

FPGA 应用实验指导书

网络安全学院

2018

目 录

1、引言	1
1.1 初学者开发板主要特性.....	2
1.2 SPARTAN 3E 系列 FPGA 特性	3
1.3 SPARTAN 3E 系列 FPGA 主要技术特征.....	3
2、XILINX ISE DESIGN SUITE 10.1 简介	4
2.1 ISE DESIGN SUITE 10.1 综述	4
2.2 ISE DESIGN SUIT 10.1 的创新特性	4
2.3 ISE DESIGN SUIT10.1 主要组件	6
2.3.1 ISE Foundation.....	7
2.3.2 EDK 开发工具.....	8
2.3.3 DSP 工具.....	9
2.3.4 ChipScope Pro.....	10
2.3.5 PlanAhead.....	10
3、ISE 的 FPGA 开发流程	12
4、STEP BY STEP——使用 ISE 设计 FPGA	14

4.1 启动 ISE	14
4.2 建立工程(PROJECT)开始第一个设计	15
4.2.1 设计目标	15
Step 1	16
Step 2	16
Step 3	17
Step 4	17
Step 5	18
Step 6	19
Step 7	20
Step 8	20
Step 9	22
Step 10	22
Step 11	23
Step 12 创建用户设计约束文件.....	24
Step 13	25
Step 14	27
Step 15	28
Step 15	28
Step 16	29
Step 17	30
Step 18	33
5、FPGA 开关电路设计	34
5.1 设计目标	34

5.2 功能描述	34
5.3 设计中需要解决的问题	35
5.4 建立 ISE 实现设计	39

1、引言

《FPGA 应用实验》课的主要教学内容包括：培养学生分析、设计数字系统的基本技能、使学生能够利用 EDA 开发工具，在 FPGA/CPLD 等可编程器件上，设计基本的逻辑电路模块，初步掌握实际设计各种数字系统基本方法。

实验课使用的开发工具和开发板分别为：

- ✧ FPGA 设计工具：Xilinx ISE Design Suite 10.1 开发系统；
 - ✧ FPGA 实验开发板：Xilinx Spartan-3E 初学者开发板（Xilinx Spartan-3E Starter Kit）；
- Diligent 公司设计的 Xilinx Spartan-3E 初学者开发板如图 1 所示：

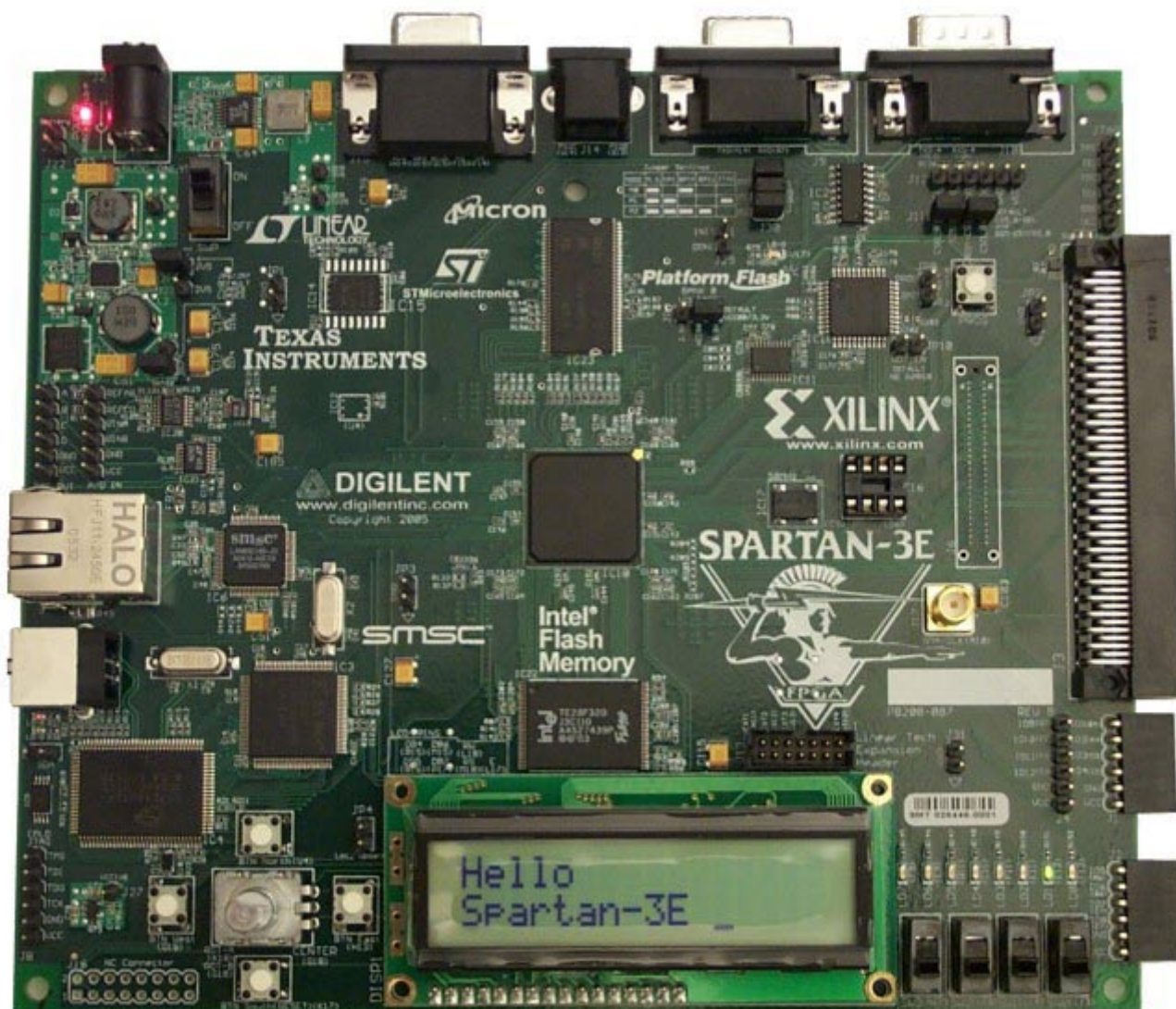


图 1-1、Xilinx Spartan-3E 初学者开发板

Spartan-3E 初学者开发板是一款全面的开发板解决方案，Spartan-3E 初学者开发板上使用 50 万门 Spartan 3E FPGA 可以构建 32 位 RISC 处理器和 DDR 接口。开发板上拥有的 Xilinx Platform Flash、USB 和 JTAG 并行编程接口、Intel StrataFlash 及 ST Microelectronics Serial Flash，提供了丰富的存储、配置选择。该开发平台与所有版本的 Xilinx ISE 工具兼容。使设计人员能够立即获得

Spartan-3E 系列的完整平台性能，是一种专门用于学习 FPGA 设计和 EDA 工具的开发、验证平台。

1.1 初学者开发板主要特性

Spartan-3E 初学者开发板的主要特性见图 1-2 所示，包括：

(1) Xilinx 器件：	<div>➤ Spartan-3E FPGA (XC3S500E-4FG320C)</div> <div>➤ CoolRunner-II CPLD (XC2C64A-5VQ44C)</div> <div>➤ 平台闪存 (XCF04S-VO20C)</div>
(2) 时钟：	<div>➤ 50 MHz 晶体时钟振荡器</div>
(3) 存储器：	<div>➤ 128Mb 并行闪存</div> <div>➤ 16Mb SPI 闪存</div> <div>➤ 64MB DDR SDRAM</div>
(4) 连接器和接口：	<div>➤ 以太网 10/100 物理层</div> <div>➤ JTAG USB 下载</div> <div>➤ 两个 9 引脚 RS-232 串行端口</div> <div>➤ PS/2 型鼠标/键盘端口、带按钮的旋转编码器</div> <div>➤ 4 个滑动开关</div> <div>➤ 八个独立 LED 输出</div> <div>➤ 四个瞬时接触按钮</div> <div>➤ 100 引脚扩展连接端口</div> <div>➤ 三个 6 引脚扩展连接器</div>
(5) 显示器：	<div>➤ 2×16 字符 LCD</div>

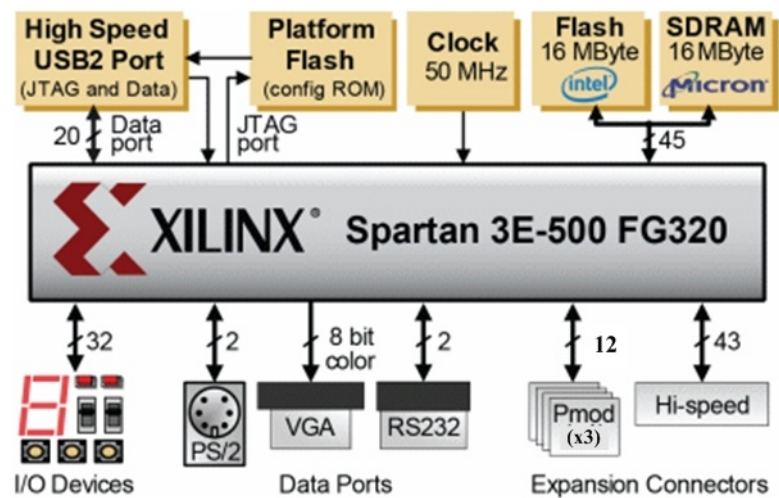


图 1-2、Xilinx Spartan-3E 初学者开发板的关键特性

1.2 Spartan 3E 系列 FPGA 特性

特性	优点
最低的逻辑成本	Spartan 3E FPGA系列90nm工艺器件实现了业界最低的单位逻辑成本。
实现成本最低的连接功能	系统连接功能包括物理并行I/O接口和大宽带所需要的协议。Spartan 3E器件I/O管数支持全部的Select I/O,为快速、灵活的电器接口实现了廉价的高速连接功能，可完成理想的桥接功能，提供如下连接功能解决方案：兼容PCI 32/33和64/66,兼容PCI-X 100MHZ,物理连接和系统元件；18种I/O标准，DDR I/O寄存器，DCM。
成本最低的，高性能DSP解决方案	Spartan 3E FPGA有助于高效构建DSP解决方案，可以基地的价格提供每秒高达91亿次的乘累加（MAC），用于实现紧密的DSP结构，例如MAC引擎及自适应和全并行FIR滤波器等应用。
成本最低的嵌入式处理解决方案	在Spartan 3E FPGA中，可有效构建MicroBlaze 32位软处理器，其成本不超过0.48美元。用户可以使用Spartan 3E FPGA的MicroBlaze 32位软处理器将完整的处理引擎、全部的控制功能与附加支持逻辑集成到单个成本效益型平台内，提供了完整的、带有硬件、软件、工具与设计实例的定制解决方案。
灵活的加载方式	相对于传统的FPGA配置方式，Spartan 3E系列的FPGA率先引入SPI与BPI两种新的Flash配置模式，极大的降低了中小容量FPGA的系统整体成本。

1.3 Spartan 3E 系列 FPGA 主要技术特征

型号	系统 门数	Slice 数目	分布式 RAM容量	块RAM 容量	专用乘 法器数	DCM 数目	最大可用 I/O数	最大差分 I/O对数
XC3S100E	10万	960	15kB	72kB	4	2	108	40
XC3S250E	25万	2448	38kB	216kB	12	4	172	68
XC3S500E	50万	4656	73kB	360kB	20	4	232	92
XC3S1200E	120万	8672	136kB	504kB	28	8	304	124
XC3S1600E	150万	14752	231kB	648kB	36	8	376	156

关于 Xilinx Spartan-3E 初学者开发板详尽的信息，请参考 Xilinx 公司编写的《**Spartan-3E FPGA Starter Kit Board User Guide UG230 (v1.2) January 20, 2011**》。

2、Xilinx ISE Design Suite 10.1 简介



图 2-1、Xilinx ISE Design Suite 10.1 sp2 启动窗口

2.1 ISE Design Suite 10.1 综述

目前，FPGA 设计人员希望设计工具不仅支持先进的生产工艺（65nm），还要同时提供更好的工具性能、更高的效率 and 更丰富的功能，更快实现设计时序收敛和设计反复，快速解决时序以及低功耗等问题。此外，由于工程浩大，必须通过团队合作来完成设计，因此要求设计工具满足团队设计所有要求，通过一个集成常见应用环境的工具来提高团队生产力，并通过片上系统 FPGA 促进真正的系统级解决方案。鉴于此，Xilinx 推出了新一代 ISE Design Suite 10.1 版设计套件，从正面解决 FPGA 设计师所面临的严峻挑战，并且第一次提供了一个统一了逻辑、DSP 以及嵌入式应用设计人员需要的解决方案。ISE Design Suite 10.1 为设计的每一步提供了直观的生产力增强工具，覆盖从系统设计探索、软件开发和基于 HDL 硬件语言设计、直到验证、调试和 PCB 设计集成的全部设计流程。

2.2 ISE Design Suite 10.1 的创新特性

在过去的几年内，ISE 设计工具一直被用户评为业界最佳解决方案，ISE Design Suite 10.1 继承了 ISE 以前版本的全部优点；此外，它还具备以下 8 个创新特点，为大规模、复杂 FPGA 设计提供更高的性能和更高的生产力。

1、一个套件统一提供全面的客户解决方案

和以往版本相比，ISE Design Suite 10.1 的最大特点就是融合了 Xilinx 公司发布的所有软件包，为不同用户提供了统一的开发平台，同时支持逻辑、DPS 和嵌入式设计的全面设计环境，且具备完全的互操作能力。同时，协调提供了完整的客户解决方案，无缝集成了系统级设计、IP 核、RTL 设计、功能验证、RTL 综合、布局布线、功率分析、ChipScope 调试、嵌入式系统软硬设计以及完善的第三方 EDA 工具。通过电子化交付流程保证用户快速方便地获得所有产品的最新更新和评估。更为重要的是，ISE Design Suite 10.1 提供了一个可定制的环境，可通过定制来适合

设计人员的不同需要；无论用户选择何种定制方案，都通过一个单一序列号来管理，无须像以前版本中的 ISE Foundation、EDK 以及 ChipScope 等采用独立的序列号控制。

2、编译速度提高两倍

ISE Design Suite 10.1 以平均运行速度提高两倍的特性极大地加快了设计实施速度，使得设计人员可在一天内完成多次反复设计。这对于团队设计是非常重要的，更快的运行速度极大地节约了开发时间，加快了产品的上市速度。

3、SmartXplorer 技术提供的设计性能提升高达 38%

ISE Design Suite 10.1 的另一个重要意义是及时采用了 SmartXplorer 技术，这一技术专门为解决设计人员所面临的时序收敛和生成力这两大艰巨挑战而开发。SmartXplorer 技术支持在多台 Linux 主机上进行分布式处理，可在一天时间内完成多次实现过程。通过分布式处理和多种实施策略来确定最优策略，性能（最高时钟频率）可以提升多达 38%。同时，SmartXplorer 技术还为用户利用独立时序报告监控每个运行实例提供了相应的工具，通过多组策略支持进入更广泛深入的实现探索。此外，还通过对主流和大规模密度器件进行算法优化和微调，改善大规模（DSP48、块 RAM）的布局，利用总线敏感的 IO 布局工具，将总线布置在一起，使性能平均再提高 8%。

4、集成的 PlanAhead Lite 提供了终极生产力

集成的 PlanAhead Lite 工具，为用户提供了强大的布局规划和分析功能，并提供动态局部重配置的功能。

首先，简化管理目标 FPGA 和 PCB 之间接口的复杂度，并提高用户设计的性能，当与 ISE 结合使用时，可获得 30% 的性能提升。同时，支持设计分析和布局规划，为用户提供了可视化显示关键路径和布局规划以提高性能。

其次，PlanAhead 技术支持在设计早期阶段智能地实现管脚定义（PinAhead），从而避免了在设计后期经常发生的与引脚布局相关的修改。在过去，这种修改通常需要利用交互式引脚布局才能完成设计规模检查。在 PlanAhead 工具中，引脚分配完成后，还可使用逗号分割值（CSV）文件或通过 VHDL 或 Verilog 头文件输出 IO 端口信息。

第三，提供了从前端到后端的动态局部重配置的简化方案，动态局部重配置技术是 FPGA 领域的最新技术之一。在 FPGA 运行时，通过 JTAG 或 SelectMAP（ICAP）重新配置部分区域，而不影响非重配置区域的正常工作。

5、添加了新的基于策略的设计实施

ISE Design Suite 10.1 推出的基于策略的设计，进一步简化了确定最优实现设置的过程，让尝试优化的工作由工具来完成，不必用户输入复杂的设计。设计人员可规定和设置自己独特的设计目标，可以是性能最大、优化器件利用、降低动态功耗或者是实施时间最短。利用这一资源面积优化策略，逻辑资源利用情况平均可减少 10%。

6、与第三方 EDA 厂家广泛合作，可提供更优化的验证能力

ISE Design Suite 10.1 还受益于 Xilinx 公司同业界领先 EDA 供应商的良好合作，通过广泛联合提供更好的设计验证能力。Xilinx 公司和 Mentor Graphics 公司的强强联合，提供了业界第一个 IEEE IP 加密硬 IP 模型，使运行速度缩短多达一倍。新的性能优化 BRAM、DSP 以及 FIFO 仿真模型进一步将 RTL 仿真运行时间缩短了一倍。由于在设计中，有 80%左右的时间是用来验证设计的，因此上述改进无疑加快了产品面市时间。

7、简化系统设计

为帮助用户更快实现嵌入式和 DSP 设计，ISE Design Suite 10.1 还对 Xilinx 嵌入式和 DSP 工具进行了进一步的易用性简化。例如：统一的互操作性保证了用户可以在 ISE Design Suite 10.1 中容易地增添 System Generator 模块。EDK 和 System Generator 技术之间不同工具的集成得到进一步的增强，提供了集成领域专用的设计环境，可在 System Generator 中导入/导出 EDK 项目用于硬件协同仿真、方便地将 DSP 设计从 System Generator 设计集成到 ISE 中以及利用 System Generator 自动生成用于 EDK 的 DSP 加速器，为同时涉及嵌入式和信号处理的更复杂 FPGA SoC 设计提供支持。

8、增强功率分析和优化功能

业界研究表明，满足功率预算是 FPGA 设计人员面临的一项越来越大的挑战，特别是工艺几何尺寸的不断缩小进一步加剧了这一问题。ISE Design Suite 10.1 为用户提供了在设计过程中尽早分析功率要求的功能，同时还可以在设计过程中优化动态功率。第二代 XPower 功率分析工具提供了改善的用户接口，按照模块、结构层次、电源轨和使用的资源来分析功率更为容易，因此进一步增强了功率估算功能。信息可以通过文本和 HTML 报告格式给出。与其它逻辑供应商提供的静态估算网页相比，这是一项巨大进步，同时在提供准确的功耗信息方面是一个飞跃。ISE Design Suite 10.1 提供了便捷全面的功率优化功能，利用集成的“功率优化设计目标”功能，用户可以简单地完成功率优化流程。通过映射和布局布线算法的改进，对于采用 65nm Virtex-5 器件和 Spartan-3 Generation FPGA 的设计动态功率平均可降低 10%和 12%。

2.3 ISE Design Suite 10.1 主要组件

ISE Design Suite 涉及了 FPGA 设计的各个应用方面，包括逻辑开发、数字信号处理系统以及嵌入式系统开发等 FPGA 开发的主要应用领域，主要包括 ISE Foundation、嵌入式开发套件(EDK)、

System Generator、AccelDSP 综合工具、ChipScope Pro 分析仪、PlanAhead 设计和分析工具等组成部分，其完整的开发功能如图 2-2 所示。

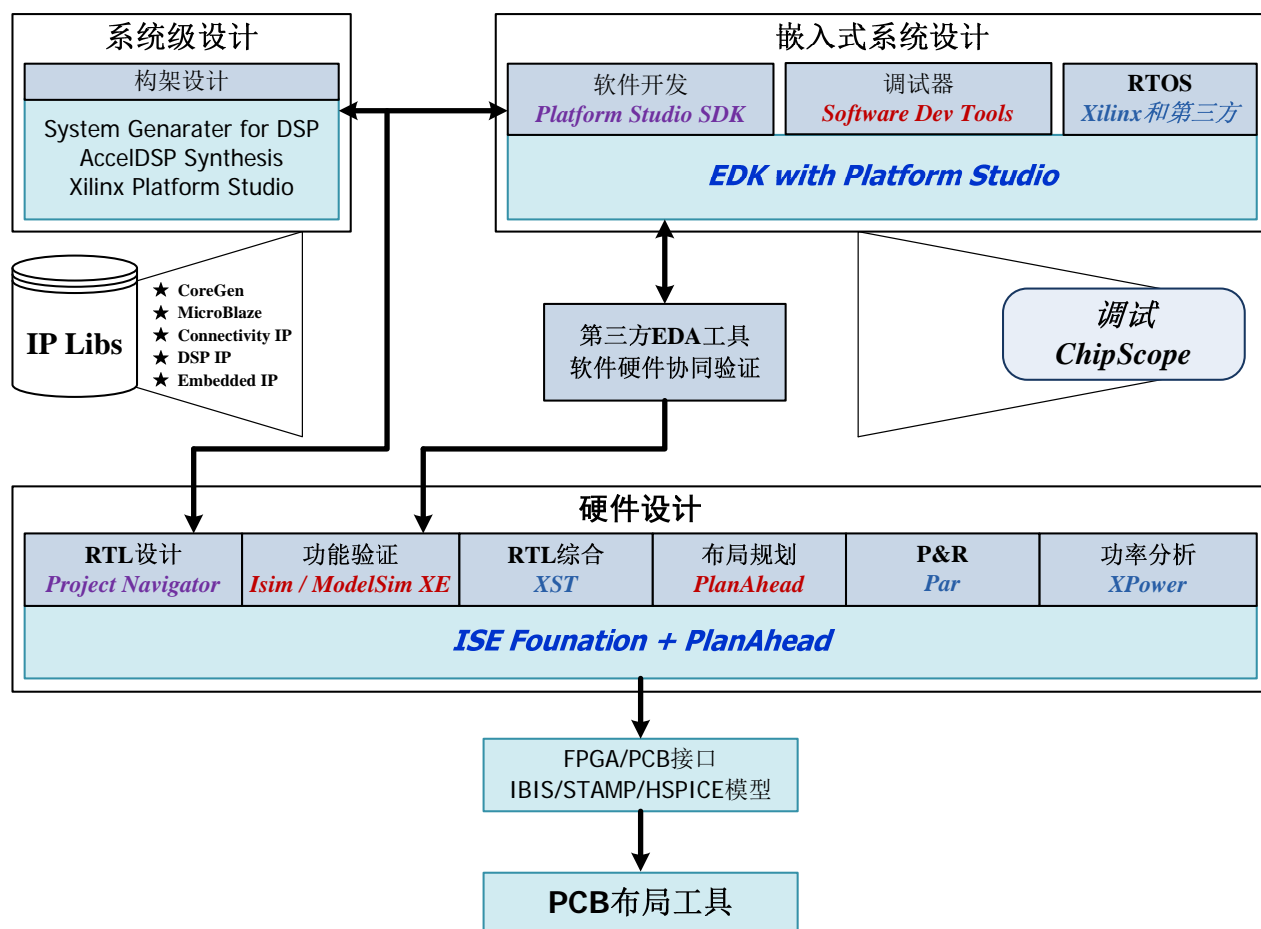


图 2-2 ISE Design Suite 的完整功能

2.3.1 ISE Foundation

ISE Foundation 软件是 Xilinx 公司推出的 FPGA/CPLD 集成开发环境，不仅包括逻辑设计所需的一切，还具有大量简便易用的内置式工具和向导，使得 I/O 分配、功耗分析、时序驱动设计收敛、HDL 仿真等关键步骤变得容易而直观。

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。

（1）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（ISE Text Editor），用于原理图编辑的工具 ECS（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。

（2）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 Synplify 公司的 Synplify，实现无缝链接。

(3) 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。

(4) 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。

(5) 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 2-1 所示。

表 2-1 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 Bencher	MAP	BitGen
StateCAD 状态机编辑器	(Synplify	(ModelSim)	Place and Route	IMPACT
Core Generator	LeonardoSpectrum)		Xpower	
Constraint Editor				

2.3.2 EDK 开发工具

嵌入式系统包括硬件开发和软件开发两部分, 因此嵌入式开发工具指可生成硬件平台, 并能编辑、编译、链接、加载和调试高级编程语言 (通常是 C 或 C++), 最终将其运行在处理器引擎上的综合开发工具。EDK 就是 Xilinx 公司推出的基于 FPGA 的嵌入式开发工具, 包括了嵌入式硬件平台开发工具 (Platform Studio)、嵌入式软件开发工具 (Platform Studio SDK)、嵌入式 IBM PowerPC 硬件处理器核、Xilinx MicroBlaze 软处理器核、开发所需的技术文档和 IP, 为设计嵌入式可编程系统提供了全面的解决方案。

EDK 10.1 版还包括了最新的 IP 内核以优化系统设计。同时还包括了 SPI、DDR2、DMA、PS2 和支持 SGMII 的三模式以太网 MAC 等外设, FlexrayTM 外设选项, 以及用于 DMA 的 PCI Express 驱动支持。改进后的多端口存储器控制器以及存储器接口生成器 (MIG) 工具为存储密集应用提供了更为强大和丰富的接口选择。此外, 对软核处理器 MicroBlaze 的内核 IP 进一步优化和更新, 从而可以提供更大的缓存接口灵活性。

此外, 在利用 Virtex-5 FXT 平台进行嵌入式处理系统架构开发和编程的过程中, EDK 10.1 对其进行了进一步的简化。首先, 添加了自动设计向导, 为设计人员实施高性能 128 位处理器局域总线 (PLB) 提供了一步步的指导, 使得配置支持共享式和点到点系统连接非常简便。其次, 提供了新的辅助处理器单元控制器 (APU) 工具, 对 PowerPC 440 处理器模块提供协处理支持。

APU 还可以用来建立与高速 FPGA 硬件的直接接口，完成 PowerPC 440 处理器代码映射并提供支持软件和硬件优化的分析。

2.3.3 DSP 工具

目前的 FPGA 芯片不再扮演胶合逻辑的角色，而成为数字信号处理系统的核心器件。在芯片内，不仅包含了逻辑资源，还有多路复用器、存储器、硬核乘加单元以及内嵌的处理器等设备，并且还具备高度并行计算的能力，使得 FPGA 已成为高性能数字信号处理的理想器件，特别适合于完成数字滤波、快速傅立叶变换等。但遗憾的是，FPGA 并未在数字信号处理领域获得广泛应用，主要原因就是：首先，大部分 DSP 设计者通常对 C 语言或 MATLAB 工具很熟悉，却不了解硬件描述语言 VHDL 和 Verilog HDL；其次，部分 DSP 工程师认为对 HDL 语言在语句可综合方面的要求限制了其编写算法的思路。基于此，Xilinx 公司推出了简化 FPGA 数字处理系统的集成开发工具 DSP Tools，快速、简易地将 DSP 系统的抽象算法转化成可综合的、可靠的硬件系统，为 DSP 设计者扫清了编程的障碍。DSP Tools 主要包括 System Generator 和 Accel DSP 两部分，前者与 Mathworks 公司的 Simulink 实现无缝链接，后者主要针对 c/m 语言。

1、System Generator

System Generator 是 Xilinx 公司的系统级建模工具，在很多方面扩展了 MathWorks 公司的 Simulink 平台，提供了适合硬件设计的数字信号处理（DSP）建模环境，加速、简化了 FPGA 的 DSP 系统级硬件设计。System Generator 提供了系统级设计能力，允许在相同的环境内进行软、硬件仿真、执行和验证，并不需要书写 HDL 代码。此外，System Generator 工具还能完成高级提取，自动编译生成 FPGA 代码，也可通过低级的提取，对 FPGA 的底层资源进行访问，从而实现高效率 FPGA 设计构建。目前，基于 System Generator 的设计方法已在复杂系统实现中展现了强大的潜能，必将成为未来流行的 FPGA 开发技术之一。

2、Accel DSP

AccelDSP 是一款第三方综合软件，可将 MATLAB 浮点算法转换成为可综合 RTL 代码。Xilinx AccelDSP 是目前业界唯一能够将 MATLAB 浮点算法转换成为可综合 RTL 代码的开发工具。该工具可自动地进行浮点-定点转换，生成可综合的 VHDL 或 Verilog 代码，并创建用于验证的测试平台，同时还可以生成定点 C++ 模型或由 MATLAB 算法得到 System Generator 块。AccelDSP 综合工具是 Xilinx XtremeDSP 解决方案的重要组成部分。AccelDSP 产品体系由两个主要模块构成：AccelDSP 综合器和 AccelWare IP。其中，AccelDSP 综合器是一个综合和验证的环境，可以自动将 MATLAB 浮点代码转换成为定点代码，然后生成可综合的 VHDL 或 Verilog 代码，为设计者提供了验证算法和实现算法的功能。

2.3.4 ChipScope Pro

逻辑分析仪（Logic Analyzer）是 FPGA 调试阶段不可缺少的工具，但是传统逻辑分析仪有两个弊端：首先价格昂贵；其次需要使用大量探头，不仅不合实际且操作麻烦。Xilinx 公司为了解决用户的这两个难题，推出了在线逻辑分析仪（ChipScope Pro），通过软件方式为用户提供稳定和方便的解决方案。该在线逻辑分析仪不仅具有逻辑分析仪的功能，而且成本低廉、操作简单，因此具有极高的实用价值。ChipScope Pro 既可以独立使用，也可以在 ISE 集成环境中使用，非常灵活，为用户提供方便和稳定的逻辑分析解决方案，支持 Spartan 和 Virtex 全系列 FPGA 芯片。

ChipScope Pro 将逻辑分析器、总线分析器和虚拟 I/O 小型软件核直接插入到用户的设计当中，可以直接查看任何内部信号或节点，包括嵌入式硬或软处理器。信号在操作系统速度下或接近操作系统速度下被采集，并从编程接口中引出，再将采集到的信号通过 ChipScope Pro 逻辑分析器进行分析，从而为设计解放了更多的引脚。利用 FPGA 的可重编程性能，可以在几分钟或几小时内确定设计问题并修改设计；内置的软件逻辑分析器可以用来识别设计问题并进行调试，包括高级触发、过滤和显示选项，无需重新综合即可改变探针指向；可利用远程控制（从办公室到实验室，或在全球范围内）通过互联网连接进行调试；此外还包括 Agilent 科技推出的、用于实现功能强大的验证功能的逻辑分析器可选配件，可以探测包括从 FPGA 内部到板上任何地方的交叉互联信号。

2.3.5 PlanAhead

PlanAhead 工具简化了综合与布局布线之间的设计步骤，能够将大型设计划分成较小的、更易于管理的模块，并集中精力优化各个模块。此外，还提供了一个直观的环境，为用户设计提供原理图、平面布局规划或器件图，可快速确定和改进设计的层次，以便获得更好的结果和更有效地使用资源，从而获得最佳的性能和更高的利用率，极大地提升了整个设计的性能和质量。PlanAhead 的主要功能包括：

1、轻松实现引脚规划的 PinAhead 技术

PlanAhead 包含 PinAhead 技术，可以帮助用户更好地处理引脚分配的复杂性问题。PinAhead 提供了一个以全自动或半自动方式将 I/O 端口分配到物理封装引脚上的环境。

2、整合了 ExploreAhead

ExploreAhead 是一种实现探索工具，通过管理多个实现运行。ExploreAhead 允许用户根据他们指定的策略或者作为工厂默认方法发售的预定策略，执行多个实现操作。在 Linux 环境下，ExploreAhead 具有在远程主机上运行设计的能力。

3、可完成基于模块的增量设计

PlanAhead 提供了层次化、基于模块的、模块化和增量设计方法，让设计者只需改变一部分设计，而保持其它部分的完整性，从而缩短了设计迭代。即使是在经常改动的情况下，它也能让用户保持所需的性能。

4、提高设计的信号完整性

PlanAhead 提供了检查加权平均 SSO (WASSO) 分析限制的功能。这使得设计者能够更轻松的限制 FPGA 输出处的触地反弹数量，并能够防止发生 FPGA 引起的其它器件的操作失误。

5、支持部分重配置

PlanAhead 简化了针对部分重配置的、功能强大但复杂的设计流程。部分重配置是一种独特的方法，可以在静态部分仍然工作的情况下改变设计的动态部分。部分重配置可以让用户减小设计的尺寸、重量、成本和功耗。

6、基于 TimeAhead 的延时估计

TimeAhead 是一种灵活的、集成到 PlanAhead 中的时序分析器，它让用户在进行布局和布线之前就可以估计布线延迟。采用基于 PlanAhead 模块的方法，可在完成布局和布线的时候，提高时序估计的准确度。

ISE Design Suit 10.1 是 Xilinx 推出的新一代设计工具，和以前的开发软件相比，有许多创新设计和重大突破，为用户提供统一的整体设计工具，涵盖 FPGA 逻辑、嵌入式和 DSP 开发，能够突破性提升设计生产力、性能和功耗。ISE Design Suit 10.1 支持操作系统包括：Windows XP、及多种发行版的 Linux。

3、ISE 的 FPGA 开发流程

在 ISE Design Suite 下的 FPGA 开发流程如下图所示：

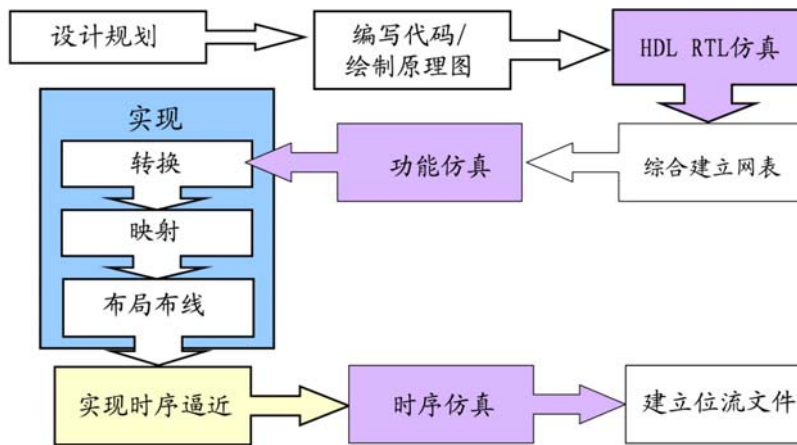


图 3-1、FPGA 开发设计流程

- ✓ 设计规划：根据设计的要求，设计系统功能框图，规划各个功能模块；
- ✓ ISE 有两种设计输入方法：硬件描述语言 HDL 或原理图；
 - ——结构向导和核生成器 Core Generator 可以辅助设计输入。
- ✓ 不管你采用何种设计方法，都需要一个工具来生成 EDIF 网表，以便对 Xilinx 的 FPGA 编程
 - ——适用的综合工具有 Synopsys Synplify、Mentor Graphics Precision RTL、XST
- ✓ 对设计进行仿真使你的设计按照你的设想动作。



图 3-2、对设计进行仿真

- ✓ 一旦建立了网表你就可以来实现设计；
- ✓ 在实现时能产生许多输出文件
 - 报告
 - 时序仿真网表
 - 平面布局文件
 - FPGA 编辑器文件
 - 其它更多

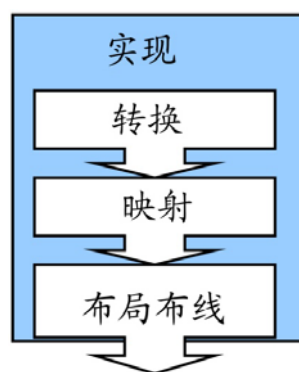


图 3-3、实现一个设计

什么是实现？

- ✓ 不仅仅是指“布局布线”
- ✓ 实现包括很多步骤
 - 转换：将多个设计文件合并为一个网表
 - 映射：将网表中的逻辑符号（门）组装到物理元件（CLB 和 IOB）中；
 - 布局布线：将元件放置到器件中，并将它们连接起来，同时提取出时序数据，并生成各种报告；
 - 每个步骤都会生成一些文件，使得可以使用 Xilinx 的其它工具，例如：Floorplanner、FPGA Editor、Xpower、Multi-Pass Place & Route；

时序逼近流程

时序逼近流程是一个推荐的设计方法，可以帮助设计满足它们的时序目标：

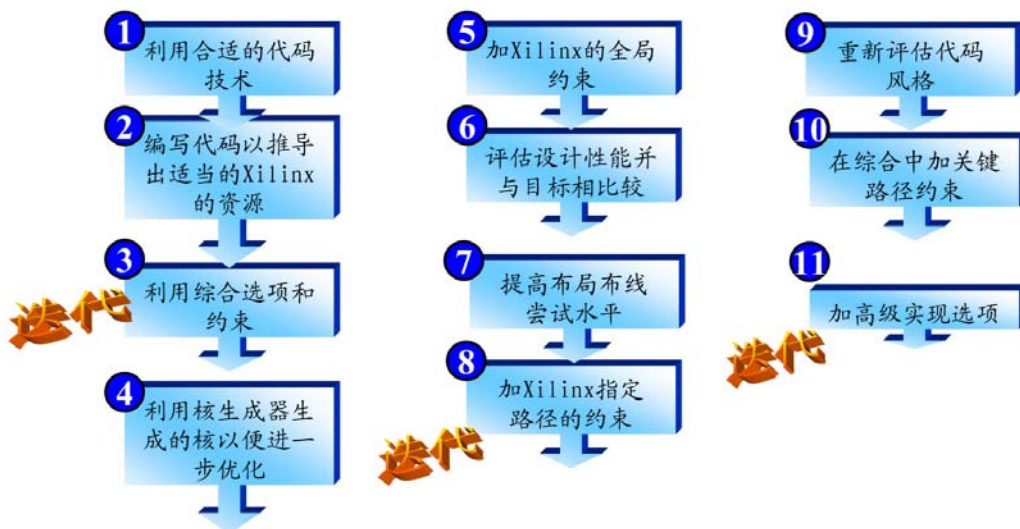


图 3-4、时序逼近流程


下载

- ✓ 一旦一个设计实现完成后，你必须建立一个 FPGA 可以识别的文件：
 - 这个文件叫位流文件，一个 BIT 文件（以 .bit 为扩展名）；
- ✓ 这个 BIT 文件可以被直接下载到 FPGA 中，或者可以被转换为存贮编程信息的 PROM 文件。

4、Step by Step——使用 ISE 设计 FPGA

下面介绍在 Windows XP 操作系统中, ISE Design Suite 10.1 开发系统的基本使用方法, 这里, 以 Verilog HDL 语言作为设计输入。

4.1 启动 ISE

双击桌面的快捷图标 , 或: 【开始】→【所有程序】→【Xilinx ISE Design Suite 10.1】→【ISE】→【Project Navigator】, 启动并进入如图 4-1 所示的 ISE 软件的集成开发环境。

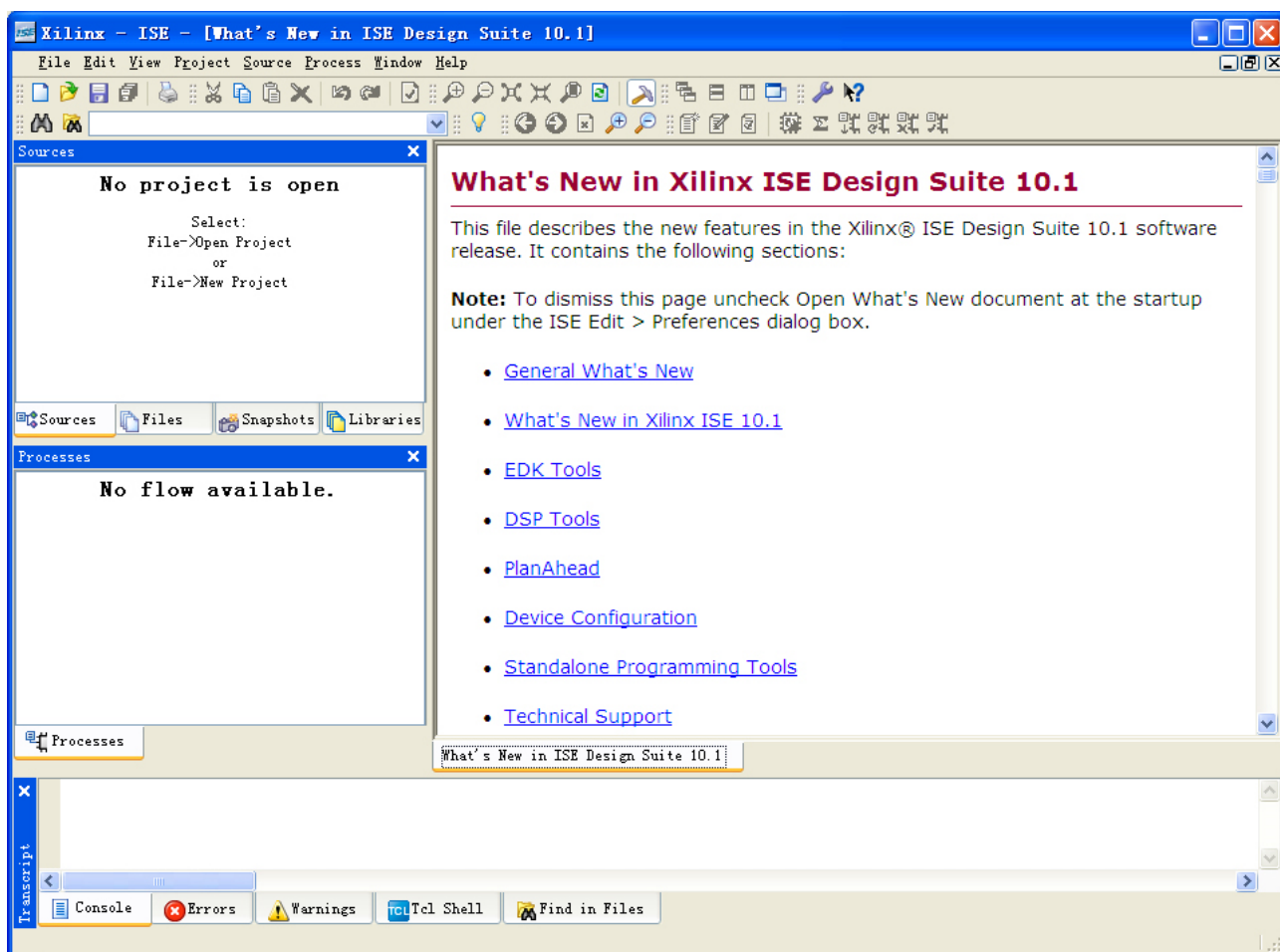


图 4-1、FPGA 的集成开发环境 ISE Design Suite 10.1

ISE Design Suite 10.1 集成开发环境主要包括以下几部分:

- ✓ 项目标题栏: 显示当前正在操作的项目的项目名及其所在的路径;
- ✓ 菜单栏: 命令操作菜单, 为用户提供了一个良好的命令菜单, 所有的命令都包括在这些菜单中;
- ✓ 工具栏: 常用命令组成的快捷工具栏, 为用户提供了一种更快捷的命令操作方法;

- ✓ 项目源文件（Sources in Project）窗口：显示本工程文件的结构，即：显示所有与项目相关的设计文件，使用户能够简单方便地对各文件进行操作。包括所有设计文件：原理图、HDL 源文件、测试文件。
- ✓ 当前源文件进程窗口（Processes for Current Source）：显示能够对项目源文件窗口中所选定的源程序执行的各种处理。包括：
 - 综合 HDL 设计或原理图设计
 - 仿真测试
 - 转换
 - 映射
 - 布局布线
 - 实现器件编程
 - 报告生成
 - 等等处理过程
- ✓ 编辑窗口：源程序文件的编辑修改；
- ✓ 信息输出窗口：显示处理过程运行结果的全部信息、错误提示信息、警告提示信息等。

4.2 建立工程(Project)开始第一个设计

4.2.1 设计目标

在这个设计中，要求：

利用 **Spartan-3E** 初学者开发板的上 4 个滑杆控制开关控制开发板上的 8 个 LED 的“点亮”和“熄灭”。**Verilog** 模块的功能是读入开发板的 4 个滑杆开关的状态，并把它输出到 8 个发光二极管。

4 个滑杆开关和 8 个 LED 发光二极管如图 4-2 所示。

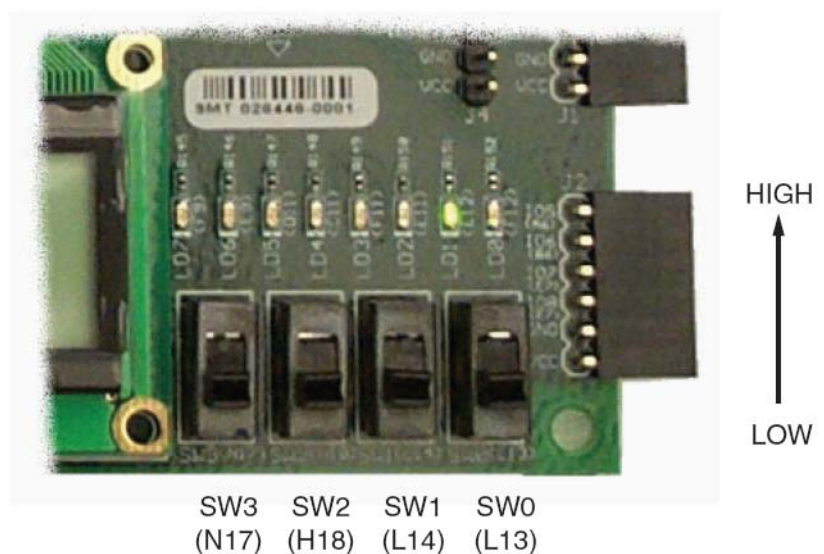


图 4-2、Spartan-3E 初学者开发板的 4 个滑杆开关和 8 个 LED

Step 1

ISE 使用工程（project）来管理项目及其所属项目文件。因此，在 ISE 下的任何设计必须从新建工程文件开始。

单击菜单栏的【File】选项，在弹出的下拉菜单中选择【New Project】命令，这时会出现如图 4-3 所示的 New Project 对话框。在 Project Name 栏内输入该工程的文件名，如：SwitchLED。可以点击 Project 编辑框旁的浏览按钮，选择该工程若存放的路径。

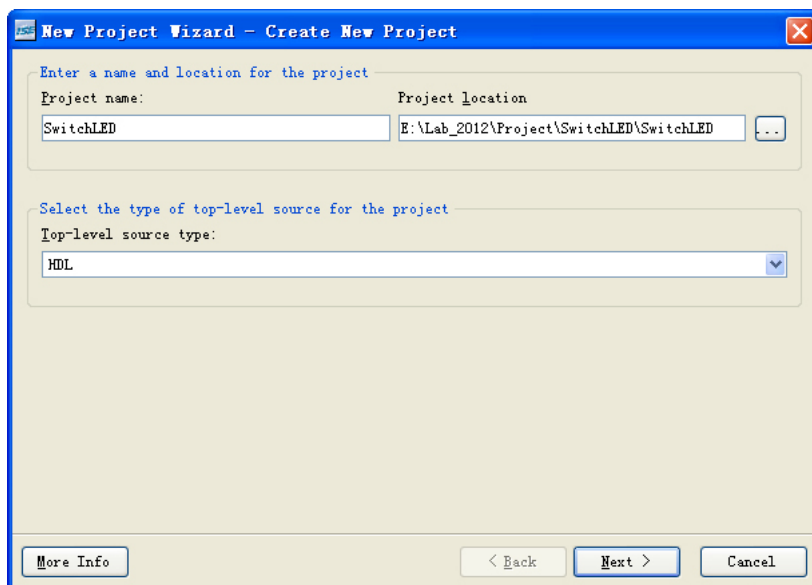


图 4-3、新建工程对话框

Step 2

然后，单击 **Next >**，进入 FPGA 器件属性设置对话框，如图 4-4 所示。

在这个对话框中，使用“**Value 下拉列表框**”，对工程文件进行基本的属性设置。其中包括：器件的类型（Product Category）、器件的系列的名称（Family），器件具体型号（Device）、器件的封装（Package）、器件的速度等级（Speed），以及使用的综合工具（Synthesis Tool）、仿真器（Simulator）和首选硬件设计语言（Preferred Language）。

器件的类型和具体型号必须按照所使用的目标器件进行选择。在 Xilinx Spartan-3E 初学者开发板中使用的是 Xilinx 公司的 Spartan3E 系列的 FPAG 芯片 XC3S500E，点击 Value，在弹出的下拉列表中，按照图 4-4 所示，选择相应的器件系列、型号、封装、速度，以及综合工具、仿真器和设计语言。

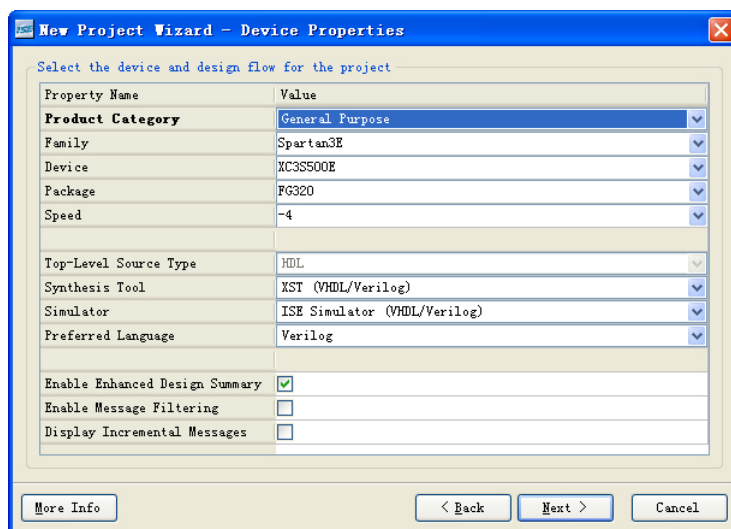


图 4-4 新建工程的 FPGA 器件属性（Device Properties）设置

如果系统中安装了第三方的综合工具，如：Synopsys Synplify Pro、Mentor Graphics Precesion 等，点击 **Synthesis Tool**，可以选择综合工具，这时，会弹出 ISE 软件支持的综合工具，如 XST（ISE 本身自带的综合工具）、Synplify、Precesion 等，这里选择 XST。

Step 3

单击 **Next >**，出现如图 4-5 所示对话框，此时，可以向新建的工程中添加设计源文件；

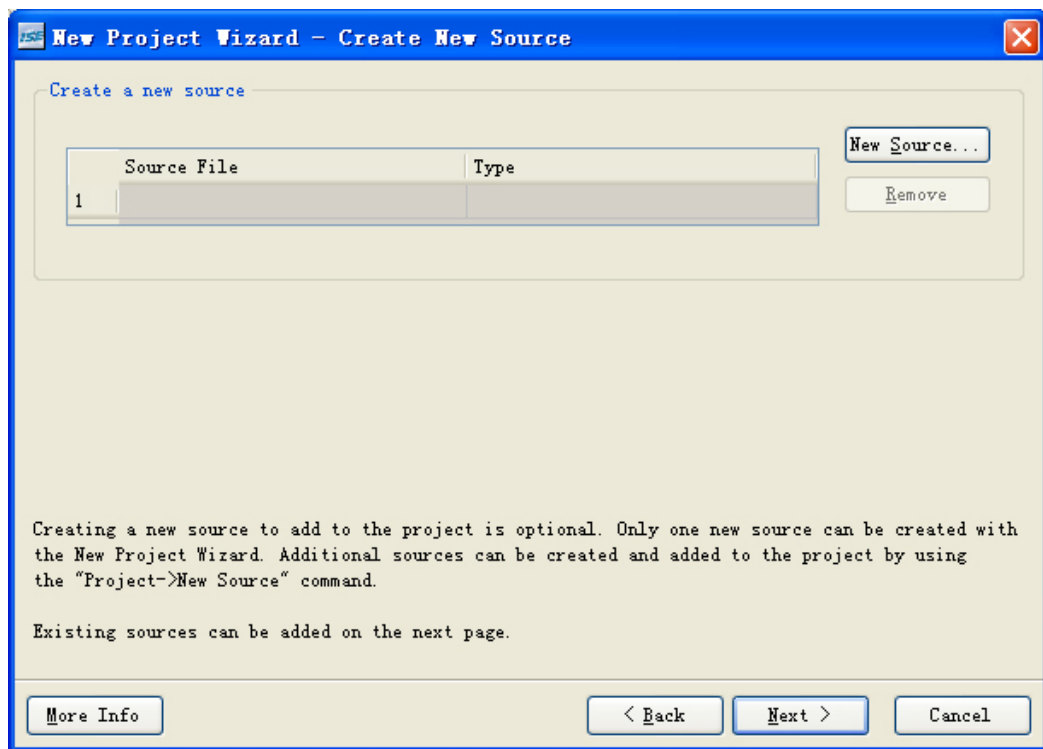


图 4-5 新建工程的创建新的设计源文件

Step 4

单击 **New Source...**，为工程创建新的 Verilog 设计源文件，文件名为：SwitchLED，如图 4-6 所示：

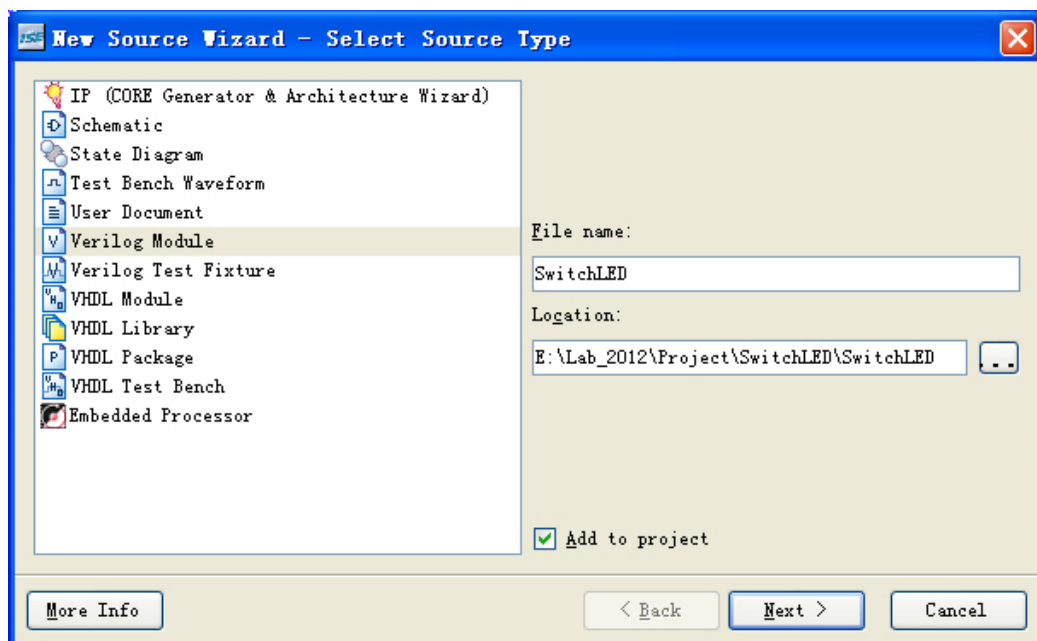


图 4-6 创建新的 Verilog Module 源文件

Step 5

单击 **Next >**，进入 Define Module（Verilog 模块定义）对话框，**这里，定义：**

- 输出端口 LEDOut[7:0]——“Port Name”为 LEDOut，“Direction”为“output”，选择：Bus ☒，MSB 为 7，LSB 为 0；
- 输入端口 SlideSwitch[3:0]：——“Port Name”为 SlideSwitch，“Direction”为“input”，选择：Bus ☒，MSB 为 3，LSB 为 0；

如图 4-7 所示。

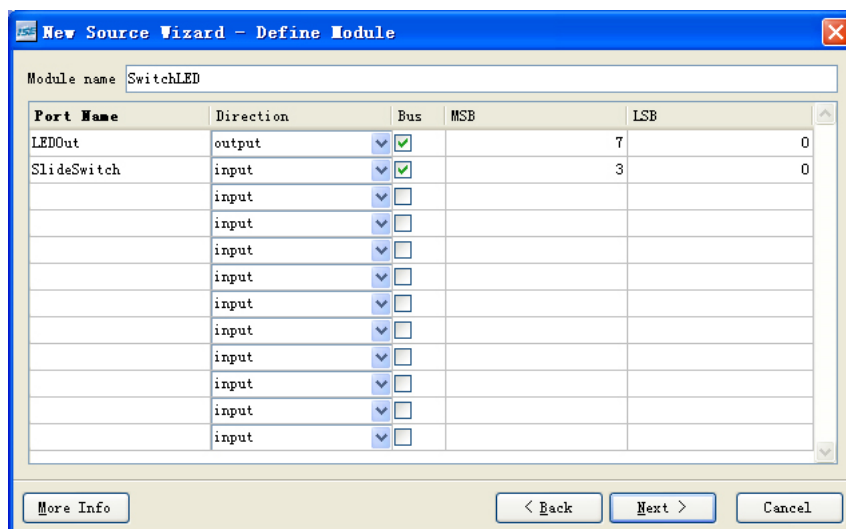


图 4-7 定义新创建的 Verilog Module 输入/输出端口

Step 6

单击 **Next >** ，一个新创建的 Verilog 设计文件（框架）将添加到工程中，如图 4-8 所示：

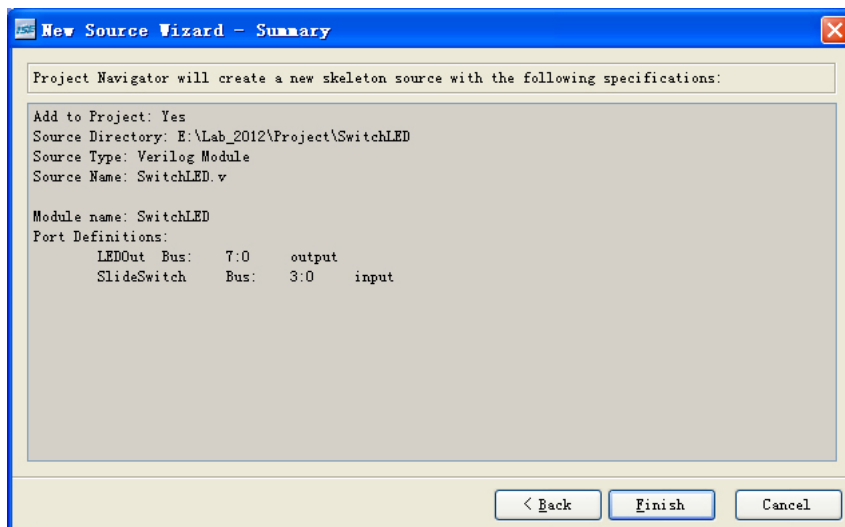


图 4-8 一个新创建的 Verilog 设计文件框架添加到工程中

单击 **Finish** ，出现图 4-9，这是由于工程存放的文件目录没有创建；

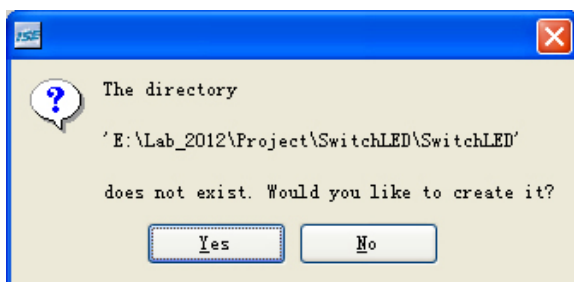


图 4-9 创建工程目录

单击 **Yes** ，出现图 4-10，建立新的工程目录，并将新创建的 Verilog 设计文件 SwitchLED.v 添加到工程中；

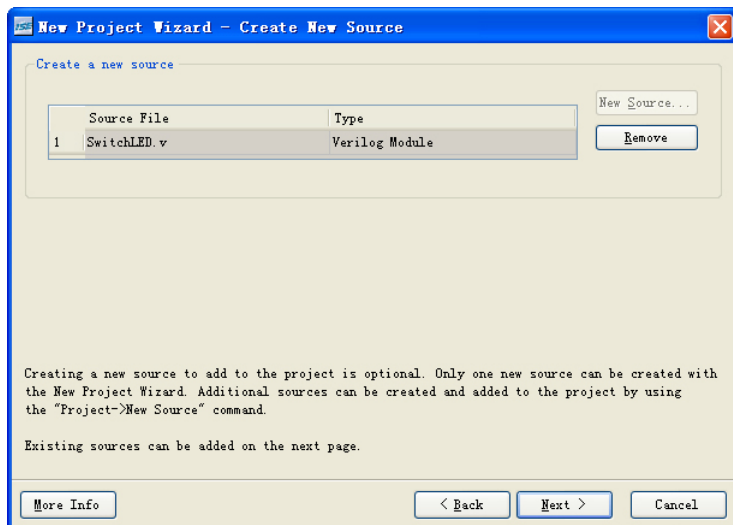


图 4-10 创建一个新的设计源文件

Step 7

单击 **Next >** ，出现添加已有的设计源文件对话框，如图 4-11 所示：

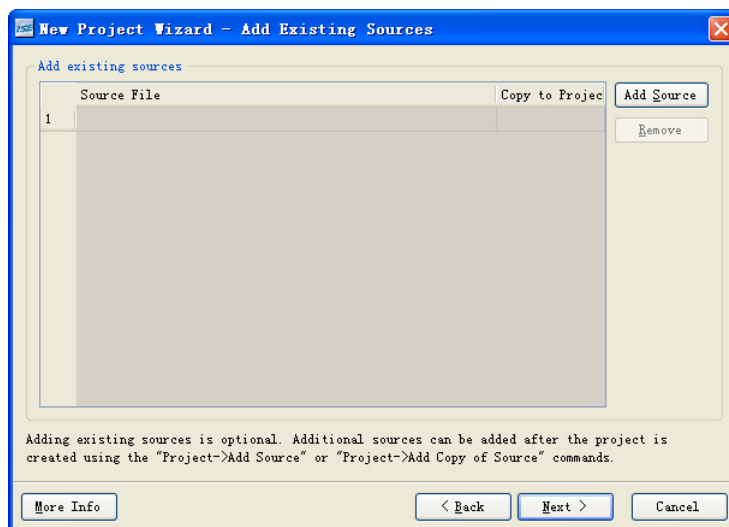


图 4-11、添加已存在的设计源文件对话框

如果已有的 Verilog 设计文件，可以点击 **Add Source** 添加设计源文件，在这个设计练习中，没有已编辑的设计文件，单击 **Next >** ，出现新创建的工程概述对话框，如图 4-12 所示：

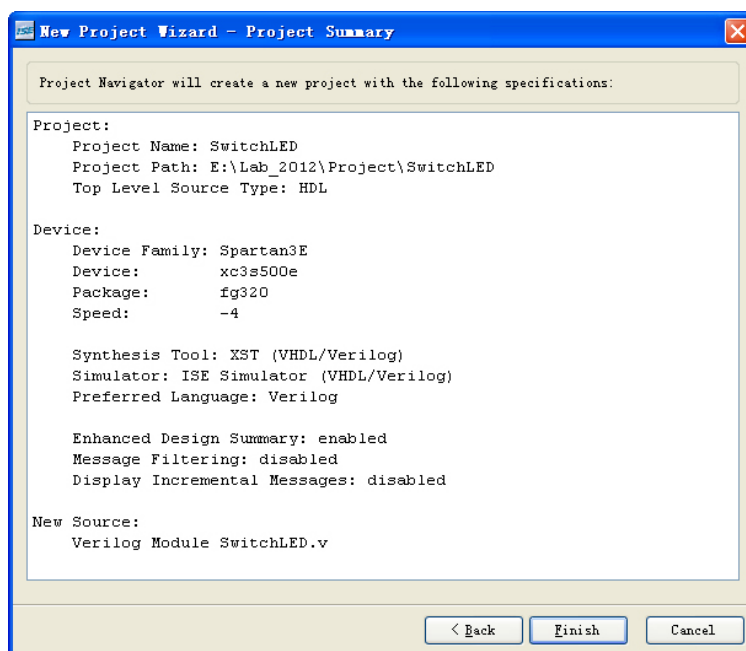


图 4-12、新创建的工程概述对话框

Step 8

点击单击 **Finish** ，完成新建工程，此时的 ISE 如图 4-13 所示。

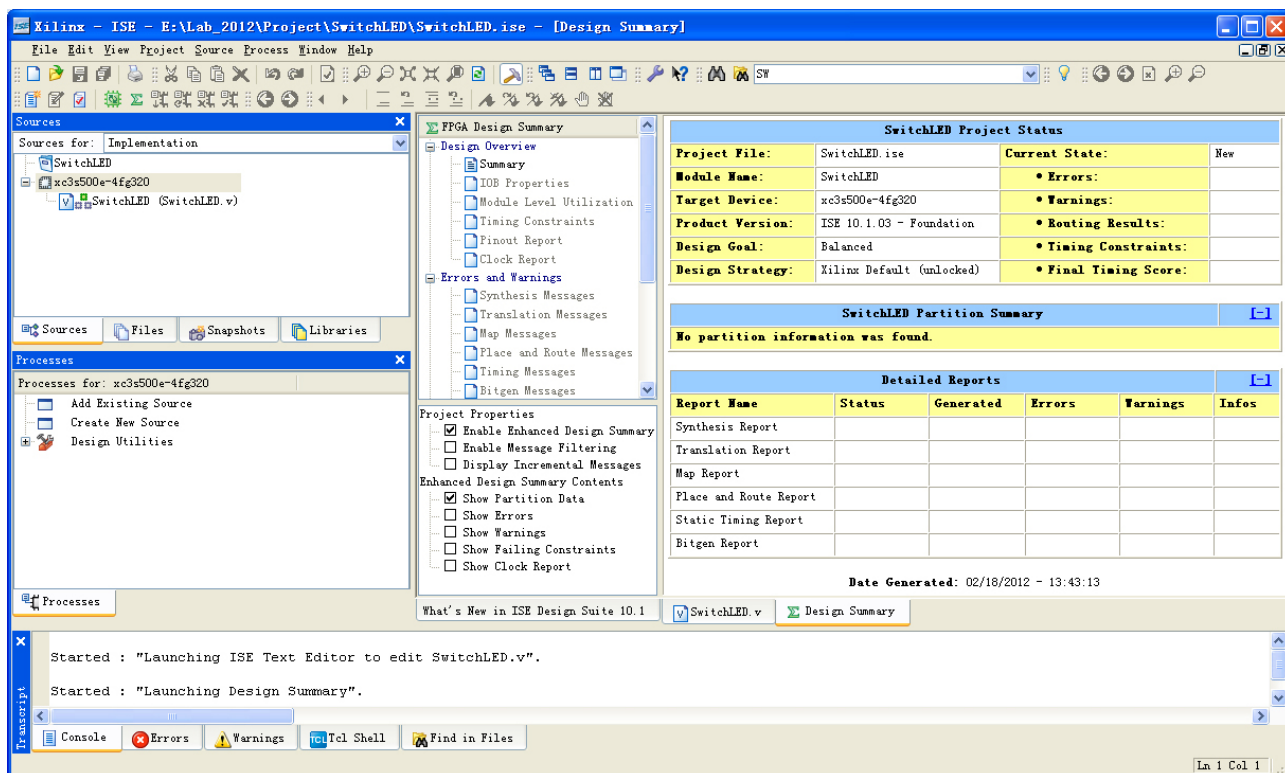


图 4-13、在 ISE 中创建或打开工程

点击 ISE 的 Workspace 上的 Tab 按钮“SwitchLED.v”，可以看到刚刚建立 Verilog 设计文件，如图 4-14 所示，图 4-14 中的 Sources 窗口中，SwitchLED 是工程项目名，xc3s500e-4fg320 是所选用的器件。

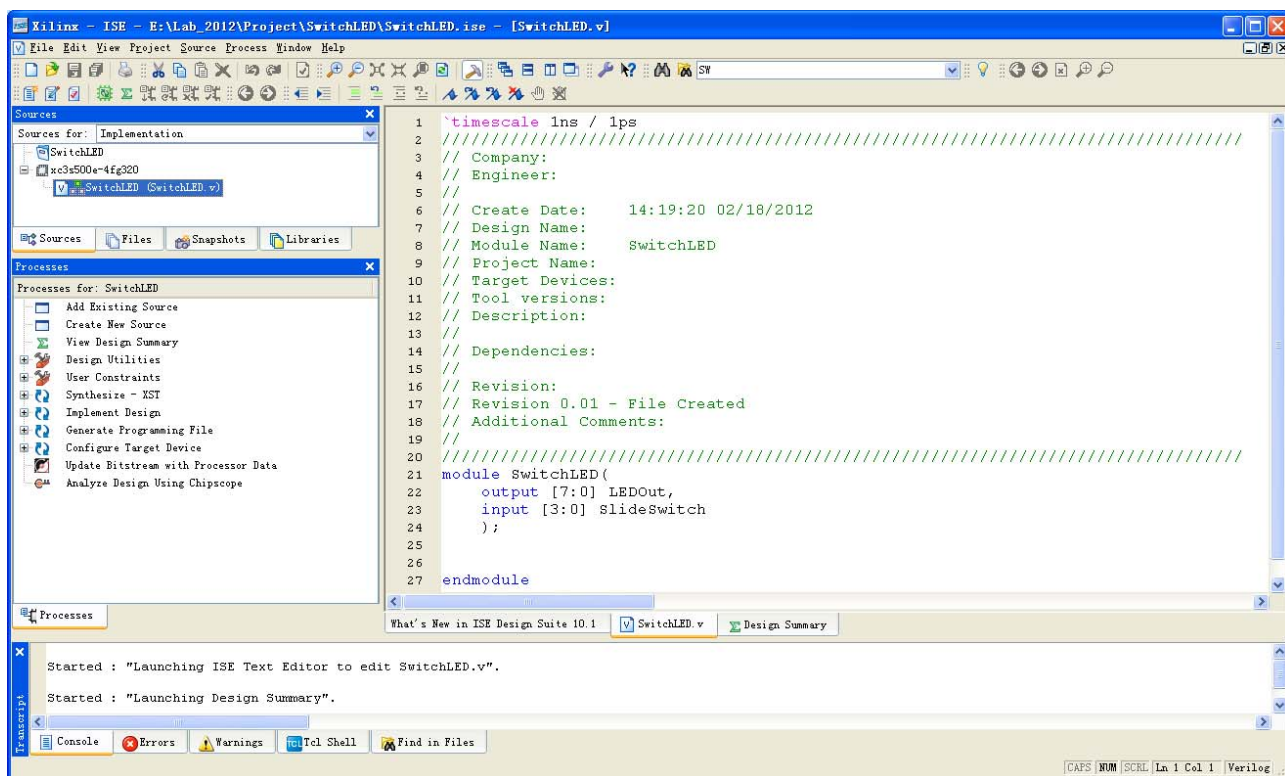


图 4-14、新建的 Verilog 设计文件

Step 9

在如图 4-14 所示的编程窗口中，输入如下代码：

```
// 开始添加 verilog 源代码
reg [7:0] LEDConnect;

always @(*) begin
    {LEDConnect[7], LEDConnect[0]} = {SlideSwitch[0], SlideSwitch[0]};
    {LEDConnect[6], LEDConnect[1]} = {SlideSwitch[1], SlideSwitch[1]};
    {LEDConnect[5], LEDConnect[2]} = {SlideSwitch[2], SlideSwitch[2]};
    {LEDConnect[4], LEDConnect[3]} = {SlideSwitch[3], SlideSwitch[3]};
end

assign LEDOut = LEDConnect;
// 完成添加 verilog 源代码
```

完成整个 Verilog HDL 程序设计，完成代码编写后，Verilog 模块 SwitchLED 如图 4-15 所示，最后，点击工具栏内的保存快捷图标，保存整个工程项目。

```
20 ///////////////////////////////////////////////////
21 module SwitchLED(
22     output [7:0] LEDOut,
23     input [3:0] SlideSwitch
24 );
25 // 开始添加 Verilog 源代码
26 reg [7:0] LEDConnect;
27
28 always @(*) begin
29     {LEDConnect[7], LEDConnect[0]} = {SlideSwitch[0], SlideSwitch[0]};
30     {LEDConnect[6], LEDConnect[1]} = {SlideSwitch[1], SlideSwitch[1]};
31     {LEDConnect[5], LEDConnect[2]} = {SlideSwitch[2], SlideSwitch[2]};
32     {LEDConnect[4], LEDConnect[3]} = {SlideSwitch[3], SlideSwitch[3]};
33 end
34
35 assign LEDOut = LEDConnect;
36 // 完成添加 Verilog 源代码
37 endmodule
```

图 4-15、完成设计后的 Verilog 模块 SwitchLED

Step 10

当完成设计后，在“Source”窗口选择“Source for Implementation”，然后，选中“SwitchLED.v”源文件，此时，“Process”窗口出现用于“Implementation”的各种处理工具，如图 4-16 所示。

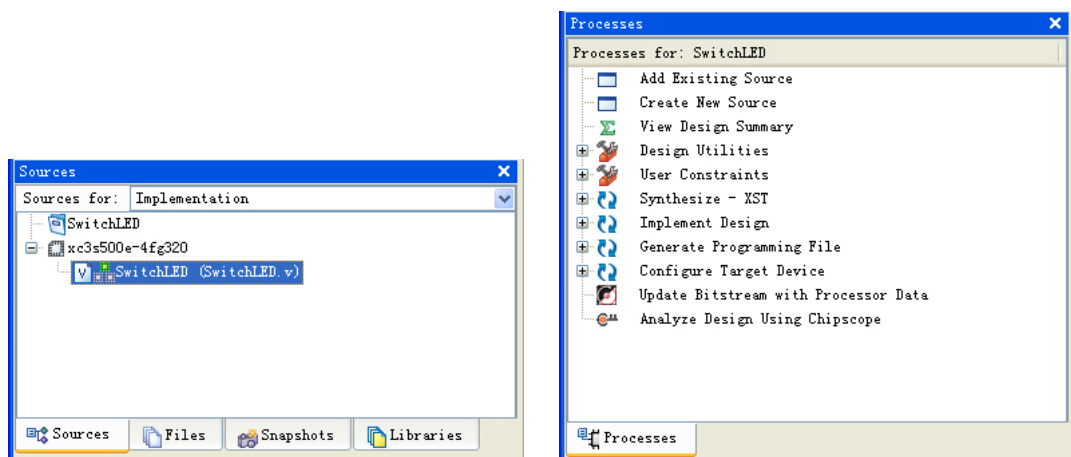


图 4-16、用于设计实现的处理工具

点击“Synthesis - XST”左边的符号⊕，层次展开后，如图 4-17 所示。

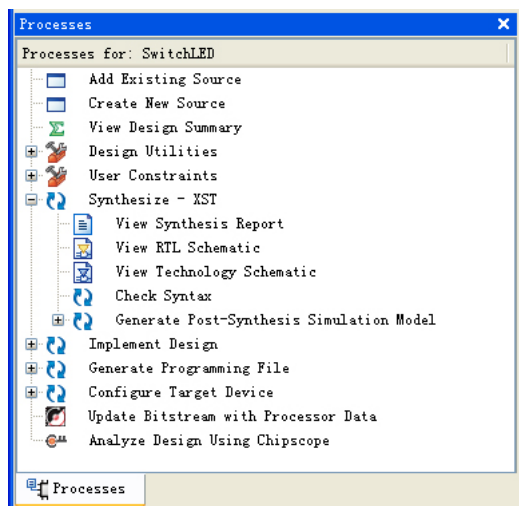


图 4-17、“Synthesis - XST”展开后的“Process”窗口

Step 11

双击“Check Syntax”进行源文件语法检查，如果源文件没有错误，完成检查后，“Process”窗口显示如图 4-18 所示。

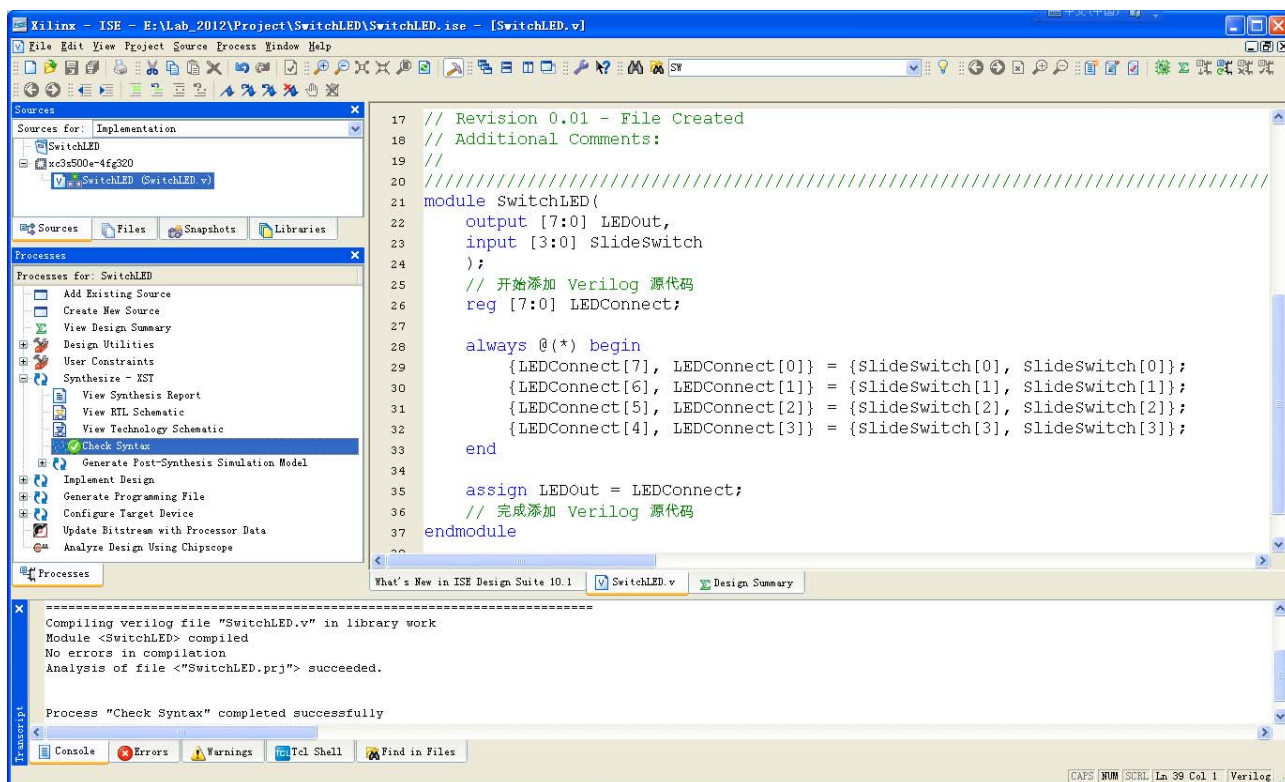


图 4-18、完成语法检查

Step 12 创建用户设计约束文件

用户设计约束文件的后缀是.ucf，所以一般也被称为 UCF 文件。

创建约束文件有两种方法，一种是通过新建文本文件的方式，另一种则是利用“Process”窗口的“User Constraints”工具创建完成。

如果根据 Xilinx 的开发板手册，《**Spartan-3E FPGA Starter Kit Board User Guide UG230 (v1.2) January 20, 2011**》（为了方便描述，以后简称：UG230 手册），采用新建文本文件的方式创建用户约束文件非常简单。

新建一个源文件：

在文件类型中选取“Implementation Constrains File”，在“File Name”中输入“SwitchLED”。单击“Next”，然后，再单击“Finish”按钮完成约束文件的创建。如图 4-19 所示。

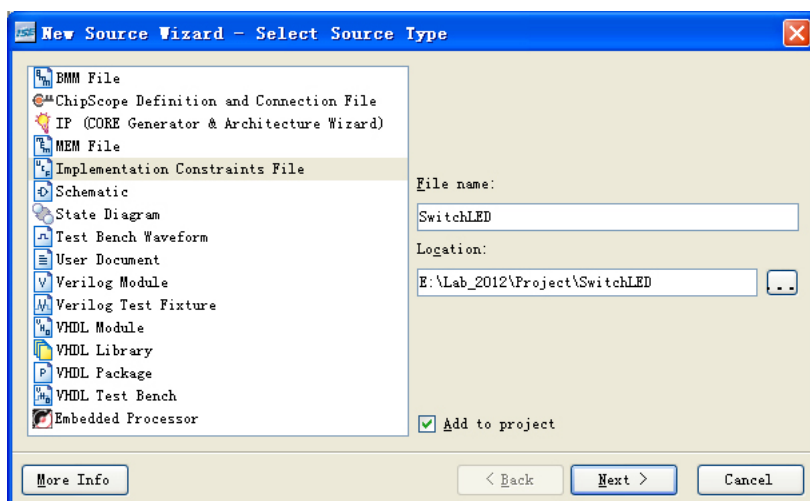


图 4-19、创建用户约束文件

点击“Source”窗口“SwitchLED (SwitchLED.v)”左边的展开符号 \oplus ，选中用户约束文件“SwitchLED.ucf”，此时，“Process”窗口如选图4-20所示，双击“Edit Constraints (Text)”在工作空间打开SwitchLED.ucf，此时为一空白文件。

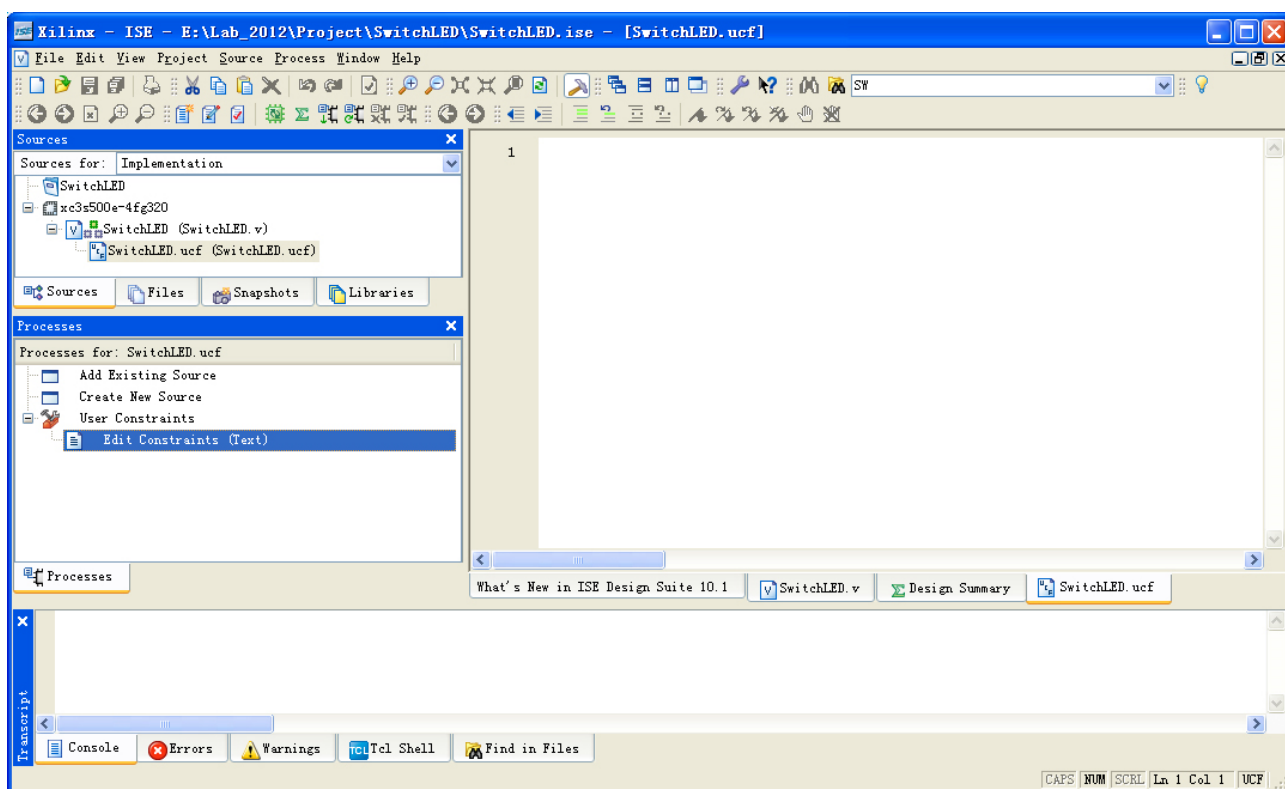


图 4-20

Step 13

参考 UG230手册，在第20页，见 *Figure 2-11: UCF Constraints for Eight Discrete LEDs*,

将下面内容拷贝到刚刚建立的SwitchLED.ucf中：

```
NET "LED<7>" LOC = "F9" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<6>" LOC = "E9" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<5>" LOC = "D11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<4>" LOC = "C11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<3>" LOC = "F11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<2>" LOC = "E11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<1>" LOC = "E12" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<0>" LOC = "F12" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
```

然后，从UG230手册的第16页中，见 *Figure 2-2: UCF Constraints for Slide Switches*，将下面内容拷贝到刚刚建立的SwitchLED.ucf中：

```
NET "SW<0>" LOC = "L13" | IOSTANDARD = LVTTTL | PULLUP ;
NET "SW<1>" LOC = "L14" | IOSTANDARD = LVTTTL | PULLUP ;
NET "SW<2>" LOC = "H18" | IOSTANDARD = LVTTTL | PULLUP ;
NET "SW<3>" LOC = "N17" | IOSTANDARD = LVTTTL | PULLUP ;
```

接下来，编辑SwitchLED.ucf，将其中的“LED”替换为“LEDOut”，将其中的“SW”替换为“SlideSwitch”，此时，SwitchLED.ucf的内容如图4-21所示：

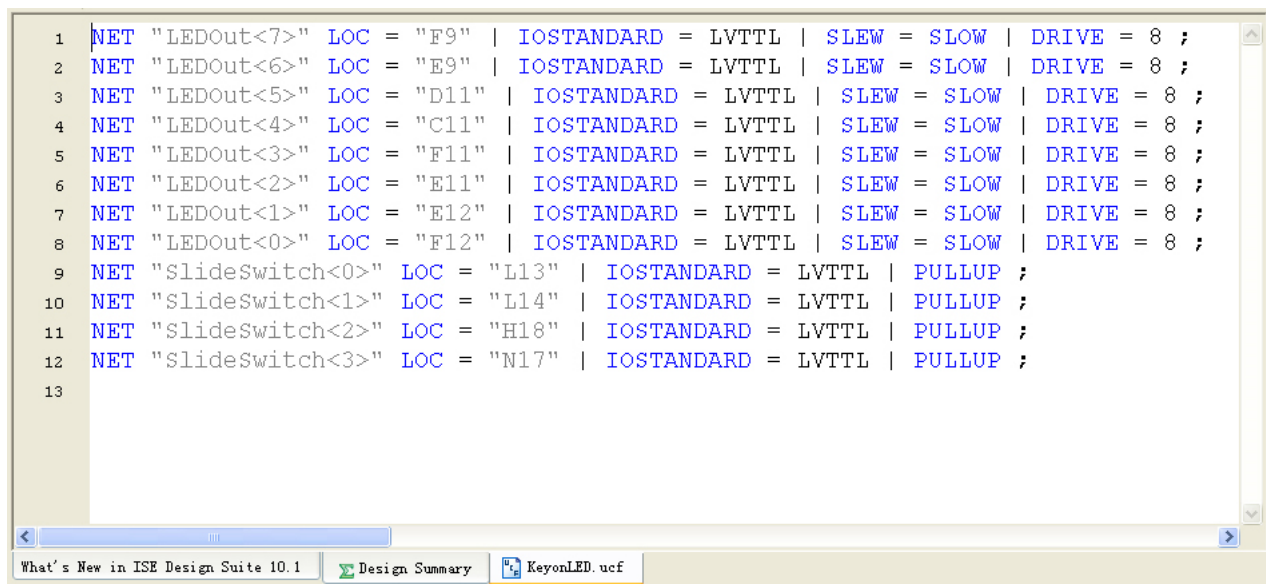


图 4-21、编辑后的 SwitchLED.ucf 文件

这样，输出端口 LEDOut[7:0]就和Spartan3E XC3S500E-4FG320 FPGA芯片的管脚：F9, E9, D11, C11, F11, E11, E12, F12 绑定在一起；而开发板上的 8 个发光二极管 LED 7 ~ LED 0 （从左到右）的一端分别与FPGA 管脚 F9, E9, D11, C11, F11, E11, E12, F12 相连，8 个发光二极管另一端接地，当FPGA 管脚 F9, E9, D11, C11, F11, E11, E12, F12 上为高电平时，

相应的LED点亮。

输入端口 SlideSwitch[3:0] 和 FPGA芯片的管脚：L13, L14, H18, N17 绑定在一起，而开发板上的 4 个滑杆开关（从左到右）SW3, SW2, SW1, SW0 分别与FPGA的管脚N17, H18, L14, L13相连，当滑杆开关处在ON（向开发板内推动）时，相连的FPGA管脚接到 3.3V 高电平，当滑杆开关处在OFF位置（向开发板外边推动）时，相连的FPGA管脚接地（0V低电平）。

按照所设计的Verilog 模块：

```
// 开始添加 Verilog 源代码
```

```
reg [7:0] LEDConnect;
```

```
always @(*) begin
```

```
    {LEDConnect[7], LEDConnect[0]} = {SlideSwitch[0], SlideSwitch[0]};
```

```
    {LEDConnect[6], LEDConnect[1]} = {SlideSwitch[1], SlideSwitch[1]};
```

```
    {LEDConnect[5], LEDConnect[2]} = {SlideSwitch[2], SlideSwitch[2]};
```

```
    {LEDConnect[4], LEDConnect[3]} = {SlideSwitch[3], SlideSwitch[3]};
```

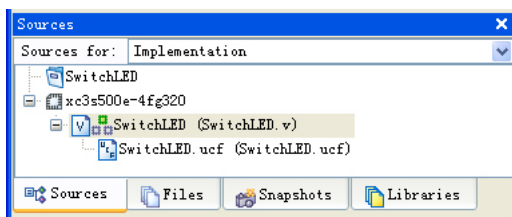
```
end
```

```
assign LEDOut = LEDConnect;
```

```
// 完成添加 Verilog 源代码
```

经过综合布线后，SW0 与 LED7 和 LED0 相连，SW1 与 LED6 和 LED1 相连，SW2 和 LED5 和 LED2 相连，SW3 和 LED4 和 LED3 相连，这样，利用 SW3~SW0 可以控制 LED7~LED0 的“点亮”和“熄灭”。

Step 14



在工程源文件（Sources）窗口中，选中 Verilog 设计文件：SwitchLED.v，在进程窗口（Processes）中双击 **Synthesize – XST**，开始综合。如图 4-22 所示。

图 4-22、对设计模块进行综合

在进行综合处理时，对 Verilog 模块进行语法检查，综合完成后，点击“Synthesis - XST”左边的符号⊕，展开后，双击“View RTL Schematic”，可以观察 RTL 原理图，见图 4-23。

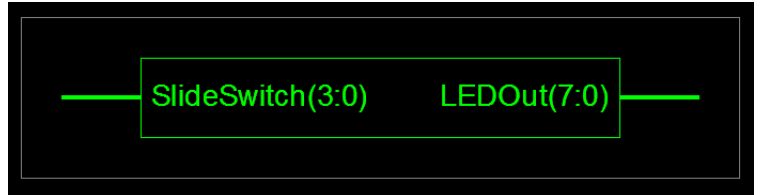


图 4-23、综合后的 RTL 原理图

Step 15

双击设计实现“Implement Design”命令，运行相关进程，ISE 将运行所有必要的步骤实现设计：

- Translate（转换）
- Map（映射）
- Place & Route（布局布线）

在这个过程中：

- 绿色的对号“✓”表示步骤成功完成；
- 黄色的感叹号“！”表示：包含有系统给出的警告，有关警告的更多详细的信息可以在 Transcript 窗口中获取；
- 黄色的问号“？”表示文件过时；
- 而红色的“X”意味着有错误。

Step 15

双击“Generate Programming file”，产生用于下载的位流文件（这里，为：SwitchLED.bit）。

Step 16

成功完成上面的步骤后，点击“Configure Target Device”左边的符号 \oplus ，展开后，双击“Manage Configuration Project (iMPACT)”工具选项，启动 Xilinx 的 FPGA 器件的编程和下载配置工具见图 4-24。

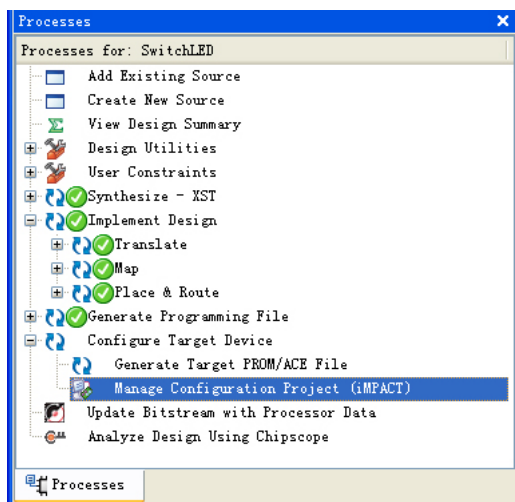


图 4-24、准备启动 iMPACT 下载配置工具

Xilinx 的 FPGA 器件的编程和下载配置工具（iMPACT）启动后，首选弹出如图 4-25 所示的 iMPACT 编程配置方式选择对话框：

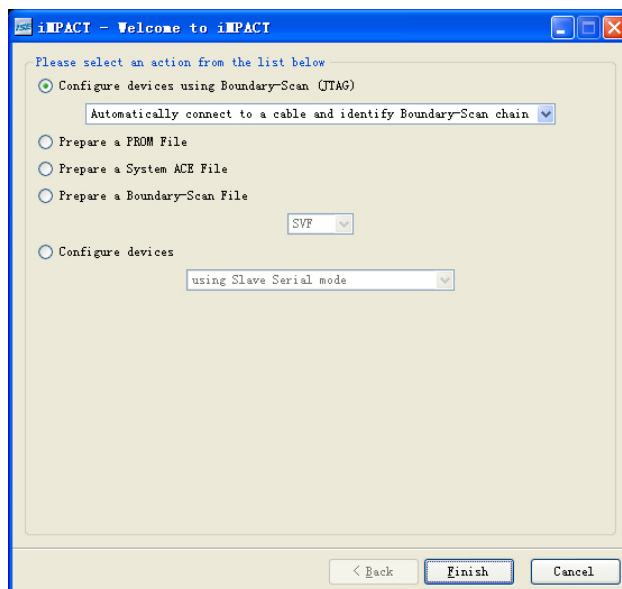


图 4-25、iMPACT 编程配置方式选择对话框

默认为：“Configure device using Boundary-Scan (JTAG)（使用边界扫描(JTAG)方式配置器件）”，点击“Finish”，出现 iMPACT 窗口和“Assign New Configuration File”（指定新配置文件）对

话框，如图 4-26 所示。在边界扫描链中有 3 个可编程芯片，其中一个为 FPGA XC3S500E，另外两个一个是 XC2C64A CoolRunner CPLD，和一个 Xilinx 4 Mbit Flash 用于保持配置位流文件。

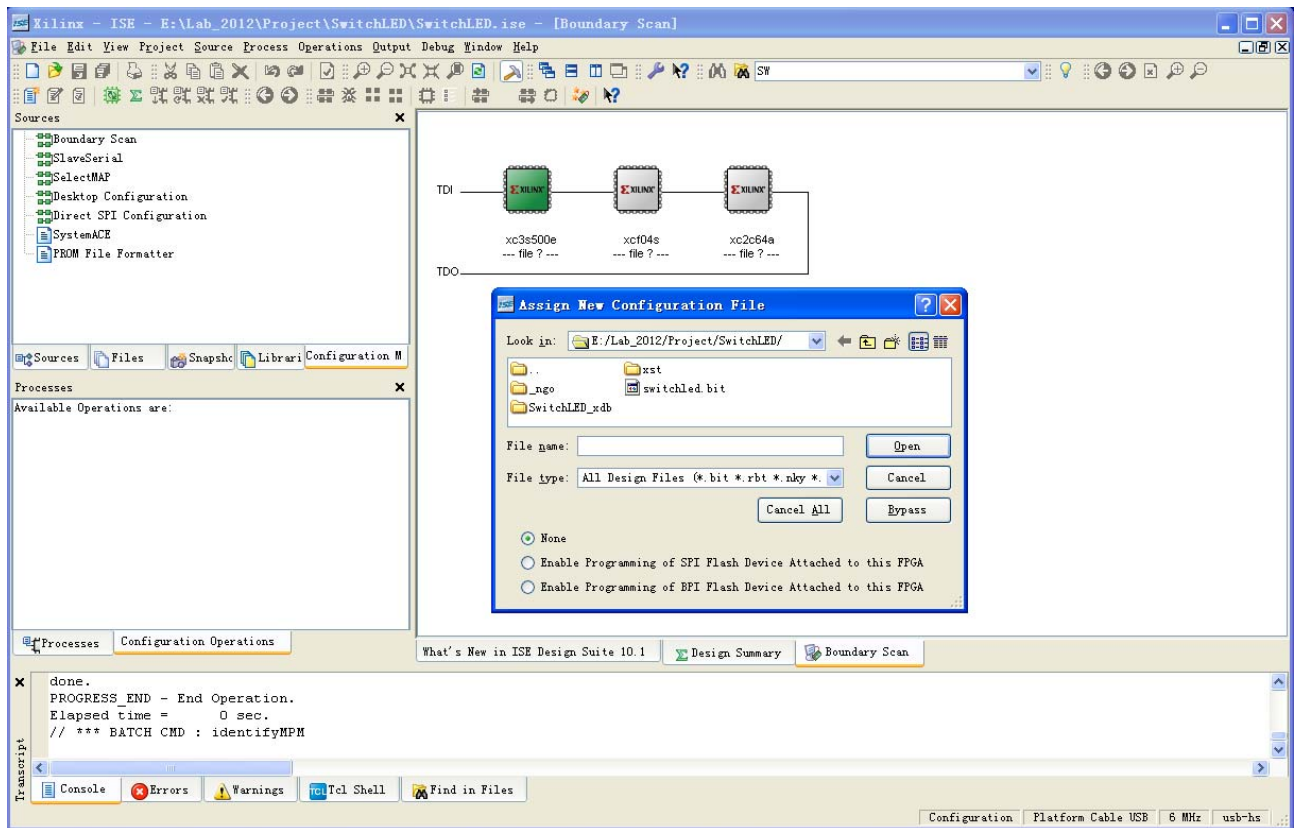


图 4-26、iMPACT 窗口和 “Assign New Configuration File” 对话框

Step 17

当前被配置的器件为 FPGA xc3s500e，显示为绿色，选择已生成的位流文件：SwitchLED.bit，然后，点击 “Open” 打开，如图 4-27 所示。

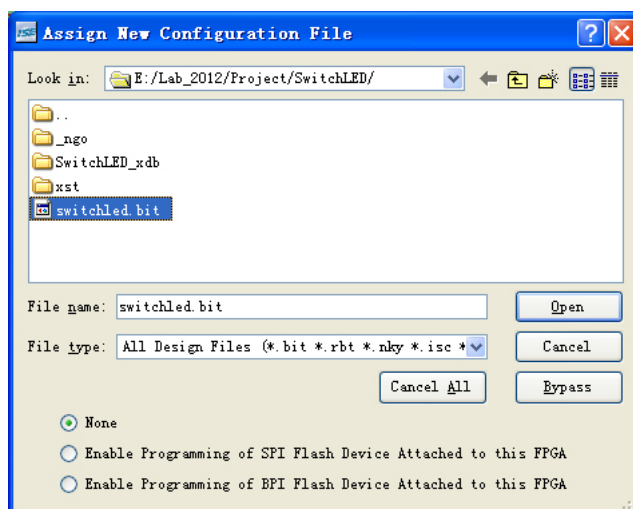


图 4-27、为 FPGA xc3s500e 指定配置文件

然后, 出现为 xcf04s 指定新配置文件对话框, 点击“Cancel”, 接着是为 xc2c64a, 点击“Cancel”。此时, 出现器件编程属性对话框, 如图 4-28, 点击“OK”。如有修改, 则点击“Apply”然后在点击“OK”。

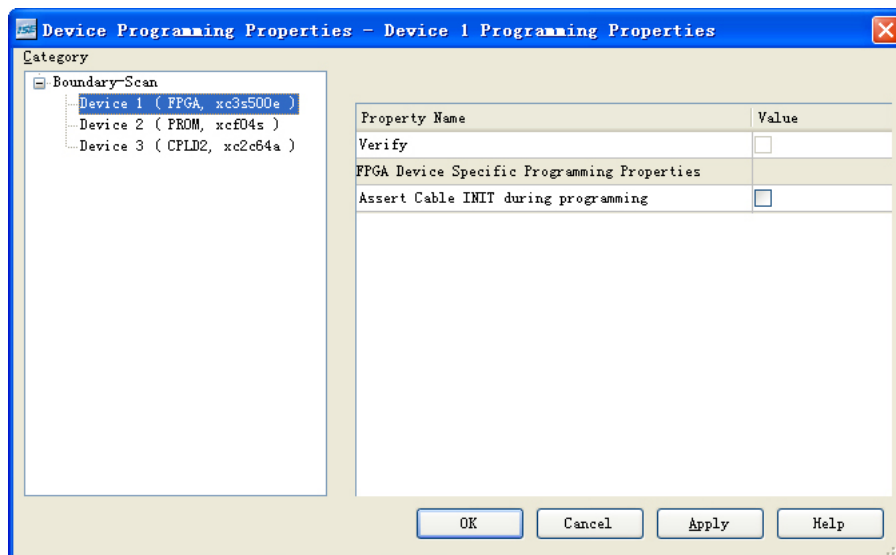


图 4-28、器件编程属性对话框

这时, iMPACT 窗口如图 4-29 所示。

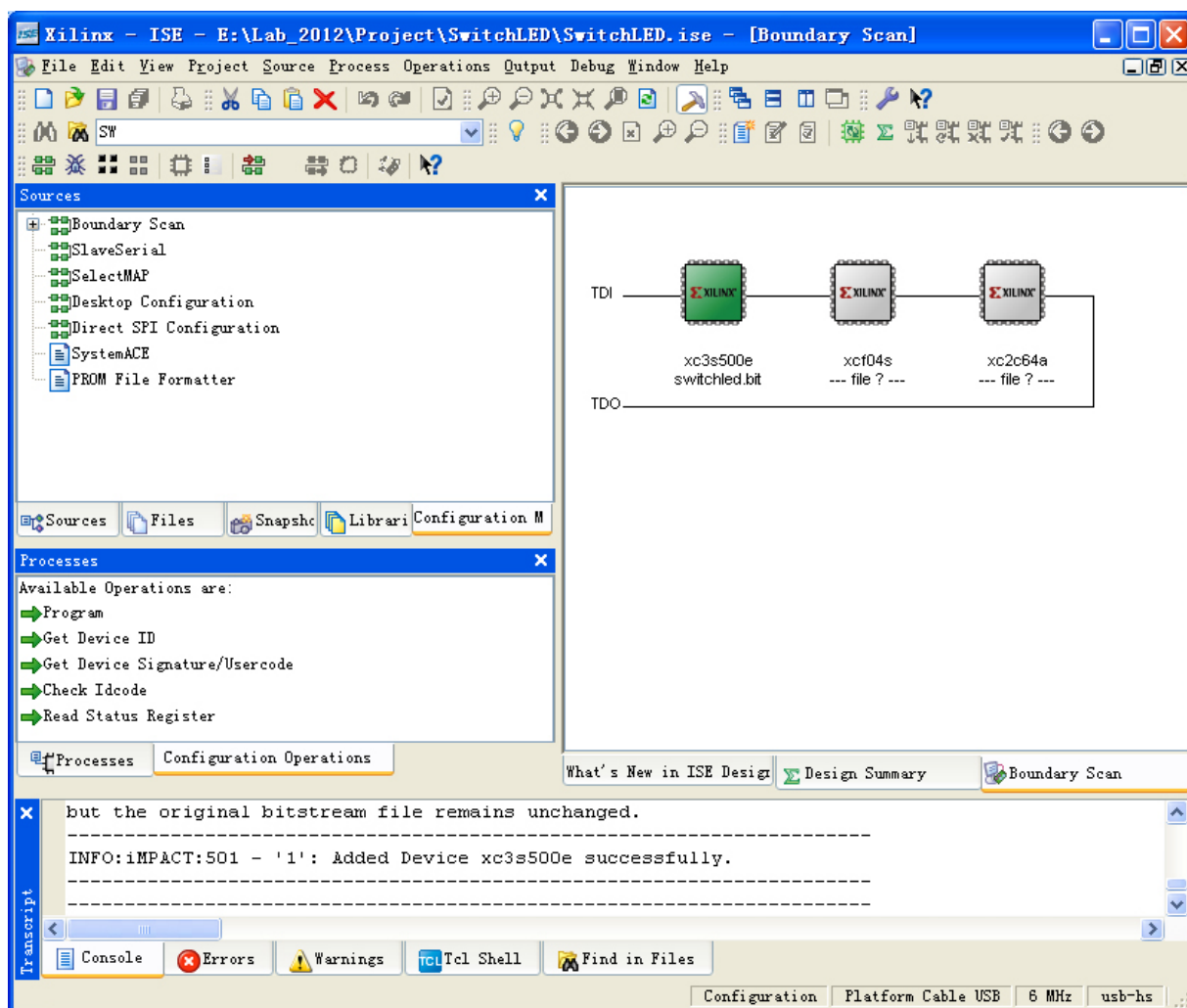


图 4-29、已添加的器件

用鼠标右键点击芯片 xc3s500e，弹出菜单，如下图所示，点击 **Program**，开始使用位流文件 SwitchLED.bit 对 FPGA 器件进行编程，也就是下载生成的位流文件 SwitchLED.bit，如图 4-30 所示，编程过程中出现编程下载进度指示窗口。

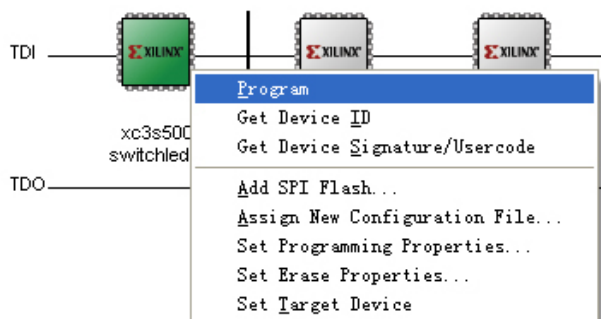
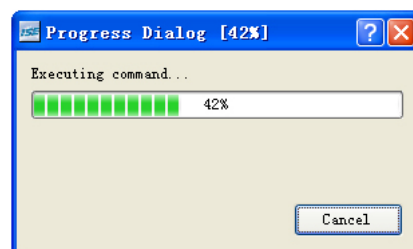


图 4-30、 a、选择编程器件



b、编程进度对话框

下载完毕弹出 Programming Success，表示程序下载成功，如图 4-31 所示，至此，完成了一个简单的 FPGA 开发实验。



图 4-31、下载成功标志

Step 18

现在，在开发板上进行实验操作：

- (1) 当上下推动 SW0 时，LED7 和 LED0 点亮或熄灭；
- (2) 当上下推动 SW1 时，LED6 和 LED1 点亮或熄灭；
- (3) 当上下推动 SW2 时，LED5 和 LED2 点亮或熄灭；
- (4) 当上下推动 SW3 时，LED4 和 LED3 点亮或熄灭；

5、FPGA 开关电路设计

5.1 设计目标

设计一个发光二极管的控制电路模块，利用Spartan-3E 初学者开发板 4 个按键开关（Push-button switches）BTN_EAST，BTN_SOUTH，BTN_WEST和BTN_NORTH，以及4 个滑杆开关（Slide switches）SW3，SW2，SW1和SW0，控制 8 个发光二极管（LED0 ~ LED7）。4 个按键开关和4个滑杆开关，以及8个LED发光二极管如图5-1所示。

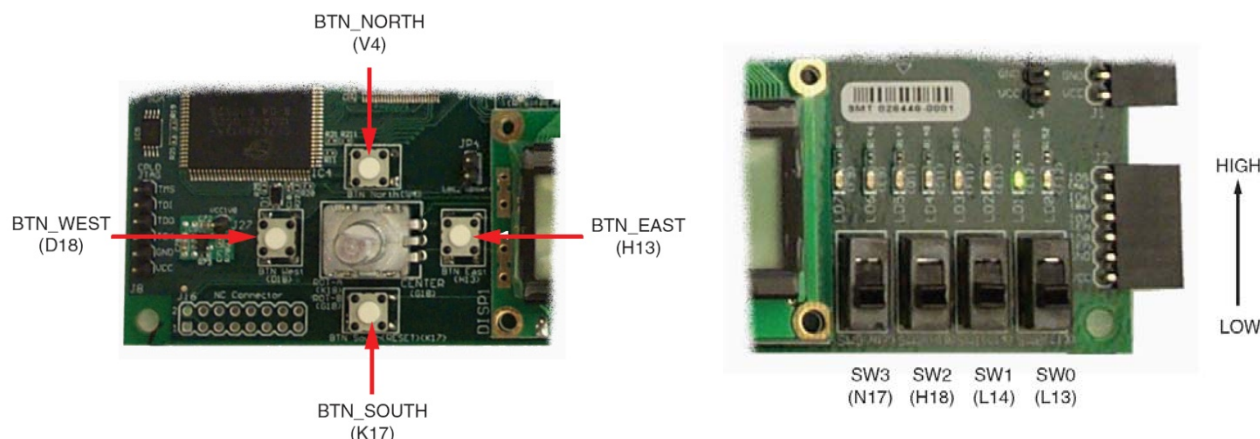


图 5-1、Spartan-3E Starter Kit 的 4 个按键开关、4 个滑杆开关和 8 个 LED

5.2 功能描述

控制电路的功能：

- （1）按键开关 BTN_EAST，BTN_SOUTH，BTN_WEST 和 BTN_NORTH 分别控制 LED0~LED3；
- （2）滑杆开关 SW0~SW3 分别控制 LED4~LED7；

控制电路工作状态：Verilog 模块根据开关的状态，控制 8 个发光二极管。

（0）开始，全部 8 个 LED 均为关闭（熄灭）状态；

（1）当按下 4 个按键中的某一个时，如：按下 BTN_EAST，则对应的 LED0 打开（点亮），并一直保持点亮状态，直到再次按下其对应的按键，将其关闭。

（2）当滑杆开关 SW0~SW3 打开（处在 HIGH 位置）时，则对应的 LED4~LED7 打开（点亮），直到 SW0~SW3 关闭（处在 LOW 位置）。

5.3 设计中需要解决的问题

在 Spartan-3E Starter Kit 的 4 个按键当作开关使用时，实际上是 4 个琴键开关（keyboard switch），琴键开关的功能与任何一种键盘乐器的琴键一样。如：当我们按下钢琴的琴键时，钢琴产生乐音；当按键的手指松开时，钢琴就不再继续发出声音。

与之类似，假设 LED 的信号引脚与按键的相连，由于 LED 是高电平有效，根据手册 UG230，当我们按下按键时，输出一个高电平到 LED 的引脚，点亮 LED；见图 5-2。当我们松开按键时，输出变成了低电平，同时 LED 熄灭。因此，直接使用按键控制 LED 的引脚不能实现设计要求的功能。

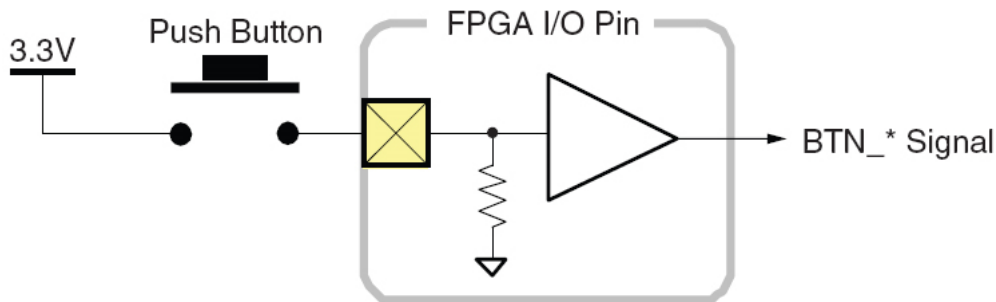


图 5-2、Spartan-3E Starter Kit 的 4 个按键的原理图

要实现设计要求的功能，必须将 4 个琴键开关，转换成乒乓开关（toggle switch），乒乓开关也称作拨动开关，在我们房间中任何一个控制照明电路的开关都是乒乓开关，其功能就是按下一次，就在“打开”和“关闭”这两种状态之间切换一次。

要将琴键开关转换成乒乓开关，必须记录下按键的动作，一个直观的想法就是，利用一个状态寄存器记录按键的工作，如：寄存器的初始状态为高电平，按下一次按键，寄存器的状态改变一次，即，从高电平转换成低电平，再从低电平变成高电平，如此反复变化。

下面的问题就是状态寄存器根据什么条件进行变化，根据 Spartan-3E 初学者开发板的电路工作情况，最直接的方法就是检测按键信号的下跳沿，当检测到一个按键信号的下跳沿时，寄存器的状态变化一次。

在理想情况下，未按下按键时，在下拉电阻的作用下，按键信号引脚的输出为低电平，当按下按键时，按键的信号引脚与 3.3V 接通，此时，按键的信号引脚输出为高电平。因此，上面的方法是非常合理的。然而，关键的问题是，由于按键的机械结构是由弹簧和触点构成的，当按下按键时，由于在弹簧的作用下，开关的接触点产生回跳，在这种情况下，施加一个按键动作产生的电信号通常不是纯净的阶跃信号，可能是由一组由多个振荡脉冲的组成的瞬变信号引导，然后，才进入稳定状态。这些振荡脉冲的持续时间长度，足以使在系统时钟脉冲控制下工作的逻辑电路，检测到新的变化（下降沿再次出现）。如果直接检测按键信号的下跳沿，在一次按

键动作的过程中，将记录多次按键动作，从而导致最终保存在状态寄存器中的状态，难以确定。在所有的数字设计中，这个问题都将出现。

下面，解决这个问题。

一般情况下，FPGA 的响应时间在 ns 级，而振荡脉冲的范围在 μs 级。多个振荡脉冲总的可以使用一个移位寄存器消除这种开关接触回跳产生的振荡脉冲。

消除回跳振荡脉冲的电路模块如下：

```
module debounce( output reg pb,      // 消除回跳振荡信号后的按键动作信号输出
                 input keydown,      // 按键动作信号输入
                 input clk_100Hz     // 时序电路的 100Hz 时钟信号输入
                 );
    reg [3:0] shift_pb;
    always @( posedge clk_100Hz )
    begin
        shift_pb[2:0] <= shift_pb[3:1];
        shift_pb[3] <= keydown;

        if ( shift_pb == 4'b0000 )
            pb <= 1'b0;
        else pb <= 1'b1;
    end
endmodule
```

上面的模块的设计原理是，消除回跳脉冲的电路在 100Hz 的时钟脉冲的控制下工作，4bit 寄存器的在 40 ms 的时间间隔中，如果有一位为 1，则电路的输出为高电平，否则，电路的输出为低电平。

这样就有效地消除了开关回跳产生的振荡信号，从而保证按键动作状态寄存器记录下正确的状态。

在上面的 Verilog 模块 debounce.v 中，需要 100Hz 的时钟信号输入，而在 Spartan-3E 初学者开发板上有一个 50MHz 的时钟振荡器，如图 5-3 所示，可以向 FPGA 输入 50MHz 的时钟信号源。

现在需要设计一个分频电路模块，对输入到 FPGA 的 50MHz 时钟信号进行分频，产生消除回跳模块需要的 100Hz 时钟信号。



On-Board 50 MHz Oscillator
CLK_50MHz: (C9)

图 5-3、开发板上的 50MHz 时钟振荡器

下面分析如何从 50MHz 的系统全局时钟信号产生 100Hz 时钟信号：

(1) 50.000MHz = 50,000,000(dec) Hz = 2FA_F080 (hex) Hz

= 10_1111_1010_1111_0000_1000_0000 (bin) Hz

(2) 对系统全局时钟进行分频，产生 100 Hz 的时钟脉冲，即需要产生每 1/200(s) 翻转一次的时钟脉冲，1/200(s)等于 250,000(dec)个系统时钟脉冲，即：

$$\frac{50,000,000(\text{dec})\text{Hz}}{200(\text{dec})\text{Hz}} = 250,000(\text{dec}) = 3_D090(\text{Hex}) = 11_1101_0000_1001_0000(\text{bin}) \text{ (pulses)}$$

产生 100Hz 时钟信号的 Verilog 模块 clock_div.v 如下：

```
module clock_div( output reg clk_100Hz,    // 100 Hz 时钟输出信号
                  input  clk               // 系统时钟输入信号
                );
    parameter PULSESCOUNT = 18'h3_D090, // = 11_1101_0000_1001_0000(bin) (16bits)
              RESETZERO = 18'h0;

    reg [17:0] counter;    // 计数器, 18 bits (11_1101_0000_1001_0000(bin))
                          // 用于对系统时钟脉冲进行计数, 以产生 100Hz 输出时钟信号

    always @(posedge clk) begin    // -- 由 clock 信号的上升沿触发
        if ( counter < PULSESCOUNT )    // -- 18'h3_D090 个系统时钟脉冲等于 1/200(s)
            counter <= counter + 1'b1;
        else begin
            clk_100Hz <= ~clk_100Hz;
            counter <= RESETZERO;
        end
    end
end
endmodule
```

完成了消除回跳模块 debounce.v 和时钟分频模块 clock_div.v，现在，利用这两个低层模块设计顶层开关控制模块 Control_LED.v。顶层模块 Control_LED.v 如下：

```

`include "clock_div.v"
`include "debounce.v"

module Control_LED( output [7:0] LEDOut,
                    input [3:0] button, slide,
                    input clock );

    parameter  NBTN = 4;

    wire p_clk100Hz;
    wire [NBTN-1 :0] p_toggle;

    reg [NBTN-1 :0] state_toggle;    // 保持 4 个乒乓开关的状态寄存器

    // 100Hz 时钟信号产生模块调用实例
    clock_div m_clkgen( .clk_100Hz(p_clk100Hz), .clk(clock) );

    // 使用循环生成语句
    genvar k;

    generate for ( k = 0; k < NBTN; k = k + 1 )
    begin: CTRL_LEDS
        // 消除开关接触回跳产生的振荡脉冲电路模块调用实例
        debounce m_deb ( .pb(p_toggle[k]),
                        .keydown(button[k]),
                        .clk_100Hz(p_clk100Hz) );
        // 将琴键开关转换成乒乓开关
        always @( posedge p_toggle[k] )
            state_toggle[k] <= state_toggle[k] + 1'b1;

        // 使用 4 个按键控制 LED4~LED7
        assign LEDOut[k+NBTN] = state_toggle[k];
    end
endgenerate

    // 使用 4 个滑杆开关控制 LED0~LED3
    assign LEDOut[NBTN-1:0] = slide;

endmodule

```

5.4 建立 ISE 实现设计

首先在 ISE 建立中建立一个工程，工程名为：Control_LED，然后，如图 5-3b 所示，将前面设计的三个 Verilog 源程序文件 clock_div.v、debounce.v 和 Control_LED.v 添加到工程 Control_LED 中。

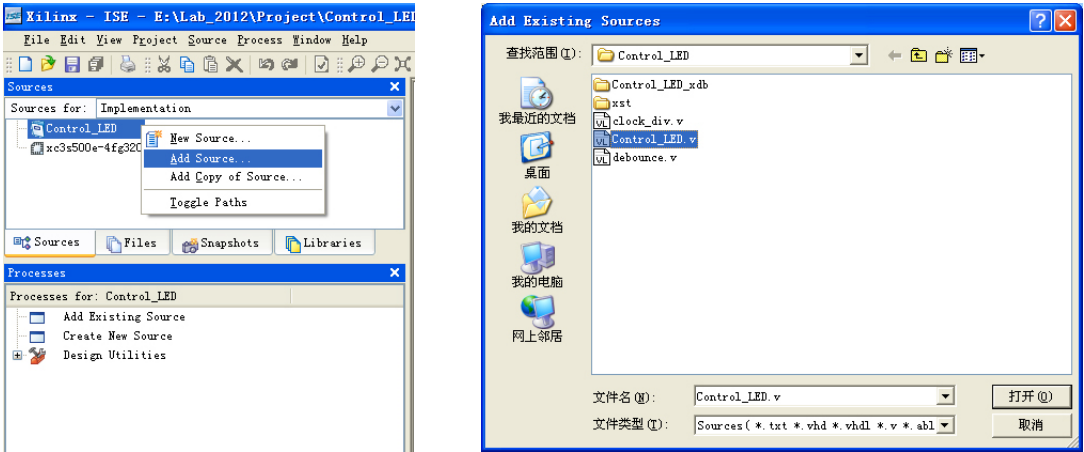


图 5-3、向 ISE 工程添加设计文件

在“Add Existing Sources”对话框中，**只选择顶层设计文件：Control_LED.v。**

然后，点击“打开”，此时，弹出如图 5-4 所示的对话框，点击其中模块名，如“debounce”右侧的下拉列表按钮，可以选择此模块在工程中用于“Implementation”、“Simulation”还是“All（Implementation & Simulation）”，如图 5-5 所示。

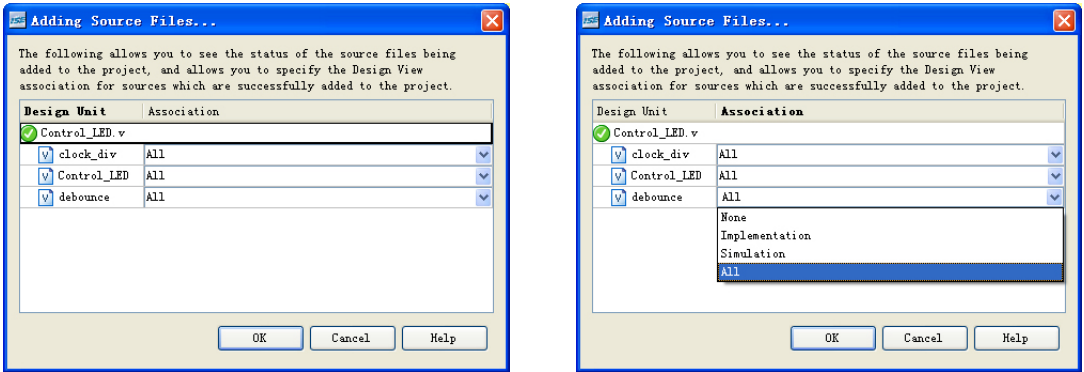


图 5-4、指定加入的文件用途

对于用于综合和实现 FPGA 设计的 Verilog 源文件，其默认是选择“All”，点击“OK”，将 Verilog 源程序文件添加到工程里。此时，在“Sources”窗口中列出了新添加的文件，如图 5-5 所示，低层文件 clock_div.v 和 debounce.v 列在顶层文件 Control_LED.v 下。

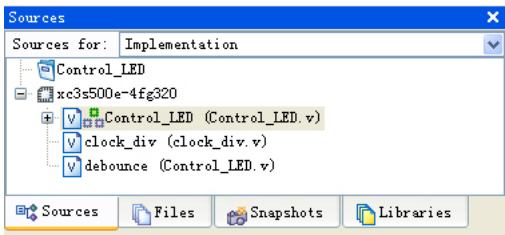


图 5-5、设计文件添加到工程

当完成 Verilog 模块的设计后，接下来，就是在工程中添加用户约束文件。采用在第 4 章中介绍的方法，为工程添加用户约束文件：Control_LED.ucf

```
NET "clock" LOC = "C9" | IOSTANDARD = LVCMOS33 ;
NET "LEDOut<7>" LOC = "F9" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDOut<6>" LOC = "E9" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDOut<5>" LOC = "D11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDOut<4>" LOC = "C11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDOut<3>" LOC = "F11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDOut<2>" LOC = "E11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDOut<1>" LOC = "E12" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LEDOut<0>" LOC = "F12" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "button<0>" LOC = "H13" | IOSTANDARD = LVTTTL | PULLDOWN ;
NET "button<1>" LOC = "K17" | IOSTANDARD = LVTTTL | PULLDOWN ;
NET "button<2>" LOC = "D18" | IOSTANDARD = LVTTTL | PULLDOWN ;
NET "button<3>" LOC = "V4" | IOSTANDARD = LVTTTL | PULLDOWN ;
NET "slide<0>" LOC = "L13" | IOSTANDARD = LVTTTL | PULLUP ;
NET "slide<1>" LOC = "L14" | IOSTANDARD = LVTTTL | PULLUP ;
NET "slide<2>" LOC = "H18" | IOSTANDARD = LVTTTL | PULLUP ;
NET "slide<3>" LOC = "N17" | IOSTANDARD = LVTTTL | PULLUP ;
```

接下来，依次运行以下工具：

- (1) Synthesize – XST; (2) Implement Design; (3) Generate Programming file;
- (4) Configure Target Device ⇨ Manage Configuration Project (iMPACT) ⇨ Program

然后，在开发板上进行实验操作：

- (1) 当上下推动 SW0 时，LED0 点亮或熄灭；
- (2) 当上下推动 SW1 时，LED1 点亮或熄灭；
- (3) 当上下推动 SW2 时，LED2 点亮或熄灭；
- (4) 当上下推动 SW3 时，LED3 点亮或熄灭；
- (5) 当按下 BTN_EAST 时，LED4 点亮或熄灭；
- (6) 当按下 BTN_SOUTH 时，LED5 点亮或熄灭；
- (7) 当按下 BTN_WEST 时，LED6 点亮或熄灭；
- (8) 当按下 BTN_NORTH 时，LED7 点亮或熄灭；