FPGA 异构计算
——基于OpenCL的开发方法

黄乐天　范兴山　彭军　蒲宇亮　编著

西安电子科技大学出版社
http://www.xduph.com

FPGA 异构计算

——基于 OpenCL 的开发方法

黄乐天　范兴山　彭　军　蒲宇亮　编著

西安电子科技大学出版社

内 容 简 介

近年来，异构计算得到了业界的普遍关注。作为高性能计算的一种主流解决方案，CPU+GPU的异构计算模式已经得到了产业界和学术界的广泛关注。从 2011 年 Altera 公司发布支持利用 OpenCL 来开发 FPGA 的 SDK 工具以后，采用 CPU+FPGA 构成异构计算系统成为另一种具有竞争力的解决方案。本书主要介绍了 FPGA 异构计算系统的基本架构和开发方法，并以多个不同的案例为读者展示了如何利用几种常用的优化方法来进一步提升系统性能。

本书既可以作为高性能异构计算领域研发者的参考书籍，也可以作为有兴趣掌握这一新技术的电子工程师、软件工程师或在校学生的入门教程。

图书在版编目(CIP)数据

FPGA 异构计算：基于 OpenCL 的开发方法/黄乐天等编著.
—西安：西安电子科技大学出版社，2015.7
ISBN 978-7-5606-3770-9

Ⅰ. ① F… Ⅱ. ① 黄… Ⅲ. ① 可编程序逻辑器件—系统设计
Ⅳ. ① TP332.1

中国版本图书馆 CIP 数据核字(2015)第 155958 号

策　　划　刘小莉
责任编辑　刘小莉　毛红兵
出版发行　西安电子科技大学出版社(西安市太白南路 2 号)
电　　话　(029)88242885　88201467　　邮　　编　710071
网　　址　www.xduph.com　　电子邮箱　xdupfxb001@163.com
经　　销　新华书店
印刷单位　陕西华沐印刷科技有限责任公司
版　　次　2015 年 7 月第 1 版　2015 年 7 月第 1 次印刷
开　　本　787 毫米×1092 毫米　1/16　印张 9.5
字　　数　216 千字
印　　数　1～1000 册
定　　价　28.00 元

ISBN 978 – 7 – 5606 – 3770 – 9 / TP
XDUP 4062001-1

序　言

很高兴为黄乐天老师的书《FPGA 异构计算——基于 OpenCL 的开发方法》写序言。

OpenCL 是一个为异构平台编写程序的框架，此异构平台可由CPU、GPU或其他类型的处理器组成。OpenCL 由一门用于编写 kernels(在 OpenCL 设备上运行的函数)的语言(基于C99)和一组用于定义并控制平台的 API 组成。OpenCL 提供了基于任务分割和数据分割的并行计算机制。

OpenCL 类似于另外两个开放的工业标准OpenGL和OpenAL，这两个标准分别用于三维图形和计算机音频方面。OpenCL 扩展了 GPU 用于图形生成之外的能力。OpenCL 由非盈利性技术组织Khronos Group掌管。

OpenCL 最初由苹果公司开发，且拥有其商标权，并在与AMD、IBM、英特尔、Altera和nVIDIA技术团队的合作之下初步完善。随后，苹果将这一草案提交至Khronos Group。

OpenCL 是一项新的技术，其目的是拓展硬件支持范围，并利用软件实现跨平台计算。对于无 FPGA 开发经验的软件工程师来讲，使用 OpenCL 来开发基于 FPGA 的高能效异构计算系统是一种非常便捷的方法。

黄老师的这本书是基于 Altera 较早期的 DE4 开发板做的研究，这是 FPGA 业界第一本介绍在 FPGA 上实现运行 OpenCL 的书籍。

该书详细地阐述了最基本的 OpenCL 学习流程。对于刚刚接触 OpenCL 的老师和学生来讲，本书无疑会帮助大家缩短实现异构计算的时间。

OpenCL 是今后大数据计算、深度数据解析以及医疗、制药、高清视频的理想平台。它也是本行业发展的趋势所在。

我推荐这本书，同时也希望黄老师能再接再厉，不断地写出更优秀的作品。

陈卫中
Altera 公司大学计划中国区经理
2014 年 12 月于成都

自序——大变局

接触到 OpenCL 是在三年前，当时参加 Altera 的全国教师大会，知道 Altera 准备推动这样一种新的 FPGA 开发方法。当时的感觉是：不关我们的事。觉得用这样一种新的语言来开发 FPGA 的方法在成熟之前，我们应该不会去使用它。另外更多的还有疑虑。C to Silicon 的口号喊了很多年，从 20 世纪 90 年代初就开始喧嚣，前前后后好几轮，每一轮一开始都轰轰烈烈，但是到后面都无疾而终。而这一次基于 OpenCL 的 FPGA 开发方法是否能带来不一样的结果呢？说实话，我心里是存在着很大的疑虑。我相信这也是现在大多数听到用 OpenCL 来开发 FPGA 的工程师朋友们或多或少存在的疑虑。

相信第一次听到使用 OpenCL 来开发 FPGA 的朋友们心中一定会有这样的疑问：OpenCL 相对于我们传统的 HDL 优势在哪里？使用 OpenCL 来开发 FPGA 会不会遇到什么瓶颈和问题？我们也是带着这样的疑问参加了 2012 年底在厦门召开的 Altera 的教师大会。在会上我听到了负责 Altera 公司 OpenCL 技术支持的工程师陈涪师兄对 FPGA 异构计算的详细介绍，并通过会下的交流才终于认识到——这一次不同了！不同之处在于，这并不是简单的 FPGA 开发方法的变革，而是 FPGA 应用领域和使用对象的变革。FPGA 的应用将转向一个新的领域——高性能异构计算和海量数据处理，而 FPGA 的使用对象也不是传统的硬件工程师，很有可能是以开发软件为本职工作的程序员。换言之，从目前来看，OpenCL 和 HDL 之间其实并不存在可以比较的先决条件，因为这是针对不同的应用领域和不同的开发目的而设计出的不同开发方法。因此，本书并没用采用“基于 OpenCL 的 FPGA 设计”之类的书名，而是选择了“FPGA 异构计算”这一更符合目前 FPGA 设计方法变革的题目。当然，由于这套异构计算系统是依靠 OpenCL 来开发的，所以在后面又加了一个副标题而最终定名为“FPGA 异构计算——基于 OpenCL 的开发方法”。

那么 FPGA 异构计算到底能解决什么问题？主要面向哪些行业或者领域呢？我认为，在现阶段以及未来很长一段时间内，FPGA 异构计算主要解决的都是高性能、高能效大数据处理这一主要问题。说具体一点，其所面对的具体领域就是：金融计算、期货定价、石油勘探、海量图像检索等。这些领域通常被各类高性能计算系统所盘踞，并随着高性能计算系统的发展而不断向前发展。随着高性能计算系统的发展，多核和异构计算系统正在发生转变。目前大量采用 CPU+GPU 异构架构的高性能计算系统已经被纷纷研发出来。但正如现在媒体报道的，这类超级计算机的能耗是大的惊人的。如何用更低的功耗提供更强大的计算能力就成为高性能计算向前发展必须要解决的问题。其实我们从“比特币矿机”的发展历程就可以知道，采用 FPGA 来加速运算会获得比 GPU 更高的能效比。因此，采用 FPGA 替代 GPU 作为未来高性能计算的加速器，应该是现阶段的 FPGA 异构计算发展的主旋律。换言之，目前需要关注的是 FPGA 如何替代 GPU 的话题，而不是 OpenCL 如何替代 HDL 的话题。这是两种异构计算系统的竞争，而不是两种开发语言的竞争。

最后说一下本书所面向的读者，通过前面的介绍大家应该明白，本书面向的读者不是使用现有 FPGA 开发方法的硬件工程师，也不是要教育大家去放弃现在的 FPGA 开发方法

而转移到另外一种方法上。本书的主要目的是为读者介绍一种新的异构计算系统及其编程方法，是为那些高性能计算系统及软件的开发者介绍如何用 FPGA 构建一种更高能效的异构计算系统的。但遗憾的是，由于国内高性能计算方面研究的滞后，尤其是高性能计算应用研究的滞后，可能导致最需要这本书的读者不一定有机会接触到这本书。我国传统的文理分科、专业划分等问题造成了学科上的割裂。传统上的学经济、金融、管理学科的人大多数文科出身，对数学工具的使用不够熟练，更别提对高性能计算方法和技术手段的掌握了。反观欧美发达国家，计算机技术已经是任何一个学科的基础技术，包括音乐、艺术在内的我们认为完全依靠个人灵感的学科，欧美国家的研究者都在尝试着对其建模并用计算方法加以分析，更别说像金融、经济等这类本身就具备极强规律性的学科了，从 2006 年起就已经看到欧美学者将 FPGA 用于金融计算的报道。在 2014 年的电子设计自动化领域顶级会议之一的“Design, Automation & Test in Europe Conference”(DATE)上，已经有采用 FPGA 异构计算的方法来实现期权定价的论文发表。由于我们对于金融、期货是外行，在请教了相关方面的专家以后，他们表示该论文使用的算法是金融界相当“简单”和“原始”的算法。而当我们提出是否有兴趣进一步合作开展相关研究时，他们却基本都顾左右而言它。在国内想要做跨学科研究之难，由此可见一斑。不过值得欣喜的是，国内已经有以华西证券为代表的一系列具有前瞻眼光的行业公司开始认识到 FPGA 异构计算系统的优势，也有像百度、腾讯这样的互联网大公司在尝试使用这一方法。希望本书能够起到抛砖引玉的作用，为推广 FPGA 异构计算技术在国内的发展做出一点小小的贡献。

由于本书中的设计案例是基于较早期版本的 Altera OpenCL SDK for OpenCL 和较旧的 DE4 开发平台完成的，所以随着软硬件版本的升级，一些具体的开发流程和操作指令可能发生了变化。由此给各位读者造成了困扰和不便，敬请谅解。

在此特别感谢 Altera 公司大学计划陈卫中先生以及师研女士的大力支持，特别感谢 Altera 公司陈涪师兄的技术指导，特别感谢在本书编辑出版过程中提供大力帮助的西安电子科技大学出版社各位领导和编辑，特别感谢参与本书编辑和校对的褚廷斌、李嵩、董芸、王君实、贺江等同学。

编者著

2015 年 4 月于成都

目　录

第一章 FPGA 异构计算

本章引入了 OpenCL 的开发方法。不论你以前是不是 FPGA 的开发者，本章都值得阅读。本章首先介绍了异构计算的发展历程，引入 FPGA 异构平台，其次阐述了 FPGA 的发展史及其内部结构原理和演变历史。通过阅读本章，了解到使用 OpenCL 开发 FPGA 并非简单的开发语言的变化，更多的是 FPGA 应用领域与开发思想的转变。

1.1 异构计算的发展

异构计算是当今半导体行业的一个热门词汇，从芯片厂商到普通消费者，都对其表现出了极大的兴趣，应用范围也越来越广泛。异构计算(Heterogeneous Computing)，又译异质运算，主要是指使用不同类型指令集和体系架构的计算单元组成系统的计算方式。但事实上，异构计算却并不是什么新概念，而是由来已久。早在 20 世纪 80 年代中期，异构计算技术就诞生了。它主要是指使用不同类型指令集、体系架构的计算单元组成混合系统的一种特殊计算方式。常见的计算单元类别包括：CPU(中央处理器)、GPU(图形处理器)、协处理器、DSP(信号处理器)、ASIC(专用集成电路)、FPGA(现场可编程门阵列)等。

为什么需要异构计算？原因很简单：我们需要越来越强大、越来越高效的计算系统。在过去，随着半导体技术的进步和频率的提升，绝大多数计算机应用不需要结构性的变化，或者特定的硬件加速，即可不断提升性能，但是现代应用经常会碰到内存、功耗方面的限制。此时，引入特定单元让计算系统变成混合结构就成了必然，每一种不同类型的计算单元都可以去执行自己最擅长的任务。

对于普通读者来说，最熟悉的异构计算平台架构莫过于 CPU + GPU 架构。这是因为此种架构在 PC 机中是常见的组合，也得益于近年来以 NVIDIA、ATI(后合并于 AMD)为代表的 GPU 厂商大张旗鼓地宣传使用 GPU 可以极大地加速通用计算。为达到这一目的，各大 GPU 厂商分别对原本主要面向图形图像处理的 GPU 架构加以改造，推出了适用于通用计算目的的 GPU，也就是所谓的 GPGPU(General Purpose GPU，可翻译为通用计算图形处理器)。

在数年之前，GPU 还只能处理图形渲染一件工作，而通用计算的大门打开之后，GPU 被导入高并行计算领域，并成为超级计算机的新核心。这体现了一种思想上的跨越，并在超级计算领域首先得以实现。再往后，AMD 的 APU 将 CPU 与 GPU 融为一体，由此拉开了异构计算的大门。CPU 与 GPU 的高度融合已是大势所趋，但这不只是硬件层面的变更，更多的是计算理念的变革。如何将不同的计算任务自动分配给最适宜于处理该任务的芯片，借此实现最高的能效比以及最高的晶体管利用率，成为探索新的编程模式或者计算模式要面临的重大问题。

AMD 公司是推动异构计算的先行者。2006 年 7 月底，AMD 宣布以 54 亿美元的高价

收购 ATI，完成了 CPU 行业、GPU 行业有史以来最重量级的一次整合。自收购达成起，AMD 也成为全球唯一一家同时拥有高性能 CPU 处理器、GPU 显卡、芯片组主板的半导体企业。在宣布收购的第二天，AMD 便迫不及待地宣布要把高性能 CPU、GPU 做到同一颗芯片上，并在三个月后将这一项目命名为“Fusion”(融聚)，又过了两个月再次提出了全新的“APU”(加速处理器)概念。

从最初宣布到最终发布，APU 花了足足四年半的光阴，其间经历了 GPU 加速计算的崛起、GPU/GPU 地位之争，直到 2010 年初才开花结果。APU 硬件为实现异构计算奠定了基础，但异构计算的真正关键在于标准化，这样才能解决程序开发的问题。针对 PC 机，异构计算平台必须具有三个基本要素：拥有一套业界共同遵循的规范，拥有一套标准化的应用程序接口(API)，拥有相应的开发语言和开发工具、编译器。

OpenCL 的诞生为异构计算奠定了坚实的基础。OpenCL 的全称为“Open Computing Language”(开放运算语言)，它是第一个面向异构计算环境的开放式、免费标准，也是一个统一的编程环境。借助 OpenCL，软件开发人员能够为高性能计算服务器、桌面计算系统、手持设备编写高效轻便的代码，这些代码可支持诸如多核心处理器(CPU)、图形处理器(GPU)、Cell 类型架构以及数字信号处理器(DSP)等各种不同的异构环境，在游戏、娱乐、科研、医疗等各种领域都有着广阔的发展前景。

苹果公司是 OpenCL 规范的发起者，在 2008 年 6 月的 WWDC 大会上，苹果就提出了这套规范，旨在建立一个通用的开放 API，并在此基础上开发 GPU 通用计算软件。随后，Khronos Group 宣布成立 GPU 通用计算标准工作组，以苹果的提案为基础，创立 OpenCL 行业规范。仅仅半年后，OpenCL 1.0 标准就正式发布。2010 年 6 月 14 日，OpenCL 1.1 发布。2011 年 11 月 15 日，OpenCL 1.2 发布。

OpenCL 工作组的成员包括：3Dlabs、AMD、苹果、ARM、Codeplay、爱立信、飞思卡尔、GraphicRemedy、IBM、Imagination Technologies、英特尔、诺基亚、NVIDIA、摩托罗拉、QNX、高通、三星、Seaweed、德州仪器和瑞典 Ume 大学等。作为半导体业界三巨头的英特尔、NVIDIA 和 AMD 都是这个组织的核心成员，但是微软公司并不在此列，微软希望打造自己的通用计算标准，这就是 C++ AMP。

OpenCL 在诞生之后就获得相当广泛的支持，也被认为是异构计算的基础。这套标准的诞生，对于 AMD 和英特尔意义重大。AMD 虽然拥有顶级的图形技术，但一直无法单独提供完整的开发环境，软件开发者如果要利用 AMD 的 GPU 进行通用计算，就不得不面临为硬件直接编程的窘境，这也导致了 AMD 在 GPU 通用计算市场的份额低得可怜。在 OpenCL 诞生之后，业界有公开的标准可遵循，软件开发可以脱离硬件进行，事实上也让 AMD 的异构计算平台脱离了羁绊。类似的事情也发生在英特尔的身上，但英特尔的图形技术较为初级，过去从未涉及通用计算领域，但 OpenCL 的诞生还是能够让英特尔的计算平台获得一些好处。

在 OpenCL 之前，NVIDIA 其实就拥有自己的异构计算平台，也就是著名的 CUDA。CUDA 以 NVIDIA 的 GPU 为基础，提供了一整套的开发环境，同时也为通用计算应用推出专门的 Tesla 超级计算产品。在 OpenCL 出现之后，CUDA 的动向也为业界所关注，但 NVIDIA 自身也作为 OpenCL 的核心成员，该标准的发布便有 CUDA 的巨大贡献。凡是支

持CUDA的GeForce GPU，也能够同时支持OpenCL，这意味着NVIDIA的GPU产品可以同步跨越两种平台，这对于NVIDIA而言也并不是坏事。

当传统的GPU以及CPU厂商为OpenCL标准而欢欣鼓舞的时候，FPGA厂商也嗅到了OpenCL所带来的新机遇。2011年11月，Altera“悄悄地”发布了Altera面向FPGA的OpenCL计划，内容仅仅是一个网页和一份8页的白皮书，没有可以实现的工程。在OpenCL计划推出之初，本书的作者和大多数读者朋友一样并未对此给予过多的关注。因为在FPGA的发展历史上曾经有过许多次高层次综合(High Level Synthesis)方面的尝试，但无一不是“无疾而终”。但这一次，OpenCL确实给我们带来了不一样的东西。

传统的FPGA设计极为复杂，不仅设计思路同在CPU、GPU上设计软件完全不同，还需要考虑如约束等非常细节的物理底层问题。因为传统FPGA设计不仅门槛很高，而且设计周期也远长于纯软件设计。如何让FPGA比较简便地实现通用计算的加速而无需通用计算程序设计者精通底层问题的细节，是FPGA厂商一直以来关注的问题，也是FPGA公司的合作商一直以来努力的方向。早在2008年左右，Xtreme数据公司就开发了置入式模块可以替代母板上的AMD Opteron处理器，而且不需要改变电路板设计。用户可以根据多处理器设计需要，合理地结合使用Opteron处理器和FPGA协处理器。几乎同一时期，SRC公司与Altera开发了MAP系列协处理器产品。这些模块通过存储器总线接口与AMD或者Intel处理器连接，数据带宽高达14 GB/s。SRC还提供Carte工具链，转换C语言或者FORTRAN程序，可在FPGA协处理器上更快地运行。两大FPGA公司各自开发了可以直接将C语言转换为硬件逻辑的工具，如Altera公司的C2H和Xilinx公司的AutoESL等。但这些工具或者方法过于“执着”地和硬件描述语言比拼效率，因而忽略了从编程的易用性上对编程模型加以改进，从而很难真正得到那些不懂硬件细节的程序员们的青睐。

OpenCL的出现及其在产业界的迅速推广，为FPGA在高性能计算领域找到了一条“康庄大道”。OpenCL语言对并行运算的天然适应性使得困扰传统高层综合研究中并行化的问题得到了完美的解决。如果能把OpenCL标准引入到FPGA中，开发人员利用OpenCL就能够自然地描述在FPGA中实现的并行算法，其抽象级要比VHDL或者Verilog等硬件描述语言(HDL)高得多。FPGA公司也从利用高层语言替代硬件描述语言“自我革命”的困境中转向了和GPU等计算单元在同一开发环境下对等的竞争，从而使得竞争从开发方法的竞争转向了更加宏大的体系结构的竞争。FPGA的厂商完全可以沿着GPU厂商已经走过的道路一路追赶过去，并凭借自己的特色在追赶的过程中不断地开疆拓土。希望我们能和各位读者一起，见证并参与到这一竞争中去。

1.2 FPGA发展简介

1.2.1 FPGA发展史

当1985年全球首款FPGA产品——XC2064诞生时，FPGA还只是整个电子系统的一个配角。最初FPGA只是用于胶合逻辑(Glue Logic)，用来连接从胶合逻辑到算法逻辑再到数字信号处理、高速串行收发器和嵌入式处理器，FPGA真正地从配角变成了主角。近30

年的 FPGA 的发展史，就是一部电子行业设计工具与设计方法不断变迁的历史。每当 10 年过去了以后，FPGA 都以全新的面目出现在工程师们的面前。

在 FPGA 出现的时候，还有多种不同结构的可编程逻辑(PLD)被工业界采用，包括 PAL(Programmable Array Logic，可编程逻辑阵列)、EPLD(Erasable Programmable Logic Device 可擦除可编程逻辑器件)以及在此基础上发展起来的 CPLD(Complex Programmable Logic Device，复杂可编程逻辑器件)等。CPLD 与 FPGA 在当时代表了可编程器件的两种不同的发展思路。CPLD 依靠复杂的逻辑块来实现高密度的组合逻辑，而 FPGA 则依靠大量而小型化的查找表(LUT，Look Up Table)配合触发器来实现组合和时序逻辑功能。

在 20 世纪 80 年代中期，可编程器件从任何意义上来讲都不是当时的主流，虽然其并不是一个新的概念。可编程逻辑阵列(PAL)在 1970 年左右就出现了，但是一直被认为速度慢，难以使用。1980 年之后，可配置可编程逻辑阵列(PAL)开始出现，可以使用原始的软件工具提供有限的触发器和查找表实现能力。这使得其可以通过编译码、地址转换、状态机等简单的逻辑功能实现多个通用或专用芯片之间的“胶合”。可编程逻辑器件被视为小规模/中等规模集成胶合逻辑的替代选择，并被逐步接受，但是当时可编程能力对于大多数人来说仍然是陌生和具有风险的。1980 年底直至 1990 年初，工程师对可编程逻辑器件的需求一直都是提高逻辑密度。随着逻辑门数量的增加，应用开发人员开始考虑使用 CPLD。它具有优异的扇入和确定性，是实现解码器和状态机等需要大量逻辑资源的数字逻辑模块的最佳选择。对于需要大量寄存器的功能，设计人员之所以选择 FPGA，是因为它在触发器数量上有很大的优势。

随着器件复杂程度的增加，人们越来越难以直接理解器件，但设计工具越来越智能，因此，对手动编辑的需求降低了。在 20 世纪 80 年代末期，出现了一种全新的设计风格，它基于硬件描述语言(HDL)实现自动逻辑综合。HDL 采用名为寄存器传送级(RTL，Register Transfer Level)逻辑的新形式来描述硬件功能。使用 Verilog-HDL 语言(及其 ADA 衍生竞争对手)无意中得到的结果是，综合产生了有很多寄存器的同步设计。这些设计直观上更适合在有大量寄存器的 FPGA 体系结构中实现，而不是在组合逻辑 CPLD 中实现。由于 FPGA 结构的限制，早期尝试面向 FPGA 进行逻辑综合失败了。但设计人员仍然开始思考采用 Verilog 等文本语言进行逻辑设计，开始考虑 FPGA 不仅仅是接口单元，而是实现其设计完整功能模块的方法。

20 世纪 90 年代，信息技术在经历了多年的独立发展之后在通信领域找到了重要的交汇点。光纤技术、计算机技术、数字信号处理技术、集成电路设计技术的成熟都为通信网络迅猛发展提供了关键的支撑。“信息高速公路”这一今天听起来有些陌生甚至不伦不类的名词在当年作为“高科技”的代名词被人们反复提起。无论是美国还是中国，各大运营商都致力于不断拓宽网络带宽从而提升传输速率。这一时期通信技术不断演进，通信标准不断更新，与之相适应的就是通信设备需要不断的升级。可编程逻辑器件开发周期相对较短且容易升级的特点使得其在通信与网络设备上得到了大量的应用。

随着芯片逻辑容量的增长，PLD 的应用范围也出现了变化。当 CPLD 还受限于几千个等价逻辑门时，设计人员采用芯片主要是实现中央微处理器或者微控制器(MCU)芯片外围的简单功能。较小的器件用作地址解码器、中断控制器或者总线扩展器，较大的器件可以是以快速状态机实现处理某些实时操作——当软件完成这些操作显得太慢的时候。当可编

程逻辑器件的规模达到 10K～20K 逻辑门时，设计人员能在 PLD 上开发基于小型 CPU 的子系统，例如，总线接口和可编程串行接口控制器。与 ASIC(Application Specific Integrated Circuit，专用集成电路)或 ASSP(Application Specific Standard Parts，专用标准产品)相比，这类设计要慢一些，能效也低。但使用 PLD 的设计却是完全用户可定义的，并支持可重新编程。这对于需要不断升级更新从而支持新标准和新协议的通信网络设备而言是非常重要的优势。大部分这类子系统设计都将使用大量的寄存器，很多都需要大量的内部 SRAM。而这些无疑都更适合采用 FPGA 这种架构而非 CPLD。

在当时的市场上，Xilinx 一直坚持着其细粒度、岛状布局的 FPGA 架构，而 Altera 一直在坚守自己发明的 CPLD。当发现上述趋势以后，Altera 设计人员总结出市场需要一类新产品。与 FPGA 相似，它应该采用精细粒度逻辑单元，每一个单元都含有 LUT 和寄存器，但是像 CPLD 一样，它应该使用层次化的确定性互联，以保持时序的可预测性。与此同时，Altera 借鉴了某类逻辑门阵列在体系结构中嵌入了 SRAM 模块，在以后的发展历程中将有越来越多不同规格的 SRAM 模块被嵌入到 FPGA 中。这种嵌入式的 SRAM 模块可以用作缓冲或者高速暂存存储器、寄存器文件，或者实现复杂函数发生器的查找表。1995 年，Altera 推出了第一款类似 FPGA 的系列产品——FLEX 系列器件。

FLEX 系列器件最终达到了 250K 的等效逻辑门容量，内嵌的 SRAM 高达 40 K，这在当时看来是一个“惊人”的数据。这意味着既可以利用它实现某些非常复杂的状态机和算术功能而不再只是传统的接口和胶合逻辑应用，工程师们可以利用它实现整个子系统，例如以太网接口中的介质访问控制器，又可以利用它实现各类信号处理加速器，如纯逻辑实现的有限冲击响应(FIR，Finite Impulse Response)滤波器或快速傅里叶变换(FFT，Fast Fourier Transformation)算法等。虽然 Altera 最初的目的是彰显 FLEX 器件与 FPGA 的不同，但 FPGA 却成为大规模 PLD 的总称，Altera 也就逐渐采用了这一术语。但直至今日，大家从 Altera 公布的数据手册上面依然可以看到 FLEX 系列器件对后续产品的影响，从而导致了这两家 FPGA 公司在最终产品的架构上实际存在着明显的区别。

实现子系统一般需要多个时钟，每一时钟与自己的锁相环(PLL)同步，但是分立 PLL 价格昂贵，占用较大的电路板空间。从 1996 年开始，Altera 推出了具有内部可编程 PLL 的 FLEX 器件。这是一个重大的转变，它意味着开始集成常用的非 PLD 硬件模块以改进系统总体设计。从这时起，FPGA 就已经不再是单纯的可编程逻辑器件而是向一种系统级通用器件转变。与此同时，IP(Intellectual Property，知识产权)核复用的方法在 IC 设计领域开始流行，而可编程逻辑行业也在第一时间跟进了这种设计方法。Altera 开始开发 IP 库，应用了设计工具，增加了新特性以帮助实现 IP 重用。新特性包括能够通过简单的用户界面设置可重用 IP 模块的参数，第三方供应商能够以加密形式提供 IP 模块。采用这种设计方法可以避免设计团队从头设计一个新的产品，当器件规模超过 100K 等效逻辑门时，这种设计方法和设计思想的改进显得尤为必要。

容量的增加也带来了验证和调试的问题。与 ASIC 设计不同，由于可编程逻辑具备可以重编程的特性，在早期，可编程逻辑的开发者并不十分重视对设计进行早期的验证。当等效逻辑门超过 100K 时，依靠设计人员通过采用逻辑分析仪来观察外部引脚，从而发现问题的概率已几乎为零。FPGA 的设计人员开始利用类似 ASIC 设计的方法仿真相关部分，在针对 FPGA 将其综合到网表中之前，采用 RTL 仿真器测试他们的寄存器传送级(RTL)代码。但完

成这一转变却并不容易，在很长一段时间内，FPGA 的设计人员(尤其是亚洲地区，以喜好动漫的日本人为代表)更加习惯于采用绘制图形化波形文件方法来输入测试激励。Altera 将 Quartus II 自带的支持这一功能的仿真器一直保留到了其 9.1 版本发布(在 10.0 版本发布后彻底取消了这一功能)，但是在仿真中可行的 RTL 在实际中可能还无法工作。如何在实际的工作环境中捕捉 FPGA 内部运行状态就成了一个新的问题。1999 年，Altera 推出了 SignalTap 逻辑分析器：这一硬件特性支持用户在系统运行时监视 FLEX 器件中的每一个寄存器。

当多种标准 IP-Core 都逐渐被编写出来后，一个越发明显的需求开始出现——处理器是否也可以用 IP-Core 的形式实现。大约在 2000 年，Altera 推出了 Nios，这一微处理器内核是专门针对 FPGA 架构加以优化的。在当时，Nios 可以达到 50 MHz 的运行速度，已经超过了一般的单片机。原本处理器与 FPGA 的融合看起来应该是水到渠成的事情，但二者融合之路却显得如此艰难。在 2001 年，Altera 开始尝试在 APEX 系列器件中嵌入以 200 MHz 速度运行的 ARM922 CPU 内核。这是一次伟大的尝试，但也是一次无果的尝试。此后，在 FPGA 中嵌入处理器硬核几经反复，Altera 一直坚持使用 Nios 软核的方案，并在很多场合得到了有效的应用。而 Xilinx 在 PowerPC 硬核和 MicroBaze 软核之间徘徊。到了 2011 年，一种以 ARM A9 处理器硬核为中心的新型嵌入方式才被两大 FPGA 公司认可为主流方向，并成为两大公司主推的方案。时至今日，处理器与 FPGA 的融合仍然在继续。

高速接口的物理层(PHY)是下一个被集成到 FPGA 中的硬核单元。2001 年，Altera 发布 Mercury 系列 FPGA，它具有内置为硬核 IP 的 1.25 Gbps 收发器。模块包括 1.25 GHz 模拟驱动器、接收器、混合信号时钟数据恢复电路等，该模块可以从接收波形中重建最初的数据。集成的硬核单元可以让应用工程师不必再考虑使用外置的驱动芯片，而为了应对越来越高速的信号处理需求，Mercury 器件开始集成专用的 8 × 8 乘法器子模块。一年后，Altera 推出了 Stratix 器件，转向了嵌入式全数字信号处理(DSP)构建模块。时至今日，高速收发器和嵌入式数字信号处理模块已经成为 FPGA 的标配。

当越来越多的专用硬核被集成到 FPGA 中以后，FPGA 的设计方法需要发生根本性的变化。在 IC 设计领域已经逐步得到认同的 SoC 设计方法同样被引入到 FPGA 设计领域，这一方法的核心在于围绕 CPU 内核展开设计，以 CPU 引出的系统总线为主干，其他模块都挂在这一总线上。Altera 再次积极响应，创新实现了 SOPC Builder。这一工具具有交互式的导航用户界面，用于在 FPGA 上开发基于 CPU 的系统。用户标明把哪些模块装配到哪里，由工具生成所需要的 RTL。这一设计方法从 Altera 推出 Nios 之后开始萌芽，时至今日，SOPC Builder 已经发展到了下一代设计环境 Qsys。这一方法的出现标志着 FPGA 开始从外围芯片转变为系统芯片。以 CPU 为中心的 SoC 带来了新需求，通常会有标准外部总线，例如 PCI 或者 USB，在片内外设控制器和外部器件之间还会有很多串行或者并行连接。这种多样性意味着更多的引脚、I/O 上更多的信号和更大的电压变化以及更多的时钟域，而这些变化反映在越来越复杂的 FPGA I/O 单元和时钟网络上。

半导体工艺一直在不断改进，晶体管密度也越来越高。相应的 CPU 生产商关注的重点从提高时钟频率转向在管芯上集成两个、四个甚至更多的 CPU 内核，这就是多核体系结构。多核思路体现在 FPGA 使用上有两个明显的方向。一个思路就是简单地复制 CPU 内核。这相对比较容易将多个处理器内核编译到 FPGA 中，但是将其连接起来就不那么容易了。可编程逻辑提供了丰富的资源，设计人员几乎可以实现从阵列到紧耦合内核直至共享 L2 高速

缓存的所有设计。利用 SoPC Builder 可以设计实现基于 Avalon 总线上的多 CPU 系统。多核的另一个思路则采用了不同的方法：异构系统，实现一个 CPU 内核例化的同一总线、IP 和工具支持，同时实现 CPU 内核和多个对等的外部加速器，从而导致完全不同的多核设计思路——以软件为中心的方法。

设计同构多核系统非常直观也极其简单，因而成为早期多核计算发展的主流技术。设计者需要确定合理的多核规模，多出一个或者两个也有可能降低效率。根据期望的线程之间共享的存储器等级，选择合理的互联体系结构。在 CPU 之间划分软件线程、仿真系统，并重复直至符合规范要求，这一过程一直是以硬件为中心的。首先选择合理的硬件架构，再编写合理的软件适配到硬件平台中。而以软件为中心的设计方法则要复杂得多，需要首先分析代码并找到热点，对于计算开销很大的代码部分设计定制加速器。这一方法从已有的 CPU 内核上可以工作的软件开始，经过多次迭代最终产生定制满足实际系统软件要求的多个硬件加速器。

但如何设计这些硬件加速器并使之与处理器配合起来很好地完成工作并不是一件简单的事情。传统的设计方法是采用 HDL 开发出等价的加速器后再挂在系统总线上，CPU 通过访问加速器的接口使用硬件加速器的功能，通常需要逻辑设计工程师和软件开发工程师相互配合才能开发这种复杂的流程，这对于在通用平台上开发 C 代码的软件公司来说是很难完成的。C 语言虽然能够很好地定义顺序执行的程序，但是无法表达熟练的编程人员使用的并行处理方法。这一难题随着 Altera 推出的一组工具——支持编程人员采用越来越流行的 OpenCL 编写并行算法而得到了解答。

当 OpenCL、异构计算、SoC-FPGA 这些新兴的技术手段渐行渐近时，FPGA 作为一种通用系统芯片的地位越发凸显。可以预见，基于异构计算的思想，利用 OpenCL 开发 SoC-FPGA 将成为一种很有竞争力的开发方法，在这一刻，多年来在可编程逻辑领域甚至整个数字系统设计领域取得的进展得到了统一。

万法归一，一归何处？

1.2.2 FPGA结构的演变

FPGA 的结构是随着 FPGA 的发展不断变化的。从单一的可编程逻辑器件到今天的系统级芯片，FPGA 的内部资源从数量和种类上都有了很大的改变。我们可以从分析 Cyclone V 数据手册入手，来回顾一下 FPGA 结构的演变历程。

图 1-1 给出了 Cyclone V FPGA 的互联结构。在 FPGA 中的资源包括 LAB(Logic Array Block)、存储模块(Memory Blocks)、数字信号处理模块(Digital Signal Processing Blocks)、输入输出模块(I/O Element)等。这些资源通过本地直连线(Direct-Link)、行互联线(Row Interconnects)以及列互联线(Column Interconnects)互联在一起。从图上的布局结构可以看出当年 FLEX 系列器件布局的影子——阵列形式排列的逻辑单元。

可编程逻辑单元是 FPGA 中最基本的结构。在 Cyclone V 系列的 FPGA 中采用的 ALM 如图 1-2 所示。这原本是在 Stratix 等高端 FPGA 中才采用的结构，在 Cyclone V 中也采用了这种结构。相较于传统的可编程逻辑单元，ALM 的结构已经大为丰富。除了传统的 LUT 结构之外，ALM 中还包含有硬件加法器，可以更方便地实现各类数学运算电路。而自适应

的 LUT 分拆和组合结构也可以更加灵活地构建出不同的逻辑电路。ALM 内部包含丰富的互联线，可以支持 LUT、加法器、寄存器之间不同的级联和组合。

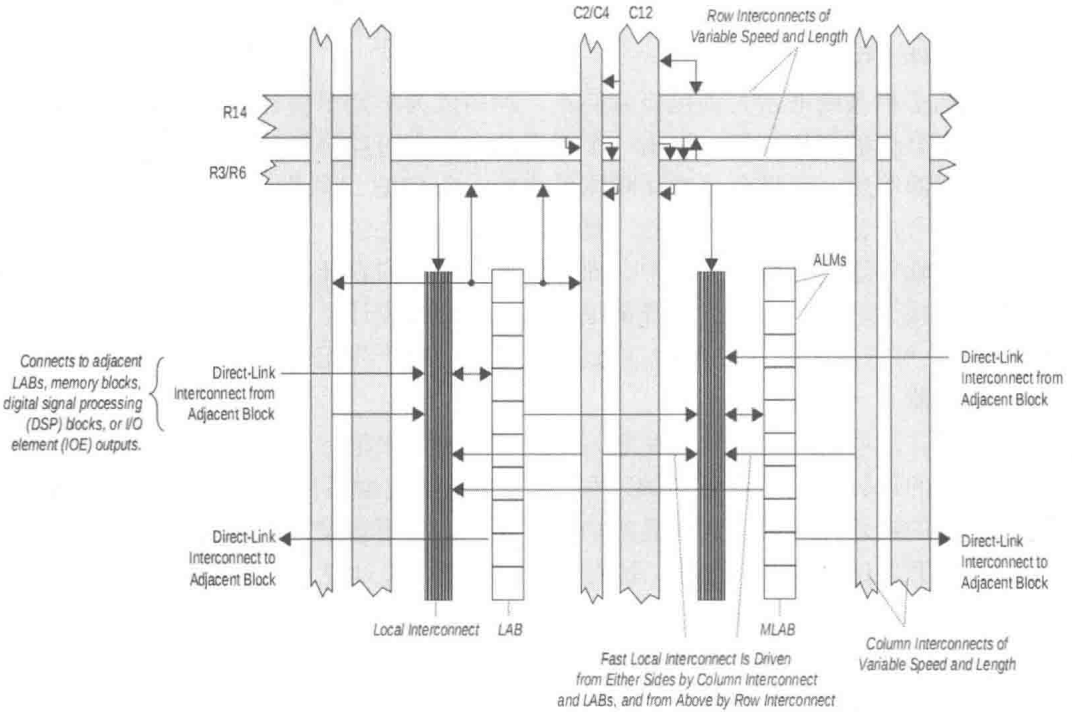

图 1-1　Cyclone V FPGA 系统互联结构

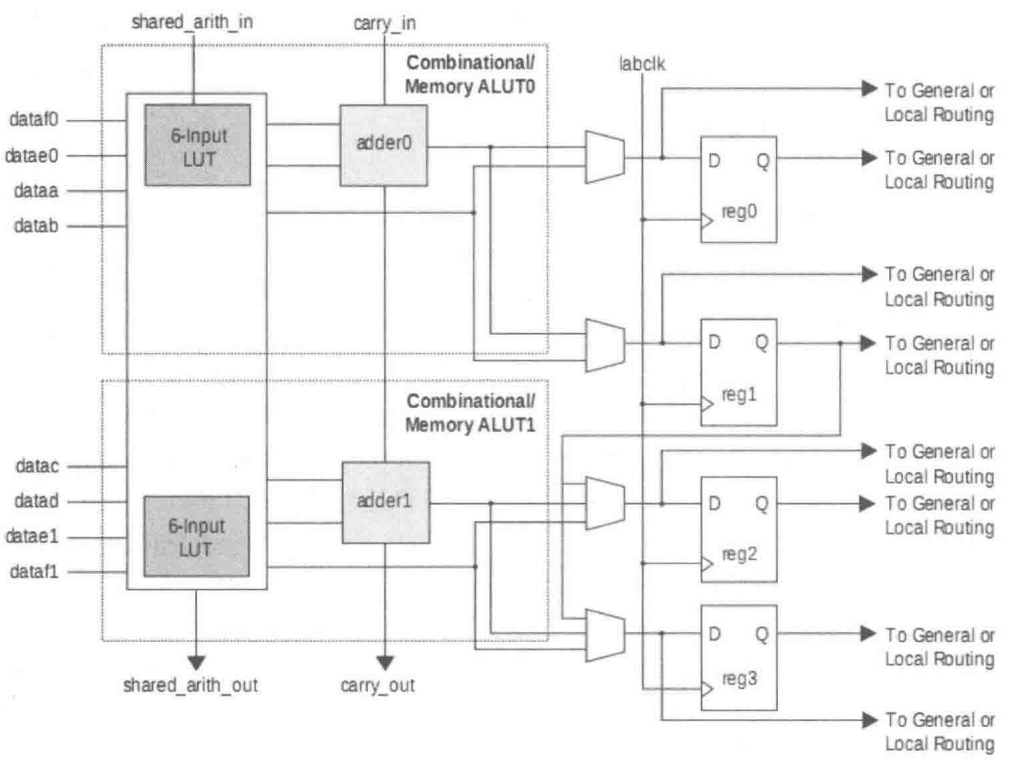

图 1-2　自适应逻辑模块(Adaptive Logic Modules，ALM)结构