生成 Verilog 描述，它适合合成到实际可以运行的硬件上去。

- 第 5 章：优化程序性能。在这一章里，我们介绍了许多提高代码性能的技术，主要思想就是让程序员通过使编译器能够生成更有效的机器代码来学习编写 C 代码。我们一开始介绍的是减少程序需要做的工作的变换，这些是在任何机器上写任何程序时都应该遵循的。然后讲的是增加生成的机器代码中指令级并行度的变换，因而提高了程序在现代“超标量”处理器上的性能。为了解释这些变换行之有效的原理，我们介绍了一个简单的操作模型，它描述了现代乱序处理器是如何工作的，然后给出了如何根据一个程序的图形化表示中的关键路径来测量一个程序可能的性能。你会惊讶于对 C 代码做一些简单的变换能给程序带来多大的速度提升。
- 第 6 章：存储器层次结构。对应用程序员来说，存储器系统是计算机系统中最直接可见的部分之一。到目前为止，读者一直认同这样一个存储器系统概念模型，认为它是一个有一致访问时间的线性数组。实际上，存储器系统是一个由不同容量、造价和访问时间的存储设备组成的层次结构。我们讲述不同类型的随机存取存储器(RAM)和只读存储器(ROM)，以及磁盘和固态硬盘[⊖]的几何形状和组织构造。我们描述这些存储设备是如何放置在层次结构中的，讲述访问局部性是如何使这种层次结构成为可能的。我们通过一个独特的观点使这些理论具体化，那就是将存储器系统视为一个“存储器山”，山脊是时间局部性，而斜坡是空间局部性。最后，我们向读者阐述如何通过改善程序的时间局部性和空间局部性来提高应用程序的性能。
- 第 7 章：链接。本章讲述静态和动态链接，包括的概念有可重定位的和可执行的目标文件、符号解析、重定位、静态库、共享目标库、位置无关代码，以及库打桩。大多数讲述系统的书中都不讲链接，我们要讲述它是出于以下原因。第一，程序员遇到的最令人迷惑的问题中，有一些和链接时的小故障有关，尤其是对那些大型软件包来说。第二，链接器生成的目标文件是与一些像加载、虚拟内存和内存映射这样的概念相关的。
- 第 8 章：异常控制流。在本书的这个部分，我们通过介绍异常控制流(即除正常分支和过程调用以外的控制流的变化)的一般概念，打破单一程序的模型。我们给出存在于系统所有层次的异常控制流的例子，从底层的硬件异常和中断，到并发进程的上下文切换，到由于接收 Linux 信号引起的控制流突变，到 C 语言中破坏栈原则的非本地跳转。

在这一章，我们介绍进程的基本概念，进程是对一个正在执行的程序的一种抽象。读者会学习进程是如何工作的，以及如何在应用程序中创建和操纵进程。我们

⊖ 直译应为固态驱动器，但固态硬盘一词已经被大家接受，所以沿用。——译者注