

EDA 课程设计总结

设计题目:基于 Quartus II 的交通灯设计

目 录

序

言.....
.....3

第 一 章 设 计 任 务 和 要
求.....
.....4

1.1 设 计 任
务.....
.....4

1. 2 设 计 要
求.....
.....4

第 二 章 电 路 工 作 原 理 及 方 案 设 计.....
4

第 三 章 单 元 电 路 设 计 与 仿
真.....
.....6

3.1 软 件 原 理
图.....
.....6

3.2	各	模	块	的	原	理	及	其	程	序		
												6

3.2.1	分	频	模	块	的	设	计	及	仿	真	图		
												6	

3.2.2	计	数	模	块	的	设	计	及	仿	真	图		
												7	

3.2.3	控	制	模	块	的	设	计	及	仿	真	图		
												9	

3.2.4	分	位	模	块	的	设	计	及	仿	真	图		
												10	

3.2.5	译	码	模	块	的	设	计	及	仿	真	图		
												11	

第	四	章		系	统	设	计		
									12

4.1	顶	层	电	路	设	计		
								12

4.2	整	体	电	路	波	形	仿	真	
图	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.</			

第	五	章		总
---	---	---	--	---

CPLD (Complex Programmable Logic Device) 是 Complex PLD 的简称,一种较 PLD 为复杂的逻辑元件。CPLD 是一种用户根据各自需要而自行构造逻辑功能的数字集成电路。其基本设计方法是借助集成开发软件平台,用原理图、硬件描述语言等方法,生成相应的目标文件,通过下载电缆(“在系统”编程)将代码传送到目标芯片中,实现设计的数字系统。

它具有编程灵活、集成度高、设计开发周期短、适用范围宽、开发工具先进、设计制造成本低、对设计者的硬件经验要求低、标准产品无需测试、保密性强、价格大众化等特点,可实现较大规模的电路设计,因此被广泛应用于产品的原型设计和产品生产(一般在 10,000

件以下)之中。几乎所有应用中小规模通用数字集成电路的场合均可应用 CPLD 器件。CPLD 器件已成为电子产品不可缺少的组成部分,它的设计和应用成为电子工程师必备的一种技能。

随着基于 CPLD 的 EDA 技术的发展和应用领域的扩大与深入,EDA 技术在电子信息、通信、自动控制用计算机等领域的重要性日益突出。作为一个学电子信息专业的学生,我们必须不断地了解更多的新产品信息,这就更加要求我们对 EDA 有个全面的认识。本程序设计的是交通灯的设计。采用 EDA 作为开发工具,VHDL 语言为硬件描述语言,quartusII 作为程序运行平台,所开发的程序通过调试运行、波形仿真验证,初步实现了设计目标

在一个交通繁忙的十字路口,没有交通灯来控制来往车辆和行人的通行,假设也没有交警,那会发生什么事情呢?后果是难以想象的,可能会陷入一片混乱,甚至瘫痪。当然我们每个人都不希望这样。我们作为社会的一员,每人都有责任为它的更加先进和快捷做出力所能及的事情。我们设计的这个信号控制系统可以通过交通灯控制东西方向车道和南北方向车

术课程内容的理解和掌握有了更深一
电子电路的基本分析方法和设计方法,
的就业或继续深造做好准备。

事故的发生概率。并且经过些次实验使得我们对电子技术
层的认识,也学会使用半导体元件和集成电路,掌握电
进一步提高分析解决实际问题的综合能力,也为将来的

备与要求

第一章 设计任务

为手动操作、自动操作和复位系统,
灯、黄灯和左拐四盏灯,每个方向设
行。红灯亮表示左转和直行车辆禁行;
车辆即将禁行;左拐灯亮表示左转车
通行的时间。

灯、黄灯和左拐灯。显示顺序:绿灯→
西两个方向各设一组倒计时显示器。
允许通行或禁止通行的时间,南北方

1.1 设计任务

设计一个基于 FPGA 的十字路口交通控制器,分为
假设南北方向和东西方向,两个方向分别设置红灯、绿
置一组倒计时显示器,用以指挥车辆和行人有序的通行
绿灯亮表示直行车辆可以通行;黄灯亮表示左转或直行
车辆可以通行;倒计时显示器用来显示允许通行或禁止通

1.2 设计要求

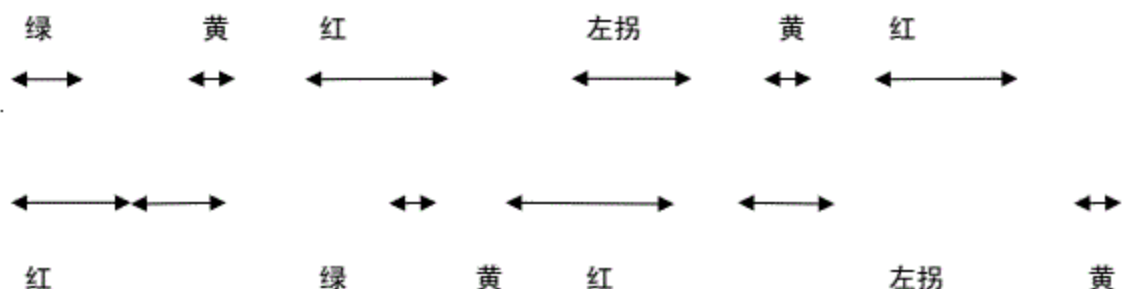
- 1 在十字路口南北和东西两个方向各设一组红灯、绿
黄灯→ 红灯→ 左拐→ 黄灯→ 红灯。在南北和东
- 2 自动控制:设置一组数码管,以倒计时的方式显示

向为主干道，左拐、绿灯、黄灯和红灯显示时间分别是 20s、20s、5s 和 20s。东西方向为次干道，左拐、绿灯、黄灯和红灯显示时间分别为 15s、15s、5s 和 25s。

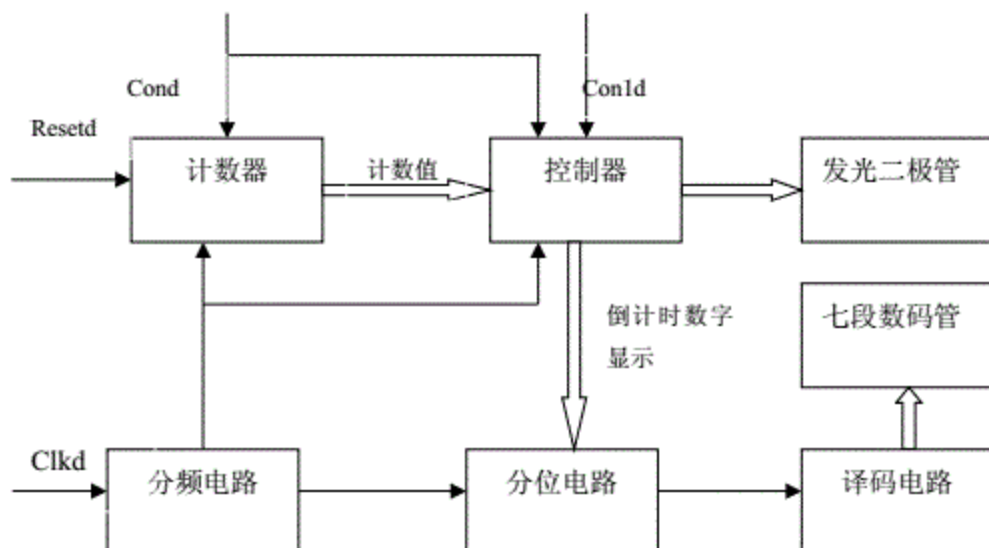
- 3 手动控制：按按钮依次执行以上显示状态，绿灯→ 黄灯→ 红灯→ 左拐→ 黄灯→ 红灯，倒计时显示为“0”。
- 4 系统设有总复位开关，可在任意时间内对系统进行复位。
- 5 通过开关按钮切换交通灯的工作状态。

第二章 电路工作原理及方案设计

交通灯的显示状态：



总体框图：



在 VHDL 设计描述中,采用自顶向下的设计思路,该思路,首先要描述顶层的接口,上面的描述已经规定了交通灯控制的输入输出信号:输入信号:复位开关信号 reset;外部时钟信号 clk。LED 七段显示数码管的输出信号 count1(6 downto 0),count2(6 downto 0),count3(6 downto 0),count4(6 downto 0);在自顶向下的 VHDL 设计描述中,通常把整个设计的系统划分为几个模块,然后采用结构描述方式对整个系统进行描述。根据实验设计的结构功能,来确定使用哪些模块以及这些模块之间的关系。通过上面的分析,不难得知可以把交通灯控制系统划分为 4 个模块:时钟分频模块,计数模块,控制模块,分位译码模块。

分频电路 把 555 多谐振荡器发出的较高频率脉冲用分频电路的到较低频率的时钟信号,本电路通过三次 10 分频分别得到 10Hz、1Hz 的时钟信号。

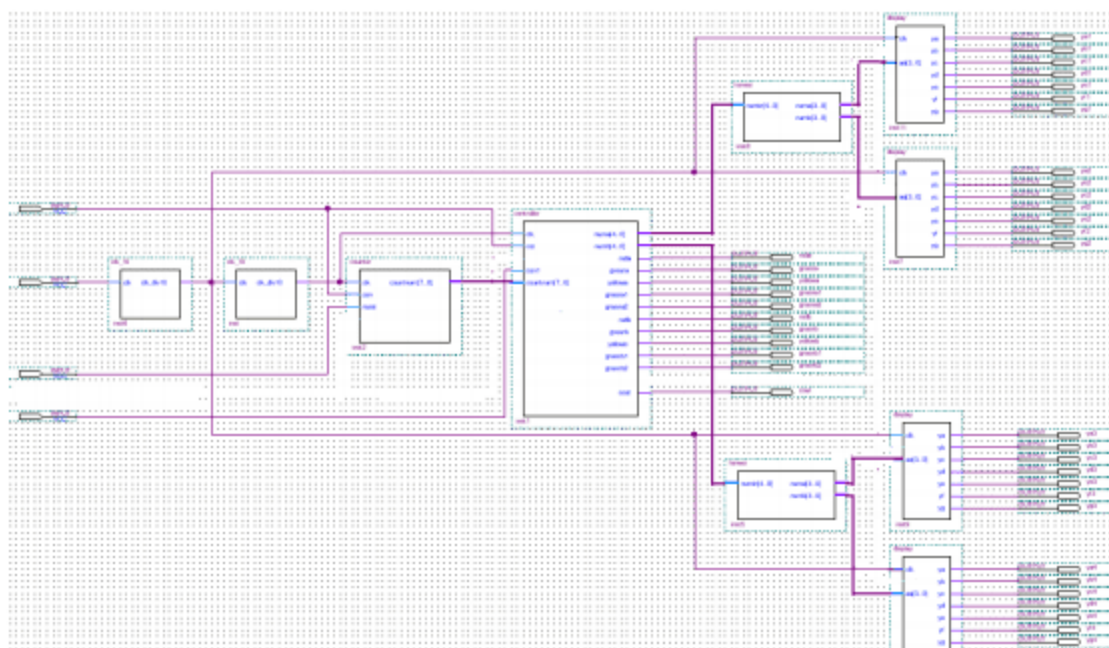
控制器电路 根据计数器的计数值控制发光二极管的亮、灭,以及输出倒计时数值给七段译码管的分位译码电路。当检测到手动控制信号 (Con='1') 时,执行手动控制;

计数器电路 这里需要的计数器的计数范围为 0~89。计到 89 后,下一个时钟沿升为 1 时,开始新一轮计数,此外当系统复位信号 (Reset='1') 使计数器异步清 '0'。手动信号 (Con='1') 使系统清 '0'。

分位译码电路 因为控制器输出的倒计时数值可能是 1 位或者 2 位十进制数,所以在七段数码管的译码电路前要加上分位电路(即将其分为 2 个 1 位的十进制数,如 20 分为 2 和 0,7 分为 0 和 7)。七段数码管的译码电路根据控制电路的控制信号,驱动交通灯的显示,通过输入二进制数值,输出信号点亮二极管,我们用的是共阳极数码管,因此译码电路输出逻辑数值 '0' 点亮二极管,译码电路输出逻辑数值 '1' 熄灭二极管。

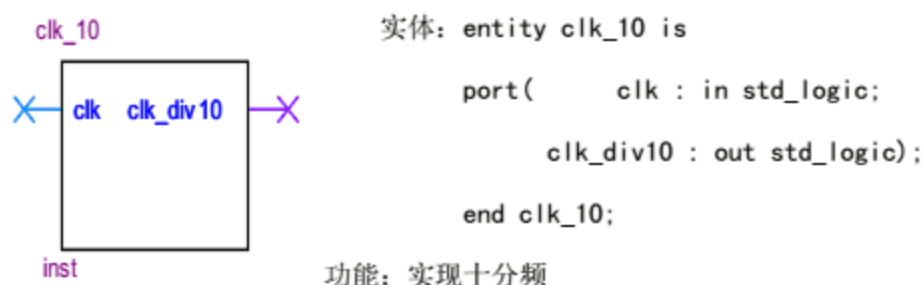
第三章 单元电路设计与仿真

3.1 软件原理图



3.2 各模块的原理及其程序

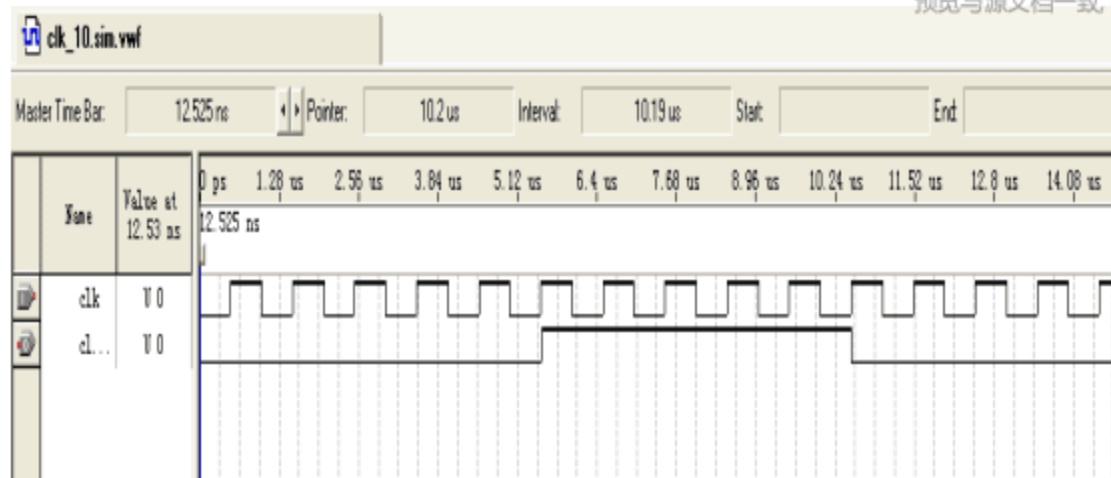
3.2.1 分频模块的设计及仿真图



接口: clk—脉冲输入

Clk-div10—十分频后脉冲输出

仿真结果如下:



原创力文档
max.book118.com
预览与源文档一致, 下载高清无水印

3.2.2 计数模块的设计及仿真图

实体: entity counter is



```
port(  clk0:in std_logic;
        con:in std_logic;
        reset:in std_logic;
        countnum:buffer integer range 0 to 89);

end counter;
```

功能: 实现 0 到 89 的计数

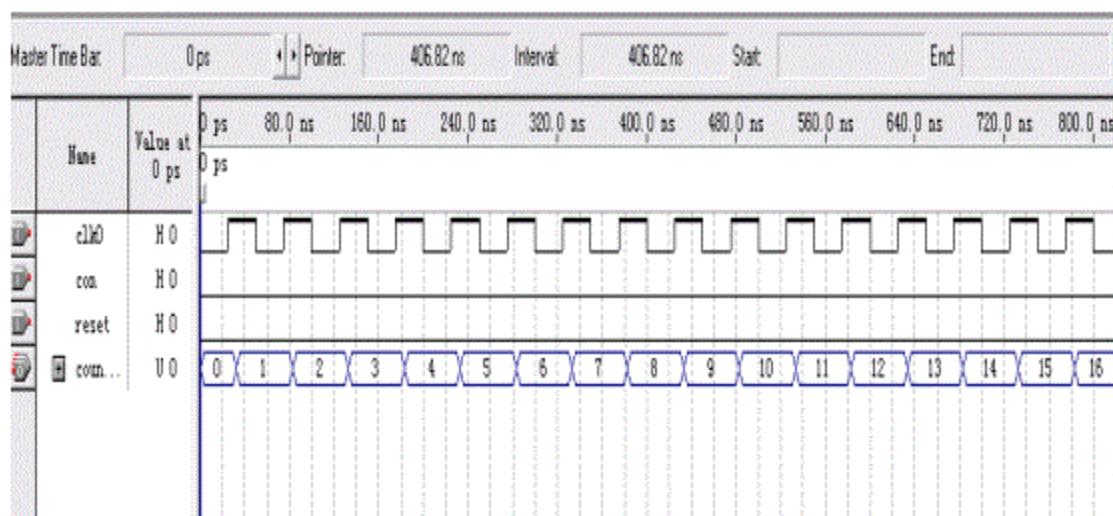
clk0—脉冲输入

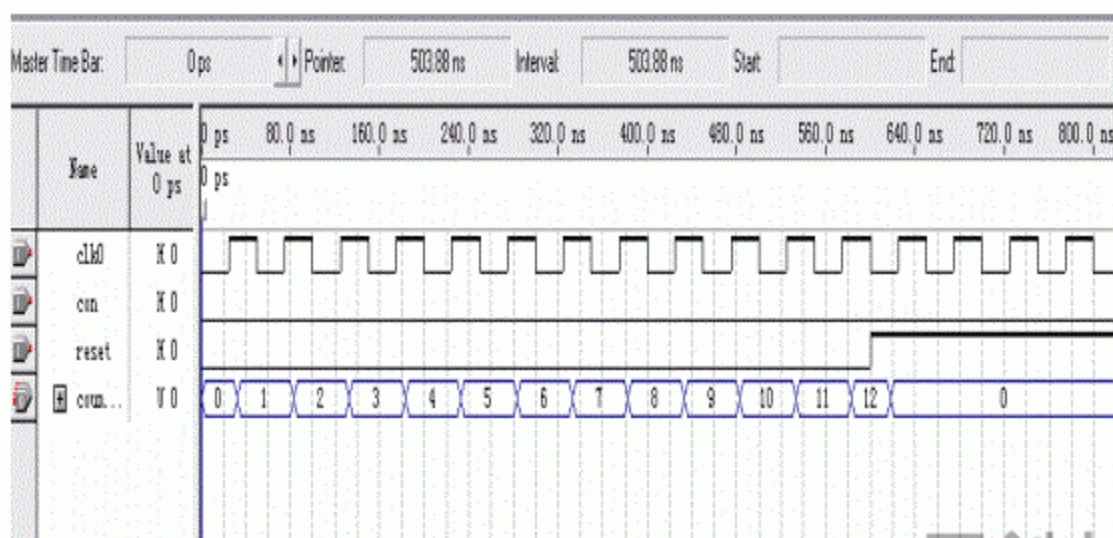
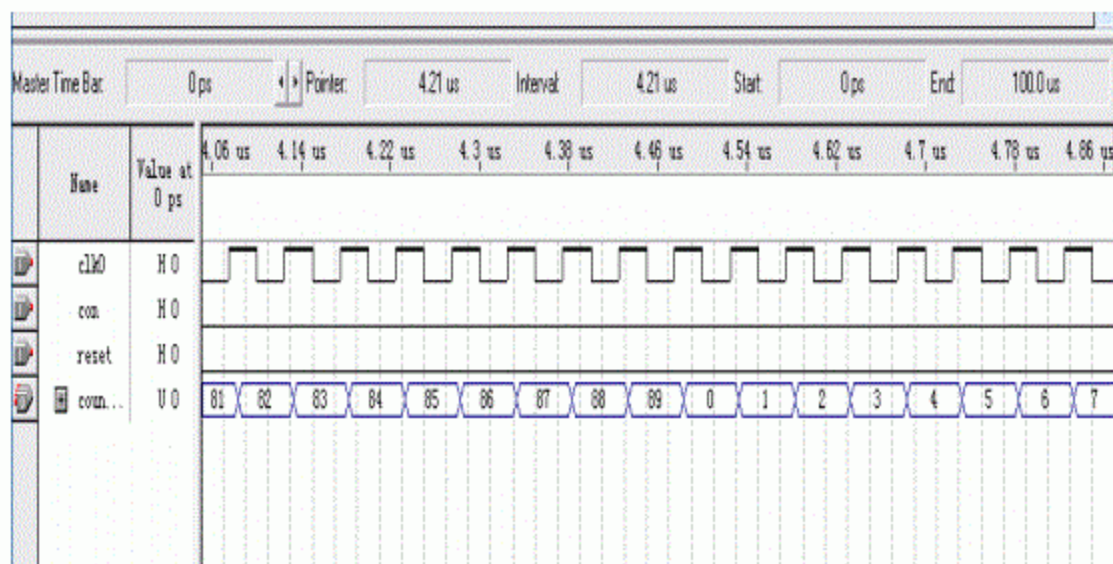
con—手动控制信号

reset—复位信号

countnum—计数输出

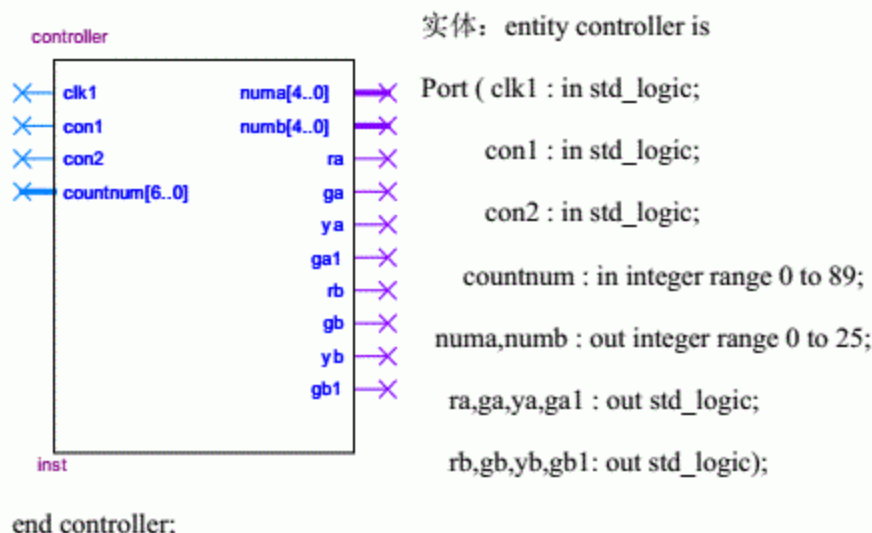
仿真波形如下:





原创力文档
max.book118.com
预览与源文档一致, 下载高清无水印

3.2.3 控制模块的设计及仿真图



功能：控制发光二极管的亮、灭，以及输出倒计时数值给七段译码管的分位译码电路。

clk1—脉冲信号输入

con1—手动控制信号

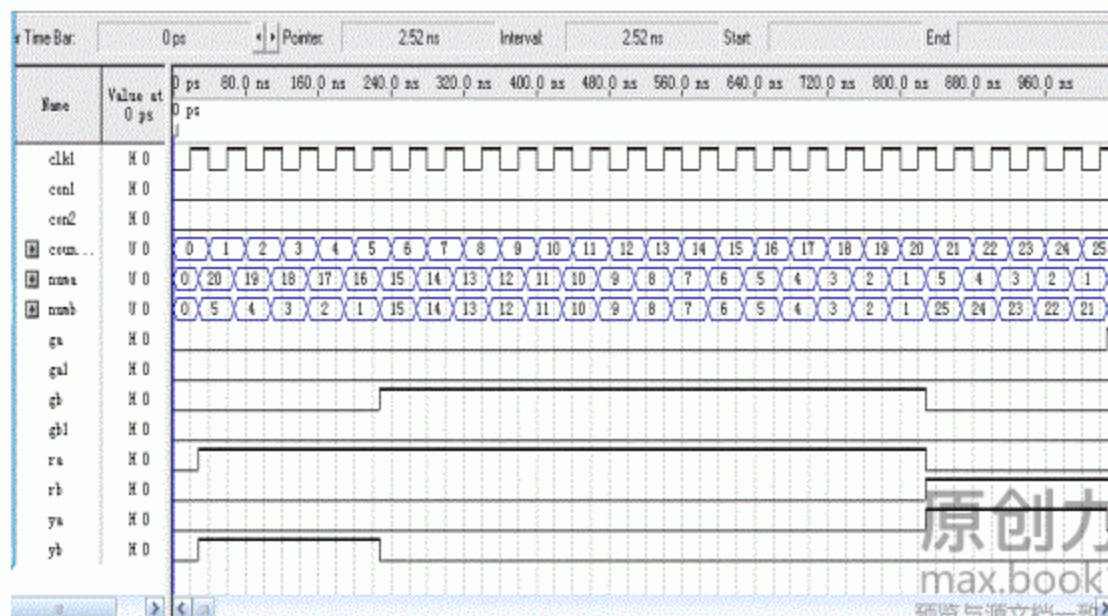
con2—状态控制信号

countnum—计数输入

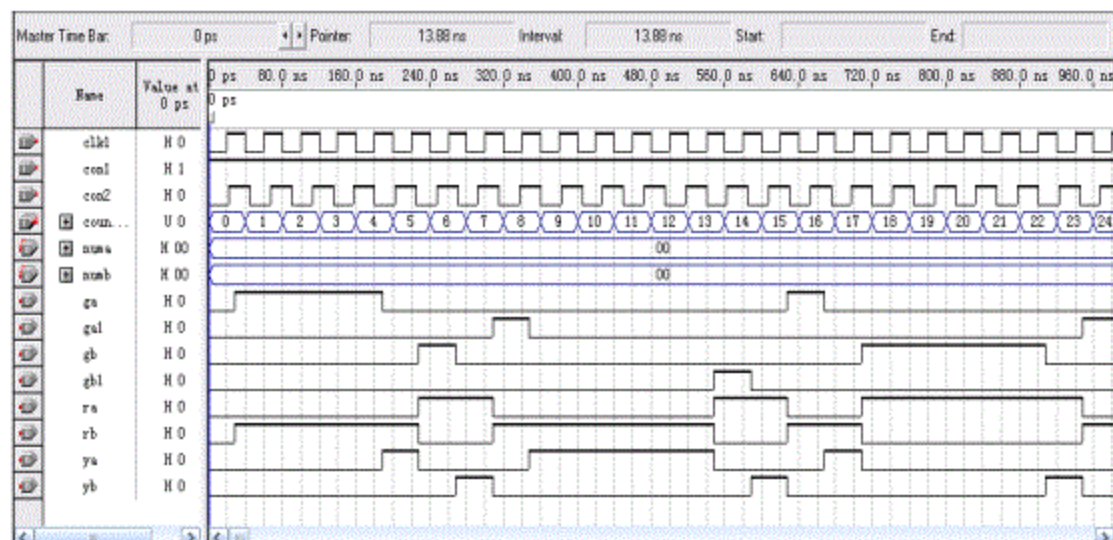
numa, numb—两个方向的倒计时数值输出

ra, ga, ya, gal, rb, gb, yb, gbl—发光二极管输出

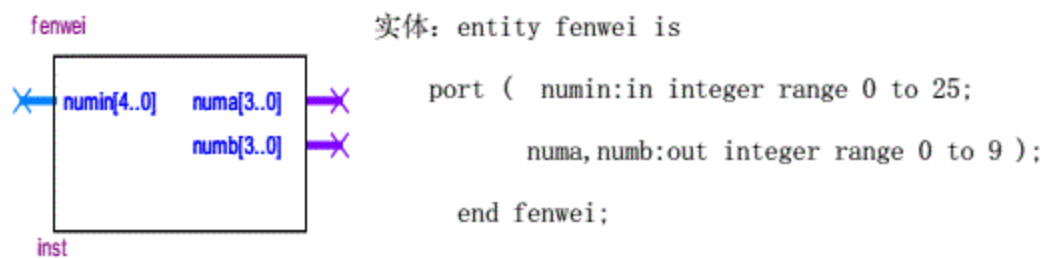
仿真波形如下：



原创力文档
max.book118.com
预览与源文档一致 下载高清无水印



3.2.4 分位模块的设计与仿真图

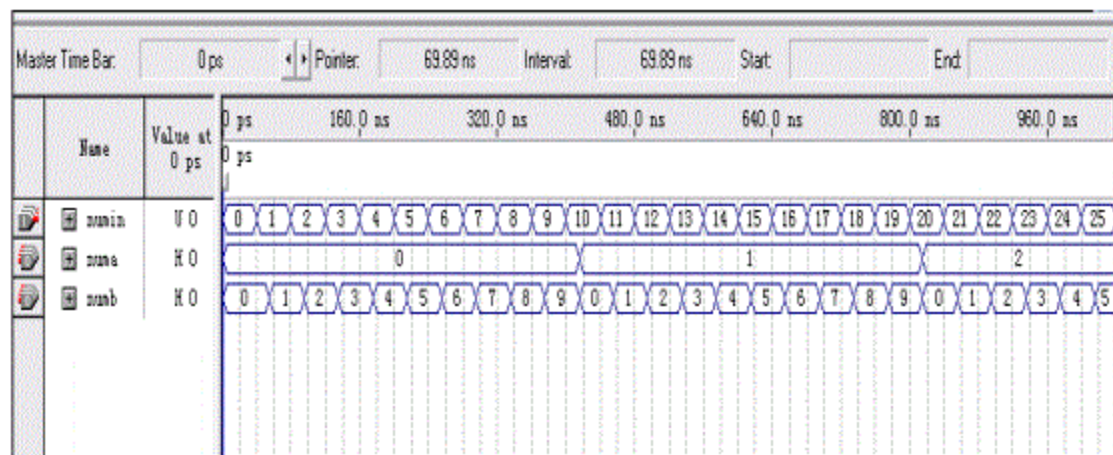


功能: 把倒计时的数值分成 2 个 1 位的十进制数。

numin: 倒计时数值输入

numa, numb—将数值分为 2 个 1 位的十进制输出

仿真波形如下:



3.2.5 译码模块的设计及仿真图

实体: entity display is

port(

clk2:in std_logic;

bb: in std_logic_vector(3 downto 0);

ya:out std_logic;

yb:out std_logic;

yc:out std_logic;

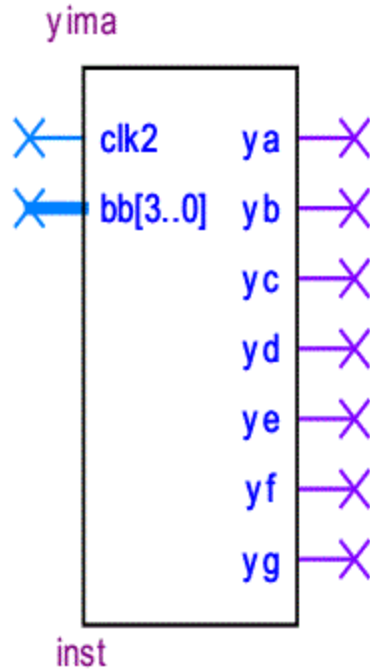
yd:out std_logic;

ye:out std_logic;

yf:out std_logic;

yg:out std_logic);

end display;



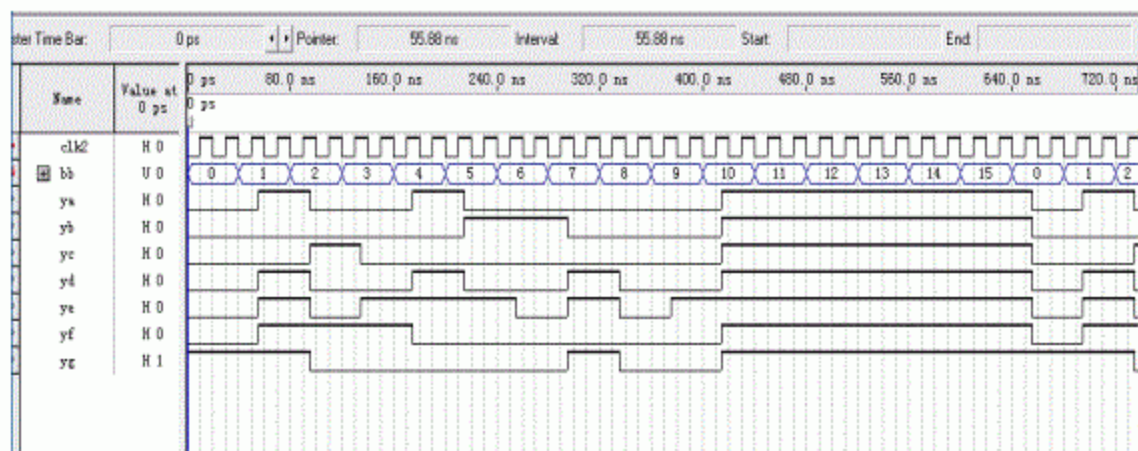
功能: 根据控制电路的控制信号, 驱动交通灯的显示, 通过输入二进制数值, 输出信号点亮二极管。

bb[3..0]——BCD 码输入

clk2——脉冲输入

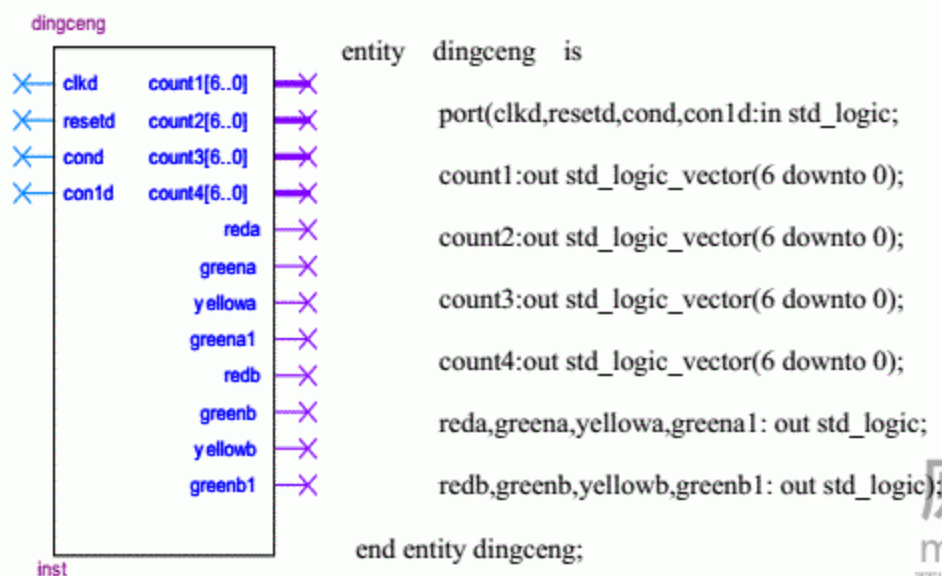
ya, yb, yc, yd, ye, yf, yg——七段数码管显示输出

仿真波形如下:



第四章 系统设计

4.1 顶层文件的设计



clk2—脉冲输入

resetd—复位信号

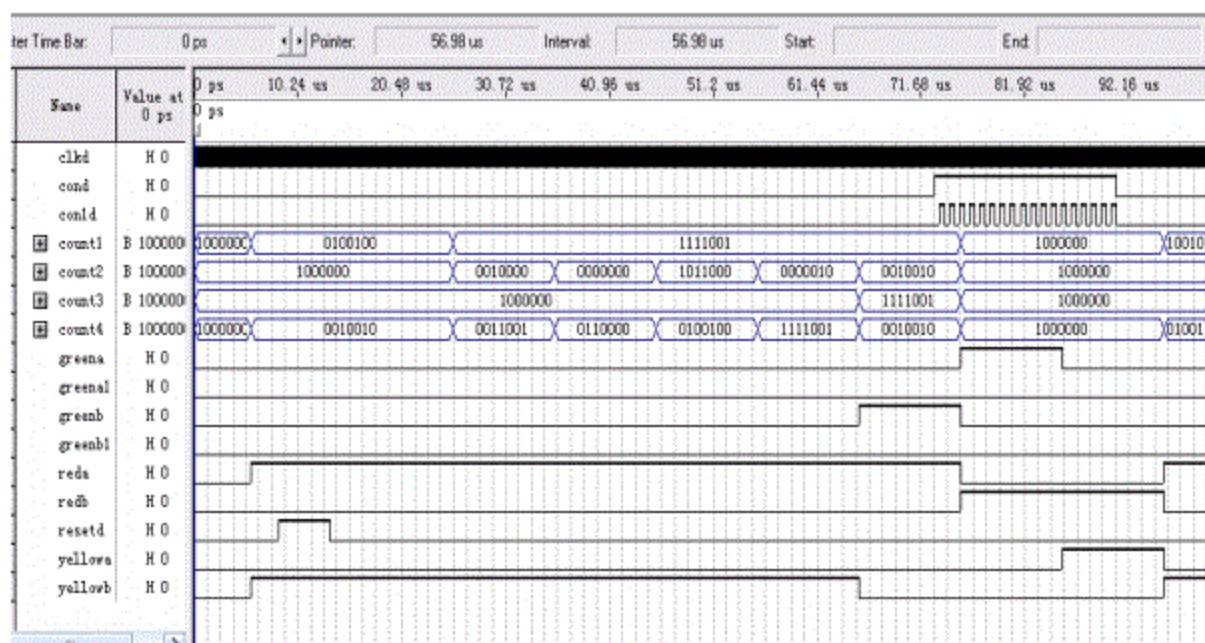
con—手动控制信号

con1d—状态控制信号

count1, count2, count3, count4—七段数码管显示

reda,greena,yellowa, greenal, redb,greenb,yellowb,greenbl—发光二极管输出

4.2 整体电路波形图



第五章 总结

课程设计对学生而言是其所对所学课程内容掌握情况的一次自我验证，从而有着极其重要的意义。通过课程设计能提高学生对其所学知识的综合应用能力，能全面检查并掌握所学内容，本学期我们进行了数字电路课程设计，老师命题，六人一大组，两人一小组进行设计（包括设计总体方案、硬件电路、软件设计、焊接、调试等工作）。趣味性强，同时也可以学到很多东西。我们做的是交通灯控制器的设计。

在这学期的课程设计中，在收获知识的同时，还收获了阅历，收获了成熟，在此过程中，我们通过查找大量资料，请教老师，以及不懈的努力，不仅培养了独立思考、动手操作的能力，在各种其它能力上也都有了提高。更重要的是，我们学会了很多学习的方法。而这是日后最实用的，真的是受益匪浅。要面对社会的挑战，只有不断的学习、实践，再学习、再实践。

通过这个程序设计让我学会一种新的语言，对数字系统结构也有了更进一步的了解和认识，对我以后的学习有很大的帮助。希望其他人在看再做类似设计时有所借鉴。

通过几天的课程设计，我对数据库软件 EDA 技术、VHDL、等系列知识都有了一定的了解。使用 EDA 技术开发页面的能力也有了很大提高。

另外，我还学到了一个人的力量毕竟有限，但是团队的力量势不可挡，我们不能只靠自己，遇到不会的要多多求助同学，

这次课程设计虽然结束了，也留下了很多遗憾，因为由于时间的紧缺和许多课业的繁忙，并没有做到最好，但是，最起码我们没有放弃，它是我们的骄傲！相信以后我们会以更加积极地态度对待我们的学习、对待我们的生活。我们的激情永远不会结束，相反，我们会更加努力，努力的去弥补自己的缺点，发展自己的优点，去充实自己，只有在了解自己的长短之后，我们会更加珍惜拥有的，更加努力的去完善它，增进它。只有不断的测试自己，挑战自己，才能拥有更多的成功和快乐！快乐至上，享受过程，而不是结果！认真对待每一个实验，珍惜每一分一秒，学到最多的知识和方法，锻炼自己的能力，这个是我们实时测量技术试验上学到的最重要的东西，也是以后都将受益匪浅的！

参考文献

- 1、《VHDL 与数字电路设计》. 卢毅, 赖杰. 科学出版社
- 2、《VHDL 程序设计》(第二版). 曾繁泰等. 清华大学出版社
- 3、《VHDL 入门与应用》陈雪松, 滕立中. 人民邮电出版社
- 4、《EDA 技术与 VHDL》(第二版). 潘松, 黄继业. 清华大学出版社
- 5、《数字电路技术基础》(第二版). 阎石. 高等教育出版社

附录 总程序

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity dingceng is
port(clkd, resetd, cond, con1d:in
std_logic;
count1:out      std_logic_vector(6
downto 0);
count2:out      std_logic_vector(6
downto 0);
count3:out      std_logic_vector(6
downto 0);
count4:out      std_logic_vector(6
downto 0);
reda, greena, yellowa, greena1: out
std_logic;
redb, greenb, yellowb, greenb1: out
std_logic);
end entity dingceng;
architecture one of dingceng is
component clk_10
port(clk:in std_logic;
      clk_div10 : out
std_logic);
end component;
component counter
port
(clk0:in std_logic;
con:in std_logic;
reset:in std_logic;
```

```

        countnum:buffer
integer range 0 to 89);
end component;
component controller
port
    ( clk1 : in std_logic;
      con1 : in std_logic;
      con2: in std_logic;
      countnum : in integer
range 0 to 89;
      numa,numb : out integer
range 0 to 25;
      ra, ga, ya, ga1: out
std_logic;
      rb, gb, yb, gb1 : out
std_logic);
end component;
component fenwei
port ( numin:in integer range 0
to 25;
      numa,numb:out integer
range 0 to 9 );
end component;
component yima
port(clk2:in std_logic;
      bb: in integer range
0 to 9;
      ya:out std_logic;
      yb:out std_logic;
      yc:out std_logic;
      yd:out std_logic;
      ye:out std_logic;
      yf:out std_logic;
      yg:out std_logic);
end component;
signal a,b,c:std_logic;
signal ww:integer range 0 to 89;
signal yy1,yy2:integer range 0
to 25;
signal tt1,tt2,tt3,tt4:integer
range 0 to 9;
begin
    u0:          clk_10          port
map(clk=>clkd,clk_div10=>a);
    u1:          clk_10          port
map(clk=>clkd,clk_div10=>b);
    u2:          clk_10          port
map(clk=>b,clk_div10=>c);
    u3:counter port
map(clk0=>c, con=>cond, reset=>res
etd, countnum=>ww);
    u4:controller port map
    (clk1=>c, con1=>cond, con2=>con1d,
countnum=>ww, numa=>yy1, numb=>yy2, r
a=>reda, ga=>greena, ya=>yellowa, g
a1=>greena1, rb=>redb, gb=>greenb,
yb=>yellowb, gb1=>greenb1);
    u5:fenwei          port
map(numin=>yy1, numa=>tt1, numb=>t
t2);
    u6:fenwei          port      map
    (numin=>yy2, numa=>tt3, numb=>tt4)
    ;
    u7:yima
port
map(clk2=>b, bb=>tt1, ya=>count1(0),
yb=>count1(1), yc=>count1(2), yd=>
count1(3), ye=>count1(4), yf=>coun
t1(5), yg=>count1(6));
    u8:yima          port      map
    (clk2=>b, bb=>tt2, ya=>count2(0), y
b=>count2(1), yc=>count2(2), yd=>c
ount2(3), ye=>count2(4), yf=>count
2(5), yg=>count2(6));
    u9:yima          port
map(clk2=>b, bb=>tt3, ya=>count3(0),
yb=>count3(1), yc=>count3(2), yd=>
count3(3), ye=>count3(4), yf=>coun
t3(5), yg=>count3(6));
    u10:yima          port
map(clk2=>b, bb=>tt4, ya=>count4(0),
yb=>count4(1), yc=>count4(2), yd=>
count4(3), ye=>count4(4), yf=>coun
t4(5), yg=>count4(6));
end architecture one;

library ieee;

```

```

use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity clk_10 is
port(      clk : in std_logic;
      clk_div10 : out std_logic);
end clk_10;
architecture one of clk_10 is
signal clk_temp : std_logic;
begin
process(clk)
variable      counter      :
std_logic_vector(2 downto 0);
constant md : std_logic_vector(2
downto 0) := "101";
begin
if(clk'event and clk='1') then
if(counter = md)then
counter := (others => '0');
clk_temp <= not clk_temp;
end if;
counter := counter + 1 ;
end if;
end process;
clk_div10 <= clk_temp;
end one;

library ieee;
use ieee.std_logic_1164.all;
entity counter is
port
(  clk0:in std_logic;
  con:in std_logic;
  reset:in std_logic;
  countnum:buffer integer range
0 to 89);
end counter;
architecture one of counter is
begin
process (reset,clk0)
begin
if reset='1' then
countnum <= 0;
elsif rising_edge( clk0 )
then
if con='1' then
countnum <= 0;
else
if countnum=89 then
countnum <= 0;
else
countnum <= countnum
+ 1;
end if;
end if;
end if;
end process;
end one;

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity controller is
port
(      clk1 : in std_logic;
      con1 : in std_logic;
      con2 : in std_logic;
      countnum : in integer range 0
to 89;
      numa,numb : out integer range 0
to 25;
      ra,ga,ya,ga1 : out std_logic;
      rb,gb,yb,gb1: out std_logic);
end controller;
architecture one of controller is
signal m : integer range 0 to 7;
begin
process
(
clk1,con1,con2,countnum )
variable
aa:std_logic_vector(7 downto 0);
begin
if (clk1'event and clk1 =
'1' )then
if con1 ='0' then
if countnum >= 65then
numb<=90-countnum;

```

```

        numa <= 90-countnum;
        if countnum >= 70 then
            aa := "00011000";
        else
            aa := "00101000";
            numa <= 70-countnum;
        end if;
    elsif countnum >= 45 then
        numb <= 65-countnum;
        numa <= 65-countnum;
        if countnum >= 50 then
            aa := "10000100";
        else
            aa := "10000010";
            numb <= 50-countnum;
        end if;
    elsif countnum >= 20 then
        numa <= 45-countnum;
        numb <= 45-countnum;
        if countnum >= 25 then
            aa := "01001000";
        else
            aa := "00101000";
            numa <= 25-countnum;
        end if;
    elsif countnum >= 0 then
        numa <= 20-countnum;
        numb <= 20-countnum;
        if countnum >= 5 then
            aa := "10000100";
        else
            aa := "10000010";
            numb <= 5-countnum;
        end if;
    end if;
    else
        if con1= '1' then
            numa <= 0;
            numb <= 0;
            if con2 = '1' then
                if m = 7 then
                    m <= 0;
                else
                    m <= m + 1;
                end if;
            end if;
            if m = 0 then
                aa := "01001000";
            elsif m= 1 then
                aa := "00101000";
            elsif m=2 then
                aa := "10000100";
            elsif m=3 then
                aa := "10000010";
            elsif m=4 then
                aa := "00011000";
            elsif m=5 then
                aa := "00101000";
            elsif m=6 then
                aa := "10000001";
            elsif m=7 then
                aa := "10000010";
            end if;
        end if;
    end if;
    ra <= aa(7);
    ga <= aa(6);
    ya <= aa(5);
    ga1 <= aa(4);
    rb <= aa(3);
    gb <= aa(2);
    yb <= aa(1);
    gb1 <= aa(0);
end process;
end one;

library ieee;
use ieee.std_logic_1164.all;
entity fenwei is
    port (
        numin:in integer
        range 0 to 25;
        numa,numb:out integer
        range 0 to 9 );
end fenwei;
architecture one of fenwei is
    begin

```

```

        process( numin )
            variable      numa1,numb1:
integer range 0 to 9;
        begin
            if numin >= 20 then
                numa1 :=2;
                numb1 := numin - 20;
            elsif numin >= 10 then
                numa1 := 1;
                numb1 := numin - 10;
            else
                numa1 := 0 ;
                numb1 := numin ;
            end if ;
            numa <= numa1;
            numb <= numb1;
        end process ;
end one ;

library ieee;
use ieee.std_logic_1164.all;
entity display is
    port(
        clk2:in std_logic;
        bb:                                     in
std_logic_vector(3 downto 0);
        ya:out std_logic;
        yb:out std_logic;
        yc:out std_logic;
        yd:out std_logic;
        ye:out std_logic;
        yf:out std_logic;
        yg:out std_logic);
end display;
architecture one of display is
    signal temp: std_logic_vector(6
downto 0);
begin
    process(bb)
    begin
        case bb is
        when "0000"    =>    temp    <=
"1000000";
            when "0001"    =>    temp    <=
"1111001";
            when "0010"    =>    temp    <=
"0100100";
            when "0011"    =>    temp    <=
"0110000";
            when "0100"    =>    temp    <=
"0011001";
            when "0101"    =>    temp    <=
"0010010";
            when "0110"    =>    temp    <=
"0000010";
            when "0111"    =>    temp    <=
"1011000";
            when "1000"    =>    temp    <=
"0000000";
            when "1001"    =>    temp    <=
"0010000";
            when others    =>    temp    <=
"1111111";
        end case;
    end process;
    ya<=temp(0);
    yb<=temp(1);
    yc<=temp(2);
    yd<=temp(3);
    ye<=temp(4);
    yf<=temp(5);
    yg<=temp(6);
end one;

```