实验名称：红绿灯 姓名： 学号：

装

订

线

浙江大学

本科实验报告

姓名：

学院：生物医学工程与仪器科学学院

系：生物医学工程系

专业：生物医学工程

学号：

指导教师：余锋

实验名称：红绿灯 姓名：__ 学号：

专业：生物医学工程
姓名：
学号：
日期：2025.5.27
地点：教 6－204

浙江大学 实验报告

课程名称：硬件描述语言 指导老师：余锋 实验名称：HDL 实验七

一、实验要求

二、实验代码与注释

三、仿真结果与分析

四、讨论与心得

装 订 线

一、实验要求

1、系统复位释放后，红灯、绿灯交替闪亮，计数端口持续时间。

具体描述：红灯亮时，计数端口数值从 4 变到 1，当变成 0 时，红灯灭，绿灯亮；绿灯亮时，计数端口数值数值从 8 变到 1，当变成 0 时，红灯亮，绿灯灭。如此循坏。

2、红灯和绿灯各由一个 fsm 控制，例如红灯的 fsm 端口如下

```
input clk_i,              时钟输入  100MHz
input rst_i,              高电平复位
output reg red_o,         红灯控制 1 代表亮 2 代表灭
output reg [3:0] led_o,       亮灯时间
input green_over_i,           握手信号举例  机制可自行设计
input green_start_i,
output reg red_over_o,
output reg red_start_o
```

同理绿灯 fsm 端口

```
input clk_i,
input rst_i,
output reg green_o,
```

实验名称：___红绿灯___ 姓名：__ 学号：

output reg [3:0] led_o,

input red_over_i,

input red_start_i,

output reg green_over_o,

output reg green_start_o

3、新建一个交通灯的设计文件，里面例化两个 fsm，端口定义

input clk_xi,

input rst_xi,

output red_xo,

output green_xo,

output reg [3:0]led_xo 计算端口，红灯亮时显示红灯计数，绿灯亮时显示绿灯计数

4、画出状态转化图 以及 状态机间握手机制

装

订

线

二、实验代码与注释

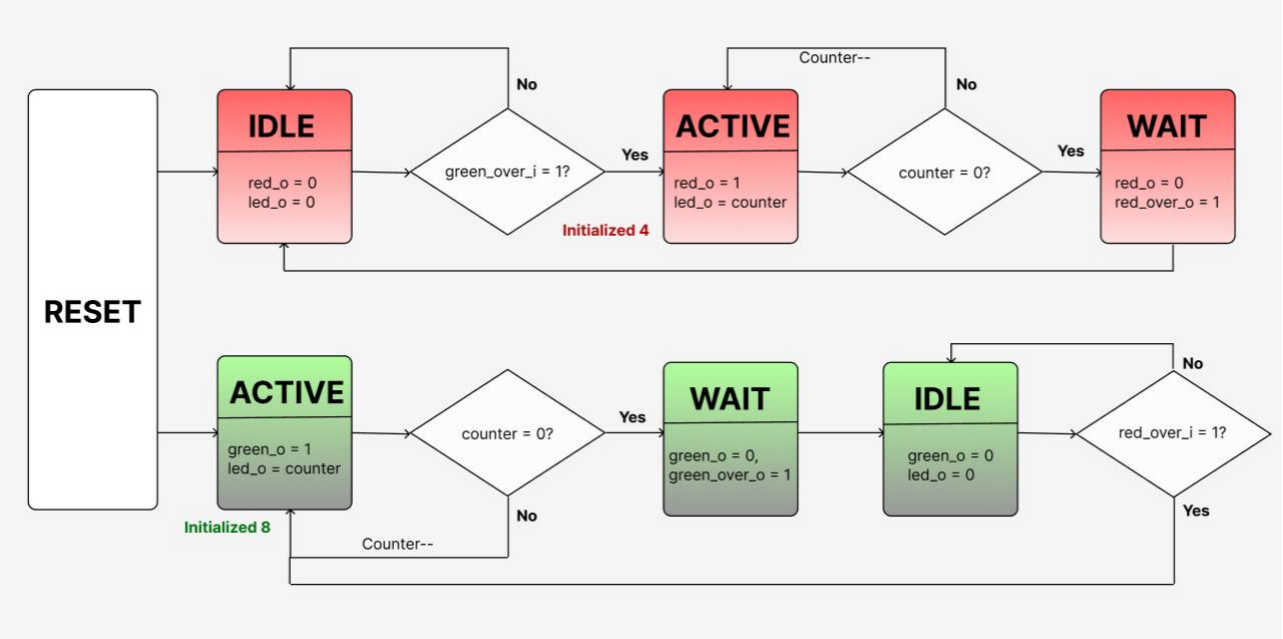


图 1 Figma 所绘制红绿灯 ASM

实验名称：___红绿灯___ 姓名：__ 学号：

装 订 线

交通灯控制系统分为三个主要模块：

red_light_fsm.v - 红灯有限状态机

green_light_fsm.v - 绿灯有限状态机

traffic_light.v - 顶层模块，例化两个 FSM 并处理信号连接

（一）红灯 FSM

```
// 红灯 FSM 模块
module red_light_fsm (
    input clk_i,                // 时钟输入 100MHz
    input rst_i,                // 高电平复位
    output reg red_o,           // 红灯控制 1 代表亮 0 代表灭
    output reg [3:0] led_o,     // 亮灯时间
    input green_over_i,         // 绿灯结束信号
    input green_start_i,        // 绿灯开始信号
    output reg red_over_o,      // 红灯结束信号
    output reg red_start_o      // 红灯开始信号
);

    // 状态定义
    parameter IDLE = 2'b00;     // 空闲状态
    parameter ACTIVE = 2'b01;   // 红灯亮状态
    parameter WAIT = 2'b10;     // 等待状态

    reg [1:0] current_state, next_state;
    reg [15:0] counter;         // 简单计数器：1000 个时钟周期递减一次
    reg [3:0] time_counter;     // 时间计数器

    // 状态转换
    always @(posedge clk_i or posedge rst_i) begin
        if (rst_i) begin
            current_state <= IDLE;
            counter <= 0;
            time_counter <= 0;
        end else begin
            current_state <= next_state;

            // 每1000 个时钟周期递减一次
            if (counter >= 16'd999) begin
                counter <= 0;
```

实验名称：___红绿灯___ 姓名：__ 学号：

```
            if (current_state == ACTIVE && time_counter > 0) begin
                time_counter <= time_counter - 1;
            end
        end else begin
            counter <= counter + 1;
        end

        // 初始化时间计数器
        if (current_state == IDLE && next_state == ACTIVE) begin
            time_counter <= 4;
        end
    end
end

// 状态机逻辑
always @(*) begin
    case (current_state)
        IDLE: begin
            if (green_over_i) begin
                next_state = ACTIVE;
            end else begin
                next_state = IDLE;
            end
        end

        ACTIVE: begin
            if (time_counter == 0) begin
                next_state = WAIT;
            end else begin
                next_state = ACTIVE;
            end
        end

        WAIT: begin
            next_state = IDLE;
        end

        default: next_state = IDLE;
    endcase
end

// 输出逻辑
always @(posedge clk_i or posedge rst_i) begin
    if (rst_i) begin
```

装 订 线

实验名称：___红绿灯___ 姓名：__ 学号：

```
            red_o <= 0;
            led_o <= 0;
            red_over_o <= 0;
            red_start_o <= 0;
        end else begin
            case (current_state)
                IDLE: begin
                    red_o <= 0;
                    led_o <= 0;
                    red_over_o <= 0;
                    if (green_over_i) begin
                        red_start_o <= 1;
                    end else begin
                        red_start_o <= 0;
                    end
                end

                ACTIVE: begin
                    red_o <= 1;
                    led_o <= time_counter;  // 显示4→3→2→1
                    red_start_o <= 0;
                    red_over_o <= 0;
                end

                WAIT: begin
                    red_o <= 0;
                    led_o <= 0;
                    red_over_o <= 1;
                    red_start_o <= 0;
                end

                default: begin
                    red_o <= 0;
                    led_o <= 0;
                    red_over_o <= 0;
                    red_start_o <= 0;
                end
            endcase
        end
    end
endmodule
```

装

订

线

实验名称：___红绿灯___ 姓名：__ 学号：

装

订

线

（二）绿灯 FSM

```
// 绿灯 FSM 模块
module green_light_fsm (
    input clk_i,                // 时钟输入 100MHz
    input rst_i,                // 高电平复位
    output reg green_o,         // 绿灯控制 1 代表亮 0 代表灭
    output reg [3:0] led_o,     // 亮灯时间
    input red_over_i,           // 红灯结束信号
    input red_start_i,          // 红灯开始信号
    output reg green_over_o,    // 绿灯结束信号
    output reg green_start_o    // 绿灯开始信号
);

    // 状态定义
    parameter IDLE = 2'b00;     // 空闲状态
    parameter ACTIVE = 2'b01;   // 绿灯亮状态
    parameter WAIT = 2'b10;     // 等待状态

    reg [1:0] current_state, next_state;
    reg [15:0] counter;         // 简单计数器：1000 个时钟周期递减一次
    reg [3:0] time_counter;     // 时间计数器

    // 状态转换
    always @(posedge clk_i or posedge rst_i) begin
        if (rst_i) begin
            current_state <= ACTIVE;  // 复位后绿灯先亮
            counter <= 0;
            time_counter <= 8;        // 绿灯初始时间为8
        end else begin
            current_state <= next_state;

            // 每1000 个时钟周期递减一次
            if (counter >= 16'd999) begin
                counter <= 0;
                if (current_state == ACTIVE && time_counter > 0) begin
                    time_counter <= time_counter - 1;
                end
            end else begin
                counter <= counter + 1;
            end

            // 初始化时间计数器
```

实验名称：___红绿灯___ 姓名：__ 学号：

```
            if (current_state == IDLE && next_state == ACTIVE) begin
                time_counter <= 8;
            end
        end
    end

    // 状态机逻辑
    always @(*) begin
        case (current_state)
            IDLE: begin
                if (red_over_i) begin
                    next_state = ACTIVE;
                end else begin
                    next_state = IDLE;
                end
            end

            ACTIVE: begin
                if (time_counter == 0) begin
                    next_state = WAIT;
                end else begin
                    next_state = ACTIVE;
                end
            end

            WAIT: begin
                next_state = IDLE;
            end

            default: next_state = IDLE;
        endcase
    end

    // 输出逻辑
    always @(posedge clk_i or posedge rst_i) begin
        if (rst_i) begin
            green_o <= 1;           // 复位后绿灯先亮
            led_o <= 8;            // 显示初始计数
            green_over_o <= 0;
            green_start_o <= 1;     // 复位后绿灯开始
        end else begin
            case (current_state)
                IDLE: begin
                    green_o <= 0;
```

装 订 线

实验名称：___红绿灯___ 姓名：__ 学号：

```
                    led_o <= 0;
                    green_over_o <= 0;
                    if (red_over_i) begin
                        green_start_o <= 1;
                    end else begin
                        green_start_o <= 0;
                    end
                end

                ACTIVE: begin
                    green_o <= 1;
                    led_o <= time_counter;  // 显示8→7→6→5→4→3→2→1
                    green_start_o <= 0;
                    green_over_o <= 0;
                end

                WAIT: begin
                    green_o <= 0;
                    led_o <= 0;
                    green_over_o <= 1;
                    green_start_o <= 0;
                end

                default: begin
                    green_o <= 0;
                    led_o <= 0;
                    green_over_o <= 0;
                    green_start_o <= 0;
                end
            endcase
        end
    end
endmodule
```

（三）顶层模块

```
// 交通灯顶层模块
module traffic_light (
    input clk_xi,               // 时钟输入
    input rst_xi,               // 复位输入
    output red_xo,              // 红灯输出
    output green_xo,            // 绿灯输出
```

装 订 线

实验名称：___红绿灯___ 姓名：__ 学号：

```
    output reg [3:0] led_xo     // 计数端口，红灯亮时显示红灯计数，绿灯亮时显示绿灯计数
);

    // 内部信号定义
    wire red_light_out;
    wire green_light_out;
    wire [3:0] red_led_count;
    wire [3:0] green_led_count;

    // 握手信号
    wire red_over, red_start;
    wire green_over, green_start;

    // 例化红灯 FSM
    red_light_fsm red_fsm_inst (
        .clk_i(clk_xi),
        .rst_i(rst_xi),
        .red_o(red_light_out),
        .led_o(red_led_count),
        .green_over_i(green_over),
        .green_start_i(green_start),
        .red_over_o(red_over),
        .red_start_o(red_start)
    );

    // 例化绿灯 FSM
    green_light_fsm green_fsm_inst (
        .clk_i(clk_xi),
        .rst_i(rst_xi),
        .green_o(green_light_out),
        .led_o(green_led_count),
        .red_over_i(red_over),
        .red_start_i(red_start),
        .green_over_o(green_over),
        .green_start_o(green_start)
    );

    // 输出赋值
    assign red_xo = red_light_out;
    assign green_xo = green_light_out;

    // 计数端口逻辑：红灯亮时显示红灯计数，绿灯亮时显示绿灯计数
    always @(*) begin
```

装 订 线

实验名称：___红绿灯___ 姓名：__ 学号：

```
        if (red_light_out) begin
            led_xo = red_led_count;
        end else if (green_light_out) begin
            led_xo = green_led_count;
        end else begin
            led_xo = 4'b0000;
        end
    end
endmodule
```

（四）Testbench

```
// 交通灯测试平台
`timescale 1ns / 1ps

module traffic_light_tb;
    // 测试信号
    reg clk_xi;
    reg rst_xi;
    wire red_xo;
    wire green_xo;
    wire [3:0] led_xo;

    // 例化被测试模块
    traffic_light uut (
        .clk_xi(clk_xi),
        .rst_xi(rst_xi),
        .red_xo(red_xo),
        .green_xo(green_xo),
        .led_xo(led_xo)
    );

    // 时钟生成 (100MHz)
    initial begin
        clk_xi = 0;
        forever #5 clk_xi = ~clk_xi;  // 10ns 周期 = 100MHz
    end

    initial begin

        $dumpfile("traffic_light.vcd");
        $dumpvars(0, traffic_light_tb);
```

装 订 线

实验名称：___红绿灯___ 姓名：__ 学号：

```
        // 初始化
        rst_xi = 1;
        #100;
        rst_xi = 0;
        // 每1000个时钟周期递减一次，总共需要(8+4)*1000=12000个周期
        // 观察2个周期需要24000个周期，约240us
        #250000;  // 250us，足够观察多个完整周期

        $finish;
    end

    // 监控输出 - 每1000个时钟周期输出一次
    always @(posedge clk_xi) begin
        if (uut.red_fsm_inst.counter == 0 || uut.green_fsm_inst.counter ==
0) begin
            $display("Time: %0t, Reset: %b, Red: %b, Green: %b, Count: %d",
                     $time, rst_xi, red_xo, green_xo, led_xo);
        end
    end
endmodule
```

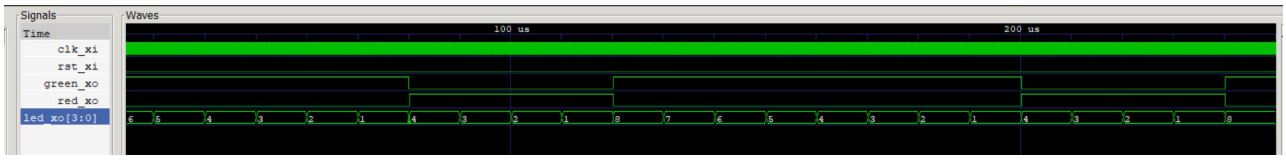
装 订 线

三、仿真结果与分析

仿真工具依然选择 iverilog + GTKWave。

（一）时序波形分析

根据 iverilog 仿真输出结果，我们可以观察到以下时序行为：



1、系统初始化：

复位释放后，绿灯（green_xo）首先亮起，计数显示 8；

2、绿灯阶段（8 个时间单位）：

实验名称：___红绿灯___ 姓名：__ 学号：

计数（led_xo）从 8 递减到 1，每个时间单位递减 1，持续亮 8 个时间周期，计数变为 0 时绿灯熄灭。

3、红绿灯切换：

绿灯熄灭时，红灯(red_xo)立即亮起，无重叠时间。握手机制确保了状态的正确切换。红灯计数从 4 开始显示

4、红灯阶段（4 个时间单位）：

计数从 4 递减到 1，每个时间单位递减 1，红灯持续亮 4 个时间周;计数变为 0 时红灯熄灭，绿灯重新亮起.

绿灯 8 个时间单位 + 红灯 4 个时间单位 =12 个时间单位为一个完整周期,且系统能够持续循环运行。

（二）握手机制验证

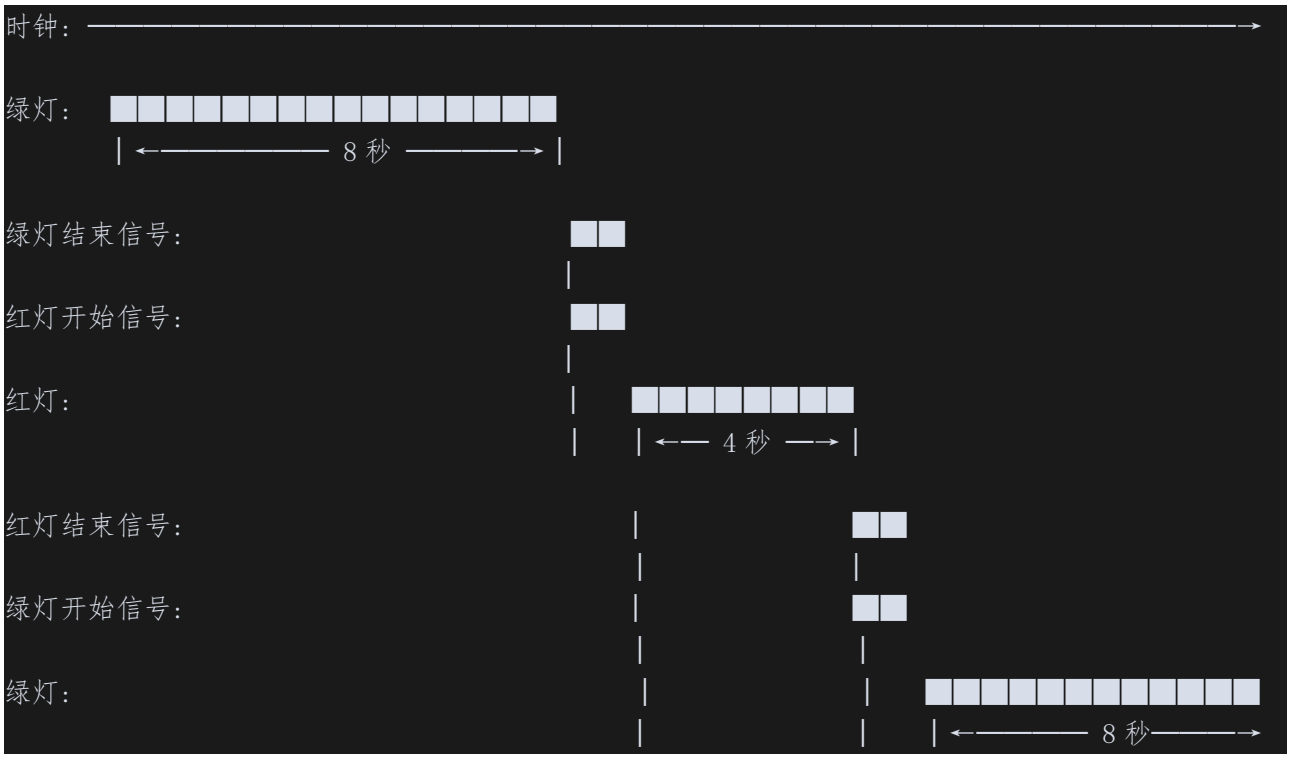
从仿真结果可以验证握手机制的正确性：

（1）无冲突切换：红灯和绿灯从未同时亮起

（2）无缺失状态：每次切换都能正确进入下一个状态

（3）计数同步：计数端口始终显示当前亮灯的剩余时间

（4）状态维持：每个状态都能维持正确的时间长度



实验名称：___红绿灯___ 姓名：__ 学号：

四、讨论与心得

本次实验采用模块化设计思想，将交通灯控制系统分解为红灯 FSM、绿灯 FSM 和顶层控制模块三个部分。在状态机设计方面，采用了三段式状态机设计方法，将时序逻辑、组合逻辑和输出逻辑分离。时序逻辑部分使用非阻塞赋值（<=）处理状态转换和计数器更新，组合逻辑部分使用阻塞赋值（=）处理状态转换条件判断，输出逻辑部分根据当前状态产生相应的输出信号。

握手机制的设计是本实验的核心技术要点。通过 red_over_o/green_over_o 信号和 red_start_o/green_start_o 信号实现两个 FSM 之间的可靠通信。每个 FSM 都设置了 WAIT 状态作为握手缓冲，确保握手信号能够正确传递。这种设计避免了两个独立 FSM 可能出现的竞争条件和状态冲突问题。

装

订

线

通过本次实验，我深入理解了有限状态机（FSM）的设计方法和应用场景。FSM 是数字系统设计中的重要工具，特别适用于需要按照特定时序执行的控制逻辑。掌握了状态编码、状态转换条件设计、输出逻辑设计等关键技术要点。三段式状态机的设计方法不仅提高了代码的可读性和可维护性，也有效避免了常见的设计错误。