HZ BOOKS
华章科技
PEARSON

OpenCL核心设计人员亲自执笔，全面而深刻地解读了OpenCL规范和编程模型，公认的OpenCL权威著作！

注重实践，通过大量案例和代码演示了基于OpenCL编写并行程序和实现各种并行算法的原理、方法、流程及最佳实践！

OpenCL Programming Guide

OpenCL编程指南

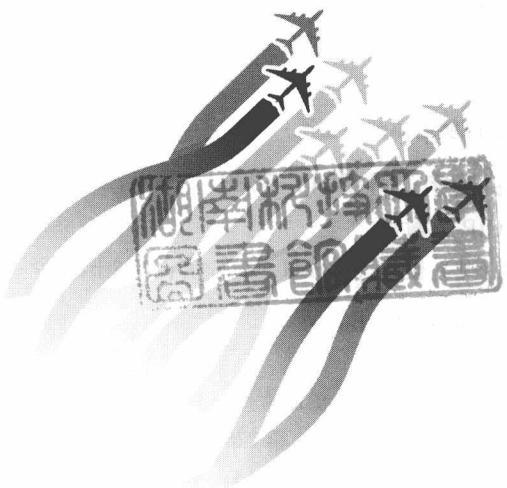
（美）Aaftab Munshi　Benedict R. Gaster　Timothy G. Mattson　James Fung　Dan Ginsburg　著
苏金国　李璜　杨健康　等译

机械工业出版社
China Machine Press

OpenCL Programming Guide

OpenCL编程指南

（美）Aaftab Munshi　Benedict R. Gaster　Timothy G. Mattson　James Fung　Dan Ginsburg　著
苏金国　李璜　杨健康　等译



机械工业出版社
China Machine Press

OpenCL 领域公认的权威著作，由 OpenCL 核心设计人员亲自执笔，不仅全面而深刻地解读了 OpenCL 规范和编程模型，而且通过大量案例和代码演示了基于 OpenCL 编写并行程序和实现各种并行算法的原理、方法、流程和最佳实践，以及如何对 OpenCL 进行性能优化，如何对硬件进行探测和调整。

本书分为两大部分：第一部分（1～13 章），从介绍 OpenCL 的核心思想和编写 OpenCL 程序的基础知识开始，对枯燥的 OpenCL 规范进行了深刻而系统的解读，旨在帮助读者全面、正确地理解 OpenCL 规范及其编程模型；第二部分（14～22 章），提供了一系列经典的案例，如图像直方图、Sobel 边界检测过滤器、并行实现 Dijkstra 单源最短路径图算法、Bullet Physics SDK 中的布模拟、用快速傅里叶变换模拟海洋、光流、OpenCL 与 PyOpenCL 结合使用，使用 OpenCL 完成矩阵相乘与稀疏矩阵矢量乘法等，目的是让读者通过案例熟练掌握编写复杂并行程序的方法和技巧。本书的附录收录了 OpenCL 规范定义的大量函数、命名常量和类型，可供程序员开发时查阅。

本书版权登记号：图字：01-2011-5828

图书在版编目（CIP）数据

OpenCL 编程指南 /（美）蒙施（Munshi, A.）著；苏金国等译 . —北京：机械工业出版社，2012. 11

书名原文：OpenCL Programming Guide

ISBN 978-7-111-39849-3

Ⅰ. O…　Ⅱ. ①蒙…　②苏…　Ⅲ. 图形软件－程序设计　Ⅳ. TP391. 41

中国版本图书馆 CIP 数据核字（2012）第 226824 号

机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码　100037）

责任编辑：盛思源

北京市荣盛彩色印刷有限公司印刷

2013 年 1 月第 1 版第 1 次印刷

186mm × 240mm · 26. 5 印张

标准书号：ISBN 978-7-111-39849-3

定价：89. 00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88378991；88361066

购书热线：（010）68326294；88379649；68995259

投稿热线：（010）88379604

读者信箱：hzjsj@ hzbook. com

译 者 序

没有人否认现在是信息爆炸的时代，日常生活中需要处理的数据也日益增长。海量的数据带来了各式各样数据处理的需求，如网络浏览、平面或者3D图形加速、数据服务器、分布式云计算等，而现在各计算平台之间的兼容性却不容乐观。

OpenCL 1.0规范在2008年12月浮出水面。OpenCL是Open Computing Language（开放计算语言）的缩写。设立OpenCL的目的就是为日益庞大的并行计算市场提供一个开放的、免费的行业标准。它让开发人员能够利用CPU、GPU等计算设备内部巨大的并行计算能力。OpenCL不仅得到了Apple等诸多大公司的支持，而且目前宣称支持OpenCL并参与其中的成员几乎涵盖了全球主要的处理器、计算芯片供应商，这一切要归功于OpenCL的开放、高度通用的设计原则。为了保证通用计算，OpenCL在五个方面进行了规定：系统调用全部的硬件资源；将C语言作为并行程序模型的基础，加快OpenCL程序的研发速度以及保证可移植性；做到与现有软件体系结构通用；实现硬件平台上的通用；提供承前启后，向前兼容的通用支持。

尽管大多数人对OpenCL的并行计算和数据处理领域的突出地位有所了解，但对如何利用OpenCL规范编写复杂的并行程序还是一头雾水，不知如何下手。特别是，尽管OpenCL规范很全面，但从诸多繁杂的细节中找出直接需要的东西并不容易，这也让很多有心加入OpenCL编程队伍的人望而却步。

不过，本书为OpenCL世界带来了福音。本书出自资深程序员之手，正是程序员需要的实用指南。本书第一部分介绍了OpenCL 1.1规范的全部内容，包括OpenCL的基本核心思想以及编写OpenCL程序的基础知识。第二部分给出了大量实际用例，可以从中了解OpenCL的各个方面在复杂应用中如何工作，学习如何在实际项目中使用OpenCL。

正如作者们所说，“我们正处在一场计算变革当中，正如当年PC的出现。更准确地说，我们很有可能要迎接一场革命，因为如果没有并行软件，异构硬件提供的高度并行性就毫无意义。但事实上，在特定的小环境之外，并行软件相当稀缺”，所以“加入我们的行列吧”！读完本书，相信你能编写出可以合理分配计算负载的复杂并行程序，充分利用OpenCL实现性能优化。

本书由苏金国、李璜、杨健康主译，乔会东、仝磊、王少轩、程芳、宋旭民、黄小钰分别对全书各章进行审阅。另外，姚曜、程龙、吴忠望、张练达、陈峰、江健、姚勇、卢鋆、张莹参与了全书的修改整理，林琪、刘亮、刘跃邦、高强和王志淋统一了全书术语，并完善了关键部分的翻译。全体人员共同完成了本书的翻译工作。不过由于水平有限，译文肯定有不当之处，敬请批评指正。

序

过去几年间，由 CPU 和 GPU 组成的异构计算机为计算领域带来了一场革命。通过将工作负载的不同部分匹配到最适合的处理器，可以极大地提升计算机的性能。

GPU 等多核处理器的出现引发了这场革命。例如，现在可以购买一块每秒执行上万亿次浮点操作（teraflops）的图形卡。这些 GPU 专门设计用来渲染酷炫的图像，不过如果工作负载合适，这些 GPU 也可以用做高性能的计算引擎，来完成从科学计算到增强现实等一系列应用。

人们很自然地提出一个问题：与传统的单核 CPU 相比，这些多核处理器为什么速度如此之快。最根本的驱动力来自创新的并行硬件。之所以并行计算比串行计算更为高效，是因为芯片本质上是并行的。现代芯片包含数十亿个晶体管。多核处理器将这些晶体管组织为多个并行处理器，分别由数百个浮点单元组成。多核处理器的速度提升还有一个重要的原因，这就是新的并行软件。要充分利用所有这些计算资源，就要求我们开发并行程序。由于软件和硬件带来的效率提升，使我们能够得到比单核 CPU 更大的每瓦特或每美元 FLOP 数。

计算系统由硬件和软件共同组成。如果没有好的编程模型，硬件就没有多大用处。CPU 的成功要归结于其编程模型的成功，具体体现为 C 语言及其后续语言。C 语言很好地抽象了一个串行计算机。要充分利用异构计算机，我们需要新的编程模型能够很好地抽象一个现代并行计算机。可以参考图形领域成熟的技术，帮助建立异构计算所需要的新的编程模型。

多年来，我一直对图形的编程模型很感兴趣，最早可以追溯到 1988 年。那时我是 PIXAR 的一名软件工程师，开发了一种 RenderMan 着色语言。十年后图形系统已经变得足够快，可以考虑为 GPU 开发着色语言了。通过与 Kekoa Proudfoot 和 Bill Mark 合作，我们开发了一种实时着色语言 RTSL。RTSL 在图形硬件上运行，可以将着色语言程序编译为像素着色程序（这是当今面向图形硬件的汇编语言）。后来 Bill Mark 加入了 NVIDIA，在那里开发了 Cg。最近，我与 Tim Foley 同在 Intel 工作，他开发了一种名为 Spark 的新的着色语言。Spark 将着色语言提高到一个新的层次，可以抽象能提供镶嵌等新功能的复杂图像管线。

在开发这些语言时，我很清楚 GPU 不仅仅可以用于处理图形数据。很多组织已经通过实验证明了图形硬件还可以用于图形以外的其他应用。这就催生了通用 GPU（General-Purpose GPU，GPGPU）运动。通过使用图形库，对之前的展示实例进行调整。为了让 GPU 得到更广泛的使用，需要一个并不限定于图形的更通用的编程环境。为了满足这个需要，我们在斯坦福大学启动了针对 GPU 的 Brook 项目。Brook 的基本思想是把 GPU 当做一个数据并行处理器。数据并行编程方法对于完成并行计算极为成功，通过 Brook，我们能够展示在 GPU 上实现数据并行编程原语。Brook 允许开发人员采用一种广泛使用的并行编程模型编写应用程序。

Brook 主要是一个概念验证项目。斯坦福大学的一名研究生 Ian Buck 后来加入 NVIDIA 开发 CUDA。CUDA 在很多重要方面延伸了 Brook。它引入了合作线程数组的概念，即线程块（thread

block）。合作线程数组可以描述 GPU 核的局部性，执行相同程序的一组线程可以通过局部内存通信，并通过栅栏同步。更重要的是，CUDA 为 GPU Computing 创建了环境，已经得到大量应用开发人员、中间件提供商和开发商的参与。

开放计算语言（Open Computing Language，OpenCL）将 GPU Computing 的核心思想进一步延伸，带来了普适异构并行计算的新纪元。OpenCL 由 Khronos Group 精心设计，并得到很多开发商和软件专家的支持。人们使用 CUDA 创建一个可以由多个开发商实现的软件标准时获得了很多经验，OpenCL 的开发也得益于这些经验。现在 OpenCL 实现可以在广泛使用的硬件上运行，包括 NVIDIA、AMD 和 Intel 的 CPU 与 GPU，还可以在基于 DSP 和 FPGA 的平台上运行。通过标准化编程模型，开发人员可以使用更多软件工具和硬件平台。

关于 OpenCL 最令人兴奋的是，它不仅对已经完成的工作进行标准化，而且表现出一个积极群体正在为推进并行计算所投入的热情和努力。例如，OpenCL 提供了在 GPU 上调度任务的创新功能。另外，OpenCL 开发人员已经结合了任务并行和数据并行计算的最佳特性。希望 OpenCL 将来的版本有同样的创新性。与其前身 OpenGL 一样，OpenCL 会在今后一段时间继续发展，开发出能提供更多功能的新版本。

本书介绍了完整的 OpenCL 编程模型。合作者之一 Aaftab 是这个系统的灵魂人物。在他的号召之下，多位 OpenCL 核心设计人员加入，从而编写了这本容易理解的权威指南。欢迎来到异构计算的新世界。

——Pat Hanrahan

斯坦福大学教授

前　言

行业专家总喜欢戏剧性变化，不是通过改进现有产品来开发新产品，他们更喜欢“变革”，或者更好地定义一个“新模式”。当然，按照技术发展的方式，最终结果很少能像这些专家预期的那么戏剧性。

不过，在过去的十年间，确实发生了一些革命性的变化。多核 CPU 的出现使得并行硬件无处不在。GPU 不再只是专用的图形处理器，它们也成为重量级的计算引擎。另外，CPU 和 GPU 的组合（也就是一般所称的异构平台）确实重新定义了计算的标准构成模块。

我们正处在一场计算变革当中，正如当年 PC 的出现。更准确地说，我们很有可能要迎接一场革命，因为如果没有并行软件，异构硬件提供的高度并行性毫无意义。事实上，在特定的小环境之外，并行软件相当稀缺。

为了与正在发生的（并行）异构计算变革同步，发动一场并行软件的革命，我们需要一个并行软件行业。不过，只有软件能够在平台之间迁移，既要保证跨开发商，也要保证跨代，这个行业才能发展壮大。解决之道就是建立异构计算的一个行业标准。

OpenCL 正是这样一个行业标准。OpenCL 由 Khronos Group（因 OpenGL 和其他标准而闻名）创建，最初萌生于软件开发商、计算机系统设计者（包括移动平台设计者）和微处理器（包括嵌入式处理器、加速器、CPU 和 GPU）制造商的一次合作。对于人们提出的问题：“在异构平台上编程时，如何保证现在创建的软件将来仍有效”，OpenCL 给出了答案。

OpenCL 诞生于 2008 年，如今可在多种平台上由多个来源得到。它一直在稳步发展，以保持与最新的微处理器同步发展。在本书中我们将重点介绍 OpenCL 1.1，全面介绍这个标准，并给出丰富的例子解释 OpenCL 在实际中的使用。

本书读者对象

本书出自程序员之手，同时面向程序员。对于要编写代码的人来说，这是一本非常注重实效的指南。我们认为读者已经了解 C 和 C++（书中的某些部分要求有 C++ 基础）。最后，我们还假设读者已经熟悉并行编程的基本概念。我们相信读者手边有一台计算机，可以编写软件，研究在书中学到的新知识。因此，本书提供了大量程序和代码段。

我们会全面介绍 OpenCL 1.1 规范，并解释如何实现大量并行算法。学完本书后，你将能够编写复杂的并行程序，将工作负载分散到异构平台的多个设备上。你还将掌握 OpenCL 中性能优化的基础，了解如何编写软件，从而探查硬件，并相应地做出调整以得到最优性能。

本书组织结构

OpenCL 规范大约有 400 页。这是一份晦涩、复杂的文档，其中满是乏味的具体细节。要想解释这个规范可不是件容易的事情，不过我们认为本书很好地做到了这一点。

本书分为两部分。第一部分描述 OpenCL 规范。前两章介绍 OpenCL 的基本核心思想以及编写 OpenCL 程序的基础知识。然后开始系统地研究 OpenCL 1.1 规范。通过结合参考材料，并给出解释说明，使本书的风格有所改变。第二部分提供了一系列案例研究。通过这些简单的示例，可以了解 OpenCL 的各个方面在复杂应用中的工作方式，从而展示如何在正式的应用项目中使用 OpenCL。下面给出更详细的介绍，帮助你了解本书。

第一部分：OpenCL 1.1 语言与 API

- **第 1 章**：这一章高度概括了 OpenCL。首先详细解释了为什么在可预见的将来异构并行平台将主宰计算领域，然后描述了 OpenCL 的核心模型和概念。在这个过程中，还将给出 OpenCL 中使用的术语，因此，即使你只打算快速浏览本书，把它作为一个 OpenCL 参考指南，这一章也很重要。
- **第 2 章**：真正的程序员要通过编写代码来学习。因此，作为对 OpenCL 介绍的补充，我们专门用一章来研究一个实际的 OpenCL 程序。通常在介绍一种编程语言时，首先会给出一个例子，在屏幕上输出“hello world”，这似乎已经成为一个惯例。但在 OpenCL 中，这是没有意义的（OpenCL 根本没有输出语句）。在数据并行编程领域，与“hello world”对应的是另外一个简单程序：让两个数组逐元素相加。这个程序将作为这一章的核心。在这一章的最后，你会充分了解 OpenCL，可以开始编写简单程序。我们强烈建议你立即着手编写自己的程序。只看书不动手编程，是无法学会一种编程语言的。动手写代码吧。
- **第 3 章**：从这一章开始，将系统研究 OpenCL 规范。要让 OpenCL 程序做些“有意思”的事情，首先需要找到可用的资源，然后准备好做有用的工作。换句话说，程序必须发现平台，为 OpenCL 程序定义上下文，并确定如何使用这些设备。这些重要内容都将在这一章介绍，并详细描述 OpenCL 平台 API。
- **第 4 章**：在 OpenCL 设备上运行的代码大多数情况下都使用 OpenCL C 编程语言来编写。在一个 C99 子集基础上，OpenCL C 编程语言提供了内核有效利用 OpenCL 设备所需的支持，其中包括一组丰富的矢量指令。这一章将详细解释这种编程语言。
- **第 5 章**：OpenCL C 编程语言 API 定义了一个庞大且复杂的内置函数集，将在这一章介绍。
- **第 6 章**：了解了用来编写内核的语言之后，接下来转向 OpenCL 定义的运行时 API。首先介绍程序和内核的过程。要记住，OpenCL 中*程序*一词有新的含义。在 OpenCL 中，*程序*专门指“动态库”，内核可以从中获取函数。

- **第 7 章**：这一章介绍缓冲区内存对象（1 维数组），还会详细讨论子缓冲区。子缓冲区是 OpenCL 1.1 中新增的一个特性，因此有 OpenCL 1.0 经验的程序员会发现这一章尤其有用。
- **第 8 章**：这一章介绍另一个内存对象：图像——这是一个非常重要的内容。由于图形和 OpenCL 之间存在紧密的关系，因此这些内存对象对于很多 OpenCL 程序员来说都很重要。
- **第 9 章**：这一章将详细讨论 OpenCL 中的事件模型。这些对象用来强制 OpenCL 中的顺序约束。基本上，通过事件编写并发代码，不论运行时如何调度工作，都能生成正确的答案。不过，从算法的角度来讲，在更高层次上，事件可以支持程序指令作为跨多个设备的有向无环图。
- **第 10 章**：很多应用可能尝试使用图形 API 来显示 OpenCL 处理的结果，或者甚至使用 OpenCL 处理由图形生成的场景。OpenCL 规范允许与 OpenGL 图形 API 互操作。这一章将讨论如何建立 OpenGL/OpenCL 共享，以及如何共享和同步数据。
- **第 11 章**：OpenCL 应用常用于 Microsoft 系列平台。OpenCL 应用包括图形时，可能需要连接到 Microsoft 的内置图形 API。在 OpenCL 1.1 中，定义了如何将一个 OpenCL 应用连接到 DirectX 10 API。这一章将介绍如何建立 OpenCL/Direct3D 共享，以及如何共享和同步数据。
- **第 12 章**：这一章讨论了 OpenCL C++ API 包装器。包装器可以大大简化用 C++ 编写的宿主机程序、寻址自动引用计数，还提供了可以查询 OpenCL 对象信息的一个统一接口。一旦掌握了 C++ 接口，可能就不想再使用常规的 C 接口了。
- **第 13 章**：OpenCL 面向的设备种类繁多，从移动电话到庞大的并行超级计算机中的节点。OpenCL 规范的大部分内容可以不加修改地应用于这些设备。不过，对于在嵌入式设备中使用的低功耗处理器，为适应其缩减的功能，需要对 OpenCL 做少量修改。这一章介绍了这些修改，在 OpenCL 规范中称为 OpenCL 嵌入式简档。

第二部分：OpenCL 1.1 案例研究

- **第 14 章**：直方图可以报告数据集中值的出现频率。例如，在这一章中，我们计算了一个彩色图像的 R、G、B 通道值的直方图。要并行生成直方图，可以对数据集的局部区域计算值，再累加这些局部值生成最终结果。这一章有两个目标：1）展示在 OpenCL 中如何处理图像；2）研究在 OpenCL 程序中高效完成直方图全局累加的技术。
- **第 15 章**：Sobel 边界过滤器是一个有向边界检测过滤器，会沿 x 轴和 y 轴计算图像梯度。在这一章中，介绍了使用一个内核应用 Sobel 边界过滤器的简单例子，展示了在 OpenCL 中内核如何处理图像。
- **第 16 章**：这一章提供了 OpenCL 中 Dijkstra 单源最短路径图算法的一个实现，可以充分利用 CPU 和多个 GPU 设备。图数据结构在很多问题中都得到了应用，从人工智能到神经影像。这个特定实现是作为 FreeSurfer 的一部分开发的，它是一个神经影像应用，目的是改善算法的性能，该算法要测量大脑皮层三角网格重构的曲率。这个例子可以很

好地说明如何使用多个 OpenCL 设备，以及同时在 CPU、多个 GPU 或者所有设备上划分工作负载。

- **第 17 章**：物理模拟是现代视频游戏的一个有力补充，目前还在不断发展中，这一章提供了使用 OpenCL 模拟布的一种方法（如战士的衣服），这是 Bullet Physics SDK 的一部分。模拟柔软物体有很多方法：Bullet 中使用的模拟方法类似于一种质点/弹簧模型，并为在现代 GPU 上执行进行了优化，同时平滑地集成其他 Bullet SDK 组件（未采用 OpenCL 编写）。这里展示了一种重要的技术，称为分批，即转换粒子网格，从而在 SIMD 体系结构上高效执行（如 GPU），同时保持质点/弹簧模型中的依赖关系。
- **第 18 章**：这一章详细介绍了 AMD 的 Ocean（海洋）模拟。Ocean 是一个 OpenCL 演示程序，使用了逆离散傅里叶变换来实时模拟海洋。快速傅里叶变换应用于随机噪声，作为频率依赖相移随时间生成。我们介绍了基于傅里叶变换方法完成的一个实现，这种方法最初由 Jerry Tessendorf 开发，已经用于很多故事片的特效处理，如《未来水世界》、《泰坦尼克号》和《第五元素》。这一章介绍了一个优化 2D DFFT 的开发，包括 OpenCL 编程中很多很有用的重要优化手段，还介绍如何将这个算法集成到应用当中，以及如何使用 OpenCL 和 OpenGL 的互操作性。
- **第 19 章**：这一章提供了用 OpenCL 完成的一个光流实现，这是计算机视觉领域的一个基本概念，用来描述图像中的运动。光流在图像稳定性、帧速率上采样有很多应用，还可以作为更高层算法的输入，如对象跟踪和手势识别。这一章给出了用 OpenCL 实现的金字塔 Lucas-Kanade 光流算法。这个实现展示了如何使用图像对象访问 GPU 硬件的纹理特性。我们还介绍如何使用 GPU 上的纹理过滤硬件完成数据的线性插值，从而得到所需的亚像素准确性，并相应地显著提升速度。另外，我们还讨论如何使用共享内存缓存反复访问数据，以及如何利用内核提前退出技术以提高效率。
- **第 20 章**：这一章的目的是介绍在 Python 中使用 OpenCL 的基本知识。本书主要强调通过 C/C++ 使用 OpenCL，不过为其他一些语言（如 Python）也开发了 OpenCL 绑定。这一章将逐步介绍 PyOpenCL，我们给出将第 8 章中的高斯图像过滤示例移植到 Python 所需要的各个步骤。除了说明从 C++ 移植到 Python 所需要的修改外，这一章还讨论在诸如 Python 等动态类型语言中使用 OpenCL 的一些好处。
- **第 21 章**：这一章讨论了一个将两个方阵相乘的程序。这个程序很简单，因此很容易了解优化其性能时对程序所做的修改。这些优化主要强调 OpenCL 内存模型，以及如何使用这个模型尽可能减少 OpenCL 程序中数据移动的开销。
- **第 22 章**：这一章介绍了使用 OpenCL 完成稀疏矩阵矢量乘法的一个优化实现。稀疏矩阵定义为很大的 2 维矩阵，其中矩阵的大部分元素都等于 0。很多领域中都使用了稀疏矩阵来描述和解决问题，如计算流体力学、计算机图形/视觉、机器人/动力学、金融建模、声学和量子化学。这个实现展示了 OpenCL 能够在硬件特定代码（快速但不可移植）和单源代码（可移植但很慢）之间搭桥，能够在多种硬件上得到一个高效的高性能实现，几乎与硬件特定实现的速度一样快。这些结果通过用 OpenCL C 编写的内核完

成，可以在任何符合规范的 OpenCL 平台上编译和运行。

附录

- **附录 A**：OpenCL 规范定义了大量函数、命名常量和类型。甚至专家级 OpenCL 程序员在编写代码时也需要查找这些细节。为了提供帮助，我们增加了一个附录，将所有这些细节集中在一起。

示例代码

本书包含大量示例程序。可以从本书网站（www. openclprogrammingguide. com）下载这些示例。

勘误

如果你发现本书中有错误，请发邮件通知我们（errors@ opencl-book. com）。本书勘误表见本书网站（www. openclprogrammingguide. com）。

致　谢

来自 Aaftab Munshi

与 Ben、Dan、Tim 和 James 合作完成本书真是荣幸。我要感谢审校人员 Andrew Brownsword、Yahya H. Mizra、Dave Shreiner 和 Michael Thurston，多谢他们花时间校对本书，并提出宝贵的反馈意见，让本书增色不少。另外要感谢 Pearson 的编辑 Debra Williams Cauley，是她的大力帮助才促成了本书的出版。

还要感谢我的女儿 Hannah 和 Ellie，以及我最爱的妻子 Karen，没有她们，本书绝无可能问世。

来自 Benedict R. Gaster

感谢 AMD 对 OpenCL 工作提供的支持。特别要感谢 4 个人，是他们帮助我了解了 GPGPU 变革：Mike Houston、Justin Hensley、Lee Howes 和 Laurent Morichetti。

如果没有 Santa Cruz 的快乐生活，没有与 Miranda、Maude、Polly 和 Meg 的海滩之旅，就不会有本书。谢谢他们！

来自 Timothy G. Mattson

感谢 Intel 允许我自由地投入 OpenCL 工作。特别感谢 Intel 的 Aaron Lefohn 在 OpenCL 发展之初引领我进入这个项目。不过，最重要的是，感谢 OpenCL 工作组非凡的人们，从这群忘我工作的专业人士身上我学到了很多。

来自 James Fung

很荣幸与合作者们共同努力完成了本书。还要感谢 NVIDIA 在我写本书时提供的所有支持，另外要感谢家人和朋友的支持和鼓励。

来自 Dan Ginsburg

我要感谢 Children's Hospital Boston 的 Rudolph Pienaar 医生和 Ellen Grant 医生在我写本书时给予的支持，感谢他们的宝贵意见和卓越见解。能与 Affie、Ben、Tim 和 James 共同完成本书真是我的莫大荣幸，他们绝对是并行计算领域的精英人物。还要感谢编辑 Debra Williams Cauley，感谢她的耐心和敬业精神，这绝对是这个项目成功的关键。

目　录