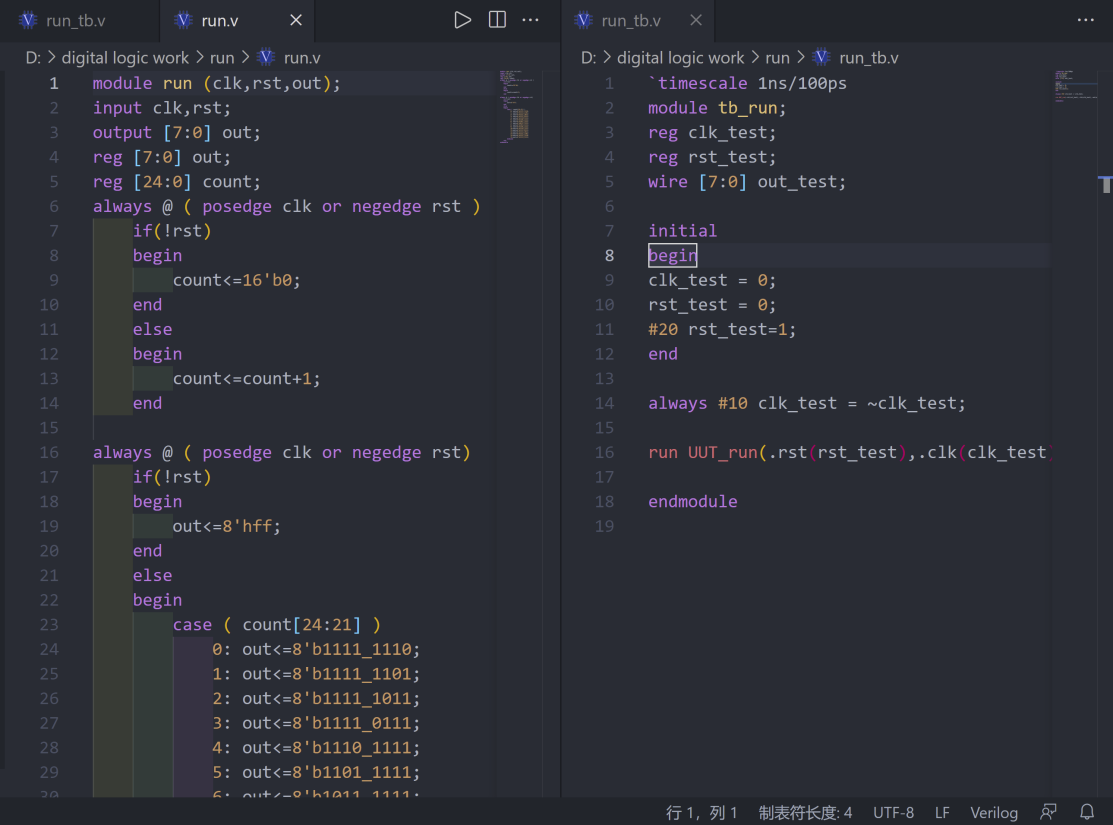
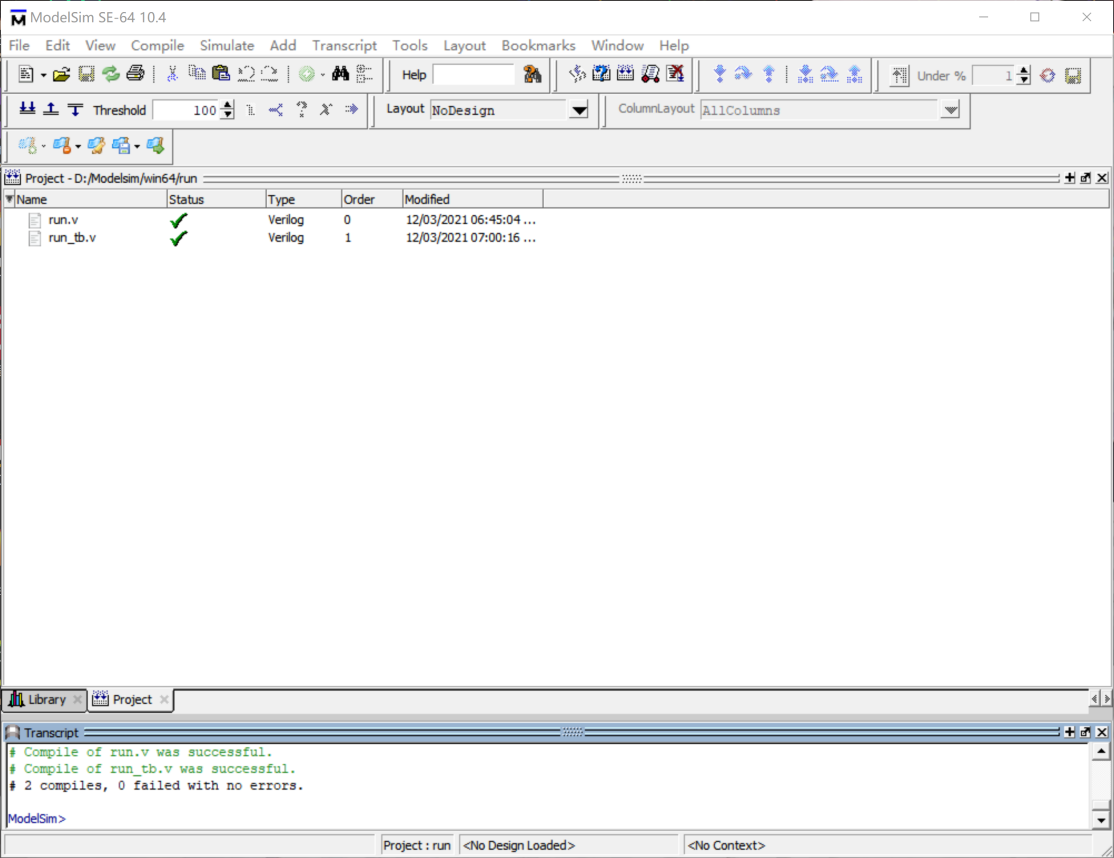
# 数字电路实验四报告

1. **QuartusII软件基本使用步骤（下）**

第一步：编码

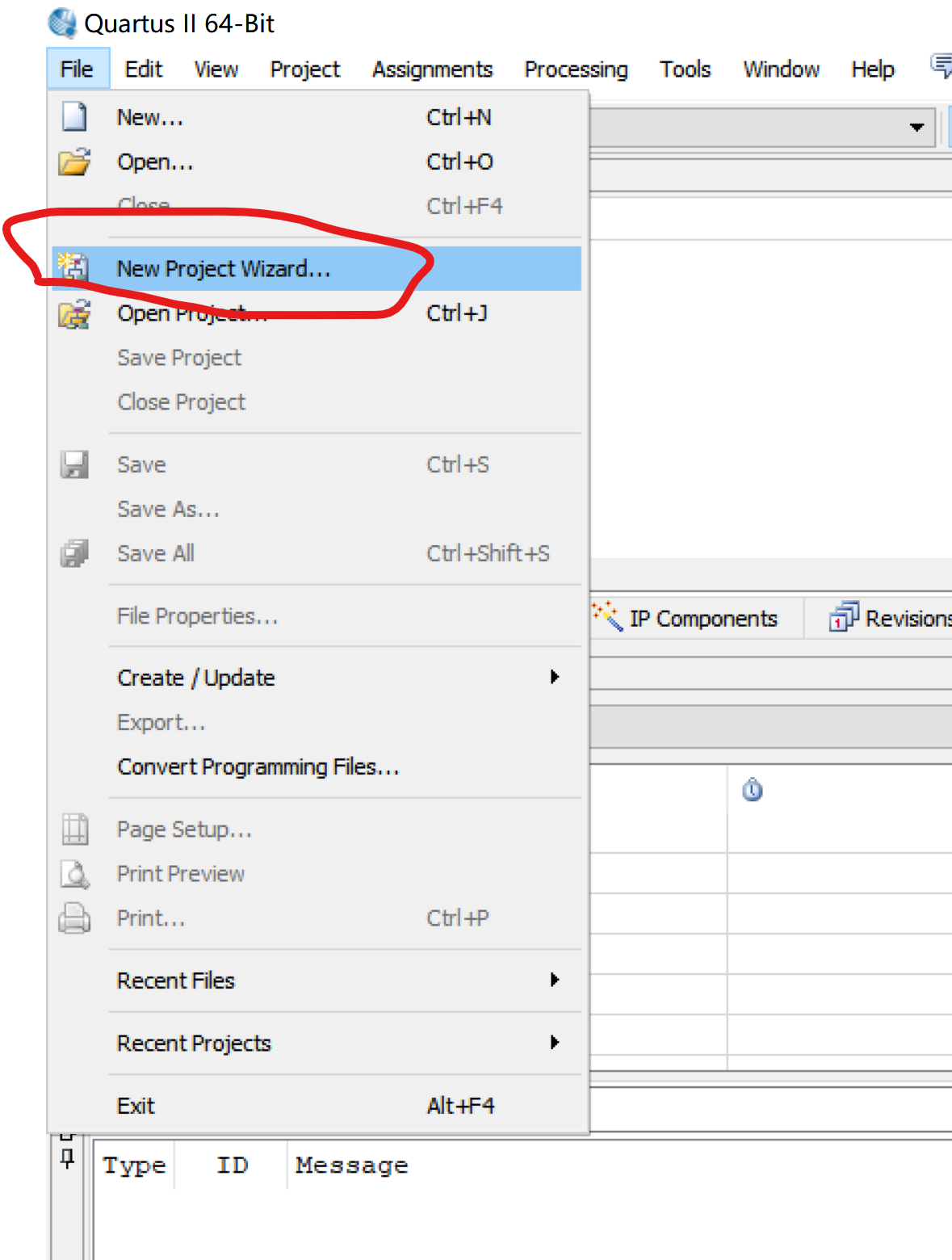
用文本编辑器正确编写源文件，并经modelsim仿真确认该电路设计正确



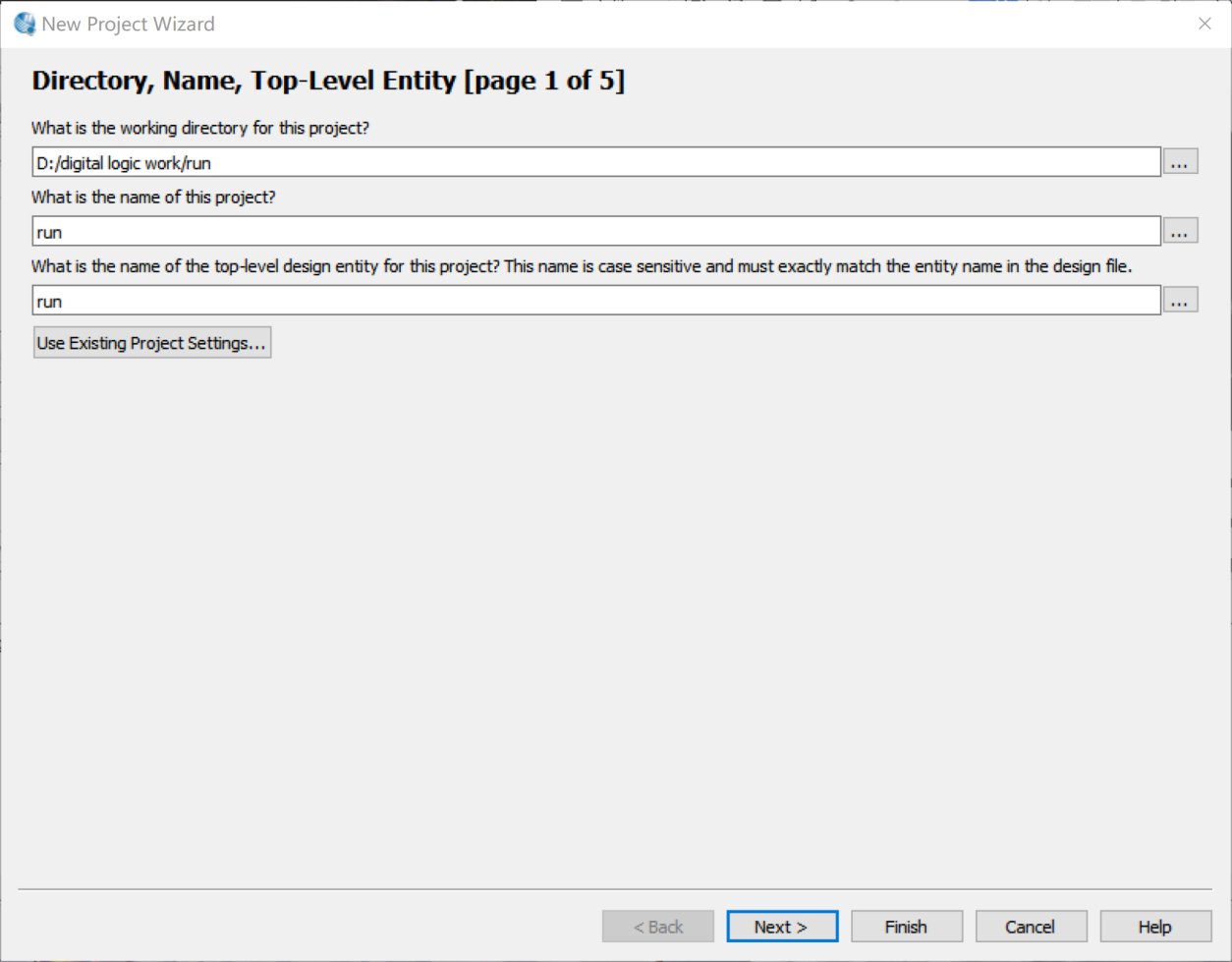


第二步：新建工程

打开Quatusll，点击左上角【File】，选择【New Project Wizard...】

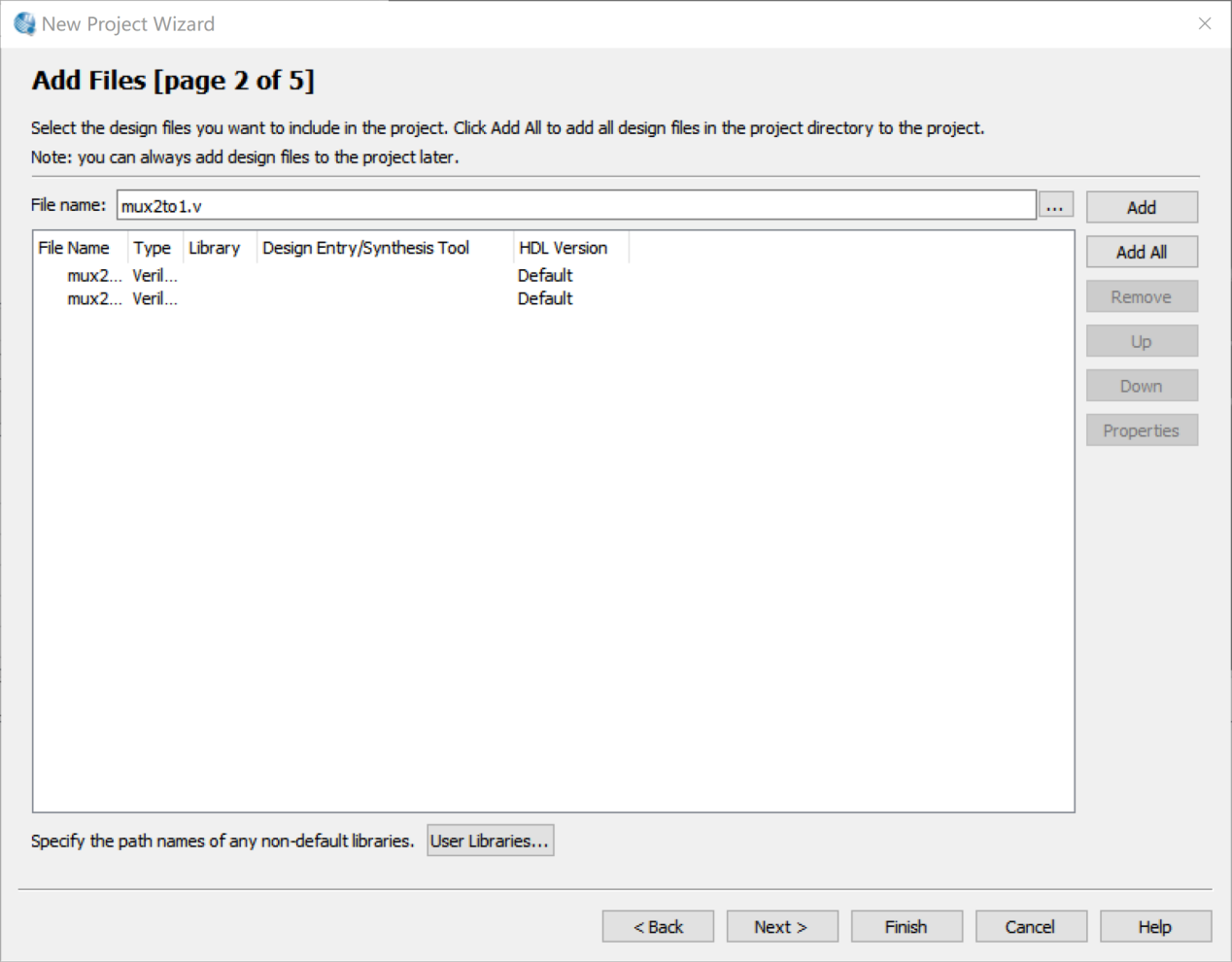


填写项目存储路径和工程名，不要出现中文路径

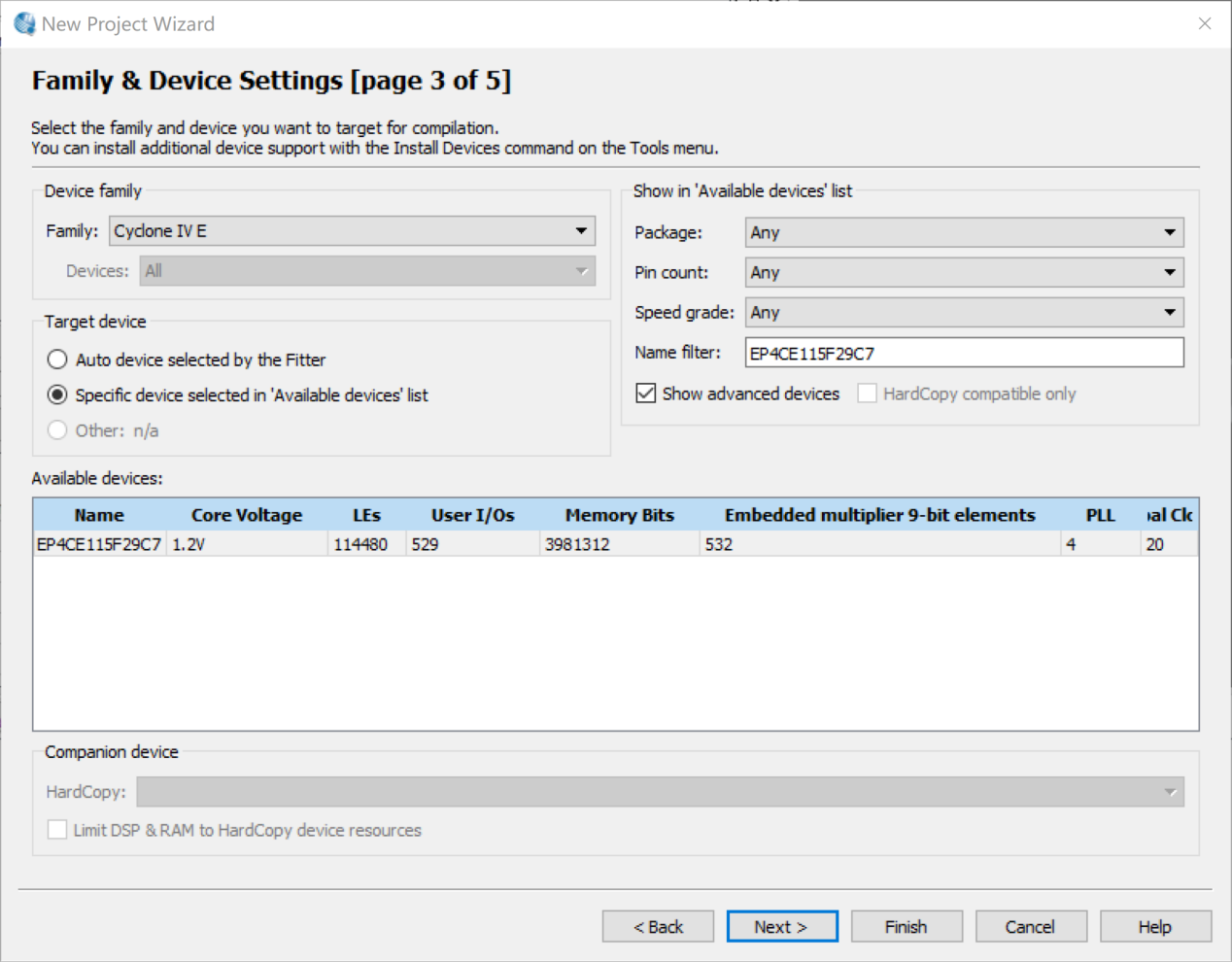


第三步:添加文件

在【File name】下选择已经存在的工程项目，利用【Add】或【Add all】命令添加文件到新工程，点击【Next】

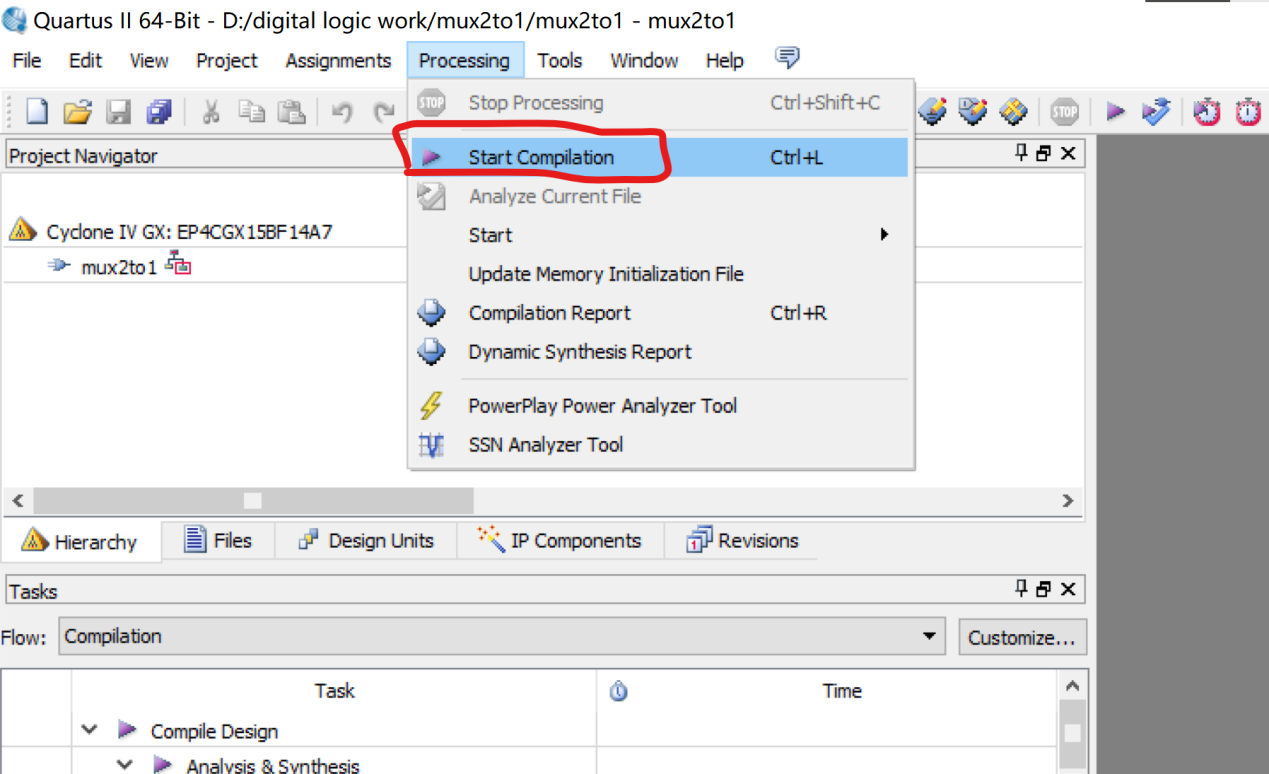


选择设备系列，并在【devices】下选择具体设备型号，这里采用Altera公司的Cyclone IV系列的EP4CE115F29C7型FPGA作为建立工程、代码编译、综合库时使用的库器件，点击【Next】



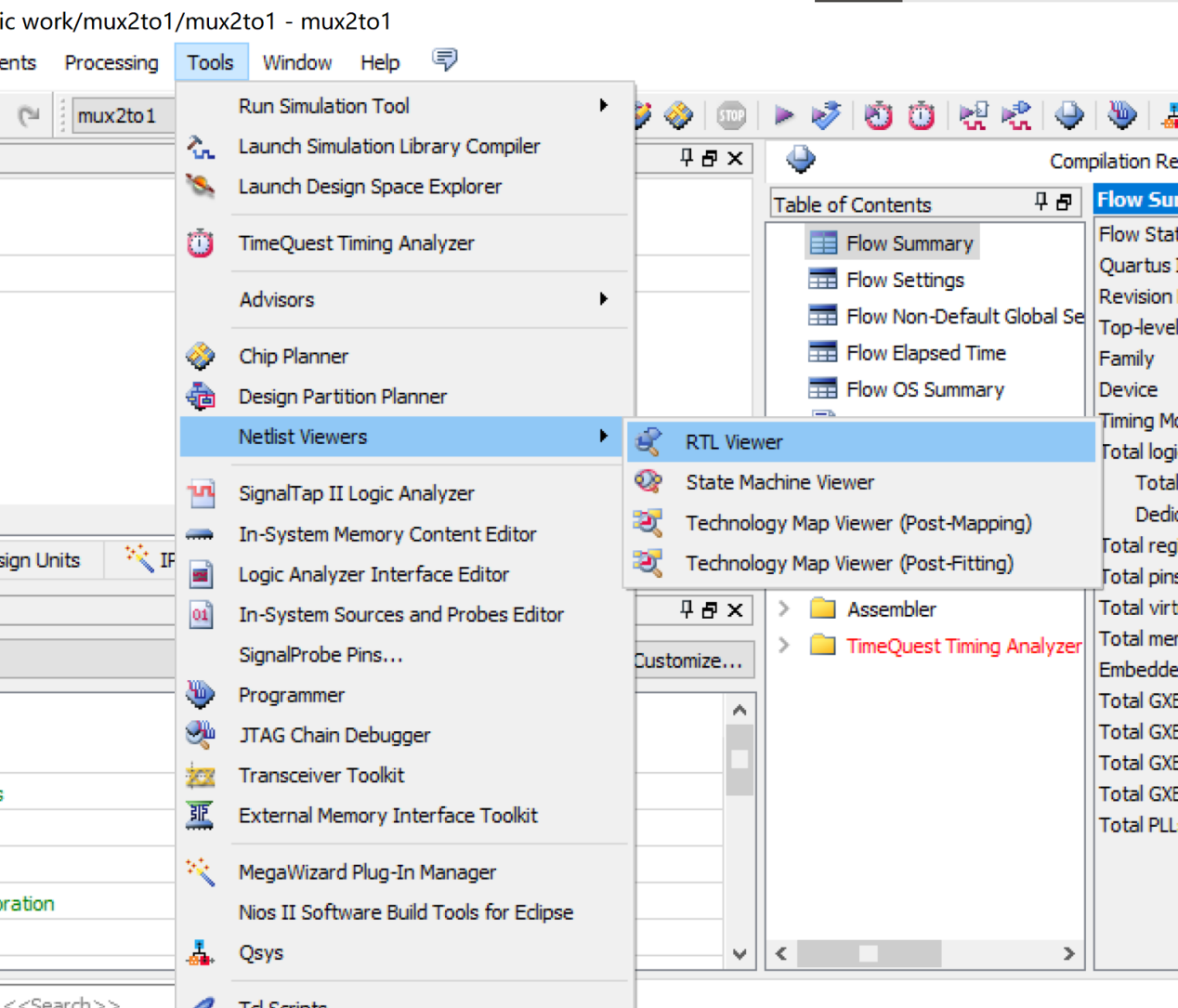
第四步:编译

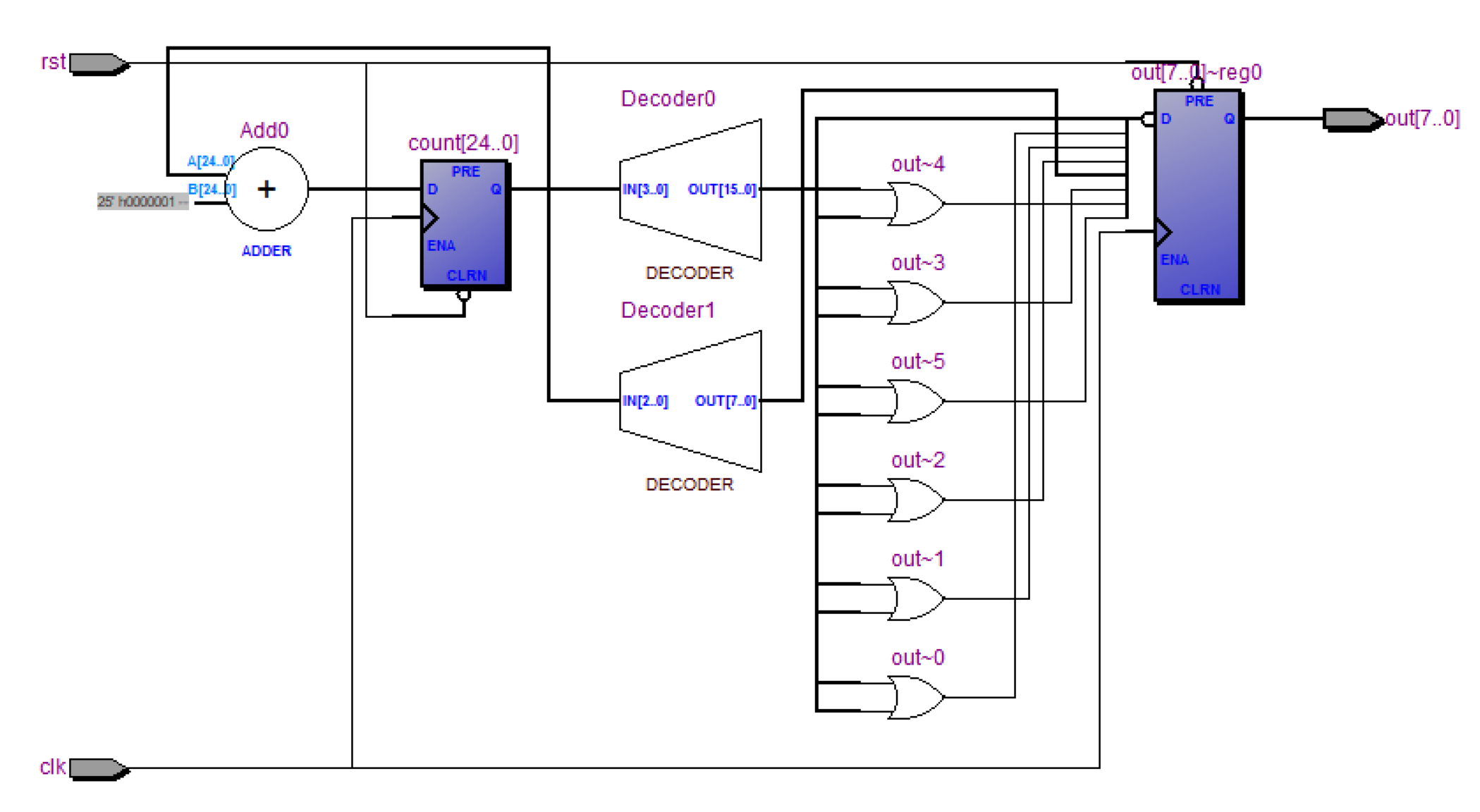
点击【Processing】，选择【Start Compilation】进行编译



第五步:查看电路结构

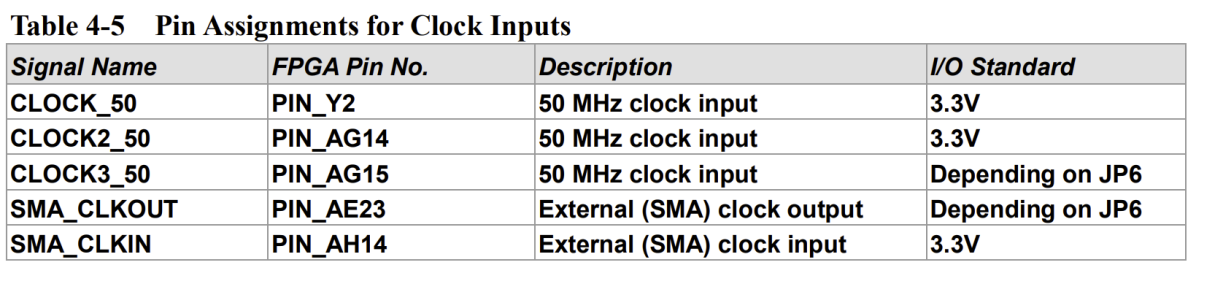
等待编译完成后，点击【Tool】，选择【RTL viewer】工具查看电路图结构是否和预期设计一致

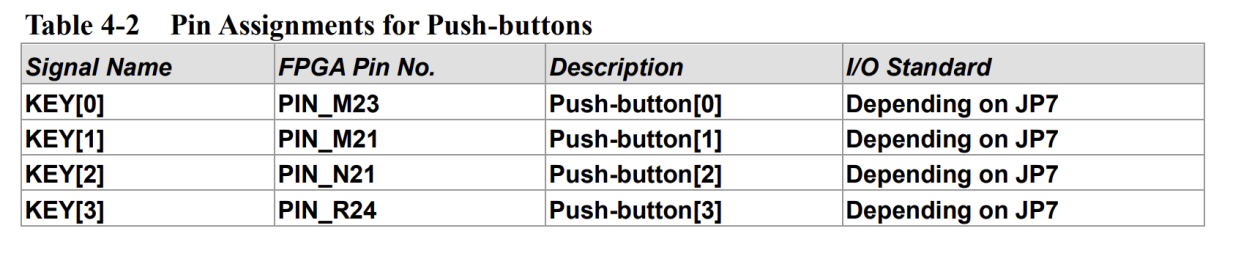
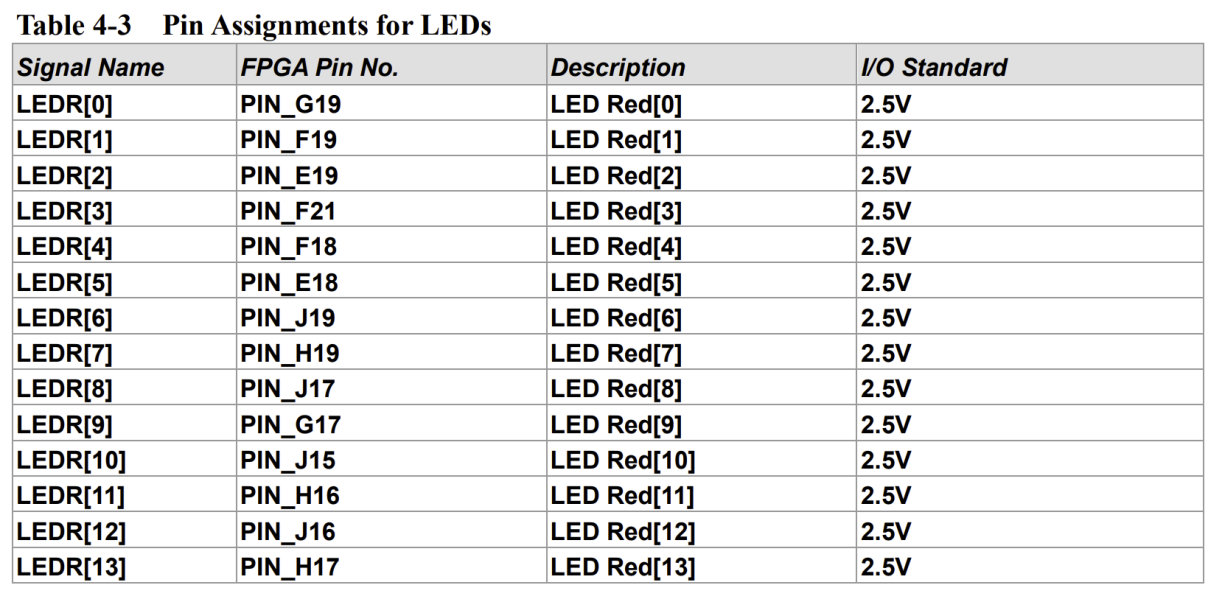


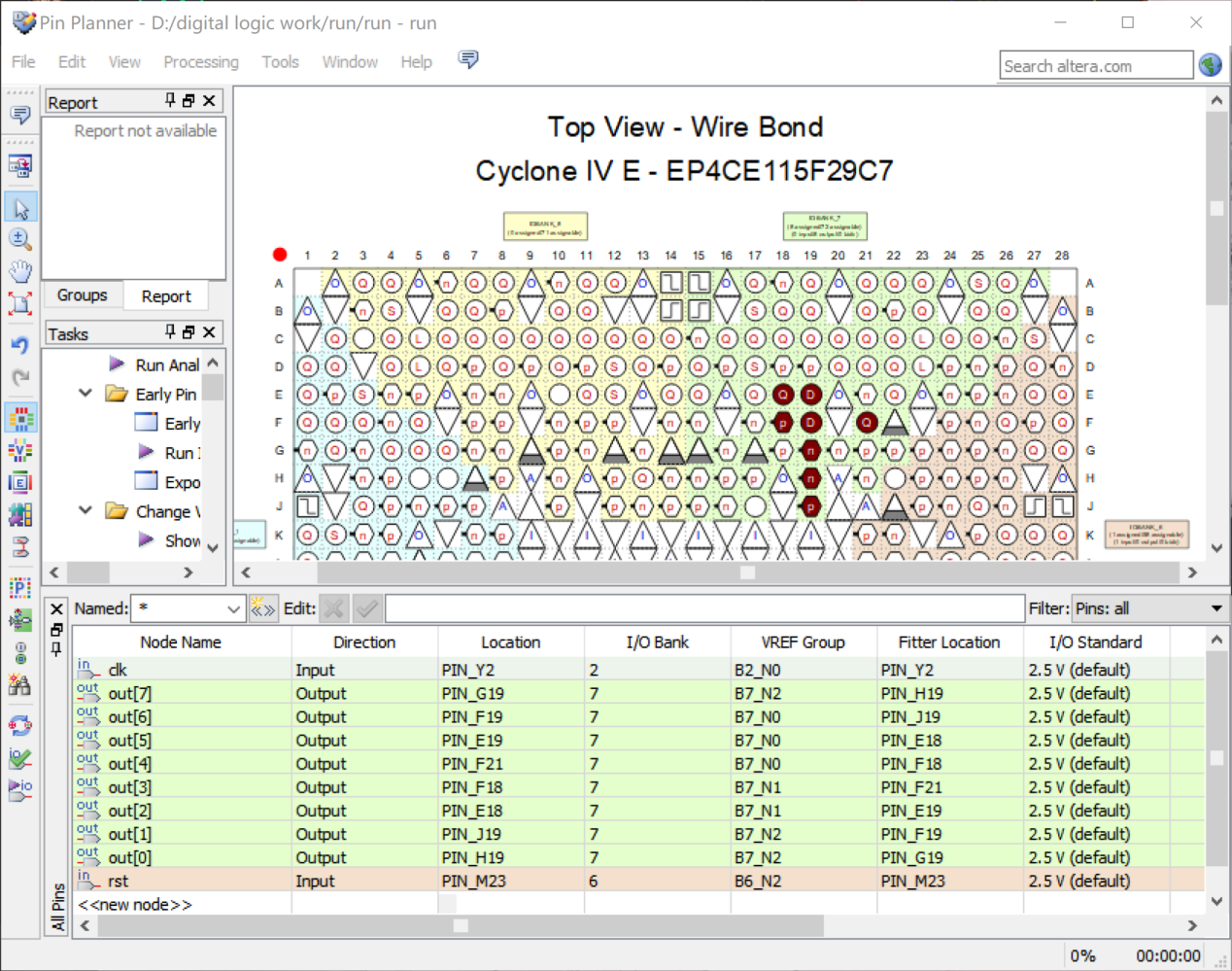


第六步：管脚绑定

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（此时同时参考DE2\_115 User Manul.pdf文件）

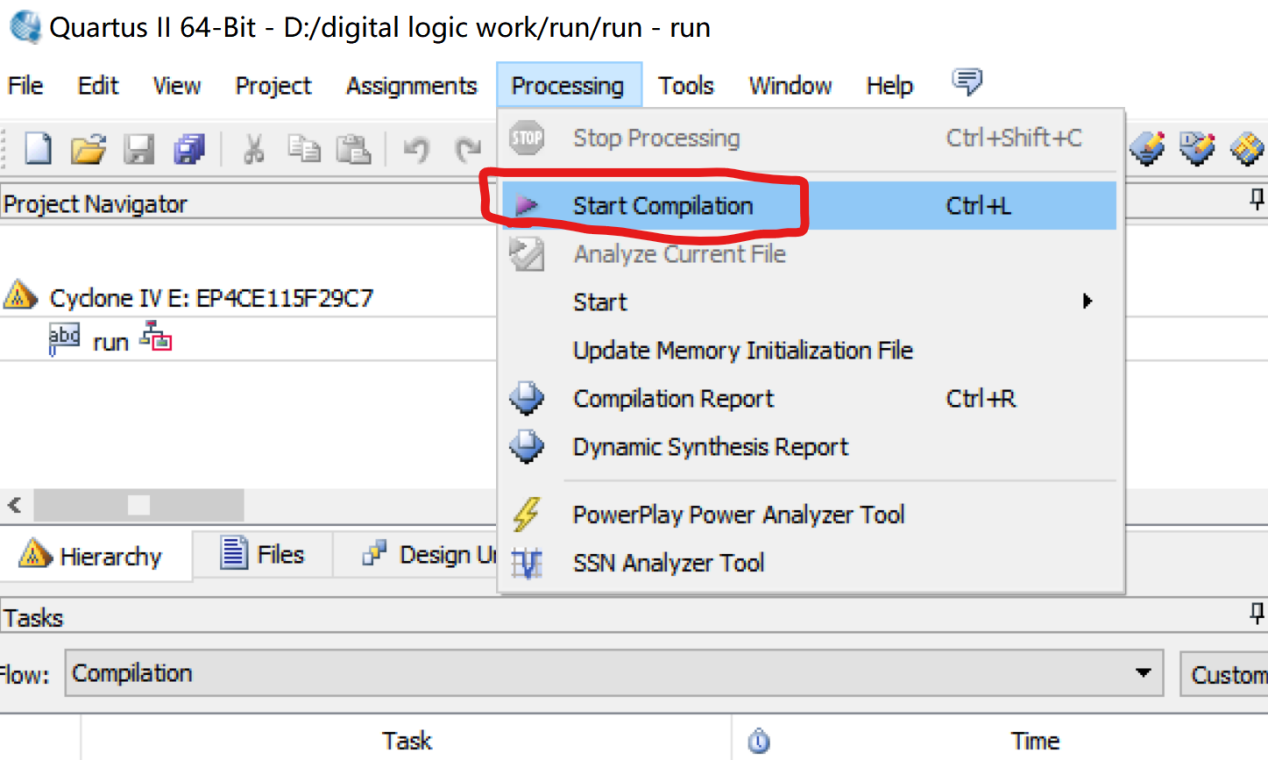




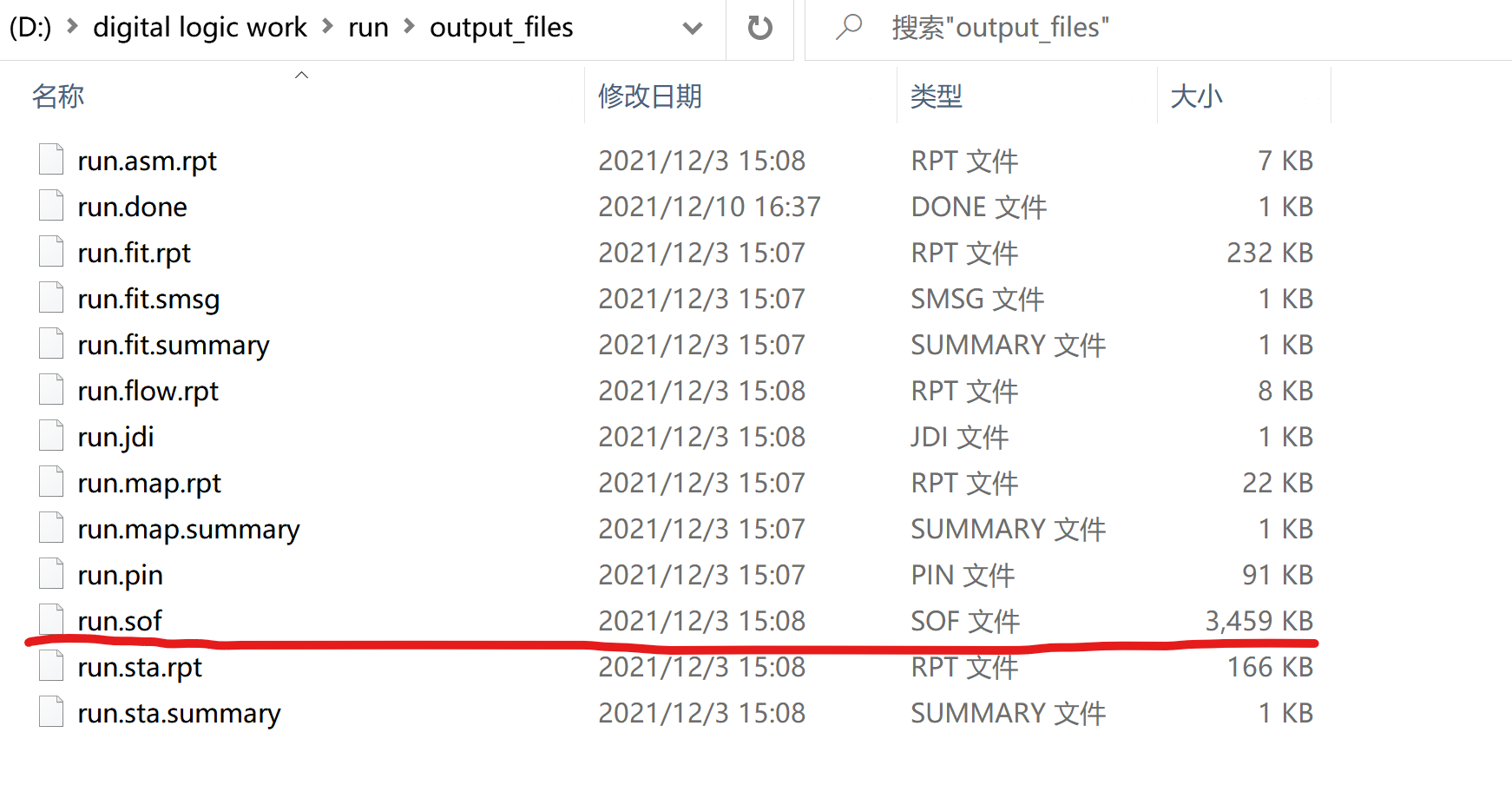


第七步：全编译生成可下载文件（\*.sof）

点击Processing->Start Compilation进行全编译



在run工程文件夹下的output\_files中查看刚刚生成的\*.sof文件



第八步：将开发板USB线连接到计算机，并开启电源，安装驱动程序

第九步：程序下载至FPGA开发板，并观察现象

点击工具栏 Programmer进行程序下载。

1. **实验内容**
2. 跑马灯设计及FPGA实现

verilog代码如下：

module run (clk,rst,out);

input clk,rst;

output [7:0] out;

reg [7:0] out;

reg [24:0] count;

always @ ( posedge clk or negedge rst )

    if(!rst)

    begin

        count<=16'b0;

    end

    else

    begin

        count<=count+1;

    end

always @ ( posedge clk or negedge rst)

    if(!rst)

    begin

        out<=8'hff;

    end

    else

    begin

        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

    end

endmodule

testbench测试文件如下：

`timescale 1ns/1ns

module tb\_run;

reg clk\_test;

reg rst\_test;

wire [7:0] out\_test;

initial

begin

clk\_test = 0;

rst\_test = 0;

#2 rst\_test=1;

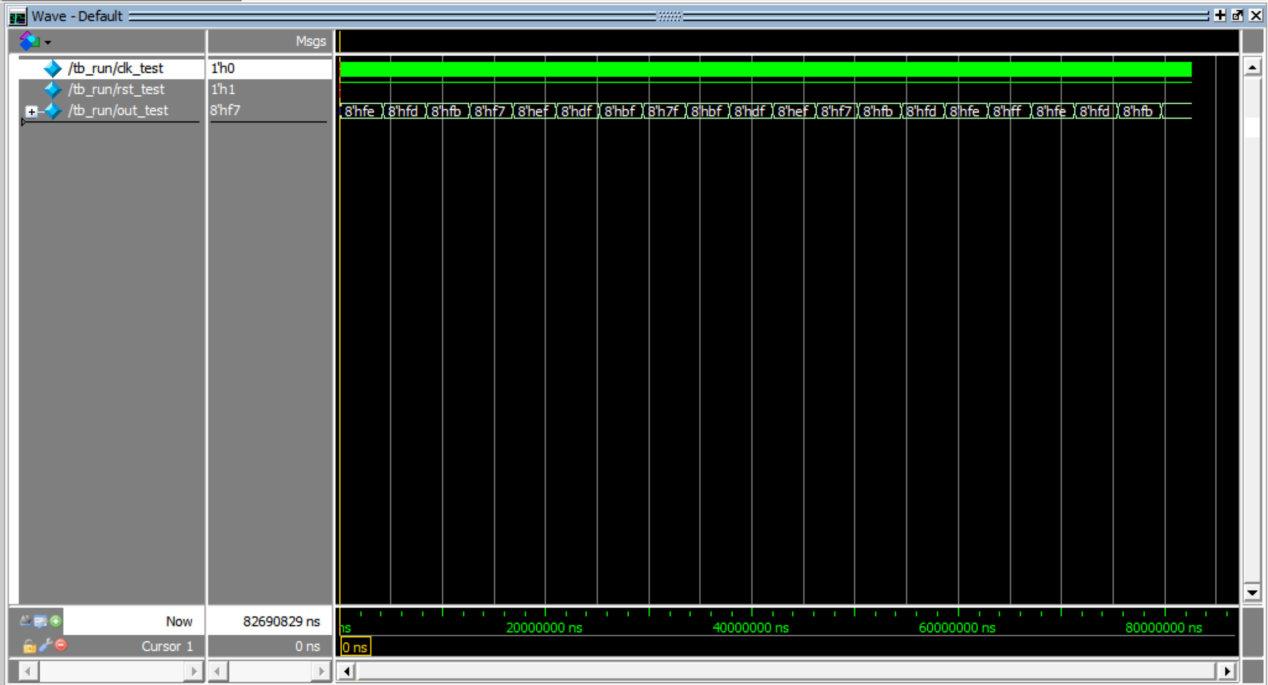
end

always #1 clk\_test = ~clk\_test;

run UUT\_run(.rst(rst\_test),.clk(clk\_test),.out(out\_test));

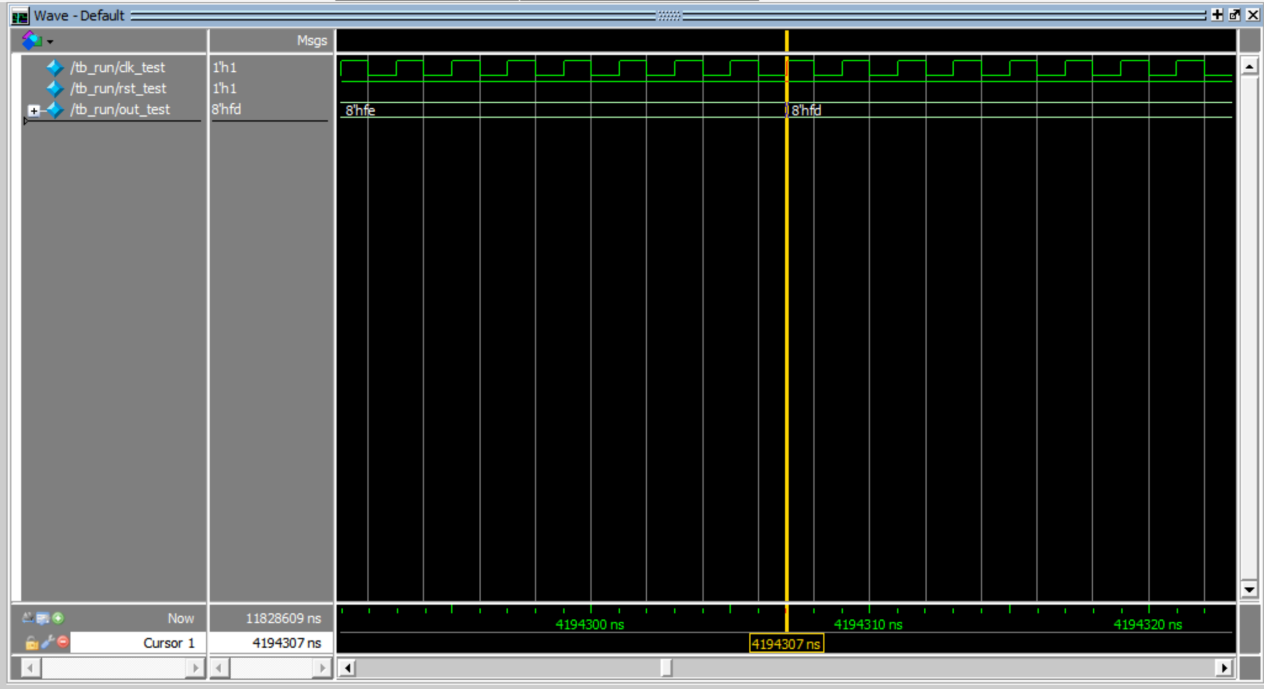
endmodule

仿真波形如下：

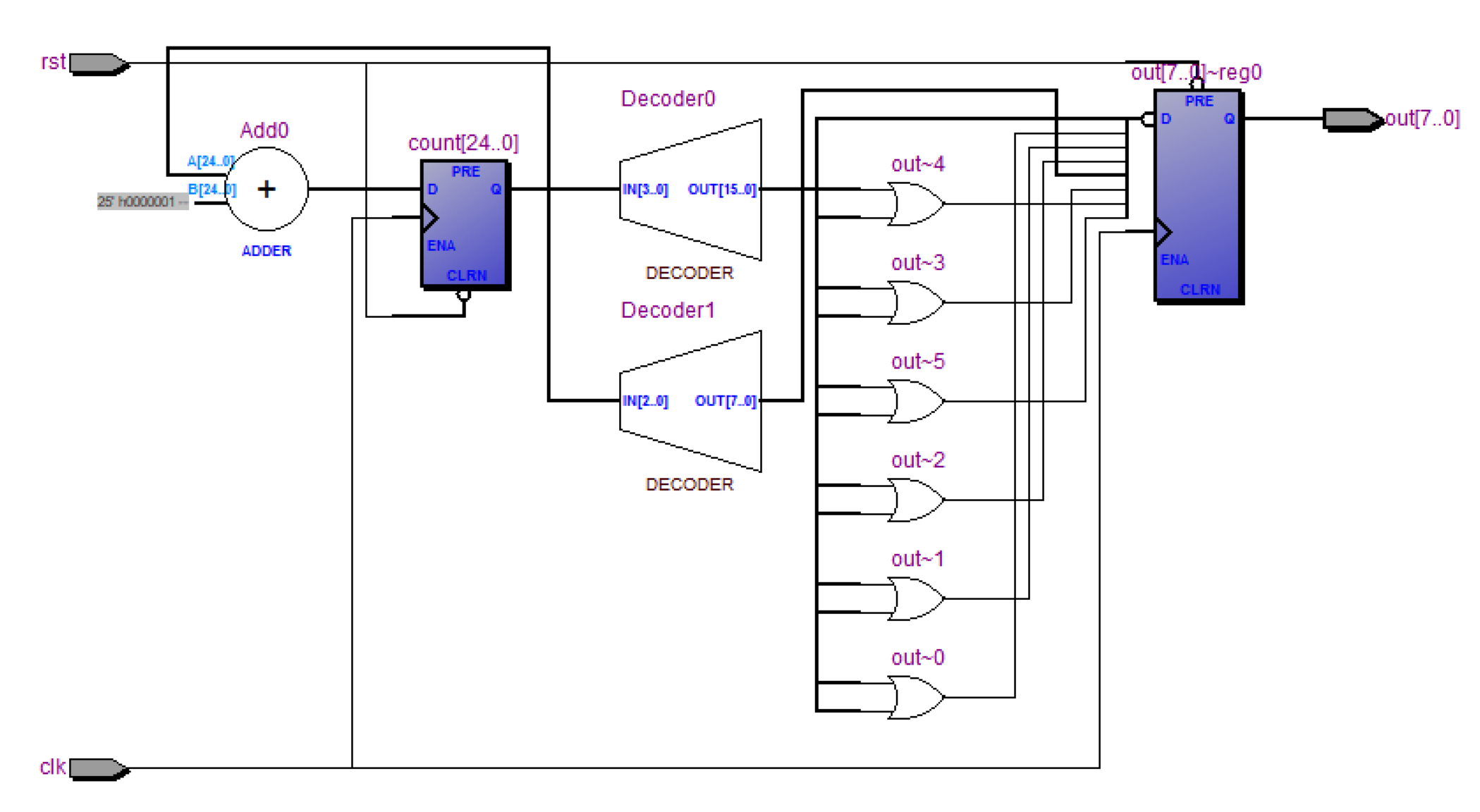


部分细节放大后如图：

![](E0I~E0N]_K`O%S49UN@]M](data:image/png;base64,)



综合电路图如下：



波形分析：

由波形图可知，时钟信号clk\_test每1ns变化一次，时钟周期为2ns

0~2ns内，rst\_test=0，执行复位功能，count\_test=0，out\_test=8'b1111\_1111=8’hff；

之后rst\_test=1，执行计数功能，count\_test++；

2ns~3ns内，因为时钟未处于上升沿且rst\_test未处于下降沿，故out\_test保持原本状态不变

3ns时刻，时钟处于上升沿状态，count\_test=0+1=1，count\_test[24:21]=0，因此out\_test=8'b1111\_1110=8’hfe；

此后每经过2ns，时钟处于上升沿时，count\_test++

直到3+2\*2^21=4194307ns时，count\_test=2^21，count\_test[24:21]=1，因此out\_test=8'b1111\_1101=8’hfd；

类似地，每隔2\*2^21ns，count\_test[24:21]++，out\_test随之变化，具体如下：

cout\_test=2时，out\_test=8'b1111\_1011=8’hfb；

cout\_test=3时，out\_test=8'b1111\_0111=8’hf7；

cout\_test=4时，out\_test=8'b1110\_1111=8’hef；

cout\_test=5时，out\_test=8'b1101\_1111=8’hdf；

cout\_test=6时，out\_test=8'b1011\_1111=8’hbf；

cout\_test=7时，out\_test=8'b0111\_1111=8’h7f；

cout\_test=8时，out\_test=8'b1011\_1111=8’hbf；

cout\_test=9时，out\_test=8'b1101\_1111=8’hdf；

cout\_test=10时，out\_test=8'b1110\_1111=8’hef；

cout\_test=11时，out\_test=8'b1111\_0111=8’hf7；

cout\_test=12时，out\_test=8'b1111\_1011=8’hfb；

cout\_test=13时，out\_test=8'b1111\_1101=8’hfd；

cout\_test=14时，out\_test=8'b1111\_1110=8’hfe；

cout\_test=15时，out\_test=8'b1111\_1111=8’hff；

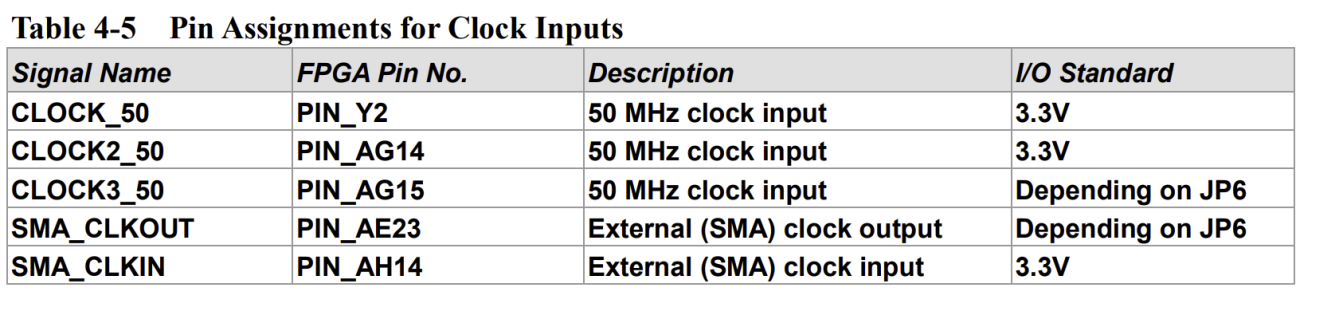
然后重复此过程

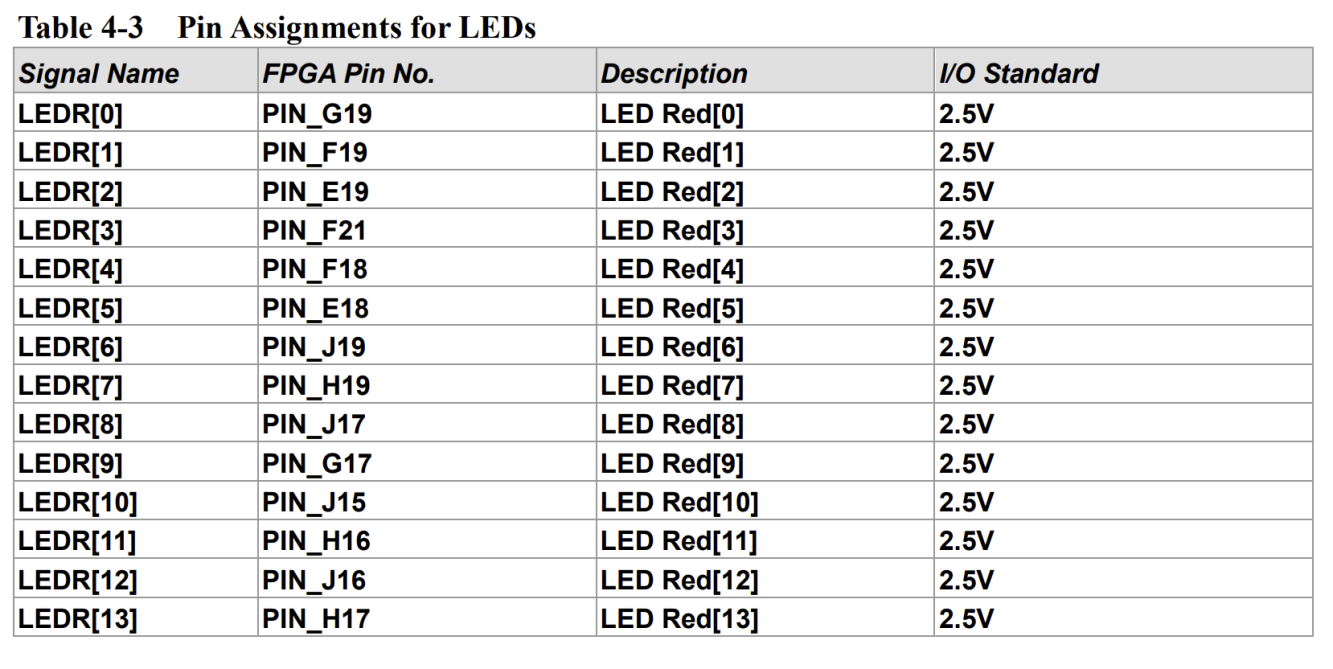
将out\_test的八个输出连接到LED灯上，即可形成跑马灯的效果

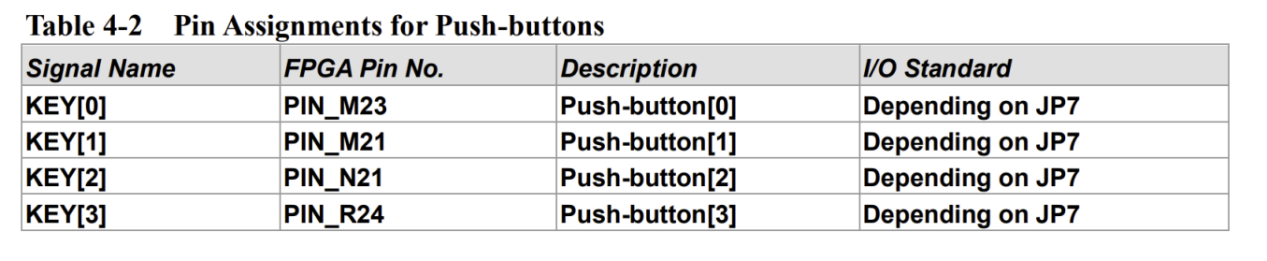
综上，该电路成功实现了跑马灯的设计

管脚绑定流程：

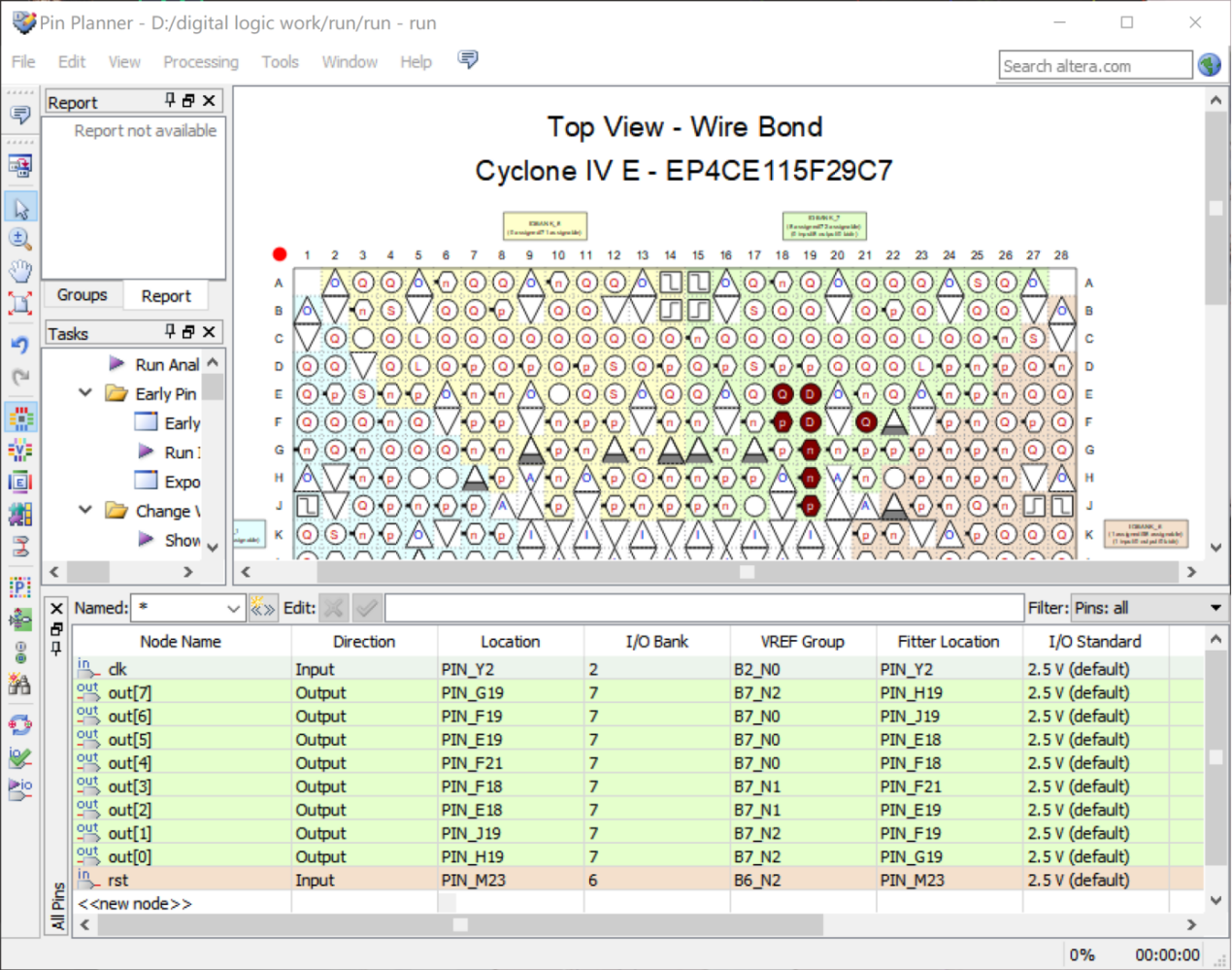
分析Verilog代码可知，此电路需要一个时钟信号管脚，一个按钮类管脚实现切换复位/计数功能，以及8个LEDs管脚用于输出，查DE2\_115 User Manul.pdf文件，找到对应的列表：







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后可得



1. 有限状态机设计

verilog代码如下：

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;

    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

    always@(negedge Resetn, posedge Clock)

        if(Resetn==0) y<=A;

        else y<=Y;

    assign z=(y==C)|(y==E);

endmodule

testbench测试文件如下：

`timescale 1ns/1ps

module tb\_sequence;

reg Clock\_test;

reg Resetn\_test;

reg w\_test;

wire z\_test;

initial

begin

Clock\_test=0;

w\_test=0;

Resetn\_test=0;

#30 Resetn\_test=1;

#40 w\_test=1;

#40 w\_test=1;

#40 w\_test=1;

#40 w\_test=0;

#40 w\_test=0;

#40 w\_test=1;

#40 w\_test=0;

#40 w\_test=1;

end

always #20 Clock\_test=~Clock\_test;

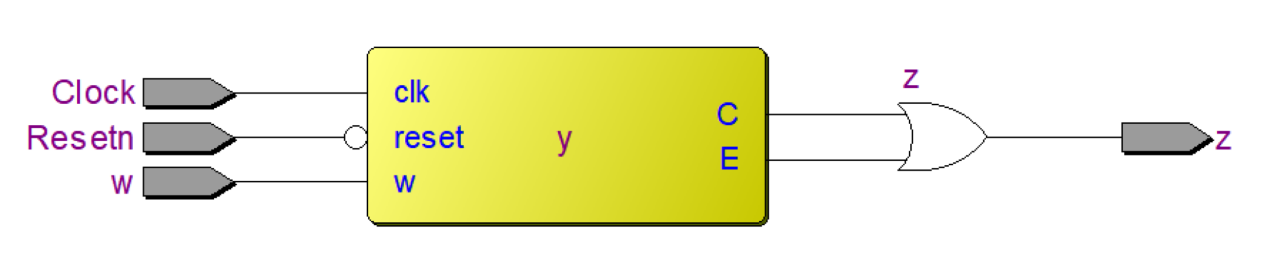
sequence UUT\_sequence(.Resetn(Resetn\_test),.Clock(Clock\_test),.w(w\_test),.z(z\_test));

endmodule

仿真波形如下：



综合电路图如下：



波形分析：

0~30ns内，Resetn\_test=0，执行复位功能，y=3’b000=3’h0，处于状态A，此时w\_test=0，次态Y=3’b001，z\_test=0；

此后Resetn\_test=1，可以进行状态跳跃

60ns时刻，时钟处于上升沿状态，y=Y=3’b001=3’h1，处于状态B，此时w\_test=0，次态Y=3’b010，z\_test=0；

70ns时刻，w\_test变为1，次态变为Y=3’b011

100ns时刻，时钟处于上升沿状态，y=Y=3’b011=3’h3，处于状态D，此时w\_test=1，次态Y=3’b100，z\_test=0；

140ns时刻，时钟处于上升沿状态，y=Y=3’b100=3’h4，处于状态E，此时w\_test=1，次态Y=3’b100，z\_test=1；

180ns时刻，时钟处于上升沿状态，y=Y=3’b100=3’h4，处于状态E，此时w\_test=1，次态Y=3’b100，z\_test=1；

190ns时刻，w\_test变为0，次态变为Y=3’b001

220ns时刻，时钟处于上升沿状态，y=Y=3’b001=3’h1，处于状态B，此时w\_test=0，次态Y=3’b010，z\_test=0；

260ns时刻，时钟处于上升沿状态，y=Y=3’b010=3’h2，处于状态C，此时w\_test=0，次态Y=3’b010，z\_test=1；

270ns时刻，w\_test变为1，次态变为Y=3’b011

300ns时刻，时钟处于上升沿状态，y=Y=3’b011=3’h3，处于状态D，此时w\_test=1，次态Y=3’b100，z\_test=0；

310ns时刻，w\_test变为0，次态变为Y=3’b001

340ns时刻，时钟处于上升沿状态，y=Y=3’b001=3’h1，处于状态B，此时w\_test=0，次态Y=3’b010，z\_test=0；

350ns时刻，w\_test变为1，次态变为Y=3’b011

380ns时刻，时钟处于上升沿状态，y=Y=3’b011=3’h3，处于状态D，此时w\_test=1，次态Y=3’b100，z\_test=0；

420ns时刻，时钟处于上升沿状态，y=Y=3’b100=3’h4，处于状态E，此时w\_test=1，次态Y=3’b100，z\_test=1；

综上，该电路成功实现了有限状态机的设计

1. **本次实验收获和心得**

本次实验学习了如何在FPGA进行设计实现，通过跑马灯的例子进行了管脚绑定的操作，成功进行了状态机的设计及测试验证，了解了在FPGA进行设计测试电路的整个流程。通过实验我明白了参考书的重要性，每一个管脚的绑定都需要仔细查阅参考手册进行匹配，应该先弄清需要实现的功能然后再从目录中寻找相应的管脚。