华南师范大学

《可编程数字系统》
课程设计技术报告

设计课题：基于 FPGA 的智能交通灯设计

指导老师：

学生姓名：Hypo

学　　号：

院　　系：物理与电信工程学院

专　　业：通信工程

目录

第一章　设计原理

1.1 使用 VHDL 语言实现对 FPGA 器件的编程

FPGA（Field－Programmable Gate Array），即现场可编程门阵列，它是在 PAL、GAL、CPLD 等可编程器件的基础上进一步发展的产物。它是作为专用集成电路（ASIC）领域中的一种半定制电路而出现的，既解决了定制电路的不足，又克服了原有可编程器件门电路数有限的缺点。

对于目前的 FPGA 器件，可以使用硬件描述语言（Verilog 或 VHDL）完成电路设计，可以经过简单的综合与布局，快速的烧录至 FPGA 上进行测试，是现代 IC 设计验证的技术主流。这些可编辑元件可以被用来实现一些基本的逻辑门电路（比如 AND、OR、XOR、NOT）或者更复杂一些的组合功能比如解码器或数学方程式。

1.2 使用的设计工具

1.2.1 硬件部分

Cyclone ii 是一款 FPGA 器件，于 2004 年由 Altera 公司推出。Cyclone II FPGA 的成本比第一代 Cyclone 器件低 30%，逻辑容量大了三倍多，同时具有丰富的 I/O 引脚。实验板采用 EP2C5T144C8 芯片并配有 SDRAM 芯片，时钟频率为 50MHz。同时，实验板包含 4 个数码管以及 5 个按键，同时将丰富的 I/O 口用排针进行了引出。从系统资源以及运行速度的方面考虑，均足以完成智能交通灯的设计。

1.2.2 软件部分

Quartus II 是 Altera 公司的综合性 PLD/FPGA 开发软件，原理图、VHDL、VerilogHDL 以及 AHDL（Altera Hardware 支持 Description Language）等多种设计输入形式，内嵌自有的综合器以及仿真器，可以完成从设计输入到硬件配置的完整 PLD 设计流程。

VHDL 语言是一种用于电路设计的高级语言。它在 80 年代的后期出现。具有设计灵活、支持广泛、易于修改、独立于器件的设计或工艺等优秀的特性。十分适合于对 FPGA 器件进行编程。

1.3 软件框图

本系统包含一个顶层以及两个子器件（分频器以及数码管显示驱动）。分频器将系统时钟分为 1000Hz,1Hz 等待信号供顶层程序使用。顶层包含了整个系统关键性控制逻辑，在接收到分频器时钟信号以及按键信号后，由不同的进程进行处理，同时不同的进程间通

过标志位信号进行通信。处理完成的信息将传递给 LED 直接输出红绿灯信号或数码管驱动器件进行进一步的处理后输出给数码管进行显示。

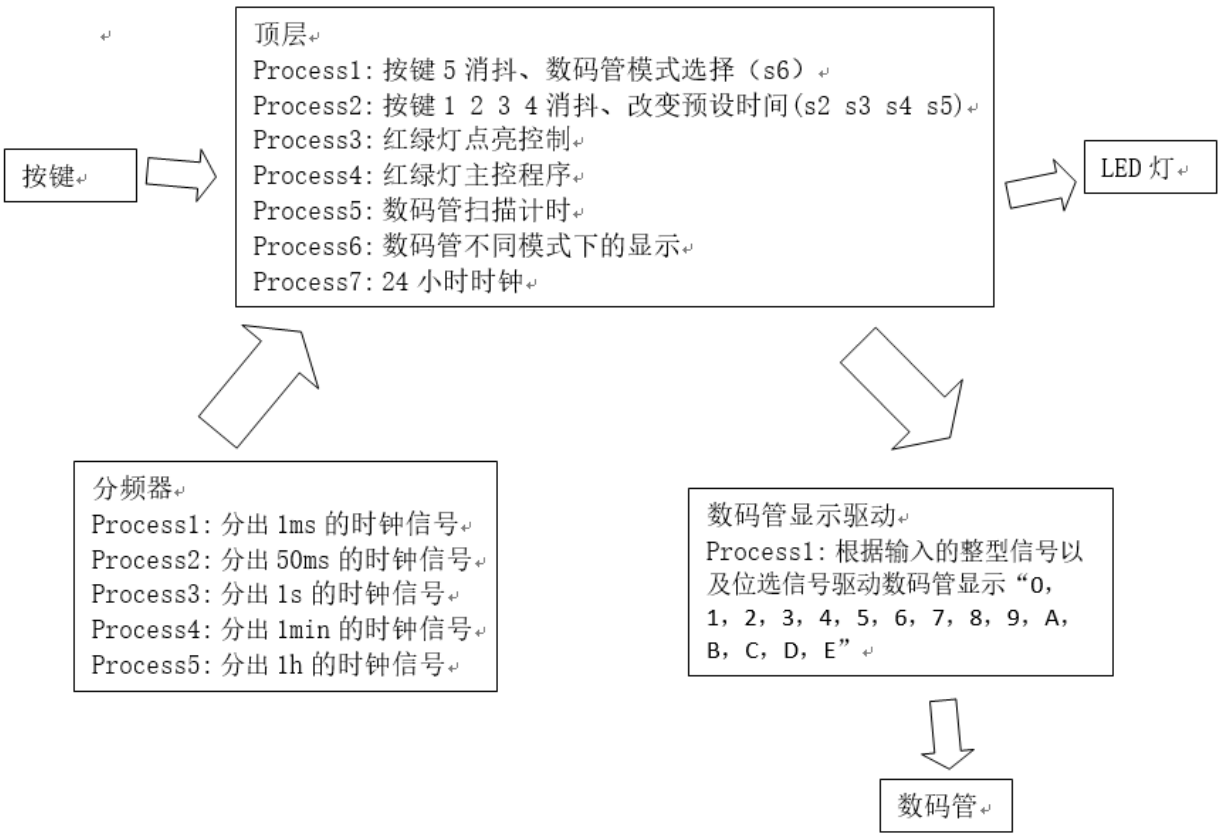

图 1.1

1.4 红绿灯拓展模块

为了达到更好的人机交互效果，本系统将红绿灯在电路板上通过不同的颜色的 LED 进行实物化。

模块原理图如下：

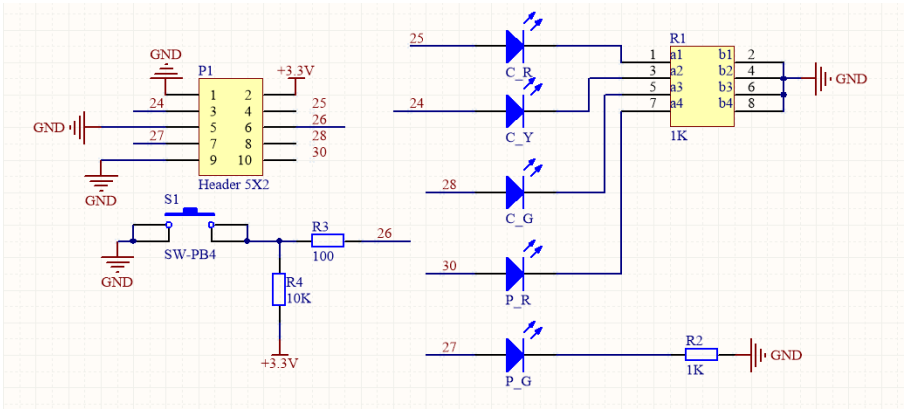

图 1.2

PCB 设计如下：

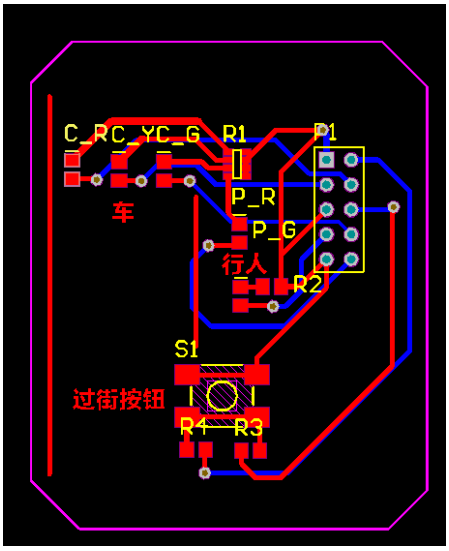

图 1.3

第二章　程序设计及分析

2.1 分频器器件的设计

本系统将分频器单独设置为一个器件，目的是便于对时间倍率进行修改以及方便调用。具体思路如下：（以 1ms 为周期的输出信号为例）

```
process(clk_in)   --1ms 系统时钟输入
begin
   if rising_edge(clk_in) then--当检测到时钟信号为上升沿时
      if(data_ms=25000) then --由计数器实现分频的功能，
                             --由50M/（25000*2）得到信号翻转周期。
                             --修改data_ms的判断值就可以实现不同的分频比
      data_ms<=0;
      Q_ms<=not Q_ms;      --对暂存状态里进行翻转
      else
      data_ms<=data_ms+1;
      end if;
   end if;
clk_out_ms<=Q_ms;          --将暂存里进行输出
end process;
```

图 2.1

2.2 数码管显示驱动器件的设计

实验板的数码管为共阳连接，含有 4 个位选信号以及 8 个段选信号，通过控制位选及段选可以实现数码管的显示。

该子器件的输入信号为 0-15 的整数，输出为显示 16 进制相应字符的段选信号。程序主体采用 case 语句，对输入信号进行选择并相应的输出。

```
process(int_in)
begin
   case int_in is
      when 0 => data_out <= "11000000"; -- 0
      when 1 => data_out <= "11111001"; -- 1
      when 2 => data_out <= "10100100"; -- 2
      when 3 => data_out <= "10110000"; -- 3
      when 4 => data_out <= "10011001"; -- 4
      when 5 => data_out <= "10010010"; -- 5
      when 6 => data_out <= "10000010"; -- 6
      when 7 => data_out <= "11111000"; -- 7
      when 8 => data_out <= "10000000"; -- 8
      when 9 => data_out <= "10010000"; -- 9
      when 10 => data_out <= "10001000"; -- A
      when 11 => data_out <= "10000011"; -- b
      when 12 => data_out <= "10100111"; -- c
      when 13 => data_out <= "10100001"; -- d
      when 14 => data_out <= "10000110"; -- E
      when 15 => data_out <= "10001110"; -- F
      when others => NULL;
   end case;
end process;
```

图 2.2

2.3 顶层器件设计

2.3.1 按键消抖

本系统对于按键采取了消抖的处理办法，可以有效的增加稳定性。消抖的实现原理基于计数器，当检测到有按键按下则开始计数，当计数达到设定值时给出按键按下的信号并执行相应功能，其中 key5 实现模式选择，其余按键实现对时间设定的更改。

```
process(sysclk,key5)      ------按键5消抖+数码管模式选择（s6）
variable COUNT1 :integer range 0 to 1000000;
begin
if key5='0' then
   if RISING_EDGE(sysclk) then
      if COUNT1<1000000 then
      COUNT1:=COUNT1+1;
      else COUNT1:=COUNT1;
      end if;
      if COUNT1=999999 then
      data_mode <= data_mode+1;
      else data_mode <= data_mode;
      end if;
   end if;
else COUNT1:=0;
END if;
if(data_mode = 4)then
data_mode <= 0;
end if;
end process ;
```

图 2.3

2.3.2 数码管扫描显示

实验板采用了四个共阳的数码管，若需要让每个数码管都显示不一样的内容，则需要对数码管进行扫描显示。本系统采用 1000Hz 的频率对数码管进行扫描显示，实际效果可以稳定的显示而不会出现闪烁的现象。

程序设计方面采用了两个 process，分别对数码管位选进行扫描以及输入段选信号

```
process(tmp_sig_ms,data_count_seg)   ---生成数码管位选扫描
begin
if rising_edge(tmp_sig_ms) then
data_count_seg <= data_count_seg+1;
   if(data_count_seg = 4) then
   data_count_seg <= 0;
   end if;
end if;
end process;
```

图 2.4

```
process(sysclk)    --数码管扫描各模式显示样式
begin
if(data_mode = 0)then-------------------
   if(data_count_seg = 0)then---位选扫描输入
   seg7com <="0111";--位选
   tmp_seg7data <= data_min_low;--段选
   elsif (data_count_seg = 1)then
   seg7com <="1011";
   tmp_seg7data <= data_min_high;
   elsif (data_count_seg = 2)then
   seg7com <="1101";
   tmp_seg7data <= data_h_low;
   else
   seg7com <="1110";
   tmp_seg7data <= data_h_high;
   end if;
```

图 2.5

2.3.3 24 小时时钟

为了实现白天与夜间的模式切换效果，需要加入 24 小时制的时钟。本系统对于时钟的实现依然基于计数器。具体流程如下：

对分频器输出的分钟信号进行检测，当检测到上升沿时执行计数，当达到该位最大值时执行进位与清零。

```
process(sysclk)   --24小时时钟计数
begin
if rising_edge(tmp_sig_min) then --输入信号为分钟，当分钟输入信号为上升沿时触发
data_min_low<=data_min_low+1;     --开始计数
   if(data_min_low = 9)then       --分钟低位达到最大值，执行进位
   data_min_low <=0;              --清零
   data_min_high<=data_min_high+1;
      if(data_min_high = 5)then   --满一小时进位给小时
      data_min_low <=0;           --清零
      data_min_high <=0;
      data_h_low <= data_h_low+1;
      data_h <= data_h+1;
         if(data_h_low = 9 )then  --小时低位满进位给小时高位
         data_min_low <=0;         --清零
         data_min_high <=0;
         data_h_low <=0;
         data_h_high <=data_h_high+1;
         end if;
      end if;
   end if;
end if;
if(data_h = 24)then               --满24小时，全部清零
data_min_low <=0;
data_min_high <=0;
data_h_low <=0;
data_h_high <=0;
data_h <= 0;
end if;
if(data_h >= data_time_nightin or data_h <data_time_nightout)then --白天黑夜判断
night <='1';
else
night <='0';
end if;
end process;
```

图 2.6

2.3.4 改变时间设定

本程序实现了对行人通行时间，汽车通行时间以及夜间时间段的设定。设定方式为由按键实现加减。主要代码已经内嵌于按键消抖程序中。

```
if(key1='0')then
   if(data_mode = 2)then
   data_time_cp <= data_time_cp+1;
   elsif(data_mode =3)then
   data_time_nightout <= data_time_nightout+1;
   end if;
end if;
if(key2='0')then
   if(data_mode = 2)then
   data_time_cp <= data_time_cp-1;
   elsif(data_mode =3)then
   data_time_nightout <= data_time_nightout-1;
   end if;
end if;
if(key3='0')then
   if(data_mode = 2)then
   data_time_pp <= data_time_pp+1;
   elsif(data_mode =3)then
   data_time_nightin <= data_time_nightin+1;
   end if;
end if;
 if(key4='0')then
   if(data_mode = 2)then
   data_time_pp <= data_time_pp-1;
   elsif(data_mode =3)then
   data_time_nightin <= data_time_nightin-1;
   end if;
end if;
```

图 2.7

2.3.5 交通灯主控程序

本程序实现了对白天夜间模式的判断以及不同模式下对红绿灯的控制。程序框图如下：

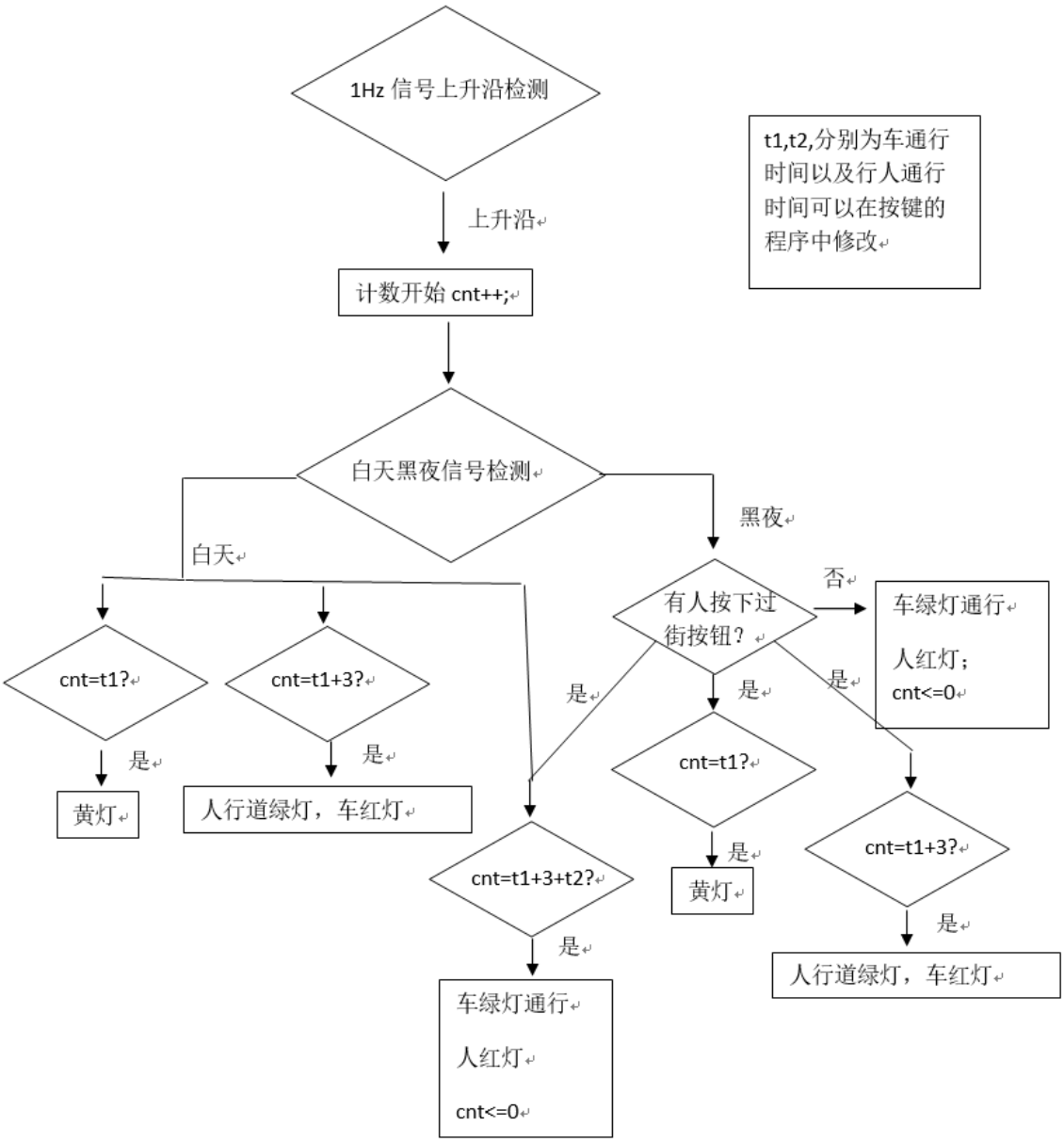

图 2.8

具体程序代码详见附录。

第三章　使用说明及实验结果

3.1 使用说明

本系统基本实现了题目要求的所有功能。具备了白天、夜间模式的功能及其自动切换及其时间点设置，行人、汽车通行时间设置，24 小时电子钟，通行时间倒数等功能。同时为了实现更好的人机交互以及获得更明显的实验效果，接入了专用的红绿灯模块并将电子时钟加快了 1000 倍，红绿灯计时器加快了 10 倍。

具体使用方法如下：

（1）初始值：系统上电后时钟初始值为 0 点。夜间时间初始值为 22：00-8：00，行人通行时间，车辆通行时间初始值分别为 15s,30s.

（2）系统模式选择：数码管共用 4 种显示模式，可以通过 key5(s6)对模式进行切换（上电后将默认进入模式 0）。同时，在不同模式下按键也具有不同的功能。

模式 0：电子时钟显示模式（为上电后的默认显示模式），左边两个数码管显示小时，右边两个数码管显示分钟。四个数码管共同构成当天系统时间（是白天、夜间的切换的基准时间）。

模式 1：通行剩余时间显示。右边两个数码管将显示红绿灯当前状态还将持续的剩余时间。当进入夜间模式且无行人通过时，数码管将显示----。

模式 2：行人，车辆通行时间设置模式。此模式下左边两个数码管显示车辆通行时间，右边两个数码管为行人通行时间。按下 S2 将使车辆通行时间+1s,按下 S3 将使车辆通行时间+1s。同样，S4 与 S5 将分别控制行人通行时间的改变。

模式 3：夜间时间点设置模式。此模式下左边两个数码管显示进入白天的时间点，右边两个数码管显示进入夜间的时间点（单位均为小时）。同样，此时 S2、S3 将控制进入白天时间点，S4、S6 将控制进入夜间的时间点。

（3）白天模式:当系统时钟为设定的白天区间内，交通灯将执行白天模式，此时人行道和车行道的红绿灯自动切换，人行道和车行道的通行时间分别为 30 秒和 15 秒。同时车行道的绿灯到红灯的切换有 3 秒过渡时间（亮黄灯）。

（4）夜间模式：当系统时钟为设定的夜间区间内，交通灯将执行白天模式，无行人过马路时，车行道持续亮绿灯让车通行，直到有行人按过街按钮，才执行人、车通行的切换。行人按下按钮后延时 30 秒，切换到人行道通行 15 秒，然后切换到车通行。此后，若

无人按过街按钮，则保持车通行状态，若继续有人按下过街按钮，则执行人、车分别 15 秒和 30 秒的轮换通行。车行道的绿灯到红灯的切换有 3 秒过渡时间（亮黄灯），人行道则只有红、绿灯，且无过渡时间。

3.2 实验结果

具体实验结果可以扫描以下二维码观看。

图 3.1

部分视频截图如下：

图 3.2


为方便演示，时钟速度加快1000倍
红绿灯切换速度加快10倍
按下过街按钮后，执行过街程序

图 3.3


为方便演示，时钟速度加快1000倍
红绿灯切换速度加快10倍
按下显示模式切换按键，显示人
与车的通行剩余时间，
----代表夜间模式下车一直通行

图 3.4

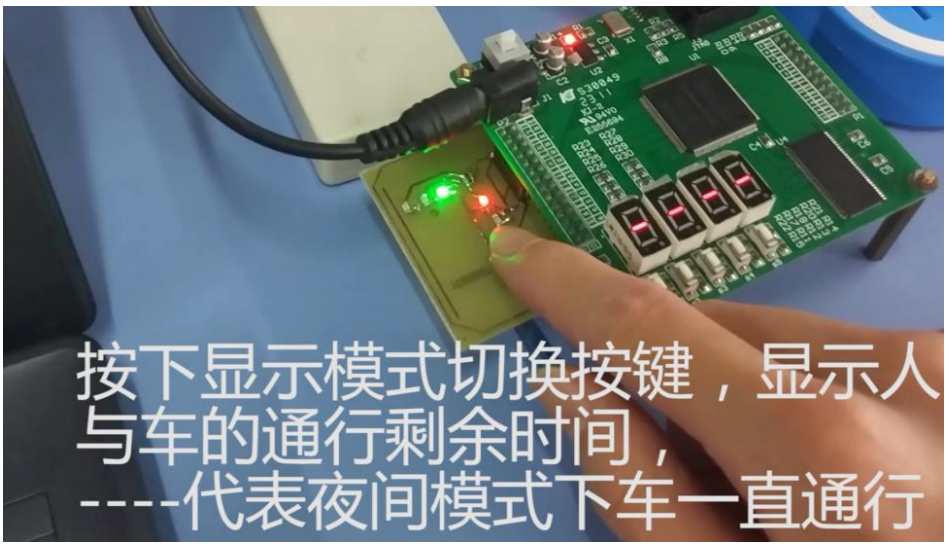
按下显示模式切换按键，显示人
与车的通行剩余时间，
----代表夜间模式下车一直通行

图 3.5

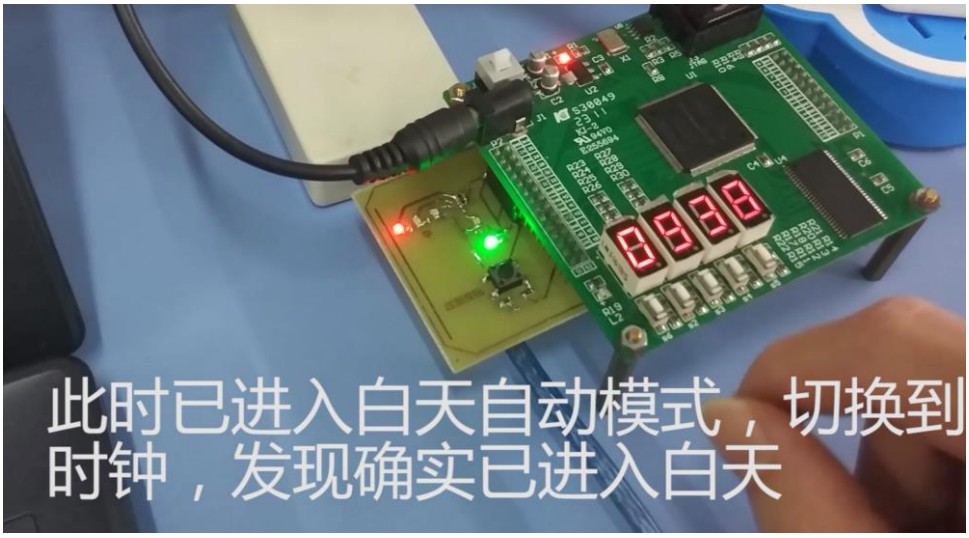

图 3.6

图 3.7

图 3.8

第四章　结论与体会

通过借助网络工具以及对书本的学习，本次成功的完成了智能交通灯的设计，并实现了题目要求的所有功能。但是过程中却不是很顺利，主要体现在按键的使用以及不同 process 间通过信号量进行通信等，特别是当使用较多信号进行参数传递时经常忽略信号量的值不能在两个 process 中同时进行修改，而导致不能成功通过编译。另外对于编程的整体思路也与之前接触过的单片机大为不同，单片机的 MCU 只支持单个进程运行，需要加入各种中断处理后才可以做到宏观上的多进程并行运行，而 FPGA 则支持数个进程同时运行，大大的提高了代码运行速度的同时却导致了整体设计构思发生了较大的转变。另外，对于程序模块化的设计也有了较为初步的理解，通过模块化的方法可以使代码的可读性、可移值性均大大的提高，同时还十分便于修改，甚至由于可以实现模块化调用而大大缩减了代码量。

总结：FPGA 很强大，我学会了很多。

附录　程序源码

```
--@By Hypo 20170520  交通灯主程序
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY test IS
    PORT(
        key1,key2,key3,key4,key5,key6: IN STD_LOGIC;--按键输入
        sysclk: IN STD_LOGIC;--时钟输入
         led_c_r,led_c_y,led_c_g,led_p_r,led_p_g: OUT STD_LOGIC;--led 输出
         seg7data: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);--数码管位选
         seg7com: OUT STD_LOGIC_VECTOR(3 DOWNTO 0)     --数码管段选
         );
END test;

ARCHITECTURE main OF test IS

    COMPONENT seg7led IS    ----------------------数码管
        PORT (
            int_in:            IN INTEGER    RANGE 0 TO 16;
            data_out: OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
        );
    END COMPONENT seg7led;

    COMPONENT clkdiv IS---------------------------------分频器
        PORT( clk_in:IN STD_LOGIC;         --时钟信号输入
                clk_out_ms:OUT STD_LOGIC;    --毫秒信号输出
             clk_out_s:OUT STD_LOGIC;    --秒信号输出
                clk_out_min:OUT STD_LOGIC;    --分钟信号输出
             clk_out_h:OUT STD_LOGIC;    --时钟信号输出
                clk_out_50ms : OUT STD_LOGIC
            );
    END COMPONENT clkdiv;

    ------------------------------------------------数码管扫描
    SIGNAL tmp_seg7data :integer range 0 to 16;
    SIGNAL data_count_seg:integer range 0 to 10;

    ------------------------------------分频器输出
    SIGNAL  tmp_sig_s,tmp_sig_ms,tmp_sig_min,tmp_sig_h,tmp_sig_50ms  :STD_LOGIC;--分频器输出时钟信号

    ------------------------------------------------24 小时时钟
```

```
    SIGNAL data_min_low:integer range 0 to 20;
    SIGNAL data_min_high:integer range 0 to 10;
    SIGNAL data_min:integer range 0 to 100;
    SIGNAL data_h_low:integer range 0 to 20;
    SIGNAL data_h_high:integer range 0 to 10;
    SIGNAL data_h:integer range 0 to 100;

    ------------------------------------------------数码管显示模式切换
    SIGNAL data_mode: integer range 0 to 10:=0;
    SIGNAL data_count_key:integer range 0 to 100;

    ------------------------------------------------交通灯主控程序
    SIGNAL led_p :integer range 0 to 3;--车道灯
    SIGNAL led_c :integer range 0 to 3:=1;--人行道灯
    SIGNAL key_flag :integer range 0 to 1:=0;
    SIGNAL data_time_count:integer range 0 to 1000;--时间计数
    SIGNAL data_mode_led: integer range 0 to 10:=0;--红绿灯剩余时间显示

    ------------------------------------------------------------夜间模式初始化设置
    SIGNAL data_time_nightin: integer range 0 to 100:=22;
    SIGNAL data_time_nightout: integer range 0 to 100:=8;
    SIGNAL night :STD_LOGIC:='0';

    ------------------------------------------------------------红绿灯时间设置
    SIGNAL data_time_cp : integer range 0 to 100:=30;--车通行 30 秒
    SIGNAL data_time_pp : integer range 0 to 100:=15;--人通行 15 秒

BEGIN

    div: clkdiv PORT MAP (      ----------分频器
                clk_in      => sysclk,
                clk_out_s => tmp_sig_s,
                clk_out_ms => tmp_sig_ms,
                clk_out_min => tmp_sig_min,
                clk_out_h => tmp_sig_h,
                clk_out_50ms => tmp_sig_50ms);

    dis: seg7led PORT MAP (     ----------数码管
                int_in      => tmp_seg7data,
                data_out => seg7data
            );

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

```
process(sysclk,key5)        ------按键 5 消抖+数码管模式选择（s6）
variable COUNT1 :integer range 0 to 1000000;
begin
if key5='0' then
      if RISING_EDGE(sysclk) then
            if COUNT1<1000000 then
            COUNT1:=COUNT1+1;
            else COUNT1:=COUNT1;
            end if;
            if COUNT1=999999 then
            data_mode <= data_mode+1;
            else data_mode <= data_mode;
            end if;
      end if;
else COUNT1:=0;
END if;
if(data_mode = 4)then
data_mode <= 0;
end if;
end process ;

-----------------------------------------------------------------------

process(sysclk)   --------按键 1 2 3 4 消抖+改变设定时间(s2 s3 s4 s5)
variable COUNT1 :integer range 0 to 1000000;
begin
if key1='0' or key2='0' or key3='0' or key4='0' then
      if rising_edge(sysclk) then
            if COUNT1<1000000 then
            COUNT1:=COUNT1+1;
            else COUNT1:=COUNT1;
            end if;
            if COUNT1=999999 then
                if(key1='0')then
                      if(data_mode = 2)then
                      data_time_cp <= data_time_cp+1;
                      elsif(data_mode =3)then
                      data_time_nightout <= data_time_nightout+1;
                      end if;
                 end if;
                if(key2='0')then
                      if(data_mode = 2)then
                      data_time_cp <= data_time_cp-1;
```

```
                    elsif(data_mode =3)then
                    data_time_nightout <= data_time_nightout-1;
                    end if;
                end if;
                if(key3='0')then
                    if(data_mode = 2)then
                    data_time_pp <= data_time_pp+1;
                    elsif(data_mode =3)then
                    data_time_nightin <= data_time_nightin+1;
                    end if;
                end if;
                  if(key4='0')then
                    if(data_mode = 2)then
                    data_time_pp <= data_time_pp-1;
                    elsif(data_mode =3)then
                    data_time_nightin <= data_time_nightin-1;
                    end if;
                end if;
            end if;
        end if;
    else COUNT1:=0;
    end if;
    end process;

-----------------------------------------------------------------------

    process(sysclk)                          ---红绿灯点亮定义
    begin
    if(led_c = 0)then
    led_c_r <= '1';
    led_c_g <='0';
    led_c_y <='0';
    elsif(led_c = 1)then
    led_c_r <='0';
    led_c_g <='1';
    led_c_y <='0';
    else
    led_c_r <='0';
    led_c_g <='0';
    led_c_y <='1';
    end if;
    if(led_p = 0)then
    led_p_r <= '1';
    led_p_g <= '0';
```

```
        else
        led_p_r <= '0';
        led_p_g <= '1';
        end if;
        end process;

-------------------------------------------------------------------------

        process(sysclk)                                 ---红绿灯主控程序
        begin
        if rising_edge(tmp_sig_s)then
        data_time_count<=data_time_count+1;
            if(night='1')then-----------黑夜模式
                if(key_flag=1)then
                    if(data_time_count=data_time_cp)then
                    led_p <=0;      --进入黄灯
                    led_c <=2;
                    data_mode_led <=3;
                    end if;
                    if(data_time_count=(data_time_cp+3))then
                    led_p <=1;      --行人通行
                    led_c <=0;
                    data_mode_led <=2;
                    end if;
                    if(data_time_count=(data_time_cp+3+data_time_pp))then
                    led_p <=0;      --车通行
                    led_c <=1;
                    key_flag <=0;
                    data_time_count <=0;
                    data_mode_led <= 0;
                    end if;
                else
                led_p <=0;
                led_c <=1;
                data_mode_led <= 0;
                end if;
            end if;

            if(night='0')then-------------白天模式
                if(data_time_count =data_time_cp)then
                led_p <=0;        --进入黄灯
                led_c <=2;
                data_mode_led <=3;
                end if;
```

```
                if(data_time_count = (data_time_cp+3))then
                led_p <=1;      --行人通行
                led_c <=0;
                data_mode_led <=2;
                end if;
                if(data_time_count >=(data_time_cp+3+data_time_pp) )then
                led_p <=0;      --车通行
                led_c <=1;
                data_time_count <=0;
                data_mode_led <=1;
                end if;
            end if;
        end if;

        if(key6='0' and key_flag = 0)then
        data_time_count <= 0;
        data_mode_led <=1;
        end if;

        if(key6='0')then
        key_flag <=1;
        end if;

        end process;

-------------------------------------------------------------------------

      process(tmp_sig_ms,data_count_seg)    ---生成数码管位选扫描
        begin
        if rising_edge(tmp_sig_ms) then
        data_count_seg <= data_count_seg+1;
            if(data_count_seg = 4) then
            data_count_seg <= 0;
            end if;
        end if;
        end process;

-----------------------------------------------------------------------------

        process(sysclk)       --数码管扫描各模式显示样式
        begin
        if(data_mode = 0)then----------------                                    ----模式 0，时钟显示
            if(data_count_seg = 0)then---位选扫描输入
            seg7com <="0111";--位选
```

```
        tmp_seg7data <= data_min_low;--段选
        elsif (data_count_seg = 1)then
        seg7com <="1011";
        tmp_seg7data <= data_min_high;
        elsif (data_count_seg = 2)then
        seg7com <="1101";
        tmp_seg7data <= data_h_low;
        else
        seg7com <="1110";
        tmp_seg7data <= data_h_high;
        end if;
    elsif(data_mode = 1)then----------------------------------------模式 1，红绿灯剩余时间显示
        if(data_mode_led = 0)then--------夜间模式一直车走，显示----
            if(data_count_seg = 0)then
            seg7com <="0111";
            tmp_seg7data <= 16;
            elsif (data_count_seg = 1)then
            seg7com <="1011";
            tmp_seg7data <= 16;
            elsif (data_count_seg = 2)then
            seg7com <="1101";
            tmp_seg7data <= 16;
            else
            seg7com <="1110";
            tmp_seg7data <= 16;
            end if;
        elsif(data_mode_led = 1)then----车通行剩余时间
            if(data_count_seg = 0)then
            seg7com <="0111";
            tmp_seg7data <= (data_time_cp - data_time_count)rem 10;
            elsif (data_count_seg = 1)then
            seg7com <="1011";
            tmp_seg7data <= (data_time_cp - data_time_count)/ 10;
            elsif (data_count_seg = 2)then
            seg7com <="1101";
            tmp_seg7data <= 16;
            else
            seg7com <="1110";
            tmp_seg7data <= 16;
            end if;
        elsif(data_mode_led = 2)then----人通行剩余时间
            if(data_count_seg = 0)then
            seg7com <="0111";
            tmp_seg7data <= ((data_time_cp+3+data_time_pp) - data_time_count)rem 10;
```

```
            elsif (data_count_seg = 1)then
            seg7com <="1011";
            tmp_seg7data <= ((data_time_cp+3+data_time_pp) - data_time_count)/ 10;
            elsif (data_count_seg = 2)then
            seg7com <="1101";
            tmp_seg7data <= 16;
            else
            seg7com <="1110";
            tmp_seg7data <= 16;
            end if;
        else                                    ----黄灯则显示 00
            if(data_count_seg = 0)then
            seg7com <="0111";
            tmp_seg7data <= 0;
            elsif (data_count_seg = 1)then
            seg7com <="1011";
            tmp_seg7data <= 0;
            elsif (data_count_seg = 2)then
            seg7com <="1101";
            tmp_seg7data <= 16;
            else
            seg7com <="1110";
            tmp_seg7data <= 16;
            end if;
        end if;
elsif(data_mode = 2)then-----------------------------------模式 2，红绿灯时间显示
        if(data_count_seg = 0)then
        seg7com <="0111";
        tmp_seg7data <= data_time_pp rem 10;
        elsif (data_count_seg = 1)then
        seg7com <="1011";
        tmp_seg7data <= data_time_pp / 10;
        elsif (data_count_seg = 2)then
        seg7com <="1101";
        tmp_seg7data <= data_time_cp rem 10;
        else
        seg7com <="1110";
        tmp_seg7data <= data_time_cp /10;
      end if;
else-------------------------------------------------------模式 3，白天黑夜设定时间显示
        if(data_count_seg = 0)then
        seg7com <="0111";
        tmp_seg7data <= data_time_nightin rem 10;
        elsif (data_count_seg = 1)then
```

```
        seg7com <="1011";
        tmp_seg7data <= data_time_nightin / 10;
        elsif (data_count_seg = 2)then
        seg7com <="1101";
        tmp_seg7data <= data_time_nightout rem 10;
        else
        seg7com <="1110";
        tmp_seg7data <= data_time_nightout / 10;
        end if;
    end if;
    end process;


-----------------------------------------------------------------

    process(sysclk)   --24 小时时钟计数
   begin
   if rising_edge(tmp_sig_min) then --输入信号为分钟，当分钟输入信号为上升沿时触发
    data_min_low<=data_min_low+1;      --开始计数
        if(data_min_low = 9)then          --分钟低位达到最大值，执行进位
        data_min_low <=0;                   --清零
        data_min_high<=data_min_high+1;
            if(data_min_high = 5)then    --满一小时进位给小时
            data_min_low <=0;                --清零
            data_min_high <=0;
            data_h_low <= data_h_low+1;
            data_h <= data_h+1;
                if(data_h_low = 9 )then    --小时低位满进位给小时高位
                data_min_low <=0;             --清零
                data_min_high <=0;
                data_h_low <=0;
                data_h_high <=data_h_high+1;
                end if;
            end if;
        end if;
   end if;
    if(data_h = 24)then                    --满 24 小时，全部清零
    data_min_low <=0;
    data_min_high <=0;
    data_h_low <=0;
    data_h_high <=0;
    data_h <= 0;
    end if;
    if(data_h >= data_time_nightin or data_h <data_time_nightout)then --白天黑夜判断
```

```
        night <='1';
        else
        night <='0';
        end if;
        end process;

------------------------------------------------------------

END main;


-----------------------------------分频器器件---------------------------------------
library ieee;
USE ieee.std_logic_1164.all;
ENTITY clkdiv IS
            PORT( clk_in:IN STD_LOGIC;          --时钟信号输入
                        clk_out_ms:OUT STD_LOGIC;    --时钟信号输出
                    clk_out_s:OUT STD_LOGIC;      --时钟信号输出
                        clk_out_min:OUT STD_LOGIC;
                        clk_out_h:OUT STD_LOGIC;
                        clk_out_50ms : OUT STD_LOGIC
                        );
end clkdiv;

architecture div of clkdiv is

SIGNAL data_ms:integer range 0 to 1000000;
SIGNAL data_s:integer range 0 to 100000000;
SIGNAL data_min:integer range 0 to 100;
SIGNAL data_h:integer range 0 to 100;
SIGNAL data_50ms:integer range 0 to 10000000;
SIGNAL Q_ms,Q_s,Q_min,Q_h,Q_50ms:std_logic;

begin

process(clk_in)    --1ms  系统时钟输入
begin
    if rising_edge(clk_in) then--当检测到时钟信号为上升沿时
            if(data_ms=25000) then --由计数器实现分频的功能，
                                        --由 50M/（25000*2）得到信号翻转周期。
                                                --修改 data_ms 的判断值就可以实现不同的分频比
          data_ms<=0;
          Q_ms<=not Q_ms;          --对暂存状态量进行翻转
            else
```

```
        data_ms<=data_ms+1;
          end if;
    end if;
clk_out_ms<=Q_ms;              --将暂存量进行输出
end process;


process(clk_in)   --50ms  当前输出为实际时间（最终没有用到）
  begin
    if rising_edge(clk_in) then
     if(data_50ms=1250000) then --此句为你想要的分频比，in/out=data*2+1 ; data=(in/out   -1)/2
        data_50ms<=0;
       Q_50ms<=not Q_50ms;
     else
        data_50ms<=data_50ms+1;
     end if;
    end if;
clk_out_50ms<=Q_50ms;
end process;


process(clk_in)   --1s  已缩短 10 倍
  begin
    if rising_edge(clk_in) then
     if(data_s=2500000)  then  --data_s=25000000  正 常    data_s=2500000  时  1s  已 经 缩 短  10  倍
--data_s=250000 时 1s 已经缩短 100 倍
        data_s<=0;
       Q_s<=not Q_s;
     else
        data_s<=data_s+1;
     end if;
    end if;
clk_out_s<=Q_s;
end process;

process(Q_ms)   --1min  已缩短 1000 倍
  begin
    if rising_edge(Q_ms) then
     if(data_min=30) then
        data_min<=0;
       Q_min<=not Q_min;
     else
        data_min<=data_min+1;
     end if;
```

```
    end if;
clk_out_min<=Q_min;
end process;


process(Q_min)   --1h  已经缩短 1000 倍（最终没有用到）
   begin
    if rising_edge(Q_min) then
     if(data_h=30) then
          data_h<=0;
         Q_h<=not Q_h;
      else
          data_h<=data_h+1;
      end if;
    end if;
clk_out_h<=Q_h;
end process;

end div;


----------------------------------数码管显示驱动----------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY seg7led IS
      PORT(
           int_in:            IN INTEGER    RANGE 0 TO 16 ;
           data_out: OUT STD_LOGIC_VECTOR(7 downto 0)
           );
END seg7led;

ARCHITECTURE example OF seg7led IS
BEGIN

      process(int_in)
      begin
           case int_in is
                when 0 => data_out <= "11000000"; -- 0
                when 1 => data_out <= "11111001"; -- 1
                when 2 => data_out <= "10100100"; -- 2
                when 3 => data_out <= "10110000"; -- 3
                when 4 => data_out <= "10011001"; -- 4
                when 5 => data_out <= "10010010"; -- 5
```

```
                when 6 => data_out <= "10000010"; -- 6
                when 7 => data_out <= "11111000"; -- 7
                when 8 => data_out <= "10000000"; -- 8
                when 9 => data_out <= "10010000"; -- 9
                when 10 => data_out <= "10001000"; -- A
                when 11 => data_out <= "10000011"; -- b
                when 12 => data_out <= "10100111"; -- c
                when 13 => data_out <= "10100001"; -- d
                when 14 => data_out <= "10000110"; -- E
                when 15 => data_out <= "10001110"; -- F
                when 16 => data_out <= "10111111"; -- --
                when others => NULL;
            end case;
        end process;

END example;
```