

- <http://geda.weul.org>, 开源EDA工具的收藏集。
- www.veripool.com, 基于Verilog的工具的收藏集, 也是Dinotrace和Verilator的主页。
- <http://ghdl.free.fr>, 一个使用gcc的开源VHDL前端工具。
- <http://asics.ws>, 一些开源IP核。

在网上冲浪时, 可以找到许多其他的与EDA和FPGA相关的开源项目, 不幸的是, 大多数处于停滞状态或者由于没有达到应有的功能而被放弃了。也许你偶然碰到一些有用的工具, 或者你自己开发了有用的工具, 那么请联系email:

[418] max@techbites.com, 有可能在本书下一版中包含进去。

第 26 章 FPGA未来的发展

26.1 一种担忧

1980年我开始设计大型计算机CPU的职业生涯时，并没有今天所使用的任何技术和工具，没有电路图输入工具包，就用铅笔和纸张画出门级电路图；没有逻辑仿真器（有些早期的版本可供使用，但我们没有），就靠目视检查来验证设计，是否通过验证要取决于看电路图的其他工程师，当他们说：“我看这个电路图没有问题”时就表示设计通过了验证。

像Verilog和VHDL这样复杂的HDL更是遥不可及的事情，而且逻辑综合这样的工具在某一天出现的可能性对我们来说简直就是零。谈到逻辑优化和最小化，我们的设计团队中有一位中国工程师非常精于此道，设计交给他后，只用一天左右的时间就能提交出一个优化的版本。时序分析方面，我们还要再返回到铅笔和纸张上，手工计算延迟路径（连最基本的电子计算器我们都没有）。

那个时期，工作中所用的是几微米的ASIC工艺，一般只包含几千个逻辑门（那时还没有发明FPGA）。如果那时有人说，2003年将会以90nm工艺设计ASIC和SoC，其中包含的逻辑门会达到几十到几百万门，而且还有可重配置器件，就像今天的基于SRAM的FPGA一样，我会对此一笑了之。同样，如果说有一天我将拥有一台个人计算机，配置了几百MB的内存，CPU主频达到2GHz以上，硬盘容量60GB，完全可以运行今天的EDA工具，那么我一样会平静地笑一下，然后溜之大吉^①。电子学发展得确实太快，任何预测都可能成为人们谈论的笑柄。

26.2 下一代结构和技术

26.2.1 十亿晶体管级器件

我十分确信，下一代FPGA将会含有十亿以上的晶体管（之所以这么自信是因为Xilinx公司最近宣布要推出这类器件）。这些芯片以2003年后期和2004早期出现的90nm工艺制造，紧接着还会有更大容量的器件，以2004年和2005年间的65nm到70nm工艺制造。

^① 第一台IBM个人计算机直到1981年才出现。

26.2.2 超快速I/O

回顾第21章讨论过的吉比特收发器，今天的高端FPGA一般都会包含多个这种收发器模块，每一个收发器都支持多个通道，每个通道的真实数据传输率为2.5Gbit/s，所以要实现10Gbit/s的速率必须将四个通道结合起来。此外，还要用一个外部器件将输入的光学信号转换为四路电信号，以便传送给FPGA；反过来传输时，这个外部器件要从FPGA中接收四路电信号并将这些电信号转换为一路输出光学信号。写作本书时，有些FPGA已经具备了在内部接收和产生10Gbit/s速率光学信号的能力。

未来某些阶段可能出现另一种技术，就是FPGA之间或者FPGA与ASIC之间可以进行无线或类无线的芯片间通信。在这里使用的类无线，是指一些现在正处于试验阶段的技术，例如Sun微系统公司正在以极快的低功耗电容耦合为基础开发芯片间通信技术，这一技术需要电路板上相关的芯片安装得非常靠近，不过提供的芯片间的信号速度应该比今天最好的板级互连技术还要快60倍以上。

26.2.3 超快速配置

现在的绝大多数FPGA都使用串行位流或者只有8位的并行位流进行配置，这严重限制了这些器件在可重配置计算中的应用。大约在20世纪90年代中期，英国Pilkington微电子公司（Pilkington Microelectronics, PMEL）提出了一种新奇的FPGA结构，对主要的输入和输出引脚进行复用，使用它们装载配置数据。这种结构提供了一种极宽的总线，可达到256引脚/位以上，因此对器件进行配置能在瞬间（jiffy）完成^①。

举一个例子来说明这种结构的可用性，考虑这样一个事实，即各种各样的压缩/解压缩（compressor/decompressor, CODEC）算法都可以对音频和视频数据进行压缩和解压缩。如果有一个系统需要解压缩各种不同算法压缩的文件，那么这个系统就必须支持各种不同的CODEC。

假定用FPGA来执行这一解压缩任务，那么使用传统的器件时，每一种CODEC就需要一个器件或者一个更大规模的器件并从中为每一种CODEC分配不同的区域。如果让系统停下来以便重新配置FPGA执行不同的算法，可能你不愿意这么做，因为配置一个大的器件会用去1秒到2.5秒的时间，这对于一个终端用户来说太长（现在需要的是立即满意）。相比而言，如果使用PMEL的结构，在处理

^① “jiffy”的官方定义是“很短的时间间隔”、“瞬间”或者“即时”。工程师可能会将“jiffy”看作是计算机系统时钟的一次周期性变化时间，这一时间的长短和各个国家的主干电力网周期有关，例如在美国和加拿大，相当于一秒种的1/60，而在英国和其他大部分地区都相当于1/50秒。最近，认为jiffy等于1/100秒开始变得流行起来，物理学家为了增加乐趣，有时候也认为jiffy相当于光在真空中传播一英尺所用的时间（这一时间非常接近于1ns）。

文件前就可以将重配置数据加载进去，如图26-1所示。

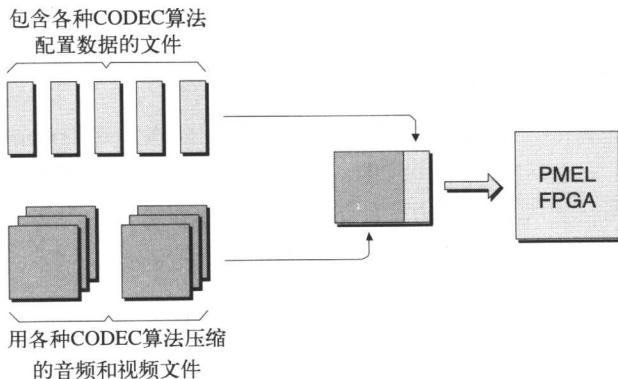


图26-1 宽配置总线

这种方法的思路是通过宽的总线将配置数据以极快的速度读入，并在极短的时间内配置好器件，接着立即执行音频和视频文件的解压缩任务。如果下一个要处理的文件需要另一种CODEC，就选用对应的配置文件对器件重新编程即可。422

这一概念可以用于多种应用。不幸的是，这一技术在中途就夭折了，但是不远的将来重新出现类似的技术并不是不可能的^①。

26.2.4 更多的硬IP

90nm及以下工艺中，完全有可能在一个芯片上集成更多的晶体管，因此几乎可以肯定的是，我们将会看到越来越多的硬IP模块，例如通信功能模块、专用处理功能模块和微处理器外围设备等。

26.2.5 模拟与混合信号器件

传统的FPGA厂商都有一个强烈的愿望，就是要尽可能多地“抢夺”电路板的功能并将这些功能吸收到他们的器件中。短期内，FPGA器件可能先包含一些具有模拟功能的硬IP模块，例如模数转换（analog-to-digital, A/D）和数模转换（digital-to-analog, D/A）单元，这些模块有些特征是可编程的，例如量化的阶数（宽度）和所支持的模拟信号的动态范围，它们还有可能包含放大和滤波功能，以及信号适应功能。

此外，多年以来许多公司已经开发出了各种各样的现场可编程模拟阵列（field-programmable analog arrays, FPAA）^②。这样，有很多机会使主流的数字FPGA器件

^① 第23章中介绍的现场可编程节点阵列（field programmable node array, FPNA）中，有些就使用了宽总线配置方案。

^② 例如，Anadigm (www.anadigm.com) 公司生产一些很有意思的器件。

可以包含真正可编程的模拟功能区域，这些模拟功能与纯FPA器件提供的模拟功能是相似的。

423

26.2.6 ASMBL与其他结构

正如本章提到的，Xilinx公司正式宣布了他们即将采用的Application Specific Modular Block，(ASMBL™) 结构，其主要思想是有一个基于列的结构，Xilinx公司的开发人员付出大量的努力设计了各种不同的列结构，例如：

- 通用可编程逻辑
- 存储器
- DSP为中心的功能
- 处理功能
- 高速I/O功能
- 硬IP功能
- 混合信号功能

Xilinx将以现货方式供应这些器件，对于不同应用领域的器件也会混合不同的

424

列结构类型，如图26-2所示。

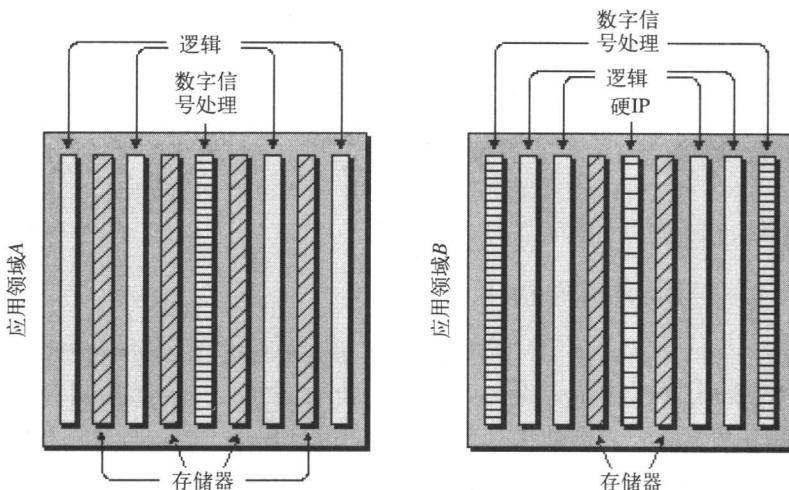


图26-2 使用基本的ASMBL结构创建各种现货供应的面向专用领域的器件

当然，其他FPGA厂商毫无疑问也在积极开发各自的下一代器件。未来几年内，我们有望看到新结构的一次爆发。

26.2.7 不同的结构粒度

第4章中讨论过，一些FPGA厂商和大学生已经花了大量的时间研究三输入、

四输入、五输入甚至六输入LUT的优点。

过去，某些器件混合使用了各种不同的LUT，例如三输入和四输入的LUT，这样可以大大提高器件的利用率。由于各种各样的原因，今天绝大多数的FPGA只包含一种四输入LUT，但是未来完全有可能出现混合使用各种不同大小LUT的器件。

26.2.8 ASIC结构中的嵌入式FPGA内核

在90nm工艺节点上开发一个现代的ASIC芯片，其成本是极其巨大的，而且，一旦完成设计并制成了芯片，所使用的算法和芯片功能就被有效地“固化在硅中”不能更改，这就是说，如果你在将来想要做一些修改，就不得不重新进行设计，制造一套新的光掩模（成本大约为100万美元），然后再制造一个全新的芯片。

为了解决这些问题，有些用户对于将FPGA内核嵌入到ASIC结构中比较感兴趣，也就是说，你可以在多种应用中使用同一个设计，而不用制造一套新的掩模。写作本书时，该技术最新的表现形式为IBM公司和Xilinx公司提出的XBlue结构，这些器件以90nm工艺制造，于2004年开始供货。

作者也认为，结构化ASIC的使用将有所增加，这类器件与嵌入式FPGA内核有异曲同工之妙，因为他们的设计风格和工具具有许多共同之处。

425

26.2.9 ASIC和FPGA结构中嵌入FPNA内核或者相反

第23章中，我们讨论了FPGA和ASIC结构中嵌入FPNA的概念，也讨论了在FPNA结构中嵌入基于FPGA的节点，如果这些实现了，有一天我们也将设计出嵌入了FPGA内核的ASIC，这个FPGA内核本身还包含一个嵌入式FPNA内核，而这个FPNA内核又包含了基于FPGA的节点，这让人大脑有点晕了。

26.2.10 基于MRAM的器件

第2章中，我们介绍了MRAM的概念。MRAM的存储单元具有结合SRAM的高速度、DRAM的大容量和FLASH的非易失性的功能，而消耗的功率却很少。

基于MRAM的存储芯片预计于2005年前后面世。一旦这些存储芯片推向市场，其他的基于MRAM技术的FPGA器件就会很快出现。

26.3 设计工具

上面已经讨论过，下一代FPGA器件将含有十亿以上的晶体管。现有的基于HDL的设计流程在RTL抽象级上输入设计的方法在当前的一代器件上已经开始显得有点捉襟见肘，用不了多久，这种方法就会消失。

要取得进展，一个很有用的方法就是使用第11章中介绍的纯C/C++的设计流程，以便提高设计的抽象层次。但是，实际上我们需要的是真正的系统级设计环境，可以帮助用户在极高的抽象层次上探索设计空间，除了算法建模与验证外，这些环境还要有助于划分设计的软件和硬件模块。

426 这些系统级环境也要能够提供性能分析功能以帮助用户评估哪些模块用软件实现时太慢而需要用硬件来实现，以及哪些用硬件实现的模块实际上用软件就可以实现，以便优化芯片的资源使用。

长久以来，人们一直在讨论此类问题，各种可用的环境和工具也在朝着解决这些问题的方向前进。但实际上，这些环境和工具在性能和易用性方面还有很长的路要走。

26.4 期待意外的发生

这是本章的结尾也是这本书的结尾。但是在结束前，我只想重申一下，大家对于未来的猜想可能只是真实情况的一种粗浅的反映，一些器件工艺和设计工具仍然在构思当中，它们最终出现的时候（基于过去的经验，这种情况要比我们想象的要快），我们都将会激动不已。

427

附录A 信号完整性简介

A.1 开始之前

材料对电流产生的阻碍作用称为电阻 (resistance, R)，单位为欧姆 (ohm)。

欧姆 (用希腊字母 “ Ω ”) 取名于德国物理学家 Georg Simon Ohm，他于1827年定义了电压、电流和电阻之间的关系。

讨论这个话题之前，了解如下内容非常重要：信号完整性 (signal integrity, SI) 是一个极其复杂和极难理解的题目，如果不仔细，很快就会晕头转向。由于这个原因，本附录中有意只介绍那些较为重要的信号完整性概念，如果有兴趣学习更多这方面知识，可以阅读如下两本书：一本是信号完整性专家Eric Bogatin写的*Signal Integrity – Simplified*，另一本是Howard W. Johnson写的*High Speed Signal Propagation: Advanced Black Magic*。

信号完整性所涉及的方面非常广泛，包括信号通过一条导线时衰减的方式，以及信号从异常中断的导线末端有效地“反馈”回来的方式（就像在走廊里扔一个球碰到墙壁后反弹回来）。但是，这里我们将集中讨论隐藏在串扰之后纠集在一起的各种信号完整性效应。

串扰感应出的噪声或者脉冲干扰 (glitch) 以及延迟与从电路板级看到的硅芯片内部各种问题都有关系。由于这个原因，我们先介绍引起这些效应的根本原因，然后分别考虑这些效应在芯片级和电路板级上的表现。

429

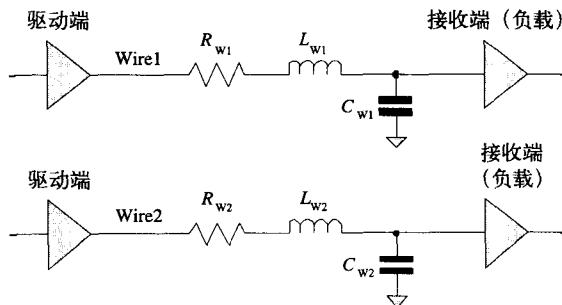
A.2 容感耦合 (串扰)

导体具有存储电荷的能力，这种属性称为电容 (capacitance,C)，单位是法拉 (Farads, F)。

法拉取名于英国科学家Michael Faraday，他于1821年发明了第一台电动机。

考虑两条导线Wire1和Wire2，每一条导线由一个逻辑门驱动，并驱动一个单负载。在一种理想并且稍微简单的情况下，两条导线都是直线，没有弯曲和间断，而且每条导线可以用一个串联电阻、一个串联电感和一个电容来表示，如图A-1所示。

这个例子中，电容 C_{w1} 和 C_{w2} 是以地为参考平面的，最简单的电容由两个金属片中间夹一层称为电介质的绝缘层组成。这就是说，如果上述两条导线非常靠近，那么从外界来看，这两条导线实际上构成了一个基本的电容。我们可以在电路图中增加一个符号M来表示两条导线之间交互电容，如图A-2所示。

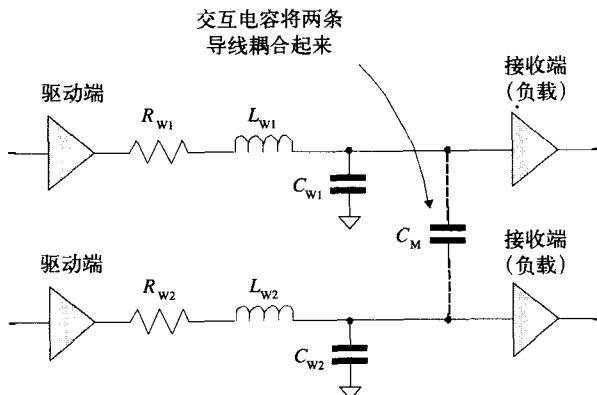


图A-1 理想情况下的两条导线

当其中一条导线上的信号正在发生变化时，导线之间的耦合电容将会在另一条导线上感应出电荷，结果可能造成噪声和延迟效应，下一节将要讨论这个问题。

430

如前面所说的，每一条导线都存在一定的电感，简单地说，电感就是导体所具有的一种属性，流过导体的电流发生变化时会在导体周围产生磁场；相反，导体周围的磁场发生任何变化也会在导体内感应出电流的变化。



图A-2 两条靠近的导线由一个交互电容耦合起来

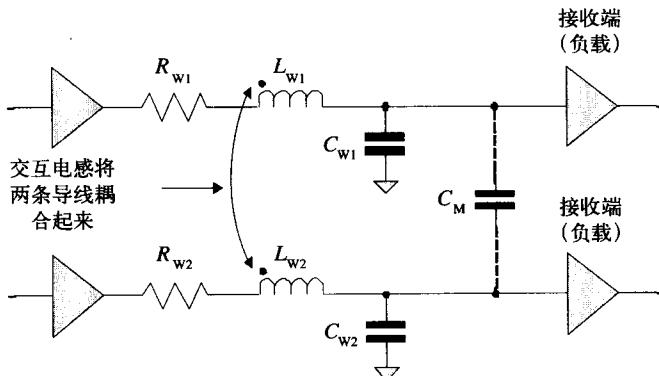
1831年，英国科学家Michael Faraday发现变化的电磁场可以在靠近的导体内感应出电流，后来这种效应被称为电感（inductance，L）。

电感的符号用大写字母L表示，是为了纪念俄国物理学家Heinrich Lenz，他于1833年发现了电磁感应现象中力、电压和电流之间的关系。

电感的单位是亨利（Henry，H）。亨利取名于美国科学家Joseph Henry，他和法拉第几乎同时独立地发现了电磁感应现象。

这就是说，其中一条导线上的信号发生变化时，导线中电流的变化和导线本身的电感会在导线周围建立一个磁场，这个磁场扩散出去时，它与附近导线的电感交互作用，可能就会引起噪声和延迟效应。这种交互电感可以通过在每个电感符号上增加一点来表示（如

图A-3所示)。



图A-3 被交互电感耦合的两条非常靠近的导线

A.3 芯片级效应

A.3.1 芯片级效应以阻容效应 (RC) 为主

早期的IC使用的是铝 (Al) 导线，电阻相对高。器件的特征尺寸随着新工艺的出现也在不断缩小，相应的铝导线的电阻开始增大到令人难以接受的程度。431

IC制造商长久以来就想使用铜 (Cu) 导线，因为铜是人们所知道的最好的导体之一，尤其适合于高频应用。但是，铜也有些缺点，就是很容易扩散到硅中，因此最终会使芯片失效。直到20世纪90年代后期，IBM通过设置阻挡层才解决了这个问题。

1925年以前，美国也接受铝的另一种拼法，即Aluminium。1925年以后，美国化学会公开决定在其出版物中使用aluminum这种拼法。

铜的使用历史超过一万年，是人类使用的最古老的金属。

地球上大多数生物都有血液，血液的红色是由铁基的血色素引起的。但是，还有些原始生物的血液是绿色的，那是因为血液中含有铜基的色素，称为cuproglobin。

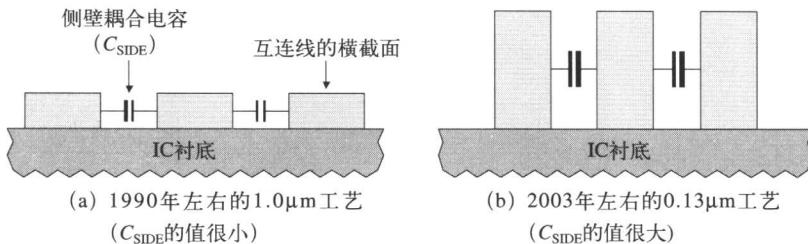
虽然铜的电阻比铝要低得多，但是由于IC上的信号线非常精细，所以它们的电阻仍然非常大。结果就造成信号沿IC中导线传播时的延迟效应倾向于以阻容 (RC) 特性为主。

这时候，电感 (L) 效应通常可以忽略不计，只在涉及电源网络时才给予考虑。电源网络一般使用更宽的导线因而电阻较低，所以电阻、电感和电容 (RLC) 特性都需要考虑。432

A.3.2 日趋严重的侧壁电容耦合 (sidewall capacitive coupling)

早期的IC制造工艺中，导线的高宽比非常小，也就是说它们的宽度比高度要大得多，如图A-4a所示。但是随着特征尺寸越来越小，导线的高宽比变得很大，即导线的高度远远

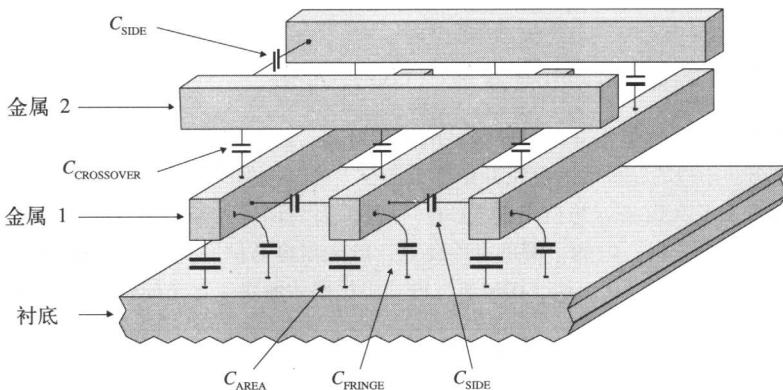
大于宽度，如图A-4b所示。



图A-4 随着特征尺寸的减小侧壁电容不断增大

(并不是按比例增加，图中只是说明了高宽比)

结果，相对于衬底电容 C_{AREA} （导线底部到衬底之间的电容）和 C_{FRINGE} （侧壁到衬底之间的电容）而言，相邻导线侧壁之间的耦合电容(C_{SIDE})却急剧增大。另外，现在的IC器件具有很高的集成密度，内部通常有八个以上的金属层，各个相邻的金属层之间会有严重的电容耦合现象，用 $C_{\text{Crossover}}$ 来表示，如图A-5所示。



图A-5 互连线之间的电容效应

这些因素加在一起使下面即将讨论的串扰噪声和时序效应变得极其复杂。

A.3.3 串扰感应脉冲干扰

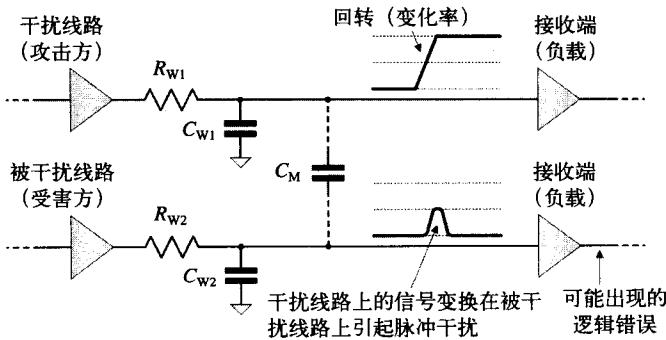
相邻导线上的信号发生逻辑跳变时，导线间的耦合电容将引起电荷的转移。根据信号的翻转速率（即上升跳变和下降跳变的速率）和交互串扰电容(C_M)的大小，可能发生严重的串扰感应脉冲干扰（图A-6所示）。

433

1975年，美国利用MOS工艺制造了基于6502的KIM-1微型计算机。

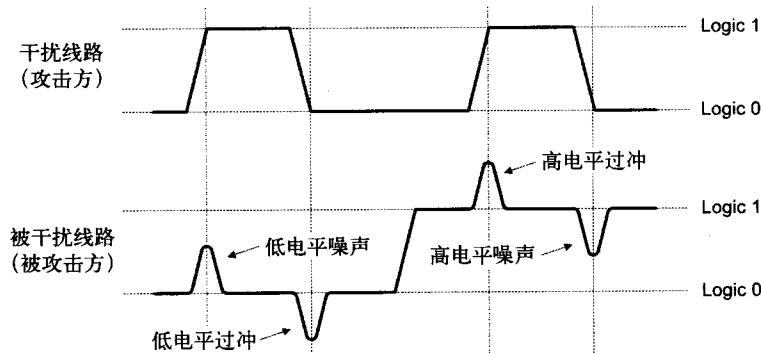
这个例子中，干扰线路（aggressor）上的一次逻辑跳变在相邻的被干扰线路（victim）上的接收负载输入端引起一个脉冲干扰。当然，这个例子只是说明了一种非常简单的情况，

实际上，每条导线可能由多个分段组成，这些分段又分布在多个金属层上，所以电阻 (R_{W1} 和 R_{W2})、电容 (C_{W1} 和 C_{W2}) 将由各个分段上的多个元素组成，同样，交互耦合串扰电容 (C_M) 也可能由多个元素组成。



图A-6 串扰感应脉冲干扰

图A-6中所示的脉冲干扰只不过是四种一般可能性之一，即干扰线路上的上升或下降跳变可能在被干扰线路上感应出逻辑0或逻辑1，如图A-7所示。



图A-7 串扰感应脉冲干扰的类型

如果被干扰线路上的低电平噪声和高电平噪声超过了接收端负载的输入开关阈值，就可能发生逻辑错误。有些情况下，这一错误可能表示随后要载入寄存器或者锁存器的值是错误的；而有些情况，这一错误可能引起锁存器意外地执行一次数据载入、置位或者复位。被干扰线路上的低电平下冲和高电平过冲可能造成其他的问题，因为它们会在多个晶体管中引起意外的电荷转移，形成逻辑门，可能会降低电路的性能。这些效应被统称为热电子效应 (hot electron effects)，虽然它们在现在的IC制造工艺中并不是主要的威胁，但是随着器件的几何尺寸逐渐进入到深亚微米 (deep-submicron, DSM) 和甚深亚微米 (ultra-deep-submicron, UDSM) 领域，它们会变得越来越严重。

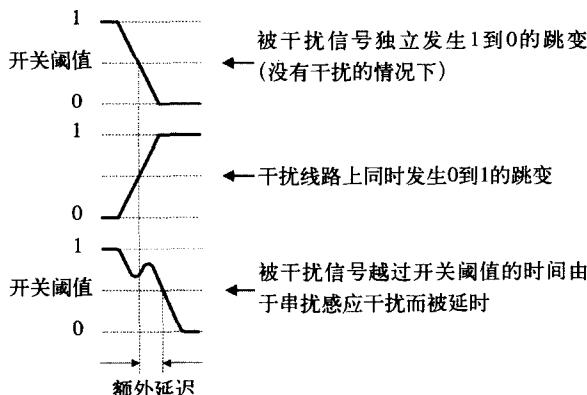
任何在 $0.5\mu\text{m}$ 以下的IC制造工艺都称为深亚微米工艺(deep submicron, DSM)。

进入甚深亚微米 (UDSM) 的具体时间点并不是很好定义，或者说，不同的人有不同的定义。

A.3.4 串扰感应延迟效应

435

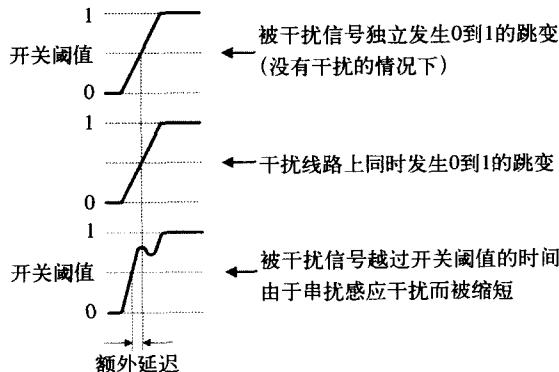
干扰线路和被干扰线路上同时出现逻辑转换时，情况变得更加复杂，例如，两者反方向跳变时，被干扰线路上的变化可能变慢，如图A-8所示。



图A-8 串扰感应信号延时

如果被干扰线路上的信号独立发生转换，它将用一段固定的时间越过接收器件的开关阈值（为了方便讨论，假定逻辑0和逻辑1之间的一半为开关阈值），但是，干扰线路上同时发生的逻辑转换将引起被干扰线路上出现脉冲干扰，这一干扰将使得被干扰线路上的信号在接收器件开关阈值以上维持一段时间，这样就可能导致信号不能满足下游器件的建立时间要求。

当被干扰线路上的信号转换被干扰线路上同时发生的同方向转换所补偿时，就会出现另一种情况，那就是被干扰线路上的信号变快了，如图A-9所示。

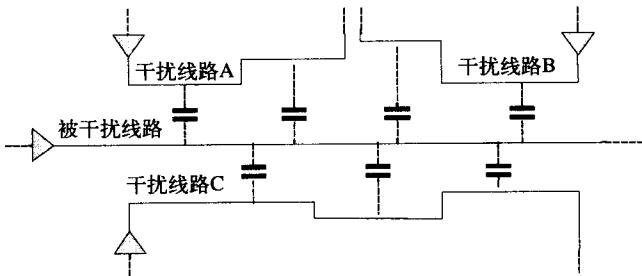


图A-9 串扰感应使信号加速

这种情况下，干扰方所引起的脉冲会导致被干扰方上的信号过早地越过负载或者接收器件的开关阈值。这可能引起信号不能满足下游器件的保持时间要求。

A.3.5 多干扰线路情况

实际上，上面提到的例子是过分简化的，在真实的设计中，每一条被干扰线路可能受到多个干扰线路的影响，如图A-10所示。



图A-10 多干扰线路的情况

要对现在的设计进行精确的分析，就必须单独分析和考虑每个干扰源的影响。

A.3.6 米勒效应 (Miller effect)

米勒效应指当电容器两个端子上的电压同时发生变化时，端子之间的有效电容容量也将发生变化。这一效应在芯片级显得特别重要。

电子电路中的总线是指执行同样的功能，传递类似数据的一组信号。

例如，当考虑总线中间的一个信号时，米勒效应的实际意义就很明显了。如果总线上被选定的信号周围有一个或多个信号也出现同样极性（方向）的电平转换，那么和这个信号相关的电容容量就会变小，而且其传播延迟也将减小（这不包含前面介绍的串扰感应的延迟效应）。

相比而言，如果总线中选定的信号与周围的信号出现了相反的跳变，那么与这个信号相关的电容容量就会变大，其传播延迟也会增大。

以刚刚提到的芯片级效应开始，是因为这样可以为IC设计工程师提供熟悉的开始点。但实际上，片内的信号完整性效应（不包括封装问题）很少被使用FPGA的工程师关注到，因为这些效应由FPGA厂商在幕后处理了。相比之下，当把一个FPGA集成到电路板环境中时，板级信号完整性效应就会受到极大的关注。

A.4 板级效应

A.4.1 板级效应以感容效应 (LC) 为主

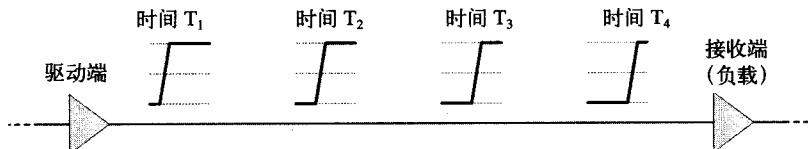
PCB上的铜导线电阻和耦合效应比起来几乎可以忽略不计，这是因为板级导线一般有

125μm宽、18μm厚，其横截面积比芯片级导线要大得多，导体的横截面积越大，电阻就越小。与电阻比起来，电感和电容耦合效应就显得严重得多，所以电路板上的信号线以感容耦合为主要考虑对象。

A.4.2 换一种思考方式

PCB的另一个名称是PWB，即印制线路板。使用最普遍的电路板材是FR4，FR代表flame retardant，即阻燃。

现在的高速、高性能PCB，导线几乎无一例外地扮演着传输线的角色，这就意味着我们不得不将一个信号边缘看作是一个移动的波形，沿着时间轴向前传播，如图A-11所示。

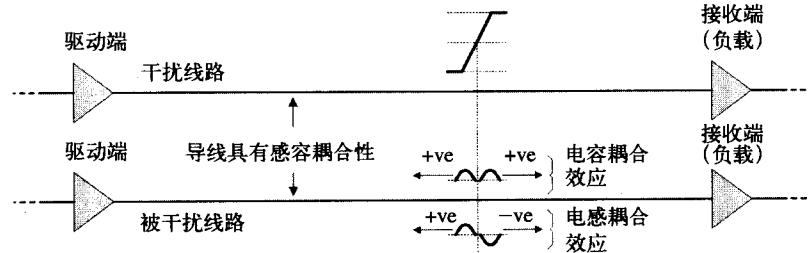


图A-11 沿时间轴移动的信号边缘

在这种传输线中，信号的传播延迟要比信号本身的转换时间长得多，只有一个地方可能出现电容或者电感耦合情况，那就是移动边缘的当前位置。这就是说，我们必须将导线分成一系列RLC片段来考虑（为了简单，图中并没有表示出来）。

A.4.3 电容和电感耦合效应

考虑两条非常靠近的板级导线时，情况开始变得有趣起来。假如我们正在寻找一个在干扰线路上传播的移动边缘，干扰线路与临近的被干扰线路具有电感和电容耦合关系，如图A-12所示。



图A-12 电容和电感耦合效应

在电容耦合情况下，干扰线路上的移动边缘将在正反两个方向上在被干扰线上感应出正电流脉冲；而在电感耦合情况下，干扰线路上的移动边缘将在正方向上在被干扰线上感应出负电流脉冲，而在反方向上感应出正电流脉冲。

这也就是说，对于近端噪声（从导线驱动端看到的噪声），电容和电感耦合效应会相互

增强；但是对于远端噪声（从导线负载端看到的噪声），它们却相互抵消。所以人们希望的最佳状况是电容和电感耦合效应能在数值上比较接近，以便它们在远端能相互抵消。但不幸的是，这种情况只有在信号周围的电介质层为均匀同质时才会发生，例如等高带状层。而在实际情况中，信号周围的电介质通常都是异质的，例如表面的导线，其上方是空气，而下方是FR4，这种情况下，电感耦合效应没有变化，但是电容耦合效应减小了，这就相对提高了电感耦合效应的量，从而使接收端上产生远端噪声。在一般的电路板上，电感性的噪声通常是电容性噪声的二到四倍。

如果出现任何情况使返回路径产生问题，那么电感耦合效应将急剧增大到电容耦合效应的十到三十倍，这时，串扰主要由电感耦合效应决定，由此带来的噪声称为开关噪声（switching noise）。多条信号路径共享同一条返回路径时，穿过返回（接地）连接的开关噪声称为接地反弹（ground bounce）。

440

A.4.4 反米勒效应

在讨论芯片级效应时，介绍了米勒效应这一概念，其意思是，如果与某个信号非常接近的一个或者多个信号和该信号朝相同的极性（方向）变化，那么此信号相关的电容容量将会减小，而其传播延迟也会减小。

但是，正如前面提到的，芯片级信号的传播延迟以电阻电容（RC）效应为主，而板级信号的传播延迟则以电感电容（LC）效应为主。这就是说，如果一个或者多个板级信号与非常靠近的信号发生同向的变化，那么该信号相关的电感会增大。在一个异质的电介质层中，电感耦合效应相对于电容耦合效应占有更大的比例，而且信号线上的电感越大，引起的传播延迟也越大。

相对而言，如果板级信号与选定的信号出现相反方向的变化，那么该信号相关的电感将减小，而且其传播延迟也将降低。

A.4.5 传输线效应

当然，除了上面提到的一些效应外，还有传输线效应，另外还要考虑末端电阻，例如，在输出端使用串联电阻，而在输入端使用并联电阻，但这类问题早已经在标准的教科书里讨论过，这里跳过不谈。

441

A.5 你也可以做一些事情使问题更容易

然而，从本质上讲，将一个FPGA器件连接到电路板上产生的信号完整性问题中有70%到80%与电路板没有关系，而是与FPGA的封装有关。

事实上，芯片的封装应该尽可能多地设置电源和接地焊盘对，这些焊盘对应该均匀地分布在封装的底层，以便为I/O焊盘提供充足的附近的返回路径。而实际上，电源和接地焊盘往往是团簇式地聚在一起，让I/O焊盘组自己尽最大可能去利用这些返回路径。

制定规则会使问题变得容易，可以选择使用差分输出对作I/O，尤其是在总线和高速互

连线中，这样做比较合适，当然也会令所用到的引脚数量加倍，但是如果引脚有富余时，还是非常值得的。

另一个需要考虑的地方和FPGA内部提供的可编程末端电阻有关，它们的使用是可选的，因为既可以使用板级的分立元件，也可以在必要的时候使用这些内部电阻。这些内部末端电阻主要是用在板级布线空间比较狭小的情况下，但是它们也存在信号完整性的问题。从过去的经验来看，任何信号，只要上升/下降时间在500ps或者以下，使用外部末端电阻就会引起信号的不连续，所以这种情况下应该总是使用片内的末端电阻。

附录B 深亚微米延迟效应

B.1 引言

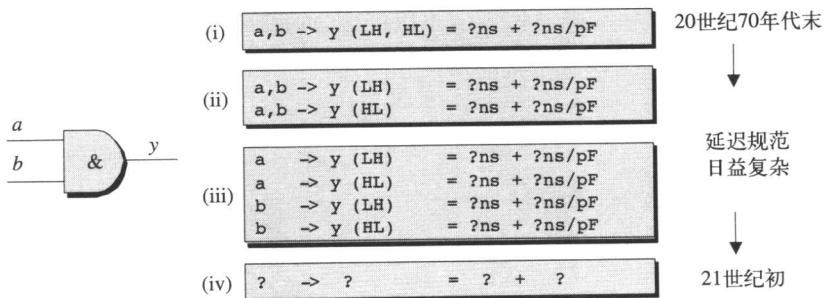
本附录的内容摘自作者的另一本书*Designus Maximus Unleashed*, 得到了出版商的允许。

设计ASIC和ASSP时, 需要考虑的时序效应是极其复杂的, 随着每一种新的制造工艺的出现, 这些效应也变得越来越复杂。某些时候(具体的时间不是很好定义, 或者不同的人有不同的定义, 但是在这里姑且认为是 $0.5\mu\text{m}$ 工艺出现的时候), 我们开始进入深亚微米(deep-submicron, DSM) 延迟效应的领域内。

当然, 使用FPGA的一个最大的优点是, 设计生产这些器件的厂商处理大部分深亚微米延迟效应带来的问题, 这些问题对终端用户(设计工程师)来说也就是透明的。有了这样的基础, 其实不需要在这里讨论深亚微米时序问题, 但另一方面, 这类问题又会经常听到, 而我从来没有遇到对这些效应的浅显易懂的介绍, 正是因为这个原因, 提出以下的概要说明以飨读者。

B.2 延迟规范的发展

大约在20世纪70年代末和80年代初, ASIC设计工程师的生活要比现在简单, 针对早期制造工艺(几个微米)的延迟规范还没有什么发展。例如一个简单的双输入与门, 其数据手册上标明的输入到输出延迟对所有的输入端都是一样的, 而且输出的上升和下降转换延迟也是一样(如图B-1所示)。



图B-1 随着时间的推移, 延迟规范变得越来越复杂

但是随着器件几何尺寸的缩小, 延迟规范变得越来越复杂。接下来的发展就是区分了输出转换的上升和下降变化(如图B-1ii所示), 然后又为每一条输入到输出的路径设置了不同的延迟(如图B-1iii所示)。

444

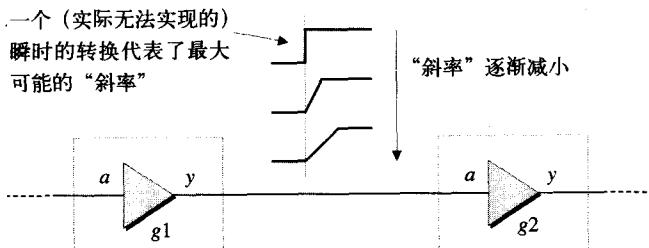
所有这些早期的延迟一般都是用“?ns +?ns/pF”格式指定的。第一部分（?ns）表示逻辑门本身的固有延迟，单位是纳秒（ns）^①；第二部分是由电容性负载^②引起的附加延迟，单位是纳秒每皮法（ns/pF）。我们将看到，这种形式的规范无法处理深亚微米工艺中的延迟特性，也不能用在RLC互连延迟的领域，下文将会讨论。

B.3 一些定义

在深入讨论亚微米延迟之前，首先有必要介绍一下各种定义与术语。

B.3.1 信号斜率（Signal slopes）

一个信号的斜率是指该信号从逻辑0到逻辑1，或者相反变化时的速率。瞬时的逻辑变换在真实的世界里是不可能实现的，因此一般用一个最大可能的斜率值来表示，如图B-2所示。



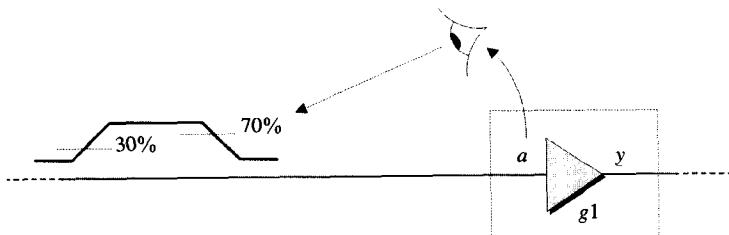
图B-2 信号的斜率是逻辑转换所用的时间

信号的斜率是许多因素的函数，这些因素包括驱动门的输出特性、互连导线的特性以及负载门的输入特性。

B.3.2 输入开关阈值

445

所谓输入开关阈值，是指从负载门输入端看到的输入信号转换的开始点，也就是说，输入信号越过阈值电平的开始点，或者也可以说是下游逻辑门注意到输入有变化的时刻。输入开关阈值通常用逻辑0和逻辑1之间（电压）差值的百分比来表示，而且每一个输入端对应的上升和下降变换的开关阈值可能是不同的，如图B-3所示。



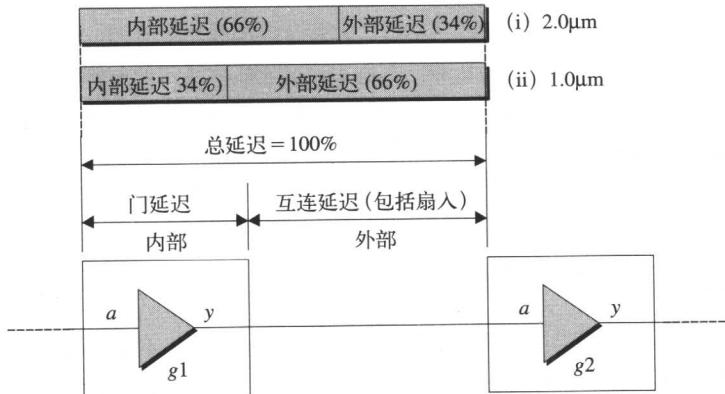
图B-3 上升和下降变换的开关阈值可能不同

^① 现在的器件速度更快，因此其延迟是用皮秒（ps）衡量的。

^② 电容的基本单位是法拉（Farad），取名于英国科学家Michael Faraday，他于1821年制造了第一台电动机。

B.3.3 内部延迟与外部延迟

内部延迟是指逻辑功能内部的延迟效应，而外部延迟是指与芯片互连相关的延迟，如图B-4所示。



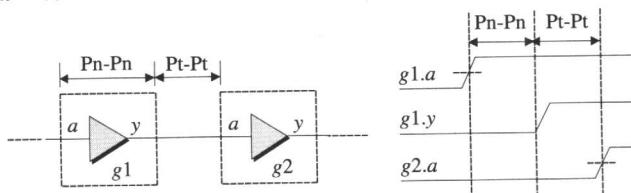
图B-4 内部延迟与外部延迟

在早期的几微米工艺中，内部延迟要远远大于外部延迟。例如，用2.0 μm 工艺制造的器件，其内部延迟一般约占总延迟的三分之二，如图B-4a所示。随着器件几何尺寸的减小，外部延迟变得越来越突出，到1.0 μm 工艺的器件出现时，内部延迟和外部延迟的相对关系发生了倒置，如图B-4b所示。

这种趋势还将继续下去，因为互连线的几何尺寸并没有按晶体管和逻辑门的缩小速度而减小，今天的深亚微米工艺中，外部延迟占到了总路径延迟的80%以上。

B.3.4 Pn-Pn和Pt-Pt延迟

在很大程度上来说，引脚到引脚 (pin-to-pin, Pn-Pn) 和点到点 (point-to-point, Pt-Pt) 延迟分别是内部延迟和外部延迟的较为现代的说法。Pn-Pn延迟是指从逻辑门的输入端产生变化至该逻辑门的输出端响应到这个变化的延迟；而Pt-Pt延迟是指从一个驱动门的输出端到负载门的输入端的延迟，如图B-5所示。^①



图B-5 Pn-Pn延迟与Pt-Pt延迟

^① 应该注意到，电路板布线工程师不会太关注器件内部发生的事情，他们通常将这些器件看成“黑盒子”。这里之所以提到这一点，是因为板级工程师可能使用“pin-to-pin”表示电路板级的线路延迟。

更精确的描述为，Pn-Pn延迟是指从逻辑门输入端的信号达到输入开关阈值的时刻开始，到逻辑门的输出端开始响应时为止的时间；Pt-Pt延迟是指从驱动门的输出端开始变化的时刻至负载门响应到这一变化并达到其输入开关阈值时的一段时间，这很难理解。

我们将逻辑门输出开始响应的时刻看作Pn-Pn延迟的结束点和Pt-Pt延迟的开始点，这里之所以强调，有许多的原因。过去，这些延迟都是从输出到达逻辑0和逻辑1之间的一半时开始算起的，这可以被接受，因为负载门都假定以50%作为输入开关阈值，但是，考虑一下输出端的上升变换，并假定负载门对上升变换的输入开关阈值为30%，如果计算延迟时是从输出达到50%算起，那么负载门在我们认为输出改变之前就“看到”这一变换是完全有可能的，而且，在混合信号（模拟与数字）仿真时，从数字域中的逻辑门输出变换传递到模拟域中，唯一有意义的时间就是逻辑门输出开始变换的时刻。

B.3.5 状态相关与斜率相关

逻辑门的一个输入端所具有的任何属性（包括Pn-Pn延迟），只要是这个逻辑门其他输入端逻辑值的函数，就称为状态相关（state dependent）；同样，逻辑门的某个输入端的属性（包括Pn-Pn延迟）是该输入端信号斜率的函数时，就称为斜率相关（slope dependent）。现在来看，状态相关和斜率相关这些定义还没有太明确的意义，但是在读完本章后，一切都会很快展现出来的。

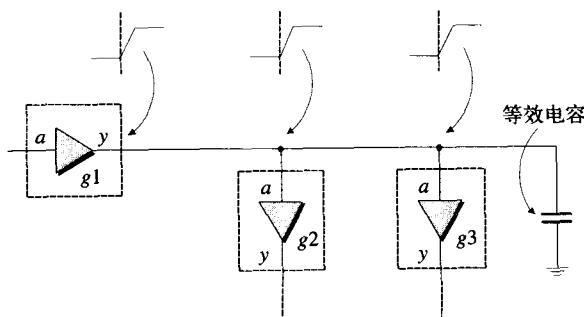
448

B.4 可选择的互连模型

随着硅芯片中结构的几何尺寸越来越小，器件内部集成的逻辑门数量越来越多，互连延迟也变得更加严重，而且为了精确地描述相关的延迟效应还需要越来越复杂的算法。互连模型主要有下面介绍的几种。

B.4.1 集总负载模型

前面已经提到过，在早期的几微米工艺中，Pn-Pn门延迟要远远大于Pt-Pt互连延迟。另外，信号的上升时间和下降时间一般要比信号在互连线中传输的时间还要长。这些情况下，使用集总负载模型（The lumped-load model）来表示互连通常就足够了，如图B-6所示。

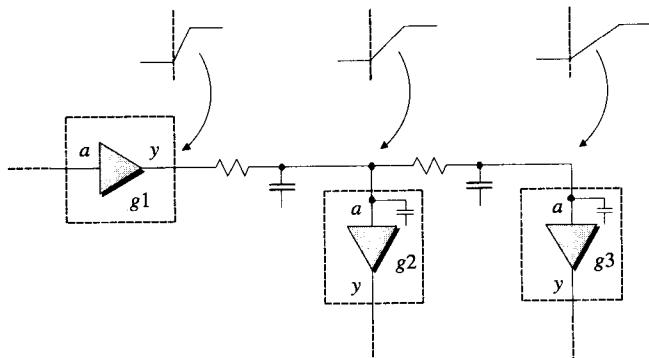


图B-6 集总负载互连模型

这其中的思想是，将所有的导线电容和负载门输入电容集合在一起形成一个单一的等效电容，这个电容乘以驱动门的驱动能力（单位是“ns/pF”）就是Pt-Pt延迟。集总负载模型具有如下特征：导线上所有节点是在同一时刻并以同一斜率开始变化。这一模型也可称为纯RC模型。

B.4.2 分布式RC模型

到了20世纪80年代中期，不断减小的器件几何尺寸要求使用比集总负载模型更精确的互连线描述方法，这样，分布式RC模型就应运而生（这里的R和C分别代表电阻和电容，如图B-7所示）。

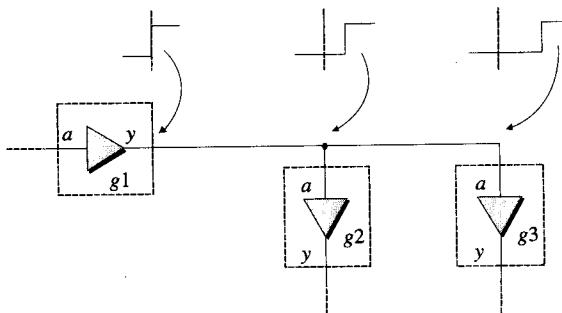


图B-7 分布式RC互连模型

在分布式RC模型中，每一段导线都被看成是一个RC网络。分布式RC模型具有如下特征：导线上的所有节点在同一时刻开始变化，但是变化斜率不同，换句话说就是，信号在沿着导线传播的过程中，其边缘逐渐会变得平坦。

B.4.3 纯LC模型

在电路板级，高速互连开始成为传输线的特征，这种纯LC模型（这里的L和C分别代表电感和电容）可以描述沿着导线传播的像波峰一样的急剧变化，如图B-8所示。

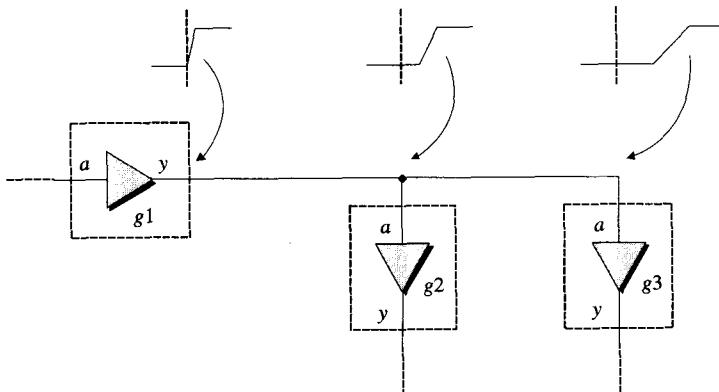


图B-8 纯LC互连模型

纯粹的传输线效应在硅芯片内部是不会出现的，但是一些较大的亚微米器件确实开始出现这类延迟效应的某些特征，后面将会讨论。

B.4.4 RLC模型

具有深亚微米几何尺寸的大型器件中，信号的高速度，再加上长导线会导致互连线产生一些传输线类型的延迟效应。但是，片内互连线的阻抗本性不支持纯粹的LC效应；取而代之的是，这些导线可能会呈现出RLC延迟效应，如图B-9所示。



图B-9 RLC互连模型

RLC模型具有如下特征：既有互连线中LC元件产生的离散波峰效应，又结合了RC元

451

件引起的信号边缘效应。

B.5 深亚微米延迟效应

B.5.1 针对路径Pn-Pn延迟

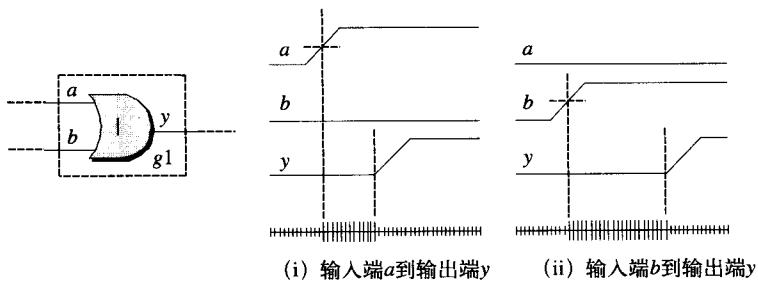
每一条输入到输出的路径通常都有各自的Pn-Pn延迟。例如，一个二输入或门，输入端 a 上的一个变化引起输出端 y 的变化（如图B-10a所示），其延迟与输入端 b 的变化引起输出端 y 产生同样变化的延迟是不同的（如图B-10b所示）。

452

同样，输出端的上升变化和下降变化也都有各自的Pn-Pn延迟。仍以上面说的或门为例，输入端 a 的变化引起输出端 y 产生上升变化的延迟与输入端 a 引起输出端 y 产生下降变化的延迟是不同的。

需要注意的是，这个例子中假定输入开关阈值为50%，而且，Pn-Pn延迟是从输入端的信号达到输入开关阈值的时刻算起，到输出端开始响应为止。

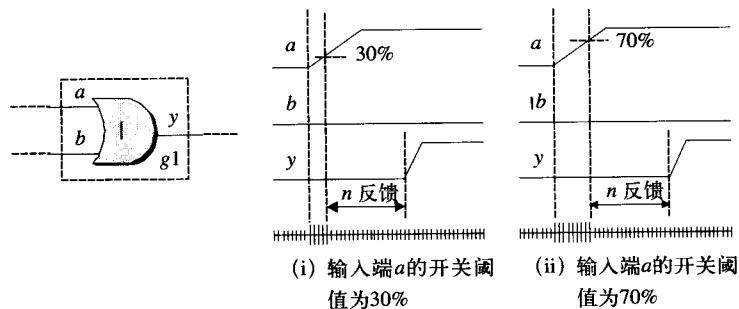
针对路径和针对变换的Pn-Pn延迟不仅仅局限于深亚微米工艺中，它们的出现其实不足为怪，不过这里提出来是为后面讨论的问题做准备。



图B-10 针对路径的Pn-Pn延迟

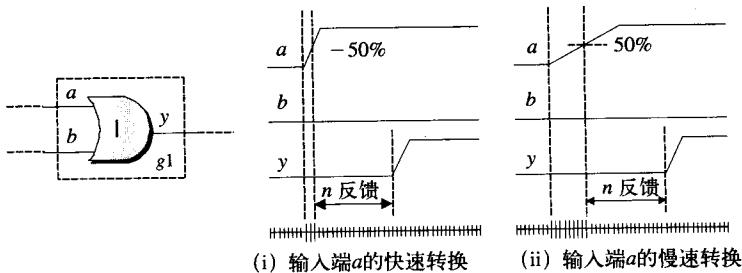
B.5.2 阈值相关的Pn-Pn延迟

Pn-Pn延迟取决于输入端的开关阈值，至少，门延迟直到其输入端信号达到开关阈值时才真正开始。例如，如果输入端a的上升变化的输入开关阈值为30%，如图B-11a所示，那么逻辑门开始响应的时刻肯定要比输入开关阈值为70%时更早，如图B-11b所示。



图B-11 阈值相关的Pn-Pn延迟

另外，输入信号的斜率也会影响信号达到输入开关阈值的时间。举一个简单的例子，假设输入端a的开关阈值为50%，如果输入端a的信号是一个变化很陡峭的信号（如图B-12a所示），那么相比一个变化比较缓慢的信号而言，逻辑门检测到信号的变化会更早，如图B-12b所示。



图B-12 输入信号的斜率影响输入端检测到信号的时间

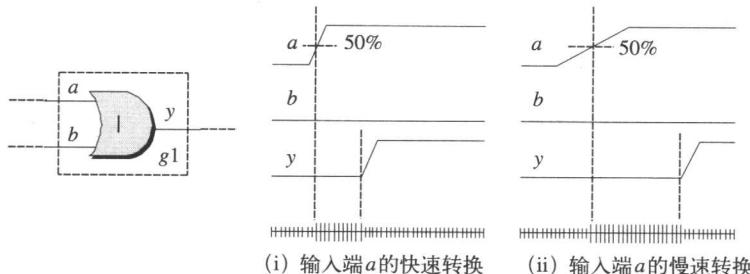
尽管这影响了Pn-Pn延迟开始的时间点，但是却与下一节介绍的斜率相关的Pn-Pn延迟完全不同。

B.5.3 斜率相关的Pn-Pn延迟

事实上，前面的例子有些简单，因为例子中的两个Pn-Pn延迟是相同的，与输入信号的斜率无关。有些CAD工具厂商将前面提到的情况称为“斜率相关”，但这是对这个术语的误用法，实际上，许多深亚微米工艺中的延迟效应可能确实是斜率相关的，也就是说这些

Pn-Pn延迟可能与输入信号的斜率直接有关。

我们考虑一下，当输入端的信号达到输入开关阈值的时刻会发生什么。这一点的Pn-Pn延迟是输入信号变化率的函数，例如，输入信号变化率较快时，相应的Pn-Pn延迟就很短，如图B-13a所示；而信号变化率比较慢时，Pn-Pn延迟则很长，如图B-13b所示。



图B-13 输入信号的斜率影响输入端检测到信号的时间

实际上，图B-13中所示的效应，即信号变化斜率越小，引起的Pn-Pn延迟越大，这只是一种情况而已，这种特殊的情况适用于某些逻辑门或者制造工艺，其中主要的延迟效应是，构成这些逻辑门的晶体管的开关速度与输入端的电荷变化率有直接的关系。相比之下，有些工艺中，信号变化斜率越小，引起的Pn-Pn延迟也越小（从输入开关阈值算起），造成这种情况的原因是，足够长的信号斜率使得内部的晶体管预充电到接近开关阈值的某一点。这样，当输入信号实际到达输入开关阈值时，逻辑门其实已经要转换了，因此看起来逻辑门的转换速度比输入端为陡峭信号时还要快。

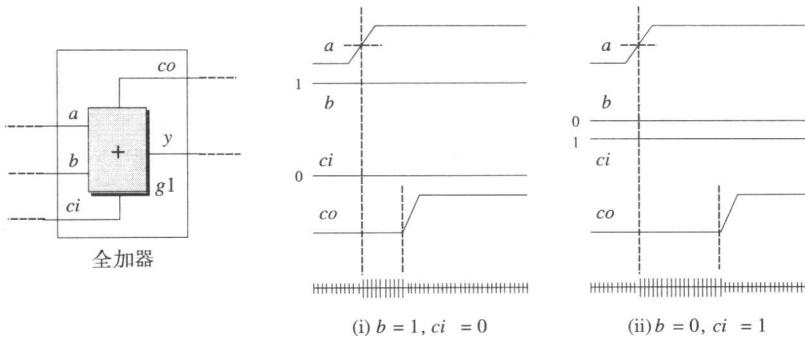
实际应用过程中，这两种效应可能会同时出现。这种情况下，在输入端施加一个变化陡峭的信号会引起某个确定的Pn-Pn延迟，随着信号逐渐倾斜，Pn-Pn延迟也逐渐增大。但是，到了某一点时，输入端信号进一步倾斜反而会使得Pn-Pn延迟减小，到了一定的程度后，可能比原来的陡峭信号引起的Pn-Pn延迟都要小^①。

B.5.4 状态相关的Pn-Pn延迟

除了与信号斜率相关外，Pn-Pn延迟常常还和状态有关，也就是说，这些延迟还依赖于其他输入端的逻辑值，如图B-14所示。

① 因此，有些人可能会认为，电子学很无聊。

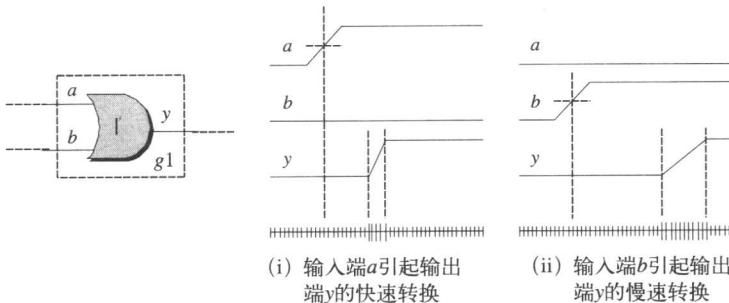
这个例子说明了两种情况，都是输入端 a 上的信号引起输出端 co 发生同样的变化（这里是指逻辑值）。但是，即使假设输入端 a 上的信号在两种情况下具有相同的斜率和开关阈值，由于输入端 b 和输入端 ci 的逻辑值不同，两种情况下的Pn-Pn延迟也可能是不同的。



图B-14 状态相关的Pn-Pn延迟

B.5.5 路径相关的驱动能力

从这里开始，生活才真正变得有意思起来。直到现在，我们还只是考虑了影响逻辑门Pn-Pn延迟的效应，其实其中许多效应也会影响逻辑门输出端的驱动能力，例如，一个逻辑门的驱动能力可能与路径有关，如图B-15所示。



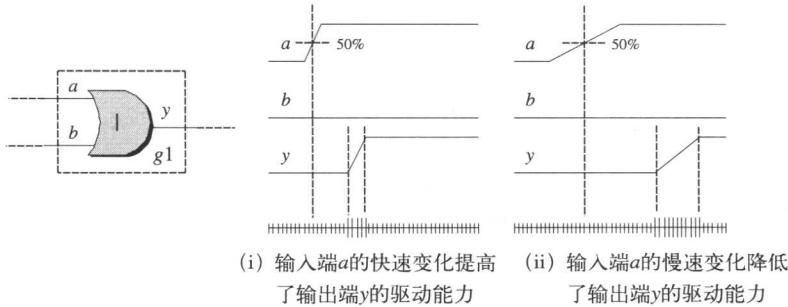
图B-15 路径相关的驱动能力

本例中，除了输入端 a 和输入端 b 的Pn-Pn延迟不同以外，逻辑门的驱动能力（输出信号的斜率）还取决于哪个输入端引起了输出端发生转换。这种现象原来只存在于MOS工艺中，一般不会在双极工艺中出现，例如TTL工艺。但是，随着深亚微米时代的到来，许多更复杂、更深奥的延迟效应开始在多种工艺中显现出来，而与传统的各种工艺边界没有什么关系。

B.5.6 斜率相关的驱动能力

除了依赖于哪个输入引起了输出的转换（前面讨论过）外，逻辑门的驱动能力也可能

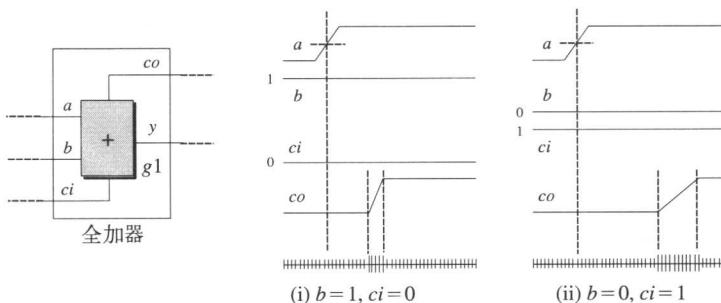
与输入端信号的斜率有关。例如，输入端a的一个快速变化可能引起输出端的一个快速变化，如图B-16a所示；而同一个输入端若有一个慢速变化的信号可能会影响逻辑门的驱动能力，从而引起输出信号的斜率变小，如图B-16b所示。



图B-16 斜率相关的驱动能力

B.5.7 状态相关的驱动能力

另一个可能影响输出端驱动能力的因素是，输入端的逻辑值与实际引起输出端发生逻辑转换的逻辑值不同。这一效应称为状态相关的驱动能力，如图B-17所示。
458



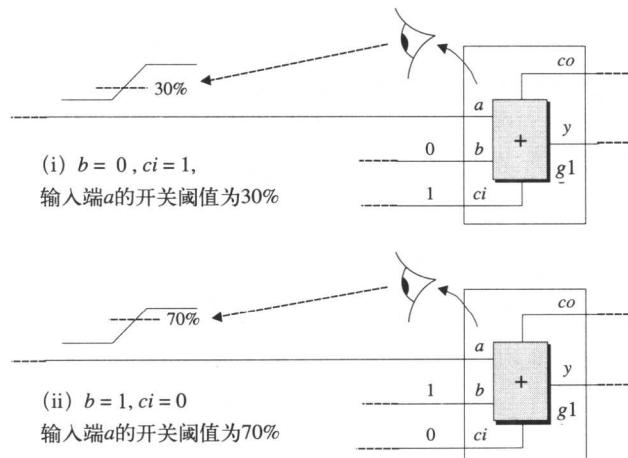
图B-17 状态相关的驱动能力

图B-17说明了两种情况，都是输入端a上的信号引起输出端co发生同样的变化（这里是指逻辑值）。但是，即使假设输入端a上的信号在两种情况下具有相同的斜率和开关阈值，由于输入端b和输入端ci的逻辑值不同，两种情况下逻辑门的驱动能力（即输出信号的斜率）也可能不同。

B.5.8 状态相关的开关阈值

你肯定已经注意到，前面关于状态相关驱动能力的讨论中有这样一句话“假设输入端a上的信号在两种情况下具有相同的斜率和开关阈值”，如果这句话在你的大脑中引起些许警惕，那么至少说明这些讨论正在锻炼你在深亚微米设计领域内的生存能力。

但是，事情出现了奇怪的转折，因为输入端的开关阈值可能还依赖于电路状态，也就是说，这个开关阈值可能与其他输入端的逻辑值有关，如图B-18所示。

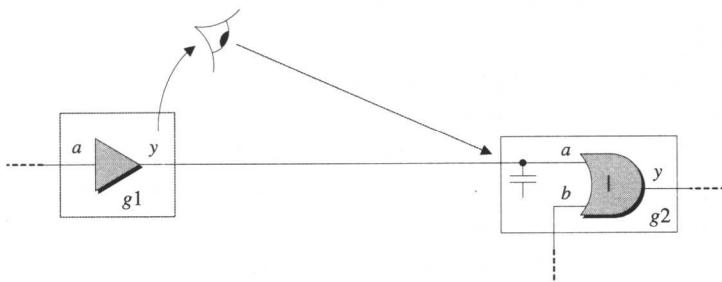


图B-18 状态相关的输入开关阈值

本例中，输入端a的开关阈值（即该输入端检测到输入逻辑转换的开始点）依赖于输入端b和输入端ci上的逻辑值。

B.5.9 状态相关的末端寄生效应

除了输入开关阈值和状态有关外，其他一些与输入有关的更深层次的特性（例如寄生效应）也可能与其他输入端的逻辑值有依赖关系。例如，一个二输入或门，如图B-19所示。



图B-19 状态相关的末端寄生效应

输入端g2.a的末端电容（从驱动端输出g1.y看过去）可能与另一个输入端g2.b上的逻辑值有依赖关系。如果g2.b为逻辑0，那么输入端g2.a上的逻辑转换将引起或门的输出发生跳变，这时，从g1.y（驱动g2.a的门输出端）看过去的电容就相当大；但是，如果g2.b是逻辑1，那么输入端g2.a的变化就不会引起或门的输出发生转换，这时，从g1.y端看到的电容就很小。

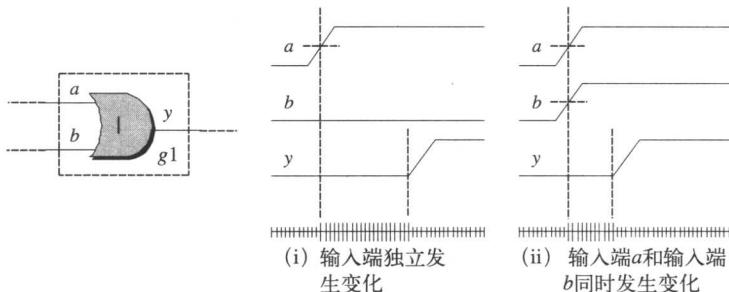
你或许会有疑问：“在或门的输出不发生跳变的情况下，难道还要关心输入端a的寄生

460 电容是否不同吗？就不能在或门将要发生转换时设定电容的值吗？实际上，如果g1.y只驱动一个g2.a，这完全没有问题，但是，如果将电路修改一下，使g1.y驱动两个或更多的负载门，问题就出来了。

这种特殊的效应首先出现在ECL（即发射极耦合逻辑）技术中。其实，早在20世纪80年代后期，在ECL门阵列技术中，一个负载门的末端电容（驱动门看到的）由于这种状态关联性而出现的改变将接近100%。但是，这种效应不仅仅局限于ECL技术中，再强调一次，随着我们在深亚微米领域研究的深入，延迟效应开始在多种工艺中显现出来，而与传统的各种工艺边界没有什么关系。

B.5.10 多输入转换的Pn-Pn延迟效应

此前，我们仅仅考虑了单个输入端的信号引起输出发生变化的情况，其实，多个输入发生转换时，情况要复杂得多。例如，还以一个二输入或门来说明，如图B-20所示。



图B-20 多输入转换的Pn-Pn延迟效应

为了简单起见，这里假定输入端a和b是完全对称的，也就是说，两者的输入开关阈值和Pn-Pn延迟完全相同。

首先，考虑这种情况，单个输入端（如输入端a）的变化引起输出的变化，如图B-20a所示，这时的Pn-Pn延迟一般和这个单元的数据手册中描述的延迟一致。但是，如果两个输入端同时变化，那么Pn-Pn延迟可能会减小到数据手册中的数值的一半左右。

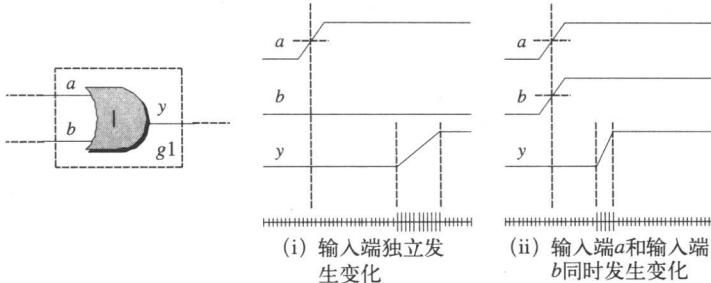
这两个例子（单个输入端独立变化与多个输入端同时变化）为我们提供了最极端的两种延迟结果。然而，仍有必要考虑输入端不同时发生变化，但变化却发生得比较靠近的情况。例如，图B-20中的或门，假设两个输入端初始值都是逻辑0，现在，如果输入端a发生了上升变化，那么其Pn-Pn延迟与数据手册中一致，但是这一延迟还没有完全结束时，另一个输入端b也发生了一个上升变化，结果是，实际的Pn-Pn延迟处于两种最极端延迟之间。

B.5.11 多输入转换的驱动能力效应

除了改变Pn-Pn延迟外，多输入转换现象也可能影响逻辑门的驱动能力，而且会进一步影响输出信号的变化斜率，如图B-21所示。

所有这些多输入转换效应能够通过简单的线性近似进行估计。但不幸的是，今天的验

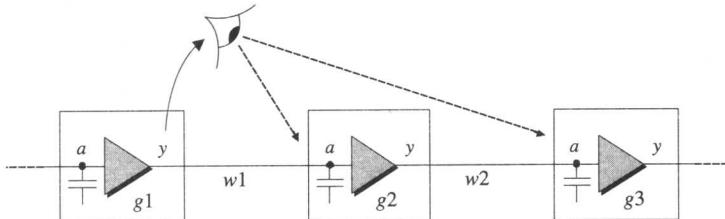
证工具，例如静态时序分析（STA）和数字逻辑仿真，都不能很好地处理这种运算。



图B-21 多输入转换的驱动能力效应

B.5.12 反射寄生效应

在不久前的工艺中，假设寄生效应只在有限的范围内存在，并且一般只影响最接近的逻辑门。这种假设是十分安全的，例如，图B-22中所示的三个逻辑门。



图B-22 反射寄生效应

传统上，假定逻辑门 g_2 为导线 w_2 和逻辑门 g_3 缓冲了 g_1 的输出是安全的。这样，输出 $g_1.y$ 将只能看到如下这些寄生效应，即导线 w_1 的电容和逻辑门的输入端 $g_2.a$ 。

在深亚微米领域内，这些假设就不再有效了。回到图B-22中所示的三个逻辑门，导线 w_2 和逻辑门输入端 $g_3.a$ 的一部分寄生效应完全有可能穿过逻辑门 g_2 反射回去，从而对输出端 $g_1.y$ 产生影响。另外，如果逻辑门 g_2 是一个多输入门，例如二输入异或门，那么这些穿过 g_2 门的部分寄生效应将是状态相关的，也就是说，它们可能会根据 g_2 其他输入端的逻辑值而产生变化。

写作本书时，反射寄生效应还保持着相当低阶的效应，但是，随着我们继续朝更新的工艺节点前进，这些效应将很有可能变得更加严重。

B.6 总结

本章介绍的大多数延迟效应都是存在的，即使在几微米工艺中也是如此，但是这些效应中有许多都是三阶或者四阶的，因此相当微弱，几乎可以忽略不计。随着器件的几何尺

寸由 $0.5\mu\text{m}$ 进入到 $0.35\mu\text{m}$ 工艺中，这些效应中有些就开始变成二阶甚至一阶的，而且，在工作于低电压下的新的工艺中，这些效应的重要性还在继续增加。

不幸的是，许多验证工具并不能跟上硅工艺的发展水平。除非这些工具能被增强到全面考虑深亚微米效应，否则，设计人员还得继续使用限定性的设计规则来保证设计的正确性。因此，设计工程师会发现，要完全了解即将到来的新的、令人激动的工艺进展的潜力几乎是不可能的。

附录C 线性移位寄存器

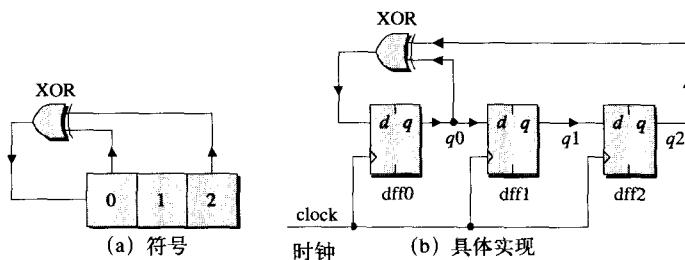
C.1 环形蛇 (Ouroboros)

本附录的内容摘自作者写的一本书*Bebop to the Boolean Boogie* (一本电子学领域非传统的指南, 第二版), 书号为ISBN 0-7506-7543-8, 摘录得到了出版商的许可。

环形蛇 (Ouroboros) 是一条咬着自己尾巴的大毒蛇的图案, 这条蛇形成了一个环形, 世界上有很多古老文化用这个图案来描述不朽和重生^①。电子学领域内的“环绕蛇”就是线性移位寄存器 (linear feedback shift register, LFSR), 标准移位寄存器的输出经过特别的组合并反馈到输入, 这样就形成了无限的循环, 产生一系列输出。

C.2 多点到一点的实现方式

线性移位寄存器的构造相当简单, 但是在很多应用领域内都有用。一种比较普遍的线性移位寄存器的构造形式是使用一个简单的移位寄存器, 并将寄存器链中的两个或多个抽头 (tap) 反馈回输入端 (如图C-1所示)。



图C-1 使用异或门反馈路径构成线性移位寄存器

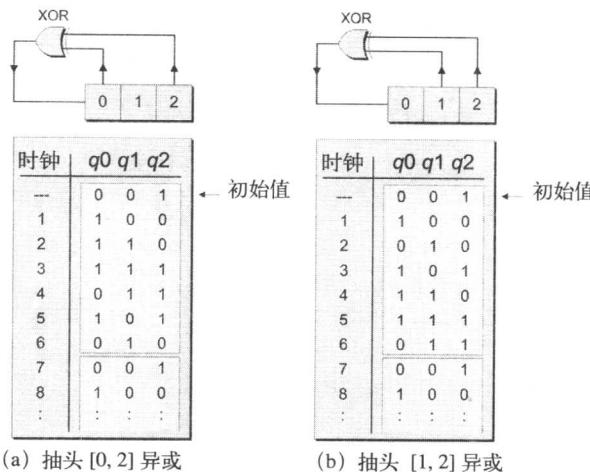
在这个例子中, 抽头就是位0和位2, 一种简单易行的表示方法是 $[0, 2]$ 。所有的寄存器共享一个时钟输入, 为了清楚起见, 符号图中并没有画出时钟。线性移位寄存器的输入数据由各抽头的异或或者同或产生, 其他的比特作为标准的移位寄存器使用。

线性移位寄存器产生的输出序列顺序由它的反馈函数 (异或和同或) 和所选择的抽头决定。例如, 有两个3位的线性移位寄存器, 使用异或反馈函数, 第一个的抽头是 $[0, 2]$, 第二个的抽头是 $[1, 2]$, 如图C-2所示。

^① 不要和两头蛇混淆, 两头蛇是指古典神话中的一种大毒蛇, 两头各有一个蛇头, 可以朝两个方向移动。

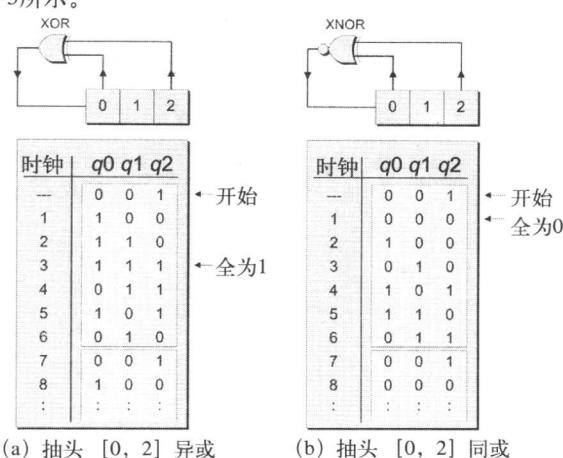
466

两个线性移位寄存器以同样的初始值开始，但是由于它们的抽头不同，它们的输出序列很快就会出现差异。有些情况下，线性移位寄存器将在输出了很有限的数值之后开始重复。



图C-2 选择不同抽头的差异

但是，图C-2中的两个线性移位寄存器都达到了最大长度，因为它们的输出在返回到初始值之前经历了所有可能的值（不包括所有比特都是0的情况）。一个含有 n 位的二进制数可以表示 2^n 个不同的数值，但是一个有 n 个寄存器的线性移位寄存器则只能表示 $(2^n - 1)$ 个不同的数值。例如，一个3位的二进制数可以表示 $2^3 = 8$ 个数值，但是图C-2中所示的3位线性移位寄存器只能表示 $2^3 - 1 = 7$ 个数值。这是因为使用异或反馈函数的线性移位寄存器不会越过所有位为0这种禁止出现的数值，而使用同或反馈函数的线性移位寄存器则不会越过全为1的数值^①，如图C-3所示。



图C-3 异或与同或反馈函数的比较

① 如果一个线性移位寄存器无意中进入了“禁止数值状态”，那么在没有外部事件干涉的情况下它将永远锁定在这个数值上。

C.3 更多的抽头组合

每一个线性移位寄存器都有许多种抽头组合能产生最大长度序列，问题是怎样将不能产生最大长度序列的组合同能产生的组合区分开来，因为不恰当地选择抽头可能会使寄存器进入一个只有有限个状态的循环。

作者随意写了一个简单的C程序，可以计算出使得2到32位线性移位寄存器达到最大长度的抽头序号。图C-4中列出了这些数值，其中的星号（*）表示该序列的长度是一个素数。

分别使用异或反馈函数与同或反馈函数的线性移位寄存器的抽头可能是相同的，但是它们产生的结果却不同。前面提到过，不同的抽头组合可能都可以实现最大长度线性移位寄存器，但是它们产生的序列却是不同的。例如，对于10位线性移位寄存器，有两种抽头组合可以产生最大长度序列，分别是[2, 9] 和 [6, 9]。还有20种4抽头组合、28种6抽头组合和10种8抽头组合都可以实现最大长度序列^①。

很重要的一点是，图C-4中所示的抽头组合对于某些应用可能不是最好的，例如，原始的多项式，其序列值在“随机”空间中均匀分布，它们只适合使用所产生的结果中的部分抽头组合。如果在实际设计中使用线性移位寄存器，要选择最优的抽头组合，最好的参考资料是W. Wesley Peterson和E.J. Weldon Jr.合著，由麻省理工学院出版社出版的*Error-Correcting Codes*。另外，也可参考25.8节中的CRC工具。

一点到多点的实现方式

考虑一个8位线性移位寄存器，产生最大长度序列需要的最少的抽头数量为4个。实际设计中，异或门只有两个输入端，所以一个四输入异或门必须用三个两输入异或门连成两级逻辑来实现。有时候即使线性移位寄存器支持最少两个抽头，但是可能有些特殊的原因却需要使用更多的抽头，例如八个抽头，这样将需要三级异或门逻辑。

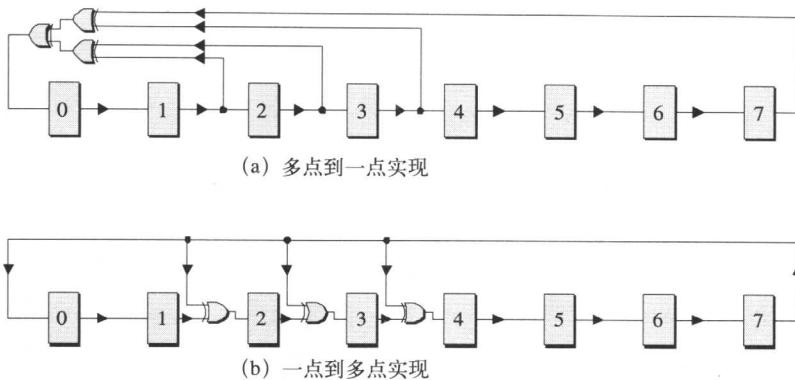
但是，反馈路径中的组合逻辑级数越多，对电路的最大时钟频率的影响就越大。一个

#位号	循环长度	抽头
2	3 *	[0,1]
3	7 *	[0,2]
4	15	[0,3]
5	31 *	[1,4]
6	63	[0,5]
7	127 *	[0,6]
8	255	[1,2,3,7]
9	511	[3,8]
10	1,023	[2,9]
11	2,047	[1,10]
12	4,095	[0,3,5,11]
13	8,191 *	[0,2,3,12]
14	16,383	[0,2,4,13]
15	32,767	[0,14]
16	65,535	[1,2,4,15]
17	131,071 *	[2,16]
18	262,143	[6,17]
19	524,287 *	[0,1,4,18]
20	1,048,575	[2,19]
21	2,097,151	[1,20]
22	4,194,303	[0,21]
23	8,388,607	[4,22]
24	16,777,215	[0,2,3,23]
25	33,554,431	[2,24]
26	67,108,863	[0,1,5,25]
27	134,217,727	[0,1,4,26]
28	268,435,455	[2,27]
29	536,870,911	[1,28]
30	1,073,741,823	[0,3,5,29]
31	2,147,483,647 *	[2,30]
32	4,294,967,295	[1,5,6,31]

图C-4 从2位到32位最大长度线性移位寄存器的抽头信息

^① Xilinx公司的应用程序说明XAPP052中列出了更长的表，涵盖了2到168位的线性移位寄存器。

解决方法是将上面讨论的多点到一点的实现方式颠倒过来，变为一点到多点的实现方式，如图C-5所示。



图C-5 多点到一点实现与一点到多点实现

用传统的多点到一点方式实现的8位线性移位寄存器的抽头为 [1, 2, 3, 7]。为了将这个线性移位寄存器转换为一点到多点的实现方式，只要将最高位的抽头（本例中是位7）直接反馈到最低位即可，这一位也可以单独同其他抽头（本例中是指 [1, 2, 3]）异或。

尽管这两种方式都能得到最大长度线性移位寄存器，但是它们产生的实际输出序列却是不同的。主要的优点是，使用一点到多点的实现方式时，反馈路径中的组合逻辑永远不会超过一级，因此不用去考虑使用了多少个抽头。

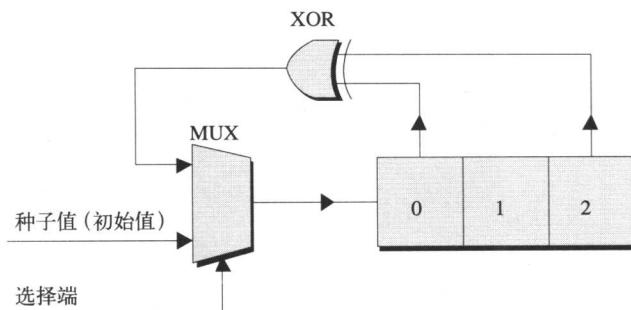
当然，FPGA有额外的考虑，那就是一个四输入查找表（LUT）的延迟与二输入、三输入和四输入的异或门相同。这种情况下，要实现一个超过四个抽头的线性移位寄存器时，只有多点到一点方式具有优势。

C.4 为线性移位寄存器设定初始值

基于异或逻辑的线性移位寄存器有一个缺点，如果所有寄存器的值都变成0，那么它将陷入死循环，无休止地对所有的0进行移位操作，同样，对于同或逻辑的线性移位寄存器也会全为1的死循环，这在以功耗为第一考虑因素的电路中尤其受到关注。每一个寄存器可以随机地初始化为逻辑0或者逻辑1，因此线性移位寄存器也可以从“禁止”数值中被“唤醒”。出于这个原因，有必要用一个初始值（种子值）对线性移位寄存器进行初始化。

基于同或逻辑的线性移位寄存器有一个有趣的特点，它允许所有的寄存器都为0，也就是说，用一个公共的清零信号就可以为同或的线性移位寄存器提供一个全为0的初始值。

为了载入一个专门的初始值，一个方法就是使用带有复位（reset）和置位（set）输入端的寄存器。用一个控制信号连接到一部分寄存器的复位端和另一部分寄存器的置位端，当这个控制信号有效时，线性移位寄存器将会载入一个硬线逻辑决定的初始值。但是对于某些特定的应用来说，却希望初始值能够改变，为此，可以在线性移位寄存器的输入端增加一个多路器，如图C-6所示。

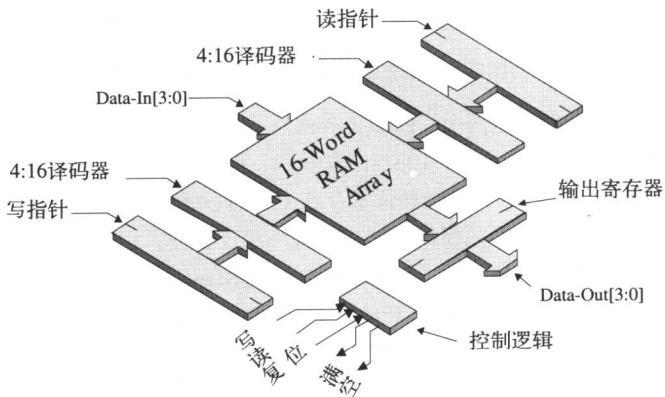


图C-6 载入可选择初始值的电路

选择多路器的种子值作为输入时，器件就是一个标准的移位寄存器，需要任何种子值都可以载入。载入种子值之后，再选择反馈路径，器件又回到了线性移位寄存器的操作模式。

C.5 在FIFO中的应用

一个线性移位寄存器能够产生一系列不同的数值序列，其实这一点在许多应用中都无关紧要，例如，一个4位宽、16字深的FIFO存储器，如图C-7所示。



图C-7 一个16字的FIFO

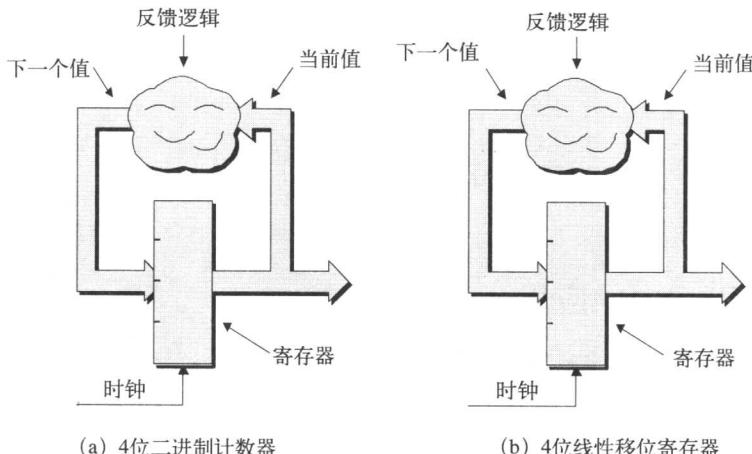
除了一些控制逻辑和一个输出寄存器外，FIFO还包含一个读指针和一个写指针。这些指针是4位的寄存器，经过4:16的译码器后从存储阵列中的16个字中选中一个。

读指针和写指针围绕着存储阵列永无休止地互相追赶。写输入端的一个有效边缘可以使得输入总线上的数据写入到写指针指向的字空间，然后写指针加一指向下一个空的字空间。同样，读输入端的一个有效边缘也会使得读指针指向的字被复制到输出寄存器中，然后读指针加一指向下一个含有数据的字空间^①。另外还需要一些逻辑来探测FIFO的满状态

^① 这些讨论假定使用先写后加和先读后加的技术，但是也有些FIFO使用先加后写和先加后读的方法。

和空状态，但是这与我们讨论的没有关系。

16字FIFO的写指针和读指针通常用4位二进制计数器来实现。但是，对于这种特定的应用，直观的反映表明二进制序列并没有什么本质上的优点，4位线性移位寄存器产生的序列同样能很好地满足要求。实际上，这两种实现形式的操作方式非常相似，如图C-8中的框图所示。



图C-8 二进制计数器与线性移位寄存器

几秒钟，你就会发现这两个框图除了名称外，其余部分都完全相同。关键是4位的二进制计数器的组合反馈逻辑需要很多的与门和或门，而4位线性移位寄存器的反馈逻辑只要一个二输入的异或门即可。因此线性移位寄存器需要的连线更少，占用的硅芯片面积更小。

另外，线性移位寄存器的反馈逻辑只经过一级逻辑门，但是二进制计数器的反馈逻辑却要经过多级逻辑门。这就是说，新的数据能够更快地到达线性移位寄存器中，这样可以提高时钟频率。在一些深度很大的FIFO中，需要更多比特的读写指针，这些差异就更加显著了。因此，有眼力的FIFO设计人员就有了一种有趣的选择，那就是线性移位寄存器^①。

C.6 将LFSR产生的序列修改为 2^n 个值

在上面提到的FIFO应用中，使用4位线性移位寄存器的唯一缺点是产生的序列值只有15个(2^4-1)，而二进制计数器则可以产生16个(2^4)。根据应用的不同，设计工程师可能不会将这看成是一个主要的问题，尤其是在更大的FIFO中，但是，如果必须让一个线性移位寄存器产生所有可能的值，也有一个简单的方法，如图C-9所示。

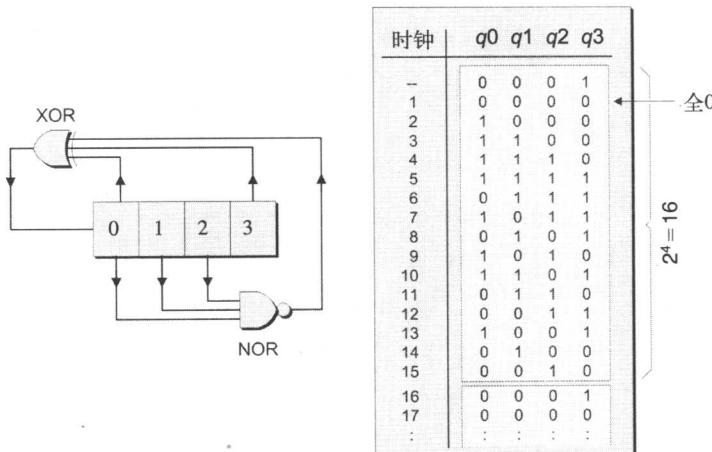
474

为了得到一个全0的数值，前一个值必须是最高位^②为1，其余位为0，在未修改的线性

^① 格雷码计数器(Gray Counter)也可以做到这一点，将在其他时间另行讨论。

^② 移位寄存器常使用这种方法，即最高位(MSB)在寄存器的右手一侧，而最低位(LSB)在左手一侧，这同我们通常使用方法正好相反。

移位寄存器中，下一个时钟到来时将在最低位产生一个1，其余位为0。但是，在图C-9所示的修改后的线性移位寄存器中，或非门的输出除了两种情况外都是0，这两种情况是：所有位都是0和除了最高位外都是0。这两个数值会使得或非门的输出为1，从而令异或门的输出与修改前相反，接下来就会使序列变为全0，然后线性移位寄存器又会继续其正常的转换过程。在使用同或反馈逻辑的线性移位寄存器中，用与门代替或非门就可以产生全为1的序列值。

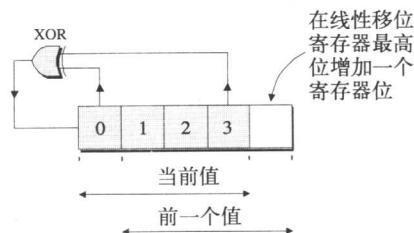


图C-9 修改线性移位寄存器产生 2^n 个序列值

C.7 访问以前的值

在有些应用中，必须利用寄存器以前的值。例如在某种FIFO中，当写指针指向读指针所指位置的前一个位置时就满足了full条件。所以，比较器必须比较写指针的当前值与读指针的前一个值。同样，对于empty条件，也是读指针指向了写指针所指位置的前一个位置时出现。即另一个比较器必须比较读指针的当前值和写指针的前一个值。

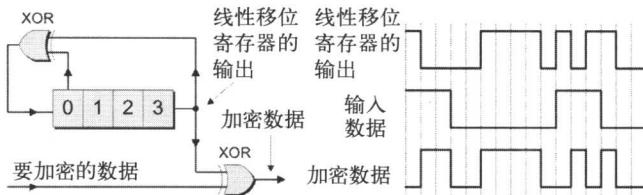
使用二进制计数器（假如由于某些原因，我们决定在FIFO中使用它们）时，有两种方法可以访问序列中的前一个值。第一种方法需要一套额外的影子寄存器（shadow register），每一次计数器增加的时候，首先将前一个值复制到影子寄存器中；另一种方法是使用组合逻辑对当前值译码得到前一个值。不幸的是，这两种方法都需要一些附加的逻辑。相比而言，线性移位寄存器则在本质上具有记住前一个值的能力，唯一需要做的只是在最高位再增加一个寄存器位而已，如图C-10所示。



图C-10 访问线性移位寄存器的前一个值

C.8 在加密和解密中的应用

线性移位寄存器产生的一系列值还可以用于对数据的加密（加扰）和解密（解扰）。如图C-11所示，将一个数据位流与线性移位寄存器的输出进行异或就可以对该数据流的加密。

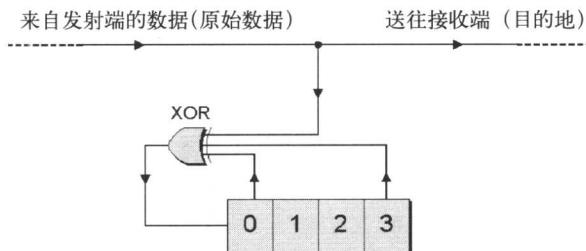


图C-11 使用线性移位寄存器对数据加密

接收端收到加密的数据流后，将它们与同样的线性移位寄存器的输出进行异或就可以对数据解密。显然，这只不过是一种非常简单的加密形式，很不安全，但是简单易行，成本低廉，因此在某些应用中也可能是有用的。

C.9 在循环冗余校验（CRC）中的应用

线性移位寄存器的传统应用是计算循环冗余校验码（cyclic redundancy check），用来探测数据通信中的错误。计算过程中，使用发送的数据流来修改反馈到线性移位寄存器的数值，如图C-12所示。



图C-12 CRC计算

最后一个CRC值存储在线性移位寄存器中，这个值称为校验和，与数据流中的任何一位都有关系。发送器发送完所有的数据后，再将校验和发送给接收器，接收器中含有同样的CRC计算电路，对收到的数据进行校验并产生校验和。接收器收到所有的数据后就将自己产生的CRC校验和与发送器发送过来的校验和进行比较，由此判断数据传输过程中是否发生了错误。

这种错误探测方式在传送数据量比较少的情况下非常有效。但缺点是必须等到所有数据传送完成才能知道是否有错误发生，如果传输过程中出现了错误，就必须重复整个传输过程。

实际应用中，人们对于一个4位CRC计算器保证传输数据的完整性能力并没有足够的信心，因为它只能产生15个不同的值。这就导致一个问题，称为混淆现象（aliasing），也

就是说，最后的CRC校验和与期望的相同，但是这个值其实是由多个错误彼此抵消而产生的。不过，随着CRC计算器位数的增加，多个错误互相抵消产生正确的校验和的可能性几乎等于零。由于这个原因，CRC计算器通常使用16位（可以产生65 535个不同的数值）或者更多的位数。

许多标准通信协议中都规定了CRC计算器中所使用的位数以及所用的抽头。选择抽头的原则是，只有一位的数据发生错误也会引起最后的校验和出现最大可能的中断。这样，除了用“最大长度”(maximal length)来衡量线性移位寄存器外，还可以用“最大位移量”(maximal displacement)。

除了在通信系统中检测数据的完整性外，CRC在许多领域都有应用，例如计算机病毒的检测。为了方便讨论，这里对计算机病毒的定义是：为了各种各样的目的而在计算机系统中发布的一个能够自我复制的程序。这样的目的非常广泛，从简单的恶作剧（例如显示一些滑稽或恼人的消息），到极其恶毒的做法（例如破坏数据或者造成操作系统崩溃）。

计算机病毒通过将自己依附在其他程序中达到隐藏和传播的目的。无论何时，只要带有病毒的程序一执行，就会激活病毒复制自己，但是操作系统粗略的检查只会显示出请求的文件正在执行。为了防止这种攻击，可以预先为系统中的每一个程序产生一个校验和，这个校验和的值是基于程序文件的二进制指令产生的。此后，反病毒软件就可以重新计算每一个程序的校验和，并与原始的校验和相比较。如果同一个程序的两个校验和有差异则说明这个程序已经被病毒感染^①。

C.10 在数据压缩中的应用

上面讨论的CRC计算器也可以用于数据压缩。电路板测试中有这种应用，即功能测试。将包含几千个元器件和众多导线的电路板通过边缘的连接器插入到功能测试机中，这些连接器可能含有几百个引脚。

测试机在电路板的输入端施加一组测试用例，并为板级各种传播延迟留足时间，然后用输出端的实际输出与系统中预先存储的一组值比较。对于一系列的测试用例不断重复这一过程，这些测试用例一般有成百上千组。

如果电路板在初步测试中出现问题，接下来可以利用一种更复杂的分析技术，即引导式探针(guided probe)来确定问题的根源。这种情况下，测试机指示操作者在电路板上某个特殊的位置放置探针，然后再重新运行一遍所有的测试用例。之后，测试机将探针得到的实际数值与系统中存储的预期值相比较。这个过程，即放置探针并运行测试，一直重复下去直到测试机找到出问题的元器件或者导线。

在支持引导式探针策略时需要考虑的一个主要问题是必须在系统中存储的对比用数据量。考虑一下一个测试序列中包含10 000种模式，驱动一块有10 000条导线的电路板。如果数据没有压缩，那么系统中必须为每一条导线存储10 000位的数据，整个电路板就需要存

^① 很不幸，计算机病毒的编写者通常都十分老练、精明，有些病毒具有计算自己的CRC的能力。

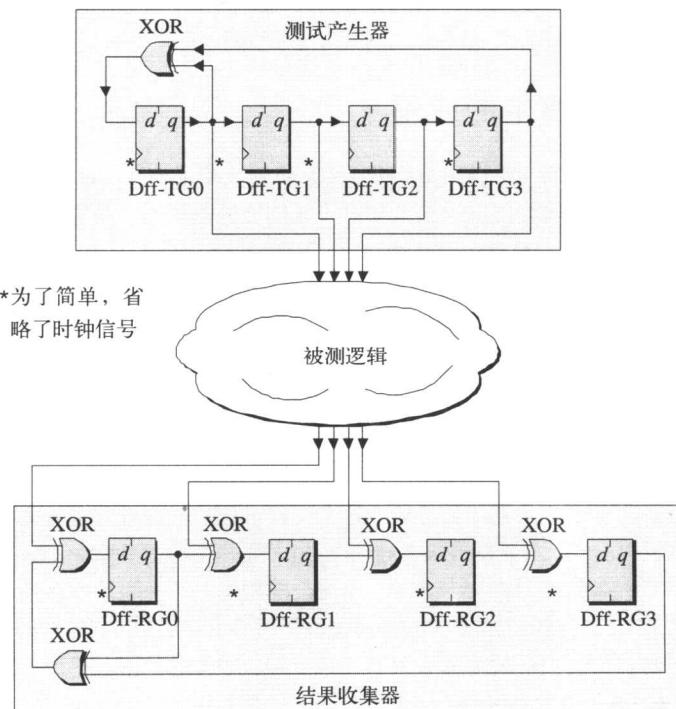
当这种病毒依附到一个程序中时，可以用一些虚假的二进制数值来掩护自己，这样当反病毒软件检查时返回的校验和与原始值相同。

储1亿位的数据。另外，每使用一个引导式探针，测试机就必须将探针得到的10 000位的数据与系统中存储的10 000位数据进行比较。所以，使用未经压缩的数据时，用在存储和处理上的费用将非常昂贵。

针对这些问题的一个解决方法是使用基于线性移位寄存器的CRC计算器。每一条导线的预期值序列先经过一个16位的用软件实现的CRC计算器。同样，引导式探针得到的实际数值也要经过同样的用硬件实现的CRC计算器。这种情况下，计算出来的校验和也称为签名 (signature)，引导式探针基于这一技术进行的处理过程称为签名分析 (signature analysis)。这样处理后，与使用的测试用例数量没有关系，系统仅仅需要为每条导线存储两个字节的数据。而且，每增加一个引导式探针，测试机只要比较探针产生的两个字节数据和存储的两个字节的预期值即可。这样，使用压缩数据后就比使用未压缩数据节约了巨大的存储空间和比较时间。

C.11 在内建自测试中的应用

复杂集成电路中可能使用的一种测试策略是内建自测试 (built-in self-test, BIST)。使用了BIST技术的器件含有一些专门的测试向量产生电路和结果收集电路，两者都可以实现线性移位寄存器实现，如图C-13所示。



图C-13 内建自测试 (BIST)

构成测试向量产生器的线性移位寄存器负责产生测试输入序列，而构成结果收集器的线性移位寄存器则捕获输出结果。可以看到，结果收集器中的线性移位寄存器经过了修改，这样可以接收并行数据。

另外，还需要一些额外的电路负责为测试向量产生器载入新的种子值，从结果收集器中获取最后的数值。为了简单起见，这些逻辑并没有在图中标出。

还需要注意，两个线性移位寄存器并不需要有同样多的位数，因为被测逻辑的输入数量可能与输出数量不同。

还有一点，测试向量产生器中所有的触发器都使用同一个时钟。同样，结果收集器中所有的触发器也使用同一个时钟，这两个时钟可能相同也可能不同（使用不同的时钟时，这些时钟必须同步）。为了简单，图C-13中没有标出时钟信号。

C.12 在伪随机数产生中的应用

许多计算机程序都依赖于随机性元素，电脑游戏（例如Space Invaders）就使用了随机事件来增加游戏者的乐趣，图形软件使用随机数来产生复杂的图形。所有的计算机仿真软件都利用随机数更精确地描述真实世界，例如，一些数字逻辑仿真软件（见第19章）就得益于随机测试激励的使用（例如使用外部中断）。随机测试激励可以完成更加真实的设计验证，从而能显示出结构化测试中难以发现的一些问题。

随机数产生器既可以用软件实现，也可以用硬件实现，尽管大多数产生的都不是真正的随机数，但是表面上看起来却是随机的，因此称为伪随机（pseudorandom）。实际上，伪随机数和真正的随机数相比还有一个优点，因为大多数计算机应用中通常都需要具有重复性，例如，设计人员重复一个数字仿真往往期望得到与上次同样的仿真结果，但是，设计人员也应该能够修改伪随机数产生器的种子值，以便按要求生成不同的序列值。

产生伪随机数的方法有很多种，其中之一就是使用线性移位寄存器，其抽头的选择经过特殊处理，便于提供一个良好的伪随机源。

C.13 最后的说明

线性移位寄存器具有构造简单、应用面广的特点，但是必须注意的是，如何为某种应用选择最佳的多项式（最终归结为选择抽头）是一项需要很多技巧和创意的任务。（据说常为某神秘艺术的主人而保留，千万不要说数学可怕到了足以使一个成年人垮掉，并且害怕得哭起来，别让我接触割圆多项式^①，那可是选择抽头的关键）。

^① 主要是因为我对割圆多项式到底是什么还一无所知。

术 语 表

ACM (adaptive computing machine) ——自适应计算机，是一类新型的数字集成电路 (IC)，具有包含大颗粒算法单元的节点式架构，每秒可以进行成百上千次重配置 (即自适应性)。

Adaptive computing machine——见ACM。

Address bus——地址总线，一组由处理器 (或类似器件) 使用的单向信号线，指向感兴趣的存储器地址。

A/D (analog to digital) ——模拟信号转换为数字信号。

Analog——模拟信号，其数值连续变化最接近真实世界的情况，只要测量技术允许就能达到足够的精度。

Analog circuit——模拟电路，产生和处理模拟信号的电路。

Analog to digital——见A/D。

Analogue——analog的英国写法。

Antifuse technology——反熔丝技术，制造可编程集成电路 (IC) 的一种技术，使用被称为反熔丝的导线作为可编程元件。新购进的反熔丝可编程器件没有任何完整的内部连接，编程时可在器件的输入端施加较高电压和电流的脉冲来有选择地“生成”某些内部连接。

Application-specific integrated circuit——见ASIC。

Application-specific standard part——见ASSP。

ASIC (application-specific integrated circuit) ——专用集成电路，面向专门的应用并定制化设计的集成电路，通常包含数百万个逻辑门，可实现极其大而复杂的功能。除了其设计生产是为满足某特定公司用户订单的要求外，ASIC类似于专用标准器件。

ASIC cell——ASIC单元，ASIC制造商定义的单元库中的一个逻辑功能。

Assertions/properties——断言/属性 (property) 这一术语来源于模型检查领域，指的是 (形式上) 要验证的设计所具有的某种功能性行为，例如请求后，希望在10个时钟周期内得到应答；断言 (assertion) 来源于仿真领域，指的是仿真期间要监控的设计所具有的某种功能性行为 (如果断言触发则标识出“违例”)。目前，可以在统一的环境和方法中使用形式化工具和仿真工具，因而属性和断言倾向于可交替使用。

ASSP (application-specific standard part) ——专用标准部件，面向专门的应用并定制化设计的集成电路，通常包含数百万个逻辑门，可实现极其大而复杂的功能。除了是面向多用户市场，包含在其产品中外，ASSP类似于专用集成电路 (ASIC)。

Asynchronous——异步，指信号不依赖于时钟信号而是对输入信号的变化立即做出反应。

Ball grid array——见BGA。

Bare die——裸片，未封装的集成电路。

Basic cell——基本单元，预先定义好的一组没有相互连接的晶体管和电阻。将这个基本单元规则地复制便形成门阵列形式的ASIC。

Bebop——一种爵士音乐，节奏很快，二战后的十年内非常流行。

BGA (ball grid array) ——球形栅格阵列，与插针栅格阵列 (PGA) 相似的封装技术。对于PGA，器件的外部管脚整齐地排列在封装体的焊垫上；对于BGA，则是一些很小的锡球被粘在焊垫上。

BiCMOS (bipolar-CMOS) ——双极CMOS。作技术解时，是指每个逻辑门的逻辑功能使用低功耗的CMOS晶体管实现，而输出则用驱动能力较强的双极型晶体管实现；作器件解时，是指内部逻

辑门由低功耗CMOS实现，而其输出引脚由高驱动能力的双极型晶体管驱动。

Binary digit——二进制数，一个二进制数通常简写为“bit”，即位，其值可以是0或1。

Binary encoding——二进制编码，当状态机需要最少的状态变量时使用这种状态分配方式。

Binary logic——二进制逻辑，使用两种不同的电平分别代表二进制数0和1，相应的逻辑值为False和True。

Bipolar junction transistor——见BJT。

BIST (built-in self-test)——内建自测试，在元件内部增加附加的逻辑以进行自测试的一种测试策略。

Bit——位，二进制数的简写，数值可以是0或1。

Bit file——位文件，见Configuration file。

BJT (bipolar junction transistor)——双极结型晶体管。

Bubble——在逻辑门符号的输入端使用的小圆圈，表示低电平有效；在输出端使用这个小圆圈表示输出的是反相或互补信号。有些工程师喜欢用bubble这个术语。

Boolean algebra——布尔代数，表示逻辑表达式的一种数学方法。

Built-in self-test——见BIST。

Bus——总线，是指信号的集合，这些信号执行同样的功能，传送同类数据。通常用向量来表示，例如，一条8位数据总线可表示为data[7:0]。

Byte——字节，8位二进制数（即8 bit）组成一个字节。

Cache memory——小型高速存储器，通常用SRAM实现，用来为CPU缓冲一些来自低速、低成本存储器件（如DRAM）的数据，还可以存储与程序相关的活动指令和活动数据^①，但大多数指令和数据存储在低速存储器中。

Capacitance——电容容量，两个非常靠近但相互绝缘的导体被施加不同的电压时具有存储电荷的能力，电容容量单位是法拉（farad）。

Cell——见ASIC cell、Basic cell、Cell library和Memory cell。

Cell library——单元库，由ASIC制造商定义的逻辑功能集合。设计人员可以决定使用何种类型的单元来进行连接以使器件实现期望的功能。

Central processing unit——见CPU。

Ceramic——陶瓷，无机非金属材料，例如氧化铝（alumina）、氧化铍（beryllia）、滑石（steatite）、镁橄榄石（forsterite）等，在高温中烧结而成，常被用来做电子器件的衬底和封装材料。

CGA (column grid array)——柱形栅格阵列，与插针栅格阵列（PGA）相似的封装，但在焊垫上是一些小锡柱。

Channel——有通道门阵列中两列基本单元之间的区域；MOSFET晶体管的源区和漏区之间的间隙。

Channeled gate array——有通道门阵列，按照基本单元栅格排列组织的专用集成电路（ASIC），各排列之间的区域称为通道。

Channelless gate array——无通道门阵列，按照一个大的单一的基本单元排列组织的专用集成电路（ASIC），也称为“单元海”或“门海”器件。

Checksum——检验和，利用线性反馈移位寄存器（linear feedback shift register, LFSR）或者相应的软件算法进行循环冗余码检测（cyclic-redundancy check, CRC），所得结果称为检验和，在功能测试的引导式探针（guided probe）中也可称为“签名”。

Chemical mechanical polishing——见CMP。

Chip——芯片，集成电路的通俗叫法。

^① 在这里，“活动”指的是程序正在使用的当前数据和指令或者操作系统认为程序要立即使用的数据和指令。

Chip scale package——见CSP。

Circuit board——电路板，多种电路互连技术的总称，包括刚性板、柔性板以及刚柔结合板，有单面板、双面板、多层板和散线板之分。

CLB (configurable logic block)——可配置逻辑块，Xilinx公司定义的术语，是比slice高一个层次的逻辑块。Xilinx公司的FPGA中，有些FPGA的每个CLB包含两个slice，而有些则含有4个slice。也可参考词条LAB、LC、LE和Slice。

Clock tree——时钟树，是指时钟信号在整个芯片中的布线方式。之所以称为时钟树，是因为主时钟不断分支（就像树枝一样，路径中的寄存器可看作是分支末端的“叶”）。使用这种结构可保证时钟信号尽可能同时到达所有触发器。

CMOS (complementary metal oxide semiconductor)——互补金属氧化物半导体，使用互补方式将NMOS晶体管和PMOS晶体管连接在一起构成逻辑门。

CMP (chemical mechanical polishing)——化学机械抛光，对晶圆（wafer）进行全局平面化的一个过程，即把增加金属布线（跟踪）层时引起的“突起”研磨掉，使晶圆的表面光滑而平整。

Column grid array——见CGA。

Combinatorial logic——见Combinational logic。

Combinational logic——组合逻辑，使用与门、或门、与非门、或非门等基本逻辑门完成逻辑功能，任何输出值直接与当前输入值组合相关。也就是说，输入信号的任何变化都将立即通过这些基本门电路传播到电路的输出端（也见Sequential logic）。

Complementary output——互补输出，是指使用两个输出端分别传送互补（相反）的逻辑值。其中一个称为实际输出，另一个则称为互补输出。

Complex programmable logic device——见CPLD。

Conditioning——见Signal conditioning。

Configurable logic block——见CLB。

Configuration commands——配置命令，配置文件中的一些指令，用来指导器件操作相关的配置数据。也可参考词条Configuration data 和 Configuration file。

Configuration data——配置数据，可用来直接定义可编程逻辑元件的状态。也可参考词条Configuration commands和 Configuration file。

Configuration file——配置文件，这个文件含有器件的配置信息，将这些配置信息下载到FPGA中就可以对其进行编程（配置）以实现特定的功能。对基于SRAM技术的FPGA，配置文件中既包含配置数据，也包含配置命令。在使用配置文件对器件进行配置的过程中，所传送的数据被称为配置比特流（configuration bitstream）。也可参考词条Configuration commands和Configuration data。

Constraints——约束，在形式验证中，这一术语来自模型检查领域。形式验证工具在执行其功能时会考虑所有可能的输入组合，因此有必要将输入限制在合理的行为上，否则对于那些在实际设计中不会发生的属性违例也会报错。

Core——见Hard core和Soft core。

Corner condition——见Corner case。

Corner case——边界条件，与具体设计密切相关但很难达到和实施的功能性条件，即是一种极端情况。

CPLD (complex PLD)——复杂可编程逻辑器件，含有大量简单可编程逻辑器件（SPLD），如PAL，这些SPLD共用片内的可编程内连矩阵资源。

CPU (central processing unit)——中央处理单元，计算机的大脑，负责执行决策处理和数值运算等。

CRC (cyclic redundancy check)——循环冗余码校验，用来检查数据在通信过程中是否发生错误，通常使用线性反馈移位寄存器（LFSR）来实现。数据压缩等应用中也会用到这种运算。

CSP (chip scale package)——芯片级封装，封装后的集成电路与裸片大小差不多。

Cyclic redundancy check——见CRC。

D/A (digital to analog)——数字向模拟信号转换

Data bus——数据总线，计算机使用的双向信号线，用来在CPU和存储器之间传递数据，具体地说，是指在各功能模块之间传递数据的一组信号。

Data-path function——数据路径功能模块，是指处理数据的数字加法器、计数器和乘法器等。

DCM (digital clock manager)——数字时钟管理器，有些FPGA的时钟管理器是基于锁相环(phase-locked loops, PLL)的，而有些则基于数字延迟锁相环(digital delay-locked loops, DLL)。Xilinx公司使用的DCM指的是一种高级时钟管理器，是锁相环的超集。也可参考词条DLL和PLL。

Declarative——声明，在形式验证中，声明是指设计的结构化上下文中存在的断言、属性、事件或约束等，连同设计中其他结构化元素（例如采用结构化实例形式的模块）一起被评估。换句话说，声明后的断言/属性总是“打开/活动”(on/active)的，而进程中对应的概念只在执行HDL代码中某个特殊的路径时才是“打开/活动”(on/active)的。

Deep submicron——见DSM。

Delay-locked loop——见DLL。

DeMorgan transformation——DeMorgan转换，就是将布尔表达式转换为更方便、更容易理解的形式。

Die——裸片，未封装的集成电路。

Digital——数字化，就是将信号用有限个离散状态来表示，这称为量化，然后用数字表示其中的一个状态，数字化的精度取决于量化的阶数。

Digital circuit——数字电路，使用逻辑门产生和处理数字信号的电路。

Digital clock manager——见DCM。

Digital delay-locked loop——见DLL。

Digital signal processing/processor——见DSP。

Digital to analog——见D/A。

Diode——二极管，具有两个终端单向导电性的元件，反方向应用时类似一个未闭合的开关。目前，提到二极管这个术语，几乎总是指半导体器件，其实真空管也有同样的功能。

Discrete device——分立元件，一般是指采用独立封装的电子元件，例如电阻、电容、二极管和晶体管等。与由少数基本逻辑门组成的简单集成电路连接时，偶尔也会用到这个术语。

DLL (digital delay-locked loop)——数字延迟锁相环，有些FPGA的时钟管理器基于锁相环，而有些则基于数字延迟锁相环，顾名思义，数字延迟锁相环本质上是数字的。DLL的支持者称，DLL在精度、稳定性、电源管理、抗干扰性以及抖动性能方面具有优势，至于为什么没有简写为DDLL，目前还不得而知。也可参考词条PLL。

DSM (deep submicron)——深亚微米，一般用来指集成电路中含有小于0.5微米的结构。

DSP (1) (digital signal processing)——数字信号处理，表达和处理数字信号的电子学分支。处理方式包括压缩、解压缩、调制、错误修正、滤波以及对音频（声音、音乐等）、视频、图像和类似数据的处理，例如电信、雷达和包括医学成像在内的图像处理等；**(2) (digital signal processor)**——数字信号处理器，一种专用微处理器，在处理某种特定数据时比通用微处理器更快也更高效。

Dynamic formal verification——动态形式验证，设计的某些部分很难通过仿真来验证，因为它们往往深深地隐藏在设计中，很难由输入信号控制。为了解决这个问题，一些验证方案在仿真中设置边界条件，当满足这些边界条件时就自动中止仿真器，然后调用静态形式验证引擎尽可能深入地对边界条件进行评估，这种将仿真与传统静态形式验证结合起来的方式称为动态形式验证。也可参考词条Corner case、Formal verification和Static formal verification。

Dynamic RAM——见DRAM。

ECL (emitter-coupled logic)——发射极耦合逻辑，使用双极结型晶体管进行特殊的连接和配置组成的逻辑门，即用BJT的发射极作为耦合端，而不是用传统的集电极。

Edge sensitive——边缘敏感，逻辑功能模块的输入在发生逻辑跳变时才会影响模块。

EEPROM or E²PROM (electrically erasable programmable read-only memory)——电可擦除只读存储器，内容可由设计人员进行电编程的存储器集成电路，同时内容也可以进行电擦除，使器件可重复编程。

Electrically erasable programmable read-only memory——见EEPROM。

Emitter-coupled logic——见ECL。

EPROM (erasable programmable read-only memory)——可擦除只读存储器，内容可由设计人员进行电编程的存储器集成电路，同时也可以将器件暴露在透过石英窗的紫外(UV)光下对存储的内容进行擦除。

Equivalency checking——见Formal verification。

Equivalent gate——等效门，基于ASIC的一个概念，每一种逻辑功能模块都对应着各自的等效门数，以便对逻辑功能模块和器件进行比较；然而，等效门的定义也由于读者的不同而有所差别。

Erasable programmable read-only memory——见EPROM。

Event——事件，形式验证中，事件同断言/属性相同，一般而言，事件可能被看作是断言/属性的一个子集。但是，它们也有差别，断言/属性一般用来捕获异常行为，而事件则用来指定期望的正常行为以便进行功能覆盖率分析。

Falling edge——见Negative edge。

FET (field-effect transistor)——场效应管，通过控制（或“门”）信号产生的电磁场来对晶体管进行开关。

Field-effect transistor——见FET。

Field-programmable gate array——见FPGA。

Field-programmable interconnect chip——见FPIC^①。

Field-programmable interconnect device——见FPID。

FIFO (first in first out)——先进先出存储器，读取数据的顺序和写入数据的顺序完全一致，即先入者先出，后入者后出。

Finite state machine——见FSM。

Firm IP——固IP，在FPGA的相关领域，固IP是指高级功能模块库。但是，和软IP不同的是，固IP已经被优化过，经过了映射、布局和布线，最终表现为一组可编程逻辑块（可能结合了一些硬IP块，比如乘法器等）。每一个预定义的固IP都可以在设计中被实例化一次或多次。也可参考词条Hard IP 和 Soft IP。

Firmware——固核，是指被下载到非易失性存储器中的程序或指令序列。

First in first out——见FIFO。

FLASH memory——闪速存储器，结合了EPROM和E²PROM优点的一种新颖的存储技术。之所以使用FLASH，即闪速，是因为与EPROM相比，它有更快的存取速度。

Formal verification——形式验证，不久前，大多数设计工程师都认为形式验证和等效性检查是同义词。在本书中，等效性检查器是指一种工具，能使用形式化（非常精确的）技术来比较同一个设计的RTL版本和门级网表之间的差异，以此来决定这两个版本是否具有同样的功能。实际上，也可以将等效性检查看作是形式验证的子集，称为模型检查，可用来探测系统的状态空间以便测试某种“属性”是否为真。也见Static formal verification和Dynamic formal verification。

FPGA (field-programmable gate array)——现场可编程门阵列，内部包含可配置逻辑块和可配置内连线资源的数字集成电路。设计工程师可以配置（编程）FPGA实现非常多的功能。

FPIC (field-programmable interconnect chip)——现场可编程互连芯片，FPID的另一个所有名。

FPID (field-programmable interconnect device)——现场可编程互连器件，可以将逻辑器件连在一起并能用基于SRAM的标准FPGA的配置方式进行动态重配置的器件。由于每一个FPID可能

^① FPIC是Aptix公司的注册商标。

有1 000多个引脚，因此只有少数这种器件需要放置在电路板上。

FR4——使用最普遍的电路板绝缘基材，FR4是利用环氧树脂(epoxy)将玻璃纤维粘结而成。高温高压下，玻璃纤维熔化并结合在一起，然后凝固，因此电路板具有一定的强度和硬度。前两个字母，即FR，代表“flame retardant”，而知道那个“4”代表什么的人就没有几个了。从技术层面讲，FR4是一种玻璃纤维，有人将这些化合物称为玻璃纤维板或玻璃纤维层，但这并不常见。

Full custom——全定制，一种专用集成电路，制造IC所用的每一层掩模完全由设计工程师自己设计，ASIC制造商不在衬底上提供任何单元库或预制器件。

Functional latency——功能潜伏，在任何既定的时间，一个器件或系统中，只有一部分逻辑功能在起作用，即处于激活状态。

Fuse——见 Fusible link technology和Antifuse technology。

Fusible-link technology——熔丝连接环技术，使用细微的熔丝作为编程元件来生产可编程集成电路。这种集成电路在出厂时，内部所有连接都是完整的，在输入端施加适当的高电压和高电流脉冲就可以有选择地将内部的连接打断，从而实现对器件编程的目的。

FSM (finite state machine)——有限状态机，使用有限个状态以及这些状态之间有序的转换实现某项功能，具体实现可以是硬件，也可以是软件。

GAL (generic array logic)^①——通用阵列逻辑，Lattice半导体公司发明，是PAL器件的一个变种。

Garbage in garbage out——见GIGO。

Gate array——门阵列，一种专用集成电路(ASIC)，内部含有制造商预制的阵列方式排列的元件，这些元件(晶体管与电阻器)之间没有互连，并以基本单元方式组织。设计人员依据单元库中的单元以及单元之间的互连资源可以定义器件的功能，然后由制造商按照功能定义生产制造掩模。

Generic array logic——见GAL。

Geometry——几何尺寸，集成电路中内部结构的尺寸，这些结构通常包括导线宽度和晶体管的沟道长度。其他一些特征尺寸都来源于与这些结构尺寸的比率。

Giga——单位限定词，符号为G，表示 10^9 。例如，3GHz就代表 3×10^9 Hz。

GIGO (garbage in garbage out)——无用的输入输出，电子工程师的玩笑话，计算机程序员对此也很熟悉。

Glue logic——胶合逻辑，用来将较大的逻辑块、功能单元或器件连接(胶合)在一起的少量简单逻辑。

Gray code——格雷码，是指一组二进制数序列，相邻的两个值只有一位不同，例如00, 01, 11, 10。

Ground plane——接地层，衬底中的一个传导层，为其他元件提供接地和参考电平。可能存在多个相互绝缘的接地层。

Guard condition——监护条件，状态机中两个状态之间相互转换时必须满足的条件，通常是一个布尔表达式。

Guided probe——引导式探针，功能测试的一种形式，测试电路板时可引导操作员隔离出有问题的元器件或线路。

Hard core——硬核，在数字电子学中，核(core)通常是指一种规模较大的通用逻辑功能模块，它可以作为一个构造块成为另一个更大的芯片设计的一部分。例如，如果一个ASIC包含一个嵌入式微处理器，那么这个微处理器就被称为“微处理器核”。这类核还包括微控制器核、数字信号处理器核、通讯功能核(如UART)等。这些核可能由设计团队内部开发，但一般都是从第三方IP厂商处购买。

在ASIC和FPGA两种实现技术中，对于硬核的理解会有所不同。对于ASIC而言，硬核就是一个逻辑门组成的电路块，其内部逻辑门的相对物理位置和互连关系都已经定义好且不能改变。布局布线软件会将这个逻辑块看作一个黑盒子，也就是说，布局布线软件只能决定其整体位置，而不能改变

^① GAL是Lattice Semiconductor公司的注册商标。

其内部，黑盒子的内部完全是锁定的，布局布线软件的输出随后会被用来生成制造芯片用的光刻掩模。比较起来，在FPGA中的情况有所不同，任何硬核都是使用内嵌在FPGA内部的硬线块来进行物理实现的。

一个设计可能含有一个或多个硬核，同时也可能结合一些软核以及用户定义的其他逻辑块。也可参考词条 Soft core。

Hardware——硬件，通常的理解是指组成电子系统的任何物理设备，包括元器件、电路板、电源、机壳和监视器等。

Hard IP——硬IP，对于FPGA而言，硬IP是指预先实现了的逻辑块，例如微处理器核、Gbit接口、乘法器、加法器、MAC核等等，这些核都依据功耗、硅芯片真实状态和性能要求进行了尽可能高效的设计。每个FPGA器件族有不同的这样的块组合可供使用，内部的可编程逻辑块数量也有所不同。也可参考词条Soft IP 和 Firm IP。

Hardware description language——见HDL。

HDL (hardware description language)——硬件描述语言，现在的数字集成电路可能包含数百万逻辑门，使用原理图方式来管理如此复杂的设计是完全不可能的。因此，与原理图方式相反，高端IC设计现在都是用文本格式的硬件描述语言来描述的，流行的硬件描述语言有Verilog、VHDL、SystemVerilog和SystemC等。

HDL synthesis——HDL综合，逻辑综合的新名称。也可参考词条Logic synthesis 和 Physically aware synthesis。

Hertz——见Hz。

High-impedance state——高阻态，是指信号没有任何驱动时的状态。高阻态常用字母Z表示。

Hz (hertz)——赫兹，频率的单位。1Hz就等于每秒循环或振动一次。

IC (integrated circuit)——集成电路，在单块半导体材料的表面上制造电阻、二极管和晶体管等元件并组成具有特定功能的电路。

ICR (in-circuit reconfigurable)——在电路可配置，基于SRAM的或类似的器件在不脱离应用系统的情况下可动态地进行重新配置。

Impedance——阻抗，电路中的电阻、电容和/或电感器件，以及一些寄生效应产生的对电流的阻碍作用。

Implementation-based verification coverage——基于实现的验证覆盖率，对具体实现中的微观结构作验证时使用的一个衡量表征。这涉及在RTL代码中插入能产生某些特定边界条件的描述，例如，FIFO的深度以及FIFO处于“几乎满”和“满”时的边界条件。这些具体实现中的细节在规范级几乎是不可见的。也可参考词条Macroarchitecture definition、Microarchitecture definition和Specification-level coverage。

In-circuit reconfigurable——见ICR。

Inductance——电感，流过导体的电流可以感应出电磁场，具有存储能量的特性，电感的基本单位是亨（henry）。

In-system programmable——见ISP。

Integrated circuit——见IC。

Intellectual property——见IP。

IP (intellectual property)——知识产权，当一个设计团体接受任务要设计一个复杂的集成电路，而不是重新发明轮子时，他们可能会购买一个或多个由其他人创建的功能模块，这些功能模块就是知识产权或IP。IP的范围非常广，从简单的逻辑到复杂的通信模块和微处理器，不一而足，像微处理器这样复杂的模块通常称为核。也可参考词条Hard IP、Soft IP和Firm IP。

ISP (in-system programmable)——系统内可编程，基于EEPROM、FLASH、SRAM或类似的集成电路，在不脱离电路板的情况下能够重新被编程，以实现新功能。

JEDEC (Joint Electronic Device Engineering Council)——电子元件工业联合会，专门创建、

批准、仲裁和审查电子元件方面的工业标准。在可编程逻辑方面，JEDEC是指一个文本文件，包含器件的编程信息，文件格式是JEDEC批准的标准，通常称为JEDEC文件。

Jelly-bean logic——芝麻逻辑，功能简单并相对固定的小型集成电路，例如含有4个2输入“与”门的集成电路。

Joint Electronic Device Engineering Council——见JEDEC。

Kilo——千，单位限定词，符号为K，即 10^3 ，例如，3 KHz 代表 3×10^3 Hz。

LAB (logic array block)——逻辑阵列块，Altera公司对含有许多逻辑单元（logic elements，LE）的可编程逻辑块的命名。也可参考词条CLB、LC、LE和Slice。

LC (logic cell)——逻辑单元，Xilinx公司的FPGA内建的逻辑块，一个LC含有一个4输入查找表（LUT）、一个多路器（multiplexer）和一个寄存器（register）。也可参考词条CLB、LAB、LE和Slice。

LE (logic element)——逻辑单元，Altera公司当前的FPGA内建的逻辑块，一个LE含有一个4输入查找表（LUT）、一个多路器（multiplexer）和一个寄存器（register）。也可参考词条CLB、LAB、LE和Slice。

Least-significant bit——见LSB。

Least-significant byte——见LSB。

Level sensitive——电平敏感，逻辑功能模块的逻辑转换仅与输入信号的当前电平值有关，而与逻辑值的转换边缘无关。

LFSR (linear feedback shift register)——线性反馈移位寄存器，将寄存器链中的两个或更多寄存器进行“异或”或者“同或”后作为移位寄存器链的输入。

Linear feedback shift register——见LFSR。

Literal——布尔等式中的变量（非真即假）。

Logic function——逻辑函数，对数字进行有关的操作并返回一个数值值。

Logic array block——见LAB。

Logic cell——见LC。

Logic element——见LE。

Logic gate——逻辑门，基本逻辑功能的物理实现。

Logic synthesis——逻辑综合，自动将高抽象级别的文本式设计描述（通常用硬件描述语言在寄存器传输级描述）转换为等效的寄存器和布尔表达式的过程。综合工具自动执行简化和最小化操作，最终输出是门级网表。也可参考词条HDL synthesis and Physically aware synthesis。

Lookup table——见LUT。

LSB——(1) (least-significant bit) 最低位，二进制数的最后一一位，通常是指最右边的一位；(2) (least-significant byte) 最低字节，多字节中的最末一个字节，通常指最右边的一个字节。

LUT (lookup table)——查找表，在FPGA中有两种基本的可编程逻辑块实现方式可以形成中等粒度的结构，分别是多路器（MUX, multiplexer）和查找表（LUT, lookup table）。其中的查找表使用一组输入信号作为查找索引或指针。也可参考词条CLB、LAB、LC、LE和Slice。

Macroarchitecture definition——宏观结构定义，一项设计，它由原始概念开始，而其高层次定义则由系统结构设计人员决定。宏观结构正是在这个阶段制定的，例如，将设计划分为硬件和软件两个部分，选择微处理器核和总线结构等等。据此制定出的规范就可以交由硬件工程师来开始宏观结构的定义过程。也可参考词条Microarchitecture definition。

Magnetic random-access memory——见MRAM。

Magnetic tunnel junction——见MTJ。

Mask——见Photo-mask。

Mask programmable——掩模可编程，只读存储器类器件在制造过程中使用一套独特的光刻掩模就可被编程。

Maximal displacement——最大位移，选择线性反馈移位寄存器的几个抽头以便只改变输入数据中的一个位就能引起寄存器内容的最大中断。

Maximal length——最大长度， n 位线性反馈移位寄存器（LFSR）在返回到原始值前经历的 $(2^n - 1)$ 个状态。

Maxterm——最大项，逻辑函数的每个输入逻辑变量以原变量或者反变量的形式出现，且只出现一次，进行或运算，这种包含所有输入逻辑变量的或项称为最大项。

MCM (multichip module)——多芯片组件，高级互连和封装技术，就是将多个IC裸片直接安装在公共基板上形成更加复杂的集成电路。

Mega——单位限定词，符号为M，表示一百万或 10^6 。例如，3MHz代表 3×10^6 Hz。

Memory cell——存储单元，存储器的最小单位，可存储一位二进制数。

Memory word——存储字，包含多个存储单元。存储字中所有存储单元的存取操作都是同时进行的。

Metalization layer——金属层，集成电路中的可导层，有选择地进行蚀刻后形成逻辑门之间的内连线。集成电路中可能有多个被绝缘层分开的金属层。

Metal-oxide semiconductor field-effect transistor——见MOSFET。

Microarchitecture definition——微观结构定义，一项设计，从原始概念开始，而其高层次定义又是由系统结构设计人员制定。据此制定出的规范就可以交由硬件工程师来开始微观结构定义部分的开发过程，例如细化控制结构、总线结构和基本的数据路径等。举一个简单的例子，例如FIFO就需要赋予一定的属性，如数据宽度和深度，还有一些行为特征，如块写、非块读以及空或满时表现什么行为等等。微观结构定义常常是在头脑激荡会或者称为自由讨论会上完成的，可能包含某些固定的操作，如并行对串行，设计的流水线部分对非流水线部分，共享公共资源等，例如两次操作共享一个乘法器与复制资源的使用等。

Micro——微，单位限定词，符号为μ，代表百万分之一，例如， $3\mu\text{s}$ 就是 $3 \times 10^{-6}\text{s}$ 。

Microcontroller——见μC。

Microprocessor——见μP。

Milli——毫，单位限定词，符号为m，代表千分之一，例如， 3ms 就是 $3 \times 10^{-3}\text{s}$ 。

Minimization——最小化，将布尔表达式化简为最简单形式的过程。

Minterm——最小项，逻辑函数的每个输入逻辑变量以原变量或者反变量的形式出现，且只出现一次，进行与运算，这种包含所有输入逻辑变量的与项称为最小项。

Mixed signal——混合信号，指集成电路中既有模拟元件又有数字元件。

Model checking——见 Formal verification。

Moore's law——摩尔定律，1965年，Gordon Moore（1968年与人合办Intel公司）指出，存储器件的容量大约每18个月就翻一番，这后来被称为摩尔定律，并被应用到许多电子学领域。

MOSFET (metal-oxide semiconductor field-effect transistor)——金属氧化物半导体场效应管。

Most-significant bit——见MSB。

Most-significant byte——见MSB。

MRAM (magnetic RAM)——磁性随机存取存储器，一种2005年前后上线的存储器，具有可将SRAM的高速度、DRAM的大容量和FLASH的非易失性以及低功耗特点结合在一起的潜力。

MSB——(1) (most-significant bit) 二进制数的最高一位，通常是指最左边的一位；(2) (most-significant byte) 多字节中的最高一个字节，通常指最左边的一个字节。

MTJ (magnetic tunnel junction)——磁性隧道结，一种三层结构，中间为一层很薄的绝缘层，两面各有一层铁磁体。两条导线（一条“行”线，一条“列”线）的交叉点间放置一个MTJ就构成一个MRAM的存储单元。

Multichip module——见MCM。

Multiplexer (digital) ——多路器，使用二进制数（或地址）从多个输入中选择一个数据作为输出数据。

Nano——纳，单位限定词，符号为 n，代表10亿分之一（即 10^{-9} ）。例如，3 ns代表 3×10^{-9} s。

Negative edge——下降沿，信号从逻辑1跳变到逻辑0。

Nibble——见 Nybble。

NMOS (N-channel MOS) ——N沟道MOS管，指MOSFET器件中半导体材料的掺杂顺序，也就是说，在P型材料上构造N型区域。

Noise——噪声，在电子信号传输过程中加入到其中的各种无用信号。电容电感耦合以及外部电磁干扰都可能引起噪声。

Nonrecurring engineering——见NRE。

Nonvolatile——非易失性，指存储器件所存储的数据在掉电后不丢失。

NPN (N-type_P-type_N-type) ——指双极结型晶体管中半导体的掺杂顺序。

NRE (nonrecurring engineering) ——在本书中是指与设计ASIC、ASSP和FPGA相关的成本。

N-type——N型半导体，所掺杂质可以提供电子。

Nybble——4位二进制数。

Ohm——欧姆，电阻单位，常用希腊字母Ω表示，例如 $1M\Omega$ 表示一百万欧姆。

One-hot encoding——独热编码，状态机的一种状态编码方式，每个状态由一个单独的状态变量表示，任何时间只有一个这种变量是活动的（即“热的”）。

One-time programmable——见OTP。

Open Vera Assertions——见OVA。

Open Verification Library——见OVL。

Operating system——操作系统，负责管理计算机的核心操作和底层用户接口的程序集合。

OTP (one-time programmable) ——只能被编程一次的可配置（编程）器件，如SPLD、CPLD或FPGA。

OVA (OpenVera Assertions) ——为了最高效地描述断言/属性而专门设计的一种形式验证语言。OVA在创建复杂而规则的表达式方面具有强大能力，并且使用非常短的代码就可以描述复杂的行为。

Synopsys公司以IBM的Sugar语言为基础开发了OVA，后来将这种语言捐献给了Accellera组织（www.accellera.org）的SystemVerilog委员会，并形成了SystemVerilog Assertions。也可参考词条PSL、Sugar、SVA。

OVL (Open Verification Library) ——开放验证库，由Accellera组织（www.accellera.com）管理的断言/属性模型库，有VHDL和Verilog 2001两个语言版本。

Pad grid array——见PGA。

PAL (programmable array logic) ^①——可编程阵列逻辑，器件内部有“与门”阵列和“或门”阵列，其中与门阵列是可编程的，而或门阵列不可编程。也可参考词条PLA、PLD和PROM。

Parasitic effects——寄生效应，由磁轨或器件拓朴内部干扰阻抗、电容或电感引起的效应。

PCB (printed circuit board) ——印制电路板，电路板的一种，导线被“印制”在电路板的一面或两面，也可能含有内部的一些信号层以及电源层和接地层。在美国，普遍使用另一个名称，即printed wire board (PWB)。

Peta——单位限定词，符号为P，代表 10^{15} ，即1000T。例如，3PHz就是 3×10^{15} Hz。

PGA (1) (pad grid array) ——焊盘栅格阵列，在封装基座上使用焊盘式栅格阵列作为器件的外部连接方式；**(2) (pin grid array)** ——插针栅格阵列，在封装基座上使用插针式栅格阵列作为器件的外部连接方式。

Phase-locked loop——见PLL。

① PAL是Monolithic Memories公司的注册商标。

Physically aware synthesis——物理综合，对大多数人而言，物理综合是将实际的布局信息和设计中的各种逻辑单元联系起来，并利用这些信息估计精确的线路延时，然后再利用这些延时对布局进行微调和其他优化。有趣的是，物理综合总是在使用相对传统的逻辑/HDL综合引擎首次成功地综合后才开始的。也可参考logic synthesis。

Photo-mask——光掩模，一张非常薄的材料，上面刻有集成电路版图图形，有图形的位置可以让紫外线（ultraviolet, UV）穿过，其他位置阻挡紫外线，这样就可以在集成电路表面上形成需要的结构。

Pico——皮，单位限定词，符号为 p，代表 10^{-12} 。例如，3ps 就是 3×10^{-12} s。

Pin grid array——见PGA。

PLA (programmable logic array)——可编程逻辑阵列，传统可编程逻辑器件中，用户可配置自由度最大的一种，因为“与”门阵列和“或”门阵列都是可编程的。也可参考词条PAL、PLD和PROM。

PLD (programmable logic device)——可编程逻辑器件，一种集成电路，其内部结构由制造商预先确定，但是工程师可以对器件进行重新配置（编程）以实现多种功能，本书中假定PLD包含简单PLD（SPLD）和复杂PLD（CPLD）。同FPGA相比，这些器件所含的逻辑门数量相当有限，实现的功能也更少也更简单。

PLI (programming-language interface)——编程语言接口，是伴随Verilog（硬件描述语言）和Verilog-XL（仿真器）一起出现的概念。这类概念更通用的名称是应用程序接口（即API）。API是一个软件函数库，通过它，外部程序可以将数据传递到应用程序中，也可以从应用程序中访问数据。因此，Verilog PLI实际上是一个API，允许用户扩展Verilog语言及其仿真器的功能。

PLL (phase-locked loop)——锁相环，有些FPGA的时钟管理器是基于锁相环（phase-locked loops, PLL）的。从20世纪40年代开始，模拟电路中就已经使用锁相环了，不过最近几年用数字方法实现的锁相环在处理数字信号方面也取得了令人满意的效果。今天的PLL既可以用模拟技术实现由可以用数字技术实现。也可参考词条DLL。

PMOS (P-channel MOS)——P沟道MOS管，指MOSFET中半导体材料的掺杂顺序，也就是说，在N型材料上构造P型结构。

PNP (P-type_N-type_P-type)——指双极结型晶体管中半导体的掺杂顺序。

Positive edge——上升沿，信号从逻辑0向逻辑1的转换。

Power plane——电源层，衬底或底层中的一个导电层，可为元器件提供电源。有可能存在多个相互绝缘的电源层。

Pragma——编译指令，pragmatic information的缩写，是指插入到源代码（包括C/C++和HDL代码）中的某种专门的伪注释，它们可以由编译器解释，也可以被其他工具使用（这是一个通用术语，除了形式验证技术外，其他许多工具都在使用基于pragma的技术）。

Primitives——基本元件，简单逻辑功能，例如BUF、NOT、AND、NAND、OR、NOR、XOR和XNOR等逻辑门，也可以称为基本逻辑门，即primitive logic gates。

Printed circuit board——见PCB。

Printed wire board——见PWB。

Procedural——进程，在形式验证环境中，进程是指在一个执行过程或者一组连续语句中（例如VHDL语言中的process块或者Verilog语言中的“always”块）描述的断言/属性/事件/约束（所以有时候也称为“上下文中的断言/属性”）。这种情况下，断言/属性是内建在设计逻辑中，通过一组连续语句中的路径才能对其进行评估。

Product-of-sums——和之积，真值表中所有输出为0的项对应的最小项进行逻辑“与”所得的布尔表达式。

Product term——乘积项，一组布尔变量的“与”运算。

Programmable array logic——见PAL。

Programmable logic array——见PLA。

Programmable logic device——见PLD。

Programmable read-only memory——见PROM。

Programming-language interface——见PLI。

PROM (programmable read-only memory)——可编程只读存储器，器件内部的“或”阵列是可编程的，而“与”阵列不可编程。通常被看作存储器件，其内容可由设计者进行（一次）电编程。也可参考词条PAL、PLA和PLD。

Properties/assertions——见Assertions/properties。

Property-specification language——见PSL。

Pseudorandom——伪随机序列，人为制作但看似随机的序列值，该序列也具有重复性。

PSL (property-specification language)——属性规范语言，为了最高效地描述断言/属性而专门设计的一种形式验证语言。PSL在创建复杂而规则的表达式方面具有强大能力，并且可以使用非常短的代码就描述复杂的行为。这一工业标准语言以IBM公司的Sugar语言为基础，由Accellera (www.accellera.org) 组织控制。也可参考词条OVA、Sugar和SVA。

P-type——P型半导体，其中所掺的杂质能够提供空穴，使得掺杂后的半导体能够吸收电子。

PWB (printed wire board)——印制线路板，电路板的一种，其中的导线被规则地印制在基板的单面或双面，也可能包含内部信号层、电源层和接地层。在欧洲和亚洲主要使用另外一个名称，即PCB。

QFP (quad flat pack)——四角扁平封装，表面安装技术(SMT) 中使用最普遍的一种封装形式，引脚分布在正方形封装的四周。

Quad flat pack——见QFP。

Quantization——(1) 将模拟信号转换为数字信号的一个步骤。首先是对模拟信号定时进行采样，每次采样，模拟信号的整个范围的值被分成一系列离散区间。量化就是决定当前采样值落在哪个离散区间内，也可参考词条Sampling；(2) 将浮点数转换为定点数的过程。

RAM (random-access memory)——随机存取存储器，除非另外说明，RAM是指集成电路形式的半导体器件。

Random-access memory——见RAM。

Read-only memory——见ROM。

Read-write memory——见RWM。

Real estate——指衬底上的可用面积。

Register transfer level——见RTL。

Rising edge——见Positive edge。

ROM (read-only memory)——只读存储器，只能读出数据不能写入数据的存储器件。除非另外说明，ROM是指集成电路形式的半导体器件。

RTL (register transfer level)——寄存器传输级，硬件描述语言(HDL) 是一种专门描述电子电路功能的语言，在使用硬件描述语言描述数字电路时，这种语言可以从不同的抽象级别来获取（描述）电路功能。最简单的抽象级别是门级网表(gate-level netlist)，在这一层次中数字电路的功能被描述成一组相互连接的基本逻辑门（如与门、或门、与非门、或非门等）。更高级的一个抽象级称为寄存器传输级(RTL)，这种情况下，电路被描述成一系列寄存器、布尔等式、控制逻辑（如if-then-else表达式）以及复杂的事件序列（例如，如果时钟信号clock从0变到1，那么将寄存器B和寄存器C所存储的内容相加后存入寄存器A中）。在RTL级描述设计时，最流行的的语言是VHDL和Verilog (SystemVerilog语言也开始流行起来)。

RWM (read-write memory)——读写存储器，随机存取存储器的另外一个（或许是更合适的）名称。

Sampling——采样，将模拟信号转换为数字信号的一个中间过程，实际上就是在特定的时间点得

到模拟信号的值。也可参考词条Quantization。

Schematic——电路原理图。

Sea of cells——单元海，无通道门阵列的通俗叫法。

Sea of gates——门海，无通道门阵列的通俗叫法。

Seed value——初始值，线性反馈移位寄存器（LFSR）或者随机数产生器的初始值。

Semiconductor——半导体，导电性能介于导体和绝缘体之间的一类材料。

Sequential logic——时序逻辑，逻辑输出不仅与当前输入有关，而且还与前一次输入有关。也就是说，逻辑输出依赖于一个输入“序列”。也可参考词条Combinational logic。

Signal conditioning——信号整形，通常是指对模拟信号进行的放大、滤波以及其他处理。

Signature——签名，是指引导式探针功能测试中的CRC校验和。

Signature analysis——签名分析，基于签名的引导式探针功能测试技术。

Silicon chip——硅芯片，尽管有许多半导体材料可供选择，但使用最普遍的还是硅半导体，因此集成电路也常常被称为“硅芯片”，简称“芯片”。

Simple PLD——见SPLD。

Single sided——单面板，只在单面印制导线的印制电路板。

Skin effect——集肤效应，是指导体传输高频信号时，电子仅在导体的外表面（“集肤”）中运动的现象。

Slice——Xilinx公司的术语，是指介于logic cell（LC）和configurable logic block（CLB）之间的一个逻辑实体。写作本书时，一个“slice”包含两个LC。也可参考词条CLB、LAB、LC和LE。

SoC (system on chip)——片上系统或系统芯片，SoC是指一种集成电路，其中不但有硬件部分，还有嵌入式软件部分。几年前，一个电子系统通常由许多集成电路组成，每个集成电路都有自己特定的功能（如微处理器、通信模块、存储器件等）。但是，在今天的许多高端应用中，所有这些功能都可以集成到一个单芯片上，例如ASIC或FPGA，因此叫作片上系统或者系统芯片。

Soft core——软核，在数字电子学中，核(core)通常是指一种规模较大的通用逻辑功能模块，它可以作为一个构造块成为另一个更大的芯片设计的一部分，例如，如果一个ASIC包含一个嵌入式微处理器，那么这个微处理器就被称为“微处理器核”。这类核还包括微控制器核、数字信号处理器核、通讯功能核（如UART）等。这些核可能由设计团队内部开发，但一般都是从第三方IP厂商处购买。

这里所说的软核，其逻辑功能常常用RTL级的VHDL或者Verilog代码进行描述。这种情况下，需要将软核与构成设计的其他模块一起进行综合和布局布线（有时候软核也可能是门级网表或者电路图形式，但这些情况非常少见，尤其是电路图形式更少见）。软核的一个优点是可以由最终用户进行修改，例如，可以删除或修改某些功能。

在ASIC和FPGA两种实现技术中，对于软核的理解会有所不同。在ASIC实现中，假定软核以RTL形式提供，然后与设计的其他RTL代码一起综合成门级网表，再进行布局布线，所得到的结果可以用来生成光掩模，最后利用光掩模来制造硅芯片，这就是说软核的最终实现形式将是硬连线逻辑门（它们本身由晶体管组成）以及它们之间的连线。比较起来，在FPGA设计中所得到的网表则用来生成配置文件，以便对器件内部的查找表（LUT）和可配置逻辑块（CLB）进行编程。

一个设计中可能包含若干个软核和若干个硬核，以及一些用户定义的逻辑块。也可参考词条Hard core。

Soft IP——软IP，在FPGA领域内，软IP是指高级功能的源代码级库，可以被包含在用户设计中，这些功能通常使用硬件描述语言，如Verilog或VHDL在寄存器传输级进行描述。设计人员无论使用什么样的软IP，都会被集成到设计主体（也是RTL级描述）中去，然后这些代码就被综合成一些可编程逻辑块（也可能需要结合一些硬IP，例如乘法器等）。也可参考词条Hard IP 和 Firm IP。

Software——软件，是指由硬件执行的程序或指令序列。

Solder——焊料，一种铅锡合金，具有十分低的熔点，用来焊接其他熔点较高的金属。焊料一般含有60%的锡和40%的铅，铅的含量越高，焊料就越软，熔点也越低；反之，铅含量越低，焊料就越

硬，熔点越高。

Specification-based verification coverage——基于规范的验证覆盖率，对高层次功能或者宏观结构定义作验证时使用的一个衡量表征。这一概念包括设计中的I/O行为，设计能处理的事务类型（包含不同事务类型之间的关系）以及必需的数据转换等。也可参考词条Macroarchitecture definition、Microarchitecture definition和Implementation-level coverage。

SPLD (simple PLD)——简单可编程逻辑器件。最初，所有的PLD都含有一定数量的逻辑门，功能也十分简单，这些器件有PAL，PLA，PROM，GAL，但是，当更复杂的PLD（CPLD）出现后，人们就将比CPLD简单的PLD器件统称为简单可编程逻辑器件，即SPLD。

SRAM (static RAM)——静态RAM，其内部每个单元的核心由4到6个晶体管构成的锁存器(latch)或触发器(flip-flop)组成。之所以称为静态RAM，是因为其中的存储单元一旦被写入数据就会保持不变，除非故意修改所存储的内容或者去掉器件的电源。

Standard cell——标准单元，ASIC的一种形式，与门阵列不同的是，它不用基本单元这个概念，也没有任何预先制作好的元件。ASIC厂商为器件的每一个制造阶段都制作了光掩模，允许客户使用最少数量的晶体管创建各种逻辑功能。

State diagram——状态图，描述状态机的图形方式。

State machine——见FSM。

State variable——状态变量，用来描述状态机各种状态的寄存器，其值代表状态机的当前状态。

Static formal verification——静态形式验证，不用进行仿真就可以检查全部状态空间的形式验证工具。缺点是只能用于整体设计的某些局部，因为如果设计中使用了复杂的属性，其状态空间就会呈指数式急剧增加，很快会陷入“状态空间爆炸”的问题。也可参考词条Formal verification和dynamic formal verification。

Static RAM——见SRAM。

Structured ASIC——结构化ASIC，一种新型的ASIC，就是将预制好的同种模块（或称为片，tile）在器件内部排成阵列形式。这些模块可能含有一些通用逻辑（如逻辑门、多路器或查找表等），以及一个或多个寄存器，也有可能含有局部RAM，由于多个模块混合在片内，因此大多数金属层也是预先定义好的。许多结构化ASIC只需要对两三个金属层进行定制（如果只有一个金属层，就有必要对唯一的一个via层进行定制），这就极大地降低了生产成本，节约了时间，因为不用制作全套的光掩模了。

Sum-of-products——积之和，真值表中所有输出为1的项对应的最小项进行逻辑“或”所得的布尔表达式。

SVA (SystemVerilog Assertions)——Verilog语言中没有断言(assert)这种语句，但是SystemVerilog对此进行了增强，已包含这种功能。此外，Synopsys公司于2002年将其*OpenVera Assertions (OVA)*技术捐献给了负责SystemVerilog的Accellera委员会，此后，SystemVerilog开发者吸取了OVA的优点，并对语法和语义进行了少量修改，在此基础上便形成了SystemVerilog Assertions(SVA)。

Synchronous——(1) 同步信号，是指信号值只在时钟的有效边缘作用下才发生变化；(2) 同步系统，是指系统的所有操作都是在时钟信号的同步下进行的。

Synthesis——见Logic synthesis和Physically aware synthesis。

Synthesizable subset——可综合子集，硬件描述语言，如Verilog和VHDL在最初被构思时，都是以仿真和将想法文档化为目标。但这里存在一个小小的偏差，即对于高抽象层次设计（包含行为级），逻辑仿真器可以很好地工作，而早期的综合工具却最多只能接受RTL级的设计描述，因此，设计工程师就被迫使用所选HDL的一个可综合子集进行设计。也可参考词条HDL和RTL。

System gate——系统门，FPGA厂商为了将其FPGA器件与ASIC器件进行比较就需要建立一个基准，系统门就是在这个过程中遇到的问题之一。例如，如果有一个现成的ASIC设计，包含500 000等效逻辑门，要将这个设计移植到FPGA中，那么怎么才能知道某种特定的FPGA器件是否能“容纳”该设计呢？为了解决这个问题，一些FPGA厂商在20世纪90年代初期开始讨论“系统门”的概念。有些人认

为这是在FPGA设计中使用ASIC术语的可贵尝试，但也有人认为这纯粹是一种市场策略，不会得到任何人的关注。

System on chip——见SoC。

SystemVerilog——一种硬件描述语言，写作本书时，该语言还是一个由Accellera组织(www.accellera.com)管理的开放标准。在本书被翻译成中文版时，该语言已经被IEEE批准为IEEE标准，即IEEE 1800。

SystemVerilog Assertions——见SVA。

Tap——抽头，线性反馈移位寄存器中各个寄存器的输出称为抽头，选择其中一部分进行异或可产生线性反馈移位寄存器的下一个输入数据。

Tera——单位限定词，符号为T，表示 10^{12} 。例如，3THz就是 3×10^{12} Hz。

Tertiary——三进制数字系统。

Tertiary digit——三进制数，常缩写为trit，取值范围为0、1和2。

Tertiary logic——三值逻辑，一种试验中的技术，是指逻辑门的功能基于三种不同的电平，分别表示三进制数0、1、2，它们的逻辑值分别是假(False)、真(True)和可能(Maybe)。

Time of flight——传输时间，信号从一个逻辑门、集成电路或光电子元件传到另一个元件所经历的时间。

Toggle——翻转，逻辑输出或逻辑值由原值跳变到反值。

Trace——见Track。

Track——导线，将电子元件连接起来，也可称为trace或signal。在集成电路中，这些内连线常被称为金属层(metalization)。

Transistor——晶体管，具有三个引脚端的半导体器件，在数字电路中，可以认为它是一个开关。

Tri-state function——三态逻辑，其输出可以有三种状态，即0、1和Z(高阻态)。这种逻辑处于高阻态时不输出任何值，此时可以认为它同电路的其他部分是断开的。

Trit——三进制数，tertiary digit的缩写，取值范围为0、1和2。

Truth table——真值表，将输入和输出按列排成表格，各种输入组合对应不同的输出，全部可能值都写入表内，可以方便地表示数字电路的功能。

TTL (transistor-transistor logic)——晶体管-晶体管逻辑，由双极结型晶体管经过特殊配置实现的逻辑门。

Transistor-transistor logic——见TTL。

UDL/I——流行的硬件描述语言中，Verilog最初是为仿真而设计的，VHDL则是作为一种设计文档和规范语言而创建的，当然也考虑了仿真。最终的结果是这两种语言都可以用来描述可仿真的结构，但不可综合。为了解决这些问题，日本电子工业发展协会(Japan Electronic Industry Development Association, JEIDA)于1990年开发出了自己的硬件描述语言，取名为Unified Design Language for Integrated Circuits(UDL/I)，即集成电路统一设计语言，它的最主要优点是同时考虑了仿真与综合问题。UDL/I的开发环境包括一个仿真器和一个综合工具，它们都可以免费得到(包括源代码)。然而，到UDL/I语言出现的时候，Verilog和VHDL已经在很大的范围内获得了应用，因此，在日本以外的地区，这种语言从来没有引起更多的关注。

μC (microcontroller)——微控制器，使用专门的输入、输出和控制逻辑(如计数器、计时器等)进行优化后的微处理器。

μP (microprocessor)——微处理器，单集成电路上实现的通用计算机(在一组相关的芯片上实现时也可称为芯片组，即chipset)。

ULA (uncommitted logic array)——自由逻辑阵列，门阵列器件最初的名称，现在已经废弃不用。

Uncommitted logic array——见ULA。

Vaporware——雾件，既可能是硬件，也可能是软件，但它只存在于要出售它的人的头脑中。

Verilog——一种硬件描述语言，开始是一家公司的私有财产，后来发展成为IEEE管理下的一项开放标准。

VHDL——美国国防部提出的一种硬件描述语言，现在已经发展成为一个开放的标准。VHDL是VHSIC HDL的简写，而VHSIC又是very high-speed integrated circuit的简写。

Via——过孔，孔壁有可导材料，可以将基材中的两个或更多的可导层连接以实现互通。

VITAL——VHDL语言在数字电路的高抽象级建模方面具有优势，但在sign-off开关级仿真中其时序精度达不到要求。鉴于此，1992年的设计自动化会议（Design Automation Conference，DAC）发起了开发VITAL的动议，VITAL就是VHDL Initiative toward ASIC Libraries的简写，主要是为了增强VHDL语言在ASIC和FPGA设计环境中的时序建模能力。最终的实现包含了ASIC和FPGA基本元件库以及对这些库模型进行时序延迟信息反标的方法。

Volatile——易失性，是指存储器件所存储的数据在掉电后会消失，例如，SRAM或DRAM形式的随机存取存储器。

Word——字，指一组信号，它们共同参与某项任务，并且传输和存储同种数据。例如，计算机数据总线上的数据就可以称为“数据字”。

索引

索引中的页码为英文原书页码，与本书中页边标注的页码一致。

& (与) 31
^ (异或) 31
! (或) 31
! (非) 31
? (无关项) 304

0-In Design Automation (0-In设计自动化公司),
xvi, 118, 205, 334
1076 (IEEE VHDL standard) (国际电子电气工程师协会 VHDL硬件描述语言标准), 167
10-gigabit Ethernet (10吉比特以太网), 357
1364 (IEEE Verilog standard) (国际电子电气工程师协会 Verilog硬件描述语言标准), 166
4004 microprocessor (4004微处理器), 28
4000-series ICs (4000系列集成电路), 27
5400-series ICs (5400系列集成电路), 27
64-bit/66-bit (64b/66b)encoding (64比特/66比特编码), 360
7400-series ICs (7400系列集成电路), 27
8-bit/10-bit (8b/10b)encoding (8B/10B编码),
358

A

ACM (自适应计算机), 388
Actel Corp (ACTEL公司), xvi, 115
Adaptive Computing Machine, 见 ACM
Adder, embedded (嵌入式加法器), 79
Alan Turing (阿兰·图灵), 221
Aldec Inc (Aldec公司), xvi, 118, 215
Algorithms, systolic (算法, 脉动), 67
Altera Corp (Altera公司), xvi, 37, 115, 119
 LAB (逻辑阵列块), 76
 LE (逻辑单元), 75
AMBA (AMBA总线), 241
Amplify (放大), 297
Anadigm Inc. (Anadigm公司), 115
Analog-to-digital (模拟数字转换), 217

Antifuse(s) (反熔丝), 12
-based FPGA (基于……的FPGA), 61, 101
Anti-Miller Effect (反米勒效应), 441
API (应用程序接口), 164
Application (应用程序)
 integrated circuit (专用集成电路), 见ASIC
 standard part (专用标准器件), 见ASSP
Architectural definition (结构定义)
 Macroarchitecture (宏观结构), 193
 Microarchitecture (微观结构), 193
Architecturally-aware design flow (结构化设计流程), 159
Architectures (FPGA) [体系结构 (FPGA)], 57
ARM, 241
 ARM9, 385
ASIC, 2, 42
 gate-level (门级), 180, 181
 cluster-level (团簇级), 183
 RTL-level (RTL级), 184
 cell (单元), 45
 design flow (设计流程)
 HDL-based (基于HDL的), 157
 schematic-based (基于原理图的), 141
full custom (全定制的), 42
gate arrays (门阵列), 44
 channeled (有沟道的), 44
 channeled-less (无沟道的), 44
 sea-of-cells (单元海), 45
 sea-of-gates (门海), 45
standard cell devices (标准单元器件), 46
structured ASIC (结构化ASIC), 47
 -to-FPGA migration (移植到FPGA), 296
ASMBL, 424
Assertion-based verification (基于断言的验证),
 见ABV
Assertion/property coverage (断言/属性覆盖率),
 340
Assisted Technology (断言技术), 41

ASSP, 2

design starts (设计开端), 3

Asynchronous structures (异步结构), 126

Augmented C/C++ based design flow (基于增强的C/C++的设计流程), 205

Automatic test pattern generation (自动测试模式产生), 见ATPG

Axis System (自动系统), xvi, 257

B

Ball grid array, 见BGA

Basic cell (基本单元), 44

Baud rate (波特率), 362

Bell Labs (贝尔实验室), 26

BFM (总线功能模型), 323

BGA (球形栅格阵列), 269

Binary Decision diagrams (二进制决策图), 见BDD

digit (数字二进制决策图), 14

Binit (二进制数字), 15

BIST (内建自测试), 131, 480

Bit (位), 14

file (位文件), 99

Bitstream, 比特流

configuration bitstream (配置比特流), 99

encryption (加密), 61

BJT (双极结型晶体管), 26

Block (框图)

-based design (基于设计的框图), 262

(embedded) RAMs (嵌入式RAM框图), 78

Boolean Algebra (布尔代数), 154

Boundary scan (边界扫描), 112

Branch coverage (分支覆盖率), 339

Bus (总线)

functional model (总线功能模型), 见BFM

interface model (总线接口模型), 见BIM

C

C54xx, 385

C/C++

-based design flows (C/C++为基础的设计流程), 193

augmented C/C++ based (基于增强的

C/C++), 205

pure C/C++ based (基于纯C/C++), 209

SystemC-based (基于SystemC), 198

Model of CPU (CPU模型), 253

Cache logic (高速缓存逻辑), 376

CAD (计算机辅助设计), 44, 141

Cadence Design Systems (Cadence设计公司), xvi, 117, 165, 257

CAE (计算机辅助工程), 140

Carry chains, fast (进位链, 快速), 77

Cell, 见 ASIC cell and Basic cell

Cell library (单元库), 45

Certify (验证), 294

Channeled ASICs (通道式ASIC), 44

Channel-less ASICs (无通道式ASIC), 44

CheckerWare Library (CheckerWare库), 334, 336

Chemical mechanical polishing (化学机械抛光), 见CMP

CIDs (连续相同数值), 360

CLB (可配置逻辑块) 76

Clock

balancing (时钟平衡), 127

domains (时钟域), 127

enabling (时钟使能), 128

gating (门控时钟), 128

managers (时钟管理单元), 85

recovery (时钟恢复), 367

trees (时钟树), 84

cluster-Level SVP (团簇级SVP), 183

clusters/clustering (团簇), 183

CMOS (互补金属氧化物半导体), 26

CMP (化学机械抛光), 320

Coarse-grained (粗粒度), 55, 66, 381

CODEC 218, 422

Code coverage (代码覆盖率), 339, 412

assertion/property coverage (断言/属性覆盖率), 340

branch coverage (分支覆盖率), 339

condition coverage (条件覆盖率), 339

Covered (utility), 412

expression coverage (表达式覆盖率), 339

functional coverage (功能覆盖率), 340

implementation-level coverage (实现级覆盖率), 340

- property/assertion coverage (属性/断言覆盖率),
340
specification-level coverage (规范级覆盖率),
340
state coverage (状态覆盖率), 339
Co-Design Automation, 170
Combinational
 logic (组合逻辑), 31, 71
 loops (组合反馈环), 126
Comma characters/detection (逗号字符), 364
Condition coverage (条件覆盖率), 339
Constraints (formal verification) [约束 (形式验证)], 330
Complementary metal-oxide semiconductor (互补金属氧化物半导体), 见CMOS
Complex PLD see CPLD (复杂可编程序逻辑器件)
Computer-aided design, 见CAD计算机辅助设计 engineering (计算机辅助工程), 见CAE
Configurable (可配置)
 I/O (输入/输出), 90
 impedances (阻抗), 91, 273
 logic analyzer module (逻辑分析器模块), 见 CLAM
 logic block (逻辑块), 见CLB
Configuration (配置)
 bitstream (比特流), 99
 cells (单元), 99
 commands (命令), 99
 data (数据), 99
 file (文件), 99
 modes (模式), 105, 106, 113
 port (端口), 102, 105
Configuring/programming FPGA (配置/编程
 FPGA), 99
 bit file (比特文件), 99
 configuration
 bitstream (配置比特流), 99
 cells (单元), 99
 commands (命令), 99
 data (数据), 99
 file (文件), 99
 modes (模式), 105, 106, 113
 port (端口), 102, 105
 JTAG port (边界扫描端口), 111
parallel load (并行下载)
 (FPGA as master) (FPGA作主端), 108
 (FPGA as slave) (FPGA作从端), 110
serial load 串行下载
 (FPGA as master) (FPGA作主端), 106
 (FPGA as slave) (FPGA作从端), 111
via embedded processor (通过嵌入式处理器),
 113
Confluence, 401
Consecutive identical digits, 见CIDs
Constants(using wisely) (常量), 174
Core (核), 46
 generators (生成器), 290
hard cores (固核), 81, 241
 ARM, 241
 MIPS, 241
 PowerPC, 241
soft cores (软核), 83, 243
voltage (电压), 91
CoreConnect, 241
CoWare, 219, 243
CPLD, 2, 28, 37
CRC (循环冗余校验), 477
Crosstalk (串扰), 430
CUPL, 41, 156
CVS, 409
Cycle-based-simulation (基于周期的仿真), 311
Cyclic redundancy check (循环冗余校验), 见
 CRC

D

- Daisy (菊花链), 141
Data I/O (数据I/O), 41
DCM (数字时钟管理), 85
Debussy, 313, 326
Declarative (声明), 332
Deep submicron (深微米, 见DSM), 58
DEF, 186
Delay, 延迟
 chains (延迟链), 127
formats/models (延迟格式/模型), 306
 3-band delays (三段式延迟), 310
 inertial delays (惯性延迟), 309
 transport delays (传输延迟), 309

- locked loop (延迟锁相环), 见DLL
- Design (设计)**
- capture/entry (graphical) (设计捕获/入口 (图形的)), 161
 - Compiler FPGA (FPGA编译器), 294
 - exchange format (交换格式), 见DEF flows (设计流程)
 - architecturally-aware (结构意识的流程), 159
 - C/C++ based (C/C++为基础的流程), 193
 - augmented C/C++ based (增强C/C++为基础的流程), 205
 - SystemC-based (SystemC为基础的流程), 198
- DSP-based (DSP为基础的流程), 218
- embedded processor-based (嵌入式处理器为基础的流程), 239
- HDL/RTL-based (硬件描述语言/寄存器传输级为基础的流程), 154
- ASIC(early) [集成电路 (早期)], 157
 - FPGA(early) [现场可编程门阵列 (早期)], 158
 - FPGA (early) 143
 - (today) [现场可编程门阵列 (如今)], 151
- DesignPlayer, 338
- Device selection (FPGA) (FPGA器件选择), 343
- Differential pairs (差分对), 354
- Digital (数字的)
- clock manager (数字时钟管理), 见DCM
 - delay-locked loop (延迟锁相环), 见DLL
 - to-analog (数转模), 218
- Distributed (分布式的)
- RAM, 72
 - RC model (分布式RC模型), 450
- DLL, 88, 128
- Domain-specific language (特定范围语言), 见DSL
- DRAM, 21, 28
- DSL, 226
- DSM (深亚微米), 58, 435, 443
- delay effects (延迟效应), 443
- DSP (数字信号处理)
- based design flows (DSP为基础的设计流程), 217
- hardware implementation (硬件实现), 221
- software implementation (软件实现), 219
- DTA (动态时序分析), 321
- Dual-port RAM (双口RAM), 77
- DUT (被测器件), 322
- Dynamic (动态的)
- formal (形式), 329, 335
 - RAM, 见DRAM
 - timing analysis (动态时序分析), 见DTA
- Dynamically reconfigurable 动态重配置
- interconnect (动态重配置互联), 373
 - logic (动态重配置逻辑), 373
- E**
- e (verification language/environment) [e (验证语言/环境)], 325
- Eagles (and jet engines) [鹰 (喷气式发动机)], 99
- ECL (发射极耦合逻辑), 26, 309
- EDGE, 383
- EDIF (电子设计交换格式), 194, 289
- EEPROM (电可编程逻辑器件), 20, 29
- EEPROM (电可擦除只读存储器), 19
- based FPGAs (基于电可擦除只读存储器的FPGA), 64
- EETimes, xv
- Electrically erasable (电可擦除的PLD), 见EEPLD
- programmable read-only memory (可编程只读存储器), 见EEPROM
- Electronic system level (电子系统级), 见ESL
- Embedded (嵌入式的)
- adders (加法器), 79
 - multipliers (乘法器), 79
 - processor (处理器)
 - based design flow (嵌入式处理器为基础的设计流程), 239
 - cores (处理器核), 80
 - hard cores (硬核), 81
 - soft cores (软核), 83
- Emitter-coupled logic (发射级耦合逻辑), 见ECL
- Encoding schemes (编码方案)
- 64-bit/66-bit (64b/66b), 360

- 8-bit/10-bit (8b/10b), 358
 SONET Scrambling, 360
 Encryption (加密), 476
 EPLD (可擦除的可编程逻辑器件), 19, 20
 EPROM (可擦除可编程只读存储器), 17
 Equalization (均衡化), 366
 Equivalency checking (等效性检查), 327
 Equivalent gates (等效门), 95
 Erasable (可擦除的)
 programmable read-only memory (可擦除可编程只读存储器), 见EPROM
 Error-Correcting Codes (book) (错误校正码), 469
 ESL, 246
 Event, 事件
 -driven simulation (事件驱动仿真), 299
 wheel (事件轮), 300
 Events (formal verification) [事件 (形式验证)], 331
 Eye (眼)
 diagrams (眼图), 369
 mask (眼图波罩), 370
- F**
- Fabric (结构、组织), 57
 Fast
 -and-dirty synthesis (快速初级综合), 180
 carry chains (快速进位链), 77
 Fourier Transform (快速傅里叶变换), 见FFT
 Signal Database (快速信号数据库), 见FSDB
 FET (场效应管), 见MOSFET
 FFT (快速傅里叶变换), 68, 389, 399
 Field
 -effect transistor (场效应管), 见MOSFET
 programmable
 analog array (可编程模拟阵列), 见FPAA
 gate array (现场可编程门阵列), 见FPGA
 interconnect, 互联
 chips (现场可编程互联芯片), 见FPIC
 node array (现场可编程节点阵列), 见FPNA
 FIFO (先入先出), 335
 LFSR applications (线性反馈移位寄存器应用), 472
 Fine (精细)
- gained (精细粒度), 54, 66, 381
 -tooth comb (精细梳理), 297
 Fixed-point representations (定点表示法), 229
 FLASH
 -based FPGA (基于FLASH技术的FPGA), 64
 memory (闪存), 20
 PLD, 29
 Flat schematics (平面框图), 148
 Floating
 gate (浮动门), 17
 -point representations (浮点表示法), 228
 unit (浮点运算单元), 见FPU
 Flows, design (流程, 设计)
 architecturally-aware (结构敏感的), 159
 C/C++ based (C/C++为基础的流程), 193
 augmented C/C++ based (增强型C/C++为基础的设计流程), 205
 pure C/C++ based (纯C/C++为基础的设计流程), 209
 System C - based (System C为基础的设计流程), 198
 DSP - based (DSP为基础的设计流程), 218
 embedded processor-based (嵌入式处理器为基础的设计流程), 239
 HDL/RTL-based (硬件描述语言/寄存器传输级为基础的流程), 154
 ASIC(early), 157
 FPGA (early), 158
 schematic - based (原理图为基础的设计流程), 134
 Flying Circus (飞行马戏团), 409
 Formal verification (形式验证), 326, 413
 assertions versus properties (断言对属性), 330
 constraints (约束), 330
 declarative (声明), 332
 dynamic formal (动态形式), 329, 335
 equivalency checking (等效性检查), 327
 procedural (进程), 331
 properties versus assertions (属性对断言), 330
 special languages (专门的语言), 332
 static formal (静态形式), 329, 334
 FORTRAN, 41, 228
 FPAA, 115, 423
 FPGA, 1, 49

- antifuse - based (反熔丝为基础的FPGA), 61, 101
 applications (FPGA的应用), 4
 architectures (FPGA的架构), 57
 bitstream encryption (比特流加密), 61
 CLB (可配置逻辑块), 76
 clock (时钟)
 managers (时钟管理), 85
 trees (时钟树), 84
 configurable I/O (可配置I/O), 90
 impedances (可配置阻抗), 91, 273
 configuring (配置), 99
 bit file (配置比特文件), 99
 configuration (配置)
 bitstream (配置比特流), 99
 cells (配置单元), 99
 commands (配置命令), 99
 data (配置数据), 99
 file (配置文件), 99
 modes (配置模式), 105, 106, 113
 port (配置端口), 102, 105
 JTAG port (边界扫描端口), 111
 parallel load (并行下载)
 serial load (串行下载)
 via embedded, processor (通过嵌入式处理器
 下载), 113
 DCM (数字时钟管理), 85
 design flow, 设计流程
 HDL - based (硬件描述语言为基础的设计
 流程), 158
 schematic-based (原理图为基础的设计流
 程), 143, 151
 device selection (器件选择), 343
 EEPROM - based (基于EEPROM), 64
 EPROM-based (基于EPROM), 64
 Exchange (交换), 271
 FLASH-based (基于FLASH), 64
 future developments (未来的发展), 420
 general-purpose I/O (普通用途的I/O), 90
 gigabit transceivers (吉比特收发设备), 92, 354
 hard cores (硬核), 81
 Hybrid FLASH-SRAM-based (以FLASH和
 SRAM混合为基础的), 65
 I/O (输入/输出), 90
 LUT (查找表), 69, 101
 -based (基于查找表), 69
 mux-based (基于多路器), 68
 security issues (安全考虑), 60
 soft cores (软核), 83
 speeds grades (速度等级), 350
 SRAM-based (SRAM为基础的), 59, 102
 -to-ASIC migration (向ASIC的移植), 294
 -to-FPGA migration (向FPGA的移植), 293
 versus ASIC design styles (与ASIC设计类型的
 比较), 121
 FPIC (现场可编程互连芯片), 374
 FPID (现场可编程互连器件), 374
 FPNA (现场可编程节点阵列), 116, 381
 FPU (浮点运算单元), 397
 FR4 439
 Frequency synthesis (频率合成), 86
 FSDB, 304
 Full custom ASICs (全定制专用集成电路), 42
 Functional
 coverage (功能覆盖), 340
 representations (功能表达), 155
 verification (功能验证), 133
 Fusible links (可熔连线), 10
 Future Design Automation (未来的设计自动化),
 205
- ## G
- Gain-based synthesis (基于增益的合成), 181
 GAL, 36
 Gartner DataQuest, xv
 Gated clocks (门控时钟), 128
 Gate (门),
 Array ASIC (门阵列ASIC), 44
 -level (门级),
 abstraction (门级抽象 (提取)), 154
 netlist (门级网表), 134
 SVP 180, 181
 Gates
 equivalent gates (等效门), 95
 system gates (系统门), 95
 Gateway Design Automation, 163
 General-purpose I/O (通用I/O), 90
 General array logic (通用阵列逻辑), 见GAL
 Geometry (几何学), 58

- George Boole, 154
 Germanium (锗), 26
GHDL, 303
 Gigabit transceivers (吉比特收发器), 92, 354
 clock recovery (时钟恢复), 367
 comma characters / detection (逗号字符/探测), 364
 configurable stuff (可配置元素), 364
 differential pairs (差分对), 354
 encoding schemes (编码方案)
 equalization (均衡化), 366
 eye diagrams (眼图), 369
 ganging multiple blocks (组合多个模块), 362
 jitter (抖动), 369
 pre-emphasis (预加重), 365
 standards (标准), 357
 10-gigabit Ethernet (10吉比特以太网), 357
 Fibre Channel, 357
 InfiniBand, 357
 PCI Express, 357
 RapidIO, 357
 SkyRail, 357
 Giga Test Labs, xvi
 Gilbert Hyatt, 28
 Glitch (瞬时脉冲干扰), 433
 Global reset/initialization (全局复位/初始化), 129
 Glue logic (粘合逻辑), 4
 GNU, 408
 Goering,Richard, xv
 GOLD code generator (GOLD码生成器), 389
 Graphical design entry (图形设计输入), 161
 Granularity (粒度)
 coarse-grained (粗粒度), 55, 66, 381
 fine-gained (细粒度), 54, 66, 381
 medium-gained (中等粒度), 55, 381
 Green Hills Software Inc., 118
 Guided probe (引导式探针), 479
- H**
- Handle - C, 206
 Hard cores (硬核), 81, 241
 ARM, 241
 MIPS, 241
- PowerPC, 241
Hardware
 description language (硬件描述语言), 见HDL
 modeler, 254
 verification language (硬件验证语言), 见HVL
 HDL (硬件描述语言), 153
 RTL (寄存器传输级), 155, 303
 Superlog, 170
 SystemC, 171, 198
 SystemVerilog , 170
 assert statement (assert语句), 336
 UDL/I, 169
 Verilog , 163
 VHDL, 165, 167
 VITAL, 167
 HDL/logic synthesis (硬件描述语言/逻辑综合), 160, 314
 HDL/RTL-based design flow (硬件描述语言/逻辑综合为基础的设计流程), 154
 Hemoglobin (血色素), 432
 Hertz (赫兹), 86
 Hierarchical schematics (分层次电路图), 149
 High-impedance (高阻抗), 304
 HILO logic simulator (HILO逻辑仿真器), 163
 HOL, 416
 Hot (high energy) electron injection [热 (高能) 电子注入], 18
 HVL, 325
 Hyatt,Gilbert, 28
 Hybrid FLASH-SRAM-based FPGAs (基于FLASH和SRAM混合技术的FPGA), 65
- I
- IBIS (versus SPICE), 272
 Icarus, 119
 Verilog , 411
 IDE, 244
 IEEE 1076, 167
 IEEE 1364, 166
 Implementation-level coverage (实现级覆盖率), 340
 Incremental
 design (增量设计), 263
 place-and-route (增量布局和布线), 190

Inertial delay model (惯性延迟模型), 309

InfiniBand, 357

In-place optimization (在现场优化), 见IPO

Instruction set simulator (指令集仿真器), 见ISS

In-system programmable (在系统可编程), 见ISP

Integrated

circuit (集成电路), 见IC

development environment (集成开发环境), 见IDE

Intel, 17, 28

Intellectual property [知识产权 (智力产权)], 见IP

International Research Corporation, 28

Inter-symbol interference (符号间干扰), 见ISI

InTime Software xvi, 185

I/O (输入/输出), 90

IP, 46, 287

core generators (核生成器), 290

ParaCore Architect (ParaCore架构), 397

System Generator 235, 291

firm IP (固核), 94

hard IP (硬核), 93

open source IP (开源IP), 417

source of IP (IP资源), 287

Ipflex Inc., 116, 382

IPO (在现场优化), 185

ISI (符号间干扰), 360

ISP (在系统可编程), 1

ISS (指令集仿真器), 254

Italian Renaissance (意大利文艺复兴), 40

Ivan Sutherland, 182

J

Japan Electronic Industry Development

Association (日本电子产业发展联合会), 见JEIDA

JEDEC, 41

JEIDA, 169

Jelly-bean

devices (器件), 27

logic (逻辑), 1

Jiffy (瞬间), 421

Jitter (抖动), 86, 369

John

Bardeen, 26

Birkner, 41

Wilder Tukey, 14, 15

JTAG, 132, 251

Port, 111

Jurassic, 443

K

Kilby, Jack, 27

L

LAB, 76

Language reference manual (语言参考手册), 见LRM

Latches (锁存器), 129

Latch inference (锁存器推断), 174

Latency, 125

Lattice Semiconductor Corp., 115

Launchbird Design Systems Inc., xvi, 118, 401

LC, 74

LE, 75

LEF, 186

Leopard Logic Inc., 115

Lewis, Carol, xv

LFSR (线性反馈移位寄存器), 389, 465

BIST applications (内建自测试应用), 480

CRC applications (CRC应用), 477

encryption application (加密应用), 476

many-to-one (多到一), 465

maximal length (最大长度), 467

one-to-many (一到多), 469

previous value (前一个值), 475

pseudo-random numbers (伪随机数), 482

seeding (种子值), 470

taps (抽头), 465

Library 库

cell library (单元库), 45

symbol library (符号库), 141

Linear feedback shift register (线性反馈移位寄存器), 见LFSR

Linus Torvalds, 407

Linux, 407

LISP, 408

- Literal, 33
Logic
 analyzers (virtual) [逻辑分析仪（虚拟）], 280
 array block (逻辑阵列块), 见LAB
 cell (逻辑单元), 见LE
 element (逻辑单元), 见LE
 levels (逻辑级), 125
 simulations (逻辑仿真), 134
 cycles-based (基于周期), 311
 event-driven (事件驱动), 299
 HILO, 163
 Verilog-XL, 163
 Synthesis (综合), 160, 314
Logical
 effort (the book), 182
 exchange format (逻辑交换格式), 见LEF
Logic/HDL synthesis (逻辑/硬件描述语言综合), 160, 314
 Lookup table (查找表), 见LUT
 Loops, combinational (反馈环, 组合), 126
 LRM (语言参考手册), 166
 Lumped load model (集总负载模型), 449
 LUT (查找表), 50, 69, 101
 3, 4, 5或6-input (3, 4, 5或6输入), 71
 as distributed RAM (作为分布式RAM), 72
 as shift register (作为移位寄存器), 73
 -based FPGAs (基于查找表的FPGA), 69
- M**
- MAC (乘累加), 80
 Macroarchitecture definition (宏结构定义), 193
 Magama Design Automation, xvi, 182
Magnetic
 RAM, 见MRAM
 tunnel junction (隧道结), 见MJT
 make (utility) (用途), 408
 MandrakeSoft, 410
 Many-to-one LFSRs (多到一的线性移位寄存器), 465
 Mapping (映射), 144
 Marcian “Ted” Hoff, 28
 Mask-programmed devices (掩模可编程序器件), 14
 Mask (掩模), 见photo-mask
 MATLAB, 219, 226
 M-code (M代码), 226
 M-files (M文件), 226
 Maximal length LFSR (最大长度线性移位寄存器), 467
 Mazor, Stan, 28
 MCM (多芯片组件), 82, 241
 M-code, 226
 Medium-grained (中等粒度), 55, 381
 Mentor Graphics Corp., xv, 117, 141, 209, 257
 Metalization layers (金属层), 14, 134
 Metal-oxide semiconductor field-effect Transistor, 见MOSFET
 MetaPRL, 416
 M-files, 226
 Micromatrix, 43
 Micromosaic, 43
 Microarchitecture definition/exploration (微结构定义/探测), 193, 223
 MicroBlaze, 244
 Microprocessor (微处理器), 28
 Micros, 1
 Miller Effect (米勒效应), 438
 MIPS, 241
 Mixed-language (混合语言)
 designs (混合语言设计), 169
 environments/simulation (混合语言环境/仿真), 214, 236, 305
 MJT 磁性隧道结), 23
 Model checking (模型检查), 327
 ModelSim, 215, 306
 Modes, configuration (模式, 配置), 105, 106, 113
 Modular design (模块化设计), 262
 Monolithic Memories Inc., 36, 37
 Monty Python, 409
 Moorby, Phil, 163
 MOSFET (金属氧化物半导体场效应晶体管), 26
 Motorola, 116, 382
 MPEG (运动图像专家组), 383
 MRAM (磁性随机存取存储器), 22, 63, 426
 Multichip module (多芯片组件), 见MCM
 Multipliers ,embedded (乘法器, 嵌入式), 79

Multiply-and-accumulate, 见MAC
 Murphy,Capt.Edward, 169
 Murphy's Law (墨菲法则), 169
 Mux-based FPGAs (基于多路器的FPGA), 68

N

Nadamuni , Daya xv
 Nano, 58
 Negative slack (负slack), 317
 Netlist,gate-level (门级网表)
 Nexar, 257
 Nios, 244
 NMOS, 26
 Noble Peace Prize (诺贝尔和平奖), 98
 Non - recurring engineering (非重现工程学), 见
 NRE
 volatile (非易失性), 14
 Novas Software Inc., 118, 304, 313, 326
 Noyce,Robert, 27
 NRE (非重现工程学), 3
 Nibble (半字), 108
 NuSMV, 406, 415

O

OCI (片内仪器), 280
 OEM (原始设备制造商), 116
 On-chip instrumentation, 见OCI
 One
 -hot encoding (独热编码), 131, 334
 -time programmable (一次可编程), 见OTP
 -to-many LSFR (一到多线性移位寄存器)
 OpenCores, 417
 Open (开)
 Source (源)
 IP (开源IP), 417
 tools (开源工具), 407
 SystemC Initiative, 见OSCI
 Vera Assertions, 见OVA
 Verification Library, 见OVL
 OpenSSH, 410
 OpenSSL, 410
 Original equipment manufacturer, 见OEM
 Origin of FPGA (FPGA的起源), 25

OSCI, 198
 OTP, 1, 12
 Ouroboros, 465
 OVA (开放Vera断言), 336
 OVI (开放Verilog国际联盟), 166
 OVL (开放验证库), 337, 417

P

Packing (封装), 145
 PACT XPP Technologies AG, 116, 382
 PAL (可编程阵列逻辑), 36
 MegaPAL, 37
 PALASM, 41, 156
 ParaCore Architect (ParaCore架构), 397
 Parallel load (并行下载)
 (FPGA as master) (FPGA作为主模式), 108
 (FPGA as slave) (FPGA作为从模式), 110
 Patent (EP0437491-B1)(专利(EP0437491-B1)), 296
 PCB (印制电路板), 239, 267
 PCI, 94
 Express, 357
 Performance analysis (性能分析), 340
 PERL, 409
 PGA (插针栅格阵列), 267
 Phase (相位)
 -locked loop (锁相环), 见PLL
 shifting (相移), 87
 Phil Moorby, 163
 Physically-aware synthesis (物理综合), 161, 314
 Photo-mask (光刻掩模), 14
 PHY, 357
 PicoBlaze, 244
 PicoArray, 384
 PicoChip Designs Ltd ., xvi, 116, 382, 384
 Pilkington, 421
 Alastair, 421
 Microelectronics, 见PMEL
 Pin grid array, 见PGA
 Pipelining (流水线), 122, 123
 wave pipelining (波动流水线), 124
 PLA (可编程逻辑阵列), 33
 Place-and-route (布局布线), 146

incremental (增量), 190
Platform FPGA (FPGA平台), 53
PLD (可编程逻辑器件), 2
 GAL (通用阵列逻辑), 36
 PAL (可编程序阵列逻辑), 36
 PLA (可编程序逻辑阵列), 33
 PROM (可编程序只读存储器), 15, 30
PLI (编程语言接口), 164
 PLL (锁相环), 88, 128
PMEL, 421
PMOS, 26
Point-contact transistor (点接触式晶体管), 26
Positive slack (正slack), 317
PowerPC, 241
Pragma (编译指令), 205, 332
Pragmatic information 见**pragma**
Precision C, 209
Pre-emphasis (预加重), 365
Print circuit board (印制电路板), 见**PCB**
Procedural (进程), 331
Processor cores, embedded (嵌入式处理器核), 80
 hard cores (硬核), 81
 soft cores (软核), 83
Process (technology) node [过程 (工艺) 节点], 58
Product term (乘积项), 33
 sharing (共享), 35
Programmable (可编程的)
 array logic (可编程阵列逻辑), 见**PAL**
 logic
 array (可编程逻辑阵列), 见**PLA**
 device (可编程逻辑器件), 见**PLD**
 read-only memory (可编程只读存储器), 见
 PROM
Programming FPGA (对FPGA编程/配置), 见
 configuring
programming language interface (编程语言接口),
 见**PLI**
PROMELA, 404, 414
Property/assertion coverage (属性/断言覆盖率), 340
Properties versus assertions (属性对断言), 330
Property specification language, 见**PSL**
Pseudo-random numbers (伪随机数), 482

PSL (属性规范语言), 337
Pure
 C/C++ based design flow (基于纯C/C++的设计流程), 209
 LC model (纯LC模型), 450
Python, 405, 409, 413

Q

Q90C1xx, 244
QoR, 159
Quagmire (system gates), 97
Quality-of-Results, 见**QoR**
Quantization (量化), 229
Quartz Logic Corp., 71, 115
QuickSliver Technology Inc., xvi, 116, 382, 388

R

Rad-hard, 62
RAM, 14
 block (embedded) RAM [块 (嵌入式) RAM], 78
 dual-port RAM (双口RAM), 77
 embedded (block) RAM [嵌入式 (块) RAM], 78
 single - port RAM (单口RAM), 77
Random access memory, 见**RAM、DRAM、MRAM和SRAM**
Rapid I/O (快速I/O), 357
RC (可重配置计算), 5, 374
 cache logic (高速缓存逻辑), 376
 dynamically reconfigurable (动态可重配置)
 interconnect (动态可重配置互连线), 373
 logic (动态可重配置逻辑), 373
 virtual hardware (虚拟硬件), 376
RCA, 26, 27
Read-only memory (只读存储器), 见**ROM**
Real-time operating system (实时操作系统), 见
 RTOS
Reconfigurable computing (可重配置计算), 见
 RC
Red Hat (红帽公司), 410
Register transfer level (寄存器传输级), 见**RTL**

- Reincarnation (accidental), 73
 Renaissance (文艺复兴), 40
 Replication (复制), 316
 Resource sharing (资源共享), 130, 175, 222
 Resynthesis (二次综合), 316
 Retiming (时序重调), 316
 Richard Goering, xv
 RLC model (RLC模型), 451
 Robert Noyce, 27
 ROM (只读存储器), 14
 Rossum, Guido Van, 409
 RTL (寄存器传输级), 155, 303
 -level SVP (寄存器传输级硅虚拟原型), 184
 RTOS (实时操作系统), 196, 246
- S**
- SATS (时空分割), 391
 Schematic (s), (原理图)
 -based design flow (基于原理图的设计流程), 134
 ASIC (early) (早期的ASIC原理图), 141
 FPGA (FPGA原理图)
 flat (扁平原理图), 148
 hierarchical (分层次原理图), 149
 SDF (标准延迟格式), 147, 164, 304
 Seamless (无缝的), 257
 Sea-of-cells/gates (单元/门海), 45
 Security issues (安全问题), 60
 Shannon, Claude, 154
 Shockley, William, 26
 SI (信号完整性), 272, 429
 Signal integrity (信号完整性), 见SI
 SignalTap, 281
 Signatures (签名), 480
 Signetics, 41
 SilverC, 394
 Silverware, 394
 Simple PLD, 见SPLD
 Simpod Inc., 254
 Simucad Inc., 118
 Simulation, (仿真)
 cycle-based (基于周期的仿真), 311
 event-driven (事件驱动的仿真), 299
 primitives (仿真原语), 301
- Simulink, 219, 394
 Single-port RAM (单口RAM), 77
 Sirius, 383
 SkyRail, 357
 Slack, 182, 317
 Slice, 75
 Smith, Gray, xv
 SoC, 4
 Soft cores (软核), 83, 243
 MicroBlaze, 244
 Nios, 244
 PicoBlaze, 244
 Q90C1xx, 244
 Software (软件), 15
 SONET Scrambling, 360
 SPARK C-to-VHDL, 209
 Spatial and temporal segmentation, 时空分割,
 见SATS
 Special formal verification languages (专用形式
 验证语言), 332
 OVA, 336
 PSL, 337
 Sugar, 336
 Specman Elite, 326
 SPEEDCompiler, 338
 Speed grades (FPGA) ((FPGA)速度等级), 350
 SPICE (versus IBIS), 272
 SPIN (model checker), [SPIN (模型检测)], 406, 414
 SPLD, 2, 28
 Sproull, Bob, 182
 SRAM (静态随机存储器), 21, 28
 -based FPGAs (基于SRAM的FPGA), 59, 102
 first SRAM (第一个SRAM), 28
 SSTA (统计静态时序分析), 319
 STA (静态时序分析), 147, 306, 319
 Standard
 cell ASICs (标准单元ASIC), 46
 delay format (标准延迟格式), 见SDF
 Stan Mazor, 28
 State
 coverage (状态覆盖), 339
 machine encoding (状态机编码), 131
 one-hot (独热状态), 131
 Static

- formal (静态格式), 329, 334
 RAM, 见SRAM
 timing analysis (静态时序分析), 见STA
 Statistical static timing analysis (统计静态时序分析), 见SSTA
 Stephen Williams, 411
 Stephen Hofstein, 26
 Stripe, The, 81
 Structural representations (结构化表示), 155
 Structured ASICs (结构化ASIC), 47
 Sugar, 336
 Sum-of-products (积之和), 33
 Superlog, 170
 Sutherland, Ivan, 182
 Switch-level (开关级), 154
 Symbol library (符号库), 141
 Symbol (in data transmission) (数据传输中的符号), 360
 Synopsys Inc., xvi, 117, 294
 Synthesis (综合)
 HDL/logic (HDL/逻辑综合), 160, 314
 physically-aware (物理综合), 161, 314
 replication (综合复制), 316
 resynthesis (重综合), 316
 retiming (重定时), 316
 System
 gates (系统门), 95
 Generator (系统生成器), 235, 291
 HILO, 310
 -level (系统级)
 design environments (系统级设计环境), 227
 representations (系统级表示), 156
 -on-Chip (片上系统), 见SoC
 SystemC, 171, 198
 -based design flow (基于SystemC的设计流程), 198
 model of CPU (CPU的SystemC模型), 253
 System Verilog, 170
 assert statement (System Verilog断言语句), 336
 Systolic algorithms (收缩算法), 67
- T
- TDM (时分复用), 130
 TDMA (时分多址), 383
 Technology node (技术结点), 58
 Tenison Technology Ltd., 338, 412
 Tertiary logic, 304, 325
 Texas Instruments, 27
 The Mathworks Inc., xvi, 118, 219
 Timed C domain, 214
 Time-division
 multiple access, 见TDMA
 multiplexing, 见TDM
 Timing analysis/verification (时序分析/验证), 133
 dynamic timing analysis (动态时序分析), 见DTA
 static timing analysis (静态时序分析), 见STA
 Transistor (晶体管), 26
 bipolar junction transistor (双结型晶体管), 见BJT
 field-effect transistor (场效应管), 见MOSFET
 -transistor logic (晶体管-晶体管逻辑), 见TTL
 Transmission line effect (传输线效应), 441
 Transport delay model (传输延迟模型), 309
 Triple redundancy design (三重冗余设计), 62
 Tri-state buffers (三态缓冲), 176
 Trit (三极), 304
 TTL (晶体管-晶体管逻辑), 26, 309
 Tukey, John Wilder, 14, 15
 Turing
 -complete (图灵完整性), 389
 Machine (图灵机), 221
- U
- UDL/I, 169
 UDSM, 58, 435, 443
 delay effects (延迟效果), 443
 ULA, 44
 Ultradeep submicron, 见UDSM
 Ultraviolet (紫外线), 见UV
 Uncommitted logic array, 见ULA
 Untimed C domain (不定时C域), 214
 UV, 19
 Taps (抽头), 465

V

- Valid (有效性), 141
 Value change dump (值变化存储), 见VCD
 VCD, 304, 326, 411
 Verification (验证)
 environments (验证环境), 324
 e (e验证), 325
 OpenVera, 336
 Vera, 336
 formal (形式验证), 见formal verification
 functional (功能验证), 133
 IP (IP验证), 322
 Reuse (复用验证), 329
 timing (时序验证), 133
 Verilog
 Icarus Verilog, 411
 OVI, 166
 the language (语言), 163
 the simulator (仿真器), 163
 Verilog 2001(2K1), 167
 Verilog 2005, 167
 Verilog 95, 167
 Verilog-XL, 163
 Verisity Design Inc, xvi, 118, 325
 VHDL, 165, 167
 International, 170
 VITAL, 167
 VHSIC (甚高速集成电路), 167
 VI, 161, 408
 Virtual, 虚拟
 hardware (虚拟硬件), 376
 logic analyzers (虚拟逻辑分析), 280
 Machine Works (虚拟机工作), 282
 VirtualWires, 282
 Visibility into the design (设计可见性), 250, 277
 multiplexing (多路复用), 278
 special circuitry (专用电路), 280
 virtual logic analyzers (虚拟逻辑分析), 280
 VirtualWires, 282

W

- Visual interface, 见 VI
 VITAL, 167
 Volatile, 14
 VTOC, 338, 412
W
 Walsh code generator (Walsh 代码生成器), 389
 Walter Brattain, 26
 Wave pipelining (波浪流水线), 124
 W-CDMA, 383
 Wideband code division multiple access (宽带码分多址存取), 见W-CDMA
 William Shockley, 26
 Williams Stephen, 411
 Wind River System Inc, 118
 Work functions (操作函数), 186
 Wortsel Grinder Mark 4, 121
 Wrapper (node) [封装 (结点)], 389

X

- X (unknown) [X (未知量)], 304
 XAUI, 363
 Xblue architecture (XBlue 架构), 425
 Xilinx Inc, xv, 25, 115, 119, 235, 424
 CLB, 76
 DCM, 85
 LC, 74
 slice, 75
 XM Radio, 383
 XoC, 257

Y

- Years, FPGA years (年, FPGA年), 98

Z

- Z (high-impedance) (高阻抗), 3~4

The Design Warrior's Guide to FPGAs

Devices, Tools and Flows

FPGA设计指南 器件、工具和流程

本书涵盖了你想了解的关于FPGA的一切！

——Richard Goering, *EE Times* 杂志主编

……一部必读之作……可读性极强，而且提供了深入和丰富的信息。

——EDN 杂志

自20世纪80年代中期诞生以来，FPGA技术因为成本较低、使用灵活，展现出蓬勃的发展态势，有望成为主流的电路设计解决方案。其应用范围遍及通信、消费电子、汽车电子、工业控制等领域。

本书是FPGA领域最负盛名的入门教程之一，超越了一般的同类图书。书中用简明扼要、通俗易懂的语言向读者阐述了FPGA的基本概念、工作原理、编程方法和FPGA设计中遇到的各种知识、器件、工具和流程。在此基础上，作者总结自己多年从业经验，通过许多实例讨论了实际设计中丰富的技术细节。

本书特色

- 内容全面，几乎涵盖所有FPGA相关知识
- 语言简练诙谐，带领读者轻松走进技术核心
- 示例、图解详尽，贴近实战，便于读者理解

Clive “Max” Maxfield 世界半导体设计界知名专家，FPGA专业网站Programmable Logic DesignLine (www.pldesignline.com) 主编。Maxfield先生拥有丰富的电路设计和开发经验，更以杰出的写作才能享誉全球。他曾经长期为EDN、EE Design等一流杂志和网站撰写专栏，并撰写了多部电子技术方面的畅销书。读者可以通过他的个人网站www.techbites.com/team-max.html和博客www.pldesignline.com/blogs/与他联系。

本书译自原版 *The Design
Warrior's Guide to FPGAs*,
并由Elsevier授权出版。



本书相关信息请访问：[图灵网站](http://www.turingbook.com) <http://www.turingbook.com>
读者/作者热线：(010) 88593802
反馈/投稿/推荐信箱：contact@turingbook.com

分类建议 电子电气/电子技术

人民邮电出版社网址 www.ptpress.com.cn

ISBN 978-7-115-16862-7



9 787115 168627 >

ISBN 978-7-115-16862-7/TN

定价：49.00 元