电子设计自动化C 实现、电路设计和工艺技术

编辑们谨向电子设计自动化 (EDA) 领域的无名英雄致谢，他们除了个人、公司或学术议程之外，还致力于推动该领域的发展。 这些男人和女人以多种方式提供服务——他们主持小型会议、编辑技术期刊、在标准委员会任职等等。 这些主要是志愿工作不会让任何人变得富有或出名，尽管投入了时间和精力，但它们确实为 EDA 的显着和持续发展做出了巨大贡献。 我们向这些人表示敬意，他们没有得到应有的荣誉。

目录

第 i 部分 — RtL 到 GDSii，或综合、布局和布线

作者介绍：
Luciano Lavagno 于 1992 年从加州大学伯克利分校（加利福尼亚州伯克利）获得 EECS 博士学位，并于 1993 年从意大利都灵理工大学获得 EECS 博士学位。他是两本关于异步电路设计的书和一本关于硬件的书的合著者 /嵌入式系统的软件协同设计，以及 200 多篇科学论文。 他是 15 项美国专利的发明者。 1993 年至 2000 年间，他是 POLIS 项目的架构师，该项目是加州大学伯克利分校、Cadence Design Systems、Magneti Marelli 和都灵理工大学之间的合作项目，该项目开发了一个完整的硬件/软件协同设计环境，用于控制 主导嵌入式系统。 2003年至2014年间，他是Cadence C-to-Silicon高级综合系统的创建者和架构师之一。
自2011年起，他担任意大利都灵理工大学教授。 他一直在该领域的多个国际会议（例如 DAC、DATE、ICCAD、ICCD、ASYNC、CODES）以及各种研讨会和专题讨论会的技术委员会中任职。 他曾担任 DAC 和 TPC 的技术项目主席以及 CODES 的总主席。 他曾担任 IEEE TCAS 和 ACM TECS 的副主编。 他是 IEEE 的高级会员。
他的研究兴趣包括异步低功耗电路的综合、混合硬件和软件嵌入式系统的并行设计、数字电路的高级综合、无线传感器网络的硬件组件和协议的设计和优化。 ，以及 WSN 的设计工具。

伊戈尔·L·马尔科夫目前正在密歇根州安娜堡市的密歇根大学休假，他曾在那里任教多年。 他于 2014 年加入 Google。他还在斯坦福大学教授 VLSI 设计。 他研究制造计算机的计算机，包括电子设计自动化、安全硬件和新兴技术的算法和优化技术。 他是 IEEE 院士和 ACM 杰出科学家。 他与人合着了五本书，拥有四项美国专利和 200 多篇审稿出版物，其中一些出版物荣获最佳论文奖。 马尔科夫教授是 DAC 奖学金、ACM SIGDA 杰出新教师奖、NSF 职业奖、IBM 合作伙伴奖、微软 A. 理查德·牛顿突破性研究奖和首届 IEEE CEDA 早期职业奖的获得者。 在 2011 年重新设计 ACM 计算分类系统期间，Markov 教授领导了硬件树方面的工作。 在他的指导下完成了十二篇博士论文的答辩； 其中三人获得优秀论文奖。

Grant E. Martin 是加利福尼亚州圣何塞 Cadence Design Systems Inc. 的一位杰出工程师。 在此之前，格兰特在苏格兰巴勒斯工作了6年； 加拿大北电/ BNR 10年； 在 Cadence Design Systems 工作了 9 年，最终成为其实验室的 Cadence 研究员； 和 Tensilica 合作了 9 年。 2013 年，Cadence 收购了 Tensilica，他重新加入 Cadence，并一直在 Cadence IP 集团的 Tensilica 部门工作。 他于 1977 年和 1978 年在加拿大滑铁卢大学获得数学（组合和优化）学士和硕士学位。

Grant 是《Surviving the SoC Revolution: A Guide to Platform-Based Design》（1999 年）和《System Design with SystemC》（2002 年）的合著者，也是《Winning the SoC Revolution: Experiences in Real Design》和《UML for Real: Design of Real》两书的合编者。 嵌入式实时系统，2003 年 6 月，全部由 Springer 出版（最初由 Kluwer）。 2004 年，他与 Vladimir Nemudrov 共同撰写了第一本俄语版的 SoC 设计书籍，由莫斯科 Technosphera 出版。 在过去十年中，他与人合编了《数字系统开发和验证分类法》（Springer，2005 年）和《SoC 设计 UML》（Springer，2005 年），并在本世纪末与人合着了《ESL 设计和验证：处方》 电子系统级方法论（Elsevier Morgan Kaufmann，2007）和 ESL 模型及其应用：电子系统级设计实践（Springer，2010）。

他还发表了许多论文、演讲和教程，并参加了许多主要会议的小组讨论。 他于 2001 年夏天担任 VSI 联盟嵌入式系统研究组的联合主席，并担任 2005 年和 2006 年 DAC 方法技术计划委员会的联合主席。他还是 Springer 嵌入式系统系列的联合编辑。 他特别感兴趣的领域包括系统级设计、基于 IP 的片上系统设计、基于平台的设计、DSP、基带和图像处理以及嵌入式软件。 他是 IEEE 的高级会员。

Louis K. Scheffer 于 1974 年和 1975 年在加利福尼亚州帕萨迪纳的加州理工学院获得学士和硕士学位，并于 1984 年在加利福尼亚州帕洛阿尔托的斯坦福大学获得博士学位。1975 年至 1981 年，他在惠普工作，担任 芯片设计师和 CAD 工具开发人员。 1981 年，他加入 Valid Logic Systems，负责硬件设计、开发原理图编辑器并构建 IC 布局、布线和验证系统。 1991 年，Valid 与 Cadence Design Systems 合并，此后 Scheffer 博士致力于布局和布线、布局规划系统以及信号完整性问题，直到 2008 年。

2008年，谢弗博士将研究领域转向神经生物学，通过使用电子显微镜图像重建大脑回路来研究大脑的结构和功能。 由于 EDA 不再是他的日常必需品（尽管他的研究使用了许多源自 EDA 的算法），因此他非常感谢 Igor Markov 承担了这部分书籍的工作。 Lou 还对搜寻地外文明 (SETI) 感兴趣，在 SETI 研究所艾伦望远镜阵列的技术顾问委员会任职，并且是《SETI-2020》一书的合著者，此外还发表了该领域的多篇技术文章 。

RTL 到 GDSII，或综合、布局和布线

第一版序言

电子设计自动化 (EDA) 是工程艺术领域的巨大成功。 在过去的四分之一个世纪中，改进的工具已将设计人员的生产力提高了一千多倍。 如果没有 EDA，摩尔定律将仍然是无用的好奇心。 如果没有这些复杂的工具，就无法设计或调试任何一个十亿晶体管芯片，因此如果没有 EDA，我们就不会有笔记本电脑、手机、视频游戏或任何我们认为理所当然的其他电子设备。

在构建更大芯片能力的推动下，EDA 开发人员基本上跟上了步伐，这些巨大的芯片仍然可以设计、调试和测试，事实上，上市时间也在缩短。

EDA 的故事比集成电路 (IC) 制造的进步复杂得多，后者基于关键尺寸的简单物理缩放。 另一方面，EDA 通过一系列范式转变而发展。 本书共49章，每一章在几十年前都还只是某些专家眼中的闪光。 然后它成为一个研究课题，然后成为一种学术工具，然后成为一两家初创企业的焦点。 几年之内，它就得到了大型商业 EDA 供应商的支持，现在已经成为传统观念的一部分。 尽管用户总是抱怨今天的工具不太适合今天的设计，但生产力的总体提高是显着的。 毕竟，还有哪些领域会像 1999 年的国际半导体技术路线图那样，抱怨三十年来生产率的复合年增长率仅为 21%?

EDA 工具的未来是什么? 当我们审视 2005-2006 年期间电子和集成电路设计的状况时，我们发现我们可能很快就会进入该学科的一个重大变革时期。 集成电路的经典缩放方法在过去 401 年中跨越了器件尺寸的多个数量级，看起来只能持续几代或工艺节点（尽管这在过去已经被争论过很多次，并且总是如此） 事实证明这一预测过于悲观）。 传统的晶体管和布线很可能会被新的纳米和生物技术所取代，而我们目前才刚刚开始尝试这些技术。 这一深刻的变化肯定会对用于设计集成电路的工具和方法产生相当大的影响。 我们应该花精力研究 CAD 以实现这些未来技术，还是继续改进我们当前使用的工具?

经过进一步考虑，很明显当前的 EDA 方法还有很大的生命力。 与在当前设计方法的发展还剩下至少十年的时间，而且还有数十万或数百万种设计必须制作新的 IC 或使用它们的可编程版本，因此忘记当今的 EDA 方法还为时过早。 即使技术改变为全新的形式和结构，当今的许多 EDA 概念也将被重用并发展为远远超出当前范围和思维的设计技术。

IC 的 EDA 领域已经发展到远远超出了任何一个人可以掌握全部，甚至可以了解所有方面进展的程度。 因此，迫切需要为这个极其广泛和多样化的主题创建一个快照。 学生需要一种学习当今广泛使用的设计工具所涉及的许多学科和主题的方法。 随着设计变得越来越跨学科，电子设计师和 EDA 工具开发人员需要扩大他们的范围。 一个子主题中使用的方法很可能适用于新主题的出现。 所有电子设计都可以利用该领域的综合参考文献。

考虑到这一点，我们邀请了来自 EDA 所涉及的所有学科的许多专家来贡献章节，总结并全面概述他们的特定主题或领域。 可以理解，这些写于 2004 年至 2005 年的章节代表了现有技术的一个快照。 然而，作为调查和概述，它们保留了持久的教育和参考价值，对于未来许多年的学生和从业者来说都是有用的。

由于要涵盖大量主题，我们决定将手册分成两本书。 《IC 系统设计、验证和测试的电子设计自动化》是第一卷，涵盖系统级设计、微架构设计以及验证和测试。 IC 实现、电路设计和工艺技术的电子设计自动化是第 2 卷，涵盖了经典的“RTL 到 GDSII”设计流程，包括综合、布局和布线，以及模拟和混合信号设计、物理验证、分析和设计。 提取和技术 CAD 主题。 这些大致对应于 IC 设计中经典的“前端/后端”划分，其中前端（或逻辑设计）侧重于确保设计做正确的事情（假设可以实现），而后端 （或物理设计）集中于生成所需的详细工具，同时采用给定的逻辑功能。 尽管存在局限性，这种分离多年来一直存在——独立于实现的完整且正确的逻辑设计仍然是 IC 设计流程的两个主要部分之间的绝佳切换点。由于 IC 设计人员和 EDA 开发人员经常关注逻辑/物理划分的一侧，因此这似乎也是划分本书的好地方。

本卷《IC 实现、电路设计和工艺技术的电子设计自动化》首先概述了经典的 RTL 到 GDSII 设计流程，然后立即进入“综合、布局和布线”的逻辑综合方面。 功率分析和优化方法在流程的几个阶段重复出现。 最近，标准 IC 流程中的等效性检查变得更加可靠和自动化。 然后，我们将看到有关布局和布线的章节以及静态时序分析和结构化数字设计的相关主题。 标准后端流程依赖于标准数字库和设计数据库，并且必须生成能够很好地融入封装、电路板和混合电路的 IC 设计。 相对较新的对设计闭合的强调将流程的许多方面结合在一起。 事实上，关于设计收敛的第 13 章非常适合在关于设计流程的第 1 章之后阅读。

在深入探讨模拟和混合信号设计领域之前，该手册先介绍了适用于 FPGA 设计的特殊方法——这是一个使用底层固定但可重新编程平台进行快速 IC 设计的不断增长的领域。 然后我们转向模拟设计，其中涵盖模拟方法、高级建模和布局工具。 物理验证、分析和提取涵盖设计规则检查设计的可制造性转换、电源噪声和其他噪声问题的分析以及布局提取。 最后，该手册将工艺模拟、器件建模以及高级寄生参数提取视为 IC CAD 技术的各个方面。

对于所有参与并有兴趣学习电子设计及其相关工具和方法的人来说，这本手册及其两本书构成了一本有价值的学习和参考书。 我们希望所有读者都会对它感兴趣，并希望它将成为一个受欢迎的资源。

®

路易斯·K·谢弗 卢西亚诺·拉瓦尼奥 格兰特·E·马丁

是 MathWorks, Inc. 的注册商标。

MATLAB

如需了解产品信息，请联系：

MathWorks 公司

苹果山大道 3 号

内蒂克，MA 01760-2098 美国电话：508-647-7000

传真：508-647-7001

电子邮件：info@mathworks.com 网址：www.mathworks.com

设计流程

1.1 设计流程的演变

规模化推动数字集成电路 (IC) 实现从主要使用独立综合、布局和布线算法的设计流程转变为用于设计收敛的集成构建和分析流程。 本章将概述互连延迟上升的挑战如何催生出一种新的思考和集成设计收敛工具的方法（参见第 13 章）。 功耗、可布线性、可变性、可靠性、不断增加的设计尺寸以及不断增加的分析复杂性等规模挑战将继续挑战设计收敛的当前技术水平。

现代电子设计自动化 (EDA) 流程从使用寄存器传输级 (RTL) 语言（例如 Verilog 或 VHDL）对设计进行高级描述开始，通过抽象电路实现问题来减少设计工作。 自动化工具将标准单元库中的 RTL 与定制设计的宏单元合成为逻辑门，将逻辑门放置在平面图上，并布线连接它们。 组成电路的各种扩散层、多晶硅层和金属层的布局在 GDSII 数据库格式中指定，以供制造。

RTL 到 GDSII 的流程在过去 30 年中发生了重大变化。 CMOS 技术的持续扩展显着改变了各个设计步骤的目标。 由于缺乏良好的延迟预测因素，导致最近的设计流程发生了重大变化。 在本章中，我们将描述是什么推动了设计流程从一组单独的设计步骤转变为完全集成的方法，以及我们看到的为应对最新挑战而将发生的进一步变化。

与 Alberto Sangiovanni-Vincentelli 在《EDA 浪潮》[1] 中确定的 EDA 时代类似，我们区分了 RTL 到 GDSII 计算机辅助设计流程发展的三个主要时代：发明时代、 实施时代和整合时代。 在发明时代，发明了逻辑综合、布局、布线和静态时序分析。 在实施时代，通过设计复杂的数据结构和先进的算法，它们得到了极大的改进。 这使得每个设计步骤中的软件工具都能跟上快速增加的设计尺寸。 然而，由于缺乏良好的预测成本函数，无论每个步骤的实施效率如何，都无法通过一组离散步骤来执行设计流程，需要通过流程进行多次迭代才能完成设计。 这导致了集成时代的到来，其中大多数设计步骤都在集成环境中执行，由一组增量成本分析器驱动。

让我们更详细地了解每个时代，并描述它们的一些特征以及 EDA 设计流程中步骤的变化。

1.2 发明时代

早期，发明了布线、布局、时序分析和综合的基本算法。 大多数物理设计算法的早期发明都是由封装和电路板设计驱动的。 空间非常宝贵，只有少数布线层可用，而且引脚也有限。 需要放置的分立元件相对较少。 高复杂度的优化算法不是问题，因为我们处理的组件很少。

在这个时代，发明了基本的分区、布局和路由算法。 物理设计流程中的一个基本步骤是将电路细分为更小的部分，以实现

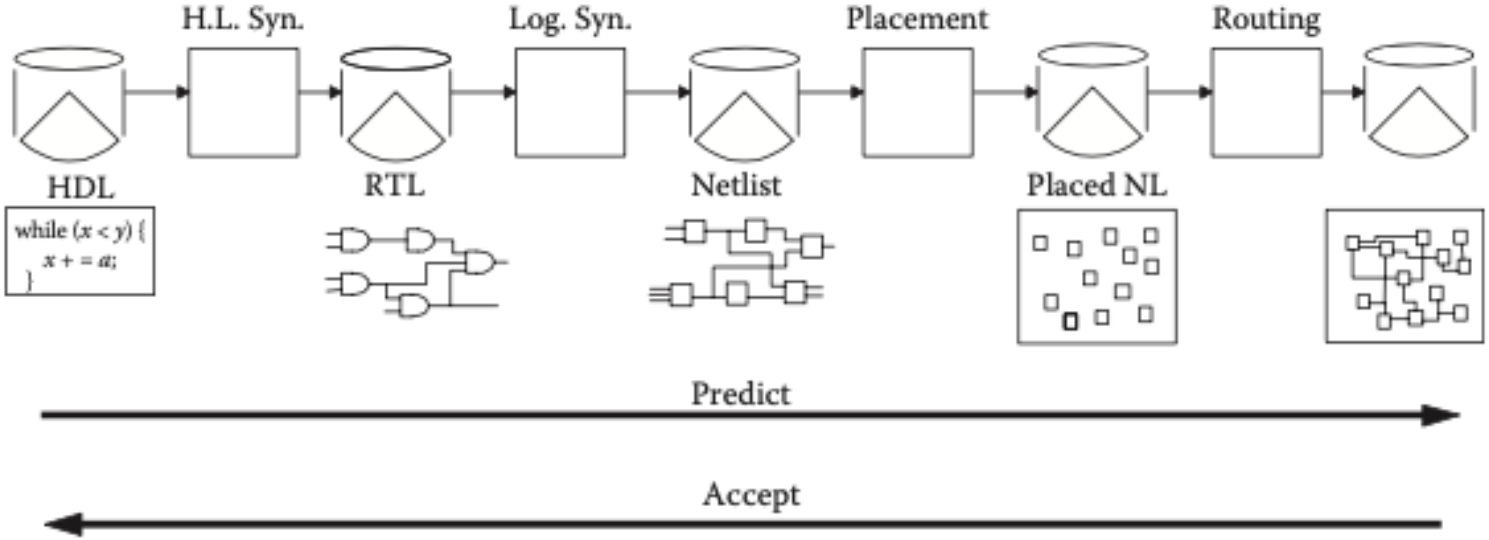
简化平面图和布局。 最大限度地减少电路分区之间的导线交叉非常重要，这样可以专注于分区内更快的局部优化，减少分区之间的限制，并最大限度地减少有限全局布线资源的使用。 1970 年，Kernighan 和 Lin [2] 开发了一种最小割分区启发式方法，将电路划分为两个相等的门组，并在组之间交叉的网络数量最少。 模拟退火 [3] 算法率先用于布局，并允许部署广泛的优化标准。 发明了通道、开关盒和迷宫路由[4]的基本算法。 通过利用受限的拓扑和设计尺寸，可以设计最佳算法来处理这些特殊情况

1.3 理论实施

随着 IC 的出现，越来越多的注意力转移到设计自动化算法来处理它们，而不是电路板。 传统的 CMOS 缩放使这些设计的尺寸增长得非常快。 随着设计规模的扩大，设计工具的实现对于跟上日益庞大的设计并控制设计时间变得极其重要。 新的实现和数据结构被开创，最有效扩展的算法成为标准。

随着设计尺寸开始增大，新的抽象层被发明。 标准单元的发明允许人们将单元的详细物理实现与布局和布线算法看到的足迹图像分开。 随着标准单元概念的引入，布线、布局以及后来的综合算法的大规模应用开始兴起。

标准单元的发明可以与印刷机的发明相比较。 虽然以前就知道手工书籍写作，但这是一个劳动密集型的过程。 通过保持字母的高度固定并让每个字母的底部宽度根据字母的大小而变化，实现了印刷开发的显着自动化。 类似地，在标准单元专用 IC (ASIC) 设计中，人们使用常见高度但宽度不同的标准单元，具体取决于单个标准单元的复杂性。 这些库



（第 12 章）创建了显着水平的标准化并实现了很大程度上的自动化。

第一个门阵列的发明将标准化提升到了一个更高的水平。

这种标准化催生了 ASIC 商业模式，为自动化工具创造了巨大的市场机会，并催生了许多创新。 逻辑综合 [5] 的发明是为了弥补语言描述与这些标准单元实现之间的差距。

在实施时代，设计流程可以由一系列离散步骤粘贴在一起（见图 1.1）。 RTL 逻辑综合转换 Verilog 或 VHDL 描述，对网表执行与技术无关的优化，然后将其映射到具有技术门的网表，然后进行进一步的技术相关的门级网表优化。 门和路由将它们连接在一起。 最后，使用时序模拟器使用有限数量的提取接线寄生电容数据来验证时序。

在这个时代，数字电路的尺寸持续快速增长，根据摩尔定律，尺寸每两年翻一番[6]。 逻辑综合允许从高级 RTL 设计中快速创建包含数百万个门的网表。 IC 的门数已从 1984 年的 40,000 个门增长到 2000 年的 40,000,000 个门，再到 2014 年的十亿门设计。

像四叉树 [7] 和 R 树 [8] 这样的新数据结构允许在几何空间中进行非常有效的搜索。 布尔决策图 [9] 的应用可以在更大的逻辑分区上实现高效的布尔推理。

在实现分区算法方面取得了很大进展。 Fiduccia 和 Mattheyses [10] 给出了 Kernighan 和 Lin 分区算法的更有效版本。他们使用一种特定的算法来选择要穿过切割的顶点，从而节省了运行时间，并允许处理不平衡的分区和不均匀的边权

重。 使用谱方法[11]的实现被证明对于某些问题非常有效。 Yang [12]通过迭代应用网络流算法证明了优于上述两种方法的结果。

优化二次线长度成为布局的圣杯。 二次算法通过部署高效的二次优化算法并与各种类型的划分方案混合来充分利用这一点[13]。

逻辑综合中的原始技术，例如内核和立方体分解，一次应用于网络的小分区。 更有效的算法，如基于测试生成的全局流[14]和冗余去除[15]，可以应用于更大的设计。 通过时序仿真完全覆盖所有时序路径变得太不切实际，因为它对设计尺寸的指数依赖性，并且基于[17]中的早期工作发明了静态时序分析[16]。

TABLE 1.1 Gate Delays

Gate	Logical Effort	Intrinsic Delay	FO4 Delay
INV	1.00	1.00	5.00
NAND2	1.18	1.34	6.06
NAND3	1.39	2.01	7.57
NAND4	1.62	2.36	8.84
NOR2	1.54	1.83	7.99
NOR3	2.08	2.78	11.10
NOR4	2.63	3.53	14.05

随着更大的设计出现更多的布线层，允许单元上布线以及单元内和单元间布线区域的共享。 网格布线抽象与标准单元模板很好地匹配，并成为大幅提高布线速度的基础。 诸如全局路由、交换机盒和区域路由之类的分层路由抽象被首创，作为解耦路由问题的有效方法。

适用于无需分区的大规模设计的算法的复杂度必须小于 O(n2)，并且最好不大于 O(n log n)。 这些使用上述数据结构和算法的进步来解决复杂性，使设计工具能够处理大型的实际问题。 然而，为此类算法找到合适的成本函数变得越来越困难。 在设计流程的早期准确预测物理效应变得更加困难。

让我们讨论在实施期间重要设计指标的预测如何随着时间的推移而演变。 在实施时代的初期，大多数重要的设计指标（例如面积和延迟）都很容易预测。 （性能、功耗和面积是当今设计的相应关键指标。）每个离散设计步骤中的优化算法均由依赖于这些预测的目标函数指导。 只要能够非常准确地预测这些指标的最终值，RTL 到 GDSII 流程确实可以按一系列相当独立的步骤执行。 然而，重要设计指标的预测变得越来越困难。 正如我们将在以下部分中看到的，这导致了设计收敛流程的根本性变化。 简单的设计步骤顺序已经不够了。

让我们更详细地看看一类预测函数，即电路延迟的估计。 在早期技术中，沿路径的延迟主要由门的延迟决定。 此外，大多数门的延迟非常相似。 这样一来，只要知道关键路径上有多少个门，就可以通过计算一条路径上的门的层数并乘以典型的门延迟来合理地预测该路径的延迟。 因此，一旦逻辑综合确定了每条路径上的逻辑级数，就可以知道电路的延迟。 事实上，在早期的时序优化中，使用多个门尺寸来保持不同负载下的延迟合理恒定，而不是实际改善门的延迟。 在映射到依赖于技术的标准单元之后，该区域可以合理地通过将细胞面积相加可以很好地预测。 随后的布局或布线步骤都不会显着改变这两个数量。 在那个时代，电力和噪音并不是很受关注。

在较新的库中，具有不同逻辑复杂性的门的延迟开始显着变化。 表 1.1 显示了不同类型门的相对延迟。 逻辑努力特性表明门的延迟如何随负载而增加，而固有延迟是门延迟的与负载无关的贡献[18]。 该表的第四列显示，更复杂的 NOR4 逻辑门的扇出 4 (FO4) 延迟可能是简单反相器延迟的三倍。 （FO4 延迟是反相器驱动负载电容的延迟，负载电容是反相器输入引脚电容的四倍。FO4 延迟是测量栅极延迟的一个非常有用的指标，因为它基本上独立于工艺技术和操作条件。静态栅极 在各种此类条件下，FO4 延迟仅变化 20% [19]。）

因此，逻辑电平的简单相加不足以估计路径延迟。 以合理的精度预测设计的延迟需要知道逻辑实际映射到哪些门。 有必要在综合系统中包含静态时序分析引擎（第 6 章）来计算这些延迟。 计时与综合的结合是迈向集成时代的第一步。 这种趋势在20世纪80年代逐渐开始； 但到 20 世纪 90 年代初，集成静态时序分析工具对于准确预测延迟至关重要。 一旦将网表映射到特定技术并且可以近似门负载，时序分析器就可以对延迟进行相当准确的预测。

当时，近似门负载相对容易。 负载主要由所驱动的栅极的输入电容决定。 事实上，通过不准确的电线负载模型来估计电线的电容几乎不是什么问题。 因此，只要后续步骤中不修改网表，延迟预测就相当可靠。

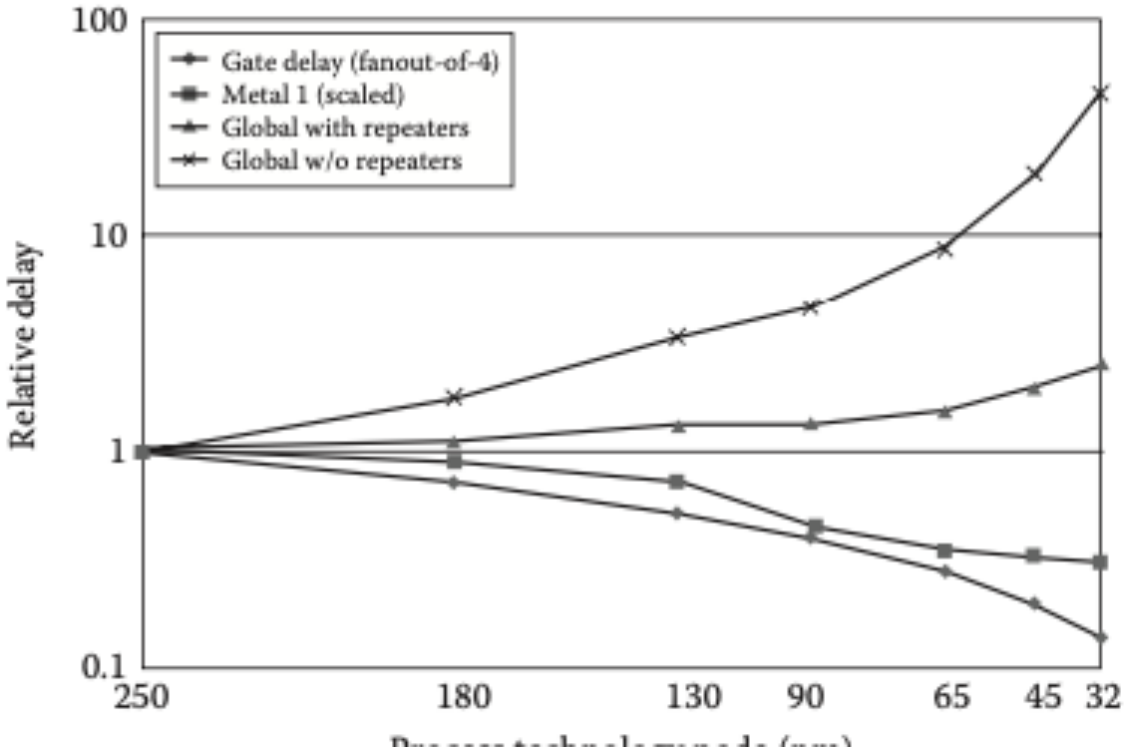
到 20 世纪 90 年代中期，这些基于门延迟的预测开始失去准确性。 栅极延迟越来越依赖于驱动的负载电容以及栅极输入信号（输入转换）的上升和下降速率。 与此同时，由导线电容引起的负载比例开始增加。 了解物理设计对于合理预测路径的延迟至关重要。 最初，主要只是需要放置单元格。 布局会影响延迟，但线路的影响要小得多，因为任何接近最小长度的线路都会有类似的负载。

在新技术中，越来越多的延迟开始转向互连。 栅极延迟和线延迟 (RC) 都开始变得重要起来。 图 1.2 显示了一系列技术中栅极延迟和互连延迟的比较。 通过全局路由的斯坦纳树近似，可以合理地预测网络的长度。 使用这些长度，可以计

算较长网络的延迟。 来自短网的负载不是很重要，对这些负载的猜测仍然是合适的。 很快，人们就清楚地意识到，很好地缓冲长网是非常重要的。 在图 1.2 中，我们看到在 130 nm 节点附近，在正确位置插入中继器的网络与无缓冲网络之间的差异开始对延迟产生重大影响。 在自动化布局和布线中，长线的缓冲成为物理设计的一个组成部分。 当今在长线上插入中继器（缓冲器或逆变器）的标准方法是 van Ginneken 的动态编程缓冲算法 [20] 及其衍生算法。 Van Ginneken 的缓冲算法有运行时复杂度为 O(n2)，其中 n 是可能的缓冲区位置的数量，但这已改进为近线性 [21]。

最近，电线的物理环境变得更加重要。 Elmore RC 线延迟模型不再准确，必须使用分布式 RC 网络模型，考虑电阻屏蔽 [22] 和交叉耦合。 随着线间距与线高度之间的比率减小，线之间的交叉耦合电容增加。 此外，暴露晶片时的光学邻近效应根据相邻导线之间的距离而变化，并且蚀刻导线时产生的影响也会影响它们的电阻。 因此，详细布线中的实际布线对于预测延迟非常重要。 此外，路由拥塞的全局路由估计可能与实际采用的路由有很大不同，并且由于严重的详细路由拥塞，全局路由可能会错误地预测发生路由违规的位置[23]。
传统上在逻辑综合中完成的网表优化已成为布局布线的重要组成部分。 必须重新设计用于处理静态（不变）网表的布局和布线系统。
直到最近，后路由优化仅限于微小的变化，这些变化会导致最小的扰动，以避免需要完全重新路由。 其中包括封装兼容的单元交换，特别是通过将单元交换到低阈值电压来修复关键路径延迟，以及通过将具有时序裕量的路径交换到高阈值电压来最小化泄漏功率； 手动更改路线以克服严重拥堵； 手动工程变更单 (ECO)； 并使用金属可编程备用单元在设计周期的最后阶段进行修复，从而避免了重做基础层的需要。 然而，全局布线估计与详细布线提取的电容和电阻之间的差距迫使在详细布线之前采取额外的保守态度，例如，避免使用小驱动强度单元，这些单元将受到额外布线电容的严重影响，从而在布线后留下重要的优化机会 。

1.4 时代融合



可预测性的下降持续存在，并让我们牢牢地踏入了整合时代。 以下是这个时代的一些特征：

- 后期设计步骤的影响正在增加。
- 预测是困难的。
- 更大的设计允许更少的手动干预。
- 新的成本函数变得越来越重要。
- 设计决策是相互作用的。

■ 激进的设计可以减少保护带。

连续设计步骤（例如中继器插入、栅极尺寸调整和布局步骤）之间的迭代不仅变得繁琐且缓慢，而且常常甚至无法收敛。 EDA 研究人员和开发人员已经探索了解决此收敛问题的几种可能的解决方案：

将后续设计步骤的控件插入到设计源中。

■ 最后解决问题。

■ 改进预测。

■ 在不同领域同时进行设计。

事实证明，插入控件非常困难。 如图 1.3 所示，源代码修改影响最终设计结果的路径可能非常间接，并且很难理解特定控件对特定设计和工具方法的影响。 在设计流程早期插入的控件可能会对最终结果产生与预期或预期截然不同的效果。 设计流程中的后期修复需要大量增加手动设计工作。 事实证明，改进预测非常困难。 基于增益的延迟模型用区域可预测性换取了显着的延迟可预测性，并提供了一些暂时的缓解，但总的来说，改进预测极其困难。 主要杠杆似乎是通过集成综合、布局和布线域并将它们与适当的设计分析器耦合来进行并发设计。

时序/综合集成之后，布局驱动（物理）综合是集成路径上的下一个主要步骤。 布局算法已添加到集成静态时序分析和逻辑综合环境中。 集成良好的物理综合系统对于解决日益大型芯片的设计收敛问题至关重要。

这种整合趋势仍在继续。 门到门延迟取决于导线长度（在综合期间未知）、导线层（在布线期间确定）、相邻导线的配置（例如，距离——近/远——在详细布线之前未知），以及相邻电线上信号的到达时间和斜率。 因此，在最新技术中，我们看到大部分布线需要完成才能很好地处理设计的时序。 局部拥塞问题可能会迫使大量路线绕道而行。 需要考虑到这一点，并且需要更早地布线大量网络以实现设计流程收敛。 网络之间的耦合会影响设计较大部分的噪声和延迟。 因此，了解特定网络的邻居对于进行所需详细程度的分析至关重要，并且需要完成本地路由的重要部分。 此外，功耗已成为一个非常重要的设计指标，而噪声问题开始显着影响延迟，甚至使设计无法正常工作。 除了集成静态时序分析之外，还需要包括功耗和噪声分析。

例如，现在常用的是用于修复布线后时序违规的空白布线后优化。 添加的单元格和调整大小的单元格放置在可用空白区域的合法位置，以避免干扰其他单元格的放置，并最大限度地减少对详细布线的任何更改在其他单元之间，但允许对连接到修改单元的网络进行重大重新路由。 较大的单元，特别是具有多个标准单元行高的单元，可能仍然不允许在布线后优化中移动以限制扰动。

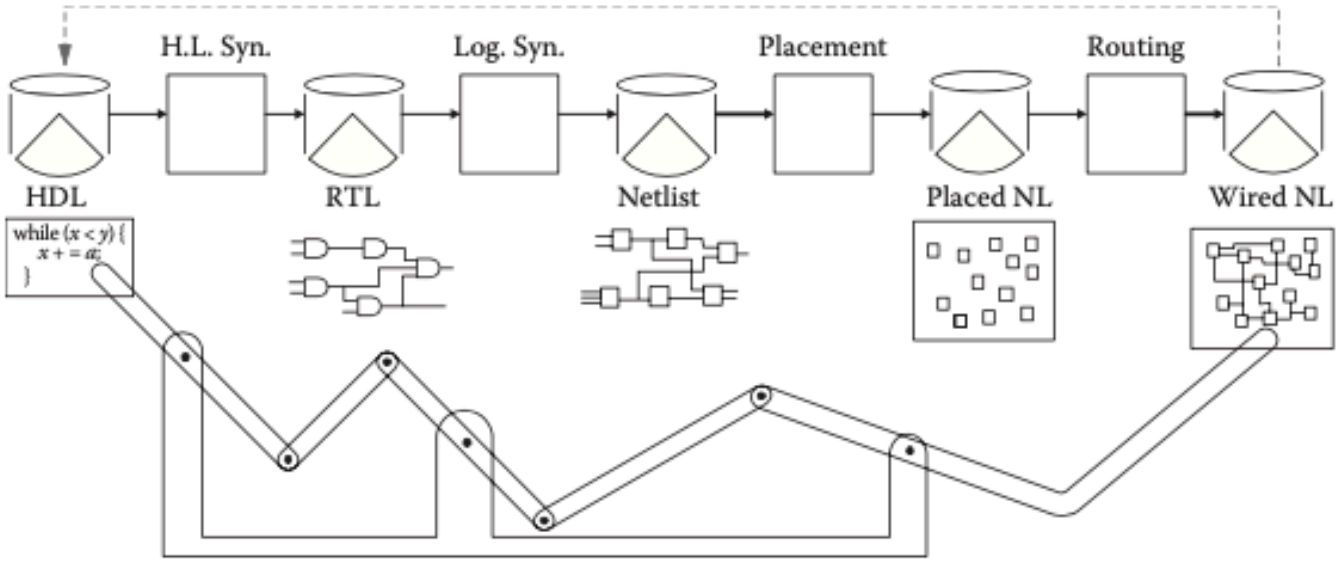


FIGURE 1.3 Controlled design flow.

考虑在后布线空白动态功率优化中通过扩大单元尺寸来降低短路功率的分析，短路功率由输入转换和负载电容决定。 这需要对新的小区位置进行增量评估，并对详细路由进行相应的更改。 通过信号完整性串扰分析来检查多个角落和模式的建立和保持时序影响。 单元的输入转换要考虑对扇入负载电容的影响以及影响其扇出的单元输出转换。 变化的转换和提取的线负载会影响动态功耗，并且该工具还必须确保总功率不会下降，其中包括活动模式下的动态功率和泄漏功率。 这种自动化的布线后优化可以减少高达 5% 的电池面积，有助于提高产量并降低制造成本； Mentor Graphics 的 Olympus-

SoCTM 布局布线工具 [24] 使布线后动态功耗降低了 12%，从而降低了总功耗并延长了电池寿命。 严重的建立和保持时序违规也可以通过自动方式修复。 考虑到这些改进是在设计流程的后期实现的，这些改进为设计团队提供了巨大的价值。

这些分析和优化算法需要以增量方式工作，因为当其他算法频繁进行设计更改时，运行时限制会阻止我们重新计算所有分析结果。 在集成时代，不仅单个算法很重要，而且它们集成以实现设计目标的方式也已成为差异化因素。 在快速增量时序、功率和面积计算器的指导下，这些算法之间的细粒度交互已变得至关重要。

在融合时代，最大的进步体现在算法之间的协作方式上。 原始综合、布局和布线算法的大多数原理及其高效实现仍然适用，但它们在 EDA 工具中的使用发生了显着变化。 在其他情况下，所需的增量导致了有趣的新算法——例如，通过布局和全局布线尝试不同的局部优化，以选择加速时序关键路径的最佳可行解决方案[25]。 在注重集成的同时，我们必须保持专注于各个工具领域的高级问题的能力，以解决技术带来的新问题，并继续提高设计算法的能力。

为了实现这一目标，我们看到了 EDA 工具开发的转变，该工具以四个相互关联的原则为指导：

1.工具必须集成。

2. 工具必须是模块化的。

3. 工具必须增量运行。

4. 工具必须仅稀疏地访问它们需要的数据，以减少内存使用和运行。

让我们在以下各节中更详细地了解其中的每一个。

1.4.1 工具集成

工具集成允许他们直接相互通信。 通过从公共用户界面启动传统点工具的执行（从文件读取输入并将输出保存到文件），可以提供表面形式的集成。 更紧密的集成形式允许工具在并发执行时进行通信，而不仅仅是通过文件进行通信。 这种紧密集成通常是通过在公共数据库上构建工具（参见第 15 章）或通过标准化消息传递协议来完成的。

工具集成可以在不同领域重用功能，因为消除了重复文件读写的开销。 这有助于减少开发资源需求并提高应用程序之间的一致性。 尽管工具集成可以实现增量工具操作，但它并不需要它。 例如，可以集成布局和布线程序，并且在开始

布线之前仍然可以完成布局运行。 精心设计的应用程序可以更轻松地与通用数据库模型上的其他应用程序集成，即使它最初是作为独立应用程序编写的。
实现工具集成需要设计表示中一组商定的语义元素，工具可以通过这些语义元素进行通信。 这些元件通常由单元、引脚和网络及其连接、单元放置位置和布线组成。 单元包括标准单元、宏和分层设计模块。 各个应用程序将使用特定于领域的信息来增强此通用数据模型。 例如，静态时序分析工具通常包括延迟和测试边缘数据结构。 为了使集成工具接受公共数据模型元素的查询，它必须能够有效地找到与这些元素关联的特定于域的数据的组件。 尽管这可以通过名称查找来完成，但当集成工具在公共内存空间中操作时，使用直接指针会更有效。 反过来，这要求公共数据模型为应用程序提供将私有指针附加到公共模型元素和回调的方法，以在元素更新时保持它们的一致性（参见第 1.4.3.3 节）。

1.4.2 模块化

模块化工具是以小的、独立的部分开发的。 这有几个好处。 它简化了增量开发，因为如果原始实现是模块化的，则新算法可以更容易地替换旧算法。 它有利于重复利用。 较小的模块化实用程序更容易重用，因为它们的副作用较少。 它简化了代码开发和维护，使问题更容易隔离。 模块更容易独立测试。 工具模块化应该对应用工程师和经验丰富的用户可见并可用，使他们能够通过扩展语言集成模块化实用程序。
过去，一些项目的失败很大程度上是由于缺乏模块化[26]。 为了将与通用数据模型相关的所有行为收集到一个地方，他们还集中控制了哪些应该是特定于应用程序的数据。 这使得数据模型变得过于庞大和复杂，并抑制了各个应用程序对数据结构的调整和重组。

1.4.3 增量工具

增量操作的工具可以更新设计信息或设计本身，而无需重新访问或重新处理整个设计。 这使得集成工具之间能够进行细粒度的交互。 例如，静态时序分析中的增量处理能力有助于逻辑综合更改设计并评估该更改对时序的影响，而无需执行完整的时序分析。
增量处理可以减少分析和优化工具之间循环所需的时间。 因此，它可以提高工具交互的频率，从而更好地理解每个优化决策的后果。

一组增量应用程序之间的操作顺序很重要。 如果诸如综合之类的工具调用另一个增量工具（如时序分析），则它需要立即访问所调用工具报告的其操作的效果。 因此，增量调用的行为必须就像每个增量更新在引发它的事件之后立即发生一样。
增量操作的一个示例是针对单元大小调整的快速假设本地时序分析，如果更改是有害的，则可以快速回滚到原始设计状态和时序状态。 时序分析的局部区域可能仅限于单元、单元的扇出（因为输出转换影响其延迟）、单元的扇出（负载受到影响）以及扇出的扇出（扇入转换随负载变化而变化） 。 在提交调整大小或执行全局成本分析之前，不允许时间更改传播到该区域之外。 完全准确的分析需要将到达时间传播到定时端点，然后传播回所需的时间，这可能会导致运行时间非常昂贵。 即使在完整的增量时序分析更新中，也只会遍历和更新那些受影响的路径，并且在确定更改没有影响的情况下（例如，如果它仍然是次关键侧路径），时序不会进一步传播。
增量工具需要具备四个基本特征：
1. 自治以避免显式调用和控制其他引擎，例如，通过注册回调以便向引擎通知影响它的事件
2. 延迟计算以推迟和最小化所需的额外计算
3. 更改通知，通知其他引擎更改，以便他们可以进行关联必要时进行更新
4. 可逆性，可快速撤消更改、保存和恢复
我们将在以下各节中进一步解释这些内容。

1.4.3.1 自治

自主性意味着应用程序启动促成另一个增量工具引擎中的更改的事件不需要显式通知增量工具这些事件，并且使用来自增量工具的结果的应用程序不需要显式启动或控制该增量工具中的增量处理。 工具。 避免显式更改通知非常重要，因为它可以简化对设计的更改，并且在将新的增量工具添加到设计工具环境时无需更新应用程序。 注册回调后，增量应用程序可以收到相关事件的通知。

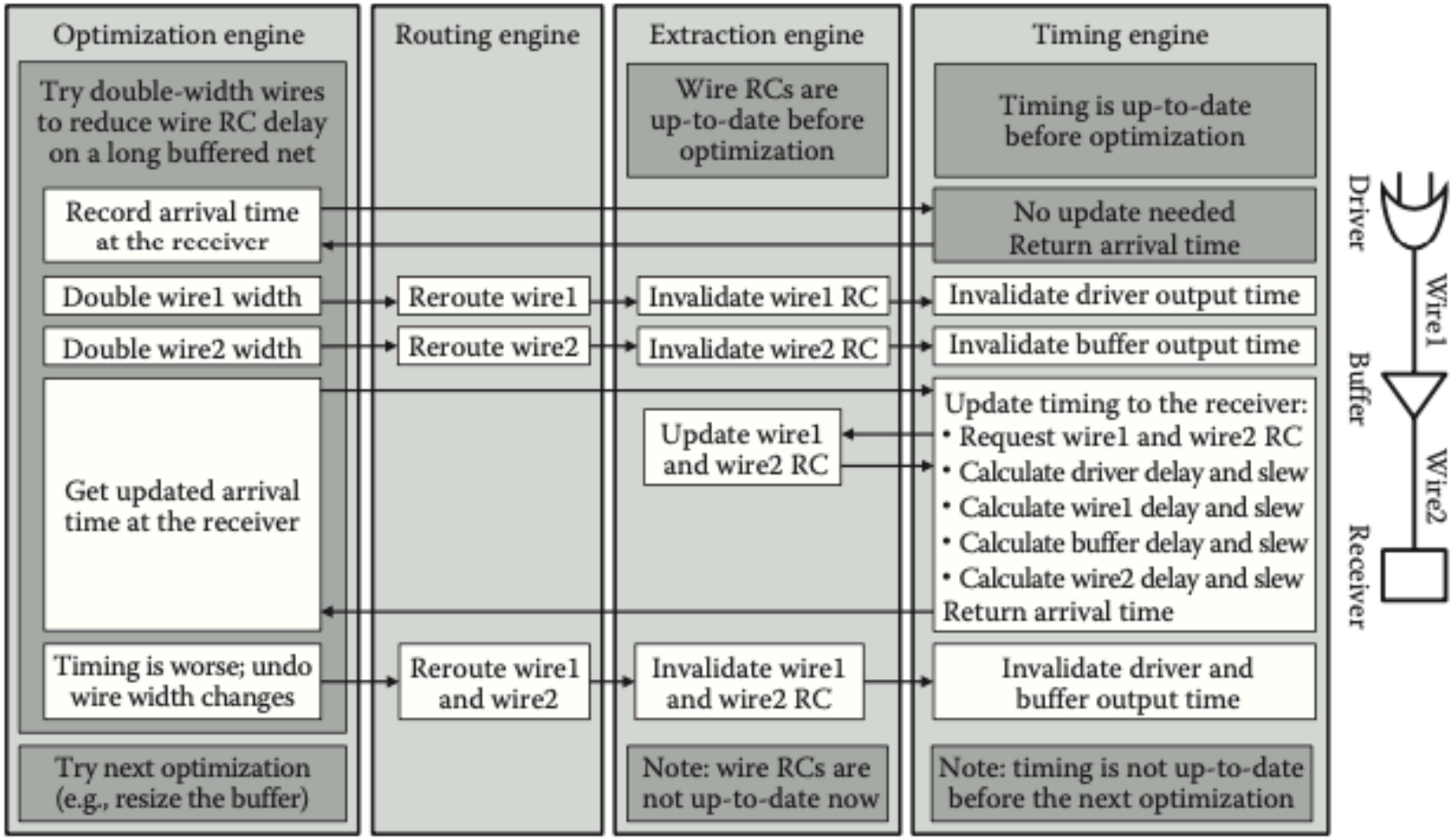


FIGURE 1.4 This example shows change notification callbacks from the routing engine to the extraction and timing analysis engines. There are lazy updates to the wire RC extraction and the tim-

避免对其他增量工具的显式控制很重要，这样调用者就不需要了解增量算法的细节，有利于未来算法的改进并减少出错的可能性。 它还简化了其他应用程序对增量工具的使用。

1.4.3.2 惰性评估（全面和部分）

惰性评估意味着增量工具应尽最大可能尝试推迟处理，直到需要结果为止。 当某些结果从未使用时，这可以节省大量的处理时间。 例如，考虑一个逻辑综合应用程序，它通过一系列单独的引脚断开和重新连接来完成某些逻辑转换。 如果增量时序分析器在每个单独操作之后更新时序结果，则最终会多次重新计算时间值，而实际上仅使用最后计算的值。 惰性求值简化了重新计算值时的控制流程。 如果我们有相互依赖的分析函数，当收到网表更改通知时，它们都尝试更新结果，那么必须对回调进行排序，以确保更新以正确的顺序发生。 例如，如果在提取更新之前进行计时更新，则计时结果将不正确，因为正在使用过时的提取结果。 通过惰性评估，每个应用程序仅在收到设计更改通知时执行无效操作，然后通过需求驱动的重新计算正确排序更新。 网表更改后，当请求时序时，时序引擎会在执行延迟计算之前请求提取结果。 提取引擎发现现有的提取结果已因网表更改的回调而失效，因此它会更新这些结果，然后时序引擎才会更新时序分析。 图 1.4 显示了一个例子。

如果一旦请求任何信息就执行所有挂起的更新，则惰性评估可以是完全的，或者如果仅更新提供所请求的结果所需的那些值，则惰性评估可以是部分的。 如果使用部分惰性求值，应用程序仍必须保留足够的信息，以便能够确定哪些信息尚未更新，因为稍后可能会请求该信息。 部分时序分析器[27]采用部分惰性评估，通过对设计进行层次化并基于此层次化限制到达传播和所需时间，从而在工具的运行时提供显着的好处。

1.4.3.3 更改通知（回调和非定向查询）

通过更改通知，增量工具可以通知其他应用程序与其相关的更改。 这不仅仅是提供一种从增量工具查询特定信息片段的方法，因为需要该信息的应用程序可能不知道已发生的所有更改。 在最简单的情况下，启动设计更改的工具可以假

设它知道分析结果（例如时序）的后续更改将在哪里发生。 在这种情况下，不需要更改通知。 但在其他情况下，工具可能不仅需要响应公共数据模型语义元素的直接变化，还需要响应特定应用程序域内的二次变化（例如，时序变化）。 更改通知很重要，因为应用程序可能不知道在哪里可以找到影响它们的所有增量更改。 例如，考虑逻辑综合使用的增量放置工具。 逻辑综合也许能够确定由于某些逻辑变化而将直接替换的块。 但是，如果这些块的替换产生连锁反应，导致其他块的替换（例如，为第一组替换块打开空间），则逻辑综合确定哪些块在该块中将会困难得多。 第二组。 如果没有更改通知，逻辑综合系统将需要在每次转换之前和之后检查设计中的所有模块，以确保它已考虑到该转换的所有后果。

变更通知可能会在突发事件发生后立即发出，也可能会推迟到收到请求为止。 可以通过回调例程提供即时更改通知，应用程序可以向该例程注册并在发生某些设计更改时调用该例程。

延迟变更通知需要增量工具来积累变更信息，直到请求的应用程序准备好为止。 该请求应用程序将发出无向查询，以询问自某个检查点（撤消设计更改所需的同一类型的检查点）以来发生的特定类型的所有更改。 这对于分析结果尤其重要，因为评估例程可能只对一系列更改的累积效果感兴趣，并且可能既不需要也不想支付立即更改通知的代价（在非惰性评估中）。

无向查询的典型用途是获取有关设计中发生的更改的信息，以便决定是否反转导致更改的操作。为此，不仅需要有关设计的结果状态的信息，还需要有关设计的结果状态的信息。 还关于原始状态，以便可以确定增量。 （事情变得更好还是更糟？）因此，进行无向查询的应用程序需要指定测量变化的起点。 这应该使用与提供可逆性相同的检查点功能。

1.4.3.4 可逆性（保存/恢复和重新计算）

对候选变更的试验评估非常重要，因为任何特定的设计变更都可能导致许多微妙的影响，而且大多数设计问题的复杂性也是如此。 应用程序进行试验设计更改，检查该更改的影响（部分由与其集成的增量工具确定），然后决定是否接受或拒绝该更改。 如果这样的更改被拒绝，我们需要确保能够准确地恢复之前的设计状态，这意味着所有增量工具结果必须是可逆的。

每个增量工具都应该处理其负责的数据更改的逆转。 在某些情况下，例如时序分析，更改的数据（例如，到达和所需时间）可以从其他设计数据确定性地导出，并且重新计算它们可能比存储和恢复它们更有效。 在其他情况下，例如增

量放置，更改涉及不可逆的启发式方法，并且必须将先前的状态数据保存在例如本地堆栈中。 处理更改逆转的增量工具应该是实际负责存储受影响数据的工具。 因此，由逻辑综合发起的对网表的更改应该由数据模型（或构建在其之上的独立层）而不是逻辑综合本身来逆转。

所有此类模型更改都应通过中央撤消设施进行协调。 这样的设施允许应用程序设置它可以返回的检查点。 可能有要撤消的信息的应用程序会在建立此类检查点或请求恢复到先前的检查点时向要调用的设施注册回调。

为撤消更改而对各种应用程序进行回调的顺序需要小心。 一种方法是检查增量应用程序之间的依赖性，并决定消除冲突的总体顺序。 一个更简单的解决方案是确保每个原子更改都可以反转，并让中央撤消工具以与最初进行更改的顺序相反的顺序调用反转所有更改。

应用程序可以通过两种方式之一撤消数据更改，如前所述。 保存/恢复应用程序（例如，放置）可以仅存储先前的状态并恢复它，而不需要调用其他应用程序。 重新计算应用程序（例如，时序分析）可以重新计算可以从模型状态唯一确定的先前状态数据。 重新计算应用程序通常不必注册特定的撤消回调，但为了允许它们进行操作，必须发出模型更改回调（以及所有其他更改通知回调）以反转模型更改，就像它们用于原始更改一样。

这样的中央撤消工具还可用于帮助捕获模型更改信息。 这可以是保存到文件（例如 ECO 文件），也可以传输到当前应用程序需要与之同步的同一台计算机或另一台计算机上同时执行的并行进程。

即使我们选择保留更改的结果，我们也可能需要撤消它然后重做。 典型的增量处理环境可能首先识别时序问题区域，然后尝试并评估几种替代转换。 在尝试每种替代方案之前，必须恢复原始模型状态，并且在评估所有替代方案之后，重新应用给出最佳结果的替代方案。

为了确保我们在重做转换时得到与最初执行转换时得到的结果相同的结果，我们还需要能够撤消撤消操作。 即使对于确定性转换，确切的结果也可能取决于访问某些对象的顺序，并且这种顺序可能不是语义不变量，因此在撤消更改时可能会改变。 此外，某些转换可能需要进行大量分析才能确定要进行的确切更改，并且当我们重做转换时，我们希望跳过此分析。 避免不确定性行为也是非常可取的； 否则，在调试期间重现错误行为会充满复杂性。出于这些原因，中央撤消/ECO 设施应该能够存储和管理检查点树，而不仅仅是单个检查点。 对电路的试验转换也可能是嵌套的，因此我们需要能够堆叠多个检查点。

重要的是要记住，没有什么灵丹妙药可以将非增量工具变成增量工具。 除了拥有通用的基础设施（包括通用数据模型和回调机制）之外，还需要开发适当的增量算法。
遵循其中一些准则还可以鼓励增量设计自动化工具的开发。 重写工具是昂贵的主张。 很少有新想法可以改变工具的所有方面。 增量开发可以提高工具界面的稳定性，这对于集成应用程序尤其重要。 它还可以更便宜地实施和评估新想法，并且可以更快地提供所需的功能。

1.4.4 稀疏访问

稀疏访问是指仅加载执行给定任务所需的设计部分和相关文件，并推迟加载其他数据直到需要时。 在某些情况下，这可以将内存使用量和运行时间减少一个数量级。
例如，验证与电源门控睡眠标头单元的正确连接的设计检查只需遍历单元引脚和网络连接，并且如果通过命名约定识别单元类型，则可以避免加载库。 所遍历的网表部分可能进一步限于仅连接到睡眠标头单元和驱动睡眠控制信号的单元的网络，以确保逻辑连接到始终开启的电源。
对网表或寄生数据的稀疏访问通常需要使用索引将内存分页到磁盘，以指定在何处查找给定单元或网络的数据。 同样，可以对库内的单元进行索引以提供单独的访问，而不是加载整个库，这可以将加载复合电流源 (CCS) 库的内存开销减少一个数量级或更多，因为仅使用几百个单元 数千中。 这可能需要对库进行预先表征，例如，提供可用于修复保持违规的延迟缓冲区的缓存列表，以及在面积或功率与延迟权衡方面达到帕累托最优的那些子集。

1.4.4.1 单片 EDA 工具和内存限制

设计公司的内部工具通常是快速点工具，它们仅加载设计的必要部分以及相关的技术文件或库，以最大限度地减少启动运行时间开销，因为正在执行的任务可能会在一分钟到一分钟内运行。 小时。 这可以通过每晚在服务器场上的所有设

计块上并行运行数十个设计检查的示例来说明。 在启动需要大量时间的进一步运行之前，此类点工具对于分析和检查设计是否存在任何重大问题非常有用。
相比之下，商业 EDA 工具通常被设计为以整体方式使用。 将在该工具内执行一个或多个主要流程步骤，例如综合、布局和优化； 时钟树综合（CTS）； CTS后优化； 详细的路线； 和后路由优化。 整个设计、技术文件和库都被加载，占用大量内存。 与正在执行的较大流程步骤相比，几分钟的初始加载运行时间较短。
商业 EDA 工具现在在跨多个流程和操作角落分析和优化大型设计时面临着严格的内存限制。 30 个角点和模式或更多的组合并不少见，例如慢速、典型和快速工艺角点； 低温、室温和高温； –5%电压、典型电压和+5%电压； 电源电压模式，例如 0.7、0.9 和 1.1 V； 以及睡眠、待机、唤醒和扫描操作模式。 每个 CCS 库可以是 10 GB，因此仅加载所有这些角落的库就可以占用 100 GB 或更多内存。 当在顶层以平面方式进行分析时，当今的设计通常介于数十万个门到数亿个门的块之间，并且还可能占用 100 GB 或更多内存，特别是使用标准寄生交换格式 (SPEF) – 带注释的寄生电容。 当在高端 16 CPU 核心服务器上运行多线程以减少运行时间（否则需要数天）时，还必须复制一些数据以运行多线程，以避免内存读/写冲突并限制同步数据所需的暂停。 如今的服务器群只有几台具有 256GB 内存的机器，而具有更高内存的服务器非常昂贵。
EDA 开发人员使用各种标准编程技术来减少内存使用。 例如，矢量数据结构比链表占用更少的内存（存储相同的数据），而多个布尔变量可以打包为单个 CPU 字中的位，并在需要时使用位掩码提取。 重复的数据可以被压缩，例如，在布局与原理图验证期间[28]，其中布局数据的分层表示表示包含许多重复几何形状的各个库单元。 跨回归套件仔细监控内存使用情况、运行时间和结果质量（例如电路时序、面积和功耗），以确保 EDA 工具中的代码更改不会降低结果或工具性能。

1.4.4.2 EDA 并行和分布式计算

EDA 供应商正在将工具迁移到以分布式方式工作，其中一台机器上的控制器将任务分配给其他机器上的工作人员以并行执行。 多线程和 fork-join [28] 也是 EDA 软件中常用的并行技术，它们可以与分布式计算相结合。 这些方法可以将分析、优化和验证的运行时间减少一个数量级。 通过适当的排队，还可以更有效地使用共享资源。

RTL 到 GDSII 流程中使用的许多算法是图算法、分支定界搜索或线性代数矩阵计算 [29]。 虽然大多数这些算法可以并行化，但阿姆达尔定律 [30] 由于计算的串行部分以及通信开销、缓存一致性以及需要进程之间同步的瓶颈而限制了加速。 并行性带来的性能提升也经常被夸大[31]。

并行化软件的大部分开发时间实际上集中在减少串行部分的运行时间，例如，避免共享资源上的互斥锁，以最大限度地减少一个进程必须等待另一个进程完成的停顿。 串行算法的快速且内存高效的实现，例如 Dijkstra 的最短路径算法 [32]，在 EDA 中仍然非常重要，因为通常没有更好的并行算法 [31]。

作为 EDA 中如何使用分布式计算的一个示例，几个重要时序角的优化可能需要具有大量内存的服务器，但内存较低的服务器可以验证其他角不会发生时序违规。 对导致大多数建立和保持时序违规的主要角点进行分析减少了对其他角点进行分析的需要。 然而，仍然存在一些情况，在一个优势角球的修复可能会导致另一个角球的违规，所以它不能完全避免。 控制器和工作人员无法承受加载所有角落的库的内存开销，也无法承担额外的分析运行时间。 分布式优化允许保持在服务器内存限制内。 一名工作人员仅在主要角点进行分析来执行优化，其他工作人员则验证一组更改不会违反其他角点的约束。

将大型设计划分为可以单独处理的较小部分是实现并行性的常见方法[29]。 这可以通过最小割分区[12]来完成，以最小化存在次优性的跨境约束。 然而，由于再收敛时序路径、基于锁存器的时序、导线之间的交叉耦合以及限制简单最小切割划分方法的其他因素，一些区域可能过大。 需要调整区域的大小，以平衡工人之间的负载。 如何最好地对设计进行子分区是 EDA [29] 中的一个关键问题，并且因应用而异。 例如，时序优化可以仅考虑关键时序路径，而区域优化可以尝试调整每个不固定的单元的大小。

设计的某些部分可以以分布式方式并行快速优化。 为了确保工人对设计的更改之间的一致性而施加的边界约束确实降低了最优性。 个别工作人员使用稀疏访问来读取他们正在优化的设计部分，尽管逻辑连接可能不够，因为交叉耦合分析考虑了物理上附近的电线。

当并行执行多个更改时，并行计算可能是不确定性的根源。 为了避免这种不确定性，对分布式工作线程的增量分析可能需要使用具有周期性同步点的一致电路快照，在这些点上以固定的确定性顺序完成或撤消更改和分析更新。 某些操作可能还需要串行进行，以确保行为一致。

1.5 未来的扩展挑战

在前面的部分中，我们主要关注持续扩展如何改变我们在整个设计流程中预测延迟的方式。 这是过去二十年来设计流程变化的主要驱动力之一。 然而，新的挑战要求我们重新思考自动化设计流程的方式。 在以下各节中，我们将更详细地描述其中的一些内容，并认为它们会导致设计收敛，除了需要对可布线性、功耗、噪声和可变性（使用所需的增量提取工具）进行集成的增量分析之外， 完善的增量时序分析。

1.5.1 动态功率和泄漏功率

传统上，工具侧重于最小化关键路径延迟和电路面积。 随着技术尺寸的缩小，晶体管的密度增加，减轻了面积的限制，但增加了功率密度（每单位面积消耗的功率）。 散热限制了最大芯片功率，进而限制了开关频率，从而限制了芯片的运行速度。 通过 90 nm 技术，甚至一些高端微处理器设计人员也发现功耗成为性能的主要限制 [33]。 第 3 章提供了本小节讨论之外的功耗分析和优化的更多详细信息。

根据摩尔定律提高电路速度的代价是动态功率的更快增加。 由于使用电源电压 Vdd 切换电容 C 所产生的动态功率为 fCV2dd/2。 增加电路速度会成比例地增加开关频率f，并且单位面积的电容也会增加。

晶体管电容与晶体管栅极氧化层厚度 tox 成反比（Cgate = εoxWL/tox，其中 εox 是栅极介电常数，W 和 L 是晶体管宽度和长度）。 栅极氧化物厚度随晶体管长度线性减小，以限制短沟道效应并保持栅极驱动强度以提高电路速度 [34]。 随着器件尺寸线性缩小，晶体管密度呈二次方增加，单位面积的电容线性增加。 此外，导线电容相对于栅极电容有所增加，因为导线间隔更近且纵横比更高，以限制导线电阻，从而限制相应的导线 RC 延迟。

如果电源电压保持恒定，则由于开关频率的增加和电路电容的增加，每单位面积的动态功率的增加速度慢于二次方。 为了降低动态功耗，电源电压已按比例降低。 然而，这会降低驱动电流，从而降低电路速度。 饱和驱动电流 ID,sat =

kW (Vdd - 2Vth)a Ltox，其中 Vth 是晶体管阈值电压，对于最新技术而言，指数 a 介于 1.2 和 1.3 之间 [35]。 为了避免速度降低，阈值电压已随电源电压降低。
静态功耗随着晶体管阈值电压和栅极氧化层厚度的减小而增加。 当晶体管栅源电压低于 Vth 并且晶体管名义上关闭时，亚阈值泄漏电流会以指数方式取决于 Vth (Isubthreshold = ke-qVth /nkT ，其中 T 是温度，n、q 和 k 是常数）。 当栅极氧化物变得非常薄时，电子穿过薄栅极氧化物的量子隧道概率非零。 虽然栅极隧道电流比亚阈值漏电流小几个数量级，但由于栅极氧化物厚度的减少，它的增加速度要快得多[36]。
自动逻辑综合和布局布线工具主要关注电路内逻辑评估时的动态功耗，因为当电路处于活动状态时，静态功耗仅占总功耗的极小部分。 例如，自动时钟门控减少了不必要的逻辑切换。 管理待机静态功耗留给设计人员通过使用诸如关闭未使用的模块的电源或选择具有较低泄漏的标准单元库等方法来处理。

1.5.1.1 漏电功率
通过使用高阈值电压睡眠晶体管连接到电源轨可以降低待机功耗[37]。 在待机状态下，这些晶体管被关闭，以阻止从电源到接地轨的亚阈值泄漏路径。 这种技术称为功率门控。 当电路处于活动状态时，这些睡眠晶体管处于开启状态，并且它们的尺寸必须足够宽，以仅导致电压摆幅小幅下降，但又不能宽到消耗过多功率。 为了支持这种技术，布局布线软件必须支持与睡眠晶体管提供的虚拟电源轨的连接以及同时进入待机状态的单元集群，以便它们可以共享公共睡眠晶体管以减少开销。
在最新的 45-28 nm 工艺技术中，当电路处于活动状态时，泄漏可贡献高达总功率的 40% 左右。 有一个交易——动态功率和泄漏功率之间的关断。 降低阈值电压和栅极氧化层厚度允许使用更窄的晶体管实现相同的驱动电流，相应地具有更低的电容和降低的动态功率，但这会增加泄漏功率。 增加晶体管沟道长度提供了一种减少泄漏的替代方法，但代价是增加了栅极延迟，并且栅极电容稍高，因此动态功率更高。 交替沟道长度更便宜，因为交替阈值电压需要单独的注入掩模，这增加了制造费用。
设计人员现在通常使用具有多个阈值电压和多个沟道长度的标准单元库，例如，六种阈值电压和沟道长度替代方案的组合：低/标称/高Vth，具有常规和+2 nm沟道长度。 低阈值电压和较短沟道长度的晶体管减少了关键路径的延迟。 高阈

值电压和更长的沟道长度晶体管减少了具有松弛的栅极的泄漏功率。 EDA 工具必须支持在具有较小和较大泄漏的单元之间进行选择，以适应电路环境中栅极的延迟要求。
在过去的几年里，使用拉格朗日松弛等快速全局优化方法，在泄漏功率最小化方面取得了显着的进步。 这使得在一小时内优化百万门网表成为可能[38]，并且通过多线程可以进一步显着加速。
当试图最小化总功率时，工具必须将泄漏功率与动态功率同等对待，这一点至关重要。 优先优化泄漏功率或动态功率不利于活动模式下的总功耗。 在功耗优化期间还必须考虑其他操作角点和模式，因为并非所有门都可以断电，因此对待机功耗也有设计限制。

1.5.1.2 动态功率
多栅极器件（例如三栅极 FinFET）的漏电功率已显着降低 [39]。 通过 22–14 nm FinFET 工艺技术，我们发现当电路处于活动状态时，泄漏功率占总功率的 10%–30%，具体取决于开关活动和阈值电压选择。 因此，设计人员增加了动态功耗最小化技术的使用。
工业界和学术界在寄存器聚集和多位触发器交换以减少时钟负载和时钟功率方面取得了重大进展。 其他研究主要集中在优化时钟树以权衡延迟、时钟偏差和时钟功率。 通过多源 CTS 可以实现 30–50 ps 范围内的时钟偏差，通过自动时钟网格综合 [40] 可以实现 10–30 ps 范围内的时钟偏差，这为设计提供了进一步的机会，通过在时钟偏差之间进行权衡来实现更高的性能 和时钟电源。
一些行业设计已经使用细粒度电压岛来降低功耗[41]，但由于迄今为止复杂的设计方法和有限的市场，这尚未得到 EDA 供应商工具的支持。

还有机会结合栅极尺寸/Vth 分配和缓冲来自动优化线宽和线距，以优化延迟与功耗。 为了降低高活动导线的动态功率，可以优先布线以缩短它们，并且可以使用更宽的间距来减少它们与邻居的耦合电容。 缩小栅极尺寸会降低开关功率，但在某些情况下，可能最好增大栅极尺寸或插入缓冲器以减少压摆和短路功率。 供应商 EDA 工具仍然很少对线间距或线宽度进行优化，忽略了 (1) 增加线电容和 (2) 减少线电阻和减少 RC 延迟之间的权衡。 线宽和间距的非默认规则 (NDR) 通常由设计人员手动提供，并提供一些有限的工具支持来在可用的 NDR 之间进行选择。

对于目前还缺乏自动化支持的设计人员来说，降低峰值功率和毛刺的技术（无论是峰值功率还是平均功率）也很感兴趣。 可以通过插入去尖峰脉冲锁存器或通过平衡路径延迟来减少毛刺，但动态时序分析的运行时间成本很高，因此需要简化的快速方法。 还可以重新映射逻辑以更好地平衡路径延迟或减少逻辑内的切换活动，但重新映射的运行时间成本相当高。

1.5.2 复杂设计规则下的布局和可布线性

制造更小特征尺寸的多重图案显着增加了设计规则的复杂性，从而使布局和布线变得复杂。 由于同一层上的特征之间的相互作用必须位于不同掩模上才能实现更高分辨率，并且由于多图案掩模的潜在未对准，设计规则更加复杂。 有关设计规则的更多详细信息，请参阅第 20 章。

光刻光源的波长对 IC 制造中可以解决的特征的最小线宽施加了经典的瑞利限制，如第 21 章所述，该章对多重图案化进行了更深入的讨论。 超过 65 nm 工艺技术的分辨率受到氟化氩激光器提供的 193 nm 波长的限制。 为了进一步扩展，45 nm 器件需要浸没式光刻或双重图案化，32 至 14 nm 的技术需要浸没式光刻和双重图案化[42]。 三重图案化将首先用于 10 nm 工艺技术 [43]。 多重图案化需要额外的掩模来暴露需要较小分辨率的层，从而产生额外的制造成本。 更高、更宽的金属层可以更便宜地添加，无需多重图案化或浸没式光刻。

用于制造较小特征的其他解决方案尚未适用于大规模生产。 13.5 nm 波长的极紫外 (EUV) 光刻有望以更简单的设计规则以更低的成本打印更小的晶体管 [44]。 然而，由于各种问题，特别是难以实现 100 W 电源以提供良好的曝光吞吐量，EUV 的引入已被进一步推迟到至少 7 nm 工艺技术 [45]。 最近，ASML 报道了台积电 24 小时内生产 1022 片晶圆以及 110 W EUV 功率能力的记录[46]。 电子束可用于写入小图案，但速度太慢而无法实用，除非它们已广泛用于测试制造的 IC [47]。 7 nm 的 EUV 可能需要双重图案化，而 5 nm 技术的 EUV 则需要这种分辨率增强技术 [43]。 因此，代工厂致力于使用多重图案化来制造更小的工艺技术，并且他们正在准备替代解决方案，例如自对准四重图案化，以防 EUV 未能及时为 7 纳米工艺技术做好准备。 布局和布线的多图案设计规则约束现在是主要问题。

自 32nm 以来，需要水平边缘型布局限制 [48]，以提高产量和可制造性设计。 例如，如果相邻电池边缘上多晶硅的布局形状对产量不利，则可能需要留出间距以允许插入虚拟多晶硅，以最大限度地减少相邻电池之间的光学干扰[49]。 这会导致不同的水平边缘间距限制，具体取决于单元的布局。

在 10 nm 工艺技术中，我们现在还看到垂直边缘邻接限制。 引入此类限制是为了遵守更保守的设计规则，例如，为附近的过孔提供足够的空间以连接到单元引脚或电源/接地金属带。 另一个目标是防止单元内电线的颜色选择之间发生冲突，因为相邻电线可能不具有相同的颜色分配。 （在多重图案化的背景下，着色是指将布线的一部分分配给哪个图案。）垂直限制更加复杂，指定距离范围，以便单元的某些部分不能垂直邻接。 早期的工具支持已经针对这些垂直限制提供，但需要对布局器和布线器进行进一步更新，以便在优化过程中更好地考虑这些问题。 库交换格式（LEF）[48]也需要更新以添加这些垂直放置约束的规范。

在全局布局中，通过估计所需的额外间距暂时增加（膨胀）单元的大小可以帮助解决边缘类型布局限制并减少布线拥塞[50,51]。 布线阻塞、布线 NDR 和预布线（例如电源、接地和时钟网格）会加剧布线拥塞。 详细布局还必须在布局合法化期间考虑这些因素，因为如果单元引脚位于预布线旁边或下方，或者由于 NDR 需要额外的间距，则可能无法连接单元引脚 [51]。

随着多重图案设计规则复杂性的增加，图案必须更加规则，并且可能仅限于每层的单向段，以提高可制造性[52]，不允许错误的布线并需要更多的过孔，从而增加布线拥塞。 在允许错误路由的情况下，存在差异对于非首选方向，不同的设计规则对电线之间的间距限制要宽得多。 路由器已经更新以支持这些更复杂的设计规则，并且现在还必须考虑过孔的电阻和电容以进行准确的寄生提取。

还有在单元之间插入虚拟金属填充的规则，以减少电介质厚度的变化，但金属填充会增加导线耦合电容，从而增加导线延迟[53]。 金属填充增加了复杂性，即缩小电池尺寸可能会导致间距违规，因为在没有足够空间用于金属填充的电池之间可能会引入间隙。 类似地，由于单元内的布局不同，尺寸缩小的单元可能具有不同的边缘类型，如果不同的边缘类型需要更宽的间距，则可能会导致与相邻单元的布局冲突。

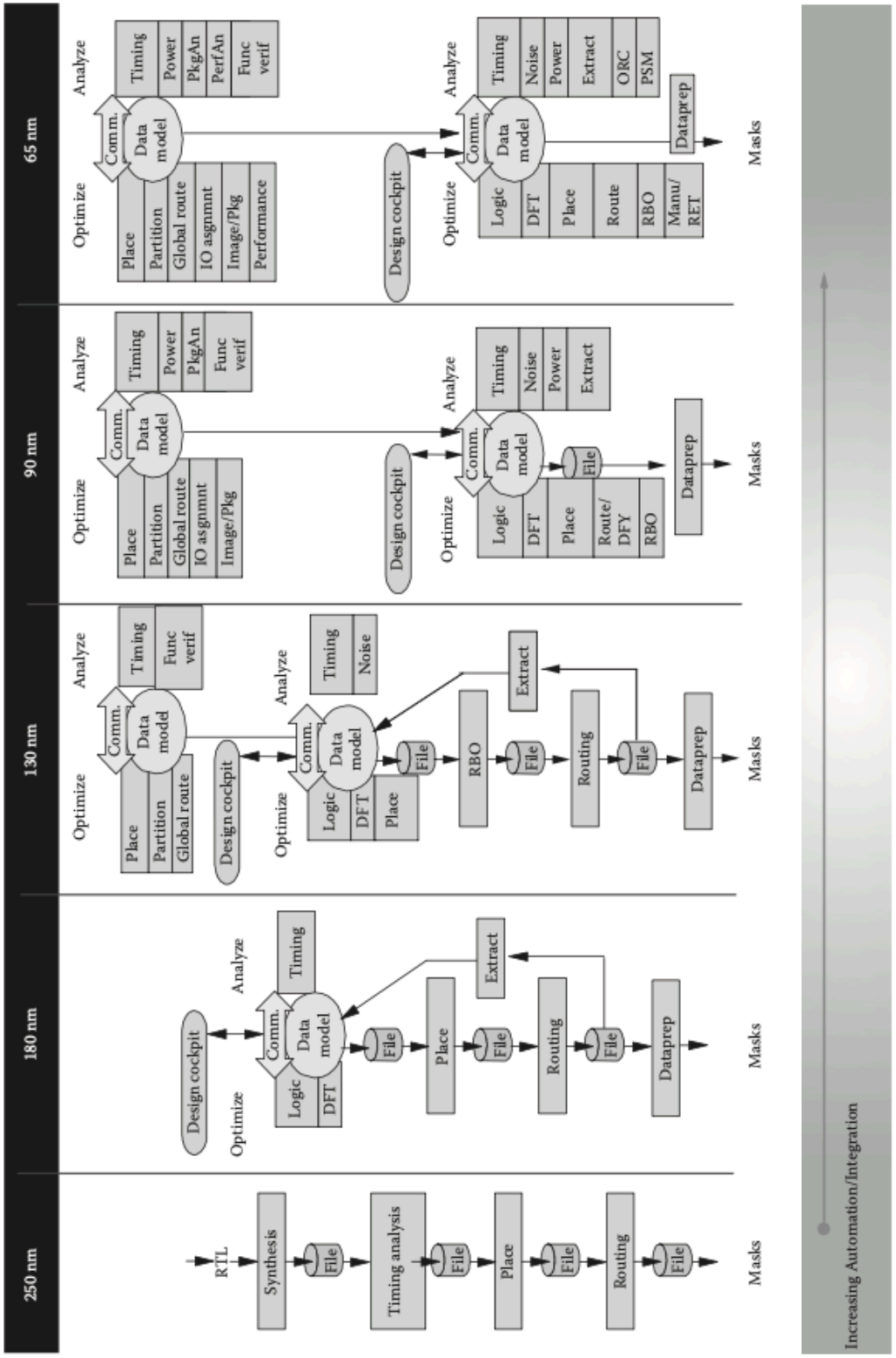
当执行多轮图案化（通常是两轮或三轮）时，每轮的形状通过颜色来标识。 由于多重图案化时不同颜色的掩模之间的未对准误差，同一金属层中的导线耦合电容随着颜色的选择而不同[54]。 根据代工厂和制造工艺的不同，一些 EDA 流

程预分配布线跟踪给定的颜色，而其他设计流程是无色的，由代工厂对提供的 GDSII 布局进行着色。 当未确定导线颜色时，如在无色设计流程中，可以在时序分析中考虑通过添加两个额外的最小和最大角来进行导线电阻和电容提取[55]。 单元内导线的颜色分配会影响单元延迟特性，因此如果标准单元库提供不同的着色替代方案，EDA 工具可能会选择不同的单元实现，这些实现仅在单元内导线的颜色分配上有所不同。 全局和详细的路由器还需要了解电线颜色，要么保守地估计最坏情况颜色选择的电容和电阻，要么最好根据时序关键性优化电线颜色选择。
一些间距限制不需要很严格，但可以提高良率，因此在机会允许的情况下可以满足这些限制，而不会降低关键路径延迟并因此降低时序良率。 当今的 EDA 工具严格执行间距限制。 设计小组已经构建了快速增量工具，以使用具有稀疏访问的内部数据库实现或供应商工具中脚本化的较慢 TCL 实现来提供此功能 [56]。 原生 EDA 工具在布局期间仍然有更多机会支持良率分析和优化。

1.5.3 过程和操作条件的可变性

历史上，人们认为在一个慢角和一个快角进行静态时序分析足以确保满足时序约束。 最坏情况的电路延迟是根据标准单元库的慢角估计的：这是慢速工艺角，具有较低电源电压（例如，–10% Vdd）和高温（例如，100°C）的慢速工作角 ）。 快速时序路径和最坏情况动态功耗造成的保持时间违规是从快速工艺角进行估计的，其中快速操作角具有较高的电源电压（例如，+10% Vdd）和低温（例如，0°C）。 由于工艺可变性导致较高（较低）阈值电压和较长（较短）沟道长度，p 型和 n 型 MOSFET 都将出现慢（快）工艺角点，这也会减少（增加）亚阈值漏电流。 可以在快速工艺和高温角估计最坏情况的泄漏功率。
在实践中，缓慢和快速的工艺变化以及芯片上点温度的差异的混合可能会导致单个工艺角无法预测的不可预见的故障。 同样在 90 nm 和更小的工艺技术中，延迟的反向温度依赖性增加了低温下的额外较慢的操作角，以及静态时序分析的进一步复杂性 [57]。 温度反转是由于漏极电流与载流子迁移率和阈值电压的竞争温度依赖性造成的。 电源电压较高时，迁移率的影响更为显着；电源电压较低时，阈值电压的影响更为显着。 迁移率随着温度的升高而降低，从而降低了漏极电流，从而减慢了器件的速度。 阈值电压随着温度的升高而降低，从而使电路速度更快。 由于这些竞争效应，路径延迟可能不再随温度单调增加，而是可能随温度先减小然后增加[57]。

如今，结
点和模式
时序签核
常见。
多数制造
正常电路
能速度更
均功率更
于这种过
分析，设
常高。
尽管在当
中某些电
到更严格
但其他元
却有所增
如，晶体
压或沟道
幅减小可
漏的大幅
逻辑单元
晶体管的
更加复
学干涉会



Integrated design flow.

合多种角
的优化和
已变得很
此外，大
的芯片在
条件下可
快并且平
低。 对
于保守的
计成本非

今的工艺
路元件受
的控制，
件的变化
加。 例
管阈值电
长度的小
能导致泄
增加。
的电线和
布局变得
杂。 光
改变最终

的布局。 相移光刻试图纠正这个问题。 不同的蚀刻速率对较窄的线有更大的影响。
某些单元布局更有可能降低产量，例如，由于栅极延迟增加，甚至完全失效（例如断线）。 理想情况下，不会使用产量较低的单元布局，但这些单元可能具有较高的速度。 为了满足延迟要求，产量的小幅下降是可以接受的。 为了支持良率权衡，代工厂必须与客户共享一些良率信息以及相应的价格点。
产量和变异性数据可以注释到标准单元库中，使软件能够对时序、功率和产量进行统计分析。 这需要将过程变异性详细分解为系统（例如，空间相关）和随机分量。 有了这些信息，工具就可以估计满足延迟和功率限制的芯片产量。 这并不简单，因为时序路径在统计上是相关的，并且变异性也在空间上相关。 然而，这些相关性可以用保守的方式来解释，这种方式仍然不如最坏情况角点分析保守。
第 6 章讨论了与静态时序分析相关的工艺和操作条件的变化。第 22 章详细讨论了工艺变化和可制造性设计。

1.5.4 电路可靠性和老化

电路运行中的瞬态毛刺可能是由交叉耦合噪声以及阿尔法粒子和中子轰击引起的。 此外，电路老化问题也越来越引起人们的关注，包括电迁移、热载流子注入和负偏置阈值不稳定性。 第 13 章将更详细地讨论这些内容。
随着电路频率的提高和导线之间的耦合电容的增大，交叉耦合噪声变得更加显着。 导线交叉耦合电容随着导线间隔更近和纵横比更高而增加，这已被用于随着尺寸缩小而减少导线 RC 延迟。 可以布置电线以减少交叉耦合噪声（例如，通过扭转或用地线屏蔽）。 有一些工具可用于分析交叉耦合噪声，还有一些工具支持半屏蔽或全屏蔽电线，方法是将电线的一侧或两侧靠近电源或接地网络，但会造成额外的布线拥塞和绕行。
由于栅极电容随着器件尺寸的减小和电源电压的减小而减小，因此存储的电荷量会减少，并且更容易被α粒子或中子撞击所破坏。随着器件尺寸的缩小，由α粒子引起的软错误率会大幅增加[58]。 通过使用绝缘体上硅技术和其他制造方法或使用对瞬态脉冲更耐受的锁存器可以降低软错误率[58]。
为了对故障进行容错，电路可以具有错误检测和纠正或额外的冗余。 对逻辑综合和映射到此类电路的工具支持将简化设计人员的任务。

电迁移会导致开路或短路[59,60]，从而妨碍正确的电路操作，随着电流密度随着电线变窄而增加，这变得更加成问题。 设计人员现在需要更好的 EDA 工具支持来分析电迁移 [61] 和自动化解决方案，例如加宽具有高开关活动的电线以及时钟单元周围的额外间距。 在 10 nm 技术中，时钟单元可能还需要在其旁边插入去耦电容器以限制电压降。

1.6 结论

在本章中，我们研究了 RTL 到 GDSII 的设计流程如何随着时间的推移而发生变化，并将继续发展。 如图 1.5 所示，我们预计将继续将更多分析功能集成到集成环境中，以应对鲁棒性设计和功耗设计。 我们概述了典型的 EDA 流程及其步骤，并概述了进一步开发 EDA 工具的动机。

IC 设计公司迫切需要更快的周转时间来迭代 EDA 流程。 设计中晶体管数量继续快速增加，集成了更多组件，更小的 10 和 7 nm 工艺技术正在积极研究和开发，3D IC 即将出现。 这些新技术的更复杂的设计规则推动了主要 EDA 工具的就位和布线开发。 为了实现更高的性能和更低的功耗，必须改善从 RTL 估计和综合到后布线优化的相关流程步骤之间

的相关性。 这就需要进一步的工具集成，分析更多的流程角落和更多的操作模式，并选择什么是适当的抽象级别，以便在设计流程的早期实现更高的准确性。 所有这些都增加了计算需求，这促使增加 EDA 工具的并行性，支持更大的数据集，但仍然在相对严格的内存限制内。 随着 EDA 市场持续增长，尽管随着行业内的进一步整合而日趋成熟，EDA 行业和学术界仍在继续进行大量研究和开发，以应对这些困难的设计挑战。
以下章节更详细地描述了各个 RTL 到 GDSII 流程步骤，检查每个流程步骤中使用的算法和数据结构。 设计数据库、单元库和工艺技术的附加基础设施也会在后面的章节中讨论。 我们希望这些能够鼓励读者进一步创新，更深入地了解 EDA。

逻辑综合

2.1 简介

逻辑综合的根源可以追溯到乔治·布尔（George Boole，1815-1865）对逻辑的处理，即现在所谓的布尔代数。 1938 年香农 [1] 的发现表明，二值布尔代数可以描述开关电路的运行。 早期，逻辑设计涉及将真值表表示操纵为卡诺图 [2,3]。 基于卡诺图的逻辑最小化是由一组关于如何组合图中的条目的规则指导的。 人类设计师只能使用包含四到六个变量的卡诺图。 逻辑最小化自动化的第一步是引入了可以在计算机上实现的 Quine–McCluskey [4,5] 过程。 这种精确

最小化技术提出了素蕴含项和最小成本覆盖的概念，这将成为两级最小化的基石。 早期研究的另一个领域是有限状态机 (FSM) 的状态最小化和编码 [6-8]，这项任务是设计师的祸根。 逻辑综合的应用主要在于数字计算机设计。 因此，IBM和贝尔实验室在逻辑综合的早期自动化中发挥了关键作用。 从离散逻辑组件到可编程逻辑阵列 (PLA) 的发展加速了对高效两级最小化的需求，因为最小化两级表示中的项会减少相应 PLA 的面积。 MINI [9] 是一个早期的基于启发式的两级最小化器。 Espresso [10] 是对 MINI 的改进； 它使用 unate 递归范式 (URP) 作为许多优化步骤的中心主题。
然而，两级逻辑电路在超大规模集成（VLSI）设计中的重要性有限； 大多数设计使用多层逻辑。 用于设计多级电路的早期系统是 IBM 的逻辑综合系统（LSS）[11]。 它使用局部转换来简化逻辑。 LSS 和 Yorktown Silicon Compiler [12] 的工作促进了 20 世纪 80 年代逻辑综合研究的快速进展。 几所大学通过向公众提供他们的研究成果做出了贡献，其中最著名的是加州大学伯克利分校的 MIS [13] 和科罗拉多大学博尔德分校的 BOLD [14]。 十年之内，该技术迁移到电子设计自动化公司提供的商业逻辑综合产品中。
过去二十年，逻辑综合领域取得了巨大进步。 它极大地提高了数字电路设计的生产力，使团队能够充分利用通过减小特征尺寸而提供的大量晶体管。 本章简要概述了该领域取得的快速进展。 更全面的处理可以在 De Micheli [15]、Devadas 等人的书籍中找到。 [16]、Hassoun 和 Sasao [17]、Hachtel 和 Somenzi [18]。

2.2 行为和寄存器传输级综合

为了提高设计人员的工作效率，能够在更高的抽象级别指定和优化设计至关重要。 人们对使用硬件描述语言 (HDL) 进行行为指定的电路的综合进行了大量研究。 行为综合的目标是将行为 HDL 规范转换为寄存器传输级 (RTL) 规范，该规范可用作门级逻辑综合流程的输入。 行为综合的总体概述可以在[19-23]中找到。
行为优化决策由基于硬件资源数量和所需状态的成本函数指导。 这些成本函数提供了实现设计所需的组合和时序电路的粗略估计。
在行为优化工具中，前端解析器将设计的行为 HDL 描述转换为控制和数据流图（CDFG）[23]。 有时，会创建单独的控制流图 (CFG) 和数据流图。 该CDFG受到高级转换的影响，其中许多转换基于编译器优化[24]，例如常量传播、循环展开、死代码消除、公共子表达式消除、代码移动[25]和数据 流量分析。 在此步骤中执行的一些特定于硬件的优化包

括进行特定于硬件的转换（转换为执行 2 的幂乘法）、最小化逻辑级数以实现加速以及增加并行性。 调度、资源分配和共享的任务生成设计的 RTL 描述的 FSM 和数据路径。

调度[26]将操作分配给时间点，而分配将每个操作或变量分配给硬件资源。 调度确定 CFG 中状态开始和结束的位置，从而生成设计的 FSM。 调度通常先于分配，尽管它们可以交织在一起。 正在使用的调度算法包括尽可能快的调度算法和尽可能晚的调度算法。 其他算法包括强制导向调度[27]、列表调度[28,29]、基于路径的调度[30]和符号调度[31]。

给定时间表，分配操作可以优化实现设计所需的硬件数量。 分配由三部分组成：功能单元分配、寄存器分配、总线分配。 分配期间的目标是最大限度地共享硬件单元。 低功率分配已在[32]中进行了研究，而[33]报告了低功率的联合调度和分配。 低功耗行为综合已在[34-36]中进行了研究。

行为综合通常忽略所需控制逻辑的大小和延迟。 CDFG 表示与 RTL 网络中使用的表示显着不同。 因此，[37]中使用了行为网络图（BNG）。 BNG是一个RTL网络，能够表示非预定的行为描述。 由于布线延迟在 VLSI 设计中变得越来越重要，因此早期估计此类延迟非常有帮助。 用于高级综合的快速总线延迟预测器[38]满足了这一需求。 在[39]中，“不关心”的概念已被用于行为优化。 在可测试性的高级综合领域已经进行了大量研究。 有关详细信息，我们建议感兴趣的读者参阅有关该主题的调查论文[40]。 行为综合系统的例子可以在[41,42]中找到。 奥林巴斯[43]系统结合了行为和逻辑综合。

2.3 两级最小化

两级逻辑最小化可以说是逻辑综合的主力。 其早期和最直接的应用包括基于 PLA 的设计的逻辑最小化。 从那时起，它被广泛用于执行多级技术无关的逻辑优化中的节点优化。两级逻辑最小化是经典的unate覆盖问题（UCP）的一个实例[44-48]。 对于具有 n 个输入和 1 个输出的逻辑函数，覆盖矩阵具有 O(2n) 行（对应于最小项）和 O(3n/n) 列（对应于函数的素数）。 在 UCP 的典型解决方案中，我们迭代行和列优势以及行单例（本质素数）提取的步骤，直到覆盖矩阵无法进一步减少（称为“循环核心”）。 此时就需要使用分支定界等技术来解决循环核心。 早期的解决方案包括 Quine–McCluskey 方法 [4,5]。 最大独立素数集可用于限制解决方案成本。 20 世纪 90 年代初，开发了两种基于签名立方体计算的精确技术。 在这些方法之一[49]中，覆盖矩阵的大小被减小（行和列），从而产生更有效的解决方案。 另一种方法[50]基于降序二元决策图（ROBDD）[51]的使用。 创建所有素数和小项的特征 ROBDD，并根据 ROBDD 操作制定优

势步骤。 类似地，在[52]中，作者隐式创建了循环核心，其计算成本比以前的方法显着降低。 在[44-46]中，改进的边界和修剪方法在 SCHERZO 中被引入和实现。 戈德堡等人的方法。 [47,48]基于执行分支定界，其目标是证明给定的子空间不能产生更好的解决方案（消极思维），而不是试图通过分支找到更好的解决方案（积极思维）。 在[53]中，作者提供了一种技术，结合使用零抑制二元决策图（BDD）[54]（表示数据）和拉格朗日松弛来求解 UCP 的整数公式，产生了显着的改进。

Unate 覆盖可能是解决两级最小化问题的一种昂贵（尽管精确）的方法。 还开发了几种启发式方法。 MINI [9] 是早期启发式两级最小化器之一。 ESPRESSO [10] 对 MINI 进行了改进，利用 URP 作为操作的核心。 在这种启发式方法中，素数永远不会被枚举。 相反，运算是对素数的子集执行的。 在 ESPRESSO 中，Reduce（以有序方式减少立方体，使得新的一组立方体仍然是一个覆盖）、Expand（将立方体扩展为素数，删除被覆盖中其他一些立方体覆盖的立方体）的操作 ，和Irrundant（删除覆盖层中的冗余立方体）被迭代，直到不可能进一步改进。 这些算法基于对覆盖层中最二元变量的辅因子分解，直到获得单元叶为止。 对unate叶子进行高效的运算，然后将结果向上递归合并，直到得到对原cover的运算结果。 当归约、扩展和冗余迭代遇到局部极小值时，将调用 LASTGASP 算法，该算法尝试以选择性方式添加更多素数，以试图逃离局部极小值。 运行 LASTGASP 后，我们再次迭代减少、扩展和冗余，直到无法改进为止。 ESPRESSO 需要计算函数的补集，这对于诸如阿喀琉斯之踵函数之类的函数来说可能变得难以管理。 在[55]中，提出了减少偏移计算以避免这种潜在的立方体爆炸。

ESPRESSO 产生接近最佳的结果，与精确技术相比，速度显着加快。 为了进一步缩小 ESPRESSO 和精确方法之间的差距，可以使用 ESPRESSO [56] 的迭代应用。 在该技术中，在 ESPRESSO 迭代之后，从一开始就选择性地提取立方体，并将其视为无关立方体。 以这种方式迭代 ESPRESSO 已被证明可以弥合 ESPRESSO 和精确技术之间的（尽管很小）最优性差距，并具有适度的运行时间损失。

ESPRESSO-MV [57] 是 ESPRESSO 对多值最小化的推广。 两级逻辑最小化可应用于最小化互联网协议（IP）路由表[58,59]。 针对此应用定制的 ESPRESSO 硬件实现 [60,61] 已报告了显着的加速。

布尔关系是不完全指定函数 (ISF) 的概括，因为它们是一对多多输出布尔映射。 最小化此类关系涉及找到与该关系兼容的最佳两级逻辑函数。 Boo 的最小化精益关系可以被视为二元覆盖问题（BCP）。 在[62]中，提供了一个类似Quine-

McCluskey的过程来寻找布尔关系的最佳两级表示。 该解决方案基于应用于 BCP 的分支定界覆盖方法。 [63] 中提供了求解 BCP 的隐式技术，并与显式 BCP 求解器进行了比较。在[64]中，提供了一种分支定界BCP算法，其输入指定为多个ROBDD的结合。 多值关系的启发式最小化，使用 Brayton 等人的两级逻辑最小化范式。 [10]，在[65]中提出。 这种基于多值决策图 [66] (MDD) 的方法是在名为 GYOCRO [65] 的工具中实现的。

2.4 多级逻辑最小化

2.4.1 独立于技术的优化

逻辑功能的典型实际实现利用逻辑元件的多级网络。 例如，基于标准单元的逻辑网表就利用了这样的网络。 多级逻辑网络可以抽象为有向无环图（DAG），其中边代表线路，节点代表存储元素或组合逻辑基元。 通常，我们认为每个节点都有一个输出，尽管这也可以概括。 节点的组合逻辑可以用多种方式表示； 然而，两层覆盖是最常用的方法。

根据设计的 RTL 描述，我们可以构建相应的多级布尔网络。 在执行技术相关的优化之前，使用多种技术无关的技术对该网络进行优化。 与技术无关的优化期间的典型成本函数是逻辑函数分解表示的总字面量*计数（与电路面积密切相关）。

许多与技术无关的优化可以在多级布尔网络上执行。 我们提出了几种这样的优化，并简要讨论了每种优化的显着技术。

2.4.1.1 划分

如果 f = gh+r，则函数 g 是 f 的除数，其中 r 是余数函数，h 是商函数。 请注意，h 和 r 可能不唯一。 如果 r = 0，我们将该过程称为“因式分解”。 除法可以有两种类型：布尔[13,67,68]和代数[13,69,70]。 一般来说，布尔除法探索所有可能除 f 的函数。 因此，它的计算成本很高。 代数除法不太灵活，但速度极快，因为不执行布尔运算。 可以对g、h和r递归地进行除法以获得初始的多级网络。

2.4.1.2 核化

除法中的一项重要决策是除数 g 的选择。 内核[69]就是用于此目的。 内核是无立方†主除数。‡可以使用代数运算有效地计算内核。 使用双立方内核[71]可以显着加速内核处理，并且与成熟的内核提取相比，质量损失最小。
前面解释的划分技术可用于通过多种方式优化多级网络：

■ 如果网络中存在可以划分为另一个节点f的节点g，则将g代入f。

■ 通过提取一个或多个节点之间的公共子表达式，我们也许能够用更少的文字重新实现一个网络。 我们找到几个节点的内核，选择一个最好的内核，并用该内核的逻辑功能创建一个新节点。 现在，我们将新节点替换为设计中的所有剩余节点。

■ 我们可以通过将节点折叠到其扇出中来消除节点，以摆脱局部最小值并启用进一步的多级优化。

2.4.2 多级不关心

使用多级无关可以非常有效地减少网络的字面量。 通常在上述结构优化结束时调用基于无关优化的多级优化。 多级无关首先被计算为主要输入变量以及内部节点变量的函数。 然后，通过执行（通常基于 ROBDD）图像计算，我们发现多级图像不关心节点的 fanin 变量。 这允许我们使用新计算的无关项在节点上执行两级最小化，从而减少文字计数。

早期的多级无关计算方法 [72] 仅限于 NOR 门网络。 相应的无关计算技术被称为“转换”方法。 定义了两组允许功能*——最大允许功能集 (MSPF) 和兼容允许功能集 (CSPF)。 两者的缺点是它们被定义为全局函数。 此外，MSPF不兼容，因为如果某个节点j处的功能改变，则可能需要重新计算网络中其他节点的MSPF。 CSPF 取消了此限制。

在转导方法之后不久，就开发出了更通用的多级无关技术[73,74]。 对于这些方法，多级不关心被计算为外部不关心 (XDC)、可满足性不关心 (SDC) 和可观察性不关心 (ODC) 的析取。 每个输出的 XDC 通常被指定为全局功能，而电路的 SDC 则对网络中不会出现的本地逻辑条件进行编码。 网络

SDC 为 åi(yi Å fi)，其中 f 是网络节点 y 的逻辑函数。 计算 ODC ii

为 Õk (¶zk / ¶yi)，其中 ¶zk /¶yi 是主输出 zk 相对于 yi 的布尔差

并对全局空间中的最小项进行编码，其中 zk 对 yi 的变化敏感。

然后将上述不关心的析取成像回节点 yi 的本地 fanins，然后相对于这些本地不关心的覆盖 fi 最小化。 两级

最小化就是用于此目的。

如果一个节点相对于这些不关心的（包括 ODC）被最小化，它的结果

设计中其他节点的 ODC 的更改。 当节点一个接一个地最小化时，这可能并不重要。 然而，如果优化上下文是同时利用所有节点的无关上下文，则这并不能保证正确的结果。 对于此类应用，可以使用兼容输出无关（CODC）[75]。 在计算任何设计的 CODC 后，可以针对其 CODC 优化节点，而无需重新计算其他节点的 CODC。

前面概述的无关计算依赖于图像计算，这需要计算电路节点的全局 ROBDD。 这限制了无关计算的有效性。 最近，引入了近似 CODC [76] 和 ODC [77] 计算。 结果表明，通过在所考虑的节点的正向和反向上使用小拓扑深度的窗口，可以设计出鲁棒且有效的无关计算。 在[76]中，k = 4的深度导致了30%的加速和25%的内存需求减少，同时获得了传统CODC技术80%的文字减少。

该领域的另一个最新进展是引入了要区分的函数对集（SPFD）[78]。 SPFD 最初是在现场可编程门阵列 (FPGA) 优化的背景下引入的。 [79] 中确定了它们对一般逻辑网络优化的适用性。 节点的SPFD是ISF的集合。**因此，节点的SPFD比该节点的无关节点封装了更多的信息。 待区分的SPFD可以表示为二部图。 节点的 SPFD 对主输入空间中的点对集进行编码，必须区分这些点对或为其分配不同的函数值。 这些对在节点的 fanin 之间重新分配，从而在节点的 fanin 处产生新的 SPFD。 对这些图进行着色会导致 fanin 函数的新实现。 在[80]中，演示了基于 ROBDD 的基于 SPFD 的网络优化包的实现。 提供了导线替换和扇宁最小化的结果，比基于 CODC 的优化提高了约 10%。 研究还表明，SPFD 提供的灵

活性包含传统的基于无关优化的灵活性。 随后，基于 SPFD 的优化也在其他环境中得到了证明，包括 FPGA 综合的重新布线 [81]、功率和延迟最小化 [82]、使用多值 SPFD 的 PLA 网络的布线去除 [83] 以及拓扑约束 逻辑综合姐姐[84]。 [85] 中报道了 SPFD 的连续扩展。

2.4.3 技术相关的优化

到目前为止，第 2.4 节中讨论的与技术无关的多级逻辑优化已经利用了简单的成本函数，例如文字计数。 技术相关的优化将优化的技术无关的电路转换为给定技术中的门网络。 在技术映射期间和之后，简单的成本估算被更具体、实施驱动的估算所取代。 通过映射创建的电路结构对于最终的优化质量至关重要。 本节介绍技术映射以及技术映射后可用的几个局部优化。

2.4.3.1 技术映射

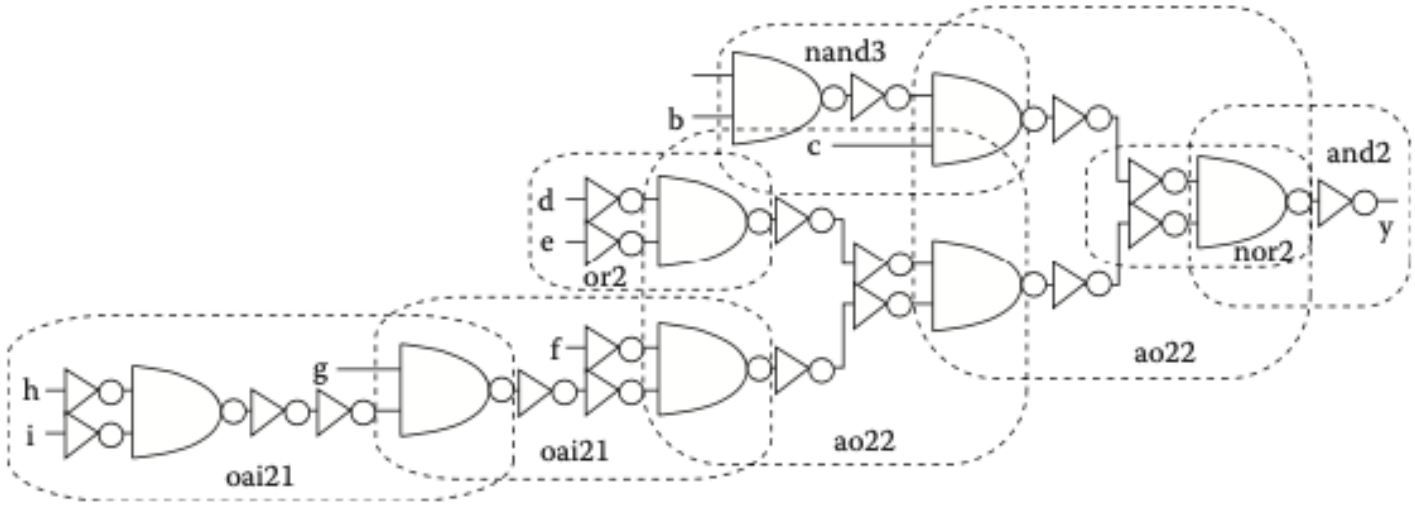
映射受到技术库中可用门（逻辑功能）、每个门的驱动器尺寸以及每个门的延迟、功率和面积特性等因素的限制。 第 2.5.1 节给出了有关技术库的更多详细信息。 技术映射的初始工作依赖于基于规则的启发式变换[86,87]。 这些后来被严格的算法方法所取代。 从概念上讲，技术映射的过程最好通过三个通用步骤来描述。 第一步称为“主题图构建”，将优化的独立于技术的电路分解为 DAG，由一组原始门（例如与非门和反相器）组成。 对于给定的电路，有许多逻辑上等效的分解，适当的选择对于良好的映射质量至关重要。 库中的逻辑门也被分解为同一组原语门。 这些被称为“模式”。 第二步称为“模式匹配”，生成库中逻辑门与主题图中每个原语门的匹配。 第三步称为“覆盖”，通过选择上一步中生成的模式子集来构建主题图的平铺。 覆盖的第一个约束是主题图的每个原始门都包含在至少一个选定的模式中。 第二个约束要求所选模式的输入是另一个所选模式的输出。 承保的目的是找到最低成本的承保。 成本函数可以是面积、延迟、功率或它们的组合。

匹配可以通过多种方法执行：使用图同构的结构匹配或使用 BDD 的布尔匹配。 在树的结构匹配的情况下，可以使用有效的字符串匹配技术[88]。 布尔匹配[89,90]更强大，因为它可以在任意输入/输出相位分配和（库门的）输入排列下检测独立于主题图的局部分解的匹配。*避免组合爆炸 由于排列和阶段，已经提出了签名[91,92]。

用于覆盖步骤的技术取决于我们是在 DAG 还是树上操作。 DAG 覆盖可以表示为 BCP [93]。 对于非常大的电路来说它不可行。 Keutzer [94] 描述了一种方法，其中主题图被分成一组无扇出树。 每棵树都经过最佳映射（针对区域）。 因此，DAG 覆盖通过一系列树映射步骤来近似。 树覆盖是使用动态规划来解决的，这会在成本函数的某些条件下产生最佳覆盖[95]。 基于最小面积树的技术映射可在主题图中的节点数量和库中的模式数量的时间多项式中求解。 Rudell [93] 扩展了技术映射以包括使用分级技术的时序。 Touati [96]考虑使用线性延迟模型的最佳延迟映射。 当延迟模型复杂时，覆盖过程需要在主题图中的每个节点存储面积-延迟成本曲线。 这可能会导致内存消耗较高。 罗伊等人。 [97]提出了一种压缩算法，该算法产生一个以用户指定的阈值为界的最佳解决方案。 延迟约束下的最小面积映射更加困难，并且没有已知的最佳算法[98]。 应用树木覆盖的主要缺点是双重的。 首先，将 DAG 分解为树会导致产生过多的小树，从而失去了原始 DAG 的最优性。 为了将 DAG 转换为树而切割的多扇出点的负载估计的不可预测性会导致延迟估计错误。 如果将延迟模型修改为与负载无关[99]，则 DAG 覆盖是可行的。 斯托克等人。 [100] 和 Kukimoto 等人。 [101]提出了涵盖基于增益和负载无关延迟模型下的 DAG 方法。 由于技术映射的质量对主题图的初始分解非常敏感，Lehman 等人。 [102]提出覆盖步骤内的分解。 提高树木覆盖构建的电路质量的一个简单方法是使用反相器对启发式[16]。

为了说明最小面积的基于树的技术映射，请考虑将 10 个组合逻辑门分解为由双输入与非门和反相器组成的模式的库，如图 2.1 所示。 每个门可能有多个分解（每个门仅显示一个）。 考虑一棵逻辑树及其相关的分解，使用双输入与非门和反相器将其分解为主题图，如图 2.2 所示。 观察到在一根电线上添加了两个反相器，以便能够检测到更多模式，如上述反相器对启发式所建议的那样。 主题图上库模式匹配的子集如图 2.3 所示。 请注意，门的输出处可能会发生多个匹配。 技术映射使用动态规划从一组候选匹配中选择门处模式的最佳匹配。 在这种方法中，在门处实现图案的面积成本最小是从树的输入到输出计算的。 模式的选择意味着在最终实现中也必须选择模式的输入。 因此，选择图案的最小面积成本可以计算为图案面积与在图案的输入处实现信号的最小面积成本之和。 由于解决方案是从输入到输出构建的，

因此后者的数量很容易获得。 图 2.4 显示了选择门来实现 y 处逻辑的两个选项。 选择nor2门（面积为3）需要输入到要实施的模式，面积成本分别为3和12，从而产生总面积成本为18。选择aoi21门（面积为3）需要输入到 实施模式的面积成本分别为 2 和 12，总面积成本为 17。



尽管面积和延迟一直是测绘中的传统目标函数，但功率现在也成为一个核心问题。 延迟约束下低功耗的树覆盖方法模拟了延迟约束下的最小区域覆盖。 蒂瓦里等人。 [103] 和 Tsui 等人。 [104]讨论以低功耗为重点的技术映射。

2.4.3.2 基于逻辑努力的优化

电路的延迟可以使用逻辑和电气努力的概念来表达[105]。 门的延迟可以表示为

d = t(p + gh) 这里，t 是与工艺相关的参数，而 p 是栅极的寄生延迟（由于栅极输出的源极/漏极电容而产生）。 参数g 是门的逻辑工作量，它表征门驱动输出电流的能力。 我们假设逆变器具有统一的逻辑努力。 其他门的逻辑工作取决于它们的拓扑，并描述了这些门在产生输出电流方面（与反相器相比）差多少，假设输入电容与反相器的电容相同。 参数 h 是栅极的电作用（或增益），是栅极引脚的输出电容与输入电容的比率。 请注意，p 和 g 与门的尺寸无关，而 h 则取决于尺寸。

在[105]中，作者通过为电路路径上的每个拓扑级别分配相等的延迟预算来最小化电路路径上的延迟。 由此产生的解决方案可以最大限度地减少延迟，并且解决方案不一定是面积最小的。 在[106]中，作者解决了扇出优化问题，以最小化源极栅极的输入电容，受接收器时序限制，而不考虑面积。 在[107]中，作者解决了在驱动器电容限制下最小化缓冲区的问题。 在[108]中，逻辑工作的思想被应用于技术映射，解决了多扇出点的负载分配问题。 [109,110]中引入了类似的互连工作概念，结合了逻辑大小调整和缓冲区插入的任务。 逻辑工作也可以应用于常规结构的设计，例如加法器[111,112]。

2.4.3.3 其他技术相关的优化

技术映射后，将执行一组与技术相关的优化。 门尺寸、门复制和德摩根变换对电路进行局部更改。 扇出优化的影响更为广泛。 关键路径重组可以产生重大的非局部变化。 这些方法具有以下共同策略。 第一阶段确定需要修改的电路部分以获得可能的好处。 第二阶段选择这些部分的子集进行更改，修改电路，并接受更改（如果它带来改进）。 这些步骤依赖于准确的增量时序分析来做出此决定（参见第 2.5.2.1 节）。

早期的尺寸调整工作主要关注晶体管并使用简单的延迟模型（例如 RC 树）。 这使得目标函数凸且易于优化[113-115]。 非线性规划技术也已被使用[116-118]。 然而，在 ASIC 环境中，逻辑优化只能调整门的尺寸，而不能调整单个晶体管的尺寸。 因此，下一波研究集中在这方面[119,120]。 此外，简单的延迟模型在现代技术中不再准确。 考德特等人。 [121]提供了关于浇口尺寸的出色调查。

扇出优化旨在将信号最佳地分配到各个输入引脚。 这对于驱动大量扇出的门至关重要。 伯曼等人。 [122]，胡佛等人。 [123]、Singh 和 Sangiovanni-Vincentelli [124] 描述了扇出问题的解决方案。 Kung [106] 解决了缓冲区大小丰富的库的扇出优化问题。 Rezvani 等人使用逻辑努力模型。 [107]讨论了面积和延迟的扇出优化工具。

巴特利特等人。 [125]是第一个关注在 SOCRATES 系统中延迟约束下最小化电路面积的人。 辛格等人。 [126]描述了一种使用加权最小切割启发式来识别再合成区域的策略。 权重是根据加速潜力和优化过程中产生的面积损失的估计来确定的。 Fishburn [127] 描述了一种加速组合逻辑的启发式方法。 Yoshikawa 等人进行了进一步的改进。 [128]。

2.5 逻辑综合的使能技术

dynamic programming. In this approach, the least area cost of implementing a pattern at a gate

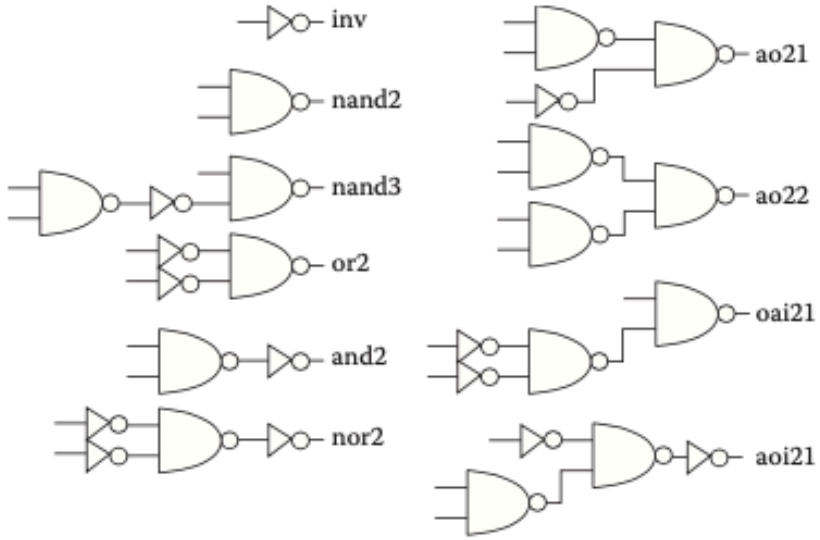
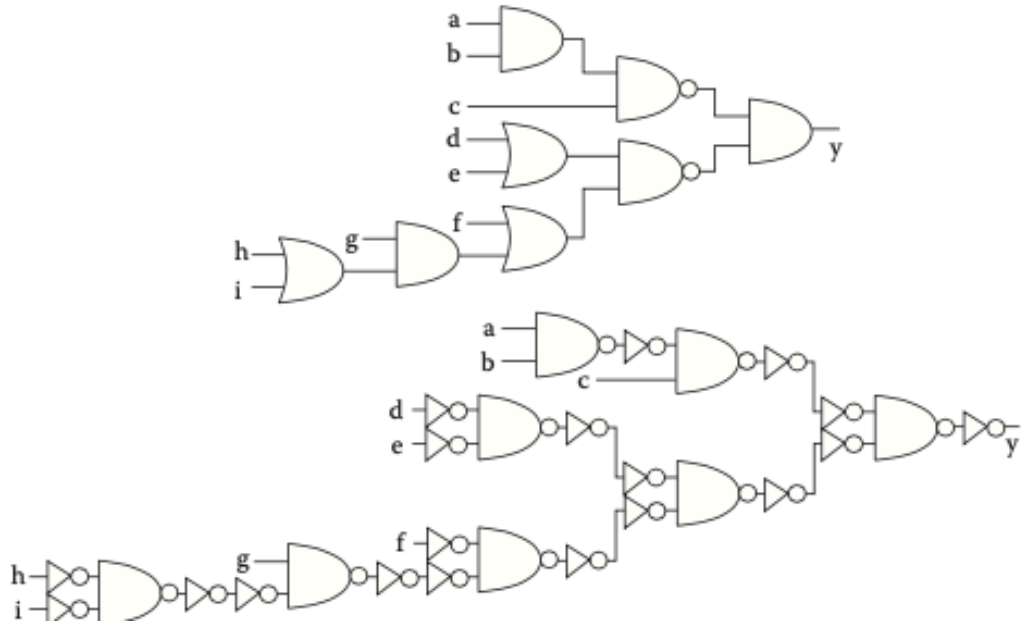


FIGURE 2.1 Decomposition of library gates into patterns.



一些技术在逻辑综合的快速发展中发挥了关键作用。 我们在本节中讨论一些支持性研究。逻辑综合在商业上取得成功的一个关键原因是各种半导体代工厂的技术库模型的可用性。 两种流行的库描述商业格式是 Synopsys 的 Liberty [129] 和高级库格式 [130]。 图书馆数据可以分为两个不同的部分：

1.技术数据：包括工作条件（电源和温度）和线负载模型等信息。 线负载模型是对典型网络上观察到的电容的统计估计，作为引脚数量和设计尺寸（以面积为单位）的函数。 当准确估计净电容所需的物理数据不可用时，它在综合过程中用作净电容的代理（更多信息请参阅第 2.7 节中的讨论）。

2. 库单元数据：库中的每个单元都具有准确的时序、功率和面积特征。 延迟是电路参数（例如输入转换、输出负载、阈值电压和临界尺寸）和工作条件（例如本地电源和接地电平、温度）的函数。 出于综合目的，延迟通常被建模为输入转换和输出负载的非线性函数。 它被指定为二维表。 向更小的几何形状的转变迫使使用多项式来捕获对许多参数的依赖性。 基于表的方法不会随着参数数量的增加而扩展内存使用量。 时序信息还包括诸如建立和保持约束之类的约束。 功率数据包括内部开关功率和泄漏功率。

过去十年争论的一个问题是标准单元库中不同逻辑函数的性质和数量[131,132]。 我们目前拥有包含数百个逻辑函数的库，每个函数都有多种大小。 最近的方法已经使用逻辑工作的概念来建模门延迟[105,133]。 这种方法假设门的面积是其负载的连续函数。 因此，可以调整门的大小以保持延迟恒定。 这是在[99]中首次提出的，并且在[102]中探讨了它在技术映射中的使用。 为了使这种方法发挥作用，库中的每个门都需要有足够多的尺寸，以便最后一步中离散化的误差最小[134,135]。

2.5.2 时序分析

一个好的逻辑优化系统需要使用高效的时序分析引擎。 在本节中，我们将简要介绍该领域的一些工作。

2.5.2.1 增量静态时序分析

第 6 章详细介绍了静态时序分析。逻辑综合系统中高效时序优化的一个关键要求是拥有一个与网表表示相结合的快速增量静态时序分析器。 技术相关的优化（第 2.4.3 节）在映射电路上运行。 通常，会对电路进行局部更改。 每次更改后，都会查询设计以检查时序是否有所改善。 如果更改的影响是局部的，则对整个设计调用静态时序分析会浪费计算资源。 例如，当调整门的大小时，该门的传递扇出锥体中的所有门以及直接扇入门都是到达时间变化的候选门。 定时的实际更新是由对定时信息的查询触发的，并且也由查询的位置决定。 这称为级别限制[137]。 然而，并非所有到达时间都可能改变，因为其他路径可以主导到达时间。 这称为优势限制。 增量静态时序分析的目标是执行更新因逻辑优化更改而失效的时序信息所需的最少计算。 时序信息包括净电容、到达时间、所需时间、裕量、转换时间和引脚到引脚延迟。 已发表的关于综合期间网表更改背景下的增量静态时序分析的著作之一是[136]。

2.5.2.2 错误路径

第 6 章详细介绍了错误路径。我们感兴趣的是在处理错误路径的上下文中执行的逻辑优化。 科伊策等人。 [138]证明冗余对于减少延迟不是必需的。 他们提供了一种算法（以作者的名字命名为 KMS 算法），该算法可以导出等效的冗余电路，而不会增加延迟。 他们的方法将著名的跳进位加法器转换为一种新颖的冗余设计。 萨尔达尼亚等人。 [139]研究电路结构与冗余和延迟的关系。 Kukimoto 和 Brayton [140] 将错误路径的概念扩展到所有输入到达考虑因素下的安全可替换性。

2.6 顺序优化

从历史上看，第一个逻辑综合方法本质上是组合的。 然后，开发了顺序技术来操纵和优化单个 FSM。 随后，开发了操纵 FSM 层次结构的顺序技术。 FSM 与有限自动机[141]类似，不同之处在于 FSM 产生输出信号，而自动机不产生输出，只是接受或拒绝输入序列。 Moore 机 [142] 是其输出函数仅取决于当前状态的 FSM，而 Mealy 机 [142] 是其输出取决于当前状态以及所施加的输入的 FSM。

组合电路实现布尔函数，该函数仅取决于应用于电路的输入。 非循环电路必然是组合的，而循环电路可能是组合的[144-147]。 然而，这种电路在实际中并不常用。

本节的其余部分讨论同步时序电路。 此类电路通常使用时钟锁存器或触发器来实现状态元件。 相反，异步时序电路没有时钟。 同步顺序行为可以以由存储器元件和组合逻辑组成的网表的形式指定，如状态转换图（STG）或转换关系。 转移关系通常使用 ROBDD 隐式表示。 为了避免内存爆炸，转换关系的 ROBDD 可以以分区的方式表示[148]。 通常，FSM 行为通过允许从给定输入状态和输入组合转换到不同状态（具有不同输出）来不确定地表达。 非确定性允许 FSM 行为的紧凑表示。 此外，转换和输出函数可能未完整指定。 在这种情况下，任何行为都被认为是允许的。 当然，任何实现都必须是确定性的并且完全指定的； 然而，在优化过程中放松这些限制使我们能够以紧凑的方式表达多种允许的行为。

2.6.1 状态最小化

给定一个具有符号状态和转换的 STG，我们执行状态最小化以组合等效状态（给定相同的输入序列，产生相同的输出序列的状态）。 这被转换为定点计算。 通常，对于完全指定的机器，在步骤 i 处，我们构造一组等效状态集，这些状态集形成原始状态空间的划分，并通过分离可以通过附加步骤区分的状态来细化该集合。 从包含状态空间中所有状态的初始集合开始，一直持续到收敛。 对于完全指定的机器，问题具有多项式复杂性。

对于不完全指定的机器，我们计算素数兼容集，对于这些机器，问题是非确定性多项式（NP）困难[148]。 [149] 中给出了确定性不完全指定 FSM 的早期方法之一。 在[150]中，表明可以使用素数兼容找到不完全指定机器的最小状态覆盖。 在[151]中，使用相容概念引入了一种有效的最小化方法。 在[152]中，报告了精确的和启发式的最小化结果，并在名为 STAMINA 的工具中实现。 [153] 中报告了兼容器的隐式（使用 ROBDD）计算，允许该技术处理极大量的兼容器。 在[154]中，作者基于隐式计算的广义兼容概念解决了非确定性 FSM 的最小化问题。

Kim 和 Newborn [155] 基于输入无关序列最小化 FSM 网络的早期工作之一是计算驱动机器的输入无关序列。 后来，Rho 和 Somenzi [156] 表明，对于不完全指定的机器，存在一个类似的机器，必须最小化它才能优化两个交互的完全指定的机器。 在[157]中，引入了计算通用 FSM 网络中所有输入无关序列的隐式过程。 Watanabe 和 Brayton [158] 的工作计算了 FSM 网络中任何 FSM 的最大允许行为集。 由此产生的不确定性机器被称为“E-机器”。 [159] 中报告了该机器的状态最小化。

在[160]中，解决了在基于语言包含的形式属性验证的背景下最小化交互FSM网络的问题。 [161] 解决了合成低功耗 FSM 的问题，而 [162] 报告了扩展 FSM 的基于无关的最小化。

FSM 的分解在[163]中得到解决，其目标是降低功耗。 使用 Kernighan–Lin 式[164]分区过程。 在[165]中，FSM分解的目标是I/O最小化，它是使用基于Fiduccia-Mattheyses的[166]分区方法来实现的。

2.6.2 状态分配

给定最小化的 STG，我们执行状态分配以将二进制代码分配给每个符号状态。 二进制代码的选择对最终综合设计的面积和功耗有很大影响。 状态分配可以被视为编码问题，使用输入编码、输出编码或输入输出编码。

编码问题分为两个步骤。 第一步，最小化多值表示，以及对用于符号状态的代码的一组约束。 下一步涉及找到满足这些约束的编码。 编码表示具有与最小化多值表示相同的大小。 可以为 FSM 的两级或多级实现执行编码。 在两级实现中，我们尝试最小化立方体的数量，而在多级实现中，我们尝试最小化设计中的文字数量。

输入约束是面嵌入约束 - 它们指定任何符号都可以分配给布尔 n 立方体的单个面，而该面不与其他符号共享。 输出约束是支配*或析取†约束。 在[167]中，表明找到满足输入约束的最小长度代码是NP困难的[148]。

在[168,169]中，报告了最小两级实现的基于输入编码的状态分配。 [170] 中给出了产生多级实现的解决方案。 Devadas 和 Newton [171] 的输出编码过程生成带有嵌套连接项的析取约束。 Villa 和 Sangiovanni-Vincentelli [172] 的 NOVA 算法利用了支配约束。 在[167,173]中，同时处理输入编码和输出编码约束。 [174-176]中报告了多级编码。 [177] 中报告了以最小化 FSM 转换关系的 ROBDD 大小为目标的状态分配。 [178] 探讨了并行状态分配，[179-183] 研究了低功耗状态分配。 可测试性的状态分配在[183]中进行了研究。

2.6.3 重新定时

一旦生成了顺序网表，就可以通过重定时来实现进一步的优化，这涉及在逻辑元件之间移动寄存器，以保持电路的顺序行为的方式。 重定时的目标可能是最小化时钟周期（最小周期重定时）或寄存器数量（最小区域重定时），或者最小化受最大时

钟周期约束的寄存器数量（受限最小区域重定时）。 重定时也可以与重新合成相结合。 在所有重定时方法中，电路都表示为图形，边代表电线，顶点代表门。 顶点的权重对应于门延迟，边的权重对应于该边上的寄存器的数量。

早期的重定时工作[184]基于混合整数线性规划方法。 后来，基于松弛的[185]方法被报道。 一般来说，重定时电路需要计算新的初始状态[186]。 可复位电路[187]可用于实现重定时设计，以便可以轻松计算初始化序列，该初始化序列可用于在上电后使机器进入已知状态。 在[188]中，在重定时方法中集成了最小化重定时后寻找新的等效初始状态的工作的技术。 在[189]中，描述了一种保证等效初始状态存在的最小区域重定时方法。

雷蒂米水平敏感设计的 ng 在[190-193]中进行了研究。 [194] 研究了最小周期和受限最小区域重定时的有效实现，而 Maheshwari 和 Sapatnekar [195] 则展示了有效的最小区域重定时。

在[196]中，该方法是通过将寄存器移动到电路边界来组合重定时和重新综合，然后优化组合逻辑并将寄存器移回设计中。 流水线逻辑的外设重定时[197]涉及重定时，使得内部边缘具有零权重，尽管外围边缘可能具有负权重。 一旦重定时合法化（所有权重都大于或等于零），我们就获得了功能等效电路（在应用初始化序列后表现出等效性）。 另一种重新综合方法[198]利用重定时引起的寄存器等效来优化逻辑。

重定时已在 FPGA 背景下进行了研究 [199,200] 以及可测试性保存和增强 [201,202] 背景下。 低功耗重定时在[203]中描述。 最近的重定时方法考虑了 DSM 延迟约束 [204]、互连和门延迟 [205] 以及时钟分配 [206,207]。 重定时用于指导[208]中的状态分配。

2.7 物理综合

随着 20 世纪 90 年代末技术的进步，逻辑综合过程中物理效应显然不容忽视。 我们简要总结了导致这场危机的一些技术问题。 晶体管尺寸的减小意味着门更小、速度更快。 缩小栅极尺寸意味着栅极输入引脚的电容会减小。 为了合并更多逻辑，路由资源无法以相应的方式扩展。 因此，金属互连层的数量从三层增加到更多（当前技术中常见的是 8-10 层）。 为了降低互连的电阻，必须增加横截面积。 如果增加导线的宽度，则可布线性会受到影响，因为在给定区域中可以封装的导线会减少。 因此，需要增加金属互连的高度。 这提高了互连线到相邻互连线的横向电容。 直到 20 世纪 90 年代中期，综合和物理

设计（布局和布线）的耦合一直非常弱。 一个简单的净重概念足以表明时间关键性对于放置来说就足够了。 网表签核的概念是代工厂、设计团队和工具供应商之间可行的商业模式。 合成期间门的延迟估计依赖于对门驱动的网络上的电容的准确估计。 净电容是引脚电容和导线电容（来自用于电气连接网络上所有引脚的互连的负载）的总和。 综合使用电线负载模型（第 2.5.1 节）作为电线电容的代理。 只要后布线电容不占总净电容的很大一部分，综合过程中延迟计算的不准确就不会造成问题。 在 0.25 mm 处，导线电容开始主导引脚电容。 因此，如果没有正确估计导线电容的方法，技术相关的优化效果就会降低。 反过来，这需要访问单元和引脚的物理布局。 布局算法的时序模型和分析能力相对较弱。 回想一下，静态时序分析主要是在综合的背景下开发的。 这些因素导致设计流程中的综合、布局和布线步骤不收敛。 此外，由于强大的横向耦合电容，在前几代技术中被忽视的信号完整性问题等影响已不能再被忽视。

如今，物理综合已成为综合中取代布局和后期时序校正的关键步骤。 该领域的早期工作侧重于在逻辑优化过程中纳入物理效应[209-212]。 科伊策等人。 [213]在一篇受邀论文中讨论了这种不连续性的技术和业务问题。 第二代的努力着眼于综合和布局的更紧密结合[214-216]。 第三代研究[217,218]使商业产品进入市场。 戈斯蒂等人。 [219]改进了之前[209]提出的同伴放置模型，以获得更好的面积和延迟性能。 逻辑工作的概念[133]支持推迟依赖于技术的优化直到某些布局信息可用的另一种观点。 物理综合使用两组不同的优化来实现最终设计的收敛。 布局期间的逻辑综合能够改变布局期间的网表结构。 这使得诸如有效缓冲长线和扩大弱门的尺寸等操作成为可能。 放置会影响单元拥塞、单元重叠以及线长度过长的网络。 这两种技术协同工作，比单独使用任何一种技术都能产生更好的结果。

卡拉格等人。 [220]讨论布局驱动的逻辑优化策略。 还研究了物理信息的逻辑重组[221]和技术映射[222]。 Murgai [225] 使用布局驱动的缓冲区优化研究区域恢复。 库德瓦等人。 [224]讨论了定义网表结构时在独立于技术的优化阶段测量可布线性的指标。 Saxena 和 Halpin [225] 讨论了将中继器纳入布局公式中。

2.8 多值逻辑综合

多值逻辑综合近年来受到广泛关注。 尽管对多值电路[226-228]以及存储器[55]实现的研究已有报道，但此类电路很难设计和制造，二值电路受到更多关注就证明了这一点。

多值技术的作用可以说是在综合的早期阶段。 执行多值逻辑优化后，可以将设计编码为二进制。 此时，可以应用传统的二值逻辑优化流程。

早期的多值技术包括将 ESPRESSO 推广到多个值 [57]。 ESPRESSO 中使用的减少偏移计算也有一个多值对应项[230]。 ROBDD 也已推广到多个值 [66]。 许多其他二值逻辑优化技术已推广到多个值，例如代数除法[231]、因式分解[232]、广义余因式分解[233]、无关[234]、冗余去除[236]、线去除 [237]和可满足性（SAT）[238]。 在[239]中，解决了非确定性多值网络简化问题。 使用非确定性多值关系来表达节点的最大灵活性。 在[240]中，报告了一种将多值代数运算简化为二进制的技术，从而显着加速多值网络上的运算。 [241,242] 中讨论了多值、多级综合技术。 除了编码之外，MVSIS [242] 还实现了许多早期技术。

除了动态重新编码[243]之外，还实现了静态[241]和动态[242]MDD变量重新排序。 基于MDD的合成[244]和功能分解[245]方法也已被报道。 ATPG 的 D 算法已推广到多个值 [246]。 在[247]中，作者描述了顺序多值网络的故障模拟。

2.9 总结

在本章中，我们探索了一些最先进的关键逻辑综合技术。 我们的重点是基于 CMOS 标准单元设计范例的主流合成方法。

我们在本章中介绍了高级、二级和多级综合以及顺序综合。 涵盖了相关的支持技术以及结合物理设计和逻辑综合的最新方法。

从早期的逻辑综合技术开始，一直在稳步推动抽象级别的提高。 与此同时，技术考虑迫使逻辑综合变得具有物理意识。 未来，我们将看到一些挑战。 改进的多值逻辑综合技术将为高级综合提供更有效的算法方法。 与此同时，逻辑综合生成的网表的质量（从物理设计的角度来看）将仍然很重要。 优化的容量和速度将继续成为平面数百万门综合的障碍。

底层实现风格的变化，例如从 CMOS 的转移，可以成为逻辑综合方法新研究的驱动力。 目前，量子计算的逻辑综合令人兴奋不已。

2.10 本条件的更新

在过去的十年中，逻辑综合领域面临着一些新的挑战。 显然，随着技术节点的进步，逻辑综合优化对物理设计的可行性产生重大影响。 较新的技术节点（22、16/14 和 10 nm）增加了对逻辑综合的需求，以在优化过程中准确估计新的物理效应。

与延迟和面积一样，电力已成为一个重要的问题。 泄漏和动态功率优化都已被广泛研究，并通过商业综合工具以不同程度的精度执行。

行业也发展到要求商业综合工具提供的所有逻辑综合优化都可以通过独立工具进行正式验证，以确保功能正确性。 这一新要求引发了人们对形式验证和逻辑综合的研究兴趣，特别是顺序优化。

在过去十年中，随着设计尺寸继续增加一个数量级以上，逻辑综合能力的提高已成为利用并行化和多核 CPU 的研究和开发的活跃领域。

2.10.1 物理效应

物理综合仍然是设计流程中的关键步骤，并且已针对新技术节点进行了积极增强。 互连布线估计仍然是逻辑综合期间正确建模互连延迟和拥塞的主要挑战。 首先，随着新技术节点的互连延迟继续变得更加重要，延迟估计得到了增强。 更精确的 CMOS 单元和互连延迟计算用电流源模型取代了非线性延迟模型 [248,249]。 其次，由于早期布局的不准确、影响最终布线、金属层分配和 VIA 估计的阻塞约束，逻辑综合期间的互连布线估计仍然具有挑战性。 话虽这么说，基于 Steiner 的路由估计已经得到了广泛的增强 [250,251]，并且仍然是逻辑综合中用于路由估计的主要技术。

物理综合侧重于与技术相关的优化，例如与布局、合法化和路由密切相关的重新映射、大小调整和缓冲[252-254]。 在较新的技术节点，堆叠中的不同金属层每单位长度具有非常不同的电阻特性。 对于给定距离，互连延迟根据所使用的金属层而显着变化（金属层越高，导线长度越长，延迟越小）。 物理综合已得到增强，以确定哪些延迟关键的网络连接应分配给更高的金属层——也称为层提升[255]。 最近的工作[256]探索了物理综合流中每个优化的不同拥塞模型和指标。 随着技术节点的进步，非默认规则 (NDR) 最近已添加到物理综合产品中，以解决电迁移限制并通过允许更宽的线宽和间距来减少寄生变化。 评估 NDR 的影响是另一个日益严峻的挑战。

传统逻辑综合的面积和延迟优化逻辑结构通常在可布线性和引脚可访问性方面表现出较差的物理特性。 较低几何形状的引脚可达性差可能会导致严重拥塞。 人们已经提出了各种技术来使逻辑综合生成简化布局和布线的逻辑结构，特别是对于数据路径和多路复用/选择逻辑[257-259]。 此外，逻辑综合面临着额外的挑战，即在流程的早期、技术无关的阶段（当设计尚未映射时）对物理效应进行建模。

2.10.2 布尔优化和技术映射

随着设计复杂性和尺寸的增加，并重用更多的 IP 核，冗余和重复的逻辑变得越来越普遍。 为了解决这个问题，逻辑优化借鉴了 ATPG 和先进的形式验证技术，包括 SAT [260]、结构散列、模拟和最近称为 SAT 扫描的混合技术，该技术利用底层图和逆图作为基础 各种逻辑转换。 总的来说，这些新的优化技术可以减少合成电路的最终门数和电路深度。 类似地，Mishchenko 在布尔网络中引入了灵活性计算技术 [261]，Chatterjee 描述了技术映射的改进 [262]，引入了新的布尔匹配算法并利用了顺序验证方面的进步。

除了重定时和时钟门控之外，顺序合并、顺序复制和顺序相位反转等顺序优化已在商业逻辑综合工具中普遍使用，并引发了验证方面的高级研究[263,264]。

2.10.3 低功耗

时钟门控是降低动态功耗最常用的设计技术。 时钟门控元件可以（1）由设计人员根据设计知识手动推断，或（2）由逻辑综合工具自动推断。 自动时钟门控最初是通过分析下一状态函数来确定时钟门控的条件[265,266]。 最近，时钟门控已扩展到识别时钟不需要使用模拟和 SAT 技术组合进行切换的条件 [267]。 顺序时钟门控在顺序分析的基础上进一步扩展了时钟门控[268-270]，为形式顺序验证带来了新的挑战[271]。 此外，功耗感知技术映射方法以及高效的开关活动计算也得到了研究[272]。 寄存器传输级分析和优化（如操作数隔离或逻辑门控）已有效降低高切换逻辑（如数据路径）的动态功耗[273]。 标准单元库已得到增强，包括具有共享时钟引脚的多位触发器（例如，2 位或 4 位），与等效的单位元件相比，可提供良好的动态功耗节省。 通过物理感知映射到这些单元可以降低功耗，而不会降低时序和拥塞。

另一方面，通过提供更广泛的库单元，并在不同电压阈值（Vts）下进行延迟/面积/泄漏权衡，可以降低泄漏功率[274-277]。 逻辑综合过程中的优化可以在时序关键路径上利用低 Vt 单元（具有高泄漏功率和快速延迟），在非时序关键路径上利用高 Vt 单元（具有低泄漏功率和慢延迟）。

最后，商业逻辑综合工具现在支持具有自动隔离和电平转换器单元插入功能的多电源电压设计技术。 业界还趋于一致的电源意图设计规范语言标准[278]。

2.10.4 容量和速度

由于利用 IP 重用设计更大、更复杂的 SoC 的行业趋势，运行时和内存使用方面的容量要求显着增加。 已经提出了基于粗粒度划分的分布式并行化[279]。 基于多线程的并行化最适合低级可并行算法。 Catanzaro 讨论了逻辑综合中使用的当前基于图的算法的并行化挑战，并介绍了基于低级分区的并行逻辑优化框架[280]。 多年来，分布式综合技术与低级多线程算法相结合，逐渐带来了切实的加速。

2.10.5 生态综合

在 IC 行业，通常的做法是制定工程变更单 (ECO) 来修复功能设计错误和时序收敛错误，甚至对已实现的设计进行小的功能更改。 随着设计规模和复杂性显着增加，手动实施 ECO 变得更加困难。 因此，ECO 综合最近被引入，并提供自动技术，以便在设计周期后期在现有的前掩模版或后掩模布局上实现 ECO [281]。 ECO 综合将首先确定 ECO 变化带来的最小功能差异。 人们已经进行了大量的研究，使用基于可满足性的证明和插值技术来正式确定两个实现之间的最小逻辑门集[282]。 这种功能逻辑差异（补丁逻辑）将被物理优化并适当地映射到可用单元上。 已经开发出特定的 ECO 感知逻辑和物理综合技术，以确保最佳实施[283-285]。

2.10.6 更新结论

在本次更新中，我们总结了过去十年逻辑综合领域的关键技术进展。 我们完全期望逻辑综合生成的网表质量对于确保物理设计流程的收敛仍然至关重要。 随着新技术要求的确定[286,287]，逻辑综合将继续面临新的挑战，这些挑战将推动令人兴奋的研究并为商业工具的创新创造机会。

从电路到寄存器传输级的功耗分析和优化

当今 VLSI 设计的复杂性和速度必然带来一定程度的功耗，如果不加以解决，将会导致难以忍受的散热问题。 这些电路的运行只能归功于不同设计抽象级别的积极降低功耗技术。 另一方面，移动设备和物联网的趋势推动了对节能电路的需求以及最大限度地延长电池寿命的要求。 为了应对这些挑战，电子设计自动化 (EDA) 的复杂设计方法和算法被开发出来。

CMOS 技术成功的关键特征之一是其固有的低功耗。 它使电路设计人员和 EDA 工具能够专注于最大化电路性能和最小化电路面积。 CMOS 技术的另一个有趣的特性是其良好的缩放特性，这使得特征尺寸能够稳步减小，从而允许在单个芯片上运行大量且异常复杂的系统，并在高时钟频率下工作。

随着 20 世纪 80 年代末第一批便携式电子系统的出现，功耗问题开始发挥作用。 在这个市场中，电池寿命是产品商业成功的决定性因素。 显而易见的是，每个芯片面积上有源元件的集成度不断提高将导致集成电路的能耗过大。 出于经济和环境原因，高功耗是不希望的，并且还会导致高散热。 为了保持这种电路在可接受的温度水平下工作，可能需要昂贵的散热系统。

除了全芯片功耗之外，也许更重要的是，电路中的局部区域（即所谓的热点）常常会散发过多的热量。 当检测到此类情况时，可以通过选择性地关闭电路中未使用的部分来缓解此问题。 术语“暗硅”用于描述集成电路中许多可用计算元件无法同时使用的情况[1]。 这些因素导致功耗上升，成为与性能和芯片尺寸同等的主要设计参数，并限制了 CMOS 技术的持续扩展。

为了应对这一挑战，过去二十年来，人们投入了大量研究来开发用于功耗优化的 EDA 工具。 最初的努力集中在电路和逻辑级工具上，因为这些级别的 EDA 工具更加成熟和可塑性更强。 如今，很大一部分 EDA 研究的目标是系统级或架

构级功耗优化（分别是《IC 系统设计、验证和测试的电子设计自动化》的第 7 章和第 13 章），鉴于其研究的广度，它们有望产生更高的总体影响。 应用。 与优化工具一起，需要有效的功率估计技术，既作为电路功耗满足某个目标值的绝对指标，又作为设计空间探索期间不同替代方案功率优点的相对指标。

本章概述了针对低功耗设计和综合提出的关键 CAD 技术。 我们从第 3.1 节开始，描述不同抽象级别的功率估计问题和方法，从而定义以下各节中介绍的工具的目标。在第 3.2 节和第 3.3 节中，我们回顾了电路和逻辑级别的功耗优化技术

分别抽象。

3.1 功率分析

鉴于功耗在电路设计中的重要性，需要 EDA 工具来提供电路的功耗估算。 在评估不同的设计时，需要这些估计来帮助确定最节能的替代方案。 由于多种替代方案可能需要功率估计，因此当可以保留估计的相对保真度时，有时会为了工具响应速度而牺牲准确性。 其次，在制造之前需要进行准确的功耗估计，以保证电路满足分配的功率预算。

获得功率估计比电路面积和延迟估计复杂得多，因为功率不仅取决于电路拓扑，还取决于信号的活动。

通常，设计探索是在每个抽象级别进行的，从而激发不同级别的功率估计工具。 抽象级别越高，有关实际电路实现的信息越少，这意味着对功率估计精度的保证越少。

在本节中，我们首先讨论 CMOS 电路中功耗的组成部分。 然后我们讨论如何在不同的设计抽象级别估计这些组件中的每一个。

3.1.1 CMOS 电路中的功率元件

数字CMOS电路的功耗一般分为三个部分[2]:

1. 动态功率（Pdyn）

2. 短路功率（Pshort） 3. 静态功率（Pstatic）

总功耗由这些组件的总和给出:

(3.1) P = Pdyn + Pshort + Pstatic

动态功率分量 Pdyn 与栅极输出 Cout 处负载电容的充电和放电有关。 这是可以集中在栅极输出端的寄生电容。 如今，该组件仍然是 CMOS 门功耗的主要来源。

作为说明性示例，请考虑图 3.1 中所示的反相器电路（为了形成通用 CMOS 栅极，底部晶体管 nMOS 可以由 nMOS 晶体管网络代替，顶部晶体管 pMOS 可以由 pMOS 互补网络代替 晶体管）。 当输入变低时，nMOS 晶体管截止，pMOS 晶体管导通。 这在电源电压和 Cout 之间创建了一条直接路径。 电流 IP 从电源流出，将 Cout 充电至电压电平 Vdd。 从电源汲取的电荷量为 CoutVdd，从电源汲取的能量等于 CoutVdd2。 电容器中实际存储的能量 Ec 只是其中的一半，Ec = 1⁄2CoutVdd2。 另一半则消耗在 pMOS 晶体管所代表的电阻中。 在随后的低到高输入转换期间，pMOS 晶体管截止，nMOS 晶体管导通。 这会将电容器 Cout 接地，导致电流 In 流动。 Cout 放电，其存储的能量 Ec 消耗在 nMOS 晶体管代表的电阻中。 因此，每次输出发生转变时都会消耗等于 Ec 的能量。 给定时间 T 内的 N 个门转换，该时间段内的动态功耗由下式给出

(3.2) Pdyn = Ec´ N / T = 1⁄2 Cout Vdd2 TN

在同步电路的情况下，门每个时钟周期进行的平均转换次数的估计值 α Tclk = 1/fclk 可用于计算平均动态功耗

(3.3) Pdyn = Ec´a´ fclk = 1⁄2CoutVdd2afclk

Cout 是 Cint、Cwire 和 Cload 三个部分的总和。 其中，Cint代表栅极的内部电容。 这包括连接到输出的漏极区域的扩散电容。 Cload 表示该逻辑门驱动的晶体管的栅极电容之和。 Cwire是用于互连栅极的布线的寄生电容，包括布线与衬底

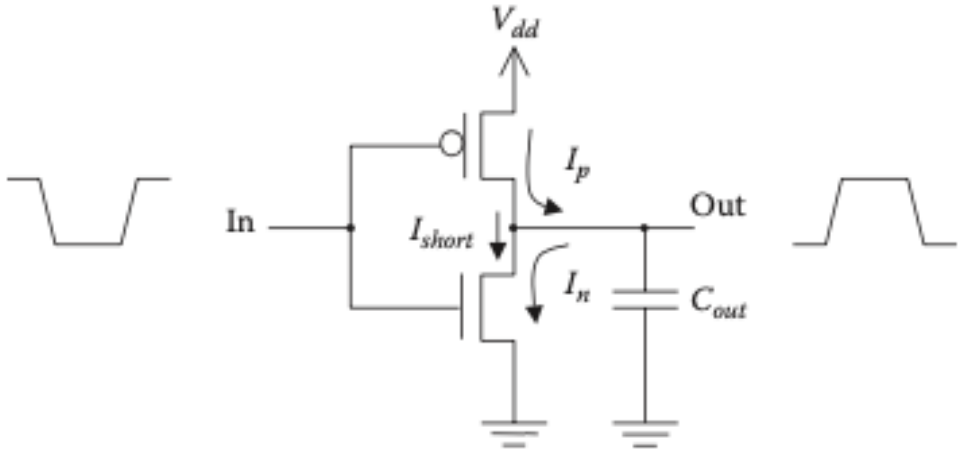


FIGURE 3.1 Illustration of the dynamic and short-circuit power components.

之间的电容、相邻布线之间的电容以及由于电场边缘效应而产生的电容。 术语 αCout 通常称为开关电容，它测量一个时钟周期内充电或放电的电容量。

短路功率分量 Pshort 也与栅极的开关活动有关。 在输入信号从一个电压电平转变到另一个电压电平期间，有一段时间pMOS和nMOS晶体管都导通，从而创建从Vdd到地的路径。 因此，每次门开关时，在此期间流经两个晶体管的电流都会消耗一定量的能量，如图 3.1 中的 Ishort 所示。 短路功率由输入电压 Vin 保持在 VTn 和 Vdd-VTp 之间的时间决定，其中 VTn 和 VTp 分别是 nMOS 和 pMOS 晶体管的阈值电压。 精心设计以最小化低斜率输入斜坡，即通过适当调整晶体管尺寸，可以将该组件限制在总功率的一小部分； 因此，它通常被认为只是二阶效应。 给定每个输出转换的短路电流所携带的平均电荷量 Qshort 的估计值，短路功率可通过以下公式获得

(3.4) Pshort =QshortVddafclk

静态功率分量 Pstatic 是由 MOS 晶体管中的漏电流引起的。 顾名思义，该组件与电路活动无关，并且只要电路通电就存在。 MOS晶体管（MOSFET）的源极和漏极区域可以与衬底形成反向偏置的寄生二极管。 这些二极管存在漏电流。该电流非常小，与动态功耗相比通常可以忽略不计。 即使 MOSFET 处于截止区，即当栅源电压 VGS 低于阈值电压时，由于源极和漏极之间的载流子扩散，也会出现另一种类型的漏电流， VT。 在此区域中，MOSFET 的行为类似于双

极晶体管，并且亚阈值电流与 VGS–VT 呈指数关系。 随着晶体管尺寸的减小，每个新技术节点的漏电流往往会增加，从而推高静态功耗的相对重量。 通过引入高 K 介电材料和新的栅极几何架构，这个问题得到了缓解 [3]。

另一种可能导致 CMOS 静态功耗的情况是当降级电压电平（例如，nMOS 传输晶体管的高输出电平）施加到 CMOS 栅极的输入时。 降低的电压电平可能使 nMOS 和 pMOS 晶体管都处于导通状态，导致短路电流持续流动。 这也是不希望的，在实践中应该避免。

此条件对于纯 CMOS 设计风格来说是成立的。 在某些专用电路中，即出于性能原因，可以使用替代设计风格。 当输出恒定在一个电压电平时，某些设计风格会产生电流，从而导致静态功耗增加。 一个例子是多米诺骨牌设计风格，如果门的输出恰好与预充电值相反，则需要在每个时钟周期对预充电节点进行充电。 另一个例子是伪 nMOS 逻辑系列，其中 CMOS 门的 pMOS 网络被始终导通的单个 pMOS 晶体管取代。 只要输出处于逻辑 0（即，当存在通过 nMOS 网络直接接地的路径时），这种逻辑样式就会呈现恒定电流。

3.1.2 电路级分析

电路级的功耗估计通常使用电路级模拟器获得，例如 SPICE [4]。 给定用户指定的输入值的代表性序列，模拟器求解电路方程以计算电路中所有节点的电压和电流波形。 通过对从电源汲取的电流值 Iavg 进行平均，模拟器可以输出电路消耗的平均功率，P = IavgVdd（如果使用多个电源，则总平均功率将是从所有电源汲取的功率之和 来源）。

在这个级别，可以使用电路器件的复杂模型。 这些模型可以精确计算功率的三个组成部分：动态功率、短路功率和静态功率。 由于电路是在晶体管级别描述的，因此不仅可以为 CMOS 计算正确的估计，还可以为任何逻辑设计风格甚至模拟模块计算正确的估计。 在电路布局和布线之后，仿真可以处理反向注释的电路描述，即具有真实的互连电容和电阻值。 由此获得的功率估计可以非常接近实际制造的电路的功耗。

问题在于，这种详细的模拟需要解决复杂的方程组，并且仅适用于小型电路。 另一个限制是输入序列必须非常短，因为模拟非常耗时。 因此，所得的功率估计可能无法很好地反映输入的真实统计数据。 由于这些原因，成熟的电路级功

率估计通常仅用于精确表征小电路模块。 为了将电路级仿真应用于更大的设计，可以采用非常简单的有源器件模型。 当然，这种简化意味着准确性的损失。 另一方面，大规模并行计算机将这些方法的适用性扩展到更大的设计[5]。

开关级仿真是一种限制情况，其中晶体管模型被简单地简化为开关，开关可以打开或关闭，并具有一些相关的寄生电阻和电容值。 这种简化的模型允许在更长的输入序列下估计更大的电路模块。 开关级仿真仍然可以相当准确地对功率的动态和短路分量进行建模，但对于泄漏功率来说，情况已不再如此。 在早期的技术节点，设计人员愿意忽略这个功率组件，因为它只占总功率的一小部分，可以忽略不计，但现在它的相对重要性正在增加。 然后必须使用专门定制的工具独立执行泄漏功率估计。 人们提出了许多不同的方法，其中一些将在下一节中介绍。

在中等复杂度的解决方案中，PrimeTime PX 是 Synopsys 静态时序分析工具的附加组件 [6]，提供的功耗估计精度接近 SPICE。 该工具采用给定晶体管尺寸的当前模型的表查找，并使用电路分区来独立地求解每个分区上的电路方程。 尽管由于不考虑不同分区之间的相互作用而引入了一些错误，但该技术极大地简化了要解决的问题，从而允许对大型设计进行快速电路级估计。

3.1.3 静态功耗估算

通常使用亚阈值模型来执行静态功耗分析，以估计每单位微米的泄漏，然后进行外推以估计整个芯片的泄漏。 通常，堆叠因子（器件堆叠导致的泄漏减少）是此扩展的一阶组成部分，用于修改所分析器件的总有效宽度[7]。 分析可以被视为通过堆叠因子对该总宽度的修改。

大多数泄漏分析工作都使用 BSIM2 亚阈值电流模型 [8]：

(VGS-VT-gIIIVSB+hVDS)æ -VDS ö (3.5) Isub=Ae nVTH ç1-eVTH ÷ èø

在这里

VGS、VDS 和 VSB 分别是栅极-源极、漏极-源极和源极-体电压，VT 是零偏置阈值电压

VTH 是热电压 (kT/q)

γ′ 是线性化身体效应系数

η 是漏极诱导势垒降低 (DIBL) 系数

A=mC (W/L)V2 e1.8 0ox effTH

BSIM2 泄漏模型包含了我们目前关注的所有泄漏行为。 总之，它解释了漏电随着阈值电压和栅源电压的降低呈指数增加。 它还解释了泄漏的温度依赖性。

通过将公式 3.5 应用到芯片中的每个晶体管来计算漏电流可能非常耗时。 为了克服这一障碍，人们研究了在更高抽象层次上处理泄漏的经验模型[9,10]。 例如，一个简单的经验模型如下[10]：

(3.6) Ileak = Ioff Wtot Xt Xs

在这里

Ioff 是在实际硅片上测量的单个晶体管每微米的漏电流

给定温度

Wtot 是晶体管总宽度（所有 N 和 P 器件的总和）

Xs 是基于晶体管堆栈泄漏小于的观察的经验堆栈因子

单一设备

Xt 是温度系数，用于将 Ioff 缩放至适当的结温

兴趣

Ioff 值通常在室温下指定（因此需要温度系数来转换为感兴趣的温度）。

静态功率的另一个主要组成部分是栅极泄漏。 仅当栅极氧化物足够薄以致穿过氧化物的直接量子隧道效应变得相当大时，栅极泄漏才有效地成为一阶效应。 栅极泄漏电流量与晶体管宽度成正比，因此，栅极泄漏估计的主要工作是估计总晶体管宽度 Wtot，类似于亚阈值电流所需的宽度。 栅极漏电流的确切值取决于栅极至源极和漏极至源极电压 VGS 和 VDS，而这些电压又取决于栅极输入值。 因此，晶体管宽度的基于状态的加权可用于更准确的栅极泄漏估计。 然而，这需要额外的工作来估计每个门的状态概率[11]。

目前，VT 足够高，使得亚阈值电流由总有功电流的动态分量主导。 另一方面，与栅极和阱泄漏分量相比，亚阈值电流在总待机电流中占主导地位。 随着氧化物厚度不断减小，人们担心栅极泄漏将成为泄漏的主要来源。 然而，金属栅极和 3D FinFET 等新技术的引入减缓了氧化层更薄的趋势，因此，至少在更多技术节点中，亚阈值泄漏将继续主导栅极泄漏。

3.1.4 逻辑电平功耗估计

第 3.1.1 节中促进逻辑级功率估计的一个关键观察结果是，如果门的输入上升得足够快，则每个输出转换所消耗的能量不依赖于晶体管的电阻特性，而只是一个 电容负载 Cout 的函数，栅极驱动，Ec = 1⁄2CoutVdd2。 给定寄生栅极和导线电容模型，允许在电路的门级描述中计算每个门 i 的 Cout_i，逻辑级的功率估计减少为计算每个门在给定时间段内进行的转换数量 ，即门的开关活动。 这对应于参数 N 或 α，我们只需分别应用方程 3.2 或 3.3 即可获得功率。

当然，这个估计仅指动态功率分量。 对于总功耗，我们必须考虑泄漏功率，这意味着上一节中描述的方法必须补充逻辑电平估计。 在许多情况下，逻辑级的功率估计用作指导逻辑级功率优化技术的指标，其通常以动态功率降低为目

标，因此，仅需要对此组件的估计。 开关活动计算有两类技术，即基于模拟的分析和概率分析（也分别称为动态和静态技术）。

3.1.4.1 基于仿真的技术

在基于仿真的开关活动估计中，使用高度优化的逻辑仿真器，允许快速仿真大量输入向量。 这种方法提出了两个主要问题：要模拟的输入向量的数量和用于逻辑门的延迟模型。

对门延迟进行建模的最简单方法是假设所有门和线路的延迟为零，这意味着电路中的所有转换同时发生。 因此，每个门对于每个输入向量最多进行一次转换。 实际上，逻辑门具有非零传输延迟，这可能会由于不同的信号传播路径而导致逻辑门的输入处的转换到达时间不同。 因此，门的输出可能响应于单个输入向量而多次切换。 图 3.2 显示了一个说明性示例。

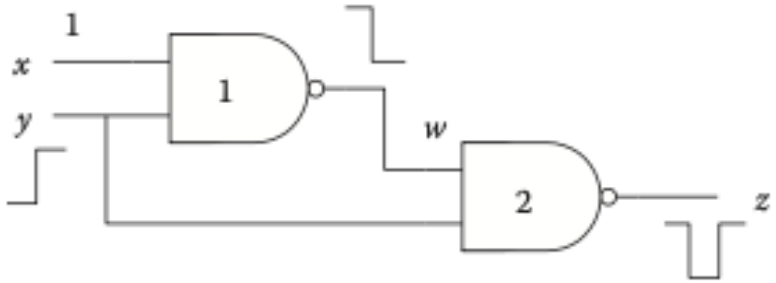


FIGURE 3.2 Example of a logic circuit with glitching and spatial correlation.

考虑到最初信号 x 设置为 1，信号 y 设置为 0，这意味着信号 w 和 z 都设置为 1。如果 y 转换为 1，则 z 将首先通过切换为 0 来响应此转换。 然而，大约在同一时间，w 切换到 0，从而导致 z 切换回 1。

这种杂散活动可能占整个开关活动的很大一部分，在具有高度再收敛信号的电路（例如乘法器）的情况下，该比例可能超过 50% [12]。 因此，逻辑级功率估计中的门延迟建模至关重要。 为了准确估计开关活动，模拟必须使用通用的

$$N \geq \left(\frac{z_{\theta/2} s}{\bar{p} \varepsilon} \right)^2$$

N 是输入向量的数量

p 和 s 是功率 zθ/2 的测量平均值和标准偏差，从正态分布获得

实际上，对于典型的组合电路以及合理的误差和置信水平，即使对于复杂的逻辑电路，获得总体平均开关活动所需的输入向量的数量通常也非常小（数千）。 然而，在许多情况下，需要电路中每个节点的准确平均开关活动。 低开关节点的高精度可能需要大量的输入向量。 设计人员可能需要放宽这些节点的精度，因为这些节点对电路的动态功耗影响较小。

尽管如此，当今的高度并行架构仍有助于快速模拟大量输入向量，从而提高此类基于蒙特卡罗的估计方法的准确性 [15]。

3.1.4.2 概率技术

概率技术背后的思想是直接传播输入统计数据以获得电路中每个节点的切换概率。 这种方法可能非常有效，因为只需要通过电路一次。 然而，它需要一个新的模拟引擎以及一组用于传播信号统计数据的规则。 例如，与门的输出为 1 的概率与将其每个输入设置为 1 的条件的交集相关。如果输入是独立的，那么这只是以下概率的乘积： 每个输入的计算结果为 1。对于任何逻辑门和不同的统计数据（即转移概率），都可以得出类似的规则。 尽管所有这些规则都很简单，但还有一系列新的复杂问题需要解决。 其中之一是延迟模型，如前所述。 在一般延迟模型下，每个门可以响应于单个输入变化而在不同时刻进行切换。 因此，我们需要计算每个时刻的切换概率。 假设图 3.2 电路中的门的传输延迟为 Δ1 和 Δ2，则意味着信号 z 将有一定概率在瞬时 Δ2 处进行转换，并有其他概率在瞬时 Δ1 + Δ2 处进行转换。 当然，信号 z 的总开关活动将是这两个概率的总和。

另一个问题是空间相关性。 当两个逻辑信号一起分析时，如果它们没有任何公共输入信号的支持，则只能假设它们是独立的。 如果存在一个或多个共同输入，我们就说这些信号是空间相关的。 为了说明这一点，再次考虑图 3.2 的逻辑电路，并假设两个输入信号 x 和 y 是独立的，并且有 px = py = 0.5 为 1 的概率。然后，pw 为 w 为 1 的概率， 就是 pw = 1 – pxpy = 0.75。 然而，pz = 1 – pwpy = 0.625 并不成立，因为信号 w 和 y 不是独立的： pwpy = (1 – pxpy).py = py – pxpy （注意 pypy = py），给出 pz = (1 – py + pxpy) = 0.75。 这表明不考虑空间相关性可能会导致计算中出现重大错误。

输入信号也可以是空间相关的。 然而，在许多实际情况中，这种相关性被忽略，要么是因为它根本不知道，要么是因为对这种相关性进行建模很困难。 对于能够解释信号之间所有类型的相关性的方法，请参阅[16]，但由于其高度复杂性，它不能应用于非常大的设计。

第三个重要问题是时间相关性。 在概率方法中，平均切换

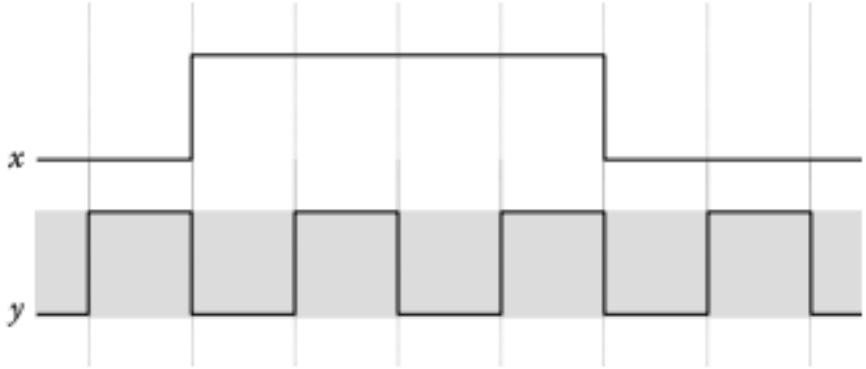
活动是根据信号从 0 转换到 1 或 1 到 0 的概率计算的。

相关性衡量信号在当前时刻为 0 或 1 的概率

值为 0 或 1。这意味着计算信号为 1 的静态概率是不够的，

我们需要直接计算转移概率，以便获取时间相关性

考虑到。 考虑图 3.3 中的信号 x 和 y，其中垂直线表示时钟周期。



这两个信号为 0 或 1 的周期数相同，因此，概率

信号为 1 的概率为 p1 = p1 = 0.5（而信号为 0 的概率为 p0 = p0 = 0.5）。 如果我们只有xy xy

考虑这个参数，从而忽略时间相关性，两者的转移概率

信号是相同的，可以计算为 a = p01 + p10 = p0p1 + p1p0 =0.5。 然而，我们可以

看到，在所描述的时间间隔内，信号 x 在三个时钟周期内保持低电平，仍然

高电平持续另外三个周期，并且具有一个带有上升沿转换的时钟周期和另一个时钟周期

具有下降的过渡。 对时钟周期数进行平均，得到 px00 = 38 = 0.375，

px01 = 18 = 0.125，p1x0 = 18 = 0.125，p1x1 = 38 = 0.375。 因此，实际平均开关激活

x 的效度为 a = p01 + p10 = 0.25。 至于信号 y，它永远不会保持低电平或高电平，进行转换 xxx

在每个时钟周期。 因此，p00 = p11 = 0 且 p01 = p10 = 4 = 0.5，实际平均切换 0110yy yy8

y 的活度为 ay = py + py = 1.0。 此示例说明了对时间相关性进行建模的重要性，并表明概率技术需要与转换概率结合使用，以实现准确的切换活动估计。

事实证明，这些问题的精确建模使得平均开关活动的计算成为 NP 难题，这意味着精确的方法仅适用于小型电路，因此没有什么实际意义。 已经提出了许多不同的近似方案[17]。

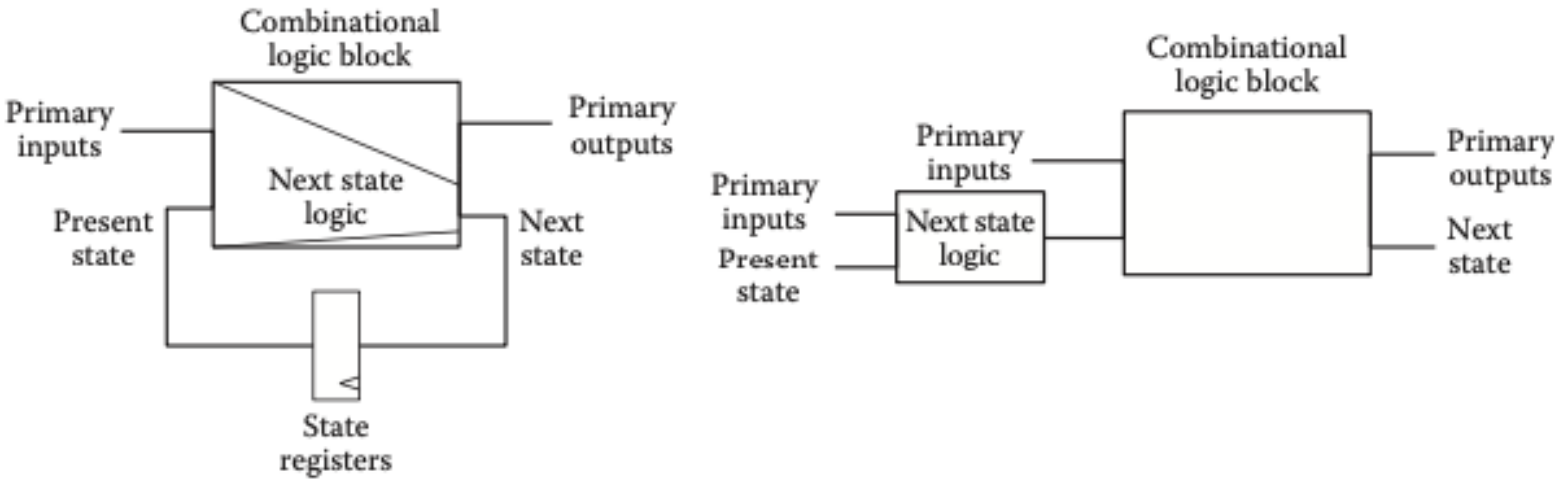


FIGURE 3.4 Creating temporal and spatial correlation among state signals.

3.1.4.3 时序电路

计算时序电路的开关活动要困难得多，因为必须以连续的方式访问状态空间。代表性的方式保证了状态信号概率的准确性。 对于基于仿真的方法，这一要求可能意味着输入序列太大，并且在实践中很难保证收敛。

概率方法可以有效地应用于时序电路，因为可以从电路中导出状态信号的统计数据。 精确的解决方案需要计算时序电路中所有状态对之间的转移概率。 在许多情况下，枚举电路的状态是不可能的，因为这些状态与电路中的顺序元件的数量呈指数关系。 一个常见的近似是计算过渡每个状态信号的概率[18]。 为了部分恢复状态信号之间的空间相关性，典型的方法是复制生成下一个状态信号的逻辑子集并将其附加到当前状态信号，如图 3.4 所示。 然后，将组合电路的方法应用于该修改的网络，忽略重复的下一状态逻辑块中的切换活动。

3.1.5 分析寄存器传输级别

在寄存器传输级（RTL），电路是根据从简单逻辑门到成熟乘法器的各种复杂性的互连模块来描述的。 此级别的功率估计确定每个模块输入端的信号统计数据，然后将这些值馈送到模块的功率模型以评估其功耗。 这些模型通常可通过模块库获得。 获得这些功率模型的一种方法是使用逻辑或电路级估计器来表征模块，这一过程称为宏建模。 我们参考《IC 系统设计、验证和测试的电子设计自动化》第 13 章，其中更详细地讨论了该主题。

3.2 电路级功率优化

从功耗建模方程中可以看出，鉴于其二次效应，电源电压的降低对功耗降低的影响最大。 这是降低功耗的最大来源，并广泛应用于整个半导体行业。 然而，除非伴随适当的工艺缩放，否则降低 Vdd 的代价是增加传播延迟，因此需要使用技术来恢复损失的性能。

降低工作频率 fclk 也可以降低功耗。 在性能要求较低的情况下，这可能是一个有吸引力的选择。 然而，电路的功率效率并未提高，因为每次操作的能量保持恒定。

一个更有趣的选择是通过重新设计电路来解决开关电容项 αCout，从而减少总体开关活动或减少总体电路电容或通过两者的组合，其中高电容节点中的开关活动为 可以通过将其替换为具有较低电容的节点中的较高开关来减少。

然而，漏电流引起的静态功耗带来了一系列不同的挑战。 如公式 3.5 所示，降低 Vdd 也会降低漏电流。 然而，减少转变次数以减少开关电容几乎没有什么好处，因为无论栅极输出中是否存在转变，都会消耗泄漏功率。 减少泄漏的最有效方法是有效地切断电路的电源，这称为电源门控。 其他技术是由漏电流与阈值电压VT 的关系激发的。 提高阈值电压会降低漏电流。 公式 3.6 激发了利用漏电流与电路拓扑关系的其他技术。

下面，我们将简要讨论为解决上述每一点而开发的关键电路级技术。 有大量已发表的作品详细介绍了这些技术和其他技术，建议感兴趣的读者从更深入地涵盖这些主题的书籍 [19] 和概述论文 [20] 开始。

3.2.1 晶体管尺寸

栅极的传播延迟（通常简称为延迟）取决于栅极输出电阻和总电容（互连和负载）[2]。 晶体管尺寸（或栅极尺寸）有助于通过增加栅极强度来减少延迟，但代价是增加面积和功耗。 相反，通过降低栅极强度，可以降低开关电容，从而降低功率，但代价是增加延迟。 这种权衡可以针对定制设计手动执行，也可以通过使用自动化工具来执行。

直到最近，高性能 CPU 的大部分通常都是定制设计的。 即使现在，高性能设计中对性能最关键的部分也混合了合成和定制设计的部分。 此类设计可能涉及手动调整晶体管，以扩大关键路径上的驱动器尺寸。 如果不必要地增大了太多晶体管的尺寸，某些设计可能会在电路功率延迟曲线的陡峭部分上运行。 此外，所使用的逻辑系列的选择（例如静态逻辑与动态逻辑）也会极大地影响电路的功耗。 传统上对性能的重视往往会导致过度设计，从而浪费功率。 然而，对低功率的强调促使人们识别这种功率浪费的来源。 这种浪费的一个例子是电路路径的设计速度比其需要的速度快。 对于综合模块，综合工具可以通过缩小此类路径中的设备尺寸来自动降低功耗。 另一方面，对于手动设计的块，缩小尺寸可能并不总是能够完成。 因此，自动化缩小规模工具可以产生巨大影响。 与手动设计方法相比，此类工具的优点是节省功耗并提高生产率。

现在，使用多阈值电压（“multi-VT”）结合传统晶体管尺寸来降低泄漏功率已成为一种广泛使用的设计技术。 这里的主要思想是在关键路径中使用较低 VT 晶体管，而不是大型高 VT 晶体管。 然而，该技术由于 VT 较低而增加了亚阈值泄漏。 因此，有选择地使用低VT晶体管并优化其使用以实现电容电流和漏电流之间的良好平衡以最小化总电流非常重要。 这种考虑现在是后综合或布局后自动化工具和流程的一部分，可识别低 VT 和高 VT 替代。 例如，在布局后时序分析之后，布局工具可以在增量模式下运行以执行两件事：将低 VT 单元插入关键路径以提高速度，并将较高 VT 单元插入非关键路径以再次降低泄漏。

定制设计人员可以灵活地手动选择晶体管参数来生成定制单元。 然而，大多数综合设计只能选择从单元库中的不同门或单元中进行挑选。 这些库通常具有从高性能（高功率）到低功率（低性能）范围内的单元选择。 在这种情况下，晶体管尺寸问题简化为在初始合成流程期间或在后合成流程中调整初始选择的最佳单元选择问题。 这是一个活跃的学术研究领域[21]，也是商业工具中的一个关键优化选项[19]。

3.2.2 电压缩放、电压岛和可变 VDD

如前所述，降低Vdd是降低功耗最有效的方法。 因此，业界已稳步转向更低的 Vdd。 事实上，即使考虑到维持计算吞吐量所需的系统架构修改，降低电源电压也是低功耗运行的最佳选择。 电压缩放的另一个问题是，为了保持性能，阈值电压也需要按比例缩小，因为电路速度大致与 (Vdd–VT) 成反比。 通常，如果速度不会受到太大影响，Vdd 应大于 4VT。 随着阈值电压降低，亚阈值漏电流呈指数增加。 VT 每降低 0.1V，亚阈值电流就会增加 10 倍。

在纳米技术中，随着VT的进一步降低，亚阈值电流已成为整个芯片电流的重要组成部分。 当特征尺寸为 0.18 m 及以下时，泄漏功率开始侵蚀较低 Vdd 的优势。 此外，动态电路、高速缓存、感测放大器、PLA等的设计在较高的亚阈值泄漏电流下变得困难。 较低的 Vdd 还会加剧噪声和可靠性问题。 为了对抗亚阈值电流的增加，已经开发了多种技术，如第 3.2.5 节中所述。

电压岛和可变 Vdd 是可在电路级使用的电压缩放的变体。 电压缩放主要取决于技术并且通常应用于整个芯片。 电压岛更适合片上系统设计，将具有不同性能要求的不同功能模块集成到单个芯片上。 有关电压岛的更多详细信息，请参阅 RTL 功耗分析和优化技术一章。 可变电压和电压岛技术是互补的，可以在同一块上实现并同时使用。 在可变电压技术中，电源电压根据吞吐量要求而变化。 对于较高吞吐量的应用，电源电压随着工作频率的增加而增加，对于较低吞吐量的应用则相反。 有时，这种技术也用于控制功耗和表面温度。 片上传感器测量温度或电流要求并降低电源电压以减少功耗。 通过应用多阈值电压器件、多通道长度器件以及堆叠和停车状态技术，可以在器件级实现泄漏功率减轻。 以下部分详细介绍了这些技术。

3.2.3 多阈值电压

多阈值电压（最常见的是高 VT 和低 VT 选项）多年来已在许多（如果不是大多数）CMOS 工艺上可用。 对于任何给定的电路块，设计人员可以选择使用一个或另一个 VT 或两者的混合。 例如，默认使用高 VT 晶体管，然后选择性地插入低 VT 晶体管。 由于待机功耗对低 VT 晶体管的数量非常敏感，因此它们的使用量约为晶体管总数的 5%–10%，通常仅限于修复关键时序路径，否则会发生泄漏 功率可能会急剧增加。 例如，如果低VT值比高VT值低110mV，则低VT值使用20%将使芯片待机功耗增加近500%。 低 VT 插入不会影响有源功率组件或设计尺寸，而且它通常是布局后阶段最简单的选择，可将布局扰动降至最低。 默认使用高 VT 晶体管并仅选择性使用低 VT 晶体管的明显候选电路是 SRAM，其功率主要由泄漏决定，并且较高的 VT 通常还会提高 SRAM 稳定性（较长通道也是如此）。 低VT晶体管的主要缺点是，由于掺杂而导致的延迟变化在高阈值晶体管和低阈值晶体管之间不相关，因此需要更大的时序裕度，并且需要额外的掩模步骤，这会产生额外的工艺成本。

3.2.4 长沟道晶体管

使用比标称沟道长度更长的晶体管是降低泄漏功率的另一种方法[22]。 例如，通过将晶体管拉得比最小尺寸的晶体管长 10 nm（长 L），DIBL 就会衰减，并且在 90 nm 工艺上漏电可以减少 7×-10×。 通过这一改变，可以在保持性能的同时消除总 SRAM 泄漏分量的近 20%。 由于沟道电阻增加而导致的驱动电流损耗（大约 10%–20%）可以通过增加宽度

来补偿，或者由于影响是在单个栅极级上，因此对于大多数设计来说可以忽略不计。 22]。 使用长 L 对 SRAM 特别有用，因为它们的整体性能对晶体管延迟相对不敏感。 如果明智地使用，它也可以应用于其他电路。 与多阈值电压相比，长沟道插入具有相似或更低的工艺成本——它表现为尺寸增加而不是掩模成本。 它可以降低工艺复杂性，并且不同的通道长度可以跟踪工艺变化。 它可以适时应用于现有设计以限制泄漏。 一个潜在的损失是栅极电容的增加。 总有功功率不增加如果受影响门的活动因子较低，则影响显着，因此在选择目标门时也应考虑这一点。

目标门的选择由两个主要标准驱动。 首先，晶体管必须位于具有足够时序余量的路径上。 其次，应首先从所选路径中选择泄漏最高的晶体管。 第一个标准确保满足绩效目标。 第二个标准有助于最大限度地减少泄漏功率。 为了利用所有可用的正时序裕度并避免错误，最建议在设计后期阶段进行长 L 插入。

可以通过使用使用长L晶体管设计的标准单元或通过从晶体管级设计中选择单独的晶体管来执行长L插入。 只有后者才适用于全定制设计。 两种方法各有优点和缺点。 对于单元级方法，低性能单元采用长L晶体管设计。 为了减少泄漏，非关键路径上的高性能单元被替换为具有长 L 的低性能单元。 如果占用空间和端口位置相同，则此方法可以简化物理聚合。 不幸的是，这种方法需要更大的细胞库。 它还需要一种微调的合成方法，以确保选择长L单元，而不是性能较低的标称沟道长度单元。 晶体管级流程有其自身的优点。 统一流程可用于自定义块和自动放置和路由块。 尽管如上所述有长L的空间，但只需要一个标称细胞库。

3.2.5 拓扑技术：堆叠和停车状态

另一类技术利用了泄漏功率对逻辑门拓扑的依赖性。 这种技术的两个例子是堆叠和停放状态。 这些技术基于以下事实：“关闭”晶体管的堆叠比堆叠中只有单个器件关闭时的泄漏要少。 这主要是由于堆叠中截止晶体管中栅源电压 VGS 的自反向偏置造成的。 图 3.5 说明了四个串联晶体管的电压分配[10]。 正如我们所看到的，当晶体管靠近堆栈顶部时，VGS 的负值更大。 具有最负 VGS 的晶体管是堆栈泄漏的限制器。 此外，由于反向偏置的体源电压（体效应），顶部三个晶体管的阈值电压增加。

自反向偏置和体效应均以指数方式减少泄漏，如公式 3.5 所示。 最后，亚微米 MOSFET 的整体漏电流也受到 DIBL 效应的调节。 随着 VDS 的增加，源极和漏极之间的沟道能垒降低。 因此，漏电流随 VDS 呈指数增加。

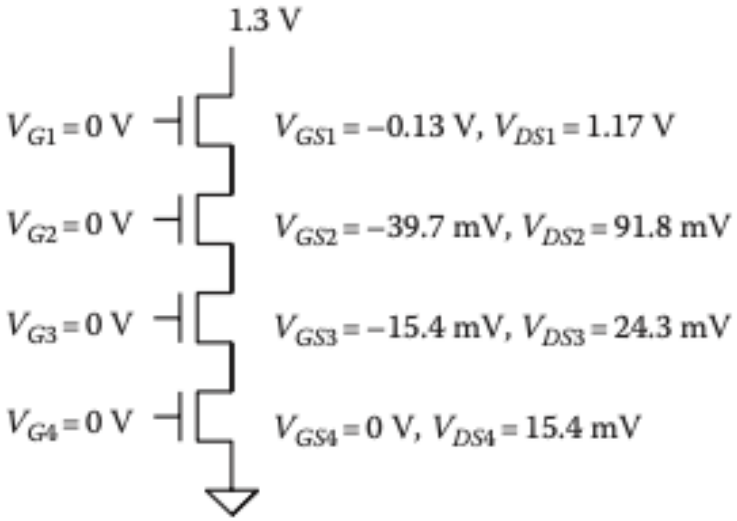


FIGURE 3.5 Voltage distribution of stacked transistors in OFF state.

这三种效应的结合导致从堆栈顶部到底部的 VDS 分布逐渐减小，因为所有串联的晶体管必须具有相同的漏电流。 因此，显着降低的 VDS（堆叠晶体管的有效泄漏）比单个晶体管低得多。

表 3.1 量化了全静态四输入 NAND 门的亚阈值泄漏电流的基本特性。 最小泄漏条件发生在“0000”输入向量（即所有输入 a、b、c 和 d 均处于逻辑零）。 在这种情况下，所有 PMOS 器件均“ON”并且输出节点和地之间存在通过四个 NMOS 器件堆叠的漏电路径。 当所有 NMOS 器件都导通并且由四个并联 PMOS 器件组成的泄漏路径存在于电源和输出节点之

间时，“1111”输入情况下会出现最大漏电流。 最小和最大泄漏条件之间的叠加因子变化反映了泄漏对输入矢量的依赖程度。 在四输入 NAND 情况下，我们可以得出结论，最小和最大情况之间的泄漏变化约为 40 倍（参见表 3.1）。 这些值是使用精确的类似 SPICE 的电路模拟器在 0.18 μm 技术库上测量的。 平均漏电流的计算基于所有 16 个输入向量均等概率的假设。

堆叠技术利用前面描述的效果来增加堆叠深度[23]。 示例之一是睡眠晶体管技术。 该技术在堆栈中插入一个额外的串联设备，并在堆栈整体关闭的周期内将其关闭。 这是以检测关闭状态的额外逻辑以及额外设备的额外延迟、面积和动态功耗为代价的。 因此，该技术通常在更高的粒度级别上应用，使用在更大的逻辑块之间共享的睡眠晶体管。 事实上，大多数实际应用都以非常高的粒度应用该技术，可以轻松确定存储器和 ALU 等大型电路块的睡眠状态（即非活动状态）。 在这个层面上，该技术可以被视为类似于电源门控，因为当不需要电路输出（即不活动）时，它将电路块与电源轨隔离。 功率门控是一种非常有效且越来越流行的减少泄漏技术，并且得到商业 EDA 工具的支持 [24]，但它主要应用于微架构或架构级别，因此这里不再进一步讨论。

停车状态技术背后的主要思想是在不使用时强制电路中的门进入低泄漏逻辑状态[25]。 如前所述，漏电流高度依赖于堆栈中导通和截止晶体管之间的拓扑关系，因此漏电流取决于输入值。 该技术避免了额外堆叠设备的开销，但需要额外的逻辑来生成所需的状态，这会产生面积和开关功率成本。 该技术不适用于随机逻辑，但如果仔细实施结构化数据路径和存储器阵列，它可以在关闭状态下节省大量泄漏功率。

考虑到所引入设备的面积和交换开销，人们在使用这些技术时需要小心。 当少量晶体管可以为宽逻辑锥体添加额外的堆叠长度或对其电源进行门控时，堆叠是有益的。 由睡眠晶体管或功率门控引入的延迟还意味着，只有当目标电路块保持在关闭状态足够长的时间以弥补驱动进入和离开关闭状态的转换的开销时，这些技术才是有益的。 这些限制可以通过仔细的手动干预或通过自动化工具的适当设计意图提示来克服。

泄漏功率降低仍将是一个活跃的研究领域，因为泄漏功率本质上是通过电压缩放限制动态功率降低的因素。 随着晶体管技术缩小到更小的特征尺寸，使得在同一芯片上集成更多数量的器件成为可能，材料和晶体管设计的进一步进步预计

将允许对功率（动态和泄漏）和 性能权衡。 这需要与功率分析的进步相结合，以了解纳米级效应，而纳米级效应迄今为止还不足以保证详细的功率模型。 与这些模型相结合，需要开发新的电路技术来解决这些影响。 随着这些电路技术获得更广泛的可接受性和适用性，将这些技术纳入自动合成流程的算法研究将继续进行。

TABLE 3.1 Stacking Factors of Four-Input NAND

	Minimum	Maximum	Average
Stacking factor Xs	1.75	70.02	9.95
Input vector (a b c d)	(1 1 1 1)	(0 0 0 0)	—

3.2.6 逻辑类型

通常认为动态电路比静态电路消耗更多的功率。 虽然具有恒定输入的静态 CMOS 门的功耗仅限于泄漏功率，但动态门可以在某些输入条件下连续对其输出电容进行预充电和放电。

例如，如果图 3.6a 中的 NAND 门的输入稳定，则输出也稳定。 另一方面，图3.6b的动态与非门，在恒定输入A = B = 1的情况下，将不断升高和降低输出节点，从而导致高能耗。

由于多种原因，动态逻辑系列在许多高速、高密度设计（例如微处理器）中是首选。 首先，动态门需要更少的晶体管，这意味着它们不仅占用更少的空间，而且表现出更低的电容负载，因此可以提高运行速度并降低动态功耗。 其次，输出节点的评估可以仅通过N型MOSFET晶体管进行，这进一步有助于性能的提高。 第三，从Vdd到地之间不存在

直接路径，从而有效地消除了短路功率分量。 最后，动态电路本质上不会产生任何杂散活动，这可以显着降低功耗。 然而，动态电路的设计提出了几个问题，这些问题已通过不同的设计系列得到解决[26]。

传输晶体管逻辑是另一种设计风格，其低功耗优点已被指出，主要是由于输入信号路径的电容负载较低。 问题是这种设计风格可能意味着更大的电路。

考虑到时序电路元件对逻辑芯片功耗的贡献，时序电路元件因其所选择的逻辑风格而受到特别关注。 这些存储元件（触发器或锁存器）是时钟网络的端点，并且构成芯片开关电容的最大部分，因为它们的输入切换速率（每个时钟沿）及其总数（ 特别是在管道深度较浅的高速电路中）。 因此，这些存储元件受到了很多关注[27]。 例如，双边沿触发触发器已被提议作为传统单边沿触发触发器的低功耗替代方案，因为它们提供了将有效时钟频率降低一半的机会。 设计简易性、设计可移植性、可扩展性、鲁棒性和噪声敏感性之间的权衡，更不用说面积和性能的基本权衡，要求只有在仔细考虑特定设计后才能做出这些选择 应用。 这些权衡也随着技术节点的不同而变化，因为选择时必须考虑泄漏功耗。

一般来说，只要基本电路设计技术不断发展以克服限制或利用技术扩展提供的机会，人们就可以期待电路风格的研究和创新继续下去。

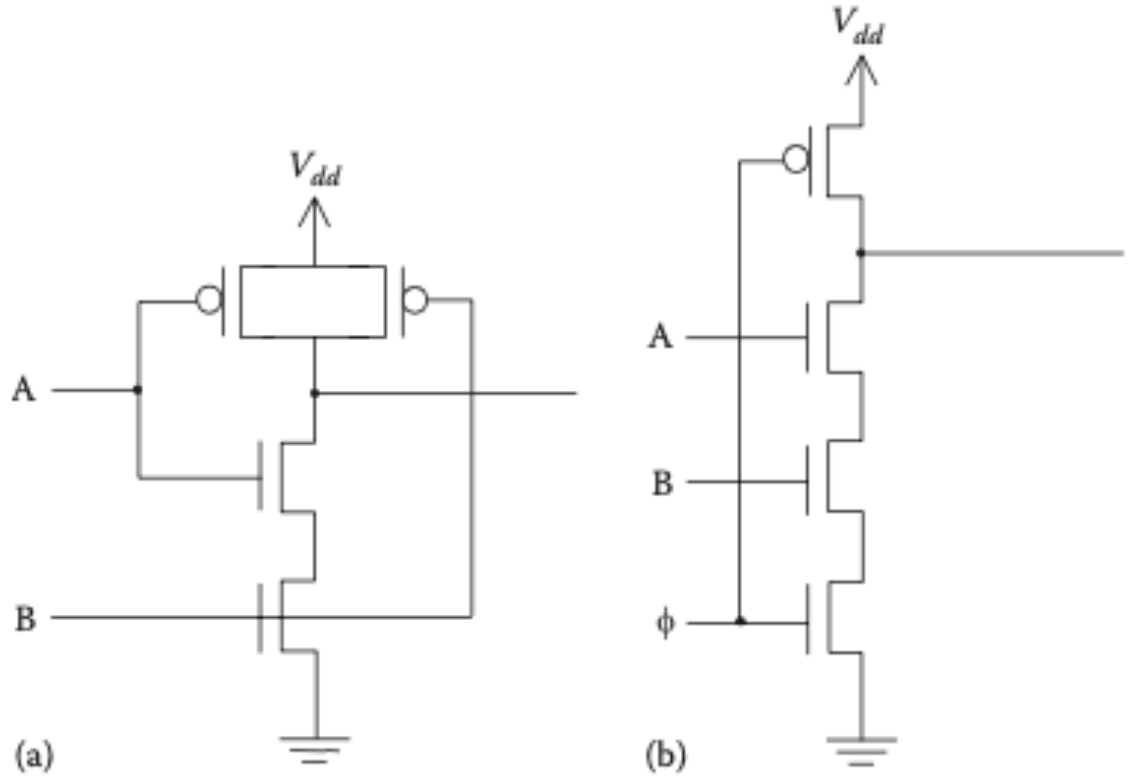


FIGURE 3.6 NAND gate: (a) static CMOS and (b) dynamic domino.

3.3 低功耗逻辑综合

低功耗逻辑综合领域已开展了大量 CAD 研究。 通过将功耗添加为综合工具的参数，可以节省电力，而不会造成延迟损失或延迟损失最小。

3.3.1 逻辑因式分解

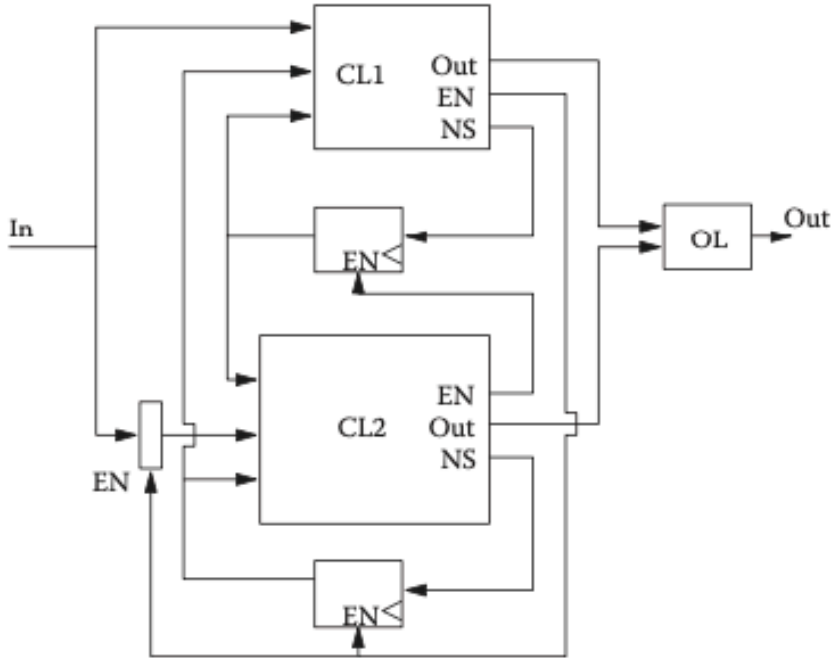


FIGURE 3.8 Implementation diagram of a decomposed FSM for low power.

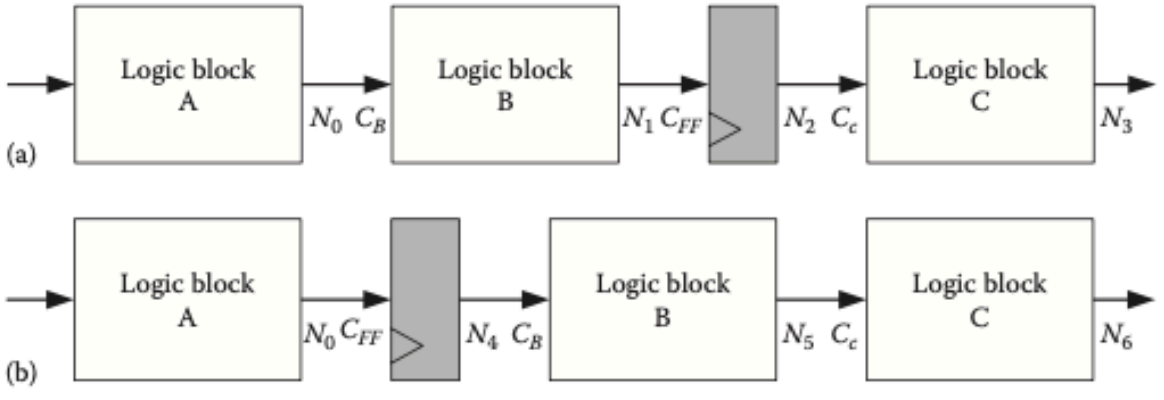


FIGURE 3.9 Two retimed versions, (a) and (b), of a network to illustrate the impact of this operation on the switched capacitance of a circuit.

Other techniques based on blocking input signal propagation and clock gating, such as pre-computation, are covered in some detail in Chapter 13 of *Electronic Design Automation for IC System Design, Verification, and Testing.*

独立于技术的优化的主要手段是逻辑表达式的因式分解。 例如，表达式 xy ∨ xz ∨ wy ∨ wz 可以分解为 (x ∨ w)(y ∨ z)，从而大大减少晶体管数量。 可以在多个函数中找到公共子表达式并重用。 对于区域优化，生成给定表达式的几个候选除数（例如，内核），并选择那些最大程度地减少文字计数的除数。 尽管最大限度地减少晶体管数量通常可以降低功耗，但在某些情况下，总有效开关电容实际上会增加。 当以功耗为目标时，成本函数必须考虑开关活动。 为低功耗内核提取提出的算法计算与每个内核的选择相关的切换活动。 内核选择基于面积和切换活动的减少[28]。

3.3.2 不关心优化

考虑适当的无关集来优化多级电路。 结构

逻辑电路的特性可能意味着给定逻辑门的某些输入组合永远不会发生。

这些组合形成了门的可控性或可满足性无关集。 相似地，

可能存在一些输入组合，其中门的输出值不用于

计算电路的任何输出。 这些组合的集合称为

可观察性不关心集。 尽管最初使用无关集来最小化面积，

已经提出了使用无关集来减少切换激活的技术

逻辑门输出端的有效性[29]。 给出静态 CMOS 门的转移概率

由 a = 2 p0 p1 = 2 p1 (1 - p1 ）（忽略时间相关性）。 该函数的最大值 x x1x x x

发生在 px = 0.5 时。 因此，为了最小化切换活动，策略是在函数的起始处（如果 p1x > 0.5）或在偏移（如果 p 1x < 0）中包含来自无关集的最小项。 5.

3.3.3 路径平衡

在典型的组合逻辑电路中，杂散转换占开关活动功率的很大一部分[30]。 为了减少杂散开关活动，在电路中每个门处汇聚的路径的延迟应该大致相等，这个问题称为路径平衡。 在上一节中，我们讨论了可以调整晶体管尺寸以最大限度地降低功耗，主要代价是延迟不在关键路径上的信号。 这种方法还具有有助于路径平衡的附加功能。 或者，可以通过重构逻辑电路来实现路径平衡，如图 3.7 所示。

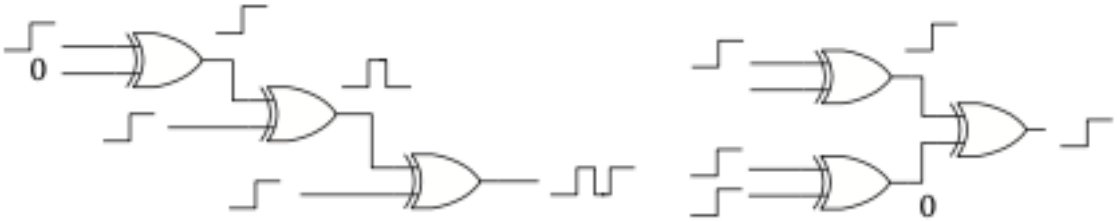


FIGURE 3.7 Path balancing through logic restructuring to reduce spurious transitions.

路径平衡对传播延迟极其敏感，当考虑工艺变化时，路径平衡成为一个更加困难的问题。 [30] 中的工作通过延迟和虚假活动估计的统计方法解决了路径平衡问题。

3.3.4 技术映射

技术映射是根据特定技术库中可用的逻辑元件来实现逻辑电路的过程。 与每个逻辑元件相关的是有关其面积、延迟以及内部和外部电容的信息。 优化问题是找到满足延迟约束的实现，同时最小化作为面积和功耗函数的成本函数[31,32]。 为了最大限度地减少功耗，具有高开关活动的节点被映射到复杂逻辑元件的内部节点，因为门内部的电容通常要小得多。

在许多情况下，逻辑门的输入在布尔意义上是可交换的。 然而，在特定的门实现中，等效引脚可能呈现不同的输入电容负载。 在这些情况下，应执行栅极输入分配，以便具有高开关活动的信号映射到具有较低输入电容的输入。

此外，大多数技术库都包含不同尺寸（即驱动能力）的相同逻辑元件。 因此，在低功耗的技术映射中，选择每个逻辑元件的大小，使得以最小功耗满足延迟约束。 该问题是上一节中描述的晶体管尺寸问题的离散对应问题。

3.3.5 状态编码

时序电路的综合为功率优化提供了新的途径。 状态编码是将唯一的二进制代码分配给有限状态机 (FSM) 中的每个状态的过程。 虽然这种分配不影响FSM的功能，但它决定了FSM实现中组合逻辑块的复杂性。 低功耗状态编码使用启发式方法，将最小汉明距离代码分配给由具有较大遍历概率的边连接的状态[33]。 状态转移图（STG）中给定边被遍历的概率由 STG 处于边的起始状态的稳态概率乘以与该边相关的输入组合的静态概率得出。 边缘。 每当使用该边沿时，只有少量状态信号（最好是一个）会发生变化，从而导致组合逻辑块中的总体开关活动减少。

3.3.6 FSM分解

FSM 分解已被提议用于 FSM 的低功耗实现。 基本思想是将原始 FSM 的 STG 分解为两个耦合的 STG，这两个 STG 一起具有与原始 FSM 相同的功能。 除了涉及从一个子 FSM 中的一种状态到另一子 FSM 中的状态的转换之外，仅需要对其中一个子 FSM 进行计时。 状态选择的策略是仅为其中一个子FSM选择少量状态。 这种选择包括搜索一小群状态，使得集群中状态之间转换的概率总和很高，并且与集群外状态之间的转换概率非常低。 目的是拥有一个大部分时间处于活动状态的小型子 FSM，从而禁用较大的子 FSM。 当两个子 FSM 都处于活动状态时，具有少量到/自另一个子 FSM 的转换对应于最坏情况。 每个子 FSM 都有一个额外的输出，用于禁用另一个子 FSM 的状态寄存器，如图 3.8 所示。 该额外输出还用于停止大型子 FSM 输入处的转换。 [34]中提出了一种仅使用电路技术来执行这种分解的方法，因此不需要任何 STG 的推导。

3.3.7 重新定时

重定时最初被提出作为一种通过移动电路中的寄存器同时保持输入输出功能来提高吞吐量的技术。 使用重定时来最小化开关活动是基于对寄存器输出的转换明显少于其输入的观察。 特别是，不存在任何故障。 通过重定时跨节点移动寄存器可能会改变电路中多个节点的切换活动。 在图 3.9a 所示的电路中，开关电容由 N0CB + N1CFF + N2CC 给出，而其重定时版本中的开关电容（如图 3.9b 所示）为 N0CFF + N4CB + N5CC。 这两个电路之一的开关电容可能要小得多。 已经提出了放置寄存器的启发式方法，使得驱动大电容的节点在给定吞吐量约束的情况下减少开关活动[35]。

3.4 总结

本章介绍了在较低设计抽象层次上降低数字电路功耗的方法。 电源电压的降低对功率影响较大； 然而，它也会降低性能。 我们描述的一些技术应用局部电压降低和动态电压控制来最大限度地减少性能损失的影响。