第二节

使用 Qsys 创建硬件平台

通过上一章节如何构建一个Qsys系统的学习，我们知道了构建一个Qsys系统需要三个工具，它们分别是：Quartus II、Qsys 和 Nios SBT for Eclipse。下面我们就使用这三个工具来手把手教大家创建一个最基本的 Qsys 系统，并在该 Qsys 系统上开发一个简单的应用程序。通过 Qsys 系统创建和应用程序的开发，我们将引导读者学会 Quartus II、Qsys 和 Nios II SBT for Eclipse 三种开发工具的使用和操作。

§1.4 Qsys的开发流程

在开始创建 Qsys 系统前，我们先来介绍一下 Qsys 的开发流程。Qsys 的开发流程主要有以下三个步骤：

(1) 采用 Qsys 集成开发工具定制SOPC 系统模块，包括 Nios II 处理器以及相应的外设模块；

(2) 采用 Quartus II 软件建立工程与顶层实体，完成硬件语言逻辑电路。

(3) 采用 Nios II SBT for Eclipse 编写与 Qsys 系统相配套软件使其运行于 Nios II 处理器上。

在上述步骤里，采用 Quartus II、Qsys 中进行的设计我们一般称为硬件设计或硬件开发，而采用Nios II SBT for Eclipse所完成的设计我们称为软件设计或软件开发。对于比较简单的系统，一个人便可执行所有设计；对于比较复杂的系统，硬件和软件设计可以分开进行。接下来我们根据图 1.14 所示的开发流程图，分别对硬件开发和软件开发进一步介绍。

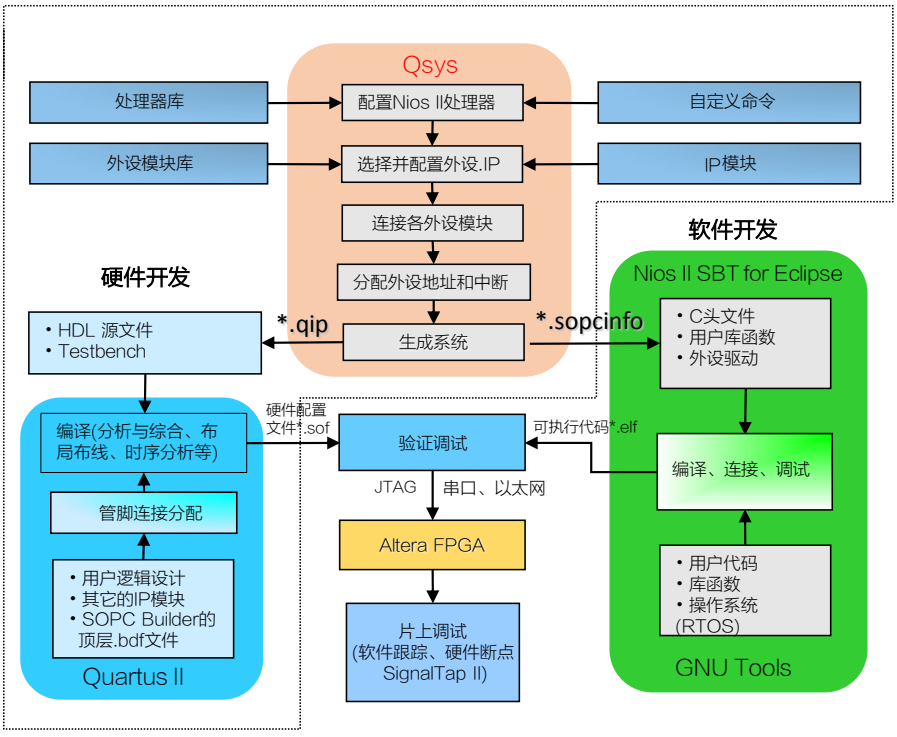


图 1.14 Qsys 系统的开发流程

首先我们介绍的是硬件开发，硬件设计的具体步骤如下：

(1) 使用 Quartus II 软件创建一个新工程，选取具体的 Altera FPGA 器件型号；当工程创建完毕后，在 Quartus II 软件中打开 Qsys 软件，从 Qsys 软件的处理器库和外设模块库中选择合适的 CPU、存储器以及各外围器件（如片内存储器、PIO、定时器、UART 等 IP 核），并定制和配置它们的功能；分配外设地址及中断号；设定复位地址；最后生成系统。用户也可以添加用户自身定制指令逻辑到 Nios II 内核以加速 CPU 性能，或添加用户外设以减轻 CPU 的任务。

(2) 使用 Qsys 软件生成系统后，将其集成到整个 Quartus II 工程中。可以在 Quartus II 工程中加入 Nios II 系统以外的逻辑，大多数的 SOPC 设计都包括 Nios II 系统以外的逻辑，这也是 SOPC 系统的优势所在。用户可以集成自身定制的硬件模块到 SOPC 设计，或集成从 Altera 或第 3 方 IP 供应商中得到的其他现成 IP 核模块。

(3) 使用 Quartus II 软件为 Nios II 系统上的各 I/O 分配引脚，另外还要根据要求进行硬件编译选项或时序约束的设置；最后编译 Quartus II 工程，在编译过程中 Quartus II 将对 Qsys 生成的系统的 HDL 设计文件进行布局布线，从 HDL 源文件综合生成一个适合目标器件的网表，生成 FPGA 配置文件（.sof）。

(4) 使用 Quaruts II 中的 Programmer 软件和 Altera 下载器（USB-Blaster），将配置文件下载到目标板上。当校验完当前硬件设计后，可将新的配置文件下载到目标板上的非易失性存储器里（如 EPCS 器件）。下载完硬件配置文件后，软件开发者就可以将此目标板作为软件开发的初期硬件平台进行软件功能的开发验证了。

接下来我们介绍的是软件开发， 软件设计的具体步骤如下：

(1) 打开 Nios II SBT for Eclipse 软件，将 Qsys 软件生成的.sopcinfo 文件导入到 Nios II SBT for Eclipse 软件中即可进行用户程序开发。这一步可以在 Qsys 生成系统模块后立即进行，也可以在整个系统编译完成，并将.sof 文件下载到目标板后进行。利用 Nios II SBT for Eclipse 软件开发这个过程与传统嵌入式系统的软件开发类似，唯一的不同在于，软件所运行的嵌入式系统是自己定制的、裁剪过的，因此，可能受硬件的局限会小一些。

(2) 当在 Nios II SBT for Eclipse 软件中完成了用户程序时，便可对用户软件进行编译，如果编译没有错误，软件便会生成可执行文件*.elf，将其下载到目标板中便可执行程序（下载*.elf 前，要先给目标板下载.sof 文件，否则将会出现错误提示）。

(3) 修改用户程序直到硬件和软件设计都达到设计要求，利用 Flash 下载工具，将配置文件*.sof 和可执行文件*.elf 编程到 Flash 中即可完成上电程序自启动。

至此，Qsys 的开发流程我们就讲解完了，读到这里，你是不是已经有些按耐不住了。别急，在下一小节中。我们将对上述过程进行亲手实践，手把手教你构建一个最基本的 Qsys 系统。

§ 1.5 手把手教你构建一个Qsys系统

通过前面 Qsys 开发流程的学习，我们知道想要创建一个 Qsys 系统，我们需要使用 Qsys 软件，接下来我们是不是就要打开 Qsys 软件，来创建 Qsys 硬件系统了呢？且慢，且慢，大家

还记得我们上一章节中的草图吗？在开始动手之前，我们应该根据我们所要实现的任务功能，来规划出我们的 Qsys 系统。通过规划出的 Qsys 系统，我们才能够知道，我们应该选择什么样的 IP 核，来构建我们的 Qsys 系统。接下来我们就来规划一下我们本次实验的 Qsys 系统，由于我们本次实验想要实现的功能是，利用按键中断来控制 LED 亮灭。所以，我们就可以根据这个功能，简单的画出我们的 Qsys 系统框架图，如图 1.15 所示。

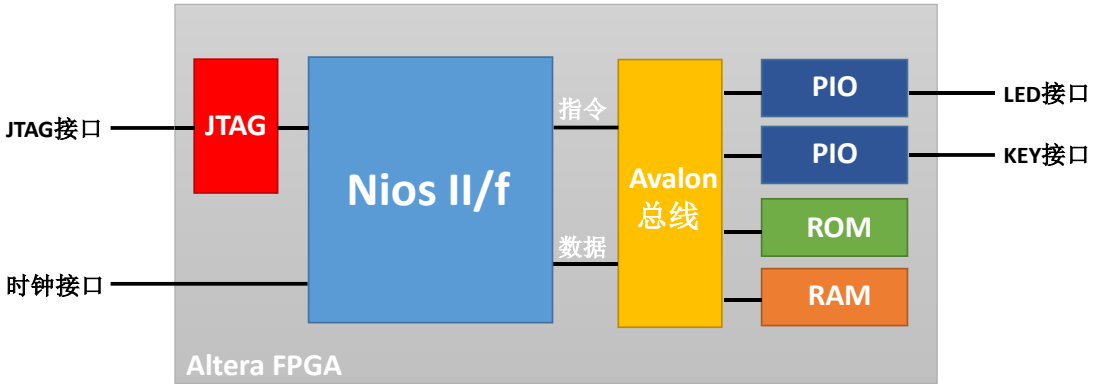


图 1.15 利用按键中断控制 LED 亮灭的 Qsys 系统框图

在该图中，我们可以看到，我们创建的 Qsys 系统有 Nios 核、系统时钟、JTAG 接口、片内 ROM、片内 RAM 和 PIO 接口，PIO 接口上又接有 LED 和按键两个外设。通过前面我们对系统概念的讲解，我们可以知道，Nios II 是 Qsys 系统总的调控中心，负责中断分配、地址管理、内存调度等总的控制任务。通用输入输出接口 PIO、存储器 RAM 和 ROM 等，这些外设控制器负责与外部设备连接，控制外设的行为。Nios II 和各个外设控制器之间的通信是通过 Avalon 总线。由此可见，该框架图就是一个典型的系统。

接下来我们就可以动手进行操作了，在操作过程中，大家可以一边看着文档，一边跟着我们的文档一步步自己尝试操作。当然，这个操作过程，我们在视频教程中也有详细的讲解，如果你不喜欢跟着文档操作，那么你可以跟着我们的视频教程进行操作。

1.5.1 使用 Quartus II 创建新工程

下面我们就来创建一个新工程。在创建工程之前，我们建议大家在硬盘中专门建立一个文件夹专用于存储自己的 Quartus II 工程，这个工程目录的路径名应该只有字母、数字和下划线，以字母为首字符，且不要包含中文和其他符号。（想要成为一个优秀的嵌入式工程师，首先要养成一个良好的习惯）。创建好了我们的专属文件夹后，我们便可以打开 Quartus 软件。在这里我们另外补充说明一下，Qsys 和 Nios II SBT for Eclipse 这两个软件不需要我们另外安装，我们在安装Quartus软件时，它们会一并安装，如果没有安装该软件的朋友，可以参考我们的《Quartus 软件安装篇》来进行安装。我们这里使用的软件是 Quartus II 13.1 版本，Quartus II 13.1 版本目前是最稳定的一个版本。

回到我们之前的话题，大家看图 1.16，这个就是 Quartus II 软件操作界面，学习过我们《软件工具篇》的朋友，想必 Quartus II 这个软件用的很熟了，这里我们就不在去介绍它了，不熟悉的朋友可以去看我们的《软件工具篇》来学习 Quartus II 软件的使用。

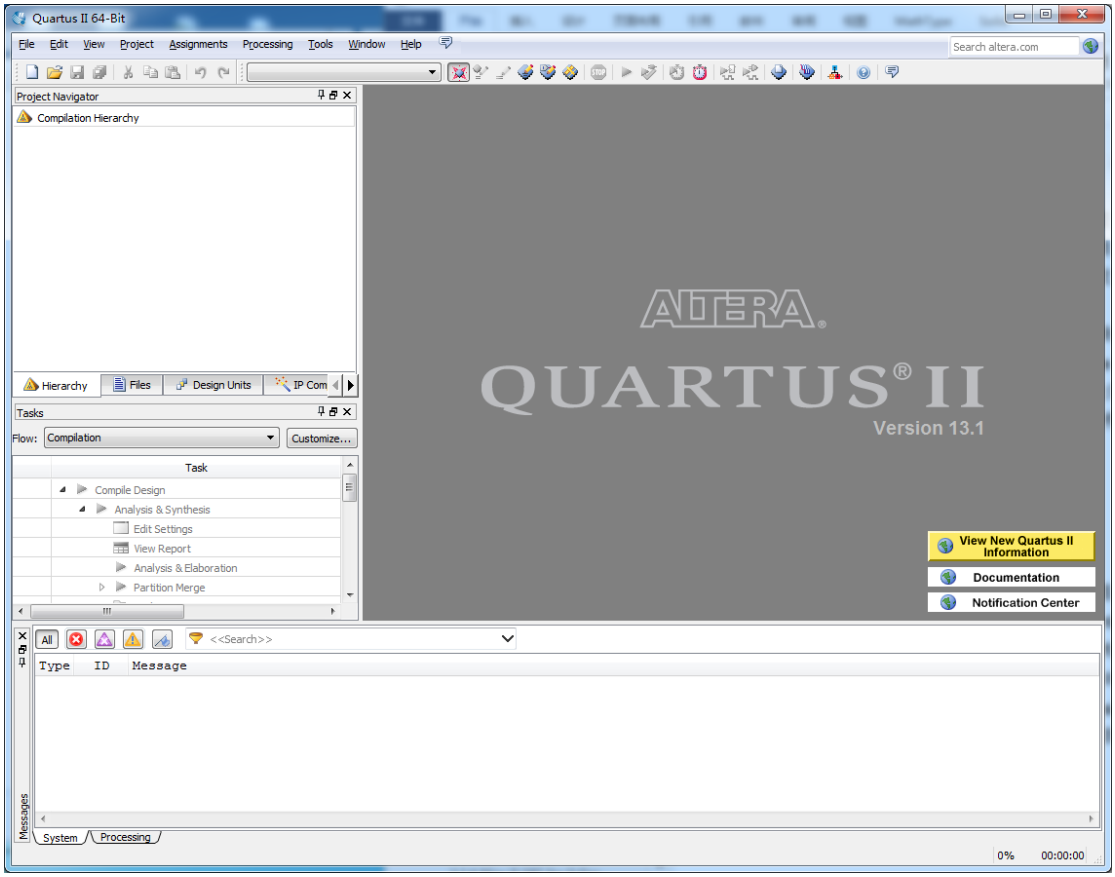


图 1.16 Quartus II 13.1 软件界面

我们在 Quartus II 软件的菜单栏中找到【File】→【New Project Wizard…】来新建一个工程。这里要注意不要把【New…】误认为【New Project Wizard…】。新建工程向导说明页面如图 1.17 所示。

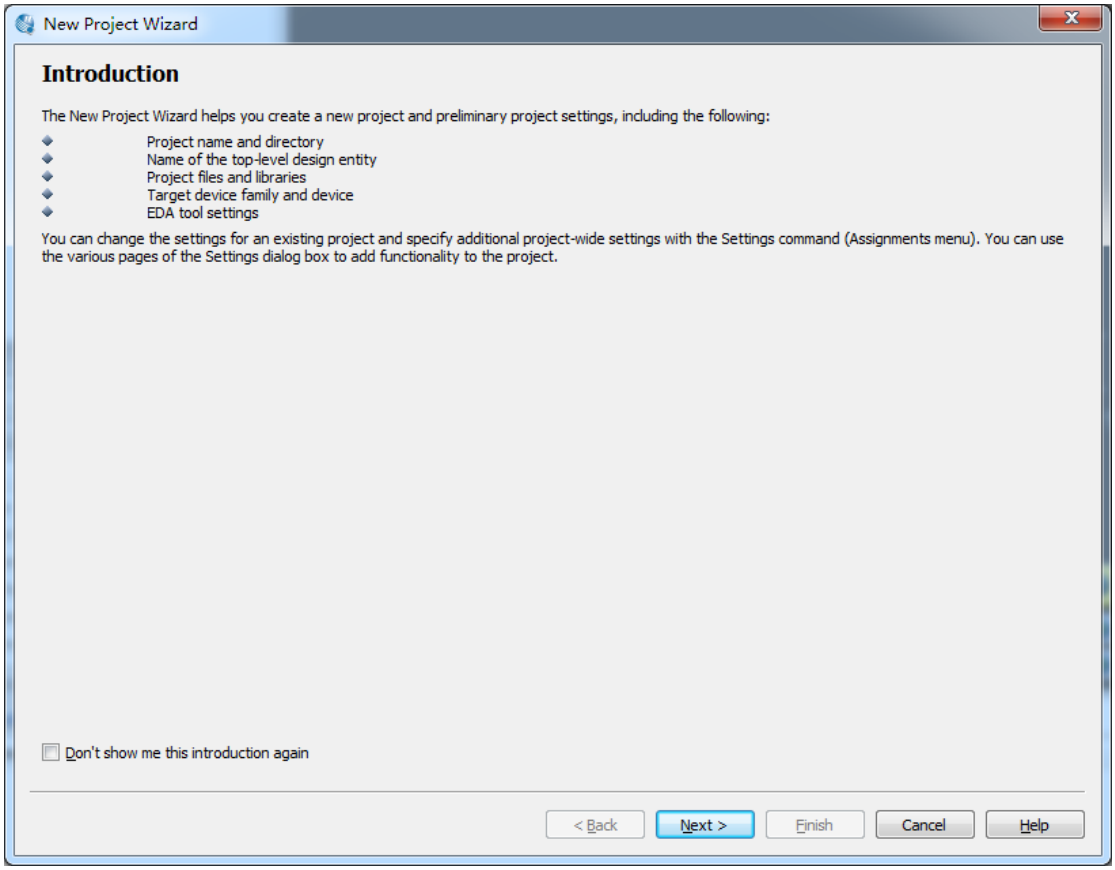


图 1.17 新建工程向导说明页面

在“Introduction”页面中，我们可以了解到在新建工程的过程中要完成哪些工作，这些工作主要包括以下 5 点：

(1) 指定项目目录、名称和顶层实体；
(2) 指定项目设计文件；
(3) 指定该设计的 Altera 器件系列；
(4) 指定用于该项目的其他 EDA 工具；
(5) 项目信息报告；

单击下面【 Next> 】按钮进入图 1.18 所示页面。任何一项设计都是一项工程，必须为此工程建立一个放置与此工程相关的所有文件的文件夹，此文件夹将被 Quartus II 默认为工作库。通常，不同的设计项目最好放在不同的文件夹中，而同一工程的所有文件都必须放在同一文件夹中。

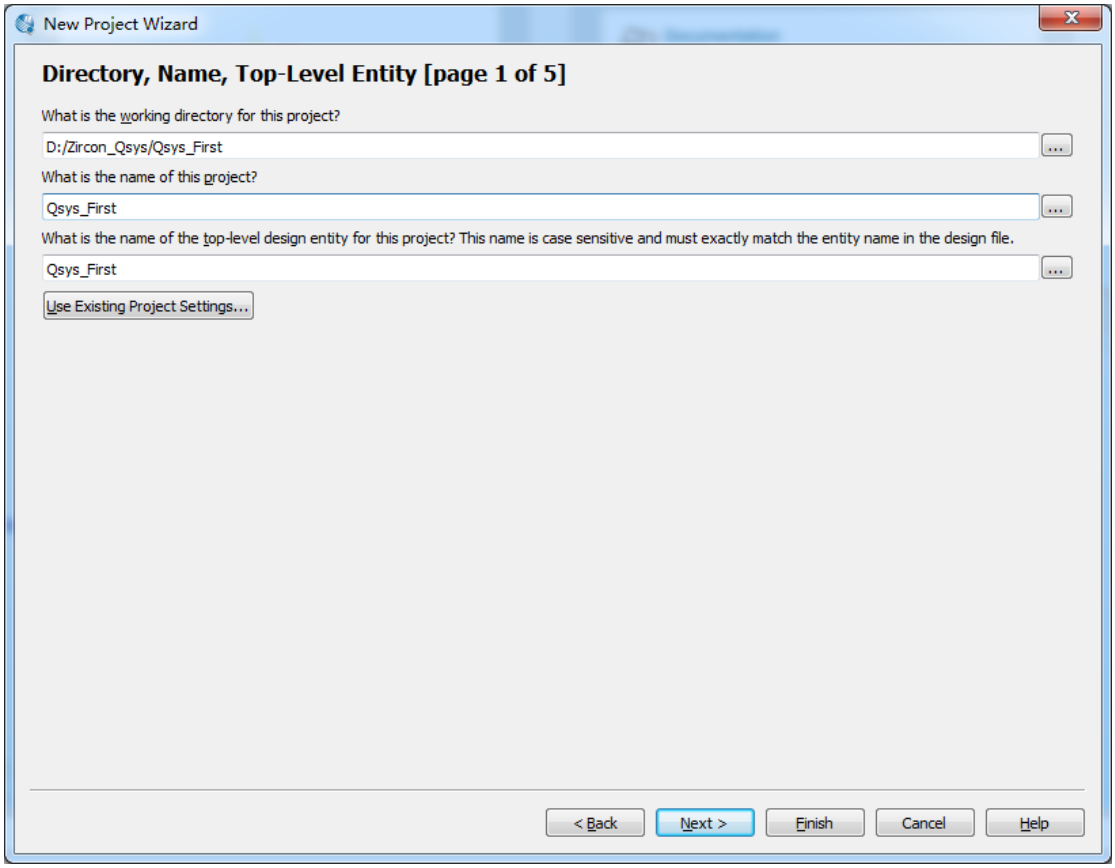


图 1.18 新建工程路径、名称、顶层实体指定页面

在这个页面中，我们可以看到有三个文本框，第一个文本框是让我们填写工程文件夹路径，第二个文本框是让我们填写工程名，第三个文本框是让我们填写顶层文件的实体名。这里我们设置的工程路径为 D:/Zircon_Qsys/Qsys_First 文件夹，这个 Zircon_Qsys 文件夹就是我们专门用来存放 Quartus II 工程的，这个 Qsys_First 文件夹就是我们现在创建的工程文件夹，为了方便工程的查找使用，我们这里就把工程名和顶层文件的实体名都命名为 Qsys _First。填写好了以后，我们就可以点击【Next>】进入图 1.19 所示页面。

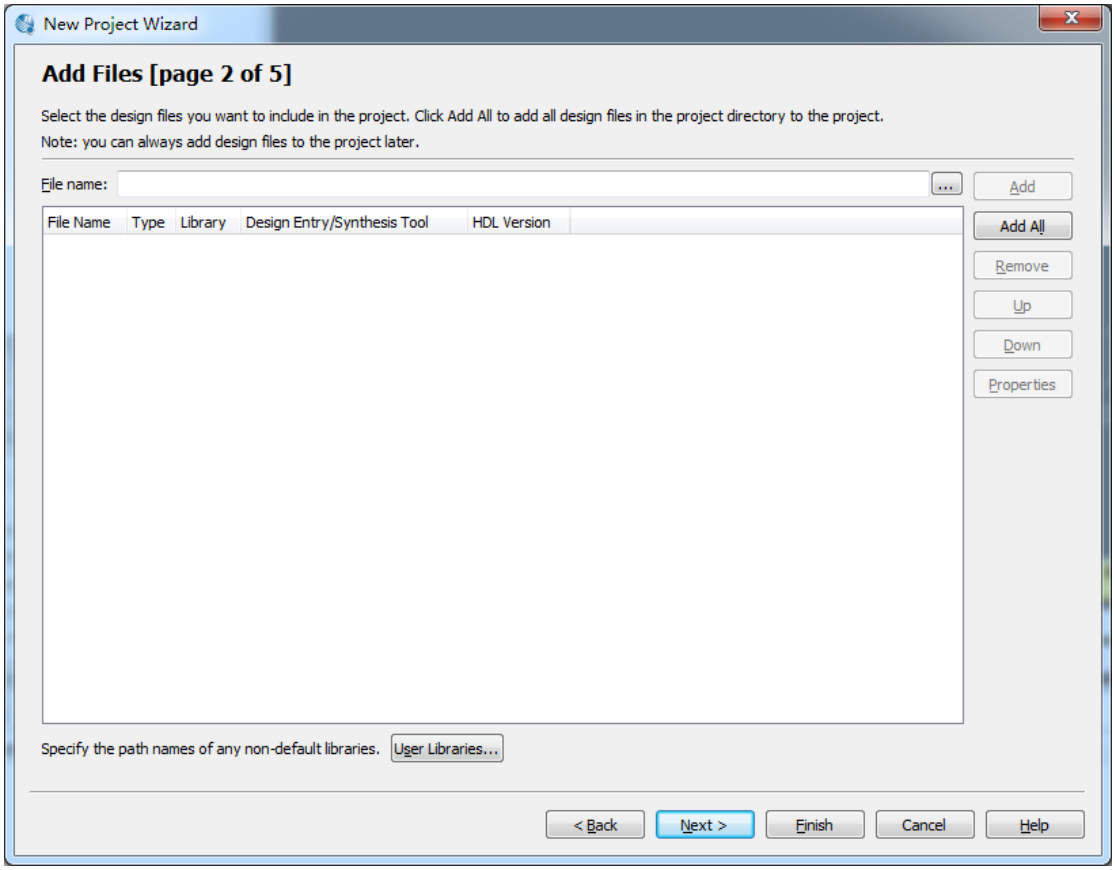


图 1.19 工程添加文件页面

在这个页面中，我们可以添加已经设计好的文件，由于我们是新工程，并没有文件可以添加，所以我们就不需要任何操作，直接点击【Next>】，进入图 1.20 所示页面。

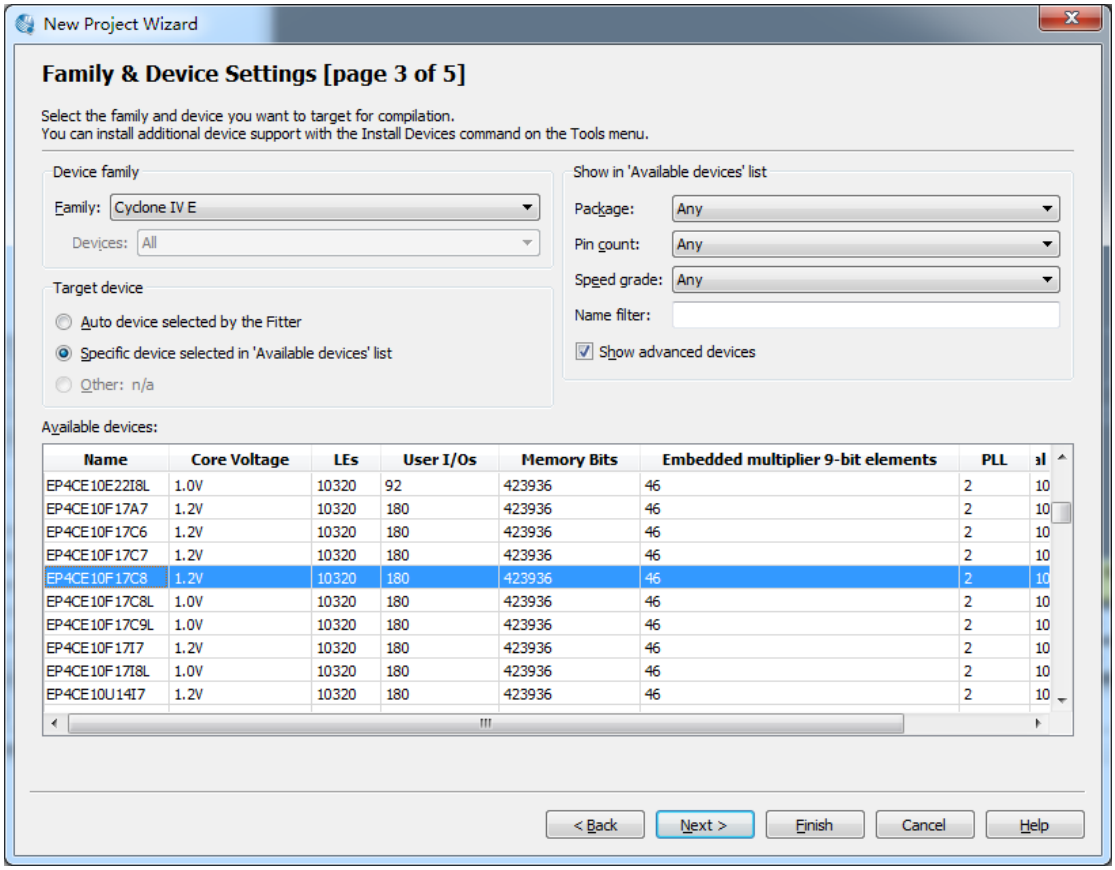


图 1.20 新建工程选择器件页面

在这个页面中，我们需要选择目标器件，在选择目标器件时，我们需要根据实际所用的FPGA芯片来进行选择，我们查看开发板主芯片，可以从主芯片上看到，我们使用的是 Cyclon IV E 系列的“EP4CE10F17C8”。接下来我们就可以在下面的芯片列表中找到EP4CE10F17C8这个芯片，找到这个芯片后，我们通过鼠标左键点击选中这个芯片就可以了。如果我们对这款芯片比较熟悉，知道这个芯片的封装和引脚数，那么我们就可以通过这边的筛选功能，来加快芯片查找的速度。大家看，我们还能够在这个窗口中查看到每个芯片器件的内核电压，逻辑资源，用户 I/O 等芯片的一些基本信息。选择完器件后，单击【Next>】进入图 1.21 所示页面。

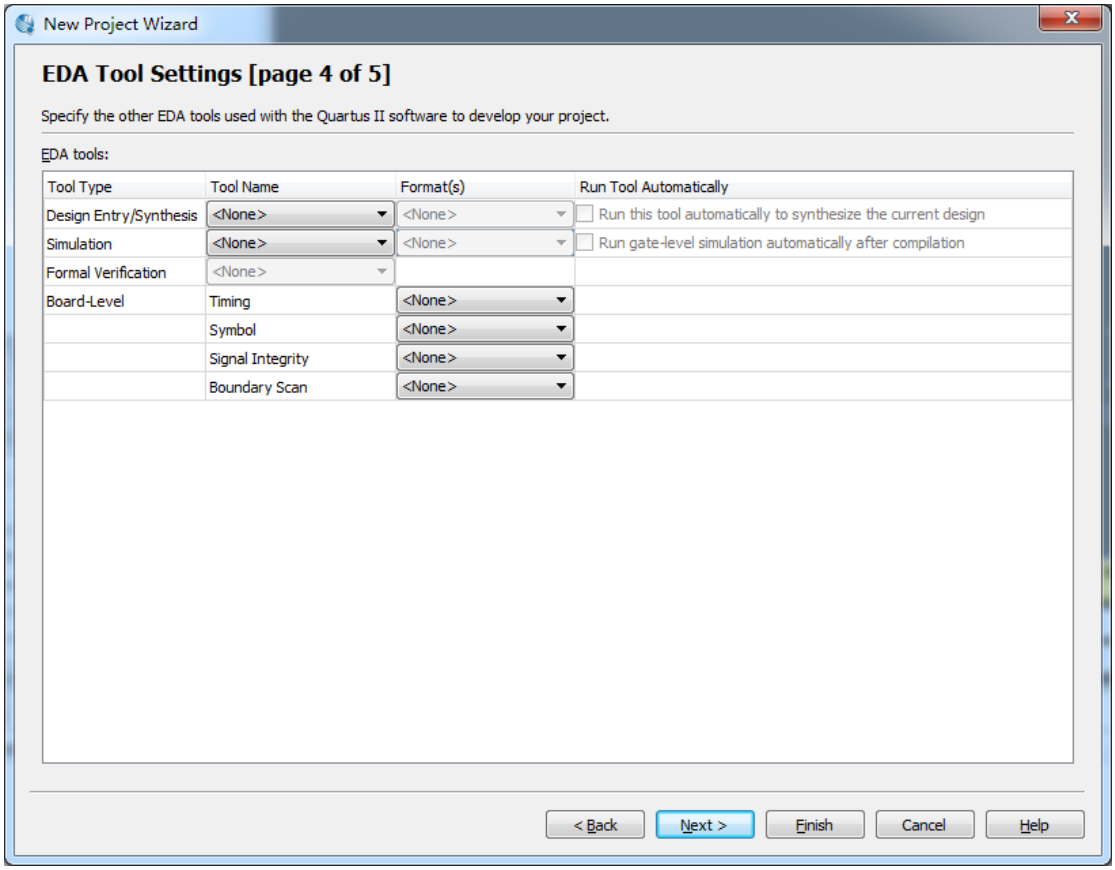


图 1.21 新建工程选择器件页面

在这个页面中，我们可以看出它是用来选择其他的 EDA 工具的，可以选择综合工具，还可以选择仿真工具，由于我们本次实验着重介绍Qsys这个软件，所以我们在这里就不介绍其他的EDA工具了，我们将这里的所有选项全都更改为<None>。接下来我们点击【 Next> 】进入图1.22所示页面。

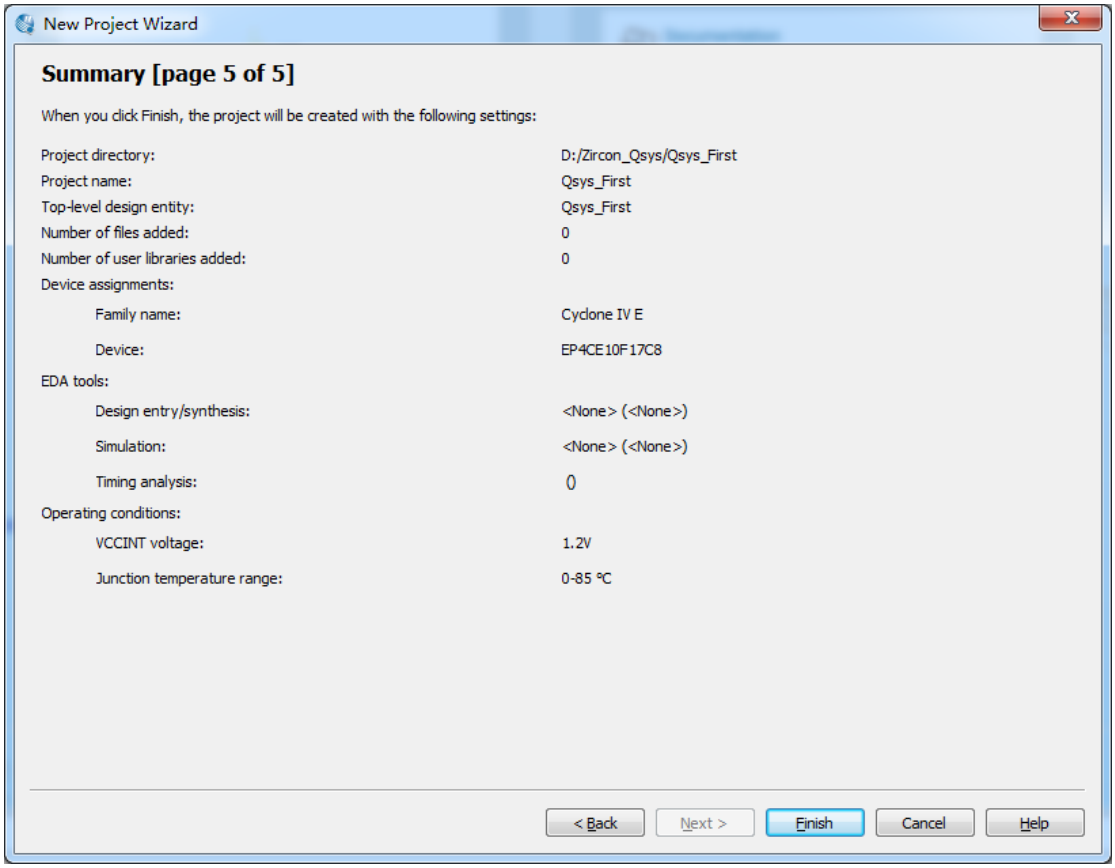


图 1.22 新建工程配置信息报告页面

在这个页面中，我们可以看到整个工程的创建信息，这些信息都是我们前面已经设置好的信息，我们粗略的扫了一下并没有发现错误的设置，下面我们就可以点击【Finish】即可完成工程的创建。这里需要注意的是，工程建立完毕后，还可以根据设计中的实际情况，通过选择【Assignments】菜单下的【Device…】或者【Settings…】对工程进行重新设置。

1.5.2 使用 Qsys 软件创建 Qsys 系统

创建好工程后，我们就要使用 Qsys 软件来创建 Qsys 硬件系统了。我们在 Quartus II 菜单栏中选择【Tools】→【Qsys】选项来启动 Qsys 软件，如图 1.23 所示。

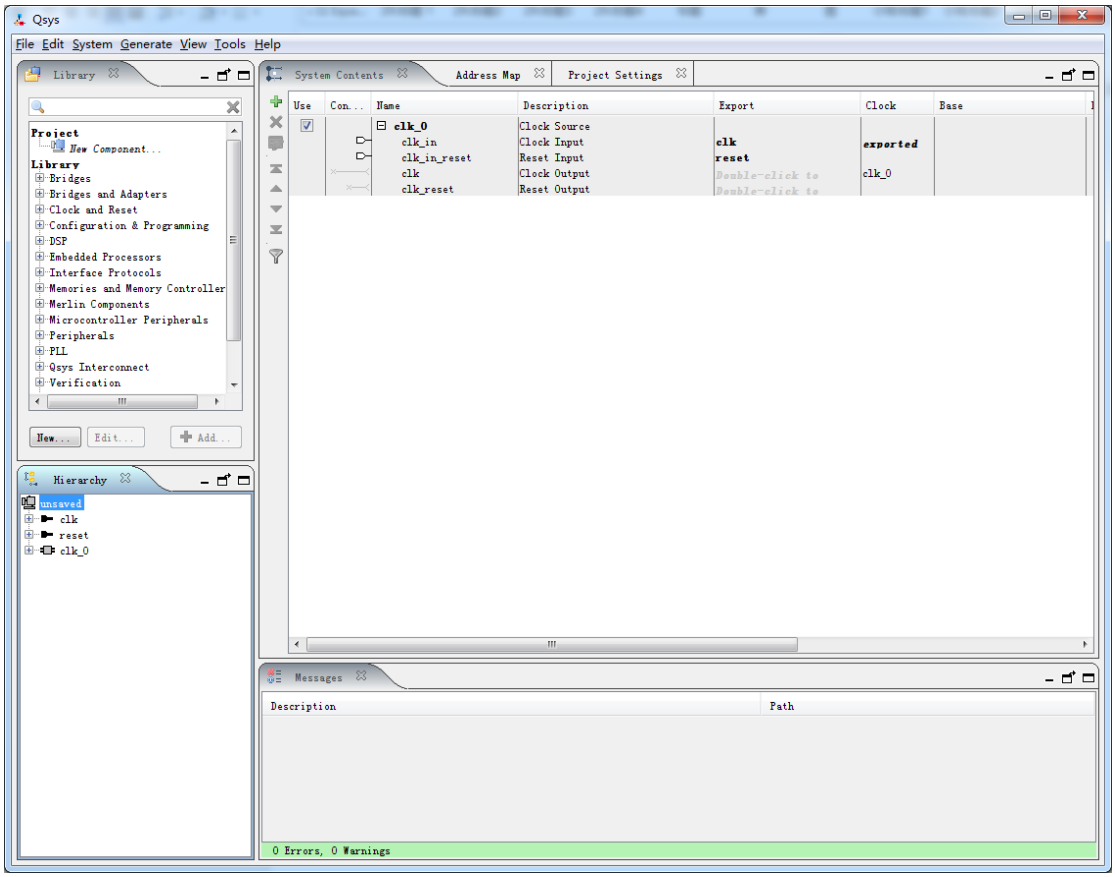


图 1.23 Qsys 软件界面图

大家看，这个就是 Qsys 软件工具界面，在这个界面中，我们可以看到系统默认为我们添加了一个时钟 IP 核组件，在定制系统前，我们首先就需要指定系统的时钟频率，Qsys 将利用这个时钟频率来产生时钟分频或波特率等，需要注意的是，所设置的频率要与系统实际运行的时钟频率相匹配。这里我们双击时钟组件或者右键时钟组件点击【Edit… 】选项，就可以修改时钟频率，如图 1.24 所示。

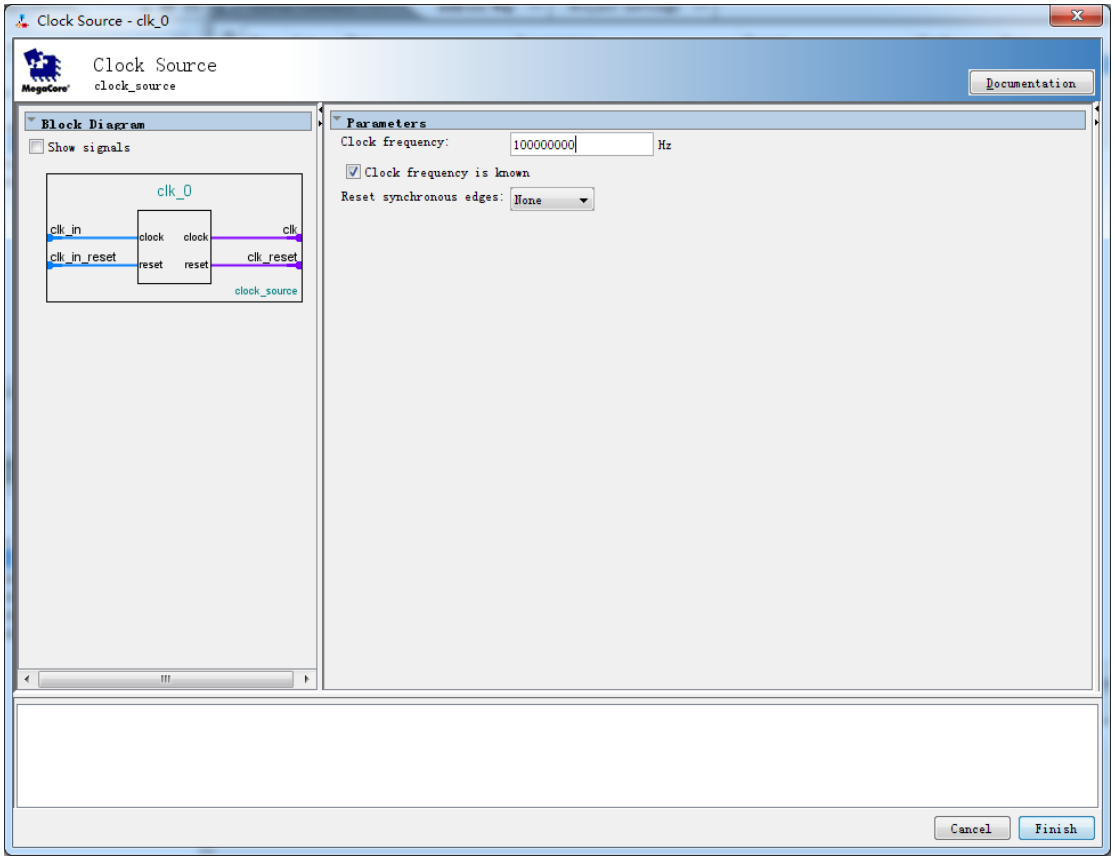


图 1.24 修改时钟频率页面

在该页面中，我们可以看到默认的时钟频率是 50MHz，这里我们就将它修改成 100MHz，修改时钟频率可以提高 Nios II 的运行速度。修改好了以后，我们直接点击【 Finish 】即可完成修改。时钟 IP 核配置好了以后，我们就可以根据之前规划好的 Qsys 系统框架图，来一步步添加我们的 IP 核。首先我们添加是 Nios II IP 核，没有使用过 Qsys 软件的朋友也许会有这么一个疑问，如何添加 Nios II IP 核呢？ 其实添加 Nios II IP 核是非常简单的。在图 1.23 所示的 Library 窗口中我们可以看到 Altera 为我们提供的所有的 IP 核组件，我们可以在 Embedded Processors 中找到 Nios II Processor 内核组件，也可以在上面的搜索栏中输入 Nios 来快速找到 Nios II Processor 内核组件。找到 Nios II Processor 后，我们双击 Nios II Processor，便会弹出 Altera Nios II 配置向导页面，如图 1.25 所示。

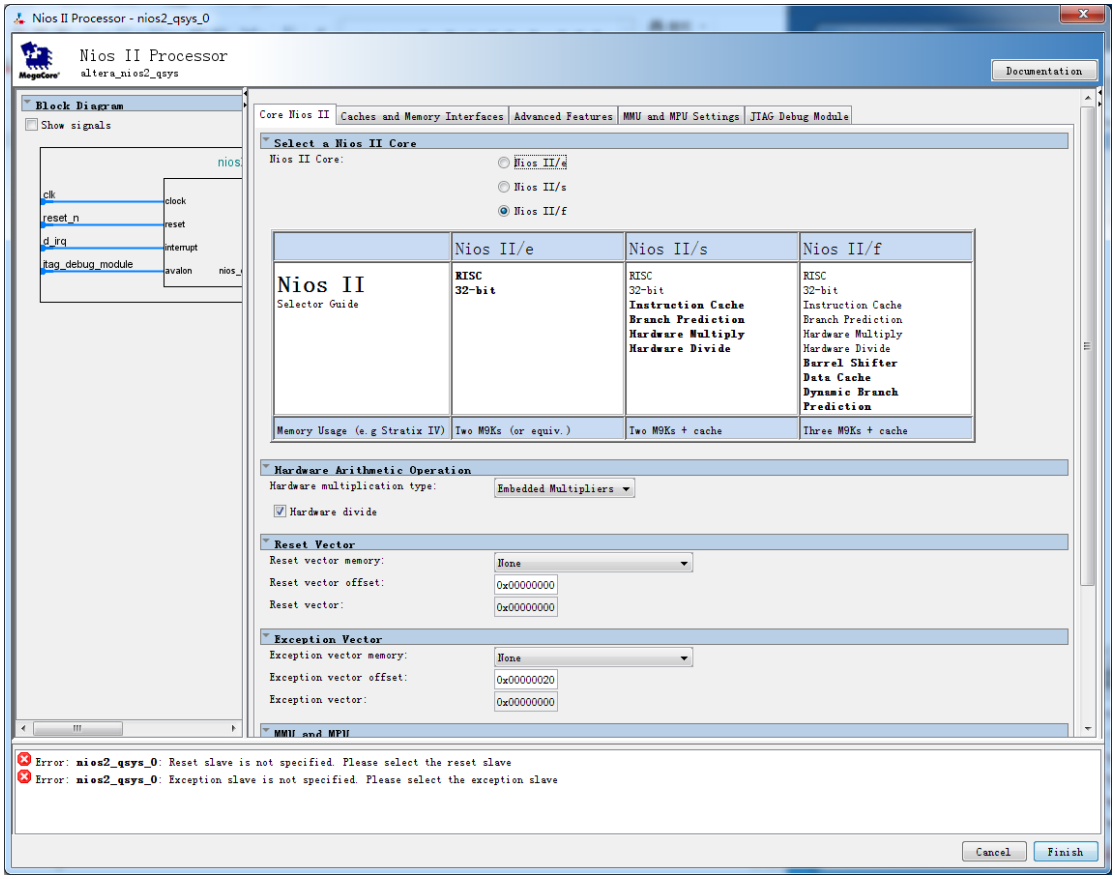


图 1.25 Nios II 内核配置向导页面

大家看，这个就是 Nios II IP 核的设置页面，在这个设置页面中，我们可以设置不同等级的 Nios，e 等级的 Nios II 它消耗的逻辑资源最少，但是运行速度最慢，f 等级的 Nios II 它消耗的逻辑资源最多，但是它的运行速度是最快的，而 s 等级的 Nios II 处于 e 和 f 之间。看到这里，也许有朋友会问，什么时候用 e 等级？什么时候用 f 等级？，关于 Nios 等级的选择，我们是要根据应用要求和逻辑资源来决定的。如果你的应用对速度要求比较高，并且逻辑资源也很充足，那么你可以选择 f 等级的 Nios，否则只能选择低等级的 Nios。我们本次实验对 Nios 等级并没有什么要求，这三种等级的 Nios 我们都可以选择使用，在这里我们选择的是 Nios II/f。

选择完了 Nios 等级，接下来我们再来看硬件运算操作（Hardware Arithmetic Operation），在 Nios II/s 和 Nios II/f 标准中想必大家都有看到 Hardware Multiply（硬件乘法器）和 Hardware Divide（硬件除法器），Nios II/s 和 Nios II/f 的 DMIPS 性能主要依赖于硬件乘法选项。这里我们就说下什么是 DMIPS（MIPS 相信大家还是比较熟悉的，它是 Million Instructions Per Second 的缩写，即计算机每秒钟执行的百万指令数，D 是 Dhrystone 的缩写，它表示在 Dhrystone 这样一种测试方法下的 MIPS，Dhrystone 是一种整数运算测试程序，主要用于测整数计算能力）。由于我们选择的是 Nios II/f，所以这里我们选择使用硬件乘法器和硬件除法器。其他选项的设置，如复位地址和异常地址，我们将在稍后的设置中进行说明，这里我们先不更改默认即可。

下面我们来看图 1.26 所示的 Caches and Memory Interfaces 页面。

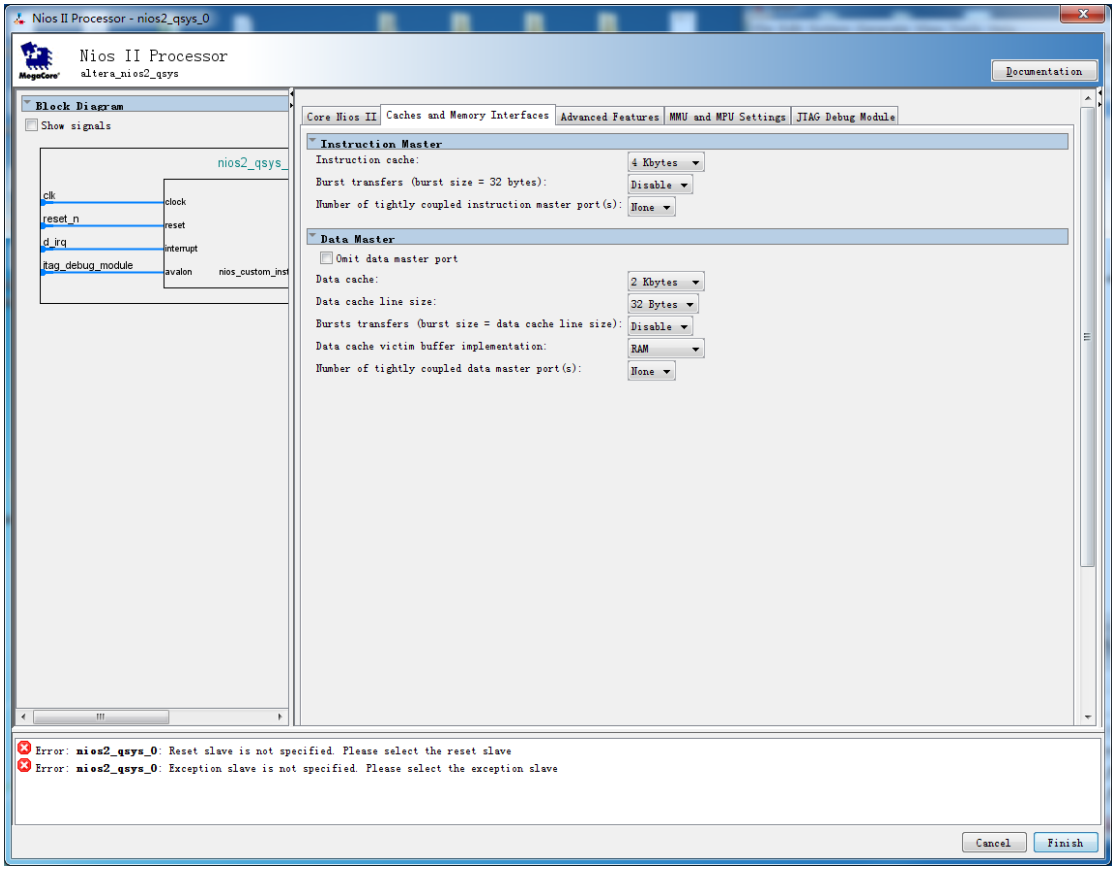


图 1.26 Caches and Memory Interfaces 页面

在图 1.26 页面中我们可以看到 Instruction Master (指令主端口) 和 Data Master (数据主端口)，我们可以在这里设置它们各自 Cache 容量的大小和突发传输的方式，需要注意的是，Tightly Couplein Instruction Master Prot(紧耦合指令主端口) 和 Tightly Couplein Data Master Prot (紧耦合数据主端口) 的设置，该选项用于在 Nios II CPU 核里面构建与 CPU 外部的存储器紧密耦合的数据端口。通过该端口与存储器交换数据比通过 Avalon 总线块，如果选择了这个端口，那么必须指定片内存储器，并且手工将端口与存储器相连。关于指令和数据，还有紧耦合详细内容可参考 3.1.5 节中的介绍。这里我们使用默认的设置即可。

下面我们来看图 1.27 所示的 Advanced Features 页面。

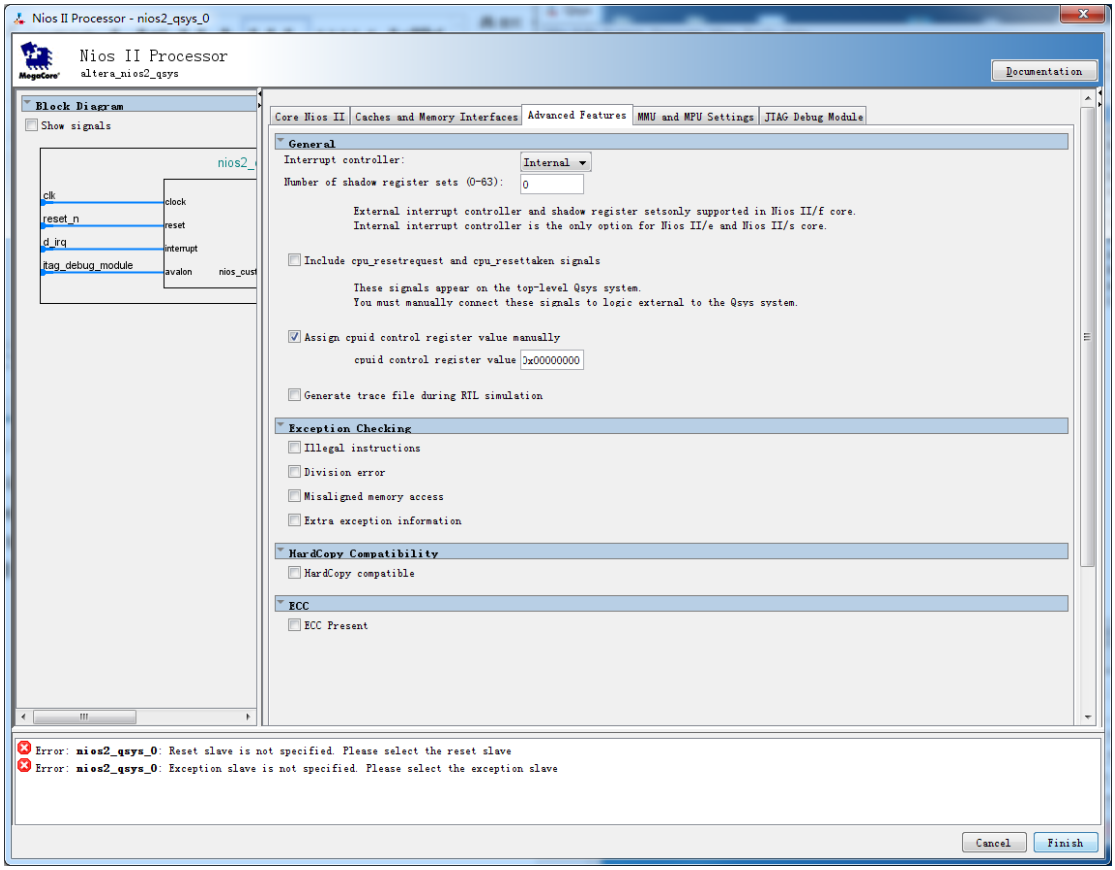


图 1.27 Advanced Features 页面

从图 1.27 中我们可以看到中断控制配置有两种，一种是内部控制器，另一种是外部控制器；内部控制器只能用于 Nios II/e 和 Nios II/s，而外部控制器通常和影子寄存器配合使用，这种配合使用可以大大减少中断延迟，这里要注意的是，外部控制器只能用于 Nios II/f。Nios II 处理器可以选择一个或多个影子寄存器组，它是通用寄存器组的一个完整的备用，用于在 ISR 中维持一个独立的上下文环境，关于影子寄存器和外部中断的详细内容可参考第 3.1.4 节中的介绍。

下面我们再来看一下检查设置提供的几个选项：illegal instruction（非法指令）、division error（除法错误）、misaligned memory access（非法存储器访问）：不一致的指令/数据地址导致出现非法存储器访问、extra exception information（异常指令）、HardCopy compatible（硬拷贝兼容）、ECC present（检测错误）。由于上面设置我们几乎用不到，这里我们就不在进一步进行介绍，直接默认设置即可。

下面我们来看图 1.28 所示的 MMU and MPU SettingsAdvanced Features 页面。

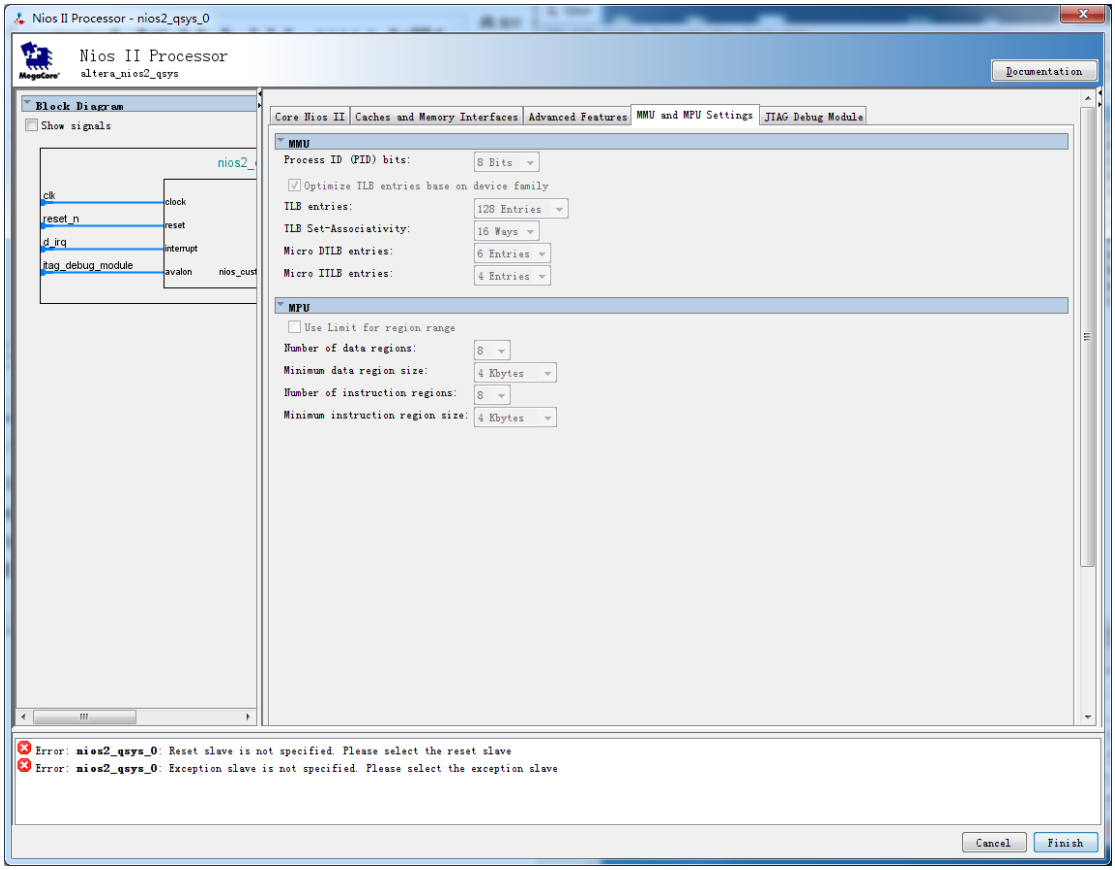


图 1.28 MMU and MPU Settings 页面

从该图中我们可以看到，这里面主要有两个设置，一个是 MMU，另一是 MPU，所谓 MMU（Memory Management Unit）就是存储器管理单元。所谓MPU，就是内存保护单元（即:Memory Protection Unit），由于这里我们同样也没有用到 MMU 和 MPU，这里我们也是默认设置，并没有任何改动，关于 MMU 和 MPU 详细内容可参考 3.1.5 节中的介绍。

下面我们来看图1.29所示的JTAG Debug Module页面。为了方便调试，为CPU加入JTAG调试模块，JTAG调试模块要占用较多的逻辑单元，如果整个系统已经调试完毕了，可以选用No Debugger，以减少系统占用资源。JTAG 调试模块根据功能的不同，分成了 4 级。具体 JTAG 调试模块内容可参考 3.1.6 节中的介绍。这里我们同样不改变其中的设置，使用默认设置即可。

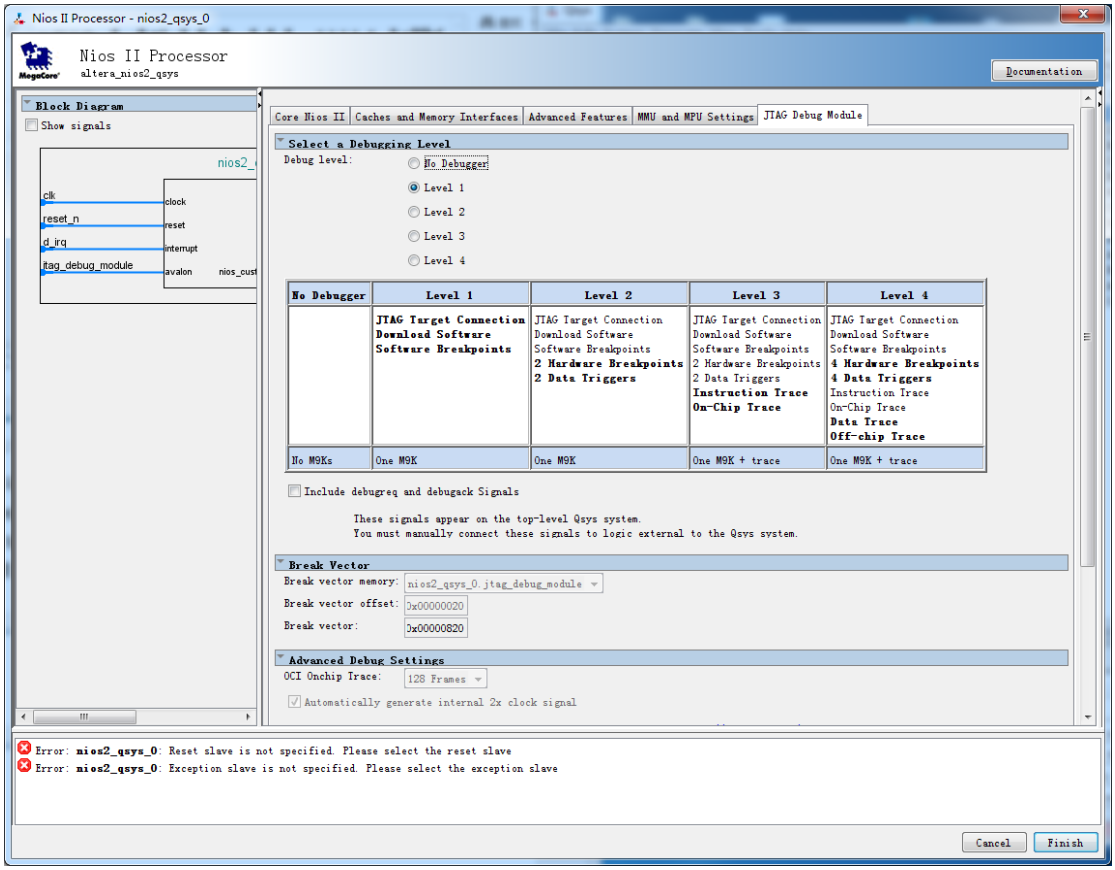


图 1.29 JTAG Debug Module 页面

至此，我们点击【Finish】来完成 Nios II 处理器的配置，生成一个带有 JTAG 调试接口的 Nios II/f 行 CPU 核，这时，我们会发现主界面窗口中出现了 Nios II 组件，并且还会在 Messages 窗口中出现错误信息，如图 1.30 所示，这些信息我们暂时不必关心，将在后面步骤中对它们进行处理。

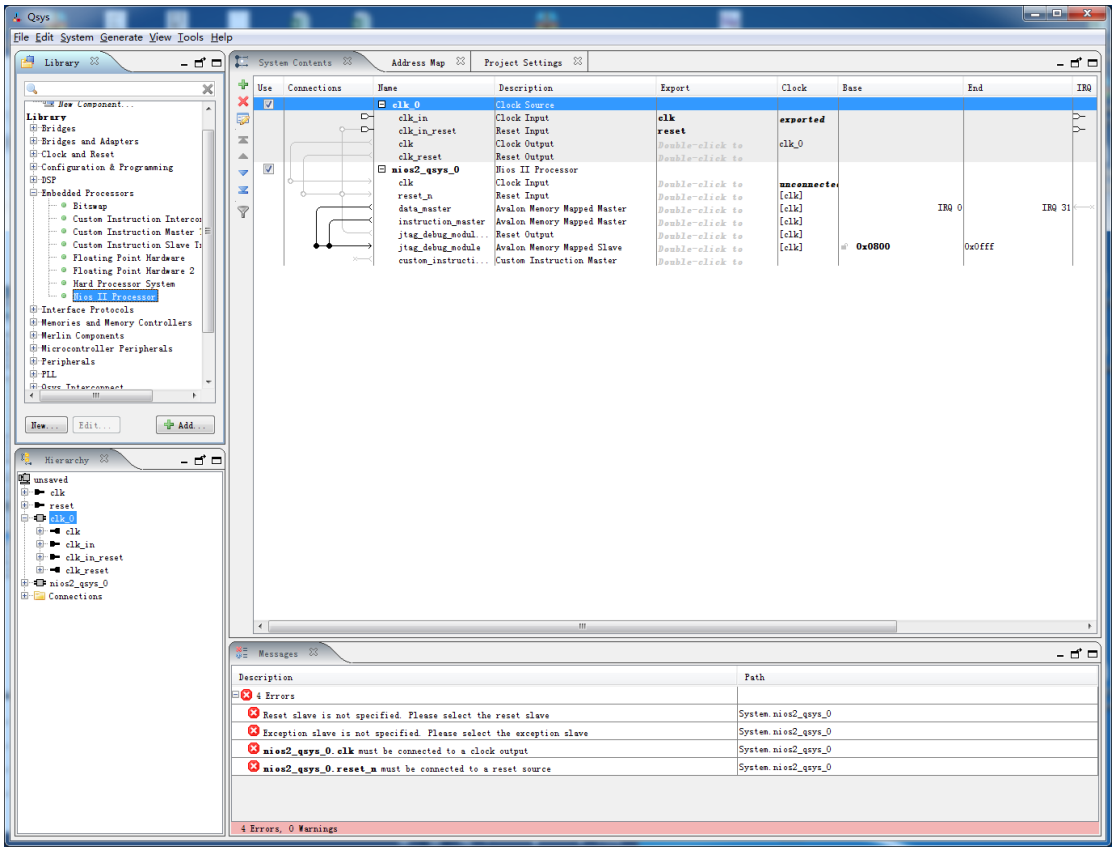


图 1.30 添加 Nios II 核后的页面

通过前面系统概念的学习我们知道，每个处理器系统至少要有一个存储器用于存储数据和指令，接下来我们便来添加一个片内存储器，用于存储程序代码以及程序运行空间，我们在 Library 库中搜索直接 Memory 找到 On-Chip Memory(RAM or ROM)，双击即可弹出片内存储器配置向导页面，如图 1.31 所示。

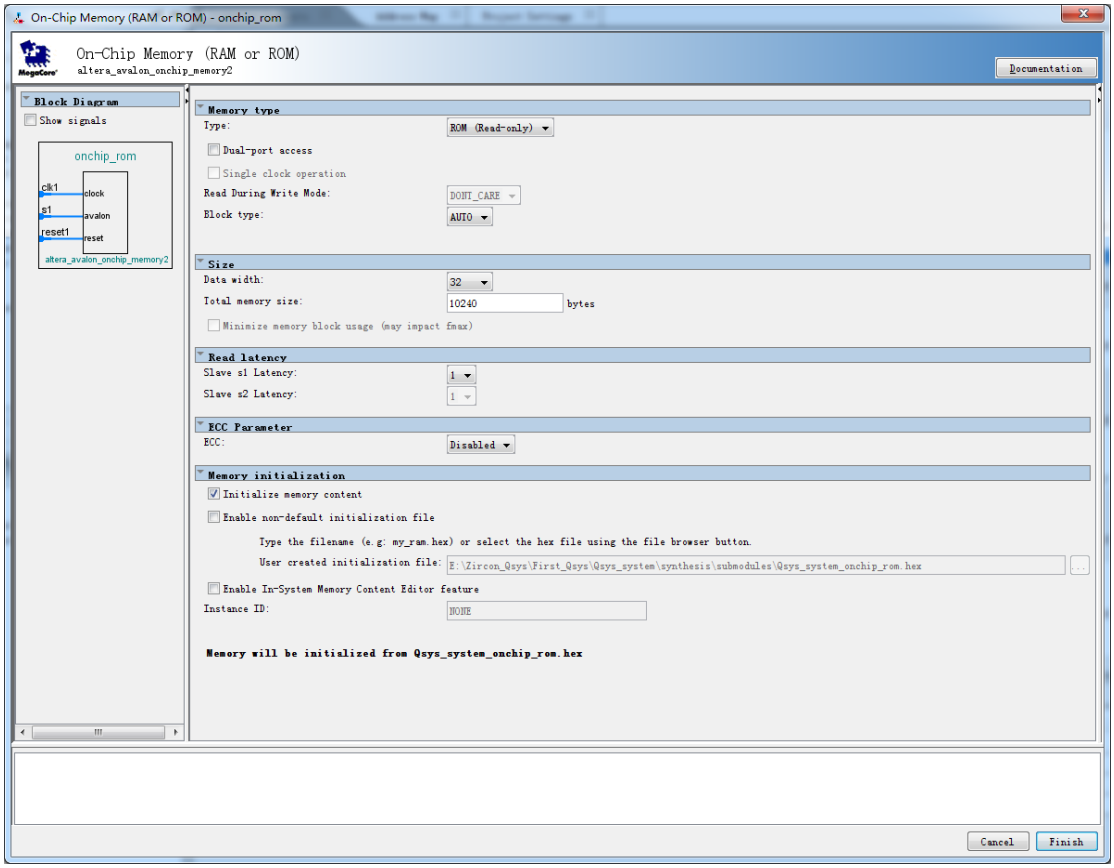


图 1.31 片内存储器 ROM 的配置向导页面

在图 1.31 中的 Memory type 选项区域中选择 ROM，即指定为 ROM 型。FPGA 内部其实并没有专用的 ROM 硬件资源，实现 ROM 的思想是对 RAM 赋初值，并保持该初值，使其为只读的。ROM 的内容在对 FPGA 进行配置时，一起写入 FPGA 中。在 Total memory size 文本框中输入 10240，即指定 10KB 的存储容量。其他的配置我们不做改变，也同样使用默认设置。在这里需要注意的是：ROM 存储器在上电时可以配置使用 onchip_mem.hex 文件进行初始化。onchip_mem.hex 在 Quartus II 中可以由用户编辑生成。在这里，由于我们使用 onchip_ROM 来存储用户程序，onchip_mem.hex 将由 Nios II SBT for Eclipse 编译生成，文件的内容即用户程序。我们点击【Finish】即可完成片内存储器的配置，便可以在 Qsys 界面中看到添加的片内存储器组件。

下面我们使用同样的方法再添加一个 onchip_RAM，和 onchip_ROM 不同的是，我们在选择类型时选择 RAM 类型，RAM 的存储大小我们设置 20480bytes，也就是 20KB，RAM 的其他设置也不做改变，使用默认设置即可，RAM 的配置页面如图 1.32 所示。

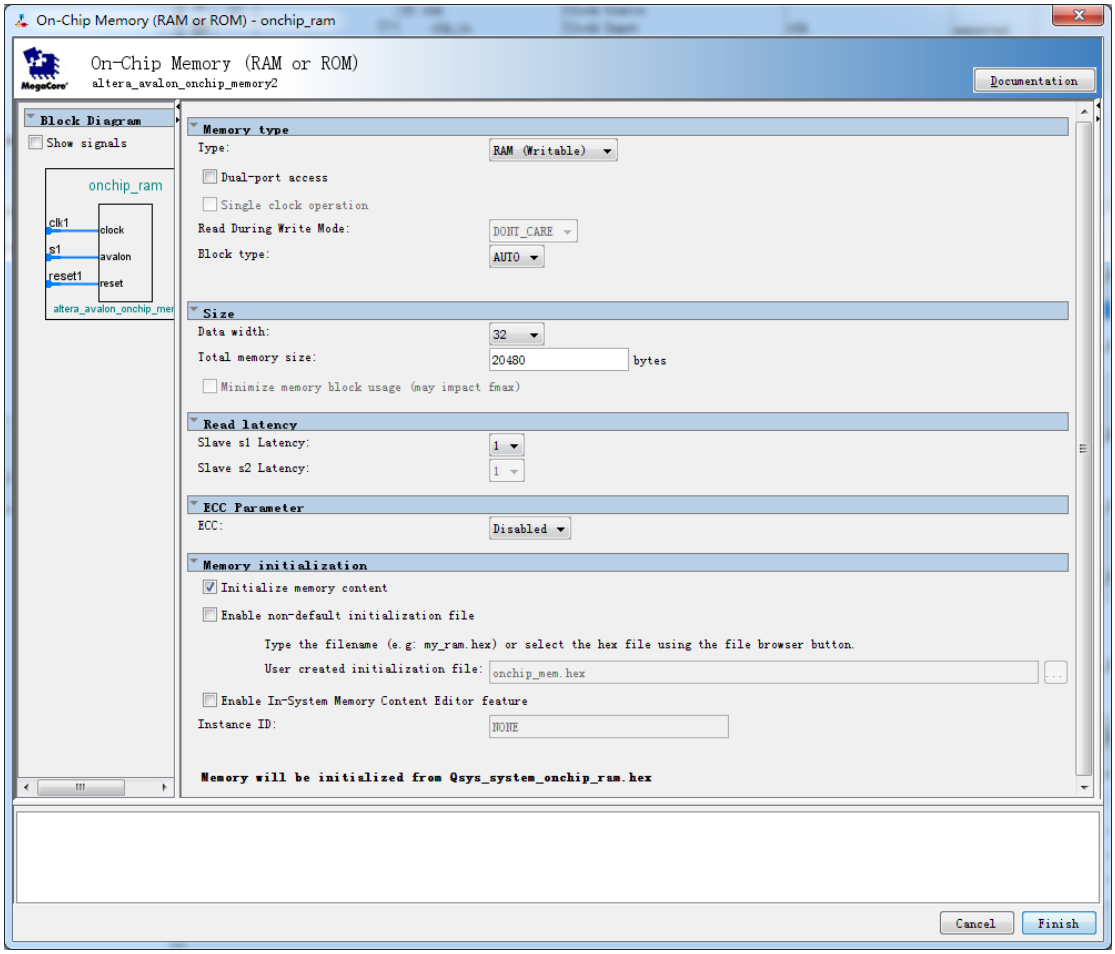


图 1.32 添加片内存储器 RAM 的配置向导页面

下面我们就需要添加 JTAG UART 的组件来实现 PC 和 Nios II 系统间的串行通信。我们需要在 Library 库中找到 JTAG UART 组件，双击 JTAG UART 组件，将会弹出 JTAG UART 配置向导页面，如图 1.33 所示。

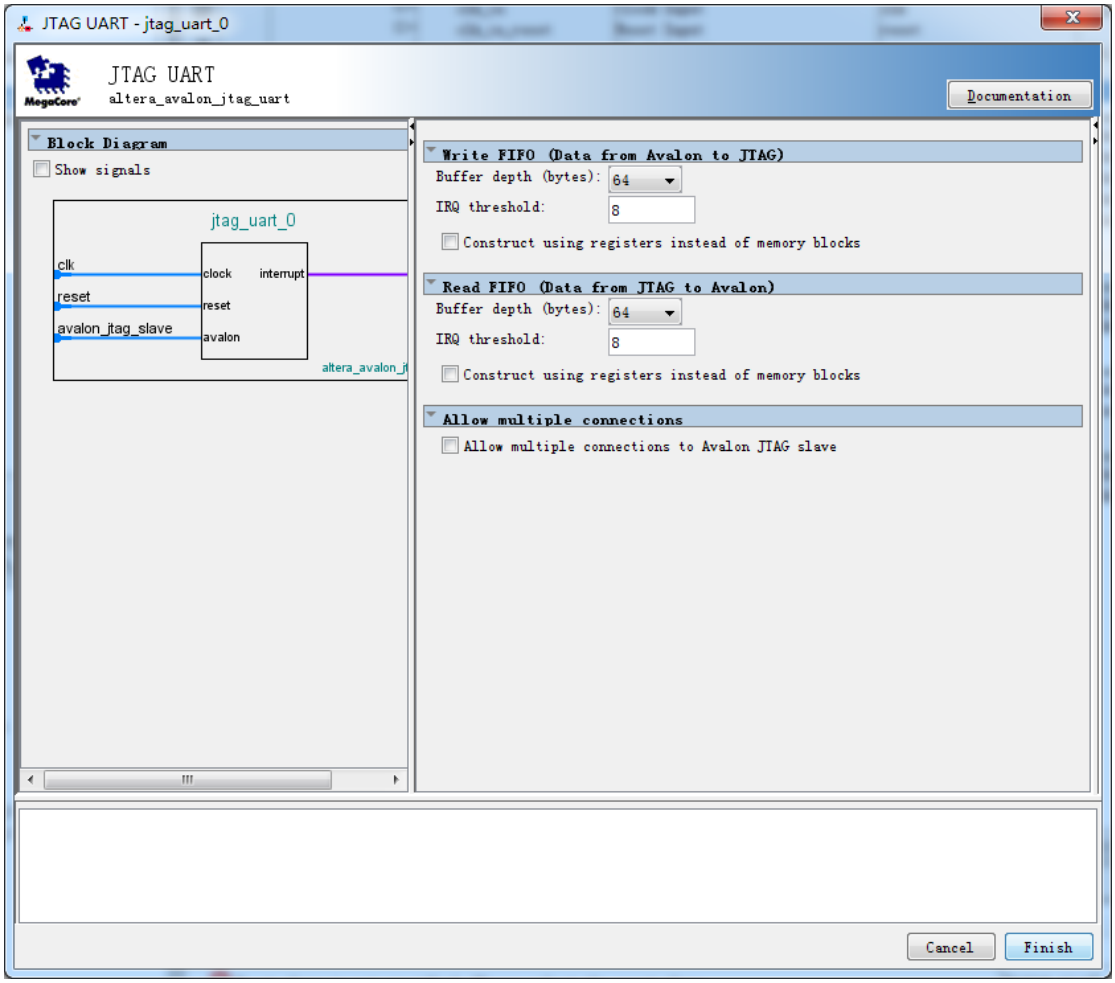


图 1.33 JTAG UART 配置向导页面

这里我们也可以直接使用默认设置，点击【Finish】完成配置即可，关于 JTAG UART 详细内容请参考 4.7 节中的介绍。

下面我们需要添加 System ID，这里我们有一点要注意，之前在 SOPC Builder 中 System ID 是自动生成的，但是在 Qsys 里已经不会再自动生成了，我们在 IP 核库中搜索到 System ID 后，双击 System ID，将会弹出 PIO 配置向导页面，如图 1.34 所示。

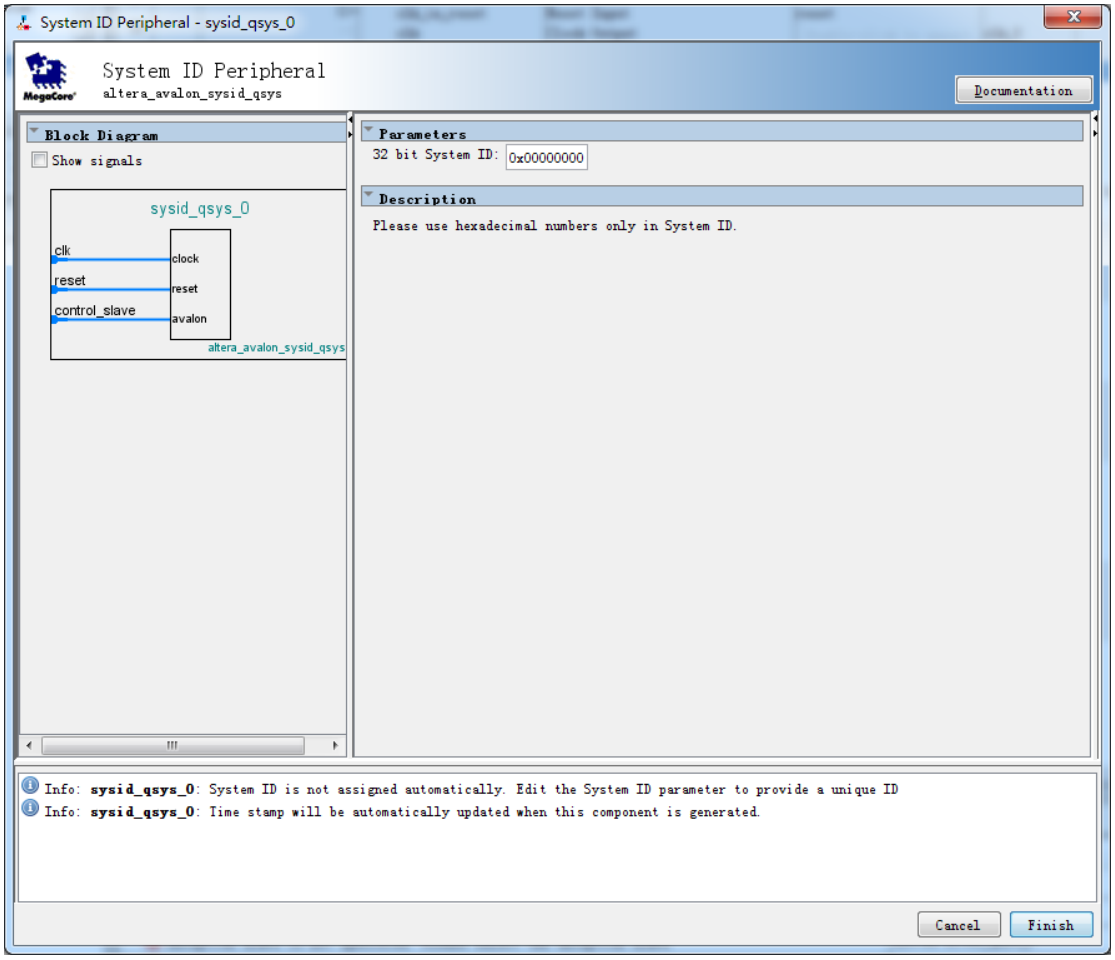


图 1.34 System ID 配置向导页面

这里我们使用默认配置，点击【 Finish 】完成配置即可，关于 System ID 详细内容请参考 4.5 节中的介绍。

下面我们需要添加 PIO，PIO 为 Nios II 处理器系统接收输入信号以及输出信号提供了一种简易的方法。我们需要在 Library 库中找到 PIO 组件，双击 PIO 组件，将会弹出 PIO 配置向导页面，如图 1.35 所示。

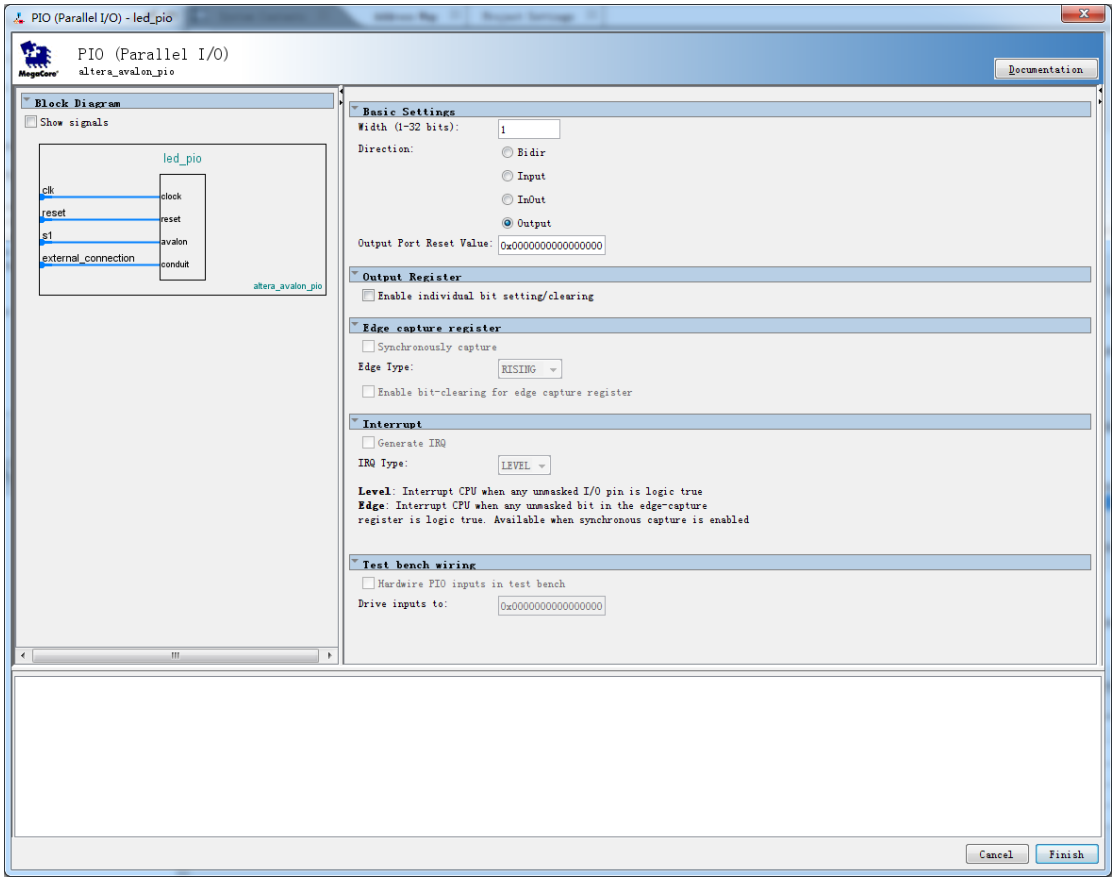


图 1.35 输出 PIO 的配置向导页面

从该图中可以看出，我们将宽度 8 改为了 1，其余的我们使用默认配置，点击【Finish】完成配置即可。接下来我们利用同样的方法再来添加一个 PIO，不过，这里要添加的 PIO 是和上面的 PIO 配置是完全不同的，上面的 PIO 我们用作了输出端口，而下面我们添加的 PIO 是用它来作为输入端口，并且还要带有中断功能，由于我们的按键悬空时是低电平，按下时是高电平，所以我们这里选择了边沿上升沿触发方式。关于PIO详细内容请参考4.1节中的介绍。我们直接给我 PIO 的配置页面向导图，如图 1.36 所示。

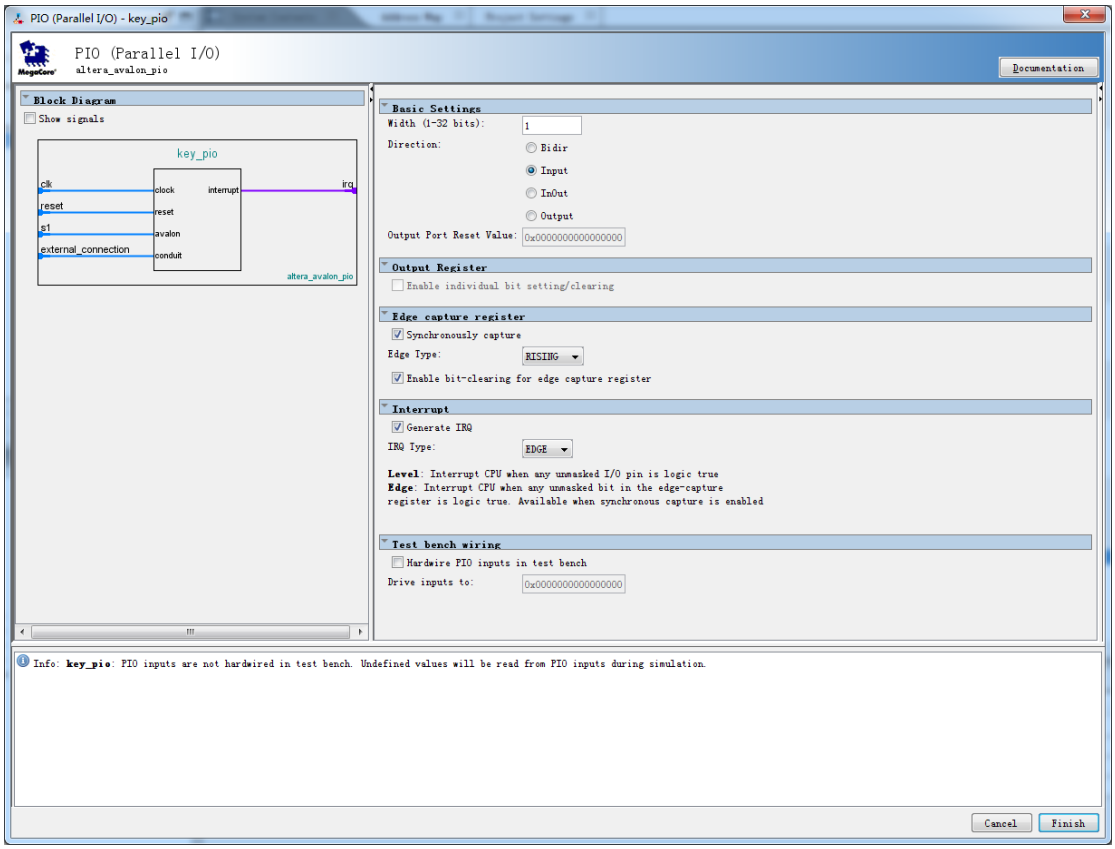


图 1.36 输入 PIO 的配置向导页面

完成了所有的组件添加以后我们需要对每个组件进行重命名，如图 1.37 所示。这里需要注意的是，由于命名没有一个规范，可以任意命名，我们尽量给出每个硬件外设的描述名称，因为在编写后面的用户程序中，我们会用到这些名称。

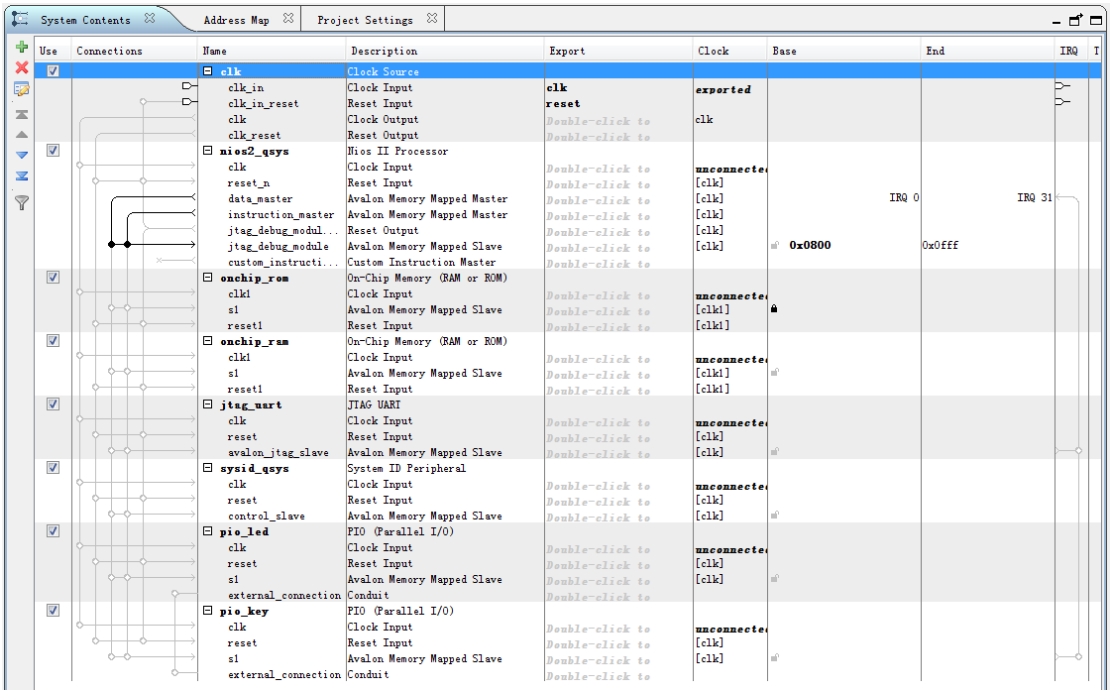


图 1.37 组件添加完毕后的页面

接下来我们就需要玩上一玩“连连看”，在之前的SOPC Builder版本中，添加完了组件以后，SOPC Builder会自动连接添加的组件，而在Qsys中，Qsys是不会自动连线的，这里我们需要我们手动连线，对于一个初用Qsys软件的读者来说，什么时候连接数据端口，什么时候连接指令端口是一件很头疼的事情，这里我们可以遵循以下规则：数据主端口连接存储器和外设元件，而指令主端口只连接存储器元件。关于指令和数据总线的详细内容可参考3.1.5节中的介绍。

因此，我们可以按照上述规则进行如下连线，如果是存储器类IP核，比如onchip_RAM和onchip_ROM等，我们需要将其Avalon Memory Mapped Slave端口连接到Nios II处理器核的data_master和 instruction_master端口上，如果是非存储器IP核，比如PIO外设，或者是System ID和JTAG UART等，我们只需要将其Avalon Memory Mapped Slave端口连接到Nios II处理器核的data_master端口上即可，而时钟端口和复位端口，我们需要全部连接，各个组件连接完毕如图1.38所示。

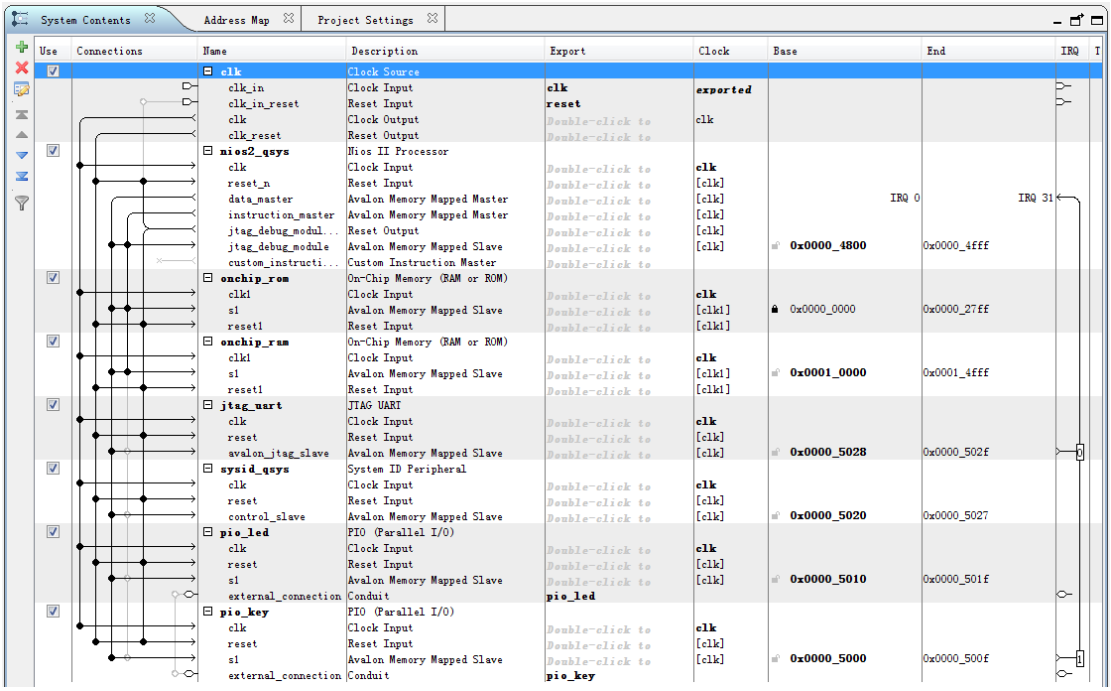


图1.38 端口连接完毕的组件页面

细心的读者会发现，在图1.38中不仅仅将线连接完毕，而且还多出了几个小细节，首先我们会发现在onchip_rom的地址上面，原本没有上锁的地址，却上了锁，其次在Export这一栏中出现了一个pio_led和pio_key的端口，这两个端口的作用是引出Nios II系统，如何引出呢，我们只需要在需要引出的端口上面双击即可，名字也是可以任意修改的。这里有一点需要注意，需要引出的端口是不需要连线的。最后一点，在IRQ一栏中出现了一个0和一个1，这里代表了我们为jtag_uart设置了一个0中断，为pio_key设置了一个1中断，关于中断的详细内容可参考3.1.4节中的介绍。接下来我们需要设置Nios II复位和异常地址，这里我们需要双击Nios II核，返回Nios II配置页面中对复位变量和异常变量进行设置，如图1.39所示。

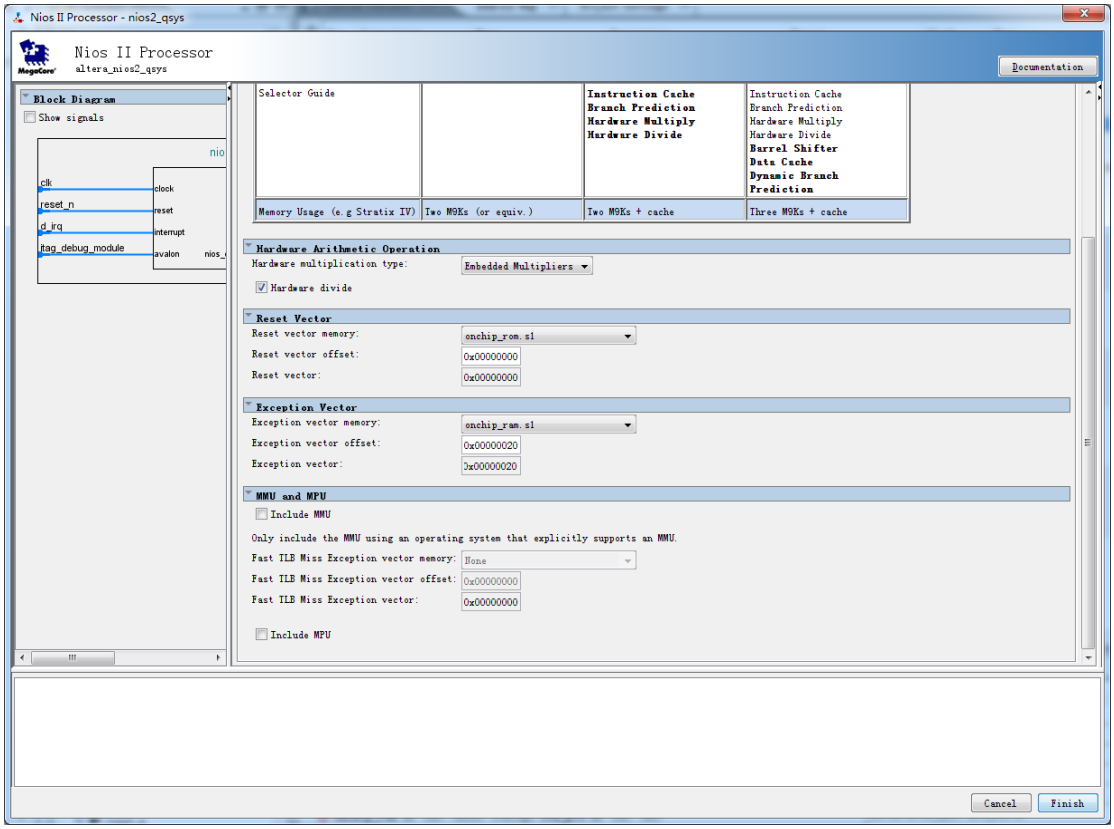


图 1.39 设置 Nios II 复位地址和异常地址

在这里，本系统上电后，从内部 ROM 开始运行，所以复位地址的 memory 设置为 onchip_ROM,offset 地址为 0x00000000。异常向量表放在内部 RAM 中，所以异常地址的 memory 设置为 onchip_RAM,offset 地址为 0x00000020。这里有一点要注意，Reset Address 和 Exception Address 的 Offset 只有在多处理器的系统中才进行设置，且其值必须是 0x20 的倍数。如果地址设置违反规则，信息窗口会给出错误提示。对于复位地址和异常地址具体的详细过程会在 3.3.1 节中介绍。

完成了复位地址和异常地址的设置后，Qsys 系统的设计基本上就要接近尾声了，返回到 Qsys 界面窗口，我们仍能够看到漫天红叉，接下来我们只需要轻轻一点，所有红叉将会消失，即可完成 Qsys 系统的设计。下面我们点击菜单栏中的【 System 】→【 Assign Base Addresses 】，Qsys 系统会自动为我们分配地址 Qsys 系统不仅仅只提供了自动分为基地址，还有自动分配中断等命令，对于许多系统，包括该设计，Assign Base Addresses 能满足要求。当然，用户可以自己调整基地址和中断优先级来满足系统的要求，Nios II 处理器内核可寻址 31 位地址范围。用户必须分配 0x00000000~0x7FFFFFFF 之间的基地址。由于 Qsys 不处理软件操作，所以它不能做出关于最好的中断分配。因此，我们这里建议用户不采用自动分配中断，而自己来分配中断优先级。

点击了自动分配地址以后，系统中的红叉统统消失不见了，下面我们就可以通过点击菜单栏中的【 Generate 】→【 Generate… 】选项来生成 Qsys 系统，如图 1.40 所示。

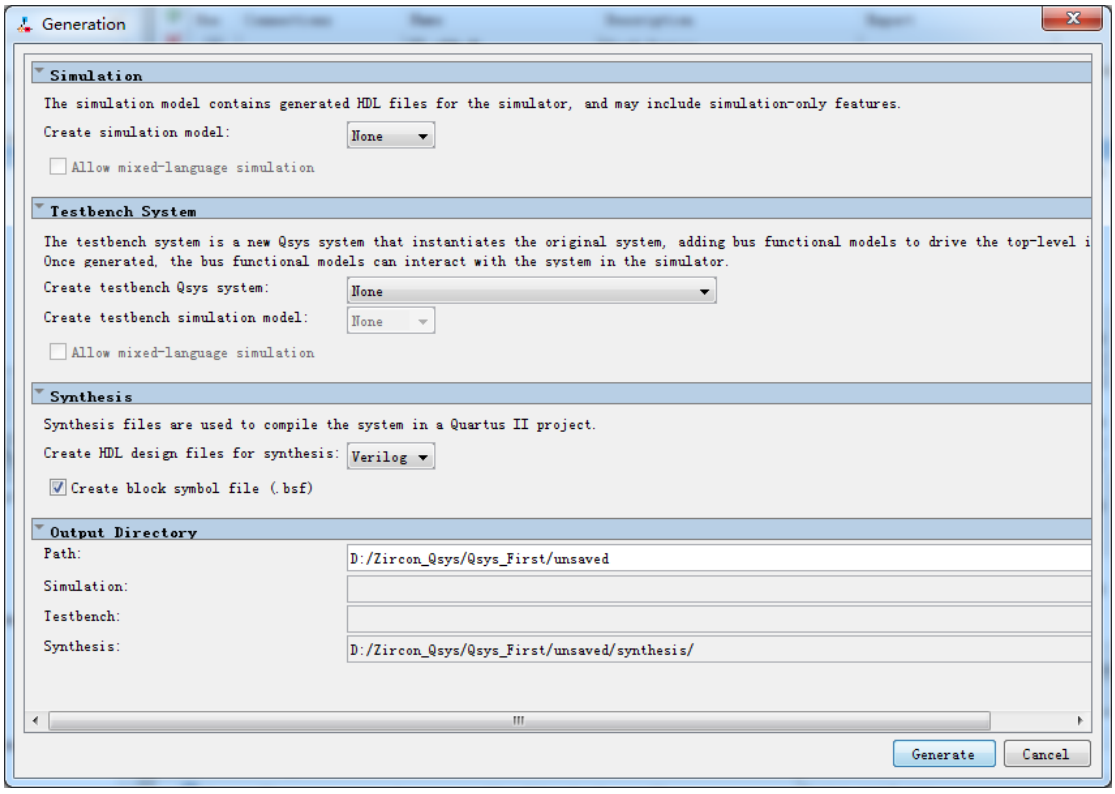


图 1.40 生成 Qsys 系统设置页面

由于我们不涉及仿真，所以我们便可以将 Simulation 和 Testbench System 都设为 None 即可。下面我们点击【Generate】，会提示如图 1.41 所示。

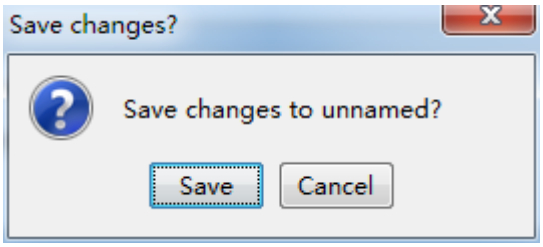


图 1.41 保存 Qsys 系统

我们单击【Save】，会给出保存路径对话框，如图 1.42 所示。这里我们就保存在我们的 Quartus II 工程“Qsys_First”文件夹下，系统可以任意命名，我们这里便命名为“Qsys_system”。

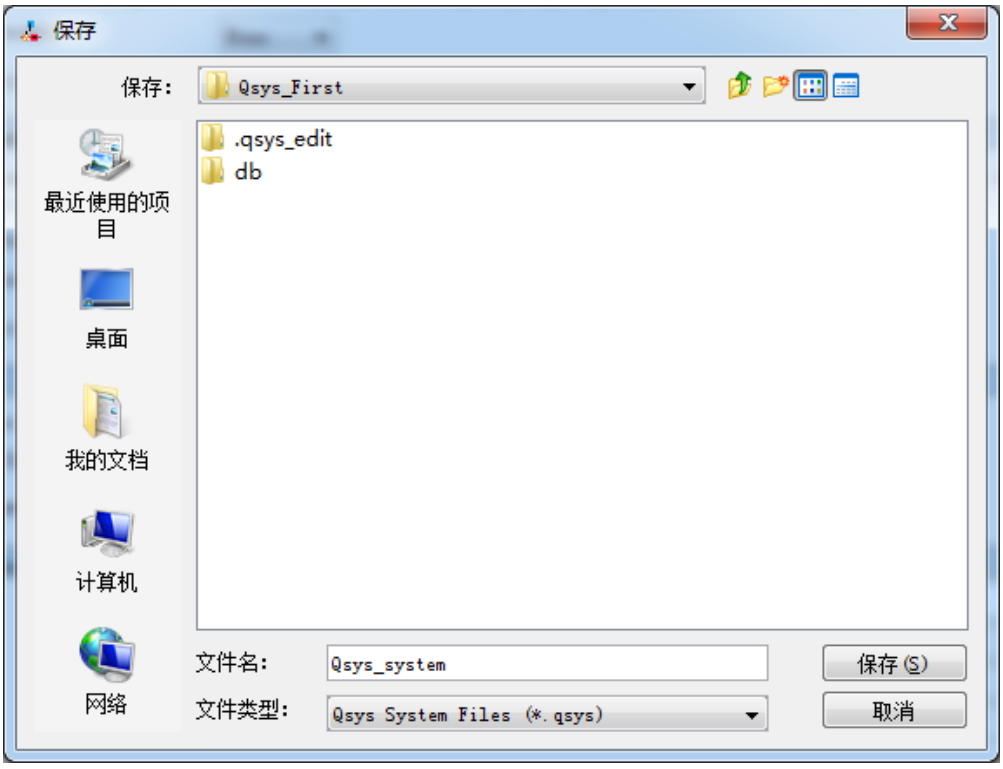


图 1.42 保存 Qsys 系统路径对话框

单击保存即可进行系统生成。在系统生成过程中，Qsys 会执行一系列操作，Qsys 会为添加的所有组件生成Verilog HDL源文件，并生成每个组件以及连接组件的片内总线结构、仲裁和中断逻辑。Qsys 会为系统生成 Nios II SBT for Eclipse 软件开发所需的硬件抽象层（HAL）、C 以及汇编头文件。这些头文件定义了存储器映射、中断优先级和每个外设寄存器空间的数据结构。关于硬件抽象层（HAL）详细内容请参考 3.2 节中的介绍。这样的自动生成过程有助于软件设计者处理硬件潜在的变化性，如果硬件改变了，Qsys 会自动更新这些头文件。Qsys 也会为系统中现有的每个外设生成定制的C和汇编函数库。如果添加了片内存储器，Qsys还将为片内ROM、RAM 生成其初始化所使用的HEX 文件。在生成阶段的最后一步，Qsys 创建适合与系统组件的总线结构，把所有的组件连接在一起。生成过程时间随计算机的性能而不同，电脑配置越高编译速度就会越快，一般需要几分钟。Generate Completed 中显示系统生成的过程。当系统生成结束后，会出现图 1.43 所示内容。

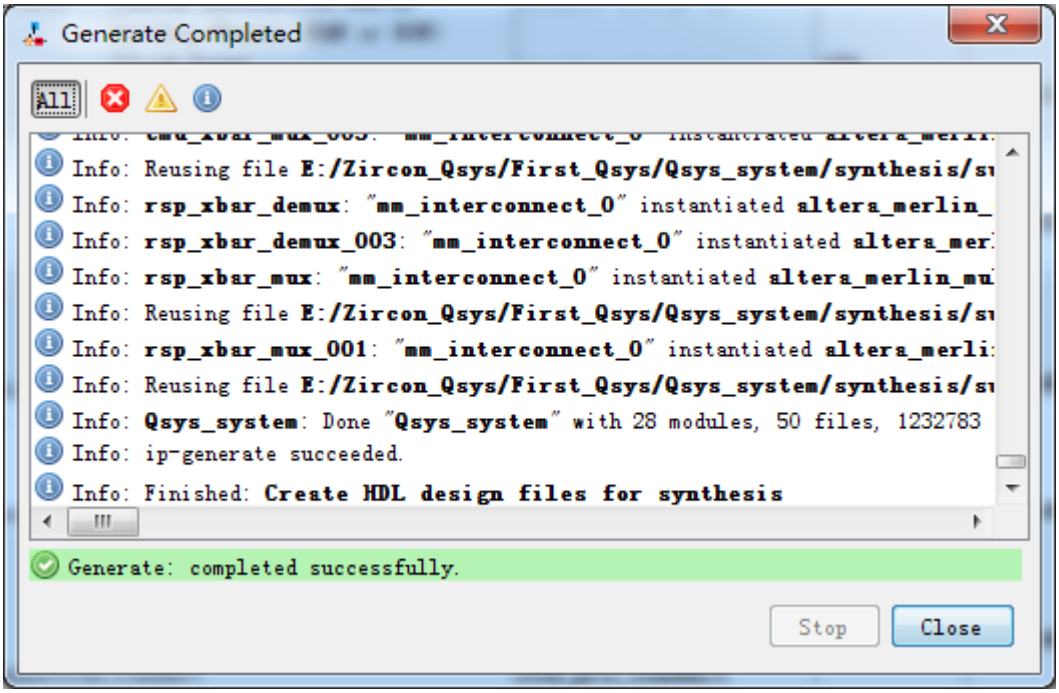


图 1.43 Qsys 系统成功生成

我们单击【 Close 】退出并返回到 Quartus II 软件即可。至此，已经完成 Qsys 系统的创建，生成系统结束后，要将系统集成到 Quartus II 硬件工程并使用 Nios II SBT for Eclipse 来开发软件，可以先将系统集成到 Quartus II 中，也可以先使用 Nios II SBT for Eclipse 进行软件开发。当然，如果是多人合作开发，两者可同时进行。接下面我们就来简单介绍一下 Qsys 系统为我们生成的若干文件，具体的文件目录结构如图 1.44 所示。

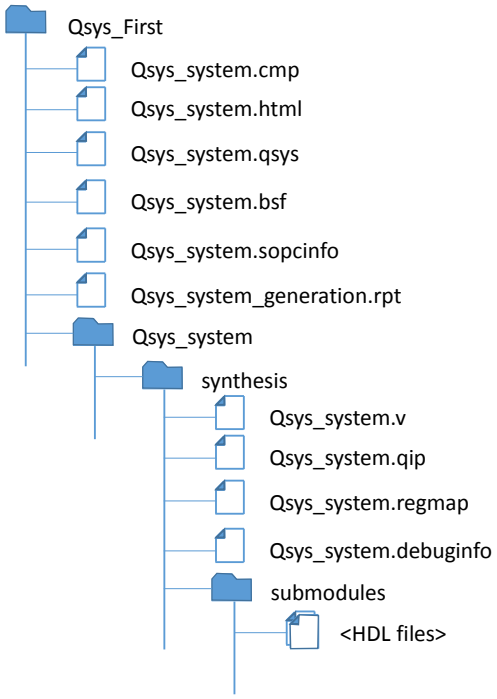


图 1.44 Qsys 系统生成的具体文件目录结构图

下面我们就来简单的介绍一下这些文件：

(1) Qsys_system.cmp：这是一个文本文件，包含本地通用端口定义，可用于 VHDL 设计文件中。

(2) Qsys_system.html：系统生成报告文件，主要包含了组件的连接，内存映射地址及参数分配等信息。

(3) Qsys_system.qsys：这个文件包含了 Qsys 系统中所有添加的 IP 核、及核连接和核参数。

(4) Qsys_system.bsf：这是 Qsys 生成的一个顶层的符号文件，用于添加到 Quartus II 工程顶层文件。

(5) Qsys_system.sopcinfo：这个文件包含了完整的 Qsys 系统描述，后续的软件项目依据此文件自动生相关的软件驱动。

(6) Qsys_system_generation.rpt：Qsys 生成日志文件，主要记录 Qsys 在系统生成中遇到的问题。

(7) Qsys_system.v：这个文件是描述 Nios II 系统的硬件设计文件。Quartus II 软件将使用这些 HDL 文件来编译整个 FPGA 设计。

(8) Qsys_system.qip：这是 Qsys 的 IP 核文件，在 Quartus II 工程中，不将这个文件加入到工程项目中，编译时会报错。

(9) Qsys_system.regmap：如果 IP 系统中包含注册信息，Qsys 生成 regmap 文件.regmap 文件描述的寄存器映射信息主站和从站接口。这个文件的补充文件.sopcinfo提供更详细的信息登记系统。这使得在系统控制台中注册显示视图和用户可自定义的统计数据。

(10) Qsys_system.debuginfo：包含后代信息。用于传递系统控制台和总线分析仪工具包的信息关于Qsys互联。总线分析工具使用该文件来确定调试组件在Qsys互联

在以上这么多的输出文件中，我们主要用到三个文件，它们分别是 Qsys_system.qsys、Qsys_system.qip 和 Qsys_system.sopcinfo。Qsys_system.qsys 主要用于 Qsys 系统的移植。Qsys_system.qip 主要用于 Quartus II 硬件工程项目中，而 Qsys_system.sopcinfo 则用于 Nios II SBT for Eclipse 软件工程项目中。

1.5.3 集成 Qsys 系统到 Quartus II 工程中

Qsys 系统建立完毕后，下面我们就需要来建立顶层文件，顶层模块用于将真个工程的各个模块包含在里面，Quartus II 编译时将这些模块整合在一起。顶层文件相当于传统电路设计中的电路板（PCB），用于将各种功能的芯片焊接在上面。我们选择菜单栏中的【File】→【New…】选项，则会弹出图 1.45 所示页面。

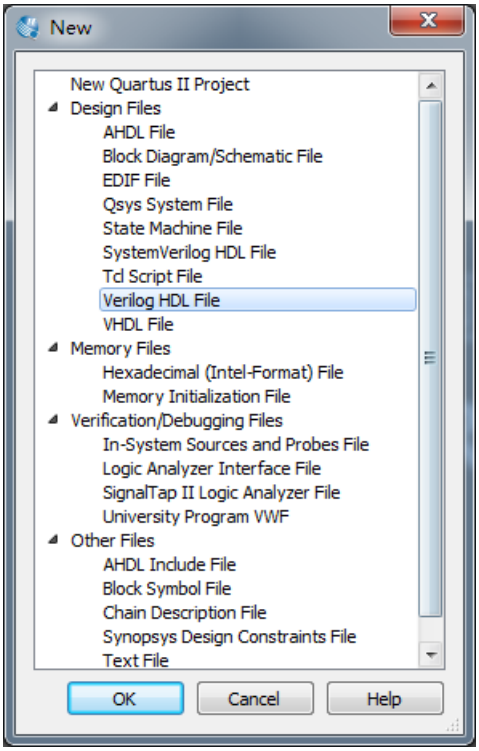


图 1.45 新建 Verilog 文件页面

这里需要注意的是，如果你喜欢以原理图方式管理顶层文件，那么你就要选择 Block Diagram/Schematic File作为顶层文件，如果你喜欢以代码方式管理顶层文件，那么就需要选择 Verilog HDL File 文件来作为顶层文件，两种方法都可以完成 Qsys 系统的创建。这里我们就以 Verilog HDL File 文件为例进行介绍，我们选中 Verilog HDL File 后，点击【 OK 】即可。接下来我们便需要在顶层文件中编写代码，如代码 1.1 所示。

代码 1.1 Qsys_First.v 代码

```
1  module Qsys_First
2  (
3    //输入端口
4    CLK_50M,RST_N,KEY_PIO,
5    //输出端口
6    LED_PIO
7  );
8
9  //-------------------------------------------------------------------------
10 //--    外部端口声明
11 //-------------------------------------------------------------------------
12 input                CLK_50M;
13 input                RST_N;
14 input                KEY_PIO;
15 output               LED_PIO;
16
17 //-------------------------------------------------------------------------
```

```
18  //--    内部端口声明
19  //------------------------------------------------------------------------------
20  wire                  clk_100m;
21
22  //------------------------------------------------------------------------------
23  //--    逻辑功能实现
24  //------------------------------------------------------------------------------
25  PLL                   PLL_Init
26  (
27     .inclk0            (CLK_50M    ),
28     .c0                (clk_100m   )
29  );
30
31  //------------------------------------------------------------------------------
32
33  Qsys_system           Qsys_system_Init
34  (
35     .clk_clk           (clk_100m   ),    //    clk.clk
36     .reset_reset_n     (RST_N      ),    //   reset.reset_n
37     .pio_led_export    (LED_PIO    ),    //pio_led.export
38     .pio_key_export    (KEY_PIO    )     //pio_key.export
39  );
40
41  endmodule
```

代码编写完毕后，我们就需要进行保存，我们可以点击菜单栏【 File 】→【 Sava 】进行保存，也可以点击快捷栏中的保存图标进行代码保存，单击保存出现如图 2.32 所示。

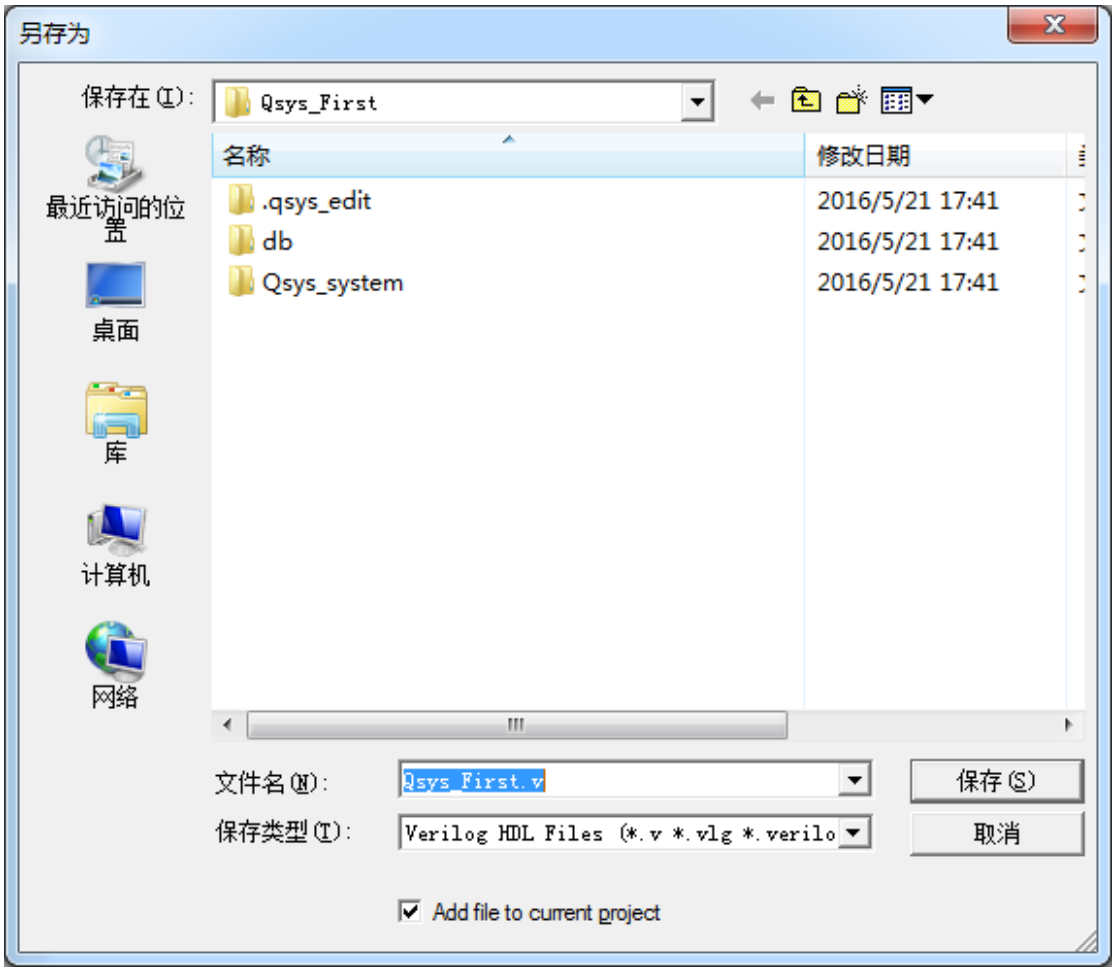


图 1.46 保存顶层文件路径

在上述代码中，我们主要介绍两点：第一点是 PLL IP 核是如何使用的，这里我们并不给出详细讲解，只给出创建过程，想要深入了解 PLL IP 核的读者，可以参考《软件工具篇中》的介绍来进一步学习。第二点是 Qsys_system 代码是如何编写的。首先我们介绍的是 PLL IP 核的创建。在 Quartus II 软件中，我们单击菜单栏中的【Tools】→【MegaWizard Plug-In Manager】出现图 1.47 所示页面。

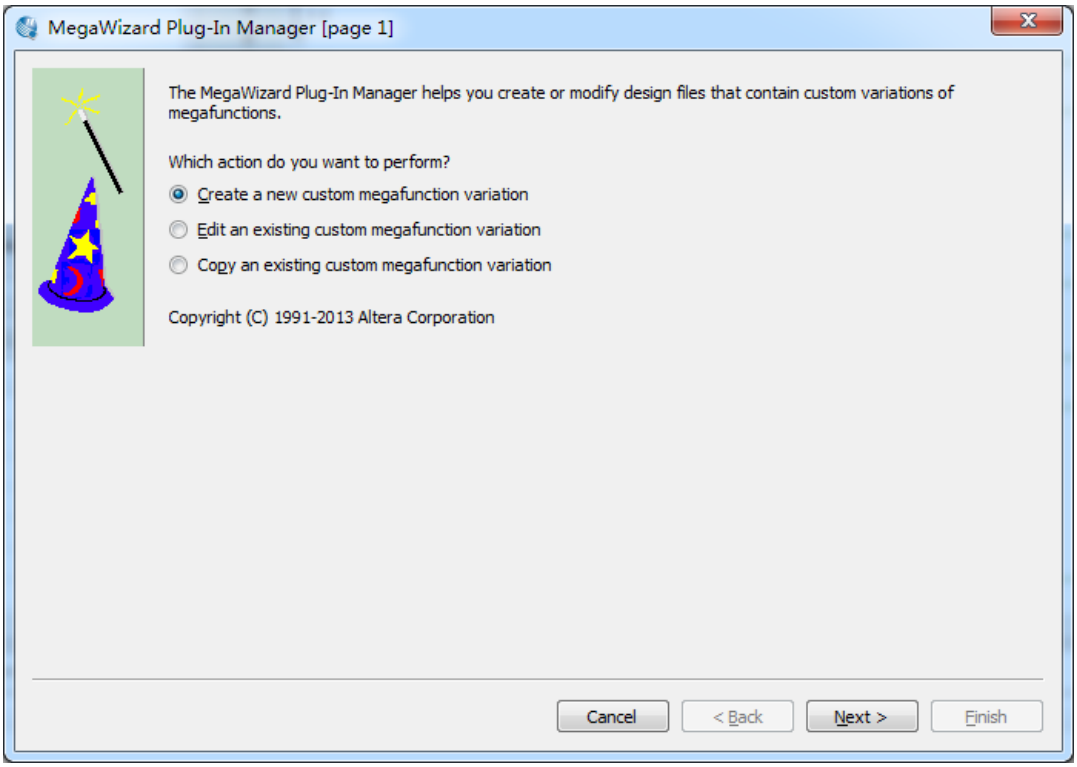


图 1.47 创建 IP 核向导页面

我们这里直接点击【 Next> 】, 我们可以通过搜索栏进行搜索 ALTPLL IP 核, 也可以直接找到 I/O 文件夹下的 ALTPLL IP 核, 这里我们需要给 PLL IP 核命名, 这里同样也可以任意命名, 我们这里就命名为 PLL。如图 1.48 所示。

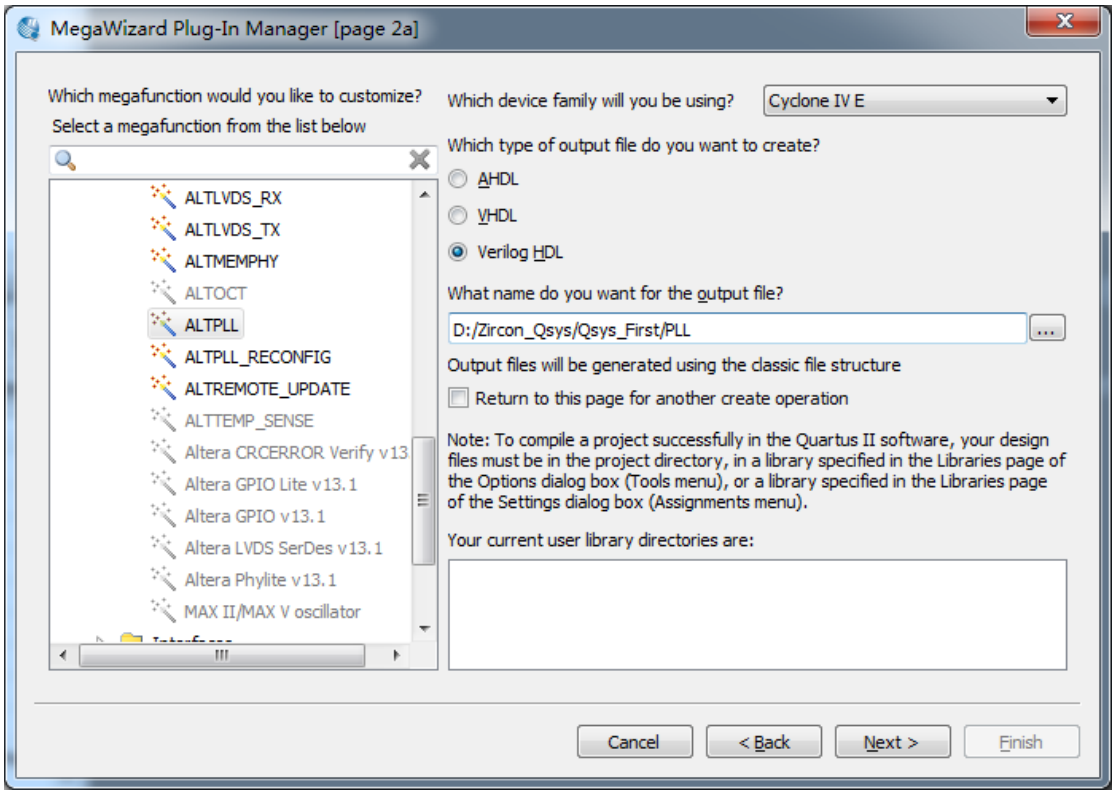


图 1.48 选择创建 PLL 向导页面

接下来我们点击【Next>】进入 PLL IP 核配置向导页面，如图 1.49 所示。由于我们使用的开发板晶振时钟为 50MHz，所以我们这里 PLL IP 核的时钟输入需要设置成 50Mhz。

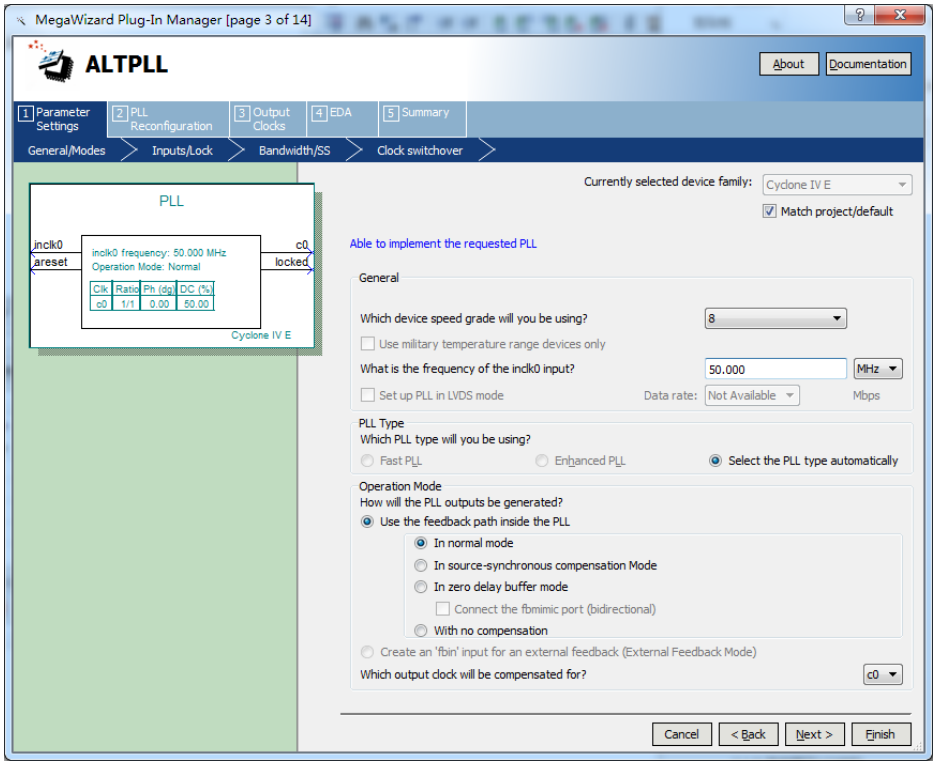


图 1.49 PLL IP 核输入时钟配置向导页面

接下来我们点击【Next>】进入“Inputs/Lock”页面，如图 1.50 所示。由于我们这里用不到上面的两个功能，所以我们这里直接将勾号取消掉。

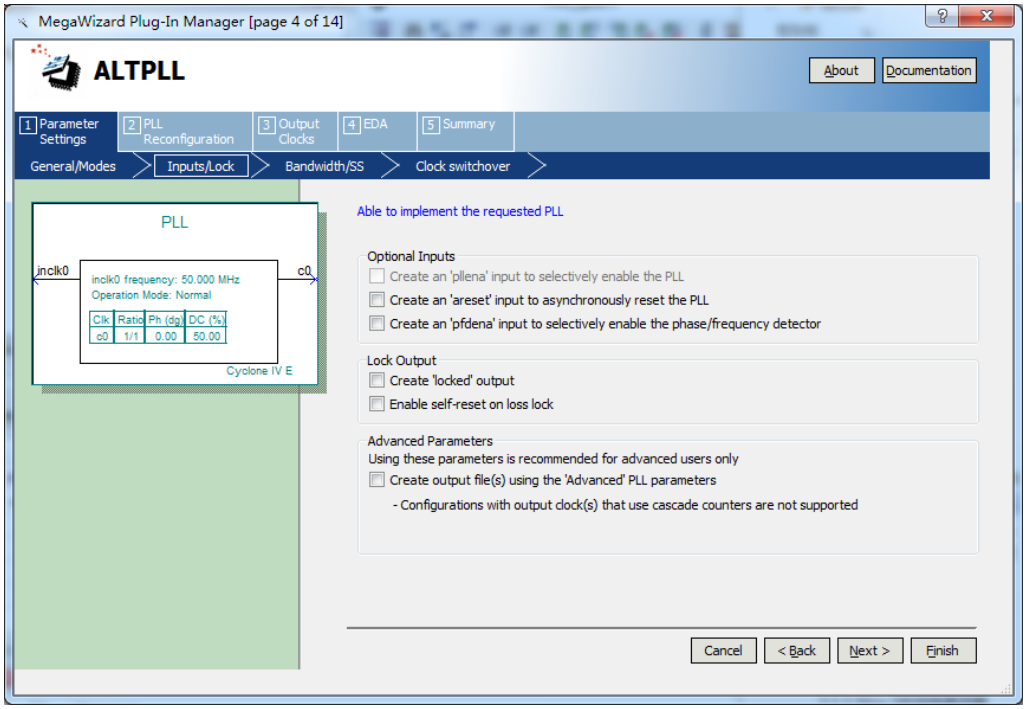


图 1.50 PLL IP 核“Inputs/Lock”配置向导页面

接下来我们一路狂点【Next>】，直到进入“Output Clocks”页面，如图 1.51 所示。由于我们的 Qsys 系统的时钟频率为 100MHz，我们这里的输出时钟也需要设置成 100MHz，以供给 Qsys 系统使用。

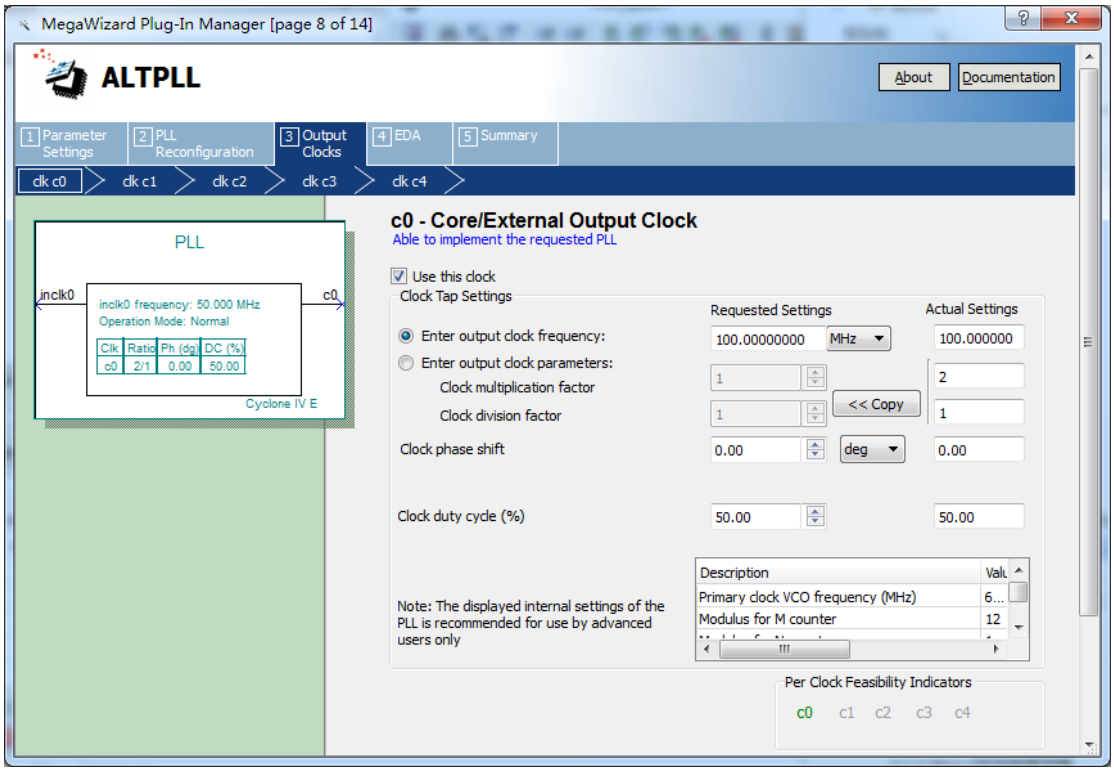


图 1.51 PLL IP 核输出时钟配置向导页面

接下来我们便可以直接点击【Finish】，进入最后的总结页面。如图 1.52 所示。这里我们需要将 PLL_inst.v 勾选上，PLL_inst.v 文件是 ALTPLL IP 核自动生成的使用模板代码，我们只需要将模板中的代码复制到顶层文件中修改端口便可以直接使用。

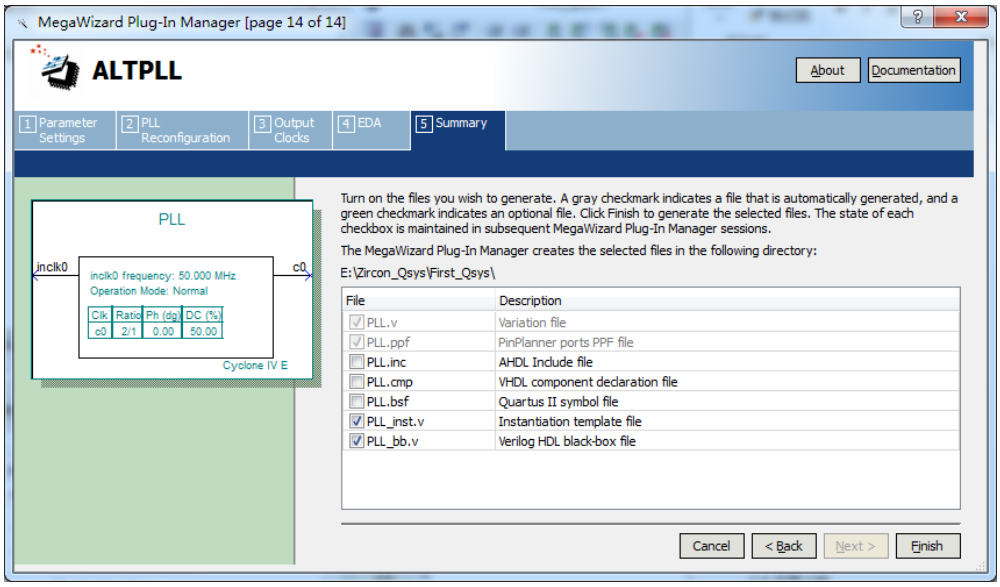


图 1.52 PLL IP 核最后的配置总结页面

我们点击【Finish】完成 PLL IP 核的创建。Quartus II 软件会自动将我们创建的 PLL IP 核自从添加到工程项目中，我们可以在工程文件夹 Qsys_First 中找到生成的 PLL_inst.v 文件，这里我们就不在给出代码内容以做对比。

接下来我们要介绍的是，第二点是 Qsys_system 代码是如何编写的，我们返回到 Qsys 系统软件中，我们单击菜单栏中的【Generate】→【HDL Example…】选项，弹出图 1.53 所示内容。我们只需要点击【Copy】，便可以直接复制到顶层文件中进行修改。

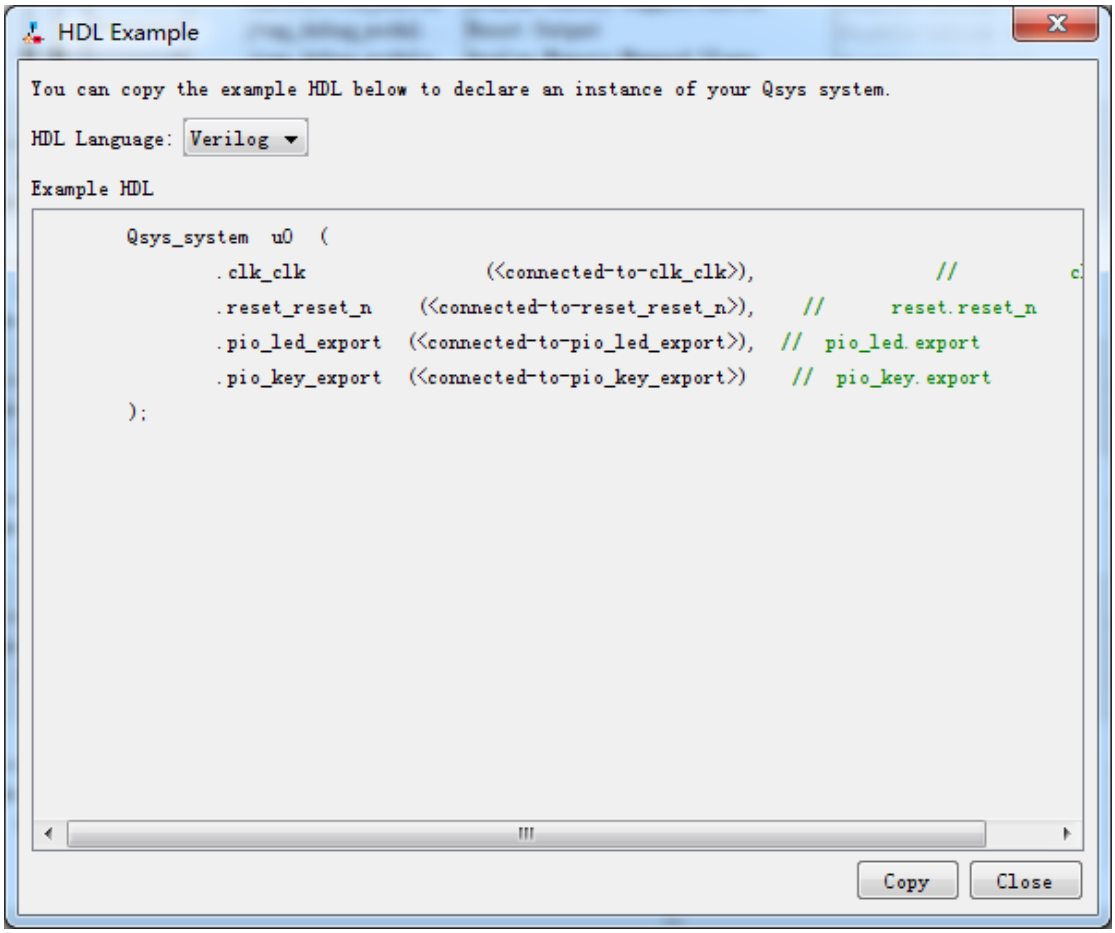


图 1.53 Qsys 系统自动生成的 HDL 代码

接下来我们就需要将 Qsys 系统添加到 Quartus II 工程项目中，进行编译，看看有没有出现一些由于操作失误而引起的小错误。我们可以通过 Quartus 软件菜单栏【Assignment】→【Settings】进入图 1.54 所示页面。

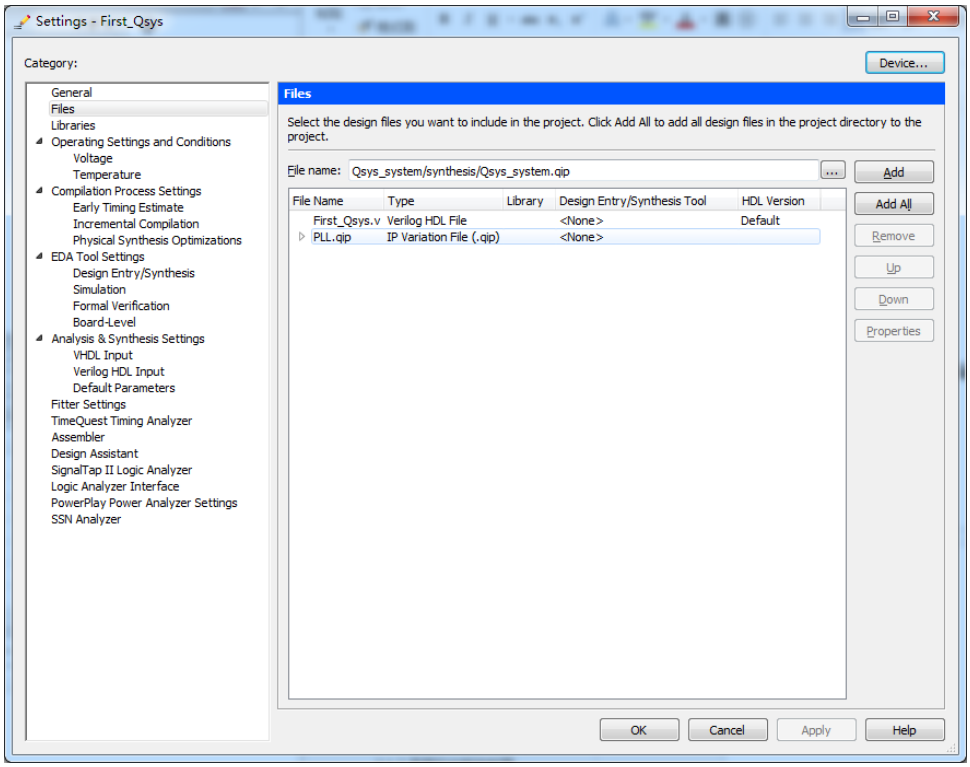


图 1.54 将 Qsys 系统添加到 Quartus 项目工程中

我们点击【…】找到 Qsys_system/synthesis/Qsys_system.qip，这里要注意，一定要记得点击【Add】后，再去点击【OK】，否则将会出现没有添加成功尴尬场面。添加完毕后，我们会在 Project Navigator 窗口中看到我们工程中的所有的文件，如图 1.55 所示。

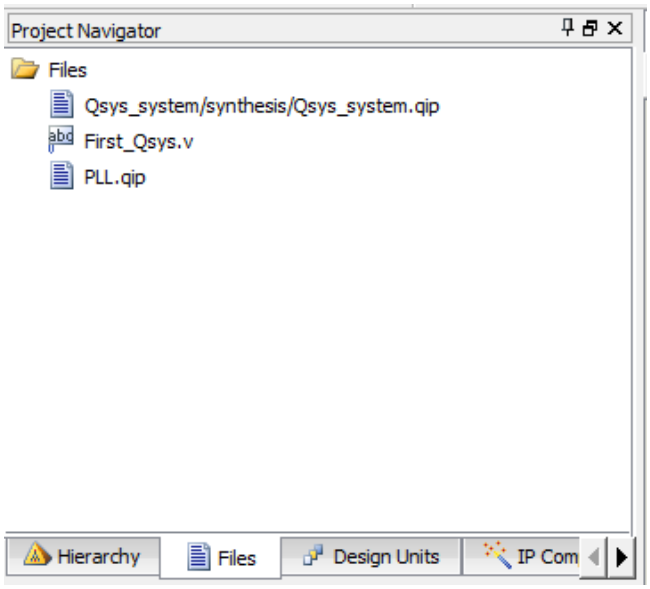


图 1.55 Project Navigator 窗口页面

这时，我们便能够进行编译查错了，我们可以通过 Quartus II 软件菜单栏中的【Processing】→【Start Compilation】来进行编译，也可以通过快捷栏中的快捷键进行编译，这里的编译速度也是随着电脑性能的越高而越快。我们稍等片刻后。编译就会完成，出现如图 1.56 所示。

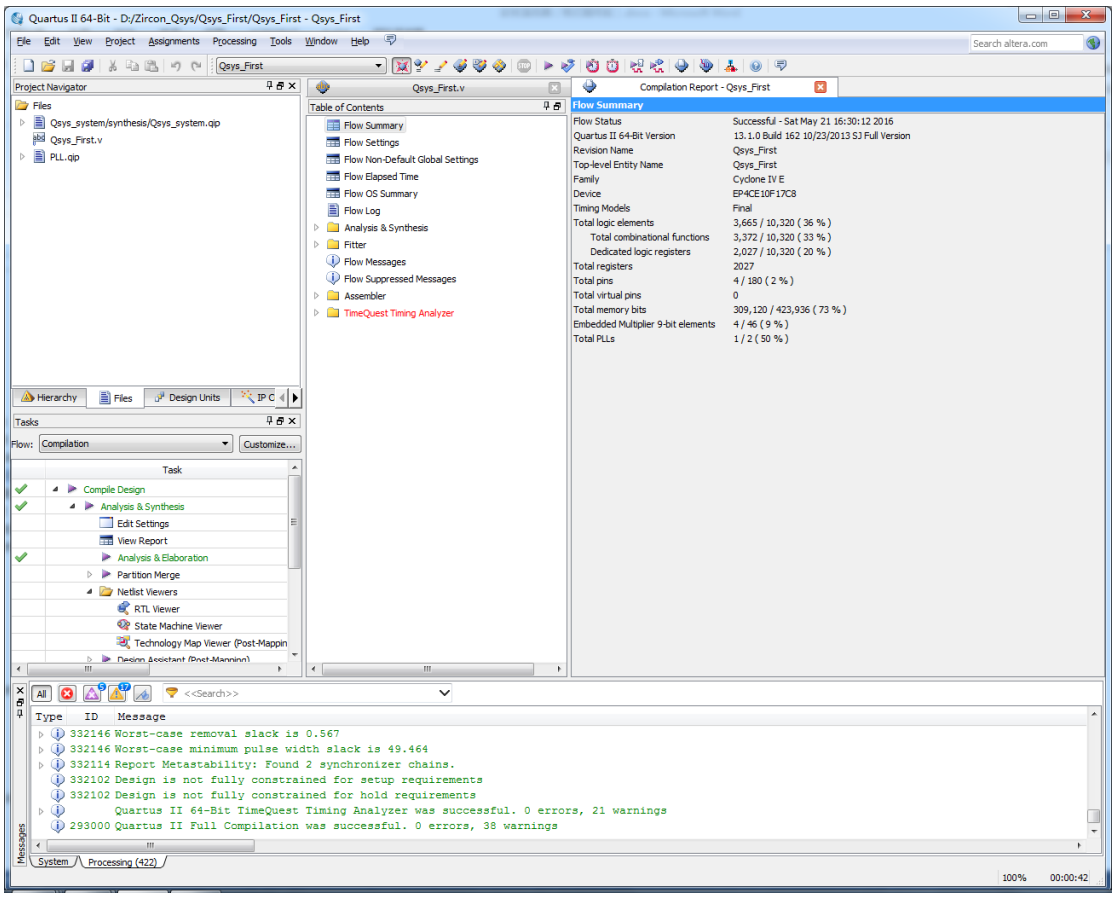


图 1.56 编译完成生成报告窗口页面

接下来我们就需要进行 Quartus II 软件中的扫尾工作，配置 IO，分配管脚，下面我们依次进行设置。首先我们可以通过 Quartus II 软件菜单栏中的【Assignment】→【Device】，然后我们在 Device 界面中找到【Device and Pin Options…】进入图 1.57 所示页面配置 IO。

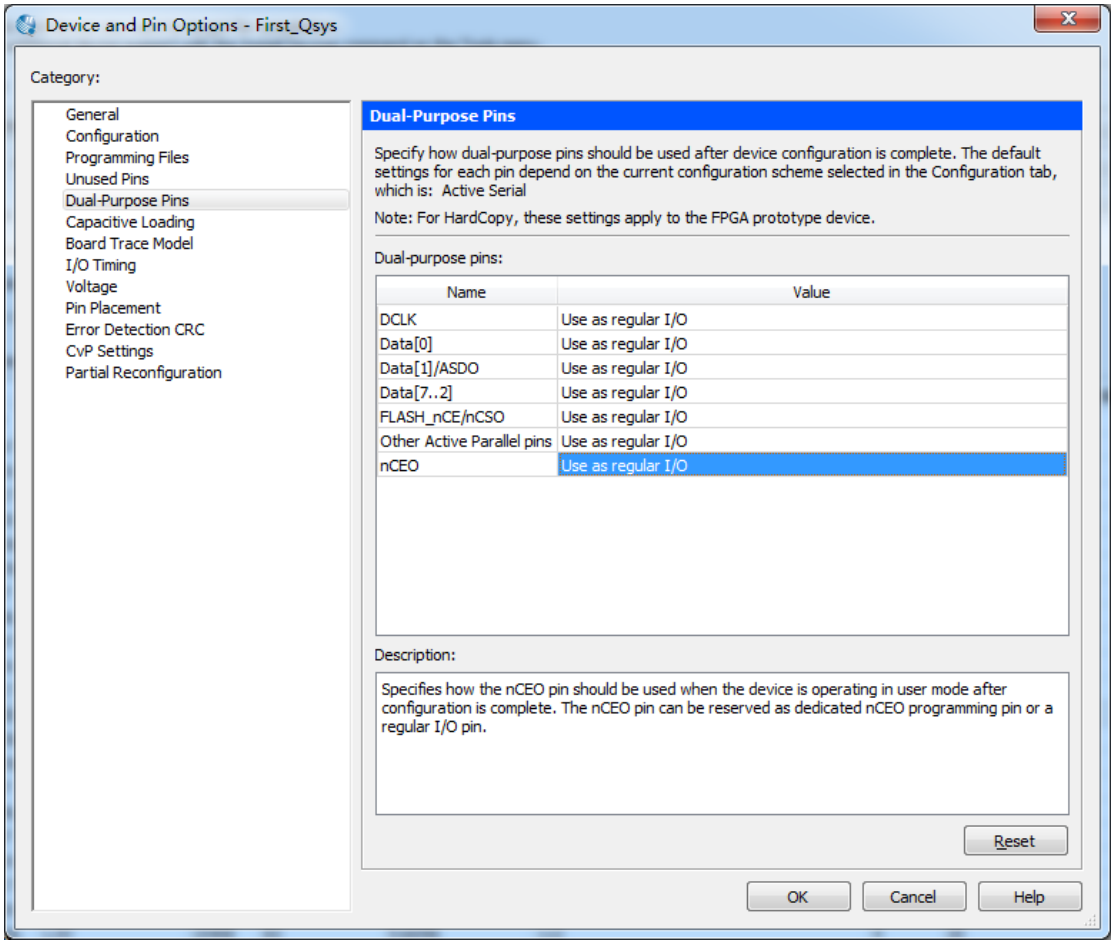


图 1.57 设置 IO 界面

接下来我们便将进入 Unused Pins 页面，对没有使用引脚的进行设置，按照图 1.58 所示，将未使用引脚设置为高阻输入，这样上电后 FPGA 的所有不使用引脚都将进入高阻抗状态。

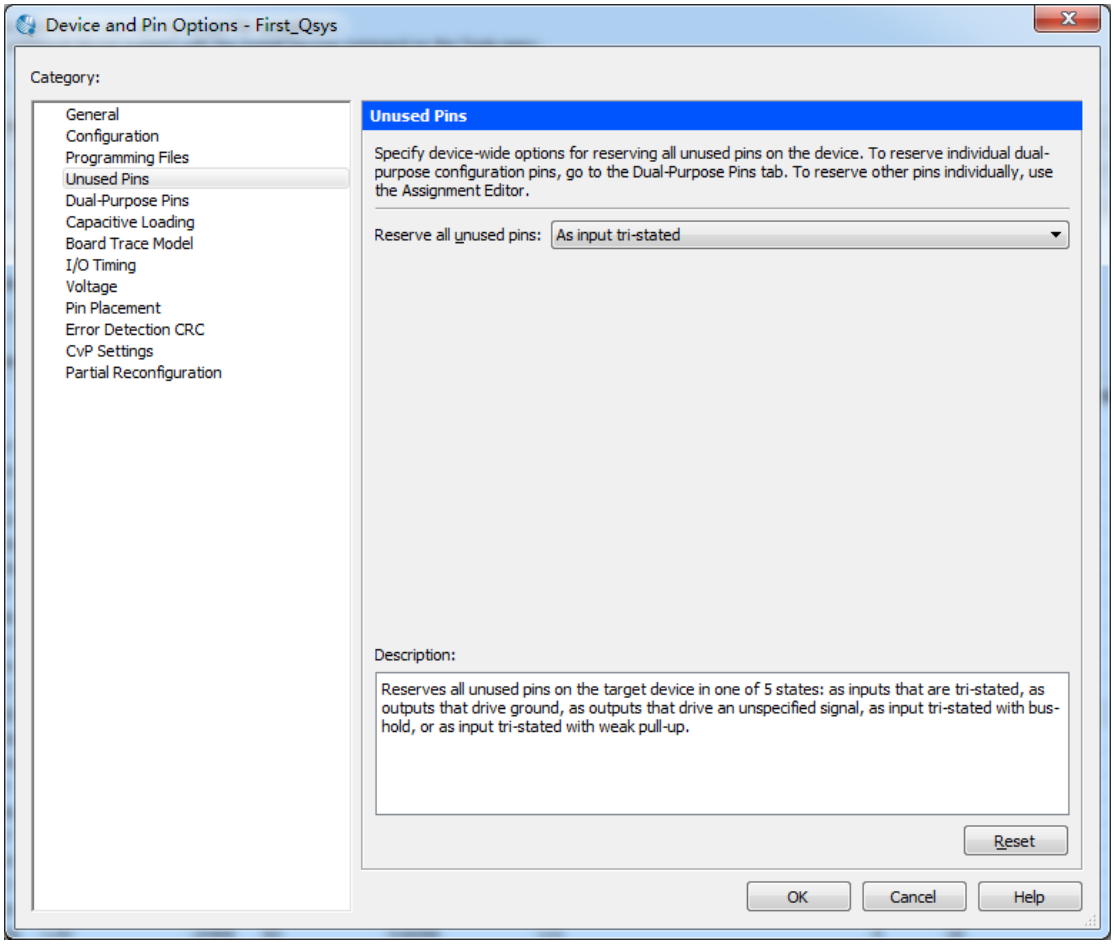
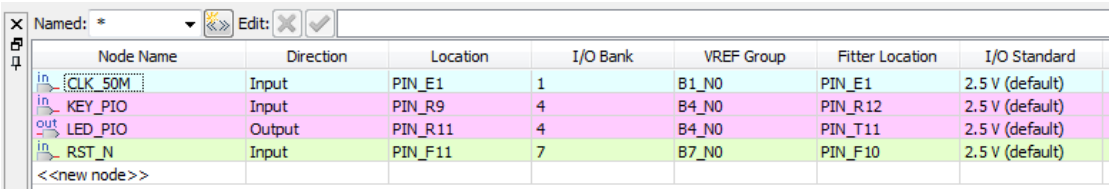


图 1.58 未使用引脚设置界面

最后我们通过 Quartus II 软件菜单栏中的【 Assignments 】→【 Pin Planner 】选项进行分配管脚，如图 1.59 所示。



Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard
CLK_50M	Input	PIN_E1	1	B1_N0	PIN_E1	2.5 V (default)
KEY_PIO	Input	PIN_R9	4	B4_N0	PIN_R12	2.5 V (default)
LED_PIO	Output	PIN_R11	4	B4_N0	PIN_T11	2.5 V (default)
RST_N	Input	PIN_F11	7	B7_N0	PIN_F10	2.5 V (default)
<<new node>>						

图 1.59 管脚分配界面

最后我们在进行一次全编译，在编译过程中，可能产生很多警告信息，但这些不会影响设计结果。成功编译硬件系统后，将产生 Qsys_First.sof 的 FPGA 配置文件输出。下面我们就来讲解一下将.sof 文件下载到目标 FPGA 器件的步骤。

(1) 通过 USB-Blaster 下载器连接开发板和计算机，接通 A4 开发板电源，如图 1.60 所示；

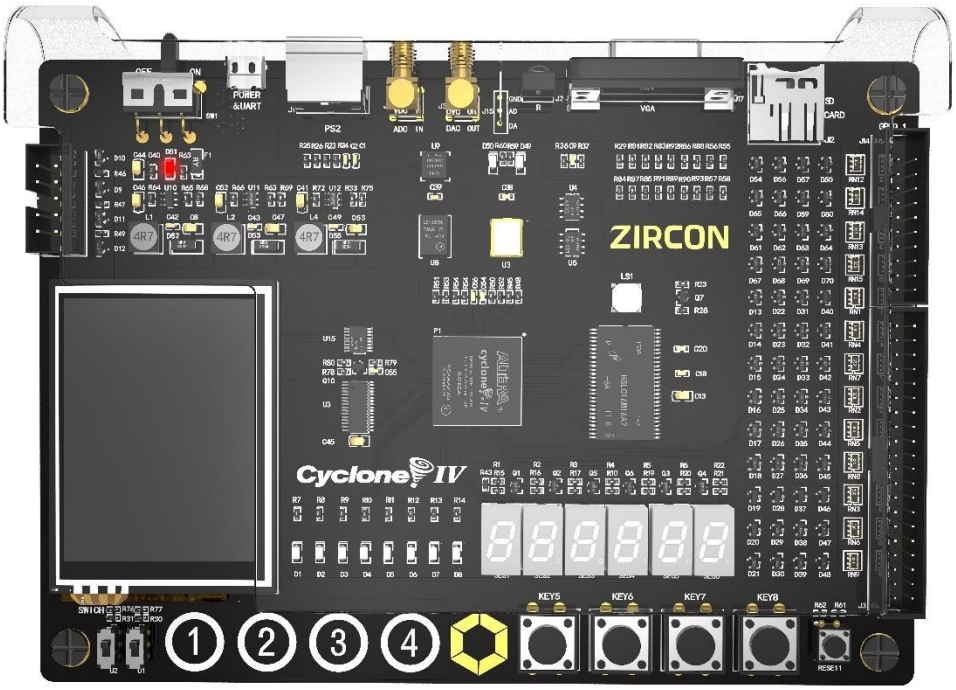


图 1.60 接通 A4 开发板电源示意图

(2) 在 Quartus II 软件中选择【 Tools 】→【 Programmer 】, 打开编程器窗口后确保编程器窗口的 Hardware Setup 栏中硬件已经安装。如果硬件没有安装,可以参考《 驱动安装指导手册 》中的介绍。如图 1.61 所示;

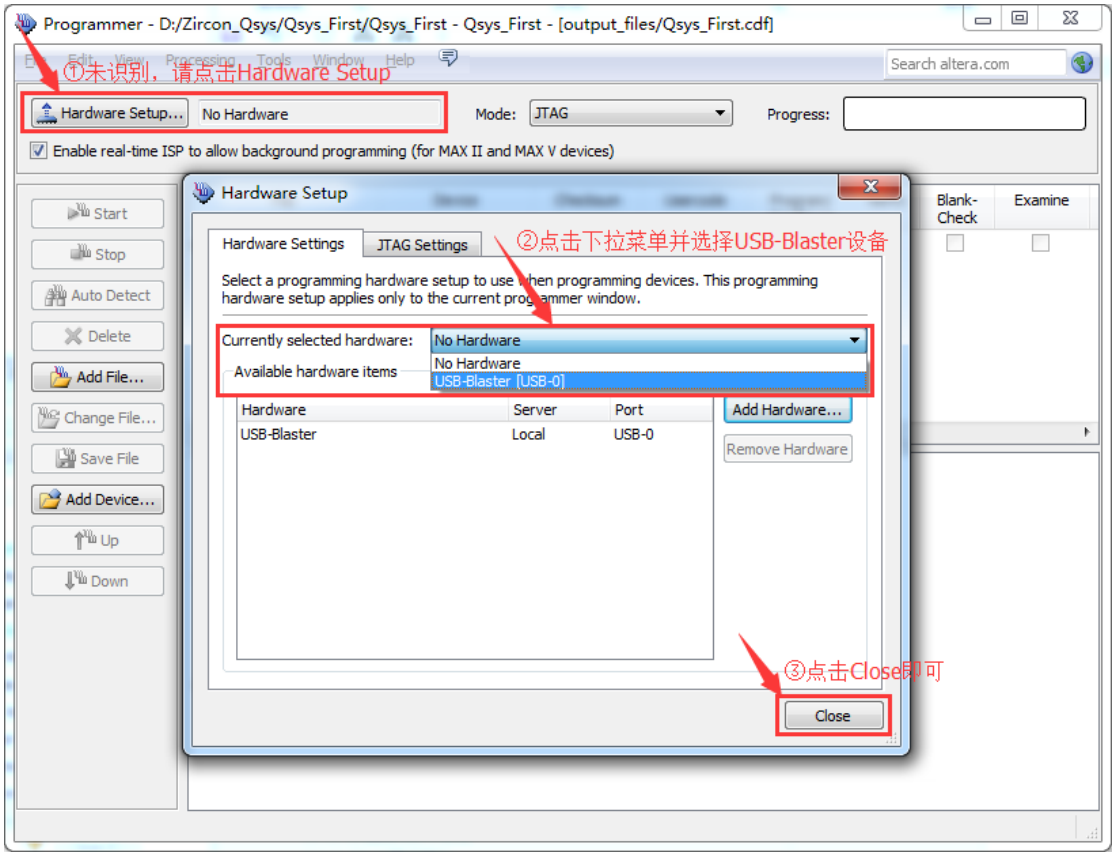


图 1.61 USB-Blaster 设备连接图

(3) 打开配置文件（Qsys_First.sof），确认下载模式，确保 Program/Configure 下的方框选中，最后点击【Start】按钮，开始使用配置文件对 FPGA 进行配置，Progress 栏显示配置进度，如图 1.62 所示。

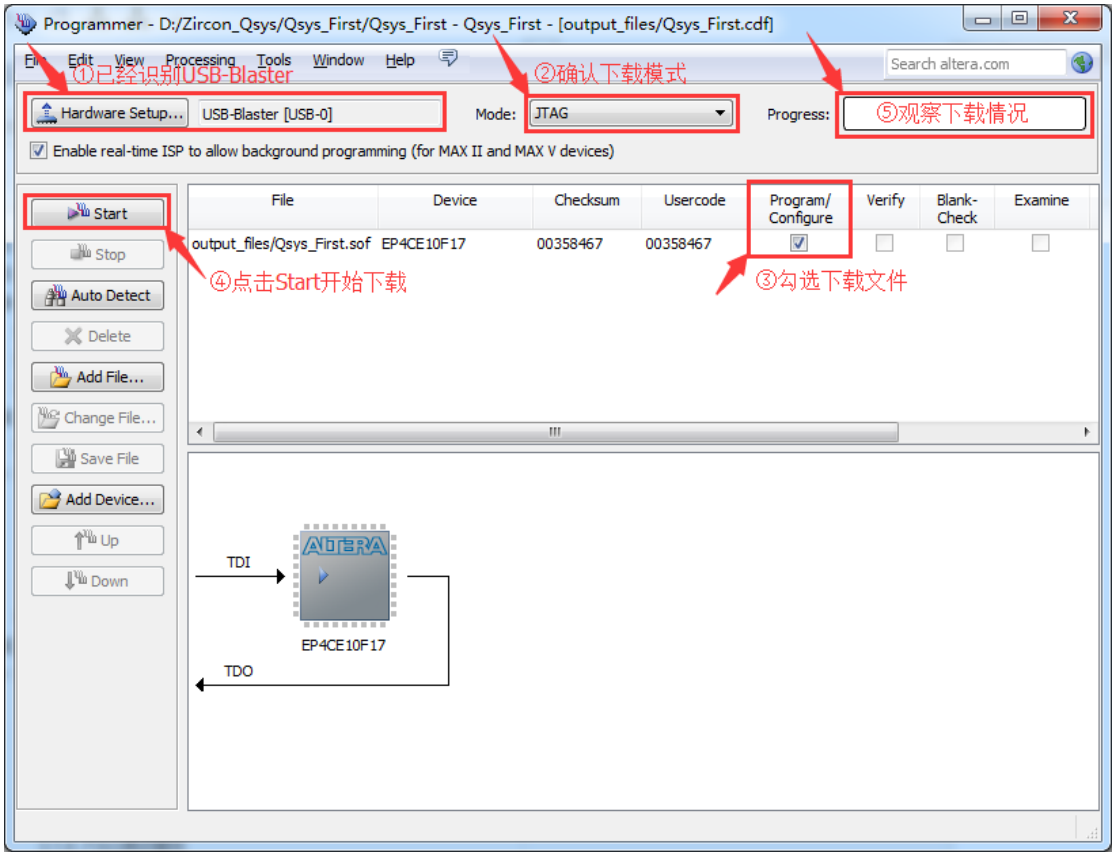


图 1.62 设备连接完成下载图

本节只讲述了将配置文件下载到 FPGA 中，掉电后 FPGA 中的配置数据将丢失，可以将配置文件写入掉电保持的 EPCS 器件。在上电时使用 EPCS 对 FPGA 进行配置，详细内容见 2.3 节中的介绍。至此，硬件部分设计完成，下面进行基于 Nios II SBT for Eclipse 的软件方面的设计。