘要

数字钟是用数字电路技术实现计时的钟表。与机械钟相比，其准确性与直观性更高，使用寿命也更长。本文基于 Quartus II 软件，通过实物电路与 VHDL 语言编程结合的方式实现数字钟，其具有 24 小时计时，时钟星期、时、分、秒快速校对，时钟保持、清零，整点报时，铃声，万年历等功能。

本文采用了自顶向下的模块式分析设计方法，首先分析了多功能数字钟的设计要求及所需实现的功能，将整个电路分为分频模块，校时模块，清零保持模块，计时模块，音乐模块，显示模块，万年历模块等。

在具体设计时，采用的是自底向上的设计方法。即首先设计各种基础模块，然后设计功能模块，最后再进行综合设计。在设计同时，对每个模块所需要的功能进行仿真验证或直接下载到实验平台进行验证。在验证了功能的正确性后，连接各个模块组成一个总的电路。最后对总的电路分配好引脚，下载到实验板上。

实验的最终结果，在满足基本功能的情况下进行了创新，达到了不错的效果。

本报告首先分析了数字时钟的工作原理，并解释说明各子模块的设计原理及调试过程。其次对实验最终结果进行总结。在报告的最后总结了实验过程中出现的问题困难以及相应的解决方法。

关键词：QuartusII 多功能数字时钟 万年历 模块设计 仿真

Abstract

The digital clock is a timepiece that uses digital circuit technology to achieve timing. Compared with mechanical clocks, it is more accurate and intuitive, and has a longer service life. Based on Quartus II software, this paper realizes the digital clock through the combination of physical circuit and VHDL language programming. It has 24-hour timing, fast calibration of clock week, hour, minute and second, clock hold, clear, hourly chime, ringtone, perpetual calendar, etc. Features.

This paper adopts the top-down modular analysis design method. Firstly, it analyzes the design requirements of the multi-function digital clock and the functions that need to be realized. The whole circuit is divided into frequency division module, calibration module, zero-keeping module, timing. Module, music module, display module, perpetual calendar module, etc.

In the specific design, the bottom-up design method is adopted. That is, first design various basic modules, then design functional modules, and finally carry out comprehensive design. At the same time of design, the functions required for each module are simulated or directly downloaded to the experimental platform for verification. After verifying the correctness of the function, connect each module to form a total circuit. Finally, the pins are allocated to the total circuit and downloaded to the experiment board.

The final result of the experiment was innovated with basic functions and achieved good results.

This report first analyzes the working principle of the digital clock and explains the design principle and debugging process of each sub-module. Secondly, the final results of the experiment are summarized. At the end of the report, the problems encountered during the experiment and the corresponding solutions were summarized.

key words: QuartusII,Multifunction digital clock, Perpetual calendar, Module design, simulation

目录

1	设计要求说明	1
1.1	设计基本要求	1
1.2	设计提高部分要求	1
2	方案论证（整体电路的工作原理）	2
3	各子模块设计原理	2
3.1	分频模块	2
3.1.1	2分频	2
3.1.2	4分频	3
3.1.3	10分频	3
3.1.4	12分频	5
3.1.5	1000分频	5
3.1.6	分频模块	7
3.2	总计时模块	7
3.2.1	由模7、模24、模60模块组成的基本计时模块	7
3.2.2	校时模块	13
3.2.3	清零、保持模块	13
3.2.4	消颤模块	13
3.2.5	报时模块	16
3.2.6	总计时模块	19
3.3	音乐模块	19
3.3.1	音符的产生	19
3.3.2	音乐模块	21
3.4	显示模块	25
3.4.1	模8计数器	26
3.4.2	八选一	28
3.4.3	3线-8线译码器	29
3.4.4	七段译码器	30

3.4.5 显示模块	31
3.5 万年历模块	33
3.5.1 日计数	33
3.5.2 月计数	35
3.5.3 年计数	38
3.5.4 日进制判断信号judge	40
3.5.5 万年历模块	42
4 多功能数字时钟总电路	46
5 调试、仿真、编程下载	46
6 结论	48
7 实验感想	48
7.1 实验过程中遇到的问题及解决问题的方法	48
7.2 实验的收获与感受	49
7.3 期望及要求	50
参考文献	50

1 设计要求说明

利用QuartusII软件设计一个数字计时器，可以完成00:00:00到 23:59:59的计时功能，并在控制电路的作用下具有保持、清零、快速校时、快速校分、整点报时等功能。并下载到SmartSOPC实验系统中。

1.1 设计基本要求

时钟的基本设计具体要求如下：

- 1、能进行正常的时、分、秒计时功能；
- 2、分别由六个数码管显示时分秒的计时；
- 3、 K_1 是系统的使能开关 ($K_1 = 0$ 正常工作, $K_1 = 1$ 时钟保持不变)；
- 4、 K_2 是系统的清零开关 ($K_2 = 0$ 正常工作, $K_2 = 1$ 时钟的分、秒全清零)；
- 5、 K_3 是系统的校分开关 ($K_3 = 0$ 正常工作, $K_3 = 1$ 时可以快速校分)；
- 6、 K_4 是系统的校时开关 ($K_4 = 0$ 正常工作, $K_4 = 1$ 时可以快速校时)；
- 7、使时钟具有整点报时功能（当时钟计到59' 53" 时开始报时，在59' 53" , 59' 55" ,59' 57" 时报时频率为512Hz,59' 59" 时报时频率为1KHz）。

1.2 设计提高部分要求

完成基本设计要求后，可选做提高性要求。实验中我做的功能有：

- 1、星期计时功能；
- 2、铃声功能；
- 3、万年历功能。

2 方案论证（整体电路的工作原理）

按功能，时钟可以分为以下几个模块：分频模块、计时模块、清零保持模块、校时模块、音乐模块、显示模块、万年历模块等。其中，需要用到开关的部分都需要加消颤功能。根据各模块功能的要求，各大模块又可以分成几个小的基本模块，模块间联系大致如图1。

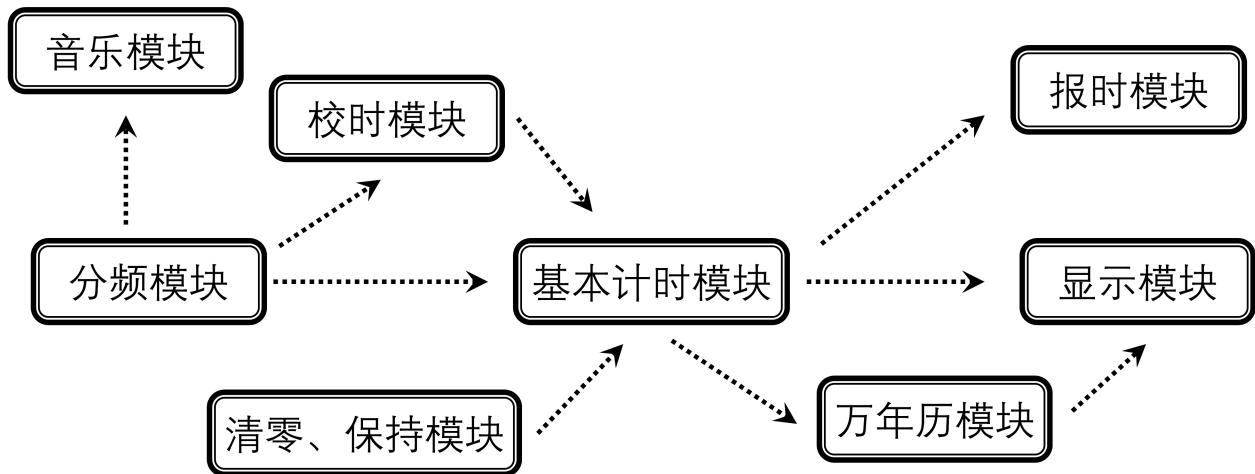


图1 各模块间联系

首先由分频模块产生脉冲信号，输入到基本计时模块中，计时模块的时、分、秒和星期等各位信号均输出到显示模块中，而计时电路的分位和秒位信号也输出到报时模块中，而清零、保持模块，校时模块都是在此基础上对基本计时模块的控制。万年历模块则是在基本模块上的附加模块。

3 各子模块设计原理

3.1 分频模块

分频模块实现的功能是将实验箱提供的48MHZ的高频信号变成所需的1Hz、2Hz、500Hz和1kHz的信号，其中，1kHz 用于报时最后一声，500Hz 用于报时的前几声，2Hz 用于校时，1Hz 用于计时。

3.1.1 2分频

借助D触发器实现二分频，原理图见图2，仿真波形见图3，封装图见图4。

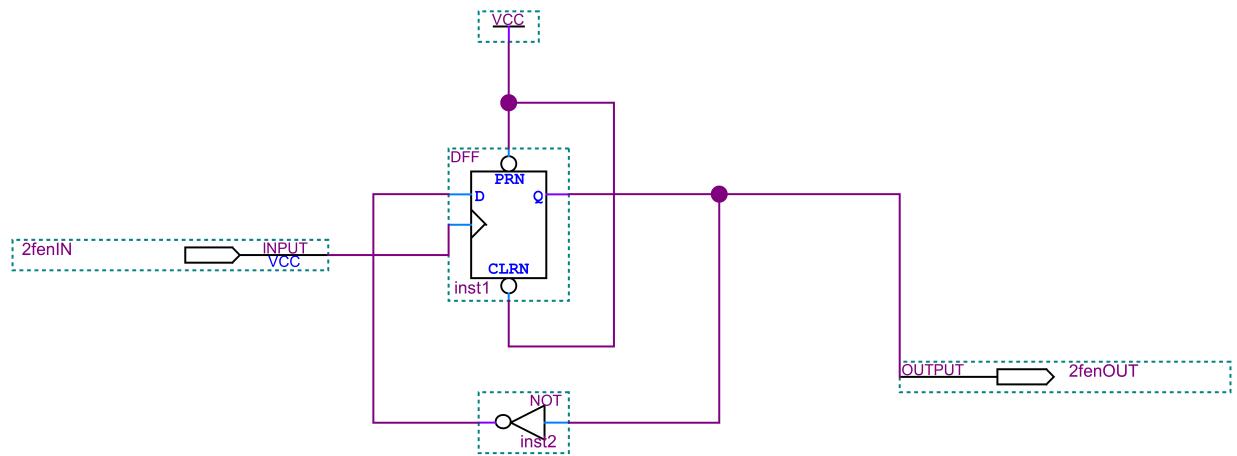


图 2

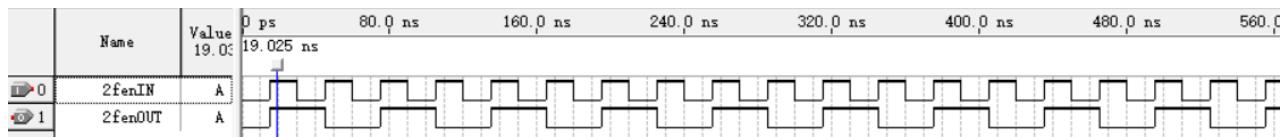


图 3

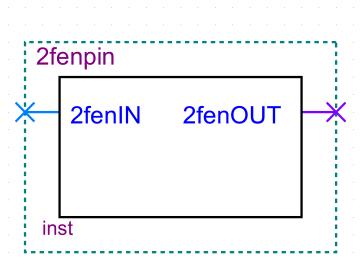


图 4

3.1.2 4分频

将2个二分频模块级联实现4分频，原理图见图5，仿真波形见图6，封装图见图7。

3.1.3 10分频

借助74161芯片和2分频电路实现10分频，当74160计数到4时，同步置数到0，

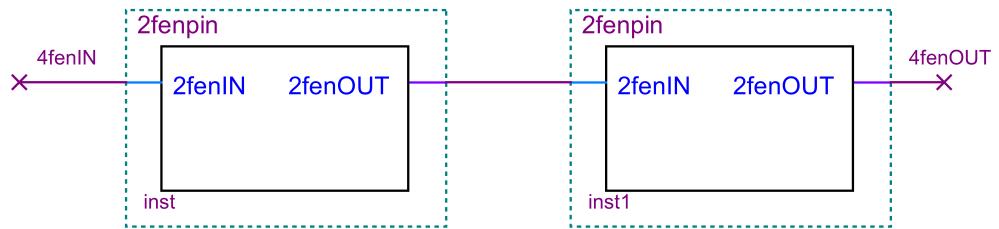


图 5

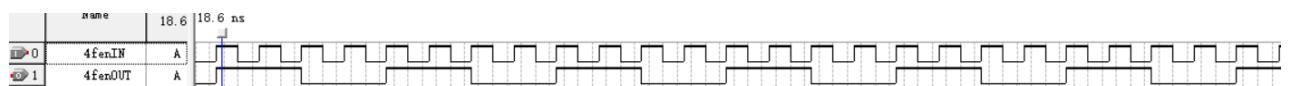


图 6

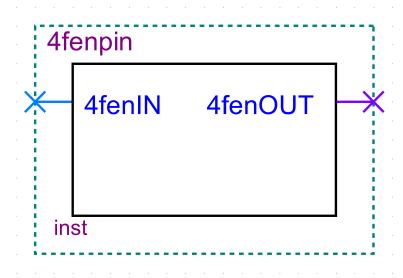


图 7

实现循环，同时输出1次高电平给2分频电路，实现占空比为 $\frac{1}{2}$ 的10分频。原理图见图8，仿真波形见图9，封装图见图10。

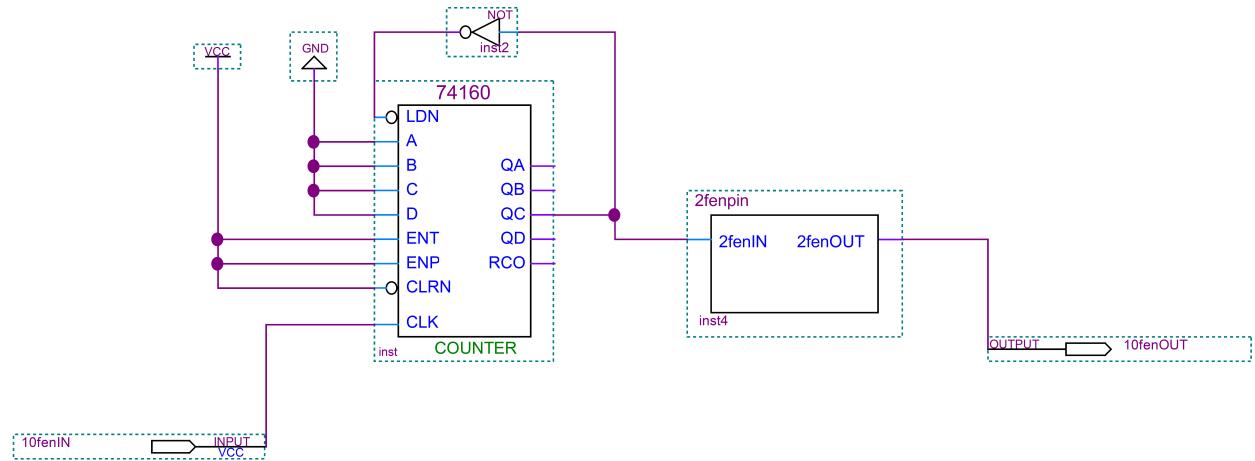


图 8

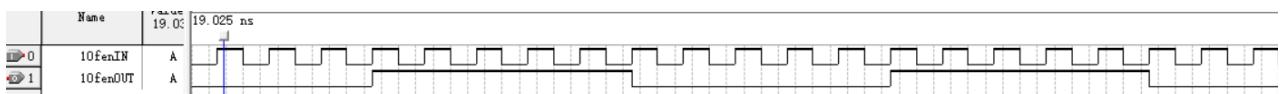


图 9



图 10

3.1.4 12分频

借助74161芯片实现12分频，当74161计数到11时，同步置数到0，实现循环，同时输出1次高电平，实现分频。原理图见图11，仿真波形见图12，封装图见图13。

3.1.5 1000分频

将3个10分频级联，得到1000分频。原理图见图14，封装图见图15。

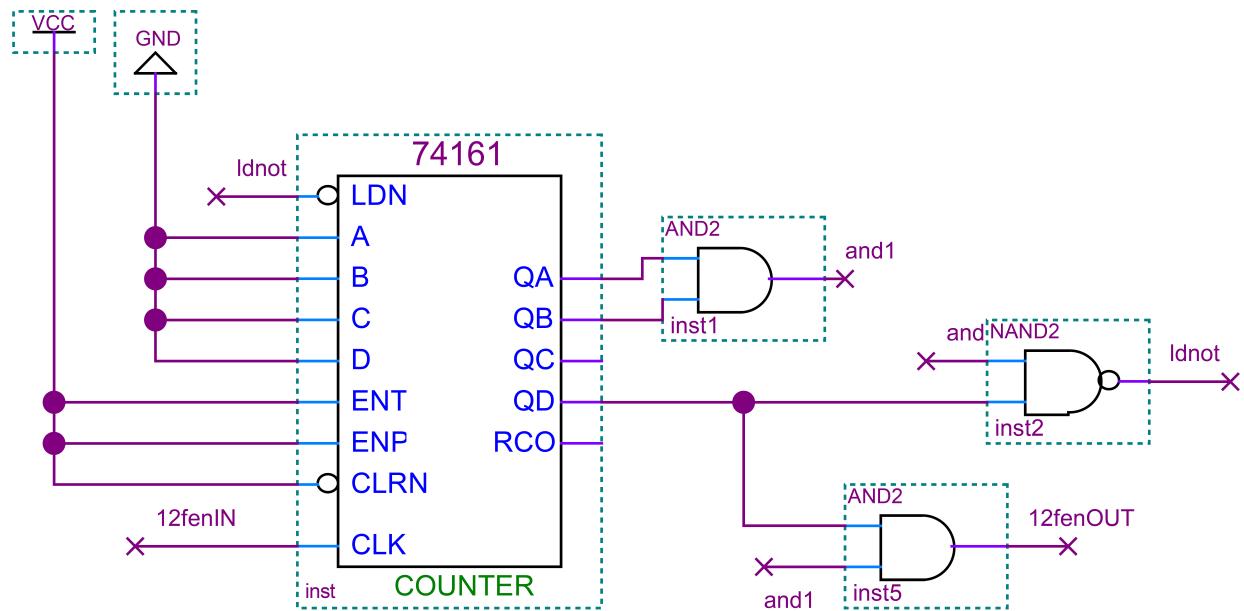


图 11

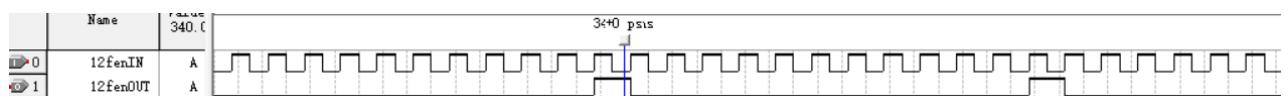


图 12

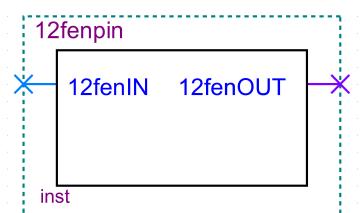


图 13

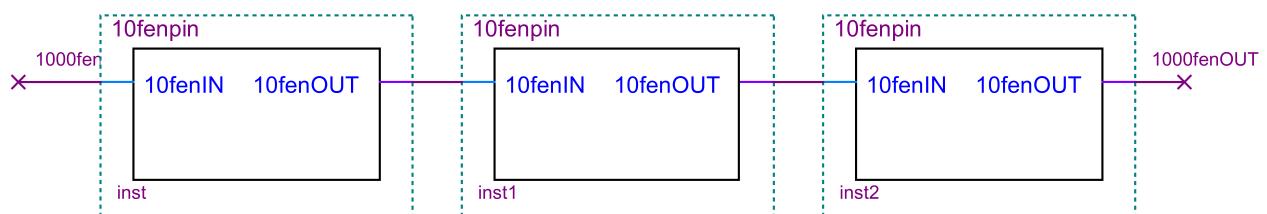


图 14

3.1.6 分频模块

最终，通过级联以上的各种分频器，实现分频模块。原理图见图16，仿真波形大致情况见图17，封装图见图18。

3.2 总计时模块

3.2.1 由模7、模24、模60模块组成的基本计时模块

利用VHDL语言的IF语句，对计数的条件进行判断。例如模7，则在计数到0111（7）时跳转到0001，即可实现数字1到7的循环计数。mo24，mo60的代码也类似，只是判断的条件不同。

VHDL代码实现模7 模7的代码如下，仿真情况见图19，封装图见图20。

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 ENTITY mo7 IS
5 PORT
6 ( en    :IN  std_logic;
7 clear:IN  std_logic;—when 0, output is always 0
8 clk   :IN  std_logic;
9 ql    :buffer std_logic_vector(3 downto 0)
10 );
11 END mo7;
12 ARCHITECTURE beh OF mo7 IS
13 BEGIN
14 PROCESS(clk, clear)
15 BEGIN
16 IF (clear = '0') THEN
17 ql<="0000";
18 ELSIF (clk'EVENT AND clk = '1') THEN
19 if (en = '1') then
```

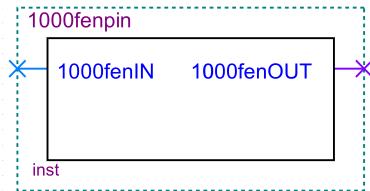


图 15

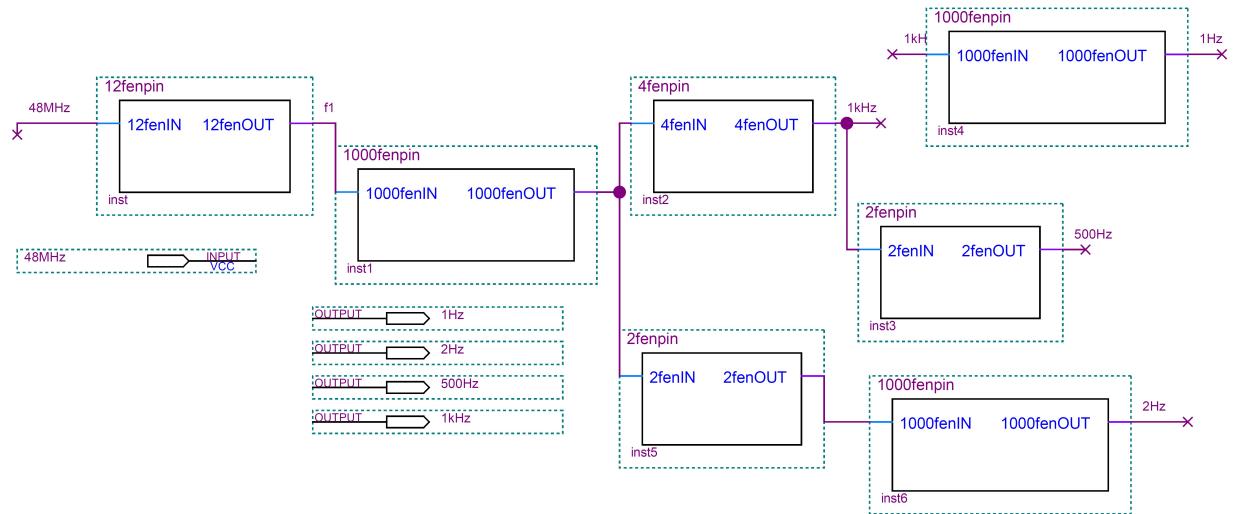


图 16

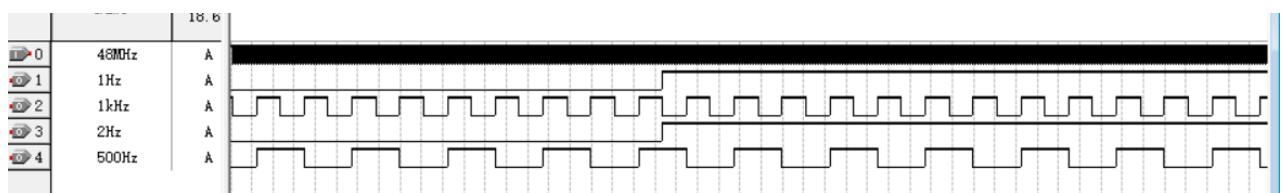


图 17

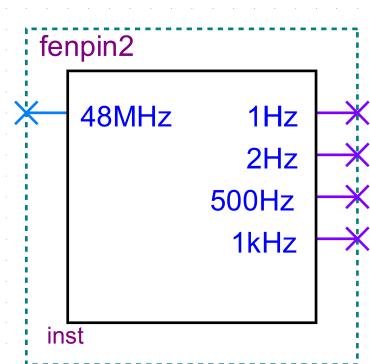


图 18

```

20 if( ql=7) then
21 ql<="0001";
22 else
23 ql<=ql+1;
24 end if;
25 end if;
26 END IF;
27 END PROCESS;
28 END beh ;
29 }

```

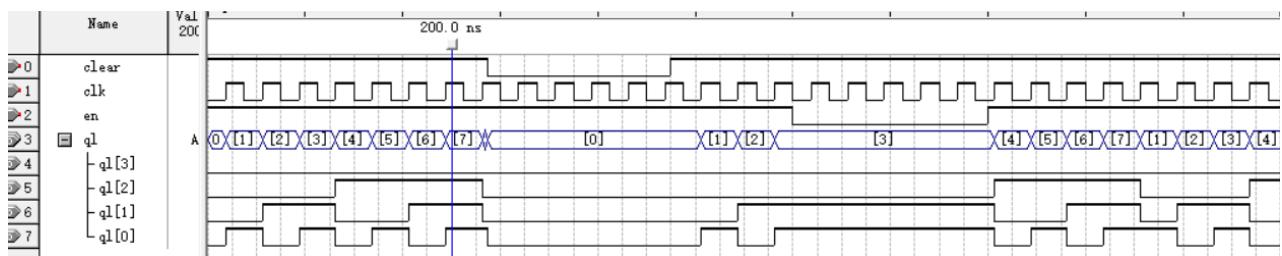


图 19

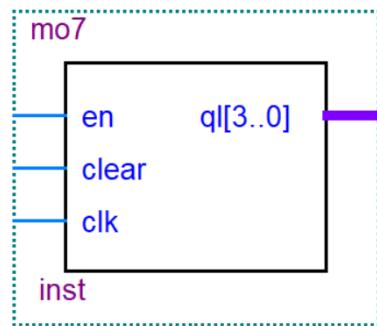


图 20

VHDL代码实现模24 模24的代码如下，仿真情况见图21，封装图见图22。

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 ENTITY mo24 IS

```

```

5  PORT
6  ( en      :IN  std_logic ;
7  clear:IN  std_logic ;—when 0 , output is always 0
8  clk     :IN  std_logic ;
9  cout    :out  std_logic ;—jinwei output
10 qh     :buffer std_logic_vector(3 downto 0);—shiwei output
11 ql     :buffer std_logic_vector(3 downto 0)
12 );—gewei output
13 END mo24;
14 ARCHITECTURE beh OF mo24 IS
15 BEGIN
16 cout<='1'when(qh="0010"and ql="0011"and en='1')else '0';
17
18 PROCESS(clk , clear )
19 BEGIN
20 IF (clear ='0')THEN
21 qh<="0000";
22 ql<="0000";
23 ELSIF( clk 'EVENT AND clk ='1')THEN
24 if(en ='1')then
25 if(( ql=3 and qh=2) or ql=9) then
26 ql<="0000";
27 if(qh=2)then
28 qh<="0000";
29 else
30 qh<=qh+1;
31 end if;
32 else
33 ql<=ql+1;
34 end if;
35 end if;

```

```

36 END IF ;
37 END PROCESS;
38 END beh ;

```

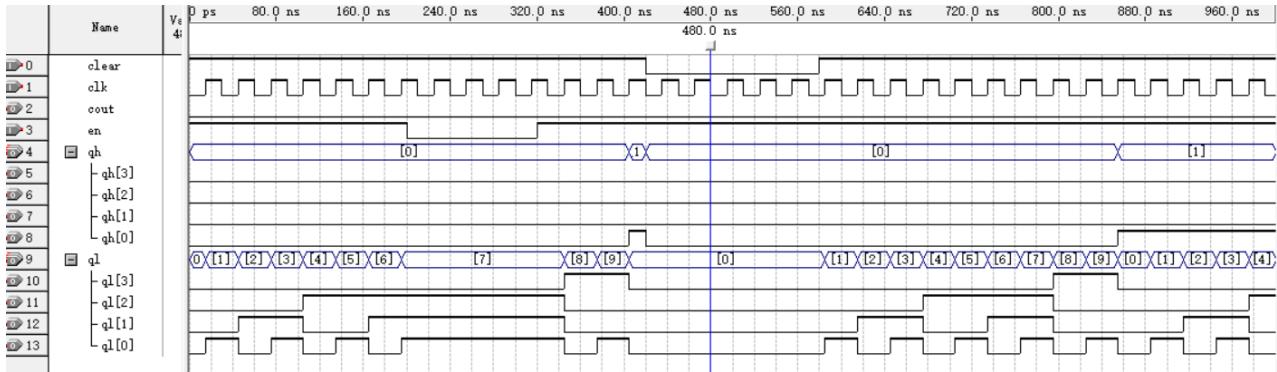


图 21

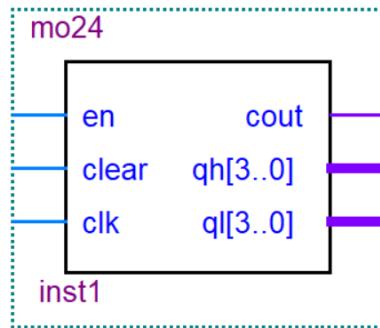


图 22

VHDL代码实现模60 模60的代码如下，仿真实情况见图23，封装图见图24。

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 ENTITY mo60 IS
5 PORT
6 ( en :IN std_logic;
7 clear:IN std_logic;—when 0, output is always 0
8 clk :IN std_logic ;

```

```

9 cout :out std_logic;—jinwei output
10 qh :buffer std_logic_vector(3 downto 0);—shiwei output
11 ql :buffer std_logic_vector(3 downto 0)
12 );—gewei output
13 END mo60;
14 ARCHITECTURE beh OF mo60 IS
15 BEGIN
16 cout <='1'when(qh="0101"and ql="1001"and en='1')else '0';
17
18 PROCESS(clk, clear)
19 BEGIN
20 IF (clear='0')THEN
21 qh<="0000";
22 ql<="0000";
23 ELSIF (clk'EVENT AND clk='1')THEN
24 if (en='1')then
25 if (ql=9) then
26 ql<="0000";
27 if (qh=5 and ql=9) then
28 qh<="0000";
29 else
30 qh<=qh+1;
31 end if;
32 else
33 ql<=ql+1;
34 end if;
35 end if;
36 END IF;
37 END PROCESS;
38 END beh;

```

3.2.2 校时模块

校时由二输入与门、非门和或门这三种逻辑门构成。当 $K_3 = 0$ （分位校时开关）时，时钟端clk输入1Hz的信号，时钟正常计数，当 $K_3 = 1$ 时，时钟端clk输入2Hz的信号，进行快速校分。原理图见图25，仿真情况见图26，封装图见图27。

对星期位、时位的校时电路原理类似，这里不赘述。

而事实上，为了实现同步时序基本计时，在基本计时模块，是将低位的进位信号作为高位的使能信号来实现时钟信号同步的，仅仅是改变时钟端信号并不能实现校时，还需要使能端的配合。由于对使能端的操作也涉及到保持功能，因此我将使能端的处理放在清零、保持模块完成。

3.2.3 清零、保持模块

清零 清零功能是通过控制各位计数器清零端的电平高低来实现的。因此，只需使清零开关按下时各计数器的清零端均接入低电平、清零开关断开时各清零端均接入高电平即可。

保持 保持功能是通过控制计数器的使能端实现的。当 $K_1 = 0$ ，通过一个非门接到使能端使计数器正常计数；当 $K_1 = 1$ 时，使能端变成0，计数器停止计数，实现计时保持功能。

而由于所有其他位的计数都依靠或直接或间接秒位的进位信号，因此使秒位使能端失效即可实现保持功能。而为了秒位的进位信号不干扰校时，任意一个校时开关开启，也同步让秒位使能失效。

在校时模块中，需要对使能端操作，因此在保持模块里，对应的校时开关开启，则对应的使能信号便有效。为避免干扰，低位的进位信号必须在其他校时开关均关闭时才能影响使能端，于是，这一部分的原理图见图28，封装图见图29。

3.2.4 消颤模块

由于开关的接通和关闭有可能产生毛刺，出现竞争冒险现象，所以要对所有的开关进行消颤处理。具体是通过一个D触发器对开关信号进行延时，从而

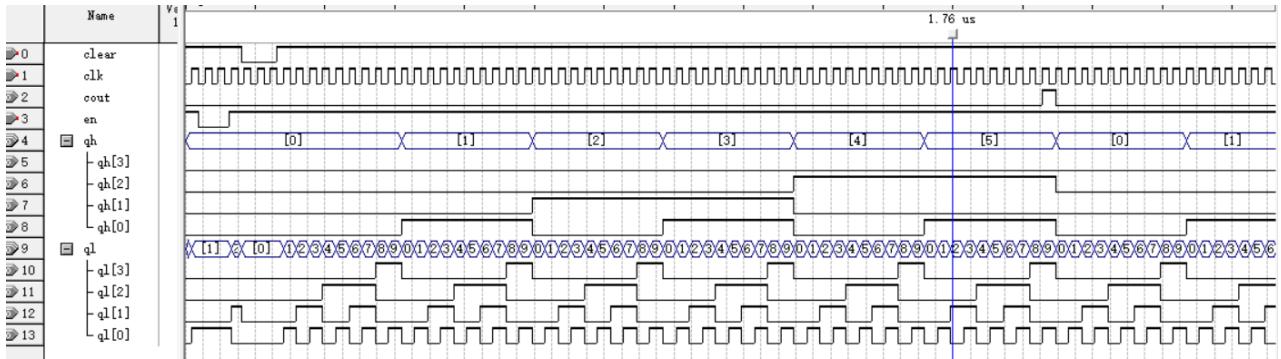


图 23

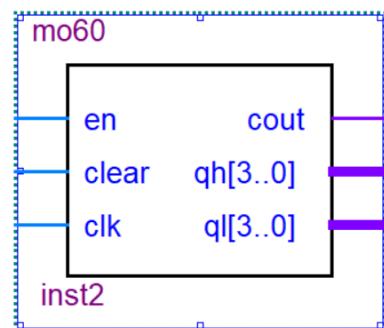


图 24

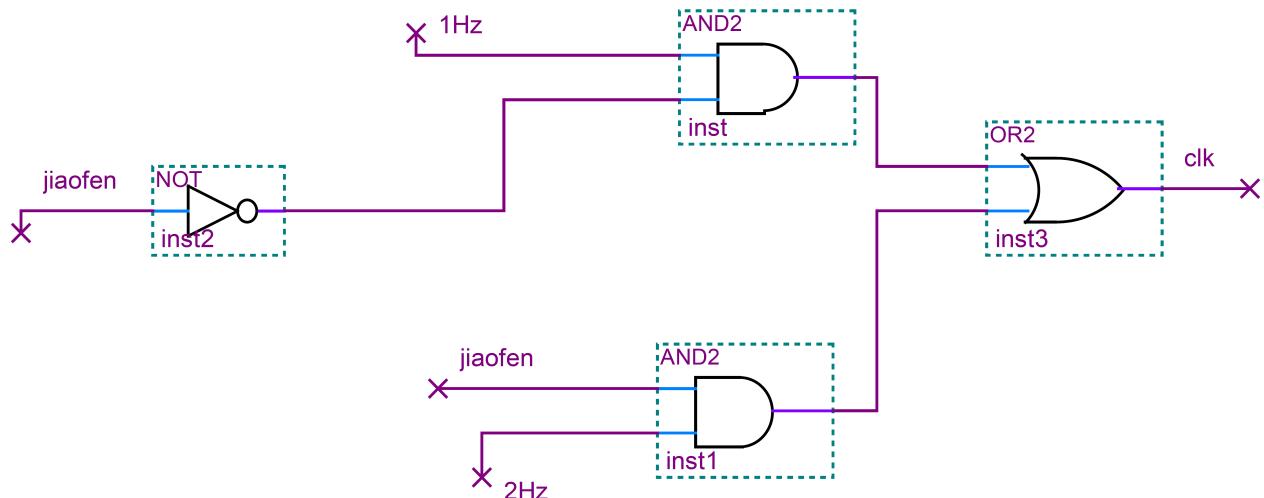


图 25

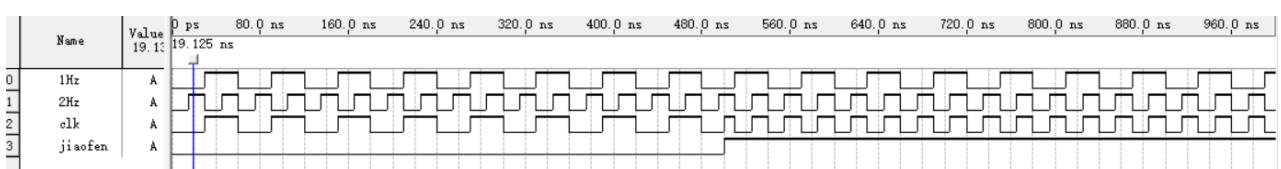


图 26

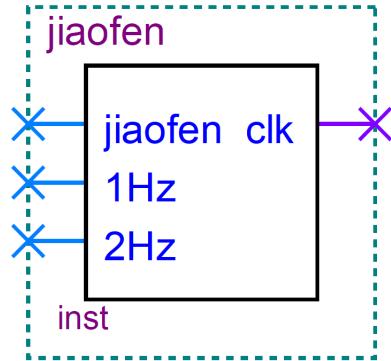


图 27

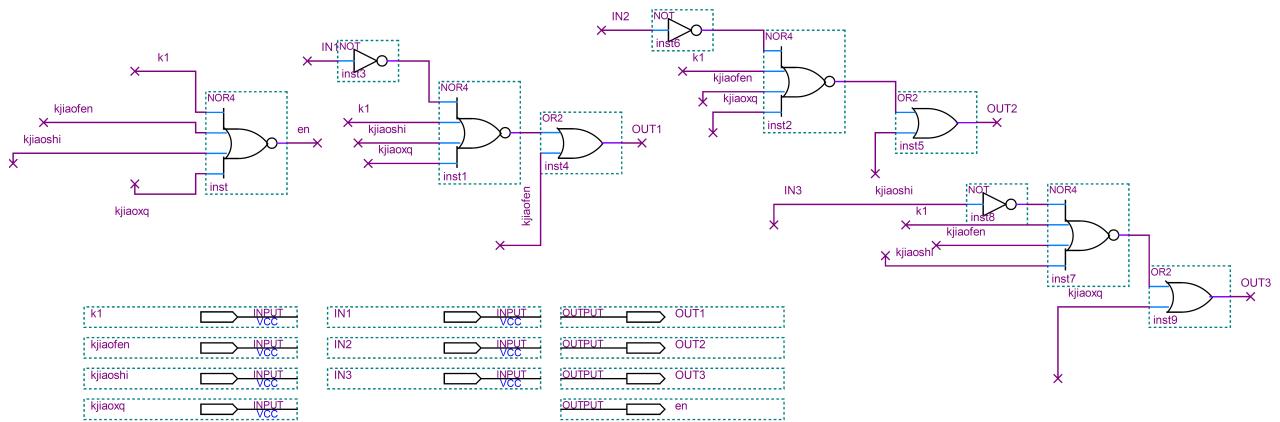


图 28

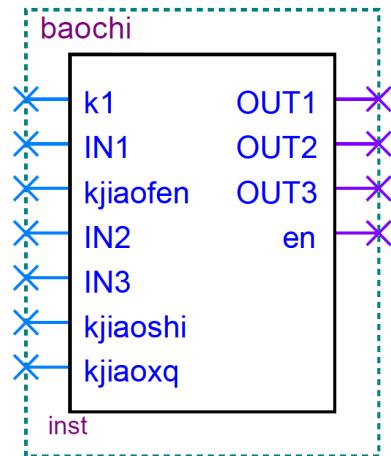


图 29

得到稳定可靠的开关信号，消颤电路的原理图见图30，封装图见图31，整个消颤模块的电路图见图32，封装图见图33。

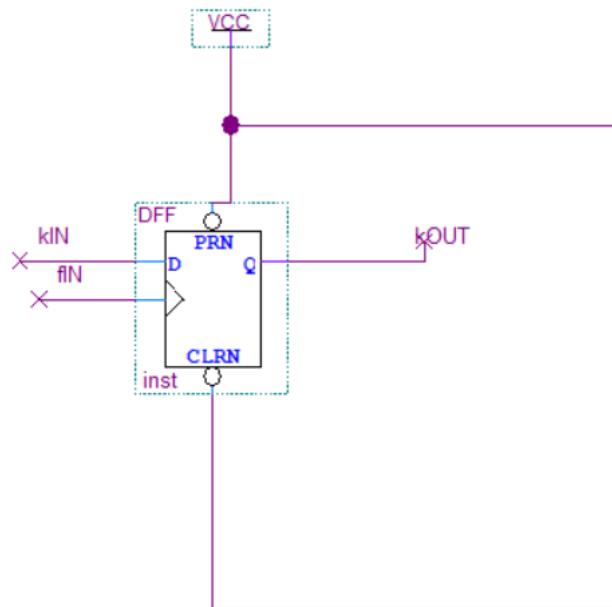


图 30

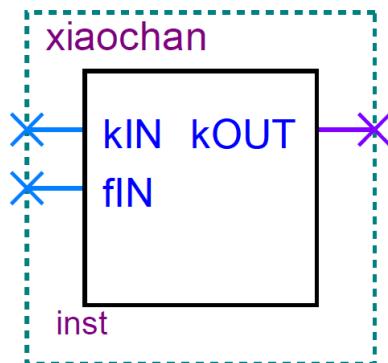


图 31

3.2.5 报时模块

报时模块要求数字钟在 59 分 53 秒、59 分 55 秒、59 分 57 秒时低音报时，在 59 分 59 秒时高音报时，需要使用 1000Hz、500Hz 两种信号分别驱动。列出各个时刻的各位输出状态见表1，分析报时条件。

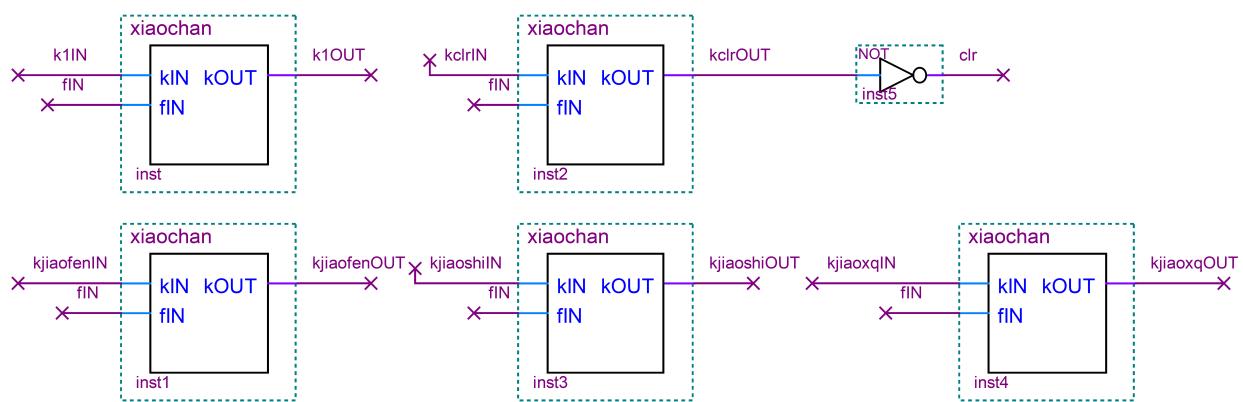


图 32

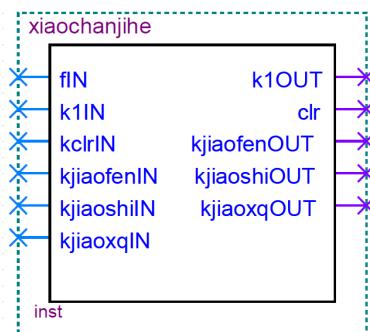


图 33

表 1 报时输出状态

时刻	分十位	分个位	秒十位	秒个位	频率
59分53秒	0101	1001	0101	0011	500Hz
59分55秒	0101	1001	0101	0101	500Hz
59分57秒	0101	1001	0101	0111	500Hz
59分59秒	0101	1001	0101	1001	1kHz

经过分析，可得蜂鸣器的输入端应满足的表达式为：

$$\left\{ \begin{array}{l} F = F_1 \cap F_2 \\ F_1 = fs[3] \cap fs[1] \cap fg[4] \cap fg[1] \cap ms[3] \cap ms[1] \cap mg[1] \\ F_2 = 500HZ \cap mg[2] + 500Hz \cap mg[3] + 1KHz \cap mg[4] \end{array} \right.$$

原理图见图34，封装图见图35。

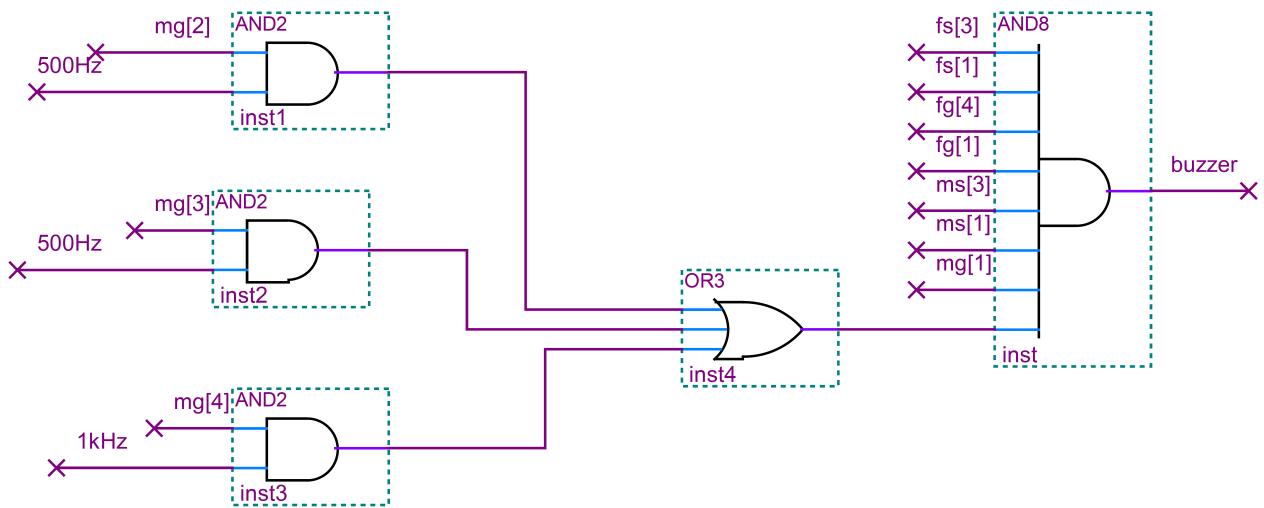


图 34

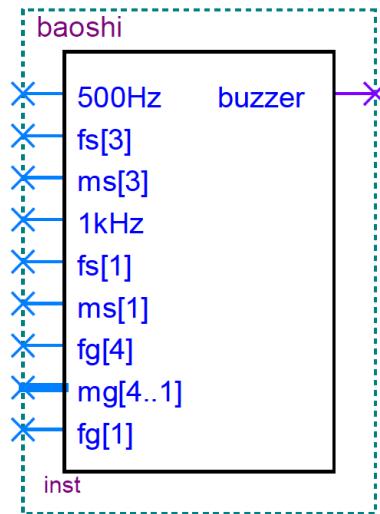


图 35

3.2.6 总计时模块

将以上的各个模块相连，组成总的计时模块。原理图见图36，封装图见图37。

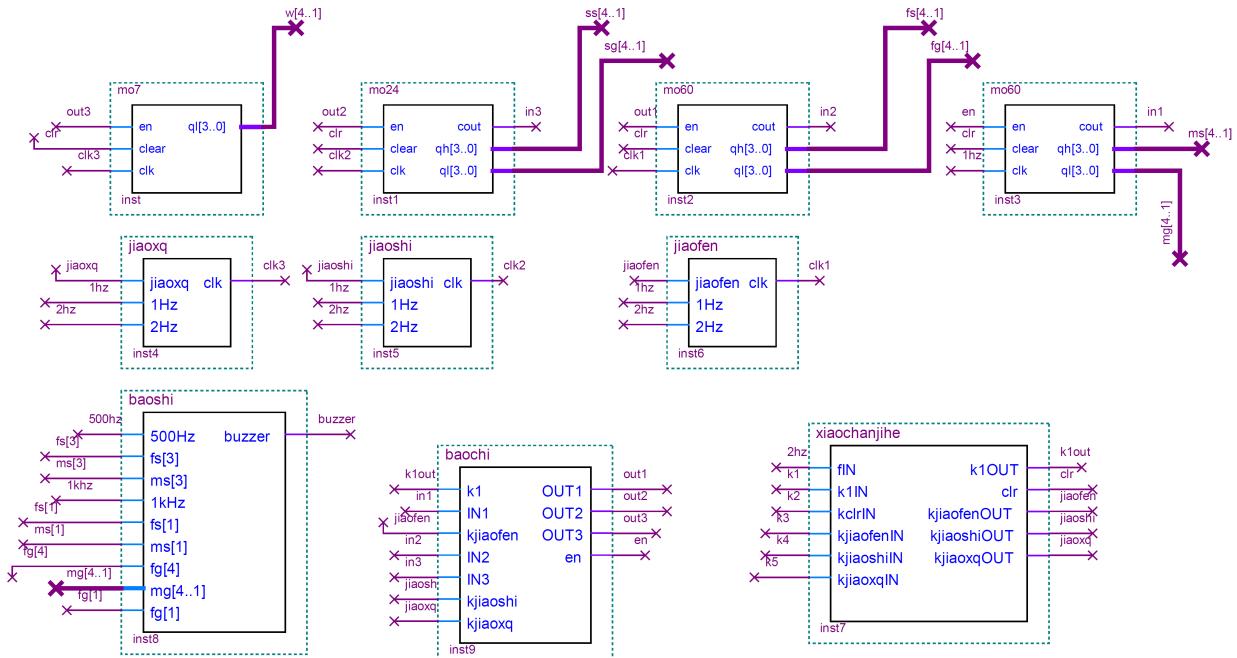


图 36

3.3 音乐模块

这个模块可以看作我学习VHDL语言过程中的一个试验，因此与整个时钟的联系并不是十分紧密，仅仅是通过某个开关（ K_6 ）来控制其有无。

3.3.1 音符的产生

音乐的产生主要是通过输出不同频率的信号来控制蜂鸣器发音，通过查阅资料可以得到音符的频率关系见表2

此时便不方便通过分配器产生音符了，需要通过代码实现。用VHDL语言，对时钟信号进行上升沿检测，并记录检测的数（从0开始计数）。如果是N分频，每当计数到 $\frac{N}{2} - 1$ 时将信号翻转一次，将信号输出。由此，基于表2对48MHz进行分频。得到N的关系见表3

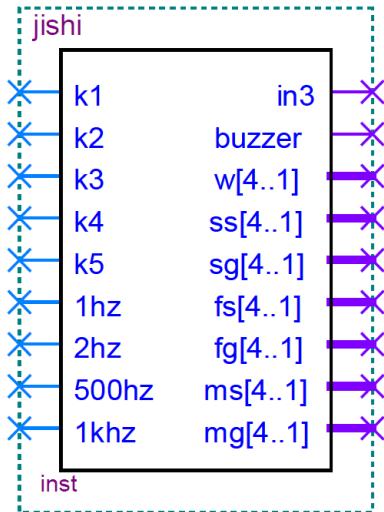


图 37

表 2 各音符对应频率

音符	频率Hz	音符	频率Hz	音符	频率Hz
低1Do	262	中1Do	523	高1Do	1047
低2Re	294	中2Re	587	高2Re	1175
低3Mi	330	中3Mi	659	高3Mi	1319
低4Fa	349	中4Fa	698	高4Fa	1397
低5So	392	中5So	784	高5So	1568
低6La	440	中6La	880	高6La	1760
低7Si	494	中7Si	988	高7Si	1967

表 3 音调与计数间关系

音符	N	音符	N	音符	N
低1Do	91742	中1Do	45861	高1Do	22932
低2Re	81715	中2Re	40864	高2Re	20429
低3Mi	72814	中3Mi	36401	高3Mi	18201
低4Fa	68727	中4Fa	34358	高4Fa	17183
低5So	61223	中5So	30611	高5So	15305
低6La	54544	中6La	27271	高6La	13635
低7Si	48592	中7Si	24295	高7Si	12148

3.3.2 音乐模块

得到各个音符后，设定每个音符的时间，用两只老虎的音调。再用一个变量对整首歌进行计数，这个变量也代表了时间长度，通过循环，使得整首歌不断播放。得到代码如下，封装图见38。

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 ENTITY music IS
4 PORT(
5 clk : in std_logic;
6 beep: out std_logic
7 );
8 end music;
9 architecture beh of music is
10 type state_type is (do_l,re_l,mi_l,fa_l,sol_l,la_l,si_l,
11 do_m,re_m,mi_m,fa_m,sol_m,la_m,si_m,do_h,re_h,mi_h,fa_h
12 ,sol_h,la_h,si_h,none);
13 signal counter : integer range 0 to 100000 := 0;
14 signal count : integer range 0 to 99 := 0;
15 signal beep_reg: std_logic;
16 signal clk1khz,clk4hz : std_logic;
17 signal note : state_type ;
18 begin
19 beep <= beep_reg;
20 beep_pro : process(clk)
21 variable cnt : integer range 0 to 100000 := 0;
22 begin
23 if clk'event and clk='1' then
24 if cnt < counter then
25 cnt := cnt + 1 ;
26 else
```

```

25 cnt := 0 ; beep_reg <= not beep_reg ;
26 end if ;
27 end if ;
28 end process beep_pro ;
29 clk1khz_pro : process(clk)
30 variable cnt : integer range 0 to 23999;
31 begin
32 if clk'event and clk='1' then
33 if cnt = 23999 then
34 cnt := 0 ; clk1khz <= not clk1khz ;
35 else
36 cnt := cnt + 1;
37 end if ;
38 end if ;
39 end process clk1khz_pro ;
40 clk4hz_pro :process(clk1khz)
41 variable cnt : integer range 0 to 124 := 0;
42 begin
43 if clk1khz'event and clk1khz = '1' then
44 if cnt = 124 then
45 cnt := 0 ; clk4hz <= not clk4hz ;
46 else
47 cnt := cnt + 1;
48 end if ;
49 end if ;
50 end process clk4hz_pro ;
51 count_pro : process(clk4hz)
52 begin
53 if clk4hz'event and clk4hz ='1' then
54 if count <= 66 then
55 count <= count + 1;

```

```

56 else
57 count <= 0;
58 end if;
59 end if;
60 end process count_pro;
61 note_pro : process( note )
62 begin
63 case note is
64 when do_l => counter <= 91742 ;
65 when re_l => counter <= 81715 ;
66 when mi_l => counter <= 72814 ;
67 when fa_l => counter <= 68727 ;
68 when sol_l => counter <= 61223 ;
69 when la_l => counter <= 54544 ;
70 when si_l => counter <= 48592 ;
71 when do_m => counter <= 45861 ;
72 when re_m => counter <= 40864 ;
73 when mi_m => counter <= 36401 ;
74 when fa_m => counter <= 34358 ;
75 when sol_m => counter <= 30611 ;
76 when la_m => counter <= 27271 ;
77 when si_m => counter <= 24295 ;
78 when do_h => counter <= 22932 ;
79 when re_h => counter <= 20429 ;
80 when mi_h => counter <= 18201 ;
81 when fa_h => counter <= 17183 ;
82 when sol_h => counter <= 15305 ;
83 when la_h => counter <= 13635 ;
84 when si_h => counter <= 12148 ;
85 when others => counter <= 0;
86 end case;

```

```

87 end process note_pro;
88 music_pro : process(count)
89 begin
90 case count is when 0 => note <= do_m ;
91 when 1 => note <= do_m ;
92 when 3 => note <= re_m ;
93 when 5 => note <= mi_m ;
94 when 7 => note <= do_m ;
95 when 9 => note <= do_m ;
96 when 11 => note <= re_m ;
97 when 13 => note <= mi_m ;
98 when 15 => note <= do_m ;
99 when 17 => note <= mi_m ;
100 when 19 => note <= fa_m ;
101 when 21 => note <= sol_m ;
102 when 22 => note <= sol_m ;
103 when 23 => note <= sol_m ;
104 when 25 => note <= mi_m ;
105 when 27 => note <= fa_m ;
106 when 29 => note <= sol_m ;
107 when 30 => note <= sol_m ;
108 when 31 => note <= sol_m ;
109 when 33 => note <= sol_m ;
110 when 34 => note <= la_m ;
111 when 35 => note <= sol_m ;
112 when 36 => note <= fa_m ;
113 when 37 => note <= mi_m ;
114 when 38 => note <= mi_m ;
115 when 39 => note <= do_m ;
116 when 40 => note <= do_m ;
117 when 41 => note <= sol_m ;

```

```

118 when 42 => note <= la_m ;
119 when 43 => note <= sol_m ;
120 when 44 => note <= fa_m ;
121 when 45 => note <= mi_m ;
122 when 46 => note <= mi_m ;
123 when 47 => note <= do_m ;
124 when 48 => note <= do_m ;
125 when 49 => note <= re_m ;
126 when 50 => note <= re_m ;
127 when 51 => note <= sol_l ;
128 when 52 => note <= sol_l ;
129 when 53 => note <= do_m ;
130 when 54 => note <= do_m ;
131 when 55 => note <= do_m ;
132 when 57 => note <= re_m ;
133 when 58 => note <= re_m ;
134 when 59 => note <= sol_l ;
135 when 60 => note <= sol_l ;
136 when 61 => note <= do_m ;
137 when 62 => note <= do_m ;
138 when 63 => note <= do_m ;
139 when others => note <= none ;
140 end case ;
141 end process music_pro ;
142 end beh ;

```

3.4 显示模块

在数码管上的显示为动态扫描，因为同一时间只能有一个数码管有效。而当扫描的频率比较大时，就可以认为所有的数码管都是亮的。数码管有八个，而数字有七个，为了美观，在星期与时分秒中间加入了短横线。显示模块由

模8模块、八选一模块、七段译码器模块、三八译码器模块组成。

3.4.1 模8计数器

模8计数器是为了同时只让一个数码管有效，同时作为八选一选择器的输入端，选择输出的数字。其原理与计时电路中的原理类似，代码如下，封装图见图39。

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 ENTITY mo8 IS
5 PORT
6 (
7 en :IN std_logic;
8 clk :IN std_logic;
9 q :buffer std_logic_vector(2 downto 0)
10 );
11 END mo8;
12 ARCHITECTURE beh OF mo8 IS
13 BEGIN
14 PROCESS(clk)
15 BEGIN
16 IF (clk 'EVENT and clk = '1') THEN
17 if (en = '1') then
18 q<=q+1;
19 end if;
20 end if;
21 END PROCESS;
22 END beh;
```

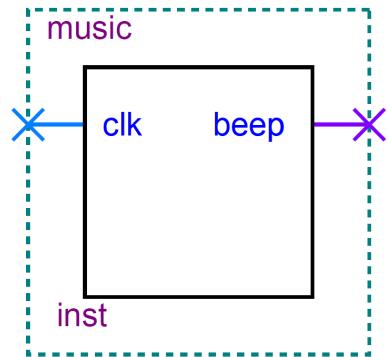


图 38

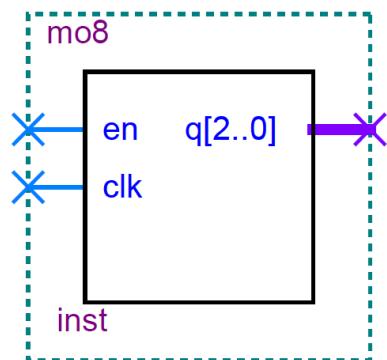


图 39

3.4.2 八选一

八选一用来选择当前有效的数码管应该显示的数字，有星期、短横线、时十位、时个位、分十位、分个位、十位、秒个位，这八种状态。其中短横线用1111来代替，而在之后的七段译码器中，将1111译为数码管中只有 g 段亮的短横线。代码如下，封装图见图40。

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 entity chose8to1 is
5 port(wl, hh, hl, fh, fl, mh, ml: in std_logic_vector(3 downto 0)
6 ;
7 sel:in std_logic_vector(2 downto 0);
8 q:out std_logic_vector(3 downto 0));
9 end chose8to1;
10 architecture chose of chose8to1 is
11 begin
12 process(sel, wl, hh, hl, fh, fl, mh, ml)
13 begin
14 case sel is
15 when "000"=>q<=wl;--week
16 when "001"=>q<="1111";--short line, to devide week and
times
17 when "010"=>q<=hh;--shiwei of hour
18 when "011"=>q<=hl;--gewei of hour
19 when "100"=>q<=fh;--shiwei of minite
20 when "101"=>q<=fl;--gewei of minite
21 when "110"=>q<=mh;--shiwei of seconds
22 when "111"=>q<=ml;--gewei of seconds
23 when others=>q<="0000";-- don't care
end case;
```

```

24 end process;
25 end chose;

```

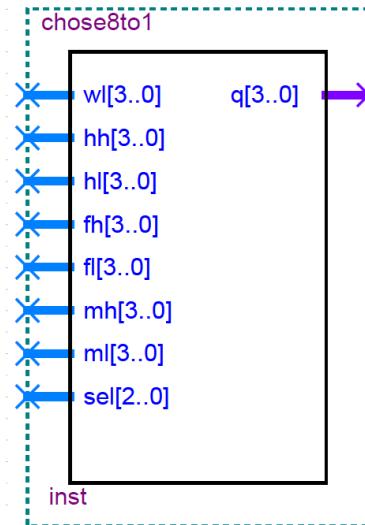


图 40

3.4.3 3线-8线译码器

将计数器输出的三位数据译码成八个状态，作为数据分配器使用，控制数码管有效与否。代码如下，封装图见图41。

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 entity translate3to8 is
5 port(a:in std_logic_vector(2 downto 0);
6 y:out std_logic_vector(7 downto 0));
7 end translate3to8;
8 architecture translate of translate3to8 is
9 begin
10 process(a)
11 begin
12 case a is
13 when "000"=>y<="01111111";--wk

```

```

14 when "001"=>y<="10111111";—short line
15 when "010"=>y<="11011111";
16 when "011"=>y<="11101111";
17 when "100"=>y<="11110111";
18 when "101"=>y<="11111011";
19 when "110"=>y<="11111101";
20 when "111"=>y<="11111110";
21 when others=>y<="11111111";—don't choose
22 end case;
23 end process;
24 end translate;

```

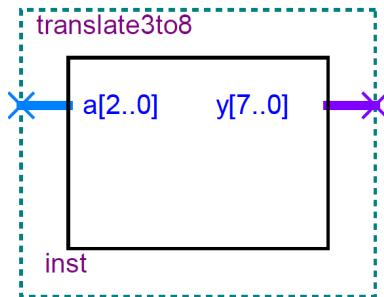


图 41

3.4.4 七段译码器

将二进制的数字译码为数码管中 a、b、c、d、e、f、g 段的显示情况，能直观地显示出数字来。我们所用的数码管为共阳。其中，转换方案见表4。由此，得到代码如下，封装图见图42

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 entity yimaqi is
5 port(a:in std_logic_vector(3 downto 0);
6 y:out std_logic_vector(6 downto 0));
7 end yimaqi;

```

```

8 architecture yima of yimaqi is
9 begin
10 process(a)
11 begin
12 case a is--0 means light;1 means dark
13 when "0000"=>y<="1000000";
14 when "0001"=>y<="1111001";
15 when "0010"=>y<="0100100";
16 when "0011"=>y<="0110000";
17 when "0100"=>y<="0011001";
18 when "0101"=>y<="0010010";
19 when "0110"=>y<="0000010";
20 when "0111"=>y<="1111000";
21 when "1000"=>y<="0000000";
22 when "1001"=>y<="0010000";
23 when "1111"=>y<="0111111";
24 when others=>y<="1111111";--no words
25 end case;
26 end process;
27 end yima;

```

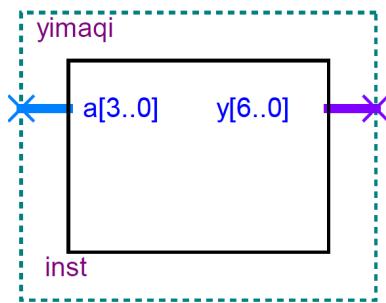


图 42

3.4.5 显示模块

将以上几个部分组合，得到显示模块如图43，封装图见图44。

字型	a	b	c	d	e	f	g
0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	1
2	0	0	1	0	0	1	0
3	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0
5	0	1	0	0	1	0	0
6	0	1	0	0	0	0	0
7	0	0	0	1	1	1	1
8	0	0	0	0	0	0	0
9	0	0	0	0	1	0	0

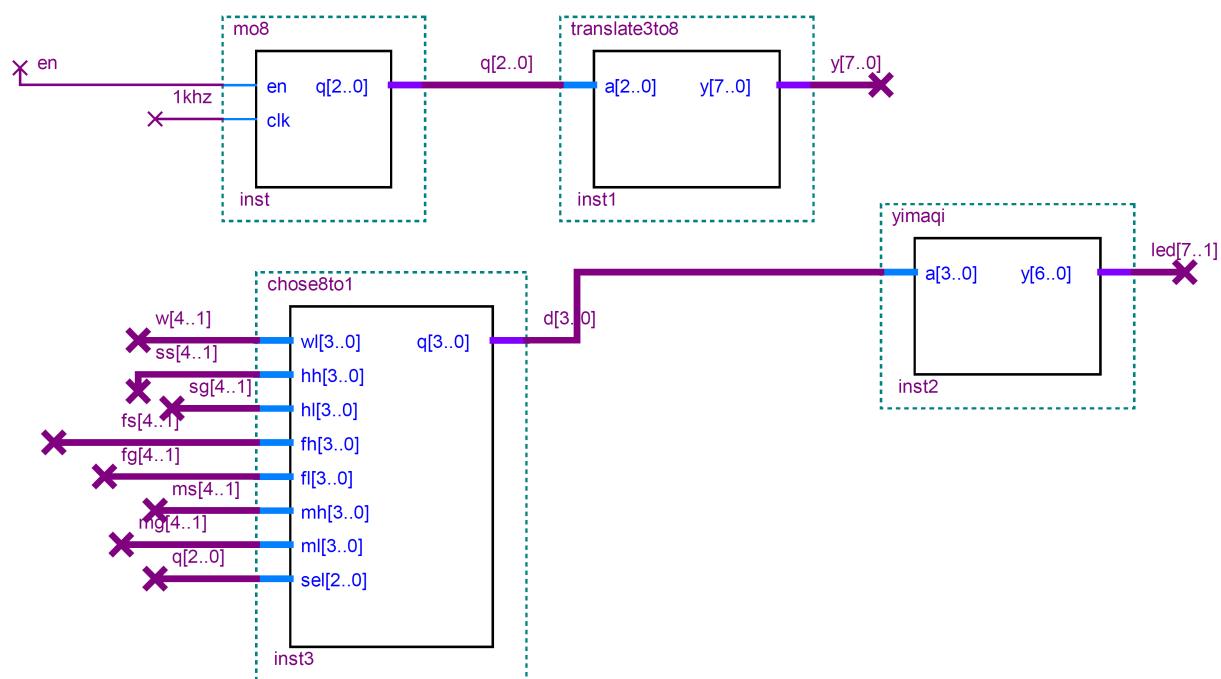


图 43

3.5 万年历模块

要实现万年历，即是对日、月、年的显示，其中日有28、29、30、31四种情况。而年有润年平年之分。在我的时钟中，是通过每四年输出一个信号来判断润年与否的，因此对于整百年的情况，这里无法判断。

3.5.1 日计数

根据年、月的不同，日会有28、29、30、31四种情况，因此引入一个2位的judge信号，根据judge确定进制。juege为00时，采用28进制；juege为01时，采用29进制；juege为10时，采用30进制；juege为11时，采用31进制。VHDL代码如下，仿真情况见图45~48，封装图见49。

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 ENTITY day IS
5 PORT
6 (
7 clear :IN std_logic;
8 judge :IN std_logic_vector(1 downto 0);--judge how many days
9 clk :IN std_logic;
10 dh :buffer std_logic_vector(3 downto 0);
11 dl :buffer std_logic_vector(3 downto 0);
12 rco : OUT std_logic;
13 en :IN std_logic
14 );
15 END day;
16 ARCHITECTURE beh OF day IS
17 BEGIN
18 PROCESS(clk, judge, clear)
19 BEGIN
```

```

20 IF( clk 'EVENT and clk ='1')THEN
21 IF( en='1')then
22 IF( clear ='0')THEN
23 dh<=" 0000 "; dl<=" 0000 ";
24 else
25 case judge is
26 when" 00 "=>—28
27 if( dh=2 and dl=8)then
28 rco <='1'; dh<=" 0000 "; dl<=" 0001 ";
29 else if( dl=9)then
30 dh<=dh+1; dl<=" 0000 "; rco <='0';
31 else rco <='0'; dl<=dl+1;
32 end if;
33 end if;
34 when" 01 "=>—29
35 if( dh=2 and dl=9)then
36 rco <='1'; dh<=" 0000 "; dl<=" 0001 ";
37 else if( dl=9)then
38 dh<=dh+1; dl<=" 0000 "; rco <='0';
39 else rco <='0'; dl<=dl+1;
40 end if;
41 end if;
42 when" 10 "=>—30
43 if( dh=3 and dl=0)then
44 rco <='1'; dh<=" 0000 "; dl<=" 0001 ";
45 else if( dl=9)then
46 dh<=dh+1; dl<=" 0000 "; rco <='0';
47 else rco <='0'; dl<=dl+1;
48 end if;
49 end if;
50 when others =>—31

```

```

51 if (dh=3 and dl=1)then
52 rco <='1';dh<="0000";dl<="0001";
53 else if (dl=9)then
54 dh<=dh+1;dl<="0000";rco <='0';
55 else rco <='0';dl<=dl+1;
56 end if;
57 end if;
58 end case;
59 end if;
60 end IF;
61 end if;
62 END PROCESS;
63 END beh;

```

3.5.2 月计数

对于月而言，就是一个模12计数器，VHDL代码如下，仿真情况见图50，封装图见51。

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 ENTITY month IS
5 PORT
6 ( en :IN std_logic;
7 clear:IN std_logic;—when 0, output is always 0
8 clk :IN std_logic;
9 rco :out std_logic;—jinwei output
10 mh :buffer std_logic_vector(3 downto 0);—shiwei output
11 ml :buffer std_logic_vector(3 downto 0)
12 );—gewei output
13 END month;

```

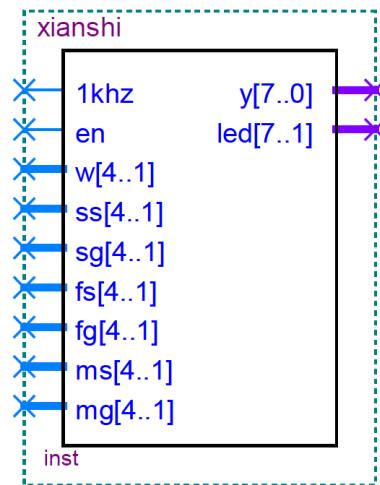


图 44

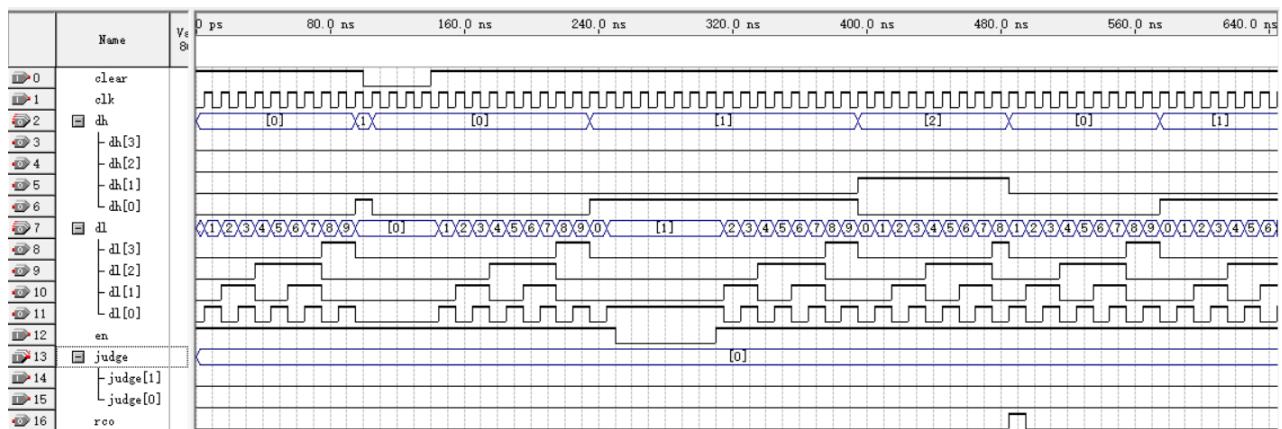


圖 45

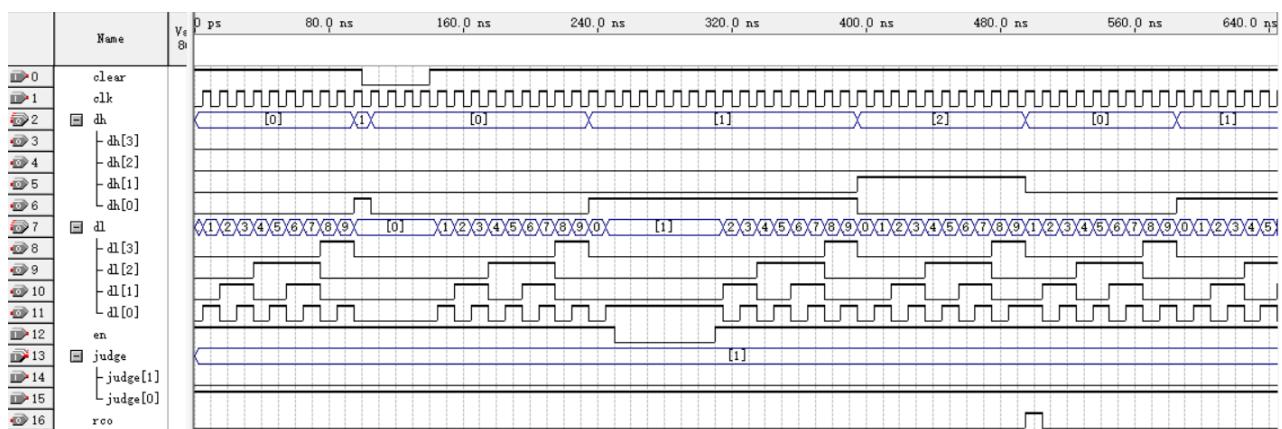


圖 46

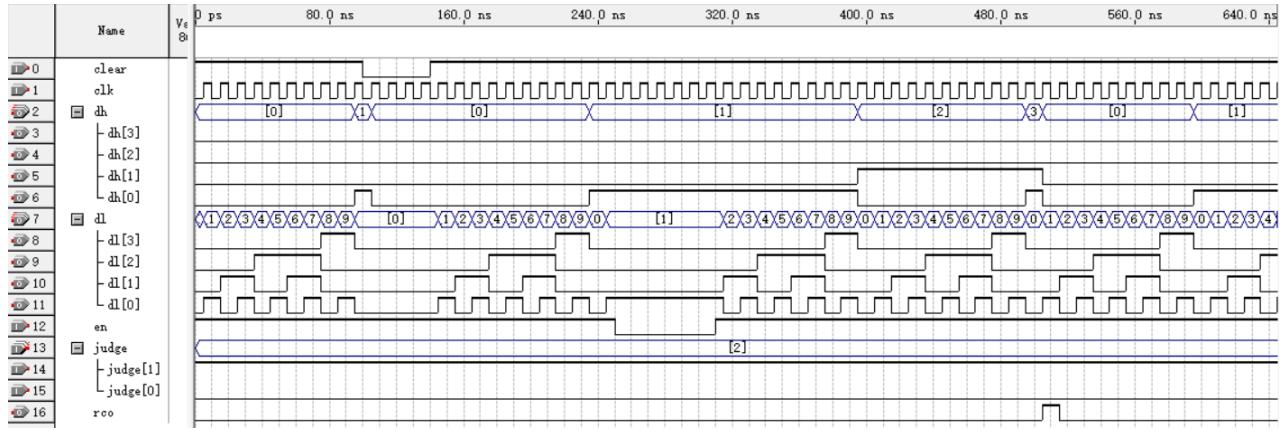


图 47

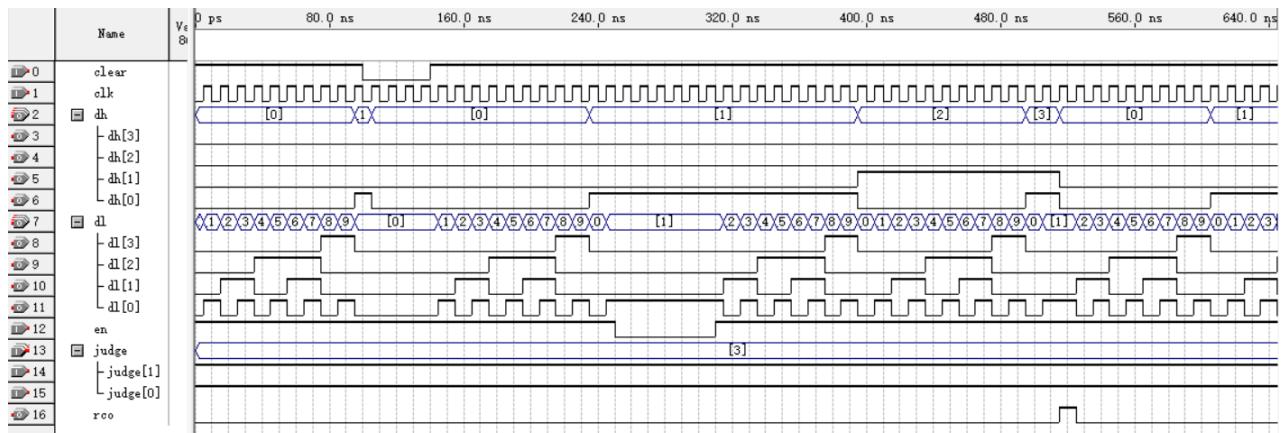


图 48

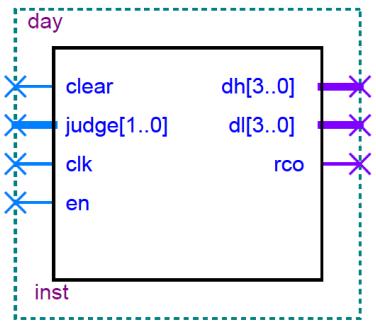


图 49

```

14 ARCHITECTURE beh OF month IS
15 BEGIN
16 rco <='1'when(mh="0001"and ml="0010"and en='1')else '0';
17
18 PROCESS(clk, clear)
19 BEGIN
20 IF (clear='0')THEN
21 mh<="0000";
22 ml<="0000";
23 ELSIF (clk'EVENT AND clk='1')THEN
24 if(en='1')then
25 if((ml=9) or (ml=2 and mh=1))then
26 if(ml=2 and mh=1)then
27 ml<="0001";
28 mh<="0000";
29 else ml<="0000";
30 mh<="0001";
31 end if;
32 else ml<=ml+1;
33 end if;
34 end if;
35 END IF;
36 END PROCESS;
37 END beh;

```

3.5.3 年计数

对于年而言，需要输出一个信号，表明本年是否是润年，这里采用的方法是每四年输出一个run信号，judge可由此判断是否为润年。VHDL代码如下，仿真情况见图52，封装图见53。

```

1 library ieee;
```

```

2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 ENTITY year IS
5 PORT
6 (
7 clear :IN std_logic;
8 clk :IN std_logic;
9 run :OUT std_logic;
10 r :buffer std_logic_vector(1 downto 0);
11 y1 :buffer std_logic_vector(3 downto 0);
12 y2 :buffer std_logic_vector(3 downto 0);
13 y3 :buffer std_logic_vector(3 downto 0);
14 y4 :buffer std_logic_vector(3 downto 0);
15 en :IN std_logic
16 );
17 END year;
18 ARCHITECTURE beh OF year IS
19 BEGIN
20 PROCESS(clk, clear)
21 BEGIN
22
23 IF (clk 'EVENT and clk = '1') THEN
24 IF (clear = '0') THEN
25 y1<="0000"; y2<="0000"; y3<="0000"; y4<="0000"; r<="00";
26 else
27 if (en = '1') then
28 if (y4=9)then
29 y4<="0000"; y3<=y3+1;
30 else if (y3=9)then
31 y3<="0000"; y2<=y2+1;
32 else if (y2=9)then

```

```

33 y2<="0000";y1<=y1+1;
34 else if(y1=9)then
35 y1<="0000";
36 else y4<=y4+1;
37 end if;
38 end if;
39 end if;
40 end if;
41 if(r=3)then
42 run<='1';r<="00";
43 else r<=r+1;run<='0';
44 end if;
45 end IF;
46 end if;
47 end if;
48 END PROCESS;
49 END beh;
```

3.5.4 日进制判断信号judge

judge需要通过当前年份（run）来判断是否为润年，结合月最终给出每个月的具体日数，VHDL代码如下，仿真情况见图54，封装图见55。

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 ENTITY judge IS
5 PORT
6 (
7 run :IN std_logic;
8 mh :IN std_logic_vector(3 downto 0);
9 ml :IN std_logic_vector(3 downto 0);
```

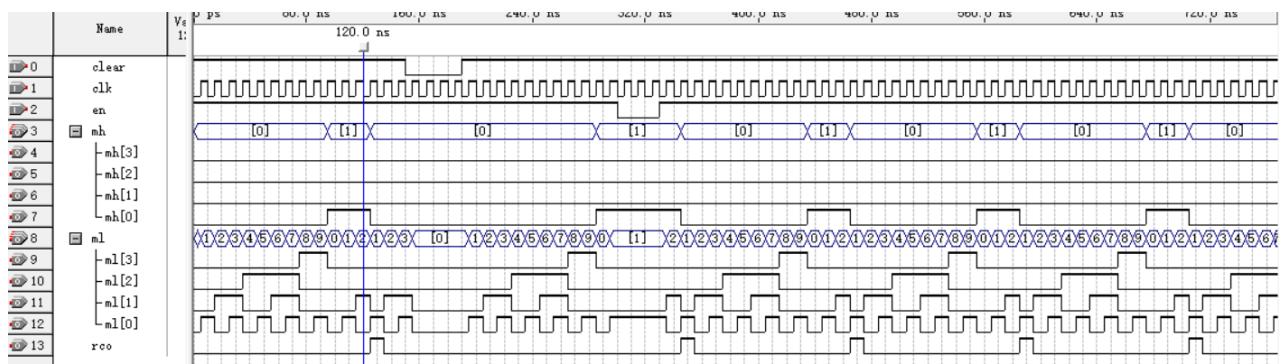
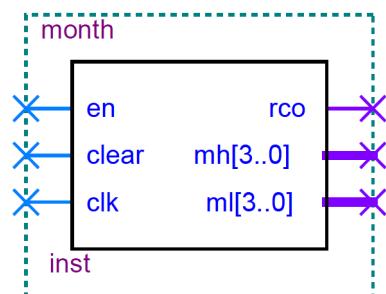


图 50



冬 51

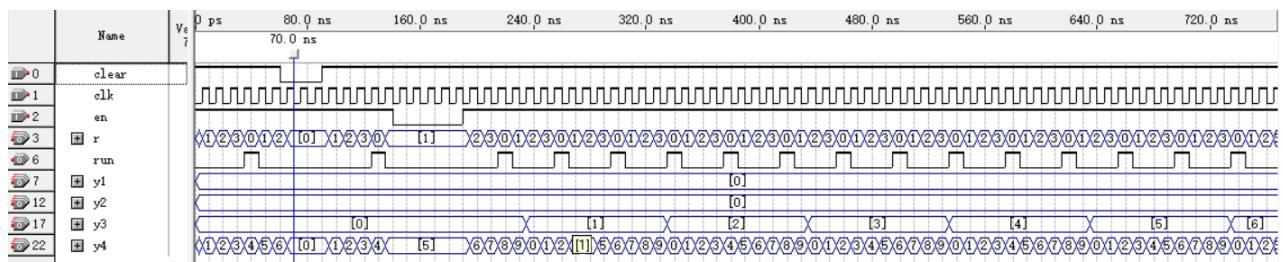


图 52

```

10 j : buffer std_logic_vector(1 downto 0)
11 );
12 END judge;
13 ARCHITECTURE beh OF judge IS
14 BEGIN
15 PROCESS(run ,ml ,mh)
16 BEGIN
17 IF (ml=1 or ml=3 or ml=5 or ml=7 or ml=8 or ( ml=0 and mh
    =1) or ( ml=2 and mh=1 ) )THEN
18 j<="11";
19 else if( ml=4 or ml=6 or ml=9 or ( ml=1 and mh=1 ) )then
20 j<="10";
21 else if(( ml=2 and mh=0) and run='1')then
22 j<="01";
23 else j<="00";
24 end if;
25 end if;
26 end IF;
27 END PROCESS;
28 END beh ;

```

3.5.5 万年历模块

将以上计数器组合成万年历模块如图56，仿真波形大致情况见图57，封装图见图58。

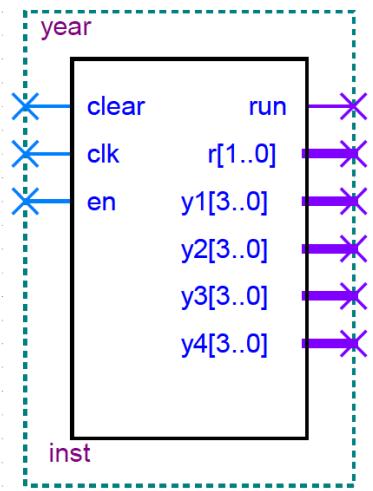


图 53

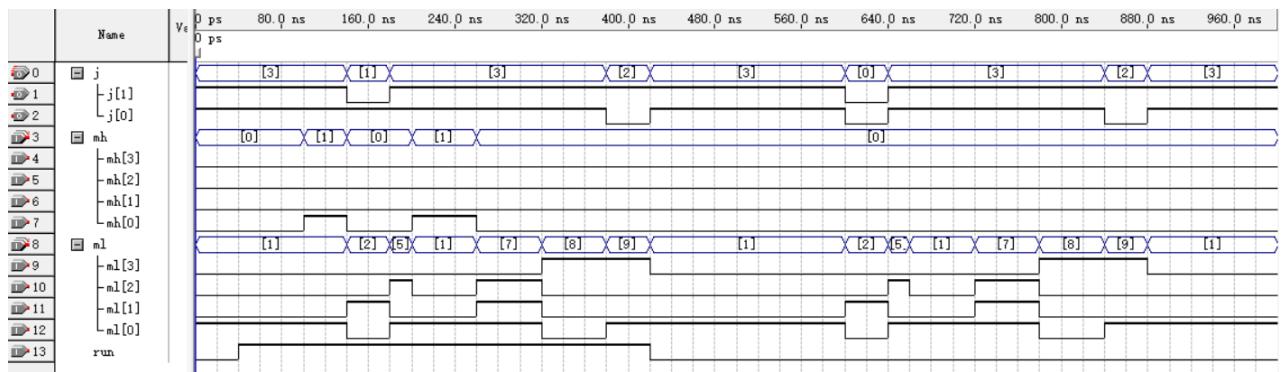


图 54

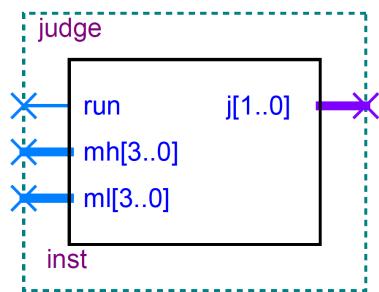


图 55

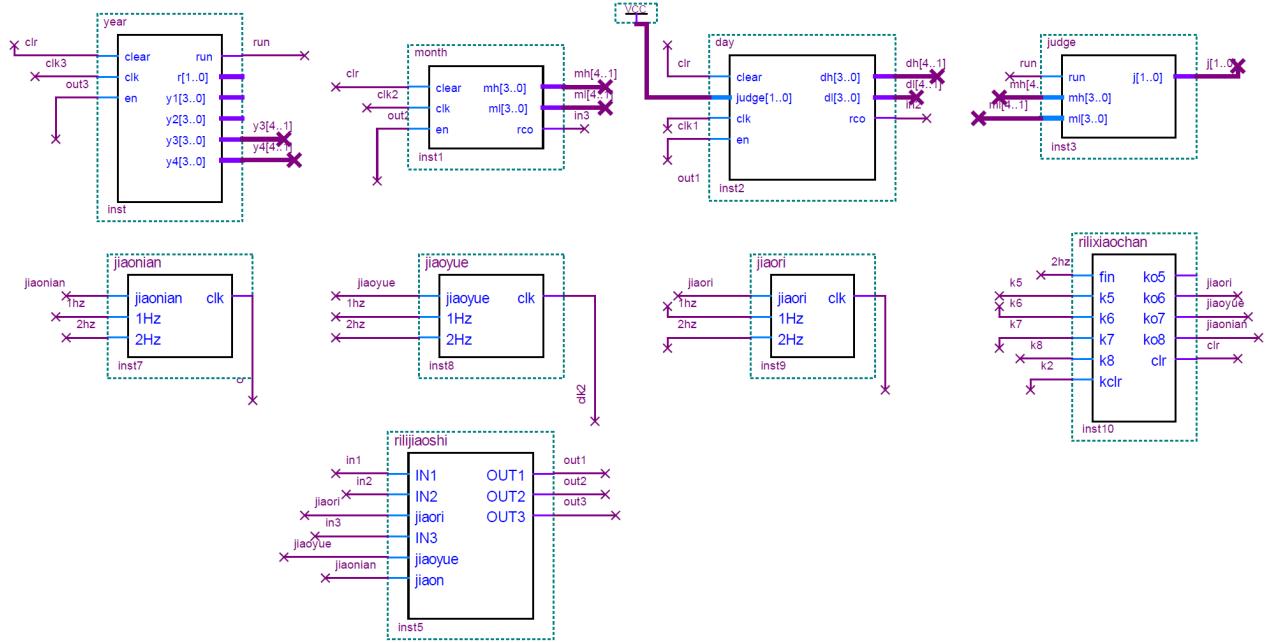


图 56

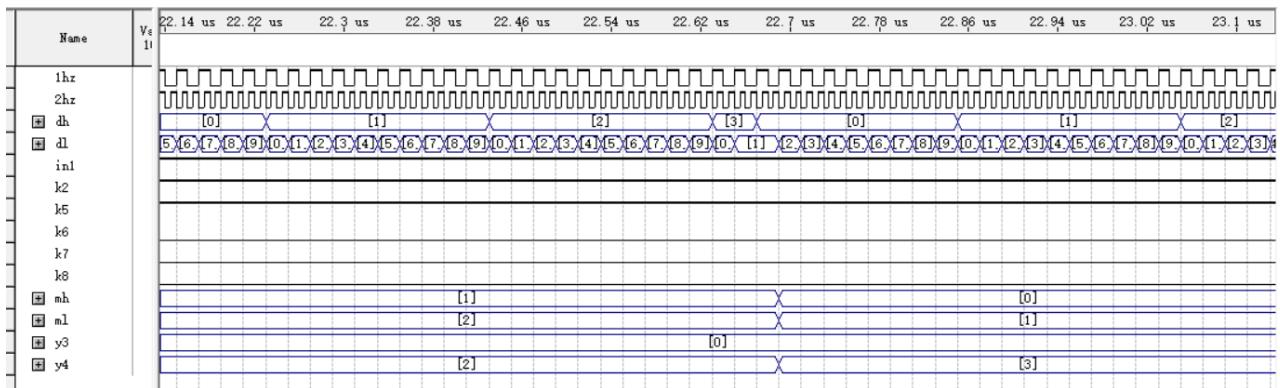


图 57

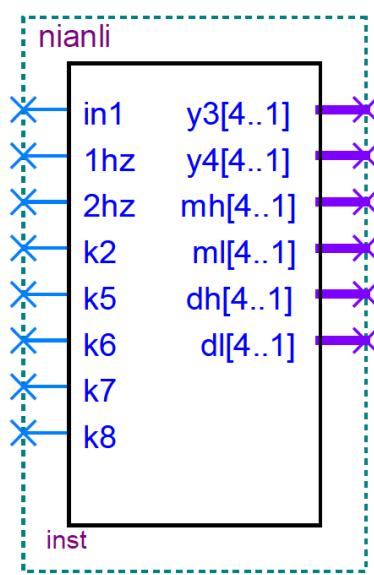
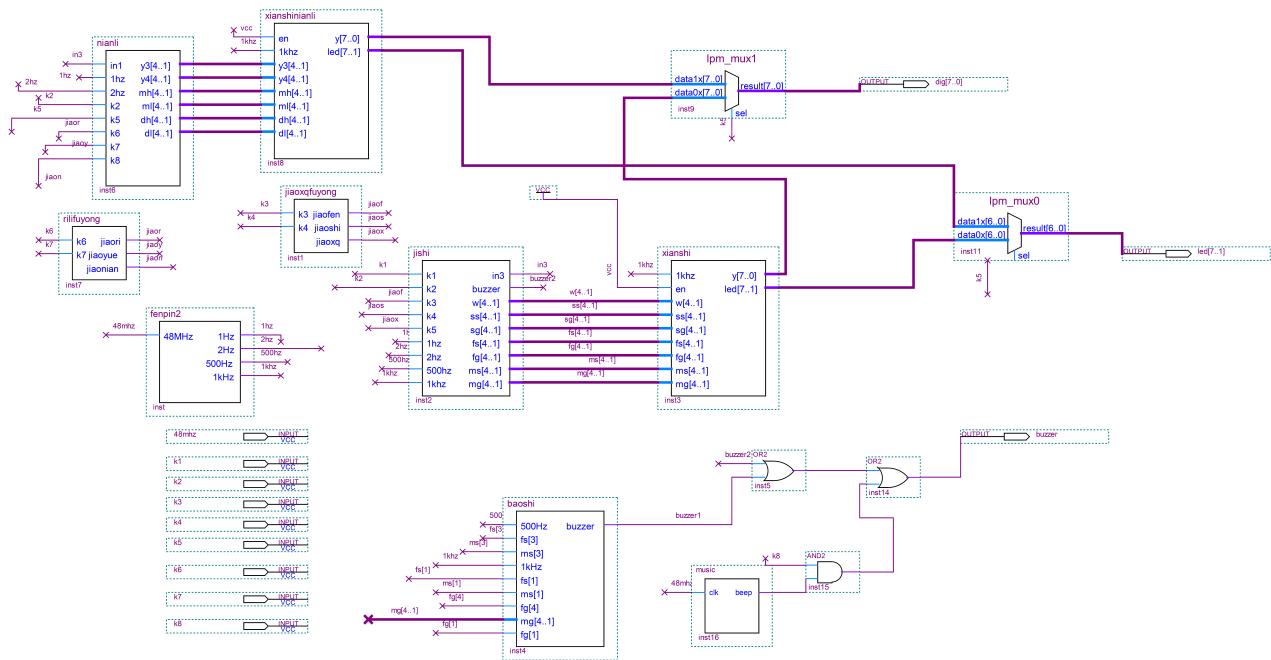


图 58

4 多功能数字时钟总电路

将以上几个模块相连，借助 IPM_MUX 实现显示复用，借助74139实现开关复用，多功能数字时钟总电路如图59。



冬 59

5 调试、仿真、编程下载

电路设计完成后，还要对已完成的电路图保存后进行编译，检查有无出错的地方，比如线路定义出错，引脚未定义等软件能检查出来的错误，这是最基本的检查，只能检查出一些语法运用的错误。

然后还要对其进行波形仿真验证，来检查电路设计的正确与否，直至调试的仿真图完全正确为止，仿真得到的波形图在各个模块的论证时已经给出。

波形仿真完成后，就要下载到实验箱上进行验收。下载方法比较繁琐，通常要设置一些程序的初始值、使能端等，但这是必须的一步，否则可能烧坏实验箱上的某些原件，造成较大的损失。

将编译好的程序管脚分配，然后下载到实验系统中进行调试和验证。其中管脚分配如图60。

	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
1	48mhz	Input	PIN_A10	7	B7_N0	2.5 V (default)	
2	buzzer	Output	PIN_F8	8	B8_N0	2.5 V (default)	
3	dig[7]	Output	PIN_J13	6	B6_N0	2.5 V (default)	
4	dig[6]	Output	PIN_H16	6	B6_N0	2.5 V (default)	
5	dig[5]	Output	PIN_L16	5	B5_N0	2.5 V (default)	
6	dig[4]	Output	PIN_L14	5	B5_N0	2.5 V (default)	
7	dig[3]	Output	PIN_D16	7	B7_N0	2.5 V (default)	
8	dig[2]	Output	PIN_A18	7	B7_N0	2.5 V (default)	
9	dig[1]	Output	PIN_E14	7	B7_N0	2.5 V (default)	
10	dig[0]	Output	PIN_H15	6	B6_N0	2.5 V (default)	
11	k1	Input	PIN_V9	3	B3_N0	2.5 V (default)	
12	k2	Input	PIN_U10	4	B4_N0	2.5 V (default)	
13	k3	Input	PIN_B9	8	B8_N0	2.5 V (default)	
14	k4	Input	PIN_B10	7	B7_N0	2.5 V (default)	
15	k5	Input	PIN_R18	5	B5_N0	2.5 V (default)	
16	k6	Input	PIN_R17	5	B5_N0	2.5 V (default)	
17	k7	Input	PIN_P18	5	B5_N0	2.5 V (default)	
18	k8	Input	PIN_P17	5	B5_N0	2.5 V (default)	
19	led[7]	Output	PIN_C14	7	B7_N0	2.5 V (default)	
20	led[6]	Output	PIN_E13	7	B7_N0	2.5 V (default)	
21	led[5]	Output	PIN_D12	7	B7_N0	2.5 V (default)	
22	led[4]	Output	PIN_C12	7	B7_N0	2.5 V (default)	
23	led[3]	Output	PIN_F12	7	B7_N0	2.5 V (default)	
24	led[2]	Output	PIN_E11	7	B7_N0	2.5 V (default)	
25	led[1]	Output	PIN_F10	7	B7_N0	2.5 V (default)	

图 60

6 结论

运行后可实现的功能有：

- 1、开关全部置0时，进行星期、时、分、秒计时，由7个数码管分别显示星期、时、分、秒，有整点报时功能；
- 2、 K_1 是系统的使能开关（ $K_1 = 0$ 正常工作， $K_1 = 1$ 时钟保持不变）；
- 3、 K_2 是系统的清零开关（ $K_2 = 0$ 正常工作， $K_2 = 1$ 时钟的所有位，包括万年历部分，全部清零）；
- 4、 K_3 是系统的校分开关（ $K_3 = 0$ 正常工作， $K_3 = 1$ 时可以快速校分）；
- 5、 K_4 是系统的校时开关（ $K_4 = 0$ 正常工作， $K_4 = 1$ 时可以快速校时）；
- 6、而 K_4, K_5 合起来进行校星期，当 K_4, K_5 均为1时，可以快速校星期；
- 7、 K_5 是万年历显示开关（ $K_5 = 0$ 显示星期、时、分、秒， $K_5 = 1$ 时显示年-月-日）；
- 8、 K_6 是系统的校日开关（ $K_6 = 0$ 正常工作， $K_6 = 1$ 时可以快速校日）；
- 9、 K_7 是系统的校月开关（ $K_7 = 0$ 正常工作， $K_7 = 1$ 时可以快速校月）；
- 10、而 K_6, K_7 合起来进行校年，当 K_6, K_7 均为1时，可以快速校年；
- 11、 K_8 是音乐开关（ $K_8 = 0$ 无音乐， $K_8 = 1$ 时循环播放两只老虎的音乐）；
- 12、界面更直观，星期显示，则星期与时分秒间用短横线间隔显示；万年历显示，则年、月、日间用短横线间隔显示（年只显示低2位）。

7 实验感想

7.1 实验过程中遇到的问题及解决问题的方法

实验过程中遇到的问题主要有：

- 1、在仿真总分频器时，发现仿真波形结果根本没有变化。我以为是电路出了问题。后来经过同学的提醒，是因为分48MHz需要很长的截止时间。短时间仿真得不到变化波形是正常的。于是我在仿真时去掉了1000分频（也作为实验中的仿真波形图收录在报告中），终于产生了变化的波形。
- 2、在下载到实验箱运行时也出现了问题，实验板上没有任何的显示。在多次检查了电路后，没有发现原理性的错误，这使我有点不知所措。后来我重新检查管脚分配，发现是管脚分配有问题，改正这个错误后，就显示出来了。
- 3、在实验报时功能时，蜂鸣器并没有报时，我再次一个模块一个模块地检查检查电路，发现是因为写错了结点的名称，导致报时功能失效。改正后，报时模块可以实现报时功能。
- 4、在做提高功能万年历模块时，由于对于日的多种情况不知道怎么处理，一度毫无进展。后来参考了网上的一些方案，联想到可以用一个变量来确定具体的日子。

7.2 实验的收获与感受

原本以为多功能数字钟的设计和电工电子综合实验II差不多，只不过是将连线接面包板改为了电脑画原理图，但在实际操作时却发现并不是这么简单，不仅每个模块都需要仿真来判断是否正确，而且在连成完整电路后下载运行时又总出现一些摸不着头脑的问题。这时，常常需要我们冷静思考，仔细检查，甚至需要一个模块一个模块地检索。

这次实验，我对数字电路的知识有了更深的了解。同时，也对VHDL语言有了更多的认识，对语法细节也关注了许多。不过由于时间不长，很多代码都是在CSDN上查找类似功能进行修改。

在实验过程中，我体会到了自顶向下的设计思想。在实验前，首先要将整个设计的思路理清，明白要实现什么功能，大体怎样布局。自顶向下，将一个复杂的电路拆分成几个模块，然后再开始动手设计，确保每一部分都能实现对应的功能。

最后，非常感谢老师和周围同学在本次实验中给我的帮助。这次实验，我体验到了设计数字逻辑电路的不易与乐趣。以后有机会还是要多多尝试做做电

路设计，将学到的知识应用于实际中。相信这次实验也会给今后的学习和生活带来帮助。

7.3 期望及要求

这次实验的结果总体还算圆满，但也存在一些小小的遗憾。由于时间的关系，想实现定时播放音乐的计划落空，只能改为开关控制音乐有无。原本的思路是当计数到某一时间时调用音乐模块（相当于开启开关）。

这次实验，学习了VHDL的编写方法，但部分还是借助了原理图来实现。希望有机会能对VHDL语言作更深入的研究与学习。

参考文献

- [1] 数字逻辑电路与系统设计.蒋立平.北京:电子工业出版社,2008.7
- [2] VHDL数字电路设计教程.佩德罗尼.北京:电子工业出版社, 2013.1
- [3] <https://blog.csdn.net/icurious/article/details/58580733>
- [4] <http://www.51hei.com/bbs/dpj-29177-1.html>
- [5] 钟其明.基于VHDL的任意进制计数器设计[J].科学咨询(科技·管理),2011(12):95.
- [6] <https://blog.csdn.net/wangshuilang11/article/details/2968874>
- [7] <https://blog.csdn.net/chenjieb520/article/details/7315827>