实验四 综合应用设计

1. 实验目的

1 掌握一般电子系统的设计、开发、调试和验证方法；

2 掌握Verilog语言基本语法、编程及调试方法；

3 掌握Vivado开发平台及FPGA开发板的使用。

1. 实验内容

1 设计制作一个数字钟或交通灯控制器;

2 对电路进行仿真测试，分析结果；

3 下载到FPGA开发板做硬件验证测试。

三、实验要求

1 画出模块的电路图；

2 分析电路的仿真波形和板卡验证结果；

3 记录设计和调试过程。

1. 实验过程

题目要求：

**设计一个由一条主路和一条支路汇合成的十字路口交通灯控制器（EGo1开发板） 。**

1. **主、支路各设三个LED灯，分别代表红、黄、绿灯；**
2. **主、支路各设置两个显示数码管，倒计时显示；**
3. **信号灯变换次序：**

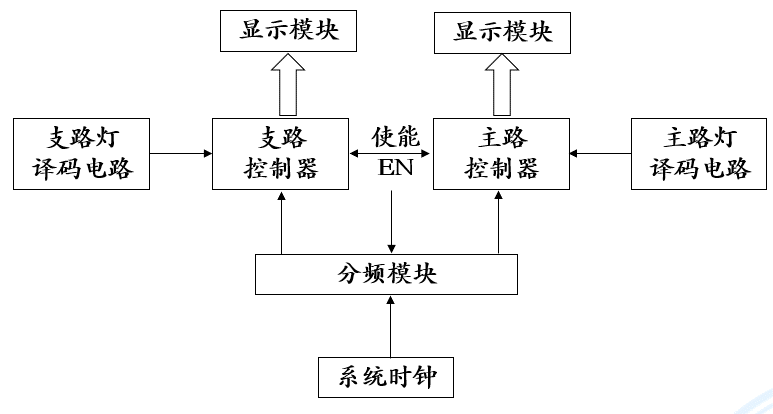
**主路绿灯、支路红灯30秒；**

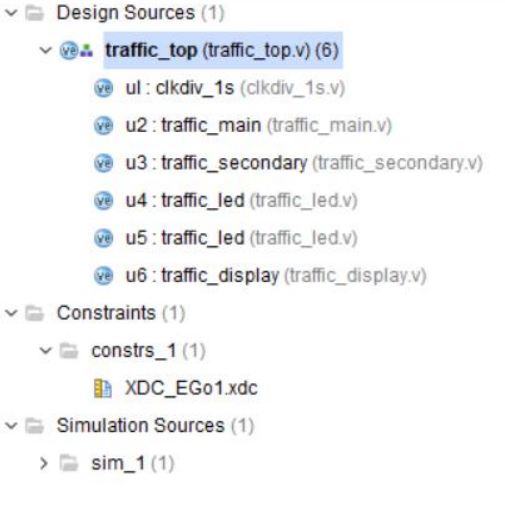
**主路黄灯、支路红灯5秒；**

**主路红灯、支路绿灯20秒；**

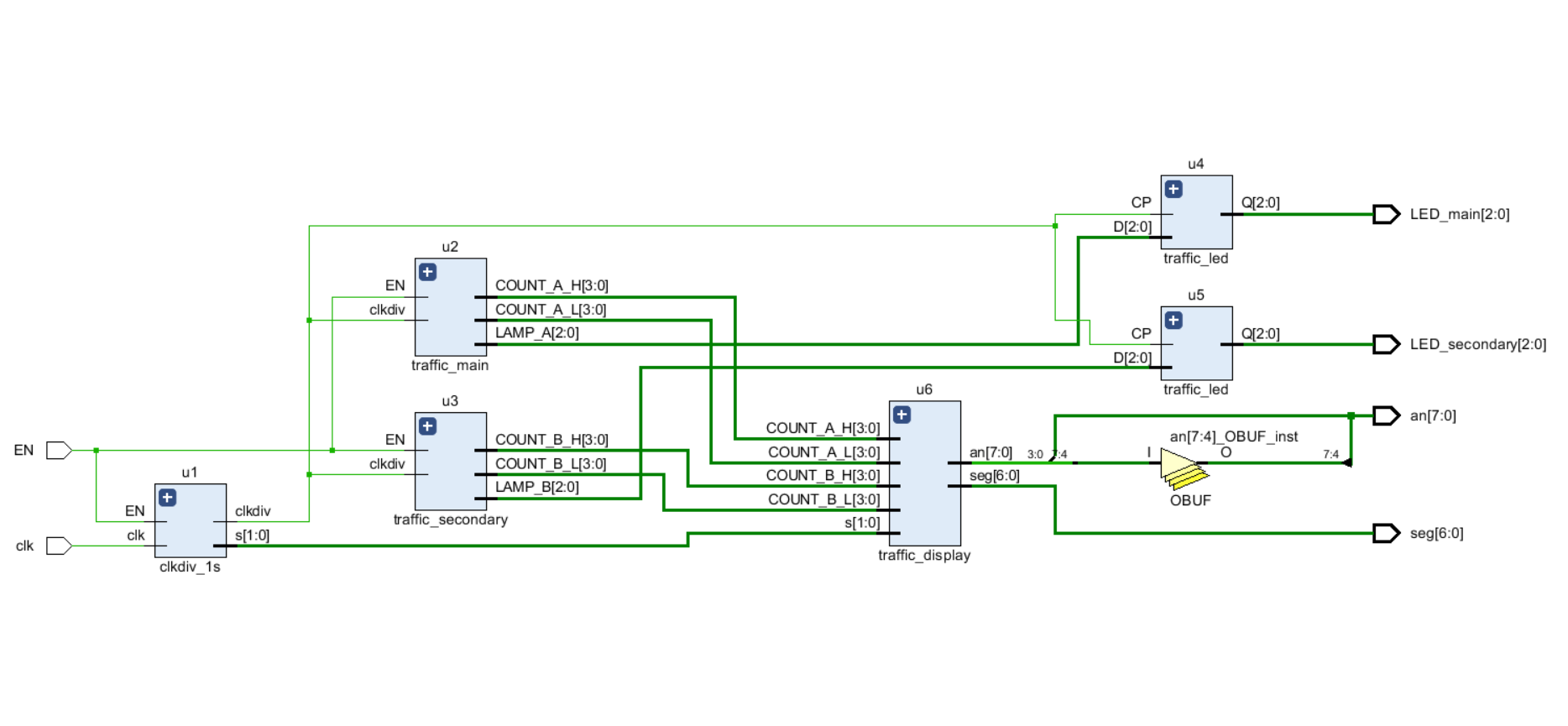
**主路红灯、支路黄灯5秒。**

**1 系统整体设计方案；**





RTL分析：



**2 各模块的主要功能及实现/核心代码：**

**traffic\_top模块：**

module traffic\_top(

input EN,**//使能信号**

output [2:0] LED\_main,LED\_secondary,**//分别表示主路和支路的红绿黄三个灯哪一个灯是亮着的**

output [7:0] an,**//用于控制七段数码管上的数字显示的八位输出信号，本题中只用到了四位**

input clk,**//同步时钟信号**

output [6:0] seg**//用于驱动七段数码管的七位输出信号。**

);

wire clk\_1s;**//** **通过将时钟分频器（clkdiv\_1s）连接到交通模块，用于生成1 Hz的时钟信号clk\_1s的信号线**

wire [1:0] sel;**//** **两位信号线，用于表示显示的是哪一位数码管**

clkdiv\_1s u1(.clk(clk),.EN(EN),.clkdiv(clk\_1s),.s(sel));**//** **clkdiv\_1s模块负责使用输入的clk和EN信号生成1Hz的时钟信号（clk\_1s）。它还提供一个两位选择信号（sel），用于表示显示的是哪一位数码管**

wire [3:0] COUNT\_A\_H,COUNT\_A\_L; **//分别表示A的倒计时高位和低位**

wire [2:0] LAMP\_A;

traffic\_main u2( .clkdiv(clk\_1s),.EN(EN),.COUNT\_A\_H(COUNT\_A\_H),.COUNT\_A\_L( COUNT\_A\_L),.LAMP\_A(LAMP\_A));

**//** **traffic\_main模块负责主路交通信号控制模块，接收clkdiv信号、EN信号，并生成主交通信号的高位计数值和低位计数值COUNT\_A\_H和COUNT\_A\_L，以及灯控信号LAMP\_A。**

wire [3:0] COUNT\_B\_H,COUNT\_B\_L;

wire [2:0] LAMP\_B;

traffic\_secondary u3(.clkdiv(clk\_1s),.EN(EN),.COUNT\_B\_H(COUNT\_B\_H),.COUNT\_B\_L(COUNT\_B\_L),.LAMP\_B(LAMP\_B));

**//** **traffic\_secondary模块负责支路交通信号控制模块，接收clkdiv信号、EN信号，并生成主交通信号的高位计数值和低位计数值COUNT\_B\_H和COUNT\_B\_L，以及支路灯控信号LAMP\_B。**

traffic\_led u4(.D(LAMP\_A),.CP(clk\_1s),.Q(LED\_main));

traffic\_led u5(.D(LAMP\_B),.CP(clk\_1s),.Q(LED\_secondary));

**//** **traffic\_led模块的实例u4,u5用于控制交通信号灯的LED显示，分别接收灯控信号LAMP\_A和LAMP\_B，时钟信号clk\_1s，并将输出连接到主灯的LED\_main和辅助灯的LED\_secondary**

traffic\_display u6(.s(sel),.COUNT\_A\_H(COUNT\_A\_H),.COUNT\_A\_L(COUNT\_A\_L),.COUNT\_B\_H(COUNT\_B\_H),.COUNT\_B\_L(COUNT\_B\_L),.seg(seg),.an(an));

endmodule

**// traffic\_display模块用于控制七段数码管的显示**

**clkdiv\_1s模块：**

module clkdiv\_1s(

input clk,

input EN,

output clkdiv,**//** **分频后的时钟信号，周期为1秒**

output [1:0] s

);

reg [30:0] q;**//用于计数的31位寄存器**

reg [16:0] a;

reg temp\_clk;

assign s = a[16:15];**//** **将a的第16位和第15位赋值给 s，用于选择信号**

assign clkdiv = temp\_clk;

always @ (posedge clk )**//上升沿触发**

begin

if ( EN )

if ( q == 50000000-1 )

begin

q <= 0;

temp\_clk <=~ temp\_clk;

end

else q <= q+1;

**//clk的频率为100MHZ，一秒钟内clk就有100M个周期，触发了100M个上升沿，每触发一次上升沿就令q加1，直到q等于50M，此时将q置零同时让temp\_clk翻转，这样就达到了一秒钟temp\_clk翻转两次的结果，使得temp\_clk的周期为1s。**

else q <= 0;**//若EN为0，则将q置零**

end

always @ ( posedge clk )

a <= a+1;

**//每个时钟上升沿都使 a 增加 1。这个计数器的最高位是 a[16]，输出 s 是寄存器 a 的最高两位，每经过32768个上升沿信号都会使s加一 ，s在后续traffic\_display 模块中充当数码管动态扫描信号**。

**traffic\_main模块:**

module traffic\_main(

input clkdiv,

input EN,

output reg[3:0] COUNT\_A\_H,**//高位计数器输出**

output reg[3:0] COUNT\_A\_L,**//低位计数器输出**

output reg[2:0] LAMP\_A **//灯控信号**

);

reg[2:0] count\_a ;**//表示下一次状态变化（主路灯和支路灯有一个倒计时结束）时该显示哪一个灯**

reg[3:0] num\_a\_h,num\_a\_l;

reg temp\_a;**//状态变量**

always @ ( posedge clkdiv )**//clk上升沿触发**

begin

COUNT\_A\_L <= num\_a\_l; COUNT\_A\_H <= num\_a\_h;

if( EN )

begin

if( !temp\_a ) begin**//状态转化语句**

temp\_a <= 1;**//进入if语句后将temp\_a置1，保证倒计时结束之前都不会再次进入if语句实现状态转化，当倒计时即将结束时将temp\_a置零，下一次上升沿触发时进入if语句实现状态转化**

case ( count\_a )

0: begin

num\_a\_h <= 4'b0011;num\_a\_l <=4'b0000;

LAMP\_A <= 3'b001;count\_a <= 1;

**//表示绿灯，倒计时30秒，结束后亮黄灯**

end

1:begin

num\_a\_h <= 4'b0000;num\_a\_l <= 4'b0101;

LAMP\_A <= 3'b010;count\_a <= 2;

**//表示黄灯，倒计时5秒，结束后亮红灯**

end

2:begin

num\_a\_h <= 4'b0010;num\_a\_l <= 4'b0101;

LAMP\_A <= 3'b100;count\_a <= 0;

**//表示红灯，倒计时25秒，结束后亮绿灯**

end

default: LAMP\_A <= 3'b111;

endcase

end

else begin

if ( {num\_a\_h,num\_a\_l} > 8'b0000\_0001 )

if ( num\_a\_l == 0 )

begin

num\_a\_l <= 4'b1001;

num\_a\_h <= num\_a\_h-1;

end

**//若需要倒计时-1时倒计时低位为0，则将倒计时高位-1，将倒计时低位置成9**

else num\_a\_l <= num\_a\_l-1;

**//若需要倒计时-1时倒计时低位不为0，则直接将倒计时低位-1即可**

if( {num\_a\_h,num\_a\_l} == 8'b0000\_0010 ) temp\_a <= 0;

**//若倒计时为2，倒计时即将结束，此时进入循环，将temp\_a置零，方便下一次上升沿触发时进入上述的状态转化语句。至于此处为什么是2而不是0或1，因为实际有延迟的存在，需要用2个时钟信号的时间来应对延迟**

end

end

else begin

LAMP\_A <= 3'b100;temp\_a <= 0; count\_a <= 0;

**//使能信号EN为0时设置成红灯亮，将temp\_a和count\_a置零**

end

end

endmodule

综上所述，当clk信号上升沿触发且EN有效时，若状态变量temp\_a为0，则意味着倒计时结束，程序进入状态转化语句，根据count\_a的值来确定转化到哪一个灯，倒计时设置成多杀。若状态变量temp\_a为1，则说明倒计时没有结束，进入倒计时递减语句，若倒计时减至2，则将temp\_a置0，下一次信号上升沿触发时进入状态转化语句。traffic\_main 模块实现了主要交通信号灯的计时和灯光状态控制功能，产生30秒绿灯，5 秒黄灯和 25 秒红灯的信号

**traffic\_secondary模块:**

module traffic\_secondary(

input clkdiv,

input EN,

output reg[3:0] COUNT\_B\_H,**//高位计数器输出**

output reg[3:0] COUNT\_B\_L,**//低位计数器输出**

output reg[2:0] LAMP\_B); **//灯控信号，顺序为红黄绿**

reg[2:0] count\_b; **//表示下一次状态转化（主路灯和支路灯有一个倒计时结束）时该 显示哪一个灯**

reg[3:0] num\_b\_h,num\_b\_l;

reg temp\_b;**//状态变量**

always @ ( posedge clkdiv ) begin**//clk上升沿触发**

COUNT\_B\_L <= num\_b\_l; COUNT\_B\_H <= num\_b\_h;

if( EN ) begin

if( !temp\_b ) begin**//状态转化语句**

temp\_b <= 1; **//进入if语句后将temp\_b置1，保证倒计时结束之前都不会再次进入if语句实现状态转化，当倒计时即将结束时将temp\_b置零，下一次上升沿触发时进入if语句实现状态转化**

case ( count\_b )

0: begin

num\_b\_h <= 4'b0011;num\_b\_l <=4'b0101;

LAMP\_B <= 3'b100;count\_b <= 1;

**//表示红灯，倒计时35秒，结束后绿灯亮**

end

1:begin

num\_b\_h <= 4'b0010;num\_b\_l <= 4'b0000;

LAMP\_B <= 3'b001;count\_b <= 2;

**//表示绿灯，倒计时20秒，结束后黄灯亮**

end

2:begin

num\_b\_h <= 4'b0000;num\_b\_l <= 4'b0101;

LAMP\_B <= 3'b010;count\_b <= 0;

**//表示黄灯，倒计时5秒，结束后红灯亮**

end

default: LAMP\_B <= 3'b111;

endcase

end

else begin

if ( {num\_b\_h,num\_b\_l} > 8'b0000\_0001 )

if ( num\_b\_l == 0 )

begin

num\_b\_l <= 4'b1001;

num\_b\_h <= num\_b\_h-1;

**//若需要倒计时-1时倒计时低位为0，则将倒计时高位-1，将倒计时低位置成9**

end

else num\_b\_l <= num\_b\_l-1;

**//若需要倒计时-1时倒计时低位不为0，则直接将倒计时低位-1即可**

if( {num\_b\_h,num\_b\_l} == 8'b0000\_0010 ) temp\_b <= 0;

**//若倒计时为2，倒计时即将结束，此时进入循环，将temp\_a置零，方便下一次上升沿触发时进入上述的状态转化语句。至于此处为什么是2而不是0或1，因为实际有延迟的存在，需要用2个时钟信号的时间来应对延迟**

end

end

else begin

LAMP\_B <= 3'b100;temp\_b <= 0; count\_b <= 0;

**//使能信号EN为0时设置成红灯亮，将temp\_b和count\_b置零**

end

end

endmodule

仿照主路控制器即可形成支路控制器,在主路控制器的代码基础上,其中要把红灯改为 35 秒, 绿灯改为 20 秒,黄灯改为 5 秒,同时灯的变换顺序也要改变。traffic\_secondary 模块模拟了一个交通灯系统的支路交通路线的控制逻辑。 它与 traffic\_main 模块的工作原理类似,主要区别在于初始状态和状态持续时间的设置。 此模块通过变量的变化模拟了交通灯的状态（红、绿、黄灯）以及每个状态对应的剩余时间的显示，产生了35秒红灯，20秒绿灯和5秒黄灯的信号

traffic\_display模块:

该模块是用于显示倒计时的模块，适用于交通灯倒计时显示的场景。下面详细分析其工

作机制：

1、输入输出：

 s：2 位的输入信号，用于选择显示的内容。它可以取值为 0，1，2，3。

 COUNT\_A\_H、COUNT\_A\_L：各为 4 位的输入信号，分别表示 A 路口交通灯的倒计时高位和低位数字。

 COUNT\_B\_H、COUNT\_B\_L：各为 4 位的输入信号，分别表示 B 路口交通灯的倒计时高位和低位数字。

 seg：7 位的输出信号，用于控制数码管的 7 个段（a, b, c, d, e, f, g）。

 an：8 位的输出信号，用于选择数码管的位。

2、always 块作用：

 第一个 always块：在任何输入改变的时候都会执行，根据输入 s 的值决定了显示哪个数码管（an），以及选择显示的内容（digit）。比如当 s 为 0 时，an 选择第一位数码

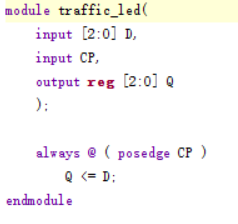
管，同时选择显示 COUNT\_B\_L（即 B 路口交通灯的倒计时低位）。an决定了哪一位数码管显示，经过1s后an会依次从0变化到3，再将主路灯和支路灯的高低位依次赋值给digit保证了四个数码管都有显示

 第二个 always 块：在任何输入改变的时候都会执行，根据选择的显示内容 digit，将

其转换为对应的 7 段数码管显示代码（seg）。比如当 digit 是 4'b0001（即数字 1）时，

seg 将会被赋值为 7'b000\_0110，这就是在 7 段数码管上显示数字 1 的代码。

**traffic\_led模块:**



该模块是一个简单的正边沿触发的 D 触发器。在这个模块中，输入D 是一个 3 位的信号，表示交通灯当前状态,输入 CP 是一个时钟信号，输出 Q 是一个 3 位的信号, 用于控制交通 灯指示灯的亮灭。当 CP 信号的边沿从低变为高（即正边沿）时，D 的值将被储存，并在输出 Q 上显示。此时，Q 的值会立即更新为 D 的当前值。这就是 D 触发器的工作原理。

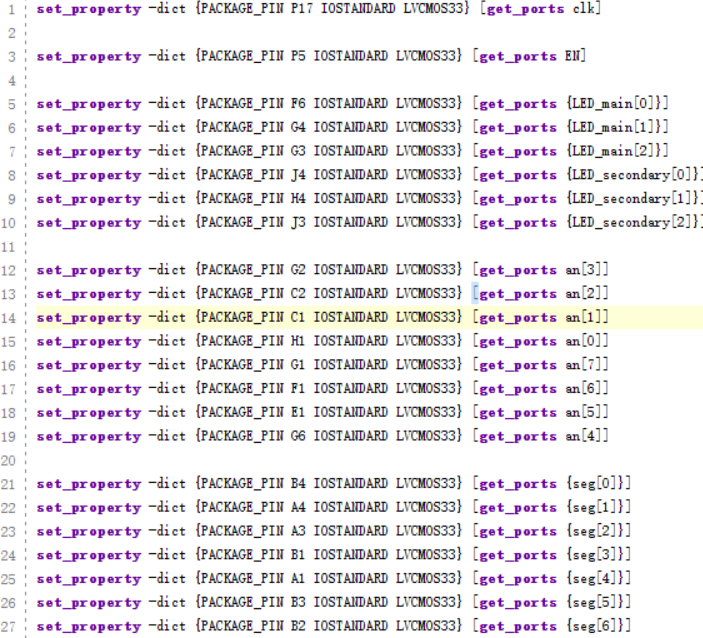
这个模块的作用是在每个时钟周期（由 CP 信号控制）时，更新交通灯的状态。D 表示

交通灯的状态（如红绿黄），而 Q 表示输出到交通灯的实际信号。当时钟信号 CP 在正边沿

到来时，交通灯的状态会被更新为 D 的值。

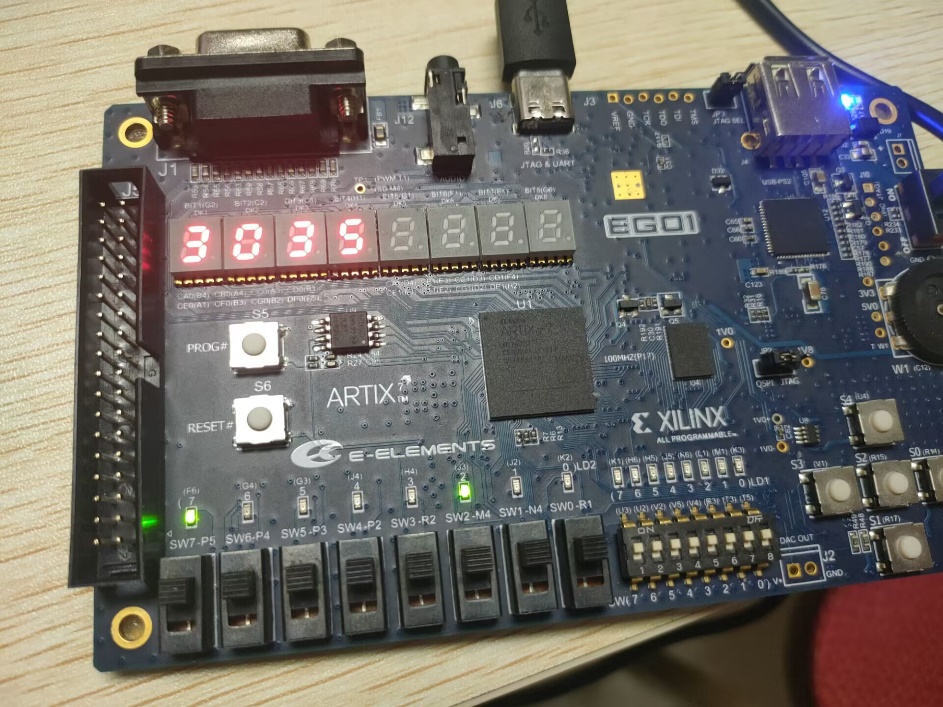
总的来说，这个模块的作用是在每个时钟周期更新交通灯的状态。

约束文件：

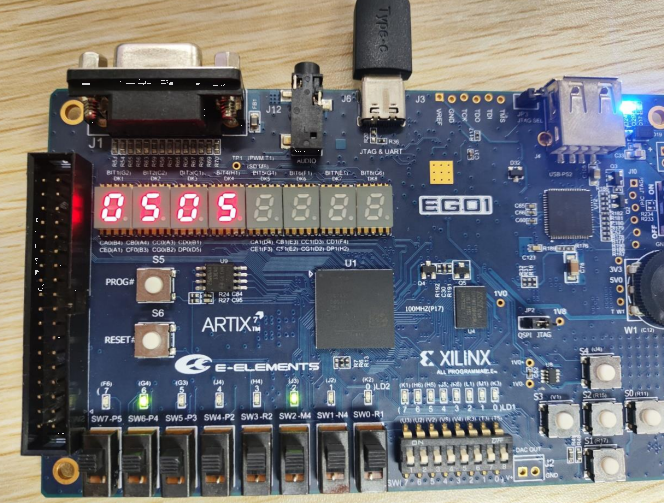


**3 系统验证结果。**

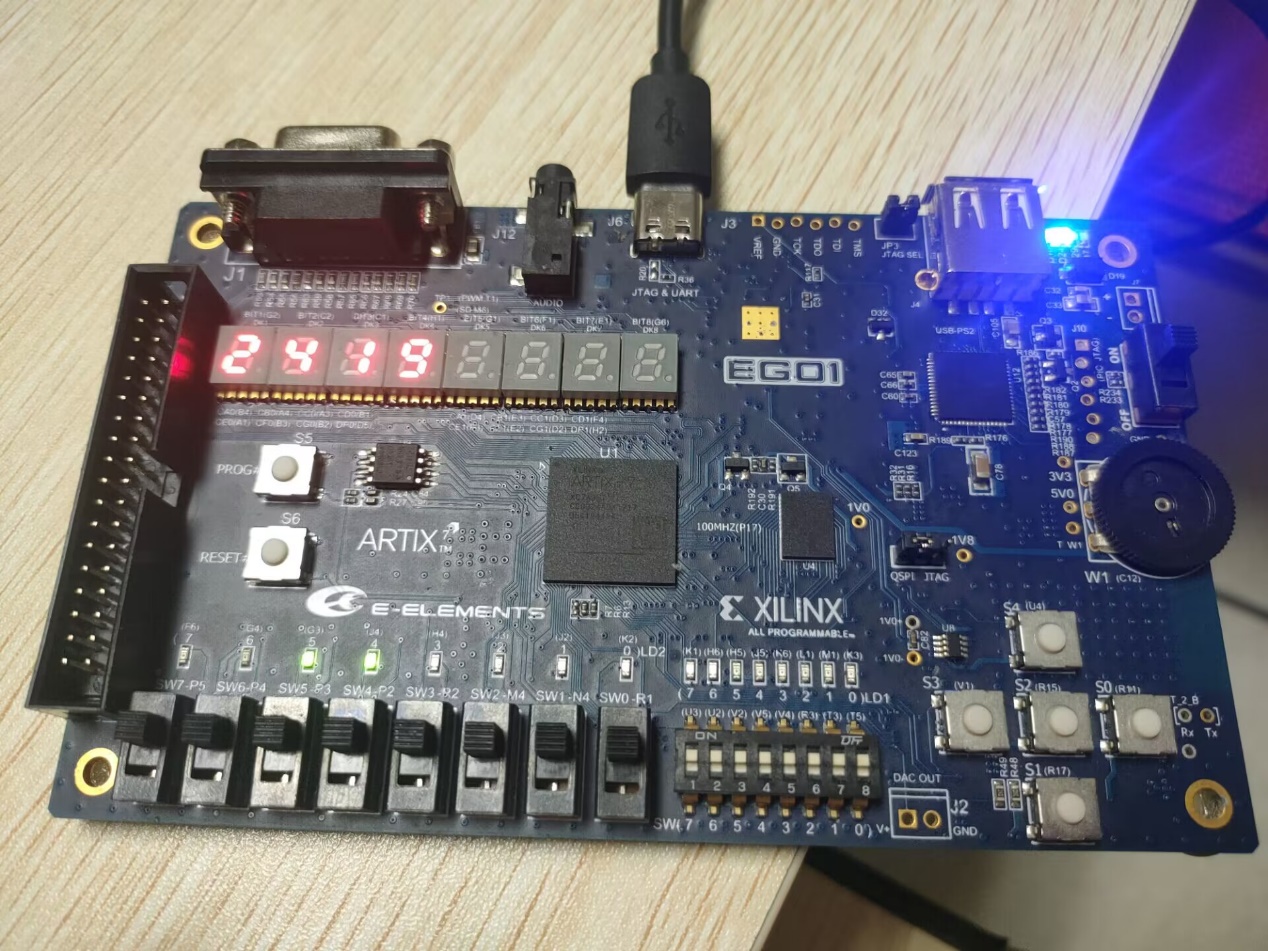
主路绿灯、支路红灯 30 秒



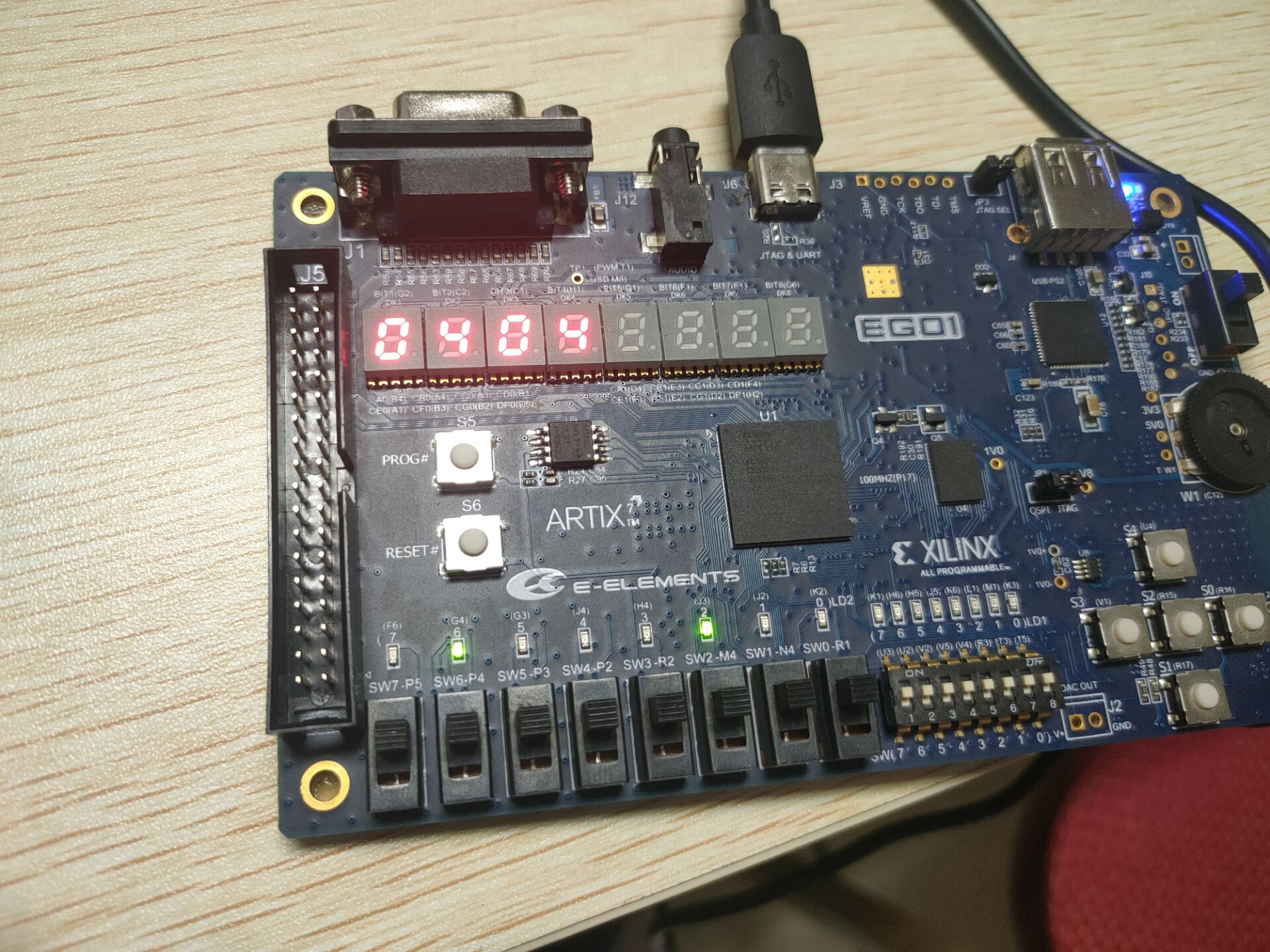
主路黄灯、支路红灯 5 秒



主路红灯、支路绿灯 20 秒



主路红灯、支路黄灯 5 秒

****

最终程序烧录至开发板后运行结果如图。可以看到，开发板上倒计时和指示灯状态变化,与预期一致，合乎所编写的程序逻辑，说明交通信号灯系统设计成功。

五、实验中遇到的问题及心得体会

**记录实验过程中遇到的问题和调试解决过程，心得体会等。**

本次实验是数电的最后一次实验，也是最有挑战性的一次实验。虽然，我对 Verilog 硬件设计从最初的茫然变为可以独立编写自己所想的设计模块，着实是有了长足的进步。但是，对于本次实验，我刚开始仍旧毫无思路以至于无从下笔。在旁听了几次老师的讲解后，我对系统的框架有了初步的认知。于是，我仿照主路信号灯的控制逻辑，尝试着编写了支路信号灯的控制逻辑。随着实验的不断上手，我对 Verilog 语言的理解也渐渐深入，补全系统框架也不再是一件困难的事情了。我综合了前三次实验所学的知识，参考相关资料和代码设计了一个交通信号灯的控制系统，在完成系统的编写后，我将程序烧录至开发板上进行验证。没想到我所编写的程序一次就满足了系统的设计要求，这给了我极大的满足感。通过这次实验，我不仅对 Verilog 这一硬件描述语言有了更深的认知，更提高了我对数电开发的兴趣，四次实验让我对硬件开发语言有了初步的认识，熟悉了 FPGA 开发板的使用。同时也增强了自己的动手能力和硬件查错能力