实验四 状态机设计与 FPGA 实现

学号 2023370018
姓名 梁桐
班级 10012209
课程代码 U10M21001.01

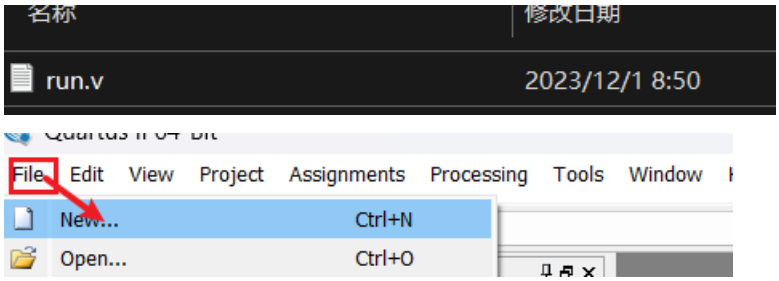
一．实验目的：

1． 掌握可综合 Verilog 语言进行状态机设计及测试验证；
2． 学习如何在 FPGA 进行设计实现。

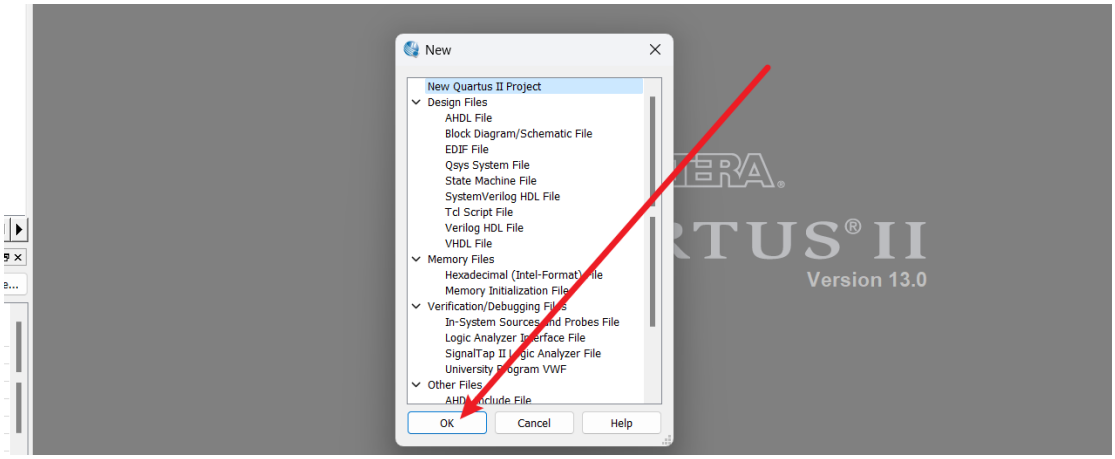
二．实验内容：

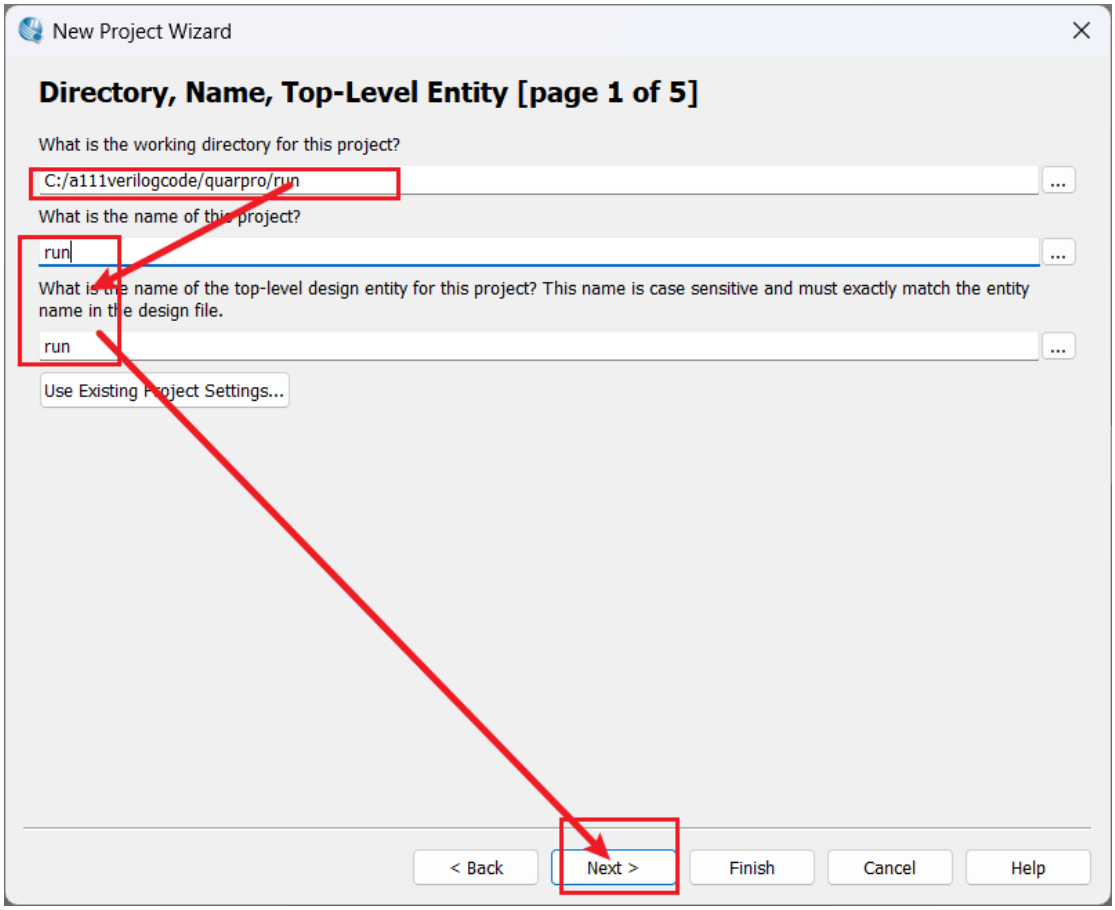
A. QuartusII 软件基本使用步骤下

(1).新建一个文件夹，将 shared.v 文件放入文件夹中；双击运行 Quartus II 13.0sp1 (64-bit)，单击 Fire，单击 new

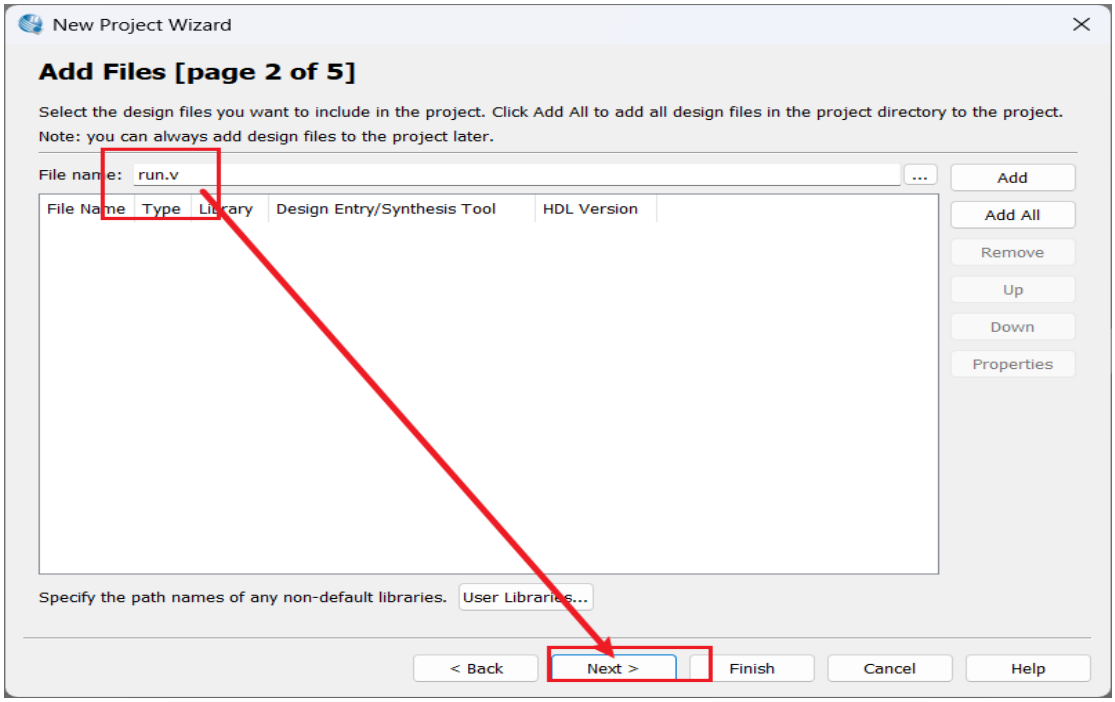


(2)单击 OK，单击 Next，输入刚才创建的文件的地址，将下面两个红框中的名称改为 run(与设计文件的 model 名称一致)，单击 Next

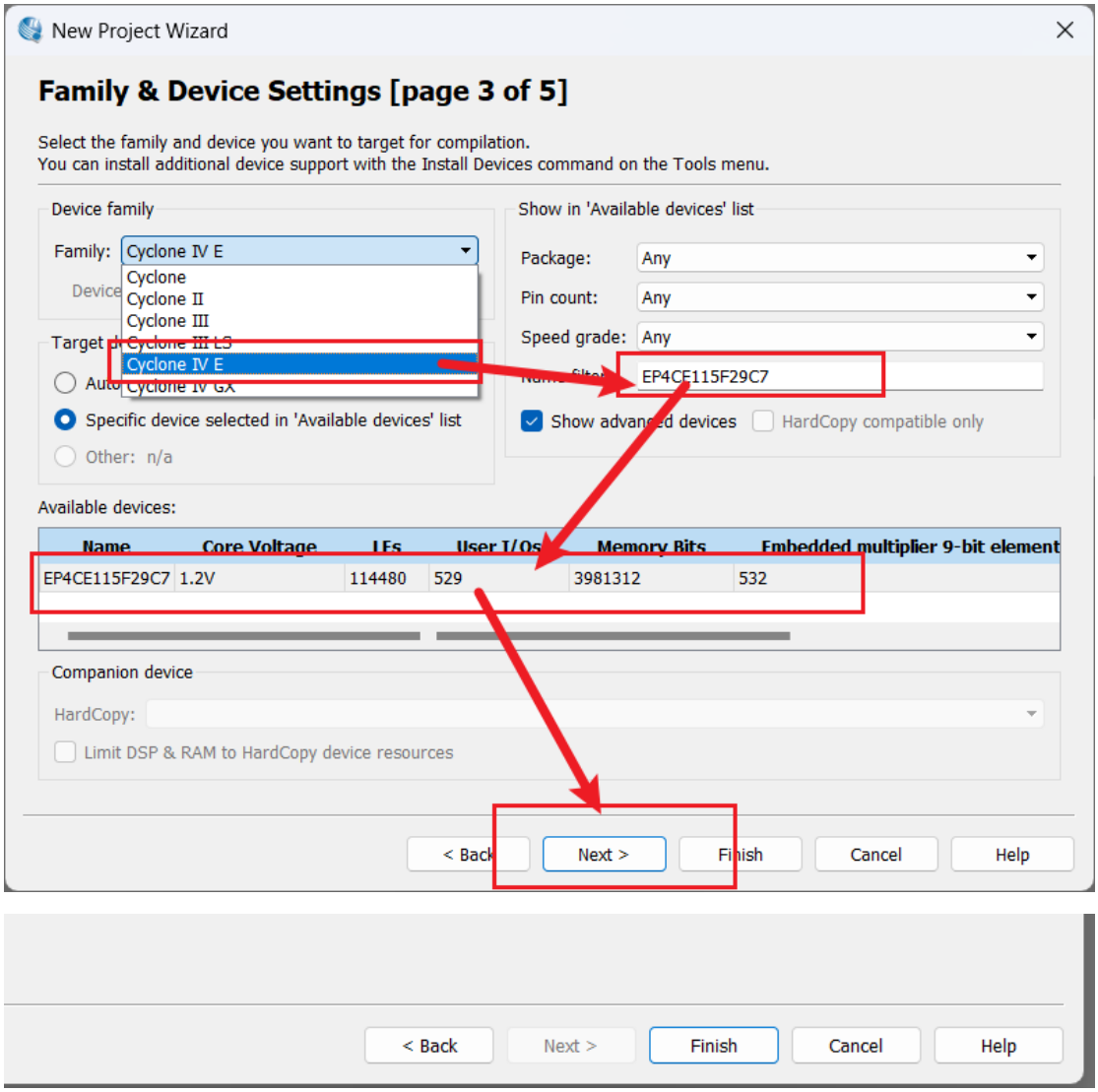




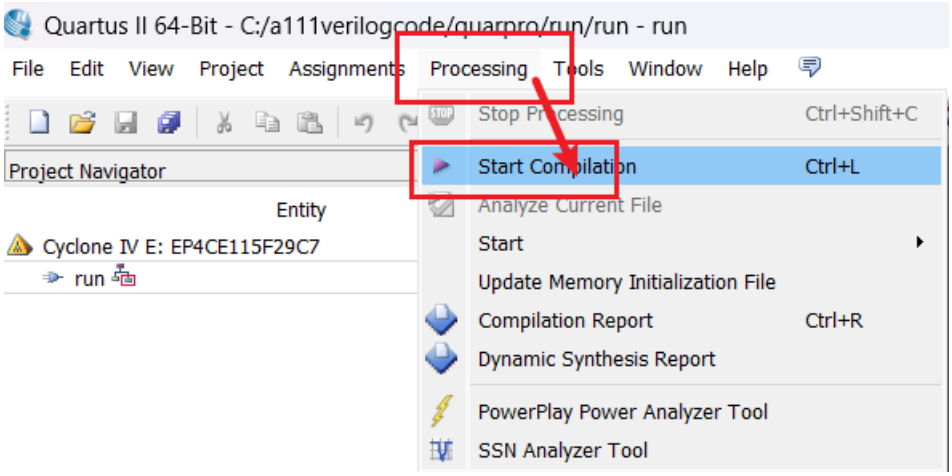
（3）File name 选项选中 run.v 文件，点击 Next，



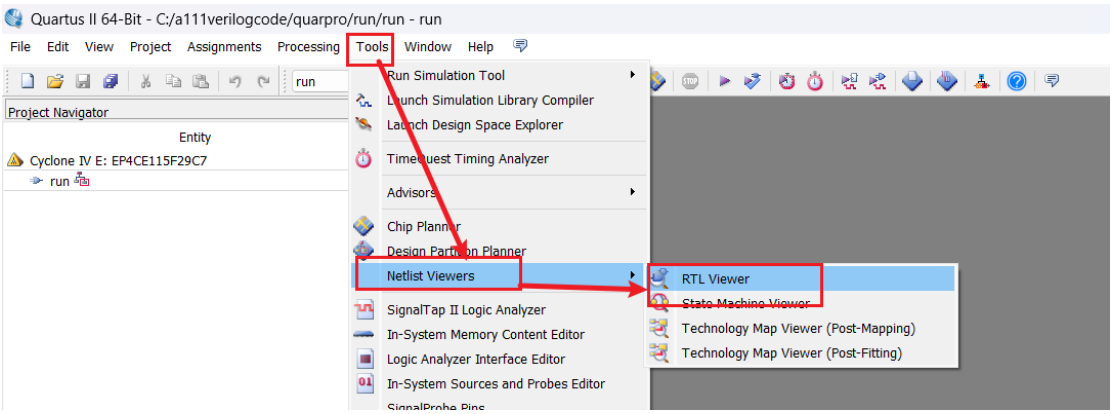
(4)Family 选项选择 Cyclone IV E,右侧输入 EP4CE115F29C7，选中之后点击 Next，Next，Finish



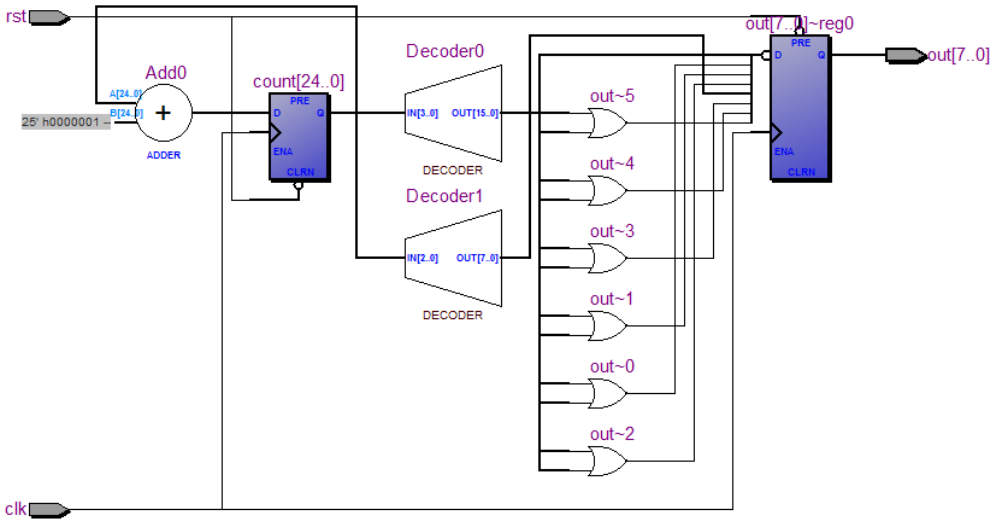
(5)单击 Processing,再单击 Start Compilation



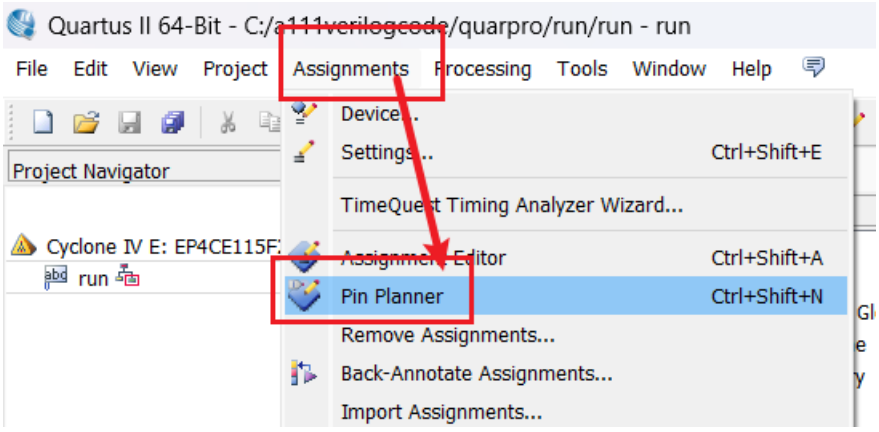
(6)单击 Tools，再单击 Netlist Viewers,再单击 RTL Viewer



(7)生成电路图

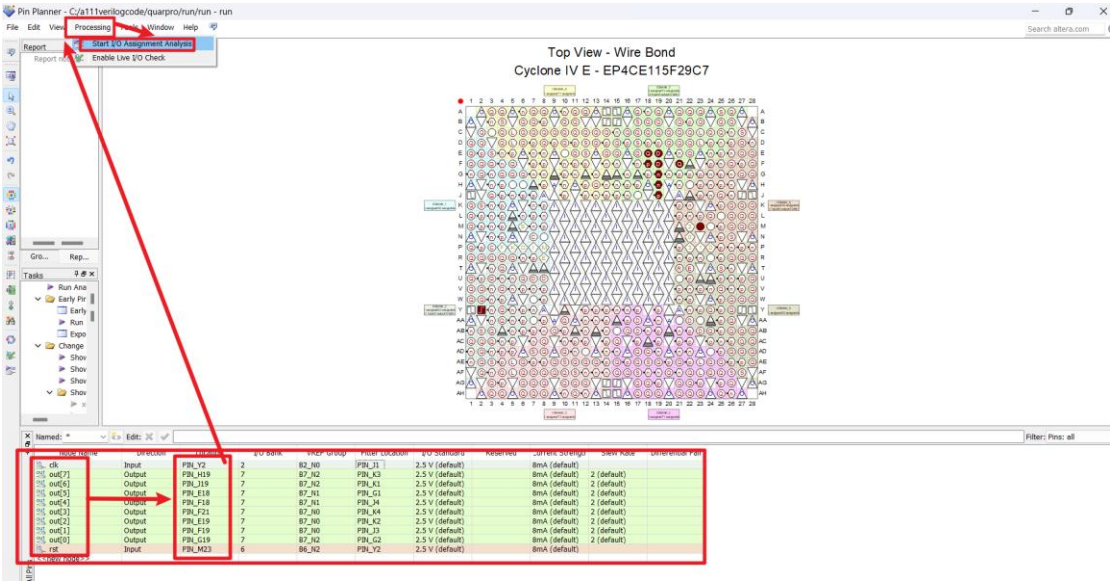


（8）使用 Assignment->pin planner 将设计的全部输入/输出接口与开发板的对应管脚进行一一对应，全编译生成可下载文件（*.sof）Processing->Start Compilation

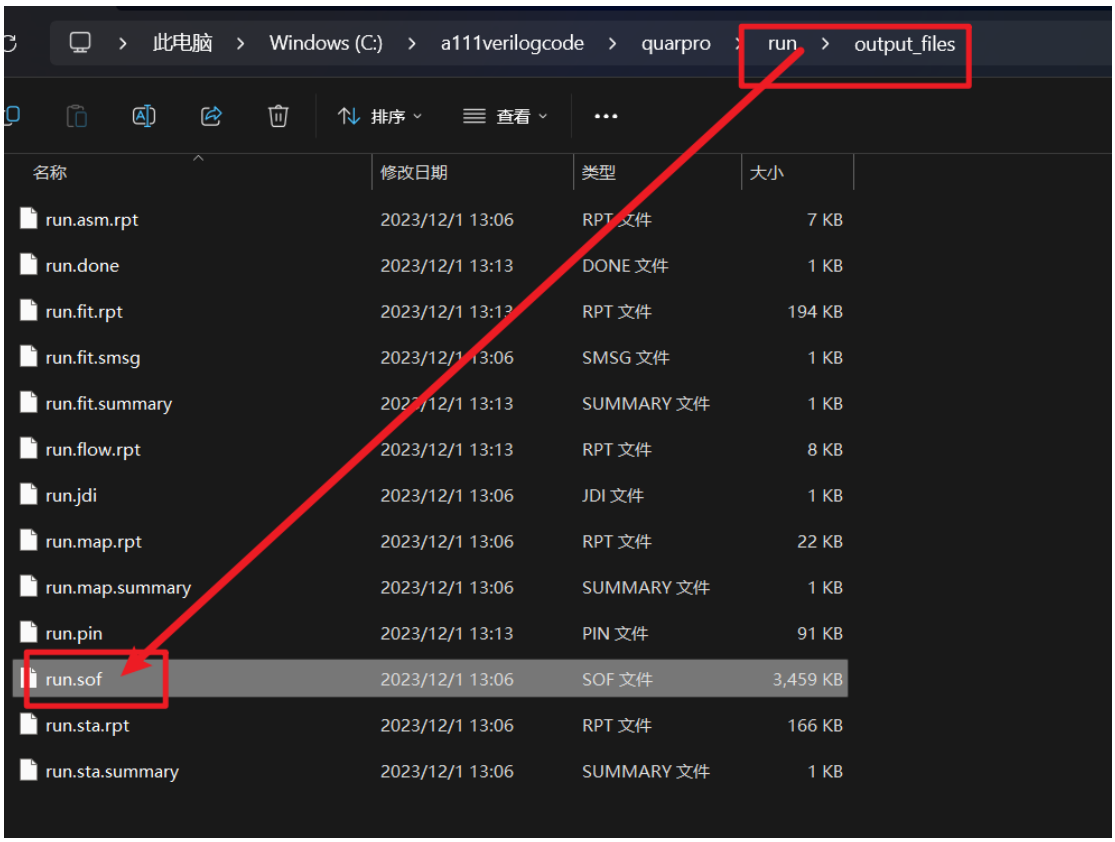


PIN_Y2 -to clk
PIN_H19 -to out[7]
PIN_J19 -to out[6]
PIN_E18 -to out[5]
PIN_F18 -to out[4]

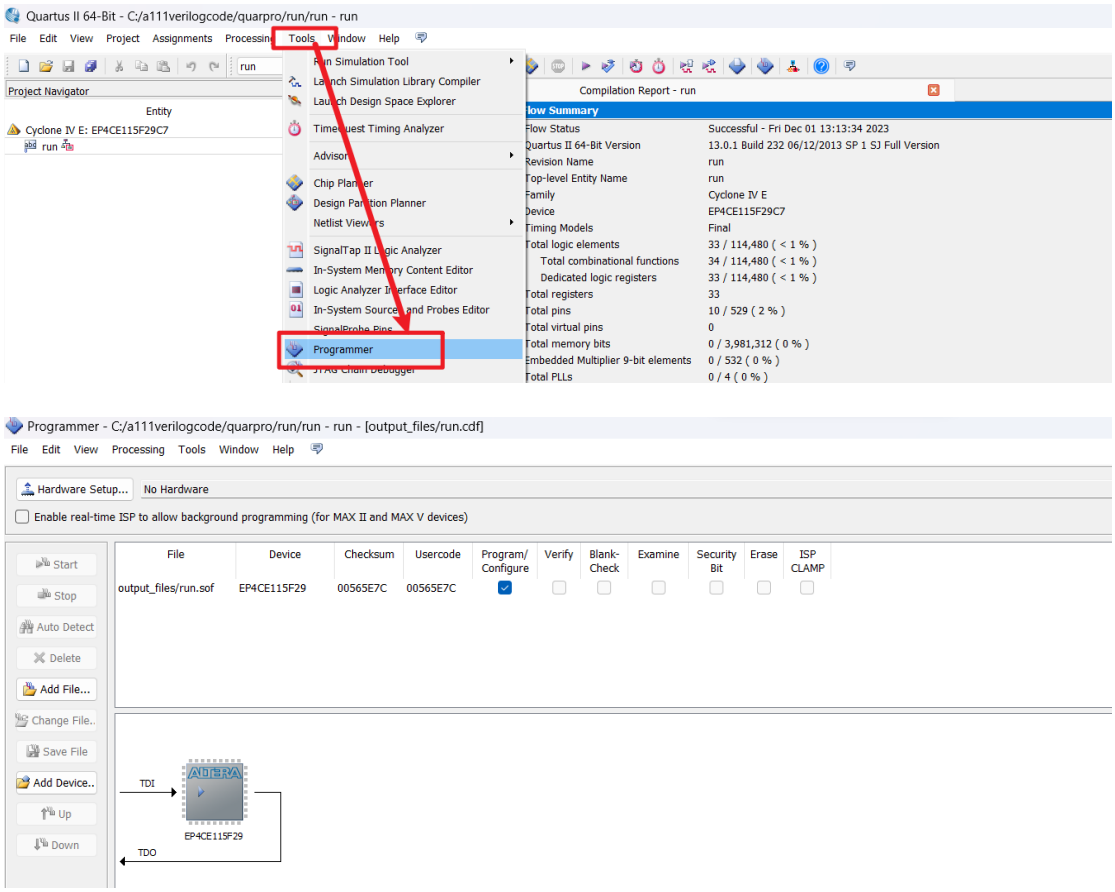
PIN_F21 -to out[3]
PIN_E19 -to out[2]
PIN_F19 -to out[1]
PIN_G19 -to out[0]
PIN_M23 -to rst



（9）在文件夹中可查找到 run.sof 文件



（10）点击 Tools,单击 Programmer 进行程序下载。



B.

1. 跑马灯设计及 FPGA 实现（run.v）

代码

```
module run(clk, rst, out);
    input clk, rst;
    output [7:0] out;
    reg [7:0] out;
    reg [24:0] count;

    // 计数器逻辑
    always @(posedge clk or negedge rst)
        if (!rst)
            count <= 25'b0;
        else
            count <= count + 1;
```

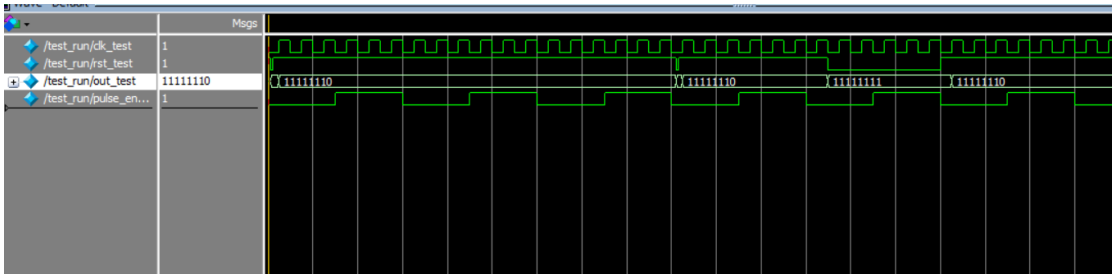
```
// 输出逻辑
    always @(posedge clk or negedge rst)
        if (!rst)
            out <= 8'hff;
        else
            case (count[24:21])
                0:    out <= 8'b1111_1110;
                1:    out <= 8'b1111_1101;
                2:    out <= 8'b1111_1011;
                3:    out <= 8'b1111_0111;
                4:    out <= 8'b1110_1111;
                5:    out <= 8'b1101_1111;
                6:    out <= 8'b1011_1111;
                7:    out <= 8'b0111_1111;
                8:    out <= 8'b1011_1111;
                9:    out <= 8'b1101_1111;
                10: out <= 8'b1110_1111;
                11: out <= 8'b1111_0111;
                12: out <= 8'b1111_1011;
                13: out <= 8'b1111_1101;
                14: out <= 8'b1111_1110;
                15: out <= 8'b1111_1111;
            endcase
endmodule
```

测试代码

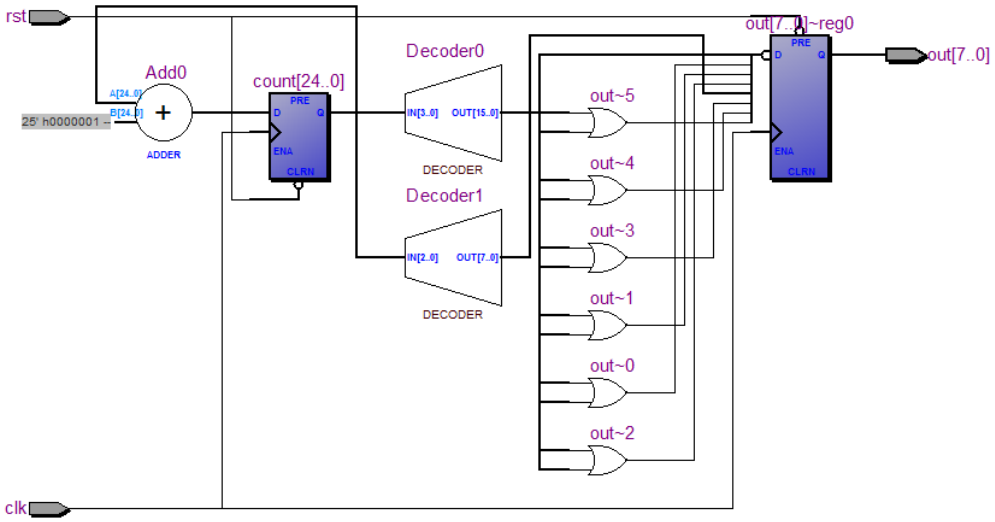
```
`timescale 1ns/1ps

module test_run;
    reg clk_test;              // 时钟信号
    reg rst_test;              // 复位信号
    wire [7:0] out_test; // 输出信号

    // 时钟信号生成
    initial begin
        clk_test = 0;
        forever #5 clk_test = ~clk_test; // 时钟周期为 10 个时间单位
    end

    // 复位信号初始化
    initial begin
        rst_test = 1;
```

```
        rst_test = 1;
        #1 rst_test = 0;
        #1 rst_test = 1;
        #180 rst_test = 0;
        #1 rst_test = 1;
    end

    // 生成脉冲信号
    reg pulse_enable = 0;
    always #30 pulse_enable = ~pulse_enable;

    // 实例化 topic41 模块
    run UUT_run (
        .clk(clk_test),
        .rst(rst_test),
        .out(out_test)
    );

    // 在仿真中输出时钟和复位信号
    always begin
        #5 $display("Time=%0t, clk=%b, rst=%b", $time, clk_test, rst_test);
    end

    // 在仿真中输出模块的输出信号
    always @(posedge clk_test) begin
        $display("Time=%0t, out=%h", $time, out_test);
    end

    // 在仿真中输出脉冲信号
    always @(posedge clk_test) begin
        if (pulse_enable)
            $display("Time=%0t, Pulse Signal: %b", $time, pulse_enable);
    end

    // 额外的测试用例
    initial begin
        #200 $display("Additional Test Case: Wait for 200 time units");
        #50;
        rst_test = 0;
        #50;
        rst_test = 1;
        #100 $finish;
    end
endmodule
```

波形截图



RTL 图形



2. 有限状态机设计（教材 Figure 6.86）

代码

```
module Sequence(Clock, Resetn, w, z);
    input Clock, Resetn, w;
    output z;
    reg [3:1] y, Y;
    parameter [3:1] A = 3'b000, B = 3'b001, C = 3'b010, D = 3'b011, E = 3'b100;

    // 定义下一个状态的组合电路
    always @(w, y)
        case (y)
            A: if (w) Y = D;
                else Y = B;
            B: if (w) Y = D;
                else Y = C;
            C: if (w) Y = D;
                else Y = C;
            D: if (w) Y = E;
                else Y = B;
```

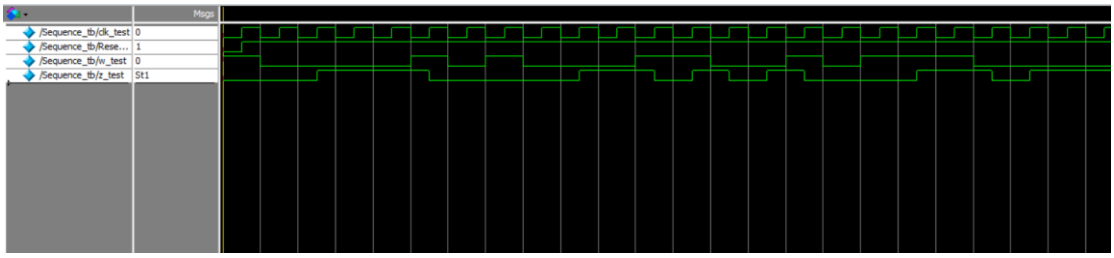
```
            E: if (w) Y = E;
                else Y = B;
            default: Y = 3'bxxx;
        endcase

    // 在时钟上升沿和负边沿复位时更新状态
    always @(negedge Resetn, posedge Clock)
        if (Resetn == 0) y <= A;
        else y <= Y;

    // 将输出 z 与状态 C 和 E 相关联
    assign z = (y == C) || (y == E);
endmodule
```

测试代码

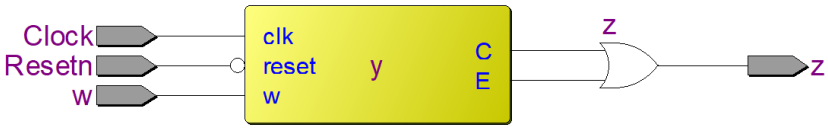
```
module Sequence_tb;
    reg clk_test, Resetn_test, w_test;
    wire z_test;

    initial begin
        clk_test = 0;
        Resetn_test = 0;
        w_test = 1;
    end

    always #10 clk_test    = ~clk_test ;

    initial begin
        #10;
        Resetn_test = 1;
        w_test = 1;
        #10 w_test = 0;
        #20 w_test = 0;
        #20 w_test = 0;
        #20 w_test = 0;
        #20 w_test = 1;
        #20 w_test = 0;
        #20 w_test = 1;
        #20 w_test = 0;
        #20 w_test = 0;
        #20 w_test = 0;
```

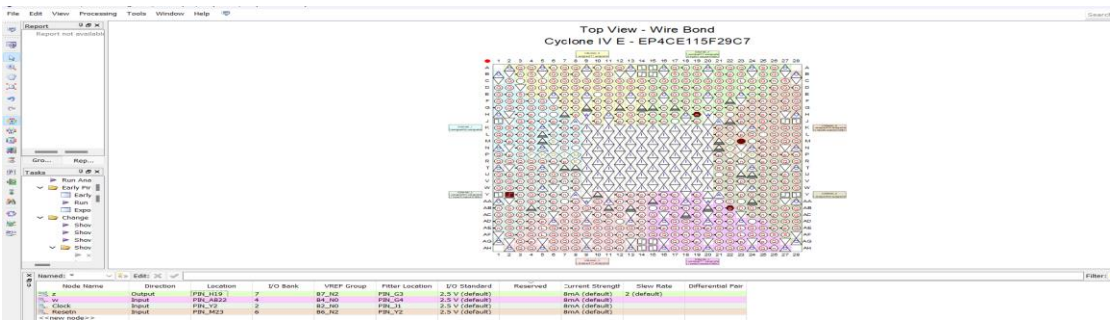
```
        #20 w_test = 1;
        #20 w_test = 1;
        #20 w_test = 0;
        #20 w_test = 0;
        #20 w_test = 1;
        #20 w_test = 0;
        #20 w_test = 1;
        #20 w_test = 1;
        #20 w_test = 1;
        #20 w_test = 0;
    end

    Sequence UUT_Sequence (
        .Clock(clk_test),
        .Resetn(Resetn_test),
        .w(w_test),
        .z(z_test)
    );
endmodule
```

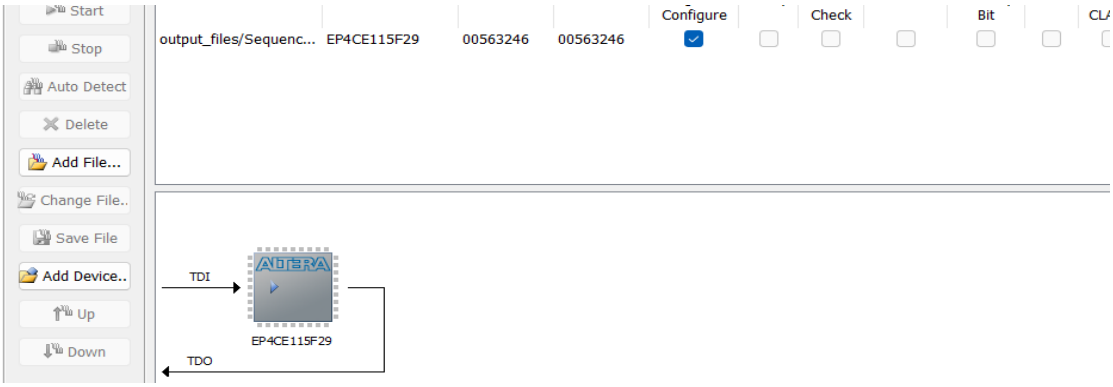
波形截图



RTL 图形







三．实验收获与心得

本次实验主要进行了Verilog进行状态机设计及测试验证和FPGA设计实现。我有以下心得:
管脚绑定是 FPGA 设计中非常关键的一步，确保设计正确地映射到硬件上。使用 Quartus 的 Pin Planner 工具可以方便地将设计中的输入/输出接口与 FPGA 的物理管脚进行对应。确保管脚绑定正确，是解决实际硬件问题的关键一步。
将代码的执行效果反映到硬件上的重要性不可忽视。实践是巩固理论知识、培养技能的最佳途径。通过不断地将设计下载到实际硬件上，你能够更深入地理解数字电路的行为，更好地发现和解决问题。
Quartus II 作为一款强大的 FPGA 设计工具，在本次实验中展现了其全方位的功能。通过这次实验，我不仅学到了状态机设计和 Verilog 语言的实际应用，还掌握了使用 Quartus II 进行 FPGA 设计、仿真和实际运行的全过程。这将为我未来在数字电路设计和 FPGA 应用领域提供坚实的基础。