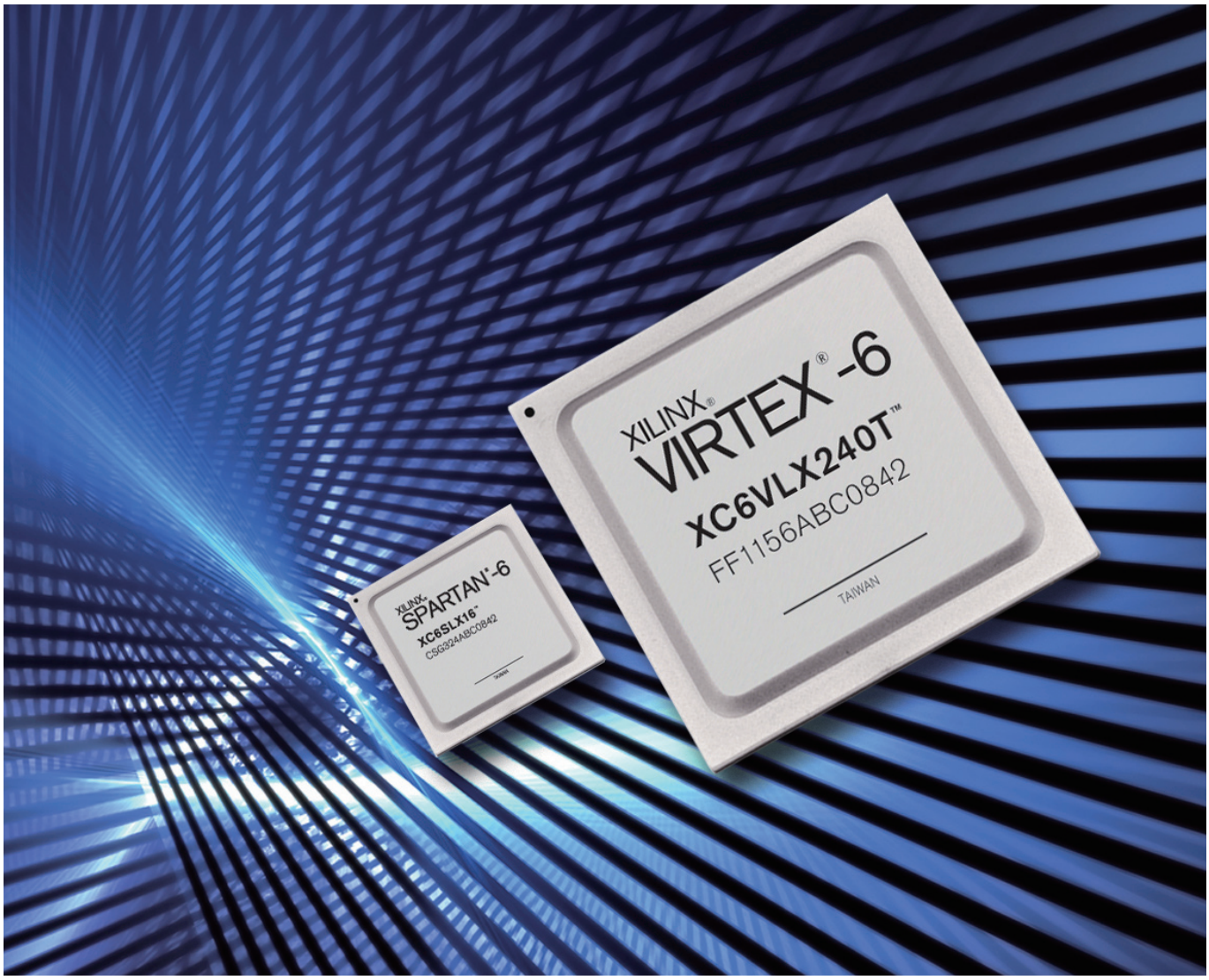
《电子工程师创新设计必备宝典系列之FPGA开发全攻略》



FPGA开发全攻略——

工程师创新设计宝典

下册

技巧篇

2009年2月 1.0版

前言

张国斌
电子书主编
2009年2月25日

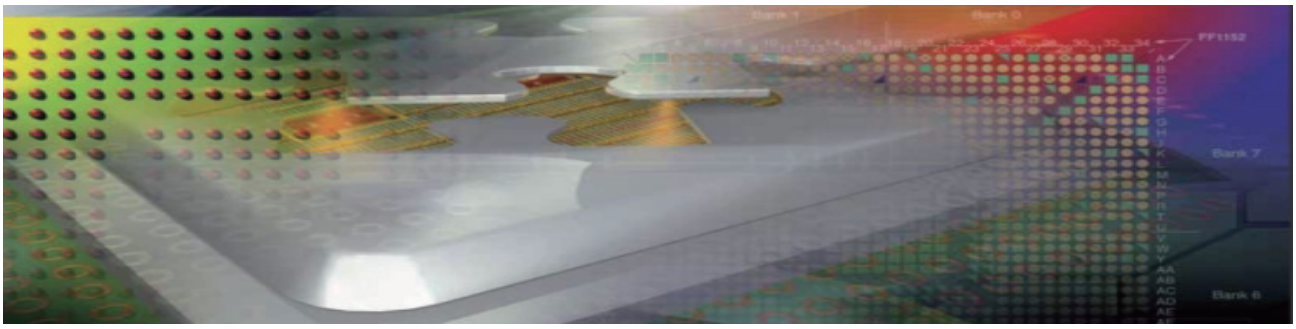
2008年，我参加了几次可编程器件供应商举办的技术研讨会，让我留下深刻印象的是参加这些研讨会的工程师人数之多，简直可以用爆满来形容，很多工程师聚精会神地全天听讲，很少出现吃完午饭就闪人的现象，而且工程师们对研讨会上展出的基于可编程器件的通信、消费电子、医疗电子、工业等解决方案也有浓厚的兴趣，这和其他器件研讨会形成了鲜明的对比。

Garnter和iSuppli公布的数据显示：2008年，全球半导体整体销售出现25年以来首次萎缩现象，但是，可编程器件却还在保持了增长，预计2008年可编程逻辑器件(PLD)市场销售额增长7.6%，可编程器件的领头羊美国供应商赛灵思公司2008年营业收入预计升6.5%！在全球经济危机的背景下，这是非常骄人的业绩！也足见可编程器件在应用领域的热度没有受到经济危机的影响！这可能也解释了为什么那么多工程师对可编程器件感兴趣吧。

在与工程师的交流中，我发现，很多工程师非常需要普及以FPGA为代表的可编程器件的应用开发知识，也有很多工程师苦于进阶无门，缺乏专业、权威性的指导，在Google上搜索后，我发现很少有帮助工程师设计的FPGA电子书，即使有也只是介绍一些概念性的基础知识，缺乏实用性和系统性，于是，我萌生了出版一本指导工程师FPGA应用开发电子书的想法，而且这个电子书要突出实用性，让大家都可以免费下载，并提供许多技巧和资源信息，很高兴美国赛灵思公司对这个想法给予了大力支持，赛灵思公司亚太区市场经理张俊伟小姐和高级产品经理梁晓明先生对电子书提出了宝贵的意见，并提供了大量FPGA设计资源，也介绍了一些FPGA设计高手参与了电子书的编撰，很短的时间内，一个电子书项目团队组建起来，北京邮电大学的研究生田耘先生和赛灵思公司上海办事处的苏同麒先生等人都参与了电子书的编写，他们是有丰富设计经验的高手，在大家的共同努力下，这本凝结着智慧的FPGA电子书终于和大家见面了！我希望这本电子书可以成为对FPGA有兴趣或正在使用FPGA进行开发的工程师的手头设计宝典之一，也希望这个电子书可以对工程师们学习FPGA开发和进阶有实用的帮助！如果可能，未来我们还将出版后续版本！

目录

第六章、FPGA应用开发实例



6.1 如何克服 FPGA I/O 引脚分配挑战

作者：Brian Jackson

产品营销经理Xilinx, Inc.

brian.jackson@xilinx.com

对于需要在 PCB 板上使用大规模 FPGA 器件的设计人员来说，I/O 引脚分配是必须面对的众多挑战之一。由于众多原因,许多设计人员发表为大型 FPGA 器件和高级 BGA 封装确定 I/O 引脚配置或布局方案越来越困难。但是组合运用多种智能 I/O 规划工具，能够使引脚分配过程变得更轻松。

在 PCB 上定义 FPGA 器件的 I/O 引脚布局是一项艰巨的设计挑战，即可能帮助设计快速完成，也有可能造成设计失败。 在此过程中必须平衡 FPGA 和 PCB 两方面的要求，同时还要并行完成两者的设计。 如果仅仅针对 PCB 或 FPGA 进行引脚布局优化，那么可能在另一方面引起设计问题。

为了解引脚分配所引起的后果，需要以可视化形式显示出 PCB 布局和 FPGA 物理器件引脚，以及内部 FPGA I/O 点和相关资源。 不幸的是，到今天为止还没有单个工具或方法能够同时满足所有这些协同设计需求。

然而，可以结合不同的技术和策略来优化引脚规划流程并积极采用 Xilinx® PinAhead 技术等新协同设计工具来发展出一套有效的引脚分配和布局方法。 赛灵思公司在 ISE™ 软件设计套件 10.1 版中包含了 PinAhead。

赛灵思公司开发了一种规则驱动的方法。首先根据 PCB 和 FPGA 设计要求定义一套初始引脚布局，这样利用与最终版本非常接近的引脚布局设计小组就可以尽可能早地开始各自的设计流程。 如果在设计流程的后期由于 PCB 布线或内部 FPGA 性能问题而需要进行调整，在采用这一方法晨这些问题通常也已经局部化了，只需要在 PCB 或 FPGA 设计中进行很小的设计修改。

步骤1： 评估设计参数

那么，从哪里开始呢？ 首先应当尽早制定 I/O 分配策略。 但没有优化工具或完整的网表，完成这一任务可能很困难。

首先，让我们先回答几个问题来确定 PCB 物理参数和限制：

PCB 板有几层、走线宽度以及过孔尺寸多大?

PCB 参数对可使用的 FPGA 封装类型 (如 BGA) 有限制吗?

PCB 上有没有 FPGA 必须使用的固定接口位置? 其它芯片、连接器或布局限制?

哪些高速接口需要特别关注?

能否将布局策略可视化，从而保证最短互连?

你会发现画一张 PCB 布局图很有帮助。PCB 布局图上应当包括所有主要元器件以及关键接口和总线，从而可以帮助确定最佳的 FPGA 引脚分配。 请注意将元器件画在 PCB 板的实际安装面上。 标注出需要特别关注的接口，如高速总线和差分对 (图 1)。

下一步，检查 FPGA 器件的布局来了解芯片上的物理资源所在。 列出设计中使用的不同电压和时钟，开始隔离设计需要的接口。 然后确定设计是否使用特殊的 I/O 接口资源，如千兆收发器 (GT)、BUFR、IODELAY 以及数字时钟管理器。这些资源可能需要将有关的 I/O 引脚布署得尽量互相靠近。

现在需要确定设计中使用的 PowerPC™、DSP48 和 RAM16 等 FPGA 资源的位置。 将连接到 I/O 组的任何相关 I/O 尽量置于尽相关资源最近的地方。 然后看一下能否将某些 I/O 信号组合到接口，这对于引脚分配很有帮助。 最后，确定 FPGA 的配置模式。

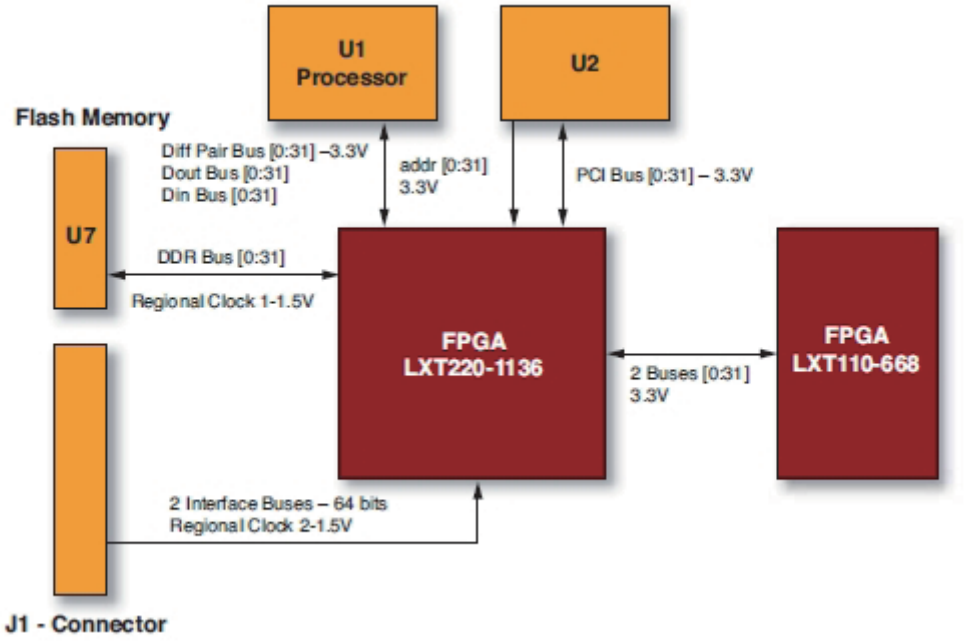


图6–1 PCB连接图

步骤2： 定义引脚布局要求

一旦了解了主要的 FPGA 接口并创建了物理布局的原型，就可以定义引脚布局了。 有些设计人员喜欢使用包含所有 I/O 信号数据表来保持与引脚的对应。 你可以按电压、时钟、接口或总线对它们进行分组。 这一方法

确实非常有用，因为它可帮助你将信号组合成组，从而在分配引脚时可以按组进行。 这一阶段，你还会发现为了实现最优 PCB 布线，有些关键接口必须置于器件的某个边，或者利用外部物理引脚。

在考虑到 FPGA 和 PCB 要求并确定了主要的接口位置以后，下一步是根据所有这些条件将引脚分配给 I/O 组。 这也是真正开始工作的地方。 在当前的设计流程中，引脚分配时一项耗费时间的任务，在解决任何性能和信号完整性问题的过程中可能会涉及许多尝试和错误。 传统上，设计人员都是徒手画图来完成这项任务的，因为 EDA 和芯片供应商没有提供帮助设计人员将 FPGA 和 PCB 引脚布局可视化的工具。

但现在赛灵思公司提供了相应的工具。 在 ISE Foundation™ 软件工具 10.1 版本中包含的 PlanAhead Lite 是 PlanAhead™ 设计、分析和平面布局工具的简化版。 其中包括的针对 PCB 和 FPGA 设计的 PinAhead 的工具使得 I/O 引脚配置更为容易。这里我们不打算详细介绍该工具的所有细节，而只是看一下如何将其用于 I/O 引脚分配。 如果你希望了解 PinAhead 的更详细信息，包括视频展示和教程，请访问 www.xilinx.com/planahead 。

步骤3： 利用PinAhead进行引脚分配

PinAhead 环境提供了一组不同的视图。利用这些视图可以帮助完成 I/O 端口信息与物理封装引脚或裸片 I/O 盘 (Pad) 的对应和分配 (图 2)。 PinAhead 的图形环境与 PlanAhead 类似，在器件视图中清晰地显示出芯片上的 I/O 盘和相关资源，并在封装视图中显示出物理器件引脚。 视图同时显示出 I/O 端口和物理引脚信息，这样可以通过交叉选取来试探逻辑设计和物理器件资源的对应。

你可以在没有设计网表的情况下使用 PinAhead 来尝试器件资源，或者直接开始 I/O 引脚规划流程。 封装引脚视图 (“Package Pins”view) 根据器件数据表列出了器件封装技术参数，因此大多数情况下在进行引脚配置时都不再需要去参考器件数据手册。 封装引脚视图以列表形式对 I/O 组 (bank) 进行了分类，因此可以同时在器件和封装视图中交叉选择和高亮显示 I/O 组。 视频清晰显示出物理引脚位置和裸片中的 I/O 盘的关系，从而简化了 I/O 组的优化选择。 封装引脚视频还显示了 I/O 组中每一引脚的信息。

你可以利用 PinAhead 接口从头开始创建 I/O 端口，也可以从 CSV 格式数据表、HDL 源文件头或综合后的网络和 UCF 格式约束文件中导入 I/O 端口。 I/O 端口视图 (“I/O Ports”view) 显示出设计中定义的所有 I/O 端口信号，总线文件夹则显示分组的总线和差分对信号。

你可以按不同方式对封装引脚和 I/O 端口视图进行排序。 可以切换列表视图显示基于分类的列表或全部列表，或者点击鼠标对封装引脚视图进行排序，显示所有可用的全局时钟或地区时钟引脚。 同时还可以将信息导出到 CSV 格式数据表，做为引脚配置的出发点。

PinAhead 还提供了一个界面，支持有选择地禁止 PinAhead 将 I/O 端口分配给某些 I/O 引脚、I/O 引脚组或 I/O 组。 可以在封装引脚、器件或封装视图中选择和禁止引脚。 例如，你可以对封装引脚视图 (Package Pins view) 排序并禁止所有 VREF 引脚。

PinAhead 允许将相关的 I/O 端口和总线组合为“接口”(interface)。 这样组合使你可以将相关 I/O 端口做为单个实体处理，从而简化了 I/O 端口管理和分配任务。 接口组合功能可以更容易地可视化显示和管理与特定逻辑接口相关联的所有信号。

可方便地在设计间拷贝接口，或者利用接口组合生成特定接口的 PCB 原理图符号。 组合后的接口在 I/O 端口视图中以可扩展文件夹的形式出现，通过在视图中选择 I/O 端口并将其拖动到接口文件夹，可以将额外的 I/O 端口添加到接口组合中。

当创建 I/O 端口时，可将其分配到封装引脚或 I/O 盘 (pad)。 在此之前，最好先检查一下 I/O 端口的最初 PCB 互连草图并与 PCB 设计人员协商，了解布放不同 I/O 端口接口的相关位置和其它需考虑的因素。 适当的总线顺序和边缘距离有 PCB 布线非常有帮助，可以大大节约设计时间。

通过将单个引脚、总线和接口拖动到器件或封装视图，可以将它们分配到 I/O 引脚。 利用不同的分配模式，可以将引脚组分配给选定的 I/O 引脚。 可用的模式包括“Place I/O Ports in an I/O Bank,”“Place I/O Ports in Area,”以及“Place I/O Ports Sequentially.”。

每种模式提供了将 I/O 端口分配到引脚的不同分配方式。 利用这些模式，可以通过鼠标光标处弹出的窗口了解你所分配的端口数量信息。 直到分配了所有选定 I/O 端口之前，这一模式一直保持。

器件视图 (Device view) 以图形方式显示所有时钟区域和时钟相关的逻辑对象，从而使时钟相关的 I/O 分配更容易、更直观。 选择一个时钟区将会显示所有 I/O 组、时钟相关的资源以及与其相关的器件资源。

PlanAhead 软件试图保证你在引脚分配时始终符合规则。 在你的指引下，PlanAhead 工具将差分以端口分配给适当的引脚对。 当交互式指定 I/O 端口时，工具会运行规则检查 (DRC) 来保证布局是合乎规则的。

工具缺省设置运行在交互 DRC 模式，当然你也可以选择关闭这一模式。 工具会检查电压冲突、VREF 引脚或 I/O 标准冲突，以及位于 GT 器件附近的噪声敏感引脚。 当发现错误或问题时，工具会显示一条提示信息 (Tooltip)，告诉你为什么不能够将某个 I/O 端口分配给特定的引脚。

通过激活 PinAhead 的“Autoplace”命令，还可以让其自动分配所有或任何选择的 I/O 端口到封装引脚。 Autoplace 命令将会遵守所有 I/O 标准和差分对规则，并正确布署全局时钟引脚。 该命令还会尝试尽量将 I/O 端口组合为接口 (interface)。

器件视图 (Device view) 以图形方式显示所有时钟区域和时钟相关的逻辑对象，从而使时钟相关的 I/O 分配更容易、更直观。 选择一个时钟区将会显示所有 I/O 组、时钟相关的资源以及与其相关的器件资源。 通过可用资源与其物理关系的探索，区域时钟规划过程变得更容易。

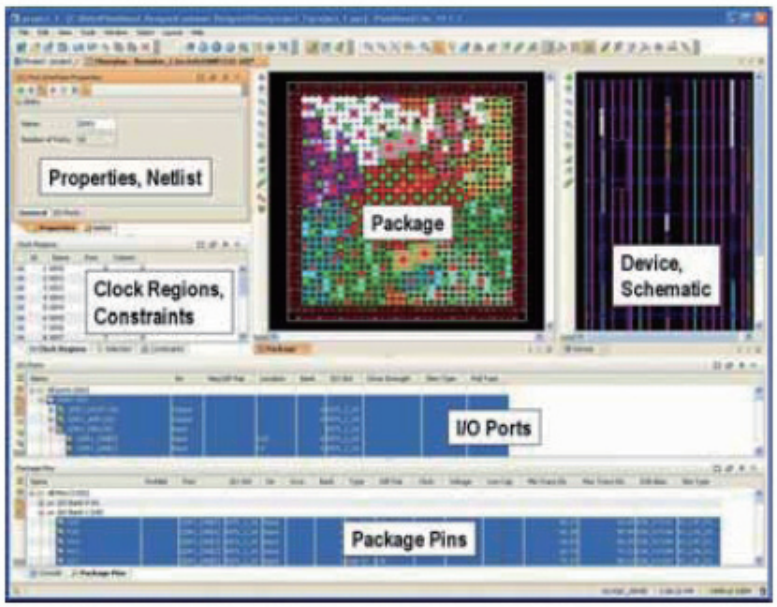


图6–2. PinAhead 环境

还可利用 PinAhead 布局设计中其它与 I/O 相关的逻辑，如 BUFG、BUFR、IODELAY、IDELAYCTRL 和 DCM。 利用 PlanAhead 中的“Find”命令，可以方便地定位这些对象和布放点。 要想有选择地察看的扩展逻辑和逻辑连接，请使用工具中的原理图视图 (“Schematic”view)。

通过在 PlanAhead 软件中的某个视图中选择特定的 I/O 相关逻辑并将其拖动到器件视图 (Device view) 中的选定位置，就可以锁定其布局。PlanAhead 软件将会自动判断，仅允许将有关逻辑放在合适的位置。 在拖动设计中的逻辑对象时，动态光标会显示出适当的布局位置。

步骤4： 为最终核签(Sign–Off)运行DRC和WASSO

一旦完成引脚分配，就可利用 PinAhead 丰富的 DRC 规则来进行核签前的 DRC 检查，保证在运行 PlanAhead 软件实施工具前设计是无错的。 工具中的众多 I/O 和时钟相关规则可保证 I/O 布局是合法的。 可利用 PlanAhead 的 DRC 对话框来选择相应的规则。

如果工具发现违反规则的情况，将会显示带有错误信息的 DRC 结果表。 选择相应的错误信息可以更进入地了解有关情况。

PlanAhead 还提供了加权平均同步转换输出 (WASSO) 分析功能，可帮助识别引脚分配引起的潜在信号完整性问题。 为工具提供 PCB 设计的寄生参数特性，PlanAhead 软件将会分析不同的 I/O 组以及其近邻，并报告每一 I/O 组的利用情况和状态。

步骤5： 导出I/O引脚分配数据

你可以将 I/O 端口列表和封装引脚信息从 PlanAhead 软件导出为 CSV 格式文件、HDL 头或 UCF 文件。CSV 文件包括有关器件封装引脚的所有信息，以及与设计相关的 I/O 引脚分配和配置。 列表中的封装引脚部分是数据表中定义 I/O 端口的很好起点。

你还可以利用该数据表自动生成设计小组开始 PCB 布局所需要的 PCB 原理图符号。 然而，有时这些符号对于原理图来说太长了，可能需要将它们缩短为几个符号。 利用 PinAhead 中的创建的接口组可以快速做到这一点。以原理图符号形式提供这些 I/O 引脚配置为 PCB 设计人员开始 PCB 布局提供了很好的基础。 因为如果在引脚分配的最初就考虑到 PCB 接口，那么最很可能与最终的引脚配置比较接近。

如果确实需要改变引脚来方便布线，那么改变也比较容易，因为需要改变的引脚可能已经在 I/O 组内了。这种方法不会对 FPGA 设计造成太大的影响。 通过在 PCB 和 FPGA 设计人员之间传递修改过的引脚布局数据表或 UCF 文件，可以保证两个部分之间的任何修改是同步的。

为防止信号噪声以及支持某些 FPGA 功能，你还可能希望将未用的引脚或特定配置的引脚连接到 VCC 或 GND。 赛灵思目前正在致力于在 PinAhead 的下一版本中提供这一功能。通过一个界面方便 FPGA 设计人员指导此类引脚，并在输出的 CSV 数据表中包含相应的引脚连接。 这样 PCB 设计人员就可以更容易识别相关引脚并正确连接之。

未来，随着 FPGA 集成更复杂的功能以及使用更先进的封装，发展可靠的 FPGA 和 PCB 引脚布局方法势在必行。 PinAhead Lite 在帮助实现基于协同设计理念的引脚布局策略方面已经能够提供很大帮助，但我们仍然已经在致力于改进这一工具以帮助设计人员更好地应对引脚布局方面的挑战。

6.2 用 Xilinx XtremeDSP 视频入门套件加速 FPGA 上的视频开发

作者：Tom Hill

美国赛灵思公司

DSP 部高级市场营销经理

tom.hill@xilinx.com

随着下一代视频压缩标准问世，行业从基本视频处理向更复杂的集成处理解决方案转移，这使得系统的要求超越了独立 DSP 力所能及的视频性能。FPGA(如 Xilinx® Spartan™–3A DSP) 以不到 $30 的价格提供 20 GMACs 以上的 DSP 性能，从而为成本敏感型军事、汽车、医疗、消费、工业和安全应用填补了这一空白。只有 FPGA 能够为整套端对端视频解决方案提供逻辑、嵌入式处理、OS 支持和驱动器。

妨碍开发人员将 FPGA 用于视频应用的因素并非他们缺乏对 FPGA 性能优势的了解，而是缺乏使用其设计流程的经验，对于那些习惯于用 C 语言编程的传统 DSP 程序开发人员来说尤为如此。

您可以利用 FPGA 的灵活性来配置针对特定应用而优化的硬件架构，以此发挥该器件的性能优势。这种灵活性为开发过程增加了自由度，同时也促进了其复杂性。

XtremeDSP™ 视频入门套件 (VSK) 提供了一种完善而易用的设计环境。这款开发套件包括应用示例并且完全支持标准的 赛灵思 工具流程，这有助于加速设计过程，并且还仍然能实现最终产品差异化。

XtremeDSP VSK – Spartan–3A 版简介

XtremeDSP 视频入门套件 – Spartan–3A 版是由 Spartan–3A DSP 3400A 开发平台、FMC 视频子卡和 VGA 摄像头组成的视频开发平台。

Spartan–3A DSP 3400A 开发平台 (可单独购买) 是围绕 Spartan–3A DSP XC3SD3400A 器件制造的。此器件具有 126 个嵌入式 DSP 模块，可用来实现具有协处理功能的高性能视频处理系统。

FMC 视频子卡增加了以下接口，从而扩展了 Spartan–3A DSP 3400A 开发平台的视频 I/O 功能：

- DVI–I 输入接口，具有数字和模拟两种模式
- 复合输入输出接口

- S 视频输入输出接口
- 两个摄像头输入接口

视频开发工具

使用 赛灵思嵌入式开发套件 (EDK) 和 System Generator for DSP 工具，您无需 RTL 知识或经验也能为 VSK 创建视频应用。EDK 是一种用来设计嵌入式可编程系统的功能齐全的解决方案，其中包括 Platform Studio 工具套件、嵌入式 IP 核和 MicroBlaze™ 嵌入式处理器。

System Generator for DSP 提供了一个 Simulink 模块集，其中包括 100 多个经赛灵思 优化的 DSP 构建模块，从而使 FPGA 设计能够使用 The MathWorks 的 Simulink/MAT–LAB 建模环境。

使用基础平台开发视频应用

称为基础平台的嵌入式系统提供了一个框架，您可以从中使用 VSK 来开发视频应用。基础平台是使用 Xilinx Platform Studio 的 Base System Builder (BSB) 创建的嵌入式系统，其中包括一个 MicroBlaze 嵌入式处理器。

此框架可以为新设计提供起点，也可以作为便利的移植途径用来移植在基于处理器的系统上开发的现有应用。在 MicroBlaze 处理器上，您可以轻而易举地为外部处理器重新编译任意 C 代码；高性能视频链一旦接入，便可以从软件移植到 FPGA 架构。

为了协助这种移植，VSK 包括了一个定制外设 IP 库，您可以使用 Platform Studio 方便地将其中的定制外设添加到基础系统。您可以连接到视频接口、管理数据帧以及执行存储器访问和基本视频处理。这些定制外设包括：

- DVI 输入
- DVI 输出
- 摄像头
- 视频帧缓冲器控制器 (VFBC)
- 视频处理流水线

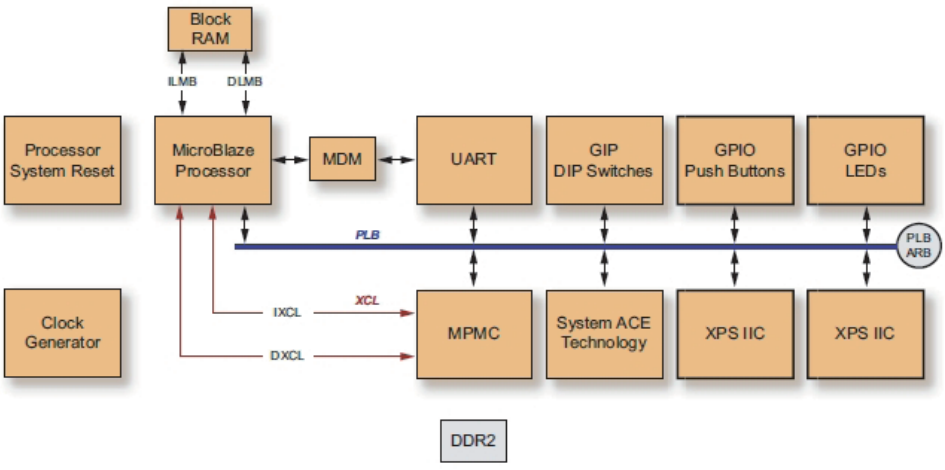


图6–3 基础平台框图

Xilinx VFBC 非常适合需要用二维数据的硬件控制来实现实时操作的视频应用。VFBC 的主要功能包括运动估计、视频缩放、屏幕显示和视频采集，这些功能可用于视频监控、视频会议和视频广播。

用 VSK 参考设计迅速启动开发过程

VSK 提供三个参考设计，用来快速启动在 赛灵思 FPGA 上运行的视频应用的开发过程。每个参考设计都是在基础平台上构建的，并且使用了 VSK 的 IP 库中的定制外设。表 1 列出了各参考设计及其所显示的视频处理和连接功能。这些参考设计旨在提供一个起点，可以在此基础上进一步开发。图 2 所示为如何将 DVI 穿越端口参考设计接入基础系统。

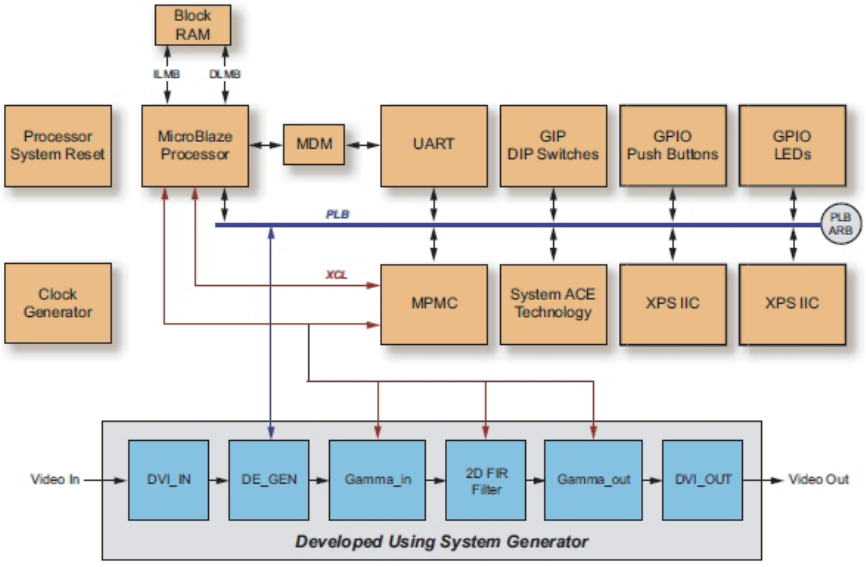


图6–4– 带视频流水线的基础系统

用基于模型的设计创建视频应用

要加速 FPGA 上的视频应用，就需要将性能关键型操作从处理器上运行的软件移植到硬件。VSK 支持多种硬件设计流程，其中包括使用 VHDL/Verilog 来发挥坚实的硬件设计背景的流程，也包括借助较多抽象建模环境 (包括 C、MATLAB 和 Simulink) 而需要很少或完全不需要硬件设计经验的流程。

The MathWorks 的 Simulink 是一种基于模型的设计环境，可用来开发视频系统的算法模型。The MathWorks 为 Simulink 提供了一个可选的视频与成像模块集，其中包括一组丰富的视频构建模块，可用来方便地处理流式视频并且在模型中的每一步显示结果。

您可以首先使用浮点数据类型以及高层视频和成像模块为视频处理算法本身建立抽象模式，然后以您认为能够权衡复杂性、系统成本和性能的方式来优化算法。

System Generator for DSP 提供了一组丰富的针对 赛灵思 器件优化的 DSP 构建模块，因此可以将 Simulink 用于 赛灵思 FPGA 设计。通过紧密集成，可以将在 System Generator 中采集的 DSP 设计转换成适合 Platform Studio 的定制外设，并且使用处理器局部总线或快速单工链路总线将其连接到基础系统。

图 6–5 所示为用 System Generator 创建的摄像头视频处理流水线的示例，此示例包括在 VSK 所附带的摄像头帧缓冲器参考设计中。

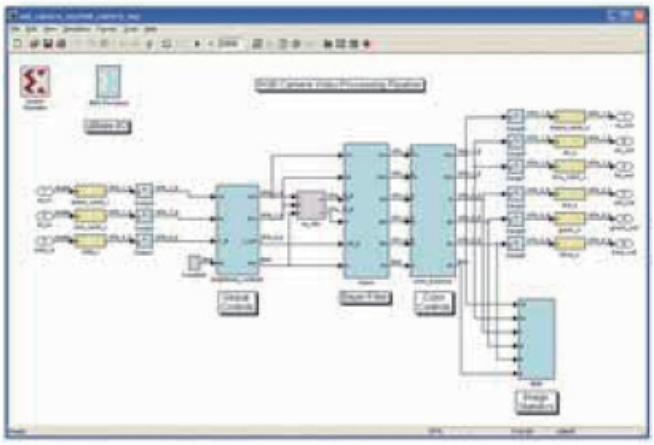


图6–5 摄像头视频处理流水线的 System Generator 图

System Generator 使用 Spartan–3A DSP 3400A 开发平台来支持硬件在环协同仿真。您可以使用此平台将 Simulink 仿真的性能加速 1,000 倍。由于此加速，可以使用通过 The MathWorks 的数据采集工具箱读入 Simulink 的实时视频流来进行视频算法开发和调试。

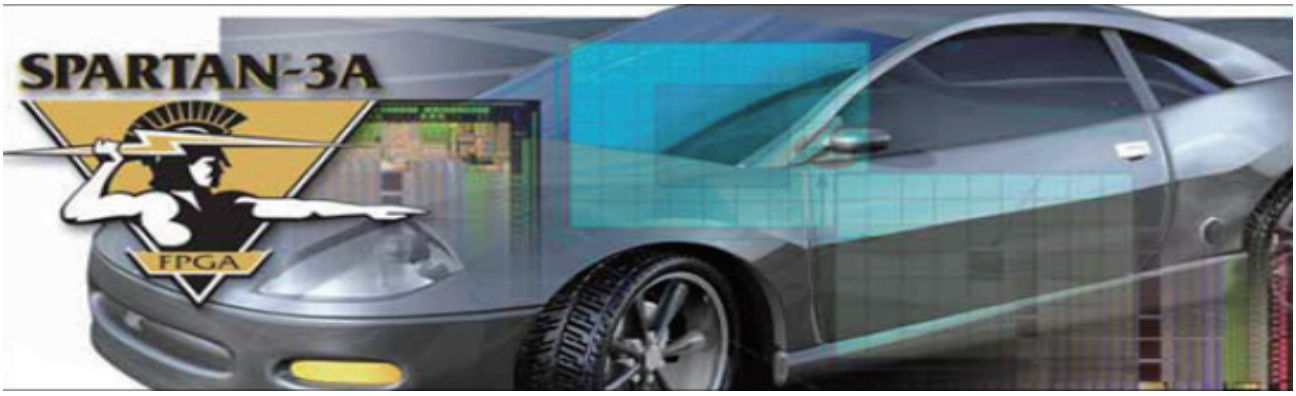
参考设计	功能描述
DVI 穿越端口	• 从输入端口采集视频流 • 对视频流执行实时图像处理 • 显示经过处理的视频
DVI 帧缓冲器	• 从 DVI 源采集视频流 • 在外部存储器中缓冲视频流 • 显示经过缓冲的视频 • 报告存储器带宽占用数据
摄像头帧缓冲器	• 从摄像头采集视频流 • 对视频流执行处理 • 在外部存储器中缓冲视频流 • 以另一种速率显示经过处理的视频 • 使用微处理器配置视频流水线的各种特征

表6–1 VSK 参考设计概览

结论

XtremeDSP 视频入门套件 – Spartan–3A DSP 版以不到 $1,600 的价格提供完善的视频开发解决方案，从而保持很低的开发成本。使用其中包含的 DSP 和嵌入式设计工具，无需 RTL 设计经验就能迅速完成视频系统的 FPGA 开发。

DSP 优化型 FPGA 平台 (如 Virtex–5 SXT 器件、Virtex–4 SX 器件和 Spartan–3A DSP) 特别适合用来满足安全、广播、工业、消费、医学和汽车应用领域中高性能视频和图像处理应用的成本和性能要求，同时还可以防止产品过早淘汰。有关更多信息，请访问 www.xilinx.com/cn/s3adsp_vsk。



6.3 用 Spartan–3A DSP 器件实现汽车应用中的块匹配

作者：Daniele Bagni Paul Zoratti

赛灵思 公司

DSP 专员 汽车高级系统架构师

daniele.bagni@xilinx.com paul.zoratti@xilinx.com

汽车工程师采用多种智能技术帮助人们安全驾驶汽车。汽车系统中的主要技术包括雷达、超声和摄像 / 视觉感测。这些技术统称驾驶员辅助 (DA) 系统，用于在恶劣条件和危险路况下协助安全驾驶。

第一代摄像 DA 系统目前可见于各种生产用车型。这类系统大多为驾驶员提供车辆周边环境的视频图像。最常见的是泊车 / 倒车辅助系统，这种系统用后视摄像头拍摄本车后面的景物，并且在无线电 / 导航系统的屏幕上或者在仪表板中的小型显示器上显示图像。

第二代摄像系统正处于开发测试阶段，目前使用有限。第二代系统并非仅为驾驶员提供图像，而是运用图像处理与解析从视频流中提取信息，并且对车辆环境进行表征和评估。必要时驾驶员会收到相应警示。

随着工程师们获取车辆环境表征方面的实际经验，未来的 DA 技术会更复杂，将为消费者提供更强大的实用工具并且增强其他汽车子系统的性能。图 1 概括了目前和未来的多种 DA 功能。

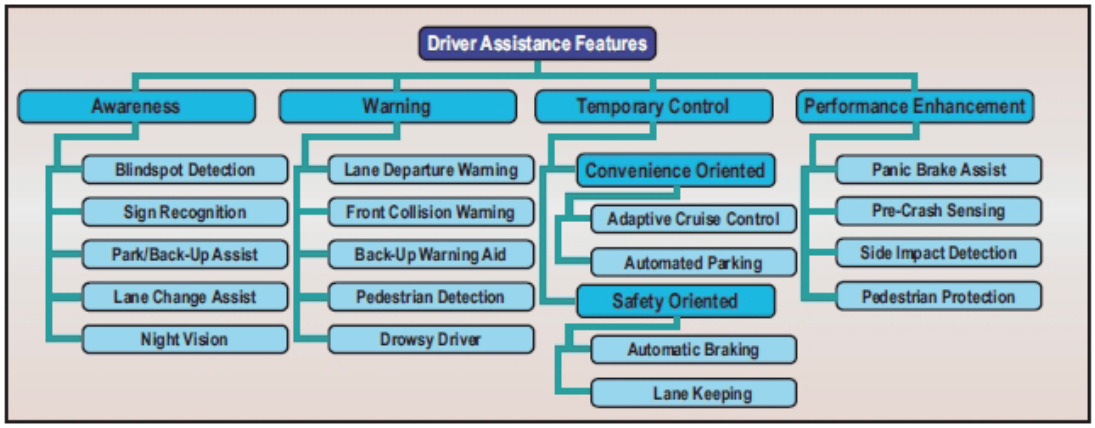


图6–6– 驾驶员辅助功能

一、高级处理要求

DA 系统的处理要求可能超过目前汽车级串行 DSP 处理器的能力。另外，为了提高消费价值，在一套视觉传感器上捆绑多种 DA 功能的需求日益高涨。

例如，前瞻视觉模块可能需要同时支持车道偏离警示、智能大灯控制和标志识别功能，而所有这些都需要不同的处理算法。因此，对于用 FPGA 通过原始图像数据处理、配置灵活性和器件可缩放性来提供系统价值，DA 市场提供了实实在在的机会。

视觉 DA 系统的图像处理与解析功能可以包括空间 / 时间滤波、镜头失真校正、图像清晰化、对比度增强、边沿检测、图像匹配、物体识别和物体跟踪，在某些情况下还包括图形叠加。尤其值得关注的是一种支持运动估计或立体视差计算的图像匹配功能。

为了说明 FPGA 处理的性能价值，我们来考虑这样一个视觉系统：以 30 Hz 帧速率 (fps) 生成视频的宽幅 VGA 分辨率成像器件 (752 x 480 像素)，而且需要估计帧间物体运动 (或流动)。有一种算法 (也适用于立体视差计算) 是将图像划分为若干个块 (如大小为 4 x 4 像素),然后判断第一帧中各图像块对第二帧中指定搜索区 (如 20 x 20 像素) 内某位置的匹配条件。

一种常用的匹配条件是用算子 SAD(绝对差和) 求出第一帧图像中的 4 x 4 块与第二帧图像上搜索区内的像素之间的像素灰度最小绝对误差 (MAE)。

我们的 4 x 4 块匹配示例需要 250 MMAE/s(每秒百万次 MAE 计算) 以上的性能，因为 (752 像素) x (480 行) x (20 x 20 像素搜索区) x (30 fps)/(4 x 4 像素块大小) = 270,720,000 MAE/s。MAE 表示 4 x 4 像素块的最终匹配误差，而 SAD 是指根据四个独立元素对进行计算得到的绝对差和。所以，每 MAE 需要四次 SAD 运算。

二、处理选项

由汽车设计工程师决定的处理选项包括超长指令字 (VLIW) DSP–CPU 和 FPGA。FPGA 的处理能力远远高于任何现有的 VLIW DSP–CPU。这是由于 FPGA 的架构：大量并行功能单元 (包括可编程 MAC) 使 FPGA 的性能比任何 DSP 都高出 10–30 倍 (具体性能取决于所实现的应用)，即使 FPGA 的时钟频率比 DSP–CPU 的时钟频率低得多。我们使用块匹配运算示例，是要证明 Xilinx® FPGA 的性能比任何 VLIW DSP–CPU 处理器都高。

三、VLIW DSP–CPU 处理器中的 SAD 和 MAE 计算

在一个 32 位架构的单指令多数据 (SIMD) DSP–CPU 中可实现四个 8 位像素视频数据单元的 SAD 运算，因此，仅在一个周期内即可有效执行相当于 11 条基本指令的运算，如图 6–7 所示。

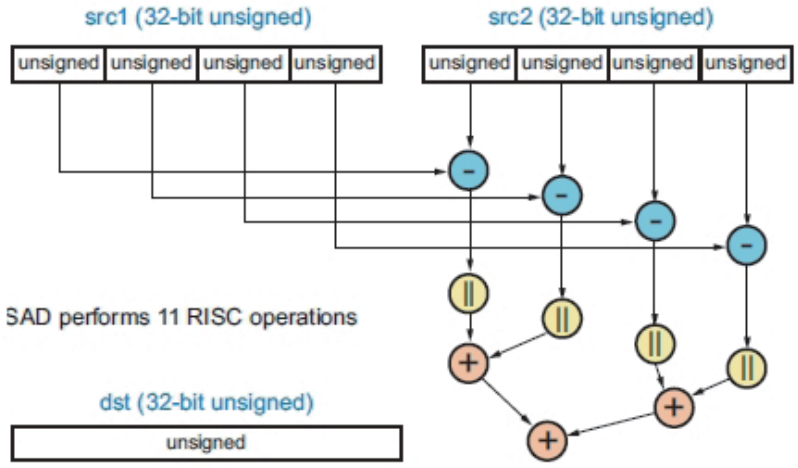


图6–7 – SIMD 示例：四路 8 位样本的 SAD 运算

例如，Nexperia PNX1500 媒体处理器配有 32 位 TriMedia VLIW–CPU，对于具有二周期延迟的 8 位像素，可以在一个时钟周期内执行两条四路 SAD 指令。算上超长指令字，就是每时钟周期最多五条基本 RISC/SIMD 指令，其中只有两条可以是 SAD 指令 (在 TriMedia 数据手册中称为“8meii”)。

所以，对 4 x 4 大小的块进行 MAE 计算需要五个时钟周期，如表 6–2 所示：两个周期用于两条四路 SAD 指令的流水线处理 (周期 1 用于 sad1/sad2，周期 2 用于 sad3/sad4)；三个周期用于部分结果的累加 (周期 3、4 和 5)。因此，如果只处理一个块，则一个 300 MHz 的 Nexperia PNX1500 处理器的处理能力最高可达 60 MMAE/s。

	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5
Cycle 1	sad1=8meii(A1,B1)	nop	sad2=8meii(A2,B2)	nop	nop
Cycle 2	sad3=8meii(A3,B3)	nop	sad4=8meii(A4,B4)	nop	nop
Cycle 3	sad12=sad1+sad2	nop	nop	nop	nop
Cycle 4	sad34=sad3+sad4	nop	nop	nop	nop
Cycle 5	tot=sad12+sad34	nop	nop	nop	nop

表6–2 – Nexperia/TriMedia VLIW DSP–CPU 上的 MAE 计算的伪汇编代码，使用四路 8 位子字并行处理

如果每次处理一个以上 4 x 4 块，最高性能可略有提高。例如，可以在七个周期内计算两个并行 4 x 4 块的 MAE，这时性能可达 85.71 MMAE/s；而处理三个块需要九个周期，即性能为 100 MMAE/s。

可并行处理的最大块数分别受限于任意长指令字中允许的 SIMD SAD 运算次数、VLIW–CPU 的通用寄存器数和优化编译器的调度算法。如果继续增加块数，整体性能会趋于饱和，因此我们考虑并行处理的 MAE 不超过三个。

德州仪器 (TI) 的 TMSD320DM6437 数字媒体处理器每周期有一条由八次基本 RISC 运算组成的长指令，分别通过两条数据通路，各通路每周期有四个时隙。其 VLIW–CPU 每周期最多可执行两条 SAD 指令 (在 TI DM6437 数据手册中称为“subabs4”)，各指令有一个周期的延迟。但是，要累加部分结果，就必须使用常数 0x01010101 执行具有三周期延迟的 SIMD MAC 运算 (称为“dotpsu4”)。

	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7	Slot 8
	L1	S1	M1	D1	L2	S2	M2	D2
Cycle 1	d1=subabs4(A1,B1)	nop	nop	nop	d2=subabs4(A2,B2)	nop	nop	nop
Cycle 2	d3=subabs4(A3,B3)	nop	sad1=dotpsu4(d1, 0x01010101)	nop	d4=subabs4(A4,B4)	nop	sad2=dotpsu4(d2, 0x01010101)	nop
Cycle 3	nop	nop	sad3=dotpsu4(d3, 0x01010101)	nop	nop	nop	sad4=dotpsu4(d4, 0x01010101)	nop
Cycle 4	nop	nop	nop	nop	nop	nop	nop	nop
Cycle 5	nop	nop	nop	nop	nop	nop	nop	nop
Cycle 6	sad13=sad1+sad3	nop	nop	nop	sad24=sad2+sad4	nop	nop	nop
Cycle 7	tot = sad13+sad24	nop	nop	nop	nop	nop	nop	nop

表6–3 – TI VLIW DSP–CPU 上的 MAE 计算的伪汇编代码，使用四路 8 位子字并行处理

所以，600 MHz 的 TI DM6437 DSP–CPU 可以用七个周期计算一个 MAE(如表 6–3 所示)，因此对于 4 x 4 像素块的最高性能为 85.71 MMAE/s。如果并行处理两个块，就需要九个周期，性能为 133.33 MMAE/s ；而三个块需要 11 个周期，性能为 163.64 MMAE/s，这仍然低于我们的 250 MSAD/s 要求。

四、VLIW DSP–CPU 性能不足

至此，我们一直假定每像素 8 位，这很适合 32 位架构的 DSP–CPU 处理器。然而，新型 CMOS 图像传感器的分辨率范围较高,即每像素 12 到 14 位。对于这些数据类型,32 位架构的传统四路 8 位子字 SIMD 不够有效，必须换用双路 16 位半字 SIMD，其中的子字并行度仅为二。因此，由于计算一个 MAE 需要较多时钟周期，最高性能大幅度下降。表 6–4 所示为在 TI VLIW DSP–CPU 上使用 16 位子字指令计算 SAD 时可能的伪汇编代码，假定延迟正确且函数发射时隙允许执行这种指令。

	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7	Slot 8
	L1	S1	M1	D1	L2	S2	M2	D2
Cycle 1	d1=sub2(A1,B1)	d3=sub2(A3,B3)	nop	nop	d2=sub2(A2,B2)	d4=sub2(A4,B4)	nop	nop
Cycle 2	ad1=abs2(d1)	nop	nop	nop	ad2=abs2(d2)	nop	nop	nop
Cycle 3	ad3=abs2(d3)	nop	nop	nop	ad4=abs2(d4)	nop	nop	nop
Cycle 4	z13=add2(ad1, ad3)	nop	nop	nop	z24=add2(ad2, ad4)	nop	nop	nop
Cycle 5	nop	nop	s13=dotp2(z13, 0x00010001)	nop	nop	nop	s24=dotp2(z24, 0x00010001)	nop
Cycle 6	nop	nop	nop	nop	nop	nop	nop	nop
Cycle 7	nop	nop	nop	nop	nop	nop	nop	nop
Cycle 8	tot = s13 + s24	nop	nop	nop	nop	nop	nop	nop

表6–4 TI VLIW DSP–CPU 上的 MAE 计算的伪汇编代码，使用双路 16 位子字并行处理

因此，一个 4 x 4 的块需要八个周期，而并行处理两个和三个块分别需要 10 个和 12 个周期。这时，相应的最高性能分别为 75 MMAE/s、120 MMAE/s 和 150 MMAE/s。这些数字都比使用 8 位子字指令得到的数字小。

五、Spartan–3A DSP FPGA 的 SAD 和 MAE 性能

为了填补 Spartan™–3 和 Virtex™–4 器件之间的处理性能空白，赛灵思推出了 Spartan 3A–DSP 1800A 和 3400A FPGA。这些器件采纳了 Virtex–4 器件中的 DSP48 Slice 的修改版。另外，3A–DSP 器件包括大量片上存储器 (Block RAM)。这两方面增强加上针对大量应用制订的价位使 3A–DSP 器件非常适合汽车视觉 DA 系统。

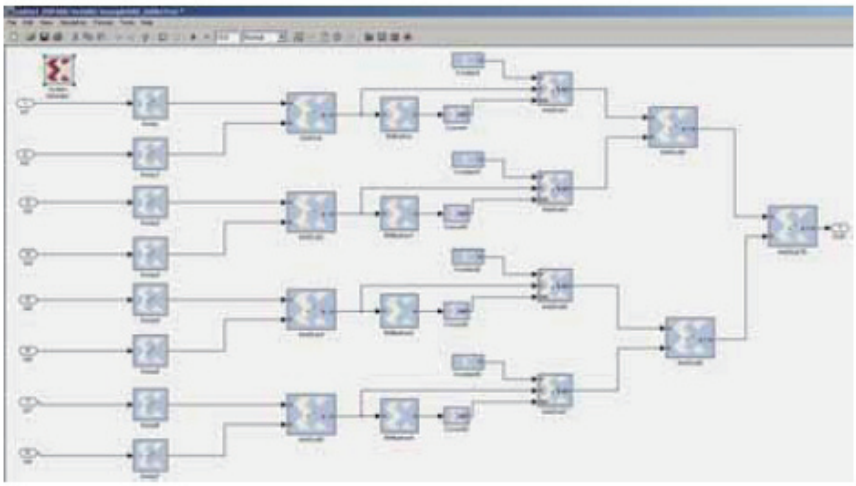


图6–8 Spartan–3A DSP 1800 器件上四个 12 位样本的四路 SAD 计算方案

图 6–8 所示为 Spartan–3A DSP 1800 (XC3SD1800A–4FG676) 器件上的四路 12 位像素的 SAD 计算方案。此实现是使用 System Generator for DSP 设计流程 (Xilinx 提供的 Simulink 工具中的数位和周期都精确的可综合库) 完成的。所需资源数量是 121 个 Slice(236 个 LUT 和 140 个触发器)。将此结构复制四次并且加上部分结果，即得到整个 4 x 4 块的计算方案，该方案需要 508 个 Slice(990 个触发器和 606 个 LUT)，具有一个周期吞吐量 (这意味着可从任意时钟周期开始计算新的 MAE) 和七个周期延迟。

如果使用 150 MHz 时钟频率 (该器件最高时钟频率为 250 MHz)，只需要两个并行结构 (约占器件面积的 6%) 即可达到 300 MMAE/s 的性能，从而满足示例应用的 250 MMAE/s 性能要求。这样可以节省大量资源用来实现其他图像处理功能、数据路由管道、存储器接口控制器以及一个用于串行处理和外部通信的 32 位 MicroBlaze™ 嵌入式处理器。

作为参考，仍然用 150 MHz 频率，Spartan 3A–DSP 1800A 器件仅使用整个 FPGA 器件的 70% 即可并行处理多达 23 个块 (70% x 16,640 Slice/508 Slice/ 块 = 23 块)。与此对应的最高性能是 3,529 MMAE/s，这至少要比 600 MHz 的 TI DSP–CPU 的最高性能高 25 倍。

六、结论

我们以汽车视觉应用为例说明了如何利用中型低成本 Xilinx FPGA 的可编程并行处理能力提供超过 VLIW DSP–CPU 的处理性能。表 4 列出了我们的分析结果。

器件	配置	时钟频率	性能
应用示例要求	752 x 480 图像 4 x 4 像素块 20 x 20 像素搜索区	不适用	
Philips Nexperia PNX 1500 VLIW DSP–CPU	8 位像素深度 并行处理：三个 4 x 4 块		
TI TMSD320DM6437 VLIW DSP–CPU	8 位像素深度 并行处理：三个 4 x 4 块		
TI TMSD320DM6437 VLIW DSP–CPU	12 位像素深度 并行处理：三个 4 x 4 块		
Xilinx Spartan–3A DSP 1800A FPGA	12 位像素深度 并行处理：两个 4 x 4 块 (约为器件资源的 6%)		
Xilinx Spartan–3A DSP 1800A FPGA	12 位像素深度 并行处理：23 个 4 x 4 块 (约为器件资源的 70%)		

表6–5 结果汇总

请注意，对于 12 位像素数据的 4 x 4 块的 MAE 计算，Spartan–3A DSP 的性能仅以四分之一时钟速度即可达到 TI TMS320DM6437 的两倍。另外，FPGA 的资源占用率仅为 6%，因此可以在同一器件上实现其他图像处理功能 (必要时可采纳并行处理)。

另一方面，VLIW DSP–CPU 在 SAD 计算期间被完全占用，消耗串行处理器长指令的可用时隙，因此很少有机会同时执行其他功能。

我们对于 FPGA 的估算时钟频率相当保守 (以 150 MHz 对 250 MHz)，对于运动估算的搜索区也是如此 (搜索区越大，需要计算的 MAE 的数量就越多)。例如，30 x 30 的搜索区需要 609 MMAE/s 的性能 (远远超过 VLIW DSP–CPU 的能力)，然而却仅占用 1800A 器件上 Slice 的 12%。

最后，我们在实现 MAE 时根本未使用 DSP48 MAC 单元：据我们估计，如果用四个 DSP48 单元取代由 100 个 Slice 组成的加法器树，则一个 12 位输入数据 MAE 的 4 x 4 块会占用 400 个 Slice(782 个触发器和 400 个 LUT) 和四个 DSP48。

因此，Spartan–3A DSP 1800A 器件非常适合需要极高处理性能、灵活性和可缩放性的视觉应用，如未来型汽车驾驶员辅助系统中的视觉应用。

6.4 利用 CoolRunner-II CPLD 设计 GPS 系统

作者：Arthur Yang/小批量部技术营销经理

赛灵思 公司

arthur.yang@xilinx.com

正如查看地图或读报纸这些方式正在迅速减少一样，“问路”这一概念也开始日渐陌生。其行将消亡可归因于全球定位系统 (GPS) 的日益普及。GPS 在汽车、移动电话、PDA 甚至手表等越来越多产品中出现。每个 GPS 供应商都推出数十种 GPS 产品，令消费者无所适选。

因此，产品的成功在于差异化和专业化。Xilinx® CoolRunner™-II CPLD 是用来增加功能或与另一器件连接的理想芯片，能使您的 GPS 产品在不超出功耗预算的情况下从众多同类产品中脱颖而出。

一、功耗优势

便携式和仪表板式 GPS 设备都必须遵守严格的功耗预算。CoolRunner-II CPLD 具有无需休眠模式状态即可达到极低功耗的优势。最小的 CoolRunner-II 器件的静态供电电流只有 13 μA。在某些便携式应用中，有一种休眠模式足矣，并且可以接受数百毫秒的唤醒时间。

然而在某些情况下，会丢失当前的设计状态。在使用休眠模式时，整个设备都会关闭。您必须依靠另一器件来轮询中断并启动唤醒序列。CoolRunner-II 器件通过其 DataGATE 功能提供一种可摆脱设计困惑的独特休眠模式；DataGATE 是自主运行的用户可配置电路，允许用户根据需要停用器件的某些部分。

通过启用 DataGATE 功能，在数纳秒内即可关闭选定的任意输入，从而将 CPLD 的功耗调节到更接近静态功耗。此操作可以阶段性进行 (例如在轮询到某个中断时)，也可根据特定的操作状态进行。这样便可将 CoolRunner-II 器件的某些部分保持活跃状态，而将其余部分置于待机状态。

我们举个例子来进一步说明。CPLD 位于微处理器、移动 SDRAM 和 SD Flash 卡之间的地址和数据总线上。数据在各器件之间移动。CPLD 监测数据活动，并且根据特定功能阻断交通。如果当前任务是从微处理器向 SD 卡移动数据，则 CPLD 用 DataGATE 功能阻断往来于 SDRAM 的所有数据。

您在 CPLD 内部编写代码即可解译正在执行的功能 (通过监视地址线或解析帧数据)，然后开启 / 关闭所需的数据通路。这是一个简单泛例，不大可能用其它可编程逻辑器件 (PLD) 通过使用输出三态来实现。

DataGATE 的优势是可以对 CPLD 的输入进行有效的三态控制。之所以能降低功耗是因为 CPLD 的 I/O 以及核心电路均处于静态，而其它解决方案则要求整个 PLD 都处于活跃模式。

二、安全优势

GPS 系统的价位使其成为颇具吸引力的产品克隆目标。为了帮助防范克隆，您可以利用 CoolRunner–II CPLD 的读写保护安全机制实现一种可防止通过超量生产进行克隆的安全系统。超量生产是指签约制造商针对给定的生产批次超量订购元件，然后超量生产与正版完全相同的产品。

其原理是系统要求必须与 CPLD 进行交互操作才可执行所需功能。有多种方法可以实现这种安全机制。最直截了当的解决方案是让 CPLD 充当数据交通警察，指挥数据在电路板上的各器件之间流动。

一种较复杂的解决方案涉及到用 CPLD 来实现密码阻断功能。这种方法是让微处理器提交一段随机数据流，该数据流经 CoolRunner–II 器件加密后返回。然后，将此数据解密并对照原始数据进行验证。

CoolRunner–II CPLD 的安全机制是通过众多编程位元实现的，所以，如果想确定哪些位元与安全机制有关，就必须从非易失性阵列内的数万位元中找出几个位元，然后才能将其破解。CoolRunner–II CPLD 的读写安全机制可阻止对原码型进行读回和附加编程。所以，器件既不允许提取已编程的 JEDEC 文件的内容，也不允许在原码型上叠加修改版代码。要对器件重新编程，必须先擦除整个器件中的代码，而这样做会丧失设计信息。

在所有这些流程中，一个关键因素是必须由可信来源 (赛灵思或授权的赛灵思分销商) 向签约制造商提供预编程的 CPLD。这样做可以防止超量生产，因为仅超量订购空白 CPLD 而无法进入编程文件是无济于事的。

三、实现 GPS 设备的产品差异性

最初的消费型 GPS 设备简单明了，只是以经纬度形式给出位置信息。今天的 GPS 设备不仅提供实时地图和方向，而且还提供 MP3 播放或通过蓝牙来集成移动电话。还有面向特定市场的 GPS 系统，其功能各异，诸如用于车内导航的即时交通信息、供长跑或自行车运动员用来测速的超小型设备以及供钓鱼者使用的带声纳的设备，甚至还有用于跟踪家养宠物的带 GPS 功能的项。

每种产品都需要有连接不同对象的接口，这正是 CoolRunner–II CPLD 的强势所在。下面专题介绍 CoolRunner–II CPLD 的一些用法。本文末尾处还列出了相关的 赛灵思应用指南。图 1 所示为可借助 Xilinx CPLD 增强 GPS 应用的功能。

SD 卡接口

SD 存储器 (有小型 SD 和微型 SD 等多种物理形式) 接口已经从几年前盛极一时的各种存储器接口中分划出来。MMC 和 Compact Flash 等其他接口仍可见使用，但所占市场份额不大。Flash 卡主要用于具有图片或 MP3 播放功能的产品，而这两种功能在 GPS 手持设备中已日趋常见。

移动 SDRAM 接口

通过将 CPLD 用作存储器接口，可以简化微处理器的代码。如果您的高端机型需要多种存储器模块但低端产品不需要，则可修改 CPLD 代码而不必更换微处理器。

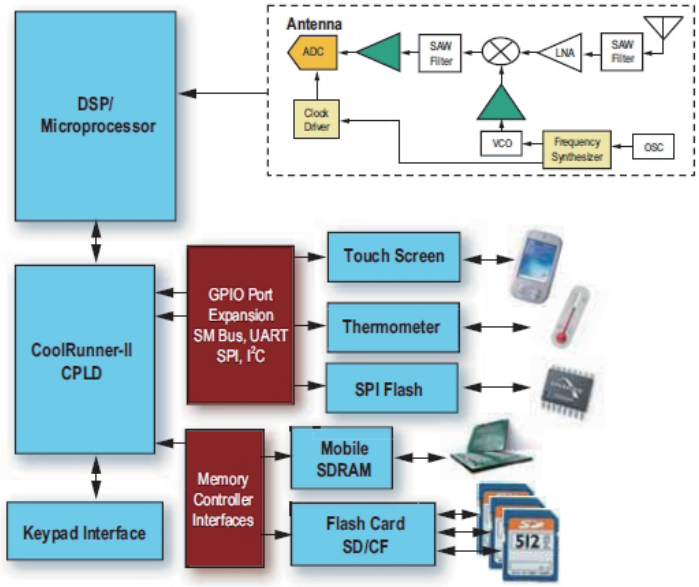


图6-9 GPS 框图

电平转换

在进入较新的消费地区时，GPS 需要连接尚未针对低工作电压优化的元件。CoolRunner-II CPLD 至少有两个 I/O 组 (在最大器件中增加到四个 I/O 组)，可轻松满足多电压接口的要求。支持的电压标准有 1.5V、1.8V、2.5V、3.3V、SSTL 2-1、SSTL 3-1 和 HSTL-1。5V 的接口需要用某种外部电路予以支持。

键盘扫描仪

许多低端 GPS 机型由于成本制约或尺寸限制而不能使用触摸屏接口。甚至在昂贵机型中也有几个用于电源或音量控制等特定功能的按钮。某些形式的键盘或按钮接口在大多数机型中都使用。这是 CoolRunner-II CPLD 的理想用法，因为 CPLD 在用户不操作时保持静态，并且可立即响应用户的按键操作而不必从休眠模式中唤醒。另外，还可以将其设计成先验证用户数据，然后再唤醒系统其余部分。例如，许多移动电话需要连续按两个键才能唤醒，以避免意外按键操作。

微处理器接口

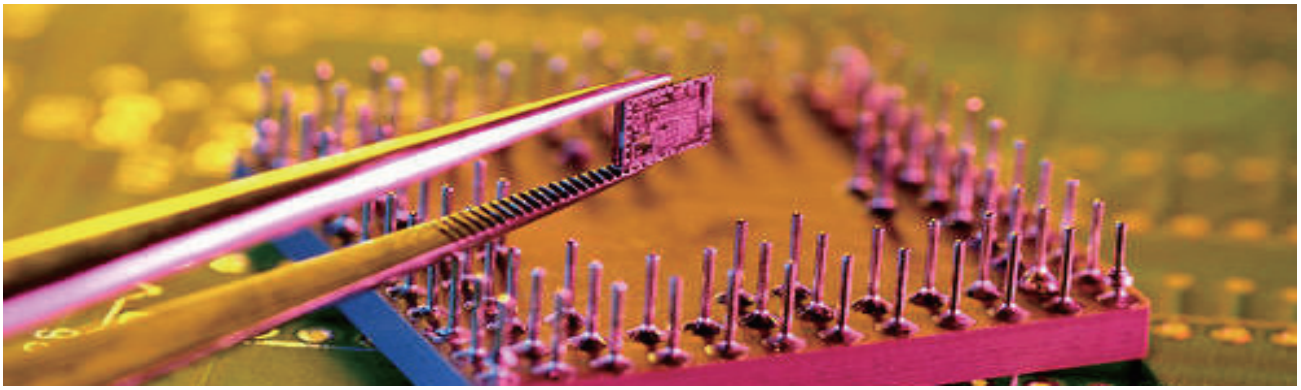
CoolRunner-II CPLD 的一种常见用途是端口扩展。许多微处理器没有足够的 I/O 供与之通信的众多器件使用。CoolRunner-II CPLD 允许产品平台设计在不变更核处理器的情况下增加和变更模块。

串行外设接口 (SPI)

SPI 是一种常见接口，用于包括 Flash 存储芯片、LCD、触摸屏和温度传感器在内的多种外设。这种接口之所以普及，是因为它是一种简单的四线接口，并且能通过 I2C 或 SMBus 提升吞吐量。

四、结论

Xilinx CoolRunner-II CPLD 在降低功耗、提高安全性以及为今天的 GPS 系统提供连接解决方案方面显示出优势。借助我们的小型化封装组合，这些器件可装入移动电话、宠物项圈和手表等极小型便携式设备。在汽车钥匙里嵌入 GPS 或许已指日可待。



6.5 利用赛灵思 EDK 工具和 IP 设计多处理器 SOC

作者：Vasanth Asokan

赛灵思公司资深软件工程师

Vasanth.asokan@xilinx.com

随着嵌入式处理需求的快速增长，系统架构正朝着多处理器设计的方向发展，以解决单处理器系统复杂度太高和计算能力不足的问题。凭借高逻辑密度及高性能硬模块，新一代 FPGA 已经使功能强大的芯片多处理 (CMP) 解决方案成为现实。现在的挑战在于如何在该解决方案的范畴内快速完成设计的开发与创建。

赛灵思嵌入式开发套件 (EDK) 工具和 IP 提供了很大的灵活性，用户可利用它们以 FPGA 逻辑为基础设计出独具特色的定制多处理解决方案，从而同时满足价格和性能目标。本文将对应用于以 PowerPC™ 和 MicroBlazeTM 嵌入式处理器为基础的赛灵思解决方案的多处理概念进行介绍。

一、应用领域

性能和功能划分是设计多处理器系统的主要推动力。总的来说，多处理在以下这些常见场合中会发挥作用：

多个独立功能。设计可能需要应对多个独立的处理任务。解决这个问题的好方法是为每个处理任务创建独立的专用处理模块，并为每个处理模块分配专用处理器和外设集。

控制或数据层面卸载。常见情况是既有实时任务 (计算或数据密集型)，也有非实时任务，从而可能导致单处理器解决方案对其无响应。对于这种情况，您可以分配一个从处理器以便及时完成实时任务。这使得主处理器可以完成其他常规任务，并且充当到主机系统的接口。主处理器同时也监控从处理器。从处理器可能包含专用功能或接口，从而能够满足计算性能要求。相应的例子包括网络负载分担、媒体处理以及安全算法等。

接口处理。对于作为多接口之间的桥梁或开关的系统，您可以分配一个从处理器用于处理每个接口上的数据，而用一个或者多个主处理器处理更高级的桥接或者开关任务。

数据流处理。对于数据流计算问题，您可以安排多个处理器以流水线的方式处理数据流。多处理器流水线的每一级都要在将数据传到下一个处理器之前完成一部分计算任务。这是提高系统吞吐量的一种有效方式。

可靠性和冗余度。您可以多次复制处理系统以提高可靠性和冗余度。

对称处理。传统的对称处理 (SMP) 是一种十分有用的解决方案，您可以利用它来提升那些不存在明确的划分边界的应用的性能。一个具有 SMP 功能的 OS 层可以管理并行任务，并且在多处理器之间自动调度这些任务。然而，SMP 使用模型不适用于赛灵思处理器，因为它们缺乏实现 SMP 所需的高速缓存一致性。

除了 SMP 场合，其他的所有应用场合均适用于赛灵思带有 EDK 工具的 FPGA。赛灵思处理解决方案的独特之处在于其可以针对应用要求来灵活定制每个处理子系统。例如，并不是所有的处理器都需要一个高速缓存或者浮点单元。通过为特定处理器分配特定的功能，您可以创建一个能够实现所有设计目标的专用解决方案。

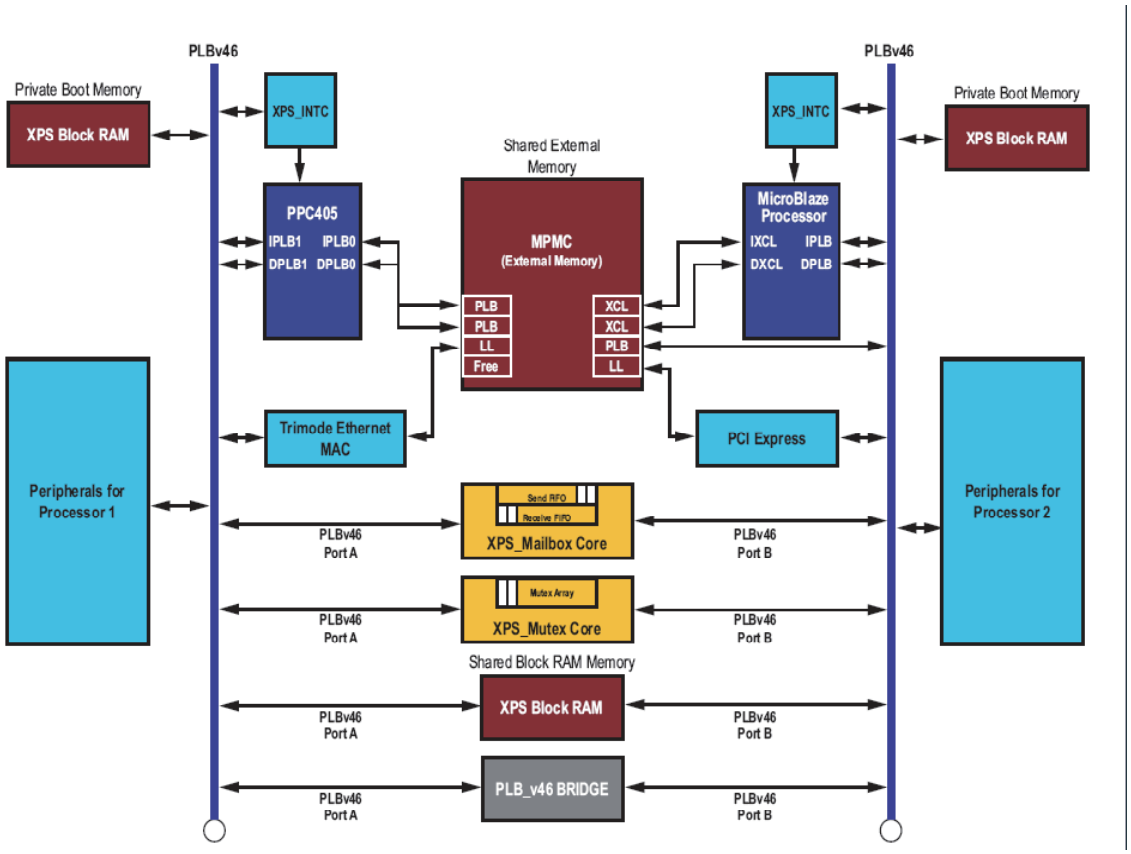
二、简单的可扩展系统架构

您可以看到，利用多处理器可以实现许多种使用模型，同时存在多种系统架构的可能性。为某一应用场合采用明确、可扩展的拓扑和架构是一件令人生畏的事情，所以定义一个可以满足大部分需求的基线架构会有所帮助。

图 6-10 给出了一个双核架构。这个架构展示了简单的可扩展多处理器系统的定义。您可以基于这个定义得到衍生的拓扑以应对设计约束和挑战。该架构的主要概念如下：

这个架构是两套完全独立的单处理器系统的简单扩展，是通过将系统与通信元件连接在一起来实现的。

共享元件全部采用多端口（或者双端口）结构。这些元件的多端口特性使得每个处理器的系统总线在静态和动态负载上都独立于其他总线。通过隔离各个处理子系统，您可以确保系统总线不会因为另一个处理器的事务执行而锁定一个处理器或者外设。所有的多端口外设都在内部完成不同端口上的访问仲裁。



Private Boot Memory: 专用引导存储器

Shared External Memory: 共享外部存储器

Peripherals for Processor1: 处理器 1 的外设

Send: 发送

Receive: 接收

Processor: 处理器

Peripherals for Processor2: 处理器 2 的外设

Shared Block RAM Memory: 共享 Block RAM 存储器

Port: 端口

图6–10双处理器架构

关键共享外设是多端口存储器控制器 (MPMC)。MPMC 通过不同的端口接口访问外部存储器。多个处理器可以通过独立端口连接到 MPMC 上。这种拓扑使得 PowerPC™ 和 MicroBlaze 处理器能够以最小的延迟和高带宽同时访问外部存储器。MPMC 目前最多可以提供 8 个端口，这样就可以将三到四个处理器连接到一个外部存储器上。

这个架构还可以在处理器之间共享内部 block RAM 存储器。片上 block RAM 共享是一种在处理器之间传输 KB 尺寸数据的高速方式。Block RAM 的访问也可以是确定性的，这对于某些应用而言是一个重要要求。

除了共享存储器，还有另外 2 个核 – XPS Mailbox 和 XPS Mutex，可以提供形式简单的处理器间通信。XPS Mailbox 核以同步或异步方式为两个处理器提供了低延迟、FIFO 风格的消息传递接口。它可用于直接传送消息或者用于传送共享存储器中存储的消息指针。您可以使用 XPS Mutex 核在 2 个处理器上为软件共享资源 (无论它们是片上资源还是片外资源) 的访问进行仲裁。总的来说，这些核可帮助您在每个处理器上创建协作软件程序。

一些系统可能期望共享非多端口外设 (比如 UART、SPI 或 I2C)。这种情况需要在没有连至外设的总线和连至外设的总线之间提供一个系统总线桥。图 6–10 给出了一个在两个处理器之间共享 UART 的总线桥的应用情况。

图 6–10 特意指出 PowerPC™ 405 是第一处理器，MicroBlaze 是第二处理器，以说明每个处理器的某些特性。然而，经过很小的改动就可以将任何一个处理器替换成其他的处理器。因此这个架构可以在不同的处理器之间实现无缝转换。

虽然图 6–10 给出了推荐的整体多处理架构，但是不同的约束可能需要您进一步改善该架构。例如，在逻辑面积和资源使用是关键考虑因素的系统中，所有处理器都可以连接到相同的系统总线上。虽然这降低了系统的确定性，增加了总线的运行负载，但是它通过消除新的系统总线以及消除 IP 的多端口需求而节省了面积。

还可实现其他衍生架构，例如在独立系统总线上连接一个高性能处理器，和在共享系统总线上连接多个低性能处理器。您还可以通过利用多级桥连接处理子系统来创建层次化拓扑。EDK 所提供的各种工具和 IP 使得您可以进一步优化这个基本拓扑直到其满足您的需要为止。

三、其他考虑因素

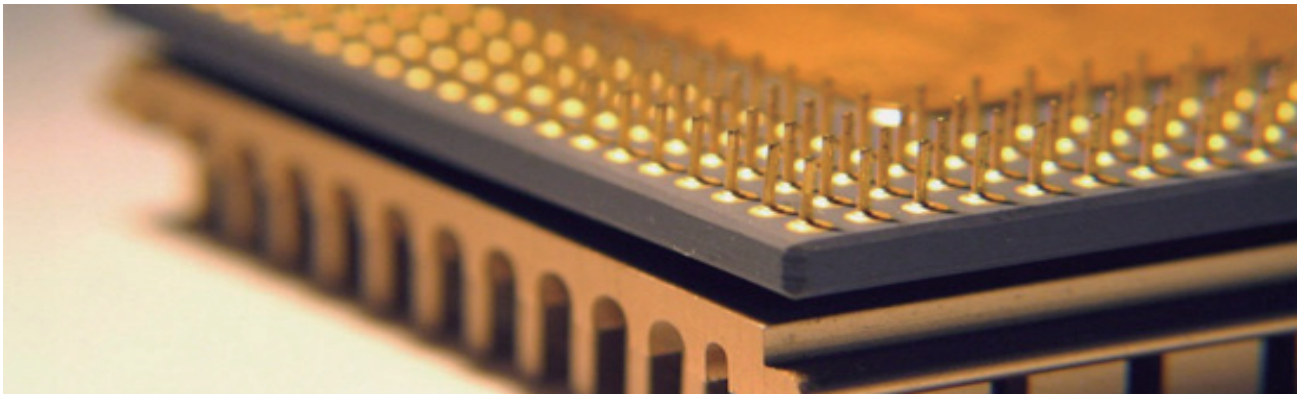
通常情况下，多处理器架构还需要考虑一些其他因素。例如，您需要在两个处理器之间以非冲突方式定义存储器映射。EDK 所提供的自动地址生成工具将这一任务简化成简单的按钮操作。

您也需要考虑您的时钟和复位网络。您可以选择以相同的速率来为所有处理器计时，或者让每个处理器使用不同的时钟域。同样，您也可以将复位域定义在不同层次上，例如仅处理器复位、处理器子系统复位和系统复位。处理器必须独立连至调试端口，从而为每个处理器提供一个独立的调试进程。

除了硬件因素，您还需要设计软件系统，以便他们能够协同运转。这包括使用共享存储器、消息传输以及一些常见的同步概念，例如障栅 (barrier) 和会合 (rendezvous) 等，从而使系统具有可预测性和同步性。同时提供了包含更高级通信范例的商用软件栈。

总结

了解多处理器系统的各种应用可能性方面的信息，敬请参照本文的白皮书版本 (www.xilinx.com/cn/support/documentation/white_papers/wp262.pdf)。 您也可以参照赛灵思应用指南 XAPP996 – 双处理器参考设计套件 – 上所介绍和提供的参考设计，网址是 www.xilinx.com/cn/support/documentation/application_notes/xapp996.pdf。请继续关注未来 EDK 工具的新特性，例如自动多处理器设计创建和协作调试支持。带有功能强大的 PowerPC™ 440 处理器的 XilinxVirtexTM–5 FXT 平台也为创建超高性能多处理器系统提供了无限可能性。



6.6 利用 JTAG 链进行更为精确的系统级和芯片级功率分析和热分析

作者：Dominic Plunkett/首席技术官

XJTAG

dominic.plunkett@xjtag.com

工程师通常采用边界扫描链对 CPLD 或闪存器件进行编程。 但工程师们应当更多的探索利用边界扫描的威力来获取电路板或系统运作的更详细信息。 随着赛灵思公司在其最新的 Virtex™–5 FPGA 器件中引入了 Xilinx® System Monitor 功能，利用传统上用于边界扫描功能和器件编程功能的联合测试行动组 (Joint Test Action Group) 测试访问端口 (TAP) 就可以收集 FPGA 内部的电压和温度信息。

配合赛灵思公司合作伙伴 XJTAG 公司的设备和测试脚本，或者通过自己编写脚本，还可以在边界扫描测试环境中更容易地验证电路不同点的模拟信号。 XJTAG 的边界扫描测试系统可帮助工程师测试诸如分享温度传感器、DAC 或 VGA 端口等器件。

一、System Monitor: 器件级测试探头

System Monitor 是赛灵思 Virtex–5 FPGA 架构中集成的模拟电路，可对芯片内部的温度和电压进行采用 (图 6–11)。 专用的 System Monitor 基于一个 200–ksps ADC，能够对 FPGA 片芯温度，以及 VCCINT 和 VCCAUX 电源电压进行连续的顺序测量。Virtex–5 器件的这一功能避免了在系统中实现外部监控所带来的复杂性和成本。更重要的是，System Monitor 功能使你可以从芯片本身内部进行电源测量，而这是外部 ADC 无法做到的。

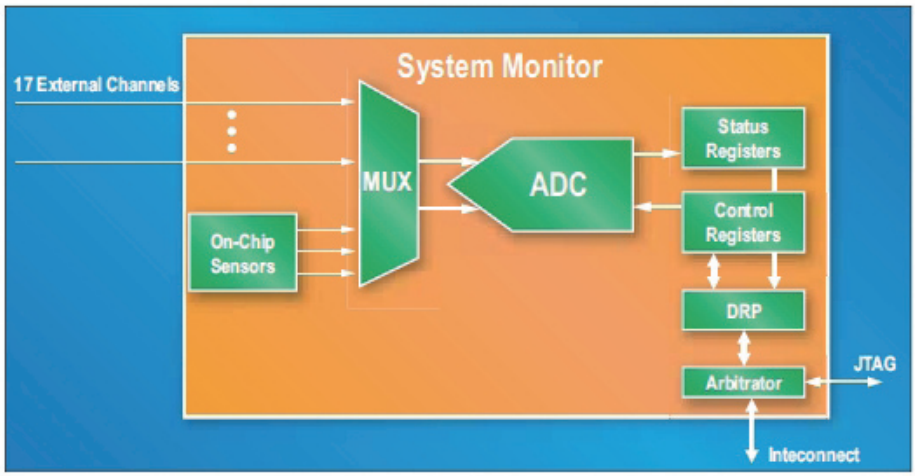


图6–11 System Monitor可以监控Virtex–5 FPGA片芯上的电压和温度以及多达17个其它可能影响总体系统性能的模拟信号源。

利用 System Monitor 可以获得准确的温度读数。 传统上工程师都是利用热敏二极管来监测芯片温度，但必须仔细注意 PCB 布局，因为二极管测量对于来自其它器件的噪声、PCB 板上其它模块的影响以及信号偏置和公差等细节都非常敏感。 新的 System Monitor 功能避免了这些问题。 通过在 FPGA 片芯上集成温度传感器，设计人员可以从片芯本身获得准确的温度读数 (图 6–12)。

这些温度和电源监测功能可用于实现上电检测或过热保护等安全保护功能。 System Monitor 的内部电源测量精度达 ±1%，而片上温度传感器的准确度在 –40° C 至 125° C 范围内也达到 ±4° C。

System Monitor 电路还集成了一个复用器，因此不仅可以支持来自芯片内部的两个电压传感通道和温度传感器，还可以接收来自 FPGA 外部的多达 17 个模拟源输入。 这意味着可以利用 System Monitor 来监控多种片外模拟信号。 支持的片外输入信号可以是单端的，也可以是差分信号，幅度可达 1.0V，因此可以连接到多种传感器，包括基于分流电阻的传感器、加速度计、位置传感器和外部温度传感器。System Monitor 控制系统还包括一个自动通道排序器，可以定义希望监测的参数。 通过设置控制寄存器就可以实现 System Monitor 的配置。要实现这一点，只需要在设计中建立 System Monitor 实例或者通过 JTAG 写入控制寄存器就可以了。 赛灵思 ISE™ 软件工具套件还提供了一款称为 System Monitor 架构向导的工具，可帮助完成 System Monitor 的实例建立过程。

图6–12 位于Virtex–5器件中的 System Monitor传感器可在产品开发、批量生产甚至现场部署后提供片上电压和温度有关的宝贵数据。

二、使用TAP接口

System Monitor 能够直接通过 FPGA 的 JTAG TAP 提供这些数据，因此赛灵思新器件为从原型或生产系统中获取详细信息提供了新的手段和机会。 更重要的是，你可以在更高的抽象层次将数据导入到扫描测试链。

虽然大多数工程师都熟悉利用边界扫描进行器件编程，但许多人并没有意识到边界扫描还是一种功能强大的非侵入式调试和测试手段。 通过 JTAG TAP 实现测试意味着不再需要将测试探头物理上连接到开发板或生产线上。 连接探头即耗费时间，同时还容易引起测试错误。

更为重要的是，对于现代 BGA– 和 CSP– 型封装的器件来说，采用物理探头直接测量是不可能的，因为球栅距太小并且还被盖在器件下面。 随着业界在新 IC 设计时越来越多地采用这些封装，采用传统测试技术能够达到测试覆盖率也相应降低。

由联合测试行动组 (Joint Test Action Group) 制定并以 IEEE1149.1 形式发布的边界扫描测试采用了四线 TAP 端口和边界扫描架构，工程师可通过在 IC 设计中实现 JTAG 来为 IC 开发周期后端的产品测试提供便利。通过四线 TAP 接口，工程师可以读取或写入 JTAG 引脚或内部寄存器数值。 通过这些引脚，可以对电路板上的器件进行测试和测试，包括 FPGA、EPROM、RAM 以及闪存器件。

利用边界扫描测试设备，不需要依赖嵌入式测试软件或测试功能就可以快速进行网表级 (net–level) 诊断。使用 JTAG 并不一定需要 PCB 能够正常运行，因此一旦原型设计组装好，甚至在电路板没有启动的情况下也可以快速验证基本功能。 同时还可以进行基本的检查、独立设置引脚或总线电平来发现短路、虚焊、断线或错误连接等问题。

系统中所有 JTAG 器件的测试端口在板级互连在一起，构成一个串行扫描链，因此可以通过板边缘的单个连接器访问这些器件。 随着板上 JTAP 兼容器件数量增多，JTAG 测试仪可以访问所有网表中越来越多的部分，从而提高了总体的测试覆盖率。

通过控制扫描链建立适当的测试模式，还可以收集那些不提供边界扫描测试接口的器件的响应，当然前提是你希望访问的器件以兼容器件方式连接到同一 JTAG 网络。 这一技术有时被称为聚类 (分组) 测试 (cluster testing,)，从而为测试外部连接器、视频芯片、IIC 器件、以太网控制器、LED 或开关等非 JTAG 器件提供了一种方法。

例如，你可以利用 JTAG 链驱动以太网 (Ethernet) 测试数据包到板上的非 JTAG 以太网控制器，并验证其响应 (图 6–13)。 同类，还可以测试 SRAM、SDRAM、DDR 和 DDR2 芯片，即使它们并不支持 JTAG。 这些例子显示出现代边界扫描设备 (如 XJTAG 系统) 如何通过将 JTAG 链的潜力发展到极致而大大扩展了测试覆盖范围。 在实践中，XJTAG 客户现在已经可以在复杂板上实现 90% 以上的测试覆盖率。

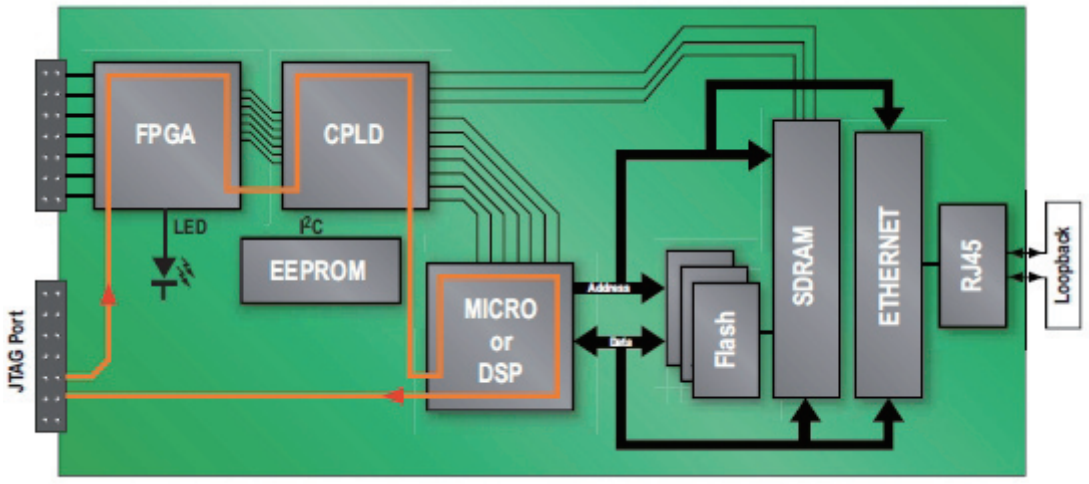


图6–13 得益于现代PCB设计中不断增强的JTAG兼容性，现在可以利用边界扫描完成更多类型的任务和测试。

三、扩展边界扫描功能

赛灵思 System Monitor 使设计人员可以方便地采集 Virtex–5 FPGA 内部的模拟数值，然而，已经熟悉边界扫描的设计人员还可以通过 TAP 接口实现另外一种非常有价值的测试。 Virtex–5 器件用户可以将 17 个外部模拟通道的数据通过 System Monitor 引入到边界扫描测试环境。

使用 Virtex–5 System Monitor 功能要求 PCB 设计工程师遵循一些基本的设计指导规则。 这些指导原则包含在 Virtex–5 System Monitor 用户指南的应用部分

(www.xilinx.com/support/documentation/ user_guides/ug192.pdf)。 一旦 PCB 提供支持，就可以充分发挥 System Monitor 的潜力了。 例如，由于 System Monitor ADC 一上电即可工作，因此可以在配置 FPGA 前利用边界扫描链获取宝贵的系统信息。 这样开发工程师可在 FPGA 配置完成前验证系统电源轨以及评估基本的冷却条件。

System Monitor 功能的采样电路是在 FPGA 芯片中以硬件形式实现的，因此可以在产品生命周期的任何时间使用其功能。 例如，如果系统始终出现过热或电源在实验室或现场环境中显示有问题，那么不必重新配置 FPGA 就可以立即通过 JTAGTAP 端口进行测试，读取相关数据。

四、XJTAG简化了边界测试的新兴应用方式

XJTAG 公司的工程师与赛灵思公司密切使用，基于 Virtex–5 LXT ML505 开发板开发了几种立即可用的测试脚本，支持 XJTAG 边界扫描测试解决方案用户从 System Monitor 访问数据。 XJTAG 可以直接将片芯温度、供电电压和片外模拟量数值显示在工作站上，从而可以方便地验证 System Monitor 和 XJTAG 采集的数值在规定的容限内。

图 6–14 是一段简单的测试脚本代码，用来检查 Virtex–5 FPGA 的工作参数。用户可利用 XJTAG 的高层脚本语言 XJEase 来生成所需要的脚本。 在图 6–14 中的 Test 函数中，首先配置 System Monitor。 然后读取相应的输入并检查之。

```
//----------------------------
Test()(INT result)
//----------------------------
INT temp;
  result := 0;
  Config_SysMon()();

  Read_Internal_Temp()(result);
  Read_VAUX()(result);
  Read_VINT()(result);
  Read_RefP()(result);
  Read_RefN()(result);
  IF (result != 0) THEN
    result := RESULT_FAIL;
  ELSE
    result := RESULT_PASS;
  END;

END;
```

图6–1 4. 下面的代码显示出如何利用XJEase 脚本语言设置和测试某些System Monitor输入

图 6–15 中的代码检查器件的温度。 如果温度超出容限，测试过程中会显示错误。 这一例子给出了如何

利用 XJTAG 网站上提供的简单功能来使用 Virtex–5 System Monitor 功能。XJTAG 还支持定制测试。例如，记录 System Monitor 数据用于故障检测或跟踪。通过创建脚本可以让 XJTAG 检查峰值、响应阈值或在 System Monitor 获取的数据上使用平均算法。

下面是用于测试 Virtex–5 FPGA 温度的代码段

```
//------------------------------------------------------------------------
----------
Read_Internal_Temp()(INT result)
//------------------------------------------------------------------------
--------

    INT temp;
 Read_Temp()(temp);

     IF ((temp < (Die_Temp – Temp_Margin)) ll (temp > (Die_Temp + Temp_Margin))) THEN
 PRINT(“Die temperature =“,temp,”C. ** OUTSIDE LIMITS **\n”);
 result := result + 1;

     ELSIF (DEBUG) THEN
PRINT(“Die temperature =“,temp,”C\n”);
 END;
END;

    //------------------------------------------------------------------------
----------
Read_Temp()(INT temp)
//------------------------------------------------------------------------
--------

    Read_Channel(0)(temp);
 temp := ((temp * 20159) / 40960) – 273;
END;
```

做为 XJTAG 环境的一部分提供的 XJEase 方便了脚本的创建。 XJEase 支持以更高层抽象语言操作原始边界扫描数据。 无论器件是否在边界扫描链上，都可以创建针对特定器件的测试脚本。 这些脚本可以存储起来，并且可供未来项目重利用。

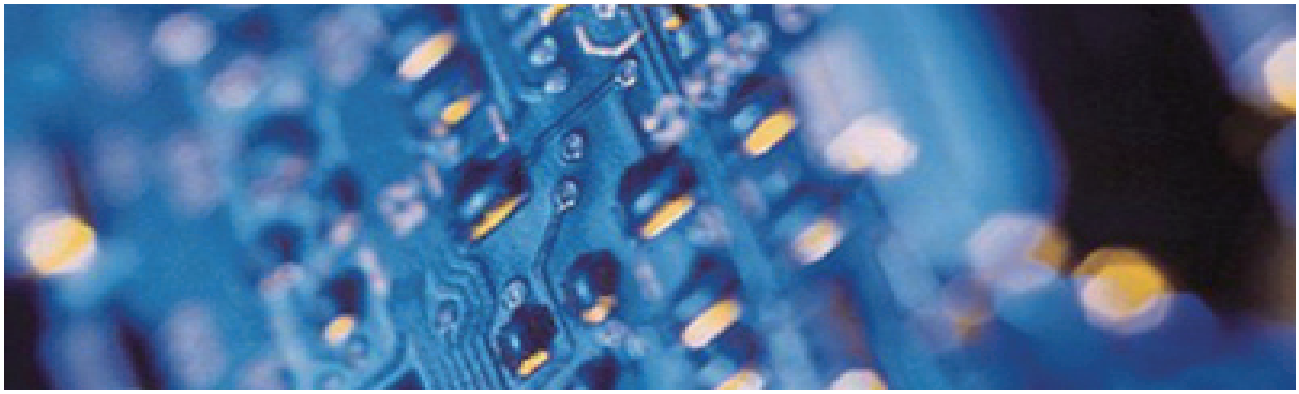
每当网表数据显示板上有变化，XJTAG 都会自动生成新的测试序列，从而节约了手工重写测试程序所需要的时间。 与此相比，较早的边界扫描测试设备生成的测试序列通常是与板的设计有关的，每次设计修改都需要进行更新，即耗时又费力。

随着这一面向器件的扫描方法在工程师群体的测试开发过程中越来越普及，许多 XJTAG 用户已经开发将经过验证的预编写脚本发布到 XJTAG 网站上。 与 XJTAG 签有技术支持合同的客户可以免费从 www.xjtag.com 网站下载这些脚本并将其集成到自己的测试程序中。 同时，如果网站上没有你所需要的完全相同器件的脚本，那么只需要简单地下载一个类似的脚本，然后再针对待测器件进行定制修改就可以了。

五、未来展望

System Monitor 和 XJTAG 之间的交互突出了边界扫描测试系统在支持创新片上功能的验证、调试、参数设置和校正方面具有极大的潜力。 通过使用 FPGA 或其它器件 (如处理器、存储器或控制 IC) 内核中集成的硬件功能，边界扫描有可能完成许多功能，如虚拟 DIP 开关、校正数字滤波器或调整模拟设置或阈值数值。 此外，这一切都可以快速实现，无论产品是处于开发阶段，已批量生产还是正在现场使用。

对于已经在使用边界扫描来测试原型或生产线产品功能的工程师来说，System Monitor 现在支持在边界扫描环境中监控关键信号。 对于更为激进的工程师，还可以充分利用这一能力对系统进行更为深入的监控。 未来，随着这一技术变得更为主流，可以预见会出现更多测试脚本用于监控产品设计中最不好监控的部分。



6.7 识别和解决赛灵思 FPGA 设计中的时序问题

作者： Chris Dunlap/现场应用工程师

美国赛灵思公司

耗费数月精力做出的设计却无法满足时序要求，这确实非常令人伤心。 然而，试图正确地对设计进行限制以保证满足时序要求的过程几乎同样令人费神。 找到并确定时序约束通常是非常令人头痛的问题。时序问题的恼人之处在于没有哪种方法能够解决所有类型的问题。 当然好消息就是通过将软件工具的潜力发挥的权限以及通过优化 RTL 代码，可以解决大多数时序问题。

由于客户对于和现场应用工程师 (FAE) 共享源代码通常非常敏感，因此我们通常都是通过将工具的潜力发挥的极致来帮助客户解决其时序问题。

但在深入探讨之前，我们首先需要对时序问题进行一点基本分析。 这里的目标是首先排除明显的问题，如将时钟引脚置于器件的上部、在器件下部驱动数字时钟管理器 (DCM)，然后再在器件上部驱动全局缓冲 (BUFG)。

有时，此类引脚布局会导致根本没有办法满足时序要求。 通过察看时序报告中的延迟，通常可以发现这些明显的时序问题。 在这些情况下，为了解决这些明显的问题，都需要利用底层规划工具 floorplanner 将造成问题的部分锁定在适当的位置。 平面布局规划工具还可以帮助以可视的方式来理解时序问题。

一、使用最新的软件工具

假设问题并非这么明显，那么为了锁定问题所在我们需要了解所使用的器件系列以及软件版本。 通常，每种器件系列对应一种最优的软件版本。 Xilinx® Virtex™–4 器件对应的最佳软件是 ISE™ 软件 9.2i 版，而对 Virtex–5 FPGA 则是 ISE 软件 10.1 版。（为准备本月的这个栏目，我使用的是 ISE 软件 v10.1 和 PlanAhead™ 软件 v10.1。）

综合工具的版本也很重要，因此当采用最新的器件架构时，下载并使用最新版软件非常重要。 软件开发几乎总是滞后于硬件功能，因此我不提倡使用旧版软件进行基于新器件的设计。不幸的是，有些客户由于担心新的和未知的软件缺陷而不愿意升级软件。 但是，在使用最新的器件时，如果希望更好地处理时序挑战，我强烈建议下载最新版软件。

拥有了最适用于目标器件系列的软件，还需要确定最佳的实施选项 (implementation option)。 不幸的是，

并没有适用于所有情况的超级选项组合。 对于设计实施工具来说，有成千上万种不同的实施选项组合。 根据所使用的实施选项不同,时序分数(所有存在错误的时序路径与时序要求的差异总和,以皮秒表示)也会有很大不同。

赛灵思的几款工具可帮助确定适用于特定设计的最佳实施选项。 ISE 软件现在包括两个工具：Xplorer 以及最近发布的 SmartXplorer。 SmartXplorer 可充分发挥多处理器优点，能够以不同选项组合运行多个实施实例。SmartXplorer 需要 Linux 支持，但使用非常容易。 其命令行很简单：smartxplorer designname.edn –p xc5vlx110t–1ff1136.

只要用户约束文件 (UCF) 和网表约束文件 (NCF) 文件名相同，SmartXplorer 会自动使用正确的选项。 唯一需要做的是编辑主机列表文件。

实施选项	时序得分	运行时间
Map : -timing -ol high -xe n -global_opt on -retiming on Par : -ol high	651262	6h 26m
Map : -timing -ol high -xe n Par : -ol high	0	4h 25m
Map : -cm area Par : -ol high -xe n	380834	5h 19m
Map : -timing -ol high -xe n -register_duplication -logic_opt on Par : -ol high -xe n	0	5h 46m
Map : -timing -ol high -xe n -global_opt on -retiming on -ignore_keep_hierarchy Par : -ol high -xe n	666567	12h 34m
Map : -cm balanced Par : -ol high	172171	4h 4m
Map : -timing -ol high -xe n -pr b -cm balanced Par : -ol high	7235	6h 33m

表6–6： 对基于Virtex–5 FPGA的设计SmartXplorer 10.1的一个例子

SmartXplorer 可以通过 SSH/rsh 安全 shell 登录到其它机器。 只需要在名为 smartxplorer.hostlist 的文件中将每台机器一行将机器名字添加进去就可以了。 如果机器有两个处理器，请将机器列出两次。 表 1 给出了 SmartXplorer 的一组结果。

PlanAhead 软件也包括了与 SmartXplorer 类似的称为 ExploreAhead 的功能。ExploreAhead 支持同时在多台 Linux 机器上分布式运行布局布线任务。 所有这些工具的目的都是类似的：确定实施工具的最佳选项组合，以获得最好的时序得分。

请注意选项的不同组合对于时序得分和运行时间的巨大影响。 仔细调整综合选项也非常重要。 例如，在综合选项中关闭结构层次 (hierarchy) 通常会大大提高性能。 综合过程中的约束条件好坏在满足时序方面的作用也很突出。

二、PlanAhead软件

在了解实施选项对最佳时序分值的影响之后，现在可以开始有效地分析时序问题了。 这时候，PlanAhead是一款非常有价值的工具，可以视觉化显示布局布线后的设计。 利用它，还可以导入时序约束并在已布局窗口交叉探查 (cross–probe) 时序失败的路径。

当工具本身的决策不好时，则可以通过平面布局模块或通过手工布局部分组件的方式来纠正。 这一过程通常需要反复多次，才能够确定时序优化的最佳设计布局方式。 PlanAhead 软件的可视化功能确实使这一工作的完成更容易了。

利用 PlanAhead 软件，首先创建项目，然后将 HDL 或网表文件导入工具中。 一旦创建了一个项目 (project)，就可以选择 File → Import Placement。 选择时序优化效果最佳的布局布线后 (post PAR) (ncd) 文件，将布局布线信息导入 PlanAhead 软件项目。

软件会将 PlanAhead 项目组织到几个不同的窗口。 左上窗口是物理分层 (physical hierarchy) 窗口，描述了设计中的当前区域组 (area group)。 选定窗口 (selection window) 在下面，包含了当前选定的数据详细信息。中间窗口是网表窗口 (netlist window)，给出了整个网表的分层结构。 最右侧窗口是器件视图 (Device view)，里面已经充满了设计实施完成后的逻辑。

然后，将时序分析报告 (TWR/TWX) 导入到 PlanAhead 工具中。 选择 File → Import TRCE Report。 这一步将时序报告数据添加到底部窗口。 按照时序余量 (timing slack) 对这一窗口进行排序可以将焦点首先集中于违反时序要求最多的地方。 经常的情况是解决了这些时序偏差最大的地方所存在的问题也就解决了整个设计的时序问题。

一旦选择了一条时序失败的路径，PlanAhead 软件就会选定时序失败路径上的实例和连接。 按 F9 键放大显示选定的部分 (参看图 6–16)。

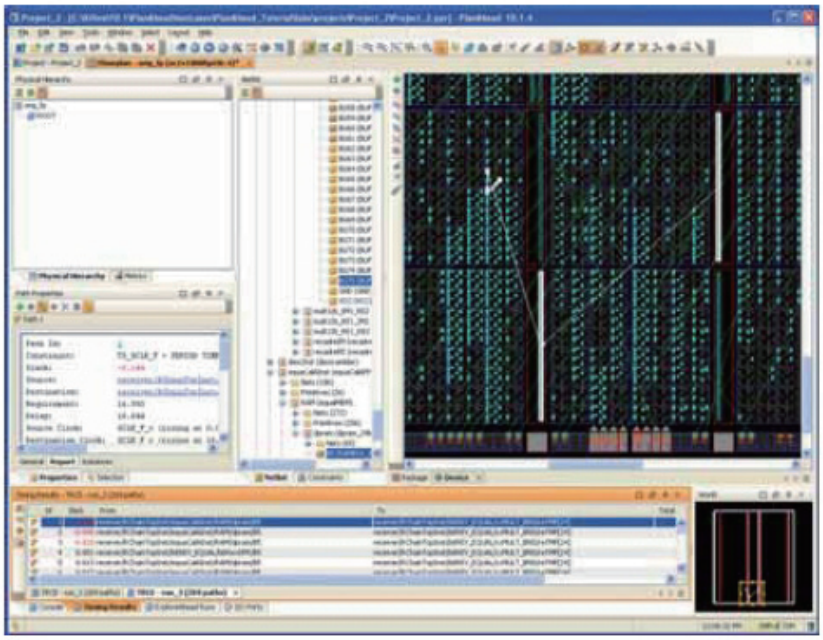


图 6–16. PlanAhead 10.1版显示出已实施的设计，一条时序失败路径高亮显示

这里显得有些复杂，必须进行一定的审查和思考才能够明显布局布线工具将基本例化单元 (primitive) 放在某个地方，以及为什么对于当前的特定设计来说还有更好的方法。你可以放大显示任何高亮的基本例化单元。鼠标点击一个基本例化单元并拖动，可以更好地观察其连接情况。 在网表窗口，还可以移动到包含了所选实例的顶层模块。 鼠标右击模块名称并为选定的实例选择颜色。 这样就可以显示出这一模块在芯片中的布局，以及组内单元是靠近还是散开的。

你会发现，有时需要更好地锁定特定的基本例化单元。 Block RAM 和 DSP 模块的自动布局是导致时序失败的常见原因。 图 6–16 中，容易发现布局布线 (PAR) 工具将 Mult18 安排的位置很不好。 时序失败路径中的块 RAM 输出连接到 Mult18，后者的输出又馈送到进位链。 块 RAM 在上部、Mult18 在下部，而逻辑部分又位于上部。 如果布线不需要上上下下、边边角角地来回绕，该路径应当可以满足时序要求。

时序问题不同，在 PlanAhead 软件中处理这些时序问题的解决方案也跟着变化。 在 PlanAhead 设计工具中解决时序问题需要不断实践。 做为例子，让我们来解决图中的时序问题。

我们的解决方案体现以下操作步骤中：

选择高亮显示失败的时序路径。

右击路径上的某个实例，选择 Highlight With → color of choice。

在左侧栏中，将底部第二个 Mult18 释放。 右击 Unplace。 这将为时序失败的块 RAM 腾出空间。

点击拖动底部的 Mult18 向左上移动一个位置。

点击拖动右边的块 RAM 到底部左侧的自由块 RAM 位置。

选定失败的时序路径，确认路径看起来是优化的 (参看图 2)。

选择 Tools → Clear PlacementConstraints。 点击第一个选项中的 Next。

选择 Unplace All But SelectedInstances。 在余下的向导步骤点击 Next。

如果希望在 PlanAhead 软件外运行实施流程，选择 File → Export Floorplan。

工具会输出一个新的文件名为 top.ucf 的 UCF 文件。 你可直接使用这一文件，或者将文件中的约束加入到原始 UCF 中。

另一个选择是在 PlanAhead 软件内运行实施工具。

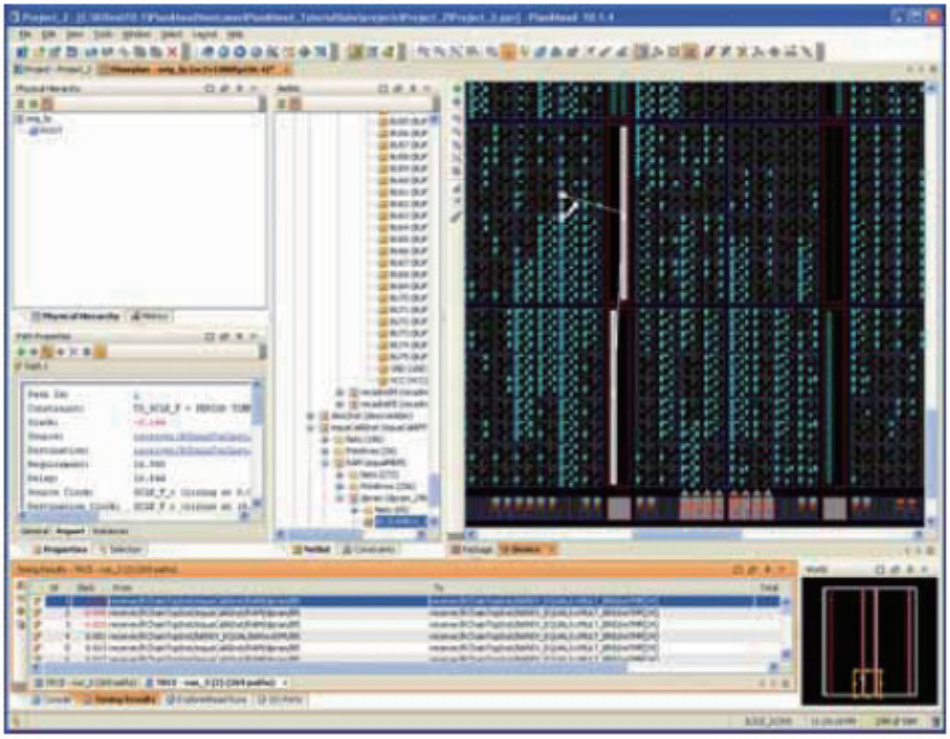


图 6–17 PlanAhead 10.1版软件显示出将DSP48和块RAM锁定后的正确路径。

选择 Tool → Run ISE Place & Route with ExploreAhead。

ExploreAhead 提供了许多很好的功能，包括：

a) 自动从器件视图 (Device view) 导入约束到 UCF。

b) 简化了从 ExploreAhead 中运行的布局和时序结果的导入。

二、Pblock和平面规划(Floorplanning)

如果发现布局中有许多时序失败路径，手工布局调整通常无法解决问题。 反过来，应当创建区域组 (area group)。

创建区域组 (Pblock) 的方式之一是在网表窗口中右击模块名字并选择 Draw Pblock。 然后在器件视图 (Device view) 中希望放置区域组的地方画一个长方形。

工具会创建一个 Pblock，同时会显示有关的详细信息。 属性窗口会显示 Pblock 需要的逻辑资源以及画出的长方形区域组中可用的资源。

设计的平面规划是一个需要深度交互的过程。 你可能会发现需要反复多次调整区域组才能够达到目标。 请记住有时创建的区域组越小越少越好。 你并不是总需要将整个模块固定为区域组。 集中于时序失败的基本例化单元，将它们组合为区域组并单独为其进行平面规划。

在此过程中，应当利用 Pblock 指标 (Metrics) 来更好地理解有关功能，如区域组中可配置的逻辑块 (CLB) 的利用情况如何 (可以在 PlanAhead 软件左上窗口中点击 Metrics 标签来查看可用的指标)。 这可帮助确定是

否某个特定区域中太拥挤以致妨碍布线。 如果确实比较拥挤，可能需要通过平面规划将设计逻辑放得散开一些。

一旦利用 PlanAhead 软件获得尽可能好的时序分数，最后的任务就是优化代码。 PlanAhead 设计工具现在支持 HDL 源文件。 根据你导入的源文件不同，你可以从时序问题出发通过交叉探查功能回查到网表或 HDL。

利用原理图视图 (Schematic view)，可以察看整个时序路径。 只需要从时序结果中选择时序路径并按 F4。仔细观察从其它模块扇入扇出的逻辑。 由于外部接口要求，工具可能经常需要将模块在芯片上散开布署。 如果是这样，可能需要在设计使用流水线。

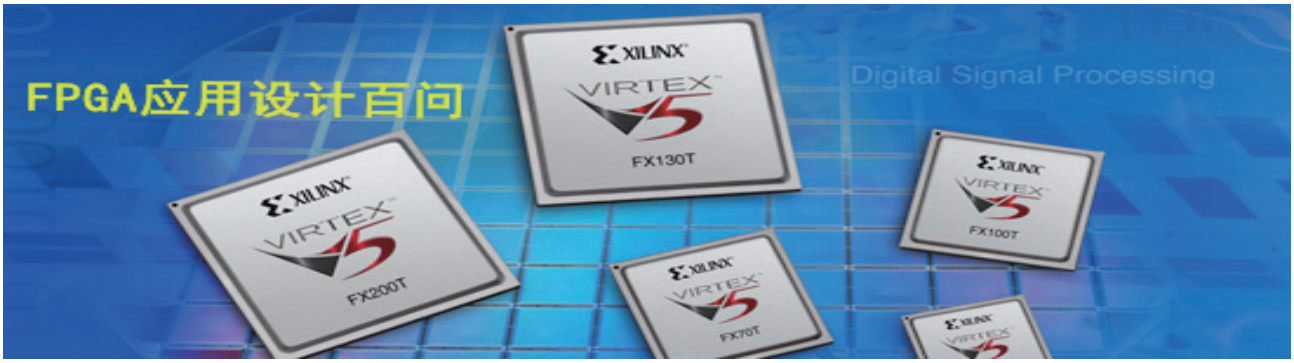
Block RAM 和 DSP 输出时序是引起时序失败的常见原因。 通过在这些模块的输出使用寄存器，通常可以记得一纳秒甚至更多的时间。

三、最后建议

如果你碰到时序收敛问题，但又没有 PlanAhead 软件，我强烈建议你尽快获得这一软件。 该工具为解决时序问题提供了极大的方便。

利用 PlanAhead 设计工具，你可以运行 ExploreAhead 来确定最佳的设计实施选项组合。 同时由于 PlanAhead 软件并没有捆绑到特定的 ISE 软件版本，因此如果安装有多个不同的 ISE 版本，那么你可以尝试不同的 ISE 软件版本。 PlanAhead 软件的交叉探查 (cross–probing) 功能使你能够更深入全面的了解设计的源和最终实现。

时序收敛确实是相当令人头痛的问题，但有了正确版本的正确工具以及全面的时序收敛计划，你肯定能够快速确定设计中的时序问题并实现时序收敛。赛灵思现场应用工程师专家 (FAE–X) 保证你拥有愉快的设计体验。



第七章、FPGA设计百问

1、问：请教控制 XST 插入 buffer 的方法

答：1、用 buffer_type 约束。具体使用方法在 XST User Guide

2、手动插入 BUFG，然后设置允许使用 BUFG 的数量，那么手动插入的将拥有高优先级而先占用了 BUFG。

2、问：ISE 软件中给出的综合报告与静态时序分析报告中都含有工作频率，请问哪一个是 FPGA 能够实际工作的频率?

答：经过映射和布线后的频率值才是 FPGA 可以真正跑到的频率值看静态时序分析报告，基本是这个数值。通过查看 ISE9.1.03 的 Implement Design>Place&Route>Generate Post–Place&Route Static Timing 的数据，可以查看到比较接近实际情况的报告数据。

3、问：在后端布局布线之前怎么确定系统的最高工作频率？是一点一点的往上升频还是有什么计算公式？在综合之后呢？综合用的 std 基本上也没有什么延迟信息吧?

答：看时序报告，里面有 fmax，推算的方法是根据关键路径来计算的，也就是逻辑中延时最长的那条路径，这条路径的能满足的建立保持时间就是电路中时序部分能达到的最小周期。

4、问：modelsim 仿真报错 comparetest.v[1]:near "t":illegel base specifer in numenic constant 是什么错误?

答：语法错误。检查第一行第一个符号，区别单引号‘和预编译符号`。

5、问：什么综合器能看到每个子模块的面积？好像 synplify 不行。有哪个综合器可以呢?

答：ISE 10.1 在 PAR 之后有报告，可以分模块报告；Synplify 老版本没看到有分模块的。

6、问：请教关于 DCM 的 LOCKED 信号，使用 DCM 的时候遇到了一些问题，在每次开关电源的时候，好像因为电磁干扰的影响 (怀疑是)，用示波器观察，DCM 的输出时钟突然没了，而输入时钟是还有的，但 LOCKED 信号一直为高。进一步观察，发现在开关电源时，输入时钟的确会产生抖动，这样导致 DCM 失锁所以产生不出时钟了吗？但为什么 LOCKED 信号一直为高？望指教，谢谢！

答：请参考 UG–190，P57 前后，Input Clock Changes 部分。

7、问：1.synplify 的综合出的网表是否是其它 EDA 可以通用的？

2.synplify 的综合出的网表与绘制 PCB 板所调用的网表有什么不同，格式是否相同？

3. 我在 synplify 中找不到其综合出的网表文件，只看到可视元件。不知最终给出的网表文件是什么？

4. 目前的 FPGA/ASIC 综合工具其综合的网表是否有统一规范？

答：1、syplify 作为第三方工具，综合处的网表是其他的 EDA 工具可以通用的，这里指的 EDA 工具是指设计 FPGA 的 EDA 工具，而不是 protel 什么的。

2、PCB 的网表和 synplify 的网表不是一个概念，pcb 的网表是分立元件的连线和约束网表，而 synplify 综合出的网表是用于 FPGA 内部布局布线用的。

3、 synplify 貌似可以综合处不同后缀名的网表，你看下你的工程名和指定的后缀，然后就可以找到了。

4、目前的 FPGA/ASIC 的综合工具的网表不是很统一的，但是大同小异，你懂的一种就很容易懂另外的。

8、问：可否解释下 FPGA 时序约束设置方法，目前在做一个项目，功能仿真正确，但想进一步优化设计，采用怎样的时序约束才能达到目标呢？

答：优化代码，可以从代码风格的角度出发，这就涉及到对整个系统的理解，例如如何对系统进行划分，是否进行有必要的流水或者重定时。时序约束包括时钟约束、关键路径的约束等等。这还是要求你对整个设计有深入的理解。时钟约束是全局约束，在 Synplify Pro 下很好实现，在 SDC 中定义时钟频率即可。在代码上的小打小闹对系统的时序影响不大，主要是对系统的分析工作做透了，像上面说的，对系统模块的划分，比如说相同功能的模块放到一个模块中去，若是可能时钟只用一个，上全局等等。

9、问：请教一个关于 fpga 芯片引脚的问题 (非常奇怪啊) 补充资料：产品型号是 Virtex II pro xc2vp40 FG676。现在发现引脚 U12,U13 和引脚 A1 短路，但是引脚 U12,U13 为 bank5 的 VCCO，外面接的是直流 3.3V，引脚 A1 为 GND，接的是数字地。同样引脚 U14,U15 为 bank4 的 VCCO，接的是直流 3.3V，它们也和引脚 A1(GND，接的是数字地) 短路。由于我们买了两片这个型号的芯片，所以两片的引脚都相同的测了一下，发现两片芯片的状况一样。

所以我觉得芯片本身可能没有问题。但是这又很让人难以理解，而且如果一加电的话，那么直流 3.3V 电源就会和数字地 (GND) 直接短路，所以我们现在迫切想得到哪位大虾的答复。

答：首先要确定你的测试操作正常，在此前提下注意：

1、搭焊或松香等脏东西未擦干净 (未焊接除外)

2、反向二极管，或者是使用时管脚已经烧坏 (如输出低电平时外加了高电平)

3、你可能被经销商忽悠了，给你的是处理过的旧片子 (遇见过多次，尤其是在中关村)

4、FPGA 下面的管脚可能短接了赛灵思的 FPGA 有些电源管脚和地之间的电阻值只有几十欧，有些万用表会报短路。

10、问：如何察看特定信号的 fanout？我的设计中信号很多，成千上万个。请问我在用 synplify 综合后能不能查看特定信号的 fanout? 如果能按从大到小排列就更好了。请指点!

答：在 RTL 视图下，选中待查看信号对应网线，然后在左边的窗口中就会高亮显示 (在以 Nets 命名的文件夹里，会显示该网线的 Fanout)

11、问：ISE 中的 PAD TO PAD CONSTRAINT 是否是包括输入输出的 pad 时延之和再加上输入输出之间的组合逻辑的时延？还是只是输入输出之间的组合逻辑的时延?

答：Xilinx PAD–to–PAD contraint 的确涉及到输入输出 PAD 时延 . 这从布局后时序报告中可以看出。

12、问：写了一个 vhdl 的小程序 , 可是在 processes of soure 里的 implement design 里只有 translate,fit, 却没有 map ,place&route. 这是为什么呢？请大侠们帮忙答疑解惑！原来用的上 xc9500xl 系列 的 xc9572xl, 结果怎么都没有 place&route. 后来换成 virtex2 系列的 xc2v80 就出现 了 , 这是什么原因？

答：cpld 是没有 place&route，是 fit

13、问：SpartanIII 芯片采用 master serial 方式配置时，是不是也需要通过下载电缆把数据下载到配置 ROM 中？那么下载电缆应该如何连接？是采用 JTAG 方式吗？那 M2,M1,M0 应该设成什么?

答：serial 串行。只需用到 d0。其他 d1–d7 可用于 user io。速度慢，不过一般选这个，Parallel 并行。

需要用到 d0–d7. 速度快，如果 jtag 链有多于一个 FPGA，则第一个用 master，其他用 slave。

如果只有一个就用 master。

每个 FPGA 可以通过 m0,m1,m2 进行设置。

配置 FPGA 或 prom 的时候都会用到 jtag。

一般调试的时候只需配置 FPGA 就可以了。

实际使用时需要先配置 prom，这样每次加电时 FPGA 会自动从 prom 中加载配置文件。当然配置模式需要设对。

jtag 信号需要

tms – jtag 模式

tdi – 输入

tdo –输出

tck – jtag 时钟

gnd –地

vcc –给 jtag 线缆供电。我用的是 2。5v 供电。不过也有 3。3v 供电的，需加串行电阻吧

配 spartan3 的时候 jtag 线缆很是个问题。最好选择 cable IV。贵，估计要 1300 +

用 cableIII 的话可能会有问题，看你运气了，呵呵。

14、问：PROM 中的配置数据是如何输入进去的呢？也是通过下载电缆吗？是不是先要把 M2,M1,M0 设成 JTAG 模式，通过下载电缆把数据下载到 PROM 中，然后再把 M2,M1,M0 设成 Master Serial 模式，这样每次启动时配置 PROM 就可以自动加载数据了？

答：jtag tdi–––> tdi FPGA tdo–––> tdi prom tdo–––>jtag tdo.

在做原理图的时候 jtag 链需要将 FPGA 和 prom 链起来。

这样 impact 就可以找到在 jtag 链上的 2 个设备。(现在假设只有一个 fpag 和 prom)

接下来你可以选择只用 bit 文件配置 FPGA，或者，事先用 impact 将 bit 文件生成一个 prom 配置的 mcs 文件 (这里可以根据 serial 或者 Parallel 方式生成不同的 prom 文件)，再用这个 mcs 文件配置 prom。一般情况下设成 Master Serial 就可以了。

在用 jtag 配置的时候这个模式好像没什么关系。

现在我的 prom 还没焊上去呢。呵呵，所以不好说。

你觉得不保险，就把 m0，m2 做成跳线。

15、问：现在的 vhdl 语言编程很多是行为模式，请问如何估算程序会消耗的最大资源问题？考虑编程的资源消耗有些什么具体的方程式和经验？逻辑行为描述是否资源消耗很大，如何优化？

答：大部分是先选择同一系列较大的芯片来设计，综合后看报告，然后再根据实际情况移植。对于一个设计，消耗的资源不要超过 80%。还有就是一些 IP Core 生产向导的最后一步也会提示你消耗多少资源。

16、问：FPGA 每次上电时，要从 PROM 中读取配置信息，那么一定需要时钟，该时钟是从哪里来的呢？FPGA 是否存在一个时钟振荡器来提供这一时钟？

答：master 的 FPGA 的 cclk 脚在上电的时候会自动产生配置时钟，这个时钟应该是内部 rc 电路产生的 (个人猜测)，如果是由外部晶振产生的话，那没有晶体的系统 (比如差分接受数据) 不就用不了 FPGA 了？

数据手册中有写：Master Slave 主串 方式下 FPGA 自己产生时钟。不需要外接 RC。

17、问：当 ise 调用 ip 核时，用 synplify 是不是不能综合 ip 核啊？只能用 xst?

答：对于 IP 的综合，因为是加密，所以综合器都是认为是黑盒子。 对于综合器而言，它只用知道端口定义

和名称就行了。所以，你在第三方综合工具，不能看到底层的结构，综合器没有对此综合。在下一步适配和布线，软件才将 IP 解析为硬件源语，形成硬件结构。

在 Matlab 中使用 VHDL 或 Verilog 也是同样的。实际中，我们关心的只是输入什么样的值，就能产生什么样的结果。 也是一种知识产权的保护措施。

在赛灵思设备中，能够直接调用第三方工具。而在 Altera 设备中，需要在第三方工具中，生成网表文件 edf 再到 Quartus 中进行适配布线。设计流程是这样的。

18、问：我做的 ddr sdram 控制器，在软件仿真阶段完全正常，但是到了 FPGA 验证的时候出现了很奇怪的现象：我的 FPGA 用了 4 片 DDR 并行工作（就是当作一片 DDR 一样的发指令）现在遇到的问题是发现在 active 后，连着 3 个 write，然后一个 write with aotoprecharge,burstlength=4，然后再读出来，发现其中有 3 片的数据在读写之后是一致的，然而，另外的一片的这 16 个数没有写进去。该现象与频率无关。然后在紧接着的地址上继续写，读，所有的数据都是正确的。我怀疑 initial 没有做好，加了好多 refresh,precharte，结果都一样，我很奇怪，为什么 3 个片子上的数据都是正确的，只有一个片子的前几位是错误的。

有可能是 initial 没有做好？还是什么原因，各位大侠有没有遇到这种情况？

我用开发板上提供的仿真程序，发现没有这种错误，对比 p&r 后的报告，我的程序和开发板上的时间参数差不多（基本一致）。暂时就能总结这么些现象，请各位大侠们指教一下，就是帮我猜猜也好，谢谢！

答：DDR SDRAM 的设计很关键的是物理层读数据如何捕获的问题。远远不止是 RTL coding 的问题。要做 DEBUG 首先要了解你的器件型号，时钟频率和 DDR SDRAM 器件型号。要关心所有的细节。DDR SDRAM 器件有工作频率下限，你的设计不会频率太低吧。

如果你的设计 DDR SDRAM 频率不高，建议不用自己做了，用 XIILINX 网上公开的参考设计就可以了。至少可以参考 MIG 工具生成的代码，和你自己的代码进行对照，用于参考。

上这个网页 http://www.xilinx.com/products/d ... ce/xaw_dram_ddr.htm

然后注册并下载 Memory Interface Generator (MIG) 工具

有个中文网站也提供资料 http://www.21ic.com/news/n4748c89.aspx，不过不知道版本是不是最新的。

另外，有一个叫做 MPMC 的 High Performance Multi Port Memory Controller (XAPP535) 可以直接拿过来用，连多端口访问仲裁器都带有了。

http://www.xilinx.com/gsrd/

http://www.xilinx.com/bvdocs/appnotes/xapp535.pdf

如果怀疑是实际调试问题而非一般代码设计问题，建议 review 芯片的复位初始化过程，注意写模式寄存器之前需要足够的等待时间。可以从 micron 网站下载一个 芯片模型，例如 http://download.micron.com/downl ... ddr2/512Mb_ddr2.zip，做一个完整的 RTL 仿真，或者利用 MIG 生成的 TESTBENCH，把你自己的代码替换进去做前仿真。

19、问：ADC 的采样速率为 250MHz, 希望用 FPGA 实现如下算法：

Pi=arctan[(Si+1+Si)/(Si+1−Si)],Si 和 Si+1 为相邻的两个采样点 .

计算得到 N 个 Pi 的值后再累加 .

请问如果需要的数据点数为 100 点 , 可以在 1uS 内算出来吗？如果可以 , 用哪个片子比较合适呢？采用什么样的设计比较合适呢？请高手指点 .

答：1，如果是 ADC，Si 和 Si+1 为相邻的两个采样点 . 数字信号只能取“0”或“1”，有 4 种可能的情况，可以用用查表法。

2，楼主是不是要处理模拟信号，应该不是 1 中所描绘的那样。关注多少位的 ADC 啊，可以用 IPcore 中的 cordic 模块,直接求。另外对于这个速度来说,可以选择 VII 以上的芯片,速度可以用 –7 或更高的，只是建议：

1、赛灵思的 IPcore 中的 cordic 模块可以处理三角函数，开根号等，另外有专门的除法模块可以使用，先算除法，然后用 cordic 就可以了。

2、对于 250M 的速度，个人觉得 – 7 的速度可以了，当然，V4 中有更高速度等级的。如果你的后续处理要有 FIR 之类的操作,V4 中有号称可以处理 400M 速率的 FIR 滤波器。不过布线的时候要很注意，接口设计可能有些困难。

20、问：分频应该是个很简单的问题。但是我有些疑问，假如我的时钟是 2MHz，需要 8Kbps 的数据源，那么它们之间的分频是不是 211/23=256 次分频。定义一个 c(7 downto 0) 的计数器就可以了。但是 2MHz 的时钟不是 2048KHz，而是 2000KHz?

答:一般来说,所说的时钟的 K,M 等应该跟数学上的概念一样,而对于计算机来说,仅在表示容量的概念上，K 才是 1024

21、问：用 synplify pro 综合设计，用 ISE 去实现设计，translate 后信号名字都变了，用 modelsim 根本看不到代码中设计的信号名，怎么办？ 有什么办法可以让信号名不变吗？ 多谢了！

答：在 synplify 的 sdc 约束文件中，适用下列语法，保持某些信号名，Net “/PATH/NET_NAME” keep 在 ise 的 ucf 中也要使用相关的约束。

22、问：我用的 FPGA 是 Virtex−2pro 自己做的下载电缆 , 出现的问题是同一个 bit 文件 , 有时候下载成功 , 有时候不成功 , 请教各位大侠是怎么回事啊

答：用示波器量一量 5V 上的纹波，如果太大，自己加一滤波电容在靠近 IC 的附近。

23、问：FPGA 能够实现固定延时吗？比如对某信号分别延时 200ns，400ns，或 2000ns(工作频率 100MHZ)

答：使用同步设计，用寄存器 + 计数器应该可以做到。寄存器保存信号值，计数器计算延迟时间。基准时

钟就是 100M 了，要求时延 (200ns，400ns，或 2000ns) 大于时钟，用不着考虑什么倍频来时延吧。个人认为，资源多的话，就利用计数器 + 块 ram 来控制；资源少的话就用多级寄存器来控制。

24、问：rocketio 如何进行管脚约束？我约束的时候，rocketIO 管脚总是棕色的不让分配，为什么？那位大哥知道告诉我。

答：UCF 文件里定义一下就 OK 了

例子如下

```
NET "xaui_rx_p<0>"LOC = "C26"  ;
NET "xaui_rx_n<0>"LOC = "D26"  ;
NET "xaui_tx_p<0>"LOC = "A24"  ;
NET "xaui_tx_n<0>"LOC = "A25"  ;
NET "xaui_refclk_p"  LOC = "F26"  ;
NET "xaui_refclk_n"  LOC = "G26"  ;
```

或者：约束 mgt 的位置就可以了。

RIO 的几个 pin 和 mgt 一一对应。如：

```
INST mgt_4/MGT4_bottom/PROTOCOL_ENG/AURA/MGT_4 LOC=GT_X1Y0;
```

25、问：spartan2 的 io 口基准电压是 3.3v 的，能否把 5v 的时钟接入 FPGA 的时钟输入端口？有什么不好的影响吗？会不会损坏 FPGA？

答：FPGA 内部有限流二极管，一般不会损坏 FPGA。但是最好还是能加一个电阻来限流。

对于加不加都可以，应该具体问题具体分析，你做得 32。768MHz 属于低频率的时钟信号，所以对时钟波形要求不高，不加是可以，但是对于高频场合，不加还是不行的。所以设计 pcb 的时候，最好还是留个电阻的封装在晶振和 FPGA 之间，如果你不想加电阻，到时加个零欧姆的不就跟没加一样了嘛，总是有利无弊的。

26、问：我现在在用 microblaze+petalinux 构建一个路由器，其中需要加入两个 xps_ethernetlite 来做双网卡。硬件上已经做好了，现在我不知道该如何编译 petalinux 的内核使它能够识别双网卡。请指点！

答：硬件上需要在一条 plb 总线上挂接两个 xps_ethernetlite 和一个 timer。软件上修改驱动程序。修改 petalinux 中网卡驱动的 adapter.c，并建立另一个网卡驱动目录，使之共用编译得到的 ethernetlite.o 等文件。

再修改相关的 Kconfig 和 makefile 文件，并在编译内核时选择同时编译两块网卡的驱动。最终将 image.bin 文件下到板子上，系统会自动识别到两块网卡，再利用 ifconfig 命令分别将 eth0 和 eth1 配置到不同的网段即可。

27、问：为何不能识别下载电缆?

答：其实并口线长一点无所谓，连到 FPGA 上的那头短一点就好了。主要还是电磁兼容性的问题。

28、问：如何让全局时钟不需经过 bufg？我现在不加 bufg 就无法通过 map

答：如果引脚约束到了全局时钟脚上，就一定要用 bufg，否则就通不过。

29、问：请教个问题：我用 ISE 配置管脚信息，比如我设置的简单的模块输入时钟 clk，reset，code，输出 output，我怎么知道 clk，reset，code，output 等接在哪个管脚？谢谢！

答：保险的办法是查阅对应芯片系列的数据文档，可以到 Xilinx 网站 (http://china.xilinx.com/) 相应芯片系列的网页上查询到。文档中说明了每个管脚的类型说明等等。如果是开发板，一般可以查询开发板的说明文档，里面通常写明了用户可用的管脚和用途。一般来说只用注意区分时钟、一般输入输出两类信号的管脚就可以了。

如果使用 PACE 分配不上，可以考虑直接修改 UCF 文件。

ISE 有 UCF 文件配置的 Template，以及可以查阅 xilinx 芯片的使用手册或者 ISE 的帮助手册 (http://china.xilinx.com/support/sw_manuals/xilinx9/) 方便得查询到需要添加约束的类型和添加的方法。

30、问：我想更改一下原来做的一 FPGA 的分配管脚，用的是 ISE5.1，我改了 .ucf 文件后，为什么写到 FPGA 中的还是没有改过来，请问各位高手，改了 .ucf 文件后还要怎么做才改了呀，谢谢各位了。

答：双击 user constraits 里的 create timing constraints, 弹出来的 Xilinx constraints editor 窗口 里面有 Ports，点 ports , 然后在 LOCATION 里输入管脚就行了。当然还有其它办法，如：Assign Package Pins

31、问：偶现在在学赛灵思的 ISE 看到建立 UCF 文件这儿看了好久，还是不知道怎么建立一个 UCF 文件。constraints editor 也是在有 UCF 文件的情况下对它进行编辑的。那位高人点拨一下，怎么建立一个 UCF 文件?

答：新建源文件，类型选择 implement consrainsts，在 source tab 选择所建立的文件，双击 PACE，输入空间位置约束，保存退出，用文本编辑器打开该文件，输入全局时序约束。保存。

32、问：synplify 中如何对 IDELAY 进行约束

```
//
IDELAY U1
(
.O(ddr_dqs_in_d[0]),
.I(ddr0_PQXR_DDRSDRAM_DQS[0]),
```

```
.C(1'b0),
.CE(1'b0),
.INC(1'b0),
.RST(1'b00)
);
//Set IOBDELAY_TYPE attribute to FIXED for Fixed Delay Mode
//synthesis  IOBDELAY_TYPE of U1 is "FIXED";
//Set IOBDELAY_VALUE attribute to 31 for center of delay element
//synthesis  IOBDELAY_VALUE of U1 is 62;
```

以上是 xilinx 对 XST 的约束，

但是在 synplify 中它不认识 //synthesis IOBDELAY_TYPE of U1 is "FIXED";

请问 //synthesis IOBDELAY_TYPE of U1 is "FIXED"; 在 synplify 中是怎么约束的?

答：例化的时候给参数赋值，如下

```
defparam U1.IOBDELAY_TYPE = "FIXED";
defparam U1.IOBDELAY_VALUE = 62;
```

33、问：我的程序里从来没有指定过 wire 类型的变量，对组合逻辑输出变量，直接就 assign 了，是不是如果不指定为 reg 类型，那么就默认为 wire 类型?

比如：

```
module  lddata(clk,rst,cs,din,dout,tmpdata);
input clk, rst,cs;
input din;
output dout,tmpdata;
reg tmpdata;
always@(posedge clk or negedge rst)
if(!rst)
   tmpdata <=0;
else
     tmpdata<= din;
assign dout = tmpdata|cs;
endmodule
```

我的程序都是如以上结构的，没有指定过 wire 类型变量。是不是上面的 clk,rst,cs 还有 dout 都是默认为 wire 的。但我又看到有的程序也专门指定出 wire 类型，有什么讲究吗?

答：是的，不指定就默认为 1 位 wire 类型。专门指定出 wire 类型，可能是多位或为使程序易读。wire 只能被 assign 连续赋值，reg 只能在 initial 和 always 中赋值。

其实是不同的抽象级别，wire 如同 vhdl 中的 signal 类型，是和实际的物理连接对应的，而 reg 属于算法描述层次用的类型，和实际电路没有直接的对应关系，也就是说它相当于 c 语言中的变量 (int，float 等)，vhdl 中的 variable。记住这句就可以了，reg 不和实际的电路如寄存器对应，高层次的描述时用。always 其实算是算法级描述的语句，所以其中的变量必须声明为 reg，还有 initial，自己多看一些例子，会更能加深理解。

一个简单的组合逻辑的例子，用了 reg 类型

```
module mux(a,b,c,sel);
  input a,b,sel;
 output c;reg c;
always @(sel or a or b)
 if(sel ==1'b0) c=a;
 else  c=b;
endmodule;
```

但是这个综合出来就是一个简单的二选一选择器，组合逻辑电路

看它描述的方式，是不是就是把电路的行为 (功能) 描述出来了，这种就是算法级描述。

wire 表示直通，即输入有变化，输出马上无条件地反映 (如与、非门的简单连接)。reg 表示一定要有触发，输出才会反映输入的状态。

wire 实际上就是网线，负责连线，在连续赋值语句中可以用到；而且基本上只能在联续赋值语句中用到。

连续赋值语句如 assign = a&b;

reg 为寄存器类型，但不要被被他名字所迷惑，reg 可以被综合成触发器，锁存器，或者紧紧是连线。

reg 不可以在连续赋值语句中使用。只能在 always 中应用。

触发器 always @ (posedge clk) q<=din;

锁存器 always @ (g,d) if(g) q<=d;

连线 always @(a,b,sel) if(sel) q=a; else q=b;

34、问：如何在 FPGA 内部实现 2 个双口 ram ?

答：按如下程序就可以在 FPGA 内部实现 2 个双口 ram 了，可以实现乒乓操作了

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.all;
entity dualportram is
port
(
    clk : in    std_logic;
    --dout_1: inout std_logic_vector(7 downto 0);
    --dout_2: inout std_logic_vector(7 downto 0)
    dout : inout std_logic_vector(7 downto 0)
);
end dualportram;

architecture action of dualportram is

component lpmramdp_1
PORT
(
    data        : IN STD_LOGIC_VECTOR (15 DOWNTO 0);
    wren        : IN STD_LOGIC  := '1';
    wraddress : IN STD_LOGIC_VECTOR (11 DOWNTO 0);
    rdaddress : IN STD_LOGIC_VECTOR (12 DOWNTO 0);
    clock       : IN STD_LOGIC;
    q           : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
);
end component;
component lpmramdpplus
    PORT
    (
        data            : IN STD_LOGIC_VECTOR (15 DOWNTO 0);
        wren            : IN STD_LOGIC  := '1';
        wraddress       : IN STD_LOGIC_VECTOR (12 DOWNTO 0);
        rdaddress       : IN STD_LOGIC_VECTOR (13 DOWNTO 0);
```

```
        clock           : IN STD_LOGIC ;
        q               : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
    );
end component;
signal wrCount_1 : std_logic_vector (11 DOWNTO 0);
signal rdCount_1 : std_logic_vector (12 DOWNTO 0);
signal dataIn_1  : std_logic_vector (15 downto 0);
signal dataOut_1 : std_logic_vector (7 downto 0);
signal wrCount_2 : std_logic_vector (12 DOWNTO 0);
signal rdCount_2 : std_logic_vector (13 DOWNTO 0);
signal dataIn_2  : std_logic_vector (15 downto 0);
signal dataOut_2 : std_logic_vector (7 downto 0);
signal flag:std_logic;
begin
process(clk)
begin
  if rising_edge(clk) then
    wrCount_1 <= wrCount_1 + 1;
    rdCount_1 <= rdCount_1 + 1;
    wrCount_2 <= wrCount_2 + 1;
    rdCount_2 <= rdCount_2 + 1;
    flag <= not flag;
    if (flag = '1') then
      dout <= dataOut_1;
    else
      dout <= dataOut_2;
    end if;
  end if;
end process;

u1: lpmramdp_1
port map
(
```

```
        data            => dataIn_1,
        wren            => '1',
        wraddress       => wrCount_1,
        rdaddress       => rdCount_1,
        clock           => clk,
        --q              => dout_1
        q         => dataOut_1
    );
    u2: lpmramdpplus
        PORT map
        (
      data            => dataIn_2,
        wren            => '1',
        wraddress       => wrCount_2,
        rdaddress       => rdCount_2,
        clock           => clk,
        --q              => dout_2
        q         => dataOut_2
        );
    end;
```

35、问：最近看了一个经验丰富的大哥写的 FPGA 设计技巧，其中有这样一段话“禁止用计数器分频后的信号做其它模块的时钟，而要用改成时钟使能的方式，否则这种时钟满天飞的方式对设计的可靠性极为不利，也大大增加了静态时序分析的复杂性。”对这段话我很是不明白，是不是这样的话会使扇出加大？另外我想知道，他所说的 " 改成时钟使能的方式 " 的具体方法是怎样的？大家能给出个具体例子吗？

答：这种说法不正确，在某些情况下，我们是需要将计数器的输出作为涉及内部的时钟的。例如在分频时钟的目标寄存器数量巨大，并且对面积要求很紧时。我们往往采用直接将分频信号作为时钟的方式，而不是把其作为后续寄存器的使能信号。总之，要实际问题实际分析，只要能够实现设计并且满足 spec 的方法都是可以考虑采用的。

在高速电路中使用分频后的时钟，它与原时钟相比有抖动，这样在对同步要求比较高的电路中会引起一些异常错误，虽然你的设计可能是对的。比如 clk0 是 clk 分频后的时钟，使用 clk0 你产生了信号 a，clk 产生了信号 b，按照设计信号 a 和信号 b 是在同一个时钟沿产生的，但在实际电路中因为两个时钟之间的抖动，信号 a 常常超前或者落后信号 b 而且每次都是不确定的，这样将 a 和 b 进行运算时它们之间的结果也是不确定的。所以在高速电路中经常把分频后的时钟作为使能，如果作为使能信号这里

clk0 必须只有一个 clk 主钟宽度

```
process(clk)
if CLK'event and CLK = '1';
  if clk0=1 then
     b<=__;
endif;
endif;
end process;
process(clk)
if CLK'event and CLK = '1';
     a<=__;
endif;
end process;
```

需要确认两点：

pin assignment 的时候是否指定的是合适的 pin, FPGA 不是每个 pin 都适合做 clock 输出。

FPGA 并不知道你的该信号是时钟，它认为是一般的信号。因此你在输出该信号时最好使用 GBUF 或者 CLKBUF 过一下。

36、问：关于上升沿和下降沿触发的讨论

答：发现一些同仁提出上升沿和下降沿计数的问题，工作中也碰到一些同事问及此问题。现在我把我多年来一直采用的办法奉上，但愿对初学者有所帮助。

以一个最简单的计数器为例：Port(

```
clock:in std_logic;
pulse:in std_logic;
q:out std_logic_vector(3 downto 0)
);
```

q 输出为对 pulse 跳变沿的递增计数。clock 为系统高速时钟。

```
Process(clock) begin
if rising_edge(clock) then
dly1pul <= pulse;
dly2pul <= dly1pul;
end if;
End process;
```

```
en <= dly1pul and not dly2pul;-- 上升沿
--en <= not dly1pul and dly2pul;-- 下降沿
--en <= dly1pul xor dly2pul;-- 上升沿和下降沿
Process(clock) begin
if rising_edge(clock) then
if en = '1' then
cnt <= cnt + 1;
end if;
end if;
End process;
q <= cnt;
```

单对于此小问题，当然采用倍频实现双沿计数也是可行的，但是我们不要忘记，倍频器在很多 CPLD 或 FPGA 中是不支持的，即便支持其资源也是很宝贵的。

我看到的一些设计中，动辄采用某一信号作为时钟，应该说这种做法是欠妥的。因为不是全局时钟的时钟信号最大扇出是有限的，其很难保证时钟延时应小于信号延时的基本要求。当遇到要对某个信号的跳变沿处理时，建议采用上述小例子中 en 信号的处理办法。

37、问：一个值得去探究的问题，如何用硬件描述语言把十进制形式的数据的各个位数提取出来？这是 C 语言的译码方式，相信大家都会经常用到，把数据变为十进制的形式显示，提取出各位数保存到各个内存，这种方法通用简单。

```
.....
  unsigned char fre[11];       // 频率显示寄存器
  fre[10]=frequency_measure/1000000000;
  fre[9]=(frequency_measure%1000000000)/100000000;
  fre[8]=(frequency_measure%100000000)/10000000;
  fre[7]=(frequency_measure%10000000)/1000000;
  fre[6]=(frequency_measure%1000000)/100000;
  fre[5]=(frequency_measure%100000)/10000;
  fre[4]=(frequency_measure%10000)/1000;
  fre[3]=(frequency_measure%1000)/100;
  fre[2]=(frequency_measure%100)/10;
  // 频率小数部分
  fre[1]= frequency_measure%10;       // 十分位
```

fre[0]= frequency_measure%1; // 百份位

…

而用硬件描述语言用什么办法可以办到把十进制形式的数据的各个位数提取出来，达来以上的效果，这样就可以不用通过微处理器那么麻烦用 C 语言提取数，直接译码在数码管上显示，本人认为这个值得大家参详一下。

答：这个代码实现了 64 位 bin 转 20 位十进制 . 没有实现参数调整位宽，一共用了 62 个 clock cycle，154 个 cell，考虑到 80bit 的输出 , 这样的资源消耗算少的。

已仿真成功，供参考。

```
`timescale 1ns/1ps
module bin2bcd(clk,start,in,out,done);
parameter BIN_WIDTH = 64;
parameter BCD_WIDTH = 80; //(Log(2^BIN_WIDTH)+1)*4
parameter CNT_WIDTH = 6;
input clk,start;
input [BIN_WIDTH-1:0] in;
output [BCD_WIDTH-1:0] out;
output done;
reg [BCD_WIDTH-1:0] out;
reg [CNT_WIDTH-1:0] count;
reg done;
always @(posedge clk)
begin
if (start)
begin
 out<=in[BIN_WIDTH-1:BIN_WIDTH-2];
 count<=BIN_WIDTH-2;
 done<=0;
end
else begin
 if (count>1)
 begin
  count<=count-1;
  if ({out[2:0],in[count-1]}>4) out[3:0]<={out[2:0],in[count-1]}+3; else out[3:0]<={out[2:0],in[count-1
```

```
]};
  if (out[6:3]>4)   out[7:4]  <=out[6:3]+3  ; else out[7:4]  <=out[6:3];
  if (out[10:7]>4)  out[11:8] <=out[10:7]+3 ; else out[11:8] <=out[10:7];
  if (out[14:11]>4) out[15:12]<=out[14:11]+3; else out[15:12]<=out[14:11];
  if (out[18:15]>4) out[19:16]<=out[18:15]+3; else out[19:16]<=out[18:15];
  if (out[22:19]>4) out[23:20]<=out[22:19]+3; else out[23:20]<=out[22:19];
  if (out[26:23]>4) out[27:24]<=out[26:23]+3; else out[27:24]<=out[26:23];
  if (out[30:27]>4) out[31:28]<=out[30:27]+3; else out[31:28]<=out[30:27];
  if (out[34:31]>4) out[35:32]<=out[34:31]+3; else out[35:32]<=out[34:31];
  if (out[38:35]>4) out[39:36]<=out[38:35]+3; else out[39:36]<=out[38:35];
  if (out[42:39]>4) out[43:40]<=out[42:39]+3; else out[43:40]<=out[42:39];
  if (out[46:43]>4) out[47:44]<=out[46:43]+3; else out[47:44]<=out[46:43];
  if (out[50:47]>4) out[51:48]<=out[50:47]+3; else out[51:48]<=out[50:47];
  if (out[54:51]>4) out[55:52]<=out[54:51]+3; else out[55:52]<=out[54:51];
  if (out[58:55]>4) out[59:56]<=out[58:55]+3; else out[59:56]<=out[58:55];
  if (out[62:59]>4) out[63:60]<=out[62:59]+3; else out[63:60]<=out[62:59];
  if (out[66:63]>4) out[67:64]<=out[66:63]+3; else out[67:64]<=out[66:63];
  if (out[70:67]>4) out[71:68]<=out[70:67]+3; else out[71:68]<=out[70:67];
  if (out[74:71]>4) out[75:72]<=out[74:71]+3; else out[75:72]<=out[74:71];
  if (out[78:75]>4) out[79:76]<=out[78:75]+3; else out[79:76]<=out[78:75];
 end
 else if (count==1)
 begin
  out={out[78:0],in[0]};
  done<=1;
  count<=0;
 end
end
end
endmodule
```

38、问：在 EDK 的 Assembly 界面中和 mhs 语法中，一般都是把整个 bus 整体连接到另一个 bus 的。如果说把 busA[1] 赋值到 busB[0]，这样的写法是不行的。如果要从一个 bus 里抽出一根 net，该怎么办呢?

答：暂时想出来的解决方法：

方法 1：用网线连接符号 "&"。这种方法是把原来的想法反过来——本来是要分割 bus，现在是连接 bus。也就是说，给分割后的网线取名，比如 busA_0, busA_1。到整条 bus 的地方，就叫 busA_0 & busA_1。

方法 2：用 IP Catalog 里的 Utility –-> Utility Bus Split 来分割 Bus。

问题是这个 IP 只能把一个 Bus 分割为两部分，如果要去除一个 Bus

方法 3：自己写一个 IP 吧！

39、问：用什么查看 EDIF 网表？

答：1. GateVision

2. Synplify_pro

3. Xilinx PlanAhead

40、问：FPGA Editor 可以做些什么？

答：1、检查布局布线结果——检查 DCM，BRAM，Slice，IOB 等等的配置方式；查看布线方式，是不是走了全局布线通道等。

2、修改 DCM, BRAM, Slice, IOB 等的配置方式，比如修改 DCM 倍频系数，Pin 的输出电压标准等。

3、与 Timing Analyzer 配合实现 CrossProbing，查看 Timing 瓶颈。

4、添加 Probe，将内部信号引到 Pin 上以方便示波器观察 (可直接生成 bit 文件)。

5、更改 ChipScope 的 ILA Core 的一些配置，比如说改变 ILA 采样时钟。

6、更改布局，比如说，换一个 Pin 来输出信号。

7、改变布线。

8、Direct Routing，将所有的布局布线信息都记录下来。

41、问：关于 ISE 版本使用策略，请提些建议？

答：对于纯 ISE 用户：

1、新的工程尽量用最新的 ISE 版本

2、已有的设计，如果正在使用的版本没有暴露什么缺陷，就不要更新到新的 ISE 了，因为更新有时候会带来问题。

3、如果使用的 ISE 不是当前最新版本，那么请升级到这个主要版本的最高版本并打上所有的 Service Pack 和 IP Update。所有的升级补丁可以在 www.xilinx.com/download 找到。

对于 EDK、ChipScope Pro、System Generator 用户：

1、已有的设计，如果正在使用的版本没有暴露什么缺陷，就不要更新到新的 ISE 了，因为更新有时候

会带来问题。

2、在 10.1 版本以前，由于 EDK, ChipScipe Pro, System Generator 的发布都会晚于同版本的 ISE 一到两个月，而且这些软件的使用都需要严格与 ISE 的版本匹配，比如 ISE8.2 不能和 EDK8.1 混用，所以，使用这些软件不必急着使用新版本的 ISE。

3、10.1 以后，ISE、EDK、ChipScope 等软件和补丁都是同时放出的，所以可以立即更新到最新的补丁版本。

注 1：Xilinx 软件由于使用 Java，所以不要将它们安装在带空格的路径下！给 SysGen 宿主的 Matlab 也一定要注意这一点。

注 2：升级 ISE 会带来 Map/Par 算法的更新，通常这些更新会提高整个设计的 Performance，但是 Performance 降低 (比如占用面积增大) 也是有可能的。

注 3：最新的软件 Update 都可以在 www.xilinx.com/download 找到。

42、问：时序约束是综合器要求的基本参数，可有点不明白。

以 Synplify 为例，在 ISE 环境下很容易增加一个频率约束，比如 Max Freq.=100M.

有了这个综合参数，似乎自动就意味着 Period 约束为 10ns, 这样，还需要单独增加 Peroiod=10 这个约束吗？请高手赐教。

答：强烈建议

1. 综合时期的约束，以及布线时期的约束 都要分别加上。

2. 而且在为了获得更好的布线结果，对于系统时钟和关键模块的局部综合，综合时期的约束要比布线时期的约束要略高一些。

我是 3 年前确定这个规则的，后来听赛灵思的讲座，发现赛灵思也是这样建议的，特别是第一点。

43、问：一片 FPGA 内使用多个 DCM，需要注意的事项？

答：一片 FPGA 内使用多个 DCM，时钟从一个 clk 输入，走到两个 DCM 的 clk_in，然后让 DCM 操作。这时，需要注意：

1、用 CoreGen 生成 DCM 模块的时候，clk_in source 是 internal，不要他直接连接 pin，加 buffer。

2、手动例化一个 IBUFG，然后把 IBUFG 的输出连接到两个 DCM 的 clk_in

通常，如果没有设置 clk_in source 是 internal，完全按照使用一个 DCM 的流程，就会造成 clk_in multiple driver。

如果还想让这两个 DCM 输出的信号相位对齐，这个 ISE 是不能自动做到的。FPGA 只能做到一个 DCM 的输出是相位对齐的。而时钟 pin 到两个 DCM 的路径和 DCM 输出的路径都有不同的延时，因此对相位还有要求可能就要自己手动调整 DCM 的位置了。

44、问：我现在在 FPGA 下载了一个单片机的核，我编写了一段单片机的程序，希望将编译后的程序直接下载到 FPGA 中的单片机中去执行，请问各位知道该如何将编译的 16 进制代码下载（相当于单片机的烧录）到 FPGA 中吗？谢谢。

注：用过单片机的都知道编译生成的是 16 进制的代码，如下所示是一个编译的 16 进制文件：

:020000040000FA

:02000000100AE4

:10002000000C0600010C260066031709140A0F0CC9

:080030003000F002190A00087B

:00000001FF

本单片机的指令码是 12 位的，所以如果仅仅从表面上看将其直接下载到 FPGA 中的配置 RAM 或 ROM 中是不可行的（我个人以为是不行的，因为其个数不是 12 的整数倍）。所以各位知道该如何处理吗？另外，如何将文件写入到 FPGA 的 RAM 块中？

答：首先单片机的代码应该是下载到单片机的程序空间内，这是一块 rom，这个单片机 ip 应该有程序 rom 的接口，你只需要在 FPGA 内部实现一个 rom（可以是你要的 12bit 位宽），然后在生成 rom 的时候把初始化 rom 的文件设置为你的单片机的 16 进制程序文件。这样 FPGA 编译下来你的程序文件就编译到 FPGA 的配置文件中了。仅供参考。

45、问：clk 采外部输入的一组 data(10bit)，如何添加约束，使 10 条线的 setup time 一样？请帮忙简单写下采用哪种约束，语法。

答：使 10 条线的 setup time 一样是几乎无法实现的。因为 FPGA 内部走线的长度不可能完全一样。但是可以加约束使 10 根线的 setup/hold time 都满足。

这时你需要告诉 ISE 你的外部情况时什么样子的，这样他就根据你的外部条件来走线尽量满足你的约束。在 ISE 里，应该用：offset 来约束，具体语法和含义，可以看他的文档，或是去他的网站找。

46、问：如何在 FPGA 中实现大于 2ns 小于 5ns 的延时呢，试图用非门，可是综合的时候总是优化掉了，如何通过设置约束保存非门呢，最好是原语。

答：延迟的设计其实是比较复杂的。如果想延迟 5ns 实际上在 FPGA 中是比较难实现的。

简单的延迟可以用逻辑门加上 KEEP attribute 或 syn_keep attribute 来实现的，可以保证你的延迟逻辑不被优化掉。但是现在 FPGA 中的延迟：逻辑门延迟只是小部分，更过的是布线延迟，这个布线延迟每次 P&R 后是不尽相同的。所以用这种方法是很难得到一个稳定的延迟的。

（如果你想把这个延时的精度控制在 1ns 就需要下面的内容了）

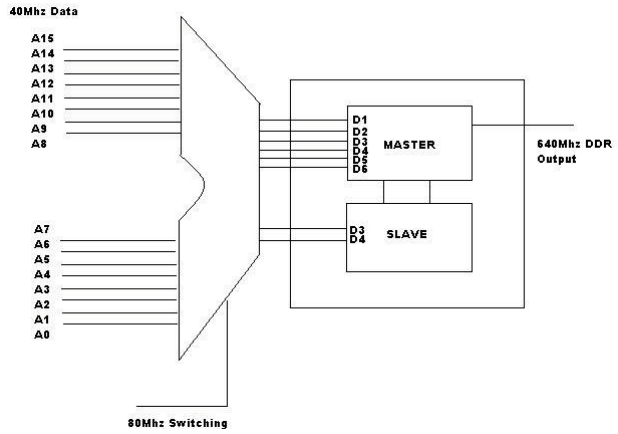
稍好的办法是逻辑门 +keep 属性 + 分组延迟约束，这种方法需要你对约束编辑比较了解。

还有更好的办法就是用逻辑门 +keep 属性 +planAhead 工具（或者 FPGA editor），对底层的 FPGA 布

线及 CLB/LUT 的布局位置进行约束。也可以看看 mig2.0 生成的 dqs 延迟的约束，然后再参考使用 planAhead 工具 (或 FPGA editor)。

这是一个艰苦的过程，把这个搞明白了，FPGA 约束、接口、时序的很多东西也就基本通了。会了这个你也就有能力解决很多高速接口的问题了。

47、问：我正在尝试用 OSERDES 实现 16：1 串行化，但是我发现在使用 DDR 主从模式时，我能达到的最大串行化是 10：1，我如何用现有的 OSERDES 实现 16：1？如图所示



答：做如下检查：

1、OSERDES 输入寄存器不用使能端，它被设置为一直使能；

2、OSERDES 的使能端连接到串行器的最后寄存器上，通过使能禁止输出但其他信息可以继续通过；

3、数据在 CLKDIV 的上升沿装载进 OSERDES；

4、在给定的 DATA_WIDTH 的比特数目达到后，数据从并行寄存器传输到并串转换寄存器；

5、所以有这样可以在所有数据在移出之前把并行数据置入 OSERDES；

6、在你的案例中，用 CLKDIV/2 时钟进行数据 MUX；

48、问：我正用 PROM XCF32p 配置 virtex 5，我想知道是否有可能把信息从 DATA(0:7) 输入而不是从 JTAG 存储到 PROM，而且用存储在 PROM 里的新信息重新配置 Virtex？如果不行的话，是否有可能模拟出 JTAG 的时序来将数据写入 PROM 而不是通过 PC 写入信息?

答：恐怕数据不能直接从 D[0:7] 写入 PROM，因为他们只能被用作输出管脚。但是我们可以来模拟得到 JTAG 的信号控制 PROM。这里有一些不同的方法：

http://www.xilinx.com/support/documentation/application_notes/xapp975.pdf

(Low–Profile In–System Programming Using XCF32P Platform Flash PROMs)

http://www.xilinx.com/support/documentation/application_notes/xapp424.pdf

(Embedded JTAG ACE Player)

http://www.xilinx.com/support/documentation/application_notes/xapp058.pdf

(Xilinx In–System Programming Using an Embedded Microcontroller)

XAPP058 参考外部 uC, 但有可能通过一个系统总线（例如 FPGA 中的 PCI 插槽）用一个 MicroBlaze 实现。

49、问：请教如何将引脚的 BUFG 属性改成 IBUF，前面看了很多朋友的关于全局时钟设定的问题。但是具体的在哪个地方设置引脚的 BUFG 属性一直没有找到，请教各位了。

答：实例化一个 BUFG，例如：

BUFG drive_clk(.I(clk_in),.O(clk_bufg));

clk_in 是时钟输入，clk_bufg 就可以作为你说的全局时钟了。

输入的引脚先例化一个 IBUFG 再走 BUFG 即全局时钟网络；

输出的引脚直接由 BUFG 输出就可以了；

如果 DCM 资源不紧张的话，输入时钟最后经过 DCM 之后再使用，这样可以提高时钟质量，降低时钟 skew。

50、问:请教如何在 FPGA 中验证 DDR2 控制器，用 MIG 生成的控制器，修改了管脚和顶层的 HDL，编译通过。想下到 FPGA 中验证一下。有没有做过 DDR2 的？你们是怎么在 FPGA 中验证的？

答：MIG 生成的 DDR2 控制器是 3 个 FIFO 接口：写数据、读数据、地址。

生成时可选择带 testbench 的，那个可以下到 FPGA 里边，大体思路是写入再读回，在读回数据时本地再产生一个跟预期读回数据相同的数据，然后比较，如果不相同，error 信号有显示。可以修改 testbench 的逻辑，对存储器做遍历性的写入、读回测试。

51、问：什么是差分时钟？

答：常规的时钟线都是一根，就是晶振的输出脚，但是时钟在长路径传输时会畸变，导致最终坏掉。为了能够长路径传递时钟并且保持时钟波形、偏移等性能，采用了差分对时钟。

差分对只是信号的驱动形式，采用差分驱动，就如以太网 TX+、tx–，还有 RS422 的 ABYZ 等。

52、问：如何在 ISE 里面实现对数运算了？请教各位，在不使用查表法的情况下，如何用算法在 ISE 内实现对数运算了？ ISE 里面没有 IPCORE 可以用，有没有人做过这方面的东西？请教一下算法的公式和大概流程，多谢先！

答：可以用 ISE 里的 CORDIC 算法 IPcore，算出 arctanh，然后乘一个固定系数就可以了

53、问：请教一个关于 FPGA 芯片引脚的问题

产品型号是 Virtex II pro xc2vp40 FG676。现在发现引脚 U12,U13 和引脚 A1 短路，但是引脚

U12,U13 为 bank5 的 VCCO，外面接的是直流 3.3V，引脚 A1 为 GND，

接的是数字地。同样引脚 U14,U15 为 bank4 的 VCCO，接的是直流 3.3V，它们也和引脚 A1(GND，接的是数字地) 短路。由于我们买了两片这个型号的芯片，所以两片的引脚都相同的测了一下，发现两片芯片的状况一样。所以我觉得芯片本身可能没有问题。但是这又很让人难以理解，而且如果一加电的话，那么直流 3.3V 电源就会和数字地 (GND) 直接短路，所以我们现在迫切想得到哪位大虾的答复。

答：不知你是怎么量的短路。

1、直接量未焊接的 LSI 很容易损坏，最好焊上了再量。

2、假如是已经焊上发现的短路，没准是

2–1、电路图设计错了 (引脚号定义错)

2–2、PCB 设计错了 (量空板看看)

2–3、焊接问题，搭锡 (可能性较小，两块板都搭锡？)

3、片内有嵌位二极管，正好被你量了个正向导通？ (可能性太小)

54、问：请问时序分析中如何划分关键路径与非关键路径呢？如何定义他们？

答：关键路径，是你设计中延时的最长路经，综合器指示作为一个指示告诉你，如果你觉得该延时可以容忍是不用考虑的，比如设计频率为 50MHZ，而关键路径有 21ns 的延时，故该电路将无法满足设计的频率要求，就需要对关键路径中的各级延时进行分析，通过电路级和系统级的优化减小关键路径的延时问题，如果对于你的系统只有 1MHZ 的需求那么 21ns 的延时相对而言有较为充分的电路建立、保持时间，故该关键路径可以忽略！！

从香农那可以了解数字信号在通过模拟网络的情况，这里讨论的是数字信号在通过数字网络的情况，模拟网络表现在 FPGA 中就是布线延时，布线串扰，布线阻抗等等，这些东西是 FPGA 芯片厂商要解决的东西。数字网络就是我们自己实现的组合网络，时序逻辑。

FPGA 设计软件 timing 报告中给出的数值应该是前者和后者两者的组合，不过前者仿真太难了，估计给出的也是个估值，不过却 " 无伤大雅 ",FPGA 布线是自动的，其信号完整性我们不可以干预，由厂家保证。

而我们可以做的，就是要保证后者。即大家通常所说的时序分析，通过组合网络，时序逻辑的数字信号情况。当然，实际模拟网络和数字网络并没有明确的分界线。数字网络中我们只关心 period,setup,hold 这些参数，至于方波是不是真的那么方，由于不是主要矛盾，就忽略了。

55、问：我在使用综合带有 XILINX rom(名称：rom_test) 核模块的时候，出现了一个警告：@W: | Creating black box for empty module rom_test

IP 核是一个黑箱到是正确的，但是为什么会提示这个核是一个空模块呢？ 高手帮忙看看啊，谢谢啦

答：这个只是工具告诉你有这么一个 black box，工具这样做是做是为了给你足够的信息，让你去判断正误只要你有 rom_tets.edn/rom_test.ngc 等 IP 就不用管这个了。

56、问：我在设计中用到赛灵思的 DLL(sptant2 系列)。

但是 DLL 的时钟输入频率较低(几百 khz)，正常情况下，DLL 可以正常输出，但有时候运行一会儿，DLL 无输出，一直为低电平。(程序还在 FPGA 中，没有跑飞(已经验证))。请问，是不是低频情况下不能用 DLL，或者是其他原因那?

答：在低频时相对性周期抖动 (Cycle–to–Cycle Jitter) 必须在 300ps 以内，所以我认为你应该检查下你的输入时钟。

57、问：logic delay 和 route delay 到底是什么啊? logic delay 与 route delay 是在综合报告中看吗?？两者有什么差别呢?

答：FPGA 的综合与布线实际就是通过开关矩阵连通里面的 component，就是 CLB，或者说 LUT、触发器等构成的单元。逻辑延迟指的是 component 自身的延迟,例如经过了非门,那么可能有几个 ns 的延迟，经过了 LUT，也会有若干个 ns。布线延迟则指 component 之间连线导致的时间延迟。FPGA 里面的开关矩阵，长线，短线等都会导致时间延迟。

58、问：一个 bit 文件竟然可以分开放在 3 个 rom 中去配置 FPGA 吗?

答：当然可以了，你把他串到一块就可以，以前小的那种我是用 3 个 rom，现在也有大的，但是体积也大，现在基本用一个大的。

59、问：我想调用赛灵思，IP 核中的 RAM，但是发现 RAM 的读和写的使能信号不能分开使用，它只有两个控制信号 WEA(写使能)，ENA(写，读，复位的使能信号)，WEA 和 ENA 都为高时，写出来的东西马上就读出来了，WEA 为低，ENA 为高时就单纯的通过地址线读，此时写无效。我想实现的功能是读和写我都能够控制，想在 RAM 写满后，边读出来一个边写一个。我应该怎么办?

答：你需要控制读写的地址线，读和写的时候分别将地址线加一，就可以得到特定地址的数据了。

60、问：一直推崇同步设计，请问异步电路的好处在哪? 不可能没有好处吧?

比如异步计数器，速度又慢，我就不知道怎么还有存在的必要呢?

异步 reset 呢? 是不是没一点好处? 异步电路究竟在什么地方有他发挥之处呢?

答：这个问题问的很好，很有深度，我个人觉得异步设计的一个好处就是结构比较简单，至少耗费的资源要比同步设计来得少。

1、异步设计比同步设计面积小、功耗低、速度快；

2、同步设计中，许多地方需要使用异步方法，比如，大多数外部接口电路；

3、为什么设计中多用同步设计? 因为没有好的 tool，不能很好地达到可控、可测、可重用的目的。

61、问：我想在后方的时候看到内部模块的信号，所以用了 keep hierachy，就是在 ISE 的综合的属性里设置的，但发现只能看到顶层例化模块的端口信号，其再进一层的信号就看不到了？有没有办法进一步看顶层文件例化的 module 的内部的信号？我看了 XST 的文档，上面说用
(* keep_hierarchy = "{yes}"*)(用在 verilog 中)，但不知道具体位置放在什么地方。试了几个地方都不行！谢谢！

答：keep hhierarchy 不行。你该针对某个信号，直接使用 keep 属性。

你看 cgd 文档里，直接搜 keep 关键字，有详细说明。

62、问：我想用 Sparten 3 做 LVDS 输出，但不知道他和 90lv032 能否对接，传输距离多远，在 ise 中如何实现呢？就这一问题，还想请教一下，在 Viretex-4 里怎么做？ LVDS 的传输模式有好几种，有直接并行的，有直接串行的，也有串化解串的。现在想问的是在“串化解串”的模式下，能传输多远？电路原理图怎么画，有没有这方面的应用实例？

答：LVDS 本身只是一个电平标准，有细分对应几个摆幅的子标准，在具体应用中要考虑发送方与接收方之间的耦合方式。通常有直流耦合和交流耦合两种，直流耦合需要核算更多的技术细节，交流耦合的设计相对容易一些。详细信息可以参考 TI 和 MAXIAM 的相关应用笔记，了解充分的背景知识有助与你看懂 Virtex-4 的数据手册。这是第一步。

接下来，考虑传输距离。影响传输举例的主要有几方面：驱动端输出摆幅，数据通路上的衰减情况，接收端的灵敏度。其他方面还有阻抗匹配的问题还有串扰的问题。 从数据手册上，你可以了解到 FPGA 器件的 输出摆幅 和 接收端的灵敏度。同时也要了解对端器件的 输出摆幅 和 接收端的灵敏度。接着计算你的传输介质的通路衰减，FR4 材料的 PCB,roger 材料的 PCB, 以及不同直径的同轴电缆或者双绞线都有不同的衰减率。查出传输介质的通路衰减，大致就可以知道传输距离。 举个例子，在两端 Virtex-II 的的情形，细同轴电缆，SDR，150Mbps，我的设计大约能传 15 米多，如果电缆质量好，还可以传递更远，但是 <50M。

传输的距离远近直接影响物理层信号质量。在数据通路设计方面，还要考虑链路层的传送方式，例如 17 路 LVDS 并行总线用源同步方式，单路 LVDS 并行总线带编码时钟，或者单路 LVDS 并行总线不带编码时钟。 不同的链路层有不同的内部处理方式。或者说，即使物理层上单个 LVDS 差分对的传输距离是有保证的，也需要在链路层上对对应的设计，以便保证在链路层级别上单通道或多通道组合后是正确的。

Serialize /Deserialize 模式下，传输距离和直接串行模式是完全一样的。关于 Serialize /Deserialize 模式讲起来话就长了。如果我当年的设计在 VIRTEX-4 上，用 Serialize /Deserialize 模式就太简单了，速度也可以做得更高。技术日新月易啊。

关于 PCB 设计，简单，找一个来抄就好了。我是不偏好完全自己做的，抄抄改改就好了，留出更多的时间做自己业务最关键的部分。

ML450 开发板，我借用过几天，评估 350M DDR LVDS 设计。可以对你有帮助。

Virtex–4 ML450 Source Synchronous Interfaces Tool Kit (HW–V4–ML450–USA)

Xilinx Development Boards

http://www.xilinx–china.com/xlnx ... key=HW–V4–ML450–USA

ML405 PCB

http://www.xilinx–china.com/prod ... iles/ml450_pcba.pdf

ML405 PCB GERBER 文件

http://www.xilinx–china.com/prod ... 50_gerbersource.ZIP

原理图

http://www.xilinx–china.com/prod ... schematicsource.ZIP

参考设计

http://www.xilinx–china.com/prod ... ference_designs.htm

16 Channel 4:1 DDR LVDS

Application Note: XAPP700

XAPP704: High Speed Single Data Rate LVDS Transceiver

16 Channel 4:1 SDR LVDS

Application Note: XAPP704

XAPP705: High Speed Dual Data Rate LVDS Transceiver

16 Channel 4:1 DDR LVDS

Application Note: XAPP705

如果你的设计潜在芯片采购量价值比较高，比如说单个设计芯片采购量超过 100 万人民币一年。就直接联系代理商，找赛灵思原厂的专家直接支持。

63、问：LUT 是实现组合逻辑的 SRAM, 怎样实现一个时序的移位寄存器，是不是必须加一个触发器来配合 LUT？

答：Xilinx Virtex 结构中的 LUT 不是简单的组合逻辑。当它被配置为 16x1 RAM 时，写操作是同步的。当它被配置为移位寄存器时，则无需消耗任何 flip–flop 资源。事实上 Virtex LUT 的内部电路比看起来更复杂。

64、问：请问哪位前辈知道，在 ISE 中如何使用 Synplify 的综合结果?

答：两种方法，一种，直接 property 把综合工具设置成 synplify，这是 batch mode 调用 synplify。 还一种，单独用 synplify 综合后，ise 建工程的时候选择顶层为 edif 文件。

65、问：求助贴：ise 和 edk 混合使用碰到了问题在 EDK 中设置 project options 中的 Implement Design in ISE

把 edk 中的设计导出到 ise 中，这个没有什么问题。我碰到的问题是：导出完成后，把 project option 中的选项改过来（仍改为：Implement Design in XPS, 像 edk 默认的那样），然后再依次点 generator netlist, 这一步也没有问题，但是当 generator bitstream 时却出现这样的错误提示：

Applying constraints in "system.ucf" to the design...

INFO:NgdBuild:757 – "system.ucf" Line 144: The constraint for NET 'fpga_0_SysACE_CompactFlash_SysACE_MPIRQ_pin' is being attached to the equivalent NET 'SysACE_CompactFlash_SysACE_IRQ'.

INFO:NgdBuild:757 – "system.ucf" Line 145: The constraint for NET 'fpga_0_SysACE_CompactFlash_SysACE_MPIRQ_pin' is being attached to the equivalent NET 'SysACE_CompactFlash_SysACE_IRQ'.

ERROR:NgdBuild:755 – "system.ucf" Line 362: Could not find net(s) 'fpga_0_net_gnd_pin' in the design. To suppress this error specify the correct net name or remove the constraint. The 'Allow Unmatched LOC Constraints' ISE property can also be set (–aul switch for command line users).

ERROR:NgdBuild:755 – "system.ucf" Line 363: Could not find net(s) 'fpga_0_net_gnd_pin' in the design. To suppress this error specify the correct net name or remove the constraint. The 'Allow Unmatched LOC Constraints' ISE property can also be set (–aul switch for command line users).

ERROR:NgdBuild:755 – "system.ucf" Line 364: Could not find net(s) 'fpga_0_net_gnd_pin' in the design. To suppress this error specify the correct net name or remove the constraint. The 'Allow Unmatched LOC Constraints' ISE property can also be set (–aul switch for command line users).

ERROR:NgdBuild:755 – "system.ucf" Line 365: Could not find net(s) 'fpga_0_net_gnd_pin' in the design. To suppress this error specify the correct net name or remove the constraint. The 'Allow Unmatched LOC Constraints' ISE property can also be set (–aul switch for command line users).

ERROR:NgdBuild:755 – "system.ucf" Line 370: Could not find net(s) 'fpga_0_net_gnd_2_pin' in the design. To suppress this error specify the correct net name or remove the constraint. The 'Allow Unmatched LOC

Constraints' ISE property can also be set (-aul switch for command line users).

… …

… …

问题看起来像 design 里没有 'fpga_0_net_gnd_pin' 引脚 , 但是我查看了 mhs 文件和 ucf 文件 , 貌似没有什么问题 . 令我十分不解。请大侠帮忙看一下 , 有没有碰到这个问题的同学给些建议。

答：EDK 中的工程，导出到 ISE 中以后，会是一个子模块，而对应的引脚有些在 ISE 工程的顶层里面没有连接，需要修改约束文件,或者将这些引脚引到顶层端口。导出到 ISE 的时候,UCF 就被改了,回到 XPS 的时候，UCF 没有改回来 。另外，做子模块时需要在 ISE 中例化，而且必须在 ISE 中管教名字与 XPS 中一致。

66、问：前仿真正确，综合，布局布线均通过，加的时钟约束都满足。但用 modesim 进行后仿真时没有信号输出 (在用 320M 的时钟采样时，寄存器的输出有信号，输出就没有信号)，这是什么原因啊？怎么解决这个问题啊！ 请达人指点！万分感谢！

答：注意三点：

1、检查你的后仿真设置是否正确？库编译否？延时文件是否反标?

2、如果直接在 FPGA 上运行，结果是否准确?

3、激励文件中的一些指标要符合实际情况，如时钟等。

67、问：请问 MUXCY_L 和 MUXCY 用法有什么不同?

MUXCY_L : 2-to-1 Multiplexer for Carry Logic with Local Output

MUXCY : 2-to-1 Multiplexer for Carry Logic with General Output

The LO output can only connect to other inputs within the same CLB slice.

CLB : configurable logic block (CLB)

答：功能是一样的，但输出所在的网络不同

68、问：FPGA 引脚锁定问题求助，一个 FPGA 中有几个模块，它们之间有相互关系，在配置引脚时，某一个模块的引脚没有配置，会对电路的其他功能造成影响吗，还是仅仅有该模块相关的功能丢失？从实际上讲。谢谢。

答：对设计中没有约束的管脚，布局布线工具会将其任意绑定到没有约束设置的管脚上。但如果这个 FPGA 管脚在你的开发板上已经有了其他设置 (比如因为某种原因上拉或者下拉)，那么就有可能产生影响。曾经遇到过有一根起 reset 作用的 pin 没有约束到，结果工具任意的绑定，在某一次布局布线的时候就绑定到了一个下拉管脚上，导致系统起不来。

69、问：ASIC 与 FPGA 设计的差别？ Verilog 代码上有何不同？

答：区别还是蛮大的。主要的一点是，它们对应的最底层器件不一样；ASIC 设计需要将语言对应到 foundry 提供库，而 FPGA 则要对应赛灵思的 LUT 等或 ALTERA 的 LE。所以有些语句在 ASIC 和 FPGA 中是不一样的。

70、问：MATLAB 和 ModelSim 的联合仿真怎么做？

答：就是用 matlab 里面的 simlink 这个工具箱，里面有具体实例，可以看 Sysgen 安装目录中的 sysgen_user.pdf，具体见 http://www.xilinx.com/support/documentation/sw_manuals/sysgen_user.pdf

71、问：virtex5 中的 MAC 内是否有 CRC 功能？使用 virtex5 中的 IPCORE 生成的 MAC，里面是否有 CRC 功能，谢谢！

答：V5 有 CRC32 的硬核

72、问：最近做了块 XC5VLX110 的 FPGA 板，电装完后，测量 VCCIN 管脚和 GND 之间只有 12 欧姆的电阻，谁能告诉我这个阻值是否正常？谢谢！

答：正常的。一般静态电阻依据工艺、晶体管的数目不同而不同。我接触过的许多 asic 芯片和 FPGA 芯片的静态电阻很多都只有十几个欧姆。

73、问：关于双向口的仿真，如果双向口用作输入口，输出口该怎么设置？

答：做仿真时，软件会自动地将 IO 口（包括双向口）的引脚本加入到 . SCF 文件中去 . 先新建一个 SCF 文件，然后在 NODE–>ENTER NODES FROM SNF–>LIST, 将列出的所有 IO 引脚（包括了双向口）都加入仿真文件中，就可以进行仿真了 .

74、问：FPGA 输出的 VGA 信号 可以直接连接到 CRT 显示器么？

查看资料说 VGA 信号是模拟信号，而 FPGA 出来的是数字信号，不知道可不可以直接连接到 CRT 显示器，如果需要转换的话，又改怎么做呢？请做过的友人援手！

答：数字信号是只有 0 和 1，模拟信号是一个时钟周期可以携带更多的数据量，而且是连续变化的。用 AD7125 去把 8bit 的数字信号变为模拟信号。

如果不知道怎么接，随便找个赛灵思的 demo 板，看看原理图就知道了。

75、问：在看一个速度和面积优化的文档时，有一句话：You will obtain much better results if the hierarchy of the design is flattened 。对这句话不是很理解，是要把设计分解为比较细的模块呢，还是尽量把设计放在一个文件中？请高手指点，谢谢！

答：设计保持层次化还是扁平？你的设计可能有好几层，如果层次化、占面积小、速度慢，扁平化恰相反，也就是你如果不保持设计的 hierarchy，由软件帮你 flattened，你的设计经过 ISE 编译后的结果相对于保持 hierarchy 的结果有优化。

76、问：初学 EDK，对地址的设置有点疑问：新添加一个 IP 后，它的地址是否可以在保证不冲突的情况下随意设置还是有一定的规则？

答：如果对系统不熟悉可以采用自动生成地址，如果熟悉可以自己设定，如果有冲突，系统会报错的。

77、问：关于 VHDL 的问题：process(a, b, c) begin… end process; 如果 a、b、c 同时改变，该进程是否同时执行三次？

答：PROCESS STATEMENTS 中的执行跟逻辑有关系，假如是同步逻辑，则在每次时钟的触发沿根据 A, B, C 的条件来执行一次；假如是异步逻辑，则根据判断 A、B、C 的条件来执行 . 一般我们都推荐使用同步逻辑设计

78、问：怎样把 V-4 器件上产生的 cordic 移植到 V-5 的器件上实现？ V-4 的器件支持 cordic v3.0 IP 核，但 V-5 不支持，请教有人知道怎样把在 V-4 上产生的核移植到 V-5 上吗？不胜感激！

答：在 V4 下生成 IP，然后将芯片改为 V5 即可。

79、问：看到不同的资料上关于 CLB 的说法，有的说一个 CLB 包含两个 SLICE，而有的说四个 SLICE，我不知道到底有几个，还是跟器件的型号有关？

答：没错，与器件有关系的，V5 一个 CLB 包含 2 个 SLICE，V4 一个 CLB 包含 4 个 SLICE

80、问：我在网上看到，对于比较大型的设计最后分模块综合和布线之后再整体综合布线。可是我又一个疑问，就是：什么才叫做分块综合呢？举个例子，我写了个 UART，里面有并串转换、计时等小模块；开始我做好功能划分之后就动手编写各个小模块，然后把小模块 分别综合、仿真没问题之后，再将各个小模块在顶层中例化，然后综合顶层 module，再布局布线，这样叫分模块综合么？我觉得这样不是吧，因为最后还是综合布线顶层的整体模块呀。分块仿真综合时仅仅是验证各个小模块是否正确与否而已。那么什么叫分块综合？怎么才能做到分块综合布线呢？我使的是 ISE9，请高手给讲解下，多谢了。

答：分块综合的好处主要是让综合器对该模块的逻辑优化最佳而不受其他模块影响。各个模块完成后，在顶层分别例化各个模块，并声明成黑盒子，利用分块综合后输出的网表文件连接成一个总体的网表文件即可。

81、问：赛灵思 FPGA 在 配置期间 IO 口处于什么状态？根据论坛中一篇帖子的说法，配置期间 IO 口处于

高阻态。但是，我在实际测量中却发现输出引脚在配置期间处于高电平。实际中需要将输出引脚在配置期间设置为低电平。请教怎么操作啊?

答：HSWAP 引脚

'L' 时为上拉

'H' 时为高阻

82、问：v5 中 ibufds(差分转单端的输入接口) 的 100 欧匹配电阻怎么激活?

用的 v5 的片子，输入信号为差分输入，电平标准为 lvds25，调用 ibufds 时，怎么例化里面的 100 欧姆的匹配电阻啊? 语法上说把 ibufds 中的差分终端 (DIFF_TERM) 属性设为 TRUE 就可以了，可是感觉没有起作用啊！！硬件上还需要什么支持的地方吗?

答：如果你知道语法可以自己在约束文件中加，不知道的话通过引脚分配窗口可以选择配置该电阻。

83、问：我们为什么要写 testbench ?

答：经常看到论坛里有人问我们为什么要写 testbench，总是觉得不好回 答，下面是整理出来的一些理由供大家参考。

与写 testbench 相对应的功能手段还有画波形图，两者相比，画波形图的方法更加直观和易于入门，那为什么我们还要写 Testbench 呢? 原因有以下五点：

第一、画波形图只能提供极低的功能覆盖。

画波形无法产生复杂的激励，因此它只产生极其有限的输入，从而只能对电路的极少数功能进行测试；而 testbench 以语言的方式描述激励源，容易进行高层次的抽象，可以产生各种激励源，轻松地实现远高于画波形图所能提供的功能覆盖率。以 PCI 转以太网电路设计为例，该设计并不复杂，但却已经需要考虑多种情况：PCI 的配置读写、存储器读写等操作；以太网的短包、长包等。如果这些激励都用画波形图完成，其工作量是难以想象的；用 testbench 则可以轻松完成这些工作。

第二、画波形无法实现验证自动化。

对于规模设计来说，仿真时间很长，如果一个需要仿真一天设计在检错时仅通过画波形图来观测，将几乎不能检查出任何错误；而 testbench 是以语言的方式进行描述的，能够很方便地实现对仿真结果的自动比较，并以文字的方式报告仿真结果。我们甚至可以在下班时开始仿真，然后第二天早上上班时再查看验证结果。

第三、画波形图难于定位错误。

用画波形图进行仿真是一种原始的墨盒验证法，无法使用新的验证技术；而 testbench 可以通过在内部设置观测点，或者使用断言等技术，快速地定位问题。

第四、画波形的可重用性和平台移植性极差。

如果将一个 PCI 转 100Mb 以太网的设计升级到 PCI 转 1000Mb 以太网，这时原来画的波形图将不

得不重新设计，耗费大量的人力物力及时间；但若使用 testbench，只需要进行一些小的修改就可以完成一个新的测试平台，极大地提高了验证效率。

第五、通过画波形的验证速度极慢。

Testbench 的仿真速度比画波形图的方式快几个数量级。在 Quartus 下画波形需半个小时才能跑出来的仿真结果，在 ModelSim 下使用 testbench 可能只需几秒钟就可以完成。

84、问：请问大家 Modelsim 仿真的时候怎么把时间锁死在 1us 呢？怎么改啊？程序如下：

```
'timescale  1ns/1ns
module frequencyfix_v;
// Inputs
reg clk;
reg reset;
reg cin;
// Outputs
wire [15:0] qo;
wire[3:0]out1,out2,out3,out4;
parameter delay1=1;
parameter delay2=20;
// Instantiate the Unit Under Test (UUT)
frequency uut (
  .clk(clk),
  .reset(reset),
  .qo(qo),
  .cin(cin),
  .out1(out1),
  .out2(out2),
  .out3(out3),
  .out4(out4)
);
initial begin
  // Initialize Inputs
  clk = 0;
  reset = 1;
```

```
  cin = 0;
  // Wait 100 ns for global reset to finish
  #30 reset=0;
#10000 $finish; // 设置了结束时间是 10us, 为什么不起作用呢?
  // Add stimulus here
end
always
#delay1 cin=~cin;
always
#delay2 clk=~clk;
initial
begin
$monitor($time,,,"clk=%b,reset=%b,cin=%b,qo=%b,out1=%b,out2=%b,out3=%b,out4=%b",clk,re
set,cin,qo,out1,out2,out3,out4);
end
endmodule
```

请大家多多指教！！！谢谢

答：其实你的设置已经发挥作用了。不过 Modelsim 下，使用系统任务 finish 会自动退出，不能直接在仿真窗口看波形。建议使用stop,这样仿真自动暂停,可以直接查看波形,也可以在输出窗口看到打印的信息。

85、问：我用 EDK 编译一个程序，有比较多的 .C .H 文件，我在主函数文件包含了引用的其他文件的 C 函数 (这些函数也是我写的)，但是连接过程中，报错，找不到我引用的 C 函数，比如：我的主函数在文件 MAINING.C, 我写了另外一个文件 READ.C, 在里面我定义了一些函数，并且我写了一个 .H 文件，包含了 READ.C 的 C 函数，我将这个头文件写在 MAINING.C 中，但是编译报错： /cygdrive/c/testtcpip/iofordm9000a-102/TestApp_Peripheral/src/TestApp_Peripheral.c:105: undefined reference to `arp_retransmit'

```
/cygdrive/c/testtcpip/iofordm9000a-102/TestApp_Peripheral/src/TestApp_Peripheral.c:115:
undefined reference to `age_arp_cache'
undefined reference to
```

undefined reference to，这个是什么错误？为什么找不到我的 C 函数呢？不是已经包含了吗？是编译环境需要设置吗？似乎不对，这个不是库函数，是我写的函数啊？请赐教！

答：undefined reference to 是一个错误，网上常讨论，一、这个肯定是连接时候发生的错误，第一步，查找不到的函数是库函数，还是自己的函数，无论那种，加上头文件就可以了，二，可以查看 .O 文件，

每个 C 文件肯定要生成 O 文件，没生成就肯定找不到。回到我的 问题，我的问题很简单，我把一些 C 文件的后缀名字 C 大写了，比如 A.C, 一些 C 文件的后缀名字 C 小写了，比如 B.c ,这个问题在单片机编程没有问题,比如 51,但是用 32 位就有问题,我估计是这样,单片机只有 C 编译，链接，但是 MICROBLAZE 的 GCC 不同，可以用 C++, 可以 C，所以有 GCC++,GCC，编译，大写 C 文件表示为 C++ 文件,小写为 C 文件,所以链接器不给链接在一起,但是估计修改链接文件也可以连接，我不知道如何修改，只是全部小写就没有错误了。

86、问：最近看 V5 数据手册上说使用同一个块状 RAM，SDP 的位宽可为真双口的两倍 (好像赛灵思和 Altera 的 FPGA 都有这个说法)。我想了一下，以本身一个端口位宽 32b 为例。在使用 SDP 情况下可以进行 64b 的读写，实现起来无非是同时用两个端口所有的数据位，这样就可以实现。但这样来就出现个问题，那就是在一个端口写 (读) 的时候就要屏蔽另一个端口的读 (写)。这个功能通过外围逻辑可以实现，但实现起来给时序带来很多问题,有可能一个端口的操作将被屏蔽很多个周期。请问谁能给我解惑一下,谢谢!

答：不是那样的。读和写是并行操作的。只要读的地址不是正在写的地址就可以

87、问：当前系统有多片 Xilinx Virtex-5 FPGA，FPGA 之间需要传递信号，是否可以直接通过用户 I/O 的硬连线来实现？应该注意什么问题？如果不行，应该怎么做？

答：可以是可以，不过延时会较内部来的大很多，如果是逻辑功能互连也无可厚非如果是逻辑间通信的话，建议还是走些通信协议

88、问：我打算在 OPB 总线挂双口 RAM，一端 AD 采集数据，一端接 OPB，但是怎么挂双口 RAM 呢？我有三个问题：1。双口 RAM 是在 ISE 里面用核生成器生成的，如何把他弄成一个 IP 核？如果自己写的逻辑直接拷贝到 EDK 下就可，但是 CORE GEN 生成的，怎么弄？

2。弄好核，把核的代码加在 EDK 工程下的哪个文件里面？

3。加完后如何和 OPB 连接？有人告诉我用 OPB BRAM CONTR 的 IP，但是具体怎么连接起来，就是 RAM 一端如何连接到 IP 核的线上？请赐教!

答：使用 EDK->Hardware->Create or Import Peripheral core gen 生成的 IP 会有一个 *.v 文件来说明他的 IO，其实就是一个 wrapper，其内部调用的是网表文件，应该是一个 ngc 文件，或者 edf 文件。

然后把这里你生成的所有的 IP 相关文件，比如 *.v 文件，和网表文件，一起放到 /{EDK project}/pcores/{user IP}/hdl/ 目录下，这个目录下本来就应该有一个 user logic.v 文件，你在这个 user logic.v 中调用这个 coregen 生成的 IP 即可。

当然我默认你这里的 IP 使用了 opb 或者 plb 或者 fsl 总线。以及你使用的 HDL 语言是 Verilog，VHDL 语言的话，其实是一样的，只是文件的后缀也许会有一些不同。

89、问：时钟抖动和漂移的区别？谁能给一点时钟抖动和漂移的详细解释资料，两者的区别是什么？

答：简单的说，如果把实际时钟和理想时钟做一个 CYCLE TO CYCLE 的比对，可以分离出来两种误差，一种时长期的漂移 (WANDER)，一种是短时的各周期之间的抖动 (jitter)。造成 wander 的原因通常是温飘，电压漂移等固有的或者缓慢变化的因素。造成 JITTER 的原因通常是 CROSS TALK,HEAT NOISE 等原因，这些因素会附加在每个时钟周期上造成相邻的周期之间都会有随机偏差。

90、问：关于 FPGA 的时钟设计有几个问题不是很明白，请大家指点：

1.FPGA 的全局时钟缓冲器 (BUFG) 资源是否是有限的？我在做逻辑仿真的时候发现 BUFG 用多了就会出现无法布线的错误，是否是由于 BUFG 资 源不够所造成的？

2. 是否所有的时钟信号都需要经过 BUFG 来增加其驱动能力？我的设计中包含了：外部器件的时钟输出和内部的时钟调用。

3. 一个 BUFG 是否可以驱动多个时钟信号？

答：BUFG 连接的是芯片中的专用时钟资源，目的是减少信号的传输延时，提高驱动能力，对于时序电路中的关键时钟信号，这是非常重要的，关系到系统设计的成功与否。

如果内部产生的时钟，只在局部模块使用，可以考虑不使用 bufg。

一个 BUFG 只能驱动一个时钟。

91、问：我在 FPGA 设计中碰到这个问题：我的信号经过几个延时单元，每经过一个延时单元输出的信号加给 D 触发器，结果发现 D 触发器的输出有几个是个亚稳定的状态，在该状态下，触发器的输出不能识别为 1 或 0，输出为“X”。我觉得是因为触发和时钟的两个沿正好碰到了一起，所以输出不稳定。但我的触发信号不能随意改变，时钟信号也是一定的，即不能改变他们的时续关系，那我怎么样才能使它们输出都是在稳定状态呢？请问哪里有关于在 Virtex-6 内针对高速存储接口内建内存控制器的信息？

答：1、你多用几个 D 锁存器，一般，都是使用两个的。某书上说，通常通过 3 个 D 触发器以后的亚稳态出现的概率几乎为 0。 当然，多加了触发器，可能你就要少用几个延时的东东了。

2、或者使用边沿提取的方法。

另：不是说提倡同步设计吗？那么触发应该是时钟信号了，可你的描述，触发用的是其他信号。“因为触发和时钟的两个沿正好碰到了一起”。

92、问：在编译前设定一个模块的 Synthesis Style 为 FAST 是否一定比不设定 (NONE) 要节省 LC 资源？

答：在布局布线的过程中，Synthesis Style 的设置会影响到资源的利用率和速度的快慢，一般情况下：设置为 FAST 主要是为了提高设计的速度．在软件中除了综合类型的设置，还有一项是选择优化的目的：oPTIMIZE–>AREA OR SPEED. 选择 AREA 可以节省设计所占用的资源。

93、问：为了保证设计可靠性，需要重点关注哪些方面？

答：关于可靠性 FPGA 设计的几点建议。

1、使用完全同步设计 . 异步设计对路径延迟非常敏感，因此不很可靠 . 异步电路的一个例子是使用组合反馈的 SR 闭锁。

2、绝不使用组合逻辑控制时钟信号 . 因为在任何门控制时钟信号上可能产生短时脉冲干扰，最终导致错误触发 flip–flop。

3、绝不要依靠门延迟。

4、FPGA 的电源和接地引脚附近应该放置足够多的旁路电容器 . 使用优质高频响应电容器。

5、在 FPGA 上始终使用全局时钟缓冲来驱动内部时钟信号 . 并且已经仔细设计了这些时钟缓冲和关联时钟配电网，以将畸变减至最小。

94、问：Synthesis Style 设为 FAST 后，发现速度有所提升，同时使用的资源也减少了，资源和速度似乎兼得了，那么是否所有的模块都可以设定为 FAST 呢？

答：将 SYNTHESIS STYLE 设置为 FAST 主要是为了提高系统性能 . 但是有一点要记住的是，软件的设置不是在任何情况下对所有的设计都表现出相同的结果 . 针对这个设计模块，将 SYNTHESIS STYLE 设置为 FAST 可能对资源和速度都有了优化，但这并不说明对所有的模块都有相同的效果，但是可以试一试 . 设计优化是一个原则与经验，技巧相结合的过程，我们只掌握一定的原则与方法，根据我们自己的经验，运用一定的技巧，才能将一个设计做到最优化 .

95、问：有时候碰到这样的问题：在 iMPACT 中执行 Initialize 命令后出现一连串的 Error，无论如何找不到 FPGA 了。如何解决？

答：下面的方法可能可以为你解决：

1、首先确认并口是否打开：

在 BIOS 设置中找到 Parallel Port 的选项，一般情况下将它设置到 EPP+ECP(增强型并口) 模式。

2、重新安装 ISE 附带的并口驱动程序：

通常情况下产生这种问题的最主要的原因是 Xilinx ISE 的并口驱动被覆盖，或者你安装 ISE 的时候就没有安装并口驱动程序。因此，解决方法就是重新安装并口驱动程序。提醒：你只需要安装并口驱动程序，而不需要安装整个 ISE，整个过程只需要不到 1 分钟就可以解决的。

96、问：请教赛灵思软件安装事项？

答：赛灵思全部软件都不能安装在带空格带中文字符的目录中，也就是说不能装在 Program Files 这个目录下。建议所有软件都装在某个盘的根目录下。

需要注意的是，SysGen 因为需要安装在 Matlab 的 toolbox 目录下，因此 Matlab 也不可以安装在带空

格带中文的目录中!

97、问：为什么赛灵思器件中 BRAM 大小是 18K？

答：18K 是为了存放校验位的考虑。通常需要校验时，每 8bit 需一位校验位，因此长度是 16+2=18。

但是，BRAM 并没有产生校验位的功能，他内部所有的位功能都是一致的，都可以用作存储。

在使用 BRAM 时，存储深度以 2k 为分界点，如果存储深度 <=2k，即存储形式为 9bit x 2k，那么此时可以用满 18K 的 BRAM；而如果存储深度 >2k，即 4bit x 4K 或 2 bit x 8k 这种形式，则最多只能用到 16K 的 BRAM。

98、问：如何在 ISE 中生成 ucf、xcf、和管脚分配 (assign package pins)？

就是如何生成约束文件，并进行管脚分配 (Assign Package Pins)?

其中在管脚分配中，有多项选择：

例如，Loc 项，需要从 Bank0---Bank7 中选择，那么制定 Bank 的标准是什么呢？随便吗?

I/O Std 项：AGP、CTT、GTL、GTLP、HSTL_I、HSTL_III、LVCOMS2、LVTTL、SSTL2 、SSTL3 等电平标准，怎么来对应我要分配 的 I/O 脚呢?

Termination 项：PULLUP、 PULLDOWN 、KEEPER 这三个被选，我怎么知道那个 I/O 脚该上拉电阻还是下拉呢，KEEPER 什么意思?

Slew 项：fast slow 选项，怎么选呢?

Delay 项：BOTH 、IBUF 、IFD 、NONE 这四个备选项什么意思，我怎么知道那个 IO 选对应的备选项呢？谢谢指点!

答：1、建立 ucf 文件：建一个文本文件，存成后缀为 .ucf 的文件，在工程中 add source 这个文件就可以。

2、管脚分配：(1) 直接编辑 ucf 文件，ucf 文件可以用记事本直接打开，也可以选中 ucf 文件，再 process 中点击“Edit documents”可以打开。

编辑时，“yourname” LOC = FPGA 的管脚。即指定了管脚位置。

“yourname” IOSTANDARD = LVTLL 等电平标准。即指点了改管脚的电平标准。

(2) 点击 process 中的“assign Package Pins ”，在打开的界面中，把左边的管脚直接拖进图中的管脚就可以。或者在右边列表中直接输入。

3、bank 的电平标准可以查看器件的 user guide，里面有一章介绍得很详细，每个 bank 都可以随意设置为该器件支持的电平标准，不同的电平标准在一个 bank 中要注意它们的电平要一致，比如都为 3.3v，电平可以为 LVTTL、LVCOMS33。

4、你自己的管脚要什么标准，需不需要上拉，和你自己的设计有关系。

5、skew 选 fast，IO 转化时快，但电流大，功耗大。

skew 选 slow，IO 转化慢，但功耗小。

查看器件的 DC AC Switching 特性手册有详细说明。

6、Delay，输入不需要延迟，就选 NONE。

建议你先仔细看看用户手册，IO 一章会详细说明的。

99、问：子模块的网表分别存在多个目录怎么办？

答：当子模块以网表形式提供时，NgdBuilder (Translate) 会搜索工程目录下的 edn, ngc 等网表文件。如果存在子目录中，在 Translate 属性中的 Macro Search Path 填写目录名。

如果有多个子目录，不是在 Macro Search Path 中填写多个目录名，而要在下面一条填写其他 Translate option 的地方写上 –sd –sd 。每个 –sd 后只能写一个目录名。这一点在 dev.pdf 中提到。

100、问：当一个大程序中，有一些函数从来没有被调用过，用 GCC 编译仍然会把他们保留在最终的输出 elf 中。怎样去除这些没有被调用过的函数呢？

答：XPS –-> Software –-> Software Platform Settings –-> Software Platform –-> extra_compiler_flags = –g –ffunction–sections –fdata–sections

Project –-> Right Click –-> Set Compiler Settings –-> Paths and options –-> Other compiler options to append –-> –ffunction–sections –fdata–sections –Wl,––gc–sections

注意：

如果是一个有 interrupt 的系统，用了以上方法会导致 interrupt vector 和 interrupt handler 也被 remove 掉。暂时没有解决办法。

第八章、FPGA开发资源总汇

一、官网大全

1、赛灵思官方网站 http://china.xilinx.com/

2、赛灵思大学计划官方网站 http://china.xilinx.com/univ/

3、赛灵思开放源码硬件社群 http://www.openhw.org

4、赛灵思网上技术支持中心 http://china.xilinx.com/support/mysupport.htm

5、赛灵思开发工具下载中心 http://china.xilinx.com/support/download/index.htm

6、赛灵思 IP 核中心 http://china.xilinx.com/ipcenter/

7、赛灵思第三方合作伙伴信息 http://china.xilinx.com/alliance/index.htm

8、赛灵思技术解决方案大全 http://china.xilinx.com/technology/index.htm

9、赛灵思重要合作伙伴安富利 http://www.avnet.com/

10、赛灵思官网论坛 http://forums.xilinx.com/xlnx/

专业网站、FPGA开发博客信息

1、《FPGA 实用开发教程》田耘等编辑 http://www.eefocus.com/html/08-11/415501121236t2kE.shtml

2、RickySu 的博客 www.rickysu.com

3、FPGA 专业书籍网店 www.china-pub.com

4、EDA 中国门户网 www.edacn.net

5、电子创新网 www.eetrend.com

6、电子工程专辑 www.eetchina.com

7、电子设计技术 www.ednchina.com

8、电子系统设计 www.ed-china.com

9、21IC 中国电子网 www.21ic.com

10、电子产品世界 www.eepw.com.cn

第九章、编委信息与后记

《FPGA 开发全攻略》经过紧张有序的协作后终于面市了，在此特别向参与此书编撰的作者们表示衷心的感谢，他们是：

赛灵思高级产品经理　梁晓明

赛灵思亚太区市场经理　张俊伟

云创工作室北京邮电大学信息工程学院学生　田耘、徐文波等

赛灵思 PAE　苏同麒

中科院上海技术物理研究所　童 鹏　胡以华

希望这本追求基础、实用与深度的电子书能给 FPGA 开发者带来实际的帮助！也欢迎工程师朋友就 FPGA 开发中的需求与我们交流，以便我们可以开发后续版本提供更好的内容，提升本土 FPGA 应用创新水平！

再次感谢大家的支持！

张国斌
richard@eetrend.com
《FPGA 开发全攻略》电子书主编

第十章、版权声明

5、作者联系方式 richard@eetrend.com