深圳大学实验报告

课程名称:	硬件描述语言及数字系统设计
实验项目名称:	实验三 七段数码管显示
))/ m-i	r ^
学院 <u>:</u>	医学院
专业:	生物医学工程
指导教师:	但果、董磊
报告人: 陈焕鑫	_ 学号 <u>: 2016222042</u> 班级: <u>生工2班</u>
实验时间:	2018.10.24
实验报告提交时间	J: <u>2018.11.2</u>

一、实验平台:

安装了 ISE 软件的 PC 计算机 硬件描述语言及数字系统设计实验平台 JTAG 下载线

二、实验目的:

当你完成整个项目之后, 你将会学会一下内容:

- 掌握七段数码管显示原理
- 能够用拨位开关控制数码管的显示

三、实验内容:

- •用 VHDL 代码描述一个七段数码管的译码电路,并编写测试激励进行测试
- ·编写引脚约束文件,用 ISE 软件生产.bit 文件,下载到硬件描述语言及数字系统设计实验平台进行板级验证

四、实验原理:

七段数码管实际上是由八个发光二极管 (a, b, c, d, e, f, g, dp) 组成,如图 4-1 所示,每个发光二极管对应数码管中的一段,不同发光二极管的组合形成不同的字符。如发光二极管除了 dp 段以外都发光,显示出来的字符就是"8",b,c两个二极管发光则形成"1"等。

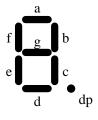


图 4-1

为了使发光二级管正确发光,需要有"七段数码管译码器","七段数码管译码器"的作用是把 0-F 字符的二进制信号转变成七个发光二极管的发光信号。

"七段数码管译码器"由四个输入端来控制,七个输出端 a, b, c, d, e, f, g 输出二极管的发光信号(小数点暂时不用)。

七段数码管分为共阳型和共阴型,如图 4-2 所示。硬件描述语言及数字系统设计实验平台上的七段数码管属于共阳型,因此译码输出为"1"表示发光二极管熄灭,译码输出为"0"则表示发光二极管点亮。七段数码管的各段按照从高位到低位的顺序依次是 dp g g e d c b a,例如,参照图 4-1,就可以得出"0"字符的译码结果为 11000000.用 4 位二进制数字表示 0-F,那么就可以得出七段数码管的译码表如表 4-1 所示

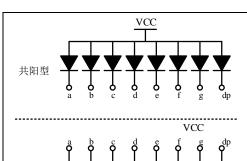


图 4-2

四位 输入	八位输出(dpgfedcba)	显示字符
0000	11000000 (0xC0)	0
0001	11111001 (0xF9)	1
0010	10100100 (0xA4)	2
0011	10110000 (0xB0)	3
0100	10011001 (0x99)	4
0101	10010010 (0x92)	5
0110	10000010 (0x82)	6
0111	11111000 (0xF8)	7
1000	10000000 (0x80)	8
1001	10010000 (0x90)	9
1010	10001000 (0x88)	A
1011	10000011 (0x83)	b
1100	11000110 (0xC6)	С
1101	10100001 (0xA1)	d
1110	10000110 (0x86)	Е
1111	10001110 (0x8E)	F

表 4-1 七段数码管译码表

```
五、实验方法步骤及 VHDL 代码:
 根据分频原理进行代码设计,设计一个产生 1Hz 的时钟信号。
 具体的 VHDL 代码如下所示:
--模块名称: clk gen 1hz
--摘要提示:
--当前版本: 1.0.0
--模块作者:
--完成日期: 20xx 年 xx 月 xx 日
--内容提要:
--需要注意:
--取代版本:
--模块作者:
--完成日期:
--修改内容:
--修改文件:
library ieee;
use ieee.std logic 1164.all;
use ieee.std logic arith.all;
use ieee.std_logic_unsigned.all;
----ENTITY DECLARATION-----
entity clk gen 1hz is
generic(
CNT MAX : integer := 50000000;
CNT HALF: integer := 25000000
);
port(
 rst_n_i : in std_logic;
 clk_i : in std_logic;
  clk_o : out std_logic
  );
end clk_gen_1hz;
-----ARCHITECTURE STRUCTURAL------
architecture rtl of clk_gen_lhz is
signal s_cnt : integer range 0 to CNT_MAX := 0;
```

```
begin
 process(clk_i, rst_n_i, s_cnt)
 begin
  if(rst n i = '0')then
   s_cnt <= 0;
   clk o <= '0';
  elsif rising_edge(clk_i) then
   if (s cnt = CNT MAX) then
    s cnt <= 0;
     clk o <= '0';
    elsif (s_cnt = CNT_HALF) then
     s_cnt <= s_cnt + 1;
     clk o <= '1';
    else
     s cnt <= s cnt + 1;
    end if;
  end if;
 end process;
end rtl;
 秒钟计数器模块-sec counter,每接收到一次1Hz的上升沿信号,秒钟计数器就加1,
直到计数达到 59 时,再接收到 1Hz 信号计数器将清零,同时分钟进位端输出"1"
--模块名称: sec_counter
--摘要提示:
--当前版本: 1.0.0
--模块作者:
--完成日期: 20xx 年 xx 月 xx 日
--内容提要:
--需要注意:
--取代版本:
--模块作者:
--完成日期:
--修改内容:
--修改文件:
library IEEE;
use IEEE.STD LOGIC 1164.ALL;
use IEEE.STD LOGIC ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
entity sec_counter is
   Port (
      clk i : in STD LOGIC;
     rst n i : in STD LOGIC;
      sec cnt o : out STD LOGIC VECTOR (5 downto 0);
      min_carry_o : out STD_LOGIC
     );
end sec counter;
architecture rtl of sec counter is
 signal s carry : std logic := '0';
 signal s sec : std logic vector(5 downto 0) := "0000000";
begin
 process(clk i, rst n i)
 begin
   if(rst n i = '0')then
    s carry <= '0';
    s sec <= "000000";
   elsif rising_edge(clk i)then
     if(s sec = "111011") then
     s sec <= "000000";
     s carry <= '1';
    else
      s sec <= s sec + "000001";
     s carry <= '0';
    end if;
   end if;
 end process;
 sec cnt o <= s sec;
 min_carry_o <= s_carry;</pre>
end rtl;
 分钟计数器模块-min_counter,每接收到一次分钟进位输出的上升沿信号,分钟计数器
就加1,直到计数达到59时,再接收到分钟进位信号计数器将清零,同时时钟进位端输
--模块名称: min counter
--摘要提示:
--当前版本: 1.0.0
--模块作者:
--完成日期: 20xx 年 xx 月 xx 日
--内容提要:
```

```
--需要注意:
--取代版本:
--模块作者:
--完成日期:
--修改内容:
--修改文件:
library IEEE;
use IEEE.STD LOGIC 1164.ALL;
use IEEE.STD LOGIC ARITH.ALL;
use IEEE.STD LOGIC UNSIGNED.ALL;
entity min counter is
   Port (
      clk_i : in STD_LOGIC;
      rst_n_i : in STD_LOGIC;
      min cnt o : out STD LOGIC VECTOR (5 downto 0);
      hour carry o : out STD LOGIC
    );
end min counter;
architecture rtl of min counter is
 signal s carry : std logic := '0';
 signal s min : std logic_vector(5 downto 0) := "000000";
begin
 process(clk i, rst n i)
 begin
   if(rst n i = '0')then
    s_carry <= '0';
    s min <= "000000";
   elsif rising edge(clk i)then
     if(s min = "111011")then
      s min <= "000000";
      s carry <= '1';
      s_min <= s_min + "000001";
      s carry <= '0';
    end if:
   end if;
 end process;
 min_cnt_o <= s_min;</pre>
 hour carry o <= s carry;
```

```
end rtl;
 时钟计数器模块-hour counter,每就收到一次时钟进位输出的上升沿信号,时钟计数
器就加1,直到计数达到23时,再接收到时钟进位信号计数器将清零
--模块名称: hour counter
--摘要提示:
--当前版本: 1.0.0
--模块作者:
--完成日期: 20xx 年 xx 月 xx 日
--内容提要:
--需要注意:
--取代版本:
--模块作者:
--完成日期:
--修改内容:
--修改文件:
library IEEE;
use IEEE.STD LOGIC 1164.ALL;
use IEEE.STD LOGIC ARITH.ALL;
use IEEE.STD LOGIC UNSIGNED.ALL;
entity hour counter is
  Port (
     clk i : in STD LOGIC;
     rst n i : in STD LOGIC;
      hour_cnt_o : out STD_LOGIC_VECTOR (5 downto 0)
    );
end hour_counter;
architecture rtl of hour counter is
 signal s hour : std logic vector(5 downto 0) := "000000";
 process(clk i, rst n i)
 begin
  if(rst n i = '0')then
    s hour <= "000000";
   elsif rising_edge(clk i) then
    if(s hour = "10111") then
      s_hour <= "000000";
    else
```

```
s_hour <= s_hour + "000001";</pre>
     end if;
   end if;
 end process;
 hour_cnt_o <= s_hour;</pre>
end rtl;
 求商模块-calc_mod,接收计数器输出的信号,转换为整型,求得除以10之后的商,
将以 BCD 码的格式其输出
--模块名称: calc mod
--摘要提示:
--当前版本: 1.0.0
--模块作者:
--完成日期: 20xx 年 xx 月 xx 日
--内容提要:
--需要注意:
--取代版本:
--模块作者:
--完成日期:
--修改内容:
--修改文件:
library IEEE;
use IEEE.STD LOGIC 1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD LOGIC UNSIGNED.ALL;
entity calc mod is
  Port (
      data i : in STD LOGIC VECTOR (5 downto 0);
      data_o : out STD_LOGIC_VECTOR (4 downto 0)
    );
end calc mod;
architecture rtl of calc mod is
 signal s_data : integer range 0 to 59;
 signal s mod : integer range 0 to 9 ;
begin
 s_data <= conv_integer(data_i);</pre>
 process(s data)
```

```
begin
   if (s_data >= 0 and s_data <= 9) then</pre>
    s mod <= 0;
   elsif (s data >= 10 and s data <= 19) then</pre>
    s mod <= 1;
   elsif (s_data >= 20 and s_data <= 29) then</pre>
    s mod <= 2;
   elsif (s_data >= 30 and s_data <= 39) then</pre>
    s mod <= 3;
   elsif (s_{data} >= 40 and s_{data} <= 49) then
    s mod <= 4;
   elsif (s data \geq= 50 and s data \leq= 59) then
    s mod <= 5;
   end if;
 end process;
   data o <= conv_std_logic_vector(s_mod,5);</pre>
end rtl;
 求模模块-calc_rem,接收计数器输出的信号,转换为整型,求得除以10之后的余数,
并将其以 BCD 码的格式输出
_____
--模块名称: calc rem
--摘要提示:
--当前版本: 1.0.0
--模块作者:
--完成日期: 20xx 年 xx 月 xx 日
--内容提要:
--需要注意:
--取代版本:
--模块作者:
--完成日期:
--修改内容:
--修改文件:
library IEEE;
use IEEE.STD LOGIC 1164.ALL;
use IEEE.STD LOGIC ARITH.ALL;
use IEEE.STD LOGIC UNSIGNED.ALL;
entity calc_rem is
   Port (
```

```
data i : in STD LOGIC VECTOR (5 downto 0);
      data o : out STD LOGIC VECTOR (4 downto 0)
     );
end calc rem;
architecture rtl of calc_rem is
 signal s data : integer range 0 to 59;
 signal s rem : integer range 0 to 9;
begin
 s data <= conv integer(data i);</pre>
 process(data i)
 begin
   if (s data >= 0 and s data <= 9) then</pre>
    s rem <= s data;
   elsif (s_data >= 10 and s_data <= 19) then</pre>
    s rem <= s data - 10;
   elsif (s data >= 20 and s data <= 29) then</pre>
     s rem <= s data - 20;
   elsif (s data >= 30 and s data <= 39) then</pre>
    s_rem <= s_data - 30;
   elsif (s data >= 40 and s data <= 49) then</pre>
    s rem <= s data - 40;
   elsif (s data >= 50 and s data <= 59) then</pre>
    s_rem <= s_data - 50;
   end if;
 end process;
 data o <= conv std logic vector(s rem,5);</pre>
end rtl;
 因为实验箱上有 8 个数码管,为满足整个数码管在 50Hz 的闪烁频率以内,每个数码
管的切换频率为 400Hz。根据分频原理进行代码设计,设计一个 400Hz 的时钟信号发生
器,用于七段数码管的扫描。因为时钟晶振的频率是 50MHz,因为产生一个 400Hz 的信
号需要计数 125000 次。
--模块名称: clk_gen_400hz
--摘要提示:
--当前版本: 1.0.0
--模块作者:
--完成日期: 20xx 年 xx 月 xx 日
--内容提要:
--需要注意:
```

```
--取代版本:
--模块作者:
--完成日期:
--修改内容:
--修改文件:
library IEEE;
use IEEE.STD LOGIC 1164.ALL;
use ieee.std logic arith.all;
use ieee.std_logic_unsigned.all;
entity clk gen 400hz is
  generic(
  CNT MAX : integer := 125000;
  CNT HALF : integer := 62500
  );
  Port (
  rst n i : in STD LOGIC;
   clk_i : in STD_LOGIC;
    clk o : out STD LOGIC
  );
end clk gen 400hz;
architecture rtl of clk gen 400hz is
 signal s_cnt : integer range 0 to CNT_MAX := 0;
begin
 process(clk_i, rst_n_i, s_cnt)
 begin
  if(rst_n_i = '0')then
    s cnt <= 0;
    clk o <= '0';
   elsif rising_edge(clk i)then
    if(s cnt = CNT MAX) then
     s_cnt <= 0;
     clk o <= '0';
    elsif(s_cnt = CNT_HALF) then
      s cnt <= s cnt + 1;
      clk o <= '1';
     else
      s_cnt <= s_cnt + 1;
     end if;
   end if;
```

```
end process;
end rtl;
 数码管切换模块-shift,模块接收400Hz的信号,每一次上升沿时将输出的信号进行一
次移位。
--模块名称: shift
--摘要提示:
--当前版本: 1.0.0
--模块作者:
--完成日期: 20xx 年 xx 月 xx 日
--内容提要:
--需要注意:
--取代版本:
--模块作者:
--完成日期:
--修改内容:
--修改文件:
library IEEE;
use IEEE.STD LOGIC 1164.ALL;
entity shift is
  Port (
     clk_i : in STD_LOGIC;
     rst n i : in STD LOGIC;
      shift o : out STD LOGIC VECTOR (7 downto 0)
    );
end shift;
architecture rtl of shift is
 signal s shift : std logic vector(7 downto 0) := "01111111";
begin
 process(rst_n_i, clk_i, s_shift)
 begin
  if(rst_n_i = '0')then
    s shift <= "011111111";
   elsif rising_edge(clk i) then
    s shift <= s shift(6 downto 0) & s shift(7);
   end if;
 end process;
 shift o <= s shift;</pre>
```

```
end rtl;
 译码器-decode,将接收端接收到的 BCD 数字翻译成数码管显示各数字所需要的编码,
方便对数码管进行控制。
--模块名称: decode
--摘要提示:
--当前版本: 1.0.0
--模块作者:
--完成日期: 20xx 年 xx 月 xx 日
--内容提要:
--需要注意:
--取代版本:
--模块作者:
--完成日期:
--修改内容:
--修改文件:
library IEEE;
use IEEE.STD LOGIC 1164.ALL;
entity decode is
   Port (
   seg7 BCDcode i : in STD LOGIC VECTOR (4 downto 0);
   seg7 decode o : out STD LOGIC VECTOR (7 downto 0)
   );
end decode;
architecture rtl of decode is
begin
 process(seg7_BCDcode_i)
 begin
   case seg7 BCDcode i is
     when "00000" => seg7 decode o <= "11000000";</pre>
     when "00001" => seg7 decode o <= "11111001";</pre>
     when "00010" => seg7_decode_o <= "10100100";</pre>
     when "00011" => seg7 decode o <= "10110000";</pre>
     when "00100" => seg7 decode o <= "10011001";</pre>
     when "00101" => seg7 decode o <= "10010010";</pre>
     when "00110" => seg7 decode o <= "10000010";</pre>
     when "00111" => seg7_decode_o <= "11111000";</pre>
     when "01000" => seg7 decode o <= "10000000";</pre>
```

```
when "01001" => seg7 decode o <= "10010000";</pre>
    when others => seg7 decode o <= "11000000";</pre>
  end case;
 end process;
end rtl;
 七段数码管显示模块-Seg7 disp, 该模块控制 8 个数码管中每个数码管的显示, 由 shift
模块发送来的信号控制所需要点亮的数码管,即扫描,切换频率为 400Hz。其中第三个
和第六个数码管固定显示"-",第一和第二个数码管显示时钟,第四和第五个数码管显
示分钟,第七和第八个数码管显示秒钟,显示的数字都是实时从各计时模块处理后发上
来的数据
--模块名称: Seg7_disp
--摘要提示:
--当前版本: 1.0.0
--模块作者:
--完成日期: 20xx 年 xx 月 xx 日
--内容提要:
--需要注意:
--取代版本:
--模块作者:
--完成日期:
--修改内容:
--修改文件:
library IEEE;
use IEEE.STD LOGIC 1164.ALL;
entity Seg7 disp is
  Port (
  disp hour h i : in STD LOGIC VECTOR(7 downto 0);
  disp hour l i : in STD LOGIC VECTOR(7 downto 0);
  disp min h i : in STD LOGIC VECTOR(7 downto 0);
  disp min l i : in STD LOGIC VECTOR(7 downto 0);
  disp sec h i : in STD LOGIC VECTOR(7 downto 0);
  disp_sec_l_i : in STD_LOGIC_VECTOR(7 downto 0);
  seg7 code o :     out     STD LOGIC VECTOR(7 downto 0)
  );
end Seg7 disp;
architecture rtl of Seg7 disp is
```

```
begin
 process(shift i)
 begin
   case shift i is
    when "111111110" => seg7 code o <= disp sec l i;
    when "111111101" => seg7_code_o <= disp_sec_h_i;</pre>
    when "11111011" => seg7 code o <= "101111111";</pre>
    when "11110111" => seg7 code o <= disp min l i;</pre>
    when "11101111" => seg7 code o <= disp min h i;
    when "110111111" => seg7 code o <= "101111111";</pre>
    when "101111111" => seg7 code o <= disp hour l i;
    when "01111111" => seg7 code o <= disp hour h i;
    when others => seg7 code o <= "101111111";</pre>
   end case;
 end process;
end rtl;
 七段数码管总控制模块-Seg7 digital LED,这一部分将 clk gen 400hz 模块、shift 模块、
decode 模块、Seg7_disp 模块集成在一起,封装成一个总的七段数码管控制模块,同时提
供了时钟一些接口: 50mhz 信号接入口, 复位信号接入口, 时钟高位信号接入口, 时钟
低位信号接入口,分钟高位信号接入口,分钟低位信号接入口,秒钟高位信号接入口,
秒钟低位信号接入口,数码管扫描控制端,七段数码管显示控制端。将数码管封装成这
样的一个模块,使得对数码管的控制不需要考虑底层的内容,操控起来更加简单
--模块名称: Seg7 digital LED
--摘要提示:
--当前版本: 1.0.0
--模块作者:
--完成日期: 20xx 年 xx 月 xx 日
--内容提要:
--需要注意:
--取代版本:
--模块作者:
--完成日期:
--修改内容:
--修改文件:
library IEEE;
use IEEE.STD LOGIC 1164.ALL;
use IEEE.STD LOGIC ARITH.ALL;
use IEEE.STD LOGIC UNSIGNED.ALL;
entity Seg7 digital LED is
```

```
Port (
      clk 50mhz i : in STD LOGIC;
      rst n i : in STD LOGIC;
                   in STD LOGIC VECTOR (4 downto 0);
     hour h i :
     hour l i :
                    in STD LOGIC VECTOR (4 downto 0);
      min_h_i :
                    in STD_LOGIC_VECTOR (4 downto 0);
                    in STD_LOGIC_VECTOR (4 downto 0);
      min l i :
      sec h i :
                    in STD LOGIC VECTOR (4 downto 0);
     sec l i :
                    in STD LOGIC VECTOR (4 downto 0);
      seg7 sel o : out STD LOGIC VECTOR (7 downto 0);
      seg7 decode o : out STD LOGIC VECTOR (7 downto 0)
    );
end Seg7 digital LED;
architecture rtl of Seg7 digital LED is
 component clk gen 400hz is
  port(
    rst n i : in STD LOGIC;
   clk i : in STD LOGIC;
    clk o : out STD LOGIC
   );
 end component;
 component shift is
  port(
    clk i : in STD LOGIC;
   rst n i : in STD LOGIC;
    shift o : out STD LOGIC VECTOR (7 downto 0)
   );
 end component;
 component decode is
   port(
    seg7 BCDcode i : in STD LOGIC VECTOR (4 downto 0);
    seg7 decode o : out STD LOGIC VECTOR (7 downto 0)
   );
 end component;
 component Seg7 disp is
  port(
    shift i :
              in STD LOGIC VECTOR(7 downto 0);
    disp hour h i : in STD LOGIC VECTOR(7 downto 0);
    disp_hour_l_i : in STD_LOGIC_VECTOR(7 downto 0);
    disp min h i : in STD LOGIC VECTOR(7 downto 0);
```

```
disp min l i : in STD LOGIC VECTOR(7 downto 0);
     disp sec h i : in STD LOGIC VECTOR(7 downto 0);
     disp sec l i : in STD LOGIC VECTOR(7 downto 0);
     seg7 code o :    out    STD LOGIC VECTOR(7 downto 0)
   );
  end component;
 signal s 400hz : std logic;
 signal s shift : std logic vector(7 downto 0);
 signal s hour h : std logic vector(7 downto 0);
 signal s hour 1 : std logic vector(7 downto 0);
 signal s min h : std logic vector(7 downto 0);
 signal s min 1 : std logic vector(7 downto 0);
 signal s sec h : std logic vector(7 downto 0);
 signal s sec 1 : std logic vector(7 downto 0);
begin
 U1 : clk gen 400hz port map(
                clk i => clk 50mhz i,
                rst n i => rst n i,
                clk o => s 400hz
               );
 U2 : shift port map
                clk i \Rightarrow s 400hz,
                rst n i => rst n i,
                shift o => s shift
 U3 : decode port map(
                 seg7_BCDcode_i => hour_h_i,
                 seg7 decode o => s hour h
           );
 U4 : decode port map (
                 seg7 BCDcode i => hour l i,
                seg7 decode o => s hour l
           );
 U5 : decode port map(
                 seg7 BCDcode i => min h i,
                 seg7 decode o => s min h
           );
 U6 : decode port map(
                 seg7 BCDcode i => min l i,
```

```
seg7 decode o => s min l
           );
 U7 : decode port map (
                 seg7 BCDcode i => sec h i,
                seg7_decode_o => s_sec_h
           );
 U8 : decode port map (
                 seg7_BCDcode_i => sec_l_i,
                 seg7 decode o => s sec l
           );
 U9 : Seg7 disp port map
                 shift i => s_shift,
                 disp hour h i => s hour h,
                disp_hour_l_i => s_hour_l,
                disp min h i => s min h,
                disp min l i => s min l,
                disp_sec_h_i => s_sec_h,
                disp sec l i => s sec l,
                seg7_code_o => seg7_decode_o
 seg7 sel o <= s shift;</pre>
end rtl;
```

最后是 clock 模块,此模块综合了前面创建的所有的模块,形成一个电子钟。提供的接口为: 50mHz 时钟信号接口,复位信号接口,七段数码管扫描端还有七段数码管显示端。50mHz 的时钟源分别在 clk_gen_1hz 和 clk_gen_400hz 模块中产生 1Hz 和 400Hz 的信号,1Hz 输入到 sec_counter 中产生秒数,进而控制 min_counter 和 hour_counter 产生分钟和时钟。三个计时器产生的数字都输出到 calc_mod 和 calc_rem 中提取出高位和低位,将结果输入到对应的 Seg7_digital_LED 模块的管脚,Seg7_digital_LED 模块会在其内部做各种处理,最后输出结果到七段数码管上。

--模块名称: clock
--摘要提示:
--当前版本: 1.0.0
--模块作者:
--完成日期: 20xx 年 xx 月 xx 日
--内容提要:
--需要注意:
---取代版本:
--模块作者:

```
--完成日期:
--修改内容:
--修改文件:
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity clock is
  Port (
    clk_50mhz_i : in STD_LOGIC;
    rst n i : in STD LOGIC;
    seg7 sel o : out STD LOGIC VECTOR (7 downto 0);
    seg7 code o : out STD LOGIC VECTOR (7 downto 0)
   );
end clock;
architecture rtl of clock is
 component clk gen 1hz is
   port(
     rst_n_i : in STD_LOGIC;
     clk i : in STD LOGIC;
      clk_o : out STD LOGIC
    );
 end component;
 component sec_counter is
   port(
     clk i : in STD LOGIC;
     rst n i : in STD LOGIC;
     sec cnt o : out STD LOGIC VECTOR (5 downto 0);
     min_carry_o : out STD_LOGIC
    );
 end component;
 component min counter is
  port(
     clk i : in STD LOGIC;
      rst_n_i : in STD_LOGIC;
      min cnt o : out STD LOGIC VECTOR (5 downto 0);
      hour carry o : out STD LOGIC
    );
 end component;
 component hour counter is
```

```
port(
    clk i : in STD LOGIC;
    rst n i : in STD LOGIC;
    hour cnt o : out STD LOGIC VECTOR (5 downto 0)
   );
end component;
component calc mod is
 port(
    data i : in STD LOGIC VECTOR (5 downto 0);
    data o : out STD LOGIC VECTOR (4 downto 0)
   );
end component;
component calc rem is
 port(
    data i : in STD LOGIC VECTOR (5 downto 0);
    data o : out STD LOGIC VECTOR (4 downto 0)
   );
end component;
component Seg7_digital_LED is
 port(
    clk 50mhz i : in STD LOGIC;
    rst n i : in STD LOGIC;
    hour h i :
                   in STD LOGIC VECTOR (4 downto 0);
    hour l i :
                   in STD LOGIC VECTOR (4 downto 0);
    min h i :
                   in STD LOGIC VECTOR (4 downto 0);
    min l i :
                   in STD LOGIC VECTOR (4 downto 0);
    sec h i :
                   in STD LOGIC VECTOR (4 downto 0);
                   in STD LOGIC VECTOR (4 downto 0);
    sec l i :
    seg7 sel o : out STD LOGIC VECTOR (7 downto 0);
    seg7 decode o : out STD LOGIC VECTOR (7 downto 0)
   );
end component;
signal s 1hz : std logic;
signal s 1min : std logic;
signal s 1hour : std logic;
signal s calc sec : std logic vector(5 downto 0);
signal s calc min : std logic vector(5 downto 0);
signal s calc hour : std logic vector(5 downto 0);
signal s_sec_h : std_logic_vector(4 downto 0);
signal s sec l : std logic vector(4 downto 0);
```

```
signal s min h : std logic vector(4 downto 0);
 signal s min 1 : std logic vector(4 downto 0);
 signal s hour h : std logic vector(4 downto 0);
 signal s hour 1 : std logic vector(4 downto 0);
begin
 U1 : clk_gen_1hz port map(
                  rst n i => rst n i,
                  clk i \Rightarrow clk 50mhz i,
                  clk o => s 1hz
                 );
 U2 : sec counter port map (
                  clk i \Rightarrow s 1hz,
                  rst_n_i => rst_n_i,
                  sec cnt o => s calc sec,
                  min carry o => s 1min
 U3 : min_counter port map(
                  clk i => s 1min,
                  rst n i => rst n i,
                  min_cnt_o => s_calc_min,
                  hour carry o => s 1hour
                 );
 U4 : hour counter port map (
                  clk i => s 1hour,
                  rst n i => rst n i,
                  hour_cnt_o => s_calc_hour
                 );
 U5 : calc mod port map (
                  data_i => s_calc sec,
                  data_o => s_sec_h
                 );
 U6 : calc mod port map (
                  data_i => s_calc_min,
                  data o => s min h
 U7 : calc mod port map
                  data_i => s_calc_hour,
                  data o => s hour h
                 );
 U8 : calc rem port map
                 data i => s calc sec,
                  data o => s sec l
                 );
```

```
U9 : calc_rem port map
                  data_i => s_calc_min,
                  data_o => s_min_l
                );
 U10 : calc_rem port map (
                  data_i => s_calc_hour,
                  data_o => s_hour_l
                );
 U11 : Seg7_digital_LED port map(
                  clk_50mhz_i => clk_50mhz_i,
                  rst_n_i => rst_n_i,
                  hour_h_i => s_hour_h,
                  hour_l_i => s_hour_l,
                  min h i => s min h,
                  min_l_i => s_min_l,
                  sec_h_i => s_sec_h,
                 sec_l_i => s_sec_l,
                 seg7 sel o => seg7 sel o,
                  seg7 decode o => seg7 code o
                );
end rtl;
```

六、综合、仿真结果及分析:

1、RTL级电路综合:

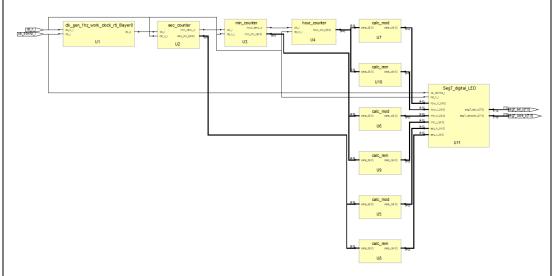
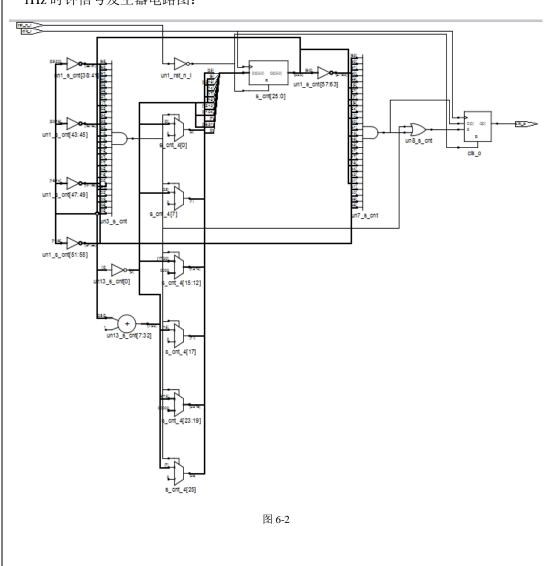
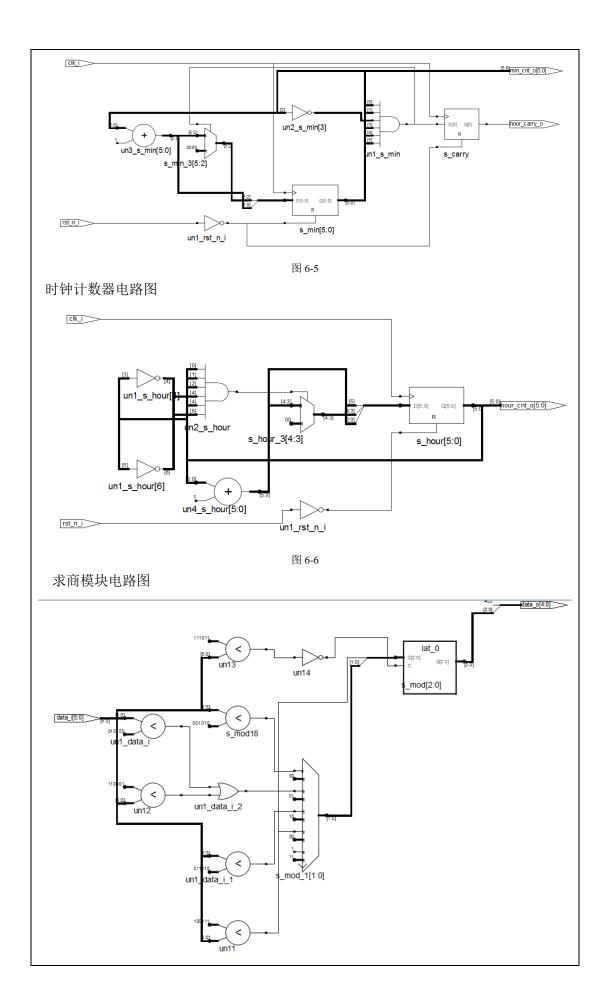


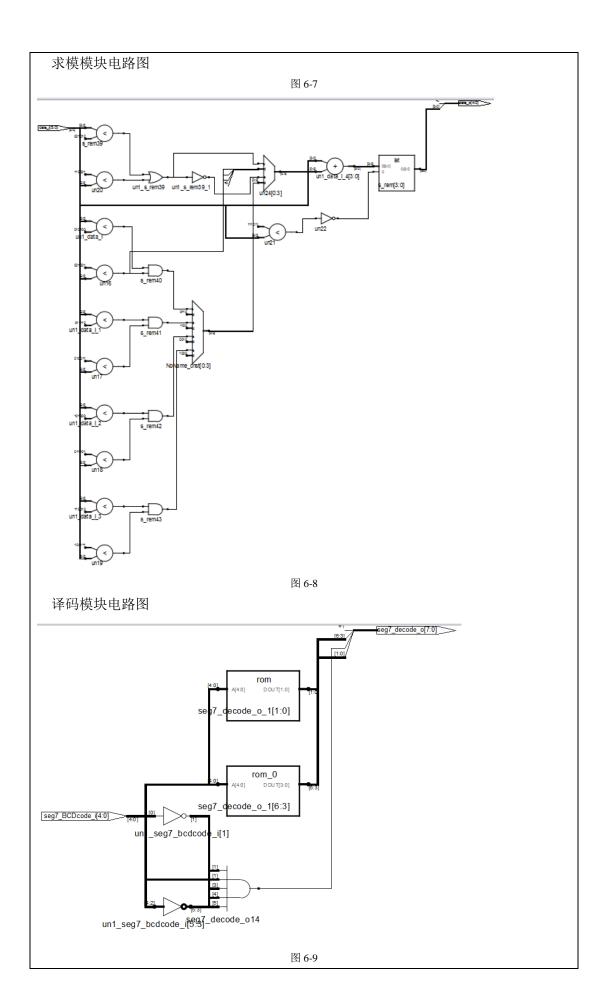
图 6-1

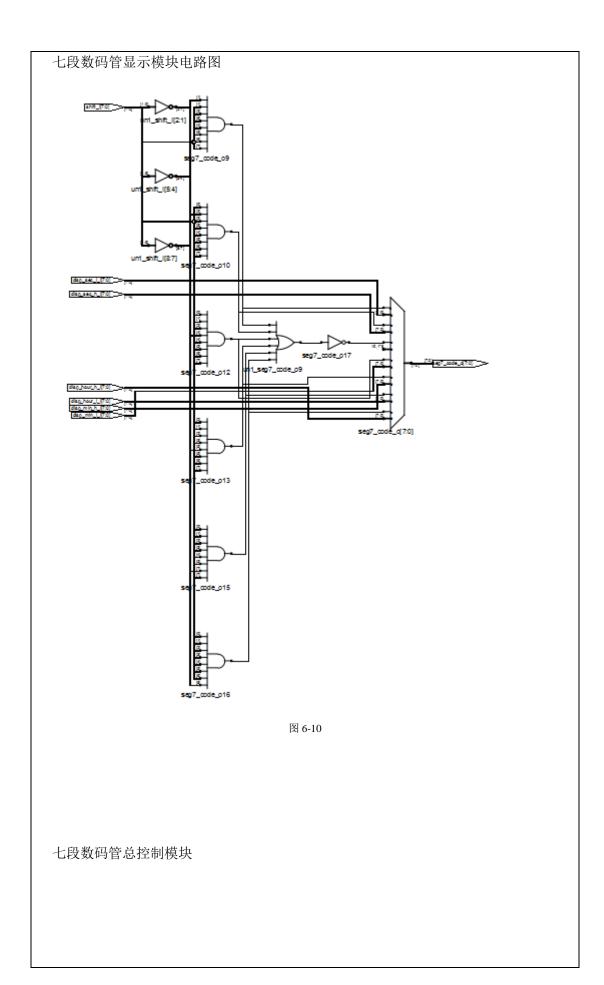
1Hz 时钟信号发生器电路图:

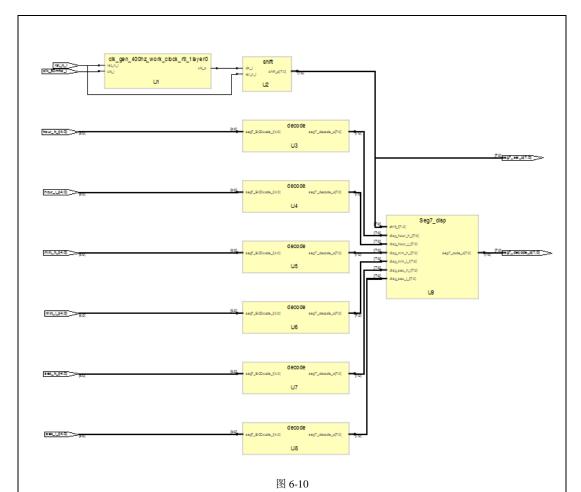


400Hz 时钟信号发生器电路图 s_cnt[16:0] un1_s_cnt[51:63] 13_s_cnt[0] un13_s_cnt[16:32] s_cnt_4[11] s_cnt_4[16:13] 图 6-3 秒钟计数器电路图: min_carry_o un2_s_sec[3] rst_n_i s_sec[5:0] un1_rst_n_i 图 6-4 分钟计数器电路图









2、电路仿真

测试 sec_counter 和 decoder 的时序:

编写测试激励 Test Bench 代码如下:

--模块名称: decoder tb

--摘要提示:译码器的测试激励文件

--当前版本: 1.0.0

--模块作者:

--完成日期: 20xx 年 xx 月 xx 日

--内容提要: --需要注意:

--取代版本:

--模块作者:

--完成日期:

--修改内容:

--修改文件:

LIBRARY ieee;

USE ieee.std_logic_1164.ALL;

```
ENTITY decode tb IS
END decode tb;
ARCHITECTURE behavior OF decode tb IS
   -- Component Declaration for the Unit Under Test (UUT)
   COMPONENT decode
   PORT (
       seg7_BCDcode_i : IN std_logic_vector(4 downto 0);
       seg7_decode_o : OUT std_logic_vector(7 downto 0)
      );
   END COMPONENT;
  --Inputs
  signal seg7_BCDcode_i : std_logic_vector(4 downto 0) := (others =>
'0');
 --Outputs
  signal seg7 decode o : std logic vector(7 downto 0);
  -- No clocks detected in port list. Replace <clock> below with
  -- appropriate port name
BEGIN
 -- Instantiate the Unit Under Test (UUT)
  uut: decode PORT MAP (
        seg7_BCDcode_i => seg7_BCDcode_i,
        seg7 decode o => seg7 decode o
      );
  -- Stimulus process
  stim proc: process
  begin
     -- hold reset state for 100 ns.
     wait for 100 ns;
   seg7 BCDcode i <= "10001";</pre>
     -- insert stimulus here
     wait;
  end process;
```

END;

仿真得到的结果如图 6-11 所示



```
输入 10001 (9), 输出的结果是 11000000, 数码管显示 9, 说明模块设计是正确的
--模块名称: sec counter
--摘要提示: 秒钟计数器的测试激励文件
--当前版本: 1.0.0
--模块作者:
--完成日期: 20xx 年 xx 月 xx 日
--内容提要:
--需要注意:
--取代版本:
--模块作者:
--完成日期:
--修改内容:
--修改文件:
LIBRARY ieee;
USE ieee.std logic_1164.ALL;
END sec counter tb;
ARCHITECTURE behavior OF sec counter tb IS
   -- Component Declaration for the Unit Under Test (UUT)
   COMPONENT sec counter
   PORT (
      clk_i : IN std_logic;
      rst n i : IN std logic;
      sec cnt o : OUT std logic vector(5 downto 0);
      min carry o : OUT std logic
      );
   END COMPONENT;
```

```
--Inputs
  signal clk i : std logic := '0';
  signal rst n i : std logic := '0';
 --Outputs
  signal sec_cnt_o : std_logic_vector(5 downto 0);
  signal min carry o : std logic;
  -- Clock period definitions
  constant clk i period : time := 20 ns;
BEGIN
 -- Instantiate the Unit Under Test (UUT)
  uut: sec counter PORT MAP (
        clk_i => clk_i,
       rst n i => rst n i,
        sec cnt o => sec cnt o,
        min_carry_o => min_carry_o
      );
  -- Clock process definitions
  clk i process :process
  begin
   clk i <= '0';
   wait for clk_i_period/2;
   clk i <= '1';
   wait for clk_i_period/2;
  end process;
  -- Stimulus process
  stim_proc: process
  begin
    -- hold reset state for 100 ns.
    wait for 100 ns;
    wait for clk_i_period*10;
     -- insert stimulus here
   rst n i <= '1';
     wait;
  end process;
```

END;

仿真结果如图 6-12 所示:

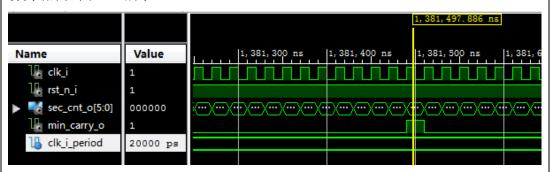


图 6-12

每 60s 会输出一个分钟进位,每个时钟上升沿都会改变 sec_cnt_o 的输出(图片被缩略未显示),证明模块设计是正确的。

七、FPGA 板级验证及结果分析

```
1、在工程中加入引脚约束文件 clock.ucf, 代码如下
```

```
Net "clk_50mhz i" LOC=P129;
               LOC=P85;
Net "rst_n_i"
Net "seg7 sel o<7>" LOC=P8;
Net "seg7 sel o<6>" LOC=P14;
Net "seg7 sel o<5>" LOC=P15;
Net "seg7 sel o<4>" LOC=P16;
Net "seg7 sel o<3>" LOC=P17;
Net "seg7 sel o<2>" LOC=P20;
Net "seg7 sel o<1>" LOC=P21;
Net "seg7_sel_o<0>" LOC=P22;
Net "seg7 code o<7>" LOC=P139;
Net "seg7 code o<6>" LOC=P140;
Net "seg7 code o<5>" LOC=P142;
Net "seg7 code o<4>" LOC=P2;
Net "seg7 code o<3>" LOC=P3;
Net "seg7 code o<2>" LOC=P4;
Net "seg7 code o<1>" LOC=P5;
Net "seg7 code o<0>" LOC=P7;
```

综合之后生成 bit 文件烧进实验箱中观察到如下图 7-1 和图 7-2 的现象:



图 7-1



图 7-2

经过验证,电子钟可以正确显示时间,而且各级的进位也没有问题。

实验总结: 通过这次实验的学习,我学会了利用 FPGA 芯片的模块化的方法实现系统,懂得了将大的问题细分成一个个小的模块,逐个实现它们,最后在组装在一起,来实现我们所需要的功能。掌握了分频的方法。成功编写出了电子钟模块,能够将各种模块例化并使用它们,即项层模块调用底层模块,而且学会了嵌套调用模块。	
大的问题细分成一个个小的模块,逐个实现它们,最后在组装在一起,来实现我们所需要的功能。掌握了分频的方法。成功编写出了电子钟模块,能够将各种模块例化并使用	实验总结:
要的功能。掌握了分频的方法。成功编写出了电子钟模块,能够将各种模块例化并使用	通过这次实验的学习,我学会了利用 FPGA 芯片的模块化的方法实现系统,懂得了将
	大的问题细分成一个个小的模块,逐个实现它们,最后在组装在一起,来实现我们所需
	■ ■要的功能。掌握了分频的方法。成功编写出了电子钟模块,能够将各种模块例化并使用

指导教师批阅意见:	
成绩评定:	
PARTI K.	
	指导教师签字:
备注:	年 月 日
- 1 招生由始孫日武山家孙昊 可担捉索	

- 注: 1、报告内的项目或内容设置,可根据实际情况加以调整和补充。
 - 2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。