赛灵思 中国通讯
Xilinx News

第四十四期 2012年 夏季刊 Issue 44, Summer 2012

赛灵思 Vivado 设计套件震撼登场 针对未来十年 “All Programmable” 器件的颠覆之作

FPGA IP 核软硬件协同验证采用形式验证技术

用纯硬件解决方案加速部分重配置进程

以 FPGA 为起点的智能高速交易平台

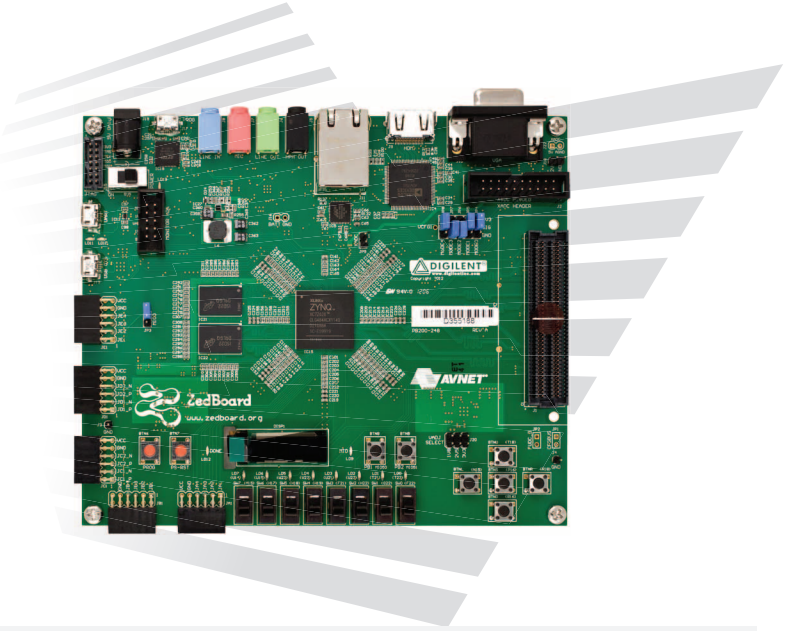
XILINX®

请即浏览赛灵思中国通讯网络版的全部精彩内容

www.xilinx.com/cn/xcell44

行业首款面向开源社区的

Zynq™-7000 EPP 开发套件



ZedBoard

DESIGNED BY AVNET

ZedBoard是一款基于赛灵思Zynq™-7000可扩展处理平台(EPP)的低成本开发板，也是行业首款面向广大开源社区的Zynq™-7000 EPP可扩展处理平台开发套件。开发板为基于Linux、安卓、Windows或其它操作系统/实时操作系统的设计开发提供了所需的一切。另外，该平台提供数款扩展连接器，便于用户访问处理系统和可编程逻辑。Zynq-7000 EPP紧密集成了ARM®处理系统和7系列可编程逻辑，充分利用它们的优势，并结合ZedBoard可以开发出独树一帜且功能强大的设计。为推动ZedBoard套件的创新分享和交流还专门打造了www.zedboard.org开源社区，用户可以通过这个社区与其他同样从事Zynq设计的工程师开展各种各样的协作。

套件组成部分：

- Avnet ZedBoard 7020 基础板
- 12 V AC/DC 电源
- 4 GB SD 卡
- Micro-USB 电缆
- USB 适配器: Micro-B （公头）对 Standard-A（母头）
- 入门指南
- ISE WebPACK™， 配器件专用 ChipScope 许可证

特性：

- Zynq-7000 EPP XC7Z020-CLG484-1
- 存储器
  - 512 MB DDR3
  - 256 Mb Quad-SPI Flash
  - 4 GB SD 卡
- 板载 USB-JTAG 编程
- 10/100/1000 以太网
- USB OTG 2.0 和 USB-UART
- PS & PL I/O 扩展 (FMC, Pmod™, XADC)
- 支持多显示器 (1080p HDMI, 8-bit VGA, 128 x 32 OLED)
- I²S 音频编解码器

目标应用：

- 视频处理
- 电机控制
- 软件加速
- Linux/Android/实时操作系统开发
- 嵌入式 ARM 处理
- 通用 Zynq™-7000 EPP 原型设计

Price : USD$ 395.00

Part Number : AES-Z7EV-7Z020-G

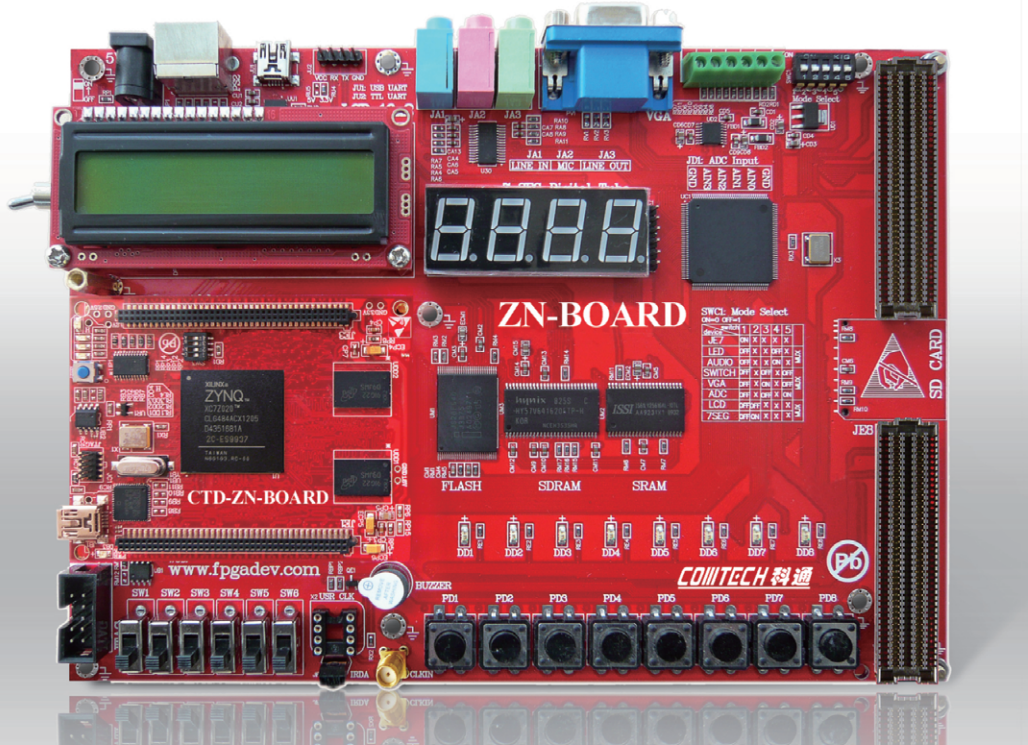
更多关于ZedBoard的信息，请访问：
http://www.zedboard.org 或联系以下安富利办事处

北京：010-8206 2488
武汉：027-8732 2806
成都：028-8652 8262
厦门：0592-516 3621
重庆：135-9422 8267
沈阳：024-8290 2597
青岛：0532-8097 0716
香港：00852-2176 5388
上海：021-3367 8387
南京：025-8483 8137
杭州：0571-8580 0667
西安：029-8833 6372
广州：020-2283 8106
深圳：0755-8378 1886

www.em.avnetasia.com

AVNET 安富利电子元件

ZN A SOLUTION TO ADDRESS YOUR NEED



主要参数

主芯片	XC7Z020 CLG484
配置方案	16MB SPI Flash SD卡接口 JTAG 调试接口
储存器	DDR3 SDRAM 16MB SPI Flash I2C接口EEPROM
通信 & 网络接口	千兆以太网 USB OTG/Host CAN 总线 USB UART串口
显示接口	HDMI/VGA 高清输出 8个 LED
扩展接口	2个 FMC LPC 连接器 IIC HUB/Expander
时钟方案	200MHz系统主时钟 156.25MHz I2C可编程时钟 33.33MHz CPU时钟
控制接口和IO	8个用户按键 6个用户开关 8 个指示灯
供电系统	12V 便携电源或ATX电源
模拟接口	XADC 扩展口

选配FMC子卡，可支持专业领域应用

- 高清视频和图像处理
- 机器视觉和视频分析
- 工业仪表和测试测量
- 通用软件无线电平台
- 汽车辅助驾驶和导航

更多详细信息请联系：xilinx_enquiry@comtech.com.cn

COMTECH 科通
www.comtech.com.cn

深圳 电话: 86-0755-26743210/0755-26988221/0755-26743087
北京 电话：86-010-51726678/79/80/81/82
上海 电话：86-021-23218585/51696680
南京 电话：86-025-86816659 / 86816709
杭州 电话：86-0571-51020941/42/43/44/45/46
成都 电话：86-028-85131563/85131763/85187804/85131589/85139576
西安 电话：029-84508551 / 84508553
武汉 电话：86-027-87690655/87777241/87568667
香港 总机：852-27301522

赛灵思 Vivado 设计套件震撼登场 针对未来十年“All Programmable”器件的颠覆之作

赛灵思采用先进的 EDA 技术和方法，推出了全新的工具套件，可显著提高设计生产力和设计结果质量，使设计人员能够用更少的芯片更快地打造出更好的系统。

作者：Mike Santarini
赛灵思公司《Xcell 杂志》发行人
mike.santarini@xilinx.com

历经四年的开发和一年的试用版本测试，赛灵思 Vivado 设计套件终于震撼登场，并通过其早期试用计划开始向客户隆重推出，随后将于今年夏季公开发布。Vivado 提供高度统一的设计环境，并配有全新生成的系统级和芯片级工具，所有这些都构建在共享、可扩展的数据模型架构和通用调试环境的基础上。该套件还是一款基于业界标准的开放式环境，诸如 AMBA® AXI4 互联、IP-XACT IP 封装元数据、工具命令语言 (Tcl)、Synopsys 设计约束 (SDC) 以及其他有助于设计流程满足用户需求的业界标准。赛灵思设计的这款 Vivado 设计套件支持各类可编程技术的组合使用，并可扩展到 1 亿个 ASIC 等效门的设计。

赛灵思市场营销与公司战略高级副总裁 Steve Glaser 表示："在过去的四年中，赛灵思将半导体技术的创新推向了一个全新的高度，并全面释放了可编程器件的系统级能力。在这段时间里，赛灵思已经发展成一家能够开发 All Programmable 器件的公司，其可编程功能超越了传统的可编程逻辑和 I/O，实现了软件可编程的 ARM 子系统，3D 堆叠芯片和模拟混合信号。借助获奖的 Zynq ™ -7000 EPP（可扩展式处理平台）器件、采用 3D 堆叠硅片互联（SSI）技术的 Virtex®-7 器件以及全球最先进的 FPGA，我们将实现全新的可编程系统集成度。现在有了 Vivado，我们将推出最先进的工具套件，通过采用面向未来十年 "All Programmable" 器件，帮助客户提高生产力。"

Glaser 指出赛灵思开发的 "All Programmable" 器件使客户能够实现全新的可编程系统集成度，提高系统性能，降低材料成本和总系统功耗，最终提高设计生产力，从而帮助他们加速创新技术与产品的上市进程。为实现这一目标，赛灵思需要开发一个如同其新的芯片一样具有创新性的工具套件。要提高设计生产力，该套件应能够解决一系列集成和实现方面的瓶颈问题。

Steve Glaser 指出，客户所面临的一系列集成瓶颈问题包括集成算法 C 和寄存器传输级（RTL）的 IP；混合 DSP、嵌入式、连接功能和逻辑域；验证模块和 "系统"，以及设计和 IP 的重用等。同时他们还面临几大实现方面的瓶颈问题，其中包括层次化芯片的规划与分区；多域和多芯片的物理优化；多元的 "设计" 与 "时序" 收敛；以及后期的 ECO 和设计修改的连锁效应。

赛灵思推出的全新 Vivado 设计套件不仅能够解决上述瓶颈问题，同时还可使用户充分利用我们的 "All Programmable" 器件的系统集成能力。

在 Vivado 设计套件的开发过程中，赛灵思以业界标准为标杆并采用了先进的 EDA 技术与方法。为此，无论是需要高度自动化按键式流程的客户，还是需要实际操作性极强的可修改流程的客户，他们现在能够比以往更快更高效地进行设计（甚至包括赛灵思最大型的 FPGA 设计），同时还是在一个熟悉而直观的先进的 EDA 环境下工作。

Vivado 设计套件为客户提供一款具有完整系统可编程功能的新型工具套件，该套件功能远远超越了赛灵思为时甚久的旗舰型 ISE® 设计套件的功能。为帮助客户顺利过渡到 Vivado 设计套件的使用，赛灵思将继续开发 ISE 设计套件并坚定地为采用 7 系列及更早期的赛灵思 FPGA 技术的客户提供 ISE 支持。今后 Vivado 设计套件将成为赛灵思的旗舰设计环境，支持所有 7 系列器件及赛灵思未来器件。

赛灵思公司设计方法市场营销高级总监 Tom Feist 预计，一旦客户启用 Vivado 设计套件，就会立即体会到其相对于 ISE 的明显优势。

Feist 说："与同类竞争工具相比，Vivado 设计套件的运行时间可缩短高达 4 倍，能够显著提升用户的设计生产力。同时该设计套件纯熟地运用了多种业界标准，诸如 System Verilog、SDC（Synopsys 设计约束）、C/C++/System C、ARM AMBA AXI-4 互联、互动 TCL（工具命令语言）脚本。Vivado 设计套件的其它突出优势包括为 Vivado 的众多报告和设计视图提供全面的交叉探测功能、高级图形化 IP 集成功能、以及首款得到 FPGA 厂商全面支持的商用高层次综合技术（C++ 到 HDL 综合）。

面向新一代可编程设计的工具

赛灵思早在 1997 年就推出了 ISE 设计套件。ISE 套件采用了当时非常具有创新性的基于时序的布局布线引擎，这是 1995 年 4 月赛灵思收购 NeoCAD 获得的。在其后 15 年的时间里，随着 FPGA 能够执行日趋复杂的功能，赛灵思为 ISE 套件增添了许多新技术，包括多语言综合与仿真、IP 集成以及众多编辑和测试实用功能，努力不断从各个方面改进 ISE 设计套件。Feist 表示，赛灵思通过借鉴 ISE 设计套件的所有经验和关键技术，并充分利用最新 EDA 算法、工具和技术，才打造出了这一颠覆性的全新 Vivado 设计套件。

Feist 表示："Vivado 设计套件将显著提升当今设计的生产力，且能够轻松实现升级扩展，应对 20nm 芯片及更小工艺技术所带来的容量和设计复杂性挑战。在过去 15 年时间里，EDA 技术取得了长足的发展。我们是从头开始开发这套工具的，所以我们能够在套件中采用最先进的 EDA 技术和标准，让其具有很强的前瞻性。"

确定性的设计收敛

任何 FPGA 厂商的集成设计套件的核心都是物理设计流程，包括综合、布局规划、布局、布线、功耗和时序分析、优化和 ECO。有了 Vivado，赛灵思打

造了一个最先进的设计实现流程，可以让客户更快地实现设计收敛。

可扩展的数据模型架构

为减少迭代次数和总体设计时间，并提高整体生产力，赛灵思用一个单一的、共享的、可扩展的数据模型建立其设计实现流程，这种框架也常见于当今最先进的 ASIC 设计环境。Feist 说："这种共享、可扩展的数据模型可让流程中的综合、仿真、布局规划、布局布线等所有步骤在内存数据模型上运行，故在流程中的每一步都可以进行调试和分析，这样用户就可在设计流程中尽早掌握关键设计指标的情况，比如时序、功耗、资源利用和布线拥塞等。而且这些指标的估测将在实现过程中随着设计流程的推进而更趋于精确。"

具体来说，这种统一的数据模型使赛灵思能够将其新型多维分析布局布线引擎与套件的 RTL 综合引擎、新型多语言仿真引擎，以及 IP 集成器 (IP Integrator)、引脚编辑器 (Pin Editor)、布局规划器 (Floor Planner)、器件编辑器 (Device Editor) 等各工具紧密集成在一起。客户可以通过利用该工具套件的全面交叉探测功能来跟踪并交叉探测原理图、时序报告、逻辑单元或其它视图，直至 HDL 代码中的给定问题。

Feist 说："用户现在可以对设计流程中的每一步进行分析，而且环环相扣。综合后，我们还可对设计流程的每一步进行时序、功耗、噪声和资源利用分析。这样我们就能够很早发现时序或功耗问题并通过几次迭代快速、前瞻性地解决问题，而不必等到布局布线完成后通过长时间执行多次迭代来解决。"

Feist 指出，这种可扩展数据模型提供的紧密集成功能还增强了按键式流程的效果，从而可满足用户对工具实现最大自动化，完成大部分工作的期望。同时，Feist 还表示，这种模型还能够满足客户对更高级的控制、更深入的分析以及掌控每个设计步骤进程的需要。

芯片规划层次化，快速综合

Feist 说，Vivado 为用户提供了设计分区功能，可以分别处理综合、实现、验证等设计工作。因此客户在执行大型项目时，可以通过分组设计加速设计进程。同时，新的设计保存功能可以实现时序结果的重复利用，并且可以实现设计的部分可重配置功能。

Vivado 还包括一个全新的综合引擎，旨在处理数以百万计的逻辑单元。新型综合引擎的关键优势是能够为 System Verilog 提供强大支持。Feist 指出："相比市场上其它任何工具而言，Vivado 的综合引擎能够对 System Veriog 语言的可综合子集提供更强大的支持。"它的综合速度是赛灵思 ISE 设计套件中赛灵思综合技术 (XST)

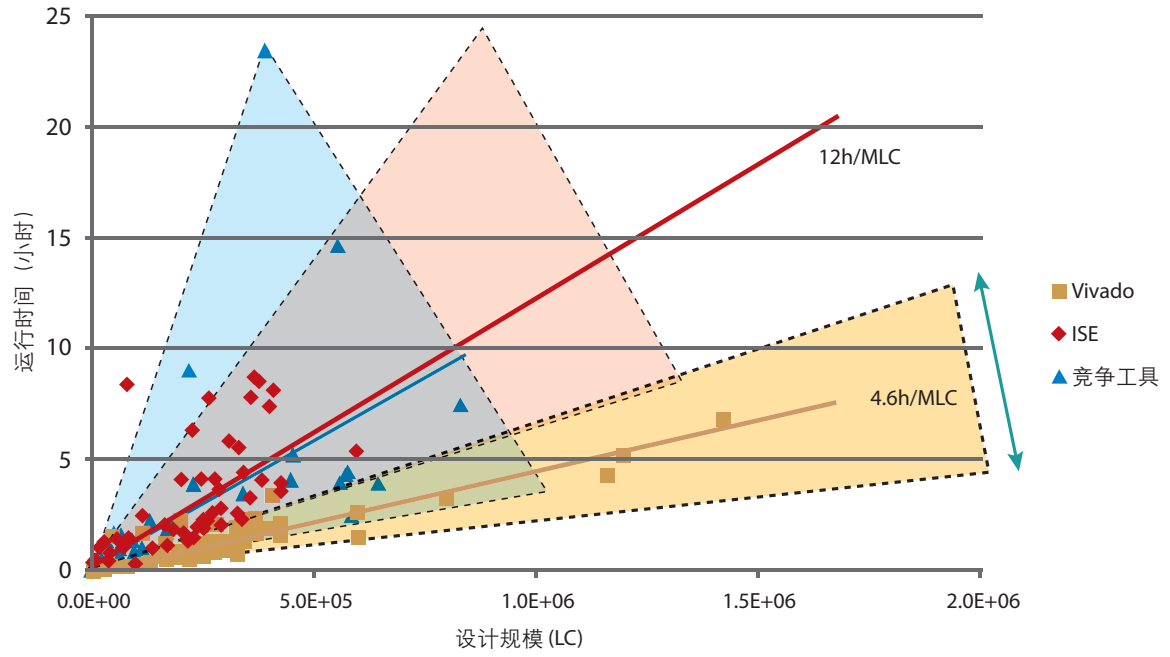
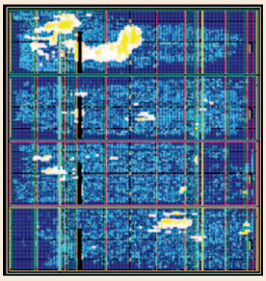
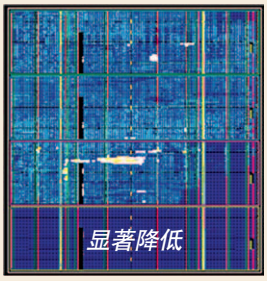


图 1 - 与其它 FPGA 工具相比，Vivado 设计套件能够以更快的速度、更优异的质量完成各种规模的设计

	ISE	Vivado
部分重配置运行时间	13 小时	5 小时
内存使用	16 GB	9 GB
走线长度及拥塞情况		

*Zynq 仿真平台

图 2 - Vivado 设计套件的多维分析算法可创建专门针对最佳时序、拥塞和走线长度（而不仅仅只是针对最佳时序）优化的布局。

的三倍，并支持“快速”模式，使设计人员能够迅速把握设计的面积和规模。 另外，也让他们调试问题的速度比之前采用 RTL 或门级原理图快 15 倍。随着越来越多的 ASIC 设计人员转向可编程平台，赛灵思还在整个 Vivado 设计流程中充分发挥了 Synopsys 设计约束 (SDC) 的优势。标准的使用开启了一个新的自动化水平， 客户现在可以访问业界先进的 EDA 工具来生成约束、检查跨时钟域、形式验证， 甚至是利用像 Synopsys PrimeTime 那样的工具进行静态时序分析。

多维度分析布局器

Feist 解释说，上一代 FPGA 设计套件采用单维基于时序的布局布线引擎，通过模拟退火算法随机确定工具应在什么地方布置逻辑单元。使用这类路由器时，用户先输入时序，模拟退火算法伪随机地布置功能“尽量”与时序要求吻合。Feist 说：“在当时这种方法是可行的，因为设计规模非常小，逻辑单元是造成延迟的主要原因。但今天随着设计的日趋复杂化和芯片工艺技术的进步，互联和设计拥塞一跃成为延迟的主因。

采用模拟退火算法的布局布线引擎对低于 100 万门的 FPGA 来说是完全可以胜任的，但对超过这个规模的设计，引擎便不堪重负。不仅仅有拥塞的原因，随着设计的规模超过 100 万门，设计的结果也开始变得更加不可预测。”

着眼于未来数百万门规模的设计，赛灵思为 Vivado 设计套件开发了新型多维分析布局引擎，其可与当代价值百万美元的 ASIC 布局布线工具中所采用的引擎相媲美。该新型引擎通过分析可以找到从根本上能够最小化设计时序、拥塞和走线长度问题的解决方案。Feist 表示：“Vivado 设计套件的算法从全局进行优化，实现了最佳时序、拥塞和走线长度，它对整个设计进行通盘考虑，不像模拟退火算法只着眼于局部调整。这样该工具能够迅速、决定性地完成上千万门的布局布线，同时保持始终如一的高结果质量（见图 1）。由于它能够同时处理三大要素，也意味着可以减少重复运行流程的次数。”

为展现工具在大型复杂设计中的这种优势，赛灵思分别在 ISE 设计套件和 Vivado 设计套件中用按键式流程方式同时运行针对赛灵思 Zynq-7000 EPP 仿真平台开发的原始 RTL。这两款工具均是针对赛灵思全球最大容量的 FPGA 器件——采用堆叠硅片互联技术的 Virtex-7 2000T FPGA 而构建的。Vivado 设计套件的布局布线引擎仅耗时 5 个小时就完成了 120 万个逻辑单元的布局，而 ISE 设计套件 13.4 则耗时长达 13 个小时（图 2）。而且采用 Vivado 设计套件实现的设计拥塞明显降低（设计中显示为灰色和黄色的部分），器件占用面积较小，这说明总体走线长度缩短。Vivado 设计套件实现方案还体现出更出色的内存编译效率，仅用 9GB 就实现设计要求的内存，而 ISE 设计套件则用了 16GB。

Feist 表示：“从本质上来说， 你看到的就是 Vivado 设计套件可满足所有约束要求，而且实现整个设计仅占用了 3/4 的器件资源。这意味着用户可以为自己的设计添加更多的逻辑功能和片上存储器，甚至可以采用更小型的器件。”

功耗优化和分析

当今时代，功耗是 FPGA 设计中最关键的因素之一。因此，Vivado 设计套件重点放在先进的功耗优化技术的开发上，帮助用户进一步降低设计功耗。Feist 表示：“我们在先进的功耗优化技术的开发过程中采用了目前高级 ASIC 工具套件用到的先进时钟门控制技术，通过该技术可以分析设计逻辑，同时消除不必要的翻转。具体来说，先进的功耗优化技术侧重于翻转因子‘alpha’，它能够将动态功耗降低 30%。”赛灵思去年在 ISE 设计套件中开始应用该技术，并一直沿用至今。Vivado 将继续加强这一技术的应用。

此外，有了这一新的可扩展的数据共享模型，用户可以在设计流程的每一个阶段得到功耗的估值，从而可以在问题发展的前期就能预先进行分析， 从

而能够在设计流程中，先行解决问题。

简化工程变更单（ECO）

增量式流量让快速处理少量的设计修改成为可能，每次修改后只需重新实现该小部分设计，从而加快迭代速度。它们还能在每次增量式修改后实现性能保存，从而无需多次设计迭代。为此，Vivado 设计套件还包括一个 Vivado 器件编辑器，这是由最受欢迎的 ISE FPGA 编辑器工具扩展而来。Feist 说，在一个布局布线设计上使用 Vivado 器件编辑器，设计人员现在能够在设计周期的后期制作工程变更单 (ECO)，即移动示例，重新布线，连接寄存器至主输出端作为调试管脚，修改数字化时钟管理器 (DCM) 或查找表（LUT）的参数，无需通过返回设计重新综合和实现。他说，目前整个业界没有任何其它 FPGA 设计环境能够提供如此高的灵活性。

流程自动化，非流程强制化

在 Vivado 设计套件构建过程中，赛灵思工具团队遵循这样的原则“自动化，而非强制性人性化设计方式”。Feist 说：“不管用户用 C、C++、SystemC、VHDL、Verilog、System Verilog、MATLAB 还是 Simulink 开始编程，也不管他们用的是我们的 IP 还是第三方的 IP，我们提供了一种方式，可帮助他们实现所有流程的自动化，提高设计生产力。我们还充分考虑到我们的用户的各种技能水平和偏好，既能满足需要全按键式流程的客户的要求，也能满足在设计流程的每一步都进行分析的客户的要求，甚至还能满足那些认为用 GUI 的是低手，喜欢用 TCL 以命令行或批处理模式完成全部设计流程的客户的要求。用户能够根据自己的特定需求，选用套件功能。”

> 这些工具可满足各种水平的用户的需求，既能满足需要全按键式流程的客户的要求，也能满足在设计流程的每一步都进行分析的客户的要求。

IP 封装器、集成器和目录

赛灵思的工具架构团队把重点放在新套件专门的 IP 功能设计上，以便于 IP 的开发、集成与存档。为此，赛灵思开发出了 IP 封装器、IP 集成器和可扩展 IP 目录三种全新的 IP 功能。

Feist 表示：“今天很难找到不采用 IP 的 IC 设计。我们采用业界标准，提供专门便于 IP 开发、集成和存档 / 维护的工具，这都有助于我们生态系统合作伙伴中的 IP 厂商和客户快速构建 IP，提高设计生产力。目前已有 20 多家厂商提供支持该最新套件的 IP。”

采用 IP 封装器，赛灵思的客户、赛灵思公司自己的 IP 开发人员和赛灵思生态环境合作伙伴可以在设计流程的任何阶段将自己的部分设计或整个设计转换为可重用的内核，这里的设计可以是 RTL、网表、布局后的网表甚至是布局布线后的网表。IP 封装器可以创建 IP 的 IP-XACT 描述，这样用户使用新型 IP 集成器就能方便地将 IP 集成到未来设计中。IP 封装器在 XML 文件中设定了每个 IP 的数据。Feist 说一旦 IP 封装完成，用 IP 集成器功能就可以将 IP 集成到设计的其余部分。

Feist 说：“IP 集成器可以让客户在互联层面而非引脚层面将 IP 集成到自己的设计中。可以将 IP 逐个拖放到自己的设计图（canvas）上，IP 集成器会自动提前检查对应的接口是否兼容。如果兼容，就可以在内核间划一条线，然后集成器会自动编写连接所有引脚的具体 RTL。”

Feist 表示：“一旦用 IP 集成器在您的设计中集成了四五个模块，您也可以取出已用 IP 集成器集成的四五个模块的输出，然后通过封装器再封装。这样就成了一个其他人可以重新使用的 IP。这种 IP 不一定必须是 RTL，可以是布局后的网表，甚至可以是布局布线后的网表模块。这样可以进一步节省集成和验证时间。”

第三大功能是可扩展 IP 目录，它使用户能够用他们自己创建的 IP 以及赛灵思和第三方厂商许可的 IP 创建自己的标准 IP 库。赛灵思按照 IP-XACT 标准要求创建的该目录能够让设计团队乃至企业更好的组织自己的 IP，供整个机构共享使用。Feist 称赛灵思系统生成器 (System Generator) 和 IP 集成器均已与 Vivado 可扩展 IP 目录集成，故用户可以轻松访问已编目的 IP 并将其集成到自己的设计项目中。

Vivado 产品营销总监 Ramine Roane 指出：“以前第三方 IP 厂商用 Zip 文件交付的 IP 格式各异，而现在他们交付的 IP，不仅格式统一，可立即使用，而且还与 Vivado 套件兼容。”

VIVADO HLS 把 ELS 带入主流

可能 Vivado 设计套件采用的众多新技术中，最具前瞻性的要数新的 Vivado

HLS（高层次综合）技术，这是赛灵思 2010 年收购 AutoESL 后获得的。在收购这项业界最佳技术之前，赛灵思对商用 ESL 解决方案进行了广泛评估。市场调研公司 BDTI 的研究结果帮助赛灵思做出了收购决策（见赛灵思中国通讯杂志第 36 期“BDTI 研究认证以 DSP 为核心的 FPGA 设计的高层次综合流程”*http://china.xilinx.com/china/xcell/xl36/2-7.pdf*）。

Feist 表示：“Vivado HLS 全面覆盖 C、C++、SystemC，能够进行浮点运算和任意精度浮点运算。这意味着只要用户愿意，可以在算法开发环境而不是典型的硬件开发环境中使用该工具。这样做的优点在于在这个层面开发的算法的验证速度比在 RTL 级有数量级的提高。这就是说，既可以让算法提速，又可以探索算法的可行性，并且能够在架构级实现吞吐量、时延和功耗的权衡取舍。”

设计人员使用 Vivado HLS 工具可以通过各种方式执行各种功能。为了演示方便，Feist 讲解了用户如何通过一个通用的流程进行 Vivado HLS 开发 IP 并将其集成到自己的设计当中。

在这个流程中，用户先创建一个设计 C、C++ 或 SystemC 表达式，以及一个用于描述期望的设计行为的 C 测试平台。随后用 GCC/G++ 或 Visual C++ 仿真器验证设计的系统行为。一旦行为设计运行良好，对应的测试台的问题全部解决，就可以通过 Vivado HLS Synthesis 运行设计，生成 RTL 设计，代码可以是 Verilog，也可以是 VHDL。有了 RTL 后，随即可以执行设计的 Verilog 或 VHDL 仿真，或使用工具的 C 封装器技术创建 SystemC 版本。然后可以进行 System C 架构级仿真，进一步根据之前创建的 C 测试平台，验证设计的架构行为和功能。

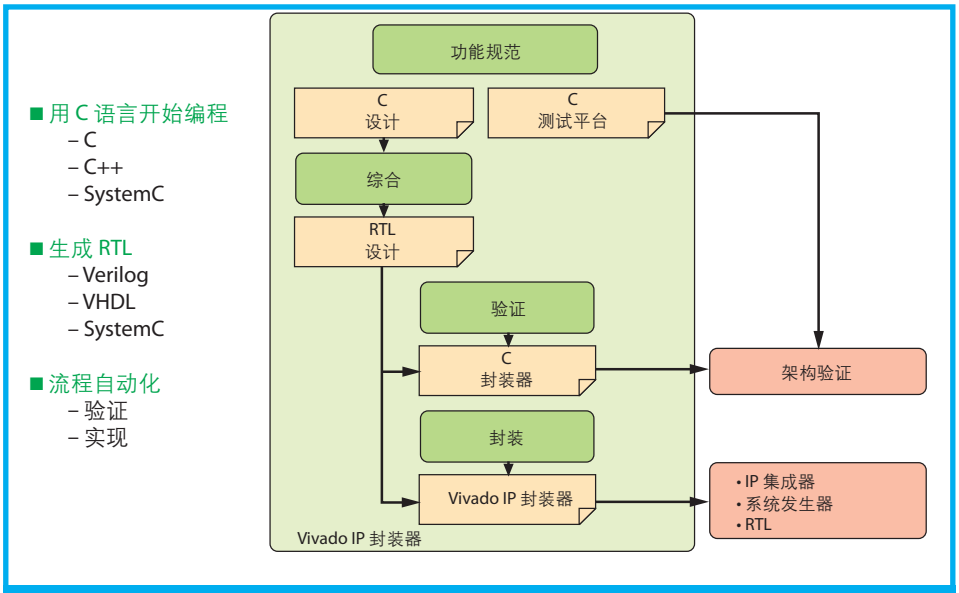


图 3 - Vivado HLS 支持设计团队直接从系统级开始他们的设计。

设计固化后，就可以通过 Vivado 设计套件的物理实现流程来运行设计，将设计编程到器件上，在硬件中运行和 / 或使用 IP 封装器将设计转为可重用的 IP。随后使用 IP 集成器将 IP 集成到设计中，或在系统生成器 (System Generator) 中运行 IP。

这只是使用该工具的方法之一。实际上在即将发行的赛灵思《Xcell 杂志》中，安捷伦的 Nathan Jachimiec 和赛灵思的 Fernando Marinez Vallina 将介绍如何使用 Vivado HLS 技术（在 ISE 设计套件的流程中称为 AutoESL 技术）为安捷伦开发 UDP 包引擎。

VIVADO 仿真器

除了 Vivado HLS，公司还为该套件新开发了一种同时支持 Verilog 和 VHDL 的混合语言仿真器。Feist 表示，只需要单击鼠标，用户就可以启动行为仿真，然后从集成波形查看器中查看结果。通过采用最新性能优化的仿真内核，可加速行为级仿真速度，执行速度比赛灵思 ISE 设计套件仿真器快三倍。采用硬件协仿真，门级仿真速度则可加快 100 倍。

2012 供货情况

之前赛灵思 ISE 设计套件针对不同类型设计人员（逻辑、嵌入式、 DSP 和系统）所发行的四个版本， 赛灵思将推出 Vivado 设计套件的两个版本。其中， Vivado 基础设计版本包括新型 IP 工具和 Vivado 的综合 - 比特流流程。而 Vivado 系统版本则包括设计版本的所有工具、系统生成器和赛灵思的最新 Vivado HLS 工具。

Vivado 设计套件 2012.1 版本目前已随早期试用计划推出。如需了解更多详情，敬请联系您所在地的赛灵思代表。2012.2 版本将于第二季度中期公开发布，今年晚些时候还将推出 WebPACK。目前支持服务尚未到期的 ISE 设计套件用户除了 ISE 之外，将免费得到全新的 Vivado 设计套件。

对使用 28nm 器件之前器件的用户，赛灵思将继续提供对 ISE 设计套件的支持。如需了解更多 Vivado 详情，敬请访问 *www.xilinx.com/cn/design-tools*。

采用高级综合工具提供优化的数据包引擎设计

用 AutoESL 创建具有无需处理器的架构内 UDP 网络数据包引擎

作者：Nathan Jachimlec，博士
研发工程师
安捷伦科技 TLO
nathan_jachimiec@agilent.com

Fernando Martinez Vallina，博士
软件应用工程师
赛灵思公司
vallina@xilinx.com

得益于硬化三模以太网 MAC (TEMAC) 原语的可用，千兆位以太网是将工作站或笔记本电脑连接到基于 FPGA 的嵌入式平台的最常见的互连方法之一。基于以太网的 FPGA 设计开发的主要障碍在于，互联网协议 (IP) 栈的处理应配备什么样的处理器。我们通过采用 AutoESL 高层次综合工具开发出高性能 IPv4 用户数据报协议 (UDP) 数据包传输引擎，就可以解决这一问题。

我们在安捷伦测量研究实验室工作的研发小组根据互联网工程任务组关于 UDP、地址解析协议 (ARP) 和动态主机配置协议 (DHCP) 各协议间数据包交换的征求意见稿 (RFC)，编写了原始的 C 语言源代码。该设计无需 CPU 即可实现硬件数据包处理引擎。这种架构能够以线速处理流量，时延极低，逻辑资源面积占用少。使用 AutoESL，能够轻松调整用户接口，以适应一个或多个 FIFO 流，或多个 RAM 接口端口。AutoESL 是赛灵思 ISE® 设计套件的新增功能，在最新 Vivado 设计套件中又称为 Vivado ™ HLS（见封面文章）。

IPV4 用户数据包协议

互联网协议第 4 版 (IPv4) 是互联网的主流协议，第 6 版 (IPv6) 还在逐渐推广中。大多数开发人员讨论 IP 时，他们通常指的是传输控制协议（即 TCP）。这是一种基于连接的协议，可提供可靠性和拥塞管理。但对视频流、语音通信、游戏或分布式传感器网络等众多应用而言，提高带宽和最小化时延会影响可靠性。因此这类应用一般使用 UDP 替代 TCP。

UDP 是一种无连接传输模式，本身不具可靠性。如果数据包丢失，被复制或者乱序发送，发送方不会知道，用户应用负责执行某种包检测并纠错。从这个方面说，UDP 又称“不可靠协议”，但与 TCP 相比，它能提供更高的性能。每一种支持 IP 的操作系统基本上都提供 UDP 支持。高级软件编程语言将网络流称为“套接字”，而 UDP 则称为“数据报套接字”。

传感器网络架构

我们在安捷伦科技公司开发的是一种基于 LAN 的传感器网络，通过赛灵思 Virtex®-5 FPGA 可连接到一个模数转换器 (ADC)。该 FPGA 负责汇聚数据，然后按请求将一系列采样发送到预先设定的 IP 地址（即主机 PC）。由于我们的 FPGA 的 Block RAM 几乎全部专用于信号处理，因此没有足够的存储空间来容纳软处理器的固件。我们也可以选择实现一组极简略的网络功能，用于将传感器数据通过 UDP 回传到主机。由于需要高带宽和低时延，UDP 数据包流是优选的网络模式。

由于数据的时间敏感性，一组新的采样数据比重新传输丢失的采样更重要。我们面临的两大挑战之一是应避免主机设备过载。这就是说我们必须找到一种能够有效处理大量输入采样的方法。第二大挑战是在下一组采样溢出内部缓冲区之前迅速格式化 UDP 数据包并计算所需的 IP 报头字段以及虽属可选但也必要的 UDP 有效载荷校验和。

HDL 初始设计

鉴于有现成的虚拟代码，用 HDL 实现数据包引擎比较简单直观，但对我们的 FPGA 硬件而言不是最好的。来自各种源文件的 C 语言和虚拟代码可以简化验证工作。另外，像开源数据包分析器 Wireshark 这样的工具和像 Java 这样的高级语言可以简化仿真流程和实验室验证工作。

使用现成的虚拟代码开发用于生成数据包报头的 Verilog 会涉及到：对状态机进行编码、读取采样 FIFO，以及将数据包汇聚到 RAM 缓存等工作。我们把设计分为 RX Flow、TX Flow 和 LAN MCU 三大模块，如图 1 所示。当来自 LAN 的数据包抵达时，RX Flow 负责检查数据包，并将其发送给设备核心或 LAN MCU，以供处理，比如在处理 ARP 或 DHCP 数据包时就是这样。

TX Flow 数据包引擎负责从 TX FIFO 中读取 N 个 ADC 采样，并计算运行中有效载荷校验和，以便进一步计算出 UDP 校验和。TX FIFO 负责对新到达的采样进行缓冲，同时 LAN MCU 负责准备待发送数据包的有效载荷。在获得最后一个请求的采样后，LAN MCU 负责计算剩余的 IP/UDP 数据包的报头字段。在网络技术中，这个过程称为 TX 校验和卸载。

一旦生成数据包字段，LAN MCU 将数据包发到 TEMAC 处，以供发送，但 LAN MCU 会保留数据包直到 TEMAC 确认成功发出（而非成功被目的地设备接收）。在第一个数据包等待 TEMAC 发出时，新的传感器采样会抵达 TX FIFO。当第一个数据包发送完毕，数据包引擎会释放缓冲区，供下一个数据包缓冲。这个过程采用双缓冲方式。如果 TEMAC 发出出错信号且下一个发送缓冲区溢出在即，则会丢弃这个数据包，以便让下一一个采样集继续，但会提示异常。由于采样集的时戳是我们数据包格式的组成部分，主机会发现根据采样集的非连续性并进行适当调节。

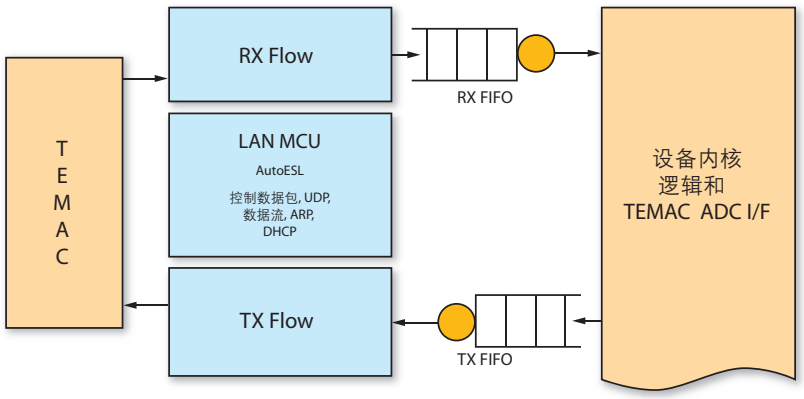


图 1 - 我们的 UDP 数据包引擎设计由 RX Flow、TX Flow 和 LAN MCU 三大模块构成
图中文字如下：

发送数据包的时延包括读取 N 个 ADC 采样所花的周期，以及生成数据包报头字段所花的周期。数据包报头字段包括 IPv4 标志、源地址段和目的地址段、UDP 伪报头，以及 IP 和 UDP 校验和。计算校验和相当难，因为它们需要读取整个数据包，但它们的位置又在有效载荷字节的前面。

摸黑编写 HDL 代码

为满足该传感器网络的高带宽、低时延要求，我们需要一种理想的硬件设计来实现所需采样率。在未进行布局规划的情况下，我们开始用 Verilog 实现的直观方法不能满足 125MHz 的时钟速率要求，且生成 IP/UDP 数据包报头字段需要 17 个时钟周期。在我们开发 HDL 初步设计时，ChipScope ™ 对掌握 TEMAC 接口的微妙之处起到了关键作用，但也阻碍了实现 125MHz 时钟速率的目标。额外的逻辑采集电路改变了关键路径，需要手动进行布局规划才能实现时序收敛。

关键路径负责计算 IP 和 UDP 报头校验和，因为我们的直观设计使用四运算元加法器将我们设计各种状态下的多个报头字段加总。我们的 HDL 设计尝试了一种“激进”调度算法，试图在状态机的每个周期中尽可能多地完成工作。通过删除 ChipScope，并进行布局规划，我们可实现时序收敛。

HDL 设计仅使用用作数据包发送缓冲区的 32 位宽 Bloc RAM 的一个端口。我们选择 32 位宽存储器是因为这是 BRAM 原语的原始宽度，支持字节使能写入访问，从而避免对发送缓冲区采用“读取 - 修改 - 写入”的访问方式。

采用字节使能，有限状态机 (FMS) 直接写入到 RAM 地址上需要修改的报头字段字节。但这个根据底层赛灵思器件架构和算法知识创建的看似优秀的设计方案，如果不手动布局四输入加法器，却是不能满足时序要求的非理想设计。

因为 UDP 算法已经以各种 C 语言代码的形式提供，或已经编写成 IP 相关 RFC 技术文档中的虚拟代码，所以用 C 语言重新编写 UDP 数据包引擎并不繁琐，而且有利于更深入了解数据包报头处理。在借鉴虚拟代码的基础上编写 Verilog 可以加快编码速度，但这种方法会影响性能，因为没有全面分析涉及到的数据和控制流程。

AUTOESL 的优势

AutoESL 能够抽象化 FIFO 和 RAM 接口，经证明这对性能优化大有裨益。由于能够直接用 C 语言进行编码，我们现在可以轻松地将 ARP 和 DCHP 程序纳入我们的数据包引擎中。图 2 是我们设计的流程图。我们的 HDL 设计使用字节宽度的 FIFO 接口连接到设计的汇聚和传感器接口，后者仍然保留 Verilog 设计。另外，我们的 Verilog 设计利用 32 位存储器接口收集 4 字节采样数据，然后将其以 32 位字的方式存储在 RAM 发送缓冲区中。

AutoELS 通过“阵列整形”指令优化了存储器接口，这样发送缓冲区虽然是用 C 语言代码编写为 8 位存储器，却变成了 32 位存储器。这就意味着 C 语言代码不必对报头字段进行大量的位操作，否则它们需要移位才能放入 32 位字。这也减轻了小端字节与大端字节的排序问题。通过这样的优化，负责计算数据包校验和以及生成报头字段的 TX 卸载功能发生的时延从原来用 Verilog 编程所需的 17 个时钟周期锐降低到了 7 个时钟周期，同时也轻松满足了时序要求。AutoESL 在将来还有改进的空间，因为当前版本还不能对 RAM 的写入执行字节使能。字节使能存储器支持已纳入该工具的长期发展规划。

由于赛灵思 Block RAM 与生俱来拥有双端口，我们还意外地发现 AutoESL 的另一项优化功能，即它能够同时访问我们存储器的两个端口。我们的 Verilog 设计保留了发送缓冲区的第二个端口，这样通过它到 TEMAC 的接口可以直接访问缓冲区，而无需进行任何仲裁。通过让 AutoESL 优化我们真正的双端口 RAM，它就能够从缓冲区的两个不同位置进行读 / 写操作，从而让生成报头所需的周期数减半。如此大幅的时延的降低，即便用 Verilog 创建一个专门用于存储器第二端口的简单裁决器也是值得的，这样通过 TEMAC 接口就可以访问 AutoESL 占用的存储器端口。

我们通过指令控制了发送缓冲区和采样 FIFO 接口的位宽。但令人遗憾的是 AutoESL 不能自动优化设计。

AutoESL能够抽象化 FIFO 和 RAM 接口，经证明这对性能优化大有裨益。

必须尝试各种指令，通过试错法找出哪种指令能够带来优化。对我们的设计来说，目标是减少处理数据包字段所需的时钟周期数，同时能以125MHz 的时钟速率运行。

“阵列整形”和环路“流水线”指令对优化设计而言，至关重要。整形指令能够修改 RAM 和 FIFO 接口的位宽，最终实现每个时钟周期多个报头字段并行处理和回写至存储器。要使时钟周期数最少的理想组合是将发送缓冲区的位宽设为 32。由于不可能迫使采样更快到达，因此负责发送 ADC 采样的 FIFO 的宽度对降低总时延没有影响。

环路流水线指令也极为重要，因为它向编译器指示自 FIFO 接口压入和弹出（Push and Pop）的环路可以连续运行。另外，如果没有流水线指令，由于调度原因，在 FIFO 的弹出之间，AutoESL 需要占用 3 至 20 个时钟周期。因此在存储器间传输数据时，应尽量使用流水线，这对实现低时延至关重要。

赛灵思 Block RAM 还提供 1 至 3 个时钟周期的可编程数据输出时延。使用三个周期的读取时延即可实现最短的“时钟至 Q”延迟。要实验不同的读取时延，只需修改针对 RAM 原语或“内核”资源的“时延”指令。由于 AutoESL 执行的调度算法的原因，给对 RAM 的访问增加三个时钟周期的读取时延，最终对总的数据包报头生成周期而言只会增加一个时钟周期的时延。额外两个时钟周期的存储器时延可延长设计时间，从而有助于布局布线。

我们还在 AutoESL 设计中实现了 ARP 和 DHCP 程序。因为用 Verilog 编程工作量很大，我们在前面避免这样做。虽然难度不大，但用 Verilog 编写 ARP 和 DHCP 极度繁琐，而且要求大量的状态才能运行。比如 ARP 请求 / 响应交换就需要 70 多种状态。Verilog 有限状态机 (FSM) 的一个编码错误就需要几天才能纠正。仅这个原因，许多设计人员就愿意使用 CPU 来运行这些网络例程。

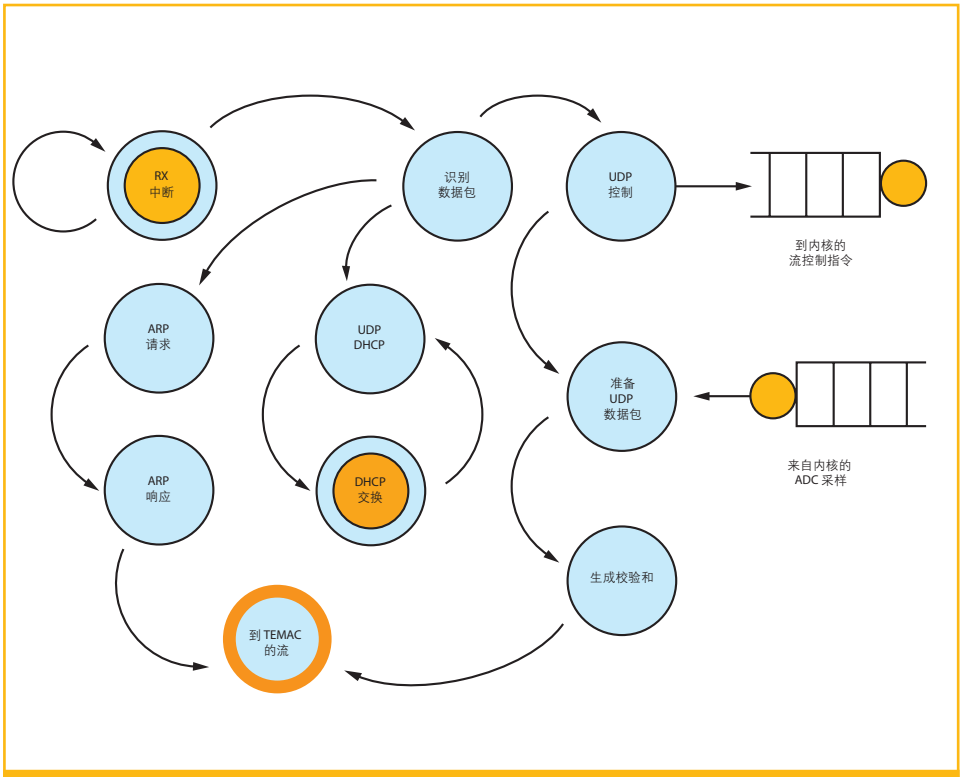


图 2 - 含 ARP 和 DHCP 在内的数据包引擎流程

总之，AutoESL 擅长于为 UDP 数据包引擎生成可综合的网表。AutoESL 生成的模块嵌入在我们之前已有的 ADC 模块和 TEMAC 接口模块之间，用于执行必要的数据包字段生成及其它任务。我们可以把 AutoESL 创建的设计集成到我们的内核设计中，并用 Mentor Graphics 的 ModelSim 对设计进行仿真，以验证其功能性。采用该优化设计，与我们最初的 HDL 设计相比，我们可以在减少综合、映射和布局布线工作量的情况下实现时序收敛。同时我们拥有丰富得多的功能，比如 ARP 和 DHCP 支持功能。

将我们用 Verilog 编写的原始设计和我们为创建“LAN MCU”和“TX Flow”模块而使用 AutoESL 完成的混合设计进行比较，结果令人印象深刻。表 1 是查找表 (LUT) 使用情况对比表。HDL 设计的 TX Flow 模块尺寸缩小 37% 以上，但 AutoESL 设计融入了更多功能。最引人瞩目的是 AutoESL 设计将完成数据包报头生成所需的时钟周期数压缩了 59%。表 2 列出了“TX Offload”算法的时延。

HDL 设计的关键路径是计算 UDP 校验和。将其与 AutoESL 设计进行对比，不难发现 HDL 设计有 10 层逻辑，总路径延迟达 6.4 纳秒，而 AutoESL 经过优化，仅有 3 层逻辑，路径延迟仅为 3.5 纳秒。我们 HDL 设计开发周期大约为一个月。用 AutoESL 开发虽

TX Flow 资源使用情况			
	HDL – TX	AutoESL - TX	增加 (%)
LUT	858	1,372	37.5

表 1 - AutoESL 设计使用更多的查找表但也融入更多功能

时延			
	HDL	AutoESL	增加 (%)
时钟周期	17	7	58.8%

表 2 - AutoESL降低了“TX Offload”算法的时延

然时间相当，但实现了更加丰富的功能，同时更能驾驭工具的使用。

时延和吞吐量

AutoESL 与 HDL 设计相比，明显优势在于它能够执行控制和数据流分析，并根据分析结果对运算进行重新排序，尽可能缩小时延，提高吞吐量。具体到我们的个案，我们使用激进算法，尽可能在每个时钟周期内完成大量算术运算。该工具重新安排校验和计算，这样仅需两个输入加法器，但其调度方式能够避免增大总体时延。

软件编译器本来就能完成这类工作。随着状态机日益复杂化，HDL 设计人员相对于无所不能的编译器处于不利地位。由于交付设计的时间限制，HDL 设计人员一般没有机会探索两种以上架构方案的效果，而这对实现低功耗设计来说，至关重要。

AutoESL 工具最明显的优势在于能够尝试多种方案，比如修改 FIFO 和 RAM 的位宽、将大型 RAM 分区为多个小存储器、对算术运算进行重新排序、将单端口 RAM 改用为双端口 RAM，这在 Verilog 设计中会非常繁琐。在 HDL 设计中，每种方案也需要多花一天时间来编写代码，然后修改测试平台，以验证功能的正确性。而使用 AutoESL 只需几分钟就可完成上述这些修改，整个过程完全无缝且无需对源代码大刀阔斧地修改。

在 Verilog 中修改大型状态机会极度繁琐。AutoESL 这类工具的问世让人不禁想起了微处理器设计人员开始使用微程序设计，告别为像 8086 和 68000 这样的早期微处理器手动编写微代码状态机的日子。随着 RISC 架构和硬件描述语言的诞生，微编程现在已是面临失传的技艺，但其教训应当汲取，即抽象化是控制复杂性所必需的。正如微编程为状态机设计提供更高一层抽象，总体而言，AutoESL 或高级综合工具也是一样。这种层次的工具能够让设计人员更专注于算法本身，而不是把重心放在那些容易出错、难以修改、是否能够满足未来需求等底层实现工作上。

利用 FPGA 加速分布式计算

采用赛灵思部分重配置技术的 SoC 网络为使用庞大激励数据集进行测试的算法提供云计算。

作者：Frank Opitz，理学硕士
汉堡应用技术大学
工程和计算机科学教员
计算机科学系
Opitz.frank@googlemail.com

Edris Sahak，理学硕士
汉堡应用技术大学
工程和计算机科学教员
计算机科学系
edris.sahak@haw-hamburg.de

Bernd Schwarz，教授，工程学博士
汉堡应用技术大学
工程和计算机科学教员
计算机科学系
Schwarz@informatik.haw-hamburg.de

高校和私企正在应用分布式平台，而不是安装速度更快、耗电更大的超级计算机来解决日益复杂的科学算法，针对 SETI@home 这样的项目，他们则使用数以千计的个人计算机来计算它们的数据。[1, 2] 当前的分布式计算网络一般使用 CPU 或 GPU 来计算项目数据。

FPGA 也正被像 COPACOBANA 这样的项目所采用，该项目使用 120 个赛灵思 FPGA 通过暴力处理来破解 DES 加密文件。[3] 不过在这个案例中，FPGA 都被集中布置在一个地方，这种方案不太适合那些预算紧张的大学或企业。目前并未将 FPGA 当作分布式计算工具，这是因为它们的使用需要借助 PC，才能用新的比特流不断地重新配置整个 FPGA。但是现在有了赛灵思部分重配置技术，为分布式计算网络设计基于 FPGA 的客户端完全可行。

我们汉堡应用技术大学的研究小组为这样的客户端创建了一个原型，并将其实现在单个 FPGA 上。我们的设计由静态和动态两大部分组成。其中静态部分在 FPGA 启动时加载，与此同时用静态部分实现的处理器从网络服务器下载动态部分。动态部分属部分重配置区域，提供共享的 FPGA 资源。[4] 采用这种配置，FPGA 可以位于世界上的任何地方，用较低的预算就能够为计算项目提供强大的计算能力。

分布式 SOC 网络

由于具有信号并行处理能力，FPGA 能够使用比微处理器慢 8 倍的时钟，低 8 倍的功耗实现比其快三倍的数据吞吐量。[5] 为利用该强大的计算能力实现高数据输入速率，设计人员一般将算法实现为流水线，比如 DES 加密。[3] 我们开发分布式 SoC 网络 (DSN) 原型的目的是加快算法的速度和使用分布式 FPGA 资源处理大型数据集。我们的网络设计采用“客户端 - 代理 - 服务器”架构，故我们可以将所有注册的片上系统 (SoC) 客户端分配给每一个网络参与方的计算项目（如图 1 所示）。这在将每一个 SoC 客户端连接到唯一的项目的“客户端 - 服务器”架构中是无法实现的。

另外，我们选择“代理 - 服务器”架构可以将每个 FPGA 的 TCP/IP 连接数量减少到一个。DSN FPGA 负责运算使用专用数据集的算法，而“代理 - 服务器”则负责管理 SoC 客户端和项目客户端。代理调度连接的 SoC 客户端，让每个项目在相同的时间几乎拥有相同的计算能力，或者在 SoC 的数量少于计算请求的项目时分时复用 soc 客户端。

项目客户端提供部分重配置模块 (PRM) 和激励输入数据集。在连接到“代理 - 服务器”之后，项目客户端将 PRM 比特文件发送给服务器，然后由服务器将它们分配给带有空闲的部分可重配置区域 (PRR) 的 SoC 客户端。SoC 客户端的静态部分是一个基于 MicroBlaze™ 的微控制器，用接收到的 PRM 动态重新配置 PRR。接下来，项目客户端开始通过“代理 - 服务器”发送数据集并从 SoC 客户端接收计算的结果。根据项目客户端的需要，举例来说，它可以比较不同的计算结果，或根据计算目的评估计算结果。

SOC 客户端

我们为随 ML605 评估板配套提供的赛灵

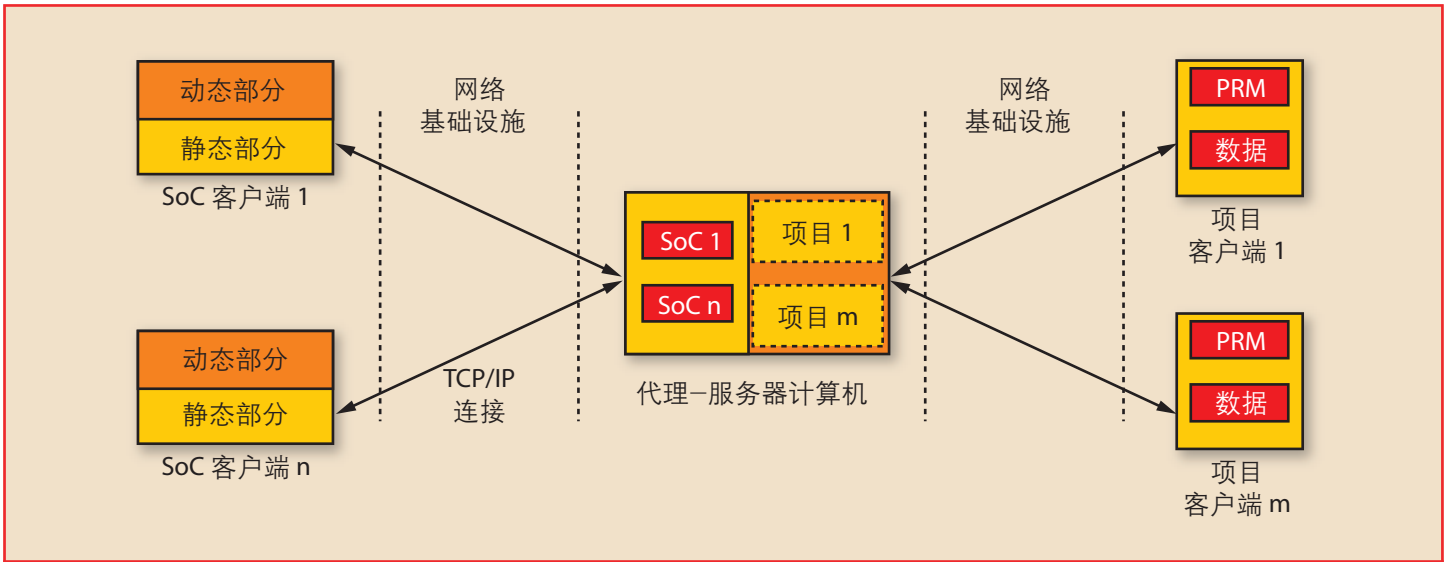


图 1 - 带有由 FPGA 实现，并由中央代理 - 服务器管理的 SoC 客户端的分布式 SoC 网络。项目客户端负责分配部分可重配置模块和数据集。SoC 客户端的动态部分通过 PRR 提供资源，静态部分内含的微控制器负责处理重配置工作。

MicroBlaze 处理器负责运行客户端软件，客户端软件管理部分重配置以及比特流和数据交换。

思 Virtex®-6 FPGA（XC6VLX240T）开发了 SoC 客户端。MicroBlaze™ 处理器负责运行客户端软件，客户端软件负责管理部分可重配置以及比特流和数据交换（如图 2 所示）。用户逻辑封装 PRR 的处理器本地总线 (PLB) 外设用以连接静态部分和动态部分。在动态部分驻留的是接收到的 PRM 提供的加速器 IP 核使用的 FPGA 共享资源。为存储接收到的数据和计算完成的数据，我们选择了 DDR3 存储器而非 CompactFlash，因为 DDR 存储器有更高的数据吞吐量和无限制的写入访问次数。PRM 存储在专用数据段内，以控制其大小，避免与其它数据集发生冲突。该数据段大小为 10 MB，足以存储完整的 FPGA 配置。因此每一个 PRM 都应该与这个数据段的大小匹配。

我们还为接收及结果数据集创建了不同的数据段。这些数据段的大小有 50 MB，能够为比如图像或加密文本文件等提供足够的寻址空间。管理这些数据段主要依靠 10 个管理结构。该管理结构包括每个数据集对的起始 / 终点地址，以及指示结果数据集的标志。

为将静态部分连接到 PRR，我们对赛灵思 EDK 提供的 IP 连接进行评估，比如快速单向链路 (FSL)、PLB 从和 PLB 主等。我们选择将 PLB 主 / 从结合使用，以取得便于配置的 IP，可以在无需 MicroBlaze 提供支持的情况下发送和接收数据请求，从而大幅降低每个字传输占用的时钟周期。

对客户端 - 服务器通信，FPGA 的内部以太网 IP 硬核是处理器系统静态部分不可或缺的外设。借助本地链路 TEMAC 对存储器控制器的软件直接存储器访问 (SDMA) 功能，可减轻数据和比特文件传输带来的 PLB 载荷。在接收 1,518 个字节的帧后，SDMA 生成中断请求，调用 lwip_read() 函数来处理这段数据。Lwip_write() 函数告知 SDMA 通过到 TEMAC 的发送通道执行 DMA 传输功能。

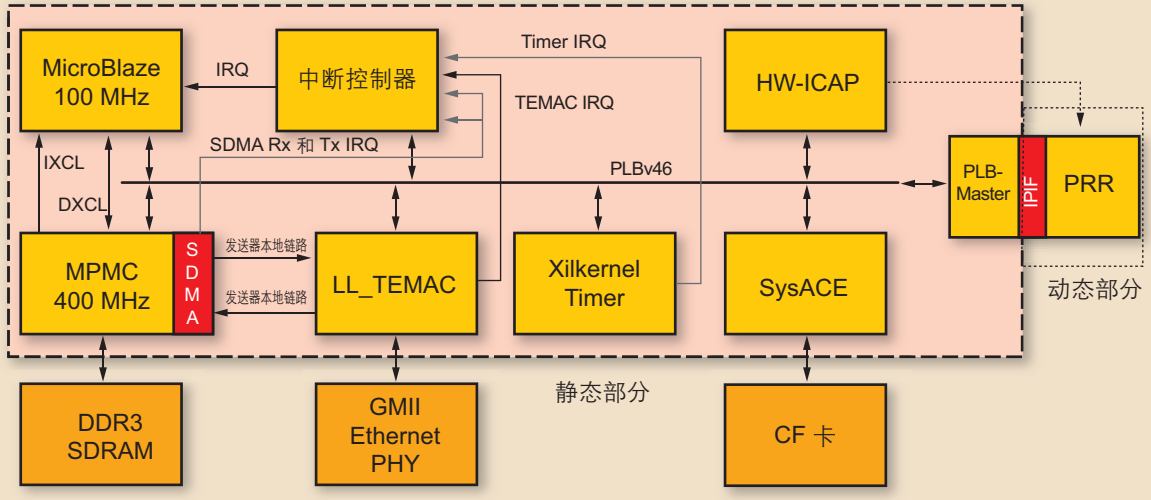


图 2 - SoC 客户端是一个配备静态部分和总线主外设的处理器系统，其包含部分可重配置区域 (PRR)。用 Virtex-6 FPGA XC6VLX240 在 ML605 开发板上实现。

我们把 Xikernel（一种用于赛灵思嵌入式处理器的内核），当作 SoC 客户端软件的底层实时操作系统加以实现，以便使用用于 TCP/IP 服务器连接的套接字模式发挥轻量级 TCP/IP 栈

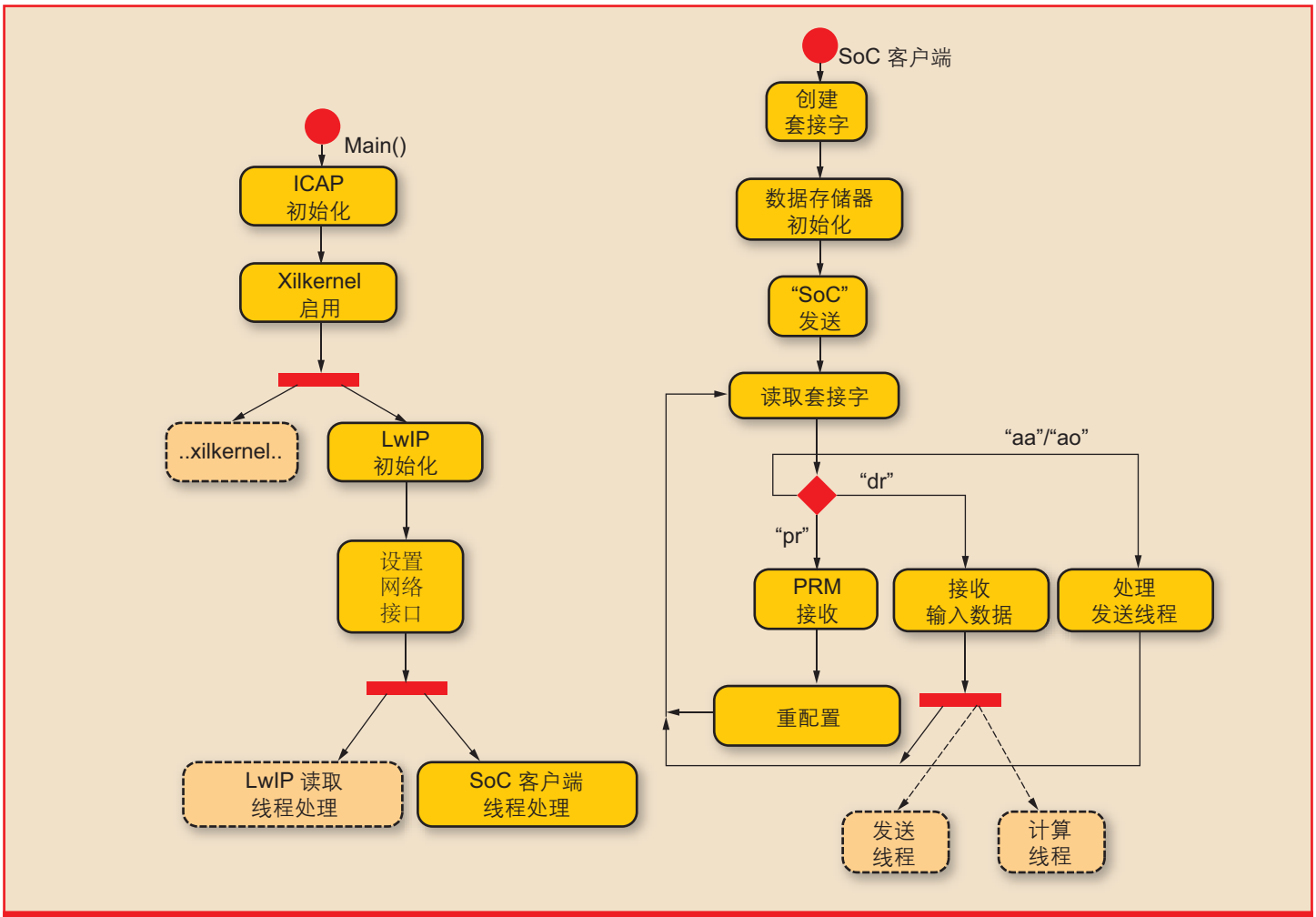


图 3 - SoC 客户端软件初始化和处理周期包括用 PRM 重新配置 PRR，从服务器恢复数据集，开始处理以及数据集恢复到服务器线程。
黑条表示从 Xilkernel 库调用 sys_thread_new() 来创建线程。

(LwIP) 库的作用。图 3 概述了客户端线程的初始化、建立、发送和处理顺序。SoC 客户端线程初始化到服务器的连接，并接收存储在 DDR3 存储器中的 PRM 比特流（“pr”），从而应用 XILMFS 文件系统。随后 Xps_hwicap（硬件内部配置接入点）用 PRM 重新配置 PRR。最后，由总线主外设设置一个状态位，命令 SoC 客户端向服务器发送请求。服务器用数据集（“dr”）做出响应，SoC 客户端把该数据集存储在板载存储器上。这些数据文件包含有内容顺序，比如“output_length+“ol”+data_to_compute”。output_length 是字节长度，用来保留结果数据的存储范围，后接字符对“ol”。对首个接收到的“dr”消息，会生成一个计算线程和一个发送线程。

计算线程将输入 - 结果数据集的地址发送到 PRR 外设的从接口，并启动 PRM 的自动数据集处理功能。管理结构为每个数据集提供这些地址，并在确保结果数据完全可用后设置“完成”标志。在目前的客户端软件概念版本中，计算线程和发送线程通过该结构通信，由发送线程反复检查完成位，并将 lwip_write() 调用存储在存储器中的结果。

在测试 SoC 客户端时，我们发现如果在 PRR 重配置过程中启用全部中断，Xilkernel 的定时器会产生调度函数访问 MicroBlaze，使重配置过程随机发生卡住的状况。如果禁用全部中断，或在没有 Xilkernel 的支持下，对 SoC 客户端的 MicroBlaze 处理器使用独立的软件模块，就不会发生这种情况。

配备例化 PRM 的总线主控外设

为在 PRM 和外部存储器之间实现自控激励数据和结果的交换，我们将总线主外设构建为一个带数据路径和控制路径的处理器元件（如图 4 所示）。在数据路径中，我们在两个深度均为 16 个字的 FIFO 模块之间嵌入 PRM 接口，以补偿通信和数据传输延迟。数据路径的两个 FIFO 均直接连接到 PLB 的总线主接口。这样，我们通过

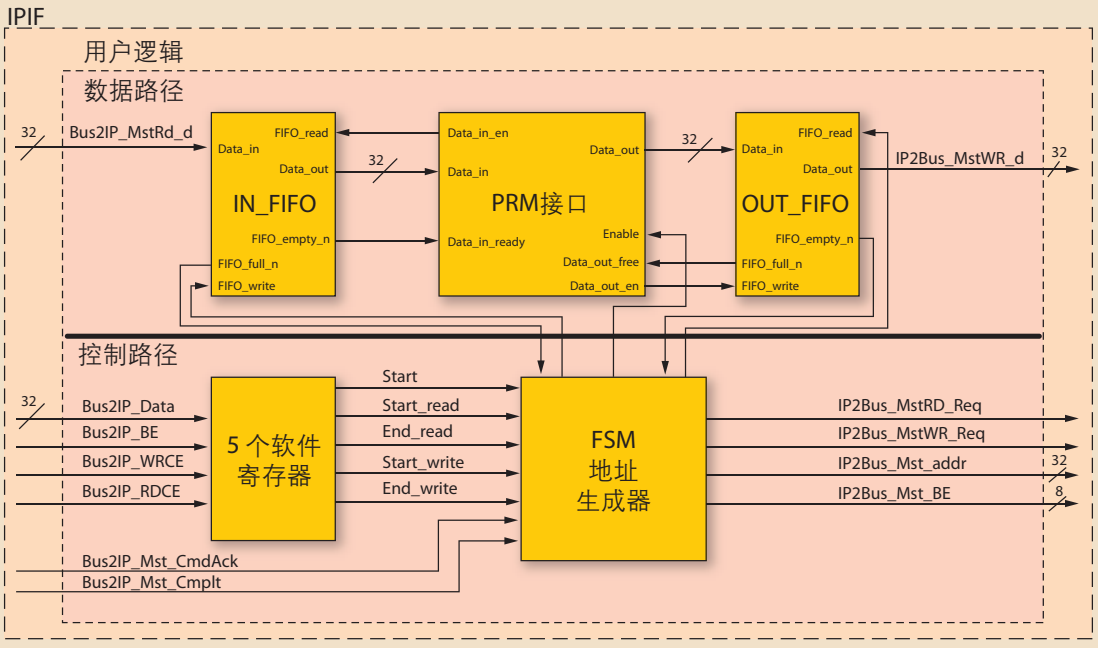


图 4- 总线主外设以处理器元件的方式运行。
PRM 接口包括具备 PRM 组件实例化的动态部分。

有限状态机 (FSM) 的直接数据传输，大幅降低时间。由于不采用软件，所以 MicroBlaze 的寄存器文件中不发生中间数据存储。本 RISC 处理器的“加载 - 存储”架构一直需要占用两个总线传输周期，用于从某个地址加载 CPU 寄存器，然后将寄存器的内容存储到另一个 PLB 连接的设备。由于从 MicroBlaze 到存储器控制器的 DXCL 数据高速缓存链路构成 PLB 的旁路，因此这些“加载 - 存储”周期在时序上不能得到改善。这是因为接收到的数据和发送的计算结果都是逐字一次性处理，没有发挥高速缓存的作用。由此 PRR 外设的活动与 MicroBlaze 主软件的处理脱钩，因而 PRR 数据传输不会导致更多的 Xilkernel 环境切换。但仍然不可避免地出现两个主设备竞争总线访问的情况。

外设的从接口含有四个基于软件驱动的寄存器，可为控制路径提供输入和输出数据集的起始地址和终点地址。另一个软件寄存器为 FSM 设定“起始”位，用于初始化主数据传输周期。完整的数据处理周期的状态经第五个软件寄存器的地址提供给客户端软件。

根据控制路径的 FSM 的状态图可以看出，应该采取让到 PLB 的写入周期优先的策略（图 5）。从 OUT_FIFO 提取数据优先于向 IN_FIFO 写入数据，防止 OUT_FIFO 为满时，阻止 PRM 处理算法。读取或写入外部存储器可交替进行，因为每次只能使用一种总线访问方式。当来自客户端的计算线程的软件复位启动 FSM（图 3）时，第一件事就是从外部存储器读取（状态 READ_REQ）。自此，总线主设备就受状态 STARTED 提供的转换条件所提供的决策逻辑的控制（表 1）。

IN_FIFO	OUT_FIFO	存储器访问	下一状态
不理会	未空	写入	WRITE_REQ
未满	空	读取	READ_REQ
满	空	—	STARTED

表 1 - 写入优先的状态 STARTED 的 FSM 控制决策

Mealy FSM 输出（标记 Exit/）让地址计数器在总线传输完成时递增。这里两个计数器都直接导入到 FSM 代码中。一般情况下我们倾向于将定时器和地址计数器分开实现为仅用 FSM 输出使能的单独时钟进程，以便让计数器的保持小规模的转换逻辑以及尽量避免将多路复用器输入用于计数器状态反馈。对于此点，XST 综合编译器的结果将 RTL 原理图清楚地体现为并行于可加载计数器的 FSM 抽

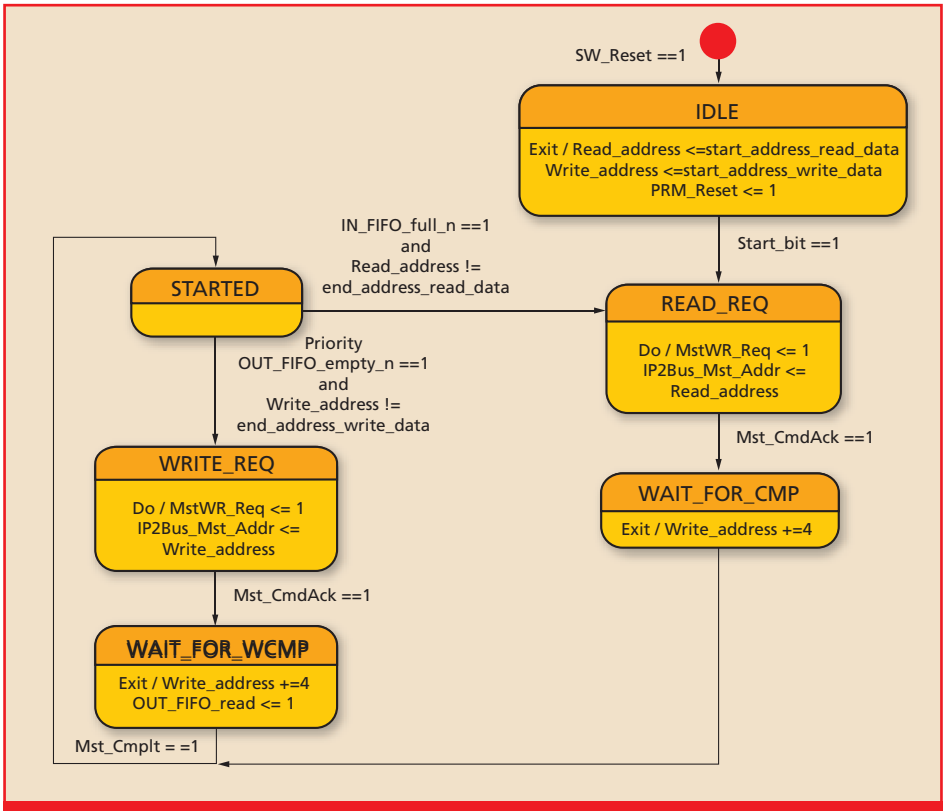


图 5 - 总线主外设的控制路径的状态图。对总线的写入请求设置为优先，以避免 OUT_FIFO 为满时，中断 PRM 算法处理。UML 建模方法：Moore 输出标记为 Do/，Mealy 输出标记为 Exit/。

象，其中的时钟使能输入由预期状态解码逻辑驱动。尽管行为级的 VHDL 编码方式更容易让人理解，使用 FPGA 资源和简单原语也不会影响功能。

用 PLANAHEAD 设置动态部分

FPGA 中静态和动态部分的配置这一设计流程是一个复杂的开发过程，需要用 PlanAhead™ 物理设计约束工具进行多步操作。第一步就是给在 ML505 开发板上实现的由 PetaLinux 驱动的动态重配置平台编写设计流程脚本。[6] 就当前迭代而言，将 PRR 直接集成到外设的用户逻辑中的设计步骤与过去通过添加总线宏和器件控制寄存器 (DCR) 用作 PRM 的 PLB 接口、添加 PLB-DCR 桥接器实现总线宏的做法相比更实际。

下面的代码摘自 PlanAhead 项目的 UCF 文件，说明我们如何使用 AREA_GROUP 约束确定动态部分的大小和位置：

```
INST "
dyn_interface_0/dyn_interface_
0/USER_LOGIC_I/PRR"
AREA_GROUP =
"pblock_dyn_interface_0_USE
R_LOGIC_I_PRR ";
AREA_GROUP "
pblock_dyn_interface_0_USER
_LOGIC_I_PRR "
RANGE=SLICE_X0Y0
:SLICE_X57Y239 ;
AREA_GROUP "
pblock_dyn_interface_0_USER
_LOGIC_I_PRR "
RANGE=RAMB18_X0Y0:RAMB18_
X3
Y95 ;
AREA_GROUP "
pblock_dyn_interface_0_USER
_LOGIC_I_PRR "
RANGE=RAMB36_X0Y0:RAMB36_
X3
Y47 ;
```

内部的部分重配置区域的命名方法通过为其指定实例名 PRR，并将实例名相连（prm_interface.vhd）。对我们希望囊括在所需的 PRR 中的全部 FPGA 资源而言，我们用设置左下方坐标和右上方坐标的方法来划定一个矩形区域。

这种特殊的方法只能覆盖 Slice 和 BRAM，因为可用的 DSP 元件属于专用时钟区域，归多端口存储器控制器 (MPMC) 设计使用（表 2）。

为避免 ISE® 生成的 PRM 网表使用专属资源，我们将综合选项设置为：dsp_utilization_ratio = 0；use_dsp48 = false；iobuf = false。最后，从 FPGA 编辑器观察到静态部分的布局完全与 PRR 分开，PRR 在本例中占用的资源极少（图 6）。

配备图像处理 PRM 的 SOC 客户端

我们使用在 PRM 中实现的 Sobel/ 中值过滤器验证 SoC 客户端的运算能

资源	数量
LUT	55,680
FD_LD	111,360
SLICEL	7,440
SLICEEM	6,480
RAMBFIFO36E1	192

表 2 - 给 SoC 客户端的动态部分分配的资源

力及其 TCP/IP 服务器通信功能（图 7）。我们使用赛灵思系统生成器（System Generator）开发图像处理邻域运算。赛灵思系统生成器让我们享有 Simulink® 仿真和自动 RTL 代码生成的便利。解串器将输入的像素流转换成 3x3 像素阵列，然后依次排列成一个覆盖整个图像的掩膜，为滤波器的并行乘积加总提供输入，或为后续的中值过滤器比较提供输入。[7] 过滤器的输入和输出像素向量的宽度为 4 位，故我们插入一个 PRM 封装器，以多路复用同步 FIFO 提供的 32 位输入向量的 8 个四位元。使用 MATLAB® 脚本，我们将 800 x 600 PNG 图像转换为四位灰度像素，用作 PRM 的输入激励。在过滤器的输出端，8 个四位寄存器顺序写入和级联，将字传输给 OUT-FIFO（图 4）。

表 3 是 SoC 客户端三个运算步骤（接收 PRM 比特文件、重配置 PRR、图像处理序列）的时序测量结果。我们用数字示波器在 XGpio_WriteReg() 调用触发的 GPIO 输出处测量第一次到最后一次数据传输周期，采集接收与图像处理周期。

重配置时间间隔都是一样的，因为没有 Xilkernel 调度事件干扰基于软件的 HWICAP 操作。受 FSM 控制的 HWICAP 操作在没有 MicroBlaze 互动的情况下，可以超过 112 KBps 的重配置速度实现更短的用时，甚至在启用中断的情况下也不例外。

在从代理向 SoC 客户端发送 PRM 的过程中，连接很快中断。因为每传输 100 个字节仅 1 毫秒的延迟，SoC 客户端的通信非常畅通。由于与图像处理周期同步，正常的 Xilkernel 线程导致 PLB 访问竞争，因此 SoC 客户端在典型状态下运行。二值化序列的用时为 600 x 800/100MHz=4.8

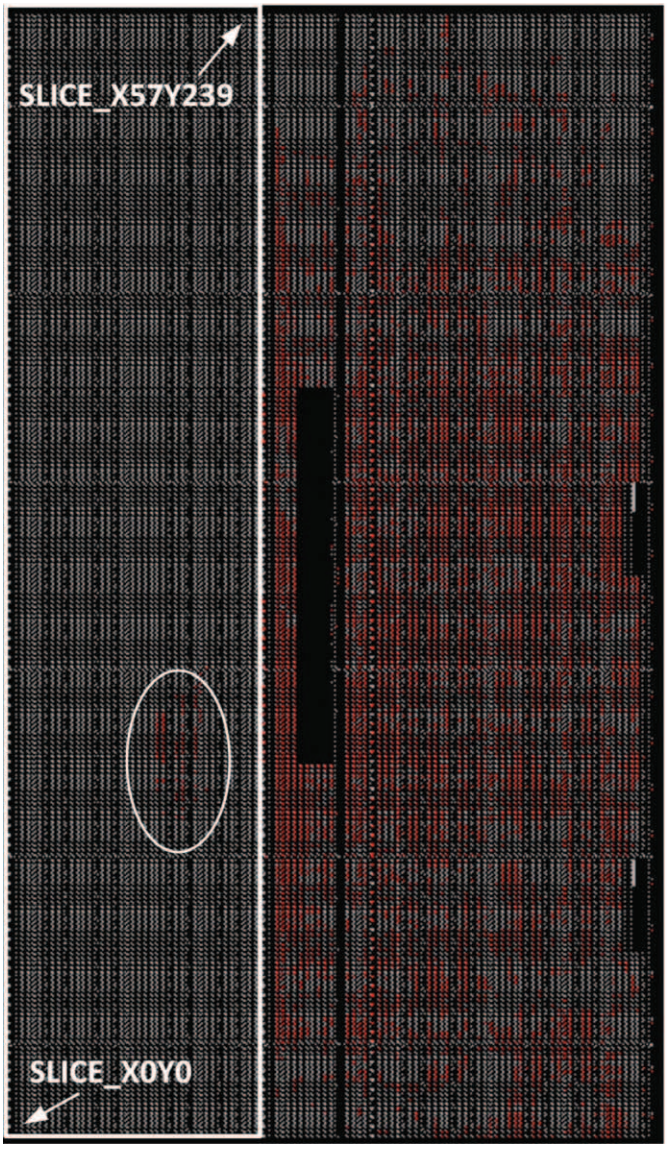


图 6 – 静态部分（右边）和动态部分（左边，白色椭圆形区域）根据 PRR 指定的区域进行资源布局

		用时		
过滤器模块	Slice 数	PRM 接收（秒）	重配置 3.5MB 比特文件（秒）	图像处理（秒）
二值化	3	77	31.25	25.25
腐蚀过滤器 3x3	237	73	31.25	85.93
中值过滤器 3x3	531	73	31.25	77.09
Sobel 过滤器 3x3	479	73	31.25	86.45

表 3 - 时间测量结果，用使能中断重配置。处理器和外设时钟速率为 100MHz

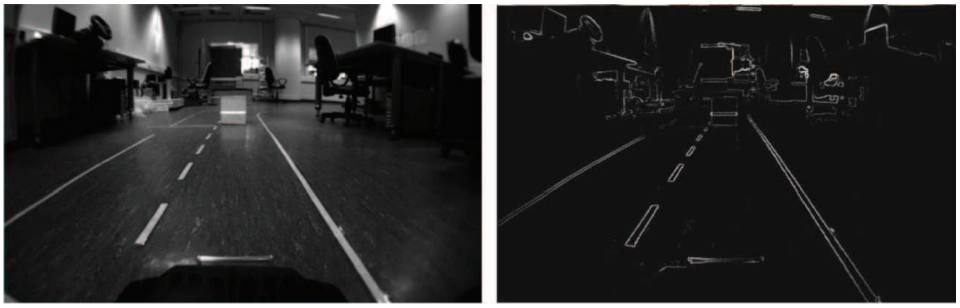


图 7 – 用于边缘检测的 PRM 处理结果。
左图为 PRM 的灰度输入激励图像，
右图则为 带 Sobel/ 中值滤波器组合的 PRM 的响应

毫秒，因为只需要进行一次比较。这个序列嵌套在两次经 PLB 的图像传输中，根据功能性总线仿真的结果，每个字至少需要使用五个时钟周期，故：2 x 5 x 600 x 800/(8 x 100MHz)=6 毫秒。由于所有测量的数据传输值都大于我们先前的预估值，我们需要对由总线读取、FIFO 写入和清空、图像处理流水线和总线写入组成的完整时序链条的构成进行详细分析。

部分重配置的力量

在运算复杂算法时，发挥分布式计算网络的力量是理想的选择。这些网络的流行设计目前仅限于使用 CPU 和 GPU。我们的基于 FPGA 的分布式 SoC 网络架构原型运用 FPGA 的并行信号处理特性来计算复杂算法。

赛灵思部分重配置技术是运用分布在世界各地的 FPGA 共享资源的关键。在我们的架构中，SoC 客户端的静态部分用更新的加速器以自控方式对 FPGA 的动态部分进行重配置。我们必须改进 SoC 客户端，使之能够使用使能中断运行 HWICAP，以实现完全的反应能力。这个方向的第一步是实现 FSM 控制的重配置，不给处理器带来负担。不过我们还需要分析 PLB 传输影响以及 MPMC 瓶颈问题。

为管理 SoC 客户端，使用与 LwIP 链接的 Xilkernel 确保重配置驱动程序、动态部分的总线接口及其它应用的线程保持同步。我们进一步重点分析客户端 - 服务器系统的时序和动态部分的处理周期，以期发现有更高数据吞吐量和可靠通信能力的软件 / RTL 模型配置。

我们的 SoC 客户端的下一阶段的设计必须考虑 AXI4 总线功能。一般来说 PRM 交换可视为与一组软件任务共同执行的额外硬件任务。最后且同样重要的是，我们仍然在优化服务器的软件设计，以期达到更高的客户满意度。

参考资料：

1. 非正式 BOING 维基，《伯克利开放式网络计算（BONIC）常见问题问答：BONIC 介绍》，http://www.boinc-wiki.info/

2. Markus Tervooren，all project stats.com，http://www.allprojectstats.com/

3. S. Kumar、J. Pelzl、G. Pfeiffer、M. Schimmler 和 C. Paar，《用 COPACOBANA 破解加密，成本低廉的并行代码破解器，或如何用 8980 欧元破解 DES》，http://www.copacobana.org/paper/CHES2006_copacobana_slides.pdf

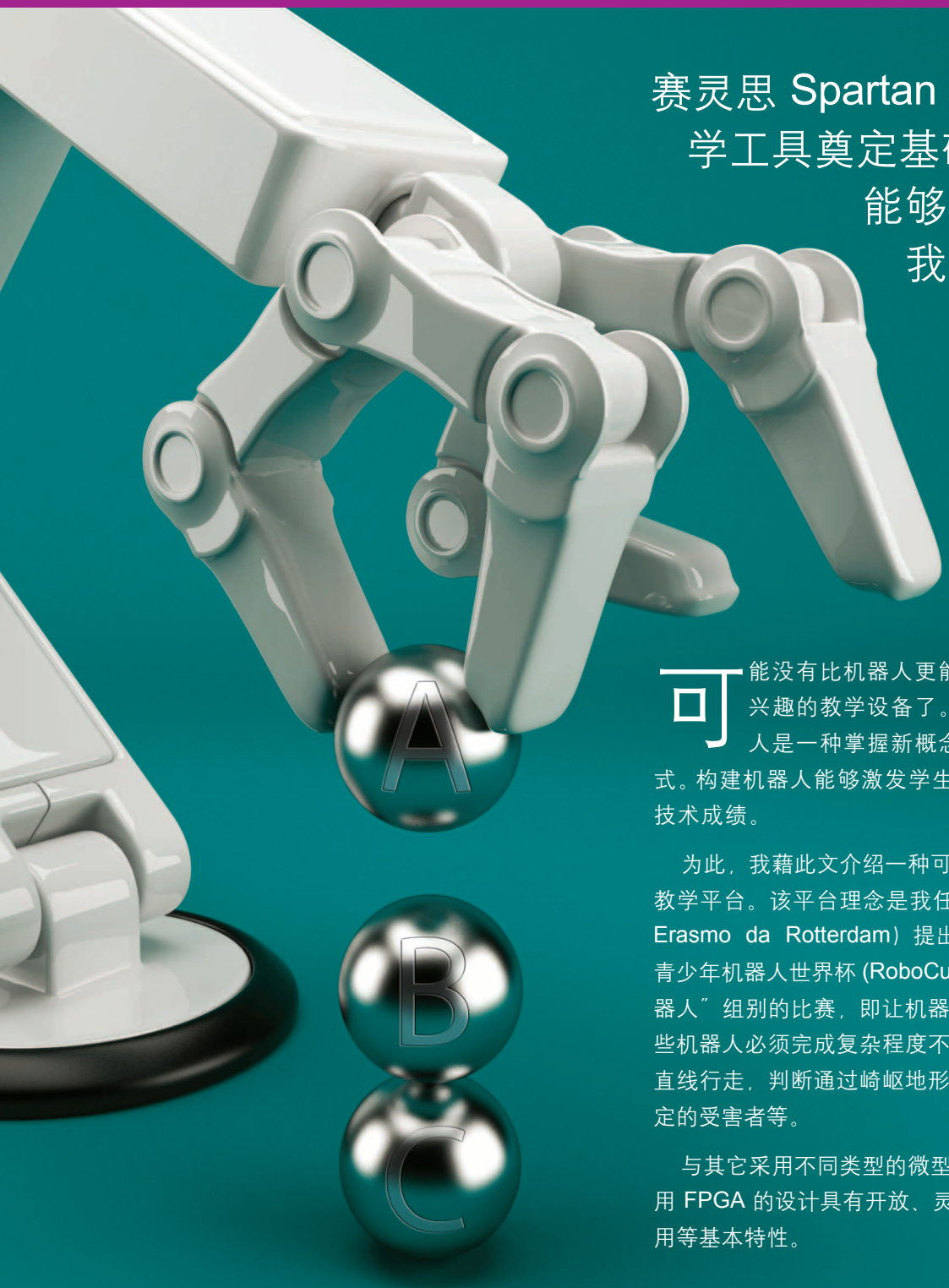
4. Frank Opitz，《开发基于 FPGA 的分布式计算平台》，硕士论文，汉堡应用技术大学，2001 年，http://opus.haw-hamburg.de/volltexte/2012/1450/pdf/Masterarbeit_Frank_Opitz.pdf

5. Ivo Bolsens，《现代 FPGA 编程》，http://www.xilinx.com/univ/mpsoc2006keynote.pdf

6. Armin Jeyrani Mamegani，《SoC-FPGA 部分和动态重配置实现方法及其评估》，硕士论文，汉堡应用技术大学，2010 年，http://opus.haw-hamburg.de/volltexte/2010/1083/pdf/MA_A_Jeyrani.pdf。

7. Edris Sahak，《基于 SoC 的图像处理流水线的部分重配置》，学士论文，汉堡应用技术大学，2011 年，http://opus.haw-hamburg.de/volltexte/2011/1420/pdf/BA_E_Sahak.pdf。

FPGA 为高中教学机器人提供灵活的平台



赛灵思 Spartan FPGA 为强大的教学工具奠定基础。这种教学工具能够根据学生的需要自我改造。

作者：Giulio Vitale
教授兼 FPGA 设计顾问
意大利巴拉塔 ITCS Erasmo da Rotterdam
gvitale@tiscali.it

可能没有比机器人更能吸引初中生及高中生对科学技术感兴趣的教学设备了。对初中生和高中生来说，摆弄机器人是一种掌握新概念，亲眼见识技术使用情况的实践方式。构建机器人能够激发学生们克服智力难题，取得优异的科学、技术成绩。

为此，我藉此文介绍一种可供教师用于技校的机器人教学活动的教学平台。该平台理念是我任教的高中（意大利米兰附近的 ITCS Erasmo da Rotterdam）提出来的。当时只是为了入围 RoboCup 青少年机器人世界杯 (RoboCup Junior)。我们队参加的是“救援机器人”组别的比赛，即让机器在重现的灾害场景中发现受害者。这些机器人必须完成复杂程度不同的多重任务，比如在平坦路面上沿直线行走，判断通过崎岖地形上各障碍物的路径，以及拯救部分指定的受害者等。

与其它采用不同类型的微型计算机的普通传播平台相比，我们采用 FPGA 的设计具有开放、灵活、可不断发展演进，以及可重复使用等基本特性。

我们使用 Digilent Nexys2 教学开发板，在赛灵思 Spartan®-3E 器件的基础上制作了该机器人的 2011 版。在撰写本文时，我们正在把这个版本移植到 Spartan-6 FPGA 上，以便参加将在 2012 年 4 月举行的下一届意大利青少年机器人世界杯比赛。按计划，下一版本（2013 版）将运行于 Zynq ™ -7000 可扩展处理平台器件上。

用这种方法，我们设计出的一种救援机器人能够年复一年不断发展演进，从第一种原型发展到了现有的 Nessie 2011 版本（Nessie 由 Nexys 和尼斯湖水怪（Loch Ness Monster）而得名，尼斯湖水怪的长脖子与我们的机器人相似）。FPGA 具有高度的灵活性，可以在不改变其基本物理结构以及保持相同的设计基础架构同时，让机器人的架构随学生知识的增长而重构。

挑战

ITCS Erasmo da Rotterdam 是一所位于意大利北部米兰近郊的技校，学生参差不齐，难以教导他们深入学习科技知识。

从四年前开始，我决定创办一个开放空间，后来我取名为“机器人教学永久实验室”，在那里学生们可以采用与标准课堂授课完全不同的方法开展实验。在此，他们可以在互动环境中接触技术学科，探讨一些异乎寻常的主题内容，选修一些自己感兴趣的课程，安排好自己工作并从实践中直接得到反馈。换句话说，他们根据由来已久且为人熟知的“情景认知”模型 [1] 在“积极学习空间”中开展实验。在这里，同学们不仅可以相互协作或同辅导员合作，同时还可在达成共识的基础上追求共同的目标。

在这个学习空间中，学生扮演掌握多种解决方案的“认知学徒”。而教师则充当“专家”角色，负责提出实际任务和策略的具体流程，让学生们独立尝试，需要时才予以辅导。

为不同学科的汇聚和知识的交流提供肥沃的温床，机器人是自然之选。我们认为参加青少年机器人世界杯比赛的乐趣强烈地刺激了学生对科技实验的积极参与。

解决方案

我认识到，要让这种方法有效，我必须提议数字电子学和信息学等普通课程涵盖的课题，但主要针对哪些比学生们能够独立解决的应用更复杂的应用。这样他们就需要分工合作，或者需要得到能够提出正确模型的资深教师的协助。

我知道需要制作什么，但我不知道怎么制作。一切都必须在实验室中提出和开发出来，由学生共同探讨设计，寻找解决方案。

在经过一番深思熟虑之后，我得出一个结论：最可行的就是采用一种基于某种灵活的平台（如 FPGA），而不是标准微型计算机上的解决方案。这是因为 FPGA 是唯一能够满足特性要求且能够跟上实验活动的动态和演变范围的器件。

我最初选择的是基于 Spartan-3E 的教学板卡，因为它能够提供我们一直寻求的必要特性，即开放性、灵活性、演进能力、硬件的重复利用性以及性能的丰富性。

- **开放性**。由于学生必须积极参与整个设计流程，从传感器接口到 CPU，再从 CPU 到激励器。
- **灵活性**。因为系统的整个架构以及器件的性质和类型不应事先确定，必须从创造性学习环境所激发的研究过程中产生。
- **演进能力**。因为在每次青少年机器人世界杯结束后，学生都必须认识到他们作品的缺陷，了解如何进行适当修改，试图找出更先进的解决方案。该系统必须与学生的专业知识保持同步发展。
- **可重用性**。以避免硬件和学校经费不必要的浪费。
- **低成本、高性能**。我们必须控制没有得到完全定义但需要高度并行运行的大量器件和外设。该 CPU 的

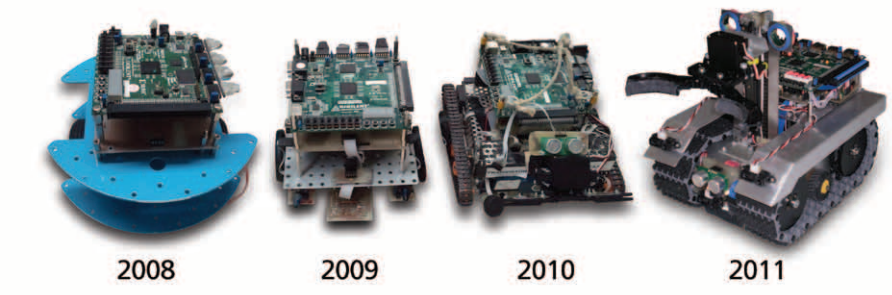


图 1 - 借助 FPGA 的高灵活性，同一平台可以随学生理解的深入不断演进，并能根据需要重用硬件和重新设计。
照片展示了 Nessie 从 2008 年到 2011 年的演进情况。

功能应非常强大，但架构相对简单，易于接口。

NESSIE 2012：方框图及描述

我们用基于板载 Spartan 3E-1200 的 Diligent 板卡来实现我们的目标，这成为四年期项目开发的统一主线。从图 1 可以看到，学生们设计的救援机器人发生了明显的变化。2008 年它还只是勉强能动，到 2011 年它让我们从 65 支参赛团队中脱颖而出，顺利杀入 15 强，成功进入决赛。

学生的专业知识年复一年地增长，为救援机器人的进一步改进奠定了基础。这些改进我们准备用在参加今年 4 月举办的青少年机器人世界杯大赛的救援机器人上。

首先我们从 Spartan-3E 升级到 Spartan-6 系列，将标准处理器本地总线的总线基础架构升级为 AXI4 接口。其次，我们对部分用于跟踪基准线的关键传感器进行了修改，重新设计了电机接口，将用于自动速度控制的 PID 算法直接移植到了 FPGA 架构中。

图 2 是目前设计的完整的系统方框图。从这个图上可以清楚地看到丰富的教学课题，便于教员在数字控制系统课程中讲授。同样显而易见的是，与采用标准微型计算机制作的机器人平台相比，从性能方面来说，该系统具有高度的并行性。

此外，机器人实验室中开展的活动对我们学校普通教学课程带了了良好影响。FPGA 已成为让技术理论迅速有效地变为现实的工具，从而激发学生对涉及的课题产生出浓厚的兴趣。

管理 NESSIE 的行走问题

在有 Spartan-6 系列提供丰富的可用资源的情况下，我面临的问题是如何将在连续系统上分析得到的 PID 控制方法变为现实，运用到数字系统中，让学生有机会亲手解决方程组的数字化问题，并立即将其对应到数字电子电路的元件。

我从典型的 PID 方程式入手：

$$a(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau + K_D \frac{de(t)}{dt}$$

将其转换成另一种典型的有限差分算法：

$$a(n) = K_P \left[e(n) + \frac{T_C}{T_I} \sum_{j=0}^{n} e(j) + \frac{T_D}{T_C} (e(n) - e(n-1)) \right]$$

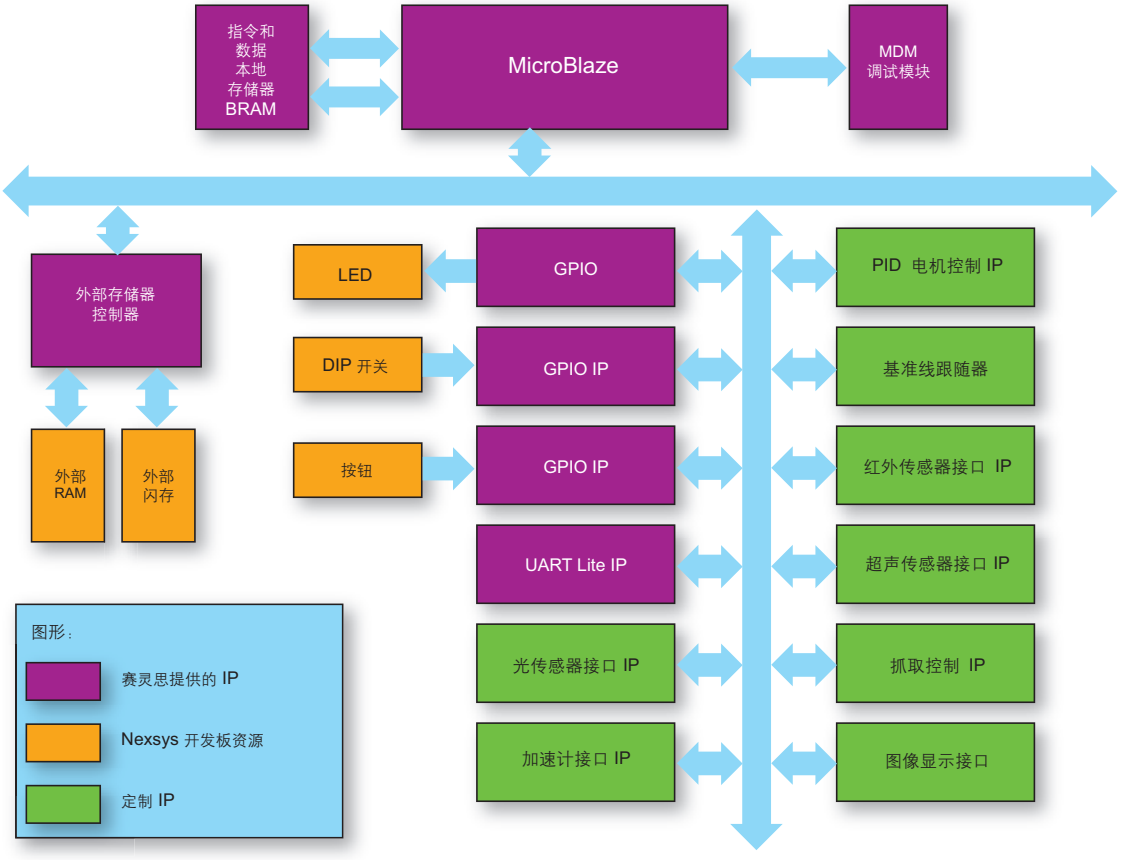


图 2 - 显示了目前 Spartan-6 重配置设计 Nessie 2012 版的方框图。它将参加今年春季举办的青少年机器人世界杯比赛。

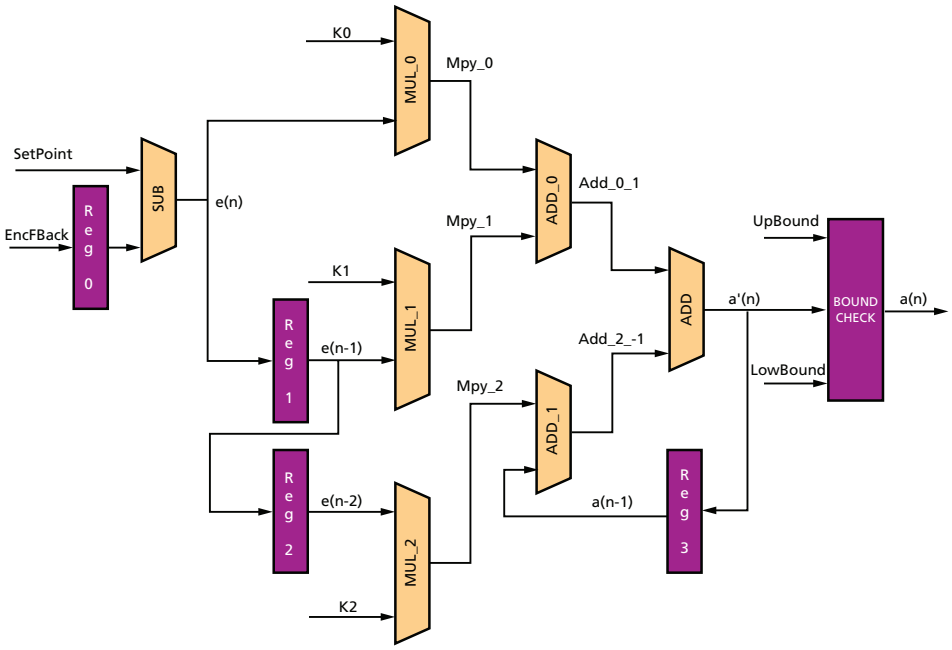


图 3 - 控制机器人两个电机速度的 PID 算法的 RTL 草图

其中，K_p 为比例放大率，T_I 和 T_D 为积分运算和导数运算的时间常数，T_C 为采样周期。根据 W. Zhao 等撰写的《用于小型机器人的闭环控制系统的 FPGA 实现方案》[2] 一文中提出的建议，我把该模型转换为如下的非常简单的迭代算法：

$$\Delta a(n)=a(n)-a(n-1)$$

其中

Δa(n) 可通过下列算式得出：

$$\Delta a(n)=K_0 \cdot e(n)+K_1 \cdot e(n-1)+K_2 \cdot e(n-2)$$

迭代的步骤为：

$$a(n)=a(n-1)+\Delta a(n)$$

K_i 系数可通过下列算式得出：

$$K_0 = K_P \cdot \left(1+\frac{T_C}{T_I}+\frac{T_D}{T_C}\right)$$

$$K_1 = -K_P \cdot \left(1+2\frac{T_D}{T_C}\right)$$

$$K_2 = K_P \cdot \frac{T_D}{T_C}$$

其中 PID 参数可以采用 Ziegler Nichols 方法实证性地微调。

图 3 大致绘出了 FPGA 实现方案的 PID RTL 模型。图 4 是我们为 Nessie 参加 2012 年青少年机器人世界杯大赛设计的完整行走控制 IP。

从老师的观点出发，这种方法极富成效，因为它采用线性转换，且其从概念到立即实现物理系统这个相关转化容易做到。这个过程鼓励学生尝试更多实验，深入掌握和学习这个系统。

如何让 NESSIE 的眼睛感知光线

作为一个救援机器人，Nessie 的任务是在人为重现的灾难场景中发现受害者。机器人在崎岖地形上行进、越过障碍物和瓦砾、搜寻待拯救的受害者，都需要用视觉跟随白色背景上的一道黑线。

有效的跟踪系统是决定青少年机器人世界杯大赛中所取成绩的关键。成功完成有引导的赛段是机器人有足够的时间完成其它比赛项目的必备起点，包括避开障碍物和瓦砾，以及拯救受害者等。

我们将通过使用内置一个光积分与吸持电路的 128x1 线性传感器阵列（即 Taos 1401R-LH）来解决 Nessie 的视觉功能问题。

该线性阵列由一个集成有 128 个二极管的模块和一个使用两个电容对二极管产生的电荷进行积分和保持的模拟电路组成。其中一个电容负责采集电流，另一个则负责复制回来并在扫描和采集下一个电荷前予以保持。该传感器实际上同时执行两项操作：完成积分、采集新的测量值，同时以扫描的方式读取上一个周期中累计的电荷。一次扫描周期中积分的电荷扫描结束时即被转移到保持电容中。

每个周期开始于“开始积分”（Start Integration）信号的上升沿。在第一个时钟周期内，该信号会断开一个内部开关，以隔离保持电容，同时删除存储在积分电容器中的电荷。所有 128 个二极管同时并行执行该过程。随后周期进行新的积分过程，在这个过程中传感器读取新的亮度数据的全部 128 个值。与此同时，在每个上升沿，保持电容器中存储的电荷输出到输出放大器上，在模拟输出引脚上读出跨越放大器两端的电压。该电压被读取到移位寄存器中。通过这种方法，就可以一边读取上一次由光照所积累的电荷而产生的电压，一边在积分电容器中累计当前采集的值。

在此，FPGA 再度证明其价值。释放 CPU 负载，刷新积累的电流，并保存转换结果值以备进一步处理。图 5 即是我们为此目的设计的 IP 的图示。它还说明了“双缓存”机制的作用，即使用 Spartan 器件上提供的内部 BRAM，可以通过处理采集到的上一帧来覆盖新帧的采集。通过这种方法，

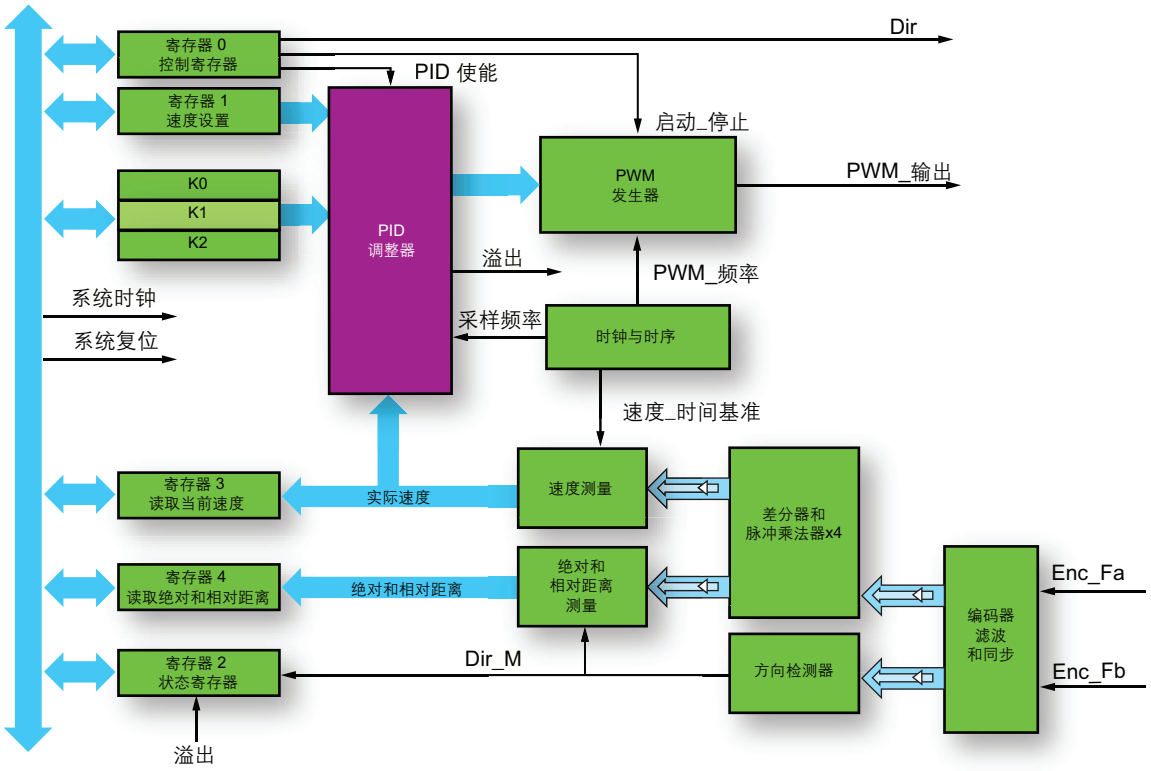


图 4 - 我们专为 Nessie 2012 设计的自动电机控制 IP 的完整方框图

我们可以加快图像处理速度，提取出机器人导航所需的全部信息。

高中水平的数字处理

鉴于本机器人活动的教学性质，我也使用本设计向学生讲授部分简单的数字图像处理元件。

从图 2 中可以看到，我们给设计加装了一个小型图像显示器，以便显示 Taos 传感器看到的内容，同时了解如何进行处理才能提取出用于引导 Nessie 完成其功能所必需的信息。图 6 是一帧中采集的原始图像。

在设计 Nessie 时，学生们必须找出一种操控扫描到的线的方法，从而在可能存在的瓦砾和废弃物之间识别黑线的位置，并定义控制两个电机速度的算法，让机器人正确地跟踪黑线行进。

通过这种方法，学生掌握了如何处理像素这样的简单问题，学习到不同的数学运算在实现这项功能上的效果。

具体来说，为了判断黑线的位置，他们会找到一些基本的运算方法：设定阈值、椒盐滤波、边缘探测和线段分割。

扫描到的线需要按不同的光强度和长度分段组合，而且所有这些对象必须正确排序，才能识别方向，引导机器人前行。这就需要在硬件 PID 环路之外使用软件实现第二个控制环路，用于优化跟踪基准线运动，根据传感器轴线和黑线之间的角度以及侧面白色区域的相对大小调整两个电机的相对速度。

激励工具

该系统的复杂程度对我的学生的年龄是适合的，经证明是一种引导他们接触更高难度的内容的良好方式，比如二维图像处理。这些初始难度的问题易于入门，能激励学生不断提升自己的技术和科学知识，通过解决更为复杂的问题掌握新的技能。根据我在高中教学的长期经验，我可以放心地说，有了这样的经验，学生会更有兴趣和动力继续技术学习，能更有把握地进入高等学府深造以及在所从事的专业领域就业。

在我讲授采用这种灵活的 FPGA 平台的机器人技术的三年中，我发现我的学生对“实时”计算机系统的内在构造有了更深入的认知。他们还学习到如何设计出更具原创性的内部外设，如何增强团队合作能力，如何增强独立面对问题并找到正确的解决方案的能力。

鉴于本项目的教育目的，我们在 ITCS Erasmo da Rotterdam 的团队正在鼓励学生用他们在 Nessie 上所学的知识为通过意大利的州考试，顺利毕业

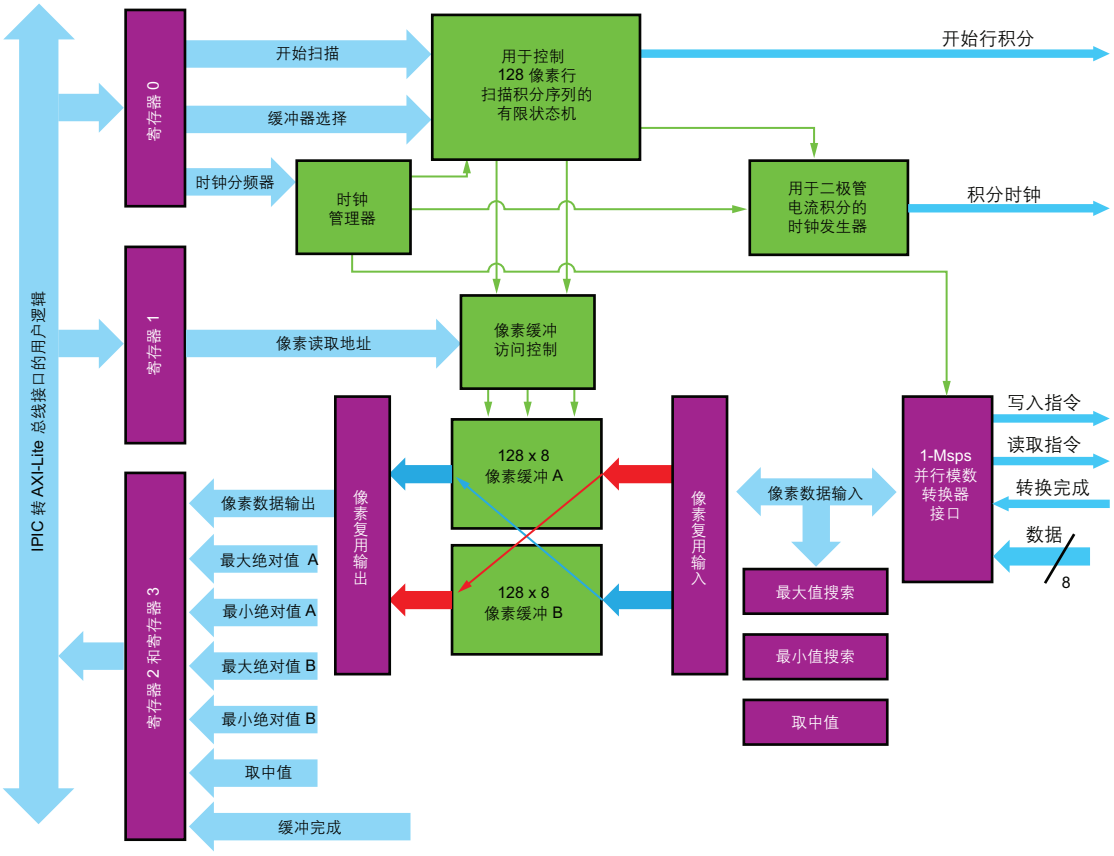


图 5 - MicroBlaze® 和 128x1 Taos 1401 线性阵列光传感器之间的接口方框图

铺平了道路。该考试有一项单独的内容，就是学生必须论述自己的研究道路。Nessie 事实上激励部分学生以优异的成绩取得电子技术学位。

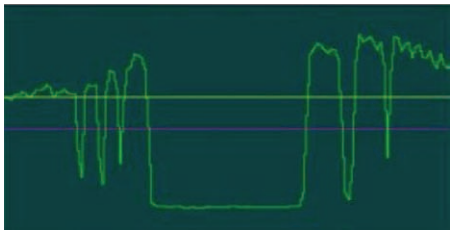


图 6 - 采集到白背景下黑线及附近的瓦砾的原始像素线，
尚未进行阈值处理和椒盐过滤

参考资料：

1. J. S. Brown，A. Collins 和 P. Duguid，《情景认知和学习文化》，教育研究者，18 卷 1 号（1989 年 1 月到 2 月），
http://people.ucsc.edu/~gwells/Files/Courses_Folder/ED%20261%20Papers/Situated%20Cognition.pdf

2.W. Zhao、B. H. Kim、A. C. Larson 和 R. M. Voyles，《用于小型机器人的闭环控制系统的 FPGA 实现方案》，第 12 届国际先进机器人大会会议记录（ICAR 05），
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.81.8719。

以 FPGA 为起点的智能高速交易平台

运用新框架加速超低时延金融系统的应用开发

作者：Rafeh Hulays 博士
业务开发部副总裁
AdvancedIO Systems 公司
rhulays@advancedio.com

自电子交易问世以来，对速度的追逐无止境，时刻要求搭建速度最快、最智能的交易平台。智能和高速就意味着金钱。能够率先发现交易机会并在其上完成交易的交易平台将占上风。响应时间已经从“秒”一路降低到“毫秒”、“微秒”。要实现微秒和亚微秒响应时间，凭借传统的软件架构或简单的硬件架构几无可能，正是这一事实促使在超低时延系统中引入 FPGA 技术。但是，FPGA 编程要求使用 HDL 机器语言开发交易策略并将现有交易策略（主要用 C 语言和 C++ 语言编写）移植到 HDL 机器语言中。这与使用标准的 C 和 C++ 编程在技能上有根本的不同，同时需要增加人员、工具和时间。

有现成的 C-to-HDL 编辑器工具可用，可将现有的 C 代码移植到硬件描述语言 (HDL)，比用 HDL 从头开发代码速度快。但是，负责移植代码工作的人员必须熟悉硬件编程，或者必须遵循具体指南，才能生成可接受的代码。即便这样，生成的代码仍然可能会比用 HDL 直接开发的代码晦涩低效。

为了降低在 FPGA 以太网卡上直接开发 HDL 代码所带来的风险，同时节省开发时间，AdvancedIO 公司率先尝试将 FPGA 框架用于 10G 以太网 (10GE) 通信。我们的 expressXG 开发框架工具套件为保证金融业务的快速部署提供了必要的基础设施，可无缝移植到最新一代 FPGA 卡上。该工具套件可集成并优化应用开发所需的所有必要的核心功能，能够最大程度地降低第三方许可成本，让开发周期缩短数月。开发团队可随后将 C-to-HDL 编译器工具集成在框架中，从而为开发应用的不同部分提供多重选择。

实际上两种方法都在项目中有一席之地。在优先考虑效率和代码紧凑度，或组件在多个项目中使用的情况下，最好使用 expressXG 框架直接开发 HDL 组件。如果着重考虑产品上市时间和定制化，或者要求以 C 语言编写代码的情况下，最好使用 C-to-HDL 编译器。这种方法可以克服众多金融从业人员对 FPGA 的畏惧，同时不会影响性能。

算法交易概览

说到证券、衍生品、期货及其它金融工具的交易，总会让人联想起塞满几百人，叫嚷声此起彼伏的交易大厅。人们或漫无目的地转悠，或紧盯计算机屏幕。现实是，当今美国有 70% 以上的交易是由运行在计算机服务器上的算法完成的。在金融业，速度就是一切。哪家公司能够率先发现机遇并把握机遇，就能够赢得丰厚的利润。对某些交易策略而言，率先进入队列就能赢得交易。这也就毫不奇怪众多公司竭尽所能地确保自身相对竞争对手的优势，哪怕是仅仅 1 微秒。

在算法交易系统 (ATS) 中，由运行在高性能计算机上的算法处理交易，制定决策。这种系统运用多种技术来最大程度的降低时延，赢得竞争优势。具体包括与各个交易所共址、使用最短路径网络，以及使用 10G E 来降低时延这一日益风靡的做法。为在竞争中赢得优势，主要的交易公司同时采用上述所有技术。

证券市场上的交易仅限于经纪人 - 交易员和属于交易所成员的市场庄家机构之间。部分这类机构为其他交易公司提供服务，使他们能够通过他们的帐户使用“电子式专属线路下单 (DMA)”给交易所下订单。提供“电子式专属线路下单”的机构必须建立交易前风险控制制度，以限制财务风险，

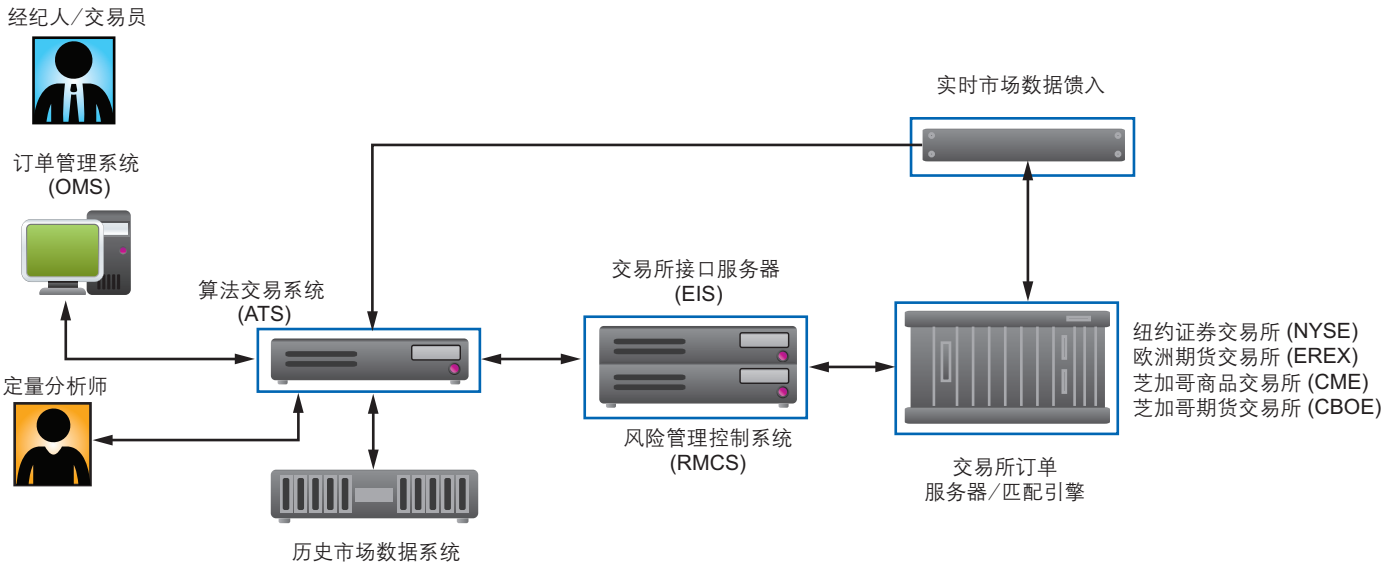


图 1 - 交易系统的简图。
蓝色框标出了 FPGA 将能够显著改善其性能的组件。

FPGA 能够提供并行处理能力，这是通用处理器无法匹敌的。设计人员都在努力将尽可能多的功能集成到 FPGA 中。

满足合规性要求。因此，他们必须采用配备有软件的服务器，检查所有经手他们的账户的交易。这种检查必须以线速进行，因为任何延迟都会让他们及其客户处于不利境地。现有的基于软件的交易前风险管理平台需要数十微秒才能完成金融交易上要求的策略检查。很明显这对当今的交易员来说速度不能满足要求。

图 1 显示了简化交易系统的各个组件。实际上，ATS 可以更多的组件，比如用于执行更复杂算法的服务器或计算云，这里我们不作讨论。交易系统可以直接连接到交易所，也可以通过一个或多个交易所接口服务器、电子通信网络或其它方式连接到交易所。另外，对稳健可靠的交易基础设施而言，防火墙和入侵检测系统也是关键的组件。

基于软件的解决方案的局限性

ATS 必须能够接收多种市场馈入，解码使用的不同协议（如 FIXFAST、ITCH 和 OUCH 等多种名字各异的协议），过滤需要的交易代码并将其发送到预定义的“篮子”中，供交易算法 (TA) 分析。一旦交易算法判定必须执行某项交易，就向订单管理系统 (OMS) 发出交易请求，由订单管理系统与交易所接口服务器或交易所本身通信，下订单、接收交易确认。ATS 服务器还能用极其精确的时戳给输入的数据加戳，然后将数据送去存档。加盖时戳和市场数据采集既可在同一服务器（让数据经内部网络进入数据服务器）完成，也可两个独立的服务器上执行。

ATS 能够以超过 6Gbps 的总突发数据速率从多个交易所接收数据。近期纳斯达克 (Nasdaq) 宣称已准备为自己的数据中心客户提供 40Gbps 的连接，正等待监管部门的批准。数据速率的快速攀升给系统处理器带来了巨大压力，同时也从重要的交易数据分析和决策任务上分散了处理能力。根据使用商用 10GE 网络接口卡对文件服务器测试的结果可以看出，在通过服务器自身的本地堆栈进行 TCP 传输时，仅处理 IP 协议就几乎完全占用了 2.2GHz 的处理器，且实现的数据速率不足 5Gbps。为应对这种困局，出现了一种使用一个称为 TCP 卸载引擎 (TOE) 的硬件为网络协议栈提速的方法。该方法能在一定程度上改善响应时间。

即便在网卡上采用了 TOE，到达主处理器的数据量仍然相当庞大，足以导致不可接受的延迟。为降低时延，有必要先在网卡上过滤相关数据，然后再传输给主系统处理器。虽然所有的工作负荷都在 FPGA 层面完成，但响应时间明显加快。另外 FPGA 可提供与负荷水平无关的相同响应时间。这对常规的处理器是无法实现的。在处理器轻载时，响应时间可以预测，但在重载时，响应时间会延长，直至无法预测。

目前处理器和以太网卡之间通过 PCI Express® 总线来通信。理论上 8 信道的 PCI Express Gen 2 总线能提供 4Gbps 的峰值吞吐量。但 PCI Express 受器件驱动程序和操作系统中断处理内

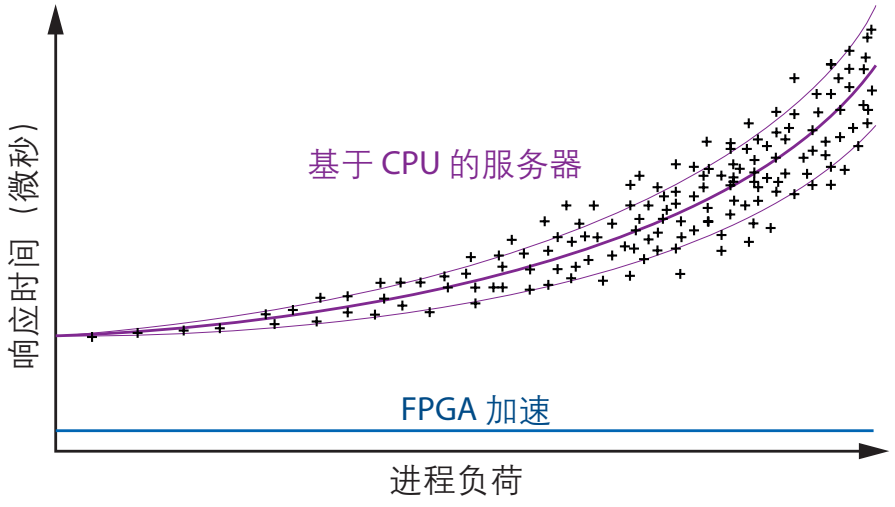


图 2 - 运行在基于 CPU 的服务器上的应用和运行在基于 FPGA 以太网卡（FPGA 加速）上的应用的响应时间

在的时延的牵制。避免通过 PCI Express 总线与主机处理器相连，在网卡上执行金融计算，有着明显优势。

FPGA 技术是金融应用的理想选择，因为它能够实时处理海量数据，且时延低、一致性好。简言之，FPGA 能够提供并行处理能力，这是通用处理器无法匹敌的。由于 FPGA 能够给交易系统带来更强劲的性能和极快的速度，设计人员都在努力将尽可能多的功能（比如与交易所通信的 OMS 和部分算法）集成到 FPGA 中，以降低响应时间。图 2 对基于 FPGA 和 CPU 的系统的响应时间进行了比较。FPGA 不仅速度明显提高，而且性能随负荷的增加保持稳定，相比之下基于 CPU 的解决方案响应时间随负荷增大明显延长。

FPGA 技术的优势

FPGA 技术能够提供最佳解决方案，轻松应对高带宽、实时、高计算强度金融交易应用所面临的各种难题，这是其他技术无法匹敌的。由 FPGA 在以太网卡上执行交易的解决方案具有多重优势：

- 在贴近网络物理接口的 FPGA 上执行交易可避免主机总线、主机处理器和操作系统带来的时延。这样可以显著改善交易的响应时间。
- FPGA 提供线速性能，能够瞬时执行算法中用于发现并把握交易机会的部分，不给他人察觉机遇的机会。
- FPGA 在运行过程中可重新编程，便于修改参数、更新算法，保持领先于竞争对手。
- FPGA 特别擅长并行处理，能够同时执行多个交易。

超低时延金融应用难以实现的原因在于它们必须在极高带宽下以线速运行，必须执行复杂的算法。要实现卓越的高性能，必须针对应用设计专门的架构。这就必须考虑如何在高计算强度的算法和满足交易的极快响应速度要求之间进行取舍，实现最佳平衡。设计人员必须想尽一切办法消除瓶颈，最大限度地降低通信时延和处理时延。为避免与主机系统总线有关的时延，应选用功能强大的 FPGA 来实现交易算法的关键部分，同时匹配大量的 SRAM 和 SDRAM 存储器端口和低时延通信端口。

与为单核或多核处理器开发软件相比，使用线程的硬件模块编程 FPGA 一般情况下工作量更大，而且要求具备专业技能。另外，还需要投入大量精力和资金学习实现 FPGA 解决方案的有案可循和无案可循的微妙之处，尤其对极高速解决方案而言更是如此。这些特点可能会增大 FPGA 的认知风险，延长设计时间，导致部分项目经理和开发团队趋利避害，转而寻求次优的软件实现方案。另外现有的交易算法几乎都是清一色用 C 和 C++ 编写，将它们移植到 HDL 代码中不是一件能一蹴而就的小事。

目前已有多种简化 FPGA 编程工作的方法，比如嵌入式硬核、软件内核和将 C 代码移植到 HDL 语言中的各种工具等。虽然每种方法都有自己定位，存在一定的性能缺陷，但每种方案都能加快应用的部署。C-to-HDL 工具已成为简化 FPGA 应用开发的明确选择。但是这些工具生成的代码相当晦涩，难以优化。简言之，在目前没有什么能够取代直接 HDL 编程。

直接 HDL 编程与 C-TO-HDL 工具的对比

乔治华盛顿大学开展了一项研究[1,2]，对高性能可重配置计算机中用于为 FPGA 生成 HDL 代码的高级语言 (HLL) 工具进行了评估，并制定了相应标准用于衡量各种工具相对于直接 HDL 编程的效率和生产率。研究人员选择了四种工作负荷，并邀请拥有不同经验水平的用户使用各种工具来实现代码。结果显示，各种 HLL-to-HDL 工具无一例外地缩短了开发时间，其中有的工具使开发时间缩短了高达 61%。但结果也显示得到的代码无一例外地比直接用 HDL 开发的代码低效，占用面积增加了 36% 之多，频率下降达 50%。另外研究还发现在使用部分工具时，吞吐量显著下降。[2] 而且这些工具生成的代码相当晦涩，难以进行 HDL 层面的调试。

不过这里应该说明的是测试中使用的四种设计相对简单。某些金融算法和应用要复杂得多，用 HDL 直接编

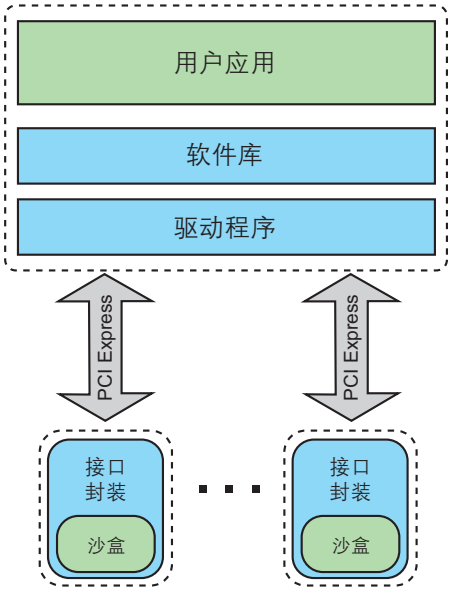


图 3 - 完整的系统级 FPGA 开发框架解决方案，包括经优化的低时延 Linux 驱动程序和 API 以及图 4 所示的经优化的控制器

程方法来实现也更艰难。我们预计随着更复杂的算法用 FPGA 来实现，使用 HLL-to-HDL 工具应该能够进一步节省时间。已有现成的 C 语言算法，需要移植到 HDL 中的情况尤为如此，但需要先在 C 语言中编写，然后移植的情况却并非如此。

需要注意的是，C 语言和 HDL 有着根本性的差异，不是用 C 语言编写的一切内容都能够正确无误地转换为 HDL。表 1 所示的是厂商目前使用的几种 C 语言变体或子集，现在正在尝试在 OpenGL 上实现标准化。OpenGL 是一种用于新型处理器跨平台并行编程的开放标准，目前已在 FPGA 上使用。[3]

用 HDL 编写的 FPGA 开发框架

另一种简化 FPGA 开发工作并缩短开发周期的方法是使用用 HDL 直接编写的、对时延和性能进行了高度优化的框架（如图 3 和图 4 所示）。这种框架是以太网协议与接口、存储器控制器以及主机架构接口的深度抽象，故能减少实际人员为实现定制算法付出的精力和时间。这样开发人员能够集中精力进行应用开发和集成，不必担心如何让各种外部接口在 FPGA 卡上工作。合适的开发框架应保证应用在 FPGA 器件系列直接的可移植性以及在同一系列 FPGA 卡之间的可移植性。这样可以显著降低未来移植或升级的成本。

我们已在 AdvancedIO Systems 公司提供的各系列高性能 10GE 卡上实现并验证了我们的开发框架。这些 10GE 卡广泛用于国防、金融和电信行业等多种市场应用领域。

V5022 卡是专为金融交易优化的产品，采用赛灵思 Virtex®-6 HXT FPGA，拥有能够实现大容量、超低时延交易解决方案的应用架构所需的全部必要组件。Virtex-6 HXT 系列可为实现复杂算法提供大量逻辑资源。它拥有 2 个高达 8GB 的独立 Bank、533MHz DDR3 SDRAM 和 4 个高达 144Mb 的独立 Bank、350MHz QDRII+SRAM，是需要缓冲或超高速查找表的高级算法的理想选择。V5022 卡有四个 10GE 端口，能够大幅降低从光缆到 FPGA 器件内部的 MAC 接口（L2）的时延。

V5022 支持 PCI Express Gen2 主机接口，采用卡间高速端口确保系统中各卡之间的超高速通信，无需使用主机系统总线，可进一步降低时延。这为实现更加复杂的交易算法提供了更强大的高速处理能力。

该开发框架提供的主要功能必须集成在开发板上，才能保障一切工作正常。说采集、集成和优化这样的功能的逻辑是一件既费时又费钱的工作，应该不算过分。因此这种开发框架为项目经理带来了巨大的价值，有助于加速产品上市进程。

AdvancedIP 投入了大量的精力，以确保其框架经优化后能够提供最优异的性能，空间占用小，运行效率高。这样可以让 FPGA 的大部分资源用于应用开发。比如，当 V5022 使用赛灵思 Virtex-6 HX565T 时，该框架占用

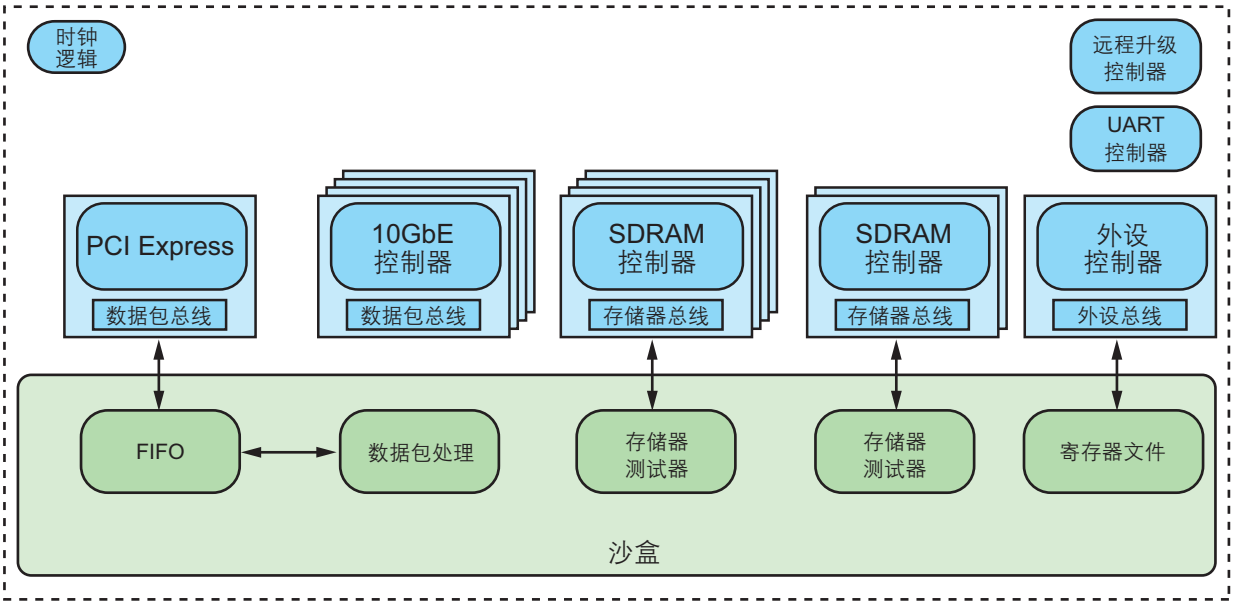


图 4 - AdvancedIO 公司提供的 expressXG FPGA 开发框架的高级视图。
最上面一排的组件（蓝色）是集成在硬件中的控制器。下面的组件是供开发使用的基础架构。

	工具	开发时间（小时）	频率（MHz）	面积占用率（%）
1	C	3.0	–	–
2	Impulse-C	6.0	125	21.25
3	Handel-C	7.5	200	20.25
4	Carte-C	6.5	100	19.50
5	Mitrion-C	10.75	100	22.75
6	SysGen	9.0	200	19.00
7	RC Toolbox	9.0	200	19.00
8	HDL	15.25	200	16.75

表 1 - 从 HLL-to-HDL 工具与直接用 HDL 开发的对比可以看出，虽然显著节省了时间，但得到的代码比用 HDL 直接创建的低效。[1]

FPGA 的资源不足 7.5%。这其中包括 PCIe® 接口，四个 10GE 接口、两个 SDRAM 控制器和四个 SRAM 控制器。

该开发框架提供了一个“沙盒”供编程人员开发自己的应用。它有数个易于理解的对外接口，便于在 FPGA 卡上迅速集成应用。另提供示例代码和示例，展示如何开箱即用地运用接口，布置系统数据流，从而增强开发人员的信心。

节省数月时间

为降低在 FPGA 器件上开发 HDL 代码涉及的风险，缩短开发时间，我们的 FPGA 开发框架集成并优化了在 FPGA 卡上开发应用所需的全部控制器。这至少可以缩短项目数月的开发时间。我们还建议将 C-to-HDL 编译器工具集成在开发框架中，因为研究结果表明，虽然工具生成的代码效率低于 HDL 直接编码，但它能够显著缩短开发时间。

在现实情况中，不存在一种能够适应各种情况的万灵丹式的方法。在权衡开发方法时，设计小组应有许多工具和选项可供使用和选择。在要求效率或紧凑代码时，或者在组件用于多个项目中时，最佳方法是使用 FPGA 开发框架开发 HDL 组件。若优先考虑产品上市时间，以及已经有 C 语言代码可用的情况下，最佳方法是使用 C-to-HDL 编译器工具。最好是使用 FPGA 开发框架在 HDL 中开发 TCP/IP 和 UDP/IP 堆栈等固定功能的模块，

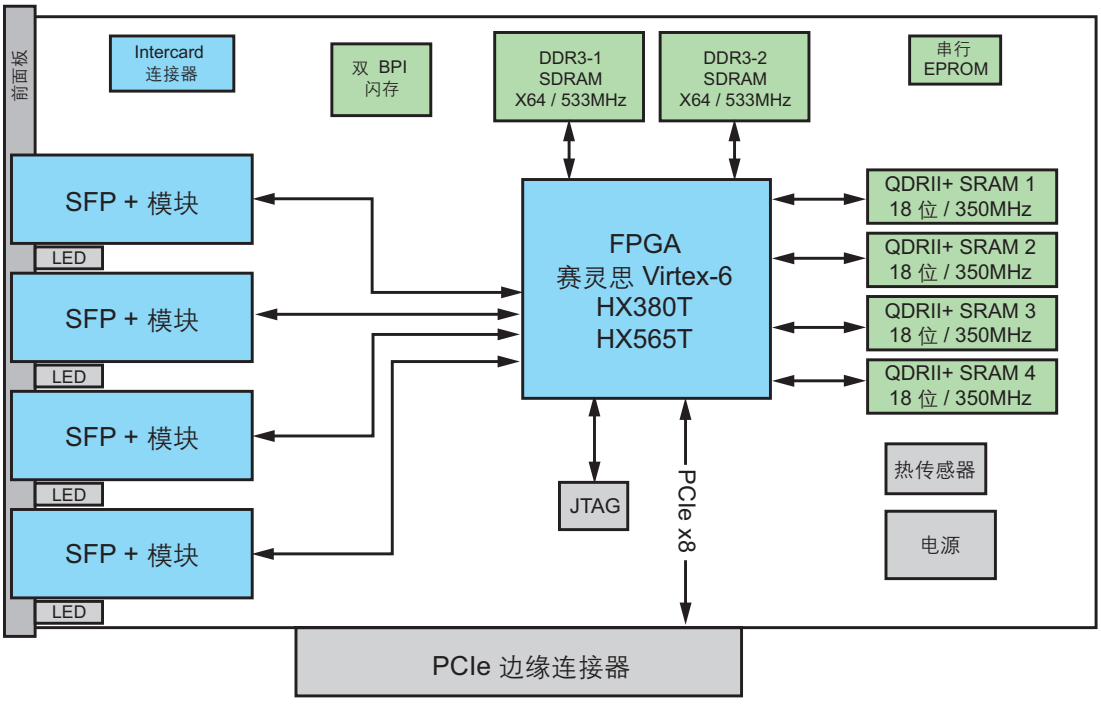


图 5 - V5022 的高级原理图显示了 FPGA 设计人员在实现解决方案时实现通信功能所需的各种组件

同时需要经常修改的算法可以使用赛灵思 AutoESL [4] 或 Impulse[5] 高层次综合工具等高级语言工具来开发。

金融交易系统的应用

图 7 是需要在很高水平上实现的算法交易系统的不同组成部分。ATS 必须具备读取一个或多个市场数据馈入，过滤数据并发送到不同“篮子”中用于分析，制定交易决策以及与一个或者多个交易所通信的能力。

交易逻辑和策略组件会经常变动，且不同的交易机构有自己的一套做法。他们最好是使用 C-to-HDL 编译器来实现，以满足产品上市时间要求，而且如果市场要求，可以在后期转为使用效率更高的实现方式（使用 FPGA 开发框架）。另一方面，网络协议和金融协议不会经常发生变化，用高效的方法实现这些协议会严重影响系统性能。因此我们建议使用 expressXG 开发框架在 HDL 直接实现这些协议。

速度、响应性和预测性

金融交易行业，在速度和超低时延的推动下，FPGA 技术不断得到推广和普及。这个行业的用户面临着诸多挑战，包括不熟悉 FPGA 设计、技能要求不同、有庞大的高级语言代码库等。我们的 expressXG FPGA 开发平台能

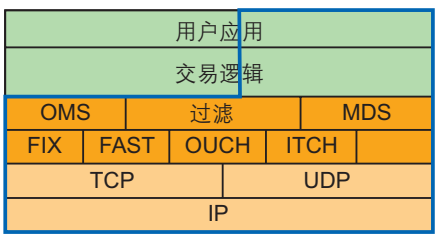


图 7 - ATS 各组件中，蓝色线标出了 FPGA 有助于提高其性能的软件组件

图 6 - V5022 是一个在金融交易系统中广泛部署的单插槽半长卡

够简化在基于 FPGA 的高性能以太网 PCI Express 卡上进行的应用开发工作，并缩短应用开发时间。为便于移植现有的 C 语言代码，或者在产品上市速度压倒一切的情况下，我们建议将一个 C-to-HDL 编译器集成在该框架中。

我们认为集成有 C-to-HDL 编译器的 expressXG 有助于 FPGA 技术的采用，提高交易系统的速度、响应能力和预测能力。关于 expressXG 系统的更多信息，敬请访问：*http://www.advancedio.com/*。

参考资料

1.E. El-Araby、S.G. Merchant 和 T. El-Ghazawi 共同编著的《用于评估用于高性能可重配置计算机的高级设计方法的框架》，摘自 2011 年 1 月第 22 卷第一期 33-45 页《IEEE 交易并行和分布式系统》。摘要见：http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5445087。

2.E. El-Araby、M. Taher、M. Abouellail、T. El-Ghazawi 和 G.B. Newby 共同编著的《可重配置计算机高级编程的比较分析：方法和实证研究》，2007 年第三次南方可编程逻辑大会，2007 年 2 月，摘要见：http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4234328。

3.Chronus 集团，http://www.khronos.org/opencl/。

4. 赛灵思公司，http://www.xilinx.com/tools/autoesl.htm。

5. 脉冲加速技术

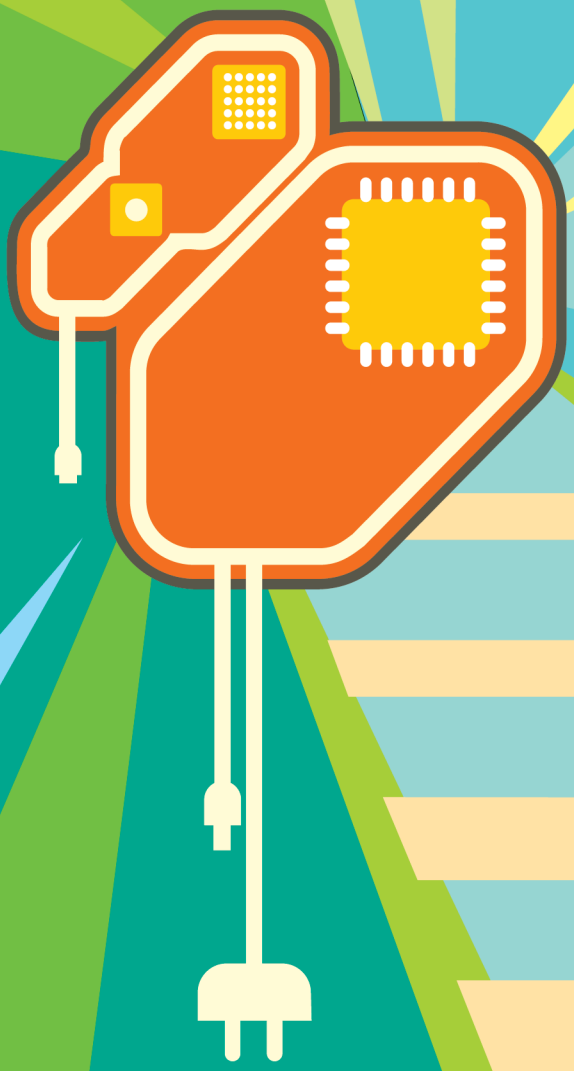
用纯硬件解决方案加速部分重配置进程

对视频等时序关键型应用，采用纯硬件解决方案可提升赛灵思 FPGA 的运行能力。

作者：Sebastlen Lammonier
FPGA 设计师
Sagem DS（赛峰集团）
sebastien.lamonnier@sagem.com

Marc Thoris
FPGA 项目经理
Sagem DS（赛峰集团）
marc.thoris@sagem.com

Marlène Ambielle
FPGA 设计师
Sagem DS（赛峰集团）
marlene.ambielle@sagem.com



在视频处理等众多新型应用中，尽可能缩短 FPGA 重配置时间对避免丢失过多图像至关重要。部分重配置指用户在不影响 FPGA 周边逻辑单元的情况下对其的一小部分进行重新配置的技术。如果要人眼观察不到图像的闪烁，重配置所花的时间不得超过 40 毫秒。除了最小型的 FPGA，对重配置整个器件而言这点时间太短。但在某些特定的情况下，该重配置时间还需要进一步压缩。于是部分重配置技术应运而生，因为部分重配置的比特流比完全重配置的比特流小，所以重配置所花的时间也更少。

我们这些在 Sagem DS 工作的开发人员已设计出一种技术，能够让 FPGA 设计人员以极快的速度完成部分重配置工作。我们使用赛灵思 ML507[1] 开发板来测试、验证解决方案和测量时序。一般情况下该开发板由一片 Virtex®-5 FPGA（XC5VFX70T-FFG1136）、一片 CPLD（用作路由组件）和两片 XCF32P 存储器（赛灵思平台闪存）组成。

MICROBLAZE 与硬件解决方案的对比

在许多技术文档中，部分重配置（PR）技术使用像 MicroBlaze® 这样的内部控制器或外部处理器。根据具体的配置，在 FPGA 内实现处理器需要占用开发时间，消耗大量的器件资源。同样，使用外部处理器会增加成本，占用电路板空间。另外，像 PLB 或 AXI 这样的总线存在时延，这样会延长重配置时间。

基于上述种种原因，我们采一款基于小型状态机的纯硬件解决方案，并采用内部配置访问端口 (ICAP) 接口加载比特流。这种方法具有多种优势：不存在时延，这种方法基本不占用资源（在 FPGA 上占用的查找表不足 300 个），而且设计人员可以优化部分重配置的时序。

开发流程概览

从 VHDL 概念到比特流和部分比特流的创建，除了没有嵌入式处理器，我们的纯硬件部分重配置流程与赛灵思辅导教程、用户指南 [2] 和应用指南中介绍的一般流程一样。用户必须在 PlanAhead ™ 中定义可重配置区域（RP），并为每个区域导入可重配置模块（RM）。对于所有的 configuration runs 静态逻辑都可以从以前跑出的 runs 导入。

在部分重配置的过程中，FPGA 必须处于从模式。这就是说，可用的接口只有 JTAG、从串、从并（Slave SelectMap）或 ICAP。完成重配置需用外部组件驱动 FPGA 的 CCLK；ICAP 在 FPGA 首次启动时不能访问。为节省重配置所需的时间，可以不使用串行接口。这样就给我们留下至少两种接口供选择：SelectMap 和 ICAP。

第一种选择是使用 SelectMap 接口供全部和部分比特流加载。这种配置方法需要在创建比特流的时候增加一个 Bitgen 选项（-g Persist）。这样就由 FPGA 保持对 SelectMap 引脚的控制，以便加载部分比特流。另外在使用 SelectMap 的情况下，没有信号非常准确地提示流程结束（比如完全配置的完成（DONE）信号）。所以难以确切地知道部分重配置是什么时间结束的。用户必须创建一个模块，用于估计所有配置数据完成发送的时间。

这就是为什么我们最终选择使用 ICAP 原语加载部分比特流的原因。ICAP 不是一种像 SelectMap 这样的自配置接口，所以我们在使用 SelectMap 实现 FPGA 的首次启动之后，就由用户代码完全控制 ICAP 原语。

与纯粹的 SelectMap 设计相比，ICAP 具有两大优势。首先，转换（SelectMap 用于首次启动，ICAP 用于部分比特流加载）对用户是透明的，无需使用 Bitgen 选项。因此用户可以控制存储器引脚。其次，也是最主要的优势，ICAP 能够不断将状态反映到输出上。如果 FPGA 处于重配置模式，这个状态就会发生改变。这样用户就能够"看到"部分重配置的结束。

重配置功能和组件

在我们公司，我们使用 FPGA 开

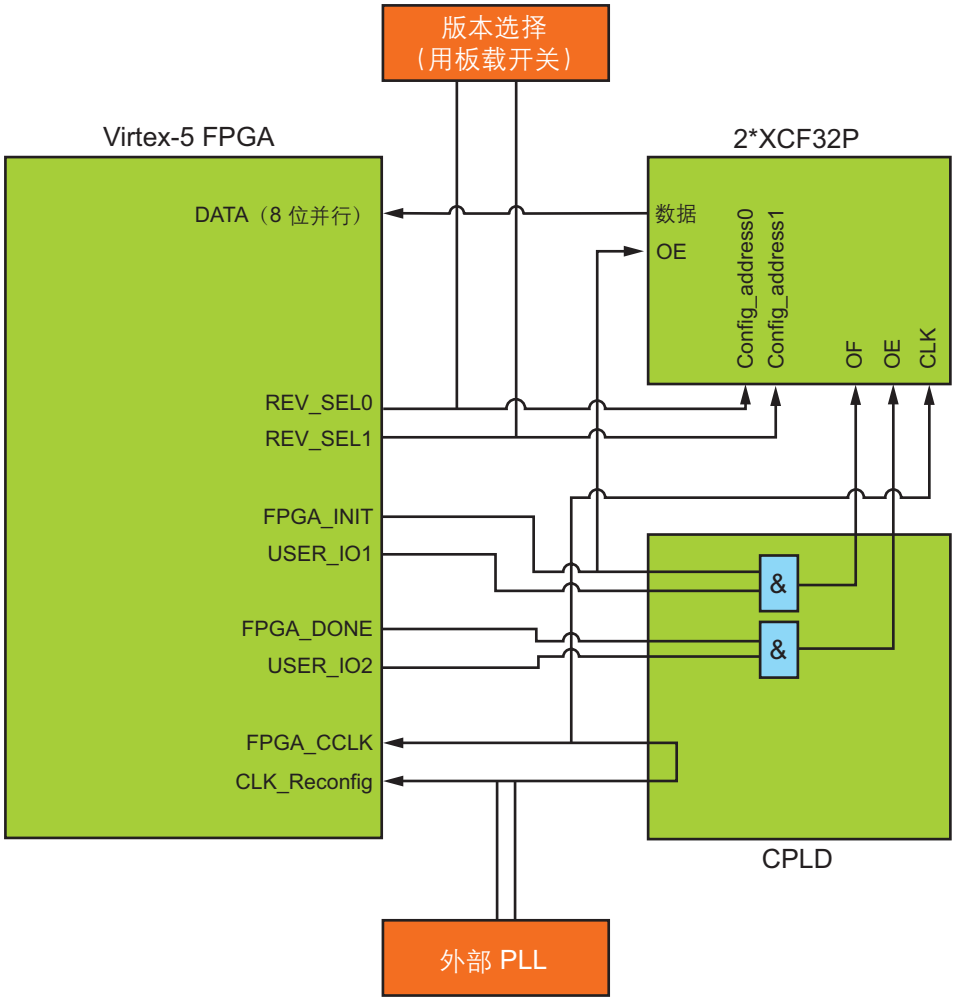


图 1 - 部分比特流加载、ICAP 同步和去同步

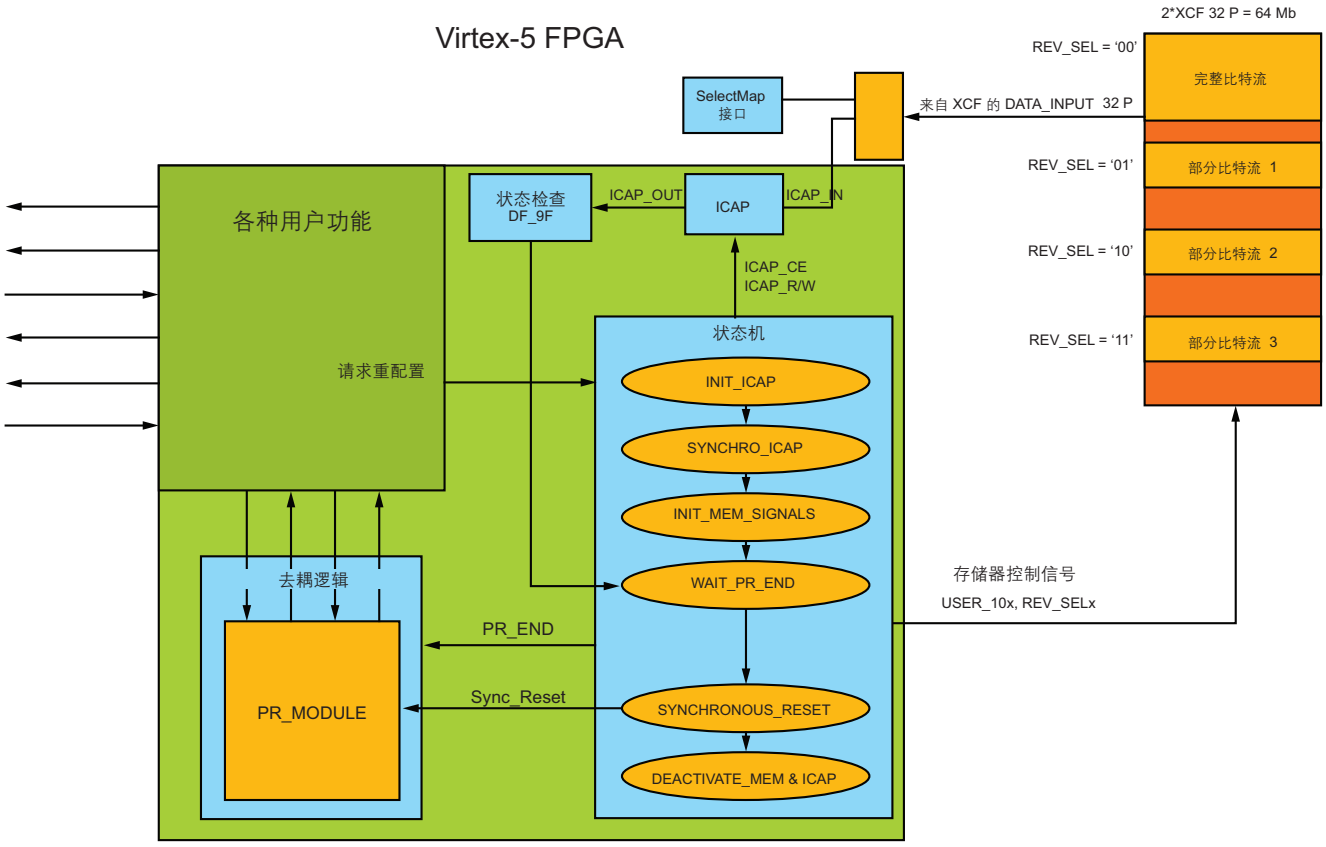


图 2 - FPGA 内的硬件 PR 控制模块

发视频处理应用。这些功能会用到逻辑元件、部分 BRAM 组件和大量 DSP。研究这三种元件的重配置时间非常有意义，因为它能指导我们在回答下面两个问题的基础上设定可重配置区域的大小：要满足我们的应用要求的 40 毫秒图像刷新时间，我们应如何设定可重配置区域的大小？我们需要在这个区域中放入多少个不同的元件？我们通过在 ML507 上测试不同类型的配置，回答了这两个问题。

如图 1 所示，在 ML507 上我们选择外部锁相环 (PLL) 用作部分重配置控制的基准时钟。部分连接并非我们刻意选择，而是电路板设计所需。该原理图没有显示避免走线网络电平冲突的保护电路。时钟频率为 33MHz，是开发板上两个 XCF32P 存储器的最高速率。[3] 数据总线宽度为 8 位，可实现高达 264Mbps 的数据传输速率。

FPGA 上最小的可重配置区域是一帧。帧大小随 FPGA 的类型变化。例如，在 Virtex-5 上，一帧的大小是 20 CLB，[4] 但在 Virtex-6 上，帧的大小是其 2 倍。部分比特流就是基本帧、BRAM、DSP 和一些配置字的组合。配置接口和部分比特流的结构（内含的元件数量）是缩短部分重配置时间的关键。

重配置流程

图 2 是为部分重配置设计的 FPGA 的粗略架构。接通开发板上的开关即可启动流程。FPGA 通过并行 SelectMap 接口加载全部比特流。此后由用户代码控制图中浅绿和深绿色区域中的所有模块。去耦逻辑是一个围绕部分重配置模块设计的专门区域。

在重配置过程中，连接网络走线上可能会出现未知状态。这个区域可以让用户在静态区域和可重配置区域之间的连接网络走线上保持已知状态。去耦逻辑实际上很小，仅由每个网络上的一个 D 触发器和一个多路复用器组成。信号“PR_END”负责控制多路复用器。

现在开始对存储器中的部分比特流的新配置对 PR_MODULE（例如直方图修正函数）进行重配置。“请求重配置”信号启动部分重配置流程，随后是一系列状态（图 2 中的状态清单并未完全列举）。

- 状态 1，ICAP 初始化，用于赋予控制信号以默省值（CE 和 RW 高电平）。
- 状态 2，让 ICAP 准备从存储器接收数据。ICAP 现已激活，目前处

于写模式（向 FPGA 里写入配置）。

- 状态 3，负责控制存储信号，并初始化从存储器到 FPGA 的部分比特流传输序列。同时，去耦逻辑负责保持静态区域和可重配置区域之间的信号。这样可以防止不需要的数据在用户功能中传输。
- 状态 4 为等待状态。在这期间，ICAP 将配置帧载入 FPGA，同时“状态检查”模块读取 ICAP 输出状态。在检测到“去同步”字时即从等待状态中退出。
- 状态 5，负责启动 PR_MODULE 内的同步复位功能，进行复位，使得新的逻辑元件处于已知状态。
- 最终状态释放去耦逻辑，禁用 ICAP 和存储器。

ICAP 内部

用户无需准确了解或关注 ICAP 的工作原理。这是因为部分比特流已经提供了 ICAP 实现该应用所需的一切。尽管如此，还是应该了解 ICAP 中的两个 32 位字：同步和去同步。

第一个字，“同步”，在输入上体现为 5599AA66h。这个字让 ICAP 的输出从 9Fh 变为 DFh（DFh 意为“组件同步”）。在 ICAP 输出状态为 DFh 时，FPGA 负责加载新的配置帧。

在配置数据发送后，比特流中包含一个“去同步”字，即 000000B0h。当 ICAP 接收到这个字，其输出状态变为 9Fh，说明该组件已经去同步。因此，检查 ICAP 的输出状态就能够让我们准确地掌握部分重配置时间。图 3 是带有相关信号的帧的重配置时间。

实验和结果

设计诸如自动直方图修正 (AHC) 等视频处理函数，需要使用逻辑单元、BRAM 和大量的 DSP。由于对照明系统、热显影和电视视频来说，修正量并非一样多，FPGA 必须具备自适应能力，以满足最低响应时间要求（低于 40 毫秒，即眼睛暂留一幅图像的时间）。图 4(a)、(b) 和 (c) 显示的结果让我们对我们所用接口的重配置时间有了准确的认识。我们使用微型重配置模块（一个只有逻辑，一个有逻辑和 BRAM，最后一个有逻辑和 DSP）来测量这些时序结果。结果表明，各组件的时间是线性的。当然，这个结果只对我们的接口规范有效（33MHz 时钟，8 位宽总线）。

更大功能的重配置（比如处理 AHC 的功能的重配置）与我们的产品更密切相关，也检验了 FPGA 的运行情况。简单介绍一下 AHC 功能。一幅电视图形有 Y、Cr 和 Cb 三个分量。Y 代表亮度，Cr 和 Cb 分别代表红色和蓝色。AHC 模块必须将图像转换为 RGB 格式，并对每种颜色进行特殊处理。对黑白图像而言，可以通过求最大值、最小值、平均值，然后运用放大、偏移或同时放大和偏移来增加亮度（对暗图形）和对比度。用 FPGA 资源来衡量，这种校正需要占用约 5,000 个查找表 (LUT)、4,000 个触发器、100 个 DSP Slice 和 20 个 BRAM Slice。

图 5 是我们为 AHC 功能选择的可重配置区域。该区域内含 7,840 个查找表 (LUT) 和触发器、112 个 DSP

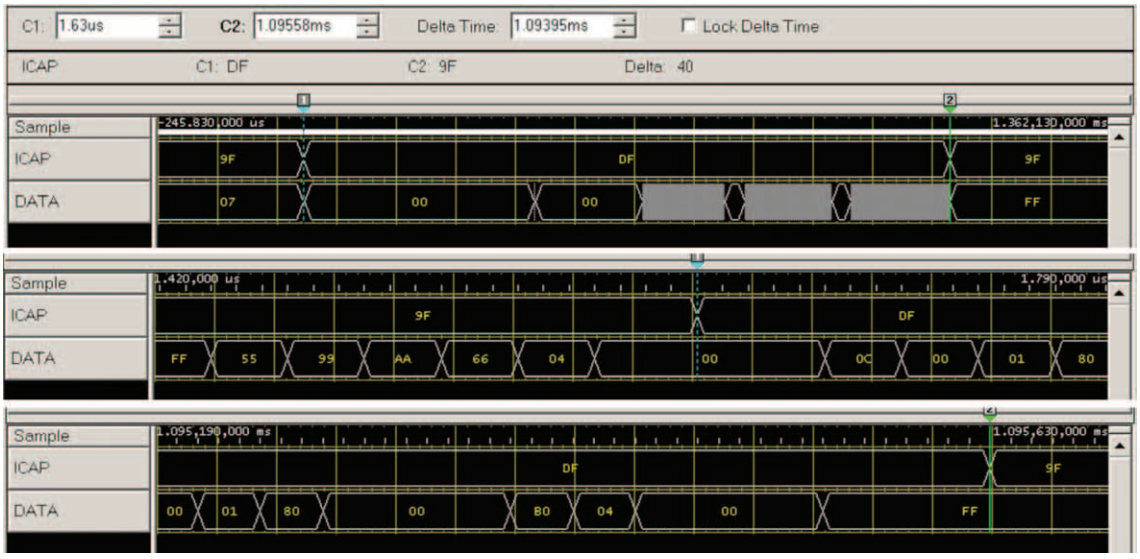


图 3 - 六个 Slice 组成的配置帧区域的 ICAP 时序

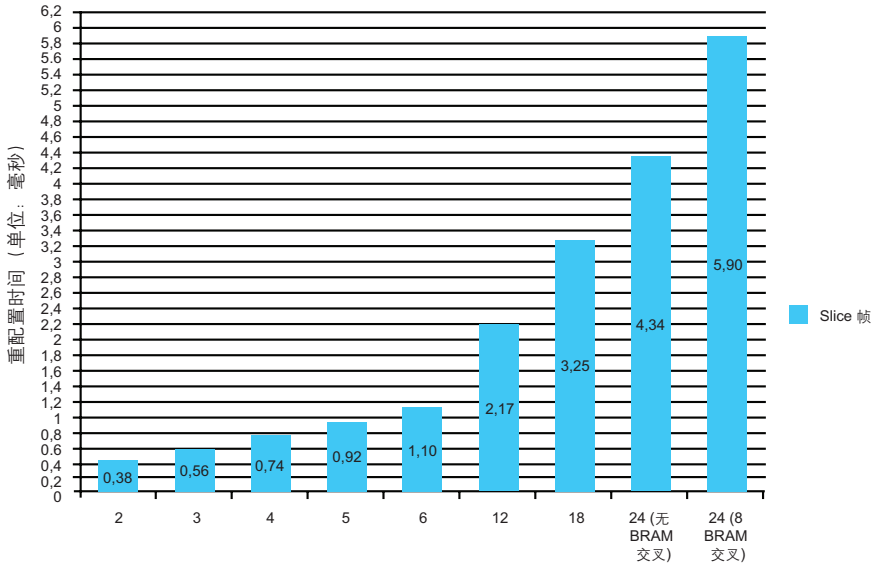


图 4 (a) - 仅有 slice 帧的重配置模块所需的重配置时间

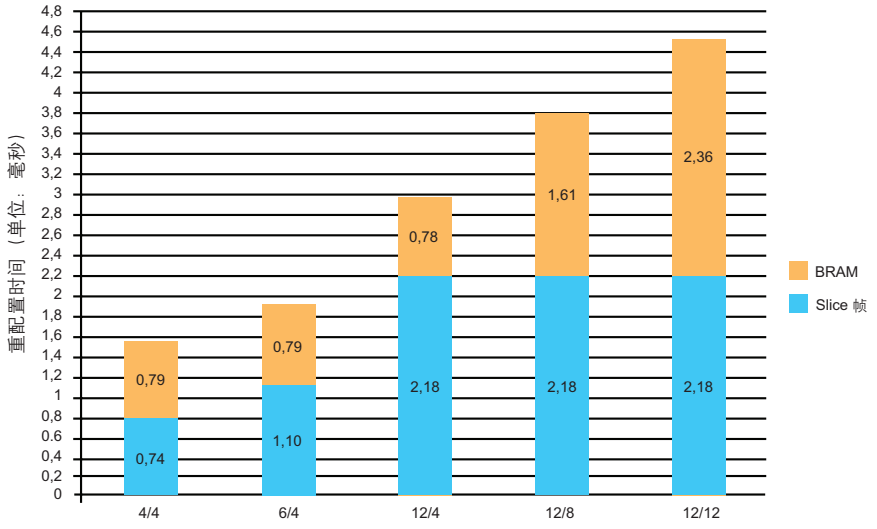


图 4 (b) - 含有 slice 帧和 BRAM 帧的重配置模块所需的重配置时间

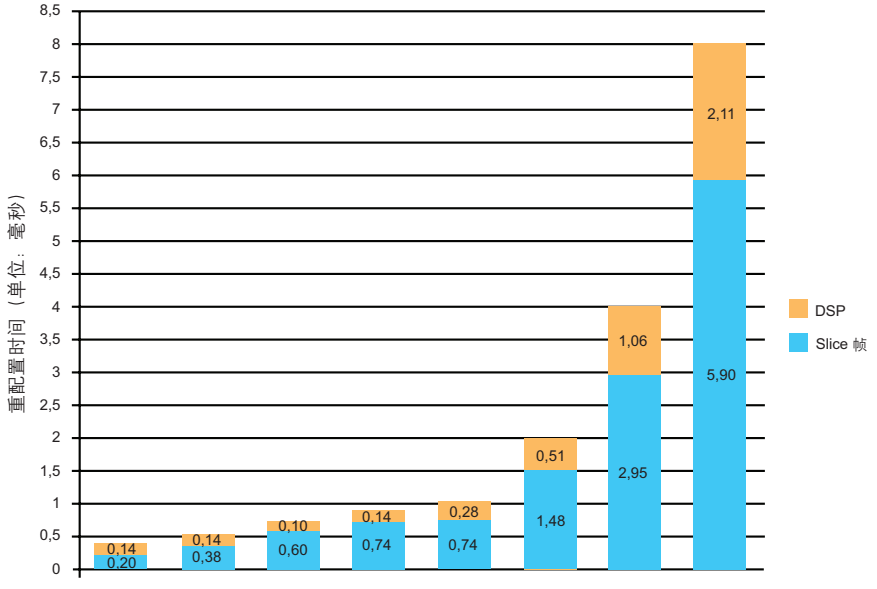


图 4 (c) - 含有 slice 帧和 DSP 帧的重配置模块所需的重配置时间

Slice 以及 28 个 BRAM Slice。在该设计中，我们遵循了赛灵思提出的可重配置区域中的逻辑密度不得高于 80% 的建议。Virtex-5 架构的一个 Slice 中有 4 个 LUT 和 4 个触发器。一个可重配置逻辑模块 (CLB) 有两个 Slice，故一个帧中含有 160 个 LUT 和 160 个触发器。我们的区域中含有 49 个帧。

根据图 4(a)、(b) 和 (c)，我们可以用下列方法计算出重配置时间。逻辑需要 9.8 毫秒、112 个 DSP 需要 2 毫秒，28 个 BRAM 需要 6.4 毫秒。这个区域共需要 18.2 毫秒即可完成重配置。这比眼睛暂留一幅图像所需的 40 毫秒要低得多。图 5 是可重配置区域的 PlanAhead 视图。

无需软件开发

如图 1 所示，我们可以使用定制板上的逻辑组件替代 CPLD。目前该解决方案运行良好，但尚未达到理想状态。我们可以使用更宽的总线（ICAP 支持多达 32 位并行）和更高的频率来改进该可重配置接口。

纯硬件部分可重配置与 MicroBlaze 解决方案相比，由于在系统开发过程中，无需进行软件开发，从而显著节约了开发成本。通过使用 ICAP 接口，设计人员能够完全掌控 FPGA 中工作情况，同时 ICAP 还极大地改善了系统的性能和功能。最后，尤其是对我们的视频处理应用而言，纯硬件部分重配置是一种能够增强我们产品性能的重要功能。

参考资料：

1.ML505/6/7 原理图，版本 A，修订版 2，赛灵思，2008 年 1 月，http://www.xilinx.com/support/documentation/boards_and_kits/ml50x_schematics.pdf

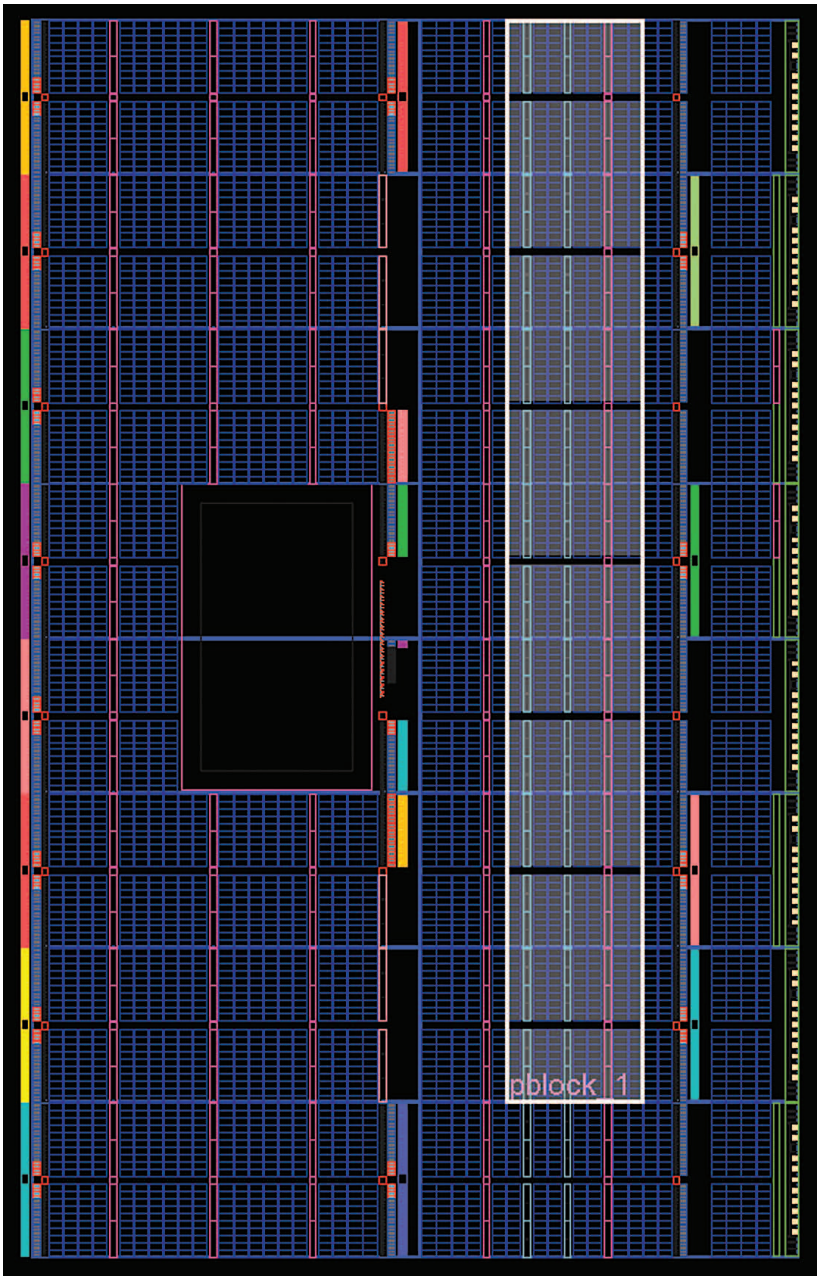


图 5 - 可重配置区域的 PlanAhead 视图

2.《部分可重配置用户指南》(UG702)，13.1 版，赛灵思，2011 年 3 月 http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_1/ug702.pdf

3.《平台闪存系统内可编程配置 PROM》(DS123)，2.18 版，赛灵思，2010 年 5 月 http://www.xilinx.com/support/documentation/data_sheets/ds123.pdf

4.《Virtex-5 FPGA 配置用户指南》(UG191)，3.9.1 版，赛灵思，2010 年 8 月 http://www.xilinx.com/support/documentation/user_guides/ug191.pdf

面向未来十年的 “All Programmable” 器件 赛灵思Vivado 设计套件 震撼登场

一个以IP及系统为中心的工具套件震撼登场，把可编程系统的集成度和实现速度提升至原来的4倍

2012 年 4 月 25 日，中国，北京 — 全球可编程平台领导厂商赛灵思公司 （Xilinx, Inc. (NASDAQ:XLNX) ）今天全球公开发布以 IP及系统为中心的新一代颠覆性设计环境 Vivado 设计套件，致力于在未来十年加速 “All Programmable” 器件的设计生产力。Vivado不仅能加速可编程逻辑和 IO 的设计速度，而且还可提高可编程系统的集成度和实现速度，让器件能够集成 3D堆叠硅片互联技术、ARM 处理系统、模拟混合信号 (AMS) 和绝大大部分半导体IP 核。Vivado 设计套件突破了可编程系统集成度和实现速度两方面的重大瓶颈，将设计生产力提高到同类竞争开发环境的4 倍。

Vivado 设计环境

Vivado设计套件包括高度集成的设计环境和新一代系统到 IC 级别的工具，这些均建立在共享的可扩展数据模型和通用调试环境基础上。这也是一个基于 AMBA AXI4 互联规范、IP-XACT IP 封装元数据、工具命令语言 (Tcl)、Synopsys 系统约束 (SDC) 等有助于根据客户需求量身定制设计流程并符合业界标准的开放式环境。赛灵思构建的 Vivado 工具将各类可编程技术结合在一起，可扩展实现多达 1 亿个等效 ASIC 门的设计。

为了解决集成的瓶颈问题，Vivado IDE 采用了用于快速综合和验证 C 语言算法 IP 的 ESL 设计、实现重用的标准算法和RTL IP 封装技术、标准IP 封装和各类系统构建块的系统集成、可将仿真速度提高 3 倍的模块和系统验证功能，以及可将性能提升百倍以上的硬件协同仿真功能。

为了解决实现的瓶颈，Vivado 工具采用层次化器件编辑器和布局规划器、速度提升了3 至 15 倍且为 SystemVerilog 提供业界领先支持的逻辑综合工具、速度提升 了4 倍且确定性更高的布局布线引擎、以及通过分析技术可最小化时序、线长、路由拥堵等多个变量的 “成本” 函数。此外，增量式流程能让工程变更通知单 (ECO) 的任何修改只需对设计的一小部分进行重新实现就能快速处理，同时确保性能不受影响。最后，Vivado 工具通过利用最新共享的可扩展数据模型，能够估算设计流程各个阶段的功耗、时序和占用面积，从而达到预先分析，进而优化自动化时钟门等集成功能。

博通公司欧洲硬件开发工程经理 Paul Rolfe 指出：“Vivado 设计套件与Virtex-7 2000T FPGA的组合改变了可编程逻辑产业发展的模式。Vivado 使博通无需进行任何手动布局规划或分区工作，就能够设计出业界最大容量的 FPGA。赛灵思在芯片和软件双方面的创新让我们印象深刻。”

供货情况

Vivado 设计套件 2012.1 版本现已作为早期试用计划的一部分推出。客户可联系所在地的赛灵思代表。今夏早些时候将公开发布 2012.2 版本，今年晚些时候还将推出 WebPACK。目前采用 ISE 设计套件版本的客户将免费获得最新 Vivado 设计套件版本和IDS。赛灵思将继续为针对 7 系列及早期产品设计的客户提供 ISE 设计套件支持。如需更多信息，敬请访问以下网址：www.xilinx.com/cn/design-tools。

如何在 FPGA 设计中使用 CORDIC 算法

虽然 CORDIC 是实现 DSP 和数学函数最重要的算法之一，但许多设计人员并不熟悉。

作者：Adam P. Taylor
首席工程师
阿斯特里姆公司 (EADS Astrium)
aptaylor@theiet.org

大多数工程师在碰到需要在 FPGA 中实现诸如正弦、余弦或开平方这样的数学函数时，首先会想到的是用查找表，可能再结合线性内插或者幂级数（如果有乘法器可用）。不过对这种工作来说，CORDIC 算法是工具库中最重要的工具之一，只是鲜有工程师知晓。

CORDIC 的意思是坐标旋转数字计算机，是 Jack Volder 在 1959 年为康维尔公司 (Convair) B-58A "盗贼" 项目设计新的导航计算机时发明的。这是一种设计用于计算数学函数、三角函数和双曲函数的简单算法。

这种算法的真正优势在于只需要采用极小型的 FPGA 封装就可以实现它。CORDIC 只需要一个小型查找表，加上用于执行移位和加法的逻辑。重要的是这种算法不需要专门的乘法器或除法器。

这种算法是 DSP 以及工业与控制应用最有用的工具之一，其最为常见的用途是实现如表 1 所示的传统数学函数，可在器件缺少专用乘法器或 DSP 模块的情况下提供器件所需的乘法器、除法器或更有意义的函数。举例来说，设计人员在众多小型工业控制器中使用 CORDIC 来实现数学传递函数和真正的 RMS 测量。工程师也在生物医学应用中使用 CORDIC 来进行快速傅里叶变换 (FFT) 计算，以分析多种生理信号的频谱。在本应用中，结合传统的数学函数，设计人员使用 CORDIC 实现 FFT 旋转因子。

CORDIC 详解

CORDIC 算法可以采用线性、圆或双曲线三种配置中的任

配置	旋转	向量
线性	Op Y = X × Y	Op Z = X/Y
双曲线	Op X = CosH(X) Op Y = SinH(Y)	Op Z = ArcTanH
圆函数	Op X = Cos(X) Op Y = Sin(Y)	Op Z = ArcTanH (Y) Op X = SQR(X^2 + Y^2)

表 1 - 根据 CORDIC 配置和模式得出的函数

何一种进行运算。而对于每种配置，算法又可以在旋转或向量任一模式下执行。在旋转模式下，输入向量按一定的角度旋转，而在向量模式下，算法将输入向量旋转到 X 轴，同时记录旋转角度。

另外，可以从 CORDIC 的输出求出其它函数。许多情况下，甚至可以使用另一个配置截然不同的 CORDIC 来实现这些函数，如下所示：

Tan = Sin / Cos

TanH = Sinh / Cosh

指数 = Sinh + Cosh

自然对数 = 2 x ArcTanH

（其中，X = 自变量 +1，Y = 自变量 –1）

SQR = (X^2 – Y^2)

（其中，X = 自变量 +0.25，Y = 自变量 –0.25）

下面的统一算法涵盖了 CORDIC 的所有三种配置。该算法有 X、Y 和 Z 三种输入。表 2 是根据配置预先计算的查找表值，表 3 则是根据运算模式（向量或旋转）在启动时的初始化方式。

$$X_i+1 = X_i - m \times Y_i \times d_i \times 2^i$$

$$Y_i+1 = Y_i + X_i \times d_i \times 2^i$$

$$Z_i+1 = Z_i - d_i \times e_i$$

其中 m 表示双曲线（m=-1），线性（m=0）或旋转（m=1）配置。e_i 值则为不同配置下的对应的旋转角度。e_i 值一般在 FPGA 中通过小查找表实现，如表 2 所示。

在这个等式中，di 为旋转方向，这取决于运算模式。在旋转模式下，如果 Zi < 0，则 di = -1，否则 di = +1；在向量模式，如果 Yi < 0，则 di = +1，否则 di = -1。

在圆函数旋转模式或双曲线旋转模式下，输出结果将有增量，这部分增量可以通过下面等式中定义的旋转次数

配置	e_i
线性	2^{-i}
双曲线	ArcTanH(2^{-i})
圆函数	ArcTan(2^{-i})

表 2 - CORDIC 旋转角度

来预先求得。

$$An = \sum \sqrt{(1 + 2^{-2i})}$$

这部分增量一般反馈到算法的初始设置中，以免对结果进行后缩放。

设计人员在工作时必须牢记 CORDIC 算法只能在严格收敛域内运算。可能需要进行一定的预缩放，以确保其能够按预期执行。需要指出的是，决定执行的迭代（串行）或级数（并行）的次数越多，得到的运算结果就越准确。经验法则告诉我们，如果需要 n 位的精度，就需要进行 n 次迭代或 n 个级数。不过在横切代码前，所有这些都可以采用诸如 Excel 或 MATLAB® 等简易工具轻松完成建模，从而确保能够用选定的迭代次数得到精确的结果。

根据定义，CORDIC 算法只能在有限的输入值范围内收敛（工作）。对采用圆函数配置的 CORDIC 算法而言，只要角度不大于查找表中的角度之和（即在 -99.7° 到 99.7° 之间）即可保证收敛。对大于这个范围的角度，必须使用三角恒等式将其转换为这个范围内的角度。采用线性配置时，其收敛亦如此。但是，要实现双曲线配置下的收敛，必须重复进行一定数量的迭代（4、13、40、K…3K+1）。在这种情况下最大输入值 θ 约为 1.118 弧度。

模式	Y	X	Z
圆函数旋转	A_n	0	自变量 Z
圆函数向量	1	自变量 X	0
双曲线旋转	A_n	0	自变量 Z
双曲线向量	自变量 + 1	自变量 - 1	0
线性旋转	0	自变量 X	自变量 Z
线性向量	自变量 Y	自变量 X	0

表 3 - 根据表 1 中的数学运算的初始化设置

模式	x	y	z	方向	增量
初始	0.607253	0.000000	0.017453		
0	0.607253	0.607253	-0.767945	1	0.707107
1	0.910879	0.303626	-0.304298	-1	0.894427
2	0.986786	0.075907	-0.059319	-1	0.970143
3	0.996274	-0.047442	0.065036	-1	0.992278
4	0.999239	0.014826	0.002617	1	0.998053
5	0.998776	0.046052	-0.028623	1	0.999512
6	0.999496	0.030446	-0.012999	-1	0.999878
7	0.999734	0.022637	-0.005186	-1	0.999969
8	0.999822	0.018732	-0.001280	-1	0.999992
9	0.999859	0.016779	0.000673	-1	0.999998
10	0.999842	0.017756	-0.000304	1	1
11	0.999851	0.017268	0.000185	-1	1
12	0.999847	0.017512	-0.000060	1	1
13	0.999849	0.017390	0.000063	-1	1
14	0.999848	0.017451	0.000001	1	1
	余弦	正弦		A_n	0.607253
CORDIC	0.999848	0.017451			
实际	0.999848	0.017452			

表 4 - 圆函数旋转模式的 CORDIC 的 Excel 模型

CORDIC 应用

设计人员在从数字信号处理和图像处理到工业控制等多种应用中采用 CORDIC 算法。最基本的用法是将 CORDIC 与相位累加器结合，以生成正弦波和余弦波，供 I 调制和 Q 调制使用。使用该算法生成这些波形，如果生成正确，可以实现高度的无杂散动态范围 (SFDR)。对大多数信号处理应用而言，需要良好的无杂散动态范围。

在机器人技术领域，CORDIC 被用于运动学领域，对判断机器人的关节、四肢的位置和运动非常有用。在该应用中，可以使用圆函数向量模式的 CORDIC 算法轻松将坐标值与新坐标值相加。在图像处理领域，像光照和向量旋转这样的三维运算最好选用该算法来实现。

在 EXCEL 中建模

在横切代码前，建立 CORDIC 算法模型最直观的方法之一是填制简单的 Excel 数据表。这样可以先用浮点数系统，再用可扩展的定点数系统为迭代次数和增量（An）建模，以便为仿真过程中的代码验证提供参考。

从表 4 的 Excel 模型中可以看到，将初始的 X 输入设为 An，可以减少结果后处理工作量。初始自变量设为 Z，单位为弧度，和结果一样。

实现 CORDIC

如果没有其他更好的选择，在 FPGA 中实现 CORDIC 算法的最简单方法就是使用像赛灵思 CORE Generator® 这样的工具。CORE Generator 提供了全面的接口，供用户定义 CORDIC 的确切功能（旋转、向量等），如图 1 所示。

令人遗憾的是，CORE Generator 不提供 CORDIC 在线性模式下工作的选项（该工具确实提供有执行这些功能的独立内核）。不过只需要几行就可以编写出实现该算法

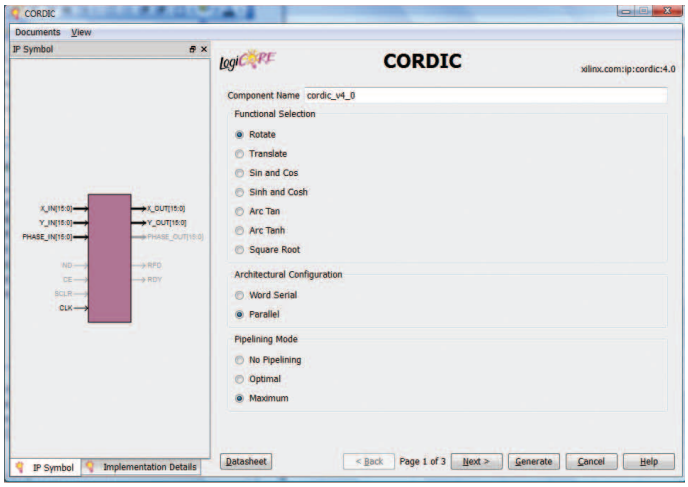


图 1 - 使用 CORE Generator 可以轻松实现众多常见的 CORDIC 配置

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY synth_cordic IS PORT(

clk : IN std_logic;
resetn : IN std_logic;
z_ip : IN std_logic_vector(16 DOWNTO
0); --1,16
x_ip : IN std_logic_vector(16 DOWNTO 0);
--1,16
y_ip : IN std_logic_vector(16 DOWNTO 0);
--1,16
cos_op : OUT std_logic_vector(16 DOWNTO
0); --1,16
sin_op : OUT std_logic_vector(16 DOWNTO
0)); --1,16
END ENTITY synth_cordic;
ARCHITECTURE rtl OF synth_cordic IS

TYPE signed_array IS ARRAY (natural RANGE
<> ) OF signed(17 DOWNTO 0);

--ARCTAN Array format 1,16 in radians
CONSTANT tan_array : signed_array(0 TO 16)
:= (to_signed(51471,18),
to_signed(30385,18),
to_signed(16054,18),to_signed(8149,18),
to_signed(4090,18), to_signed(2047,18),
to_signed(1023,18), to_signed(511,18),
to_signed(255,18), to_signed(127,18),
to_signed(63,18), to_signed(31,18),
to_signed(15,18),
to_signed(7,18),to_signed(3,18),
to_signed(1,18), to_signed(0, 18));

SIGNAL x_array : signed_array(0 TO 14) :=
(OTHERS => (OTHERS =>'0'));
SIGNAL y_array : signed_array(0 TO 14) :=
(OTHERS => (OTHERS =>'0'));
SIGNAL z_array : signed_array(0 TO 14) :=
(OTHERS => (OTHERS =>'0'));

BEGIN

--convert inputs into signed format

PROCESS(resetn, clk)

    BEGIN
      IF resetn = '0' THEN
        x_array <= (OTHERS => (OTHERS =>
'0'));
        z_array <= (OTHERS => (OTHERS =>
'0'));
        y_array <= (OTHERS => (OTHERS =>
'0'));
      ELSIF rising_edge(clk) THEN
        IF signed(z_ip)< to_signed(0,18)
THEN
          x_array(x_array'low) <=
signed(x_ip) + signed('0' & y_ip);
          y_array(y_array'low) <=
signed(y_ip) - signed('0' & x_ip);
          z_array(z_array'low) <=
signed(z_ip) + tan_array(0);
        ELSE
          x_array(x_array'low) <=
signed(x_ip) - signed('0' & y_ip);
          y_array(y_array'low) <=
signed(y_ip) + signed('0' & x_ip);
          z_array(z_array'low) <=
signed(z_ip) - tan_array(0);
        END IF;
    FOR i IN 1 TO 14 LOOP
      IF z_array(i-1) < to_signed(0,17)
THEN
          x_array(i) <= x_array(i-1) +
(y_array(i-1)/2**i);
          y_array(i) <= y_array(i-1) -
(x_array(i-1)/2**i);
          z_array(i) <= z_array(i-1) +
```

```
tan_array(i);
          ELSE
            x_array(i) <= x_array(i-1) -
(y_array(i-1)/2**i);
            y_array(i) <= y_array(i-1) +
(x_array(i-1)/2**i);
            z_array(i) <= z_array(i-1) -
tan_array(i);
          END IF;
        END LOOP;
      END IF;
   END PROCESS;
cos_op <=
std_logic_vector(x_array(x_array'high)(16
DOWNTO 0));
sin_op <=
std_logic_vector(y_array(y_array'high)(16
DOWNTO 0));
END ARCHITECTURE rtl;
```

在 FPGA 中实现 CORDIC 有两种基础拓扑，一种是状态机法，一种是流水线法。

如果对处理时间要求不是特别严格，可以使用状态机实现该算法，每周期计算一次 CORDIC 迭代，直到完成要求的周期次数。如果需要高计算速度，并行架构则更合适。上述代码采用 15 级并行旋转模式 CORDIC 计算。它使用如表 2 所示的简单圆函数模式 ArtTan(2-i) 查找表，结合简单的阵列结构实现并行级。

在五花八门的各种方法中，CORDIC 算法证明自己是一种简单且功能强大的算法，值得所有 FPGA 设计人员关注。更值得一提的是，使用内核生成工具或手动编码可以轻松实现 CORDIC。

FPGA IP 核软硬件协同验证采用形式验证技术

借助全新的形式验证技术，学术界和产业界研究人员可对赛灵思软核中紧密集成的硬件和固件进行全面验证。

作者：Markus Wedler
教授，工程学博士
柏林理工大学
markus.wedler@tu-berlin.de

Eric Crabill
IP 设计工程师
markus.wedler@tu-berlin.de

Graham Schelle
研究员级高级工程师
赛灵思公司
graham.schelle@xilinx.com

Patrick Lysaght
高级总监
赛灵思公司 patrick.lysaght@xilinx.com

从移动设备到汽车乃至工业机械，越来越多的产品采用需要软硬件协同工作的高级处理技术来执行一系列艰巨的任务。不过，随着系统日趋复杂性，设计人员在验证软硬件是否能够协同工作方面也面临着日益严峻的挑战。过去数十年来，企业和研究人员已推出一系列相关方法。不过，要想验证软硬件是否能如愿协同工作，仍然困难重重。

10 多年来，形式验证技术一直是设计团队进行硬件准确性验证工作最具发展潜力的技术之一。最近据英特尔公司透露，该公司工程师正是采用了这种验证技术，才解决了上世纪 90 年代 Pentium I 微处理器浮点单元问题。[1] 形式验证技术随后备受英特尔及众多其它硬件设计公司推崇，不过这种技术仍是一种比较冷门的选择。软硬件协同验证的形式验证技术在业界仍未得到广泛采用。

OneSpin Solutions 公司、凯泽斯劳滕大学 (University of Kaiserslautern) 和赛灵思的研究人员近期联合对如何采用形式验证技术全面验证赛灵思同时内含嵌入式固件和硬件组件的 IP 软核进行了一项调研。研究发现，在可扩展的形式验证环境中，可以捕获固件和硬件之间的互动。这项调研工作是产业界和学术界的一项重要合作，充分利用了近期与硬件有关的固件形式验证技术的进步，而且采用了间隔属性检查 (IPC) 这种有界模型检验方式。

模型检验和 IPC

形式验证技术采用数学方法，以确保设计满足严格的功能准确性规范要求。它从设计模型、系统工作环境描述以及设计可能遇到的一系列属性入手。随后要验证这些属性能否适用于所有情况，是否会出现属性失效的情况。设计人员越来越多地结合采用形式验证和较传统的仿真验证技术，从而发挥二者的最佳优势。举例来说，等效性检验和断言源于形式验证技术，不过现已广泛融入仿真验证流程之中。

模型检验是一种先进的系统自动形式验证技术，这些系统可作为有限状态机，系统规范则被视为时序逻辑的一系列属性。每个属性指定一段时间内与系统状态集相关的有效（或无效）逻辑行为。举例来说，RESET 属性断言：当 RESET 信号处于活跃状态，不管系统在一段时间内转换为何种状态，都将返回 RESET 状态。这些属性能以熟悉的语言格式表达，如 SystemVerilog 断言或 SVA（见侧栏：“间隔属性检查的工作原理”）。

一系列属性构成系统规范的抽象模型。模型检验软件负责处理每个属性，为全面验证属性，将涵盖设计所有可达的状态。如果属性失效，那么模型检测器会返回反例，说明属性未满足重设要求。

模型检验工作自动执行，相对迅速。与仿真验证技术不同的是，它能全面测试系统模型，从而常常能够发现隐蔽的问题。有时这些问题会出现在不起眼的角落里，有时那些为仿真测试台创建刺激测试模型和断言的验证工程师很容易忽视问题的起因。

现代系统有大量的状态变量，通常会出现所谓“状态空间爆炸”问题，可达的状态呈几何级数增长。由于状态空间非常复杂，很容易超出传统模型检测器的能力范围，从而限制这种技术的实用性，所以才需要使用间隔属性检查等更新的模型检验方法，也就是我们在这项工作中所使用的方法 [2]。

IPC 是一种专用的有界模型检测器。与其它有界模型检测器一样，IPC 会将属性范围限制在有限的时钟周期数之内，从而控制整体复杂性，并采用布尔可满足性 (SAT) 解算器来执行实际的模型检验工作。IPC 和传统有界模型检测器的区别在于，它的时钟周期窗口能允许属性断言在任意时间点上启动。

IPC 还可提供一种解决状态空间爆炸问题的简单方法。IPC 方法通过设计抽象来指导用户创建代表验证的设计关键功能的概念状态机 (CSM)。CSM 可捕获给定设计中所有基本操作或事务处理。在抽象的最高层，CSM 的每次状态转换都代表一个原子运动，被唯一的属性所覆盖。实际上，由于 CSM 是设计提取后的抽象视图，因此在相应的 RTL 代码中，每个 CSM 事务处理都对应于多达数百个相关信号活动的时钟周期。适当的属性可捕获全部最相关的状态以及周期精度级的输入输出信号行为，支持提取的完整周期。每个属性指定多周期时间间隔中的预期行为，精确对应于 CSM 的一次转换或操作。因此该方法就叫做间隔属性检查，也被称作操作式间隔属性检查。

CSM 抽象有意不捕获底层 RTL 设计的所有细节，而只是详细阐述对捕获完整系统行为至关重要的控制状态子集，以缩减状态空间和降低状态转换复杂性。这样，IPC 就不同于基于断言的标准验证（该验证需要设计人员向 RTL 代码添加局部信号级断言，并通过仿真或尽可能使用形式验证技术来进行检查）。很多情况下这种断言会检查设计人员在实施 RTL 代码相应部分时所推断的先决条件。这些局部断言与 IP 模块规范的关系可能不够清楚。此外，这种手动生成的断言在整个代码库中不规则分布，难以分析其是否覆盖了整个设计功能。

下面，对所有可能的状态转换进行自动全面的探索，并对其相应的属性予

间隔属性检查的工作原理

下例给出了简单 SVA 式属性的伪代码，它描述了处理器中 ADD 指令的操作情况。我们可以看到，在时间为 0 时，ADD 指令及其两个操作数被提取，两个周期后，寄存器读数准确更新为两个操作数之和。

```
Property
ADD_INSTRUCTION;
  t##0 ready_to_issue_instr() and
  t##0 decode(ADD,op1,op2,res)
implies
  t##1  pc_updated() and
  t##2  reg(res)=reg(op1)+reg(op2)
endproperty
```

以验证。任何属性失效会突出显示，并提供导致失效的反例细节。带有 OneSpin 360 MV 形式验证工具的操作式 IPC 还能自动进行完整性验证分析。这也可以采用同样先进的用于验证 CSM 属性的属性检查技术（采用高效的布尔可满足性解算器）来完成。

完整性证明能够确保每个可能的输入序列都有唯一的操作属性链来确定设计行为。链的属性通过 CSM 中的抽象开始状态和结束状态链接在一起，而输入触发器则匹配考虑中的各个输入迹线。此外，分析过程中要检查每个属性是否分别指定了特定时间间隔内所有相关的输出信号，并确保各时间间隔彼此相连，不会存在输出行为描述上的任何漏洞。

这样，证明完整性时就能整体检查属性集。如果检查通过，就不会出现属性集无法确定设计行为的输入情况。因此，属性集也能涵盖完整的设计功能。

随着越来越多地要求对嵌入式固件及其与硬件互动进行验证，采用形式验证方法始终是一个挑战，也需要不断积极研究。分别对固件和硬件组件进行验证一直是标准做法，但这最终需要花费大量时间进行全系统集成。此前，验证工程师采用间隔属性检查，主要是用于硬件子系统的形式验证。不过近期调研发现，已找到将这种方法扩展应用于含有硬件和固件的更完整系统上的途径。这项工作的重点就是通过将 IPC 应用到赛灵思商用 IP 软核（即软错误缓解 (SEM) 内核）来评估描述固件、硬件及二者之间互动情况的统一形式验证环境 (*http://www.xilinx.com/cn/products/intellectual-property/SEM.htm*)。该内核包括寄存器传输级 (RTL) 逻辑和一个 PicoBlaze™ 微控制器。

SEM 内核

为更好地了解验证工作，我们先进一步了解一下这个 IP 软核。这个赛灵思 IP 软核不仅能检测、归类并纠正赛灵思 FPGA 配置存储器中的错误，而且还可为测试目的注入错误。

新型半导体产品容易受到高能级辐射的干扰。比方说，高能中子产生单粒子翻转 (SEU)，直接影响芯片，进而导致配置存储器单元状态的变化。为了缓解上述影响，赛灵思 FPGA 配置帧的存储器单元都采用单错误纠正 / 双错误检测硬件模块保护机制。对航空器仪表等特定应用而言，还应采取更为精密的纠错机制，以抵御极高能级辐射带来的影响。SEM 内核可通过赛灵思 FPGA 中的 FRAME_ECC 端口来监控纠错码 (ECC) 模块的状态，然后针对这些情况提供适当的解决方案。

SEM 内核采用赛灵思 PicoBlaze 微控制器来监控配置存储器单元的状态，并根据需要采取纠错措施。设计人员可将 SEM 内核集成到设计中，并与设计中的其它电路系统组合在一起，实现更高级的防辐射保护机制，满足应用需求。如果检测到配置存储器错误，SEM 内核能直接纠正，或将错误情况通报给更高一级的系统设计。

正确操作 SEM 内核至关重要，因为其唯一目的就是确保用户 FPGA 电路的准确性。赛灵思已对该内核进行了全面验证，不过应当指出的是，由于种种原因，该 IP 的验证确实非常艰难。

该内核与 UART、前面提到的 FRAME_ECC、FPGA 的内部配置访问端口 (ICAP) 和定制错误注入接口等接口进行通信。虽然我们对这些接口了如指掌，但却很难以各种可能的输入组合加以操作，让嵌入式 PicoBlaze 微控制器唱重头戏。SEM 内核功能的形式验证要求我们捕获如下三者之间的互动情况：用汇编代码编写的 PicoBlaze 固件、封装逻辑以及规范文档中所述系统资源的外部接口。

为完成上述任务，我们采用 IPC 来验证 SEM 内核中与硬件相关的软件的准确性。

采用 IPC 对固件和硬件进行形式验证

在对总线协议的启动代码或驱动程序及其运行所在的硬件等低级固件进行形式验证时，设计团队往往面临着巨大挑战。近期，德国凯泽斯劳滕大学电子设计自动化集团的一个团队介绍了一种将 IPC 扩展应用于到软硬件协同验证的方法 [3]。其中要解决的难题就是如何处理包含成百上千代码行代码的状态空间爆炸问题。

该团队的主要观点是，利用间隔属性抽象为有限状态序列集，这些序列集用大量代码行表示，在此基础上进行操作式属性检查。这样，该技术可在单个概念状态机中将软硬件事件结合在一起。通用计算模型采用 IPC 方法，首次支持软硬件交互表示和调试。当然，这种方法也存在局限性，也不适用于所有类型的软件。特别需要指出的是，大量使用递归的代码不在当前讨论范畴之中。

验证 SEM 内核时，我们采用固件控制流程图 (CFG) 来生成属性模板。基本模块之间的每个转换都被视为独立的属性，由 PicoBlaze 微控制器内置的寄存器或外部事件所触发。给定周期中描述抽象开始 / 结束状态的功能仅取决于 PicoBlaze 架构状态及任何外部刺激。

IPC 需要描述 SEM 内核在断言开始 / 结束时的状态，这时我们需要检查 PicoBlaze 固件和 FPGA 逻辑的 RTL 代码。图 1 显示了固件和硬件状态的组合。需要注意的是，PicoBlaze 微控制器在真正实现软件有限状态机，且其行为直接影响到封装硬件。如果可能，单独验证固件需要一个硬件总线功能模型 (BFM) 接口，实际上这会产生又一个行为测试平台。不过，

IPC 的扩展使我们能在统一的验证环境中对包含硬件和固件的全系统的行为进行建模。我们本来能在指令集仿真器中运行固件并对其进行编译，但却无法在一个环境中全面捕获硬件和固件系统行为。而使用 IPC，我们则可以在统一的硬件 / 固件验证环境实现上述操作。

在构建属性时，我们可反复限制其复杂性，从而在 OneSpin 360 MV 工具中我们就任何给定属性的复杂性及其评估时间进行折中，实现最佳平衡。在本例中，我们发现让属性的间隔在 100 个周期以下比较好，这样属性验证可在 30 分钟内完成。

SEM 内核还涉及到更深层次的设计因素，也有助于降低属性复杂性。SEM 内核固件和 PicoBlaze 微架构的有关方面以及如何实施以简化属性创建等，都与此相关。

首先，我们可利用 SEM 内核嵌入式固件的某些特性：

- 固件镜像可实现软件有限状态机，其某些特性有助于正式描述处理器状态。特别需要指出的是，固件不会动态分配存储器，也不会调用无限递归。这是一般低级嵌入式软件的典型情况。固件的这两大特性能大幅简化处理器状态及其存储器的建模。
- 向验证工程师提供全局变量和局部变量的详尽内存映射。SEM 内核可提供封装 RTL 中用于触发固件状态转换的全部外部变量，以便配合供 OneSpin 360 MV 工具，共同探索。
- 最后，固件机代码存储在 SEM 内核的 ROM 中。由于 ROM 可提供可视化验证流程，因此无需全面验证 PicoBlaze 微控制器上可运行的任何程序，而只需验证 ROM 中实际存储的程序。换言之，经现场验证适用于所有固件的 PicoBlaze 微控制器，我们不必再次验证。我们可集中精力验证 PicoBlaze 微控制器与 SEM 内核的固件以及 wrapper RTL 之间的互动情况。

其次，我们还可利用 PicoBlaze 微架构的特性来描述固件镜像的状态。PicoBlaze 微架构的一些特性能简化其在形式验证工具流程中的集成。

- 指令的执行井然有序。由于 SEM 内核中指令执行连续进行，因此我们能确切知道固件内某条指令相对于其它指令的启动时间。
- 每条指令的解码或执行需要两个周期。由于 SEM 内核固件工作中时延是确定的，因此我们能创建出囊括多条指令的属性，而这些指令则能根据确定的总时延加以执行。
- PicoBlaze 微架构支持有限堆栈深度。堆栈内容是 SEM 内核状态的关键部分，但有限深度情况下，该状态不会超过设定的深度。由于属性验证的复杂性随状态数量的增加而增大，因此堆栈内部设定的深度可简化验证工作。

描述某个周期内处理器的状态时，架构状态数量的描述相对简单。这种可管理的状态描述可直接向 OneSpin 360 MV 的属性生成工具提供信息。

有了这些简化因素，我们就要描述相关的状态转换了。我们可将作为各个设计状态的固件基本模块映射后直接描述。从定义上看，基本模块包括线性指令序列。每个条件跳转或条件函数调用都可决定一个基本模块或两个潜在后续模块的终点。指令的目标地址标志着新基本模块的起点。控制流程图包含基本模块及后续关系。图中每个边缘都对应于固件的条件分支，并标明分支条件。

如果用高级语言来实现固件，则编译器可自动生成 CFG。不过，借助（符号）指令集仿真器，我们同样也可从汇编代码中抽象 CFG。该技术还能支持仅提供汇编代码的传统平台的协同验证。

验证 SEM 内核时，我们采用固件 CFG 来生成属性模板。基本模块间的每次转换都视为 PicoBlaze 内置寄存器或外部事件触发的独立属性。任何给定周期中描述抽象开始 / 结束状态的功能仅取决于 PicoBlaze 架构状态和所有外部刺激。

作为外部刺激到架构状态映射的

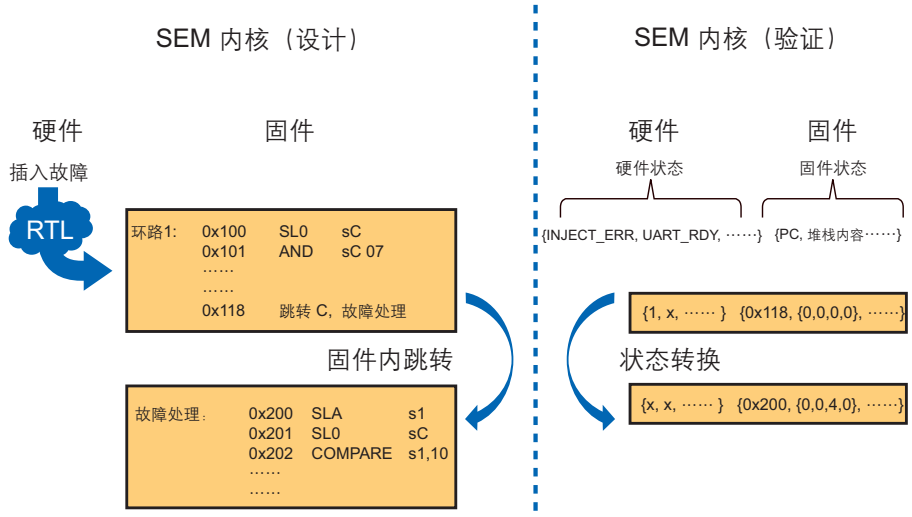


图 1 - OneSpin 360 MV 工具中 SEM 内核的硬件和固件状态转换为组合验证状态的示例

PicoBlaze 微架构的一些特性能简化其在形式验证工具流程中的集成。指令的执行井然有序，执行每条指令需要两个周期。此外，有限堆栈深度这一特性也能简化验证工作。

一部分，实践证明我们必须指定每个条件跳转或函数调用的分支条件。我们关心的是设计的整体行为，因此我们要指定外部输入事件或封装电路重要的状态寄存器（而不是嵌入式处理器的局部状态寄存器）的条件。触发条件下评估的封装寄存器可成为抽象状态定义的一部分。

SEM 内核固件和 PicoBlaze 微控制器本身都能进行条件简化，因此我们能就处理器状态和外部刺激定义所有固件状态转换。这种状态涵盖固件和硬件，可将设计的整体行为与 OneSpin 360 MV 工具中的抽象概念有限状态机相关联。

属性的生成

我们发现，SEM 内核的固件和硬件均能用 1700 种属性描述，这些属性捕获了设计及其接口的端对端功能。我们用 OneSpin 360 MV 来检查属性，并探索整个属性集的完整性。属性能并行验证，服务器集群中属性验证最长大概需要 30 分钟才能完成。

乍然一看，总共 1700 个属性好像很多。不过，大多数属性（约 1500 个）的验证只涉及顺序 UART 的等待循环，顺序 UART 除转存控制器的状态信息外，主要用于调试目的。切记，每个属性都是一个以条件跳转结束的基本软件模块，因此每个 UART 等待循环都会在形式模型中创建唯一的属性。

就设计的全面验证来说，我们发现等待循环期间无法预见的负面效果不会破坏抽象状态内容。

总之，生成的属性贯穿固件的整个控制流程，描述了相应固件基本模块执行过程中设计的输入 / 输出行为。就本案例研究而言，上述属性配合现有的验证测试平台，可让您对内核功能的准确性信心百倍。

生成的属性还反映出一个情况，即 SEM 内核错误注入测试特性可将错误注入到某个配置存储器位置，不过只有在不按 SEM 内核产品文档建议的方式操作错误注入端口时才会发生此类情况。虽然该问题在正常操作条件下不会发生，但赛灵思还是更新了 SEM 内核特性，从而彻底杜绝了这种可能性。

IPC 源于欧洲

操作式间隔属性检查技术来自欧洲 Infineon Technologies 的工业研究。内核技术由 OneSpin Solutions 推出并实现了商业化，该公司还在不断开发和完善该技术，并提供了名为 360 MV 的全面 CAD 环境，让该技术更加适合非专业人士的需要。自 IPC 问世以来，德国凯泽斯劳滕大学的研究人员就同 Infineon Technologies 和 OneSpin Solutions 积极合作，进一步推进 IPC 的基本理论，并推广其应用。

致力于打造高质量

在我们的调研中，我们演示了用于形式验证赛灵思 SEM 内核中高度集成的硬件和固件的 IPC 方法。在统一的验证环境中，用 1700 个并行验证的属性对 SEM 内核进行了全面验证。在验证过程中，采用了最新形式验证技术和工具。验证结果则能让您对 SEM 内核的功能准确性信心大增，同时突现了赛灵思继续致力于打造高质量 IP。

References

1. Jones, R.B.; O'Leary, J.W.; Seger, C.-J.H.; Aagaard, M.D.; Melham, T.F.; "Practical formal verification in microprocessor design," Design & Test of Computers, IEEE, vol. 18, no. 4, pp. 16-25, July/August 2001

2. Nguyen, M.D.; Thalmaier, M.; Wedler, M.; Bormann, J.; Stoffel, D.; Kunz, W.; "Unbounded protocol compliance verification using interval property checking with invariants," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 27, no. 11, pp. 2068-2082, November 2008

3. Nguyen, M.D.; Wedler, M.; Stoffel, D., and Kunz, W.; "Formal hardware/software co-verification by interval property checking with abstraction," Design Automation Conference (DAC), 48th ACM/EDAC/IEEE, pp. 510-515, June 5-9, 2011

新的工具可简化 FPGA 综合

如何通过 RTL 分析、SDC 约束和综合向导以更快的速度推出更出色的 FPGA 设计

作者：Shakeel Jeeawoody
市场营销副总裁
Blue Pearl Software 公司
shakeel@bluepearlsoftware.com

大多数 FPGA 设计人员都充满热情地开展专业化问题解决和创造性工作，当然，他们工作压力也相当大，工作流程也非常单调乏味。幸运的是，EDA 公司和 FPGA 厂商不断开发新的工具和方法，推进繁琐任务的自动化，帮助设计团队集中精力做好创造性工作。下面我们就来看看 FPGA 工具流程的演进发展，了解一下现代 FPGA 团队是如何利用 RTL 分析、约束生成和综合导向来减少设计迭代的。

如果您已经是一名 FPGA 设计专业人士，那么将拥有辉煌的职业发展前景，因为越来越多传统上需要 ASIC 实现的设计现已改用 FPGA。随着新一代芯片工艺技术的推出，设计 ASIC 的成本正呈几何级数增加。与此同时，FPGA 厂商则能利用最新工艺技术实现新一代产品，且不会让客户承担过重的成本负

担。

但不容乐观的是，FPGA 设计相当复杂，需要跟 ASIC 流程一样复杂的工具流程，这往往需要整个设计团队的共同努力才能完成，而不能光靠一名设计人员。因此，FPGA 设计团队在着手 ECO 或新项目之前应认真分析现有的工具套件。那么好消息呢？就是新一代 EDA 工具如雨后春笋般涌出，可助他们一臂之力。设计人员可选择采用标准数据格式且易于安装和使用的工具，简化流程集成工作，而且能够在选定的平台（不管是 Windows 还是 Linux）上实现本机运行。

FPGA 工具流程的发展演进

这些年来，FPGA 设计日趋复杂，工具流程也随之发展，而且越来越像 ASIC 流程。上世纪 90 年代，FPGA 流程（见图 1 的流程 A）跟当时的简易 ASIC 流程一样，最初以 RTL 为基础，并采用综合及布局布线工具。随着设计变得进一步复杂化，FPGA 团队在流程中增加了时序分析功能，帮助客户确保设计能按指定的频率运行。今天的 FPGA 已经发展为庞大的系统平台，设计团队通常要通过 RTL 分析来最小化设计迭代，并确保设计能够实现相应的性能目标。

进而言之，由于今天的 FPGA 设计项目非常庞大复杂，所以设计人员需要想尽一切办法更好地了解设计的规模和复杂性，以便更好地控制流程中的工具，加速设计上市进程。现代 FPGA 设计团队正在采用一种新型方法，那就是在整个设计流程中贯穿约束机制。我们不妨看看当下最流行的、现已得到赛灵思最新 Vivado ™ 流程支持的一种约束方法——Synopsys 设计约束 (SDC) 格式，以及了解如何通过 SDC 让设计项目受益。

什么是 SDC？

SDC 是一款基于 Tcl 的格式，可用来设定设计目标，包括设计的时序、功耗和面积约束。一些产品能读取或写入 SDC。一些示例 SDC 约束包括时序约束（如创建时钟、创建生成时钟、设置输入延迟和设置输出延迟）和时序例外（如设置错误路径、设置最大延迟、设置最小延迟以及设置多周期路径）。这些 SDC 约束通常应用于寄存器、时钟、端口、引脚和网络（连线）等设计对象。

需要指出的是，尽管 SDC 是标准化格式，但生成的 SDC 和读取 SDC 之间还是略有差异（不同工具之间有差异）。了解这些差异并积极采取措施，有助于避免意外情况的发生。

SDC 不应过于复杂

SDC 最常见的应用就是约束综合。一般说来，设计人员要考虑设计的哪些方面需要约束，并为其编写 SDC。设计人员通常要执行流程 B 中描述的流程，首次肯定无法进行时序收敛。随后要反复手动盲目尝试添加 SDC，以实现时序收敛，或让设计能在指定的频率上工作。许多从事过上述工作的设计人员都抱怨说设计迭代要花好几个星期，往往会拖延设计进程。

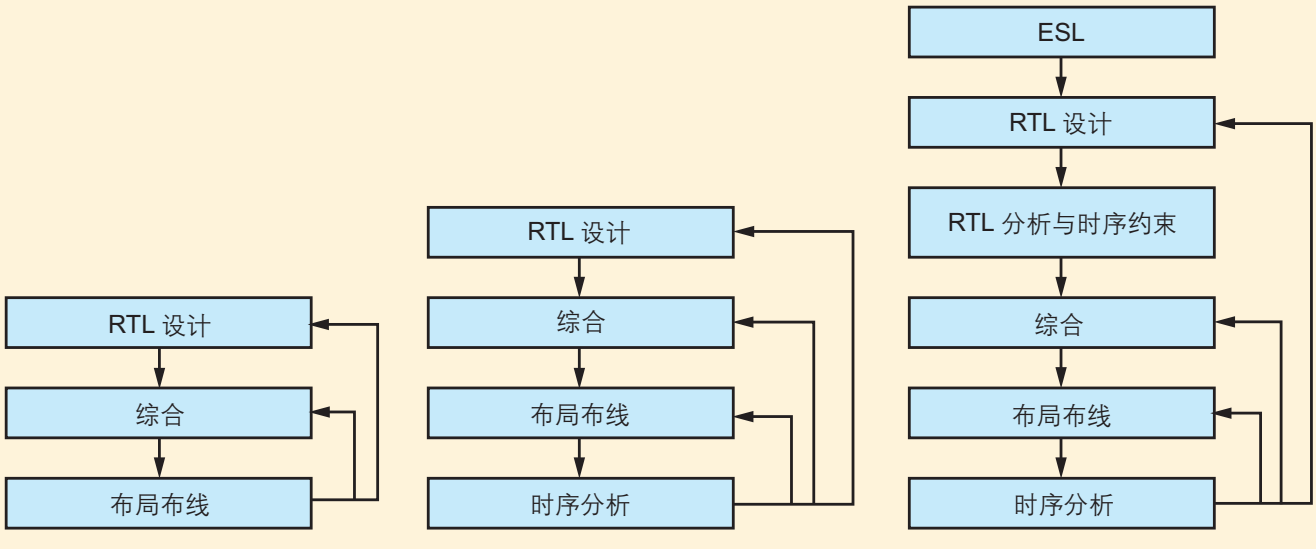


图 1 - FPGA 工具流程随时间不断发展，越来越像 ASIC。

迭代的另一个问题在于，数名设计人员可能在不同的地点为 SDC 设计不同的模块。这样设计工作会变得非常复杂，设计团队必须想办法验证 SDC，避免在芯片级封装阶段出现层级名称的冲突。要确保进行有效的设计协作，就必须采用适当的工具和方法。

流程 C 是现代化流程，除了流程 B 的工具之外还采用了分析、SDC 约束和高层次综合技术，在解决上述问题方面发挥了重大作用。

综合向导

对典型的 FPGA 设计而言，综合解决方案还处于探索阶段，不管是面积、速度还是功耗的优化，都存在多个局部最大值和局部最小值。利用智能向导，我们能实现最佳解决方案，避免综合工具聚集到任意的局部最小值。最有效的向导之一就是采用错误路径和多周期路径，避免综合工具为不必要的组件浪费宝贵的优化时间。

不过，找到设计中的所有错误路径 (FP) 和多周期路径 (MCP) 并不容易。花上足够的时间，我们能找到一些简单的 FP 和 MCP，不过一些涉及状态机和计数器的复杂 FP 和 MCP（特别是在多个层级中）则很难找到。幸运的是，FPGA 设计人员可采用 Blue Pearl Software 等创新公司推出的工具执行自动化 FP 和 MCO 生成，从而确保完整性、全面性和准确性。此外，这些工具还能为每个 FP 和 MCP 提供不同的机制，包括原理图、断言和审核路径，从而让用户验证其正确性。

由于 FPGA 厂商和商用 EDA 厂商的合作进一步加强，采用通用接口，设计团队就能够将 Blue Pearl 软件套件集成到他们所青睐的工作流程中。既然赛灵思的最新 Vivado 设计套件支持 SDC，那么在不同工具之间沟通设计意图就变得极其简单（图 2）。

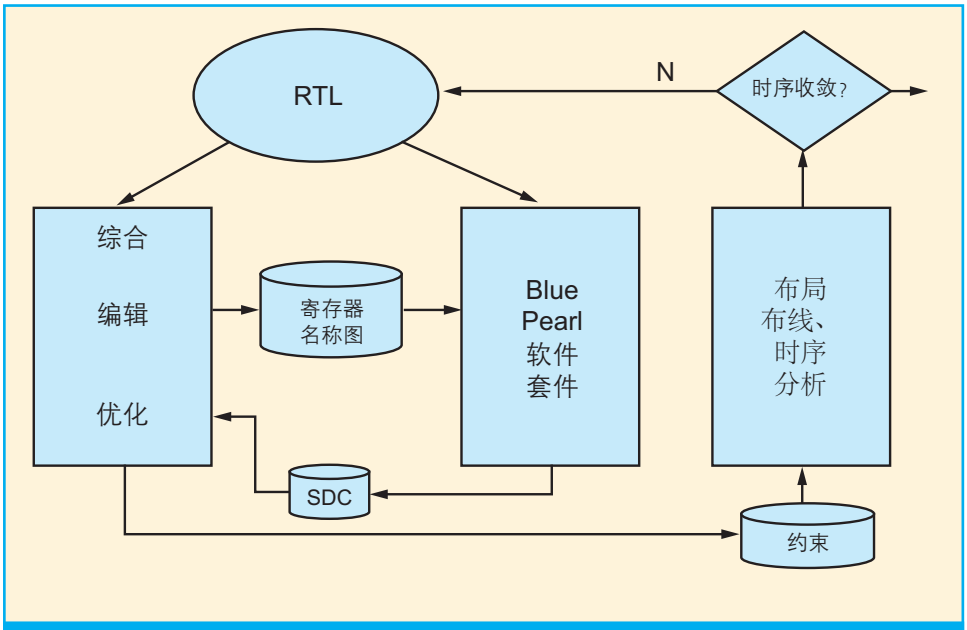


图 2 - Blue Pearl 软件套件与赛灵思的 Vivado 设计套件如何协同工作。

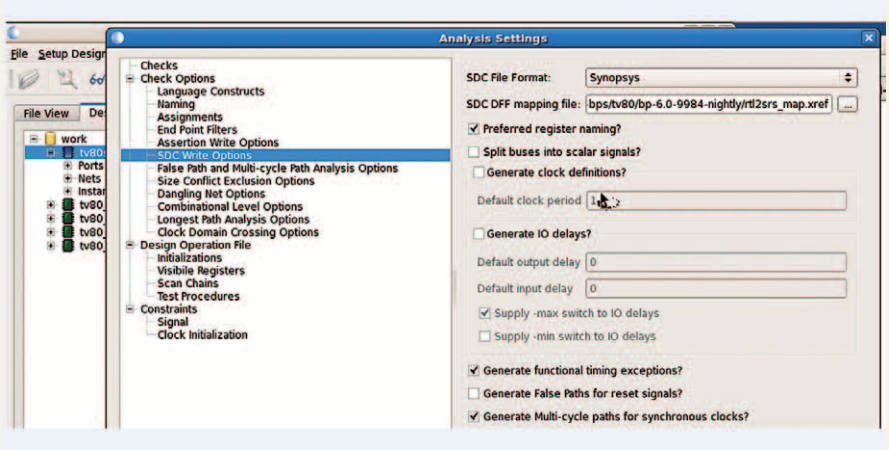


图 3 - Blue Pearl 软件可简化格式问题。

除了与赛灵思及其他 FPGA 厂商协作外，Blue Pearl 公司还同 Synopsys 开展密切合作。这两家公司共同研究如何让综合工具接受尽可能多的自动生成的 SDC，同时避免设计人员进行任何手动修改。由于 SDC 格式对不同工具的使用差异很小，因此工作团队快速明确命名方案是顺利实现互操作性的一大挑战。

这里的解决方案是在综合的第一阶段（编译）后截取映射名称，在 Blue Pearl 软件套件的 SDC 生成工具中使用名称（见图 3），并为综合工具的第二阶段（优化）提供适当的 SDC。该方法给 FPGA 设计人员提供了一个最佳解决方案，无需花时间处理格式化问题。

以下给出非优化型约束编写示例:

Blue Pearl 软件套件可将我们的示例设计的 QoR 提升 20%，该示例采用多个 IP 核，其中包括 Verilog 的 R1200 和 VHDL 的 AES 加密。

set_false_path -from [get_cells {i_tv80_core.SP[*]}] -to [get_cells {i_tv80_core.i_reg.RegsL}]

优化后则为：

set_false_path -from [get_cells {i_tv80_core.SP[*]}] -to [get_cells {i_tv80_core.i_reg.RegsL_2[7:0]}]

能取得哪些实际的效果?

Blue Pearl 软件套件能实现一些任务的自动化，设计人员对其结果质量 (QoR) 很满意。表 1 显示了用 Blue Pearl 软件套件自动生成 SDC，能将示例设计的 QoR 提升 20%，该示例采用多个 IP 核，其中包括 Verilog 的 R1200 和 VHDL 的 AES 加密。

运行 1 未采用 Blue Pearl 软件，结果没有实现时序收敛。设计人员用 RTL 设计或工具约束进行迭代以满足 60MHz 的要求很容易就要花上好几个星期的时间。在运行 2 中，Blue Pearl 软件套件几分钟就能生成 SDC，而自动生成的 SDC 足以指导下游工具满足时序要求。

显然，对 FPGA 设计人员来说，降低压力、简化工作的一个好办法就是跟别人一样添加 RTL 分析、SDC 生成和综合向导工具。如欲了解更多信息，欢迎访问：*www.bluepearlsoftware.com*。

	运行 1	运行 2
流程	类似于流程 B（见图 1）	类似于流程 C（见图 1）
涉及的工具	Synopsys：Synplify Pro 赛灵思：Virtex-5、XC5VLX50、FF1153,-1	Synopsys：Synplify Pro 赛灵思：Virtex-5、XC5VLX50、FF1153、-1 Blue Pearl 软件：6.0 版
设计频率	用 Synplify Pro 全局设为 60 MHz	用 Synplify Pro 全局设为 60 MHz
Blue Pearl SDC	未采用	采用
设置违规（布局布线后）	-3.57 纳秒	无

表 1 - 两种运行相比较，反映出工作流程中添加 Blue Pearl 套件的优势。

应用指南

如果希望进一步了解 FPGA 如何适用于广泛应用，我们建议阅读以下应用指南。

XAPP588：VIRTEX-5QV FPGA 外部配置管理

http://www.xilinx.com/cn/support/documentation/application_notes/xapp588_v5qv_ex_config_mgmt.pdf

赛灵思 Virtex®-5QV 是一款采用耐辐射设计的高逻辑密度器件，其包含 12 个晶体管 (12T) 配置存储器单元。Virtex-5QV FPGA 不仅是高可靠性空间应用的最高速、最大型的 FPGA，同时也是轨道应用的首选可编程逻辑器件。该 FPGA 的翻转率极低，每年仅为发生 5 次粒子翻转。不过，如 Y.C. Yang 在本应用指南中指出的那样，设计人员还能根据需要实现配置管理，通过定期进行软重启以确保功能长期可靠运行。

在轨道应用中，由于高能带电粒子会导致可编程逻辑器件的配置存储器出现单粒子翻转 (SEU)，从而给应用带来了极大挑战。一旦 SEU 影响到设计最核心的存储器单元，最终就会影响到应用功能。此外，FPGA 控制电路系统中的 SEU 还会导致单粒子功能中断 (SEFI)。既然这两种情况都不可取，因此设计人员应想办法通过配置管理来恢复正常的预期功能。

本应用指南介绍的配置管理方案采用外部器件来监控和清理 Virtex-5QV FPGA，以降低 SEU 和 SEFI 风险。外部器件应当是符合空间应用规范要求的 FPGA 或 ASIC，当然也可能是微处理器，负责执行 SEU 检测与校正、部分重配置、盲擦 (blind scrubbing) 以及 SEFI 检测与缓解。

参考设计采用 Virtex-5QV 器件放射性测试过程中使用的配置监控 IP。由于 Virtex-II Pro FPGA 是赛灵思测试装置中的外部配置管理器，因此 Yang 用 ISE® 10.1.03 软件实现了该参考设计。不过，参考代码应完全可移植到今后更高的 ISE 版本上，因为它要负责推断设计原语，而且要在 XST 中可综合。

XAPP553：可扩展 SERDES 成帧器接口 (SFI-S) 支持 7 系列 FPGA

http://www.xilinx.com/cn/support/documentation/application_notes/xapp553-scalable-serdes-framer-interface.pdf

可扩展 SERDES 成帧器接口 (SFI-S) 是一款光学互联论坛 (OIF) 标准，定义了典型光学通信线路卡上器件之间的电子连接。n 位宽 SFI-S 配置包含 n 个数据通道和一个控制通道，用于接口歪斜补偿。Julian Kain 撰写的应用指南介绍了一种专门针对基于 GTX 或 GTH 串行收发器的赛灵思 7 系列 FPGA 的 10 数据通道 SFI-S 设计，用于实现总带宽速率高达 111.8 Gbps 的双向接口。经硬件验证的 Verilog HDL 参考设计可提供出色的歪斜补偿，并能精细控制歪斜跟踪。Verilog HDL 源码能根据需要随时修改。采用 PRBS31 生成器和检查器逻辑的可综合示例设计支持简单

的仿真和参考设计的硬件演示。

XAPP521：实现赛灵思视频流接口与 AXI4-STREAM 协议的桥接

http://www.xilinx.com/cn/support/documentation/application_notes/xapp521_XSVI_AXI4.pdf

ARM® 内核 AMBA® 规范（4.0 版本）AXI 互联标准包括三种高级可扩展接口 v4 (AXI4) 互联协议，分别为 AXI4 互联、AXI4-Lite 和 AXI4-Stream。赛灵思 AXI 视频直接存储器访问 (VDMA) 内核提供 AXI4-Stream 接口支持视频数据。需要视频数据在存储器中缓冲的视频应用要通过 AXI4-Stream 接口连接到 AXI VDMA。前版赛灵思视频 IP 核采用赛灵思视频流接口 (XSVI) 视频数据协议。Steve Elzinga 和 Chris Martin 共同编著的应用指南介绍了如何将 XSVI 与 AXI4-Stream 接口桥接，使具有赛灵思视频 IP 核及 XSVI 接口的视频设计能采用 AXI VDMA。两位作者还提供了一个嵌入式参考设计来介绍上述桥接机制的功能。如果赛灵思视频 IP 不包含 AXI4-Stream 接口，那么设计人员可用该解决方案来构建视频系统。

XAPP888：MMCM 和 PLL 动态重配置

http://www.xilinx.com/cn/support/documentation/application_notes/xapp888_7Series_DynamicRecon.pdf

本应用指南介绍了一种可动态修改赛灵思 7 系列 FPGA 混合模式时钟管理器 (MMCM) 时钟输出频率、相位移以及占空比的方法。同样，它还介绍了如何通过动态重配置端口 (DRP) 来修改锁相环 (PLL)。作者 Jim Tatsukawa 介绍内部 DRP 控制寄存器的行为后，紧接着提供了一个采用状态机驱动 DRP 的参考设计，确保寄存器得到有序控制。参考设计支持两个重配置状态地址，不过由于其模块化特性，设计人员能对其进行扩展以支持更多状态。每个状态都对 MMCM 或 PLL 进行全面的重配置，从而可修改大部分参数。该设计占用的 7 系列 FPGA 资源极低，仅需要 27 个 Slice。

XAPP520：7 系列 FPGA 高性能 I/O BANK 接口采用 2.5V 和 3.3V I/O 标准

http://www.xilinx.com/cn/support/documentation/application_notes/xapp520_7Series_HPIO.pdf

赛灵思 7 系列 FPGA 的 I/O 可分为高范围 (HR) bank 和高性能 (HP) bank 两大类。HR I/O bank 工作电压为 1.2~3.3 V，而 HP I/O bank 经优化，工作电压为 1.2~1.8V。本应用指南介绍了 7 系列 HP I/O bank 接口使用 2.5V 和 3.3V 电压的情况。不同的接口选择取决于性能需要、功能和信号类型（输入、输出或双向）。作者 John Rinck 和 Austin Tavares 探索了添加电阻、场效应晶体管 (FET) 开关、电平转换器和其它赛灵思 FPGA 的不同选择。这些策略相结合，几乎能满足所有设计、成本和性能需求。

XAPP517：ICAP 配合 SEM 控制器的双重用途

http://www.xilinx.com/cn/support/documentation/application_notes/xapp517_DualUse_ICAP_SEM.pdf

小 John Ayer 撰写的该应用指南介绍了 Spartan®-6 和 Virtex-6 器件中用户设计和软错误缓解 (SEM) 控制器之间共享内部配置访问端口 (ICAP) 的方法。这种共享可实现不同的目的。参考设计采用了回读器件 IDCODE 示例。这种方法同样能帮助客户创建需要访问 ICAP 的多重启动设计。

本设计表明，通过采用 ICAP 控制信号上的多路复用器，仅在控制器闲置状态下切换到脱离 SEM IP 的控制，我们就能将 SEM 控制器使用的 ICAP 与其它制定用户功能共享。采用赛灵思 ML605 和 SP605 参考板以及 ChipScope™ Pro 分析器，SEM 控制器将释放 ICAP ，从而让器件回读 IDCODE，再次控制并成功纠正注入错误。

赛灵思工具和 IP 更新

赛灵思正在不断改进其产品、IP 和设计工具，努力帮助设计人员提高工作效率。我们在此汇报旗舰 FPGA 开发环境（ISE® 设计套件）以及赛灵思 IP 的最新升级情况。通过产品升级，ISE 设计套件的逻辑、嵌入式、DSP 和系统四大版本将得到显著改进并新增一些新特性。保持 ISE 及时更新升级是确保最佳设计结果的简单方式。

2012 年 5 月 8 日，赛灵思发布了 ISE 14.1 设计套件，您可从赛灵思下载中心：www.xilinx.com/cn/download 下载。ISE 设计套件的各个版本现均已公开提供包含 XA Zynq-7000 等在内的 Zynq™-7000 EPP 系列。ISE 以比特流生成的方式为其编程提供支持。另外，Virtex®-7 XT FPGA 系列现也支持比特流生成。ISE 的最新版本现可对 ArtixTM-7 FPGA GTPE2 提供支持。

如需了解 ISE 设计套件的更多信息或下载 30 天免费评估版，敬请访问：www.xilinx.com/cn/ise。

ISE 设计套件：逻辑版本

Front-to-Back FPGA 逻辑设计

最新版本号：14.1

最新发布日期：2012 年 5 月

前一版本：13.4

最新版亮点：

集成 Mentor Graphics 的 ModelSim 和 Questa 高级仿真器：PlanAhead™ 设计工具现在允许用户在项目设置中选择这些 Mentor Graphics 产品作为目标仿真器。仿真需要库编译，可使用 Tcl 命令“compxlib”来完成，这样用户可以使用多个带有自身属性集的仿真文件集。

集成嵌入式开发套件：PlanAhead 设计工具现在可以使用 .xmp 源文件类型创建并向项目添加 Xilinx Platform Studio 子系统。双击 .xmp 源文件类型就可以启动 Platform Studio 来生成和定制嵌入式子系统。

ISE 设计套件：嵌入式版本

集成嵌入式设计解决方案

最新版本号：14.1

最新发布日期：2012 年 5 月

前一版本：13.4

最新版亮点：

所有 ISE 设计套件版本都包含上述逻辑版本所涉及到的改进。嵌入式版本现在也提供对 Zynq-7000 EPP 的支持，供开发裸机产品和基于 Linux 的产品，以及 MicroBlaze™ 处理器更新。其它改进如下。

新型低时延中断模式：控制器直接提供中断向量，根据系统设计，最高可将时延响应降低 10 倍。

新的互换指令：提供用于字节和半字长更换的指令，有助于支持 AXI 大端字节和 AXI 小端字节之间的字节序转换。

更多器件支持：赛灵思现在已在整个 7 系列产品上对 MicroBlaze 处理器进行了验证。

系统高速缓存：嵌入式版本现在在 MicroBlaze 处理器和外部存储器控制器之间为基于 AXI 的系统添加了新的嵌入式系统高速缓存 IP 外设。MicroBlaze 处理器将该系统高速缓存 IP 核用作二级高速缓存，根据具体的系统因素、设计类型或连接点，可起到降低时延和提升性能的作用。

I/O 模块：赛灵思将一系列新的可配置通用嵌入式处理器外设封装在在单个 IP 模块中，用于连接至 MicroBlaze 处理器的数据侧 LMB 总线。这样可简化标

准微控制器系统的定义、配置和部署，便于用户无缝地将 MicroBlaze 处理器的微控制器系统（MCS）设计从逻辑版本移植到嵌入式版本。

关于 ISE 嵌入式版本更新的信息，敬请访问：*www.xilinx.com/cn/embedded*。

ISE 设计套件：DSP 版本

针对高性能 DSP 系统

最新版本号：14.1
最新发布日期：2012 年 5 月
前一版本：13.4

最新版亮点：

所有 ISE 设计套件版本都包含上述逻辑版本所涉及到的改进。下列改进系针对 DSP 版本。

集成 System Generator for DSP PlanAhead：PlanAhead 设计工具现在可以通过 .sgp 源文件类型创建并向项目中添加 DSP 子系统。双击 .sgp 源文件类型，启动 MathWorks 的 Simulink® 工具来生成和定制 DSP 子系统。

性能提示和模块集：“性能提示”工具条可打开“高性能设计”文档。另外模块集现在为 BRAM 配置中的嵌入式寄存器提供 FIFO 支持。

赛灵思 IP 核更新

IP 核名称：ISE IP Update 14.1
IP 核类型：全部

目标应用：赛灵思不仅开发 IP 核，还与第三方 IP 核供应商合作共同帮助客户加速产品上市进程。赛灵思 FPGA 与 IP 核的强大组合可以提供类似 ASSP 那样的功能和性能，而且其灵活性比 ASSP 更高。

最新版本号：14.1
最新发布日期：2012 年 5 月

最新版亮点：

版本 14.1 对 IP 进行了改进并新增了一些 IP 核，旨在完善对 AXI、Zynq-7000 EPP 和 MicroBlaze 处理器的支持。

新款 CORE Generator™ 软件和 IP 核：

- AXI Quad SPI — 该 IP 核支持现场执行（XIP）模式和针对性能的架构改进。作为现有客户的默省选项，该核继续在传统模式下工作。
- AXI 性能监测器 — 该内核用于测量系统中特定主 / 从（AXI4、AXI4-Lite、AXI4-Stream）设备的总线时延、特定时间段内的存储器流量以及其它性能指标。
- 处理系统 7 — 这是供处理系统（PS）和可编程逻辑（PL）之间的 Zynq-7000 EPP 逻辑连接使用的封装器 IP，可辅助用户添加定制 IP 或其它 EDK IP。
- AXI 系统高速缓存 — 当用在 MicroBlaze 处理器和外部存储器控制器之间时，该内核可当作 MicroBlaze 处理器的二级高速缓存模块使用。
- 嵌入式 I/O 模块 — 赛灵思已经将这种在 MicroBlaze 处理器 MCS 中引入的通用 I/O 外设子集移植到嵌入式版本，以实现兼容性。

AXI4 接口支持：最新版本的 CORE Generator IP 经更新后，可支持质量级 AXI4 接口。关于 AXI IP 支持的更多详情，敬请访问：*http://www.xilinx.com/cn/ipcenter/axi4_ip.htm*。

- 关于 AXI4 支持的一般信息，敬请访问：*http://www.xilinx.com/cn/ipcenter/axi4.htm*。
- 关于 14.1 版本中 IP 核的全部列表，敬请访问：*http://www.xilinx.com/cn/ipcenter/coregen/updates_14_1_2012_1.htm*。
- 关于 IP“新特性和已知问题”的更多详情，敬请参考《IP 版本应用指南》（XTP025）：*http://www.xilinx.com/cn/support/documentation/ip_documentation/xtp025.pdf*。

文档浏览器

赛灵思文档浏览器设计用于帮助用户直观查找并下载根据其需求定制的技术文档。借助内建的警示机制，用户可以放心地了解到最新的信息。该警示系统可以通知用户何时有新文档供下载，下载的文档何时在网上进行了更新。用户可从赛灵思支持网站：*http://www.xilinx.com/cn/support* 上下载**赛灵思文档浏览器**。

赛灵思正式发货全球首款异构3D FPGA-Virtex®-7 H580T带来突破性带宽和信号完整性

Virtex®-7 H580T 带来突破性带宽和信号完整性

继实现 28nm 工艺节点，并为我们带来 All Programmable 全新设计套件 Vivado 后，赛灵思公司又在近日宣布正式发货 Virtex®-7 H580T FPGA —全球首款 3D 异构 All Programmable 产品。Virtex-7 HT 采用赛灵思的堆叠硅片互联 (SSI) 技术，是提供业界带宽最高的 FPGA，可提供多达 16 个 28 Gbps 收发器和 72 个 13.1 Gbps 收发器，也是唯一能满足关键 Nx100G 和 400G 线路卡应用功能要求的单芯片解决方案。结合赛灵思领先的 100G 变速机制（gearbox）、以太网 MAC、OTN 和 Interlaken IP，Virtex-7 HT 可为客户提供不同的系统集成度，从而满足他们在向 CFP2 光学模块转型时对空间、功耗和成本的要求。

异构架构的特点是能够把不同的技术结合在一起，从而提供针对客户需求而优化的器件。就 Virtex-7 H580T 器件而言，赛灵思选择了在 40nm 技术节点上已得到检验的成熟的收发器技术。这种异构化架构可以为核心 FPGA 和 28 Gbps 收发器芯片提供独立的技术选项，从而避免浪费系统功耗和对计算任务毫无助益的高漏电晶体管对 FPGA 造成的负担。而在芯片上采用 28 Gbps 收发器且独立于核心 FPGA 架构，进一步实现了卓越的噪声隔离功能，实现最佳的整体信号完整性和系统空间余量，并针对设计收敛和更快上市，大大提升了生产力。

从使用角度来看，除了需要 Vivado 设计套件之外，异构架构不会给设计方法带来巨大影响。这种架构的关键特点之一就是我们能够依照自然分区确立每个裸片（器件的各个裸片）的边界——这在传统的单芯片 FPGA 架构中通常要走长线。这就意味着我们不用在设计工具上花费很多精力以适应器件需要。同时，我们的客户也不必对设计方法或流程进行重大调整。

赛灵思公司有线通信系统架构师 Mark Gustlin 指出：“Virtex-7 H580T 具有 8 个 28 Gbps 收发器和更大的逻辑容量，是唯一一款可集成更多线路卡功能的 FPGA 器件，使设计人员能够在单芯片上实现双 100G OTN 转发器。与 Virtex-7 H580T 相比，以 ASSP 为基础的解决方案还有一年多才会面世，而且需要 5 个器件来实现同等功能，此外功耗至少增加 40%，成本增加 50%。”与竞争对手相比，赛灵思所推出的是一个完整的、采用硅中介层技术 All Programmable 产品，并已面向客户出货。

依元素科技培训课程时间表 2012/7 至 2012/9

培训课程	培训时间	7 月	8 月	9 月
使用 7 系列产品进行设计	2 天	2-3 日 北京	2-3 日 深圳	3-4 日 北京
使用 PlanAhead 分析与设计工具进行高级设计	2 天	4-5 日 北京	5-6 日 深圳	
Xilinx 部分重配置工具和技术	2 天	5-6 日 深圳	20-21 日 北京	10-11 日 北京
利用 Spartan-6 和 Virtex-6 系列进行设计	3 天	18-20 日 深圳	8-10 日 上海	5-7 日 深圳
		23-25 日 上海	6-8 日 成都	12-14 日 上海
Xilinx FPGA 的基本 DSP 实现技术	2 天	11-12 日 北京	27-28 日 北京	
使用 PlanAhead 分析与设计工具进行基本设计	2 天		16-17 日 武汉	27-28 日 北京
FPGA 设计基础	1 天	20 日 武汉	15 日 西安	14 日 成都
面向性能的设计	2 天	23-24 日 武汉	29-30 日 北京	5-6 日 上海
Xilinx FPGA 的信号完整性和电路板设计	3 天	23-25 日 北京	22-24 日 成都	19-21 日 北京
高级 FPGA 设计	2 天	26-27 日 深圳	20-21 日 上海	10-11 日 深圳
利用 Virtex-5 FPGA 系列进行设计	1 天	27 日 北京		
设计 LogiCORE PCI Express 系统	2 天	26-27 日 北京	9-10 日 北京	18-19 日上海
利用 VHDL 进行设计		16-17 日 深圳		17-18 日 深圳
嵌入式系统开发	2 天	19-20 日 成都	16-17 日 上海	20-21 日 北京
利用 System Generator 进行 DSP 设计	2 天			6-7 日 西安
利用以太网 MAC 控制器进行设计	2 天	30-31 日 成都	23-24 日 深圳	19-20 日 武汉
利用千兆位级串行 I/O 进行设计	3 天	25-27 日 北京		26-28 日 上海
利用 ChipScope Pro 调试和验证	1 天	31 日 北京	20 日 上海	3 日 深圳
嵌入式系统软件开发	2 天		16-17 日 北京	24-25 日 武汉
嵌入式开放源码 Linux 开发	2 天		13-14 日 武汉	27-28 日 深圳
Xilinx 在线培训课程 （WebEx）	培训课程	7 月	8 月	9 月
在线老师现场授课 (学员于线上学习，老师提供最新的实验项目的现场操作和答疑并进行专业辅导，直接带给学员 FPGA 项目设计的亲身体验。	FPGA 设计基础（免费）	3 日	6 日	7 日
	面向性能的设计	9-10 日	20-21 日	13-14 日
线上授课老师都获 Xilinx 认证，并具有丰富的 FPGA 系统项目经验。	高级 FPGA 设计	16-17 日	23-24 日	27-28 日
现场的课堂教学和实验	PlanAhead 分析与设计	23-24 日	9-10 日	24-25 日
答疑（Q&A)(现场解答学员在学习和实验中遇到的问题)	利用 Spartan-6 系列进行设计	26-27 日	7-8 日	17-18 日

有关报名注意事项：

请联系：北京：电话：010-8275-7632, 传真：010-8275-6745

深圳：电话：0755-86186715，传真：0086-755-86186700,

电子邮件：training@e-elements.com

地址：北京市海淀区北三环西路 32 号恒润国际大厦 801 室

网址：www.e-elements.com

赛灵思 中国销售代表

ALIGNMENT 缘隆有限公司

- 北京 电话：(010) 6266 9572
- 成都 电话：(028) 8509 1261
- 上海 电话：(021) 6439 2771
- 深圳 电话：(0755) 8253 7068
- 南京 电话：(025) 8638 0963

赛灵思 中国 / 香港地区分销商

AVNET 安富利电子元器件部

- 香港 电话：(852) 2176 5388
- 北京 电话：(010) 8206 2488
- 成都 电话：(028) 8652 8262
- 上海 电话：(021) 3367 8330
- 深圳 电话：(0755) 8378 2949

COMTECH 科通 科通数字技术部

- 香港 电话：(852) 2730 1054
- 北京 电话：(010) 5172 6678
- 成都 电话：(028) 8513 1563
- 上海 电话：(021) 5169 6680
- 深圳 电话：(0755) 2698 8221

XILINX®

欢迎各位作出反馈讯息和建议

传真：(852)2429-6772

电邮：xcell-china@xilinx.com

赛灵思 中国 /香港代表处

香港	电话：(852)2424 5200	传真：(852)2494 7159	电邮：ask-china@xilinx.com
上海	电话：(86)21-5131 6060	传真：(86)21-5198 1020	电邮：ask-china@xilinx.com
深圳	电话：(86)755-8660 6588	传真：(86)755-2583 0986	电邮：ask-china@xilinx.com

更多的联络点请查询：www.xilinx.com/cn　　技术支持：www.xilinx.com/cn/support

XILINX®

性能加倍， 功耗减半

2011 WINNER ACE AWARDS EETimes

赛灵思7系列FPGA， 无需妥协的创新！

全新7系列FPGA器件建立在行业唯一的统一架构之上，为您的创意变成现实提供充分灵活的选择！

满足您提高性能、降低功耗的设计需求；利用新一代ISE设计套件为您的开发加速！

创新，用您需要的性能和灵活性，引领世界不断进步！

www.xilinx.com/cn/7

ARTIX 7™ 超低功耗 超低成本

KINTEX 7™ 超值价格 超高性能

VIRTEX 7® 超高系统性能 超大容量

赛灵思公司

香港 电话: (852)2424 5200
传真: (852)2494 7159

上海 电话: (021)5131 6060
传真: (021)5198 1020

深圳 电话: (0755)8660 6588
传真: (0755)2583 0986

ALIGNMENT 中国销售代表 缘隆有限公司

- 北京 电话 ：(010) 6266 9572
- 成都 电话 ：(028) 8509 1261
- 上海 电话 ：(021) 6439 2771
- 深圳 电话 ：(0755) 8253 7068
- 南京 电话 ：(025) 8638 0963

AVNET 安富利电子元件 中国/香港地区代理商 安富利电子元件部

- 香港 电话 ：(852) 2176 5388
- 北京 电话 ：(010) 8206 2488
- 成都 电话 ：(028) 8652 8262
- 上海 电话 ：(021) 3367 8387
- 深圳 电话 ：(0755) 8378 1886

COMTECH 科通 中国/香港地区代理商 科通数字技术部

- 深圳 电话 ： (0755) 2698 8221
- 北京 电话 ： (010) 5172 6678
- 上海 电话 ： (021) 5169 6680
- 武汉 电话 ： (027) 8769 0655
- 成都 电话 ： (028) 8513 1563