中山大学课程设计论文

交通灯信号控制器的设计

学　　院：数据科学与计算机学院

专　　业：软件工程（移动信息工程）

指导教师：李国桢、保延翔

完成时间：2016 年 12 月 24 日

数字设计与计算机体系结构课程设计论文
交通灯信号控制器的设计

组员：郑晓如
郑海君
赵钰莹
张燕梅

摘要

城市道路交叉口是城市道路网络的基本节点，也是网络交通流的瓶颈。目前，大部分无控制交叉口都存在高峰小时车流混乱、车速缓慢、延误情况严重、事故多发、通行能力和服务水平低下等问题。特别是随着城市车流量的快速增长，城市无控制道路交叉口的交通压力越来越大。因此，做好基于 EDA 技术平台的交叉口信号控制设计是缓解交通阻塞、提高城市道路交叉口车辆通行效率的有效方法。交通信号控制的目的是为城市道路交叉口(或交通网络)提供安全可靠和有效的交通流，通常最为常用的原则是车辆在交叉口的通过量最大或车辆在交叉口的延误最小。

有限状态机是指输出取决于过去输入部分和当前输入部分的时序逻辑电路。一般来说，除了输入部分和输出部分外，有限状态机还含有一组具有“记忆”功能的寄存器，这些寄存器的功能是记忆有限状态机的内部状态，它们常被称为状态寄存器。在有限状态机中，状态寄存器的的下一个状态不仅与输入信号有关，而且还与该寄存器的当前状态有关，因此有限状态机又可以认为是组合逻辑和寄存器逻辑的一种组合。其中，寄存器逻辑的功能是存储有限状态机的内部状态；而组合逻辑又可以分为次态逻辑和输出逻辑两部分，次态逻辑的功能是确定有限状态机的下一个状态，输出逻辑的功能是确定有限状态机的输出。

本次课程设计正是基于有限状态机的设计，其基本原理是在合适的时钟信号的控制下，使主干道与支道的红黄绿灯循环显示，在 vivado 的开发环境下用硬件描述语言 Verilog 编辑文本程序并进行波形仿真验证其正确性，使其最终的显示结果符合设计要求。

关键词：有限状态机、verilog、交通灯、vivado

一、选题背景

在数字设计与计算机体系结构的课程中，我们学习了从计算机组织和设计到更细节层次的内容功能。而有限状态机是时序逻辑电路的核心部分，作为一名数据科学与计算机学院的学生，只有学好有限状态机的知识，才能对时序逻辑电路有更好的了解，才能使自己的知识体系更加完整。而且，设计交通灯信号控制器与设计一个原件相比，更能锻炼一个团队对较大的任务的驾驭能力，所以我们选择设计一个交通灯信号控制器。

二、技术路线

本次的课程设计我们选择使用 Verilog 语言来完成。原因有以下：A、老师建议使用 HDL 语言；B、硬件描述语言能在各个抽象级别上进行设计，比画原理图方便；C、使用硬件描述语言设计起来使脉络显得更加清晰易懂。硬件描述语言主要两种，其中一种是 VHDL，另一

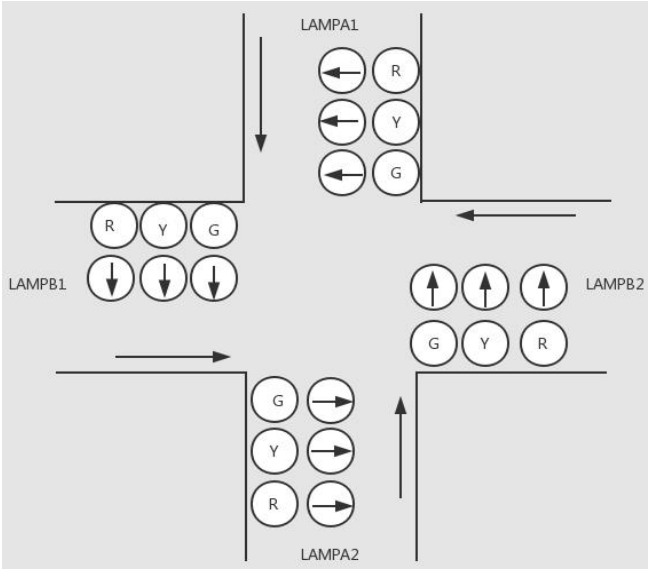
种就是 Verilog。其实对两种 HDL 语言，我们的课程都有所涉及。针对小组成员情况，我们选择使用 Verilog。因为 VHDL 比较严谨，更适合设计复杂的设计，而 Verilog 语言很灵活，与 C 语言很相近，学习起来很简单，能在比较短的时间里面掌握，最重要的是跟我们的编程习惯比较相近。

在之前的实验课中，我们都是使用 vivado 2016.3 来完成的，因此对 vivado 2016.3 软件的使用较为熟悉，而且 vivado 2016.3 功能强大，所以我们选择在 vivado 2016.3 环境下进行设计与仿真。

三、实现过程

1、设计内容

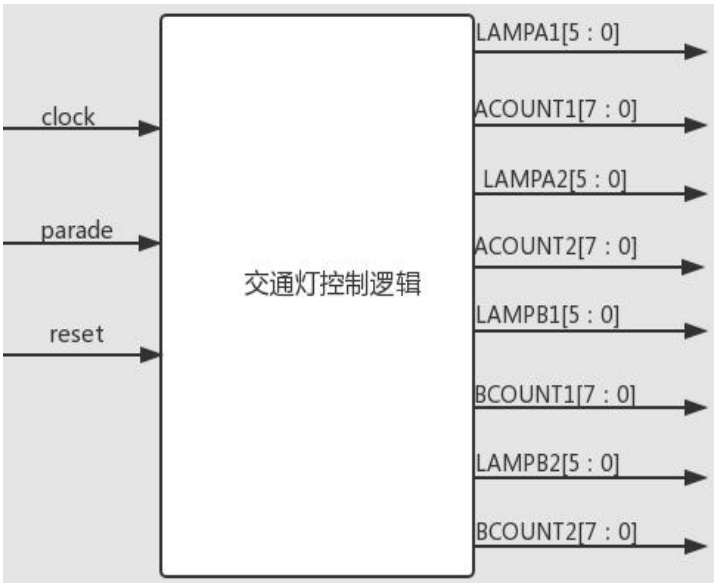
某十字路口，在 A 方向和 B 方向上各有两组信号灯，每组六盏灯，分别是直行绿（G）、直行黄（Y）、直行红（R）、左弯绿、左弯黄和左弯红。四组信号灯共计 24 盏灯，另外设置了四组倒计时显示牌，倒计时显示使用 2 位的八段码实现。A 方向上的两组信号灯显示情况一样，同样，B 方向上的两组信号灯显示情况一样。交通信号灯控制示意图如图（1）所示。为保证交通安全，在出现故障时信号灯要倒向安全的一侧，因此设计时要保证不能在红灯时显示绿色，在绿灯时显示红色。还应考虑一种特殊情况，即当有乐队等人群以分散的模式在 A 方向漫步时，通过按键进入游行模式，此时保持 A 方向上的灯是绿色，直到游行结束，通过按键返回正常模式。



图（1）交通信号灯控制示意图

2、引脚简介

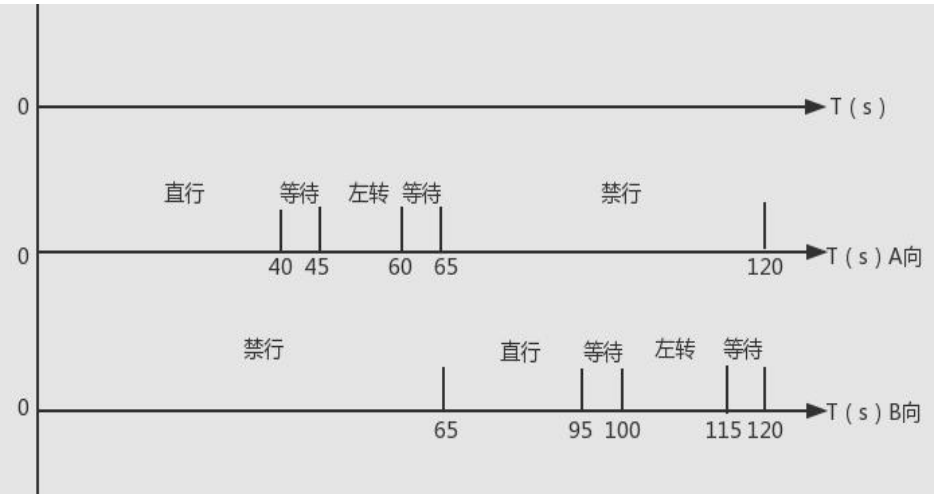
交通信号灯控制器的引脚情况如图（2）所示，输入有三个信号，clock 为时钟信号，时钟频率为 1HZ，reset 为复位信号，当为高电平时系统开始工作，parade 为判断信号，当为高电平时进入游行模式，当为低电平时为正常模式；输出有八个信号：LAMPA1(LAMPA2)、LAMPB1(LAMPB2)、ACOUNT1(ACOUNT2)、BCOUNT1(BCOUNT2)。



图（2）交通信号灯控制器引脚图

3、交通灯控制器的状态转换描述

在设计中，对 A 和 B 方向分别编写状态机代码，但是注意 AB 两方向的交通是相关的，所以在时钟的统一步调下，AB 方向的红绿灯时间一定要配合好，使得交通顺畅有序，正常状态下，AB 方向的时间配合如图（3）所示，状态的转换按照图中所示时间节点所示，整个周期为 120 秒。



图（3） 交通信号灯 AB 方向时间匹配图

正常状态下，同一方向的信号灯亮灭相关性如下所示：

• 方向 A 直行红灯亮，方向 B 要么直行绿灯亮，要么直行黄灯亮，要么直行红灯亮并且左转绿灯亮；

• 方向 A 直行绿灯亮，方向 B 直行红灯亮；

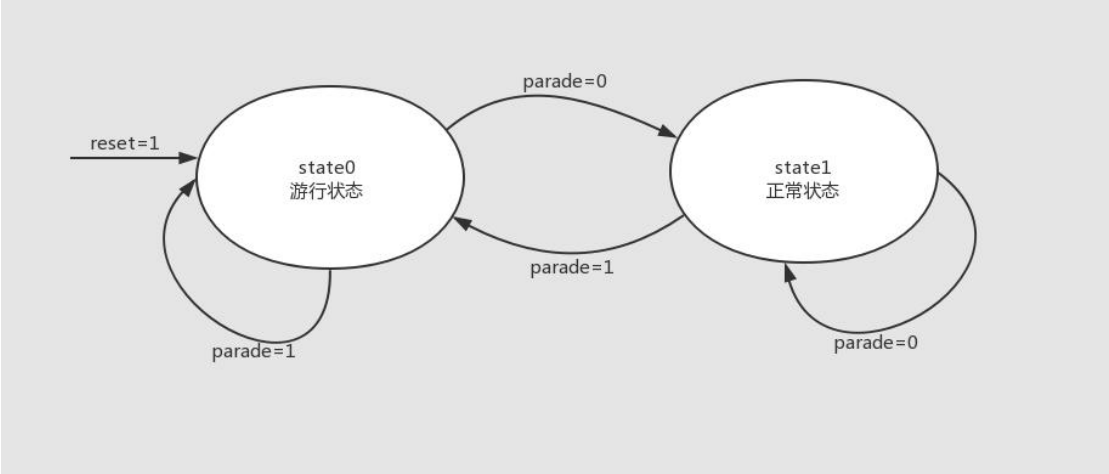
• 方向 A 直行红灯亮并且左转绿灯亮，方向 B 直行红灯亮；

• 方向 A 直行黄灯亮，方向 B 直行红灯亮。

4、 交通灯控制器的状态转移图

（1）游行状态下

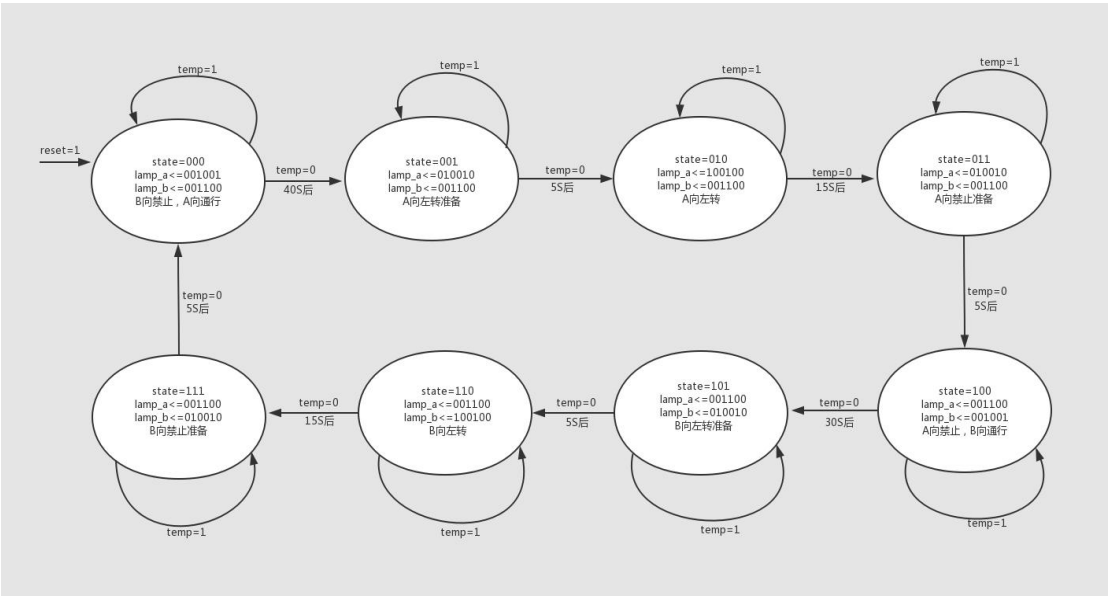
如图（4）所示，为交通灯信号控制器在游行状态与正常状态下的转移图，当输入信号 parade=1 时，无论交通灯处于何种模式，都将进入 A 向直行（即游行状态），直到 parade=0（即游行结束），则进入正常模式。



图（4）交通灯信号控制器在游行状态与正常状态下的转移图

（2）正常状态下

如图（5）所示，为正常状态下交通灯信号控制器 AB 方向的状态转移图，该交通灯信号控制器一共有 8 个状态，为了方便代码实现，我们又将该状态转换图分解为两个较为简单的状态转换图，分别为交通灯信号控制器 A 方向状态转换图（图（6））和 B 方向状态转换图（图（7））。



图（5）交通灯信号控制器 AB 方向的状态转移图

下面是 A 方向状态机（图（6））的运行过程，共有五个工作状态，工作情况如下：

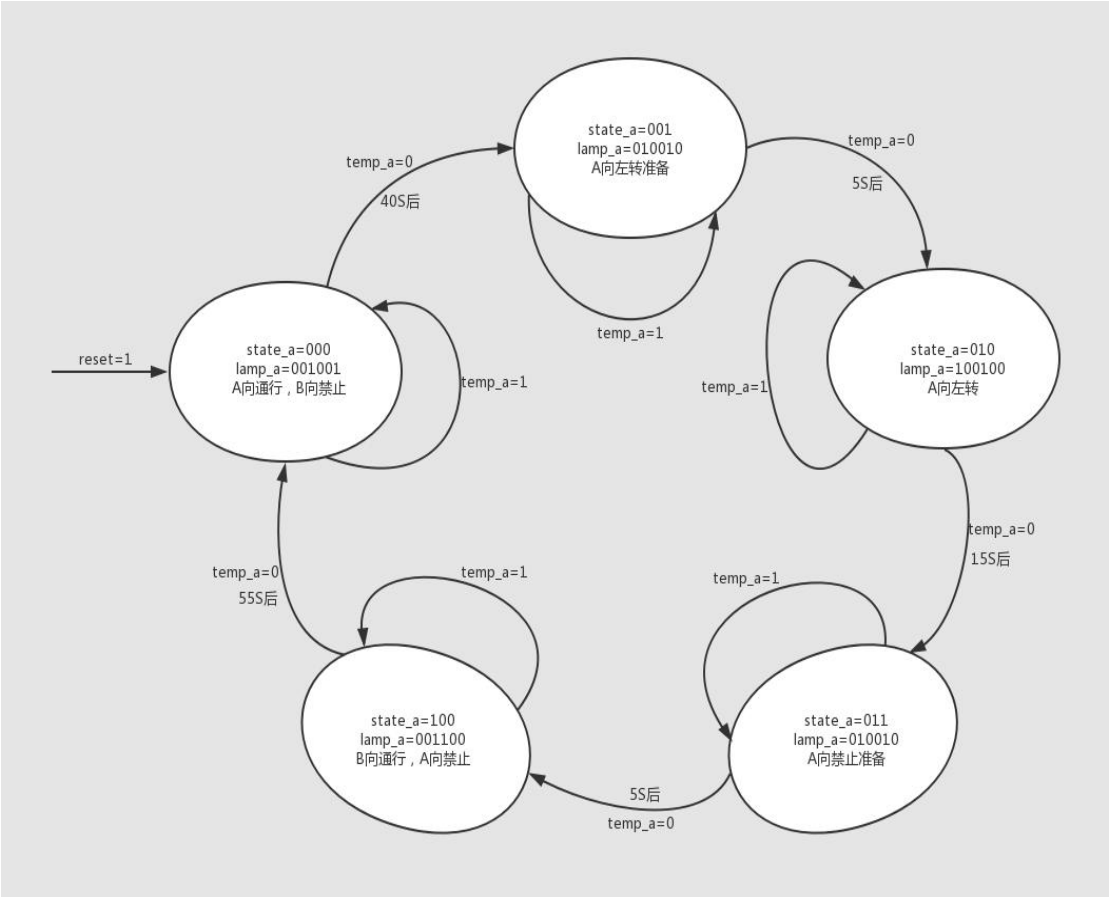
state0：A 向通行，此时 A 向直行绿灯亮，并且左转红灯亮；B 向禁行，此时 B 向直行红灯亮并且左转红灯亮，倒计数 40S 后进入 state1；

state1：A 向左转准备状态，此时 A 向黄灯亮，并且左转黄灯亮；B 向禁行，此时 B 向直行红灯亮并且左转红灯亮，倒计时 5S 后进入 state2；

state2：A 向左转状态，此时 A 向直行红灯亮，并且左转绿灯亮；B 向禁行，此时 B 向直行红灯亮并且左转红灯亮，倒计时 15S 后进入 state3；

state3：A 向禁止准备状态，此时 A 向直行黄灯亮，并且左转黄灯亮；B 向禁行，此时 B 向直行红灯亮并且左转红灯亮，倒计时 5S 后进入 state4；

state4：A 向禁止状态，此时 A 向直行红灯亮，并且左转红灯亮；B 向直行，此时 B 向绿灯亮并且左转红灯亮，倒计时 55S 后返回 state0。



图（6）交通灯信号控制器 A 方向的状态转移图

下面是 B 方向状态机（图（7））的运行过程，共有五个工作状态，工作情况如下：

state0：B 向通行，此时 B 向直行绿灯亮，并且左转红灯亮；A 向禁行，此时 A 向直行红灯亮并且左转红灯亮，倒计数 30S 后进入 state1；

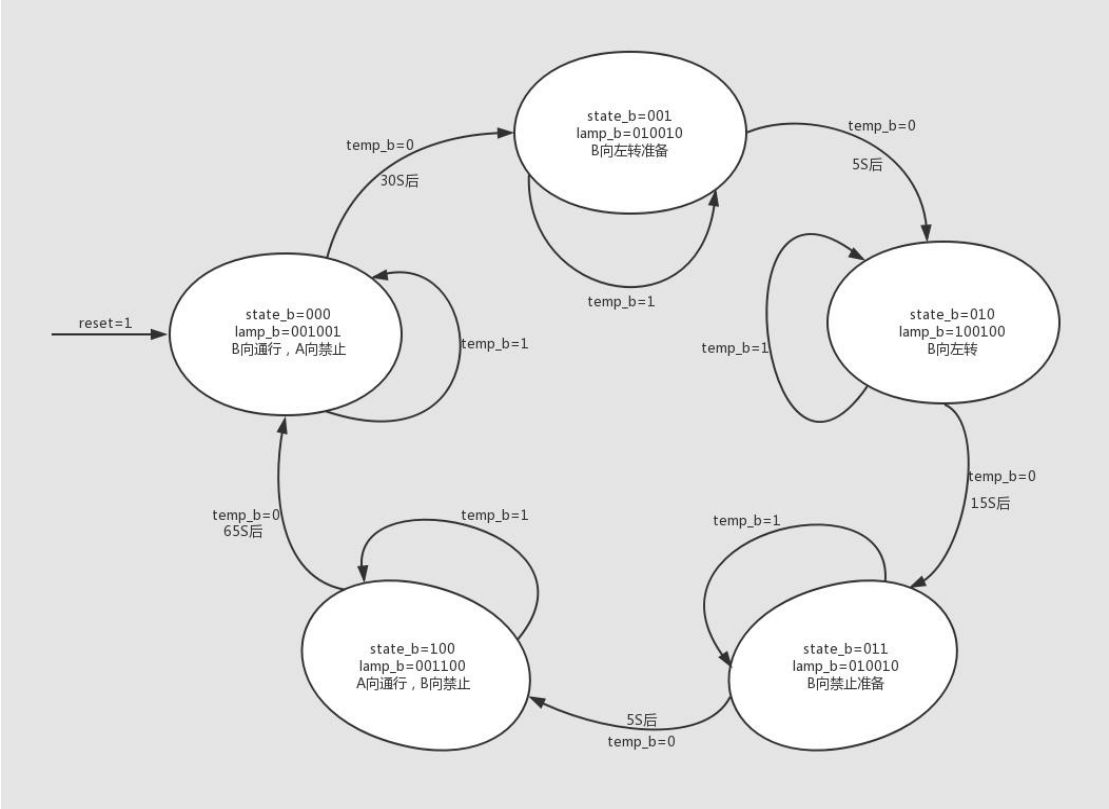
state1：B 向左转准备状态，此时 B 向黄灯亮，并且左转黄灯亮；A 向禁行，此时 A 向直行红灯亮并且左转红灯亮，倒计时 5S 后进入 state2；

state2：B 向左转状态，此时 B 向直行红灯亮，并且左转绿灯亮；A 向禁行，此时 A 向直行红灯亮并且左转红灯亮，倒计时 15S 后进入 state3；

state3：B 向禁止准备状态，此时 B 向直行黄灯亮，并且左转黄灯亮；A 向禁行，此时 A

向直行红灯亮并且左转红灯亮，倒计时 5S 后进入 state4；

state4：B 向禁止状态，此时 B 向直行红灯亮，并且左转红灯亮；A 向直行，此时 A 向绿灯亮并且左转红灯亮，倒计时 65S 后返回 state0。



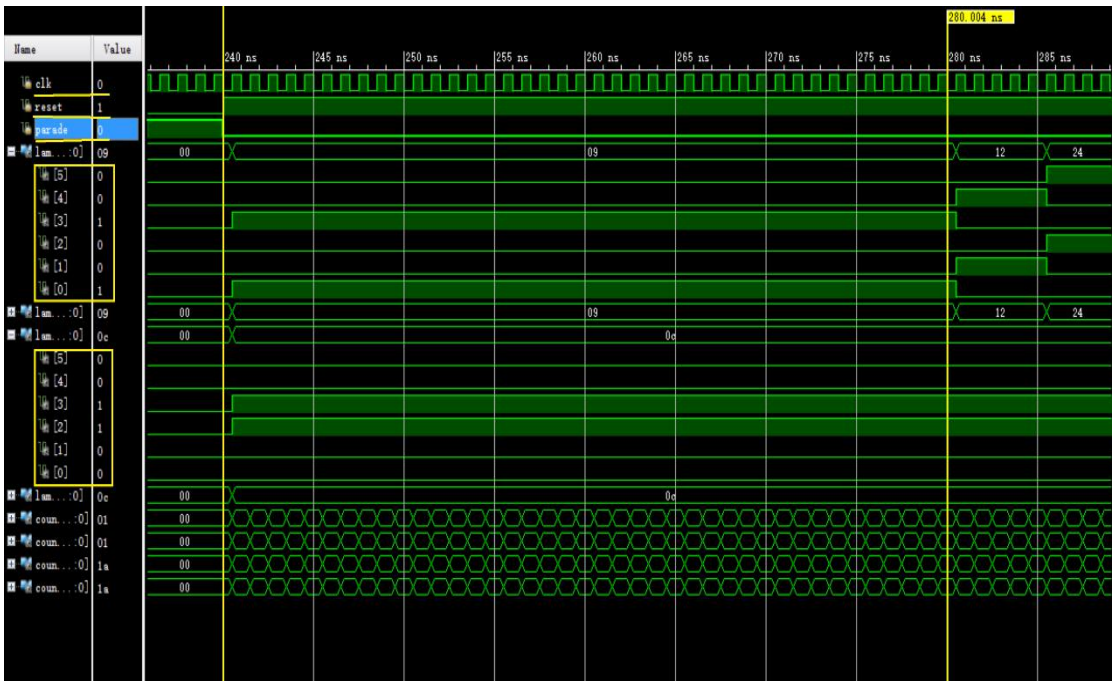
图（7）交通灯信号控制器 B 方向的状态转移图

四、仿真波形分析

注：由于 vivado 的仿真波形时间长度局限在 1000 000ps，如果我们使用代码中原来周期为 1s 的 clk，则会导致波形无法全部显示，所以我们便在模拟波形写 Testbench 的 top 文件时，将 clk 由 1ns 为周期来代替，因此在以下的波形中，1ns 均代表实际的 1s。波形显示的不足之处在于，我们无法通过 button 控制 reset 和 parade 两个输入信号，所以只能选择让其作为有周期的时钟信号输入，在观察波形图的时候，要注意，只有当 reset=1 时，系统才正常工作。

1、当 parade = 0 时，是非游行状态，交通灯正常工作

（1.1）State 000 时的 lamp_a 和 lamp_b 的显示情况：

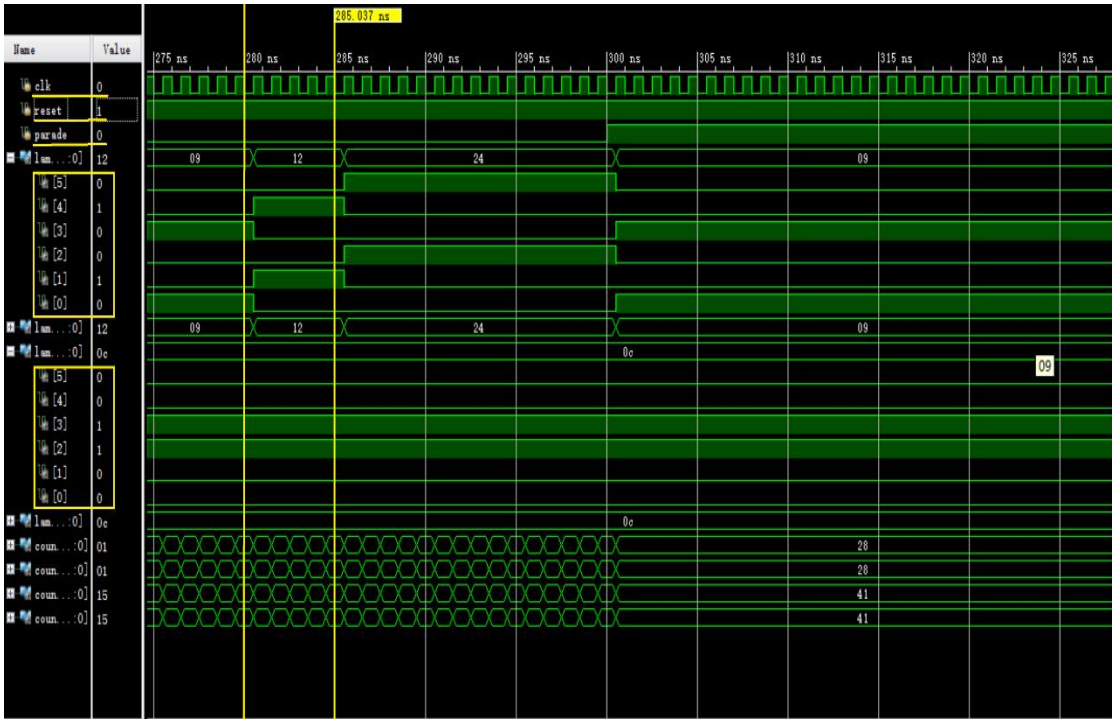


State 000 是 A 通行，B 禁行。应该是 A 向直行绿灯和左转红灯亮，B 向直行红灯和左转红灯亮（由左边黄色方框可看出）。

（1.2）State 000 时的 count_a（A 路口交通灯转变到下一状态还需倒计时的秒数）和 count_b 的初值：

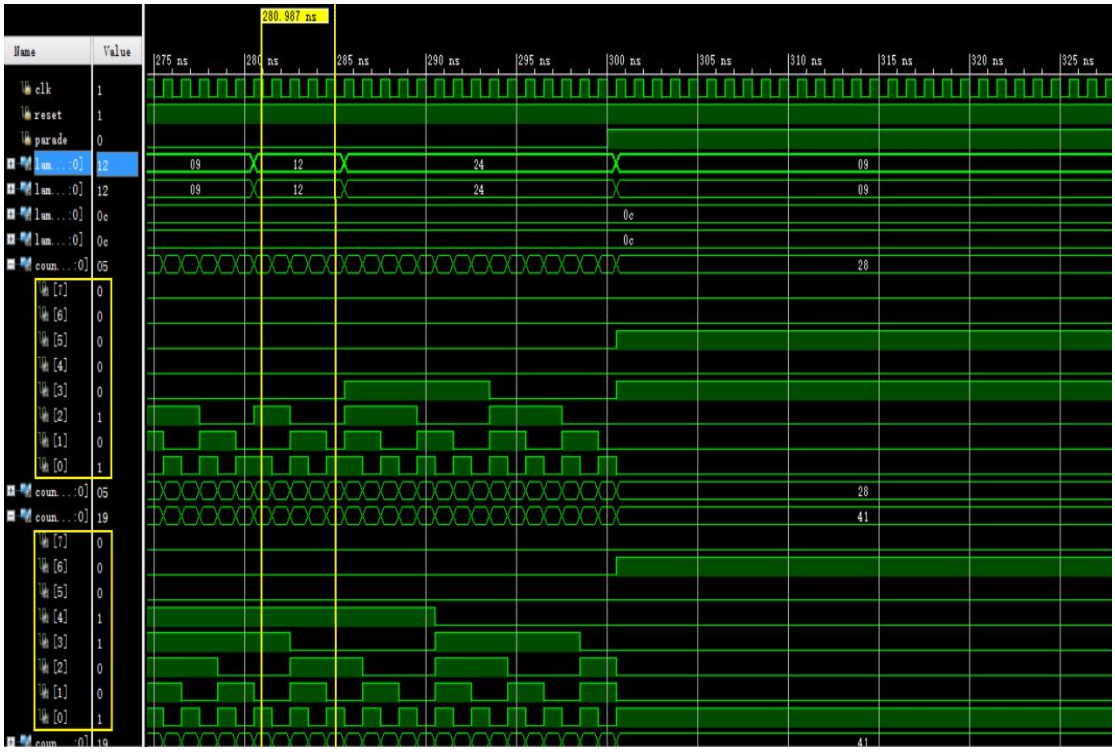
count_a = 2^5+2^3 = 40 (A 方向：由“A 向通行”转变到“A 向左转准备”需要 40s)，
count_b = 2^6+2^1 = 65（B 方向：由“B 向禁行”转变到“B 向通行”需要 65s ）。

（2.1）State 001 时的 lamp_a 和 lamp_b 的显示情况：



State 001 是 A 向左转准备，B 禁行。应该是 A 向直行黄灯和左转黄灯亮，B 向直行红灯和左转红灯亮（由左边黄色方框可看出）。

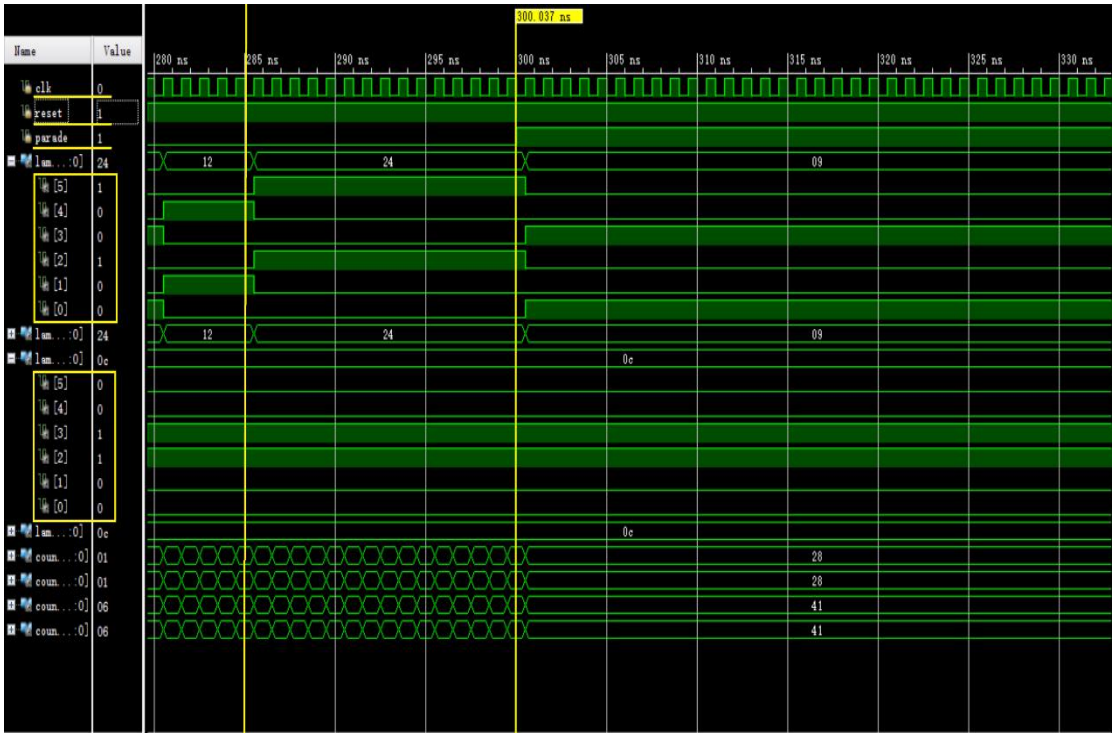
（2.2）State 001 时的 count_a（a 路口交通灯转变到下一状态还需倒计时的秒数）和 count_b 的初值：



count_a = 5 (A 方向：由“A 向左转准备”转变到“B 向左转”需要 5s)，

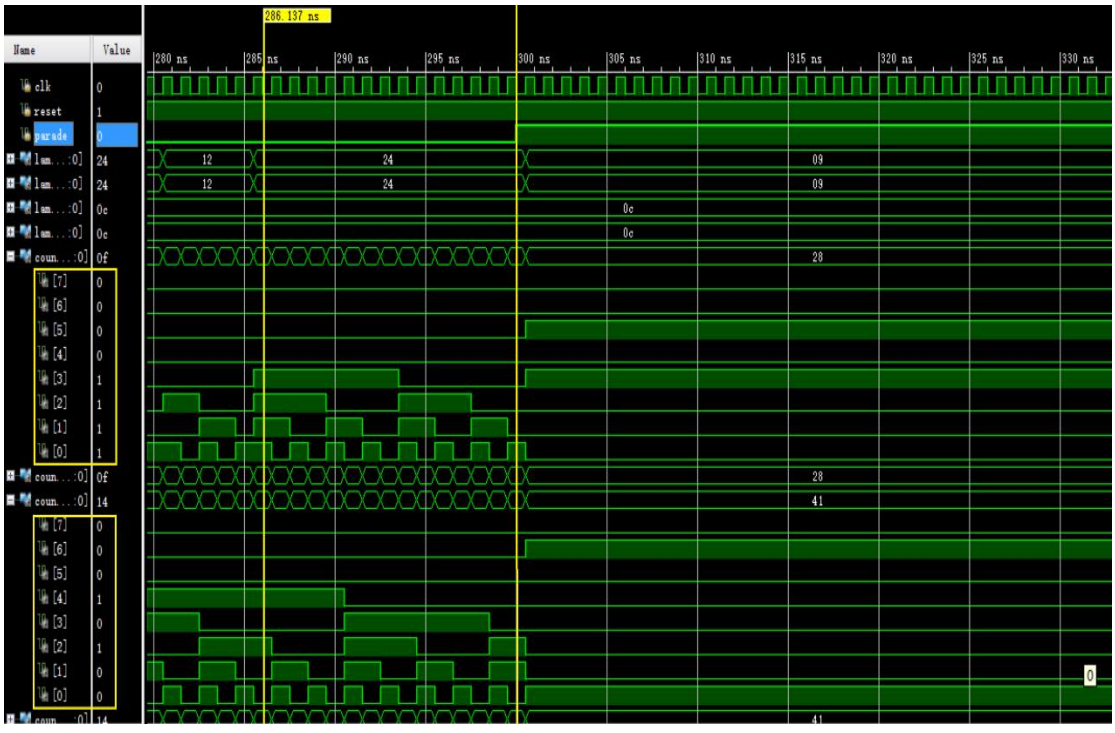
count_b = 25 = 65-40（B 方向：由“B 向禁行”转变到“B 向通行”需要 65-40 = 25s ）。

（3.1）State 010 时的 lamp_a 和 lamp_b 的显示情况：



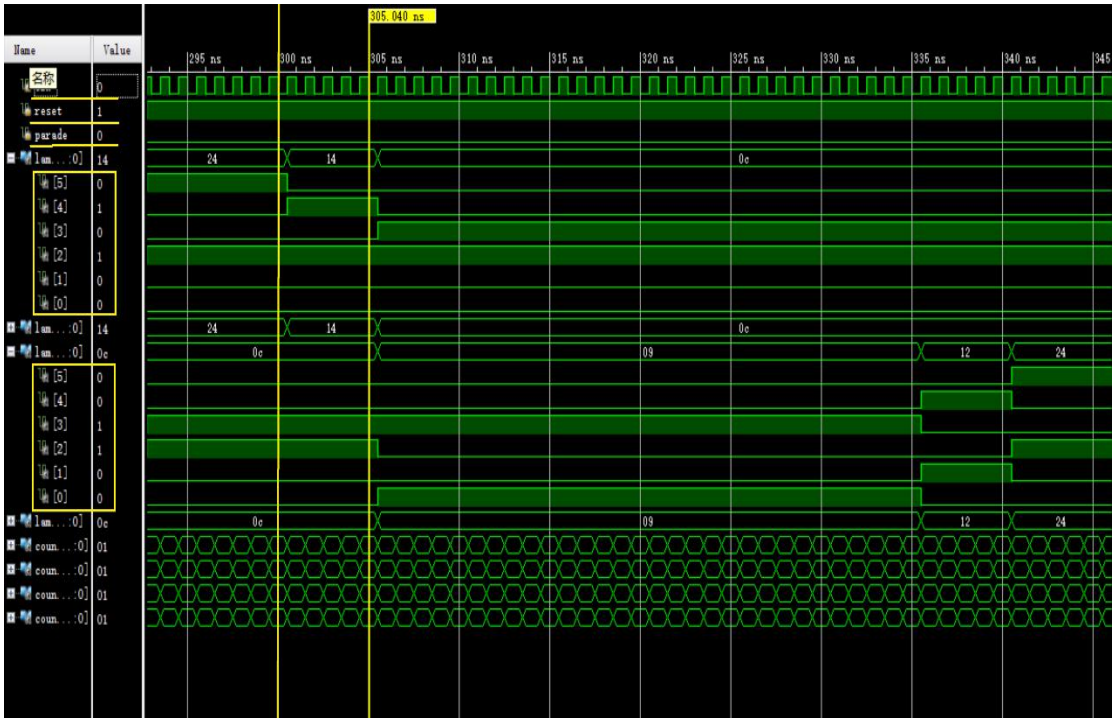
State 010 是 A 左转，B 禁行。应该是 A 向直行红灯和左转绿灯亮，B 向直行红灯和左转红灯亮（由左边黄色方框可看出）。

（3.2）State 010 时的 count_a（a 路口交通灯转变到下一状态还需倒计时的秒数）和 count_b 的初值：



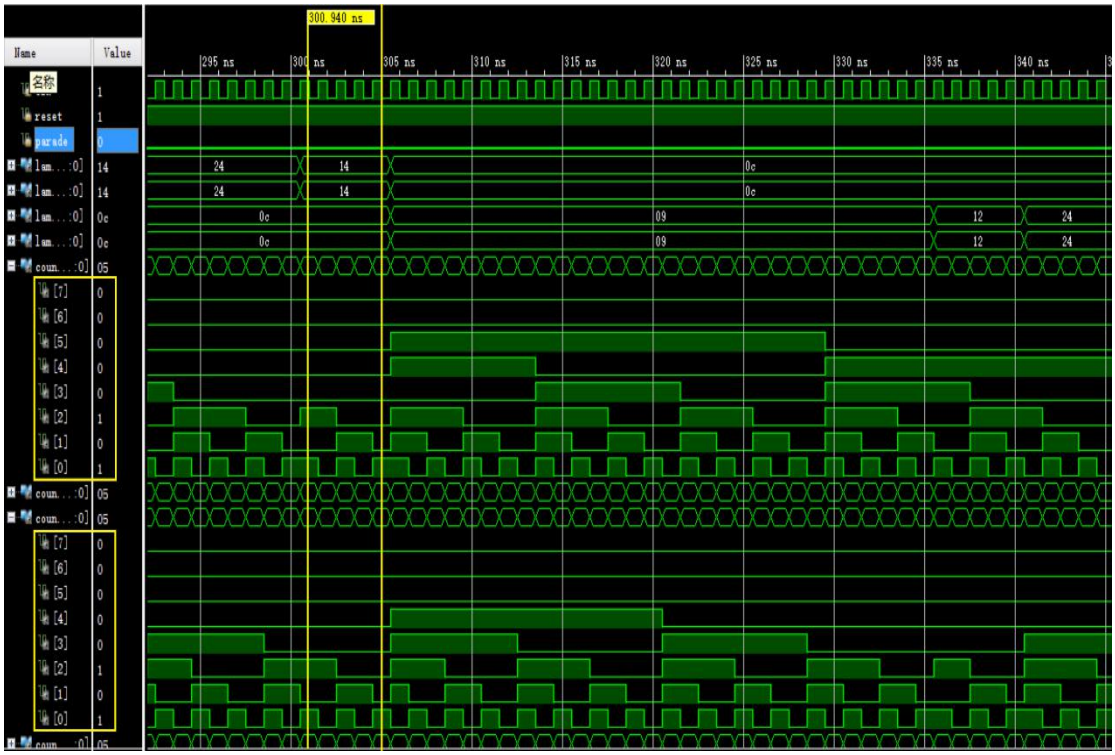
count_a = 15 (A 方向：由“A 向左转”转变到“A 向禁止准备”需要 15s)，
count_b = 20=25-5（B 方向：由“B 向禁行”转变到“B 向通行”需要 20s ）。

（4.1）State 011 时的 lamp_a 和 lamp_b 的显示情况：



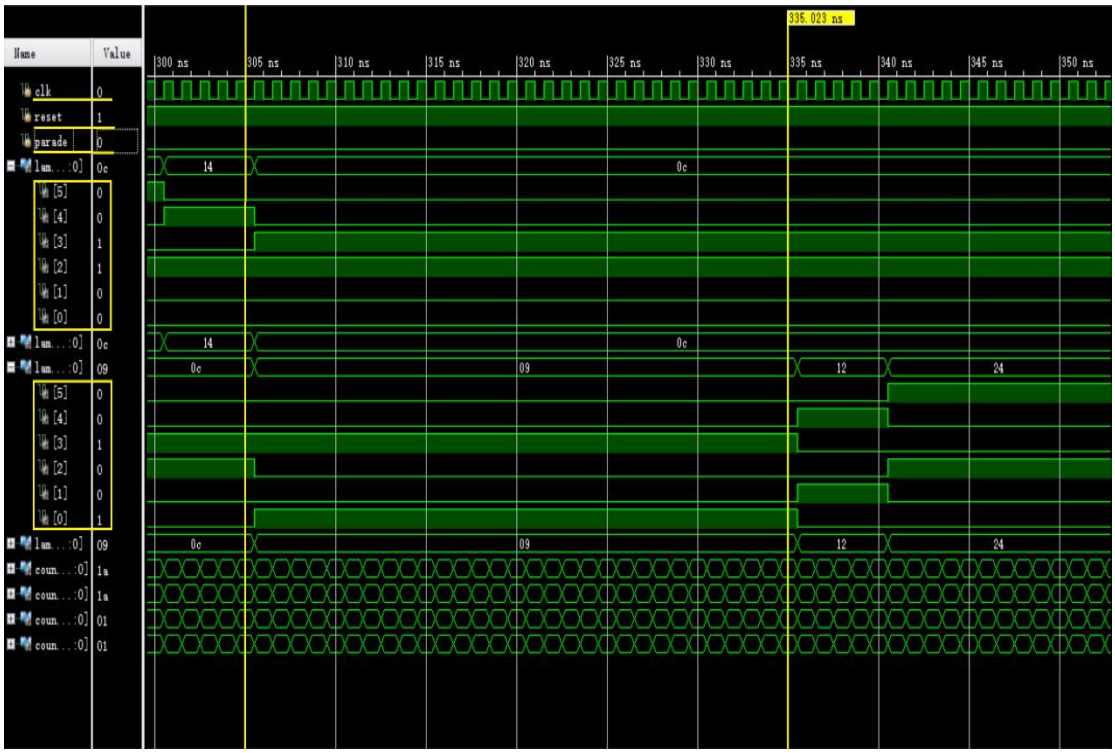
State 011 是 A 向禁止准备，B 禁行。应该是 A 向直行红灯和左转黄灯亮，B 向直行红灯和左转红灯亮（由左边黄色方框可看出）。

（4.2）State 011 时的 count_a（a 路口交通灯转变到下一状态还需倒计时的秒数）和 count_b 的初值：



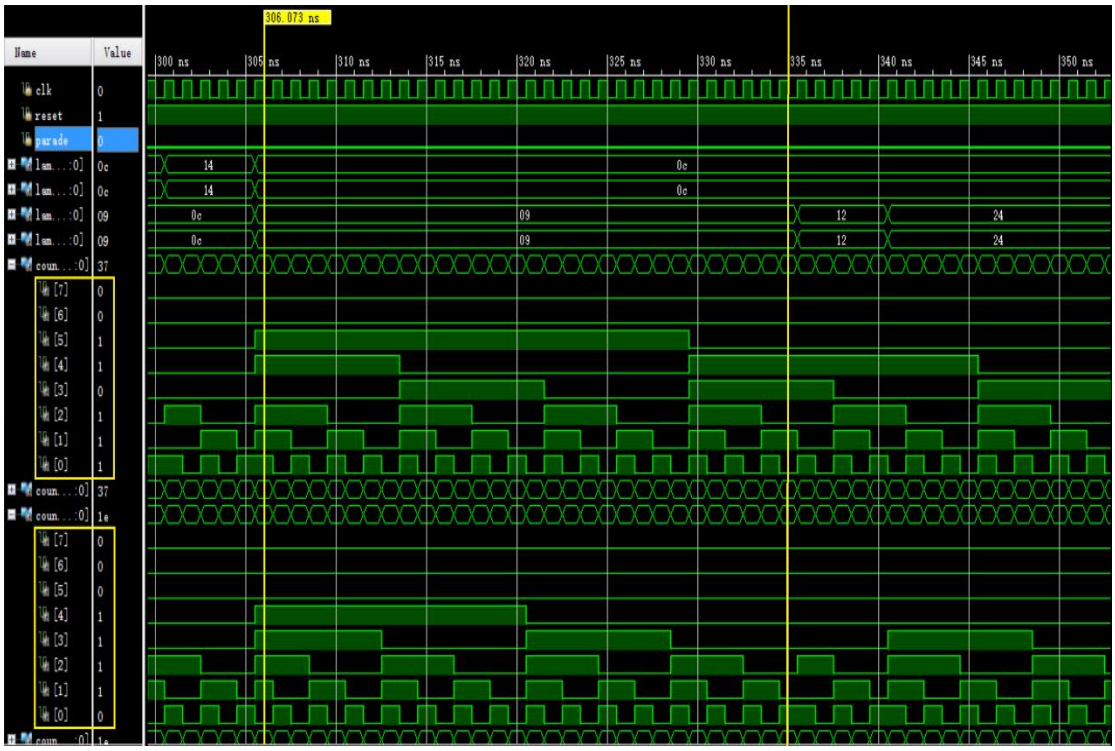
count_a = 5 (A 方向：由“A 向禁止准备”转变到“A 向禁止，B 向通行”需要 5s)，
count_b = 5（B 方向：由“B 向禁行”转变到“B 向通行”需要 20-15 = 5s ）。

（5.1）State 100 时的 lamp_a 和 lamp_b 的显示情况：



State 100 是 A 向禁行，B 向通行。应该是 A 向直行红灯和左转红灯亮，B 向直行绿灯和左转红灯亮（由左边黄色方框可看出）。

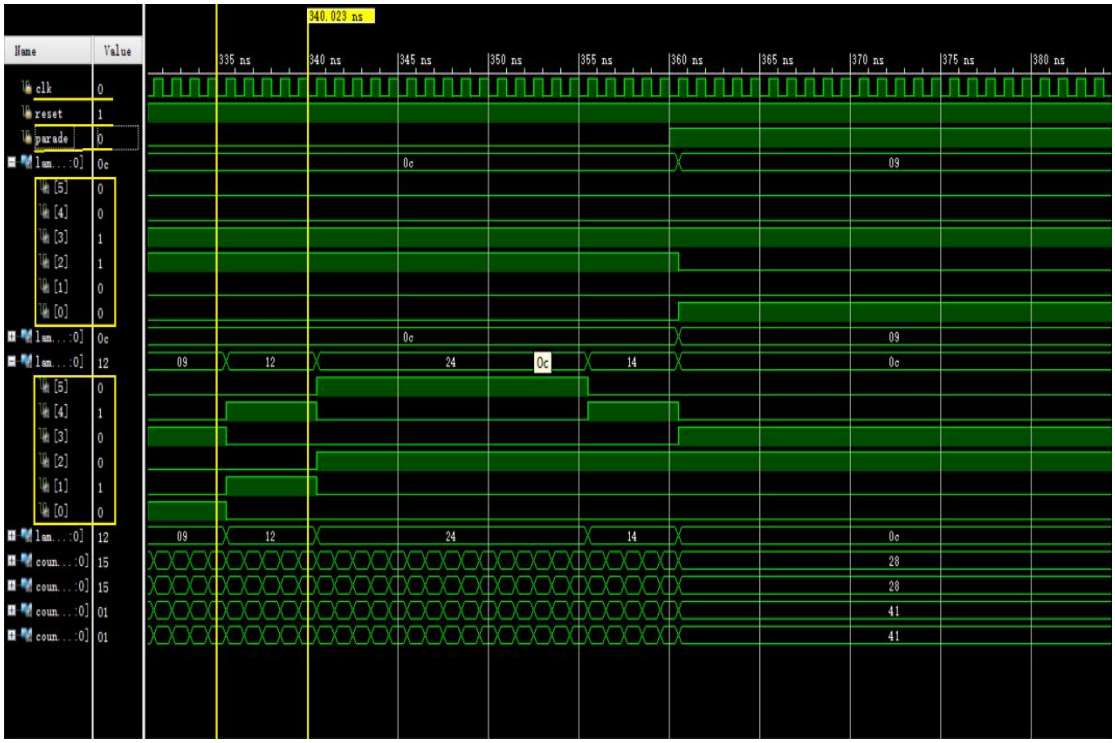
（5.2）State 100 时的 count_a（a 路口交通灯转变到下一状态还需倒计时的秒数）和 count_b 的初值：



count_a = 55 (A 方向：由“A 向禁行”转变到“A 向通行”需要 55s)，

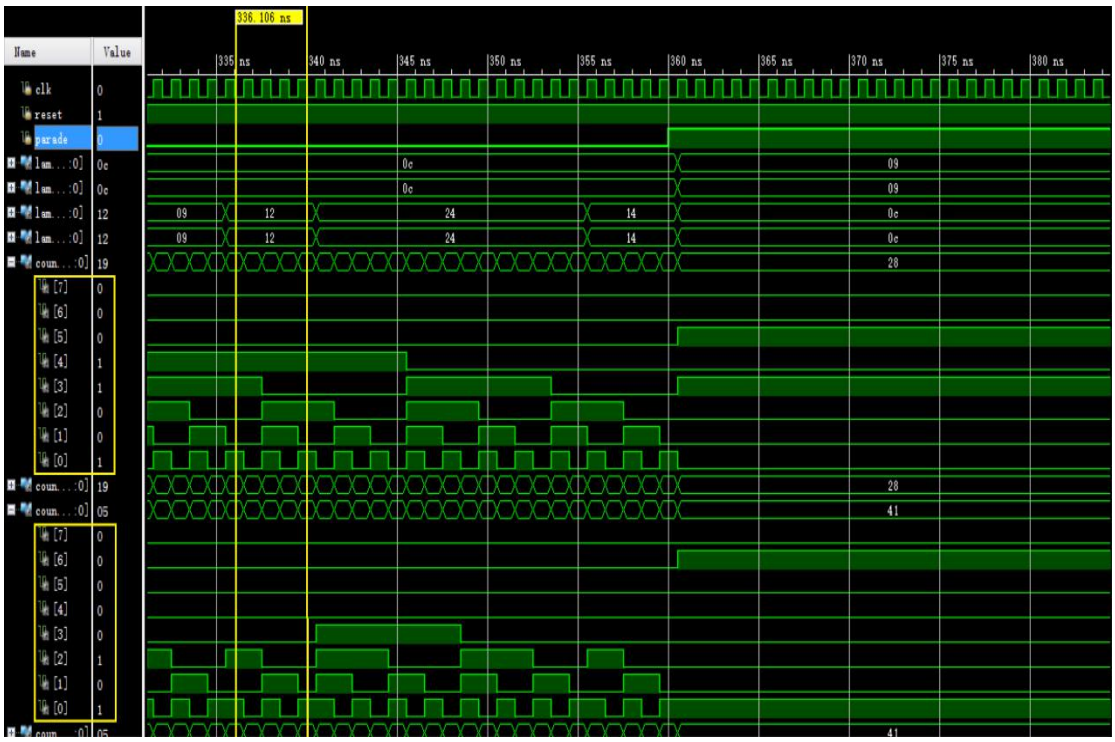
count_b = 30（B 方向：由“B 向通行”转变到“A 向左转准备”需要 30s ）。

（6.1）State 101 时的 lamp_a 和 lamp_b 的显示情况：



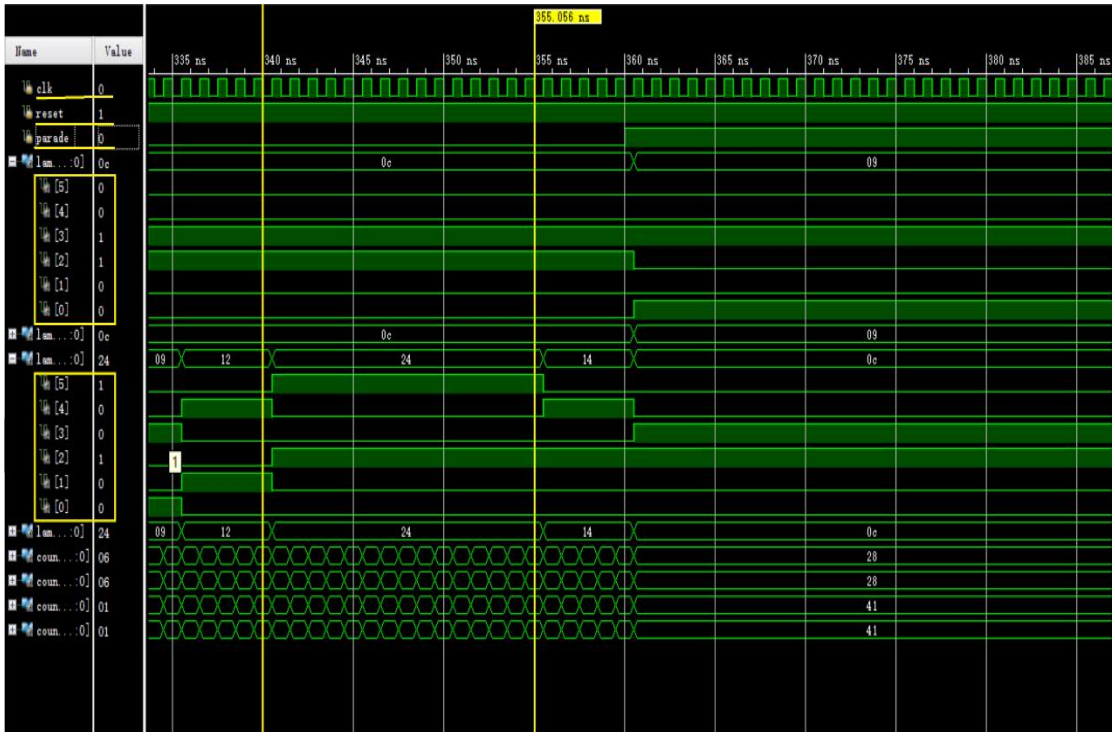
State 101 是 A 向禁行，B 向左转准备。应该是 A 向直行红灯和左转红灯亮，B 向直行黄灯和左转黄灯亮（由左边黄色方框可看出）。

（6.2）State 101 时的 count_a（a 路口交通灯转变到下一状态还需倒计时的秒数）和 count_b 的初值：



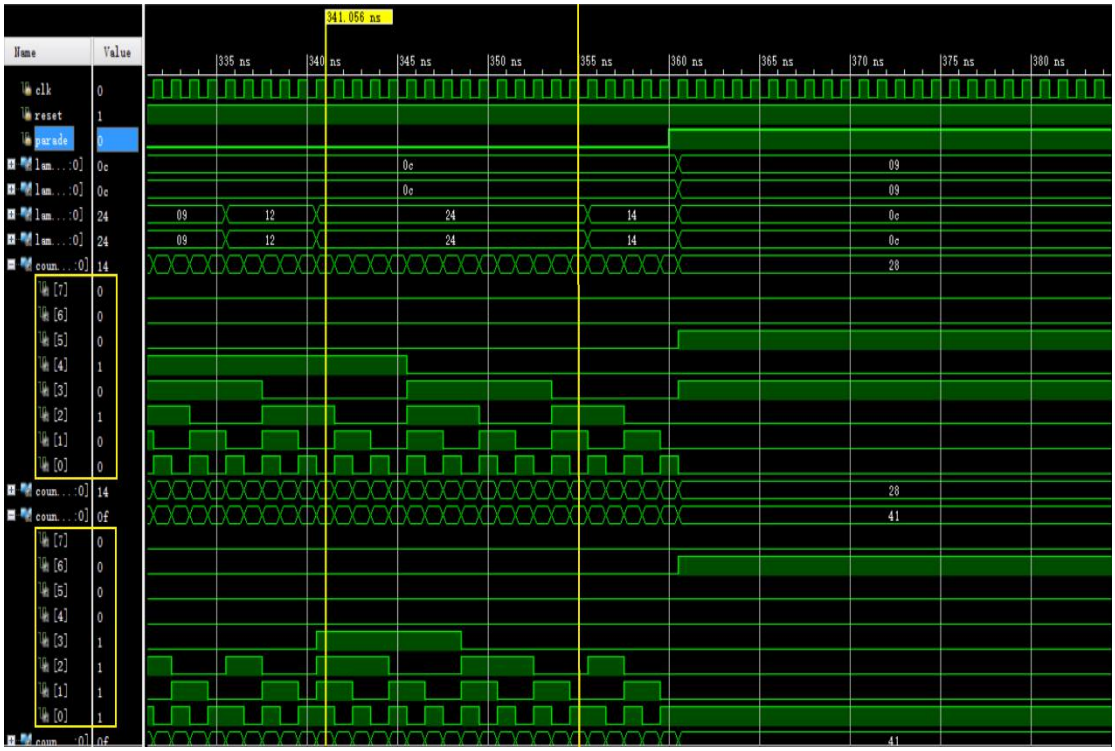
count_a = 25 = 55-30 (A 方向：由“A 向禁行”转变到“A 向通行”需要 25s)，
count_b = 5（B 方向：由“B 向左转准备”转变到“B 向左转”需要 5s ）。

（7.1）State 110 时的 lamp_a 和 lamp_b 的显示情况：



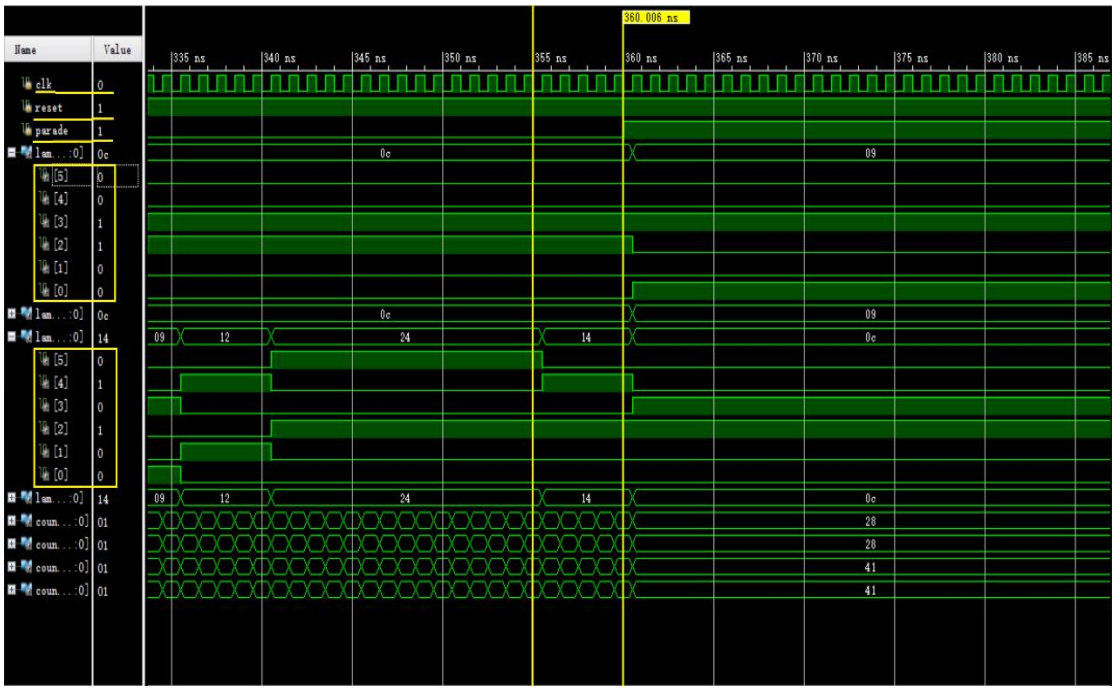
State 110 是 A 禁行,B 左转。应该是 A 向直行红灯和左转红灯亮，B 向直行红灯和左转绿灯亮（由左边黄色方框可看出）。

（7.2）State 110 时的 count_a（a 路口交通灯转变到下一状态还需倒计时的秒数）和 count_b 的初值：



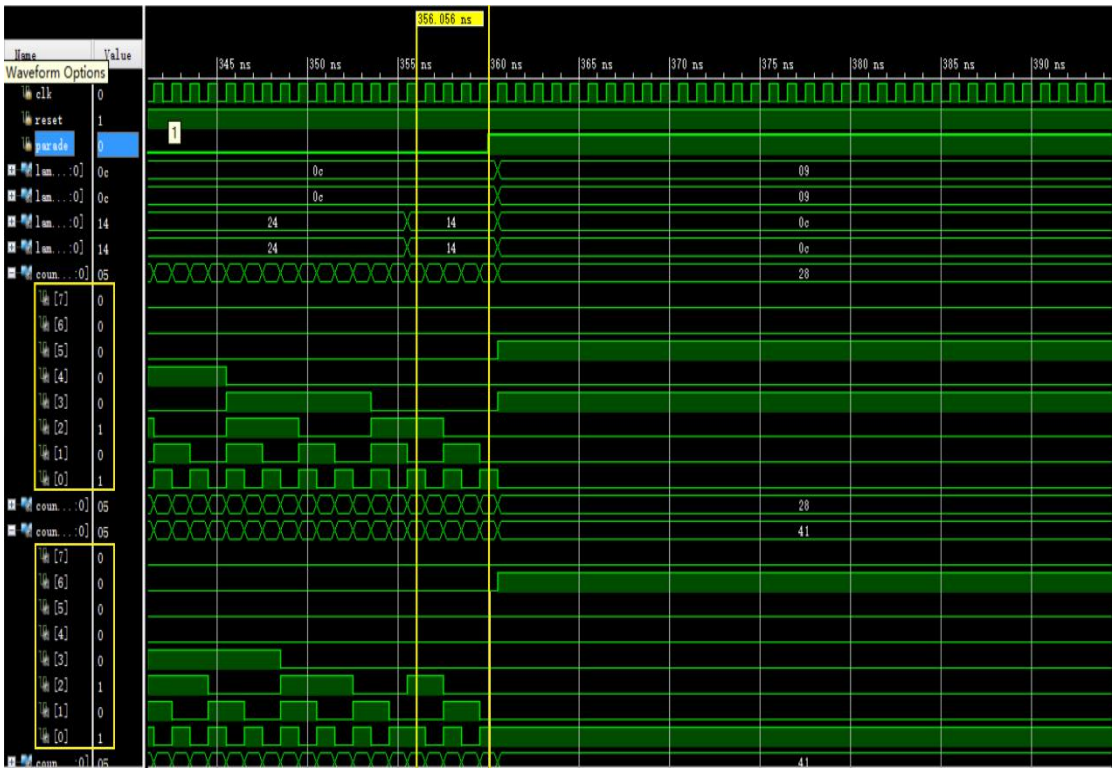
count_a = 20= 25-5 (A 方向：由“A 向禁行”转变到“A 向通行”需要 20s)，
count_b =15（B 方向：由“B 向左转”转变到“B 向禁行准备”需要 15s ）。

（8.1）State 111 时的 lamp_a 和 lamp_b 的显示情况：



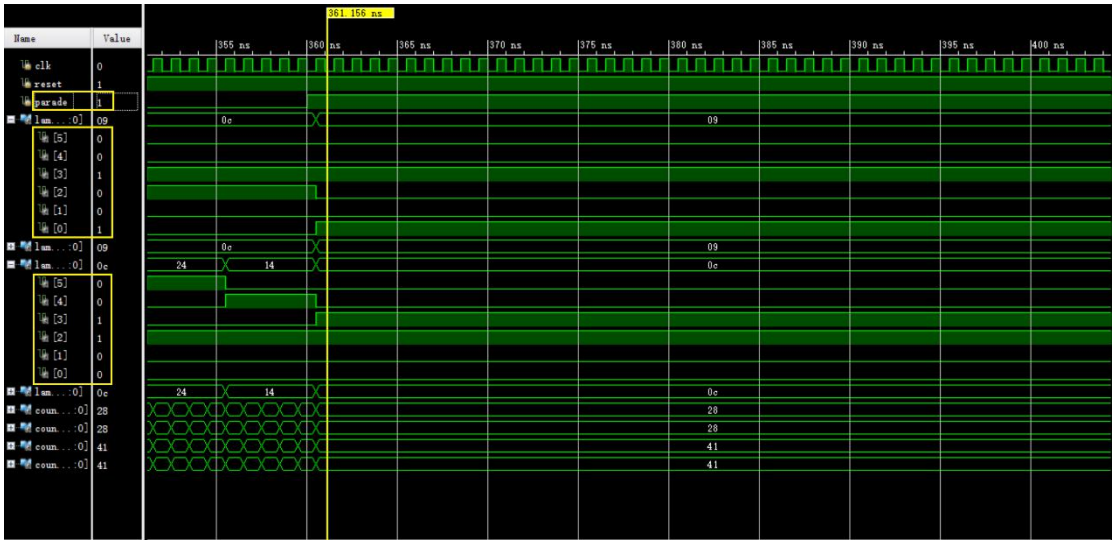
State 111 是 A 向禁行，B 禁行准备。应该是 A 向直行红灯和左转红灯亮，B 向直行红灯和左转黄灯亮（由左边黄色方框可看出）。

（8.2）State 111 时的 count_a（a 路口交通灯转变到下一状态还需倒计时的秒数）和 count_b 的初值：



count_a = 5 (A 方向：由“A 向禁行”转变到“A 向通行”需要 5s)，
count_b = 5（B 方向：由“B 向禁行准备”转变到“B 向禁行”需要 5s ）。

2、当 parade＝1 时，进入游行状态：A 向一直为直行绿灯，B 向一直禁行。



五、效果分析

本次课程设计，成功达到了预期效果，充分考虑了现实生活中的交通情况，具有较强的实用性，并在此基础上，增加了一个游行模式，避免当地举办大型活动时造成人流阻塞，使其更加人性化。

在 vivado 2016.3 环境下进行波形仿真后发现其波形（时间间隔与显示状态）与预期效果一致！

六、总结

1、由于总体设计思路较为清晰，此次基于 verilog 语言的交通灯控制器系统的设计过程虽多有波折，但总体较为顺利。

2、在系统具体框架的具体设立上，我们小组群策群力，集思广益，在查阅多方相关资料、进行一定的知识储备积累后，结合自身实际水平，最终选定了交通灯控制器的设计系统为论文设计方向，并在仔细推敲、讨论、反思以及改进之后，小组整体制订了较为详细和清晰的总体思路。

3、基于清晰的交通灯控制器的设计思路，我们充分考虑了交通灯控制系统在现实生活中的实用性、可行性和科学性，仔细制订了的交通灯运行的有限状态机，并以此为基础开展 verilog 主体程序的编写。

4、在编写交通灯控制系统的过程中，刚开始，我们很困惑，倒计时和转换状态这两个事情怎么联系在一起，经过很多尝试之后，便找到了答案：可以另外声明一个 reg 型变量 tem，当倒计时变到 2 时，就将 tem 置 0，然后转换状态。

5、在仿真的时候,要另写一个 simulate 的 top 代码,由于系统有两个输入 reset 和 parade 都应该由 button 实现，但波形仿真无法输入 button 类型，只好用时钟类型，这导致我们看波形时有些不便，因为只有 reset=1 时的波形是有效的，

而由于 reset 是周期信号，所以有大篇幅的无效波形，这是我们需要改进的地方，我们会继续考虑如何进行优化改进！

七、组员分工

<table>
<tr><th>姓名</th><th>学号</th><th>工作比重</th><th colspan="2">详细分工</th></tr>
<tr><td>郑海君</td><td>15352437</td><td>25%</td><td rowspan="2">（1）确定本次课程设计的选题；
（2）课程设计论文的完成；</td><td rowspan="2">（1）代码实现
（2）波形分析</td></tr>
<tr><td>赵钰莹</td><td>15352433</td><td>25%</td></tr>
<tr><td>张燕梅</td><td>15352423</td><td>25%</td><td rowspan="2">（1）资料搜集
（2）有限状态机的设计
（3）代码调试与仿真</td></tr>
<tr><td>郑晓如</td><td>15352440</td><td>25%</td></tr>
</table>

八、参考资料

1、赵倩、叶波、林丽萍、周多、王晓华编著，Verilog 数字系统设计与 FPGA 应用，清华大学出版社；

2、王金明编著，徐志军主审，Verilog HDL 程序设计教程，人民邮电出版社。

九、附录

//源程序代码

```
module traffic_light(
    input clock, reset, parade,// 1 Hz 时钟信号，同步复位 , parade 为 游行信号
    output[5:0] lamp_a1, lamp_a2, lamp_b1, lamp_b2,// 四组信号灯
    output[7:0] count_a1, count_a2, count_b1, count_b2// 倒计时牌
    );
reg[5:0] lamp_a, lamp_b; // 左转绿，左转黄，左转红，直行红，直行黄，直行绿
reg[7:0] num_a, num_b; // 内部倒计时
reg temp_a, temp_b; // 状态和计数转换信号
reg[2:0] state_a, state_b; // a，b 的状态寄存器
reg[7:0] red_a, yellow_a, green_a, left_a, red_b, yellow_b, green_b, left_b;
assign count_a1 = num_a, count_a2 = num_a;
assign count_b1 = num_b, count_b2 = num_b;
```

```
assign lamp_a1 = lamp_a, lamp_a2 = lamp_a;
assign lamp_b1 = lamp_b, lamp_b2 = lamp_b;

always@(posedge clock)
begin
    if( !reset )// 复位信号在低电平时工作，此时一切复位
    begin
        green_a <= 8'd40; // a 向 通行 40s
        yellow_a <= 8'd5; // a 向 左转准备 5s , a 向 禁止通行准备 5s
        left_a <= 8'd15; // a 向 左转 15s
        red_a <= 8'd55; // a 向 禁止通行 55s

        red_b <= 8'd65; // b 向 禁止通行 65s
        green_b = 8'd30; // b 向 通行 30s
        yellow_b <= 8'd5; // b 向 左转准备 5s , b 向 禁止通行准备 5s
        left_b <= 8'd15; // b 向 左转 15s

        lamp_a <= 'b000000; //左转绿，左转黄，左转红，直行红，直行黄，直行绿 均不亮
        state_a <= 'b000; // 0 状态
        temp_a <= 0; // 状态和计数转换信号
        num_a <= 0; // 内部倒计时
    end
    // reset == 1,正常工作；
    else
    begin
        if( !parade )// parade == 0，非游行状态，交通灯正常工作
        begin
            //当 temp_a == 0 时，状态变换
            if( !temp_a )
            begin
                temp_a <= 1;
                case( state_a )
                    // a 向 通行
                    'b000:
                    begin
                        num_a <= green_a;// 内部倒计时 就是 绿灯的倒计时
                        lamp_a <= 'b001001; // 左转红,直行绿 亮
                        state_a <= 'b001; // 当green_a从40s倒计时到0s时，temp_a==0,
                                          //便可进行判断，变为下一个状态
                    end
                    'b001:
                    begin
                        num_a <= yellow_a;// 内部倒计时 就是 黄灯的倒计时
                        lamp_a <= 'b010010; // 直行黄，左转黄 亮
```

```
                state_a <= 'b010; //   当 yellow_a 从 5s 倒计时到 0s 时，temp_a==0,
                                      //便可进行判断，变为下一个状态
            end
            'b010:
            begin
                num_a <= left_a;// 内部倒计时 就是 左转灯的倒计时
                lamp_a <= 'b100100; //   左转绿,直行红 亮
                state_a <= 'b011; //   当 left_a 从 15s 倒计时到 0s 时，temp_a==0,
                                      //便可进行判断，变为下一个状态
            end
            'b011:
            begin
                num_a <= yellow_a;// 内部倒计时 就是 黄灯的倒计时
                lamp_a <= 'b010100; //   左转黄，直行红 亮
                state_a <= 'b100; //   当 yellow_a 从 5s 倒计时到 0s 时，temp_a==0,
                                      //便可进行判断，变为下一个状态
            end
            'b100:
            begin
                num_a <= red_a;// 内部倒计时 就是 红灯的倒计时
                lamp_a <= 'b001100; //   左转红,直行绿 亮
                state_a <= 'b000; //   当 green_a 从 40s 倒计时到 0s 时，temp_a==0,
                                      //便可进行判断，变为下一个状态
            end
            default: lamp_a <= 'b000000;
        endcase
    end
    // 当 temp_a == 1 时，状态不变换
    else
    begin
        if(num_a > 1)
        begin
            if(num_a[3:0] == 0)
            begin
                num_a[3:0] <= 'b1111;
                num_a[7:4] <= num_a[7:4] - 1;
            end
            else
                num_a[3:0] <= num_a[3:0] - 1;
            // num_a = num_a - 1;
        end
        if(num_a == 2)// 下个 clock 到来时，num_a 为 1，然后变为下一个状态
            temp_a <= 0;
    end
```

```
        else   // parade == 1，a 向游行状态，一直是 a 向直行绿灯，b 禁行
        begin
            num_a <= green_a;// 内部倒计时 就是 绿灯的倒计时
            lamp_a <= 'b001001;
        end

    end
end

always@(posedge clock)
begin
    if( !reset )
    begin
        lamp_b <= 'b000000; //  左转绿，左转黄，左转红，直行红，直行黄，直行绿  均
不亮
        state_b <= 'b000; // 0 状态
        temp_b <=   0; // 状态和计数转换信号
        num_b <= 0; // 内部倒计时
    end
    // reset == 1,正常工作；
    else
    begin
        if( !parade )
        begin
            //当 temp_b == 0 时，状态变换
            if( !temp_b )
            begin
                temp_b <= 1;
                case( state_b )
                    // b 向 禁止
                   'b000:
                    begin
                        num_b <= red_b;
                        lamp_b <= 'b001100;
                        state_b <= 'b001;
                    end
                    'b001:
                    begin
                         num_b <= green_b;
                        lamp_b <= 'b001001;
                        state_b <= 'b010;
                    end
                    'b010:
                    begin
```

```
                    num_b <= yellow_b;
                    lamp_b <= 'b010010;
                    state_b <= 'b011;
                end
                'b011:
                begin
                    num_b <= left_b;
                    lamp_b <= 'b100100;
                    state_b <= 'b100;
                end
                'b100:
                begin
                    num_b <= yellow_b;
                    lamp_b <= 'b010100;
                    state_b <= 'b000;
                end
                default: lamp_b <= 'b000000;
            endcase
        end
        else
        begin
            if(num_b > 1)
            begin
                if(num_b[3:0] == 0)
                begin
                    num_b[3:0] <= 'b1111;
                    num_b[7:4] <= num_b[7:4] - 1;
                end
                else
                    num_b[3:0] <= num_b[3:0] - 1;
            end
            if(num_b == 2)
                temp_b <= 0;
        end
    else // 游行状态，b 向禁行
    begin
        num_b <= red_b;
        lamp_b <= 'b001100;
    end
  end
end
endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2016/12/24 12:30:50
// Design Name:
// Module Name: simu
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module simu(
);
// testbench 时钟信号
reg clk = 0;
//always #500000000 clk <= ~clk;
always #0.5 clk <= ~clk;
// 调用 test 模块
reg reset = 0;
always #120 reset <= ~reset;

wire[5:0] lamp_a1, lamp_a2, lamp_b1, lamp_b2;// 四组信号灯
wire[7:0] count_a1, count_a2, count_b1, count_b2;// 倒计时牌
traffic_light mytraffic_light( clk, reset,// 1 Hz 时钟信号，同步复位
        lamp_a1, lamp_a2, lamp_b1, lamp_b2,// 四组信号灯
        count_a1, count_a2, count_b1, count_b2// 倒计时牌
      );
endmodule
```