**湖南大学电气与信息工程学院**

**本科生课程设计**



**课 题： 交通灯控制电路设计**

**学 院： 电气与信息工程学院**

**姓 名： 黄涛**

**同 组 者： 徐梦杰 吴红 熊若锟**

**学 号： 202107030225**

**班 级： 自动化2102班**

**设计时间： 2023年12月11日**

1. **项目名称**

交通灯控制电路设计

1. **项目内容**
2. **项目目的**

1. 学习VHDL硬件语言，并熟练掌握基本程序的编写过程以及方法。

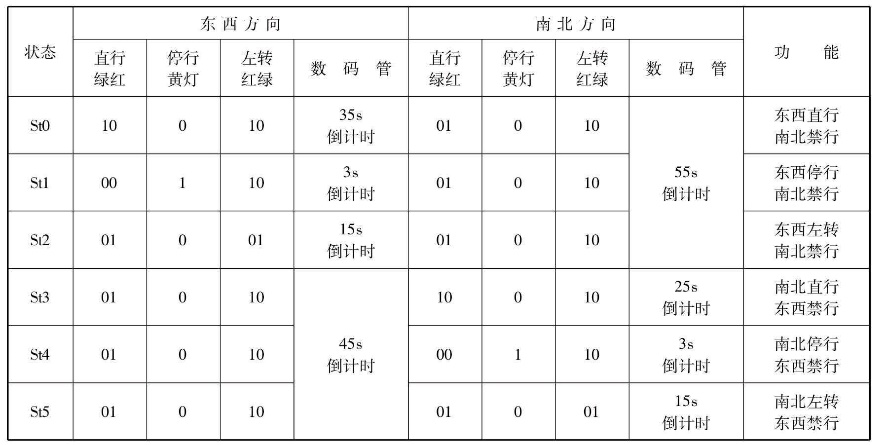
2. 掌握FPGA设计软件ISE Design Suite 14.7和QuartusII14.0的使用。

3. 学习分析实际案例，并能够完整地利用软件仿真出交通灯工作时的波形。

4. 学习程序的下载方法，完成在开发板上实现交通灯相关功能。

1. **相关任务指标**

1.假设一个十字路口为东西南北走向，东西、南北两个方向分别用红、绿、黄3种颜色的LED来指示交通状态。LED的显示规律如下：东西方向的直行绿灯亮，而南北方向的直行红灯亮→东西方向直行绿灯灭后黄灯亮，南北方向仍然是直行红灯亮→南北方向的直行绿灯亮，而东西方向的直行红灯亮一→南北方向直行绿灯灭后黄灯亮，东西方向仍然是直行红灯亮→东西方向的直行绿灯亮，而南北方向的直行红灯亮，如此循环。东西方向是主干道，南北方向是支干道，要求两条道路上的车辆交替运行。

2.要求主干道每次通行时间为55s,支干道每次通行时间为45s。在绿灯灭后，要求黄灯先亮3s,然后红灯亮。在主干道和支干道上还有左转指示，主干道直行40s内，其左转红灯亮，左转绿灯灭40s后至55s内，其左转红灯灭，左转绿灯亮。支道直行30s内，其左转红灯亮，左转绿灯灭，30s后至45s内，其左转红灯灭，左转绿灯亮。

3.当有紧急车辆到达时，东西、南北两方向上的交通信号灯全闪，以便让紧急车辆通过。紧急车辆通过后，交通灯恢复中断前的状态。

4.交通灯具有复位功能，在复位信号有效的情况下，将交通灯的运行状态复位。

1. **项目设计思路**
2. **模块设计**

## 时钟分频模块

DE2-115开发板晶振为50MHz，必须将其分频为周期为1s的时钟脉冲，才能够实现按秒倒计时，同时，如果要使交通灯在有紧急情况时闪烁，需要用到2Hz的脉冲控制，因此，本设计采用两个计数变量依次将50MHz分频为2Hz和1Hz；

## 数码管显示模块

程序中采用十六进制处理数据，而数码管以十进制的方式显示数字，因此务必要通过找到十六进制和十进制数字之间的联系，将程序运行后得到的十六进制结果转化为十进制，进而发送给七段数码管显示；

## 控制模块

通过对当前状态的分析，得到交通灯的直行红黄绿灯以及左转红绿灯的状态，并在开发版上显示出来。以及通过对状态的分析，得到不同方向交通灯的倒计时时间，并对其进行操作；

## 编程思路

在程序编写方面，通过对于VHDL语言的学习，建立起模块化的编程思想，按照实体—>结构体—>进程的格式，使用不同的进程完成各个功能模块程序的编写，方便调用，增强程序的可读性。

1. **硬件配置**

可利用2位数码管作为东西方向倒计时的显示器；2位数码管作为南北方向倒计时的显示器；5位灯做东西方向直行红黄绿灯以及左转红绿灯的指示；5位灯做南北方向直行红黄绿灯以及左转红绿灯的指示；1位switch开关用作复位信号的输入；1位switch开关用作紧急信号的输入。

1. **功能组成**
2. 输入模块

本系统采用键盘作为主要输入部分，配合开关实现数据输入功能。本系统所用到的开关如下所示：

|  |  |
| --- | --- |
| SW[5] | SW[4] |
| rst | prefe |

各按键对应功能如下：

rst：复位开关。开关打开系统转到st0状态，东西方向倒计时显示35s，南北方向倒计时显示55s，东西方向指示灯显示直行绿灯亮和左转红灯亮，南北方向显示直行红灯亮和左转红灯亮；开关关闭则进入st0状态；

Prefe：紧急功能开关。开关打开代表有紧急情况，南北方向指示灯和东西方向指示灯闪烁10s；

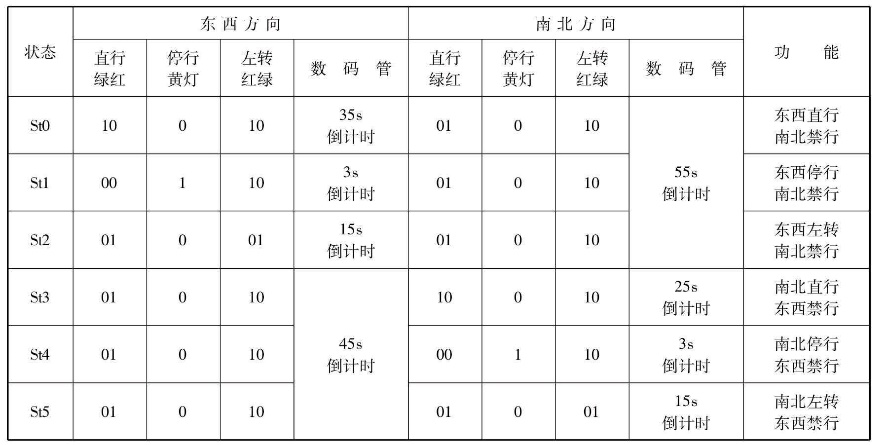
1. 控制模块

控制部分作为整个交通灯控制器系统的核心，它通过输入模块实现数据信息装载处理，并将处理结果通过显示模块显示出来。其涉及到数据的装载、状态转换控制、各状态计时、各状态亮灯情况等。

1. 显示模块

显示部分采用2位数码管来显示南北方向倒计时时间，用2位数码管来显示东西方向方向倒计时时间。

**其中各模拟状态对应的实际功能如下表：**



1. **工作过程**

1.复位按键rst置‘1’，开关打开系统转到st0状态，东西方向倒计时显示35s，南北方向倒计时显示55s，东西方向指示灯显示直行绿灯亮和左转红灯亮，南北方向显示直行红灯亮和左转红灯亮；置‘0’后进入st0状态；

2.紧急功能开关置‘1’，南北方向指示灯和东西方向指示灯闪烁10s；置‘0’后返回原状态。

1. **开发板模块**
2. **功能模块**
3. 控制模块

键盘：键盘控制仅在定时模式中被利用，按动键盘来产生输入信号，系统收到后执行对应的定时操作。其中键盘的消抖是通过检测键盘松开的动作来实现的。

开关：开关打开则相当于对应的输入信号置‘1’，关闭则相当于置‘0’系统收到后执行对应的定时操作。

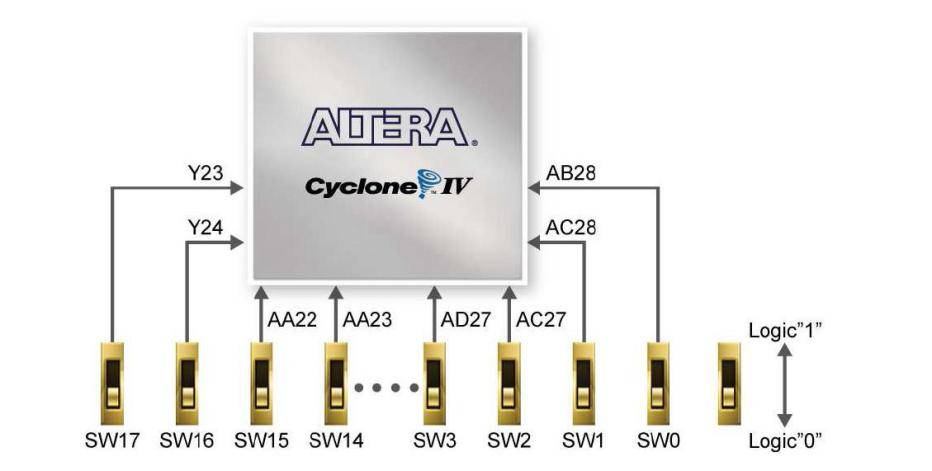
管脚设置如下：

rst（SW[7] PIN\_Y23，复位信号）

prefe（SW[6] PIN\_Y24，紧急情况信号）



开关电路如下图所示



1. 显示模块

数码管：数码管接收系统发出的响应指令。用2位数码管来显示南北方向倒计时时间，用2位数码管来显示东西方向方向倒计时时间。

LED灯：LED灯接收系统发出的响应指令。5位灯做东西方向直行红黄绿灯以及左转红绿灯的指示；5位灯做南北方向直行红黄绿灯以及左转红绿灯的指示。

管脚设置如下：

**LIGHT**（LED0 PIN\_G19，正常工作时该状态灯亮）

**GZLIGHT**（LED1 PIN\_F19，发生故障时该状态灯亮）

**SPEAKER1**（LED2 PIN\_E19，模拟蜂鸣器响，工作结束时灯亮）

**DJ[2…0]**（模拟电机正、反转及暂停状态。正转DJ[0]LED3 PIN\_F21，暂停DJ[1]LED4 PIN\_F18，反转DJ[2]LED5 PIN\_E18）

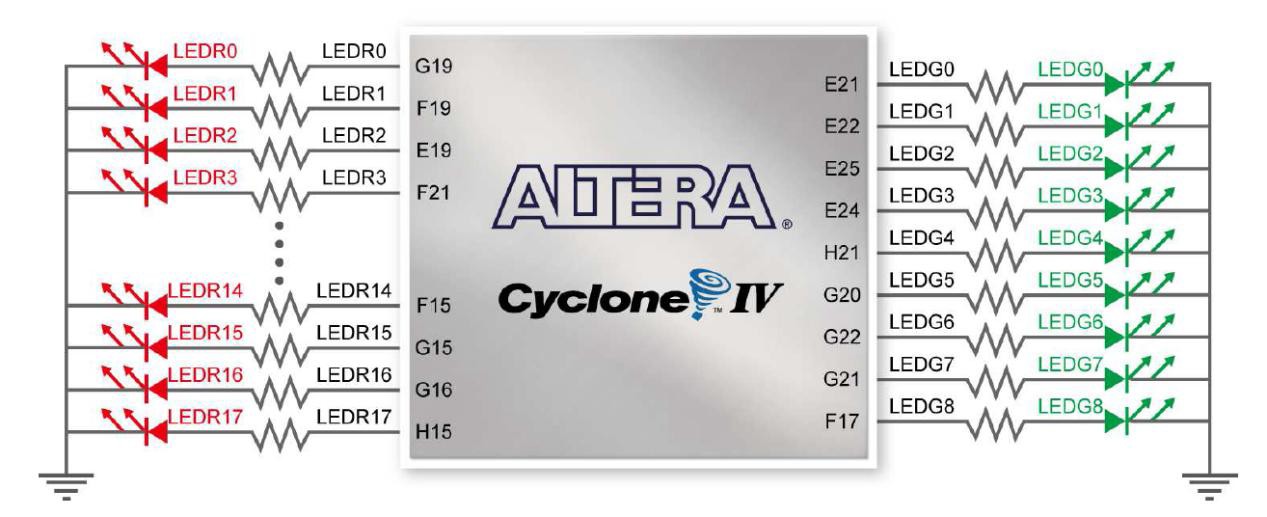
**DIG\_TIL[6…0]**（倒计时低位的数码管 0 显示，6…0：PIN\_H22，PIN\_J22，PIN\_L25，PIN\_L26，PIN\_E17，PIN\_F22，PIN\_G18））

**DIG\_TIH[6…0]**（倒计时高位的数码管 1 显示，6…0：PIN\_U24，PIN\_U23，PIN\_W25，PIN\_W22，PIN\_W21，PIN\_Y22，PIN\_M24）

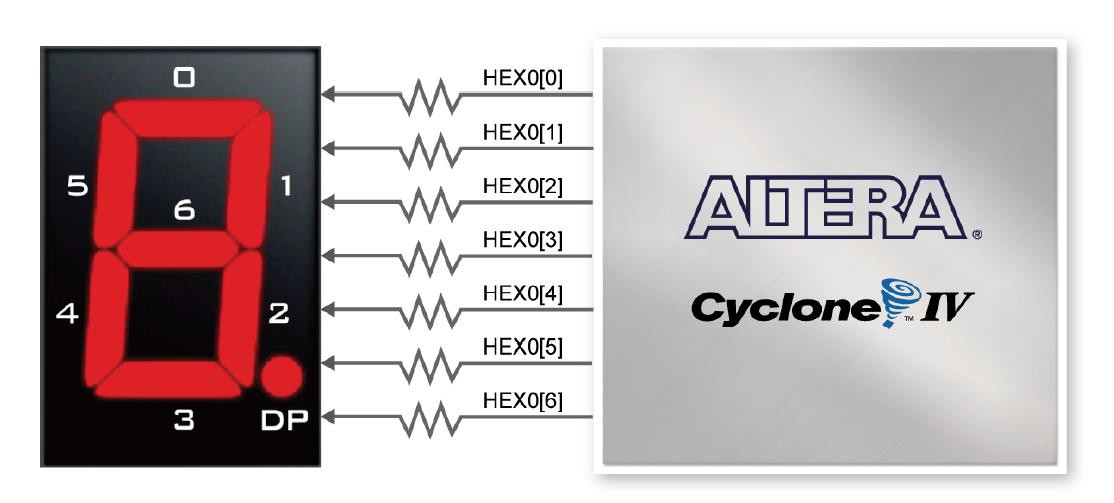
**DIG\_ST[6…0]**（状态显示数码管2，6…0：PIN\_W28，PIN\_W27，PIN\_Y26，PIN\_W26，PIN\_Y25，PIN\_AA26，PIN\_AA25）。

### 

### LED连接如图所示

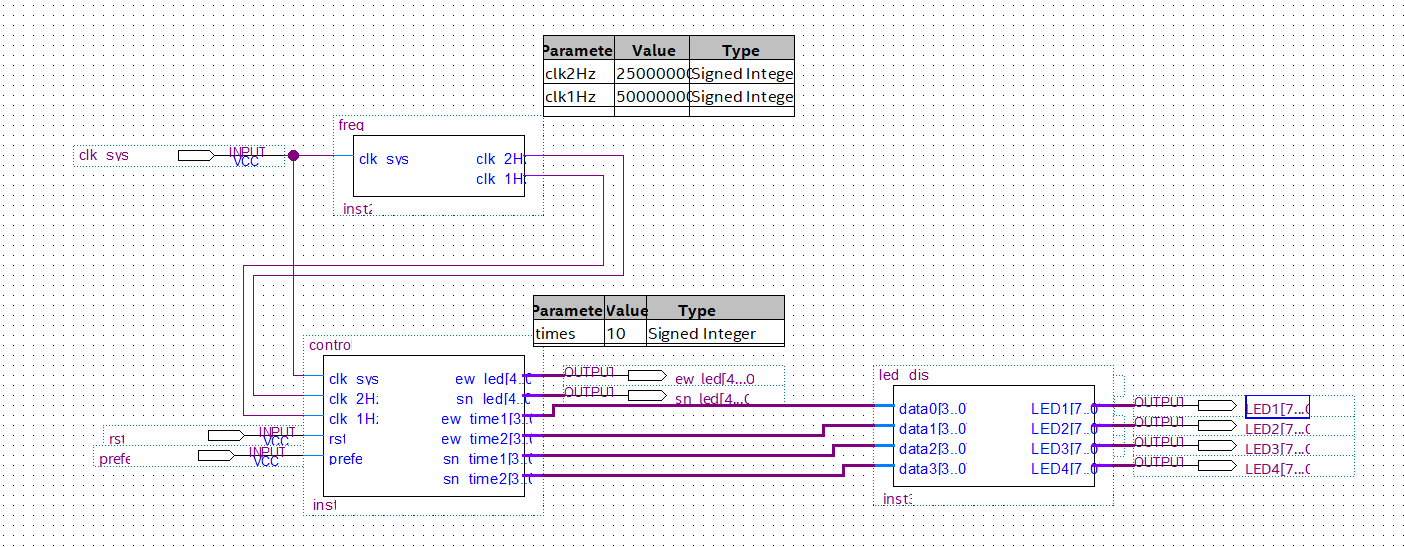


### 共阳极数码管电路如图所示



1. **分析仿真**

**仿真的步骤以及波形**

1. **原理图**
2. **实验代码**

1.顶层模块

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity traffic is

port(clk\_sys:in std\_logic;

rst:in std\_logic;

prefe:in std\_logic;

ew\_LED:out std\_logic\_vector(4 downto 0);

sn\_LED:out std\_logic\_vector(4 downto 0);

LED1:out std\_logic\_vector(7 downto 0);

LED2:out std\_logic\_vector(7 downto 0);

LED3:out std\_logic\_vector(7 downto 0);

LED4:out std\_logic\_vector(7 downto 0));

end traffic;

architecture top of traffic is

component freq is

port(clk\_sys:in std\_logic;

clk\_2Hz,clk\_1Hz:out std\_logic);

end component freq;

component control is

port(clk\_sys,clk\_2Hz,clk\_1Hz:in std\_logic;

rst:in std\_logic;

prefe:in std\_logic;

ew\_led:out std\_logic\_vector(4 downto 0);

sn\_led:out std\_logic\_vector(4 downto 0);

ew\_time1:out std\_logic\_vector(3 downto 0);

ew\_time2:out std\_logic\_vector(3 downto 0);

sn\_time1:out std\_logic\_vector(3 downto 0);

sn\_time2:out std\_logic\_vector(3 downto 0));

end component control;

component led\_disp is

port(

data0,data1,data2,data3:in std\_logic\_vector(3 downto 0);

LED1:out std\_logic\_vector(7 downto 0);

LED2:out std\_logic\_vector(7 downto 0);

LED3:out std\_logic\_vector(7 downto 0);

LED4:out std\_logic\_vector(7 downto 0));

end component led\_disp;

signal clk2Hz,clk1Hz:std\_logic;

signal dat0,dat1,dat2,dat3:std\_logic\_vector(3 downto 0);

begin

u0:freq port map(clk\_sys=>clk\_sys,

clk\_2Hz=>clk2Hz,clk\_1Hz=>clk1Hz);

u1:control port map(rst=>rst,prefe=>prefe,clk\_sys=>clk\_sys, clk\_2Hz=>clk2Hz,clk\_1Hz=>clk1Hz,ew\_led=>ew\_LED,

sn\_led=>sn\_LED,ew\_time1=>dat0,ew\_time2 =>dat1；

sn\_time1=>dat2,sn\_time2 => dat3);

u2:led\_disp port map(data0=>dat0,data1=>dat1,

data2=>dat2,data3 => dat3,

LED1=>LED1,LED2=>LED2,LED3 =>LED3,LED4=>LED4);

end top;

2.分频模块

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

use ieee.std\_logic\_arith.all;

entity freq is

generic(

clk2Hz:integer:=25000000; --设 置 2Hz 的分频系数

clk1Hz:integer:=50000000); --设 置 1Hz 的分频系数

port(

clk\_sys:in std\_logic;

clk\_2Hz,clk\_1Hz:out std\_logic);

end freq;

architecture one of freq is

begin

p1:process(clk\_sys)

variable cnt2:integer range 0 to clk2Hz;

begin

if clk\_sys 'event and clk\_sys ='1' then

if cnt2=clk2Hz-1 then

cnt2:=0;

else

cnt2:=cnt2+1;

end if;

if cnt2<integer(clk2Hz/2)then clk\_2Hz<='1';

else clk\_2Hz<='0';

end if;

end if;

end process p1;

p2:process(clk\_sys)

variable cnt3:integer range 0 to clk1Hz;

begin

if clk\_sys 'event and clk\_sys ='1' then

if cnt3=clk1Hz-1 then

cnt3:=0;

else

cnt3:=cnt3+1;

end if;

if cnt3<integer(clk1Hz/2)then clk\_1Hz<='1';

else clk\_1Hz<='0';

end if;

end if;

end process p2;

end one;

3.控制模块

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity control is

generic(times:integer:=10); --设置优先通过时间

port(clk\_sys,clk\_2Hz,clk\_1Hz:in std\_logic;

rst:in std\_logic;

prefe:in std\_logic; --优先通过按钮

ew\_led:out std\_logic\_vector(4 downto 0);

sn\_led:out std\_logic\_vector(4 downto 0);

ew\_time1:out std\_logic\_vector(3 downto 0);

ew\_time2:out std\_logic\_vector(3 downto 0);

sn\_time1:out std\_logic\_vector(3 downto 0);

sn\_time2:out std\_logic\_vector(3 downto 0));

end control;

architecture two of control is

type states is(st0,st1,st2,st3,st4,st5);

signal current\_state,next\_state:states;

signal ew\_q:std\_logic\_vector(4 downto 0);

signal sn\_q:std\_logic\_vector(4 downto 0);

signal ew\_times:std\_logic\_vector(7 downto 0);

signal sn\_times:std\_logic\_vector(7 downto 0);

begin

pl:process(clk\_1Hz)

variable cnt:integer range times downto 0;

begin

if rst = '1' then

current\_state<=st0;

ew\_times <="00110101";

sn\_times <="01010101";

elsif prefe ='1' then

for i in times downto 0 loop

ew\_q(0)<=clk\_2Hz; sn\_q(0)<=clk\_2Hz;

ew\_q(1)<=clk\_2Hz; sn\_q(1)<=clk\_2Hz;

ew\_q(2)<=clk\_2Hz; sn\_q(2)<=clk\_2Hz;

ew\_q(3)<=clk\_2Hz; sn\_q(3)<=clk\_2Hz;

ew\_q(4)<=clk\_2Hz; sn\_q(4)<=clk\_2Hz;

end loop;

elsif clk\_1Hz 'event and clk\_1Hz='1' then

case current\_state is

when st0 =>

ew\_q(3 downto 0)<="0010";

sn\_q <= "10010";

ew\_times <="00110101";

sn\_times<="01010101";

if ew\_times<4 then

ew\_q(4)<=clk\_2Hz;

else

ew\_q(4)<='1';

end if;

if ew\_times =0 then

current\_state<=st1;

ew\_times <="00000011";

else

current\_state <=st0;

if ew\_times(3 downto 0)=0 then

ew\_times <= ew\_times-7;

else

ew\_times <= ew\_times-1;

end if;

end if;

if sn\_times(3 downto 0)=0 then

sn\_times <= sn\_times-7;

else

sn\_times <= sn\_times -1;

end if;

when st1 =>

ew\_q <="00110";

sn\_q <="01010";

if ew\_times=0 then

current\_state <= st2;

ew\_times<="00010101";

else

current\_state <= st1;

if ew\_times(3 downto 0)=0 then

ew\_times <= ew\_times-7;

else

ew\_times <= ew\_times-1;

end if;

end if;

if sn\_times(3 downto 0)=0 then

sn\_times <= sn\_times-7;

else

sn\_times <= sn\_times-1;

end if;

when st2=>

ew\_q <= "01001";

sn\_q <= "01010";

if ew\_times=0 then

current\_state <= st3;

sn\_times <= "00100101";

ew\_times <= "01000101";

else

current\_state <= st2;

if ew\_times (3 downto 0)=0 then

ew\_times <= ew\_times-7;

else

ew\_times <= ew\_times-1;

end if;

if sn\_times (3 downto 0)=0 then

sn\_times <= sn\_times -7;

else

sn\_times <=sn\_times -1;

end if;

end if;

when st3 =>

ew\_q <="01010";

sn\_q(3 downto 0) <= "0010";

if sn\_times<4 then

sn\_q(4) <= clk\_2Hz;

else

sn\_q(4) <= '1';

end if;

if sn\_times =0 then

current\_state <=st4;

sn\_times <="00000011";

else

current\_state <= st3;

if sn\_times(3 downto 0)=0 then

sn\_times <= sn\_times-7;

else

sn\_times <= sn\_times-1;

end if;

end if;

if ew\_times(3 downto 0)=0 then

ew\_times <= ew\_times-7;

else

ew\_times <= ew\_times-1;

end if;

when st4 =>

ew\_q <= "01010";

sn\_q <= "00110";

if sn\_times =0 then

current\_state <= st5;

sn\_times <= "00010101";

else

current\_state <= st4;

if sn\_times (3 downto 0)=0 then

sn\_times <= sn\_times-7;

else

sn\_times <= sn\_times-1;

end if;

end if;

if ew\_times(3 downto 0)=0 then

ew\_times <= ew\_times-7;

else

ew\_times <= ew\_times-1;

end if;

when st5 =>

ew\_q<="01010";

sn\_q<="01001";

if sn\_times =0 then

current\_state <= st0;

ew\_times <= "00110101";

sn\_times <= "01010101";

else

current\_state <= st5;

if sn\_times(3 downto 0)=0 then

sn\_times <=sn\_times-7;

else

sn\_times <=sn\_times-1;

end if;

if ew\_times(3 downto 0)=0 then

ew\_times <= ew\_times-7;

else

ew\_times <= ew\_times-1;

end if;

end if;

when others =>null;

end case;

end if;

end process pl;

ew\_time1<=ew\_times(3 downto 0);

ew\_time2<=ew\_times(7 downto 4);

sn\_time1<=sn\_times(3 downto 0);

sn\_time2<=sn\_times(7 downto 4);

ew\_led<=ew\_q;

sn\_led<=sn\_q;

end two;

4.数码管显示模块

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity led\_disp is

port(

data0,data1,data2,data3:in std\_logic\_vector(3 downto 0);

LED1:out std\_logic\_vector(7 downto 0);

LED2:out std\_logic\_vector(7 downto 0);

LED3:out std\_logic\_vector(7 downto 0);

LED4:out std\_logic\_vector(7 downto 0));

end led\_disp;

architecture three of led\_disp is

begin

p1:process(data0)

begin

case data0 is

when"0000"=>LED1<="01000000"; --0

when"0001"=>LED1<="01111001"; --1

when"0010"=>LED1<="00100100"; --2

when"0011"=>LED1<="00110000"; --3

when"0100"=>LED1<="00011001"; --4

when"0101"=>LED1<="00010010"; --5

when"0110"=>LED1<="00000010"; --6

when"0111"=>LED1<="01111000"; --7

when"1000"=>LED1<="00000000"; --8

when"1001"=>LED1<="00010000"; --9

when"1010"=>LED1<="00001000"; --A

when"1011"=>LED1<="00000011"; --B

when"1100"=>LED1<="01000110"; --C

when"1101"=>LED1<="00100001"; --D

when"1110"=>LED1<="00000110"; --E

when"1111"=>LED1<="01111111"; --熄 灭

when others=>null;

end case;

end process p1;

p2:process(data1)

begin

case data1 is

when"0000"=>LED2<="01000000"; --0

when"0001"=>LED2<="01111001"; --1

when"0010"=>LED2<="00100100"; --2

when"0011"=>LED2<="00110000"; --3

when"0100"=>LED2<="00011001"; --4

when"0101"=>LED2<="00010010"; --5

when"0110"=>LED2<="00000010"; --6

when"0111"=>LED2<="01111000"; --7

when"1000"=>LED2<="00000000"; --8

when"1001"=>LED2<="00010000"; --9

when"1010"=>LED2<="00001000"; --A

when"1011"=>LED2<="00000011"; --B

when"1100"=>LED2<="01000110"; --C

when"1101"=>LED2<="00100001"; --D

when"1110"=>LED2<="00000110"; --E

when"1111"=>LED2<="01111111"; --熄 灭

when others=>null;

end case;

end process p2;

p3:process(data2)

begin

case data2 is

when"0000"=>LED3<="01000000"; --0

when"0001"=>LED3<="01111001"; --1

when"0010"=>LED3<="00100100"; --2

when"0011"=>LED3<="00110000"; --3

when"0100"=>LED3<="00011001"; --4

when"0101"=>LED3<="00010010"; --5

when"0110"=>LED3<="00000010"; --6

when"0111"=>LED3<="01111000"; --7

when"1000"=>LED3<="00000000"; --8

when"1001"=>LED3<="00010000"; --9

when"1010"=>LED3<="00001000"; --A

when"1011"=>LED3<="00000011"; --B

when"1100"=>LED3<="01000110"; --C

when"1101"=>LED3<="00100001"; --D

when"1110"=>LED3<="00000110"; --E

when"1111"=>LED3<="01111111"; --熄 灭

when others=>null;

end case;

end process p3;

p4:process(data3)

begin

case data3 is

when"0000"=>LED4<="01000000"; --0

when"0001"=>LED4<="01111001"; --1

when"0010"=>LED4<="00100100"; --2

when"0011"=>LED4<="00110000"; --3

when"0100"=>LED4<="00011001"; --4

when"0101"=>LED4<="00010010"; --5

when"0110"=>LED4<="00000010"; --6

when"0111"=>LED4<="01111000"; --7

when"1000"=>LED4<="00000000"; --8

when"1001"=>LED4<="00010000"; --9

when"1010"=>LED4<="00001000"; --A

when"1011"=>LED4<="00000011"; --B

when"1100"=>LED4<="01000110"; --C

when"1101"=>LED4<="00100001"; --D

when"1110"=>LED4<="00000110"; --E

when"1111"=>LED4<="01111111"; --熄 灭

when others=>null;

end case;

end process p4;

end three;