

实验十二 SOPC 软硬件系统搭建

一、实验目的

1. 设计一个基本的 SOPC 的硬件系统，掌握 NiosII 系统设计的思想。
2. 熟悉 Nios II IDE 使用环境，编写简单软件程序测试 Jtag uart 通讯是否正常。

二、实验设备

硬件： B-ICE-EDA/SOPC 创新开发实验平台

B-ICE-EDA/SOPC 核心板

软件： Quartus II 9.0

Nios II 9.0

三、实验内容

通过标准 C 函数在屏幕上面打印 “hello from Nios II”。

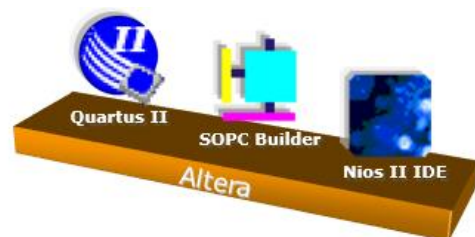
四、实验原理：

1. SOPC 设计流程：

SOPC 设计包括硬件和软件两部分。

硬件设计：基于 Quartus II 和 SOPC Builder

软件设计：基于 Nios II IDE



▪ Quartus

- 完成 NiosII 整个系统的设计、分析、综合、硬件优化和适配
- 配置文件编程下载、硬件系统测试

▪ SOPC Builder

- NIOSII 系统硬件开发环境
- 实现 Nios II 系统配置和生成
- Nios II 系统相关的监控和软件调试平台的生成

▪ NIOS II IDE

- 完成基于 Nios II 系统的软件开发和调试
- 将 FPGA 配置信息写入 Flash 或者 EPCS

2. 构建一个基本的 SOPC 硬件系统，需要添加的 IP 核模块包括：

- Nios II CPU
- Timer
- SDRAM 控制器
- External Flash 总线
- External Flash 接口
- Avalon 三态桥
- Button PIO
- JTAG UART 接口
- LED 灯

五、实验步骤

第一部分：硬件系统设计

1. 新建工程：

在 D 盘中新建一个文件夹 D:\ clock，此文件夹用于存放整个工程。

打开 Quartus II ，在菜单中选择 File—> New Project Wizard 将会出现一个信息框，这个对话框介绍创建工程步骤，可以直接选 Next，这时会出现如图 1 所示的对话框。这里需输入的是欲创建工程的基本信息，三个输入栏中分别输入的是工程将被保存的路径及工程文件夹、工程的名称和顶层实体的名称。建议工程名与顶层实体名称保持一致。输入完毕我们就可以点击 Next。

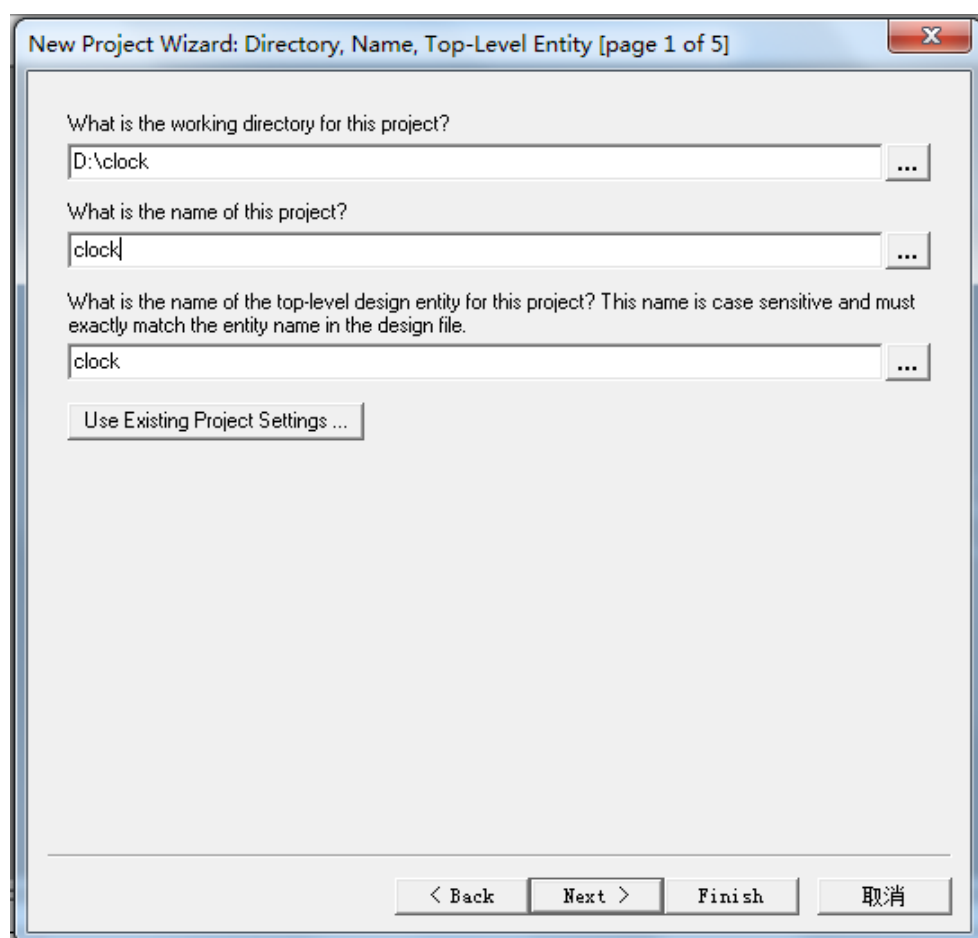


图 1 新建工程基本信息对话框

然后出现图 2 所示的添加工程文件对话框。在这里需要做的是将已经写好的 Verilog/VHDL 文件加入到工程中。本次实验，可以直接点击 Next，以后再添加 Verilog/VHDL 文件的工作。

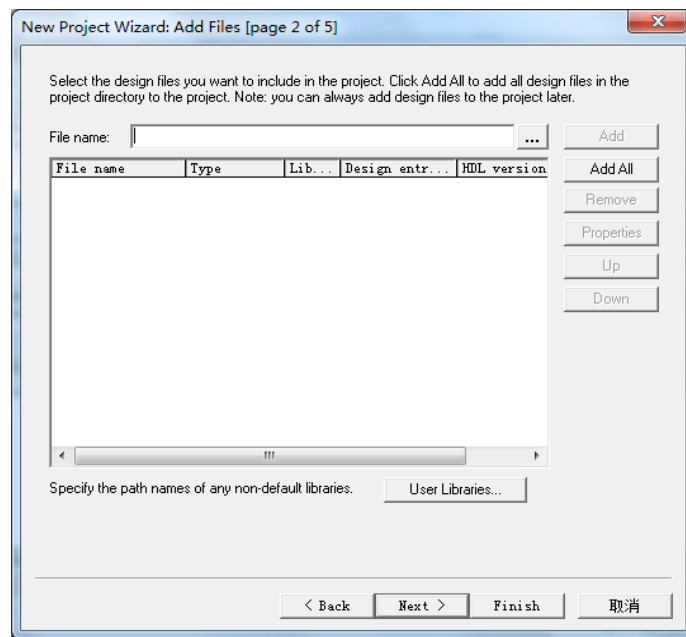


图 2 添加工程文件对话框

然后出现图 3 所示对话框，这里我们需要完成的是选择器件的工作。选择 **EP3C55F484I7**，选择完成后，点击 Next。

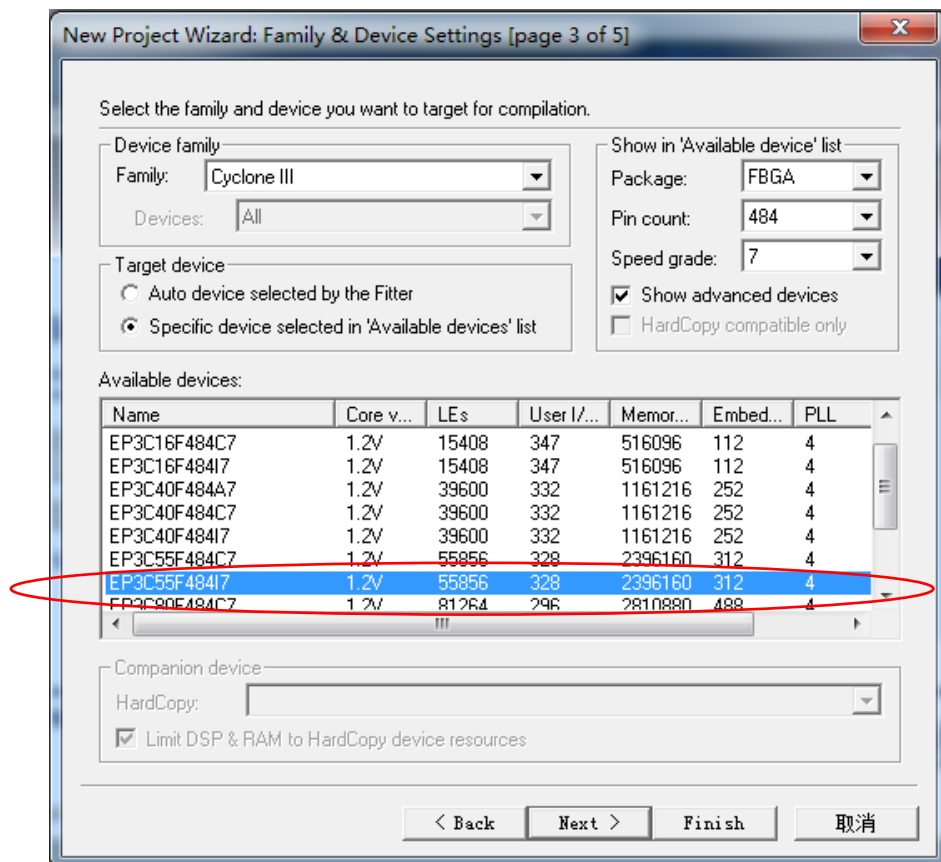


图 3 目标器件选择对话框

然后出现图 4 所示对话框，这里询问是否选用第三方 EDA 工具，我们不选用，直接点击 Next。

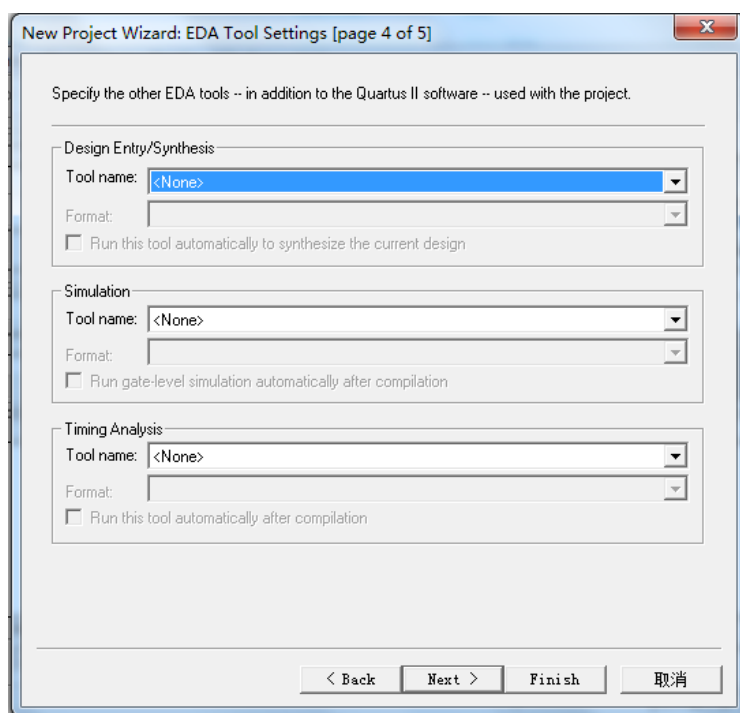


图 4 EDA 工具选择对话框

然后出现图 5 所示对话框，该对话框给出了所生成工程的信息，点击 Finish 就完成了工程创建。

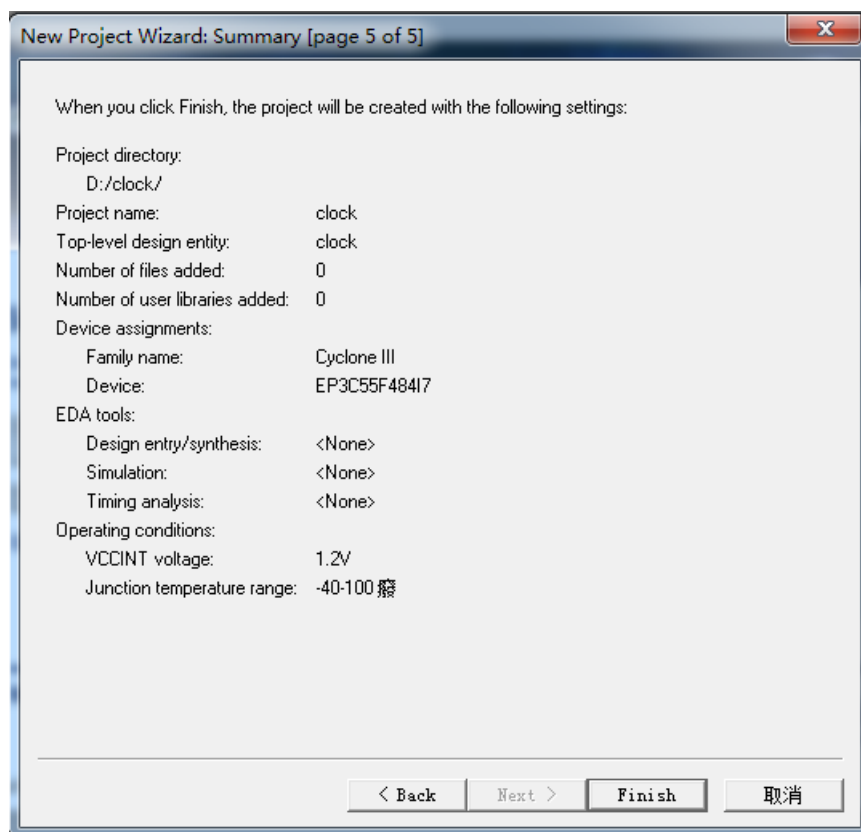


图 5 生成工程的信息

点击 Finish 按钮，Quartus II 会自动打开这个工程，可以看到顶层实体名出现在工程导航窗口中。如图 6 所示。

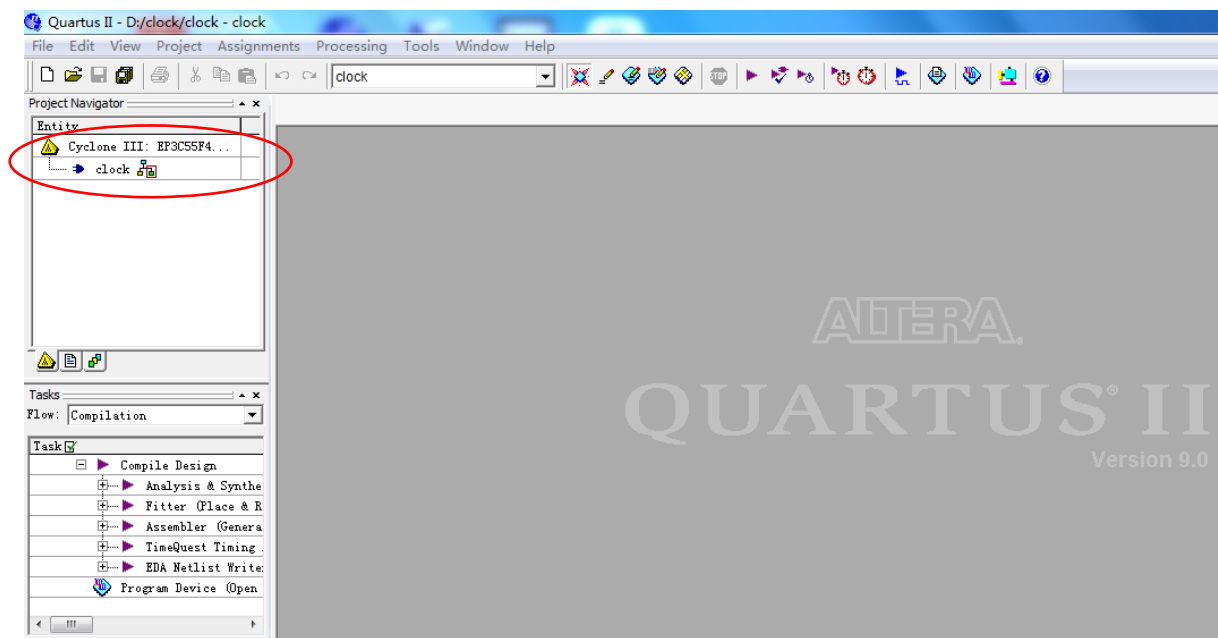


图 6

3. 创建 Nios II 系统模块

3.1 创建顶层实体

选择 File→New; 在 Device Design File 页中, 选择 Block Diagram / Schematic File, 即原理图文件(也可以选择硬件描述语言的文件形式), 单击 OK。如图 7 所示。

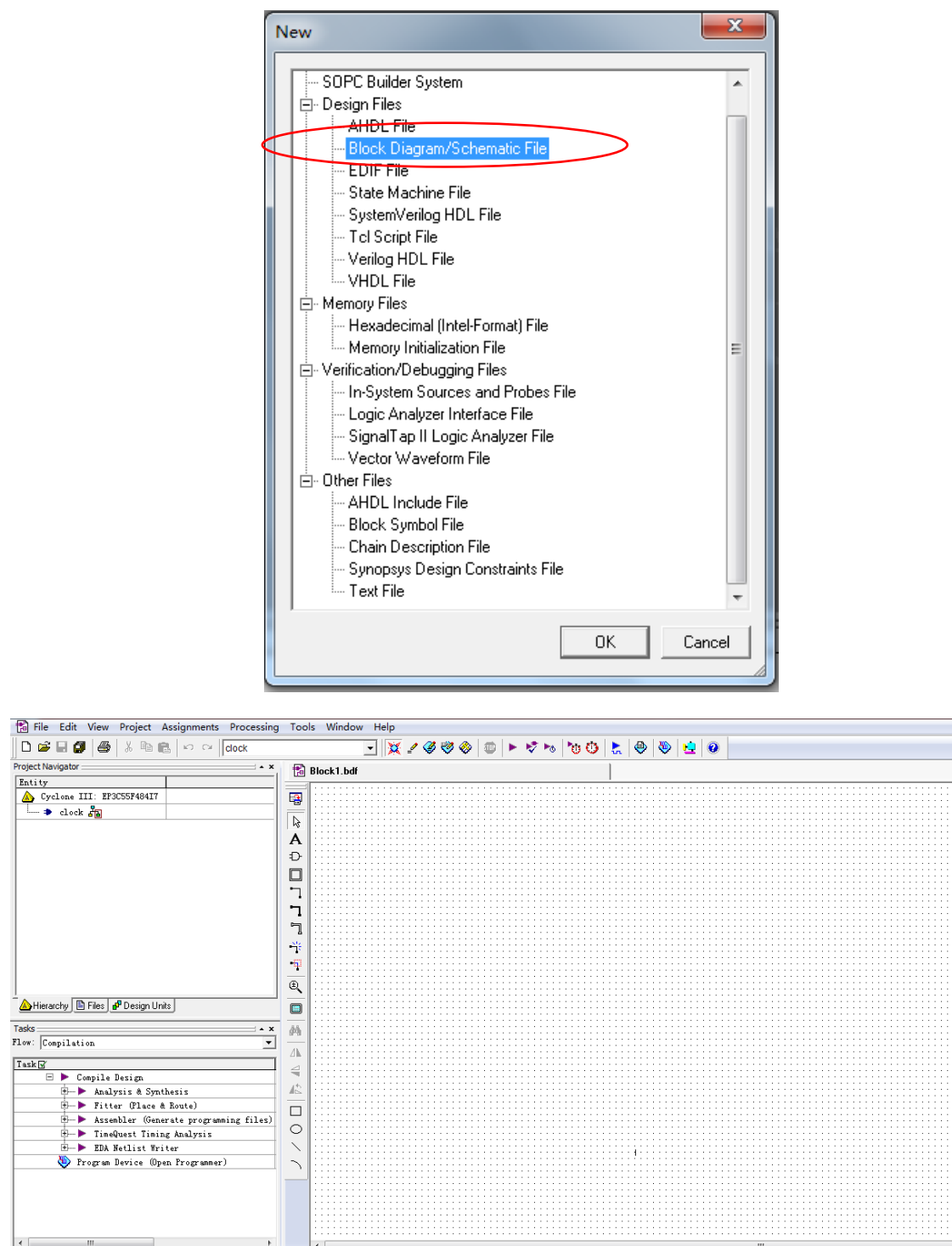


图 7

选择 File→Save As，显示的目录为之前设置的工程目录，文件名为之前设置的顶层实体名。如图 8 所示。保存后，关闭窗口。

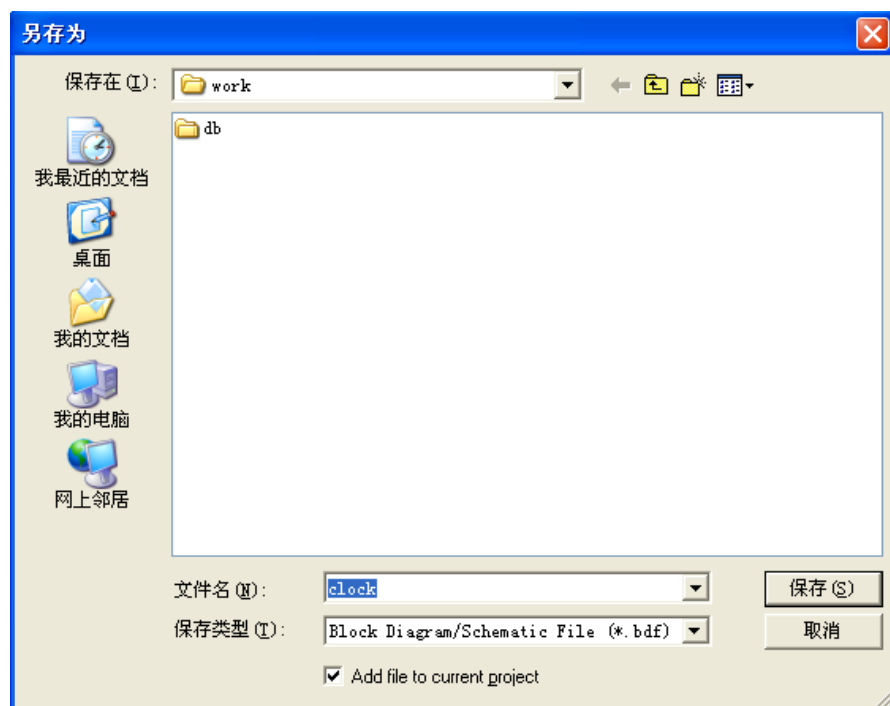



图 8

3.2 创建 Nios II 系统模块

(1) 创建系统

单击 Quartus II 软件上方工具栏中的  按钮，或选择 *Tools -> SOPC Builder...* 打开 SOPC Builder。在弹出的对话框中填入片上系统的名称。注意不要与工程名称相同。

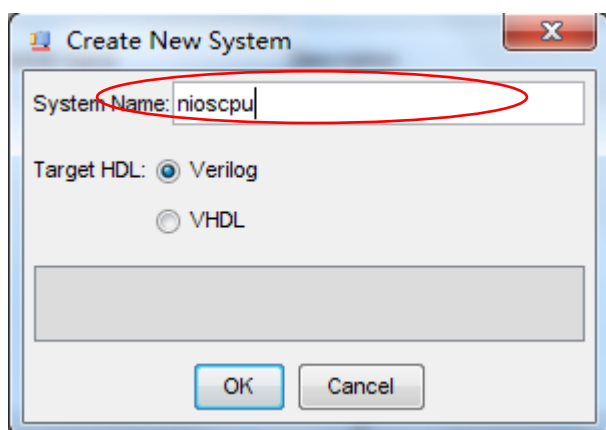


图 9

(2) 设置系统主频和指定目标 FPGA

设置系统的时钟频率 50MHz，在 Device Family 选择 Cyclone III。

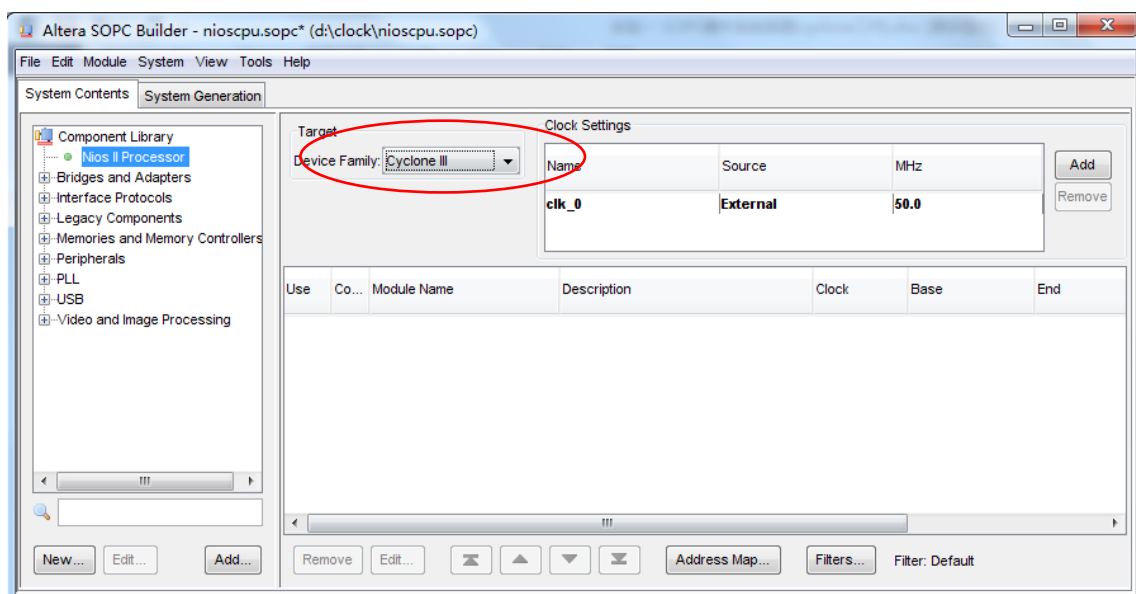


图 10

(3) 加入 Nios II CPU

双击 Altera SOPC Builder -> Nios II Processor，根据需要选择 Nios III 核

Hardware Multiply 选择 none，不选择 Hardware Divide

注意：①Embedded Multipliers：使用专门的内嵌硬件乘法单元（乘法速度最快）。
②Logic Elements，使用逻辑单元也就是 FPGA 中的查找表（速度较慢）。③None：只能通过软件模拟乘法，速度最慢。

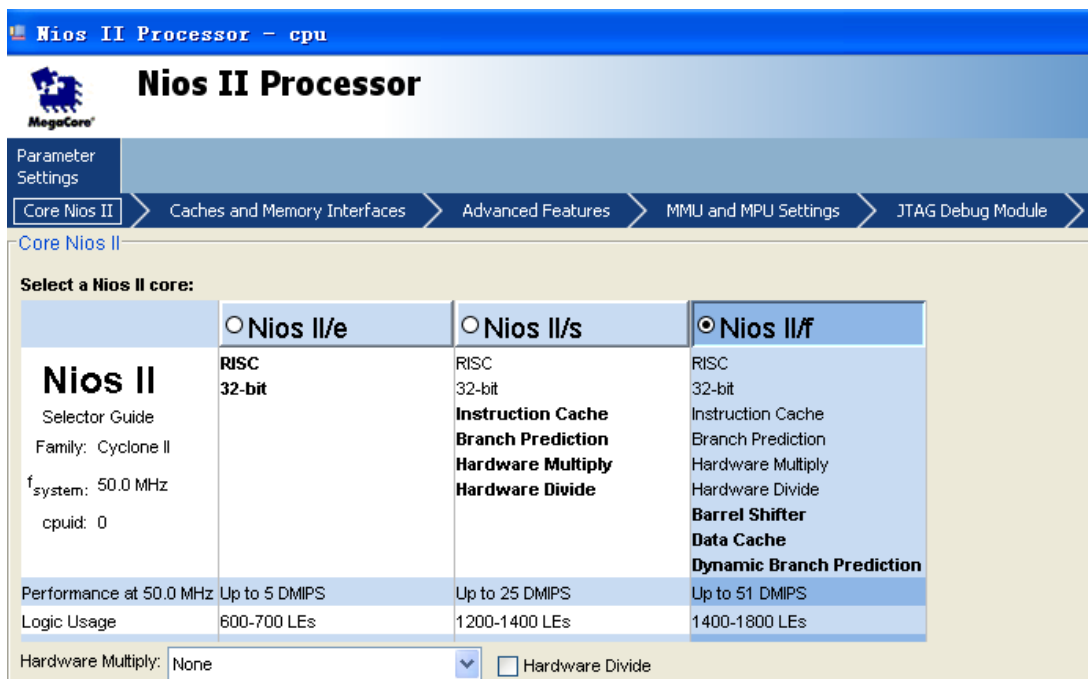


图 11

点击 Next，进入缓存设置窗口，如图 12；
可设 Instruction Cache 为 4Kbytes，Data Cache 2Kbytes

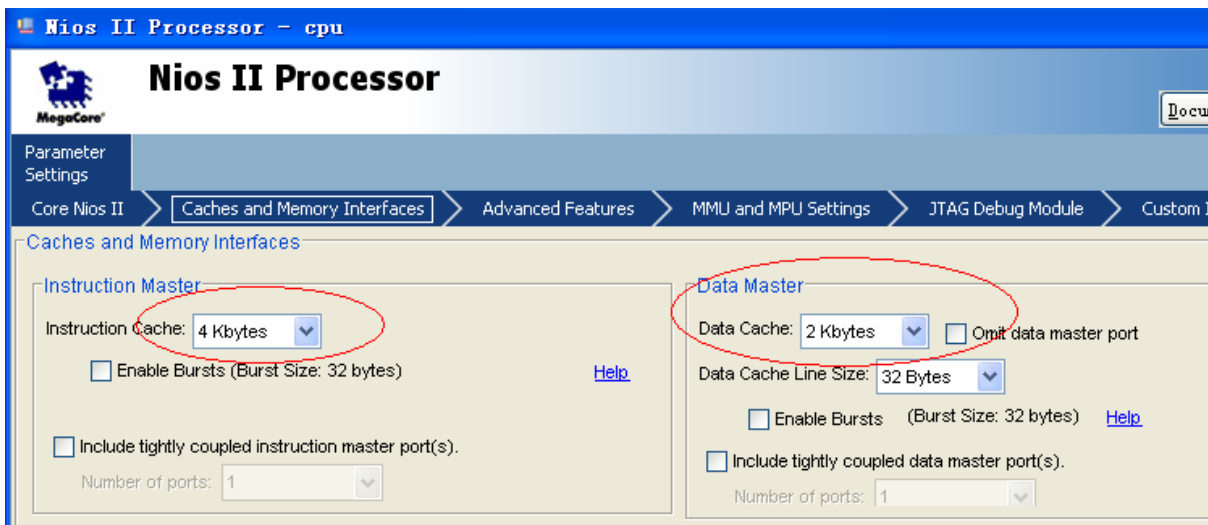


图 12

JTAG 调试级别选择
选择 Level1：该级别支持软件的断点调试。
JTAG 调试模块要占用较多的逻辑资源，系统调试完毕了可以选用 No Debugger

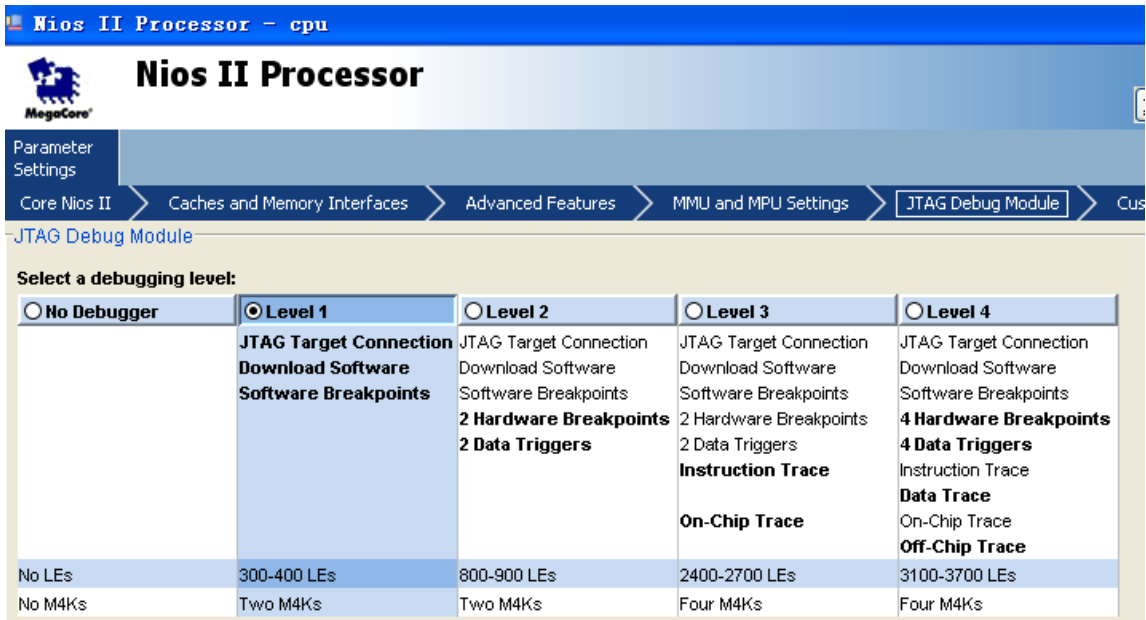


图 13

自定义指令的设置，不作任何的设置。点击 Finish 完成 NIOS II 处理器的添加，如图 14 所示。

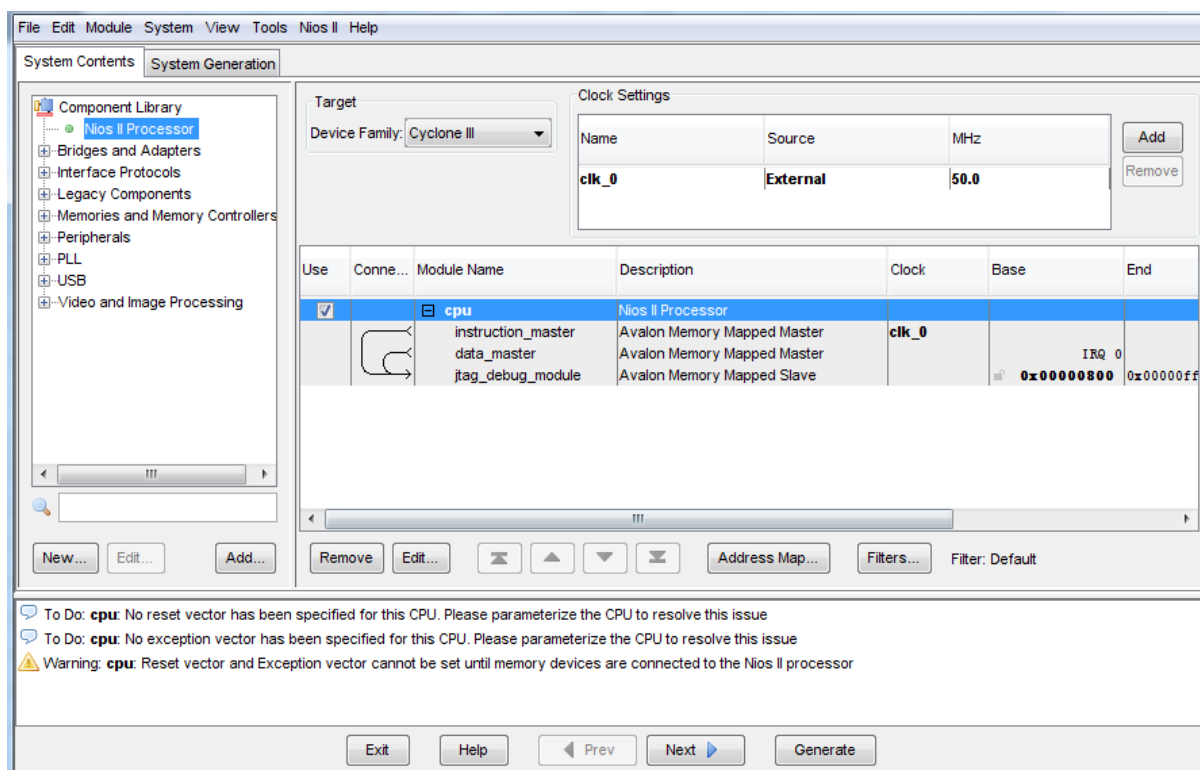


图 14

(4) 加入添加其它外设 IP 模块

除了 Nios II CPU，最小系统还需要添加的 IP 模块包括：

- ❖ External Flash 接口
- ❖ External Flash 总线
- ❖ SDRAM 控制器
- ❖ JTAG_UART 接口
- ❖ BUTTON 按键
- ❖ LED 灯

添加外部 FLASH 接口

根据 **GX-SOPC-EP3C55F484** 核心板上 **FLASH** 芯片 的型号 **AM29LV320DT120** (**2M*16bit**) 来配置 Flash 接口参数。参数设置如下图所示，地址宽度为 21，数据宽度为 16bit。

双击 *Memory and Memory Controllers ->Flash->Flash Memory(CFI)*，外部 flash 接口；
在 Attributes 中，Presets 列表中选择相应的 flash。选择 **custom**。
若所使用 Flash 没有在列表中：自定义宽度和时序

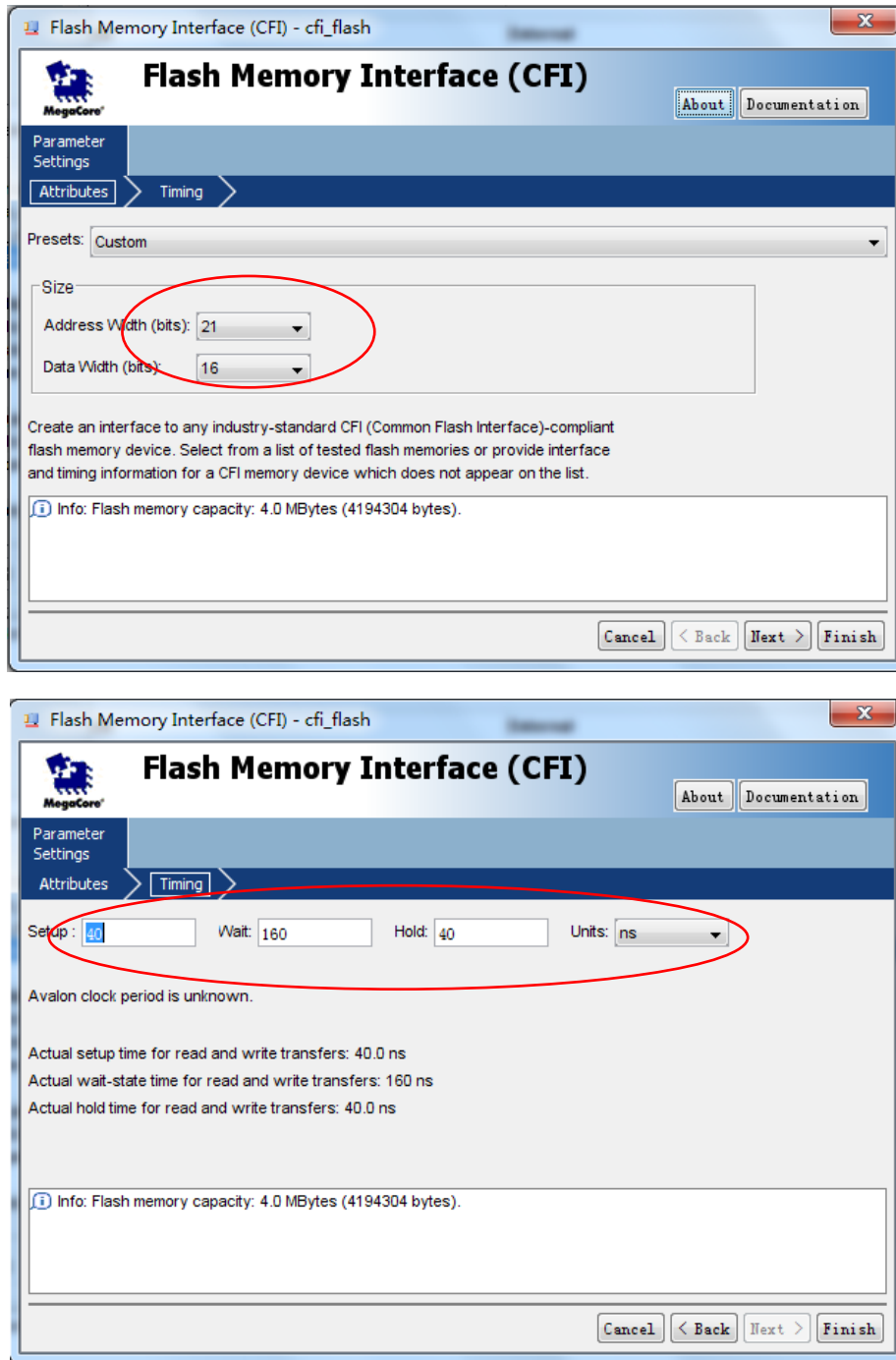


图 15

添加 flash 总线（Avalon 三态总线桥）

接下来，我们要建立一个 Avalon 三态桥，在 NIOS 系统中，要实现与 FPGA 片外

存储器通信，就必须在 Avalon 总线和连接外部存储器的总线之间添加一个桥，这个桥就是 Avalon 三态桥。

双击 Bridges and Adapters -> Avalon-MM Tristate Bridge: 再添加一个 Avalon 三态总线桥，重命名为 tri_state_bridge.

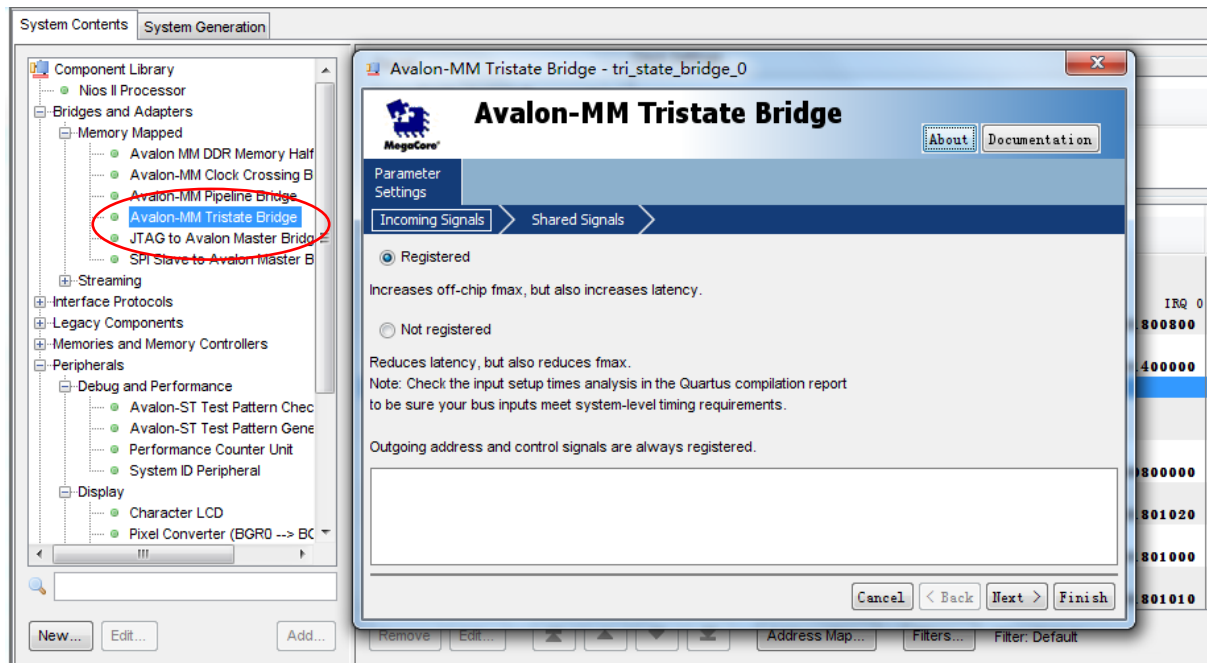


图 16

添加外部 SDRAM 控制器

双击 *Memory and Memory Controllers* -> *SDAM*->SDRAM Controller, 添加一个外部 SDRAM 存储器的接口，按下图进行设置，点 Finish 完成，重命名为 sdram。

根据 GX-SOPC-EP3C55F 核心板上 SDRAM 的型号 K4S641632K 芯片手册，12 个地址线，16 个数据线进行设置。

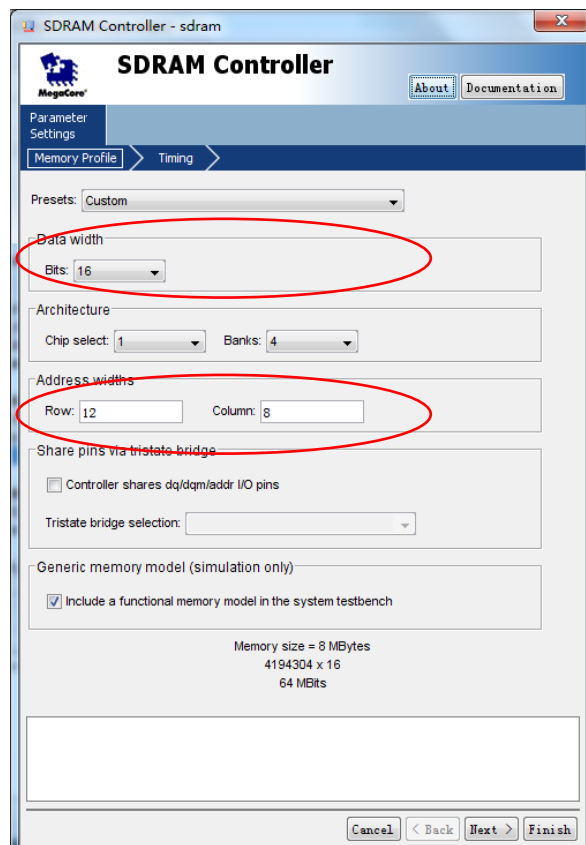


图 17

添加 JTAG UART 串行接口：

在 SOPC builder 中添加组件 JTAG UART。JTAG UART 核是在 PC 主机和 FPGA 上的 SOPC Builder 系统间进行串行通信的一种实现方式。JTAG UART 核在许多设计中用来替代 RS-232 完成与 PC 主机的字符输入/输出。

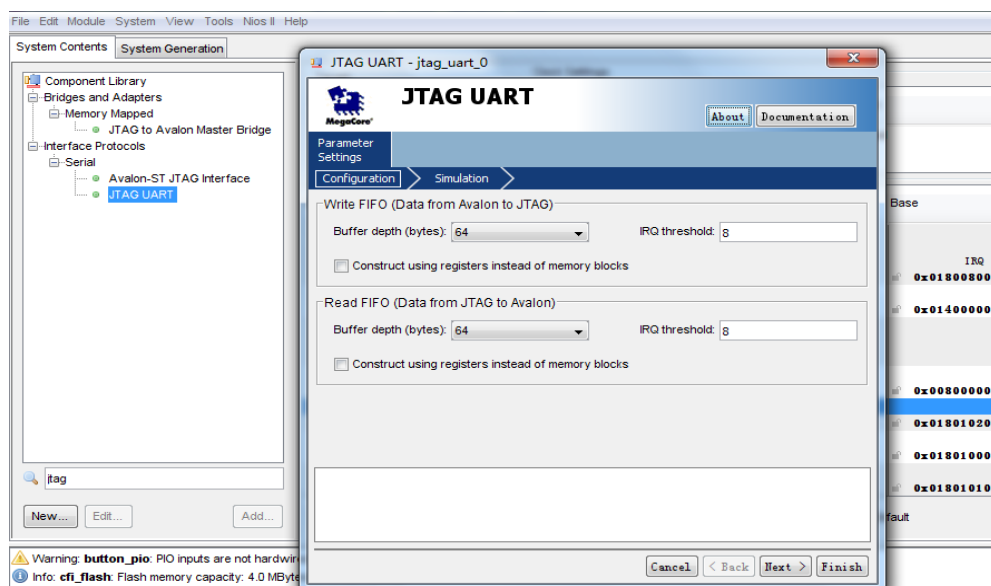


图 18

添加 LED 灯：

双击 Peripherals -> Microcontroller Peripherals -> PIO (Parallel I/O)，8 位，输出口，按下图设置。重命名为 led_pio.

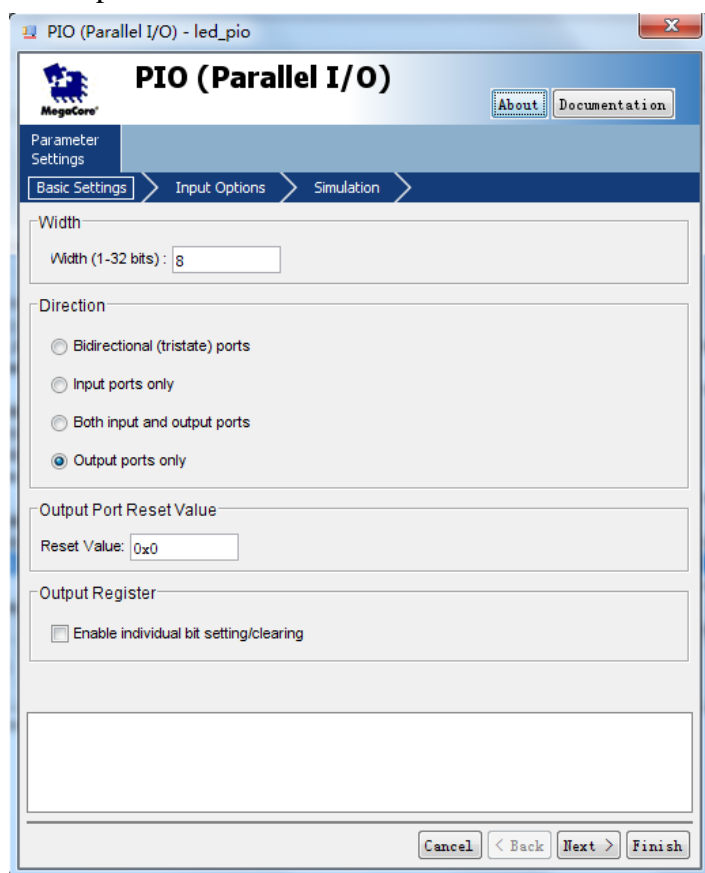


图 19

添加 **Button PIO**

双击 Peripherals -> Microcontroller Peripherals -> PIO (Parallel I/O)，按图中所示设置。

(1) 在 Basic Settings，设 width 为 8，direction 为输入。

(2) 在 Input Options，Edge Capture Register 选中 Synchronously Capture，选择 Either Edge；

(3) 完成后，选择 Rename，重命名为 button_pio。

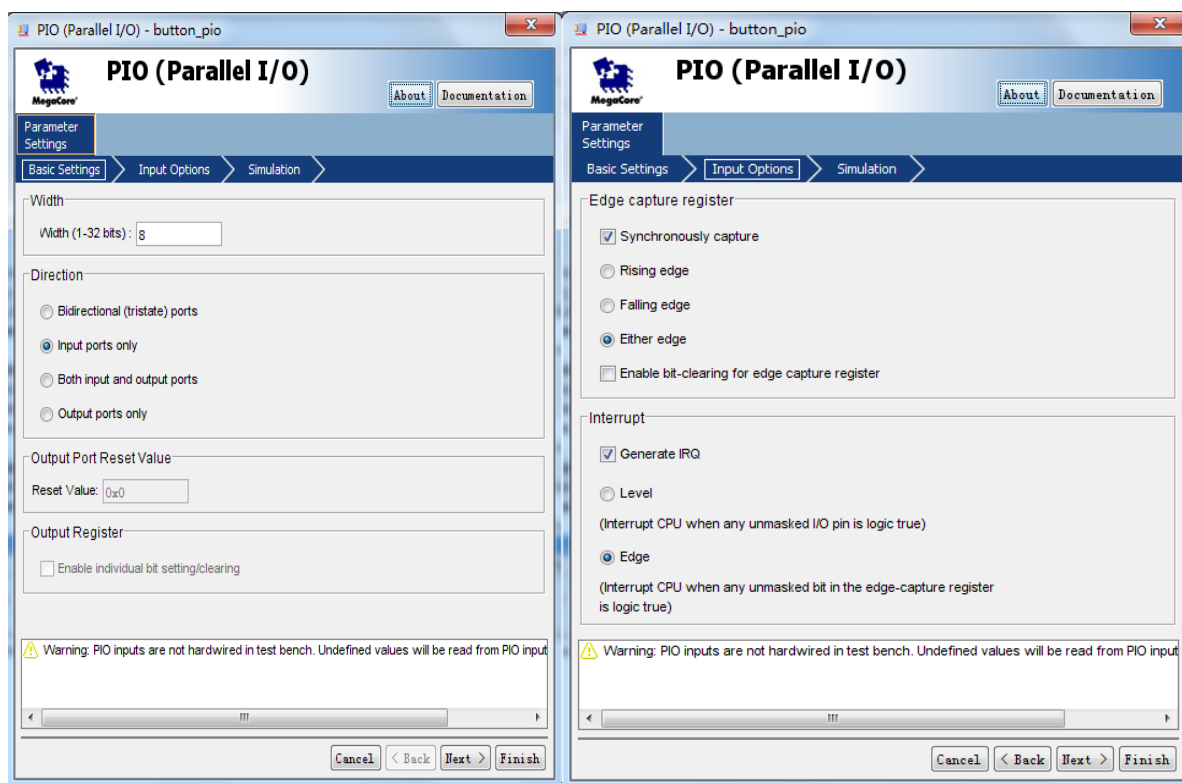


图 20

- ❖ 所有添加的 IP 连接是系统自动完成的。对于三态桥和外部存储器接口的连接，系统的自动连接可能和用户的开发板不匹配，用户需要进行手动的更改。
- ❖ 主要是外部的 **cfi_flash** 的 **s1** 与 **tristate_bridge** 的 **tristate_master** 连接。完成后如图所示。（注意图中将该处的白点点黑，表示正确连好！）

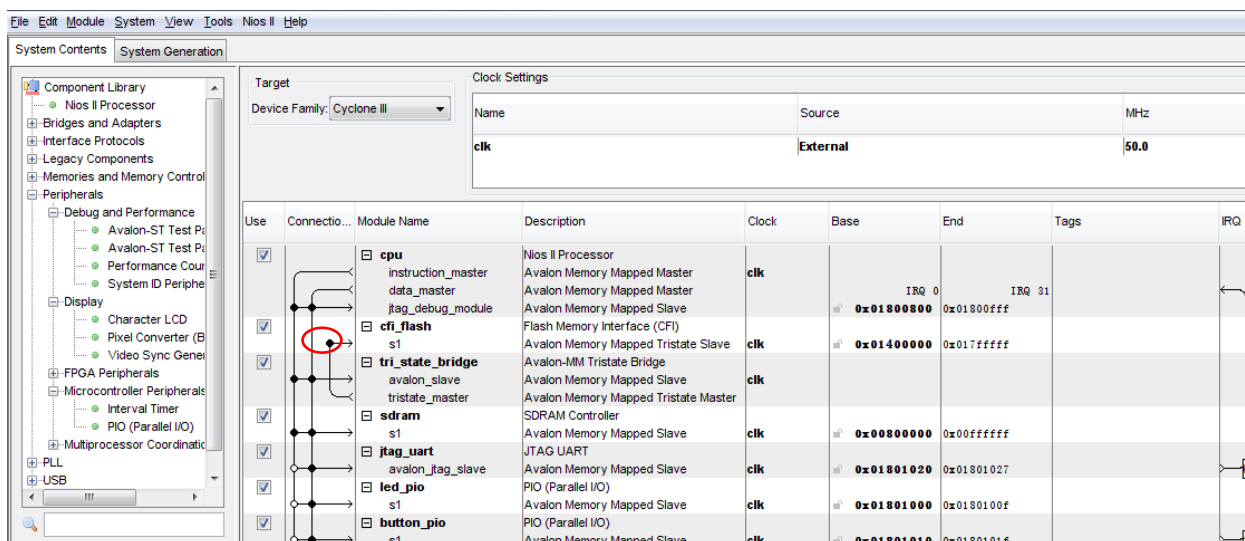


图 21

(4) 自动分配地址和中断号

系统的每个组件都需要一个地址才能正常工作。某些组件，如定时器（Interval Timer）还需要分配一个 IRQ 号。如果发现各组件的地址或者 IRQ 号出现冲突，可以选择菜单栏上 **System -> Auto-Assign Base Addresses** 以及 **System -> Auto-Assign IRQs** 自动设定地址和 IRQ。（注：系统 IRQ 可以是 0 到 31 的整数，数值越小优先级越高。

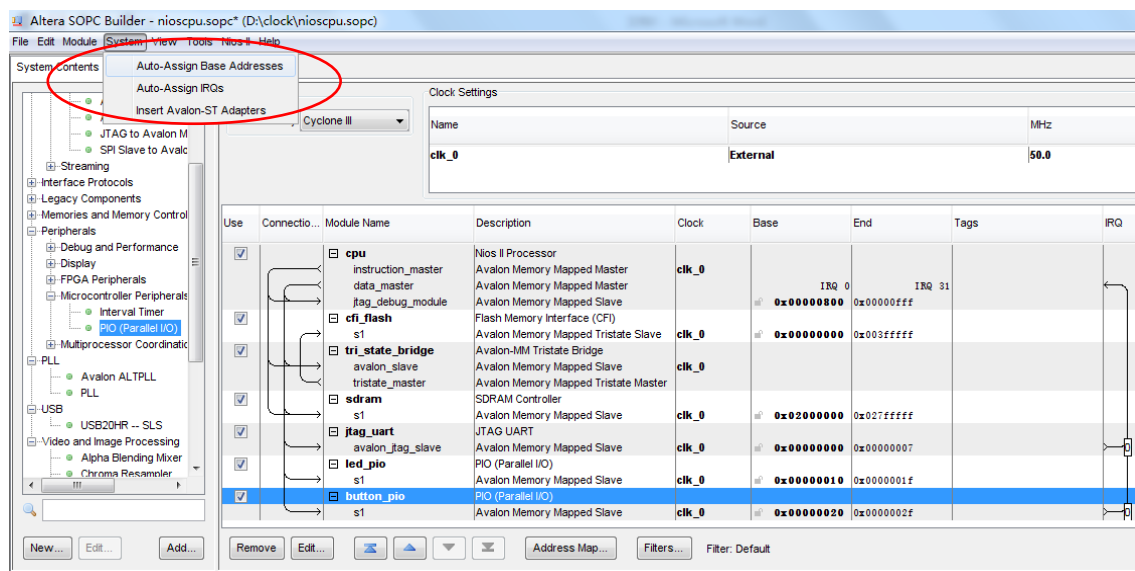


图 22

(5) 配置 NIOS II 系统

双击 CPU，按照下图配置。

Reset Vector: 选择存放 Boot Loader 的存储器和设置 Boot Loader 在存储器中的偏移。
选 cfi_flash，偏移选择默认；

Exception Vector: 选择存放异常向量表存放的存储器和异常向量表在存储器偏移,选择 edram，偏移选择默认；

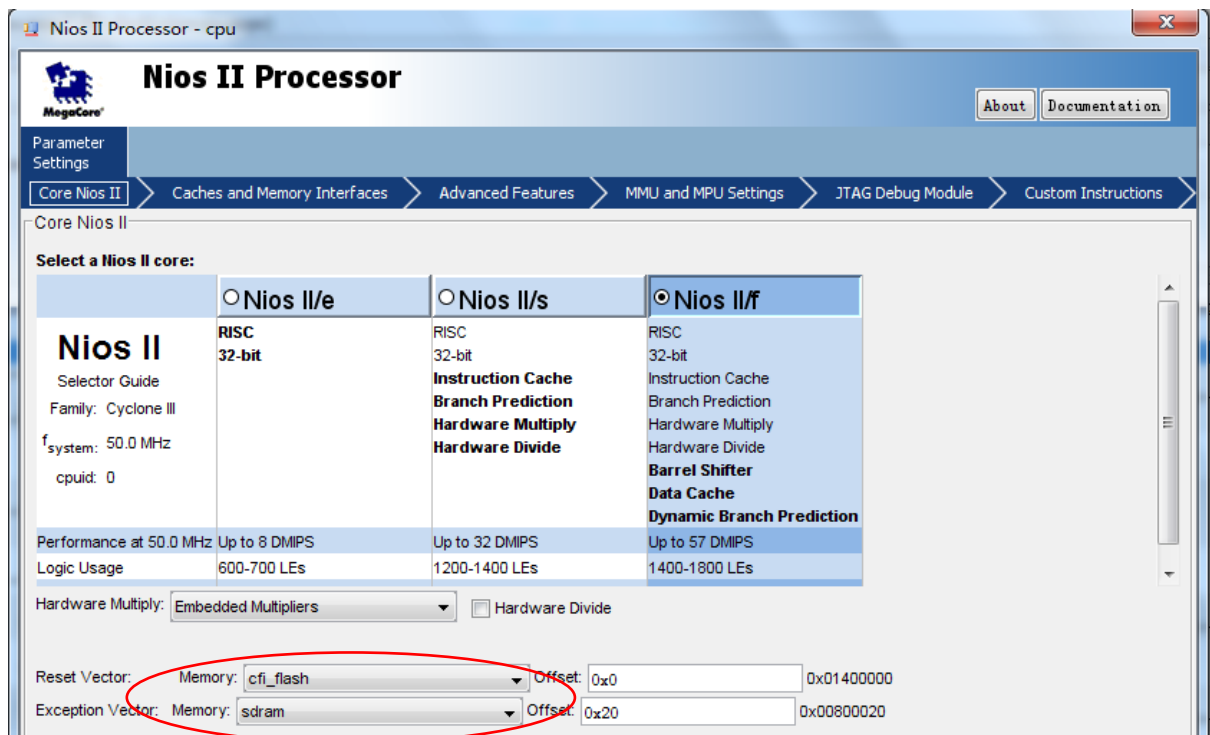


图 23

(6) 生成 NIOS II 并加入到工程中

至此系统已经构造完毕，点击下方 **Generate** 生成系统。程序将提示需要先保存，选是即可。大约需要 1~2 分钟，出现以下画面说明 SOPC 系统已经生成完毕。

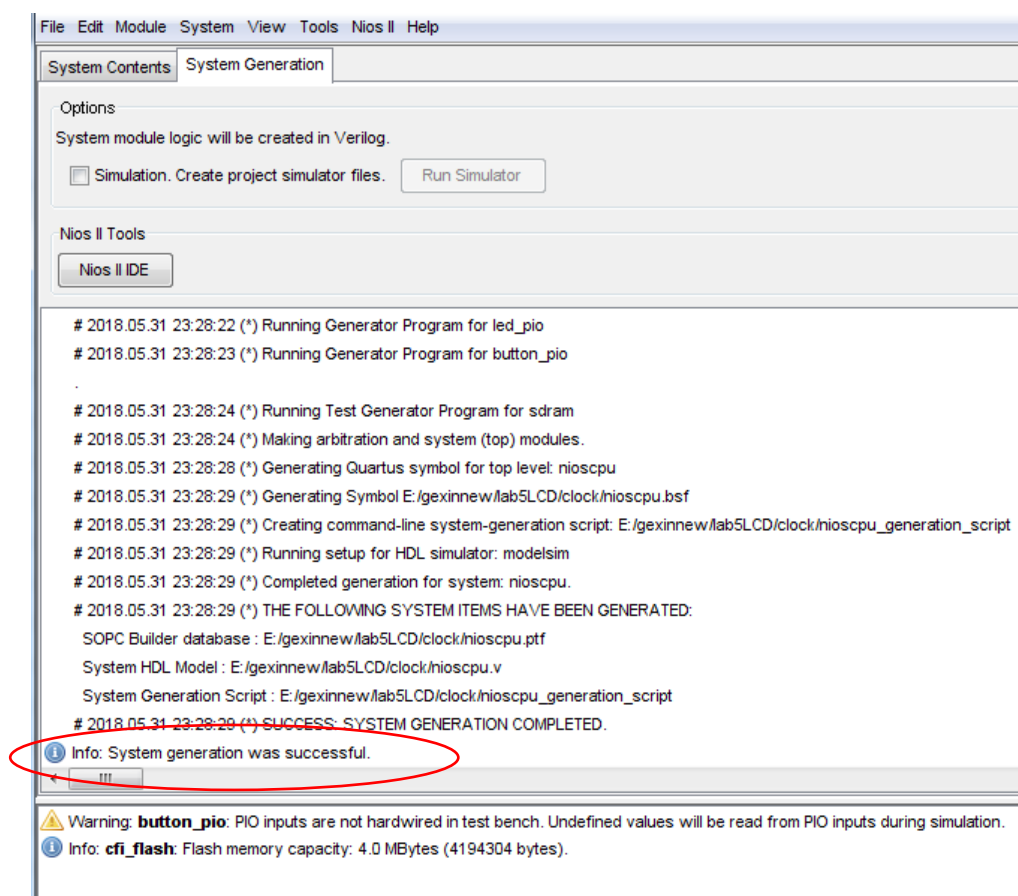


图 24

系统生成完成后，SOPC Builder 为这个定制的 NIOS II 系统模块创建了一个符号，Nios II 系统再加入到工程中：

打开顶层实体（BDF），任意处双击，出现 Symbol 对话框

在 Symbol 对话框中单击 Project 来展开工程目录，其下出现 nioscpu，选中它，右侧出现了系统的符号表示；

点 OK，NIO2 出现在 BDF 窗口中，创建的系统加入到工程

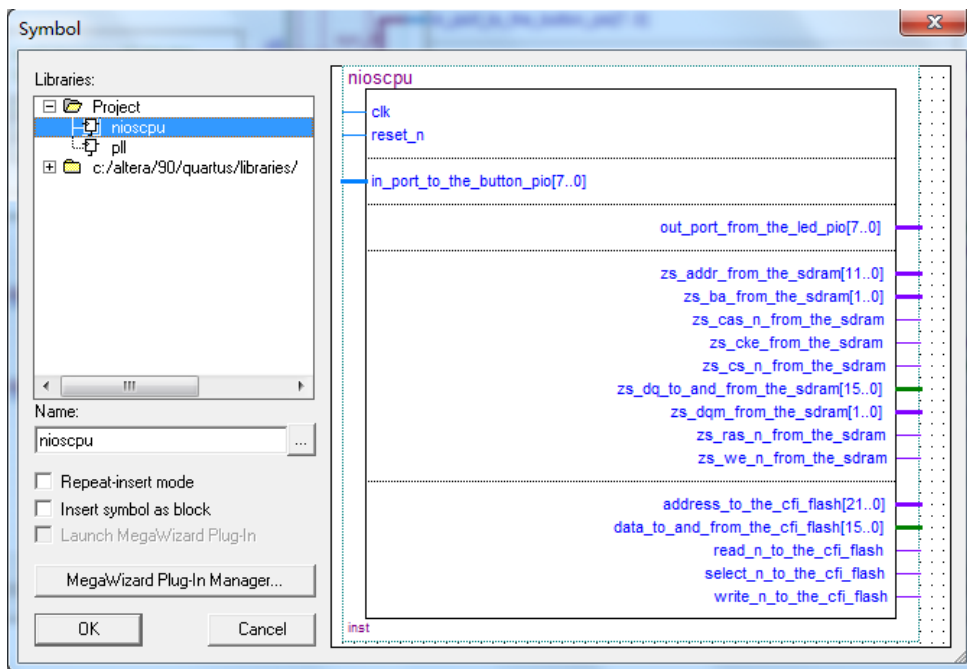


图 25

(7) 加入嵌入式锁相环

- ❖ 嵌入式锁相环有两个时钟输出，一个输出 SDRAM 提供时钟，另一个时钟的输出为 NIOS II CPU 提供时钟。
- ❖ 加嵌入式锁相环步骤如下：

点击 Tools→MegaWizard Plug-In Manager，出现 MegaWizard Plug-In Manager 向导窗口,点击 next

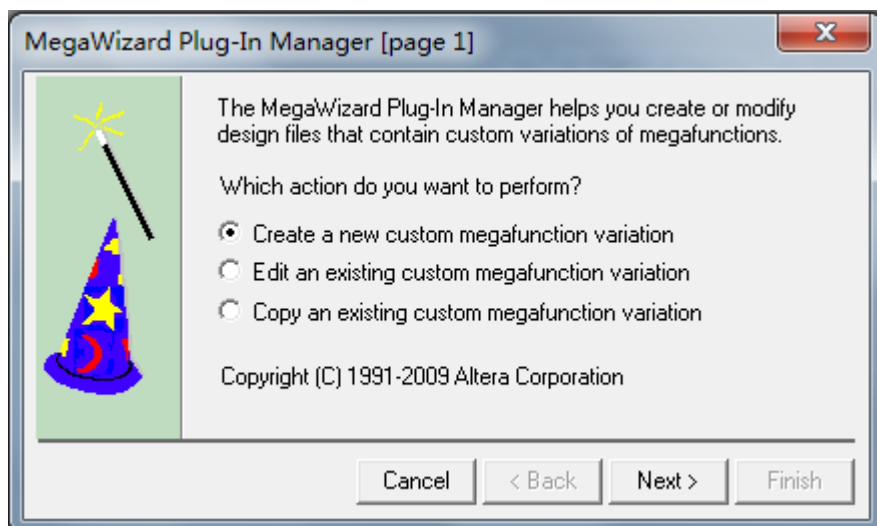


图 26

I/O 下面选择 ALTPLL，器件选择 Cyclone III，输出文件类型选择 verilog，文件名为 pll.v

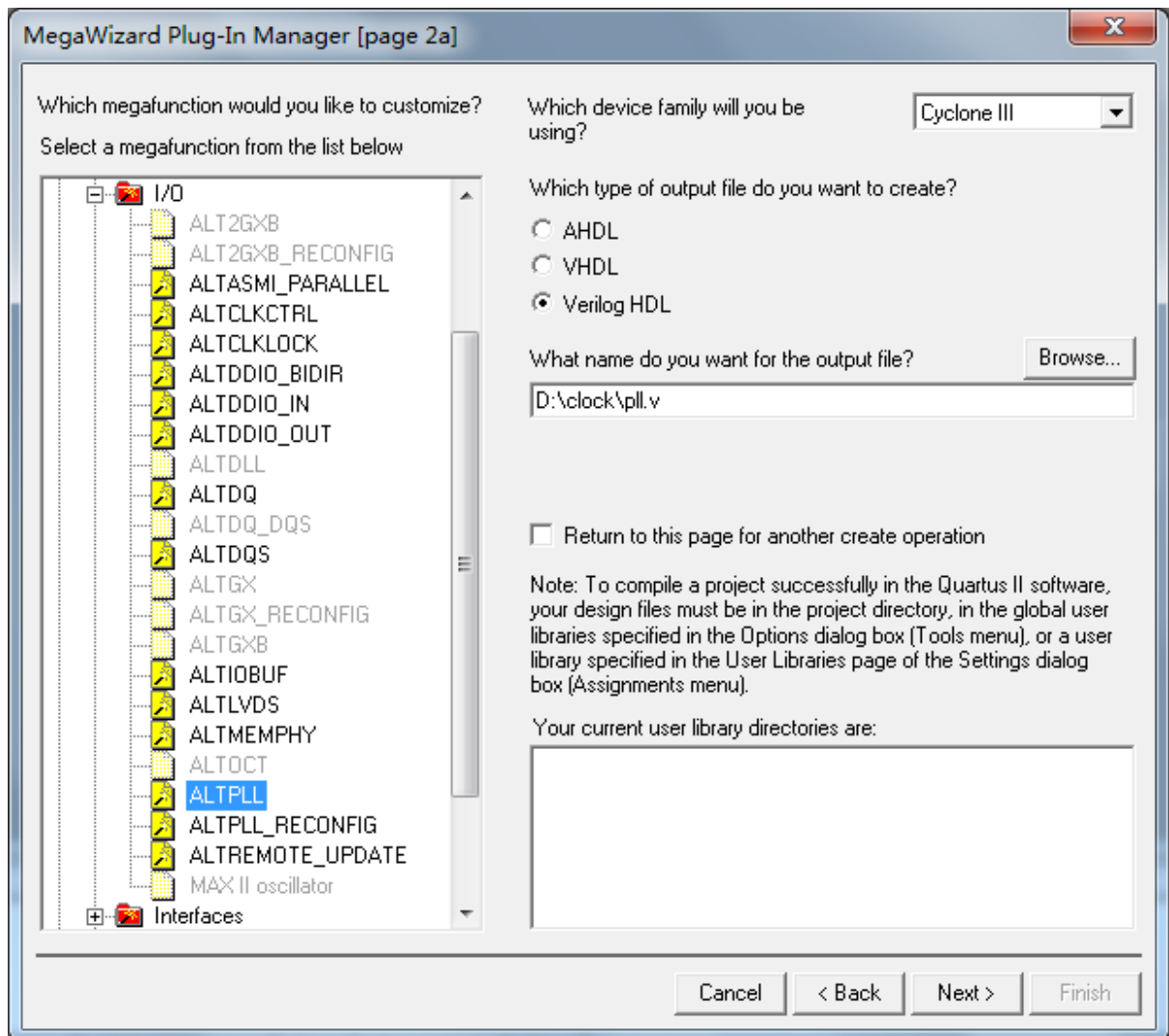


图 27

点击 next，按照下图进行设置

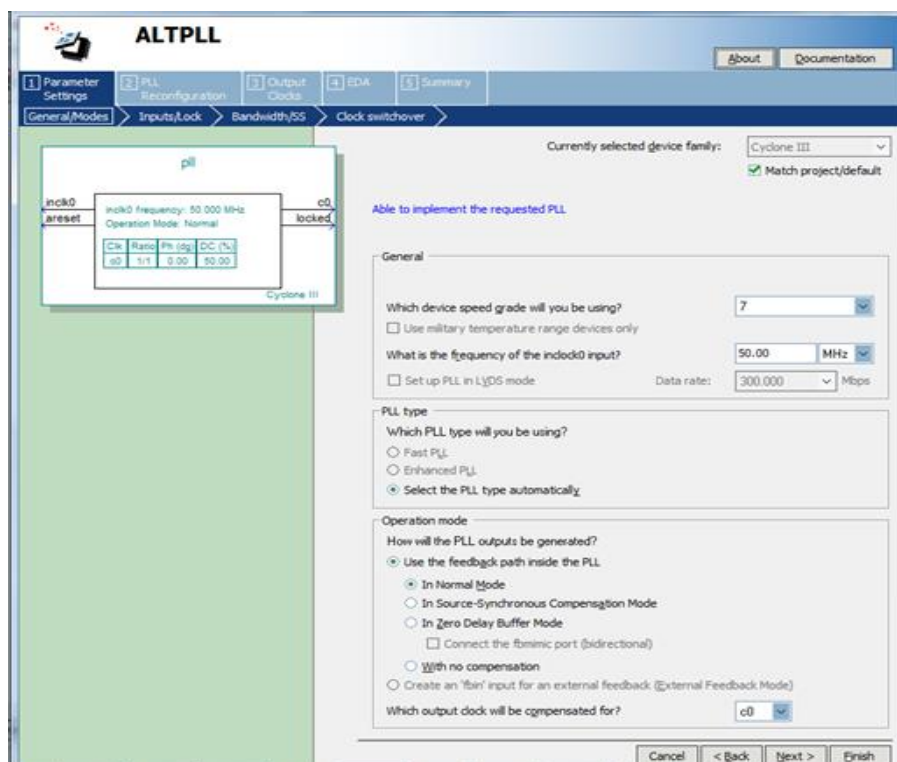


图 28

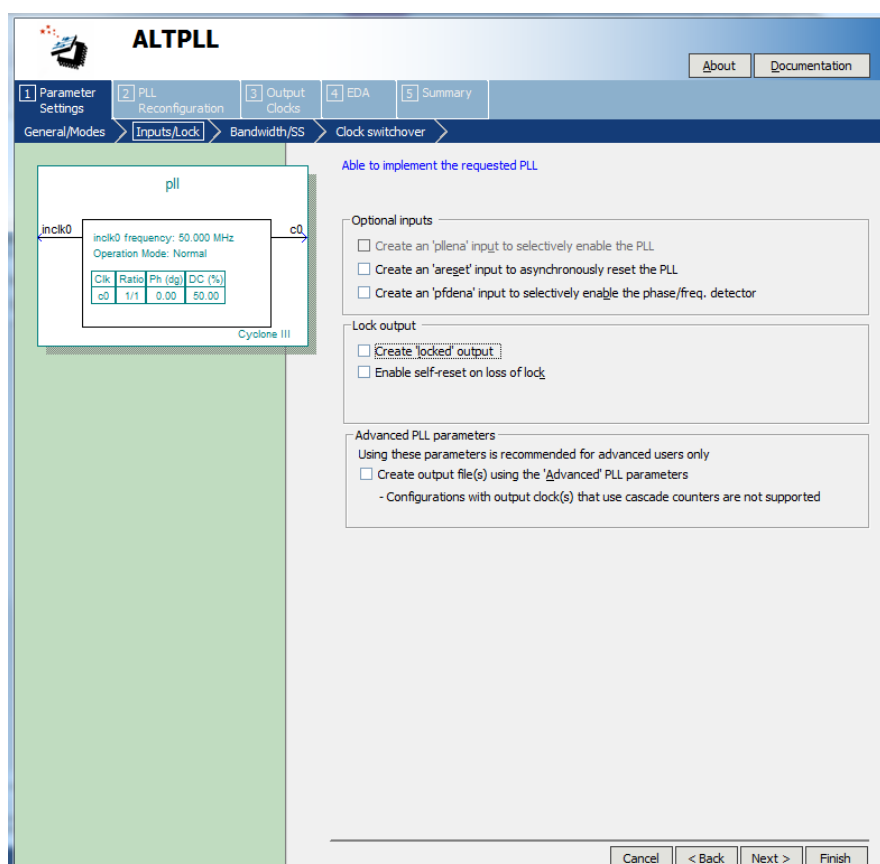


图 29

点击 Next, (或者选择选项页 3) 将 C0 的频率设为 50MHZ

The screenshot shows the ALTPLL configuration tool interface. The top navigation bar includes tabs for Parameter Settings, PLL Reconfiguration, Output Clocks (selected), EDA, and Summary. Below this, a sub-navigation bar shows clock selection options: clk c0, clk c1, clk c2, clk c3, and clk c4. The main area is divided into two panes. The left pane displays a block diagram of the PLL with an input clock 'inclk0' and an output clock 'c0'. A table within the diagram shows the configuration for clock 'c0':

Clk	Ratio	Ph (deg)	DC (%)
c0	1/1	0.00	50.00

The right pane is titled 'c0 - Core/External Output Clock' and includes a status message 'Able to implement the requested PLL'. It features a 'Use this clock' checkbox which is checked. Below this, the 'Clock Tap Settings' section contains several parameters with 'Requested settings' and 'Actual settings' columns:

Parameter	Requested settings	Actual settings
Enter output clock frequency:	50.00000000 MHz	50.000000
Enter output clock parameters:		
Clock multiplication factor	1	1
Clock division factor	1	1
Clock phase shift	0.00 deg	0.00
Phase shift step resolution(ps)		
Clock duty cycle (%)	50.00	50.00

At the bottom right, there is a 'Per Clock Feasibility Indicators' section with a row of indicators for clocks c0, c1, c2, c3, and c4. The 'c0' indicator is highlighted in green.

图 30

单击 next 进入 C1 输出时钟的设置。将 C1 的频率设为 50MHZ。

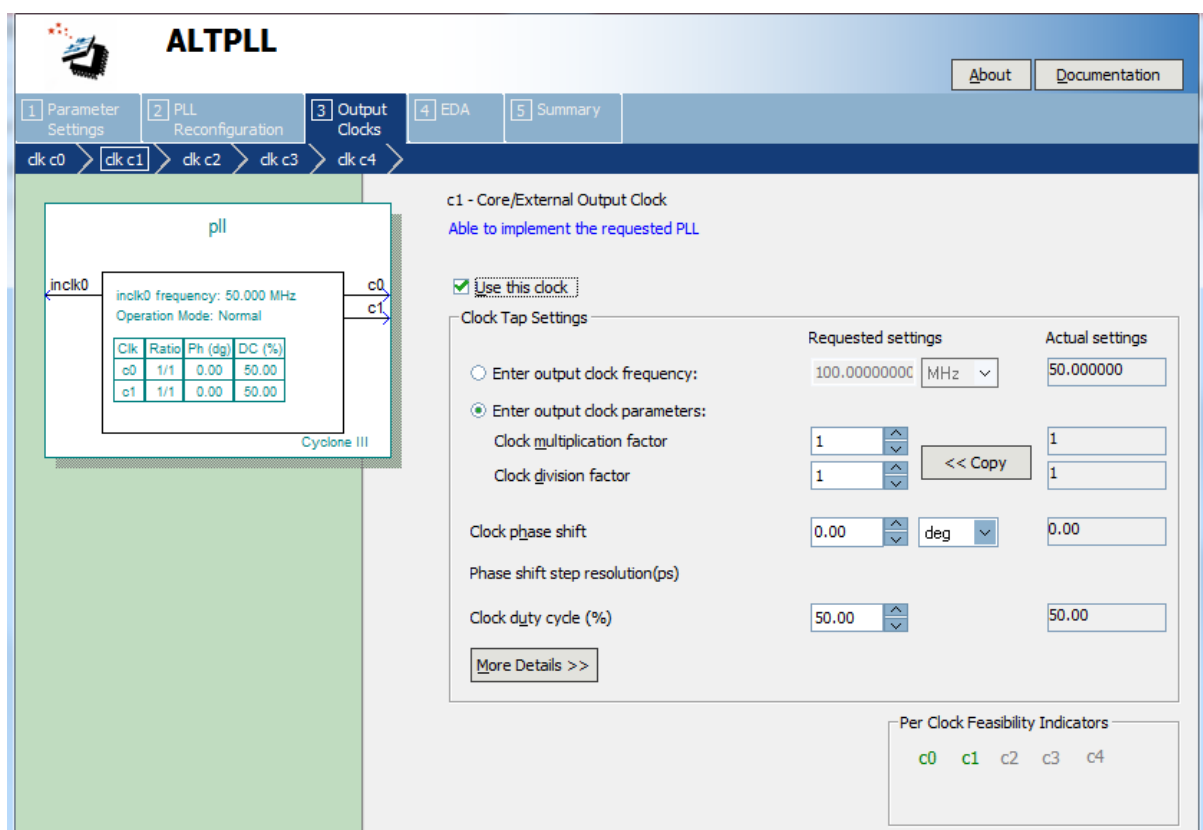


图 31

单击 next，不用设 C2

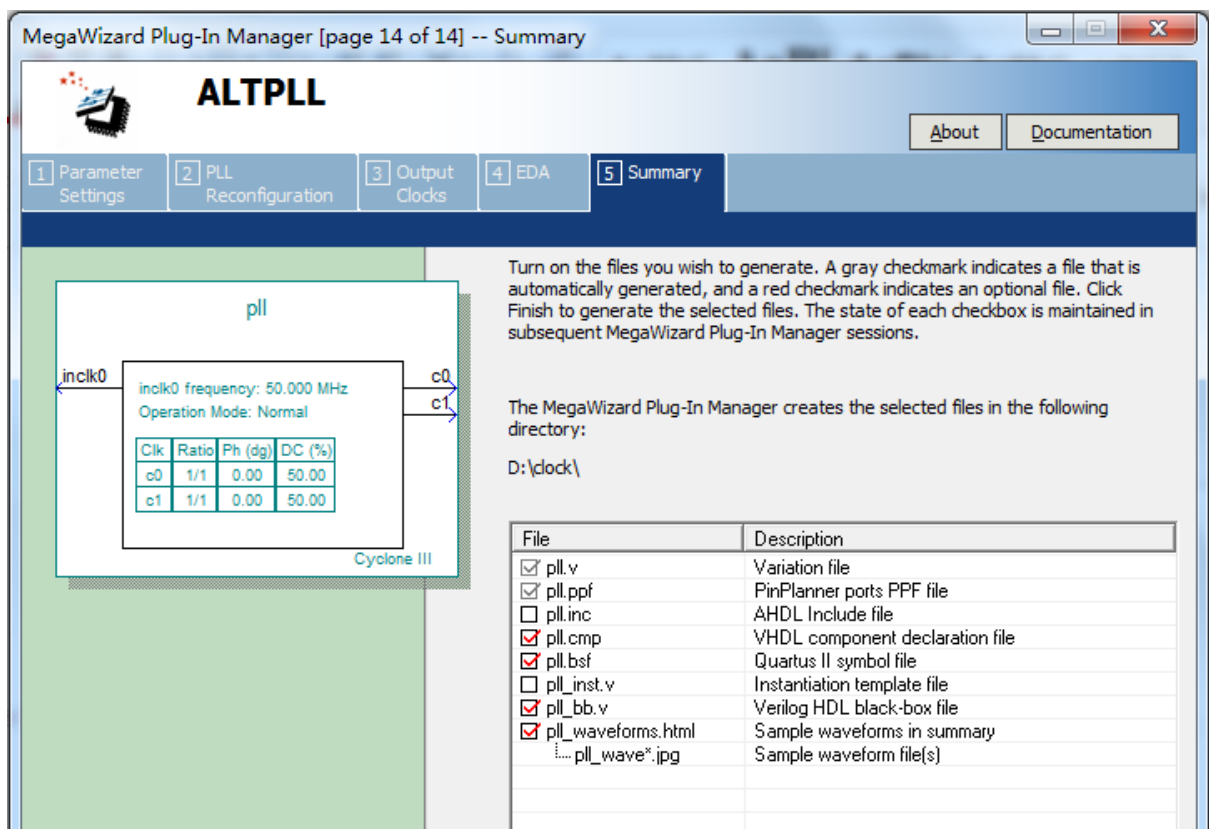


图 32

点击 Finish 完成 PLL 的生成。

在顶层实体的 bdf 窗口中双击鼠标，出现 Symbol 添加窗口，在 project 下面选择刚才建立的 pll，点击 OK。点击 bdf 窗口的空白处，将嵌入式锁相环加入到了工程中。

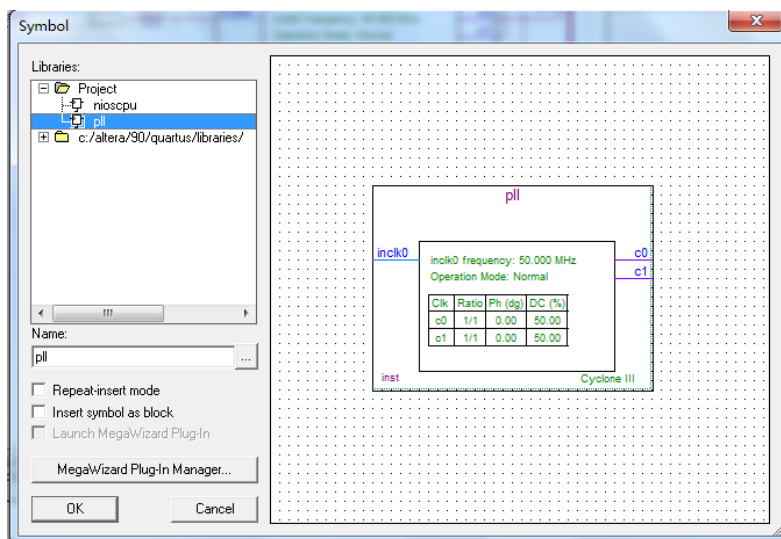


图 33

(8) 加入引脚

顶层实体的 bdf 窗口的空白处双击鼠标，出现 Symbol 添加窗口，选择 pin，选择相

应类型的引脚，(输入、输出和双向)，点击 OK。

点击 bdf 窗口的空白处，即将引脚加入到了工程中。

重复上面的步骤添加为各个端口添加相应类型的引脚。

命名引脚方法：双击引脚的“pin_name”，对其编辑。对于总线型的引脚，引脚名称[总线位数]，如 ddr_a[12..0]。

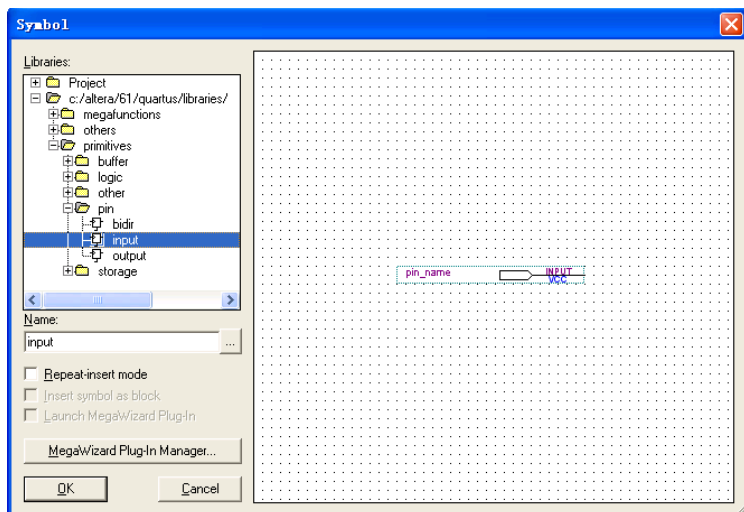


图 34

将嵌入式锁相环和系统模块等连接起来，将引脚连接到相应的端口上。

特别注意：

1. sdrn 和 flash 的数据线是双向口(颜色是绿色)，因此引脚选择 BIDIR，避免后面无法对数据进行读写。
2. 引脚命名完全按照图中命名，因为后续采用脚本文件来添加引脚。

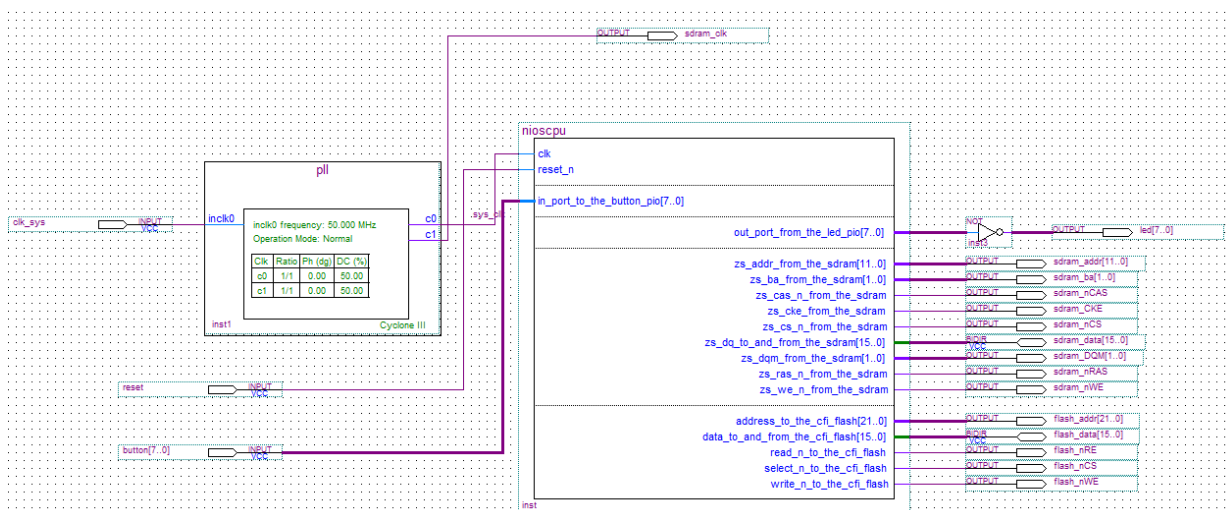


图 35

4. 设计优化

设计优化：节省占用 FPGA 的面积和提高系统速度。

- ❖ 选择 Assignment 菜单下的 Settings 命令
- ❖ 在 Analysis & Synthesis Settings: Optimization Techniques 栏中, 有 Speed、Balanced 和 Area 3 种优化选择, Balanced 是软件缺省的优化选择。

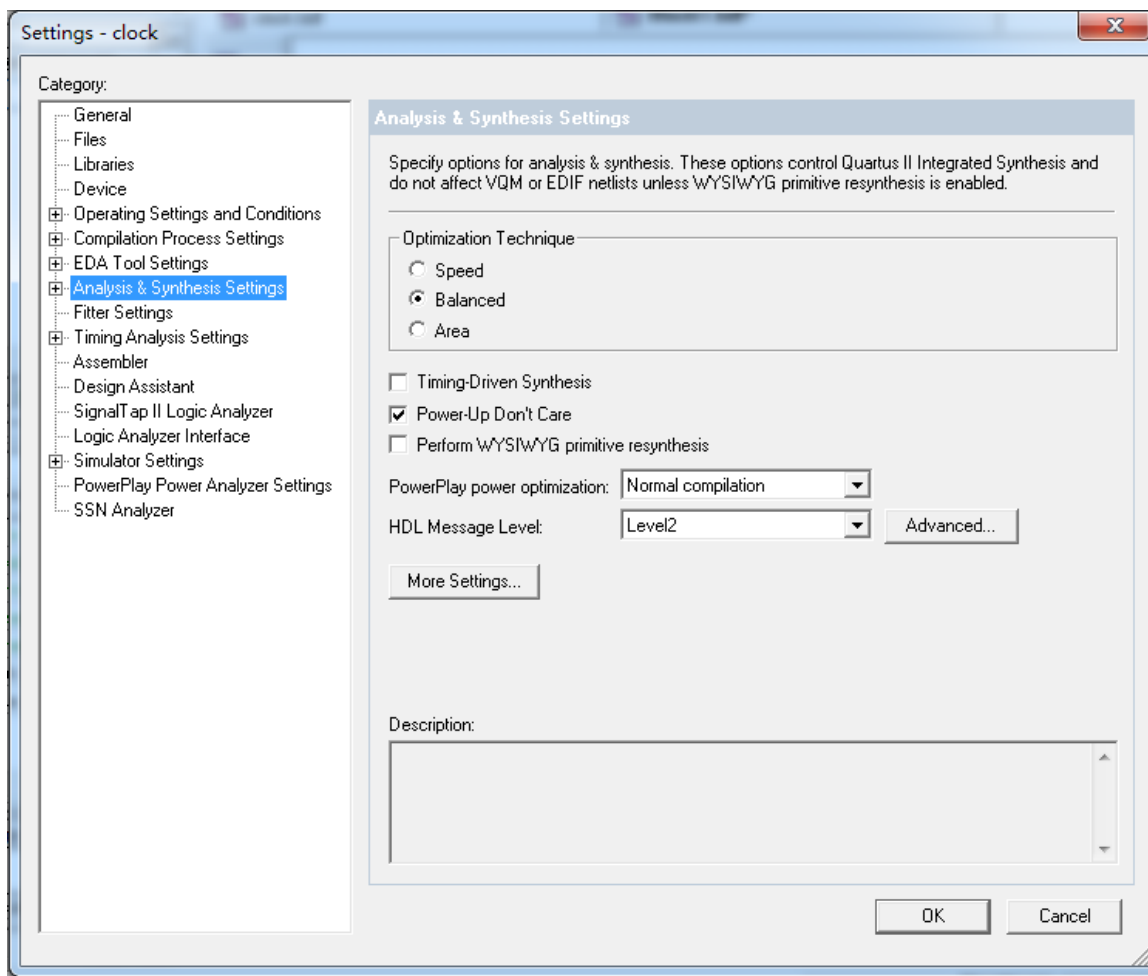


图 36

5. 编译

5.1 编译设置：合理编译的设置可以提高工程编译的速度，优化器件的资源利用，甚至降低系统的功耗!编译之前须对未使用的引脚做设置：（**注意：将未使用的引脚设置成 As inputs, tri-stated.**）

在 Settings 对话框中的 Device 中，单击 Device and Pin Options，出现 Device and Pin Options。

单击 Unused Pins，在 Reserve all unused pins 下选择，As inputs tri-stated。

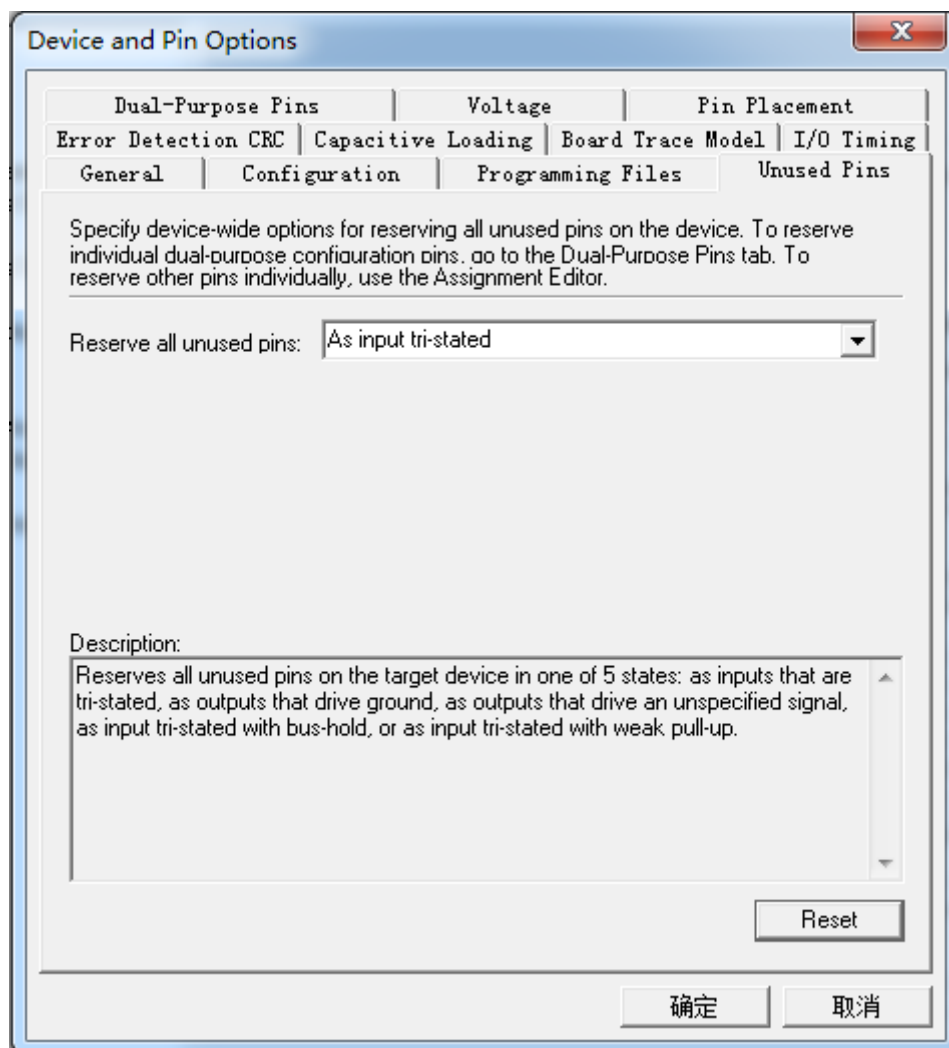


图 37

5.2 引脚分配

引脚分配：使编译器能把设计的信号分配到目标器件上的特定引脚上。

两种引脚分配方法：

- ❖ 第一种：使用 Assignments Editor 或 pins 或 pin planner

这种分配方法效率较低，适合较少管脚的分配。

- ❖ 第二种：使用 TCL 脚本一次性分配所有的引脚，引脚多时采用这种方法，节省时间。

注意：将 clock.tcl 拷贝到 clock 工程文件夹。

Tools 菜单下，选 TCL script，出现下图

选择引脚分配文件 clock.tcl，点 RUN 即可完成引脚分配。

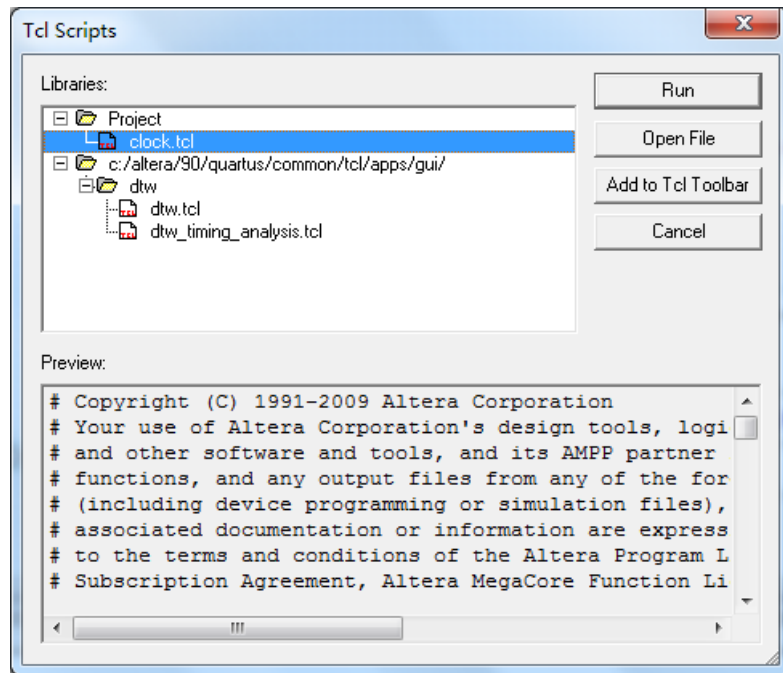


图 38

脚本文件中没有的引脚可以手动添加，也可以添加到 TCL 脚本文件中。

5.3 编译用户设计

- ❖ 选择 Processing 菜单中的 Start Compilation
- ❖ 或点击工具栏中的 Start Compilation 按钮进行编译



图 39



图 40

编译成功之后，出现 Compile Report.

点击相关内容查看相应的编译信息，如 Analysis & synthesis、Fitter、Timing Analyzer 等。

6. 编程下载

❖ 编译成功之后，Quartus II 编译器生成配置

两个文件: .sof 和.pof

❖ .sof 文件一般在调试时下载到 FPGA 的 SRAM 中

❖ .pof 文件是用于 EPCS 的编程文件。

❖ 实验平台采用的下载线为 USB Blaster。

下载:

将生成的.sof 文件下载到 FPGA 中:

(1) 同过 USB Blaster 电缆将目标板和计算机相连，接通目标板的电源。

(2) 在 Quartus II 软件中选择 Tools→Programmer，打开编程器的窗口，可以看到配置文件 clock.sof。

选中 Program/Configure，然后点击 start 开始按钮，开始下载，可以从 Progress 栏看到下载进度。

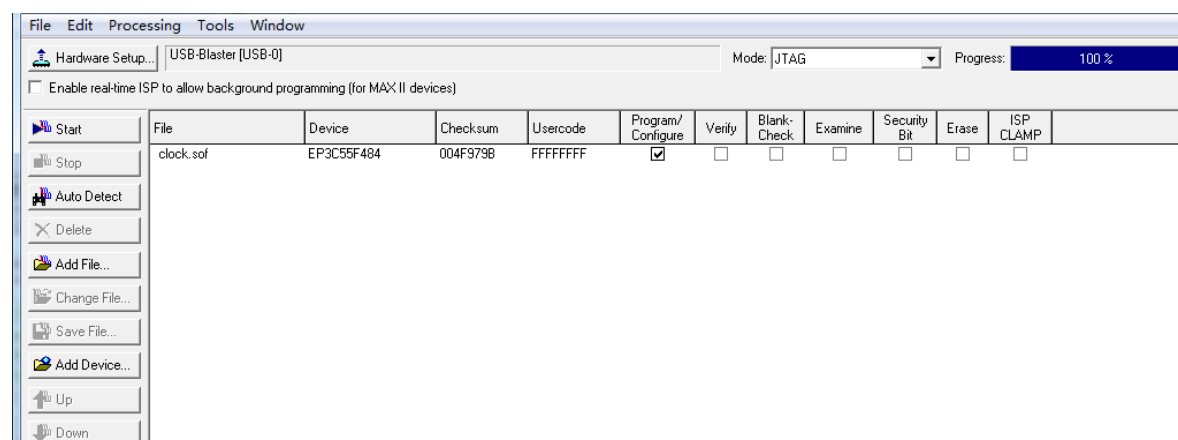


图 41

整个硬件系统就设计完成！但是此硬件系统未加入任何逻辑，因此，还看不出效果。

第二部分：软件系统设计：

测试 JTAG_UART 显示功能：

1、在 Quartus II 软件中，打开 clock 工程

2、双击 NIOS II IDE 图标,

(实验室环境下安装了 9.0 和 15.0 两个版本，如果出现以下环境设置提示信息，请选择“是”。如果在编译过程的错误提示中出现“15.0”，就是因为此处错误选择了“否”。)



第一次打开的时候会提示选择工作空间。

也可在程序打开后选择菜单栏 File -> Switch Workspace...

选择 <工程所在目录>\software 作为 NIOS II 的工作空间。确认以后软件会重新启动。

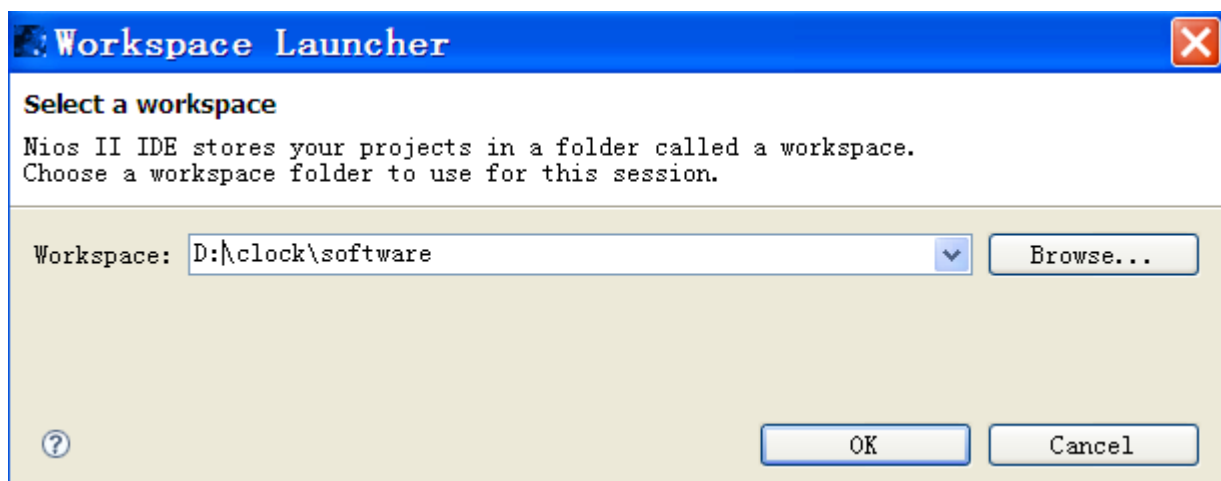


图 42

选择 <工程所在目录>\software 作为 NIOS II 的工作空间。确认以后软件会重新启动。

在欢迎界面中选择 Workbench，



图 43

进入主界面

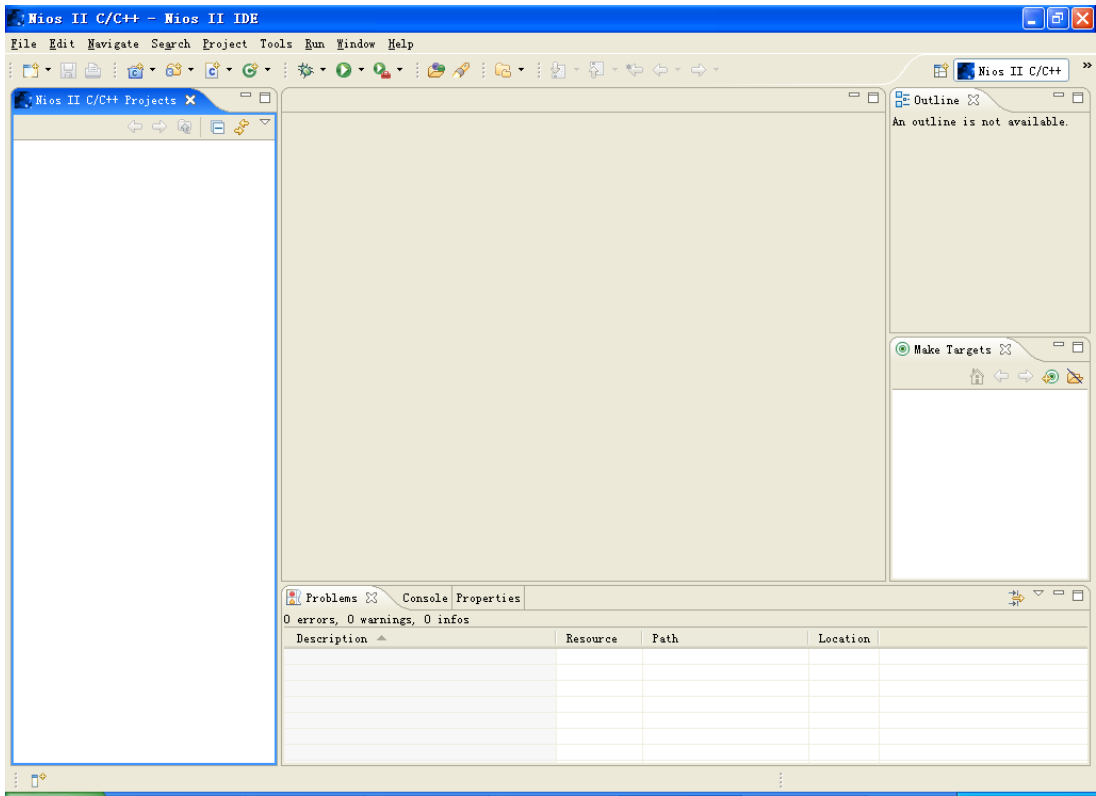


图 44

在 NIOS II IDE 软件环境中点击 New->Nios II C/C++ Application,建立一个工程

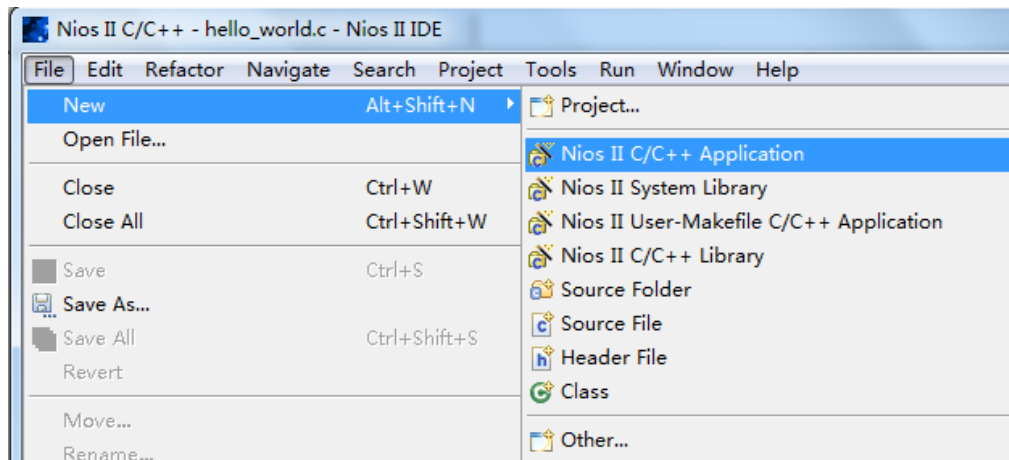


图 45

在标准板中选择 Hello World,

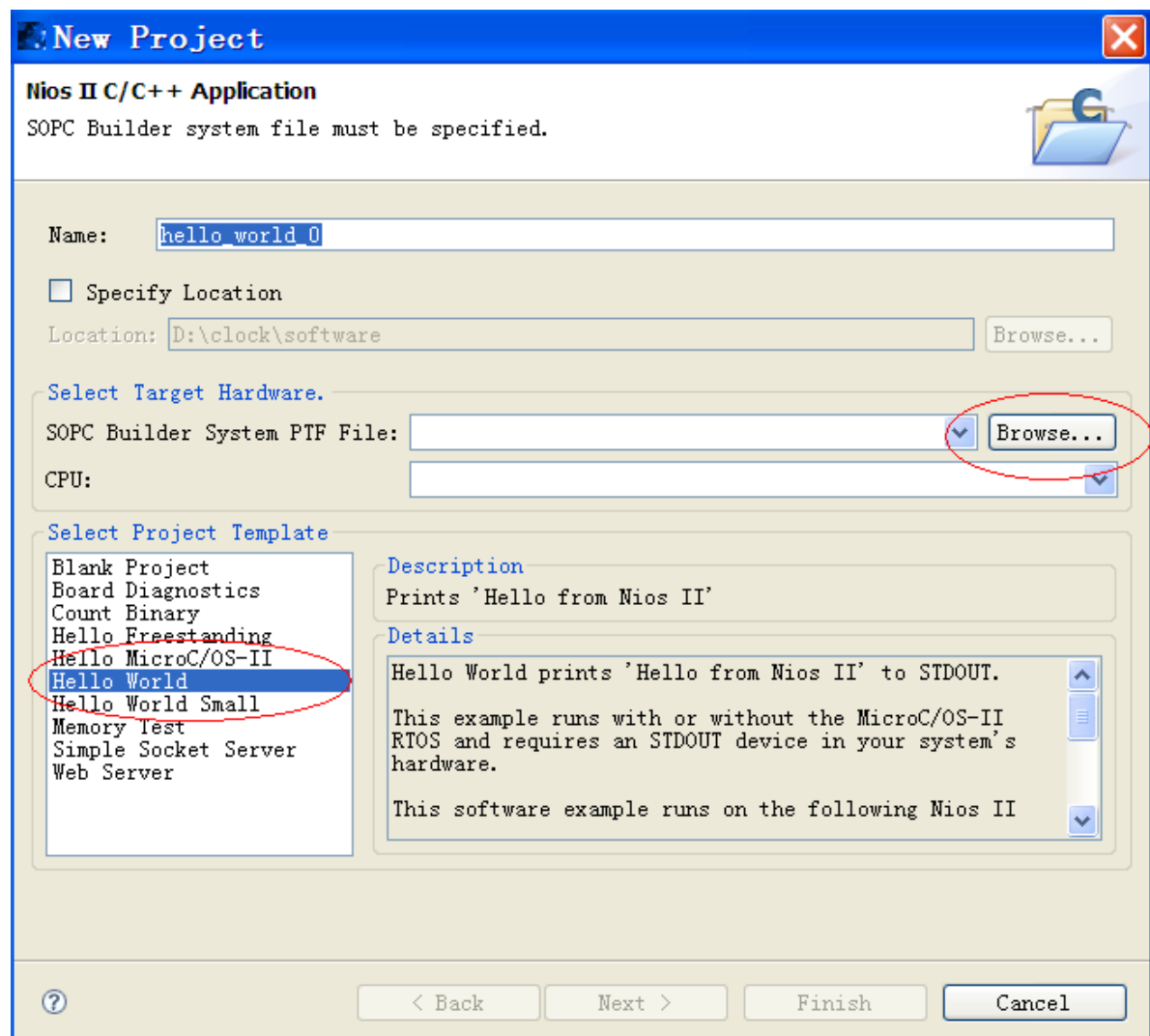


图 46

在 SOPC Builder System PTF File 点击 Browse，选择 nioscpu.ptf

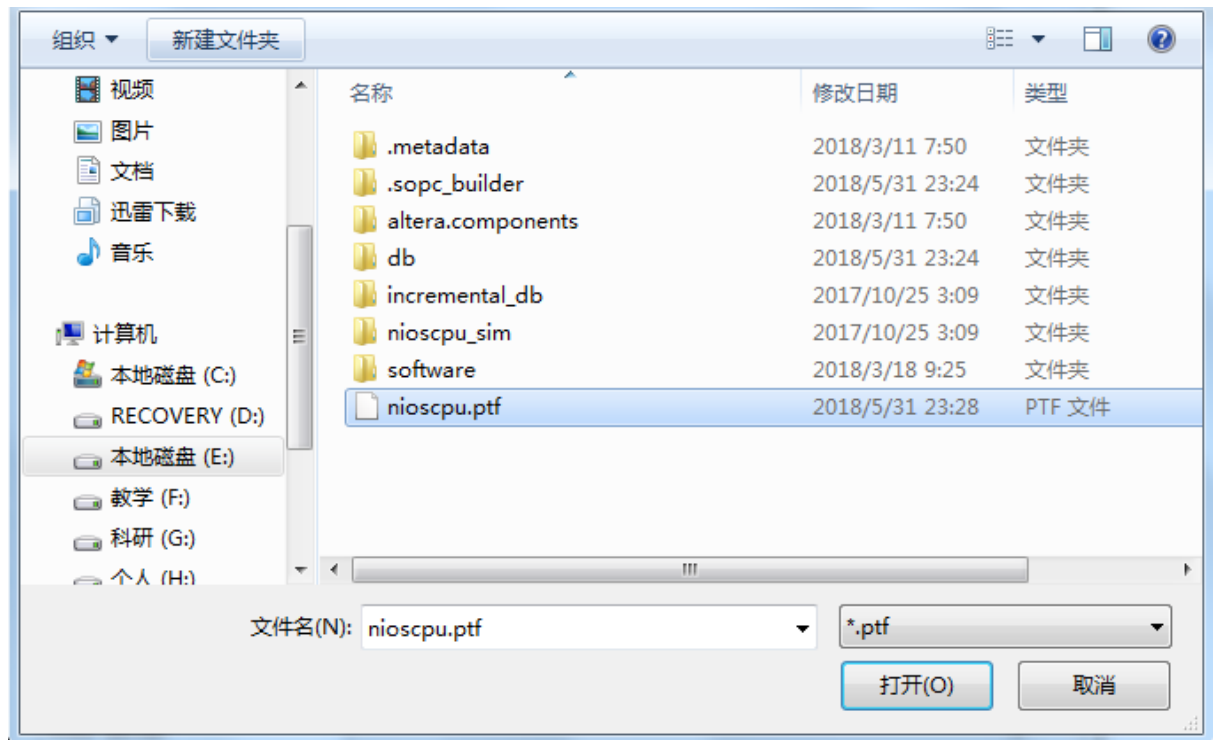


图 47

C/C++的应用工程名 Name 为 hello_word_0，点击 Next

点 Next >，在下一个界面中选择第一项，Create a new system library，点击 Finish 完成。

此时，Nios II IDE 左侧工程列表将多出 3 个工程

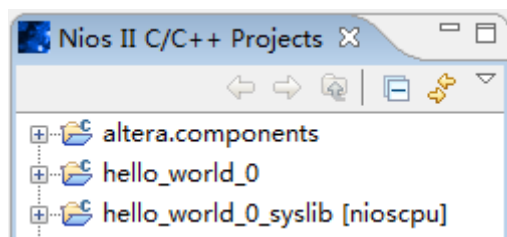


图 48

软件程序如下：

hello_world.c 文件

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    printf("Hello from Nios II!\n");
```

```
return 0;
}
```

修改后，保存文件。

右键单击工程 `hello_world_0_syslib` -> Properties，将 `stdout`、`stderr` 和 `stdin` 设置成 `jtag_uart`，即 `jtag_uart` 被设置成为一个标准输出设备，支持 `printf` 函数。

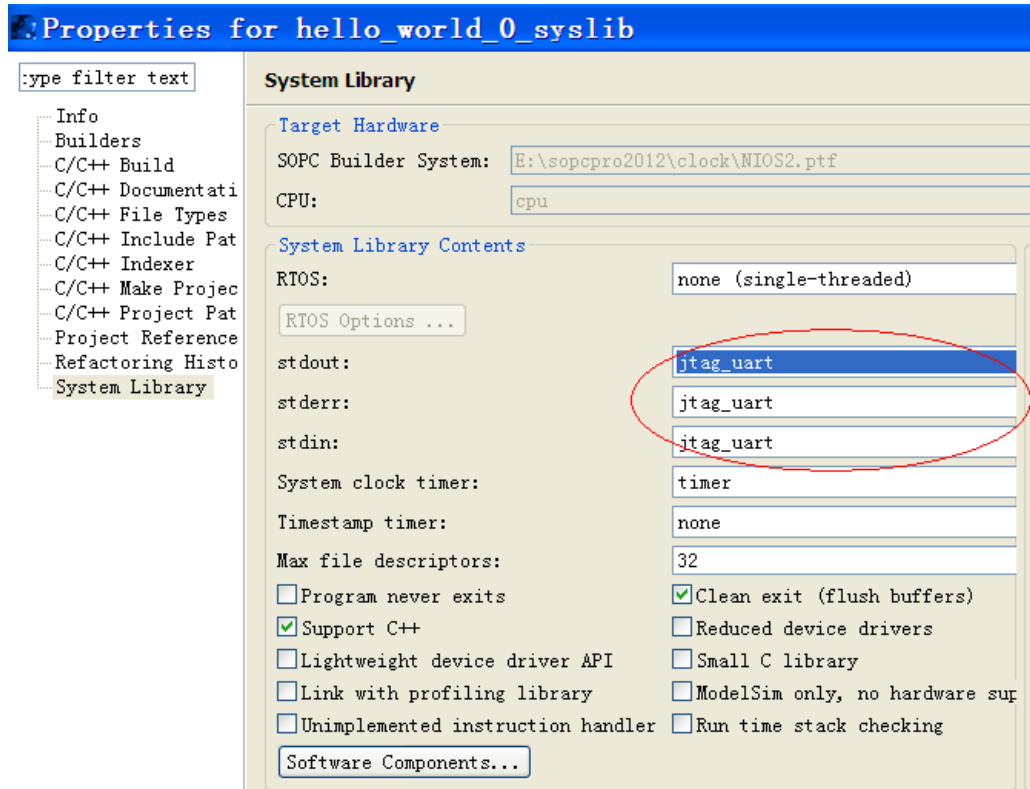


图 49

3、鼠标选择 `hello_world_0`，点击鼠标右键，进行编译工程

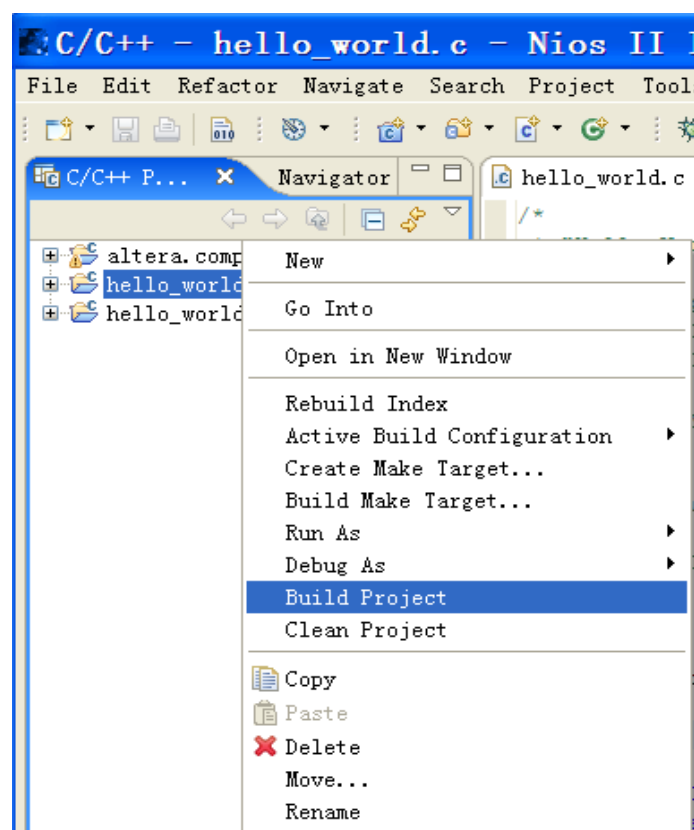


图 50

- 失败：编译的错误和警告，在 console 窗口给出，根据该信息修改代码。
- 成功：出现 Binaries 目录，及一个可执行文件 hello_world_0.elf。是能在 Nios II 处理上运行的可执行和连接格式（Executable And Linked Format File—.elf）

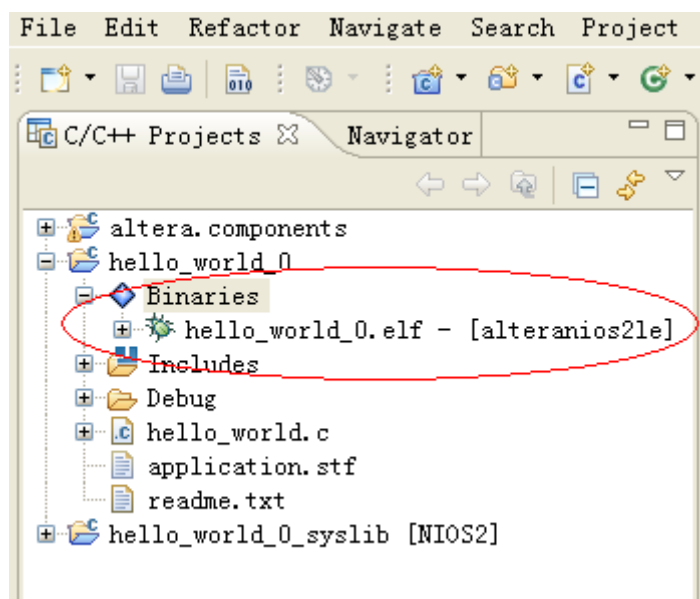


图 51

4、软硬分别下载：

在 Quartus II 软件中，下载硬件工程 “clock.sof”

然后到 NIOS II IDE 软件环境， 选择菜单栏 Run -> Run...， 出现以下对话框

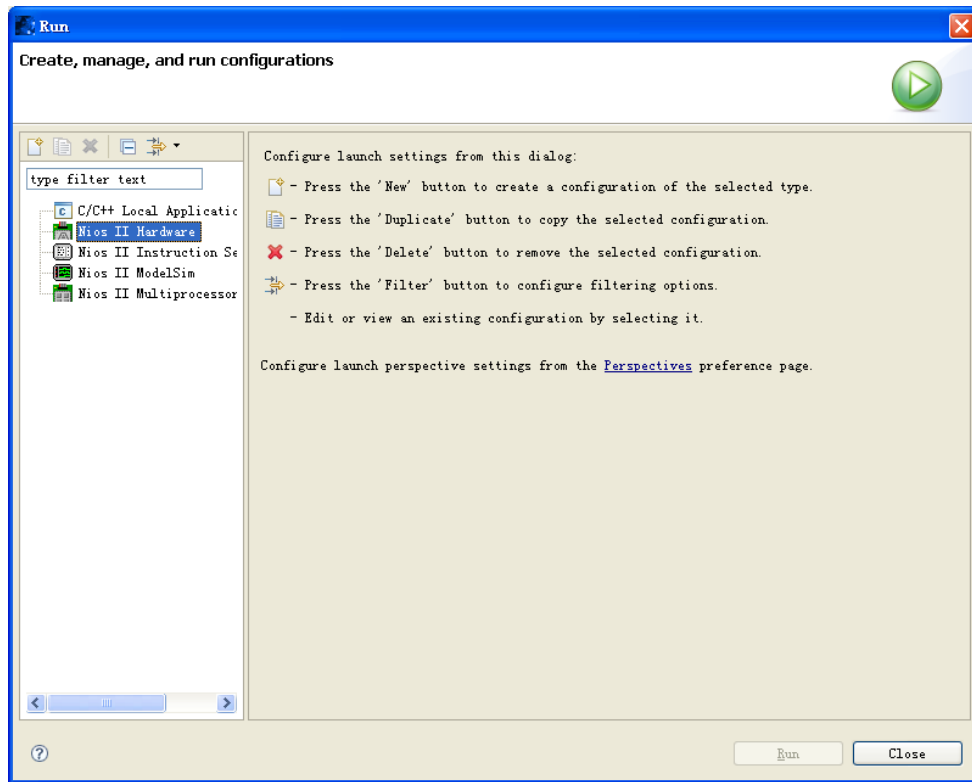



图 52

选择 Nios II Hardware， 点击左上角， 出现以下界面。

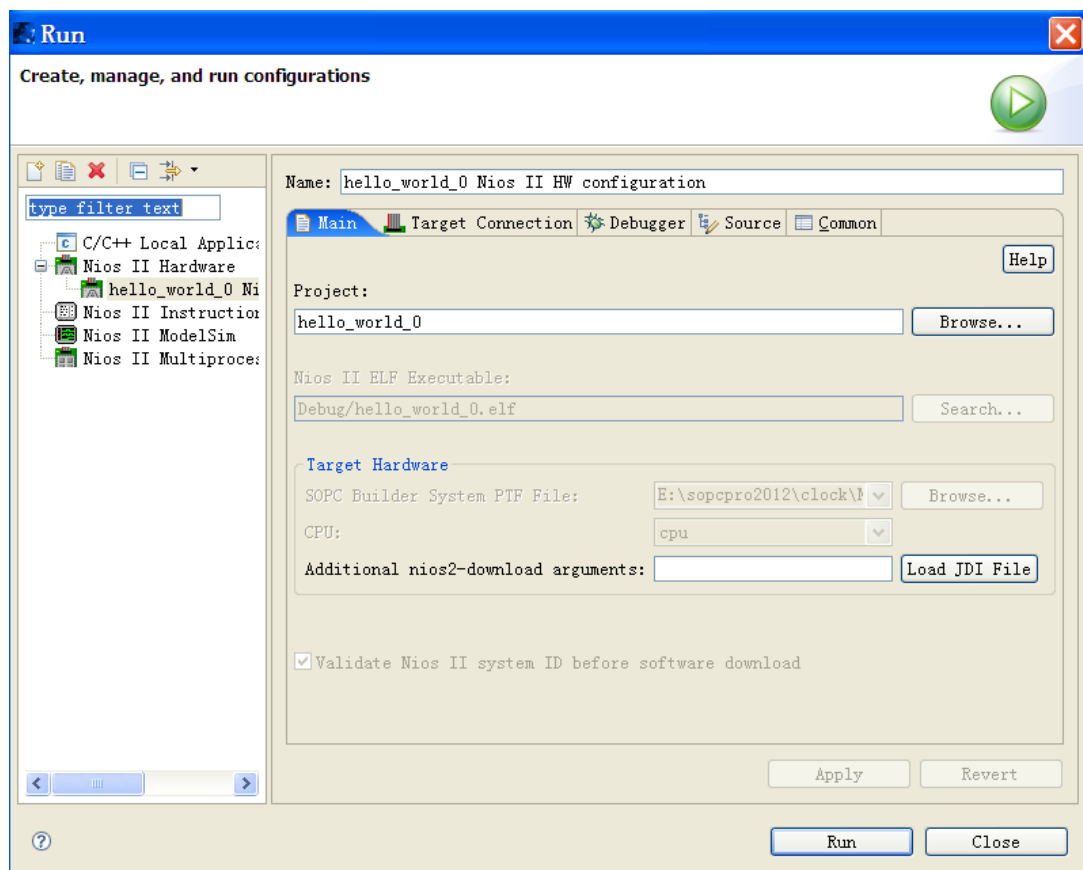


图 53

选择 Target Connection 选项页，做如下设置（注意：通信终端选择 jart_uart 哦！）

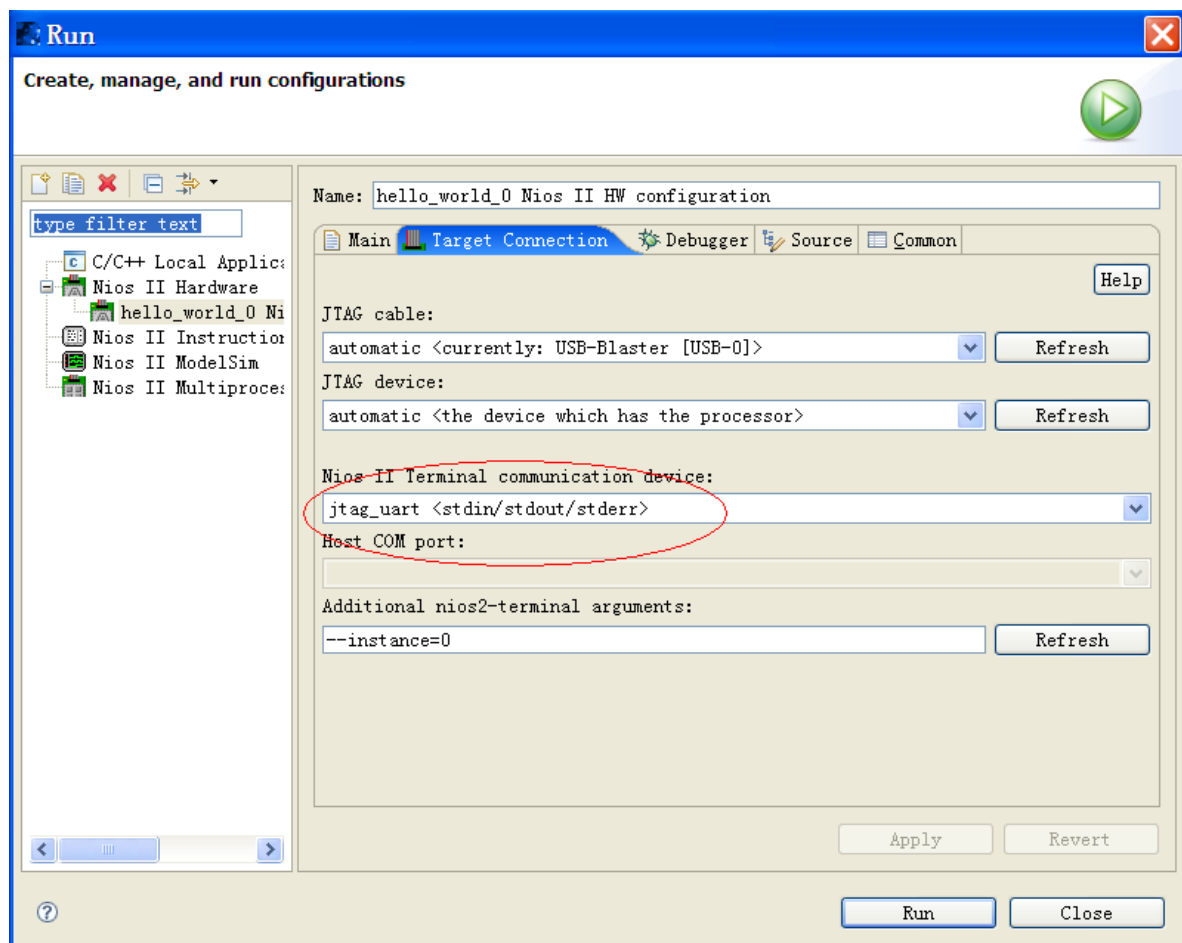


图 54

点击 Run,

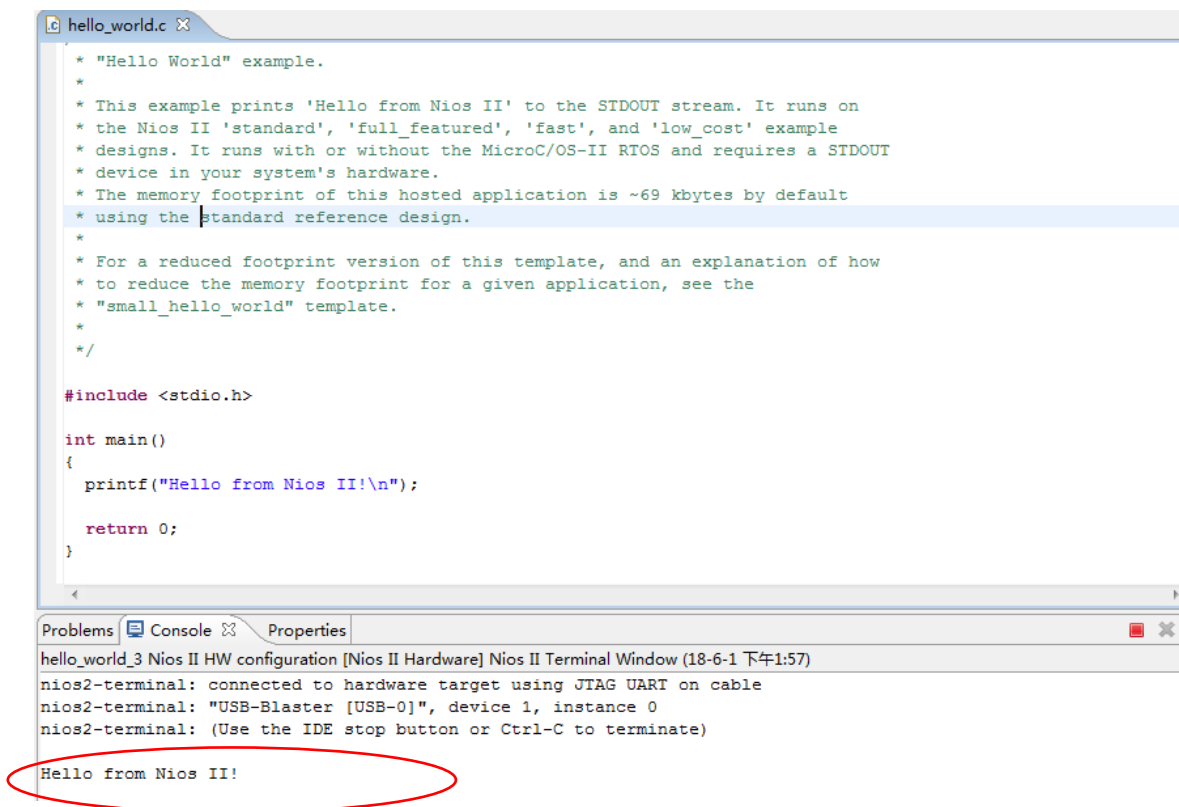


图 55

结果在控制窗口中打印“Hello from Nios II”。

恭喜大家！完成了 SOPC 的软硬协同设计之旅~~