华清远见 FARSIGHT 始于2004 | 嵌入式培训专家

10年口碑积累，成功培养50000多名研发工程师，铸就专业品牌形象

华清远见的企业理念是不仅要做良心教育、做专业教育，更要做受人尊敬的职业教育。

《FPGA 应用开发入门与典型实例》（修订版）

作者：华清远见

专业始于专注 卓识源于远见

第 1 章　FPGA 系统设计基础

本章目标

- 了解 FPGA 技术的发展历史和动向
- 了解 FPGA 的典型应用领域
- 了解主流的 FPGA 芯片厂家及其代表产品
- 了解工程项目中 FPGA 芯片选择策略和原则

专业始于专注 卓识源于远见

开发板免费领取试用 开发板免费领取试用

1.1 FPGA 技术的发展历史和动向

1.1.1 FPGA 技术的发展历史

纵观数字集成电路的发展历史，经历了从电子管、晶体管、小规模集成电路到大规模以及超大规模集成电路等不同的阶段。发展到现在，主要有 3 类电子器件：存储器、处理器和逻辑器件。

存储器保存随机信息（电子数据表或数据库的内容）；处理器执行软件指令，以便完成各种任务（运行数据处理程序或视频游戏）；而逻辑器件可以提供特殊功能（器件之间的通信和系统必须执行的其他所有功能）。

逻辑器件分成两类：

① 固定的或定制的。

② 可编程的或可变的。

其中，固定的或定制的逻辑器件通常称为专用芯片（ASIC）。ASIC 是为了满足特定的用途而设计的芯片，例如 MP3 解码芯片等。其优点是通过固化的逻辑功能和大规模的工业化生产，降低了芯片的成本，同时提高了产品的可靠性。随着集成度的提高，ASIC 的物理尺寸也在不断的缩小。

但是，ASIC 设计的周期很长，而且投资大，风险高。一旦设计结束后，功能就固化了，以后的升级改版困难比较大。电子产品的市场正在逐渐细分，为了满足快速产品开发，产生了现场可编程逻辑器件（FPGA）。自 1984 年 Xilinx 公司推出了第一片现场可编程逻辑器件（FPGA）至今，FPGA 已经历了 20 几年的快速发展历程。特别是近几年来，更是发展迅速。FPGA 的逻辑规模已经从最初的 1000 个可用门发展到现在的 1000 万个可用门。

FPGA 技术之所以具有巨大的市场吸引力，其根本原因在于：FPGA 不仅可以解决电子系统小型化、低功耗、高可靠性等问题，而且其开发周期短、投入少，芯片价格不断下降。FPGA 正在越来越多地取代传统上 ASIC，特别是在小批量、个性化的产品市场方面。

1.1.2 FPGA 技术的发展动向

随着芯片设计工艺水平的不断提高，FPGA 技术呈现出了以下 4 个主要的发展动向。

1．基于 FPGA 的嵌入式系统（SoPC）技术正在成熟

System on Chip（SoC）技术在芯片设计领域被越来越广泛地采用，而 SoPC 技术是 SoC 技术在可编程器件领域的应用。这种技术的核心是在 FPGA 芯片内部构建处理器。Xilinx 公司主要提供基于 Power PC 的硬核解决方案，而 Altera 提供的是基于 NIOSII 的软核解决方案。

Altera 公司为 NIOSII 软核处理器提供了完整的软硬件解决方案，可以让客户短时间完成 SoPC 系统的构建和调试工作。

如图 1.1 所示，是 Altera Stratix III FPGA 基于 NIOS II 解决方案的内部结构图。

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

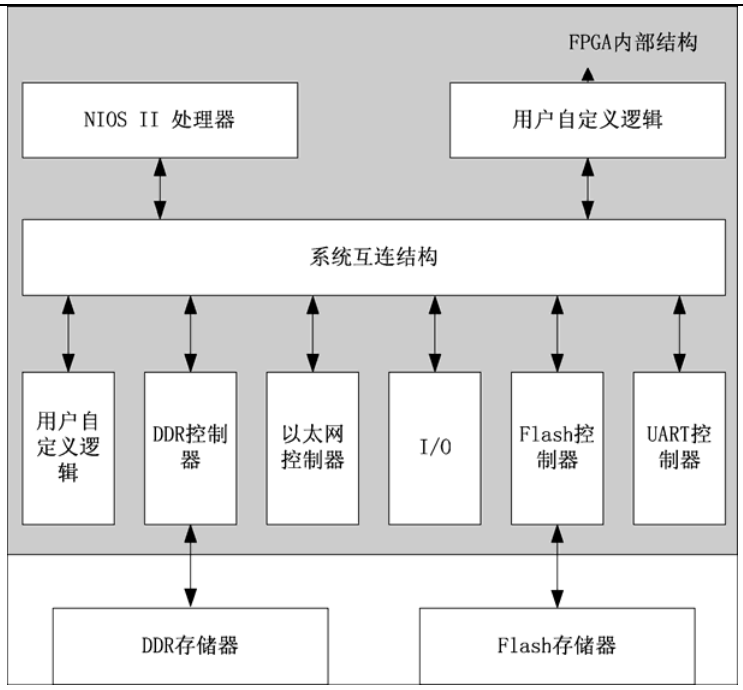


图 1.1　NIOS II 解决方案

2．FPGA 芯片向高性能、高密度、低压和低功耗的方向发展

随着芯片生产工艺不断提高，FPGA 芯片的性能和密度都在不断提高。早期的 FPGA 主要是完成接口逻辑设计，比如 AD/DA 和 DSP 的粘合逻辑。现在的 FPGA 正在成为电路的核心部件，完成关键功能。
在高性能计算和高吞吐量 I/O 应用方面，FPGA 已经取代了专用的 DSP 芯片，成为最佳的实现方案。因此，高性能和高密度也成为衡量 FPGA 芯片厂家设计能力的重要指标。
随着 FPGA 性能和密度的提高，功耗也逐渐成为了 FPGA 应用的瓶颈。虽然 FPGA 比 DSP 等处理器的功耗低，但是要明显高于专用芯片（ASIC）的功耗。FPGA 的厂家也在采用各种新工艺和技术来降低 FPGA 的功耗，并且已经取得了明显的效果。
例如，Altera 公司的 StratixIII 系列 FPGA 的功耗比上一代产品 StratixII 系列降低了 50%以上。

3．基于 IP 库的设计方法

未来的 FPGA 芯片密度不断提高，传统的基于 HDL 的代码设计方法很难满足超大规模 FPGA 的设计需要。随着专业的 IP 库设计公司不断增多，商业化的 IP 库种类会越来越全面，支持的 FPGA 器件也会越来广泛。作为 FPGA 的设计者，主要的工作是找到适合项目需要的 IP 库资源，然后将这些 IP 整合起来，完成顶层模块设计。由于商业的 IP 库都是通过验证的，因此整个项目的仿真和验证工作主要就是验证 IP 库的接口逻辑设计的正确性。
目前，由于国内的知识产权保护的相关法律法规还不尽完善，基于 IP 库的设计方法还没有得到广泛应用。但是随着 FPGA 密度不断提高和 IP 库的价格逐渐趋于合理化，这种设计方法将会成为主流的 FPGA 设计技术。

4．FPGA 的动态可重构技术

FPGA 动态重构技术主要是指对于特定结构的 FPGA 芯片，在一定的控制逻辑的驱动下，对芯片的全部或部分逻辑资源实现高速的功能变换，从而实现硬件的时分复用，节省逻辑资源。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

由于密度不断提高，FPGA 能实现的功能也越来越复杂。FPGA 全部逻辑配置一次的需要的时间也变长了，降低了系统的实时性。局部逻辑的配置功能可以实现“按需动态重构”，大大提高了配置的效率。

动态可重构的 FPGA 可以在系统运行中对电路功能进行动态配置，实现硬件的时分复用，节省了资源，主要适用于以下两个系统设计。

① 最新通信系统。

FPGA 的动态重构特性可以适应不同体制和不同标准的通信要求，满足软件无线电技术的发展和第三代（3G）和第四代（4G）移动通信系统的需要。

② 重构计算机：FPGA 具有并行处理能力和动态配置能力，可自动改变硬件来适应正在运行的程序，产生了基于这种软硬件环境的全新概念的计算机。

1.2 FPGA 的典型应用领域

1.2.1 数据采集和接口逻辑领域

1．FPGA 在数据采集领域的应用

由于自然界的信号大部分是模拟信号，因此一般的信号处理系统中都要包括数据的采集功能。通常的实现方法是利用 A/D 转换器将模拟信号转换为数字信号后，送给处理器，比如利用单片机（MCU）或者数字信号处理器（DSP）进行运算和处理。

对于低速的 A/D 和 D/A 转换器，可以采用标准的 SPI 接口来与 MCU 或者 DSP 通信。但是，高速的 A/D 和 D/A 转换芯片，比如视频 Decoder 或者 Encoder，不能与通用的 MCU 或者 DSP 直接接口。在这种场合下，FPGA 可以完成数据采集的粘合逻辑功能。

2．FPGA 在逻辑接口领域的应用

在实际的产品设计中，很多情况下需要与 PC 机进行数据通信。比如，将采集到的数据送给 PC 机处理，或者将处理后的结果传给 PC 机进行显示等。PC 机与外部系统通信的接口比较丰富，如 ISA、PCI、PCI Express、PS/2、USB 等。

传统的设计中往往需要专用的接口芯片，比如 PCI 接口芯片。如果需要的接口比较多，就需要较多的外围芯片，体积、功耗都比较大。采用 FPGA 的方案后，接口逻辑都可以在 FPGA 内部来实现了，大大简化了外围电路的设计。

在现代电子产品设计中，存储器得到了广泛的应用，例如 SDRAM、SRAM、Flash 等。这些存储器都有各自的特点和用途，合理地选择储存器类型可以实现产品的最佳性价比。由于 FPGA 的功能可以完全自己设计，因此可以实现各种存储接口的控制器。

3．FPGA 在电平接口领域的应用

除了 TTL、COMS 接口电平之外，LVDS、HSTL、GTL/GTL+、SSTL 等新的电平标准逐渐被很多电子产品采用。比如，液晶屏驱动接口一般都是 LVDS 接口，数字 I/O 一般是 LVTTL 电平，DDR SDRAM 电平一般是 HSTL 的。

在这样的混合电平环境里面，如果用传统的电平转换器件实现接口会导致电路复杂性提高。利用 FPGA 支持多电平共存的特性，可以大大简化设计方案，降低设计风险。

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

1.2.2 高性能数字信号处理领域

无线通信、软件无线电、高清影像编辑和处理等领域，对信号处理所需要的计算量提出了极高的要求。传统的解决方案一般是采用多片 DSP 并联构成多处理器系统来满足需求。
但是多处理器系统带来的主要问题是设计复杂度和系统功耗都大幅度提升，系统稳定性受到影响。FPGA 支持并行计算，而且密度和性能都在不断提高，已经可以在很多领域替代传统的多 DSP 解决方案。
例如，实现高清视频编码算法 H.264。采用 TI 公司 1GHz 主频的 DSP 芯片需要 4 颗芯片，而采用 Altera 的 StratixII EP2S130 芯片只需要一颗就可以完成相同的任务。FPGA 的实现流程和 ASIC 芯片的前端设计相似，有利于导入芯片的后端设计。

1.2.3 其他应用领域

除了上面一些应用领域外，FPGA 在其他领域同样具有广泛的应用。
（1）汽车电子领域，如网关控制器/车用 PC 机、远程信息处理系统。
（2）军事领域，如安全通信、雷达和声纳、电子战。
（3）测试和测量领域，如通信测试和监测、半导体自动测试设备、通用仪表。
（4）消费产品领域，如显示器、投影仪、数字电视和机顶盒、家庭网络。
（5）医疗领域，如软件无线电、电疗、生命科学。

1.3 FPGA 的工艺结构

随着 FPGA 的生产工艺不断提高，各种新技术被广泛应用到 FPGA 芯片的设计生产的各个环境。其中，生产工艺结构决定了 FPGA 芯片的特性和应用场合。
如图 1.2 所示是 FPGA 的主要几种生产工艺及典型产品。

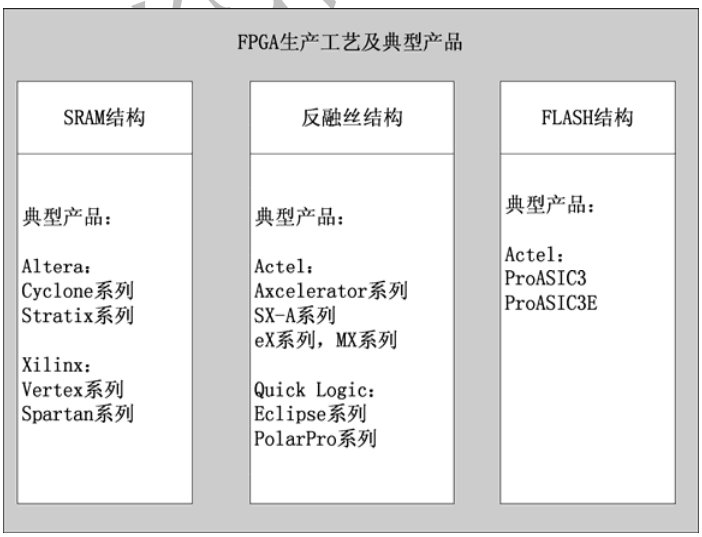


图 1.2 FPGA 生产工艺及典型产品

1.3.1 基于 SRAM 结构的 FPGA

目前最大的两个 FPGA 厂家 Xilinx 和 Altera 的所有 FPGA 产品都是基于 SRAM 工艺来实现的。这种工艺的优点是可以用较低的成本来实现较高的密度和较高的性能；缺点是掉电后 SRAM 会失去所有配置，导致每次上电都需要重新加载。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

重新加载需要外部的器件来实现，不仅增加了整个系统的成本，而且引入了不稳定的因素。加载的过程容易受到外界干扰而导致加载失败，也容易受到“监听”而破解加载文件的比特流。
虽然基于 SRAM 结构的 FPGA 存在这些缺点，但是由于其实现成本低，还是得到了广泛的应用，特别是民用产品方面。

1.3.2 基于反融丝结构的 FPGA

Actel 公司擅长出品反融丝结构的 FPGA。这种结构的 FPGA 只能编程一次，编程后和 ASIC 一样成为了固定逻辑器件。Quick Logic 公司也有类似的 FPGA 器件，主要面向军品级应用市场。
这样的 FPGA 失去了反复可编程的灵活性，但是大大提高了系统的稳定性。这种结构的 FPGA 比较适合应用在环境苛刻的场合，比如高振动，强电磁辐射等航空航天领域。同时，系统的保密性也得到了提高。
这类 FPGA 因为上电后不需要从外部加载配置，所以上电后可以很快进入工作状态，即 “瞬间上电”技术。这个特性可以满足一些对上电时间要求苛刻的系统。由于是固定逻辑，这种器件的功耗和体积也要低于 SRAM 结构的 FPGA。

1.3.3 基于 Flash 结构的 FPGA

Flash 具备了反复擦写和掉电后内容非易失特性，因而基于 Flash 结构的 FPGA 同时具备了 SRAM 结构的灵活性和反融丝结构的可靠性。这种技术是最近几年发展起来的新型 FPGA 实现工艺，目前实现的成本还偏高，没有得到大规模的应用。
系统安全的角度来看，基于 Flash 的 FPGA 具有更高的安全性，硬件出错的几率更小，并能够通过公共网络实现安全性远程升级，经过现场处理即可实现产品的升级换代。这种性能减少了现场解决问题所需的昂贵开销。
在 Flash 器件中集成小型的 NVM（Non Volatile Memory，非易失性存储器）模块可以在某些消费电子和汽车电子应用中实现授权技术。这种 NVM 可以存储安全通信所需的密钥，或者针对基于广播的系统实现机顶盒设备的串行化。
可重编程的 NVM 在编程时需要一定的电压，因此 SRAM 用户必须从外部提供这种电压。基于 Flash 的 FPGA 采用内部电荷泵进行编程，不需要集成 NVM 模块，而基于 SRAM 的 FPGA 通常缺乏这种功能。
Flash 器件的工作频率可达 350MHz，利用率超过 95%，而 SRAM FPGA 一般能够达到的利用率仅为 70～75%。Flash FPGA 在加电时没有像 SRAM FPGA 那样大的瞬间高峰电流，并且 SRAM FPGA 通常具有较高的静态功耗和动态功耗。
例如，一块 40 万门的基于 Flash 的 FPGA 需要 20mA 的静态电流，然而同等规模的基于 SRAM 的 FPGA 所需的电流达 100mA。SRAM FPGA 的功耗问题往往迫使系统设计者不得不增大系统供电电流，并使得整个设计变得更加复杂。

1.4 主流的 FPGA 芯片厂家及其代表产品

目前市场上 FPGA 芯片主要来自 Xilinx 公司和 Altera 公司。这两家公司占据了 FPGA 80%以上的市场份额，其他的 FPGA 厂家产品主要是针对某些特定的应用。比如，Actel 公司主要生产反融丝结构的 FPGA，以满足应用条件极为苛刻的航空、航天领域产品。
下面介绍 Xilinx 和 Altera 两家公司的代表产品。

1.4.1 Xilinx 公司的代表产品

华清远见
华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

1．面向高性能的 Virtex™-5 FPGA 系列

终极系统集成平台——Virtex-5 系列 FPGA 提供了 4 种新型平台，每种平台都在高性能逻辑、串行连接功能、信号处理和嵌入式处理性能方面实现了最佳平衡。现有的 3 款平台如下。

（1）Virtex-5 LX 平台：针对高性能逻辑进行了优化。

（2）Virtex-5 LXT 平台：针对带有低功耗串行连接功能的高性能逻辑进行了优化。

（3）Virtex-5 SXT 平台：针对带有低功耗串行连接功能的 DSP 和存储器密集型应用进行了优化。

2．面向低成本的 Spartan-3 系列 FPGA

90nm Spartan-3 系列 FPGA 的发售量已经超过 3000 万片，是业内首款大容量 FPGA 系列产品，带有多个针对特定领域进行了优化的平台。

（1）面向数字信号处理的 Spartan-3A DSP 平台。

这个平台对 DSP 进行了优化，适于那些需要集成式 DSP MAC 和扩展存储器的应用。特别适于那些需要低成本 FPGA 来实现信号处理（如军用无线电、监视照相机、医学成像等）的应用设计。

（2）面向非易失性应用的 Spartan-3AN 平台。

这个平台主要针对需要非易失性、系统集成、安全、大型用户 Flash 的应用。特别适于空间敏感型或安全应用，以及低成本嵌入式控制器。

（3）面向主流应用的 Spartan-3 平台。

① Spartan-3A 平台：针对 I/O 进行了优化。

这个平台是针对那些 I/O 数和性能比逻辑密度更重要的应用，特别适于桥接、差分信号和存储器接口这些需要宽接口或者多个接口以及一定处理能力的应用。

② Spartan-3E 平台：针对逻辑进行了优化。

这个平台是针对那些逻辑密度比 I/O 数更重要的应用，特别适于逻辑集成、DSP 协处理和嵌入式控制，这些需要进行大量处理和窄接口或者少量接口的应用。

③ Spartan-3 平台：针对密度最高和管脚数较多的应用。

这个平台是针对那些高逻辑密度和高 I/O 数都很重要的应用，特别适于高度集成的数据处理应用。

1.4.2 Altera 公司的代表产品

1．面向高性能的 StraitixIII 系列 FPGA

和 Xilinx 的 Virtex-4 系列对应，Altera 公司也推出了 StratixIII 系列 FPGA 体系结构。StratixIII 系列不仅性能比上一代提高很多，更重要的是静态和动态功耗比前代 FPGA 低了 50%。

Stratix III 器件经过设计，支持高速内核以及高速 I/O，并且具有非常好的信号完整性。例如，它能够实现 400MHz DDR3 的 FPGA。这种性能的提高源于以下几点。

- 增强 DSP 模块，方便实现了信号处理算法。
- 经过优化的内部存储器，改进了信号完整性存储器接口。
- 高性能外部存储器接口。
- 改进了布线体系结构。
- 灵活的 I/O 支持最新的外部存储器标准。

为了给客户的设计应用提供最好的性价比解决方案，Altera Stratix III FPGA 提供 3 种型号，分别针对逻辑、DSP 和存储器以及收发器进行了优化。

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

2．面向低成本的 Cyclone III 系列 FPGA

低成本 Cyclone III FPGA 是 Altera Cyclone 系列的第三代产品。Cyclone III FPGA 系列前所未有地同时实现了低功耗、低成本和高性能，进一步扩展了 FPGA 在成本敏感、大批量领域中的应用。
Cyclone III FPGA 采用 TSMC 公司的 65-nm 低功耗（LP）工艺技术。Cyclone III 器件对芯片和软件采取了更多的优化措施，在所有 65-nm FPGA 中是功耗最低的，在对成本和功耗敏感的大量应用中，提供丰富的特性推动宽带并行处理的发展。
Cyclone III 系列包括 8 个型号，具有 5k～120k 个逻辑单元（LE），最多有 534 个用户 I/O 引脚。Cyclone III 器件具有 4MB 嵌入式存储器、288 个嵌入式 18×18 乘法器、专用外部存储器接口电路、锁相环（PLL）以及高速差分 I/O 等。

1.5 工程项目中 FPGA 芯片选择策略和原则

由于 FPGA 具备设计灵活、可以重复编程的优点，因此在电子产品设计领域得到了越来越广泛的应用。在工程项目或者产品设计中，选择 FPGA 芯片可以参考以下的几点策略和原则。

1.5.1 尽量选择成熟的产品系列

FPGA 芯片的工艺一直走在芯片设计领域的前列，产品更新换代速度非常快。稳定性和可靠性是产品设计需要考虑的关键因素。厂家最新推出的 FPGA 系列产品一般都没有经过大批量应用的验证。选择这样的芯片会增加设计的风险。
而且，最新推出的 FPGA 芯片因为产量比较小，一般供货情况都不会很理想，价格也会偏高一些。如果成熟的产品能满足设计指标要求，那么最好选这样的芯片来完成设计。
例如，要用 FPGA 设计一块数据采集卡。采用 Altera 公司的 Cyclone、CyloneII 和 CycloneIII 等 3 个系列的芯片都可以完成这个功能。考虑到 Cyclone 和 CyloneII 是成熟产品，同时 CyloneII 又是 Cyclone 的升级产品，因此选择 CyloneII 是比较理想的方案。

1.5.2 尽量选择兼容性好的封装

FPGA 系统设计一般采用硬件描述语言（HDL）来完成设计。这与基于 CPU 的软件开发又有很大不同。特别是算法实现的时候，在设计之前，很难估算这个算法需要占多少 FPGA 的逻辑资源。
作为代码设计者，希望算法实现之后再选择 FPGA 的型号。但是，现在的设计流程一般都是软件和硬件并行开始设计。也就是说，在 HDL 代码设计之前，就开始硬件板卡的设计。这就要求硬件板卡具备一定的兼容性，可以兼容不同规模的 FPGA 芯片。
幸运的是，FPGA 芯片厂家考虑到了这一点。目前，同系列的 FPGA 芯片一般可以做到相同物理封装兼容不同规模的器件。例如，Xilinx 的 Spartan3 系列 FPGA，在 BGA456 封装下，可以选择 3S200、2S400、3S1000、3S1500 这 4 种型号的 FPGA。
正是因为这一点，将来的产品就具备非常好的扩展性，可以不断地增加新的功能或者提高性能，而不需要修改电路板的设计文件。

1.5.3 尽量选择一个公司的产品

如果在整个电子系统中需要多个 FPGA 器件，那么尽量选择一个公司的产品。这样的好处不仅可以降低采购成本，而且降低开发难度。因为开发环境和工具是一致的，芯片接口电平和特性也一致，便于互联互通。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

问题 Xilinx 公司和 Altera 公司的 FPGA 产品哪个好一些？

很多第一次接触 FPGA 的设计师在芯片选型的时候都有过这个疑问。其实这两个最大的 FPGA 厂家位于美国的同一座城市，人员和技术交流都很频繁，因此产品各有的优势和特色，很难说清楚谁好谁坏。

在全球不同的地区，这两家公司的 FPGA 芯片产品的市场表现会有所差别。在中国市场，两家公司可以说是平分秋色，在高校里面 Altera 的客户会略多一些。针对特定的应用，两个厂家的产品目录里面都可以找到适合的系列或者型号。

比如，针对低成本应用，Altera 公司的 Cyclone 系列和 Xilinx 公司的 Spartan3 系列是对应的。针对高性能应用，Altera 公司的 Stratix 系列和 Xilinx 公司的 Virtex 系列是对应的。所以，最终选择那个公司的产品还是看开发者的使用习惯。

联系方式

集团官网：www.hqyj.com　　嵌入式学院：www.embedu.org　　移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn　　物联网学院：www.topsight.cn　　研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路银海大厦 A 座 8 层，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218

华清远见

开发板免费领取试用　开发板免费领取试用

华清远见 FARSIGHT 始于2004 | 嵌入式培训专家

10年口碑积累，成功培养50000多名研发工程师，铸就专业品牌形象

华清远见的企业理念是不仅要做良心教育、做专业教育，更要做受人尊敬的职业教育。

《FPGA 应用开发入门与典型实例》（修订版）

作者：华清远见

专业始于专注 卓识源于远见

第 2 章　从零开始设计 FPGA 最小系统

本章目标

- 了解 FPGA 硬件系统的调试步骤和方法
- 了解 FPGA 常用接口电路的原理和设计
- 掌握 FPGA 最小系统的概念和组成电路

专业始于专注 卓识源于远见

开发板免费领取试用　开发板免费领取试用

2.1 FPGA 最小系统的概念

FPGA 最小系统是可以使 FPGA 正常工作的最简单的系统。它的外围电路尽量最少，只包括 FPGA 必要的控制电路。

一般所说的 FPGA 的最小系统主要包括：FPGA 芯片、下载电路、外部时钟、复位电路和电源。如果需要使用 NIOS II 软嵌入式处理器还要包括：SDRAM 和 Flash。一般以上这些组件是 FPGA 最小系统的组成部分。红色飓风 II 代 Altera 开发板功能框图如图 2.1 所示。

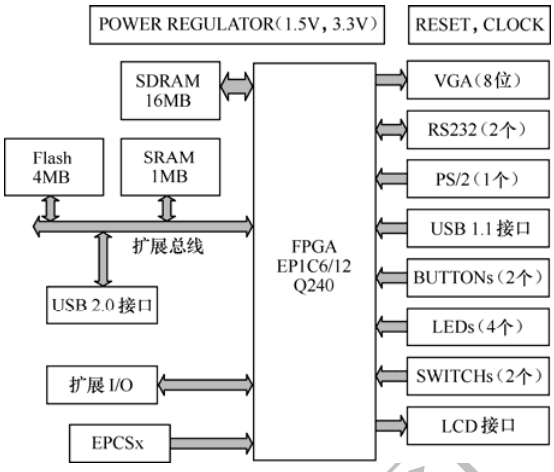


图 2.1 红色飓风 II 代 Altera 开发板功能框图

红色飓风 II 代 Altera 开发板的总线接口信号如图 2.2 所示。

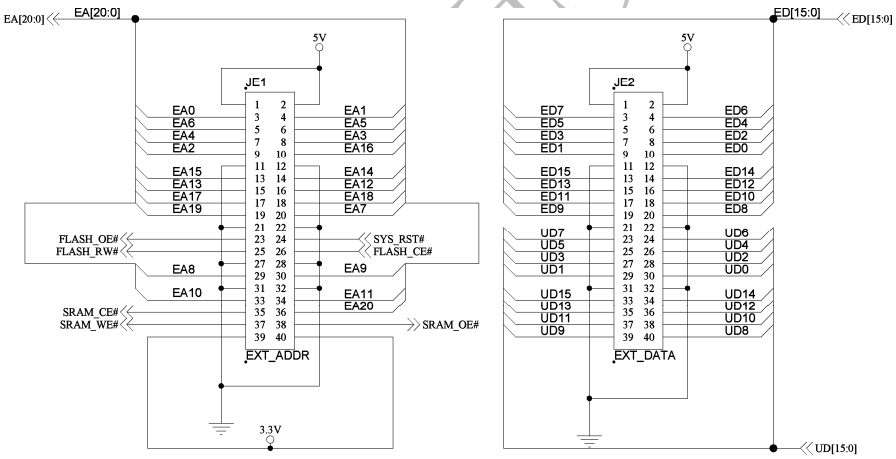


图 2.2 开发板的总线扩展接口

2.2 FPGA 最小系统电路分析

2.2.1 FPGA 管脚设计

FPGA 的管脚主要包括：用户 I/O（User I/O）、配置管脚、电源、时钟及特殊应用管脚等。其中有些管脚可有多种用途，所以在设计 FPGA 电路之前，需要认真的阅读相应 FPGA 的芯片手册。

下面以 Altera 公司的 Cyclone 系列 FPGA 为例，介绍 FPGA 的各种功能管脚。

（1）用户 I/O。

I/O*num*（LVDS*num*n）：可用作输入或输出，或者双向口，同时可作为 LVDS 差分对的负端。其中 *num* 表示管脚序号。

一般在绘制 FPGA 原理图时，将同一种功能和用途的管脚放在一个框图中，如图 2.3 所示是用户 I/O 的原理图。

开发板免费领取试用 开发板免费领取试用

（2）配置管脚。

- MSEL[1..0]：用于选择配置模式。FPGA 有多种配置模式，比如主动、被动、快速、正常、串行、并行等，可以此管脚进行选择。
- DATA0：FPGA 串行数据输入，连接至配置器件的串行数据输出管脚。
- DCLK：FPGA 串行时钟输出，为配置器件提供串行时钟。
- nCSO（I/O）：FPGA 片选信号输出，连接至配置器件的 nCS 管脚。
- ASDO（I/O）：FPGA 串行数据输出，连接至配置器件的 ASDI 管脚。
- nCEO：下载链器件使能输出。在一条下载链（Chain）中，当第一个器件配置完成后，此信号将使能下一个器件开始进行配置。下载链的最后一个器件的 nCEO 应悬空。

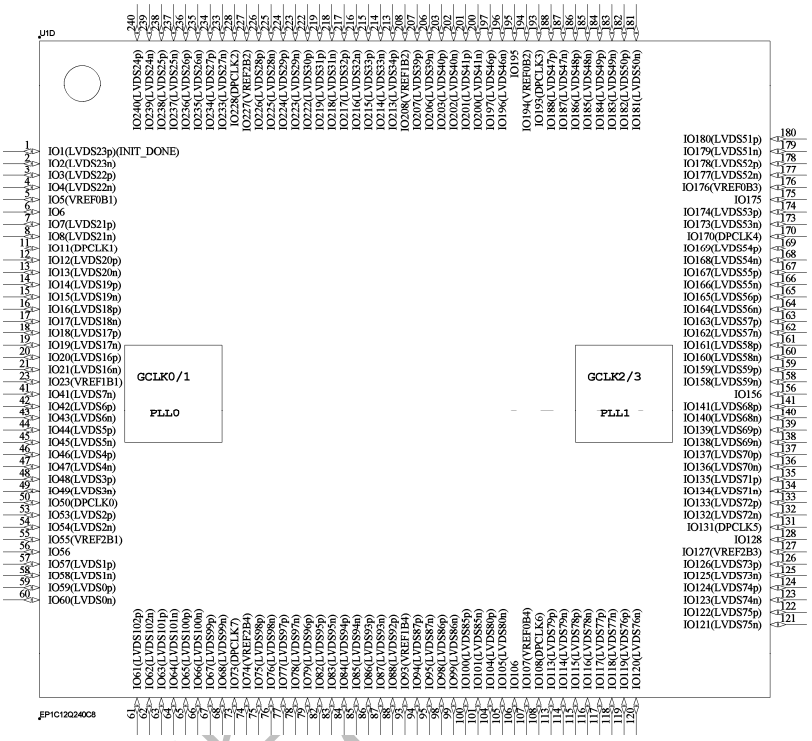


图 2.3　FPGA 用户 I/O 原理图

- nCE：下载链器件使能输入，连接至上一个器件的 nCEO。下载链第一个器件的 nCE 接地。
- nCONFIG：用户模式配置起始信号。
- nSTATUS：配置状态信号。
- CONF_DONE：配置结束信号。

如图 2.4 所示是 FPGA 配置管脚原理图。

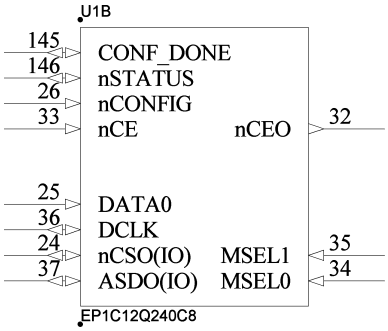


图 2.4　FPGA 配置管脚原理图

（3）电源管脚。

- VCCINT：内核电压。通常与 FPGA 芯片所采用的工艺有关，例如 130nm 工艺为 1.5V，90nm 工艺为 1.2V。
- VCCIO：端口电压。一般为 3.3V，还可以支持选择多种电压，如 5V、1.8V、1.5V 等。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

- VREF：参考电压。
- GND：信号地。

如图 2.5 所示是 FPGA 电源管脚原理图。

（4）时钟管脚。

- VCC_PLL：锁相环管脚电压，直接连 VCCIO。
- VCCA_PLL：锁相环模拟电压，一般通过滤波器接到 VCCINT 上。
- GNDA_PLL：锁相环模拟地。
- GNDD_PLL：锁相环数字地。
- CLK*num*(LVDSCLK*num*p)：锁相环时钟输入。支持 LVDS 时钟输入，p 接正端，num 表示 PLL 序号。
- CLK*num*(LVDSCLK*num*n)：锁相环时钟输入。支持 LVDS 时钟输入，n 接负端，num 表示 PLL 序号。
- PLL*num*_OUTp(I/O)：锁相环时钟输出。支持 LVDS 时钟输入，p 接正端，num 表示 PLL 序号。
- PLL*num*_OUTn(I/O)：锁相环时钟输出。支持 LVDS 时钟输入，n 接负端，num 表示 PLL 序号。

如图 2.6 所示是 FPGA 时钟管脚原理图。

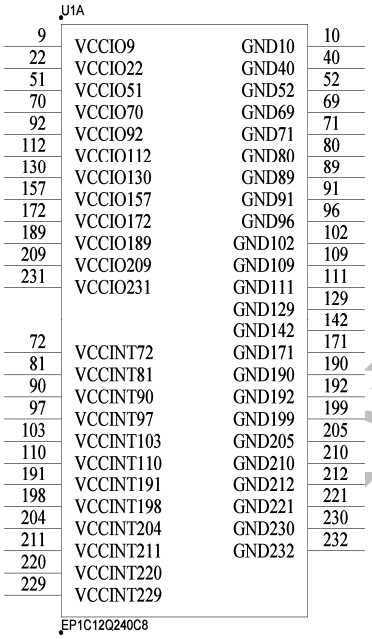


图 2.5　FPGA 电源管脚原理图

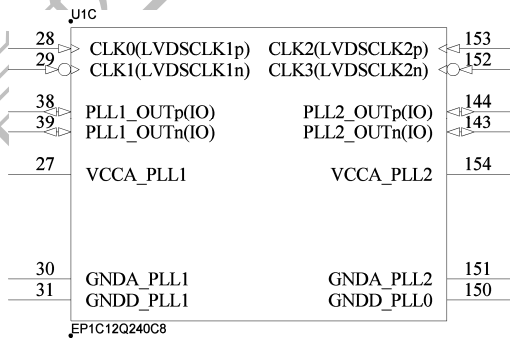


图 2.6　FPGA 时钟管脚原理图

另外，FPGA 的管脚中，有一些是全局时钟，这些管脚在 FPGA 中已经做好了时钟树。使用这些管脚作为关键时钟或信号的布线可以获得最佳性能。

（5）特殊管脚。

- VCCPD：用于选择驱动电压。
- VCCSEL：用于控制配置管脚和锁相环相关的输入缓冲电压。
- PORSEL：上电复位选项。
- NIOPULLUP：用于控制配置时所使用的用户 I/O 的内部上拉电阻是否工作。
- TEMPDIODEn/p：用于关联温度敏感二极管。

2.2.2　下载配置与调试接口电路设计

FPGA 是 SRAM 型结构，本身并不能固化程序。因此 FPGA 需要一片 Flash 结构的配置芯片来存储逻辑配置信息，用于进行上电配置。

以 Altera 公司的 FPGA 为例，配置芯片分为串行（EPCSx 系列）和并行（EPCx 系列）两种。其中 EPCx

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

系列为老款配置芯片，体积较大，价格高。而 EPCSx 系列芯片与之相比，体积小、价格低。
另外，除了使用 Altera 公司的配置芯片，也可以使用 Flash+CPLD 的方式去配置 FPGA。
在把程序固化到配置芯片之前，一般先使用 JTAG 模式去调试程序，也就是把程序下载到 FPGA 芯片上运行。虽然这种方式在断电以后程序会丢失，但是充分利用了 FPGA 的无限擦写性。
所以一般 FPGA 有两个下载接口：JTAG 调试接口和 AS（或 PS）模式下载接口。所不同的是前者下载至 FPGA，后者是编程配置芯片（如 EPCSx），然后再配置 FPGA。
如图 2.7 和图 2.8 所示分别是 JTAG 模式和 AS 模式的电路原理图。

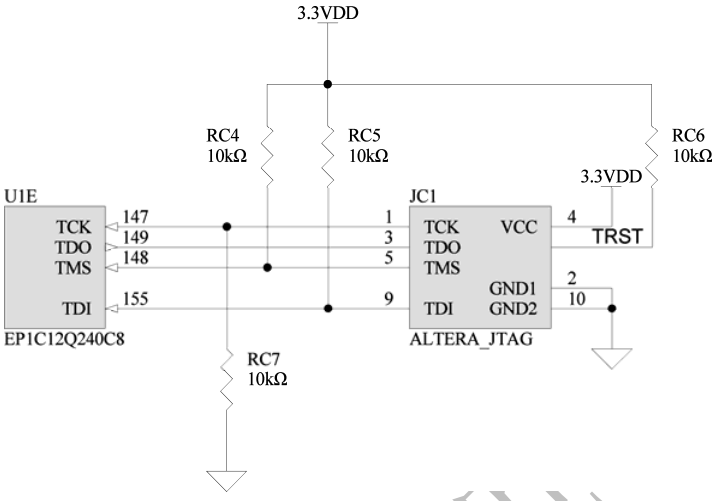


图 2.7　JTAG 模式原理图

2.2.3　高速 SDRAM 存储器接口电路设计

SDRAM 可作为软嵌入式系统的（NIOS II）的程序运行空间，或者作为大量数据的缓冲区。SDRAM 是通用的存储设备，只要容量和数据位宽相同，不同公司生产的芯片都是兼容的。
一般比较常用的 SDRAM 包括现代 HY57V 系列、三星 K4S 系列和美光 MT48LC 系列。例如，4M×32 位的 SDRAM，现代公司的芯片型号为 HY57V283220，三星公司的为 K4S283232，美光公司的为 MT48LC4M32。这几个型号的芯片可以相互替换。
SDRAM 典型电路如图 2.9 所示。

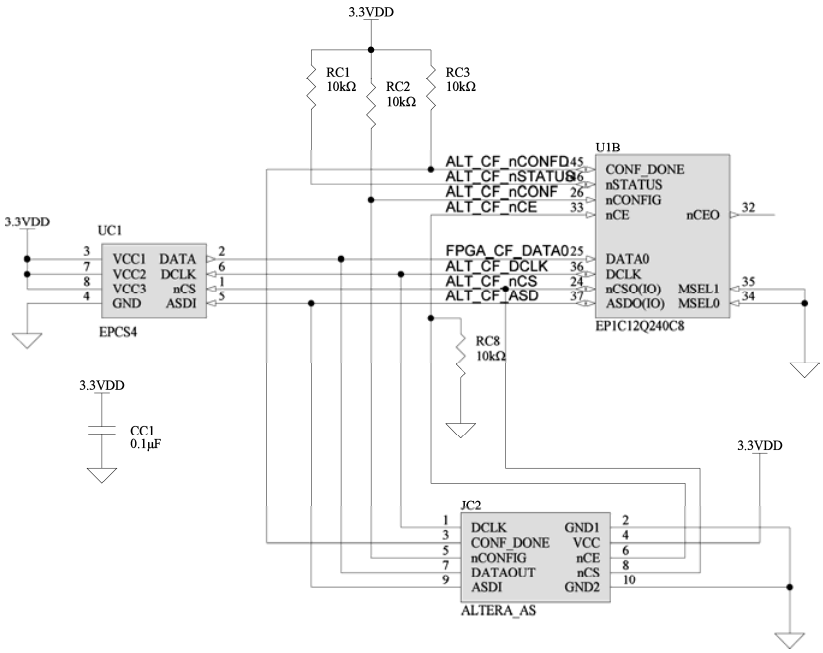


图 2.8　AS 模式原理图

开发板免费领取试用　开发板免费领取试用

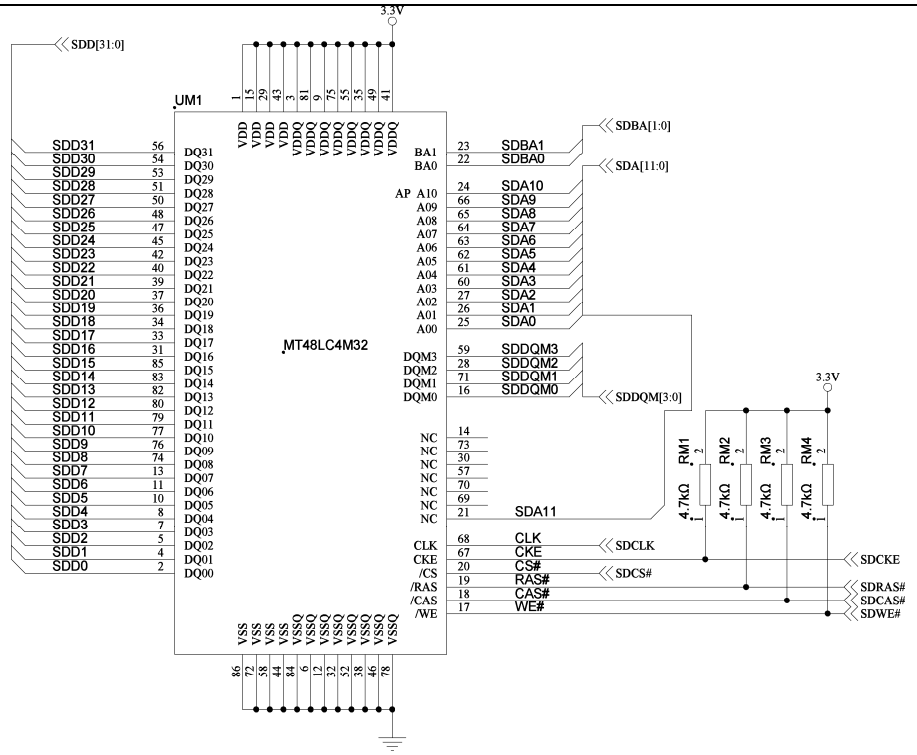


图 2.9　SDRAM 典型电路

2.2.4　异步 SRAM（ASRAM）存储器接口电路设计

由于 ASRAM 的读写时序相对比较简单，因此一般使用 SRAM 作为数据的缓冲，但其成本相对 SDRAM 高。而且作为异步设备，ASRAM 对于时钟同步的要求也不高，可以在低速下运行。ASRAM 主要为 8 位和 16 位数据宽度，用户可根据需要进行选择。

ASRAM 的典型电路如图 2.10 所示。

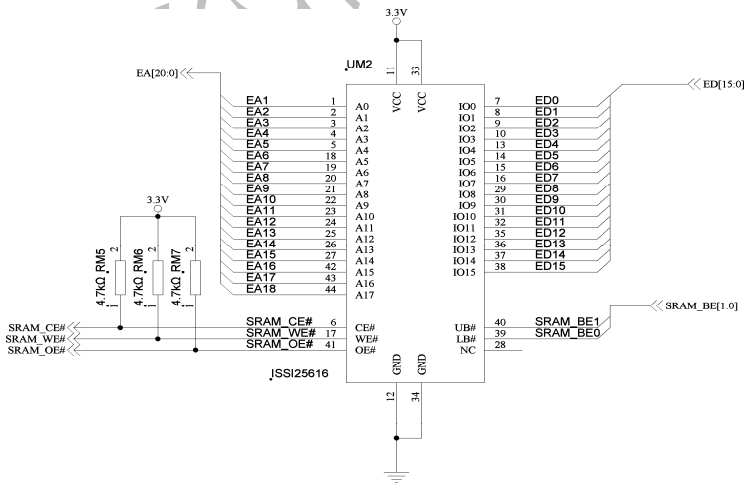


图 2.10　ASRAM 典型电路

2.2.5　Flash 存储器接口电路设计

Flash 可作为软嵌入式系统的程序存储空间，或者作为程序的固件空间。最常使用的是 AMD 公司或者 Intel 公司的 Flash。在小容量的 Flash 选择上，AMD 公司的 Flash 性价比较高，而高容量的 Flash 选择上，Intel 公司的 Flash 性价比较高。

Flash 同样也可以通过设置实现 8 位和 16 位的数据位宽，下面是几种典型的 Flash 应用。

16 位模式下的（AMD）Flash 连接如图 2.11 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

8 位模式下的（AMD）Flash 连接如图 2.12 所示。

8 位模式下（Intel）Flash 连接如图 2.13 所示。

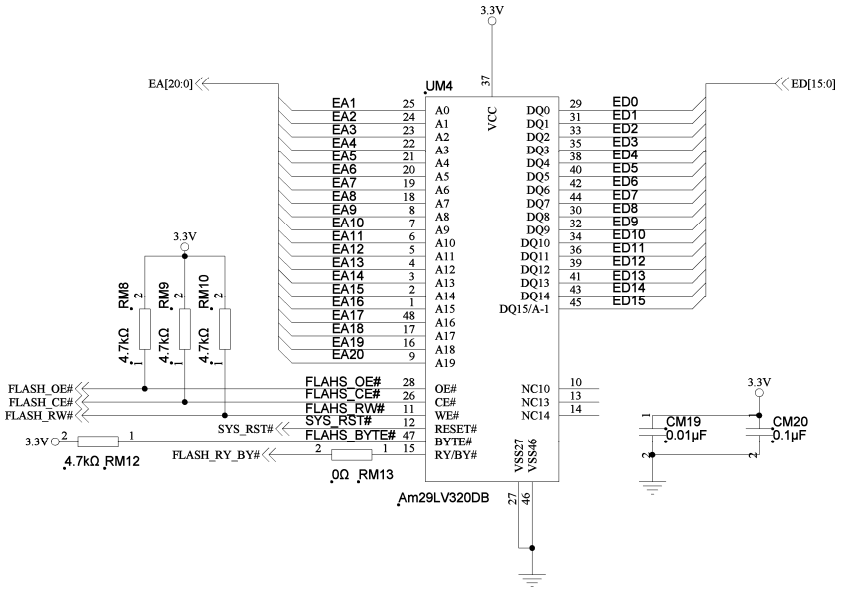


图 2.11　16 位模式下（AMD）Flash 连接

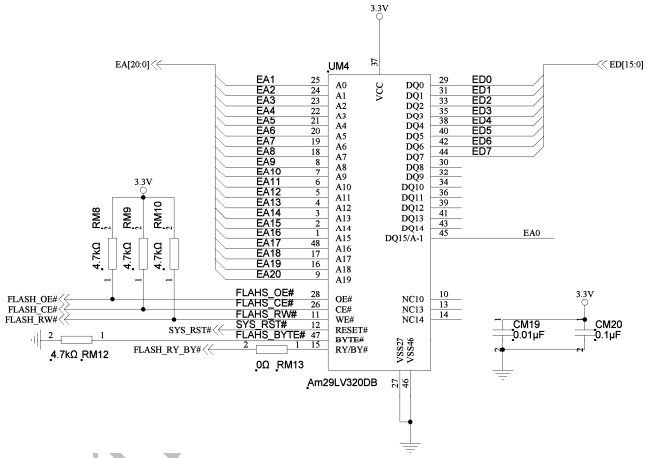


图 2.12　8 位模式下（AMD）Flash 连接

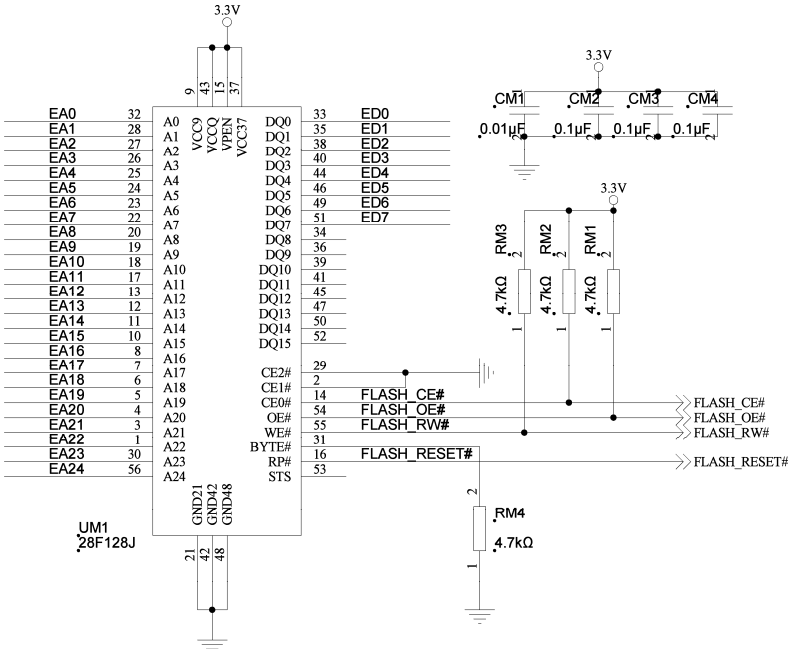


图 2.13　8 位模式下（Intel）Flash 连接

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

2.2.6 开关、按键与发光 LED 电路设计

发光 LED 参考电路如图 2.14 所示。

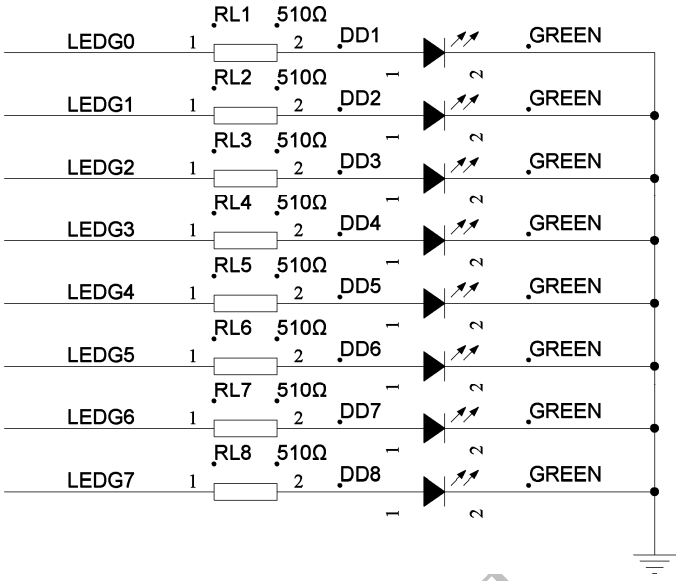


图 2.14 数码管参考电路

拨码开关参考电路如图 2.15 所示。

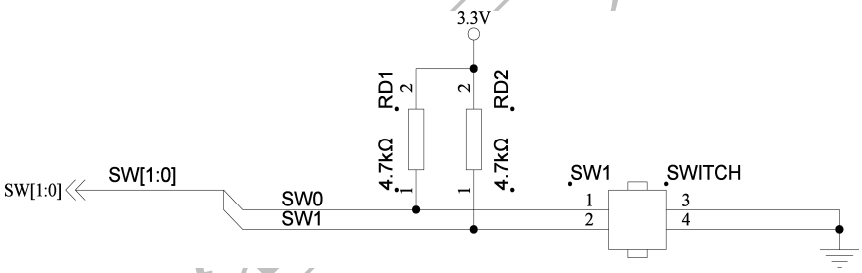


图 2.15 拨码开关参考电路

按键开关参考电路如图 2.16 所示。

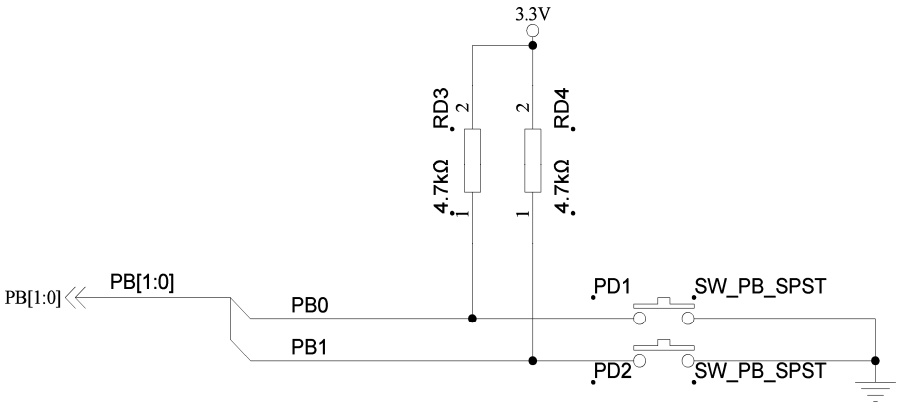


图 2.16 按键开关参考电路

2.2.7 VGA 接口电路设计

红色飓风开发板提供了 VGA 显示功能与接口，可以用普通的 VGA 电缆连接到计算机的显示器上。VGA 连接器定义如图 2.17 所示。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

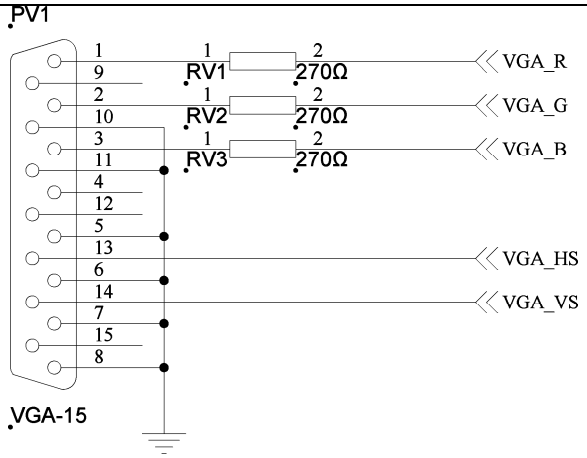


图 2.17　VGA 连接器定义

包括的信号有 Red（R）、Green（G）、Blue（B）、Horizontal Sync（水平扫描 HS）以及 Vertical Sync（垂直扫描 VS）。系统结构示意图如图 2.18 所示。

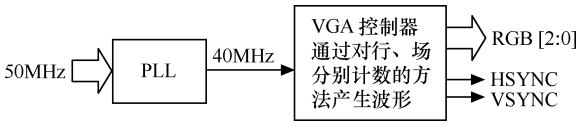


图 2.18　VGA 接口结构示意图

2.2.8　PS/2 鼠标及键盘接口电路设计

早期的 PS/2 鼠标及键盘采用 5V 电压标准，目前的 PS/2 鼠标及键盘主要采用 3.3V 电压标准，如图 2.19 所示的参考电路可以实现对两种标准的兼容。

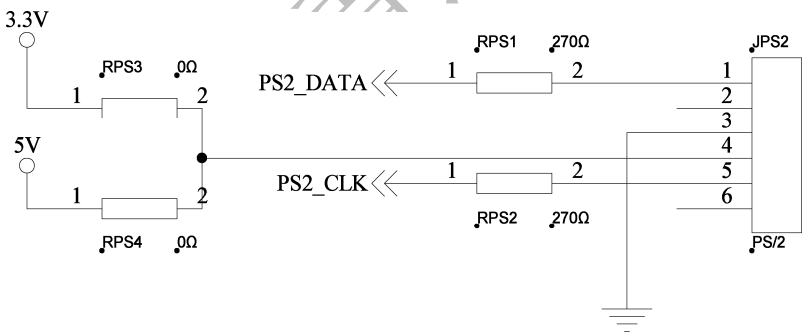


图 2.19　PS/2 参考电路

2.2.9　RS-232 串口

RS-232 接口定义如表 2.1 所示。

表 2.1　　RS-232 接口定义表

25 芯	9 芯	信号方向来自	缩　　写	描　　述
2	3	PC	TXD	发送数据
3	2	调制解调器	RXD	接收数据
4	7	PC	RTS	请求发送
5	8	调制解调器	CTS	允许发送

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

6	6	调制解调器	DSR	通信设备准备好
7	5		GND	信号地
8	1	调制解调器	CD	载波检测
20	4	PC	DTR	数据终端准备好
22	9	调制解调器	RI	响铃指示器

DTE DCE 设备信号电流方向如表 2.2 所示。

表 2.2　　DTE DCE 设备信号电流方向表

9 芯 DTE	25 芯 DTE	电 流 方 向	缩写 DCE	描述 DCE
3	2	DTE→DCE	2	3
2	3	DTE←DCE	3	2
7	4	DTE→DCE	4	7
8	5	DTE←DCE	5	8
6	6	DTE←DCE	6	6
5	7	DTE←DCE	7	5
1	8	DTE←DCE	8	1
4	20	DTE→DCE	20	4
9	22	DTE←DCE	22	9

RS-232 参考电路如图 2.20 所示。

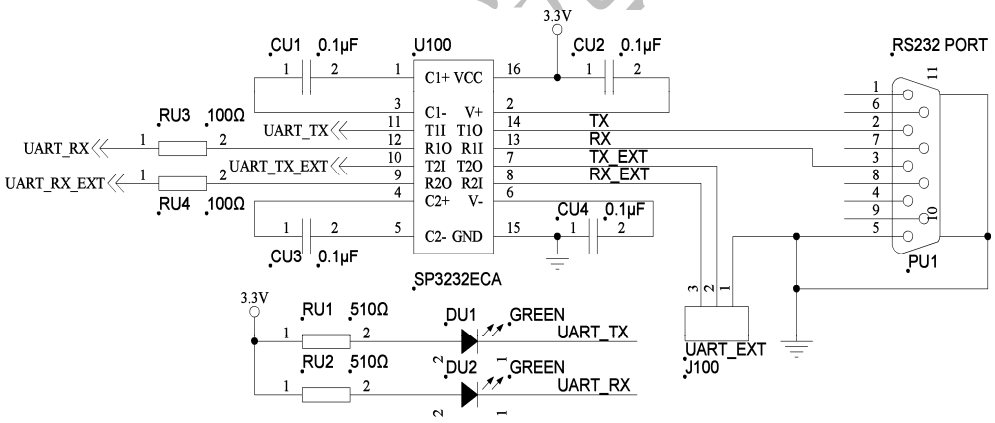


图 2.20　RS-232 参考电路

2.2.10　字符型液晶显示器接口电路设计

字符型液晶显示器电路原理图如图 2.21 所示。

第 1 脚：VSS 为地电源。

第 2 脚：VDD 接 5V 正电源。

第 3 脚：V0 为液晶显示器对比度调整端，接正电源时对比度最弱，接地电源时对比度最高，对比度过高时会产生“鬼影”，使用时可以通过一个 10kΩ的电位器调整对比度。

第 4 脚：RS 为寄存器选择线，高电平时选择数据寄存器低电平时选择指令寄存器。

第 5 脚：RW 为读写信号线，高电平时进行读操作，低电平时进行写操作。当 RS 和 RW 共同为低电平时可以写入指令或者显示地址，当 RS 为低电平、RW 为高电平时可以读忙信号，当 RS 为高电平、RW 为低电平时可以写入数据。

第 6 脚：E 端为使能端，当 E 端由高电平跳变成低电平时，液晶模块执行命令。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

第 7～14 脚：D0～D7 为 8 位双向数据线。

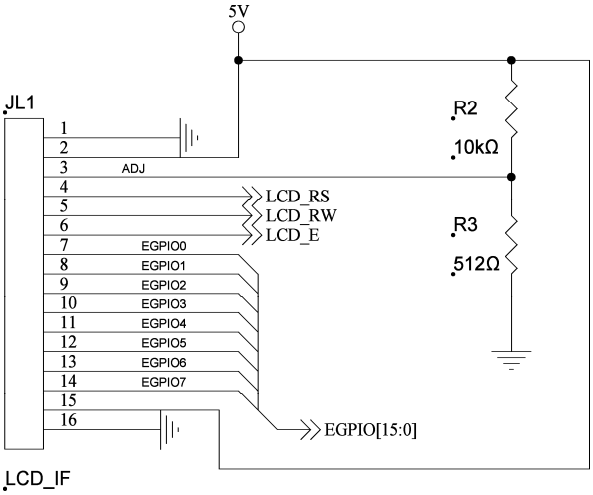


图 2.21　字符型液晶显示器电路原理图

2.2.11　USB 2.0 接口芯片 CY7C68013 电路设计

通过开发系统上的 USB 接口，可以用 USB 电缆直接将系统与主机（比如 PC 机）相连。USB 接口使用了 Cypress 公司的 CY7C68013 芯片来实现，其电路原理图如图 2.22 所示。

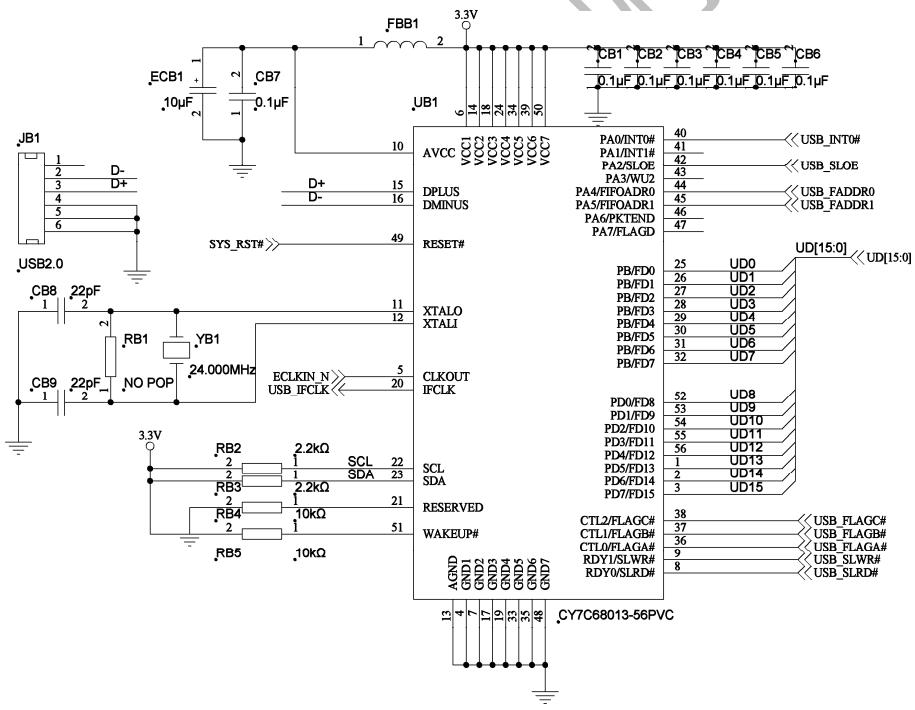


图 2.22　CY7C68013 原理图

1．芯片介绍

Cypress Semiconductor 公司的 EZ-USB FX2 是世界上第一款集成 USB 2.0 的微处理器。它集成了 USB 2.0 收发器、SIE（智能串行引擎）、增强的 8051 微控制器和可编程的外围接口。FX2 这种独创性结构可使数据传输率达到 56MB/s，即 USB 2.0 允许的最大带宽。

在 FX2 中，智能 SIE 可以硬件处理许多 USB 1.1 和 USB 2.0 协议，从而减少了开发时间和确保了 USB 的兼容性。GPIF（General Programmable Interface）和主/从端点 FIFO（8 位或 16 位数据总线）为 ATA、UTOPIA、EPP、PCMCIA 和 DSP 等提供了简单和无缝连接接口。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

CY7C68013 集成了以下特性。

（1）USB 2.0 收发器、SIE（智能串行引擎）和增强性 8051 微处理器。

（2）软件运行。8051 程序从内部 RAM 开始运行，可借助下列几种方式进行程序装载。

- 通过 USB 下载。
- 从 EEPROM 中装载。
- 通过外部存储器设备。

（3）4 个可编程（BULK/INTERRUPT/ISOCHRONOUS）端点，可选双缓冲、三缓冲和四缓冲。

（4）8 位或 16 位外部数据接口。

（5）通用可编程接口（GPIF）。

- 可以直接连接到并口，分为 8 位和 16 位。
- 可编程波形描述符和配置寄存器。
- 支持多个 Ready 输入和 Control 输出。

（6）集成标准 8051 内核，且具有下列增强特性。

- 可以达到 48MHz 时钟。
- 每条指令占 4 个时钟周期。
- 2 个 UARTS。
- 3 个定时/计数器。
- 扩展的中断系统。
- 两个数据指针。

（7）采用 3.3V 电源系统。

（8）矢量 USB 中断。

（9）独立的数据缓冲区供 SETUP 和 DATA 包控制传输。

（10）集成 I^2C 控制器，运行速度可达 100kHz。

（11）4 个 FIFO，可与 ASIC 和 DSP 等无缝连接。

（12）专门的 FIFO 和 GPIF 自动矢量中断。

（13）可用于 DSL Modems、ATA 接口、相机、Home PNA、WLAN、MP3 播放器、网络等。

2．USB 启动方式和枚举

上电时，内部逻辑会检查连接到 I^2C 总线上的 EEPROM 中的第一个字节（0xC0 或 0xC2）。如果是 0xC0，就会使用 EEPROM 中的 VID/PID/DID 来替代内部存储值；如果是 0xC2，内部逻辑就会把 EEPROM 中的内容装入到内部 RAM 中；如果没有检查到 EEPROM，FX2 就会使用内部存储的描述符来枚举。其缺省值是 0x04B4/ 0x8613/ 0xxxyy。

当首次插入 USB 时，FX2 会通过 USB 电缆自动枚举并下载固件和 USB 描述符表。然后 FX2 将再次枚举，通过下载的信息来定义设备。这两个步骤就叫做重枚举，当设备插入时它们就立即执行。

3．程序/数据存储器

（1）内部数据 RAM。

FX2 的内部数据 RAM 被分成 3 个不同的区域：低（LOW）128 字节，高（Upper）128 字节和特殊功能寄存器（SFR）空间。低 128 字节和高 128 字节是通用 RAM，SFR 包括 FX2 控制和状态寄存器。

（2）外部程序存储器和数据存储器。

FX2 有 8KB 片上 RAM（位于 0x0000～0x1FFF 范围内）和 512 字节 Scratch RAM（位于 0xE000～0xE1FF）。尽管 Scratch RAM 从物理上来说位于片内，但是通过固件可以把它作为外部 RAM 一样来寻址。FX2 保留 7.5KB（0xE200～0xFFFF）数据地址空间作为控制/状态寄存器和端点缓冲器。

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

注意 只有数据内存空间保留，而程序内存（0xE000～0xFFFF）并不保留。

4．端点缓冲区

FX2 包含 3 个 64 字节端点缓冲区和 4KB 可配置成不同方式的缓冲，其中 3 个 64 字节的缓冲区为 EP0、EP1IN 和 EP1OUT。

EP0 作为控制端点用，它是一个双向端点，既可为 IN 也可为 OUT。当需要控制传输数据时，FX2 固件读写 EP0 缓冲区，但是 8 个 SETUP 字节数据不会出现在这 64 字节 EP0 端点缓冲区中。

EP1IN 和 EP1OUT 使用独立的 64 字节缓冲区，可配置为 BULK、INTERRUPT 或 ISOCHRONOUS 传输方式，这两个端点和 EP0 一样只能被固件访问。这一点与大端点缓冲区 EP2、EP4、EP6 和 EP8 不同，这 4 个端点缓冲区主要用来和片上或片外进行高带宽数据传输而无需固件的参与。EP2、EP4、EP6 和 EP8 是高带宽、大缓冲区，它们可被设置成不同的方式来适应带宽的需求。

5．外部 FIFO 接口

EP2、EP4、EP6 和 EP8 大端点缓冲区主要用来进行高速（480Mbit/s）数据传输。可以通过 FIFO 数据接口与外部 ASIC 和 DSP 等处理器无缝连接来实现高速数据传输。它具有的通用接口有：Slave FIFO 或 GPIF（内部主）、同步或异步时钟、内部或外部时钟等。

6．中断资源

FX2 的中断结构是在一个标准 8051 单片机的基础上增强和扩展了部分中断资源，中断资源如表 2.3 所示。

表 2.3 FX 中断资源表

FX2 中断	中 断 来 源	中 断 向 量	优 先 级
IE0	INT0 Pin	0x0003	1
TF0	Timer0 Overflow	0x000B	2
IE1	INT1 Pin	0x0013	3
TF1	Timer1 Overflow	0x001B	4
RI_0 & TI_0	USART0 Rx & Tx	0x0023	5
TF2	Timer2 Overflow	0x002B	6
Resume	WAKEUP/WU2 Pin	0x0033	0
RI_1 & TI_1	USART1 Rx & Tx	0x003B	7
USBINT	USB	0x0043	8
I2CINT	I^2C BUS	0x004B	9
IE4	GPIF/FIFOs/INT4 Pin	0x0053	10
IE5	INT5 Pin	0x005B	11
IE6	INT6 Pin	0x0063	12

其中 27 个 USB 请求共享 USB 中断，14 个 FIFO/GPIF 源共享 INT4。

芯片的详细介绍与使用方法可参看芯片数据手册。

2.2.12 电源电路设计

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

电源是整个系统能够正常工作的基本保证，如果电源电路设计的不好，系统有可能不能工作，或者即使能工作但是散热条件不好，导致系统不稳定等异常情况。所以如何选用合适的电源芯片，以及如何合理地对电源进行布局布线，都是值得下大功夫研究的。

在选用电源之前要仔细阅读 FPGA 的芯片手册，一般来说 FPGA 用到的管脚和资源越多，那么所需要的电流就越大，当电路启动时 FPGA 的瞬间电流也比较大。通过数据手册中提供的电气参数，确定 FPGA 最大需要多大的电流才能工作。

下面是几种常使用的 FPGA 参考电源。

- AS117 可以提供 1A 电流，线型电源（适用 144 管脚以下、5 万逻辑门以下的 FPGA）。
- AS2830(或 LT1085/6)可以提供 3A 电流，线性电源(适用 240 管脚以下、30 万逻辑门以下的 FPGA)。
- TPS54350 可以提供 3A 电流，开关电源（适用大封装大规模的高端 FPGA）。

AS2830 电源应用电路如图 2.23 所示。

对于线性电源芯片，输出电压和输入电压的关系为：Vout= (1+RP3/RP2)×Vref。

Vref 一般是 1.25V，输出假定输入 Vin 为 5V，Vout 为 1.5V，那么 RP2/RP3=1/5，而 RP3 一般要求 100～150Ω，那么可以选 RP3=100Ω，RP2=500Ω。如果采用了固定电平输出的芯片，只需要把 RP3 焊 0Ω，RP2 不焊接即可。

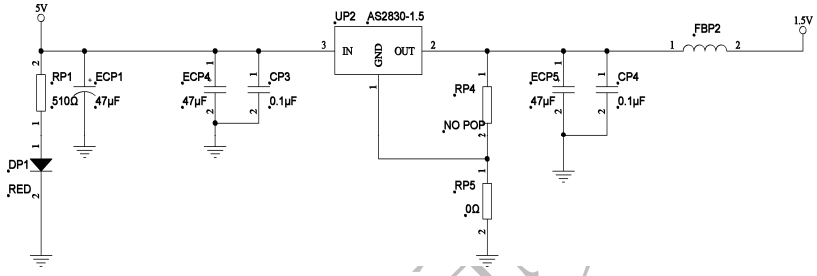


图 2.23　AS2830 电源应用电路

2.2.13　复位电路设计

一般复位电路采用的是低电平复位，只有个别单片机采用高电平复位方式。

常见的电平复位电路分为芯片复位和阻容复位。前者的复位信号比较稳定，而后者容易出现抖动。因此在成本允许的范围内我们一般推荐使用芯片复位。

常用的芯片复位有 MAX708S/706S 系列，它们可提供高、低电平两种复位方式和电源监控能力（监控电源电压低到一定程度自动复位）。

IMP811 是一款比较低廉的复位芯片，只有低电平复位功能，但是其体积非常小。

阻容复位典型连接电路如图 2.24 所示。

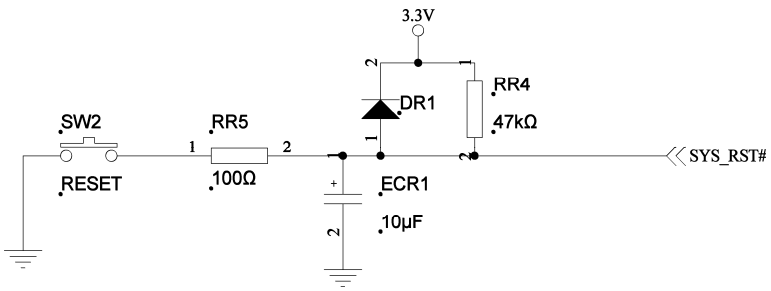


图 2.24　阻容复位典型连接电路

MAX708S 典型连接电路如图 2.25 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

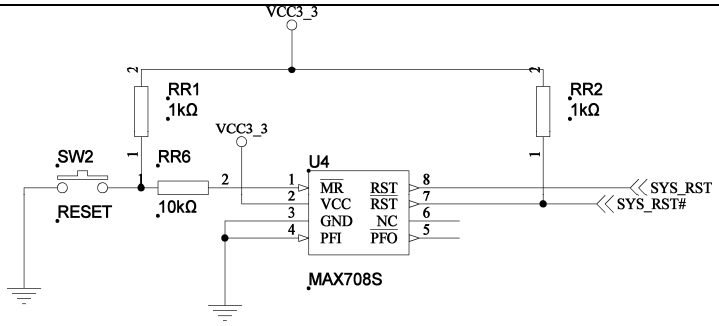


图 2.25　MAX708S 典型连接电路

IMP811 典型连接电路如图 2.26 所示。

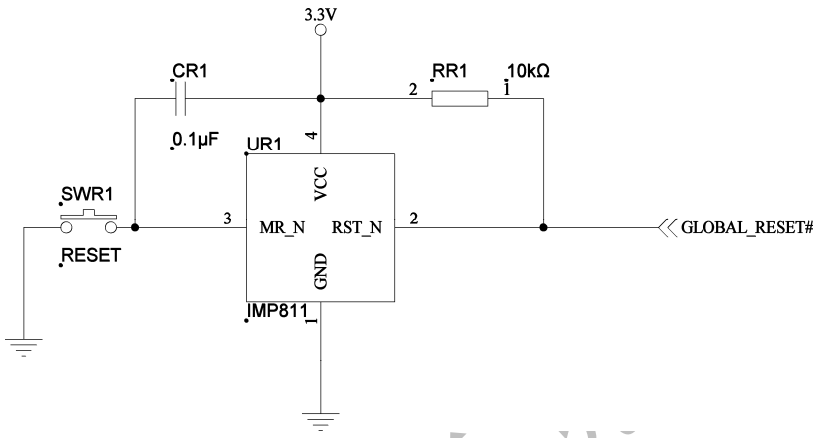


图 2.26　IMP811 典型连接电路

2.2.14　时钟电路设计

时钟电路典型连接如图 2.27 所示。

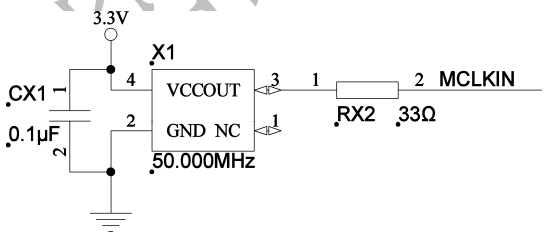


图 2.27　时钟电路典型连接

2.3　FPGA 硬件系统的设计技巧

FPGA 的硬件设计不同于 DSP 和 ARM 系统，比较灵活和自由。只要设计好专用管脚的电路，通用 I/O 的连接可以自己定义。因此，FPGA 的电路设计中会有一些特殊的技巧可以参考。

2.3.1　管脚兼容性设计

前面的内容提到过，FPGA 在芯片选项的时候要尽量选择兼容性好的封装。那么，在硬件电路设计时，就要考虑如何兼容多种芯片的问题。

例如，红色飓风 II 代－Altera 的开发板就是兼容了 EP1C6Q240 和 EP1C12Q240 两个型号的 FPGA。这两个芯片有 12 个 I/O 管脚定义是不同的。在 EP1C6Q240 芯片上，这 12 个 I/O 是通用 I/O 管脚，而在 EP1C12Q240 芯片上，它们是电源和地信号。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

为了能保证两个芯片在相同的电路板上都能工作，我们就必须按照 EP1C12Q240 的要求来把对应管脚连接到电源和地平面。因为，通用的 I/O 可以连接到电源或者地信号，但是电源或者地信号却不能作为通用 I/O。在相同封装、兼容多个型号 FPGA 的设计中，一般的原则就是按照通用 I/O 数量少的芯片来设计电路。

2.3.2 根据电路布局来分配管脚功能

FPGA 的通用 I/O 功能定义可以根据需要来指定。在电路图设计的流程中，如果能够根据 PCB 的布局来对应的调整原理图中 FPGA 的管脚定义，就可以让后期的布线工作更顺利。

例如，如图 2.1 所示，SDRAM 芯片在 FPGA 的左侧。在 FPGA 的管脚分配的时候，应该把与 SDRAM 相关的信号安排在 FPGA 的左侧管脚上。这样，可以保证 SDRAM 信号的布线距离最短，实现最佳的信号完整性。

2.3.3 预留测试点

目前 FPGA 提供的 I/O 数量越来越多，除了能够满足设计需要的 I/O 外，还有一些剩余 I/O 没有定义。这些 I/O 可以作为预留的测试点来使用。

例如，在测试与 FPGA 相连的 SDRAM 工作时序状态的时候，直接用示波器测量 SDRAM 相关管脚会很困难。而且 SDRAM 工作频率较高，直接测量会引入额外的阻抗，影响 SDRAM 的正常工作。

如果 FPGA 有预留的测试点，那么可以将要测试的信号从 FPGA 内部指定到这些预留的测试点上。这样既能测试到这些信号的波形，又不会影响 SDRAM 的工作。

如果电路测试过程中发现需要飞线才能解决问题，那么这些预留的测试点还可以作为飞线的过渡点。

2.4 FPGA 硬件系统的调试方法

随着 FPGA 芯片的密度和性能不断提高，调试的复杂程度也越来越高。BGA 封装的大量使用更增加了板子调试的难度。所以在调试 FPGA 电路时要遵循一定的原则和技巧，才能减少调试时间，避免误操作损坏电路。

一般情况下，可以参考以下步骤进行 FPGA 硬件系统的调试。

（1）首先在焊接硬件电路时，只焊接电源部分。使用万用表进行测试，排除电源短路等情况后，上电测量电压是否正确。

（2）然后焊接 FPGA 及相关的下载电路。再次测量电源地之间是否有短路现象，上电测试电压是否正确，然后将手排除静电后触摸 FPGA 有无发烫现象。

如果此时出现短路，一般是去耦电容短路造成，所以在焊接时一般先不焊去耦电容。FPGA 的管脚粘连也可能造成短路，这时需要对比电路图和焊接仔细查找有无管脚粘连。

如果出现电压值错误，一般是电源芯片的外围调压电阻焊错，或者电源的承载力不够造成的。若是后者，则需要选用负载能力更强的电源模块进行替换。如果 FPGA 的 I/O 管脚与电源管脚粘连，也可能出现电压值错误的现象。

如果出现 FPGA 发烫，一般是出现总线冲突的现象。这种情况下需要仔细检查外围总线是否出现竞争问题。特别是多片存储器共用总线时候，比如 ASRAM 和 Flash 芯片复用一套总线，如果片选信号同时有效就出现总线的冲突。

（3）以上步骤均通过后，将电路板上电运行。然后把下载电缆接到 JTAG 接口上，在主机中运行 Quartus II 软件，并打开 Programmer 编程器，单击其中的“Auto Detect”按钮进行 FPGA 下载链路自动检测。若能正确检测到 FPGA，说明配置电路是正确连接的。

自动检测 FPGA 下载链路如图 2.28 所示。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

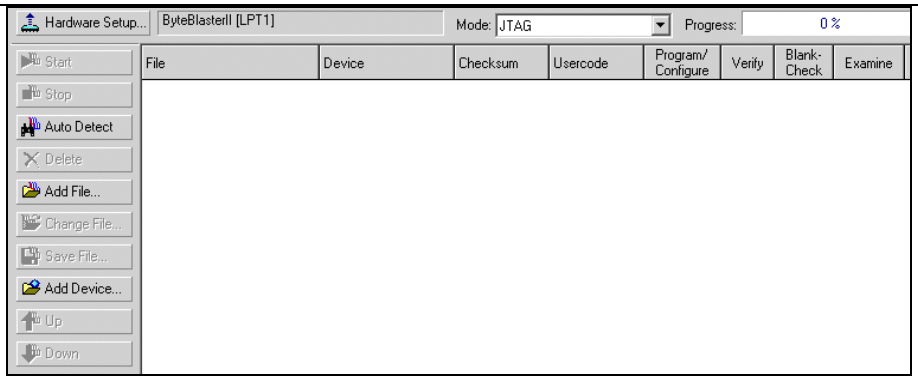


图 2.28　自动检测 FPGA 下载链路

（4）焊接时钟电路、复位电路及数码管电路，并向 FPGA 下载一个数码管跑马灯程序。若程序能够正确运行，说明 FPGA 已经可以正常工作了。

（5）最后焊接所有其他电路，并进行整体功能测试。

2.5　典型实例 1：在 Altera 的 FPGA 开发板上运行第一个 FPGA 程序

2.5.1　实例的内容及目标

1．实例的主要内容

本节旨在通过给定的工程实例——“蜂鸣器播放梁祝音乐”来熟悉 Altera Quartus II 软件的基本操作、设计、编译及仿真流程。同时使用基于 Altera FPGA 的开发板将该实例进行下载验证，完成工程设计的硬件实现，熟悉 Altera FPGA 开发板的使用及配置方式。

在本节中，将主要讲解下面一些知识点。

- Quartus II 工程创建及属性设置。
- Quartus II 源文件设计输入方式。
- Quartus II 约束设计。
- Quartus II 工程编译。
- Quartus II 功能仿真。
- Quartus II 时序仿真。
- Quartus II 硬件下载。

通过这些知识点，按照下面提供的训练流程，读者可以迅速地掌握使用 Quartus II 软件进行 FPGA 开发的方法。

2．实例目标

通过详细的流程讲解，读者应达到下面的目标。

- 熟悉 Quartus II 软件的操作环境。
- 熟悉 Quartus II 软件开发 FPGA 的基本流程。
- 可独立使用 Quartus II 软件开发新工程。

2.5.2　平台简介

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

本实例基于红色飓风 II 代 Altera 板，在此开发板上集成了 Altera 的 Cyclone 一代 FPGA 及相关的丰富外设资源。

此开发板更加详细的信息、扩展附件及使用方法，可以到红色飓风的官方网站：http://www.fpgadev.com 获取更多的信息。

本实例将在这个开发板上对 Altera 的 FPGA 设计流程做一个全面的介绍，依照此例程的流程便可在该开发板上运行一个 FPGA 程序。

如图 2.29 所示为此开发板的外观图。

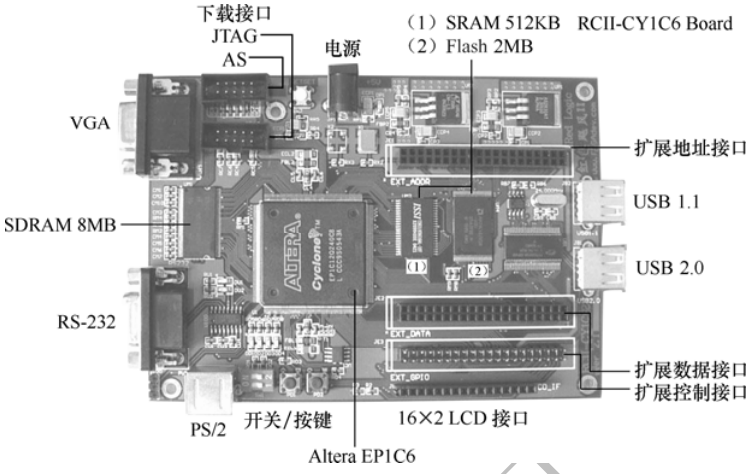


图 2.29 红色飓风 II 代 Altera 板外观图

2.5.3 实例详解

本节将使用图解的方式将整个流程一步一步展现给读者，使读者能够轻松掌握 Quartus II 的开发流程。

1．工程创建及属性设置

（1）启动 Quartus II 软件。

安装 Quartus II 软件后，在桌面或者程序组中启动 Quartus II 软件。

（2）打开新工程向导。

启动软件后，选择“File”菜单的“New Project Wizard”选项，打开新建工程向导，如图 2.30 所示。在新建工程向导的 “介绍”页面中，单击“Next”按钮进入下一页。

（3）设置工程属性。

如图 2.31 所示，在新建工程向导的第一页对工程工作目录、工程名称以及顶层模块名称进行设置。

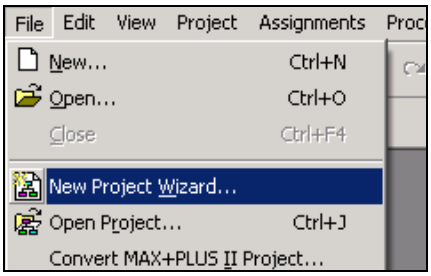


图 2.30 新建工程

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

图 2.31　设置工程属性

（4）添加设计文件。

在新建工程向导的第二页选择为工程添加设计文件，如图 2.32 所示。

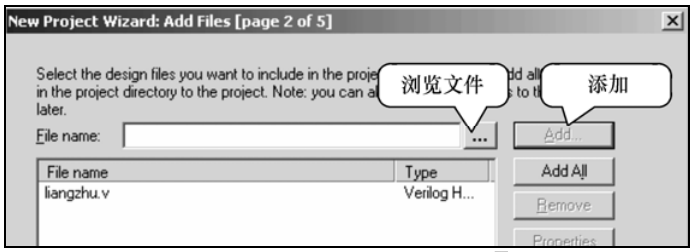
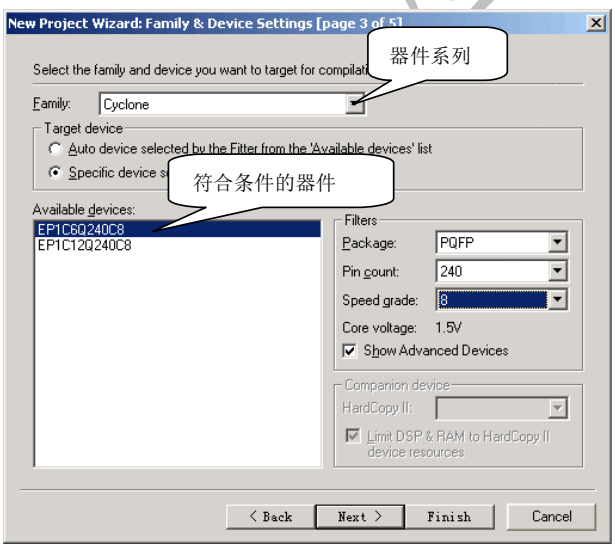


图 2.32　添加设计文件

（5）选择 FPGA 器件。

在新建工程向导的第三页，为工程配置相应的器件型号和参数，如图 2.33 所示。选取的器件型号将在完全编译时将工程设计映射到对应的器件逻辑资源上。



封装
管脚数
速度等级

图 2.33　选择器件型号

（6）完成工程创建。

配置完器件属性后，选择 Quartus II 默认的综合工具、仿真工具及时序分析工具，完成工程的创建。

选择软件左侧的工程浏览器的 按钮来管理已添加的文件。如图 2.34 所示，左边为工程层次窗口，右边为设计文件窗口。

开发板免费领取试用　开发板免费领取试用

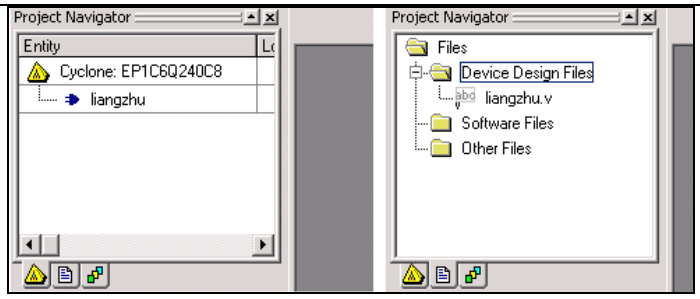


图 2.34　工程结构窗口

2．设计输入

（1）添加设计文件。

如果在创建工程时没有为工程添加设计文件，可以选择“Project”菜单的“Add/Remove Files in Project”选项，为工程添加设计文件。在本实例中，读者可向工程添加实例代码中的 liangzhu.v 文件。

添加后，在工程浏览器中双击 liangzhu.v 图标，即可查看该 Verilog 设计文件，如图 2.35 所示。

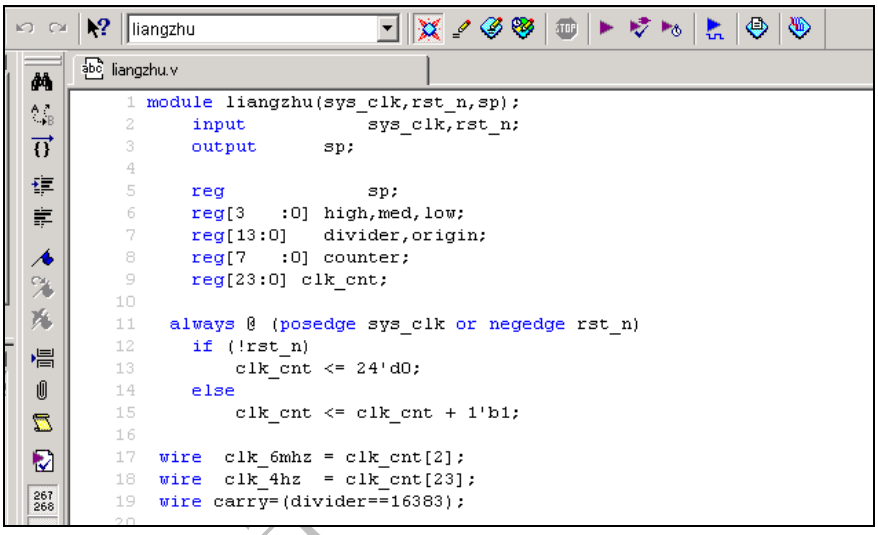


图 2.35　Verilog 设计文件

（2）配置器件属性。

同样的，如果在创建工程时没有为工程配置器件型号及属性，可以在工程浏览器的工程实体图标上单击右键，选择“Device”选项，为工程配置器件属性，如图 2.36 所示。

若在建立工程时已经配置好，则可单击“Device & Pin Options…”按钮，进一步设置器件的相关属性，如图 2.37 所示。

图 2.36　器件选择

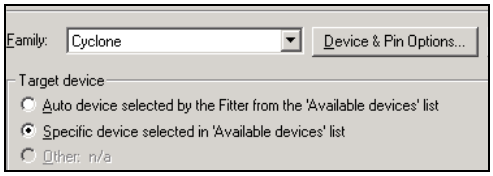


图 2.37　器件属性配置

在如图 2.38 所示的对话框中，可对 FPGA 所使用的配置芯片及未用管脚等进行配置。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

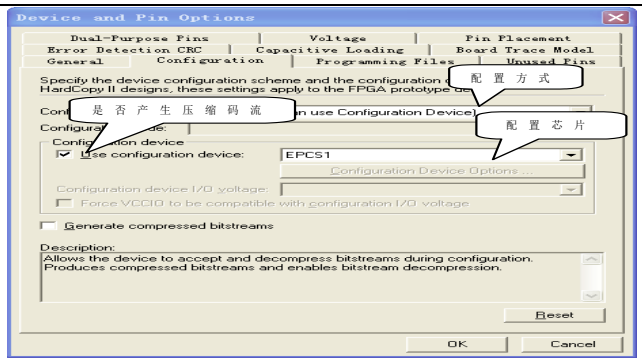


图 2.38　指定配置模式

选择“Unused Pins”选项卡，将不使用的管脚配置为三态，如图 2.39 所示。

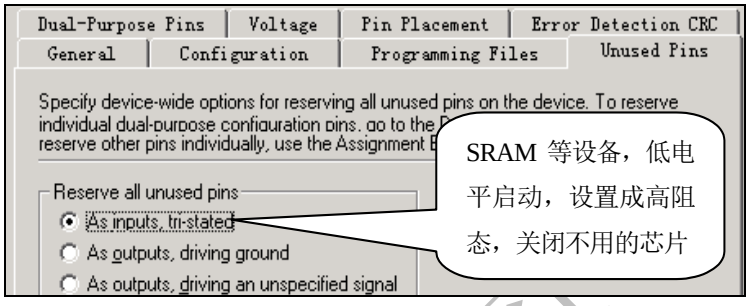


图 2.39　未用管脚置三态

（3）分析工程。

编写设计输入文件后，首先要进行工程分析，目的是为了检查设计输入的语法。单击“Start Analysis & Synthesis”按钮分析工程，如图 2.40 所示。

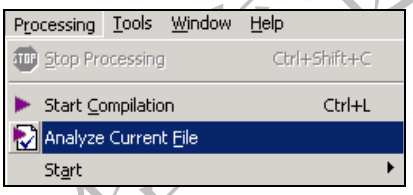


图 2.40　分析工程选项

查看信息栏，修改所有出现的错误，直到分析通过，如图 2.41 所示。

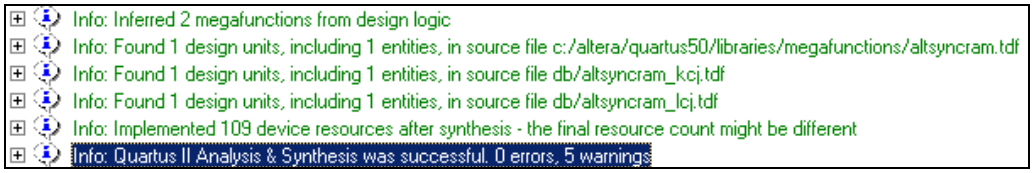


图 2.41　分析工程结果，没有报错

3．约束设计

（1）管脚分配。

管脚分配的作用在于将设计输入文件的端口与实际的器件进行映射，实现设计输入模块端口在实际器件管脚上的实例化。在 Quartus II 软件中可以在管脚分配主窗口中对管脚的分配进行设置，如图 2.42 所示。

	To	Location	I/O Bank	I/O Standard	General Function
1	rst_n	PIN_131	3	LVTTL	Row I/O
2	sp	PIN_62	4	LVTTL	Column I/O
3	sys_clk	PIN_153	3	LVTTL	Dedicated Clock
4	<<new>>	<<new>>			

开发板免费领取试用　开发板免费领取试用

图 2.42　管脚分配

通过工具栏中的快捷按钮可以帮助用户快速的进行管脚分配，如图 2.43 所示。

通过管脚分配主窗口中的管脚信息可以看到管脚的 I/O 种类、管脚的序号、Bank 所在的位置、I/O 使用的电压标准等，如图 2.44 所示。

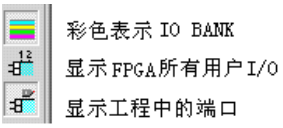


图 2.43　指定管脚快捷按钮

1	rst_n	PIN_131
2	sp	PIN_62
3	sys_clk	PIN_153

图 2.44　管脚信息

（2）其他约束。

除了对工程的管脚进行约束外，Quartus II 软件还允许用户对其他的一些约束进行设置，例如面积约束、速度约束、时钟约束、资源约束等。这些约束都属于较为严格的工程设置，在本实战训练中无需考虑。

4．编译工程

（1）完全编译。

有了完整的设计输入、完整的约束条件后，就可以对工程进行完全编译了，如图 2.45 所示是完全编译的选项。

根据工程复杂度的不同，Quartus II 在进行完全编译时所消耗的时间差异也很大。在工程浏览器中显示了编译的类别及进度，帮助用户了解编译的进程。

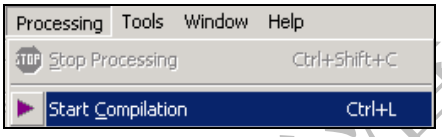


图 2.45　完全编译选项

Module	Progress
Full Compilation	29 %
Analysis & Synthesis	100 %
Fitter	16 %
Assembler	0 %
Timing Analyzer	0 %

图 2.46　编译状态

若在编译过程中出现错误提示，用户可在信息栏中查看错误的信息，修改所有存在的错误后重新进行编译，直到能够无错误地完成编译，如图 2.47 所示。

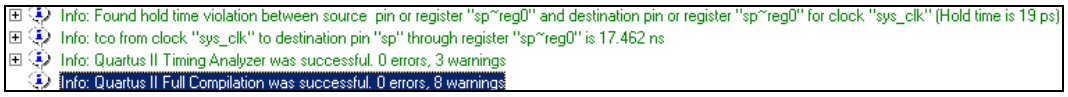


图 2.47　编译结果信息栏

（2）编译报告。

Quartus II 编译结束后，为用户提供了一个完整而详细的编译报告。通过该报告，用户可以查看工程使用资源的情况及系统可以达到的性能，如图 2.48 所示。

5．功能仿真

（1）建立仿真文件。

选择 Quartus II 软件“File”菜单的“New”选项，打开新建其他文件对话框，选择新建波形图文件，如图 2.49 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

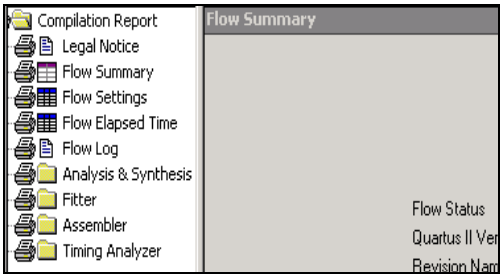


图 2.48　编译结果报告

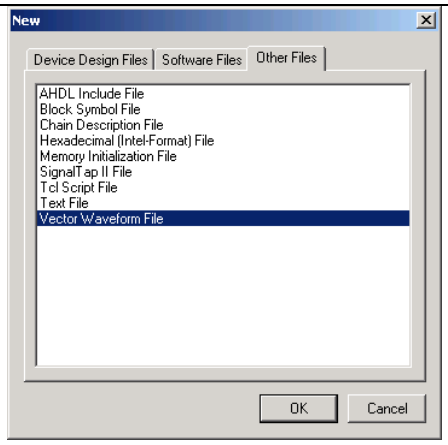


图 2.49　新建波形图

（2）添加观察信号。

用户可以通过双击波形窗口中的空白区域为工程添加需要观察的信号，如图 2.50 所示。

图 2.50　仿真波形窗口

在打开的插入节点或总线对话框中，打开“Node Finder…”（节点查找器），如图 2.51 所示。通过节点查找器，用户可以方便地选择需要观察的信号。

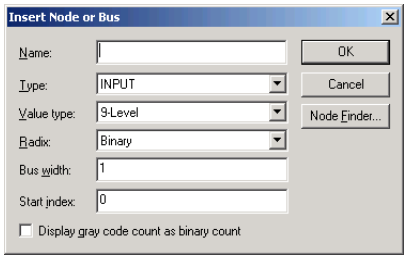


图 2.51　插入节点或总线对话框

选择 Filter：“Pins：all”，然后点击 list 列出所用输入/输出端口，如图 2.52 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

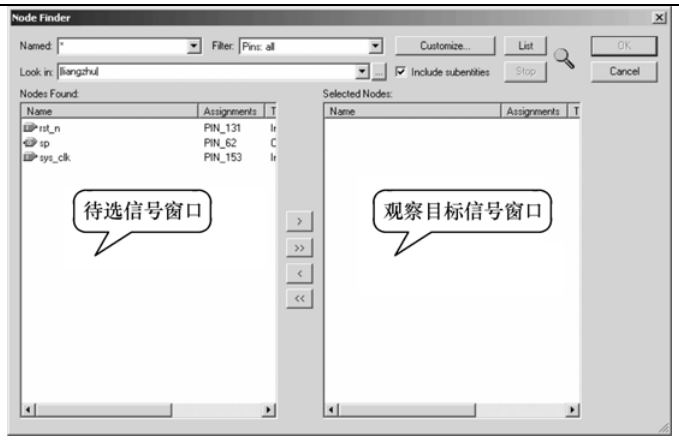


图 2.52　节点查找器

选中所有信号，单击 › 按钮，即可将选中信号加入观察目标窗口中，如图 2.53 所示。

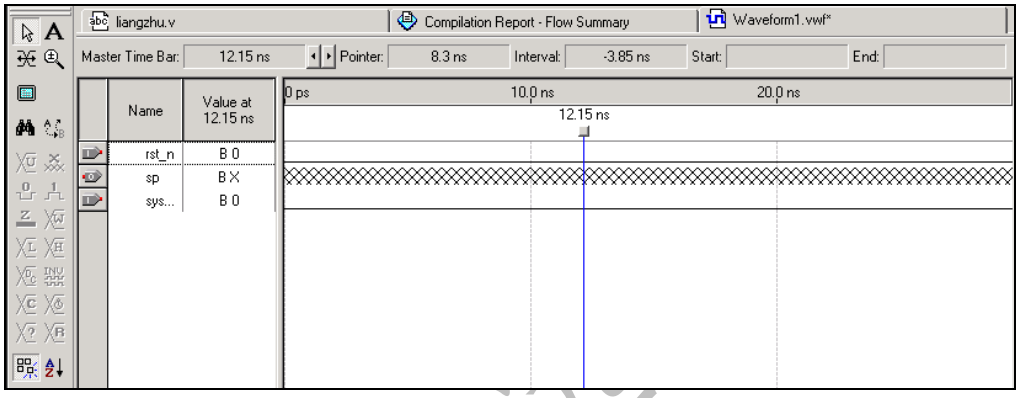


图 2.53　已添加好的信号

（3）设置仿真时间最小间隔。

考虑到硬件环境提供的是 50MHz 的时钟，即时钟周期为 20ns，设置“Grid Size”为 20ns。选择“Edit”菜单的“Grid Size”选项进行设置，如图 2.54 所示。

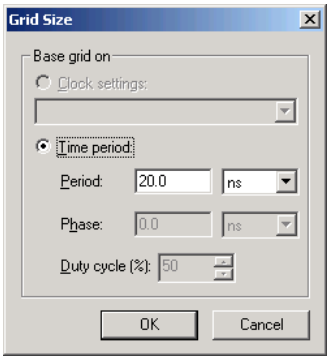


图 2.54　设置仿真时间间隔

（4）设置仿真时间长度。

选择“Edit”菜单的“End Time”选项，设置仿真时间长度为 10ms，如图 2.55 所示。

（5）添加激励信号。

选中 sys_clk 信号，单击时钟按钮，将该信号设置为时钟波形，周期为 20ns，如图 2.56 所示。此信号代表了系统时钟。

图 2.55　设置仿真结束时间

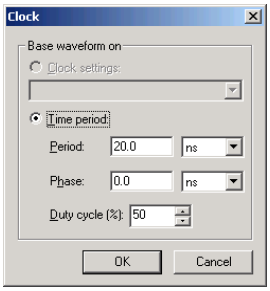


图 2.56　设置时钟

选中 rst_n 信号，单击高电平按钮，将该信号设置为 1。此信号代表了复位信号。

添加激励信号后，选择保存文件，此时的波形如图 2.57 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

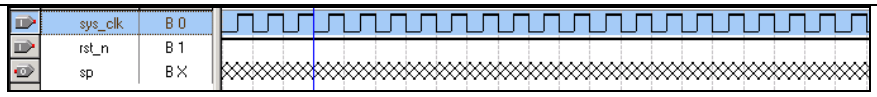


图 2.57　以设置好的输入波形

（6）生成功能仿真网表。

选择“Tools”菜单的“Simulator Tool”选项，打开仿真器，如图 2.58 所示。

将仿真器的仿真模式设置为“Functional”（功能仿真），如图 2.59 所示。

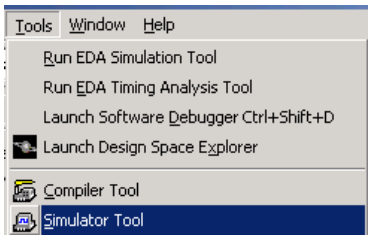


图 2.58　仿真器选项

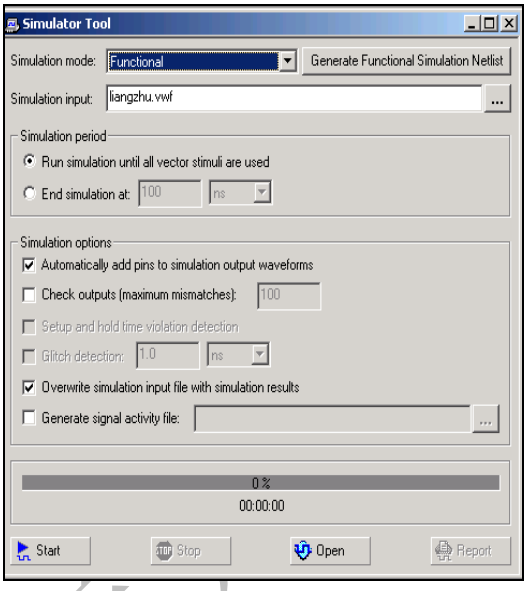


图 2.59　仿真对话框

单击“Generate Functional Simulation Netlist”按钮产生仿真网表。仿真网表是将工程设计文件进行编译及映射后生成的用于进行仿真的文件，仿真器根据仿真网表进行仿真，直接反应了工程设计文件的真实情况。

（7）开始功能仿真。

单击“Start”按钮开始进行功能仿真，如图 2.60 所示，可以看到仿真的进度，根据工程的复杂度，仿真过程所消耗的时间也有所不同。

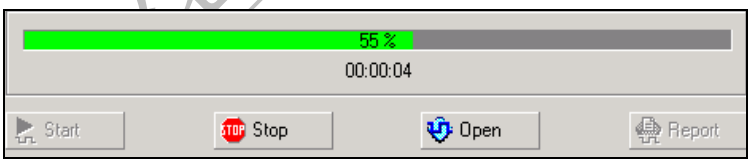


图 2.60　仿真进度

仿真完成后单击“open”按钮打开仿真结果，如图 2.61 所示。

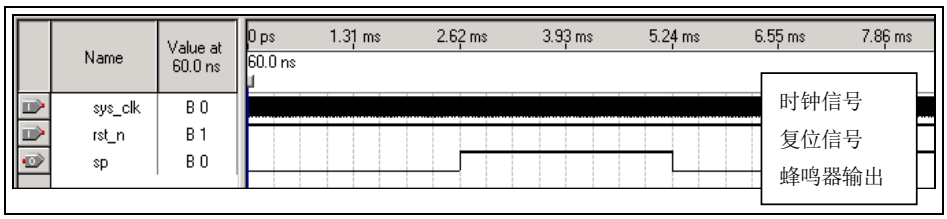


图 2.61　仿真波形结果

6．时序仿真

功能仿真后，如果波形没有问题，开始做时序仿真，检查波形延时对设计是否有影响。

选择“Tools”菜单的“Simulator Tool”选项，打开仿真器，选择仿真模式为“Timing”，即时序仿真模式，如图 2.62 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

图 2.62　选择时序仿真

单击“Start”按钮，开始时序仿真。时序仿真比功能仿真要慢一些。
仿真完成后，查看仿真结果。通过波形可以看到产生了 7.06ns 的延时，如图 2.63 所示。

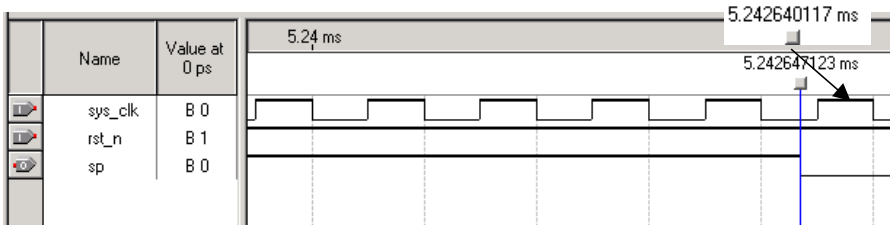


图 2.63　查看仿真结果

可以看到，功能仿真并不包含延迟，而时序仿真则会根据具体的器件参数配置及资源使用情况将延迟仿真出来。功能仿真主要用于验证工程设计文件逻辑的正确性，而时序仿真更能体现真实的硬件运行过程中设计文件的执行过程。

7．下载程序

仿真验证结束后，用户就可以将工程下载到实际的开发板上进行验证了。
选择“Tools”菜单的“Programmer”选项，打开下载器界面，如图 2.64 所示。

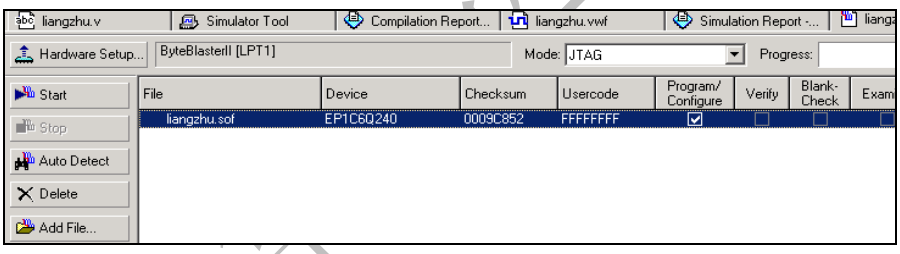


图 2.64　下载界面

首先单击“Hardware Setup”按钮，选择下载电缆，如 ByteBlasterII（LPT1），即打印机接口下载线，然后选择下载模式为 JTAG（在线调试）模式或者 AS（固化至 EPCS 配置芯片）模式，并选择对应的下载文件。连接好下载电缆后，选中“Program/Configure”即可下载。

8．结果检验

下载结束后，用户应该可以在开发板的蜂鸣器上听到演奏的梁祝乐曲。如果采用的是 JTAG 下载模式，那么将开发板断电后，重新上电，则乐曲不会再继续演奏；若使用的是 AS 下载模式，则乐曲还会再继续演奏。
若下载后没有听到乐曲，用户应检查几个主要容易出问题的步骤。例如，设计输入是否完整，管脚是否分配，下载电缆是否正确连接等。

2.5.4　小结

上述流程就是一个完整的 FPGA 设计流程。虽然此实例实现的功能比较简单，但对于初学者来说，是一个不错的入门实例。其主要目的是让初学者对 FPGA 的设计有一个初步的了解。通过该实例来熟悉 Quartus II

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

软件的使用，为今后的学习打下基础。
在实际的 FPGA 设计中，在每个阶段都会遇到很多问题。为了实现最终的目标，需要充分地利用 Quartus II 设计软件。这需要一个比较长时间的学习和积累，希望通过这个实例能让初学者更快地入门。

2.6 典型实例 2：在 Xilinx 的 FPGA 开发板上运行第一个 FPGA 程序

2.6.1 实例的内容及目标

1．实例的主要内容

本节旨在通过给定的工程实例——“按键开关控制 LED”来熟悉 Xilinx ISE 软件的基本操作、设计、编译及仿真流程。同时使用基于 Xilinx FPGA 的开发板将该实例进行下载、验证及调试，完成工程设计的硬件实现，熟悉 Xilinx FPGA 开发板的使用及配置方式。
在本训练中设计软件采用 ISE 7.1i，实现功能是利用 4 个按键开关来控制 8 个 LED 灯。具体的显示方案是由 4 个按键开关控制 8 个 LED 灯，根据按键开关按下的不同，会有不同的灯点亮。
在本训练中，将主要讲解下面一些知识点。

- Xilinx ISE 工程创建及属性设置。
- Xilinx ISE 源文件设计输入方式。
- Xilinx ISE 综合。
- Xilinx ISE 行为仿真。
- Xilinx ISE 约束设计。
- Xilinx ISE 布局布线。
- Xilinx ISE 时序仿真。
- Xilinx ISE 硬件下载。

通过这些知识点，按照下面提供的训练流程，读者可以迅速地掌握使用 Xilinx ISE 软件进行 FPGA 开发的方法。

2．实例目标

通过详细的流程讲解，读者应达到下面的目标。

- 熟悉 Xilinx ISE 软件的操作环境。
- 熟悉 Xilinx ISE 软件开发 FPGA 的基本流程。
- 可独立使用 Xilinx ISE 软件开发新工程。

2.6.2 平台简介

本实例基于红色飓风 II 代 Xilinx 板，此开发板集成了 Xilinx 的 Spartan3 40 万门 FPGA 及相关的丰富外设资源。
此开发板更加详细的信息、扩展附件及使用方法，可以到红色飓风的官方网站：http://www.fpgadev.com 获取更多的信息。
本实例将通过一个简单的实例在这个开发板上对 Xilinx 的 FPGA 设计流程做一个全面的介绍，依照此例程

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

的流程便可在该开发板上运行一个 FPGA 程序。
如图 2.65 所示为此开发板的一个外观图。

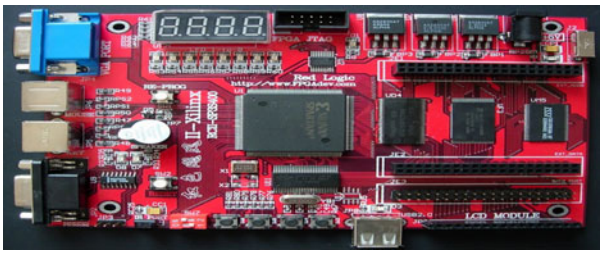


图 2.65　红色飓风 II 代 Xilinx 板外观图

2.6.3　实例详解

本节同样将使用图解的方式将整个流程一步一步展现给读者，使读者能够轻松掌握 ISE 的开发流程。

1．工程创建及属性设置

（1）启动 Project Navigator。
安装好 ISE7.1 套件后，可以按照以下的方法打开 Project Navigator。
① 在桌面上面双击 Xilinx ISE 7.1i 的快捷方式图标，如图 2.66 左图所示。
② 选择“开始”→“所有程序”→“Xilinx ISE 7.1i”→“Project Navigator”，打开 Project Navigator，如图 2.66 右图所示。
打开后的 Project Navigator 界面如图 2.67 所示。可以看到，Project Navigator 的界面是一个标准的 Windows 软件视窗。分别包括标题栏、菜单栏、工具栏、工程浏览器、进度浏览器、主工作区、信息栏、状态栏。

图 2.66　ISE 的启动

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

图 2.67　Project Navigator 主界面

（2）创建一个新的工程。

单击“File”→“New Project…”，弹出如图 2.68 所示对话框。

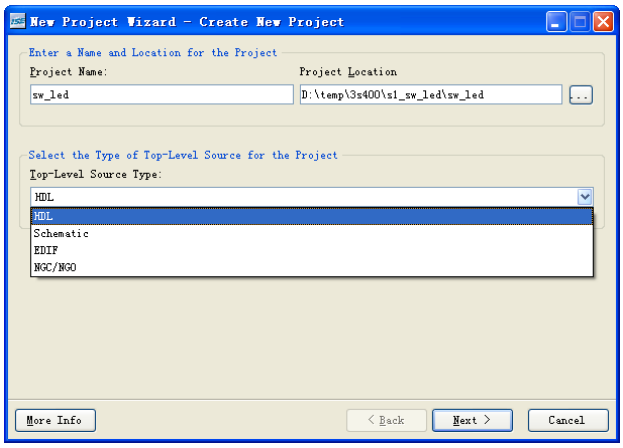


图 2.68　新建工程对话框

在这里填写将要创建的工程的名称（Project Name）、路径（Project Location）和工程的顶层模块类型（Top-Level Module Type）。

顶层模块类型我们主要使用前面两种：HDL(Hardware Design Language)硬件设计语言模式和 Schematic 原理图模式，这里选择 HDL。

（3）设置工程属性。

填写好后，单击“Next”按钮，在如图 2.69 所示的对话框中设置工程的属性。

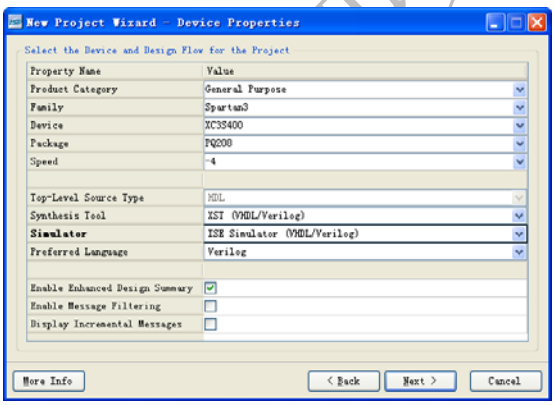


图 2.69　工程属性对话框

在这个对话框可设置的属性定义，如表 2.4 所示。

表 2.4　　属性定义

Device Family	使用的 FPGA 的种类
Device	使用的 FPGA 的型号
Package	使用的 FPGA 的封装
Speed Grade	使用的 FPGA 的速度级别
Top-Level Module Type	顶层模块类型
Synthesis Tool	综合工具
Simulator	仿真工具
Generated Simulation Language	仿真模块语言类型

本实例选用的都是 ISE 自己带的综合工具和仿真工具，这里也可以选择第三方的应用软件，如图 2.70 和图 2.71 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

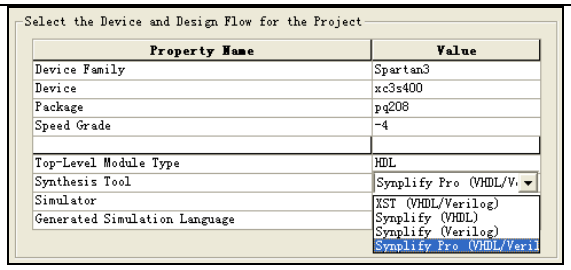


图 2.70　工程属性对话框选择第三方综合工具

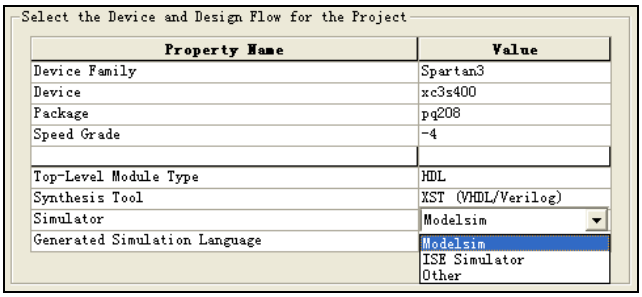


图 2.71　工程属性对话框选择第三方仿真工具

ModelSim 和 Synplify Pro 是比较通用的第三方仿真和综合软件。在这里如果我们选择了使用第三方的软件进行综合及仿真的话，在后面执行相应步骤的时候 ISE Navigator 就会自动寻找并打开相应的软件。

（4）添加设计文件。

填写好 FPGA 型号和使用的综合，仿真软件后，单击“Next”按钮打开如图 2.72 所示的创建源文件对话框。

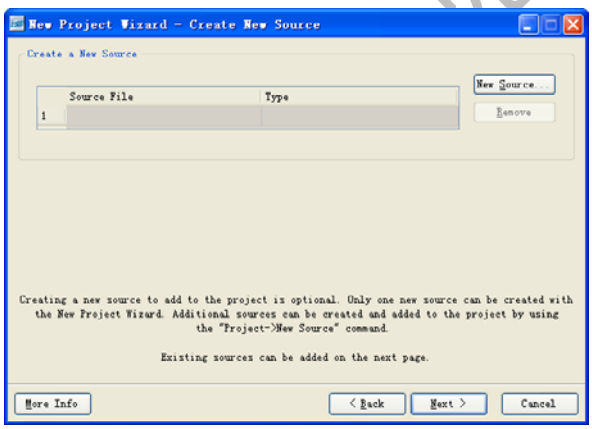


图 2.72　创建源文件对话框

在这个对话框里面，允许用户为即将建立的工程创建一个新的源文件，并且这里只可以创建一个源文件。其他的要在工程建立以后创建，也可以在创建工程以后再建立所有的源文件。直接单击“Next”按钮，进入如图 2.73 所示的添加源文件对话框。

在这个对话框里面为即将建立的工程添加已经存在的源文件。如果没有现成的源文件，直接单击“Next”按钮，完成新工程的创建。

（5）完成工程创建。

在如图 2.74 所示的新工程信息对话框中，列出了新建工程的相关参数及属性。

在这个对话框里面显示将要创建的工程的全部信息，确认无误后单击“Finish”按钮，Project Navigator 将自动创建一个名为 sw_led 的工程。如图 2.75 所示是新工程的界面。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

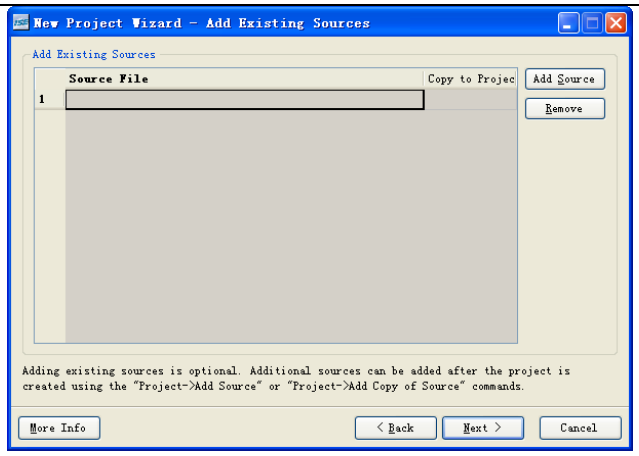


图 2.73　添加源文件对话框

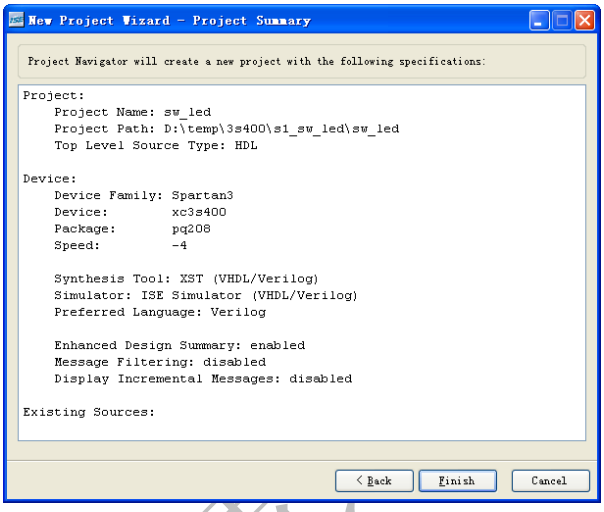


图 2.74　工程信息对话框

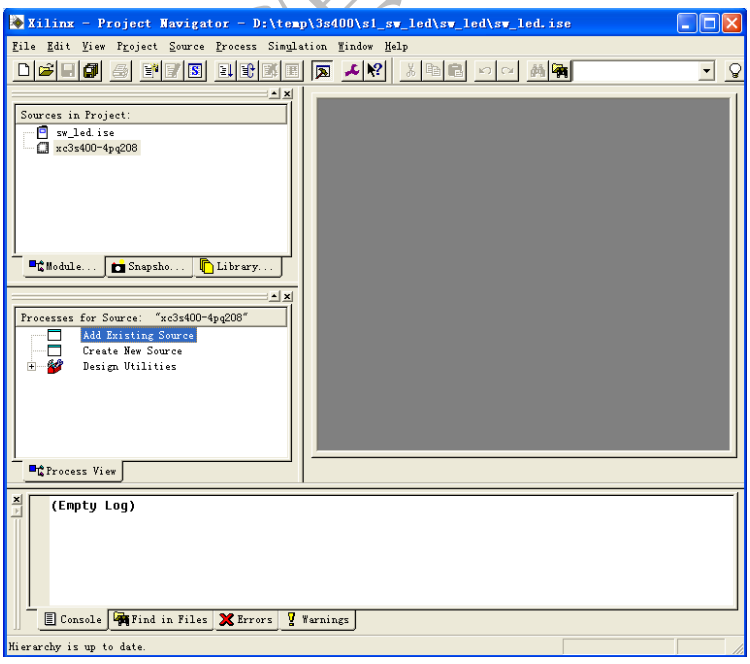


图 2.75　工程创建后的界面

2．设计输入

创建好工程以后就要为工程添加源文件了，具体方法如下。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

为工程添加源文件有两种方式，可以双击“Process View”对话框里面的“Create New Source”，也可以在“Module View”对话框里面的“xc3s400-4pq208”图标上面单击鼠标右键，选择“New Source…”选项，如图 2.76 所示。

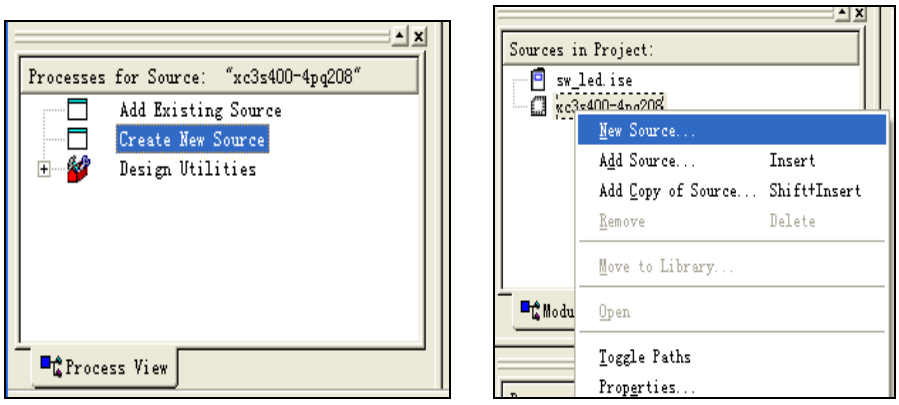


图 2.76　添加源文件的两种方式

选择“New Source”，弹出如图 2.77 所示的新建源文件对话框。

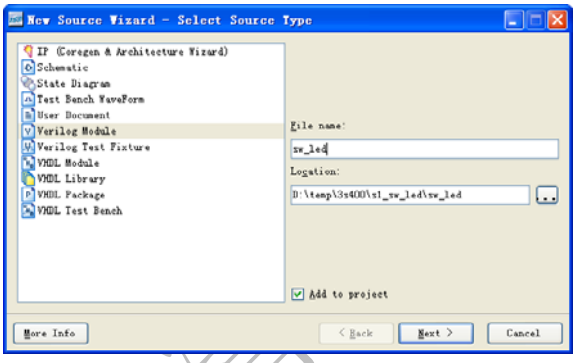


图 2.77　新建资源对话框

在右面的“File Name”栏里面填写要生成的源文件的名字，在“Location”栏填写源文件保存的路径，一般位于工程文件夹里面，没有特殊需要不必更改。另外一定要选择“Add to project”选项，然后在左边的一排图标里面选择源文件的类型后单击“Next”按钮，进入如图 2.78 所示的 Verilog 源定义对话框。

可以在上面的对话框里面输入源文件的模块名称和管脚定义，也可以先不输入，后面写程序的时候自己输入。单击“Next”按钮，完成源文件的创建，在如图 2.79 所示的对话框中列出了新建源文件的一些信息。确认信息无误后，单击“Finish”按钮，ISE 将生成名为 sw_led.v 的源文件，如图 2.80 所示。

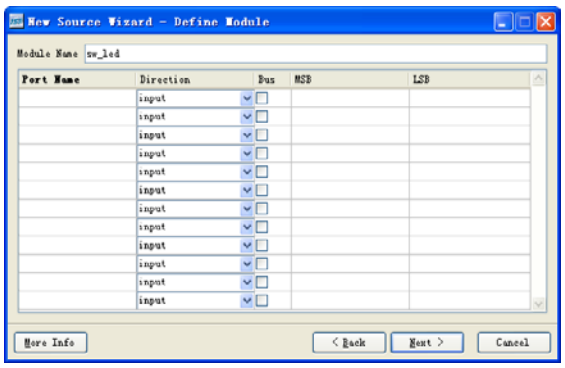


图 2.78　新建 Verilog 文件设置对话框

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

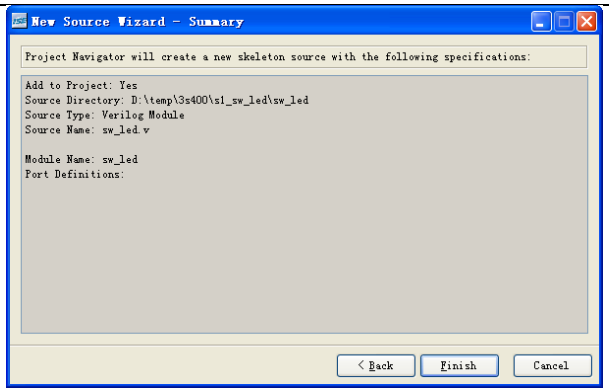


图 2.79　新建源文件信息对话框

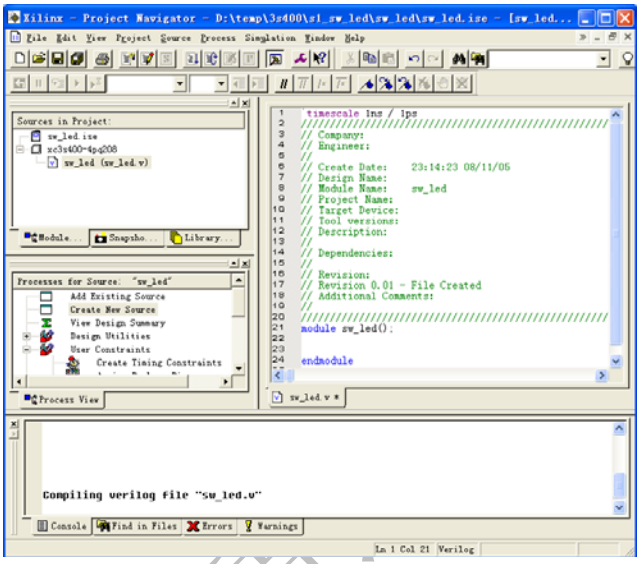


图 2.80　添加新资源后工程界面

用户可以在工作区中开始进行设计的输入，也可参考实例代码中的例程，将代码直接复制到新建的 Verilog 文件下。输入好程序以后，保存源文件，完成设计的输入。

3．综合及仿真

（1）综合。

在“Process View”对话框里面双击“Synthesize – XST”，开始进行工程的综合，如图 2.81 所示。

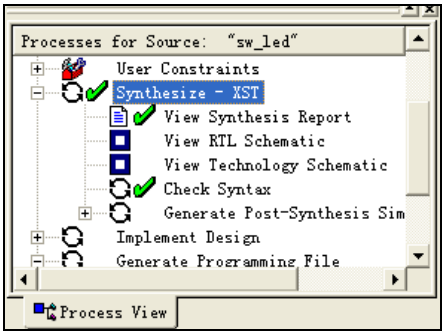


图 2.81　用 ISE 综合工具进行综合

综合主要检查源文件程序里面的语法错误（Check Syntax），双击“View Synthesis Report”可以观察综合的结果报告，如图 2.82 所示。

开发板免费领取试用　开发板免费领取试用

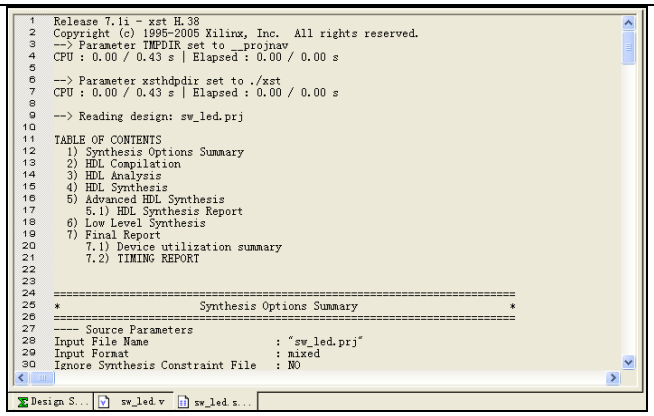


图 2.82　综合报告

如果在这步发现源程序的设计语言有语法毛病，就会弹出 Error 警告。这样就可以根据报错的位置，在源程序里面查找错误位置，修改好以后重新进行综合，直至无错误为止。

（2）综合后仿真。

这里的仿真是综合后的仿真，也可以称为功能仿真。仿真时并不考虑工程的约束条件及器件的资源使用状况，因此不会产生仿真延迟。这里使用的是 ISE 自带的仿真工具进行综合后仿真。下面是仿真的步骤。

① 产生仿真文件。

如图 2.76 的方式，在“New Source”对话框里面选择创建 Test Bench Wavefrom 文件，如图 2.83 所示。单击“Next”按钮，在如图 2.84 所示对话框中为 Test Bench 指定源文件。如图所示在源文件对话框中只有一个文件，这是因为我们只为工程添加了一个设计文件。如果我们为工程添加了几个设计文件，在这个对话框中，将出现多个源文件。此时用户可以选择需要进行仿真的源文件即可。

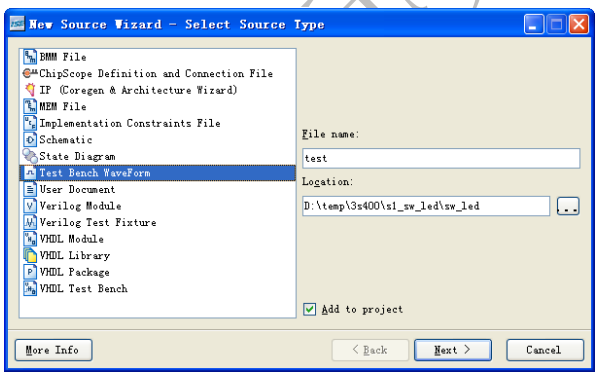


图 2.83　新建 Test Bench 对话框

在本训练中，将要进行仿真的源文件是 sw_led，如图 2.84 所示。单击“Next”按钮完成仿真文件的产生，如图 2.85 所示。

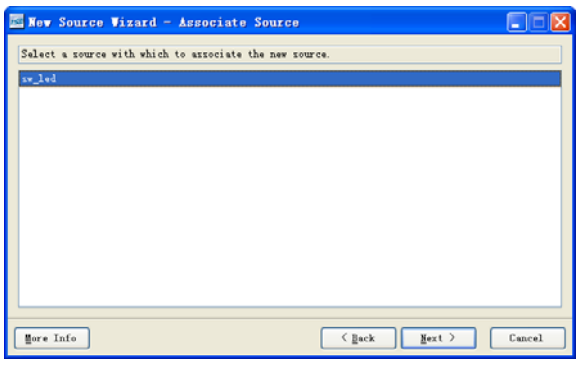


图 2.84　为 Test Bench 指定源文件

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

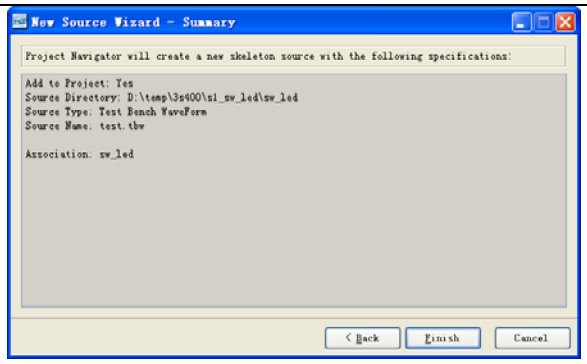


图 2.85　Test Bench 信息对话框

单击“完成”按钮，ISE 将创建名为 test.tbw 的仿真文件，同时弹出如图 2.86 所示的对话框。在该对话框中，用户可以对仿真的时间参数进行设置。

在这个对话框中，包括时钟信息、时钟时间信息、全局信号、仿真时间等多个参数都可以进行设置。按照仿真需求修改好时间参数以后，单击“OK”按钮就能在 ISE 的工作区里面看见新建的波形文件了，如图 2.87 所示。

在波形文件里面，clk 是时钟信号，在前面的时间参数设计里面已经设置好了。用户若需要对时钟进行更改，可选择执行“Test Bench”→“Rescale Timing…”命令，弹出修改时间参数的对话框，如图 2.88 所示。

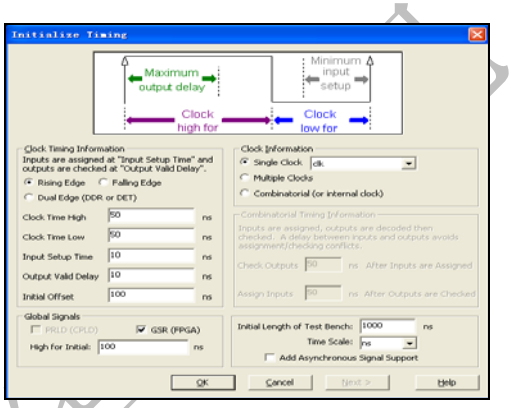


图 2.86　仿真时间参数设置对话框

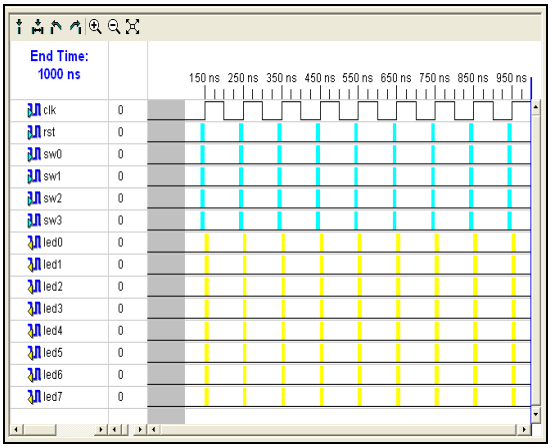


图 2.87　时钟波形

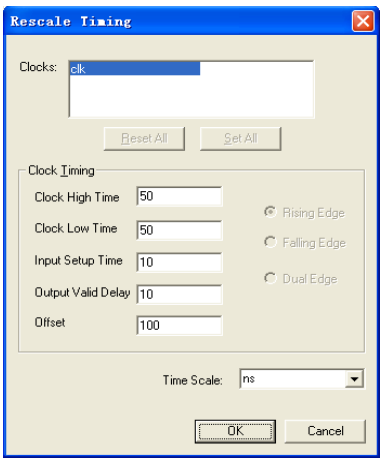


图 2.88　修改时间参数对话框

同样也可以执行“Test Bench”→“Set End of Test Bench…”，在弹出的对话框里面修改仿真波形的截止时间，如图 2.89 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

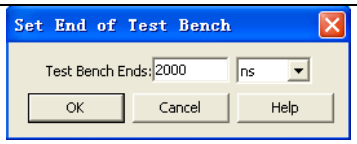


图 2.89　截止时间设置对话框

单击“OK”按钮就会发现仿真波形的时间长度由 1000ns 变成了 2000ns。

② 添加仿真激励。

要进行仿真，还需要为仿真添加激励。这里可以通过点击波形图中的蓝色方块来设置输入波形电平的高低，如图 2.90 所示。

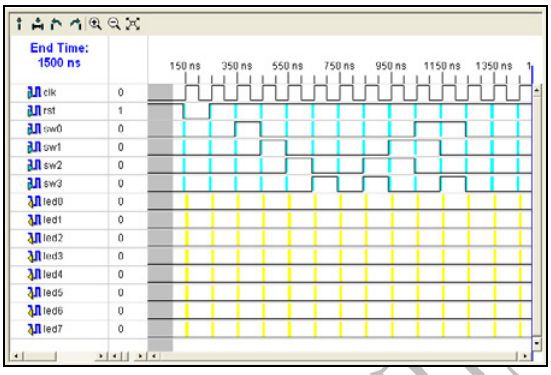


图 2.90　设置输入电平

设置好以后一定不要忘记保存波形文件。保存以后就会在工程浏览器的当前工程的子目录下看见刚生成测试的文件，如图 2.91 所示。

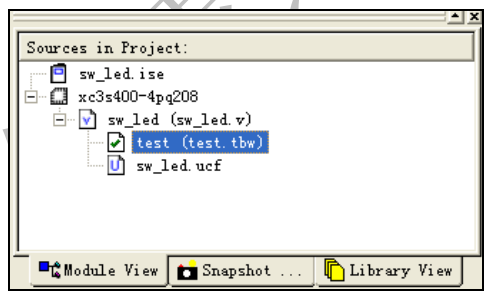


图 2.91　保存仿真文件到工程

③ 仿真。

首先在工程浏览器的【Module View】里面选择“test（test.tbw）”。

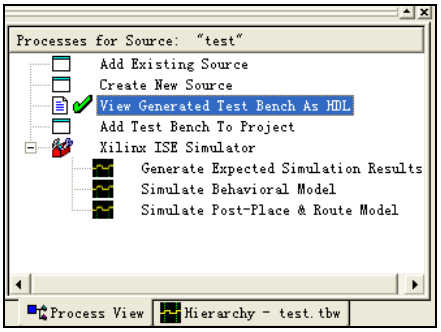


图 2.92　由波形文件生成测试代码

然后在进度浏览器的“Process View”对话框里面双击“View Generated Test Bench As HDL”图标，就能看见 ISE 根据刚才设置的波形文件自动生成的测试文件，如图 2.93 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

```
////////////////////////////////////////////////////////////////////////////////
// Copyright (c) 1995-2003 Xilinx, Inc.
// All Right Reserved.
////////////////////////////////////////////////////////////////////////////////
//   ____  ____
//  /   /\/   /
// /___/  \  /    Vendor: Xilinx
// \   \   \/     Version : 7.1i
//  \   \         Application : ISE Foundation
//  /   /         Filename : test.tfw
// /___/   /\     Timestamp : Mon Oct 02 09:52:07 2006
// \   \  /  \
//  \___\/\___\
//
//Command:
//Design Name: test
//Device: Xilinx
//
 timescale 1ns/1ps

module test;
    reg clk = 1'b0;
    reg rst = 1'b1;
    reg sw0 = 1'b0;
    reg sw1 = 1'b0;
    reg sw2 = 1'b0;
    reg sw3 = 1'b0;
    wire led0;
    wire led1;
    wire led2;
```

图 2.93　生成的测试代码

在进度浏览器中双击【Generate Expected Simulation Result】，弹出两个对话框，都选择是（Y），将看见生成的预期输出波形，如图 2.94 所示。

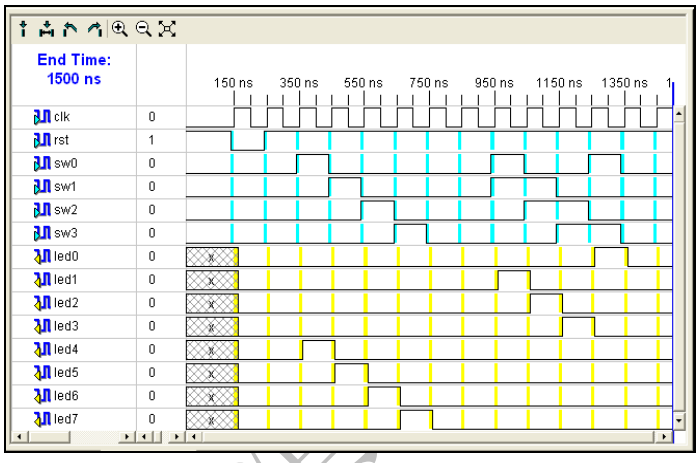


图 2.94　预期仿真波形

在进度浏览器中双击“Add Test Bench to Project”，将测试波形添加到工程里面。将会看到工程浏览器中 test 文件的图标由 test (test.tbw) 变成 test (test_tb.v) 。

如图 2.95 所示，在进度浏览器中双击“Simulate Behavioral Model”将对工程进行行为仿真，仿真结果如图 2.96 所示。

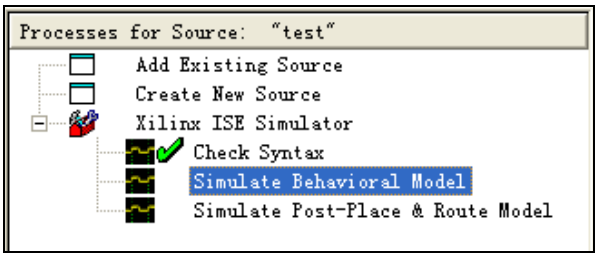


图 2.95　运行功能仿真

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

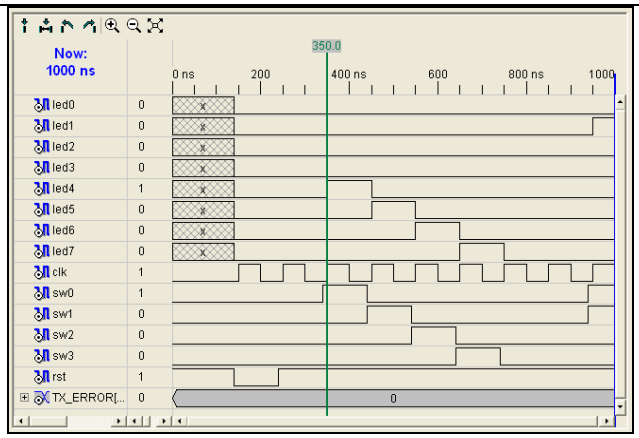


图 2.96　功能仿真结果

从图中可以看出功能仿真结果符合设计要求，可以继续进行下面的设计。如果此处仿真发现设计功能不符合要求，则要对原代码进行修改，直到仿真结果符合设计要求为止。

4．约束设计

在源程序输入完成以后，就可以设置约束，规划布局布线了。

约束有很多种：时序约束（Timing Constrains）、管脚约束（Assign Package Pins）以及面积约束（Area Constraints）等。

如图 2.97 所示是 ISE 进度浏览器中包含的用户约束的选项。

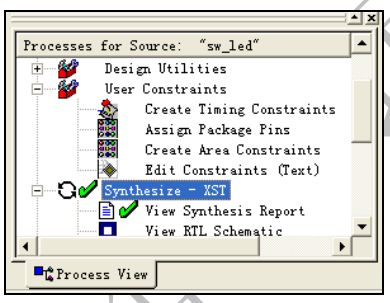


图 2.97　设置约束

（1）时序约束。

时需约束主要约束设计的时序和时钟频率，双击图 2.97 中的“Create Timing Constraints”图标，ISE 将打开约束编辑器，如图 2.98 所示。

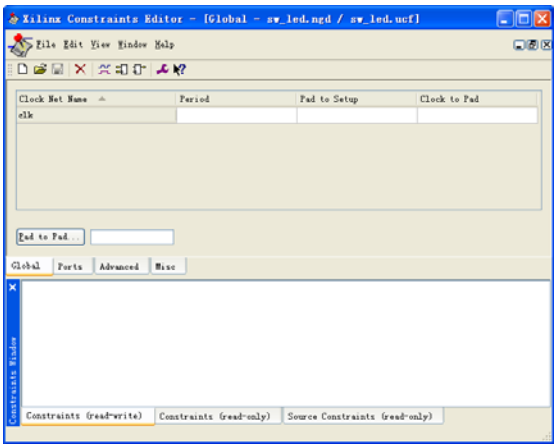


图 2.98　时序约束界面

在约束编辑器里面有 4 个复选页，定义如表 2.5 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

表 2.5　　属性定义

属　性	定　义
Global	用于附加全局约束，包括周期约束，输入延迟约束和输出延迟约束
Ports	附加端口约束，可以对每一个端口进行单独设置，包括物理位置、输入延迟和输出延迟。同时，也可以设置分组，进行分组约束
Advanced	附加分组约束（TNM/TNM_NET/TIMEGRP）和时序约束（FROM_TO/TIG O/OFFSET）等
Misc	附加专用约束，包括电压、初始值等

注意　附加约束的原则是先加全局约束后加分组约束。

在本实例中只做简单的时钟约束。在 clk 的 period 中输入：20ns HIGH 50%，即设置时钟的周期和占空比。设置后保存，完成时序约束。

（2）管脚约束。

管脚约束即约束工程设计源文件与选定器件对应的输入/输出管脚属性。双击图 2.97 中的“Assign Package Pins”图标，ISE 将打开 PACE 工具，如图 2.99 所示。

通过图 2.99 的“Design Object List”对话框里面的 Loc 列，可以为设计添加输入/输出管脚，添加形式为“Pxxx”或者“pxxx”。其中 P/p 代表 Pin，xxx 是数值。

（3）面积约束。

面积约束目的在于规划 FPGA 里面的逻辑使用大小。双击图 2.97 中的“Create Area Constraints”图标，ISE 同样打开 PACE 工具，面积约束和管脚约束用的是一个界面。在面积约束里面我们可以对设计使用的资源面积加以规划，控制使用资源在 FPGA 里面的位置。

这里只使用到管脚约束，按照按键和 LED 灯与 FPGA 的连接，输入管脚分配，然后保存分配。在保存选项中，选择“XST Optional”，如图 2.100 所示，关闭对话框，完成约束设计。

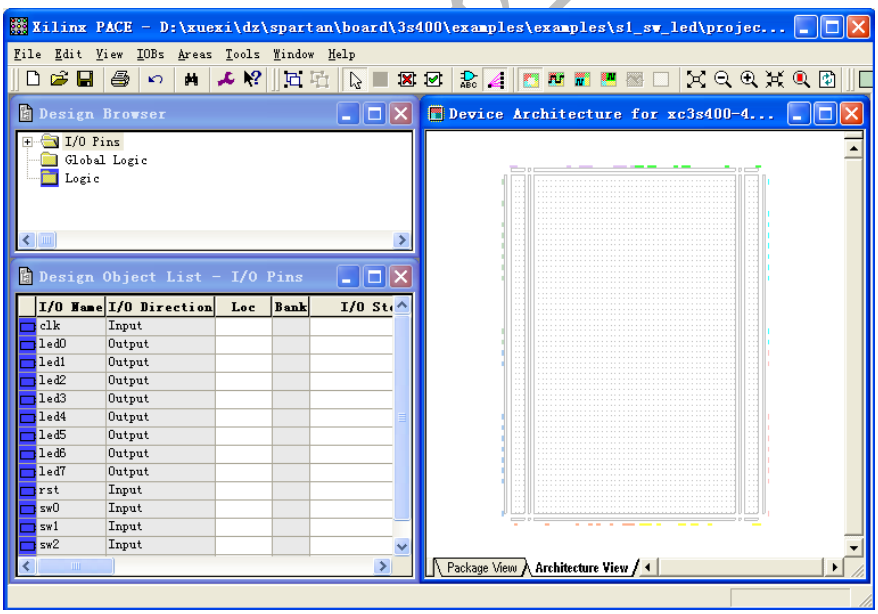


图 2.99　管脚约束工具 PACE

开发板免费领取试用　开发板免费领取试用

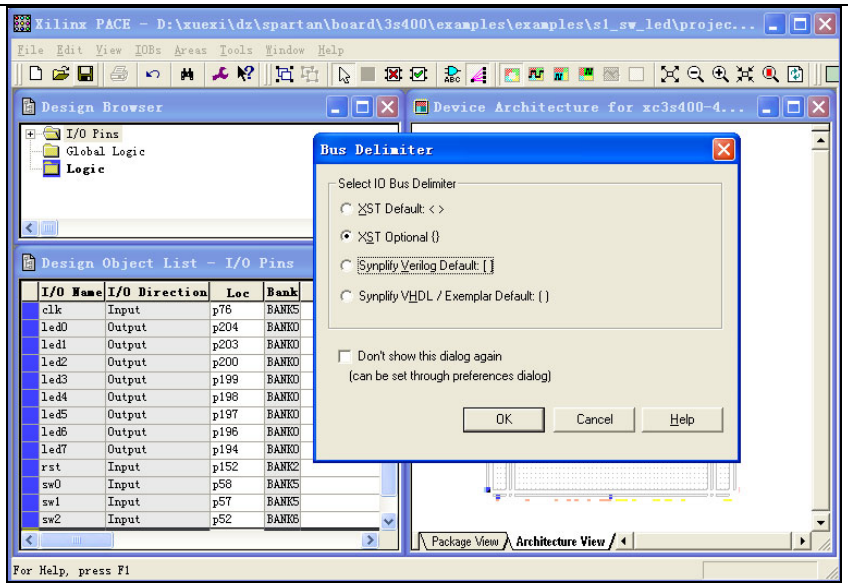


图 2.100　保存管脚约束结果

5．布局布线

设置好了各种约束以后就可以进行布局和布线了。双击进度浏览器中的“Implement Design”，开始进行布局布线实现。与综合相同，如果在“Implement Design”的右边出现绿色的对号，说明已经完成布局布线，没有发现错误，如图 2.101 所示。

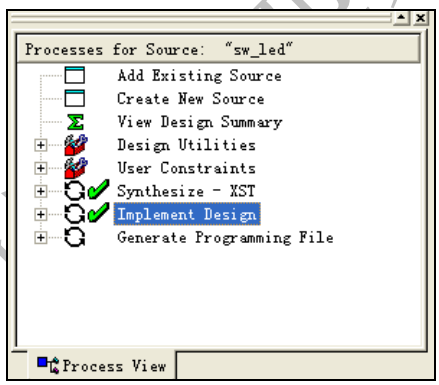


图 2.101　运行布局布线

如果实现过程中出现错误，在信息栏中根据错误的提示，修改相关的约束设计，再进行布局布线的实现，直至没有错误为止。

6．布局布线后仿真

在工程浏览器中选择测试文件，双击进度浏览器中“Implement Design”下面的“Simulate Post-Place & Route Model”，对工程进行布局布线后仿真，仿真结果如图 2.102 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

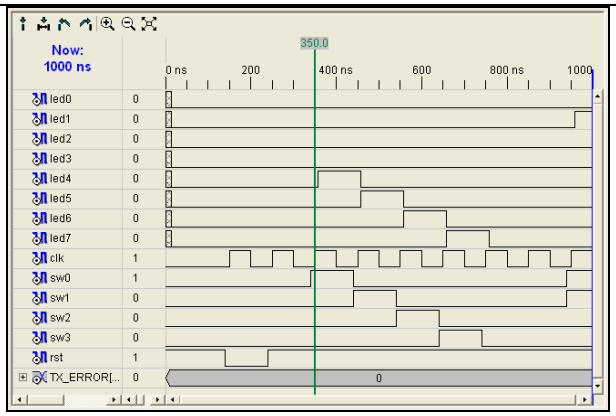


图 2.102　时序仿真结果

通过与行为仿真结果图 2.96 对比，可以发现，布局布线后仿真加入了布线的延迟，输出的结果相对于时钟有相应的延迟。在工作区里面单击鼠标右键，在弹出的菜单里面选择“Add Measure”选项，可以通过放大工作区和拖动时间测量游标，读取时间延迟，如图 2.103 所示。

由上图可以看出，尽管输出相对于时钟存在一定的延时，但输出结果是正确的，符合设计要求，可以进行下载和在线调试了。

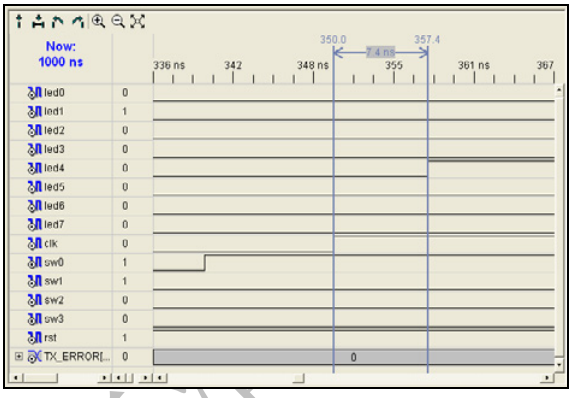


图 2.103　测量时间延迟

7．下载程序

布局布线完成以后就可以将程序下载到开发板上进行调试。开发板提供两种下载方式：JTAG 模式和 PROM 模式。区别在于 JTAG 模式是用于在线调试使用的，断电后不保留程序；而 PROM 模式则是将程序烧写在存储器里面，上电后自动加载。

（1）JTAG 模式下载。

如图 2.104 所示，在进度浏览器中“Generate Programming File”下面可以看到有 3 个选项，分别是编程文件生成报告、生成编程文件和使用 iMPACT 配置器件。

在“Generate Programming File”图标上面单击右键，选择“Properties…”，弹出如图 2.105 所示的属性对话框，在其中可以设置下载文件的属性。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

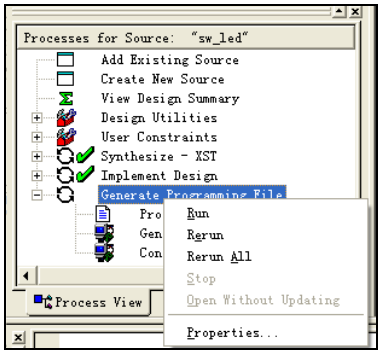


图 2.104 生成下载文件

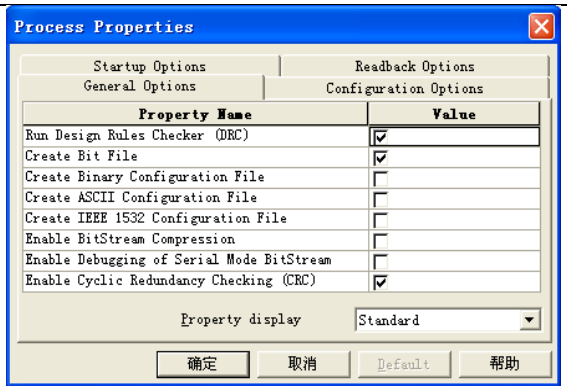


图 2.105 “General Options”选项卡

在该对话框中有 4 个选项卡，可以对相应的下载参数进行设置。

“General Options”选项卡里面的“Create Bit File”选项生成用于 JTAG 模式下载的二进制下载文件。确定选择该项，如图 2.105 所示。

在“Startup Options”选项卡里面将“FPGA Start-Up Clock”后面的内容改为“JTAG Clock”，如图 2.106 所示，在这里要根据不同的下载模式正确地选择时钟。

在“Readback Options”选项卡里面将“Security”后面的内容置为“Enable Readback an Reconfiguration”，如图 2.107 所示，然后单击“OK”按钮，完成下载的属性设置。

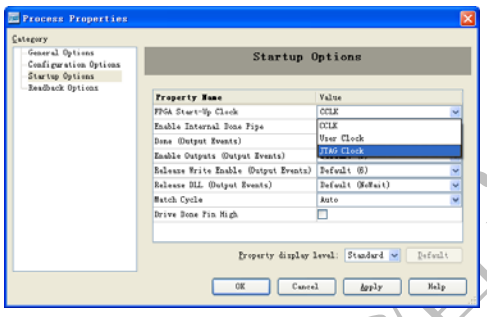


图 2.106 “Startup Options”选项卡

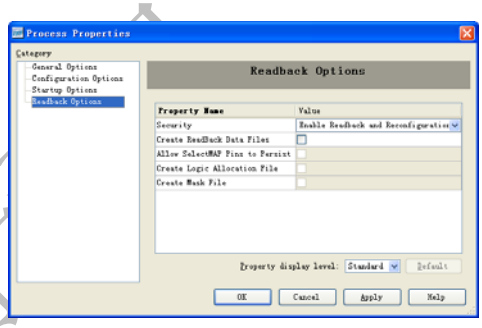


图 2.107 “Readback Options”选项卡

如图 2.108 所示，双击进度浏览器里面的“Generate Programming File”图标，ISE 将生成供 JTAG 模式下载的 bit 文件。“Generate Programming File”左边的绿色对号说明已经成功生成了下载 文件。

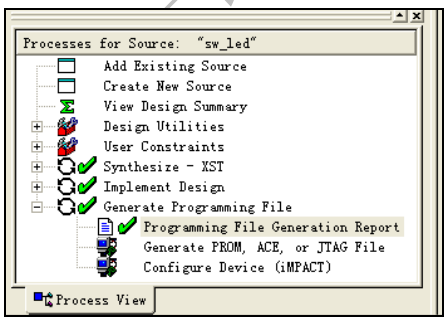


图 2.108 生成下载文件

双击“Generate Programming File”下面的“Configure Device（iMPACT）”图标，打开如图 2.109 所示的 iMPACT 工具。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

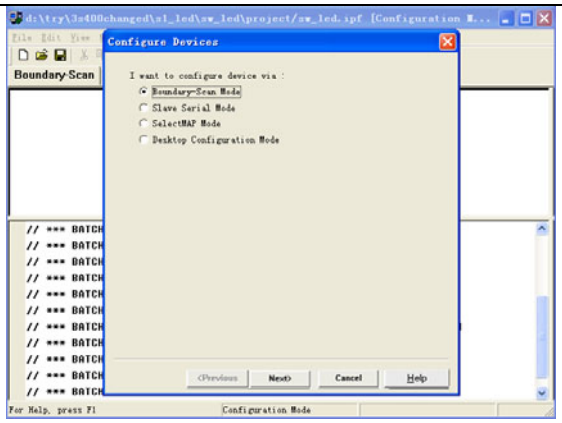


图 2.109　iMPACT 工具

在如图 2.109 所示的对话框里面选择器件配置的模式，这里选择边界扫描模式“Boundary-Scan Mode”，单击“下一步”按钮，如图 2.110 所示。

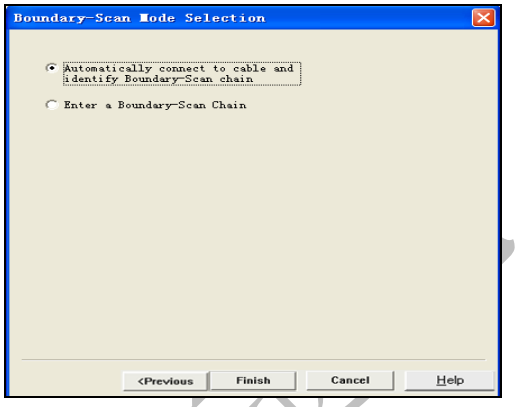


图 2.110　边界扫描模式选择对话框

选择自动加载边界扫描链路，单击完成。

由于开发板上面的 JTAG 和 PROM 下载是在一条链路上的，所以会弹出如图 2.111 所示的对话框，单击“确定”按钮就能看见 ISE 扫描到的开发板上面的下载链，如图 2.112 所示。

图 2.111　边界扫描检测结果对话框

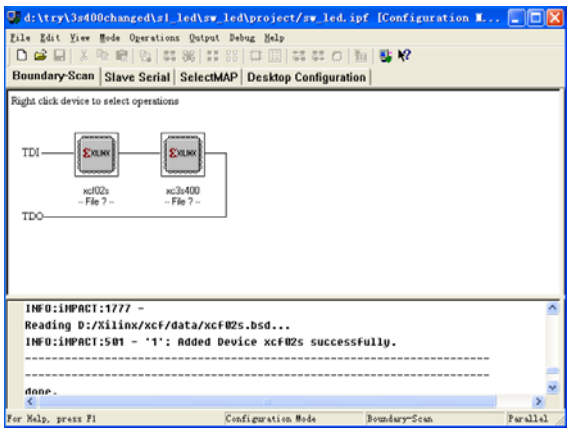


图 2.112　ISE 扫描下载链结果

其中下面标有 xcf02s 的图标代表的是 PROM，下面标有 xc3s400 的图标代表 FPGA。

双击 FPGA 图标，在弹出的对话框里面选择刚才生成的二进制下载文件（后缀为.bit）。配置好以后右键单

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

击 FPGA 图标，在弹出的菜单里面选择“Program…”选项，如图 2.113 所示。此时将会出现如图 2.114 所示的下载属性设置对话框。

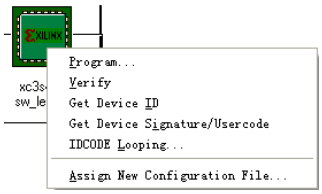


图 2.113　设置配置文件

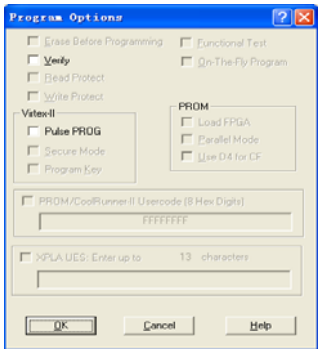


图 2.114　配置设置对话框

单击“OK”按钮就可以开始下载了。若下载成功将会有如图 2.115 所示的下载成功提示。

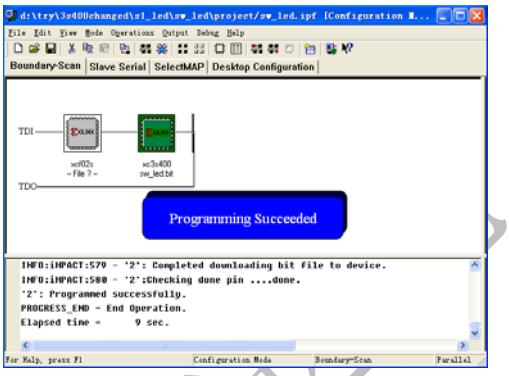


图 2.115　JTAG 下载成功

（2）PROM 模式下载。

PROM 模式下载是把程序存在一块存储器里面（开发板上面使用的是 Xilinx 公司的 xcf02s）。当开发板上电的时候，FPGA 就自动加载存储器里面的程序。下面是 PROM 模式下载的具体操作方法。

要进行 PROM 模式下载，首先要生成相应的二进制下载文件，双击进程浏览器中“Generate Programming File”下面的“Generate PROM, ACE, or JTAG File”图标，弹出如图 2.116 的对话框。

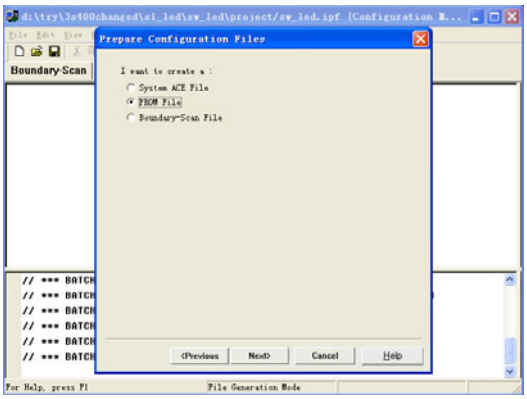


图 2.116　生成 PROM 文件对话框

选择“PROM File”选项，单击“下一步”按钮，打开如图 2.117 所示的 PROM 文件配置对话框。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

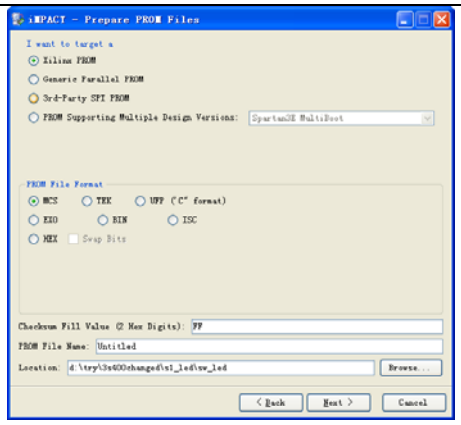


图 2.117　PROM 文件设置对话框

在该对话框中，选择生成一个 Xilinx PROM，后缀选择 MCS，文件格式选择 FF（十六进制），在“PROM File Name”后面填写生成的 PROM 下载文件的名称，在“Location”后面填写 PROM 文件保存地址。配置后单击“Next”按钮，出现 PROM 器件配置对话框，如图 2.118 所示。

选择 PROM 的型号，要与开发板上面使用的对应。这里选择 xcf02s，单击“Next”按钮。

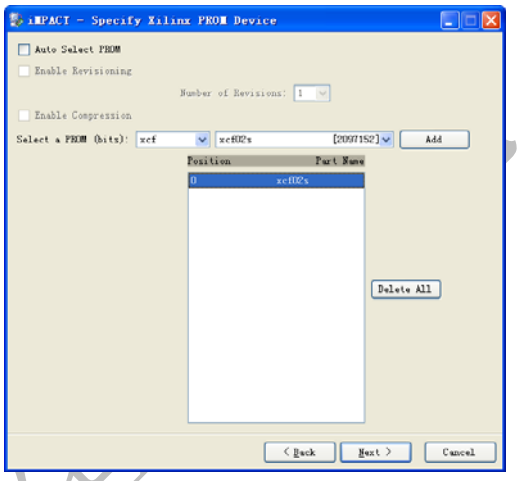


图 2.118　选择 PROM 型号

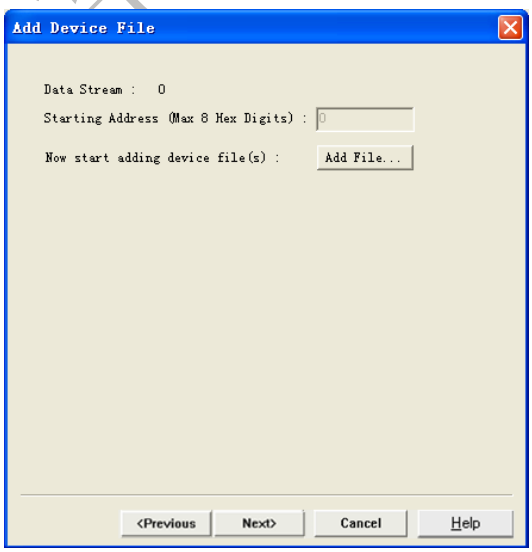


图 2.119　为生成 PROM 文件添加二进制文件

在出现的如图 2.119 所示对话框中单击“Add File…”按钮，在弹出的对话框里面选择 JTAG 下载的时候使用的二进制文件（后缀为.bit），ISE 将通过这个二进制文件生成 PROM 文件。

开发板免费领取试用　开发板免费领取试用

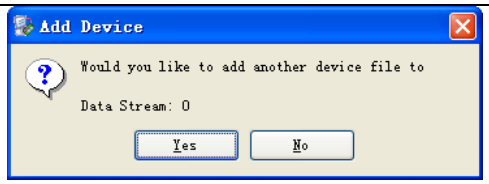


图 2.120　是否还要加载一个设计文件对话框

选择完一个二进制文件后，会弹出如图 2.120 所示的对话框，询问是否还要加载一个设计文件。单击“NO”按钮，完成 PROM 模式下载的配置。

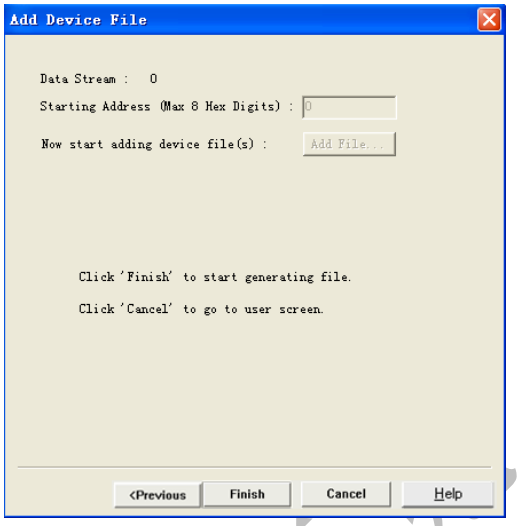


图 2.121　生成 PROM 文件

在出现的图 2.121 中单击“Finish”按钮，ISE 就会自动为我们生成后缀为.MCS 的十六进制 PROM 下载文件。生成成功后，将出现如图 2.122 所示的提示。

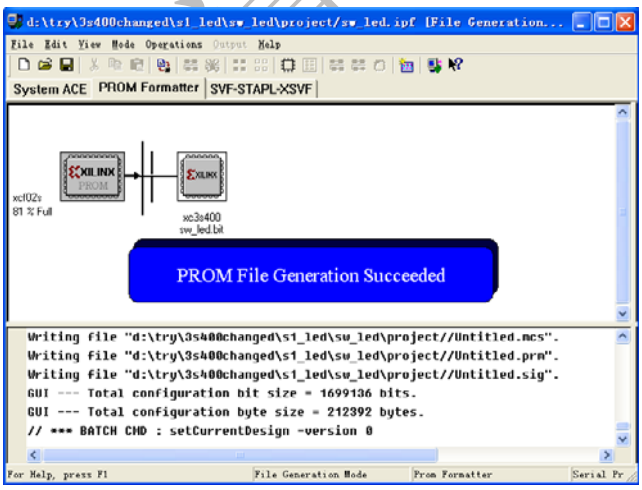


图 2.122　PROM 文件生成成功提示

生成了 PROM 的下载文件以后，再按照 JTAG 下载的方法，扫描下载链。将生成的.MCS 文件加载到 PROM 图标中去，然后再在 PROM 图标上面点击右键，选择“Program…”选项，如图 2.123 所示。

如图 2.124，在出现的下载属性配置对话框中选择相应的选项后，单击“OK”按钮就开始 PROM 的下载了（PROM 下载相对比较慢，并且最好先擦除 PROM 里面的内容再下载）。成功下载会出现如图 2.125 所示的提示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

图 2.123　下载 PROM 器件

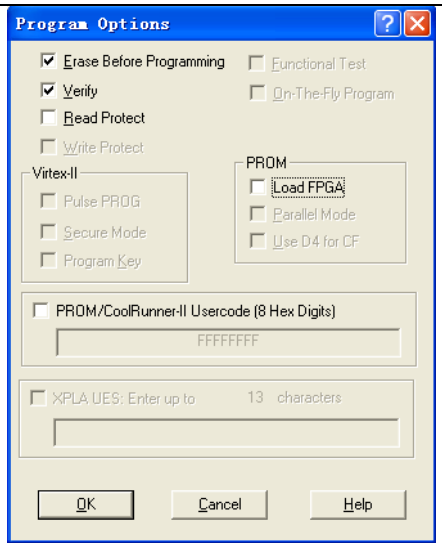


图 2.124　下载 PROM 器件设置对话框

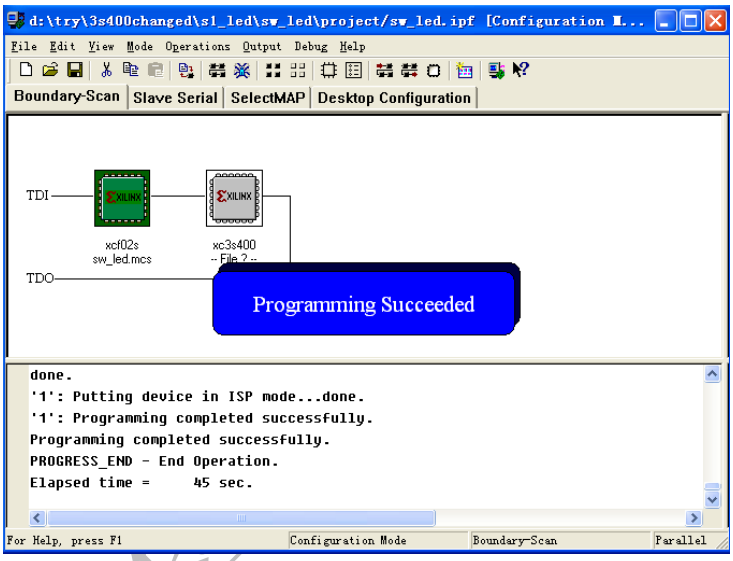


图 2.125　完成对 PROM 器件的烧写

需要注意的是，如果选用 JTAG 下载模式，则下载成功后，FPGA 就可实现预想的功能。如果选用 PROM 下载模式，下载成功后，要重新上电后 FPGA 才能正常工作。

因为 PROM 直接方式是将文件下载到 PROM 器件中，并没有直接下载到 FPGA 里，要重新上电后由 FPGA 自动从 PROM 器件中加载配置文件，才能实现预想的功能。

当然用户也可以在如图 2.124 所示的属性配置中选择“Load FPGA”选项。这样 PROM 下载完成后，下载的逻辑将自动加载至 FPGA 内，而无需重新上电。

2.6.4　小结

上述步骤就是利用 ISE 进行的一个完整的 FPGA 设计流程。虽然例程实现的功能比较简单，但对于初学者来说，是一个不错的入门实例。此实例的主要目的是让初学者对 FPGA 的设计有一个初步的了解，并熟悉 ISE 软件的使用，为今后的学习打下基础。

联系方式

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

集团官网：www.hqyj.com 嵌入式学院：www.embedu.org 移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn 物联网学院：www.topsight.cn 研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路银海大厦 A 座 8 层，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218

华清远见

开发板免费领取试用 开发板免费领取试用

华清远见 FARSIGHT 始于2004 | 嵌入式培训专家

10年口碑积累，成功培养50000多名研发工程师，铸就专业品牌形象
华清远见的企业理念是不仅要做良心教育、做专业教育，更要做受人尊敬的职业教育。

《FPGA 应用开发入门与典型实例》（修订版）

作者：华清远见

专业始于专注 卓识源于远见

第 3 章　硬件描述语言 Verilog HDL 基础

本章目标

- 初步掌握 Verilog HDL 语言的基本语句
- 学会用 Verilog HDL 语言设计简单的组合逻辑和时序逻辑电路
- 通过典型实例，熟悉 FPGA 设计的完整流程

专业始于专注 卓识源于远见
开发板免费领取试用 开发板免费领取试用

3.1 Verilog HDL 语言简介

3.1.1 Verilog HDL 的历史和进展

1．什么是 Verilog HDL

Verilog HDL 是硬件描述语言的一种，用于数字电子系统设计。它允许设计者用它来进行各种级别的逻辑设计，可以用它进行数字逻辑系统的仿真验证、时序分析、逻辑综合。它是目前应用最广泛的一种硬件描述语言之一。

2．Verilog HDL 的历史

Verilog HDL 是在 1983 年由 GDA（GateWay Design Automation）公司的 Phil Moorby 首创的。Phil Moorby 后来成为 Verilog-XL 的主要设计者和 Cadence 公司（Cadence Design System）的第一个合伙人。
在 1984 年～1985 年，Moorby 设计出了第一个关于 Verilog-XL 的仿真器，1986 年，他对 Verilog HDL 的发展又做出了另一个巨大贡献：即提出了用于快速门级仿真的 XL 算法。
随着 Verilog-XL 算法的成功，Verilog HDL 语言得到迅速发展。1989 年，Cadence 公司收购了 GDA 公司，Verilog HDL 语言成为 Cadence 公司的私有财产。1990 年，Cadence 公司决定公开 Verilog HDL 语言，于是成立了 OVI（Open Verilog International）组织来负责 Verilog HDL 语言的发展。

3．Verilog HDL 的进展

基于 Verilog HDL 的优越性，IEEE 于 1995 年制定了 Verilog HDL 的 IEEE 标准，即 Verilog HDL1364-1995。其后，又在 2001 年发布了 Verilog HDL1364-2001 标准。
据有关文献报道，目前在美国使用 Verilog HDL 进行设计的工程师大约有 60000 人，全美国有 200 多所大学教授用 Verilog 硬件描述语言的设计方法。在我国台湾地区几乎所有著名大学的电子和计算机工程系都讲授 Verilog 有关的课程。

3.1.2 VHDL 和 Verilog HDL 语言对比

Verilog HDL 和 VHDL 都是用于逻辑设计的硬件描述语言，并且都已成为 IEEE 标准。VHDL 是在 1987 年成为 IEEE 标准，Verilog HDL 则在 1995 年才正式成为 IEEE 标准。
之所以 VHDL 比 Verilog HDL 早成为 IEEE 标准，这是因为 VHDL 是美国军方组织开发的，而 Verilog HDL 则是从一个普通的民间公司的私有财产转化而来。
VHDL 其英文全名为 VHSIC Hardware Description Language，而 VHSIC 则是 Very High Speed Integrated Circuit 的缩写，意为甚高速集成电路，故 VHDL 其准确的中文译名为甚高速集成电路的硬件描述语言。

1．共同点

Verilog HDL 和 VHDL 作为描述硬件电路设计的语言，其共同的特点在于。

- 能形式化地抽象表示电路的结构和行为。
- 支持逻辑设计中层次与领域的描述。
- 可借用高级语言的精巧结构来简化电路的描述。

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

- 具有电路仿真与验证机制以保证设计的正确性。
- 支持电路描述由高层到低层的综合转换。
- 硬件描述与实现工艺无关（有关工艺参数可通过语言提供的属性包括进去）。
- 便于文档管理，易于理解和设计重用。

2．不同点

但是 Verilog HDL 和 VHDL 又各有其自己的特点。

由于 Verilog HDL 早在 1983 年就已推出，因而 Verilog HDL 拥有更广泛的设计群体，成熟的资源也远比 VHDL 丰富。

与 VHDL 相比，Verilog HDL 的最大优点是：它是一种非常容易掌握的硬件描述语言，只要有 C 语言的编程基础，通过二十学时的学习，再加上一段时间的实际操作，可在二～三个月内掌握这种设计技术。

而掌握 VHDL 设计技术就比较困难。这是因为 VHDL 不很直观，需要有 Ada 编程基础。

目前版本的 Verilog HDL 和 VHDL 在行为级抽象建模的覆盖范围方面也有所不同。一般认为 Verilog HDL 在系统级抽象方面比 VHDL 略差一些，而在门级开关电路描述方面比 VHDL 强得多。

3.2 Verilog HDL 程序基本结构

Verilog HDL 是一种用于数字逻辑电路设计的语言。用 Verilog HDL 描述的电路设计就是该电路的 Verilog HDL 模型。Verilog HDL 既是一种行为描述的语言，也是一种结构描述的语言。也就是说，既可以用电路的功能描述，也可以用元器件和它们之间的连接来建立所设计电路的 Verilog HDL 模型。Verilog 模型可以是实际电路的不同级别的抽象。这些抽象的级别和它们对应的模型类型共有以下 5 种。

- 系统级（system）：用高级语言结构实现设计模块的外部性能的模型。
- 算法级（algorithm）：用高级语言结构实现设计算法的模型。
- RTL 级（Register Transfer Level）：描述数据在寄存器之间流动和如何处理这些数据的模型。
- 门级（gate-level）：描述逻辑门以及逻辑门之间的连接的模型。
- 开关级（switch-level）：描述器件中三极管和储存节点以及它们之间连接的模型。

一个复杂电路系统的完整 Verilog HDL 模型是由若干个 Verilog HDL 模块构成的，每一个模块又可以由若干个子模块构成。其中有些模块需要综合成具体电路，而有些模块只是与用户所设计的模块交互的现存电路或激励信号源。利用 Verilog HDL 语言结构所提供的这种功能就可以构造一个模块间的清晰层次结构来描述极其复杂的大型设计，并对所作设计的逻辑电路进行严格的验证。

Verilog HDL 行为描述语言作为一种结构化和过程性的语言，其语法结构非常适合于算法级和 RTL 级的模型设计。这种行为描述语言具有以下功能。

- 可描述顺序执行或并行执行的程序结构。
- 用延迟表达式或事件表达式来明确地控制过程的启动时间。
- 通过命名的事件来触发其他过程里的激活行为或停止行为。
- 提供了条件、if-else、case、循环程序结构。
- 提供了可带参数且非零延续时间的任务（task）程序结构。
- 提供了可定义新的操作符的函数结构（function）。
- 提供了用于建立表达式的算术运算符、逻辑运算符、位运算符。
- Verilog HDL 语言作为一种结构化的语言也非常适合于门级和开关级的模型设计。因其结构化的特点又使它具有以下功能。
  - — 提供了完整的一套组合型原语（primitive）;
  - — 提供了双向通路和电阻器件的原语;
  - — 可建立 MOS 器件的电荷分享和电荷衰减动态模型。

Verilog HDL 的构造性语句可以精确地建立信号的模型。这是因为在 Verilog HDL 中，提供了延迟和输出强

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

度的原语来建立精确程度很高的信号模型。信号值可以有不同的强度，可以通过设定宽范围的模糊值来降低不确定条件的影响。

Verilog HDL 作为一种高级的硬件描述编程语言，有着类似 C 语言的风格。其中 if 语句、case 语句等和 C 语言中的对应语句十分相似。如果读者已经掌握 C 语言编程的基础，那么学习 Verilog HDL 并不困难，只要对 Verilog HDL 某些语句的特殊方面着重理解，并加强上机练习就能很好地掌握它，利用它的强大功能来设计复杂的数字逻辑电路。下面将介绍 Verilog HDL 中的基本结构和语法。

3.2.1 Verilog HDL 程序入门

首先来看几个 Verilog HDL 程序，然后从中分析 Verilog HDL 程序的特性。

例 3.1：加法器。

```
module  adder  ( count,sum,a,b,cin ) ;        //加法器模块端口声明
     input [2:0] a,b;                         //端口说明
     input   cin;
     output  count;
     output [2:0] sum;
     assign {count,sum} = a + b + cin;        //加法器算法实现
endmodule
```

这个例子通过连续赋值语句描述了一个名为 adder 的三位加法器可以根据两个三比特数 a、b 和进位（cin）计算出和（sum）和进位（count）。从例子中可以看出整个 Verilog HDL 程序是嵌套在 module 和 endmodule 声明语句里的。

例 3.2：比较器。

```
module  compare  ( equal,a,b ) ;      //比较器模块端口声明
        output  equal;                //输出信号 equal
        input [1:0] a,b;              //输入信号 a、b
        assign  equal=(a==b)? 1: 0;//如果 a、b 两个输入信号相等,输出为 1,否则为 0
endmodule
```

这个程序通过连续赋值语句描述了一个名为 compare 的比较器。对两比特数 a、b 进行比较，如 a 与 b 相等，则输出 equal 为高电平，否则为低电平。在这个程序中，“/*.....*/”和“//.....”表示注释部分，注释只是为了方便程序员理解程序，对编译是不起作用的。

例 3.3：使用原语的三态驱动器。

```
module  trist2(out,in,enable);     //三态启动器模块端口声明
    output  out;                   //端口说明
    input   in, enable;
    bufif1  mybuf(out,in,enable); //实例化宏模块 bufif1
endmodule
```

这个例子描述了一个名为 trist2 的三态驱动器。程序通过调用一个在 Verilog 语言库中现存的三态驱动器实例元件 bufifl 来实现其功能。

例 3.4：自行设计的三态驱动器。

```
module  trist1(out,in,enable);     //三态启动器模块端口声明
    output  out;                   //端口说明
    input  in, enable;
    mytri  tri_inst(out,in,enable);//实例化由 mytri 模块定义的实例元件 tri_inst
endmodule
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

```
    module  mytri（out,in,enable）;     //三态启动器模块端口声明
    output  out;                        //端口说明
    input  in, enable;
    assign  out = enable? in : 'bz;     //三态启动器算法描述
endmodule
```

这个例子通过另一种方法描述了一个三态门。在这个例子中存在着两个模块。模块 trist1 调用由模块 mytri 定义的实例元件 tri_inst。模块 trist1 是顶层模块。模块 mytri 则被称为子模块。
通过上面的例子可以看到。

- Verilog HDL 程序是由模块构成的。每个模块的内容都是嵌在 module 和 endmodule 两个语句之间。每个模块实现特定的功能。模块是可以进行层次嵌套的。正因为如此,才可以将大型的数字电路设计分割成不同的小模块来实现特定的功能,最后通过顶层模块调用子模块来实现整体功能。
- 每个模块要进行端口定义,并说明输入输出口,然后对模块的功能进行行为逻辑描述。
- Verilog HDL 程序的书写格式自由,一行可以写几个语句,一个语句也可以分写多行。
- 除了 endmodule 语句外,每个语句和数据定义的最后必须有分号。
- 可以用“/*.....*/”和“//.....”对 Verilog HDL 程序的任何部分作注释。一个好的、有使用价值的源程序都应当加上必要的注释,以增强程序的可读性和可维护性。

3.2.2 模块的框架

模块的内容包括 I/O 声明、I/O 说明、内部信号声明和功能定义。

1．I/O 声明

模块的端口声明了模块的输入输出端口，其格式如下：

```
Module     模块名（端口 1，端口 2，端口 3，端口 4, …）;
```

2．I/O 说明

I/O 说明的格式如下：

```
输入口: input  端口名 1，端口名 2，…，端口名 i;        //（共有 i 个输入口）
输出口: output 端口名 1，端口名 2，…，端口名 j;        //（共有 j 个输出口）
```

I/O 说明也可以写在端口声明语句里，其格式如下：

```
module  module_name（input port1,input port2,…,output port1,output port2… ）
```

3．内部信号声明

在模块内用到的和与端口有关的 wire 和 reg 变量的声明，如下所示：

```
reg ［width-1 : 0］ R 变量 1，R 变量 2 …;
wire ［width-1 : 0］ W 变量 1，W 变量 2 …;
```

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

4．功能定义

模块中最重要的部分是逻辑功能定义部分，有 3 种方法可在模块中产生逻辑。
（1）用“assign”声明语句。

```
assign a = b & c;
```

这种方法的句法很简单，只需写一个“assign”，后面再加一个方程式即可。例子中的方程式描述了一个有两个输入的与门。
（2）用实例元件。

```
and and_inst（ q, a, b ）;
```

采用实例元件的方法在电路图输入方式下，调入库元件。键入元件的名字和相连的引脚即可，表示在设计中用到一个跟与门（and）一样的名为 and_inst 的与门，其输入端为 a、b，输出为 q。要求每个实例元件的名字必须是惟一的，以避免与其他调用与门（and）的实例混淆。
（3）用“always”块。

```
always @（posedge clk or posedge clr） begin    //时钟上升沿触发，异步清零
   if（clr）  q <= 0;                            //清零
   else  if（en） q <= d;                        //使能赋值
   end
```

采用“assign”语句是描述组合逻辑最常用的方法之一，而“always”块既可用于描述组合逻辑，也可描述时序逻辑。上面的例子用“always”块生成了一个带有异步清除端的 D 触发器。
“always”块可用很多种描述手段来表达逻辑，例如上例中就用了“if...else”语句来表达逻辑关系。如按一定的风格来编写“always”块，可以通过综合工具把源代码自动综合成用门级结构表示的组合或时序逻辑电路。
需要注意的是，如果用 Verilog 模块实现一定的功能，首先应该清楚哪些是同时发生的，哪些是顺序发生的。
上面 3 个例子分别采用了“assign”语句、实例元件和“always”块。这 3 个例子描述的逻辑功能是同时执行的。也就是说，如果把这 3 项写到一个 Verilog 模块文件中去，它们的次序不会影响逻辑实现的功能。这 3 项是同时执行的，也就是并发的。
然而，在“always”模块内，逻辑是按照指定的顺序执行的。“always”块中的语句称为“顺序语句”，因为它们是顺序执行的。请注意，两个或更多的“always”模块也是同时执行的，但是模块内部的语句是顺序执行的。
看一下“always”内的语句，就会明白它是如何实现功能的。“if…else… if”必须顺序执行，否则其功能就没有任何意义。如果 else 语句在 if 语句之前执行，功能就会不符合要求。为了能实现上述描述的功能，“always”模块内部的语句将按照书写的顺序执行。

3.3 Verilog HDL 语言的数据类型和运算符

3.3.1 常用数据类型

Verilog HDL 中总共有 19 种数据类型，数据类型是用来表示数字电路硬件中的数据储存和传送元素的。在本书中，我们先只介绍 4 个最基本的数据类型，它们分别是：reg 型，wire 型，integer 型和 parameter 型。其他数据类型在后面的章节里逐步介绍，读者也可以查阅附录中 Verilog HDL 语法参考书的有关章节逐步掌握。其他的类型如下：
large 型、medium 型、scalared 型、time 型、small 型、tri 型、trio 型、tri1 型、triand 型、trior 型、trireg 型、

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

vectored 型、wand 型和 wor 型。

这些数据类型除 time 型外都与基本逻辑单元建库有关，与系统设计没有很大的关系。在一般电路设计自动化的环境下，仿真用的基本部件库是由半导体厂家和 EDA 工具厂家共同提供的。系统设计工程师不必过多地关心门级和开关级的 Verilog HDL 语法现象。

Verilog HDL 语言中也有常量和变量之分，它们分别属于以上这些类型。下面对最常用的几种进行介绍。

3.3.1.1 常量

常量是在程序运行过程中其值不能被改变的量。下面首先对在 Verilog HDL 语言中使用的数字及其表示方式进行介绍。

1．数字

（1）整数。

在 Verilog HDL 中，整型常量有以下 4 种进制表示形式。

① 二进制整数（b 或 B）。

② 十进制整数（d 或 D）。

③ 十六进制整数（h 或 H）。

④ 八进制整数（o 或 O）。

数字表达方式有以下 3 种。

① <位宽><进制><数字>，这是一种全面的描述方式。

② <进制><数字>，在这种描述方式中，数字的位宽采用缺省位宽（这由具体的机器系统决定，但至少 32 位）。

③ <数字>，在这种描述方式中，采用缺省进制十进制。

在表达式中，位宽指明了数字的精确位数。例如：一个 4 位二进制数数字的位宽为 4，一个 4 位十六进制数数字的位宽为 16（因为每单个十六进制数要用 4 位二进制数来表示），如下例所示：

```
    8'b10101100     //位宽为 8 的数的二进制表示，'b 表示二进制
    8'ha2           //位宽为 8 的数的十六进制，'h 表示十六进制。
```

（2）x 和 z 值。

在数字电路中，x 代表不定值，z 代表高阻值。一个 x 可以用来定义十六/八/二进制数的四/三/一位二进制数的状态。z 的表示方式同 x 类似。z 还有一种表达方式是可以写作?。在使用 case 表达式时建议使用这种写法，以提高程序的可读性，如下例所示：

```
    4'b10x0         //位宽为 4 的二进制数从低位数起第二位为不定值
    4'b101z         //位宽为 4 的二进制数从低位数起第一位为高阻值
    12'dz           //位宽为 12 的十进制数其值为高阻值（第一种表达方式）
    12'd?           //位宽为 12 的十进制数其值为高阻值（第二种表达方式）
    8'h4x           //位宽为 8 的十六进制数其低四位值为不定值
```

（3）负数。

一个数字可以被定义为负数，只需在位宽表达式前加一个减号，并且减号必须写在数字定义表达式的最前面。注意减号不可以放在位宽和进制之间，也不可以放在进制和具体的数之间，如下例所示：

```
    -8'd5           //这个表达式代表 5 的补数（用 8 位二进制数表示）
    8'd-5           //非法格式
```

（4）下划线（underscore_）。

下划线可以用来分隔数字的表达以提高程序可读性。但不可以用在位宽和进制处，只能用在具体的数字之

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

间，例如：

```
16'b1010_1011_1111_1010          //合法格式
8'b_0011_1010                    //非法格式
```

当常量不声明位数时，默认值是 32 位，每个字母用 8 位的 ASCII 值表示，例如：

```
10 = 32'd10 = 32'b1010           //十进制和二进制
1=32'd1=32'b1                    //十进制和二进制
-1=-32'd1=32'hFFFFFFFF           //十进制和十六进制
'BX=32'BX=32'BXXXXXXX…X          //默认声明为 32 位
"AB"=16'B01000001_01000010       //每个字母用 8 位表示
```

2．参数（Parameter）

在 Verilog HDL 中用 parameter 来定义常量，即用 parameter 来定义一个标识符代表一个常量，称为符号常量，即标识符形式的常量。采用标识符代表一个常量可提高程序的可读性和可维护性。parameter 型数据是一种常数型的数据，其说明格式如下：

```
Parameter    参数名 1 = 表达式，参数名 2 = 表达式，…，参数名 n = 表达式;
```

parameter 是参数型数据的确认符，确认符后跟着一个用逗号分隔开的赋值语句表。在每一个赋值语句的右边必须是一个常数表达式。也就是说，该表达式只能包含数字或先前已定义过的参数，例如：

```
parameter  msb=7;                                  //定义参数 msb 为常量 7
parameter  e=25, f=29;                             //定义两个常数参数
parameter  r=5.7;                                  //声明 r 为一个实型参数
parameter  byte_size=8, byte_msb=byte_size-1;      //用常数表达式赋值
parameter  average_delay = (r+f)/2;                //用常数表达式赋值
```

参数型常数经常用于定义延迟时间和变量宽度。在模块或实例引用时可通过参数传递改变在被引用模块或实例中已定义的参数。下面将通过一个例子进一步说明在层次调用的电路中改变参数常用的一些用法。

```
module Decode(A,F);                                //模块声明
     parameter Width=1, Polarity=1;                //参数声明
     ……
endmodule

module Top;
     wire[3:0] A4;                                 //连线资源声明
     wire[4:0] A5;
     wire[15:0] F16;
     wire[31:0] F32;
     Decode  #(4,0) D1(A4,F16);                    //模块引用，并传递参数（4,0）
     Decode  #(5)  D2(A5,F32);                     //模块引用，并传递参数（5）
endmodule
```

在引用 Decode 实例时，D1 和 D2 的 Width 将采用不同的值，分别为 4 和 5，且 D1 的 Polarity 将为 0。可用例子中所用的方法来改变参数，即用“#（4，0）”向 D1 中传递“Width=4，Polarity=0”，用“#（5）”向 D2 中传递“Width=5，Polarit=1”。

开发板免费领取试用 开发板免费领取试用

3.3.1.2 变量

变量是在程序运行过程中，其值可以改变的量。在 Verilog HDL 中变量类型有很多种，这里只对常用的几种变量进行介绍。

1．网络类型变量

网络类型表示结构实体（例如门）之间的物理连接。网络类型的变量不能储存值，而且它必需受到驱动器（例如门或连续赋值语句，assign）的驱动。如果没有驱动器连接到网络类型的变量上，则该变量就是高阻的，即其值为 z。

常用的网络类型变量包括 wire 型和 tri 型。这两种变量都是用于连接器件单元，它们具有相同的语法格式和功能。之所以提供这两种名字来表达相同的概念是为了与模型中所使用的变量的实际情况相一致。

wire 型变量通常是用来表示单个门驱动或连续赋值语句驱动的网络型数据，tri 型变量则用来表示多驱动器驱动的网络型数据。如果 wire 型或 tri 型变量没有定义逻辑强度（logic strength），在多驱动源的情况下，逻辑值会发生冲突，从而产生不确定值。

表 3.1 所示为在同等驱动强度下，两个驱动源驱动的 wire 型和 tri 型变量的真值表。

表 3.1　　wire/tri 型变量真值表

wire/tri 型变量双驱动源运算结果				
驱动源 1 / 驱动源 2	0	1	x	z
0	0	x	x	0
1		1	x	1
x	x	x	x	x
z	0	1	x	z

wire 型变量常用来表示用于以 assign 关键字指定的组合逻辑信号。Verilog 程序模块中输入/输出信号类型缺省时自动定义为 wire 型。wire 型变量可以用作任何方程式的输入，也可以用作“assign”语句或实例元件的输出。wire 型变量的声明格式如下：

```
wire [n-1:0] 变量名 1,变量名 2,…,变量名 i;   //共有 i 条总线，每条总线内有 n 条线路
```

也可以如下表示：

```
wire [n:1] 变量名 1,变量名 2,…,变量名 i;    //共有 i 条总线，每条总线内有 n 条线路
```

其中，wire 是 wire 型变量的确认符，[n-1:0] 和 [n:1] 代表该变量的位宽，即该变量有几位，最后跟着的是变量的名字。如果一次定义多个变量，变量名之间用逗号隔开。声明语句的最后要用分号表示语句结束。如下所示：

```
wire  a;                          //定义了一个一位的 wire 型变量
wire [7:0] b;                     //定义了一个八位的 wire 型变量
wire [4:1] c, d;                  //定义了两个四位的 wire 型变量
```

2．寄存器型变量

寄存器是数据储存单元的抽象。寄存器型变量的关键字是 reg。通过赋值语句可以改变寄存器储存的值，其作用与改变触发器储存的值相当。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

Verilog HDL 语言提供了功能强大的结构语句使设计者能有效地控制是否执行这些赋值语句。这些控制结构用来描述硬件触发条件，例如时钟的上升沿和多路器的选通信号。reg 类型变量的缺省初始值为不定值，即 x。

reg 型变量常用来表示用于“always”模块内的指定信号，常代表触发器。通常，在设计中要由“always”块通过使用行为描述语句来表达逻辑关系。在“always”块内被赋值的每一个信号都必须定义成 reg 型。和 wire 型变量类似，reg 型变量的声明格式如下：

```
reg [n-1:0] 变量名 1,变量名 2,…,变量名 i;      //共有 i 条总线，每条总线内有 n 条线路
```

也可以如下表示：

```
reg [n:1]   变量名 1,变量名 2,…,变量名 i;      //共有 i 条总线，每条总线内有 n 条线路
```

其中，reg 是 reg 型变量的确认标识符，[n-1:0] 和 [n:1] 代表该变量的位宽，即该变量有几位（bit），最后跟着的是变量的名字。如果一次定义多个变量，变量名之间用逗号隔开。声明语句的最后要用分号表示语句结束。如下所示：

```
reg  rega;                    //定义了一个一位的名为 rega 的 reg 型变量
reg [3:0]  regb;              //定义了一个四位的名为 regb 的 reg 型变量
reg [4:1]  regc, regd;        //定义了两个四位的名为 regc 和 regd 的 reg 型变量
```

reg 型变量可以赋正值，也可以赋负值。但当一个 reg 型变量是一个表达式中的操作数时，它的值将被当作是无符号值，即正值。例如：当一个四位的寄存器用作表达式中的操作数时，如果开始寄存器被赋以值-1，则在表达式中进行运算时，其值被认为是+15。

3．存储器型变量

Verilog HDL 通过对 reg 型变量建立数组来对存储器建模，用于描述 RAM 型存储器、ROM 存储器和 reg 文件。数组中的每一个单元通过一个数组索引进行寻址。由于在 Verilog 语言中没有多维数组存在，因此 memory 型数据是通过扩展 reg 型数据的地址范围来生成的。其格式如下：

```
reg [n-1:0] 存储器名[m-1:0];
```

或：

```
reg [n-1:0] 存储器名[m:1];
```

在这里，reg [n-1:0] 定义了存储器中每一个存储单元的大小，即该存储单元是一个 n 位的寄存器。存储器名后的 [m-1:0] 或 [m:1] 则定义了该存储器中有多少个这样的寄存器。最后用分号结束定义语句。下面举例说明：

```
reg [7:0]  mema[255: 0];          //定义一个名为 mema 的 256×8 的存储器
```

这个例子定义了一个名为 mema 的存储器，该存储器有 256 个 8 位的存储器。该存储器的地址范围是 0～255。需要注意的是，对存储器进行地址索引的表达式必须是常数表达式。

另外，在同一个数据类型声明语句里，可以同时定义存储器型数据和 reg 型数据。 例如：

```
parameter  wordsize=16, memsize=256;                    //定义两个参数
reg [wordsize-1:0] mem[memsize-1:0],writereg, readreg;  //使用可变参数来定义存储器
```

尽管 memory 型数据和 reg 型数据的定义格式很相似，但要注意其不同之处。如一个由 n 个 1 位寄存器构成的存储器组是不同于一个 n 位的寄存器的，如下所示：

```
reg [n-1:0] rega;              //一个 n 位的寄存器
reg mema [n-1:0];              //一个由 n 个 1 位寄存器构成的存储器组
```


开发板免费领取试用 开发板免费领取试用

一个 n 位的寄存器可以在一条赋值语句里进行赋值，而一个完整的存储器则不行，例如：

```
rega =0;                          //合法赋值语句
mema =0;                          //非法赋值语句
```

如果想对 memory 中的存储单元进行读写操作，必须指定该单元在存储器中的地址。下面的写法是正确的。

```
mema[3]=0;                    //给 memory 中的第 3 个存储单元赋值为 0。
```

进行寻址的地址索引可以是表达式，这样就可以对存储器中的不同单元进行操作。表达式的值可以取决于电路中其他的寄存器的值。例如可以用一个加法计数器来做 RAM 的地址索引。

3.3.2 常用运算符

Verilog HDL 语言的运算符范围很广，其运算符按其功能可分为以下几类。

- 算术运算符：（+, - , × ， /,% ）。
- 赋值运算符：（=,<= ）。
- 关系运算符：（>,<,>=,<= ）。
- 逻辑运算符：（ &&,||,! ）。
- 条件运算符：（ ?: ）。
- 位运算符：（ ~ ,|,^,&,^~ ）。
- 移位运算符：（ <<,>> ）。
- 拼接运算符：（ { } ）。
- 其他

在 Verilog HDL 语言中运算符所带的操作数是不同的，按其所带操作数的个数运算符可分为以下 3 种。

单目运算符（unary operator）：可以带一个操作数，操作数放在运算符的右边。

二目运算符（binary operator）：可以带两个操作数，操作数放在运算符的两边。

三目运算符（ternary operator）：可以带三个操作数，这三个操作数用三目运算符分隔开。

例如：

```
clock = ~clock;          // ~ 是一个单目取反运算符,clock 是操作数。
c = a | b;               // | 是一个二目按位或运算符,a 和 b 是操作数。
r = s ? t : u;           // ?: 是一个三目条件运算符,s，t，u 是操作数。
```

下面对常用的几种运算符进行介绍。

1．基本的算术运算符

在 Verilog HDL 语言中，算术运算符又称为二进制运算符，共有下面几种。

- +:（加法运算符或正值运算符,如 ega+regb、+3 ）。
- −:（减法运算符或负值运算符，如 rega−3、−3 ）。
- ×:（乘法运算符，如 rega×3 ）。
- /:（除法运算符，如 5/3 ）。
- % :（模运算符或求余运算符，要求%两侧均为整型数据，如 7%3 的值为 1 ）。

在进行整数除法运算时，结果值要略去小数部分，只取整数部分。而进行取模运算时，结果值的符号位采用模运算式里第一个操作数的符号位，例如：

```
10%3        1       //余数为 1
11%3        2       //余数为 2
12%3        0       //余数为 0，即无余数
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

```
-10%3        -1      //结果取第一个操作数的符号位,所以余数为-1
11%3         2       //结果取第一个操作数的符号位,所以余数为 2.
```

注意 在进行算术运算操作时，如果某一个操作数有不确定的值 x，则整个结果也为不定值 x。

2．位运算符

Verilog HDL 作为一种硬件描述语言是针对硬件电路而言的。在硬件电路中信号有 4 种状态值 1、0、x 和 z。在电路中信号进行与或非时，反映在 Verilog HDL 中则是相应的操作数的位运算。Verilog HDL 提供了以下 5 种位运算符。

- ~ ：（取反）
- & ：（按位与）
- | ：（按位或）
- ^ ：（按位异或）
- ^~：（按位同或（异或非））

说明：

- 位运算符中除了 ~ 是单目运算符以外，均为二目运算符,即要求运算符两侧各有一个操作数。
- 位运算符中的二目运算符要求对两个操作数的相应位进行运算操作。

下面对各运算符分别进行介绍。

- “取反”运算符 ~

～是一个单目运算符，用来对一个操作数进行按位取反运算。如表 3.2 所示为单目运算符～的运算规则表。

表 3.2 ～ 运算规则表

～运算	
操 作 数	结 果
1	0
0	1
x	x

举例说明：

```
rega='b1010;          //rega 的初值为'b1010
rega=~rega;           //rega 的值进行取反运算后变为'b0101
```

- “按位与”运算符&

按位与运算就是将两个操作数的相应位进行与运算，其运算规则如表 3.3 所示。

表 3.3 & 运算规则表

& 运算			
操作数 1 / 操作数 2	0	1	x
0	0	0	0
1	0	1	x
x	0	x	x

- “按位或”运算符 |

按位或运算就是将两个操作数的相应位进行或运算，其运算规则如表 3.4 所示。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

表 3.4 | 运算规则表

| 运算			
操作数 1 / 操作数 2	0	1	x
0	0	1	x
1	1	1	1
x	x	1	x

- “按位异或”运算符^（也称之为 XOR 运算符）

按位异或运算就是将两个操作数的相应位进行异或运算，其运算规则如表 3.5 所示。

表 3.5 ^ 运算规则表

^ 运算			
操作数 1 / 操作数 2	0	1	x
0	0	1	x
1	1	0	x
x	x	x	x

- “按位同或”运算符^~

按位同或运算就是将两个操作数的相应位先进行异或运算再进行非运算，其运算规则如表 3.6 所示。

表 3.6 ^~ 运算规则表

^~ 运算			
操作数 1 / 操作数 2	0	1	x
0	1	0	x
1	0	1	x
x	x	x	x

- 不同长度的数据进行位运算

两个长度不同的数据进行位运算时，系统会自动将两者按右端对齐。位数少的操作数会在相应的高位用 0 填满，以使两个操作数按位进行操作。

3．逻辑运算符

在 Verilog HDL 语言中存在 3 种逻辑运算符。

- &&:（逻辑与）
- ||:（逻辑或）
- ！:（逻辑非）

“&&”和“||”是二目运算符，它要求有两个操作数，如（a>b）&&（b>c），（a<b）||（b<c）。“!”是单目运算符，只要求一个操作数，如!（a>b）。如表 3.7 所示为逻辑运算的真值表。它表示当 a 和 b 的值为不同的组合时，各种逻辑运算所得到的值。

表 3.7 逻辑运算真值表

操 作 数		逻辑运算及结果			
a	b	!a	!b	a&&b	a||b
真	真	假	假	真	真

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

真	假	假	真	假	真
假	真	真	假	假	真
假	假	真	真	假	假

逻辑运算符中“&&”和“||”的优先级别低于关系运算符，“!”的优先级别高于算术运算符，例如。

```
(a>b)&&(x>y)        可写成:     a>b && x>y
(a==b)||(x==y)      可写成:     a==b || x==y
(!a)||(a>b)         可写成:     !a || a>b
```

为了提高程序的可读性，明确表达各运算符间的优先关系，建议使用括号。

4．关系运算符

关系运算符共有以下 4 种。

- a＜b:（a 小于 b）
- a＞b:（a 大于 b）
- a＜=b:（a 小于或等于 b）
- a＞=b:（a 大于或等于 b）

在进行关系运算时，如果声明的关系是假的（flase），则返回值是 0；如果声明的关系是真的（true），则返回值是 1；如果某个操作数的值不定，则关系是模糊的，返回值是不定值。
所有的关系运算符有着相同的优先级别。关系运算符的优先级别低于算术运算符的优先级别，例如。

```
a < size-1          //这种表达方式等同于下面一行的表达方式
a < (size-1)
size - (1 < a)     //这种表达方式不等同于下面一行的表达方式
size - 1 < a
```

从上面的例子可以看出这两种不同运算符的优先级别。当表达式 size -（1<a）进行运算时，关系表达式先被运算，然后返回结果值 0 或 1 被 size 减去。而当表达式 size -1<a 进行运算时，size 先被减去 1，然后再同 a 相比。

5．等式运算符

在 Verilog HDL 语言中存在 4 种等式运算符。

- ==:（等于）
- != :（不等于）
- ===:（等于）
- !==:（不等于）

这 4 个运算符都是二目运算符，它要求有两个操作数。“==”和“!=”又称为逻辑等式运算符，其结果由两个操作数的值决定。由于操作数中某些位可能是不定值 x 和高阻值 z，结果可能为不定值 x。
“===”和“!==”运算符则不同，它在对操作数进行比较时，对某些位的不定值 x 和高阻值 z 也进行比较。两个操作数必需完全一致，其结果才是 1，否则为 0。“===”和“!==”运算符常用于 case 表达式的判别，所以又称为“case 等式运算符”。
这 4 个等式运算符的优先级别是相同的。下面画出“==”与“===”的真值表，帮助理解两者间的区别。

表 3.8　　等式运算符真值表

	=== 运算			
操作数 1	0	1	x	z

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

操作数 2				
0	1	0	0	0
1	0	1	0	0
x	0	0	1	0
z	0	0	0	1

== 运算				
操作数 1 / 操作数 2	0	1	x	z
0	1	0	x	x
1	0	1	x	x
x	x	x	x	x
z	x	x	x	x

下面举一个例子说明“==”与“===”的区别。

```
if（A==1'bx） $display（ "AisX" ）;     //当 A 等于 X 时，这个语句不执行
if（A===1'bx） $display（ "AisX" ）;    //当 A 等于 X 时，这个语句执行
```

6．移位运算符

在 Verilog HDL 中有两种移位运算符。

- <<:（左移位运算符）
- >>:（右移位运算符）

其使用方法如下：

```
a >> n;
a << n;
```

a 代表要进行移位的操作数，n 代表要移几位。这两种移位运算都用 0 来填补移出的空位。下面举例说明：

```
module  shift;
    reg [3:0]  start, result;
    initial begin
        start  = 1;                 //start 在初始时刻设为值 0001
        result = （start<<2）;      //移位后，start 的值 0100，然后赋给 result
    end
endmodule
```

从上面的例子可以看出，start 在移过两位以后，用 0 来填补空出的位。进行移位运算时应注意移位前后变量的位数，下面举例说明。

```
4'b1001<<1 = 5'b10010;              //左移 1 位后用 0 填补低位
4'b1001<<2 = 6'b100100;             //左移 2 位后用 00 填补低位
1<<6 = 32'b1000000;                 //左移 6 位后用 000000 填补低位
4'b1001>>1 = 4'b0100;               //右移 1 位后，低 1 位丢失，高 1 位用 0 填补
4'b1001>>4 = 4'b0000;               //右移 4 位后，低 4 位丢失，高 4 位用 0 填补
```

7．位拼接运算符（Concatation）

开发板免费领取试用 开发板免费领取试用

在 Verilog HDL 语言有一个特殊的运算符：位拼接运算符{}。用这个运算符可以把两个或多个信号的某些位拼接起来进行运算操作。其使用方法如下：

```
{信号 1 的某几位，信号 2 的某几位，..，..，信号 n 的某几位}
```

即把某些信号的某些位详细地列出来，中间用逗号分开，最后用大括号括起来表示一个整体信号，例如：

```
{a,b[3:0],w,3'b101}
```

也可以写成为：

```
{a,b[3],b[2],b[1],b[0],w,1'b1,1'b0,1'b1}
```

在位拼接表达式中不允许存在没有指明位数的信号。这是因为在计算拼接信号的位宽的大小时必需知道其中每个信号的位宽。

位拼接也可以用重复法来简化表达式，如下所示：

```
{4{w}}          //等同于{w,w,w,w}
```

位拼接还可以用嵌套的方式来表达，如下所示：

```
{b,{3{a,b}}}    //等同于{b,a,b,a,b,a,b}
```

用于表示重复的表达式必须是常数表达式，如上例中的 4 和 3。

8．缩减运算符（reduction operator）

缩减运算符是单目运算符,也有与、或、非运算。其与、或、非运算规则类似于位运算符的与、或、非运算规则,但其运算过程不同。位运算是对操作数的相应位进行与、或、非运算,操作数是几位数，则运算结果也是几位数。而缩减运算则不同,缩减运算是对单个操作数进行与、或、非递推运算,最后的运算结果是一位的二进制数。

缩减运算的具体运算过程如下。

（1）先将操作数的第一位与第二位进行与、或、非运算。

（2）将运算结果与第三位进行与、或、非运算，依次类推，直至最后一位。

例如：

```
reg [3:0] B;
reg C;
C = &B;
```

相当于：

```
C =( ( B[0]&B[1] ) & B[2] ) & B[3];
```

由于缩减运算的与、或、非运算规则类似于位运算符与、或、非运算规则，这里不再详细讲述，可参照位运算符的运算规则介绍。

9．优先级别

各种运算符的优先级别关系如表 3.9 所示。

表 3.9　　运算符优先级别表

运　算　符	优 先 级 别
！～	高
* / %	

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

+ − << >> < <= > >= == != === !== & ^ ^~ | && || ? :	 低

3.4 Verilog HDL 语言的赋值语句和块语句

3.4.1 非阻塞赋值和阻塞赋值

在 Verilog HDL 语言中，信号有两种赋值方式：非阻塞（Non_Blocking）赋值方式和阻塞（Blocking）赋值方式。

（1）非阻塞赋值方式。

```
典型语句: b <= a;
```

① 块结束后才完成赋值操作。
② b 的值并不是立刻就改变的。
③ 这是一种比较常用的赋值方法，特别在编写可综合模块时。

（2）阻塞赋值方式。

```
典型语句: b = a;
```

① 赋值语句执行完后，块才结束。
② b 的值在赋值语句执行完后立刻就改变。
③ 可能会产生意想不到的结果。

非阻塞赋值方式和阻塞赋值方式的区别常给设计人员带来问题。问题主要是给“always”块内的 reg 型信号的赋值方式不易把握。到目前为止，前面所举的例子中的“always”模块内的 reg 型信号都是采用下面的这种赋值方式：

```
b <= a;
```

这种方式的赋值并不是马上执行的，也就是说“always”块内的下一条语句执行后，b 并不等于 a，而是保持原来的值。“always”块结束后，才进行赋值。而另一种赋值方式阻塞赋值方式，如下所示：

```
b = a;
```

这种赋值方式是马上执行的，也就是说执行下一条语句时，b 已等于 a。尽管这种方式看起来很直观，但是可能引起麻烦。下面举例说明。

例 3.5：非阻塞赋值。

```
always @( posedge clk ) begin
     b<=a;
     c<=b;
end
```

例 3.5 中的“always”块中用了非阻塞赋值方式，定义了两个 reg 型信号 b 和 c。clk 信号的上升沿到来时，b 就等于 a，c 就等于 b，这里应该用到了两个触发器。需要注意的是赋值是在“always”块结束后执行的，

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

c 应为原来 b 的值。这个“always”块实际描述的电路功能如图 3.1 所示。

例 3.6：阻塞型赋值。

```
always @(posedge clk) begin
    b=a;
    c=b;
end
```

例 3.6 中的“always”块用了阻塞赋值方式。clk 信号的上升沿到来时，将发生如下的变化：b 马上取 a 的值，c 马上取 b 的值（即等于 a）。综合的电路如图 3.2 所示。

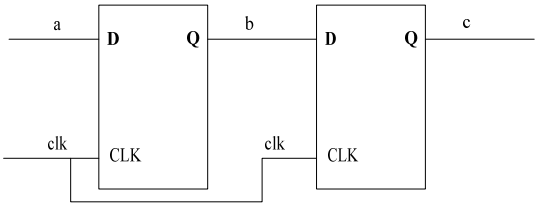


图 3.1　非阻塞赋值综合电路

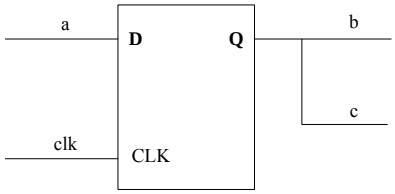


图 3.2　阻塞赋值综合电路

它只用了一个触发器来寄存 a 的值，并同时输出给 b 和 c。这不是设计者的初衷，如果采用例 3.5 所示的非阻塞赋值方式就可以避免这种错误。

3.4.2　块语句

块语句通常用来将两条或多条语句组合在一起，使其在格式上看更像一条语句。块语句有两种：一种是 begin_end 语句，通常用来标识顺序执行的语句，用它来标识的块称为顺序块；另一种是 fork_join 语句，通常用来标识并行执行的语句，用它来标识的块称为并行块。下面进行详细的介绍。

1．顺序块

顺序块有以下特点。

（1）块内的语句是按顺序执行的，即只有上面一条语句执行完后下面的语句才能执行。

（2）每条语句的延迟时间是相对于前一条语句的仿真时间而言的。

（3）直到最后一条语句执行完，程序流程控制才跳出该语句块。

顺序块的格式如下：

```
begin
    语句 1;
    语句 2;
    ......
    语句 n;
end
```

或者：

```
begin:块名
    块内声明语句
    语句 1;
    语句 2;
    ......
    语句 n;
```

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

```
    end
```

其中：

（1）块名即该块的名字，是一个标识符，其作用后面再详细介绍。

（2）块内声明语句可以是参数声明语句，reg 型变量声明语句，integer 型变量声明语句或者 real 型变量声明语句。

下面举例说明。

例 3.7：顺序块。

```
    begin
        areg = breg;
        creg = areg;   //creg 的值为 breg 的值
    end
```

从该例可以看出，第一条赋值语句先执行，areg 的值更新为 breg 的值。然后程序流程控制转到第二条赋值语句，creg 的值更新为 areg 的值。因为这两条赋值语句之间没有任何延迟时间，creg 的值实为 breg 的值。当然可以在顺序块里延迟控制时间来分开两个赋值语句的执行时间，如例 3.8 所示。

例 3.8：加延时顺序块。

```
    begin
        areg = breg;
        #10 creg = areg;   //在两条赋值语句间延迟 10 个时间单位
    end
```

2．并行块

并行块有以下 4 个特点。

（1）块内语句是同时执行的，即程序流程控制一进入该并行块，块内语句则开始同时并行地执行。

（2）块内每条语句的延迟时间是相对于程序流程控制进入到块内时的仿真时间的。

（3）延迟时间是用来给赋值语句提供执行时序的。

（4）当按时间时序排序在最后的语句执行完后或一个 disable 语句执行时，程序流程控制跳出该程序块。

并行块的格式如下：

```
    fork
        语句 1;
        语句 2;
        .......
        语句 n;
    join
```

或者：

```
    fork:块名
        块内声明语句
        语句 1;
        语句 2;
        ......
        语句 n;
    join
```

其中：

（1）块名即标识该块的一个名字，相当于一个标识符。

开发板免费领取试用 开发板免费领取试用

（2）块内说明语句可以是参数说明语句、reg 型变量声明语句、integer 型变量声明语句、real 型变量声明语句、ime 型变量声明语句或者事件（event）说明语句。

下面举例说明。

例 3.9：并行块 1。

```
fork
     #50   r = 'h35;      //在绝对时间 50 单位后，r 被赋值
     #100  r = 'hE2;      //在绝对时间 100 单位后（非绝对时间 150），r 再次被赋值
     #150  r = 'h00;
     #200  r = 'hF7;
     #250  -> end_wave;  //在绝对时间 250 单位后，触发事件 end_wave
join
```

在这个例子中用并行块来替代前面例子中的顺序块来产生波形，用这两种方法生成的波形是一样的。

3．块名

在 Verilog HDL 语言中，可以给每一个块取名字，只需将名字加在关键词 begin 或 fork 后面即可，这样做的原因有以下几点。

（1）这样可以在块内定义局部变量，即只在块内使用的变量。

（2）这样可以允许块被其他语句调用，如被 disable 语句调用。

（3）在 Verilog 语言里，所有的变量都是静态的，即所有的变量都只有一个惟一的存储地址，因此进入或跳出块并不影响存储在变量内的值。

基于以上原因，块名就提供了一个在任何仿真时刻确认变量值的方法。需要注意的是，块名和变量名一样，都不能是关键词。Verilog 的关键词参见 3.4.3 小节。

4．起始时间和结束时间

在并行块和顺序块中都有一个起始时间和结束时间的概念。对于顺序块，起始时间就是第一条语句开始被执行的时间，结束时间就是最后一条语句执行完的时间。而对于并行块来说，起始时间对于块内所有的语句是相同的，即程序流程控制进入该块的时间，其结束时间是按时间排序在最后的语句执行完的时间。

当一个块嵌入另一个块时，块的起始时间和结束时间是很重要的。跟在块后面的语句只有在该块的结束时间到了才能开始执行，也就是说，只有该块完全执行完后，后面的语句才可以执行。

在 fork_join 块内，各条语句不必按顺序给出，因此在并行块里，各条语句在前还是在后是无关紧要的，如下所示。

例 3.10：并行块 2。

```
fork
     #250  -> end_wave;      //按下面几条语句顺序执行结果和例［3.10］的执行结果一样
     #200  r = 'hF7;
     #150  r = 'h00;
     #100  r = 'hE2;
     #50   r = 'h35;
join
```

在这个例子中，各条语句并不是按被执行的先后顺序给出的，但同样可以生成前面例子中的波形。

3.4.3 关键词

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

在 Verilog HDL 中，所有的关键词是事先定义好的确认符，用来组织语言结构。关键词是用小写字母定义的，因此在编写原程序时要注意关键词的书写，以避免出错。下面是 Verilog HDL 中使用的关键词（请参阅附录：Verilog 语言参考手册）：

always、and、assign、begin、buf、bufif0、bufif1、case、casex、casez、cmos、deassign、default、defparam、disable、edge、else、end、endcase、endmodule、endfunction、endprimitive、endspecify、endtable、endtask、event、for、force、forever、fork、function、highz0、highz1、if、initial、inout、input、integer、join、large、macromodule、medium、module、nand、negedge、nmos、nor、not、notif0、notif1、or、output、parameter、pmos、posedge、primitive、pull0、pull1、pullup、pulldown、rcmos、reg、releses、repeat、mmos、rpmos、rtran、rtranif0、rtranif1、scalared、small、specify、specparam、strength、strong0、strong1、supply0、supply1、table、task、time、tran、tranif0、tranif1、tri、tri0、tri1、triand、trior、trireg、vectored、wait、wand、weak0、weak1、while、wire、wor、xnor、xor。

在编写 Verilog HDL 程序时，变量名、端口名、块名等的定义不要与这些关键词冲突。

3.5 Verilog HDL 语言的条件语句

3.5.1 if 语句

if 语句是用来判定所给定的条件是否满足，根据判定的结果（真或假）决定执行给出的两种操作之一。Verilog HDL 语言提供了 3 种形式的 if 语句。

（1）无分支。

语法形式：

```
if （表达式） 语句;
```

例如：

```
if （a > b） out1 <= int1;    //若 a 大于 b，将 int1 赋予 out1
```

（2）单级分支。

语法形式：

```
if（表达式）语句 1;
else 语句 2;
```

例如：

```
if（a>b） out1<=int1;      //若 a 大于 b，将 int1 赋予 out1；否则，将 int2 赋予 out1
else out1<=int2;
```

（3）多级分支

语法形式：

```
if（表达式 1）          语句 1;
else  if（表达式 2）    语句 2;
else  if（表达式 3）    语句 3;
...
else  if（表达式 m）    语句 m;
else                    语句 n;
```

例如：

```
if（a>b）          out1<=int1;       //若 a 大于 b，将 int1 赋予 out1
else  if（a==b）   out1<=int2;       //否则，如果 a 等于 b，将 int2 赋予 out1
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

```
else              out1<=int3;       //否则，将 int3 赋予 out1
```

关于 if 语句有如下 6 点说明。

（1）3 种形式的 if 语句中，在 if 后面都有“表达式”，一般为逻辑表达式或关系表达式。系统对表达式的值进行判断，若为 0、x 或 z，按“假”处理；若为 1，按“真”处理，执行指定的语句。

（2）第二、第三种形式的 if 语句中，在每个 else 前面有一分号，整个语句结束处有一分号。

这是由于分号是 Verilog HDL 语句中不可缺少的部分，这个分号是 if 语句中的内嵌套语句所要求的。如果无此分号，则出现语法错误。

但应注意，不要误认为上面是两个语句（if 语句和 else 语句）。它们都属于同一个 if 语句。else 子句不能作为语句单独使用，它必须是 if 语句的一部分，与 if 配对使用。

（3）在 if 和 else 后面可以包含一个内嵌的操作语句，也可以有多个操作语句，此时用 begin 和 end 这两个关键词将几个语句包含起来成为一个复合块语句如下所示。

```
if（a>b）begin                    //使用 begin_end 语句实现多个赋值操作
    out1<=int1;
    out2<=int2;
end
else begin
    out1<=int2;
    out2<=int1;
end
```

注意 在 end 后不需要再加分号，因为 begin_end 内是一个完整的复合语句，不需再附加分号。

（4）允许一定形式的表达式简写方式，例如：

```
if（expression）  等同于  if（ expression == 1 ）
if（! expression）等同于  if（ expression != 1 ）
```

（5）if 语句的嵌套。

在 if 语句中又包含一个或多个 if 语句，称为 if 语句的嵌套，一般形式如下：

```
if（expression1）
    if（expression2）     语句 1     （内嵌 if）
    else   语句 2
else if（expression3）    语句 3     （内嵌 if）
else   语句 4
```

应当注意 if 与 else 的配对关系，else 总是与它上面的最近的 if 配对。如果 if 与 else 的数目不一样，为了实现程序设计者的企图，可以用 begin_end 块语句来确定配对关系，例如：

```
if（） begin
    if（）  语句 1     （内嵌 if）
end
else  语句 2
```

这时 begin_end 块语句限定了内嵌 if 语句的范围，因此 else 与第一个 if 配对。注意 begin_end 块语句在 if_else 语句中的使用，因为有时 begin_end 块语句的不慎使用会改变逻辑行为，如下所示：

```
if（index>0）                                    //内嵌 for 语句，无 else 分支
   for（scani=0;scani<index;scani=scani+1）     //内嵌 if 语句
     if（memory［scani］>0） begin              //使用 begin_end 语句，有 else 分支
         $display（"..."）;
```


开发板免费领取试用 开发板免费领取试用

```
            memory[scani]=0;
        end
        else                                    //此处为内嵌 if 语句的分支
            $display("error-indexiszero");
...
```

尽管程序设计者把 else 写在与第一个 if（外层 if）同一列上，希望与第一个 if 对应，但实际上 else 是与第二个 if 对应，因为它们相距最近。正确的写法如下：

```
if(index>0) begin                           //内嵌 for 语句，有 else 分支
    for(scani=0;scani<index;scani=scani+1)  //内嵌 if 语句
        if(memory[scani]>0) begin          //使用 begin_end 语句，无 else 分支
            $display("...");
            memory[scani]=0;
      end
end
else                                        //此处为外部 if 语句的分支
    $display("error-indexiszero");
```

（6）if_else 例子。

下面这段程序用 if_else 语句来检测变量 index 以决定 modify-seg1、modify-seg2、modify-seg3 中哪一个的值应当与 index 相加作为 memory 的寻址地址。并且将相加值存入寄存器 index 以备下次检测使用。程序的前 10 行定义寄存器和参数。

```
reg [31:0]  instruction, segment_area[255:0];   //定义寄存器
reg [7:0]   index;
reg [5:0]   modify_seg1, modify_seg2, modify_seg3;
parameter                                       //定义参数
    segment1=0,  inc_seg1=1,
    segment2=20, inc_seg2=2,
    segment3=64, inc_seg3=4,
    data=128;
//检测寄存器 index 的值
if(index<segment2) begin                        //index<20 时，执行下列操作
    instruction = segment_area[index + modify_seg1];
    index = index + inc_seg1;
end
else  if(index<segment3) begin                  //20<index<64 时，执行下列操作
    instruction = segment_area[index + modify_seg2];
    index = index + inc_seg2;
end
else  if (index<data) begin                     //64<index<128 时，执行下列操作
    instruction = segment_area[index + modify_seg3];
    index = index + inc_seg3;
end
else                                            //index>128 时，执行下列操作
    instruction = segment_area[index];
```

3.5.2 case 语句

开发板免费领取试用 开发板免费领取试用

case 语句是一种多分支选择语句，if 语句只有两个分支可供选择，而实际问题中常常需要用到多分支选择。Verilog 语言提供的 case 语句直接处理多分支选择。case 语句通常用于微处理器的指令译码，它的一般形式如下：

（1）case （表达式） <case 分支项> endcase
（2）casez（表达式） <case 分支项> endcase
（3）casex（表达式） <case 分支项> endcase

<case 分支项>的一般语法格式如下：

```
分支表达式:       语句
缺省项（default 项）:      语句
```

关于 case 语句的几点说明如下。

（1）case 括弧内的表达式称为控制表达式，case 分支项中的表达式称为分支表达式。控制表达式通常表示为控制信号的某些位，分支表达式则用这些控制信号的具体状态值来表示，因此分支表达式又可以称为常量表达式。

（2）当控制表达式的值与分支表达式的值相等时，就执行分支表达式后面的语句。如果所有的分支表达式的值都没有与控制表达式的值相匹配的，就执行 default 后面的语句。

（3）default 项可有可无，一个 case 语句里只能有一个 default 项。下面是一个简单的使用 case 语句的例子。该例子中对寄存器 rega 译码以确定 result 的值。

```
reg [15:0] rega;
reg [9:0]  result;
case（rega）
    16 'd0:  result = 10 'b0111111111;      //rega 等于 0 时
    16 'd1:  result = 10 'b1011111111;      //rega 等于 1 时
    16 'd2:  result = 10 'b1101111111;      //rega 等于 2 时
    16 'd3:  result = 10 'b1110111111;      //rega 等于 3 时
    16 'd4:  result = 10 'b1111011111;      //rega 等于 4 时
    16 'd5:  result = 10 'b1111101111;      //rega 等于 5 时
    16 'd6:  result = 10 'b1111110111;      //rega 等于 6 时
    16 'd7:  result = 10 'b1111111011;      //rega 等于 7 时
    16 'd8:  result = 10 'b1111111101;      //rega 等于 8 时
    16 'd9:  result = 10 'b1111111110;      //rega 等于 9 时
    default: result = 'bx;                  //rega 不等于上面的值时
endcase
```

（4）每一个 case 分项的分支表达式的值必须互不相同，否则就会出现矛盾现象（对表达式的同一个值，有多种执行方案）。

（5）执行完 case 分项后的语句，则跳出该 case 语句结构，终止 case 语句的执行。

（6）在用 case 语句表达式进行比较的过程中，只有当信号的对应位的值能明确进行比较时，比较才能成功，因此要详细说明 case 分项的分支表达式的值。

（7）case 语句的所有表达式的值的位宽必须相等，只有这样控制表达式和分支表达式才能进行对应位的比较。一个经常犯的错误是用'bx、'bz 来替代 n'bx、n'bz，这样写是不对的，因为信号 x、z 的缺省宽度是机器的字节宽度，通常是 32 位（此处 n 是 case 控制表达式的位宽）。

case 语句与 if 语句的区别主要有以下两点。

（1）与 case 语句中的控制表达式和多分支表达式相比，if 结构中的条件表达式更为直观一些。

（2）对于那些分支表达式中存在不定值 x 和高阻值 z 时，case 语句提供了处理这种情况的手段。下面的两个例子介绍了处理 x、z 值 case 语句。

例 3.11：case 语句 1。

开发板免费领取试用 开发板免费领取试用

```
case (select[1:2])
     2 'b00:  result = 0;            //select[1:2]等于00时
     2 'b01:  result = flaga;        //select[1:2]等于01时
     2 'b0x,
     2 'b0z:  result = flaga? 'bx: 0; //select[1:2]等于0x和0z时，执行表达式
     2 'b10:  result = flagb;        //select[1:2]等于10时
     2 'bx0,
     2 'bz0:  result = flagb? 'bx :0; //select[1:2]等于x0和z0时，执行表达式
     default: result = 'bx;          //select[1:2]不等于上面的值时
endcase
```

例 3.12：case 语句 2

```
case(sig)
    1 'bz:    $display("signal is floating"); //sig为高阻时，打印输出
    1 'bx:    $display("signal is unknown");  //sig为不定状态时，打印输出
    default:  $display("signal is %b", sig);  //为其他时，即0或1时，打印输出
endcase
```

针对电路的特性，Verilog HDL 提供了 case 语句的其他两种形式用来处理不必考虑的情况（don't care condition）。其中 casez 语句用来处理不考虑高阻值 z 的比较过程，casex 语句则将高阻值 z 和不定值都视为不必关心的情况。

所谓不必关心的情况，即在表达式进行比较时，不将该位的状态考虑在内。这样在 case 语句表达式进行比较时，就可以灵活地设置，以对信号的某些位进行比较。如表 3.10 所示为 case、casez、casex 的真值表：

表 3.10　　case 语句真值表

case	0	1	x	z
0	1	0	0	0
1	0	1	0	0
x	0	0	1	0
z	0	0	0	1
casez	0	1	x	z
0	1	0	0	1
1	0	1	0	1
x	0	0	1	1
z	1	1	1	1
casex	0	1	x	z
0	1	0	1	1
1	0	1	1	1
x	1	1	1	1
z	1	1	1	1

下面给出两个例子来分别说明 casez 语句和 casex 语句。

例 3.13：casez 语句。

```
reg[7:0] ir;
casez(ir)
    8 'b1???????: instruction1(ir);          //只判断ir的最高位
    8 'b01??????: instruction2(ir);          //只判断ir的高2位
    8 'b00010???: instruction3(ir);          //只判断ir的高5位
```

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

```
        8 'b000001??: instruction4（ir）;          //只判断 ir 的高 6 位
    endcase
```

例 3.14：casex 语句。

```
    reg［7:0］ r, mask;
    mask = 8'bx0x0x0x0;
    casex（r^mask）                                //判断 r^mask 的结果
        8 'b001100xx: stat1;                       //不考虑低 2 位
        8 'b1100xx00: stat2;                       //不考虑第 3、4 位
        8 'b00xx0011: stat3;                       //不考虑第 5、6 位
        8 'bxx001100: stat4;                       //不考虑高 2 位
    Endcase
```

3.5.3 其他条件语句

上面提到的 if 语句和 case 语句都只能应用于 always 语句内部。如果需要在 always 语句之外应用条件语句，可以采样这样的语法结构：

```
    assign data = (sel)?  a : b;
```

上面的语句的含义相当于：

```
    if （sel = 1）
        data = a;
    else
        data =  b;
```

3.6 Verilog HDL 语言的其他常用语句

3.6.1 循环语句

在 Verilog HDL 中存在着 4 种类型的循环语句，用来控制执行语句的执行次数。

（1）forever：连续的执行语句。

（2）repeat：连续执行一条语句 n 次。

（3）while：执行一条语句直到某个条件不满足。如果一开始条件即不满足（为假），则语句一次也不能被执行。

（4）for 通过以下 3 个步骤来决定语句的循环执行。

① 先给控制循环次数的变量赋初值。

② 判定控制循环的表达式的值，如为假则跳出循环语句，如为真则执行指定的语句后，转到步骤③。

③ 执行一条赋值语句来修正控制循环变量次数的变量的值，然后返回步骤②。

下面将详细地对各种循环语句进行介绍。

1．forever 语句

forever 语句的格式如下：

```
    forever 语句;
```


开发板免费领取试用 开发板免费领取试用

或者：

```
forever begin
    多条语句
end
```

forever 循环语句常用于产生周期性的波形，用来作为仿真测试信号。它与 always 语句不同之处在于它不能独立写在程序中，而必须写在 initial 块中。

2．repeat 语句

repeat 语句的格式如下：

```
repeat（表达式） 语句;
```

或者：

```
repeat（表达式） begin
    多条语句
end
```

在 repeat 语句中，其表达式通常为常量表达式。下面的例子中使用 repeat 循环语句及加法和移位操作来实现一个乘法器。

```
parameter size=8,longsize=16;                    //参数声明
reg [size:1] opa, opb;                           //寄存器声明
reg [longsize:1] result;
begin: mult                                      //为 begin_end 模块定名模块名
    reg [longsize:1] shift_opa, shift_opb;       //寄存器声明
    shift_opa = opa;                             //将 opa、opb 的值赋为 shift_opa、shift_opb
    shift_opb = opb;
    result = 0;
    repeat（size） begin                          //循环次数
        if（shift_opb[1])
        result = result + shift_opa;             //加法操作
        shift_opa = shift_opa <<1;               //左移 1 位
        shift_opb = shift_opb >>1;               //右移 1 位
    end
end
```

3．while 语句

while 语句的格式如下：

```
while（表达式） 语句
```

或者：

```
while（表达式） begin
    多条语句
end
```


开发板免费领取试用 开发板免费领取试用

下面举一个 while 语句的例子，该例子用 while 循环语句对 rega 这个 8 位二进制数中值为 1 的位进行计数。

```
begin: count1s
    reg[7:0] tempreg;
    count=0;
    tempreg = rega;
    while(tempreg) begin              //当 tempreg 中有不为 0 的位时，循环执行
        if(tempreg[0])  count = count + 1;    //低位为 1 时，计数
        tempreg = tempreg>>1;                 //否则右移 1 位,此时高位用 0 填补
    end
end
```

4．for 语句

for 语句的一般形式为：

```
for(表达式 1; 表达式 2; 表达式 3)  语句
```

它的执行过程如下。

① 先求解表达式 1。

② 求解表达式 2，若其值为真（非 0），则执行 for 语句中指定的内嵌语句，然后执行步骤③；若为假（0），则结束循环，转到步骤⑤。

③ 若表达式为真，在执行指定的语句后，求解表达式 3。

④ 转到步骤②继续执行。

⑤ 执行 for 语句下面的语句。

for 语句最简单的应用形式是很易理解的，其形式如下：

```
for(循环变量赋初值; 循环结束条件; 循环变量增值)
    执行语句
```

for 循环语句实际上相当于采用 while 循环语句建立以下的循环结构：

```
begin
    循环变量赋初值;
    while(循环结束条件) begin
        执行语句
        循环变量增值;
    end
end
```

这样对于需要 8 条语句才能完成的一个循环控制，for 循环语句只需两条即可。

下面分别举两个使用 for 循环语句的例子。例 3.15 用 for 语句来初始化 memory。例 3.16 则用 for 循环语句来实现前面用 repeat 语句实现的乘法器。

例 3.15：for 语句 1。

```
begin: init_mem
    reg[7:0] tempi;
    for(tempi=0;tempi<memsize;tempi=tempi+1)  //使用 for 循环语句初始化存储器
        memory[tempi]=0;
    end
```

例 3.16：for 语句 2。

开发板免费领取试用 开发板免费领取试用

```
parameter  size = 8, longsize = 16;
reg[size:1] opa, opb;
reg[longsize:1] result;
begin: mult
    integer bindex;
    result=0;
    for( bindex=1; bindex<=size; bindex=bindex+1 ) //使用 for 循环语句实现前面的乘法器
        if(opb[bindex])
            result = result + (opa<<(bindex-1)); //加法并移位
end
```

在 for 语句中，循环变量增值表达式可以不必是一般的常规加法或减法表达式。下面是对 rega 这个 8 位二进制数中值为 1 的位进行计数的另一种方法，如下所示：

```
begin: count1s
    reg[7:0] tempreg;
    count=0;
    for( tempreg=rega; tempreg; tempreg=tempreg>>1 ) //循环变量增值表达式使用右移操作
            if(tempreg[0])
                count=count+1;
end
```

3.6.2 结构说明语句

Verilog 语言中的任何过程模块都从属于以下 4 种结构的说明语句。

（1）initial 说明语句。

（2）always 说明语句。

（3）task 说明语句。

（4）function 说明语句。

initial 和 always 说明语句在仿真的一开始即开始执行。initial 语句只执行一次，always 语句则是不断地重复执行，直到仿真过程结束。在一个模块中，使用 initial 和 always 语句的次数是不受限制的。

task 和 function 语句可以在程序模块中的一处或多处调用，其具体使用方法在第 4 章中详细介绍。这里只对 initial 和 always 语句加以介绍。

1．initial 语句

initial 语句的格式如下：

```
initial begin
    语句 1;
    语句 2;
    ......
    语句 n;
end
```

举例说明。

例 3.17：initial 语句 1。

```
initial begin
```

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

```
        areg=0;                              //初始化寄存器 areg
        for（index=0; index<size; index=index+1）
            memory［index］=0;               //初始化一个 memory
    end
```

在这个例子中用 initial 语句在仿真开始时对各变量进行初始化。

例 3.18：initial 语句 2。

```
    initial begin
        inputs = 'b000000;                  //初始时刻为 0
        #10 inputs = 'b011001;              //赋值时刻为 10
        #10 inputs = 'b011011;              //赋值时刻为 20
        #10 inputs = 'b011000;              //赋值时刻为 30
        #10 inputs = 'b001000;              //赋值时刻为 40
    end
```

从这个例子中，我们可以看到 initial 语句的另一个用途，即用 initial 语句来生成激励波形作为电路的测试仿真信号。一个模块中可以有多个 initial 块，它们都是并行运行的。initial 块常用于测试文件和虚拟模块的编写，用来产生仿真测试信号和设置信号记录等仿真环境。

2．always 语句

always 语句在仿真过程中是不断重复执行的，其声明格式如下：

```
    always <时序控制>  <语句>
```

always 语句由于其不断重复执行的特性，只有和一定的时序控制结合在一起才有用。如果一个 always 语句没有时序控制，则这个 always 语句将会发成一个仿真死锁，例如：

```
    always  areg = ~areg;
```

这个 always 语句将会生成一个 0 延迟的无限循环跳变过程，这时会发生仿真死锁。如果加上时序控制，则这个 always 语句将变为一条非常有用的描述语句，例如：

```
    always #half_period  areg = ~areg;
```

这个例子生成了一个周期为 period（2×half_period） 的无限延续的信号波形，常用这种方法来描述时钟信号，作为激励信号来测试所设计的电路。

```
    reg［7:0］ counter;
    reg tick;
    always @（posedge areg） begin
        tick = ~tick;                  //tick 反相
        counter = counter + 1;         //计数器递增
    end
```

这个例子中，每当 areg 信号的上升沿出现时，把 tick 信号反相，并且把 counter 增加 1。这种时间控制是 always 语句最常用的。

always 的时间控制可以是沿触发也可以是电平触发的，可以单个信号也可以多个信号，中间需要用关键字 or 连接，如：

```
    always @（posedge clock or posedge reset） begin   //由两个沿触发的 always 块
        …
    end
```


开发板免费领取试用 开发板免费领取试用

```
always @( a or b or c ) begin                    //由多个电平触发的 always 块
   …
end
```

沿触发的 always 块常常描述时序逻辑，如果符合，可综合风格要求，用综合工具自动转换为表示时序逻辑的寄存器组和门级逻辑。
电平触发的 always 块常常用来描述组合逻辑和带锁存器的组合逻辑，如果符合，可综合风格要求，转换为表示组合逻辑的门级逻辑或带锁存器的组合逻辑。

3.7 Verilog HDL 语言实现组合逻辑电路

数字逻辑电路分为两种，分别是组合逻辑与时序逻辑。
（1）组合逻辑：输出只是当前输入逻辑电平的函数（有延时），与电路的原始状态无关的逻辑电路。也就是说，当输入信号中的任何一个发生变化时，输出都有可能会根据其变化而变化，但与电路目前所处的状态没有任何关系。
其中组合逻辑是由与、或、非门组成的网络。常用的组合电路有：多路器、数据通路开关、加法器、乘法器等。
（2）时序逻辑：输出不只是当前输入的逻辑电平的函数，还与电路目前所处的状态有关。
时序逻辑由多个触发器和多个组合逻辑块组成的网络，常用的有：计数器、复杂的数据流动控制逻辑、运算控制逻辑、指令分析和操作控制逻辑等。同步时序逻辑是设计复杂的数字逻辑系统的核心。时序逻辑借助于状态寄存器记住它目前所处的状态。在不同的状态下，即使所有的输入都相同，其输出也不一定相同。

3.7.1 assign 语句实现组合逻辑

组合逻辑电路可以用 assign 语句实现，例如：
例 3.19：assign 加法器。

```
wire a,b,c;
assign c = a + b;              //加法器
```

例 3.19 实现的是一个简单的加法器，assign 语句也可以实现复杂一些的组合逻辑电路，例如：
例 3.20：assign 选择器。

```
wire a,b,c;
wire ena;
assign c = ena ? a : b;     //数据选择器
```

例 3.20 实现的是一个数据选择器。如果组合逻辑比较复杂，用 assign 语句书写就会比较繁琐，可读性较差。例如用 assign 语句实现一个 8 选 1 数据选择器，如下所示：
例 3.21：assign 8 选 1 选择器。

```
wire a0,a1,a2,a3,a4,a5,a6,a7,b;
wire [2:0] addr;
assign b =                    //8 选 1 数据选择器
      (addr == 3'd0) ? a0 :(addr == 3'd1) ? a1 :
      (addr == 3'd2) ? a2 :(addr == 3'd3) ? a3 :
      (addr == 3'd4) ? a4 :(addr == 3'd5) ? a5 :
      (addr == 3'd6) ? a6 : a7;
               //在该表达式中，当 addr 不等于 d0～d6 时，b 等于 a7
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

```
                    //当 addr 等于 d6 时，b 等于 a6；当 addr 等于 d5 时，b 等于 a5，且优先级
                    //高于 addr 等于 d6 时的情况，依次类推
```

所以复杂的组合逻辑电路最好用 always 块实现。
从上面的几个例子可以看出，使用 assign 语句描述组合逻辑电路时，格式为：

```
    assign 输出变量 = 输入变量之间的运算结果;
```

3.7.2 always 块实现组合逻辑

组合逻辑电路也可以用 assign 语句实现，例如：
例 3.22：always 加法器。

```
    wire a,b,c;
    always @ (a or b)       //当 a 和 b 有变化时，触发加法器操作
         c = a + b;
```

上面这个例子实现了一个加法器，如果需要实现一个数据选择器，可以书写如下：
例 3.23：always 选择器。

```
    wire a,b,c;
    wire ena;
    always @ (a or b or ena)       //当 a、b 和 ena 有变化时，进行下列操作
        if(ena == 1'b0)      c = b;
        else   c = a;
```

如果想实现一个比较复杂的组合逻辑电路，例如：
例 3.24：always8 选 1 选择器。

```
    wire a0,a1,a2,a3,a4,a5,a6,a7,b;
    wire [2:0] addr;
    always @ (a0 or a1 or a2 or a3 or a4 or a5 or a6 or a7 or addr) begin
         case(addr)            //使用 case 语句实现 8 选 1 数据选择器
              3'd0: b = a0;    //只有当 a0~a7 以及 addr 有变化时，才触发 case 的操作
              3'd1: b = a1;
              3'd2: b = a2;
              3'd3: b = a3;
              3'd4: b = a4;
              3'd5: b = a5;
              3'd6: b = a6;
              3'd7: b = a7;
         endcase
    end
```

由于在 always 块中可以使用 if、case 等语句，所以对于复杂的组合逻辑，使用 always 语句进行描述显得层次更加清楚，可读性更强。
从上面几个例子可以看出，使用 always 语句描述组合逻辑电路时，格式为：

```
    always @ (敏感变量 1 or 敏感变量 2 or 敏感变量 3 or …) begin
         各种语句的组合
    end
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

其中的敏感变量包括所有的会引起输出变化的输入变量以及相应的控制变量。另外，使用 always 语句描述组合逻辑电路时，应该使用阻塞赋值方式，即“=”，而不是“<=”。

3.8 Verilog HDL 语言实现时序逻辑电路

在 Verilog HDL 语言中，时序逻辑电路使用 always 语句块来实现。例如，实现一个带有异步复位信号的 D 触发器如下。

例 3.25：带异步复位的 D 触发器 1。

```
wire Din;
wire clock,rst;
reg Dout;
always @ (posedge clock or negedge rst)       //带有异步复位
    if(rst == 1'b0)   Dout <= 1'b0;
    else Dout   <= Din;                       //D 触发器数据输出
```

在例 3.25 中，每当时钟 clock 上升沿到来后，输出信号 Dout 的值便更新为输入信号 Din 的值。当复位信号下降沿到来时，Dout 的值就会变成 0。

必须注意的是，在时序逻辑电路中，通常使用非阻塞赋值，即使用“<=”。当 always 块整个完成之后，值才会更新，例如：

例 3.26：带异步复位的 D 触发器 2。

```
wire Din;
wire clock,rst;
reg Dout;
always @ (posedge clock or negedge rst)   //带有异步复位
     if(rst == 1'b0)   out <= 1'b0;
     else begin
          Dout <= Din;                     //D 触发器输出值还处于锁定状态
          Dout <= 1'b1;                    //D 触发器输出值依然处于锁定状态
     End                                   //D 触发器的输出为 1
```

在例 3.26 中，Dout 首先被赋值为 Din，此时 Dout 的值并没有发生改变；接着 Dout 又被赋值为 1，此时 Dout 的值依然没发生改变；直到这个 always 模块完成，Dout 的值才变成最后被赋的值，此例中 Dout 的值为 1。

在时序逻辑电路中，always 的时间控制是沿触发的，可以单个信号也可以多个信号，中间需要用关键字“or”连接，例如：

```
always @(posedge clock or posedge reset) begin    //由两个沿触发的 always 块
   …
end
```

其中有一个时钟信号和一个异步复位信号。

```
always @(posedge clock1 or posedge clock2 or posedge reset) begin
//由 3 个沿触发的 always 块
   …
end
```

其中有两个时钟信号和一个异步复位信号。

一般而言，同步时序逻辑电路更稳定，所以建议尽量使用一个时钟触发。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

3.9 Verilog HDL 语言与 C 语言的区别与联系

数字电路设计工程师一般都学习过编程语言、数字逻辑基础、各种 EDA 软件工具的使用。就编程语言而言，国内外大多数学校都以 C 语言为标准，只有少部分学校使用 Pascal 和 Fortran。
算法的描述和验证常用 C 语言来做。例如要设计 Reed-Solomen 编码/解码器，可以分为下面几个步骤。

- 先深入了解 Reed-Solomen 编码/解码的算法。
- 编写 C 语言的程序来验证算法的正确性。
- 运行描述编码器的 C 语言程序，把在数据文件中的多组待编码的数据转换为相应的编码后数据，并存入文件。
- 编写一个加干扰用的 C 语言程序，用于模拟信道。它能产生随机误码位（并把误码位个数控制在纠错能力范围内），将其加入编码后的数据文件中。运行该加干扰程序，产生带误码位的编码后的数据文件。
- 编写一个解码器的 C 语言程序，运行该程序把带误码位的编码文件解码为另一个数据文件。

比较原始数据文件和生成的文件，便可知道编码和解码的程序是否正确（能否自动纠正纠错能力范围内的错码位），用这种方法我们就可以来验证算法的正确性。但这样的数据处理其运行速度只与程序的大小和计算机的运行速度有关，也不能独立于计算机而存在。
如果要设计一个专门的电路来进行这种对速度有要求的实时数据处理，除了以上介绍的 C 程序外，还需编写硬件描述语言（如 Verilog HDL 或 VHDL）的程序。然后进行仿真以便从电路结构上保证算法能在规定的时间内完成，并能与前端和后端的设备或器件正确无误地交换数据。
用硬件描述语言（HDL）的程序设计硬件的好处在于易于理解、易于维护，调试电路速度快，有许多的易于掌握的仿真、综合和布局布线工具，还可以用 C 语言配合 HDL 来做逻辑设计的前后仿真，验证功能是否正确。
在算法硬件电路的研制过程中，计算电路的结构和芯片的工艺对运行速度有很大的影响。所以在电路结构确定之前，必须经过多次仿真。

- C 语言的功能仿真。
- C 语言的并行结构仿真。
- Verilog HDL 的行为仿真。
- Verilog HDL RTL 级仿真。
- 综合后门级结构仿真。
- 布局布线后仿真。
- 电路实现验证。

下面介绍用 C 语言配合 Verilog HDL 来设计算法的硬件电路块时考虑的三个主要问题：

1．为什么选择 C 语言与 Verilog 配合使用

首先，C 语言很灵活，查错功能强，还可以通过 PLI（编程语言接口）编写自己的系统任务直接与硬件仿真器（如 Verilog-XL）结合使用。C 语言是目前世界上应用最为广泛的一种编程语言，因而 C 程序的设计环境比 Verilog HDL 更完整。此外，C 语言可应用于许多领域，有可靠的编译环境，语法完备，缺陷较少。比较起来，Verilog 语言只是针对硬件描述的，在别处使用（如用于算法表达等）并不方便。而且 Verilog 的仿真、综合、查错工具等大部分软件都是商业软件，与 C 语言相比缺乏长期大量的使用，可靠性较差，亦有很多缺陷。所以，只有在 C 语言的配合使用下，Verilog 才能更好地发挥作用。
面对上述问题，最好的方法是 C 语言与 Verilog 语言相辅相成，互相配合使用。这就是既要利用 C 语言的完整性，又要结合 Verilog 对硬件描述的精确性，来更快、更好地设计出符合性能要求的硬件电路系统。
利用 C 语言完善的查错和编译环境，设计者可以先设计出一个功能正确的设计单元，以此作为设计比较的标准。然后，把 C 程序一段一段地改写成用并型结构（类似于 Verilog）描述的 C 程序，此时还是在 C 的环境里，使用的依然是 C 语言。

华清远见
华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

如果运行结果都正确，就将 C 语言关键字用 Verilog 相应的关键字替换，进入 Verilog 的环境。将测试输入同时加到 C 与 Verilog 两个单元，将其输出做比较。这样很容易发现问题的所在，更正后再做测试，直至正确无误。

2．C 语言与 Verilog 语言互相转换中存在的问题

混合语言设计流程往往会在两种语言的转换中会遇到许多难题，如下所示。

- 怎样把 C 程序转换成类似 Verilog 结构的 C 程序。
- 如何增加并行度，以保证用硬件实现时运行速度达到设计要求。
- 怎样不使用 C 中较抽象的语法（例如迭代、指针、不确定次数的循环等）。也能来表示算法（因为转换的目的是用可综合的 Verilog 语句来代替 C 程序中的语句，而可用于综合的 Verilog 语法是相当有限的，往往找不到相应的关键字来替换）。

C 程序是按行依次执行的，属于顺序结构。而 Verilog 描述的硬件是可以在同一时间同时运行的，属于并行结构。这两者之间有很大的冲突。另外，Verilog 的仿真软件也是顺序执行的，在时间关系上同实际的硬件是有差异的，可能会出现一些无法发现的问题。

C 语言的函数调用与 Verilog 中模块的调用也有区别。C 程序调用函数是没有延时特性的，一个函数是惟一确定的，对同一个函数的不同调用是一样的。而 Verilog 中对模块的不同调用是不同的，即使调用的是同一个模块，必须用不同的名字来指定。

Verilog 的语法规则很死，限制很多，能用的判断语句有限，仿真速度较慢，查错功能差，错误信息不完整。仿真软件通常也很昂贵，而且不一定可靠。C 语言的花样则很多，转换过程中会遇到一些困难。

C 语言没有时间关系，转换后的 Verilog 程序必须要能做到没有任何外加的人工延时信号，否则将无法使用综合工具把 Verilog 源代码转化为门级逻辑。

3．如何利用 C 语言来加快硬件的设计和查错

如表 3.11 所示为常用的 C 语言与 Verilog 相对应的关键字与控制结构。

表 3.11　　C 语言与 Verilog 相对应的关键字与控制结构表

C	Verilog
sub-function	module、function、task
if-then-else	if-then-else
case	case
{,}	begin、end
for	for
while	while
break	disable
define	define
int	int
printf	monitor、display、strobe

如表 3.12 所示为 C 语言与 Verilog 相对应的运算符。

表 3.12　　C 语言与 Verilog 对应运算符表

C	Verilog	功　能
*	*	乘
/	/	除

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

+	+	加
-	-	减
%	%	取模
!	!	反逻辑
&&	&&	逻辑与
‖	‖	逻辑或
>	>	大于
<	<	小于

续表

C	Verilog	功　能
>=	>=	大于等于
<=	<=	小于等于
==	==	等于
!=	!=	不等于
～	～	位反相
&	&	按位逻辑与
|	|	按位逻辑或
^	^	按位逻辑异或
～^	～^	按位逻辑同或
>>	>>	右移
<<	<<	左移
?:	?:	相当于 if-else

从上面的讲解我们可以总结如下。

- C 语言与 Verilog 硬件描述语言可以配合使用，辅助设计硬件。
- C 语言与 Verilog 硬件描述语言很像，但要稍加限制。
- C 语言的程序很容易转成 Verilog 的程序。

3.10　Verilog HDL 程序设计经验

对于 Verilog HDL 的初学者，经常会对语法中的几个容易混淆的地方产生困惑。下面列出几个常见问题和解决它们的小窍门。

1．“=”和“<=”的区分方法

前面的内容已经从原理上解释了阻塞（=）和非阻塞（<=）赋值的区别，但对于初学者来说，在实际应用过程中还会产生一些困惑。下面的方法可以帮助初学者来弄清楚两种赋值符号的应用场合。

在 always 语句中，所有的赋值符号用非阻塞的，即“<=”；　在 always 语句外，所有的赋值符号用阻塞的，即“=”。

2．“reg”和“wire”的区分方法

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

reg 类型和 wire 类型是 Verilog HDL 语法中两种最常用的变量。在对 module 定义的端口信号进行类型描述的时候，初学者会对何时需要指定为 reg 型感到困惑。可以参考下面的方法。

（1）如果这个信号需要在 always 块里面被赋值，那么必须指定为 reg 类型的。

（2）如果这个信号需要在 always 块外面被赋值，那么必须指定为 wire 类型的。如果这个信号是端口信号，那么没默认的类型就是 wire 类型的，不需要另外指定。

3.11 典型实例 3：数字跑表

3.11.1 实例的内容及目标

1．实例的主要内容

本节通过 Verilog HDL 语言编写一个具有“百分秒、秒、分”计时功能的数字跑表，可以实现一个小时以内精确至百分之一秒的计时。

数字跑表的显示可以通过编写数码管显示程序来实现，本实例只给出数字跑表的实现过程。读者还可以通过增加小时的计时功能，实现完整的跑表功能。

2．实例目标

本实例主要实现了计数及进位的设计，通过几个 always 模块的设计实现一个特定用途的模块——数字跑表。通过本实例，读者应达到下面的一些实例目标。

- 初步掌握 Verilog 语言的设计方法。
- 完成一个数字跑表的设计。

3.11.2 原理简介

本数字跑表首先要从最低位的百分秒计数器开始，按照系统时钟进行计数。计数至 100 后向秒计数器进位，秒计数器以百分秒计数器的进位位为时钟进行计数。计数至 60 后向分计数器进位，分计数器以秒计数器的进位位为时钟进行计数，读者可以自行增加小时计数器。

数字跑表巧妙地运用进位位作为计数时钟来减少计数的位数。如果统一使用系统时钟作为计数时钟，那秒计数器将是一个 6000 进制的计数器，而分计数器将是一个 3600000 进制的计数器。这样将极大的浪费 FPGA 的逻辑资源。而使用进位位作为计数时钟，只需要一个 100 进制的计数器和两个 60 进制的计数器。

如图 3.3 是本实例的数字跑表模块图。

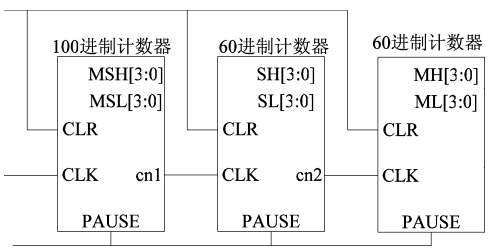


图 3.3 数字跑表模块图

在实际的设计中，为了使计数器更加简单，计数器使用高低位两个计数器实现。100 进制计数器分别是高位 10 进制计数器，低位 10 进制计数器；60 进制计数分别是高位 6 进制计数器，低位 10 进制计数器。这样整个数字跑表使用 6 个计数器实现。

同时由于 10 进制计数器重复使用了 5 次，可以使用独立的模块实现 10 进制计数器，这样就可以通过模块复用来节省整个模块使用的资源。

数字跑表提供了清零位 CLR 和暂停位 PAUSE，百分秒的时钟信号可以通过系统时钟分频提供。分频至 1/100s，即可实现真实的时间计数。详细的时钟分频设计读者可参考相关的资料实现，在本实例中不再提供。

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

3.11.3 代码分析

下面给出这个数字跑表的源代码，读者可以将这些源代码嵌入自己的工程设计中，来实现数字跑表的功能。首先给出代码中端口信号的定义，读者可根据这些端口与自己的工程设计进行连接。

- CLK：时钟信号。
- CLR：异步复位信号。
- PAUSE：暂停信号。
- MSH、MSL：百分秒的高位和低位。
- SH、SL：秒信号的高位和低位。
- MH、ML：分钟信号的高位和低位。

下面是数字跑表的 Verilog HDL 源代码及说明。

```
module paobiao(CLK,CLR,PAUSE,MSH,MSL,SH,SL,MH,ML);
     //端口说明
     input CLK,CLR;
     input PAUSE;
     output[3:0] MSH,MSL,SH,SL,MH,ML;
     //内部信号说明
     reg[3:0] MSH,MSL,SH,SL,MH,ML;
     reg cn1,cn2;                    //cn1为百分秒向秒的进位，cn2为秒向分的进位

     //百分秒计数模块，每计满100，cn1产生一个进位
     always @(posedge CLK or posedge CLR) begin
         if(CLR) begin               //异步复位
           {MSH,MSL}<=8'h00;
            cn1<=0;
          end
          else if(!PAUSE) begin      //PAUSE为0时正常计数，为1时暂停计数
               if(MSL==9) begin
                    MSL<=0;          //低位计数至10时，低位归零
                    if(MSH==9) begin
                         MSH<=0;     //低、高位计数至10时，高位归零
                         cn1<=1;     //低、高位计数至10时，触发进位位
                    end
                    else             //低位计数至10，高位计数未至10时，高位计数
                         MSH<=MSH+1;
               end
               else begin
                    MSL<=MSL+1;      //低位计数未至10时，低位计数
                    cn1<=0;          //低位计数未至10时，不触发进位位
               end
          end
     end

     //秒计数模块，每计满60，cn2产生一个进位
     always @(posedge cn1 or posedge CLR) begin
          if(CLR) begin              //异步复位
```


开发板免费领取试用 开发板免费领取试用

```
                {SH,SL}<=8'h00;
                cn2<=0;
            end
            else if(SL==9) begin
                SL<=0;              //低位计数至 10 时，低位归零
                if(SH==5) begin
                    SH<=0;          //低位计数至 10，高位计数至 6 时，高位归零
                    cn2<=1;         //低位计数至 10，高位计数至 6 时，触发进位位
                end
                else
                    SH<=SH+1;       //低位计数至 10，高位计未数至 6 时，高位计数
            end
            else begin
                SL<=SL+1;           //低位计数未至 10 时，低位计数
                cn2<=0;             //低位计数未至 10 时，不触发进位位
            end
        end

        //分钟计数模块，每计满 60，系统自动清零
        always @(posedge cn2 or posedge CLR) begin
            if(CLR) begin           //异步复位
                {MH,ML}<=8'h00;
            end
            else if(ML==9) begin
                ML<=0;              //低位计数至 10 时，低位归零
                if(MH==5)
                    MH<=0;          //低位计数至 10，高位计数至 6 时，高位归零
                else
                    MH<=MH+1;       //低位计数至 10，高位计未数至 6 时，高位计数
            end
            else
                ML<=ML+1;           //低位计数未至 10 时，低位计数
            end

    endmodule
```

通过上面的这 3 个模块，即可实现数字跑表的功能。

3.11.4 参考设计

本实例相关参考设计文件在本书实例代码的“典型实例 3”文件夹。

3.12 典型实例 4：PS/2 接口控制

3.12.1 实例的内容及目标

开发板免费领取试用 开发板免费领取试用

1．实例的主要内容

本实例通过 Verilog 编程实现在红色飓风 II 代 Xilinx 开发板上面实现对键盘、LCD、RS-232 等接口或者器件进行控制，将有键盘输入的数据在 LCD 上面显示出来，或者通过 RS-232 在 PC 机上的超级终端上显示出来。

2．实例目标

通过本实例，读者应达到下面的目标。

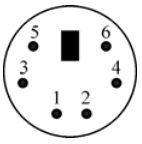
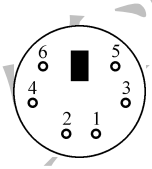
- 了解 PS/2 接口协议。
- 掌握键盘的工作原理。
- 编写 Verilog 程序实现通过开发板上 PS/2 接口读取键盘的输入信息。

3.12.2 原理简介

PS/2 键盘履行一种双向同步串行协议。换句话说每次数据线上发送一位数据，并且每在时钟线上发一个脉冲就被读入，键盘可以发送数据到主机，而主机也可以发送数据到设备。但主机总是在总线上有优先权，它可以在任何时候抑制来自于键盘的通信，只要把时钟拉低即可。

本实例要编写一个能实现 PS/2 端口功能的 Verilog 程序。首先我们要了解 PS/2 端口的结构与管脚功能定义，如表 3.13 所示。

表 3.13　　PS/2 端口结构及管脚定义

端口结构		管脚定义	
		1	数据
		2	未实现，保留
		3	电源地
		4	电源，+5V
		5	时钟
插头	插座	6	未实现，保留

可以看到，PS/2 里面只有一个数据口，若要分辨很多按键就需要一个高效率的分辨方法。键盘的处理器花费很多的时间来扫描或监视按键矩阵。如果它发现有键被按下，释放或按住键盘，将发送扫描码的信息包到计算机。

扫描码有两种不同的类型：“通码”和“断码”。当一个键被按下或按住就发送“通码”，当一个键被释放就发送“断码”。每个按键被分配了惟一的“通码”和“断码”。这样，主机通过查找惟一的扫描码就可以测定是哪个按键。

每个键一整套的通断码组成了扫描码集，图 3.4 中包含了键盘上面大部分按键的扫描码。

当按键被释放以后，键盘回在扫描码前面加上一个“F0”作为按键松开信号。同时有的按键是 Extended（扩展）键，此时要在它们的扫描码前面加上一个“E0”作为开头，这种按键松开以后将在扫描码前面附加上“E0F0”。

下面我们来了解信号是如何从键盘输入通过 PS/2 端口的数据线输入的。首先键盘要检测数据线和时钟线是否都为高，只有它们都处在高的状态才可以写数据。从键盘发送到主机的数据在时钟信号的下降沿（当时钟从高变到低）的时候被读取。

键盘主要使用一种每帧包含 11 位的串行协议：第一位是起始位，始终为“0”；接下来是 8 位数据位，排列顺序是由低到高；再后面是奇偶校验位；最后是结束位，始终为“1”。如图 3.5 所示为该协议的时序图。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

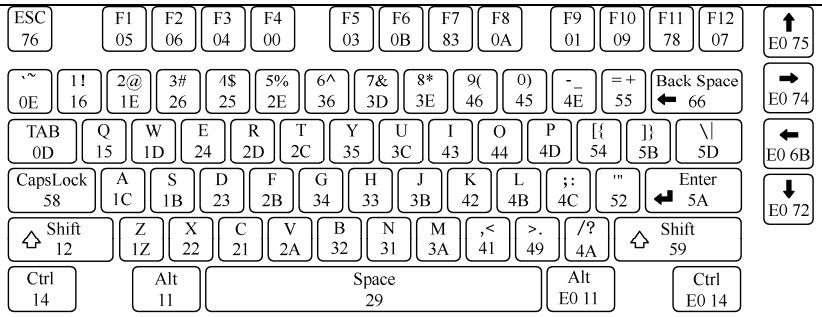


图 3.4　键盘扫描码

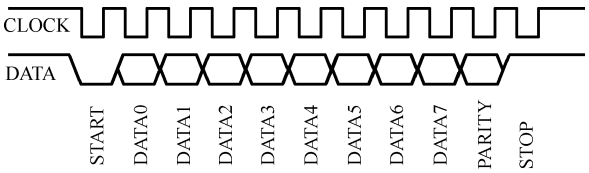


图 3.5　键盘串行协议

3.12.3　实例详解

本实例的具体的实验步骤可参见 2.6 节，在此不再详述，仅给出主要的操作流程。

（1）启动 ISE 软件。

（2）创建新工程。

（3）编写键盘串行协议。

按照图 3.5 的时序编写相应的 Verilog 代码，具体源代码参见实例代码。

（4）添加设计输入。

将编写好的接口协议加载至工程中，完成接口控制与系统的对接。

（5）设置器件及管脚约束。

按照开发板的说明进行相关的设置。

（6）下载验证。

下载程序后，将键盘接至开发板上，通过开发板的 LCD 可以看到键盘输入的字符。

3.12.4　参考设计

本实例相关参考设计文件在本书实例代码的典型实例 4 文件夹。

3.13　典型实例 5：交通灯控制器

3.13.1　实例的内容及目标

1．实例的主要训练内容

本实例通过 Verilog HDL 语言设计一个简易的交通等控制器，实现一个具有两个方向、共 8 个灯并具有时间倒计时功能的交通灯功能。

2．实例目标

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

通过本实例，读者应达到下面的目标。

- 掌握 Verilog 设计一个交通等控制器的方法。
- 初步掌握 Verilog 语言的设计方法。

3.13.2 原理简介

交通灯是城市交通中不可缺少的重要工具，是城市交通秩序的重要保障。本实例就是实现一个常见的十字路口交通灯功能。读者通过学习这个交通灯控制器，可以实现一个更加完整的交通灯。例如实现实时配置各种灯的时间，手动控制各个灯的状态等。

一个十字路口的交通一般分为两个方向，每个方向具有红灯、绿灯和黄灯 3 种，另外每个方向还具有左转灯，因此每个方向具有 4 个灯。

这个交通灯还为每一个灯的状态设计了倒计时数码管显示功能。可以为每一个灯的状态设置一个初始值，灯状态改变后，开始按照这个初始值倒计时。倒计时归零后，灯的状态将会改变至下一个状态。

值得注意的是，交通灯两个方向的灯的状态是相关的。也就是说，每个方向的灯的状态影响着另外一个方向的灯的状态，这样才能够协调两个方向的车流。如果每个方向的灯是独立变化的，那么交通灯就没有了意义。

如表 3.14 所示是两个方向（假设为 A，B 方向）灯的状态的对应情况。

表 3.14 交通灯两个方向灯状态对应表

方向 A	方向 B
红灯亮	黄灯亮或绿灯亮
直行绿灯亮	红灯亮
黄灯亮	红灯亮
左转灯	红灯亮

在实际的交通系统中，直行绿灯、左转绿灯和红灯的变化之间都应该有黄灯作为缓冲，以保证交通的安全。因此假如我们假设方向 A 的黄灯亮的时间持续 5s，直行绿灯灯亮的时间持续 40s，左转灯灯亮的时间持续 15s，则方向 B 红灯灯亮的时间持续为（直行绿灯+黄灯+左转绿灯+黄灯）所消耗的时间，一共为 65s。

同样假设方向 B 黄灯亮的时间持续 5s，直行绿灯灯亮的时间持续 30s，左转灯灯亮的时间持续 15s，则方向 B 红灯灯亮的时间持续为（直行绿灯+黄灯+左转绿灯+黄灯）所消耗的时间，一共为 55s。

具体时间参数的设定读者可以根据需要进行修改，但是一定要保证两个方向的灯的状态符合表 3.14 的要求。

3.13.3 代码分析

下面给出交通灯控制器的 Verilog HDL 源代码，首先介绍交通灯端口信号的定义及说明，读者可以通过这些端口将此交通灯模块实例化至自己的工程设计中。

- CLK：同步时钟。
- EN：使能信号，为高电平时，控制器开始工作。
- LAMPA：控制 A 方向 4 盏灯的状态；其中，LAMPA0～LAMPA3 分别控制 A 方向的左拐灯、绿灯、黄灯和红灯。
- LAMPB：控制 B 方向 4 盏灯的状态；其中，LAMPB0～LAMPB3 分别控制 B 方向的左拐灯、绿灯、黄灯和红灯。
- ACOUNT：用于 A 方向灯的时间显示，8 位，可驱动两个数码管。
- BCOUNT：用于 B 方向灯的时间显示，8 位，可驱动两个数码管。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

下面是交通灯的 Verilog HDL 源代码及说明。

```
module traffic(CLK,EN,LAMPA,LAMPB,ACOUNT,BCOUNT);
    //端口说明
    output［7:0］ ACOUNT,BCOUNT;
    output［3:0］ LAMPA,LAMPB;
    input CLK,EN;
    //内部信号说明
    reg［7:0］ numa,numb;           //ACOUNT 和 BCOUNT 的内部信号
    reg tempa,tempb;
    reg［2:0］ counta,countb;       //方向 A 和方向 B 的灯的状态
    reg［7:0］ ared,ayellow,agreen,aleft,bred,byellow,bgreen,bleft;
    reg［3:0］ LAMPA,LAMPB;

    //设置各交通灯的持续时间初始化值，红灯的值由另一个方向的黄灯和绿灯计算得出。
    always @(EN)
        if(!EN) begin               //使能信号 EN 无效时，对交通灯的计数值进行初始化
            ared       <=8'd55;     //55 s , 30 + 5 + 15 + 5
            ayellow    <=8'd5;      //5 s
            agreen     <=8'd40;     //40 s
            aleft      <=8'd15;     //15 s
            bred       <=8'd65;     //65 s , 40 + 5 + 15 + 5
            byellow    <=8'd5;      //5 s
            bleft      <=8'd15;     //15 s
            bgreen     <=8'd30;     //30 s
        end
        assign ACOUNT=numa;                    //8 位数码管输出
    assign BCOUNT=numb;                //8 位数码管输出
        //控制 A 方向 4 种灯的模块
    always @(posedge CLK) begin
        if(EN) begin                   //使能有效时，交通灯开始工作
            if(!tempa) begin
                tempa<=1;
                case(counta)           //控制灯状态的顺序
                    0: begin           //状态 0
                        numa<=agreen; //直行绿灯亮
                        LAMPA<=2;          //输出 0010
                        counta<=1;         //进入下一个状态
                    end
                    1: begin               //状态 1
                        numa<=ayellow;     //黄灯亮
                        LAMPA<=4;          //输出 0100
                        counta<=2;         //进入下一个状态
                    end
                    2: begin               //状态 2
                        numa<=aleft;       //左转绿灯亮
                        LAMPA<=1;          //输出 0001
                        counta<=3;         //进入下一个状态
```


开发板免费领取试用 开发板免费领取试用

```
                end
                3: begin                //状态 3
                    numa<=ayellow;      //黄灯亮
                    LAMPA<=4;           //输出 0100
                    counta<=4;          //进入下一个状态
                end
                4: begin                //状态 4
                    numa<=ared;         //红灯亮
                    LAMPA<=8;           //输出 1000
                    counta<=0;          //进入下一个状态（状态 0）
                end
                default:                //默认状态
                    LAMPA<=8;           //红灯亮，输出 1000
            endcase
        end
        else begin                  //每一个状态的倒计时
           if(numa>1)               //判断倒计时未归零时分别对高地位进行递减
                if(numa[3:0]==0) begin
                    numa[3:0]<=4'b1001;
                    numa[7:4]<=numa[7:4]-1;
                end
                else
                    numa[3:0]<=numa[3:0]-1;
            if (numa==2)
                tempa<=0;   //倒计时结束，返回灯状态变化判断，将进入下一个状态
        end
    end
    else begin
        LAMPA<=4'b1000;         //使能无效时，红灯亮
        counta<=0;              //返回方向 A 的状态 0(绿灯状态)
        tempa<=0;               //进入状态变化判断
    end
end
   //控制 B 方向 4 种灯的模块，模块的语言描述与方向 A 的描述基本一致，这里不再重复注释
always @(posedge CLK) begin
    if (EN) begin
        if(!tempb) begin
            tempb<=1;
            case (countb)
                0: begin
                    numb<=bred;
                    LAMPB<=8;
                    countb<=1;
                end
                1: begin
                    numb<=bgreen;
                    LAMPB<=2;
```


开发板免费领取试用 开发板免费领取试用

```
                    countb<=2;
                end
                2: begin
                    numb<=byellow;
                    LAMPB<=4;
                    countb<=3;
                end
                3: begin
                    numb<=bleft;
                    LAMPB<=1;
                    countb<=4;
                end
                4: begin
                    numb<=byellow;
                    LAMPB<=4;
                    countb<=0;
                end
                default:
                    LAMPB<=8;
            endcase
        end
        else begin //倒计时
            if(numb>1)
                if(!numb[3:0]) begin
                    numb[3:0]<=9;
                    numb[7:4]<=numb[7:4]-1;
                end
            else
                numb[3:0]<=numb[3:0]-1;
            if(numb==2)
                tempb<=0;
        end
    end
    else begin
        LAMPB<=4'b1000;
        tempb<=0;
        countb<=0;
    end
  end
endmodule
```

通过上面这个 Verilog HDL 模块，基本实现了交通灯控制器的基本功能。读者可将此设计应用于实际的硬件系统中，通过晶振、FPGA、开关、LED 灯及数码管等资源即可完成硬件实现。

3.13.4 参考设计

本实例相关参考设计文件参见实例代码的“典型实例 5”文件夹。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

联系方式

集团官网：www.hqyj.com　　嵌入式学院：www.embedu.org　　移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn　　物联网学院：www.topsight.cn　　研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路银海大厦 A 座 8 层，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218

华清远见

开发板免费领取试用　开发板免费领取试用

华清远见 FARSIGHT 始于2004 | 嵌入式培训专家

10年口碑积累，成功培养50000多名研发工程师，铸就专业品牌形象

华清远见的企业理念是不仅要做良心教育、做专业教育，更要做受人尊敬的职业教育。

《FPGA 应用开发入门与典型实例》（修订版）

作者：华清远见

专业始于专注 卓识源于远见

第 4 章　硬件描述语言 Verilog HDL 设计进阶

本章目标

- 掌握 Verilog HDL 语法里面 task 和 function 语句的使用方法
- 掌握有限状态机（FSM）的设计方法和技巧
- 熟悉复杂的 FPGA 工程设计流程

专业始于专注 卓识源于远见

开发板免费领取试用 开发板免费领取试用

4.1 task 和 function 说明语句的区别

task 和 function 说明语句分别用来定义任务和函数。利用任务和函数可以把一个很大的程序模块分解成许多较小的任务和函数便于理解和调试。输入、输出和总线信号的值可以传入或传出任务和函数。
任务和函数往往是大的程序模块中在不同地点多次用到的相同的程序段。学会使用 task 和 function 语句可以简化程序的结构，使程序明白易懂，是编写大型模块的基本功。
任务和函数有些不同，主要的不同有以下 4 点。
（1）函数只能与主模块共用同一个仿真时间单位，而任务可以定义自己的仿真时间单位。
（2）函数不能启动任务，而任务能启动其他任务和函数。
（3）函数至少要有一个输入变量，而任务可以没有或有多个任何类型的变量。
（4）函数返回一个值，而任务则不返回值。
函数的目的是通过返回一个值来响应输入信号的值。任务却能支持多种目的，能计算多个结果值，这些结果值只能通过被调用的任务的输出或总线端口送出。Verilog HDL 模块使用函数时是把它当作表达式中的操作符，这个操作的结果值就是这个函数的返回值。下面让我们用例子来说明。
例如，定义一任务或函数对一个 16 位的字进行操作让高字节与低字节互换，把它变为另一个字(假定这个任务或函数名为：switch_bytes)。
任务返回的新字是通过输出端口的变量，16 位字的字节互换任务的调用源码如下：

```
switch_bytes(old_word,new_word);
```

任务 switch_bytes 把输入 old_word 的字的高、低字节互换放入 new_word 端口输出。
而函数返回的新字是通过函数本身的返回值，16 位字的字节互换函数的调用源码如下：

```
new_word = switch_bytes(old_word);
```

下面分别介绍任务和函数语句的要点。

4.2 Verilog HDL 高级语法结构——任务（TASK）

如果传给任务的变量值和任务完成后接收结果的变量已定义，就可以用一条语句启动任务。任务完成以后控制就传回启动过程。如任务内部有定时控制，则启动的时间可以与控制返回的时间不同。
任务可以启动其他的任务，其他任务又可以启动别的任务，可以启动的任务数是没有限制的。不管有多少任务启动，只有当所有的启动任务完成以后，控制才能返回。

1．任务定义

定义任务的语法如下：

```
任务:
task <任务名>;
     <端口及数据类型声明语句>
     <语句 1>
     <语句 2>
     ...
     <语句 n>
endtask
```

这些声明语句的语法与模块定义中的对应声明语句的语法是一致的。

开发板免费领取试用 开发板免费领取试用

2．任务调用及变量传递

调用任务并传递输入/输出变量的声明语句的语法如下：

```
<任务名>(端口 1，端口 2，...，端口 n);
```

下面的例子说明怎样定义任务和调用任务。
任务定义如下所示：

```
task  my_task;
    input a, b;
    inout  c;
    output d, e;
    …
    <语句>                //执行任务工作相应的语句
    …
    c = foo1;           //赋初始值
    d = foo2;           //对任务的输出变量赋值
    e = foo3;
endtask
```

任务调用如下所示：

```
my_task(v,w,x,y,z);
```

任务调用变量(v、w、x、y、z)和任务定义的 I/O 变量(a、b、c、d、e)之间是一一对应的。当任务启动时，由 v、w 和 x 传入的变量赋给了 a、b 和 c。当任务完成后的输出又通过 c、d 和 e 赋给了 x、y 和 z。下面用一个具体的例子用来说明怎样在模块的设计中使用任务，使程序容易读懂。

```
module traffic_lights;
     reg  clock, red, amber, green;
     parameter  on=1, off=0, red_tics=350,
     amber_tics=30,green_tics=200;
     initial    red=off;                          //交通灯初始化
     initial    amber=off;
     initial    green=off;

     always begin                                 //交通灯控制时序
         red=on;                                  //开红灯
         light(red,red_tics);                     //调用等待任务
         green=on;                                //开绿灯
         light(green,green_tics);                 //等待
         amber=on;                                //开黄灯
         light(amber,amber_tics);                 //等待
     end

     task  light（color,tics）;                   //定义交通灯开启时间的任务
         output  color;
         input［31:0］ tics;
         begin
             repeat(tics) @(posedge clock);       //等待 tics 个时钟的上升沿
             color=off;                           //关灯
         end
     endtask

     always begin                                 //产生时钟脉冲的 always 块
          #100 clock=0;
          #100 clock=1;
     end
endmodule
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

这个例子描述了一个简单的交通灯的时序控制，并且该交通灯有它自己的时钟产生器。

4.3 Verilog HDL 高级语法结构——任务（function）

函数的目的是返回一个用于表达式的值。

1．函数定义语法

```
function <返回值的类型或范围> (函数名);
    <端口说明语句>
    <变量类型说明语句> begin
    <语句>
    ...
    end
endfunction
```

请注意<返回值的类型或范围>这一项是可选项，如缺省则返回值为一位寄存器类型数据。下面用例子说明：

```
function [7:0] getbyte;
    input [15:0] address;
    begin
        <说明语句>                          //从地址字中提取低字节的程序
        getbyte = result_expression;       //把结果赋予函数的返回字节
    end
endfunction
```

2．函数返回值

函数的定义蕴含声明了与函数同名的、函数内部的寄存器。如在函数的声明语句中<返回值的类型或范围>为缺省,则这个寄存器是一位的；否则是与函数定义中<返回值的类型或范围>一致的寄存器。
函数的定义把函数返回值所赋值寄存器的名称初始化为与函数同名的内部变量。上面的例子说明了这个概念：getbyte 被赋予的值就是函数的返回值。

3．函数调用

函数的调用是通过将函数作为表达式中的操作数来实现的，其调用格式如下：

```
<函数名> (<表达式><,<表达式>>*)
```

其中函数名作为确认符。下面的例子中通过对两次调用函数 getbyte 的结果值进行位拼接运算来生成一个字。

```
word = control? {getbyte(msbyte),getbyte(lsbyte)} : 0;
```

4．函数使用规则

开发板免费领取试用 开发板免费领取试用

与任务相比函数的使用有较多的约束，下面给出的是函数的使用规则。

- 函数的定义不能包含有任何的时间控制语句，即任何用 # 、@、或 wait 来标识的语句。
- 函数不能启动任务。
- 定义函数时至少要有一个输入参量。
- 在函数的定义中必须有一条赋值语句给函数中的一个内部变量赋以函数的结果值，该内部变量具有和函数名相同的名字。

5．实例详解

下面的例子中定义了一个可进行阶乘运算的名为factorial的函数，该函数返回一个32位的寄存器类型的值，该函数可后向调用自身，并且打印出部分结果值。

```
module  tryfact;                                    //模块开始
//函数的定义------------------------------
function [31:0] factorial;                          //函数开始
     input [3:0] operand;                           //函数的入口
     reg [3:0] index;                               //函数的寄存器
     begin
         factorial = operand? 1 : 0;                //若 operand 全为 0，则为 0
         for(index=2;index<=operand;index=index+1) //循环语句
              factorial = index * factorial;        //乘法器
     end
endfunction                                         //函数结束
//函数的测试--------------------------------------
reg [31:0] result;
reg [3:0] n;
initial begin
     result=1;
     for(n=2;n<=9;n=n+1) begin                      //循环语句（调用函数）
         $display("Partial result n= %d result= %d", n, result);  //每次函数的执行结果
         result = n * factorial(n)/((n*2)+1);       //调用函数
     end
     $display("Finalresult=%d",result);             //打印结果
end
endmodule                                           //模块结束
```

4.4　有限状态机的设计原理及其代码风格

由于 Verilog HDL 和 VHDL 行为描述用于综合的历史还只有短短的几年，可综合风格的 Verilog HDL 和 VHDL 的语法只是它们各自语言的一个子集。又由于 HDL 的可综合性研究近年来非常活跃，可综合子集的国际标准目前尚未最后形成，因此各厂商的综合器所支持的 HDL 子集也略有所不同。

本书中有关可综合风格的 Verilog HDL 的内容，我们只着重介绍 RTL 级、算法级和门级逻辑结构的描述，而系统级（数据流级）的综合由于还不太成熟，暂不作介绍。

由于寄存器传输级（RTL）描述是以时序逻辑抽象所得到的有限状态机为依据的，所以把一个时序逻辑抽象成一个同步有限状态机是设计可综合风格的 Verilog HDL 模块的关键。

在本章中我们将通过各种实例由浅入深地来介绍各种可综合风格的 Verilog HDL 模块，并把重点放在时序

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

逻辑的可综合有限状态机的 Verilog HDL 设计要点。至于组合逻辑，因为比较简单，只需阅读典型的用 Verilog HDL 描述的可综合的组合逻辑的例子就可以掌握。

为了更好地掌握可综合风格，还需要较深入地了解阻塞和非阻塞赋值的差别和在不同的情况下正确使用这两种赋值的方法。只有深入地理解阻塞和非阻塞赋值语句的细微不同，才有可能写出不仅可以仿真也可以综合的 Verilog HDL 模块。

只要按照一定的原则来编写代码就可以保证 Verilog 模块综合前和综合后仿真的一致性。符合这样条件的可综合模块是我们设计的目标，因为这种代码是可移植的，可综合到不同的 FPGA 和不同工艺的 ASIC 中，是具有知识产权价值的软核。

4.4.1 有限状态机（FSM）设计原理

有限状态机是由寄存器组和组合逻辑构成的硬件时序电路。有限状态机的状态（即由寄存器组的 1 和 0 的组合状态所构成的有限个状态）只可能在同一时钟跳变沿的情况下才能从一个状态转向另一个状态。

有限状态机的下一个状态不但取决于各个输入值，还取决于当前所在状态。这里指的是米里 Mealy 型有限状态机，而莫尔 Moore 型有限状态机的下一个状态只决于当前状态。

Verilog HDL 中可以用许多种方法来描述有限状态机，最常用的方法是用 always 语句和 case 语句。如图 4.1 所示的状态转移图表示了一个有限状态机，例 4.1 的程序就是该有限状态机的多种 Verilog HDL 模型之一。

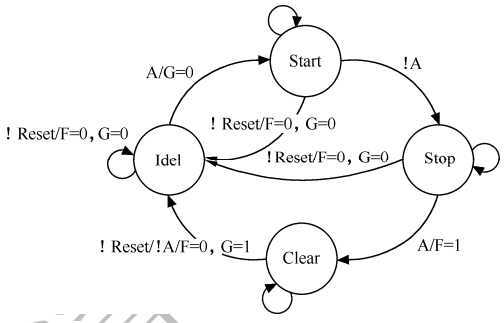


图 4.1 四状态有限状态机

图 4.1 的状态转移图表示了一个四状态的有限状态机。它的同步时钟是 Clock，输入信号是 A 和 Reset，输出信号是 F 和 G。

状态的转移只能在同步时钟（Clock）的上升沿时发生，往哪个状态的转移则取决于目前所在的状态和输入的信号（Reset 和 A）。下面的例子是该有限状态机的 Verilog HDL 模型之一。

例 4.1：Gray 码有限状态机模型 1。

```
    module  fsm (Clock, Reset, A, F, G);          //模块声明
        input Clock, Reset, A;
        output F,G;
        reg F,G;
        reg [1:0] state ;

        parameter                                 //状态声明
             Idle  = 2'b00,  Start = 2'b01,
             Stop = 2'b10, Clear = 2'b11;

        always @(posedge Clock)
             if (!Reset) begin
                 state <= Idle;  F<=0; G<=0;      //默认状态
             end
             else case (state)
                  idle:  begin                    //Idle 状态
                       if (A)  begin
                            state <= Start;
                            G<=0;
                       end
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

```
                    elsestate <= idle;
                end
                start:                          //Start 状态
                    if (!A)  state <= Stop;
                    else    state <= start;
                Stop:  begin                    //Stop 状态
                    if (A) begin
                         state <= Clear;
                         F <= 1;
                    end
                    else   state <= Stop;
                    end
                Clear: begin                     //Clear 状态
                     if (!A)  begin
                         state <=Idle;
                         F <=0; G <=1;
                     end
                else   state <= Clear;
                end
            endcase
    endmodule
```

也可以用下面的 Verilog HDL 模型来表示同一个有限状态。

例 4.2：独热码有限状态和模型。

```
    module  fsm (Clock, Reset, A,  F, G);          //模块声明
         input Clock, Reset, A;
         output F,G;
         reg F,G;
         reg [3:0] state ;

         parameter                                 //状态声明
             Idle    = 4'b1000,
             Start   = 4'b0100,
             Stop    = 4'b0010,
             Clear   = 4'b0001;

         always @(posedge clock)
             if (!Reset) begin
                  state <= Idle;  F<=0; G<=0;      //默认状态
             end
             else case (state)
                  Idle:  begin                    //Idel 状态
                      if (A)  begin
                           state <= Start;
                           G<=0;
                           end
                      else state <= Idle;
```


开发板免费领取试用 开发板免费领取试用

```
                end
                Start:                          //Start 状态
                     if (!A)  state <= Stop;
                          else  state <= Start;
                Stop:  begin                    //Stop 状态
                     if (A) begin
                               state <= Clear;
                               F <= 1;
                          end
                     else  state <= Stop;
                end
                Clear: begin                    //Clear 状态
                     if (!A)  begin
                          state <=Idle;
                          F<=0;  G<=1;
                     end
                     else   state <= Clear;
                end
                default:  state <=Idle;         //默认状态
           endcase
   endmodule
```

例 4.2 与例 4.1 的主要不同点是状态编码方式。例 4.2 采用了独热编码，而例 4.1 则采用 Gray 码，究竟采用哪一种编码好要看具体情况而定。

对于用 FPGA 实现的有限状态机建议采用独热码。因为虽然采用独热编码多用了两个触发器，但所用组合电路可省下许多，因而使电路的速度和可靠性有显著提高，而总的单元数并无显著增加。

采用了独热码后有了多余的状态，就有一些不可到达的状态，为此在 CASE 语句的最后需要增加 default 分支项，以确保多余状态能回到 Idle 状态。

另外还可以用另一种风格的 Verilog HDL 模型来表示同一个有限状态。在这个模型中，我们用 always 语句和连续赋值语句把状态机的触发器部分和组合逻辑部分分成两部分来描述，如下所示。

例 4.3：有限状态机模型 3

```
   module  fsm (Clock, Reset, A,  F, G);                    //模块声明
        input Clock, Reset, A;
        output F,G;
        reg [1:0] state ;
        wire [1:0] Nextstate;

        parameter                                           //状态声明
            Idle  = 2'b00,  Start = 2'b01,
            Stop = 2'b10,   Clear = 2'b11;

        always @(posedge Clock)
            if (!Reset) begin
                 state <= Idle;                             //复位状态
            end
            else
                state <= Nextstate;                         //状态转换
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

```
    assign Nextstate =                                    //状态变换条件
        (state == Idle ) ?  (A ?  Start : Idle):
        (state==Start ) ?  (!A ?  Stop  : Start ):
        (state== Stop ) ?  (A ?  Clear : Stop ):
        (state== Clear) ?  (!A ?  Idle  : Clear) : Idle;

    assign  F  = (( state == Stop) && A );               //状态输出
    assign  G =  (( state == Clear) && (!A || !Reset))   //状态输出
endmodule
```

下面是第 4 种风格的 Verilog HDL 模型来表示同一个有限状态。在这个模型中，我们分别用沿触发的 always 语句和电平敏感的 always 语句把状态机的触发器部分和组合逻辑部分分成两部分来描述。

例 4.4：有限状态机模型 4。

```
module  fsm (Clock, Reset, A,  F, G);              //模块声明
    input Clock, Reset, A;
    output F,G;
    reg [1:0] state, Nextstate;

    parameter                                      //状态声明
        Idle  = 2'b00,  Start = 2'b01,
        Stop = 2'b10, Clear = 2'b11;

    always @(posedge Clock)
        if (!Reset) begin
            state <= Idle;                  //默认状态
        end
        else
            state <= Nextstate;             //状态转换

    always @( state or A ) begin
        F=0;
        G=0;
        if (state == Idle) begin            //处于 Idel 状态时，对 A 判断
            if (A)
                Nextstate  =  Start;        //Start 状态
            else
                Nextstate  =  Idle;         //保持 Idel 状态
            G=1;
        end
        else if (state == Start)            //处于 Start 状态时，对!A 判断
            if (!A)
                Nextstate  =  Stop;         //Stop 状态
            else
                Nextstate  =  Start;        //保持 Start 状态
        else if (state == Stop)             //处于 Stop 状态时，对 A 判断
            if (A)
                Nextstate  =  Clear;        //Clear 状态
            else
```


开发板免费领取试用 开发板免费领取试用

```
                    Nextstate  =  Stop;          //保持 Stop 状态
            else if (state == Clear) begin       //处于 Clear 状态时，对!A 判断
                if (!A)
                    Nextstate  =  Idle;          //Idel 状态
                else
                    Nextstate  =  Clear;         //保持 Clear 状态
                F=1;
                end
            else
                Nextstate= Idle;                 //默认状态
        End
    endmodule
```

上面 4 个例子是同一个状态机的 4 种不同的 Verilog HDL 模型，它们都是可综合的，在设计复杂程度不同的状态机时有它们各自的优势。如用不同的综合器对这 4 个例子进行综合，综合出的逻辑电路可能会有些不同，但逻辑功能是相同的。

下面讲解有限状态机设计的一般步骤。

（1）逻辑抽象，得出状态转换图。

就是把给出的一个实际逻辑关系表示为时序逻辑函数，可以用状态转换表来描述，也可以用状态转换图来描述，这就需要完成以下任务。

① 分析给定的逻辑问题，确定输入变量、输出变量以及电路的状态数。通常是取原因（或条件）作为输入变量，取结果作为输出变量。

② 定义输入、输出逻辑状态的含意，并将电路状态顺序编号。

③ 按照要求列出电路的状态转换表或画出状态转换图。

这样，就把给定的逻辑问题抽象到一个时序逻辑函数了。

（2）状态化简。

如果在状态转换图中出现这样两个状态，它们在相同的输入下转换到同一状态去，并得到一样的输出，则称它们为等价状态。显然等价状态是重复的，可以合并为一个。电路的状态数越少，存储电路也就越简单。状态化简的目的就在于将等价状态尽可能地合并，以得到最简的状态转换图。

（3）状态分配。

状态分配又称状态编码。通常有很多编码方法，编码方案选择得当，设计的电路可以很简单。反之，若编码方案选得不好，则设计的电路就会复杂许多。

实际设计时，需综合考虑电路复杂度与电路性能之间的折衷。在触发器资源丰富的 FPGA 或 ASIC 设计中，采用独热编码（one-hot-coding）既可以使电路性能得到保证，又可充分利用其触发器数量多的优势。

（4）选定触发器的类型并求出状态方程、驱动方程和输出方程。

（5）按照方程得出逻辑图。

用 Verilog HDL 来描述有限状态机，可以充分发挥硬件描述语言的抽象建模能力，使用 always 块语句和 case（if）等条件语句及赋值语句即可方便实现。具体的逻辑化简及逻辑电路到触发器映射均可由计算机自动完成。上述设计步骤中的第（2）、（4）、（5）步不再需要很多的人为干预，使电路设计工作得到简化，效率也有很大的提高。

4.4.2 FSM 设计实例

例 4.5：宇宙飞船控制器的状态机。

```
    module  statmch1(   launch_shuttle, land_shuttle, start_countdown,
                        start_trip_meter, clk, all_systems_go,
                        just_launched, is_landed, cnt, abort_mission
                        );
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

```
        // I/O 说明
        output launch_shuttle, land_shuttle, start_countdown,start_trip_meter;
        input  clk, just_launched, is_landed, abort_mission,all_systems_go;
        input  [3:0]  cnt;
        reg    launch_shuttle, land_shuttle, start_countdown,start_trip_meter;
        reg  [4:0]  present_state,  next_state;
        //设置独热码状态的参数
        parameter  HOLD=5'h1, SEQUENCE=5'h2, LAUNCH=5'h4;
        parameter  ON_MISSION=5'h8,  LAND=5'h10;

        always @(negedge clk or posedge abort_mission) begin
             //把输出设置成某个缺省值,在下面的 case 语句中就不必再设置输出的缺省值
             {launch_shuttle, land_shuttle, start_trip_meter, start_countdown} = 4'b0;
             //检查异步 reset 的值,即 abort_mission 的值
             if(abort_mission)
                 next_state = LAND;
             else begin
                 //如果 abort_mission 为零,把 next_state 赋值为 present_state
                 next_state = present_state;
                 //根据 present_state 和输入信号,设置 next_state 和输出 output
                 case ( present_state )
                     HOLD:                          //HOLD 状态
                          if(all_systems_go) begin
                              next_state = SEQUENCE;
                              start_countdown = 1;
                          end
                     SEQUENCE:                      //SEQUENCE 状态
                          if(cnt==0)
                              next_state = LAUNCH;
                     LAUNCH: begin                  //LAUNCH 状态
                          next_state = ON_MISSION;
                          launch_shuttle = 1;
                          end
                     ON_MISSION:                    //ON_MISSION 状态
                          if(just_launched)
                              start_trip_meter = 1; //取消使命前，一直留在使命状态
                     LAND:                          //LAND 状态
                         if(is_landed)
                              next_state = HOLD;
                         else land_shuttle = 1;
                     default:  next_state = 'bx;//把缺省状态设置为'bx(无关)或某种已
                                                //知状态，使其在做仿真时，在复位前就
                                                //与实际情况相一致
                 endcase
             end                                //// if-else 语句结束
             present_state = next_state;        //把当前状态变量设置为下一状态，
                                                //待下一有效时钟沿来到时，当前状
                                                //态变量已设置了正确的状态值
        end                                     //always 块结束
    endmodule
```

4.4.3 设计可综合状态机的指导原则

（1）独热码。

因为大多数 FPGA 内部的触发器数目相当多，又加上独热码状态机（one hot state machine)的译码逻辑最为简单，所以在设计采用 FPGA 实现的状态机时，往往采用独热码状态机(即每个状态只有一个寄存器置位的状态机)。

（2）case 语句。

建议采用 case、casex 或 casez 语句来建立状态机的模型。因为这些语句表达清晰明了，可以方便地从当前状态分支转向下一个状态并设置输出。

采用这些语句设计状态机时，不要忘记写上 case 语句的最后一个分支 default，并将状态变量设为'bx。这就等于告知综合器：case 语句已经指定了所有的状态。这样综合器就可以删除不需要的译码电路，使生成

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

的电路简洁，并与设计要求一致。

如果将缺省状态设置为某一确定的状态(例如：设置 default：state = state1)，行不行呢？”这样做有一个问题需要注意：因为尽管综合器产生的逻辑和设置“default：state='bx”时相同，但是状态机的 Verilog HDL 模型综合前和综合后的仿真结果会不一致。

为什么会是这样呢？因为启动仿真器时，状态机所有的输入都不确定，因此立即进入 default 状态。如果通过设置将状态变量设为 state1，但是实际硬件电路的状态机在通电之后，进入的状态是不确定的，很可能不是 state1 的状态，这样就会产生不必要的冲突。

因此，还是设置“default：state='bx”与实际硬件电路相一致。但在有多余状态的情况下还是应将缺省状态设置为某一确定的有效状态，因为这样做能使状态机若偶然进入多余状态后仍能在下一时钟跳变沿时返回正常工作状态，否则会引起死锁。

（3）复位。

状态机应该有一个异步或同步复位端，以便在通电时将硬件电路复位到有效状态，也可以在操作中将硬件电路复位（大多数 FPGA 结构都允许使用异步复位端）。

（4）惟一触发。

目前大多数综合器往往不支持在一个 always 块中由多个事件触发的状态机（即隐含状态机，implicit state machines）。因此为了能综合出有效的电路，用 Verilog HDL 描述的状态机应明确地由惟一时钟触发。

（5）异步状态机。

异步状态机是没有确定时钟的状态机，它的状态转移不是由惟一的时钟跳变沿所触发。目前大多数综合器不能综合采用 Verilog HDL 描述的异步状态机。

因此应尽量不要使用综合工具来设计异步状态机。因为目前大多数综合工具在对异步状态机进行逻辑优化时会胡乱地简化逻辑，使综合后的异步状态机不能正常工作。如果一定要设计异步状态机，建议采用电路图输入的方法，而不要用 Verilog HDL 输入的方法。

（6）状态赋值。

Verilog HDL 中，状态必须明确赋值，通常使用参数 parameters 或宏定义 define 语句加上赋值语句来实现。使用参数 parameters 语句赋状态值如下所示：

```
parameter  state1 = 2 'h1,  state2 = 2 'h2;
...
current_state = state2;          //把 current state 设置成 2'h2
...
```

使用宏定义 define 语句赋状态值如下所示：

```
'define state1  2 'h1
'define state2  2 'h2
...
current_state = 'state2;          //把 current state 设置成 2 'h2
```

4.5 逻辑综合的原则以及可综合的代码设计风格

4.5.1 always 块语言指导原则

使用 always 块进行可综合的代码设计时需要注意以下几个问题。

（1）每个 always 块只能有一个事件控制“@(event-expression)”，而且要紧跟在 always 关键字后面。

（2）always 块可以表示时序逻辑或者组合逻辑，也可以用 always 块既表示电平敏感的透明锁存器又同时表示组合逻辑。但是不推荐使用这种描述方法，因为这容易产生错误和多余的电平敏感的透明锁存器。

（3）带有 posedge 或 negedge 关键字的事件表达式表示沿触发的时序逻辑；没有 posedge 或 negedge 关键字的表示组合逻辑或电平敏感的锁存器，或者两种都表示。在表示时序和组合逻辑的事件控制表达式中如

开发板免费领取试用 开发板免费领取试用

有多个沿和多个电平，其间必须用关键字“or” 连接 。

（4）每个表示时序逻辑的 always 块只能由一个时钟跳变沿触发，置位或复位最好也由该时钟跳变沿触发。

（5）每个在 always 块中赋值的信号都必需定义成 reg 型或整型。整型变量缺省为 32bit，使用 Verilog 操作符可对其进行二进制求补的算术运算。综合器还支持整型量的范围说明，这样就允许产生不是 32 位的整型量，句法结构如下：

```
integer [<msb>:<lsb>] <identifier>。
```

（6）always 块中应该避免组合反馈回路。每次执行 always 块时，在生成组合逻辑的 always 块中赋值的所有信号必需都有明确的值；否则需要设计者在设计中加入电平敏感的锁存器来保持赋值前的最后一个值。只有这样，综合器才能正常生成电路。如果不这样做，综合器会发出警告，提示设计中插入了锁存器。如果在设计中存在综合器认为不是电平敏感锁存器的组合回路时，综合器会发出错误信息(例如设计中有异步状态机时)。

用 always 块设计纯组合逻辑电路时，在生成组合逻辑的 always 块中，参与赋值的所有信号都必须有明确的值，即在赋值表达式右端参与赋值的信号都必需在 always @(敏感电平列表)中列出。

如果在赋值表达式右端引用了敏感电平列表中没有列出的信号，那么在综合时，将会为该信号产生一个隐含的透明锁存器。这是因为该信号的变化不会立刻引起所赋值的变化，而必须等到敏感电平列表中某一个信号变化时，它的作用才显现出来。

也就是相当于存在着一个透明锁存器把该信号的变化暂存起来，待敏感电平列表中某一个信号变化时再起作用，纯组合逻辑电路不可能做到这一点。这样，综合后所得电路已经不是纯组合逻辑电路了。这时综合器会发出警告提示设计中插入了锁存器，如下所示：

```
input a,b,c;
reg e,d;
always @(a or b or c) begin
     e = d & a & b;       //因为 d 没有在敏感电平列表中,所以 d 变化时,e 不能立刻变化,
                          //要等到 a 或 b 或 c 变化时才体现出来。这就是说实际上相当于存在
                          //一个电平敏感的透明锁存器在起作用，把 d 信号的变化锁存其中
     d = e | c;
end
```

（7）对一个寄存器型（reg）或整型（integer）变量的赋值只允许在一个 always 块内进行，如果在另一 always 块也对其赋值，这是非法的。

（8）把某一信号值赋为'bx，综合器就把它解释成无关状态，因而综合器为其生成的硬件电路最简洁。

4.5.2 可综合风格的 Verilog HDL 模块实例

1．组合逻辑电路设计实例

例 4.6：8 位带进位端的加法器的设计实例（利用简单的算法描述）。

```
module adder_8(cout,sum,a,b,cin);              //模块声明
     output cout;
     output [7:0] sum;
     input cin;
     input [7:0] a,b;                          //端口声明
     assign {cout,sum} = a + b + cin;          //加法器算法
endmodule
```

例 4.7：指令译码电路的设计实例（利用电平敏感的 always 块来设计组合逻辑）。

开发板免费领取试用 开发板免费领取试用

```
'define plus    3'd0                    //操作码的宏定义
'define minus   3'd1
'define band    3'd2
'define bor     3'd3'
'define unegate 3'd4

module alu(out,opcode,a,b);             //模块声明
     output [7:0] out;
     input [2:0] opcode;
     input [7:0] a,b;                   //端口声明
     reg   [7:0] out;                   //寄存器声明

     always @(opcode or a or b) begin   //用电平敏感的 always 块描述组合逻辑
          case(opcode)
               'plus: out = a + b;      //算术运算
               'minus: out = a - b;
               'band: out = a & b;      //位运算
               'bor:  out = a | b;
               'unegate: out = ~a;      //单目运算
                default: out = 8'hx;
     endcase
     end
endmodule
```

例 4.8：比较后重组信号的组合逻辑（利用 task 和电平敏感的 always 块设计）。

```
module sort4(ra,rb,rc,rd,a,b,c,d);      //模块声明
     output [t:0] ra, rb, rc, rd;
     input [t:0] a, b, c, d;            //端口声明
     reg [t:0] ra, rb, rc, rd;
     reg [t:0] va, vb, vc, vd;          //寄存器声明
     parameter t=3;                     //参数声明

     always @(a or b or c or d) begin   //用电平敏感的 always 块描述组合逻辑
         {va,vb,vc,vd}={a,b,c,d};
         sort2(va,vc);                   //信号重组
         sort2(vb,vd);
         sort2(va,vb);
         sort2(vc,vd);
         sort2(vb,vc);
         {ra,rb,rc,rd}={va,vb,vc,vd};
     end

     task sort2;                        //x 与 y 互换任务
          inout [t:0] x,y;
          reg [t:0] tmp;
          if(x > y) begin
              tmp = x;                  //使用临时变量 tmp 保存 x 的值
              x = y;
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

```
                y = tmp;
        end
        endtask
    endmodule
```

例 4.9：比较器的设计实例（利用赋值语句设计组合逻辑）。

```
    module compare(equal,a,b);              //模块声明
        output equal;
        input [size-1:0] a,b;               //端口声明
        parameter size=1;                   //参数声明
        assign  equal =(a==b)? 1 : 0;       //比较器
    endmodule
```

例 4.10：3-8 译码器设计实例（利用赋值语句设计组合逻辑）。

```
    module decoder(out,in);             //模块声明
        output [7:0] out;
        input [2:0] in;                 //端口声明
        assign  out = 1'b1<<in;         //把最低位的 1 左移 in（根据从 in 口输入的值）位
                                        //将移位结果赋予 out
    endmodule
```

例 4.11：3-8 编码器的设计实例。
编码器设计方案一。

```
    module encoder1(none_on,out,in);            //模块声明
        output none_on;
        output [2:0] out;
        input [7:0] in;                         //端口声明
        reg [2:0] out;
        reg none_on;                            //寄存器声明
        always @(in) begin: local               //in 有变化时，触发
            integer  i;                         //变量声明
            out = 0;
            none_on = 1;                        //初始化
            for( i=0; i<8; i=i+1 ) begin        //for 循环语句
                if( in[i] ) begin               //将 in 中值为 1 的位编码
                    out = i;
                    none_on = 0;
                end
            end
        end
    endmodule
```

编码器设计方案二。

```
    module encoder2 ( none_on,out2,out1,out0,h,g,f,e,d,c,b,a);     //模块声明
        input h,g,f,e,d,c,b,a;
        output none_on,out2,out1,out0;          //端口声明
        wire [3:0] outvec;                      //向量声明
        assign outvec =                         //使用 assign 语句实现输出向量赋值
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

```
            h ? 4'b0111 : g ? 4'b0110 : f ? 4'b0101:
            e ? 4'b0100 : d ? 4'b0011 : c ? 4'b0010 :
            b ? 4'b0001 : a ? 4'b0000 : 4'b1000;
      assign none_on = outvec[3];          //使用 assign 语句进行编码
      assign   out2 = outvec[2];
      assign   out1 = outvec[1];
      assign   out0 = outvec[0];
endmodule
```

编码器设计方案三。

```
module encoder3 ( none_on,out2,out1,out0,h,g,f,e,d,c,b,a);     //模块声明
      input h,g,f,e,d,c,b,a;
      output none_on,out2,out1,out0;      //端口声明
      wire [3:0] outvec;                   //向量声明
      assign {none_on,out2,out1,out0} = outvec;   //与上例的编码方式一致
      always @( a or b or c or d or e or f or g or h) begin
            if(h)      outvec=4'b0111;     //使用 if_else 语句实现向量赋值
            else if(g) outvec=4'b0110; //共 9 个分支，其中向量的低 3 位有 8 种编码方式
            else if(f) outvec=4'b0101;
            else if(e) outvec=4'b0100;
            else if(d) outvec=4'b0011;
            else if(c) outvec=4'b0010;
            else if(b) outvec=4'b0001;
            else if(a) outvec=4'b0000;
            else       outvec=4'b1000;
      end
endmodule
```

例 4.12：多路器的设计实例。

使用 assign 赋值语句、case 语句或 if-else 语句可以生成多路器电路。如果条件语句（case 或 if-else）中分支条件是互斥的话，综合器能自动地生成并行的多路器。

多路器设计方案一。

```
modul emux1(out,a,b,sel);              //模块声明
    output out;
    input a,b,sel;                     //端口声明
                                       //使用 assign 语句检查输入信号 sel 的值
    assign out = sel ? a : b;          //当 sel 为 1 时，out 为 a；否则为 b
endmodule
```

多路器设计方案二。

```
module  mux2( out,a,b,sel);          //模块声明
      output out;
      input a,b,sel;                 //端口声明
      reg out;
      always @(a or b or sel) begin //用电平触发的 always 块来设计多路器的组合逻辑
            case( sel )              //使用 case 语句检查输入信号 sel 的值
               1'b1: out = a;        //如果为 1，输出 out 为 a
```


开发板免费领取试用 开发板免费领取试用

```
            1'b0: out = b;          //如果为 0，输出 out 为 b
            default: out = 'bx;     //默认状态
        endcase
    end
endmodule
```

多路器设计方案三。

```
module mux3( out,a,b sel);                  //模块声明
    output out;
    input a, b, sel;                        //端口声明
    reg out;
    always @( a or b or sel ) begin
        if( sel )                           //使用 if_else 语句检查输入信号 sel 的值
            out = a;                        //如果为 1，输出 out 为 a
        else
            out = b;                        //如果为 0，输出 out 为 b
    end
endmodule
```

例 4.13：奇偶校验位生成器设计实例。

```
module parity( even_numbits,odd_numbits,input_bus);  //模块声明
    output even_numbits, odd_numbits;
    input [7:0] input_bus;                  //端口声明
    assign odd_numbits = ^input_bus; //当 input_bus 中 1 的个数为奇数时，输出为 1
    assign even_numbits = ~odd_numbits;  //此时输出 even_numbits 为 0
endmodule
```

例 4.14：三态输出驱动器设计实例（用连续赋值语句建立三态门模型）。
三态输出驱动器设计方案一。

```
module trist1( out,in,enable);              //模块声明
    output out;
    input in, enable;                       //端口声明
    assign  out = enable? in: 'bz;          //使用 assign 语句判断 enable 的值
endmodule
```

三态输出驱动器设计方案二。

```
module trist2( out,in,enable );    //模块声明
    output out;
    input in,enable;               //端口声明
    bufif1 mybuf1(out, in, enable); //bufif1 是一个 Verilog 门级原语（primitive）
                                   //通过实例化该原语，实现三态门的调用
endmodule
```

例 4.15：三态双向驱动器设计实例。

```
module bidir(tri_inout,out,in,en,b);        //模块声明
    inout tri_inout;
    output out;
    input in,en,b;                          //端口声明
    assign tri_inout = en? in : 'bz;        //三态门的输入为 in
    assign out = tri_inout ^ b;             //三态门的输出为 b
endmodule
```


开发板免费领取试用 开发板免费领取试用

2．时序逻辑电路设计实例

例 4.16：触发器设计实例。

```
module dff( q,data,clk);                    //模块声明
     output q;
     input data,clk;                        //端口声明
     reg q;
     always @( posedge clk ) begin          //边缘检测
          q = data;                         //通过 always 语句，实现触发器
     end
endmodule
```

例 4.17：电平敏感型锁存器设计实例一（assign 语句）。

```
module latch1( q,data,clk);                  //模块声明
     output q;
     input data,clk;                         //端口声明
     assign q = clk ? data : q;              //通过 assign 语句，实现的是一个锁存器
endmodule
```

例 4.18：带置位和复位端的电平敏感型锁存器设计实例二（assign 语句）。

```
module latch2( q,data,clk,set,reset);   //模块声明
     output q;
     input data,clk,set,reset;          //端口声明
     assign q= reset ? 0 : ( set? 1:(clk? data : q ) );
                                        //通过 assign 语句，实现的是一个锁存器
                                        //其中，set 为置位端，reset 为复位端
                                        //在 clk 为高电平时，锁存 data，否则保持 q 值
endmodule
```

例 4.19：电平敏感型锁存器设计实例三（always 块）。

```
module latch3( q, data, clk);          //模块声明
     output q;
     input data,clk;                   //端口声明
     reg q;
     always @(clk or data) begin       //电平检测
          if(clk)                      //clk 为高电平时，q 锁存 data 值
               q = data;
     end
endmodule
```

> **注意** 有的综合器会产生一个警告信息，提示将产生了一个电平敏感型锁存器。因为此例中设计的就是一个电平敏感型锁存器，所以这个警告信息是没有问题的。

例 4.20：移位寄存器设计实例。

```
module shifter( din,clk,clr,dout);        //模块声明
     input din,clk,clr;
     output [7:0] dout;                   //端口声明
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

```
    reg [7:0] dout;
    always @(posedge clk) begin
        if(clr)                          //清零
            dout = 8'b0;
        else begin
            dout = dout<<1;              //左移一位
            dout[0] = din;               //把输入信号放入寄存器的最低位
        end
    end
endmodule
```

例 4.21：8 位计数器设计实例一。

```
module counter1( out, cout, data, load, cin, clk);     //模块声明
    output [7:0] out;
    output cout;
    input [7:0] data;
    input load, cin, clk;               //端口声明
    reg [7:0] out;
    always @(posedge clk) begin         //边缘检测
        if( load )                      //加载信号检测
            out = data;
        else
            out = out + cin;
        end
    assign cout= & out & cin;           //只有当 out[7:0] 的所有各位都为 1
                                        //并且进位 cin 也为 1 时才能产生进位 cout
endmodule
```

例 4.22：8 位计数器设计实例二。

```
module counter2( out, cout, data, load, cin, clk);     //模块声明
    output [7:0] out;
    output cout;
    input [7:0] data;
    input load, cin, clk;                              //端口声明
    reg [7:0] out;
    reg cout;
    reg [7:0] preout;                                  //寄存器声明
    always @(posedge clk) begin                        //边缘检测
        out = preout;                                  //触发器
        end
    //计算计数器和进位的下一个状态，为提高性能，load 不应影响进位
    always @( out or data or load or cin ) begin
        {cout, preout} = out + cin;                    //进位操作
        if(load) preout = data;                        //判断加载信号
        end
endmodule
```


开发板免费领取试用 开发板免费领取试用

4.6 典型实例 6：状态机应用

4.6.1 实例的内容及目标

1．实例的主要内容

状态机设计是 HDL 设计里面的精华，几乎所有的设计里面都或多或少地使用了状态机的思想。状态机，顾名思义，就是一系列状态组成的一个循环机制，这样的结构使得编程人员能够更好地使用 HDL 语言，同时具有特定风格的状态机也能提高程序的可读性和调试性。

本实例通过设计一个状态机来控制红色飓风 II 代 Xilinx 开发板上面的 8 个 LED 灯循环闪烁。在本实例状态机的设计过程中，读者需要着重注意状态机设计的一些特性。

- FSM 设计方法（米勒型和摩尔型）。
- 状态机的编码（Binary、gray-code、one-hot 等）。
- 状态机的初始化状态和默认状态（完整状态机设计）。
- 状态机的状态定义风格（使用 parameter 参数定义状态）。
- 状态机的编写风格（状态任务与状态转换分离）

2．实例目的

通过本实例，读者应达到下面的目标。

- 熟悉 Xilinx ISE 开发环境。
- 熟悉红色飓风 II 代 Xilinx 开发板的使用。
- 了解状态机设计输入方法。
- 掌握状态机在系统设计中的应用。

4.6.2 实例详解

本实例的具体步骤可参见 2.6 节，在此不再详述，仅给出主要的操作流程以及状态机的设计输入方法。

（1）启动 ISE 软件。

（2）创建新工程。

（3）编写状态机 Verilog 代码。

状态机的代码可以直接进行手动编写，如图 4.2 所示。

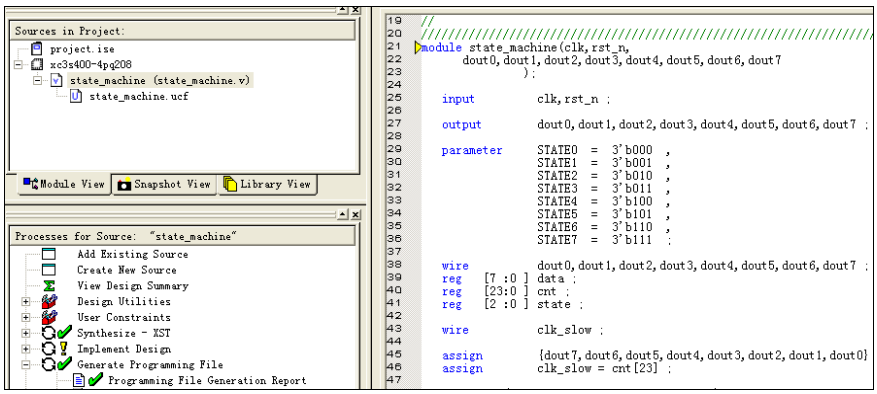


图 4.2 手动编写状态机代码

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

对于简单的状态机，使用手动输入的方式进行编写是最直接的。但是如果需要进行复杂的状态机设计，使用状态图设计输入方式则会更加方便。下面介绍使用状态图进行状态机设计的一般步骤。

使用 ISE 创建工程后，为工程添加新的设计输入，选择“State Diagram”选项，如图 4.3 所示。

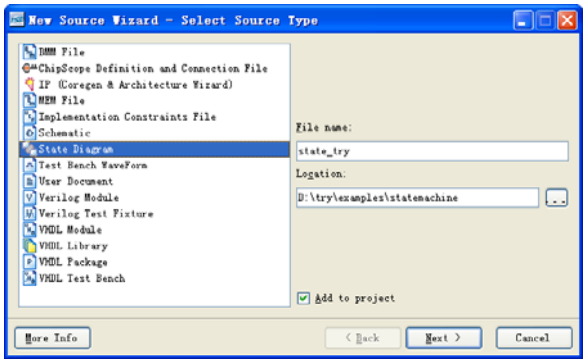


图 4.3　状态图设计输入方式

此时 ISE 将打开如图 4.4 所示的 StateCAD 工具进行状态机设计。

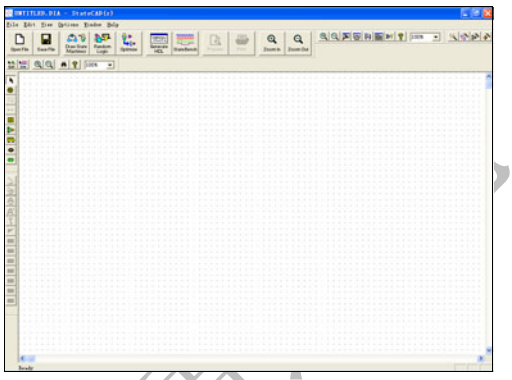


图 4.4　StateCAD 状态机设计工具

在这个专门的工具中，读者就可以很方便地进行状态机的设计。使用这个工具进行状态机设计的步骤主要分为 3 个，下面分别介绍。

（1）创建状态机。

单击工具栏上的图标，打开状态机输入向导，如图 4.5 所示。

在本实例中，我们在向导第 1 页设置一个 8 状态的状态机，如图 4.5 所示。在该页中，还可以设置状态机图形的形状。在对话框的右侧有图形的预览效果，用户可根据自己的喜好设置图形的形状。

在状态机向导的第 2 页，我们为状态机使用同步复位模式，建立一个同步状态机，如图 4.6 所示。

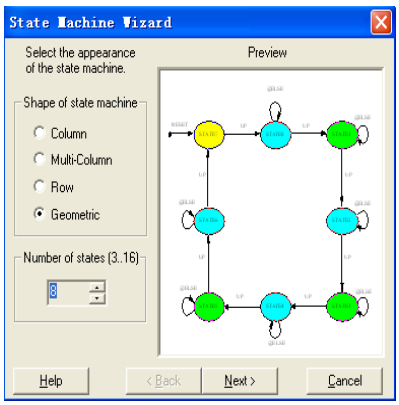


图 4.5　状态机向导

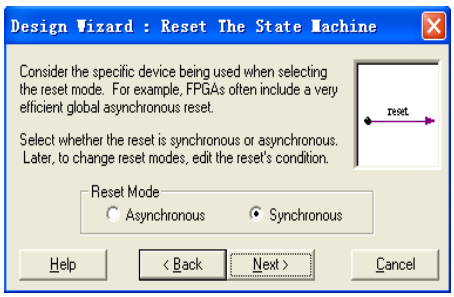


图 4.6　同步状态机

在状态机向导的第 3 页，可以设置状态机的转换模式，本实例中使用 Next 转换模式，如图 4.7 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

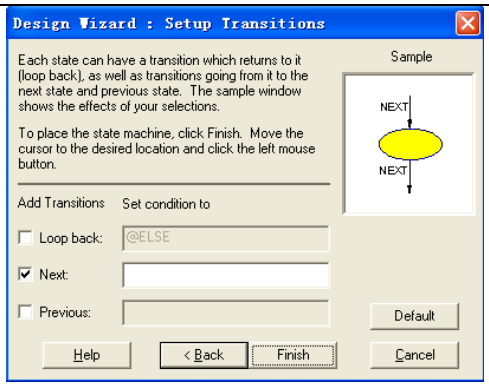


图 4.7　状态机转换模式

设置完毕后，单击“Finish”按钮即可完成状态机的创建。在 StateCAD 工具的工作区将出现如图 4.8 所示的状态机图形。

（2）添加状态任务。

创建状态机后，就要为其中的每一个状态添加状态任务。首先，双击如图 4.8 所示的图形中的 STATE0 状态，为 STATE0 添加任务。此时，将出现如图 4.9 所示的状态任务编辑器。

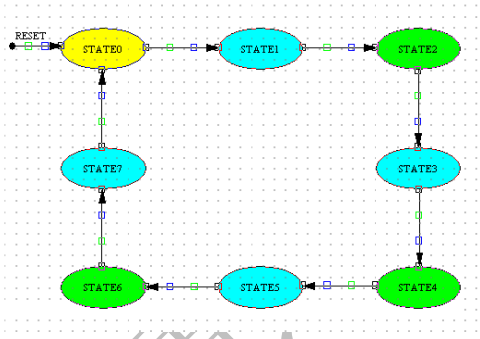


图 4.8　创建的状态机

单击“Output Wizard”按钮，打开输出向导对话框，如图 4.10 所示。

图 4.9　状态任务编辑器

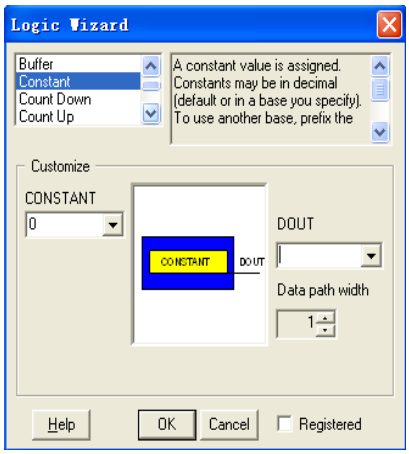


图 4.10　状态输出向导对话框

在输出向导对话框里面有很多可以选择的逻辑，根据不同的需要选择逻辑资源，并输入参数。单击“OK”按钮即可完成对 STATE0 的状态任务编辑。

按照相同的方法分别为每个状态加载任务，得到如图 4.11 所示的状态机。

（3）添加状态转换条件。

设置完状态任务后，还需要为状态之间的转换调价条件。可以双击两个状态之间的连线设定状态转移条件，此时将出现条件编辑器，如图 4.12 所示。

在该对话框中，可以在“Condition”下面填写转移条件（例如：cnt=10），也可以单击“Output Wizard”

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

按钮打开输出向导进行设置。设置结束后，单击“OK”按钮，完成两个状态之间的转换条件设置。

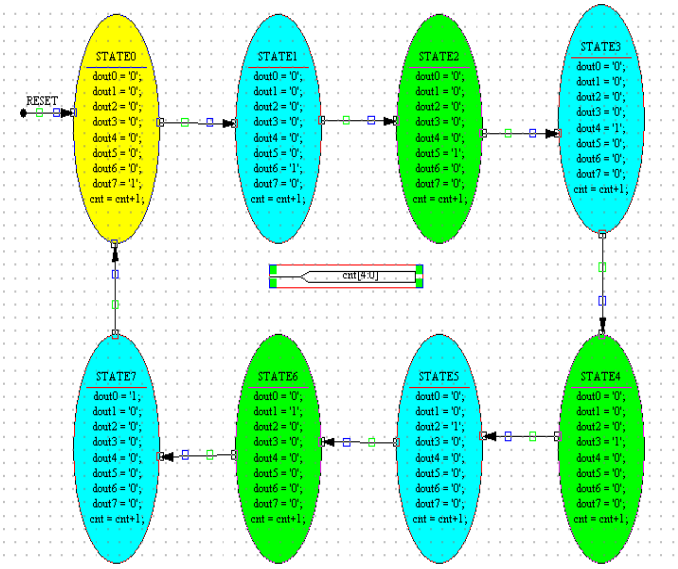


图 4.11　添加任务的状态机图形

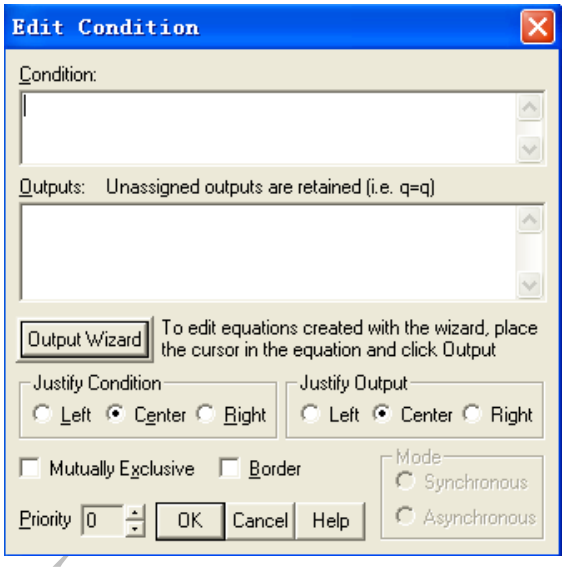


图 4.12　状态转换条件编辑器

按照同样的方法为需要添加转换条件的状态连线加载状态转移条件。完成后，得到如图 4.13 所示的状态机。

至此，一个基本的状态机已经建立完毕。这样的一个状态机已经可以在我们的系统中使用。不过在使用之前，我们还需要为 StateCAD 工具做一些设置，才能使图形形式的状态机转化为 HDL 语言的形式。

1．优化设置

首先对状态机进行优化设置，选择 StateCAD 工具条上的按钮，打开优化设置向导，如图 4.14 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

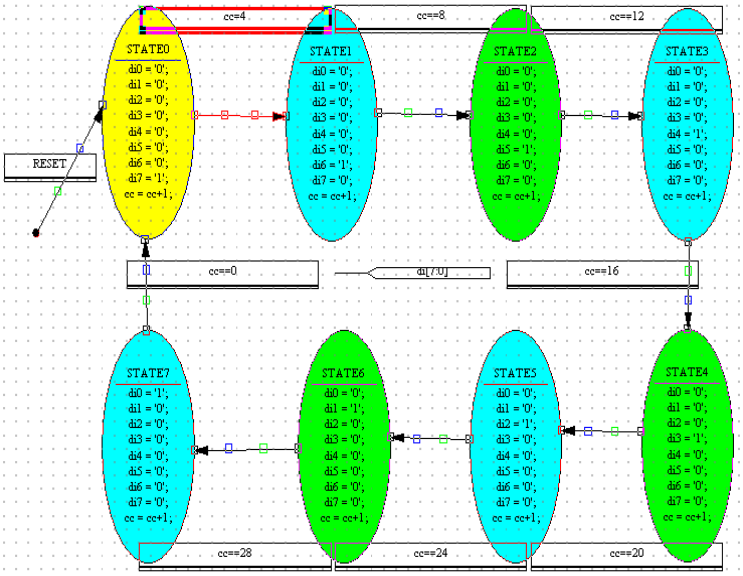


图 4.13　添加状态转换条件的状态机

图 4.14　优化设置向导

在如图 4.14 所示的向导中，提示将进行 5 个步骤的设置，下面分别介绍。

① 选择目标器件。本实例中选择 FPGA，如图 4.15 所示。

图 4.15　选择目标器件

② 选择状态机的优化方向，用户可以选择速度优先、占用面积优先以及手动设置。在本实例中我们选择速度优先，如图 4.16 所示。

③ 对状态机的性能改善方式进行选择，如图 4.17 所示，本实例选择保证覆盖面积模式。

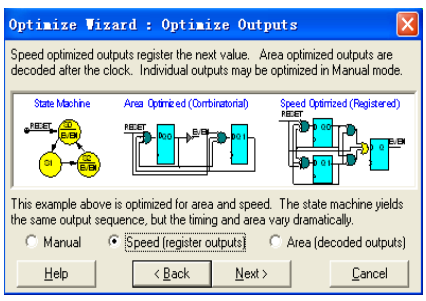


图 4.16　状态机优化方向

开发板免费领取试用　开发板免费领取试用

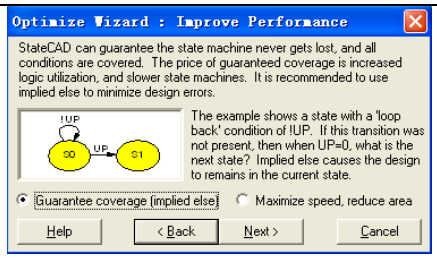


图 4.17　状态机性能改善模式

④ 加载优化参数设置，如图 4.18 所示。

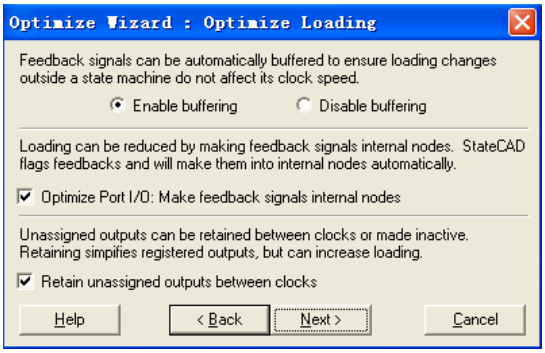


图 4.18　加载优化参数设置

⑤ 为状态机选择输出的 HDL 语言形式以及综合的 EDA 工具。在本实例中，选择 Verilog 语言及 Xilinx XST 综合工具，如图 4.19 所示。

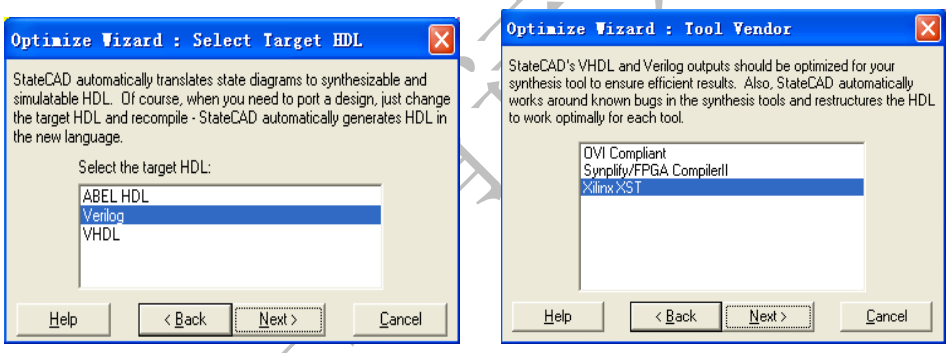


图 4.19　选择目标 HDL 语言和 EDA 工具

经过这 5 个步骤的设置，就完成了状态机的优化设置，可以开始生成 Verilog 语言形式的状态机了。

选择 StateCAD 工具栏上的按钮，开始生成 Verilog 状态机。如果设计没有错误，将出现如图 4.20 所示的结果对话框。单击“Close”按钮后，就可以看见生成的与状态机名称相同、后缀为.v 的源文件了。

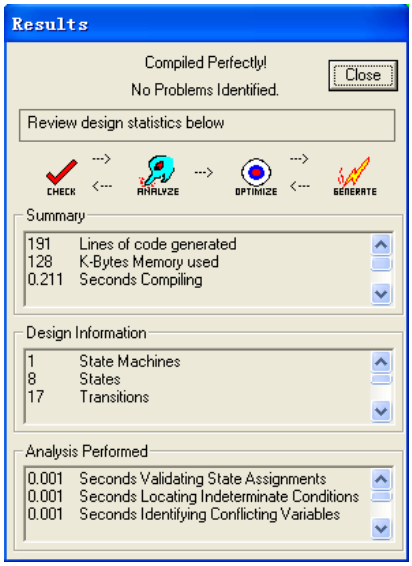


图 4.20　状态机生成结果

2．添加设计输入

将使用 StateCAD 生成的状态机源文件加入工程中，即可查看并编辑相关的状态机的代码了。

3．设置器件及管脚约束

按照开发板的说明进行相关的设置。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

4．下载验证

本实例的状态机在开发板上下载后，将能看到开发板上的 8 个 LED 循环变亮，即实现跑马灯的功能。

4.6.3 参考设计

本实例相关参考设计文件在本书实例代码的“典型实例 6”文件夹。

4.7 典型实例 7：自动转换量程频率计控制器

4.7.1 实例内容及目标

1．实例内容

本实例使用 Verilog HDL 设计一个可自动转换量程的频率计控制器。在设计过程中，使用了状态机的设计方法，读者可根据综合实例 6 的流程将本实例的语言设计模块添加到自己的工程中。

2．实例目标

通过本实例，读者应达到下面的目的。

- 掌握使用 Verilog 设计状态机的方法。
- 掌握 Verilog 设计的一般方法。

4.7.2 原理简介

频率计是电路调试里面常常用到的一种仪器。本实例提到的自动转换量程频率计控制器并不是讲解如何设计一个频率计，而是讲解如何设计这个频率计的量程转换机制。通过 Verilog 语言设计将量程转换变成自动化，自适应地将输入反映至量程上。

此自动转换量程频率计需要外部提供一个超量程信号和一个欠量程信号，代表输入比当前量程的状态。这两个信号可以通过其他的模块或者装置获得，读者可自行分析。

同时，该频率计还向外部提供一个用于选择标准时基的信号。通过该信号，频率计可以完成量程的切换和显示。读者可根据频率计其他模块的需要调整输出信号的设计。

4.7.3 代码分析

下面给出自动转换量程频率计控制器的 Verilog 源代码，首先介绍端口信号的定义及说明，读者可以通过这些端口将此控制器模块实例化至自己的工程设计中。

- clk：输入时钟。
- clear：异步复位信号。
- reset：用来在量程转换开始时复位计数器。
- std_f_sel：用来选择标准时基。

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

- cntover：代表超量程。
- cntlow：代表欠量程。

在状态机设计中，常常将状态转换和状态输出控制分为两个部分进行设计，方便语言的编写修改和读写规则。在下面的源代码中读者应该注意这个设计的特点。

```
module control(std_f_sel,reset,clk,clear,cntover,cntlow);
        //端口说明
        output[1:0] std_f_sel;
        output reset;
        input clk,clear,cntover,cntlow;

        //内部信号说明
        reg[1:0] std_f_sel;
        reg reset;
        reg[5:0] present,next;              //用于保存当前状态和次态的中间变量

        //状态编码，采用独热码
        parameter   start_fl00k =   6'b000001,    //状态A
                fl00k_cnt     =  6'b000010,     //状态B
                start_fl0k  =  6'b000100,       //状态C
                fl0k_cn     = 6'b001000,        //状态D
                start_flk    =   6'b010000,     //状态E
                flk_cnt    =    6'b100000;     //状态F

        always @(posedge clk or posedge clear) begin
         if(clear)
             present<=start_fl0k;               //异步复位至start_fl0k状态
         else
             present<=next;                     //状态转换
        end

        always @(present or cntover or cntlow) begin  //状态转换的触发信号列表
         case(present)                          //用case语句描述状态转换
              start_fl00k:                      //100k量程状态
                   next<=fl00k_cnt;
              fl00k_cnt: begin                  //100k量程控制状态
                   if(cntlow)                   //欠量程
                       next<=start_fl0k;        //进入10k量程状态
                   else
                       next<=fl00k_cnt;         //保持100k量程控制状态
              end
              start_fl0k:                       //10k量程状态
                   next<=fl0k_cnt;
              fl0k_cnt: begin                   //10k量程控制状态
                   if(cntlow)                   //欠量程
                       next<=start_flk;         //进入1k量程状态
                   else if(cntover)             //过量程
                       next<=start_fl00k;       //进入100k量程状态
                   else
                       next<=fl0k_cnt;          //保持1k量程控制状态
              end
              start_flk:                        //1k量程状态
                   next<=flk_cnt;
```


开发板免费领取试用 开发板免费领取试用

```
                flk_cnt: begin                      //1k 量程控制状态
                     if(cntover)                    //过量程
                         next<=start_fl0k;          //进入 10k 量程状态
                     else
                         next<=flk_cnt;             //保持 1k 量程控制状态
                end
                default:
                     next<=start_fl0k;              //缺省状态为 10k 量程状态
           endcase
          end

          //各状态的输出控制模块
          always @(present) begin
           case(present)
                start_fl00k: begin                  //100k 量程状态输出控制
                     reset=1;
                     std_f_sel=2'b00;
                end
                fl00k_cnt: begin                    //100k 量程控制状态输出控制
                     reset=0;
                     std_f_sel=2'b00;
                end
                start_fl0k: begin                   //10k 量程状态输出控制
                     reset=1;
                     std_f_sel=2'b01;
                end
                fl0k_cnt: begin                     //10k 量程控制状态输出控制
                     reset=0;
                     std_f_sel=2'b01;
                end
                start_flk: begin                    //1k 量程状态输出控制
                     reset=1;
                     std_f_sel=2'b11;
                end
                flk_cnt: begin                      //1k 量程控制状态输出控制
                     reset=0;
                     std_f_sel=2'b11;
                end
                default: begin                      //默认（10k 量程）状态输出控制
                     reset=1;
                     std_f_sel=2'b01;
                end
            endcase
           end
    endmodule
```

4.7.4 参考设计

本实例相关参考设计文件在本书实例代码的“典型实例 7”文件夹。

4.8 典型实例 8：使用函数实现简单的处理器

4.8.1 实例的内容及目标

1．实例内容

本实例使用 Verilog HDL 设计一个简单 8 位处理器，可以实现两个 8 位操作数的 4 种操作。在设计过程中，

开发板免费领取试用 开发板免费领取试用

使用了函数调用的设计方法。

2．实例目标

通过本实例，读者应达到下面的一些目的。

- 掌握使用 Verilog 函数设计的方法。
- 掌握 Verilog 设计的一般方法。

4.8.2 原理简介

处理器发展到现在，已经变成一个功能极其强大，设计也极为复杂的单元。现在一颗小小的 CPU 的运算能力远远的超过了以前的超级计算机的能力。虽然处理器处理能力的提高和很多方面的因素有关，可是处理器最终要完成的目标还是早期 CPU 设计时的那个目标，那就是使用操作码实现对操作数的控制和计算。在本实例中，实现的处理器是一个 8 位处理器，只实现简单的 4 种操作：相加、相减、操作数减 1 和操作数加 1。通过这 4 种操作向读者展示如何使用 Verilog 语言设计看似复杂的 CPU 单元。
当然，这个处理器只是一个算术单元，处理器还包括很多其他的功能，在此不进行介绍，感兴趣的读者可以查阅相关的书籍，按照本实例提供的设计方向补充和完善这个处理器。

4.8.3 代码分析

下面给出这个处理器的 Verilog 源代码，读者可以将此处理器模块实例化至自己的工程设计中。

```
module mpc(instr,out);
        //端口说明
        input[17:0] instr;              //输入指令
        output[8:0] out;           //输出结果
        //内部信号说明
        reg[8:0] out;
        reg func;                  //指令中提取出的操作码的内部变量
        reg[7:0] op1,op2;          //从指令中提取的两个操作数
        //函数声明
        function[16:0] code_add;  //函数的定义，返回一个17位的指令
          input[17:0] instr;       //函数的输入，采用与模块输入同样的命名，可不同
          //函数内部信号说明
          reg add_func;            //函数内部的操作码变量
          reg[2:0] code;           //操作码
          reg[7:0] opr1,opr2;      //两个操作数

          begin
              code=instr[17:16]; //输入指令instr的高2位为操作码code
              opr1=instr[7:0];    //输入指令instr的低8位为操作数opr1
              //通过case语句判断操作码的类型，获得操作数opr2
              case(code)
                   2'b00: begin
                        add_func=1;
                        opr2=instr[15:8];//从instr中取第二个操作数
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

```
                    end
                2'b01: begin
                    add_func=0;
                    opr2=instr[15:8];//从 instr 中取第二个操作数
                    end
                2'b10: begin
                    add_func=1;
                    opr2=8'd1;          //第二个操作数取为 1，实现+1 操作
                    end
                2'b11: begin
                    add_func=0;
                    opr2=8'd1;          //实现-1 操作
                    end
                endcase
                code_add={add_func,opr2,opr1};  //函数的返回值
        end
        endfunction
        //函数调用模块
        always @(instr) begin
          {func,op2,op1}=code_add(instr);   //调用函数
          if(func==1)
                out=op1+op2;                //实现两数相加或操作数 1 加 1 操作
          else
                out=op1-op2;                //实现两数相减或操作数 1 减 1 操作
          end
endmodule
```

联系方式

集团官网：www.hqyj.com　　嵌入式学院：www.embedu.org　　移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn　　物联网学院：www.topsight.cn　　研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路银海大厦 A 座 8 层，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218

开发板免费领取试用　开发板免费领取试用

华清远见 FARSIGHT 始于2004 | 嵌入式培训专家

10年口碑积累，成功培养50000多名研发工程师，铸就专业品牌形象

华清远见的企业理念是不仅要做良心教育、做专业教育，更要做受人尊敬的职业教育。

《FPGA 应用开发入门与典型实例》（修订版）

作者：华清远见

专业始于专注 卓识源于远见

第 5 章 FPGA 设计开发软件 Quartus II 的使用技巧

本章目标

- 掌握 Quartus II 软件的安装和基本操作
- 掌握 Quartus II 下 FPGA 的设计流程
- 掌握 Signal TAPII 和 LogicLock 两个重要的设计工具

专业始于专注 卓识源于远见

开发板免费领取试用 开发板免费领取试用

5.1 Quartus II 软件简介

Quartus II 设计软件是 Altera 提供的完整的多平台设计环境，能够直接满足特定设计需要，为可编程芯片系统（SOPC）提供全面的设计环境。Quartus II 软件含有 FPGA 和 CPLD 设计所有阶段的解决方案。
Quartus II 是 Altera 公司新一代的 EDA 设计工具，由该公司早先的 MAX+PLUS II 演变而来。不仅继承了 MAX+PLUS II 工具的优点，更提供了对新器件和新技术的支持，使设计者能够能轻松和全面地介入设计的每一个环节。本章将对这个软件进行全面的介绍。

5.2 Quartus II 软件新特性

可以按照个人偏好，自定义 Quartus II 软件的布局、菜单、命令和图标。当第一次打开 Quartus II 软件时，可以在标准 Quartus II 用户界面和 MAX+PLUS II 界面之间选择外观和操作习惯，或者稍候使用“Customize”对话框（Tools 菜单）选择外观和操作习惯。
“Customize”对话框也允许设置是否显示 Quartus II 或者 MAX+PLUS II 快捷菜单，是设置在菜单栏的左边还是右边。Quartus II 快捷菜单包含的菜单命令用于每个 Quartus II 应用程序，是常用处理命令。如图 5.1 所示为“Customize”对话框。
MAX+PLUS II 快捷菜单与 MAX+PLUS II 软件的 MAX+PLUS II 菜单类似，提供用于应用程序的命令，也是常用的 MAX+PLUS II 菜单命令。MAX+PLUS II 菜单上的命令与相应的 Quartus II 命令功能相同。如图 5.2 所示为 Quartus II 和 MAX+PLUS II 快捷菜单。

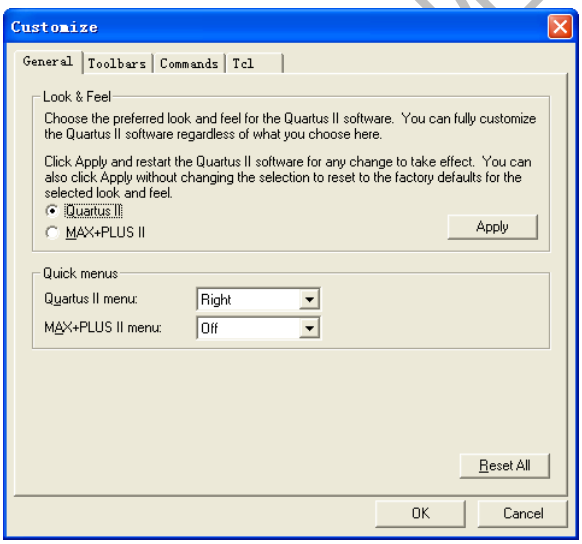


图 5.1 Customize 对话框

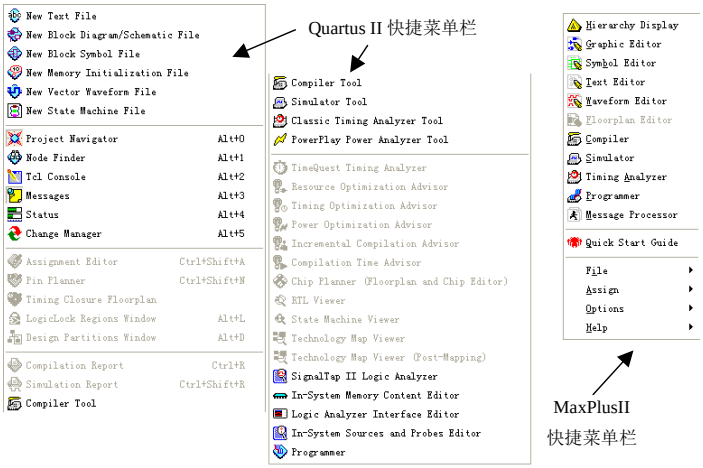


图 5.2 Quartus II 和 MAX+PLUS II 快捷菜单

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

以下步骤描述了使用 Quartus II 基本设计流程。

（1）使用 New Project Wizard（“File” 菜单）建立新工程并指定目标器件或器件系列。

（2）使用 Text Editor 建立 Verilog HDL、VHDL 或 Altera 硬件描述语言（AHDL）设计。根据需要，使用 Block Editor 建立表示其他设计文件的符号框图，也可以建立原理图。还可以使用 Mega Wizard® Plug-In Manager（“Tools” 菜单）生成宏功能模块和 IP 功能的自定义变量，在设计中将它们例化。

（3）使用 Assignment Editor、Pin Planner、Settings 对话框（“Assignments” 菜单）、Floorplan Editor、“Design Partitions” 窗口、“LogicLock” 功能指定初始设计约束。

（4）进行 Early Timing Estimate，在完成 Fitter 之前生成时序结果的早期估算。

（5）使用 SOPC Builder 或 DSP Builder 建立系统级设计。

（6）使用 Software Builder 为 Excalibur™器件处理器或 Nios®嵌入式处理器建立软件和编程文件。

（7）使用 Analysis & Synthesis 对设计进行综合。

（8）如果设计含有分区，而没有进行完整编译，则需要采用 Partition Merge 合并分区。

（9）通过使用 Simulator 和 Generate Functional Simulation Netlist 命令在设计中执行功能仿真。

（10）使用 Fitter 对设计进行布局布线。

（11）使用 PowerPlay Power Analyzer 进行功耗估算和分析。

（12）使用 Timing Analyzer 对设计进行时序分析。

（13）使用 Simulator 对设计进行时序仿真。

（14）使用物理综合、Timing Closure 平面布局图、LogicLock 功能、“Settings”对话框和 Assignment Editor 改进时序，达到时序逼近。

（15）使用 Assembler 为设计建立编程文件。

（16）使用编程文件、Programmer 和 Altera 硬件对器件进行编程，或将编程文件转换为其他文件格式以供嵌入式处理器等其他系统使用。

（17）使用 SignalTap® II Logic Analyzer、SignalProbe 功能或 Chip Editor 对设计进行调试。

（18）使用 Chip Editor、Resource Property Editor 和 Change Manager 管理工程更改。

5.3 Quartus II 软件的安装与启动

使用 PC 机安装 Quartus II 的系统配置要求如下。

- CPU：奔腾 II 400MHz 以上。
- 内存：512MB 以上。
- 大于 800MB 的安装空间。
- Windows 2000 或 Windows XP 操作系统。
- Microsoft Windows 兼容的 SVGA 显卡。
- 具有并行通信口或 USB 通信口，以便用下载电缆。

Quartus II 软件安装步骤如下。

（1）运行安装向导：Install.exe。

（2）单击 “Install Quartus II and Related Software” 按钮，进入欢迎界面，单击 “Next” 按钮进入安装 Quartus II 软件的安装向导界面。在这个安装向导界面中，选中 Quartus II，其他项目不选，单击 “Next” 按钮，进入下一步。

（3）在 “License Agreement” 对话框上选中 “I accept the terms of the license agreement” 选项，单击 “Next” 按钮，进入下一步。

（4）在 “Custom information” 对话框上，输入用户信息 “User Name：xxx，Company Name：xxx”，单击 “Next” 按钮，进入下一步。

（5）在 “Choose Destination Location” 对话框上选择安装路径，也可使用默认路径。确保硬盘上有足够的空间，单击 “Next” 按钮进入下一步。

（6）在 Setup Type 对话框上选中 “Complete” 选项，单击 “Next” 按钮，进入下一步。

（7）确认安装设置，单击 “Next” 按钮，安装向导开始复相关文件。

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

（8）在“Quartus II Talkback”对话框上，单击“确定”按钮继续安装。

（9）在“Install shield Wizard Complete”对话框上，去掉“Launch Quartus II 6.0”选项，单击“Finish”按钮，完成安装。

（10）回到安装向导主界面，单击“Install FLElm Server”按钮，启动 FLElm Server 的安装向导。在“Choose Destination Location”对话框上，单击“Next”按钮，安装到默认路径。

（11）文档复制完毕，单击“Finish”按钮，完成安装。

（12）回到安装向导主界面，单击“Exit”按钮，退出安装向导。

（13）根据 license 的安装说明安装 license。

（14）至此， Quartus II 6.0 已经可以使用了。

5.4 Quartus II 软件设计流程

利用 Quartus II 进行设计的流程如图 5.3 所示。

这是一个完整的设计流程，在实际的设计过程中，其中的一些步骤可以进行简化。简化的 Quartus II 设计过程如图 5.4 所示。

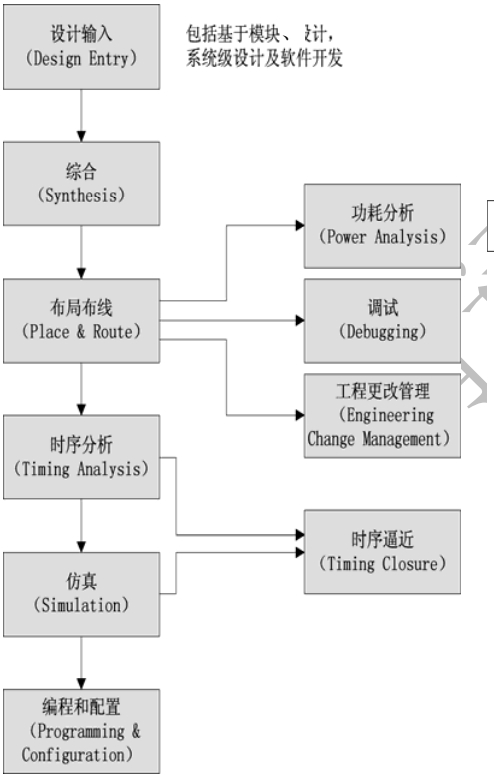


图 5.3 Quartus II 设计流程

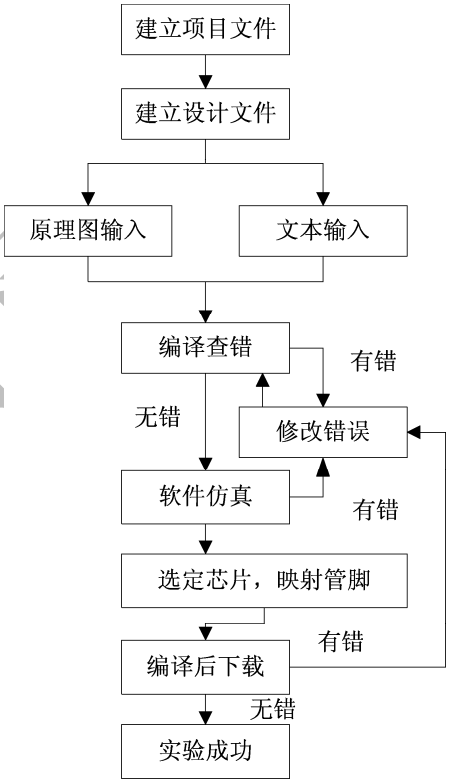


图 5.4 简化的 Quartus II 设计过程

5.5 创建工程设计文件

5.5.1 创建工程

Quartus II 软件将工程信息存储在 Quartus II 工程配置文件中，如表 5.1 所示。它包含有关 Quartus II 工程的所有信息，包括设计文件、波形文件、SignalTap® II 文件、内存初始化文件以及构成工程的编译器、仿真器和软件构建设置。

表 5.1 Quartus II 工程文件

文 件 类 型	说 明

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

Quartus II Project File（.qpf）	指定用来建立和修订工程的 Quartus II 软件版本
Quartus II Setting File（.qsf）	包括 Assignment Editor、Floorplan Editor、“Setting”对话框（“Assignment”菜单），Tcl 脚本或者 Quartus II 可执行文件产生的所有修订范围或者独立的分配。工程中每个修订有一个 QSF
Quartus II Workspace File（.qws）	包含用户偏好和其他信息，例如窗口信息、窗口打开文件及其位置
Quartus II Default Settings File（.qdf）	位于\<Quartus II system directory>\bin 目录下，包括所有全局默认工程设置。QSF 中的设置将替代这些设置

可以使用“File”菜单中的“New Project Wizard”来创建工程。使用“New Project Wizard”可以为工程指定工作目录、分配工程名称以及指定最高层设计实体的名称，还可以指定工程中使用的设计文件、其他源文件、用户库、EDA 工具以及目标器件。

建立工程后，可以使用“Assignments”菜单的“Settings”对话框中的“Add/Remove”页，在工程中添加和删除设计和其他文件。在执行 Quartus II Analysis & Synthesis 期间，Quartus II 软件将按“Add/Remove”页中显示的顺序处理文件。

如图 5.5～图 5.10 所示为使用 New Project Wizard 来创建工程的流程。

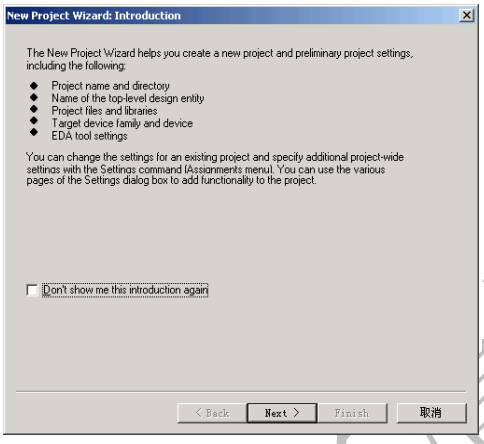


图 5.5　新建工程向导

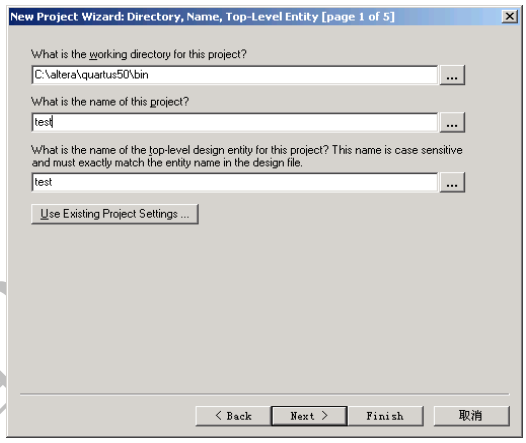


图 5.6　选择工程路径、名、顶层模块名

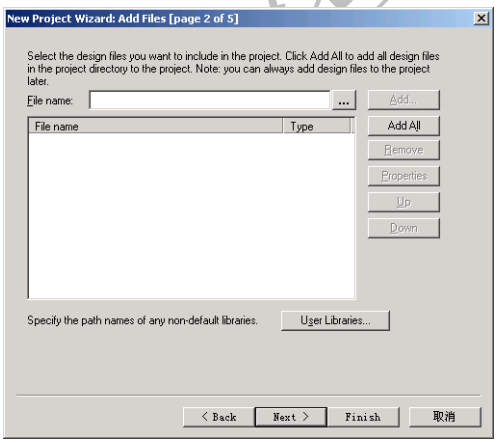


图 5.7　添加设计文件

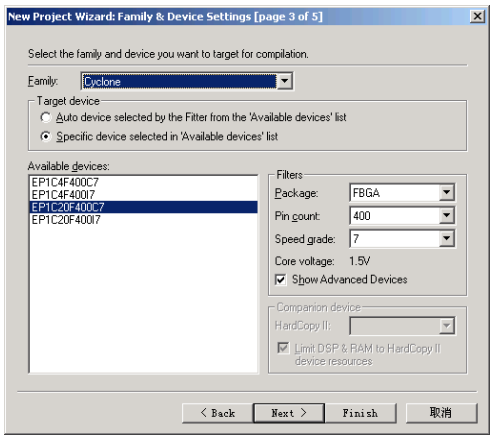


图 5.8　选择目标器件（FPGA 型号）

开发板免费领取试用　开发板免费领取试用

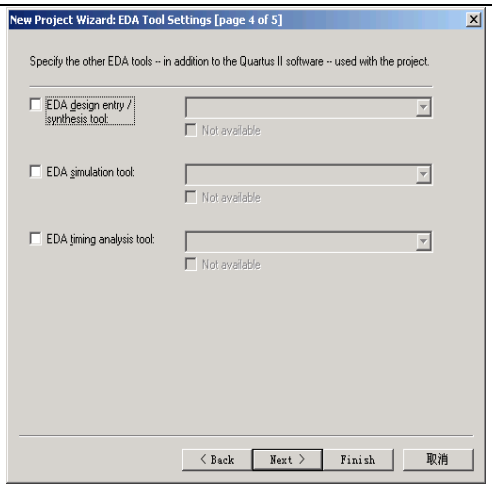


图 5.9　选择 EDA 综合、仿真、时序分析工具

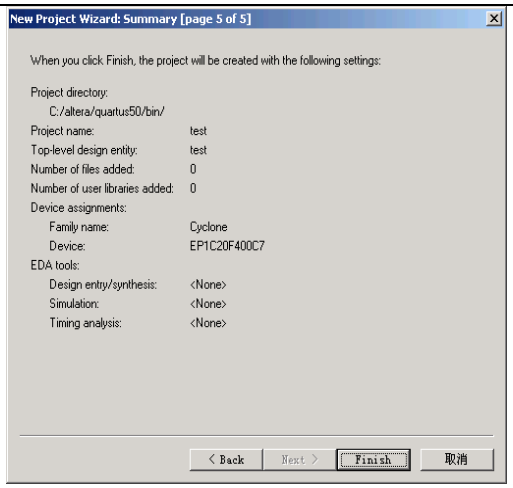


图 5.10　工程信息汇总

如果想通过 Quartus II 来查看和编辑 MAX+PLUS II 工程，还可以使用“File”菜单的“Convert MAX+PLUS II Project”命令，将 MAX+PLUS II 分配与配置文件（.acf）转换为 Quartus II 工程可编辑的文件，如图 5.11 所示。

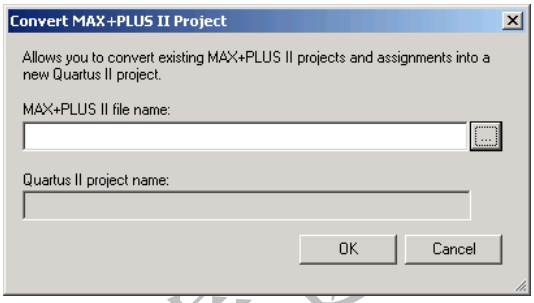


图 5.11　“Convert MAX+PLUS II Project”对话框

5.5.2　添加设计文件

1．添加设计文件方式

Quartus II 软件支持如下几种设计输入方式。

- 在 Block Editor 中建立设计。
- 在 Text Editor 中通过 AHDL、Verilog HDL 或 VHDL 设计语言建立设计。
- 采用 EDA 设计输入和综合工具生成的 EDIF 输入文件（.edf）或 VQM 文件（.vqm）建立设计。

2．支持的设计文件类型

如表 5.2 所示为 Quartus II 软件支持的设计文件类型。

表 5.2　　Quartus II 设计文件类型

类　型	描　述	扩 展 名
原理图设计文件	使用 Quartus II Block Editor 建立的原理图设计文件	.bdf
EDIF 输入文件	使用任何标准 EDIF 网表编写程序生成的 2000 版 EDIF 网表文件	.edf .edif
图形设计文件	使用 MAX+PLUS II Graphic Editor 建立的原理图设计文件	.gdf
文本设计文件	以 Altera 硬件描述语言（HDL）编写的设计文件	.tdf
Verilog 设计文件	包含使用 Verilog HDL 定义的设计逻辑的设计文件	.v .vlg

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

		.verilog
VHDL 设计文件	包含使用 VHDL 定义的设计逻辑的设计文件	.vh .vhd .vhdl
VQM 设计文件	通过 Synplicity Synplify 软件或 Quartus II 软件生成的 Verilog HDL 格式网表文件	.vqm

如图 5.12 是 Quartus II 软件所支持的设备设计文件类型。
如图 5.13 所示是 Quartus II 软件所支持的软件文件类型。

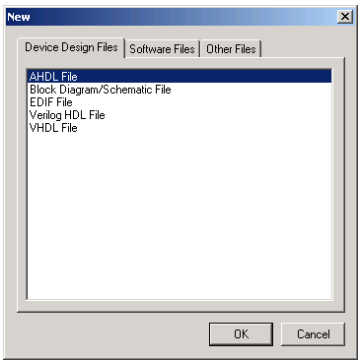


图 5.12 “Device Design Files”对话框

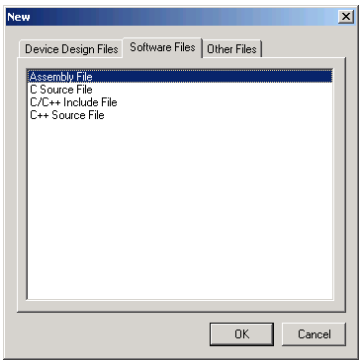


图 5.13 “Software Files”对话框

如图 5.14 所示是 Quartus II 软件所支持的其他文件类型。

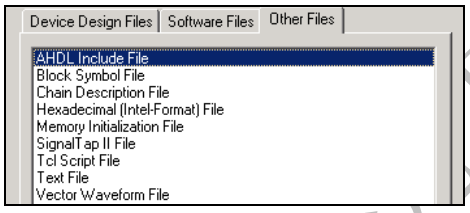


图 5.14 其他文件类型

3．使用 Block editor 设计文件

Block Editor 用于以原理图和流程图的形式输入和编辑图形设计信息。Quartus II Block Editor 可以读取并编辑原理图设计文件和 MAX+PLUS II 图形设计文件。也可以在 Quartus II 软件中打开图形设计文件并将其另存为原理图设计文件。
Block Editor 将每个流程图、原理图或符号代表的设计逻辑融合到工程中。这些图形设计文件包含块和符号，这些块和符号代表设计中的逻辑。
可以使用原理图设计文件的块建立新设计文件，或在修改块和符号时更新设计文件，也可以在原理图设计文件的基础上生成块符号文件（.bsf）、AHDL 包含文件（inc）和 HDL 文件。
还可以在编译之前分析原理图设计文件是否出错。Block Editor 还提供有助于在原理图设计文件中连接块和基本单元（包括总线和节点连接以及信号名称映射）的一组工具。可以更改 Block Editor 的显示选项，例如根据偏好更改导向线和网格间距、橡皮带式生成线、颜色和屏幕元素、缩放以及不同的块属性等。
Block Editor 具备以下的功能。

- 插入块和基本单元符号。流程图使用称为块的矩形符号代表设计实体以及相应的已分配信号，在从上到下的设计中很有用。块是用代表相应信号流程的管道连接起来的。可以只用流程图进行设计，也可以将流程图与图形单元相结合进行设计。

从块或原理图文件建立文件。在层次化设计中，可以在使用 Block Editor 中的 Create/Update 命令，利用原理图文件中的块建立其他类型的设计文件，还可以从原理图文件本身建立这些设计文件。

4．使用 Text Editor 设计文件

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

Text Editor 是一个灵活的工具，用于以 AHDL、VHDL 和 Verilog HDL 语言以及 Tcl 脚本语言输入文本型设计。还可以使用 Text Editor 输入、编辑和查看其他 ASCII 文本文件，包括为 Quartus II 软件或由 Quartus II 软件建立的那些文本文件。

5．使用 Symbol Editor 设计文件

Symbol Editor 用于查看和编辑代表宏功能、宏功能模块、基本单元或设计文件的预定义符号。每个 Symbol Editor 文件代表一个符号。对于每个符号文件，均可以从包含 Altera 宏功能模块和 LPM 函数的库中选择。可以自定义这些块符号文件，然后将这些符号添加到使用 Block Editor 建立的原理图中。Symbol Editor 读取并编辑块符号文件和 MAX+PLUS II 符号文件（.sym），并将它们转存为块符号文件。

6．使用 Verilog HDL、VHDL 与 AHDL 设计文件

可以使用 Text Editor 或其他文本编辑器建立文本设计文件、Verilog 设计文件和 VHDL 设计文件，并在分级设计中将这些文件与其他类型的设计文件相结合。Verilog 设计文件和 VHDL 设计文件可以包含由 Quartus II 支持的构造的任意组合。

它们还可以包含 Altera 提供的逻辑函数，包括基本单元和宏功能模块以及用户自定义的逻辑函数。在 Text Editor 中，使用 Create/Update 命令（“File”菜单）从当前的 Verilog HDL 或 VHDL 设计文件建立块符号文件，然后将其合并到原理图设计文件中。

同样，可以建立代表 Verilog HDL 或 VHDL 设计文件的 AHDL 包含文件，并将其合并到文本设计文件中或另一个 Verilog HDL 或 VHDL 设计文件中。

AHDL 是一种完全集成到 Quartus II 系统中的高级模块化语言。AHDL 支持布尔等式、状态机、条件逻辑和解码逻辑。AHDL 还可用于建立和使用参数化函数，并完全支持 LPM 函数。AHDL 特别适合设计复杂的组合逻辑、批处理、状态机、真值表和参数化逻辑。

7．使用宏功能模块设计文件

Altera 宏功能模块是复杂的高级构建模块，可以在 Quartus II 设计文件中与逻辑门和触发器基本单元一起使用。Altera 提供的参数化宏功能模块和 LPM 功能均为 Altera 器件结构做了优化。

必须使用宏功能模块才可以使用一些 Altera 专用器件的功能，例如，存储器、DSP 块、LVDS 驱动器、PLL 以及 SERDES 和 DDIO 电路。可以使用 Mega Wizard Plug-In Manager（“Tools”菜单）建立 Altera 宏功能模块、LPM 功能和 IP 功能，用于 Quartus II 软件和 EDA 设计输入与综合工具中的设计。

表 5.3 列出了能够由 Mega Wizard Plug-In Manager 建立的 Altera 提供的宏功能模块和 LPM 功能类型。

表 5.3　　Altera 提供的宏功能模块与 LPM 功能表

类　型	说　明
算术组件	包括累加器、加法器、乘法器和 LPM 算术功能
逻辑门	包括多路复用器和 LPM 门功能
I/O 组件	包括时钟数据恢复（CDR）、锁相环（PLL）、双数据速率（DDR）、千兆收发器块（GXB）、LVDS 接收器和发送器、PLL 重新配置和远程更新宏功能模块
存储器编译器	包括 FIFO Partitioner、RAM 和 ROM 宏功能模块
存储组件	存储器，移位寄存器宏功能模块和 LPM 存储器功能

5.6　编译及仿真工程

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

5.6.1 编译

编译的步骤如下。

（1）首先把顶层模块设置为 Top-Level Entry，如图 5.15 所示。

（2）选择“processing”菜单的“Start Compilation”选项进行编译，如图 5.16 所示。

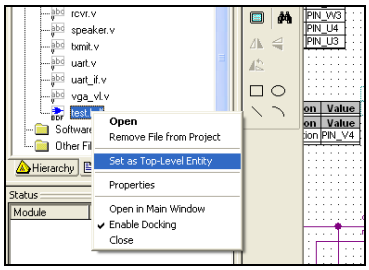


图 5.15 设置顶层模块

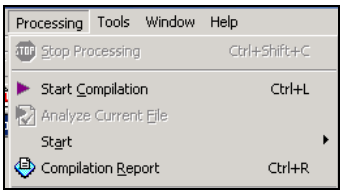


图 5.16 开始编译

（3）查看“message”栏中的错误，改正直到没有错误为止。

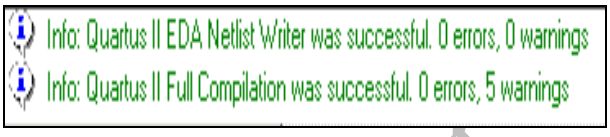


图 5.17 编译无错误

5.6.2 仿真

可以使用 Quartus II Simulator 在工程中仿真任何设计。根据所需的信息类型，可以进行功能仿真以测试设计的逻辑功能，也可以进行时序仿真。在目标器件中测试设计的逻辑功能和最坏情况下的时序，或者采用 Fast Timing 模型进行时序仿真，在最快的器件速率等级上仿真尽可能快的时序条件。

下面是利用 Quartus II Simulator 进行仿真的步骤。

（1）设置顶层模块。

Quartus II 软件可以仿真整个设计，也可以仿真设计的一部分。通过指定工程中的某一个设计实体为顶层设计实体，即可仿真该顶层实体及其所有附属设计实体，如图 5.18 所示。

（2）建立波形文件。

选择新建其他文件对话框的“Vector Waveform File”建立.vmf 文件，如图 5.19 所示。

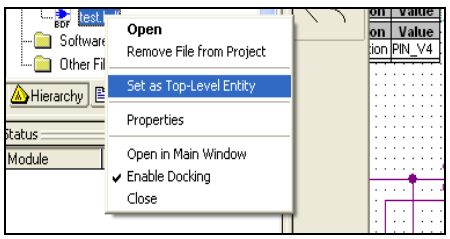


图 5.18 设置顶层模块

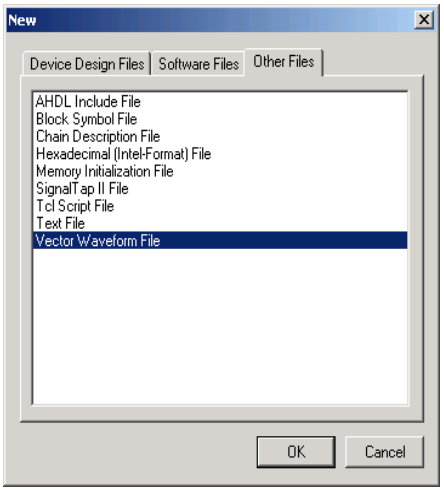


图 5.19 建立仿真波形文件

（3）添加观察信号。

通过图 5.20 ~ 图 5.22 向新建的波形文件添加观察信号。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

首先通过双击图 5.20 的空白区域添加观察信号。

然后在弹出的对话框中填写要插入的信号名，如图 5.21 所示。

图 5.20　在仿真界面中添加观察信号

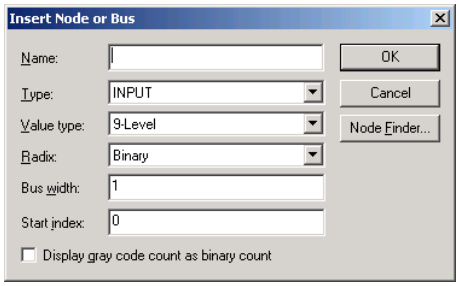


图 5.21　插入信号对话框

也可以使用 Node Finder 选择指定的信号，如图 5.22 所示。

（4）添加激励。

通过拖曳波形，产生想要的激励输入信号。可以通过如图 5.23 所示的工具条为波形图添加信号。

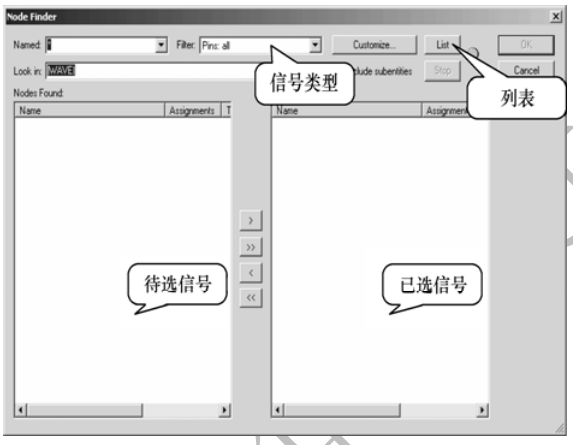


图 5.22　“Node Finder”对话框

图 5.23　波形控制工具条

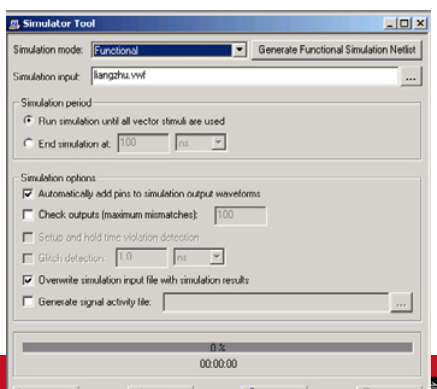
如图 5.24 所示是添加激励后的波形示例。

图 5.24　波形示例

（5）功能仿真。

添加完激励信号后，保存波形文件。选择工具栏中“Simulator Tool”工具进行仿真，如图 5.25 所示。

首先做功能仿真，先单击“Generate Functional Simulation Netlist”按钮产生仿真需要的网表文件，然后把“Overwrite simulation input file with simulation result”选中（否则不能显示仿真结果），单击“Start”按钮进行仿真。



仿真完成后，单击“Open”按扭打开仿真结果，如图 5.26 所示。

（6）时序仿真。

功能仿真正确后，可以加入延时模型，进行时序仿真。将 Simulation mode 选为 Timing 模式，如图 5.27 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

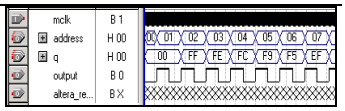


图 5.26　仿真结果

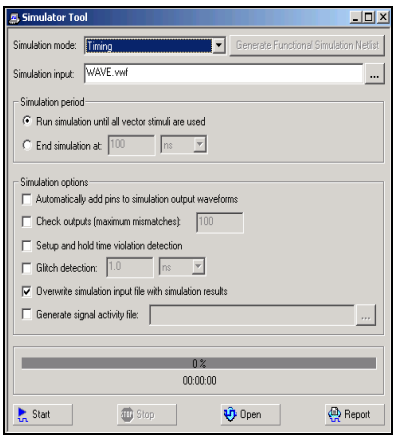


图 5.27　仿真模式设置

仿真结束后，输出的波形会出现延时，如图 5.28 所示。

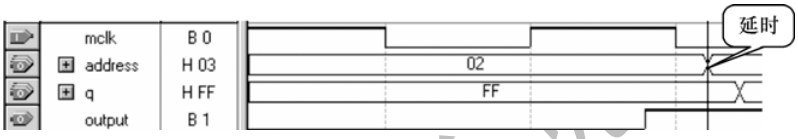


图 5.28　时序仿真结果的延时

5.7　约束及配置工程

设计好工程文件后，首先要进行工程的约束。约束主要包括器件选择、管脚分配及时序约束等。时序约束属于较为高级的应用，通过时序约束可以使工程设计文件的综合更加优化。下面对这几种约束方式进行介绍。

5.7.1　器件选择

选择“Assignments”菜单中的“Device”选项，如图 5.29 所示。

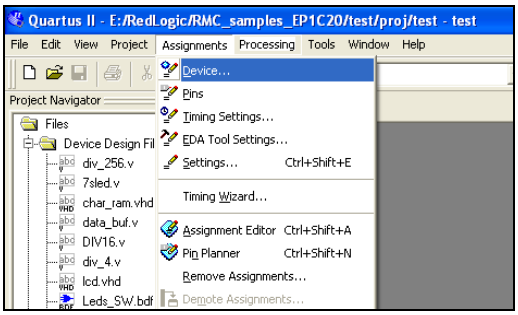


图 5.29　选择器件

在打开的器件设置对话框中，选用 cyclone 系列 ep1c12Q240c8（或 ep1c6Q240c8）型号 FPGA，速度等级为 8，如图 5.30 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

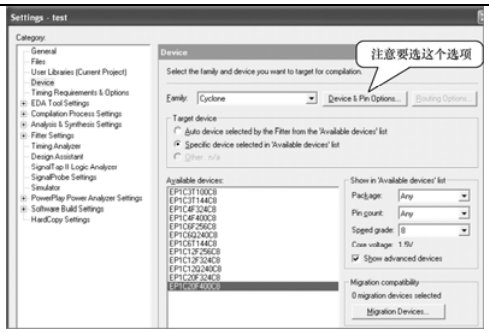


图 5.30　器件选择设置对话框

除了选择器件的型号外，还要注意要选择“Device & Pin Options…”选项，打开器件配置对话框。在“General”选项卡中设置配置的方式（AS 模式或 JTAG 模式），并选择配置器件，如 EPCS4，如图 5.31 所示。
在“Unused Pins”选项卡中可以设置未使用管脚的工作状态，如设置成输入三态，如图 5.32 所示。

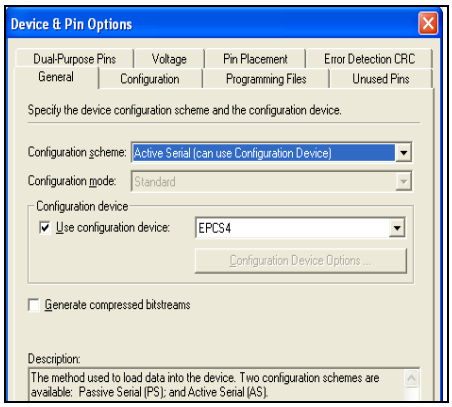


图 5.31　设置配置模式

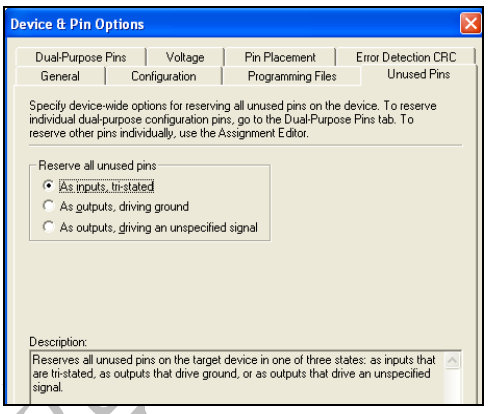


图 5.32　未使用管脚设置

5.7.2　管脚分配

选择器件后，就可以为设计文件进行管脚分配了。选择“Assignments”菜单中的“Pins”选项，打开管脚分配器，如图 5.33 所示。

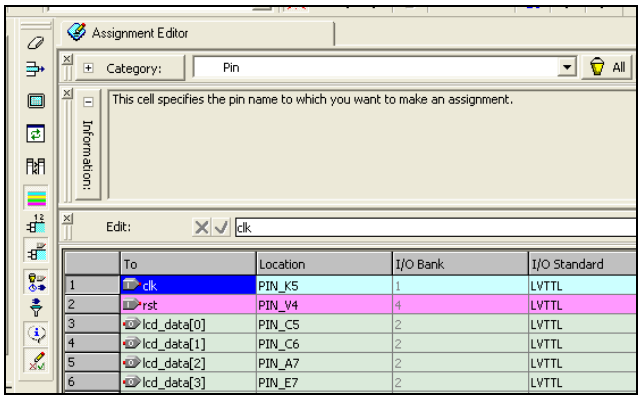


图 5.33　管脚分配器

已经编译过的所有输入输出信号都会自动显示在分配列表中，无需手动添加。按照用户的设计需求进行管脚定义，然后再编译整个工程文件，即可进行下载。

5.7.3　时序约束

分配管脚后，在执行完全编译（Full Compilation）之前，可以利用“Assignments Editor”和“Settings”菜

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

单的“Timing Requirements & Options”对系统信号的时序特性进行设置，也可以使用“Timing wizard”向导来设置时序约束条件。

设置时需约束后，经过完全编译，Timing Analyzer 即可对设计自动进行时序分析。也可以在完全编译期间进行时序分析，或在初始编译后单独进行时序分析。然后使用报告窗口、时序逼近布局图和 list_path_tcl 命令查看时序结果。

时序约束要求允许为整个工程、特定设计实体或个别实体、节点引脚等设置所需的速度性能。如果未指定时序约束条件，Timing Analyzer 将使用默认设置运行分析。

一般的时序约束包括最大频率、建立时间、保持时间、时钟至输出延时、引脚至引脚延时以及最低时序要求等。还可以对多个时钟域、路径切割选项和默认外部延时等进行设置。下面是几种常见约束的定义。

- fmax（最大频率）：在不违反内部建立时间（tsu）和保持时间（th）要求下可以达到的最大时钟频率。
- tsu（时钟建立时间）：在触发寄存器计时的时钟信号已经在时钟引脚确立之前，经由数据输入或使能端输入而进入寄存器的数据必须在输入引脚处出现的时间长度。
- th（时钟保持时间）：在触发寄存器时钟信号已经在时钟引脚确立之后，经由数据输入或使能端输入而进入寄存器的数据必须在输入引脚处保持的时间长度。
- tco（时钟至输出延时）：时钟信号在触发寄存器的输入引脚上发生转换之后，再由寄存器馈送信号的输出引脚上取得的有效输出所需的时间。
- tpd（引脚至引脚延时）：指定可接受的最少的引脚至引脚延时，引脚处信号通过组合逻辑进行传输并出现在外部输出引脚上所需的时间。

运行时序分析或者完整编译之后，在“Processing”菜单中选择“Compilation Report”选项，在弹出的报告栏中选择“Timing Analyzer”即可产看分析结果。

5.7.4 配置工程

下面来介绍最常用的两种 FPGA 配置模式。

1．JTAG 模式

JTAG 模式直接将逻辑下载至 FPGA。这种模式下载速度快，但掉电即失，适合进行电路调试时使用，下载文件类型为 sof 文件。

首先确认电缆已连接到主机与 FPGA 实验平台的 JTAG 插座上，选择“Tools”菜单的 Programmer 下载器，如图 5.34 所示。

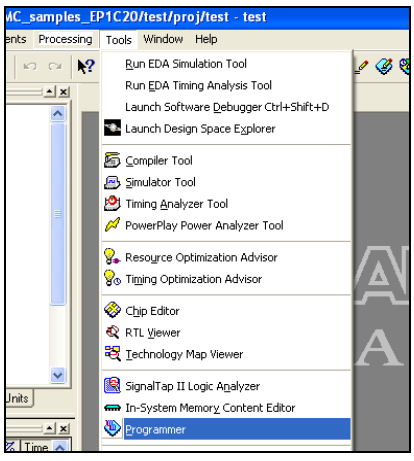


图 5.34　选择下载器

打开如图 5.35 所示的下载器后，通过“Hardware Setup”按钮选择下载电缆（如 USB、LPT、ByteBlasterII 等）。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

然后选择 JTAG 下载模式，并把“Program/Configure”勾上，即可进行下载。

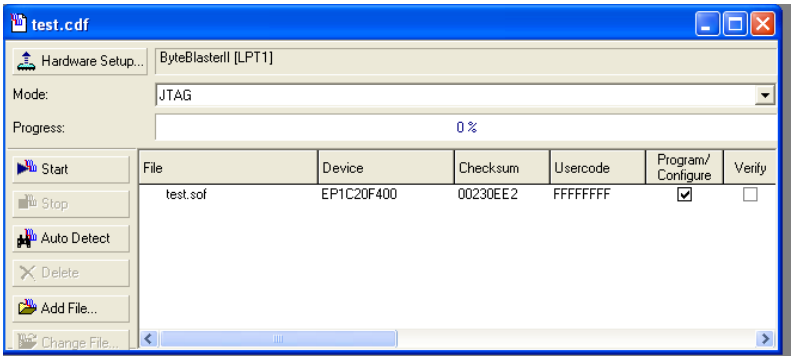


图 5.35　下载器对话框

2．AS（Active Serial）模式

AS 模式将逻辑下载至配置芯片上（如 EPCS4）。这种模式下载速度相对较慢，但掉电不失，适合调试结束后固化最后版本逻辑时使用，下载文件类型为 pof 文件。

首先确认电缆已连接至 AS 插座上。选择“Tools”菜单的 Programmer 下载器，同样选择电缆后，选用 AS 下载方式，再将“Program/Configure”勾上，即可下载，如图 5.36 所示。

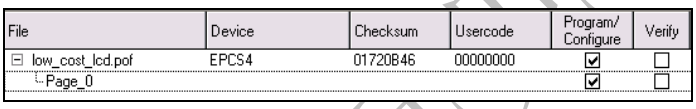


图 5.36　AS 下载模式

5.8　LogicLock 逻辑锁定工具使用技巧

5.8.1　逻辑锁定方法学

逻辑锁定方法学（LogicLock Methodology）内容就是在设计时采用逻辑锁定的基于模块设计流程（LogicLock block-based design flow），来达到固定单模块优化的目的。这种设计方法学中第一次引入了高效团队合作方法：它可以让每个单模块设计者独立优化他的设计，并把所用资源锁定。

这样在合成顶层设计时就可以保持每个模块的性能，而且它还让逻辑模块可重复使用，提高了资源利用率，缩短了设计周期。逻辑锁定的基于模块设计流程与传统设计流程对比如图 5.37 所示。

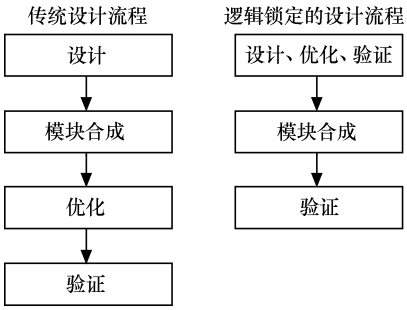


图 5.37　两种设计流程比较

5.8.2　逻辑锁定优势

性能不变，而设计周期更短。使用 LogicLock 约束，逻辑模块只需优化一次。通过约束达到性能要求后，LogicLock 区域的回传机制将此性能指标保持不变。例如，设计人员可以在设计中使用一个 PCI 宏函数，锁定布局和性能指标。在设计中，加入其他逻辑模块或者在其他区域中改变逻辑都不会影响 PCI 核的性能。LogicLock 流程减少了完成一个工程所需的设计周期数量。

更强的分层次设计流程。设计人员通过 LogicLock 流程，可以保持每个模块的特性不变，通过分层次区域实施很快地达到性能指标。采用 LogicLock 方法学，设计人员或者设计团队可以在设计中独立工作，完成布局布线，优化和对每个设计模块的验证。在集成阶段，只需对整个系统进行验证。布局和每个模块的性

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

能保持不变。

优化的设计重复使用流程。LogicLock 约束可以由外部输入。由于 LogicLock 方法允许将单个约束过的设计模块输入到多个设计中去，因此设计可以方便地重复使用。如图 5.38 所示为工程师独立开发的模块在多个工程中使用的例子，模块在每个实施中都有相同的性能。

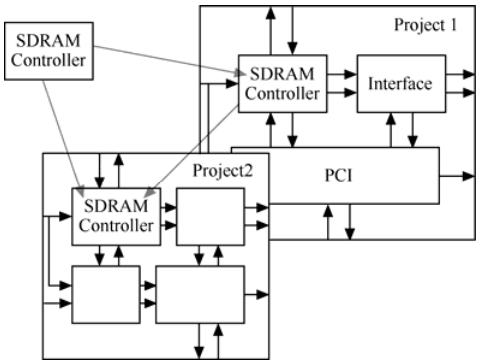


图 5.38　LogicLock 设计的重复使用

5.8.3　逻辑锁定参数设置

LogicLock 区域参数设置在 LogicLock 区域中，设计人员可以控制布局。这些区域为真正的分层次设计提供基本构架，设计中改变一个模块不会影响到区域中其他模块的性能。表 5.4 所示为控制 LogicLock 区域行为的各种参数含义。默认的选项分别是浮动、自动、关闭和硬区域。

表 5.4　　LogicLock 区域参数

参　数	选　项	特　征
状态	浮动、锁定	浮动区允许 Quartus II 软件决定器件上的区域位置。锁定区是区域中用户定义的位置，在图形平面配置图上以实线边界标出，锁定区必须有固定的大小
大小	自动、固定	自动大小区允许 Quartus II 软件决定区域的合适大小，固定区域含有用户定义的形状和大小
保留	关闭、打开	保留特性允许用户定义区域中的资源能否用于没有分配给区域的实体。如果保留打开，则只有分配给区域的项目才可以在区域内布放
执行	硬、软	软区域有更宽松的时间约束，允许总设计的性能提高后，将一些实体置于区域之外。硬区域不允许将内容放置在区域边界之外
原点	平面配置图的任一位置	原点定义为 LogicLock 区域放置在平面配置图上时其左上角位置

5.8.4　逻辑锁定流程

首先，分析整体资源利用率，如果采用的 FPGA 芯片容量足够，理论上只要保证重点模块的资源利用率，就能保证整体设计的性能；然后采用逻辑锁定的基于模块设计流程（LogicLock block-based design flow）进行设计，步骤如下。

（1）在 Quartus®中综合单个底层模块。

（2）优化重点模块，进行逻辑锁定。

（3）反标（back-annotaing）。

（4）导出模块逻辑锁定约束信息，包括原级网表（atom netlist）文件（.vqm）、布局信息（placement information）文件（qsf）和布线信息（routing information）文件（.rcf）。

（5）将这些约束文件导入顶层（top-level）工程中。

（6）编译和验证整个顶层设计。

（7）编译完成后，查看时序分析界面是否达到要求。

以上介绍的逻辑锁定的具体方法可以参照 5.11 小节。

通过应用一种新的设计方法学——逻辑锁定方法学（LogicLock Methodology），采用了模块化、团队化的设计流程，对重点模块进行优化，解决了传统设计流程无法解决的问题，同时对其他 FPGA 设计工程中类似的问题提供了可借鉴的思路。

5.9　SignalTap II 在线逻辑分析仪的使用方法

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

随着设计复杂度的增加，传统的设计验证方法需要其他技术和工具的补充，因为这些可编程芯片系统（SOPC）进行完全的真实模拟是不可在适当的时间内完成。而且，设计规模的陡增必然需要新的工具来观察已编程期间的内部操作。尤其是随着第三方 IP 使用的增加，它们需要获取内部探测来验证操作，使其和设计的其他部分相分离。最后，由于封装技术的提高，必须开发新的方法以便对日趋小型化和大规模封装的硬件（如 BGA）进行验证。

5.9.1 SignalTap II 介绍

Quartus II 5.0 支持 SignalTap II，满足这些需求。SignalTap 允许设计者在 FPGA 运行期间同时监视内部信号。通过下载电缆或传统的分析设备连接到用户的 PC 板卡上，便可以观察到这些信号的波形。使用 SingnalTap 就类似于使用逻辑分析仪，能够设置初始化、触发（内部或外部）和显示条件以及观察的内部信号，用户以此可以研究设计的运行状态。用户的分析参数可以被编译为嵌入逻辑分析仪（ELA），它和设计的其他数据一起配置 FPGA。Altera 全系列 FPGA 器件支持 SignalTap，采用 Byteblaster II 或者 USB blaster 作为器件的下载电缆。

若没有采用 SignalTap 接口，用户必须更改设计以探测内部逻辑的连线。设计的内部连线必须连接到顶层设计的管脚上。如果结点处于庞大分级设计的下层，那么改起来很复杂，同时很耗时，而且破坏了设计的完整性。ELA 接口支持拖放选择用于逻辑分析的连线。这个接口根本就无需改变设计。选择了 ELA 的输入通道之后，需要重新编译设计把 ELA 配置加入期间配置文件中。重新编译只是把一个 ELA 实例添加到现有的设计中，而无需改变已有的设计。更新后的配置文件重新配置器件后，标准逻辑分析仪就会可以检测那些被连接到器件管脚的内部信号了。

输入通道的样值存储在器件的嵌入存储块内，ELA 功能监测输入通道是否发生触发事件。一旦 ELA 存储了满足触发状态的足够数据，ELA 停止采样监测输入通道。然后数据上载到主机，显示在 Quartus 的波形编程器中。数据的主载速率取决于 JTAG TCK 信号的速率。ELA 功能会使用设计本身占用以外的器件资源。ELA 是可参数化的，因此能够使用有效的资源。

SignalTap II 支持以用户指定的格式识别和显示总线使所捕获的数据更加易懂。SignalTap II 嵌入式逻辑分析仪能够以等价的十六进制、无符号十进制、二元补码形式的符号十进制，符号大小表示法表示的符号十进制、八进制、二进制、8 比特 ASCII 等格式来显示总线。用户还可以选择条形图或者线性图表示总线时间关系。

SignalTap II 支持多文件输出数据结果，嵌入式逻辑分析仪可以采用矢量波形（vwf）、矢量表（tbl）、矢量文件（vec）、逗号分割数据（csv）和 Verilog 数值更改转存（vcd）文件格式输出所捕获的数据。这些文件格式可以被第三方验证工具读入，显示和分析 SignalTap II 嵌入式逻辑分析仪所捕获的数据。

5.9.2 使用 SignalTap II 操作流程

若要使用 SignalTap II 逻辑分析器，必须先建立 SignalTap II 文件（stp）此文件包括所有配置设置并以波形显示捕获到的信号。一旦设置了 SignalTap II 文件，就可以编译工程，对器件进行编程并使用逻辑分析器采集和分析数据。

以下步骤描述设置 SignalTap II 文件和采集信号数据的基本流程。

（1）建立新的 SignalTap II 文件。

（2）向 SignalTap II 文件添加实例，并向每个实例添加节点。可以使用 Node Finder 中的 SignalTap II 滤波器查找所有预综合和布局布线后的 SignalTap II 节点。

（3）分配一个采样时钟。

（4）设置其他选项，例如采样深度和触发级别等。

（5）完全编译工程文件。

（6）下载程序到 FPGA 中。

（7）运行硬件并打开 SignalTap II 观察信号波形。

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

5.9.3 设置触发器

如图 5.39 所示是 SignalTap II 逻辑分析仪的界面。其中实例管理器对话框识别出设计中

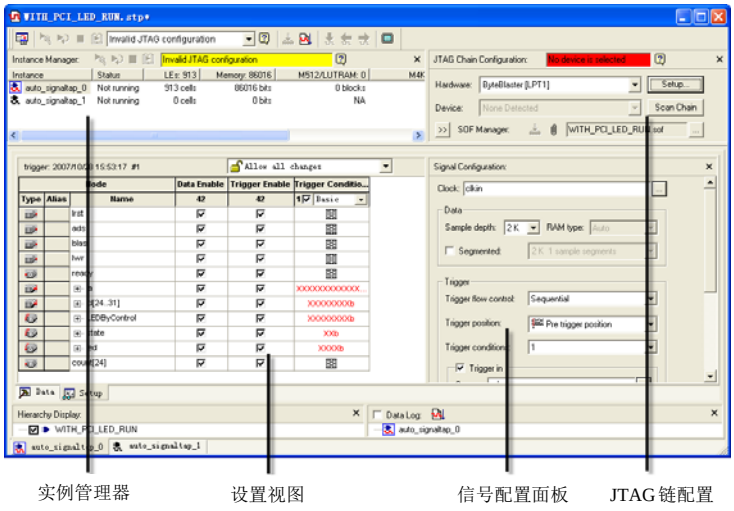


图 5.39 SignalTap II 逻辑分析仪

等待测试的所有验证过的逻辑分析仪，它们可以用来捕获并存储数据。该对话框还可以对用来生成每个分析仪的资源进行估算。

信号配置面板用于设置采样信号和触发信号。采样时钟信号支持超过 200MHz 频率，采样深度最大高达 128K。

数字示波器或逻辑分析仪中，触发器是一个重要的组成部分。触发器的性能很大程度上决定了仪器的性能。触发器是逻辑级别、逻辑边缘触发方式、逻辑样本等逻辑事件的组合。通过不同的触发方式实现对信号的不同采样。

SignalTap 接口定义的触发事件中，每个事件的输入通道可以监测 10 个基本或高级的触发级别。触发级别向 SignalTap II 逻辑分析仪指明何时开始采集数据，10 个触发级别为设置复杂的触发条件提供了足够的灵活性，帮助验证工程师分离错误或者找出问题原因，如果设置了多触发级别，直到所有的触发条件顺序满足后，才开始采集数据。

触发位置允许指定在选定实例中在触发器之前和触发器之后应采集的数据量。分段的模式允许通过将存储器分为周密的时间段，为定期事件捕获数据，而无需分配很深的采样深度。其中对环形缓冲模式支持 4 个触发位置；这样，当触发条件满足后，用户可以更多地控制应该捕获并显示什么样的数据。

- “前”触发位置向软件表明，在达到触发条件前，保存所发生采样的 12%，达到触发条件后，再保存采样的 88%。
- “中”触发位置向软件表明，在达到触发条件前，保存所发生采样的 50%，达到触发条件后，再保存采样的 50%。
- “后”触发位置向软件表明，在达到触发条件前，保存所发生采样的 88%，达到触发条件后，再保存采样的 12%。
- “连续”触发位置向软件表明，以环形缓冲的方式进行连续采样保存，直到用户中断为止。

5.10 典型实例 9：SignalTap II 功能演示

5.10.1 实例的内容及目标

1．实例的主要内容

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

本节旨在通过给定的工程实例——“正弦波发生器”来熟悉 Altera Quartus II 高级调试功能 SignalTap II 和 Intent Memory Content Editor 的使用方法。同时使用基于 Altera FPGA 的开发板将该实例进行下载验证，完成工程设计的硬件实现。在本节中，将主要讲解下面知识点。

- Quartus II 原理图输入方式。
- Quartus II 约束设计。
- Quartus II 工程编译。
- Quartus II 硬件下载。
- SignalTap II 在线调试。
- Intent Memory Content Editor 在线修改片上 ROM 数据。

通过这些知识点，按照实例的流程，读者可以迅速地掌握使用 Quartus II 软件高级调试技巧。

2．实例目标

通过详细的流程讲解，读者应达到下面的目标。

- 熟悉 Quartus II 原理图设计方法。
- 熟悉 SignalTap II 在线调试方法。
- 熟悉 Intent Memory Content Editor 在线修改片上 ROM 数据的方法。

5.10.2 实例详解

本实例将使用图解的方式将整个流程一步一步展现给读者，使读者能够轻松掌握开发的流程。

5.10.2.1 工程系统框图

本实例使用 Quartus II 自带的宏模块（MegaWizard Plug-in Manager）来设计逻辑功能，并使用嵌入式硬件逻辑分析仪观察结果。

正弦函数发生器的原理比较简单，硬件实现也比较简单：首先设计一个 ROM 用来存放正弦函数的幅度数据；用一个计数器来指定 ROM 地址（也就是相位）的增加，输出相应的幅度值。这样在连续的时间内显示的就是一个完整的正弦波形。

系统的电路图如图 5.40 所示。

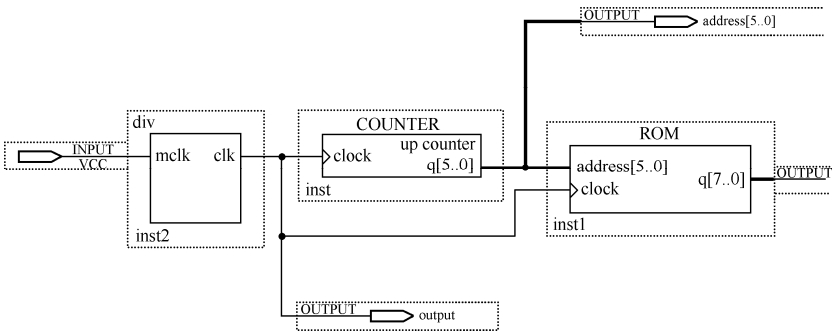


图 5.40 系统电路图

div 模块是分频模块，可以用前面设计过的分频器的程序生成模块（symbol），也可以使用 Altera 的计数器宏模块来生成分频器。

5.10.2.2 添加分频器模块

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

1．将分频文件添加到工程

将分频器模块 div.v 加入到工程中，在工程中打开分频器模块 div.v（分频器程序由读者完成），如图 5.41 所示。

2．产生.bsf 文件

选择“File”/“Create/Update”/“Create Symbol Files for Current File”选项。软件会自动分析 div.v 程序的语法错误，如图 5.42 所示。

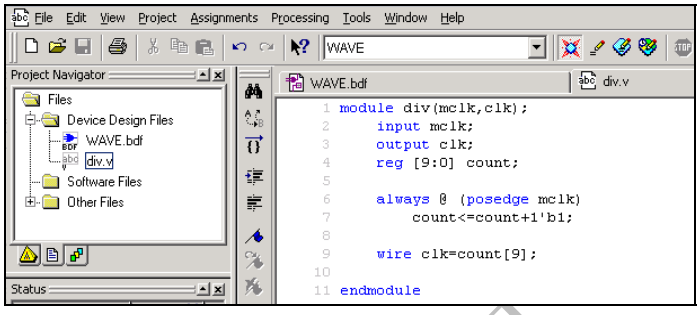


图 5.41　分频器 Verilog HDL 文件

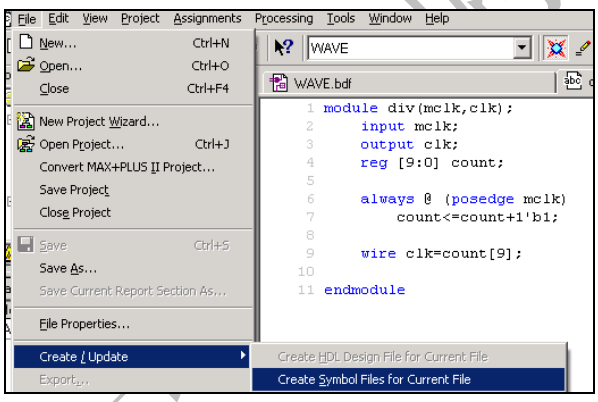


图 5.42　生成模块菜单

3．修改程序语法错误

如果出错则修改错误，直到没有错误为止。此时软件会自动生成 div.bsf 文件，此文件为原理图中的模块图形。

4．在原理图中加入分频器模块 div.bsf

在原理图文件中单击“”按钮，就可以加入刚刚建立的模块，如图 5.43 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

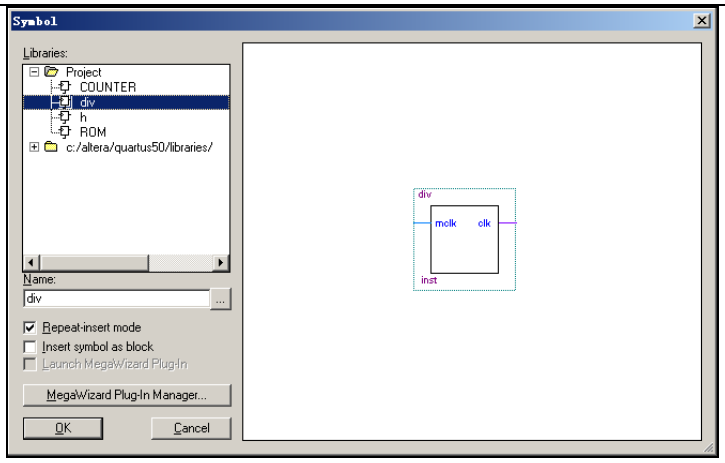


图 5.43　在原理图中添加分频器模块

5.10.2.3　添加 Quartus II 系统自带宏模块

ROM 和计数器都可以通过 Quartus II 自带的宏模块生成。

1．建立 ROM 初始化文件

在菜单中选择新建文件，在“Other Files”中选择“Memory Initialization File”选项，如图 5.44 所示。填入如图 5.45 所示的正弦波幅度数据。当然也可以用 MATLAB、C++或 Excel 的函数生成数据。保存文件，命名 sin.mif。

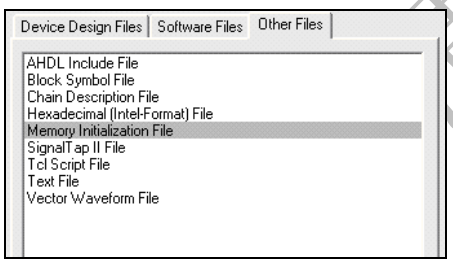


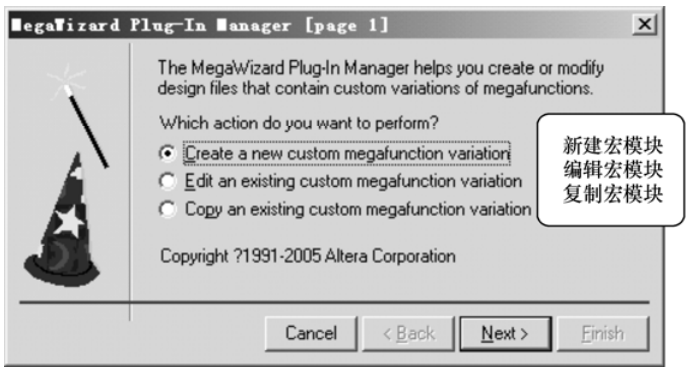
图 5.44　新建 ROM 初始化文件

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	255	254	252	249	245	239	233	225
8	217	207	197	186	174	162	150	137
16	124	112	99	87	75	64	53	43
24	34	26	19	13	8	4	1	0
32	0	1	4	8	13	19	26	34
40	43	53	64	75	87	99	112	124
48	137	150	162	174	186	197	207	217
56	225	233	239	245	249	252	254	255

图 5.45　输入正弦波幅度数据

2．添加自带宏模块

在“Tools”菜单中选择“MegaWizard Plug-Pn Manager”选项，出现如图 5.46 所示的对话框，选择新建宏模块单击“Next”按钮。



华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

图 5.46　引用自带宏对话框

图 5.47 左侧是 Quartus II 自带的免费的宏模型，可以看到 Quartus 提供了很多免费的 IP 核。其中 ROM 在 storage 目录中，计数器 counter 在 arithmetic 目录中。

3．添加 ROM

在 storage 中选择 ROM，并在右侧的选项栏中选择 Cyclone 系列芯片，output file 类型选择 Verilog HDL，并在下方选择 ROM 文件的生成地址及名称，如图 5.48 所示。

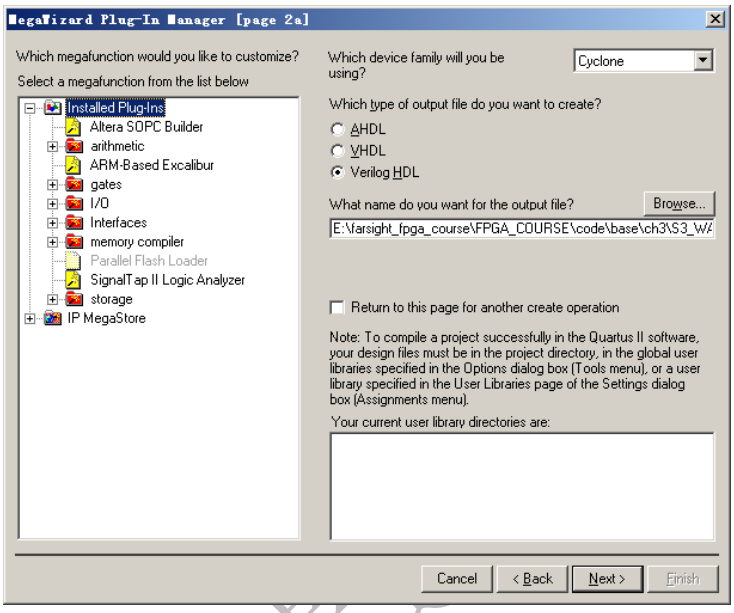


图 5.47　选择系统自带宏模块

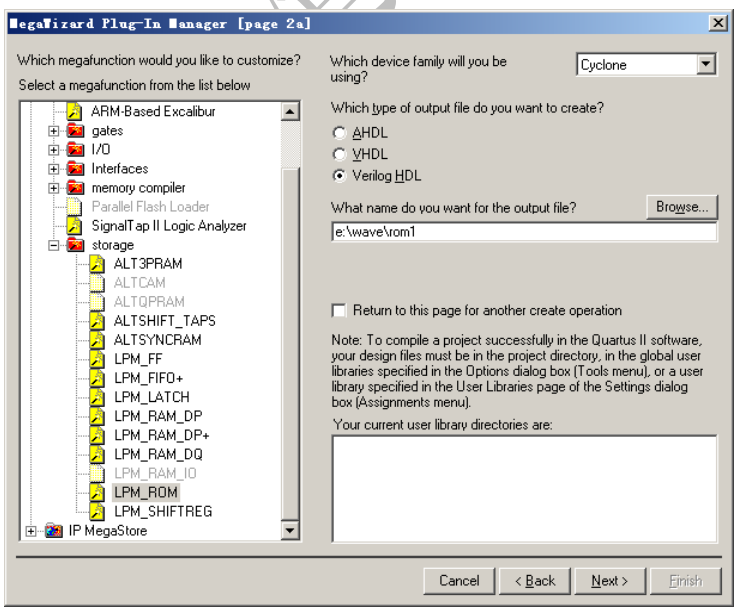


图 5.48　选择 ROM 宏模块

单击“Next”按钮，设定 ROM 基本参数，包括 ROM 的位宽 q、地址位宽（存储深度）和时钟。一般 ROM 的时钟选为单时钟控制 Single clock，如图 5.49 所示。

单击“Next”按钮，设定 ROM 其他参数，设定输出寄存器、时钟使能端和异步清零。如果选中“'q' output port”，则会在输出端加一级寄存器，如图 5.50 所示。

单击“Next”按钮，接下来在对话框中填入 ROM 初始化文件，并选中“allow in-system memory”选项，

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

并选择名称为“NONE”，如图 5.51 所示。

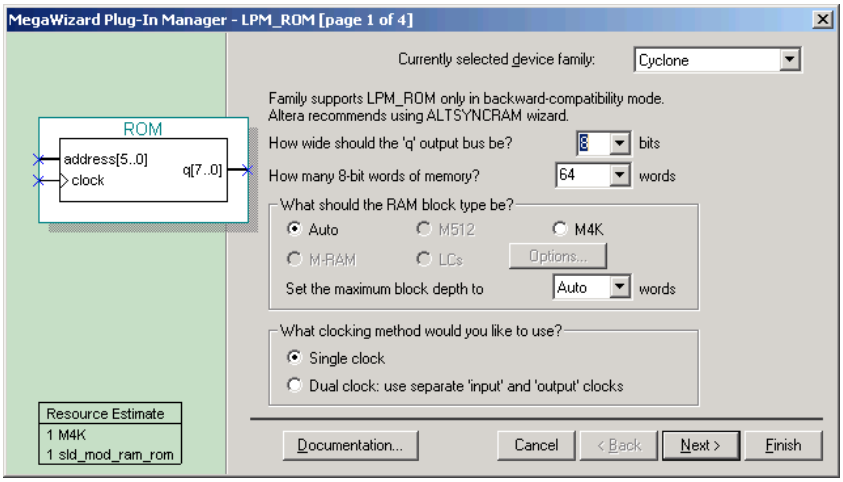


图 5.49　rom 宏模块基本参数设置

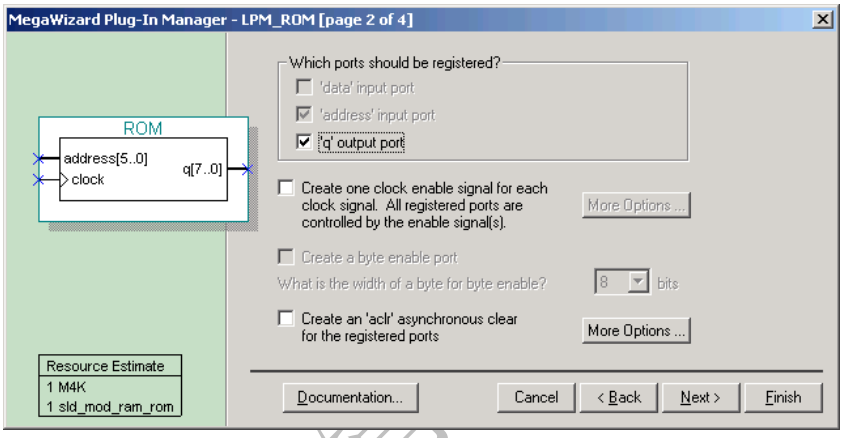


图 5.50　设置 ROM 宏模块其他参数

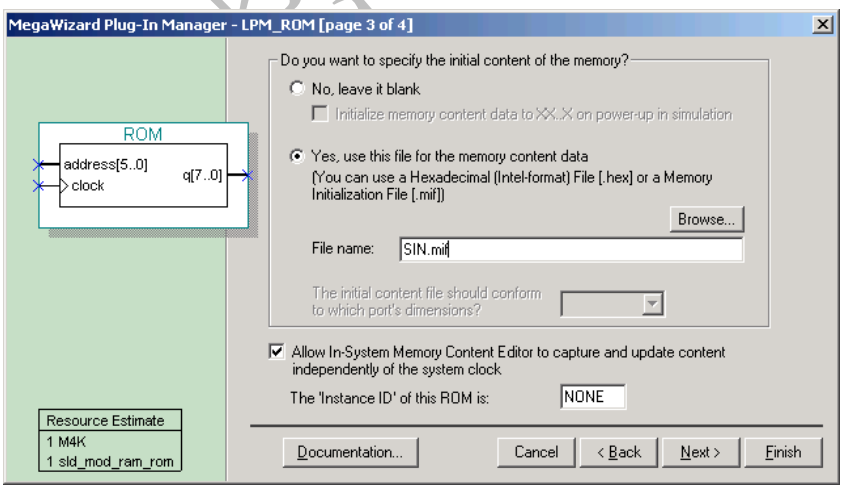


图 5.51　设置 ROM 宏模块初始化文件

单击“Next”按钮，完成设置，向导生成的文件如图 5.52 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

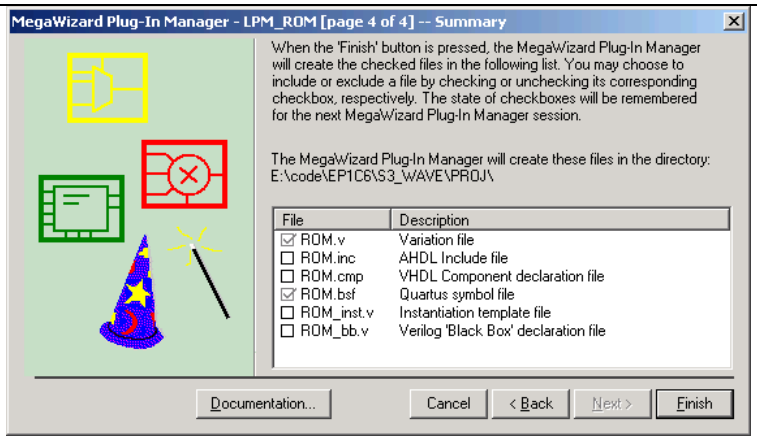


图 5.52　设置 ROM 宏模块完成

4．添加计数器

引入计数器作为 ROM 的地址，当计数器地址递增时，相应的 ROM 的地址递增（正弦波的相位增加）。同样的在“MegaWizard Plug-In Manager”中选择 arithmetic 数学库中的“LPM COUNTER”宏模块，添加计数器，如图 5.53 所示。

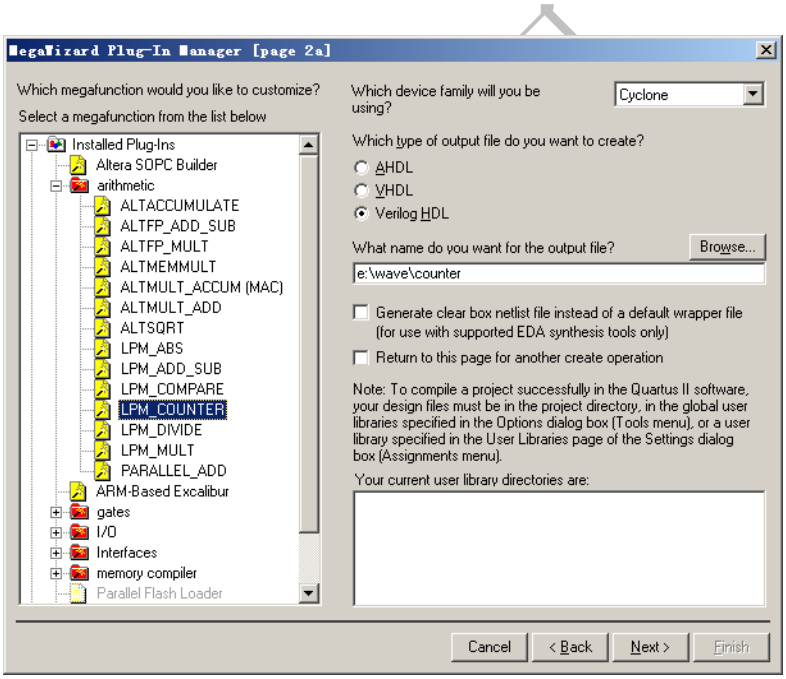


图 5.53　新建计数器模块参数

本实例中用的 ROM 宽度为 8，深度为 64，也就是数据宽度是 8 位，可以存 64 个数据。ROM 有 64 个地址，因此计数器的位宽要定义为 6 位，如图 5.54 所示。

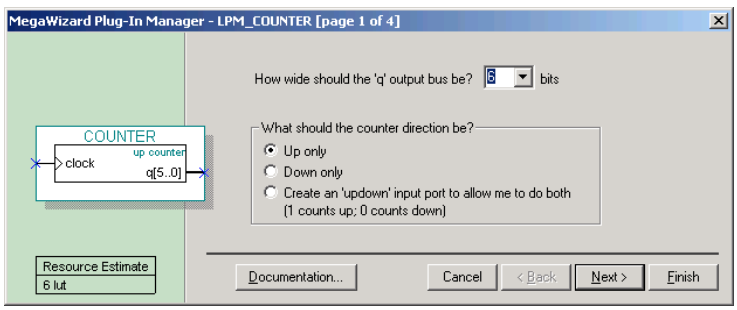


图 5.54　设置计数器宏模块参数

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

一直单击“Next”按钮，使用系统默认设置，直到生成如图 5.55 所示的文件输出。

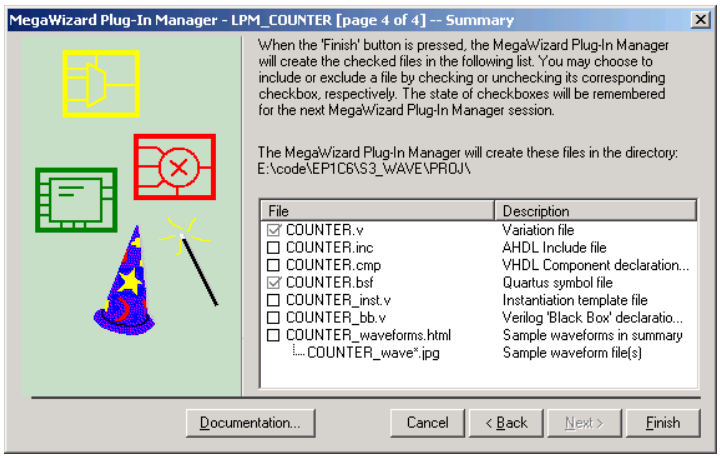


图 5.55　完成计数器宏模块设置

5.10.2.4　添加端口

要添加输入输出端口，选择按钮，打开如图 5.56 所示的对话框，然后选择需要增加的端口形式。

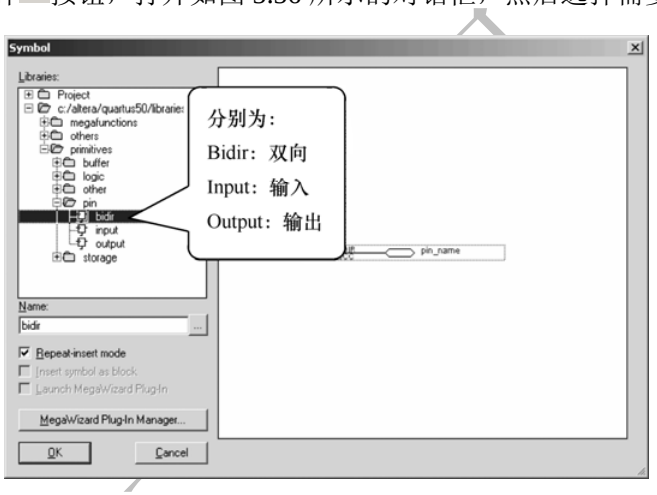


图 5.56　加入输入输出端口

添加后双击端口，给端口命名，总线型端口要命名 name[N..0]的格式，如图 5.57 所示。

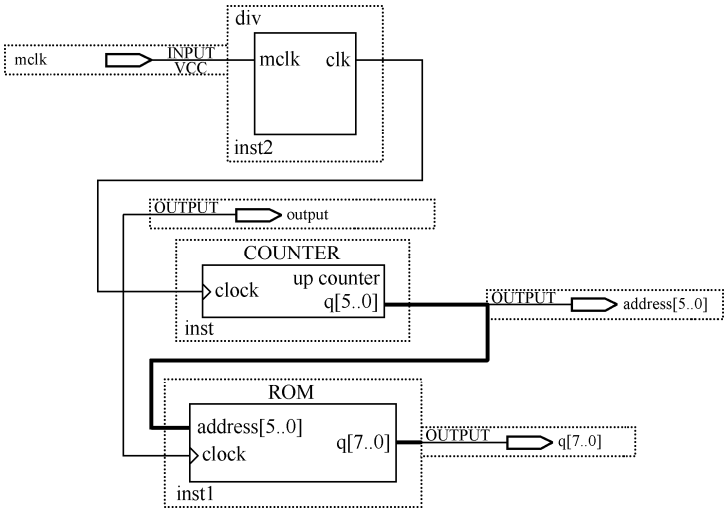


图 5.57　完成工程原理图输入

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

5.10.2.5 编译工程

按照 2.5 节进行语法分析，指定 FPGA 设备，指定管脚，整体编译程序。在本实例中只需要对主时钟管脚进行分配，即 pin_153。

5.10.2.6 使用 SignalTap II 观察波形

1．建立 SignalTap II 文件

在菜单中选择“Tools”/“SignalTap II Logic Analyzer”选项，打开如图 5.58 所示界面。

图 5.58 新建 SignalTap II 文件

作如图 5.59 所示的设置。

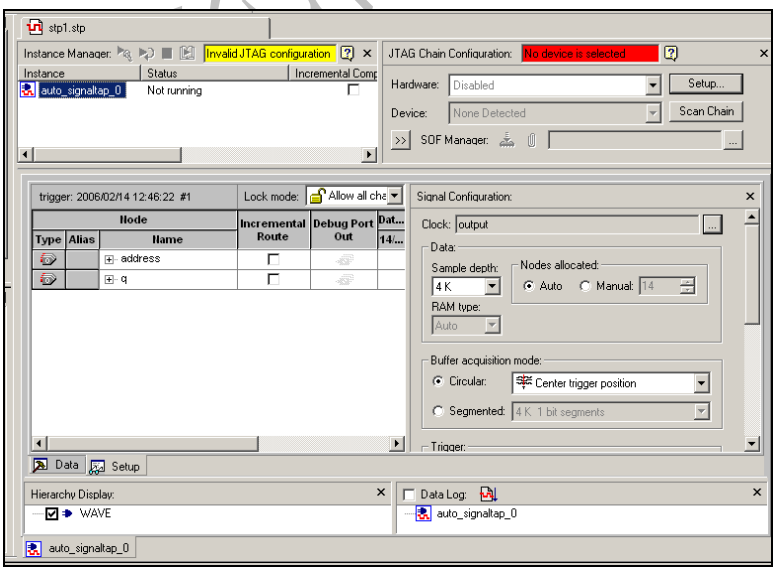


图 5.59 SignalTap II 文件设置

SignalTap II 基本设置需注意以下问题。

（1）分频时钟作为采样信号，不要放到被观察的信号中。

（2）采样信号不宜用作主时钟，可以用作分频后的时钟，采样条件可用 Center trigger position。

下载程序，就可以看仿真的波形了。

开发板免费领取试用 开发板免费领取试用

2．设置 SignalTap II 高级触发条件

在如图 5-58 所示的添加观察信号区域窗口中，将 Trigger Levels 选项改为 Advanced，将 Basic 改为 Advanced，则会弹出高级触发设置页面，如图 5.60 和图 5.61 所示。

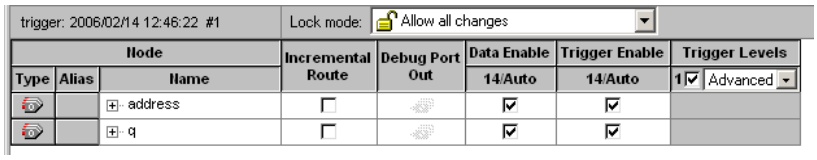


图 5.60　打开 SignalTap II 文件高级触发设置

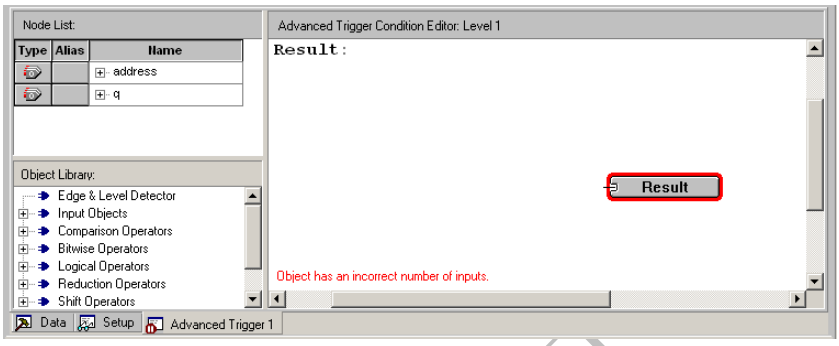


图 5.61　SignalTap II 文件高级触发设置

本例中设置为当地址信号 address＝0 时开始触发。从“node list”中拖入 address 信号，在“Object Library”/“Comparison Operator”中加入 equality，最后加入“Input Objects”/“Bus Value”。得到如图 5.62 所示的触发条件。

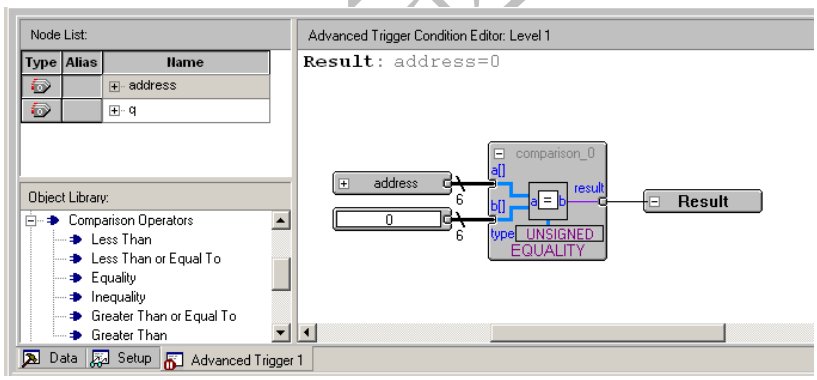


图 5.62　SignalTap II 高级触发设置结果

3．运行 SignalTap II 并观察波形

首先整体编译工程，在菜单栏中单击▶按钮，开始 JTAG 模式将程序下载到 FPGA 中运行。打开选择如图 5.63 所示的下载电缆。

图 5.63　SignalTap II 设置下载电缆

在 Instance Manager 中单击按钮进行一次触发，将会得到如图 5.64 所示的波形。

开发板免费领取试用　开发板免费领取试用

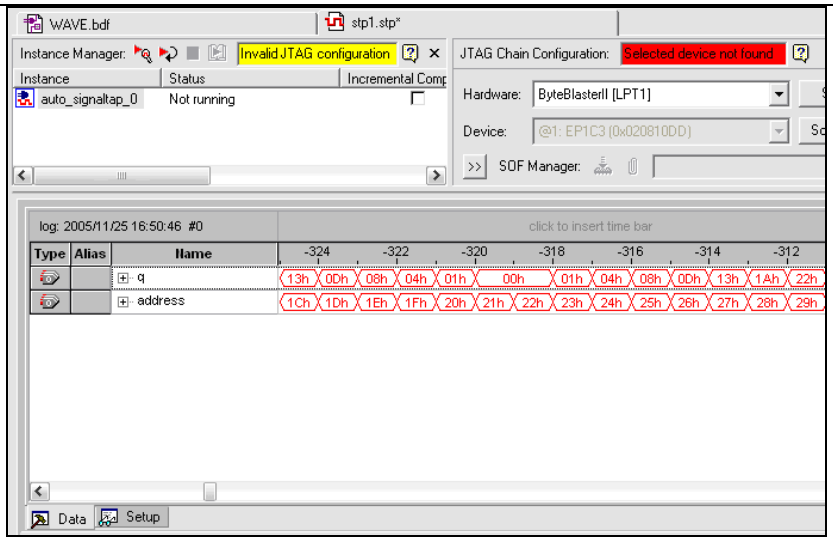
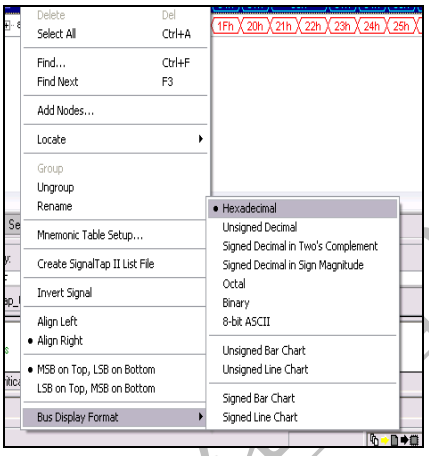


图 5.64　观察采样数据数字显示波形

因为开发板硬件没有 AD/DA 功能，要想看模拟信号的输出也可以用逻辑分析仪来完成。在信号名（比如 q 信号）上单击右键，选择“Bus Display Format”/“Unsigned Line Chart”选项，如图 5.65 所示。



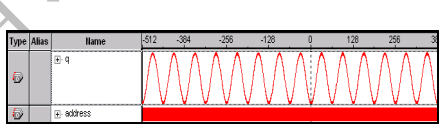


图 5.65　观察采样数据模拟显示波形　　图 5.66　观察采样数据模拟显示波形结果图

此时，就可以显示如图 5.66 所示的模拟波形了。

5.10.2.7　使用在线 ROM 编辑器

嵌入式存储数据编辑器是通过 JTAG 下载电缆来观看 FPGA 中 ROM 加载的数据的，不仅能观看，还能在线修改数据，无需重新编译与下载。

1．打开 ROM 编辑器

选择“Tools”/“In system Memory Content Editor”选项，打开如图 5.67 所示的 rom 编辑器。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

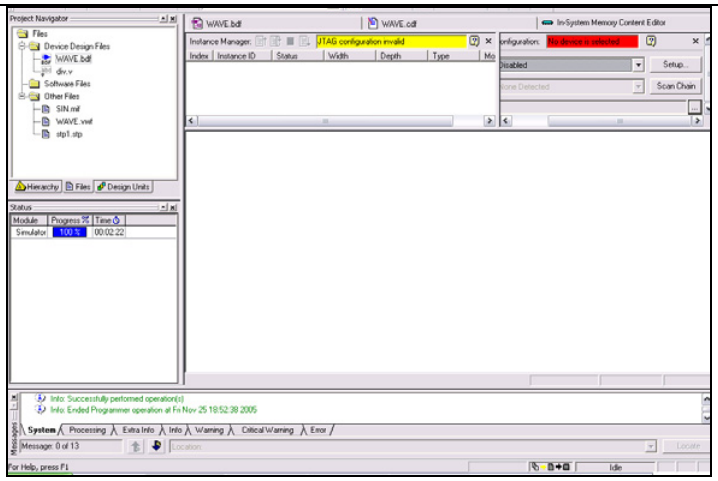


图 5.67　打开在线 ROM 编辑器窗口

2．选择下载电缆

使用打印口 LPT1 下载电缆，如图 5.68 所示。

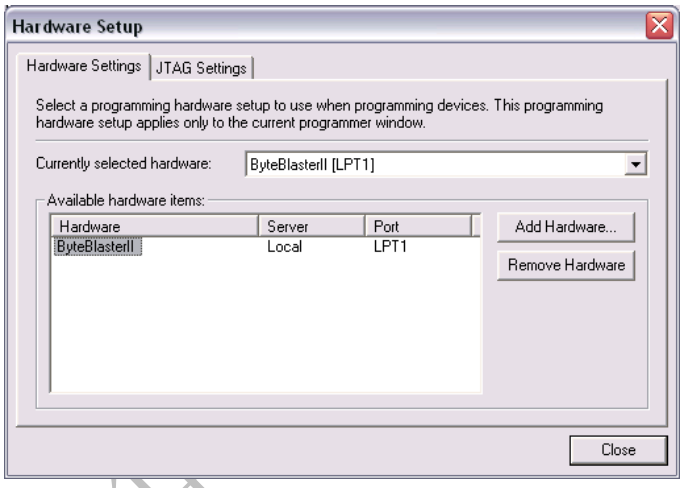


图 5.68　设置下载电缆

3．选择 ROM 标号

点击设置 ROM 参数时指定的 none 的文件，将出现 FPGA 运行之中的 ROM 的数据，如图 5.69 所示。

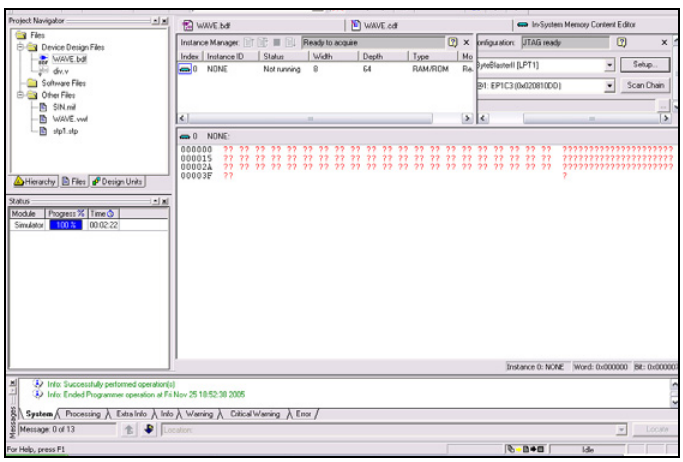


图 5.69　选择 ROM 标号

开发板免费领取试用　开发板免费领取试用

4．读取 ROM 数据

右键单击“NONE”，选择“Read Data from System Memory”选项，就可以得到 ROM 中的数据了，如图 5.70 所示。

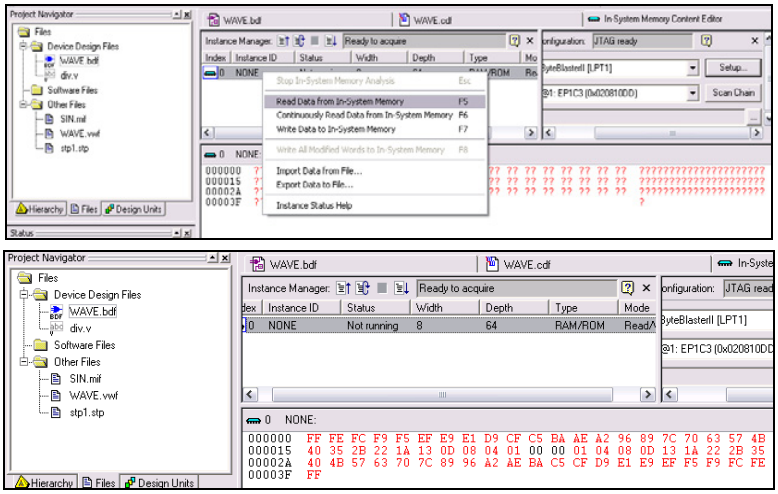


图 5.70　读取 ROM 数据

5．修改 ROM 数据

通过编辑器，可以修改 ROM 中的数据。例如可以把数据中的几个数改成 0，然后单击“write”，再回到逻辑分析仪中观察波形，就可以发现其中的变化，如图 5.71 所示。

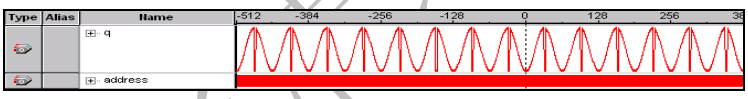


图 5.71　修改后 ROM 数据显示

5.10.3　小结

上述训练流程就是一个完整的 Quartus II 硬件调试流程。虽然此训练实现的功能比较简单，但对于初学者来说，是一个不错的入门训练。其主要目的是让初学者对的设计有一个初步的了解。通过该训练来熟悉 Quartus II 的高级调试技巧，为今后的学习打下基础。

5.11　典型实例 10：LogicLock 功能演示

5.11.1　实例的内容及目标

1．实例的主要内容

本节旨在通过 Quartus 软件自带的工程实例——“lockmult”来熟悉 Altera Quartus II 逻辑锁定功能 LogicLock 的使用方法。在本节中，将主要讲解下面知识点。

- Quartus II 约束设计。
- Quartus II 时序分析。

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

- Quartus II 逻辑锁定。

通过这些知识点，按照下面提供的训练流程，读者可以迅速地掌握使用 Quartus II 软件逻辑锁定功能的使用方法。

2．实例目标

通过详细的流程讲解，读者应达到下面的目标。

- 熟悉 Quartus II 时序分析报告的查看。
- 熟悉 SignalTap II 逻辑锁定的方法。

5.11.2 实例详解

本实例将使用图解的方式将整个流程一步一步展现给读者，使读者能够轻松掌握开发的流程。

5.11.2.1 打开工程文件

本实例以流水线乘法器为例，介绍逻辑锁定技术及相关的优化控制的基本用法。
打开位于路径 C:\altera\qdesigns50\logiclock\lockmult 中的工程 lockmult.qpf。这就是乘法器实例的顶层设计，其原理如图 5.72 所示，图中的 4 个元件 pipmult 是相同的。

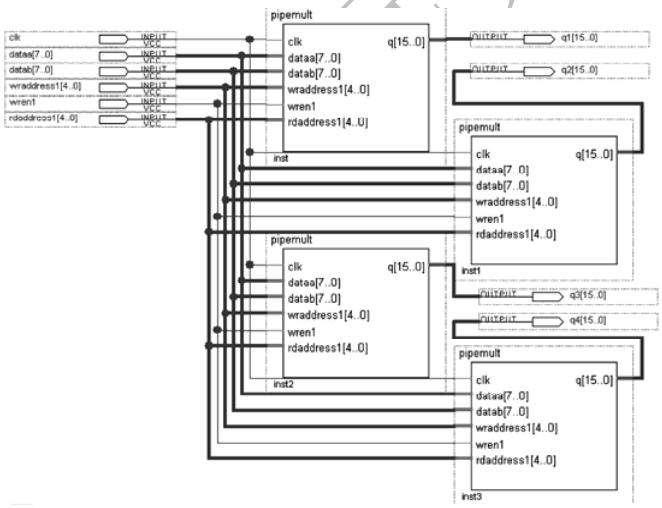


图 5.72 乘法器顶层原理图

pipemult 的电路原理图如图 5.73 所示。它是由一个 8 位流水线乘法器和一个 16 位双口 RAM 构成。

开发板免费领取试用 开发板免费领取试用

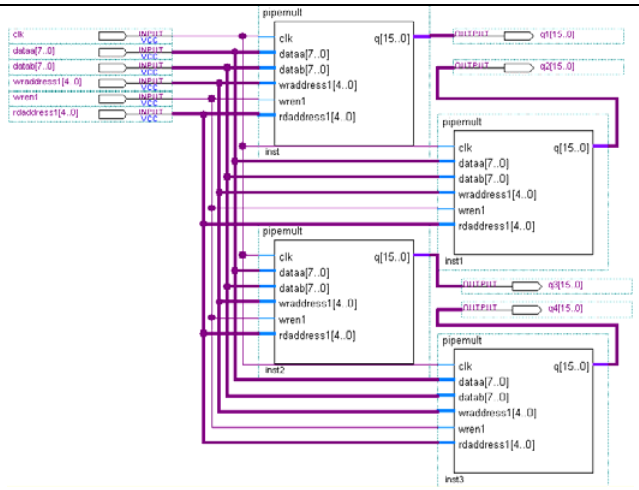


图 5.73　pipemult 内部电路原理图

选择目标器件为 EP20K100EQC208-1。

5.11.2.2　设置时序约束

在编译前设定时钟约束。在“Assignments”菜单中选择“Settings”选项。然后在“Settings”窗口的“Category”栏选择“Timing Requirements & Options”页面。

为“Clock Settings”下的“Default required fmax”输入 150 并选择 MHz，如图 5.74 所示，单击“OK”按钮，关闭窗口。

编译结束后，可在编译信息栏中看到最高时钟频率不能达到原设定的 150MHz 的要求（如图 5.75 所示）。打开“Processing”菜单中的“Complication report”窗口，单击“Timing Analyzer”可以在 Clock Setup 中看到哪些信号频率达不到要求。

5.11.2.3　使用 LogicLock 功能进行优化设计

1．建立底层模块工程

新建底层模块工程，在本实例中已经建立好底层模块工程，打开“典型实例 10”目录下的 lockmult.qpf 工程，如图 5.76 所示，可以看到乘法器的底层模块。

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

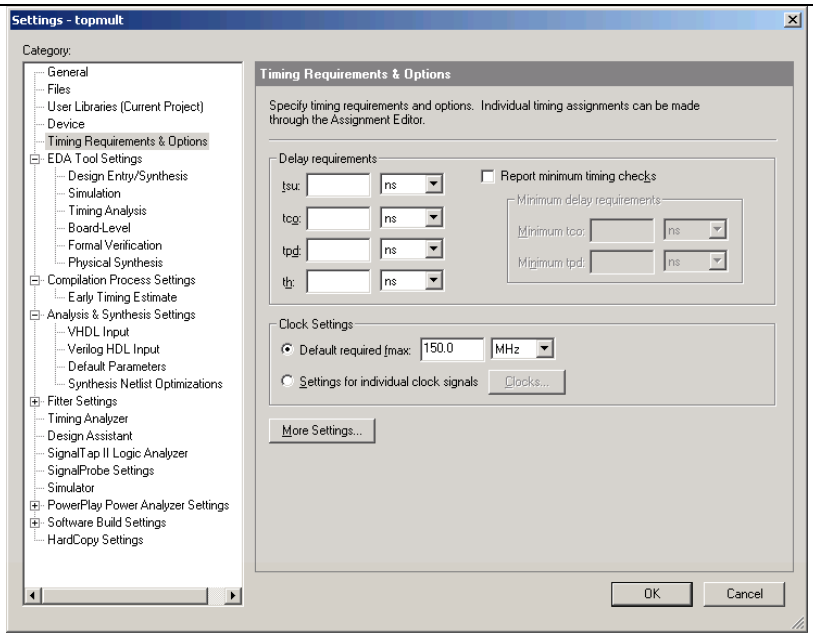


图 5.74　设计要求最高时钟频率

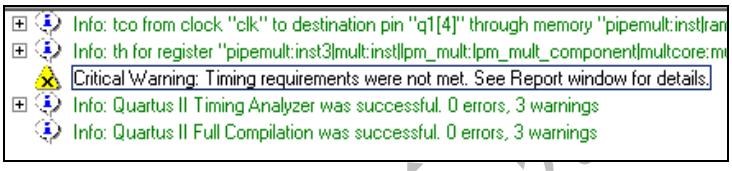


图 5.75　编译结果信息提示

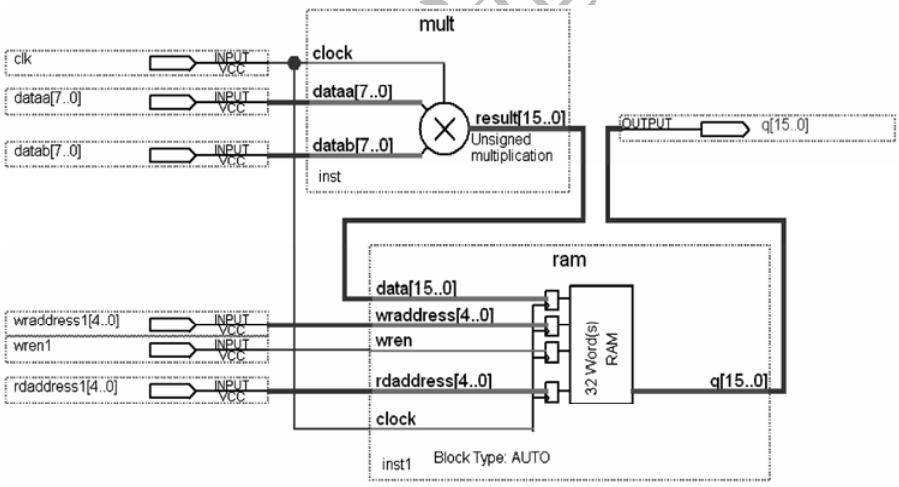


图 5.76　底层模块

指定设备：EP20K100EQC208-1，编译工程。

2．打开 LogicLock 窗口

使用快捷键“Alt＋L”或者选择“Assignment”/“LogicLock Regions Window”，打开 LogicLock 窗口，如图 5.77 所示。

开发板免费领取试用　开发板免费领取试用

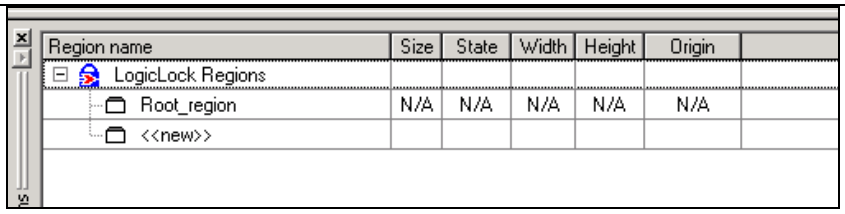


图 5.77　LogicLock 窗口

3．新建 LogicLock 区域

右键选择“Root_region”项并选择“new”，新建区域并将名称取为 lock_mult，如图 5.78 所示。

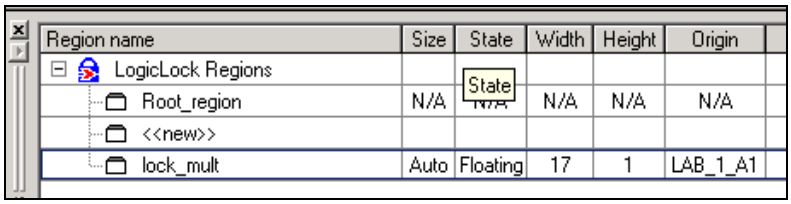


图 5.78　新建 LogicLock 区域

4．加入工程到 LogicLock 区域

右键选择新建的“lock_mult”项，打开属性窗口。将左侧的工程层次栏中的 pipemult 模块拖入，拖入后会出现 pipemult 的标志，如图 5.79 所示。

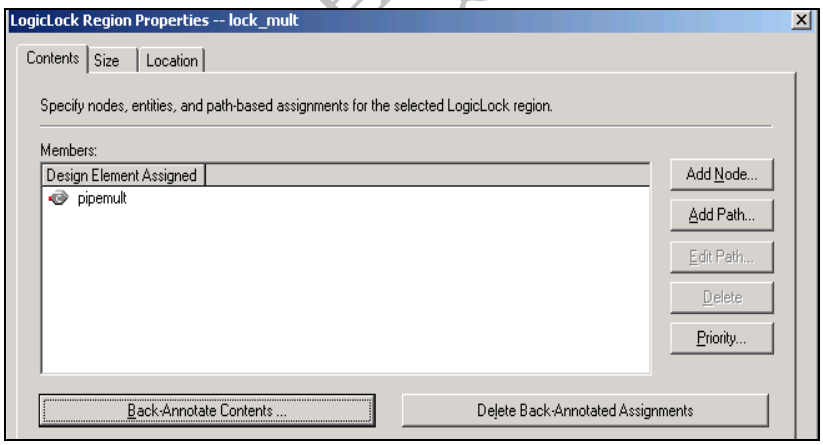


图 5.79　拖入 pipemult 模块

5．编译工程

对包含底层模块的 Logic Lock 工程进行第一次编译。

6．反标

回到主界面打开刚才的 lock_mult 属性窗口。单击“Back-Annotate Contents”按钮。反标约束信息，选择反标输出路径，如图 5.80 所示。
成功反标后会在对话框中出现如图 5.81 所示信息。

开发板免费领取试用　开发板免费领取试用

图 5.80　输出反标信息

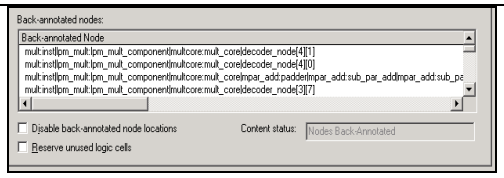


图 5.81　输出反标信息结果

7．导出 assignment 信息

选择“Assignment”/“Export Assignment”选项，如图 5.82 所示，导出约束信息。

8．打开乘法器顶层设计文件工程 topmult

使用 Quartas II 软件打开另一个工程——乘法器工程 topmult

9．替换底层模块设计文件

选择“project”/“Add/Remove Files in Project”选项，如图 5.83 所示。

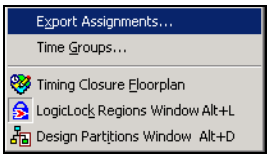


图 5.82　输出 Assignment 信息

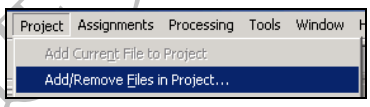


图 5.83　添加删除文件菜单

将原来的底层模块文件删除，使用刚创建的反标信息代替，如图 5.84 所示。

10．指定底层模块 Assignment 文件

回到主界面，单击左侧图标，展开 topmult，可以看到如图 5.85 所示工程层次目录。

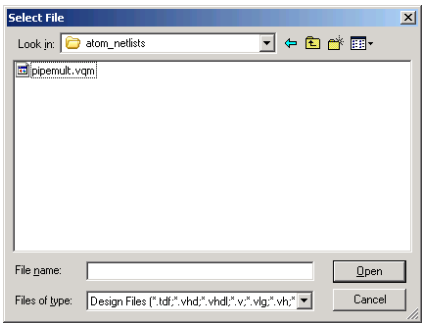


图 5.84　添加反标文件.vqm

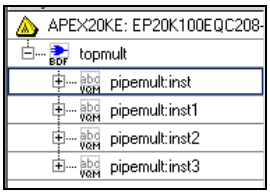
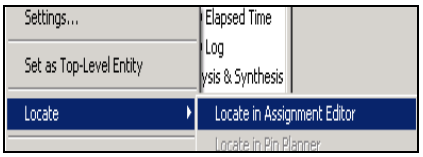


图 5.85　工程层次目录

右键选择底层模块“pipemult:inst1”，选择“Locate”/“Locatein Assignment Editor”选项，如图 5.86 所示。选择“Import File Name”，如图 5.87。



华清远见
华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

图 5.86 Locate 菜单

图 5.87 import file 窗口

选择 pipemult.qsf 文件，也就是刚才所选择输出设置信息，如图 5.88 所示。

依次对每一个底层模块重复这个步骤。

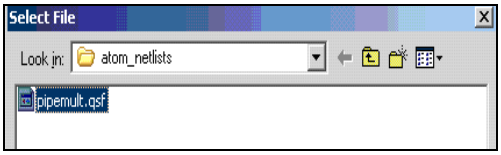


图 5.88 添加 pipemult.qsf 文件

11. 导入 Assignment 文件

选择“Assignment”/“Import Assignment”，输入刚才输出的设置文件。

12. 再次编译工程

编译整体工程，可以发现时钟频率符合要求，如图 5.89 所示。

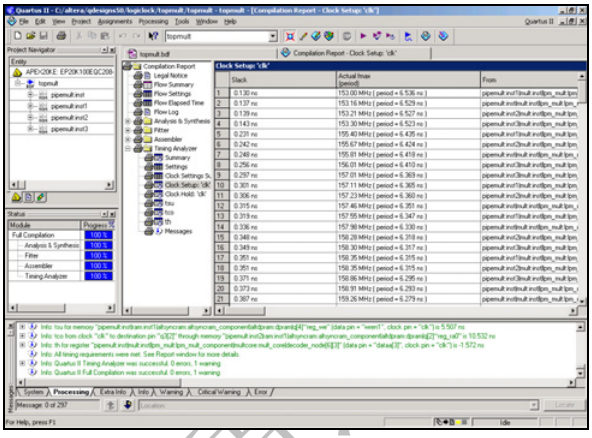


图 5.89 时序分析报告

联系方式

集团官网：www.hqyj.com 嵌入式学院：www.embedu.org 移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn 物联网学院：www.topsight.cn 研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路银海大厦 A 座 8 层，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218

开发板免费领取试用 开发板免费领取试用

华清远见 FARSIGHT 始于2004 | 嵌入式培训专家

10年口碑积累，成功培养50000多名研发工程师，铸就专业品牌形象

华清远见的企业理念是不仅要做良心教育、做专业教育，更要做受人尊敬的职业教育。

《FPGA 应用开发入门与典型实例》（修订版）

作者：华清远见

专业始于专注 卓识源于远见

第 6 章 FPGA 设计开发软件 ISE 使用技巧

本章目标

- 熟悉 ISE 软件的安装与启动
- 掌握 ISE 下 FPGA 的设计流程
- 掌握 ISE 下创建工程的方式
- 掌握 ISE 下如何编译和仿真
- 掌握在线调试工具 ChipScope Pro 的使用技巧
- 掌握 ISE 常用的增量式设计技巧

专业始于专注 卓识源于远见

开发板免费领取试用 开发板免费领取试用

6.1 ISE 软件简介

6.1.1 ISE 软件简介

Xilinx 作为当界上最大的 FPGA/CPLD 生产商之一，长期以来一直推动着 FPGA/CPLD 技术的发展。其开发的软件也不断升级换代，由早期的 Foundation 系列逐步发展到目前的 ISE 9.x 系列。
ISE 是集成综合环境的缩写，它是 Xillinx FPGA/CPLD 的综合性集成设计平台，该平台集成了设计、输入、仿真、逻辑综合、布局布线与实现、时序分板、芯片下载与配置、功率分析等几乎所有设计流程所需工具。ISE 系列软件分为 4 个系列：WebPACK、BaseX、Foundation 和 Aliance。ISE WebPACK 系列可以在 www.xilinx.com 网站上直接下载，是一个免费软件，支持一些常用的器件族；ISE BaseX 系列的器件最大规模不超过 700k 系统门；ISE Foundation 系列是最早期 Foundation 系列的延伸；ISE Alliance 系列支持的器件族最全，功能强大，是 Xilinx 的主推设计平台，所以推荐安装 ISE Alliance 系列。
ISE 的主要特点如下。

1．优良的集成环境

ISE 是一个集成环境，可以完成整个 FPGA/CPLD 开发过程。ISE 集成了很多著名 FPGA/CPLD 设计工具，根据设计流程合理应用这些工具，可以大大提高产品设计效率。

2．简洁流畅的界面风格

ISE 界面风格简洁流畅，易学易用。ISE 的界面秉承了可视化编程技术，界面根据设计流程而组织，整个设计过程只需按照界面组织结构依次单击相应的按钮或选择相应的选项即可。

3．丰富的在线帮助信息

ISE 有丰富的在线帮助信息，结合 Xilinx 的技术支持网站，一般设计过程中可能遇到的问题都能得到很好的解决。Xilinx 的官方网站上提供了相关软件（可供下载）、软件使用说明、软件更新、硬件资料、参考设计以及使用过程中常遇到的问题的解决等，此外还提供了大量的视频教程，便于用户学习。

4．强大的设计辅助功能

ISE 秉承了 Xilinx 设计软件的强大辅助功能。在编写代码时可以使用编写向导生成文件头和模块框架，也可使用语言模板（Language Templates）帮助编写代码，在图形输入时可以使用 ECS 的辅助项帮助设计原理图。
另外，ISE 的 Core Generator 和 LogiBLOX 工具可以方便地生成 IP Core（IP 核）与高效模块为用户所用，大大减少了设计者的工作量，提高了设计效率与质量。
目前 ISE 的最新版本为 ISE 9.1i。Xilinx ISE 9.1 于 2007 年 3 月发布，是业界最完整的可编程逻辑设计解决方案，用于实现最优性能、功率管理、降低成本和提高生产率。ISE 9.1i 利用新 SmartCompile 技术，来帮助用户在更少的时间内实现业内最快速的 FPGA 性能。

6.1.2 ISE 7.1i 特点

由于本书中所涉及的例程都是在 ISE 7.1i 下完成的，这里对 ISE 7.1i 的特点做重点介绍。Xilinx 于 2005 年

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

3 月推出针对 Xilinx Virtex-4 和新推出的 Spartan-3E 系列 FPGA 产品而优化的集成软件环境（ISE）7.1i 版。其相比与从前的版本有以下新特性。

1．易用性有所提高

ISE 7.1i 中新的易用性特色可以加快工程师的设计过程。在设计流程中的每一步，ISE 7.1i 都提供了显而易见的实施结果。新的设计摘要视图（Design Summary View）和消息过滤（Message Filtering）功能突出了重要的设计信息，从而减少了在详细的报告文件中搜索的需求。新的技术指示器（Technology Viewer）通过易于浏览的示意图表来显示合成后的实施结果。

2．集成了两款新的仿真器

ISE 7.1i 中还集成了两款新的仿真器，ISE Simulator 和 ModelSim Xilinx Edition-III，从而可实现更快的仿真和更大的设计容量。通过利用实时芯片上（in-silicon）调试功能来加强仿真能力，ChipScope Pro 和 ISE 7.1i 可使实时验证所需要的时间仅为 ASIC 或竞争 FPGA 验证流程的一半。ChipScope Pro 现在还允许设计人员从全球任何地方通过网络连接对系统进行验证和调试。

3．通用性能提高

通过支持 64 位 Linux，ISE 7.1i 为更高密度的设计和开发创造了优越条件。重要的是，ISE 7.1i 可直接插入到现有 EDA 设计流程中，与第三方 EDA 合作伙伴的合成、仿真、HDL 分析和验证等设计工具紧密集成。

4．支持 Spartan-3E FPGA 系列和超低功耗 Spartan-3L FPGA

对于设计人员来说，成本也是一项重大的挑战。ISE 7.1i 支持 Spartan-3E FPGA 系列和超低功耗 Spartan-3L FPGA，因而可支持额外的大批量设计，可大大节约设计者的成本。

总得来说，ISE 7.1i 独特的集成度、高速度以及易用性可以帮助设计人员解决所面临的最紧迫的一些挑战。新版工具集成了主要功耗分析、分层设计、仿真和调试等功能，还支持目前应用越来越多的基于 Linux 的设计环境。工具中还包括了针对在所有性能领域全球都最快的 FPGA Virtex-4 系列的新速度文件。

与竞争解决方案相比，ISE 7.1i 的逻辑构造性能优势高达 70%，同时在 DSP、嵌入式处理和连接功能方面也遥遥领先。设计人员可在设计中充分享受这些优势。ISE 7.1i 中还包括了对新推出的全球成本最低的 FPGA 产品 Spartan-3E 系列完全支持的功能。

6.2 ISE 软件的安装与启动

6.2.1 ISE 软件的安装

ISE 的安装改变了 license 管理方式，在安装后并不需要任何 license 支持，仅仅是在这安装过程式中输入 ISE 的注册序列号（Register ID）即可。ISE 7.1i 安装启动界面如图 6.1 所示。

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

图 6.1　ISE 7.1i 安装启动界面

安装 ISE 时只需要根据所选的版本是在 PC 机或工作站上，然后根据软件的提示安装即可，这里不做详细叙述，只对安装的几个问题进行说明。

1．环境变量的设置

安装过程结束后，为了能正常使用 ISE，还需要设置 ISE 的环境变量。假设 PC 机上 ISE 的安装目录为 C:\Xilinx。

（1）如果操作系统是 Windows 98，需要在 austoexec.bat 文件中加入：

```
set Xilinx = c:\Xilinx              //设置环境变量
set PATH = %Xilinx%\bin\nt          //设置系统路径
```

（2）如果操作系统是 Windows 2000，右键单击“我的电脑”，选择“属性”/“高级”/“环境变量”选项，在环境变量中加入：

```
变量名：Xilinx
变量值：C:\Xilinx
```

如图 6.2 所示。

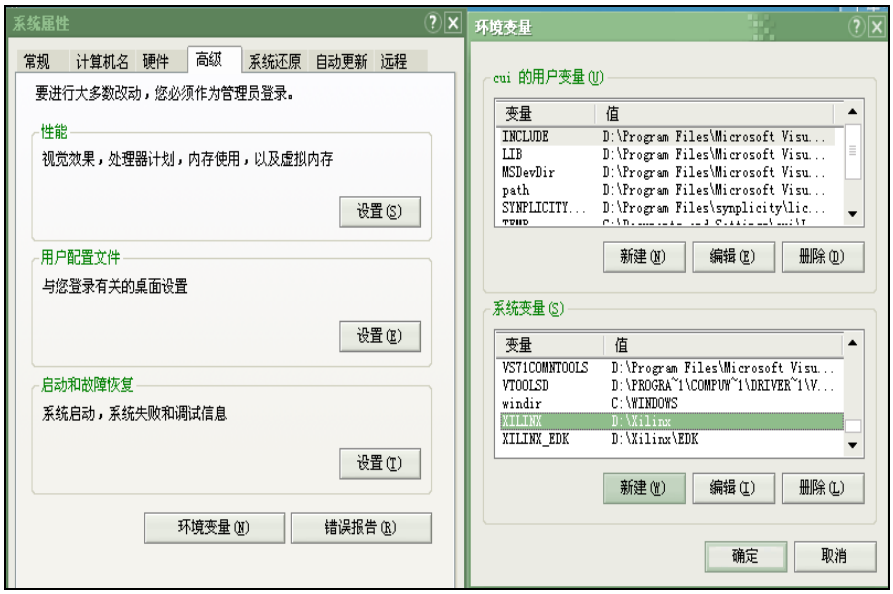


图 6.2　Windows 2000 环境变量配置

（3）如果操作系统是 Windows NT，加入过程与 Windows 2000 相似。右键单击“我的电脑”图标，选择“属性”/“高级”/“环境变量”选项，在环境变量中加入：

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

```
变量名：Xilinx
变量值：C:\Xilinx
```

（4）如果操作系统是 Window XP，可以不用做上述环境变量设置，安装完毕后，系统会自动将 Xilinx 的环境变量添加到系统中。

（5）如果操作系统是 Linux，需要编辑 WINE 配置文件/usr/local/ect/wine.conf，加入“Xilinx = C:\<your_Xilnx_dir>”，并且设置环境变量“setenv XIL_LINUX_WINE=1”。

2．第三方软件的安装

Xilinx 公司的 ISE 集成开发环境中给许多第三软件预留了接口，通过这个接口可以从 ISE 集成环境中直接启动第三方软件，常用的第三方软件（如：仿真工具 ModelSim）和综合工具（Synplify/Synplify Pro）。在安装这些第三方软件时需要注意两个问题。

（1）先安装 ISE，后安装第三方工具。

在这种情况下，ISE 会自动捕获第三方软件的路径，可以在 ISE 中直接启动第三方工具，如果有的用户要更换第三方软件版本，或先安装了第三方工具，需要在 ISE 集成开发环境的“EDIT”\“Preferences”\“Integrated Tools”选项下进行设置，如图 6.3 所示。

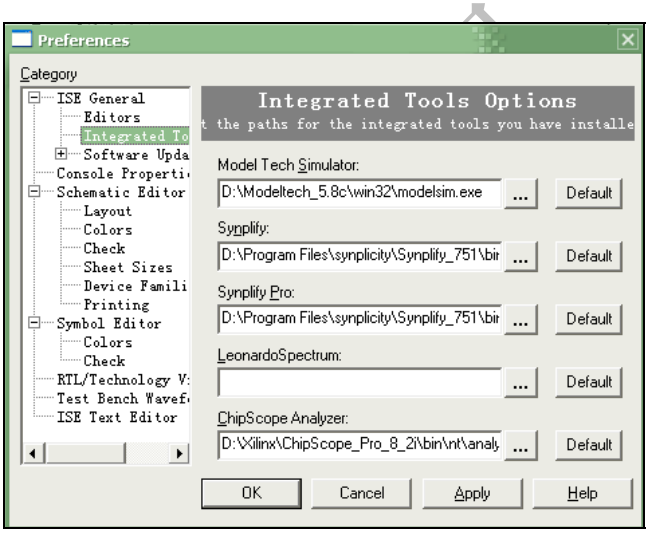


图 6.3　第三方软件接口路径设置

在该对话框中可以设置第三方软件 ModelSim、Synplify、Synplify Pro 的路径以及 ChipScope Analyzer，用户可以根据软件的安装路径进行设置和修改。

（2）第三方软件一般需要申请相应的 license，并设置 license 管理项目。

6.2.2　ISE 软件的启动

ISE 软件的启动可以通过桌面的快捷方式，如图 6.4 的左图所示：双击 Xilinx ISE 7.1i 的图标即可打开 ISE 软件。也可以选择“开始”/“所有程序”/“Xilinx ISE 7.1i”/“Project Navigator”选项来打开，如图 6.4 的右图所示。

开发板免费领取试用　开发板免费领取试用

图 6.4　ISE 软件的启动方式

ISE 软件启动后的界面如图 6.5 所示。

图 6.5 为 Project Navigator 界面，由图中可以看出，Project Navigator 的主界面由标题栏、菜单栏、工具栏、资源管理窗、当前资源操作窗、工程管理窗、HDL 编辑器工作区、信息显示窗和状态栏等部分组成。Project Navigator 是 ISE 的工程管理器，它是 ISE 所有集成工具的连接纽带。通过使用工程管理器，设计者可以创建、组织和管理自己的设计。

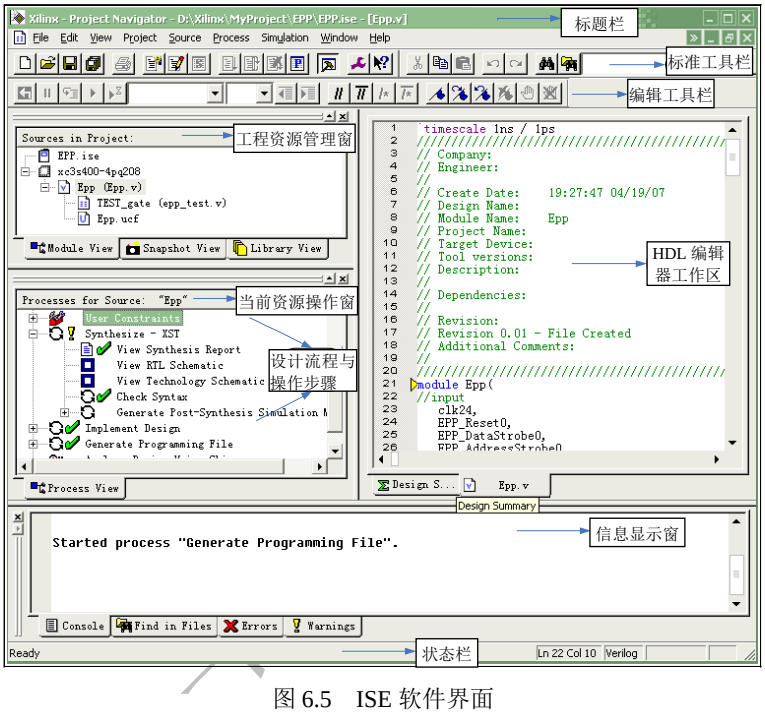


图 6.5　ISE 软件界面

6.3　ISE 软件的设计流程

Xilinx 公司的 ISE 软件是一套用以开发 Xilinx 公司的 FPGA&CPLD 的集成开发软件，它提供给用户一个从设计输入到综合、布线、仿真、下载的全套解决方案，并很方便地同其他 EDA 工具接口。

其中，原理图输入用的是第三方软件 ECS；状态图输入用的是 StateCAD；HDL 综合可以使用 Xilinx 公司开发的 XST、Synopsys 公司开发的 FPGA Express 和 Synplicity 公司的 Synplify/Synplify Pro 等；测试激励可以是图形化的 HDL Bencher，也可以由用户提供测试代码；通过 ModelSim XE（Xilinx Edition）或 ModelSim SE 进行仿真。

Xilinx 为 ModelSim 预留了接口，可以直接在 ISE 环境中打开，使用非常方便。并且 ModelSim 支持综合前、后仿真，以及时序仿真，功能很强大。

除了上述软件以外，也可以使用其他公司的相关 EDA 软件产品。

本节将对 ISE 的软件设计流程做一个全面的介绍。一般来说完整的 ISE 软件设计流程包括：电路设计与输入、功能仿真、综合、综合后仿真、实现、布局布线后仿真与验证以及下载调试等主要步骤，如图 6.6 所示。

具体讲解如下。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

1．设置工作环境

这一步并不是总是需要。通常用在第一次使用 ISE 或需要对某些项目进行修改时，一般有以下几项需要设置：这些设置主要是在“Edit”/“Preferences”下完成的，如图 6.7 所示。

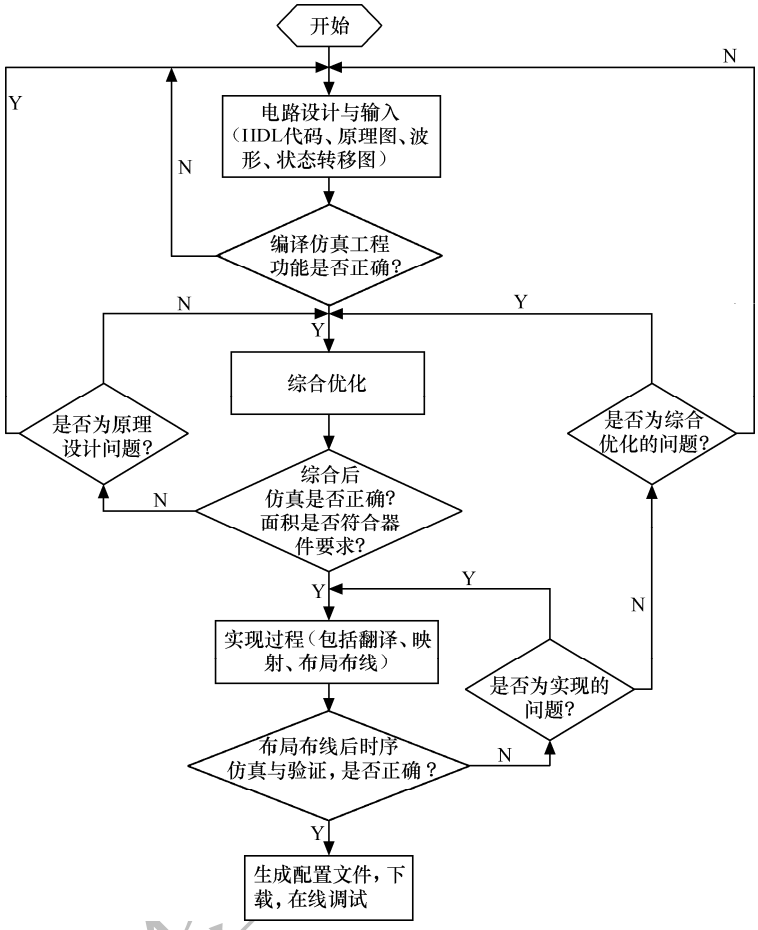


图 6.6　ISE 下 FPGA 设计流程图

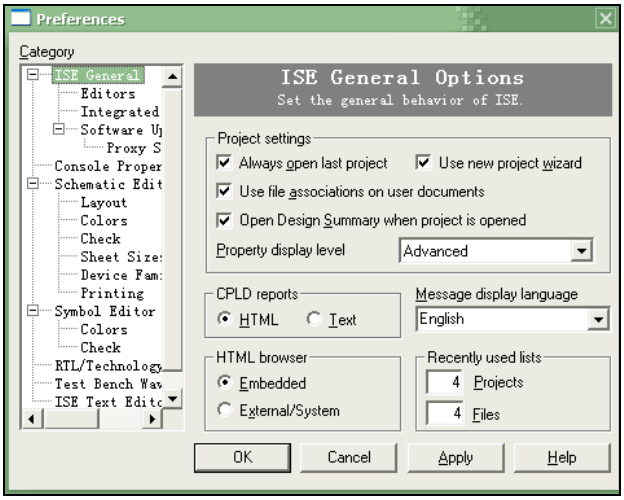


图 6.7　“Preferences”对话框

在 ISE General 中主要有下面几个选项可以进行设置。

（1）常用的。这主要是设置项目管理器中文件的显示方式、字体、窗口的显示方式等，一般用默认值就行。

（2）图形编辑器（Schematic Editor）。这里可设置跳格键（Tab）的字符个数、编辑器的字体等，另外除了直接用 ISE 提供的 HDL 编辑器外，也可以采用第三编辑器，如 gvim 就是一款非常优秀的代码编辑器，有

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

兴趣的读者可以在网上查阅相关资料。

（3）流程设置。

（4）工具设置。主要设置仿真器 ModelSim、HDL 测试台生成工具 HDL Bencher、状态图输入工具 State CAD 的工作目录。其实要设置的就是 ModelSim 的工作目录，因为后两项通常在安装完后 ISE 已经设好了。

2．新建工程

这一步和其他的软件开发一样，ISE 要求在对文件进行综合或布线之前必须要有一个存在的工程，在新建工程时，需要设置以下几点。

（1）工程名，最好用英文不要有汉字，因为 ISE 下有些工具对于含有汉字的文件目录支持的不是很好。

（2）工程所在目录。ISE 所产生的输出文件将全部放在该目录下，但对源文件的目录没有要求。

（3）器件家族。即设计中所采用的 FPGA 是 Xilinx 的哪一大类。

（4）器件型号。具体大类下的哪一种器件，此外还要设置封装和速度等级等，这些信息都可以从芯片上提供的信息直接得到。

（5）综合软件。由于 ISE 预置了 4 种可选的综合器接口（XST 为 Xilinx 自己开发，FPGA Express 是 Synopsys 公司的 OEM 版，在安装 ISE 时就已经装好了。而 Synplify/Synplify Pro 则需要另外购买并安装），所以必须选择一种作为该工程的综合器。4 种综合器全部支持 Verilog 和 VHDL。但有一点必须注意：如果设计中用到原理图，则只能选择 XST 或 FPGA Express 作综合器，因为 Synplify Pro 不支持原理图方式。

3．添加设计源文件

如果已有源文件，直接加入即可，否则可采用原理图方式或写 HDL 代码方式新建文件再加入。ISE 下支持多种新建资源类型，包括：User Document（工程说明文件）、VHDL module（VHDL 源代码文件）、VHDL Library（VHDL 库）、VHDL Package（VHDL 包）、VHDL Test Bench（VHDL 测试激励）、Verilog Module（Verilog 源代码文件）、Corgen Generator（IP 核生成工具）、Schematic（原理图）、Test Bench Waveform（测试激励波形）、BMM 文件（块 RAM 映射文件）、State Diagram（状态转移图）、UCF（用户约束文件）等。每种类型的资源在 ISE 都有对应的处理工具。

4．写测试文件

这一步可以利用 HDL Bencher 工具自动产生。测试台的主要功能是给被测实体加上输入激励，再比较其输出是否与期望值一致，并给出一些提示信息。但推荐大家自己写测试代码，通过写代码的测试灵活性更强，而且对于比较复杂的设计，有时在仿真时还需要读取文件数据，或将最终的仿真结果写进文件或打印，这在 HDL Bencher 下是很难完成的。

5．功能仿真

利用 ModelSim 来检查电路仿真结果是否正确。如果编译有错，则先将错误更改。如此反复直到仿真正确为止。这只是用于检查代码有无错误，ModelSim 最主要的作用在于通过仿真观察波形来验证设计的功能是否正确，这部分工作对于一个设计来说是非常重要的，因为如果前期的功能仿真做的不到位，会直接影响到最终电路功能的实现，必须在确保功能仿真没有问题的前提下，再进行下面的步骤。

6．综合

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

通过这一步将设计转换成具体的电路图。如果设计有错，有可能综合通不过。这就要求用户必须按照可综合代码的风格来设计。另外，综合有很多属性是可以设置的，如果对设计中的某些项目（如速度）有要求的话需要预先设置好。在 ISE 的高级版本中，XST 已经支持 Veilog 和 VHDL 混合语言代码输入，XST 的输出文件是 NGC 网表。

7．综合后仿真

综合实现后要进行综合后仿真，检验综合后功能是否符合设计要求，如果不符合要求要判断是否是原理图设计的问题，进一步对代码进行优化，直到符合设计要求为止。

8．设计用户约束文件

用户约束文件主要包括时序约束文件和管脚约束文件，时序约束可以在 ISE 自带的 Constraints Editor 下完成，管脚约束是在 PACE（约束编辑器）下完成的。约束设计完毕生成 UCF 文件。

9．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）。

10．布局布线后仿真

所谓布局布线仿真，是将 Xilinx 布线所产生的延迟加反标到电路后进行仿真。如果采用 ModelSim 仿真，在 ISE 下提供了 4 个仿真级别：Simulate Behavioral Model（行为仿真）、Simulate Post-Translate Verilog Model（翻译后仿真）、Simulate Post-Map Verilog Model（映射后仿真）、Simulate Post-Place&Route Verilog Model（布局布线后仿真）。

布局布线后生成的仿真时延文件包含的时延信息最全，不仅包含门延时，还包括实际布线延时，所以布线后仿真最准确，能较好地反映芯片的实际工作情况。

11．下载

把 Xilinx 布线后产生的结果转换成配置文件后置入 FPGA 中。下载成功后就可以测试实际电路了。如果需要脱机配置，则必须将配置文件写入外置存储器中。下载时需要将器件用 JTAG 线与 PC 机连接完成下载。其实上面的 10 个步骤并不是一定要按部就班，这取决于设计者的熟练程度和设计水平。例如测试台的编写、前仿真、后仿真并不是必须的。但为了保证设计的正确性和节约查错所耗的时间，推荐设计者一步一步操作，这样能够及时发现错误及时更正。

ISE 软件一个特点就是具有良好的开发界面，新建一个工程后，在“Processes for Source”对话框中即罗列出了 ISE 下 FPGA 设计流程中的各个环节，如图 6.8 所示，读者可将图中标注的内容与上面介绍的 FPGA 的设计流程中的各个步骤相对应，可以看出，界面上列出了设计流程的各个步骤，设计时只需要按照顺序依次实现就可以了，操作很简便。

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

由于 ISE 提供的集成开发环境非常好用，故推荐从 ISE 中来启动其他程序，当然也可以在各自的软件环境中完成上述步骤。

例如在 Synplify 中建工程、输入文件和综合，在 ModelSim 中新建工程并做仿真，用 Xilinx 的 Design Manager（这个程序在 ISE 中是找不到的，因为 ISE 是通过命令行方式调用该程序使其在后台运行，但可以在安装目录下找到它），再运行 JTAG 编程器下载。本节对 ISE 的 FPGA 设计流程做了一个简单的介绍，读者可对照 2.6 节加深对这一流程的理解。

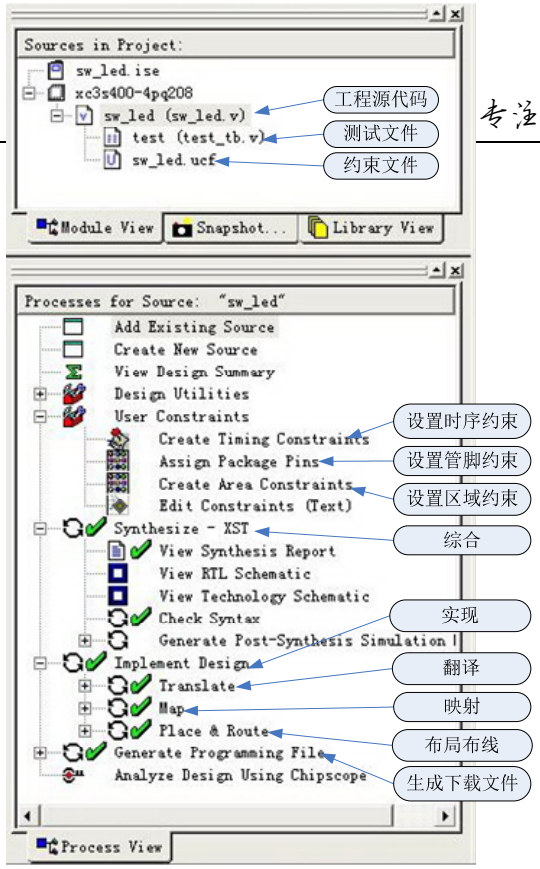


图 6.8　设计流程在 ISE 界面下的体现

6.4　创建设计工程

本节将重点讲述如何在 ISE 下创建一个新的工程。要完成一个设计，第一步要做的就是新建一个工程。具体创建一个工程有以下几个步骤。

（1）打开 Project Navigator，启动 ISE 集成环境。

ISE 的启动请参见 6.2 节。

（2）选择“File”/“New Project”菜单项，启动新建工程对话框。

会弹出如图 6.9 的对话框。

如图 6.9 所示，新建工程时需要设置工程名称和新建工程的路径，还要设置顶层模块的类型，具体各个类型解释如表 6.1 所示。

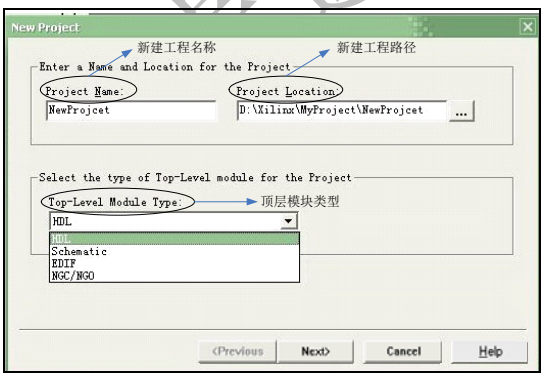


图 6.9　新建工程对话框

表 6.1　　顶层模块类型说明

顶层模块类型	类 型 说 明
HDL	硬件描述语言（Verilog 或 VHDL），用描述语言将各底层模块连接起来
Schematic	原理图，顶层模块可以用原理图将各底层模块连接起来，比较直观
EDIF	工业标准网表格式
NGC/NGO	综合后输出的文件格式，可以直接被 NGDBuild 读取

（3）设置工程属性。

启动新建工程对话框后，单击“下一步”按钮进入工程属性对话框设置，如图 6.10 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

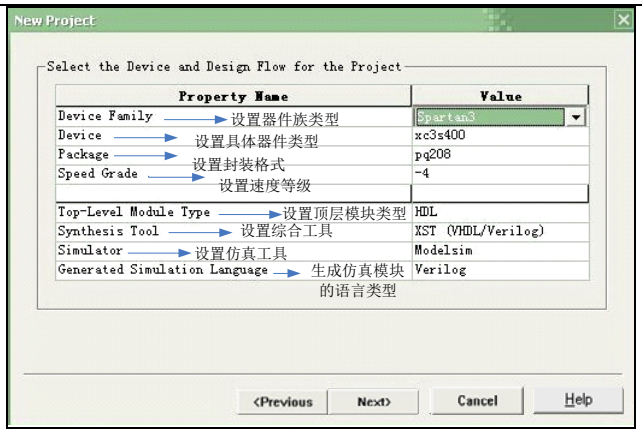


图 6.10　设置工程属性对话框

如图 6.10 所示，需要设置如表 6.2 所示内容。

表 6.2　　新建工程属性说明

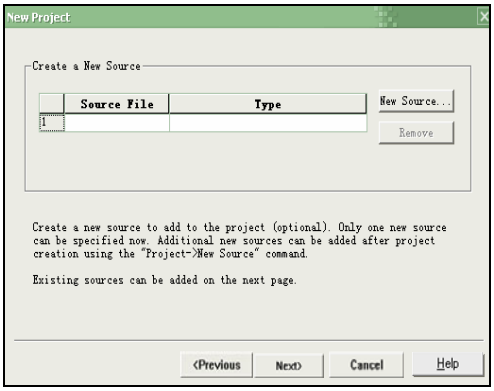
设 置 选 项	设 置 内 容
Device Family	设置 FPGA 是哪一系列的，如 Spartan3、Spartan3E、Virtex 等
Device	设置 FPGA 的具体型号，每个系列的 FPGA 下都有很多型号，要根据实际工程中应用的 FPGA 进行选择
Package	设置 FPGA 的封装格式，如 PQ208、FG456 等，对不同的封装格式，用户在约束引脚时会有所不同
Speed Grade	设置速度等级，如-4、-5、-6，数字越大速度越快
Top-Level Module type	设置顶层模块的类型
Synthesis Tool	设置设计中采用的综合工具，可以是 Xilinx 自带的 XST，如果安装了第三方工具，也可以选择第三方综合工具，如 Synplify/Synplify Pro 等
Simulator	设置设计中采用的仿真工具，可以是 ISE 自带的 ISE simulator，如果安装了第三方工具 ModelSim，也可以设置为 ModelSim
Generated Simulation Language	如果采用 ISE 自带的仿真工具 ISE Simulator，利用 HDL Bencher 可以在图形界面下编辑测试波形，直接生成测试激励文件，这里是设置生成测试激励文件的语言类型

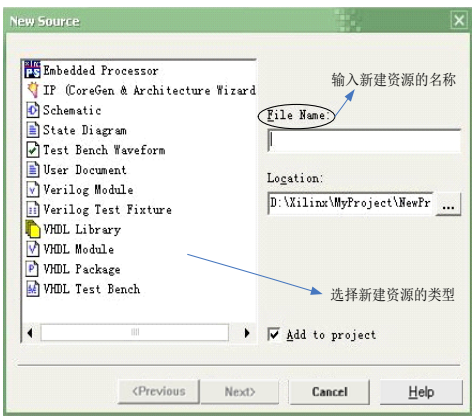
（4）为工程新建资源。

设置完工程属性后，单击“下一步”按钮，出现为工程新建资源的对话框，如图 6.11 所示。

新建工程时可以直接为新建的工程新建资源，单击“New Source”按钮会弹出新建资源的对话框，如图 6.12 所示。

这一步在新建工程时并不是必须的，如果在创建新的工程时没有为工程新建资源，可以在以后设计中再新建。如果工程创建完毕后需要新建资源，可以选择“Project”/“New Source”选项，也会弹出如图 6.12 所示的新建资源的对话框。





华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

图 6.11　为新建工程新建资源对话框　　图 6.12　新建资源对话框

具体新建资源的类型说明如表 6.3 所示。

表 6.3　　新建资源类型说明

新建资源类型	类 型 说 明	ISE 中对应的处理工具
Embedded Processor	嵌入式处理器	需要安装 Xilinx EDK 工具
IP（Corgen&Architecture Wizard）	IP 核	IP 核生成器（Core Generator）
Schematic	原理图	原理图生成器（ECS）
State Diagram	状态转移图	状态图编辑器（StateCAD）
Test Bench Waveform	测试激励波形	测试激励生成器 HDL Bencher
User Document	工程说明文件	文本编辑器
Verilog Module	Verilog 源代码	HDL 语言编辑器（HDL Editor）
Verilog Test Fixture	Verilog 测试激励	HDL 语言编辑器（HDL Editor）
VHDL Library	VHDL 库	HDL 语言编辑器（HDL Editor）
VHDL Module	VHDL 源代码	HDL 语言编辑器（HDL Editor）
VHDL Package	VHDL 包	HDL 语言编辑器（HDL Editor）
VHDL Test Bench	VHDL 测试激励	HDL 语言编辑器（HDL Editor）

（5）为工程添加现有资源。

单击“下一步”按钮，出现如图 6.13 所示对话框。如果工程的源代码已经编辑好，可以单击“Add Source”按钮为新建的工程添加资源，这一步与上一步一样，都不是必需的。如果工程已经创建完毕，可以通过选择“Project”/“Add Source”选项为工程添加资源。

（6）单击“完成”按钮，创建工程。

单击“下一步”按钮，会出现一个对话框，显示工程的相关信息，如图 6.14 所示，如有设置错误可单击“上一步”按钮进行修改。上述各步骤均设置好后，单击“完成”按钮就可以创建工程了。

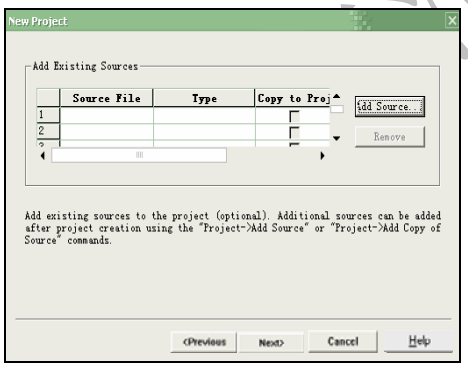


图 6.13　为新建工程添加资源对话框

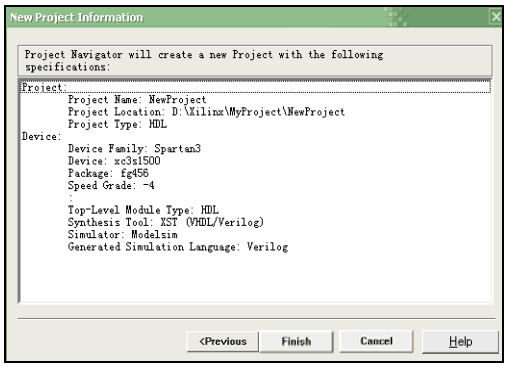


图 6.14　工程信息对话框

按上述步骤就可以完成一个新工程的创建。新工程创建完毕后，就可以通过新建或添加现有文件为工程添加资源，按照 ISE 的 FPGA 设计流程进行设计。

工程创建完毕后，如果需要修改工程的属性可以右键单击 图标。选择“Properties”选项，如图 6.15 所示，可以对工程的属性进行修改。

如果想更改工程的名称，同样右键单击图标，选择“Properties”选项，弹出如图 6.16 所示对话框，可以更改工程的名称。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

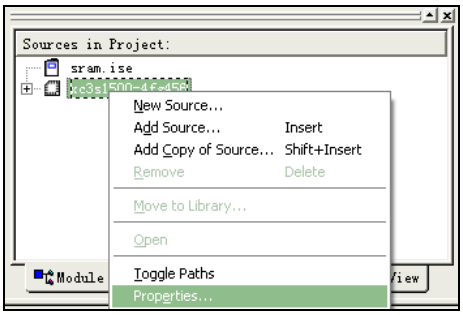


图 6.15　更改工程属性

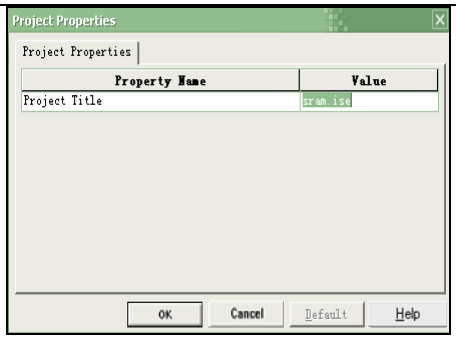


图 6.16　更改工程名称

6.5　编译与仿真设计工程

编写代码完成之后，一个很重要的工作就是验证代码功能的正确性，这就需要对代码进行编译与仿真。编译主要是为了检查代码是否存在语法错误，仿真主要为了验证代码实现的功能是否正确。

编译和仿真设计工程在整个设计中占有很重要的地位。因为代码功能不正确或代码的编写风格不好对后期的设计会有很大的影响，所以需要花很多时间在设计工程的仿真上。

在这一节中将通过一个具体的实例来介绍如何对编译工程代码以及如何使用 ISE 自带的仿真工具 ISE Simulator 进行仿真。

1．编译工程代码

编译主要是为了检测代码是否存在语法错误。在 ISE 下，源代码的编写是在 HDL Editor 下完成的，但在 HDL Editor 下没有专门用于编译代码的选项。不过在 HDL Editor 下完成代码的编写后，单击“保存”按钮，HDL Editor 就会自动对代码进行编译。如果代码存在语法错误，就会在信息显示窗中显示出来，用户可以根据显示的提示，查找语法错误并修改。

如图 6.17 所示为在输写代码时忘记分号，保存后就会有提示信息。

当不存在错误时，提示信息就不会出现“Warning”。ISE 下对于代码的编译功能并不是很强大，有很多错误是检测不出来的。例如在编写 Verilog 代码时，写 case 语句时漏写了 end case 语句，在 HDL Editor 下是检查不出来的。但这些错误在仿真或综合阶段是可以检测出来的，因此即使完成了编译没有错误，也一定要进行仿真，检查是否还存在其他的错误。

2．仿真设计工程

这里以一个具体的实例来介绍 ISE 下自带的仿真工具 ISE Simulator 的使用，代码参见本书实例代码。该例程的主要功能是根据拨码开关（sw）输入的值在数码管（seg_led）上显示相应的数值。

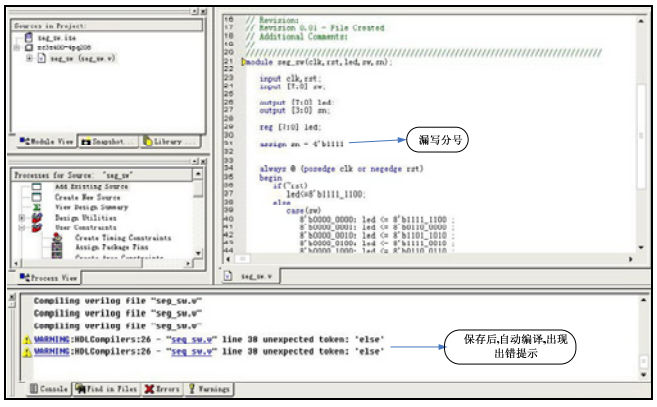


图 6.17　编译后的出错提示

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

ISE Simulator 的使用主要是借助于 ISE 的辅助设计工具 HDL Bencher（测试激励生成器）来完成的。用户将 VHDL 源代码、Verilog 源代码或 ECS 原理图等设计输入导入工程后，用户可以在图形界面下编辑测试波形，HDL Bencher 可以根据用户编辑的测试波形自动生成测试激励文件，然后调用 ISE 中的 ISE Simulator 进行仿真。可见使用 ISE Simulator 进行仿真主要分两步。

（1）调用 HDL Bencher，编辑测试波形，生成测试激励文件。

（2）调用 ISE Simulator 对工程文件进行功能仿真和时序仿真。

下面对实例做详细的介绍，具体步骤如下。

（1）启动 HDL Bencher。

首先打开实例工程，选择“Project”/“New Source”，弹出新建资源的对话框，选择“Test Bench Waveform”，输入测试激励波形文件名，如图 6.18 所示。

单击“下一步”按钮为测试激励文件选择源文件，如图 6.19 所示，选择要测试的源代码，单击“下一步”按钮。

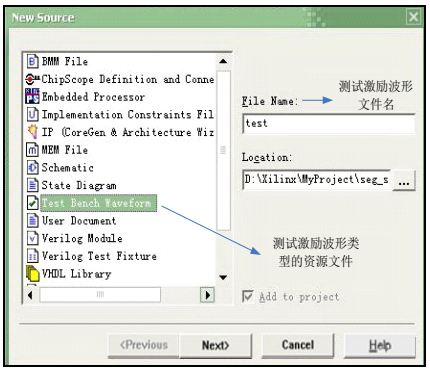


图 6.18　新建测试测试激励波形文件

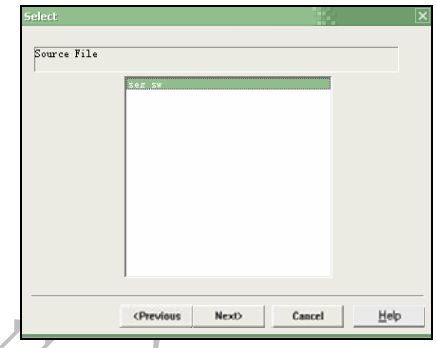


图 6.19　为被测试的源文件对话框

单击“完成”按钮确认新建资源信息，HDL Bencher 会自动启动。

（2）波形编辑。

HDL Bencher 启动后首先出现如图 6.20 所示的时钟设置对话框。

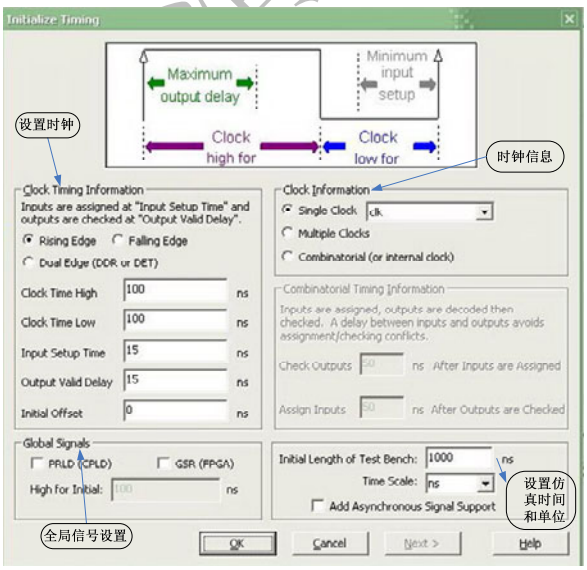


图 6.20　时钟设置对话框

如图 6.20 所示，时钟设置包括：时钟的高电平持续时间（Clock Time High）、低电平持续时间（Clock Time Low）、建立时间（Input Setup Time）和保持时间（Output Valid Delay），系统为单时钟（Single Clock）系统、多时钟（Multiple Clocks）系统以及仿真时间和单位。这里设置的仿真时间表示仿真将持续多长时间后自动停止。

如果设计存在异步时序（Asynchronous Signal Support），还要对异步时钟做相应的设置。这个设计中，只有同步时序，所以不选，与上表设置不一致之处请用户自行修改。设置完毕单击“OK”按钮后系统会自

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

动生成时钟的波形，如图 6.21 所示。

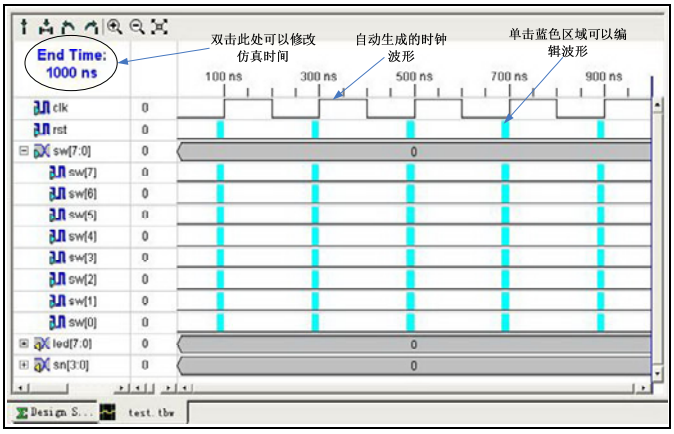


图 6.21　波形编辑界面

如图 6.21 所示，设置好时钟后，系统会自动生成时钟波形。在如图 6.18 所示的对话框中设置的“Initial Length of Test Bench”为 1000ns，这里可以看到仿真在进行了 1000ns 后就自动停止了，双击“End Time”，会弹出如图 6.22 所示的对话框，可以对仿真时间进行修改。

在设置好时钟频率、时钟建立时间和保持时间后，如果要修改，选择“Test Bench”/“Rescale Timing”会弹出如图 6.23 所示对话框，可以对时钟设置进行修改。

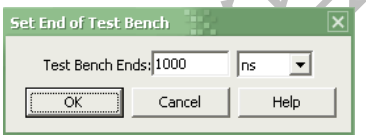


图 6.22　修改仿真时间对话框

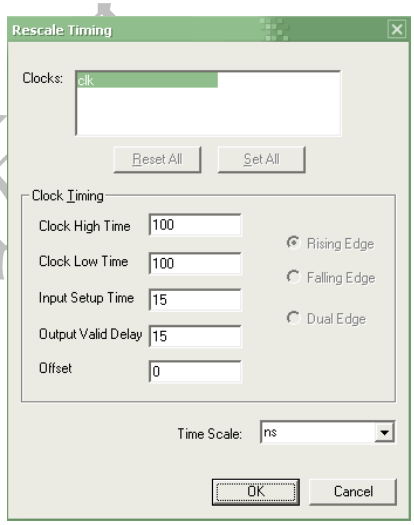
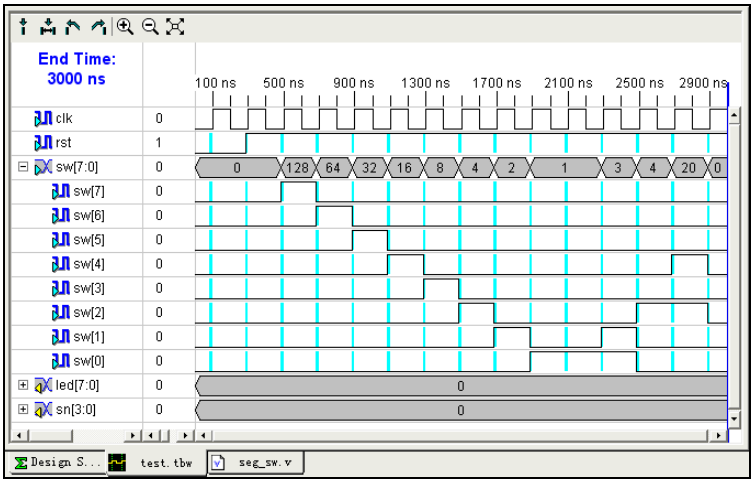


图 6.23　修改时钟设置对话框

这此设计中除时钟信号外，rst 和 sw[7:0]为输入信号，需要对 rst 和 sw 的波形进行编辑，编辑方法也比较简单，单击图中的蓝色区域就可以改变波形，根据仿真需要可任意设置波形，在这里设置仿真波形如图 6.24 所示。



华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

图 6.24　编辑波形图

波形图编辑完毕后，单击“保存”按钮，系统会为工程自动添加“test.tbw”文件。选中此文件，在当前资源操作（Process for Source）视窗中，可以看到“View Generated Test Bench As HDL”选项，如图 6.25 所示。

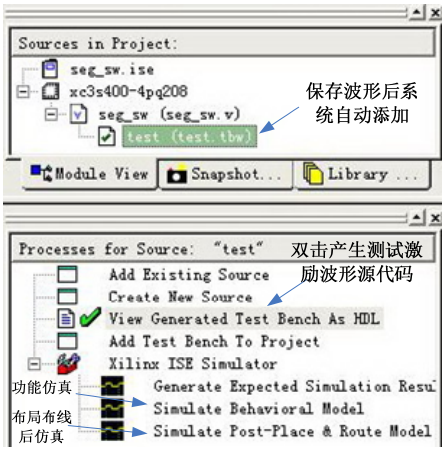
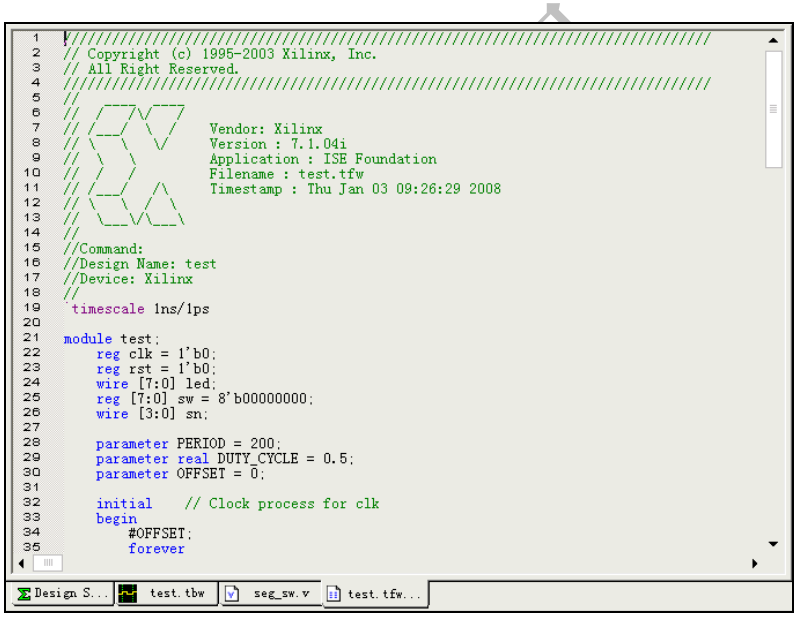


图 6.25　生成测试激励波形文件后的资源视窗

双击此选项，系统就会根据设置的波形自动生成测试激励文件的源代码，如图 6.26 所示。



```
////////////////////////////////////////////////////////////////////////////////
// Copyright (c) 1995-2003 Xilinx, Inc.
// All Right Reserved.
////////////////////////////////////////////////////////////////////////////////
//
//  ____/\/____
// /___/  \  /     Vendor: Xilinx
// \   \   \/      Version : 7.1.04i
//  \   \          Application : ISE Foundation
//  /   /          Filename : test.tfw
// /___/   /\      Timestamp : Thu Jan 03 09:26:29 2008
// \   \  /  \
//  \___\/\___\
//
//Command:
//Design Name: test
//Device: Xilinx
//
`timescale 1ns/1ps

module test;
    reg clk = 1'b0;
    reg rst = 1'b0;
    wire [7:0] led;
    reg [7:0] sw = 8'b00000000;
    wire [3:0] sn;

    parameter PERIOD = 200;
    parameter real DUTY_CYCLE = 0.5;
    parameter OFFSET = 0;

    initial    // Clock process for clk
    begin
        #OFFSET;
        forever
```

图 6.26　系统根据编辑的波形生成的测试代码

（3）对设计工程进行功能仿真。

ISE Simulator 中提供了两种级别的仿真：功能仿真和布局布线后仿真，功能仿真可以验证代码功能的正确性，布局布线后生成的仿真时延文件包含的时延信息最全，不仅包含门延时，还包括实际布线延时，所以布线后仿真最准确，能较好地反映芯片的实际工作情况。

按上述步骤用 HDL Bencher 生成测试激励波形文件后，就可对设计工程进行仿真了，首先验证设计功能的正确性，先对工程进行功能仿真。

功能仿真时在工程资源（Sources in Project）视窗中选择波形文件（test.tbw），在当前资源操作（Process for Source）视窗中，双击 Simulate Behavioral Model（如图 6.25 所示），得到功能仿真结果如图 6.27 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

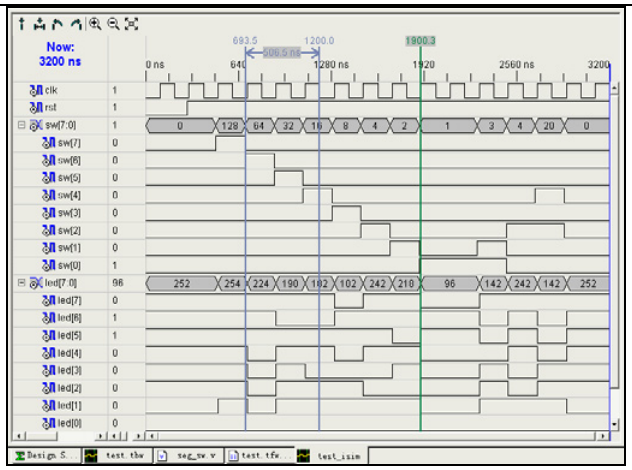


图 6.27　功能仿真结果

观察波形时，左上角有几个按键。单击可放置一条线，用于观查某一特定时刻的值。用于测量时间间隔。用于寻找前一个或下一个信号上升沿，主要用于在测量时间间隔时，定位测量直线。

另外，ISE Simulator 在仿真过程中会自动判断仿真过程中是否会有错误发生，如果有错误发生，会在 TX_ERROR 下显示出来，TX_ERROR 会自动统计错误的个数并显示，并且在 SimConsole 信号视窗中显示正确的数值和实际仿真得到的数据，用户可根据这些信息，对源代码进行修改。

（4）布局布线后仿真。

功能仿真测试功能正确后，就可以按照 ISE 下 FPGA 的设计流程对工程进行综合及布局布线。布局布线后，就可以对工程进行布局布线后仿真。

布局布线仿真时在工程资源（Sources in Project）视窗中选择波形文件（test.tbw）。在当前资源操作（Process for Source）视窗中，双击 Simulate Post-Place&Route Model（如图 6.25 所示），得到布局布线后仿真结果如图 6.28 所示。

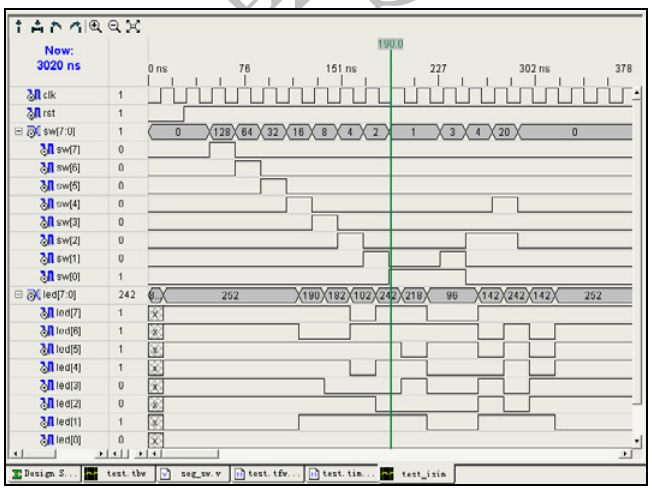


图 6.28　布局布线后仿真结果

如图 6.28 所示，布局布线后仿真结果与功能仿真结果相比，有了较大的延迟，这主要与代码中组合逻辑设置有关，在布局布线仿真后，由于加入了延时信息可能会造成功能的不正确，用户可以通过改写代码或添加相应的时序约束，来优化设计，布局布线后仿真在工程设计中占有很重要的地位，需要花很长时间来不断地仿真和优化设计。

3．小结

本节通过一个具体的实例介绍了 ISE 下自带仿真工具 ISE Simulator 的使用。ISE Simulator 的特点是可以通过 HDL Bencher 在图形界面下编辑波形，不需要用户编写测试代码，使用方便。用户也可以在 HDL Bencher 下完成波形编辑生成测试代码后，调用 ModelSim 仿真。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

通过这个例子，希望用户能够掌握 ISE Simulator 的基本用法，熟悉仿真的基本流程。仿真在 FPGA 的设计当中占有很重要位置，可以说设计者的大部分工作都是在做仿真，只有仿真结果达到要求了，才会映射到实际电路，然后在线调试。

在这里推荐大家自己编写测试激励文件。因为在一些复杂的设计当中，使用图形界面编辑激励波形是很难满足设计要求的，无法对工程进行完整的测试。

另外，ModelSim 是一款功能很强大的仿真软件，支持混合仿真。在 ISE 下为 ModelSim 预留了接口，使用也很方便，目前应用比较广泛。在初始学习阶段可以使用 ISE Simulator 作一些简单的仿真，熟悉 FPGA 的设计流程，等熟练之后，建议在 ModelSim 下完成仿真。

6.6 增量式设计（Incremental Design）技巧

本节将对 ISE 下增量式设计做一个全面的介绍。FPGA 作为一种现场可编程逻辑器件，其现场可重编程特性能够提高调试速度。每次硬件工程师可以很方便地改变设计，重新进行综合、实现、布局布线，并对整个设计重新编程。

然而当设计算法比较复杂时，每一次综合、实现、布局布线需要花很长的时间。即使仅仅改变设计中的一点，也会使综合编译的时间成倍增加。而且更为麻烦的是如果整个工程的运行频率很高，对时序的要求也很严格，这样重新布线往往会造成整个时序错乱。

运用增量式设计可以有效地解决这一问题。一方面大大节约综合、布局布线的耗时，另一方面可以继承前一设计中已有的成果，是一种比较常用的设计流程。

6.6.1 增量式设计的必要性

增量式设计（Incremental Design）方法是一种能在小范围改动情况下节约综合、实现时间并继承以往设计成果的设计手段。作为一个流程，增量设计能够极大地减小布局布线时间，并且当对一个近似完整的设计作小的变动，可以保持整个系统的性能。

在增量设计中每一个逻辑分组在 Xilinx 的 FPGA 里受到约束以使之只占有自己的空间。在设计中，对当对其中之一的逻辑分组做改动时，一个增量设计流程可以确保未做改动的逻辑分组在进行综合输出时不变化。接着布线工具对改动了的逻辑分组重新进行布局布线，而未改动的逻辑分组则继续以前的布局布线结果，这使得整个设计的布局布线时间得以削减。

增量式设计对一处复杂的设计来说是非常必要的，主要是因为增量式设计有以下两个方面的优点。

1．减小综合、布局布线的耗时

当仅对大型设计工程的局部进行改动时，增量设计流程仅仅改动的部分重新编译，如果改动模块的接口设计恰当，将不会影响其余部分的综合与实现结果，布局布线时也只对改动部分重新布线，未改动的部分保持不变，从而节约了整个编译、布局布线与优化的耗时。

2．能够很好地继承未修改区域的实现成果

这一点对于对时序要求很严格的设计来说是很有用的。如果一个设计经过多次调试，附加合适的约束，设置恰当的参数达到了最佳实现成果。但是因为对某个细节进行了修改，就需要全部重新综合、布局布线，这样可能前面所做的精心调整工作都无效了。

通过增量式设计，可以解决这一问题。对于已达到设计要求的部分将其保持不变，仅对修改的部分重新编译、布局布线，从而保证在最大程度上继承以往的实现结果。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

6.6.2 增量设计流程

具体的增量设计流程如图 6.29 所示。
增量设计的流程可归纳如下。

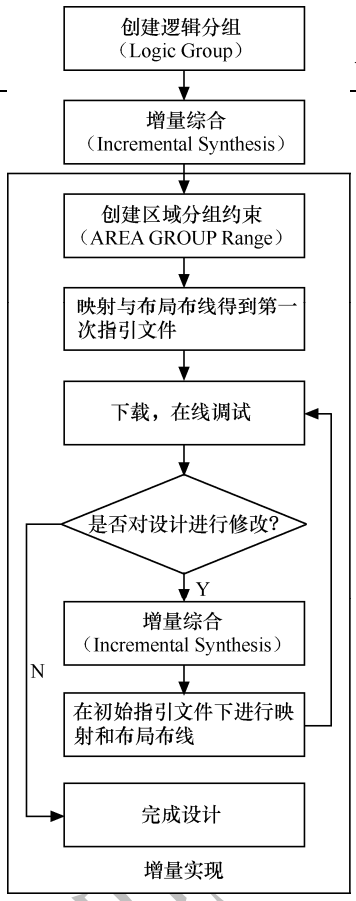


图 6.29 增量设计流程

1．创建逻辑分组（Create Logic Group）

在增量设计中为了实现减小综合、布局布线耗时，极大程度地继承未修改区域的成果，必须要求将设计分成多个逻辑分组。每个逻辑分组应该分配一定的逻辑区域，当某一逻辑分组的内容发生改变时，增量设计可以在该逻辑分组分配的逻辑区域内对其进行重新综合和布局布线，而不会影响到其他的逻辑分组。
所谓“逻辑分组”，是惟一的逻辑层次中的若干逻辑实体的划分。比如在顶层逻辑层次中每个子模块即为一个逻辑分组。在代码中即为顶层中“module（Verilog）”和“entity（VHDL）”定义的子模块的实体。在一个设计中往往将实现的不同功能设置为不同的模块，然后在一个顶层模块中实例化所有这些不同功能的模块，从而实现一个完整的功能，那么这些不同功能的模块就可以看作是不同的逻辑分组。
在进行逻辑分组时，需要考虑以下因素。
（1）设计中所有逻辑除了 IOB 和时钟逻辑，都应该包含在逻辑分组当中。
（2）顶层模块不应该包含复杂逻辑，仅仅包含一些 I/O 定义、时钟分配逻辑和所有子模块的实例化，直正的功能实体用子模块的逻辑描述。增量设计方法希望将所有的逻辑实体分割到子模块中去，而顶层模块不含任何实际的逻辑功能，以便于做相应的区域约束。
顶层包含实际逻辑功能的缺点在于：当顶层改变时，相关的 Logic Group 的接口将发生变化，从而影响 Logic Group 的结构，在做编译和布局布线时，会影响增量设计的效能。
（3）逻辑模块分组必须以寄存器输出，即用寄存器分割模块。这一点其实不仅仅是增量设计的需求，也是合理划分模块的一个基本要求。
如果采用同步时序方式设计电路，用寄存器分割逻辑模块，模块间的接口尽量简单，则时序优化路径集中在同一模块内部而不是模块之间的边界上。这样能够使综合器完整地掌握需要时序优化的路径，从而避免了因一个模块内部改变而通过边界影响到其他模块的时序这种不利于增量设计的情况发生。
（4）每个逻辑分组为其附加区域分组约束。

2．增量综合（Incremental Synthesis）

所谓增量综合是指只有改变的部分重新综合，而对未改变部分保持原有的综合结果的一种综合技术。传统的综合技术即使有微小的改动，也会对整个设计重新综合。
如果要实现增量综合必须对综合工具做相应的设置。在这里主要讲述 ISE 自带综合工具 XST 是如何实现增量综合的，对于其他综合工具如：Synplify/Synplify Pro 和 Leonardo Spectrum 综合工具，在这里不做详细介绍。
XST 支持单一工程的模块级增量综合（BLSI）。实现的方法为在 XST 的约束文件（扩展名为 xcf）中附加逻辑分组约束，从而告知 XST Logic Group 的边界。
XST 在综合时，所有的编译与优化都不超越用户在 XCF 文件中约定的 Logic Group 的边界，以达到在细微修改后仅仅对 Logic Group 内部进行重新综合的目的。这样一个逻辑分组 HDL 源代码的改变就不会对其他逻辑分组造成影响。

华清远见
华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

对于 VHDL 来说，XST 可以自动检测出哪一个逻辑分组的内容发生了改变。但对于 Verilog 来说，必须应用“resynthesize”属性。例如下面就是一个逻辑分组 A 的 Verilog 源代码发生细微的改变时的 XCF 文件范例。

```
MODEL "top" incremental_synthesis = yes;        //对 top 使能增量设计
MODEL "A" incremental_synthesis = yes;          //对 A 使能增量设计
MODEL "B" incremental_synthesis = yes;          //对 B 使能增量设计
MODEL "C" incremental_synthesis = yes;          //对 C 使能增量设计
MODEL "top" resynthesize = no;                  //通知综合工具哪个模块发生改变
MODEL "A" resynthesize = yes;                   //no 为未改变，yes 为已改变
MODEL "B" resynthesize = no;
MODEL "C" resynthesize= no;
```

对于 Verilog 设计工程，当某一逻辑分组发生改变时，要为其附加相应的综合约束，才能实现增量综合。另外需要注意的问题是，增量综合是在保留结构层次模式（Keep Hierarchy）下完成的，在进行增量综合时，在“Processes for Source”中选择“Synthesize-XST”，单击右键，设置综合属性如图 6.30 所示。

如图 6.30 所示，要完成增量综合，要完成两种设置，选择“Keep Hierarchy”设置为 YES 或 Soft，选择“Synthesis Constraints File”设置综合约束文件的路径。

增量综合后，需要检查综合报告。

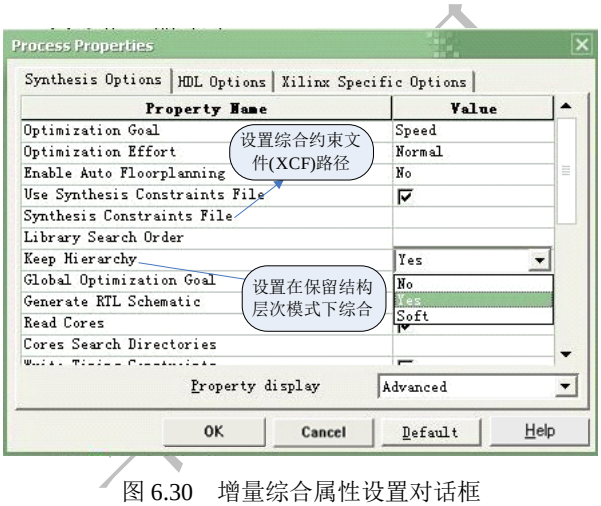


图 6.30　增量综合属性设置对话框

3．创建区域分组约束

创建区域分组约束是增量设计中最重要的一步。区域分组约束做得不好会增加综合、布局布线耗时，甚至有可能导致布局布线无法完成。区域分组约束的创建是利用 ISE 下的 PACE 工具，在 PACE 下完成区域分组约束后，ISE 会自动将其写入 UCF 文件中。

为增量设计创建合理的区域分组约束要遵循以下原则。

- 所有 I/O 引脚位置必须锁定。
- 将与 I/O 端口联系密切的区域分组布置在相应 I/O 端口的附近。
- 区域分组约束的范围不应该重叠。
- 尽量保证每个区域的资源利用率基本一致，避免出现某一区域利用率达到 99%，而另一区域的利用率只有 10%的情况。
- 如果一个区域分组中包含 FPGA 的多种资源，如：Slice、Block RAM、TBUF、Multipliers 等，这时很有必要将不同的资源设置在不同的区域，然后将几个不同位置区域拼合为一个区域分组。可以使用 PACE、Floorplanner 等工具完成，也可以在约束文件中使用如下约束:

```
INST Logic_Group_A AREA_GROUP = AG_Logic_Group_A ;          //AREA 分组
```


开发板免费领取试用　开发板免费领取试用

```
AREA_GROUP "AG_Logic_Group_A" RANGE = SLICE_X0Y20:SLICE_X20Y30 ;//SLICE 约束
AREA_GROUP "AG_Logic_Group_A" RANGE = RAMB16_X0Y2:RAMB16_X0Y2 ;//RAM 约束
AREA_GROUP "AG_Logic_Group_A" RANGE = MULT18X18_X0Y1:MULT18X18_X0Y1;//MULT 约束
AREA_GROUP "AG_Logic_Group_A" RANGE = TBUF_X0Y0:TBUF_X1Y0;      //TBUF 约束
```

4．增量实现（Incremental Implement）

第 3 步的创建区域分组约束也可以作为增量实现的内容，增量实现的步骤如下。

（1）设置区域分组约束。

（2）映射、布局布线得到初始指引文件。

（3）下载调试，需要修改时，对细微细节进行修改，然后实现增量综合。

（4）增量综合后在初始指引文件下进行第二次映射与布局布线。

（5）下载调试，如仍有问题，重复步骤（3）~步骤（5），直到符合设计要求为止。

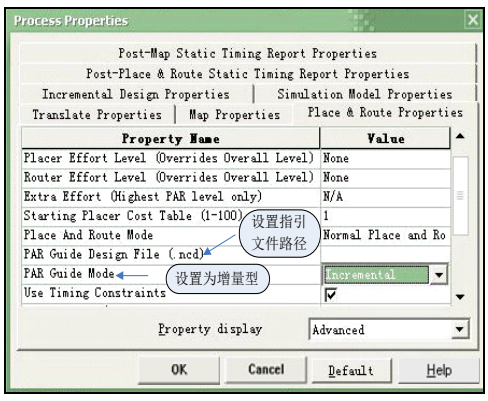
在增量实现过程中，必须要对布局布线的属性进行相关的设置。首先应该使能增量设计，具体设置如图 6.31 所示。在“Processes for Source”中选择“Implement Design”，单击右键，出现属性对话框。选择“Incremental Design Properties”，然后选择“Enable Incremental Design Flow”，使能增量设计。

图 6.31　项目管理器中启动增量设计功能对话框

在增量设计中，如果是用 Synplify/Synplify Pro 等第三方综合工具，那么需要由综合工具生成向导文件指导实现工具进行实现流程。选择如图 6.31 中的“Run Guided Incremental Design Flow”选项，表示将由指定的向导文件来指导实现工具进行实现流程。

向导文件可提供 Map.ncd 文件和 Par.ncd 文件。其中，Map.ncd 文件包含了 slices、IOBs 等映射信息。Par.ncd 文件包含了布局和布线信息。实现工具根据这两个文件的信息来确定没有改变的模块和已经改变的模块，把没有改变的模块放进向导文件所记录的上次综合实现的位置。对布线器来说，该布线资源同时也将被保留，其布线关系也不会改变。

使用向导文件，要对实现具的属性做部分修改。将“Map Guide Mode”和“PAR Guide Mode”分别改成增量型（Incremental），如图 6.32 和图 6.33 所示。



华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

图 6.32　使用指引文件指引映射　　图 6.33　使用指引文件引导布局布线

进行了上述设置后，需要指定映射和布局布线的指引文件。
整个增量实现步骤完成后，需要检查映射报告（扩展名为 map）和布局布线报告（扩展名为 par），看是否真的完成了增量设计。

6.6.3　小结

本节对增量式设计方法的概念以及设计流程做了全面的介绍，希望读者能够掌握增量设计的基本流程，并应用到自己的工程设计当中。在 6.9 节中会通过一个具体的实例来说明增量设计的整个流程。
增量设计在大型的工程设计中是很有用的，可以为设计节约大量的时间。增量设计中一个难点在于分组区域约束的设置，这不但需要对整个工程有全面的把握而且要求对于 FPGA 器件的内部结构有一定的了解，这需要在实践中不断积累经验。

6.7　片上逻辑分析仪（ChipScope Pro）使用技巧

在 FPGA 的调试阶段，传统的方法在设计 FPGA 的 PCB 板时，保留一定数量的 FPGA 管脚作为测试管脚。在调试的时候将要测试的信号引到测试管脚，用逻辑分析仪观察内部信号。
这种方法存在很多弊端：一是逻辑分析仪价格高昂，每个公司拥有的数量有限，在研发期间往往供不应求，影响进度；二是 PCB 布线后测试脚的数量就确定了，不能灵活地增加，当测试脚不够用时会影响测试，测试管脚太多又影响 PCB 布局布线。
ChipScope Pro 是 ISE 下一款功能强大的在线调试工具。面对这些问题，ChipScope Pro 都可以有效地解决。

6.7.1　ChipScope Pro 概述

ChipScope Pro 是针对 Xilinx Virtex-II pro/ Virtex/ Virtex-II/ Virtex-EM/ Spartan-IIE/ Spartan-IIE 系列 FPGA 的在线片内信号分析工具。它的主要功能是通过 JTAG 口，在线实时读取 FPGA 的内部信号。
ChipScope Pro 的基本原理是利用 FPGA 中未使用的 BlockRam，根据用户设定的触发条件将信号实时地保存到这些 BlockRam 中，然后通过 JTAG 口传送到计算机，最后在计算机屏幕上显示出时序波形。ChipScope Pro 应用的框图如图 6.34 所示。

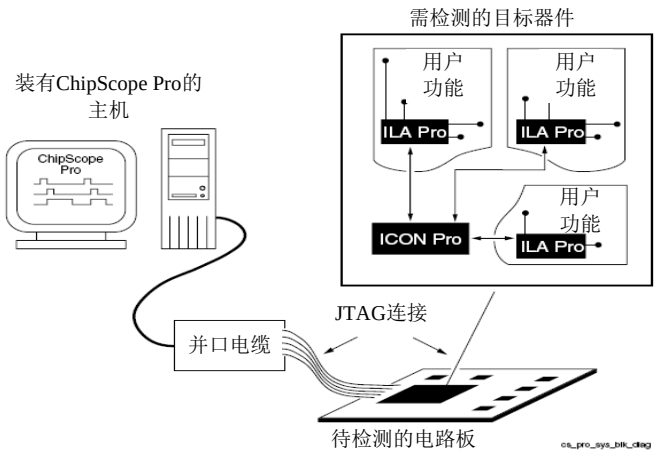


图 6.34　ChipScope Pro 应用框图

其中 ILA、ICON 是为了使用 ChipScope Pro 观察信号而插入的核。ChipScope Pro 工作时一般需要用户设计中实例化两种核：一是集成逻辑分析仪核（ILA core，Integrate Logic Analyzer core），该核主要用于提供触发和捕获的功能；二是集成控制核（ICON core，Integrated Contorller core），负责 ILA core 和边界扫描端

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

口（JTAG）的通信。

一个 ICON core 可以连接 1～15 个 ILA core。ChipScope Pro 工作时，ILA core 根据用户设置的触发条件捕获数据，然后在 ICON core 控制下，通过边界扫描端口上传到计算机，最后用 ChipScope Pro Analyzer 显示信号波形。

6.7.2 ChipScope Pro 设计流程

ChipScope Pro 工具箱中包含了 3 个工具：ChipScope Pro Core Generator、ChipScope Pro Core Inserter、ChipScope Pro Analyzer，使用 ChipScope Pro 在线调试工具的 FPGA 设计流程如图 6.35 所示。

由上述流程可知，ChipScope Pro 有两种使用方法。

第一种是由 ChipScope Pro Core Generator 根据设定条件生成在线逻辑分析仪 IP 核，包括 ICON core、ILA core、ILA/ATC core 和 IBA/OPB core 等，之后设计人员在原 HDL 代码中实例化这些核，然后进行综合、布局布线、下载配置文件，就可以利用 ChipScope Pro Analyzer 设定的触发条件，观察信号波形。

第二种是原代码完成综合后，由 ChipScope Core Inserter 工具插入 ICON core 和 ILA core 等核，它能自动完成在设计网表中插入这些核的工作，而不用手动在 HDL 代码中实例化这些核，在实际中应用的比较多，也是推荐大家使用的方法。下面就重点介绍 ChipScope Pro Core Inserter 和 ChipScope Pro Analyzer 的使用，这里以 ChipScope Pro 8.2i 为例来介绍。

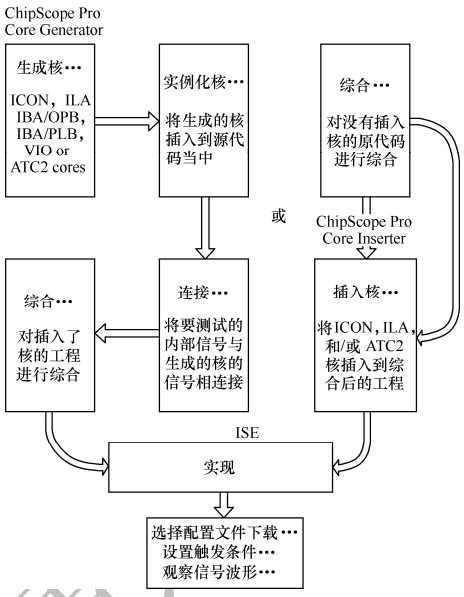


图 6.35 ChipScope Pro 设计流程

6.7.3 ChipScope Pro Core Inserter 简介

ChipScope Pro Core Inserter 的启动有两种方式。

（1）直接在 Windows 环境下运行“开始”/“程序”/“ChipScope Pro 8.2i”/“ChipScope Pro Core Inserter”命令。运行后即可得到 ChipScope Pro Core Inserter 的用户界面，如图 6.36 所示。

（2）可以通过新建资源的方法，如图 6.37 所示。

新建 ChipScope Pro Inserter 资源后，系统自动生成扩展名为 cdc 的文件。如图 6.38 所示，双击扩展名为 cdc 的文件即可启动 ChipScope Pro Inserter 界面。需要注意的是，在双击扩展名为 cdc 的文件时，系统会先对该工程文件进行综合。综合完成后才会启动 ChipScope Pro Inserter。

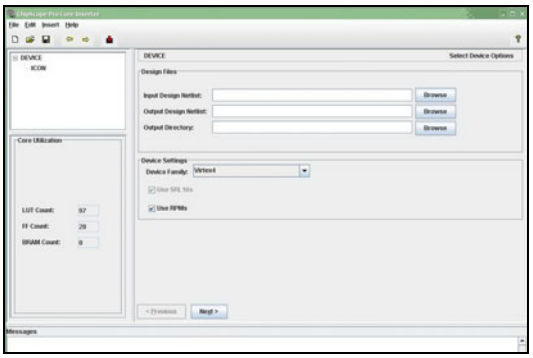


图 6.36 ChipScope Pro Core Inserter 用户界面

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

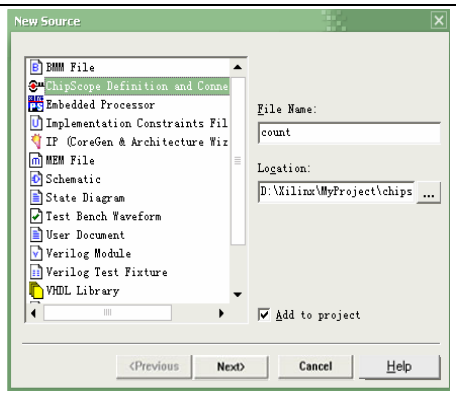


图 6.37　新建 ChipScope Pro Inserter 资源对话框

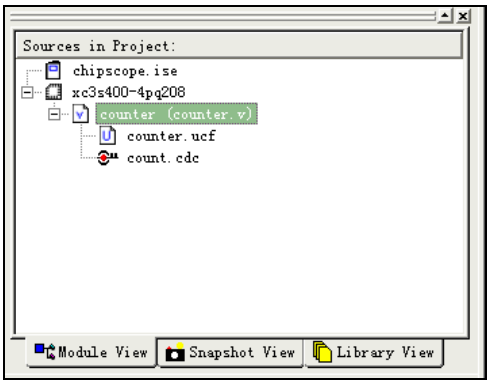


图 6.38　新建 ChipScope Pro Core Inserter 资源后的界面

这里在综合前必须先对综合属性进行设置，如图 6.39 所示，在综合属性对话框中对“synthesis options”/“keep Hierarchy”选项进行设置。设置“keep Hierarchy”为“Yes”或“Soft”。

双击扩展名为 cdc 的文件，系统完成综合后，会自动启动 ChipScope Pro Core Inserter。设计者通过 ChipScope Pro Core Inserter 对触发单元个数、触发宽度、触发条件、存储深度、采样时刻等参数进行设置。设置完毕后，在 ISE 下完成布局布线，下载配制文件，即可用 ChipScope Pro Analyzer 进行观测。

下面对 ChipScope Pro Core Inserter 的各项设置做详细的说明。

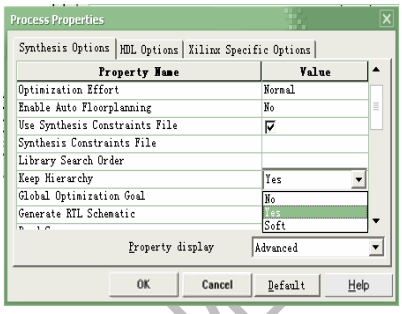


图 6.39　综合属性设置对话框

1．用户界面

启动 ChipScope Pro Core Inserter 后，显示如图 6.40 所示的界面。在“Input Design Netlist”文本框中设置输入设计网表的路径。设置好后，“Output Design Netlist”和“Output Directory”会自动生成，设计者也可自己指定。

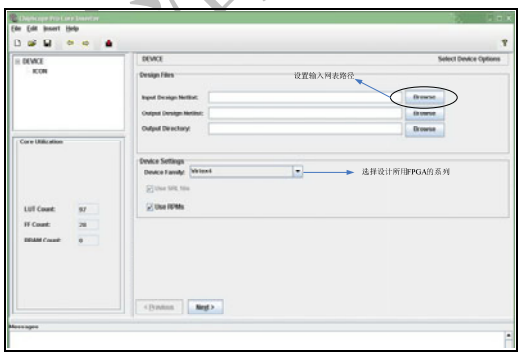


图 6.40　ChipScope Pro Core Inserter 用户界面

如果是通过新建资源的方法启动 ChipScope Pro Core Inserter，这几项显示为灰色，无需设计者设置，系统会自动找到设计网表文件。在“Device Family”下拉列表中选取设计所用的 FPGA 后，就可单击“Next”按扭，进入“Select Integrated Controller Options”对话框，如图 6.41 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

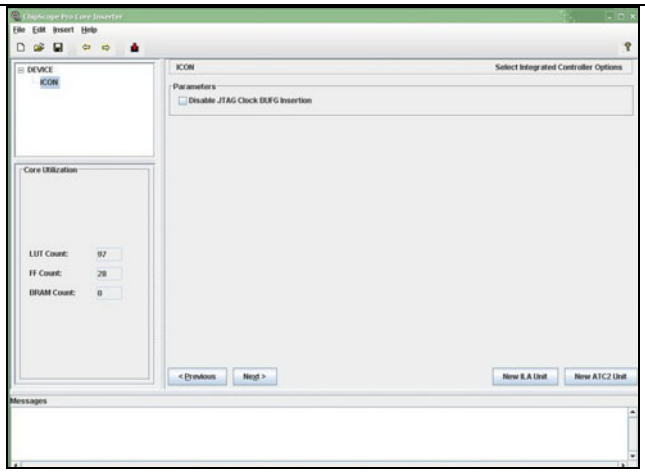


图 6.41 “Select Integrated Controller Options”对话框

2.“Select Integrated Controller Options”对话框设置

在如图 6.40 所示的“Select Device Options”对话框中，可以指定是否禁止在 JTAG 时钟上插入 BUFG。如果选中此项，JTAG 时钟将使用普通布线资源，而不是全局时钟布线。这样会在 JTAG 时钟线上产生较大的布线延时。因此在全局时钟资源足够用的情况下，应该尽量使 JTAG 时钟使用 BUFG 资源。即使由于全局时钟资源不够而不得不禁用 BUFG 时，也最好附加相应约束，使延迟抖动尽量小。推荐设计者在使用时不选此项。

单击“Next”按钮，进入“Select Integrated Logic Analyzer Options”对话框，如图 6.42 所示。

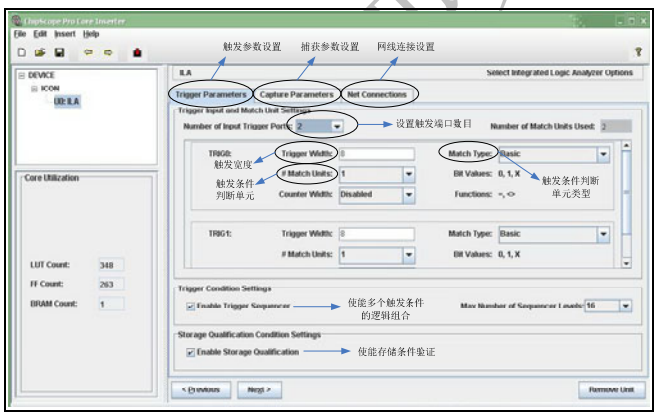


图 6.42 “Select Integrated Logic Analyzer Options”ILA 对话框

3.“Select Integrated Logic Analyzer Options”对话框设置

如图 6.42 所示，可以看到在“Select Integrated Logic Analyzer Options”ILA 对话框下有 3 个选项卡，可对触发参数、捕获参数、网线连接进行设置。

“Trigger Parameters”选项卡可对触发端口数目，每个触发端口的宽度、触发条件判断单元、触发条件判断单元的个数和类型等进行设置。

（1）触发端口数目。

在设计中可以根据需要设置多个触发端口，每个 ILA Core 最多可以有 16 个输入触发端口，每个触发端口下又可设置多个触发条件判断单元，但各个触发端口包含的触发条件判断单元数量之和不能大于 16。

（2）触发端口设置。

一个完整的触发端口设置包括：触发宽度、触发条件判断单元个数及类型的设置。触发宽度是指触发端口包含信号线的个数。通过触发条件判断单元进行判断，当信号线上的信号满足设定的条件时，ChipScope Pro 就可将其捕获并存储在 BlockRam 中，用于在 ChipScope Pro Analyzer 中显示波形。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

对触发条件可以设置个数和类型。当有多个触发条件时，可以将触发条件设置为几个触发条件的逻辑组合。触发条件判断单元实际为比较器，其类型可以有以下几种，如表 6.4 所示。

表 6.4　　触发条件判断单元的类型

类　型	数值类型	匹 配 功 能	Bit/Slice	说　明
Basic	0、1、X	=、<>	8	用于一般信号比较，是一种节约资源的类型
Basic(w/trans)	0、1、X、R、F、B	= 、 <> 、 transition detection	4	用于控制信号的比较，可以检测跳变的发生
Extend	0、1、X	=、<>、>、>=、<、<=	2	当主要考虑数据的大小时，用于地址或数据信号大小的比较
Extend(w/trans)	0、1、X、R、F、B	=、<>、>、>=、<、<= 、 transition detection	2	当数据和地址信号的大小和跳变都需要考虑时，可以用于检测跳变的发生
Range	0、1、X	=、<>、>、>=、<、<=、in range、not in range	1	当数据和地址的大小需要考虑时，可以用于检测数值是否在一定范围内
Range(w/trans)	0、1、X、R、F、B	=、<>、>、>=、<、<=、in range、not in range 、 transition detection	1	当数据和地址的大小和跳变都需要考虑时，可以用于检测跳变和数值是否在一定范围内

“Capture Parameters”选项卡可以对存储深度、采样时刻等参数进行设置，如图 6.43 所示。

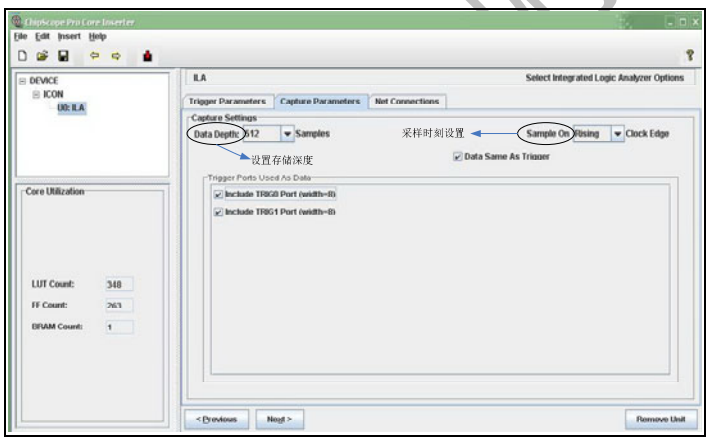


图 6.43　“Capture Parameters”选项卡设置

所谓存储深度，是指在满足触发条件后，要存储多少数据，用于最终的波形显示。ChipScope Pro 可能的最大存储深度为 16384，最大数据位宽为 256bit。实际的数据存储深度和位数由 FPGA 内部剩余的 BlockRam 的数量决定。

对于“Data Same As Trigger”选项，有时要观测的信号就是设置的触发条件中的信号，此时选中此项即可。有时设定了触发条件后，想观察别的数据信号，这时可以不选中此项，数据与触发信号完全独立。

“Net Connections”选项卡可以设置触发端口信号线与要观测的信号的连接，要观测哪些信号，就将这些信号与端口的信号线连接即可，如图 6.44 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

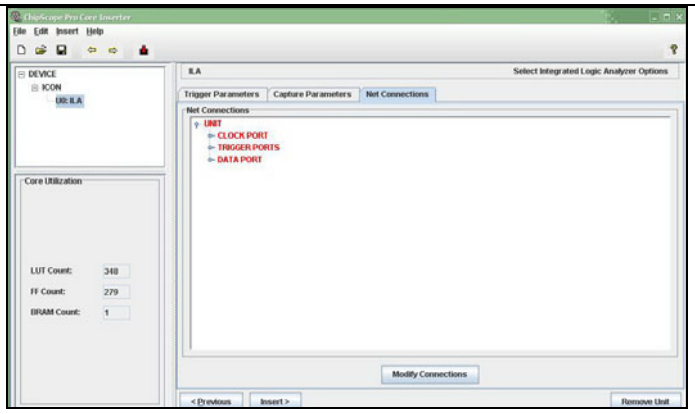


图 6.44 “Net Connections”选项卡设置

设置的连接信号可以分为 3 类:时钟信号(CLOCK PORT)、触发端口信号(TRIGGER PORTS)和数据信号(DATA PORT)。单击“Modify Connections”按钮会出现如图 6.40 所示对话框。

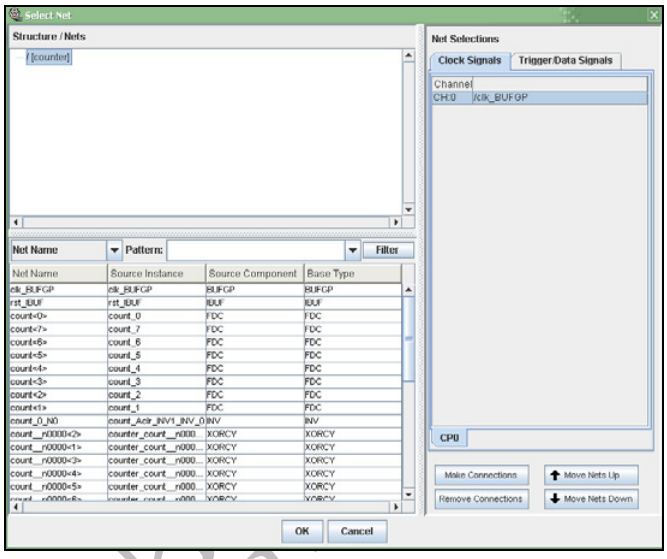


图 6.45 网线连接对话框

设置完所有信号后，端口名字会变为黑色，否则为红色。设置完上述各项后，单击“Inserter”按钮，逻辑分析仪的网表就插入到原来的设计网表当中。之后在 ISE 下完成布局布线并下载后，就可以用 ChipScope Pro Analyzer 进行观测了。

6.7.4 ChipScope Pro Analyzer 简介

将逻辑分析的核插入设计当中后，就可以运行 ChipScope Pro Analyzer 进行观测了，ChipScope Pro Analyzer 的启动方式有两种。

（1）直接运行“开始”/“程序”/“ChipScope Pro 8.2i”/“ChipScope Pro Analyzer”。

（2）在 ISE 下启动。

如图 6.46 所示，双击“Analyze Design Using ChipScope”即可启动，ChipScope Pro Analyzer 界面如图 6.47 所示。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

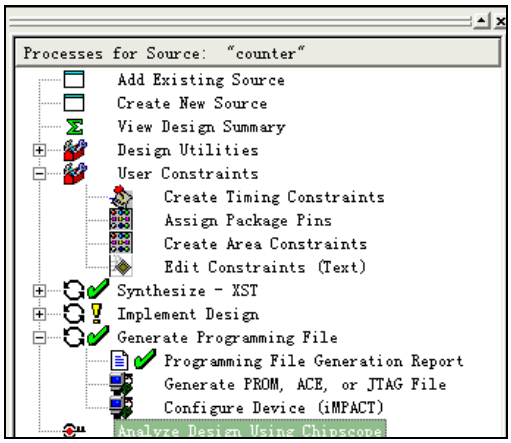


图 6.46　从 ISE 中直接启动 ChipScope Pro Analyzer

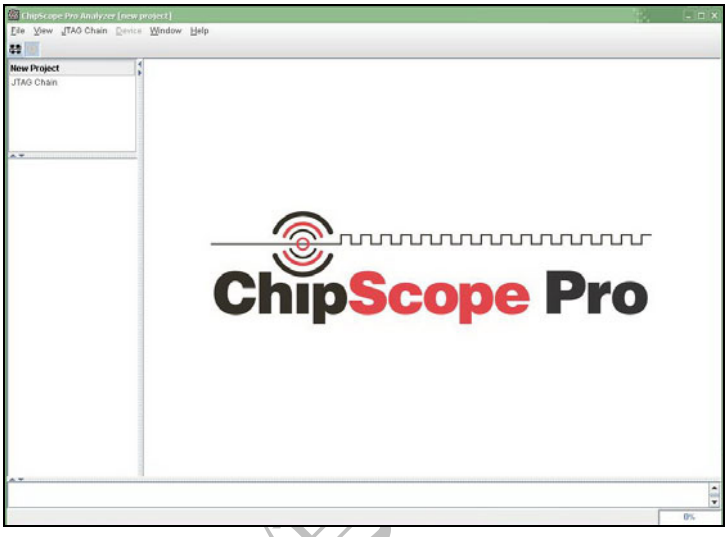


图 6.47　ChipScope Pro Analyzer 用户界面

ChipScope Pro Analyzer 使用步骤如下。

1．单击图标，打开 JTAG 并口连接电缆

在此之前要保证已将 JTAG 与器件连接好，如果连接无误，会出现如图 6.48 所示的对话框。
对话框中会显示 JTAG 连接的 FPGA 类型和所用的配置器件类型，这里使用的 FPGA 为 Spartan 3 系列，配置器件选用的是 XCF02S。

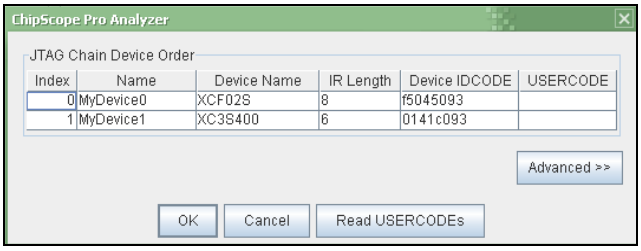


图 6.48　JTAG 正常连接后提示

2．下载配置文件

在 ISE 下完布局布线后，生成配置文件*.bit 文件。注意：ChipScope Pro 采用 JTAG 方式观测 FPGA 内部信号，这就要求在生成下载文件时。在“Generate Programming File”的属性对话框（如图 6.49 所示）中设置“Startup

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

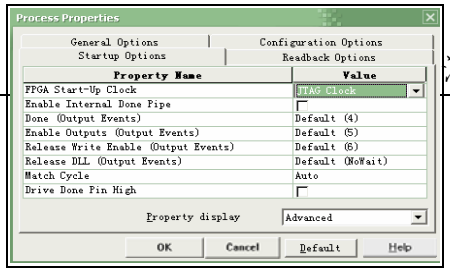


图 6.49　Generate Programming File 属性设置对话框

Options”/“FPGA Start-Up Clock”为 JTAG Clock，否则 ChipScope Pro 将无法正确配置器件。

下载配置文件时，选择“Device”/“DEV1”/“Configure”选项，如图 6.50 所示。单击后会出现如图 6.51 所示的对话框，选择要下载的*.bit 文件，对 FPGA 进行配置。

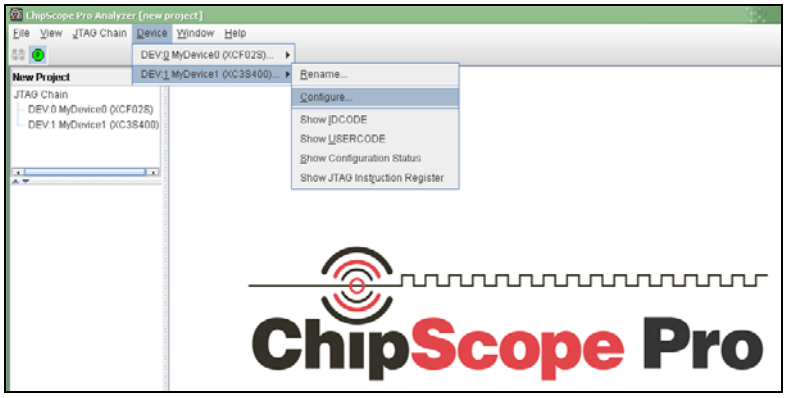


图 6.50　配置 FPGA

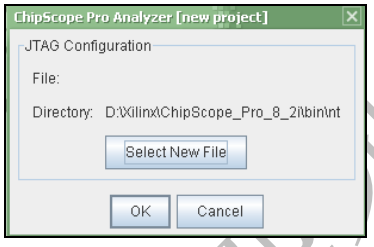


图 6.51　选择配置文件对话框

3．设置触发条件

成功完成对 FPGA 的配置后，会出现如图 6.52 所示界面。

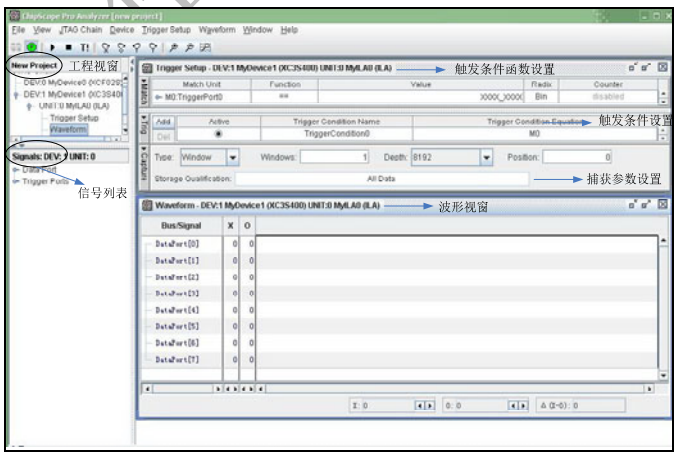


图 6.52　成功完成配置后用户界面

如图 6.52 所示，ChipScope Pro Analyzer 的界面由两部分组成。

左边一栏为工程视窗和信号列表。工程视窗的下拉列表中有“Trigger Setup”、“Waveform”选项。双击后，就会有相应的视窗在右边显示。信号列表中列出了所有信号，在这里可以增加或删除视图中的信号，对信号重命名，也可以将信号组合为总线以便于观察。

右边一栏主要有两个视窗：一个为“Trigger Setup”，用于设置触发条件；一个为“Waveform”用于观察波形。设置触发条件包括设置触发条件函数（Match）、触发条件（Trig）和捕获参数（Capture），下面分别介绍。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

（1）“Match”选项卡。

主要完成触发条件函数的设置。所谓触发条件函数是与表 6.4 中的匹配功能相对应的。要设置数值大小和函数，即选择：=、<>、>、>=、<、<=某数值时，才满足触发条件。当有多个条件时，可以分别设置每个条件的要求满足的函数和数值，如图 6.53 所示。

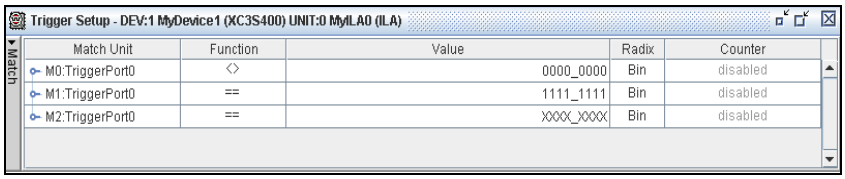


图 6.53 “Match”选项卡设置对话框

（2）“Trig”选项卡。

主要用于设置触发条件。在“Match”下设置了触发所需要满足的条件。当有多个条件时这里可以设置是让哪一个条件起作用，也可以将条件设置为几个条件的逻辑组合。或者是将几个条件设置为“条件链”，即当依次满足条件链设置的各个条件后，才可以捕获数据，如图 6.54 所示。

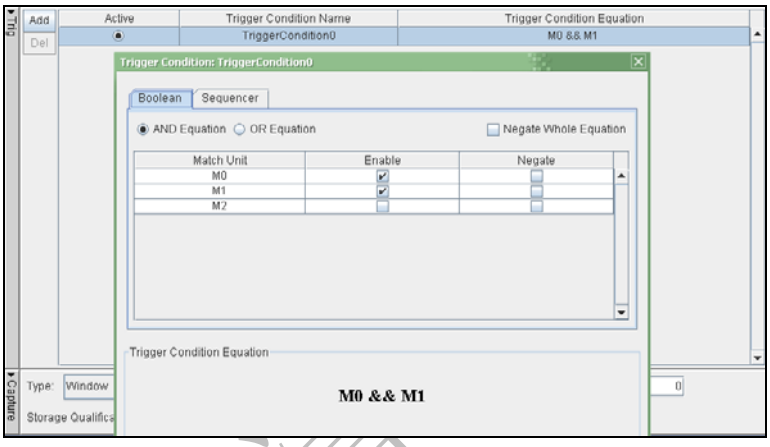


图 6.54 “Trig”选项卡设置对话框

如图 6.54 所示，单击“Trigger Condition Equation”下的选项，会弹出如图 6.55 所示的对话框。在对话框中，有两个选项卡。在“Boolean”选项卡下，可以设置哪一个条件起作用，也可将条件设置为几个条件的逻辑组合；在“Sequencer”选项卡下，可以设置条件链，如图 6.55 所示。当依次满足条件链下的几个条件后，就被触发。

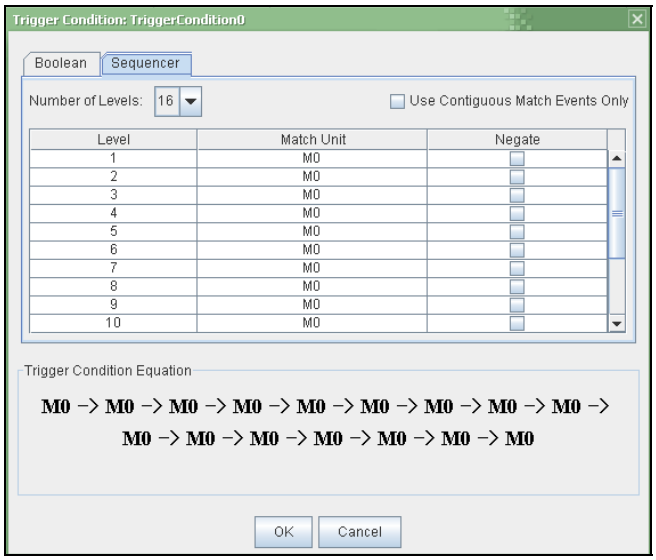


图 6.55 “Sequencer”选项卡设置对话框

开发板免费领取试用 开发板免费领取试用

4．观测波形

触发条件设置好后，单击左上角的按钮开始执行。当满足触发条后，ChipScope Pro 开始采集数据，采集到一定数目后（该数目取决于存储深度），就可以观察波形了。如图 6.56 为 ChipScope Pro 显示波形的效果图。

在波形显示窗口下可对波形进行放大和缩小，也可以进行局部放缩。当要观察总

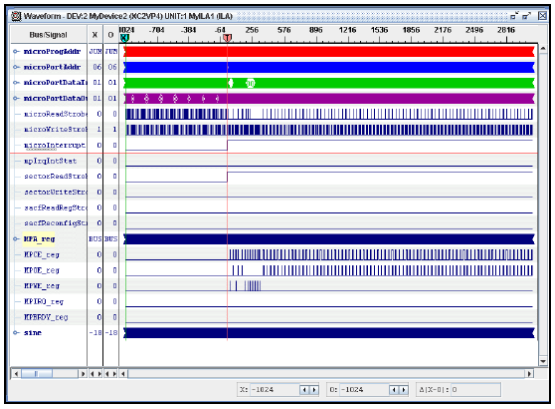


图 6.56　ChipScope Pro 波形显示效果图

线数据时，可以先选中所有总线数据，然后单击右键，选择“Add to Bus”/“New Bus”即可，如图 6.57 所示。

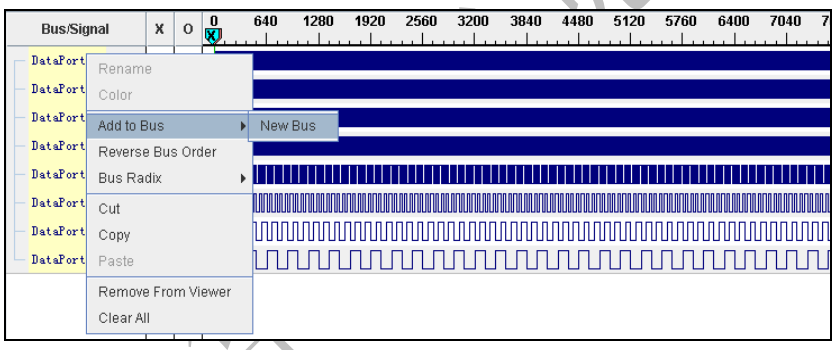


图 6.57　组合总线数据

添加成功后，就可以在新生成的 bus 下看到总线数据。如图 6.58 所示，图中 DataPort 为新生成的总线，设计者可以根据需要修改总线的名称。

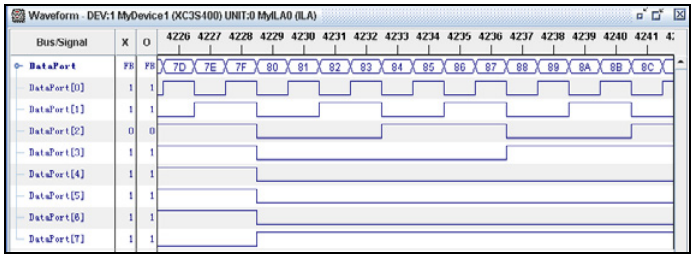


图 6.58　总线数据生成结果

6.7.5　小结

本节对在线逻辑分析工具 ChipScope Pro 作了详细介绍。通过本节的学习，读者应该掌握 ChipScope Pro 的设计流程以及使用方法。

ChipScope Pro 本身的功能很强大，在调试阶段应用很广泛。它可以观察 FPGA 内部的任何信号，使用也比较简单，希望读者能够很好地掌握。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

为了让读者能够尽快地熟悉这一工具的使用，在 6.8 节给出一个简单的实例，分别应用两种设计流程来完成，读者可参照这一例程来进一步熟悉 ChipScope Pro 的使用。

6.8 典型实例 11：ChipScope 功能演示

6.8.1 实例的内容及目标

1．实例的主要内容

本节通过一个简单的计数器，使用 ChipScope 的两种实现流程，基于 Xilinx 开发板完成设计至验证的完整过程。本实例的工作环境如下。

- 设计软件：ISE 7.1i。
- 综合工具：ISE 自带的 XST。
- 仿真软件：ModelSim SE 5.8C。
- 在线调试：ChipScope Pro 8.2i。
- 硬件平台：红色飓风 II 代 Xilinx 开发板。
- 实例内容：计数器。通过 ChipScope Pro 观测计数器的计数值，代码参见本书实例代码的“典型实例 11”文件夹。其中 count_new 文件夹对应采用流程 1 实现的工程，count 文件夹对应用流程 2 实现的工程。

使用 ChipScope Pro 进行在线调试主要有两种实现流程。

（1）基于 ChipScope Pro Core Generator 的实现流程。

- 调用 ChipScope Pro Core Generator 生成逻辑分析仪的网表文件。
- 修改用户 RTL，插入逻辑分析仪代码。
- 综合，实现，下载 bit 配置文件。
- 调用 ChipScope Pro Analyzer 观察波形。

（2）基于 ChipScope Pro Core Inserter，通过新建资源的实现流程。

- 对工程文件进行综合，生成网表文件。
- 调用 ChipScope Pro Core Inserter，插入逻辑分析核。
- 布局、布线，生成 bit 配置文件并下载。
- 调用 ChipScope Pro Analyzer 观察波形。

由上面可以看出，两种实现方式的主要区别在于生成逻辑分析核的方式不同。本实例将分别采用这两种方式应用 ChipScope Pro 观测 FPGA 内部信号。

为了便于读者的理解，本实例将编写一个简单的计数器，通过 ChipScope Pro 观测计数的数值。代码的功能仿真结果如图 6.59 所示。

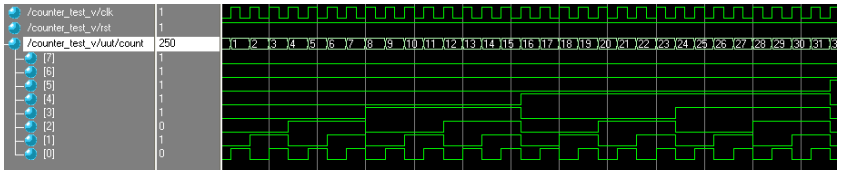


图 6.59 计数器功能仿真结果

其中，count 为 8 位计数值，采用循环计数方式。

2．实例目标

通过本实例，读者应达到下面的目标。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

- 熟悉 ChipScope 工具的使用方法。
- 熟悉基于 ChipScope Pro Core Generator 的实现流程。
- 熟悉基于 ChipScope Pro Core Inserter 的实现流程。

6.8.2 基于 ChipScope Pro Core Generator 的实现流程

基于 ChipScope Pro Core Generator 实现流程的具体步骤如下。

（1）新建工程，添加源代码。

新建 ISE 工程后，将源代码文件 counter_new.v 及约束文件 counter.ucf 添加至工程中，如图 6.60 所示。

（2）启动 ChipScope Pro Core Generator。

运行“开始”/“程序”/“ChipScope Pro 8.2i”/“ChipScope Pro Core Generator”命令，启动 ChipScope Pro Core Generator，出现如图 6.61 所示界面。

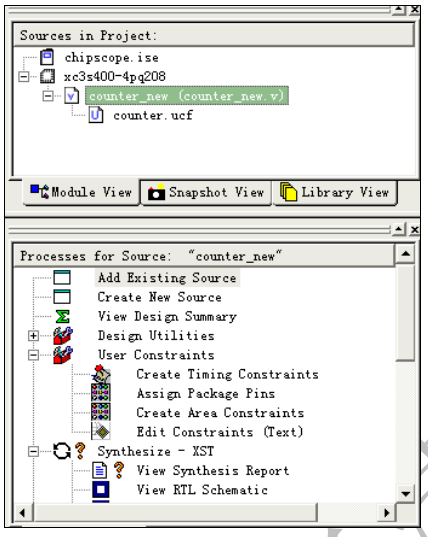


图 6.60 新建工程结果

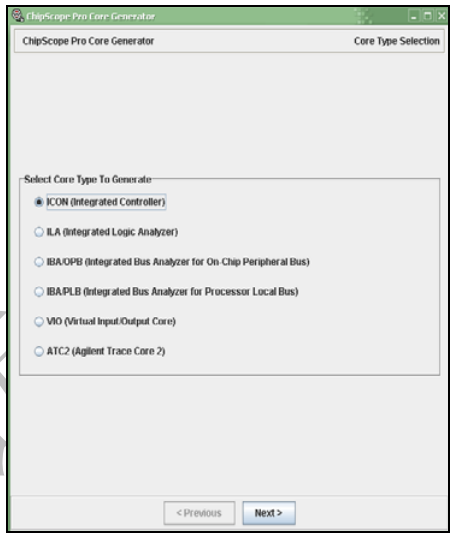


图 6.61 ChipScope Pro Core Generator 用户界面

（3）选择集成控制核（ICON 核）。

首先在核类型选择页面中选择生成 ICON 核，如图 6.61 所示。

（4）选择目录及器件。

如图 6.62 所示，在本实例中选择 Spartan3 器件。输出路径选择新建工程所在的文件夹。控制端口的数目为 1，控制端口的数目可以根据用户的需要设置。如需要观察多组数据时，可以设多组控制端口。这里只需要观察内部计数器的计数值，因此设置为 1。

（5）选择语言类型和综合工具。

如图 6.63 所示，本实例中设置语言类型为 Verilog，综合工具为 Xilinx XST。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

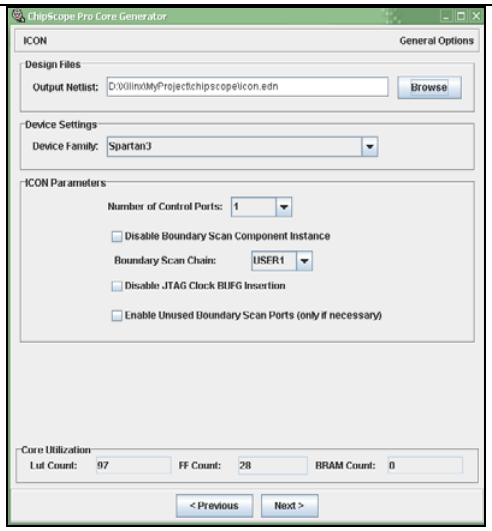


图 6.62　ICON 输出路径和 FPGA 系列设置对话框

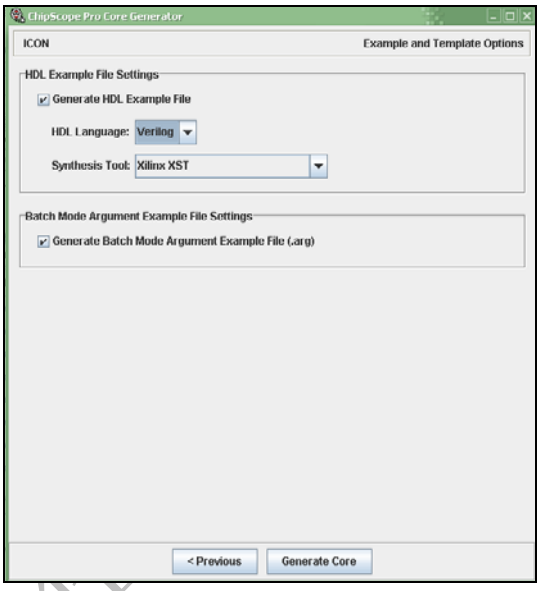


图 6.63　设置语言和综合工具

（6）生成 ICON 核。

如图 6.64 显示为生成的 ICON 核的相关信息，如发现有误，可以单击“Previous”按钮做修改后重新生成。

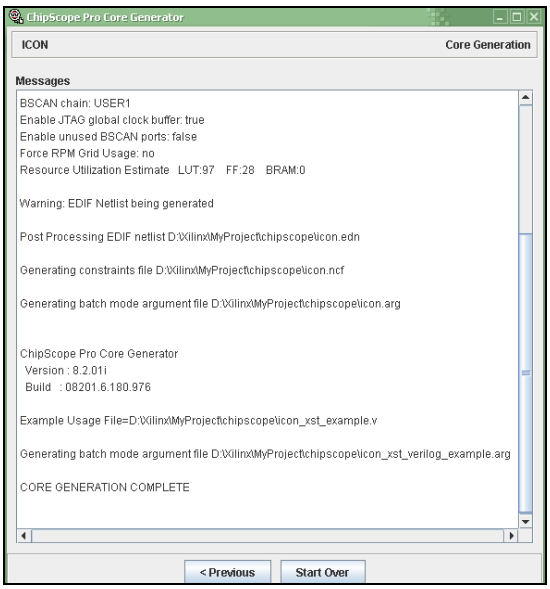


图 6.64　生成 ICON 核

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

（7）选择集成逻辑分析仪（ILA 核）。

生成 ICON 核后，单击“Start Over”按钮，回到核类型选择页，如图 6.65 所示，选择需要生成的核为 ILA。

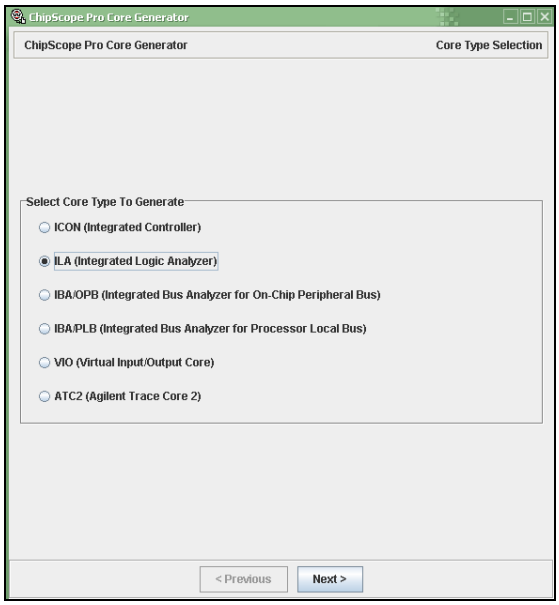


图 6.65　生成集成逻辑分析仪

（8）设置输出路径、器件参数和时钟参数。

如图 6.66 所示为设置路径为新建工程所在路径，器件为 Spartan3，采样时刻为时钟的上升沿。

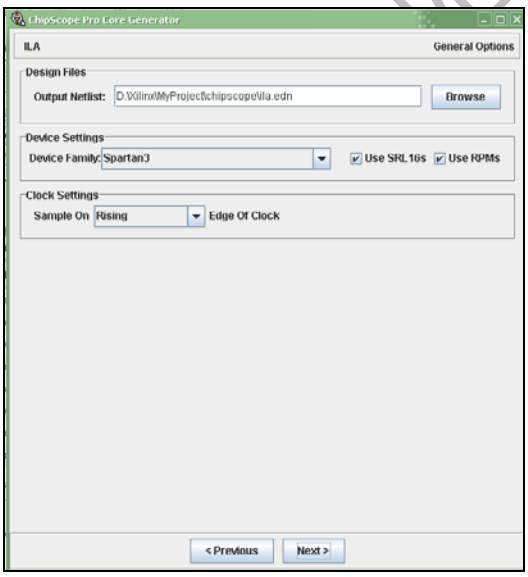


图 6.66　设置 ILA 输出路径、FPGA 器件、采样时刻对话框

（9）设置触发参数。

触发参数的设置包括：触发端口数目（Number of input Trigger ports）、每个触发端口的触发宽度（Trigger Width）、触发条件判断单元个数（Match Units）和类型（Match Type）等。

此外，“Enable Trigger Sequencer”选项用于使能触发条件链。即设置触发条件为一“条件链”，只有依次满足“条件链”上的各个条件时才会被触发。如图 6.67 所示，如有与图中参数设置不一致之处，请自行修改。

（10）设置存储深度和数据位宽。

存储深度即在满足触发条件后要采集多少数据，存储深度的大小由 FPGA 的 RAM 资源大小决定。由于 ChipScope 所采集的数据都是保存在 FPGA 内部，因此存储深度的大小不能超过 FPGA 的 RAM 的最大值。如图 6.68 所示，选择“Data Same As Trigger”选项表示数据信号与触发信号相同，数据位宽即为触发端口的触发宽度。

（11）设置语言类型和综合工具。

如图 6.69 所示，本实例中选择 Verilog 及 Xilinx XST。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

（12）生成 ILA 核。

单击图 6.69 中的“Generate Core”按钮，生成逻辑分析仪（ILA），如图 6.70 所示。

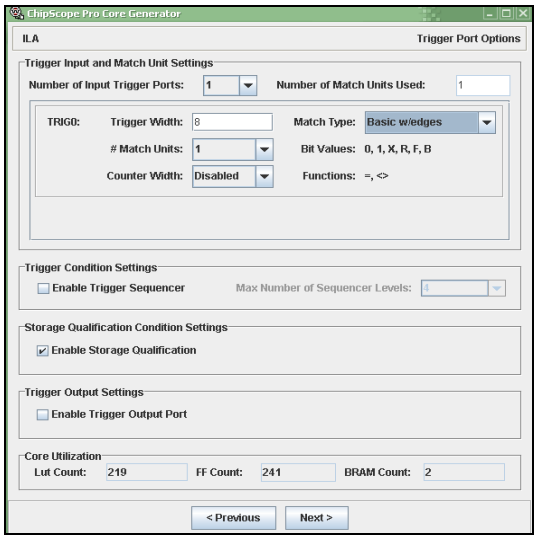


图 6.67　设置触发参数对话框

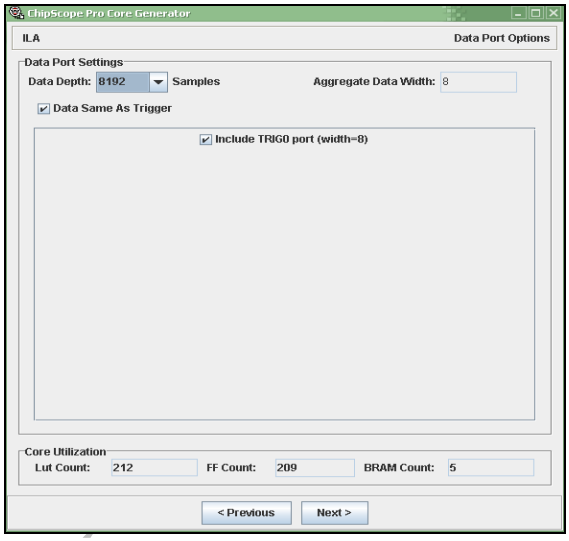
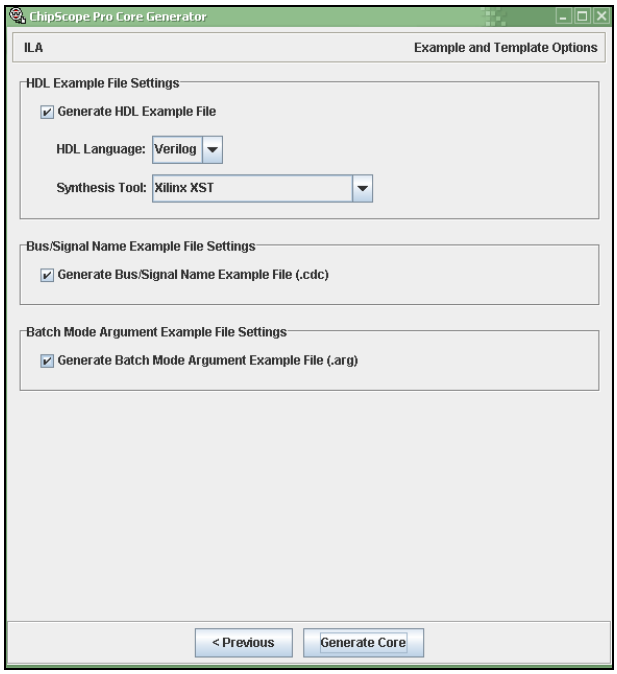


图 6.68　设置存储深度和数据位宽对话框



华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

图 6.69　设置生成实例的语言类型和综合工具

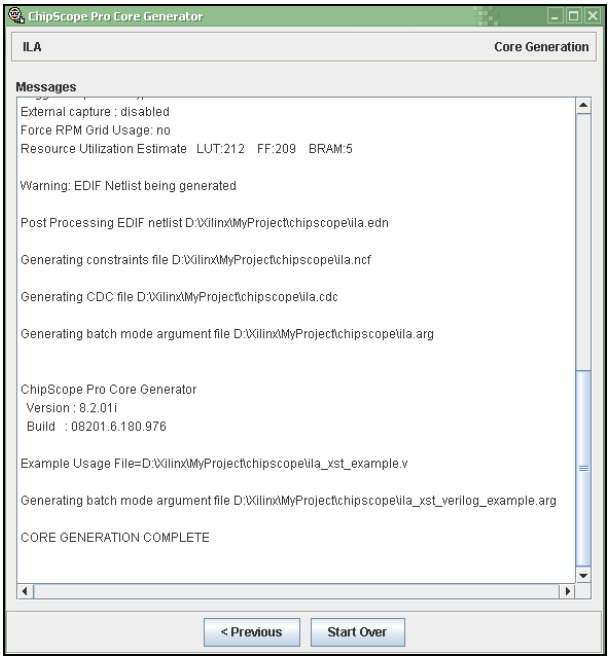


图 6.70　生成 ILA 核

（13）将 ICON 和 ILA 插入设计。

完成上述步骤后就完成了集成控制核（ICON）和逻辑分析仪（ILA）的生成，系统生成的主要文件如表 6.5 所示。

表 6.5　　ChipScope 生成文件列表

icon.edn	集成控制器的网表文件
icon.ncf	集成控制器的网表约束文件
icon_xst_example.v	集成控制器的嵌入式例子代码
ila.end	集成逻辑分析仪的网表文件
ila.ncf	集成逻辑分析仪的网表约束文件
ila_xst_example.v	集成逻辑分析仪的嵌入式例子代码

用户需要做的是根据提供的例程代码来修改自己的代码，将生成的集成控制核和集成逻辑分析仪插入到设计当中。具体修改时，需要在源文件中添加如下代码：

```
    icon i_icon                 //  ICON core instance,实例化 ICON 核
      (
        .control0 (control0)
      );
  wire [7:0] trig0;
  ila i_ila                     //  ILA core instance,实例化 ILA 核
    (
      .control (control0),
      .clk (clk),
      .trig0 (trig0)
    );
  assign trig0 = count;
```

从代码中可以看出要做的工作主要是实例化 ICON 核和 ILA 核。注意要将观察的信号（在这里为 count）与 ILA 核的输入信号相连接，系统时钟与 ILA 核的时钟输入相连接，ICON 的输出控制信号与 ILA 的输入

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

控制信号相连接。这样就可以通过修改 RTL 代码来插入集成逻辑控制器和集成逻辑分析仪了。

（14）综合，布局布线，生成配置文件并下载。

具体的操作步骤可以参看 2.6 节的介绍，这里不再详述。需要注意的是 ChipScope Pro 要通过 JTAG 接口与器件连接。生成配置文件时，时钟要设置为 JTAG Clock。

（15）启动 ChipScope Pro Analyzer。

可通过直接运行“开始”/“程序”/“ChipScope Pro 8.2i”/“ChipScope Pro Analyzer”，也可以在 ISE 集成环境下，在进程浏览器中双击“Analyze Design Using ChipScope”启动。启动后界面如图 6.71 所示。

（16）打开 JTAG 连接。

单击图标，打开 JTAG 并口连接电缆，在此之前要保证已将 JTAG 与器件连接好，如果连接无误，正常连接后会出现如图 6.72 所示的界面。

（17）设置触发条件。

触发条件设置如图 6.73 所示。

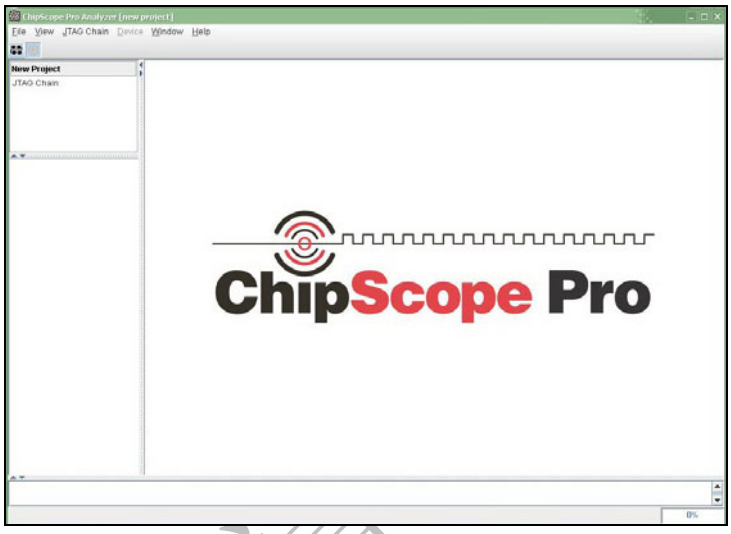


图 6.71　ChipScope Pro Analyzer 用户界面

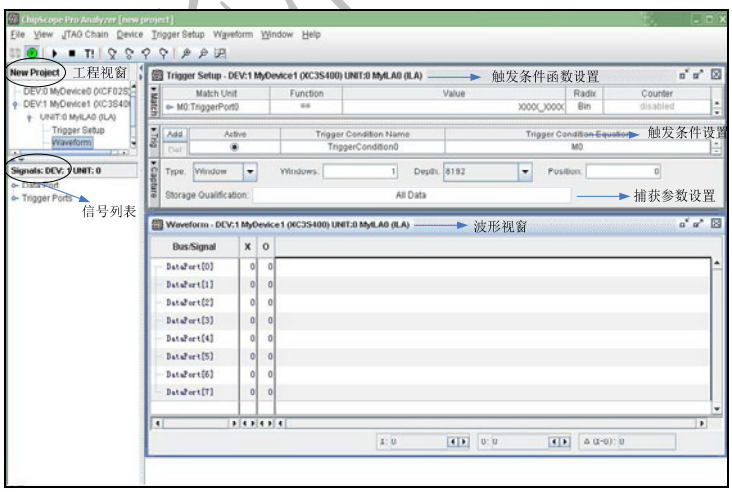


图 6.72　成功连接后用户界面

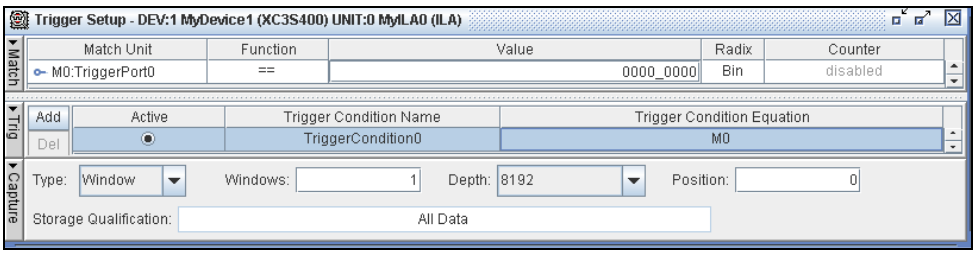


图 6.73　设置触发条件对话框

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

设置触发条件函数为“==”，数值为 0000_0000（也就是说，触发端口的信号值为 0000_0000 时开始采集数据）。设置采集深度为 8192。这里只有一个触发件 M0，只需采用默认即可。当有多个条件时，要在“Trig”选项卡下设置起作用的正确条件，也可将触发条件设置为几个条件的逻辑组合。

（18）开始采样调试。

单击左上角的运行按扭，开始采样。捕获触发条件后，ChipScope 将采集所设置存储深度的波形，如图 6.74 所示。

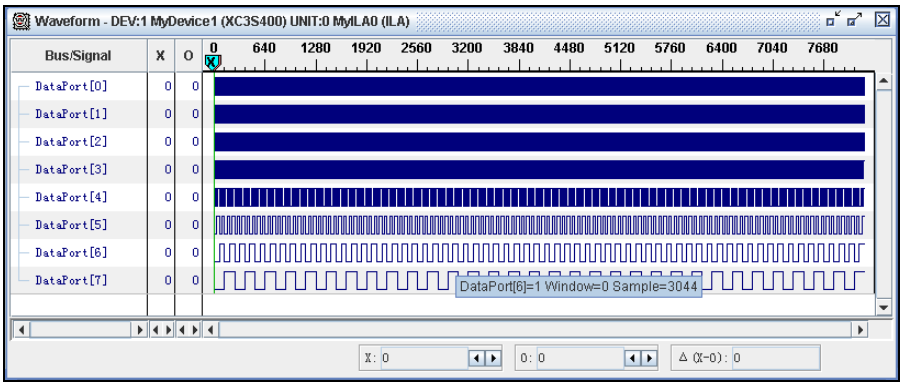


图 6.74　采集信号的波形图

在波形显示窗口下可对波形进行放大和缩小，也可以进行局部放缩。当要观察总线数据时，可以先选中所有总线数据，然后单击右键，选择“Add to Bus”/“New Bus”即可如图 6.75 所示。

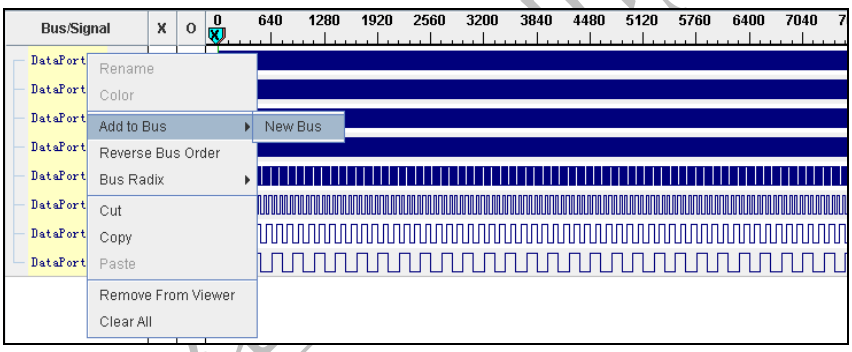


图 6.75　组合总线数据

添加成功后，就可以在新生成的 Bus 下看到总线数据，如图 6.76 所示。图中 DataPort 为新生成的总线，设计者可以根据需要修改总线的名称。

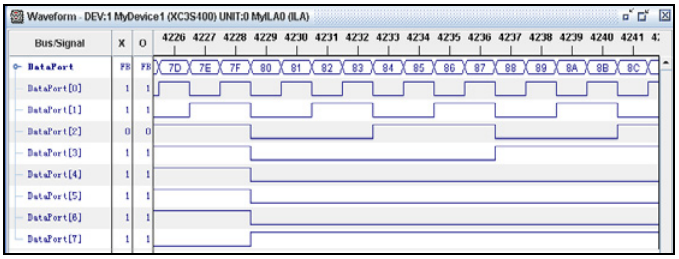


图 6.76　总线数据生成结果

通过上述步骤就完成了应用 ChipScope Pro 的一个完整的流程，这种流程是通过修改源代码来添加逻辑分析仪的。下面的流程可以不用修改设计代码，直接将逻辑分析仪插入到设计的网表文件中，使用方便，应用比较广泛。

6.8.3　基于 ChipScope Pro Core Inserter 的实现流程

基于 ChipScope Pro Core Inserter 实现流程的具体步骤如下。

（1）新建工程，添加源代码。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

参见第一种实现流程的同一步骤。

（2）新建 ChipScope Pro 资源。

选择 ISE 的“Project”/“New Source”，再选择“ChipScope Definition and Connection”选项，输入新建资源的名称：count，生成 count.cdc 资源文件。如果有现成的扩展名为 cdc 的文件，也可以通过“Project”/“Add Source”来添加，如图 6.77 所示。

完成后在工程浏览器中可以看到包含的 count.cdc 文件，如图 6.78 所示。

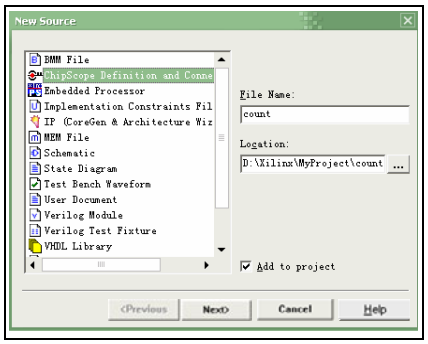


图 6.77 新建.cdc 资源对话框

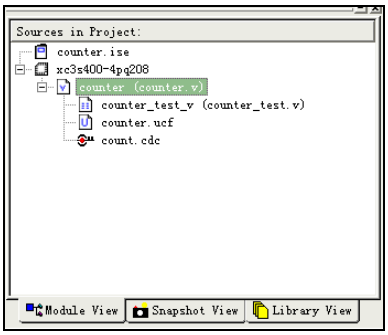


图 6.78 新建.cdc 资源结果

（3）综合。

与前一种设计流程不同，这里要先对源文件进行综合，在这种设计流程中，ILA 核是直接插入到设计综合后生成的逻辑网表中的，因此要先对源文件进行综合。

（4）设置 ICON 及 ILA 各项参数。

在工程浏览器中双击 count.cdc 文件，打开 ChipScope Pro Core Inserter 工具，如图 6.79 所示。

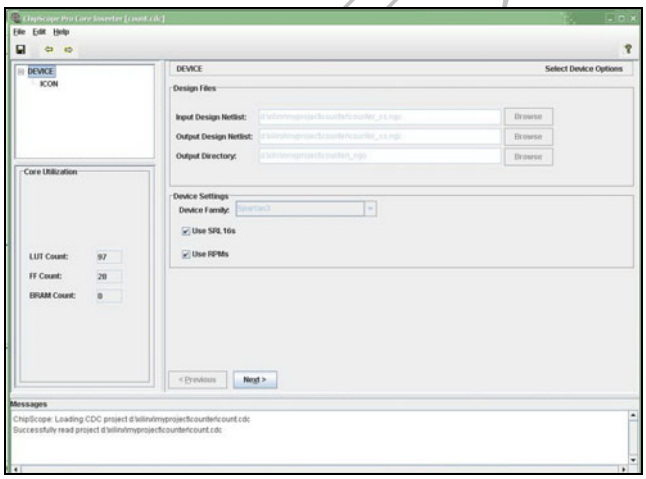


图 6.79 ChipScope Pro Core Inserter 用户界面

在图中可以看到，这里无需再设置输入网表文件的路径，系统会自动找到网表文件，并设置输出文件的路径。

单击“Next”按钮进入 ICON 核设置页面，如图 6.80 所示。

开发板免费领取试用 开发板免费领取试用

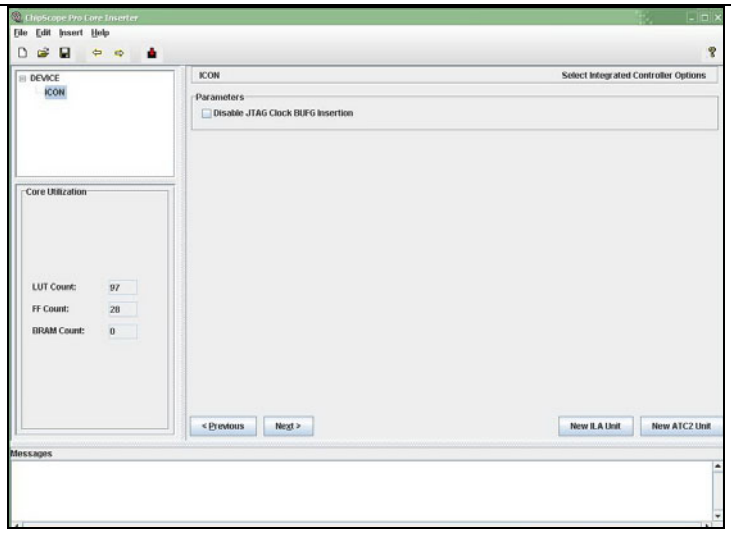


图 6.80 “Select Integrated Controller Options”ICON 对话框

其中，“Disable JTAG Clock BUFG Insertion”用于指定是否禁止在 JTAG 时钟上插入 BUFG。如果选中此项，JTAG 时钟将使用普通布线资源，而不是全局时钟布线。这里此项不选，用全局时钟布线。

完成 ICON 核的设置以后，选择单击“Next”按钮，进入 ILA 核设置页面，如图 6.81 所示。

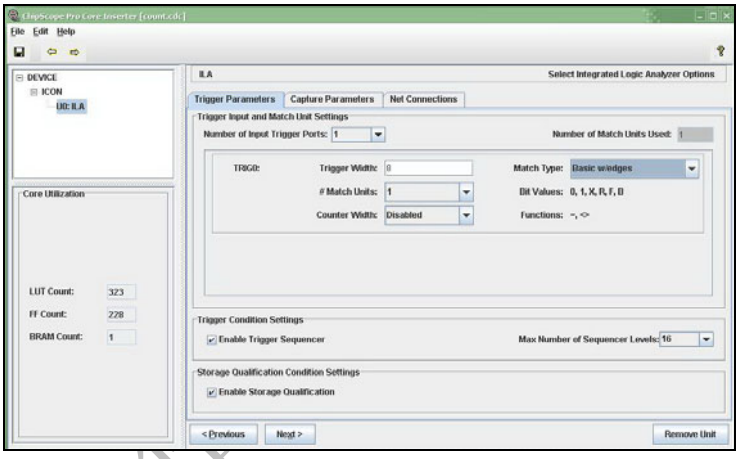


图 6.81 “Trigger Parameters”选项卡设置

在触发器设置页面中，设置触发端口数、触发宽度、触发条件单元函数和个数等。如有不一致，请读者对照上图修改，设置完毕后单击“Next”按钮出现如图 6.82 所示对话框。

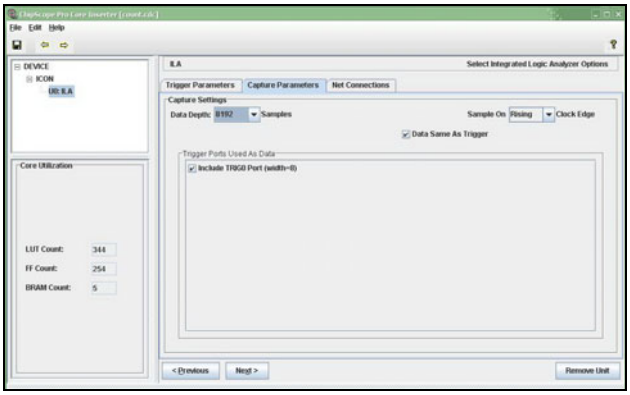


图 6.82 “Capture Parameters”选项卡设置

如图 6.82 所示，设置存储深度为 8192，采样时刻为上升沿。选中“Data Same As Trigger”，即数据信号与触发信号相同。设置完毕后单击“Next”按钮，可以看到已经定义的 ILA，如图 6.83 所示。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

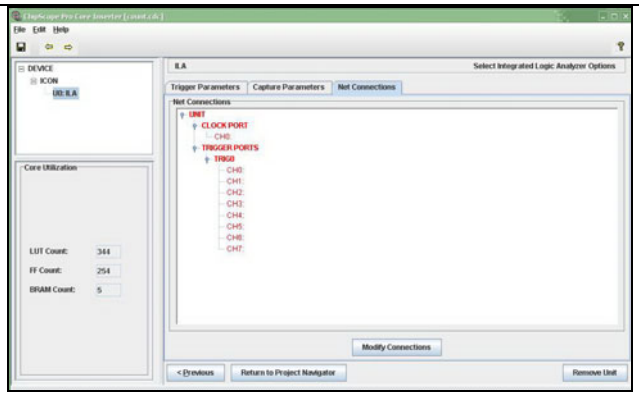


图 6.83 “Net Connection”选项卡设置

此时由于还没有对网表进行映射，因此网络线都是呈红色显示。单击“Modify Connection”按钮，设置时钟信号和 TRIG 信号与要观测的信号的映射关系，如图 6.84 及 6.85 所示。

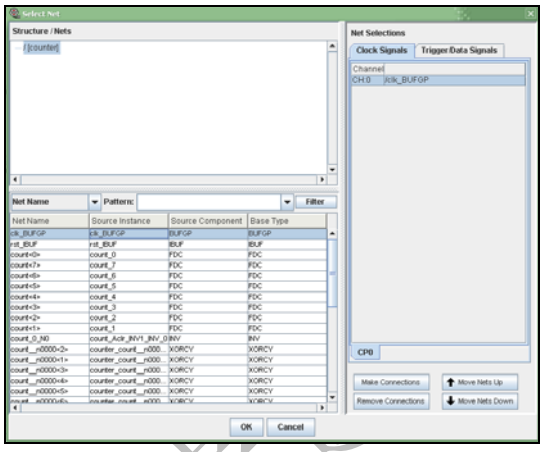


图 6.84 设置时钟信号连接

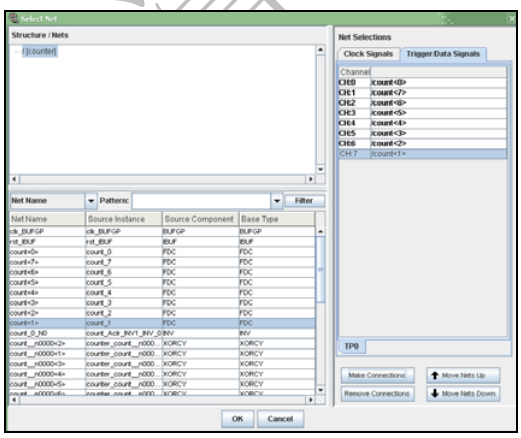


图 6.85 设置数据连接

信号连接主要完成时钟信号的连接和触发信号的映射。将 clk_BUFGP 与 Clock Signals 中的 CH0 连接，将 count<0>～count<7>的内部信号与 Trigger Data Signals 的 CH0～CH7 相连。连接方法很简单，选中要连接的信号，单击“Make Connections”按钮即可。

当所有信号都连接完毕，“Net Connection”选项卡下显示的网路线会由红色变为黑色。如果仍有信号未连接，则仍为红色，直到所有信号连接完毕。

设置完毕后，单击“Return To Project”。此时逻辑分析仪已经插入到了工程网表文件当中了，不需要用户再对源代码进行修改。

（5）布局布线，生成配置文件并下载。

对工程进行综合、布线布局，之后生成配置文件并下载。

（6）启动 ChipScope Pro Analyzer 进行观察。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

启动 ChipScope Pro Analyzer 后，剩下的步骤与前一种方法完全一致，这里就不再重复说明。

6.8.4 小结

本节通过一个具体的实例，详细演示了 ChipScope Pro 的两种使用方法。希望读者能够按照这两种流程动手练习一下，熟练掌握这一工具的使用。

在具体应用当中，第二种方法应用得比较广泛，因为无需修改源代码，而且修改逻辑分析仪的各项参数也很方便，因此推荐读者使用第二种方法。ChipScope Pro 是个功能很强大的在线逻辑分析工具，熟练地掌握它的用法并应用于自己的设计当中，将会给调试过程带来极大的方便。

6.9 典型实例 12：增量式设计（Incremental Design）演示

6.9.1 实例的内容及目标

1．实例的主要内容

6.7 节对增量式设计这一方法的基本概念和流程做了全面的介绍。本节将以一个具体的实例帮助读者熟悉增量式设计的操作流程。

本实例的源代码参见随书光盘 Example6.9。此程序为 PC 机通过串口向 SRAM 写入数据，再由 FPGA 从 SRAM 中读取数据通过串口将其送到 PC 机。

本实例的重点在于设计过程中是如何应用增量式设计的，而不是如何实现程序本身的功能。

2．实例目标

通过本训练，读者应达到下面的目标。

- 掌握增量式设计与一般设计的区别。
- 掌握增量式设计的设计流程。

6.9.2 实例详解

增量式设计的具体实现步骤如下。

（1）打开 ISE 工程。

设计的第一步为创建逻辑分组。

在本设计中，top.v 为顶层模块。顶层模块中包含两个功能模块，分别为：uart_rs232.v 和 sram.v。uart_rs232.v 用于完成串口数据传输，sram.v 用于对于 SRAM 的基本读写操作。

top.v 顶层模块中，仅包含这两个模块，没有其他复杂逻辑。并且每个逻辑分组均以寄存器输出，可将这 3 个模块看作 3 个逻辑分组，满足创建逻辑分组所必须遵循的原则。

（2）综合。

为了保证在后面的实现中能够准确地完成分组区域约束，这里需要对综合的属性进行设置，在“Processes for Source”中选择“Synthesize-XST”，单击鼠标右键，设置综合属性如图 6.86 所示。

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

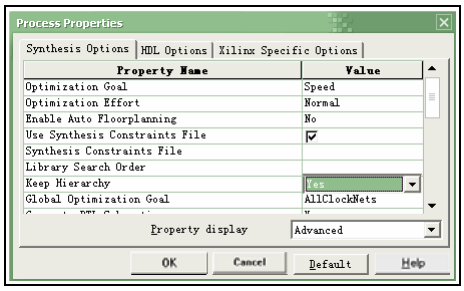


图 6.86　设置综合属性对话框

选择“Synthesis Options”/“Keep Hierarchy”，设置综合属性为保留结构层次模式。

综合完毕要查看综合报告，为了和下面流程中的增量综合结果作对比，请特别注意综合报告中如图 6.87 所示的部分。在没有进行增量综合时，要对每个模块都进行综合和优化。

```
=========================================================================
*                          Low Level Synthesis                          *
=========================================================================
WARNING:Xst:2183 - Unit sram: the following tristate(s) are NOT replaced by l

Optimizing unit <top> ...

Optimizing unit <uart_rs232> ...

Optimizing unit <sram> ...
Loading device for application Rf_Device from file '3s400.nph' in environment

Mapping all equations...
Building and optimizing final netlist ...
Found area constraint ratio of 100 (+ 5) on block top, actual ratio is 4.
FlipFlop rs232/DRI_counter_13 has been replicated 1 time(s)
FlipFlop rs232/DRI_state_FFd3 has been replicated 1 time(s)
FlipFlop rs232/REV_counter_13 has been replicated 2 time(s)
FlipFlop rs232/REV_state_FFd3 has been replicated 2 time(s)
```

图 6.87　未设置综合约束条件时的综合报告

（3）设置管脚约束。

在“Processes for Source”中选择“Assign Package Pins”启动设置管脚约束的工具 PACE，如图 6.88 所示，在 Loc 处设置各信号对应的 FPGA 管脚。具体设置可参见例程代码 Source 文件夹下的 top.ucf 文件。

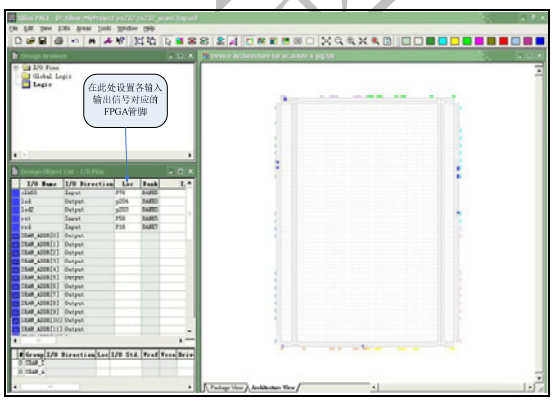
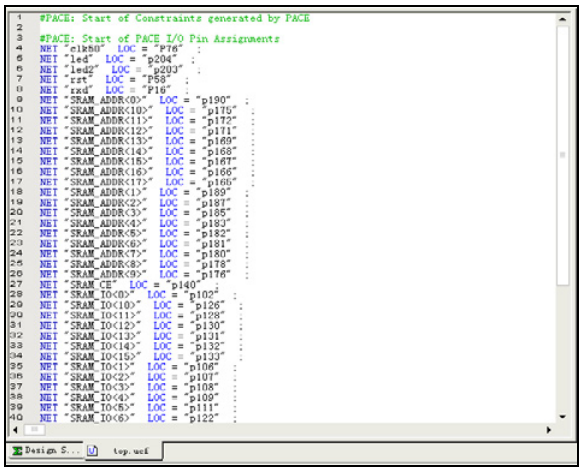


图 6.88　设置管脚约束界面

设置好管脚约束后，保存退出 PACE。在“Processes for Source”中选择“Edit Constraints Text”，即在 Text 模式下编辑约束文件。可以看到系统自动生成的管脚约束文件内容，如图 6.89 所示。



华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

图 6.89　系统生成的管脚约束文件

（4）设置分组区域约束。

在“Processes for Source”中选择“Create Area Constraints”，启动设置分组区域约束工具 PACE。分别选中 sram 和 rs232，单击左上角的图标，为每一个逻辑分组设置区域约束，如图 6.90 所示。

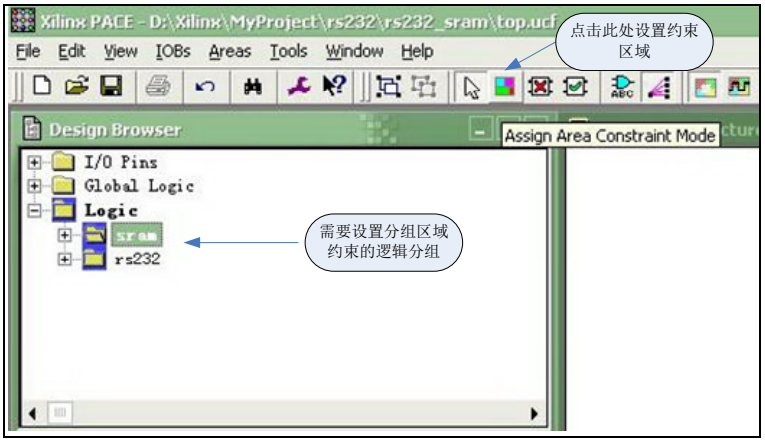


图 6.90　进入分组区域约束模式

设置区域约束结果如图 6.91 所示。

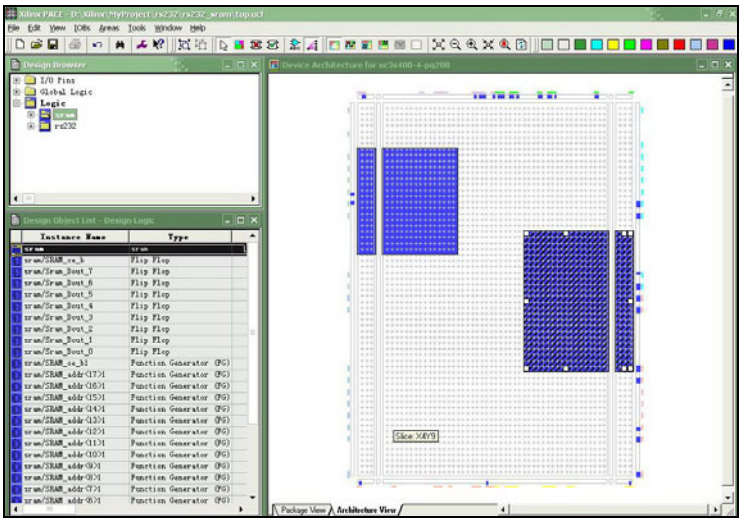


图 6.91　区域约束设置结果

如图 6.91 所示，图中左上角的区域对应的是 SRAM 逻辑分组的约束区域，右下角的区域为 rs232 逻辑分组的约束区域。采用如上的约束是综合两个逻辑分组内部逻辑的复杂性以及 I/O 位置来确定的。这里的约束并不惟一，读者可根据需要进行修改。

设置完毕后，保存设置。在“Processes for Source”中选择“Edit Constraints Text”，即在 text 模式下编辑约束文件，可看到在约束文件中新添了如下内容：

```
#PACE: Start of PACE Area Constraints
AREA_GROUP "AG_rs232" RANGE = SLICE_X0Y55:SLICE_X19Y38 ;//设置 rs232 逻辑分组区域
INST "rs232" AREA_GROUP = "AG_rs232" ;
AREA_GROUP "AG_sram" RANGE = SLICE_X34Y41:SLICE_X55Y18 ;//设置 SRAM 逻辑分组区域
INST "sram" AREA_GROUP = "AG_sram" ;
```

（5）在普通模式下对工程进行映射和布局布线。

在普通模式下（采用 ISE 下默认的实现属性）对工程进行映射和布局布线是为了得到初始的指引文件，用于在后面的增量设计中指引映射和布局布线。

首先在“Processes for Source”中选择“Implement Design”/“Map”，对工程进行映射。实现后系统会生成：top_map.ncd 和 top_map.ngm 文件，需要用这些文件来指引后面的增量设计的映射。为了避免系统将此文件覆

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

盖，将其改名为：top_map_guide.ncd 和 top_map_guide.ngm 文件。

接着在“Processes for Source”中选择“Implement Design”/“Place&Route”，对工程进行布局布线，会得到 top.ncd 文件。也将其改名为：top_guide.ncd，将其作为后面增量设计时布局布线的指引文件。

> **注意** 如果在此步骤中无法顺利地完成映射和布局布线等步骤，很可能是区域分组约束做得不合适。需要重新做区域分组约束，直到能够顺利完成映射和布局布线为止。映射完成后要查看映射报告看各逻辑分组的资源利用率。如果不合适，需要修改区域约束后重新进行映射和布局布线。映射报告需要注意的内容如图 6.96 所示。

完成映射和布局布线后的结果如图 6.92 所示：

（6）对工程进行增量综合。

如果在设计调试中发现了某个 Logic Group 需要修改，对其做微小的改动后，要对工程进行增量综合。例如可以修改 uart_rs232.v 的代码，然后进行增量综合。

本实例中的增量综合采用 ISE 自带的工具 XST。采用 XST 进行增量综合时，必须为其添加约束文件（扩展名为 xcf）。添加的约束文件可先在记事本中编辑，然后保存为扩展名为 xcf 的文件。在本实例中综合约束文件为 syn_constraint.xcf 文件，其内容如下：

```
MODEL "top" incremental_synthesis=yes;              //使用增量综合
MODEL "sram" incremental_synthesis=yes;             //使用增量综合
MODEL "uart_rs232" incremental_synthesis=yes;       //使用增量综合
MODEL "top" resynthesize=yes;                       //启动增量综合
MODEL "sram" resynthesize=no;                       //不启动增量综合
MODEL "uart_rs232" resynthesize=yes;                //启动增量综合
```

前 3 句设置各逻辑分组均采用增量综合，后 3 句通知哪个逻辑分组内容发生改变，需要重新综合，哪个逻辑分组内容未发生改变，无需重新综合。编辑完毕，将该文件添加到当前的工程当中，如图 6.92 所示，上面 syn_constranit.xcf 文件即为添加结果。

添加完毕后，要使这一约束文件有效，还要对综合属性进行设置，如图 6.93 所示。

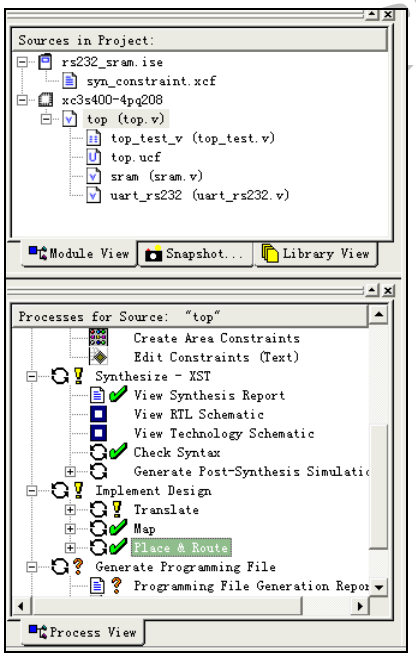


图 6.92　对工程进行映射和布局布线结果

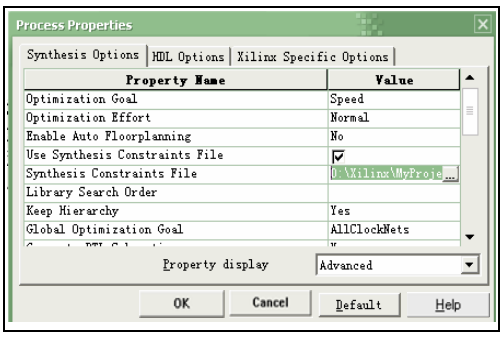


图 6.93　增量综合属性设置对话框

选择“Synthesis Options”/“Synthesis Constraints File”，根据约束文件所在位置，设置约束文件的路径。设置完毕后，对工程进行增量综合。综合完毕后，查看综合报告，注意如下内容。

将图6.94与图6.87进行比较，可以看出综合中仅对top和rart_rs232两个逻辑分组重新进行了综合和优化。SRAM逻辑分组保持不变（Unit <sram> is up to date），表明增量综合实现了。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

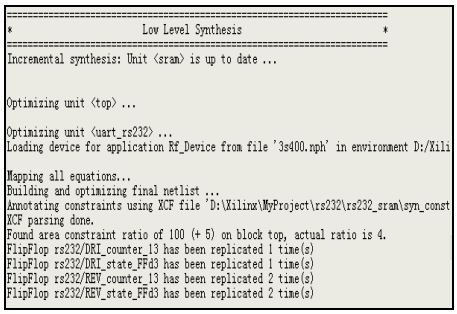


图 6.94　增量综合的综合报告图　　6.95　增量实现属性设置对话框

（7）对工程进行增量实现（Implement）。

完成增量综合后，就可以利用前面得到的初始的指引文件：top_map_guide.ncd 和 top_guide.ncd 文件来指引增量实现。

首先对工程实现的属性进行设置，具体设置如下。

单击右键，选择进程浏览器中的“Implement Design”的“Property”选项，弹出工程实现属性对话框，如图 6.95 所示。在增量设计属性页中做如下设置。

- 为“Enable Incremental Design Flow”使能增量实现。
- 为“Run Guided Incremental Design Flow”设置用指引文件来引导增量实现。
- 为“MAP Guide Design File（.ncd）”设置指引映射的指引文件 top_map_guide.ncd 路径。
- 为“PAR Guide Design File（.ncd）”设置指引布局布线的指引文件 top_guide.ncd 路径。

按上述方式设置完毕后，对工程进行映射和布局布线后，查看映射和布局布线报告，对于映射报告注意以下内容。

如图 6.96 所示，在映射报告中会对每个逻辑分组在各自约束的区域内的资源利用情况作一个总结报告。可以看到各分组所用的 LUT 和 Slices 占其约束区域总量的百分比。如果出现某一逻辑分组的使用率达 90%以上，而有些逻辑分组还不到 1%，则需要重新进行区域约束。

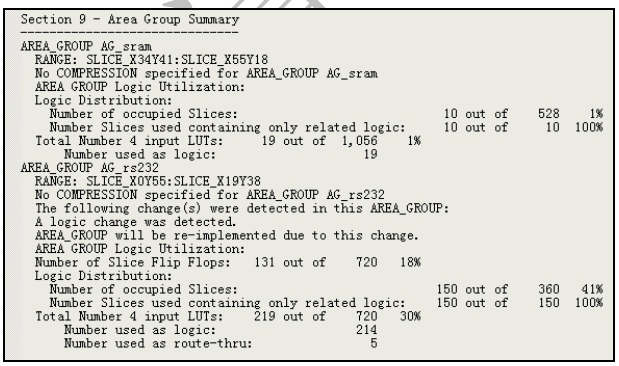


图 6.96　映射报告（Map Report）

区域约束最好在生成初始指引文件时就确认。在普通模式下完成映射后查看映射报告，查看每个逻辑分组的资源利用率。如是不合适，则更改区域约束，重新进行映射。

对于布局布线报告注意以下内容。

如图 6.97 所示，布局布线时读取了 top_guide.ncd 作为指引文件，仅有 rs_232 逻辑分组重新进行了布局布线（AG_rs232 was re-implemented）。SRAM 逻辑分组完全继承了前面已有的结果（AG_sram was fully guided），说明增量实现完成了。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

```
Loading database for application par from file:
"D:\Xilinx\MyProject\rs232\rs232_sram\top_guide.ncd"
   "top" is an NCD, version 3.1, device xc3s400, package pq208, speed -4

Finished Guide File Processing.

Xilinx Place and Route Guide Results File
=========================================

Guide Summary Report:

Incremental Design Totals:
  Area Groups Guided:     1 out of     2
  Ungrouped Logic was re-implemented.
  Area Group: AG_rs232 was re-implemented
  Area Group: AG_sram was fully guided.

For a detailed guide report refer to the "top.grf" file.

Device Utilization Summary:

   Number of BUFGMUXs                  1 out of 8      12%
   Number of External IOBs            37 out of 141    26%
      Number of LOCed IOBs            37 out of 37    100%

   Number of Slices                  160 out of 3584    4%
      Number of SLICEMs                0 out of 1792    0%
```

图 6.97　布局布线报告（Place&Route Report）

完成了布局布线后就可以下载调试了。如果仍需要改动，重复步骤（6）和步骤（7），直到设计符合要求为止。

6.9.3　小结

本节以一个具体的实例介绍了 ISE 下增量设计流程。希望读者能够按照上述步骤进一步熟悉 ISE 的增量设计流程，对增量设计有个比较全面的认识，最终将这种设计方法应用到自己的设计当中。

联系方式

集团官网：www.hqyj.com　　嵌入式学院：www.embedu.org　　移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn　　物联网学院：www.topsight.cn　　研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路银海大厦 A 座 8 层，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218

开发板免费领取试用　开发板免费领取试用

华清远见 FARSIGHT 始于2004 | 嵌入式培训专家

10年口碑积累，成功培养50000多名研发工程师，铸就专业品牌形象

华清远见的企业理念是不仅要做良心教育、做专业教育，更要做受人尊敬的职业教育。

《FPGA 应用开发入门与典型实例》（修订版）

作者：华清远见

专业始于专注 卓识源于远见

第 7 章 FPGA 系统设计的仿真验证

本章目标

- 掌握 FPGA 设计仿真验证的原理和方法
- 掌握 ModelSim 工具的仿真方法
- 掌握功能仿真和时序仿真的区别
- 掌握测试文件的编写方法

专业始于专注 卓识源于远见

开发板免费领取试用 开发板免费领取试用

7.1 FPGA 设计仿真验证的原理和方法

7.1.1 FPGA 设计仿真验证

严格来讲，FPGA 设计验证包括功能与时序仿真和电路验证。仿真是指使用设计软件包对已实现的设计进行完整测试，模拟实际物理环境下的工作情况。

功能仿真是指仅对逻辑功能进行测试模拟，以了解其实现的功能是否满足原设计的要求。仿真过程没有加入时序信息，不涉及具体器件的硬件特性，如延时特性等，因此也叫前仿真。它是对 HDL 硬件设计语言的功能实现能力进行仿真，以确保 HDL 语言描述能够满足设计者的最初意图。

时序仿真则是在 HDL 可以满足设计者功能要求的基础上，在布局布线后，提取有关的器件延迟、连线延时等时序参数，并在此基础上进行的仿真，也称为后仿真，它是接近真实器件运行的仿真。

如图 7.1 所示是 FPGA 设计仿真验证的基本方法。

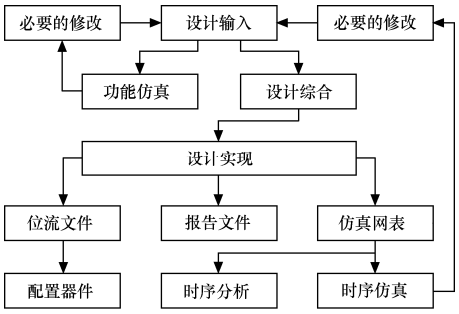


图 7.1 FPGA 设计仿真验证基本方法

7.1.2 FPGA 设计仿真的切入点

在 FPGA 的设计中，完整的设计流程往往会涉及多个 EDA 工具，比如设计输入工具、综合工具、布局布线工具、仿真工具等。如何将这些 EDA 工具进行适当的结合，在符合各个工具接口情况下发挥各个工具的特长，是每一个 FPGA 设计工程师都要面临的问题。

如图 7.2 所示是利用仿真工具 ModelSim 与综合工具 FPGA Compiler II 及布线工具 Foundation Series 或 Quartus 相配合实现 FPGA 设计的流程图。

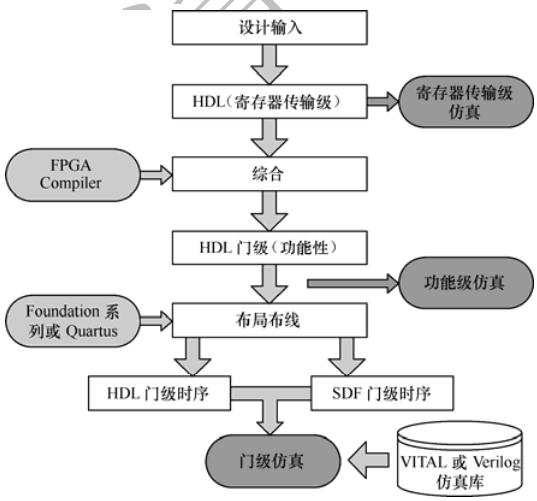


图 7.2 FPGA 设计仿真切入点

在设计输入阶段，由于 Modelsim 仅支持 VHDL 或 Verilog HDL，所以在选用多种设计输入工具时，可以使用文本编辑器完成 HDL 语言的输入。当然也可以利用相应的工具以图形方式完成输入，但必须能够导出对应的 VHDL 或 Verilog HDL 格式。

近年来出现的图形化 HDL 设计工具，可以接收逻辑结构图、状态转换图、数据流图、控制流程图及真值表等输入方式，并通过配置的翻译器将这些图形格式转化为 HDL 文件，如 Mentor Graphics 公司的 Renoir，Xilinx 公司的 Foundation 系列都带有将状态转换图翻译成 HDL 文本的设计工具。

从图 7.2 可以看出，在 FPGA 设计过程中，有 3 处可以使用 Modelsim 进行仿真。

（1）寄存器传输级（RTL）仿真。

此级仿真是对设计的语法和基本功能进行验证（不含时序信息）。在 RTL 仿真阶段，应该建立一个测试台。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

此测试台可以在整个 FPGA 流程中进行仿真验证（RTL 级、功能级、时序门级）。测试台不但提供测试激励与接收响应信息，而且可以测试 HDL 仿真流程中的关键功能（如运算部件输出值的正确性等）。测试台的产生可以直接使用文本编程得到，也可以使用图形化工具输入，再由软件翻译为 HDL 格式，例如使用 HDL Bencher 软件利用其良好的波形输入界面输入测试激励，再由其自动转化为 HDL 格式而得到。

（2）针对特定的 FPGA 专有技术的仿真。

此级仿真是在综合后、实现前而进行的功能仿真。功能仿真一般验证综合后是否可以得到设计者所需要的正确功能，而且功能仿真的结果往往与 RTL 级仿真结果相同。

（3）门级仿真。

此级仿真是针对门级时序进行的仿真，门级仿真体现出由于布局布线而产生的实际延时。在这个阶段，仿真的结果和设计者选择的 FPGA 型号有着直接的关系。由于不同的 FPGA 的内部结果有所差异，因此同一个设计在不同 FPGA 型号上得到的门级仿真结果也是不同的。

7.2 ModelSim 仿真工具简介

ModelSim 是 Model Technology（Mentor Graphics 的子公司）的 DHL 硬件描述语言的仿真软件，该软件可以用来实现对设计的 VHDL、Verilog 或者是两种语言混合的程序进行仿真，同时也支持 IEEE 常见的各种硬件描述语言标准。

无论从友好的使用界面和调试环境来看，还是从仿真速度和仿真效果来看，ModelSim 都可以算得上是业界最优秀的 HDL 语言仿真软件。它是惟一的单内核支持 VHDL 和 Verilog 混合仿真的仿真器，是做 FPGA/ASIC 设计的 RTL 级和门级电路仿真的首选；它采用直接优化的编译技术，Tcl/Tk 技术和单一内核仿真技术，具有仿真速度快，编译代码与仿真平台无关，便于 IP 核保护和加快程序错误定位等优点。

Modelsim 最大的特点是其强大的调试功能。

- 先进的数据流窗口，可以迅速追踪到产生错误或者不定状态的原因。
- 性能分析工具帮助分析性能瓶颈，加速仿真。
- 代码覆盖率检测确保测试的完备。
- 多种模式的波形比较功能。
- 先进的 Signal Spy 功能，可以方便地访问 VHDL、Verilog 或者两者混合设计中的底层信号。
- 支持加密 IP。
- 可以实现与 MATLAB 的 Simulink 的联合仿真。

目前常见的 Modelsim 分为几个不同的版本：ModelSim SE、ModelSim PE、ModelSim LE 和 ModelSim OEM。如图 7.3 所示，Modelsim 的用户界面和一般的 Windows 窗口相似，由上到下依次为：标题栏、菜单栏，工具栏，工作区和状态栏。

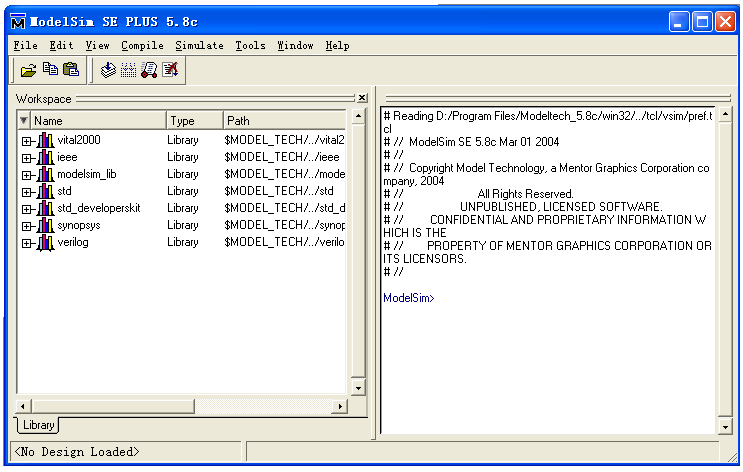


图 7.3 Modelsim 界面

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

7.2.1 标题栏

标题栏位于整个用户界面的最上方，标题栏的左边显示当前程序的名称，右边用来控制用户界面的大小，包括最小化、最大化和关闭 3 种功能按钮。

7.2.2 菜单栏

菜单栏里面一共有 8 个菜单选项，分别是：File（文件）、Edit（编辑）、View（视图）、Compile（编译）、Simulate（仿真）、Tools（工具）、Window（窗口）、Help（帮助）。下面分别具体介绍。

1.“File”菜单

文件菜单通常包含了对工程及文件等的操作。ModelSim 的文件菜单包含的命令有：New（新建）、Open（打开）、Close（关闭）、Import（导入）、Save（保存）、Delete（删除）、Change Directory（更改路径）、Transcript（对脚本进行管理）、Add to Project（为工程添加文件）、Recent Directories（最近几次的工作路径）、Recent Projects（最近几次工程）、Quit（退出）。

（1）新建文件命令（File/ New）。

单击 File/ New 命令，将会出现一个子菜单，共包含 4 个选项。

- 单击“Folder”（新建文件夹）后，会出现对话框，提示输入新建的文件夹的名字，即可在当前目录下新建一个文件夹。
- 单击“Source”（新建源文件）后，会出现源文件类型的选项（VHDL、Verilog、Other），单击可分别新建对应格式的源文件。
- 单击“Project”（新建工程）后，会出现对话框，提示在“Project Name”处输入新建工程的名称，在“Project Location”处指定新建工程的存放路径。在“Default Library Name”处指明默认的设计库的名称，用户设计的文件将编译到该库中。
- 单击“Library”（新建一个库）后，会出现对话框，提示选择“Create a New library and a logical mapping to it”（新建一个库并建立一个逻辑映像）或“A map to an existing library”（新建一个到已存在库的映像）。在“Library name”处输入新建库的名称，在“Library physical name”处输入存放库的文件名称。

（2）Open（打开文件）。

单击会出现子菜单选择打开 File（文件）、Project（工程）及 Dataset（WLF 文件）。

（3）Close（关闭）。

单击会出现子菜单选择关闭 Project（工程）或 Dataset（仿真数据文件）。

（4）Import（导入）。

导入新的库。在进行某些仿真时需要的一些仿真库可以通过该方法导入，根据提示指定仿真库的路径及目标库路径，一步一步操作完成。注意 ModelSim 安装目录下的 modelsim.ini 文件不能为只读，该文件保存了 ModelSim 的一些设置信息

（5）Save（保存）。

保存当前仿真数据。

（6）Delete（删除）。

删除指定的工程，即删除.mpf 文件，.mpf 是 ModelSim 工程的后缀名。

（7）Change Directory（改变路径）。

改变当前工作路径，ModelSim 使用的是绝对路径，而不是相对路径。这与 ISE 不同。在 ISE 中，用户可以将设计的整个目录复制到其他任何地方，只要目录完整，可以直接打开工程文件。而在 ModelSim 中，若将整个目录复制到其他地方，打开工程时其指向仍为原来工程的地址，可以通过更改路径来设置新的路

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

径。

（8）Transcript（脚本）。

单击会出现子菜单，可选择操作 Save Transcript（保存主窗口中脚本）、Save Transcript As（把主窗口中脚本另存为一个新文件）或 Clear Transcript（清除主窗口中的脚本）。

（9）Add to Project（添加到工程）。

单击会出现子菜单，可选择操作 File（添加文件到当前工程）、Simulation Configuration（添加设定的仿真配置）或 Folder（添加文件夹）。

（10）Recent Directories（最近几次工作路径）。

可以从中选取最近几次的工作路径。

（11）Recent Projects（最近几次工程）。

可以打开最近几次的工程。

（12）Quit（退出）。

退出 Modelsim。

2.“Edit”菜单

类似于 Windows 应用程序，在编辑菜单中包含了对文本的一些常用的操作。

（1）Copy（复制）。

复制选中的文档

（2）Paste（粘贴）。

把剪切或复制的文档粘贴到当前插入点之前。

（3）Select All（全选）。

选中主窗口中所有的抄本文档。

（4）Unselect All（取消全选）。

取消已选文本的选中状态。

（5）Find（查找）。

在命令窗口中查找字符或字符串。

华清远见

3.“View”菜单

类似于其他 Windows 应用程序，视图菜单可以控制在屏幕上显示哪些窗口。

（1）All Windows（所有窗口）。

打开所有的 ModelSim 窗口，试一下该命令会发现 ModelSim 打开了许多窗口，包括波形窗口、信号列表窗口、源文件窗口等。

（2）Dataflow（数据流）。

打开 Dataflow 窗口，在该窗口中显示数据的流向。

（3）List（列表）。

打开列表窗口。

（4）Process（进程）。

打开过程窗口，该窗口显示了设计中的进程所在的位置。

（5）Signals（信号）。

打开信号窗口。该窗口显示了设计中所有信号的列表。

（6）Source（源文件）。

打开源文件窗口，可以在源文件窗口中显示设计中使用的源文件。

（7）Structure（结构）。

打开结构窗口，该窗口以列表方式显示了设计中所有到的结构，双击某一结构，可以查找定义该结构的语

开发板免费领取试用 开发板免费领取试用

句。

（8）Variables（变量）。

打开变量窗口，该窗口以列表方式显示了设计中定义的所有变量。

（9）Wave（波形）。

打开波形窗口，这是我们仿真时经常需要查看的窗口，在其中显示了输入和输出的波形。

（10）Dataset。

打开 Dataset 浏览器来打开、关闭、重命名或激活一个 Dataset。用户在使用的时候会发现没有什么变化，这时候可以看看 Workspace 窗口下是不是多了一个选项卡。该选项卡显示的内容与 Structure 窗口显示的完全相同。

（11）Coverage（覆盖率）。

查看仿真的代码覆盖率。

（12）Active Processes（活动的进程）。

当前正在执行的进程。

（13）Workspace（工作区）。

打开当前的工作区。

（14）Encoding（编码）。

以不同的编码查看。

（15）Properties。

显示工作区中选中对象的属性。

4.“Compile”菜单

（1）Compile（编译）。

把 HDL 源文件编译到当前工程的工作库中。

（2）Compile Options（编译选项）。

设置 VHDL 和 Verilog 编译选项，例如可以选择编译时采用的语法标准等。

（3）Compile All（全编译）

编译当前工程中的所有文件。

（4）Compile Select（编译选中的文件）

编译当前工程中的选中文件。

（5）Compile Order（编译顺序）

设置编译顺序，一般系统会根据设计对 VHDL 自动生成编译顺序，但对于 Verilog 需要指定编译顺序。

（6）Compile Report（编译报告）

有关工程中已选文件的编译报告。

（7）Compile Summary（编译摘要）

有关工程中所有文件的编译报告。

5.“Simulate”菜单

这里的编译及运行命令类似于 Visual Ctf 等高级语言的调试时候的命令。

（1）Simulate（仿真）。

装载设计单元。

（2）Simulation Options（仿真选项）。

设置仿真选项。

（3）Run（运行）。

- Run ***ns：在该仿真时间长度内进行仿真。若要改变长度，可在 Simulation Options 中设置或在工

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

具栏中修改。

- Run-All（运行所有仿真）：进行仿真，直到用户停止它。
- Continue（继续）：继续仿真。
- Run-Next（运行到下一事件）：运行到下一个事件发生为止。
- Step（单步）：单步仿真。
- Step-Over：仿真至子程序结束。
- Restart：重新开始仿真，重新加载设计模块，并初始化仿真时间为零。

（4）Break（停止）。

停止当前的仿真。

（5）End Simulation（结束仿真）。

结束当前仿真。

6.“Tools”菜单

（1）Waveform Compare（波形比较）。

在子菜单中有具体进行波形比较的命令。

（2）Coverage（覆盖率）。

测试仿真的代码覆盖率，所谓代码覆盖率是指仿真运行到当前已运行的代码占所有代码的比例，当然是越接近 100%越好。

（3）Breakpoints（断点设置）。

单击此选项出现断点设置对话框，设置断点。

（4）Execute Macro（执行宏文件）。

所谓的宏文件就是保存后的脚本，脚本保存起来，以后可以利用该命令来重新执行。

（5）Options（选项）。

- Transcript File：设置脚本文件的保存。
- Command History：命令历史。
- Save File：保存脚本文件。
- Saved Lines：限制脚本文件的行数。
- Line Prefix：设置每一行的初始前缀。
- Update Rate：设置状态条的刷新频率。
- ModelSim Prompt：改变 ModelSim 的命令提示符。
- VSIM Prompt：改变 VSIM 的命令提示符。
- Paused Prompt：改变 Paused 的命令提示符。
- HTML Viewer：设置打开在线帮助的文件。

（6）Edit Preferences（编辑参数选取）。

设置编辑参数。

（7）Save Preferences（保存参数选取）。

设置保存用的参数。

7.“Window”菜单

（1）Initial Layout（初始化版面）。

恢复所有窗口到初始时的大小和位置。

（2）Cascade（层叠）。

使所有打开的窗口层叠。

（3）Tile Horizontally（水平平铺）。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

水平分隔屏幕，显示所有打开的窗口。

（4）Tile Vertically（垂直平铺）。

垂直分隔屏幕，显示所有打开的窗口。

（5）Layout Style（版面格式）。

- Default（默认格式）：与 Initial Layout 格式相同。
- Classic（经典格式）：采样低于 5.5 版本的格式。
- Cascade：与 Cascade 格式相同。
- Horizontally：与 Tile Horizontally 格式相同。
- Vertically：与 Tile Vertically 格式相同。

（6）Icon Children。

除了主窗口之外的其他窗口缩为图标。

（7）Icon All。

将所有窗口缩为图标。

（8）Deacon All。

将所有缩为图标的窗口还原。

8.“Help”窗口

（1）About ModelSim。

显示 ModelSim 的版本、版权等信息。

（2）Release Notes。

显示 ModelSim 的版本发布信息。

（3）Welcome Menu。

显示欢迎画面。

（4）PDF Documentation。

在子菜单中可以选择 ModelSim 的 PDF 文档。

SE HTML Documentation：ModelSim 的超文本文档。

（5）Tcl Help：Tcl 帮助文档。

Tcl 是 Tools Command Language 的缩写，它是一种可扩充的命令解释语言，具有与 C 语言的接口和命令的能力，应用非常广泛，这方面也有专门的书籍。

（6）Tcl Man Pages：Tcl 主页面。

（7）Technotes：技术文档。

7.2.3 工具栏

ModelSim 的工具栏如图 7.4 所示。从左到右依次为：打开、复制、粘贴、编译选定、编译全部、仿真、停止仿真、重新开始仿真、仿真步长、运行一步、继续运行、运行所有、单步执行、主程序的单步执行。

图 7.4 工具栏

7.2.4 工作区

由图 7.3 可以看到，在用户使用界面里面工作区占有的面积最大，工作区初始可以分为左右两部分：左边是文件或者工程列表，右边是相应的文件的显示区。

7.2.5 状态栏

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

ModelSim 的状态条如图 7.5 所示，其中左面为当前工程的名称，右面为与当前仿真相关的一些系数，如仿真时间和仿真变量等。

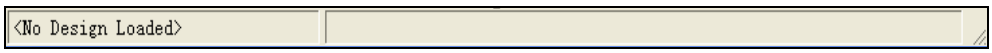


图 7.5　状态栏

7.3　ModelSim 的仿真流程

7.3.1　ModelSim 的安装

ModelSim 的最新版本可以从互连网上免费得到，需要购买的只是 License 文件。ModelSim 的下载地址为 http：//www.model.com/。打开网站页面后可以点击 Download，用户填写完一张表格以后可以得到一个小时的下载时间。

获得 License 的方法有很多种。

首先可以在线申请 License 文件，选取“开始→程序→ModelSim SE 5.8c→Submit License Request”，会打开一个网页，填写信息以后点击 Submit 就可以在线申请 License 了。

另外也可以点击“开始→程序→Modelsim SE 5.8c→License Request Instructions”会打开一个文本文档，按要求填写好后发到指定邮箱即可。

在得到 License 以后点击“开始→程序→Modelsim SE 5.8c→License Wizard”，在弹出的对话框里面指定 License 文件即可。

7.3.2　使用 ModelSim 进行功能仿真

使用 ModelSim 对设计的 HDL 程序进行仿真分为功能仿真和时序仿真两种。本节将以一个具体的实例讲解如何使用 ModelSim 对 HDL 工程进行功能仿真。时序仿真的具体内容将在 7.4 节具体介绍。

这里我们使用的例子是一个分频电路的设计。所谓分频电路是将较高频率的时钟分频，得到较低频率的时钟。分频电路的使用较为广泛。

例如，我们要编写一个显示时间的电路就需要一个分频器，将晶振的频率分频得到 1Hz 的时钟信号。分频有几种方法。对于较为规则的分频，如 2 分频、4 分频、8 分频等，可以调用 Quartus 或者 ISE 本身的库函数来实现。

对于较不规则的分频，我们也有两种方法：一种是利用计数器的某一位作为分频输出，一种是计数器计数到某一数值时，分频时钟信号翻转来实现分频。两种方法的可以从其仿真结果得到。

之所以选择这个例子，是因为这里例子有实用价值并且设计本身简单，仿真也较为简单。

（1）新建工程。

如图 7.6 所示，在 ModelSim 软件中选择“File”菜单的“New”选项，选择“Project”选项。打开如图 7.7 所示的新建工程对话框。

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

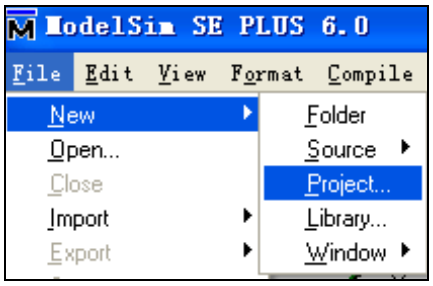


图 7.6　新建工程

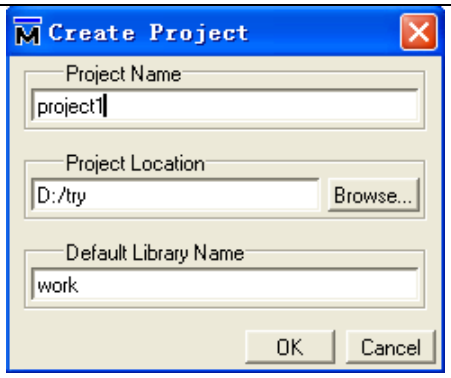


图 7.7　新建工程对话框

在该对话框中填写工程名称，路径和库。单击“OK”按钮，弹出如图 7.8 所示的添加工程项目对话框。选择向工程添加的项目类型，然后单击“Close”按钮完成工程的建立。在此例中选择“Add Exiting File”项目。

（2）加入文件。

在新建工程时，也可以不选择添加的项目种类、而是在 project 栏里面单击右键，在弹出的菜单里面选择“Add to Project → Existing File”，如图 7.9 所示。

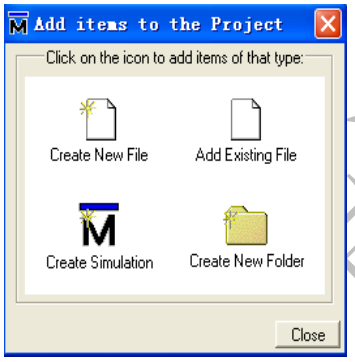


图 7.8　添加工程项目

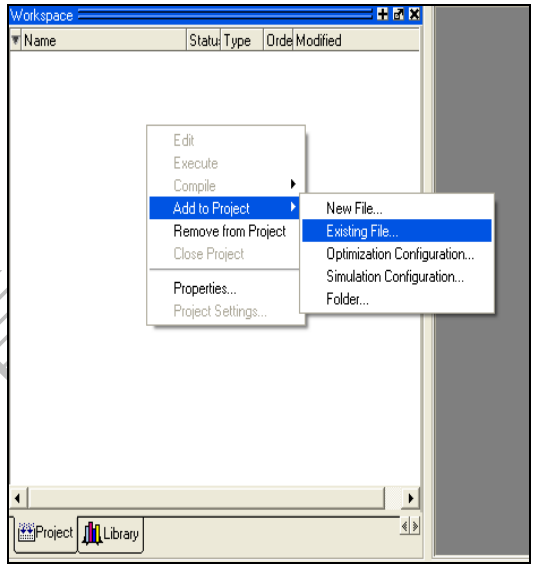
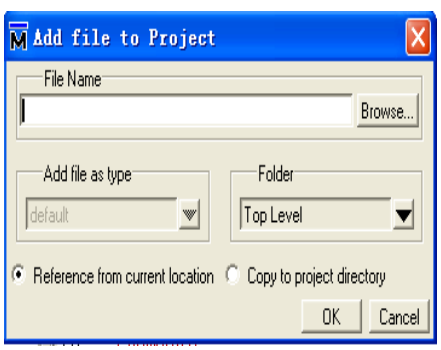


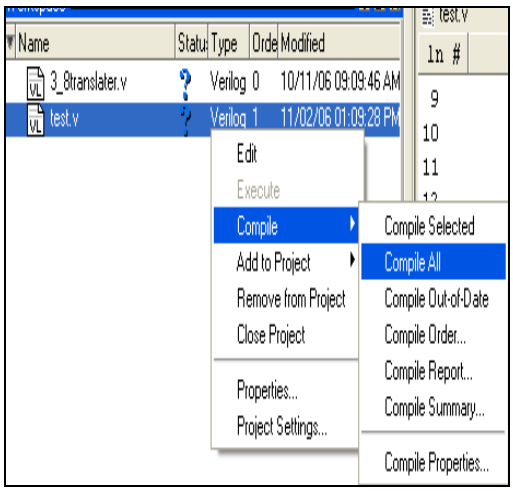
图 7.9　添加存在的工程项目

在弹出的添加文件对话框中，选择要添加的文件，如图 7.10 所示。

（3）编译文件。

在文件上面单击右键，选择“compile”选项下的“compile all”，如图 7.11 所示。





华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

图 7.10　添加文件对话框　　图 7.11　编译文件

当文件窗口中的“Status”栏的问号图标变成一个勾的图标时，说明编译成功。

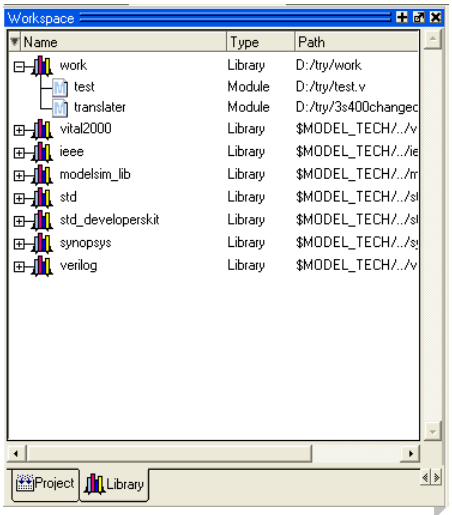
（4）仿真文件。

在 Workspace 的“Library”选项卡中，在 work 子目录里面双击 test 文件图标，就会自动完成仿真，如图 7.12 所示。

（5）查看波形。

编译成功以后，Workplace 的“sim”选项卡将变成如图 7.13 所示的样子。

右键单击 test，选择“Add”选项下面的“Add to Wave”选项，为波形窗口添加信号，如图 7.14 所示。



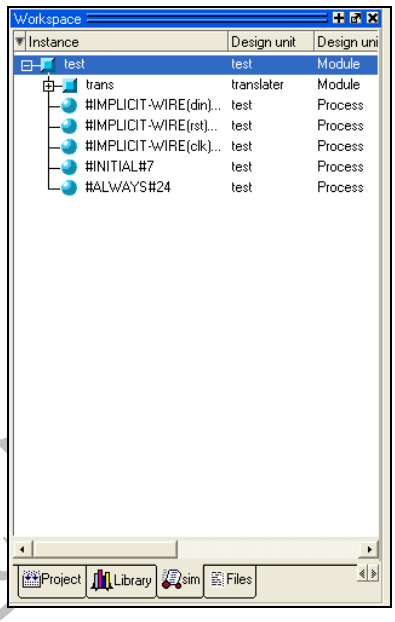


图 7.12　选择需要仿真的文件　　图 7.13　编译成功后的 test 实例

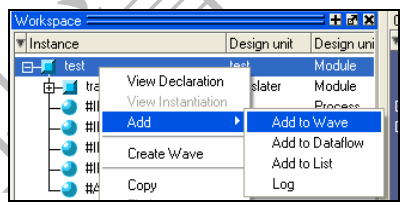


图 7.14　为波形窗口添加信号

此时即可在新弹出的波形窗口中看到已添加的信号，如图 7.15 所示。

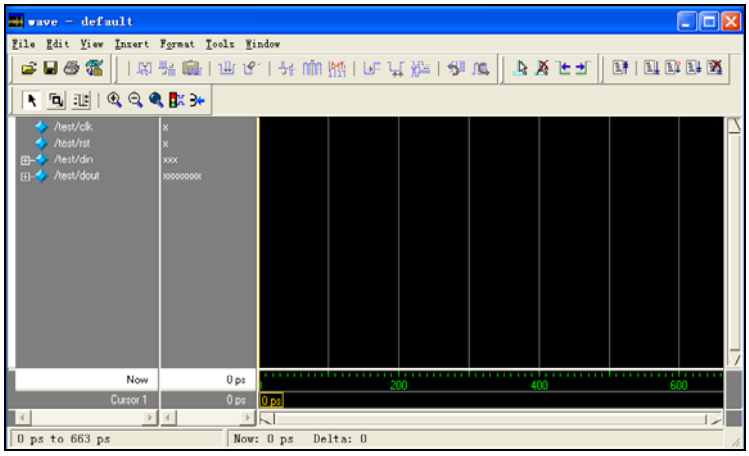


图 7.15　波形窗口

单击工具栏中的图标就能看见仿真结果了，如图 7.16 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

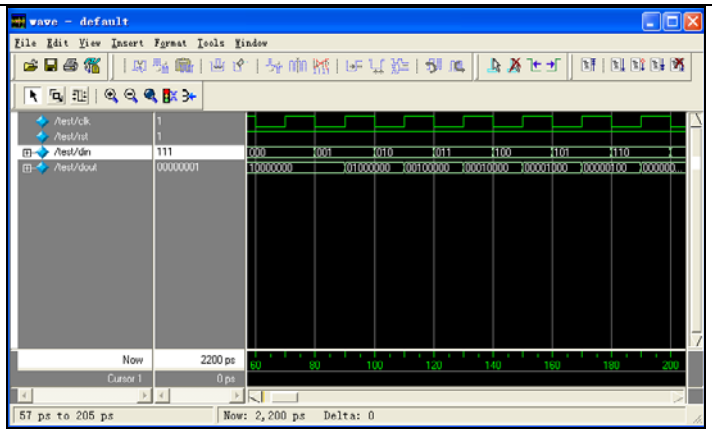


图 7.16　仿真结果

7.4　功能仿真和时序仿真的区别和实现方法

这里我们使用一个波形发生器作为例子，来说明如何使用 Modelsim 对 Quartus II 生成的 IP Core 和相应的 HDL 文件进行功能仿真和时序仿真。这个例子里面使用到了由 Quartus II 生成的一个片上 ROM 存储单元。这种存储单元和 RAM 一样，都是基本的 FPGA 片上存储单元，在以后的设计里面会经常使用到。

7.4.1　功能仿真

（1）在 Quartus II 中设置第三方仿真工具，选择“Assignments－Settings－EDA Tool Settings－Simulation”，选择“ModelSim－Verilog”。如图 7.17 所示。

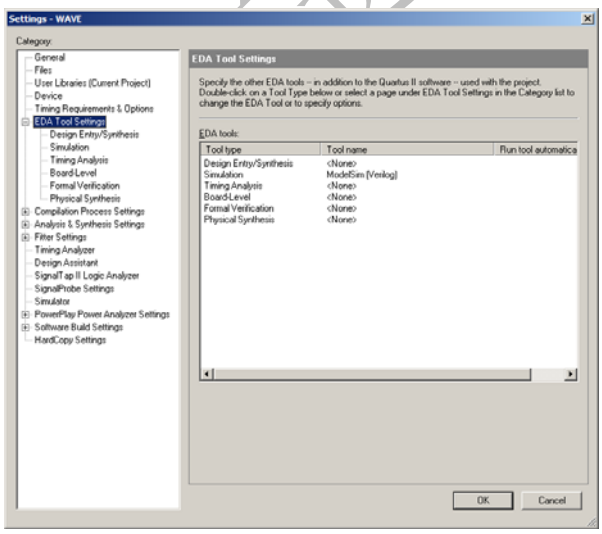
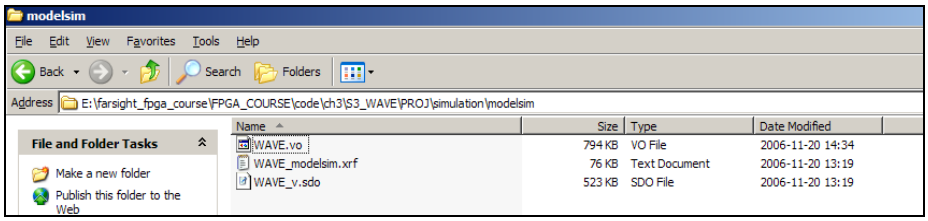


图 7.17　EDA 工具设置

（2）编译工程。编译带有 IP Core 的工程文件。

（3）编译完成后会在工程目录下生成“simulation－modelsim”的目录，如图 7.18 所示。其中包含了 3 个文件：“.vo”是仿真网表文件，可以用来代替设计文件；“.xrf”是 Quartus 编译生成的信息文件；“.sdo”是工程延时信息。



华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

图 7.18　编译生成文件

（4）加入仿真库文件。

仿真库的路径为“C：\altera\quartus50\eda\sim_lib”，包含了如下 3 个仿真库文件。

- 220model.v：带有用户原语类型的 Quartus 自带的 IP 核的库文件。
- altera_mf.v：Quartus 自带的 IP 核的库文件。
- cyclone_atoms.v：相应系列的器件库。

在本例中需要添加 altera_mf.v，cyclone_atoms.v 两个库文件。

（5）编写测试文件（详见 7.5 节）。

（6）将测试文件粘贴到刚才生成的目录中，如图 7.19 所示。

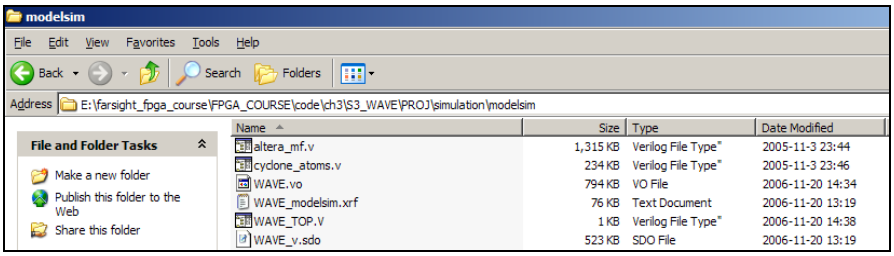


图 7.19　添加测试文件

（7）打开 modelsim。

（8）创建工程并添加源文件，如图 7.20 所示。

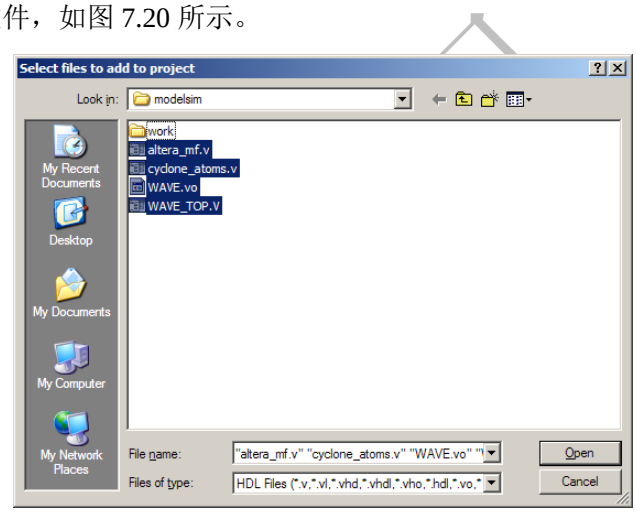


图 7.20　为工程添加源文件

（9）编译工程。结合库文件一起进行编译、如图 7.21 和 7.22 所示。

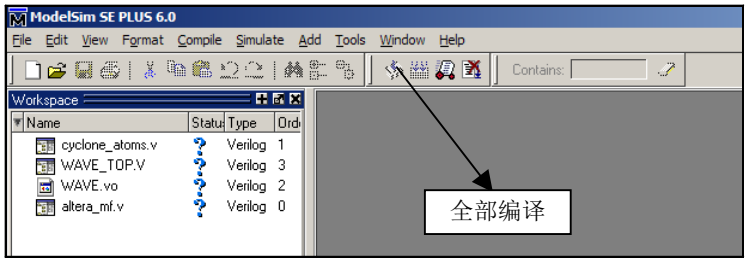
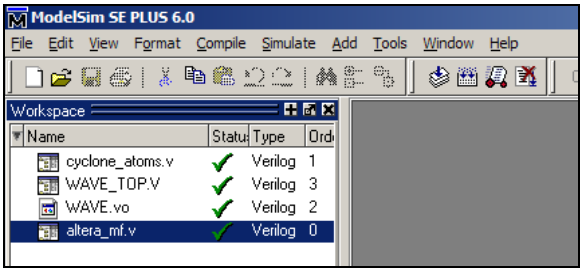


图 7.21　全部编译前



华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

图 7.22　全部编译后

（10）修改 wave.vo 文件。将文件中的延时信息注释掉，即：

```
    //initial $sdf_annotate("WAVE_v.sdo");
```

（11）重新编译 wave.vo 文件。

（12）打开开始仿真对话框，如图 7.23 所示对话框。

选择顶层模块 wave_top，开始进行仿真，如图 7.24 所示。

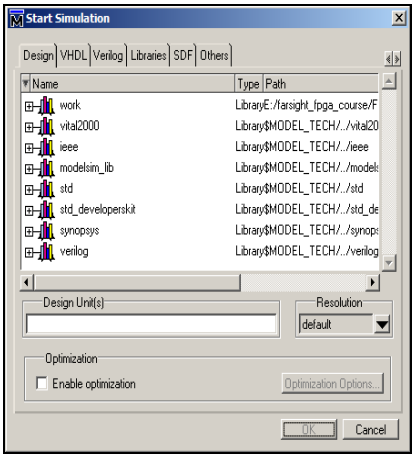


图 7.23　选择开始仿真的文件

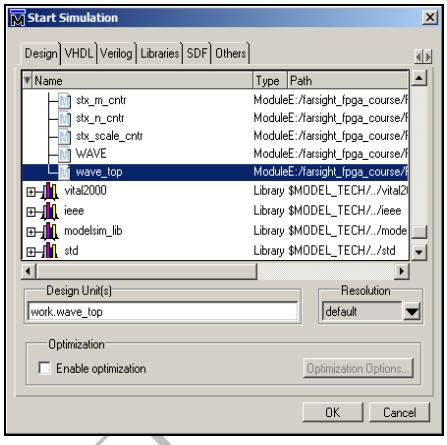


图 7.24　选择顶层模块文件

（13）添加信号。右键选择添加信号到波形图。并在命令行中敲入“run 1ms”，按“回车”键，开始仿真。结果波形如图 7.25 和图 7.26 所示，分别是二进制显示、十进制显示。

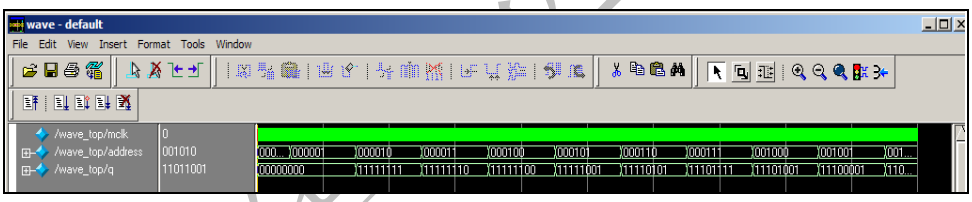


图 7.25　二进制显示

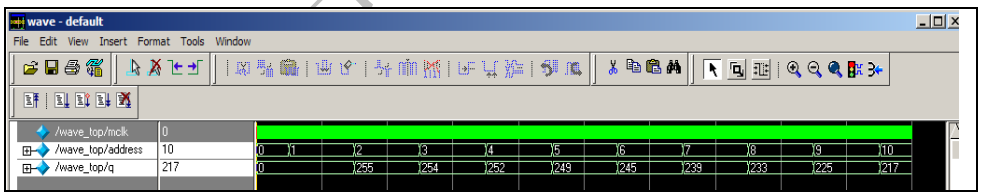


图 7.26　十进制显示

放大查看细节的波形时序，可以发现，功能仿真的结果是没有延迟的，如图 7.27 所示。

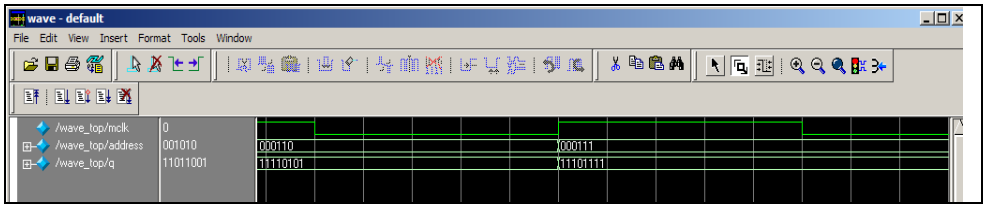


图 7.27　功能仿真结果无延迟

7.4.2　时序仿真

（1）将功能仿真第（10）步中对 wave.vo 文件做的注释改回来，重新进行编译。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

（2）重新打开开始仿真对话框，选择顶层模块，如图 7.28 所示。

（3）选择“SDF”选项卡，如图 7.29 所示。

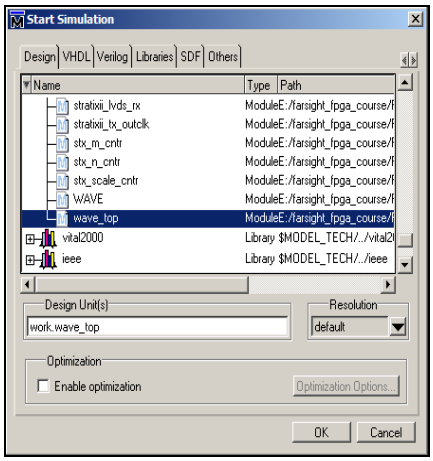


图 7.28　选择顶层模块仿真

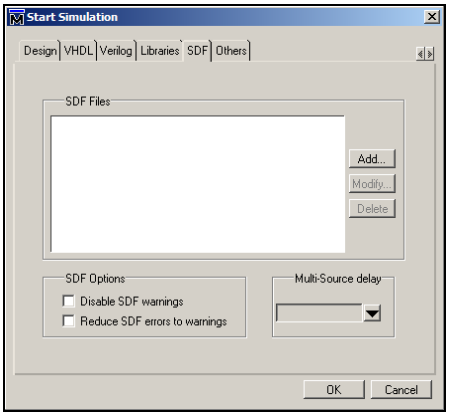


图 7.29　“SDF”选项卡

（4）加入“.sdo”文件，将两个 SDF Options 都选中，如图 7.30 所示。

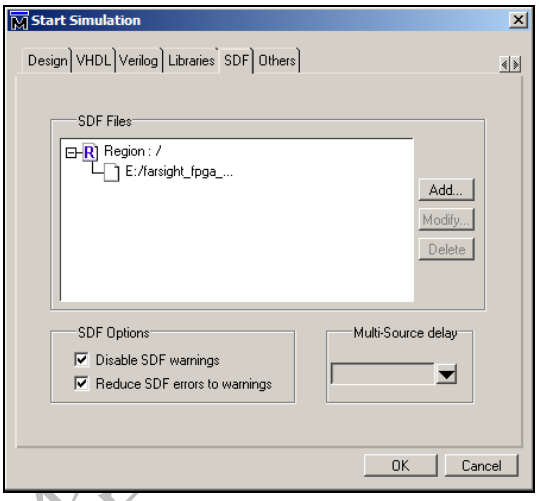


图 7.30　添加“.sdo”文件

（5）和功能仿真一样，为波形窗口添加信号，开始进行仿真。

（6）通过波形图，查看时序仿真的延时，如图 7.31 所示。

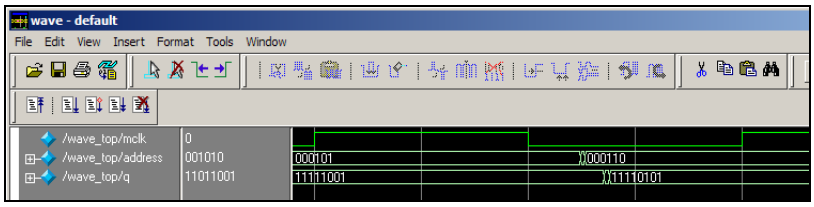


图 7.31　查看时序仿真延迟

7.5　仿真测试文件（Testbench）的设计方法

7.5.1　测试文件的用途

随着设计量和复杂度的不断增加，数字设计验证变得越来越难，所消耗的成本也越来越高。面对这种挑战，验证工程师必须依靠相应的验证工具和方法才行。对于大型的设计，比如上百万门的设计验证，工程师必须使用一整套规范的验证工具；而对于较小的设计，使用具有 HDL testbench 的仿真器是一个不错的选择。Testbench 已经成为了 HLL（High-Level-Language，高级语言）设计标准验证方法。一般来说，它能够完成

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

下面一些任务。

- 实例化 DUT（Design Under Test，被测设计）。
- 通过为模块添加测试向量对 DUT 进行仿真。
- 通过终端或波形窗口提供仿真结果。
- 比较实际输出与期望输出的差异。

一般来说，Testbench 使用工业标准 VHDL 或者 Verilog HDL 语言来描述。简单的 Testbench 通过调用用户设计的功能模块，然后进行仿真。较为复杂的 Testbench 还包括一些其他的功能，比如包含特定的激励向量或者进行实际输出与期望输出的比较等。

如图 7.32 所示是一个标准的 HDL 验证流程，图中表达了上述的 Testbench 功能。

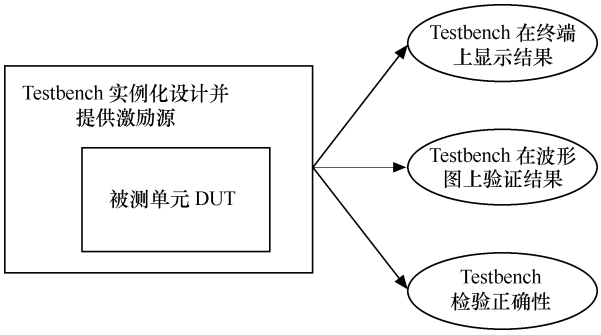


图 7.32　HDL Testbench 仿真流程

由于 Testbench 是使用 VHDL 或者 Verilog HDL 编写的，因此 Testbench 的验证流程是跨平台的。同时由于这两种语言是标准的非私有语言，所以使用它们进行验证的流程可以在未来的设计中继续使用。

下面讲解 Testbench 的设计方法。

7.5.2　测试文件设计方法

由于测试文件只是在仿真时使用，因此在综合中对 RTL 语言的语法限制在测试文件中并不存在。相反地，测试文件可以编写得更加通俗，所有的行为结构语法都可以使用，使之更加容易理解。所有的测试文件都包含了如表 7.1 所示的结构。

表 7.1　　测试文件的共同结构

VHDL	Verilog HDL
实体和结构声明（Entity，Architecture）	模块声明（Module）
信号声明（Signal）	信号生命（Signal）
顶层模块实例化	顶层模块实例化
激励向量	激励向量

正如所述，测试文件除了包含这些共有的结构意外，还完成一些额外的工作。例如显示结果、嵌入的误码检测等。下面通过一些例子来展现测试文件中常用的结构。

1．产生测试时钟

仿真必须产生时钟才能进行，只有在推进的仿真时钟中才可以判定输出的结果是否符合设计的要求。交互时钟是硬件设计语言里面最容易实现的时钟，下面通过 Verilog 语言的例子来说明如何生成测试时钟。

```
    Parameter ClockPeriod = 10;        // 声明时钟周期常量
    //时钟生成方法 1:
        initial begin
```

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

```
        forever Clock = #(ClockPeriod / 2) ~ Clock;
    end
//时钟生成方法 2:
    initial begin
        always #(ClockPeriod / 2) Clock = ~Clock;
    end
```

在两种实现方法中，都是用了 initial 块语句。在方法 1 中，使用了 forever 语句，是最常用的产生时钟的方法。在方法 2 中，使用过了 always 语句，同样也实现了时钟的生成。两种方法都产生了周期为 10 的时钟波形。

2．提供激励源

为了得到 Testbench 的验证结果，必须为 DUT 提供激励向量。并行激励模块常常被用来为测试文件提供必要的激励。有两种不同的方法来实现并行的激励：一种是绝对时间激励，另一种是相对时间激励。
在绝对时间激励中，所有的仿真时间值都是相对仿真时间零点定义的。而在相对时间激励中，一般会提供初始化值，然后等待事件来触发激励向量。两种方法都可以根据设计者的需要在同一个测试文件中使用。下面分别是两种方法产生激励的例子。
（1）绝对时间。

```
initial begin
    Reset = 1;              //仿真时间零点激励
    Load = 0;               //仿真时间零点激励
    Count_UpDn = 0;         //仿真时间零点激励
    #100 Reset = 0;         //绝对时间 100 激励
    #20 Load = 1;           //绝对时间 120 激励，相对上一个时间点 20
    #20 Count_UpDn = 1;     //绝对时间 140 激励，相对上一个时间点 20
end
```

（2）相对时间。

```
always @ (posedge clock)
    TB_Count <= TB_Count + 1;    //绝对时间的递增
initial begin
    if (TB_Count <= 5)    begin    //触发事件，产生下列激励
         Reset = 1;
         Load = 0;
         Count _UpDn = 0;
         end
    else begin              //触发事件，产生下列激励
         Reset = 0;
         Load = 1;
         Count_UpDn = 1;
    end
end
initial begin
    if (Count == 1100) begin    //触发事件，产生归零激励，并显示结果
         Count_UpDn <= 0;
         $display("Terminal Count Reached, now counting down.");
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

```
        end
end
```

值得注意的是，Verilog HDL 语言的 initial 模块之间是并行执行的，但是 initial 模块内部是顺序执行的。也就是说，测试文件的激励顺序在仿真时间零点同时启动并行模块，然后根据各个模块的内部激励顺序产生激励向量。

3．结果输出

测试文件通过关键词$display 和$monitor 来实现结果的输出。下面是使用 Verilog HDL 语言实现在终端上显示结果的例子：

```
// 在终端中打印信号的 ASCII 值
initial begin
     $timeformat(-9,1,"ns",12);                              //设置输出时钟格式
     $display(" Time Clk Rst Ld SftRg Data Sel");             //显示输入的字符串
     $monitor("%t %b %b %b %b %b %b",                        //设置输出信号的格式
     $realtime, clock, reset, load, shiftreg, data, sel); //指定输出的信号
end
```

$display 是将函数内部双引号中的字符串输出在终端中。而$monitor 则不同，因此它的输出是事件驱动的。在例子中，$monitor 信号列表中的$realtime 信号变化会触发终端显示事件的发生，该信号被设计者对应到仿真时间中，每次$monitor 的触发将会把信号列表中的信号值显示在终端中。
$monitor 语句中的“%”用于定义信号列表中信号的输出格式。例如，%t 将信号按照时间格式输出，%b 将信号按照二进制格式输出。另外，Verilog HDL 语言还提供了其他的输出格式，比如%h 为十六进制输出，%d 为十进制输出，%o 为八进制输出等。更为详细的格式输出定义可以参看 Verilog 参考手册。

7.5.3 测试常用语句

常用的 Verilog 测试用结构语句，比如$monitor、$display 和$time 在上面已经介绍过了。下面再来介绍一些其他的常用语句。

1．force/release

force 和 release 语句可以用来强制对执行过程中的寄存器或网络型信号量赋值。这两条语句共同完成一个强制赋值的过程。当一个被 force 的信号被 release 以后，这个信号将会保持当时的状态直到下一个赋值语句产生为止。
下面举个例子来说明这两条语句的使用。

```
module testbench;
    ...
    initial begin
        reset = 1;              //在仿真时间零点将 reset 激励为 1
        force DataOut = 101;    //在仿真时间零点强制使 DataOut 为 101，并保持
        #25 reset = 0;          //在仿真绝对时间 25 将 reset 激励为 0
        #25 release DataOut;    //在仿真绝对时间 50 释放
                                // DataOut 值将保持直至下一个对它的赋值语句
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

```
        ...
        end
endmodule
```

2．assign/deassign

assign/deassign 语句与 force/release 语句相类似，不过 assign/deassign 语句只能对设计中的寄存器型信号赋值。它们常常被用来设置输入值。
下面是这两个语句的例子。

```
module testbench;
    ...
    initial begin
        reset = 1;            //绝对时间零点对 reset 赋值 1
        DataOut = 101;
        #25 reset = 0;        //绝对时间 25 对 reset 赋值 0
        release DataOut;
        ...
    end
    initial begin
        #20 assign reset = 1; //此条语句覆盖之前的赋值语句（即绝对时间零点的赋值）
        #25 reset = 0;        //绝对时间 45 对 reset 赋值 0
        #50 release reset;    //绝对时间 95 释放 reset 信号
endmodule
```

3．timescales

timescale 语句用于定义测试文件的单位时间，同时也对仿真的精度有影响。它的语法定义如下：

```
‘timescale reference_time/precision
```

其中，reference_time 是单位时间的度量，precision 决定了仿真的推进延迟精度，同时也设置了仿真的推进步进单位。下面是 timescale 语句的使用范例：

```
‘timescale 1 ns / 1 ps          //度量参考为 1ns，精度为 1ps
module testbench;
    ...
    initial begin
        #5 reset = 1;          // 5 个仿真时间延迟，相当于 5×1ns = 5ns 的仿真时间
        #10 reset = 0;
        ...
        end
    initial begin
        //display 语句将在每一个仿真推进步进中执行，也就是 1ps 执行一次
        $display (“%d , Reset = %b”, $time, reset);
        end
endmodule
```

应该注意的是，如果仿真中使用了时间延迟值，那么仿真的精度应大于最小的延迟值。例如仿真中使用了 9ps 的仿真时间延迟，那么仿真的推进步进精度必须为 1ps 来保证 9ps 的延迟。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

4．Reading Memory Initialization Files

Verilog HDL 提供了$readmemb 和$readmemh 命令来读取 ASCII 文件，用于初始化 memory 的内容。这两个语句可以用于初始化 FPGA 中由 IP Core 生成的存储器宏模块，例如 RAM、ROM 等。
下面是利用这个语句对 Xilinx 的实例（design_instance）进行初始化的例子。

```
$readmemb ("<design.mif>", design_instance);
```

其中，mif 文件是 Xilinx 的 Core Generator 建立的对存储器进行初始化的文件。用户也可以自己编写这个文件的内容。

7.6 典型实例 13：SDRAM 读写控制的实现与 Modelsim 仿真

7.6.1 实例的内容及目标

1．实例的主要内容

本节旨在通过分析 SDRAM 控制器，介绍了 SDRAM 的基本工作模式。最后使用 Modelsim 对读写控制器进行仿真，帮助读者进一步了解一个真实的器件模块是如何进行 Modelsim 仿真的。

2．实例目标

通过本实例，读者应达到下面的目标。

- 了解 SDRAM 存储器的工作模式。
- 熟悉 Modelsim 仿真的基本流程。
- 可独立使用 Modelsim 仿真新工程。

7.6.2 SDRAM 简介

在高速实时或者非实时信号处理系统当中，常常使用大容量存储器实现数据缓存。而大容量存储器的控制与使用是整个系统实现过程中的重点和难点之一。
SDRAM（同步动态随即访问存储器）具有价格低廉、精密度高、读写速度快等优点，从而成为数据缓存器的首选存储介质。但是 SDRAM 的结构与 SRAM 有很大的差异，其控制时序和机制也比较复杂，这就限制了 SDRAM 的使用范围。
下面我们首先对 SDRAM 进行简单介绍。

1．SDRAM 信号

SDRAM 器件的信号可以分为控制、地址和数据信号 3 类，具体定义如表 7.2 所示。

表 7.2　　SDRAM 信号

信 号 名	信 号 类 型	信 号 描 述
CS	输入	Chip Enable，使能

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

CLK	输入	Clock，时钟
CKE	输入	Clock Enable，时钟使能
RAS	输入	Row Address Strobe，行地址选通

续表

信 号 名	信 号 类 型	信 号 描 述
CAS	输入	Column Address Strobe，列地址选通
WE	输入	Write Enable，写使能
DQML、DQMH	输入	Data Mask for Lower，Upper Bytes，高低字节屏蔽
BA	输入	Bank Address，Bank 地址
A[0:10]	输入	Address，地址
DQ[0:15]	双向	Data，数据

2．SDRAM 工作特性

通常一个 SDRAM 中包含几个 Bank，每个 Bank 的存储单元是按行和列寻址的。由于这种特殊的存储结构，SDRAM 有以下几个工作特性。

（1）SDRAM 的初始化。

SDRAM 在上电 100～200μs 后，必须由一个初始化进程来配置 SDRAM 的模式寄存器，模式寄存器的值决定着 SDRAM 的工作模式。

（2）访问存储单元。

为减少 I/O 引脚数量，SDRAM 复用了地址线。所以在读写 SDRAM 时，先由 ACTIVE 命令激活要读写的 Bank，并锁存行地址，然后在读写指令有效时锁存列地址。一旦 Bank 被激活后只有执行一次预充命令后才能再次激活同一 Bank。

（3）刷新和预充。

为了提高存储密度， SDRAM 采用硅片电容存储数据，电容总是倾向于放电，因此必须有定时的刷新周期以避免数据丢失。刷新周期可由（最小刷新周期÷时钟周期）计算获得。对 Bank 预充电或者关闭已激活的 Bank，可预充特定 Bank 也可同时作用于所有 Bank，A10、BA0 和 BA1 用于选择 Bank。

（4）操作控制。

SDRAM 的具体控制命令由一些专用控制引脚和地址线辅助完成。CS、RAS、CAS 和 WR 在时钟上升沿的状态决定具体操作动作，地址线和 Bank 选择控制线在部分操作动作中作为辅助参数输入。

由于特殊的存储结构，SDRAM 操作指令比较多，不像 SRAM 一样只有简单的读写，具体操作指令如表 7.3 所示。

表 7.3　　SDRAM 命令真值表

功　能	命 令 字	CS	RAS	CAS	WE	BA	A10	A[0:9]
取消器件选择	DSEL	H	X	X	X	X	X	X
无操作	NOP	L	H	H	H	X	X	X
读操作	READ	L	H	L	H	V	L	V
读等待/自动预充电	READAP	L	H	L	H	V	H	V

续表

功　能	命 令 字	CS	RAS	CAS	WE	BA	A10	A[0:9]
写操作	WRITE	L	H	L	L	V	L	V
写等待/自动预充电	WRITEAP	L	H	L	L	V	H	V

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

Bank 激活	ACT	L	L	H	H	V	V	V
对指定 Bank 预充电	PRE	L	L	H	L	V	L	X
对所有 Bank 预充电	PALL	L	L	H	L	X	H	X
自动刷新	CBR	L	L	L	H	X	X	X
加载模式寄存器	MRS	L	L	L	L	V	V	V

由表 7.3 可以看到，虽然 SDRAM 的容量大、速度快，但是存在存储操作困难的问题。一般的解决方案有两种，一是直接控制 SDRAM 的读写时序实现数据的存储和读取，二是编写一个 SDRAM 的读写控制器，将 SDRAM 的读写简化成 SRAM 形式，通过几个命令完成 SDRAM 的读写。

3．SDRAM 读写控制器

Xilinx、Altera、Lattice 等较大的 FPGA 制造厂商都编写了自己的 SDRAM 接口控制器。读者可以到官方网站去申请相关的控制器源代码。下面简单介绍其中一种，如图 7.33 所示是该 SDRAM 控制器总体设计框图和外部接口信号。

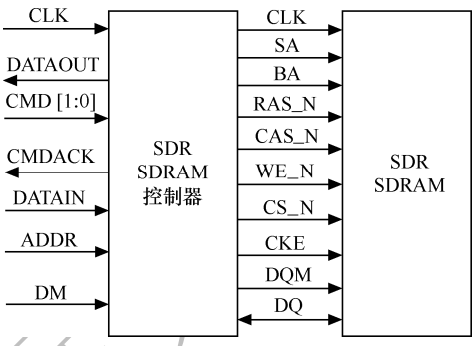


图 7.33　SDRAM 控制器框图

在图 7.33 中，控制器右端接口信号均为直接与 SDRAM 对应管脚相连的信号，在表 7.2 中已做介绍，不再重复。

控制器左端的接口信号为与 FPGA 相连的系统控制接口信号，定义如下。

- CLK：系统时钟信号。
- ADDR：系统给出的 SDRAM 地址信号。
- DATAIN：系统用于写入 SDRAM 的数据信号。
- DATAOUT：系统用于从 SDRAM 读出的数据信号。
- CMD[1:0]、CMDACK：系统和控制器的命令交互信号，参见表 7.3。
- DM：数据 Mask 信号。

一般来说，SDRAM 的读写控制时序可以分为初始化、写寄存器、自动刷新、突发模式读、突发模式写、整页读以及整页写等主要操作。具体的时序图可以查阅相关的器件数据手册，这里不再列出。

SDRAM 的读写控制也可以由如图 7.34 所示的读写状态机表示。

在 FPGA 中，实现如图 7.34 所示的状态机，再利用已有的 SDR SDRAM 控制器即可实现对 SDRAM 器件的控制。

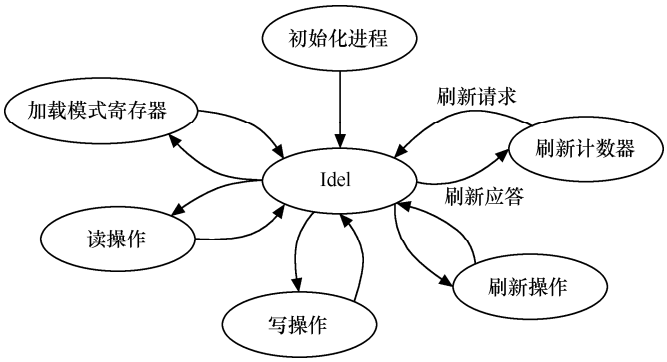


图 7.34　SDRAM 读写状态机

7.6.3　SDRAM 控制器的 Modelsim 仿真

（1）打开 ModelSim 软件。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

（2）创建工程。

如图 7.35 所示，在 Modelsim 中创建新工程，并设置工程的相关属性。

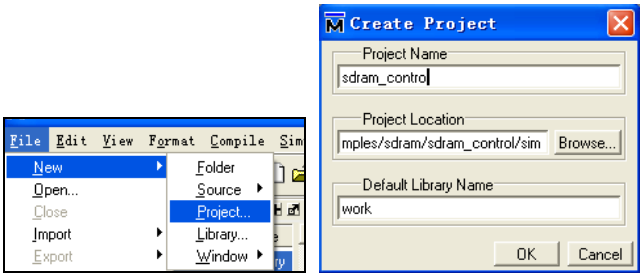


图 7.35　创建工程

（3）添加设计输入。

若要创建新的文件就选择【Create New File】图标，若要添加已经存在的文件就选择【Add Existing File】图标，如图 7.36 所示。本实例中使用已经存在的 SDRAM 控制器源文件作为设计输入，添加后，在 Workspace 浏览器中可以看到如图 7.37 的设计输入列表。

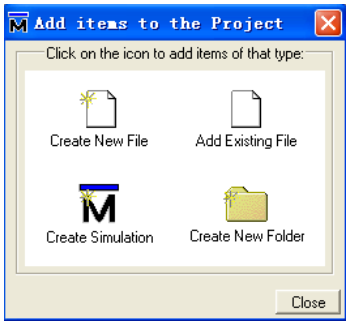


图 7.36　添加设计输入

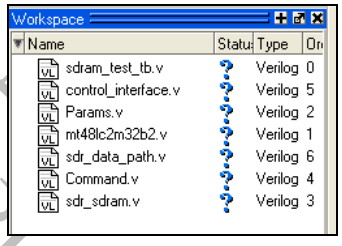


图 7.37　SDRAM 设计输入列表

（4）编译设计输入。

如图 7.38 所示，在任意一个源文件上单击右键，选择“Compile”/“Compile All”，对所有的源文件进行编译。编译后，若有错误，Modelsim 会在信息栏中显示出来。这时只要双击该错误，ModelSim 就会自动打开该错误所在的文件，并定位到出现错误所在的位置附近。若编译正确通过，源文件后面的蓝色问号就替换成为绿色的对号，如图 7.39 所示。

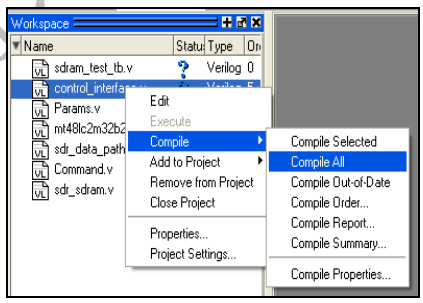


图 7.38　编译所有源文件

（5）仿真。

在 Workspace 浏览器中选择“Library”复选页，单击 Work 左边的小加号。在弹出的子菜单里面找到仿真模块“sdram_test_tb”。双击或右键选择“Simulate”选项，ModelSim 就会自动运行仿真，如图 7.40 所示。

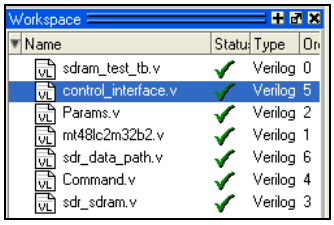


图 7.39　编译正确通过

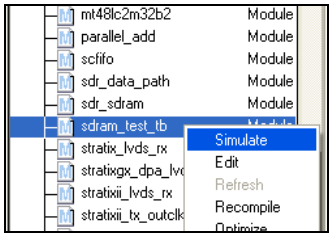


图 7.40　仿真

（6）观察波形。

在 Workspace 浏览器中选择“Sim”复选页，可以看到仿真模块的实例列表，如图 7.41 所示。

开发板免费领取试用　开发板免费领取试用

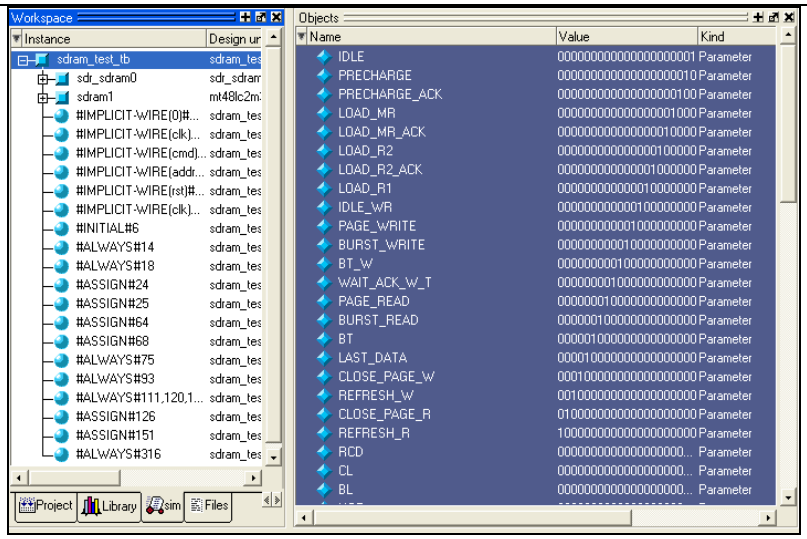


图 7.41　仿真模块实例列表

如图 7.42 所示，右键单击顶层测试模块，选择“Add”/“Add to Wave”选项，将该仿真模块的所有实例添加至波形观察器中。

添加后，ModelSim 将会自动打开一个波形观察器，并将顶层测试模块的所有寄存器和接口添加进去。回到 ModelSim 的界面，在命令输入窗口中键入“run 20us”，开始执行仿真，如图 7.43 所示。

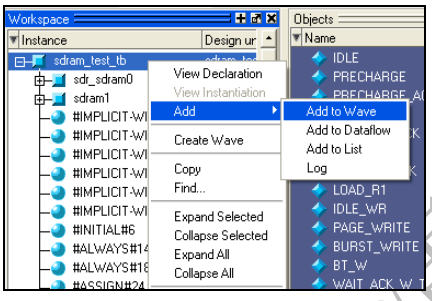


图 7.42　添加实例至波形观察器

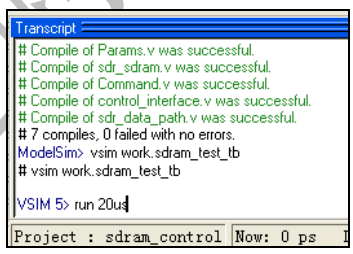


图 7.43　执行仿真

执行仿真后，经过相应的仿真时间，就可以在波形观察器中看见如图 7.44 所示的仿真结果。这个就是通过利用已有的 SDRAM 控制器及 SDRAM 器件模型，由用户编写对 SDRAM 控制器的状态机控制后得到的仿真结果。

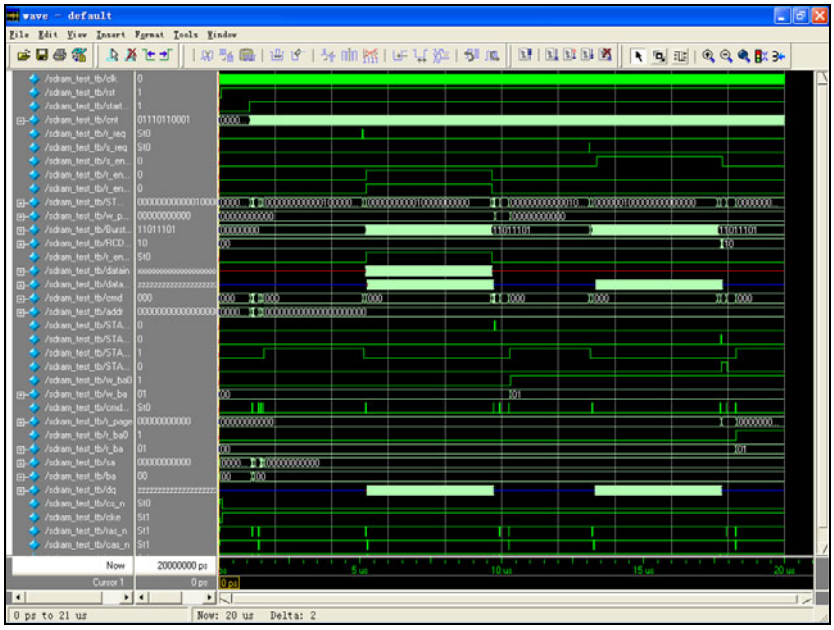


图 7.44　SDRAM 控制器仿真结果

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

7.6.4 小结

本节对数字系统中常用的存储器 SDRAM 做了初步的介绍，并在 Modelsim 中实现了对 SDRAM 控制器的仿真。通过这个实例，读者能够掌握分立器件是如何与 FPGA 逻辑实现 Modelsim 联合仿真的。
在大型的系统设计中，在系统硬件实现前对系统进行仿真是非常必要的。而这个仿真又不仅仅局限在 FPGA 等可编程逻辑器件内部，与之相关的分立器件也是系统仿真的重要组成部分。因此掌握分立器件的联合仿真是一个非常重要的技能。

联系方式

集团官网：www.hqyj.com　　嵌入式学院：www.embedu.org　　移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn　　物联网学院：www.topsight.cn　　研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路银海大厦 A 座 8 层，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218

开发板免费领取试用　开发板免费领取试用

华清远见 FARSIGHT 始于2004 | 嵌入式培训专家

10年口碑积累，成功培养50000多名研发工程师，铸就专业品牌形象

华清远见的企业理念是不仅要做良心教育、做专业教育，更要做受人尊敬的职业教育。

《FPGA 应用开发入门与典型实例》（修订版）

作者：华清远见

专业始于专注 卓识源于远见

第 8 章 基于 FPGA 的片上可编程系统(SOPC)设计

本章目标

- 掌握 SOPC 系统组成原理
- 了解 SOPC 系统的典型方案
- 掌握 NIOS II 解决方案
- 掌握 NIOS II 解决方案开发流程

专业始于专注 卓识源于远见

开发板免费领取试用 开发板免费领取试用

8.1 基于 FPGA 的 SOPC 系统组成原理和典型方案

8.1.1 SoC 及 SOPC 简介

1．SoC

SoC 即 System On Chip，是片上系统简称。它是 IC 设计与工艺技术水平不断提高的结果。SoC 从整个系统的角度出发，把处理机制、模型算法、芯片结构、各层次电路直至器件的设计紧密结合起来，在单个（或少数几个）芯片上完成整个系统的功能。所谓完整的系统一般包括中央处理器、存储器以及外围电路等。SoC 通常应用于小型的、日益复杂的客户电子设备。例如，声音检测设备的系统芯片是在。单个芯片上为所有用户提供包括音频接收端、模数转换器（ADC）、微处理器、必要的存储器以及输入/输出逻辑控制等设备。

SoC 的设计以 IP 核为基础，以分层次的硬件描述语言为系统功能和结构的主要描述手段，借助于以计算机为平台的 EDA 工具进行。

研究表明，与 IC 组成的系统相比，由于 SoC 设计能够综合并全盘考虑整个系统的各种情况，因而可以在同样的工艺技术条件下，实现更高性能的系统指标。SoC 技术也大大促进了软硬件协同设计以及计算机系统设计自动化的发展。

目前，在软硬件协同设计中最为活跃的研究工作包括系统描述、软硬件划分、软硬件协同综合以及软硬件协同模拟与验证等。如图 8.1 所示为一个较为普遍的面向 SoC 软硬件协同设计流程。

SoC 系统级设计方法基本采用层次化的设计思想和正交性的设计原则来完成系统级设计。目前，在学术界和工业界比较认可的 SoC 系统级设计方法学，从性质上基本可分为 3 大阵营：自顶向下，自底向上，上下结合和中间相遇。

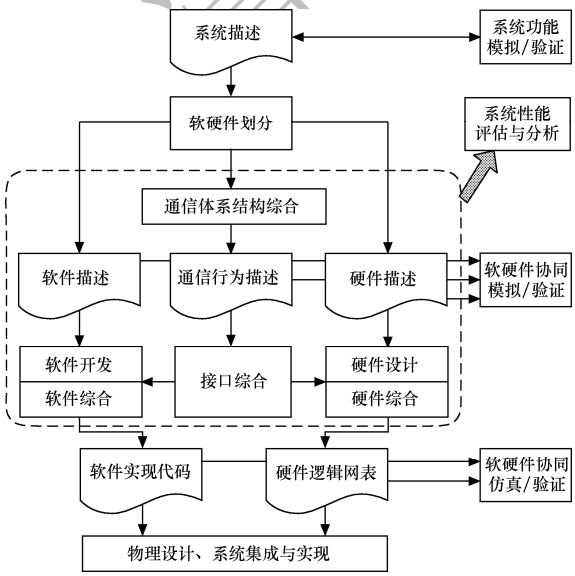


图 8.1 软硬件协同设计流程

2．SOPC

SOPC（System On a Programmable Chip：可编程芯片系统）是 Altera 公司提出来的一种灵活、高效的 SoC 解决方案。它将处理器、存储器、I/O 端口、LVDS、CDR 等系统设计需要的部件，集成到一个 PLD 器件上，构建成一个可编程的片上系统。

它是可编程系统，具有灵活的设计方式，可裁减、可扩充、可升级，并具备软硬件在系统可编程的功能。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

可编程器件内，还具有小容量高速 RAM 资源。由于市场上有丰富的 IP Core 资源可供灵活选择，用户可以构成各种不同的系统，如单处理器、多处理器系统。

有些可编程器件内还可以包含部分可编程模拟电路。除了系统使用的资源外，可编程器件内还具有足够的可编程逻辑资源，用于实现其他的附加逻辑。

SOPC 是 PLD 和 ASIC 技术融合的结果。集成了硬核或软核 CPU、DSP、存储器、外围 I/O 及可编程逻辑的 SOPC 芯片在应用的灵活性和价格上有极大的优势。所以，SOPC 代表了半导体产业未来发展的方向。

下面介绍一下 Xilinx 及 Altera 两家公司的 SOPC 的解决方案，其中将重点介绍 Altera 的解决定案。

8.1.2 Xilinx 的 SOPC 解决方案

Xilinx 公司不仅提供高端 32 位 MicroBlaze CPU 软嵌入式处理器芯核，也提供免费的低端 8 位 PicoBlaze CPU 软嵌入式处理器芯核。

一般的 C 交叉汇编程序和用户定制的 VHDL /Verilog HDL 源代码，还有大量的应用说明和参考设计，均可从 Xilinx 公司网站下载获得，它们均适用于 CoolRunner-II CPLD 以及各种 Spartan 和 Virtex FPGA 系列。

Xilinx 公司声称，该 CoolRunner-II 版本只占用 107 个宏单元，而适用于 Spartan-IIE 的 PicoBlaze 则需占用 76 个逻辑片。适用于 Virtex-II 系列 FPGA 器件的 PicoBlaze，其性能可达 55MIPS。

利用 MicroBlaze 软处理器解决方案，Xilinx 充分发挥出了嵌入式 FPGA 设计的全部潜能。MicroBlaze 核是一个 32 位哈佛 RISC 架构，具有丰富的针对嵌入式应用而优化的指令集。处理器是软核，意味着它是通过使用通用逻辑基元而非 FPGA 中的硬专用模块来实现的。

Xilinx 的 Spartan 和 Virtex 系列 FPGA 产品支持 MicroBlaze 软处理器。MicroBlaze 解决方案被设计得很灵活，能让用户控制很多特性，如缓存大小、接口和执行单元。为了以尽可能低的成本点达到面向目标应用的必要性能，可配置性允许用户对大小进行折衷来换取性能。

MicroBlaze 软核被授权成为 Xilinx 嵌入式开发套件（EDK）的一部分。EDK 是包括外围 IP 核库、曾荣获大奖的 Xilinx Platform Studio 工具套装（可用于生成直观的硬件系统）、基于 Eclipse 的软件开发环境、GNU 编译器、调试器以及更多工具在内的完整的嵌入式开发解决方案。MicroBlaze 处理器也为第三方开发工具和实时操作系统（RTOS）所支持。

对于更高的性能要求，Xilinx 在 Virtex-II 系列平台 FPGA 中植入了 32 位 PowerPC 内核来提供 DMIPS 性能。配合 Virtex-II Pro FPGA 在 450 MHz 时钟频率下的 32 位 PowerPC 内核可提供超过 680 DMIPS 的性能。由于处理器简化了设计，所用的芯片面积也非常小。例如，一个高效的设计仅使用了 Xilinx 2VP50 器件中不到 2%的片芯面积。

8.2 Altera 公司的 NIOS II 解决方案

8.2.1 NIOS 的主要特点

NIOS II 是一个用户可配置的通用 RISC 嵌入式处理器。Altera 推出的 NIOS II 系列嵌入式处理器扩展了目前世界上最流行的软核嵌入式处理器的性能。

把 NIOS II 嵌入到 Altera 的 FPGA，如 StratixII、Stratix、Cyclone II、Cyclone、APEX、ACEX 和 HardCopy 系列器件中，用户可以获得超过 200 DMIPS 的性能，可以从 3 种处理器以及超过 60 个的 IP 核中选择所需要的功能。

使用 NIOS II 处理器的用户可以根据他们的需要来调整嵌入式系统的特性、性能以及成本，快速使得产品推向市场，扩展产品的生命周期，并且避免处理器的更新换代。

下面是 NIOS II 处理器的优点和特性。

（1）提高系统性能。

- 一系列的处理器核可供选择，其中包括了超过 200 DMIPS 性能的核。
- 实现任何数量的处理器或将不同的处理器核组合在一起。

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

- 增加了已有的处理器，在 FPGA 中添加一个或更多的 NIOS II 软核处理器。

（2）更低的系统成本。

- 通过将处理器、外设、存储器和 I/O 接口集成到一个单一的 FPGA 中，从而降低了系统成本、复杂性和功耗。
- 通过将 NIOS II 处理器嵌入到低成本的 FPGA 中只需花费 35 美分或者更低（对大多数用户而言，NIOS II 所占逻辑资源的成本大约是 10～20 元人民币，具体取决于所选 FPGA 的类型和 NIOS II 的配置）。

（3）应对产品的生命周期。

- 提供易用的设计工具从而快速将产品推向市场。
- 提供永久、免费的许可，从而使基于 NIOS II 处理器的产品避免了处理器的更新换代而带来的损失。

（4）功能强大、易用的开发工具。

- 通过使用 NIOS II 集成开发环境（IDE），从而加速了软件的开发。
- 利用 Altera 的强大的 SOPC Builder 系统开发工具和 Quartus II 设计软件可以在几分钟内设计一个系统。

（5）使用完全功能的开发包。

- 使用易用的 NIOS II 开发包开始一个设计。
- 使用易用的 NIOS II 开发包开始设计。
- 可以选择具有低成本特性的 Cyclone FPGA 开发套件，或高性能的 StratixII FPGA 开发套件。

在很多对速度的要求不是很高的低端应用中，Altera 将一个软核放入 PLD，这个软核就是 NIOS。它只占芯片内部很少的一部分逻辑单元，成本很低。同 ASIC 相比较，如果将处理器放到 ASIC 中，生产的每片芯片都要付给处理器厂商专利费。况且 ASIC 的 NRE（一次性投资）大，风险也大，NIOS 则没有这个问题。NIOS 的开发工具包价格很低。

在速度要求高的高端应用，如通信领域，软核的处理速度不够。Altera 就将硬核（ARM9）集成到 APEX 器件中，还集成入 RAM 和 RAM 控制器。同时 Altera 本身在 PLD 的结构方面也不断发展和创新，推出的 HardCopy StratixIII 器件系列，是一个针对大容量设计的，从原型设计到批量生产的完整解决方案，试图成为 ASIC 的全面替代方案。

8.2.2 NIOS 技术实现方式

1．设计工具

Altera 的 Max+Plus II 曾是应用广泛、非常受欢迎的 PLD（可编程逻辑器件）设计软件。它对于 FPGA 和 CPLD 这些 PLD 器件的应用推广、电路部件和嵌入式系统的设计，以及高校有关专业的教学实习，做出了非常重要的贡献。

近年来，为适应微电子技术及其应用的飞速发展，尤其是 SoC（片上系统）技术发展的需要，Altera 推出了新版本的 PLD 设计软件 Quartus II。它继承了 Max+Plus II 的所有优点，是更加完善的 PLD 设计工具。PLD 的设计工具主要包括：不同的设计输入（包括原理图，VHDL 或 Verilog-HDL）工具、综合仿真工具、时限分析工具、功率评估工具、PLD 布局布线工具和产品验证工具等。

Altera 的 SOPC 开发工具 SOPC Builder，将软硬件的设计结合起来，提供给客户一个很好的开发环境，开创了嵌入式系统设计的新理念。

2．SOPC Builder

SOPC Builder 是一个软件工具，它属于一种基于 IP 或者平台的设计方法。利用 SOPC Builder，用户可以很

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

方便地将处理器、存储器和其他外设模块连接起来，形成一个完整的系统。

SOPC Builder 中已包含了 NIOS 处理器以及其他一些常用的外设 IP 模块。用户也可以设计自己的外设 IP。从用户的角度来看，SOPC Builder 是一个能够生成复杂硬件系统的工具。但从内部来看，SOPC Builder 包含两个主要部分：一个图形用户界面（GUI）以及一个系统生成程序。

（1）SOPC Builder 图形用户界面。

SOPC Builder 图形用户界面提供管理 IP 模块、配置系统和报告错误等功能。用户通过图形用户界面设计系统。完成设计之后，单击“Generate”按钮，则启动系统生成程序。系统生成程序通常从图形用户界面中启动。

（2）系统生成程序。

系统生成程序执行大量的功能，创建几乎所有的 SOPC Builder 输出文件（HDL 逻辑文件，C 程序的头文件和库文件，模拟文件等）。所以，SOPC Builder 可看作是一个以 IP 模块为输入，集成的系统为输出的工具。

SOPC Builder 会提示用户设置参数，并提示使用哪些可选的端口和外设。一旦向导生成了 NIOS 系统模块，则可以在设计文件中生成实例。一些常规硬软件接口、中断子程序等，都可被 SOPC Builder 完成，节约用户很多时间，而且更可靠。

3．NIOS II 指令结构

NIOS II CPU 是一种采用流水线技术、单指令流的 RISC 处理器，其大部分指令可以在一个时钟周期内完成。NIOS II 处理器又是一种软核 CPU，专门针对 Altera 的可编程逻辑器件以及片上可编程系统的设计思想，做了相应优化。

作为一种可配置的通用 RISC 处理器，它可以与用户自定义逻辑（user logic）结合构成 SoC 系统，并下载到 Altera 的可编程器件中去。32 位 NIOS 软核，结合外部闪存以及大容量存储器，可构成一个功能强大的 32 位嵌入式处理器系统

在 Altera 的 NIOS II 嵌入式处理器中，用户可以在 NIOS II 指令系统中增加用户自定义指令，以增强其对强实时软件算法的处理能力。用户自定义指令可以通过单周期或多周期操作来完成复杂的处理任务。

另外，增加的用户自定义指令同样可以访问存储器或 NIOS 系统外的逻辑。采用用户自定义指令，用户可以把一个复杂的标准指令序列，简化为一条用硬件实现的单个指令。这一特性可以用于多种情况，例如对数字信号处理（DSP）、数据包处理以及计算密集型软件进行优化。

4．Avalon 总线

Avalon 总线是一种将片上处理器和外设连接成片上可编程系统（SOPC）的一种简单总线结构。它描述了主从构件间的端口连接关系，以及构件间通信的时序关系。Avalon 总线规范提供了各种选项，来剪裁总线信号和时序，以满足不同类型外设的需要。

SOPC Builder 能够自动产生 Avalon 总线。Avalon 总线也包括许多特性和约定，用以支持 SOPC Builder 软件自动生成系统、总线和外设。片上可编程系统（SOPC）的设计人员在嵌入式系统开发和调试时，还有其他辅助工具，例如：ModelSim 模拟器，以及其他监控、调试工具等。

8.3 基于 NIOS II 的开发设计流程

NIOS II 使用 NIOS II IDE 集成开发环境来完成整个软件工程的编辑、编译、调试和下载。在采用 NIOS 处理器设计嵌入式系统时，通常会按照以下步骤。

（1）分析系统需求说明，包括功能需求和性能约束等。

（2）根据分析结果，选择片外外设或片内 IP（知识产权核）。除此以外，还要对选择的 IP 和外设进行初

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

步性能评估，以保证能够满足系统需求。

（3）设定 IP 和系统参数。

（4）确定系统互联逻辑，分配 FPGA 的引脚等。

（5）结合 NIOS 提供的软件开发包进行软件开发。

8.3.1 硬件开发流程

首先来看一下 NIOS 的硬件系统组成。

1．NIOS 硬件组成

如图 8.2 所示为典型 NIOS II 系统的硬件组成。

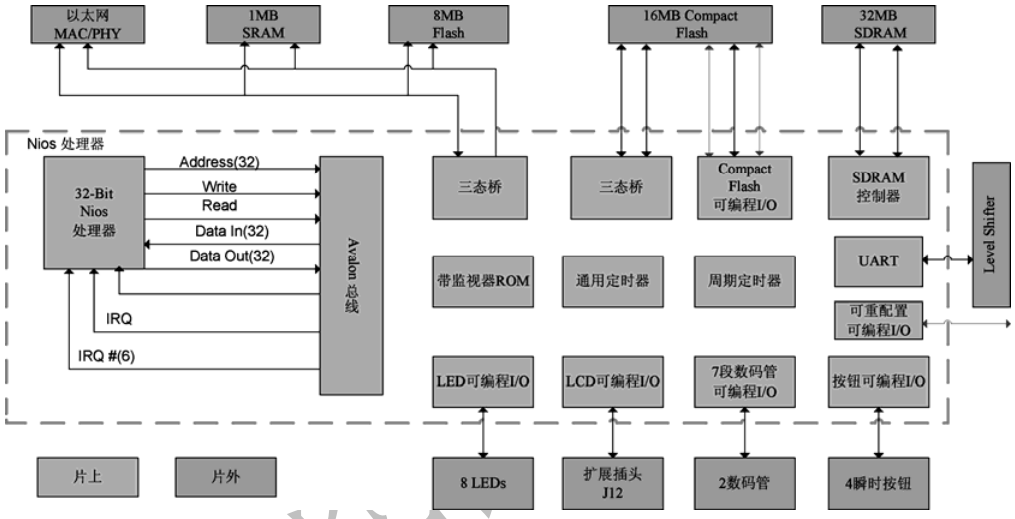


图 8.2　NIOS II 系统的硬件组成

如图 8.3 所示为典型 NIOS CPU 的系统框图。

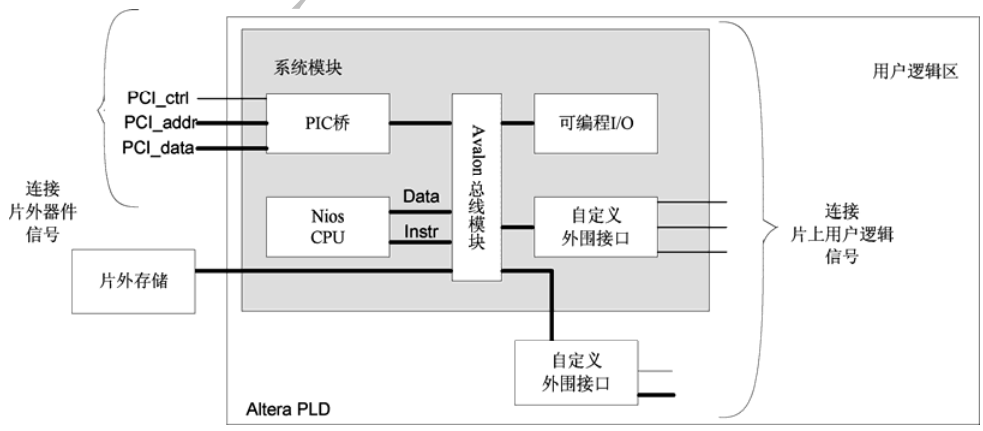


图 8.3　NIOS CPU 系统框图

我们可以将图 8.3 的系统结构划分如下。

$$\text{采用NIOS的系统}\begin{cases}\text{片外逻辑}\\ \text{片内逻辑}\begin{cases}\text{系统模块(system_module)}\\ \text{用户逻辑区域(user logic area)}\end{cases}\end{cases}$$

其中，片内逻辑是指实现在 FPGA 内部的电路设计，系统模块指的是由 SOPC Builder 自动生成的设计。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

SOPC Builder 会根据用户选择的 IP 生成相应的 HDL 描述文件（系统模块文件），这些文件与用户逻辑区域内的设计描述文件一起由 Quartus 软件综合，然后下载到 FPGA 内，这样就构成了系统的硬件基础。

系统模块包含至少一个 Avalon 主外设和整个 Avalon 总线模块。系统模块通常还包含一些 Avalon 从外设，例如 UART、PIO 和定时器等。SOPC Builder 可以帮助设计者从 IP 库寻找合适的 IP 并很快地集成一个系统，它采用图形用户界面（GUI）显示和组织 IP 模块，能够自动生成 IP 模块互连逻辑以及生成用于综合和模拟的文件，使得设计者可以轻松完成系统设计。

当用户使用 SOPC Builder 创建一个新的系统时，SOPC Builder 会为该系统自动生成一个 PTF 文件，所有的设计信息都存储在该 PTF 文件里。当使用 SOPC Builder 重新打开一个已有的系统时，SOPC Builder 会从并且只从 PTF 文件中读取系统具体设计信息。

用户逻辑区内可以包含用户自定义的 Avalon 外设，以及同系统模块无关的其他的用户自定义逻辑。硬件系统建立起来以后，用户可以利用 Altera 公司提供的 NIOS II IDE 工具结合 SOPC Builder 生成的与硬件系统对应的软件开发包来开发用户软件。

下面介绍使用这些设计工具进行 NIOS II 开发的流程。

2．NIOS II 硬件开发流程

下面是 NIOS II 硬件开发流程的主要步骤。

- 用 SOPC Builder 来选择合适的 CPU、存储器及外围器件，比如片内存储器、PIO、UART 和片外存储器接口。
- 使用 Quartus II 软件选取具体的器件，并对 SOPC Builder 生成 HDL 设计文件进行布局布线；再根据开发板分配 I/O 管脚；编译完后生成适合目标器件的网表。
- 使用下载电缆将配置文件下载到开发板上。当校验硬件或时钟完毕后，就可以开始软件开发工作了。

下面这个例子将要添加一个 NIOS II 系统得所有硬件部分，主要包括以下内容。

（1）选择 CPU。

选择 NIOS II/e 经济型处理器，其特点是占用逻辑资源少，编译速度快，不包括缓存，如图 8.4 所示。

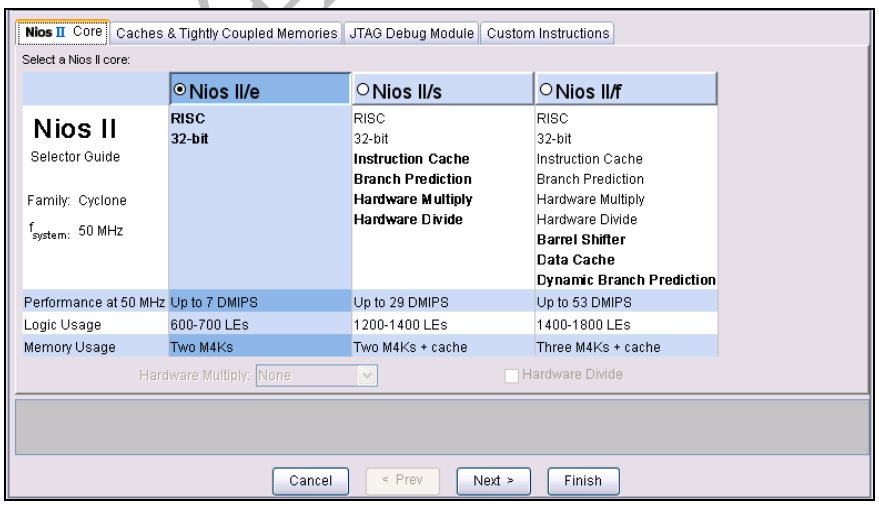


图 8.4　选择 NIOS II 处理器

（2）选择 JTAG 调试模块。

选择第一层 JTAG 硬件调试模块，如图 8.5 所示。

开发板免费领取试用　开发板免费领取试用

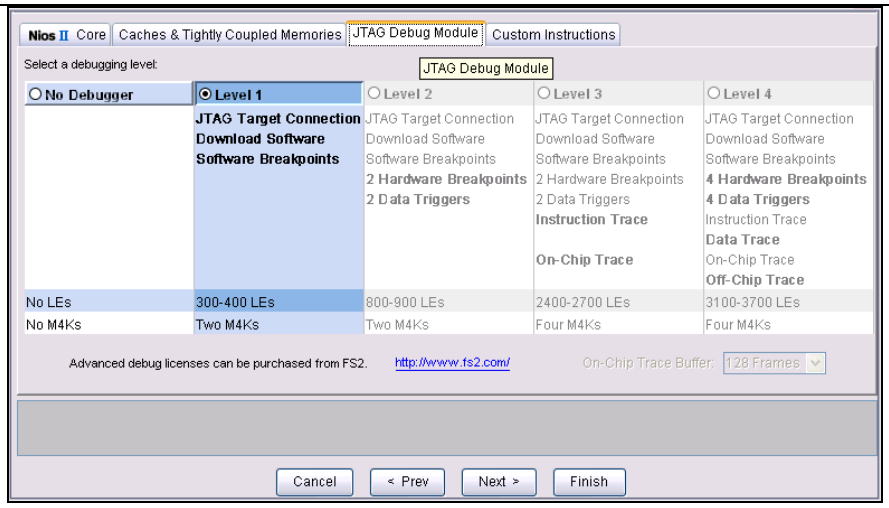


图 8.5　选择 JTAG 调试模块

（3）设置 JTAG UART 接口。

它是 NIOS II 系统嵌入式处理器新添加的接口元件，通过内嵌在 Altera FPGA 内部的联合测试行动组（JTAG）电路，在 PC 主机和 FPGA 之间进行串行字符流通信。特点是在运行时在调试区输入信息可以与 FPGA 进行交互。

如图 8.6 所示，使用默认选项即可。

（4）设置定时器。

如图 8.7 所示，定时器对于 HAL 系统库中的器件驱动非常有用，比如 JTAG UART 驱动使用定时器来实现 10s 的暂停。选择 Interval timer 在设置向导中周期选择 1msec，预设置（Preset Configurations）中选择 Full-featured。

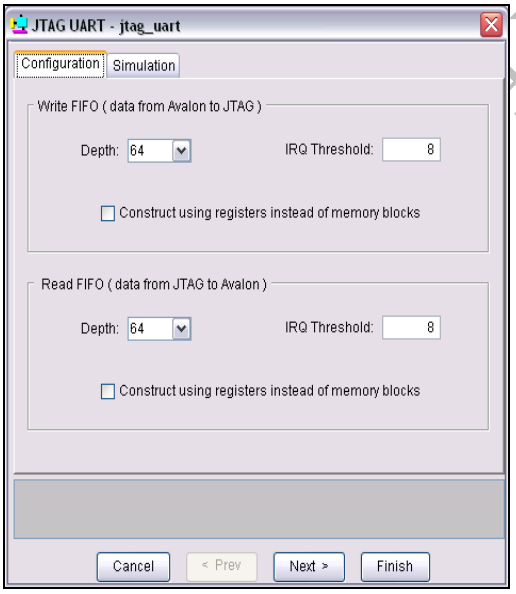


图 8.6　JTAG UART 接口

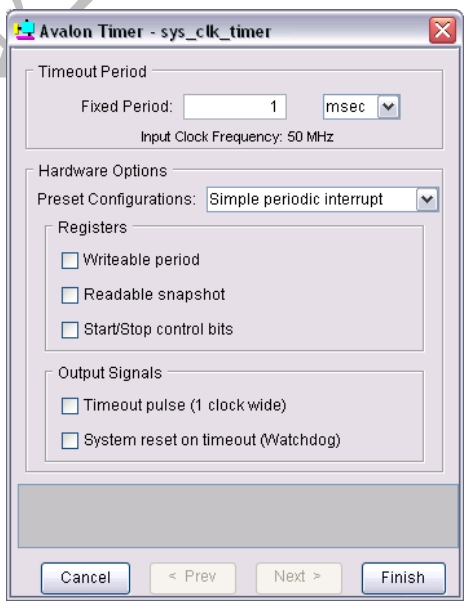


图 8.7　Avalon 总线定时器

（5）添加外部 Flash。

设置 Flash 地址宽度为 20bits，带宽为 16bits，如图 8.8 所示。

开发板免费领取试用　开发板免费领取试用

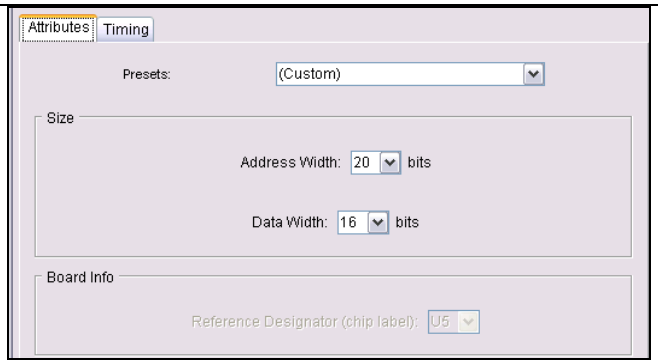


图 8.8　添加外部 Flash

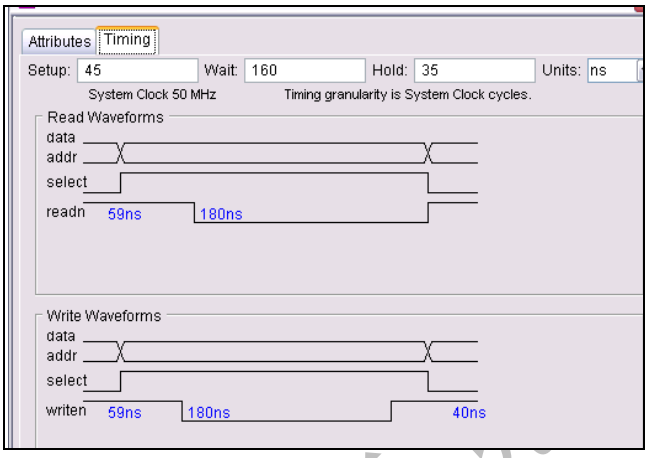


图 8.8　添加外部 Flash（续）

如果用户程序和数据比较大，超出了 EPCS1 的容量，则程序和数据可以保存在普通 Flash 中。Flash 的烧写可以采用 NIOS II IDE 的“Flash Programmer”来烧写，具体操作过程请参考/altera/kits/NIOS2/documents 目录下“ug_NIOS2_flash_programmer.pdf”。

（6）添加外部 SDRAM。

通常的系统都需要用户指定一个内存空间，这个内存是指 RAM，可以是片上的，也可以是片外的 SDRAM 或 SRAM 等。如果用户程序较大，超出了 EP1C20 所能定制的最大片上 RAM 容量，则也可以将程序放在 SDRAM 中运行。

将 SDRAM 进行如图 8.9 所示的设置。

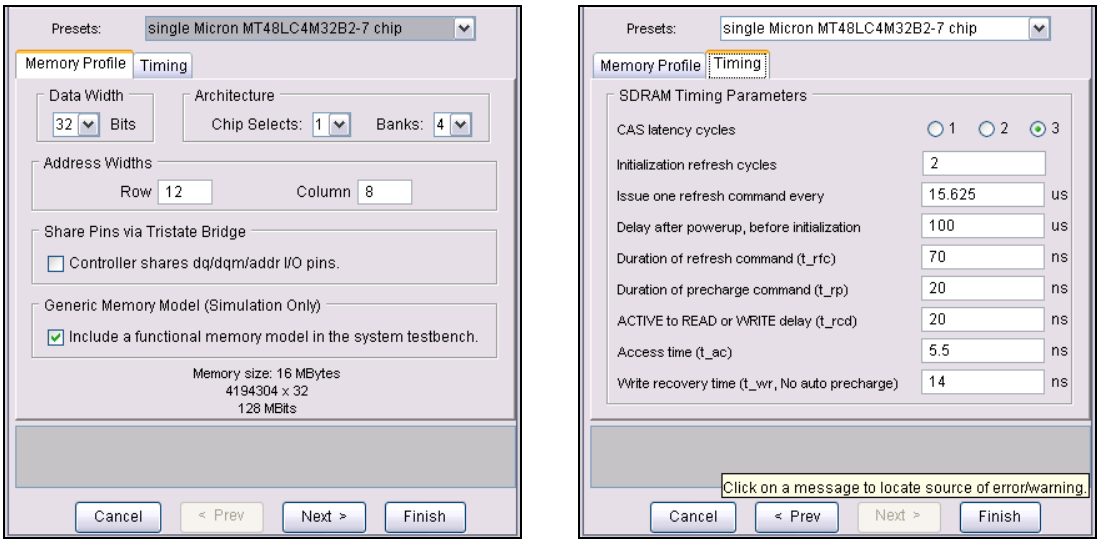


图 8.9　添加外部 SDRAM

（7）设置 Avalon 外部总线。

软件使用 Avalon 接口来连接片上元件和 Avalon 主从端口的。在 NIOS 开发板上，要实现 NIOS 系统与 FPGA 片外存储器通信，就必须在 Avalon 总线和连接外部存储器的总线之间添加一个桥，这个桥就是

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

Avalon 三态总线。

在如图 8.10 中选择为 Avalon 外部总线添加寄存器。

（8）添加串行通信设备 UART。

通用串行总线 RS-232 协议是我们最常用的通信协议，所以把它也添加到我们的系统里。注意选择所需的数据帧格式，如图 8.11 所示。

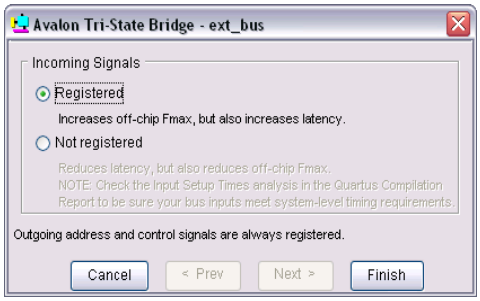


图 8.10　设置 Avalon 外部总线

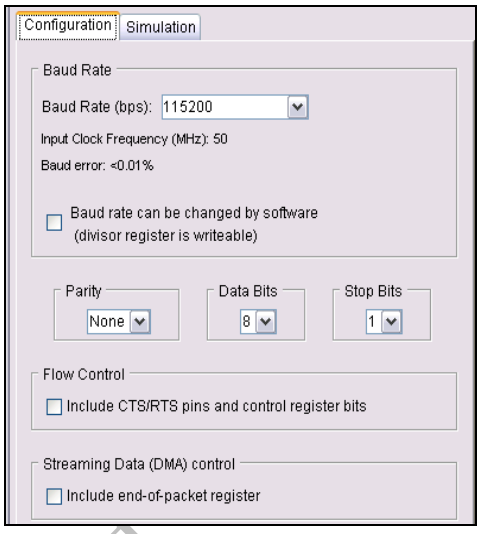


图 8.11　添加串行通信设备 UART

（9）添加 LCD 显示。

使用标准 16027 字符型液晶显示器。

（10）添加片上 RAM。

如图 8.12 所示，为 SOPC 系统添加片上 RAM 资源，并设置 RAM 参数。

（11）添加 system id。

（12）添加 8 个 LED 和输入设备按键开关。

以上几个步骤包含了一个标准 SOPC 一般所需要的设备，根据下面信息栏中的提示把 RAM 的数据线连接到总线上。接下来指定 Flash 地址为基地址，自动分配基地址及中断。最后完成的 SOPC 模块如图 8.13 所示。

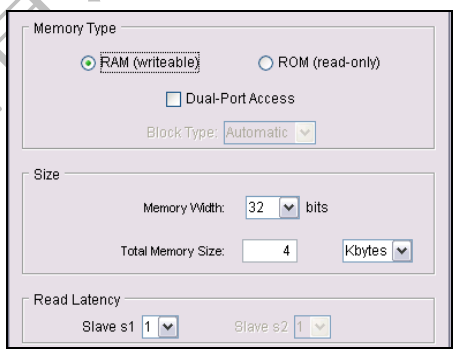


图 8.12　添加片上 RAM

接下来就可以在 Quartus II 中搭建完整的系统，加入锁相环、定义管脚等，并进行编译。这部分的内容详见本书其他章节的介绍。

需要注意的是由于 Flash 与 SRAM 共用一条数据总线，所以要把 SRAM 的片选关闭，如图 8.14 所示。

至此，整体 SOPC 硬件系统搭建完毕，其原理图如图 8.15 所示。

这样一个硬件系统规模已经比较完整了，可以作一些设计，比如 Flash 读写、液晶显示、串口传输等功能。把这些集合起来就可以用 C++软件编写一个简单的图像处理算法。

下面介绍一下 SOPC 系统的软件开发流程。

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

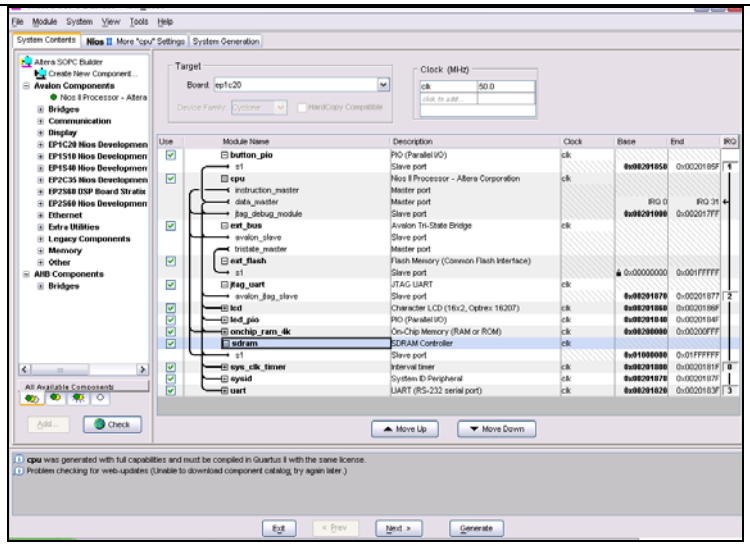


图 8.13　建立的 SOPC 模块

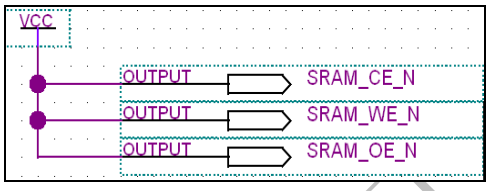


图 8.14　关闭 SRAM 片选

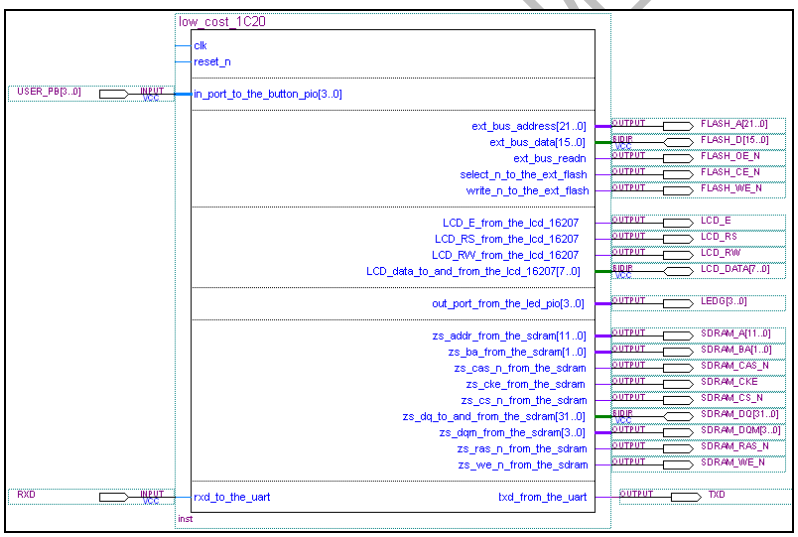


图 8.15　SOPC 系统原理图

8.3.2　软件开发流程

系统软件设计具体工作如下。

（1）在用 SOPC Builder 系统集成软件进行硬件设计同时，开始编写 C/C++软件，比如算法或控制程序。用户可以使用现成的软件库和开放的操作系统内核加快开发进程。

（2）在 NIOS II IDE 中建立新的软件工程时，IDE 会根据 SOPC Builder 对系统的硬件配置自动定制 HAL（硬件抽象层）系统库。这个系统库可以为程序和底层硬件的通信提供接口驱动程序。

（3）使用 NIOS II IDE 编译调试软件。

（4）在硬件逻辑已经下载到开发板的基础上将软件下载到开发板上并在硬件上运行。

下面主要介绍使用 SOPC Builder 来为一个 NIOS II 嵌入式处理器系统进行软件开发的过程，具体实现过程如下。

（1）进行硬件开发，如 8.3.1 小节所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

（2）启动 NIOS II IDE。

首先在 Alter SOPC Builder 中，单击 Run NIOS II IDE，启动 NIOS II IDE，如图 8.16 所示。

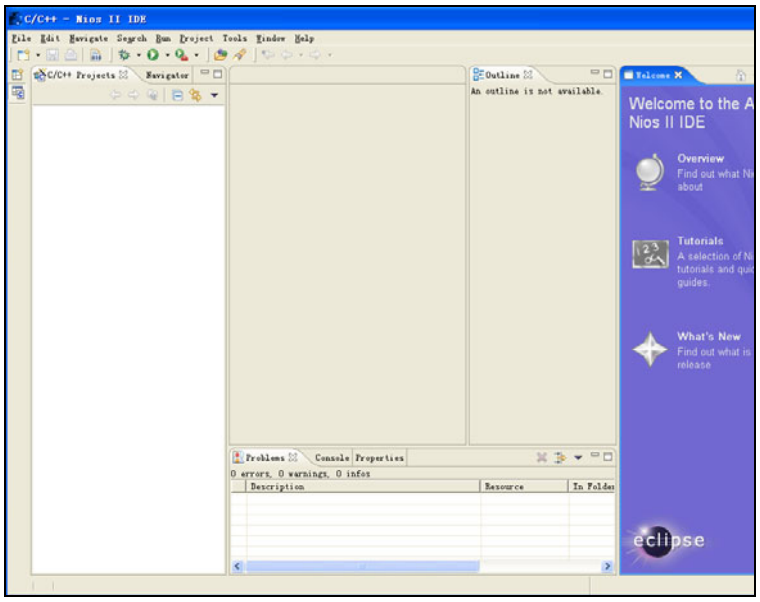


图 8.16　NIOS II IDE 界面

（3）新建软件工程。

选择“File”菜单的“New”菜单的“Project”选项，开启“New Project”对话框，选择“C/C++Application”，如图 8.17 所示。

（4）选择软件模板。

在新建软件工程对话框中单击“Next”按钮，为 SOPC 系统选择软件模板，如图 8.18 所示。

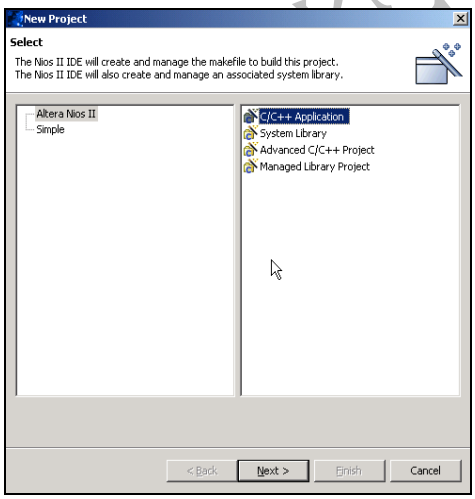


图 8.17　新建软件工程

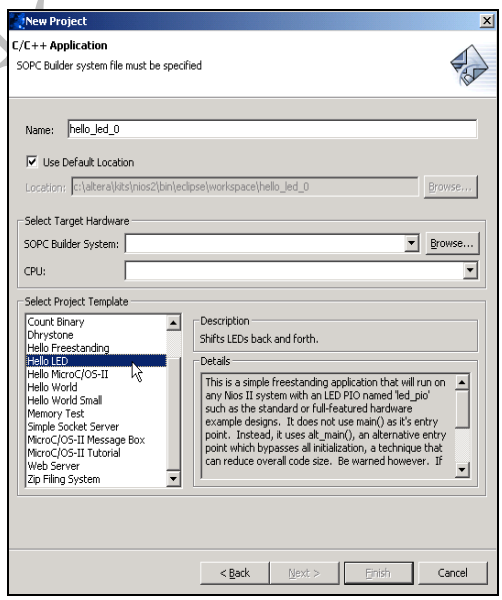


图 8.18　选择软件模板

（5）获取 SOPC 系统信息。

单击“SOPC Builder System”右边的“Browse”按钮打开目录对话框，从该工程目录下找到“NIOS II_c.ptf”，因为 NIOS II IDE 必须从这个文件获取该系统的相关信息。打开该文件后，界面显示如图 8.19 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

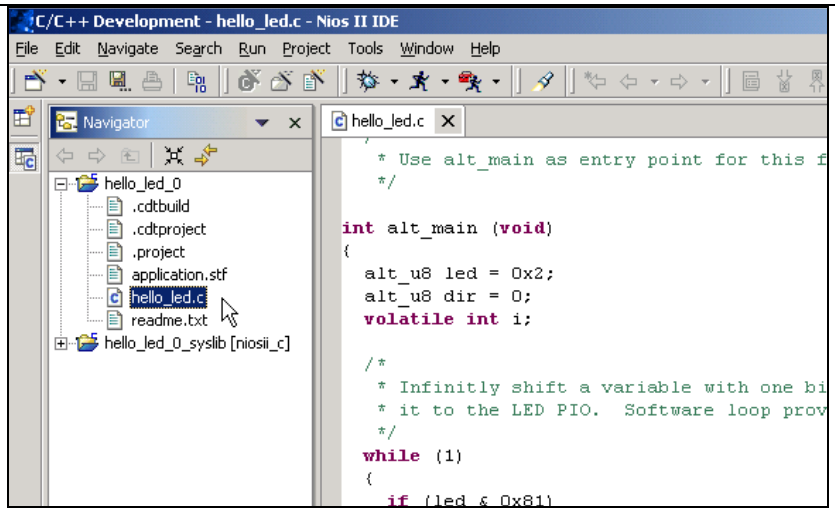


图 8.19　新建的软件工程

单击“Finish”按钮，即可建立一个名为 Hello_LED_0 的软件模板工程。

（6）编辑软件代码。

（7）设置工程优化级别。

编译之前我们先对项目进行一些设置，以使编译器编译出更高效、占用空间更小的代码。右键单击工程名称，在弹出的菜单中选择“Properties”选项，如图 8.20 所示。

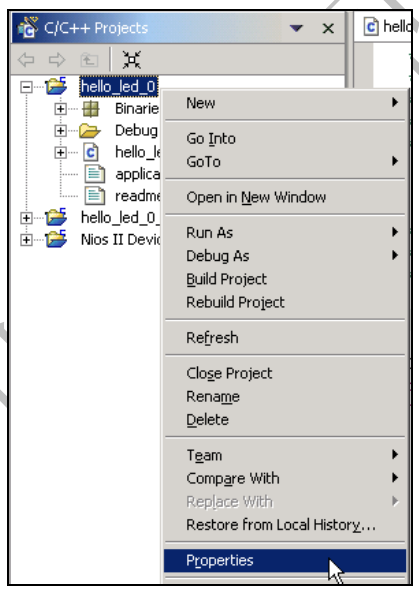


图 8.20　设置工程属性

单击后打开工程属性对话框，在“Configuration Settings”点击“General”页面，在“Optimization Level”（优化级别）中选择“Optimize size（-Os）”，如图 8.21 所示。

开发板免费领取试用　开发板免费领取试用

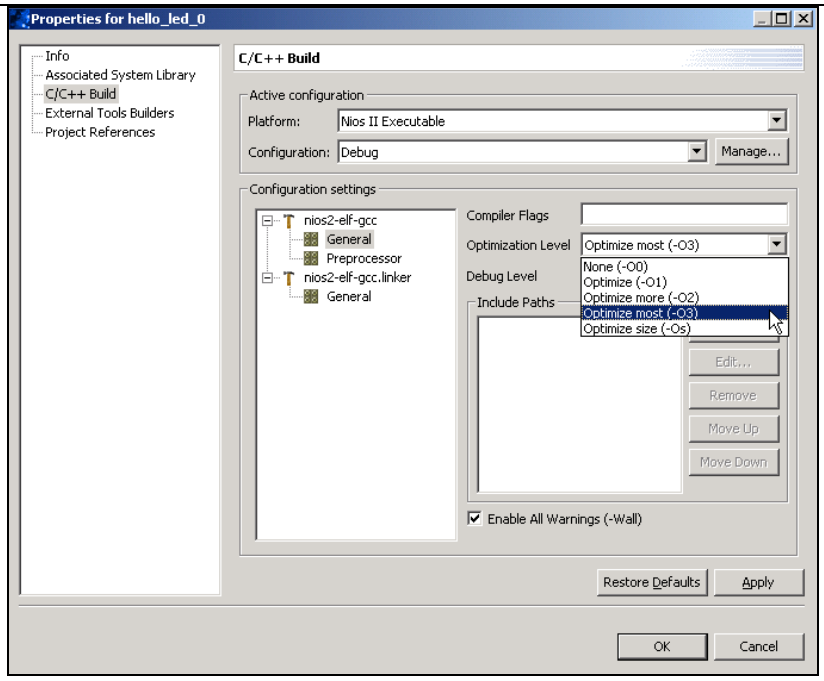


图 8.21　工程优化级别

（8）设置系统库属性。

右键单击*_syslib[NIOS II_c]工程名称，在弹出的菜单中选择“Properties”选项，如图 8.22 所示。

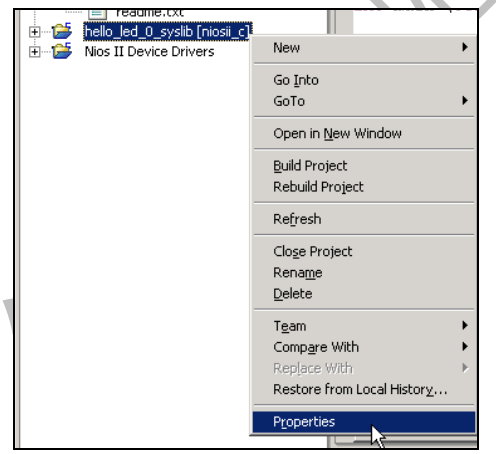


图 8.22　设置系统库属性

在系统库工程属性对话框中，同样在“Configuration Settings”点击“General”页面，在“Optimization Level”中选择“Optimize size（-Os）”，如图 8.23 所示。

开发板免费领取试用　开发板免费领取试用

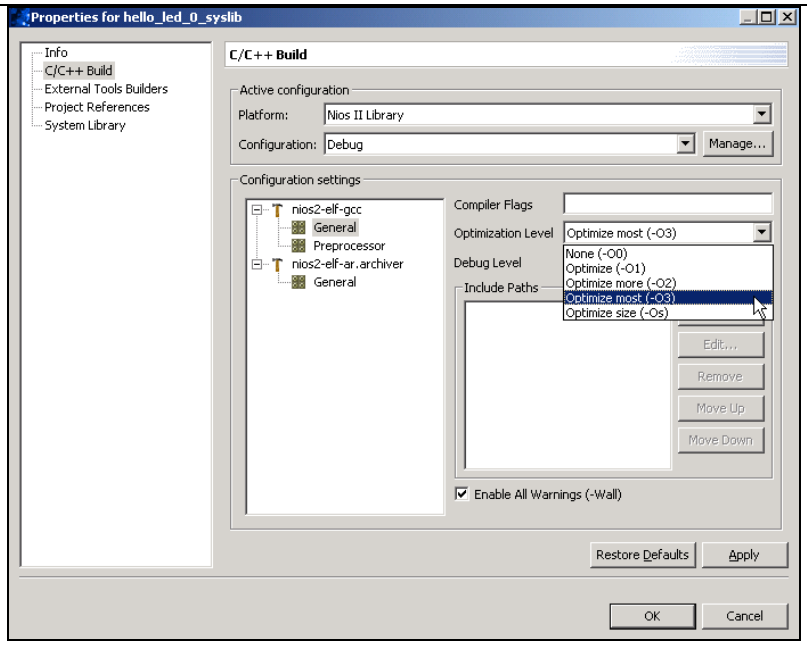


图 8.23　系统库工程优化级别

然后，单击该对话框左侧的“System Library”打开“System Library”属性页面。将“Max file descriptors:”栏改为 4，清除“Clean exit（flush buffers）”和“Link with profiling library”，选上“Reduced device drivers”和“Small C library”，如图 8.24 所示。

单击“OK”按钮，完成设置。以上设置主要目的是为了优化程序，并减少程序占用内存空间。

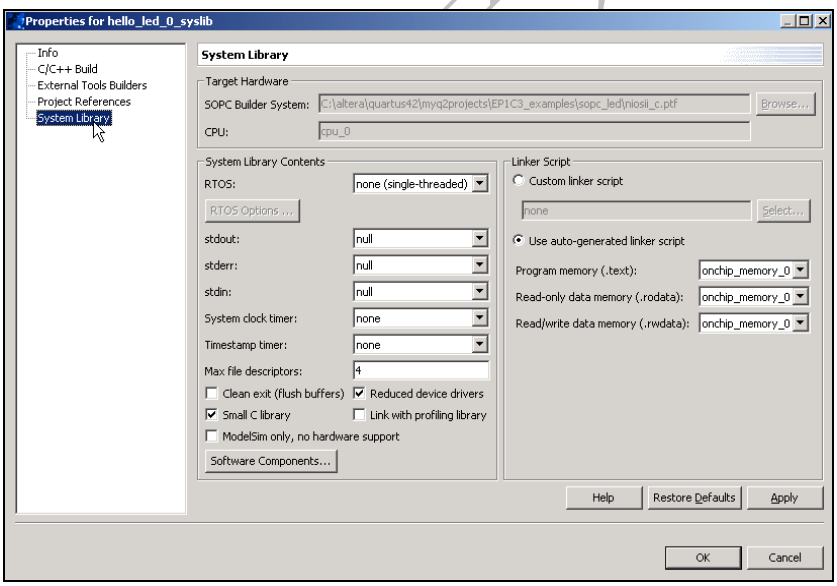


图 8.24　设置系统库其他属性

（9）编译。

右键单击工程，在弹出的菜单中选择“Build Project”选项，开始进行软件工程编译（为使编译过程更加顺利，此时最好关闭杀毒软件和其他占用电脑资源较大的软件）。

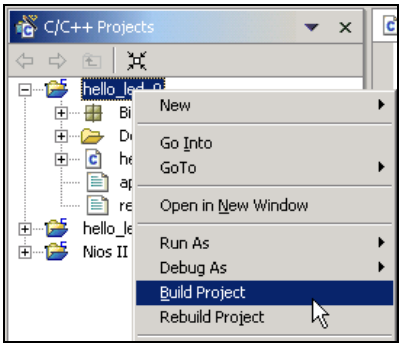


图 8.25　编译软件工程

如果工程设计无误，可以看到以下信息：“Build completed”。如果看到了该条提示，表示软件编译成功，可以看到程序占用空间等信息。

（10）调试。

将 ByteBlasterII 下载线接到 FPGA 实验平台的 JTAG 口，接上 5V 电源。选择“Run”菜单下的“Debug As”选项，选择“NIOS II Hardware”，如图 8.26 所示。

开发板免费领取试用　开发板免费领取试用

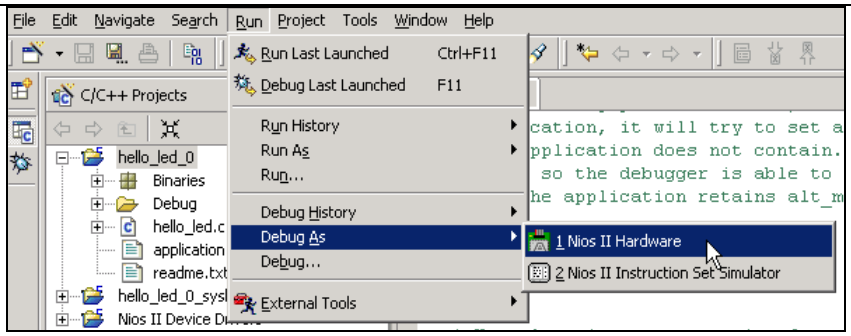


图 8.26　打开调试界面

选择后，NIOS II IDE 会打开调试界面（Debug Perspective）。可以在程序中设置断点，运行停止在断点处（图 8.27 的箭头处），如图 8.27 所示。

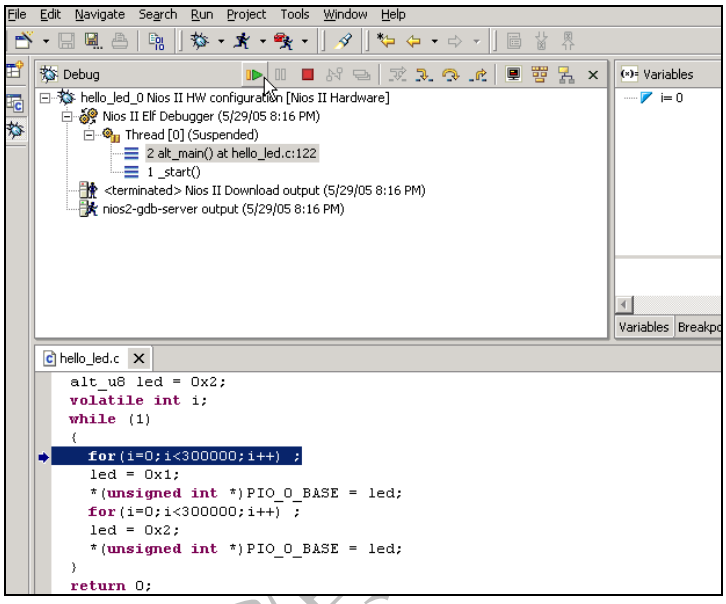


图 8.27　调试至断点

单击继续执行（Resume）按钮，程序就会继续运行。单击暂停按钮，程序暂停运行。单击停止按钮，将退出运行，如图 8.28 所示。

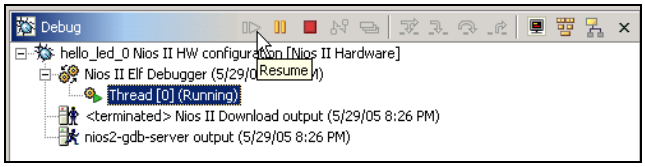


图 8.28　调试按钮

如需对程序进行修改，单击停止按钮退出运行，然后单击如图 8.29 所示的按钮回到 C/C++编辑界面。选择后，即可返回程序编辑界面修改代码，然后后再重复上述调试过程。

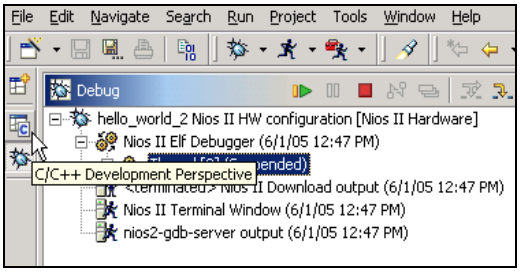


图 8.29　重新修改程序

（11）编译并下载。

编写完软件工程后，我们还要回到 Quartus II 里面对整个系统进行编译。选择“Processing”菜单下面的“Start Compilation”。编译成功以后我们就可以选择“Tools”菜单里面的“Programmer”，下载整个配置到 FPGA 中去了。至此整个 SOPC 系统的软硬件开发流程就已经介绍完毕。读者可以动手设计自己的 SOPC 系统了。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

8.4 典型实例 14：基于 NIOSII 处理器的“Hello LED”程序设计

8.4.1 实例的内容及目标

1．实例的主要内容

本节旨在通过给定的工程实例——“Hello LED”来熟悉 Altera 软嵌入式系统的软硬件设计方法。同时使用基于 Altera FPGA 的开发板将该实例进行下载验证，完成工程设计的硬件实现。本节主要讲解下面一些知识点。

- Quartus II SOPC Builder 使用方法。
- NIOSII IDE 软件开发流程。

通过这些知识点，按照本节实例的实现过程，读者可以迅速掌握 Altera 软嵌入式软硬件设计方法。

2．实例目标

通过详细的流程讲解，读者应达到下面的目标。

- 熟悉 Quartus II SOPC builder 使用方法。
- 熟悉 NIOSII IDE 软件设计方法。

8.4.2 实例详解

本实例使用图解的方式将整个流程一步一步展现给读者，使读者能够轻松掌握开发的流程。

本实例重点在于熟悉使用 Quartus II 和 SOPC Builder 来创建一个 Nios II 嵌入式处理器系统，并通过软件编程实现对实验板上面的 LED 灯的控制，具体实现过程如下。

（1）新建工程。

新建一个工程目录“sopc_led”，在此目录下建立一个名为“sopc_led”的 Quartus II 工程，并新建一个顶层图，保存于工程中。

（2）SOPC Builder 定制 NIOSII 处理器。

用 SOPC Builder 定制 NiosII 处理器及其外设。打开“Tools”/“SOPC Builder”，如图 8.30 所示。

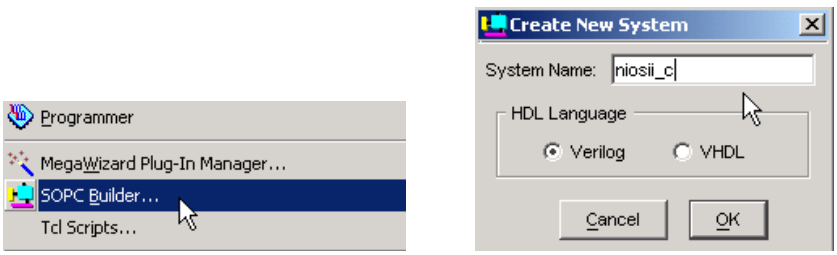


图 8.30 打开 Tools--SOPC Builder

指定系统名字，本例中我们输入 niosii_c。单击“OK”按钮进入 SOPC 定制界面。

华清远见
华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

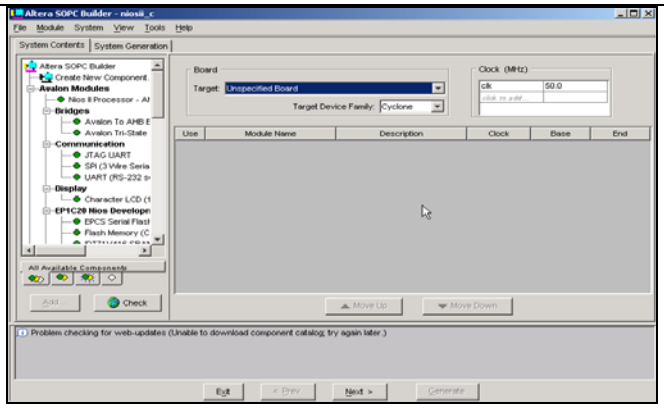


图 8.31　SOPC Builder 界面

为“Target”选择“Unspecified Board”，为“Clock（MHz）”选择时钟频率为 50.0，为“Target Device Family”选择目标器件系列为 Cyclone。

在 SOPC 定制界面的左边，我们可以看到有很多功能模块，用户可以按照需要将这些模块添加到所设计的系统中。

首先，我们需要一个 CPU。选择“NiosII Processor”/“Altera Corporation”，弹出 Altera NiosII 对话框，我们选择一个经济型的 CPU 核，即 NiosII/e，如图 8.32 所示。

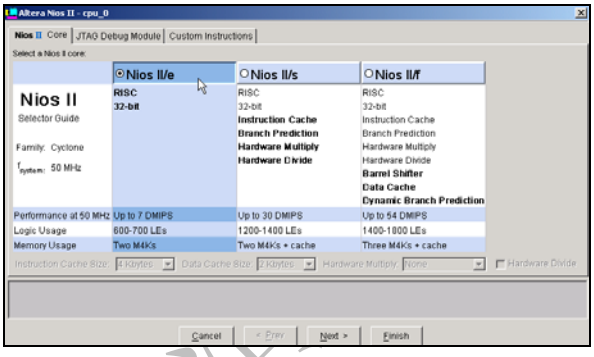


图 8.32　加入 NIOSII CPU

单击“JTAG Debug Module”标签页，选择第一级调试支持 Level1：特点是占用逻辑资源少，编译速度快，不包括缓存，如图 8.33 所示。

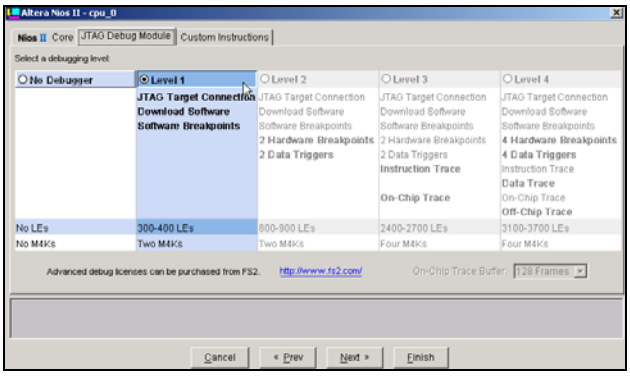


图 8.33　设置 CPU 调试模型

单击“Finish”按钮完成 NiosII CPU 的配置工作。项目中会增加一个 NiosII 处理器，名字为 cpu_0，为了简便起见，没有将它改名。改名的方法是：单击右键，选择“Rename”，输入名字后回车即可，如图 8.34。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

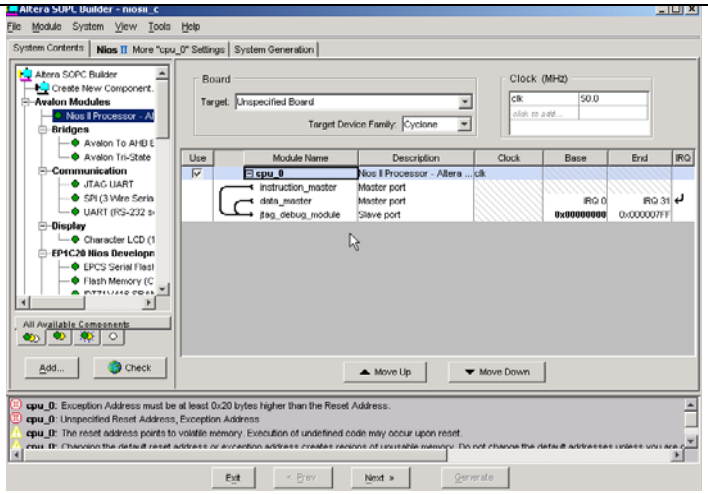


图 8.34　定制 CPU 完成图

（3）SOPC Builder 定制 JTAG 串口。

在左侧器件池中双击“JTAG UART Interface”，使用默认参数设置，如图 8.35 所示。

它是 NiosII 系统嵌入式处理器新添加的接口元件，通过内嵌在 Altera FPGA 内部的联合测试行动组（JTAG）电路，在 PC 主机和 FPGA 之间进行串行字符流通信。特点是在运行时在调试区输入信息可以与 FPGA 进行交互。

（4）SOPC Builder 定制计时器。

在左侧器件池中双击“Interval Timer”。定时器对于 HAL 系统库中的器件驱动非常有用，比如 JTAG UART 驱动使用定时器来实现 10s 的暂停。选择“Interval Timer”，在设置向导中周期选择 1 msec，预设置中选择 Full-featured，如图 8.36 所示。

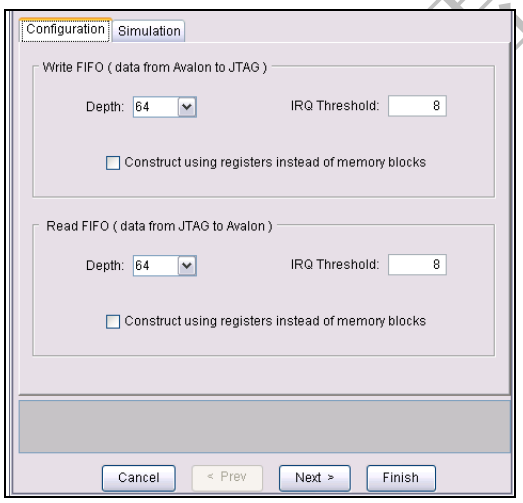


图 8.35　JTAG UART 参数设置

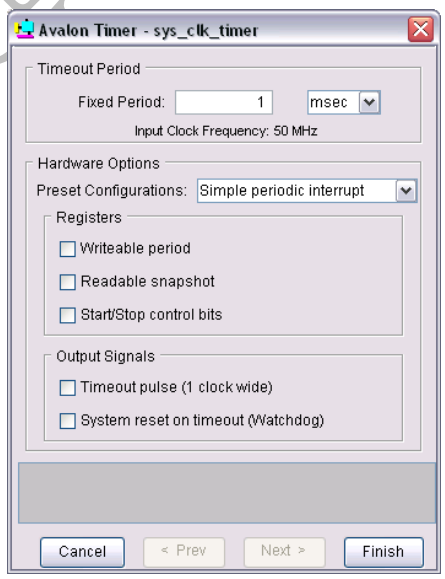


图 8.36　计时器设置

（5）SOPC Builder 定制外部 Flash。

在左侧器件池中双击“Compact Flash Interface”。在“Presets”中选自定义 Custom，地址宽度 21bits，带宽 16bits。由于 Flash 是通用设备，所以时序标签使用默认设置。如图 8.37 和 8.38 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

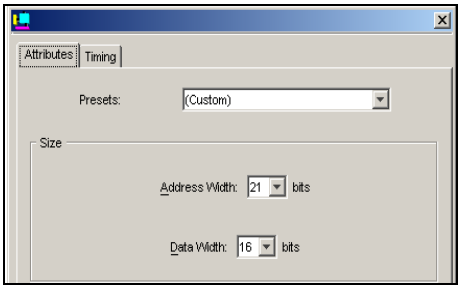


图 8.37　Flash 带宽设参数置界面

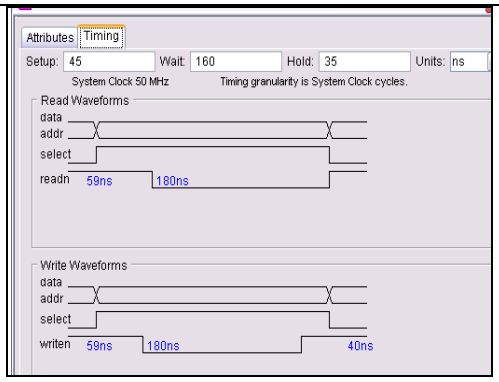


图 8.38　Flash 时序参数设置界面

（6）SOPC Builder 定制外部 SDRAM。

通常的系统都需要用户指定一个内存空间，这个内存是指 RAM，可以是片上的，也可以是片外的 SDRAM 或 SRAM 等。如果用户程序较大，超出了 FPGA 所能定制的最大片上 RAM 容量，则也可以将程序放在 SDRAM 中运行。在左侧器件池中双击“SDRAM controller”。

在“Presets”中选择“micron MT48LC4M32B2-7 chip”，其他的参数设置可以使用系统的默认选项，如图 8.39 所示。

（7）SOPC Builder 定制外部 RAM 总线。

添加外部 RAM 总线（Avalon Tri-State Bridge）。软件使用 Avalon 接口来连接片上元件和 Avalon 主从端口。

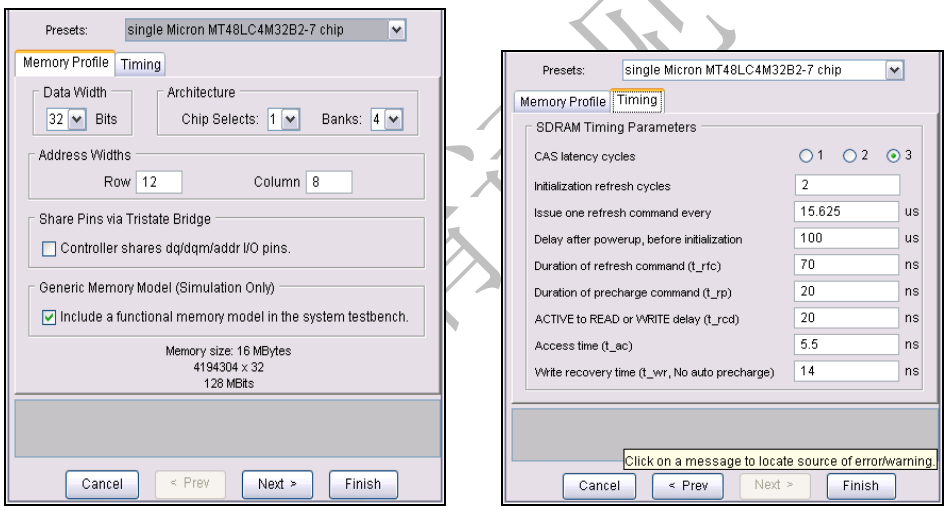


图 8.39　SDRAM 参数设置

在 Nios 开发板上，要实现 Nios 系统与 FPGA 片外存储器通信，就必须在 Avalon 总线和连接外部存储器的总线之间添加一个桥，这个桥就是 Avalon 三态总线，如图 8.40 所示。

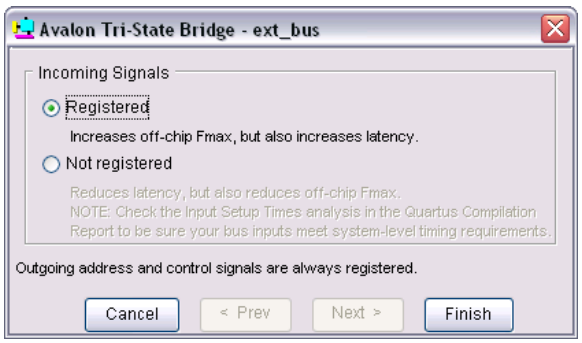


图 8.40　添加外部 RAM 总线

（8）SOPC Builder 定制串行通信设备 UART。

通用串行总线 RS-232 协议是我们最常用的通信协议，所以把它也添加到我们的系统里。注意选择所需的

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

数据帧格式，如图 8.41 所示。

（9）SOPC Builder 定制 LCD 液晶模块。

使用标准 16027 字符型液晶显示器。

（10）SOPC Builder 定制 system id。

在 SOPC Builder 中使用默认定制的 system id。

（11）SOPC Builder 定制 PIO。

添加 4 个 LED 和输入设备按键开关 PIO，这是一个标准 SOPC 一般所需要的设备。

（12）SOPC Builder 指定基地址和中断编号。

首先修改 Flash 的基地址为 0x00000000，之后在 Flash 上单击右键锁定基地址，如图 8.42 所示。

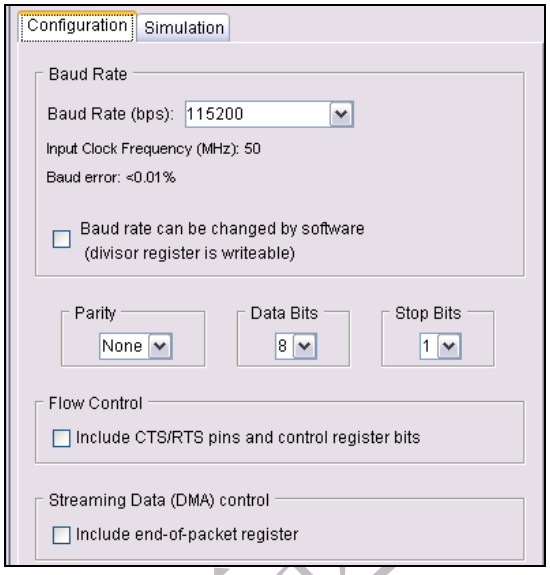


图 8.41 串行通信设备 UART 参数设置

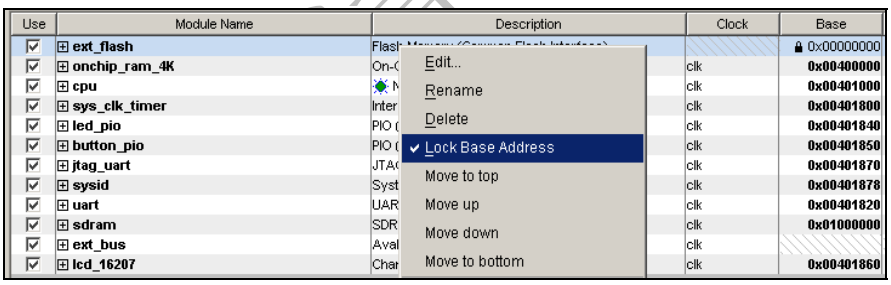


图 8.42 锁定 Flash 基地址

然后，选择“System”/“Auto-Assign Base Addresses”，让系统自动分配基地址，如图 8.43 所示。

图 8.43 自动分配系统设备地址

最后，选择“System”/“Auto-Assign IRQs”，让系统自动分配中断，如图 8.44 所示。

（13）SOPC Builder NIOSII CPU 其他设置。

单击“Nios II More “cpu_0”Settings”选项卡，进行处理器设定。在本实例中，将“Reset Address”修改为“ext_flash”，“Exception Address”修改为“sdram”，“Break Location”使用默认值，如图 8.45 所示。

图 8.44 自动分配系统设备中断编号

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

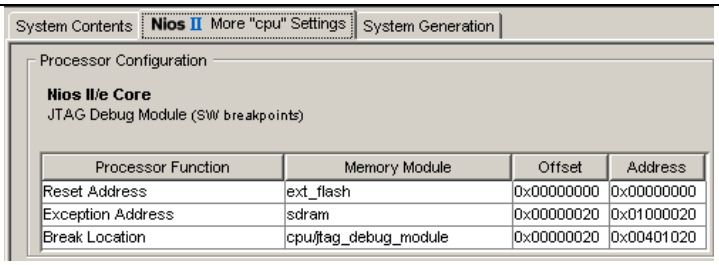


图 8.45　自动分配系统设备中断编号

（14）生成软嵌入式系统 IP 核。

单击“System Generation”选项卡，进行最后的设定并生成系统。选中“HDL Generate system module logic in Verilog”。如果需要仿真，也选择“Simulation Create ModelSim(tm) project files”，然后单击“Generate”按钮，执行系统生成的任务。

然后，耐心地等待系统的生成。一般没有问题的话，可以看到系统提示“SUCCESS: SYSTEM GENERATION COMPLETED”。如果看到此信息，则说明系统被正确生成了。如果失败，请返回并检查、修改。

（15）在 Quartus II 工程中添加上述 NIOS 系统。

类似向 Quartus II 工程添加功能模块一样，将所定制的 NiosII 软核 CPU 添加到本例工程中。双击工程顶层图空白处，弹出“Symbol”对话框，在“Project”下面可以找到刚才建立的 niosii_c 模块，如图 8.46 所示。

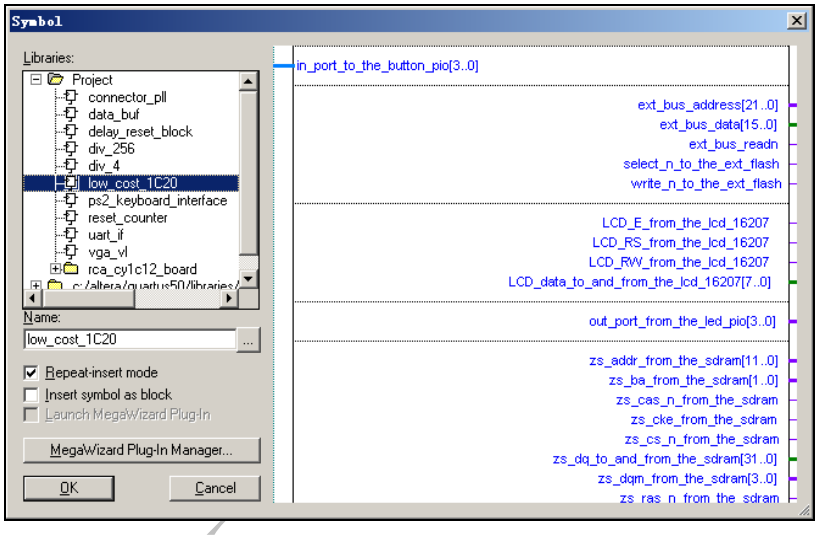


图 8.46　在 QuartusII 中添加 IP 核

单击“OK”按钮将其 niosii_c 添加到顶层图中。

（16）在 Quartus II 工程中添加端口分配管脚编译。

首先在 Quartus II 中将 NIOS 等端口添加至工程中，并为工程添加相应的管脚约束，然后即可对工程进行编译。

（17）设计该工程软件。

首先在 Altera SOPC Builder 中，点击 Run Nios II IDE，启动 Nios II IDE，如图 8.47 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

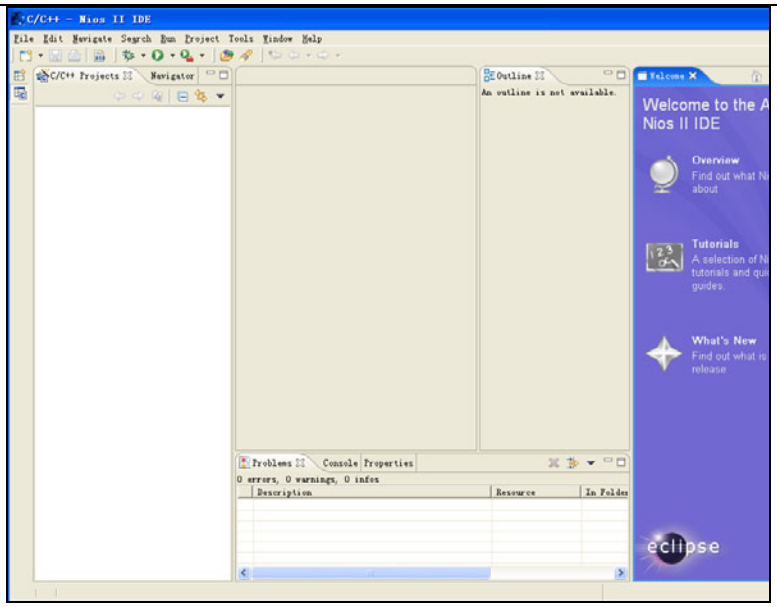


图 8.47　打开 NIOSII IDE 软件

接下来我们要设计该工程的软件。运行 Nios IDE，在 Nios II IDE 中，选择“File”/“New”/“Project”选项，开启“New Project”对话框，选择“C/C++Application”，如图 8.48 所示。

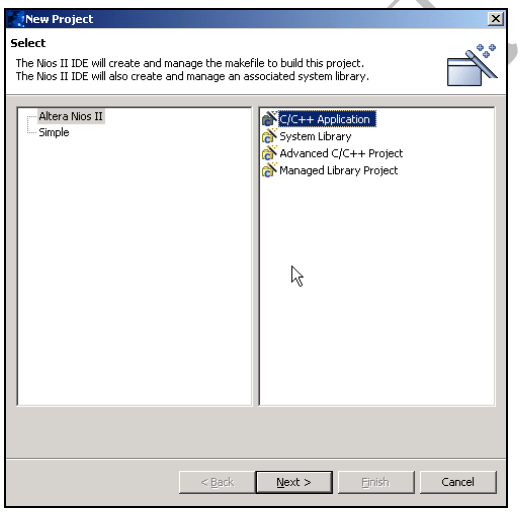


图 8.48　新建软件工程

然后单击“Next”按钮，进入下一步，在模板中选择“Hello LED”，如图 8.49 所示。

单击“SOPC Builder System”右边的“Browse”按钮打开目录对话框，从该工程目录下找到“niosii_c.ptf”，因为 NiosII IDE 必须从这个文件获取该系统的相关信息。单击“Finish”按钮，即建立一个名为 Hello_LED_0 的模板。

打开项目中的文件：hello_led.c，根据需要进行修改。在本实例程序中，“PIO_0_BASE”名称中的“PIO_0”应改为 SOPC Builder 定义时的名称，如图 8.49 中的“Hello_LED_0”。

开发板免费领取试用　开发板免费领取试用

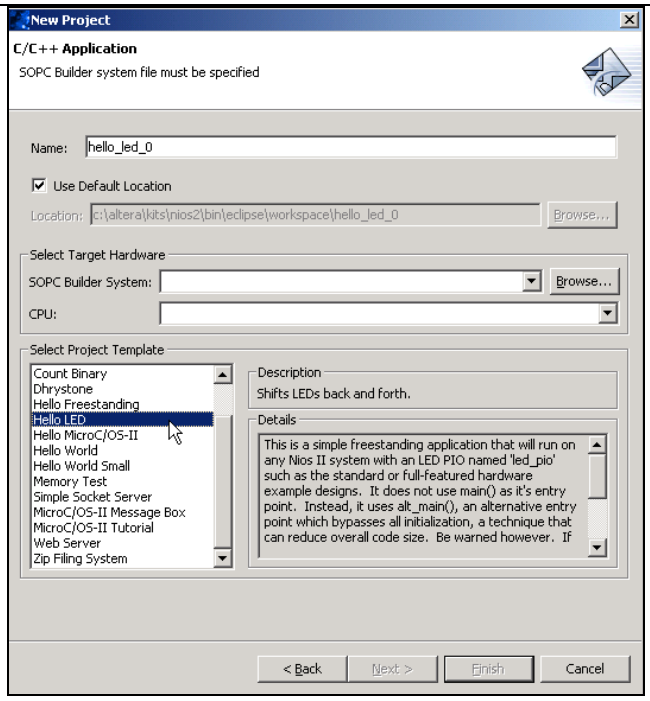


图 8.49　采用设计模板

（18）编译软件。

右键单击“hello_led_0”，在弹出的菜单中选择“Build Project”。（为使编译过程更加顺利，此时最好关闭杀毒软件和其他占用电脑资源较大的软件）。

如果无误，可以看到以下信息：Build completed。如果看到了这条提示，表示软件编译成功，可以看到程序占用空间等信息。

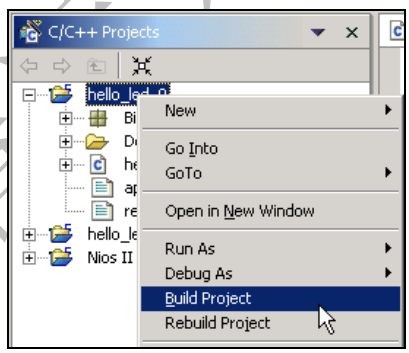


图 8.50　编译软件工程

（19）下载验证。

最后我们还要回到 Quartus II 里面选择“Tools”菜单里面的“Programmer”，下载整个配置到 FPGA 中去。

然后在 NiosII 环境中单击“Run As NiosII Hardware”按钮将 C 程序下载到实验板中。

本实例最终的硬件实现现象是 led 跑马灯。

8.4.3　小结

本节讲解了一个完整的软嵌入式设计流程。虽然此实例实现的功能比较简单，但对于初学者来说，是一个不错的入门实例。其中硬件设计部分（SoPC builder）包括开发板上所有设备，以后读者可以在此硬件平台上建立多个软件运行。

8.5　典型实例 15：基于 NIOSII 处理器的数字钟设计

8.5.1　实例的内容及目标

1．实例的主要内容

本节旨在通过给定的工程实例——“数字钟”来熟悉 Altera 软嵌入式系统的软硬件设计方法。同时使用基于 Altera FPGA 的开发板将该实例进行下载验证，完成工程设计的硬件实现。在本节中，将主要讲解以下

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

知识点。

- NIOSII IDE 软件开发流程。

读者可以迅速掌握 Altera 软嵌入式软硬件设计方法。

2．实例目标

通过详细的流程讲解，读者应达到下面的目标，

- 熟悉 NIOSII IDE 软件设计方法。

8.5.2 软件规划

电子钟的软件部分主要包括：显示、设置和时间算法 3 部分。

显示部分要求能够显示年、月、日、小时、分钟、秒，并提示时间显示状态（如设置日期、设置小时、设置分钟等）。

设置部分要求能够对电子钟显示器上面的年、月、日、小时、分钟、秒进行修改，并退出设置状态。要实现这些功能就要通过实验板上面的 4 个按键，下面是一种推荐的按键功能分配方法。

主菜单。

- SW0：设置选择按键，依次可以选择设置年、月、日、小时、分钟、秒。
- SW1：显示日期按键。
- SW2：显示时间按键。

子菜单。

- SW0：选项数值增加 1。
- SW1：选项数值减少 1。
- SW2：推出设置状态，返回主菜单。

时间算法部分比较简单，主要是要按照时间累加的原则，并设置好时间的进位要求（主要是每个月是有多少天组成的）。

8.5.3 实例步骤

（1）首先验证 8.4 节编写的硬件系统是否适合本实验硬件要求。

（2）设计软件流程图。

（3）编写、修改软件程序，下载调试。

具体的操作方式在此不再详述，相关的工程文件及代码参见本书实例代码的“典型实例 15”文件夹。

8.5.4 小结

本训练要求在实验板上面实现一个功能比较全面的电子钟。要求能够显示较为全面的时间，如：年、月、日、小时、分钟、秒并提示时间显示状态，另外还要求能够按照需要设置时间。要实现电子钟的功能重点在于软件方面的设计和优化，所以通过本实例要求用户熟练掌握软件系统的规划。

联系方式

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

集团官网：www.hqyj.com 嵌入式学院：www.embedu.org 移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn 物联网学院：www.topsight.cn 研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路银海大厦 A 座 8 层，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218

华清远见

开发板免费领取试用 开发板免费领取试用

华清远见 FARSIGHT 始于2004 | 嵌入式培训专家

10年口碑积累，成功培养50000多名研发工程师，铸就专业品牌形象

华清远见的企业理念是不仅要做良心教育、做专业教育，更要做受人尊敬的职业教育。

《FPGA 应用开发入门与典型实例》（修订版）

作者：华清远见

专业始于专注 卓识源于远见

第 9 章 FPGA 系统设计原则和技巧

本章目标

- 掌握 FPGA 系统设计的基本原则
- 掌握 FPGA 系统设计的常用技巧
- 掌握 FPGA 常用 IP 模块的使用方法

专业始于专注 卓识源于远见

开发板免费领取试用 开发板免费领取试用

9.1 FPGA 系统设计的 3 个基本原则

9.1.1 面积与速度的平衡互换原则

在 FPGA 设计领域，面积通常指的是 FPGA 的芯片资源，包括逻辑资源和 I/O 资源等。速度一般指的是 FPGA 工作的最高频率。和 DSP 或者 ARM 芯片不同，FPGA 设计的工作频率不是固定的，而是和设计本身的延迟紧密相联。

在实际设计中，使用最小的面积设计出最高的速度当然是每一个开发者追求的目标。但往往面积和速度是不可兼得的。想使用最低的成本设计出最高性能的产品是不现实的，只有兼顾面积和速度，在成本和性能之间有所取舍，才能够达到设计者的产品需求。

1．速度换面积

速度优势可以换取面积的节约。面积越小，就意味可以用更低的成本来实现产品的功能。

所谓的速度优势指的是在整个 FPGA 设计中，有一部分模块的算法运行周期较其他部分快很多，这部分模块就相对与其他的部分具有速度优势。利用这部分模块的速度优势来降低整个 FPGA 设计的使用资源就是速度换面积原则的体现。

速度换面积原则在一些较复杂的算法设计中常常会用到。在这些算法设计中，流水线设计常常是必须用到的技术。在流水线的每一级，常常有同一个算法被重复的使用，但是使用的次数不一样的现象。在正常的设计中，这些被重复使用但是使用次数不同的模块将会占用大量的 FPGA 资源。

随着 FPGA 技术的不断发展，FPGA 内部越来越多的内嵌了 DSP 乘法模块，为一些常用算法的实现提供了很大的方便，也大大提高了运算的速度和能力。因此，在以往设计中那些被重复使用的算法模块的速度可以很高，即相对其他部分具有速度优势。

利用这个特点，重新对 FPGA 的设计进行改造。将被重复使用的算法模块提炼出最小的复用单元，并利用这个最小的高速单元代替原设计中被重复使用但次数不同的模块。当然在改造的时候必然会增加一些其他的资源来实现这个代替的过程。但是只要速度具有优势，那么增加的这部分逻辑依然能够实现降低面积、提高速度的目的。

如图 9.1 所示，是一个流水线的 n 个步骤，每个步骤都相应地运算一定次数的算法，每个步骤的算法都占用独立的资源实现。其中运算次数方框的大小表示占用的设计资源。

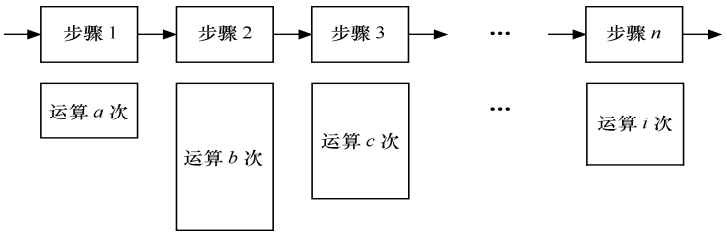


图 9.1　未使用速度换面积的流水线算法

假设这些算法中有可以复用的基本单元，并且具有速度优势，那么就可以使用如图 9.2 所示的方式实现面积的节省。在这种方法中，通过将算法提取出最小单元，配合算法次数计数器及流水线的输入输出选择开关，即可实现将原设计中复杂的算法结构简化的目的。

华清远见
华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

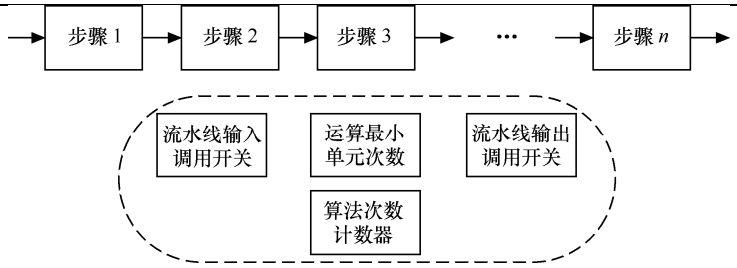


图 9.2　使用速度换面积的流水线算法

可以看到，速度换面积的关键是高速基本单元的复用。

2．面积换速度

面积换速度正好和速度换面积相反。在这种方法中，面积的复制可以换取速度的提高。支持的速度越高，就意味着可以实现更高的产品性能。在某些应用领域，比如军事、航天等，往往关注的是产品的性能，而不是成本。这些产品中，可以采用并行处理技术，实现面积换速度。

如图 9.3 所示是利用并行技术、面积（资源）复制的方法实现了高速的处理能力。

在进行 FPGA 进行设计时，我们必须注意到，FPGA 的工作频率是有限的。因为 FPGA 是工作在 TTL 电平下，该电平结构（晶体管-晶体管）所能支持的切换频率是有限的。但是在实际的产品设计中，高频的处理需求越来越多，如何解决这个矛盾？

首先使用简单的串/并转换实现多路的速度降频，如图 9.3 所示，450Mbit/s 的频率分为 3 路，每路 150Mbit/s；其次在每一路上使用相同算法但各占设计资源的处理模块进行低频（相对）的处理；最后再将每一路的处理结果进行并/串转换成为高频的输出数据。

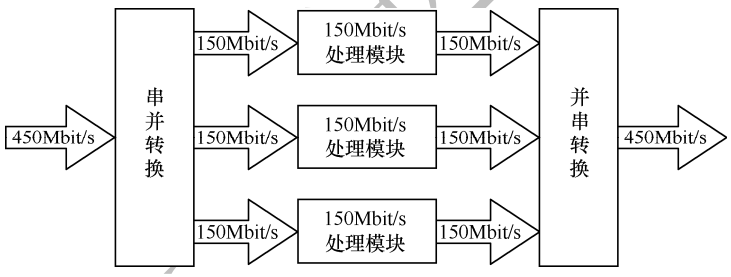


图 9.3　面积换速度实现并行高速处理

串/并和并/串转换能够支持那么高的频率吗？这个问题的解决得益于 FPGA 技术的发展。如今主流的 FPGA 器件中，都带有高速的 I/O 资源及内部 RAM 供用户使用。这些高速 I/O 资源及内部 RAM 能够实现 I/O 接口的稳定高速切换和数据总线宽度的转换。这部分的详细介绍参见 9.3 节。

9.1.2　硬件可实现原则

FPGA 设计通常会采用 HDL 语言，比如 Verilog HDL 或者 VHDL。当采用 HDL 语言来描述一个硬件电路功能的时候，一定要保证代码描述的电路是硬件可实现的。Verilog HDL 语言的语法与 C 语言很相似，但是它们之间有本质的区别。

C 语言是基于过程的高级语言，编译后可以在 CPU 上运行。而 Verilog HDL 语言描述的本身就是硬件结构，编译后是硬件电路。因此，有些语句在 C 语言的环境中应用没有问题，但是在 HDL 语言环境下就会导致结果不正确或者不理想。

比如 for 语句，以下代码在 C 语言下是一段普通代码：

```
for （i=0; i<16; i++）
   DoSomething();                    //实现函数的重复调用
```

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

在 C 语言下运行没有任何问题，但是在 Verilog HDL 的环境下编译就会导致综合后的资源严重浪费。所以 for 语句在 Verilog HDL 环境下一般只用来做仿真或者行为级的描述。那么 for 语句在 HDL 语言中如何实现呢？有几种不同的模式，其中最常见的是以下方式。

```
reg[3:0] counter;                    //调用次数计数器
always @ (posedge clk)               //计数器计数模块
    if (syn_rst)
        counter <= 4'b0;
    else
        counter <= counter + 1;
always @ (posedge clk) begin         //使用 case 语句实现分支调用，
    case (counter)                   //配合计数器计数，各分支可使用同样的执行语句
        4'b000:DoSomething;          //实现该语句的重复调用
        4'b0001: DoSomething;        //等同于高级语言中的 for 语句
        … …
        default: DoSomething;
    endcase
end
```

在 counter 计数器的不同状态可以设计不同的动作。如果都是完成一样的操作，就是循环 for 语句了。

9.1.3 同步设计原则

同步电路和异步电路是 FPGA 设计的两种基本电路结构形式。

异步设计的核心电路是由组合逻辑电路构成的，比如异步的 SRAM、FIFO 的读写控制信号，地址译码电路等。这类电路的输出信号不依赖于任何时钟信号。异步电路最大缺陷就是会产生毛刺。

同步设计的核心电路是由各种触发器构成的。这类电路的任何输出都是在某个时钟的边沿驱动触发器产生的。所以，同步设计可以很好地避免毛刺的产生。

在专用芯片（ASIC）的设计过程中，同步设计一般会比异步设计占用更多的资源。但是在 FPGA 设计过程中并不是这样。FPGA 内部的最小单元是 LE，每个 LE 里面既包括了实现异步电路需要的查找表资源，也包括了实现同步电路需要的寄存器资源。

如图 9.4 所示是 Altera 公司的 Cyclone 系列 FPGA 的 LE 结构图。从图中可以看出，这个 LE 中包含了输入输出的 MUX，同步和异步的控制逻辑，LUT 以及可编程寄存器。如果使用这个系列的 FPGA（其他的 FPGA 类似）进行同步电路或异步电路的设计，那么不管使用的是 LE 中的 LUT 还是寄存器，综合工具综合的结果都是消耗一个 LE。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

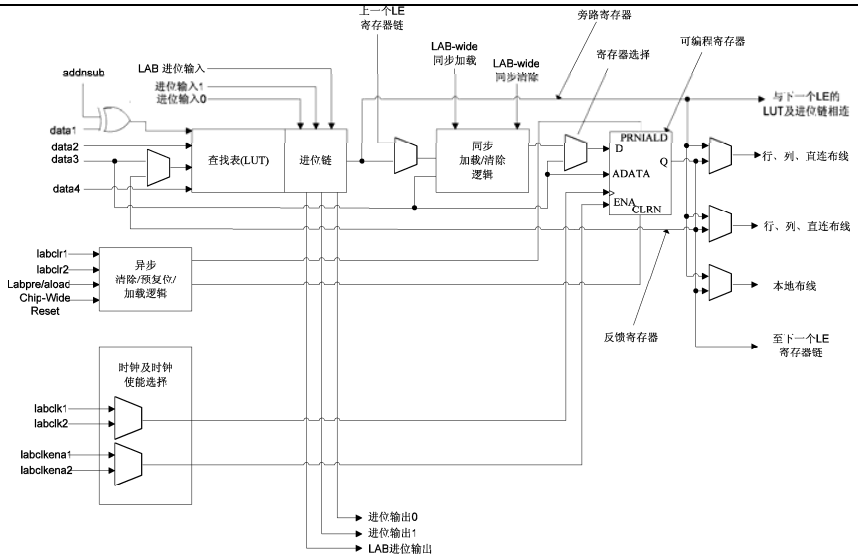


图 9.4　Cyclone LE 结构图

因此，单纯的使用异步电路也并不会节省触发器的资源。或者说，使用同步设计电路，并不会带来 FPGA 资源的浪费。但是全同步的设计对于 FPGA 的仿真验证是有好处的。因为电路的所有动作都是在相同的时钟边沿来触发，可以减少整个设计的延迟，提高系统的工作频率。

9.2　FPGA 系统设计的 3 种常用技巧

9.2.1　乒乓操作技巧

1．乒乓操作的原理

乒乓操作是 FPGA 设计中最常用的一种数据缓冲方法，简单而且实用，其原理示意图如图 9.5 所示。

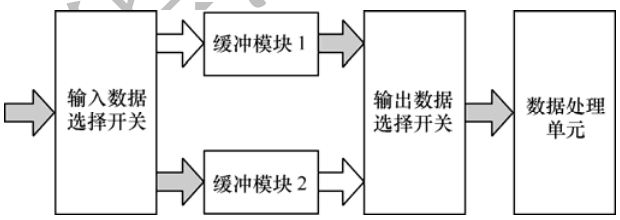


图 9.5　乒乓操作原理

输入的数据经过选择开关后，分别进入缓冲模块 1 和缓冲模块 2。当数据写入缓冲模块 1 的时候，数据处理单元从缓冲模块 2 读取数据进行处理；当数据写入缓冲模块 2 的时候，数据处理单元从缓冲区模块 1 读取数据进行处理。

这样做的目的是给数据处理单元赢得更多的处理时间，避免因为数据处理时无法持续接收而丢失有效数据。因此，乒乓操作适合那些数据处理以帧为单位，而且每帧的处理最长时间小于帧周期的情况。

2．乒乓操作的特点

（1）乒乓操作的最大特点是无缝缓冲与处理。

通过“输入数据选择开关”和“输出数据选择开关”按节拍、相互配合的切换，将经过缓冲的数据流没有时间停顿地送到“数据处理单元”，被运算与处理。

把乒乓操作模块当作一个整体，站在这个模块的两端看数据，输入数据流和输出数据流都是连续不断的，没有任何停顿。因此非常适合对数据流进行流水线式处理，常常被应用于流水线式算法。

（2）乒乓操作的第二个特点是节约缓冲区空间。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

比如在 WCDMA 基带应用中，1 帧（Frame）是由 15 个时隙（Slot）组成的。有时需要将 1 整帧的数据延时一个时隙后处理，比较直接的办法是将这帧数据缓存起来，然后延时 1 个时隙，进行处理。这时缓冲区的长度是 1 整帧数据长，假设数据速率是 3.84Mbit/s，1 帧长 10ms，则此时需要缓冲区长度是 38400bit。如果采用乒乓操作，只需定义两个能缓冲 1 个 slot 数据的 RAM（单口 RAM 即可）。当向一块 RAM 写数据的时候，从另一块 RAM 读数据，然后送到处理单元处理。此时，每块 RAM 的容量仅需 2560bit 即可，2 块 RAM 加起来也只有 5120bit 的容量。

3．乒乓操作的灵活应用

巧妙地运用乒乓操作，还可以达到用低速模块处理高速数据流的效果。

如图 9.6 所示，数据缓冲模块采用了双口 RAM，并在 DPRAM 后引入了一级处理模块。这个数据预处理根据需要可以是各种数据运算。比如在 WCDMA 设计中，对输入数据流的解扩、解扰、去旋转等。

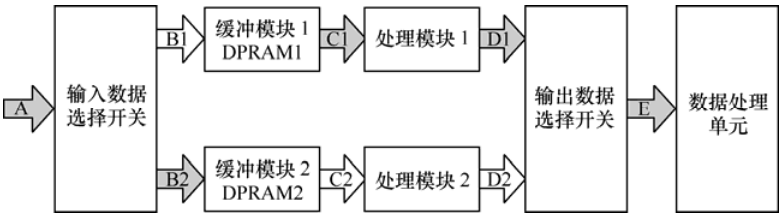


图 9.6　使用低速处理模块处理高速数据流

假设端口 A 的输入数据流的速率为 100Mbit/s，乒乓操作的缓冲周期是 10ms。我们下面一起分析一下各个节点端口的数据速率。

输入数据流 A 端口处数据速率为 100Mbit/s，在第 1 个缓冲周期 10ms 内，通过“输入数据选择开关”，从 B1 到达 DPRAM1。B1 的数据速率也是 100Mbit/s，在 10ms 内，DPRAM1 要写入 1Mbit/s 数据。

同理在第 2 个 10ms，数据流被切换到 DPRAM2，端口 B2 的数据速率也是 100 Mbit/s，DPRAM2 在第 2 个 10ms 被写入 Mbit/s 数据。周而复始，在第 3 个 10ms，数据流又切换到 DPRAM1，DPRAM1 被写入 1 Mbit/s 数据。

仔细分析一下，就会发现到第 3 个缓冲周期时，留给 DPRAM1 读取数据并送到“处理模块 1”的时间一共是 20ms。

（1）首先在第 2 个缓冲周期，向 DPRAM2 写数据的 10ms 内，DPRAM1 可以进行读操作。

（2）其次在第 1 个缓冲周期的第 5ms 起（绝对时间为 5ms 时刻），DPRAM1 就可以边向 500Kbit/s 以后的地址写数，边从地址 0 读数，到达 10ms 时，DPRAM1 刚好写完了 1Mbit/s 数据，并且读了 500Kbit/s 数据，这个缓冲时间内 DPRAM1 读了 5ms 的时间。

（3）另外在第 3 个缓冲周期的第 5ms 起（绝对时间为 35ms 时刻），同理可以边向 500Kbit/s 以后的地址写数，边从地址 0 读数，又读取了 5 个 ms，所以截止 DPRAM1 第一个周期存入的数据被完全覆盖以前，DPRAM1 最多可以读取了 20ms 时间，而所需读取的数据为 1 Mbit/s，所以端口 C1 的数据速率为：1 Mbit/s /20ms=50 Mbit/s。

因此“处理模块 1”的最低数据吞吐能力也仅仅要求为 50 Mbit/s。同理“处理模块 2”的最低数据吞吐能力也仅仅要求为 50 Mbit/s。换言之，通过乒乓操作，“处理模块”的时序压力减轻了，所要求的数据处理速率仅仅为输入数据速率的 1/2。

通过乒乓操作实现低速模块处理高速数据的实质是：通过 DPRAM 这种缓存单元，实现了数据流的串并转换，并行用“处理模块 1”和“处理模块 2”处理分流的数据，是面积与速度互换原则的有一个体现。

9.2.2　串并/并串转换技巧

串并转换是 FPGA 设计的一个重要技巧，从小的着眼点讲，它是数据流处理的常用手段；从大的着眼点讲，它是面积与速度互换思想的直接体现。

串并转换的实现方法多种多样，根据数据的排序和数量的要求，可以选用寄存器、RAM 等实现。如图 9.5

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

所示的乒乓操作举例，就是通过 DPRAM 实现了数据流的串并转换，而且由于使用了 DPRAM，数据的缓冲区可以开得很大。

对于数量比较小的设计，可以采用寄存器完成串并转换。如无特殊需求，应该用同步时序设计完成串并之间的转换。比如数据从串行到并行，数据排列顺序是高位在前，可以用下面的编码实现：

```
    prl_temp <= { prl_temp, srl_in };
```

其中，prl_temp 是并行输出缓存寄存器，srl_in 是串行数据输入。

对于排列顺序有规定的串并转换，可以用 case 语句判断实现。对于复杂的串并转换，还可以用状态机实现。串并转换的方法总的来说比较简单，在此不做更多的解释。

9.2.3 硬件流水线操作技巧

流水线处理是高速设计中的一个常用设计手段。如果某个设计的处理流程分为若干步骤，而且整个数据处理是“单流向”的，即没有反馈或者迭代运算，前一个步骤的输出是下一个步骤的输入，则可以考虑采用流水线设计方法提高系统的工作频率。

流水线设计的结构示意图如图 9.7 所示。

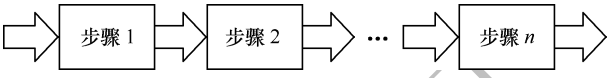


图 9.7　流水线设计的结构示意图

它的基本结构为：将适当划分的 *n* 个操作步骤单流向串联起来。

流水线操作的最大特点和要求是：数据流在各个步骤的处理从时间上看是连续的。如果将每个操作步骤假设为通过一个 D 触发器（就是用寄存器打一个节拍），那么流水线操作就类似一个移位寄存器组，数据流依次流经 D 触发器，完成每个步骤的操作。流水线设计时序示意图如图 9.8 所示。

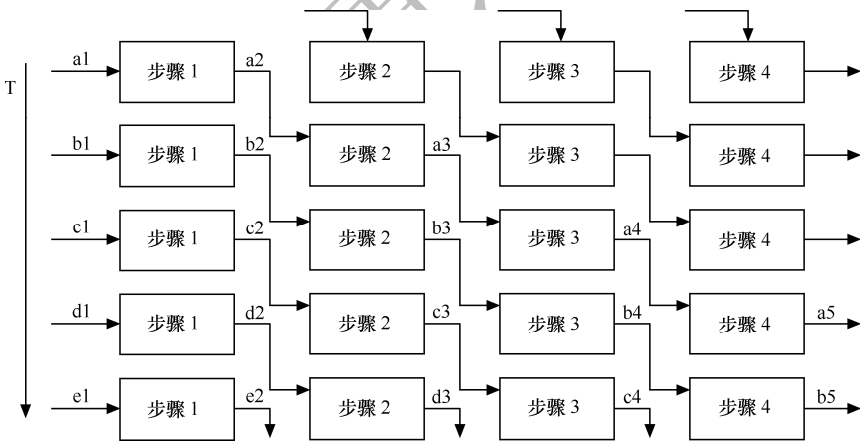


图 9.8　流水线设计时序示意图

流水线设计的一个关键在于整个设计时序的合理安排。要求每个操作步骤的划分合理。如果前级操作时间恰好等于后级的操作时间，设计最为简单，只要前级的输出直接汇入后级的输入即可。如果前级操作时间大于后级的操作时间，则需要对前级的输出数据适当缓存，才能汇入后级的输入端。如果前级操作时间恰好小于后级的操作时间，则必须通过复制逻辑，将数据流分流，或者在前级对数据采用存储、后处理方式，否则会造成后级数据溢出。

在 WCDMA 设计中经常使用到流水线处理的方法，如 RAKE 接收机、搜索器、前导捕获等。

流水线处理方式之所以频率较高，是因为复制了处理模块，它是面积换取速度思想的又一具体体现。

9.3 FPGA 系统设计的 3 种常用 IP 模块

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

FPGA 的开发工具软件，如 Quartus II、ISE 等，一般都会提供一些经过验证的 IP 模块。这些 IP 模块是芯片厂家提供的，所以只能用于该厂家的 FPGA 芯片设计中。这些 IP 主要包括以下几类。

- 算术类，如乘法器、加法器、除法器等。
- 逻辑门类，如与门、或门、非门等。
- 存储器类，如 FIFO、RAM、ROM、移位寄存器等。
- I/O 类，如双向 IO、PLL、LVDS 等。
- 接口类，如以太网 MAC、PCI 接口控制器、高速串行收发器（SERDES）等。
- 商业 IP 核，需要付费购买的。

以上的 IP 中，最常用的 3 个 IP 分别是片上存储器（RAM/ROM/FIFO），锁相环（PLL/DLL）和高速串行收发器（SERDES）。灵活使用这些 IP，可以提升设计的性能，同时降低设计的复杂度。

IP 模块的实现方式主要有 3 种：使用 HDL 代码描述、使用综合约束属性例化或类推、使用器件商的 IP core 生成器。

典型功能的 IP core 都可以通过这 3 种途径实现。其中前两种方法需要学习综合 RAM、ROM、CAM 等存储单元的 Coding Style 或约束属性，后一种方法非常方便、直接，建议初学者首先要掌握使用器件商 IP core 生成器设计 RAM、ROM、CAM 等存储单元的方法。

下面通过 Quartus II 工具分别介绍 Altera 的 3 种 IP 模块的使用方法。

9.3.1 片上存储器的使用方法

片上存储器由内嵌于 FPGA 内部的逻辑资源（分布 RAM 和块 RAM）搭建而成。搭建出来的这些片上存储器的调用方法与分立器件基本一致，但却为开发者省去了大量的 PCB 布线资源。在小容量的存储器设计中有着非常广泛的应用，同时这些片上存储器还常常用作数据缓冲及时钟域转换等工作。

首先来介绍片上 FIFO 的使用方法。

1．FIFO

（1）打开宏模块向导管理器。

在 Quartus II 中，IP 模块的生成都是通过“MegaWizard Plug-In Manager”（宏模块向导管理器）实现的，它可以通过如图 9.9 的“Tools”菜单打开。

（2）选择新建宏模块。

在宏模块向导管理器的第 1 页，选择新建一个自定义的宏模块，如图 9.10 所示。

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

图 9.9　宏模块向导管理器调用

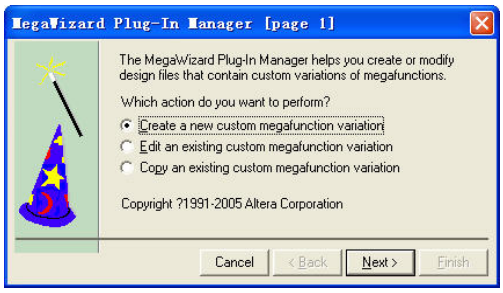


图 9.10　新建宏模块

（3）选择宏模块。

在宏模块向导管理器的第 2 页，管理器为我们提供了支持的宏模块树形目录。通过在该目录中选择相应的宏模块实现调用。同时在这一页中还可以选择应用的 FPGA 器件系列和宏模块的描述语言，并使用用户自定义的模块名，如图 9.11 所示。

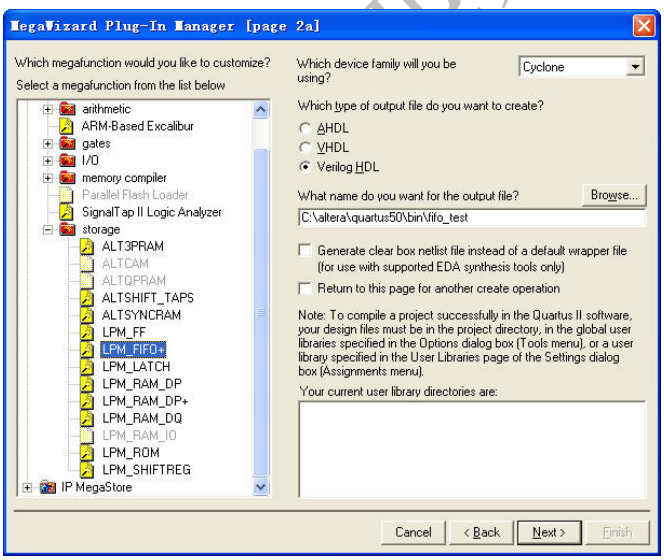


图 9.11　选择宏模块

在本例中，我们选择“LPM_FIFO+模块”进行实现。

（4）设置 FIFO 宽度和深度。

在宏模块向导管理器（FIFO）的第 3 页，可以设置 FIFO 的深度和宽度，同时在本页的左下角会计算出实现这样一个深度和宽度的 FIFO 所消耗的 FPGA 资源，如图 9.12 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

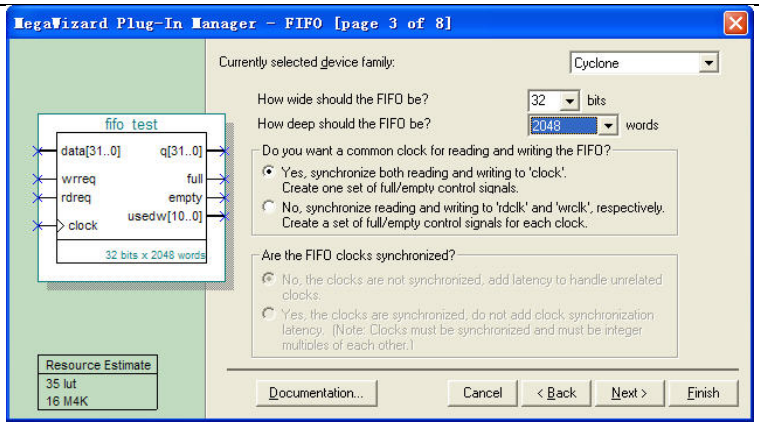


图 9.12　设置 FIFO 深度和宽度

（5）设置 FIFO 的控制信号。

在宏模块向导管理器（FIFO）的第 4 页，可以设置 FIFO 的控制信号，包括满信号 full、空信号 empty、使用字节信号组 usedw[ ]、几乎满信号 almost full（可编程）、几乎空信号 almost empty（可编程）、异步清零信号和同步清零信号。

通过选择是否打开这些信号来构造一个用户自定义的 FIFO，如图 9.13 所示。

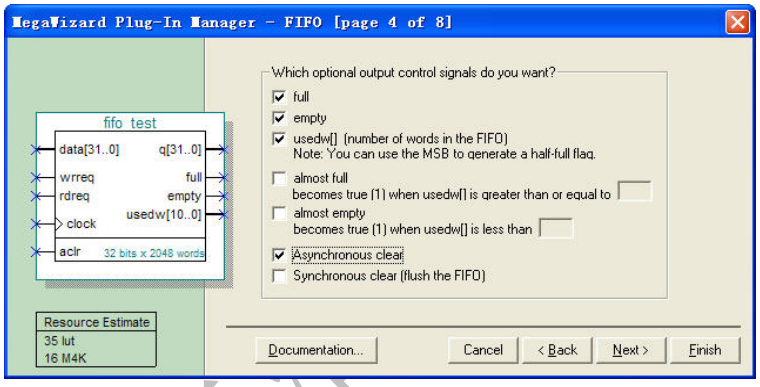


图 9.13　设置 FIFO 的控制信号

（6）设置 FIFO 的模式。

在宏模块向导管理器（FIFO）的第 6 页，可以设置 FIFO 的模式。分为 Lagacy 同步模式和 Show-ahead 同步模式。区别在于输出数据是在 FIFO 的读请求信号 rdreq 发出之前还是之后有效，用户可以根据需要进行选择，如图 9.14 所示。

（7）设置 FIFO 的外部属性。

在宏模块向导管理器（FIFO）的第 7 页，可以设置 FIFO 的外部属性，包括输出寄存器使用最佳速度策略还是最小面积策略，数据溢出及读空状态下的保护机制，还可以强制只利用逻辑单元来构造 FIFO，如图 9.15 所示。

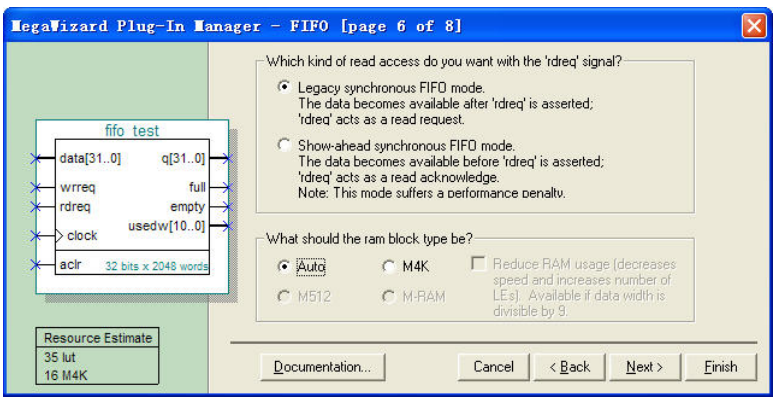


图 9.14　设置 FIFO 的模式

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

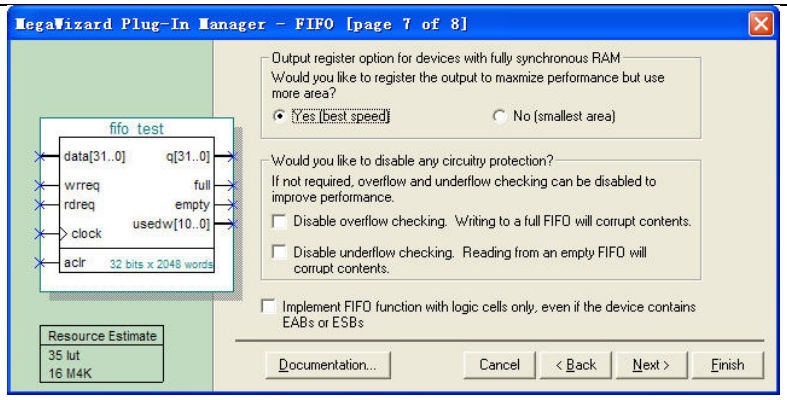


图 9.15　设置 FIFO 的外部属性

（8）选择生成的 FIFO 模块文件。

在宏模块向导管理器（FIFO）的第 8 页，也就是最后一页，可以选择生成的 FIFO 模块文件。在 Quartus II 软件中，宏模块向导管理器可为 FIFO 生成 7 个文件，如图 9.16 所示。

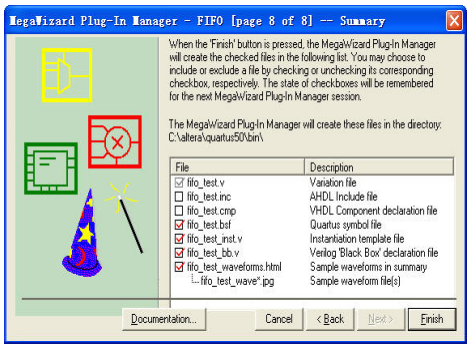


图 9.16　选择生成的 FIFO 模块文件

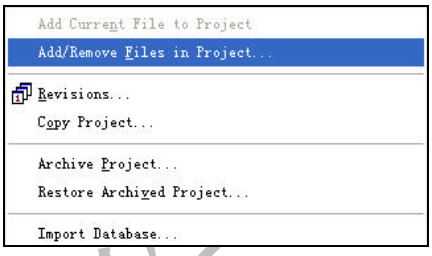


图 9.17　打开工程添加/删除文件对话框

（9）向工程添加 FIFO 模块文件。

生成 FIFO 模块的文件后，在工程的目录下生成了选择的文件。要在工程中调用这些模块，首先要将这些文件添加到工程中来。

打开“Project”菜单，选择其中的“Add/Remove Files in Project…”选项，如图 9.17 所示，打开工程添加/删除文件对话框。

在打开的对话框中，选择向工程添加 3 个文件，如图 9.18 所示。

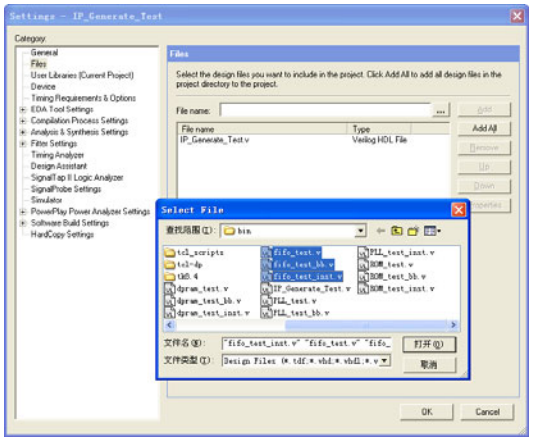
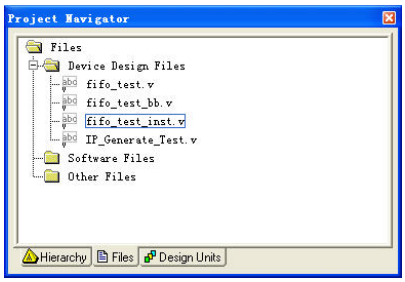


图 9.18　添加 FIFO 模块文件图



9.19　工程浏览器

添加成功后，在工程浏览器中，可以看到器件设计文件中已经包含了这 3 个文件，如图 9.19 所示。

（10）FIFO 模块实例化。

将模块文件加入工程后，设计者就可以调用这个片上 FIFO，并实例化了。

首先我们看一下 Quartus II 是如何构建这个 FIFO 的。打开 fifo_test.v 文件，这是 FIFO 模块的构造设计文

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

件。该构造设计文件主要可以分为下面 4 个部分。

第一部分是端口声明和说明，这部分是根据用户自定义选择的控制信号及深度、宽度等参数决定的，代码如下：

```
module fifo_test (
    data,                   //数据输入
    wrreq,                  //写请求
    rdreq,                  //读请求
    clock,                  //时钟
    aclr,                   //异步清零
    q,                      //数据输出
    full,                   //满信号
    empty,                  //空信号
    usedw);                 //字节使用信号
    input   [31:0]  data;   //输入数据宽度
    input   wrreq;
    input   rdreq;
    input   clock;
    input   aclr;
    output  [31:0]  q;      //输出数据宽度
    output  full;
    output  empty;
    output  [10:0]  usedw;  //自动计算出的字节使用控制信号宽度
```

第二部分是 FIFO 端口信号与 Altera 宏模块之间的连线声明，其中只有输出信号需要声明，而输入信号可以直接调用，代码如下：

```
wire [10:0] sub_wire0;                  //usedw 输出类型声明
wire  sub_wire1;                        //空信号输出类型声明
wire [31:0] sub_wire2;                  //数据输出类型声明
wire  sub_wire3;                        //满信号输出类型声明
wire [10:0] usedw = sub_wire0[10:0];    //usedw 连线
wire  empty = sub_wire1;                //空信号连线
wire [31:0] q = sub_wire2[31:0];        //数据输出连线
wire  full = sub_wire3;                 //满信号连线
```

第三部分是实例化 FIFO 所调用的宏模块，这里调用的是 scfifo 模块，代码如下：

```
scfifo   scfifo_component (             //scfifo 模块实例化
            .rdreq (rdreq),             //读请求
            .aclr (aclr),               //异步清零
            .clock (clock),             //时钟
            .wrreq (wrreq),             //写请求
            .data (data),               //输入数据
            .usedw (sub_wire0),         //字节使用信号
            .empty (sub_wire1),         //空信号
            .q (sub_wire2),             //数据输出
            .full (sub_wire3)           //满信号
            .almost_empty (),           //几乎空信号
            .almost_full (),            //几乎满信号
```


开发板免费领取试用 开发板免费领取试用

```
            .sclr ()                    //清零信号
            .);
```

第四部分是参数设置，这里设置的是 scfifo 模块的参数，包括宽度、深度以及之前生成步骤中设置的那些参数，代码如下：

```
defparam
     scfifo_component.lpm_width = 32,                        //宽度为 32 位
     scfifo_component.lpm_numwords = 2048,                   //深度为 2048 字节
     scfifo_component.intended_device_family = "Cyclone",//器件族为 Cyclone
     scfifo_component.lpm_type = "scfifo",                   //调用 lpm 为 scfifo
     scfifo_component.lpm_showahead = "OFF",                 //关闭 showahead 模式
     scfifo_component.overflow_checking = "ON",              //打开溢出校验
     scfifo_component.underflow_checking = "ON",             //打开读空校验
     scfifo_component.add_ram_output_register = "ON";  //使用输出寄存器
```

在生成的 3 个 FIFO 模块文件中，还包含一个 fifo_test_bb.v 文件，这个文件包含的是 fifo_test.v 文件的第一部分内容，也就是说这个文件是 FIFO 模块的端口声明模块。

剩下的一个文件是 fifo_test_inst.v，inst 是实例化的简称。在这个文件里面提供了 FIFO 模块实例化的模板，代码如下：

```
fifo_test   fifo_test_inst (                //fifo_test 模块实例化
     .data ( data_sig ),                    //输入数据
     .wrreq ( wrreq_sig ),                  //写请求
     .rdreq ( rdreq_sig ),                  //读请求
     .clock ( clock_sig ),                  //时钟
     .aclr ( aclr_sig ),                    //异步清零
     .q ( q_sig ),                          //输出数据
     .full ( full_sig ),                    //满信号
     .empty ( empty_sig ),                  //空信号
     .usedw ( usedw_sig )                   //字节使用信号
);
```

通过复制这个模板至工程设计文件中，即可实现 FIFO 模块的实例化。在进行实例化时，可以修改实例化的名称以及 FIFO 模块端口信号线的连线资源名称。

下面是将 FIFO 实例化在 IP_Generate_Test 工程中的代码。

```
module IP_Generate_Test(...);           //调用 FIFO 模块的设计文件模块
     ...                                //连线资源声明
     fifo_test   fifo_test_inst1 (      //实例化为 fifo_test_inst1
          .data ( fifo_data ),          //FIFO 数据输入为 fifo_data
          .wrreq (fifo_wrreq),
          .rdreq (fifo_rdreq),
          .clock (fifo_ clock),
          .aclr (fifo_aclr),
          .q (fifo_ q ),
          .full (fifo_ full ),
          .empty (fifo_empty ),
          .usedw ( )                    //未实例化的端口保留空白表示不使用该端口
          );
     ...
endmodule
```

至此，已经在一个工程中实现了片上 FIFO 的调用。

开发板免费领取试用 开发板免费领取试用

2．RAM

这里主要介绍双口 RAM 模块的生成方法，模块的构造方法及调用方法与 FIFO 模块类似，不详细介绍。

（1）打开宏模块向导管理器并新建宏模块。

具体方法参见 FIFO 的方法。

（2）选择宏模块。

在本例中，我们选择 LPM_RAM_DP 模块进行实现，如图 9.20 所示。

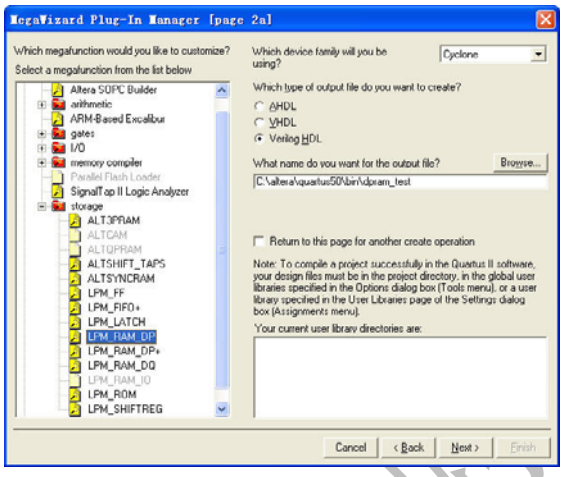


图 9.20　选择宏模块

（3）设置 DPRAM 端口数及容量单位。

在宏模块向导管理器（DPRAM）的第 3 页，可以设置 DPRAM 的端口数及容量单位。支持两种 DPRAM：1 个读端口和 1 个写端口的 DPRAM，2 个读端口和 2 个写端口的 DPRAM。可以设置 DPRAM 的容量单位为位或者是字节。同时在本页的左下角会计算出实现这样一个 DPRAM 所消耗的 FPGA 资源，如图 9.21 所示。

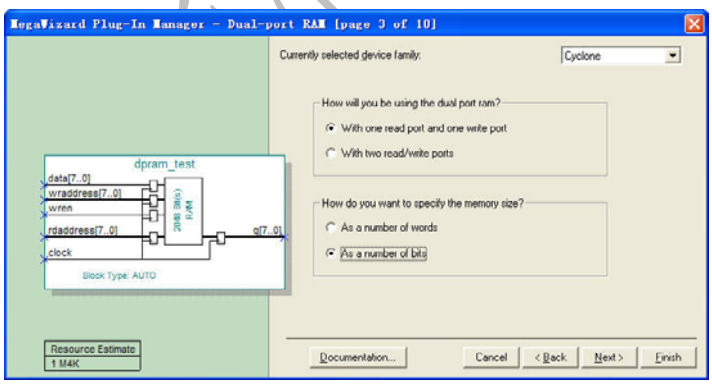


图 9.21　选择宏模块

（4）设置 DPRAM 数据宽度及容量。

在宏模块向导管理器（DPRAM）的第 4 页，可以设置 DPRAM 的数据宽度及容量，同时可以设置输入输出端口为不同的宽度实现串并或并串转换，如图 9.22 所示。

（5）设置 DPRAM 时钟及使能。

在宏模块向导管理器（DPRAM）的第 5 页，可以设置 DPRAM 的时钟及使能信号，可以将 DPRAM 设置为单时钟（输入输出使用同一个时钟），也可以使用独立的时钟。另外还可以为 DPRAM 增加读使能信号 rden，如图 9.23 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

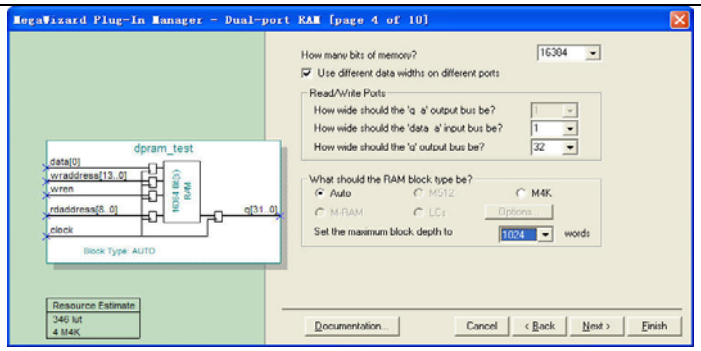


图 9.22　设置 DPRAM 数据宽度及容量

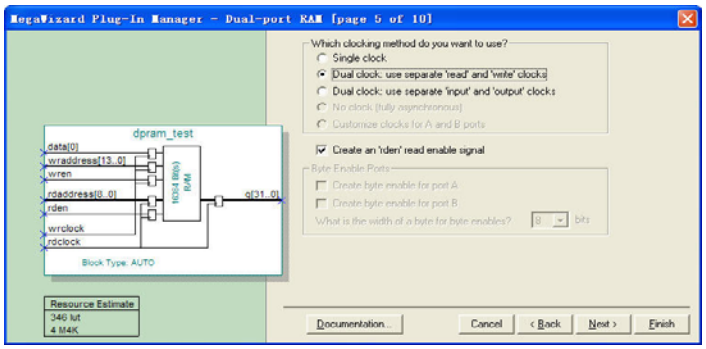


图 9.23　设置 DPRAM 时钟及使能

（6）设置 DPRAM 端口寄存器及清零信号。

在宏模块向导管理器（DPRAM）的第 7 页，可以设置 DPRAM 的端口寄存器及清零信号，可以选择是否增加端口的寄存器，如图 9.24 所示。

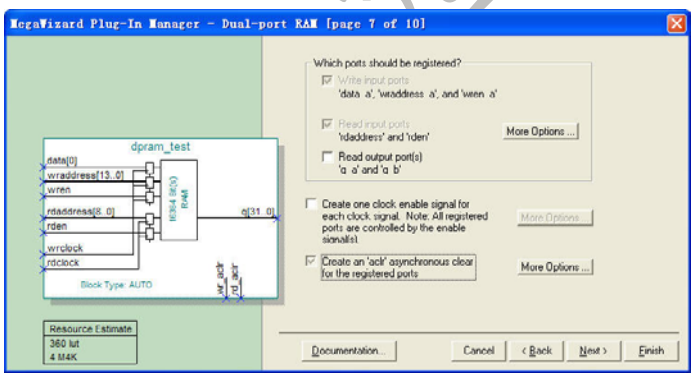


图 9.24　设置 DPRAM 端口寄存器及清零信号

（7）设置 DPRAM 初始化值。

在宏模块向导管理器（DPRAM）的第 9 页，可以设置 DPRAM 的初始化值。通过选择 MIF 文件或者 HEX 文件可以使用文件中的值对 DPRAM 进行初始化，如图 9.25 所示。

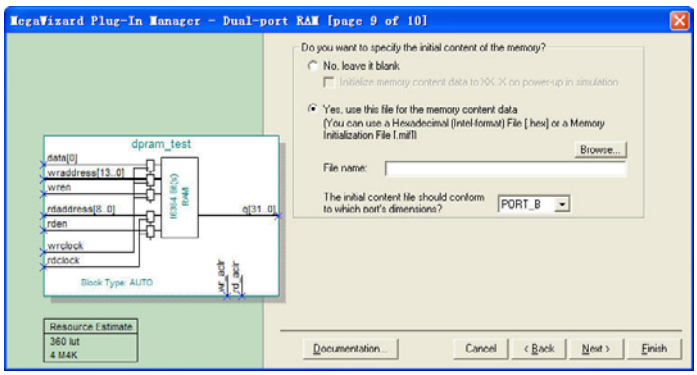


图 9.25　设置 DPRAM 初始化值

（8）生成 DPRAM 模块文件。

开发板免费领取试用　开发板免费领取试用

设置完成所有的参数，来到宏模块向导管理器（DPRAM）的最后一页。可以选择生成的 DPRAM 模块文件，共有 6 个文件可以生成，如图 9.26 所示。

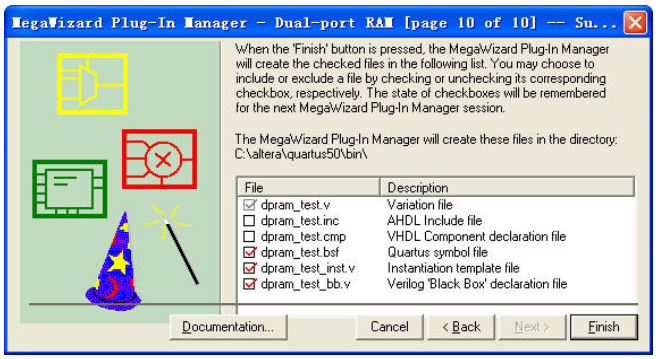


图 9.26　生成 DPRAM 模块文件

DPRAM 的调用方式与 FIFO 的类似，其构建方式及模块的声明结构可以参看 DPRAM 模块的 dpram_test.v 和 dpram_test_bb.v 文件。

3．ROM

这里主要介绍 ROM 模块的生成方法，模块的构造方法及调用方法与 FIFO 模块类似，就不详细介绍了。

（1）打开宏模块向导管理器并新建宏模块。

具体方法参见 FIFO 的方法。

（2）选择宏模块。

在本例中，我们选择 LPM_ROM 模块进行实现，如图 9.27 所示。

（3）设置 ROM 宽度、深度及时钟控制方式。

在宏模块向导管理器（DPRAM）的第 3 页，可以设置 DPRAM 的宽度、深度及时钟控制方式。ROM 宽度指的是存储器数据位宽，深度指的是具有该宽度的存储单元个数。宏模块向导管理器会根据设计者的选择自动生成 ROM 的地址总线，同时也会在左下角显示消耗的 FPGA 逻辑资源情况。

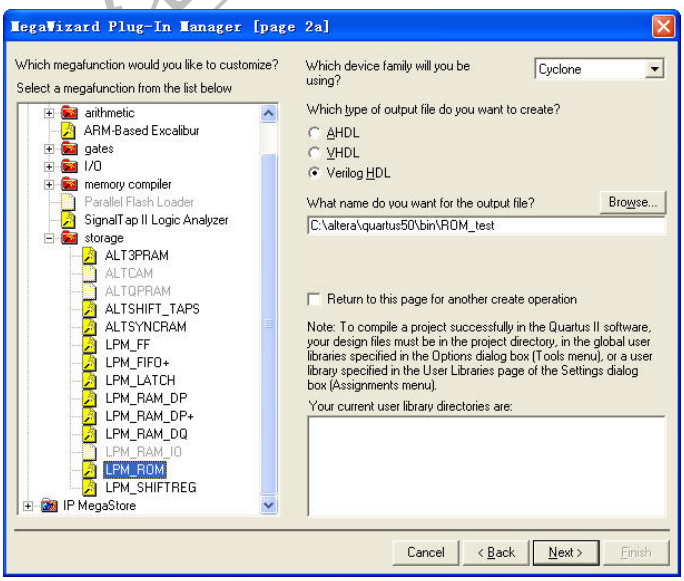


图 9.27　选择宏模块

ROM 的时钟控制方式分为两种，一种为单时钟，另一种将输入时钟和输出时钟分离。ROM 的输入输出使用的是同一个数据端口，时钟的分离实际上也是对同一个数据端口作控制，如图 9.28 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

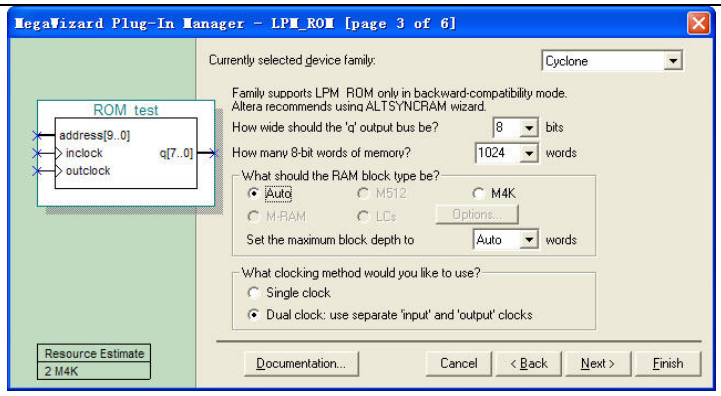


图 9.28　设置 ROM 宽度、深度及时钟控制方式

（4）设置端口寄存器及清零信号。

这个步骤与 RAM 中的第（6）步类似，不再详述，如图 9.29 所示。

（5）设置 ROM 初始化值。

同样可以参看 RAM 使用方法的第（7）步，如图 9.30 所示。

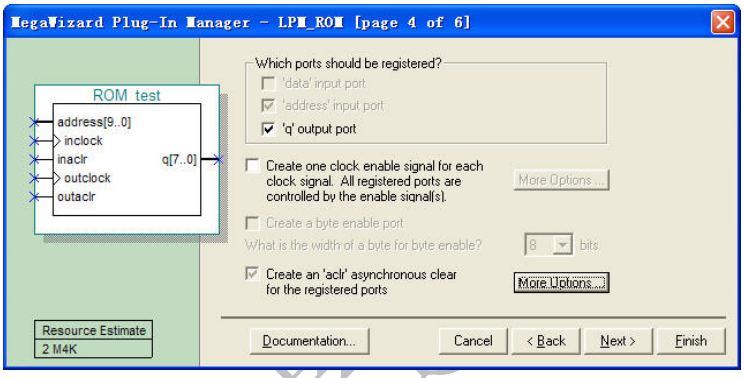


图 9.29　设置端口寄存器及清零信号

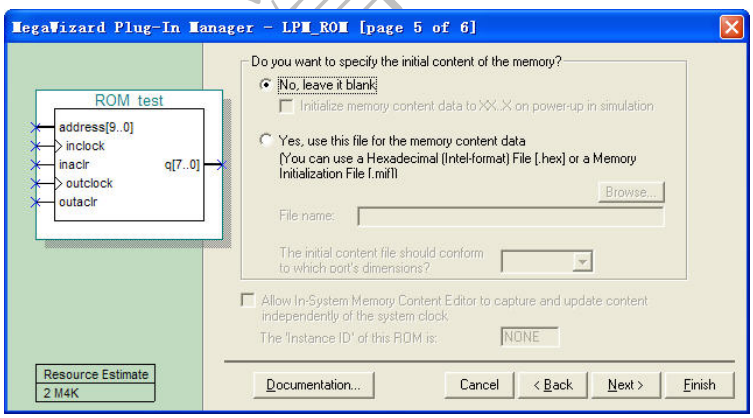


图 9.30　设置 ROM 初始化值

（6）选择生成 ROM 模块文件。

设置完成所有的参数，来到宏模块向导管理器（ROM）的最后一页。可以选择生成的 ROM 模块文件，同样有 6 个文件可以生成，如图 9.31 所示。

开发板免费领取试用　开发板免费领取试用

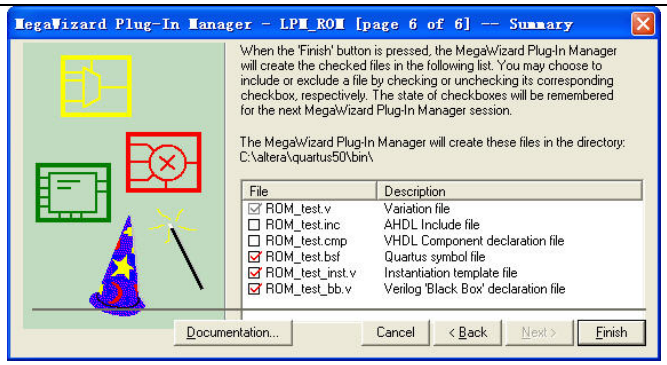


图 9.31　生成 ROM 模块文件

ROM 的调用方式与 FIFO 的类似，其构建方式及模块的声明结构可以参看 ROM 模块的 ROM_test.v 和 ROM_test_bb.v 文件。

9.3.2　锁相环的使用方法

锁相环在数字系统中的应用非常广泛，在 Altera 的 FPGA 产品中集成的锁相环是 PLL 模块，在 Xilinx 的 FPGA 产品中集成的锁相环是 DLL 模块。两者采用的技术不同，但是实现的基本功能大致相同，但也各有特点。

锁相环的原理及在 FPGA 内部的构造这里将不做介绍，只以 Altera 的 Cyclone FPGA 的 PLL 为例讲解其使用方法。

Cyclone PLL 具有时钟倍频和分频、相位偏移、可编程占空比和外部时钟输出，进行系统级的时钟管理和偏移控制。PLL 常用于同步内部器件时钟和外部时钟，使内部工作的时钟频率比外部时钟更高，时钟延迟和时钟偏移最小，减小或调整时钟到输出（TCO）和建立（TSU）时间。

（1）打开宏模块向导管理器并新建宏模块。

具体方法参见 FIFO 的方法。

（2）选择宏模块。

在本例中，我们选择 ALTPLL 模块进行实现，如图 9.32 所示。

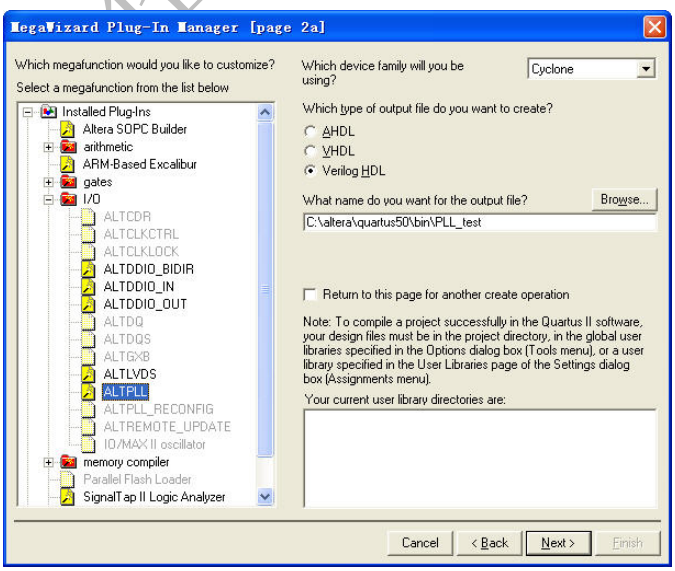


图 9.32　选择宏模块

（3）设置时钟源，PLL 类型及工作模式。

在宏模块向导管理器（PLL）的第 3 页，设计者可以设置一些 PLL 基本属性，如器件族、速度级别及基准频率，还可以设置 PLL 的类型及工作模式。

值得注意的是，这里的基准频率并不一定是最终设计者向 PLL 输入的时钟。最终的 PLL 是通过比例值来

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

对输入频率进行综合，所以 PLL 的关键参数是比例值，即图 9.33 中 PLL_test 框图中的 Ratio 值。
例如 Ratio 值为 5/2（通过后面几个步骤实现），而基准频率为 40MHz，则 PLL 的 c0 输出为 100MHz。在调用这个 PLL 时，若设计者使用 20MHz 的频率输入，则输出的频率为 50MHz，而不是 100MHz。
另外，在这页中，设计者还可以设置 PLL 的工作模式，如图 9.33 所示。

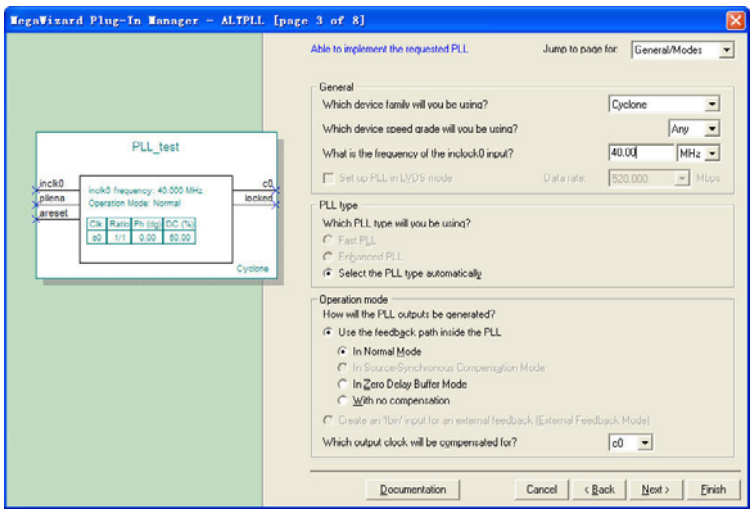


图 9.33　设置时钟源、PLL 类型及工作模式

Cyclone PLL 支持 3 种（反馈）模式：标准、零延迟缓冲和无补偿。和其他 Altera 器件系列不同，Cyclone PLL 不支持外部反馈模式。所有支持的 3 种时钟反馈模式允许倍频/分频、相位偏移和可编程占空比。下面对每种模式进行讲解。

- 标准模式。

在标准模式下，PLL 把参考时钟和逻辑阵列或 I/O 单元的端口缓存器处的时钟信号相位对齐，补偿内部全局时钟网络延迟。在 ALTPLL 宏模块向导管理器中，可以定义 PLL 的哪个内部时钟输出（c0 或 c1）应该补偿。
如果在该模式中使用外部时钟输出（PLL[2..1]_OUT），则相对于时钟输入管脚有相位偏移。如果用内部 PLL 时钟输出驱动通用 I/O 管脚，则相对应的时钟输入管脚也有相位偏移。

- 零延迟缓冲模式。

PLL 外部时钟输出管脚（PLL[2..1]_OUT）的时钟信号和 PLL 输入时钟是相位对齐的，没有延迟。如果用 c[1..0]端口驱动内部时钟管脚，那么相对于输入时钟管脚有相位偏移。

- 无补偿模式。

在该模式下，PLL 不补偿任何时钟网络。这样会有更佳的抖动性能，因为反馈到 PFD 的时钟不经过某些电路。相对 PLL 时钟输入，PLL 内部和外部时钟输出都有相位偏移。
（4）设置可选输入输出。
在宏模块向导管理器（PLL）的第 4 页，可以选择 PLL 模块的可选输出信号，如图 9.34 所示。
Cyclone PLL 有 4 个控制信号 pllena、areset、prdena 和 locked，进行 PLL 管理。

- pllena。

PLL 启动信号 pllena 用于启动 PLL。当 pllena 为低时，PLL 时钟输出端口为低，PLL 失锁。当 pllena 再次变高时，PLL 重新锁定并重新同步输入时钟。因此，pllena 是高有效信号。
因为在 Cyclone FPGA 中没有专用的 pllena 管脚，内部逻辑或任意通用 I/O 管脚都可以驱动 pllena 端口。因为每个 PLL 都有自己的 pllena 控制电路或共享通用的 pllena 电路，这样就很灵活。pllena 信号是可选的，如果软件中没有启动它，端口内部就连接到 VCC。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

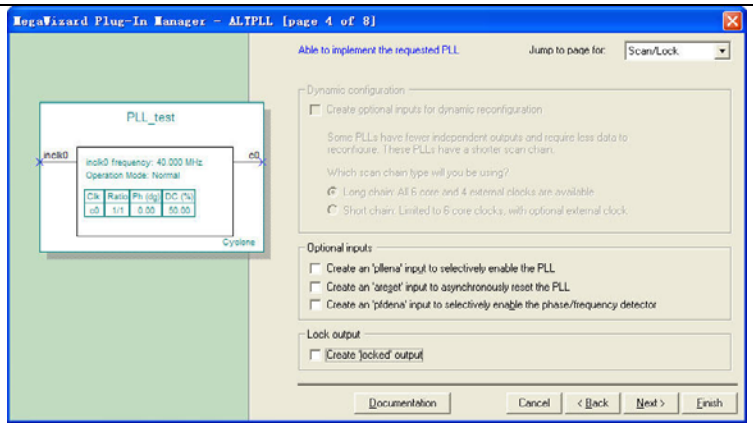


图 9.34　设置可选输入

- areset。

PLL areset 信号是每个 PLL 的复位或重新同步输入。但驱动为高时，PLL 计数器重置，清除 PLL 输出，造成 PLL 失锁，VCO 复位后回到初始设置。当 areset 再次变低，PLL 重新开始锁定，PLL 重新和输入时钟同步。如果目标 VCO 的频率低于标准频率，在锁定过程中 PLL 时钟输出起始频率值比所需值要高。areset 是高有效信号。

Cyclone FPGA 可以从内部逻辑或任意通用 I/O 管脚驱动这个 PLL 输入信号。areset 信号是可选的，如果在软件中没有使用它，该端口内部连接到 GND。

- pfdena。

pfdena 信号用可编程开关控制着 PLL 中 PFD 输出。如果把 areset 置低禁止 PFD，那么 VCO 将以最后设置的控制电压和频率值工作，长时间会漂移到更低的频率。即使每个输入时钟 PLL 时钟输出也会继续触发，但是 PLL 可能会失锁。

当 PLL 失锁或输入时钟禁止时，系统会继续运行。因为在一段时间内最后锁定输出频率不会改变，所以可以用 pfdena 端口作为关机或清除功能。为了维持这一频率，系统在关机之前有时间储存当前的设置。如果 pfdena 信号再次变高，PLL 重新锁定和输入时钟重新同步。因此 pfdena 管脚是高有效信号。

可以用任意通用 I/O 管脚或内部逻辑驱动 pfdena 输入信号。该信号是可选的，如果在软件没有使用它，该端口内部连接到 VCC。

- locked。

当 locked 输出是逻辑高电平，表明 PLL 时钟输出和 PLL 参考输入时钟稳定同相。当 PLL 开始跟踪参考时钟时，locked 端口可能会触发，无需额外电路。

PLL 的 locked 端口可以使用任意通用 I/O 管脚和内部逻辑。这个 locked 信号是可选的，在监视 PLL 锁定过程中是非常有用的。

（5）设置 c0 输出参数。

在宏模块向导管理器（PLL）的第 5 页，可以选择 PLL 模块 c0 输出参数，包括 Ratio 比值、相移量、占空比等，如图 9.35 所示。

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

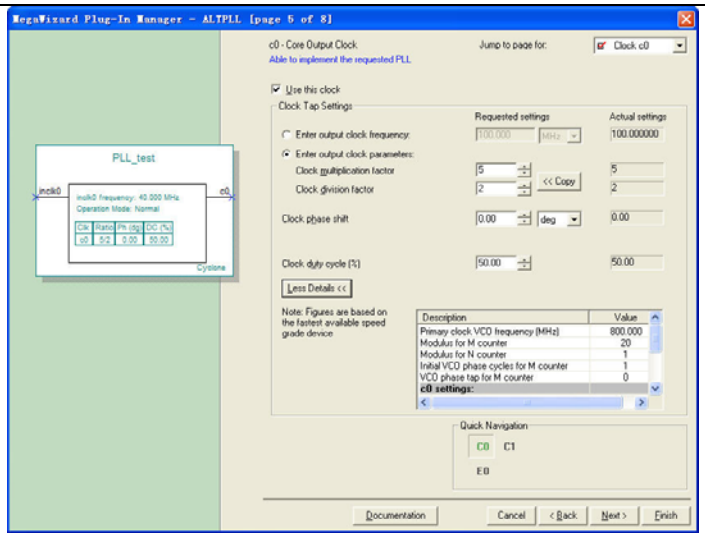


图 9.35　设置 c0 输出参数

（6）设置 c1 输出参数。

在宏模块向导管理器（PLL）的第 6 页，同样可以选择 PLL 模块 c1 输出参数，包括 Ratio 比值、相移量、占空比等，如图 9.36 所示。

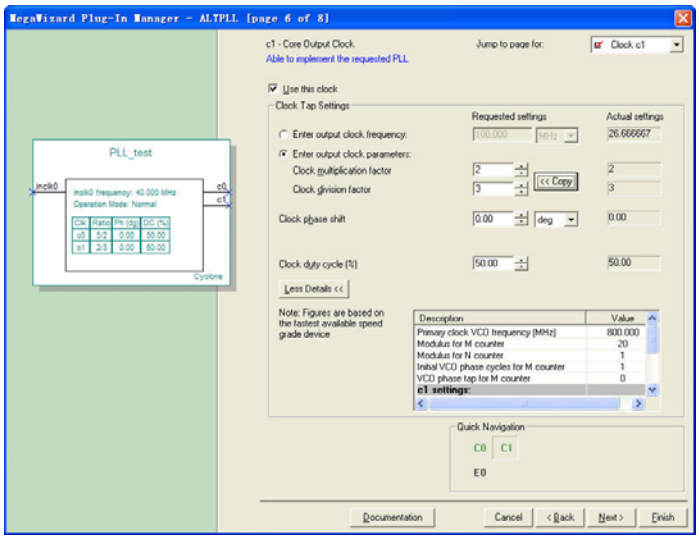


图 9.36　设置 c1 输出参数

（7）设置 e0 输出参数。

在宏模块向导管理器（PLL）的第 7 页，可以选择 PLL 模块 e0 输出参数，包括 Ratio 比值、相移量、占空比等，如图 9.37 所示。

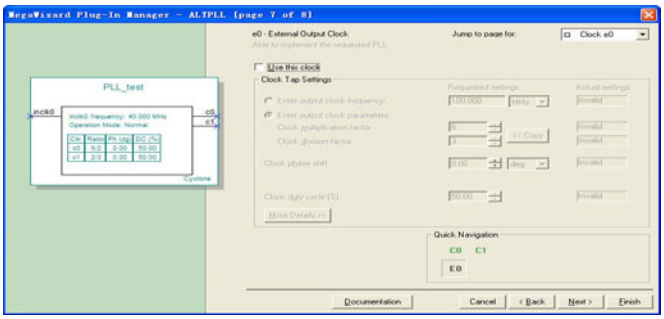


图 9.37　设置 e0 输出参数

（8）选择生成 PLL 模块文件。

在宏模块向导管理器（PLL）的第 8 页，完成 PLL 参数设置后，可以选择生成的 PLL 模块文件，共包含 6 个文件可以生成，如图 9.38 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

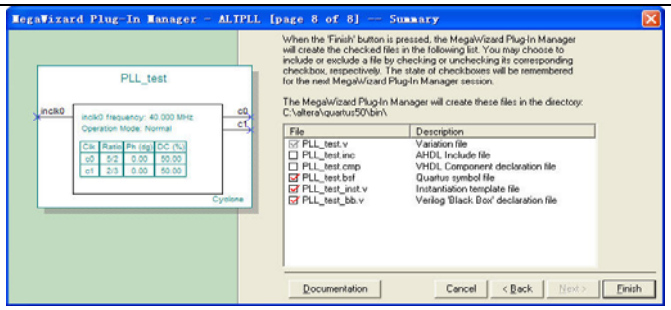


图 9.38　选择生成 PLL 模块文件

PLL 的调用方式与 FIFO 的类似，其构建方式及模块的声明结构可以参看 PLL 模块的 PLL_test.v 和 PLL_test_bb.v 文件。

9.3.3　高速串行收发器的使用方法

1．SERDES 简介

随着对信息流量需求的不断增长，传统并行接口技术成为进一步提高数据传输速率的瓶颈。过去主要用于光纤通信的串行通信技术——SERDES 正在取代传统并行总线而成为高速接口技术的主流。

SERDES 是英文 SERializer（串行器）/DESerializer（解串器）的简称。它是一种时分多路复用（TDM）、点对点的通信技术，即在发送端，多路低速并行信号被转换成高速串行信号，经过传输媒体（光缆或铜线），最后在接收端高速串行信号重新转换成低速并行信号。这种点对点的串行通信技术充分利用传输媒体的信道容量，减少所需的传输信道和器件引脚数目，从而大大降低通信成本。

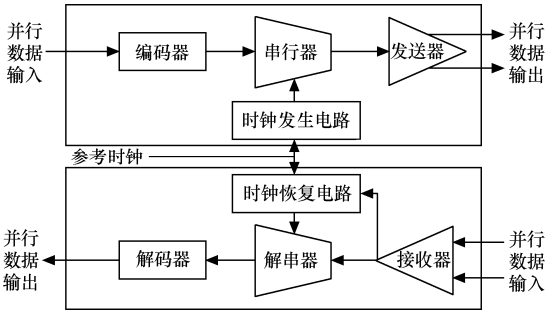
随着半导体技术的迅速发展，计算机的性能和应用取得了长足进步。可是，传统并行总线技术 PCI 却跟不上处理器和存储器的进步，而成为提高数据传输速率的瓶颈。新一代 PCI 标准 PCI Express 正是为解决计算机 I/O 瓶颈而提出的。

PCI Express 是一种基于 SERDES 的串行双向通信技术，每个通道的数据传输速率为 2.5Gbit/s，可多达 32 通道，支持芯片与芯片和背板与背板之间的通信。国际互联网络和信息技术的兴起促成了计算机和通信技术的交汇，而 SERDES 串行通信技术逐步取代传统并行总线正是这一交汇的具体体现。

2．SERDES 系统结构

基于 SERDES 的高速串行接口采用以下措施突破了传统并行 I/O 接口的数据传输瓶颈：一是采用差分信号传输代替单端信号传输，从而增强了抗噪声、抗干扰能力；二是采用时钟和数据恢复技术代替同时传输数据和时钟，从而解决了限制数据传输速率的信号时钟偏移问题。

一个典型 SERDES 收发机由发送通道和接收通道组成，如图 9.39 所示。编码器、串行器、发送器以及时钟产生电路组成发送通道，解码器、解串器、接收器以及时钟恢复电路组成接收通道。顾名思义，编码器和解码器完成编码和解码功能，其中 8B/10B、64B/66B 和不规则编码（scrambling）是最常用的编码方案。



华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

图 9.39　典型 SERDES 收发机结构图

串行器和解串器负责从并行到串行和从串行到并行的转换。串行器需要时钟产生电路，时钟发生电路通常由锁相环（PLL）来实现。解串器需要时钟和数据恢复电路（CDR），时钟恢复电路通常也由锁相环来实现，但有多种实现形式如相位插值、过剩抽样等。

发送器和接收器完成差分信号的发送和接收，其中 LVDS 和 CML 是最常用的两种差分信号标准。另外还有一些辅助电路也是必不可少的，例如环路（loopback）测试、内置误码率测试等。

3．SERDES 实现方式

SERDES 在系统中的实现虽然概念上比较简单，但是硬件实现要求很多细节正确无误，如信号端接、参考时钟的生成、锁相环（PLL）的使用、背板信号完整性和位错误率的评估等。

下面以 Altera 的 Stratix GX FPGA 为例介绍其中的内嵌的数千兆位收发器功能块。

Stratix GX 器件将高速 3.125Gbit/s 收发器串行/解串行（SERDES）技术和业界最先进的 FPGA 架构相结合。在 FPGA 中嵌入的数千兆位收发器功能块，能够在需要灵活性、高性能和最先进功能的许多新系统中使用收发器。

下面是这个数千兆位收发器功能块的主要特点。

- 支持从 622MHz ~ 3.125GHz 的所有频率。
- 每个块有 4 个独立的 3.125Gbit/s 全双工通道，每个器件多达 20 个通道（5 个块）。
- 支持 3.1875Gbit/s 的 10Gbit 光纤通道。
- 集成时钟数据恢复（CDR）、模式检测、字对齐、8b/10b 编解码器和同步功能。
- 功耗很低，每 4 通道只有 450mW（包括数千兆收发器功能块的功耗）。
- 支持动态可编程：预加重、均衡和 I/O 缓冲上的差分输出电压。
- 支持 SerialLite 协议，这是一个精简的点对点协议。
- 对中等性能信号的差分片内匹配。
- 完全实现 10Gbit 以太网物理介质接入层（PMA）和物理编码子层（PCS）功能。
- 支持灵活的时钟拓扑，每个收发器模块中有一个发送器 PLL 和四个接收器 PLL。
- 采用 1.5V、0.13μm 全铜 CMOS 工艺技术制造，支持 1.5V PCML I/O 标准。
- 包括独立的发送器和接收器节电功能，在不工作过程中减小功耗。
- 内嵌自检（BIST）功能，包括嵌入伪随机二进制序列（PRBS）图案生成和验证。
- 有 4 个独立的环回路径用于系统验证。

可以看到，Stratix GX 的这个收发器功能非常强大，远远超出了图 9.39 所示的 SERDES 基本结构图的功能。图 9.40 是它的原理框图，需要说明的是，设计者在使用这个模块时并不一定完全实现其中的每一个模块，可以根据需要只实现其中的部分内容。

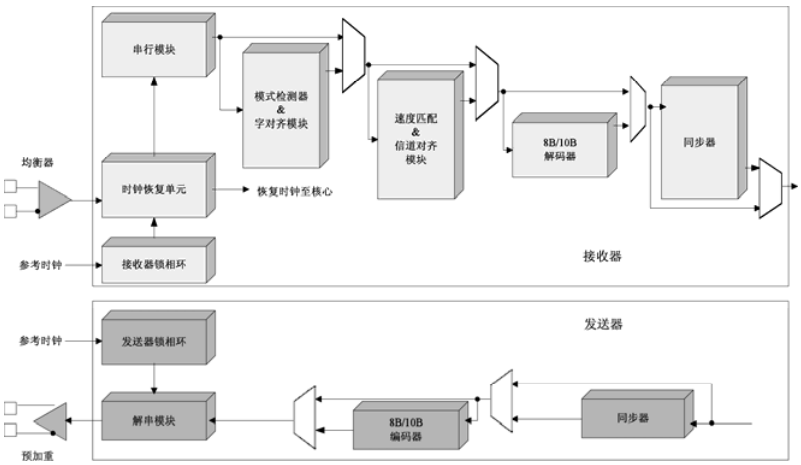


图 9.40　Stratix GX 收发机

开发板免费领取试用　开发板免费领取试用

下面介绍其中的一些模块。

（1）差分缓冲。

数千兆位收发器功能块差分 I/O 缓冲支持 1.5V PCML I/O 标准，有许多改善系统信号完整性的特性。例如，动态可重配置预加重和均衡功能，调整信号以补偿信号经过传送介质的衰减。不同的可编程 VOD 设置确保了驱动强度匹配传输线电阻和线长。另外，差分片内匹配为中等性能信号提供了合适的接收器和发送器缓冲匹配。

（2）可编程发送预加重模块。

发送预加重块使收发器（SERDES）可以驱动更长的背板或超过 1GHz 频率下的电缆。在这些频率下，通道损耗是很高的，衰减是很明显的，因为眼图的关闭无法让发送的信号继续传送。预加重信号提升信号的高频部分，补偿传输线的衰减。使用可编程的预加重设置，能够为给定的传输线选择最优的水平（或者在软件设定，或者通过内部或外部信号动态选择），让信号眼图在远端张得最开。

（3）可编程接收均衡器模块。

接收均衡器块使得收发器（SERDES）驱动更长的背板或超过 1GHz 的电缆。当信号经过接收器均衡器块时，同样可以提升信号的高频部分能量，补偿传输线的高频衰减指标。可编程均衡水平可以根据传输线进行优化（或者在软件设定或者通过内部或外部信号动态选择），让信号眼图在 CDR 单元输入端张得最开。

（4）发送器和接收器 PLL。

每个数千兆位收发器功能块有一个专用发送器 PLL 和四个专用接收器 PLL，提供灵活的时钟拓扑，支持一系列的输入数据流。对于输出传送和接收，这些 PLL 根据更低速的输入参考时钟生成所需的时钟频率。每个 PLL 支持 4、8、10、16 或 20 的倍增因子。每个外部参考时钟或 Stratix GX 内的各种时钟源都可以驱动 PLL。

（5）时钟恢复单元。

CDR 从输入串行数据流中提取时钟。恢复的时钟用于采样串行数据流、同步控制解串行器。

（6）串行/解串模块。

SERDES 模块将输入的高速串行数据转换为更低速度的并行接口，反之亦然。SERDES 模块可以配置为 8、10、16 或 20 位并行接口。

（7）模式检测器模块。

模式检测器模块识别输入数据流中的特殊模式。模式检测器包括一个内建的 8b/10b 的 K28 间隔符号检测和 SONET 的 A1A2 模式检测。在定制模式下，设计者可以创建专用模式。

（8）字对齐模块。

字对齐模块和模式检测器共同识别和调整正确的字节边界。此外，字对齐有一个定制模式，能够从 FPGA 内核逻辑人工地控制字节对齐。

（9）速率匹配器模块。

在串行数据传输中，发送和接收器件的时钟频率通常是不匹配的。这种不匹配会引起数据以略快或慢于接收器件能解释的速度传送。Stratix GX 速率匹配器从数据流中插入或删除如传送协议中定义的可移除的符号，解决了恢复时钟和 PLD 逻辑阵列时钟的频差，而不会丢失传送的数据。Stratix GX 数据匹配器为使用 8b/10b 编码数据的系统进行了优化。

（10）信道对齐器。

信道对齐器消除了实现 4 个收发器 XAUI 协议相关的信道至信道的偏移。信道对齐器消除 4 信道的偏移，为内核逻辑建立了可靠的以太网 XGMII 接口。

（11）8b/10b 编解码器。

8b/10b 编解码器模块将 8 比特的数据转换为 10 比特的数据，反之亦然。该算法平衡了串行数据流中“0”和“1”的数量，增加了变换密度，因此更易于接收器恢复串行数据。

（12）同步器。

同步器补偿了并行收发器接口和 FPGA 内核逻辑之间的相位差。

（13）内建自测。

BIST 为收发器提供了一组强大的诊断能力。它包括伪随机二进制序列（PRBS）和其他图案的生成器和检查器。BIST 也提供了 4 个环回配置用于系统诊断。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

联系方式

集团官网：www.hqyj.com　　嵌入式学院：www.embedu.org　　移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn　　物联网学院：www.topsight.cn　　研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路银海大厦 A 座 8 层，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218

华清远见

开发板免费领取试用　开发板免费领取试用

华清远见 FARSIGHT 始于2004 | 嵌入式培训专家

10年口碑积累，成功培养50000多名研发工程师，铸就专业品牌形象

华清远见的企业理念是不仅要做良心教育、做专业教育，更要做受人尊敬的职业教育。

《FPGA 应用开发入门与典型实例》（修订版）

作者：华清远见

专业始于专注 卓识源于远见

第 10 章 利用 FPGA 实现外设通信接口

本章目标

- 了解 FPGA 在外设通信接口实现方面的优势
- 掌握常用的外设接口在 FPGA 上的设计实现方法
- 掌握 FPGA 内部模块之间的接口设计方法

专业始于专注 卓识源于远见

开发板免费领取试用 开发板免费领取试用

10.1 FPGA 在外设接口实现方面的优势

10.1.1 充足的用户 I/O 资源

FPGA 的一个重要的应用领域就是数据采集和接口逻辑设计。随着芯片封装技术的提高，现在的 FPGA 已经可以在单位面积上提供更多的 I/O 管脚资源。

例如，Altera 公司低成本的 CycloneII 系列 FPGA 最大的封装为 FG896，可以给设计者提供最多 622 个 I/O 管脚。如表 10.1 所示为 CycloneII 系列 FPGA 的用户 I/O 资源。

表 10.1 CycloneII 系列 FPGA 的封装类型和最大的用户 I/O 数量

器件	144-Pin TQFP	208-Pin PQFP	256-Pin FineLine BGA	484-Pin FineLine BGA	672-Pin FineLine BGA	896-Pin FineLine BGA
EP2C5	89	142	158			
EP2C8	85	138	182			
EP2C20			152	315		
EP2C35				322	475	
EP2C50				294	450	
EP2C70					422	622

从表 10.1 中可以看出，FPGA 的管脚资源很丰富，可以满足大部分的应用需求。设计者可以根据需求，选择适合封装的芯片。因此，FPGA 比较适合接口比较多、需要大量 I/O 的应用场合。

10.1.2 灵活的可编程逻辑

标准的外设接口协议是开放的，因此不具备保密性和安全性。在某些应用场合，设计者需要在标准接口协议的基础上，重新设计接口协议来提高保密性或者其他方面的性能。FPGA 芯片灵活的可编程特性可以帮助设计者来实现这些自定义的协议。

10.1.3 支持多种电平接口标准

目前，新型的 FPGA 可以支持各种高级的 I/O 电平标准。以 Altera 公司的 CycloneII 系列 FPGA 为例，在高速差分电平标准方面，支持 LVDS、RSDS、mini-LVDS、LVPECL、差分 HSTL 以及差分 SSTL 等。在单端电平标准方面，支持 2.5V 和 1.8V 的 I 类和 II 类 SSTL 标准，1.8V 和 1.5V 的 I 类和 II 类 HSTL 以及 3.3V 的 PCI 和 PCI-X 1.0，还有 LVCMOS 和 LVTTL 等。

正是因为对多种 I/O 电平标准的支持，FPGA 可以轻松地实现标准的 PCI 接口，也可以支持高速的 DDR、DDR2 和 SDR SDRAM 以及 QDRII 的 SRAM。

10.2 利用 FPGA 实现 RS-232C 串行接口

10.2.1 RS-232C 接口概述

RS-232C 标准最初是为远程通信连接数据终端设备 DTE（Data Terminal Equipment）与数据通信设备 DCE（Data Communication Equipment）而制定的。因此这个标准的制定，并未考虑计算机系统的应用要求。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

目前，RS-232C 又广泛地应用于计算机与终端或外设之间的近端连接标准。显然，这个标准的有些规定和计算机系统是不一致的，甚至是相矛盾的。

RS-232C 标准中所提到的“发送”和“接收”，都是站在 DTE 立场上，而不是站在 DCE 的立场来定义的。由于在计算机系统中，往往是 CPU 和 I/O 设备之间传送信息，两者都是 DTE，因此双方都能发送和接收。

RS-232C 标准（协议）的全称是 EIA-RS-232C 标准，其中 EIA（Electronic Industry Association）代表美国电子工业协会，RS（Recommended standard）代表推荐标准，232 是标识号，C 代表 RS232 的最新一次修改（1969 年）。

10.2.2 RS-232C 接口的电气标准

RS-232C 采用的不是 TTL 电平的接口标准，而是负逻辑，即逻辑“1”为−3V～−15V，逻辑“0”为+3V～+15V。

RS-232C 对电器特性、逻辑电平和各种信号线功能都作了规定。

在 TxD 和 RxD 上：逻辑 1（MARK）为 3V～−15V，逻辑 0（SPACE）为+3～+15V。

在 RTS、CTS、DSR、DTR 和 DCD 等控制线上：

信号有效（接通，ON 状态，正电压）为+3V～+15V；

信号无效（断开，OFF 状态，负电压）为−3V～−15V。

RS-232C 是用正负电压来表示逻辑状态，与 TTL 以高低电平表示逻辑状态的规定不同。因此，为了能够同计算机接口或终端的 TTL 器件连接，必须在 RS-232C 与 TTL 电路之间进行电平和逻辑关系的变换。

10.2.3 RS-232C 的通信协议

所谓“串行通信”是指外设和计算机间使用一根数据信号线（另外需要地线，可能还需要控制线），数据在一根数据信号线上一位一位地进行传输，每一位数据都占据一个固定的时间长度，如图 10.1 所示。

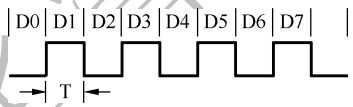


图 10.1 RS-232C 接口的数据通信方式

这种通信方式使用的数据线少，在远距离通信中可以节约通信成本，当然，其传输速度比并行传输慢。

由于 FPGA 与接口之间按并行方式传输，接口与外设之间按串行方式传输，因此，在串行接口中，必须要有“接收移位寄存器”（串→并）和“发送移位寄存器”（并→串）。典型的串行接口的结构如图 10.2 所示。

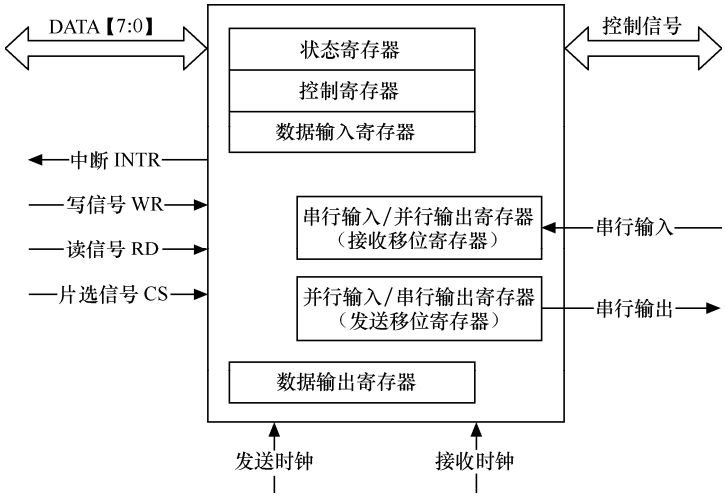


图 10.2 串行接口模块的结构示意图

在数据输入过程中，数据一位一位地从外设进入接口的“接收移位寄存器”，当“接收移位寄存器”中已

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

接收完 1 个字符的各位后，数据就从“接收移位寄存器”进入“数据输入寄存器”。

FPGA 从“数据输入寄存器”中读取接收到的字符，并行读取，即 D7～D0 同时被读至累加器中；“接收移位寄存器”的移位速度由“接收时钟”确定。

在数据输出过程中，FPGA 把要输出的字符（并行地）送入“数据输出寄存器”，“数据输出寄存器”的内容传输到“发送移位寄存器”，然后由“发送移位寄存器”移位，把数据一位一位地送到外设。“发送移位寄存器”的移位速度由“发送时钟”确定。

接口中的“控制寄存器”用来容纳 FPGA 送给此接口的各种控制信息，这些控制信息决定接口的工作方式。

能够完成上述串并转换功能的电路，通常称为“通用异步收发器”（UART：Universal Asynchronous Receiver and Transmitter），包括：双缓存发送数据寄存器、并行转串行装置、双缓存输入数据寄存器、串行转并行装置。

RS232 通信协议基本结构如图 10.3 所示，起始位低，停止位高。波特率范围是 300～115200 bit/s；8 位数据位；一位或两位停止位；奇校验、偶校验或无校验位。

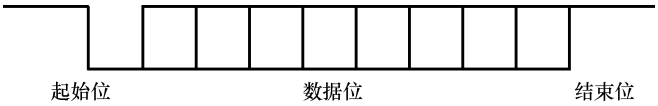


图 10.3　RS232 通信协议基本结构

10.2.4　RS-232C 接口的典型应用

RS-232C 不仅可以用来实现 FPGA 系统与 PC 之间的低速率数据传递，而且可以广泛应用到工业控制和自动化仪器仪表领域。RS-485 和 RS-422 等标准与 RS-232C 标准一样，协议部分也采用了 UART 协议，因此 FPGA 的实现逻辑是相同的，只是接口电平不同。

在 FPGA 上，可以实现多个 UART 单元。也就是说，单片 FPGA 可以支持多个 RS-232C 接口，比较适合需要多个串口的场合。传统的单片机（MCU）或者 ARM 处理器一般只能支持 2～3 个 UART 接口，在这样场合就需要外围芯片来扩展 UART。

10.3　利用 FPGA 实现 USB 2.0 通信接口

10.3.1　USB 2.0 接口的实现方式

利用 FPGA 来实现 USB 2.0 接口的方式一般有两种，一是借助外围的 USB 接口芯片，二是 FPGA 内部实现 USB 协议控制器，外部通过 USB 的 PHY 芯片来实现接口。

对于第一种方式，FPGA 不需要实现 USB 通信协议，开发的风险小，而且性能稳定，因此被大部分设计者所采用。而第二种方式需要购买 USB 的 IP 核，门槛较高。如果自己开发基于 FPGA 的 USB 协议栈，风险高，而且兼容性和稳定性很难保证。

目前市场上的 USB 2.0 接口芯片厂家和型号比较多，其中 Cypress Semiconductor 公司的 EZ-USB FX2（CY7C68013）是世界上第一款集成 USB 2.0 的微处理器，它集成了 USB 2.0 收发器、SIE（串行接口引擎）、增强的 8051 微控制器和可编程的外围接口。

FX2 这种独创性结构可使数据传输率达到 56MByte/s，即 USB 2.0 允许的最大带宽。在 FX2 中，智能 SIE 可以硬件处理许多 USB 1.1 和 USB 2.0 协议，从而减少了开发时间、确保了 USB 的兼容性。GPIF（General Programmable Interface）和主/从端点 FIFO（8 位或 16 位数据总线）为 ATA、UTOPIA、EPP、PCMCIA 和 DSP 等提供了简单和无缝连接接口。

CY7C68013 结构图如图 10.4 所示。FPGA 与 CY7C68013 的电路连接关系可以参考本书 2.2.11 小节的内容。

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

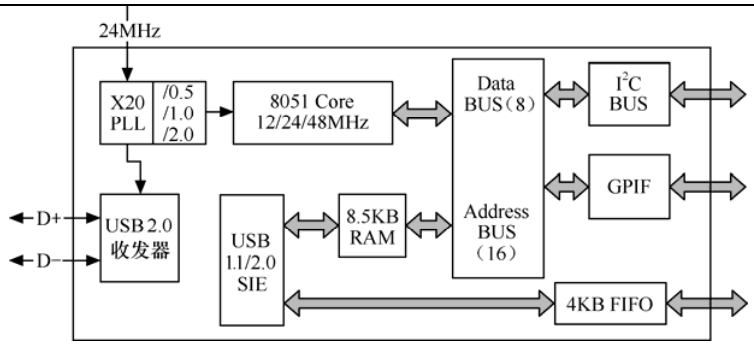


图 10.4　CY7C68013 FX2 结构示意图

10.3.2　FX2 接口芯片的 Slave FIFO 传输模式

当 FPGA 与 FX2 芯片相连时，只需要利用 FX2 作为一个 USB 2.0 数据通道来实现与主机的高速通信。FPGA 能够提供满足 Slave FIFO 要求的传输时序，可以作为 Slave FIFO 主控制器。`

Slave FIFO 传输的示意图如图 10.5 所示。

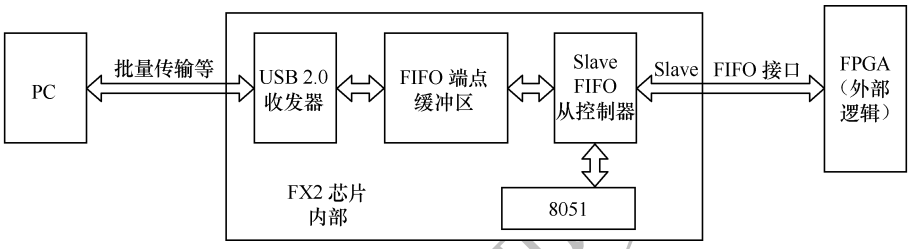


图 10.5　Slave FIFO 传输的示意图

在这种方式下，FX2 内嵌的 8051 固件的功能只是配置 Slave FIFO 相关的寄存器以及控制 FX2 何时工作在 Slave FIFO 模式下。一旦 8051 固件将相关的寄存器配置完毕，且使自身工作在 Slave FIFO 模式下后，FPGA 就可按照 Slave FIFO 的传输时序，高速与主机进行通信，而在通信过程中不需要 8051 固件的参与。

在 Slave FIFO 方式下，FPGA 与 FX2 的连接信号图如图 10.6 所示。

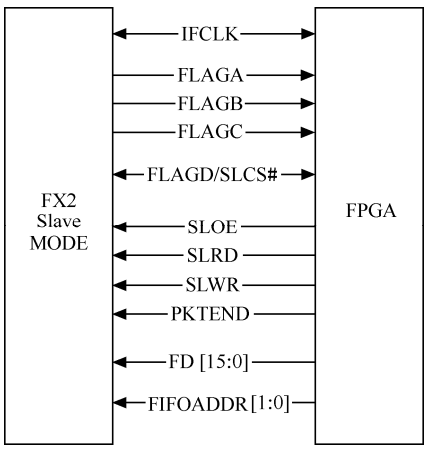


图 10.6　FPGA 与 FX2 连接的信号示意图

下面是信号线的说明。

IFCLK：FX2 输出的时钟，可作为通信的同步时钟。

FLAGA、FLAGB、FLAGC、FLAGD：FX2 输出的 FIFO 状态信息，如满、空等。

SLCS：FIFO 的片选信号，外部逻辑控制，当 SLCS 输出高时，不可进行数据传输。

SLOE：FIFO 输出使能，外部逻辑控制，当 SLOE 无效时，数据线不输出有效数据。

SLRD：FIFO 读信号，外部逻辑控制，同步读时，FIFO 指针在 SLRD 有效时的每个 IFCLK 的上升沿递增，异步读时，FIFO 读指针在 SLRD 的每个有效至无效的跳变沿时递增。

SLWR：FIFO 写信号，外部逻辑控制，同步写时，在 SLWR 有效时的每个 IFCLK 的上升沿时数据被写入，FIFO 指针递增，异步写时，在 SLWR 的每个有效至无效的跳变沿时数据被写入，FIFO 写指针递增。

PKTEND：包结束信号，外部逻辑控制，在正常情况下，外部逻辑向 FX2 的 FIFO 中写数，当写入 FIFO 端点的字节数等于 FX2 固件设定的包大小时，数据将自动被打成一包进行传输，但有时外部逻辑可能需要传输一个字节数小于 FX2 固件设定的包大小的包，这时，它只需在写入一定数目的字节后，声明此信号，此时 FX2 硬件不管外部逻辑写入了多少字节，都自动将之打成一包进行传输。

FD[15:0]：数据线。

FIFOADR[1:0]：选择 4 个 FIFO 端点的地址线，外部逻辑控制。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

10.3.3 Slave FIFO 模式的典型操作时序

1．同步 Slave FIFO 写操作

同步 Slave FIFO 写的标准连接图如图 10.7 所示。
同步 Slave FIFO 写的标准时序如下。

- IDLE：当写事件发生时，进状态 1。
- 状态 1：使 FIFOADR[1:0]指向 IN FIFO，进状态 2。
- 状态 2：如 FIFO 满，在当前状态等待，否则进状态 3。
- 状态 3：驱动数据到数据线上，使 SLWR 有效，持续一个 IFCLK 周期，进状态 4。
- 状态 4：如需传输更多的数，进状态 2，否则进状态 IDLE。

状态跳转示意图如图 10.8 所示。

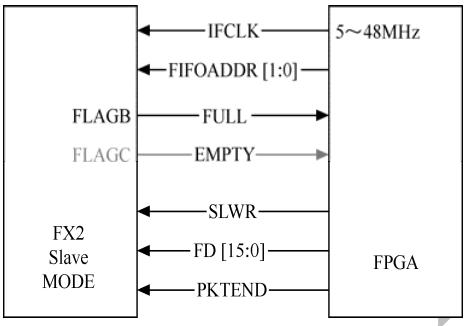


图 10.7　同步 Slave FIFO 写的标准连接图

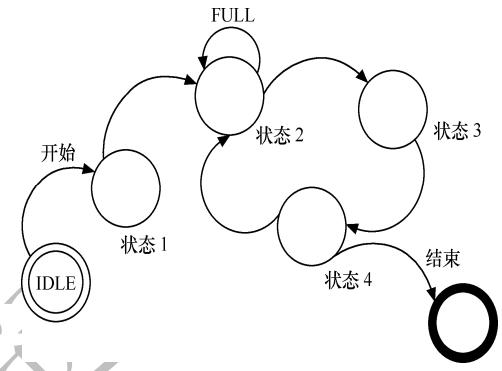


图 10.8　同步 Slave FIFO 写的标准时序

假定 FX2 设定包大小为 512 字节，当 FPGA 向 FIFO 端点中写入的数据达 512 字节时，FX2 硬件自动将已写入的 512 字节打成一包准备进行传输。这个动作就和在普通传输中，FX2 固件向 FIFO 端点中写入 512 字节后，把 512 这个数写入 EPxBC 中一样，只不过这个过程是由硬件自动完成的。在这里可以看出“FX2 固件不参与数据传输过程”的含义了。外部逻辑只需按上面的时序图所示的时序向 FIFO 端点中一个一个字节（或字）地写数，写到一定数量，FX2 硬件自动将数据打包传输，这一切均不需固件的参与，由此实现高速数据传输。

2．同步 Slave FIFO 读操作

同步 Slave FIFO 读的标准连接图如图 10.9 所示。
同步 Slave FIFO 读的标准时序如下。

- IDLE：当读事件发生时，进状态 1。
- 状态 1：使 FIFOADR[1:0]指向 OUT FIFO，进状态 2。
- 状态 2：使 SLOE 有效，如 FIFO 空，在当前状态等待，否则进状态 3，
- 状态 3: 从数据线上读数，使 SLRD 有效，持续一个 IFCLK 周期，以递增 FIFO 读指针，进状态 4。
- 状态 4：如需传输更多的数，进状态 2，否则进状态 IDLE。

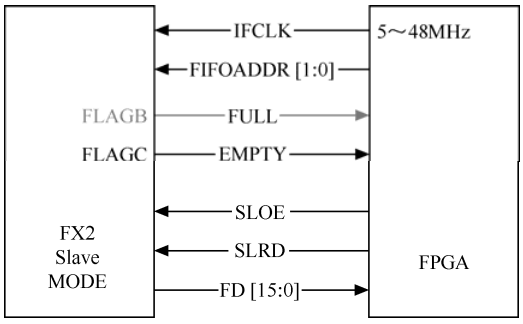


图 10.9　同步 Slave FIFO 读的标准连接图

状态跳转示意图如图 10.10 所示。

开发板免费领取试用　开发板免费领取试用

3．异步 Slave FIFO 写操作

异步 Slave FIFO 写的标准连接图如图 10.11 所示。

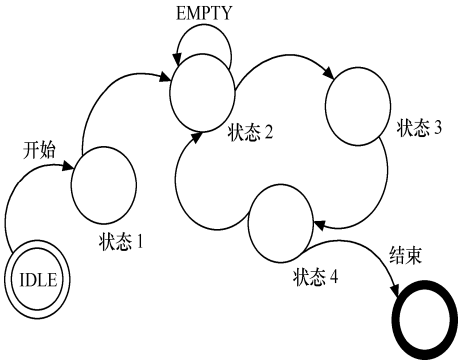


图 10.10　同步 Slave FIFO 读状态跳转示意图

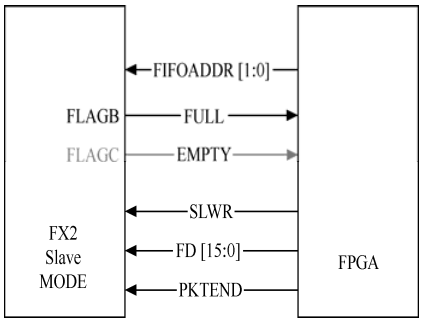


图 10.11　异步 Slave FIFO 写的标准连接图

异步 Slave FIFO 写的标准时序如下。

- IDLE：当写事件发生时，进状态 1。
- 状态 1：使 FIFOADR[1:0]指向 IN FIFO，进状态 2。
- 状态 2：如 FIFO 满，在当前状态等待，否则进状态 3。
- 状态 3：驱动数据到数据线上，使 SLWR 有效，再无效，以使 FIFO 写指针递增，进状态 4。
- 状态 4：如需传输更多的数，进状态 2，否则进状态 IDLE。

状态跳转示意图如图 10.12 所示。

4．异步 Slave FIFO 读操作

异步 Slave FIFO 读的标准连接如图 10.13 所示。

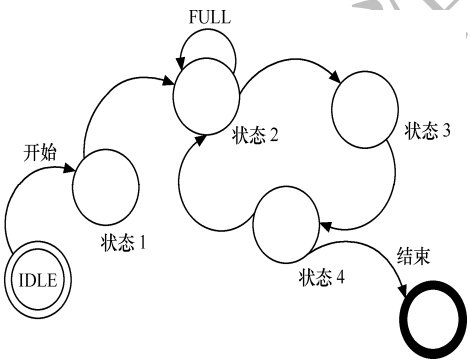


图 10.12　异步 Slave FIFO 写状态跳转示意图

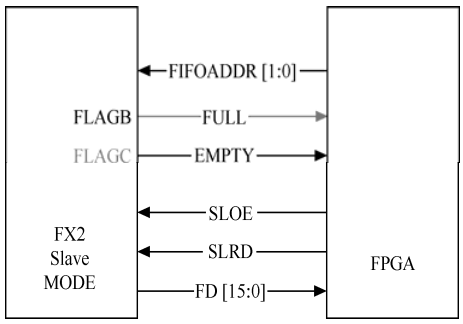


图 10.13　异步 Slave FIFO 读的标准连接图

异步 Slave FIFO 读的标准时序如下。

- IDLE：当读事件发生时，进状态 1。
- 状态 1：使 FIFOADR[1:0]指向 OUT FIFO，进状态 2。
- 状态 2：如 FIFO 空，在当前状态等待，否则进状态 3。
- 状态 3：使 SLOE 有效，使 SLRD 有效，从数据线上读数，再使 SLRD 无效，以递增 FIFO 读指针，再使 SLOE 无效，进状态 4。
- 状态 4：如需传输更多的数，进状态 2，否则进状态 IDLE。

状态跳转示意图如图 10.14 所示。

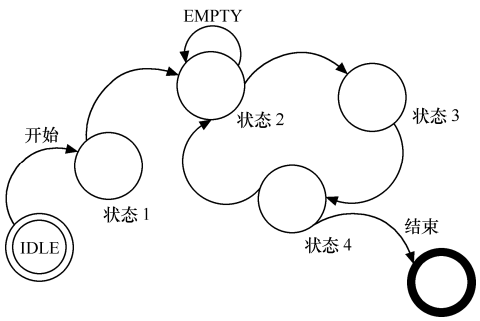


图 10.14　异步 Slave FIFO 读状态跳转示意图

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

10.3.4 FX2 的固件程序设计

在 FX2 芯片的固件程序设计中，最关键的就是系统初始化函数 TD_Init（void），下面讲解这个函数的部分代码。

```
//时钟设置
CPUCS = 0x12;        // 48MHz CLKOUT 输出使能
IFCONFIG =0x43;      // 使用外部时钟，IFCLK 输入不反向

SYNCDELAY;           // 同步延迟
EP2CFG=0xA0;         // 需要设定为 4 缓冲，每个缓冲区大小为 512 字节
SYNCDELAY;
EP4CFG=0x00;
SYNCDELAY;
EP6CFG=0xE0;
SYNCDELAY;
EP8CFG=0x00;

SYNCDELAY;
FIFORESET = 0x80;   // 激活 NAK-ALL 避免竞争
SYNCDELAY;
FIFORESET = 0x02;   // 复位 FIFO 2
SYNCDELAY;
FIFORESET = 0x06;   // 复位 FIFO 6
SYNCDELAY;
FIFORESET = 0x00;   // 取消激活 NAK-ALL

SYNCDELAY;
PINFLAGSAB = 0xE6;  // FLAGA 设定为 EP6PF, FLAGB 设定为 EP6FF
SYNCDELAY;
PINFLAGSCD = 0xf8;  // FLAGC 设定为 EP2EF, FLAGD 保留
SYNCDELAY;
PORTACFG |=  0x00;  // SLCS 有效
SYNCDELAY;
FIFOPINPOLAR = 0x00; // 所有信号低电平有效
SYNCDELAY;
```

10.3.5 USB 2.0 接口的典型应用

由于 USB 2.0 接口的数据传输速度快，而且使用方便，因此被广泛应用于各种便携设备中。例如，基于 USB 2.0 接口的数据采集终端，可以将各种类型的数据采集到电脑主机中进行处理和结果显示，其典型的结构如图 10.15 所示。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

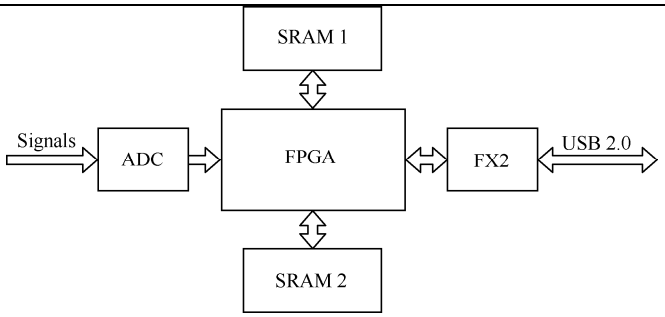


图 10.15　基于 USB 2.0 接口的数据采集终端结构框图

10.4　利用 FPGA 实现常用显示接口（Display Interface）

10.4.1　七段数码显示接口的设计与实现

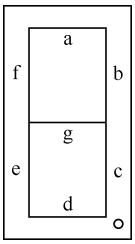


图 10.16　七段数码管的标准外观图

七段数码管因为价格低廉，使用简单，经常被用来实现一些简单的状态显示功能。七段数码管的标准外观图如图 10.16 所示。右下角的圆点用 Dp 来表示，用来实现小数点的显示。

七段数码管经常用来显示十进制或十六进制的数，所以在数据显示之前，首先要进行二进制到十进制或者十六进制的转换。将它们转换成十进制的或是十六进制的数。

七段数码管分为共阴和共阳两种类型。比如，红色飓风开发板使用的是共阴类型的数码管，信号是高电平有效。十六进制数据在显示的时候对应的各段输出对应关系如表 10.2 所示。

表 10.2　　十六进制数据与输出的映射关系

数　据	a	b	c	d	e	f	g	Dp
0	1	1	1	1	1	1	0	0
1	0	1	1	0	0	0	0	0
2	1	1	0	1	1	0	1	0
3	1	1	1	1	0	0	1	0
4	0	1	1	0	0	1	1	0
5	1	0	1	1	0	1	1	0
6	1	0	1	1	1	1	1	0
7	1	1	1	0	0	0	0	0
8	1	1	1	1	1	1	1	0
9	1	1	1	1	0	1	1	0
A	1	1	1	0	1	1	1	0
B	0	0	1	1	1	1	1	0
C	1	0	0	1	1	1	0	0
D	0	1	1	1	1	0	1	0
E	1	0	0	1	1	1	1	0
F	1	0	0	0	1	1	1	0

华清远见

开发板免费领取试用　开发板免费领取试用

红色飓风开发板上采用的是 4 位并联的数码管。4 个数码管共用 8 条数据线，控制线是独立的。如果 4 个数码管要显示不同的数值，比如显示 1234，那么就需要采用一种控制策略。最常用的方法就是动态扫描显示。

动态扫描显示利用了时分的原理和人的视觉暂留效应。例如，一个 4 位动态扫描数码显示器的显示周期可划分为 4 个阶段。

阶段 1→阶段 2→阶段 3→阶段 4

每个周期只选通一位数据。在周期 1 显示第一个数码，周期 2 显示第二个数码。在扫描 4 个阶段后，又重新按顺序循环。如果扫描的速度足够快，从视觉上看就好像 4 个数码管同时显示不同的内容。

4 位扫描数码管共有 4 组 BCD 码（4 位）输入线，8 根 8 段译码输出线和 4 根位选通线。扫描工作中，先从 4 组 BCD 数据中选出一组，通过 BCD/7 段译码器译码后输出。与此同时，3/8 译码器产生位选通信号，则在此瞬间，数码管应该为要显示的数值。然后再选出下一组数据译码后输出，位选通信号则相应下移一位，将下一数码管选通输出，如图 10.17 所示。

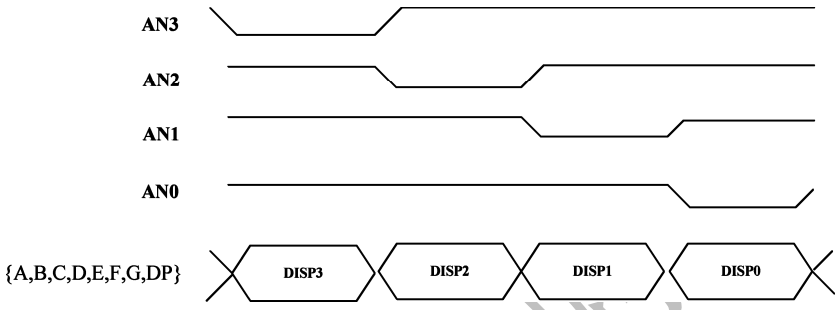


图 10.17　动态扫描原理示意图

10.4.2　字符型 LCD 显示接口的设计与实现

液晶屏（LCD）通常分为点阵型和字符型两种。字符型的液晶屏相对于数码管来说，可以显示更多的内容和字符，人机界面更为友好，而且操作简单，因此得到了广泛的应用。不同厂家的字符型 LCD 虽然型号不同，但是操作方法基本是一致的。

字符型 LCD 一般会根据显示字符的数量来确定型号，如 1602 表示这个液晶可以显示 2 行字符，每行为 16 个。红色飓风开发板上采用的字符型 LCD 的型号就是 1602。下面就以 1602 为例来介绍字符型 LCD 显示接口的设计方法。

1602 型液晶模块采用 14 针标准接口，各个管脚的定义如表 10.3 所示。

表 10.3　　1602 型液晶模块的管脚配置表

管　脚	符　号	说　明
1	VSS	器件地
2	VDD	+5V 电源
3	V0	对比度调整端，接正电源时对比度最弱，接地电源时对比度最高，对比度过高时会产生“鬼影”，使用时可以通过一个 10kΩ的电位器调整对比度
4	RS	寄存器选择，高电平时选择数据寄存器，低电平时选择指令寄存器
5	RW	读写信号线，高电平时进行读操作，低电平时进行写操作。当 RS 和 RW 共同为低电平时可以写入指令或者显示地址，当 RS 为低电平、RW 为高电平时可以读忙信号，当 RS 为高电平、RW 为低电平时可以写入数据
6	E	使能端，当 E 端由高电平跳变成低电平时，液晶模块执行命令
7～14	D0～D7	8 位双向数据线

1602 液晶模块内部的字符发生存储器（CGROM）已经存储了 160 个不同的点阵字符图形。这些字符包括：阿拉伯数字、英文字母的大小写、常用的符号和日文假名等。每一个字符都有一个固定的代码，比如大写的英文字母“A”的代码是 0100_0001B（41H）。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

显示的时候，模块把地址 41H 中的点阵字符图形显示出来，我们就能看到字母“A”。在编程实时，只需要输入相应字符的地址，液晶屏就会输出相应的字符。

FPGA 对液晶模块的写操作、屏幕和光标的操作都是通过指令编程来实现的。1602 型液晶的操作指令表如表 10.4 所示。

表 10.4　　1602 型液晶模块的指令表

序　号	指　令	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0
1	清显示	0	0	0	0	0	0	0	0	0	1
2	光标返回	0	0	0	0	0	0	0	0	1	*
3	光标或显示模式	0	0	0	0	0	0	0	1	I/D	S
4	显示开/关控制	0	0	0	0	0	0	1	D	C	B
5	光标或字符移位	0	0	0	0	0	1	S/C	R/L	*	*
6	功能设置命令	0	0	0	0	0	DL	N	F	*	*
7	字符发生器地址设置	0	0	0	1	字符发生器地址（AGG）					
8	DDRAM 地址设置	0	0	1	显示数据存储器 DDRAM 的地址（ADD）						
9	读忙标志或地址	0	1	BF	计数器地址（AC）						
10	写数据到 RAM	1	0	要写的数据							
11	从 RAM 读数据	1	1	读出的数据							

各条指令的说明如下（表 10.4 中，设 1 为高电平、0 为低电平）。

- 指令 1：清显示，指令码 01H，光标复位到地址 00H 位置。
- 指令 2：光标返回，光标返回到地址 00H。
- 指令 3：光标和显示模式设置。其中 I/D 表示光标移动方向，高电平右移，低电平左移；S 表示屏幕上所有文字是否左移或者右移，高电平表示有效，低电平则无效。
- 指令 4：显示开/关控制。其中 D 表示控制整体显示的开与关，高电平表示开显示，低电平表示关显示；C 表示控制光标的开与关，高电平表示有光标，低电平表示无光标；B 表示控制光标是否闪烁，高电平闪烁，低电平不闪烁。
- 指令 5：光标或显示移位。其中 S/C 表示高电平时移动显示的文字，低电平时移动光标；R/L 表示文字或光标的移动方向，高电平为右移，低电平为左移。
- 指令 6：功能设置命令。其中 DL 表示高电平时为 4 位总线，低电平时为 8 位总线；N 表示低电平时为单行显示，高电平时双行显示；F 表示低电平时显示 5×7 的点阵字符，高电平时显示 5×10 的点阵字符。
- 指令 7：字符发生器 RAM 地址设置。
- 指令 8：DDRAM 地址设置。
- 指令 9：读忙信号和光标地址。其中 BF 表示忙标志位，高电平表示忙，此时模块不能接收命令或者数据，如果为低电平表示不忙。
- 指令 10：写数据。
- 指令 11：读数据。

10.4.3　VGA 显示接口的设计与实现

由于 VGA 接口是模拟信号，而 FPGA 只能支持数字信号，因此需要专用的 D/A 转换芯片来实现模拟到数字的转换。常用的 DAC 芯片有 ADI 公司的 ADV7123 或者 ADV7125 等。

在某些应用场合下，VGA 显示器仅仅是用来显示文字符号或者简单图形，不需要丰富的色彩和细腻的画面。这时候，VGA 接口就可以适当地简化。

标准的 VGA 接口连接器为 DB15，在 15 个管脚里面有 5 个是最重要的。它们分别是红、绿，蓝三基色信

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

号和水平、垂直两个同步控制信号，如图 10.18 所示，其中，VGA_R、VGA_G、VGA_B、VGA_HS、VGA_VS 是直接连接到 FPGA 的管脚。

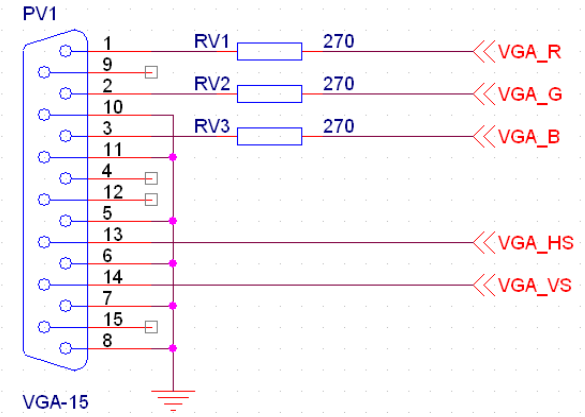


图 10.18　简化的 VGA 接口电路连接

FPGA 的管脚只有高电平和低电平两种状态，因此对于每个色彩分量信号也仅有两种状态。这样 3 个色彩分量就可以组合出 8 种颜色，如表 10.5 所示。

表 10.5　　简化的 VGA 接口色彩对照表

VGA_R	VGA_G	VGA_B	对应的显示颜色
0	0	0	黑色
0	0	1	绿色
0	1	0	蓝色
0	1	1	蓝绿色
1	0	0	红色
1	0	1	品红色
1	1	0	黄色
1	1	1	白色

VGA 信号如果要正常显示，那么输出信号必须满足一定的时序关系。如图 10.19 所示为 VGA 接口电路的通用时序关系。不同的分辨率和刷新率有不同的时序如表 10.6 所示。

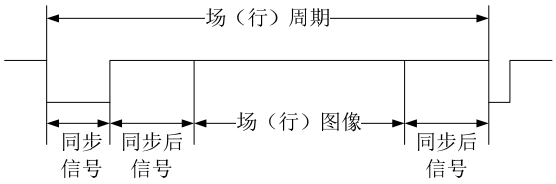


图 10.19　VGA 接口电路的通用时序关系

表 10.6　　常见分辨率的 VGA 时序参数表

显示模式	像素时钟/MHz	水平方向（以像素计算）				垂直方向（以行计算）			
		有效视频信号	同步前	同步信号	同步后	有效视频信号	同步前	同步信号	同步后
640×480 60Hz	25.175	640	16	96	48	480	11	2	31
800×600 60Hz	40.000	800	40	128	88	600	1	4	23
1024×768 60Hz	65.000	1024	24	136	160	768	3	6	29

10.5　利用 FPGA 实现 A/D、D/A 转换器接口

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

10.5.1 A/D、D/A 转换器接口的特点

A/D、D/A 转换器是 FPGA 系统设计中的常用器件，经常用来实现模拟信号和数字信号的相互转换。根据应用场合的不同，A/D、D/A 转换芯片的性能指标参数差别比较大，因此接口格式也无法统一。

10.5.2 A/D、D/A 转换器接口的实现方法

1．标准接口的 A/D、D/A 转换器与 FPGA 的接口设计

在一些典型的应用领域，需要处理的信号单一，芯片厂家设计的芯片一般会遵循一些行业标准接口。例如音频信号处理领域，很多厂家的音频 A/D、D/A 转换芯片都支持 SPI 接口。
FPGA 实现这些标准的接口，如 SPI 接口，可以通过网络或者 FPGA 厂家获得一些参考代码。FPGA 芯片的生产厂家会在其开发工具软件里面集成一些常用接口的 IP 核，设计者可以免费使用，降低了开发难度。另外，有一些开源的网站，如 http://www.opencores.org 等，也提供了免费的接口 IP 核的源代码下载服务。设计者可以下载代码后，经过仿真验证或者适当修改后，应用到自己的系统中去。

2．特殊接口的 A/D、D/A 转换器与 FPGA 的接口设计

还有很多信号处理领域，如通信系统设计领域，需要处理的信号很复杂。这种情况下，A/D、D/A 的芯片厂家一般会根据处理的精度和采样的速度要求来设计接口时序。
设计者要根据厂家提供的芯片手册给出的时序来设计接口。
有些高速的 A/D 或者 D/A 芯片会采用 LVDS 差分信号电平标准。这样的情况下， FPGA 的管脚也要采用对应的 LVDS 模式。
需要注意的是，FPGA 上每一组 LVDS 管脚是成对出现的，不是任意分组的。而且，如果一个 BANK 被设定为 LVDS，那么这个 BANK 的所有管脚都是 LVDS 模式的。

10.5.3 AD/DA 转换器接口的注意事项

1．选择正确的时钟采样边沿

对于一个边沿采样的数据采集模块，经常需要确定采用上升沿还是下降沿来采样数据。那么，选择的依据就是保证数据采样的稳定、可靠。
为了保证同步，FPGA 的采样时钟一般采用 A/D 转换芯片的输出时钟，这个时钟和数据是同步的。这个时钟和数据的关系与 A/D 转换芯片的具体型号有关。例如，在如图 10.20 所示的时序图中，每当时钟的上升沿数据产生变化。那么这样的情况下，在时钟的下降沿的时候，数据是最稳定的，适合采样。

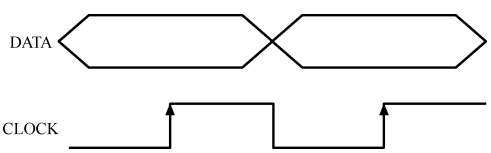


图 10.20 A/D 芯片的部分输出信号时序关系

2．选择适合的同步头检测方法

A/D 芯片的输出数据一般都是打包数据，以数据帧的格式输出。数据帧的同步格式分为两种：外同步模式和内同步模式。

开发板免费领取试用 开发板免费领取试用

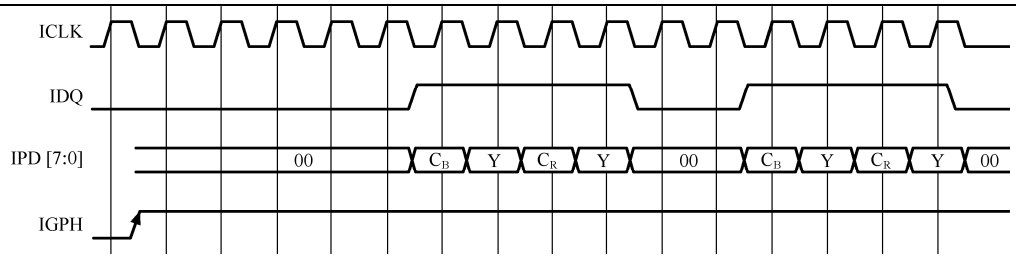


图 10.21　A/D 芯片的部分输出信号时序关系

外同步模式：是指除了时钟和数据信号外，还有同步指示信号。同步指示信号可以是脉冲信号，指示一帧有效数据的开始；也可以是电平信号，指示一帧有效数据的持续时间。

内同步模式：是指输出信号中只有时钟和数据信号，同步头隐藏在数据之中，如表 10.7 所示是 ITU656 标准的数字视频信号。

表 10.7　　ITU656 标准的数字视频信号

帧　结　构									
Blanking Period	…			80			10		
Timing Reference Code	FF			00		00		SAV	
720 Pixels YUV 4:2:2 Data	Cb0	Y0	Cr0	Y1	Cb2	Y2	…	Cr718	Y719
Timing Reference Code	FF			00		00		EAV	
Blanking Period	80			10			…		

其中，“FF 00 00 EAV”和“FF 00 00 SAV”就是帧同步头，其作用也是用来指示有效数据开始和结束。其格式为：“FF 00 00 XY”，XY 的定义如下。

SAV_EVEN=8'H80

EAV_EVEN=8'H9D

SAV_ODD =8'HC7

EAV_ODD =8'HDA

3．选择有效的缓存策略

在数据采集系统中，缓存结构的设计直接影响系统的性能。对于简单的设计，可以采用一个 FIFO 或者双口 RAM 来实现缓存。如果单缓存不能满足系统的性能要求，可以采用乒乓结构的缓存。

在有些特殊情况下，采集模块后面数据处理模块的处理时间不固定，和数据内容相关。这时候采用乒乓结构的缓存策略就不能满足要求了，因为乒乓结构的切换时间一般是固定的，而且和数据内容是无关的。还有一种比乒乓结构更复杂的缓存管理策略，那就是三缓存环形结构。如图 10.22 所示，这样的结构虽然比乒乓结构提供了更好的缓存性能，但是管理的复杂度大大提高了，系统的延迟也增加了。

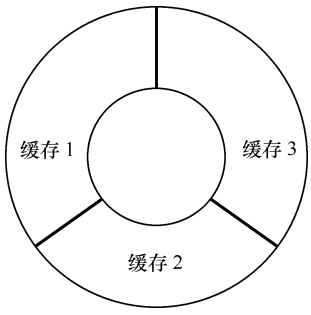


图 10.22　三缓存的环形缓存区示意图

无论是单缓存，双缓存还是三缓存结构，只要能满足系统的实时采集要求就是有效的缓存策略。

10.6　典型实例 16：RS-232C（UART）接口的设计与实现

10.6.1　实例内容及目标

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

1．实例的主要内容

本节旨在通过分析 UART 控制器，设计实现了 FPGA 通过 RS-232C 接口与 PC 机的通信。设计过程中用 Modelsim 对 UART 控制器进行仿真，帮助读者进一步了解 UART 协议的具体时序。

2．实例目标

通过本实例，读者应达到以下目标。

- 了解 UART 工作原理和时序。
- 熟悉 Modelsim 仿真的流程。
- 熟练掌握状态机的使用。
- 实现 FPGA 与 PC 机之间的 RS-232C 接口通信。

10.6.2 实例详解

系统上电复位后，先将 20 个数写入 FPGA 内部资源 BlockRam（可以用 IP 核来实现），然后根据设定的协议（该协议可自己设定）来建立 PC 机与 FPGA 的通信，此处设定的协议如下。
PC 机要从串口读取数据时，先从串口发送 E4H、00H。E4H 代表 FPGA 的地址（可以随意设定一个数），00H 表示要读取数据。FPGA 收到这两个数据后，通过串口给 PC 机反馈信号。发送 E4H、E1H，表示 FPGA 已经收到了 PC 机的命令，要开始向 PC 机发送数据。PC 端准备接收数据，建立了此连接后，PC 机可从 FPGA 读出最初写入 BlockRam 中的值了。

10.6.3 BlockRAM 的实现方法

对于 BlockRam，可以直接应用 IP 核来实现，具体实现步骤如下。
首先为工程添加新的设计文件，选择“New Source…”，如图 10.23 所示。

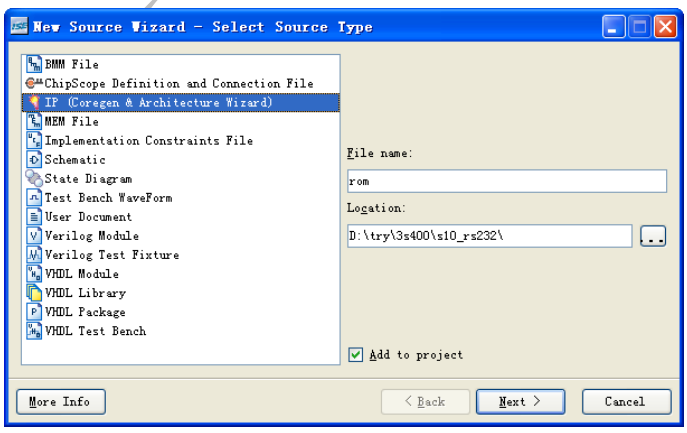


图 10.23　新建 IP 核设计文件

在弹出的对话框里面选择创建 IP（CoreGen & Architecture Wizard）类型文件，并为该文件设置文件名及路径，单击“Next”按钮，如图 10.24 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

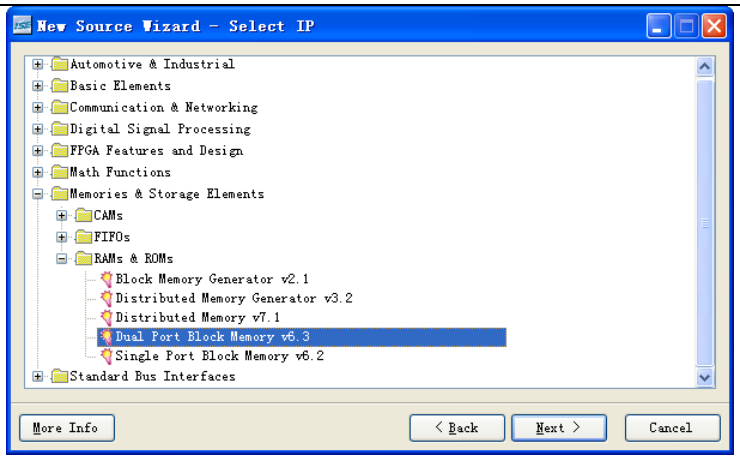


图 10.24　选择 IP 核类型

在选择 IP 核类型对话框中选择“Dual Port Block Memory 6-1”，即可生成一个双口 RAM。单击“Next”按钮进入双口 RAM 设置向导，如图 10.25 所示。

在该向导中，要对该双口 RAM 的属性进行设置。

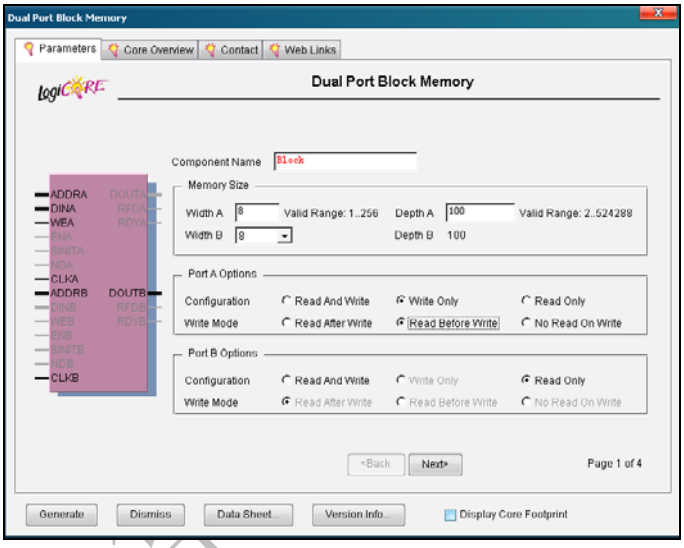


图 10.25　双口 Block Memory 设置向导

如图 10.25 所示，WidthA、WidthB 分别为写入两个口的数据的位数，Depth 为容量，在这里设为 100，实际中只用到 20 个。对 A 口，上电复位后，就将 20 个数据写入 A 口，因此 A 口设为 Write Only，对于 B 口，FPGA 向 PC 机发送数据时从 B 口读取数据，因此 B 口设为 Read Only，设置完毕后单击“Generate”按钮即可。

10.6.4　FPGA 代码的设计实现

整个代码采用了 4 个状态机来实现，其中两个状态机用于从串口读取数据，另外两个用于向串口写入数据。下面以从串口读取数据为例来说明状态机的工作过程。

从串口读取数据的两个状态机主要完成如下功能：从串口接收到字符，并判断是不是 E4H、00H（自定协议规定的内容），如果是，就给出 Response（应答）信号，通知发送数据状态机开始向 PC 机发送数据。这两个状态机，一个为主状态机，用于连续从串口读取数据；一个为辅状态机，用于从串口读取一个字节的数据。由于串口接收数据是一位一位地接收，该状态机就控制从串口接收所有位，并将结果存入一个 8 位的寄存器。

当辅状态机接收到一个字符后，就给主状态机一个 Received_char 信号。主状态机判断是不是 E4H 信号，如果是，就跳到下一个状态，等待辅状态机接收下一个字符；如果下一个接收到的字符是 00H，那么就给

开发板免费领取试用　开发板免费领取试用

出 Response 信号。

系统状态图如图 10.26 所示。

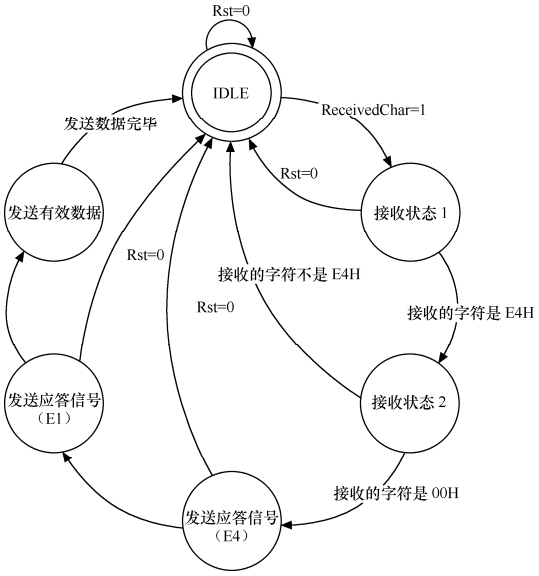


图 10.26　系统状态图

状态机描述是一种非常好的方法，用户可在此基础上加以修改，用于自己的实际设计当中。单个字符接收的状态机如图 10.27 所示。

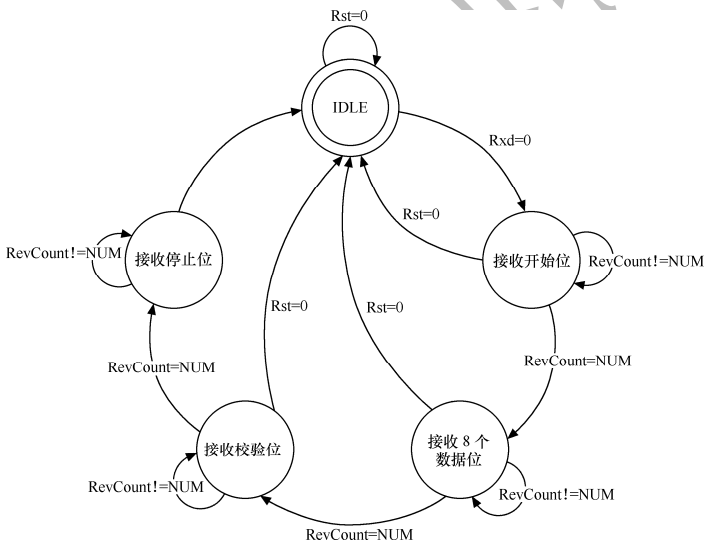


图 10.27　单字符接收状态机

对于发送数据的两个状态机，其工作机理与上面相同，不再赘述。详细设计参考实例　　代码。

10.6.5　波特率的设定

串口通信必须要设定波特率。本设计中采用的波特率为 9600kbit/s，采用的时钟为 50MHz，相当于传送一位数据需要约 5028 个时钟周期。这里采用减法计数器来控制，即计数器计到 5028 个时钟周期后，就开始传输下一位数据（也可以通过对时钟分频来实现）。

10.6.6　ModelSim 仿真验证

FPGA 从串口接收数据仿真结果，如图 10.28 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

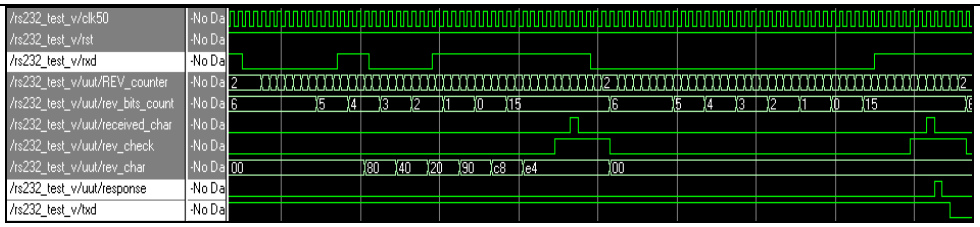


图 10.28　从串口接收数据仿真结果

rxd 为模拟 PC 机发出的 E4H、00H 信号。从图中可以看到，当 FPGA 收到这两个信号后，给出了 Response 脉冲。发送状态机收到该信号后，给出 txd 的低电平信号，通知 PC 机要开始发送数据了。

如图 10.29 所示，txd 为 FPGA 向 PC 机发送的数据。发送状态机收到 Response 信号后，给出 txd 低电平信号，开始发送数据。先发送应答信号 E4H、E1H，之后开始发送开始写入 BlockRam 的 20 个数，从 0～19。图中 raddr 为 BlockRam 的地址，rdata 为从 BlockRam 中读取的数据。

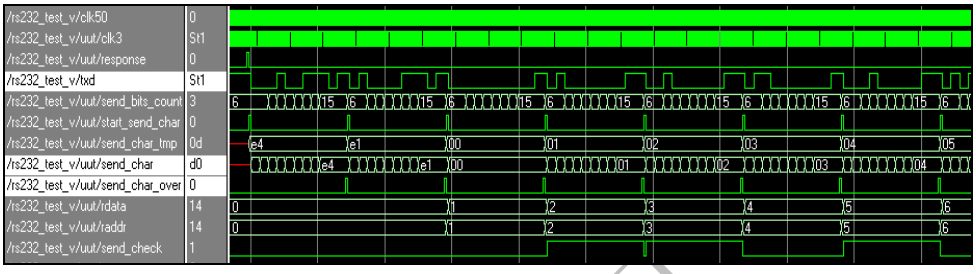


图 10.29　SDRAM 控制器仿真结果

在仿真时，为了显示方便，没有按照 5028 个时钟传输一位，而是 4 个时钟就传输一位。

10.6.7　小结

本节对数字系统中常用的 UART 控制器做了初步的介绍，并在 Modelsim 中实现了对 SDRAM 控制器的仿真，最后通过编译下载在红色飓风的开发板上实现了预定功能。

通过这个实例，读者能够掌握 UART 控制器以及 RS-232C 接口的设计实现方法，并学会将这个控制器集成到更为复杂的设计中去。

10.7　典型实例 17：USB 2.0 接口的设计与实现

10.7.1　实例的内容及目标

1．实例的主要内容

本节旨在设计实现了 FPGA 通过 FX2 USB 2.0 接口芯片与 PC 机进行高速数据通信，分为读数据、写数据和读写数据 3 部分内容。帮助读者进一步了解 USB 接口芯片的工作原理和设计方法。

2．实例目标

通过本实例，读者应达到如下目标。

- 了解 FX2 CY7C68013 芯片的工作原理和 Slave FIFO 模式时序。
- 了解 FX2 的固件设计以及 USB 驱动程序设计。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

- 熟练掌握状态机的使用。
- 实现 FPGA 与 PC 机之间的 USB 接口通信。

10.7.2 USB 接口通信实战步骤

首先创建工程并为工程添加文件，如图 10.30 所示。
然后编译工程并下载至硬件，如图 10.31 所示。

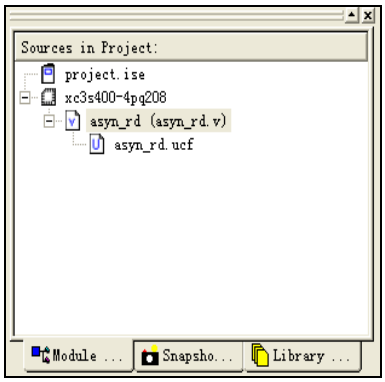


图 10.30 创建工程并添加文件

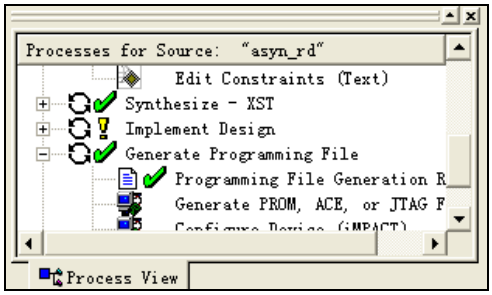


图 10.31 编译工程并下载

接下来可以加载固件了，固件程序的载入有两种方式。
（1）通过芯片的 I^2C 总线连接外部的 EEPROM，固件代码事先通过烧写器写入 EEPROM 中，USB 设备上电运行时，通过 I^2C 总线将 EEPROM 中固件代码载入。EZ-USB 支持外部 EEPROM 通过总线来下载固件，这种方式使开发者可以从外围硬件来下载 8051 程序代码，但是不利于在设备开发阶段使用。
（2）使用该芯片特有软配置功能，将固件程序存储在计算机中，当该设备接入 USB 电缆时，由于 EZ-USB 具有重新枚举的能力，所以在初始化枚举以后，用户只需要通过 Cypress 公司提供的开发软件 USB Control Panel 中 Download 项，就可以将固件载入到控制芯片中。该方法完全是软操作，不需要额外的硬件设备，方便程序的修改调试。
使用 USB Control Panel 进行固件程序下载的界面如图 10.32 所示。

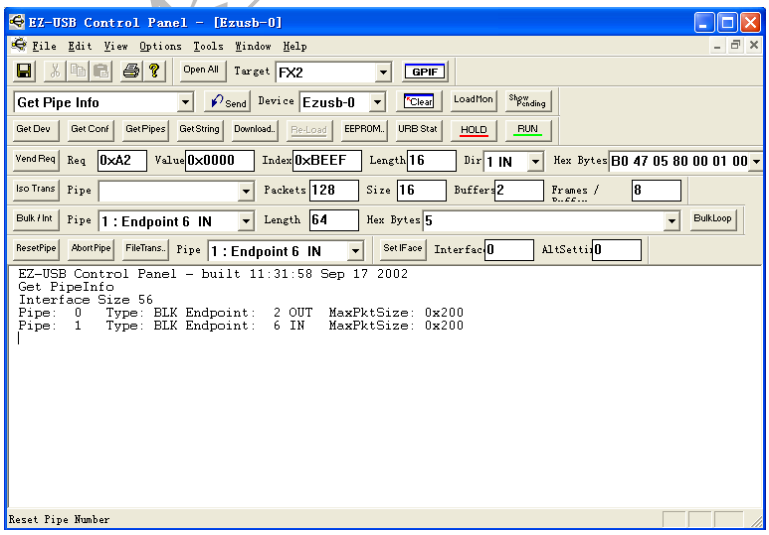


图 10.32 USB Control Panel 界面

单击“Download…”按钮，选择“slavefifo.hex”。下载固件成功以后显示如图 10.33 信息。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

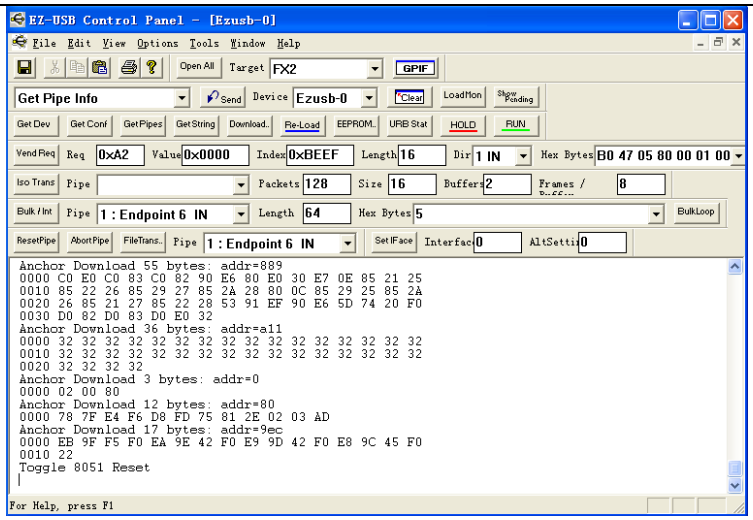


图 10.33　下载固件程序

其中，通过单击“GetPipes”按钮可以查看通道信息。

现在固件程序后，即可进行 USB 通信测试。

根据不同的程序，选择相应的测试软件，测试 USB 接口的传输速度。如图 10.34 所示是 Red Logic 工作室提供的基于红色飓风 II 的 USB 测试软件。

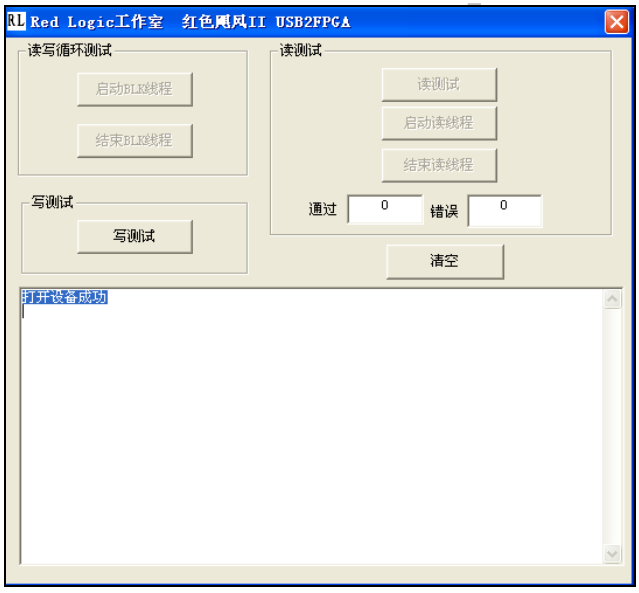


图 10.34　USB 测试软件

10.7.3　USB 接口通信实例结果

实现 FPGA 与 PC 机之间的 USB 通信，并且在 PC 机的超级终端上面测试 USB 的读写速度，如图 10.35 和图 10.36 所示。关于 EZ-USB 的详细内容，可参见工程文件夹中提供的相关文档。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

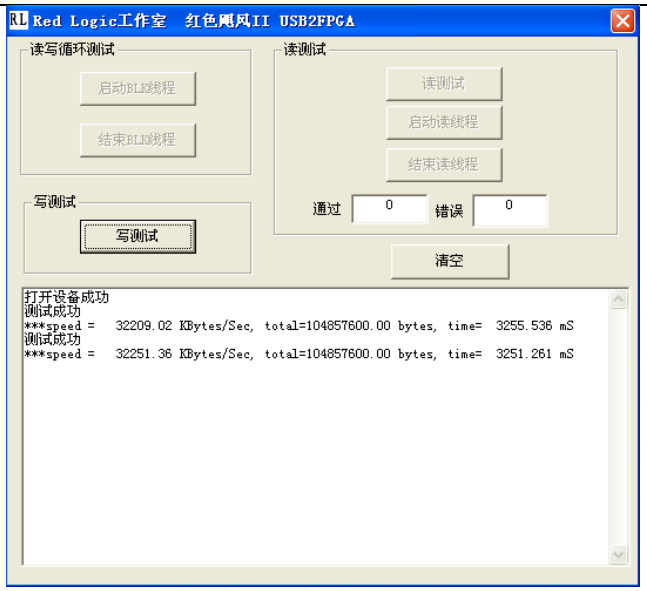


图 10.35 USB 测试结果

图 10.36 USB 测试结果

10.7.4 FPGA 代码的设计实现

本程序功能是配合 CY68013 的 Slave FIFO 接口时序。它完成接收从主机下传的 60KB 数据，写入板上 SRAM 里，然后从板上 SRAM 中读出，再上传至主机。整个传输过程通过 CY68013 的 Slave FIFO 来交互。
整个程序由一个状态机构成，包括以下状态：

```
    Parameter IDLE='H0,
              READ_EVENT='H1,
              POINT_TO_OUT_FIFO='H2,
              DATA_READY='H3,
              READ_INTERVAL='H4,
              READ='H5,
              READ_END='H6,
              WRITE_EVENT='H7,
              POINT_TO_IN_FIFO='H8,
              WRITE_READY='H9,
              WRITE='HA,
              WRITE_END='HB;
```

每个状态的作用描述如下。

开发板免费领取试用 开发板免费领取试用

- IDLE：整个操作过程（包括读 SLAVE FIFO 和写 SLAVE FIFO）的入口。对相关的寄存器进行初始化，然后转入 READ_EVENT 状态，开始读 SLAVE FIFO 操作。
- READ_EVENT：把 u_addr[1:0] 置为 'b00，指向输出 FIFO（对应端点 6），然后转入 POINT_TO_OUT_FIFO 状态。
- POINT_TO_OUT_FIFO：判断 u_flagc 是否为高（u_flagc 为高指示输出 FIFO 为空，即输出 FIFO 中有数据），如果为高，则启动读过程，把 u_sloe 置为低，转入 DATA_READY 状态，第一个 16 位数据出现在总线上；否则说明输出 FIFO 中无数据，等待。
- DATA_READY：判断 u_flagc 是否为高，如果为高，把 u_slrd 拉低，继续读取下 16 位数据。同时为把上一 16 位数据写入 SRAM 做准备（主要是 SRAM 的三态总线），同时转入 READ 状态，否则转入 POINT_TO_OUT_FIFO，等待下一次读取过程。
- READ：把上一 16 位数据写入 SRAM，同时把 u_slrd 拉高，当前 16 位数据读取结束。判断是否是 60KB 数据，如果不是，则转入 DATA_READY 状态，继续读操作；否则转入 READ_END 状态，读操作结束。
- READ_END：把相关寄存器置为初始态，转入 WRITE_EVENT 状态，开始写操作。
- POINT_TO_IN_FIFO：为从 SRAM 中读取数据作准备，转入 WRITE_READY 状态。
- WRITE_READY：判断 u_flagb 是否为高（u_flagb 为高指示输入 FIFO 非满），如果为高，则启动写过程，从 SRAM 中读取数据并送到 SLAVE FIFO 总线上，把 u_lswr 置为低，转入 WRITE 状态；否则说明输入 FIFO 已满，等待。
- WRITE：把 u_slwr 置为高，当前数据写入 SLAVE FIFO。判断是否是 60KB 数据，如果不是，则转入 WRITE_READY 状态，继续写操作；否则转入 WRITE_END 状态，写操作结束。
- WRITE_END：把相关寄存器置为初始态，转入 IDLE 状态，开始下一个 60KB 的读写操作。

状态机的源代码如下：

```
    case(STATE)
        IDLE:
            begin
            //添加 RESET 状态
            data_wr<='h0;                  // USB 接口信号初始化
            u_slwr<='b1;
            u_slrd<='b1;
            u_sloe<='b1;
            u_addr0<='b1;
            u_addr1<='b1;
            oe<='b0;

            sram_d_i<='h0;                 // SRAM 的控制信号初始化
            sram_a<='h3ffff;
            sram_re<='b1;
            sram_wr<='b1;
            wr_flag<='b0;

            STATE<=READ_EVENT;
            end
        READ_EVENT:
            begin
            wr_flag<='b1;                  // 设定读写标志
            u_addr0<='b0;                  // 指定端点 FIFO
            u_addr1<='b0;
            STATE<=POINT_TO_OUT_FIFO;
            end
        POINT_TO_OUT_FIFO:
            begin
            if(u_flagc)                    //  如果 flagc 高，FIFO 不空，开始读数据
                begin
                u_sloe<='b0;               // 开始从 FX2 的端点 FIFO 读数据
                u_slrd<='b1;
                STATE<=DATA_READY;
                end
            else
               begin                  // 如果 flagc 为低，FIFO 为空，等待 FIFO 有数据
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

```
            u_sloe<='b1;          // 停止从 FX2 的端点 FIFO 读数据
            u_slrd<='b1;
            STATE<=POINT_TO_OUT_FIFO;
            end
        end
    DATA_READY:
        begin
        if(u_flagc)               // 如果 flagc 为高，继续读取下一个数据
            begin
            u_slrd<='b0;
            sram_a<=sram_a+1;     // 把上一个读取的数据写入 SRAM
            sram_d_i<=data;
            sram_wr<='b0;
            sram_re<='b1;
            STATE<=READ;          // 完成数据写入后，进入读数据状态
            end
        else
            begin
            u_slrd<='b1;          // 如果 FIFO 空，回到等待状态
            u_sloe<='b1;
            STATE<=POINT_TO_OUT_FIFO;
            end
        end
    READ:
        Begin
        u_slrd<='b1;              // 完成上一个数据的 SRAM 写周期
        sram_re<='b1;
        sram_wr<='b1;

        if(sram_a!=ADDR_FULL)     // 如果 SRAM 地址没有到最大值，继续读操作
            STATE<=DATA_READY;
        else
            STATE<=READ_END;      // 如果 SRAM 地址到达最大值，结束读操作
        end
    READ_END:
        begin
        u_slrd<='b1;              // 回到初始状态，准备写操作
        u_sloe<='b1;
        u_addr0<='b0;
        u_addr1<='b0;
        sram_a<='h3ffff;
        STATE<=WRITE_EVENT;
        end
    WRITE_EVENT:
        begin
        u_addr0<='b0;             // 指定写数据的端点 FIFO
        u_addr1<='b1;
        oe<='b1;
        wr_flag<='b0;
        STATE<=POINT_TO_IN_FIFO;
        end
    POINT_TO_IN_FIFO:
        begin
        sram_a<=sram_a+1;         // 从 SRAM 中读取一个数据
        sram_re<='b0;
        sram_wr<='b1;
        STATE<=WRITE_READY;
        end
    WRITE_READY:
        begin
        if(u_flagb)
            begin                 // 如果 FIFO 不满，开始写数据到 FX2 的 FIFO
            data_wr<=sram_d;
            u_slwr<='b0;
            u_slrd<='b1;
            STATE<=WRITE;
            end
        else
            begin
            u_slwr<='b1;          // 如果 FIFO 已满，等待
```


开发板免费领取试用 开发板免费领取试用

```
                u_slrd<='b1;
                STATE<=WRITE_READY;
                end
            end
        WRITE:
            begin
            u_slwr<='b1;
            u_slrd<='b1;
            if(sram_a!=ADDR_FULL)
                begin
                sram_a<=sram_a+1;   // 如果 SRAM 地址没有达到最大值，继续从 SRAM 读数据
                sram_wr<='b1;
                sram_re<='b0;
                STATE<=WRITE_READY;
                end
            else
                begin
                sram_a<='h3ffff; // 如果 SRAM 地址达到最大值，复位 SRAM 地址，进入写结束
                sram_wr<='b1;
                sram_re<='b1;
                STATE<=WRITE_END;
                end
            end
        WRITE_END:
            begin
            wr_flag='b0;            // 结束写 FX2 FIFO 状态，回到初始的 IDLE 状态
            sram_a<='h3ffff;
            u_addr0<='b1;
            u_addr1<='b1;
            STATE<=IDLE;
            end
        default:
            STATE<=IDLE;
        endcase
```

10.7.5 小结

本节对利用 USB 接口芯片 FX2 来完成 FPGA 和 PC 机的高速数据传输做了介绍，并通过编译下载在红色飓风的开发板上实现了预定功能。

10.8 典型实例 18：字符 LCD 接口的设计与实现

10.8.1 实例的内容及目标

1．实例的主要内容

本节旨在设计实现 FPGA 与字符 LCD 的接口，帮助读者进一步了解字符液晶的工作原理和设计方法。

2．实例目标

通过本实例，读者应达到以下目标。

- 学习字符型液晶显示器的控制原理。
- 掌握利用 FPGA 设计驱动的基本思想和方法。
- 熟练掌握状态机的使用。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

10.8.2 字符 LCD 接口实例内容

红色飓风开发板配套的字符液晶模块为2行，每行16个字符。本实要求显示功能如下：第一行显示“Welcome RedLogic World !”，第二行显示“RedLogic World!”，由于一行只能显示16个字符，因此要整屏左移显示。

10.8.3 FPGA 代码设计实现

本设计的程序主要包含两个模块，一个模块为 char_ram，另一个模块为 LCD 的驱动模块 lcd。
char_ram 模块的主要功能是根据输入的地址，输出相应字符在 CGRM（字符存储发生器）中的地址。因为在控制 LCD 显示时，只需用户提供相应字符的地址，就可以显示出该字符。
在 char_ram 中，先设置好所有字符对应的地址，然后给需要用到的字符定义新的地址用于选择输出。
Lcd 模块用于驱动 lCD 正常工作。LCD 是一个慢显示器件，因此时钟一定要满足要求，在这里通过 50MHz 分频得到输入的时钟周期约为 100μs（约 10Hz），以满足慢显要求。LCD 的驱动模块利用一个状态机来实现。具体程序参考实例源代码。

10.8.4 ModelSim 仿真验证

通过 Add Source 的方法直接将测试代码加入到工程当中。选中要加入的测试代码文件：led_behavor_test.v，弹出如图 10.37 所示的对话框，选择“Verilog Text Fixture File”。
用户也可以直接新建一个仿真文件，选择“New Source”，选择“Verilog Test Fixture”，输入测试文件名称，如图 10.38 所示，单击“Next”按钮。
选择 lcd 模块（对哪个模块进行测试，就选择哪个文件），如图 10.39 所示。

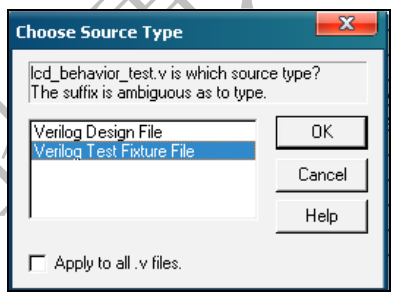


图 10.37 选择源文件类型

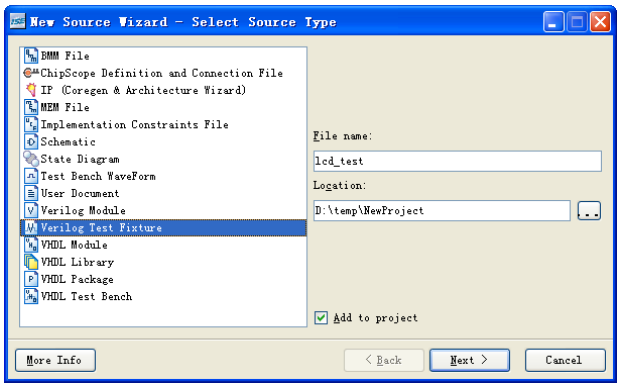


图 10.38 新建仿真源文件

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

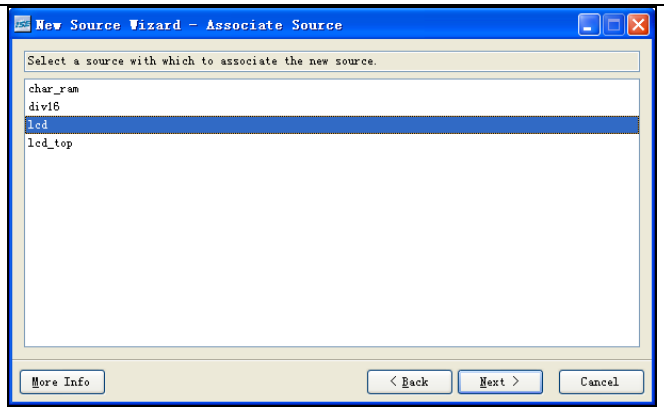


图 10.39　选择被测试模块

新建测试文件后，ISE 会自动生成一部分代码，其余内容可直接从源工程中提供的测试文件中复制。添加后选中测试文件可以看到如图 10.40 所示的信息。

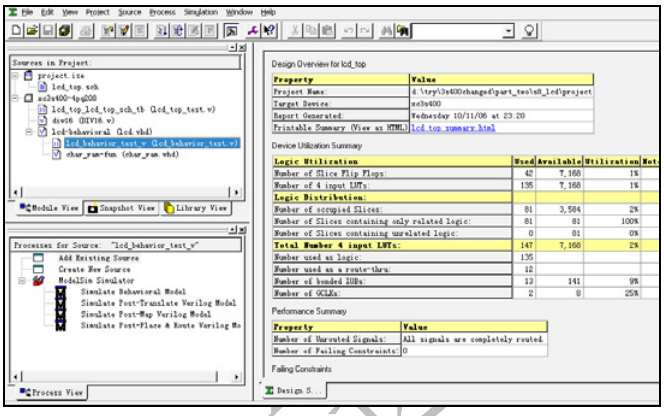


图 10.40　仿真文件信息

在 Process for source 下有 4 个 M 的图标，分别对应 4 个级别的仿真。从上至下，第一个为行为仿真，第二个为翻译后仿真，第三个为映射后仿真，第四个为布局布线后仿真。在这里先做行为仿真，只需双击第一个图标，就可启动 ModelSim 仿真，结果如图 10.41 所示。

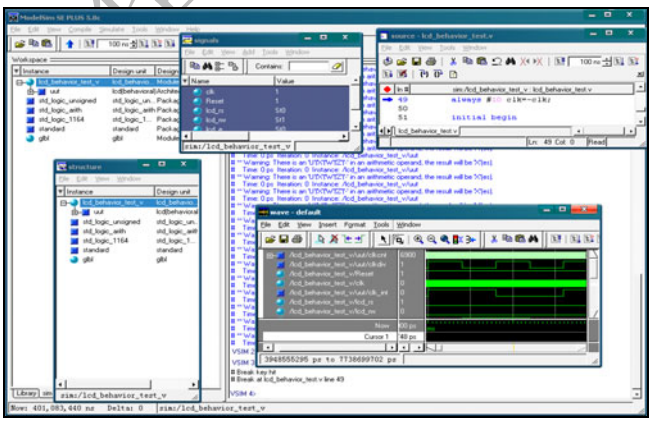
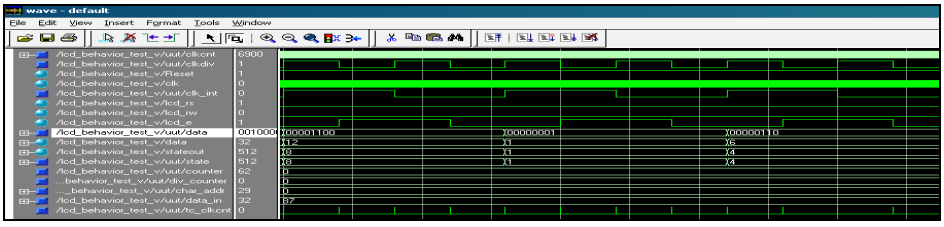


图 10.41　行为仿真

默认情况下，ModelSim 会弹出 4 个窗口，在这里主要观察 Wave 窗口，如图 10.42 所示。



华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

图 10.42　行为仿真波形窗口

上面列出的很多信号是人为添加的，具体如何添加以及具体 Wave 窗口下的许多用法在这里不做详细叙述。需要特殊说明的是，如果用户自己装的是 ModelSim XE 版本，是无法完成这个仿真的，因为这个工程代码是用 VHDL 写的，测试代码是用 Verilog 写的，ModelSim XE 版本是不支持混仿真的，建议安装 ModelSim SE 版本，SE 下是支持混合仿真的。

ModelSim 仿真结果如图 10.43 和图 10.44 所示。

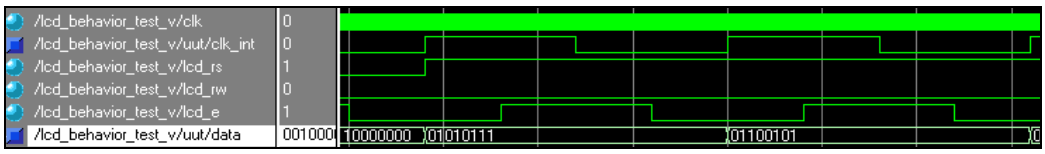


图 10.43　LCD 写数据仿真结果

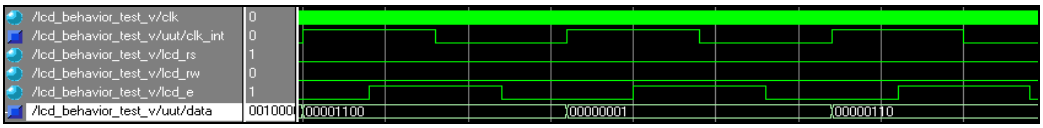


图 10.44　LCD 写指令仿真结果

这两个波形分别为写数据和写指令的仿真结果。写数据即向 DDRAM 写入要显示的字符对应的地址，其中 clk 为系统的输入时钟 50MHz，clk_int 为分频后得到的约 10Hz 输入时钟，lcd_e 为使能信号，写数据时 RS=1，R/W=0，写指令时 RS=0，R/W=0。

10.8.5　小结

本节对利用 FPGA 来驱动字符型液晶模块的方法做了介绍，并通过编译下载在红色飓风的开发板上实现了预定功能。

10.9　典型实例 19：VGA 接口的设计与实现

10.9.1　实例内容及目标

1．实例的主要内容

本节旨在设计实现了 FPGA 与 VGA 显示器的接口，帮助读者进一步了解 VGA 接口的时序和设计方法。

2．实例目标

通过本实例，读者应达到以下目标。

- 掌握 VGA 接口的标准时序。
- 掌握利用 FPGA 设计驱动的基本思想和方法。
- 熟练掌握状态机的使用。

10.9.2　VGA 接口实例内容

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

红色飓风开发板的 VGA 接口可以产生 8 种颜色，可以用来显示字符和简单图形。本实例要求通过 VGA 接口，在显示器上依次显示 8 种颜色的彩色条纹，条纹宽度相同。

10.9.3 FPGA 代码设计实现

按照 VGA 时序要求产生行/列同步信号，并且在指定的时间（有效像素期间）将要显示的颜色数据输出到 RGB。

不同的显示模式和刷新频率下，具体同步信号前、后、同步信号、有效像素数是不同的，要根据像素时钟频率来设置。以 800×600，60Hz 为例，像素时钟为 40MHz，即：

像素时钟=(800+40+128+88)×(600+1+4+23)×60=40MHz

设计中可根据系统时钟频率来选择合适的显示模式。

10.9.4 ModelSim 仿真验证

1．行显示

设定一个列计数器，计数到相应的值时，给出相应的同步信号，并在像素有效期间，输出有效的颜色数据，行计数器通过检测列同步信号来计数，检测到列同步信号，表示新一行的开始，这样就可控制在不同的行显示不同的内容。

仿真结果如图 10.45 所示。

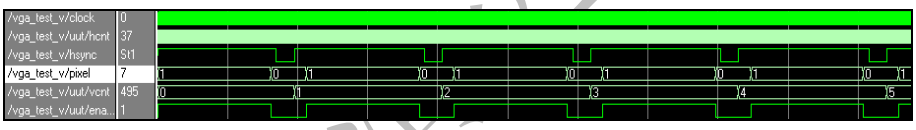


图 10.45　行显示仿真结果

其中，hcnt 为列计数，列计数到有效像素时，给出 enable 信号，只有 enable 信号为高时，才给出有效像素数据，并且在每个同步信号后，行计数器加 1。

如图 10.46 所示为行同步信号产生时仿真结果，当计数到 604-1=603 行时，给出行同步信号，同步信号低电平持续 4 个行的时间，同步信号后持续 21 行的时间，之后再从第一行第一列开始。

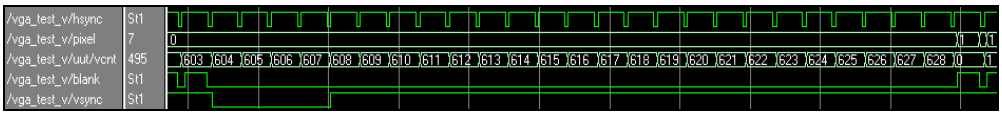


图 10.46　行同步信号产生后仿真结果

2．列显示

如图 10.47 是列显仿真结果。列显示和行显示不同在于，同一行的像素，计数到不同的值时，输出不同的颜色数据，每一行都是如此，最终就可得到列显的彩色条纹。

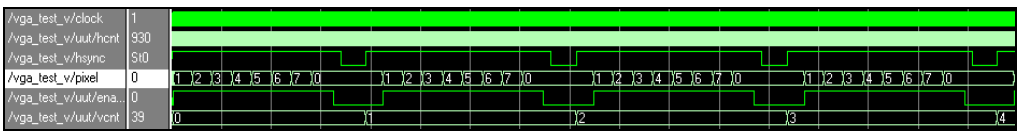


图 10.47　列显仿真结果

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

10.9.5 小结

本节对利用 FPGA 来驱动 VGA 接口显示器的方法做了介绍，并通过编译下载在红色飓风的开发板上实现了预定功能。

联系方式

集团官网：www.hqyj.com 嵌入式学院：www.embedu.org 移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn 物联网学院：www.topsight.cn 研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路银海大厦 A 座 8 层，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

华清远见 FARSIGHT 始于2004 | 嵌入式培训专家

10年口碑积累，成功培养50000多名研发工程师，铸就专业品牌形象

华清远见的企业理念是不仅要做良心教育、做专业教育，更要做受人尊敬的职业教育。

《FPGA 应用开发入门与典型实例》（修订版）

作者：华清远见

专业始于专注 卓识源于远见

第 11 章 FPGA 与 DSP 协同处理系统设计

本章目标

- 了解 FPGA+DSP 的协同处理平台的特点和适用领域
- 掌握 FPGA+DSP 的协同处理平台的构建方法和测试手段
- 掌握 FPGA 与 DSP 的接口设计方法

专业始于专注 卓识源于远见

开发板免费领取试用 开发板免费领取试用

11.1 基于 FPGA+DSP 协同处理平台的优势和适用领域

11.1.1 基于 FPGA 的信号处理系统的特点

FPGA 的一个重要的应用领域就是数字信号处理，随着 FPGA 密度和速度的提高，现在 FPGA 已经可以胜任一些原来只有专用芯片或者多 DSP 才能完成的计算任务。

1．基于 FPGA 的信号处理系统的优点

首先，FPGA 支持并行和流水结构。这样可以通过多个处理单元（PE）的并行工作，实现高性能的数字信号处理。特别适合那些结构性好、数据量大的算法实现，比如快速傅立叶变换（FFT）、数字滤波算法（FIR）等。

其次，FPGA 内部越来越多地内嵌了 DSP 乘法模块。这些模块是硬件模块，运行速度很高，特别适合那些需要大量乘法计算的算法。如表 11-1 所示为 Xilinx 公司 FPGA 的硬件乘加单元资源和性能。

表 11.1　　Xilinx 公司 FPGA 的硬件乘加单元资源和性能

最大的产品系列成员	最高的 MAC 数量	最高的时钟频率	GMAC
Virtex-5 LX330	192	550MHz	105
Virtex-4 SX55	512	500MHz	256
Virtex-4 FX140	192	500MHz	96
Virtex-4 LX200	96	500MHz	48
Virtex-II Pro100	444	300MHz	133
Spartan-3S5000	104	185MHz	19

可以看到，FPGA 的乘法运算性能最高已经超过了 100GMAC 的水平，远超过通用的 DSP 芯片，接近了专用处理芯片（ASIC）的能力。

另外，FPGA 和其他所有可编程器件一样，具体非常好的灵活性。特别是某些型号的 FPGA 已经开始支持动态配置或者局部重构，为设计高智能型的信号处理设备提供了可能。

2．基于 FPGA 的信号处理系统的不足

目前 FPGA 系统的主要设计手段还是通过硬件描述语言（HDL）来描述完成的。对于大型复杂的算法实现，设计人员需要做大量的结构设计和仿真验证工作。而且，FPGA 是基于硬件连线可编程的器件，因此不支持在线调试功能。

11.1.2 基于 DSP 的信号处理系统的特点

1．基于 DSP 的信号处理系统的优点

DSP 芯片是基于软件可编程的，开发语言主要是 C 语言，个别场合需要编写汇编语言。相对与 HDL 语言，C 语言更容易让初学者接受，因此 DSP 芯片在信号处理领域得到了广泛的应用。

C 语言属于高级语言，因此在 DSP 上描述复杂算法的难度要低于 FPGA。另外，DSP 芯片的结构也适合数据量大，重复性高的算法。特别是乘加累加计算（MAC）密集和 for 循环密集的算法。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

DSP 芯片一般采用时间上压缩和空间上并行两种主要手段来实现高性能的数据处理。时间上压缩技术包括单周期指令、专用的硬件乘法累加器以及多种寻址方式等；空间上并行技术包括哈佛式总线结构、多运算单元以及灵活的流水线技术等。

2．基于 DSP 的信号处理系统的不足

DSP 芯片是基于软件可编程的，因此不可避免的问题就是功耗大，处理能力受到主频的限制等不足。而且，DSP 芯片的数据带宽受到总线接口位宽限制，不能应付高速数据流的处理要求。

通用 DSP 的接口扩展能力有限，不能直接和 ADC 或者 DAC 芯片互联。DSP 需要外部的逻辑器件来配合才能实现数据采集或者输出，例如 CPLD 或者 FPGA。

目前大部分的 DSP 都没有内存管理单元（MMU），还不能支持复杂的操作系统，如 Linux 或者 Windows CE 等。DSP 的内存管理和保护能力比较弱，目前还不支持虚拟内存。

11.1.3 基于 FPGA+DSP 的信号处理系统优势和应用领域

1．基于 FPGA+DSP 的信号处理系统的优势

基于 FPGA+DSP 的信号处理系统可以同时具备 FPGA 和 DSP 的优势。

（1）支持更高的计算处理能力。

FPGA+DSP 的双芯片处理系统的计算能力相对单芯片系统更加强大，而且可以充分发挥两种芯片的性能潜力。

（2）灵活的系统重构方案。

FPGA 芯片和 DSP 芯片两者之间可以相互配置。通常情况下，DSP 会作为系统的主控制器，FPGA 上电后由 DSP 来完成配置。在系统的工作期间，DSP 可以根据需要，重新配置 FPGA，实现系统的功能重构。例如，在软件无线电应用中，经常需要根据发射系统调制方式的改变来动态调整接收系统功能。这种应用中就可以充分发挥 FPGA+DSP 构建的灵活性，DSP 可以根据调制方式来选择不同的配置文件加载到 FPGA 中，实现自适应的动态配置。

另外，系统工作过程中，如果 DSP 芯片在自检过程中发现功能异常，也可以请求 FPGA 芯片重新配置自己。这样的系统具备一定的自我修复能力，更加智能化。

（3）开发难度和系统成本的折衷

从控制系统成本角度来看，单芯片应该是最好的方案。但是，无论是采用 FPGA 还是 DSP，单芯片的方案开发难度都会提高。单 FPGA 的方案实现复杂的控制会比较困难，而单 DSP 的方案实现大数据量的重复计算也有很大难度。

采用了 FPGA+DSP 的方案虽然成本要高于单芯片方案，但是降低了开发难度，可以加快产品的上市时间。因此，双芯片的构建是一种开发难度和系统成本的折衷选择。同时，也降低系统开发的技术风险。

2．基于 FPGA+DSP 的信号处理系统的应用领域

基于 FPGA+DSP 的非常适合既有复杂算法处理又需要大量的重复计算应用，例如软件无线电、高清视频图像处理、信息安全等领域。

FPGA 作为 DSP 的协处理器，给 DSP 提供单元加速功能。例如，FPGA 可以实现一个浮点协处理单元（FPU），给高性能的定点 DSP 提供浮点计算能力，适合于数值分析和高精度信号处理领域。

基于 FPGA+DSP 的系统配置方式比较灵活。FPGA 和 DSP 可以单独配置，也可以相互配置。因此，这种方案可以用于智能的动态重构计算平台。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

11.2 基于 FPGA+DSP 的协同处理平台的设计流程

11.2.1 FPGA 与 DSP 的功能划分

FPGA+DSP 系统的设计流程和单独 FPGA 或者 DSP 设计流程相比，最大的不同就是在设计流程里面增加了功能划分环节。这个环节的主要目的就算将系统功能进行分解，然后分配给 FPGA 和 DSP 去执行。

1．功能划分的原则和依据

功能划分的基本原则就是把数据密集，但算法简单、重复性高的功能放在 FPGA 上执行，而把算法复杂，重复性低的功能分配给 DSP 实现，这样可以充分发挥两种芯片的特长。
功能划分的主要依据是系统的指标需求和功能定义，即系统在进行功能划分之后，不能影响系统的整体性能和指标。

2．功能划分的方法

（1）按照数据流向来划分。
这种划分方法一般应用于数据采集领域，具体来说，就是 FPGA 负责采集数据，并把数据送给 DSP 芯片，而 DSP 用来对采集到的数据进行处理。
例如，在图像处理领域一般采用 FPGA 完成图像的采集，并送给 DSP 芯片处理。FPGA 也可以对输入的图像进行预处理后再送给 DSP 完成后期的编码和传输功能。这样的系统中，数据流是单向传输的，如图 11.1 所示。
（2）按照算法流程来划分。
如果 FPGA 作为 DSP 的协处理器，那么一般采用按算法流程来划分的方法。DSP 负责整个算法的调度和数据流的控制，把需要处理的数据送给 FPGA，并从 FPGA 取回处理后的结果。
例如，在 FPGA 充当 DSP 的浮点协处理单元（FPU）的应用中，DSP 芯片在需要浮点运算的时候会把数据送给 FPGA 去处理。当 FPGA 处理完成后，会通知 DSP 取回处理结果，如图 11.2 所示。

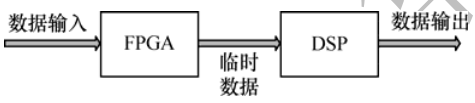


图 11.1 按照数据流向来划分功能的系统框图

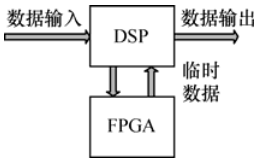


图 11.2 按照算法流程来划分功能的系统框图

11.2.2 FPGA+DSP 的系统设计流程

FPGA+DSP 的双核系统设计流程的示意图如图 11.3 所示。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

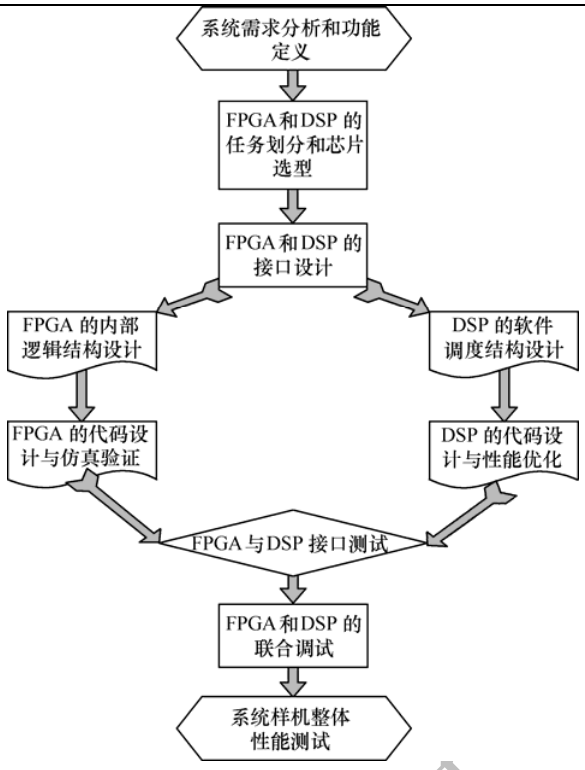


图 11.3　FPGA+DSP 的双核系统设计流程的示意图

从上面的图中可以看出，整个流程中比较重要的环节是 FPGA 与 DSP 的通信接口设计方法以及联合调试的注意事项。

11.3　FPGA 与 DSP 的通信接口设计

FPGA 与 DSP 的通信接口可以分成以下几种。

- 存储型接口，如 EMIFA、EMIFB 等。
- 主机型接口，如 HPI 等。
- 总线型接口，如 PCI、EMAC 等。
- 串口型接口，如 McBSP、McASP、SPI、UART 等。
- 特殊类接口，如 VLYNQ、VPORT 等。

以上的接口中，比较常用的接口是 EMIF 和 HPI。其中总线接口需要协议支持，开发难度较大，串行接口开发简单，但是速率较慢。VPORT 等特殊接口一般是在特定的场合下应用，不具备通用性，而且需要修改 DSP 驱动，开发周期较长。

11.3.1　基于 TMS320C64x 系列的 EMIF 接口设计

1．EMIF 简介

TMSC64x EMIF 是 TMSC621x EMIF 的增强版。它具备 TMSC621x/TMSC671x EMIF 的所有功能，并添加了以下新功能。

（1）EMIFA 上的数据总线可以是 64 位宽，也可以是 32 位宽。

（2）EMIFB 上的数据总线为 16 位宽。

（3）EMIF 时钟 ECLKOUTx 为内部生成，并基于 EMIF 输入时钟。器件复位时，以下 3 个时钟之一会被配置为 EMIF 输入时钟。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

- 内部 CPU 时钟 4 分频。
- 内部 CPU 时钟 4 分频。
- 外部 ECLKIN 与 TMSC64x EMIF 接口的存储器应按照 ECLKOUTx 的时钟频率运行。
- 可编程同步存储器接口控制器的同步控制引脚替换了固定的 SBSRAM 控制引脚。
- PDT 引脚提供外部到外部传输支持。

如图 11.4 所示为组成 EMIFA 和 EMIFB 接口的信号，表 11.2 对这些信号进行了描述。端口 A 的信号都带前缀“A”，端口 B 的信号都带前缀“B”。为方便起见，所有信号名称都省略了 EMIF 端口前缀。

EMIFA 接口具备与 8、16、32 或 64 位系统接口的功能。EMIFB 接口端口仅支持 8 位和 16 位系统，如图 11.5 所示。

外部器件（主要是存储器）通常与总线的 ED[7:0]字节侧右对齐。Endianess（字节顺序）决定 ED[7:0]位是作为字节 0（little endian（小端））还是字节 N（big endian（大端），其中 2N 是总线宽度）存取。对不同字节通道的选择，可通过低有效字节使能信号的应用（如表 11.2 所示）来完成。

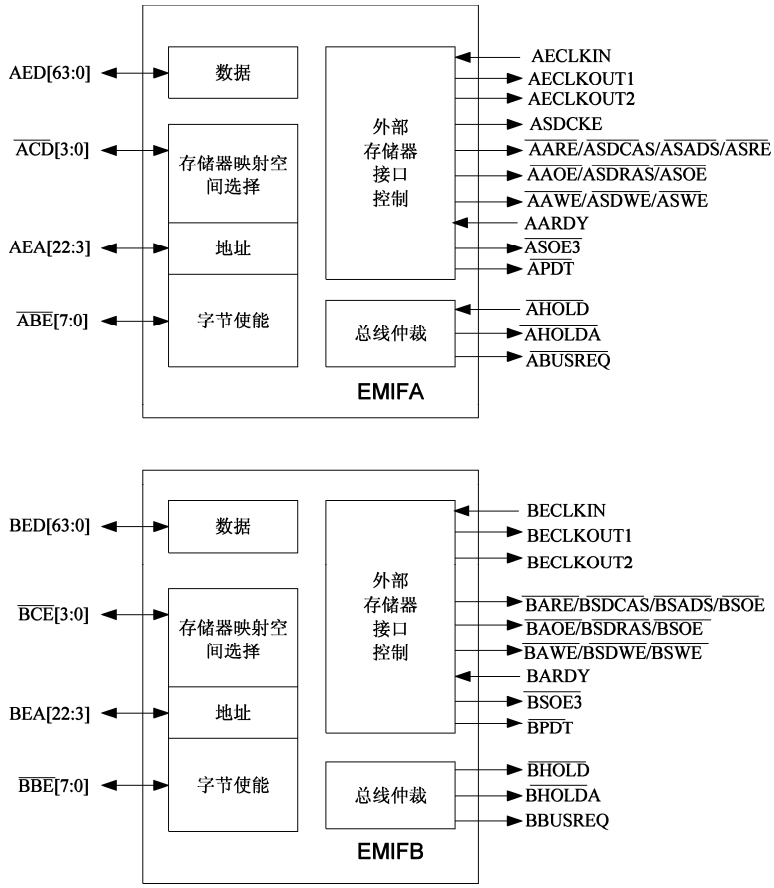


图 11.4　EMIFA 和 EMIFB 信号

表 11.2　　EMIF 信号描述

引　脚	I/O/Z	描　述
CLKOUT4	O/Z	CPU 时钟的 4 分
CLKOUT6	O/Z	CPU 时钟的 6 分
ECLKIN	I	EMIF 时钟输入
ECLKOUT1	O/Z	速率为 ECLKIN、CPU/4 或 CPU/6 时的 EMIF 输出时钟
ECLKOUT2	O/Z	速率为 ECLKIN、CLKOUT4 或 CLKOUT6 时的 EMIF 输出时钟
ED[63:0]	I/O/Z	EMIFA 64 位数据总线
ED[31:0]	I/O/Z	EMIFA 32 位数据总线
ED[15:0]	I/O/Z	EMIFB 16 位数据总线
EA[22:3]	O/Z	EMIFA 地址输出

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

EA[20:1]	O/Z	EMIFB 地址输出
CE0	O/Z	存储器空间 0 的芯片选择
CE1	O/Z	存储器空间 1 的芯片选择

续表

引　脚	I/O/Z	描　述
CE2	O/Z	存储器空间 2 的芯片选择
CE3	O/Z	存储器空间 3 的芯片选择
BE[7:0]	O/Z	EMIFA 64 位字节使能，字节使能仅针对其对应的字节通道有效 如 BE[0] 对应 ED[7:0]
BE[3:0]	O/Z	EMIFA 32 位字节使能，字节使能仅针对其对应的字节通道有效
BE[1:0]	O/Z	EMIFB 16 位字节使能，字节使能仅针对其对应的字节通道有效
ARDY	I	异步就绪输入，低速外设插入等待状态
SOE3	O/Z	CE3 的同步输出使能
AOE	O/Z	异步输出使能
SDRAS	O/Z	DRAM 存储器的行地址选通脉冲
SOE	O/Z	同步输出使能
ARE	O/Z	异步读使能
SDCAS	O/Z	SDRAM 存储器的列地址选通脉冲
SADS/SRE	O/Z	同步地址选通脉冲或读使能
AWE	O/Z	异步写选通脉冲
SDWE	O/Z	SDRAM 的写使能
SWE	O/Z	同步写使能
HOLD	I	外部总线保持请求
HOLDA	O	外部总线保持确认
BUSREQ	O	总线请求
PDT	O/Z	外设数据传输
SDCKE	O/Z	SDRAM 时钟使能
CLKOUT4	O/Z	CP U 时钟的 4 分
CLKOUT6	O/Z	CP U 时钟的 6 分
ECLKIN	I	EMI F 时钟输入
ECLKOUT1	O/Z	速率为 ECLK IN、CPU/4 或 CPU /6 时的 EMIF 输出时钟
ECLKOUT2	O/Z	速率为 ECLK IN、CLKOUT4 或 CLKOUT6 时的 EMIF 输出时钟
ED[63:0]	I/O/Z	EMIFA 64 位数据总线 14
ED[31:0]	I/O/Z	EMIFA 32 位数据总线
ED[15:0]	I/O/Z	EMIFB 16 位数据总线
EA[22:3]	O/Z	EMIFA 地址输出

续表

引　脚	I/O/Z	描　述
EA[20:1]	O/Z	EMIFB 地址输出
CE0	O/Z	存储器空间 0 的芯片选择

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

CE1	O/Z	存储器空间 1 的芯片选择
CE2	O/Z	存储器空间 2 的芯片选择
CE3	O/Z	存储器空间 3 的芯片选择
BE[7:0]	O/Z	EMIFA 64 位字节使能。字节使能仅针对其对应的字节通道有效
BE[3:0]	O/Z	BE7 对应 ED[63:56]
BE[1:0]	O/Z	BE6 对应 ED[55:48]
ARDY	I	BE5 对应 ED[47:40]
SOE3	O/E	BE4 对应 ED[39:32]
		BE3 对应 ED[31:24]
		BE2 对应 ED[23:16]
		BE1 对应 ED[15:8]
		BE0 对应 ED[7:0]

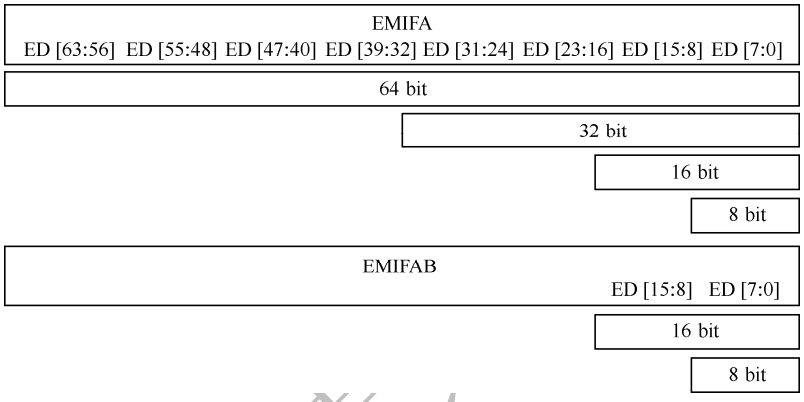


图 11.5 按 Endianess 对齐字节

有关 EMIF 接口及其功能的完整概述，请参阅相关的 TI 数据手册（tms320c64xx.pdf，其中 xx 为 15t、16t 或 18）。

2．FPGA 的 BlockRAM 简介

Xilinx 架构的 FPGA 都可以访问模块存储器（BlockRAM）。Virtex、Virtex-E 和 Spartan-II 器件中的这些 4Kbit 的模块在 Virtex-II、Virtex-II Pro 和 Spartan-3 器件中都增加到 18 Kbit 的模块。

这些模块都是完全同步、真正的双端存储器。用户可独立地从每个端口读出或向每个端口写入（但同一地址不能同时进行读和写）。另外，每个端口都有一个独立的时钟，并且对每个端口的数据宽度都可以独立进行配置。如图 11.6 所示为双端 RAM 模块的框图。

由 FPGA 逻辑和一批 Block RAM 组成的 FIFO 取决于所构建的 FIFO 要求的宽度和深度。FIFO 可使用 Xilinx CORE Generator™工具构建，或者可以用 HDL 手动组合。

使用 CORE Generator 工具构建 FIFO 具有一定优势，即设计可以达到并实现很高的性能指标。按照设计规范用 HDL 构建的设计则可赋予设计人员完全的设计自由。

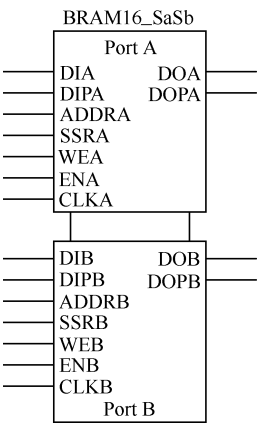


图 11.6 双端口 RAM 模块

3．FPGA 与 DSP 的 EMIF 连接

FPGA 的灵活性使我们可以将其用做具备可选择总线宽度（8 位～64 位）的各

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

类存储器而创建不同的设计。通过接口设计，可将 FPGA 用做同步或异步标准存储器，或同步或异步 FIFO。通过接口设计，可将 FPGA 以同步或异步的方式连接到 EMIF。

在同步模式中，ECLKOUTx 时钟用于驱动 FPGA 接口逻辑。此时钟甚至可以驱动整个 FPGA。FPGA 的逻辑功能、专用乘法器、PPC405 或 MicroBlaze™处理器等使其具备了强大的处理功能。这样，FPGA 就可以用做协处理器或高速数据处理和传输器件。

FPGA 的存储器容量小于 TMSC64x 类型 DSP 的寻址空间。FPGA 存储器必须使用 FPGA Block RAM 来实现。上述 TMSC64x 到 FPGA 的接口为 FIFO 结构，可以使用 Black RAM 资源来实现。

FIFO 接口可使用标准的 TMSC64x EMIF FIFO 接口机制。如表 11.3 所示为 EMIF 信号的总结。

表 11.3　　EMIF 与 FPGA 的接口信号

信 号 名 称	方　　向
CE	DSP 输出
AOE	DSP 输出
AWE	DSP 输出
ARE	DSP 输出
INTx	DSP 输入
INTy	DSP 输入
INTz	DSP 输入
ED[63:0]	DSP 双向

FIFO 要求连续地读时钟和连续地写时钟。这些时钟由 ARE 和 AWE 信号生成，使用 FPGA 的本地时钟控制功能进行布线。

DSP 具有 3.3V 的接口逻辑，所以用于连接到 EMIF 的 I/O 组（bank）必须指定为 3.3V VCCIO。如果无法指定，就必须使用电平移位器件。这一器件可为两个器件间的信号通路引入额外的时间。

4．FIFO 接口的设计实例

本设计使用了标准的 EMIF FIFO 设置。如图 11.7 所示为用于 Xilinx 公司的 Virtex-II Pro 和 Spartan-3 器件的设计示例。

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

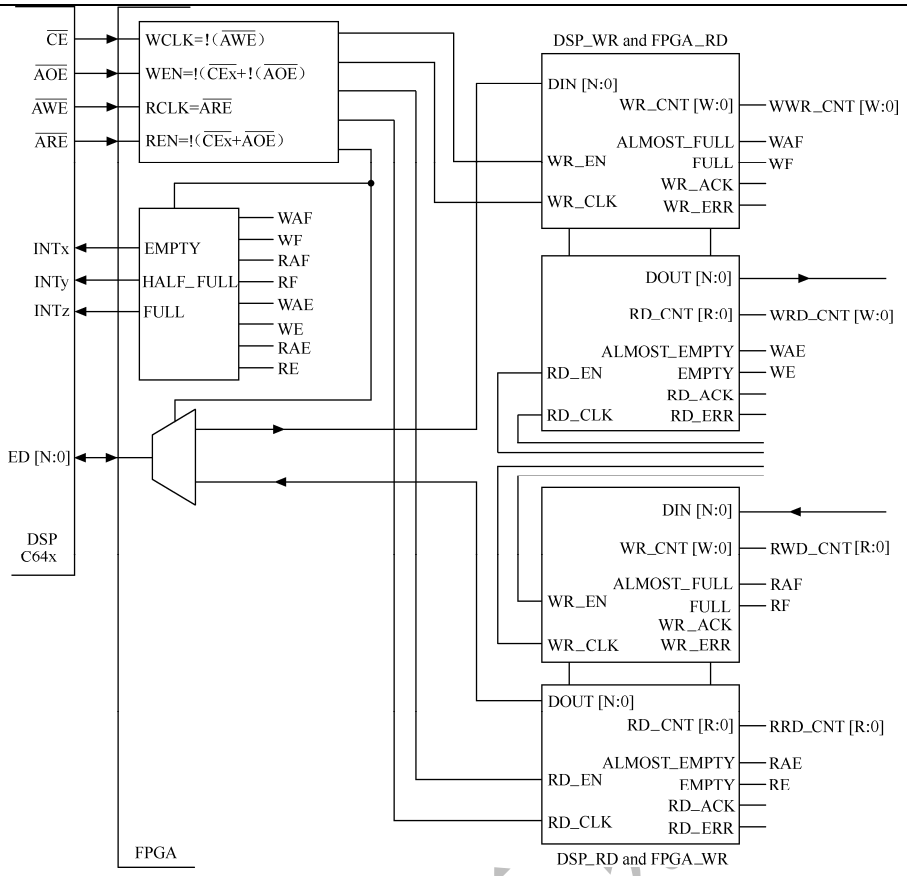


图 11.7　基于 FIFO 的 EMIF 接口实例

数据组芯片使能信号（Cex）与异步输出使能（AOE）一起，用于为设计的写或读 FIFO 部分生成使能信号。此信号还用于为数据多路复用器和标志选择逻辑生成使能信号。

FIFO 的读写时钟通过本地时钟布线直接布线到 FIFO 时钟输入。

FIFO 中使用的 RAMB16 组件由于沿着与 EMIF 相连接的 I/O 块进行排列，显著改善了时序。当 Block RAM 组件的列的大小与使用的 I/O 的数量相同时，即可构建快速的矩形接口。

本设计使用常规的 FIFO 标志输出。要对标志逻辑施加额外控制，无论是针对 DSP 端还是 FPGA 端，可使用写计数器和读计数器输出。

信号 AINIT（图 11.7 中未显示）强制所有标志为高有效状态。AINIT 发出后，在第一个 WR_CLK 上，FULL 和 ALMOST_FULL 标志变为无效，呈低状态。对于在首个 RD_CLK 边沿上的 EMPTY 和 ALMOST_EMPTY 标记，也会发生同样情况。

ALMOST_EMPTY 和 ALMOST_FULL 标志表明只剩下了一个 FIFO 位置。使用 WR_COUNT 和 RD_COUNT 输出，可以构建用户定义的 FIFO 标志。这两个计数器的值都不反映 FIFO 内容（地址）的准确位置。FIFO 的两个端口（时钟域）都存在时钟延迟，长度为一个时钟周期。

输出 WR_COUNT 和 RD_COUNT 为被延迟的内部 FIFO 高位地址计数器。为输出选择两个比特，即可将 FIFO 的位置确定到总尺寸的四分之一（如下面计数器解码示例中所示）。

计数器越宽，位置就能越精细。下面是计数器解码示例。

- COUNT[1:0] = 0b00 表明 FIFO 小于 1/4 满或 1/4 空。
- COUNT[1:0] = 0b01 表明 FIFO 在 1/4 和 1/2 满之间或 1/4 和 1/2 空之间。
- COUNT[1:0] = 0b10 表明 FIFO 在 1/2 和 3/4 满之间或 1/2 和 3/4 空之间。
- COUNT[1:0] = 0b11 表明 FIFO 大于 3/4 满或 3/4 空。

DSP 对不同标志的反应取决于 FPGA 中读写 FIFO 的状态。

11.3.2　基于 TMS320 系列 DSP 的主机接口（HPI）设计

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

1．HPI 接口简介

HPI 接口是一种数据宽度为 16 位的并行端口（C64xx 系列 DSP 中，HPI 口的数据宽度达到 32 位）。通过 HPI 口，主机可以直接对 CPU 的存储器空间进行操作。

在 C621x、C671x 系列 DSP 中，没有留出专门的 EDMA 通道来执行 HPI 口的访问操作，而是直接将 HPI 口连接到内部的地址产生硬件上，因而提高了对内部存储空间的访问速度。

HPI 口内部加入了两个 8 级深度的读写缓冲，可以执行地址自增的读写操作，提高读写操作的吞吐量。HPI 口为内部 CPU 提供了标准 32 位的数据接口，同时为外部主机也提供了一个经济的 16 位接口，所以对外部主机而言，每次读写必须执行成对的 16 位操作。

对于 C64xx 系列 DSP，HPI 可以选择 16 位（HPI16）和 32 位（HPI32）两种模式。

对于 C621x、C671x 系列 DSP，HPI 口内部有 3 个寄存器，分别是控制寄存器（HPIC）、地址寄存器（HPIA）和数据寄存器（HPID）。这 3 个寄存器可以直接被主机访问，主机每执行一次对 CPU 内部存储空间的访问都必须先对控制寄存器和地址寄存器写入相应的值，然后才能对数据寄存器进行读写操作。

HPI 口的外部接口是由数据总线 HD[15:0]以及一部分用于描述和控制 HPI 接口的控制信号组成，这些控制信号的具体类型如表 11.4 所示。

表 11.4　　HPI 信号描述

引　脚	描　述
HCNTL[1:0]	控制 HPI 的操作类型
HHCNTL	半字确认输入，“0”、“1”分别表示一次字传输的第一个半字和第二个半字
HR/W	读/写选择

续表

引　脚	描　述
HRDY	就绪状态标志
HINT	中断标志，DSP 向主机提出中断
HAS	区别地址/数据复用总线的数据与地址
HDS1、HDS2、HCS	数据选通输入，三者配合可用于产生一个 HPI 内部选通信号 HSTROBE，这个信号可以用以下公式来描述： HSTROBE = [NOT（HDS1 XOR HDS2）] OR HCS

对于一个写 HPI 口的操作，应首先使能 HCS，变化 HDS1 或 HDS2，可使 HSTROBE 信号产生一个下降沿。HPI 口在这个下降沿采样控制信号 HCNTL[1:0]、HHWIL 和 HR/W，同时在使能 HCS 的同时驱动 HRDY，以使主机进入等待状态。直到 HRDY 产生下降沿，表明 HPID 已清空，可以接收新的数据。此时 HSTROBE 也将产生一个上升沿，并采样 HD[15:0]上的数据并将其送入 HPID，以完成第一个半字的写入。

对于第二个半字的写入，由于 32 位的 HPID 已经清空，可以直接写入数据。不会出现未准备好的情况，所以 HRDY 一直保持为低，与第一个字节的写入相同。该操作也在 HSTROBE 的下降沿采样控制信号，并在 HSTROBE 的上升沿采样数据总线 HD[15:0]的数据并送 HPID，以完成一个 32 位的写入操作。

2．FPGA 与 DSP 的 HPI 连接实例

FPGA 与 HPI 的接口连接比较简单，就是将 HPI 接口相关的信号全部连接到 FPGA 的管脚。如图 11.8 和图 11.9 所示分别是 FPGA 和 TI 不同系列的 DSP 的连接框图。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

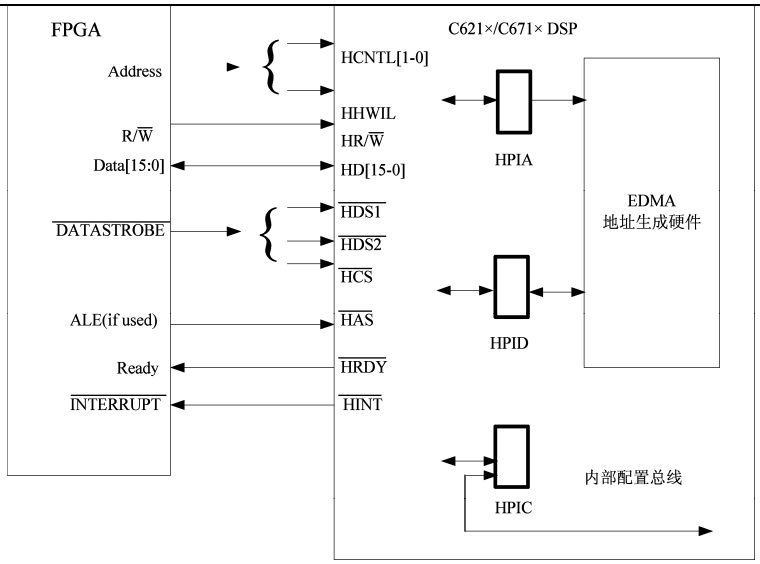


图 11.8　FPGA 与 C621x、C671x 系列 DSP 的 HPI 接口互联框图

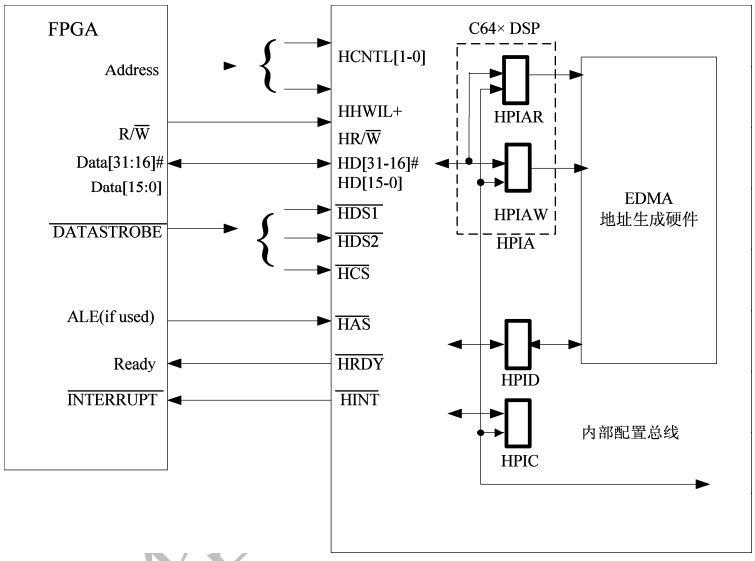


图 11.9　FPGA 与 C64x 系列 DSP 的 HPI 接口互联框图

11.4　FPGA+DSP 协同平台的调试技巧和注意事项

11.4.1　FPGA 和 DSP 的隔离调试技术

作为双芯片的协同系统，调试的开始阶段需要对每个芯片进行单独测试。这种情况下就需要避免另外一个芯片对调试产生影响，比较好的办法就是让它停止工作。

对于 FPGA 芯片，如果没有进行配置，那么所有的管脚都处于高阻状态。由于高阻态没有驱动能力，所以不会对 DSP 产生直接的影响。但是在某些情况下，会产生间接的影响。

例如，很多 DSP 会在系统复位的时候读取某些地址信号来确定启动后的工作模式。如果这些信号连接到了 FPGA 的管脚上，那么上电后这些信号就会处于不确定的状态。如果处于错误的电平状态，就会对 DSP 产生很大的影响。

解决的办法就是通过上拉或者下拉电阻来保证这些处于高阻态的信号处于正常的状态，如图 11.10 所示。这样，在 FPGA 的管脚处于高阻状态的时候，管脚会被上拉到高电平或者下拉到低电平。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

（a）上拉电路　　（b）下拉电路

图 11.10　双端口 RAM 模块

DSP 芯片和 FPGA 芯片不同，即使在没有进行配置的情况下，有些管脚也会处于输出状态，比如地址信号和控制信号。那么在单独调试 FPGA 芯片的时候，就要考虑到这些管脚是否会对 FPGA 产生影响。

特别是没有用到（unused）的 FPGA 管脚，有些系统默认的状态是低电平。这些管脚如果和 DSP 的输出管脚连接到一起，就会出现总线的冲突情况。因此，在工程项目设置的时候，要将 UNUSED PIN 指定为高阻态或者 INPUT 模式。

11.4.2　FPGA 测试点的设计

FPGA 芯片的管脚资源一般都很丰富，除了满足系统应用外，还会剩余部分管脚没有指定功能。这些剩余的管脚用来做测试点会方便系统的调试。

例如，DSP 的地址和数据信号都属于高速信号，如果直接在这些信号上加测试点测试会影响到信号的波形质量。特别是 BGA 封装的 DSP 芯片，这些管脚不能用示波器设备的探头或表笔直接测量。而且，板上的测试点太多也会影响布线的质量和美观。

充分利用 FPGA 的灵活性，可以将需要测试的信号指定到某个测试点上。这样就不需要直接测试这些信号点，而是通过测试点进行间接的测量。

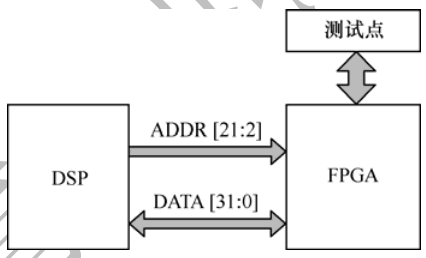


图 11.11　测试点在 FPGA＋DSP 系统中的作用

如图 11.11 所示，如果需要测试 DSP 的信号 ADDR[15] 和 DATA[0]，可以把这两个信号在 FPGA 内部指定到测试点 T1、T2。只要直接测试 T1、T2 就可以了。

```
    assign T1 = ADDR[15];  // 把 ADDR[15] 信号指定到测试点 T1
    assign T2 = DATA[0];   // 把 DATA[0] 信号指定到测试点 T2
```

此外，这些测试点还可以用来做功能扩展。因为这些测试点都是双向的普通 I/O，可以作为连接器信号和其他板卡或者系统互联。

11.4.3　借助 FPGA 的内部逻辑分析仪来辅助调试

在前面的章节中曾经介绍过 FPGA 的内部逻辑分析仪功能，如 Altera 公司的 Signal TAPII 和 Xilinx 公司的 Chip Scope 等。这些工具可以用来测试 FPGA 运行状态下信号的变化情况，特别是总线的运行状态。

在 FPGA+DSP 的系统中，DSP 的很多信号需要连接到 FPGA 的管脚。利用内部逻辑分析仪功能，就可以通过 FPGA 来观察这些信号的时序和状态，提高了调试的可视化程度。

11.5　典型实例 20：FPGA 片上硬件乘法器的使用

11.5.1　实例的内容及目标

1．实例的主要内容

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

在 FPGA+DSP 系统设计系统中，FPGA 经常作为 DSP 的协处理器来辅助完成一些计算任务。而这些计算工作中最消耗时间的就是乘法运算，因此本实例的主要内容就是帮助读者学会调用硬件乘法 IP 核。

2．实例目标

通过本实例，读者应达到以下目标。

- 了解硬件乘法器对算法的加速作用。
- 掌握 FPGA 片上硬件乘法器 IP 的调用方法。

11.5.2　硬件乘法 IP 的使用方法

1．生成硬件乘法器

如图 11.12 所示，在“New Source”里面选择“IP（CoreGen & Architecture Wizard）”，在右边的文件名里面输入“multiply”，单击“Next”按钮，打开如图 11.13 所示的 IP 核类型选择对话框。

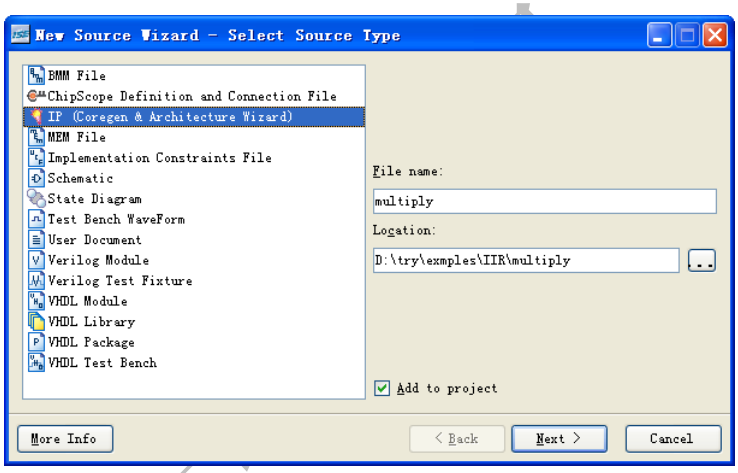


图 11.12　新建 IP 核文件

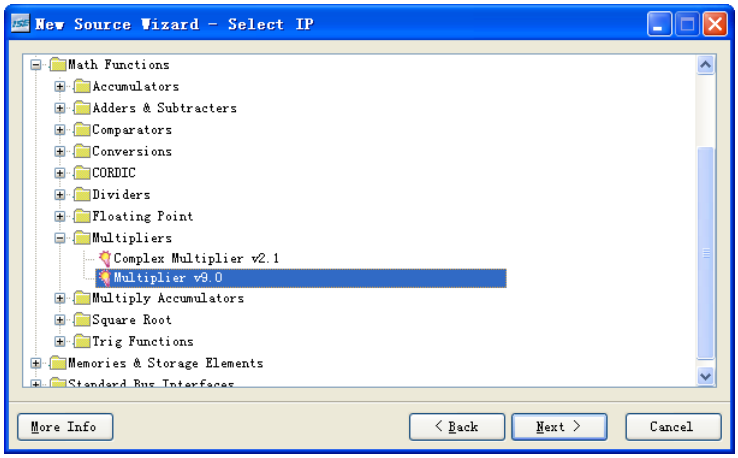


图 11.13　新建乘法器 IP 核

在 IP 核类型选择界面里面选择 Math Functions→Multipliers→Multiplier v7.0，单击“Next”按钮，打开乘法器 IP 核生成向导，如图 11.14 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

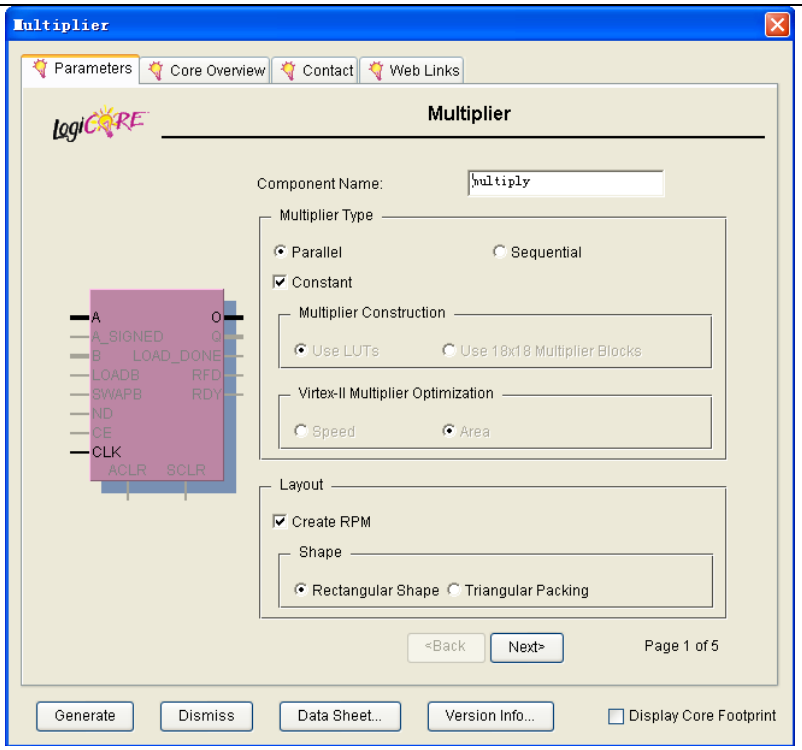


图 11.14　乘法器生成向导

在上面对话框里，“Component Name”是要生成的模块的名称。“Parallel”和“Sequential”两个选项是用来选择乘法器模式的，选择并行结构可以加快乘法器的速度，但是同时要多占用一些资源。选择“Constant”选项我们会发现乘法器的 B 输入端口被屏蔽掉了，这时 B 将作为一个固定输入。

如果不选择“Constant”选项，可以在下面的乘法器结构框里面选择乘法器使用的资源种类，选择“Use LUTs”将使用片上存储器资源，选择“Use 18*18 Multiplier Blocks”将使用 FPGA 自带的 DSP 模块。“Virtex-II Multiplier Optimization”框里面的选项只有在使用 Virtex-II 族器件的时候才能使用。

前面选择了 Constant 选项，在这个对话框里面就要对乘法器的 B 输入进行设置，在 Constant Value 后面输入 B 端口输入的数值（十进制），如图 11.15 所示。

在如图 11.16 所示的对话框里面设置输入口 A 的位宽和形式。

在如图 11.17 所示的对话框里面设置输出口的位宽和是否使用寄存器（使用输出寄存器的时候由 Q 口输出，不使用输出寄存器的时候由 O 口输出）。

在如图 11.18 所示的对话框里面选择“Minimum Pipelining”可以看见在最下面的“Information”框里面的输出延迟达到最小为 1。在“Register Option”栏里面可以设置同步复位，异步复位和时钟使能。

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

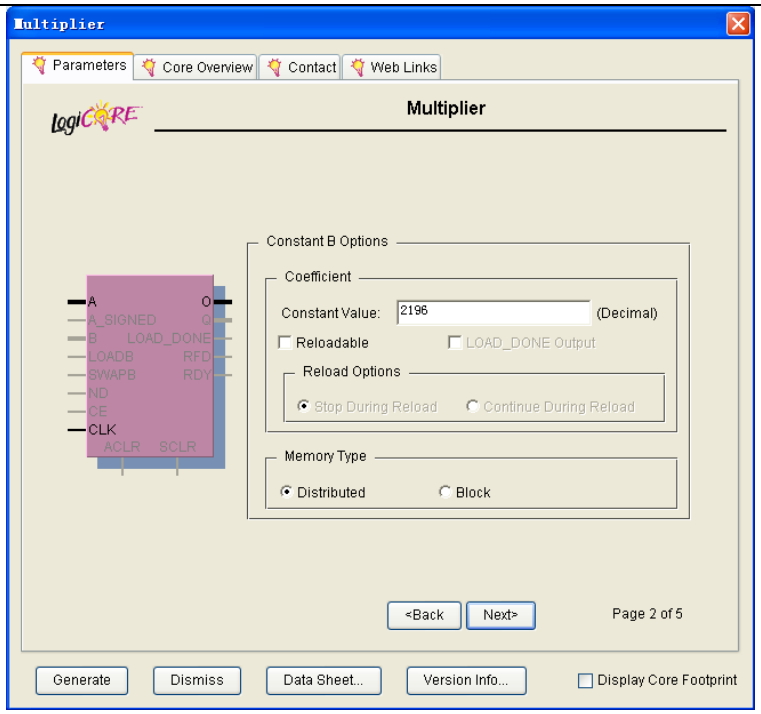


图 11.15　设置端口 B 参数

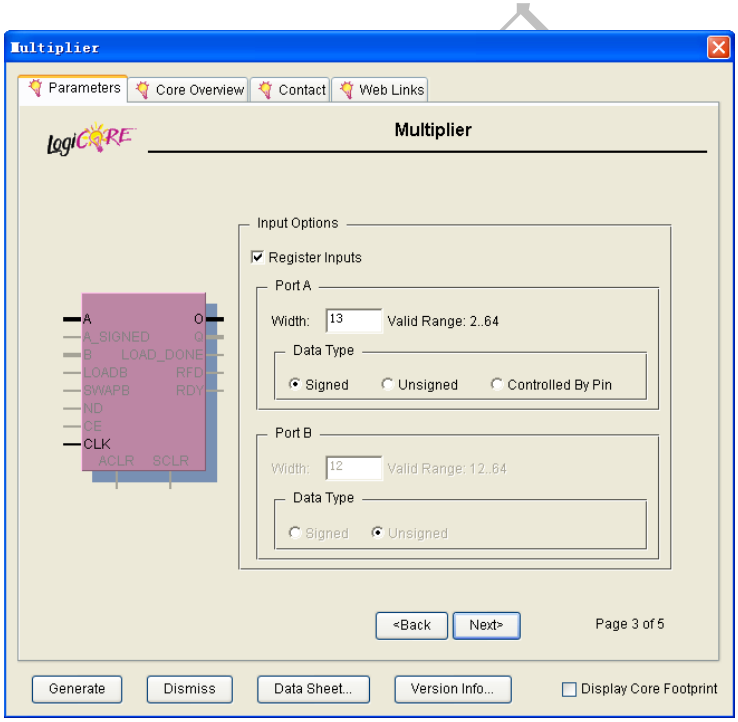


图 11.16　设置端口 A 参数

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

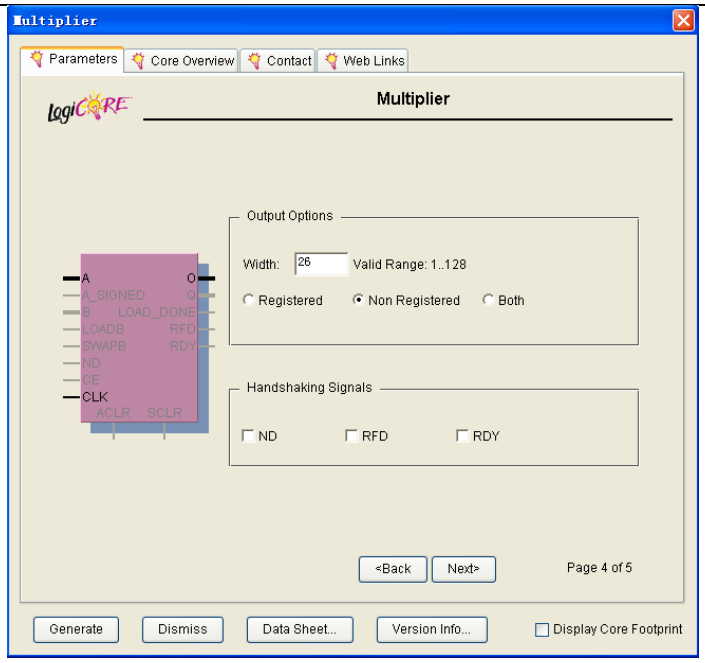


图 11.17　设置输出端口

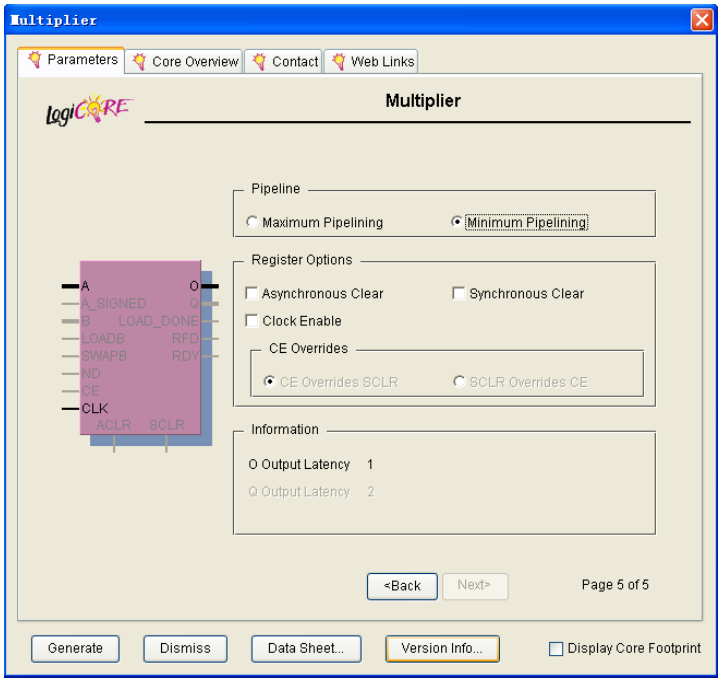


图 11.18　其他参数设置

配置完成以后单击“Generate”按钮就可以生成乘法器了。

2．使用硬件乘法器

本实例以一个 IIR 数字滤波的设计为例来演示硬件乘法器的使用方法。直接打开实例的工程文件，如图 11.19 所示。其中 IIR.v 是 IIR 滤波器的设计文件，test.tbw 为测试文件。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

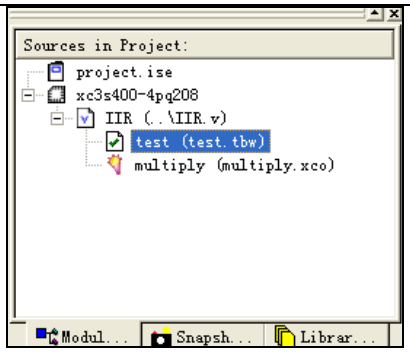


图 11.19　IIR 数字滤波器工程目录

在 IIR.v 的文件中，可以看到硬件乘法器的调用方法。

```
        reg     [12:0 ]       x1,x2,x3     ;     // 定义乘数 a
        reg     [12:0 ]       y1,y2,y3           ;
        reg     [12:0 ]       q1,q2,q3           ;


    wire    [25:0 ]       mul        [10:1 ]  ;    // 定义乘法的输出结果
    multiply m0(.clk(clk),.a(x1),.o(mul[1]));// 乘法器的调用，输入为 a，输出为 o
    multiply m1  ( .clk(clk),.a(x2),.o(mul[2] )  );
    multiply m2  ( .clk(clk),.a(x3),.o(mul[3] )  );
    multiply m3  ( .clk(clk),.a(y1),.o(mul[4] )  );
    multiply m4  ( .clk(clk),.a(y2),.o(mul[5] )  );
    multiply m5  ( .clk(clk),.a(y1),.o(mul[6] )  );
    multiply m6  ( .clk(clk),.a(y2),.o(mul[7] )  );
    multiply m7  ( .clk(clk),.a(y3),.o(mul[8] )  );
    multiply m8  ( .clk(clk),.a(q1),.o(mul[9] )  );
    multiply m9  ( .clk(clk),.a(q2),.o(mul[10])  );
```

运行仿真后，可以得到输出结果如图 11.20 所示。

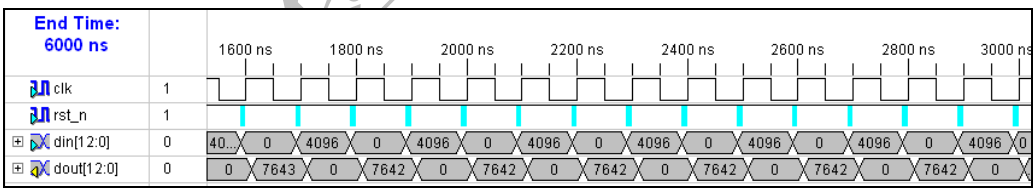


图 11.20　IIR 数字滤波器仿真结果

11.5.3　小结

本节对生成和使用 FPGA 的片上硬件乘法器 IP 核的方法做了介绍，并通过编译下载在红色飓风的开发板上实现了预定功能。

11.6　典型实例 21：整数 DCT 变换的设计与实现

11.6.1　实例的内容及目标

1．实例的主要内容

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

本节旨在设计实现了视频压缩标准 H.264 算法中的整数 DCT 变换部分，帮助读者了解并行流水设计技巧在算法优化中的作用。

2．实例目标

通过本实例，读者应达到以下目标。

- 了解数字域变换的基本原理和用处。
- 掌握 DCT 和整数 DCT 变换的变换方法。
- 学习硬件结构设计方法。
- 学习流水线设计方法。
- 编程实现 DCT 变换。

11.6.2 整数 DCT 变换的原理

1．DCT 简介

H.264 是一种图像压缩编码方法，它的变换编码和以前各种标准中的 DCT 有所不同。以前标准中直接采用 DCT 的定义进行变换，会带来两个问题。第一，需要进行浮点数操作，从而造成系统设计及运算上的复杂性；第二，由于变换核都是无理数，而有限精度的浮点数不可能精确地表示无理数，再加上浮点数的运算可能会引入舍入误差，这就使得在具体实现时会导致编解码的不匹配（mismatch），即反变换的输出结果和正变换的输入不一致。

为了解决这些问题，H.264 采用基于 4×4 块的整数操作而不是实数运算，使得变换操作仅用整数加减和移位操作就可以完成。这样既降低了设计复杂度，又避免了编解码误匹配，能够得到与 4×4 DCT 变化类似的编码效果，而由此带来的编码性能的减少微乎其微。

由于变换中无乘法，采用基于提升结构的无乘法二进制 DCT（Bin DCT），只包括加法和 16 位算术移位，这样大大减小了运算复杂度。尤其是对低端处理，减少了乘法运算且保持了整数变换的优点，精确的整数排除了编码器和解码器之间反变换的误匹配，保证了变换的效果。

2．DCT 设计原理

我们可以通过各种公式推导出整数 DCT 正变换的公式。

$$Y=\left(\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1/2 & -1/2 & -1\\ 1 & -1 & -1 & 1\\ 1/2 & -1 & 1 & -1/2\end{bmatrix} X \begin{bmatrix}1 & 1 & 1 & 1/2\\ 1 & 1/2 & -1 & -1\\ 1 & -1/2 & -1 & 1\\ 1 & -1 & 1 & -1/2\end{bmatrix}\right) \otimes \begin{bmatrix}a^2 & ab & a^2 & ab\\ ab & b^2 & ab & b^2\\ a^2 & ab & a^2 & ab\\ ab & b^2 & ab & b^2\end{bmatrix} \quad (11.1)$$

公式（11.1）中，虽然乘以 1/2 的操作可以用右移来实现，但这样会产生截断误差，因此，我们将 1/2 提到矩阵外面，并与右边的点乘合并，得到公式（11.2）。

$$\begin{aligned} Y &= (C_f X C_f^T) \otimes E_f \\ &= \left(\begin{bmatrix}1 & 1 & 1 & 1\\ 2 & 1 & -1 & -2\\ 1 & -1 & -1 & 1\\ 1 & -2 & 2 & -1\end{bmatrix} X \begin{bmatrix}1 & 2 & 1 & 1\\ 1 & 1 & -1 & -2\\ 1 & -1 & -1 & 2\\ 1 & -2 & 1 & -1\end{bmatrix}\right) \otimes \begin{bmatrix}a^2 & ab/2 & a^2 & ab/2\\ ab/2 & b^2/4 & ab/2 & b^2/4\\ a^2 & ab/2 & a^2 & ab/2\\ ab/2 & b^2/4 & ab/2 & b^2/4\end{bmatrix} \end{aligned} \quad (11.2)$$

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

这就是 H.264 中所用到的整数变换公式，其变换核 $C_f X C_f^T$ 仅用加减法（和左移）即可以实现，而后面的点乘操作可以合并到随后的量化过程中去。

H.264 中所用到的反 DCT 变换公式如下：

$$X = C_i^T (Y \otimes E_i) C_i = \begin{bmatrix} 1 & 1 & 1 & 1/2 \\ 1 & 1/2 & -1 & -1 \\ 1 & -1/2 & -1 & 1 \\ 1 & -1 & 1 & -1/2 \end{bmatrix} \left(Y \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix} \right) \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & -1 \\ 1 & -1 & -1 & 1 \\ 1/2 & -1 & 1 & -1/2 \end{bmatrix} \quad (11.3)$$

其中与 Y 点乘的操作与反量化合并，乘以系数 1/2 的操作由右移来实现，由于反量化后的结果足够地大，所以这里不会出现截断误差的问题。以上各式中，$a = \frac{1}{2}$，$b = \sqrt{\frac{2}{5}}$。

H.264 的整数 DCT 变换可以分做两步完成：先对需要做变换的矩阵的每一列做一维变换，再对其结果的每一行做一维变换，这个次序也可以反过来，先行后列。这样二维变换就可以用一维变换来实现。在具体实现过程中，为了减少运算量，每一步可以采用蝶型算法，以公式（11.2）的第一步对 X 的第一列进行一维变换为例，其运算过程如下式所示。

$$\begin{aligned} x'_0 &= x_1 + x_1 + x_2 + x_3 \\ x'_1 &= 2x_0 + x_1 - x_2 - 2x_3 \\ x'_2 &= x_0 - x_1 - x_2 + x_3 \\ x'_3 &= x_0 - 2x_1 + 2x_2 - x_3 \end{aligned} \quad (11.4)$$

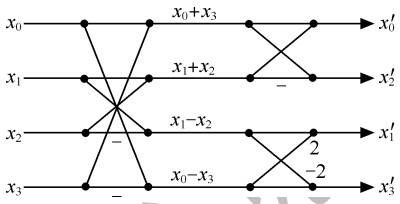


图 11.21 蝶形算法

其中 $x_n, n = 0,...,3$ 为 X 第一列的元素，$x'_n, n = 0,...,3$ 为滤波结果。由公式（11.4）可见计算有很多重复，如 $x_0 + x_3$ 就同时被计算 x'_0 和 x'_2 的公式所使用，所以可以将其暂时保存起来以避免重复计算，对应的蝶型算法如图 11.21 所示。

从图中可以看见，若按公式（11.4）计算需要进行 12 次加法、4 次乘法。而按图 11.19 中的蝶型算法仅需 8 次加法、2 次乘法，它利用了运算中的冗余，大大降低了运算量。

11.6.3 实例步骤

1．创建新工程并添加源文件

如图 11.22 所示，首先创建一个新工程并为工程添加源文件。

2．添加测试文件，并添加激励

如图 11.23 所示，为工程添加测试文件。

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

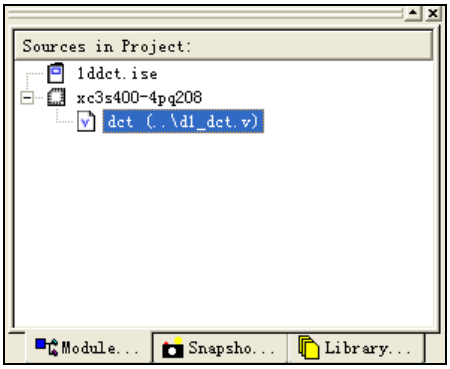


图 11.22　创建新工程并添加设计文件

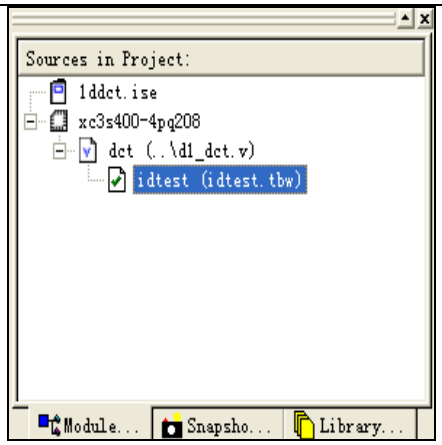


图 11.23　添加测试文件

在测试文件中为测试添加激励，如图 11.24 所示。

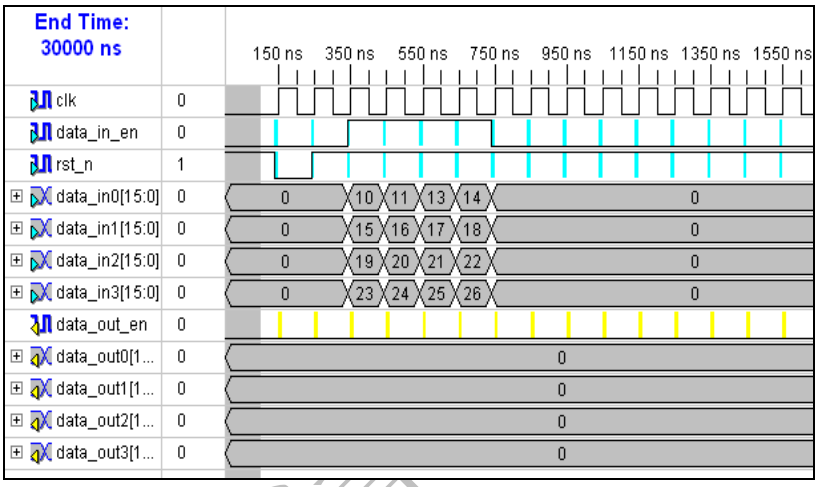
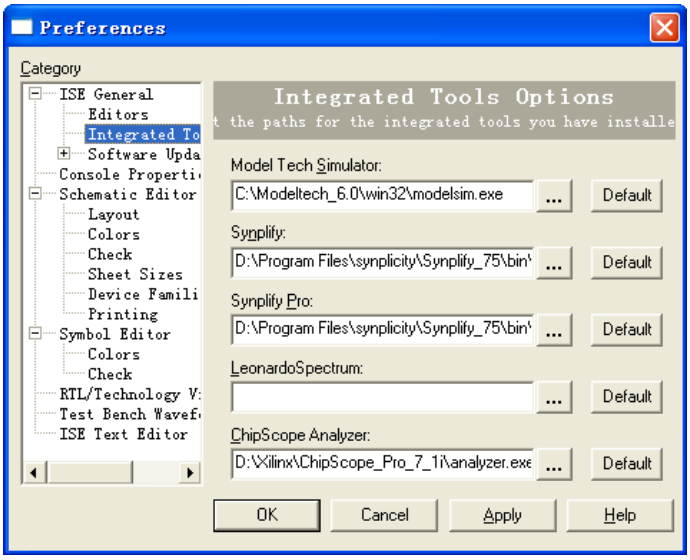


图 11.24　添加激励

3．使用 ModelSim 进行仿真

要使用 ModelSim 对工程进行仿真首先要在计算机上面安装 ModelSim 软件。安装好以后在 ISE 7.1i 的菜单里面选择 Edit→Preferences，如图 11.25 所示。

在上面的对话框里面，“Integrated Tools”复选页里面的 Model Tech Simulator 里面选择 modelsim.exe 的路径，单击“OK”按钮。



华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

图 11.25　设置仿真工具参数

然后再选择菜单里面的 View→Refresh，刷新刚才更改的设置，如图 11.26 所示。

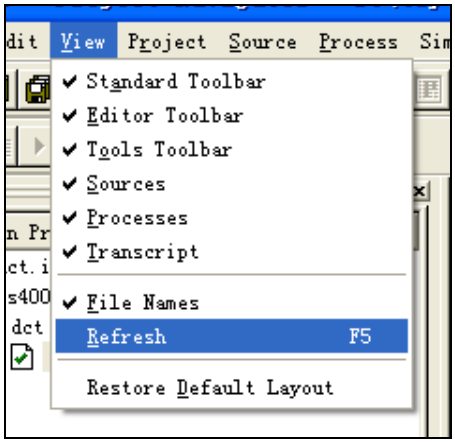


图 11.26　刷新设置

刷新后就会看见 Process View 里面的图标变成 Modelsim Simulator 的一些功能，如图 11.27 所示。

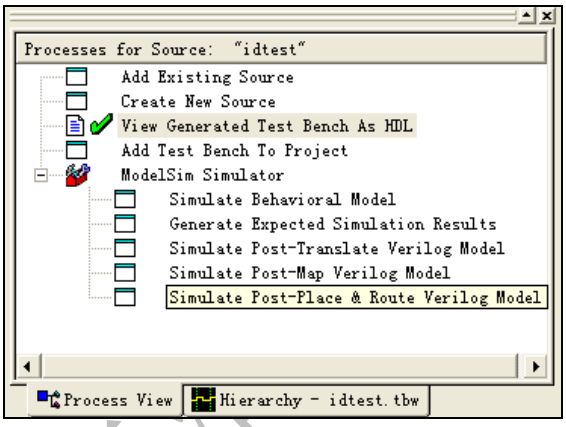


图 11.27　ModelSim 仿真选项

此时双击“Simulate Behavioral Model”就可以对工程进行行为仿真了，如图 11.28 所示。

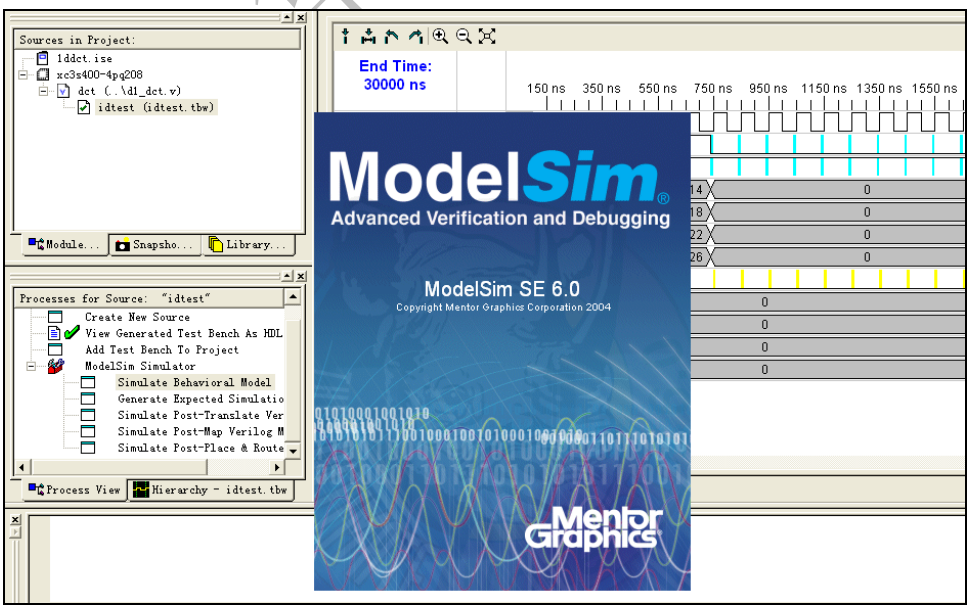


图 11.28　打开 ModelSim 进行行为仿真

在 ModelSim 的波形窗口中观察到的行为仿真结果如图 11.29 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

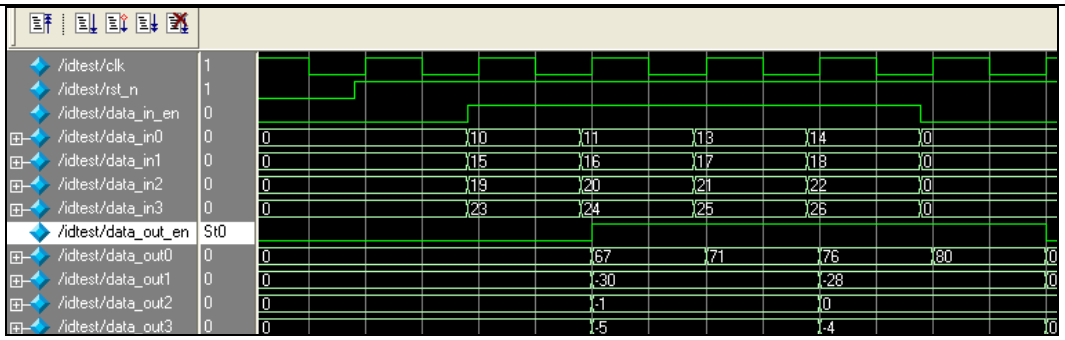


图 11.29　行为仿真结果

11.6.4　小结

本节介绍视频压缩标准 H.264 算法中的整数 DCT 变换模块的设计与实现方法，并通过 ModelSim 软件仿真验证的设计结果。

联系方式

集团官网：www.hqyj.com　　嵌入式学院：www.embedu.org　　移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn　　物联网学院：www.topsight.cn　　研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路银海大厦 A 座 8 层，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

华清远见 FARSIGHT 始于2004 | 嵌入式培训专家

10年口碑积累，成功培养50000多名研发工程师，铸就专业品牌形象

华清远见的企业理念是不仅要做良心教育、做专业教育，更要做受人尊敬的职业教育。

《FPGA 应用开发入门与典型实例》（修订版）

作者：华清远见

专业始于专注 卓识源于远见

第 12 章 数字图像倍焦系统设计与实现综合实例

本章目标

- 掌握视频信号采集和输出系统的设计方法
- 掌握视频编/解码芯片的配置方法
- 了解 FPGA 系统的优化流程和方法

专业始于专注 卓识源于远见

开发板免费领取试用 开发板免费领取试用

12.1 设计需求分析与芯片选型

在数字图像处理和通信、遥感图像分析、医学成像诊断等应用领域，为了便于显示、观察或进行进一步的处理，常常需要对原始的数字图像进行特征提取(如边缘检测、边缘锐化)、噪声平滑滤波、几何校正、尺寸缩放等处理，这类图像处理技术称为图像的预处理。在实际应用中，图像的预处理功能很多可以通过FPGA 来实现。

基于 FPGA 的图像处理系统最大的优点就是灵活性，可以实现高度集成性，缩小产品的体积。而且，可以实现创新性的设计。

本系统要实现的功能是对光学或者红外摄像机器输出的视频信号（模拟）进行数字倍焦处理。这样可以实现焦距拉长一倍的光学效果，然后仍然以原有的信号格式输出，送给显示设备。

12.1.1 需求分析

下面讲解数字图像倍焦系统的输入输出特性及系统指标。

1．系统输入信号指标

（1）视频输入制式：PAL 制式。扫描方式为隔行扫描。
（2）同步模式：内部同步。
（3）电气特性：峰峰值为 1.0V，阻抗为 75Ω，BNC 接口。
（4）分辨率：570 线 TV 标准，数字化后为 720×576，每秒 25 帧。

2．系统输出信号指标

（1）视频输出制式：PAL 或者 NTSC 可选，可以通过开关来切换。扫描方式为隔行扫描。
（2）同步模式：内部同步或者外部同步。
（3）电气特性：峰峰值为 1.0V，阻抗为 75Ω，BNC 接口。
（4）分辨率：570 线 TV 标准，模拟化之前为 720×576，每秒 25 帧。

3．其他系统指标

（1）电路板体积：长×宽×高小于 40mm×40mm×20mm。
（2）电路板供电和功耗：单＋5V 供电，功耗小于 1W。

12.1.2 芯片选型

本系统的主要功能是完成图像的实时插值放大算法，同时对体积和功耗有严格的要求。根据上面提出的指标进行如下的设计分析。

1．主芯片选型分析

对于放大算法实现，主芯片可以选择 FPGA、DSP 和 ARM 三种方案。

基于 ARM 的系统不能和视频的编解码芯片直接互联，需要外围器件较多，电路体积会比较大。基于 DSP 的系统虽然能和视频的编解码芯片直接互联，体积可以满足要求，但是功耗会超标。因此，基于 FPGA 的

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

方案是最佳的选择。
由于本系统要求能够扩展 5V 的数字 I/O，所以 FPGA 要能够兼容 5V 的信号输入。因此，选择 Altera 公司的 ACEX1K 系列 FPGA 器件。这个系列的芯片 IO 电压是 3.3V 的，但是输入支持 5V 电平。

2．信号输入输出选型分析

在视频信号处理领域，把模拟视频信号转换成数字视频信号的芯片称为视频解码器（Video Decoder），如 SAA711x 系列。而把数字视频信号转换成模拟视频信号的器件称为视频编码器（Video Encoder）。
本系统中，视频解码器选择了应用广泛的 SAA7113H，视频编码器选择了 SAA7128H。

3．其他单元电路选型分析

另外，为了满足图像插值放大算法要求，需要在信号采集系统中采样缓冲区。采集的信号速率越高，需要的缓冲区容量越大，以保证不丢失数据。
本系统中，采用了乒乓缓冲区的策略来完成数据的输入输出缓冲区管理。因此，选用了两片 SRAM 类型的存储器。相对于 SDRAM 器件，SRAM 器件的控制方法更为简单，功耗也相对较低，可以满足本系统的要求。
电源部分选择了电路简单的线性电源模块（LDO）。本系统属于低功耗系统，工作电流比较小，因此采用 LDO 芯片比较适合，如 AS1117 系列芯片。

4．电路板结构设计

本系统对体积有严格要求，如果采用标准单板设计很难达到设计要求，因此，考虑层叠式的电路结构。将整个电路分成两块电路板，通过连接器，层叠式安装。这种结构可以充分利用空间，适合这种长宽要求严格，但是高度方面相对宽松的系统。

12.2　系统工作原理分析

如前所述，本系统主要完成对输入视频图像的两倍放大。图像的放大主要是通过插值算法来实现的，下面详细分析如何应用双线性插值算法来实现倍焦功能。

1．符号约定

首先对系统工作原理分析中用到的符号做一个约定：
(i, j) 表示像素点坐标，在本系统中$1 \leqslant i \leqslant 576, 1 \leqslant j \leqslant 720$ 。
$I_p(x, y)$ 表示原始图像，其宽度为$WIDTH_p$，高度为$HEIGHT_p$。
$I_{new}(x, y)$ 表示倍焦后图像，其宽度为$WIDTH_{new}$，高度为$HEIGHT_{new}$。
通过双线性插值完成数字倍焦功能，即将原始图像插值为原来 2 倍大小，仍然以原来的分辨率输出，即$WIDTH_p \times HEIGHT_p$。

2．双线性插值原理描述

华清远见
华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

对新图像中每一点为 (i,j)，在本系统中 $1\leqslant i\leqslant 576, 1\leqslant j\leqslant 720$，查找其在原图像中的相邻位置 (p,q)、$(p,q+1)$、$(p+1,q)$、$(p+1$、$q+1)$。

(i,j) 点的灰度值 $d(i,j)$ 的计算公式为（12-1）。

$$\begin{aligned} d_p(i,j) &= (1-w_q)d(p,q) + w_q d(p,q+1) \\ d_{p+1}(i,j) &= (1-w_q)d(p+1,q) + w_q d(p+1,q+1) \\ d(i,j) &= (1-w_p)d_p(i,j) + w_p d_{p+1}(i,j) \end{aligned} \qquad (12.1)$$

公式（12-1）中需要计算的数值为 p、q、w_p、w_q，如公式（12-2）～（12-5）所示。

$$p = \left\lfloor \frac{i}{\sqrt{2}} \right\rfloor \approx \left\lfloor \frac{i}{256/181} \right\rfloor = (i\times 181) >> 8 \qquad (12.2)$$

$$q = \left\lfloor \frac{j}{\sqrt{2}} \right\rfloor \approx \left\lfloor \frac{j}{256/181} \right\rfloor = (j\times 181) >> 8 \qquad (12.3)$$

$$w_p = \frac{i}{\sqrt{2}} - \left\lfloor \frac{i}{\sqrt{2}} \right\rfloor \approx \frac{i(\bmod 256)}{\sqrt{2}} - \left\lfloor \frac{i(\bmod 256)}{\sqrt{2}} \right\rfloor \to f\left(i(\bmod 256)\right) = f\left((i\,\&\,255)\right) \qquad (12.4)$$

$$w_q = \frac{j}{\sqrt{2}} - \left\lfloor \frac{j}{\sqrt{2}} \right\rfloor \approx \frac{j(\bmod 256)}{\sqrt{2}} - \left\lfloor \frac{j(\bmod 256)}{\sqrt{2}} \right\rfloor \to f\left(j(\bmod 256)\right) = f\left((i\,\&\,255)\right) \qquad (12.5)$$

3．硬件实现的简化

为减少硬件计算的复杂性，对 p、q、w_p、w_q 的算法做如下处理。

（1）硬件乘法器是必须的，用以完成 $(i\times 181)$ 和 $(j\times 181)$ 的计算，但由于：

$$1\leqslant i\leqslant 576, 1\leqslant j\leqslant 720, \max(i,j) < 256\times 3$$

所以可以考虑只用 8bit×8bit 硬件乘法器（而不是 16bit×8bit 硬件乘法器），即如公式（12-6）所示。

$$\begin{aligned} i\times 181 &= (i\,\&\,255)\times 181 + \left(((i >> 8)\times 181) << 8\right) \\ &= \begin{cases} (i\,\&\,255)\times 181 & if\ (i >> 8) == 0 \\ (i\,\&\,255)\times 181 + (181 << 8) & if\ (i >> 8) == 1 \\ (i\,\&\,255)\times 181 + (181 << 8) + (181 << 8) & if\ (i >> 8) == 2 \end{cases} \end{aligned} \qquad \text{公式（12-6）}$$

（2）求权重 w_p、w_q 的公式已经写成关于 $(i\,\&\,255)$、$(j\,\&\,255)$ 的函数的形式，该函数可以通过离线计算并存储一张表格（“权重表”）的方法实现。该表的大小为 256×8bit。注意求 w_p、w_q 查询的是同样的表格。

4．算法优化

（1）乘法器设计的简化。

在 p、q 的计算过程中，需要一个常数乘法器(×181)，通过查表完成。而 w_p、w_q 也需要查表，查表也需要一个周期来完成。如果用同一个表，那么 p、q、w_p、w_q 需要串行查出，速度太慢。如果要同时查出来，需要多个表，片内存储器太大。

因此，在计算过程中需要使用移位和加法进行配合，来提高运算速度。

基于这个考虑，将放到 2 倍的条件放宽到 2.25 倍，即用 1.5×1.5 来替代 1.414×1.414。这样，公式（12-2）～（12-5）就简化为公式（12-7）～（12-10）的形式。

$$p = \left\lfloor \frac{i}{3/2} \right\rfloor = \left\lfloor \frac{2i}{3} \right\rfloor \qquad \text{公式（12-7）}$$

$$q = \left\lfloor \frac{j}{3/2} \right\rfloor = \left\lfloor \frac{2j}{3} \right\rfloor \qquad \text{公式（12-8）}$$

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

$$w_p = \frac{i}{3/2} - \left\lfloor \frac{i}{3/2} \right\rfloor = \frac{(2i) \bmod 256}{3} - \left\lfloor \frac{(2i) \bmod 256}{3} \right\rfloor = \left\{ \frac{1}{3}, \frac{2}{3} \right\} \quad \text{公式（12-9）}$$

$$w_q = \frac{j}{3/2} - \left\lfloor \frac{j}{3/2} \right\rfloor = \frac{(2j) \bmod 256}{3} - \left\lfloor \frac{(2j) \bmod 256}{3} \right\rfloor = \left\{ \frac{1}{3}, \frac{2}{3} \right\} \quad \text{公式（12-10）}$$

这样，w_p、w_q 不必查表产生，而是 1/3 和 2/3 交替。关于除 3 的操作，可以通过一个 256×8bit 的查找表来完成。

需要说明的是，经过上面的调整，放大的倍数为 2.25 倍，而不是 2 倍。

（2）行列的处理顺序的优化。

原算法中水平和垂直方向同时插值，需要 4 个像素类计算。但是，实际硬件实现过程中如果读取 4 个数据来计算一个像素点，那么输出很难保证连续，而且速度太慢。因此，实现过程中，对整场图像先做行插值，再做列插值。即将二维线性插值转为两个一维线性插值，如公式（12-11）所示。

$$d(i,j) = (1 - w_p)d(p,q) + w_p d(p,q+1) \quad \text{公式（12-11）}$$

其中 w_p 依次取 1/3 和 2/3。这样每次插值操作只与两个像素值有关，一个是前一个像素（寄存器中），另一个是当前读出的像素。

当接收原始帧时，在裁剪后写入 SRAM 过程中完成行插值；从 SRAM 读出过程完成列插值，并输出。

考虑到视频信号是按行扫描的，而且是隔行扫描，因此处理过程以场为单位。

（3）硬件实现简化前后的结果对比。

从图 12.1 和 12.2 的对比图中可以看出，放大 2.25 倍的图像视野上要比放大 2 倍的图像小一些，但是图像质量并没有明显降低。

图 12.1　放大 2 倍后的图像

图 12.2　放大 2.25 倍后的图像

12.3　系统原理框图

本系统采用了两片 SRAM 存储器构成乒乓缓冲结构来完成视频的采集和处理，视频的采集和输出部分采用了应用广泛的 SAA71XX 系列。

数字图像倍焦系统的原理框图如图 12.3 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

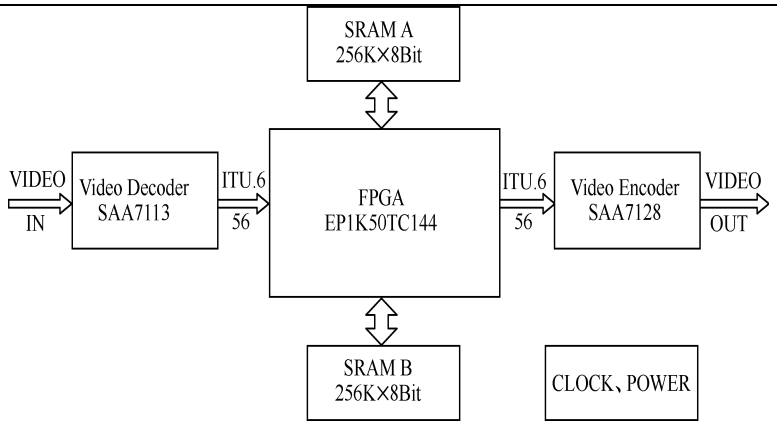


图 12.3　数字倍焦系统原理框图

系统框图各部分模块介绍如下

（1）Video Decoder 模块。

视频输入是 PAL 格式的，利用 Video Decoder 芯片可以将 PAL 格式的模拟视频信号转换成 ITU656 格式的数字视频信号。ITU656 信号就可以直接送入 FPGA 进行采集和处理了。

（2）Video Encoder 模块。

视频输出是 PAL 格式的，利用 Video Encoder 芯片可以将 ITU656 格式的数字视频信号转换成 PAL 格式的模拟视频信号。

（3）FPGA 模块。

FPGA 是系统的核心部分，完成所有数字信号的处理功能，包括图像的采集、倍焦放大以及输出等。本系统采用了 Altera 公司的 ACEX1K 系列的 EP1K50 芯片。

（4）SRAM 模块。

SRAM 作为系统的缓冲区来完成图像的暂存功能。SRAM 需要缓存行变换后的结果，每场图像尺寸为 720×288，行变换后为 720×192＝138240，近似为 139KB。因此，用一片 256KB 的 SRAM 就可以缓存一场行变换结果。

（5）CLOCK、POWER 模块。

CLKOCK 模块给系统提供时钟电路，采用晶体振荡器来实现。POWER 模块为系统提供电源。系统输入为 5V，POWER 模块可以将输入的 5V 信号转换为 3.3V 和 2.5V 电源。

下面来详细介绍 FPGA 的内部结构设计。

12.4　FPGA 内部结构设计

12.4.1　FPGA 内部结构框图

本系统的核心设计部分是 FPGA 内部结构的逻辑设计。整个系统分成两个数据通路。

1．行变换数据通路

行变换数据通路主要功能是实现数据的采集、缓存、行变换和保存行变换结果。

2．列变换数据通路

行变换数据通路主要功能是实现读取行变换结果、缓存、列变换和图像输出。

通过总线开关模块实现两个数据通路的数据交换，采用的主要措施就是利用两片 SRAM 存储器来完成乒乓缓存。FPGA 内部结构如图 12.4 所示。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

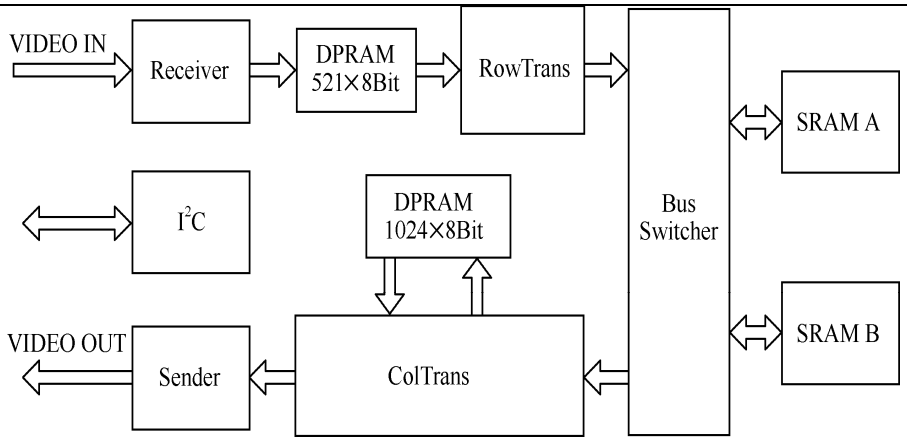


图 12.4　FPGA 内部结构框图

下面分别介绍数字图像倍焦系统中 FPGA 内部各个模块功能。

12.4.2　各个模块功能描述

在实际的设计中，主要通过自行编写模块、调用宏模块和引用开源模块 3 种方式来实现模块的设计。下面是各个模块的主要功能。

1．Receiver 模块功能

完成对 SAA7113 输出的 ITU656 格式视频信号的逐行采集，同时完成图像的裁剪工作，即原始图像的 1/2.25，分辨率为 480×192，并写入 DPRAM 中。每写完一行，给出一个行启动信号，每写完一场，给出场启动信号，均发送给 RowTrans 模块。

2．RowTrans 模块功能

完成对每行数据的线性插值。收到行启动信号之后，从 DPRAM 读出数据，计算插值后，通过总线开关写入 SRAM 中。处理之后 RAM 中的数据为 720 列×192 行。当收到场启动信号，立即切换 SRAM，开始处理下一场。

3．Bus Switcher 模块功能

总线开关模块，负责切换两片 SRAM 的控制权。

4．ColTrans 模块功能

完成从 SRAM 中读出每行数据，并写入 DPRAM 中，同时完成对每列的插值。保证 DPRAM 不读空，这样输出的码率才会恒定，满足 ITU.656 的要求。

5．Sender 模块功能

从 DPRAM 连续读出数据，按照 ITU.656 的要求发送给 Video Encoder。

6．DPRAM 模块功能

开发工具软件提供的双口 RAM 模型，用来完成数据缓存。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

12.5 系统硬件配置方案

12.5.1 FPGA 的配置

FPGA 作可编程器件，可以根据用户的需要进行现场可编程。系统可采用了 JTAG 模式和 AS 模式进行 FPGA 编程配置。

JTAG 模式编程直接对 FPGA 进行编程，使用 JTAG 模式，在 Quartus II 工具中输出 sof 文件（SRAM Object File）。其好处是编程速度快，并且由于是对 FPGA 的 SRAM 结构进行编程，编程次数要多得多；但是掉电后，SRAM 保存的编程信息将会丢失。

当程序调试完毕，需要固化的时候，可以通过 AS 模式将最终文件（pof 文件）烧写到配置芯片里面。

本系统采用的配置芯片是 EPC1441PC8。配置芯片具有掉电保存能力，在系统上电时，FPGA 首先从配置芯片中读取编程数据，并对 FPGA 进行加载。

12.5.2 Video Decoder（SAA7113H）的配置

系统上电后，Video Decoder 芯片正常工作之前是需要进行配置的。所谓的配置，就是通过 I^2C 接口对它的内部寄存器进行读写。配置的主要目的是通知 Video Decoder 系统当前采用的信号输入端口、格式和特殊要求。

本系统的工作要求是：1 路 PAL 制式视频，AI11 管脚输入，输出为 ITU656 格式，YUV 比例为 4：2：2。对应的 SAA7113H 的寄存器配置如表 12.1 所示。

表 12.1 SAA7113H 寄存器配置值

地址	内容	地址	内容	地址	内容	地址	内容	地址	内容	地址	内容
0	00	10	38	20	00	30	00	40	02	50	FF
1	08	11	0C	21	00	31	00	41	FF	51	FF
2	C0	12	E8	22	00	32	00	42	FF	52	FF
3	33	13	00	23	00	33	00	43	FF	53	FF
4	00	14	00	24	00	34	00	44	FF	54	FF
5	00	15	00	25	00	35	00	45	FF	55	FF
6	E0	16	00	26	00	36	00	46	FF	56	FF
7	0D	17	00	27	00	37	00	47	FF	57	FF
8	B8	18	00	28	00	38	00	48	FF	58	00
9	01	19	00	29	00	39	00	49	FF	59	00
A	80	1A	00	2A	00	3A	00	4A	FF	5A	07
B	47	1B	00	2B	00	3B	00	4B	FF	5B	80
C	40	1C	00	2C	00	3C	00	4C	FF	5C	00
D	00	1D	00	2D	00	3D	00	4D	FF	5D	00
E	01	1E	00	2E	00	3E	00	4E	FF	5E	00
F	2A	1F	00	2F	00	3F	00	4F	FF	5F	00

12.5.3 Video Encoder （SAA7128）的配置

系统上电后，Video Encoder 芯片正常工作之前是需要进行配置的。所谓的配置，就是通过 I^2C 接口对它的内部寄存器进行读写。配置的主要目的是通知 Video Encoder 系统当前采用的信号输出端口、格式和特殊要求。

本系统的工作要求是：1 路 PAL 制式视频输出，CVBS 管脚输出，输入为 ITU656 格式，YUV 比例为 4：2：2。对应的 SAA7128H 的寄存器配置如表 12.2 所示。

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

表 12.2 SAA7128H 寄存器配置值

地址	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
内容	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
地址	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
内容	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

续表

地址	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
内容	00	00	00	00	00	00	0D	00	1F	1F	75	3F	06	3F	00	00
地址	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
内容	00	00	00	00	00	00	00	00	1A	1A	13	00	00	00	00	00
地址	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
内容	00	00	00	68	10	97	4C	18	9B	93	9F	FF	7C	34	3F	17
地址	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
内容	00	83	83	80	8C	0F	C3	06	02	80	34	7D	AF	2D	3F	3F
地址	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
内容	00	02	46	CB	8A	09	2A	77	41	88	41	12	02	20	A0	B4
地址	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
内容	41	C3	00	3E	B8	1E	15	16	15	16	2B	D9	80	00	00	00

12.5.4 印刷电路板（PCB）设计

本系统采用了层叠式的电路板结构，两块电路板通过双排连接器组合成一个系统。其中，FPGA 相关电路在一块板上，视频的编解码器在另外一块板上。

传输一路数字视频信号需要 9 个必须的 ITU656 信号（DATA0～DATA7，CLOCK）和两个可选的同步信号（HSync，VSync）。另外，I^2C 配置接口还需要 2 个信号（SCL，SDA），加上 1 个电源信号，2 个地信号等，一共 16 个信号。

经过布局布线后，得到两个板子的版图如图 12.5 所示。

将这个版图文件提交给制版厂就可以加工设计了。

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

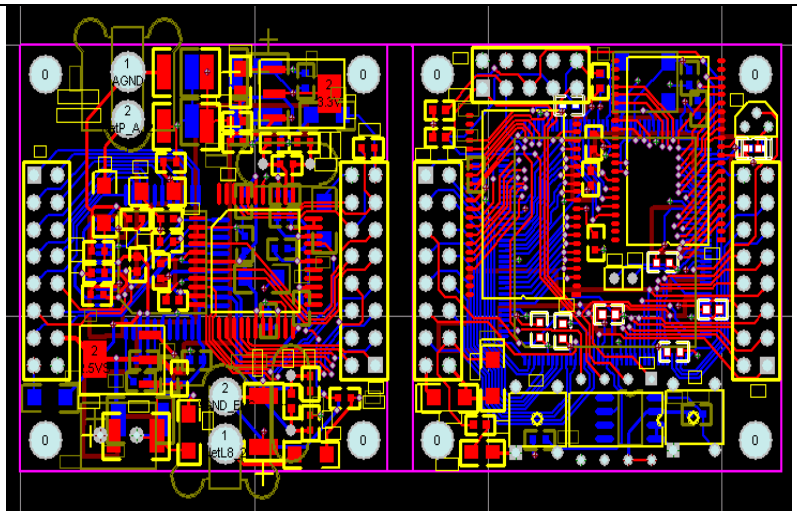


图 12.5　数字图像倍焦系统 PCB

12.6　FPGA 在其他视频和图像处理系统中的应用

目前，FPGA 已经广泛应用于视频和图像的很多领域领域，如在广播基础设施、HDTV 显示器、医疗成像、HDTV 视频会议以及视频 DVR、摄像机还有一些军事成像领域。

总的来看，视频和图像处理器的发展趋势集中于以下几方面：第一，越来越高的图像采集和显示分辨率；第二，新的压缩技术不断发展；第三，很多系统对智能方面的要求日益广泛。

1．视频广播领域

FPGA 在广播中的一些典型应用包括：音频和视频的采集、内容的准备、前处理和后处理、视频监控存储和基础设施以及内容交互、数字影院等。

2．视频监控领域

当前视频监控已经转到数字系统，相比模拟系统，数字系统更灵活、图像质量更高，并能利用 FPGA 或 DSP 实现图像的预处理和后处理，提高系统图像质量，增加智能性算法。

3．医疗成像领域

医疗成像领域也需要大量的视频处理。Altera 器件在医疗成像领域有很多成功的应用，使用 FPGA 可以提高卫生保健系统的性能和工作效率。

FPGA 在医疗成像领域的应用主要几种在以下几点：图像分析、模式识别、图像增强和修复及图像和数据压缩。FPGA 也可用于实现一些算法，如小波变换、色彩空间转换等。上述所有应用都可以通过使用 FPGA 提高系统效率。

联系方式

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

集团官网：www.hqyj.com 嵌入式学院：www.embedu.org 移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn 物联网学院：www.topsight.cn 研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路银海大厦 A 座 8 层，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218

华清远见

开发板免费领取试用 开发板免费领取试用

华清远见 FARSIGHT 始于2004 | 嵌入式培训专家

10年口碑积累，成功培养50000多名研发工程师，铸就专业品牌形象

华清远见的企业理念是不仅要做良心教育、做专业教育，更要做受人尊敬的职业教育。

《FPGA 应用开发入门与典型实例》（修订版）

作者：华清远见

专业始于专注 卓识源于远见

第 13 章 高速 PCI 信号采集卡设计与实现综合实例

本章目标

- 掌握信号采集系统设计的基本原则
- 掌握 PCI9054 芯片的接口设计
- 掌握 PCI 信号采集卡的驱动设计
- 掌握 PCI 信号采集卡的 FPGA 核心设计
- 掌握 PCI 信号采集卡的调试方法

专业始于专注 卓识源于远见

开发板免费领取试用 开发板免费领取试用

13.1 设计需求分析与功能定义

为了更好地分析信号采集系统的设计需求，获得正确的系统功能定义，首先来分析信号及信号采集系统的基本构成。

13.1.1 信号与信号采集系统

信号一般可以分为模拟信号和数字信号，在信号采集中常常称为模拟量和数字量。根据信号存在形式的差异，又可以将模拟信号分为直流信号、时域信号和频域信号，将数字信号分为开关信号和脉冲序列。
信号采集系统是将目标信号进行采集、处理并存储，形成计算机可以处理的数据格式，并最终进行各种形式输出的系统。典型的信号采集系统主要包含 3 个部分，即信号输入单元、信号处理单元和信号输出单元。如图 13.1 所示为典型的信号采集系统的框图。

1．信号输入单元

信号输入单元是接收信号源并将信号进行初步处理，形成数字信号的单元。对于模拟信号，一般需要进行信号调理及 A/D 转换。对于数字信号，一般进行数据宽度、时钟域变换等处理。

2．信号处理单元

信号处理单元是将信号输入单元输出的数字信号进行一系列处理的单元。一般使用 CPLD/FPGA、DSP 或者 ASIC 等单元进行信号处理。根据信号形式的不同，信号处理需要完成的工作也大不相同。

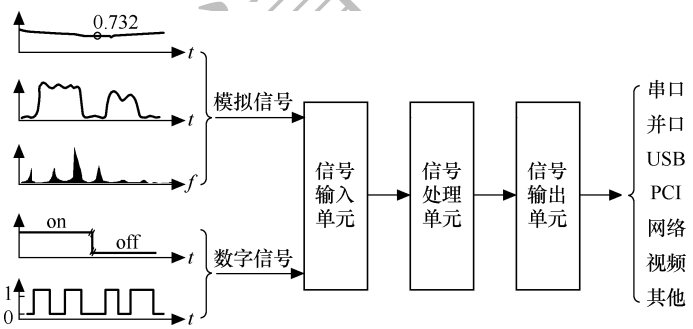


图 13.1　典型信号采集系统框图

3．信号输出单元

信号输出单元是将信号处理单元处理后的信号通过各种接口输出。一般可以通过 USB、网络、ISA 或者 PCI 总线等接口输出处理后的信号。

4．其他单元

此外，为了满足系统稳定运行，一个信号采集系统往往还包含数据缓冲区、时钟以及电源等相关的系统。

13.1.2 设计需求分析

开发板免费领取试用　开发板免费领取试用

更快、更准确地获取及处理尽量多的数据是信号采集系统一直追求的目标。随着传感器技术和信号处理技术的发展，高速传输和实时处理成为信号采集系统面临的新课题，也对典型信号采集系统的 3 个组成部分提出了更高的要求。

1．信号采集系统面临的新课题

高速传输主要影响的是信号输入和输出单元。对于数字信号，就是更高的输入频率和更多的输入通道数，对于模拟信号来说则意味着更高的分辨率，更高的采样频率以及更高的精度。同时高速传输还要求信号采集系统具有持续的以及突发的数据输出能力。

实时处理主要影响的是信号处理单元。它要求信号采集系统能够具备一定的数字信号快速处理能力，以便实时地为不同的输出接口提供合适的数据格式。由于实时处理是信号采集过程的一个中间环节，因此实时处理的延迟不能造成信号采集的中断，这也对信号处理单元提出了更高的要求。

2．本系统设计需求分析

在本案例中，信号采集系统需要采集的信号是数字信号。在高速信号传输中数字信号又常常以差分的形式存在，如 LVDS。因此，本案例信号采集系统的信号输入单元应能够支持单端的 LVTTL 电平信号以及差分的 LVDS 信号。

随着 PC 机性能的不断提高，原来许多需要服务器甚至工作站才能完成的工作现在只需要一台简单的 PC 机就能够很快实现。因此，基于 PC 机的信号采集系统是本案例的另一个需求，也就是说信号采集系统的信号输出单元是基于 PC 机的。

另外，针对不同的应用场合，面对不同的信号形式，更多的用户希望信号采集系统变成更加通用的系统，而不只是针对某种信号的专用采集系统。因此，具备二次开发即可编程能力的信号处理单元同样是本信号采集系统的一个需求。

13.1.3　选型及功能定义

本案例选择基于 FPGA、PCI 总线及 LVDS 接口的设计，能够满足高速采集及实时处理的要求。

1．信号处理单元选型

首先，选用 Altera Cyclone 系列 FPGA 综合考虑了逻辑、存储器、锁相环(PLL)和高级 I/O 接口，是价格敏感应用的最佳选择。

在本案例中选用的 Cyclone EP1C6Q240 是 Cyclone 系列 FPGA 的中级产品。具有 5980 个逻辑单元，内嵌 20 个 M4K RAM 块（128*36bit），具有 2 个 PLL，最大支持 185 个用户 I/O 管脚资源。

2．信号输入输出单元选型

其次，PCI 总线是满足工业标准的应用最为广泛的扩展标准之一。它是一种具有多路地址线和数据线的高性能的 32/64 位总线，是 PC 机中必备的总线结构。

PLX Tech 公司提供的 PCI9054 器件是一种支持 32bit/33MHz PCI 总线桥芯片。它符合 PCI 本地总线规范 2.2 版，突发传输速率达到 132MB/s，本地总线支持复用/非复用的 32 位地址/数据，支持主模式、从模式以及 DMA 传输方式。

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用

同时，为了实现信号采集系统对 LVDS 信号的支持，选用 DS90LV048/ DS90LV047 作为 LVDS 信号接收/驱动器。它们均具有 4 路 LVDS 差分接收/驱动功能，支持高达 400Mbit/s（200MHz）的转换速率。通过它们可以实现 LVDS 信号与 FPGA 支持的 CMOS/LVTTL 信号进行转换。

3．其他单元选型

另外，为了满足 PCI 总线的 DMA 传输特性，需要在信号采集系统中加入一定量的缓冲区。采集的信号速率越高，需要的缓冲区容量越大，以保证不丢失数据。在本案例中，选用了 Micron 公司提供的 MT48LC2M32 SDRAM 作为缓冲区。它是一款 64Mb（512K*32*4banks） 全同步 SDRAM。

至此，一个基于 PCI 总线，支持高速 LVDS/LVTTL 信号采集，具有实时处理能力，可现场编程的信号采集系统完成选型。

13.2 系统工作原理分析

如前所述，一个完整的信号采集系统，除了具备信号输入单元、信号处理单元和信号输出单元外，还需要缓冲区、时钟以及电源等相关系统。如图 13.2 所示是本案例信号采集系统的结构框图。

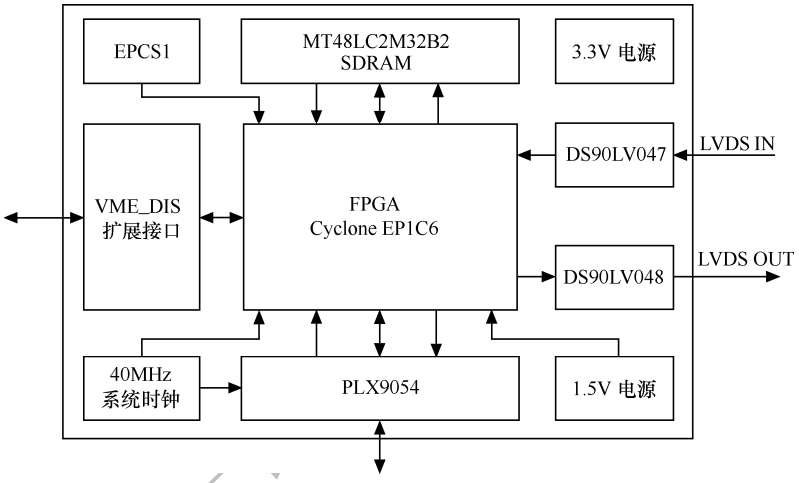


图 13.2 信号采集系统结构框图

该结构图中，方块表示对应的功能部件，连线表示互联关系。它清晰地表示了系统的组成结构及相互关系。首先将分别从 3 组总线，即数据总线、控制总线和地址总线来认识系统各主要模块之间的关系，再来分析系统的控制机制和工作原理。

13.2.1 数据总线

数据总线是系统中传送数据的链路，也是信号采集系统中最重要的链路。各功能部件的性能指标决定了数据总线的宽度及频率。如图 13.3 所示是本系统数据总线的连接示意图。

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

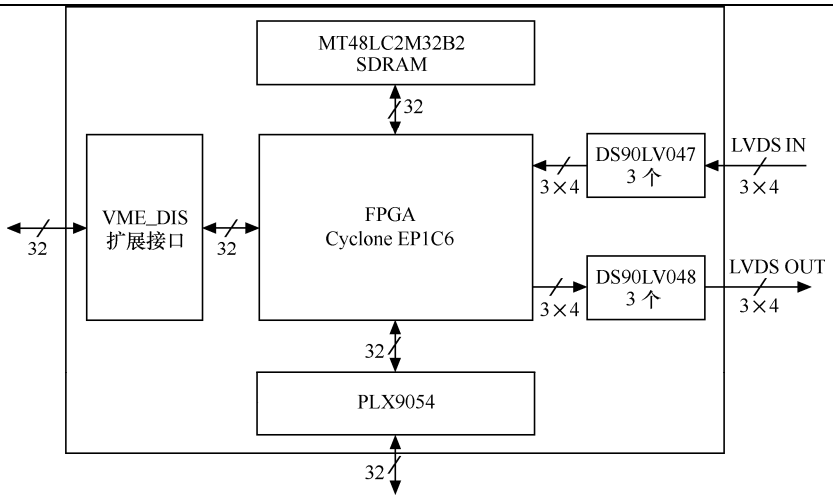


图 13.3　数据总线连接示意图

从图中可以看到，数据总线是双向的，也就是说具备了信号输入和信号输出两种用途。

首先看信号输入链路。TTL 单端信号通过 VME_DIS 扩展接口进入系统，可支持 32 路单端信号同时采集。LVDS 差分信号通过 3 个 DS90LV047 接收器进入系统，可支持 12 路差分信号同时采集。两种信号经过 FPGA 整理后传输至 SDRAM 中进行缓存，最终根据 PCI9054 的传输机制将数据传送至主机内存中。

再看信号输出链路。信号输出链路其实是信号输入链路的逆向过程。信号采集系统可以根据主机端的请求，将存储于主机上的数据通过 PCI 总线传输至 PCI9054，再经过 FPGA 控制、整理、转发至 LVDS 驱动器 DS90LV048 输出接口或 TTL 单端输出接口 VME_DIS 上。

13.2.2　控制总线

控制总线用于 FPGA 对缓冲区 SDRAM 以及 PCI9054 的控制。在 FPGA 中，可以将这两部分作为两个模块，一个是 SDRAM 控制器，另一个是 PCI 本地控制器。通过这两个控制器模块，实现 FPGA 与它们协调地工作。

SDRAM 控制器是 FPGA 内部用于控制外部 SDRAM 读写及刷新等操作的逻辑单元。 SDRAM 控制器的控制总线连接示意图如图 13.4 所示。

控制器左边的控制总线包含时钟信号 CLK、复位信号 RESET、命令信号 CMD、命令应答信号 CMDACK 以及数据有效信号 DM 等。

控制器右边的控制总线包含时钟信号 CLK、SDRAM 片选信号 CS_N、时钟时能信号 CKE、行选择信号 RAS_N、列选择信号 CAS_N 以及写使能信号 WE_N 等。通过 SDRAM 控制器，用户可以根据需要发送相应的名字对外部的 SDRAM 进行控制。

PCI 本地控制器是 FPGA 内部用于控制 PCI9054 进行参数配置和数据传输的逻辑单元。PCI 本地控制器的控制总线连接示意图如图 13.5 所示。

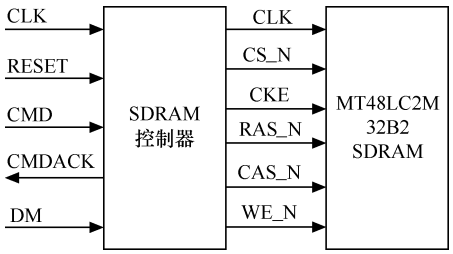


图 13.4　SDRAM 控制器控制总线连接示意图

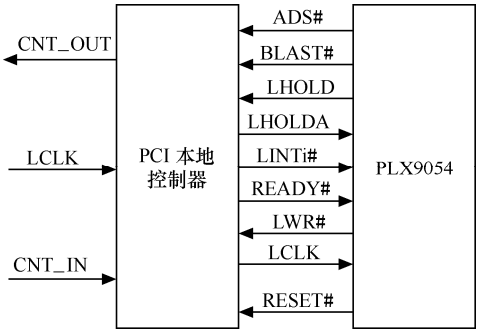


图 13.5　PCI 本地控制器控制总线连接示意图

控制器左边的控制总线包含时钟信号 LCLK、控制输入信号 CNT_IN 和控制输出信号 CNT_OUT 等。控制

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

输入输出信号是根据与 PCI9054 之间的交互产生的，实现了 FPGA 其他模块与 PCI9054 的交互控制。
控制器右边的总线则包含了 PCI9054 提供的本地端配置总线（图中未画出）以及交互时需要使用的控制线。包括了总线操作起始信号 ADS#、传输结束信号 BLAST#、本地总线占用请求信号 LHOLD、总线占用应答信号 LHOLDA、本地中断信号 LINTi#、总线准备好信号 READY#、总线读写信号 LWR#、时钟信号 LCLK 以及本地复位信号 RESET#。

13.2.3 地址总线

地址总线是存储器件控制器用于对存储空间进行寻址操作的总线，同时也可以实现编码器和译码器的作用，区分不同操作类别。如图 13.6 所示是本系统地址总线的连接示意图。

其中，SDRAM 控制器通过接收其他模块产生的地址 SDRAM_ADDR，产生 SDRAM 的行地址 SA 和列地址 BA，实现对 SDRAM 的存储空间的寻址。这种情况下，地址的变化由 FPGA 控制。

而 PCI 本地控制器则通过对 PCI9054 的本地总线地址线 LA 进行译码，结合 PCI 本地控制器的控制总线来区分 PCI 总线与 FPGA 之间的不同交互方式。这种情况下，PCI9054 完成了地址的映射和增减变化工作。

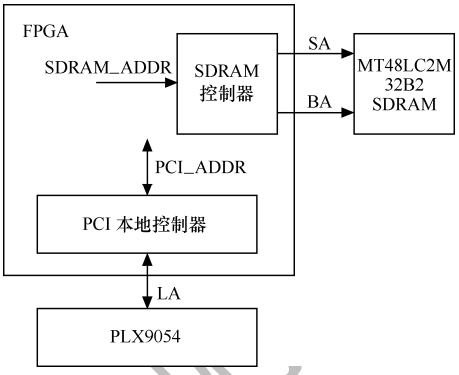


图 13.6 地址总线连接示意图

13.2.4 信号采集系统控制机制

首先，将需要采集的信号通过相应的接口接入本系统，形成一组 LVTTL 信号进入 FPGA。由于这一组信号的宽度未定，在将之存入具有 32 位数据宽度的 SDRAM 之前，往往需要进行总线宽度的控制。
一般在数据传输的接口控制中，常常采用 FIFO 或者 DPRAM 进行数据宽度控制。首先由于两种器件都具备两个数据端口，可以方便地调整前后两级逻辑的数据宽度。
其次，伴随数据宽度调整将产生时钟域变换问题。一般直接使用逻辑门搭建的数据宽度转换（串并/并串转换）还需要另外进行时钟系统的分配，且难以得到精确同步的时钟。而使用 FIFO 或 DPRAM 可以使用独立的时钟对两个数据端口进行控制。
再次，这两种器件均可以使用 FPGA 内部的逻辑资源搭建，无需使用分立元器件，不仅降低了成本也提高了系统的稳定性。
因此在本信号采集系统中使用了 DPRAM 对数据宽度进行转换。转换得到的 32 位宽度数据总线被传输至 SDRAM 中，完成信号输入单元和信号缓存单元的协调对接。
基于 PCI 的信号采集系统一般采用的是基于中断的 DMA 传输。由于 PCI 总线的发起、占用直至释放需要一定的时间，因此信号采集系统需要一个缓冲区来保证正在采集的信号被存储起来，直至下一次的 PCI 总线传输到来而缓冲区不溢出，即数据不丢失。
本系统中，当 SDRAM 中缓存的数据达到一定的量级时，FPGA 将向 PCI9054 发出本地中断，请求发起 DMA 传输。主机端捕获中断后，将启动 PCI9054 的 DMA 传输，经由 PCI 本地控制器向 SDRAM 控制器发送命令，将数据从 SDRAM 中取出，并传输至 PCI9054，最终进入主机内存中。

13.3 PCI 接口芯片 PCI9054 与 FPGA 的接口设计

13.3.1 PCI9054 的特性

PCI 接口芯片犹如一个硬核一样，完成了 PCI 局部总线端的总线交互，避免了用户直接对 PCI 局部总线进行复杂的操作。同时它为可编程逻辑器件提供了完备的本地总线接口，使用户可以较为方便地实现总线交

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

互的功能，节省了开发的周期。

PCI9054 是由美国 PLX 公司生产的先进的 PCI 桥接芯片。它采用了先进的 PLX 数据流水线结构技术，支持突发传输速率 132MB/s。PCI9054 具有如下一些特性。

- 兼容 PCI 协议 v2.2。
- 具有电源管理功能。
- 本地总线具有两个独立的 DMA 通道。
- 具有 6 个可编程 FIFO 用于无等待突发传送。
- 支持 3.3V、5V 芯片和接口电平标准。
- 本地时钟支持最高 50MHz。

如图 13.7 所示为 PCI9054 的内部结构图。

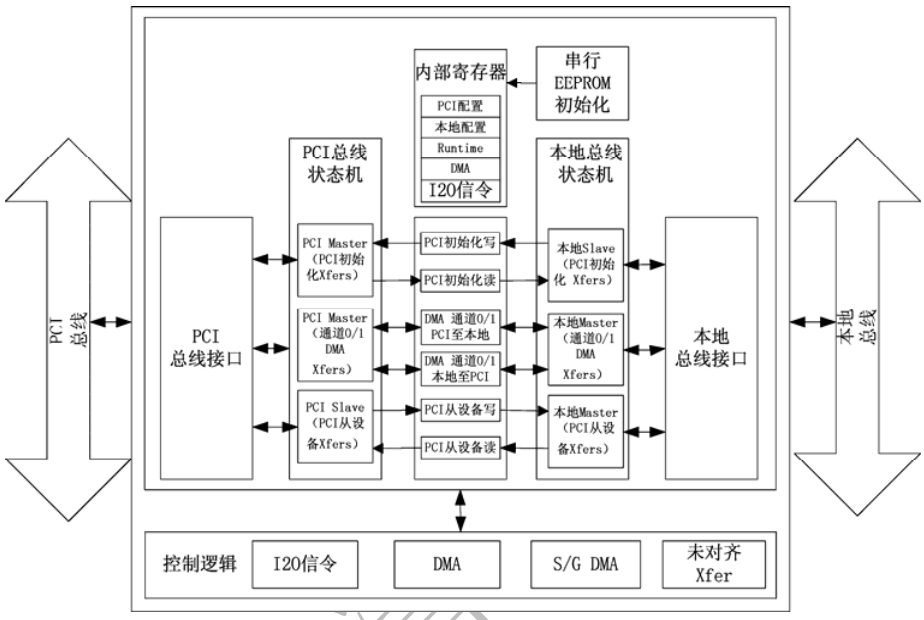


图 13.7　PCI9054 内部结构图

13.3.2　PCI9054 工作模式

PCI9054 的本地总线时钟可与 PCI 总线时钟异步，本地总线有 3 种工作模式：M 模式、C 模式和 J 模式，可通过模式选择控制引脚 MODE[1:0]进行控制，如表 13.1 所示。

表 13.1　　PCI9054 工作模式

模　　式	说　　明
M	32 位地址/32 位数据，非复用方式，用于直接连接 MPC850 或 MPC860
C	32 位地址/32 位数据，非复用方式
J	32 位地址/32 位数据，复用方式

本系统采用的是 C 模式（MODE[1:0]= 00）。事实上，C 模式能够满足绝大多数的应用需求，而且 C 模式的本地总线操作时序最简单，逻辑控制相对容易，其开发难度相对较低，因此，如无特殊需求，建议采用 C 模式。

要实现 PCI9054 C 模式与 FPGA 的接口设计，首先要明确 PCI 信号采集设备对于主机来说是主控设备（MASTER）还是目标设备（TARGET），还要明确是 memory 操作还是 I/O 操作。主控设备可以控制总线驱动地址、数据及控制信号；目标设备不能启动总线操作，只能通过主控设备来发起并驱动相应的信号。在本系统设计中，PCI 本地总线端使用的是 FPGA 而不是上位机，因此 PCI9054 对于 PCI 局部总线来讲是目标设备，由主机上其他设备作为主控设备进行 PCI 总线的操作。对于 PCI 本地总线来讲是 PCI9054 又是主控设备，通过 PCI9054 提供的本地接口启动本地端地址、数据及控制总线的操作。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

首先作为 PCI 局部总线的目标设备，PCI 局部总线通过控制 C/BE[3:0]#来发起不同的操作。如表 13.2 所示为作为目标设备的 PCI9054 局部总线命令编码表。

表 13.2　　PCI 目标设备命令编码表

命 令 类 型	C/BE[3:0]码型
I/O 读	0010
I/O 写	0011
存储器读	0110
存储器写	0111
配置读	1010
配置写	1011
存储器重复读	1100
PCI 双地址周期	1101
存储器按行读	1110
存储器写无效	1111

在实际的操作中，用户通过应用程序对 PCI 设备进行操作时，PCI9054 完成了将命令类型从 PCI 局部总线向本地总线的传递，并且这个过程对于用户来说是不可见的。用户只需要在本地总线端，即 FPGA 上实现对应命令类型的逻辑设计即可。因此，在 FPGA 内部需要对所有的命令操作进行相应的设计。

在本系统的 PCI 操作中，包含了 I/O 读、写，存储器读、写，配置读、写等操作，其中的存储器读、写操作使用的是 Scatter/Gather DMA 传输方式。在 FPGA 中通过判断 PCI9054 传递的地址映射空间类型和大小来区分主机需要进行的操作类型。

首先来认识一下 PCI 设备的空间配置情况。

13.3.3　PCI 设备空间配置

通常 PCI 设备一般有配置空间、存储器空间和 I/O 空间 3 类资源。配置空间是必须的，根据设计需要，后两种资源可以只包含其中之一。

定义配置空间的目的在于为PCI 设备提供一套适当的配置措施，使之满足现行的和可预见的系统配置机构。配置空间是一个容量为 256 字节并具有特定记录结构或模型的地址空间。在实际的系统配置中，PCI9054 的配置空间通过烧写在 EEPROM 中的内容进行配置。

PCI9054 有 Space 0 和 Space 1 两个地址空间，两个地址空间都可以配置成存储器空间和 I/O 空间。存储器空间和 I/O 空间的配置既有相同之处，也有很大的区别。其相同点为两者都以字节为单位进行空间配置。不同之处有以下几点。

- 存储器空间支持 0～4GB 的空间映射，I/O 空间只支持 0～256 字节的空间映射。
- 存储器空间支持 8/16/32 位总线位宽操作，I/O 空间只支持 32 位总线位宽操作。
- 存储器空间支持单一周期和突发模式读写操作，I/O 空间只支持单一周期读写操作。
- 编写驱动时，存储器空间用程序指针进行访问，I/O 空间则用专用的函数进行访问。

用户应根据实际的操作需求选择合适的空间配置方式和参数，以便获得正确、高效的总线传输性能。例如进行数据流的传输时，最好使用存储器空间，采用 DMA 读写方式；进行参数配置或寄存器值读取时，使用 I/O 空间，采用 I/O 读写方式。

13.3.4　PCI9054 与 FPGA 接口设计

如上所述，FPGA 需要根据 PCI9054 传递的地址映射空间类型和大小来区分主机需要进行的操作类型。因

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

此主机必须与 FPGA 达成一个一致的操作类型与地址映射空间协议，保证各个操作之间不会产生地址空间的重叠。

下面介绍一下进行 I/O 操作及 DMA 操作时，PCI9054 与 FPGA 之间的接口设计。

1．I/O 操作

如图 13.8 所示为 PCI9054 进行目标设备 I/O 读写时的结构图。其中，左图为读结构图，右图为写结构图。每个图中的左半部分为 PCI9054 与 PCI 局部总线的交互，右半部分为 PCI9054 与 FPGA 之间的交互。右半部分包含的控制线、地址线及数据线就是进行 I/O 操作时需要在 FPGA 内进行的接口设计。

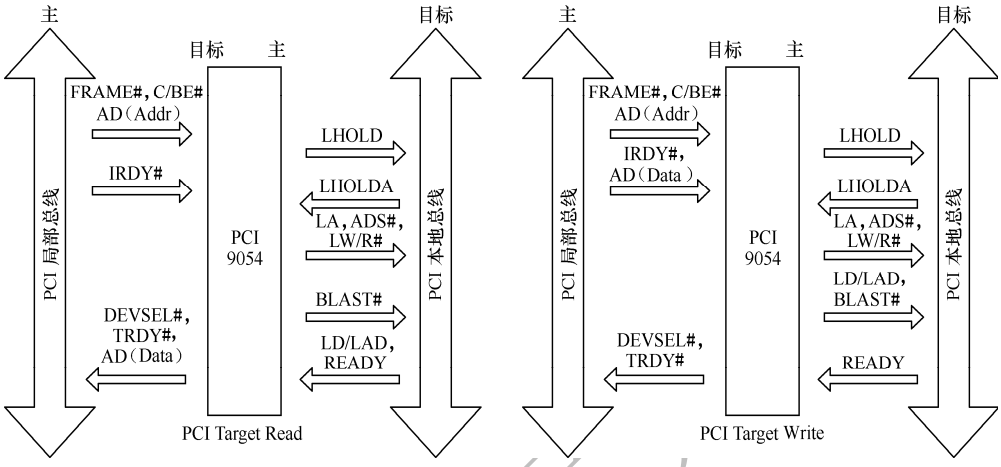
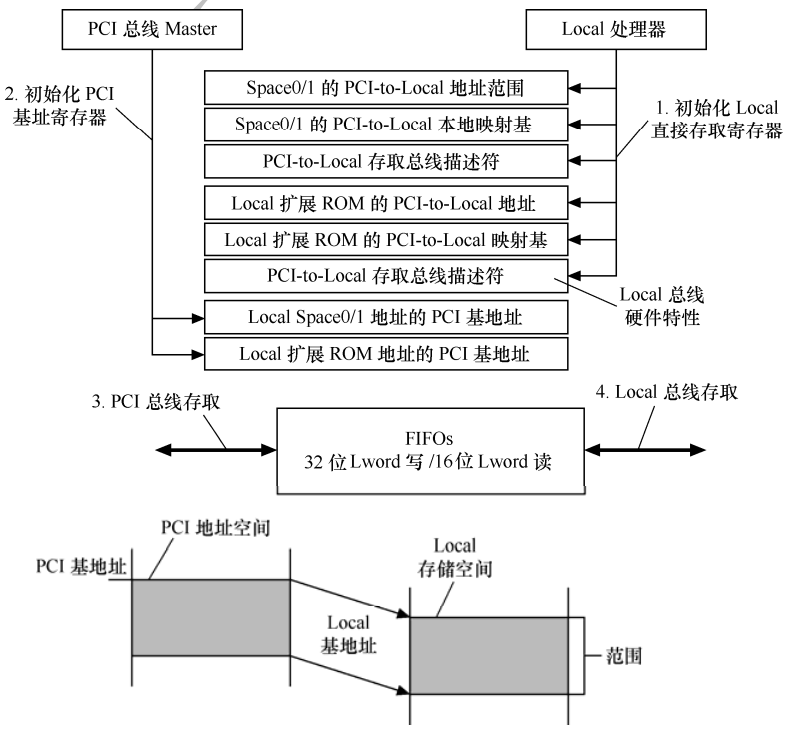


图 13.8　PCI9054 目标设备 I/O 读写结构图

首先，PCI 局部总线端的主机或其他设备作为主设备发起对 PCI9054 的 I/O 操作，此时 PCI9054 的局部总线端是目标设备。同时 PCI9054 的本地总线端又作为主设备向 FPGA 发起 I/O 操作，FPGA 作为目标设备接受总线控制。

如图 13.9 所示为 PCI9054 目标设备 Local 总线存取过程图。可以看到，存取过程主要是进行地址的映射过程。首先进行 Local 端和 PCI 端的寄存器配置，再通过 PCI9054 内部的两个目标设备 FIFO 进行存取，主机根据寄存器的值即可得到映射位置上的数据。



华清远见
华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

图 13.9　PCI9054 目标设备 Local 总线存取过程

如图 13.10 所示为 PCI9054 目标设备 Local 总线 I/O 典型波形图。它描述了 I/O 操作时 FPGA 内的时序，也是用户需要进行设计的时序。

该波形图表述了下面几个交互过程。

- 主设备（主机或其他设备）发起 Local 总线占用请求 LHOLD 信号，FPGA 应答产生 LHOLDA 信号允许该请求。
- 主设备获得允许后，发起操作起始信号 ADS#，并提供 Local 总线读写方向信号 LW/R#以及地址信号 LA[31:2]。
- FPGA 获得起始信号后，发出总线准备好信号 READY#，并执行相应的操作。
- 主设备完成操作后，发起操作结束信号 BLAST#，FPGA 捕获后，改变 READY#的状态。
- 主设备捕获 READY#状态改变后，释放 Local 总线占用信号 LHOLD。
- FPGA 应答总线释放，结束操作。

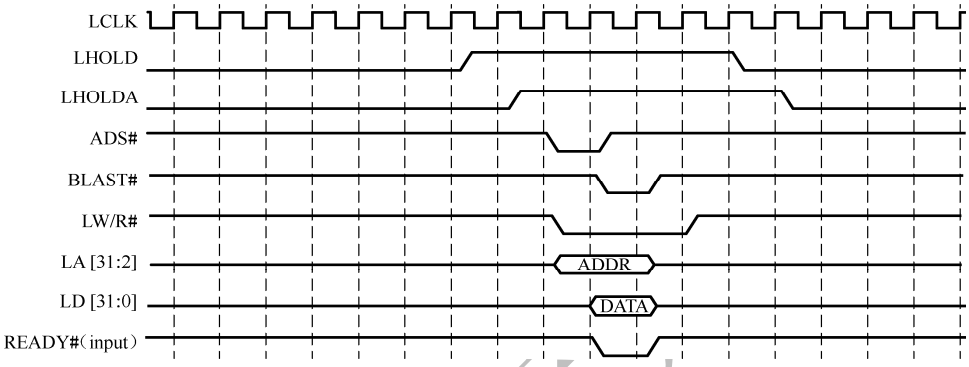


图 13.10　PCI9054 目标设备 Local 总线 I/O 典型波形图

下面是这个交互过程中对应的 Verilog 代码，其中分配的 I/O 空间大小为 64 字节，通过对 LA[4]译码实现操作类型的区分。

```
always @(posedge LCLK)
    LHOLDA <= LHOLD;                          //总线请求及应答
always @(posedge LCLK or negedge RESET_)begin
    if(!RESET_)                               //PCI9054 复位
       READY1_  <= 1'b1;
    else if(!ADS_ & LWR_ & BLAST_ & LA[31:4]==28'h0000_001)
       READY1_ <= 1'b0;                       //起始状态，通过地址线判断操作类型
    else if(ADS_ & LWR_ & BLAST_ & !READY1_)begin
        //TODO                                //中间状态，可执行相应的操作
        READY1_ <= 1'b0;
        end
    else if(ADS_ & LWR_ & !BLAST_ & !READY1_)begin
        //TODO                                //结束状态，可执行相应的操作
        READY1_ <= 1'b1;
        end
    else
        READY1_ <= 1'b1;                      //无操作状态
    end
```

2．DMA

如图 13.11 所示为 PCI9054 进行目标设备 DMA 读写时的结构图。其中，左图为 PCI-to-Local 结构图，右图

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

为写结构图。每个图中左右两部分完成的交互功能同 I/O 操作一样。

不同的是，进行 DMA 操作时，不管是与 PCI 局部总线还是 PCI 本地总线交互，PCI9054 始终是主设备。主机或 FPGA 只需发出 DMA 开始信号后，即可由 PCI9054 完成 DMA 的所有操作。可见，DMA 操作可以大幅度减轻主机端 CPU 的负担。

PCI9054 支持 Block（块）DMA 和 Scatter/Gather DMA 两种传输模式，前者使用连续的内存块进行地址映射，后者可以通过自动添加的指针链，使用不连续的内存块实现大块内存的地址映射，可提高系统内存的使用效率。

如图 13.12 所示为 Scatter/Gather DMA 模式存取过程图。可以看到，DMA 存取过程同样也是进行地址的映射过程。在这种模式下增加的是描述指针寄存器，通过它实现零散内存的连续化。

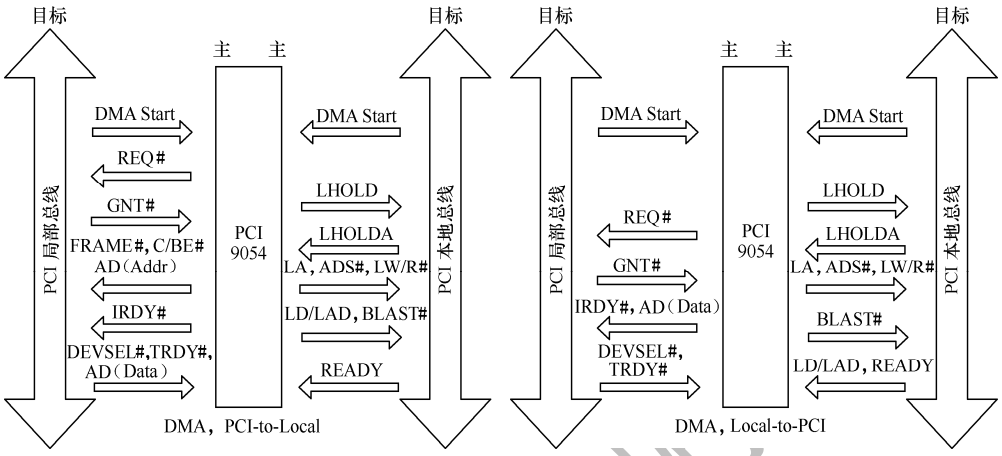


图 13.11　PCI9054 目标设备 DMA 读写结构图

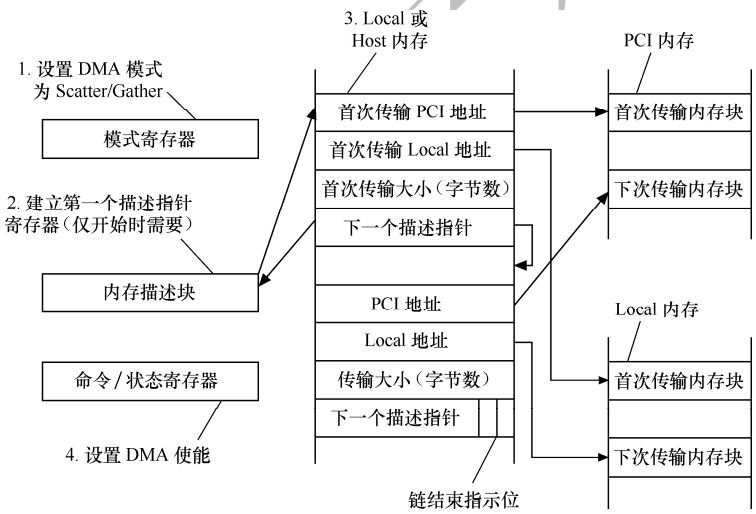
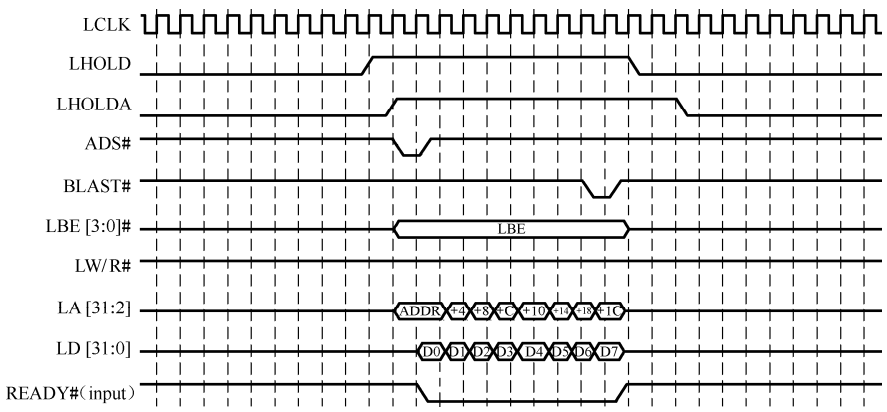


图 13.12　Scatter/Gather DMA 模式存取过程

如图 13.13 所示为 PCI9054 目标设备 Local 总线 DMA 典型波形图。它描述了 DMA 操作时 FPGA 内的时序，也是用户需要进行设计的时序。



开发板免费领取试用　开发板免费领取试用

图 13.13　PCI9054 目标设备 Local 总线 DMA 典型波形图

该波形图表述了下面几个交互过程。

- 主设备（主机或其他设备）发起 Local 总线占用请求 LHOLD 信号，FPGA 应答产生 LHOLDA 信号允许该请求。
- 主设备获得允许后，发起操作起始信号 ADS#，并提供 Local 总线读写方向信号 LW/R#以及地址信号 LA[31:2]。
- FPGA 获得起始信号后，发出总线准备好信号 READY#，并执行相应的操作；此时 LA 地址会根据 LBE 字节使能的设置每读写一次（周期）递增一次，如图中 LA 地址所示。
- 主设备完成操作后，发起操作结束信号 BLAST#，FPGA 捕获后，改变 READY#状态。
- 主设备捕获 READY#状态改变后，释放 Local 总线占用信号 LHOLD。
- FPGA 应答总线释放，结束操作。

下面是这个交互过程中对应的 Verilog 代码，其中分配的 DMA 空间大小为 1MB，通过对 LA[20]译码实现操作类型的区分。

```
always @(posedge LCLK)
    LHOLDA <= LHOLD;                          //总线请求及应答
always @(posedge LCLK or negedge RESET_)begin
    if(!RESET_)                               //PCI9054 复位
       READY2_ <= 1'b1;
    else if(!ADS_ & LWR_ & BLAST_ & LA[31:20]== 12'h001)
       READY2_ <= 1'b0;                       //起始状态，通过地址线判断操作类型
    else if(ADS_ & LWR_ & BLAST_ & !READY2_)begin
        //TODO                                //中间状态，可执行相应操作
        READY2_ <= 1'b0;
        end
    else if(ADS_ & LWR_ & !BLAST_ & !READY2_)begin
        //TODO                                //结束状态，可执行相应操作
        READY2_ <= 1'b1;
        end
    else
        READY2_ <= 1'b1;                      //无操作状态
    End
```

13.4　PCI 卡的驱动程序设计

13.4.1　WDM 驱动程序模型

设计完成的信号采集设备在插入计算机后，在对其进行控制之前，需要编写基于操作系统平台上的驱动程序。设备驱动程序是一个包含了许多操作系统可调用例程的容器，这些例程可以使硬件设备执行相应的动作，它是硬件与上层软件之间沟通的桥梁。

在本案例中，我们针对最常使用的操作系统 Windows 98/2000/XP 系统，使用了 WDM（Windows Driver Model）驱动程序模型进行程序开发。

WDM 模型是从 WinNT 3.51 和 WinNT 4 的内核模式设备驱动程序发展而来的。WDM 主要的变化是增加了对即插即用、电源管理、Windows Management Interface(WMI)、设备接口的支持。WDM 模型的主要目标是实现能够跨平台使用、更安全、更灵活、编制更简单的 Windows 设备驱动程序。

WDM 采用了“基于对象”的技术，建立了一个分层的驱动程序结构。通过 WDM 模型的引入，可以减轻设备驱动程序的开发难度和周期，逐渐规范设备驱动程序的开发，应该说，WDM 是当前基于 Windows 平

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

台的设备驱动程序的主流。

WDM 模型主要采用分层的方法，模仿面向对象的技术，先进行逻辑上的“分层”，然后将标准的实现和低层细节“封装”起来，形成“基类”，客户程序通过“继承”的方式来扩展“基类”的功能，完成所需要的实现。

13.4.2 设备和驱动程序的层次结构

在 WDM 模型中，每个硬件设备至少有两个驱动程序：一个功能驱动程序（function driver）和一个总线驱动程序（bus driver）。

如图 13.14 所示为 WDM 中设备对象和驱动程序的层次结构。

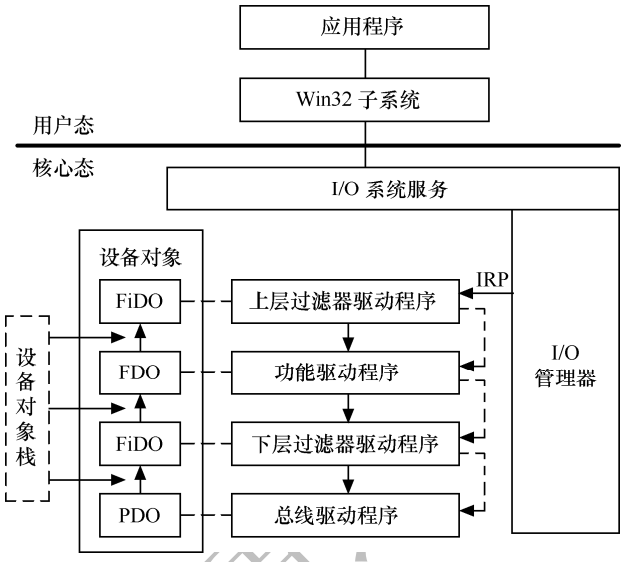


图 13.14 WDM 中设备对象和驱动程序的层次结构

1．过滤驱动程序

过滤驱动程序是一个可选项，当一个用户需要改变或新添一些功能到一个设备、一类设备或一种总线时，就可以编写一个过滤驱动程序。在设备栈里，过滤驱动程序安装在一个或几个设备驱动程序的上面或下面。

过滤驱动程序拦截对具体设备、类设备、总线的请求，做相应的处理，以改变设备的行为或添加新的功能。但过滤驱动程序只处理那些它所关心的 I/O 请求，对于其他的请求可以交给其他的驱动程序来处理，这样可以非常灵活地改变设备的行为。

2．功能驱动程序

功能驱动程序是物理设备的主要驱动程序，它实现设备的具体功能，一般由设备的生产商来编写。功能驱动程序的主要功能是：提供对设备的操作接口、操作对设备的读写、管理设备的电源策略等。

功能驱动程序由类驱动程序和微型驱动程序组成。类驱动程序实现了某一类设备的常用操作，驱动程序的开发者可以只编写非常小的微型驱动程序，去处理具体设备特殊的操作，而对于其他大量的常规操作，可以调用该类的类驱动程序，这也是 WDM 驱动程序的优点之一。

微软公司提供的类驱动程序处理常用的系统任务，比如，即插即用功能和电源管理。类驱动程序保证了操作系统在处理类似的任务时的一致性，从而提高了系统的稳定性。

设备生产商提供微型驱动程序，以实现自己设备的特殊功能，同时调用合适的类驱动程序完成其他的通用

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

工作。将大量的标准操作的代码通过各种类驱动程序来实现，并集成在操作系统中，这样的方式可以有效地减少具体设备的微型驱动程序的大小，也就减小了程序出错的可能。
如果某一类设备存在着工业标准，微软公司就会提供一个该类设备的 WDM 类驱动程序。这个类驱动程序实现了该类设备所有必须的任务，但不实现任何具体设备所特有的东西。

3．总线驱动程序

总线驱动程序为实际的 I/O 总线服务。在 WDM 的定义中，总线是用来连接其他的物理的、逻辑的、虚拟的设备。总线包括传统的总线 SCSI 和 PCI，也包括并口、串口以及 i8042 端口。微软公司已经为 Windows 操作系统提供了总线驱动程序。总线驱动程序已经包含在操作系统里了，用户不必安装。
一个总线驱动程序负责以下的工作：枚举总线上的设备，向操作系统报告总线上的动态事件，响应即插即用和电源管理的 I/O 请求，提供总线的多路存取，管理总线上的设备等。

13.4.3 PCI 设备驱动程序例程

PCI 设备的 WDM 驱动程序一般需要使用 Windows DDK（Drivers Develop Kits）及 C 语言进行开发。下面介绍一些 PCI 设备最常见的例程，这些例程将告诉我们如何对 PCI 设备进行控制。

1．DriverEntry 例程

每个 WDM 驱动程序，不管它的用途是什么，都要对外界显示一个名字为 DriverEntry 的例程。该例程初始化各种驱动程序数据结构，并为所有其他驱动程序组件准备好执行环境。主要的工作是在传递的 DriverObject 中存储一系列的回调例程指针。DRIVER_OBJECT 结构有操作系统用于存储与驱动程序有关的任何信息。
在 DriverEntry 例程中通常要完成如下步骤。

- DriverEntry 找到它将要控制的硬件。那个硬件是经过分配的，即被标志为由该驱动程序控制。
- 通过声明另一个驱动程序入口点，初始化驱动程序对象。通过把函数指针直接保存到驱动程序对象中完成声明工作。
- 如果成功，DriverEntry 应该把 STATUS_SUCCESS 返回给 I/O 管理程序。

DriverEntry 的函数原型为：

```
NTSTATUS DriverEntry (
PDRIVER_OBJECT pDriverObject,
PUNICODE_STRING pRegistryPath
)
```

它接收一个指向它本身的驱动程序对象的指针，DriverEntry 例程必须对它（指针）初始化。它还接收一个 UNICODE_STRING，它包含注册表中驱动程序服务键的路径。内核模式驱动程序根据该字符串从系统注册表中提取任何驱动程序专有的参数。注册表字符串采用如下形式：

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\DriverName
```

下面是驱动程序的 DriverEntry 例程的部分代码，里面定义了将要用到的回调函数。

```
NTSTATUS DriverEntry (
    PDRIVER_OBJECT   pDriverObject,
    PUNICODE_STRING   pRegistryPath
    )
{   …
```

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

```
    // 回调函数
    pDriverObject->DriverUnload                         = DriverUnload;
    pDriverObject->MajorFunction[IRP_MJ_CREATE]         =Dispatch_Create;
    pDriverObject->MajorFunction[IRP_MJ_CLOSE]          =Dispatch_Close;
    pDriverObject->MajorFunction[IRP_MJ_READ]           =Dispatch_Read;
    pDriverObject->MajorFunction[IRP_MJ_WRITE]          =Dispatch_Write;
    pDriverObject->MajorFunction[IRP_MJ_CLEANUP]        =Dispatch_Cleanup;
    pDriverObject->MajorFunction[IRP_MJ_DEVICE_CONTROL]
                                                          =Dispatch_IoControl;
    pDriverObject->MajorFunction[IRP_MJ_PNP]            = Dispatch_Pnp;
    pDriverObject->MajorFunction[IRP_MJ_POWER]          =Dispatch_Power;
    pDriverObject->MajorFunction[IRP_MJ_SYSTEM_CONTROL]
                                                          =Dispatch_System
                                                  Control;
    pDriverObject->DriverExtension->AddDevice             = AddDevice;
    …
    return STATUS_SUCCESS;
}
```

2．AddDevice 例程

大多数的 WDM PDO 都是在 PnP 管理器调用该程序入口点时被创建的。插入新设备后，系统启动时，总线枚举器会发现总线上的所有设备会自动寻找并安装设备的驱动程序，并由驱动程序中的处理 PnP 功能模块自动处理 AddDevice 例程及其他的 PnP 消息。

AddDevice 例程使用 IoCreateDevice 函数创建设备对象，再使用 IoCreateSymbolicLink 函数将设备组成为一个特定的设备接口，供 Win32 使用。

其函数原型为：

```
NTSTATUS AddDevice (PDRIVER_OBJECT pDriverObject, PDEVICE_OBJECT pdo)
```

必须在 DriverEntry 入口函数中进行声明，下面是该函数的部分代码：

```
NTSTATUS AddDevice (
    PDRIVER_OBJECT pDriverObject,
    PDEVICE_OBJECT pdo
    )
{…
    // 建立设备名称并创建它
    for (i=0; i < 20; i++)
    {
    //转成 String 格式
        Swprintf (DeviceName, L"\\Device\\" PLX_DRIVER_NAME_UNICODE L"-%d", i);
    //初始化 DeviceName_Unicode
        RtlInitUnicodeString (&DeviceName_Unicode, DeviceName);
    // 创建设备
        status = IoCreateDevice(
               pDriverObject,
               sizeof(DEVICE_EXTENSION),
               &DeviceName_Unicode,
               FILE_DEVICE_UNKNOWN,
               0,
               FALSE,
               &fdo
               );
    …
     //为用户应用程序创建 Win32 关联名
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

```
    //转成 String 格式
    swprintf (DeviceLinkName, L"\\DosDevices\\" PLX_DRIVER_NAME_UNICODE L"-%d", i);
    //初始化 DeviceLinkName_Unicode
    RtlInitUnicodeString (
       &DeviceLinkName_Unicode,
       DeviceLinkName
       );
    //建立设备关联符号
    status = IoCreateSymbolicLink (
          &DeviceLinkName_Unicode,
          &DeviceName_Unicode
          );
    …
    return STATUS_SUCCESS;
}
```

3．DispatchPnp 例程

支持即插即用主要是指实现一个 AddDevice 例程和一个 IRP_MJ_PNP 处理程序。这个 PnP IRP 有 8 个主要次功能代码，大多数的 WDM 驱动程序需要支持这些次功能代码。

- IRP_MN_START_DEVICE。
- IRP_MN_QUERY_REMOVE_DEVICE。
- IRP_MN_REMOVE_DEVICE。
- IRP_MN_CANCLE_REMOVE_DEVICE。
- IRP_MN_STOP_DEVICE。
- IRP_MN_QUERY_STOP_DEVICE。
- IRP_MN_CANCLE_STOP_DEVICE。
- IRP_MN_QUERY_CAPABILITIES。

还有一些不太常用的 IRP，这里就不再一一介绍，下面是这部分驱动的部分代码。

```
NTSTATUS Dispatch_Pnp(
    PDEVICE_OBJECT fdo,
    PIRP          pIrp
    )
{   …
    // 检查次功能代码
    switch (stack -> MinorFunction)
    {
        case IRP_MN_START_DEVICE:                    //配置并初始化设备
            status = HandleStartDevice(fdo, pIrp);
            break;
        case IRP_MN_STOP_DEVICE:                     //关闭设备
            status = HandleStopDevice(fdo, pIrp);
            break;
        case IRP_MN_REMOVE_DEVICE:                //关闭并删除设备
            Unlock = FALSE;
            status = HandleRemoveDevice(fdo, pIrp);
            break;
        case IRP_MN_QUERY_REMOVE_DEVICE:             //查询设备是否可被安全删除
            status = DefaultPnpHandler(fdo, pIrp);
            break;
        case IRP_MN_CANCEL_REMOVE_DEVICE:            //忽略以前的 QUERY_REMOVE
            status = DefaultPnpHandler(fdo, pIrp);
            break;
        case IRP_MN_QUERY_STOP_DEVICE:               //查询设备是否可被安全关闭
            status = DefaultPnpHandler(fdo, pIrp);
            break;
        case IRP_MN_CANCEL_STOP_DEVICE:              //忽略以前的 QUERY_STOP
            status = DefaultPnpHandler(fdo, pIrp);
            break;
        case IRP_MN_QUERY_CAPABILITIES:              //取设备能力
            status = DefaultPnpHandler(fdo, pIrp);
            break;
    … }
```


开发板免费领取试用 开发板免费领取试用

```
    return status;
}
```

这些功能代码函数都在 DriverEntry（）入口函数中进行了声明。

4．DispatchPower 例程

WDM 设备驱动程序支持电源管理，一个设备可以改变它的电源使用来响应系统电源状态变化，且在处于空闲状态时可以减少它自己的电源使用。一个休眠的设备可以唤醒系统，如当调制解调器收到到达的呼叫时。

驱动程序的电源管理例程围绕电源 IRP_MJ_POWER IRP 进行处理，这些例程处理这个 IRP，并在需要时产生这个 IRP。这个 IRP 有 4 个电源管理次功能代码。

- IRP_MN_WAIT_WAKE。
- IRP_MN_POWER_SEQUENCE。
- IRP_MN_SET_POWER。
- IRP_MN_QUERY_POWER。

这部分的代码如下：

```
NTSTATUS Dispatch_Power(
    PDEVICE_OBJECT fdo,
    PIRP         pIrp
    )
{
    NTSTATUS         status;
    PIO_STACK_LOCATION stack;
//获取指向被调用的 Irp 的栈位置的指针
    stack = IoGetCurrentIrpStackLocation(pIrp);
    switch (stack->MinorFunction)
    {
        case IRP_MN_WAIT_WAKE:                //唤醒计算机，响应一个外部事件
            status = DefaultPowerHandler(fdo, pIrp);
            break;
        case IRP_MN_POWER_SEQUENCE:           //确定设备是否真正进入特定的电源状态
            status = DefaultPowerHandler(fdo, pIrp);
            break;
        case IRP_MN_SET_POWER:                //设置系统或设备电源状态
            status = HandleSetPower(fdo, pIrp);
            break;
        case IRP_MN_QUERY_POWER:              //查询系统或设备状态变化是否可行
            status = HandleQueryPower(fdo, pIrp);
            break;
        default:                              //确定设备是否真正进入特定的电源状态
            status = DefaultPowerHandler(fdo, pIrp);
            break;
    }
    return status;
}
```

5．Unload 例程

在默认情况下，驱动程序装入之后，在重新引导之前就一直保持在系统中。要使系统可卸载，必须有一个 Unload 例程。Uload 例程在 DriverEntry 期间声明，然后，每当驱动程序被手动或自动卸载时，I/O 管理程序就调用该例程。

该例程函数原型为：

```
VOID Unload ( PDRIVER_OBJECT pDriverObject )
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

通常一个 Uload 例程执行如下工作。

- 对于某些种类的硬件，设备状态应当保存在注册表中。那样，在 DriverEntry 下一次执行时，设备能够恢复到最近已知的状态。
- 如果启动了设备中断，Unload 例程必须仅用它们，并断开它们与中断对象的连接。
- 必须释放属于驱动程式的硬件。
- 必须从 Win32 名字空间中删除符号链接名字，这可以用 IoDeleteSymbolicLink 完成。
- 必须用 IoDeleteDevice 删除设备对象自身。
- 释放驱动程序占据的任何池内存。

驱动中的部分代码如下：

```
VOID DriverUnload(PDRIVER_OBJECT pDriverObject)
{
    // 释放公共缓冲区
    if (Gbl_CommonBuffer.PciMem.PhysicalAddr != (U32)NULL)
    {
        // 释放内存描述列表（MDL）
        if (Gbl_CommonBuffer.pMdl != NULL)
        {
            IoFreeMdl(Gbl_CommonBuffer.pMdl);
            Gbl_CommonBuffer.pMdl = NULL;
        }
        // 释放公共缓冲区
        if (Gbl_CommonBuffer.pKernelVa != NULL)
        {
            MmFreeContiguousMemory(Gbl_CommonBuffer.pKernelVa);
            Gbl_CommonBuffer.pKernelVa = NULL;
        }
    }
//释放 DMA 使用的缓冲区
#if defined(DMA_SUPPORT)
    DriverBufferTerminate();
#endif
}
```

6．Dispatch 例程

驱动程序装载是第一步工作，但是最终驱动程序的作业是响应 I/O 请求——来自用户模式应用程序的请求或者来自系统其他地方的请求。Windows 2000 驱动程序通过 Dispatch 例程来处理这些请求。I/O 管理程序调用这些例程以响应请求。

要启用特定的 I/O 函数代码，驱动函数必须首先“声明”响应这样一个请求的 Dispatch 例程。声明机制是 DriverEntry 执行的工作，它把 Dispatch 例程函数地址保存在驱动程序对象的 MajorFunction 表的合适位置上。

I/O 函数代码是用于表的索引。其中，每个 I/O 函数代码（表索引）由一个 IRP_MJ_XXX 形式的惟一符号标示，在 NTDDK.h（或 WDM.h）包含文件中定义。

所有驱动程序必须支持函数代码 IRP_MJ_CREATE，因为编写此代码的目的是响应 Win32 CreateFile 调用。如果不支持该代码，Win32 应用程序将无法获取设备的句柄。同样也必须支持 IRP_MJ_CLOSE，以处理 Win32 CloserHandle 调用。

驱动程序应当支持的其他函数代码取决它控制的设备的本质。表 13.3 将用到的 I/O 函数代码与产生它们的 Win32 调用相关联。

表 13.3　　Dispatch 例程表

Win32 函数	IRP 主功能代码	驱动程序例程的名称	说　　明
CreateFile	IRP_MJ_CREATE	Dispatch_Create	请求一个句柄
CloseHandle	IRP_MJ_CLEANUP	Dispatch_Cleanup	关闭句柄

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

CloseHandle	IRP_MJ_CLOSE	Dispatch_Close	取消挂起的 IRP
ReadFile	IRP_MJ_READ	Dispatch_Read	从设备获取数据
WriteFile	IRP_MJ_WRITE	Dispatch_Write	将数据发送到设备
DeviceIoControl	IRP_MJ_DEVICE_CONTROL	Dispatch_IoControl	控制操作

13.5 主机应用程序和驱动程序的接口设计

13.5.1 驱动程序例程的封装

通过调用 PCI 设备驱动程序的例程，我们可以实现操作系统对 PCI 设备的控制。但是直接调用例程进行编程往往显得不够直接，也不具有足够的针对性。因此在高级语言里面对 PCI 设备或者信号采集设备的控制，往往是调用已经封装过的例程。

将例程进行封装，形成一些可以调用的具有针对性的 PCI 信号采集设备 API 函数，是驱动开发者比较头疼的问题。因为这个过程需要了解 Windows WDM 模型，熟悉 Windows DDK 的开发过程，而这些往往需要花费硬件开发者的大量时间。

幸运的是，有一些很好的集成开发环境帮助我们完成了这个工作。用户只需要将具体设备的参数导入开发工具的向导中，即可自动生成对应该设备的驱动程序。这些开发工具包括 Driver Studio、Windriver 等。

另外，针对基于 PCI9054 的 PCI 设备，PLX 公司已经为用户提供了较为完备的 API 函数包。用户只需要根据需要调用其中的 API 函数，即可轻松实现对 PCI9054 采集设备的控制。下面介绍几个常用的 PLX API 函数。

13.5.2 PLX API 函数

PLX 公司为其提供的动态链接库中包含了丰富的 API 函数，可以用于 PCI9054 卡的控制与传输。在这些 API 函数中，有如下的几个是常常用到的（注：在 2007 年 5 月提供的 PLX SDK Version 5.0 中，API 函数名有所更新，下面仍然按照 V5 版之前的函数名进行介绍）。

1．PlxPciDeviceOpen 函数

用于设备启动，在对设备进行任何操作之前都应该先打开设备，其函数原型为：

```
RETURN_CODE PlxPciDeviceOpen(
DEVICE_LOCATION *pDevice,       //指向包含设备信息的数据结构的指针
HANDLE *pHandle                 //指向用于存放已打开设备的句柄的指针
);
```

DEVICE_LOCATION 数据结构的定义如下：

```
typedef struct _DEVICE_LOCATION{
U8 BusNumber;          //PCI 设备所在的总线号
U8 SlotNumber;         //PCI 设备所在的插槽号
U16 DeviceId;          //PCI 设备的设备号
U16 VendorId;          //PCI 设备的厂商号
U8 SerialNumber[12];  //PCI 设备的惟一标识码
} DEVICE_LOCATION;
```

对于 PCI9054 来说，DeviceId 为 10b5，VendorId 为 5406（不同的版本可能为 9054）。SerialBumber 的格式为“<device name> - <index number>”。如：有 PLX9054 的 PCI 卡，如果它的序列号为“Pci9054-1”，表示这是第二个 PLX9054 设备，第一个为“Pci9054-0”。

华清远见

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用 开发板免费领取试用

2．PlxPciDeviceClose 函数

用于关闭已打开的 PCI 设备，其函数原型为：

```
RETURN_CODE PlxPciDeviceClose (HANDLE hDevice); //已打开的 PCI 设备的句柄
```

3．PlxPciBoardReset 函数

用于复位已经打开的 PCI 设备，其函数原型为：

```
VOID PlxPciBoardReset (HANDLE hDevice);      //已打开的 PCI 设备的句柄
```

4．PlxBusIopRead 函数

用于从本地总线读取数据（常指 Direct Slave 读模式），其函数原型为：

```
RETURN_CODE PlxBusIopRead (
HANDLE hDevice,         //已打开的 PCI 设备的句柄
IOP_SPACE IopSpace,     //PCI 总线至本地总线，映射空间 0/1
U32 address,            //若 bRemap = FALSE，代表 IOPSpace 相对地址，否则是绝对地址
BOOLEAN bRemap,         //绝对地址与相对地址布尔值
VOID *pBuffer,          //指向用户用于存放读取数据的缓冲区
U32 ByteCount,
ACCESS_TYPE AccessType);//读数据方式，支持本地总线使用 8 位、16 位或 32 位进行传输
```

5．PlxBusIopWrite 函数

用于从本地总线写数据（常指 Direct Slave 写模式），其函数原型为：

```
RETURN_CODE PlxBusIopWrite (
HANDLE hDevice,         //已打开的 PCI 设备的句柄
IOP_SPACE IopSpace,     //PCI 总线至本地总线，映射空间 0/1
U32 address,            //若 bRemap = FALSE，代表 IOPSpace 相对地址，否则是绝对地址
BOOLEAN bRemap,         //绝对地址与相对地址布尔值
VOID *pBuffer,          //指向用户用于存放读取数据的缓冲区
U32 ByteCount,          //读取的数据量大小
ACCESS_TYPE AccessType);//读数据方式，支持本地总线使用 8 位、16 位或 32 位进行传输
```

6．PlxIntrAttach 函数

用于链接中断，当有效中断产生时，将事件传递给指针句柄，其函数原型为：

```
RETURN_CODE PlxIntrAttach (
HANDLE hDevice,           //已打开的 PCI 设备的句柄
PLX_INTR intrTypes,       //含有需要指定的中断信息的数据结构(PLX_INTR)
HANDLE *pEventHdl);       //指向接收中断事件的指针句柄
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

7．PlxIntrDisable 函数

用于禁止某类中断，其函数原型为：

```
RETURN_CODE PlxIntrDisable (
HANDLE hDevice,          //已打开的 PCI 设备的句柄
PLX_INTR *pPlxIntr);     //含有需要指定的中断信息的数据结构(PLX_INTR)
```

8．PlxIntrEnable 函数

用于使能某类中断，其函数原型为：

```
RETURN_CODE PlxIntrEnable (
HANDLE hDevice,           //已打开的 PCI 设备的句柄
PLX_INTR *pPlxIntr);     //含有需要指定的中断信息的数据结构（PLX_INTR）
```

9．PlxIntrStatusGet 函数

用于返回当前终端状态，其函数原型为：

```
RETURN_CODE PlxIntrStatusGet (
HANDLE hDevice,           //已打开的 PCI 设备的句柄
PLX_INTR *pPlxIntr);     //含有需要指定的中断信息的数据结构（PLX_INTR）
```

10．PlxIntrWait 函数

用于等待中断，并可设定等待时间长度，其函数原型为：

```
RETURN_CODE PlxIntrWait (
HANDLE hDevice,          //已打开的 PCI 设备的句柄
HANDLE hEvent,           //被等待的事件句柄
U32 Timeout_ms           //为等待的时间长度，单位为 ms
);
```

由于此函数只能用在 Linux 系统中，因此在 Windows 系统中使用 PLX API 函数包等待中断时，使用另外两个 Windows API 函数进行中断等待，即 WaitSingleObject（）函数和 WaitMultipleObject（）函数。

11．PlxDmaSglChannelOpen 函数

用于开启 Scatter/Gather 模式 DMA 通道，其函数原型为：

```
RETURN_CODE PlxDmaSglChannelOpen (
HANDLE hDevice,             //已打开的 PCI 设备的句柄
DMA_CHANNEL channel,        //为被开启的 DMA 通道
DMA_CHANNEL_DESC *pDesc); //包含 DMA 通道的参数描述结构体（DMA_CHANNEL_DESC）指针
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

12．PlxDmaSglChannelClose 函数

用于关闭已经打开的 Scatter/Gather 模式 DMA 通道，其函数原型为：

```
RETURN_CODE PlxDmaSglChanneClose (
HANDLE hDevice,            //已打开的 PCI 设备的句柄
DMA_CHANNEL channel);     //将要关闭的已被开启的 DMA 通道
```

13．PlxDmaSglTransfer 函数

用于开始进行 Scatter/Gather 模式的 DMA 传输，其函数原型为：

```
RETURN_CODE PlxDmaSglTransfer (
HANDLE hDevice,                 //已打开的 PCI 设备的句柄
DMA_CHANNEL channel,            //进行 Scatter/Gather 模式 DMA 传输的通道
DMA_TRANSFER_ELEMENT *dmaData,  //指向用于 DMA 数据缓冲区结构(DMA_TRANSFER_
                                //ELEMENT)指针
BOOLEAN returnImmediate);       //取值为 FALSE 表示程序等待 DMA 传输结束后再执行下一
                                //条指令
                                //，反之，则不等待结束就执行下一条指令
```

13.5.3 API 函数调用

有了这些丰富的 API 函数资源，用户即可根据自己的需要为信号采集设备设计功能多样的应用程序。在进行应用程序的设计时，只需要将 API 函数库包含在用户的程序中即可。

PLX 公司提供的 PLX SDK 开发包中包含了完整的 API 函数库文件。将这些 API 库文件复制到用户的应用程序目录下，并在程序中 include 进去，即可调用这些 API 函数。更多的函数及用法详见 PLX SDK 的编程者开发手册。

13.5.4 PCI9054 驱动程序安装

INF 文件是安装设备驱动程序时必须使用的文件，其扩展名为.inf。驱动程序安装后将它保存在 windows/inf 目录中，是 Winodws 操作系统下用来描述设备或文件等数据信息的文件。

INF 文件全称 Information File 文件。在操作系统发现新硬件之后向系统指明应该安装的驱动程序，系统为设备提供的服务以及注册表项要修改的内容。它为该设备提供一个全面描述硬件参数和相应驱动文件(动态链接库等文件)的信息。

驱动程序安装的最终结果包括对系统的两个永久更改。

- 描述驱动程序的系统注册表项、它的卸载顺序以及任何正确的配置资料。
- 将驱动程序文件复制到一个指定的系统目录中。

INF 文件是由标准的 ASCII 码组成，可以用任何一款文字编辑器查看修改其中的内容。它是分节的，每节以“[ ]”扩起来，每一个节名最长为 255 个字符（Windows 2000/XP/2003 操作系统中）或 28 个字符（Windows 98 操作系统中）。

节与节之间的内容叫条目，每一个节是由许多的条目组成的。每一个条目都由“=”分开，如“a=b”。如果每一个条目的等号后有多个值，则每一个值之间用逗号分隔开。INF 对大小写不敏感，行注释语句

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

命令是“;”，类似 Visual Basic 里的“’”。如果一行写不下，使用“\”来换行。更多关于 INF 文件的规则和说明可以查看 Windows DDK 的说明，这里不再详述。

通过 INF 文件，用户可以允许系统自动安装信号采集设备的驱动程序，或者在对话支持下按照系统安装向导一步一步完成驱动程序安装过程。

13.6 FPGA 内部结构设计

13.6.1 构框图

基于 FPGA 及 PCI9054 的信号采集卡的核心设计部分是 FPGA 内部结构的逻辑设计。如图 13.15 所示为本系统 FPGA 内部结构框图。

其中，信号接收器是接收采集信号的模块，对 LVTTL 信号接口和 LVDS 信号接口输入的信号进行分析，并将获得的信号送入下一个模块。

内部信号源是一个用于测试的模块。通过产生一个已知的有序的信号序列送至信号接收器，最终进入主机后进行检测，即可得知整个传输过程的正确性。

双口 RAM 用于实现信号的串并转换，使输入的信号转换成 32 位数据位宽，同时实现时钟域的转换。另外通过乒乓操作实现数据的有序存取。

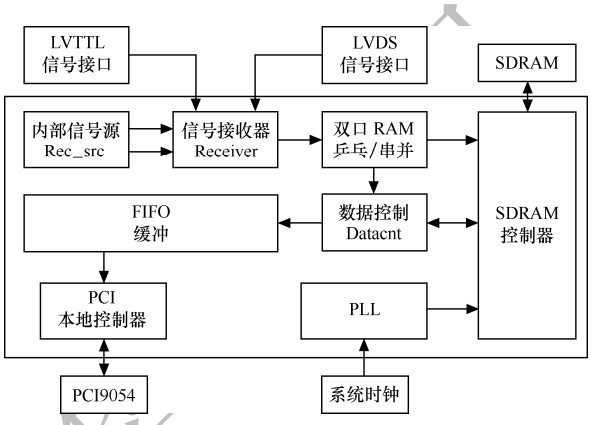


图 13.15 FPGA 内部结构框图

数据控制模块用于产生双口 RAM 和 SDRAM 控制器的地址总线、数据总线和控制总线，是双口 RAM 模块与 SDRAM 控制器的桥梁。实现双口 RAM 至 SDRAM 控制器正确的数据转移。同时，数据控制模块还负责从 SDRAM 控制器读取数据，并送至 FPGA 内的 FIFO 缓冲区中，用于 PCI 总线交互。

SDRAM 控制器是 FPGA 内用于外部 SDRAM 控制的模块。

PCI 本地控制器是 FPGA 内用于控制 PCI9054 本地总线交互的模块。

PLL 是 Altera FPGA 内提供的模拟延迟锁相环模块，可以实现系统时钟的倍频、分频及延迟等时钟控制操作。通过该模块可以实现系统内不同时钟域的时钟分配。

13.6.2 设计方法

在实际的设计中，主要通过自行编写模块、调用宏模块和引用开源模块 3 种方式来实现模块的设计。

1．自行编写模块

自行编写模块就是完全通过自行编写的逻辑完成模块的设计，可以使用硬件编程语言、状态机、电路图等方式实现。

本设计中的 PCI 本地控制器、信号接收器、内部信号源、数据控制等模块都是使用 Verilog 硬件语言自行

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

编写设计的。

2．调用宏模块

在逻辑设计中，Altera 已经为设计者提供了丰富的基本逻辑组件的模块。设计者只需使用宏模块向导进行调用，配置相应的参数，即可得到符合设计需求的模块。

本设计中的双口 RAM、PLL 及 FIFO 缓冲模块正是通过调用 Cyclone FPGA 支持的相应的 LPM_RAM_DP、ALTPLL 和 LPM_FIFO+宏模块实现的。如图 13.16 所示为 Altera 宏模块向导管理器界面。

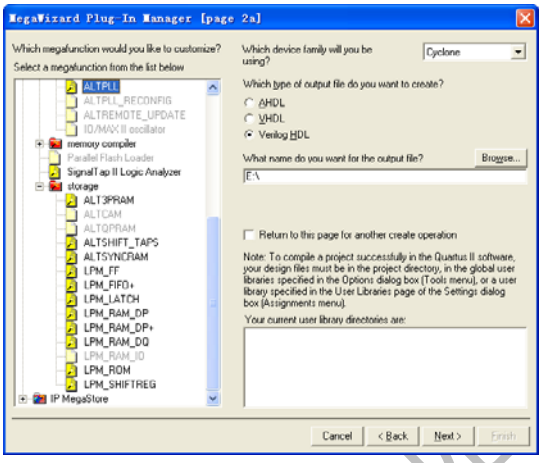


图 13.16　Altera 宏模块向导管理器

3．引用开源模块

除了设计者自行编写模块及宏模块向导管理器中提供的模块以外，设计者还可以引用开源的模块加入到系统的设计中。开源的模块往往是一些厂家对一些常见的外围器件提供的控制模块。

本设计中使用的 SDRAM 控制器正是引用了成熟的开源模块进行的设计，Altera 为 SDRAM 控制器给出了白皮书和开源模块。设计者只需将该开源模块引用至系统的设计中，进行一定的修改即可使用。

13.7　硬件系统实现

13.7.1　FPGA 配置

FPGA 作可编程器件，可以根据用户的需要进行现场可编程。为此，本系统实现了两种编程配置方式。

一种是直接对 FPGA 进行编程，使用 JTAG 模式，在 QuartusII 工具中输出 SOF 文件（SRAM Object File）。其好处是编程速度快，并且由于是对 FPGA 的 SRAM 结构进行编程，编程次数要多得多，但是掉电后，SRAM 保存的编程信息将会丢失。如图 13.17 所示为 JTAG 配置模式的硬件连接方式。

另一种是通过对配置芯片进行编程实现（AS 模式）。配置芯片具有掉电保存能力，在系统上电时，FPGA 首先从配置芯片中读取编程数据，并对 FPGA 进行加载。对配置芯片的编程使用 Active Serial 编程模式，在 QuartusII 工具中输出 POF 文件（Programmer Object File）。如图 13.18 所示是 AS 配置模式的硬件连接方式。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

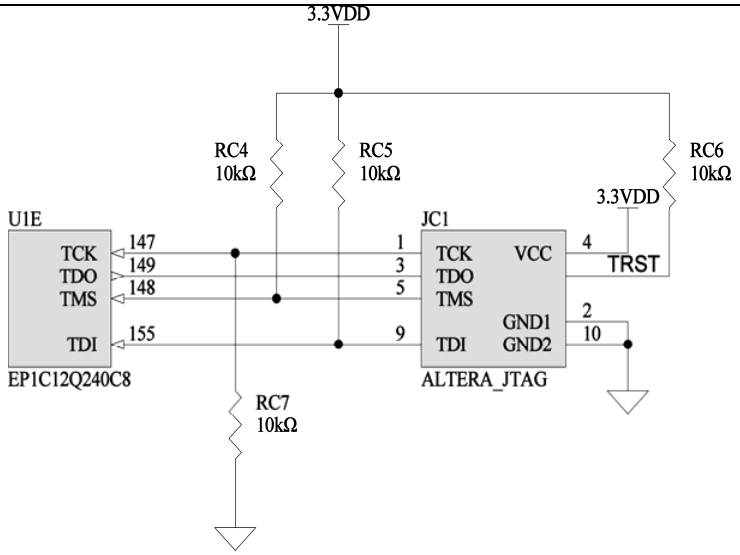


图 13.17　JTAG 模式硬件连接方式

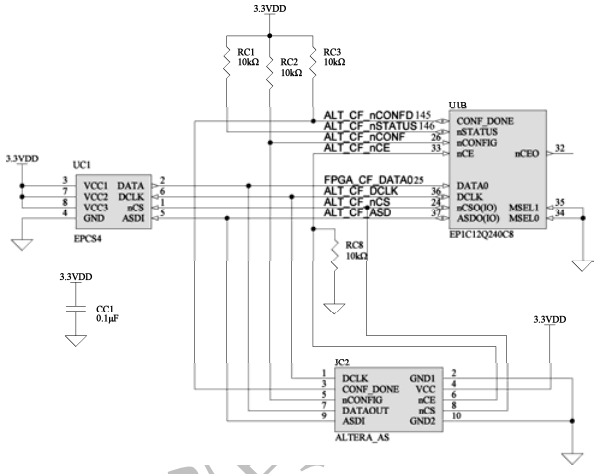


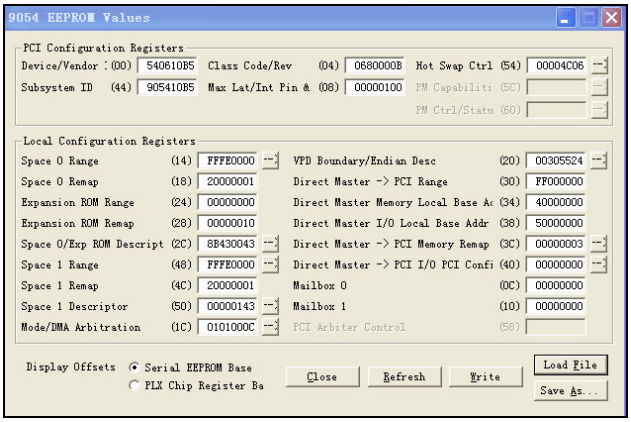
图 13.18　AS 模式硬件连接方式

13.7.2　PCI9054 配置

PCI9054 上电时，需要配置其内部的 PCI 配置寄存器和本地配置寄存器以及其他相关寄存器，用来描述 PCI9054 在系统中的操作状态。

相关的配置信息使用 EEPROM 来进行存储，EEPROM 可以通过编程器进行编程，也可以使用 PLX SDK 中的 PLXMon 工具进行编程。该工具可将 EEPROM 的配置信息导入/导出为 EEP 文件，便于配置信息的保存和交换。

如图 13.19 所示是 PCI9054 接口芯片的配置信息。



华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

图 13.19　PCI9054 接口芯片配置信息

EEP 文件是一个按照 EEPROM 寄存器顺序保存寄存器值的文件，对应于上面的配置信息，使用 ultraedit 软件查看该 EEP 文件可以看到如图 13.20 所示的信息。

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 06 54 B5 10 80 06 0B 00 00 00 00 01 00 00 00 00 ;
00000010h: 00 00 00 00 FE FF 00 00 00 20 01 00 01 01 0C 00 ;
00000020h: 30 00 24 55 00 00 00 00 00 00 10 00 43 8B 43 00 ;
00000030h: 00 FF 00 00 00 40 00 00 00 50 00 00 00 00 03 00 ;
00000040h: 00 00 00 00 54 90 B5 10 FE FF 00 00 00 20 01 00 ;
00000050h: 00 00 43 01 00 00 06 4C                         ;
```

图 13.20　EEP 文件信息

13.7.3　PCI9054 PCI 总线连接规范

开发基于 PCI 总线的信号采集设备时，必须按照 PCI 局部总线的电气规范对信号采集设备进行设计。下面介绍几个比较重要的电气规范。

1．5V 与 3.3V 信号环境

在 PC 环境中 PCI 局部总线的电气规范提供了 5V 和 3.3V 两种信号环境，这个从主板上 PCI 总线连接器上的分隔位置就可以看出。根据两种信号环境，PCI 信号采集设备的设计也要进行相应的调整。
PCI 局部总线还提供了通用板的设计方案，可以同时支持 5V 和 3.3V 信号环境。根据这种方案设计的信号采集设备在连接到不同信号环境下的 PCI 总线连接器时，就可以自适应调整设备上的供电需求。

2．扩展板技术指标

这里只描述扩展板（即信号采集设备）上的技术指标，母版的技术指标属于主板上的设计规范，有兴趣的读者可以查阅 PCI 总线规范。
首先，扩展板要指明板上电源的最大功耗，可以通过连接器上的 PRSNT1#和 PRSNT2#两个信号进行配置。如表 13.4 所示为扩展板功耗选择表。

表 13.4　　扩展板功耗选择表

PRSNT1#	PRSNT2#	扩展板功率配置
开路	开路	不存在扩展板
地	开路	最大功耗 25W
开路	地	最大功耗 15W
地	地	最大功耗 7.5W

其次，需要根据通用板的 PCI 连接器引脚分配表将 PCI9054 与连接器正确进行连接。需要注意的是，5V、3.3V 和通用板的连接器引脚分配是不一样的，这是进行 PCB 设计时需要特别注意的。
另外，PCI 局部总线规范还对走线长度做了一些限制。例如所有 32 位接口信号的最大长度走线长度为 1.5inch；时钟 CLK 的走线长度为 2.5±0.1inch，并且只能连接至一个负载上（在本案例中，只能连接至 PCI9054 上）。

13.7.4　电源系统

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

电源系统的地位十分重要，但往往设计者对电源系统的重视程度不够。一个合理、稳定的电源系统可以大大减少系统故障的发生率。设计电源时应考虑以下因素。

- 电源系统输出的电压、电流、功率等因素。
- 电源系统输入的电压、电流。
- 电源的稳定性因素。
- 电源系统的输出波纹。
- 电源系统的兼容性。
- 电磁干扰因素。
- 电源系统的体积限制。
- 电源系统功耗限制。
- 电源系统的成本因素。

以上可以看出，设计一个好的电源系统需要考虑很多因素。

在本系统的电源系统中，由于是 PCI 设备，因此 5V 和 3.3V 都可以从 PCI 连接器上取得。而 FPGA 内核需要的 1.5V 电压（范围是 1.425V～1.575V）则采用低压差线性稳压器（LDO）进行设计。

线性电源模块（LDO）优点是成本低、噪声小以及静态电流小等。同时它需要的外围电路很简单，通常只需要一两个旁路电容即可。图 13.21 所示是 LDO 模块应用示意图。

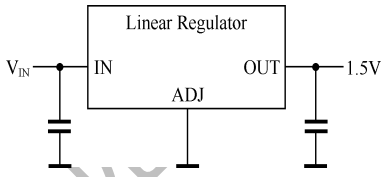


图 13.21　LDO 模块应用示意图

如果采用固定输出电源芯片，那么选择的范围就缩小了，只有个别芯片提供 1.5V 的输出，大部分电源芯片的固定输出是 1.8V、2.5V 和 3.3V，但是都具有可调电压的型号。其应用电路连接如图 13.22 所示。

VREF 一般是 1.25V，$I_{ADJ}\times R_1$ 可以忽略。假定输入 V_{IN} 为 5V，V_{OUT} 为 1.5V，那么 R1/R2=1/5，而 R1 一般要求 100~150Ω，那么可以选 R1＝100Ω，R2＝500Ω。如果采用了固定电平输出的芯片，只需要把 R2 焊 0Ω，R1 开路即可。

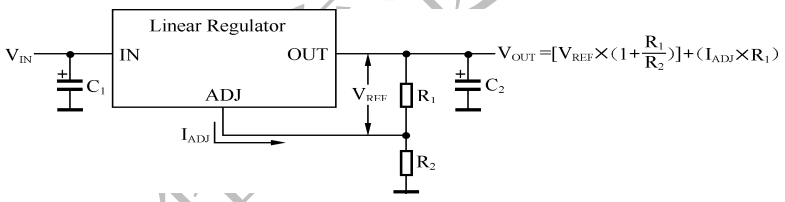


图 13.22　LDO 可调模块应用示意图

13.8　样机的调试方法和技巧

不管是复杂的电子系统还是简单的电路，样机的调试都是有一些基本步骤的。对于本案例的信号采集设备同样如此。

最先进行的就是电源系统的调试，包括是否有短路、断路，是否有虚焊，各电压系统是否正常，电源模块输出电流是否足够驱动负载等。只有电源系统正常工作，才能谈得上实现系统功能。

完成了电源系统的调试后，PCI 信号采集卡的下一个工作就是对 PCI954 进行配置。用户可以通过编程器或者 PLXMon 工具在线进行烧写。这个配置过程保证主机操作系统能够识别出硬件设备。

主机能够正确地识别硬件设备后，需要进行设备驱动程序的安装。

通过上述步骤，主机已经可以正确地识别设备，并为用户提供了可供操作的 API 接口。接下来要做的调试工作就是硬件工程师要完成的 FPGA 逻辑设计和软件工程师要完成的应用软件设计。但这两部分并不能绝对分开进行设计，因为不管是对设备的 I/O 操作还是 DMA 操作都需要进行地址空间映射，因此都需要在软件和硬件上进行相应的设计。

在本系统的 FPGA 调试和软件联调中，我们使用了两个简单的例子来验证系统是否能够正常的工作。一个是直接用 FPGA 控制的跑马灯实例，另一个是通过应用软件控制的跑马灯实例。

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

13.8.1 没有 PCI 控制的跑马灯

本实例仅用 FPGA 内部逻辑对 LED 进行控制，实现跑马灯。通过本实例可以检验 FPGA 的两种配置模式是否正常工作，同时可以学习使用 QuartusII 软件开发和配置 FPGA 的流程。

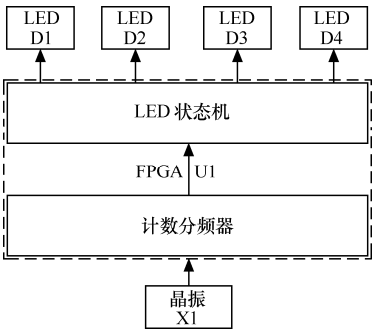


图 13.23 跑马灯系统框图

如图 13.23 是本实例的系统框图。

其中计数分频器是通过计数器将晶振的 40MHz 时钟进行分频，产生的低频时钟作为 LED 状态机的控制时钟。LED 状态机使用 4 个状态分别表示 4 个 LED 的亮灭，状态转换通过状态机的时钟上升沿触发，实现跑马灯。

下面是这个实例 FPGA 内的主要 Verilog 源代码。

```
    //--        计        数        分        频
-------------------------------------
    always @ (posedge clkin)
        count <= count+1;                //分频计数器
    assign clk = count[24];              //状态机时钟
    //--LED 控制--------------------------------
    always @ (posedge clk) begin
        case(state)                      //跑马灯状态控制量
             2'd0:  led <= 4'b0001;          //D1 亮
             2'd1:  led <= 4'b0010;          //D2 亮
             2'd2:  led <= 4'b0100;          //D3 亮
             2'd3:  led <= 4'b1000;          //D4 亮
        endcase
        state <= state+1;               //跑马灯状态变化
    end
```

13.8.2 基于 PCI 控制的跑马灯

本实例使用应用程序通过 PCI 总线（PCI9054）来对 FPGA 内部逻辑参数进行配置，实现对 LED 的控制。通过本实例可以检验开发板的驱动以及 PCI 的实时控制，同时学习应用程序和 PCI9054 进行通信的方法。

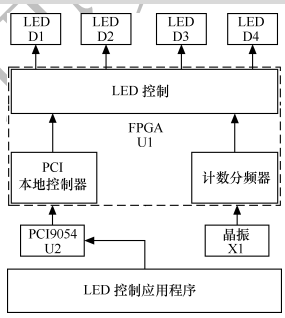


图 13.24 基于 PCI 控制的跑马灯系统框图

如图 13.24 所示为本实例的系统框图。

其中 PCI 本地控制器实现与 PCI9054 的接口通信，获取 LED 控制应用程序发出的触发信号。LED 控制器包含 LED 状态机并接受 PCI 本地控制器的触发信号，通过判别 LED 触发的类型决定启动单个 LED 控制或 LED 状态机。

下面是这个范例 FPGA 内的主要 Verilog 源代码。

```
    //--计数分频-------------------------------------
    always @ (posedge clkin)
        count <= count+1;                           //分频计数器
    assign clk = count[24];                         //状态机时钟
    //--LED 控制--------------------------------
    always @ (posedge clk) begin
        case(LEDByControl)                          //状态控制量
            8'd1:   led <= 4'b0001;                 //PCI 控制 D1 亮
            8'd2:   led <= 4'b0010;                 //PCI 控制 D2 亮
            8'd3:   led <= 4'b0100;                 //PCI 控制 D3 亮
            8'd4:   led <= 4'b1000;                 //PCI 控制 D4 亮
            8'd5:   case(state)                     //跑马灯状态控制量
                        2'b00:  led <= 4'b0001;     //跑马灯 D1 亮
                        2'b01:  led <= 4'b0010;     //跑马灯 D2 亮
                        2'b10:  led <= 4'b0100;     //跑马灯 D3 亮
                        2'b11:  led <= 4'b1000;     //跑马灯 D4 亮
                        endcase
            default: led <= 4'b0000;                //跑马灯全灭
            endcase
        state <= state+1;                           //跑马灯状态变化
    end
```

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

```
    //--PCI 本地控制器-----------------------------
    always @(posedge clkin)
        lholda <= lhold;                              //PCI 本地总线占用请求及应答
    always @(posedge clkin or negedge lrst) begin
        if(!lrst) begin                          //PCI9054 复位
            readyflag <= 1'b0;
            LEDByControl[7:0] <= 8'b0;
        end
        else if(!ads & lwr & blast & la[13:0] == 14'b1)
            readyflag <= 1'b1;                  //开始发起 LED 控制字 IO 配置
        else if(ads & lwr & blast & readyflag == 1'b1 & la[13:0] == 14'b1)
            readyflag <= 1'b1;                 //捕获发起操作
        else if(ads & lwr & !blast & readyflag == 1'b1 & la[13:0] == 14'b1) begin
            LEDByControl[7:0] <= ld[31:24];//进行 LED 控制字 IO 配置操作
            readyflag <= 1'b0;
            end
        else    begin
            LEDByControl[7:0] <= LEDByControl[7:0];//保持 LED 控制字
            readyflag <= 1'b0;
        end
    end
```

在进行 FPGA 的逻辑设计时，采用一种叫做虚拟逻辑分析仪的技术，可以增强我们诊断系统的能力。Altera FPGA 提供的虚拟逻辑分析仪称为 SignalTap，Xilinx 中也有同样功能的产品称为 ChipScope。

SignalTap II 逻辑分析仪是 Altera 第二代系统级调试工具，能够获取、显示可编程片上系统（SOPC）的实时信号，帮助工程师在其系统设计中观察硬件和软件的交互作用。在可编程逻辑市场上，SignalTap II 逻辑分析仪专用于 Quartus II 软件，与其他嵌入式逻辑分析仪相比，它支持的通道数最多，抽样深度最大，时钟速率最高。

如图 13.25 所示为控制其中一个 LED 时的 PCI 本地时序图。

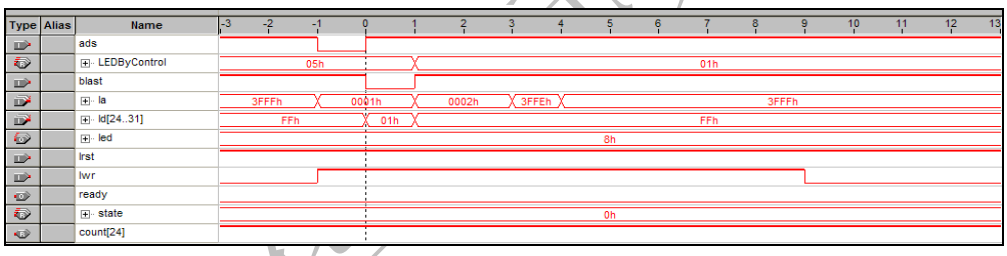


图 13.25　控制 1 个 LED 的 PCI 本地时序图

如图 13.26 所示为控制 4 个 LED 进行跑马灯时的 PCI 本地时序图。

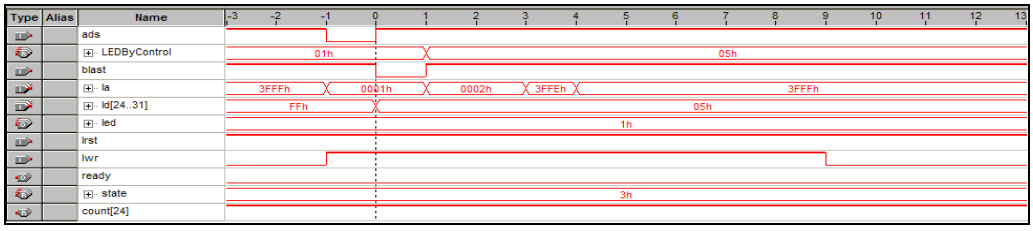


图 13.26　控制 4 个 LED 跑马灯的 PCI 本地时序图

13.9　产品稳定性和可靠性测试

完成一个产品的设计后和初步调试后，就可以对产品进行完整的测试流程。一般来说，对产品需要进行下面一些测试，通过测试后才能对产品的稳定性和可靠性得出一个结论。

1．指标测试

根据产品设计需求提出的指标进行测试。这是最基本的测试，只有通过这个测试，产品才是设计所需要的

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

产品。
在基于 FPGA 和 PCI 的信号采集系统中，指标测试主要包括：输入口的采集速度（即数字信号的频率）、FPGA 的工作频率、存储器的工作频率、PCI9054 的传输带宽等。

2．功能测试

根据不同的用户需求，FPGA 内会有不同的逻辑实现，在同一款产品上就能实现不同的功能。因此用户应根据自己的需求对产品做相应的功能测试。
在本实例系统中，功能测试主要包括软件对设备 I/O 操作及 DMA 操作的不同地址映射测试。主要是验证 FPGA 逻辑与软件设计是否正确。

3．容限测试

容限测试主要针对产品的一些关键指标进行极限测试，得到产品在一些极限条件下的工作状态。在信号采集设备中主要是对系统的采集速度及精度做容限测试。在一些情况下还会做功耗测试。

4．容错测试

容错测试是一种对抗性的测试过程。目的在于当系统出现故障时，如何进行故障的转移与恢复有用的数据。故障转移是确保测试对象在出现故障时，能成功地将运行的系统或系统某一关键部分转移到其他设备上继续运行，即备用系统就将不失时机地“顶替”发生故障的系统，以避免丢失任何数据或事务，不影响用户的使用。
要进行故障转移的全面测试，一个好的方法是将测试系统全部对象用一张系统结构图描绘出来，对图中的所有可能发生的故障点设计测试用例。例如，系统设计架构图中，如果存在单点失效的关键对象，就是设计的重大缺陷。

5．长时间验证测试

一个系统除了能够完成特定的功能，达到设计的指标外，还应该能够长时间稳定运行。因此，长时间验证测试也是测试流程中不可或缺的一部分。

6．一致性测试

在产品的设计，尤其是硬件产品的设计中，往往需要经过几个样品的设计才能最终得到满足设计需求的产品。在完成产品定型前还要进行产品的一致性测试，也就是在同等环境条件下，对产品进行同等操作，得到的结果应该是符合一致性要求的。只有符合一致性要求，产品才能定型，也才能进入批量生产的阶段。

13.10　产品定型和设计文档备案

通过测试验收后的样品即可进行产品的定型和设计文档备案工作。
产品的定型包括系统的基本组成、基本配置及互联方法，运行环境，硬件整体系统及各分系统的基本功能和主要性能指标，功能模块的划分，关键技术的使用，采购的硬件器件名称型号、生产单位、主要技术指标，主要仪器设备，电源、工艺结构设计等。

华清远见
华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用　开发板免费领取试用

对于本实例的 PCI 信号采集设备来说，具体还有单板的尺寸、单板与其他系统互联方法及标准、单板器件选型、制板加工焊接参数、性能指标及功耗、开发用仪器仪表等。也就是说，定型后的产品可以进行批量的生产，并能够获得等同效果的测试结果。

同时，一个完整的产品设计还要包括一套完善的设计文档。基于 PCI 的信号采集设备一般来说包括软硬件两部分才能实现具体的功能。因此，设计文档也包含两部分的内容。

一套完整的设计文档主要包括以下内容。

1．硬件需求说明书

主要包括硬件开发目的、基本功能、基本配置、主要性能指标、运行环境、约束条件、开发经费及进度等。

2．硬件总体设计报告

主要包括硬件系统总体结构及功能划分，系统逻辑框图，组成系统各功能模块框图，系统电路结构图及单板组成，单板逻辑框图及电路结构图以及测试方案等。

3．单板总体设计方案

主要包括单板在整机中的位置，开发目的及主要功能，单板功能描述，单板逻辑框图及功能模块说明，单板软件功能描述及功能模块划分，接口简单定义及相关板关系，主要性能指标，功耗及采用标准等。

4．单板硬件详细设计

主要包括单板逻辑框图及各功能模块详细说明，各功能模块实现方式，地址分配，控制方式，接口方式，存储器空间，中断方式，接口管脚定义，时序说明，性能指标，指示灯说明，可编程器件图，功能模块说明，原理图，物料清单及单板测试，调试计划等。

5．单板软件详细设计

主要包括编程语言，编译环境，详细的设计细节（包括中断、主程序、子程序、入口参数、出口参数、局部变量、函数调用和流程图等），相关协议等。

6．单板硬件调试文档

主要包括单板硬件功能模块划分，单板硬件各模块调试进度，调试中出现的问题及解决方法，原始数据记录 ，系统方案修改说明，单板方案修改说明，器件更换说明，原理图、PCB 修改说明，可编程器件修改说明，调试工作阶段总结，调试进展说明，下阶段调试计划及测试方案修改等。

7．单板软件调试文档

主要包括单板软件模块划分及个功能模块调试进度，单板软件调试中出现的问题及解决方法，下阶段的调

华清远见教育集团：www.hqyj.com 旗下品牌：www.embedu.org（嵌入式学院） www.farsight.com.cn（企业学院） 北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州
开发板免费领取试用 开发板免费领取试用

试计划及测试方案修改等。

8．单板系统联调文档

主要包括系统功能模块划分，系统功能模块调试进展，系统接口信号的测试原始记录及分析，系统联调中出现的问题及解决方法，系统联调性能评估等。

9．硬件信息库

主要包括硬件设计工程文件，软件设计工程文件，驱动设计文件，硬件 BOM 文件，制板文件，焊接清单，相关开发工具及环境配置，主要参考技术手册及标准等。

联系方式

集团官网：www.hqyj.com　　嵌入式学院：www.embedu.org　　移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn　　物联网学院：www.topsight.cn　　研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路银海大厦 A 座 8 层，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218

华清远见

华清远见教育集团：www.hqyj.com　旗下品牌：www.embedu.org（嵌入式学院）　www.farsight.com.cn（企业学院）　北京 | 上海 | 深圳 | 成都 | 南京 | 武汉 | 西安 | 广州

开发板免费领取试用　开发板免费领取试用