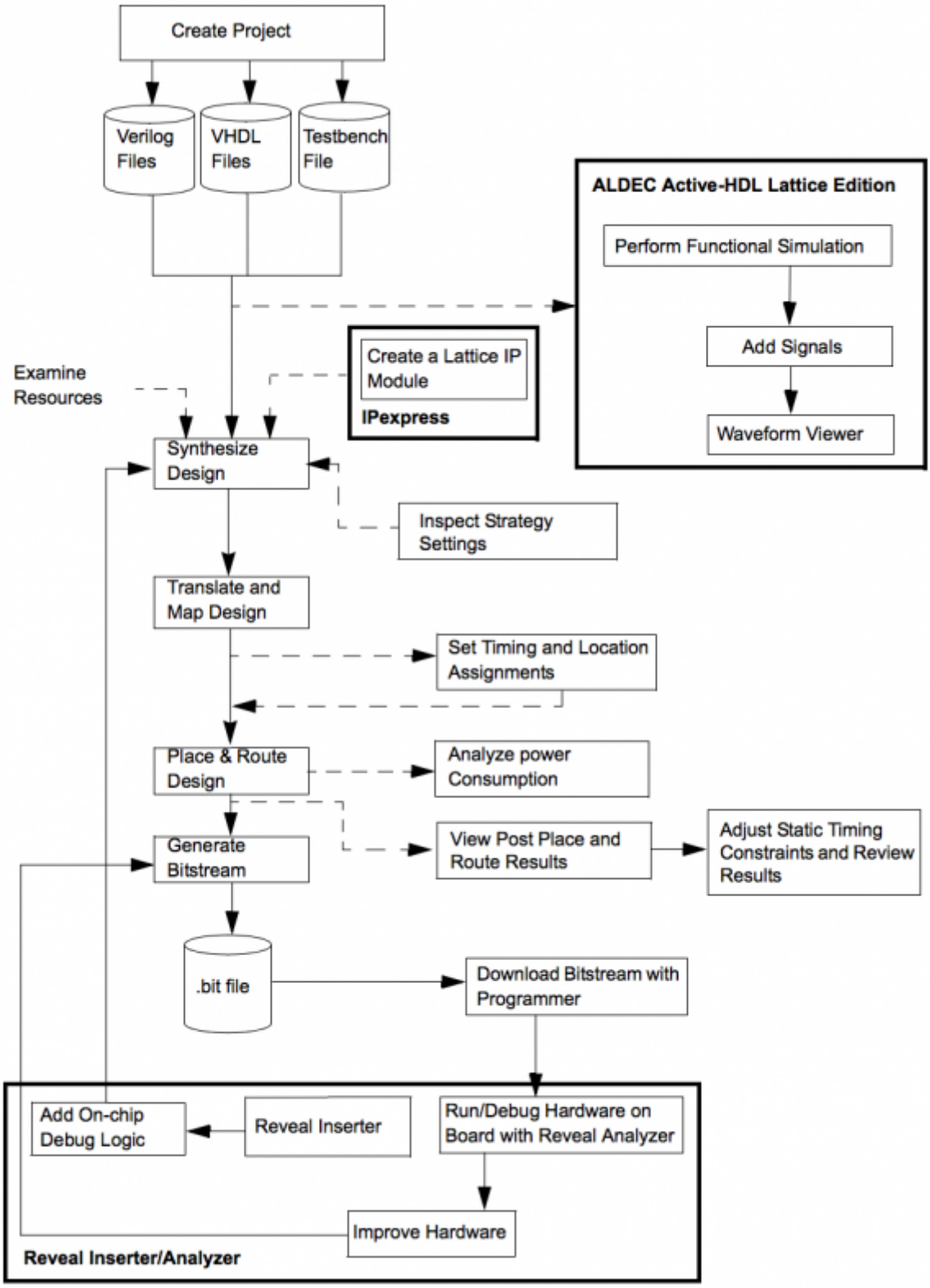
Consulte Instalación y configuración de Diamond para instalar Diamond. Si encuentra algún problema, primero puede consultar las Preguntas frecuentes sobre la instalación de Diamond . Ahora podemos usar el software Diamond para iniciar el diseño de FPGA. Consulte la siguiente figura para ver el flujo de diseño completo.

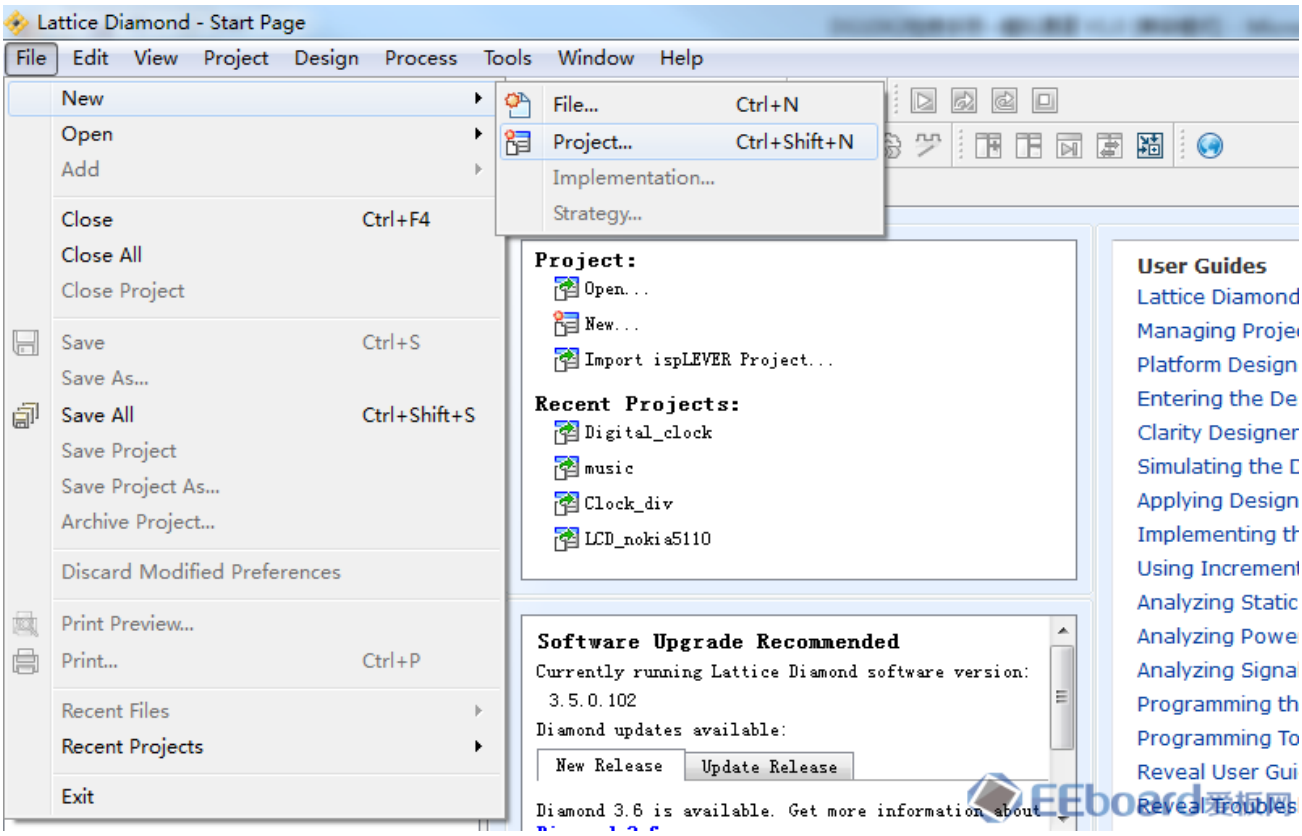


Utilice Diamond para diseñar el flujo básico de la lógica FPGA

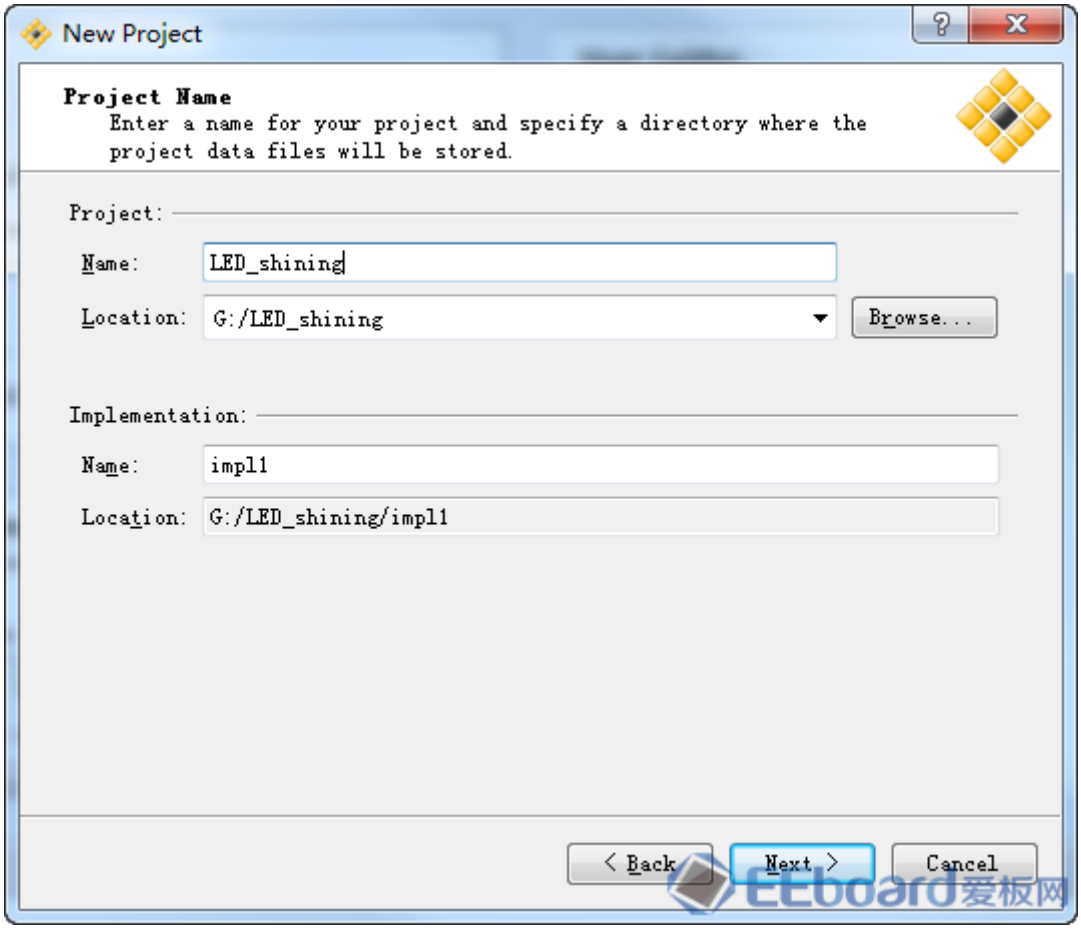
==== 1 Ejecute la primera rutina ====

A continuación podemos iniciar el desarrollo de la lógica programable, tomamos el control del LED parpadeando alternativamente como ejemplo para completar nuestro primer programa:

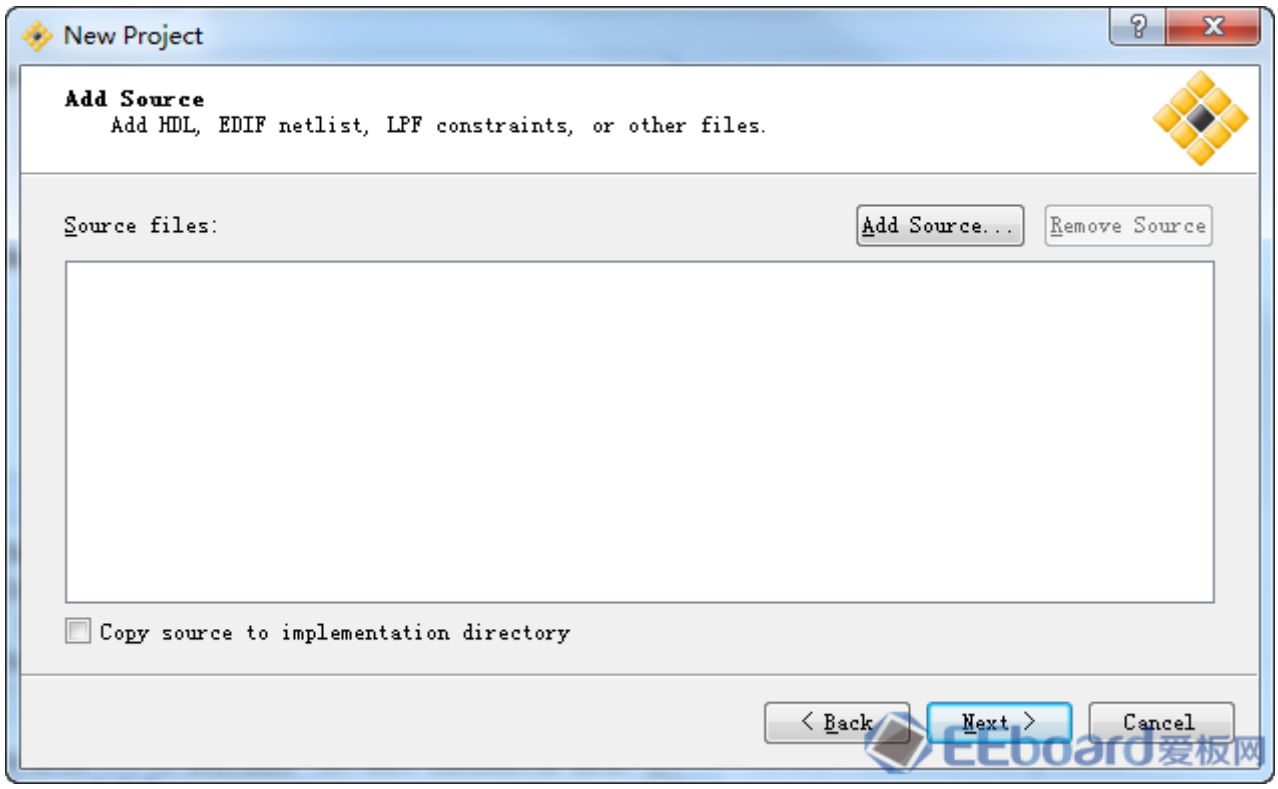
1. Haga doble clic para ejecutar el software Diamond, primero cree un nuevo proyecto: seleccione Archivo → Nuevo → Proyecto → Siguiente



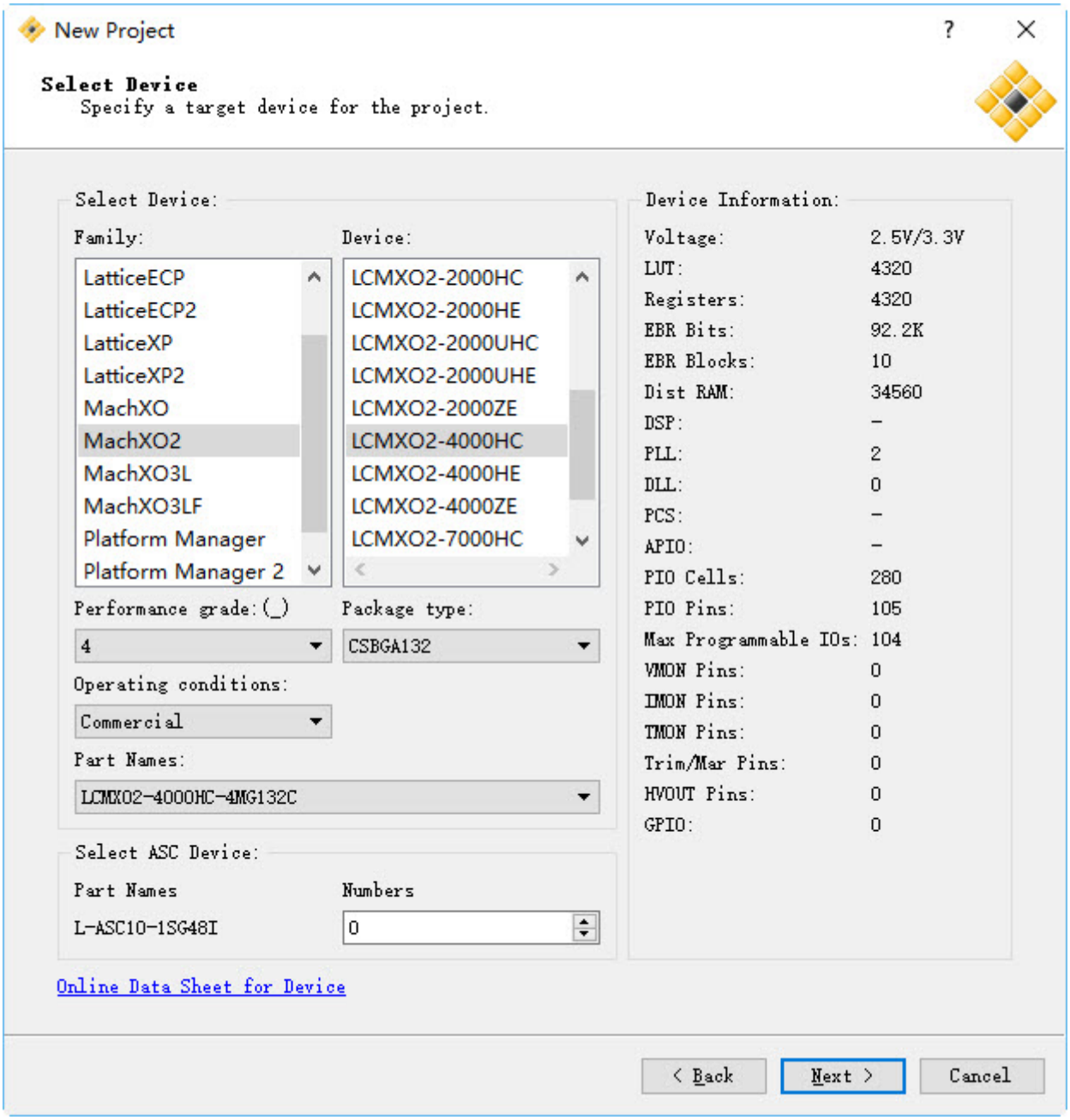
2. Denominación del proyecto: nombraremos el nuevo LED del proyecto *brillando, el directorio del proyecto G: / LED* brillando, y luego haremos clic en Siguiente



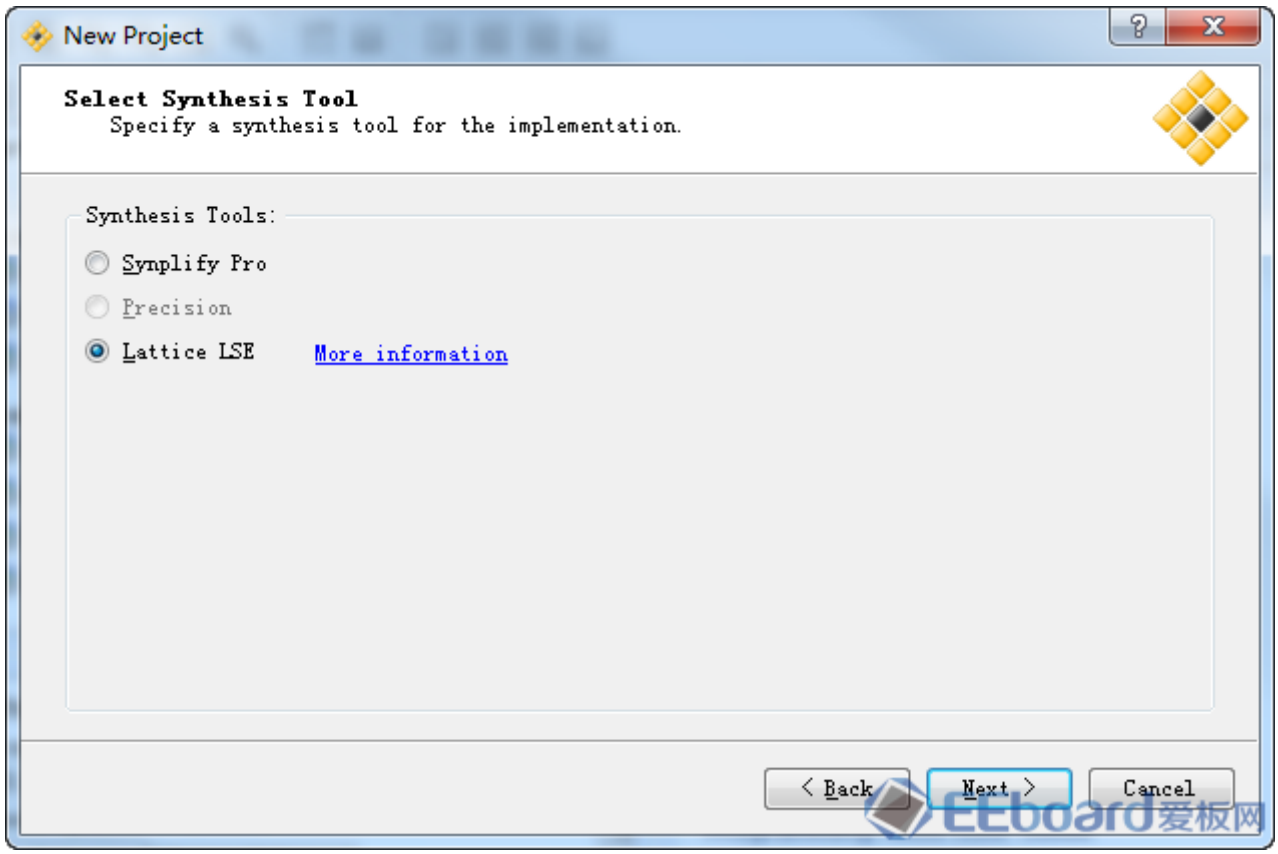
3. Agregue archivos de diseño relacionados o archivos de restricción (si ya hay archivos de diseño y archivos de restricción, podemos optar por agregarlos al proyecto): aquí creamos un nuevo proyecto, no hay archivos relacionados, no es necesario agregar, simplemente vaya a Siguiente



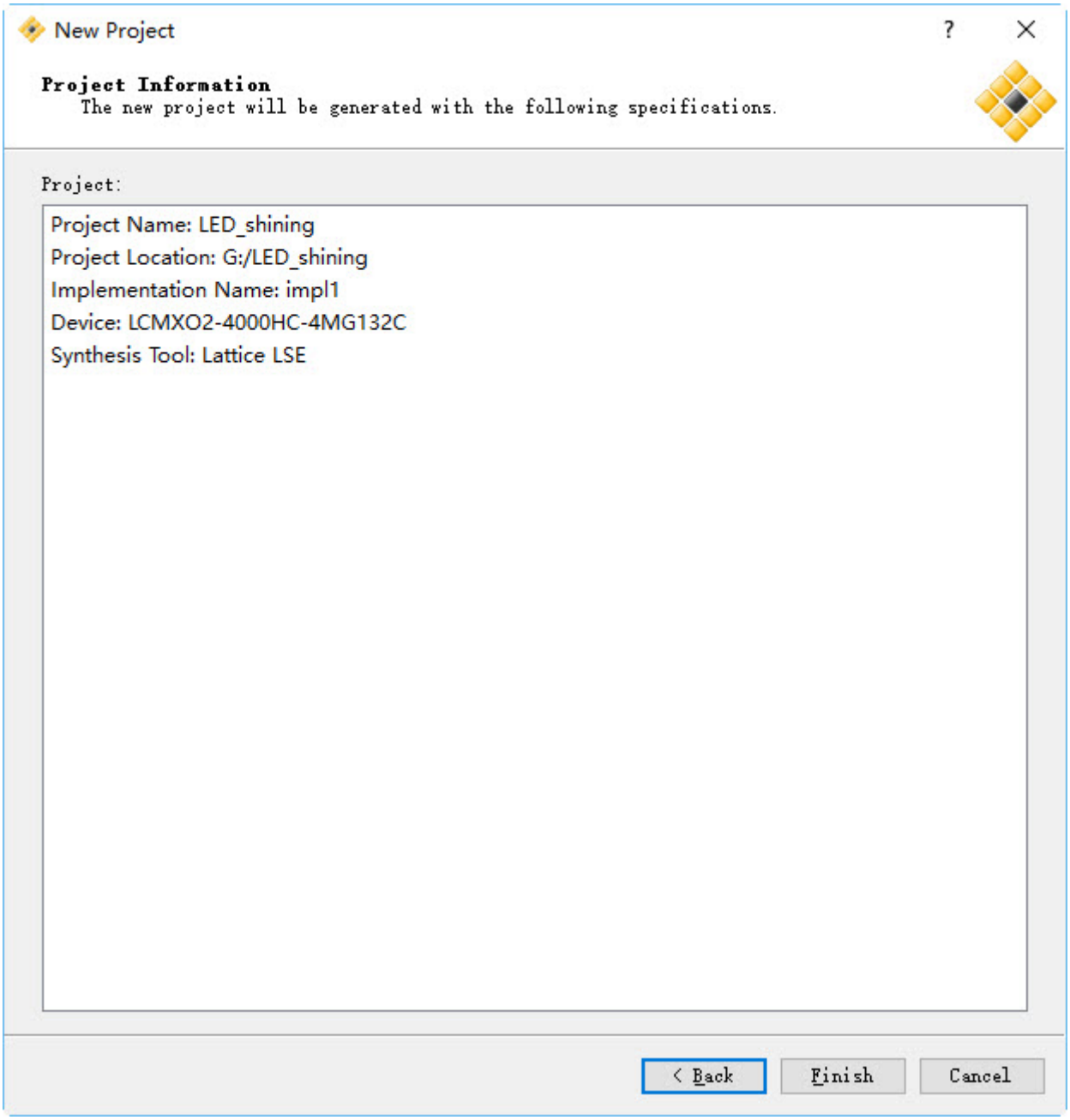
4. **Selección de dispositivo** : Configure de acuerdo con el dispositivo de placa de desarrollo de paso FPGA LCMXO2-4000HC-4MG132C, Siguiente (Se debe confirmar que el modelo del dispositivo es correcto; de lo contrario, se informará un error durante la configuración del pin)



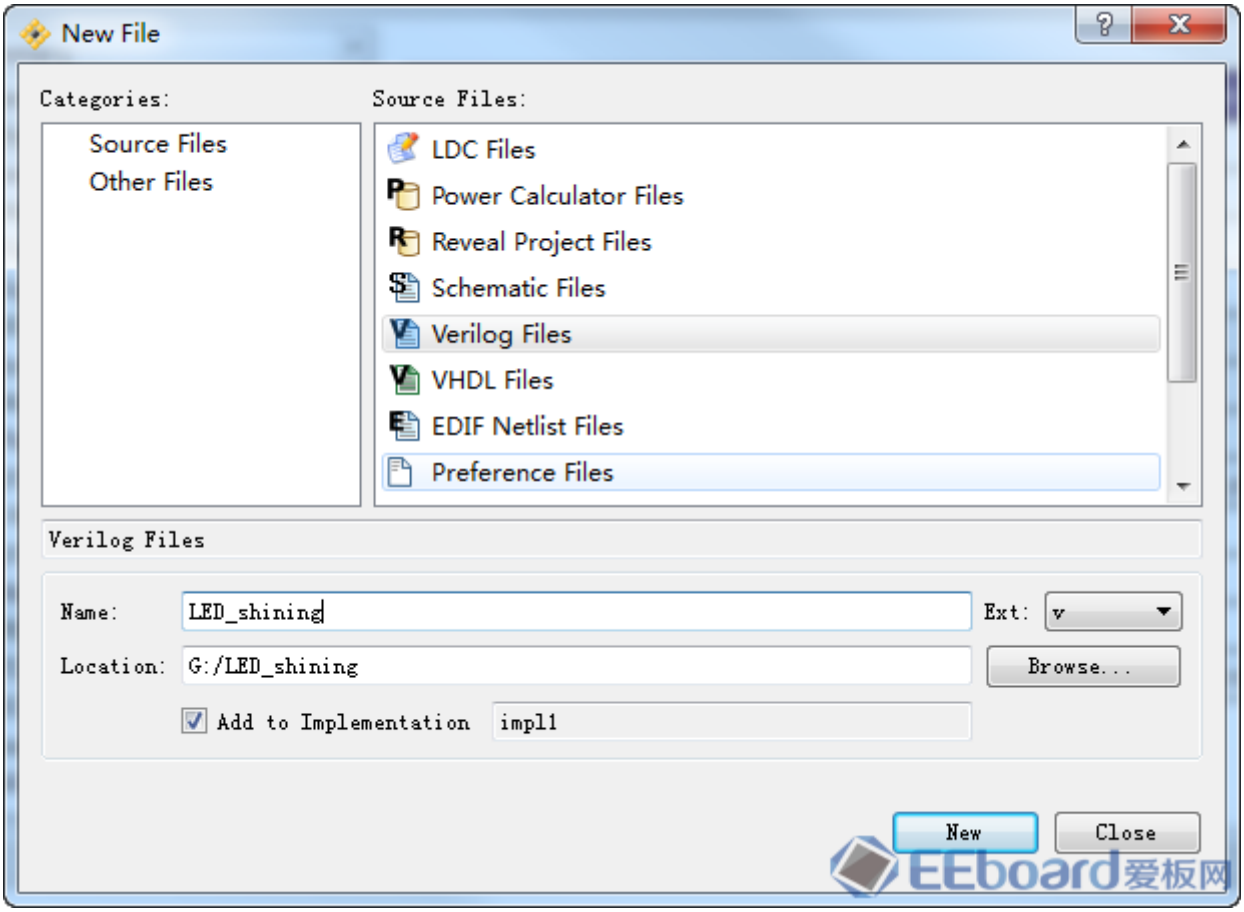
5. Elija una herramienta completa: Synplify Pro (tercero) y Lattice LSE (fábrica original) están disponibles, usaremos Lattice LSE y directamente Siguiente



6. Confirmación de la información del proyecto: toda la información seleccionada arriba está aquí, confirme que no hay problema, directamente Finalizar



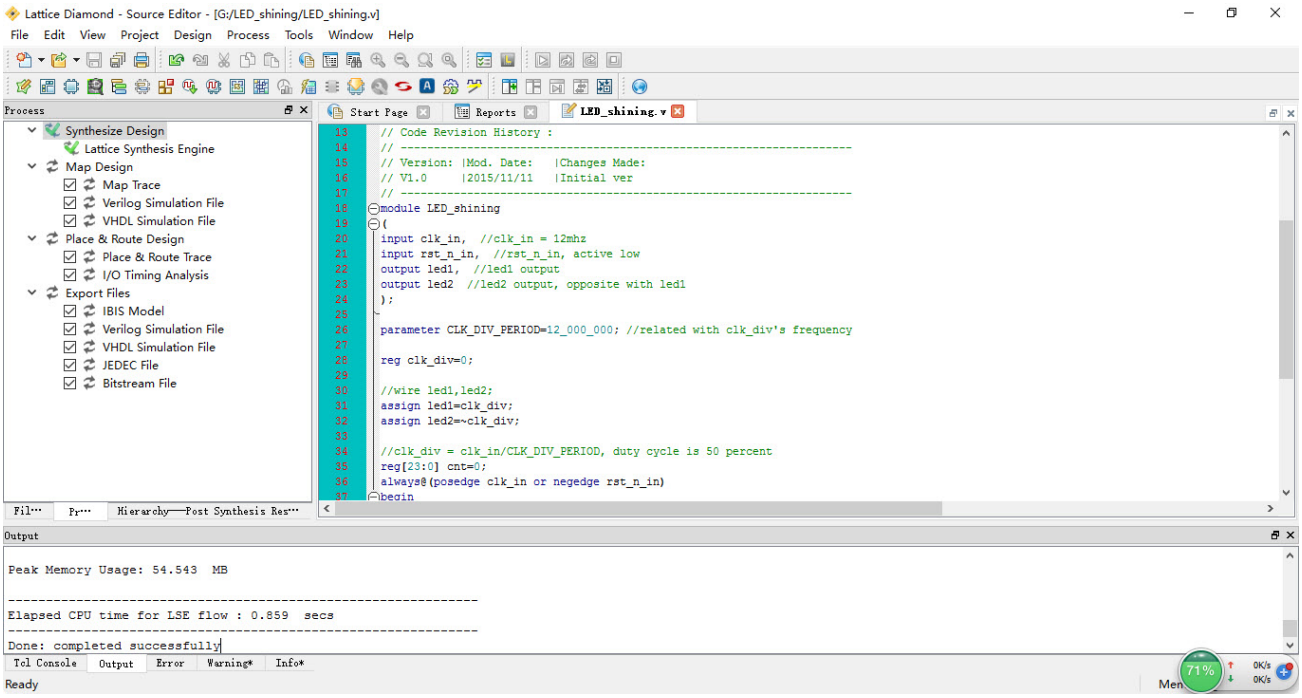
7. El proyecto ha sido construido, agregamos el archivo de diseño a continuación, seleccionamos Archivo → Nuevo → Archivo
8. 选择Verilog Files（选择自己使用的硬件描述语言），Name填写LED*shining，然后点击New，这样我们就创建了一个新的设计文件LED*shining.v，然后我们就可以在设计文件中进行编程了



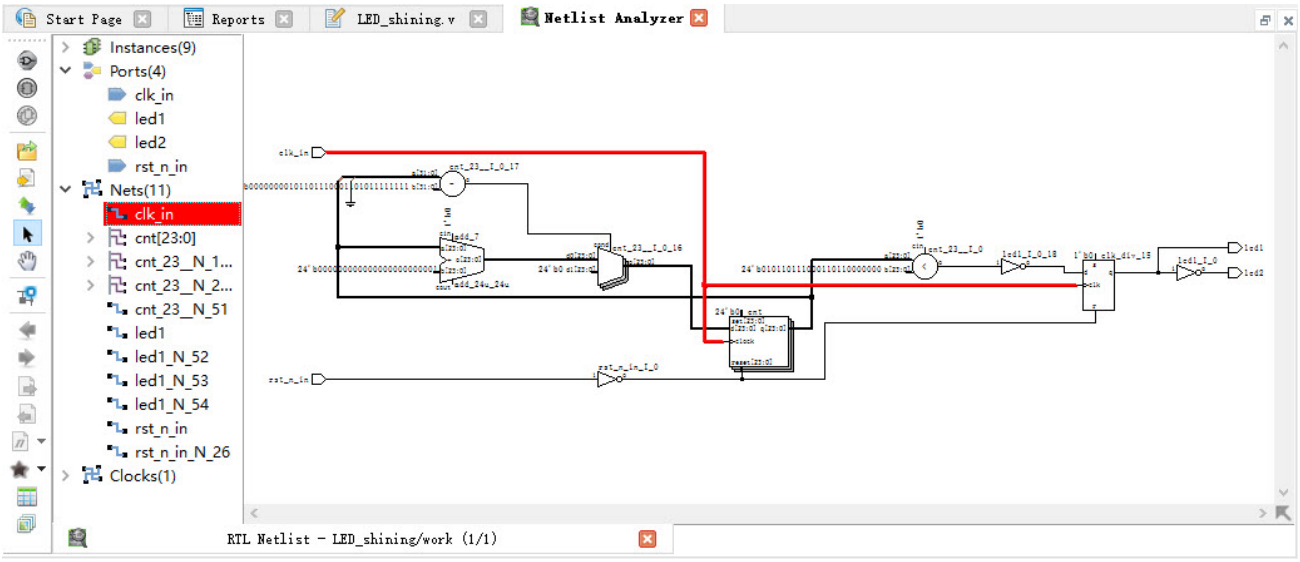
9. 程序源码已经准备好，如下，将代码复制到设计文件LED_shining.v中，并保存。

```
// ---------------------------------------------------------------------
// >>>>>>>>>>>>>>>>>>>>>>> COPYRIGHT NOTICE <<<<<<<<<<<<<<<<<<<<<<<
// ---------------------------------------------------------------------
// Module: LED_shining
//
// Author: Step
//
// Description: LED_shining
//
// Web: www.stepfpga.com
//
// ---------------------------------------------------------------------
// Code Revision History :
// ---------------------------------------------------------------------
// Version: |Mod. Date:   |Changes Made:
// V1.0     |2015/11/11   |Initial ver
// ---------------------------------------------------------------------
module LED_shining
(
input clk_in,             //输入系统12MHz时钟
input rst_n_in,           //输入复位信号
output led1,              //输出led1
output led2               //输出led2，与led1取反
);
parameter CLK_DIV_PERIOD=12_000_000; //分频常数定义
reg clk_div=0;            //定义reg型变量，用作分频后时钟输出
//wire led1,led2;           //wire型变量定义，可以省略，verilog里默认是wire型
assign led1=clk_div;      //持续赋值语句，将分频后时钟赋给led1，产生闪烁效果
assign led2=~clk_div;     //取反赋值给led2，与led1形成交替闪烁
//偶数分频电路 clk_div = clk_in/CLK_DIV_PERIOD，占空比50%，CLK_DIV_PERIOD必须为偶数
reg[23:0] cnt=0;                  //分频用的计数器，2**cnt-1>CLK_DIV_PERIOD,计数器最大值要大于分
频常数
always@(posedge clk_in or negedge rst_n_in)
begin
        if(!rst_n_in)
                begin
                        cnt<=0;
                        clk_div<=0;
                end
        else begin
                if(cnt==(CLK_DIV_PERIOD-1)) cnt<=0;
                else cnt<=cnt+1'b1;
                if(cnt<(CLK_DIV_PERIOD>>1)) clk_div<=0;
                else clk_div<=1;
        end
end
endmodule
```

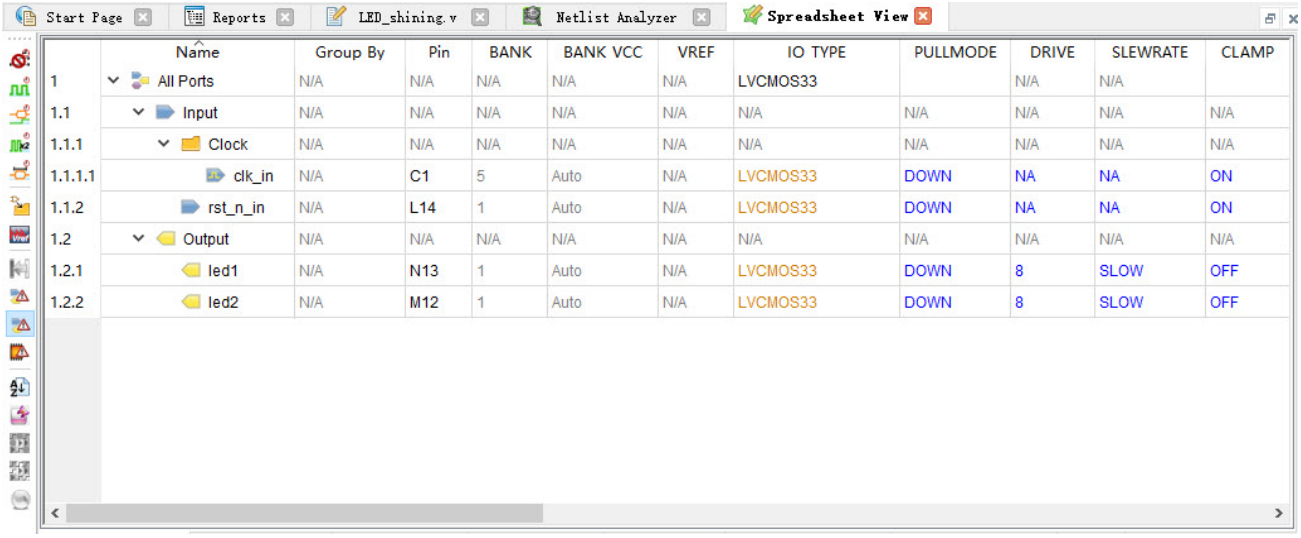
1. 程序编写完成，需要综合，在软件左侧Process栏，选择Process，双击Synthesis Design，对设计进行综合，综合完成后Synthesis Design显示绿色对勾（如果显示红色叉号，说明代码有问题，根据提示修改代码），如图



2. 通过综合工具，我们的代码就被综合成了电路，生成的具体电路，我们可以通过选择Tools → Netlist Analyzer查看（仅限Lattice的综合工具，第三方综合工具无法查看），如图

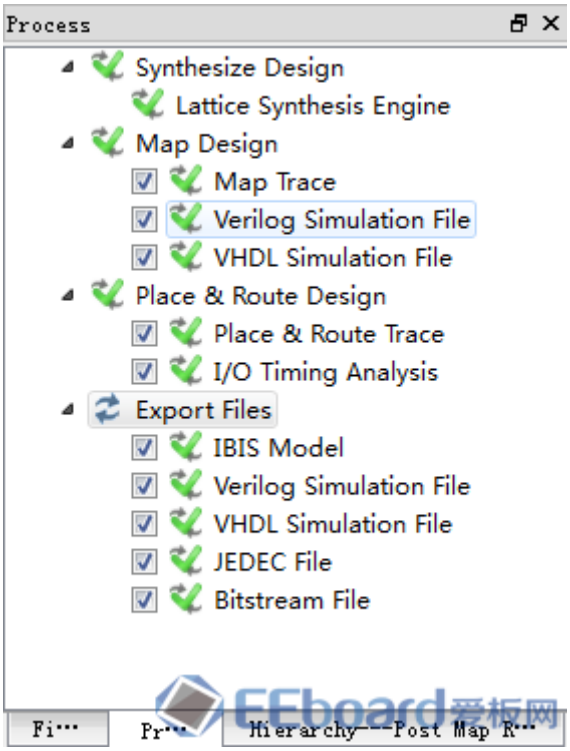


3. 综合生成电路后，分配管脚，选择Tools → Spreadsheet View,按照下图分配FPGA管脚，然后设置IO_TYPE为LVCMOS33，保存，界面如下



	Name	Group By	Pin	BANK	BANK VCC	VREF	IO TYPE	PULLMODE	DRIVE	SLEWRATE	CLAMP
1	All Ports	N/A	N/A	N/A	N/A	N/A	LVCMOS33		N/A	N/A	
1.1	Input	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.1.1	Clock	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.1.1.1	clk_in	N/A	C1	5	Auto	N/A	LVCMOS33	DOWN	NA	NA	ON
1.1.2	rst_n_in	N/A	L14	1	Auto	N/A	LVCMOS33	DOWN	NA	NA	ON
1.2	Output	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.2.1	led1	N/A	N13	1	Auto	N/A	LVCMOS33	DOWN	8	SLOW	OFF
1.2.2	led2	N/A	M12	1	Auto	N/A	LVCMOS33	DOWN	8	SLOW	OFF

4. 在软件左侧Process栏，选择Process，勾选所有选项，直接双击Export Files，所有布局布线输出依次完成，结束后，所有选项显示绿色对勾。

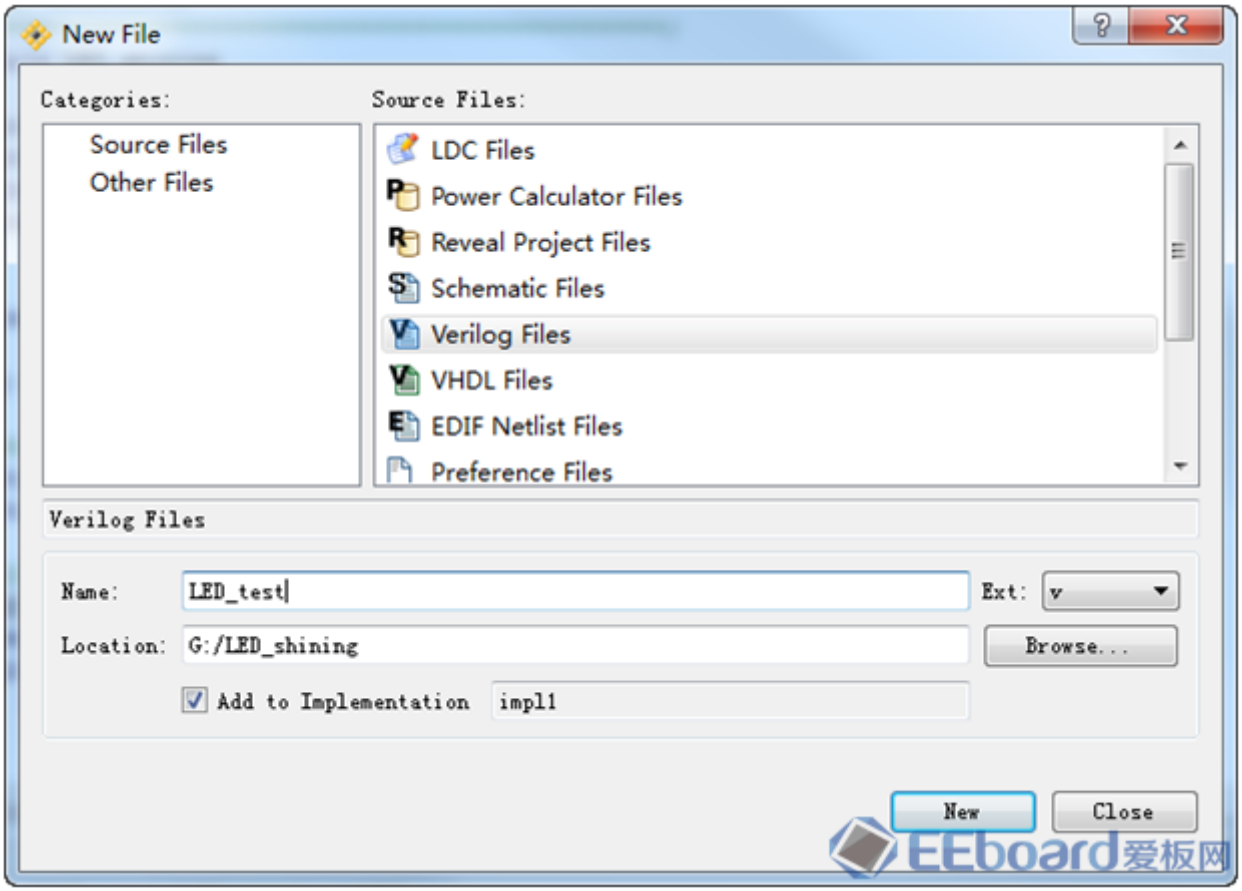


到这里完成了第一个程序流文件的生成，下面可以下载到FPGA中。

==== 2 工程仿真====

上面我们走了整个工程开发的过程，例程较为简单，对于复杂的工程开发需要预仿真和后仿真等，保证最终的程序设计逻辑和时序符合我们的设计要求。 仿真软件很多，这里我们使用软件自带的Active-HDL软件进行功能仿真：

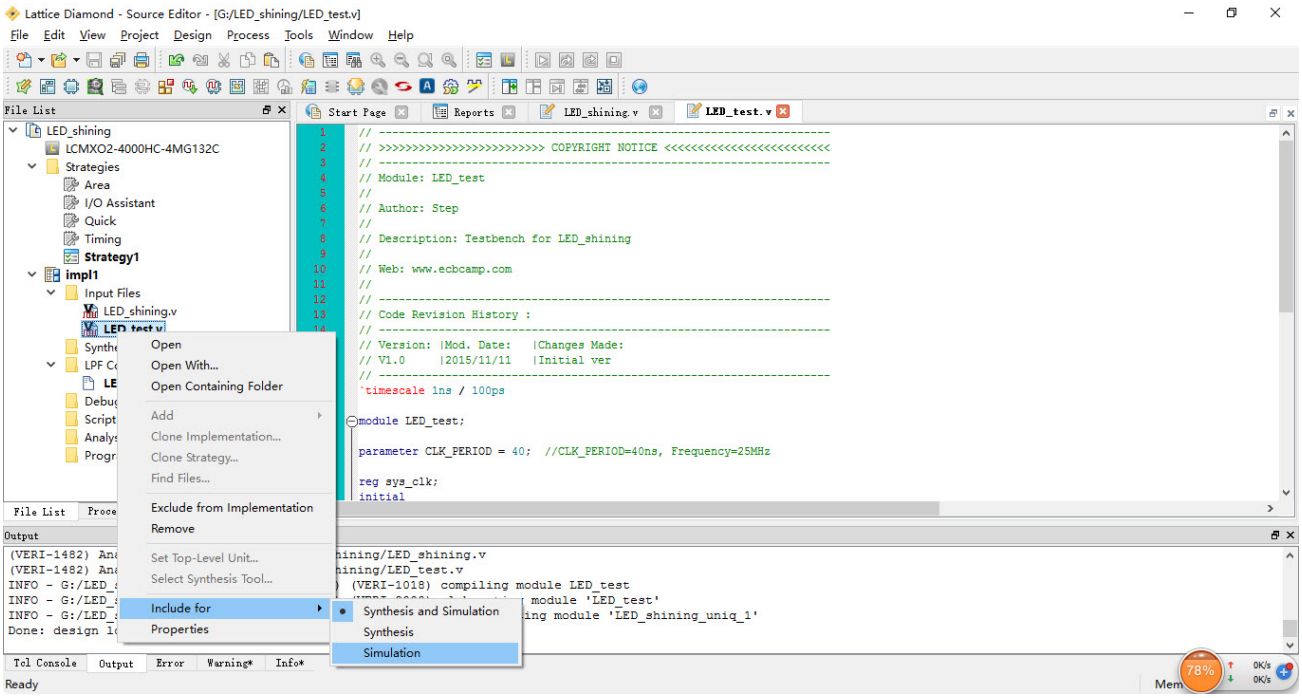
1. 首先我们添加testbench文件，和前面添加设计文件一样，File →New→File →Verilog Files，Name填写，然后New，



2. 测试源码如下，复制到LED*test.v文件并保存。为了方便仿真，我们在LED*test.v调用LED*shining模块时将CLK*DIV_PERIOD重新赋值为20：

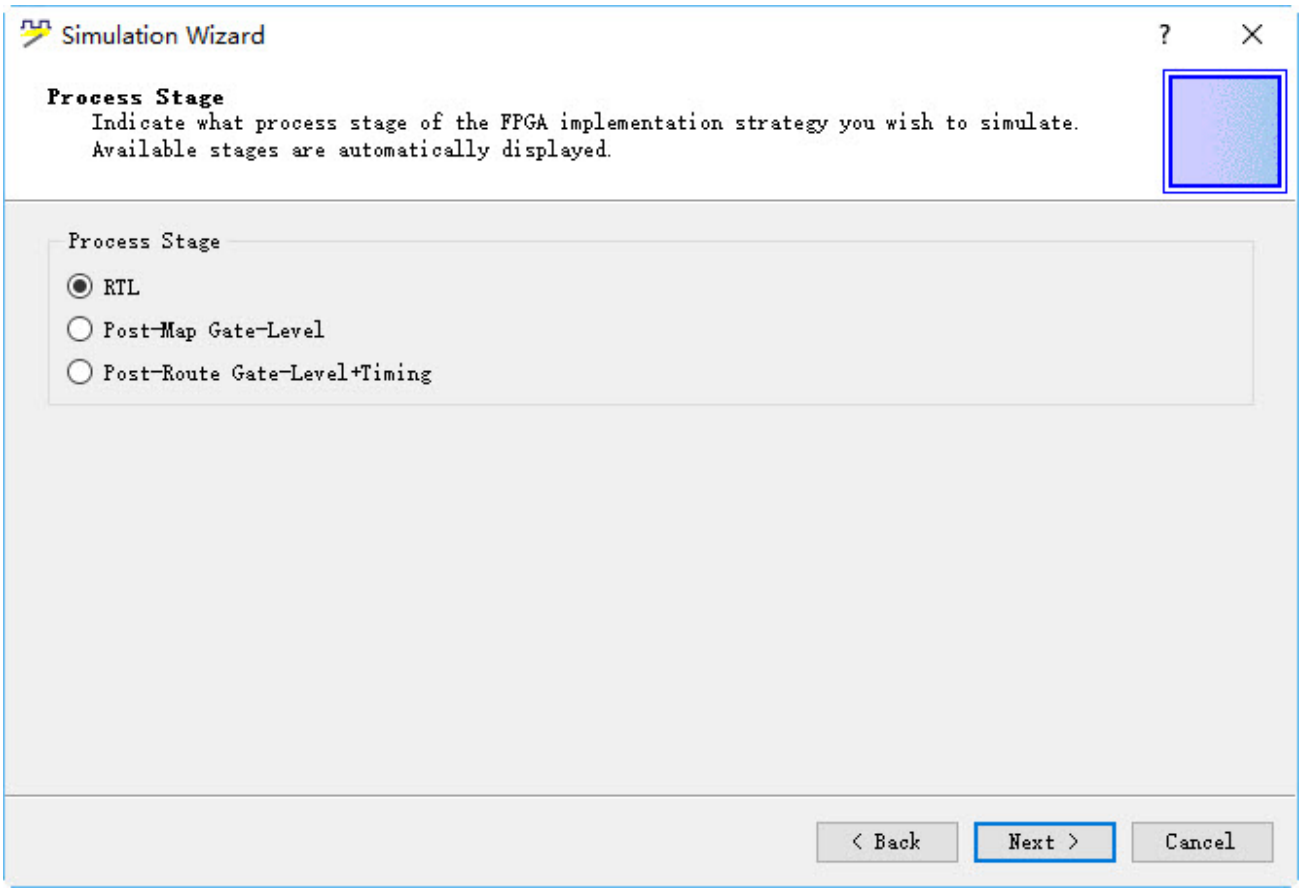
```
// ---------------------------------------------------------------------
// >>>>>>>>>>>>>>>>>>>>>>> COPYRIGHT NOTICE <<<<<<<<<<<<<<<<<<<<<<<
// ---------------------------------------------------------------------
// Module: LED_test
//
// Author: Step
//
// Description: Testbench for LED_shining
//
// Web: www.stepfpga.com
//
// ---------------------------------------------------------------------
// Code Revision History :
// ---------------------------------------------------------------------
// Version: |Mod. Date:   |Changes Made:
// V1.0     |2015/11/11   |Initial ver
// ---------------------------------------------------------------------
`timescale 1ns / 100ps

module LED_test;

parameter CLK_PERIOD = 40;
parameter CLK_DIV_PERIOD=20;

reg sys_clk;
initial
        sys_clk = 1'b0;
always
        sys_clk = #(CLK_PERIOD/2) ~sys_clk;        //产生周期为40ns的时钟激励，频率25MHz

reg sys_rst_n;
//产生一个初始100ns低电平然后变高电平的复位信号激励
initial
        begin
                sys_rst_n = 1'b0;
                #100;
                sys_rst_n = 1'b1;
        end

wire led1,led2;
//module例化
LED_shining #
(.CLK_DIV_PERIOD(CLK_DIV_PERIOD))
LED_shining_uut
(
.clk_in(sys_clk),       //传递时钟
.rst_n_in(sys_rst_n),   //传递复位信号
.led1(led1),            //传递输出led1
.led2(led2)             //传递输出led2
);
endmodule
```

1. 然后在软件左侧Process栏，选择File List，找到LED*test.v（必须保存过），点击右键，选择Include for →Simulation*

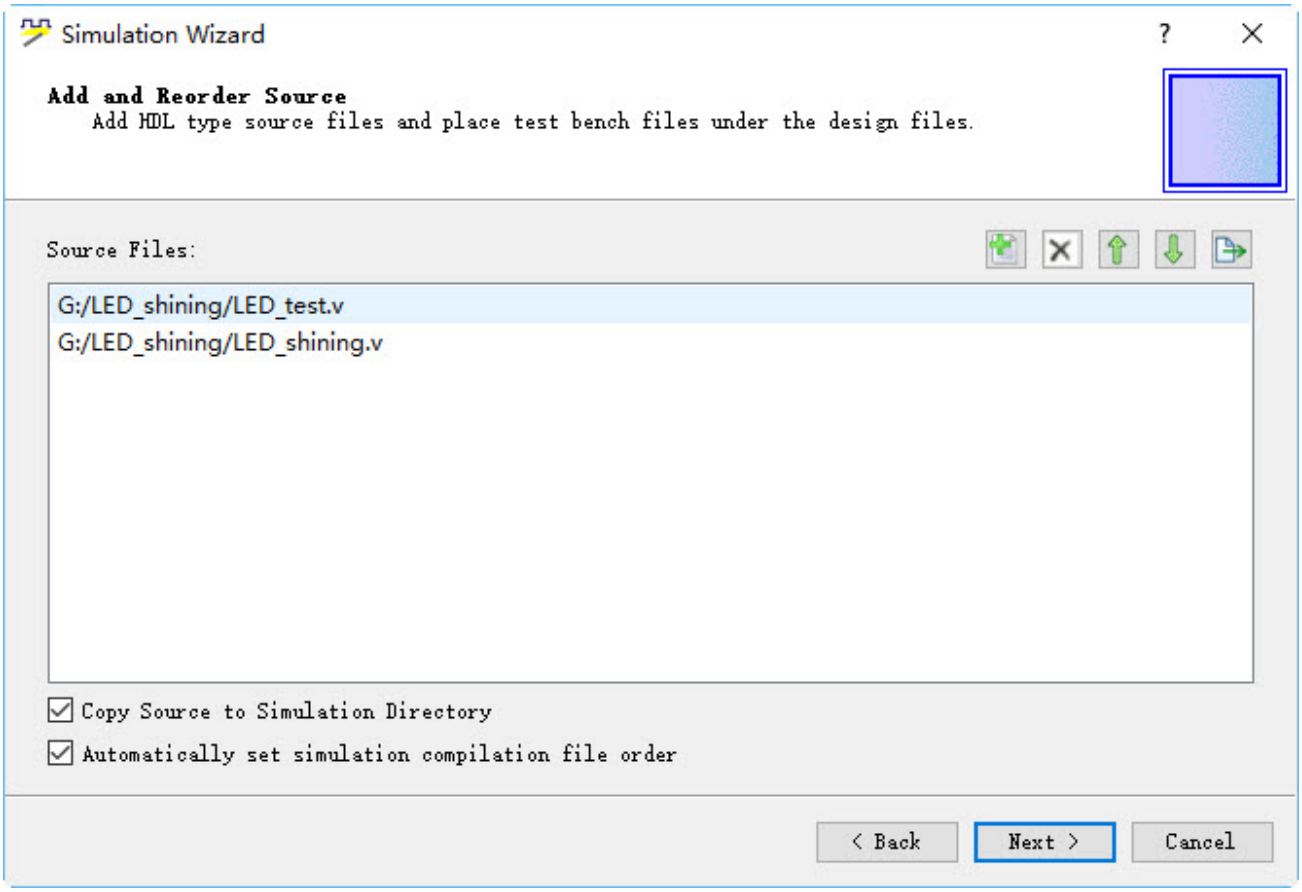


*- 准备工作完成，我们选择Tools →SimulationWizard →Next， - 建立仿真工程，ModelSim和QuestaSim需要自行安装并与Diamond关联，才能直接调用，这里我们选择Active-HDL（默认），工程名称：LED*test，工程路径默认即可：然后点击Next，

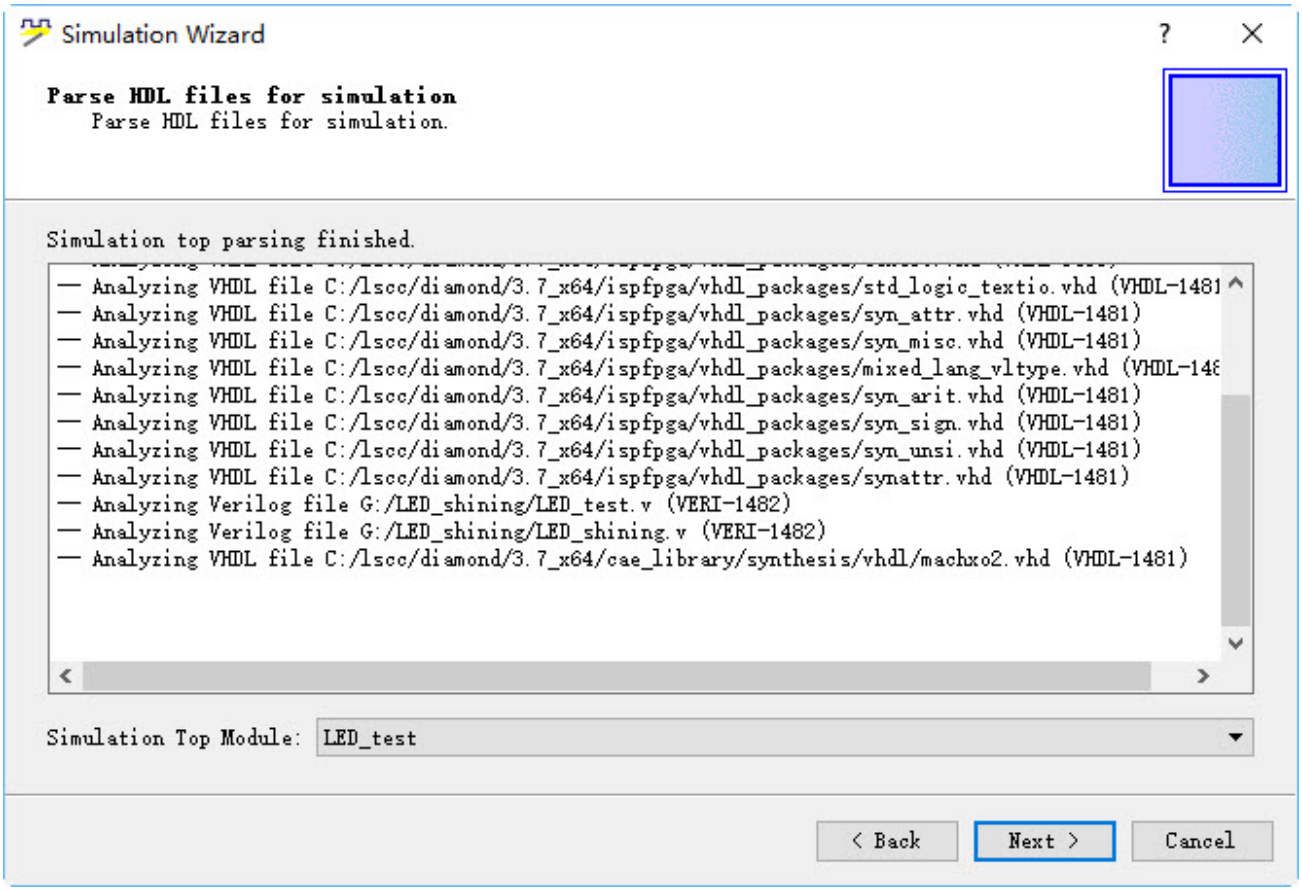
2. 选择RTL，然后Next



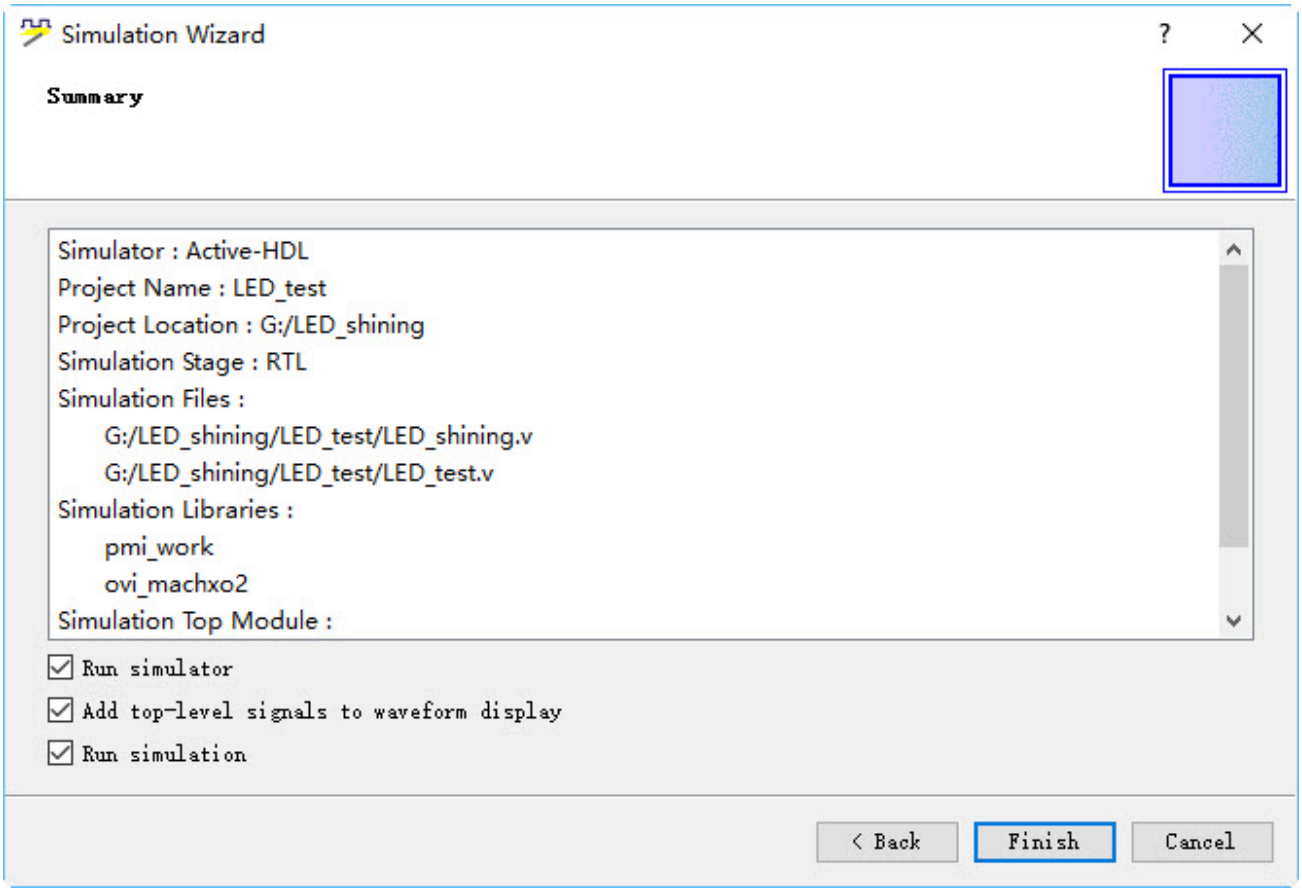
3. 勾选Copy Source toSimulation Directory，然后Next



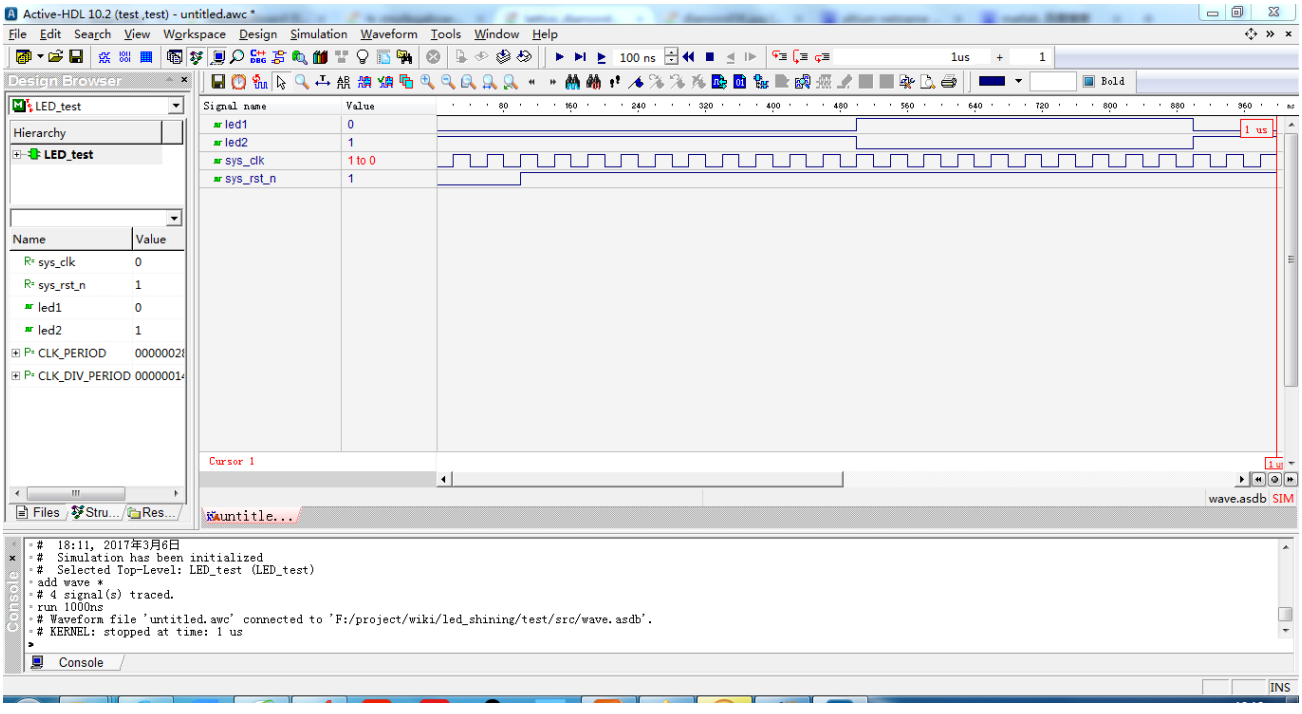
4. 点击Next



5. 点击Finish，等待仿真软件的自动运行并显示仿真时序



6. 查看仿真结果

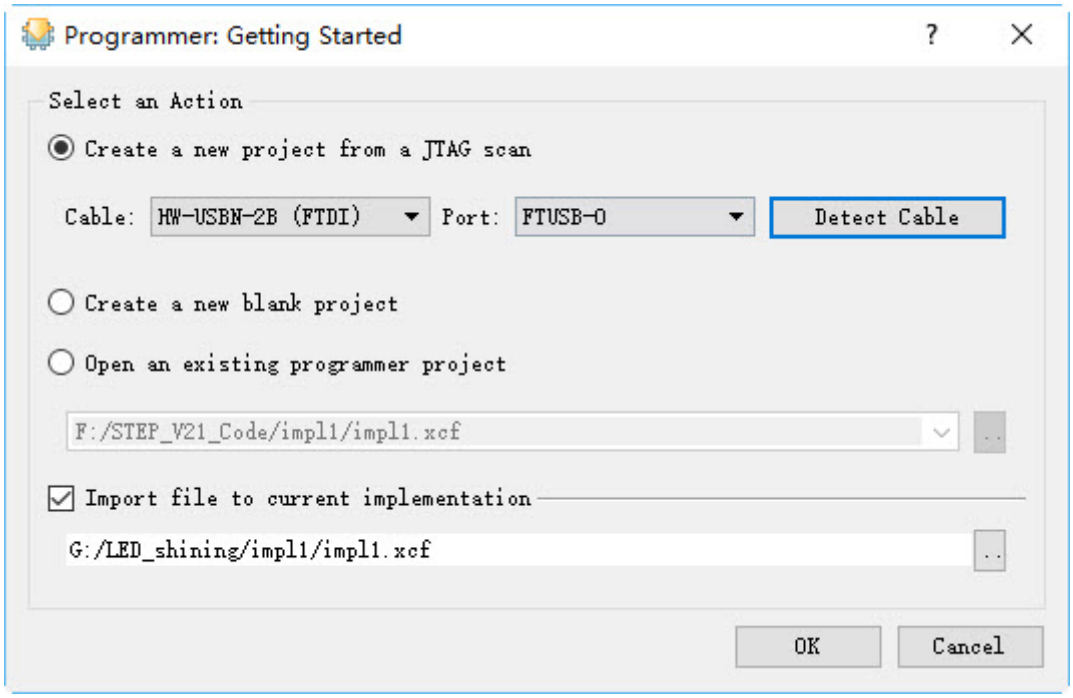


3 下载程序到FPGA

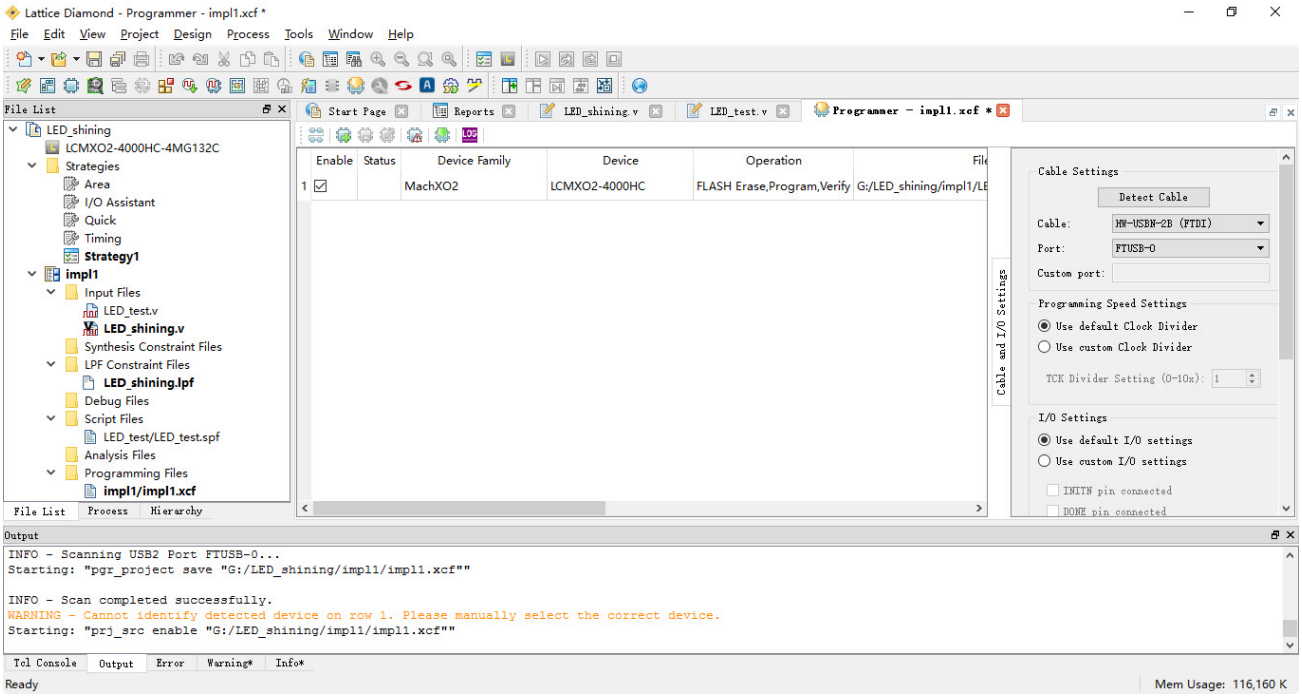
El chip de programación de STEP MXO2 V2 se ha integrado en la placa de desarrollo de pie pequeño, por lo que solo se necesita un cable Micro USB para conectarse a la computadora para completar la fuente de alimentación y las funciones de programación. Una vez instalado el controlador, puede comenzar a compilar y descargar el programa. Descargue el programa compilado en la placa de desarrollo:

1. Conecte la placa de desarrollo, el descargador y la computadora, como se muestra en la figura

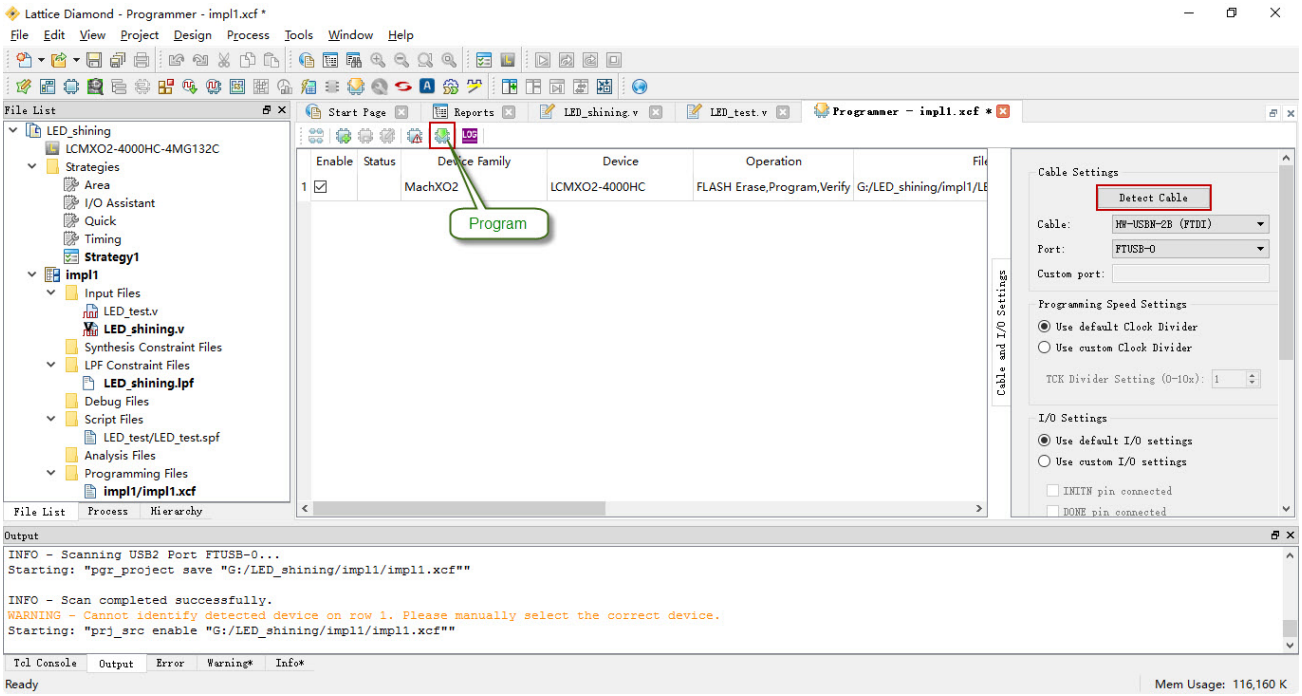
2. Seleccione Herramientas → Programador, seleccione el descargador HW-USBN-2B (FTDI) y haga clic en Aceptar,



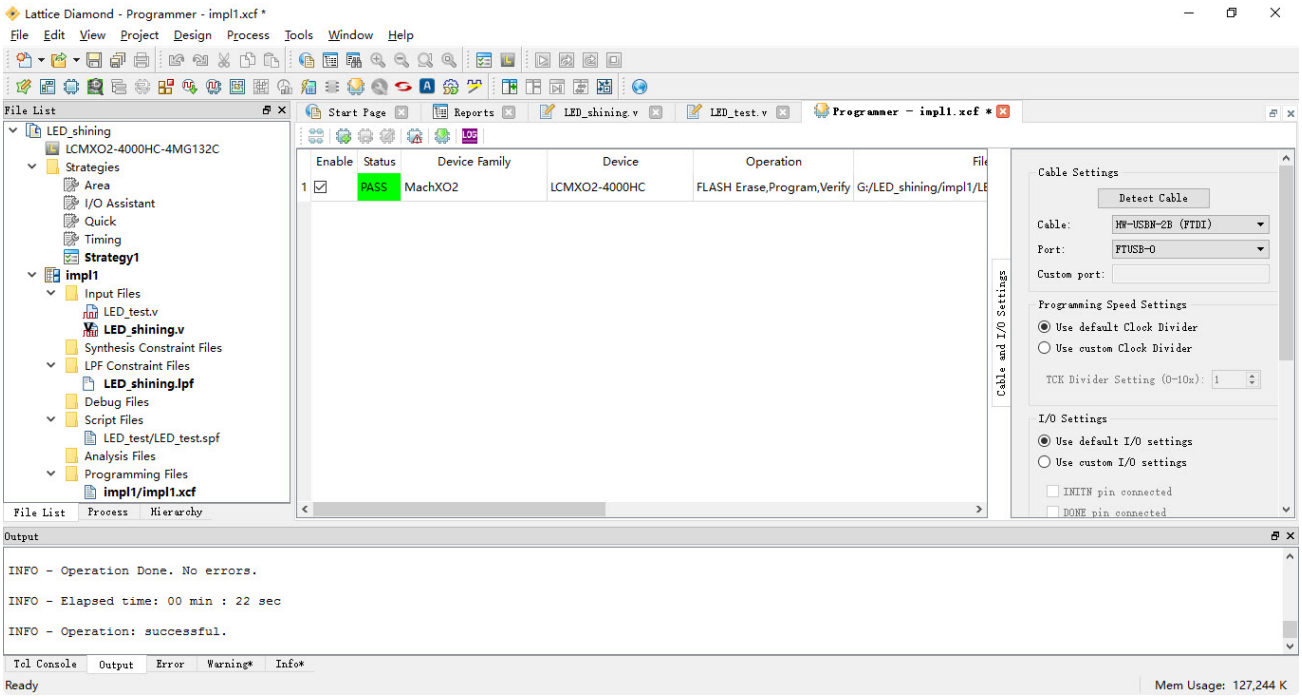
3. Entrar en la interfaz del programador



4. En la interfaz del programador, haga clic en Detectar cable a la derecha, detecte automáticamente el cable y muestre HW-USBN-2B (FTDI), luego haga clic en Programa en la figura siguiente



5. Se muestra PASS, la carga está completa y el LED de StepFPGA parpadea alternativamente.



==== 4 Introducción a STEP MXO2 ====

Aquí entendemos el proceso completo de desarrollo con el software Diamond. A continuación, comenzamos el tutorial introductorio STEP-MXO2 paso a paso en el diseño lógico programable.