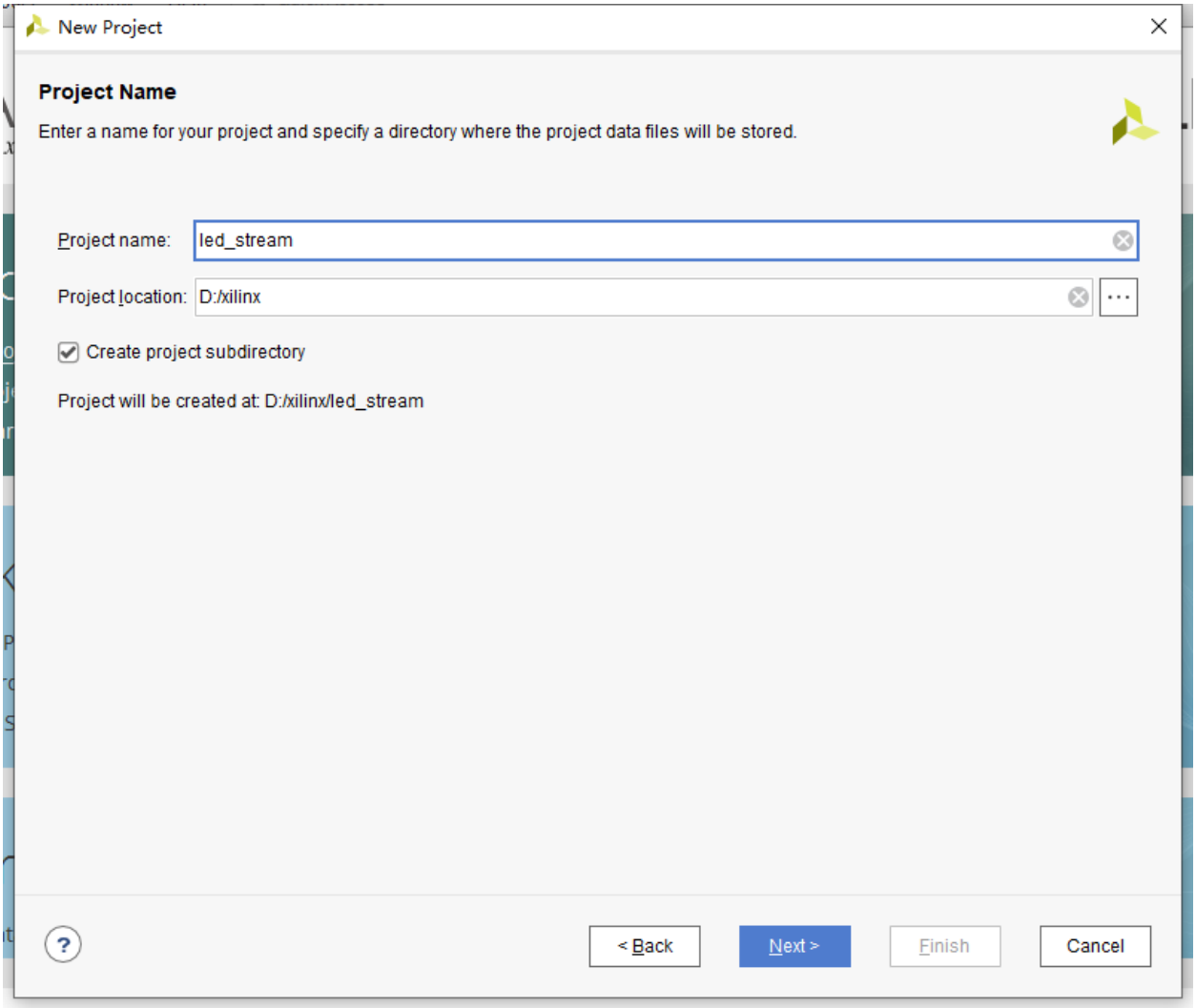
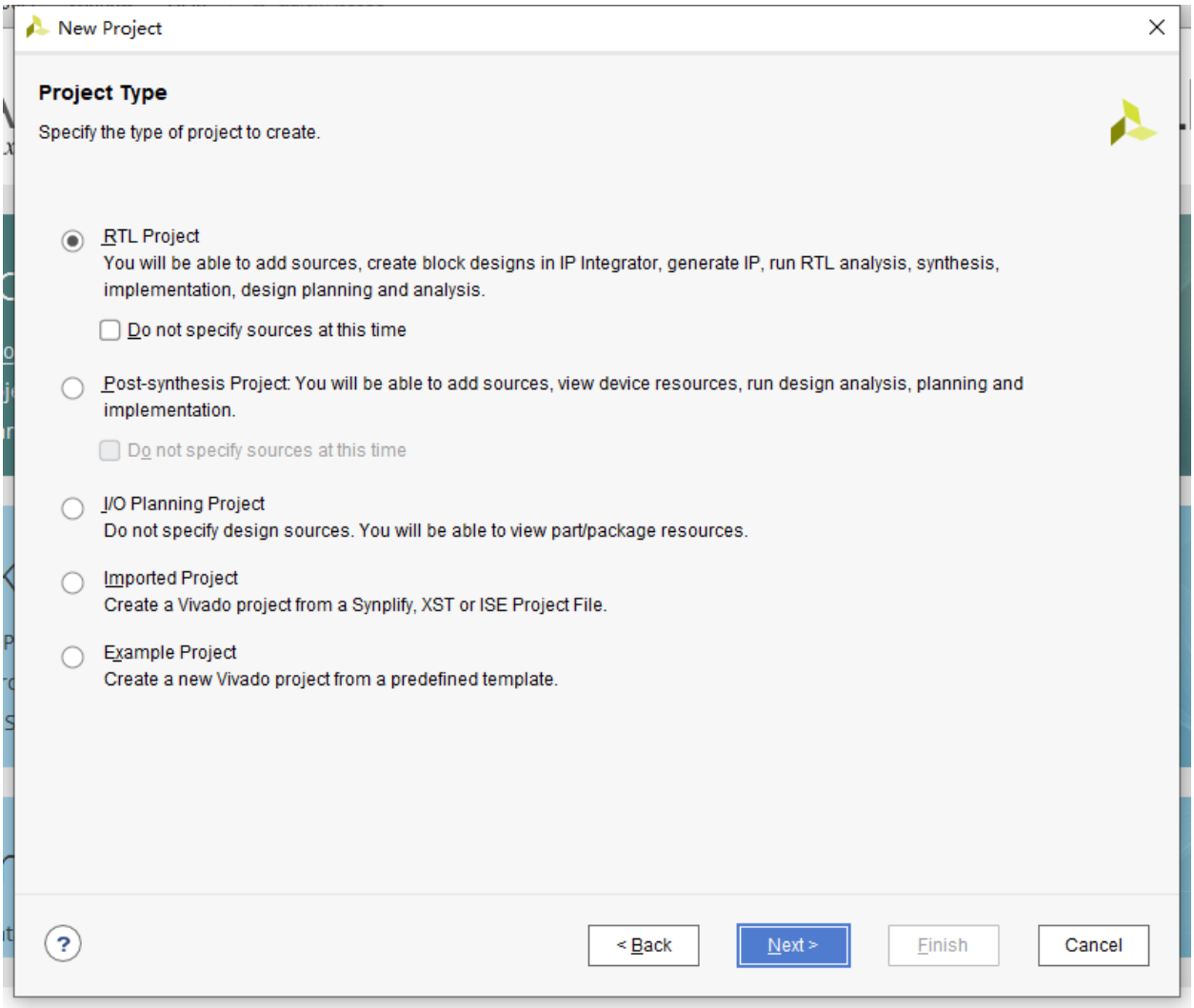
[L01]PL LED 流水灯 + ILA调试

任何软件开发第一步都是Hello World,而FPGA的Hello World就是流水灯,在C语言中我们要做一个for循环,要做一个位移,然后让灯流起来,事情就完成了.

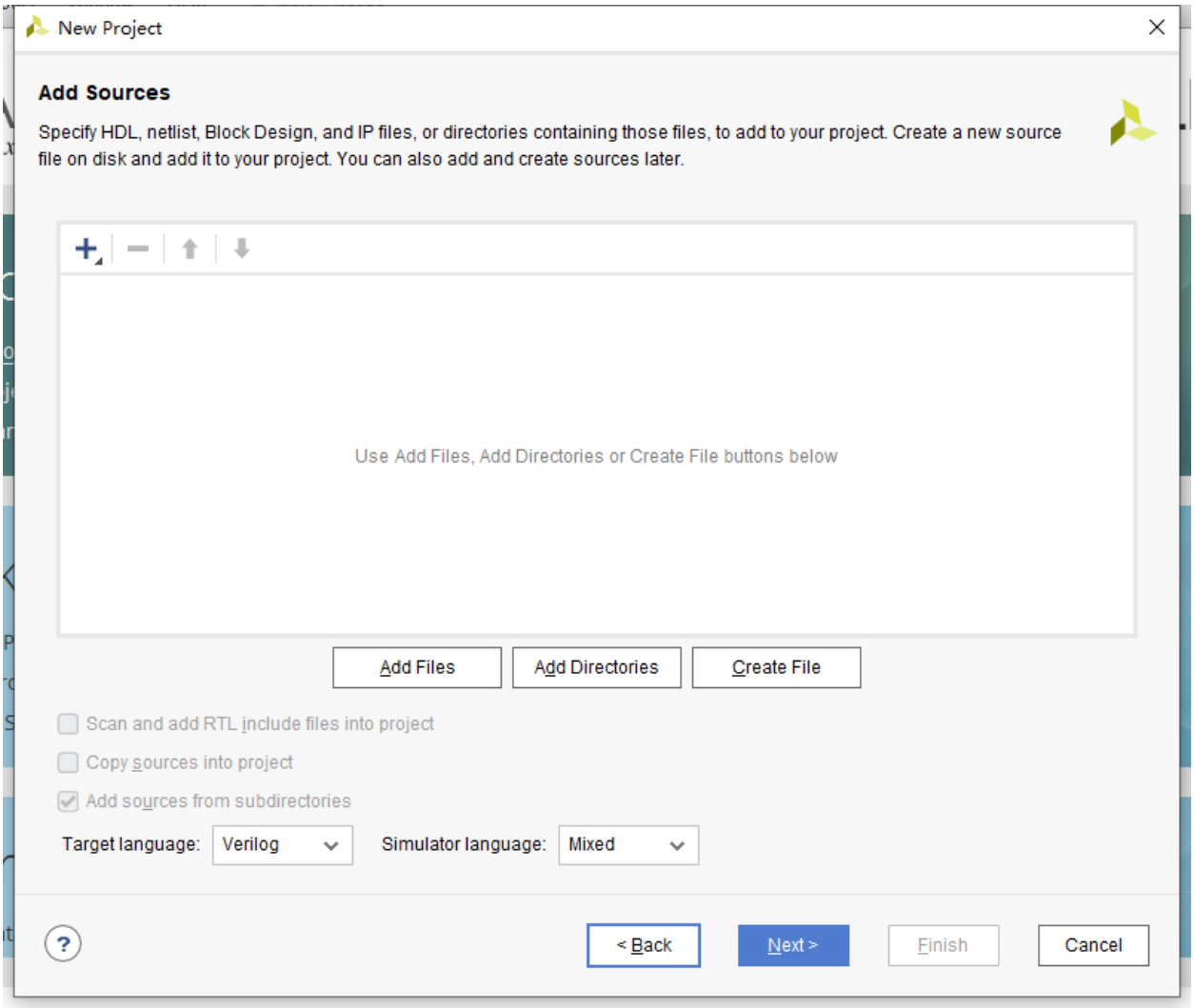
开始当然是创建工程.



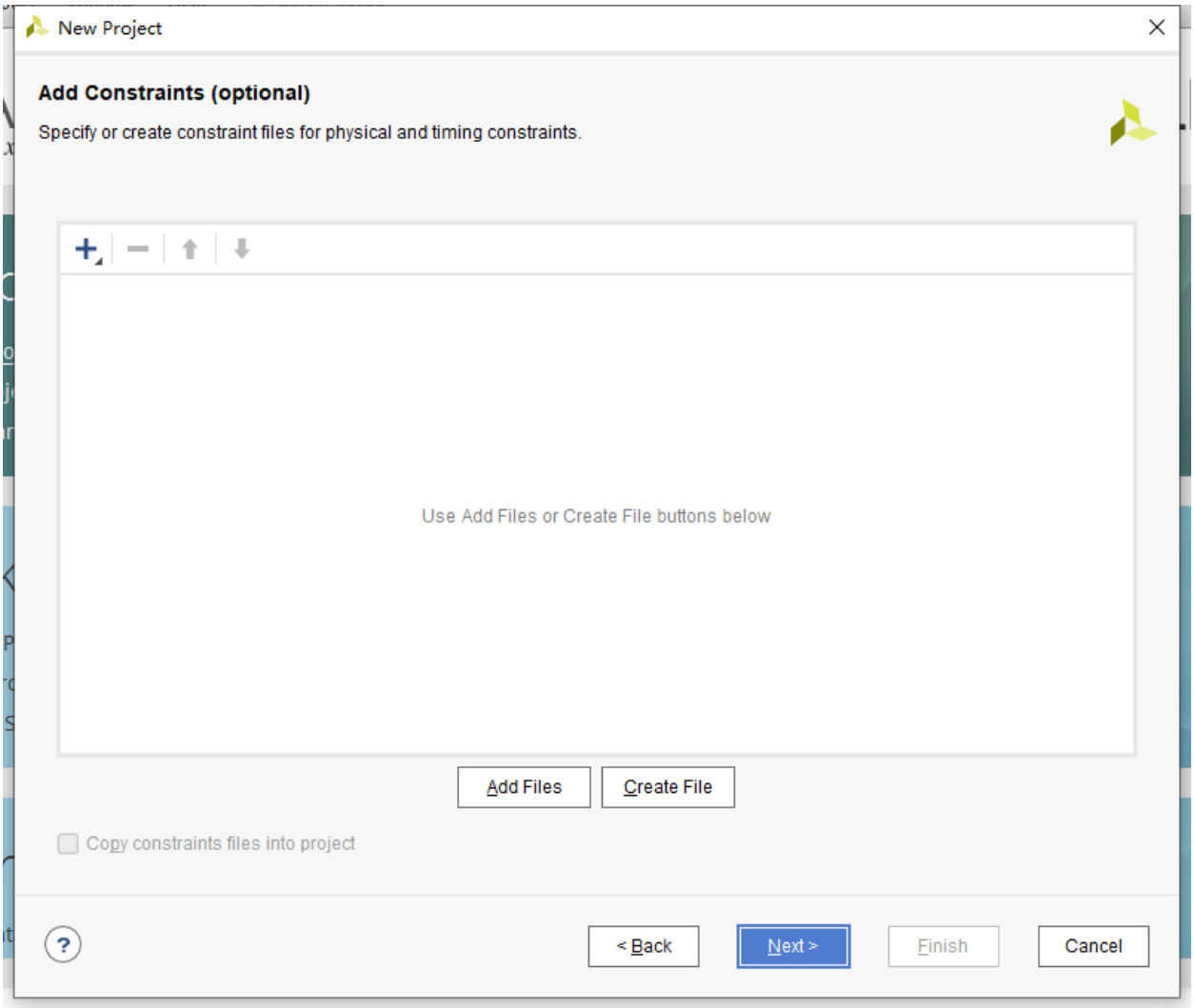
然后当然是RTL工程,什么都有.



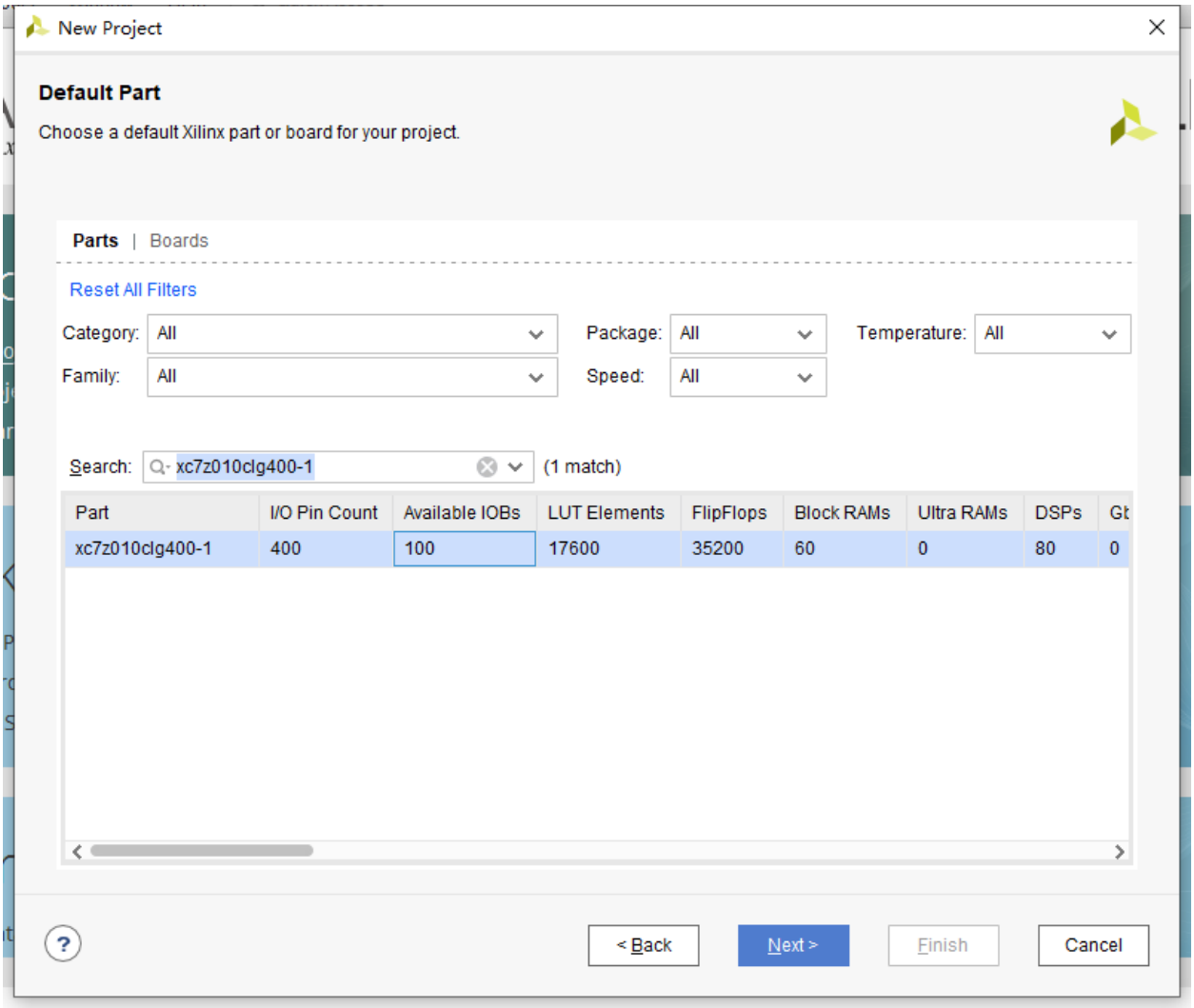
源码目前还没有,所以直接Next就行.



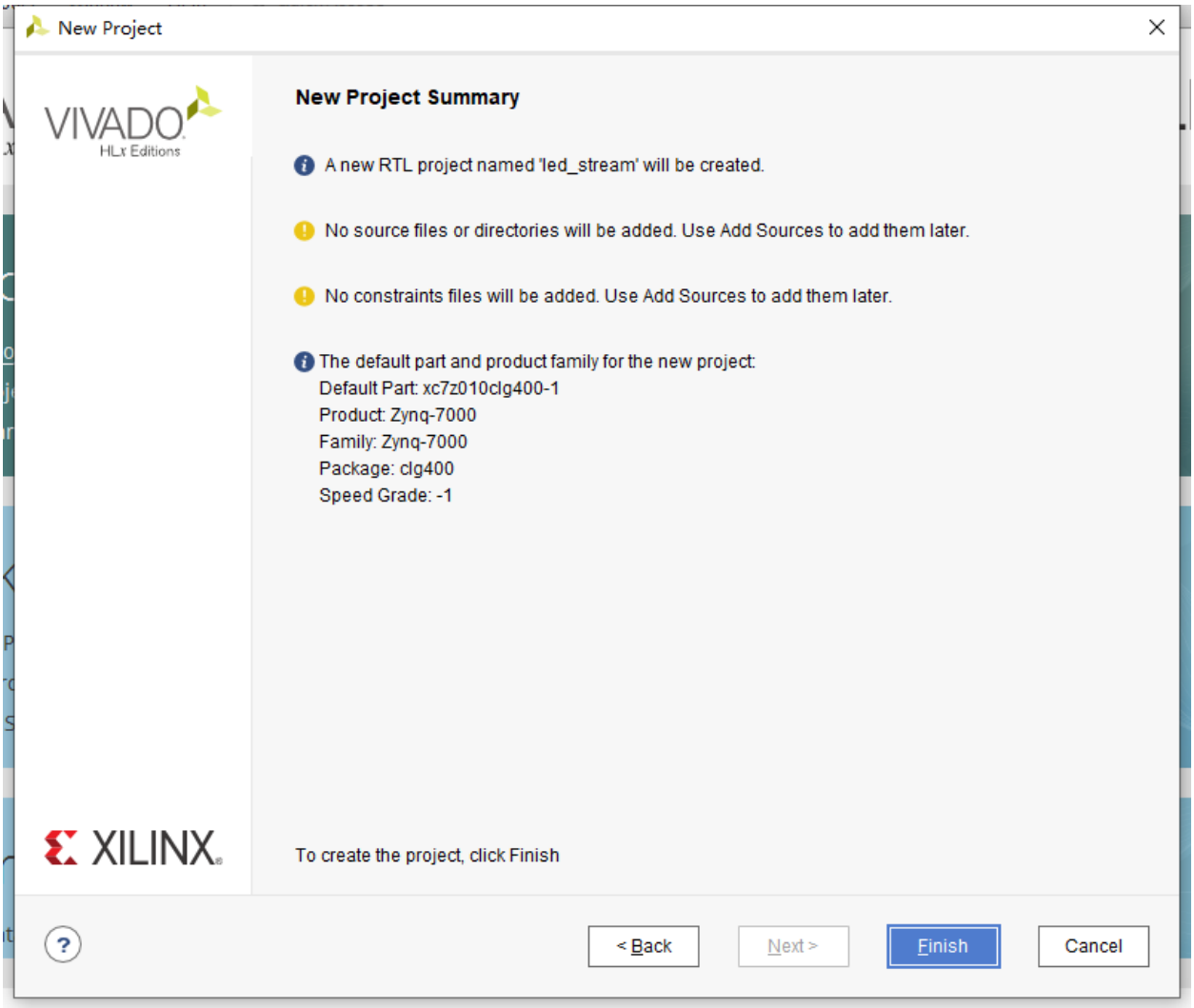
约束一样没有,所以直接Next就行.



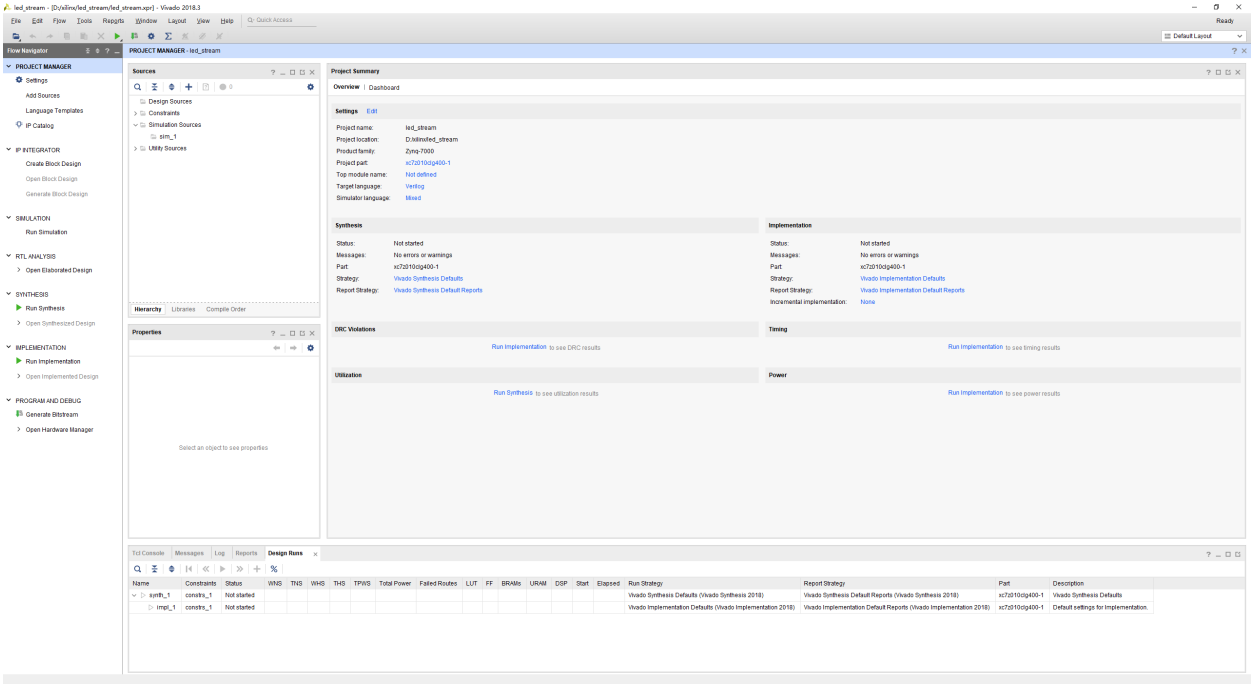
直接输入芯片型号**xc7z010clg400-1**,然后选择这个芯片.



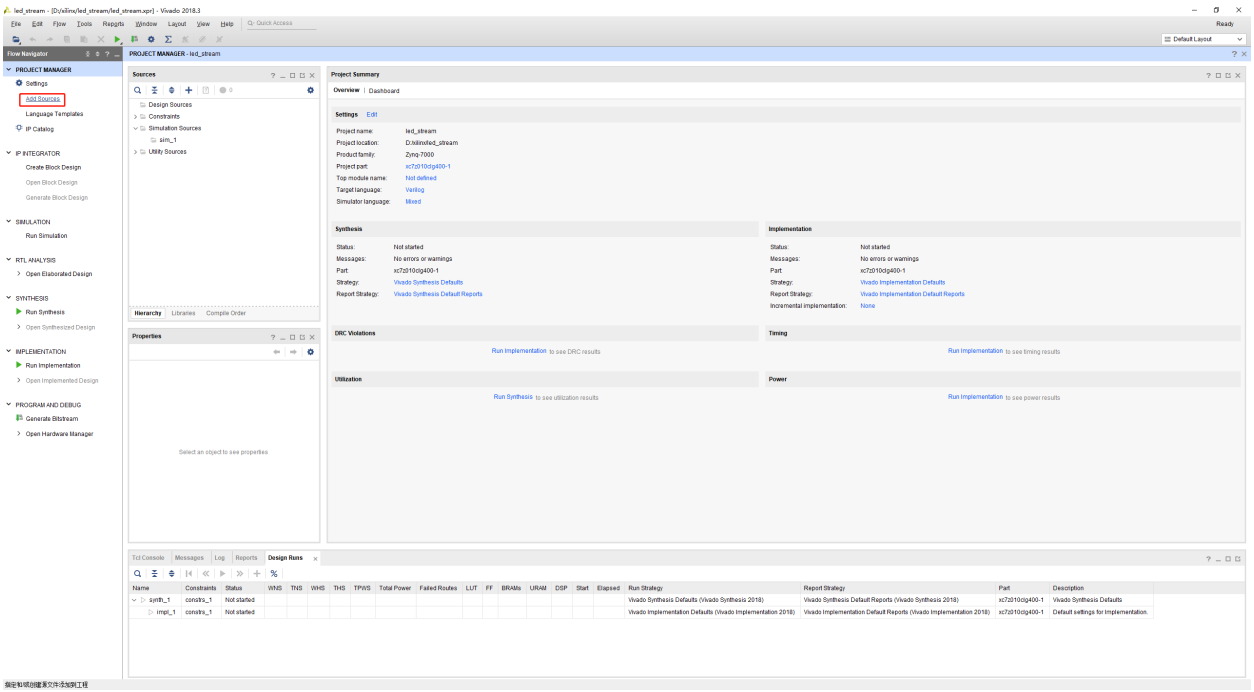
一切完成就可以,上面这些基础步骤只说一次了哈.



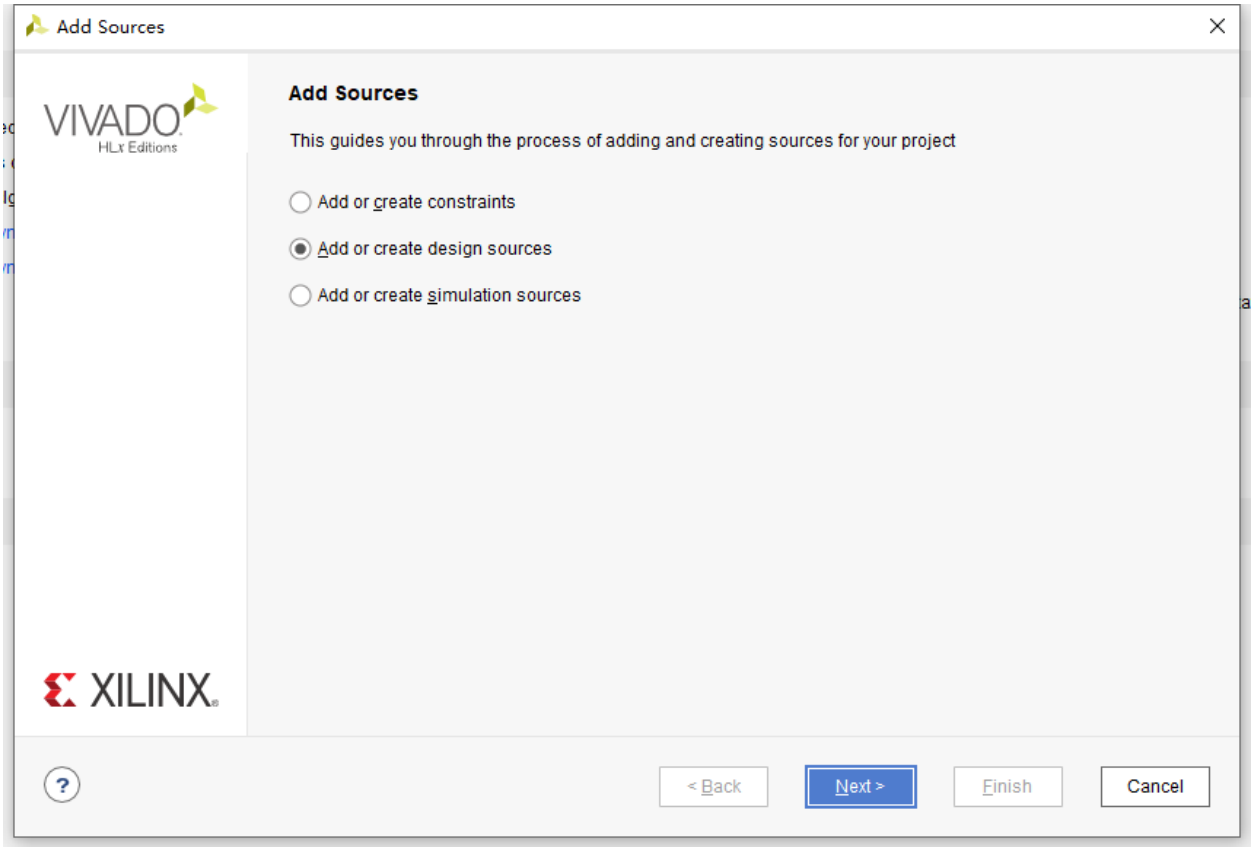
创建后的界面如图,各种工作区以后会自然熟悉.



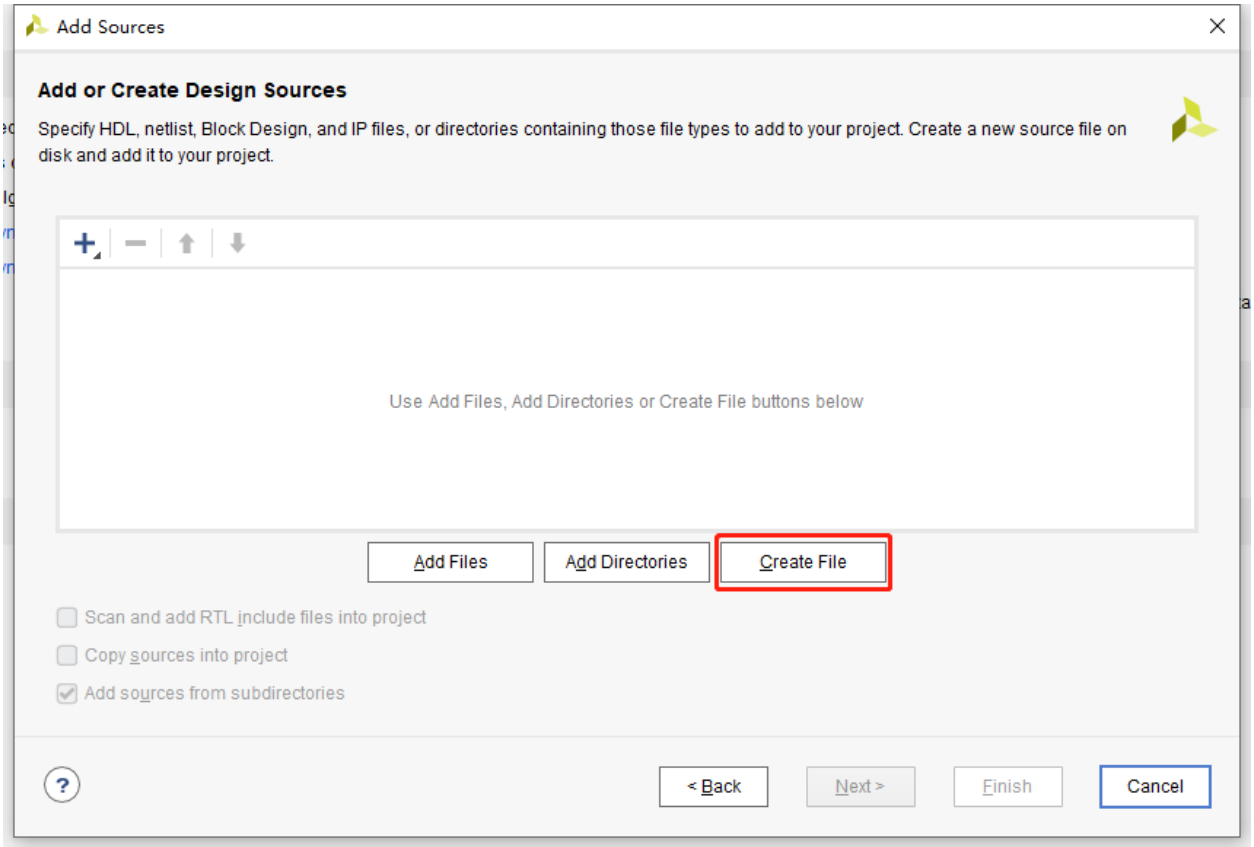
点击Add Source添加/新建代码.



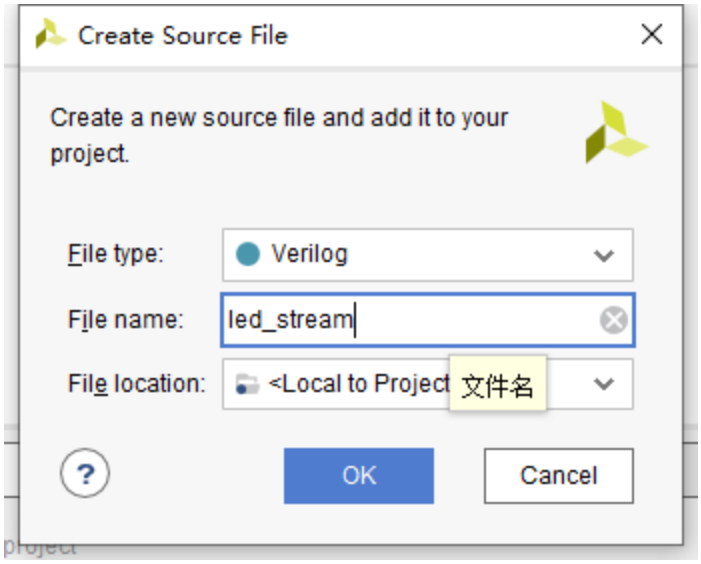
我们创建一些代码.


Add Sources
VIVADO
HLx Editions
Add Sources
This guides you through the process of adding and creating sources for your project
Add or create constraints
Add or create design sources
Add or create simulation sources
XILINX
< Back
Next >
Finish
Cancel

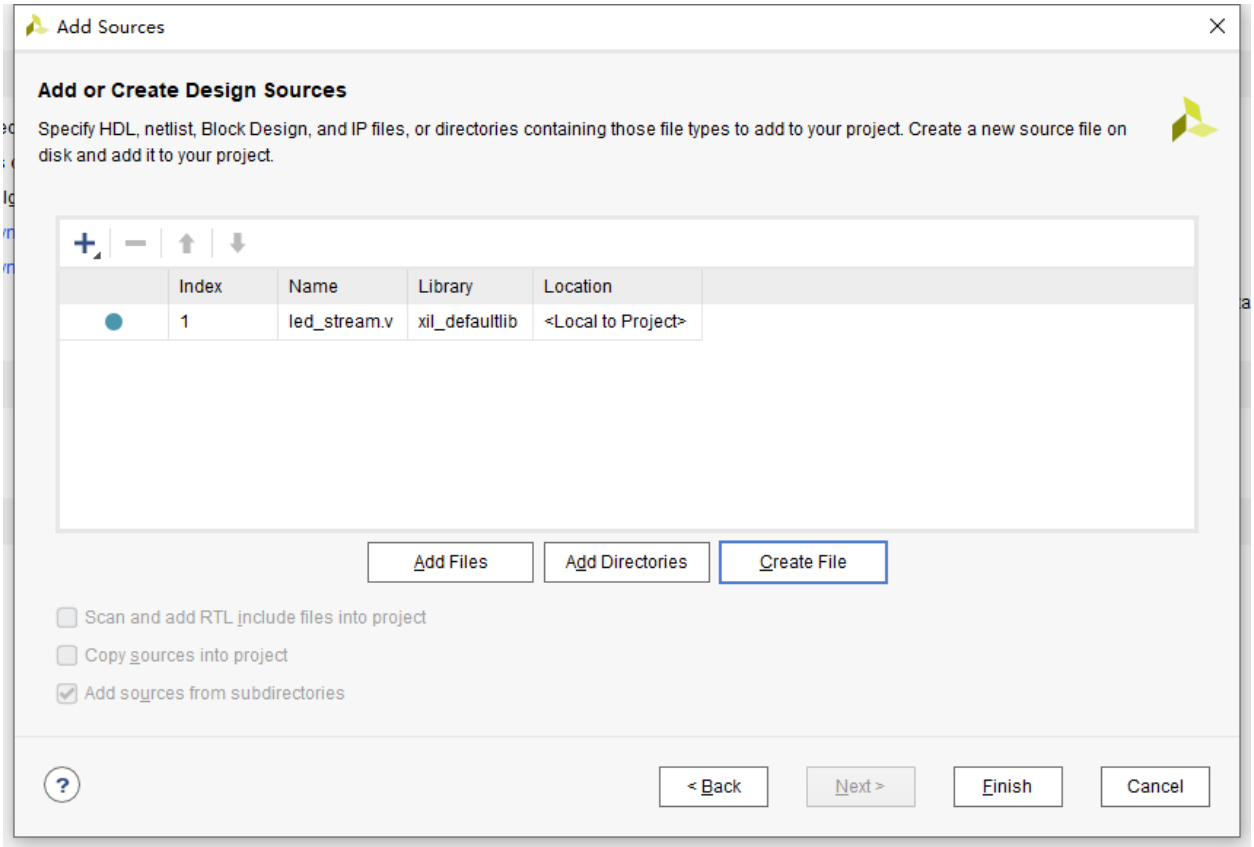
创建代码文件.



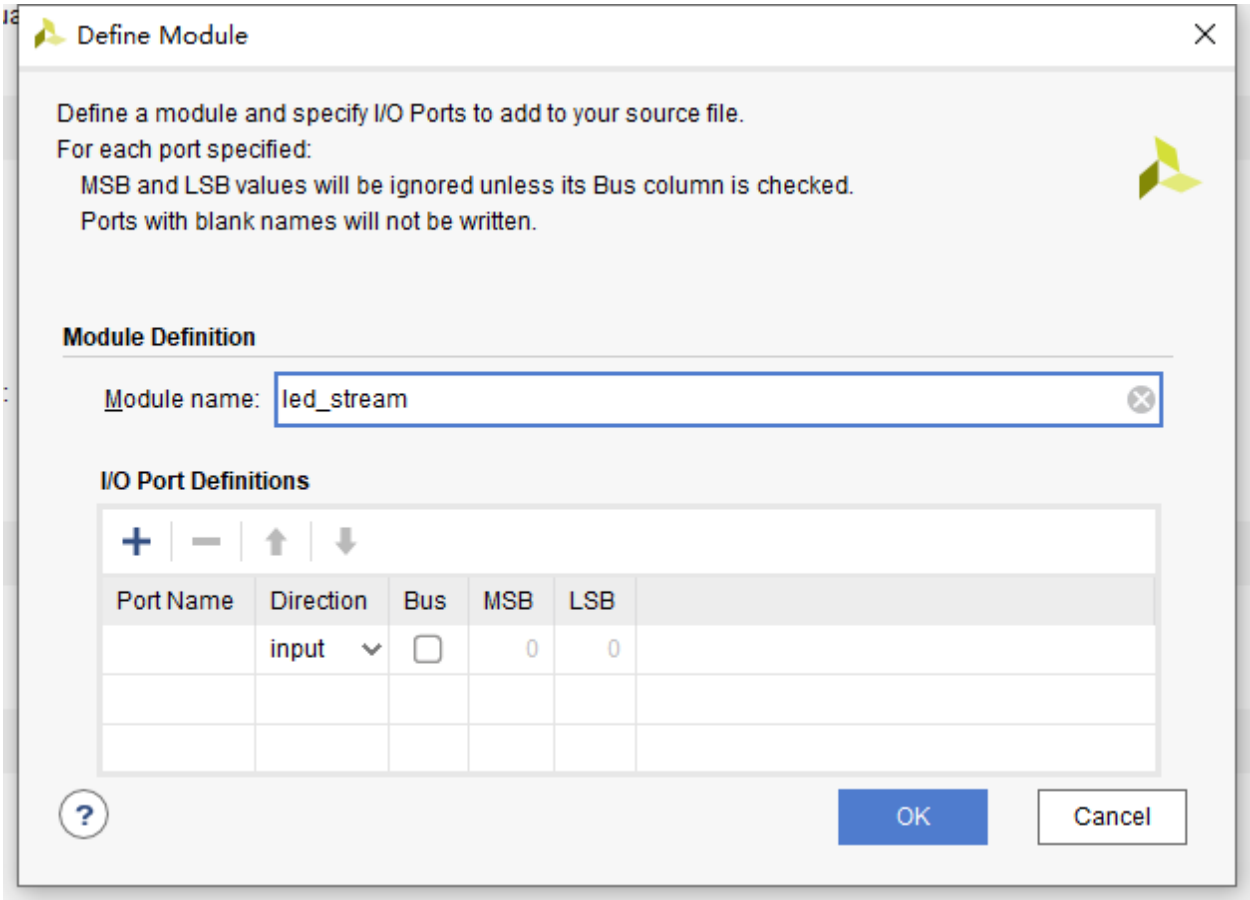
写一个文件名,一般来说入口文件和工程名一样,不过不是规定的.



接着点完成就会加入到工程里.



要给模块写一个名字,默认就好,一般和文件名一样.



默认源码是这样的.

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2022/02/19 23:07:24
// Design Name:
// Module Name: led_stream
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
```

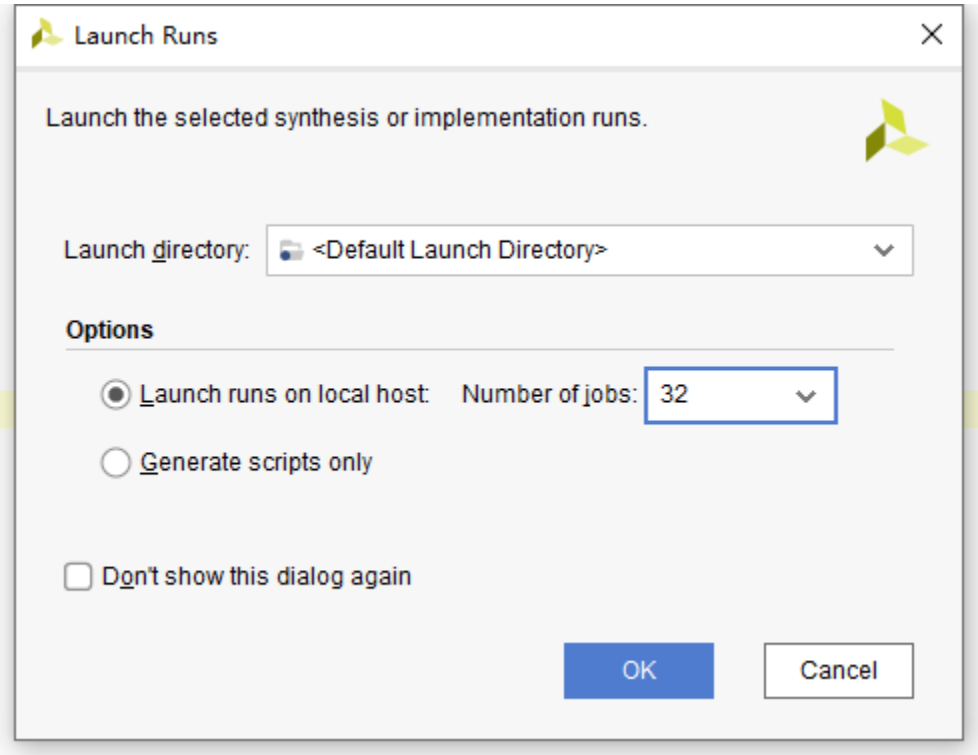
```
module led_stream(

    );
endmodule
```

添加上对应的逻辑代码.

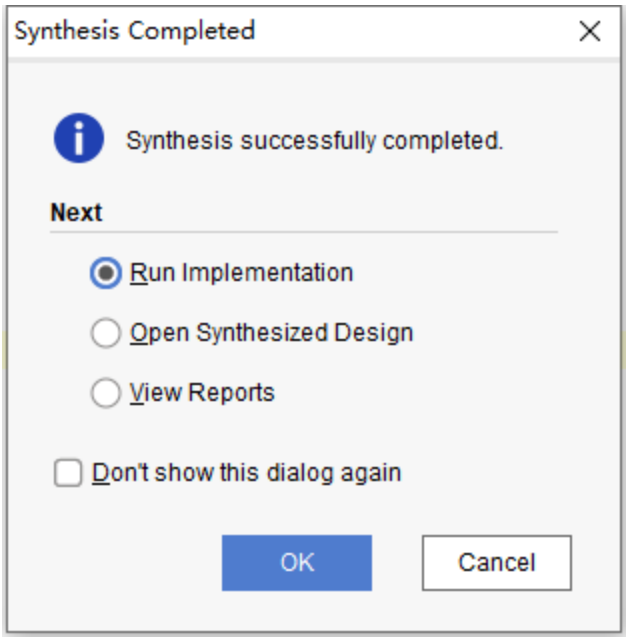
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2022/02/19 23:07:24
// Design Name:
// Module Name: led_stream
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module led_stream(output reg [3:0] led, // 显示4个LED
                  input clk,            // 时钟输入
                  input rst);           // 复位输入

    reg [31:0] cnt; // 这是一个32Bit计数器.

    always@(posedge clk or negedge rst)
    begin
        if (!rst)
        begin
            cnt <= 'b0;
            led <= 'b0001;
        end
        else
        begin
            if (cnt == 25000000)
            begin
                cnt <= 'b0;
                led <= {led[0],led[3:1]};
            end
```

```
            else
            begin
                cnt <= cnt + 'b1;
            end
        end
    end

endmodule
```

选择Run Synthesis综合一下,然后才能继续配置,建议选择多线程综合,特别是大的工程优势特别明显.不过在流水灯这里都一样是瞬间的事情.

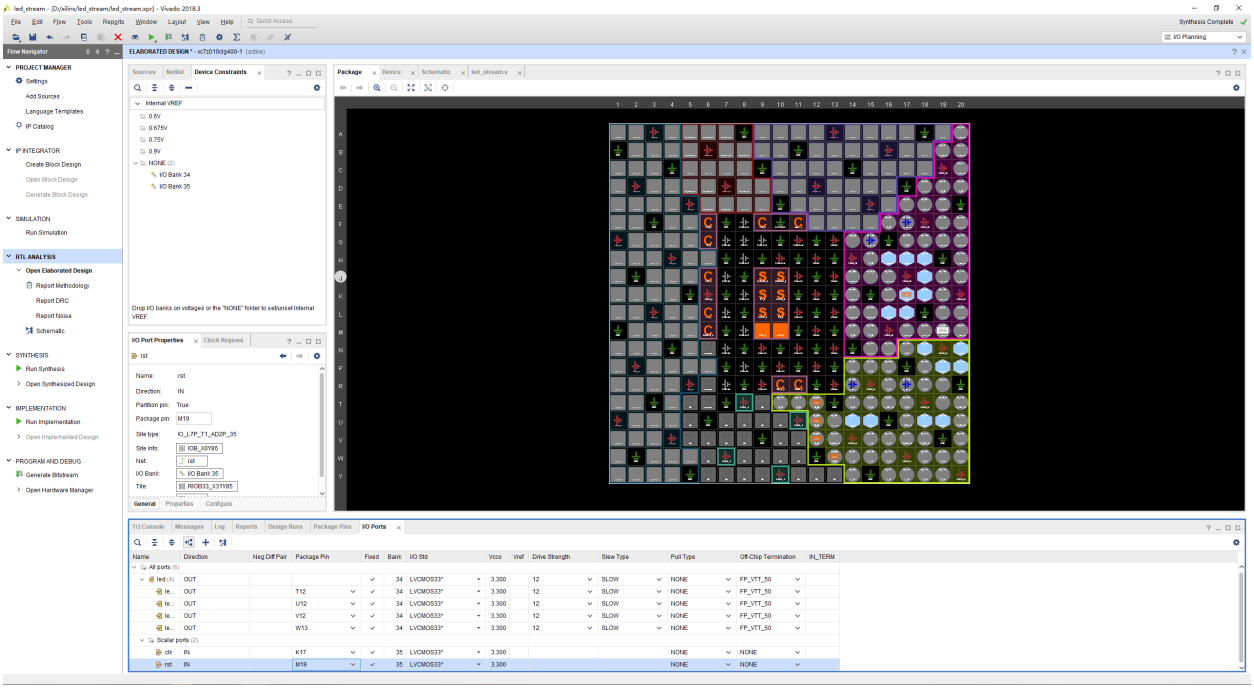


由于还没有绑定引脚,所以暂时不用实现,取消即可.

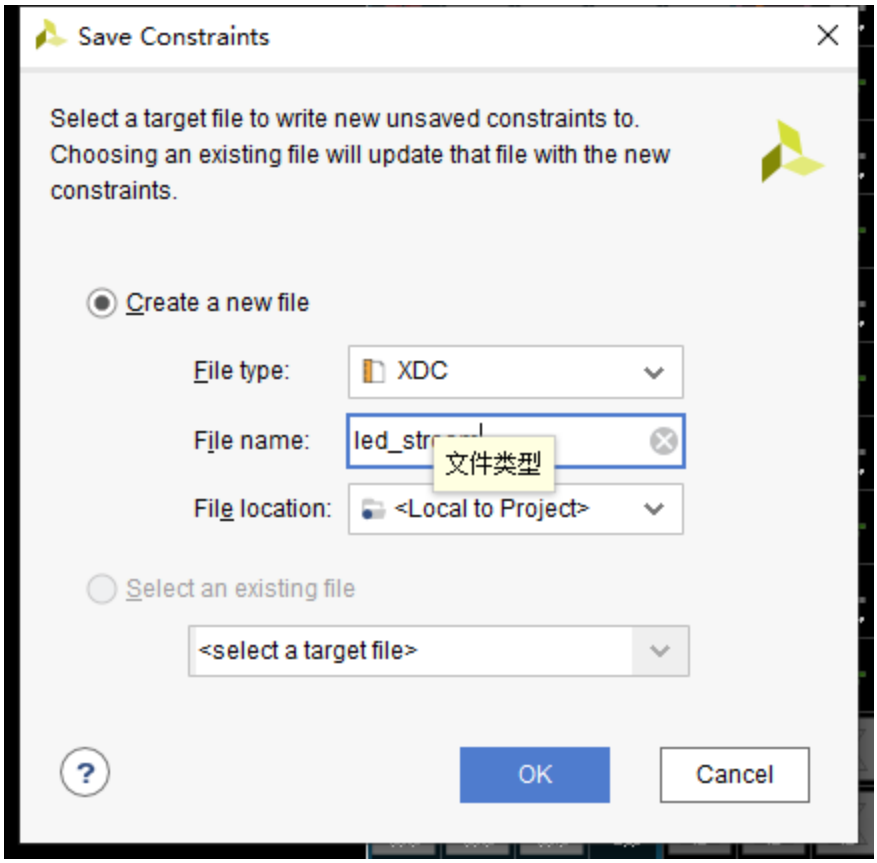


选择左侧的RTL ANALYSIS→Open Elaborated Design,然后选择右上角的I/O Planning开始配置引脚.

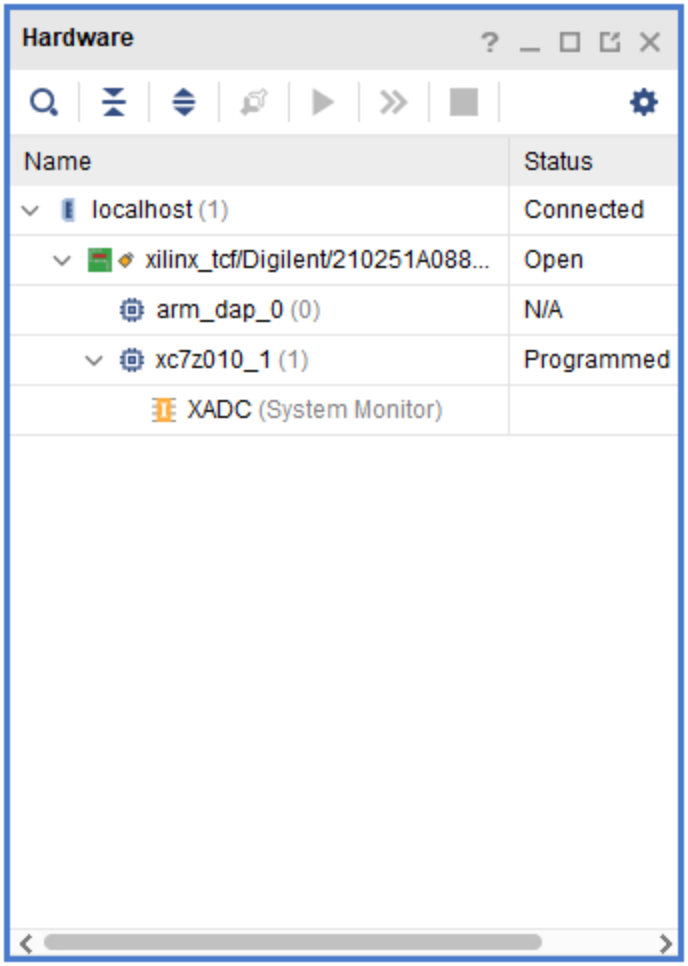
比如这里的四个LED分别在T12,U12,V12,W13,而时钟在K17,复位在M19.



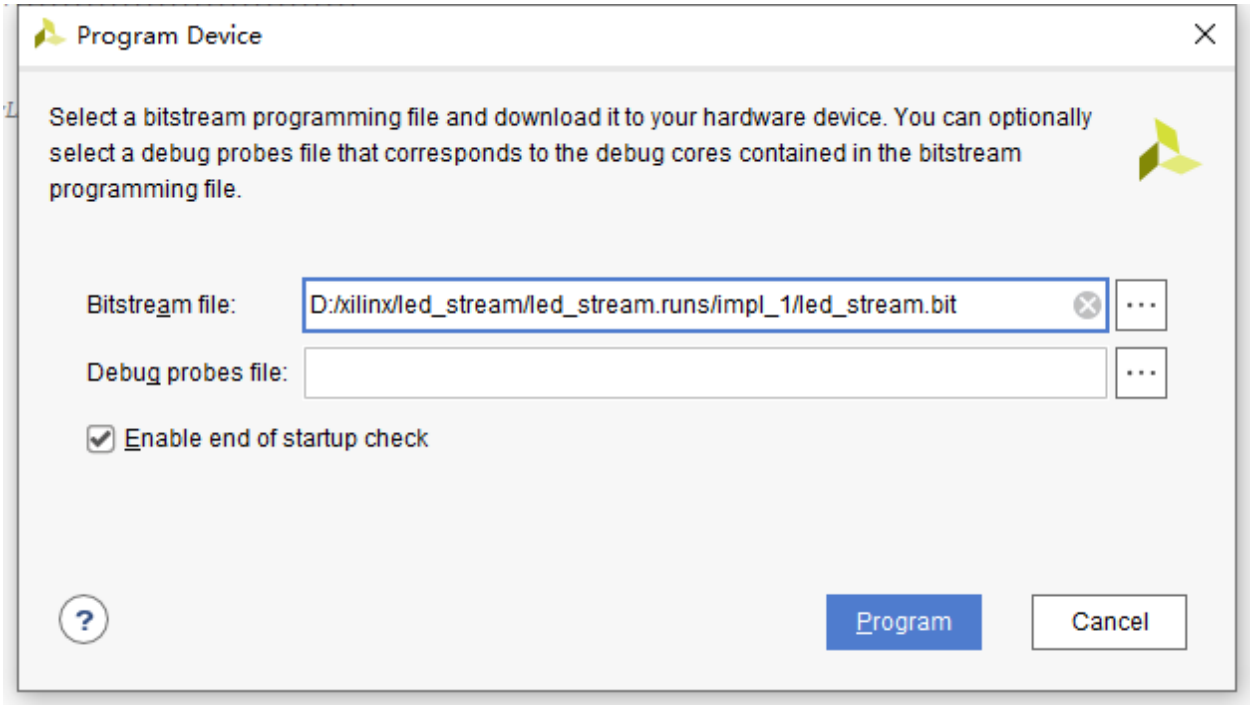
然后Ctrl+S保存一下,名字就和工程名一样,不过不是强制的.



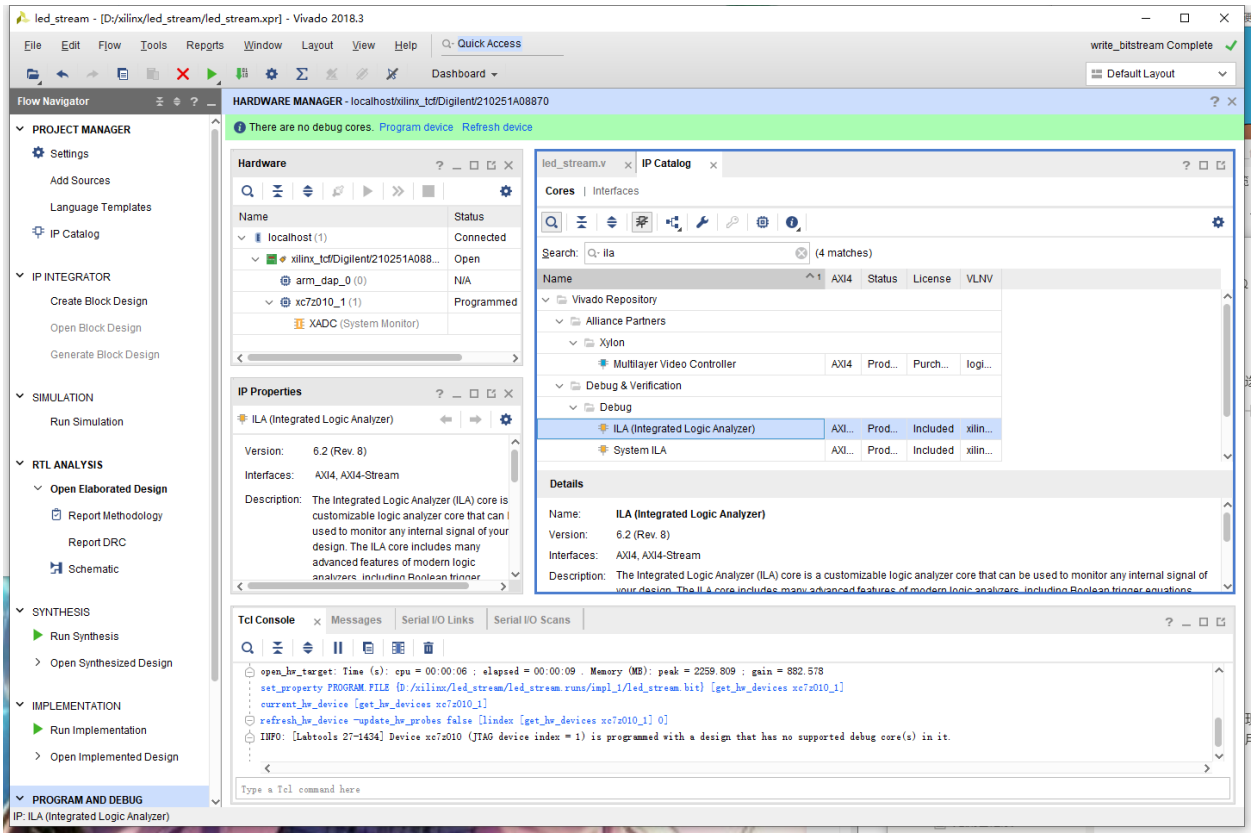
现在可以开始生成Bitstream了,生成Bitstream后Open Hardware Manager就可以看到设备.当然前提是已经连接正常,如果看不到就手动Auto Connect刷新一下.



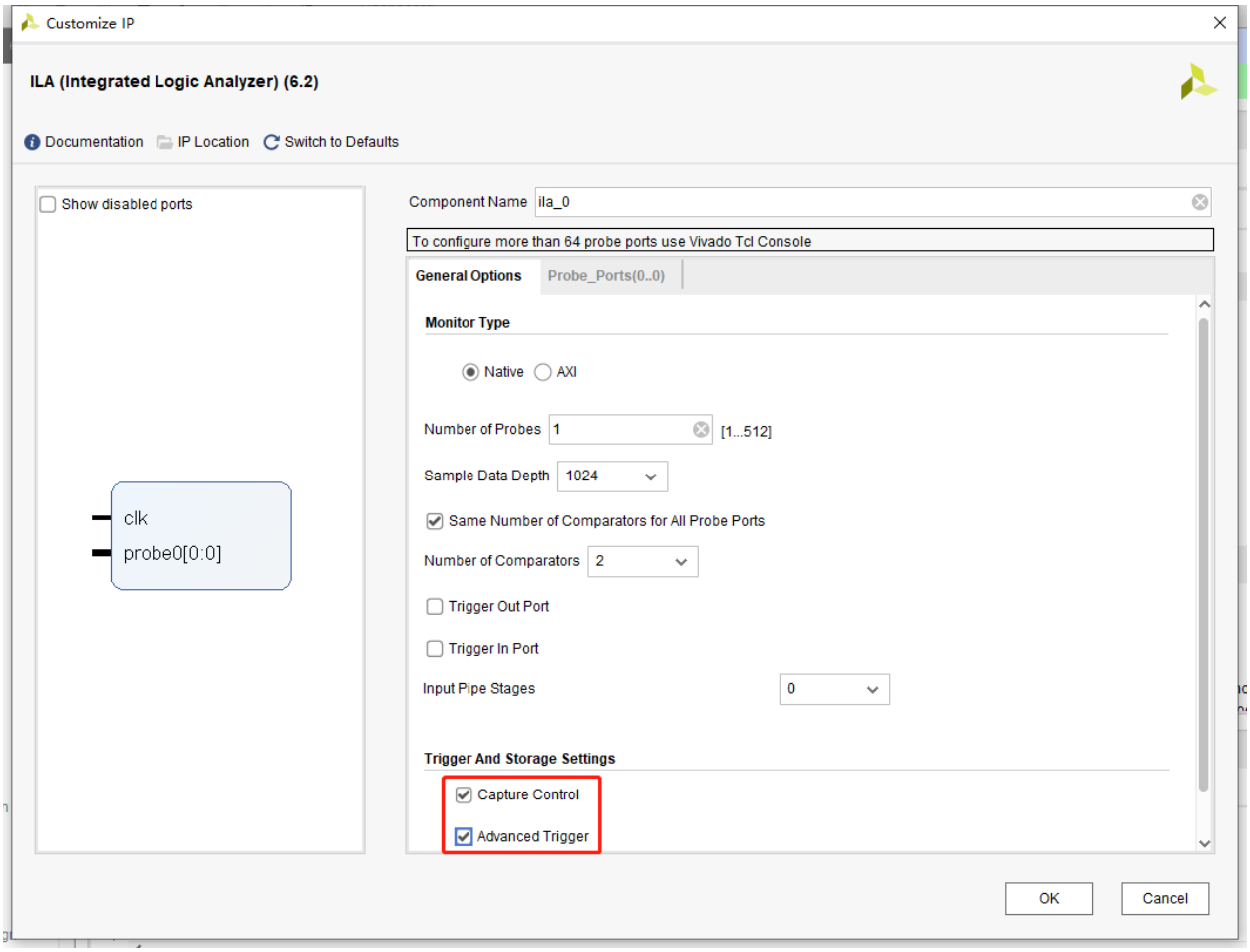
选择顶部的Program device来编程器件,默认会选上刚才生成的Bitstream.



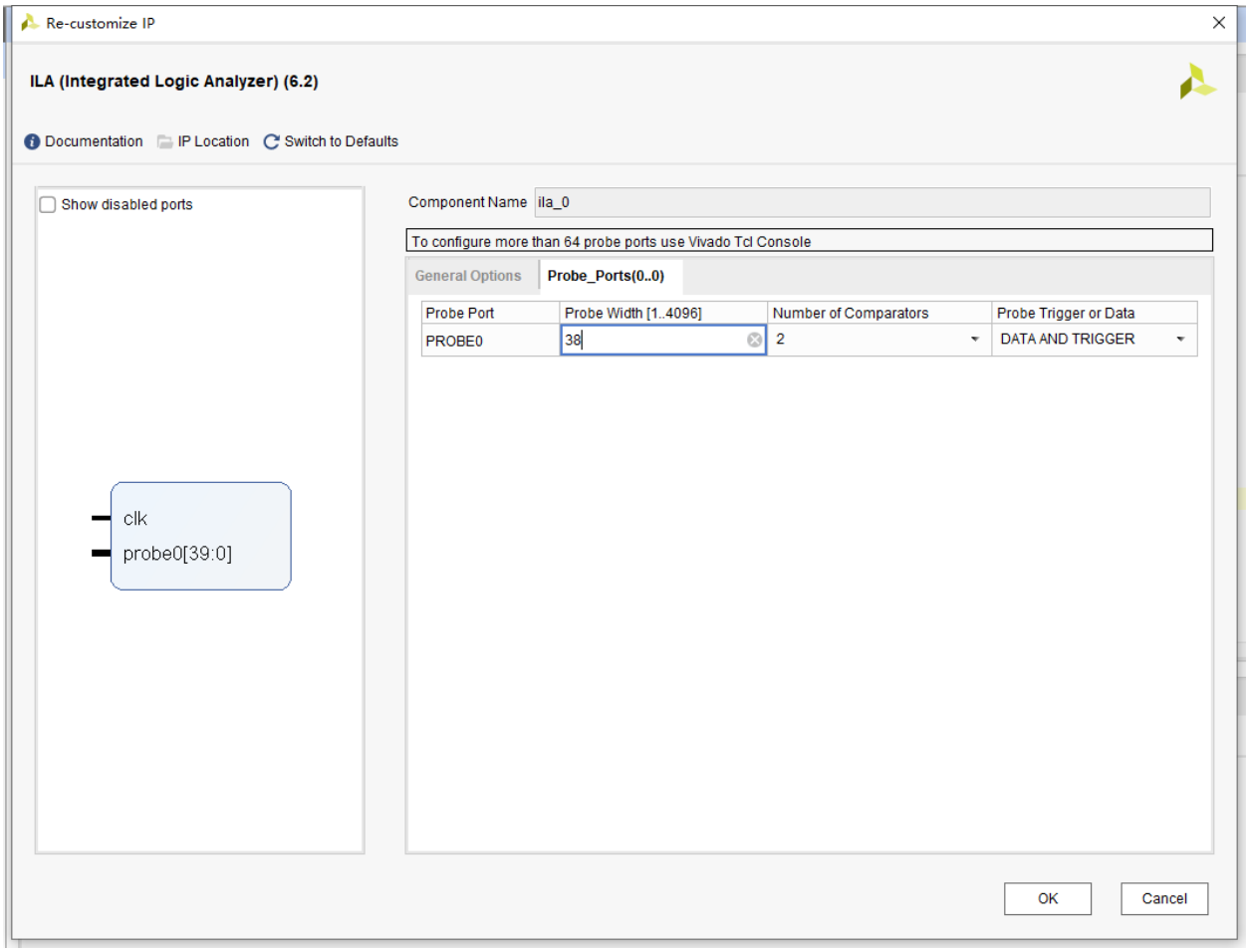
现在应该能看到流水灯了,当然现在下载到的是FPGA内部的RAM,因此掉电就没了,那怎么调试呢,可以用ILA,现在打开IP Catalog,然后搜ILA,选ILA (Integrated Logic Analyzer),注意这玩意平时也要占资源的.



尝试勾选Capture Control,Advance Trigger.



这个例子里有LED,时钟输入输出,计数器几个变量,这里我就选用38个探针,足够用了,如果写多写少了,会出现综合异常.

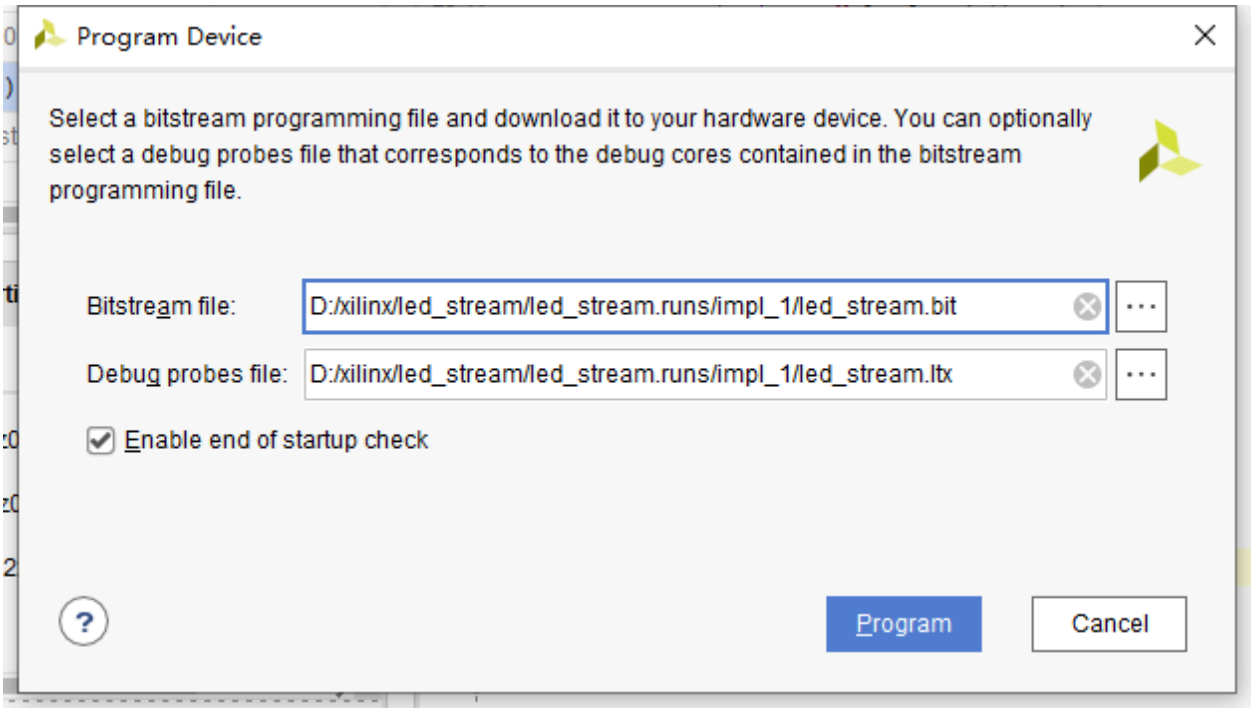


然后一路确认生成就行,最后尝试实例化他,终极代码如下.

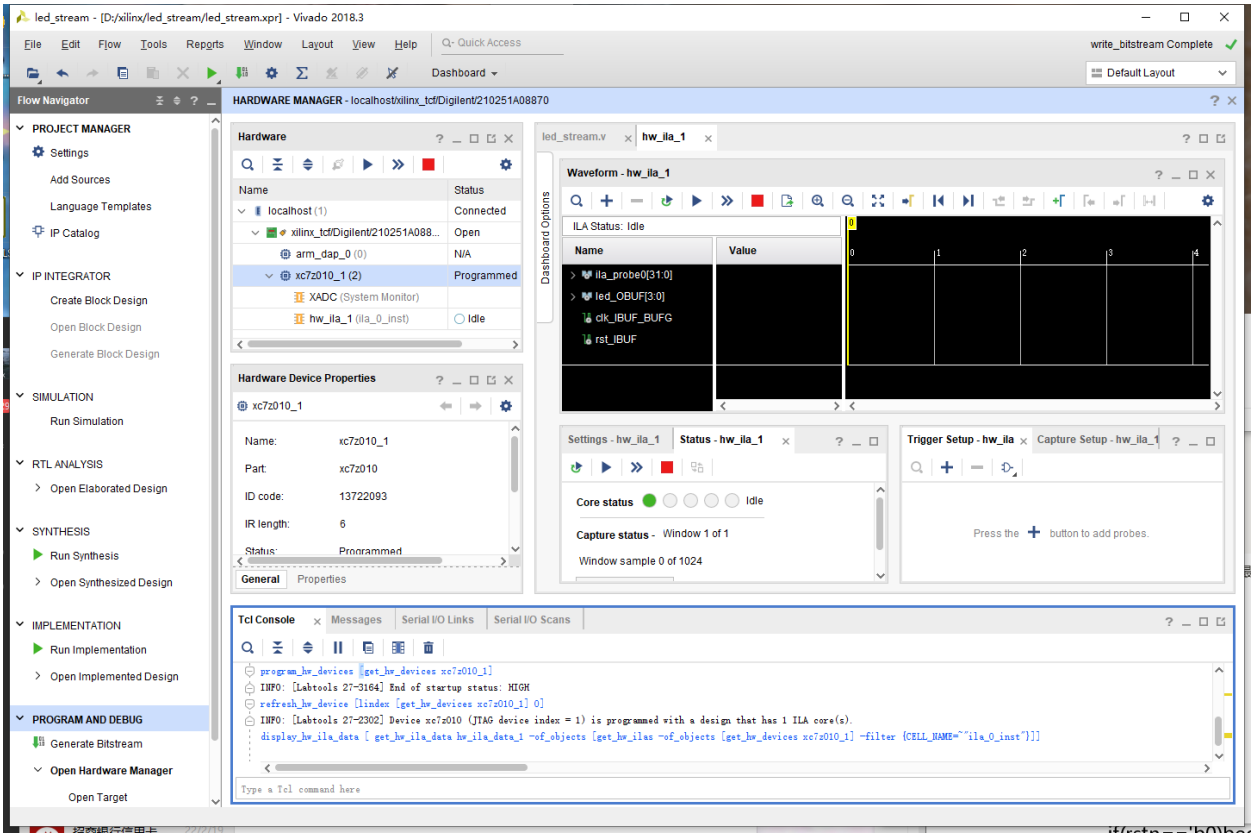
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2022/02/19 23:07:24
// Design Name:
// Module Name: led_stream
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
```

```
//////////////////////////////////////////////////////////////////////////////////


module led_stream(output reg [3:0] led, // 显示4个LED
                  input clk,            // 时钟输入
                  input rst);           // 复位输入

    reg [31:0] cnt; // 这是一个32Bit计数器.

    wire [37:0] ila_probe0; // ILA探针
    assign ila_probe0[31:0] = cnt[31:0];
    assign ila_probe0[35:32] = led[3:0];
    assign ila_probe0[36] = clk;
    assign ila_probe0[37] = rst;

    ila_0 ila_0_inst
    (
        .clk(clk),
        .probe0(ila_probe0)
    );

    always@(posedge clk or negedge rst)
    begin
        if (!rst)
        begin
            cnt <= 'b0;
            led <= 'b0001;
        end
        else
        begin
            if (cnt == 25000000)
            begin
                cnt <= 'b0;
                led <= {led[0],led[3:1]};
            end
            else
            begin
                cnt <= cnt + 'b1;
            end
        end
    end

endmodule
```

最后也要综合并生成bitstream写入FPGA.



看到ILA了,他就像逻辑分析仪一样.



功能应该能摸索出来了.