

实验 1 基于 Quartus II，设计一个计算机系统

实验内容：

- 1) 系统硬件组成包含 Nios CPU、on_chip_ram 和 JTAG UART 三个模块；
- 2) 编写 C 语言程序 “hello_world_small”；
- 3) 调试运行。

实验目的：

- 1) 掌握 Quartus II 的基本使用方法；
- 2) 学会使用 Quartus II 来设计一个计算机系统；
- 3) 结合课本知识，进一步加深对计算机系统组成的认识。

成果要求：

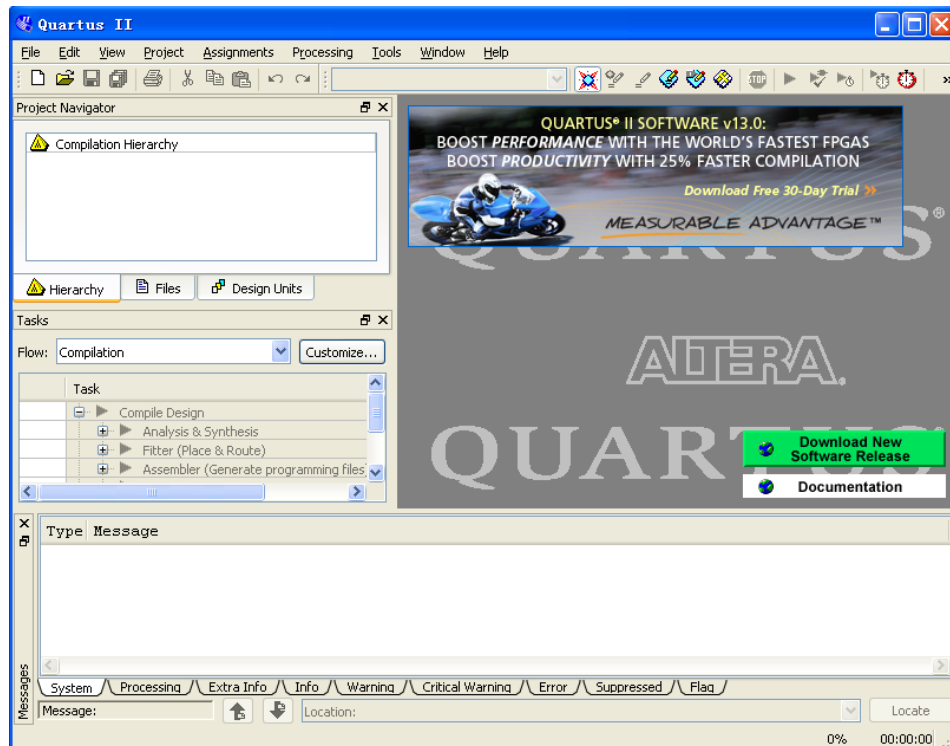
- 1) 报告设计过程、结果（原理图、仿真结果、源代码）、出现的典型问题及解决过程，要求结果体现个人设计思路，实验按照本文档应该能够很容易就完成，希望大家都能自己动手。
- 2) 实验报告请按照课程提供的“实验报告模版”来撰写。

实验环境

- 1) 下载安装 10.0_quartus_free_windows_rev2.exe
- 2) 下载安装 10.0_nios2eds_windows_rev2.exe
- 3) 如果在我们提供的 FTP 上下载的版本不能正确安装，请大家到 Alter 官方网站上下载，下载时需要注册用户，但是免费的。提示：使用迅雷下载会相当快！

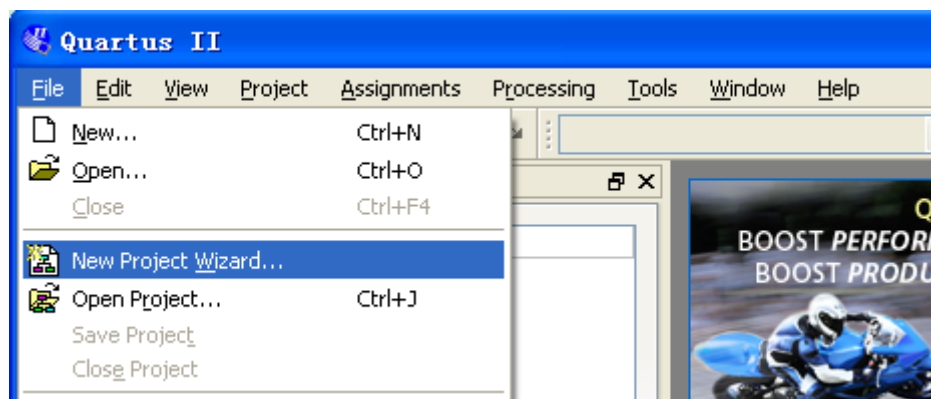
实验流程：

1. 运行 Quartus II 10.0(以下简称 Q2)，得到如下图所示的程序界面。

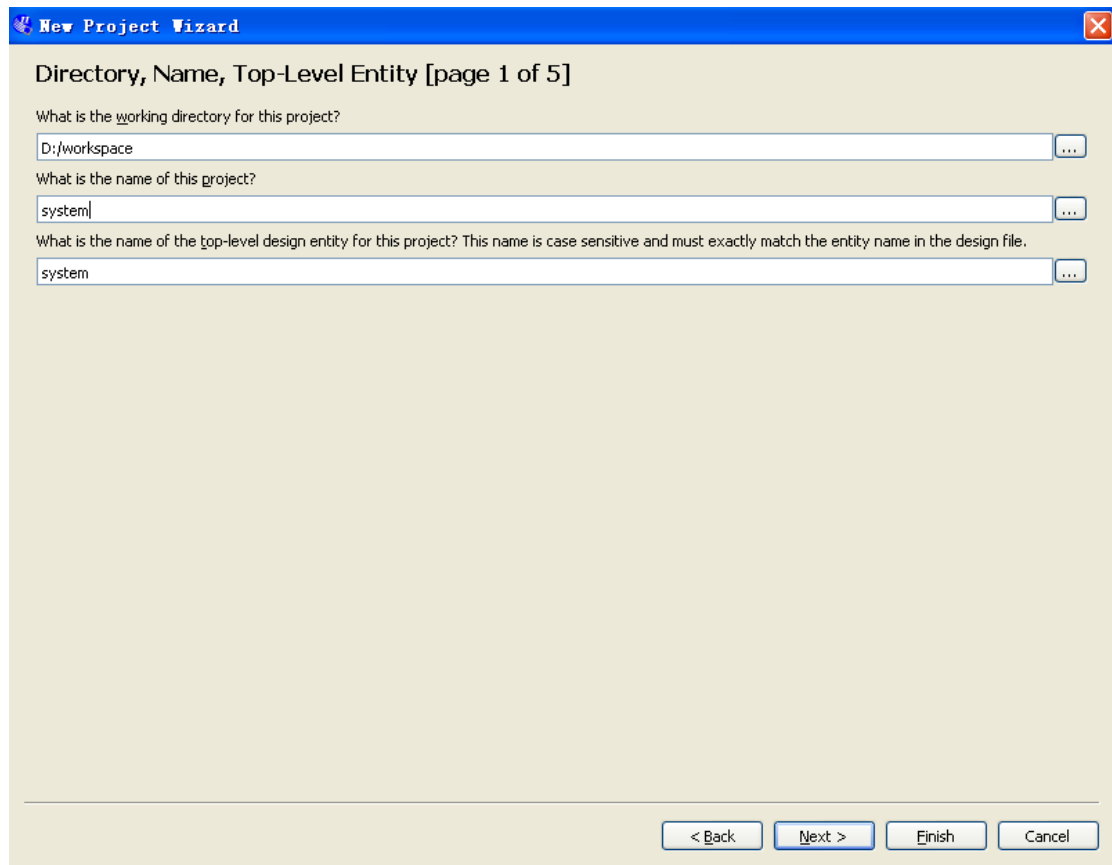


2. 新建工程

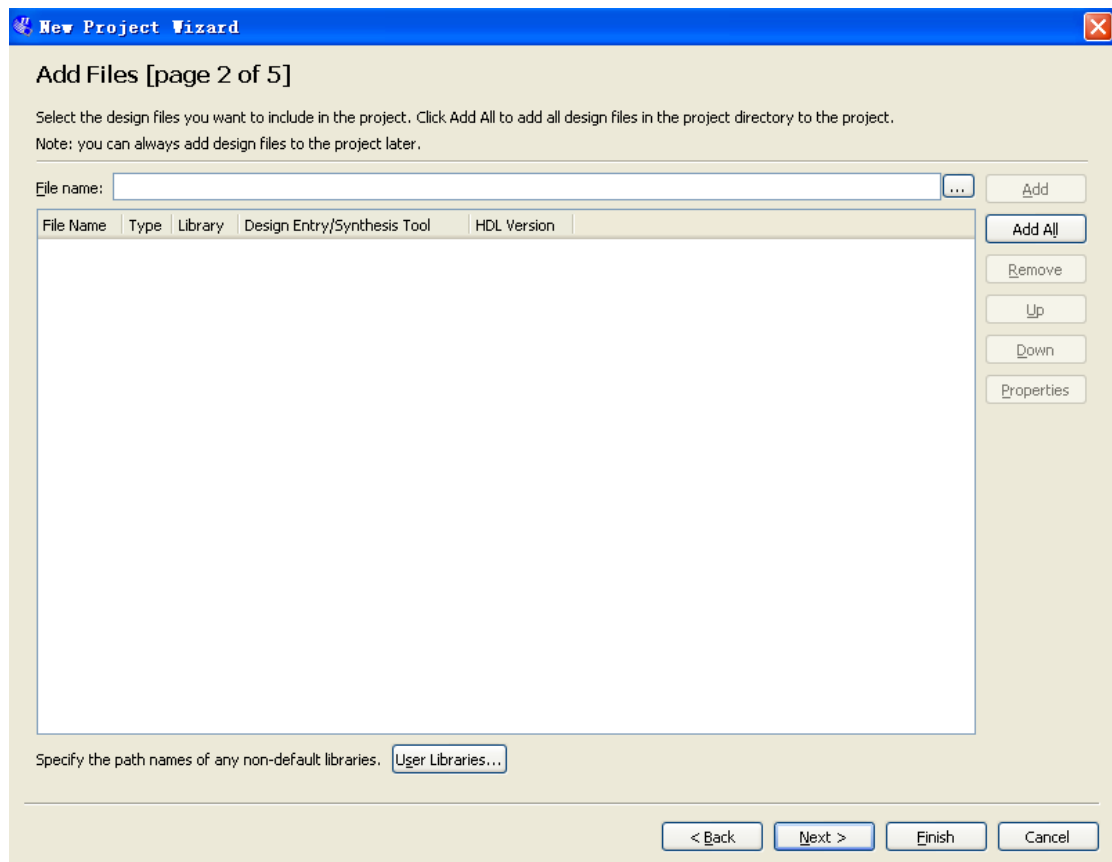
1) 运行 Q2，建立工程，File->New Project Wizard，如下图



2) 在弹出的对话框中点击 **Next**，出现如下图所示的对话框。选择工作目录 `D:/workspace`，也可以使用你自己设定的文件夹。工程必须有一个名字，通常情况下，与顶层设计实体的名字相同。如下图所示，选择 **system** 作为工程名和顶层实体名。单击 **Next**。如果工作目录(`D:/workspace`)没有创建，则 Quartus II 会弹出一个对话框，询问是否新建所需的文件夹，选择 **Yes**。因为我已经创建好 `D:/workspace` 了，所以没有出现该对话框。



3) 完成上图中工作目录等的填写后，单击 **Next**，得到如下图所示的对话框。



4) 如果没有已经存在的设计文件，点击 **Next**，则打开如下图所示的对话框。

New Project Wizard

Family & Device Settings [page 3 of 5]

Select the family and device you want to target for compilation.

Device family

Family: Stratix II

Devices: All

Target device

☒ Auto device selected by the Filter

☐ Specific device selected in 'Available devices' list

☐ Other: n/a

Show in 'Available devices' list

Package: Any

Pin count: Any

Speed grade: Any

☒ Show advanced devices

☐ HardCopy compatible only

Available devices:

Name	Core Voltage	ALUTs	User I/Os	Memory Bits	DSP	PLL	DLL	Global Clocks
EP2515F484C3	1.2V	12480	343	419328	12	6	2	16
EP2515F484C4	1.2V	12480	343	419328	12	6	2	16
EP2515F484C5	1.2V	12480	343	419328	12	6	2	16
EP2515F484I4	1.2V	12480	343	419328	12	6	2	16
EP2515F672C3	1.2V	12480	367	419328	12	6	2	16
EP2515F672C4	1.2V	12480	367	419328	12	6	2	16
EP2515F672C5	1.2V	12480	367	419328	12	6	2	16
EP2515F672I4	1.2V	12480	367	419328	12	6	2	16

Companion device

HardCopy:

☐ Limit DSP & RAM to HardCopy device resources

< Back
Next >
Finish
Cancel

5) 点击 Next，最终得到如下图。

New Project Wizard

Summary [page 5 of 5]

When you click Finish, the project will be created with the following settings:

Project directory:

D:/workspace

Project name:

system

Top-level design entity:

system

Number of files added:

0

Number of user libraries added:

0

Device assignments:

Family name:

Stratix II

Device:

AUTO

EDA tools:

Design entry/synthesis:

<None> (<None>)

Simulation:

<None> (<None>)

Timing analysis:

<None> (<None>)

Operating conditions:

Core voltage:

n/a

Junction temperature range:

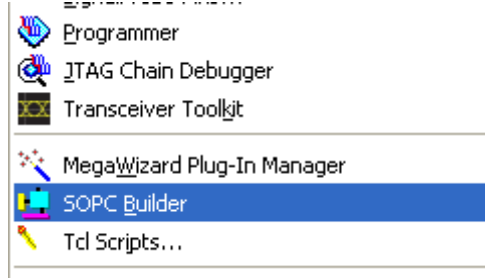
n/a

< Back
Next >
Finish
Cancel

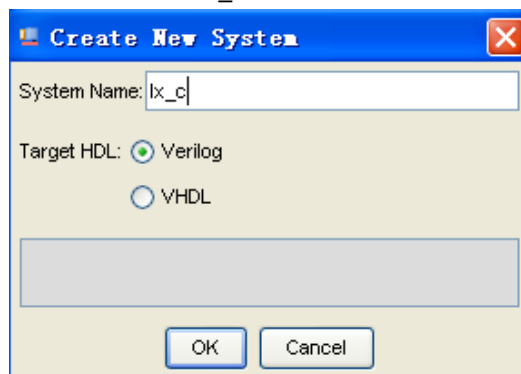
6) 点击 Finish，完成工程的创建。

其中需要注意的是，由于我们仅仅设计供仿真调试的系统，则在如上的第 4) 步骤中 fpga 器件可以选择为 Auto device selected by the Filter.

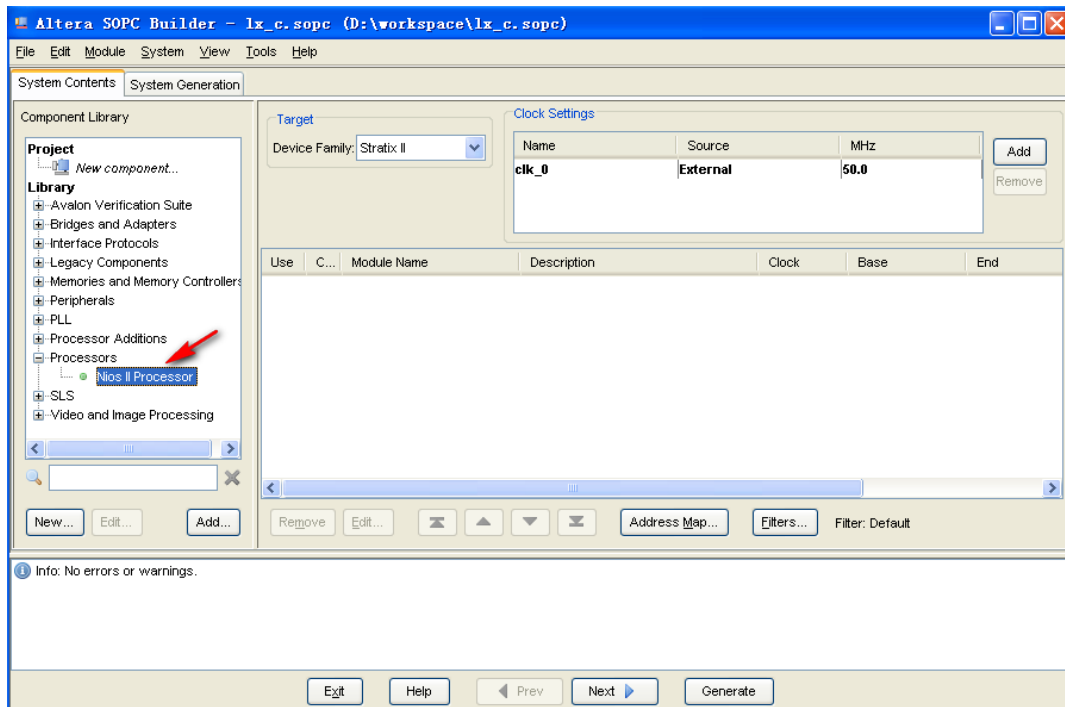
3. 用 SOPC Builder 定制 Nios II 处理器及其外设。打开 Tools->SOPC Builder，



要求指定系统名字，本例中我们输入 lx_c，



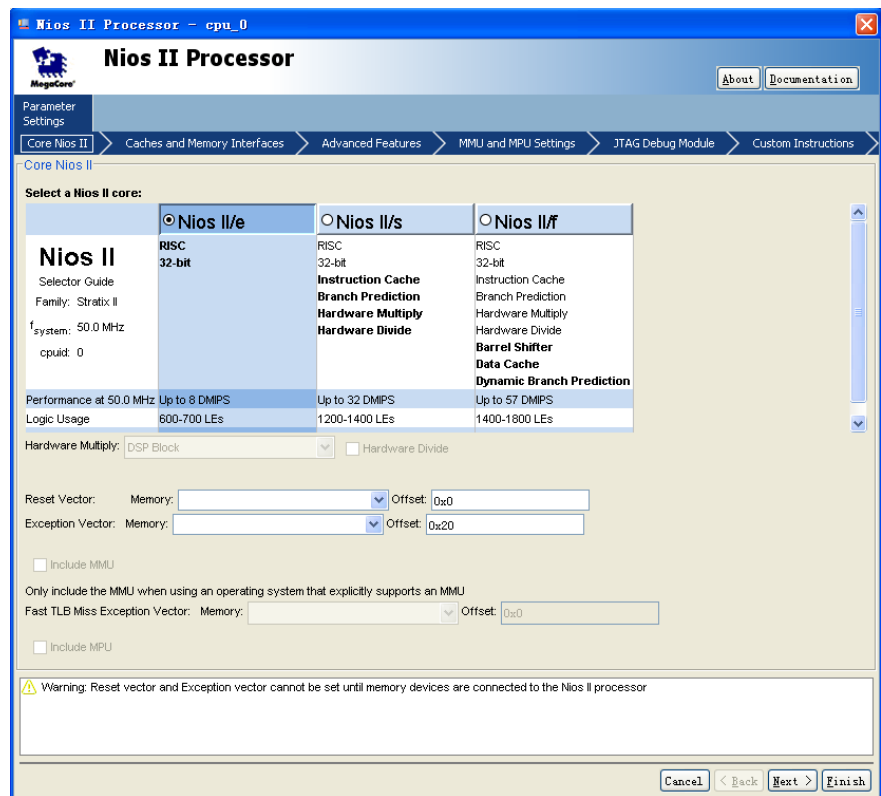
点击 OK，进入 SOPC 定制界面。



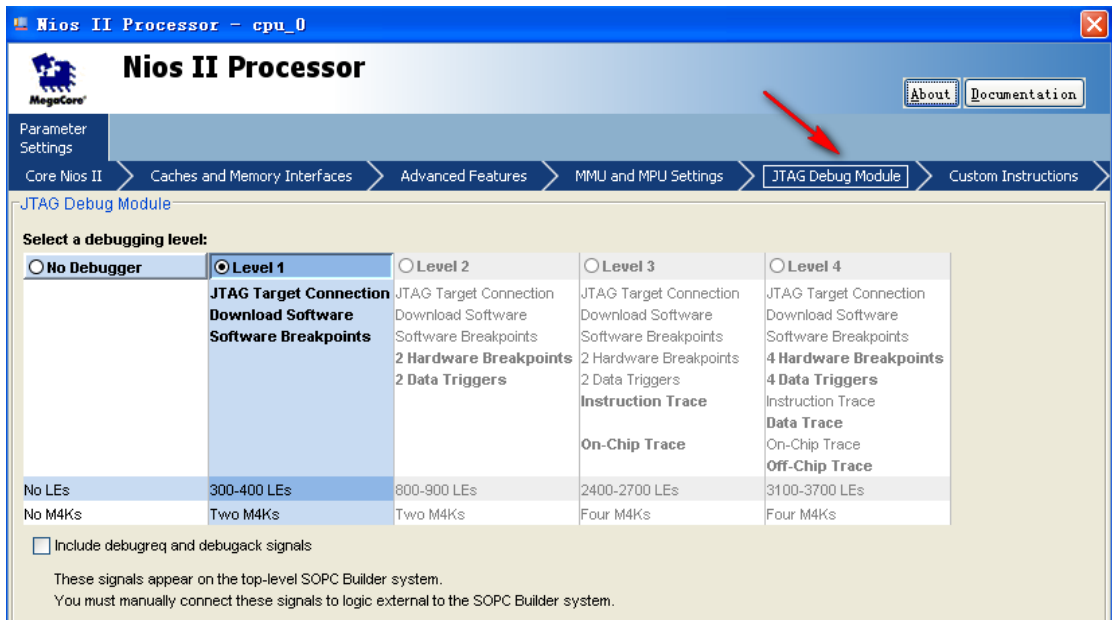
在 SOPC 定制界面的左边，我们可以看到有很多功能模块，这些功能模块，用户可以按照需要添加到所设计的系统中。

首先，我们需要一个 CPU，在上图左边的选项框中，展开 Processors 选项，然后左键双

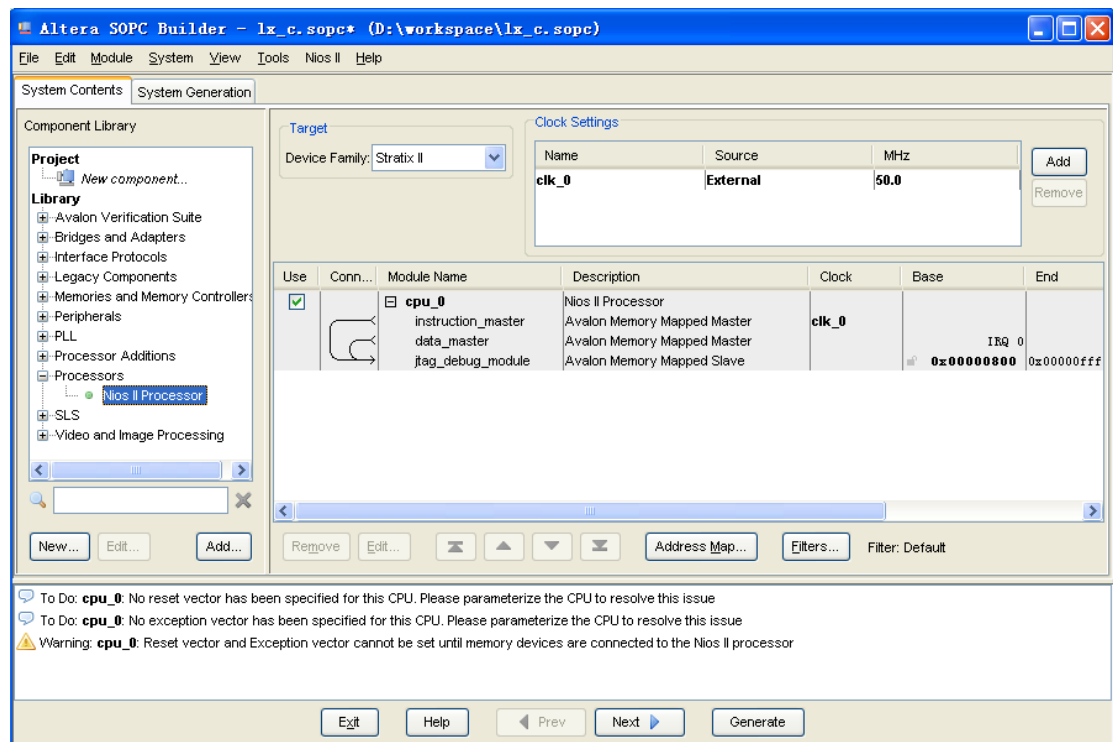
击 Nios II Processor，弹出 Nios II Processor 对话框，我们选择一个经济型的 CPU 核，即 Nios II/e，如下图所示：



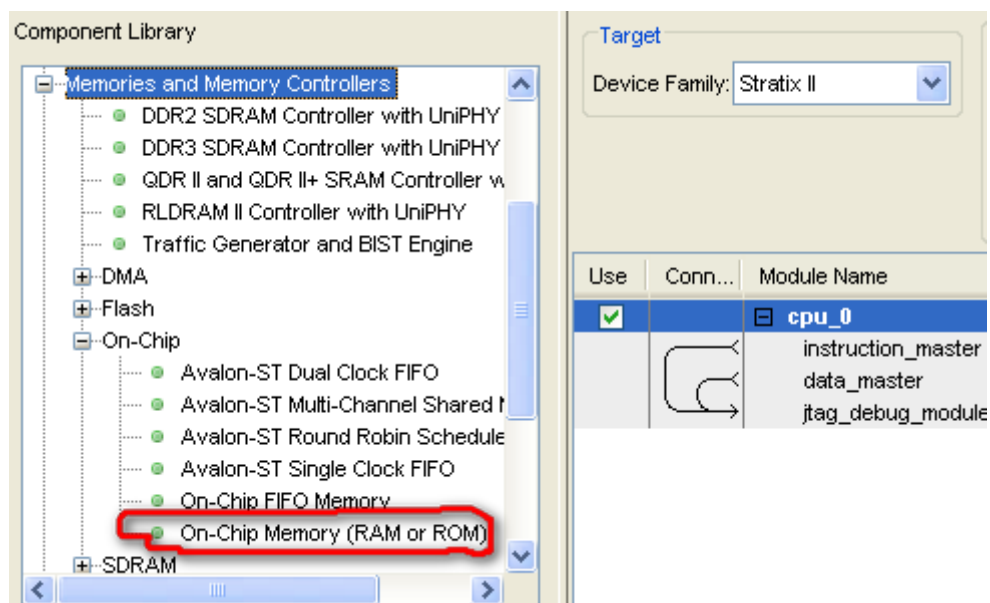
点击 JTAG Debug Module 标签页，选择第一级调试支持 Level 1:



点击 Finish 完成 Nios II CPU 的配置工作。项目中会增加一个 Nios II 处理器，名字为 cpu_0，为了简便起见，没有将它改名。改名的方法是：右键->ReName，输入名字后回车。如下图：



将上图中左边选项框内的 Memories and Memory Controllers 展开，如下图：



左键双击 On-Chip Memory (RAM or ROM)，为系统添加 RAM。Memory Type 选择 RAM；Data Width 选择 32bits，Total Memory Size 可以选择为 4K bytes，如下图所示：

On-Chip Memory (RAM or ROM) - onchip_memory2_0

On-Chip Memory (RAM or ROM)

Parameter Settings

General settings > Memory initialization

Memory type

☒ RAM (Writable) ☐ ROM (Read-only)

☐ Dual-port access

Read During Write Mode: DONT_CARE

Block type: Auto

☒ Initialize memory content

Selecting Auto block type will not infer M512s.
Memory will be initialized from onchip_memory2_0.hex

Size

Data width: 32

Total memory size: 4 KBytes

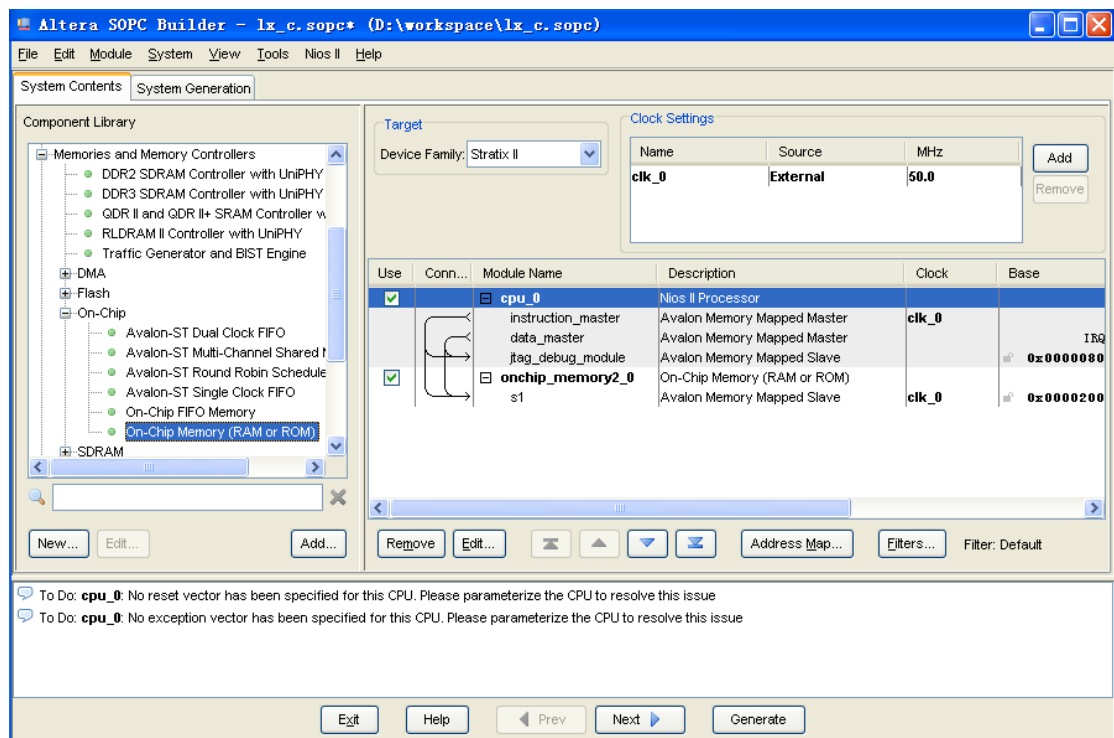
☐ Minimize memory block usage (may impact fmax)

Read latency

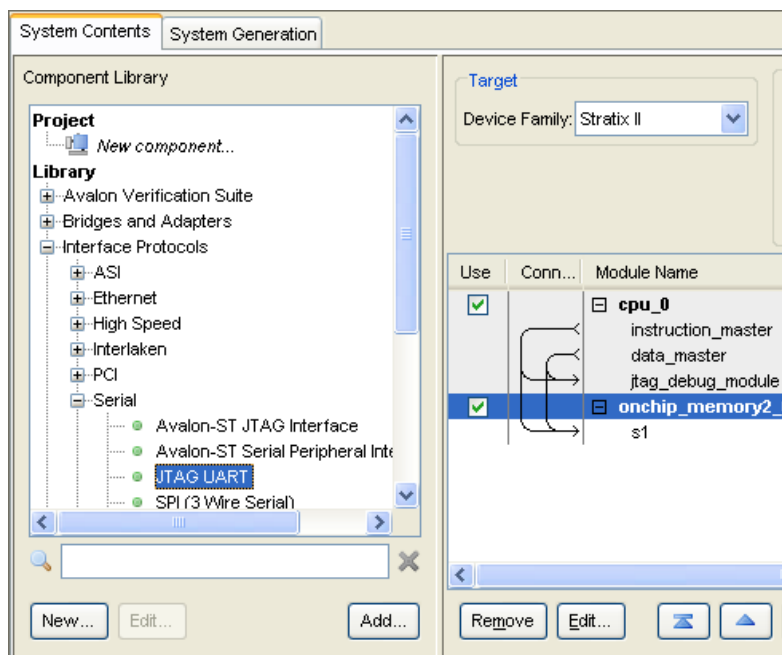
Slave s1: 1 Slave s2: 1

Cancel < Back Next > Finish

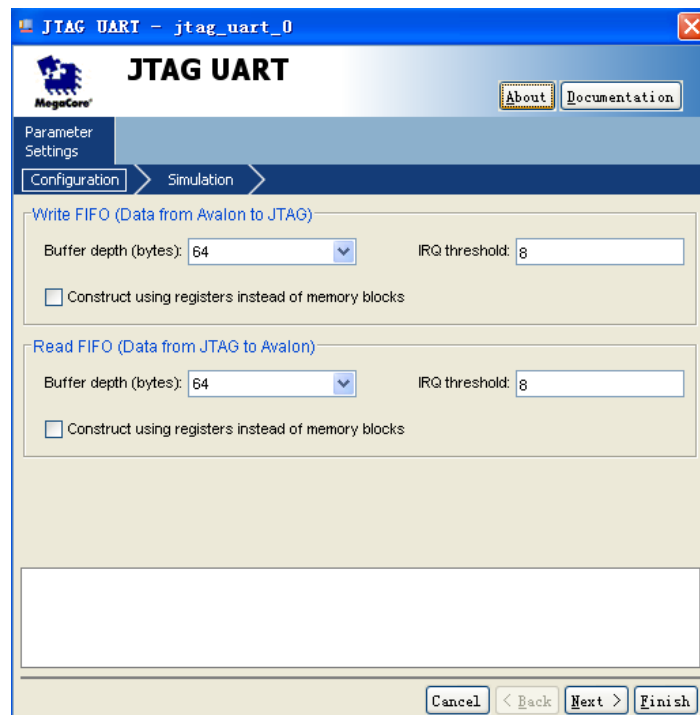
点击 Finish 确认，返回 SOPC Builder 界面：



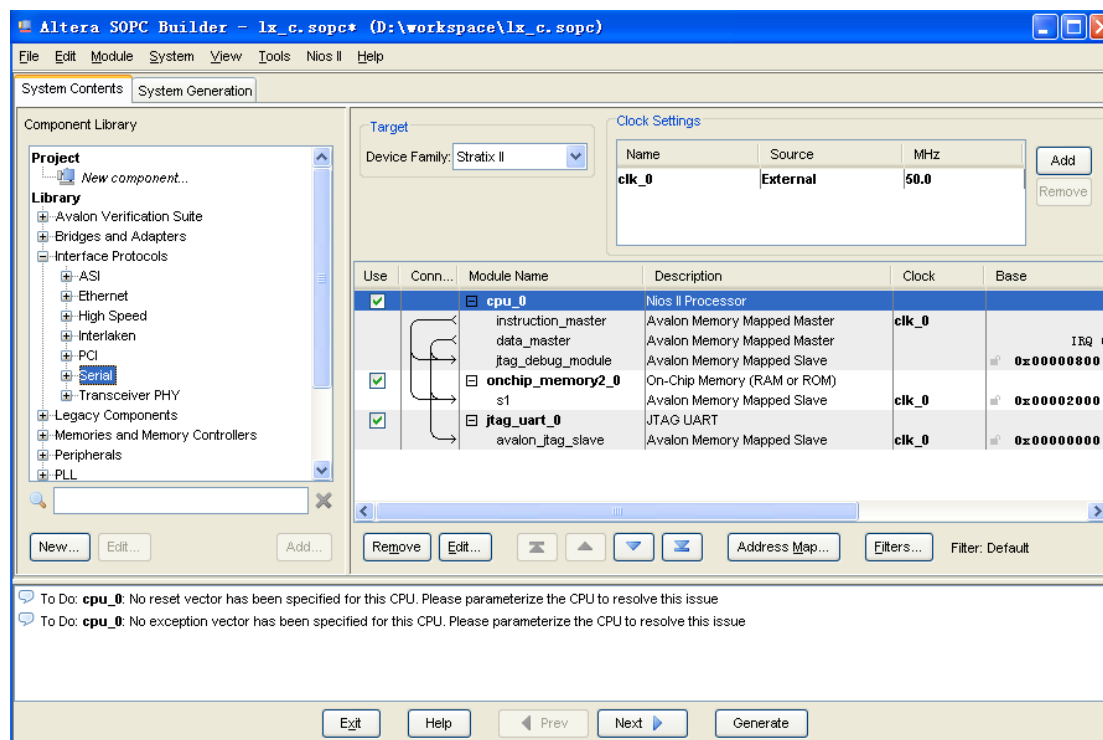
将上图左边选项框中的 Interface Protocols 展开，再展开 Serial，选中 JTAG UART，如下图所示：



左键双击 JTAG UART，在弹出的对话框中做如下图所示的配置：



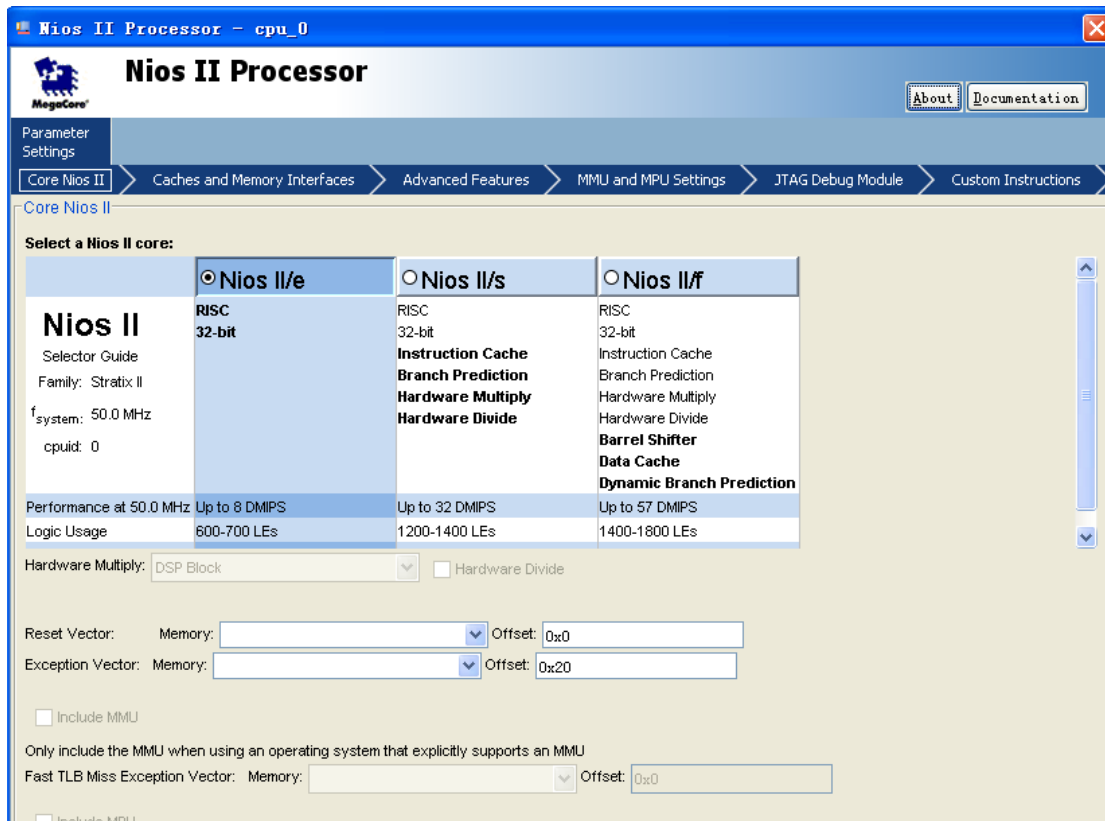
然后点击 Finish，返回 SOPC Builder 界面：



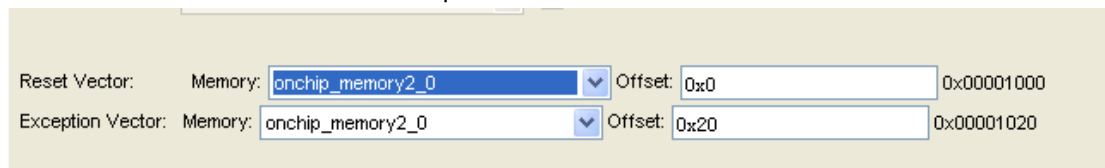
然后左键双击 `cpu_0` 选项条，如下图：

Use	Conn...	Module Name	Description	Clock	Base
<input checked="" type="checkbox"/>		cpu_0	Nios II Processor		
<input checked="" type="checkbox"/>		instruction_master	Avalon Memory Mapped Master	clk_0	

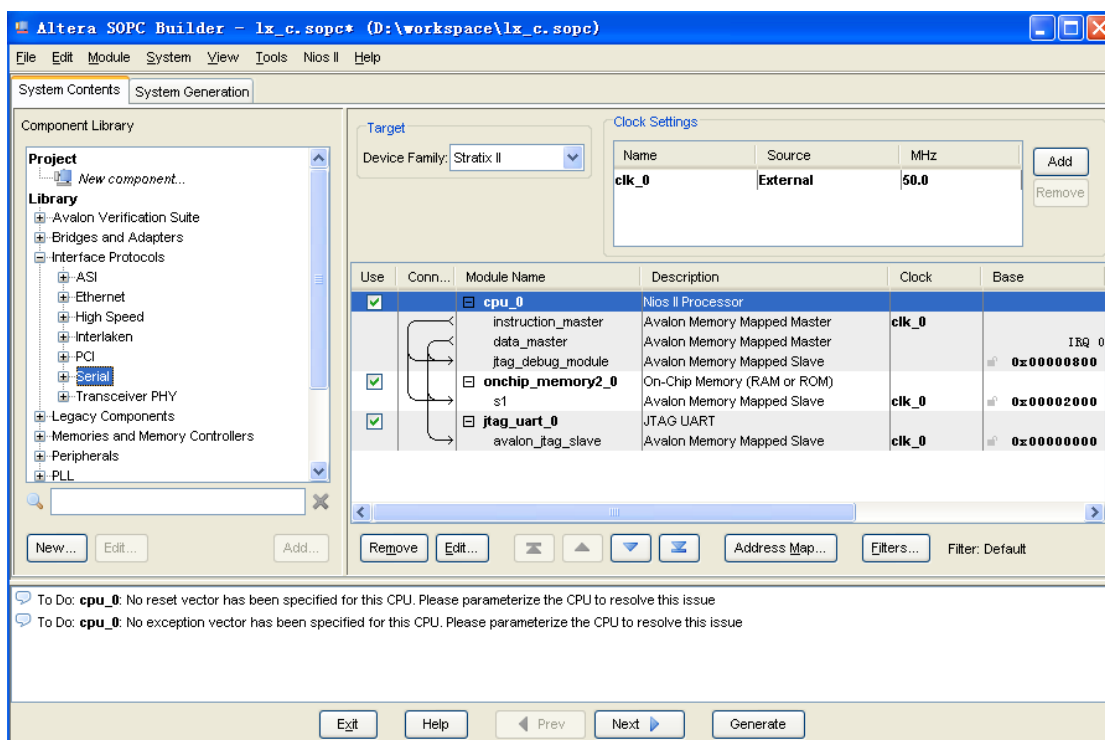
得到 Nios II Processor 的配置界面，如下图：



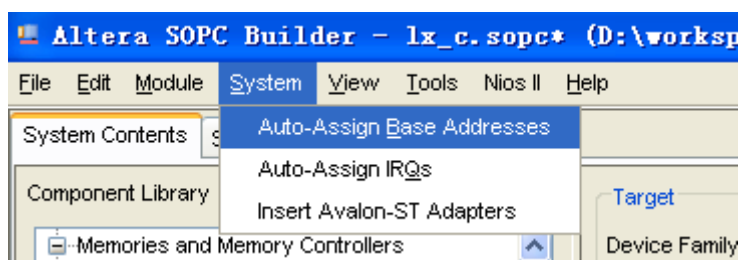
对上图中的 Reset Vector 和 Exception Vector 进行如下图所示的配置:



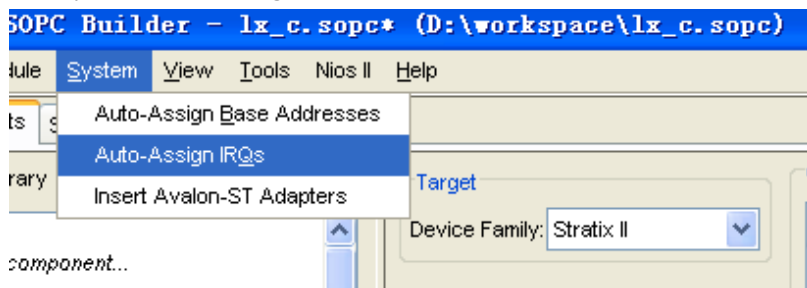
点击 Finish 确认, 返回 SOPC Builder 界面。



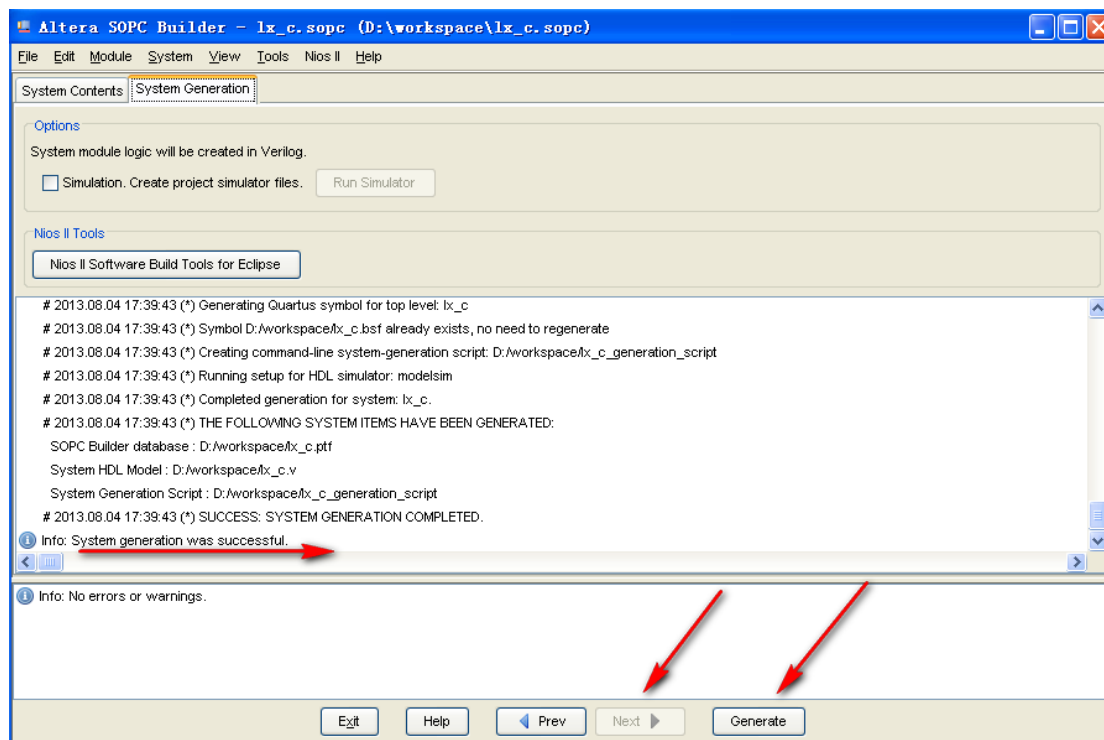
然后，选择 System->Auto-Assign Base Addresses，让系统自动分配基地址，如下图：



同理，选中 System->Auto-Assign IRQs，让系统自动分配中断，如下图：

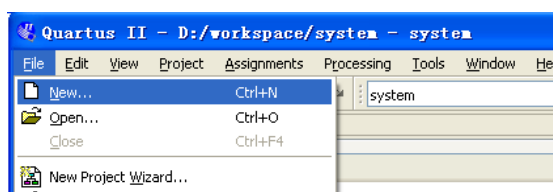


做完如上的配置后，点击 SOPC Builder 界面中的 Next，然后点击 Generate，并在弹出的对话框中选择 Save，生成系统。如果成功创建，则如下图所示：

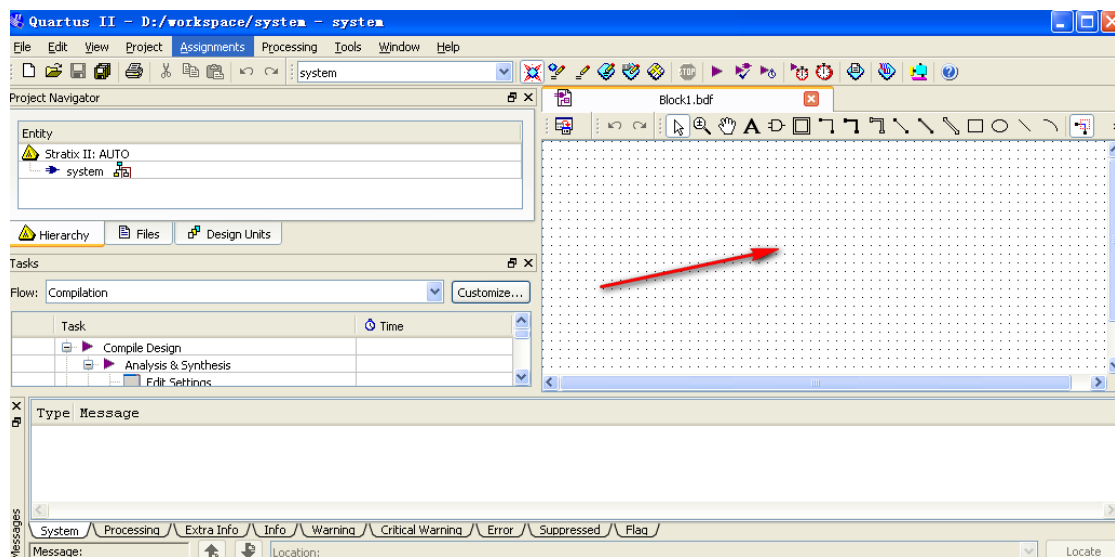


点击 Exit 退出 SOPC Builder.

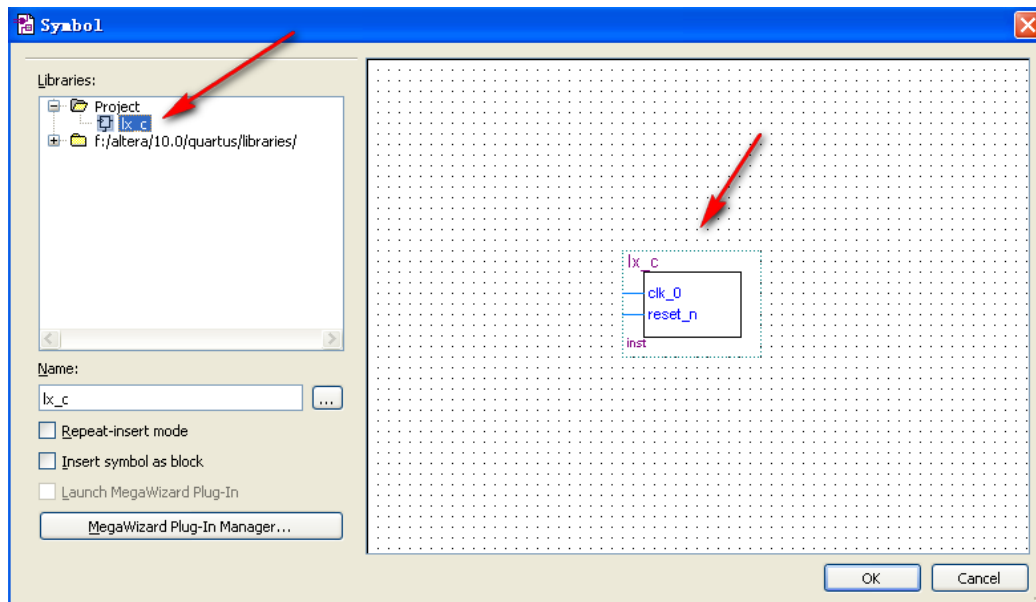
4. 在 Quartus II 工程中添加上述 Nios II 系统。
选中 Quartus II 的 File->New，如下图所示：



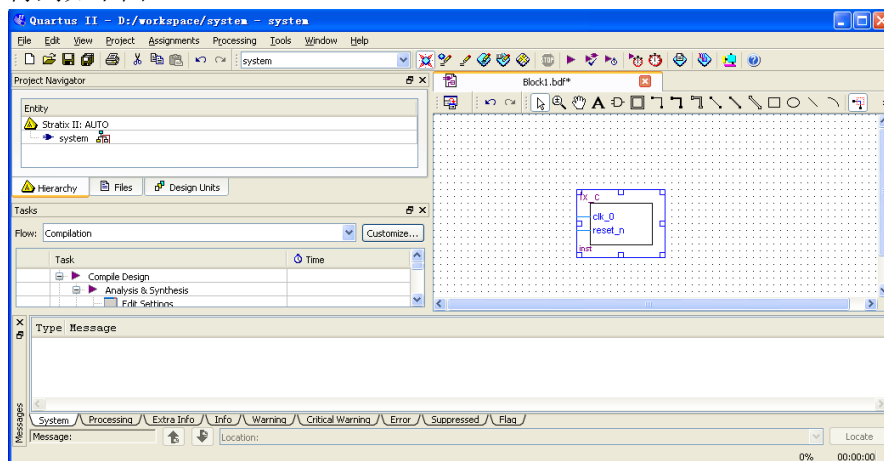
在弹出的对话框中，点击 Block Diagram/Schematic File，得到如下图：



在上图箭头所指的空白区域左键双击，得到如下图所示的对话框：

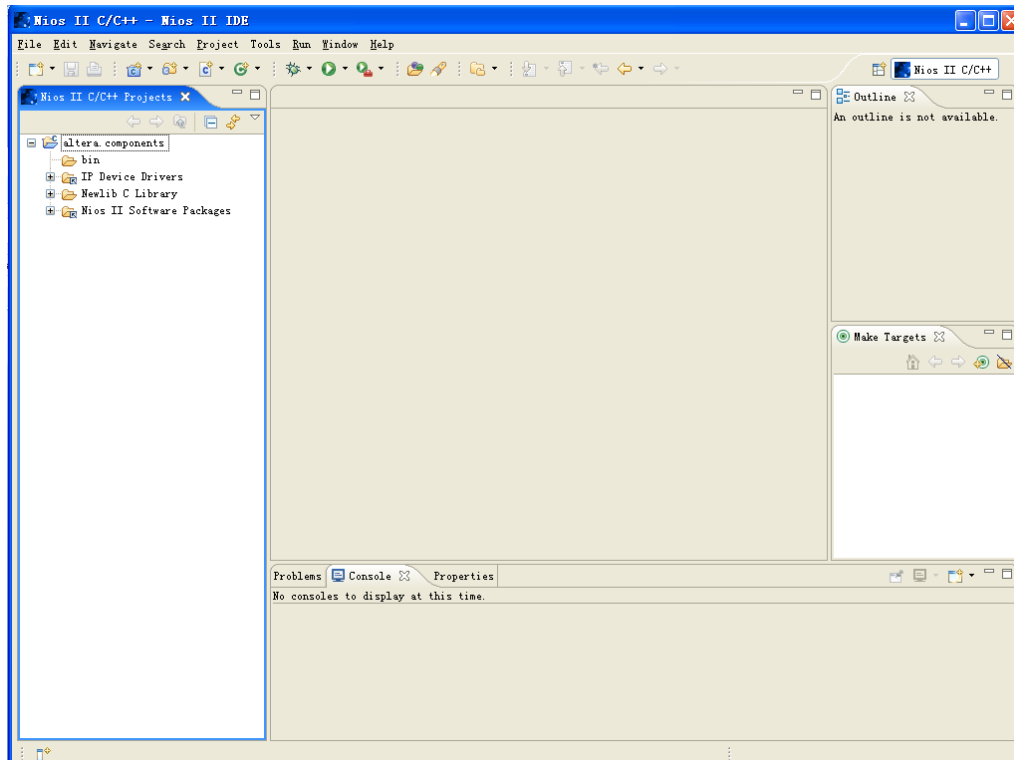


将上图中的 Project 展开，选中 lx_c，然后点击 OK，将 lx_c 拖放到刚才双击左键的空白区域，如得到如下图：

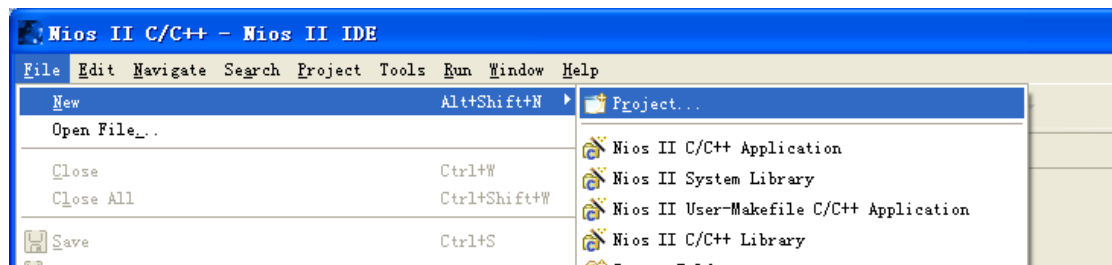


Ctrl+S 保存文件，文件名和顶层模块名相同，为 system.bdf，并保存 system.bdf 到工程目录 D:\workspace 下。

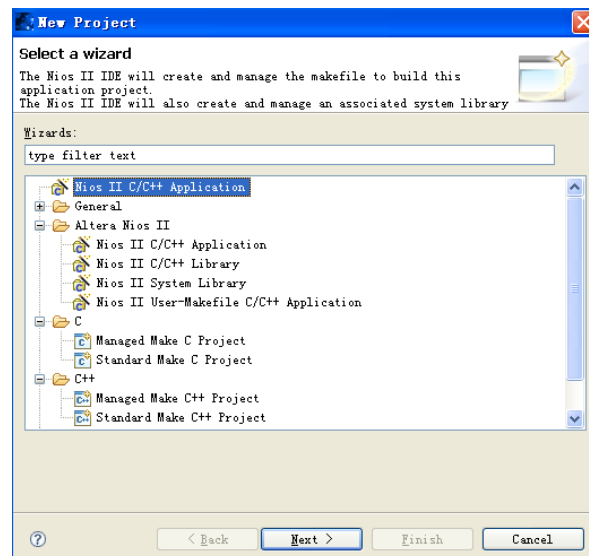
运行 Nios II IDE 10.0，如下图：



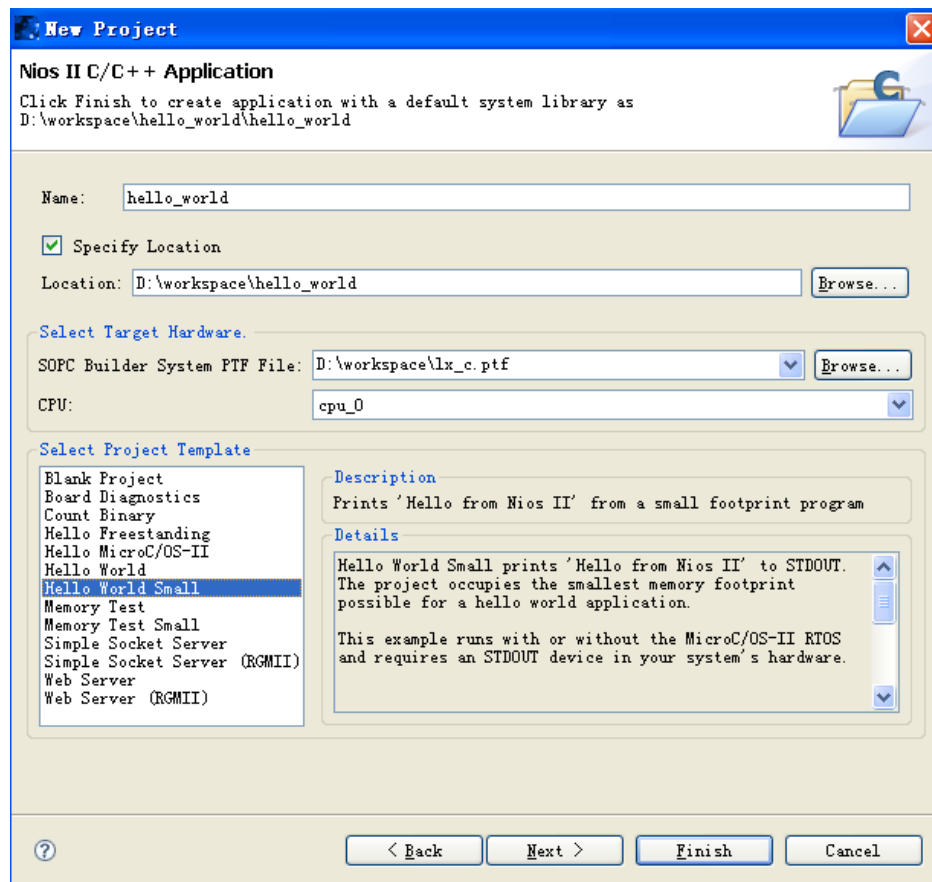
点击 File->New->Project，如下图所示：



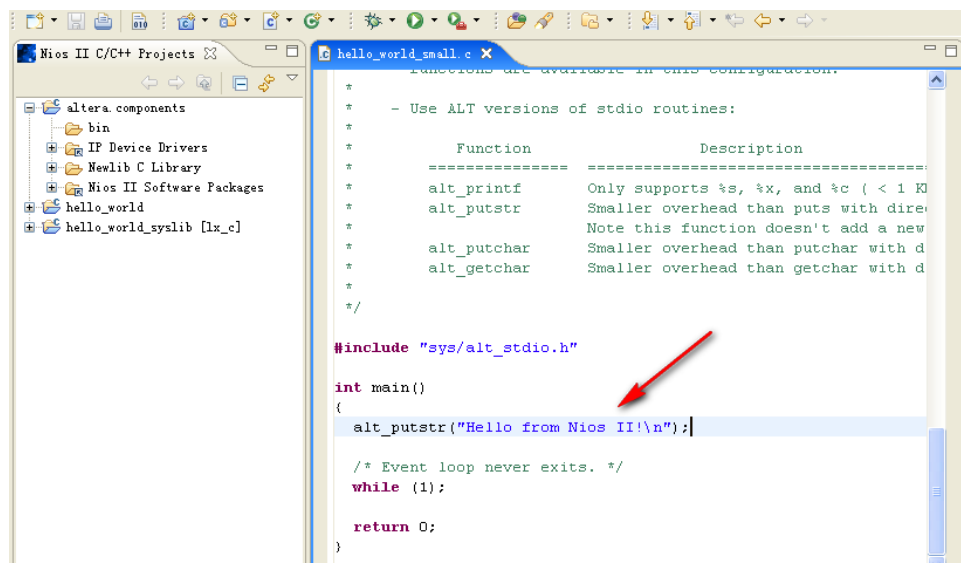
在弹出的对话框中选择 Nios II C/C++ Application，然后点击 Next，如下图：



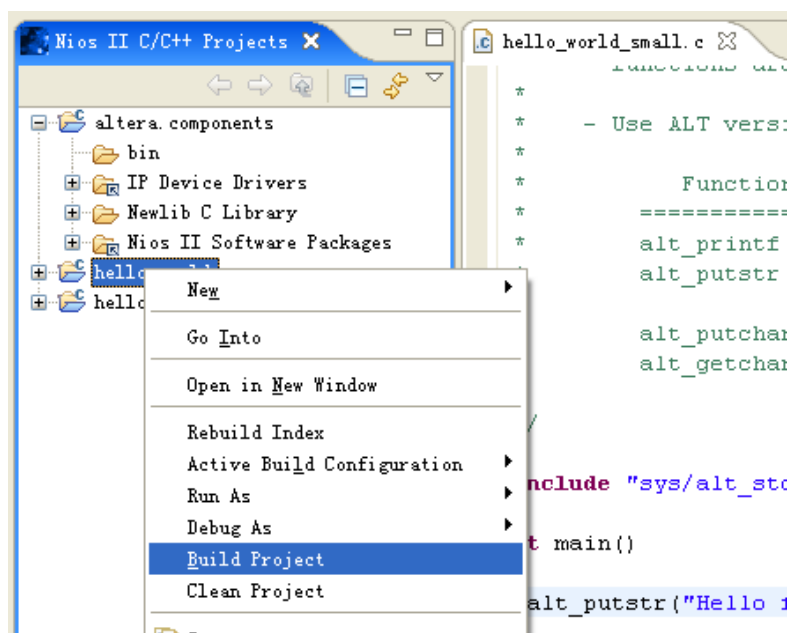
点击 Next 后，弹出如下图所示的对话框，其中将该文件取名 `hello_world`，并将该文件保存到 Quartus II 刚才创建的工程所在的目录中，也就是 `D:\workspace`，同时在 `D:\workspace` 目录中找到刚才我们用 Quartus II 创建的 `lx_c.ptf` 文件，也就是指定该 c 程序的目标 cpu，并选中 Hello World Small 模版，如下图所示：



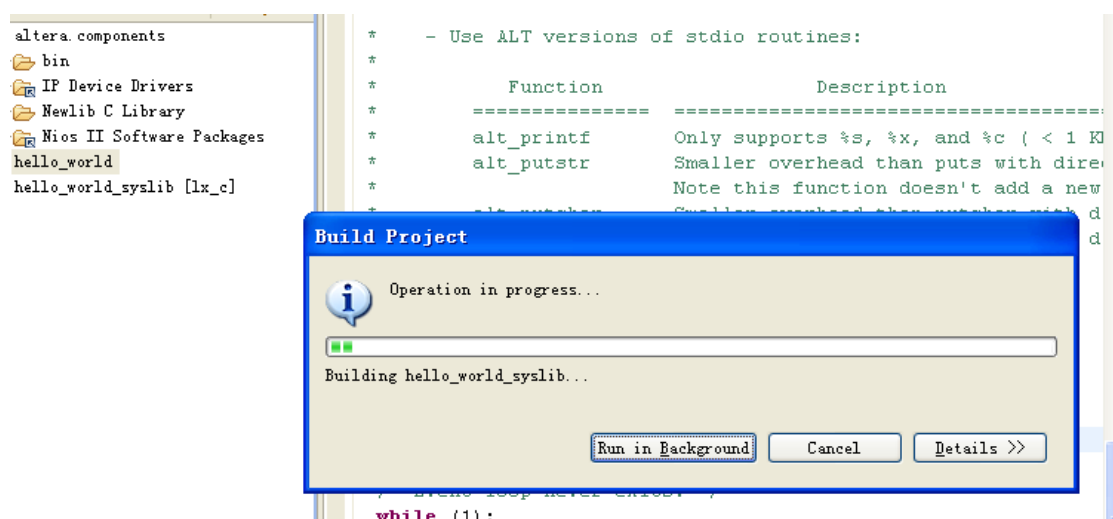
作如上图所示的配置后，点击 Finish，弹出如下图所示的对话框：



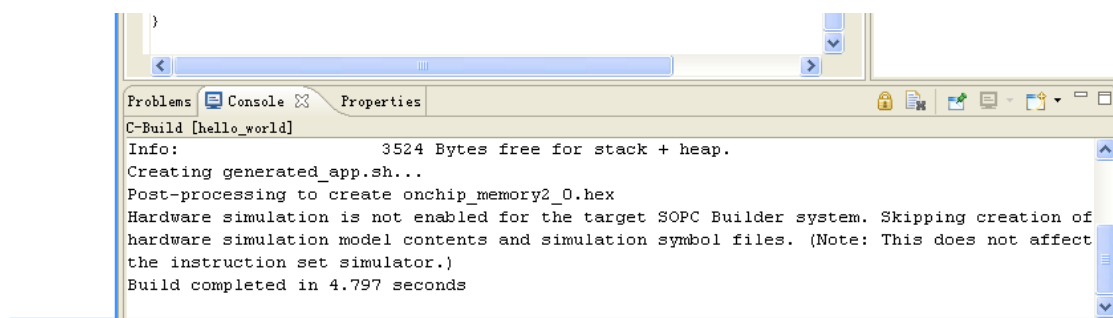
其中，箭头处的内容是可以修改的。接下来，在左面选项卡中选择 `hello_world`，并单击鼠标右键，在弹出的菜单选项中选择 `Build Project`，如下图所示：



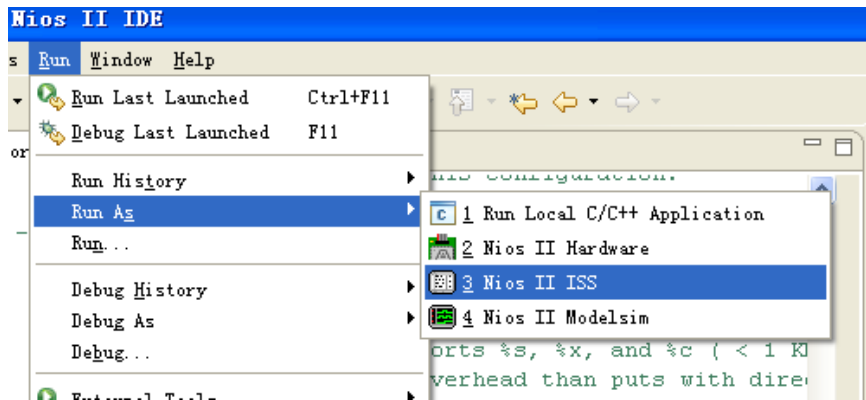
左键单击 Build Project，在弹出的对话框中显示正在进行编译，如下图所示：



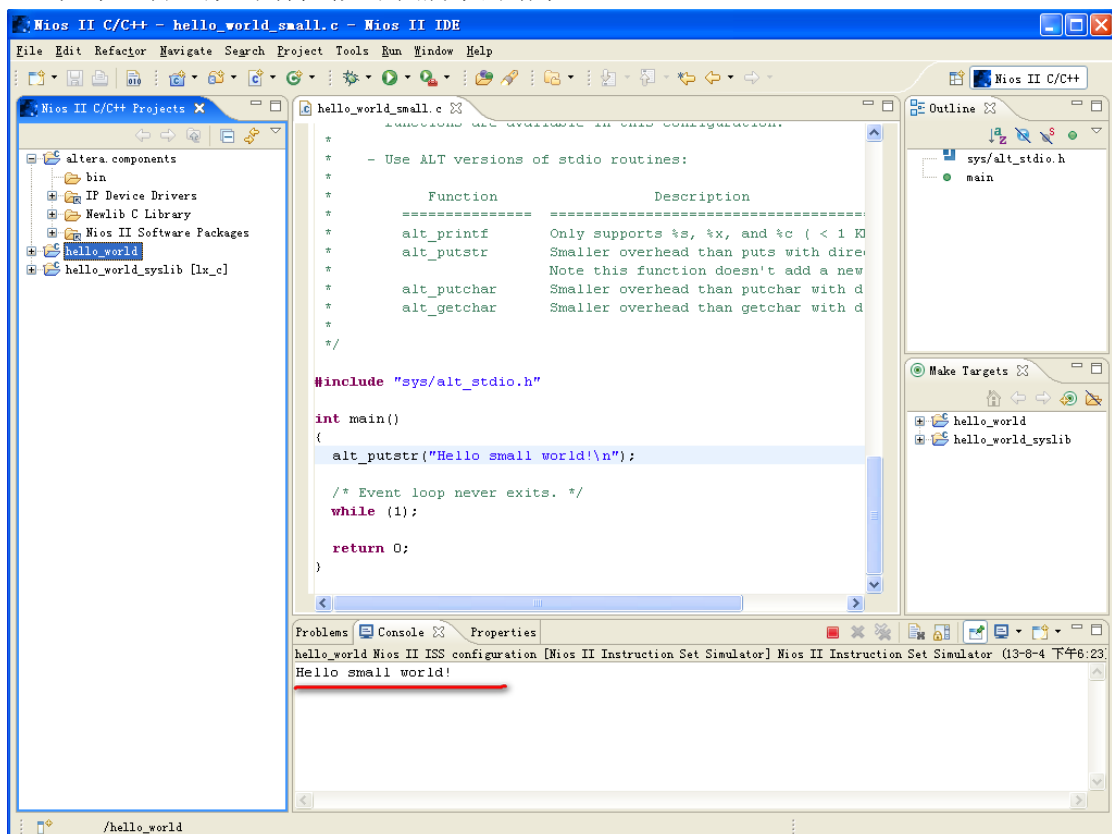
编译成功，则显示出如下图所示的内容：



成功编译后，选中菜单 Run->Run As->3 Nios II ISS，并单击 3 Nios II ISS，开始运行，如下图所示：



如果正确运行，则得到如下图所示的结果：



这样，就完成了本实验。