基于SOPC的FPGA设计实验指导

李翠锦　李成勇　代红英　编著

西南交通大学出版社

前言 PREFACE

本书是《FPGA 技术及应用》（李翠锦、朱济宇主编，西南交通大学出版社，2017）的配套实验指导书，本实验指导书选编了有代表性的实验近三十个，实验内容从简单到复杂，让使用者能够快速入手，同时本实验指导书还可以作为电子技术专业的加深课程或作为电子技术工程师的参考用书。

本书依托重庆市教委教研教改项目（项目编号：163163、172037）和重庆工程学院校内教改重点项目（项目编号：JY2015204），按照 CDIO 工程教育创新模式，结合教育部“卓越工程师教育培养计划”的实施原则，突出基本理论与实际应用相结合的特色。

本书中的实验依托北京百科荣创 EDA/SOPC 综合实验开发系统，该系统不仅可以独立完成几乎所有的 EDA 设计实验，也可以完成大多数的 SOPC 开发实验。该系统以 Altera 公司 CycloneIV E 系列的 FPGA 为核心，外部电路包括 7 寸电容触摸屏、16×16 双色 LED 点阵、7 段数码管等；控制模块如电机控制、交通灯模拟控制、按键、LED 灯控制、矩阵键盘灯；接口模块如网口、VGA、串口、SD 卡、USB、PS2、音频等。

全书由重庆工程学院李翠锦组织编写，李成勇统稿和审校，其中前言、实验一～十五由李翠锦编写、实验十六～二十二由李成勇编写、实验二十三～二十七由代红英编写。另外，在本书的编写过程中，得到了景兴红副教授的大力支持，他为本书提出了许多宝贵意见，在此表示感谢。

由于编者水平有限，书中难免存在疏漏和不足，恳请各位专家和读者批评指正。

编 者

2018 年 1 月

目录 CONTENTS

实验一　Hello 实验

一、实验目的

（1）熟悉用 Quartus Ⅱ开发 SOPC 的基本流程。
（2）熟悉用 SOPC Builder 进行 Nios Ⅱ CPU 开发的基本流程。
（3）熟悉用 Nios Ⅱ IDE 进行 C 语言编译、下载的基本过程。
（4）掌握整个 Nios Ⅱ集成开发环境。

二、硬件需求

（1）EDA/SOPC 实验开发系统一台。
（2）电源线和端口连接线若干。

三、实验原理

本实验的设计目的主要是让学生对 SOPC 有初步的认识，了解整个开发过程，并熟练掌握整个 Nios Ⅱ集成开发环境的应用。

SOPC 是 System On A Programmable Chip 的缩写，顾名思义就是把一个系统集成在单片可编程芯片中。一个最小系统应该包括中央处理单元（CPU）、随机存储器（RAM）和闪存（Flash ROM，用于存储代码、数据等），稍微复杂点的系统至少应该包括 UART、DMA、Timer、中断管理模块以及 GPIO 等。

早在 2002 年，Altera 公司就基于 SOC 的设计思想，推出了其第一款 32 位 RISC CPU 软核——Nios，那时的 Nios CPU 功能简单，执行效率低下且不支持在线调试，所以并未得到很大的推广。在 Nios CPU 基础上，Altera 公司又于 2005 年推出了其第二代 32 位 RISC CPU——Nios Ⅱ。与 Nios CPU 相比，Nios II CPU 在性能方面得到了质的提升，指令执行速度快，执行效率高，且支持 JTAG 在线调试。

Nios Ⅱ CPU 的开发流程与 Nios CPU 基本一致，唯一不同的就是 Nios CPU 的软件开发是在 Nios SDK Shell 下进行，而 Nios Ⅱ CPU 则是在 Nios Ⅱ IDE 集成环境下开发。Nios Ⅱ CPU 的基本开发流程依旧为：

（1）在 Quartus Ⅱ中新建一个工程（硬件）。
（2）在 SOPC Builder 中根据自己的需要加入各种 IP 核。
（3）利用 SOPC Builder 产生 Quartus Ⅱ能够识别的文件。
（4）在（1）中新建的工程中加入（3）中生成的文件。

（5）加入输入、输出以及双向端口，并根据需要对其命名。

（6）对（5）中命名的输入、输出核双向端口根据选定的 FPGA 进行引脚分配。

（7）编译工程。

（8）下载编辑代码到 FPGA。

（9）利用 Nios Ⅱ IDE 新建另一个工程（软件）。

（10）根据（2）中的资源，编写项目需要的代码。

（11）编译、下载并调试，查看运行结果，直到正确。

（12）如果需要，将（11）中生成的代码下载到代码 Flash 中。

SOPC 的开发流程是一个软硬件协同开发的过程。首先根据硬件需要，决定使用何种性能的 CPU，加入系统需要的外设（SRAM、Flash、Timer、UART、Timer 和 GPIO 等），此时一个基本的硬件系统便搭建起来了。利用专用工具，对这些像积木一样搭起来的系统进行编译，产生 FPGA 软件可以识别的文件，然后再用 FPGA 专用软件对这些文件进行编译，产生满足加载 FPGA 的代码，这样一个硬件平台就全部完成了。接下来的工作就是软件开发，在软件集成开发环境中编写代码，编译后，下载到 CPU 中进行调试。在整个开发过程中软件工作量相对较大。下面对 Altera 的软件开发环境作一些简要说明。

Nios Ⅱ CPU 使用的软件开发环境叫作 Nios Ⅱ IDE，它是 Nios Ⅱ系列嵌入式处理器的基本软件开发工具。所有软件开发任务都可以在 Nios Ⅱ IDE 下完成，包括编辑、编译和调试程序。Nios Ⅱ IDE 提供了一个统一的开发平台，用于所有 Nios Ⅱ处理器系统。仅仅通过一台 PC 机、一片 Altera 的 FPGA 以及一根 JTAG 下载电缆，软件开发人员就能够往 Nios Ⅱ处理器系统写入程序以及和 Nios Ⅱ处理器系统进行通讯。

Nios Ⅱ IDE 基于开放式的、可扩展 Eclipse IDE project 工程以及 Eclipse C/C++开发工具（CDT）工程。

Nios Ⅱ IDE 为软件开发提供四个主要的功能：

（1）工程管理器。

Nios Ⅱ IDE 提供多个工程管理任务，加快嵌入式应用程序的开发进度。

新工程向导——Nios Ⅱ IDE 推出了一个新工程向导，用于自动建立 C/C++应用程序工程和系统库工程。采用新工程向导，能够轻松地在 Nios Ⅱ IDE 中创建新工程。

软件工程模板——除了工程创建向导，Nios Ⅱ IDE 还以工程模板的形式提供了软件代码实例，帮助软件工程师尽可能快速地推出可运行的系统。

（2）编辑器和编译器。

Altera Nios Ⅱ IDE 提供了一个全功能的源代码编辑器和 C/C++编译器和文本编辑器——Nios Ⅱ IDE 文本编辑器是一个成熟的全功能源文件编辑器。这些功能包括：语法高亮显示 C/C++、代码辅助/代码协助完成、全面的搜索工具、文件管理、广泛的在线帮助主题和教程、引入辅助、快速定位自动纠错、内置调试功能。

C/C++编译器——Nios Ⅱ IDE 为 GCC 编译器提供了一个图形化用户界面，Nios Ⅱ IDE 编译环境使设计 Altera 的 Nios Ⅱ处理器软件更容易，它提供了一个易用的按钮式流程，同时允许开发人员手工设置高级编译选项。

Nios Ⅱ IDE 编译环境自动地生成一个基于用户特定系统配置（SOPC Builder 生成的 PTF 文件）的 makefile。Nios Ⅱ IDE 中编译/链接设置的任何改变都会自动映射到这个自动生成的 makefile 中。这些设置可包括生成存储器初始化文件（MIF）的选项、闪存内容、仿真器初始化文件（DAT/HEX）以及 profile 总结文件的相关选项。

（3）调试器。

Nios Ⅱ IDE 包含一个强大的、在 GNU 调试器基础之上的软件调试器——GDB。该调试器提供了许多基本调试功能，以及一些在低成本处理器开发套件中不会经常用到的高级调试功能。

基本调试功能——Nios Ⅱ IDE 调试器包含如下的基本调试功能：运行控制、调用堆栈查看、软件断点、反汇编代码查看、调试信息查看、指令集仿真器。

高级调试——除了上述基本调试功能之外，Nios Ⅱ IDE 调试器还支持以下高级调试功能：硬件断点调试 ROM 或闪存中的代码、数据触发、指令跟踪。

（4）闪存编程器。

使用 Nios Ⅱ处理器的设计都在单板上采用了闪存，可以用来存储 FPGA 配置数据和/或 Nios Ⅱ编程数据。Nios Ⅱ IDE 提供了一个方便的闪存编程方法。任何连接到 FPGA 的兼容通用闪存接口（CFI）的闪存器件都可以通过 Nios Ⅱ IDE 闪存编程器来编写。除 CFI 闪存之外，Nios Ⅱ IDE 闪存编程器能够对连接到 FPGA 的任何 Altera 串行配置器件进行编程。

四、实验内容

为了熟悉 SOPC 的基本开发流程，本实验要完成的任务就是设计一个最简单的系统，系统中包括 Nios Ⅱ CPU、作为标准输入/输出的 JTAG UART 以及存储执行代码 onchip_rom。通过 SOPC Builder 对系统进行编译，然后通过 Quartus Ⅱ对系统进行二次编译，并把产生的 FPGA 配置文件通过 USB 下载电缆下载到实验箱上，这时便完成了本实验中的硬件开发。接下来的工作是软件协同开发——在 Nios Ⅱ IDE 中编写一个最简单的 C 代码，对其编译后，通过 USB 下载电缆下载到 FPGA 中执行，执行的结果就是在 Nios Ⅱ IDE 的 Console 窗口打印十条信息——“Hello form Nios Ⅱ!”。

五、实验步骤

完成本实验的实验步骤为：

（1）在开始菜单中，打开 Quartus Ⅱ 12.0。

（2）点击 File 菜单中的 New Project Wizard，新建一个工程。本实验以../exp1_hello 文件夹（文件夹不能含有空格，不能带中文路径）为例，工程名称为 test，如图 1-1 所示。

（3）点击 Next 按钮，进入到添加工程文件步骤。由于工程全部为空，所以也没有文件加入，因此直接点击 Next 进入到选择芯片步骤（在 Family 下拉菜单中选择 Cyclone IV E；然后在 Available devices 中选择 EP4CE40F29C6），如图 1-2 所示。

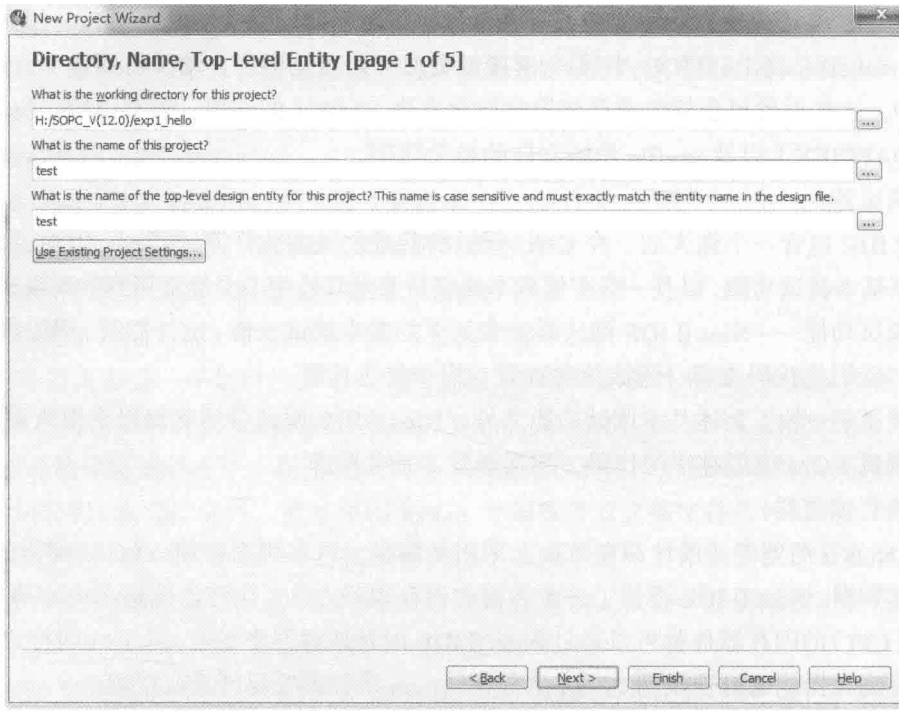


图 1-1　新建工程

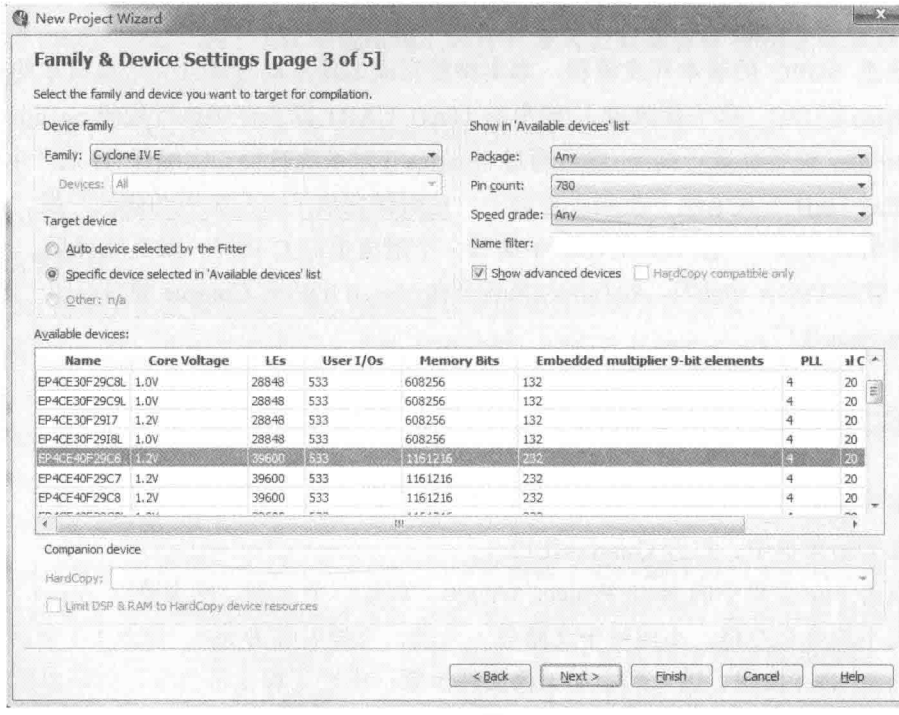


图 1-2　选择芯片

（4）FPGA 选好后，点击 Next，进入选择其他 EDA 工具窗口。本实验中不需要任何 EDA 工具，所以直接点击 Next 按钮，然后再点击 Finish 按钮，完成新工程的创建。如图 1-3 所示。

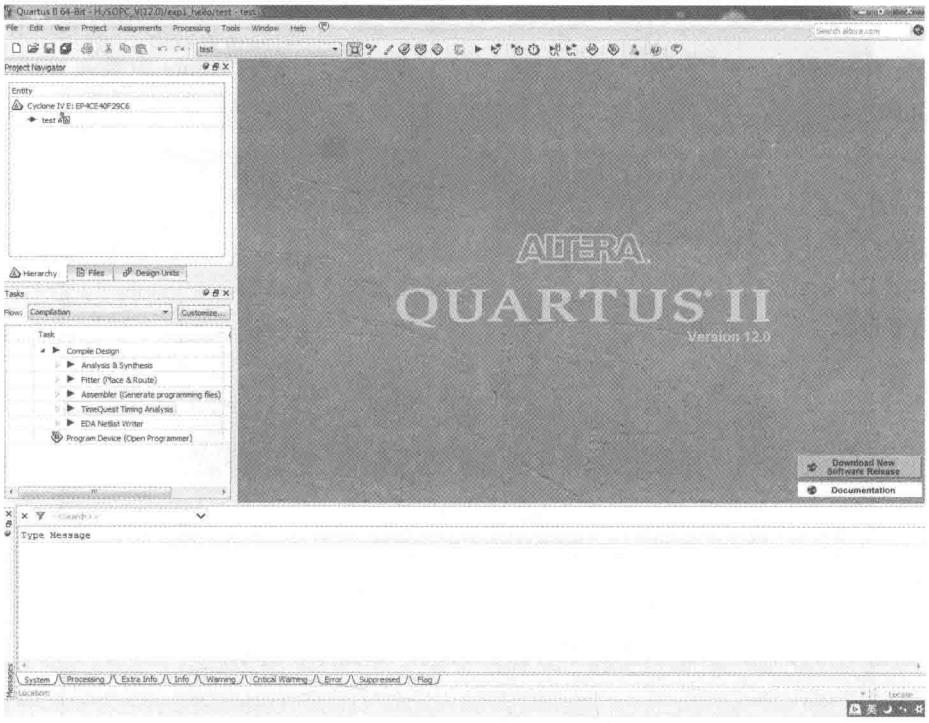


图 1-3　新建的工程界面

（5）点击 File 菜单中的 New，在 Design File 标签中选择 Block Diagram/ Schematic File。然后再选择 File>Save As，在 File name 中键入“test”，点击“保存”，新建一个工程文件。如图 1-4 所示。

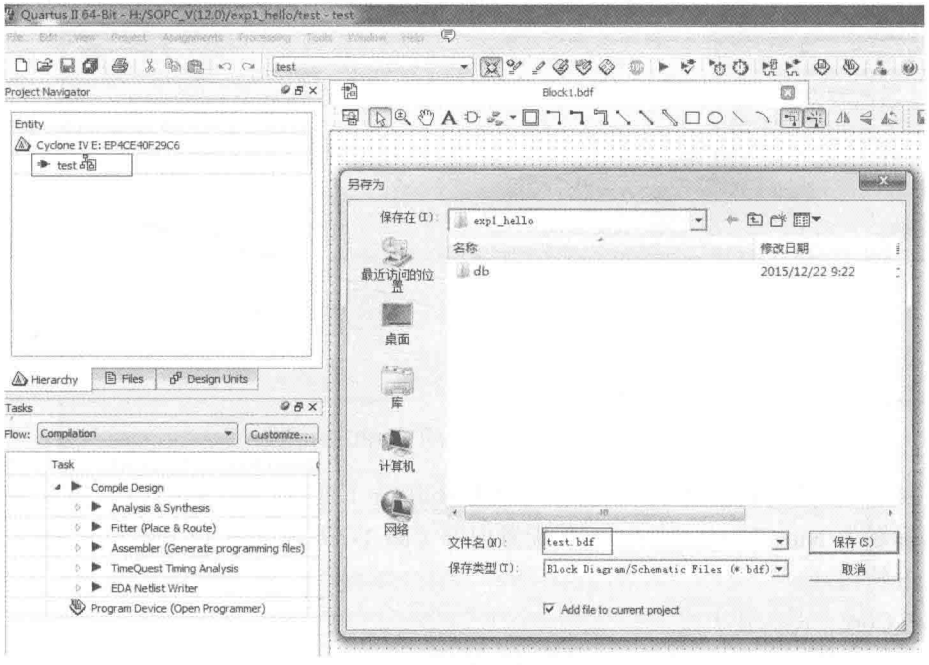


图 1-4　新建的原理图工程文件界面

（6）点击 Tools 菜单中的 SOPC Builder，启动 SOPC Builder 工具，SOPC Builder 启动时显示 Create New System 对话框，见图 1-5。在对话框中的 System Name 中键入 Kernel，并在 Target HDL 中选择 Verilog，然后点击 OK，创建一个名为 Kernel 的 NIOS II 软核。如图 1-5 所示。

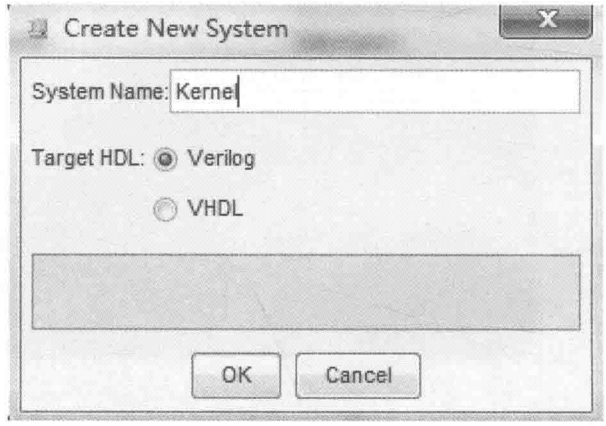


图 1-5　Create New System 对话框

（7）点击 OK 按钮，开始创建属于自己的系统。如图 1-6 所示。

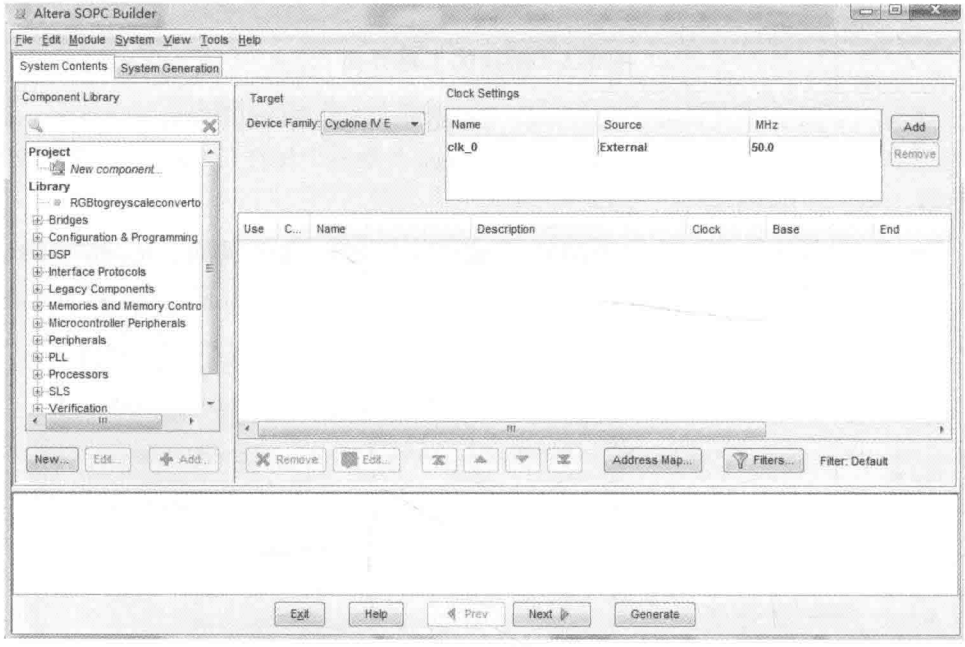


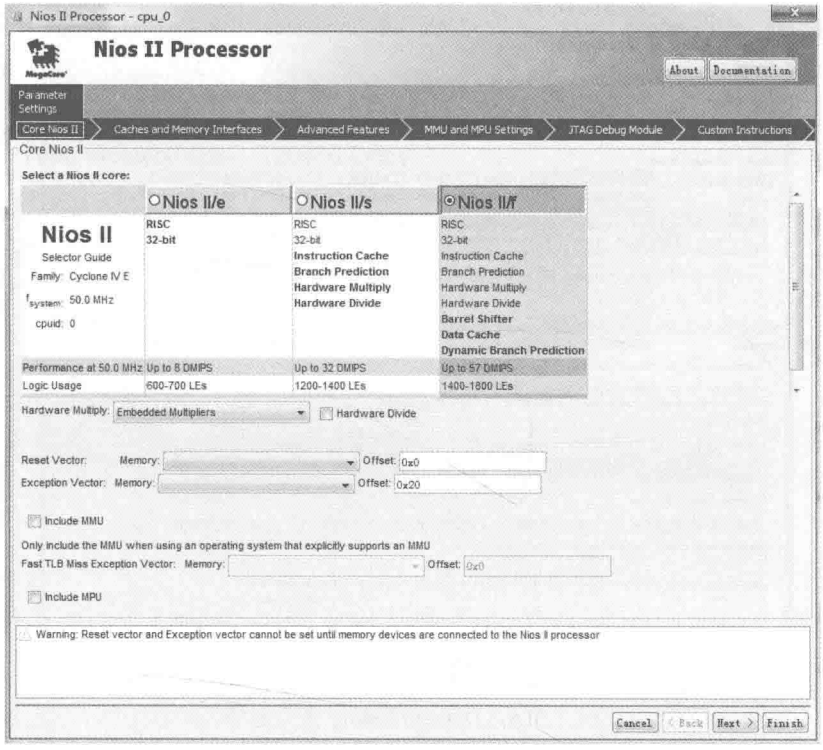
图 1-6　不带有模块的 SOPC Builder

（8）加入 32 位 Nios CPU。在 Altera SOPC Builder 下面选择 NiosII Processor，点击 Add，将会弹出标题为 Nios II Processor-cpu 的配置向导（图 1-7），按照以下方式设置参数，设置完点击 Finish 按钮。

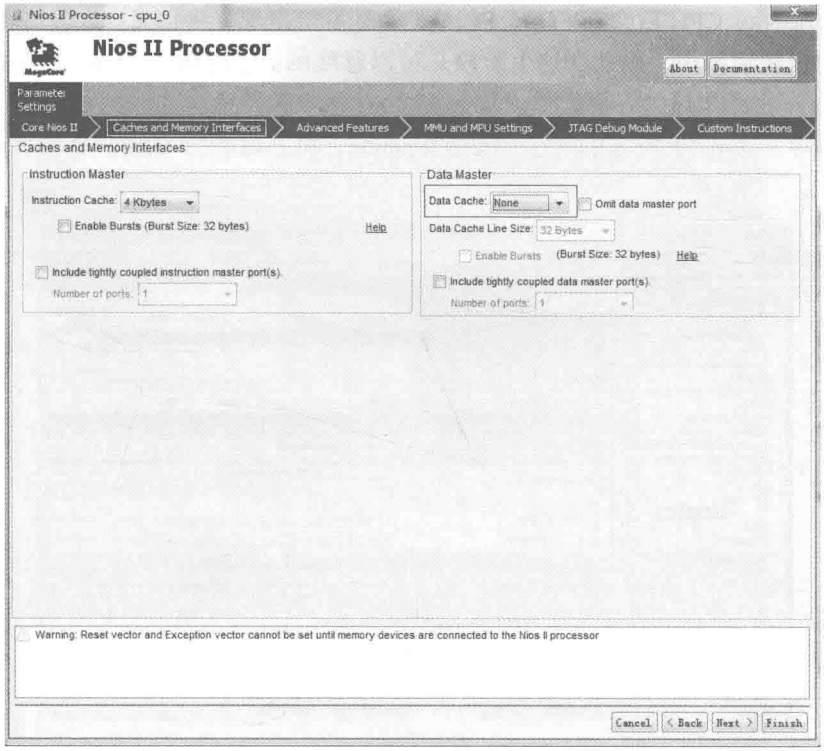
Nios II Core：Nios II /f

Caches and Memory Interfaces / Mata Master：None

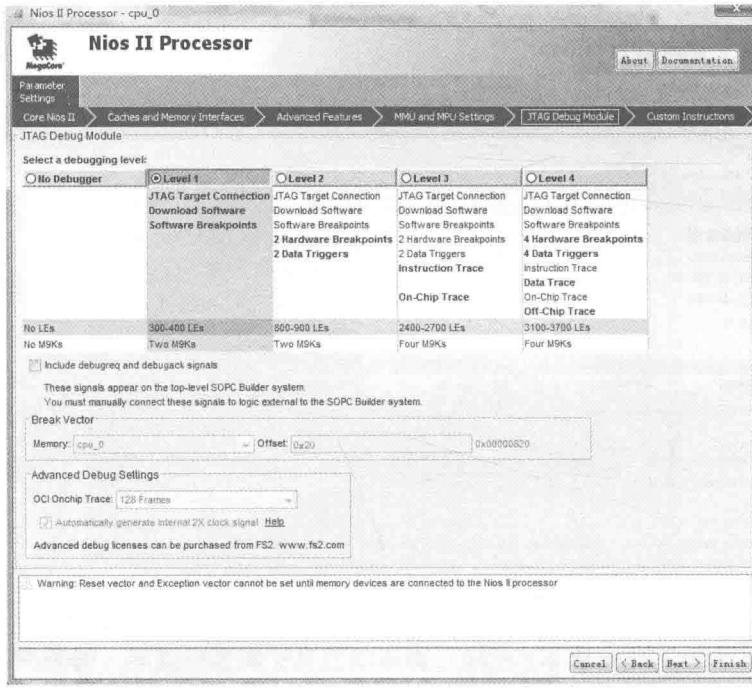
JTAG Debug Module：Level 1



（a）NIOS II Core 选型



（b）Mata Master 配置



（c）JTAG Debug Module 模式选择

图 1-7　Nios II Processor_cpu 配置

注意：加入 Nios CPU 后会在 SOPC Builder 消息窗口出现警告信息，这些信息会在后面向系统加入其他模块后消失，因此在这个阶段是可以忽略的。这时图 1-7（a）的 Reset Vector 和 Exception Vector 将无法设置，要在加入 onchip_rom 后才能设置。

（9）右键单击加入的 Nios II CPU，选择 Rename，将其命名为 CPU。将时钟修改为 clk，如图 1-8 所示。

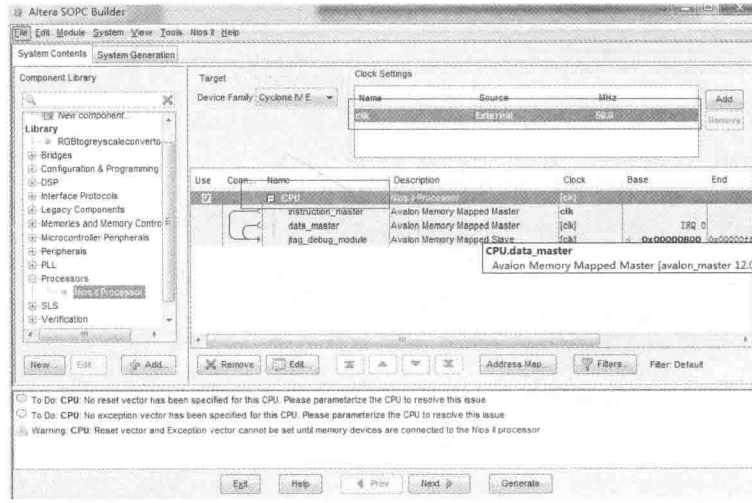


图 1-8　加入名为 CPU 的 niosii 系统

（10）加入 onchip_rom。在 Memories and memory Controllers 下的 on-chip 选择 On-Chip

Memory（RAM or ROM）并点击 Add，会出现 On-Chip Memory（RAM or ROM）配置向导，在标签中指定如下选项（图 1-9）。

— Size： Data width 32

Total memory size 20480 bytes

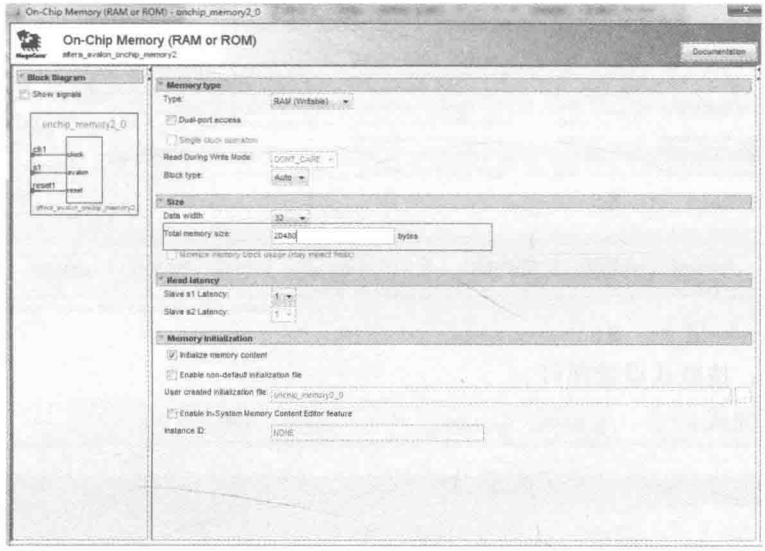


图 1-9 On-Chip Memory（RAM or ROM）设置向导

点击 Finish 按钮，鼠标右键修改名字为 onchip_memory。

此时双击 CPU，设置如图 1-10 所示。

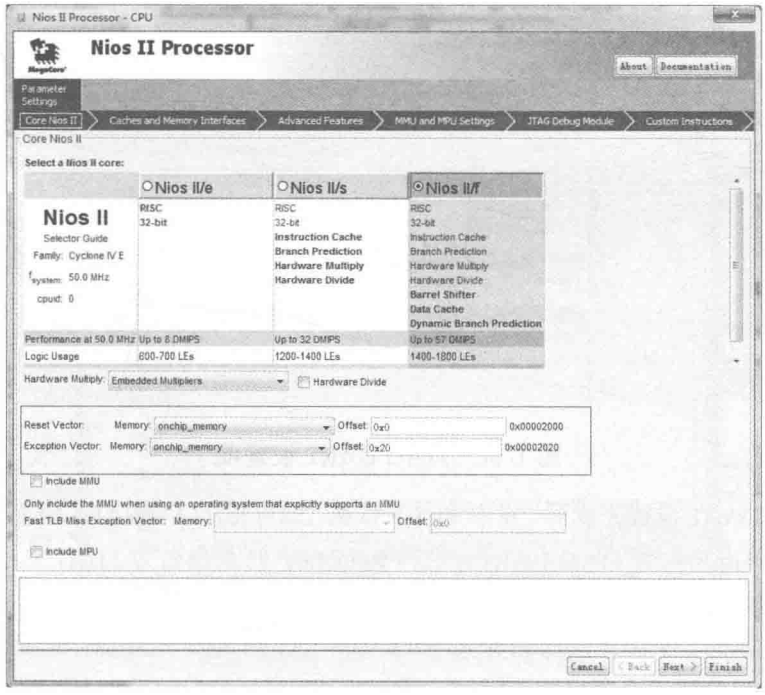


图 1-10 给 CPU 添加 ROM

Reset Vector：Memory：onchip_rom

Exception Vector：Memory：onchip_rom

点击 Finish，仍会看到红色警告，此时点击 System->Assign Base Addresses 选项，重新分配器件地址，所有警告和错误就都没有了。

（11）加入 JTAG UART。点击 Interface Protocols -> Serial 中的 JTAG UART，并作如下设置（图 1-11）：

Write FIFO :

Depth ： 64，

IRQ Threshold ： 8；

Read FIFO :

Depth ： 64，

IRQ Threshold ： 8；

Simulation：按默认设置即可。

点击 Finish 完成。

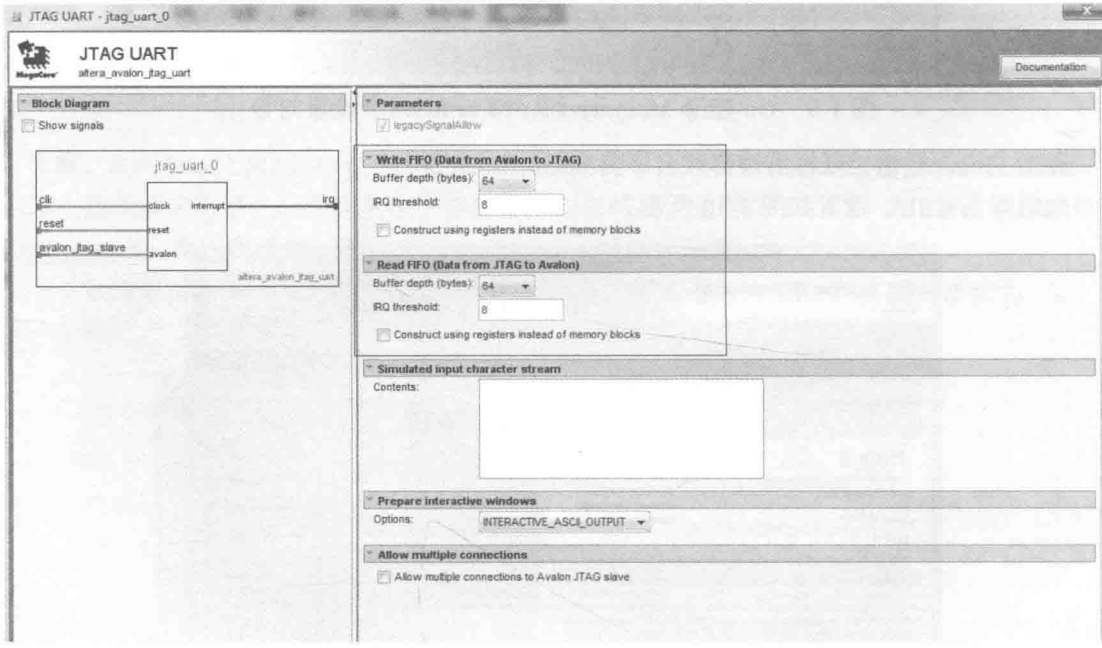


图 1-11　JTAG UART 设置项

（12）JTAG UART 设置完成后，点击 Finish 按钮，即可把 JTAG UART 添加到新建系统中。

（13）右键单击加入的 JTAG UART，选择 Rename，将其命名为 JTAG_UART。

（14）到此为止，本实验所需的系统就完成了，完成后的系统如图 1-12 所示。当软件提示基地址冲突的提示时，可点击 System 菜单下的 Auto-Assign Base Address，重新分配基地址（注意：此时以前出现的错误会消失）。

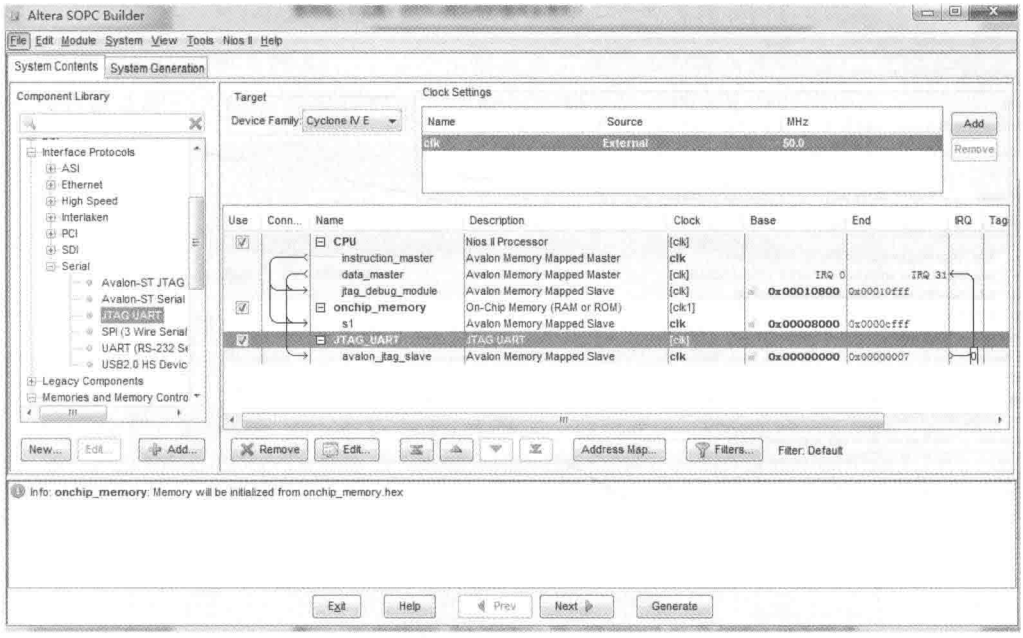
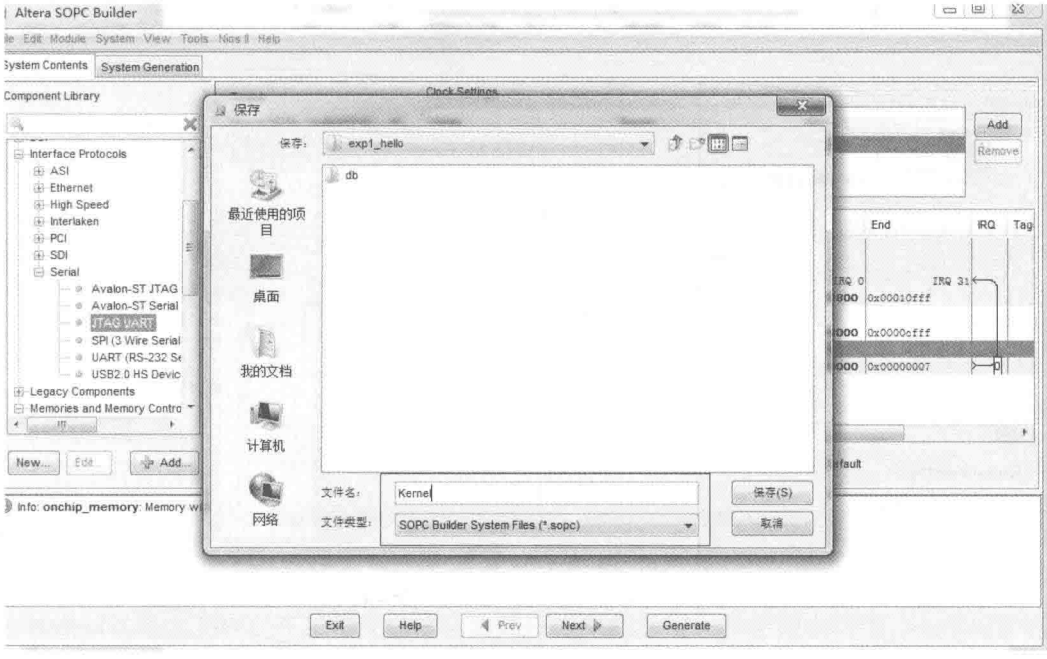
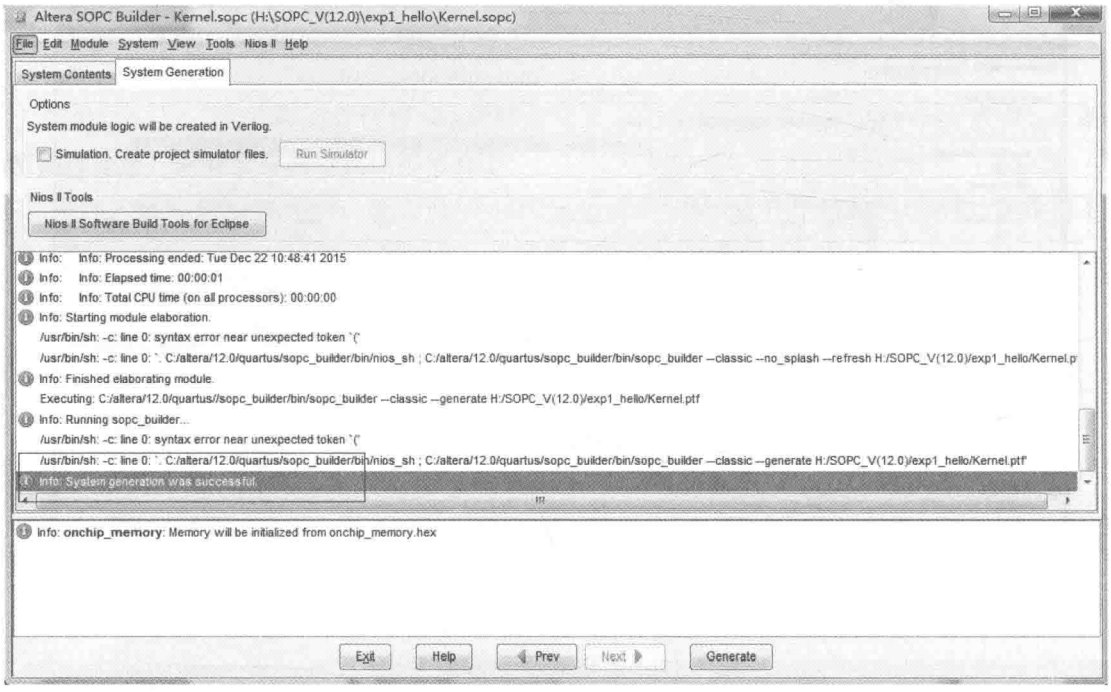


图 1-12　本实验中创建的系统

（15）到此时为止系统 IP 模块就已添加并设置好了，如图 1-12 所示。点击 System Generation，按照默认的配置，然后再点击 Altera SOPC Builder 窗口下方的 Gernerate。当提示保存信息的时候，文件名一定要和 NIOS Ⅱ软核名字一样，否则再次打开时就会报错，保存为 Kernel，如图 1-13（a）所示。在生成过程中，相关消息会出现在 System Gerneration 标签的消息框中。系统编译结束后如果编译通过，会出现如图 1-13（b）所示的提示界面。



（a）保存软核配置文件



（b）系统编译成功界面

图 1-13

（16）点击 Exit 按钮退出 SOPC Builder 窗口。重新返回到 Quartus II 12.0 窗口，在新建的原理图文件空白区域双击鼠标左键，在弹出的 Symbol 对话框中，选择 Libraries 窗口下面 Project 文件夹中的 kernel，如图 1-14 所示。

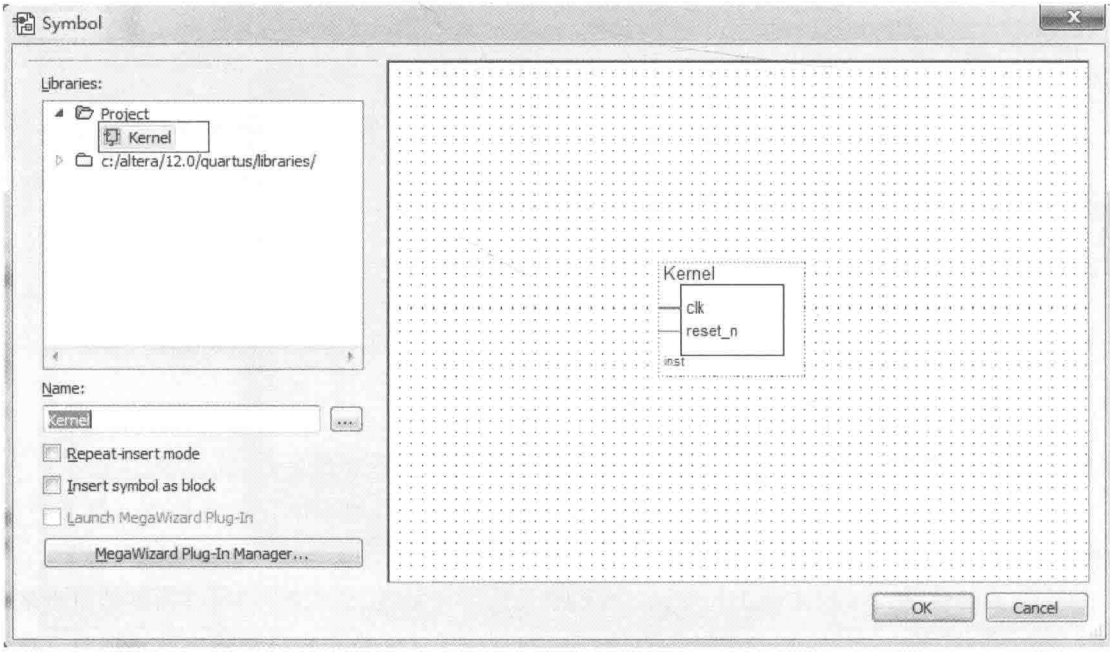


图 1-14　选择添加到 Quartus II 工程中的器件

（17）点击 OK 按钮，添加 SOPC Builder 产生的 Kernel 内核到创建的工程文件中。

（18）点击 File 菜单下的 Save（也可直接点击工具栏上的存盘按钮），此时会弹出如图 1-15 所示的对话框，直接点击保存即可。此时的工程文件如图 1-16 所示。

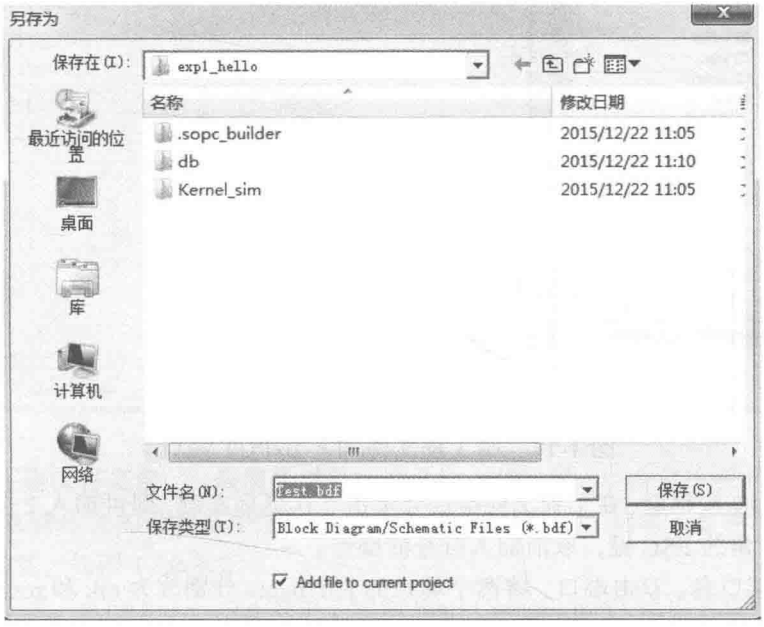


图 1-15　保存工程文件

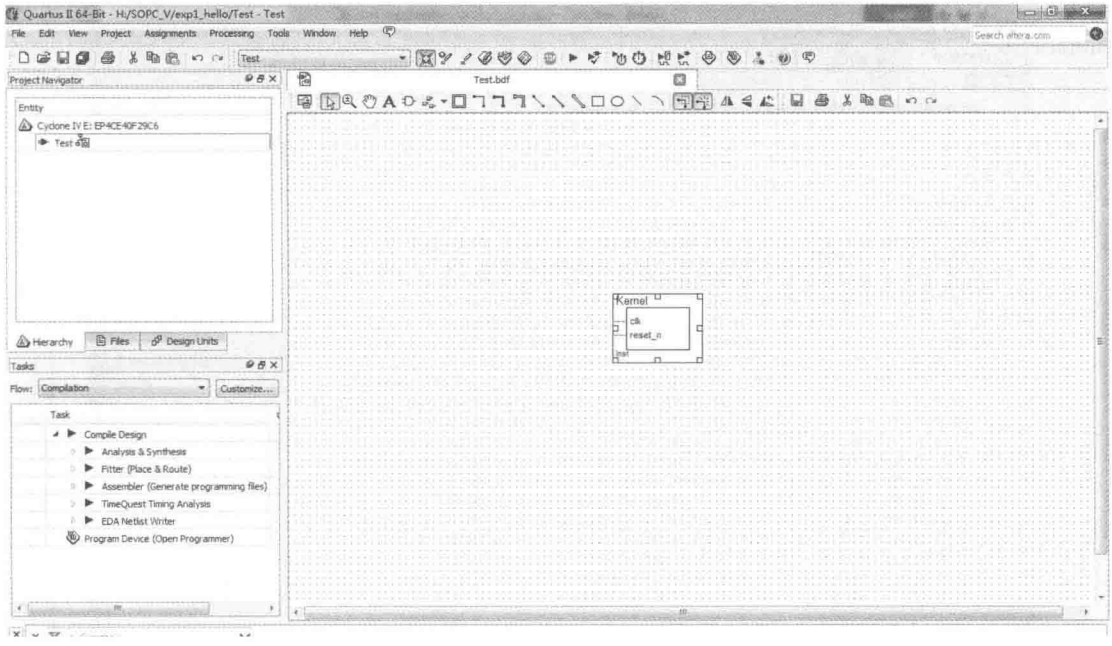


图 1-16　添加 Kernel 系统后的工程文件界面

（19）加入输入端口。在工程文件的空白处双击鼠标左键，并在弹出的 Symbol 对话框右下侧 Name 栏中键入“input”，并选中 Repeat-insert mode，如图 1-17 所示。