

(此文档为 word 格式，下载后您可任意编辑修改！)

基于FPGA的交通灯控制器设计

摘 要

超高速硬件描述语言 VHDL 是对数字系统进行抽象的行为与功能描述到具体的内部线路结构描述，利用 EDA 工具可以在电子设计的各个阶段、各个层系进行计算机模拟验证，保证设计过程的正确性，可大大降低设计成本，缩短设计周期。本文介绍的数字秒表设计，利用基于 VHDL 的 EDA 设计工具，采用大规模可编程逻辑器件 FPGA 通过设计芯片来实现系统功能。

交通灯控制系统可以实现路口红绿灯的自动控制。基于 FPGA 设计的交通灯控制系统具有电路简单、可靠性强、实时快速擦写、运算速度快、故障率低、可靠性高，而且体积小等特点。本设计采用 Altera 公司 Cyclone 系列的 EPIC3T1444C8 芯片，在 Quartus II 软件平台上使用 VHDL 语言，采用自顶向下的设计方法对系统进行了模块化设计和综合，并进行了仿真。该系统可实现十字路口红绿灯及左转弯控制和倒计时显示，仿真结果结果表明系统能够自动控制交通灯转变。

关键词：VHDL，交通灯，EDA

THE LIGHT CONTROLLER BASED ON FPGA DESIGN

ABSTRACT

With the development of the economy, microelectronic technology, computer technology and the automatic theory are developed rapidly. Its application is becoming more and more widely. But the traffic light as an important traffic, it also all the system of the traffic light, The system of traffic control which designed based on FPGA is received by more and more people.

The system of traffic control can realize the automatic control of traffic light in a intersection. The system of traffic control which designed based on FPGA , toerase function quickly with real-time applications, system Quartus II and the form of the VHDL language, we simulate the system by using EP1C3T1444C8 Mars device and the result show that the download system can automatically control the traffic lights.

KEY WORDS: VHDL, traffic light, EDA

目 录

<u>前 言</u>	
------------------	--

第 1 章	FPGA 概述	
1.1	FPGA 的简介	
1.2	FPGA 的应用	
第 2 章	VHDL 硬件描述语言	
2.1	VHDL 程序基本结构	
2.1.1	实体	
2.1.2	结构体	
2.1.3	库	
2.2	VHDL 语言	
2.2.1	VHDL 文字规则	
2.2.2	VHDL 数据对象	
2.2.3	VHDL 数据类型	
2.2.4	VHDL 顺序语句	
2.2.5	VHDL 并行语句	
第 3 章	系统设计与仿真	
3.1	系统介绍	
3.1.1	设计任务	
3.1.2	设计要求	
3.2	系统设计仿真	
3.2.1	顶层框图的设计	
3.2.2	时序状态图的设计	
3.2.3	工程设计流程框图：	
3.2.4	芯片的选择	
3.2.5	各个模块的设计与仿真	
结 论		
谢 辞		
参考文献		
附 录		
外文资料翻译		

前 言

当今社会是数字化的社会，是数字集成电路广泛应用的社会。数字集成电路本身在不断进行更新换代，随着微电子技术的发展，设计与制造集成电路的任务已不完全由半导体厂商来独立承担。系统设计师更愿意自己设计专业集成电路（ASIC）芯片，而且希望设计周期尽可能短，最好在实验室里就能设计出合适的ASIC芯片，并且立即投入实际应用之中，因而出现了现场可编程器件（FPLD）。现场可编程门阵列（FPGA）即属其中应用最广泛的一种。

随着电子技术的发展，特别是大规模集成电路和计算机技术的研制和发展，让电子产品设计有了更好的应用市场，实现方法也有了更多的选择，而电子电路的设计却变得越来越复杂，使用“语言”进行电子设计已成为一种趋势。现代电子系统设计方法是设计师自己设计芯片来实现电子系统的功能，将传统的固件选用及电路板设计工作放在芯片设计中进行。在这些专业化软件中，EDA(Electronic Design Automation)具有一定的代表性，EDA技术是一种基于芯片的现代电子系统设计方法。基于EDA技术的现场可编程门阵列(FPGA)电 (ASIC)，在数字系统设计和控制电路中越来越受到重视。VHDL语言是电子设计的主流硬件描述语言,它更适合进行行为描述,这种方式使得设计者专注于电路功能的设计，而不必过多地考虑具体的硬件结构。基于EDA技术的现场可编程门阵列(FPGA)电路,提出现场可编程门阵列(FPGA)是近年来迅速发展的大规模可编程专用集成电路(ASIC),在数字系统设计和控制电路中越来越受到重视。VHDL语言是电子设计的主流硬件描述语言,它更适合进行行为描述,这种方式使得设计者专注于电路功能的设计，而不必过多地考虑具体的硬件结构。

第 1 章 FPGA 概述

1.1 FPGA 的简介

FPGA(Field —Programmable Gate Array),即现场可编程门阵列,它是在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。

目前以硬件描述语言所完成的电路设计,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。

1.2 FPGA的应用

FPGA的应用可分为三个层面:电路设计,产品设计,系统设计。

1. 电路设计

连接逻辑,控制逻辑是 FPGA早期发挥作用比较大的领域也是 FPGA应用的基石。事实上在电路设计中应用 FPGA要求开发者要具备相应的硬件知识(电路知识)和软件应用能力(开发工具)。

2. 产品设计

把相对成熟的技术应用到某些特定领域开发出满足行业需要并能被行业客户接受的产品。这方面主要是 FPGA技术和专业技术的结合问题,重点在性能, FPGA技术在这个领域是一个实现手段, FPGA因为具备接口,控制,功能 IP,内嵌 CPU等特点有条件实现一个构造简单,固化程度高,功能全面的系统产品设计。

3. 系统级应用

系统级的应用是 FPGA与传统的计算机技术结合,实现一种 FPGA版的计算机系统如用 Xilinx V-4, V-5 系列的 FPGA实现内嵌 POWERPC CPU,然后再配合各种外围功能,这个平台上跑 LINUX 等系统这个系统也就支持各种标准外设和功能接口了,这对于快速构成 FPGA大型系统来讲是很有帮助的。

第 2 章 VHDL 硬件描述语言

2.1 VHDL 程序基本结构

一个相对完整的 VHDL 程序通常包含实体、结构体、配置、程序包和库 5 个部分。程序模板如下:

```

LIBRARY library name;
USE library name.package name.all;
ENTITY entity name IS
    GENERIC
        ( parameter name :string:=default value;
          parameter name :integer:=default value);
    PORT
        ( input name,input name: IN STD-LOGIC;
          Input vector name:IN STD_LOGIC_VECTOR( 、time、character、string、
integer 、real。

```

2.2.4 VHDL 顺序语句

VHDL 有如下六类基本顺序语句：赋值语句、流程控制语句、等待语句、子程序调用语句、返回语句、空操作语句。

赋值语句有信号赋值语句和变量赋值语句两种。流程控制语句共有五种：IF 语句、CASE 语句、LOOP 语句、NEXT 语句、EXIT 语句。

IF 语句根据条件句产生的判断结果：TRUE 或 FALSE，有条件地选择执行其后的顺序语句。CASE 语句根据满足的条件直接选择多项顺序语句中的一项执行。空操作语句不完成任何操作，它常用于 CASE 语句中，利用它来表示所余的不用条件下的操作行为——满足所有可能的条件。

2.2.5 VHDL 并行语句

结构体中的并行语句主要有七种：并行信号赋值语句、进程语句、块语句、条件信号赋值语句、元件例化语句（其中包括类属配置语句）生成语句、并行过程调用语句。

第 3 章 系统设计与仿真

3.1 系统介绍

交通灯控制器用于自动控制十字路口交通灯和计时器，指挥各种车辆和行人安全通行。

3.1.1 设计任务

设计一个十字路口交通控制器，方向分为东南西北四个方向。东西方向的红绿灯状态一样，南北方向的红绿灯状态一样。每个方向上，有四盏灯，分别是左

转灯、红灯、绿灯和黄灯。左拐灯亮表示左转车辆可以通行；红灯亮表示左转和直行车辆禁行；绿灯亮表示直行车辆和右转的车辆可以通行；黄灯亮表示左转和直行的车辆即将禁行；倒计时显示器用来显示允许通行或禁止通行的时间倒计时。

3.1.2设计要求

在十字路口东西方向和南北方向各设一组左转灯、；显示的顺序为：左转灯绿灯黄灯红灯。

在东西方向和南北方向各设一组倒计时显示器。倒计时只显示对应方向上的红灯倒计时和绿灯倒计时。其余的状态不显示倒计时。左转灯、红灯、绿灯和黄灯亮的时间分别是15秒、80秒、45秒、5秒。状态表如表3-1所示：

表 3-1 交通灯状态转换表

状态	S0	S1	S2	S3	S4	S5	
东 西 方 向	红	红	红	红 左 转	绿	黄	
亮 灯	15	45	5	15	45	5	
南 北 方 向	红 左 转	绿	黄	红	红	红	
亮灯	15	45	5	15	45	5	

3.2 系统设计仿真

对于交通灯控制器，可基于 Quartus II 软件，采用层次化混合输入方式进行设计，即顶层采用原理图设计，底层采用 VHDL 语言设计。

3.2.1顶层框图的设计

顶层原理图设计可以依据系统框图进行，有分频模块、控制模块、倒计时模块、红绿灯显示模块、译码模块、译码显示模块6 六部分组成，如图 3-1 所示：

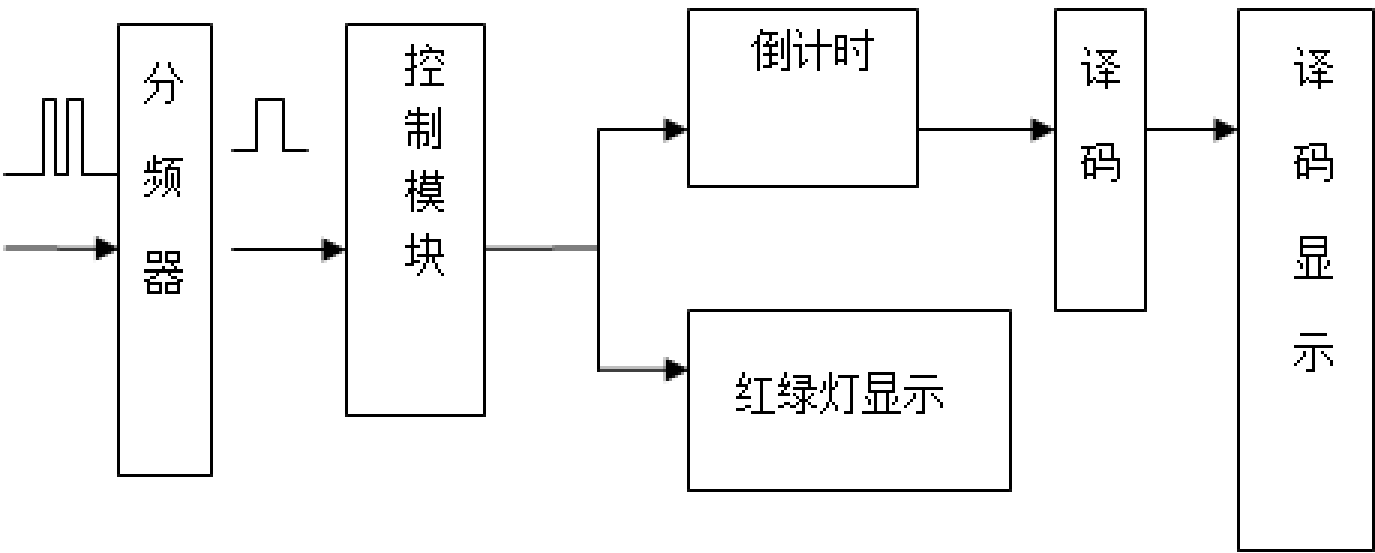


图 3-1 顶层原理图系统框图

3.2.2 时序状态图的设计

图 3-2 交通灯时序状态图

3.2.3 工程设计流程框图：

图 3-3 工程设计流程框图

3.2.4 芯片的选择

在这次毕业设计中，我们选用的 FPGA 芯片是 Altera 公司的 Cyclone（飓风）系列的芯片，型号为：EP1C3T144C8。它是基于 1.5V 的，采用 0.13um 全铜 SRAM 工艺、TQFP 封装，有 5980 个逻辑单元，2 个锁相环，20 个 M4K RAM 块，其中每个 RAM 为 4kbit，可以另加一位奇偶校验位。

3.2.5 各个模块的设计与仿真

1. 模块设计

该模块可将频率为 1KHZ 的脉冲波，经过分频变为频率为 1HZ 的脉冲波，这样我们就得到了周期为 1S 的脉冲波。分频器程序清单如下：

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity fenpinqi is
    port ( clk1khz:in std_logic;
          clk1hz:out std_logic);
end fenpinqi;
architecture a of fenpinqi is
begin
    process( clk1khz )
        variable q: integer:=0 ;
        variable r:std_logic:='0';
    begin
        if clk1khz'event and clk1khz='1' then
            if q=512 then r:=not r; q:=0;
            else q:=q+1;
```

```
        end if;

        clk1hz<=r;

        end if;

    end process;

end a;
```

分频器程序仿真结果如图 3-4 所示：

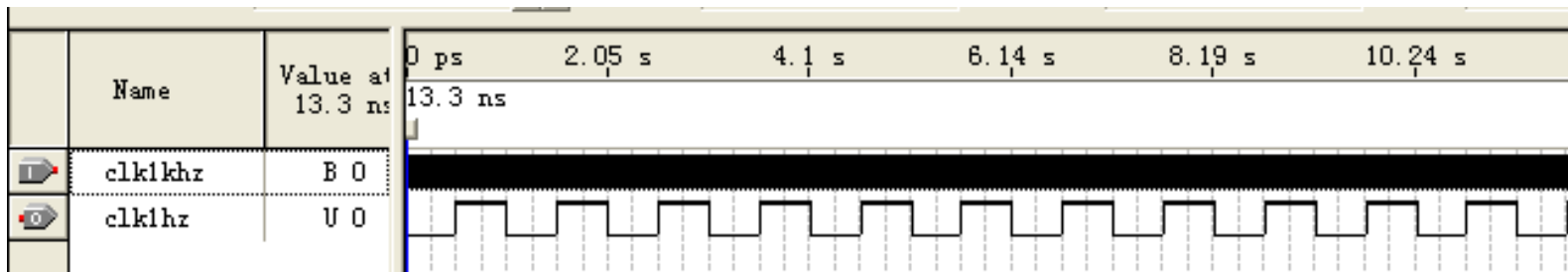


图 3-4 分频器程序仿真结果图

在分频器程序仿真结果图中，clk1khz 为输入脉冲信号，频率为 1khz；clk1hz 为输出脉冲信号，频率为 1hz。从图中，我们可以看出 clk1hz 的周期为 1s。这表明程序的确把 1khz 的信号分频成 1hz 的信号。

2. 模块设计

control 模块可根据 clk 的脉冲信号输出 S 的值，下游模块根据 S 的值来决定红绿灯的状态。同时对倒计时的信号赋初值。control 模块程序清单：

```
library ieee;

use ieee.std_logic_1164.all;

use ieee.std_logic_unsigned.all;

entity control is

    port( clk:in std_logic;

          s:out std_logic_vector(2 downto 0);

          loadat,loadbt:out std_logic;

          at,bt:out integer range 0 to 80);

end control;

architecture a of control is

    signal t:integer;

begin

    process(clk)
```

```
begin
    if clk'event and clk='1' then
        loadat<='0';loadbt<='0';
        if t=0
            at<=80;    loadat<='1';
            bt<=45;    loadbt<='1';

            bt<=80;    loadbt<='1';

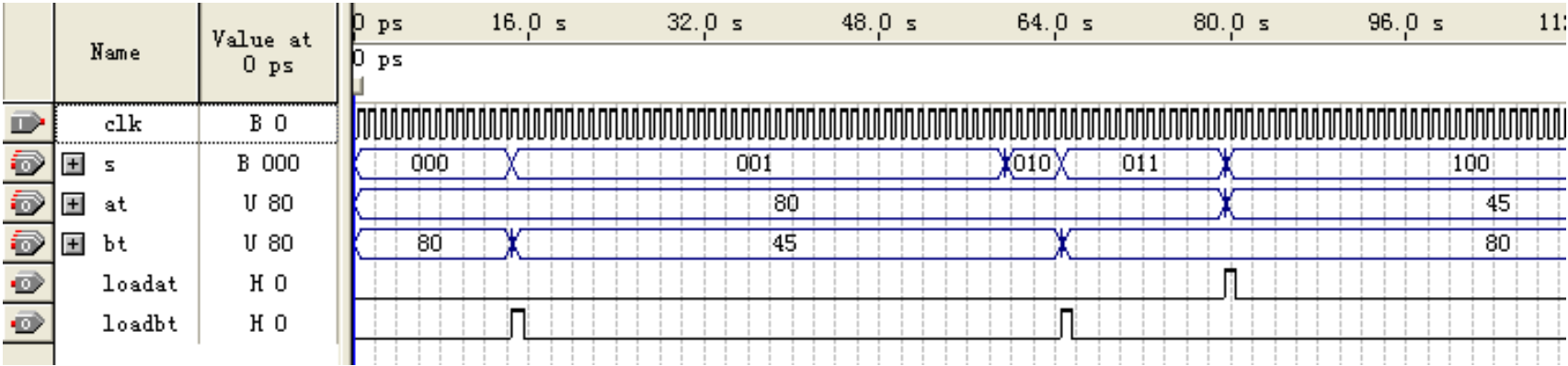
        elsif t=80 then
            loadat<='1';

        elsif t=125 then

        end if;
        t<=t+1;
    end if;
    if t=130 then    t<=0;    end if;
end process;

end a;
```

control 3-5 所示:



模块程序仿真结果图

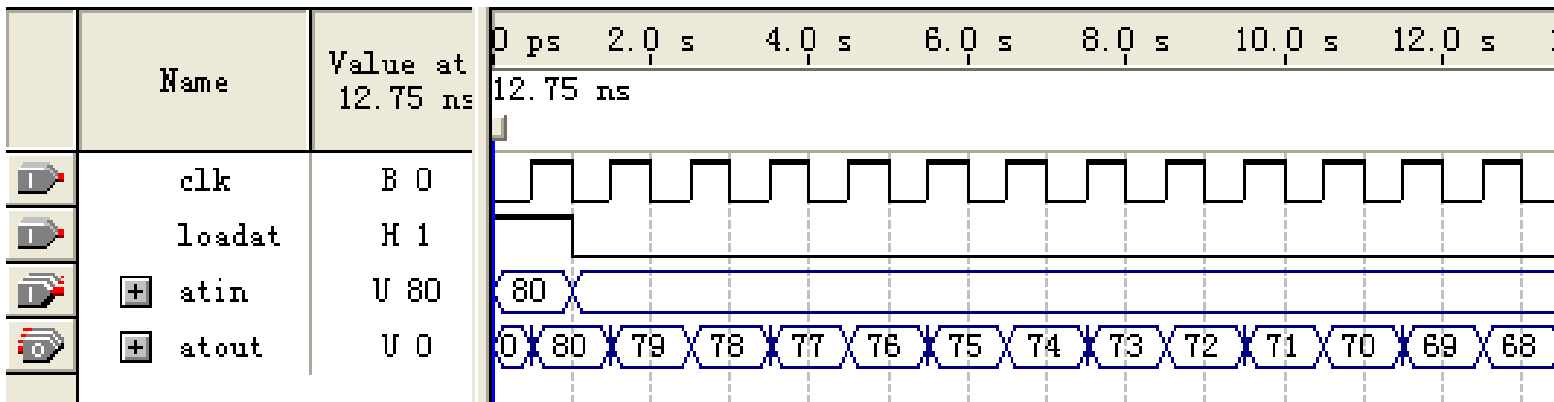
在 模块程序仿真结果图中，只有一个输入信号 **clk**，它是一个频率为 1hz 的时钟脉冲信号，输出信号有 3 个分别是 **s**、**at**、**bt**。程序中，通过变量 **t** 的值来确定输出信号 **s** 的值，并由此确定将要赋得倒计时的初值，然后赋给 **at**，**bt**。在每次给 **at**，**bt** 赋值时，都会使另一个输出信号 **loadat**，**loadbt** 变成高电平，从而可以触发下一模块的装载。

3. daojishi 模块设计

该模块用来接收 **control** 模块的输出作为初值，并进行减一操作，最后把减一得结果输出。程序里每检测到一个脉冲 **clk**，就进行一次减一操作。**atdaojishi** 模块

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity atdaojishi is
    port ( clk,loadat:in std_logic;
           atin:in integer range 0 to 80;
           atout:out  integer range 0 to 80);
end atdaojishi;
architecture a of atdaojishi is
    signal countat:integer range 0 to 80;
    begin
        process
            begin
                wait until rising_edge(clk);
                if loadat='1' then countat<=atin;
                    else countat<=countat-1;
                end if;
            end process;
            atout<=countat;
        end a;
```

atdaojishi 模块程序仿真结果如图 3-6 所示：



模块程序仿真结果图

在 atdaojishi 模块程序仿真结果图中，clk 为 1hz 的输入信号，loadat 为 at

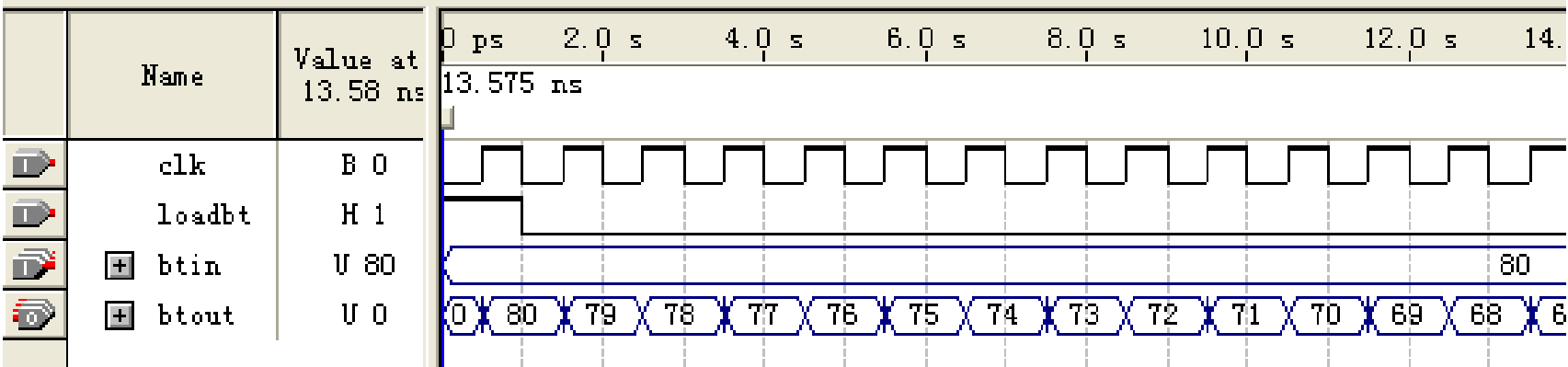
为高电平时，将输入信号 **atin** 的值装载到计数器中，开始执行减一操作，每接受一次 **clk**，执行一次，并将结果输出。

4. daojishi 模块设计

该模块用来接收 **control** 模块的输出作为初值，并进行减一操作，最后把减一得结果输出。程序里每检测到一个脉冲 **clk**，就进行一次减一操作。**btdaojishi** 模块程序清单：

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity btdaojishi is
    port ( clk,loadbt:in std_logic;
           btin:in integer range 0 to 80;
           btout:out integer range 0 to 80);
end btdaojishi;
architecture a of btdaojishi is
    signal countbt:integer range 0 to 80;
    begin
        process
        begin
            wait until rising_edge(clk);
            if loadbt='1' then countbt<=btin;
                else countbt<=countbt-1;
            end if;
        end process;
        btout<=countbt;
    end a;
```

btdaojishi 模块程序仿真结果如图 3-7 所示：



模块程序仿真结果图

模块程序仿真结果图中，clk 为 1hz 的输入信号，loadbt 为 bt 计数器的装载信号，当 loadbt 为高电平时，将输入信号 btin 的值装载到计数器中，开始执行减一操作，每接受一次 clk ，执行一次，并将结果输出。

5. Main 模块设计

```
library ieee;

use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity main is
    port (
        s:in std_logic_vector(2 downto 0);
        lrgy:out std_logic_vector(7 downto 0));
end main;

architecture a of main is
begin
    process(s)
        variable b:std_logic_vector(7 downto 0);
    begin
        case s is
```

```
        end case;

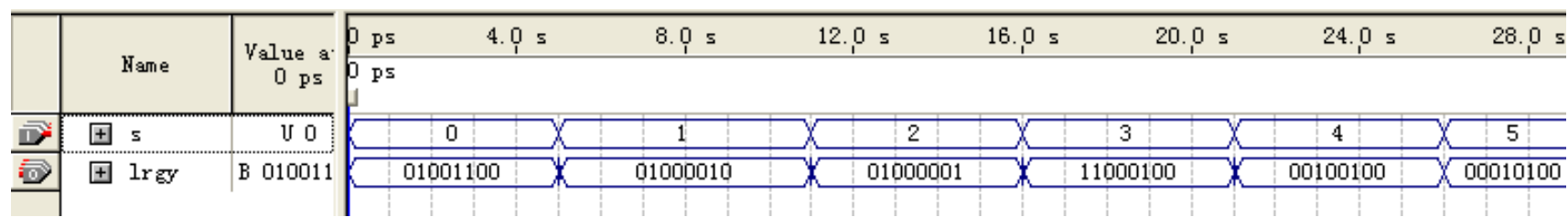
        lrgy<=b;

    end process;

end a;

main
```

3-8所示：



模块程序仿真结果图

Main 模块可根据输入的 S 信号，输出对应的红绿灯信号。直接改变并驱动红绿灯的亮和灭。

在 main 模块程序仿真结果图中，S 为输入信号，用来表明整个红绿灯的状态（一共 6 个状态：S0~S5）；lrgy 为输出信号，代表整个红绿灯的亮和灭的情况，灯的排序从高位到低位排列，方向上是先东西方向，后南北方向。也就是说，序列为：东西方向左转灯、红灯、绿灯、黄灯，然后南北方向左转灯、红灯、绿灯、黄灯。从图中我们可以看出 S 的状态从“000”到“101”，共六个状态，每一个状态都对应相应的红绿灯亮灭的状态，仿真结果完全正确，符合预期。

6. 码模块设计

译码程序将输入的用来显示的两位倒计时数值，分解成四个整形的数值，以供后面的译码显示模块继续译码显示。译码程序清单如下：

```
library ieee;

use ieee.std_logic_1164.all;

use ieee.std_logic_unsigned.all;

entity to8421bcd is

    port( at,bt:in integer range 0 to 80;

          aout1,aout2,bout1,bout2:out integer range 0 to 9);

end to8421bcd;

architecture a of to8421bcd is

begin
```

```
process  (at, bt)
begin
    if at >= 70 then
        aout1 <= 7;
        aout2 <= at - 70;
    elsif  at >= 60 then
        aout1 <= 6;
        aout2 <= at - 60;
    elsif  at >= 50 then
        aout1 <= 5;
        aout2 <= at - 50;
    elsif  at >= 40 then
        aout1 <= 4;
        aout2 <= at - 40;
    elsif  at >= 30 then
        aout1 <= 3;
        aout2 <= at - 30;
    elsif  at >= 20 then
        aout1 <= 2;
        aout2 <= at - 20;
    elsif  at >= 10 then
        aout1 <= 1;
        aout2 <= at - 10;
    elsif  at >= 0 then
        aout1 <= 0;
        aout2 <= at;
    end if;
    if bt >= 70 then
        bout1 <= 7;
        bout2 <= bt - 70;
```

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：<https://d.book118.com/138125007007006106>