

第 1 节 ISE 套件的介绍与安装

4.1.1 ISE 简要介绍

Xilinx 是全球领先的可编程逻辑完整解决方案的供应商，研发、制造并销售应用范围广泛的高级集成电路、软件设计工具以及定义系统级功能的 IP（Intellectual Property）核，长期以来一直推动着 FPGA 技术的发展。Xilinx 的开发工具也在不断地升级，由早期的 Foundation 系列逐步发展到目前的 ISE 9.1i 系列，集成了 FPGA 开发需要的所有功能，其主要特点有：

- 包含了 Xilinx 新型 SmartCompile 技术，可以将实现时间缩减 2.5 倍，能在最短的时间内提供最高的性能，提供了一个功能强大的设计收敛环境；
- 全面支持 Virtex-5 系列器件（业界首款 65nm FPGA）；
- 集成式的时序收敛环境有助于快速、轻松地识别 FPGA 设计的瓶颈；
- 可以节省一个或多个速度等级的成本，并可在逻辑设计中实现最低的总成本。

Foundation Series ISE 具有界面友好、操作简单的特点，再加上 Xilinx 的 FPGA 芯片占有很大的市场，使其成为非常通用的 FPGA 工具软件。ISE 作为高效的 EDA 设计工具集合，与第三方软件扬长补短，使软件功能越来越强大，为用户提供了更加丰富的 Xilinx 平台。

4.1.2 ISE 功能简介

ISE 的主要功能包括设计输入、综合、仿真、实现和下载，涵盖了 FPGA 开发的全过程，从功能上讲，其工作流程无需借助任何第三方 EDA 软件。

- 设计输入：ISE 提供的设计输入工具包括用于 HDL 代码输入和查看报告的 ISE 文本编辑器（The ISE Text Editor），用于原理图编辑的工具 ECS（The Engineering Capture System），用于生成 IP Core 的 Core Generator，用于状态机设计的 StateCAD 以及用于约束文件编辑的 Constraint Editor 等。
- 综合：ISE 的综合工具不但包含了 Xilinx 自身提供的综合工具 XST，同时还可以内嵌 Mentor Graphics 公司的 LeonardoSpectrum 和 Synplicity 公司的 Synplify，实现无缝链接。
- 仿真：ISE 本身自带了一个具有图形化波形编辑功能的仿真工具 HDL Bencher，同时又提供了使用 Model Tech 公司的 Modelsim 进行仿真的接口。
- 实现：此功能包括了翻译、映射、布局布线等，还具备时序分析、管脚指定以及增量设计等高级功能。
- 下载：下载功能包括了 BitGen，用于将布局布线后的设计文件转换为位流文件，还包括了 ImPACT，功能是进行设备配置和通信，控制将程序烧写到 FPGA 芯片中去。
- 使用 ISE 进行 FPGA 设计的各个过程可能涉及到的设计工具如表 4-1 所示。

表 4-1 ISE 设计工具表

设计输入	综合	仿真	实现	下载
HDL文本编辑器	XST		Translate	
ECS原理图编辑器	FPGA Express	HDL Bench	MAP	BitGen
StateCAD状态机编辑器	(Synplify	(Model Sim)	Place and Route	IMPACT
Core Generator	LeonardoSpectrum)		Xpower	
Constraint Editor				

4.1.3 ISE 软件的安装

E9.1 软件安装的基本硬件要求如下：CPU 在 P III 以上，内存大于 256M，硬盘大于 4G 的硬件环境安装。为了更好地使用软件，至少需要 512M 内存，CPU 的主频在 2GHz 以上。本书使用的集成开发环境是 ISE 9.1，仿真工具是 ModelSim 6.2b，综合工具为 Synplify Pro 8.8。其中 ISE、ModelSim 软件和 Synplify 软件不同版本之间的差异不是很大，所以操作和设计结果的差别也是很小的。具体安装过程如下：

1. 光盘放进 DVD 光驱，等待其自动运行（如果没有自动运行，直接执行光盘目录下的 Setup.exe 文件程序即可），会弹出图 4-1 所示的欢迎界面，点击“Next”进入下一页。

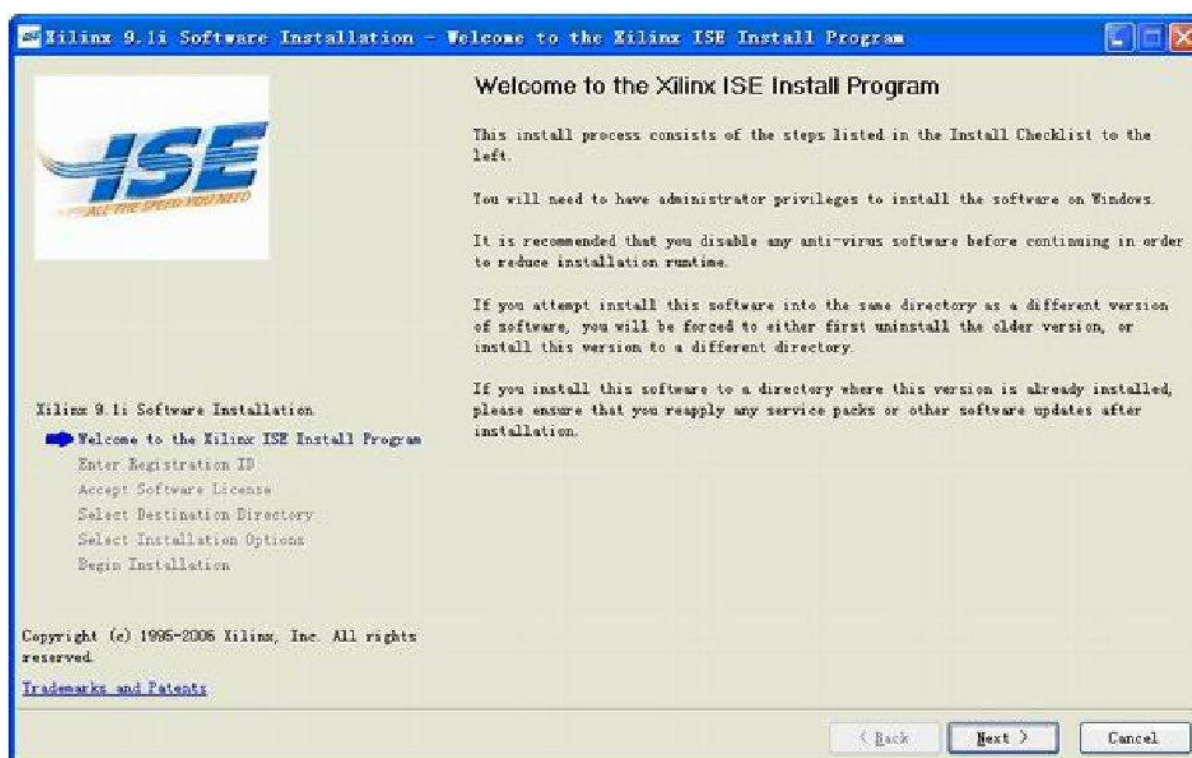


图 4-1 ISE 安装过程的欢迎界面

2. 接着进入注册码获取、输入对话框，如图 4-2 所示。注册码可以通过网站、邮件和传真方式申请注册码。如果已有注册码，输入后单击“Next”按键后继续。

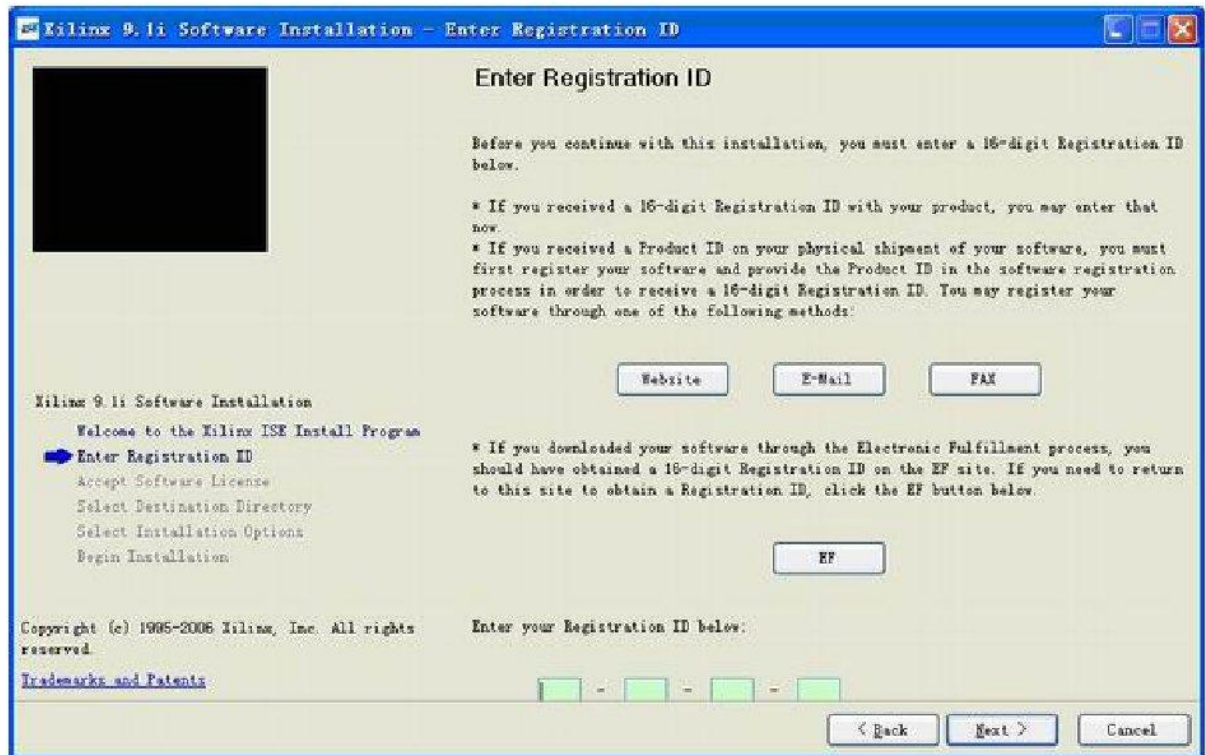


图 4-2 ISE9.1 安装程序的注册码输入界面

购买了正版软件后，最常用的方法就是通过网站注册获取安装所需的注册码。首先在Xilinx的官方主页www.xilinx.com上建立自己的帐号，然后点击图 4-1 中的“Website”按键，登陆帐号，输入CD盒上的产品序列号（序号的格式为：3 个字符+9 个数字），会自动生成 16 位的注册码，直接记录下来即可，同时Xilinx网站会将注册码的详细信息发送到帐号所对应的邮箱中。

3. 下一个对话框是Xilinx软件的授权声明对话框，选中“I accept the terms of this software license”，单击“Next”后进入安装路径选择界面，如图 4-3 所示。单击“Browse”按键后选择自定义安装路径，单击“Next”按键继续

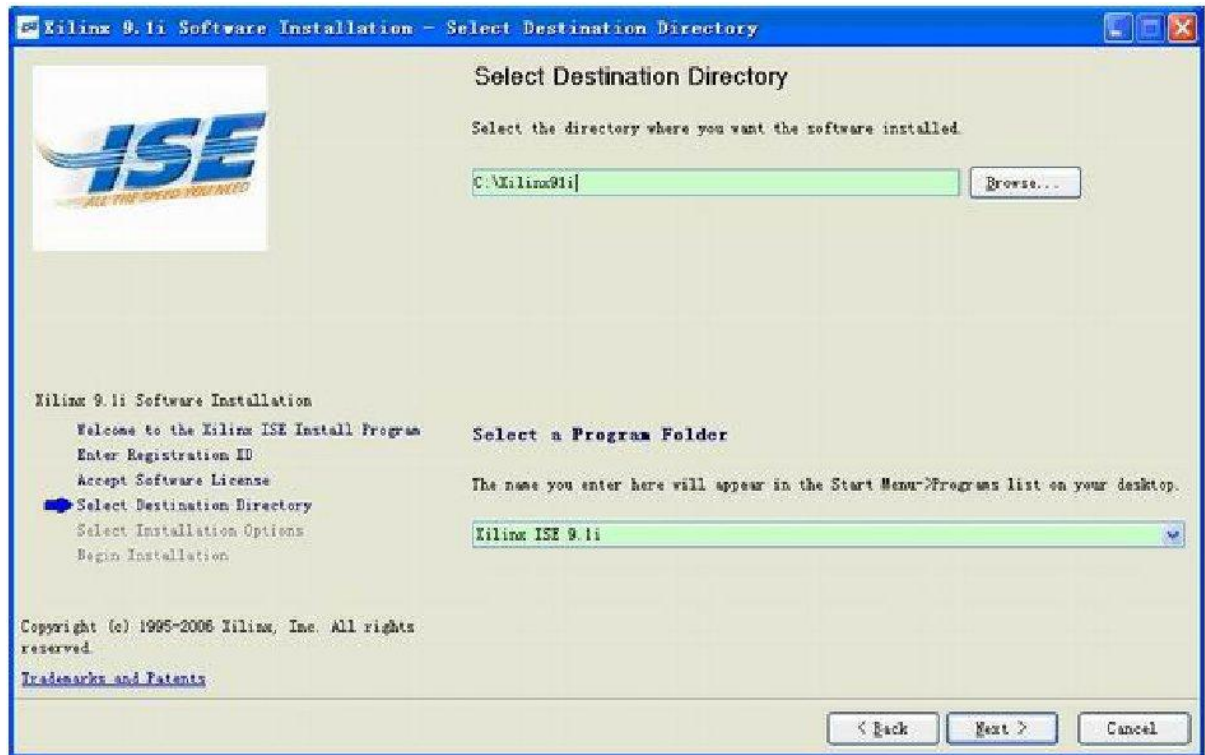


图 4-3 ISE 软件安装路径选择对话框

4. 接下来的几个对话框分别是选择安装组件选择，如图 4-4 所示，用户需要选择自己使用的芯片所对应的模块，这样才能在开发中使用这些模块。在计算机硬盘资源不紧张的情况下，通常选择“Select All”。

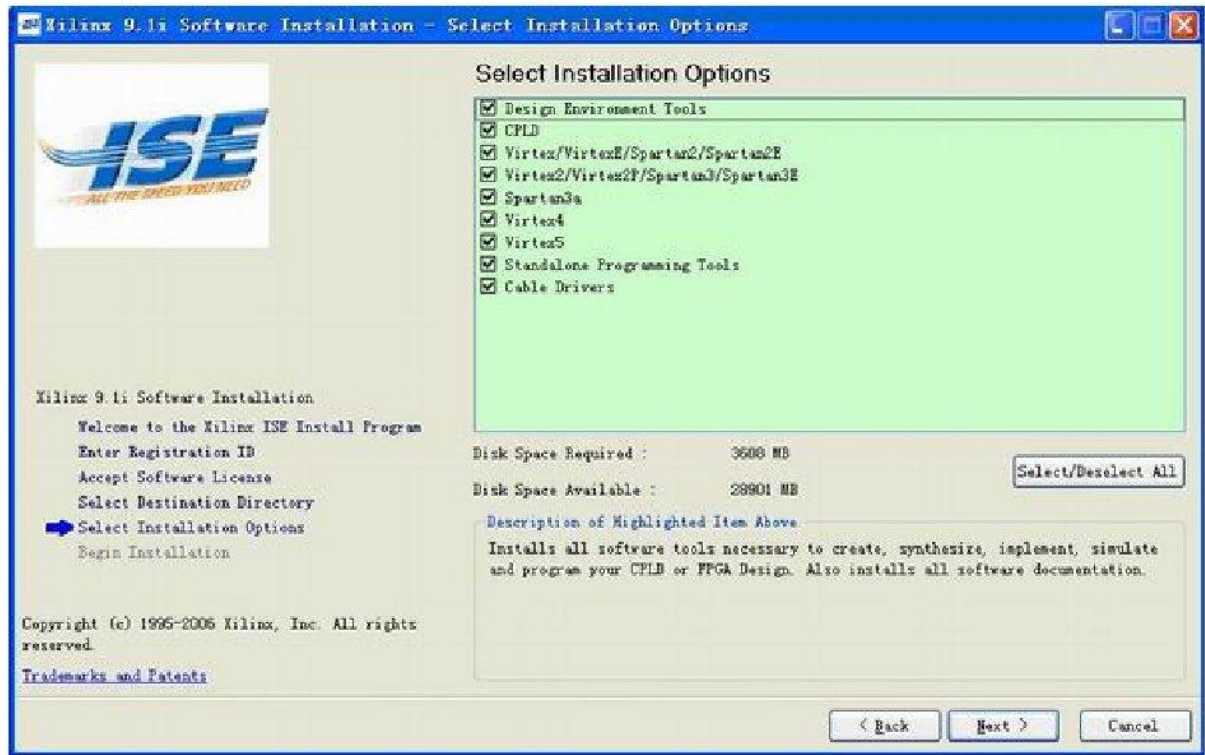


图 4-4 ISE 安装组件选择界面

5. 随后进入设置环境变量页面，保持默认即可。如果环境变量设置错误，则安装后不能正常启动 ISE。选择默认选项，安装完成后，在“我的电脑”上单击右键，选择属性 环境变量中，可看到名为“Xilinx”的环境变量，其值为安装路径。最后进入安装确认对话框，单击 **Install** 按钮，即可按照用户的设置自动安装 ISE，如图 4-5 所示。



图 4-5 ISE 安装进程示意图

6. 安装完成后，会在桌面以及程序菜单中添加 Project Navigator 的快捷方式。双击即可进入 ISE 集成开发环境。

4.1.4 ISE 软件的基本操作

1. ISE 用户界面

ISE9.1i 的界面如图 4-6 所示，由上到下主要分为标题栏、菜单栏、工具栏、工程管理区、源文件编辑区、过程管理区、信息显示区、状态栏等 8 部分。

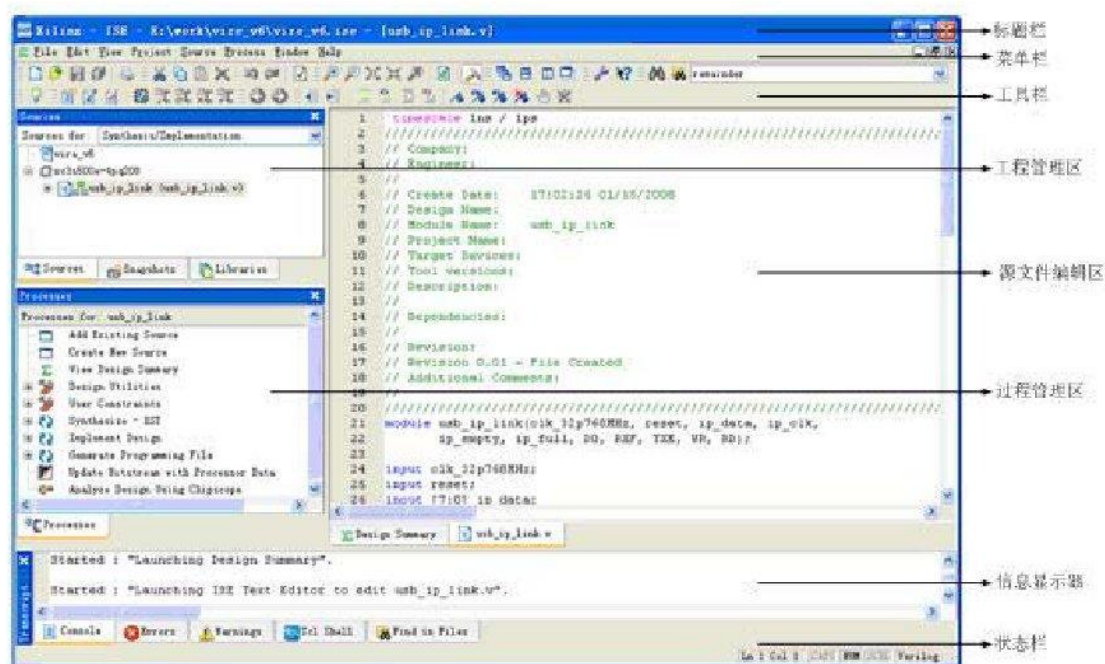


图 4-6 ISE 的主界面

- 标题栏：主要显示当前工程的名称和当前打开的文件名称。
- 菜单栏：主要包括文件（File）、编辑（Edit）、视图（View）、工程（Project）、源文件（Source）、操作（Process）、窗口（Window）和帮助（Help）等 8 个下拉菜单。其使用方法和常用的 Windows 软件类似。
- 工具栏：主要包含了常用命令的快捷按钮。灵活运用工具栏可以极大地方便用户在 ISE 中的操作。在工程管理中，此工具栏的运用极为频繁。
- 工程管理区：提供了工程以及相关文件的显示和管理功能，主要包括源文件视图（Source View），快照视图（Snapshot View）和库视图（Library View）。其中源文件视图比较常用，显示了源文件的层次关系。快照是当前工程的备份，设计人员可以随时备份，也可以将当前工程随时恢复到某个备份状态。快照视图用于查看当前工程的快照。执行快照功能的方法是选择菜单项 **Project | Take Snapshot**。库视图则显示了工程中用户产生的库的内容。
- 源文件编辑区：源文件编辑区提供了源代码的编辑功能。
- 过程管理区：本窗口显示的内容取决于工程管理区中所选定的文件。相关操作和 FPGA 设计流程紧密相关，包括设计输入、综合、仿真、实现和生成配置文件等。对某个文件进行了相应的处理后，在处理步骤的前面会出现一个图标来表示该步骤的状态。
- 信息显示区：显示 ISE 中的处理信息，如操作步骤信息、警告信息和错误信息等。信息显示区的下方有两个标签，分别对应控制台信息区（Console）和文件查找区（Find in

Files)。如果设计出现了警告和错误，双击信息显示区的警告和错误标志，就能自动切换到源代码出错的地方。

- 状态栏：显示相关命令和操作的信息。

2. ISE 菜单的基本操作

ISE 所有的操作都可通过菜单完成，下面简要介绍 ISE 的菜单命令以及功能。

(1) File 菜单

File 菜单的命令包括：New Project、Open Project、Open Examples、Close Project、Save Project As、New、Open、Save、Save As、Save All、Print Preview、Print、Recent Files、Recent Projects 以及 Exit 等。

New Project 命令：用于新建工程，是开始设计的第一步。ISE 会为新建的工程创建一个和工程同名的文件夹，专门用于存放工程的所有文件。

Open Project 命令：用于打开已有的 ISE 工程。高版本的 ISE 可以打开低版本的工程，但需要版本转换，该转换是单向的、不可逆的，因此需要做好版本备份。低版本的 ISE 不能打开高版本的 ISE 工程。

Open Examples 命令：用于打开 ISE 提供的各种类型的示例。

Close Project 命令：关闭当前工程。如果关闭前未保存文件，ISE 会提示用户保存后再退出。

Save Project As 命令：可将整个工程另存为其他名字的工程，在大型开发中，常使用该命令来完成版本备份。

New 命令：用于新建源文件，可生成原理图、符号以及文本文件。文本文件另存为时可修改其后缀名，以生成.v 或.vhd 的源文件。

Open 命令：用于打开所有 Xilinx 所支持的文件格式，便于用户查看各类文件资源。

Save、Save As 以及 Save All 命令：分别用于保存当前源文件、另存为当前源文件以及保存所有源文件。用户要在开发当中养成及时保存文件的习惯，避免代码丢失。

Print Preview 命令：用于打印预览当前文件，**Print** 用于打印当前文件。

Recent Files 命令：用于查看最近打开的文件。

Recent Projects 命令：用于查看最近打开的工程。

Exit 命令：用于退出 ISE 软件。

(2) Edit 菜单

Edit 菜单的命令包括：Undo、Redo、Cut、Copy、Paste、Delete、Find、Find Next、Find in Files、Language Templates、Select All、Unselect All、Message Filters、Object Properties 以及 Preference 等，大多数命令用于源代码开发中。

Undo 命令：用于撤销当前操作，返回到前一状态。

Redo 命令：是 Undo 命令的逆操作，用于恢复被撤销的操作。

Cut 命令：剪贴选中的代码，快捷键为“CTRL+X”。

Copy 命令：复制选中的代码，快捷键为“CTRL+C”。

Paste 命令：粘贴剪贴和复制的代码，快捷键为“CTRL+V”。

Delete 命令：删除选中的代码。

Find 命令：查找选中的文字，或寻找在其输入框中输入的内容，快捷键为“CTRL+F”。

Find Next 命令：寻找下一个要查找的内容，并跳至相应的位置，快捷键为“F3”。

Language Templates 命令：可打开语言模版，里面有丰富的学习资料，是非常完整的 HDL 语言帮助手册，其地位类似于 VisualC++ 的 MSDN。

Select All 命令：选中所有的代码，其快捷键为“CTRL+A”。

Unselect All 命令：撤销已选中的全部代码，是 Select All 的逆操作。

Message Filter 命令：过滤消息，只显示用户期望的消息。

Preference 命令：用于设定 ISE 的启动参数以及运行参数，有着众多的设置项，最常用的就是第三方 EDA 软件的关联设置，将在第 4.5 节详细介绍。

（3）View 菜单

View 菜单主要管理 ISE 软件的视图，不涉及 FPGA 开发中的任何环节，其中常用的命令有 Layout Horizontally、Layout Vertically 以及 Restore Default Layout。

Layout Horizontally 命令：将水平地排列 ISE 主界面中过程管理区、过程管理区以及代码编辑区等主要栏目。

Layout Vertically 命令：将垂直地排列 ISE 主界面中过程管理区、过程管理区以及代码编辑区等主要栏目。

Restore Default Layout 命令：将恢复 ISE 默认的主界面布局。

（4）Project 菜单

Project 菜单包含了对工程的各个操作，是设计中最常用的菜单之一，包括 New Source、Add Source、Add Copy of source、Cleanup Project Files、Toggle Paths、Archive、Take Snapshot、Make Snapshot Current、Apply Project Properties 以及 Source Control 命令。

New Source 命令：用于向工程中添加源代码，可以添加 HDL 源文件、IP Core 以及管脚和

时序约束文件。

Add Source 命令：将已有的各类源代码文件加入到工程中，Verilog 模块的后缀为.v，VHDL 模块的后缀为.vhd，IP core 源文件为.xco 文件或.xaw 文件，约束文件的后缀为.ucf。

Add Copy of source 命令，将目标文件拷贝一份添加到工程中。

Cleanup Project Files 命令：用于清空综合和实现过程所产生的文件和目录。如果在 EDIF 设计模式中，只清空实现过程所产生的文件。

Toggle Paths 命令：用于显示或隐藏非工程文件夹中的远端源文件的路径；

Archive 命令：用于压缩当前工程，包括所有的文件，默认压缩类型为.zip 格式。

Take Snapshot 命令：用于产生一个工程快照，即当前目录和远程资源的一个只读记录，常用于版本控制。

Make Snapshot Current 命令：用户恢复快照覆盖当前工程。由于该命令会将当前工程删除，所以使用前一定要做好数据备份工作。

Apply Project Properties 命令：应用工程属性，会提示用于选择相应工程。

Source Control 常用于代码的导入和导出，有 **Export** 和 **Import** 两个子命令。

(4) Source 菜单

Source 菜单主要面向工程管理区，包含了对资源文件的各个操作，每个命令的操作也都可以工程管理区单击右键弹出的对话框中点击实现，包括：**Open**、**Set as Top Module**、**Use SmartGuide**、**New Partition**、**Delete Partition**、**Partition properties**、**Partition Force**、**Remove**、**Move to library** 以及 **Properties** 等命令。

Open 命令：可打开所有类型的源文件，包括.v、.vhd、.xco、.xaw 以及.ucf 等格式。

Set as Top Module 命令：用于将选中的文件设置成顶层模块。只有设置成顶层模块，才能对其综合、实现以及生成相应的二进制比特流文件。

Use SmartGuide 命令：允许用户在本次实现时利用上一次实现的结果，包括时序约束以及布局布线结果，可节省实现的时间，但前提是工程改动不大。

New Partition 命令：新建分区，常用于区域约束。

Delete Partition 命令：删除区域约束的分区

Partition properties 命令：可设置分区属性，详细说明参考 4.4.4 节内容。

Partition Force 命令：包含“Force Synthesis Out-of-data”和“Force Implement Design Out-of-data”两个指令，分别用于分区综合和增量设计。

Remove 命令：把选中的文件从工程中删除，但仍保留在计算机硬盘上。

Move to library 命令：将选中的源文件移动到相应的库中，以便建立用户文件库。

Properties 命令：查看源文件属性，有 **Synthesis/Implementation Only**、**Simulation Only** 以及 **Synthesis/Imp+ Simulation** 三种类型，其中 **Simulation Only** 类文件只能仿真，不能被综合。

(5) Process 菜单

Process 菜单包含了工程管理区的所有操作，每个命令的操作也都可以过程管理区点击相应的图标实现，包括：**Inmolement Top Module**、**Run**、**Rerun**、**Rerun All**、**Stop**、**Open Without Updating** 以及 **Properties** 等命令。

Inmolement Top Module 命令：完成顶层模块的实现过程。

Run 命令：在工程过程栏，选中不同的操作，点击改命令，可分别启动综合、转换、映射、布局布线等过程。

Rerun 命令：重新运行 **Run** 指令执行的内容。

Rerun All 命令：重新运行所有 **Run** 指令执行的内容。

Stop 命令：停止当前操作，可中止当前操作，包括综合和实现的任一步骤。

Open Without Updating 命令：改指令用于打开相应上一次完成的综合或实现过程所产生的文件。

Properties 命令：在工程过程栏，选中不同的操作，点击该命令，可设置不同阶段的详细参数。

(6) Windows 菜单

Windows 菜单的主要功能是排列所有窗口，使其易看易管理。通过本菜单可以看到当前打开的所有窗口，并能直接切换到某个打开的窗口。由于各命令操作简单，不再介绍。

(7) Help 菜单

Help 菜单主要提供 **ISE** 所有帮助以及软件管理操作，包括：**Help Topics**、**Software Manuals**、**Xilinx on the Web**、**Tutorials**、**Update Software Product Configuration**、**Tip of the Day**、**WebUpdata** 以及 **About** 命令。

Help Topics 命令：点击后，将自动调用 **IE** 浏览器打开 **ISE** 的帮助文档。

Software Manuals 命令：点击后，将自动打开 **PDF** 文件，通过超链接到用户感兴趣的软件使用文档，其内容比网页形式的帮助文档要丰富。**Xilinx on the Web** 命令：包括完整的 **Xilinx** 网络资源，可根据需要点击查看链接。

Tutorials 命令：包括本地快速入门 **ISE** 的说明文档和 **Xilinx** 网站的入门教学内容，可点击查看。

Update Software Product Configuration 命令：用于更新 **ISE** 软件的注册 **ID**，如果试用版用户在试用期间购买了正版软件，不用卸载再重新安装，只需要通过该命令更换 **ID** 即可。

Tip of the Day 命令：每天提示，可设置或关闭在每次启动 **ISE** 时，弹出对话框，列出 **ISE** 的最新功能和一个应用技巧。

WebUpdata 命令：点击该命令，可自动连接到 **Xilinx** 的官方网站，下载最近的软件包并提

示用户安装。

About 命令： 点击该命令将弹出 ISE 的版本，包括主版本和升级号以及注册 ID。

第 2 节 HDL 代码输入

4.2.1 新建工程

首先打开 ISE，每次启动时 ISE 都会默认恢复到最近使用过的工程界面。当第一次使用时，由于此时还没有过去的工程记录，所以工程管理区显示空白。选择 **File | New Project** 选项，在弹出的新建工程对话框中的工程名称中输入“one2two”。在工程路径中单击 **Browse** 按钮，当工程放到指定目录，如图 4-7 所示。

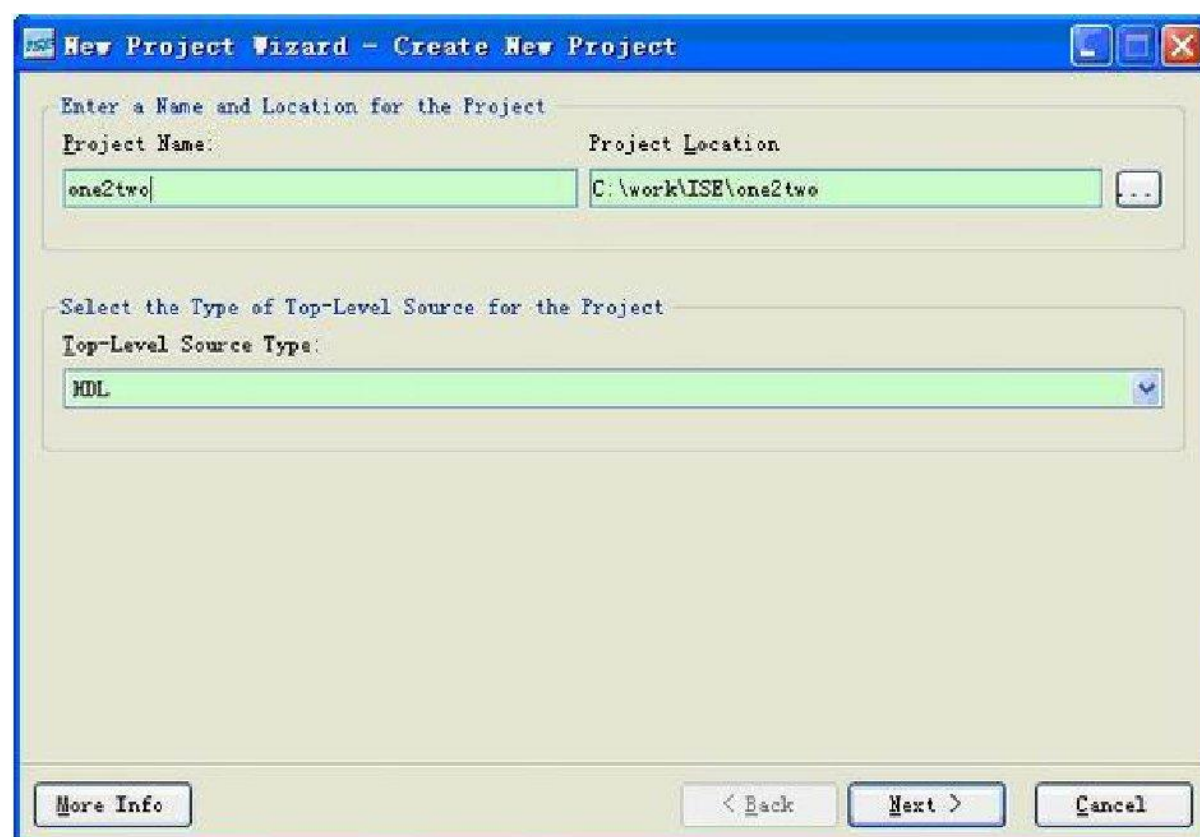


图 4-7 利用 ISE 新建工程的示意图

然后点击“Next”进入下一页，选择所使用的芯片类型以及综合、仿真工具。计算机上所安装的所有用于仿真和综合的第三方 EDA 工具都可以在下拉菜单中找到，如图 4-8 所示。在图中，我们选用了 Virtex4-10 芯片，并且指定综合工具为 Synplify (Verilog)，仿真工具选为 ModelSin-SE mixed。

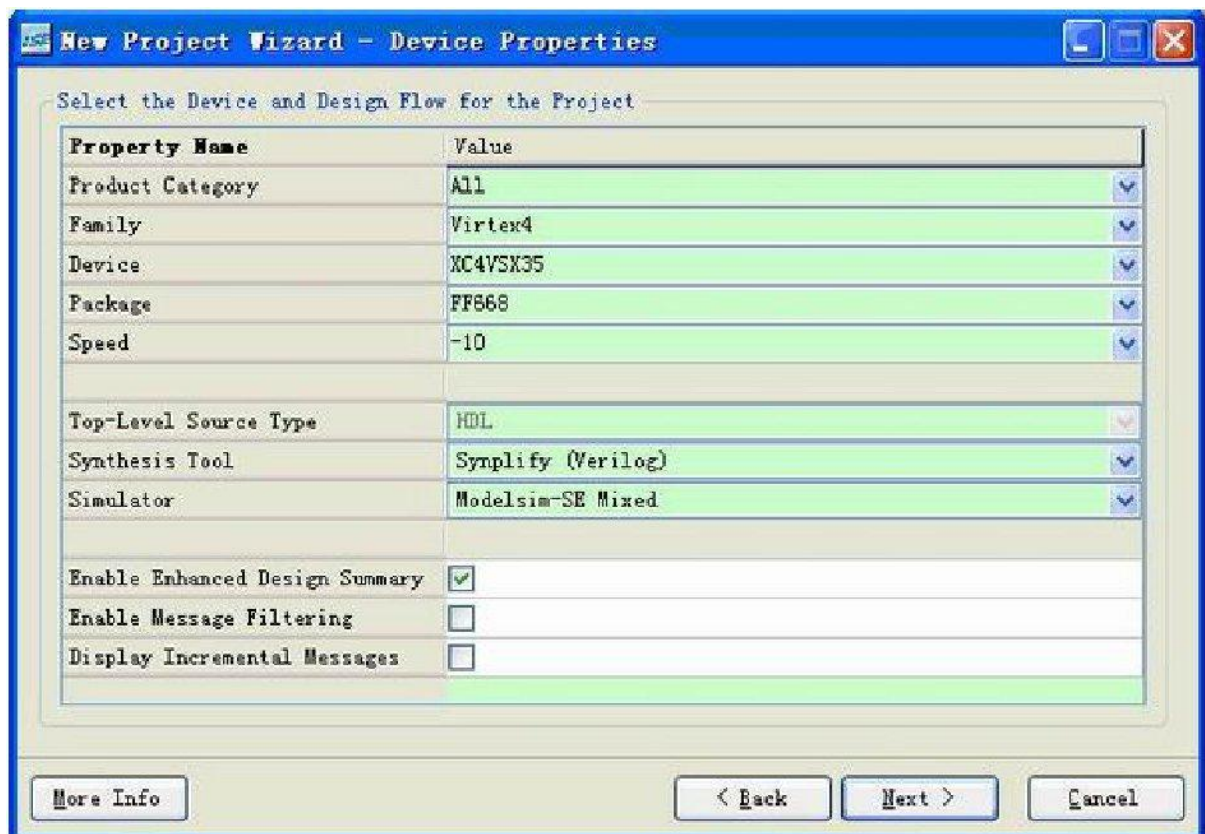


图 4-8 新建工程器件属性配置表

再点击“Next”进入下一页，可以选择新建源代码文件，也可以直接跳过，进入下一页。第 4 页用于添加已有的代码，如果没有源代码，点击“Next”，进入最后一页，单击确认后，就可以建立一个完整的工程。

4.2.2 代码输入

在工程管理区任意位置单击鼠标右键，在弹出的菜单中选择“New Source”命令，会弹出如图 4-9 所示的 New Source 对话框。

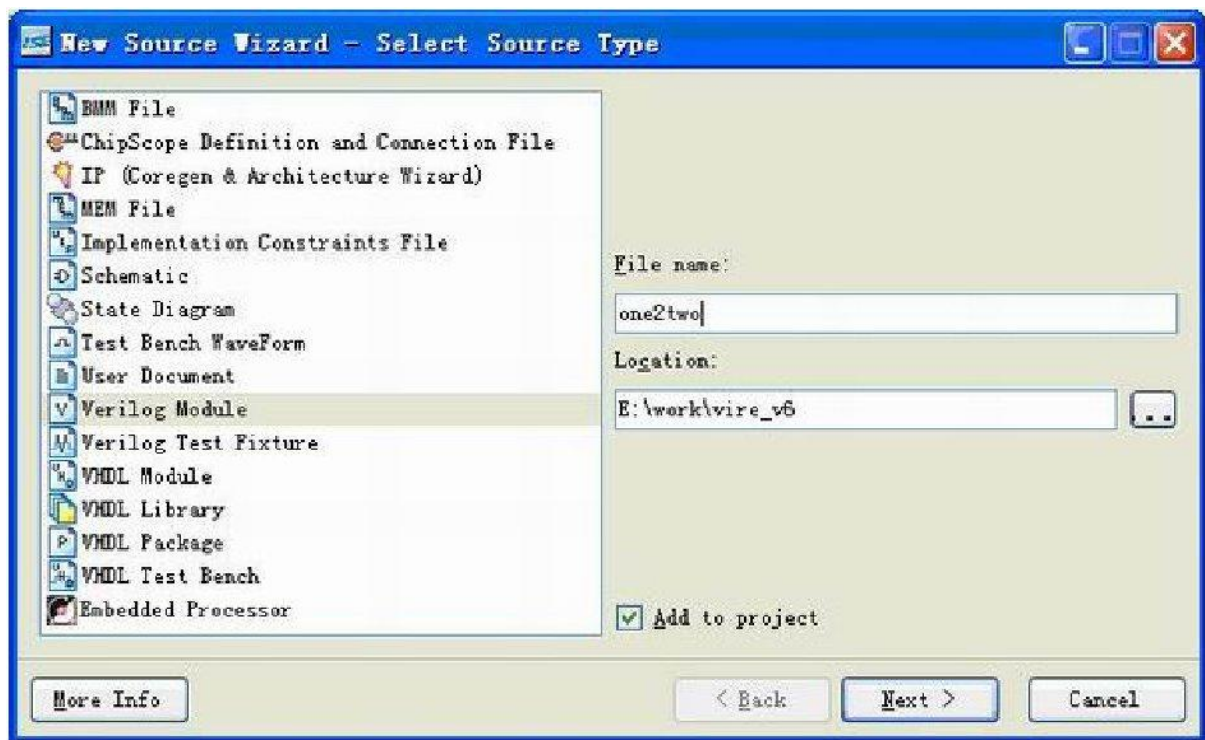


图 4-9 新建源代码对话框

左侧的列表用于选择代码的类型，各项的意义如下所示：

- BMM File:
- ChipScope Definition and Connection File：在线逻辑分析仪 ChipScope 文件类型，具有独特的优势和强大的功能，将在 M 张进行讨论。
- IP (Coregen & Architecture Wizard)：由 ISE 的 IP Core 生成工具快速生成可靠的源代码，这是目前最流行、最快速的一种设计方法，将在 4.5 节详细讨论。
- MEM File:
- Implementation Constraints File：约束文件类型。
- State Disgram：状态图类型。
- Test Bench Wavaform：测试波形类型。
- User Document：用户文档类型。
- Verilog Module：Verilog 模块类型。
- Verilog Test Fixture：Verilog 测试模块类型。
- VHDL Module：VHDL 模块类型。
- VHDL Library：VHDL 库类型。
- VHDL Packet：VHDL 包类型。

- VHDL Test Bench: VHDL 测试模块类型。

在代码类型中选择 Verilog Module 选项，在 File Name 文本框中输入 one2two，单击 Next 进入端口定义对话框，如图 4-10 所示。

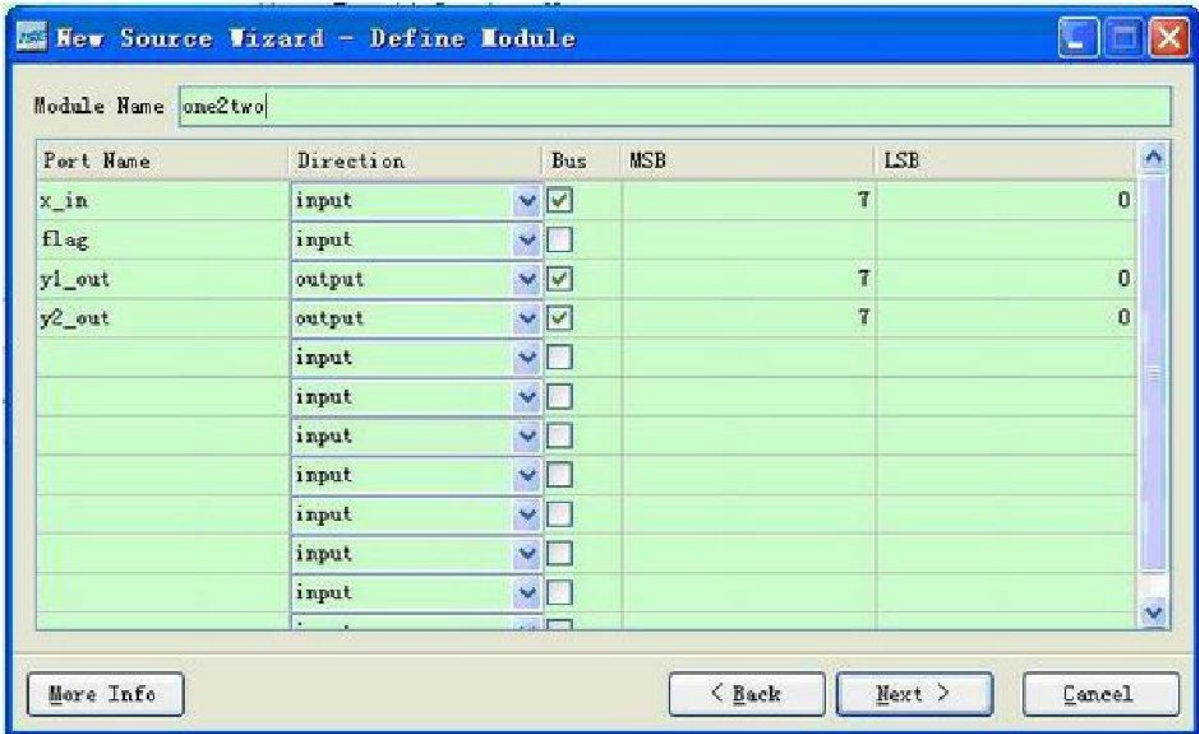


图 4-10 Verilog 模块端口定义对话框

其中 Module Name 就是输入的“one2two”，下面的列表框用于对端口的定义。“Port Name”表示端口名称，“Direction”表示端口方向（可以选择为 input、output 或 inout），MSB 表示信号的最高位，LSB 表示信号的最低位，对于单位信号的 MSB 和 LSB 不用填写。

定义了模块端口后，单击“Next”进入下一步，点击“Finish”按钮完成创建。这样，ISE 会自动创建一个 Verilog 模块的例子，并且在源代码编辑区内打开。简单的注释、模块和端口定义已经自动生成，所剩余的工作就是在模块中实现代码。填入的代码如下：

```
module one2two(x_in, flag, y1_out, y2_out);
    input [7:0] x_in;
    input flag;
    output [7:0] y1_out;
    output [7:0] y2_out;
```

```
// 以下为手工添加的代码  
assign y1_out = flag ? x_in : 8'b0000_0000;  
assign y2_out = flag ? 8'b0000_0000 : x_in;
```

endmodule

4.2.3 代码模板的使用

ISE 中内嵌的语言模块包括了大量的开发实例和所有 FPGA 语法的介绍和举例，包括 Verilog HDL/HDL 的常用模块、FPGA 原语使用实例、约束文件的语法规则以及各类指令和符号的说明。语言模板不仅可在设计中直接使用，还是 FPGA 开发最好的工具手册。在 ISE 工具栏中点击  图标，或选择菜单“Edit | Language Templates”，都可以打开语言模板，其界面如图 4-11 所示。

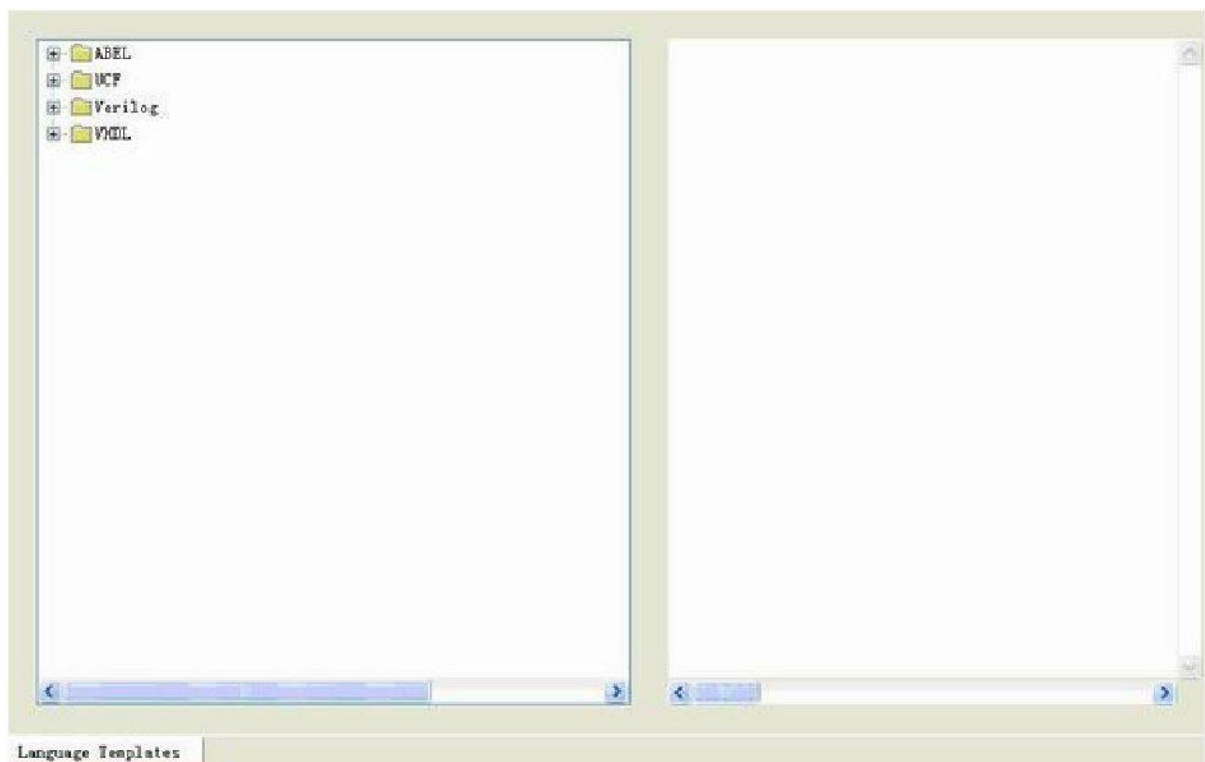


图 4-11 ISE 语言模版用户界面

界面左边有 4 项：ABEL、UCF、Verilog 以及 VHDL，分别对应着各自的参考资料。其中 ABEL 语言主要用于 GAL 和 ISP 等器件的编程，不用于 FPGA 开发。

以 Verilog 为例，点击其前面的“+”号，会出现 Common Constructs、Device Primitive Instantiation、Simulation Constructs、Synthesis Constructs 以及 User Templates 5 个子项。其中第 1 项主要介绍 Verilog 开发中所用的各种符号的说明，包括注释符以及运算符等。第 2 项主要介绍 Xilinx 原语的使用，可以最大限度地利用 FPGA 的硬件资源。第 3 项给出了程序仿真的所有指令和语句的说明和示例。第 4 项给出了实际开发中可综合的 Verilog 语句，并给出了大量可靠、实用的应用实例，FPGA 开发人员应熟练掌握该部分内容。User Templates 项是设计人员自己添加的，常用于在实际开发中统一代码风格。

下面以调用全局时钟缓冲器模版为例，给出语言模板的使用方法。在语言模板中，选择“Device Primitive Instantiation FPGA Clock Components Clock Buffers Global Clock Buffer (BUFG)”，即可看到调用全局时钟缓冲的示例代码，如图 4-12 所示。

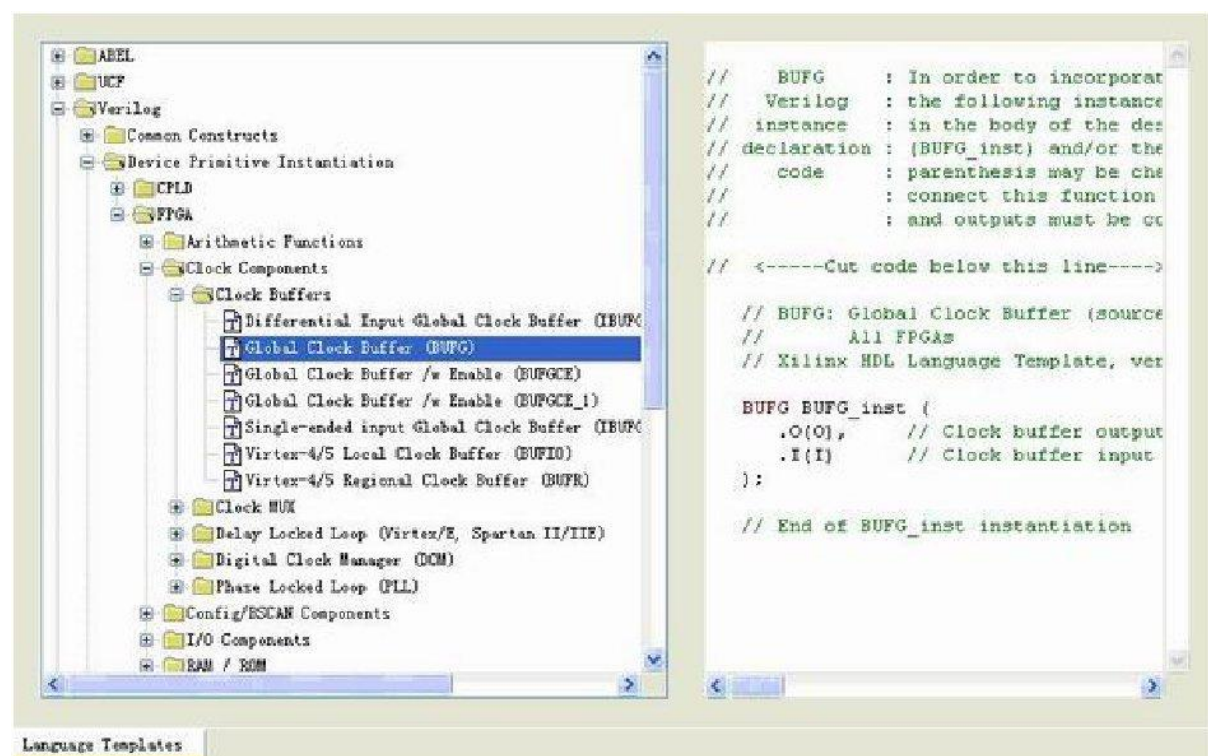


图 4-12 全局时钟缓冲器的语言模板

4.2.3 Xilinx IP Core 的使用

1. Xilinx IP core 基本操作

IP Core 就是预先设计好、经过严格测试和优化过的电路功能模块，如乘法器、FIR 滤波器、

PCI 接口等，并且一般采用参数可配置的结构，方便用户根据实际情况来调用这些模块。随着 FPGA 规模的增加，使用 IP core 完成设计成为发展趋势。

IP Core 生成器（Core Generator）是 Xilinx FPGA 设计中的一个重要设计工具，提供了大量成熟的、高效的 IP Core 为用户所用，涵盖了汽车工业、基本单元、通信和网络、数字信号处理、FPGA 特点和设计、数学函数、记忆和存储单元、标准总线接口等 8 大类，从简单的基本设计模块到复杂的处理器一应俱全。配合 Xilinx 网站的 IP 中心使用，能够大幅度减轻设计人员的工作量，提高设计可靠性。

Core Generator 最重要的配置文件的后缀是.xco，既可以是输出文件又可以是输入文件，包含了当前工程的属性和 IP Core 的参数信息。

启动 Core Generator 有两种方法，一种是在 ISE 中新建 IP 类型的源文件，另一种是双击运行[开始] [程序] [Xilinx ISE 9.1i] [Accessories] [Core Generator]。限于篇幅，本节只以调用加法器 IP Core 为例来介绍第一种方法。

在工程管理区单击鼠标右键，在弹出的菜单中选择 New Source，选中 IP 类型，在 File Name 文本框中输入 adder（注意：该名字不能出现英文的大写字母），然后点击 Next 按键，进入 IP Core 目录分类页面，如图 4-13 所示。



图 4-13 IP Core 目录分类页面

下面以加法器模块为例介绍详细操作。首先选中“Math Function Adder & Subtractor Adder Subtractor v7.0”，点击“Next”进入下一页，选择“Finish”完成配置。这时在信息显示区会出现“Customizing IP...”的提示信息，并弹出一个“Adder Subtractor”配置对话框，如图 4-14 所示。

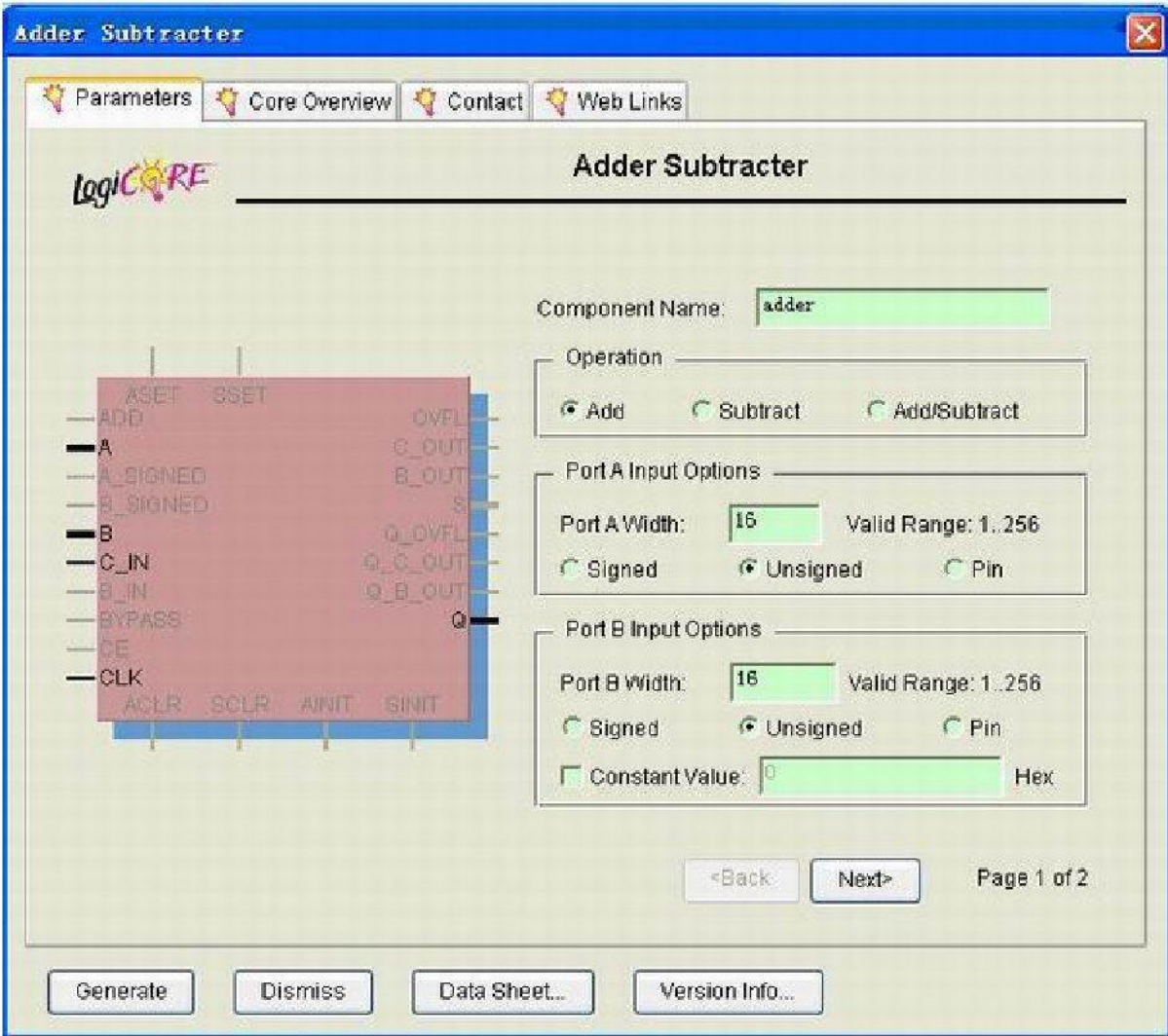


图 4-14 加法器 IP Core 配置对话框

然后，选中 `adder`，设置位宽为 16，然后点击“Generate”，信息显示区显示 `Generating IP...`，直到出现 `Successfully generated adder` 的提示信息。此时在工程管理区出现一个“adder.xco”的文件。这样加法器的 IP Core 已经生成并成功调用。

IP Core 在综合时被认为是黑盒子，综合器不对 IP Core 做任何编译。IP Core 的仿真主要是

运用 Core Generator 的仿真模型来完成的，会自动生成扩展名为.v 的源代码文件。设计人员只需要从该源文件中查看其端口声明，将其作为一个普通的子程序进行调用即可。下面给出加法器的应用实例。

例 4-1 调用加法器的 IP core，并用其实现图 4-15 所示的 2 级加法树。

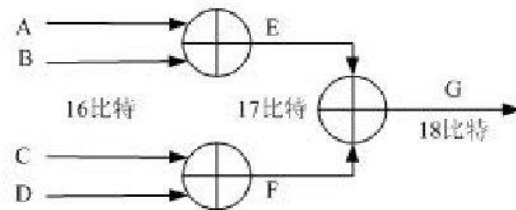


图 4-15 2 级加法器示意图

按照本节介绍的步骤生成 2 个加法器的 IP core Add16 和 Add17，前者用于实现第 1 级的加法，后者用于实现第 2 级加法，对应的代码为：

```

module addertree(clk, a1, a2, b1, b2, c);
input clk;
input [15:0] a1;
input [15:0] a2;
input [15:0] b1;
input [15:0] b2;
output [17:0] c;

```

```

wire [16:0] ab1, ab2;

```

```

adder16 adder16_1(
.A(a1),
.B(a2),
.Q(ab1),
.CLK(clk)
);

```

```

adder16 adder16_2(
.A(b1),
.B(b2),

```

```
.Q(ab2),
.CLK(clk)
);

adder17 adder17(
.A(ab1),
.B(ab2),
.Q(c),
.CLK(clk)
);

endmodule
```

上述程序经过综合后，得到如图 4-16 所示的 RTL 级结构图。

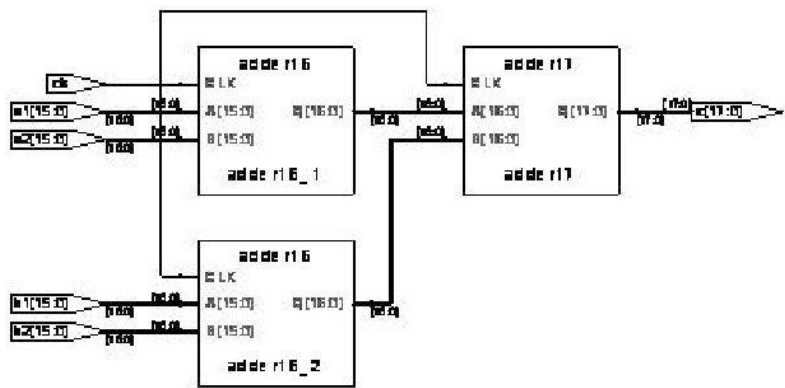
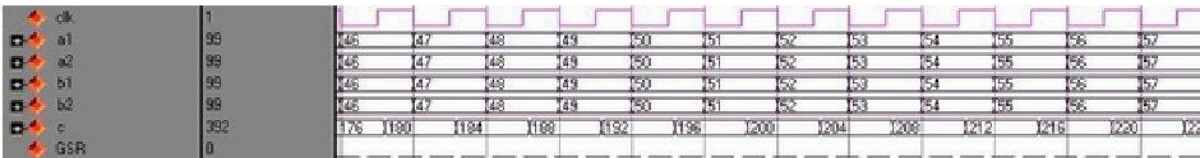


图 4-16 2 级加法树的 RTL 结构图

经过 ModelSim 6.2b 仿真测试，得到的功能波形图如图 4-17 所示。由于每一级加法器会引入一个时钟周期的延迟，因此，两级加法器就会引入 2 个时钟的周期，可以看出，仿真结果和设计分析的结果是一样的。



2. DDS 模块 IP Core 的调用实例

• DDS 算法原理

DDS 技术是一种新的频率合成方法，是频率合成技术的一次革命，最早由 JOSEPH TIERNEY 等 3 人于 1971 年提出，但由于受当时微电子技术和数字信号处理技术的限制，DDS 技术没有受到足够重视，随着数字集成电路和微电子技术的发展，DDS 技术日益显露出它的优越性。

DDS 的工作原理为：在参考时钟的驱动下，相位累加器对频率控制字进行线性累加，得到的相位码对波形存储器寻址，使之输出相应的幅度码，经过模数转换器得到相应的阶梯波，最后再使用低通滤波器对其进行平滑，得到所需频率的平滑连续的波形，其结构如图 4-18 所示。

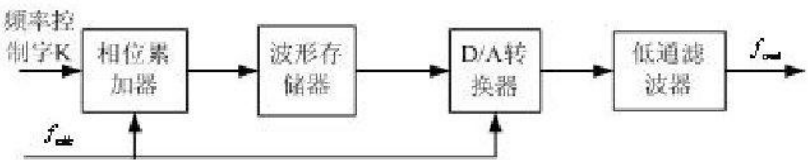


图 4-18 DDS 的结构框图

相位累加器由 N 位加法器与 N 位累加寄存器级联构成，结构如图 4-19 所示。每来一个时钟脉冲 f_{clk} ，加法器将频率控制字 K 与累加寄存器输出的累加相位数据相加，把相加后的结果送至累加寄存器的数据输入端。累加寄存器将加法器在上一个时钟脉冲作用后所产生的新相位数据反馈到加法器的输入端，以使加法器在下一个时钟脉冲的作用下继续与频率控制字相加。这样，相位累加器在时钟作用下，不断对频率控制字进行线性相位累加。由此可以看出，相位累加器在每一个时钟脉冲输入时，把频率控制字累加一次，相位累加器输出的数据就是合成信号的相位，相位累加器的溢出频率就是 DDS 输出的信号频率。用相位累加器输出的数据作为波形存储器（ROM）的相位取样地址，这样就可把存储在波形存储器内的波形抽样值（二进制编码）经查找表查出，完成相位到幅值转换。

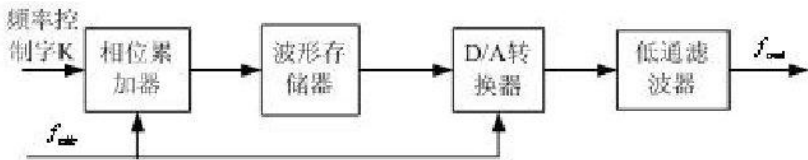


图 4-19 DDS 相位累加器

波形存储器所储存的幅度值与余弦信号有关。余弦信号波形在一个周期内相位幅度的变化关系可以用图 4-20 中的相位圆表示，每一个点对应一个特定的幅度值。一个 N 位的相位累加器对应着圆上 N 个相位点，其相位分辨率为 $\frac{2\pi}{N}$ 。若 N=16，则共有 16 种相位值与 16 种幅度值相对应，并将相应的幅度值存储于波形存储器中，存储器的字节数决定了相位量化误差。在实际的 DDS 中，可利用正弦波的对称性，可以将 N 范围内的幅、相点减小到 N/4 以降低所需的存储量，量化的比特数决定了幅度量化误差。

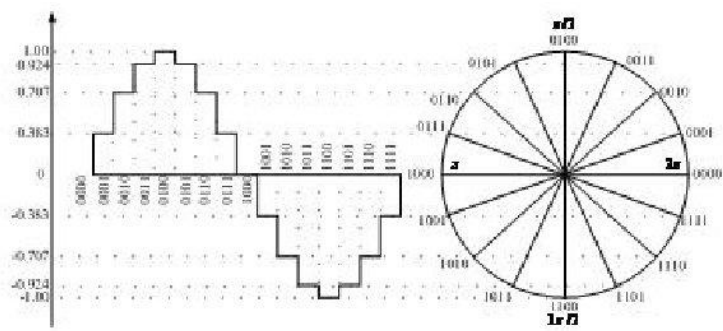


图 4-20 三角函数相位与幅度的对应关系

波形存储器的输出送到 D/A 转换器，D/A 转换器将数字量形式的波形幅值转换成所要求合成频率的模拟量形式信号。低通滤波器用于滤除不需要的取样分量，以便输出频谱纯净的正弦波信号。DDS 在相对带宽、频率转换时间、高分辨力、相位连续性、正交输出以及集成化等一系列性能指标方面远远超过了传统频率合成技术所能达到的水平，为系统提供的信号源优于模拟信号源。

DDS 模块的输出频率 f_{out} 是系统工作频率 f_{clk} 、相位累加器比特数 N 以及频率控制字 K 三者的一个函数，其数学关系由式(4.1)给出。

$$f_{out} = \frac{f_{clk} K}{2^N} \tag{4.1}$$

它的频率分辨率，即频率的变化间隔为：

$$\Delta f = \frac{f_{clk}}{2^N} \tag{4.2}$$

• DDS IP Core 的调用

DDS 模块 IP Core 的用户界面如图 4-21 所示。该 IP Core 支持余弦、正弦以及正交函数的输出，旁瓣抑制比的范围从 18dB 到 115dB，最小频率分辨率为 0.02Hz，可同时独立支持 16 个通道。其中的查找表既可以利用分布式 RAM，也可利用块 RAM。

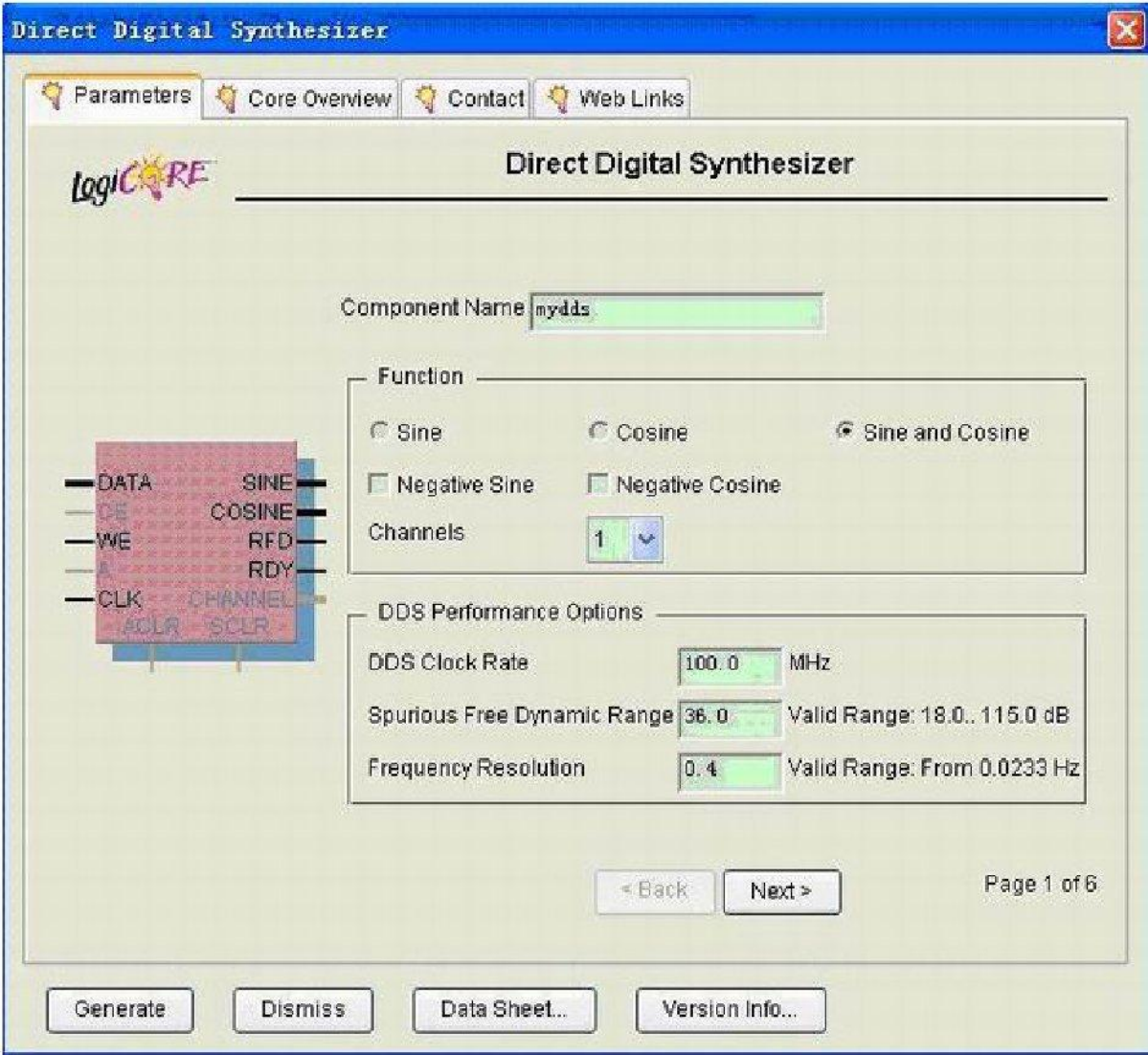


图 4-21 DDS IP Core 的用户界面

DDS 模块的信号端口说明如下：

- (1) CLK：输入信号，DDS 模块的工作时钟，对 DDS 输出信号的频率和频率分辨率有很大的影响。即式(5.40)中的 f_{clk} 。
- (2) A：输入信号，由于 DDS 模块的相位增量存储器和相位偏置寄存器共用一个数据通道，A 端口信号用于片选相位增量寄存器和偏置寄存器。当 A 端口的最高位为 1 时，相位偏置寄

寄存器被选中；当其为 0 时，则选中相位增量寄存器。其余的低 4 位比特用于片选 DDS 的输出通道，最多可以输出 16 路信号。

(3) WE: 输入信号，写有效控制信号，高有效。只有当 WE 为高时，DATA 端口的数值才能被写入相应的寄存器中。

(4) DATA: 输入信号，时分复用的数据总线，用于配置相位增量寄存器和相位偏置寄存器。

(5) ACLR: 输入信号，异步的清空信号，高有效。当 ACLR 等于 1 时，DDS 模块内部所有的寄存器都被清空，RDY 信号也会被拉低。

(6) SCLR: 输入信号，同步的清空信号，高有效。当 SCLR 等于 1 时，DDS 模块内部所有的寄存器都被清空，RDY 信号也会被拉低。

(7) RDY: 输出信号，输出握手信号。当其为高时，标志输出信号已经准备好。

(8) CHANNEL: 输出信号，输出通路的下标。用于表明当前时刻输出端为哪一路输出，其位宽由通道数决定。

(9) SINE: 输出信号，用于输出正弦的时间序列。

(10) COSINE: 输出信号，用于输出余弦的时间序列。

例 4-2 使用 DDS IP Core 实例化一个 4MHz，分辨率为 0.1Hz，带外抑制比为 60dB 的正、余弦信号发生器，假设工作时钟为 100MHz。

IP Core 直接生成 DDS 的 Verilog 模块接口为：

```
module mydds(  
    DATA,  
    WE,  
    A,  
    CLK,  
    SINE,  
    COSINE  
); // synthesis black_box  
  
input [27 : 0] DATA;  
input WE;  
input [4 : 0] A;  
input CLK;  
output [9 : 0] SINE;  
output [9 : 0] COSINE;  
.....  
endmodule
```

在使用时，直接调用 **mydds** 模块即可，如

```
module dds1(DATA, WE, A, CLK, SINE, COSINE);  
    input [27 : 0] DATA; // 经过计算，DATA= 10737418.  
    input WE;  
    input [4 : 0] A;  
    input CLK;  
    output [9 : 0] SINE;  
    output [9 : 0] COSINE;  
  
    mydds mydds1(  
        .DATA(DATA),  
        .WE(WE),  
        .A(A),  
        .CLK(CLK),  
        .SINE(SINE),  
        .COSINE(COSINE)  
    );  
  
endmodule
```

上述程序经过综合后，得到如图 4-22 所示的 RTL 级结构图。

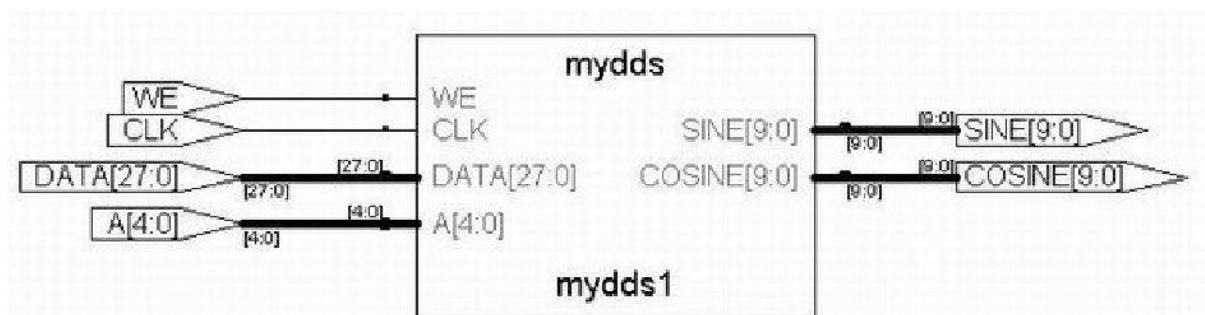


图 4-22 DDS 模块的 RTL 结构图

经过 ModelSim 仿真测试，得到的功能波形图如图 4-23 所示：

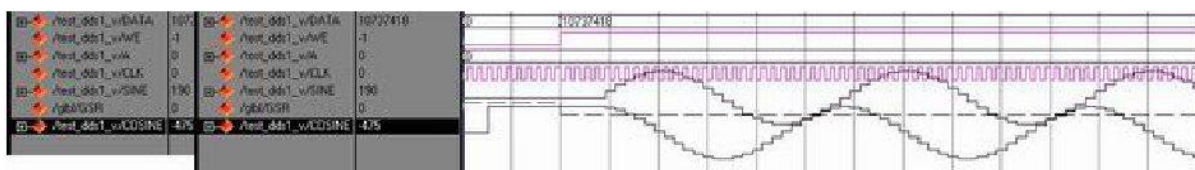


图 4-23 DDS 模块的局部功能仿真波形图

注意：经过笔者大量验证发现，在 ISE 8.2 版中使用 DDS IP core 时，只有采用 wire 型的变量控制 DDS 才能在 ModelSim 中得到正确结果，使用 reg 型的变量不能成功操控该 IP core。

第 3 节 基于 ISE 的开发流程

例 4-3 ISE 开发流程演示代码，将输入的数据加 1 寄存并输出。

```
module test(clk, din, dout);
    input clk;
    input [7:0] din;
    output [7:0] dout;

    reg [7:0] dout;

    always @(posedge clk) begin
        dout <= din + 1;
    end

endmodule
```

4.3.1 基于 Xilinx XST 的综合

所谓综合，就是将 HDL 语言、原理图等设计输入翻译成由与、或、非门和 RAM、触发器等基本逻辑单元的逻辑连接（网表），并根据目标和要求（约束条件）优化所生成的逻辑连接，生成 EDF 文件。XST 内嵌在 ISE 3 以后的版本中，并且在不断完善。此外，由于 XST 是 Xilinx 公司自己的综合工具，对于部分 Xilinx 芯片独有的结构具有更好的融合性。

完成了输入、仿真以及管脚分配后就可以进行综合和实现了。在过程管理区双击 Synthesize-XST，如图 4-24 所示，就可以完成综合，并且能够给出初步的资源消耗情况。

图 4-25 给出了模块所占用的资源。

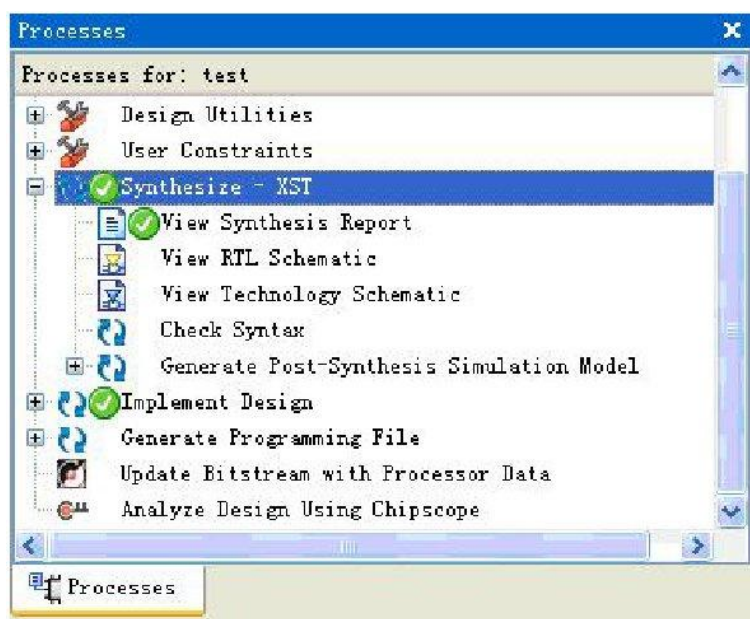


图 4-24 设计综合窗口

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of 4 input LUTs	8	30,720	1%	
Logic Distribution				
Number of occupied Slices	5	15,360	1%	
Number of Slices containing only related logic	5	5	100%	
Number of Slices containing unrelated logic	0	5	0%	
Total Number of 4 input LUTs	9	30,720	1%	
Number used as logic	8			
Number used as a route-thru	1			
Number of bonded IOBs	17	448	3%	
Number of BUFG/BUFGCTRLs	1	32	3%	
Number used as BUFGs	1			
Number used as BUFGCTRLs	0			
Total equivalent gate count for design	115			
Additional JTAG gate count for IOBs	816			

图 4-25 综合结果报告

综合可能有 3 种结果：如果综合后完全正确，则在 Synthesize-XST 前面有一个打钩的绿色小圈圈；如果有警告，则出现一个带感叹号的黄色小圆圈，如本例所示；如果有错误，则出现一个带叉的红色小圆圈。综合完成之后，可以通过双击 View RTL Schematics 来查看

RTL 级结构图，察看综合结构是否按照设计意图来实现电路。ISE 会自动调用原理图编辑器 ECS 来浏览 RTL 结构，所得到的 RTL 结构图如图 4-26 所示，综合结果符合设计者的意图，调用了加法器和寄存器来完成逻辑。

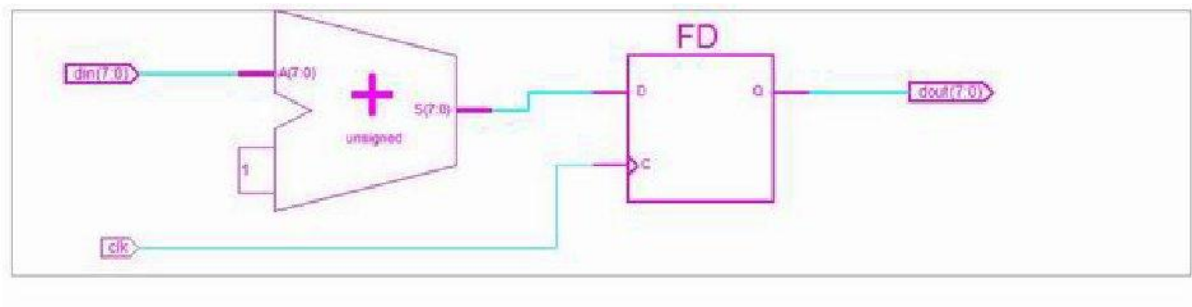


图 4-26 经过综合后的 RTL 级结构图

一般在使用 XST 时，所有的属性都采用默认值。其实 XST 对不同的逻辑设计可提供丰富、灵活的属性配置。下面对 ISE9.1 中内嵌的 XST 属性进行说明。打开 ISE 中的设计工程，在过程管理区选中“Synthesis -XST”并单击右键，弹出界面如图 4-27 所示。

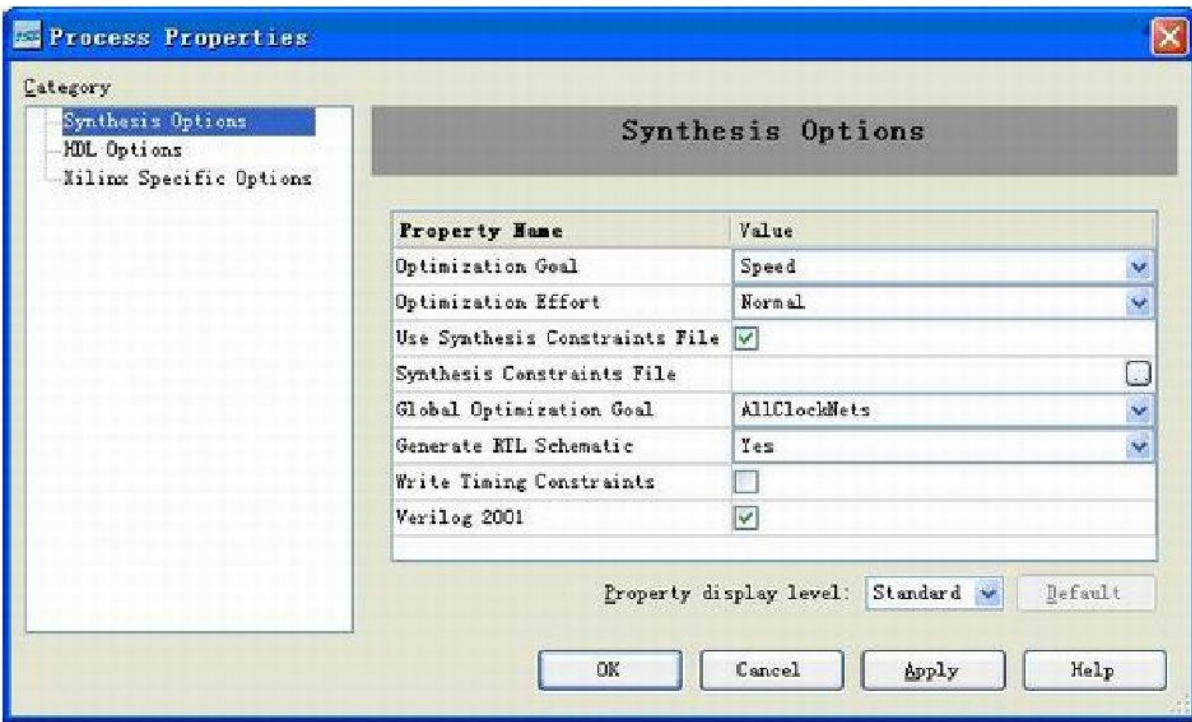


图 4-27 综合选项

由图 4-27 可以看出，XST 配置页面分为综合选项（Synthesis Options）、HDL 语言

选项（HDL Options）以及 Xilinx 特殊选项（Xilinx Specific Options）等三大类，分别用于设置综合的全局目标和整体策略、HDL 硬件语法规则以及 Xilinx 特有的结构属性。

- 综合选项参数

综合参数配置界面如图 4-27 所示，包括 8 个选项，具体如下所列：

【Optimization Goal】：优化的目标。该参数决定了综合工具对设计进行优化时，是以面积还是以速度作为优先原则。面积优先原则可以节省器件内部的逻辑资源，即尽可能地采用串行逻辑结构，但这是以牺牲速度为代价的。而速度优先原则保证了器件的整体工作速度，即尽可能地采用并行逻辑结构，但这样将会浪费器件内部大量的逻辑资源，因此，它是以牺牲逻辑资源为代价的。

【Optimization Effort】：优化器努力程度。这里有**【normal】**和**【high】**两种选择方式。对于**【normal】**，优化器对逻辑设计仅仅进行普通的优化处理，其结果可能并不是最好的，但是综合和优化流程执行地较快。如果选择**【high】**，优化器对逻辑设计进行反复的优化处理和分析，并能生成最理想的综合和优化结果，在对高性能和最终的设计通常采用这种模式；当然在综合和优化时，需要的时间较长。

【Use Synthesis Constraints File】：使用综合约束文件。如果选择了该选项，那么综合约束文件 XCF 有效。

【Synthesis Constraints File】：综合约束文件。该选项用于指定 XST 综合约束文件 XCF 的路径。

【Global Optimization Goal】：全局优化目标。可以选择的属性包括有**【AllClockNets】**、**【Inpad To Outpad】**、**【Offest In Before】**、**【Offest Out After】**、**【Maximm Delay】**。该参数仅对 FPGA 器件有效，可用于选择所设定的寄存器之间、输入引脚到寄存器之间、寄存器到输出引脚之间，或者是输入引脚到输出引脚之间逻辑的优化策略。

【Generate RTL Schematic】：生成寄存器传输级视图文件。该参数用于将综合结果生成 RTL 视图。

【Write Timing Constraints】：写时序约束。该参数仅对 FPGA 有效，用来设置是否将 HDL 源代码中用于控制综合的时序约束传给 NGC 网表文件，该文件用于布局和布线。

【Verilog 2001】：选择是否支持 Verilog 2001 版本。

- HDL 语言选项

HDL 语言选项的配置界面如图 4-28 所示，包括 16 个选项，具体如下所列：

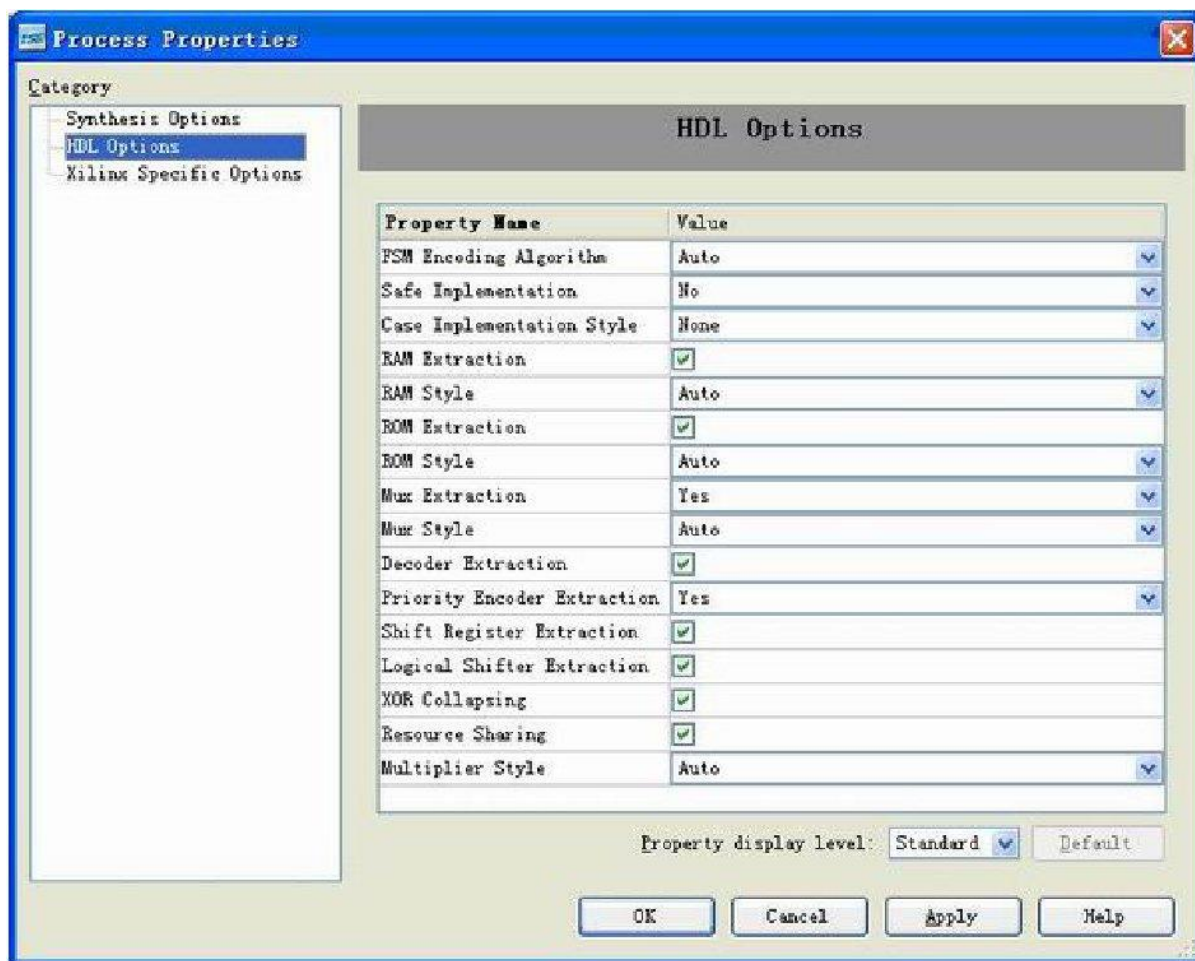


图 4-28 HDL 语言选项的配置界面选项

【FSM Encoding Algorithm】：有限状态机编码算法。该参数用于指定有限状态机的编码方式。选项有【Auto】 【One-Hot】 【Compact】 【Sequential】 【Gray】 【Johnson】、【User】、【Speed1】、【None】编码方式，默认为【Auto】编码方式。

【Safe Implementation】：将添加安全模式约束来实现有限状态机，将添加额外的逻辑将状态机从无效状态调到有效状态，否则只能复位来实现，有【Yes】、【No】两种选择，默认为【No】。

【Case Implementation Style】：条件语句实现类型。该参数用于控制 XST 综合工具解释和推论 Verilog 的条件语句。其中选项有【None】 【Full】 【Parallel】 【Full-Parallel】，默认为【None】。对于这四种选项，区别如下：（1）【None】，XST 将保留程序中条件语句的原型，不进行任何处理；（2）【Full】，XST 认为条件语句是完整的，避免锁存器的产生；（3）【Parallel】，XST 认为在条件语句中不能产生分支，并且不使用优先级编码器；（4）【Full-Parallel】，XST 认为条件语句是完整的，并且在内部没有分支，不使用锁存器和优先级编码器。

【RAM Extraction】：存储器扩展。该参数仅对 FPGA 有效，用于使能和禁止 RAM 宏接口。默认为允许使用 RAM 宏接口。

【RAM Style】：RAM 实现类型。该参数仅对 FPGA 有效，用于选择是采用块 RAM 还是分布式 RAM 来作为 RAM 的实现类型。默认为【Auto】。

【ROM Extraction】：只读存储器扩展。该参数仅对 FPGA 有效，用于使能和禁止只读存储器 ROM 宏接口。默认为允许使用 ROM 宏接口。

【ROM Style】：ROM 实现类型。该参数仅对 FPGA 有效，用于选择是采用块 RAM 还是分布式 RAM 来作为 ROM 的实现和推论类型。默认为【Auto】。

【Mux Extraction】：多路复用器扩展。该参数用于使能和禁止多路复用器的宏接口。根据某些内定的算法，对于每个已识别的多路复用/选择器，XST 能够创建一个宏，并进行逻辑的优化。可以选择【Yes】、【No】和【Force】中的任何一种，默认为【Yes】。

【Mux Style】：多路复用实现类型。该参数用于为宏生成器选择实现和推论多路复用/选择器的宏类型。可以选择【Auto】、【MUXF】和【MUXCY】中的任何一种，默认为【Auto】。

【Decoder Extraction】：译码器扩展。该参数用于使能和禁止译码器宏接口，默认为允许使用该接口。

【Priority Encoder Extraction】：优先级译码器扩展。该参数用于指定是否使用带有优先级的编码器宏单元。

【Shift Register Extraction】：移位寄存器扩展。该参数仅对 FPGA 有效，用于指定是否使用移位寄存器宏单元。默认为使能。

【Logical Shifter Extraction】：逻辑移位寄存器扩展。该参数仅对 FPGA 有效，用于指定是否使用逻辑移位寄存器宏单元。默认为使能。

【XOR Collapsing】：异或逻辑合并方式。该参数仅对 FPGA 有效，用于指定是否将级联的异或逻辑单元合并成一个大的异或宏逻辑结构。默认为使能。

【Resource Sharing】：资源共享。该参数用于指定在 XST 综合时，是否允许复用一些运算处理模块，如加法器、减法器、加/减法和乘法器。默认为使能。如果综合工具的选择是以速度为优先原则的，那么就不考虑资源共享。

【Multiplier Style】：乘法器实现类型。该参数仅对 FPGA 有效，用于指定宏生成器使用乘法器宏单元的方式。选项有【Auto】、【Block】、【LUT】和【Pipe_LUT】。默认为【Auto】。选择的乘法器实现类型和所选择的器件有关。

• Xilinx 特殊选项

Xilinx 特殊选项用于将用户逻辑适配到 Xilinx 芯片的特殊结构中，不仅能节省资源，还能提高设计的工作频率，其配置界面如图 4-29 所示，包括 10 个配置选项，具体如下所列。



图 4-29 Xilinx 指定的选项

【Add I/O Buffers】：插入 I/O 缓冲器。该参数用于控制对所综合的模块是否自动插入 I/O 缓冲器。默认为自动插入。

【Max Fanout】：最大扇出数。该参数用于指定信号和网线的最大扇出数。这里扇出数的选择与设计的性能有直接的关系，需要用户合理选择。

【Register Duplication】：寄存器复制。该参数用于控制是否允许寄存器的复制。对于高扇出和时序不能满足要求的寄存器进行复制，可以减少缓冲器输出的数目以及逻辑级数，改变时序的某些特性，提高设计的工作频率。默认为允许寄存器复制。

【Equivalent Register Removal】：等效寄存器删除。该参数用于指定是否把寄存器传输级功能等效的寄存器删除，这样可以减少寄存器资源的使用。如果某个寄存器是用 Xilinx 的硬件原语指定的，那么就不会被删除。默认为使能。

【Register Balancing】：寄存器配平。该参数仅对 FPGA 有效，用于指定是否允许平衡寄存器。可选项有【No】、【Yes】、【Forward】和【Backward】。采用寄存器配平技术，可以改善某些设计的时序条件。其中，【Forward】为前移寄存器配平，【Backward】为后移寄存器配平。采用寄存器配平后，所用到的寄存器数就会相应地增减。默认为寄存器不配平。

【Move First Flip-Flop Stage】：移动前级寄存器。该参数仅对 FPGA 有效，用于控制在进行寄存器配平时，是否允许移动前级寄存器。如果【Register Balancing】的设置为【No】，那么该参数的设置无效。

【Move Last Flip-Flop Stage】：移动后级寄存器。该参数仅对 FPGA 有效，用于控制在进行寄存器配平时，是否允许移动后级寄存器。如果【Register Balancing】的设置为【No】，那么该参数的设置无效。

【Pack I/O Registers into IOBs】：I/O 寄存器置于输入输出块。该参数仅对 FPGA 有效，用于控制是否将逻辑设计中的寄存器用 IOB 内部寄存器实现。在 Xilinx 系列 FPGA 的 IOB 中分别有输入和输出寄存器。如果将设计中的第一级寄存器或最后一级寄存器用 IOB 内部寄存器实现，那么就可以缩短 IO 引脚到寄存器之间的路径，这通常可以缩短大约 1~2ns 的传输时延。默认为【Auto】。

【Slice Packing】：优化 Slice 结构。该参数仅对 FPGA 有效，用于控制是否将关键路径的查找表逻辑尽量配置在同一个 Slice 或者 CLB 模块中，由此来缩短 LUT 之间的布线。这一功能对于提高设计的工作频率、改善时序特性是非常有用的。默认为允许优化 Slice 结构。

【Optimize Instantiated Primitives】：优化已例化的原语。该参数控制是否需要优化在 HDL 代码中已例化的原语。默认为不优化。

4.3.2 基于 ISE 的仿真

在代码编写完毕后，需要借助于测试平台来验证所设计的模块是否满足要求。ISE 提供了两种测试平台的建立方法，一种是使用 HDL Benchner 的图形化波形编辑功能编写，另一种就是利用 HDL 语言。由于后者使用简单、功能强大，所以本节主要介绍基于 Verilog 语言的测试平台建立方法。

1. 测试波形法

在 ISE 中创建 testbench 波形，可通过 HDL Benchner 修改，再将其和仿真器连接起来，再验证设计功能是否正确。首先在工程管理区将 Sources for 设置为 Behavioral Simulation，然后在任意位置单击鼠标右键，在弹出的菜单中选择“New Source”命令，然后选中“Test Bench WaveForm”类型，输入文件名为“test_bench”，点击 Next 进入下一页。这时，工程中所有 Verilog Module 的名称都会显示出来，设计人员需要选择要进行测试的模块。由于本工程只有一个模块，所以只列出了 test，如图 4-30 所示。

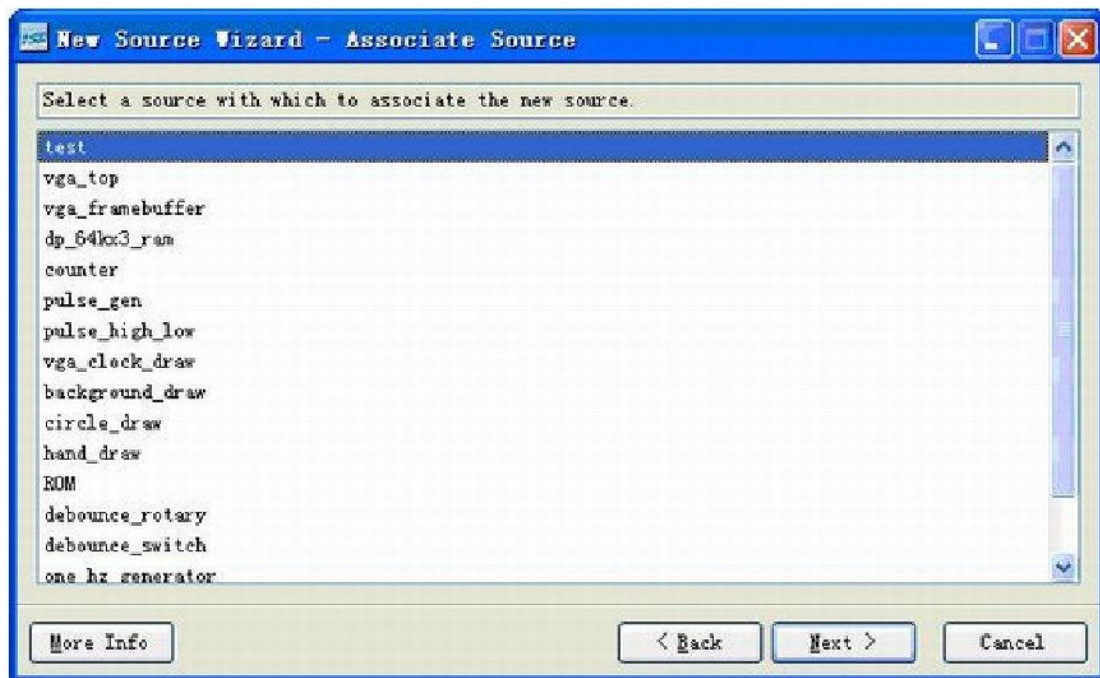


图 4-30 选择待测模块对话框

用鼠标选中 `test`，点击“Next”后进入下一页，直接点击“Finish”按键。此时 HDL Bench 程序自动启动，等待用户输入所需的时序要求，如图 4-31 所示。

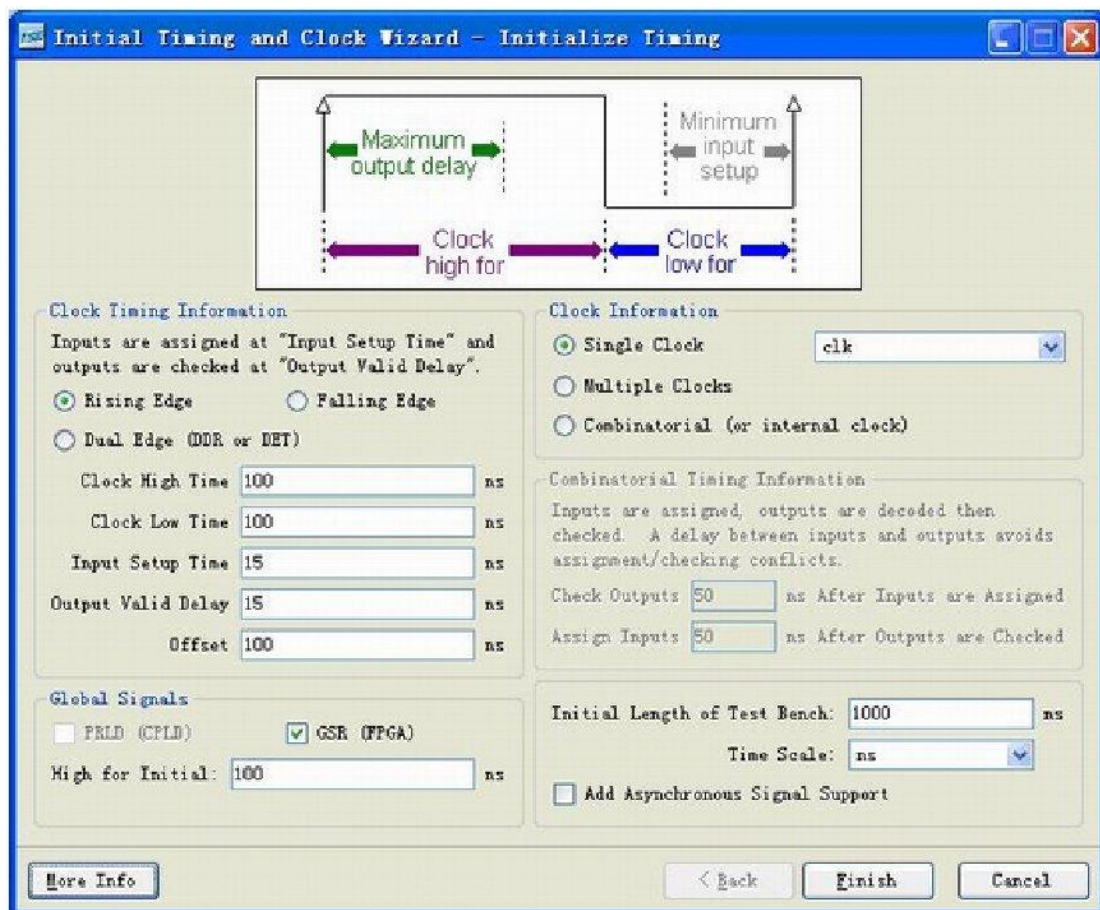


图 4-31 时序初始化窗口

时钟高电平时间和时钟低电平时间一起定义了设计操作必须达到的时钟周期，输入建立时间定义了输入在什么时候必须有效，输出有效延时定义了有效时钟延时到达后多久必须输出有效数据。默认的初始化时间设置如下：

- 时钟高电平时间（Clock High Time）：100ns
- 时钟低电平时间（Clock Low Time）：100ns
- 输入建立时间（Input Setup）：15ns
- 输出有效时间（Output Valid）：15ns
- 偏移时间（Offset）：100ns

单击“OK”按钮，接受默认的时间设定。测试矢量波形显示如图 4-32 所示。

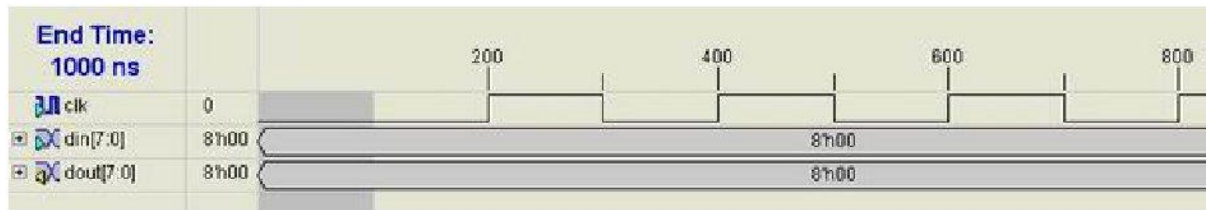


图 4-32 测试矢量波形

接下来，初始化输入（注：灰色的部分不允许用户修改），修改的方法为：选中信号，在其波形上单击，从该点击所在周期开始，在往后所有的时间单元内该信号电平反相。点击 **din** 信号前面的“+”号，在 **din[7]** 的第 2 个时钟周期内单击，使其变高；在 **din[6]** 的第 3 个时钟周期内单击，使其变高；同样的方法修改 **din[5]~din[0]** 信号，使其如图 4-33 所示。

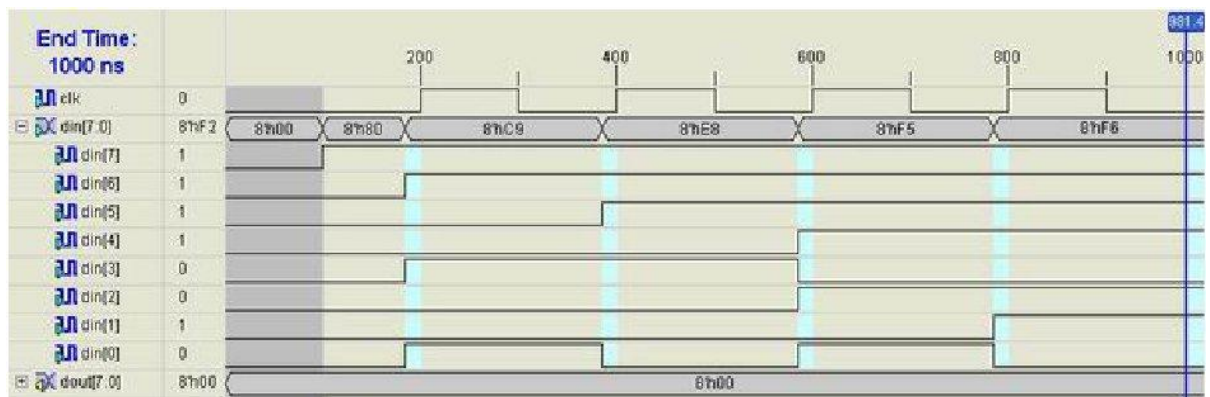


图 4-33 初始化输入

然后将 **testbench** 文件存盘，则 **ISE** 会自动将其加入到仿真的分层结构中，在代码管理区会列出刚生成的测试文件 **test_bench.tbw**，如图 4-34 所示。

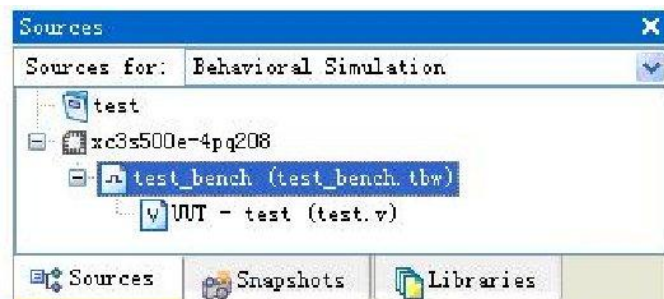


图 4-34 测试文件列表

选中 **test_bench.tbw** 文件，然后双击过程管理区的“**Simulate Behavioral Model**”，即可完成

功能仿真。同样，可在“Simulate Behavioral Model”选项上单击右键，设置仿真时间等。例 4-3 的仿真结果如图 4-35 所示。从中，可以看出，dout 信号等于 din 信号加 1，功能正确。

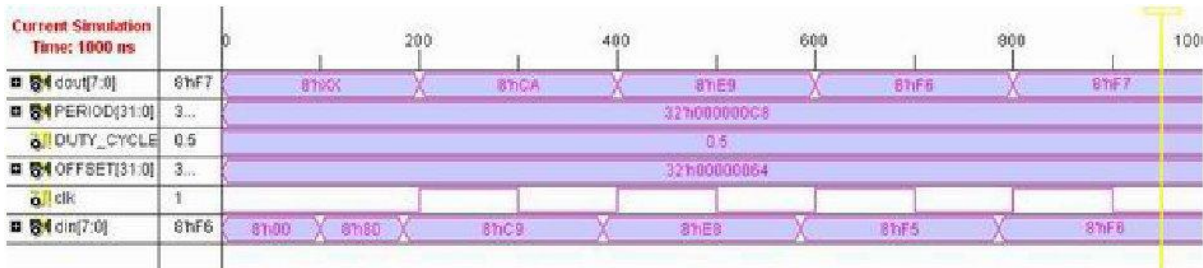


图 4-35 功能仿真结果

2. 测试代码法

下面介绍基于 Verilog 语言建立测试平台的方法。首先在工程管理区将“Sources for”设置为 Behavioral Simulation，在任意位置单击鼠标右键，并在弹出的菜单中选择“New Source”命令，然后选中“Verilog Test Fixture”类型，输入文件名为“test_test”，再点击“Next”进入下一页。这时，工程中所有 Verilog Module 的名称都会显示出来，设计人员需要选择要进行测试的模块。

用鼠标选中 test，点击“Next”后进入下一页，直接点击“Finish”按键，ISE 会在源代码编辑区自动显示测试模块的代码：

```
`timescale 1ns / 1ps
module test_test_v;
// Inputs
reg clk;
reg [7:0] din;
// Outputs
wire [7:0] dout;

// Instantiate the Unit Under Test (UUT)
test uut (
    .clk(clk),
    .din(din),
    .dout(dout)
);
```

```

initial begin
    // Initialize Inputs
    clk = 0;
    din = 0;
    // Wait 100 ns for global reset to finish
    #100;
    // Add stimulus here
end
endmodule

```

由此可见，ISE 自动生成了测试平台的完整架构，包括所需信号、端口声明以及模块调用的完成。所需的工作就是在 `initial...end` 模块中的“// Add stimulus here”后面添加测试向量生成代码。添加的测试代码如下：

```

forever begin
    #5;
    clk = !clk;
    if(clk == 1)
        din = din + 1;
    else
        din = din;
end
end

```

完成测试平台后。在工程管理区将“Sources for”选项设置为 Behavioral Simulation，这时在过程管理区会显示与仿真有关的进程，如图 4-36 所示。

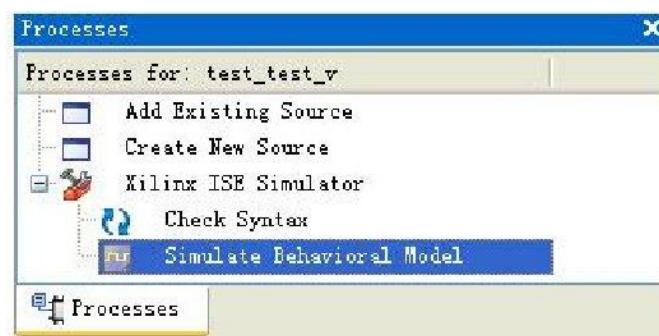


图 4-36 选择待测模块对话框

选中图 4-36 中 Xilinx ISE Simulator 下的 Simulate Behavioral Model 项，点击鼠标右键，选择弹出菜单的 Properties 项，会弹出如图 4-37 所示的属性设置对话框，最后一行的 Simulation Run Time 就是仿真时间的设置，可将其修改为任意时长，本例采用默认值。

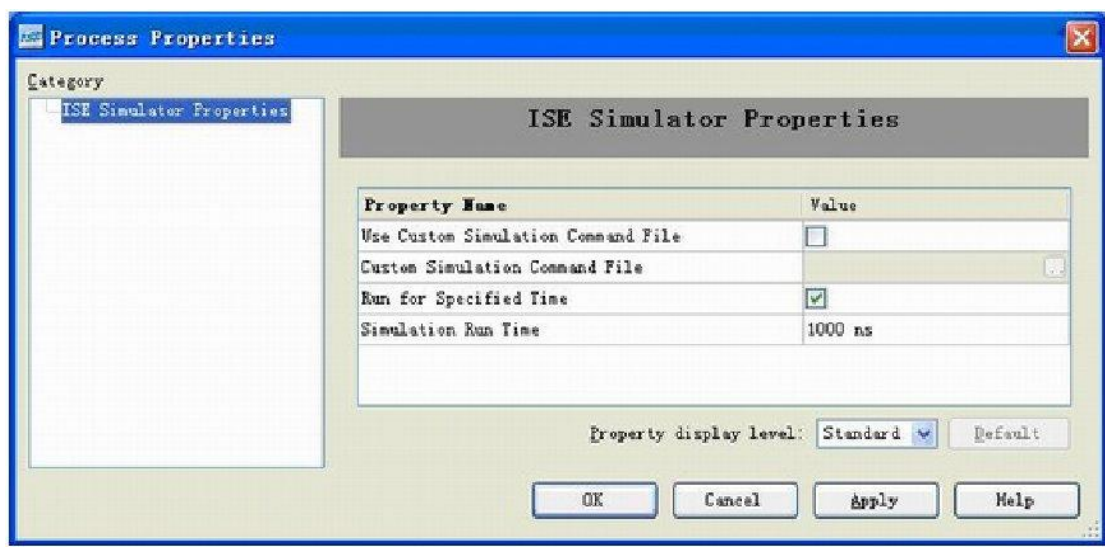


图 4-37 仿真过程示意图

仿真参数设置完后，就可以进行仿真了，直接双击 ISE Simulator 软件中的 Simulate Behavioral Model，则 ISE 会自动启动 ISE Simulator 软件，并得到如图 4-38 所示的仿真结果，从中可以看到设计达到了预计目标。

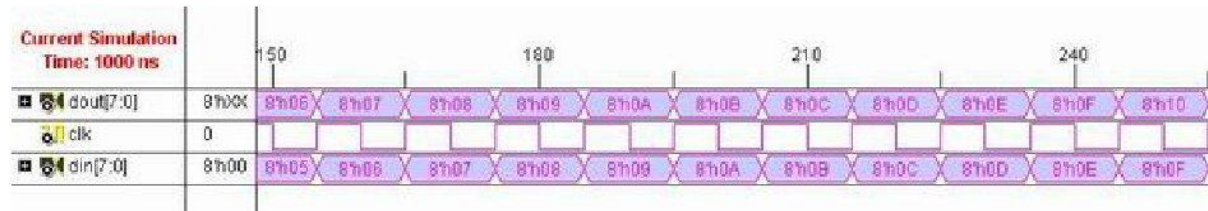


图 4-38 test 模块的仿真结果

4.3.3 基于 ISE 的实现

所谓实现（Implement）是将综合输出的逻辑网表翻译成所选器件的底层模块与硬件原语，将设计映射到器件结构上，进行布局布线，达到在选定器件上实现设计的目的。实现主要分为 3 个步骤：翻译（Translate）逻辑网表，映射（Map）到器件单元与布局布线（Place & Route）。翻译的主要作用是将综合输出的逻辑网表翻译为 Xilinx 特定器件的底层结构和硬件原语（具体的原语详见第 3 章中的原语介绍）。映射的主要作用是将设计映射到具体型号的器件上（LUT、FF、Carry 等）。布局布线步骤调用 Xilinx 布局布线器，根据用户约束和物理约束，对设计模块进行实际的布局，并根据设计连接，对布局后的模块进行布线，产生 FPGA/CPLD 配置文件。

1. 翻译过程

在翻译过程中，设计文件和约束文件将被合并生成 **NGD**（原始类型数据库）输出文件和 **BLD** 文件，其中 **NGD** 文件包含了当前设计的全部逻辑描述，**BLD** 文件是转换的运行和结果报告。实现工具可以导入 **EDN**、**EDF**、**EDIF**、**SEDIF** 格式的设计文件，以及 **UCF**（用户约束文件）、**NCF**（网表约束文件）、**NMC**（物理宏库文件）、**NGC**（含有约束信息的网表）格式的约束文件。翻译项目包括 3 个命令：

【**Translation Report**】用以显示翻译步骤的报告；

【**Floorplan Design**】用以启动 Xilinx 布局规划器（**Floorplanner**）进行手动布局，提高布局器效率；

【**Generate Post-Translate Simulation Model**】用以产生翻译步骤后仿真模型，由于该仿真模型不包含实际布线时延，所以有时省略此仿真步骤。

2. 映射过程

在映射过程中，由转换流程生成的 **NGD** 文件将被映射为目标器件的特定物理逻辑单元，并保存在 **NCD**（展开的物理设计数据库）文件中。映射的输入文件包括 **NGD**、**NMC**、**NCD** 和 **MFP**（映射布局规划器）文件，输出文件包括 **NCD**、**PCF**（物理约束文件）、**NGM** 和 **MRP**（映射报告）文件。其中 **MRP** 文件是通过 **Floorplanner** 生成的布局约束文件，**NCD** 文件包含当前设计的物理映射信息，**PCF** 文件包含当前设计的物理约束信息，**NGM** 文件与当前设计的静态时序分析有关，**MRP** 文件是映射的运行报告，主要包括映射的命令行参数、目标设计占用的逻辑资源、映射过程中出现的错误和告警、优化过程中删除的逻辑等内容。映射项目包括如下命令：

【**Map Report**】用以显示映射步骤的报告；

【**Generate Post-Map Static Timing**】产生映射静态时序分析报告，启动时序分析器（**Timing Analyzer**）分析映射后静态时序；

【**Manually Place & Route （FPGA Editor）**】用以启动 **FPGA** 底层编辑器进行手动布局布线，指导 Xilinx 自动布局布线器，解决布局布线异常，提高布局布线效率；

【**Generate Post-Map Simulation Model**】用以产生映射步骤后仿真模型，由于该仿真模型不包含实际布线时延，所以有时也省略此仿真步骤。

3. 布局和布线过程

布局和布线（**Place & Route**）：通过读取当前设计的 **NCD** 文件，布局布线将映射后生成的物理逻辑单元在目标系统中放置和连线，并提取相应的时间参数。布局布线的输入文件包括 **NCD** 和 **PCF** 模板文件，输出文件包括 **NCD**、**DLY**（延时文件）、**PAD** 和 **PAR** 文件。在布局布线的输出文件中，**NCD** 包含当前设计的全部物理实现信息，**DLY** 文件包含当前设计的网络延时信息，**PAD** 文件包含当前设计的输入输出（**I/O**）管脚配置信息，**PAR** 文件主要包括布局布线的命令行参数、布局布线中出现的错误和告警、目标占用的资源、未布线网络、网

络时序信息等内容。布局布线步骤的命令与工具非常多：

【Place & Route Report】用以显示布局布线报告；

【Asynchronous Delay Report】用以显示异步实现报告；

【Pad Report】用以显示管脚锁定报告；

【Guide Results Report】用以显示布局布线指导报告，该报告仅在使用布局布线指导文件 NCD 文件后才产生；

【Generate Post-Place & Route Static Timing】包含了进行布局布线后静态时序分析的一系列命令，可以启动 Timing Analyzer 分析布局布线后的静态时序；

【View/Edit Place Design (Floorplanner)】和【View/Edit Place Design (FPGA Editor)】用以启动 Floorplanner 和 FPGA Editor 完成 FPGA 布局布线的结果分析、编辑，手动更改布局布线结果，产生布局布线指导与约束文件，辅助 Xilinx 自动布局布线器，提高布局布线效率并解决布局布线中的问题；

【Analyze Power (XPower)】用以启动功耗仿真器分析设计功耗；

【Generate Post-Place & Route Simulation Model】用以产生布局布线后仿真模型，该仿真模型包含的时延信息最全，不仅包含门延时，还包含了实际布线延时。该仿真步骤必须进行，以确保设计功能与 FPGA 实际运行结果一致；

【Generate IBIS Model】用以产生 IBIS 仿真模型，辅助 PCB 布板的仿真与设计；

【Multi Pass Place & Route】用以进行多周期反复布线；

【Back-annotate Pin Locations】用以反标管脚锁定信息

经过综合后，在过程管理区双击“Implement Design”选项，就可以完成实现，如图 4-39 所示。经过实现后能够得到精确的资源占用情况，如图 4-40 所示。

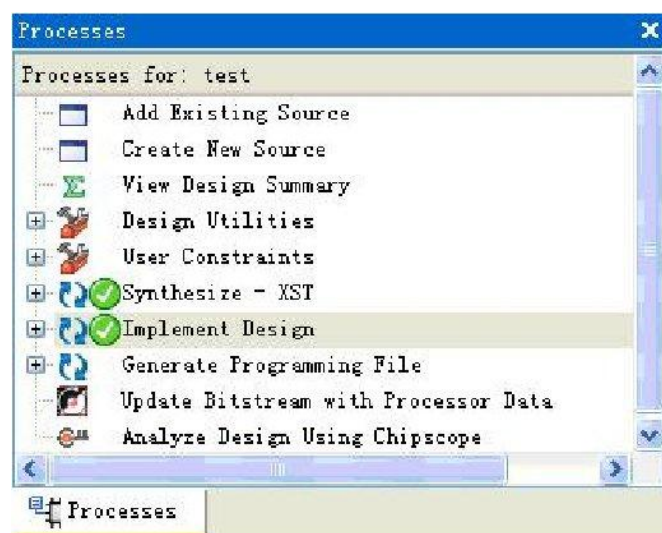


图 4-39 设计实现窗口

```

Design Summary
-----
Number of errors:      0
Number of warnings:    1
Logic Utilization:
  Number of 4 input LUTs:      8 out of 30,720    1%
Logic Distribution:
  Number of occupied Slices:      5 out of 15,360    1%
    Number of Slices containing only related logic:      5 out of      5 100%
    Number of Slices containing unrelated logic:          0 out of      5   0%
    *See NOTES below for an explanation of the effects of unrelated logic
Total Number of 4 input LUTs:    9 out of 30,720    1%
  Number used as logic:          8
  Number used as a route-thru:    1
  Number of bonded IOBs:        17 out of    448    3%
  Number of BUFG/BUFGCTRLs:      1 out of     32    3%
    Number used as BUFGs:        1
    Number used as BUFGCTRLs:    0

```

图 4-40 实现后的资源统计结果

4. 实现属性设置

一般在综合时，所有的属性都采用默认值。实际上 ISE 提供了丰富的实现属性设置。下面对 ISE9.1 中内嵌的 XST 属性进行说明。打开 ISE 中的设计工程，在过程管理区选中“Implement Design”并单击右键，弹出界面如图 4-41 所示，包括翻译、映射、布局布线以及后仿时序参数等。

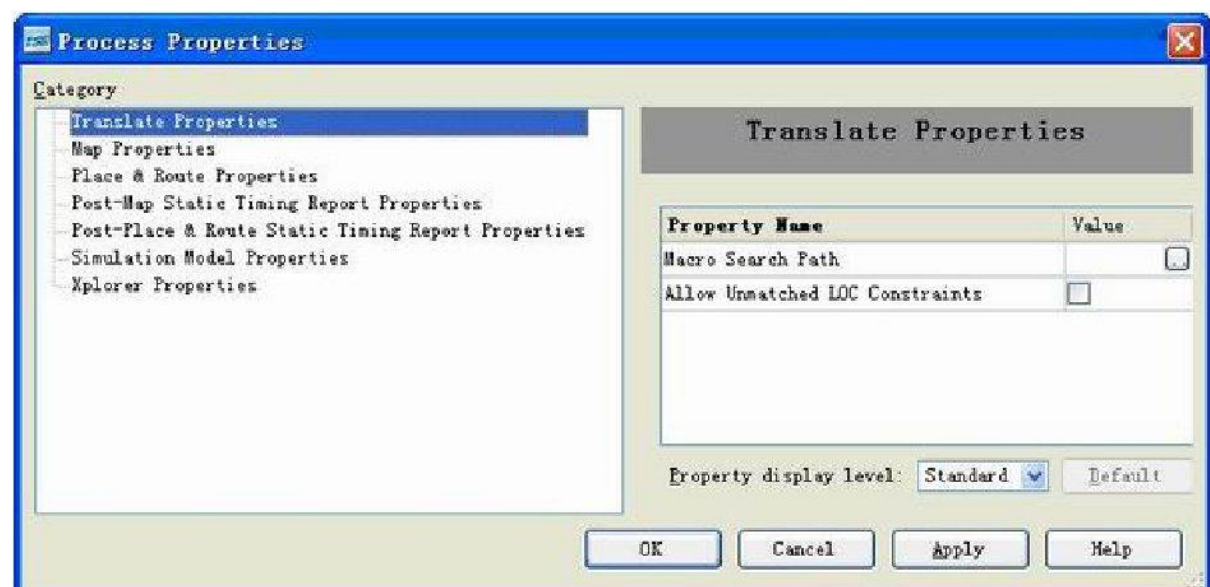


图 4-41 实现属性设置窗口

1. 翻译参数设置窗口

【Macro Search Path】：宏查找路径。用于提供宏的存放路径。

【Allow Unexpanded Blocks】：允许未展开的逻辑块。用来说明当遇到不能展开 NGD 原语的块时，NGDBuild 工具是否继续运行。如果在设计中没有较低级的模块，该参数允许 NGDBuild 运行结束而不出现错误。默认值为【False】。

2. 映射参数设置窗口

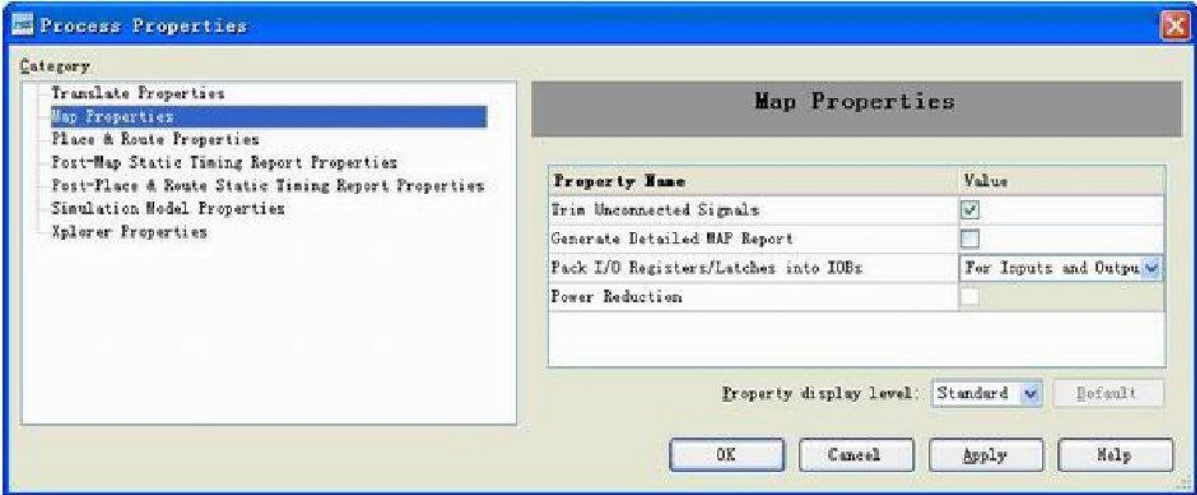


图 4-42 映射参数设置窗口

【Trim Unconnected Signals】：整理未连接的信号。该参数用于控制在映射之前，是否整理未连接的逻辑单元和连线。该参数有助于评估设计中的逻辑资源，并获得部分设计的时序信息。默认值为需要整理。

【Generate Detailed MAP Report】：生成详细的映射报告。该参数用来选择是否需要生成详细的映射报告。详细的映射报告将提示在映射时去掉的多余逻辑块和信号，以及提示展开的逻辑，交叉引用的信号、符号等。默认值为不产生详细的映射报告。

【Pack I/O Registers/Latches into IOBs】：选择输入输出块中的寄存器/锁存器。该参数用来控制是否将器件内部的输入/输出寄存器用 IOBs 中的寄存器/锁存器来取代。可以选择：①【For Inputs and Outputs】，尽可能将设计中输入/输出寄存器放入 IOBs；②【For Inputs Only】，仅考虑把输入寄存器放入 IOBs；③【For Outputs Only】，仅考虑把输出寄存器放入 IOBs；④【Off】，采用用户的设计要求进行处理，不考虑自动选择方式。

3. 布局布线参数设置窗口

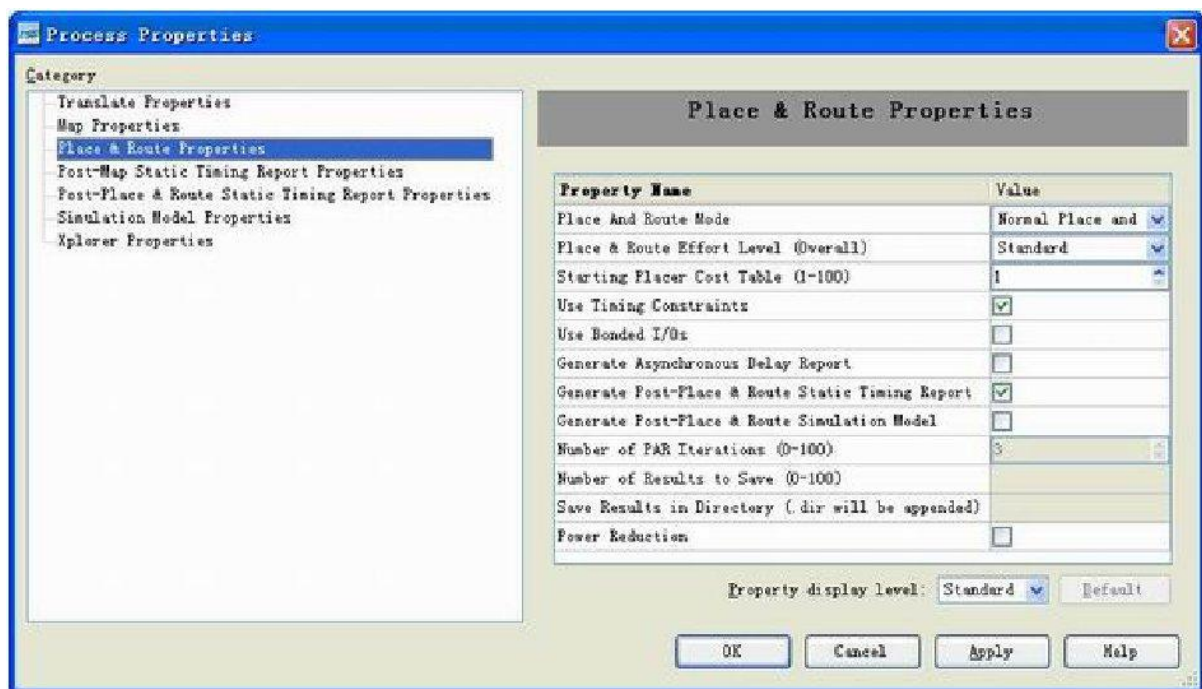


图 4-43 布局布线参数设置窗口

【Place And Route Mode】：布局布线方式。该参数用来指定采用哪种方式来进行布局布线处理。可以选择：①【Normal Place and Route】，一般的布局布线处理，该方式为默认值；②【Place Only】，运行所选择的布局布线努力程度，但不运行布线器，当选择该参数后，布局布线器至少运行一次；③【Route Only】，运行所选择的布局布线努力程度，但不运行布局器，当选择该参数后，布局布线器至少运行一次；④【Reentrant Route】，重复布线，保持布局布线方式，布线器用当前的路由再一次布线。该布线器由努力程度来控制。

【Place & Route Effort Level (Overall)】：全局的布局布线努力程度。该参数控制布局布线流程的努力程度和运行次数。根据需要可以选择【Standard】、【Medium】和【High】。如果选择【Standard】，将会有最快的运行时间，但不会有好的布局布线效果，不适合于复杂的逻辑设计；如果选择【High】，将会对逻辑设计进行反复的布局布线处理，并生成最理想的布局布线结果，对高性能、复杂和最终的设计通常采用这种模式，但比较费时。默认值为【Standard】。

【Starting Placer Cost Table (1-100)】：布局器运行开销表。默认值为 1。

【Use Timing Constraints】：使用时序约束。在布局布线期间，需使用 UCF 和 PCF 文件中时序约束条件。默认值为使用时序约束。

【Use Bonded I/Os】：使用绑定的 I/Os。该参数用来选择是否允许布局布线器将内部的输入输出逻辑放到 I/Os 脚未使用的用于绑定 I/Os 的位置。该参数也允许布线资源穿过用于绑定 I/Os 的位置。默认值为该参数无效。

【Generate Asynchronous Delay Report】：生成异步延迟报告。该参数用来选择是否

在布局布线运行时生成异步延迟报告。该报告列出了设计中所有的网线和网络上所有负载的延迟。通过执行【Asynchronous Delay Report Process】，可以打开该报告。默认值为不生成异步延迟报告。

【Generate Post-Place & Route Static Timing Report】：生成布局布线后的静态时序报告。该参数用来选择是否在布局布线后生成静态时序报告，该报告列出了设计中所有信号通道的最坏条件时序特性。通过执行【Post-Place & Route Static Timing Report Process】，可以打开该报告。默认值为生成布局布线后的静态时序报告。

【Generate Post-Place & Route Simulation Model】：生成布局布线后的仿真模型。该参数用来选择是否在布局布线后生成仿真模型。如果选择需要生成该模型，需要在【Simulation Model Properties】中选择仿真模型参数。默认值为不生成仿真模型。

- 【Number of PAR Iterations (0-100)】：
- 【Number of Results to Save (0-100)】：
- 【Save Results in Directory (.dir will be appended)】：
- 【Power Reduction】：

4. 映射后静态时序报告参数设置窗口

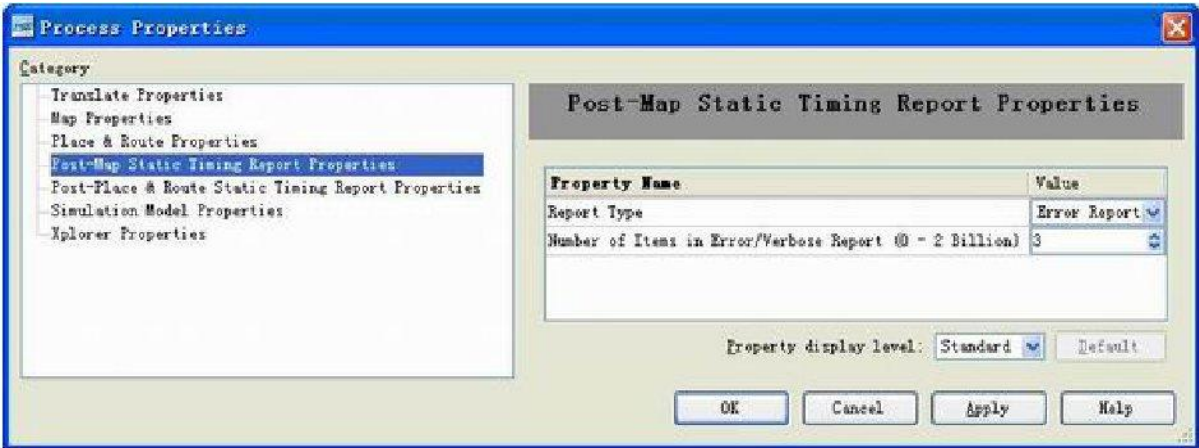


图 4-44 映射后静态时序报告参数设置窗口

- 【Report Type】：
- 【Number of Items in Error/Verbase Report (0-2 Billion)】：

5. 布局布线后静态时序报告参数设置窗口

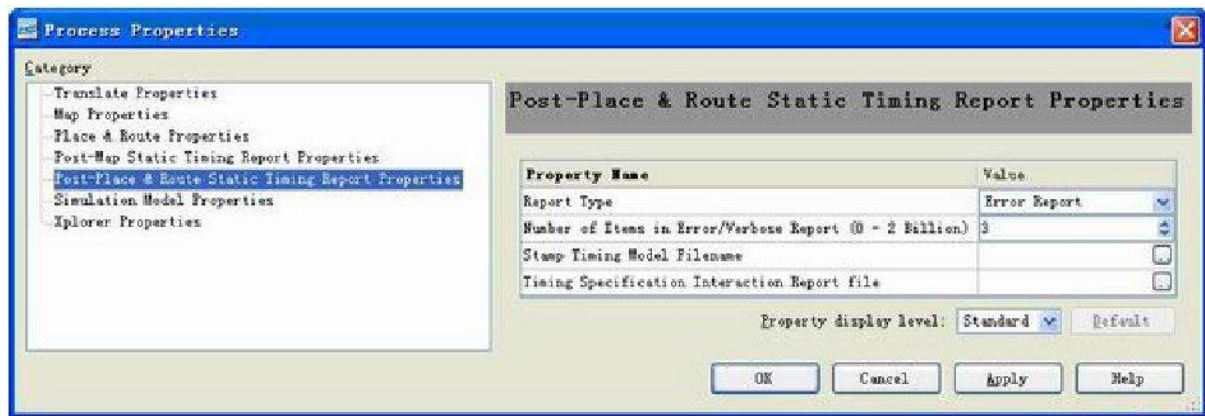


图 4-45 布局布线后静态时序报告参数设置窗口

【Report Type】：

【Number of Items in Error/Verbose Report (0-2 Billion)】：

【Stamp Timing Model Filename】：

【Timing Specification Interaction Report file】：

6. 仿真模型参数设置窗口

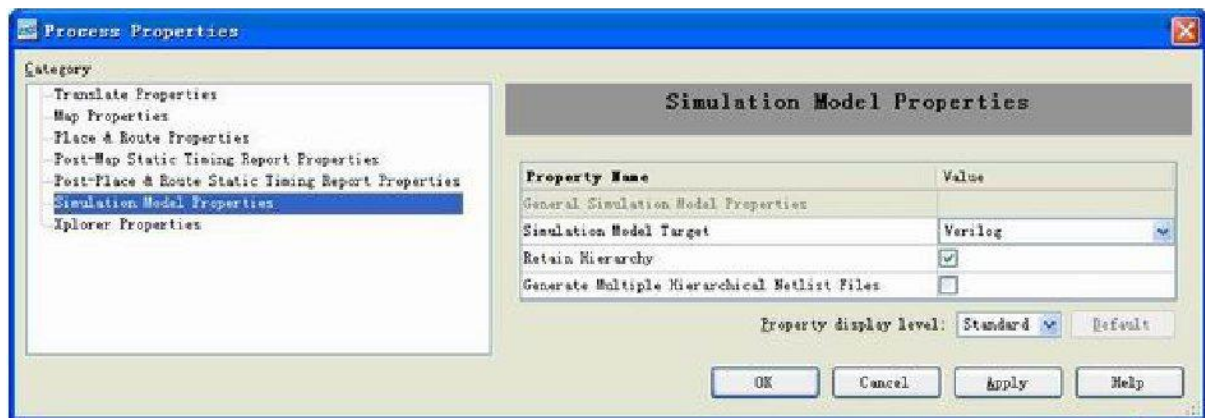


图 4-46 仿真模型参数设置窗口

【General Simulation Model Properties】：

【Simulation Model Target】：

【Retain Hierarchy】：

【Generate Multiple Hierarchical Netlist Files】：

7. Xplorer 参数设置窗口

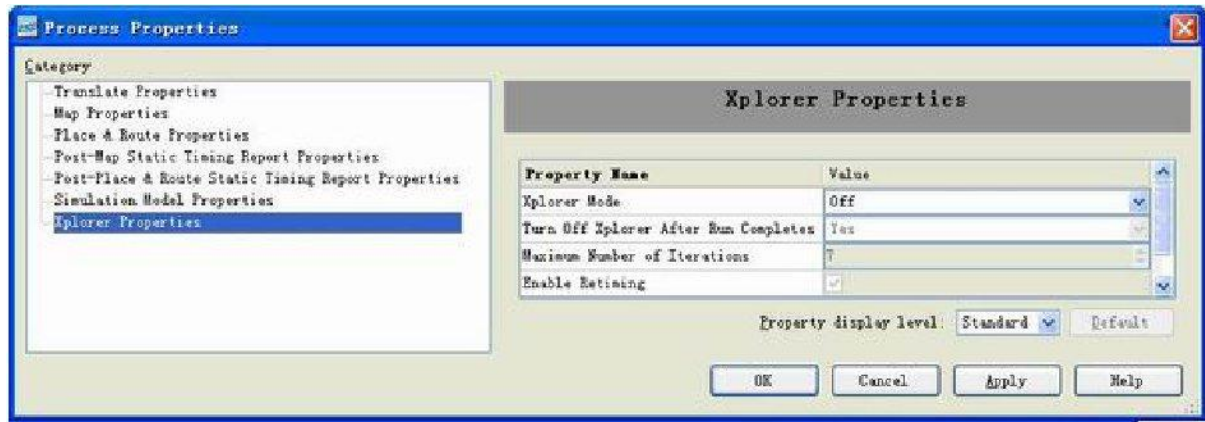


图 4-47 Xplorer 参数设置窗口

【Xplorer Mode】：

【Turn off Xplorer After Run Completes】：

【Maximum Number of Iterations】：

【Enable Retiming】：

4.3.4 基于 ISE 的硬件编程

本节简要介绍 ISE 软件中的硬件编程流程，详细的配置电路原理以及软件配置参数将在第 5 章讲解。生成二进制编程文件并下载到芯片中，也就是所谓的硬件编程和下载，是 FPGA 设计的最后一步。生成编程文件在 ISE 中的操作非常简单，在过程管理区中双击 **Generate Programming File** 选项即可完成，完成后则该选项前面会出现一个打钩的圆圈，如图 4-36 所示。生成的编程文件放在 ISE 工程目录下，是一个扩展名为 .bit 的位流文件。

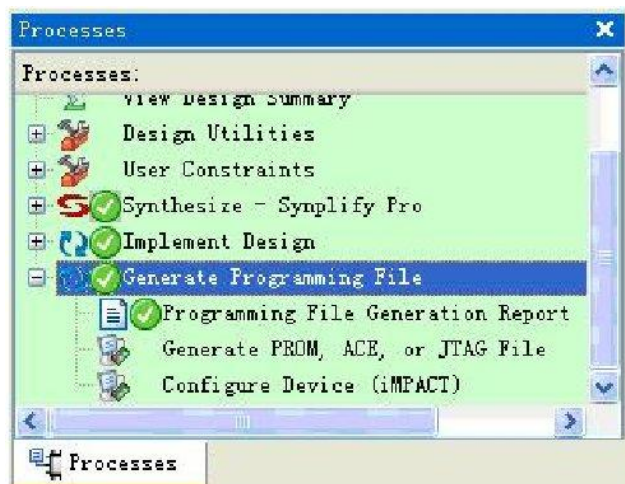


图 4-48 生成编程文件的窗口

到此，只剩下完成设计的最后一步——下载。双击过程管理区的 **Generate Programming File** 选项下面的 **Configure Device (iMPACT)** 项，然后在弹出的 **Configure Device** 对话框中选取合适的下载方式，ISE 会自动连接 FPGA 设备。成功检测到设备后，会出现如图 4-37 所示的 **iMPACT** 的主界面。

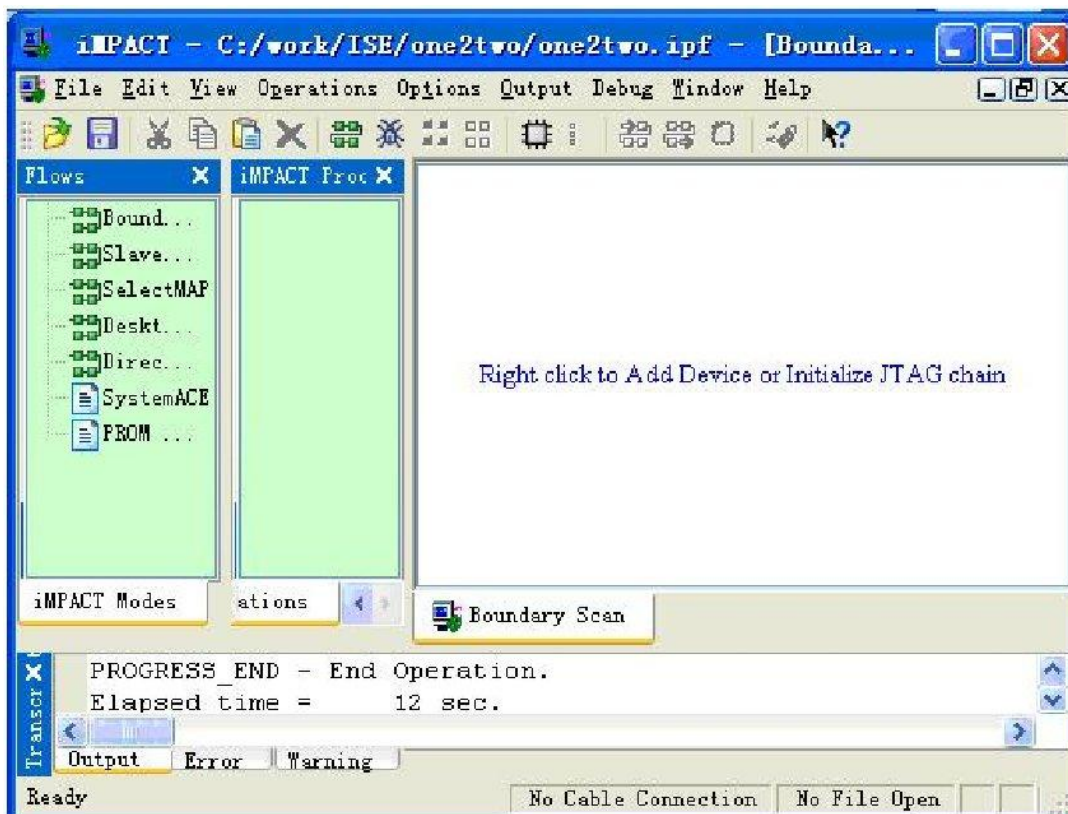
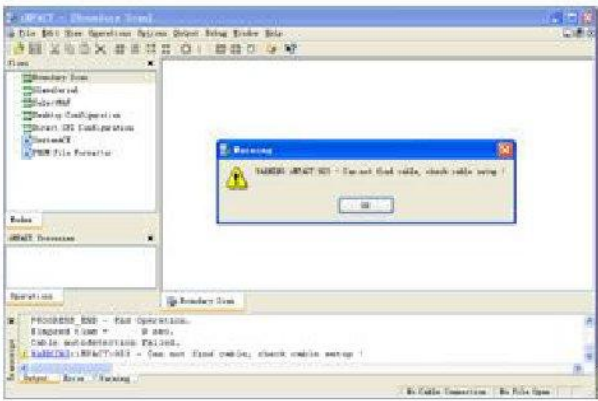
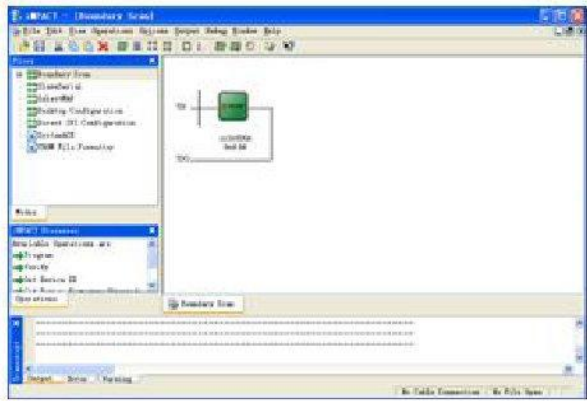


图 4-49 iMPACT 主界面

在主界面的中间区域内单击鼠标右键，并选择菜单的“Initialize Chain”选项，如果 FPGA 配置电路 JTAG 测试正确，则会将 JTAG 链上扫描到的所有芯片在 iMPACT 主界面上列出来，如图 4-35（a）所示；如果 JTAG 链检测失败，其弹出的对话框如图 4-35（b）所示。



(a) JTAG 链扫描正确后的窗口界面

(b) JTAG 链扫描正确后的窗口

界面

图 4-50 JTAG 链扫描结果示意图

JTAG 链检测正确后，在期望 FPGA 芯片上单击右键，在弹出的菜单中选择“Assign New Configuration File”，会弹出图 4-36 的窗口，让用户选择期望后缀为 .bit 的二进制比特流文件。

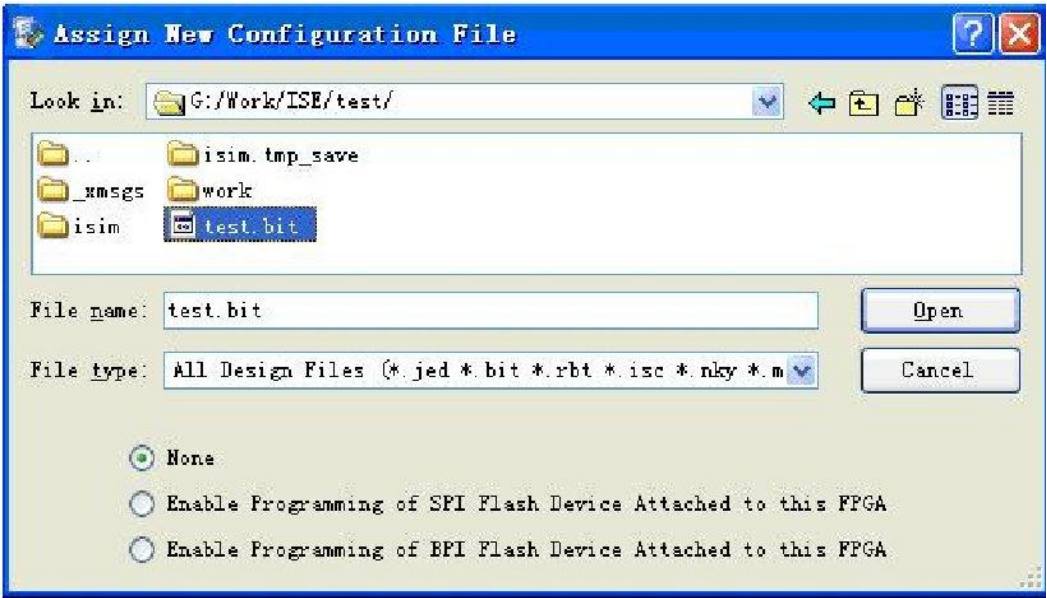


图 4-51 选择位流文件

选中下载文件后，单击“打开”按钮，在 iMPACT 的主界面会出现一个芯片模型以及位流文件的标志，在此标志上单击鼠标右键，在弹出的对话框中选择 Program 选项，就可以对 FPGA 设备进行编程，如图 4-37 所示

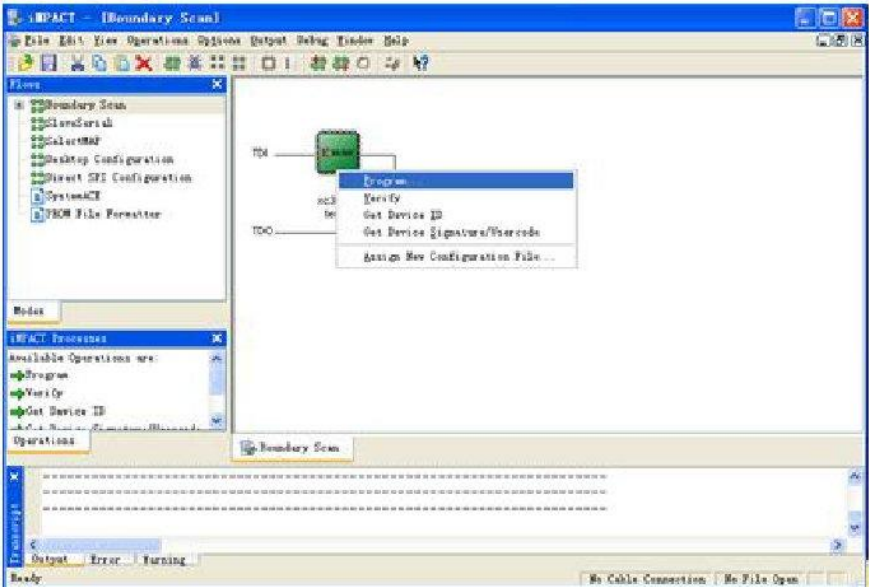


图 4-52 对 FPGA 设备进行编程示意图

配置成功后，会弹出配置成功的界面，如图 4-38 所示。

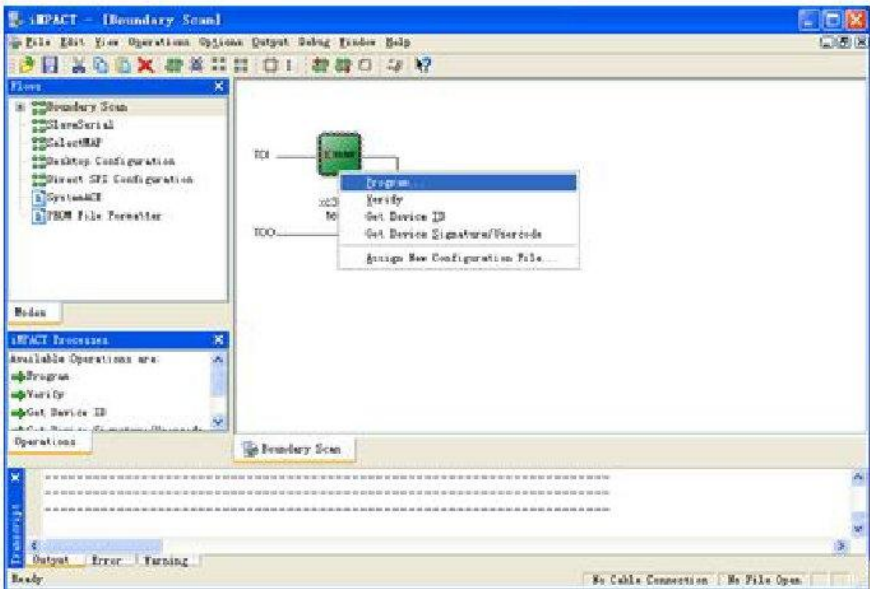


图 4-53 FPGA 配置成功指示界面

至此，就完成了整个完整的 FPGA 设计流程。当然，ISE 的功能十分强大，以上介绍只是其

中最基本的操作，更多的内容和操作需要读者通过阅读 ISE 在线帮助来了解，在大量的实际实践中来熟悉。

4.3.5 功耗分析以及 XPower 的使用

1. 功耗分析简介

在 FPGA 设计中，功耗分析是成功设计的重要环节。针对 FPGA 设计中的功耗分析，Xilinx 公司推出了简单的速查表格和专用的功耗分析工具——XPower。对于开发初期的 FPGA 功耗估算，设计者一般使用 Xilinx 公司提供的简单图表和公式。例如，XC9500 系列器件的简单功耗分析可以在其数据手册中找到估算公式。Virtex-E 和 Virtex-II 系列器件的简单功耗分析可以通过 Xilinx 公司提供的功耗估算表格完成。对于基本完成逻辑设计的 FPGA 功耗估算，设计者可以使用 XPower 进行详细的功耗分析。

在 XPower 功耗分析过程中，其功耗估算的基本方法是：（1）计算每个设计单元的功耗。（2）累加各个设计单元的功耗。由于 CMOS 电路的动态功耗很大程度上取决于电路的翻转频率，所以 XPower 采用如下公式计算单个设计单元的功耗[M]：

$$P = C \times V^2 \times E \times f \times 1000$$

XPower 用户界面

XPower 的启动方法有两种：一种是单独启动 XPower，直接点击“开始”“程序”“Xilinx ISE 9.1i”“Accessories”“Xpower”即可启动；另一种是在工程经过布局布线后，在过程管理区双击“Implementation”“Place and Router”“Analyze Power (XPower)”，则打开 XPower，并自动加载当前工程。在例 M 的工程中，采用后一种方法打开 XPower，其用户界面如图 4-54 所示。

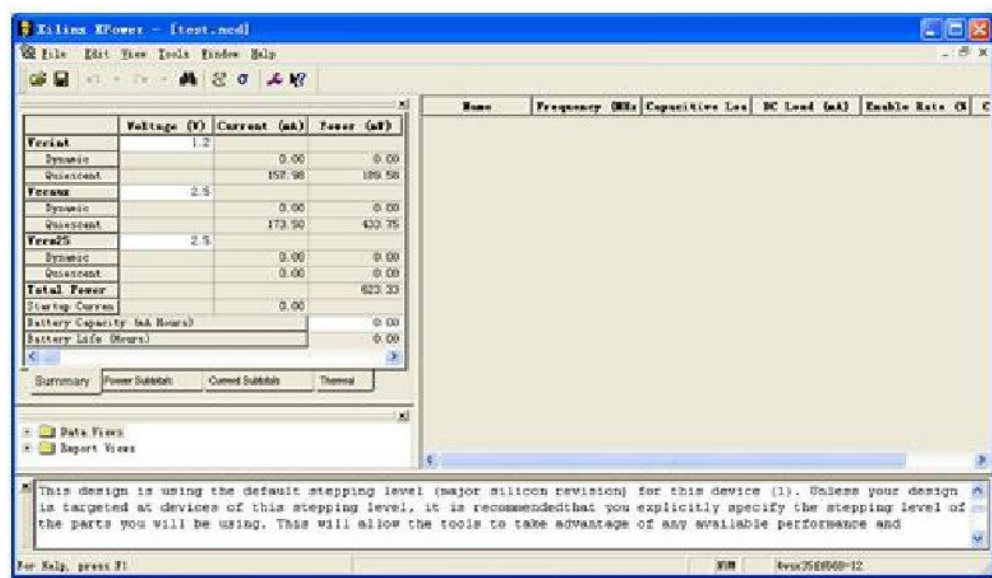


图 4-54 XPower 的启动

XPower 的用户界面由菜单栏、工具栏、功耗结果显示窗口、表格分析窗口、浏览窗口、信息显示窗口以及状态栏等部分组成。

在 ISE 过程管理区，双击“Generate Power Data”命令将自动生成设计的功耗分析报告，再双击“View XPower Report”命令来阅读报告。典型的分析报告（例 M 的功耗分析报告）如图 M 所示。报告会给出各个电压相应的电流大小以及功耗，便于设计人员设计相应的电源模块。

```

-----
Release 9.1.03i - XPower SoftwareVersion:J.33
Copyright (c) 1995-2007 Xilinx, Inc. All rights reserved.
Design:      test.ncd
Preferences: test.pcf
Part:        4vsx35ff668-12
Data version: ADVANCED,v1.0,08-26-04

```

```

Power summary:
-----
Total estimated power consumption: 623
-----
Vccint 1.20V: 158 190
Vccaux 2.50V: 174 434
Vcco25 2.50V: 0 0
-----
Inputs: 0 0
Logic: 0 0
Outputs:
Vcco25 0 0
Signals: 0 0
-----
Quiescent Vccint 1.20V: 158 190
Quiescent Vccaux 2.50V: 174 434

```

```

Thermal summary:
-----
Estimated junction temperature: 33C
Ambient temp: 25C
Case temp: 33C
Theta J-A range: 13 - 13C/W

```

```

Analysis completed: Sat Feb 09 18:33:11 2008
-----

```

图 4-55 典型的 Xpower 分析报告

• XPower 的操作流程

要准确利用 XPower 来分析功耗，关键是确定信号的工作频率、信号翻转率以及输出负载等参数。XPower 允许设计人员任意修改工作频率、信号翻转以及负载等参数，以便观察某一参数对功耗的影响。此外，XPower 支持 VCD 文件格式，可得到最全面的信号翻转信息，减少手工输入信号翻转信息的繁琐操作，提高功耗分析的效率和正确性。典型的 XPower 使用流程如下：

<1> 打开 XPower 软件，选择“File”菜单下的“Open”命令，则会弹出如图 M 的对话框。其中“Design File”栏用于输入设计文件，其后缀为.ncd；“Physical Constraints File”栏用于输入物理约束文件，其后缀为.pcf；“Setting File”栏用于输入设置文件，其后缀为.xml；“Simulation

File”栏用于输入仿真后的输出文件，其后缀为.vcd。显示类型（View Types）分为 Types 类型（分类视窗）和 Hierarchical 类型（分层视窗）两类视窗模式。

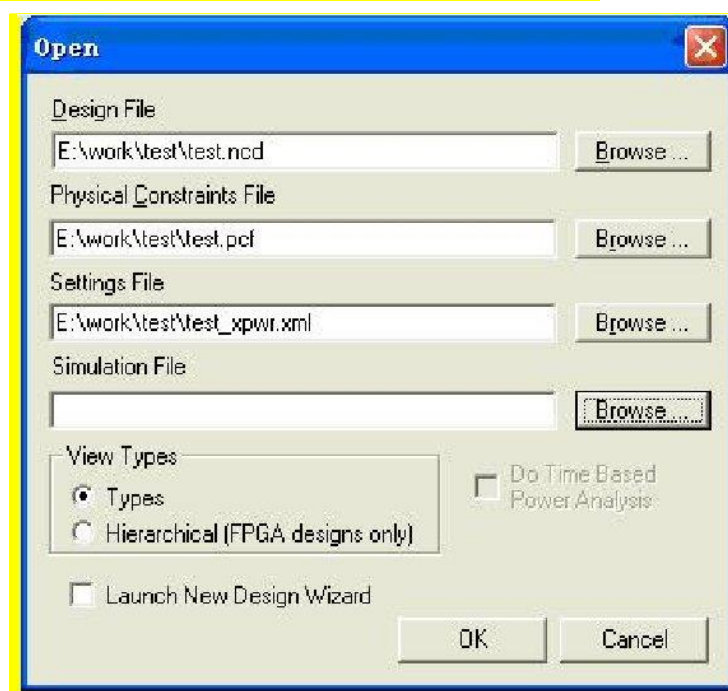


图 4-56 XPower 打开文件对话框

<2> 选中图 M 中的“Launch New Design Wizard”参数，点击“OK”按钮，启动设计向导，弹出如图 M 所示的对话框。

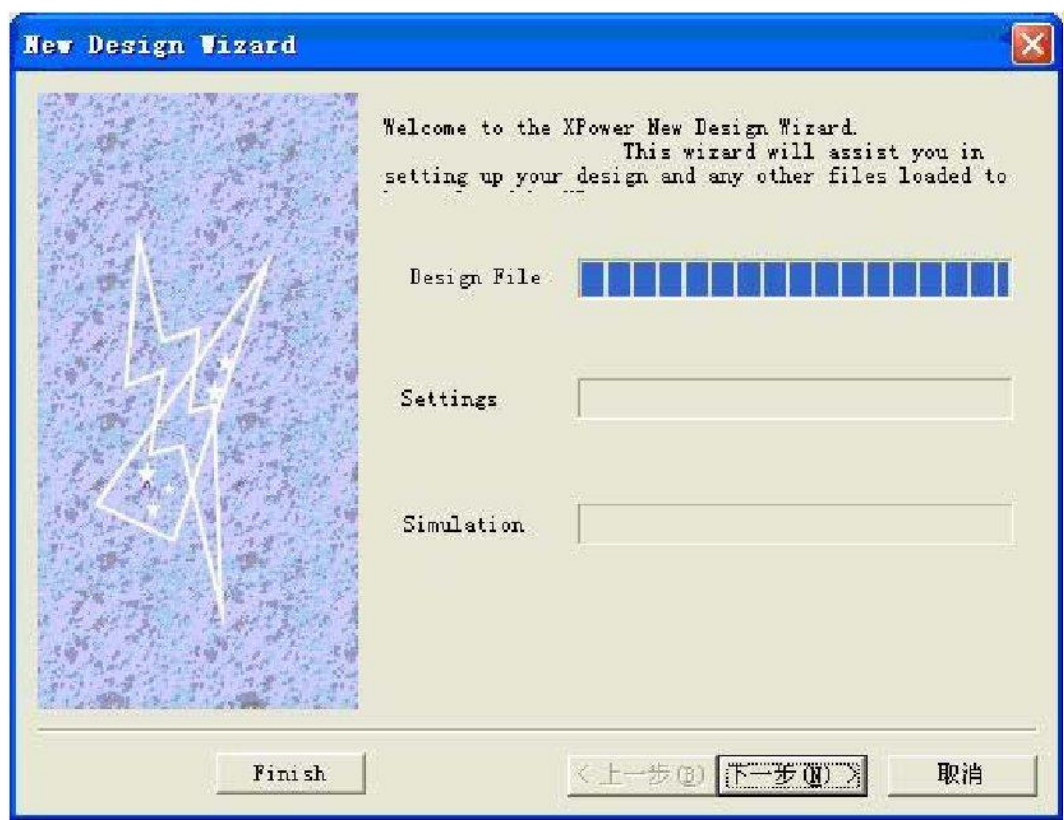


图 4-57 设计向导对话框

单击图 M 中的“下一步(N)”按钮，进入参数配置窗口，整体分为 4 个步骤。第 1 个步骤的界面如图 M 所示。



图 4-58 XPower 向导第 1 步

其中，“Voltage Source”用于设置 FPGA 芯片的电源参数，包括核电压、辅助电压以及 IO 端口参考电压，根据所选的器件输入相应的数值，一般选用默认值即可；“Battery Capacity and Battery Life”栏用于设定电池的容量和寿命，如输入电池容量为 1000mAH，则会自动给出电池的使用寿命为 3.02 小时；“Thermal”栏用于设定温度参数，“Ambient”用于设定环境温度，“Air”设定空气对流强度。

<3> 点击图 M 的“下一步 (N)”按键，进入设计向导第 2 步，设置各个信号的工作频率，包括同步信号和异步信号，其界面如图 M 所示。选中相应的信号，在“Frequency”文本框中输入大小，并选择合适的单位，单击“Apply”按键确认，也可点击“Reset”按键复位到初始值。

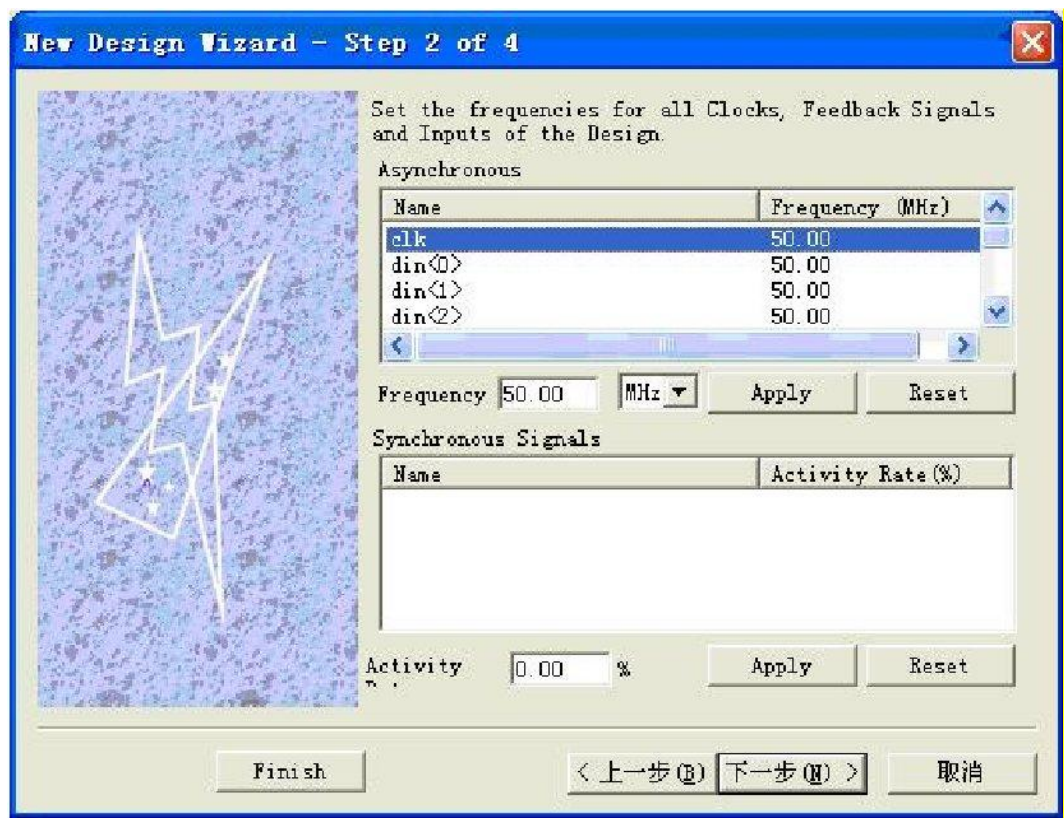


图 4-59 XPower 向导第 2 步

<4> 点击图 M 的“下一步 (N)”按钮，进入设计向导第 3 步，设置信号的输出负载电容，其界面如图 M 所示。选择信号后，输入电容大小，单击“Apply”按钮确认，也可以单击“Reset”按钮复位到初始值。

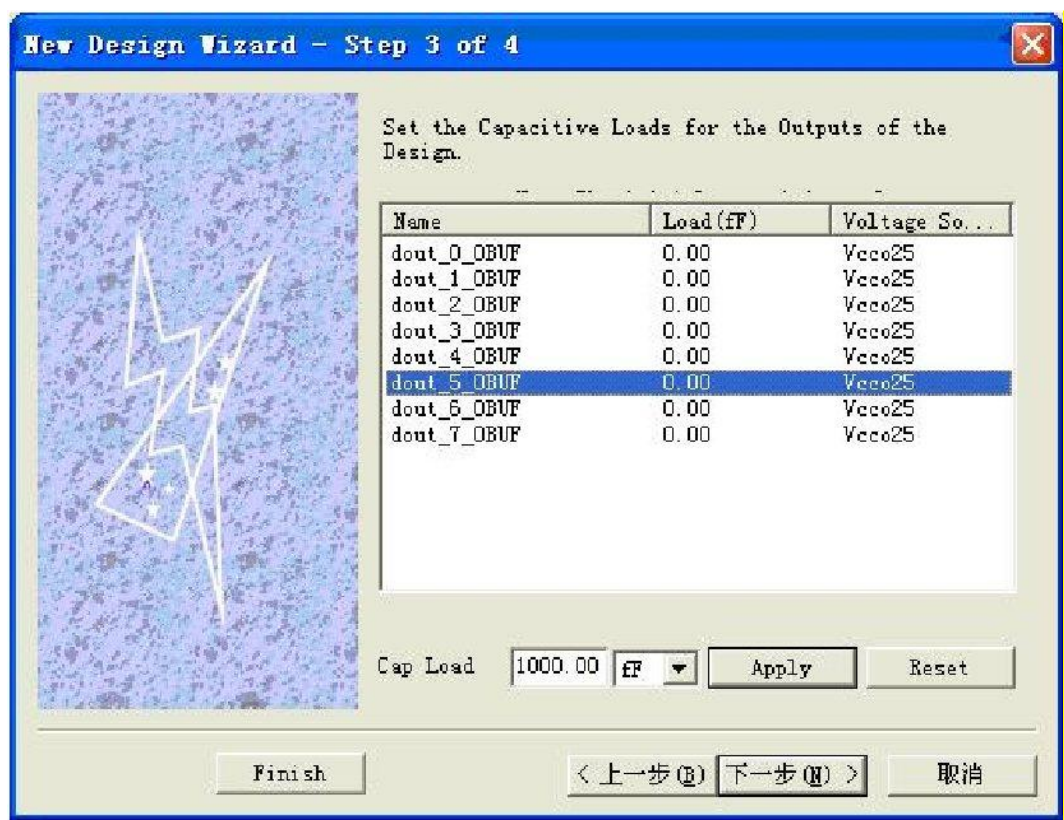


图 4- 60XPower 向导第 3 步

<5> 点击图 M 的“下一步 (N)”按钮，进入设计向导第 4 步，设置信号的直流电流负载，其界面如图 M 所示。选择信号后，输入电流大小，单击“Apply”按钮确认，也可以单击“Reset”按钮复位到初始值。此时，确认无误后，可点击“Finish”按钮完成设计向导，如有错误，可单击“上一步 (B)”返回到先前界面进行修改。

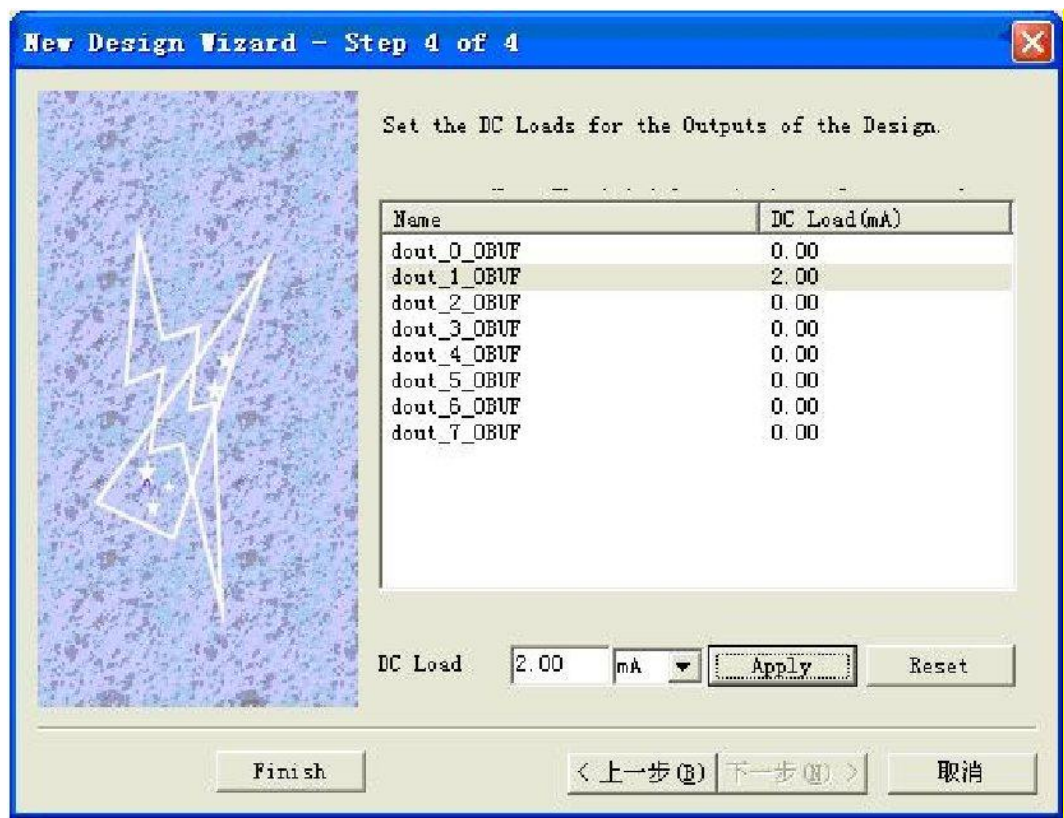


图 4-61 XPower 向导第 4 步

<6> 单击 XPower 软件的“File”菜单下的“Open Setting File...”命令，可加入设置文件；再选择“Open Simulation File...”命令，加入仿真工具生成的.vcd 文件。选择相应的约束文件后，XPower 会在信息提示窗口给出加载成功的指示信息。

等加载完约束文件后，可点击浏览器窗口的“Report Views” “Power Report (HTML)”，查看设计功耗分析结果，其界面如图 M 所示。

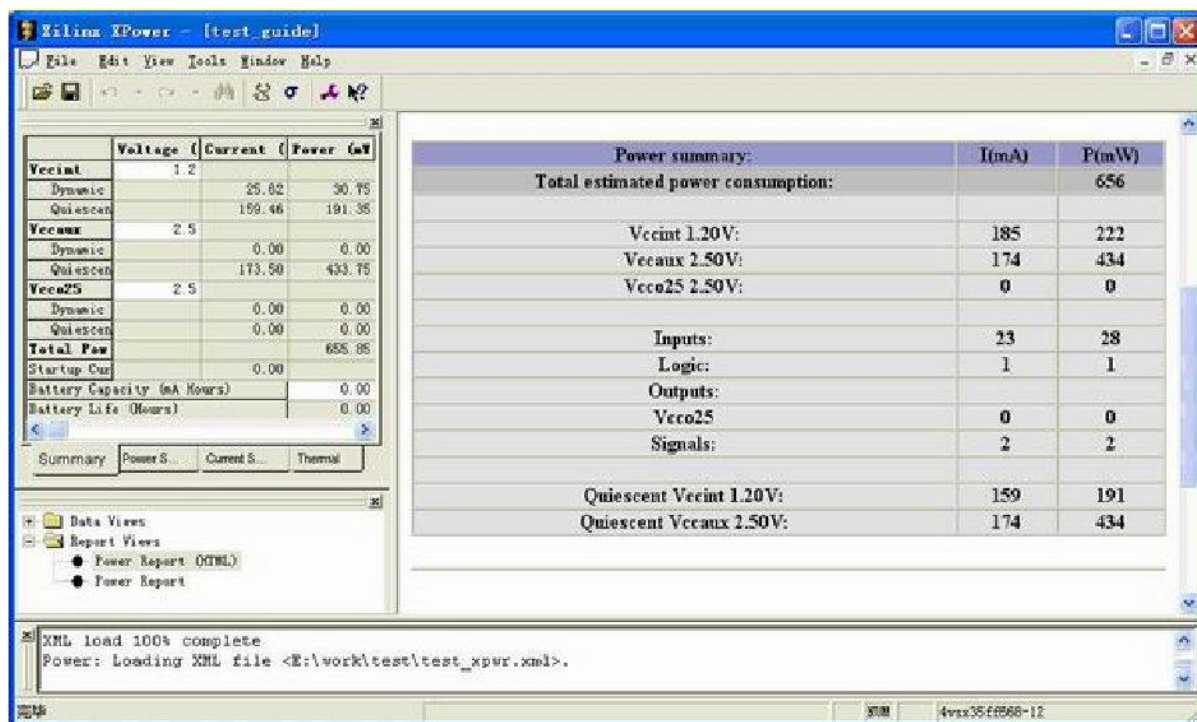


图 4-62 XPower 输出结果

VCD 文件

在 XPower 功耗分析过程中，为保证功耗分析报告基本符合实际情况，应该准确设置功耗分析参数。在 XPower 中，用户可以手工设置功耗分析参数，也可以通过读入 VCD 文件自动设置功耗分析参数。利用 VCD 文件设置功耗分析参数可以使当前设计的信号翻转频率、输出负载等参数更符合实际情况。在使用 ModelSim 进行仿真时，VCD 文件不是默认的输出文件，用户需要在仿真过程中指定输出相应的 VCD 文件。其中，仿真 Verilog HDL 设计文件时，需要在 Testbench 文件中加入适当语句，如下所示：

initial begin

```
// design.VCD 表示将要生成的 VCD 文件名
```

```
$dumpfile("design.VCD");
```

```
//testbench 表示测试名称， uut 表示待测试的模块名称
```

```
$dumpvars(1, testbench.uut);
```

end

仿真 VHDL 设计文件时，需要在 DO 文件中加入适当语句，如下所示：

```
-design.VCD 表示将要生成的 VCD 文件名;
```

.VCD file design.VCD

-- testbench 表示测试名称， uut 表示待测试的模块名称

.VCD add testbench /uut/*

3. 简易的功耗分析方法

Xilinx 提供了两种基于电子数据表和基于网站的简易功耗估计工具：一种叫做 Web Power Tools，另一种是 XPower 估计器。Web Power Tools 和 XPower 估计器可通过 http://china.xilinx.com/products/design_resources/power_central/ 获得，二者都提供了根据逻辑利用率大概估计做出的功耗估算，其区别在于：Web Power Tools 基于网站，用于早期芯片的功耗评估，支持 Virtex-II Pro、Virtex-II 以及 Virtex/Virtex-E 系列芯片；XPower 估计器基于电子数据表格，支持 Virtex5、Virtex-4、Spartan-3A DSP、Spartan-3A/3AN、Spartan-3E 以及 Spartan-3 等较新系列芯片。

Xilinx 会自动更新各个系列芯片的电子数据表格，用户下载相应的电子数据表格时，可看到最后一次更新的时间。这两种简易功耗评估工具可以仅凭设计利用率估计就能获得初步的功耗评估，而无需实际设计文件，是在设计流程的早期获得器件功耗情况的最快捷和最方便的方法。此外简易功耗评估工具不需要安装，只需要拥有互联网连接和 Web 浏览器，将电子表格下载到本地即可。

1) XPower 估计器的用户界面

每款型号所对应的 XPower 估计器的电子数据表格是不同的，下面以 Virtex-4 系列芯片的电子表格为例进行介绍。将网站上相应的电子数据表格下载到本地后，用 OFFICE 系列的 Excel 工具打开后，其界面如图 4-63 所示。

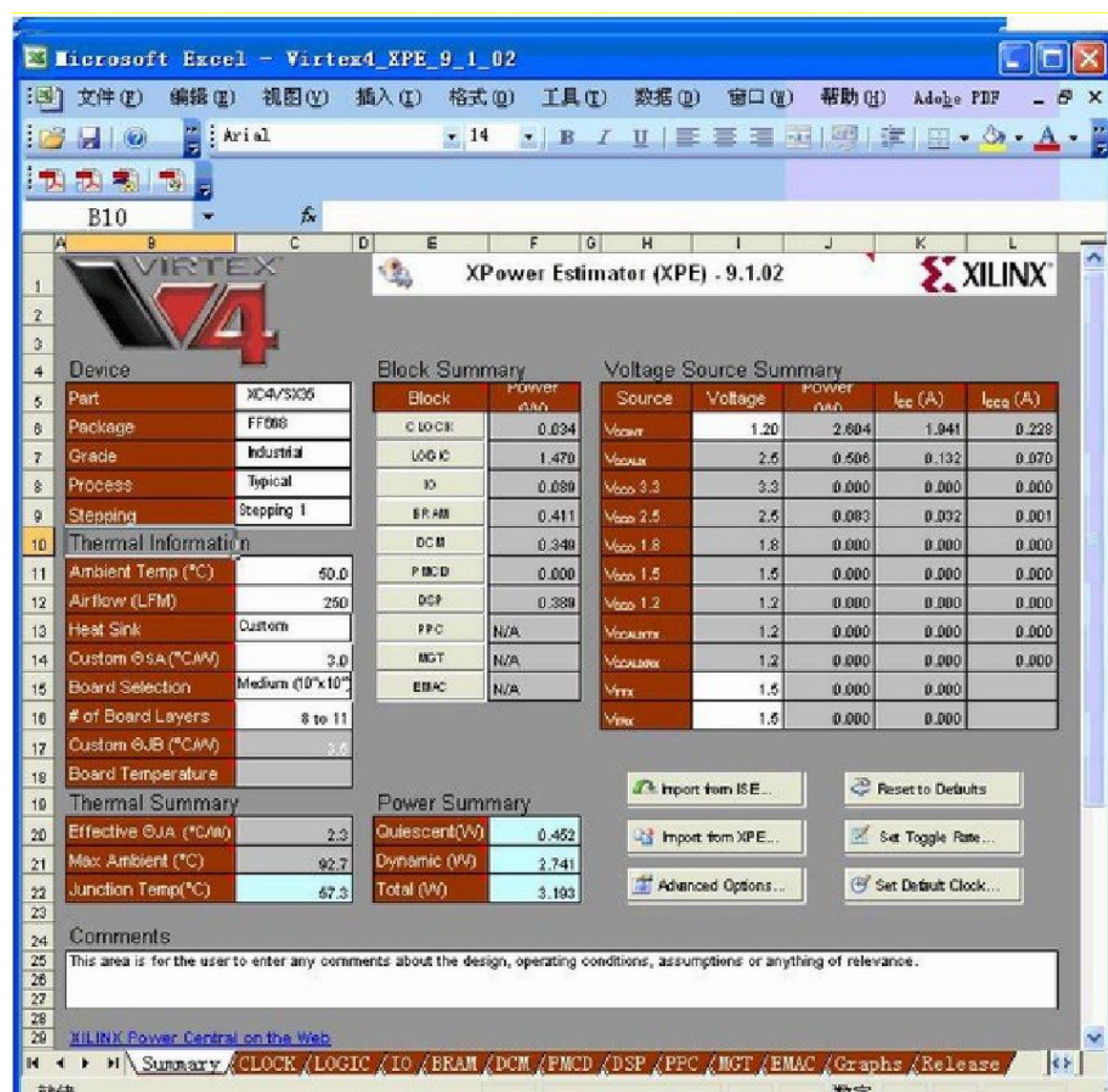


图 4-63 Virtex 系列芯片的 XPower 估计器

从图 4-63 中可以看出，XPower 评估器可分为不同的页面，包括 Summary 页面、CLOCK 页面、LOGIC 页面、IO 页面、BRAM 页面、DCM 页面、PMCD 页面、DSP 页面、PPC 页面、MGT 页面、EMAC 页面、Graphic 页面以及 Release 页面。其中，主界面处输出最终结果，可分为芯片栏、热量信息栏、热量总结栏、注释栏、块总结栏、功率总结栏、电压源总结栏以及交互操作区域组成。其中，不同的颜色表明了用户可进行的操作，详细说明如表 4-2 所列。

表 4-2 电子表格颜色说明

颜色	权限说明	补充
白色 (White)	可写、可编辑	用户可编辑、输入相关参数
灰色 (Grey)	只读、写保护	估计器的计算输出结果，不能编辑或输入数据
青色 (Cgrey)	只读	用于给出功耗估计报告
黄色 (Yellow)	可写、可编辑	专门用于警告提示
红色 (Red)	可写、可编辑	专门用于错误提示

在 **Release** 页面可以查看电子表格的更新版本记录，如本书所用的电子表格版本为 **9.1.02**，更新时间为 **2007-04-23**，其相关信息如图 4-64 所示。

Summary XPower Estimator (XPE) - 9.1.02									
1	Date	Version	Notes						
2	23-Apr-07	9.1.02	PRODUCTION						
3			- Added 4.25 and 5.0 MGT data rates and maximum process numbers						
4			- Added custom board option for thermal modeling						
5			- Fixed issue with running under Excel 2000 and 2002						
6			- Updated clock tree utilization calculations						
7									

图 4-64 Virtex 系列芯片的 XPower 估计器

2) XPower 估计器的使用流程

XPower 估计器的使用方法非常简单，设计人员直接在各个分页面输入相应的资源利用率和时钟频率，然后回到 **Summary** 页面，就可以看到各个电压的电流值，并得到 **FPGA** 的工作功率和工作温度。下面以 **XC4VSX35-12** 芯片的设计为例，介绍 XPower 估计器的使用方法。

例 4-4 使用 XPower 估计器预算 XC4VSX35-12 应用的功耗。

1. 双击打开 XPower 估计器，在 **Summaty** 页面的 **Device** 栏的 **Part** 选项的下拉框中选择“**XC4VSX35**”、**Package** 选项选择“**FF668**”、**Grade** 选项选择“**industrial**”、**Process** 选项选择“**Typical**”、**Steeping** 选项选择“**Steeping1**”，如图 4-65 所示。同时，将 **Thermal Information** 栏选择默认值。

Device	
Part	XC4VSX35
Package	FF668
Grade	Industrial
Process	Typical
Stepping	Stepping 1

图 4-65 XPower 估计器的芯片配置界面

2. 由于 Virtex-4 族的 SX 系列芯片没有 PowerPC、MGT（吉比特收发器）以及 EMAC（以太网接口）控制器，所以只需要设定时钟、逻辑、IO、BRAM、DSP 以及 DCM 即可。XPower 估计器只是粗略预算，所以在输入以上指标，应预留 15%的裕量，即所有值按照真实值的 115%输入。在时钟栏输入所有可能用到的时钟，并给出其扇出（Fanout）大小（一般设定为 10 即可），本例中，输入了 270MHz、120MHz 以及 60MHz 这 3 个频率的全局时钟，如图 4-66 所示。



Summary

Clock Tree Power

Name	Frequency (MHz)	Type	Fanout	Power (W)
clk_270MHz	270.0	Global	10	0.014
clk_120MHz	120.0	Global	10	0.011
clk_60MHz	60.0	Global	10	0.009

图 4-66 XPower 估计器的时钟配置界面

在逻辑页面，输入 a、b、c 三个模块，分别工作于不同的频率，并给出了其大致的逻辑资源和平均扇出，如图 4-67 所示。

Summary		Logic Power						
Module	Clock (MHz)	LUTs	Shift Registers	Select RAMs	FFs	Toggle Rate	Average Fanout	Power (W)
a	270.0	10000	10	10	10000	50.0%	3	1.386
b	120.0	600	1	2	1200	50.0%	3	0.046
c	60.0	1800	4	4	200	50.0%	3	0.036

图 4-67 XPower 估计器的逻辑配置界面

在 IO 页面，输入各个模块的端口数（注意：这里的端口是直接引到顶层模块，和芯片管脚所对应的端口数），选择工作电平标准以及工作频率等参数，本例的配置如图 4-68 所示。

I/O Power														
Module	I/O Standard	Clock (MHz)	Input Pins	Output Pins	BiDir Pins	Bank	Toggle Rate	Output Enable	Output Load (pF)	Data Rate	V _{CCINT} Power (W)	V _{CCAUX} Power (W)	V _{CC0} On-chip Thermal Power (W)	V _{CC0} Supply Current(A)
a	LVCNMOS 2.5V 12mA (Slow)	30.0	64	18	5	2	50.0%	100.0%	0	SDR	0.002	0.001	0.021	0.009
b	LVCNMOS 2.5V 12mA (Slow)	10.0	10	10	0	1	50.0%	100.0%	0	SDR	0.000	0.000	0.003	0.001
c	LVCNMOS 2.5V 12mA (Slow)	60.0	0	0	30	1	50.0%	100.0%	0	SDR	0.001	0.004	0.056	0.022

图 4-68 XPower 估计器的 IO 配置界面

接下来配置 BRAM 页面，需要输入 BRAM 各个端口的位宽、速率、数量以及应用类型（包括 BRAM、FIFO 以及 ECC 等类型），本例的 BRAM 使用状况如图 4-69 所示。

Block RAM Power														
Module	BRAMs	Mode	Toggle Rate	Port A					Port B					Power (W)
				Clock (MHz)	Enable Rate	Bit Width	Write Mode	Write Rate	Clock (MHz)	Enable Rate	Bit Width	Write Mode	Write Rate	
a	70 BRAM		50.0%	270.0	25.0%	18	WRITE_FIRST	50.0%	270.0	25.0%	18	WRITE_FIRST	50.0%	0.398
b	1 BRAM		50.0%	120.0	25.0%	18	WRITE_FIRST	50.0%	120.0	25.0%	18	WRITE_FIRST	50.0%	0.003
Count	10 BRAM		50.0%	60.0	25.0%	18	WRITE_FIRST	50.0%	60.0	25.0%	18	WRITE_FIRST	50.0%	0.013

图 4-69 XPower 估计器的 BRAM 配置界面

在 DCM 的配置页面中，填入所使用的 DCM 时钟的频率即可，本例用到了 270MHz、120MHz 以及 60MHz 这 3 个 DCM 输出，如图 4-70 所示。

DCM Power						
Name	Clock (MHz)	Freq Mode	V _{CCINT} (W)	V _{CCAUX} (W)	V _{CCAUXQ} (W)	
clk_270MHz	270.0	High	0.014	0.153	0.029	
clk_120MHz	120.0	Low	0.006	0.057	0.029	
clk_60MHz	60.0	Low	0.003	0.029	0.029	
Subtotal			0.024	0.239	0.087	
Utilization	37.5%	Total		0.349		

图 4-70 XPower 估计器的 DCM 配置界面

最后需要配置的是 DSP 模块，本例只在 a 模块中使用了 48 个 DSP 模块，工作在 270MHz，因此配置界面如图 4-71 所示。

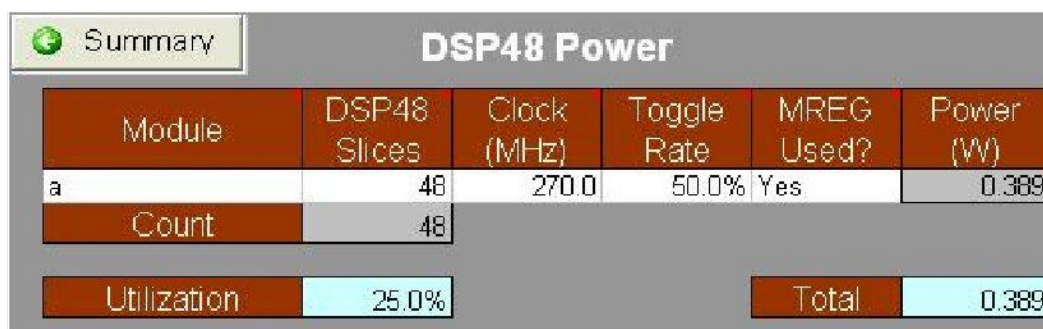


图 4-71 XPower 估计器的 DSP 配置界面

配置完后，点击 **Graphic** 页面，可以看到功耗随设计的逻辑功能、电压、电压过程温度以及工作环境温度的变化曲线，如图 4-72 所示。可以看到，本设计的逻辑所占功耗非常高，而大量的 BRAM 和 DSP 的功耗相对比较低。因此在设计中，应尽量使用芯片内部的硬核组件以降低功耗。

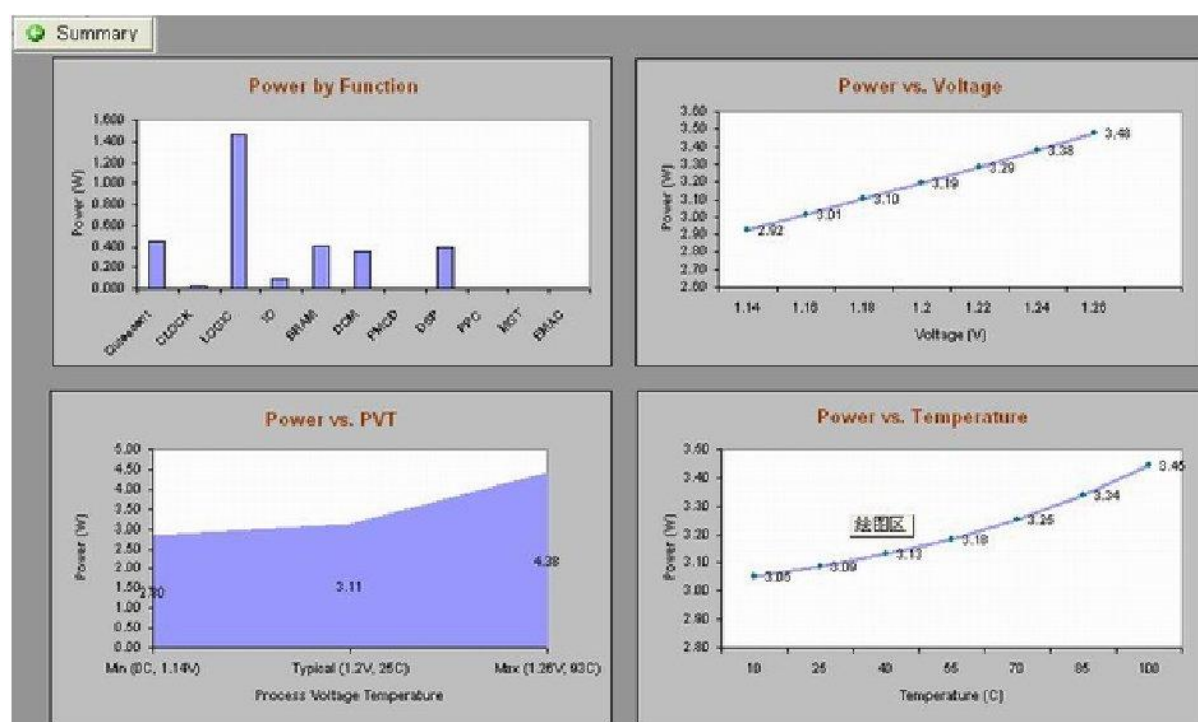


图 4-72 XPower 估计器的图形化分析结果

完成上述过程后，返回到 **Summary** 页面，可以得到所有的功耗汇总结果，以及不同电压的电流大小，为系统的电源模块设计提供大致的参考范围，如图 4-72 所示。总的功耗为 3.193W，1.2V 核电压的工作电流大致为 2.5A（1.941A+2.028A）。

式中，P 表示功耗，单位是 mW；C 表示电容，单位是 F；V 表示电压，单位是 V；E 表示翻转频率，指每个时钟周期的翻转次数；f 表示工作频率，单位是 Hz。在 XPower 中，翻转

频率既可以采用全局默认的翻转频率，也可以通过 VCD 文件获得。另外，XPower 允许手工输入各个设计单元的翻转频率。

在 XPower 功耗分析过程中，主要涉及 NCD 文件、CTX 文件、PCF 文件、VCD 文件和 PWR 文件。其中，NCD 文件是经过实现的 FPGA 设计文件；CTX 文件是经过物理实现（FIT）的 CPLD 设计文件；PCF 文件是物理设计约束文件，该文件包含当前设计的时钟频率、电压等特性参数；PWR 文件是 XPower 的功耗分析报告；VCD 文件是对当前设计进行仿真后生成的文件，该文件包含了每个设计单元的翻转频率。

2. XPower 的用户界面以及使用流程

XPower 是一种设计后工具，用于分析实际器件利用率，并结合实际的适配后（post-fit）仿真数据（VCD 文件格式），给出实际功耗数据。利用 Xpower，您可以在完全不接触芯片的情况下分析设计改变对总功耗的影响。

第 4 节 约束文件的编写

4.4.1 约束文件的基本操作

1. 约束文件的概念

FPGA 设计中的约束文件有 3 类：用户设计文件（.UCF 文件）、网表约束文件（.NCF 文件）以及物理约束文件（.PCF 文件），可以完成时序约束、管脚约束以及区域约束。3 类约束文件的关系为：用户在设计输入阶段编写 UCF 文件，然后 UCF 文件和设计综合后生成 NCF 文件，最后再经过实现后生成 PCF 文件。本节主要介绍 UCF 文件的使用方法。

UCF 文件是 ASC 2 码文件，描述了逻辑设计的约束，可以用文本编辑器和 Xilinx 约束文件编辑器进行编辑。NCF 约束文件的语法和 UCF 文件相同，二者的区别在于：UCF 文件由用户输入，NCF 文件由综合工具自动生成，当二者发生冲突时，以 UCF 文件为准，这是因为 UCF 的优先级最高。PCF 文件可以分为两个部分：一部分是映射产生的物理约束，另一部分是用户输入的约束，同样用户约束输入的优先级最高。一般情况下，用户约束都应在 UCF 文件中完成，不建议直接修改 NCF 文件和 PCF 文件。

2. 创建约束文件

约束文件的后缀是.ucf，所以一般也被称为 UCF 文件。创建约束文件有两种方法，一种是通过新建方式，另一种则是利用过程管理器来完成。

第一种方法：新建一个源文件，在代码类型中选取“Implementation Constrains File”，在“File Name”中输入“one2two_ucf”。单击“Next”按钮进入模块选择对话框，选择模块“one2two”，然后单击“Next”进入下一页，再单击“Finish”按钮完成约束文件的创建。

第二种方法：在工程管理区中，将“Source for”设置为“Synthesis/Implementation”。“Constrains Editor”是一个专用的约束文件编辑器，双击过程管理区中“User Constrains”下的“Create Timing Constrains”就可以打开“Constrains Editor”，其界面如图 4-73 所示：

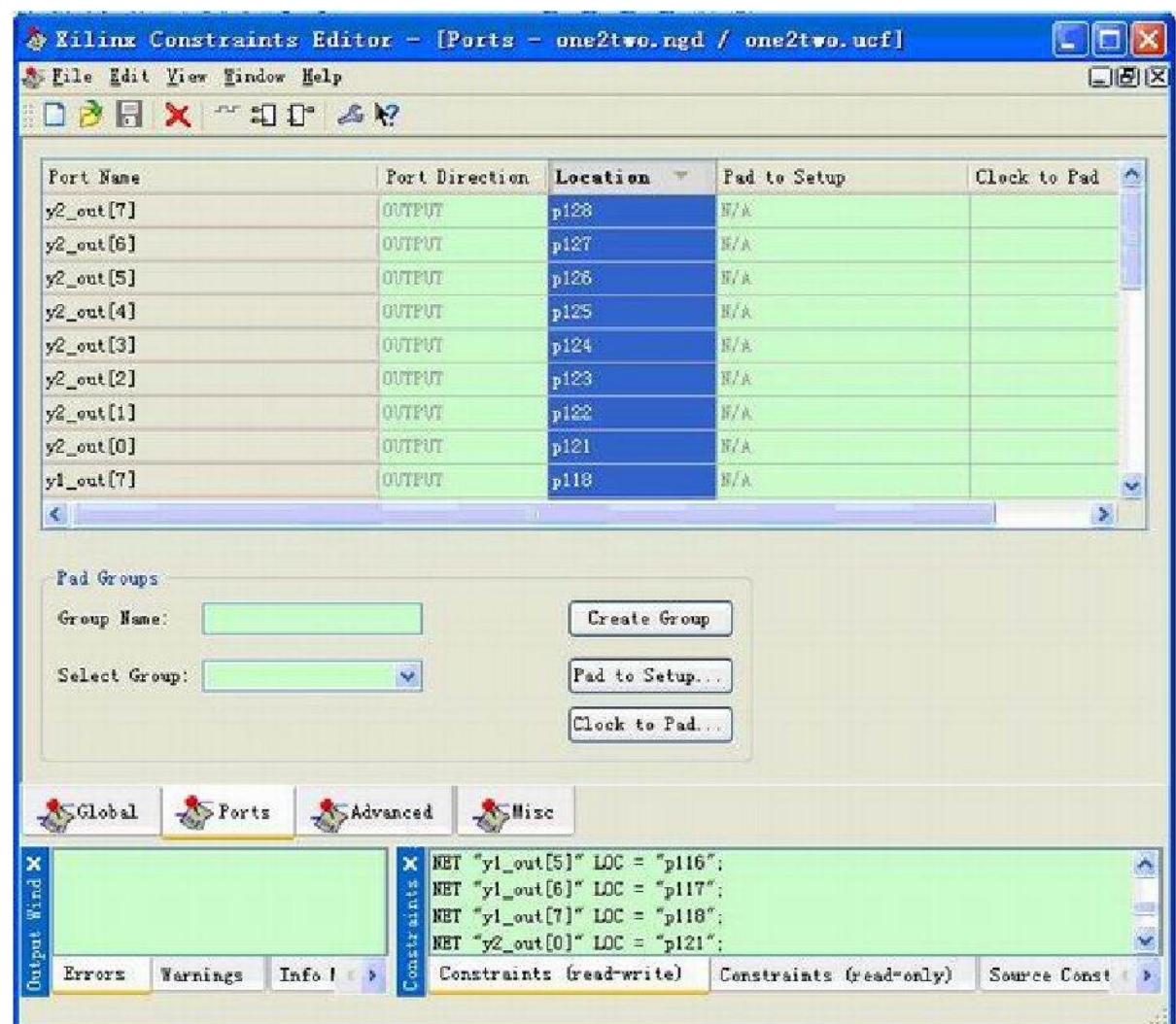


图 4-73 启动 Constrains Editor 引脚约束编辑

在“Ports”选项卡中可以看到，所有的端口都已经罗列出来了，如果要修改端口和 FPGA 管脚的对应关系，只需要在每个端口的“Location”列中填入管脚的编号即可。例如在 UCF 文件中描述管脚分配的语法为：

NET “端口名称” LOC = 引脚编号;

需要注意的是，UCF 文件是大小敏感的，端口名称必须和源代码中的名字一致，且端口名字不能和关键字一样。但是关键字 NET 是不区分大小写的。

3. 编辑约束文件

在工程管理区中，将“Source for”设置为“Synthesis/Implementation”，然后双击过程管理区中“User Constrains”下的“Edit Constraints (Text)”就可以打开约束文件编辑器，如图 4-73 所示，就会新建当前工程的约束文件。

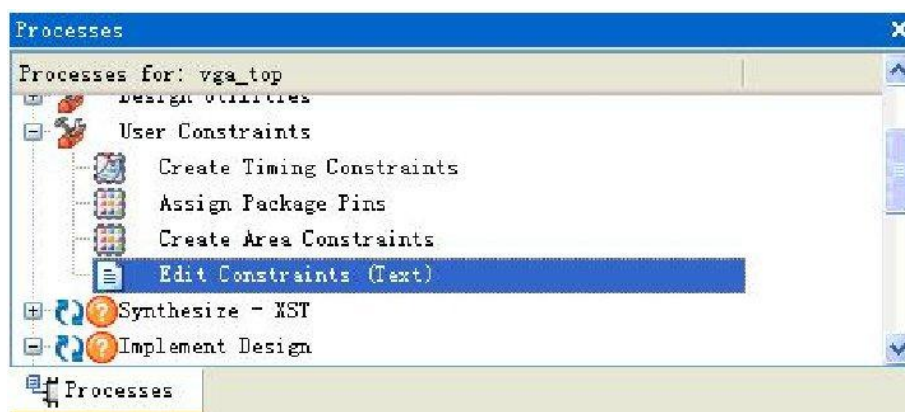


图 4-73 用户约束管理窗口

4.4.2 UCF 文件的语法说明

1. 语法

UCF 文件的语法为：

{NET|INST|PIN} "signal_name" Attribute;

其中，“signal_name”是指所约束对象的名字，包含了对象所在层次的描述；“Attribute”为约束的具体描述；语句必须以分号“;”结束。可以用“#”或“/* */”添加注释。需要注意的是：

UCF 文件是大小写敏感的，信号名必须和设计中保持大小写一致，但约束的关键字可以是大写、小写甚至大小写混合。例如：

```
NET "CLK" LOC = P30;
```

"CLK"就是所约束信号名，LOC = P30；是约束具体的含义，将 CLK 信号分配到 FPGA 的 P30 管脚上。

对于所有的约束文件，使用与约束关键字或设计环境保留字相同的信号名会产生错误信息，除非将其用" "括起来，因此在输入约束文件时，最好用" "将所有的信号名括起来。

2. 通配符

在 UCF 文件中，通配符指的是"*"和"?"。"*"可以代表任何字符串以及空，"?"则代表一个字符。在编辑约束文件时，使用通配符可以快速选择一组信号，当然这些信号都要包含部分共有的字符串。例如：

```
NET "*CLK?" FAST;
```

将包含"CLK"字符并以一个字符结尾的所有信号，并提高了其速率。

在位置约束中，可以在行号和列号中使用通配符。例如：

```
INST "/CLK_logic/*" LOC = CLB_r*c7;
```

把 CLK_logic 层次中所有的实例放在第 7 列的 CLB 中。

3. 定义设计层次

在 UCF 文件中，通过通配符*可以指定信号的设计层次。其语法规则为：

* 遍历所有层次

Level1/* 遍历 level1 及以下层次中的模块

Level1/*/ 遍历 level1 种的模块，但不遍历更低层的模块

例 4-5 根据图 4-75 所示的结构，使用通配符遍历表 4-3 所要求的各个模块。

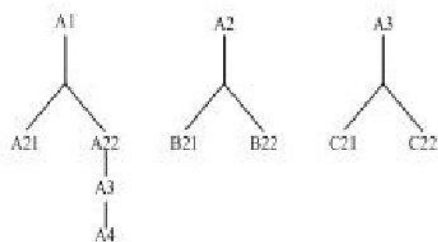


图 4-75 层次模块示意图

表 4-3 要求遍历的符号列表

要求遍历的符号	相应的约束语句
所有符号	INST * 或 INST/*
A1, B1, C1	INST/*/
A21, A22	INST A1/*/
A3	INST A1/*/*/
A3, A4	INST A1/*/*
A22, B22, C22	INST /*/*22/

4.4.3 管脚和区域约束语法

LOC 约束是 FPGA 设计中最基本的布局约束和综合约束，能够定义基本设计单元在 FPGA 芯片中的位置，可实现绝对定位、范围定位以及区域定位。此外，LOC 还能将一组基本单元约束在特定区域之中。LOC 语句既可以书写在约束文件中，也可以直接添加到设计文件中。换句话说，ISE 中的 FPGA 底层工具编辑器（FPGA Editor）、布局规划器（Floorplanner）和引脚和区域约束编辑器的主要功能都可以通过 LOC 语句完成。

• LOC 语句语法

基本的 LOC 语法为：

```
INST "instance_name " LOC = location;
```

其中“location”可以是 FPGA 芯片中任一或多个合法位置。如果为多个定位，需要用逗号“,”隔开，如下所示：

```
LOC = location1,location2,...,locationx;
```

目前，还不支持将多个逻辑置于同一位置以及将多个逻辑至于多个位置上。需要说明的是，

多位置约束并不是将设计定位到所有的位置上，而是在布局布线过程中，布局器任意挑选其中的一个作为最终的布局位置。

范围定位的语法为：

INST "instance_name" LOC=location:location [SOFT];

常用的 LOC 定位语句如表 4-4 所列。

表 4-4 常用的 LOC 定位语句

LOC 语句	说明	功能表述
INST "instance_name" LOC=P12;	单定位语句	将 I/O 管脚 P12 分配给实例信号
INST "instance_name" LOC=CLB_R3C5;	单定位语句	将逻辑置于坐标为行 3、列 5 的 CLB 中的任一个 SLICE
INST "instance_name" LOC=SLICE_X3Y2;	单定位语句	将逻辑置于 Slice xy 网格中的 (3 , 2) 位置上
INST "instance_name" LOC=TBUF_R1C2.*;	单定位语句	将逻辑至于 1 行、2 列位置的两个 TBUF 中
INST "instance_name" LOC=MULT18X18_X0Y6;	单定位语句	将乘法器逻辑置于乘法器 xy 网格中 (0 , 6) 位置上的 MULT18X18 乘法器中
INST "instance_name" LOC=clb_r4c5.s1,clb_r4c6.*;	多定位语句	将触发器置于 4 行 5 列 CLB 和 4 行 6 列的 CLB 中最右端的 Slice 中
INST "instance_name" LOC=SLICE_X2Y10,SLICE_X1Y10;	多定位语句	将逻辑置于 Slice xy 网格中 (2 , 10) 或 (1 , 10) 位置上的 Slice 中
INST "instance_name" LOC=SLICE_X3Y5:SLICE_X5Y20;	范围定位语句	将逻辑至于 4 行 4 列 CLB 左上角的任一 Silce 上
INST "instance_name" LOC=SLICE_X3Y5:SLICE_X5Y20;	范围定位语句	将逻辑至于 Slice xy 网格中，由 (3 , 5) (5 , 20) 两点为对角线的矩形中的任一 Slice 中

使用 LOC 完成端口定义时，其语法如下：

NET "Top_Module_PORT" LOC = "Chip_Port";

其中，“Top_Module_PORT”为用户设计中顶层模块的信号端口，“Chip_Port”为 FPGA 芯片的管脚名。

LOC 语句中是存在优先级的，当同时指定 LOC 端口和其端口连线时，对其连线约束的优先级是最高的。例如，在图 4-76 中，LOC=11 的优先级高于 LOC=38。

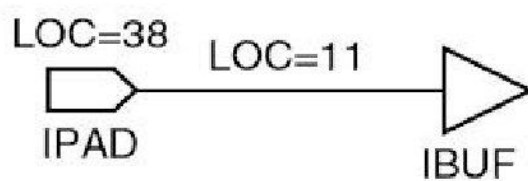


图 4-76 LOC 优先级示意图

2. LOC 属性说明

LOC 语句通过加载不同的属性可以约束管脚位置、CLB、Slice、TBUF、块 RAM、硬核乘法器、全局时钟、数字锁相环（DLL）以及 DCM 模块等资源，基本涵盖了 FPGA 芯片中所有类型的资源。由此可见，LOC 语句功能十分强大，表 4-5 列出了 LOC 的常用属性。

表 4-5 LOC 语句常用属性列表

约束类型	可用属性示例	属性含义
IO 管脚约束	P12	将信号置于由芯片引脚号定位的端口上
	A12	将信号置于由芯片引脚阵列号定位的端口上
	B, L, T, R	将信号定位到芯片特定边界中（从物理位置上划分的上、下、左、右4部分）的端口上。
	LB, RB, LT, RT, BR, TR, BL, TL	将信号定位到芯片特定边界上的一半（从物理位置上划分的左上、上右、下左、下右、左上、坐下、右上以及右下8部分）位置中的端口上。
	Bank0, Bank1, Bank2, Bank3, Bank4, Bank5, Bank6, Bank7	将信号置于特定管脚分组中的端口上。
CLBs	CLB_R4C3 (or S0 or S1)	指定确定位置的CLB来实现逻辑
	CLB_R6C8 S0 (or S1)	指定确定位置的CLB中的Slice来实现逻辑
Slice	SLICE_X22Y3	直接通过Slice坐标来指定确定位置的Slice来实现逻辑
TBUF	TBUF_R6C7 (or 0 or 1)	指定确定位置的TBUF来实现逻辑
	TBUF_X6Y7	
块 RAM	RAMB4_R3C1	指定确定位置的块RAM来实现逻辑
	RAMB16_X2Y56	
硬核乘法器	MULT18X18_X55Y82	指定确定位置的硬核乘法器来实现逻辑
全局时钟	GCLKBUF0 (or 1, 2, or 3)	指定确定位置的全局时钟缓存器来实现逻辑
	GCLKPAD0 (or 1, 2, or 3)	指定确定位置的全局时钟端口来实现逻辑
DLL	DLL0P(or S) (or 1, 2, or 3)	使用确定位置的DLL来实现逻辑
DCM	DCM_X0Y0	使用确定位置的DCM模块来实现逻辑

4.4.4 使用 PACE 完成管脚约束

ISE 中内嵌了图形化的引脚和区域约束编辑器 PACE（Pinout and Area Constraints Editor）可以将设计管脚映射到器件中，并对逻辑区块进行平面布置，方便地完成管脚约束和区域约束。在 PACE 中，可将管脚拖放到器件的显示图形上，通过容易识别的彩色编码将管脚进行逻辑分组，定义 I/O 标准和库，分配和放置微分 I/O 等。和使用约束文件相比，在中、大规模 FPGA 的开发中，能大大简化管脚约束流程。

通过检查定义的 HDL 层级和核对逻辑区块与预计的门尺寸的关系，PACE 可以实现区块映射，使区块定义变得快速、准确和容易。在 HDL 编码开始之前，就可以使用 PACE 分配管脚，

然后写 HDL 开始模板，供你编辑。可以通过标准 CSV 文件，将管脚信息导出或导入到 PCB 布局编辑器中，这大大简化了设计计划的编制。

1. PACE 用户界面

PACE 的启动方法有两种：一种是单独启动 PACE，直接点击“开始”“程序”“Xilinx ISE 9.1i”“Accessories”“PACE”即可启动；另一种是在工程经过布局布线后，在过程管理区双击“User Constraints”“Assign Package Pins”来打开 PACE，并自动加载当前工程。需要注意的是，在启动 PACE 之前，要确保相应的设计中存在 UCF 文件，否则会提示错误。这是因为，通过 PACE 完成的操作，最终的依然要写入到相应的 UCF 文件中。典型的 PACE 用户界面如图 4-77 所示。

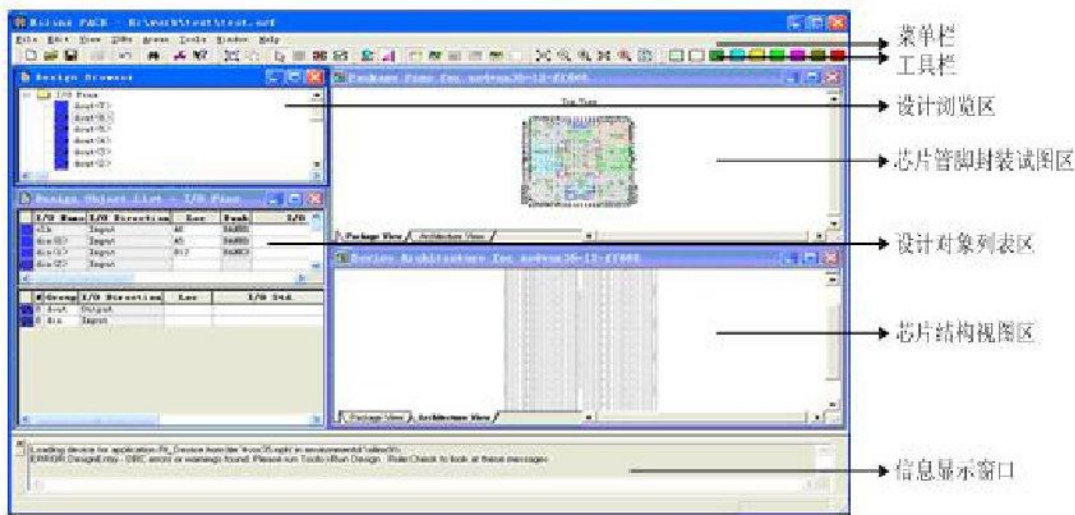


图 4-77 PACE 的用户界面

PACE 的用户界面主要由菜单栏、工具栏、设计浏览区、设计对象列表区、芯片管脚封装视图图、芯片结构视图区以及信息显示窗口组成。

2. 使用 PACE 添加 I/O 约束

在分配管脚之前，首先需要确定芯片是否选择正确，可通过点击菜单“IOB”中的“”命令来看所选芯片型号，弹出的对话框如图 M 所示。如芯片型号错误，可重新选择。



图 4-78 所选芯片的型号

其次，点击菜单“IOB”中的“”命令来禁止不可用的输入输出管脚，弹出的对话框如图 M 所示。通过该菜单可完成所有复用管脚的控制，包括芯片配置管脚以及参考电压管脚。



图 4-79 部分输入输出管脚的控制

再次，可将信号分组或组成总线模式来加快速管脚分配的速度。一般来讲，PACE 会自动将信号进行分组。此外，设计人员也可以手动添加信号分组，其方法如下：在设计信号列表区，按住“Ctrl”键，选取需要组合的多个信号，然后点击菜单“Edit”中的“Group”命令，即可将所选信号合并，并在信号列表区中的分组显示区显示出来，如图 M 所示。



图 4-80 PACE 信号合并界面

对于新添加的分组信号，PACE 会以“UserGroupN”命名，其中 N 为添加的序号，用户可直接在“Group”列的对应表格中重新命名。对应信号“#”列的表格中的数字为分组或总线中的信号个数。

最后是分配管脚。在 PACE 中有两种方法可完成管脚分配，其一就是直接将设计浏览区中“I/O Pins”目录下的信号或总线直接拖到芯片管脚封装试图区中；另一种方法是在设计信号列表区中，选中相应的信号，直接在“LOC”列所对应的表格中敲入位置。分配完毕后，点击工具栏中的“保存”按钮即可。

此外，PACE 中“IObS”菜单下的“Show Differential Pairs”命令，可在芯片管脚封装视图区列出所有的差分对，如图 M 所示，每一对差分对都通过短线连接起来。

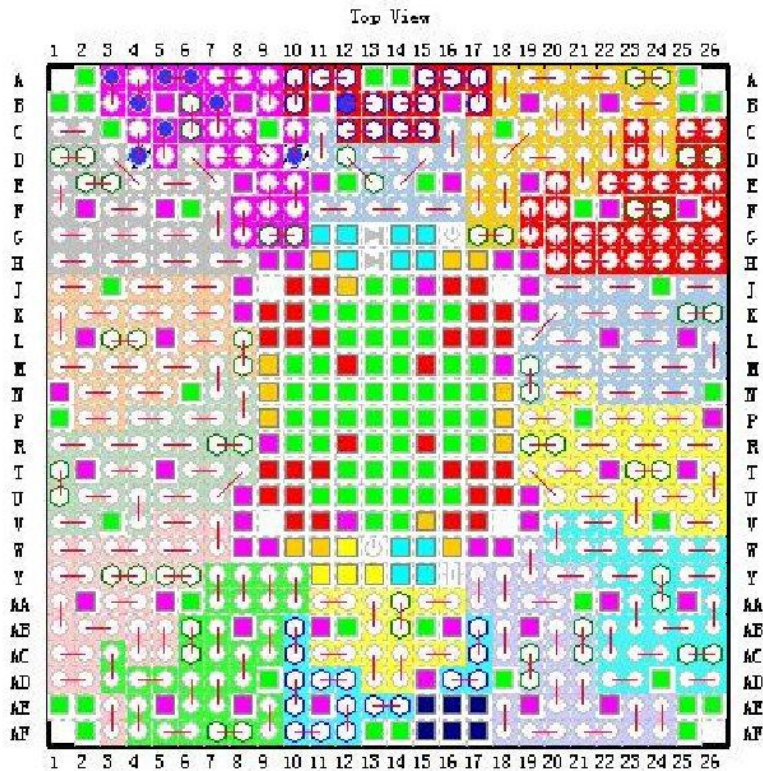


图 4-81 PACE 差分对示意图

3. 使用 PACE 添加区域约束


区域约束的主要目的是关联耦合逻辑，减少后续布线压力；其次是加大资源利用率，距离近的信号延迟不一定就小，信号线上延迟主要是来自线与线之间的转接（如 LUT，switch-box）。由于 FPGA 内部连接的结构是纵横两向的，斜向的连接延迟会大于纵横方向上最大跨度连接。所以，在做位置约束时尽量避免斜向；而区域约束要松，如果没有资源上的顾虑，约束面积建议为所需的 3 倍以上。需要注意的是，区域约束对时序的改善贡献很小，紧的约束甚至有恶化时序的可能。

通过 PACE 软件，可将设计中的所有逻辑资源，包括全局时钟缓冲器、硬核乘加器、块 RAM、硬核处理器、高速收发器以及数字时钟管理模块等模块放入器件架构（Device Architecture）的任何位置。下面通过实例介绍如何使用 PACE 完成区域约束。

例 4-6 使用 PACE 完成设计的区域约束。

（1）通过 ISE 打开设计工程以及其中的 PACE，在设计浏览区选中“Logic”文件夹，单击右键，选择“Object Properties”命令，则会显示所用的资源，包括触发器、进位符、DCM 以及

BUFG 等资源。

(2) 在 PACE 工具栏单击“”图标 (Assign Area constraint Mode)，用鼠标在器件结构窗口划出用于布局的曲域，如图 4-82 所示。

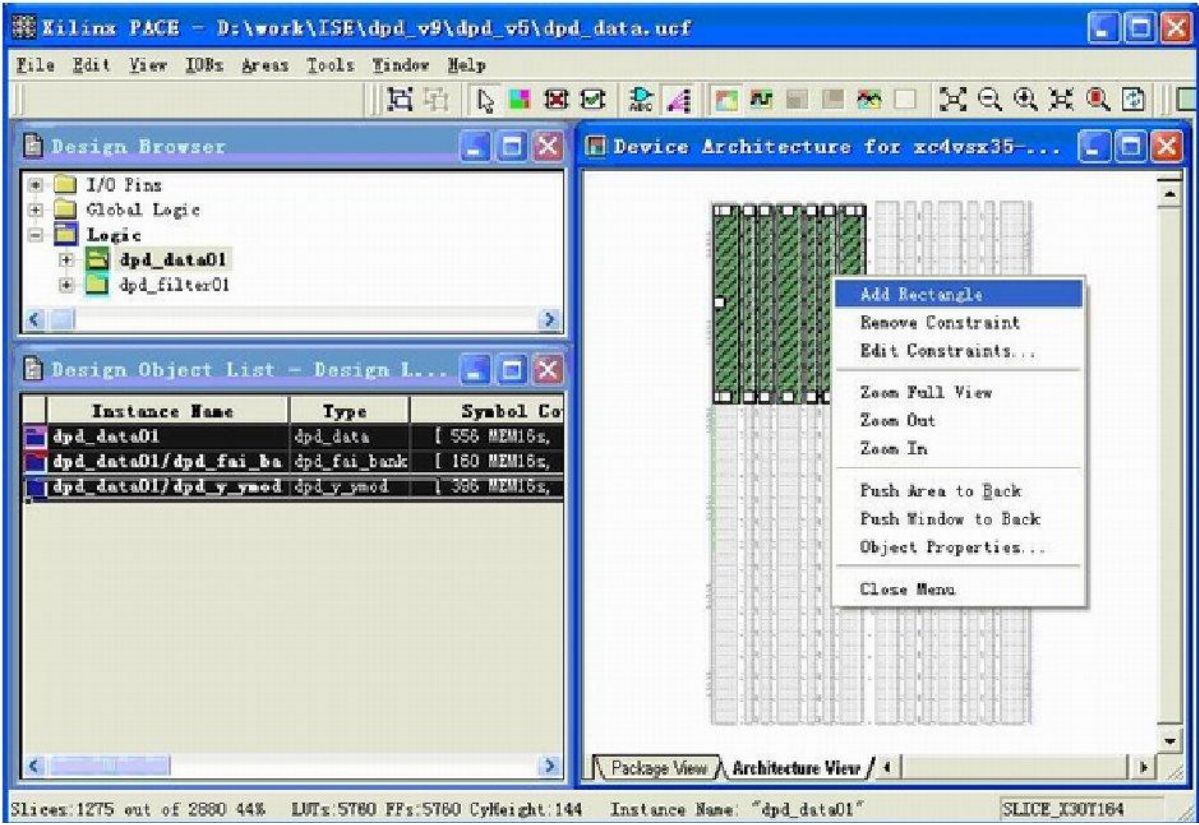


图 4-82 利用 PACE 划定约束区域

完成后区域布局划分后，保存设计，用户即可在 ISE 的 UCF 文件中查阅相应的区域约束文件，附加的区域约束如图 4-83 所示。



```
#PACE: Start of PACE Area Constraints
AREA_GROUP "AG_dpd_data01" RANGE = SLICE_X0Y189:SLICE_X39Y118 ;
INST "dpd_data01/dpd_fai_bank01" AREA_GROUP = "AG_dpd_data01" ;
INST "dpd_data01/dpd_y_ymod01" AREA_GROUP = "AG_dpd_data01" ;

#PACE: Start of PACE Prohibit Constraints

#PACE: End of Constraints generated by PACE
```

图 4-83 完成约束区域后所添加的约束文件

(3) 对于复杂图形的区域，可通过添加多个长方形来完成。首先选中已经规划的区域，然后单击右键，添加新的长方形，依次下去，直到满足要求即可。如果需要修改，双击选中某个长方形，点击右键，选择“Remove Constraint”命令，可删除相应的区域。

(4) 单击工具栏的“ ”图标，可以在器件结构中划分出禁止布局布线的区域。单击“ ”图标，可在禁止布局布线区域重新划定能用于布局布线的区域。

4. 使用 PACE 完成时序分析

目前，FPGA 的工作频率已达到数百兆以上，I/O 端口的数据速率已达到数十吉赫兹，因此在高速设计中对管脚、逻辑资源的布局就显得特别重要，在低速设计中则可以忽略芯片内部的布局布线延迟。PACE 可根据芯片尺寸、型号以及设计的约束，自动给出管脚和逻辑之间、逻辑和逻辑之间的信号延迟报告，该类延迟一般在 ps 数量级上，但对于数百兆的高速信号来讲，已经是非常宝贵的时隙裕量。PACE 软件会自动考虑输入、输出信号的抖动，将高速输出信号放在延时最小的管脚上。特别是对于时钟信号，会附加最优的布局和处理，提高同步设计性能。

一般来讲，利用 PACE 完成 FPGA 引脚时序分析的 3 个步骤如下：

- (1) 打开 PACE 软件，选择“IOB”菜单下面的“Show Flight Times”命令，启动时延分析功能。
- (2) 选择“Tools”菜单下面的“Display Overlay”命令，则会弹出延迟对话框，用不同的颜色表明不同的时延，如图 4-84 所示。该窗口只是信息输出窗口，不能操作，点击“OK”按钮即可。

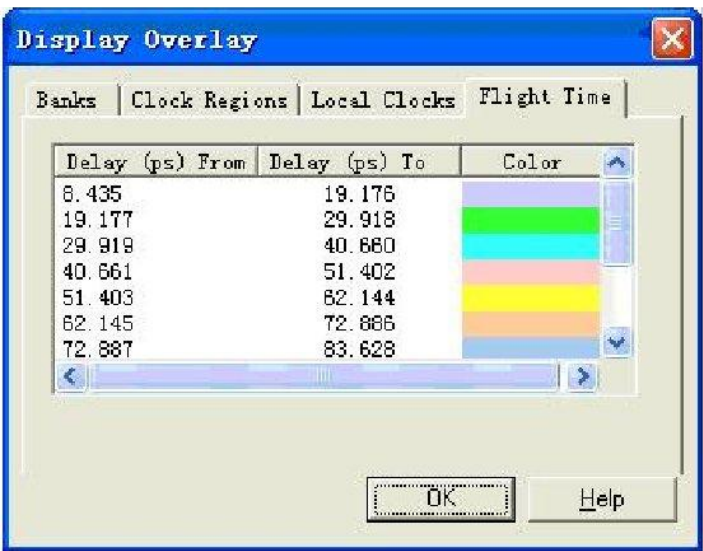


图 4-84 FPGA 引脚之间的传输延迟分类列表

然后在图 4-85 中的芯片管脚封装测试区可以看到，不同延迟的管脚用不同颜色标注，用户可对关键的输入、输出信号进行重新分配，提高设计性能。细心的读者会发现，越处于芯片外围的引脚，其时延越大。

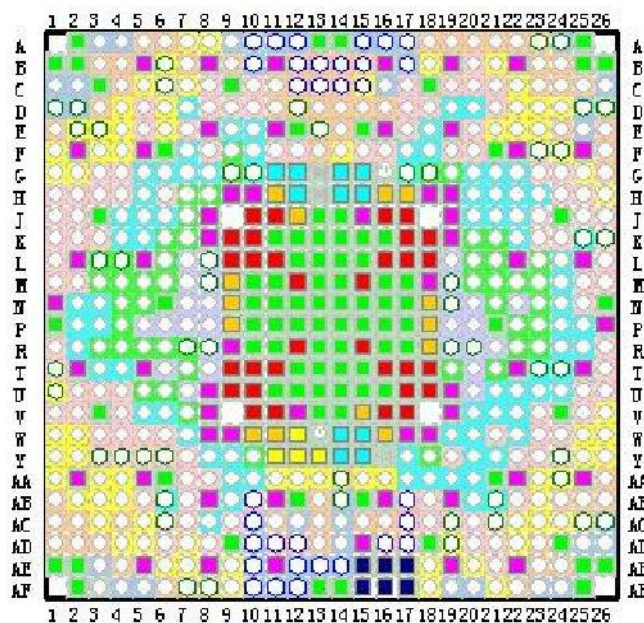


图 4-85 FPGA 引脚传输延迟分布示意图

(3) 对于多时钟设计，用户需要了解芯片的时钟分区。选择“IOB”菜单下面的“Show Clock Regions”命令，则在芯片架构视图区用不同的颜色显示 FPGA 芯片内部不同的时钟分区，如图 4-86 所示。

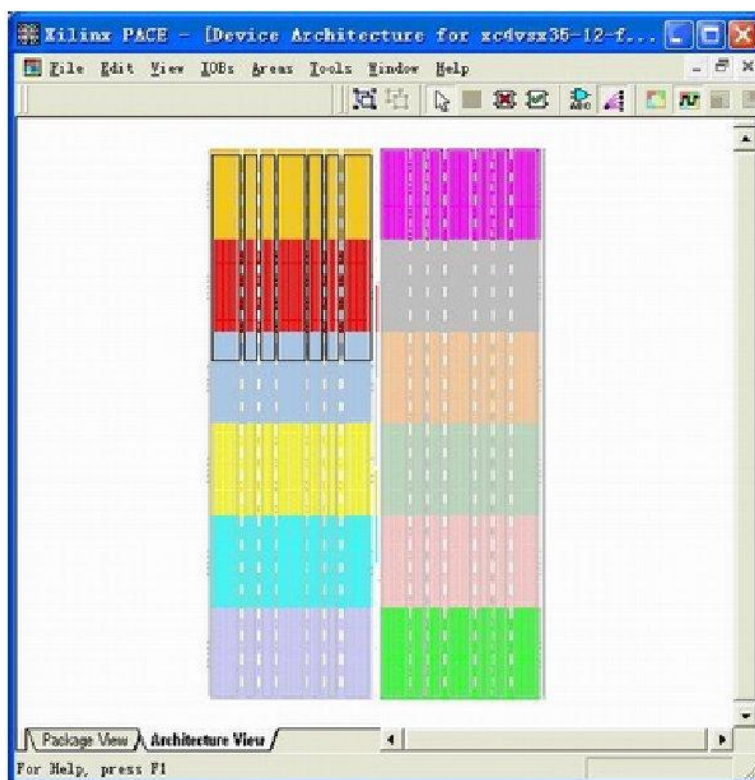


图 4-86 FPGA 时钟分区示意图

此时，设计者更关心时钟和时钟区域的对应关系，点击选择“Tools”菜单下面的“Display Overlay”命令，并选择“Clock Regions”页面，即可直观得到该关系，如图 4-87 所示。

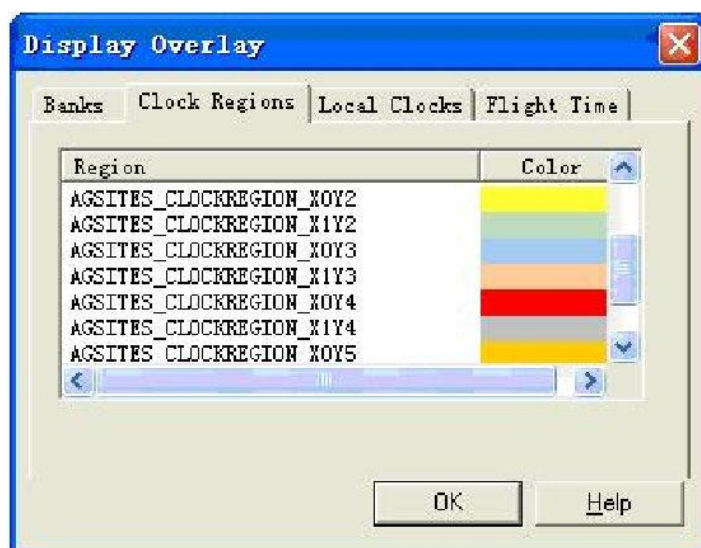


图 4-87 FPGA 时钟与时钟区域对应示意图

5. 使用 PACE 完成 DRC 分析

不同逻辑区域所允许的时钟数量是有限的，若分配到该区域的设计超过了该时钟上限，则会导致设计错误。PACE 提供了时钟分析工具来帮助用户检查此类错误，通过点击“Tools”菜单

下的“Clock Analysis”命令来分析，如果时钟信号超过设定值，则会在其“Regions Per Clock”页面中以“*”号标出。

设计规则是电路或者芯片在版图设计中所必须遵循和满足的各种规定和要求，如果不能满足，则生产出来的芯片将可能无法正常工作。DRC（Design Rule Check）即设计规则检查，就是根据设计规则所规定的各掩膜图形的最小尺寸、最小间距等几何参数，对设计进行检查，找出不满足规则的偏差和错误，为用户修正设计提供依据。目前，FPGA 芯片的管脚越来越多，因此检查时钟管脚分配、IO 端口输出电平标准与相应的 IO 电压是否一致、核电压以及辅助电压是否正确成为任务繁重且容易出错的地方。PACE 提供的 DRC 检查可自动完成上述核查，选择“Tools”菜单下的“Run Design Rule Check（DRC）”命令，即可得到设计的 DRC 结果，如图 4-88 所示。



图 4-88 PACE 的 DRC 检查结果示意图

第 5 节 ISE 与第三方软件

4.5.1 Synplify Pro 软件的使用

在 FPGA 设计中，许多设计人员都习惯于使用综合工具 Synplify Pro。虽然 ISE 软件可以不依赖于任何第三方 EDA 软件完成整个设计，但 Synplify Pro 软件有综合性能高以及综合速度快等特点，无论在物理面积上还是工作频率都能达到较理想的效果。因此如何在 ISE 中调用 Synplify Pro 综合工具，并进行无缝的设计连接仍然是设计人员需要解决的一个设计流程问题。

1. Synplify Pro 综合软件的安装

下面介绍 Synplify Pro 的安装步骤。运行安装程序，欢迎界面过后，将出现如图 4-89 所示

的安装选择界面，可以根据自己的需要选择相应的组件。然后按照默认选项继续即可完成安装。

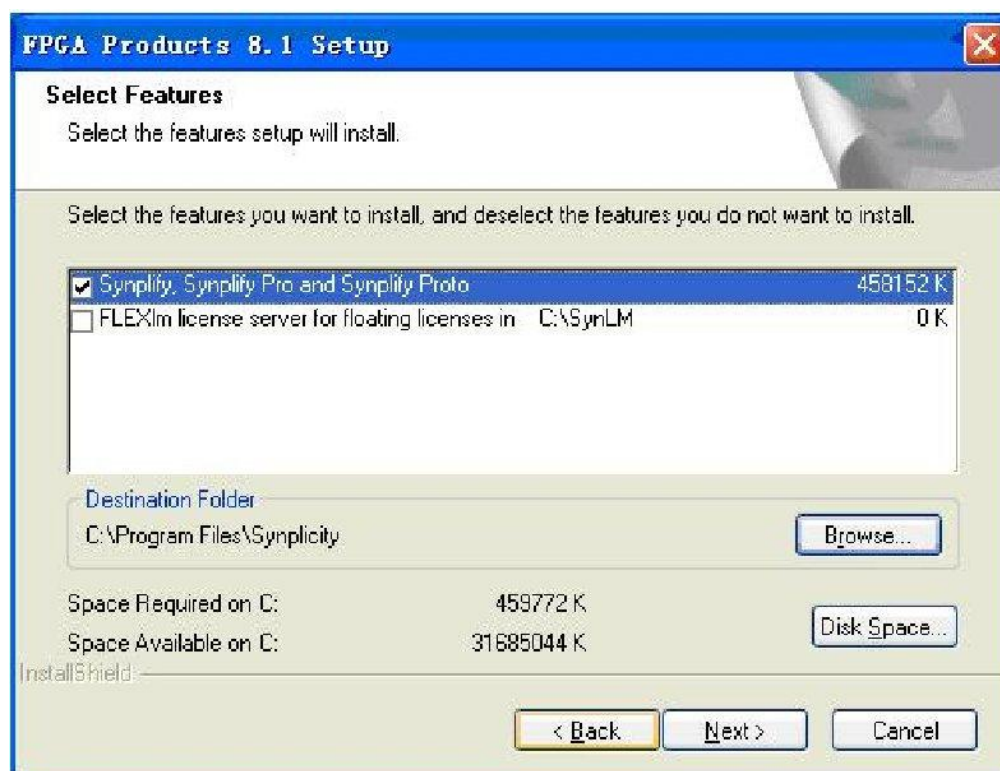


图 4-89 Synplify 的安装选择界面

在 Synplify 安装完后，还需要安装 Identify。在开始 程序 Synplify 菜单栏中会出现“Identify 211 Installation”，双击即开始安装，一般来讲，可以按照默认选项继续，直至安装完毕。安装完之后需要添加授权的 License 文件，才能正常使用。

2. 关联 ISE 和 Synplify Pro

完成了 Synplify Pro 安装后，需要将其和 ISE 软件关联后才能使用 Synplify Pro 进行综合。运行 ISE 软件，在主界面中选择“Edit|Preference”菜单项，进行“Reference”设定如图 4-90 所示。在弹出的 Preference 对话框中选择“Integrated Tools”选项卡。该选项卡用于设定与 ISE 集成的软件的路径，第三项的 Synplify Pro 就用于设定 Synplify Pro 仿真软件的路径，如图 4-91 所示。



图 4-90 选择 Preference 菜单



图 4-91 ISE 集成工具设定页面

项

单击 Synplify Pro 文本框后面的按钮，会弹出一个文件选择对话框，选择 Synplify Pro 安装路径下 bin 目录下的“synplify_pro.exe”文件即可。

注意：在“Integrated Tools”选项卡中还可以看到其他几个可以和 ISE 进行无缝链接的第三方软件，如 ModelSim、synplifyLeonardoSpectrum、Chip Scope Analyzer 等软件。

3. Synplify Pro 的使用方法简介

Synthesis 简单地说就是将 HDL 代码转化为门级网表的过程，其对电路的综合包括以下 3 个步骤：首先，HDL compilation 把 HDL 的描述编译成已知的结构元素；其次，运用一些算法，对设计进行面积优化和减小时延。在没有目标库的情况下，Synplify 只能执行一些最基本的优化措施；最后，将设计映射到指定厂家的特定器件上，并执行一些附加的优化措施，包括根据由器件供应商提供的专用约束进行优化。工程文件以*.prj 作为扩展名，以 tcl 的格式保留了以下信息：设计文件、约束文件、综合时开关选项的设置情况等。

1) Synplify Pro 用户界面介绍

Synplify Pro 是标准的 windows 应用程序，所有功能均可以通过菜单选择来实现。下面按照图 4-92 中数字所标示的次序，对其界面作简要介绍。图中 1 表示 Synplify 的主要工作窗口，在这个窗口中可以详细显示设计者所创建工程的详细信息，包括工程的源文件，综合后的各种结果文件。同时如果综合完成后，每个源文件有多少错误或者警告都会在这个窗口显示出来。图中 2 表示 TCL 窗口，在这个窗口中设计者可以通过 TCL 命令而不是菜单来完成相应的功能。图中 3 表示观察窗口，在这里可以观察设计被综合后的一些特性，比如最高工作频率等。图中 4 是状态窗口，它表示现在 Synplify 所处的状态，比如下图表示 Synplify 处于闲置状态，在综合过程中会显示编译状态、映射状态等等。图中 5 所示的一些复选框，可以对将要综合的设计的一些特性进行设置。Synplify 可以根据这些设置对设计进行相应的优化工

作。图中 6 是运行按钮，当一个工程加入之后，按这个 RUN 按钮，Synplify 就会对工程进行综合。图中 7 所示的是 Synplify 的工具栏。

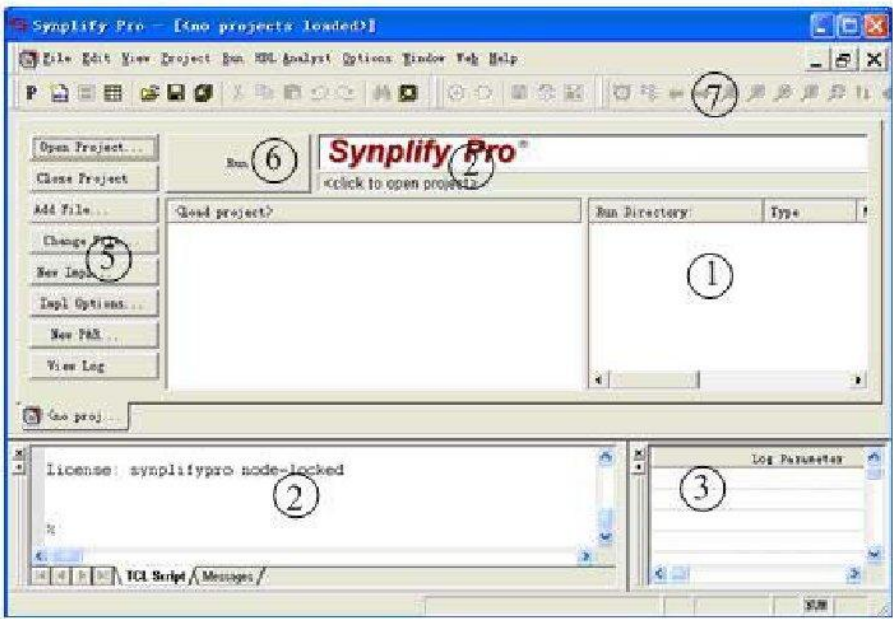


图 4-92 Synplify Pro 综合工具示意图

2) 建立工程、添加源文件

建立工程首先需要打开 Synplify Pro。点击“开始”菜单，依次选择“程序 Synplify Pro”，启动 Synplify Pro。在工程窗口中包含了以下内容：源文件信息、结果文件信息和目标器件信息。

缺省情况下，当 Synplify 启动时将自动建立一个新工程。这时，可以选择将工程，以新名字保存。如果结束了一个工程的操作，想新建一个工程，则可以选择“FILE NEW”；然后选择工程文件，就可以建立一个新的工程。这项操作也可以通过工具条来进行，单击工具条的 P 图标，则在弹出对话框选择工程文件即可。

新建工程之后，需要将源文件添加进来。点击“ADD FILE”按钮。添加源文件和约束文件。Synplify Pro 把最后编译的“module/entity and the architecture”作为顶层设计，所以需要把顶层设计文件用左键拖拉到源文件菜单的末尾处或者点击“Impl Options”按钮，在 Verilog 属性页中设置顶层模块的名称。

3) 工程属性设置

添加完源文件后需要设置工程属性，点击“Impl option”按钮出现属性页对话框，如图 4-51 所示。下面介绍常用的芯片设置、综合选项、约束设置以及实现结果选项等参数的配置。

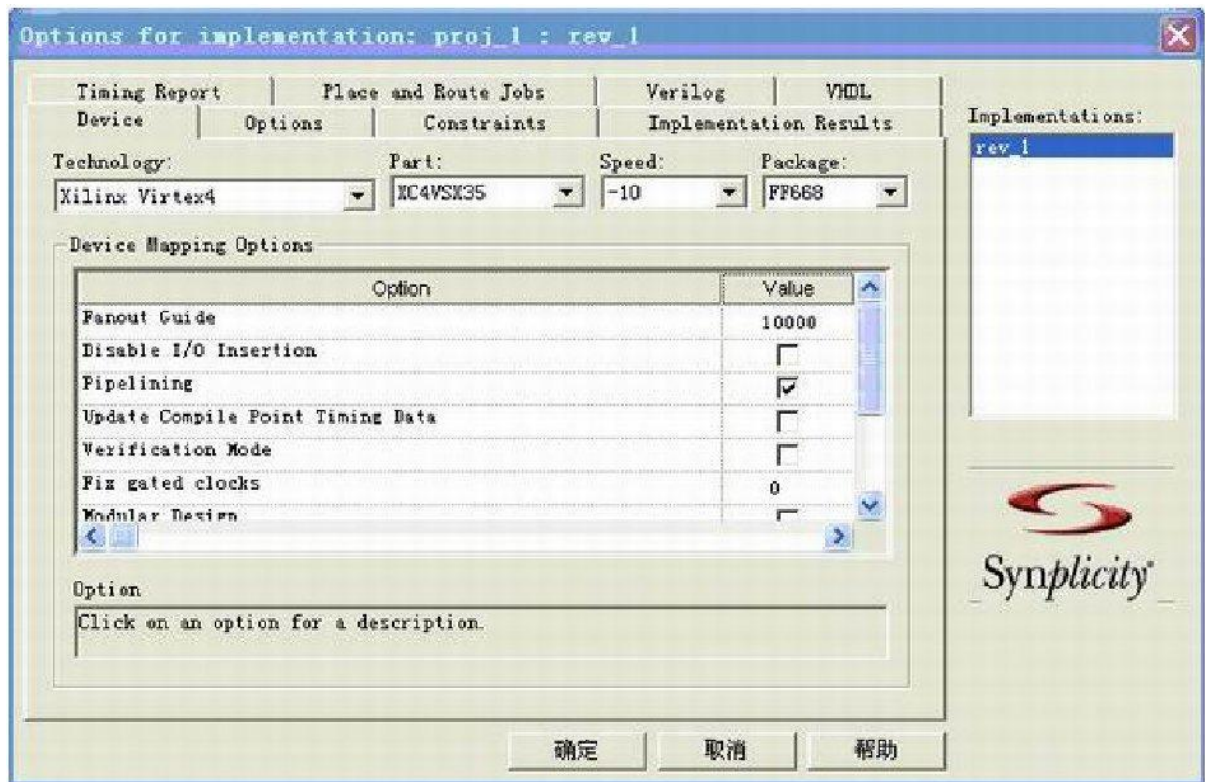


图 4-93 设置器件属性页

- 首先，设置 FPGA 芯片信息。打开“Device”属性页，分别设置器件厂家器件型号、速度级别和封装信息。根据设计的速度和面积要求。可以设置最大扇出系数，缺省是 10000。根据该工程所属模块是否和片外有信号联系，选中或者不选中“Disable I/O insert”，如果选中该选项，则 Synplify Pro 不会为输入输出信号加缓冲，缺省为不选。
- 设置通用综合选项。点击“options”属性页，选中“Symbolic FSM Compiler”，Synplify Pro 会在综合过程中启动有限状态机编译器，对设计中的状态机进行优化。选中“Resource Sharing”选项，则启动资源共享；设置了资源共享后，设计的最高工作频率会低于不选中的情况，但是资源会节约很多，因此在设计能够满足时钟频率要求的情况下，一般选中以节省资源。选中“Use FSM Explorer”选项，即可以用 synplify 内置的状态机浏览器观察状态机的各种属性。选中“Pipelining”选项，即启动流水，在高速时钟设计中，如果其他措施都不能达到目标频率则最好选中此项。
- 设置约束选项。点击“Constraints”属性页，设置模块最高工作频率以及添加约束文件（.sdc）。过严或是过松的约束都达不到最佳的效果。一般可先尝试通用的约束，如时钟扇出限制等；如果没有达到要求，可加入一些严格的具体约束，同时注意放松一些可以放松的约束。需要注意的是，综合约束的结果是估计值，应该以布局布线的结果为准。

- 设置实现结果。点击“Implementation Results”属性页，设置综合结果放置的目录，综合结果的文件名称。同时一定要将“Write Vendor Constraint File”和“Write Verification Interface Format”选项选中。

4) 时序约束

定义时间约束是为了让综合结果满足预期的时序要求，时间约束通常分为两类：一是通用时间约束，用于目标结构的时序要求；二是黑盒时间约束，用于在设计中指定为黑盒的模块时间约束。在 Synplify Pro 中，可通过 SCOPE、约束文件以及综合属性和指示等 3 种方法添加时序。本节主要介绍利用约束文件添加约束的方法。

约束文件采用 Tcl 语言，以*.sdc 保存，用来提供设计者定义的时序约束、综合属性以及 FPGA 生产商定义的属性等。约束文件既可以通过 SCOPE 创建编辑也可以使用正文编辑器创建编辑可被添加到在工程窗口的代码菜单中也可以被 Tcl 脚本文件调用。

5) 综合属性和指示

(1) 综合属性和指示简介

综合指示用于控制综合中编译阶段的设计分析，因而必须加入到源代码中。属性是在编译后读入的，因而既可以在源程序中说明，也可以在约束文件中说明。约束文件提供了较大的灵活性，使得可以仅修改约束而不用重新编译源程序，因而是强烈推荐采用的方法。

在 Verilog 源程序中，说明指示或属性采用注释的方法语法如下：

```
// synthesis directive|attribute = "value"  
或 /* synthesis directive |attribute = "value" */
```

(2) 综合指示

综合指示用于通知 Synplify Pro 软件某些用户定制的设置，常以注释的形式出现在源代码后面，Synplify 软件会自动识别相应的说明，按照用户指令完成综合。常用的综合只是如下：

① black_box_pad_pin

声明用户定义的黑盒管脚作为外部环境可见的 I/O pad。如果有不止一个端口列在双引号内，则以逗号分开。由于 Synplify 提供了预定义的 I/Os，一般不需要这一属性。其语法如下：

```
object /* synthesis syn_black_box black_box_pad_pin = "port_list" */ ;
```

例如：

```
module BS(D,IN,PAD,Q) /*synthesis syn_black_box black_box_pad_pin="PAD" */;
```

② block_box_tri_pins

声明黑盒的一个输出端口是三态，如不止一个列在双引号内，则以逗号分开。其语法如下：

```
object /* synthesis syn_black_box black_box_tri_pins = "port_list" */ ;
```

例如：

```
module BBDLHS(D,E,GIN,GOUT,PAD,Q) /* synthesis syn_black_box  
black_box_tri_pins="PAD" */ ;
```

③ full_case

仅用于 Verilog 中的 case 语句，表明所有可能的状态都已经给出，不需要其他逻辑保持信号的值，其语法如下：

```
object /* synthesis full_case */
```

其中 object 可以是 case、casex、casez、statements 和 declaration。

④ parallel_case

仅用于 Verilog 中 case 语句，表明生成一个并行的多路选择结构而不是一个优先译码结构。其语法如下：

```
object /* synthesis parallel_case */
```

其中 object 可以是 case、casex、casez、statements 和 declaration。

⑤ syn_block_box

说明一个模块或组件为黑盒，仅利用其界面进行综合，而不管内部是否为空，也不进行优化。一般应用于厂家原语或宏或 IP 等用户定义的宏。其语法如下：

```
object /* synthesis syn_black_box */ ;
```

其中 object 可以是 module 和 declaration。

⑥ syn_encoding

强制选择自动机实现的方式，其可选值（value）如下：

default: 综合根据状态的数量选择编码方式编码方式可以是 onehot gray sequential;

onehot: 采用 onehot 编码方式;

gray: 采用格雷码;

sequential: 采用自然码;

safe: 如果不能到达任一个状态时让其回到复位态。

syn_encoding 的语法如下：

```
object /* synthesis syn_encoding = "value" */;
```

其中 object 是状态寄存器定义。

⑦ syn_isclock

说明黑盒的一个输入是时钟信号。对名字为 clk 、 rclk 、 wclk 的黑盒输入信号，软件自动当作时钟，可以用这个属性说明任意输入信号为时钟信号。其语法如下：

```
object /* synthesis syn_isclock = 0|1 */;
```

其中 object 是黑盒的 input port。

例如：

```
module ram4(myclk, out, opcode, a, b) /* synthesis syn_black_box*/;
output [7:0] out;
input myclk /* synthesis syn_isclock = 1 */;
input [2:0] opcode;
input [7:0] a, b;
/* Other coding */
```

⑧ syn_keep

保证被指定的 wire 在综合中保持不动，不会被优化掉，常用于用 define_multicycle_path 或 define_false_path，用了-through 选项。如果你用了这一属性，将生成一个 keepbuf，可对其定义时间约束，且这个 Buffer 只占用一个位置，不出现在门级网表里。其语法如下：

```
object /* synthesis syn_keep = 0|1 */;
```

其中 `object` 是 `wire` 或 `reg` 声明。

⑨ `syn_noprune`

用来保持一个或多个 `component` 的实例，而不管其输出能否完成映射。一般在没有该指示的情况下，未用输出端口的实例会从 `EDIF` 文件中删除。`syn_noprune` 可被置于约束文件中，其语法如下：

`.sdc` 文件中：

```
define_attribute {module|instance} syn_noprune {0|1}
```

Verilog 中：

```
object /* synthesis syn_noprune = 0|1 */ ;
```

其中 `object` 可以是 `module`、`declaration`，也可以是实例。

⑩ `syn_preserve`

用在某些独立的寄存器上或模块，使模块中的所有寄存器在优化时保持不动，也可用于保持某个自动机在优化时不动。其语法如下：

```
object /* synthesis syn_preserve = 0|1 */ ;
```

其中 `object` 可以是寄存器定义信号，也可以是 `Module`。

- **`syn_sharing`**
确定综合时是否对运算符进行资源共享。缺省值是禁止，也可以在 `project` 视窗里设置这一选项。其语法如下：

```
object / * synthesis syn_sharing = " on|off " */ ;
```

其中 `object` 可以是 `module` 定义语句。
- **`syn_state_machine`**
对设计中的某组状态寄存器进行自动机优化，其语法如下：

```
object /* synthesis syn_state_machine = 0|1 */ ;
```

其中 `object` 是该组状态寄存器。
- **`syn_tco<n>`**
提供黑盒的输出延迟信息，其语法如下：

```
object /* syn_tcon = "[!]clock -> bundle = value" */ ;
```

其中 `bundle` 是总线或标量信号的集合。

- **syn_tpd<n>**
提供穿过黑盒的组合逻辑的传输延迟信息，其语法如下：
`object /* syn_tpdn = "[!]clock -> bundle = value" */ ;`
其中 **bundle** 是总线或标量信号的集合。
- **syn_tristate**
指定黑盒的一个输出端口为三态端口，其语法如下：
`object /* synthesis syn_tristate = 0|1 */ ;`
其中 **object** 可以是黑盒的 **output port**。
- **syn_tsu<n>**
说明一个黑盒的输入要求的建立时间，其语法如下：
`object /* syn_tsun = "[!]clock -> bundle = value" */ ;`
- **translate_on/translate_off**

用于与其他综合软件的兼容，这两者经常配对使用。在这两个指示中间的所有代码将在综合时被忽略，也可以用于在源代码中插入一段仿真代码。其语法如下：

```
/* synthesis translate_off */
```

综合时忽略的代码

```
/* synthesis translate_on */
```

6) 综合报告解读

综合报告主要由 3 部分组成：编译报告、映射优化报告以及时序报告，但是该报告是冗长的，不容易快速找出用户所关心的结果。因此，Synplify 公司提供了综合报告观察窗，如图 4-92 中第 3 部分所示，可从综合报告文件中取出重要的信息。该窗口的使用非常简单，点击空白的参数显示栏，在下拉栏中选择要查看的项目，则会在同行的右侧显示出结果，如图 4-94 所示。

Log Parameter	dpd_v5
dpd_v5 Part	xc4vzx25ff668-12
Worst Slack	996.968
System - Slack	998.238
System - Estimated Frequency	567.5 MHz
dpd_v5 I/O primitives	194
dpd_v5 Total Luts	1028 (34)
dpd_v5 Register bits (Non I/O)	2723 (84)

图 4-93 Synplify 综合结果示意图

4.5.2 ModelSim 软件的使用

ModelSim 软件是一款强大的仿真软件，具有速度快、精度高和便于操作的特点，此外还具有代码分析能力，可以看出不同代码段消耗资源的情况。其功能侧重于编译和仿真，不能制定编译的器件和下载配置的能力，所以需要和 ISE 等软件关联。

1. ModelSim 仿真软件的安装

下面介绍一下 ModelSim 的安装步骤。

- 1) 运行安装程序后，出现图 4-94 所示的界面，如果拥有有效的 License，可以选择完全版（Full Product）安装，反之，则应当选择评估版（Evaluation Edition）安装。



图 4-94 ModelSim 版本选择窗口

- 2) 选择完安装类型之后，下一个步骤就是设定安装路径，如图 4-95 所示。



图 4-95 ModelSim 安装路径选择窗口

3) 如果选择的是完全版本, 安装之后会出现 License Wizard 对话框, 如图 4-96 所示。

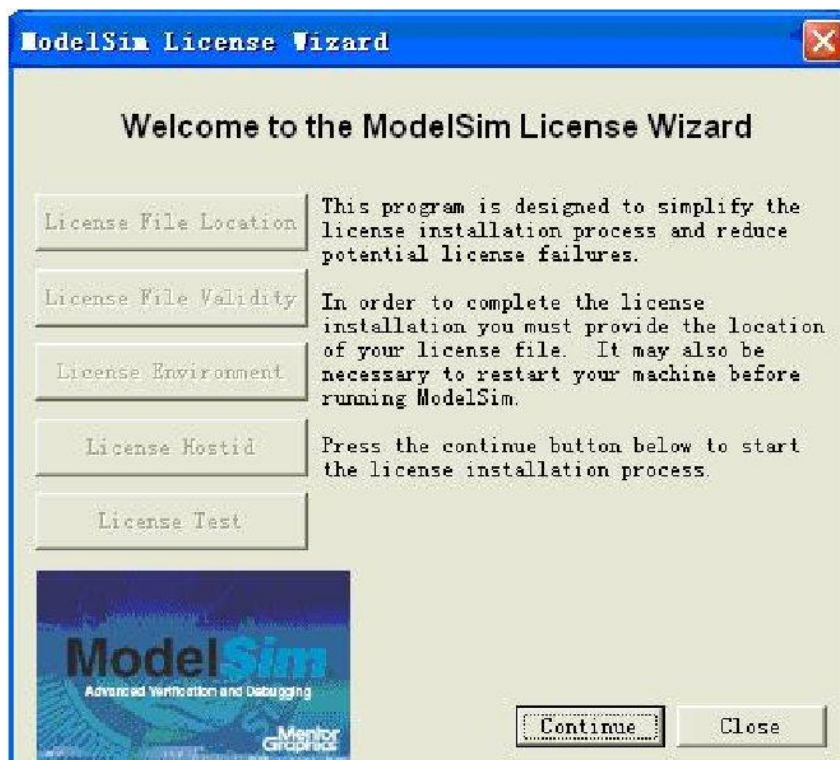


图 4-96 ModelSim 软件 License 管理向导

4) 单击 Continue 按钮后, 会出现 License 文件选择的对话框, 选择有效的 License 文件。单击 OK 按键后, 系统会自动进行一系列的有效性检查, 只有合法的 License 文件, 才能使 ModelSim 正常工作。

2. 关联 ISE 和 ModelSim

完成了 ModelSim 安装后, 需要将其和 ISE 软件关联后才能使用 ModelSim 进行仿真。运行 ISE 软件, 在主界面中选择“Edit|Preference”菜单项, 进行 Reference 设定如图 4-4 所示。

在弹出的“Preference”对话框中选择“Integrated Tools”选项卡。该选项卡用于设定与 ISE 集成的软件的路径, 第一项的“Model Tech Simulator”就用于设定 ModelSim 仿真软件的路径, 如图 4-5 所示。单击“Model Tech Simulator”文本框后面的按钮, 会弹出一个文件选择对话框, 选择 ModelSim 安装路径下 win32 目录下的“modelsim.exe”文件即可。

3. 在 ModelSim 中指定 Xilinx 的仿真库

ModelSim SE 版在发行时是不带任何 FPGA 厂家的仿真库, 因此用户必须手动编译这些库, 由此面临的一个问题就是怎样建立各 FPGA 器件的仿真库, 目前各 FPGA 厂家都支持用户编译库, 所以实现比较简单。

在 ModelSim 中编译 Xilinx 仿真库有很多方法, 下面介绍一种比较常用的方法, 分为 3 步来完成。

- 点击“开始/运行”按钮, 执行下面的命令:
“compplib -s mti_se -f all -l all -o f:\modeltech_6.2b\xilinx_libs -p f:\Modeltech_6.0\win32”
其中, f:\modeltech_6.2b 为 ModelSim 软件的安装目录, 用户可根据自己的目录来替换。
等待上述命令运行完毕, 其运行时间较长, 用户不要中途中断。
- 在 Xilinx 本地库编辑成功后, 在相应的目录下, 会自动生成 Modelsim.ini 的文件, 用任何一个文本编辑器将该文件中[Library]目录下(除 others 以外)的内容添加到硬盘上相应的另外的 ModelSim 安装目录下同名“modelsim.ini”文件中的相应[Library]位置。
- 最后进入 ISE 主界面, 点击“Edit”下拉菜单按钮, 选中下拉菜单中的“Preference”选项, 再选中“Intergal Tools”页面, 重新指定 ModelSim 可执行文件即可。退出所有软件, 以后在对 Xilinx 的设计进行仿真都不需要进行库的处理了。

4. ModelSim 使用方法简介

本节主要简单介绍 ModelSim SE 6. 6.2b 的使用方法, 主要包括建立工程和基本 Verilog 仿真, 更多的操作方法需要在实际应用中熟悉并掌握。

1) 建立工程

使用 ModelSim 建立工程主要包括 5 个基本步骤:

<1>启动 ModelSim, 选择菜单“File New Project”, 会打开“Creat Project”对话框, 如图 4-97 所示。在“Creat Project”对话框中填写“Project Name”为“test”, 然后在“Project Location”栏中选择 Project 文件的存储目录, 保留“Default Library Name”的设置为 work。点击 OK 按钮确认, 在 ModelSim 软件主窗口的工作区中即增加了一个空的 Project 标签, 同时弹出一个“Add items to the Project”对话框, 如图 4-98 所示。

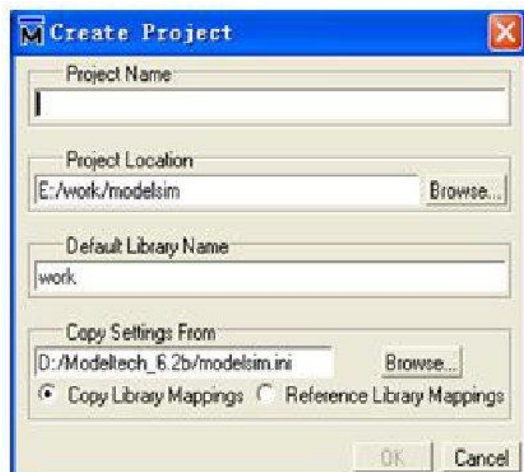


图 4-97 ModelSim 新建工程窗口

图 4-98 添加文件到工程向导示意图

<2>. 添加包含设计单元的文件。直接点击 Add items to the Project 对话框以后, 在对话框中利用“Add Existing File”或“Create New File”选项, 可以在工程中加入已经存在的文件或建立新文件。本节我们选择“Add Existing File”, 弹出“Add file to the Project”对话框, 如图 4-99 所示。点击对话框中的 Browse 按钮, 打开 ModelSim 安装路径中的 examples/tutorials/verilog/compare/ 目录, 选取 sm.v 和 sm.v 文件 (选中多个文件时, 只需要一直按住 Ctrl 按钮, 用鼠标点击即可), 再选中对话框下面的“Reference from current location”选项, 然后点击 OK 按钮确认。

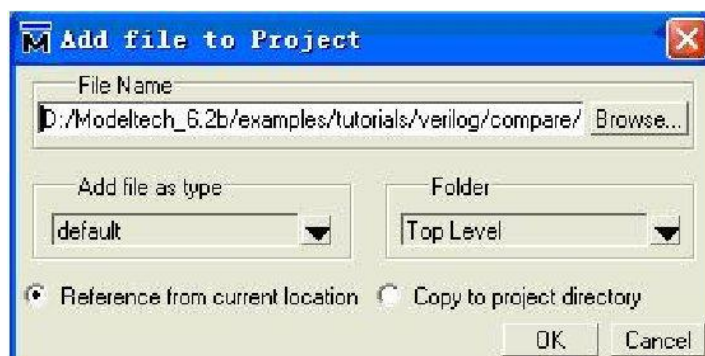


图 4-99 ModelSim 添加文件选项示意图

<3>.在工作区中的 **Project** 标签页中可以看到新加入的文件，单击右键，选取“Compile Compile All”命令对加入的文件进行编译，如图 4-100 所示。

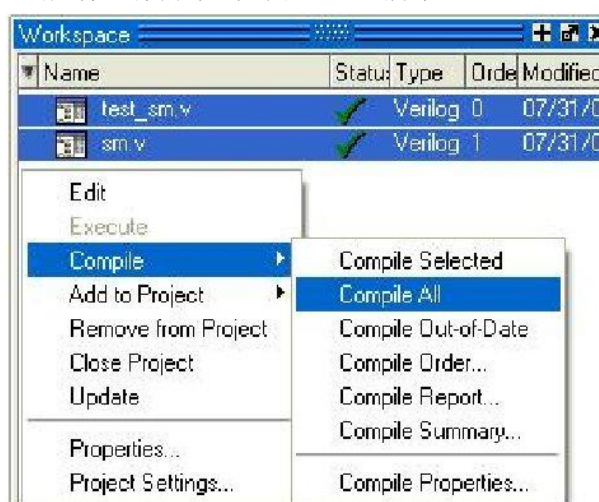


图 4-100 ModelSim 软件中的工程编译窗口

<4>. 两个文件编译完后，用鼠标点击“Library”标签栏。在标签栏中用鼠标点击 **work** 库前面的“+”，展开 **work** 库，就会看到两个编译了的设计单元。如图 4-101。



图 4-101 编译后的设计单元示意图

<5>. 导入设计单元。双击 Library 标签页中的“test_sm”，在工作区中将会出现 sim 标签，并在右边的对象窗口列出了 test_sm 单元所用到的信号，如图 4-102 所示。

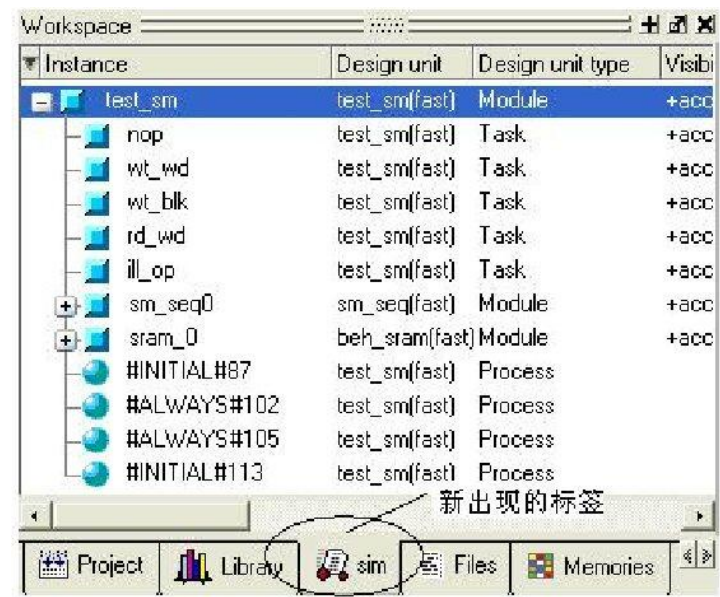


图 4-102 将 test_sm 模块加入工作区示意图

到此，一个工程就已经建立好了，接下来的就是开始运行仿真、分析和设计调试了。选择“File Close Project”可以关闭当前目录。

2) 基本 Verilog 仿真

在准备仿真的时候，需要完成（1）中的所有步骤。然后继续进行下面的步骤：

<1>. 通过选择“View <窗口名>”调出 signal、list 和 wave 窗口。 也可以通过在主窗口命令行操作区的 VSIM 提示符下输入下面的命令：view signals list wave（回车）

<2>. 向 wave 窗口添加信号。在 signal 窗口中，单击右键，在弹出的菜单中选择“Add to Wave”选项中的“Signal in design”，将设计中用到的所有信号都列在 Wave 窗口中，如图 4-103 所示。

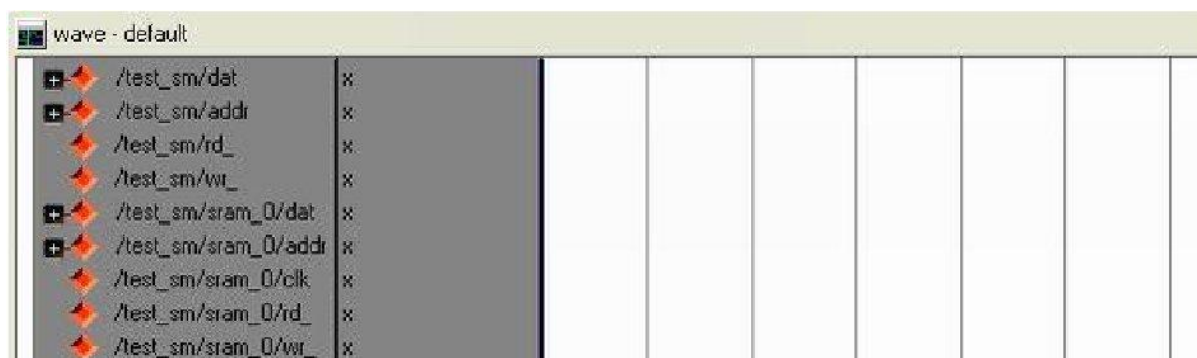


图 4-103 在 Wave 窗口中添加信号

<3>. 导入设计的时候，会在工作区打开一个新的 sim 标签，点击“+”展开设计层次结构，可以看到实例 test_sm、sm 等模块。点击 sim 标签中的顶层行保证 test_sm 模块显示在 source 窗口中。

<4>. 点击主窗口工具条的 Run 启动仿真，默认仿真长度为 100ns。或者在命令行输入 run。可以在仿真途中点击“Break”中断运行，在 source 窗口中查看中断时执行的语句。

<5>. 仿真完成，观察仿真波形如图 4-104。确认无误后退出仿真，如果有错则返回 source 区域修改代码。

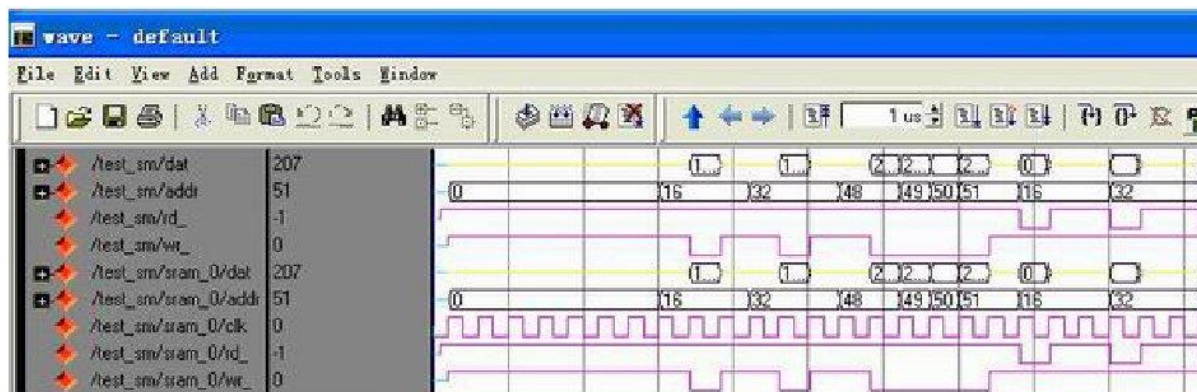


图 4-104 test_sm 模块的仿真结果示意图

4.5.3 Synplify Pro、ModelSim 和 ISE 的联合开发流程

利用 Synplify Pro、ModelSim 和 ISE 进行联合开发的步骤基本如下：当工程设计完成后，首先需要利用 ModelSim 软件完成功能仿真，然后利用 Synplify Pro 进行综合优化，再在 ISE 中完成映射与布局布线，并借助于 ModelSim 完成布局布线后的时序仿真，最后在 ISE 中完

成.bit 文件的生成和 FPGA 芯片的配置。

通常上述流程有两种实现方法：第一，将前两者作为 ISE 的第三方插件，在 ISE 中通过按键自动调用；其次，就是通过相应的接口文件，手动完成各个流程。自动调用比较简单，只需要在 ISE 中进行简单的设计即可。

1. 自动调用流程

完成了 Synplify Pro 和 ModelSim 的安装，并在集成工具设定页面完成与 ISE 的关联后，如图 4-91 所示，单击 ModelSim、Synplify /Synplify Pro 文本框后面的按钮，会弹出一个文件选择对话框，选择 ModelSim、Synplify /Synplify Pro 安装路径中 bin 目录下的*.exe 文件即可。

需要注意的是，并不是任意的 ISE 版本和任意的 ModelSim、Synplify/Synplify Pro 版本都可以实现无缝连接，只有在 ISE 发布后出现的第三方软件才能和 ISE 无缝连接，否则 Synplify/Synplify Pro 软件只能够用于综合逻辑，而不能识别 Xilinx 提供的 IP core，ModelSim 则不能用于嵌入式开发环境（EDK）设计的仿真。根据个人操作的结果，和 ISE 9.1 匹配的 Synplify /Synplify Pro 应当是 8.8 及其更高版本，相应的 ModelSim 应该为 6.1f 以后的版本。

完成了上述设定后，就可以直接在设计中调用 ModelSim 和 Synplify。在工程管理区的设计芯片上，点击右键，选择“Property”命令，即可打开用户设计的综合和仿真工具选择界面，如图 4-105 所示。在“Synthesis Tool”的下拉框中选择 Synplify（Verilog）、在“Simulator”中选择 ModelSim-SE Mixed。

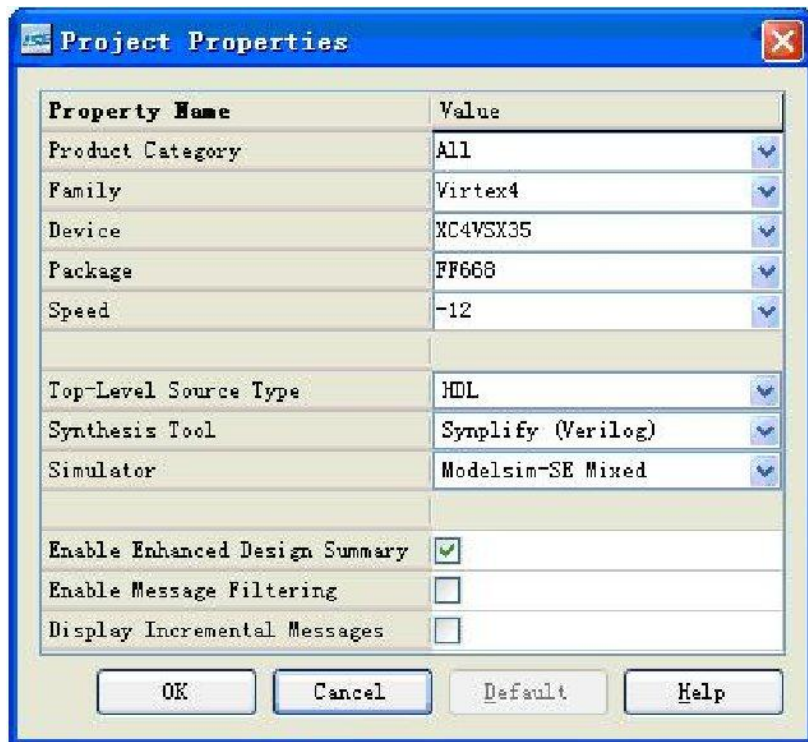


图 4-105 设计工具选择界面

2. 手动调用

由于 ModelSim 只是完成功能验证，也 ISE 没有直接的数据交互，手动操作就是分开单独操作。

手动调用 Synplify Pro 的方法比较灵活，但操作起来比较麻烦，用户可根据需要自行选择。首先单独启动 Synplify Pro 完成综合过程，再输出符合 ISE 格式的 EDIF 网表，在设置工程属性时选择 EDIF 设计流程，ISE 仅完成网表的转换、映射和布局布线等操作。同时可在 Synplify Pro 中添加时序约束。

综合完成后，生成的后缀为.edif 的文件就是综合输出的重要文件，是实现过程的输入，直接将其导入 ISE 即可。

4.5.4 ISE 与 MATLAB 的联合使用

本节主要介绍 MATLAB 设计、ISE 实现以及二者联合测试的 FPGA 开发流程，这是全书的核心思想之一，也是目前最流行的设计方法。

MATLAB 软件是 MathWorks 公司的核心产品，具有用法简单、扩展性好、资源库丰富以及与其他软件接口方便的特点，已成为从事电子信息和信号处理领域人员必备的工具软件之

一。MATLAB 和 ISE 的联合使用主要通过以下两个途径来实现：即 MATLAB 辅助 ISE 的方法，以及利用接口软件 System Generator 的方法。本书主要围绕着第一种方法展开讨论，但 System Generator 作为一种新兴的设计模式，具有强大的发展势头，这里对其也作了简单介绍，关于 System Generator 的详细讨论将在第 8 章展开。

1. MATLAB 辅助 ISE 完成 FPGA

所谓辅助，就是利用 MATLAB 来加速浮点算法的实现和功能测试。即在进行 FPGA 设计之前，先用 MATLAB 实现浮点算法，分析出算法的瓶颈所在，将程序的串行结构改造成并行结构，接着利用 MATLAB 完成定点仿真，得到满足性能需求的最小定点位宽以及中间步骤计算结果的截取范围，然后在 ISE 中完成设计。最后再利用 MATLAB 的定点仿真结果对设计进行功能验证，整个流程如图 4-106 所示。

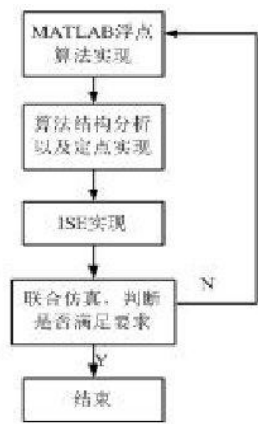


图 4-106 MATLAB 辅助 ISE 完成设计的流程图

关于怎样完成定点仿真以及模块输出结果截取的原理和方法将在第 5 章中展开详细关系讨论。此外，利用 MATLAB 对 FPGA 设计进行功能仿真还是比较关键的，下面介绍如何将 MATLAB 和 ModelSim 结合起来使用。

首先，在 MATLAB 中产生仿真所需的输入信号，以十六进制的形式存放在数据文件中，通常放在后缀为.txt 的文本文件中；其次在 ModelSim 中用 Verilog 编写仿真测试文件，并通过系统函数 \$readmemh 将上述仿真数据文件中的测试向量读入，在 ModelSim 中做功能仿真和时序仿真，并调用 \$fopen 函数打开另外一个数据文件，用 \$fdisplay 函数将仿真结果写入；再次，在 MATLAB 中将 ModelSim 仿真输出数据文件中的数据读入一个数组中，可以作图分析或者利用统计手段来分析。此外，还可以将 ModelSim 的仿真输出与 MATLAB 的浮点性能作对比，来验证设计的性能。这样比利用其他方式要方便、直观，并且具有更高的正确性。

下面举例介绍 Verilog 语言中文件输入/输出函数的使用方法。

1) 系统函数 \$fopen 用于打开一个文件，并返回一个整数的文件指针。然后，\$fdisplay 就可

以使用这个文件指针在文件中写入信息。写完后，用\$fclose 关闭文件。

其语法格式为：

```
integer <file_desc>;  
<file_desc> = $fopen("<file_name>", "<file_mode>");  
$fwrite(<file_desc>, "<string>", variables);  
$fclose(<file_desc>);
```

例如：integer W_file; //定义文件指针

```
W_file = $fopen("W_file.txt");  
$display(W_file, "@%h\n%h", a, b);  
$fclose(W_file);
```

以上语句将“a”和“b”分别显示在“@%h\n%h”中的两个%h位置，并写入Write_out_file指针所指的文件W_file.txt中。

2) 读文件操作通过\$readmemh和\$readmemb来完成，分别对应的数据文件为 16 进制和二进制。其语法格式为：

```
reg [<memory_width>] <reg_name> [<memory_depth>];  
$readmemh ("<file_name>", <reg_name>);
```

例如：reg [15:0] c [0:15];

```
$readmemh ("R_file.txt", c);
```

上例就是将R_file中的数据读入数组c中，然后就可以直接使用这些数据了。其中数据文件的格式为：

```
@2c  
45
```

其中，@2c表示地址，为 16 进制数，45 表示该地址的数据。

2. System Generator工具简介

System Generator工具由MathWorks 与 Xilinx 合作开发而成，DSP 设计人员可使用 MATLAB 和 Simulink 工具在 FPGA 内进行开发和仿真来完善 DSP 设计。新型 8.2 版本 System Generator使DSP系统和算法开发商不用写VHDL或Verilog编程，只需要利用MATLAB

及 **Simulink** 来开发他们的设计。一旦浮点建模完成，设计工程师采用 **Xilinx** 的比特及周期精确工具箱对其进行量化并自动生成 **HDL/RTL**、用于 **Xilinx FPGA** 的网表或完整的比特流，包括新的 **Virtex-5 LX** 和 **LXT** 器件。最后，设计工程师在 **Simulink** 环境内采用高带宽硬件环境来验证并调试实际 **FPGA** 上的设计。

System Generator 的关键特性主要包括：

- **DSP 建模。**利用包含信号处理（如 **FIR** 滤波器、**FFT**）、纠错（如 **Viterbi** 解码器、**Reed-Solomon** 编码器/解码器）、算法、存储器（如 **FIFO**、**RAM**、**ROM**）及数字逻辑功能的 **Xilinx** 模块集，在 **Simulink** 内构建和调试高性能 **DSP** 系统。**Xilinx** 模块集提供的模块可以使您导入 **MATLAB** 功能模块（如创建控制电路）**HDL** 模块（**System Generator** 为 **Mentor Graphics** 的 **ModelSim** 和 **Xilinx ISE** 仿真器提供了 **HDL** 协仿真接口）。
- **Simulink 的 VHDL 或 Verilog 的自动代码生成。**从 **Xilinx** 模块集实现行为（**RTL**）生成与对象明确的 **Xilinx IP** 核。
- **硬件协同仿真。**创建“**FPGA 在环路（FPGA-in-the-loop）**”仿真对象是代码生成选项，允许您验证工作硬件并加速 **Simulink** 与 **MATLAB** 中的仿真。**System Generator** 支持以太网（10/100/千兆位）、**PCI**、**Cardbus** 及硬件平台与 **Simulink** 之间的 **JTAG** 通信。
- **嵌入式系统的硬件/软件协设计。**为 **Xilinx MicroBlaze™ 32 位 RISC** 处理器构建和调试 **DSP** 协处理器。**System Generator** 提供了 **HW/SW** 接口的共享存储器提取功能，自动生成 **DSP** 协处理器、总线接口逻辑、软件驱动器及协处理器使用方面的软件技术文档。

一个典型的 **System Generator** 开发实例如图 4-107 所示，图中形如 **Xilinx** 公司标志的图标就是 **System Generator**，只需要双击图标，就可以将浮点算法自动转化成 **FPGA** 实现。**Xilinx** 公司网站上提供了 **System Generator** 完整的使用手册和丰富的实例，读者如有兴趣，可自行阅读。

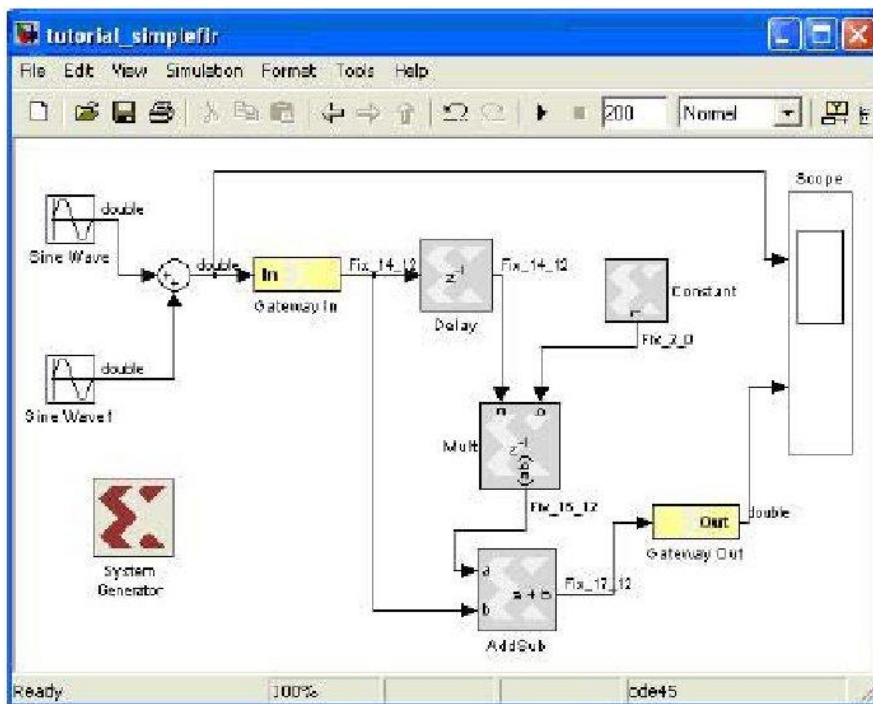


图 4-107 一个典型的 System Generator 开发实例

第 6 节 Xilinx FPGA 芯片底层单元的使用

4.6.1 Xilinx全局时钟网络的使用

在 Xilinx 系列 FPGA 产品中，全局时钟网络是一种全局布线资源，它可以保证时钟信号到达各个目标逻辑单元的时延基本相同。其时钟分配树结构如图 4-108 所示。

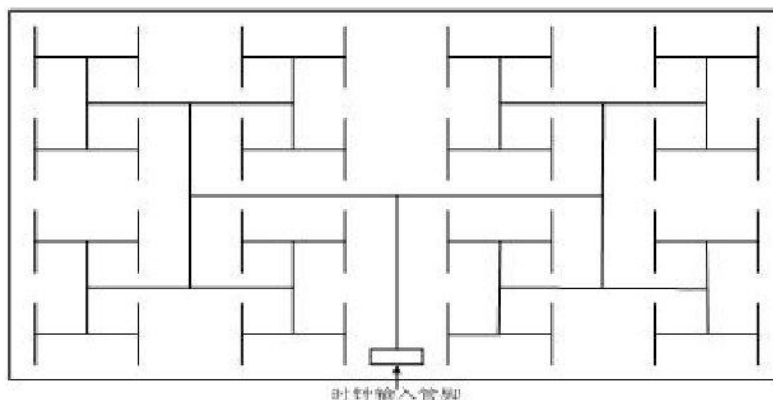


图 4-108Xilinx FPGA 全局时钟分配树结构

针对不同类型的器件，Xilinx公司提供的全局时钟网络在数量、性能等方面略有区别，下面以 Virtex-4 系列芯片为例，简单介绍FPGA全局时钟网络结构。

Virtex-4 系列FPGA利用 1.2V、90nm三栅极氧化层技术制造而成，与前一代器件相比，具备

灵活的时钟解决方案，多达 80 个独立时钟与 20 个数字时钟管理器，差分全局时钟控制技术将歪斜与抖动降至最低。以全铜工艺实现的全局时钟网络，加上专用时钟缓冲与驱动结构，从而可使全局时钟到达芯片内部所有的逻辑可配置单元，且I/O单元以及块RAM的时延和抖动最小，可满足高速同步电路对时钟触发沿的苛刻需求。

在FPGA设计中，FPGA全局时钟路径需要专用的时钟缓冲和驱动，具有最小偏移和最大扇出能力，因此最好的时钟方案是由专用的全局时钟输入引脚驱动的单主时钟，去钟控设计项目中的每一个触发器。只要可能就应尽量在设计项目中采用全局时钟，因为对于一个设计项目来说，全局时钟是最简单和最可预测的时钟。

在软件代码中，可通过调用原语 IBUFGP来使用全局时钟。IBUFGP的基本用法是：

```
IBUFGP U1(.I(clk_in), .O(clk_out));
```

全局时钟网络对FPGA设计性能的影响很大，所以本书在第 11 章还会更深入、更全面地介绍全局时钟网络以及相关使用方法。

4.6.2 DCM模块的使用

1. DCM模块的组成和功能介绍

数字时钟管理模块（Digital Clock Manager，DCM）是基于Xilinx的其他系列器件所采用的数字延迟锁相环（DLL，Delay Locked Loop）模块。在时钟的管理与控制方面，DCM与DLL相比，功能更强大，使用更灵活。DCM的功能包括消除时钟的延时、频率的合成、时钟相位的调整等系统方面的需求。DCM的主要优点在于：①实现零时钟偏移（Skew），消除时钟分配延迟，并实现时钟闭环控制；②时钟可以映射到PCB上用于同步外部芯片，这样就减少了对外部芯片的要求，将芯片内外的时钟控制一体化，以利于系统设计。对于DCM模块来说，其关键参数为输入时钟频率范围、输出时钟频率范围、输入/输出时钟允许抖动范围等。

DCM共由四部分组成，如图M所示。其中最底层仍采用成熟的DLL模块；其次分别为数字频率合成器（DFS，Digital Frequency Synthesizer）、数字移相器（DPS，Digital Phase Shifter）和数字频谱扩展器（DSS，Digital Spread Spectrum）。不同的芯片模块的DCM输入频率范围是不同的，例如：。

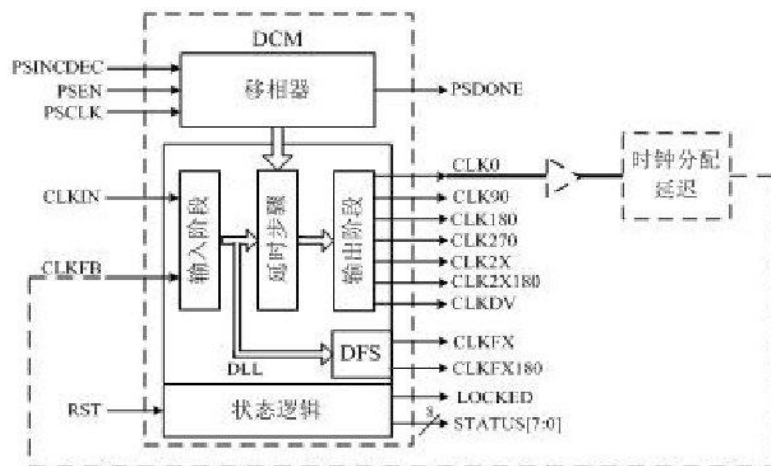


图 4-109 DCM 功能块和相应的信号

1) DLL 模块

DLL 主要由一个延时线和控制逻辑组成。延时线对时钟输入端 CLKIN 产生一个延时，时钟分布网线将该时钟分配到器件内的各个寄存器和时钟反馈端 CLKFB；控制逻辑在反馈时钟到达时采样输入时钟以调整二者之间的偏差，实现输入和输出的零延时，如图 4-110 所示。具体工作原理是：控制逻辑在比较输入时钟和反馈时钟的偏差后，调整延时线参数，在输入时钟后不停地插入延时，直到输入时钟和反馈时钟的上升沿同步，锁定环路进入“锁定”状态，只要输入时钟不发生变化，输入时钟和反馈时钟就保持同步。DLL 可以被用来实现一些电路以完善和简化系统级设计，如提供零传播延迟，低时钟相位差和高级时钟区域控制等。

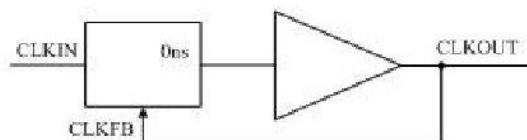


图 4-110 DLL 简单模型示意图

在 Xilinx 芯片中，典型的 DLL 标准原型如图 4-111 所示，其管脚分别说明如下：

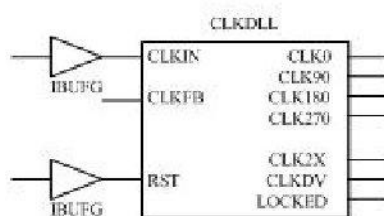


图 4-111 Xilinx DLL 的典型模型示意图

CLKIN（源时钟输入）：DLL 输入时钟信号，通常来自 IBUFG 或 BUFG。

CLKFB（反馈时钟输入）：DLL 时钟反馈信号，该反馈信号必须源自 CLK0 或 CLK2X，并

通过 IBUFG 或 BUFG 相连。

RST（复位）：控制 DLL 的初始化，通常接地。

CLK0（同频信号输出）：与 CLKIN 无相位偏移；CLK90 与 CLKIN 有 90 度相位偏移；CLK180 与 CLKIN 有 180 度相位偏移；CLK270 与 CLKIN 有 270 度相位偏移。

CLKDV（分频输出）：DLL 输出时钟信号，是 CLKIN 的分频时钟信号。DLL 支持的分频系数为 1.5, 2, 2.5, 3, 4, 5, 8 和 16。

CLK2X（两倍信号输出）：CLKIN 的 2 倍频时钟信号。

LOCKED（输出锁存）：为了完成锁存，DLL 可能要检测上千个时钟周期。当 DLL 完成锁存之后，LOCKED 有效。

在 FPGA 设计中，消除时钟的传输延迟，实现高扇出最简单的方法就是用 DLL，把 CLK0 与 CLKFB 相连即可。利用一个 DLL 可以实现 2 倍频输出，如图 4-112 所示。利用两个 DLL 就可以实现 4 倍频输出，如图 4-113 所示。

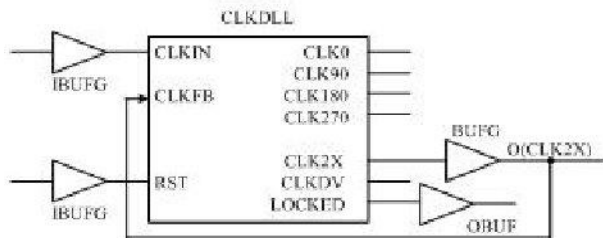


图 4-112 Xilinx DLL 2 倍频典型模型示意图

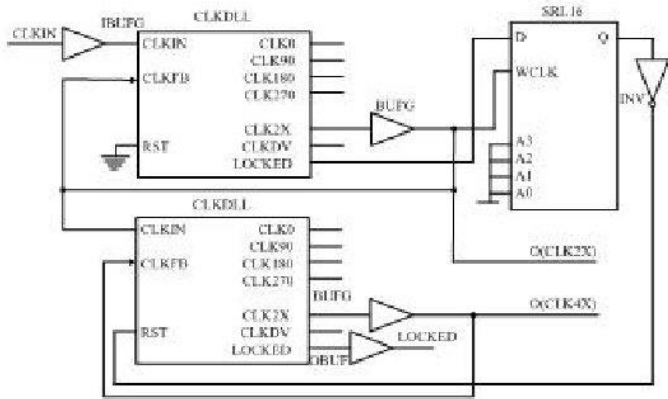


图 4-113 Xilinx DLL 4 倍频典型模型示意图

2) 数字频率合成器

DFS 可以为系统产生丰富的频率合成时钟信号，输出信号为 CLKFB 和 CLKFX180，可提供输入时钟频率分数倍或整数倍的时钟输出频率方案，输出频率范围为 1.5~320 MHz（不同芯片的输出频率范围是不同的）。这些频率基于用户自定义的两个整数比值，一个是乘因子

(CLKFX_MULTIPLY)，另外一个除因子 (CLKFX_DIVIDE)，输入频率和输出频率之间的关系为：

$$F_{CLKFX} = F_{CLKIN} \times \frac{CLKFX_MULTIPLY}{CLKFX_DIVIDE}$$

比如取 CLKFX_MULTIPLY = 3, CLKFX_DIVIDE = 1, PCB 上源时钟为 100 MHz, 通过 DCM 3 倍频后, 就能驱动时钟频率在 300 MHz 的 FPGA, 从而减少了板上的时钟路径, 简化板子的设计, 提供更好的信号完整性。

3) 数字移相器

DCM 具有移动时钟信号相位的能力, 因此能够调整 I/O 信号的建立和保持时间, 能支持对其输出时钟进行 0 度、90 度、180 度、270 度的相移粗调和相移细调。其中, 相移细调对相位的控制可以达到 1%输入时钟周期的精度 (或者 50 ps), 并且具有补偿电压和温度漂移的动态相位调节能力。对 DCM 输出时钟的相位调整需要通过属性控制 PHASE_SHIFT 来设置。PS 设置范围为 -255 到 +255, 比如输入时钟为 200 MHz, 需要将输出时钟调整 +0.9 ns 的话, $PS = (0.9ns / 5ns) \times 256 = 46$ 。如果 PHASE_SHIFT 值是一个负数, 则表示时钟输出应该相对于 CLKIN 向后进行相位移动; 如果 PHASE_SHIFT 是一个正值, 则表示时钟输出应该相对于 CLKIN 向前进行相位移动。

移相用法的原理图与倍频用法的原理图很类似, 只用把 CLK2X 输出端的输出缓存移到 CLK90、CLK180 或者 CLK270 端即可。利用原时钟和移相时钟与计数器相配合也可以产生相应的倍频。

4) 数字频谱合成器

Xilinx 公司第一个提出利用创新的扩频时钟技术来减少电磁干扰 (EMI) 噪声辐射的可编程解决方案。最先在 FPGA 中实现电磁兼容的 EMIControl 技术, 是利用数字扩频技术 (DSS) 通过扩展输出时钟频率的频谱来降低电磁干扰, 减少用户在电磁屏蔽上的投资。数字扩频 (DSS) 技术通过展宽输出时钟的频谱, 来减少 EMI 和达到 FCC 要求。这一特点使设计者可极大地降低系统成本, 使电路板重新设计的可能性降到最小, 并不再需要昂贵的屏蔽, 从而缩短了设计周期。

2. DCM 模块 IP Core 的使用

例 4-7 在 ISE 中调用 DCM 模块, 完成 50MHz 时钟信号到 75MHz 时钟信号的转换。

1) 在源文件进程中, 双击“Create New Source”; 然后在源文件窗口, 选择“IP (CoreGen &

Architecture Clocking Wizard)”，输入文件名“my_dcm”；再点击“Next”，在选择类型窗口中，“FPGA Features and Design Virtex-4”，然后选择“Single DCM ADV v9.1i”，如图 4-114 所示。

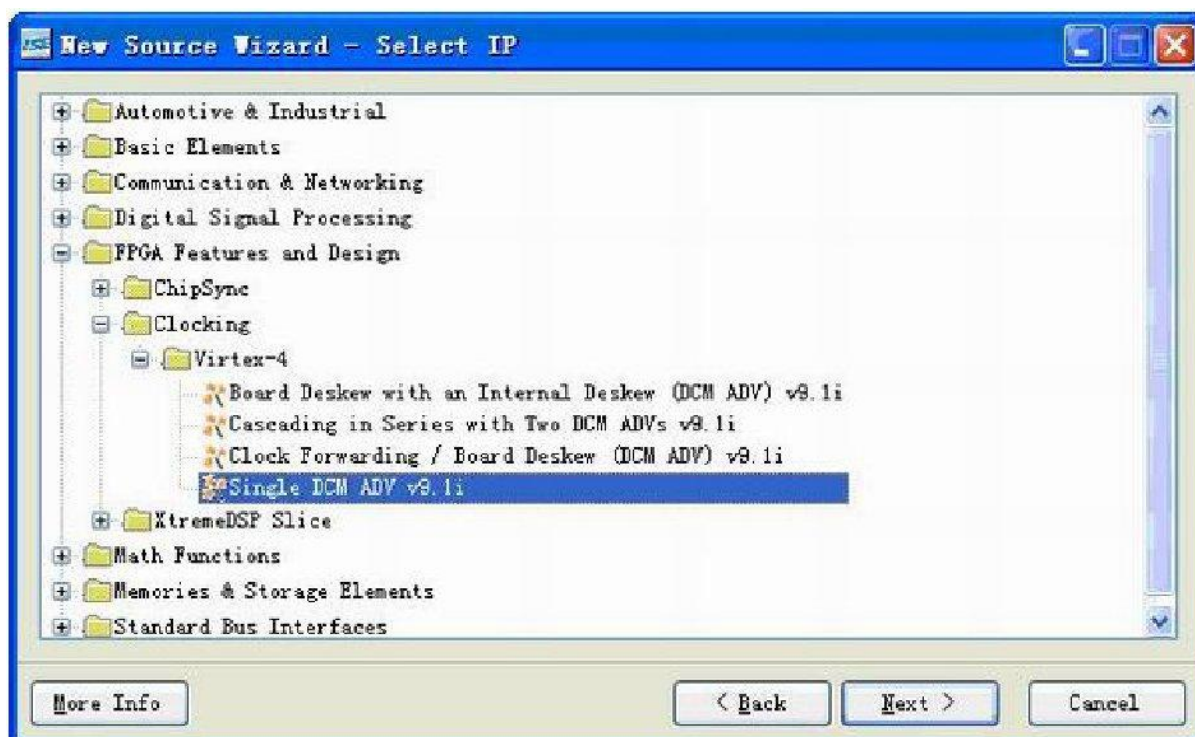


图 114 新建 DCM 模块 IP Core 向导示意图

<2> 点击“Next”，“Finish”进入 Xilinx 时钟向导的建立窗口，如图 4-65 所示。ISE 默认选中 CLK0 和 LOCKED 这两个信号，用户根据自己需求添加输出时钟。在“Input Clock Frequency”输入栏中敲入输入时钟的频率或周期，单位分别是 MHz 和 ns，其余配置保留默认值。为了演示，这里添加了 CLKFX 信号，并设定输入时钟为单端信号，频率为 50MHz，其余选项保持默认值。

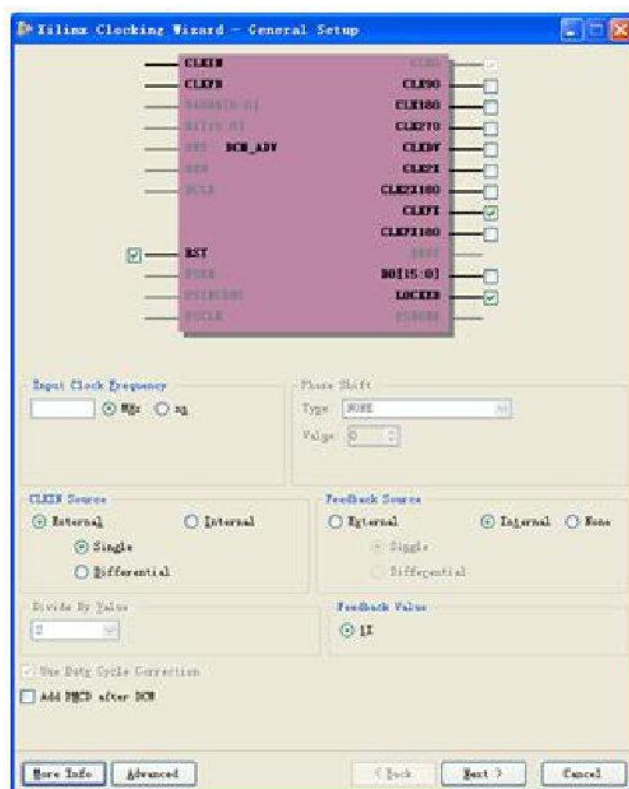


图-115 DCM 模块配置向导界面

<3> 点击“Next”，进入时钟缓存窗口，如图 4-116 所示。默认配置为 DCM 输出添加全局时钟缓存以保证良好的时钟特性。如果设计全局时钟资源，用户亦可选择“Customize buffers”自行编辑输出缓存。一般选择默认配置即可。



图 4-116 DCM 模块时钟缓存配置向导界面

<4> 点击“Next”，进入时钟频率配置窗口，如图 4-117 所示。键入输出频率的数值，或者将手动计算的分频比输入。最后点击“Next”，“Finish”即可完成 DCM 模块 IP Core 的全部配置。本例直接键入输出频率为 75MHz 即可。

Xilinx Clocking Wizard - Clock Frequency Synthesizer

Valid Ranges for Speed Grade -12

DFS Mode	F _{in} (MHz)	F _{out} (MHz)
Low	1.000 - 210.000	32.000 - 210.000
High	50.000 - 350.000	210.000 - 350.000

Inputs for Jitter Calculations

Input Clock Frequency: 50 MHz

☒ Use output frequency

75 ☐ MHz ☐ ns

☐ Use Multiply (M) and Divide (D) values

M 4 D 1

Calculate

Generated Output

M	D	Output Freq (MHz)	Period Jitter (unit interval)	Period Jitter (pk-to-pk ns)
3	2	75	0.014	0.185

More Info < Back Next > Cancel

图 4-117 指定 DCM 模块的输出频率

<5> 经过上述步骤，即可在源文件进程中看到“my_dcm.xaw”文件。剩余的工作就是在设计中调用该 DCM IP Core，其例化代码如下：

```
module dcm_top(
CLKIN_IN,
RST_IN,
CLKFX_OUT,
CLKIN_IBUFG_OUT,
```

```

CLK0_OUT,
LOCKED_OUT);
input CLKIN_IN;
input RST_IN;
output CLKFX_OUT;
output CLKIN_IBUFG_OUT;
output CLK0_OUT;
output LOCKED_OUT;
mydcm dcm1(
.CLKIN_IN(CLKIN_IN),
.RST_IN(RST_IN),
.CLKFX_OUT(CLKFX_OUT),
.CLKIN_IBUFG_OUT(CLKIN_IBUFG_OUT),
.CLK0_OUT(CLK0_OUT),
.LOCKED_OUT(LOCKED_OUT)
);
endmodule

```

<6> 上述代码经过综合 Synplify Pro 综合后，得到的 RTL 级结构图如图 4-118 所示。

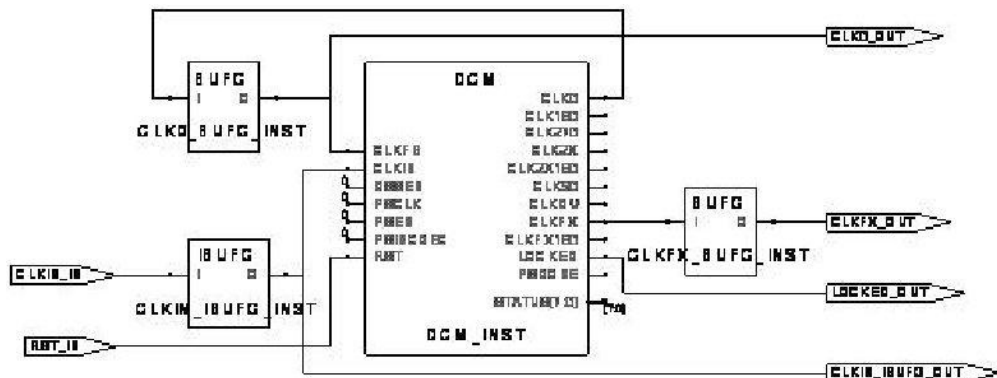


图 4-118 DCM 模块的 RTL 结构示意图

上述代码经过 ModelSim 仿真后，其局部仿真结果如图 4-119 所示。从中可以看出，当 LOCKED_OUT 信号变高时，DCM 模块稳定工作，输出时钟频率 CLKFX_OUT 为输入时钟 CLK_IN 频率的 1.5 倍，完成了预定功能。需要注意的是，复位信号 RST_IN 是高有效。

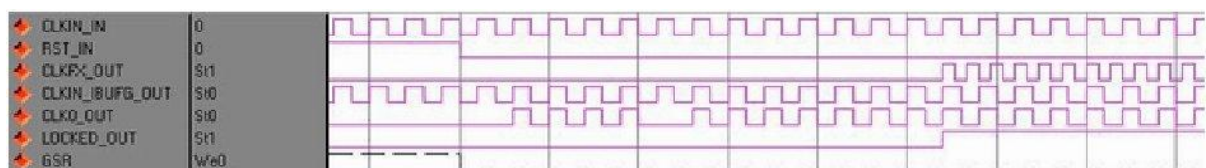


图 4-119 DCM 的仿真结果示意图

在实际中，如果在一片FPGA内使用两个DCM，那么时钟从一个clk输入，再引到两个DCM的clk_in。这里，在DCM模块操作时，需要注意两点：首先，用CoreGen生成DCM模块的时候，clk_in源是内部的，不能直接连接到管脚，需要添加缓冲器；其次，手动例化一个IBUFG，然后把IBUFG的输入连接到两个DCM的clk_in。通常，如果没有设置clk_in 源为内部的，而是完全按照单个DCM的使用流程，就会造成clk_in信号有多个驱动。此时，ISE不能做到两个DCM模块输出信号的相位对齐，只能做到一个DCM的输出是相位对齐的。而时钟管脚到两个DCM的路径和DCM输出的路径都有不同的延时，因此如果用户对相位还有要求，就需要自己手动调整DCM模块在芯片中的位置。

4.6.3 Xilinx内嵌快存储器的使用

Xilinx公司提供了大量的存储器资源，包括了内嵌的块存储器、分布式存储器以及 16 位的移位寄存器。利用这些资源可以生成深度、位宽可配置的RAM、ROM、FIFO以及移位寄存器等存储逻辑。其中，块存储器是硬件存储器，不占用任何逻辑资源，其余两类都是Xilinx专有的存储结构，由FPGA芯片的查找表和触发器资源构建的，每个查找表可构成 16 1 位的分布式存储器或移位寄存器。一般来讲，块存储器是宝贵的资源，通常用于大数据量的应用场合，而其余两类用于小数据量环境。

1. 块存储器的组成和功能介绍

在Xilinx FPGA中，块RAM是按照列来排列的，这样保证了每个CLB单元周围都有比较接近的块RAM用于存储和交换数据。与块RAM接近的是硬核乘加单元，这样不仅有利于提高乘法的运算速度，还能形成微处理器的雏形，在数字信号处理领域非常实用。例如，在Spartan 3E系列芯片中，块RAM分布于整个芯片的边缘，其外部一般有两列CLB，如图 4-120 所示，可直接对输入数据进行大规模缓存以及数据同步操作，便于实现各种逻辑操作。

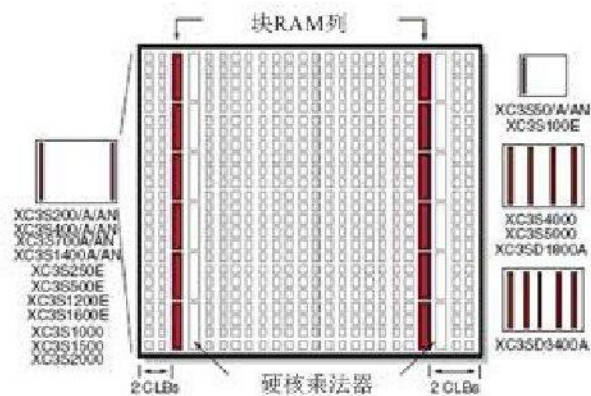


图 4-120 Spartan3E 系统芯片中块 RAM 的分布图

块 RAM 几乎是 FPGA 器件中除了逻辑资源之外用得最多的功能块，Xilinx 的主流 FPGA 芯片内部都集成了数量不等的块 RAM 硬核资源，速度可以达到数百兆赫兹，不会占用额外的 CLB 资源，而且可以在 ISE 环境的 IP 核生成器中灵活地对 RAM 进行配置，构成单端口 RAM、简单双口 RAM、真正双口 RAM、ROM（在 RAM 中存入初值）和 FIFO 等应用模式，如图 4-121 所示。同时，还可以将多个块 RAM 通过同步端口连接起来构成容量更大的块 RAM。

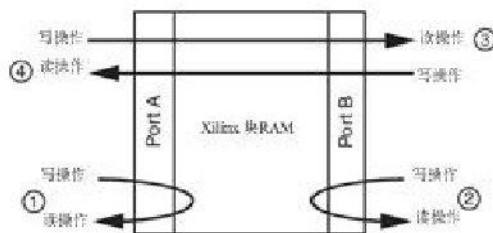


图 4-121 块 RAM 组合操作示意图

1) 单端口 RAM 模式

单端口 RAM 的模型如图 4-122 所示，只有一个时钟源 CLK，WE 为写使能信号，EN 为单口 RAM 使能信号，SSR 为清零信号，ADDR 为地址信号，DI 和 DO 分别为写入和读出数据信号。

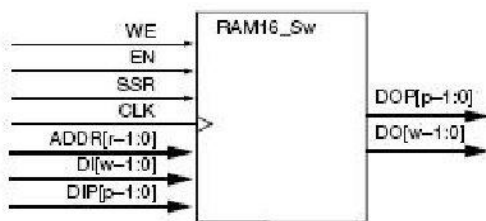


图 4-122 Xilinx 单端块 RAM 的示意模型

单端口 RAM 模式支持非同时的读写操作。同时每个块 RAM 可以被分为两部分，分别实现两个独立的单端口 RAM。需要注意的是，当要实现两个独立的单端口 RAM 模块时，首先要保

证每个模块所占用的存储空间小于块 RAM 存储空间 的 1/2。在单端口 RAM 配置中，输出只在 read-during-write 模式有效，即只有在写操作有效时，写入到 RAM 的数据才能被读出。当输出寄存器被旁路时，新数据在其被写入时的时钟上升沿有效。

2) 简单的双端口 RAM

简单双端口 RAM 模型如图 4-123 所示，图中上边的端口只写，下边的端口只读，因此这种 RAM 也被称为伪双端口 RAM（Pseudo Dual Port RAM）。这种简单双端口 RAM 模式也支持同时的读写操作。

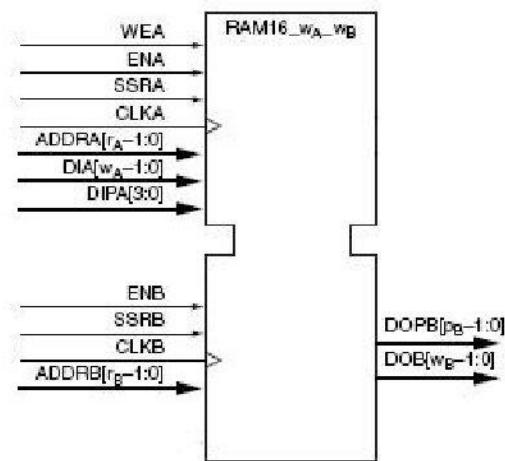


图 4-123 Xilinx 简单双端口块 RAM 的示意模型

块 RAM 支持不同的端口宽度设置，允许读端口宽度与写端口宽度不同。这一特性有着广泛地应用，例如：不同总线宽度的并串转换器等。在简单双端口 RAM 模式中，块 RAM 具有一个写使能信号 wren 和一个读使能信号 rden，当 rden 为高电平时，读操作有效。当读使能信号无效时，当前数据被保存在输出端口。当读操作和写操作同时对同一个地址单元时，简单双口 RAM 的输出或者是不确定值，或者是存储在此地址单元的原来的数据。

3) 真正双端口 RAM 模式

真正双端口 RAM 模型如图 4-124 所示，图中上边的端口 A 和下边的端口 B 都支持读写操作，WEA、WEB 信号为高时进行写操作，低为读操作。同时它支持两个端口读写操作的任何组合：两个同时读操作、两个端口同时写操作或者在两个不同的时钟下一个端口执行写操作，另一个端口执行读操作。

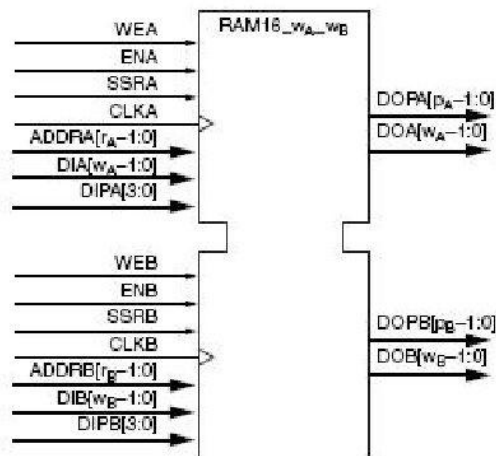


图 4-124 Xilinx 真正双端口块 RAM 的示意模型

真正双端口 RAM 模式在很多应用中可以增加存储带宽。例如，在包含嵌入式处理器 MiroBlaze 和 DMA 控制器系统中，采用真正双端口 RAM 模式会很方便；相反，如果在这样的一个系统中，采用简单双端口 RAM 模式，当处理器和 DMA 控制器同时访问 RAM 时，就会出现冲突。真正双端口 RAM 模式支持处理器和 DMA 控制器同时访问，这个特性避免了采用仲裁的麻烦，同时极大地提高了系统的带宽。

一般来讲，在单个块 RAM 实现的真正双端口 RAM 模式中，能达到的最宽数据位为 36 比特 * 512，但可以采用级联多个块 RAM 的方式实现更宽数据位的双端口 RAM。当两个端口同时向同一个地址单元写入数据时，写冲突将会发生，这样存入该地址单元的信息将是未知的。要实现有效地向同一个地址单元写入数据，A 端口和 B 端口时钟上升沿的到来之间必须满足一个最小写周期时间间隔。因为在写时钟的下降沿，数据被写入块 RAM 中，所以 A 端口时钟的上升沿要比 B 端口时钟的上升沿晚到来 1/2 个最小写时钟周期，如果不满足这个时间要求，则存入此地址单元的数据无效。

4) ROM 模式

块 RAM 还可以配置成 ROM，可以使用存储器初始化文件 (.coe) 对 ROM 进行初始化，在上电后使其内部的内容保持不变，即实现了 ROM 功能。

5) FIFO 模式

FIFO 即先入先出，其模型如图 4-125 所示。在 FIFO 具体实现时，数据存储的部分是采用简单双端口模式操作的，一个端口只写数据而另一个端口只读数据，另外在 RAM（块 RAM 和分布式 RAM）周围加一些控制电路来输出指示信息。FIFO 最重要的特征是具备“满（FULL）”和“空（EMPTY）”的指示信号，当 FULL 信号有效时（一般为高电平），就不能再往 FIFO

中写入数据，否则会造成数据丢失；当 **EMPTY** 信号有效时（一般为高电平），就不能再从 **FIFO** 中读取数据，此时输出端口处于高阻态。

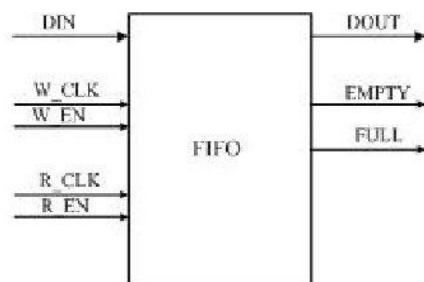


图 4-125 Xilinx FIFO 模块的示意模型

2. 块 RAM IP Core 的使用

块 **RAM** 已在本书第 3 章有过介绍，这里就不再赘述。

3. ROM 存储器 IP Core 的使用

对于 **ROM** 模块，主要是生成相应的.coe 文件。下面以一个实例介绍如何借助 **MATLAB** 生成 **ROM** 的.coe 文件。

例 4-8 生成定点正余弦波形数值，形成.coe 文件并加载到块 **ROM** 中。

整体过程主要分为下面的 3 步。

首先，利用 **MATLAB** 计算出正余弦波形的浮点值，并量化 16 比特的定点波形数值：

```
x= linspace(0,6.28,1024); //在区间[0,6.28]之间等间隔地取 1024 个点
```

```
y1=cos(x); //计算相应的正余弦值
```

```
y2=sin(x);
```

```
//由于正余弦波形的值在[0,1]之间，需要量化成 16 比特，先将数值放大
```

```
y1=y1*32678;
```

```
y2=y2*32768;
```

```
//再将放大的浮点值量化，并写到存放在 C 盘的文本中
```

```
fid = fopen('c:/cos_coe.txt', 'wt');
```

```
fprintf(fid, '%16.0f\n', y1); //在写文件的时候量化成 16 比特
```

```
fclose(fid)
```

```
fid = fopen('c:/sin_coe.txt', 'wt');
```

```
fprintf(fid, '%16.0f\n', y2);
```

fclose(fid)

其次，生成 coe 文件。在 C 盘根目录下，将 cos_coe.txt 和 sin_coe.txt 的后缀改成.coe，打开文件，把每一行之间的空格用文本的替换功能换成逗号“,”，并在最后一行添加一个分号“;”。最后在文件的最开始添加下面两行：

```
memory_initialization_radix=10;
```

```
memory_initialization_vector =
```

然后保存文件退出。

最后，将 coe 文件加载到 BLOCKROM 所生成的 ROM 中。新建一个 BLOCKRAM 的 IP core，其位置为“Memories & Storage Elements RAMs & ROMs Block Memory Generator v2.4”，

在第一页选择 single port rom，在第二页选择位宽为 16、深度为 1024，在第三页下载 coe 文件，如图 4-126 所示，然后双击“Finish”，完成 IP core 的生成。如果 coe 文件生成的不对，图中用椭圆标志之处是红色的，coe 文件错误的类型主要有数据基数不对和数据的长度不对这两类。

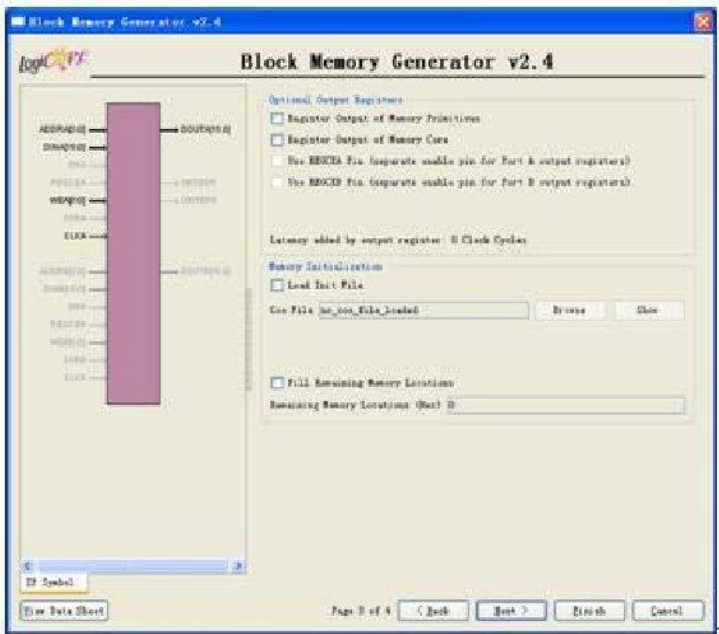


图 4-126 块 ROM 加载 coe 文件的用户配置界面

第 6 节 Xilinx FPGA 芯片底层单元的使用 2

4.6.4 硬核乘法器的使用

随着 FPGA 芯片容量的提高和工艺的发展，很多 FPGA 内部都内嵌了硬件乘法器，所以 FPGA 内部的乘法器就有两种实现方式：硬核乘法器的实现和用逻辑单元搭建的乘法器。目前由于内嵌了大量的硬核乘法器，FPGA 在数字信号处理系统方面的成本和功耗性能已经超越了专

用的 DSP 处理器。

1. 硬核乘加器的组成和功能介绍

硬核乘加器是 Xilinx XtremeDSP 解决方案的核心组成部分，从而可以独立实现 500MHz 的性能，或在整合到一系列中实现 DSP 功能，支持 40 多个动态控制的操作模式，包括乘法器、乘法器-累加器、乘法器-加法器/减法器、三输入加法器、桶形移位器、宽总线多路复用器或宽计数器，其组成结构如图 4-127 所示。此外，级联硬核乘加器，无需使用 FPGA 逻辑和路由资源。

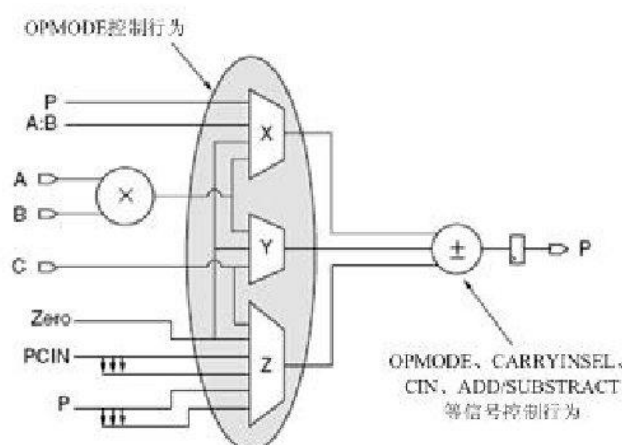


图 4-127 硬核乘加器的组成结构

图 4-127 中的 OPMODE 是乘加器工作模式配置输入，可在 ISE 中通过软件方式指定。硬核乘加器的乘法器和加法器可以单独使用，但对于一个乘加器资源，只使用了其乘法器或加法器，则另外的加法器或乘法器就不能再被使用。不同系列芯片中乘加器的特点略有不同，从整体而言，都具有以下特点：

- 18 位 18 位，两个补码乘法器具有完全准确的 36 位结果、符号可以扩展到 48 位。
- 三输入、灵活的 48 位加法器/减法器，具有可选的寄存器累加反馈。
- 40 多个动态用户控制器操作模式，使乘加器的功能适应从一个时钟周期到下一个时钟周期的变化。
- 级联的 18 位 B 总线，支持输入取样传递。
- 级联的 48 位 p 总线，支持部分结果的输出传递。
- 多精度乘法器和算法支持 17 位操作数右移位，以对准宽乘法器的部分乘积（并行或顺序乘法）。
- 对称智能舍入支持更高的计算精度。
- 控制和数据信号使用的、能够提高性能的流水线选项，可以通过配置位来进行选择。
- 输入端口“C”通常用作乘、加、进位的三操作数加或灵活的舍入模式。

- 独立的复位和时钟，实现了控制和数据寄存器。

2. 硬核乘法器 IP Core 的使用

硬核乘法器 IP Core 可以完成有符号数、以及无符号数的乘法，还能够完成输出数据的位宽截取，支持流水线操作，功能强大。其用户操作界面如图 4-128 所示，点击“Next”按钮进入下一页，可以让用户选择是使用 FPGA 芯片上的硬乘法器（Use Mults），还是用 Slice 来构建乘法器（Use LUTs）。

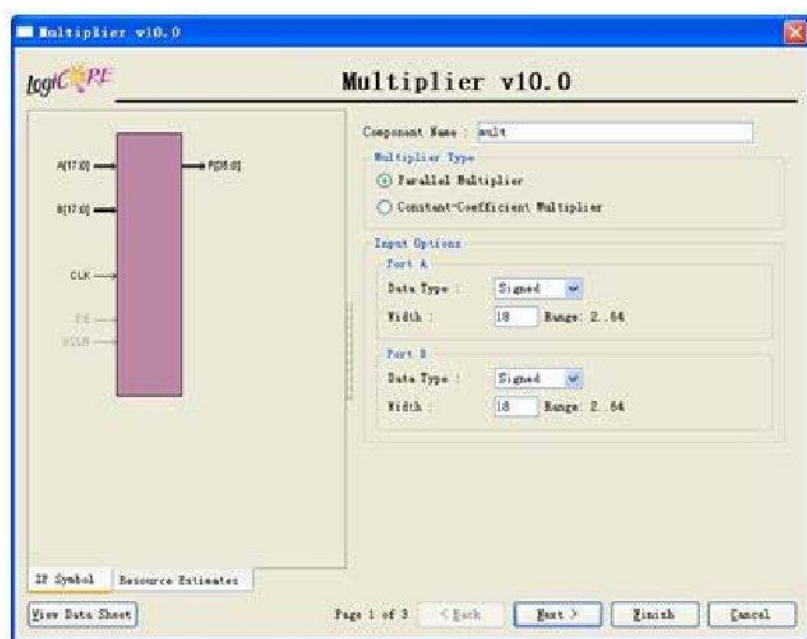


图 4-128 乘法器 IP core 用户操作界面

乘法器具有丰富的控制信号，其详细说明见下文。

A: 乘法器的一个输入操作数，在使用时应当确定其位宽，其位宽可以在右边的端口位宽编辑框可以输入，并且可以有符号数、无符号数的两种选择。

B: 乘法器的一个输入操作数，在使用时应当确定其位宽，其位宽可以在右边的端口位宽编辑框可以输入，而且可以与 A 口的输入操作数位宽不同。可以有符号数、无符号数的两种选择。（注：乘法器可以只接收 A 口的输入信号，而与一常数相乘，此常数可以为固定的或可重导入的。对于可重导入数据（RCCM）情况中，此常数可以在 B 口重新导入而得到改变。）

CLK: 乘法器的工作时钟。上升沿有效。

CE: 输入信号，指示时钟是否有效。

ND: 握手信号。

ACLR: 异步清零信号。

SCLR: 同步清零信号。

LOADB: 当 RCCM 时才有效。当此信号为高时, B 端口新的输入可以重新写入计算模块的存储单元。

SWAPB: 当 RCCM 时才有效。当此信号为高时, 在计算模块的存储单元已存有的多个常数中进行选择。

RDY: 输出的握手信号, 当其变高时表明数据有效。

RFD: 握手信号, 在其变高后的下一个时钟上升沿数据有效。

O: 异步输出信号, 位宽根据输入信号的位宽而定。

Q: 同步输出信号, 位宽根据输入信号的位宽而定。

LOAD_DONE: 当 RCCM 时才有效, 指示重新导入数据的过程完成。

例 4-9 使用 IP Core 实例化一个 16 位乘法器

IP Core 直接生成的乘法器的 Verilog 模块接口为:

```
module multiply(sclr, rfd, rdy, nd, clk, a, b, q);  
input sclr;  
output rfd;  
output rdy;  
input nd;  
input clk;  
input [17 : 0] a;  
input [17 : 0] b;  
output [35 : 0] q;  
.....  
endmodule
```

在使用时, 直接调用 multiply 模块即可, 如:

```
module multiply1 (sclr, rfd, rdy, nd, clk, a, b, q);  
input sclr, nd, clk;  
output rfd, rdy;  
input [15 : 0] a, b;  
output [31 : 0] q;  
  
multiply multiply1(.sclr(sclr), .rfd(rfd), .rdy(rdy), .nd(nd), .clk(clk),  
.a(a), .b(b), .q(q));  
endmodule
```

上述程序经过 Synplify Pro 综合后，得到的 RTL 结构如图 4-129 所示。

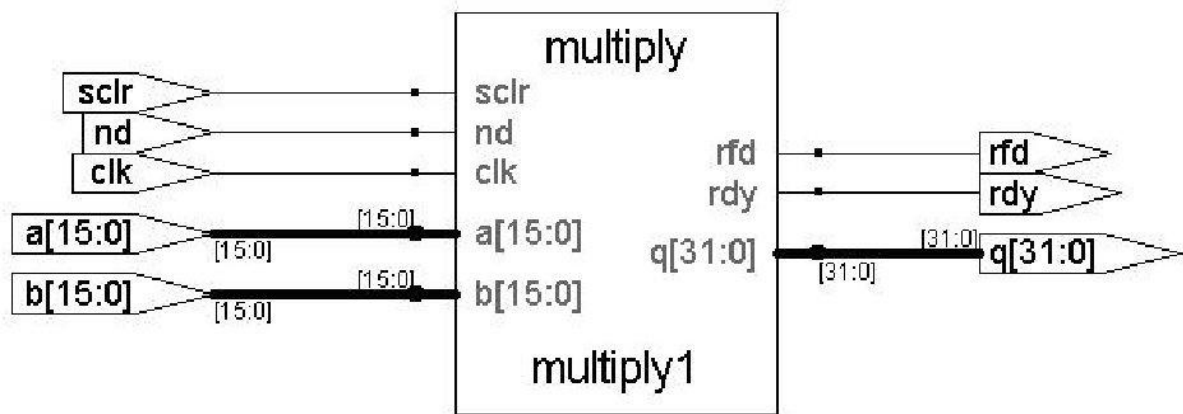


图 4-129 乘法器 IP Core 综合后的 RTL 结构图

经过仿真测试得到的功能波形图如图 4-130 所示，正确地实现了乘法功能。

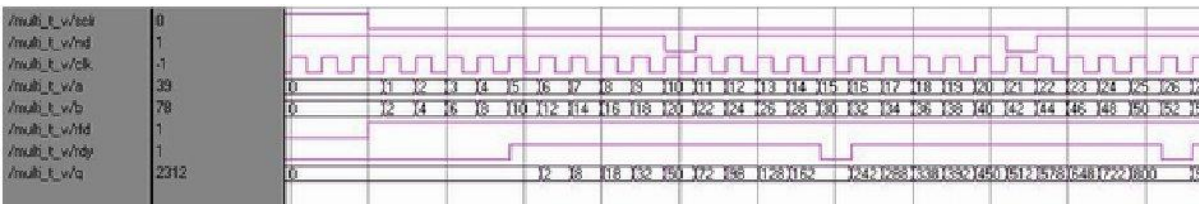


图 4-130 乘法器 IP core 的仿真波形

3. 硬核乘加器 IP Core 的使用

硬核乘加器在乘法器后面级联了一个可控加法器，都可工作在芯片的最高工作频率。下面给出一个应用实例。

例 4-10 使用硬核乘加器完成两路输入数据的相乘，并将每 8 个乘积结果累加后送出。其中输入数据为 16 比特，工作频率为 50MHz。

1) 在工程中添加硬核乘加器的 IP Core 文件，位于“FPGA Features and Design”“XtremeDSP Slice”“Multiply Accumulator v9.1i”。

2) 配置 IP Core 参数。输入数据位宽设为 16 比特，且输入输出不与其余的 DSP Slice 级联，输出位宽设置为 19 比特，如图 4-131 所示，点击“Next”进入下一页；无进位选项，且只选择加法输入，OPMODE 模式设置为“Normal accumulator mode”，如图 4-132 所示，点击“Next”进入下一页；A、B 流水线都设为 1，其余设置如图 4-133 所示，点击“Next”进入下一

页；最后一页为乘加器的配置参数列表，如图 4-134 所示，点击“Finish”按钮，即可完成全部配置。

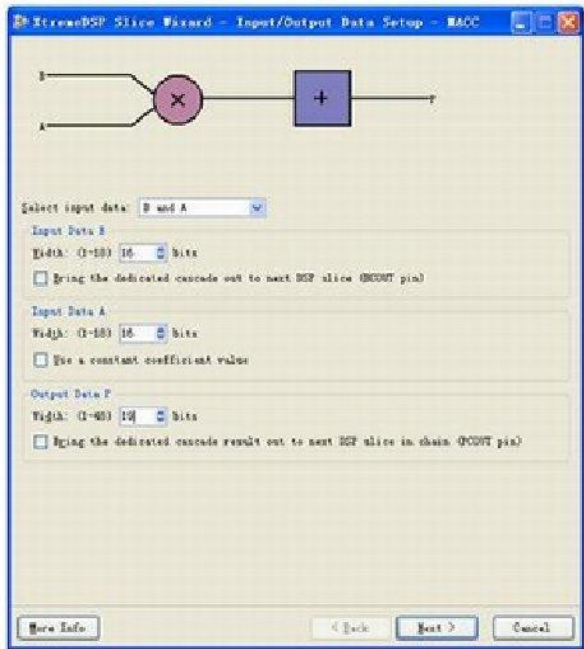


图 4-131 乘法器 IP core 的配置界面（1）

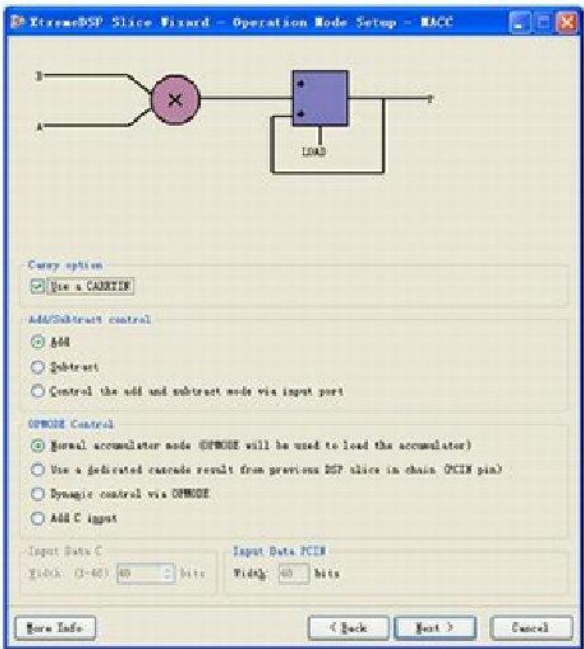


图 4-132 乘法器 IP core 的配置界面（2）

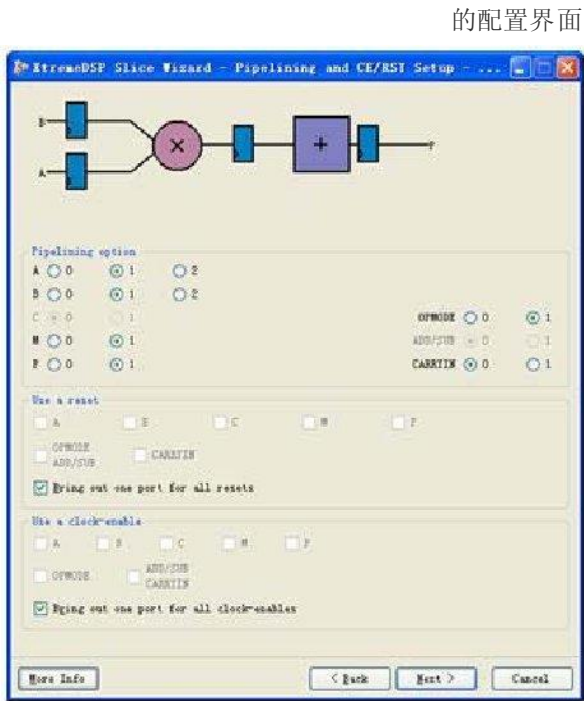


图 4-133 乘法器 IP core 的配置界面（3）

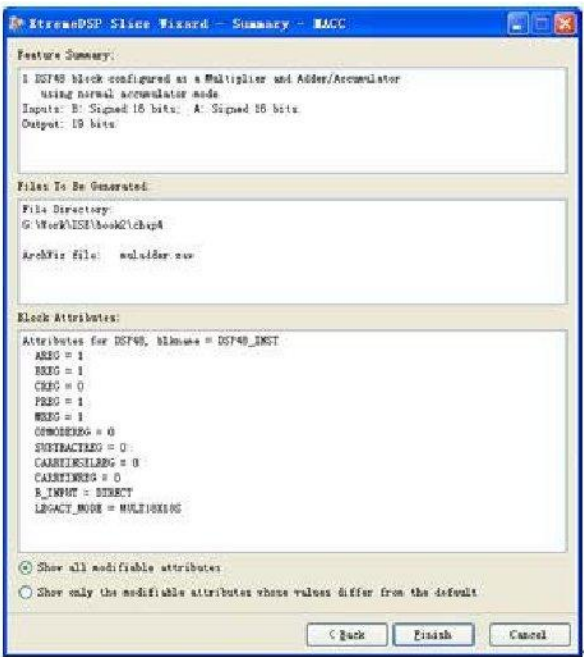


图 4-134 乘法器 IP core 的配置界面（4）

3) 在过程窗口中点击“View HDL Instantiation Template”命令，可查阅其代码例化模版，如下所列：

```

muladder instance_name (
.A_IN(A_IN),
.B_IN(B_IN),
.CE_IN(CE_IN),
.CLK_IN(CLK_IN),
.LOAD_IN(LOAD_IN),
.RST_IN(RST_IN),
.P_OUT(P_OUT)
);

```

4) 调用该 IP Core 完成设计，代码如下：

```

module my_muladder(clk_50MHz, reset, ce, dina, dinb, dout);
input clk_50MHz;
input reset;
input ce;
input [15:0] dina;
input [15:0] dinb;
output [34:0] dout;

reg [34:0] dout;
wire [34:0] p_out;
reg load = 0;
reg [2:0] cnt = 0;
always @(posedge clk_50MHz) begin
    if(reset) begin
        cnt <= 0;
        load <= 0;
        dout <= 0;
    end
    else begin
        cnt <= cnt + 1'b1;
        if(cnt == 0)
            load <= 1'b0; //直通加法器
        else
            load <= 1'b1; //load=1 累加
        if(cnt == 2)

```

```
        dout <= p_out;
    else
        dout <= dout;
    end
end
end
```

```
muladder muladder(
.A_IN(dina),
.B_IN(dinb),
.CE_IN(ce),
.CLK_IN(clk_50MHz),
.LOAD_IN(load),
.RST_IN(reset),
.P_OUT(p_out)
);
endmodule
```

上述程序经过 Synplify Pro 综合后，得到的 RTL 结构如图 4-135 所示。

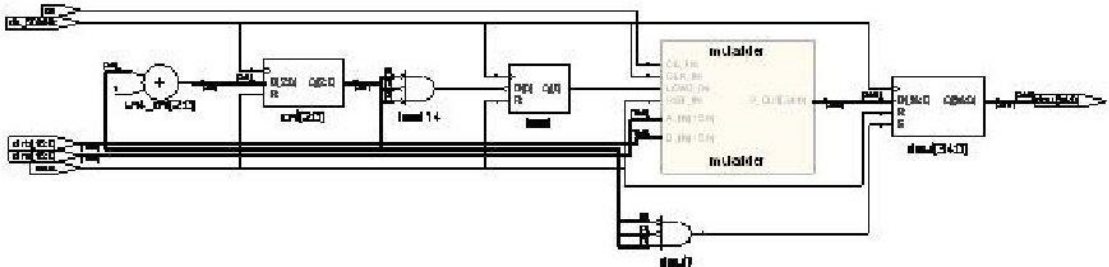
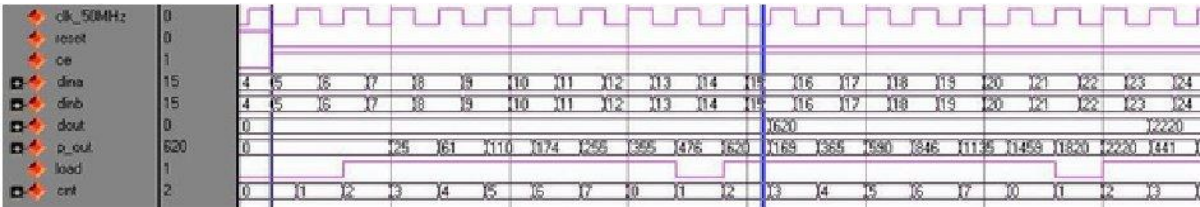


图 4-135 乘加器应用程序的 RTL 结构图

经过仿真测试得到的功能波形图如图 4-136 所示，可以看到本例正确地实现了 8 个乘积结果累加的功能。



本章详细介绍了基于 ISE 的 FPGA 设计流程以及多个辅助工具（XST、XPower、PACE、ModelSim、Synplify 以及 MATLAB）的使用方法。首先介绍了 ISE 软件主要特性及其安装流程，然后介绍了如何通过 ISE 完成 FPGA 设计，详细介绍了综合、仿真以及实现的软件操作和 XST、XPower、PACE 等工具的基本操作。之后，简单介绍了 Synplify Pro 和 ModelSim SE 的安装流程，并介绍如何在 ISE 关联 Synplify Pro 和 ModelSim SE 的使用方法以及和 ISE 的联合开发流程。再次，介绍了 MATLAB 和 ISE 的联合开发模式，其中利用 MATLAB 来辅助 ISE 开发是目前流行的设计方法之一，也是本书所强调的重点。最后，介绍了 Xilinx FPGA 底层单元（DCM、块 RAM 以及硬核乘加器）的原理和使用方法。鉴于篇幅，不可能对以上所提及软件的其他功能进行详细介绍，读者应该在实际工作中学习并熟练掌握。