实验四 综合电路设计实验报告

1. 实验题目

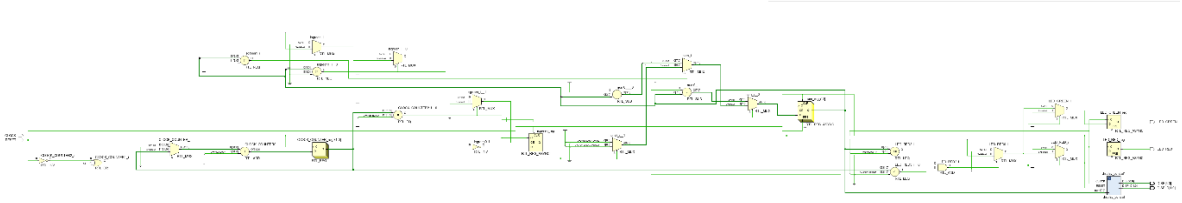
- 人行交通灯设计与实现（难度系数：0.8）

  人行交通灯描述如下：

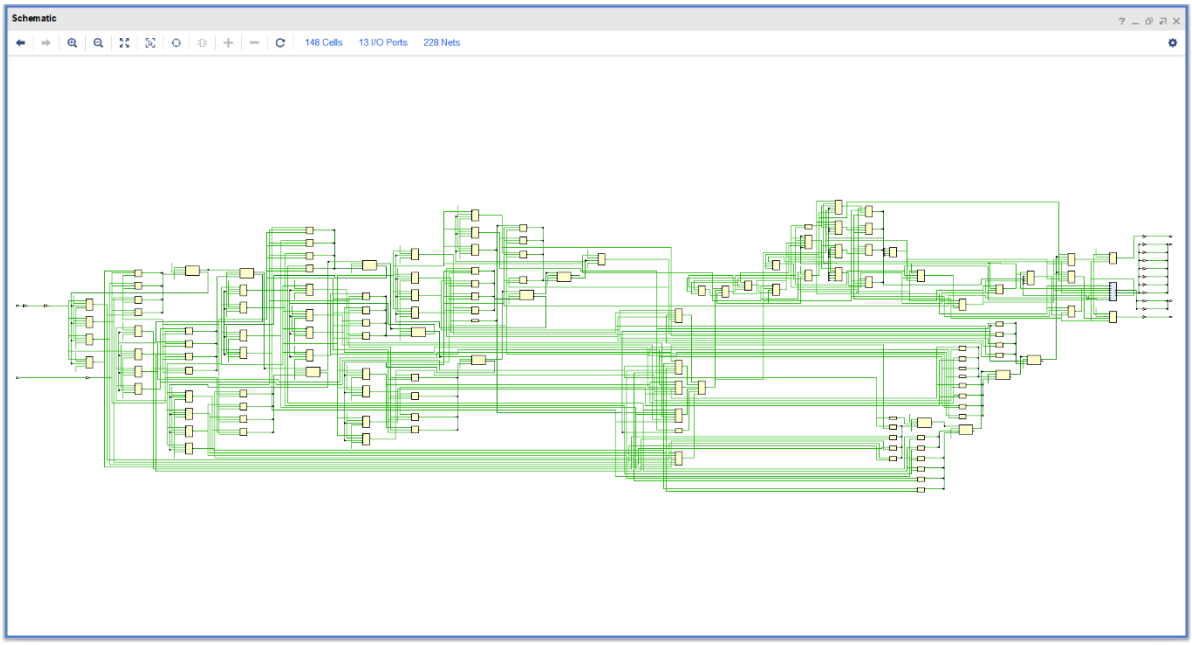
  - “人行交通灯”用两只不同颜色的 LED 灯显示；
  - 红、绿两灯点亮时间比为 30:20；
  - 红、绿两灯亮时，用两位数码管以“倒计时”方式显示剩余时间；
  - 最后三秒时“闪烁”LED 灯，以表示临近结束
  - 开机自动运行，显示时间单位为“秒（S）”。

2. 电路设计

综合前：



综合后：



电路分为三个模块：顶层控制模块（时序电路）、显示控制模块（时序电路）、十六进制码到七段数码管转换模块（组合电路）。

3. 电路实现

main.v（顶层控制模块）：

```
`timescale 1ns / 1ps
// 顶层控制模块
module traffic_light(
    input CLOCK, // 时钟信号
    input RESET, // 复位信号
    output reg LED_RED, // 红灯信号
    output reg LED_GREEN, // 绿灯信号
    output [6:0] DISP, // 七段数码管的 LED 信号
    output [1:0] DISP_D // 七段数码管的段选信号
    );
    parameter ONE_SECOND=100000000,HALF_SECOND=ONE_SECOND/2; // ONE_SEC
OND 是指多少个时钟周期被当成红绿灯的一“秒”
    parameter RED_TIME=8'h30,GREEN_TIME=8'h20,BLINK_TIME_RED=8'h03,BLIN
K_TIME_GREEN=8'h03; // 分别为红灯和绿灯的持续时间和闪烁时间（秒），2 位 BCD
码
    integer CLOCK_COUNTER=0; // 时钟计数器
    reg [7:0] num=RED_TIME; // 现在倒计时上显示的数字
    reg isgreen=1'b0; // 现在是红灯亮还是绿灯亮，0 = 红灯亮， 1 = 绿灯亮

    // 时钟计数：
    always @(posedge CLOCK or negedge RESET) begin
        if (!RESET||CLOCK_COUNTER==ONE_SECOND) begin
            CLOCK_COUNTER=0;
        end
        CLOCK_COUNTER=CLOCK_COUNTER+1;
    end

    // 控制显示的数字和红绿灯哪个亮：
    always @(posedge CLOCK or negedge RESET) begin
        if (!RESET) begin
            isgreen=1'b0; // 复位后红灯先亮
            num=RED_TIME;
        end else if (CLOCK_COUNTER==ONE_SECOND) begin
            if (num[3:0]!=4'h0) begin
                num=num-8'h01;
            end else begin
```

```
                if (num[7:4]!=4'h0) begin
                    num={num[7:4]-4'h1,4'h9};
                end else begin
                    isgreen=~isgreen;
                    num=isgreen?GREEN_TIME:RED_TIME;
                end
            end
        end
    end

    // 点亮该亮的红绿灯
    always @(posedge CLOCK or negedge RESET) begin
        if (!RESET) begin
            LED_RED<=1'b1;
            LED_GREEN<=1'b1; // 按住复位按钮时红绿灯都亮
        end else if (isgreen) begin
            LED_RED<=1'b0;
            LED_GREEN<=(num<=BLINK_TIME_GREEN&&CLOCK_COUNTER<=HALF_SECO
ND)?1'b0:1'b1;
        end else begin
            LED_RED<=(num<=BLINK_TIME_RED&&CLOCK_COUNTER<=HALF_SECOND)?
1'b0:1'b1;
            LED_GREEN<=1'b0;
        end
    end

    // 实例化显示控制模块，通过 num 来控制显示的数字
    display_control display_control1(
        .CLOCK(CLOCK),
        .RESET(RESET),
        .num(num),
        .DISP(DISP),
        .DISP_D(DISP_D)
    );

endmodule
```

display_control.v（显示控制模块）：

```
`timescale 1ns / 1ps
// 显示控制模块
module display_control(
    input CLOCK, // 时钟信号
    input RESET, // 复位信号
```

```
    input [7:0] num, // 要显示的数字，2 位 BCD 码或十六进制数
    output reg [6:0] DISP, // 七段数码管的 LED 信号
    output reg [1:0] DISP_D // 七段数码管的段选信号
    );
    parameter CLOCKS_PER_FRAME=2000000,CLOCKS_PER_HALF_FRAME=CLOCKS_PER
_FRAME/2; // CLOCKS_PER_FRAME 代表每隔多少个时钟周期就刷新一遍显示的数字
    integer CLOCK_COUNTER=0; // 时钟计数器
    wire [6:0] DISP_LO,DISP_HI; // 分别是低位和高位的七段数码管的 LED 信号

    // 时钟计数：
    always @(posedge CLOCK or negedge RESET) begin
        if (!RESET||CLOCK_COUNTER==CLOCKS_PER_FRAME) begin
            CLOCK_COUNTER=0;
        end
        CLOCK_COUNTER=CLOCK_COUNTER+1;
    end

    // 显示数字：
    always @(posedge CLOCK or negedge RESET) begin
        if (!RESET) begin
            DISP_D<=2'b11;
            DISP<=7'h7f; // 按住复位按钮时点亮所有灯
        end else if (CLOCK_COUNTER<=CLOCKS_PER_HALF_FRAME) begin
            DISP_D<=2'b01;
            DISP<=DISP_LO;
        end else begin
            DISP_D<=2'b10;
            DISP<=DISP_HI;
        end
    end

    // 实例化十六进制码到七段数码管转换模块
    num2DISP num2DISP1(
        .num(num[3:0]),
        .DISP(DISP_LO)
    );

    num2DISP num2DISP2(
        .num(num[7:4]),
        .DISP(DISP_HI)
    );
endmodule
```

num2DISP.v（十六进制码到七段数码管转换模块）：

```
`timescale 1ns / 1ps
// 十六进制码到七段数码管转换模块
module num2DISP(
    input [3:0] num, // 输入一位十六进制数
    output [6:0] DISP // 输出七段数码管的 LED 信号
    );
    assign DISP[6]=(num==4'h1||num==4'h4||num==4'hb||num==4'hc||num==4'
hd)?1'b0:1'b1; // segment a
    assign DISP[5]=(num==4'h5||num==4'h6||num==4'hb||num==4'hc||num==4'
he||num==4'hf)?1'b0:1'b1; // segment b
    assign DISP[4]=(num==4'h2||num==4'hc||num==4'he||num==4'hf)?1'b0:1'
b1; // segment c
    assign DISP[3]=(num==4'h1||num==4'h4||num==4'h7||num==4'ha||num==4'
hf)?1'b0:1'b1; // segment d
    assign DISP[2]=(num==4'h1||num==4'h3||num==4'h4||num==4'h5||num==4'
h7||num==4'h9)?1'b0:1'b1; // segment e
    assign DISP[1]=(num==4'h1||num==4'h2||num==4'h3||num==4'h7||num==4'
hc||num==4'hd)?1'b0:1'b1; // segment f
    assign DISP[0]=(num==4'h0||num==4'h1||num==4'h7)?1'b0:1'b1; // segm
ent g
endmodule
```

4. 电路验证

a) TestBench

testbench.v（时钟周期 4ns）：

```
`timescale 1ns / 1ps
module testbench(
    );
    reg CLOCK=1'b0,RESET=1'b0; // 时钟和复位信号
    wire LED_RED,LED_GREEN; // 红绿灯 LED 信号
    wire [6:0] DISP; // 七段数码管的 LED 信号
    wire [1:0] DISP_D; // 七段数码管的段选信号
    parameter PERIOD=4,HALF_PERIOD=PERIOD/2; // 时钟周期

    // 生成时钟：
    always begin
        #HALF_PERIOD;
        CLOCK=~CLOCK;
    end
```

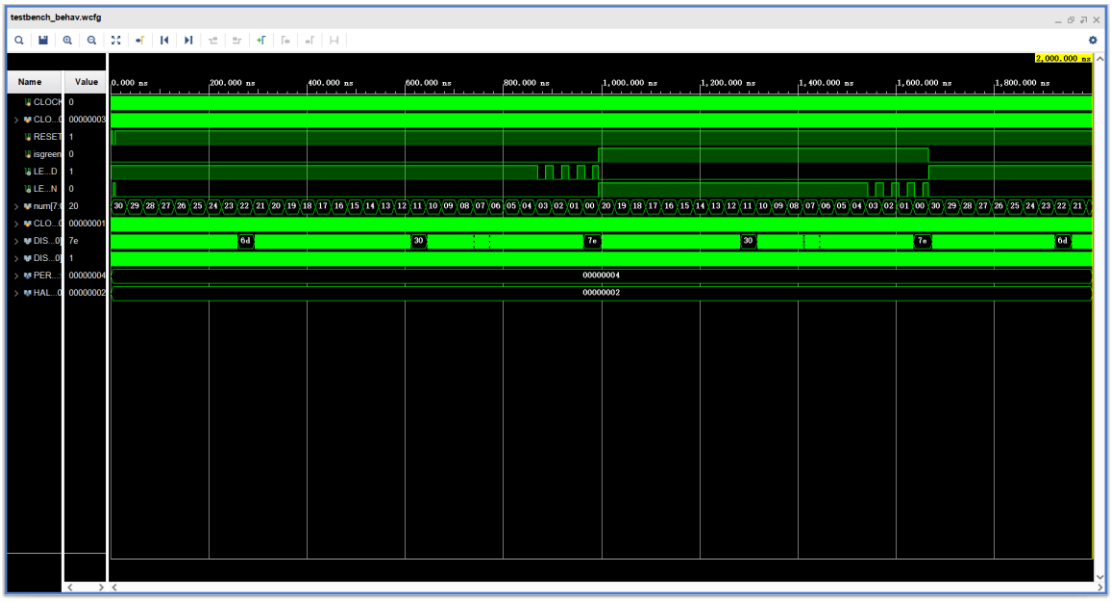
```
    initial begin
        RESET=1'b1;
        #(HALF_PERIOD*5/2);
        RESET=1'b0;
        #PERIOD;
        RESET=1'b1;
        #(PERIOD*500); // 500 个周期后结束仿真
        $finish;
    end

    // 实例化待测模块
    traffic_light traffic_light1(
        .CLOCK(CLOCK),
        .RESET(RESET),
        .LED_RED(LED_RED),
        .LED_GREEN(LED_GREEN),
        .DISP(DISP),
        .DISP_D(DISP_D)
    );
endmodule
```

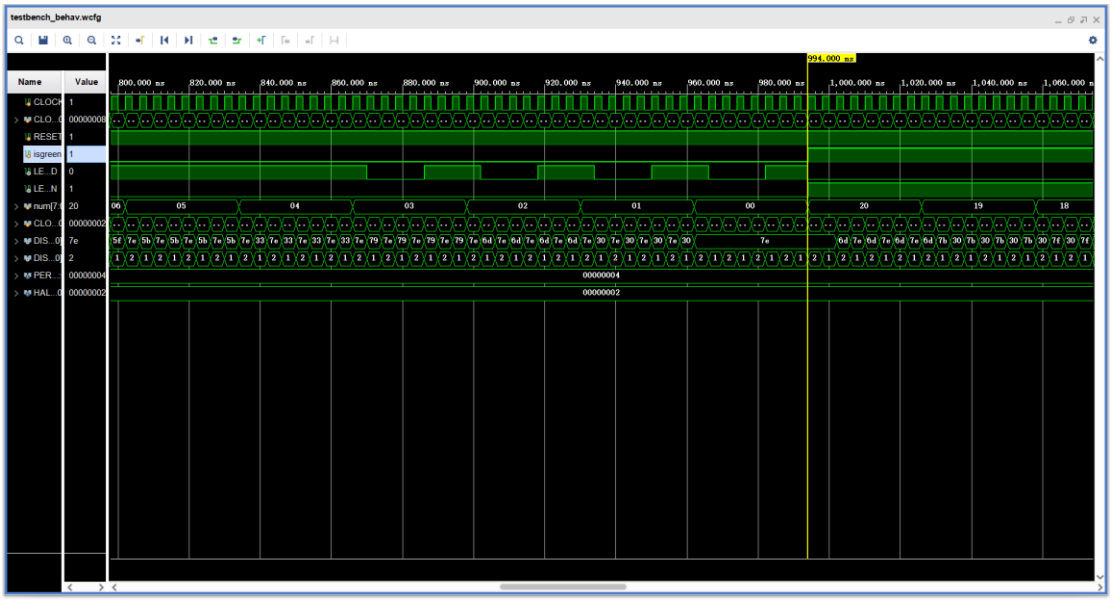
b）仿真结果

仿真时把 main.v 中的 ONE_SECOND 参数设为 8，即每秒等于 8 个时钟周期（每个时钟周期 4ns），把 display_control 中的 CLOCKS_PER_FRAME 设成 2，即每 2 个时钟周期刷新一遍显示的数字。

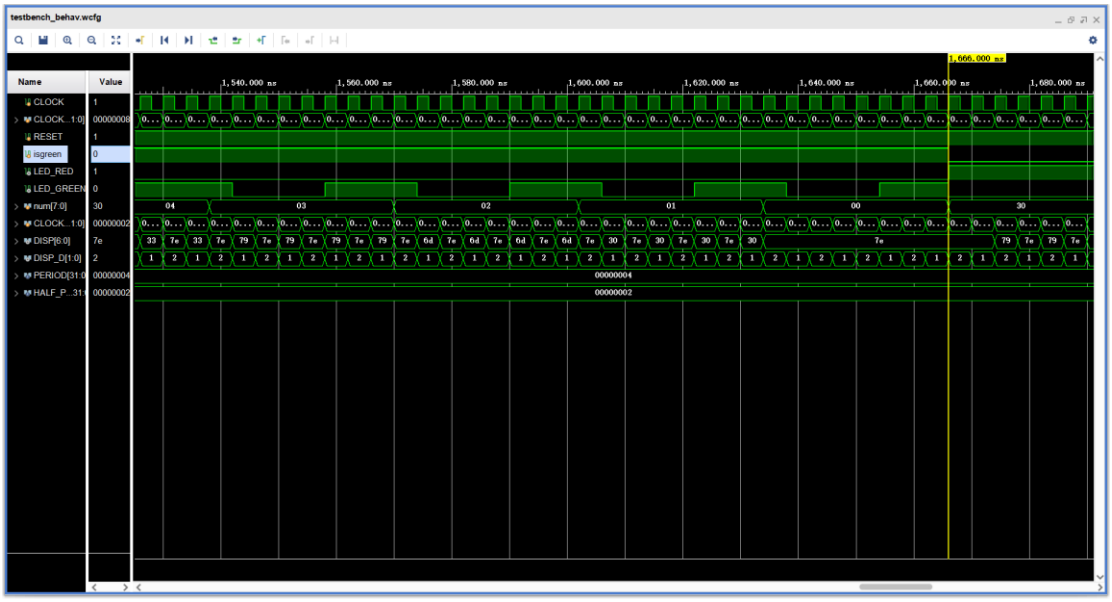
前约 500 个周期的仿真结果：



红灯变绿灯时的放大图：



绿灯变红灯：



可以看到仿真结果是正确的，不管是红灯变绿灯还是绿灯变红灯，变灯之前都闪了 3 秒。

5. 电路上板

仿真时把 main.v 中的 ONE_SECOND 参数设为 100000000，因为 T5 管脚的时钟频率是 100MHz；把 display_control 中的 CLOCKS_PER_FRAME 设成 2000000，即每 1/50 秒刷新一遍显示的数字。

管脚配置：

I/O Ports

Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength
All ports (13)									
DISP (7)	OUT			☑	35	LVCMOS25*	2.500		12
DISP[6]	OUT		B4	☑	35	LVCMOS25*	2.500		12
DISP[5]	OUT		A4	☑	35	LVCMOS25*	2.500		12
DISP[4]	OUT		A3	☑	35	LVCMOS25*	2.500		12
DISP[3]	OUT		B1	☑	35	LVCMOS25*	2.500		12
DISP[2]	OUT		A1	☑	35	LVCMOS25*	2.500		12
DISP[1]	OUT		B3	☑	35	LVCMOS25*	2.500		12
DISP[0]	OUT		B2	☑	35	LVCMOS25*	2.500		12
DISP_D (2)	OUT			☑	35	LVCMOS25*	2.500		12
DISP_D[1]	OUT		C2	☑	35	LVCMOS25*	2.500		12
DISP_D[0]	OUT		G2	☑	35	LVCMOS25*	2.500		12
Scalar ports (4)									
CLOCK	IN		T5	☑	34	LVCMOS25*	2.500		
LED_GRE	OUT		K2	☑	35	LVCMOS25*	2.500		12
LED_RED	OUT		J4	☑	35	LVCMOS25*	2.500		12
RESET	IN		P15	☑	14	LVCMOS25*	2.500		

6. 实验心得

不要在不同的 always 块中对同一个变量赋值，每个 always 块中最好只写一个 if 语句。把不同的功能分成不同的模块，可以重用一些模块。