第1章 概 述

本章简要介绍 EDA 技术、EDA 工具、FPGA 结构原理及 EDA 的应用情况和发展趋势，其中重点介绍基于 EDA 的 FPGA 开发技术的概况。

考虑到本章中出现的一些基本概念和名词会涉及较多的基础知识和更深入的 EDA 基础理论，故对于本章的学习仅要求读者做一般性的了解，无须深入探讨。因为待读者学习完本教程，并经历了本教材配置的必要实践后，对许多问题就会自然而然地弄明白。不过需要强调的是，本章的重要性并不能因此而被低估。

1.1 EDA 技术

现代电子设计技术的核心已日趋转向基于计算机的电子设计自动化技术，即 EDA（electronic design automation）技术。当今的 EDA 技术已经渗透电子系统设计的各个细分领域，包括集成电路设计与制造、印制电路板设计与制造、可编程逻辑器件应用等。针对模拟电路系统、数字电路系统、射频电路系统、微波电路与天线系统等不同类型的电子系统的设计，产生了多种多样的 EDA 软件工具及其技术。本教材主要关注的是数字电路系统设计中的 EDA 技术，就是依赖功能强大的计算机，在 EDA 工具软件平台上，对以硬件描述语言（hardware description language，HDL）为系统逻辑描述手段完成的设计文件，自动地完成逻辑编译、化简、分割、综合、布局布线以及逻辑优化、时序分析和仿真测试，直至实现既定的电子线路系统功能。EDA 技术的出现使设计者的主要工作仅限于利用软件的方式来完成对系统硬件功能的实现，这是电子设计技术的一个巨大进步。

EDA 技术在硬件实现方面融合了大规模集成电路（IC）制造技术、IC 版图设计、设计仿真验证与时序优化、IC 测试和封装、印制电路板（PCB）设计制造以及 FPGA/CPLD（field programmable gate array/complex programmable logic device，现场可编程门阵列/复杂可编程逻辑器件）编程下载和自动测试等技术；在计算机辅助工程方面融合了计算机辅助设计（CAD）、计算机辅助制造（CAM）、计算机辅助测试（CAT）、计算机辅助工程（CAE）技术以及多种计算机语言的设计概念；而在现代电子学方面则容纳了更多的内容，如电子线路设计理论、数字信号处理技术、数字系统建模和优化技术等。因此，EDA 技术为现代电子理论和设计的表达与实现提供了可能性。正因为 EDA 技术丰富的内容及其与电子技术各学科领域的相关性，其发展的历程同大规模集成电路设计技术、计算机辅助工程、可编程逻辑器件以及电子设计技术和工艺是同步的。

根据过去数十年的电子技术的发展历程，可大致将 EDA 技术的发展分为 3 个阶段。

第一阶段：20 世纪 70 年代，在集成电路制造方面，双极工艺、MOS 工艺已得到广泛的应用。可编程逻辑技术及其器件已经问世，计算机作为一种运算工具已在科研领域得到广泛应用。而在后期，CAD 的概念已见雏形，这一阶段人们开始利用计算机取代手工劳动，辅助进行集成电路版图编辑、PCB 布局布线等工作，这是 EDA 技术的雏形。

第二阶段：20 世纪 80 年代，集成电路设计进入了 CMOS（complementary metal oxide semiconductor，互补金属氧化物半导体）场效应管时代。复杂可编程逻辑器件已进入商业应用，FPGA 被发明（1985 年），相应的辅助设计软件也已投入使用；在 20 世纪 80 年代末，CAE 和 CAD 技术的应用更为广泛，它们在 PCB 设计原理图输入、自动布局布线及 PCB 分析、逻辑设计、逻辑仿真、布尔代数综合和化简等方面担任了重要的角色。特别是各种硬件描述语言的出现、应用和标准化方面的重大进步，为电子设计自动化解决电子线路建模、标准文档及仿真测试等问题奠定了基础。

第三阶段：20 世纪 90 年代，计算机辅助工程、辅助分析和辅助设计在电子技术领域获得更加广泛的应用。与此同时，电子技术在通信、计算机及家电产品生产中的市场需求和技术需求，极大地推动了全新的电子设计自动化技术的应用和发展。特别是集成电路设计工艺步入了超深亚微米阶段，百万门级以上的大规模可编程逻辑器件的陆续问世，以及基于计算机技术的面向用户的低成本、大规模 ASIC 设计技术的应用，促进了 EDA 技术的形成。更为重要的是，各 EDA 公司致力于推出兼容各种硬件实现方案和支持标准硬件描述语言的 EDA 工具软件，都有效地将 EDA 技术推向成熟和实用。

EDA 技术在进入 21 世纪后得到了更大的发展，突出表现在以下几个方面。

- 在 FPGA 上实现 DSP（digital signal processing，数字信号处理）应用成为可能，用纯数字逻辑进行 DSP 模块的设计，使得高速 DSP 的实现成为现实，并有力地推动了软件无线电技术的实用化和发展。基于 FPGA 的 DSP 技术，为高速数字信号处理算法提供了实现途径。
- 集成电路向 3D IC 方向发展，FinFET、GAAFET 使亿门级电路集成在单芯片变得容易，SoC（system on a chip，系统级芯片）需要软硬件协同设计。
- 嵌入式处理器软核的成熟，使 SOPC（system on a programmable chip，可编程片上系统）技术成为可能，即可以在单片 FPGA 中实现一个完备的可随意重构的嵌入式系统。
- 在仿真和设计两方面支持标准硬件描述语言的功能强大的 EDA 软件不断推出。
- EDA 使电子领域各学科的界限更加模糊，也更加互为包容，如模拟与数字、软件与硬件、系统与器件、ASIC 与 FPGA 等。
- 基于 EDA 的用于 ASIC 设计的标准单元已涵盖大规模电子系统及复杂 IP（intellectual property，知识产权）核模块。
- 软硬 IP 核在电子行业的产业领域广泛应用。
- SoC 高效低成本设计技术走向成熟。
- 系统级、行为验证级硬件描述语言的出现（如 System C、SystemVerilog）使复杂电子系统的设计和验证趋于简单。
- C 语言综合技术开始应用于复杂 EDA 软件工具中。使用 C 或类 C 语言对数字逻辑系统进行设计已经成为可能。HLS（high-level synthesis，高级综合）工具可以实现简单 C 程序到 HDL 的转化，而 OpenCL 工具可以促进以 CPU 为核心的 C 算法加速的应用。
- 以深度学习为代表的人工智能技术在 21 世纪 10 年代获得飞速发展，也得益于芯片集成晶体管规模越来越大，基于 FPGA 或 ASIC 良好的并行计算性能，CNN（convolutional neural network，卷积神经网络）等神经网络结构被设计到 FPGA 与 ASIC 上。

1.2　EDA 技术应用对象

一般地，利用 EDA 技术进行电子系统设计的最后目标是完成专用集成电路（ASIC）或印制电路板（PCB）的设计和实现，如图 1-1 所示。其中，PCB 设计指的是电子系统的印制电路板设计，从电路原理图到 PCB 上元件的布局布线、阻抗匹配、信号完整性分析及板级仿真，到最后的电路板机械加工文件生成，这些都需要相应的计算机 EDA 工具软件辅助设计者来完成，这仅是 EDA 技术应用的一个重要方面，但本书限于篇幅不做展开。ASIC 作为最终的物理平台，在此硬件实体上用户可通过 EDA 技术将电子应用系统的既定功能和技术指标进行具体实现。

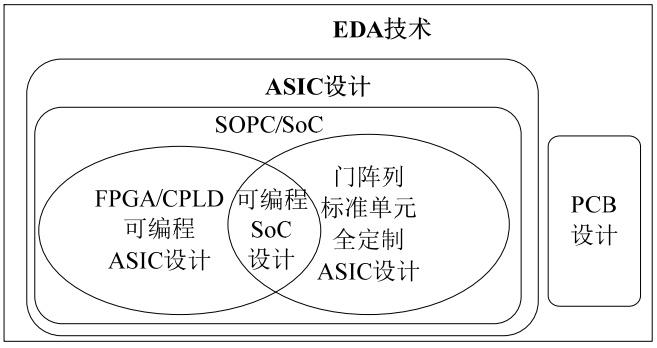


图 1-1　EDA 技术实现目标

专用集成电路就是具有专门用途和特定功能的独立集成电路器件，根据这个定义，作为 EDA 技术最终实现目标的 ASIC，可以通过 3 种途径来完成。

1．可编程逻辑器件

FPGA 和 CPLD 是实现这一途径的主流器件，它们的特点是直接面向用户、具有极大的灵活性和通用性、使用方便、硬件测试和实现快捷、开发效率高、成本低、上市时间短、技术维护简单、工作可靠性好等。FPGA 和 CPLD 的应用是 EDA 技术有机融合软硬件电子设计技术、SoC 和 ASIC 设计，以及对自动设计与自动实现最典型的诠释。由于 FPGA 和 CPLD 的开发工具、开发流程和使用方法与 ASIC 有类似之处，因此这类器件通常也被称为可编程专用 IC 或可编程 ASIC。

2．半定制或全定制 ASIC

基于 EDA 技术的半定制或全定制 ASIC，根据它们的实现工艺，可统称为掩模（mask）ASIC，或直接称 ASIC。可编程 ASIC 与掩模 ASIC 相比，不同之处在于前者具有面向用户灵活多样的可编程性，即硬件结构的可重构特性。掩模 ASIC 大致分为门阵列 ASIC、标准单元 ASIC 和全定制 ASIC。对于全定制芯片，在针对特定工艺建立的设计规则下，设计者对于电路的设计有完全的控制权，如线的间隔和晶体管大小的确定。该领域的一个例外是混合信号设计，使用通信电路的 ASIC 可以定制设计其模拟部分。

目前大部分 ASIC 是采用标准单元法（即使用库中不同大小的标准单元）设计的。在设计者一级，库（library）包括不同复杂性的逻辑元件：SSI 逻辑块、MSI 逻辑块、数据通

道模块、存储器、IP 乃至系统级模块等。库包含每个逻辑单元在硅片级的完整布局，使用者只需利用 EDA 软件工具与逻辑块描述打交道即可，完全不必关心深层次电路布局的细节。标准单元布局中，所有扩散、接触点、过孔、多晶通道及金属通道都已完全确定。当该单元用于设计时，通过 EDA 软件产生的网表文件将单元布局块“粘贴”到芯片布局之上的单元行上。标准单元 ASIC 设计与 FPGA 设计的开发流程相近。

3．可编程 SoC

可编程 SoC 主要指既含有面向用户的 FPGA 可编程功能和逻辑资源，同时也含有可方便调用和配置的硬件标准单元模块，如 CPU、RAM、ROM、硬件加法器、乘法器、锁相环等。不同厂家对可编程 SoC 的称谓并不统一，有各自的产品名称，如 SoC FPGA、MPSoC、RFSoC、自适应 SoC、FPGA SoC、PSoC 等，但有共同的特征，即均有可编程逻辑资源与处理器单元。

1.3　常用的硬件描述语言

硬件描述语言（HDL）是 EDA 技术的重要组成部分，目前常用的 HDL 主要有 Verilog HDL、VHDL、SystemVerilog 和 System C。其中 Verilog HDL 和 VHDL 在现在的 EDA 设计中使用最多，几乎得到所有主流 EDA 工具的支持。而 SystemVerilog 和 System C 这两种 HDL 还处于不断完善过程中，主要加强了系统验证方面的功能。

以下分别对 Verilog HDL、VHDL、SystemVerilog 和 System C 做简要介绍，同时说明了一些有关 Chisel 的知识。

1．Verilog HDL

Verilog HDL 是电子设计主流硬件的描述语言之一，本书将重点介绍它的编程方法和使用技术。Verilog HDL（以下简称 Verilog）最初由 Gateway Design Automation（以下简称 GDA）公司的 Phil Moorby 在 1983 年创建。起初，Verilog 仅作为 GDA 公司的 Verilog-XL 仿真器的内部语言，用于数字逻辑的建模、仿真和验证。Verilog-XL 推出后获得了成功和认可，从而促进了 Verilog HDL 的发展。1989 年，GDA 公司被 Cadence 公司收购，Verilog 语言成了 Cadence 公司的私有财产。1990 年，Cadence 公司成立了 OVI（Open Verilog International）组织，公开了 Verilog 语言，并由 OVI 负责促进 Verilog 语言的发展。在 OVI 的努力下，IEEE（Institute of Electrical and Electronics Engineers）于 1995 年制定了 Verilog HDL 的第一个国际标准—— IEEE Std 1364-1995，即 Verilog 1.0。

2001 年，IEEE 发布了 Verilog HDL 的第二个标准版本（Verilog 2.0），即 IEEE Std 1364-2001，简称 Verilog-2001 标准。由于 Cadence 公司在集成电路设计领域的影响力和 Verilog 的易用性，Verilog 成为基层电路建模与设计中最流行的硬件描述语言。

2005 年，IEEE 再次对 Verilog HDL 标准进行少量修改，发布了 Verilog-2005 标准，基本与 Verilog-2001 标准是一致的。

Verilog 的部分语法是参照 C 语言的语法设立的（但与 C 语言有本质区别），因此具有很多 C 语言的优点，从形式表述上来看，其代码简明扼要，使用灵活，且语法规定不是很严谨，很容易上手。Verilog 具有很强的电路描述和建模能力，能从多个层次对数字系统进

行建模和描述，从而大大简化硬件设计任务，提高设计效率和可靠性。Verilog 在语言易读性、层次化和结构化设计方面表现出了强大的生命力和应用潜力。因此，其支持各种模式的设计方法：自顶向下与自底向上或混合方法。在面对当今许多电子产品生命周期缩短，需要多次重新设计以融入最新技术、改变工艺等方面，Verilog 具有良好的适应性。

用 Verilog 进行电子系统设计的一个很大的优点是当设计逻辑功能时，设计者可以专心致力于其功能的实现，而不需要对不影响功能的、与工艺有关的因素花费过多的时间和精力；当需要仿真验证时，可以很方便地从电路物理级、晶体管级、寄存器传输级乃至行为级等多个层次来做描述的验证。

2．VHDL

VHDL 的英文全名是 VHSIC（very high speed integrated circuit）hardware description language，于 1983 年由美国国防部（DOD）发起创建，由 IEEE 进一步发展并在 1987 年作为“IEEE 标准 1076”（IEEE Std 1076）发布。从此，VHDL 成为硬件描述语言的业界标准之一。自 IEEE 公布了 VHDL 的标准版本之后，各 EDA 公司相继推出了自己的 VHDL 设计环境，或宣布自己的设计工具支持 VHDL。此后 VHDL 在电子设计领域得到了广泛应用，并与 Verilog 一起逐步取代了其他的非标准硬件描述语言。

VHDL 作为一种规范语言和建模语言，随着它的标准化，出现了一些支持该语言的行为仿真器。创建 VHDL 的最初目标是用于标准文档的建立和电路功能模拟，其基本想法是在高层次上描述系统和元件的行为。但到了 20 世纪 90 年代初，人们发现，VHDL 不仅可以作为系统模拟的建模工具，而且可以作为电路系统的设计工具，可以利用软件工具将 VHDL 源码自动转化为文本方式表达的基本逻辑元件连接图，即网表文件。这种方法对于电路自动设计来说显然是一个极大的推进。很快，电子设计领域出现了第一个软件设计工具，即 VHDL 逻辑综合器，它把标准 VHDL 的部分语句描述转化为具体电路实现的网表文件。

1993 年，IEEE 对 VHDL 进行了修订，从更高的抽象层次和系统描述能力上扩展了 VHDL 的内容，公布了新版本 VHDL，即 IEEE 1076-1993。现在，VHDL 与 Verilog 一样作为 IEEE 的工业标准硬件描述语言，得到了众多 EDA 公司的支持，在电子工程领域已成为事实上的通用硬件描述语言。2008 年，VHDL 标准再次进行修订，即 IEEE 1076-2008，做了非常多的语法简化，大大提高了设计者的工作效率。

VHDL 具有与具体硬件电路无关和与设计平台无关的特性，并且具有良好的电路行为描述和系统描述的能力。按照设计目的，VHDL 程序可以划分为面向仿真和面向综合两类，而面向综合的 VHDL 程序分别面向 FPGA 和 ASIC 开发两个领域。

3．SystemVerilog

SystemVerilog 是一种较新的硬件描述语言，是由 Accellera（Accellera 的前身就是 OVI）开发的。SystemVerilog 在 Verilog-2001 的基础上做了扩展，将 Verilog 语言推向了系统级空间和验证级空间，极大地改进了高密度、基于 IP 的、总线敏感的芯片设计效率。SystemVerilog 主要定位于集成电路的实现和验证流程，并为系统级设计流程提供了强大的链接能力。SystemVerilog 改进了 Verilog 代码的生产率、可读性以及可重用性。SystemVerilog 提供了更简约的硬件描述，还为随机约束的测试平台开发、覆盖驱动的验证以及基于断言的验证提供了广泛的支持。

2005 年，IEEE 批准了 SystemVerilog 的语法标准，即 IEEE Std1800 标准。

4．System C

System C 是 C++语言的硬件描述扩展，主要用于 ESL（电子系统级）建模与验证。由 OSCI（Open System C Initiative）组织进行发展。System C 并非好的 RTL 语言（即可综合的、硬件可实现描述性质的语言），而是一种系统级建模语言。将 System C 和 SystemVerilog 组合起来，能够提供一套从 ESL 至 RTL 验证的完整解决方案。

System C 源代码可以使用任何标准 C++编译环境进行编译，生成可执行文件；运行可执行文件，可生成 VCD 格式的波形文件。System C 的综合还不完善，但已经有工具支持。

5．Chisel

Chisel 是一种基于 Scala 语言的敏捷开发型硬件描述语言。Scala 是一种基于 JVM（Java Virtual Machine）的面向对象的函数式语言，常见于网络编程应用，是一种比较小众的语言。Chisel（constructing hardware in a Scala embedded language）由加州大学伯克利分校的研究团队发布，是 Scala 在电路描述上的一个扩展，编译后生成 Verilog/VHDL 代码，再交由 EDA 软件生成电路网表。该语言还在发展阶段，有待成熟。Chisel 继承了 Scala 的各种高级语法特性，大大简化了硬件电路描述的代码，有助于提高开发效率。

1.4　EDA 技术的优势

在传统的数字电子系统或 IC 设计中，手工设计占了很大的比例。设计流程中，一般先按电子系统的具体功能要求进行功能划分，然后对每个子模块画出真值表，用卡诺图进行手工逻辑简化，并写出布尔表达式，画出相应的逻辑线路图，再据此选择元器件，设计电路板，最后进行实测与调试。传统数字技术的手工设计方法的缺点如下。

（1）电路设计复杂、调试十分困难。

（2）由于无法进行硬件系统仿真，如果某一过程存在错误，查找和修改十分不便。

（3）设计过程中产生大量文档，不易管理。

（4）对于 IC 设计而言，设计实现过程与具体生产工艺直接相关，因此可移植性差。

（5）只有在设计出样机或生产出芯片后才能进行实测。

（6）所能设计完成的系统规模通常很小，抗干扰能力差，工作速度也很低。

相比之下，EDA 技术有很大的不同，其具体优势描述如下。

（1）用 HDL 对数字系统进行抽象的行为与功能描述以及具体的内部线路结构描述，从而可以在电子设计的各个阶段、各个层次进行计算机模拟验证，这不仅可以保证设计过程的正确性，而且可以大大降低设计成本，缩短设计周期。

（2）EDA 工具之所以能够完成各种自动设计过程，关键是具有各类库的支持，如逻辑仿真时的模拟库、逻辑综合时的综合库、版图综合时的版图库、测试综合时的测试库等。这些库都是 EDA 公司与半导体生产厂商紧密合作、共同开发的。

（3）一些 HDL 本身也是文档型的语言（如 VHDL），极大地简化了设计文档的管理。

（4）EDA 技术中最为瞩目的功能，即最具现代电子设计技术特征的功能是日益强大的逻辑设计仿真测试技术。EDA 仿真测试技术只需通过计算机，就能针对所设计的电子系统各个层次性能特点，完成一系列准确的测试与仿真操作。在完成实际系统的安装后，还能对系统上的目标器件进行所谓“边界扫描测试”及嵌入式逻辑分析仪的应用。这一切都

极大地提高了大规模系统电子设计的自动化程度。

（5）无论传统的应用电子系统设计得如何完美、使用了多么先进的功能器件，都掩盖不了一个无情的事实，即设计者对该系统没有任何自主知识产权，因为系统中关键性的器件往往并非出自设计者之手，这将导致该系统在许多情况下的应用直接受到限制。基于 EDA 技术的设计规则不同，又由于用 HDL 表达的成功的专用功能设计在实现目标方面有很大的可选性，它既可以用不同来源的通用 FPGA 实现，也可以直接以 ASIC 来实现，设计者拥有完全的自主权，再无受制于人之虞。

（6）传统的电子设计方法至今没有任何标准规范加以约束，因此设计效率低、系统性能差、规模小、开发成本高、市场竞争能力弱。相比之下，EDA 技术的设计语言是标准化的，不会由于设计对象的不同而改变；它的开发工具是规范化的，EDA 软件平台支持任何标准化的设计语言；它的设计成果是通用性的，IP 核具有规范的接口协议；它具有良好的可移植与可测试性，为系统开发提供了可靠的保证。

（7）从电子设计方法学来看，EDA 技术最大的优势就是能将所有设计环节纳入统一的自顶向下的设计方案中。

（8）EDA 不但在整个设计流程上充分利用计算机的自动设计能力、在各个设计层次上利用计算机完成不同内容的仿真模拟，而且在系统板设计结束后仍可利用计算机对硬件系统进行完整的测试。

1.5　面向 FPGA 的开发流程

完整地了解利用 EDA 技术进行设计开发的流程，对于正确地选择和使用 EDA 软件、优化设计项目、提高设计效率十分有益。一个完整的、典型的 EDA 设计流程既是自顶向下设计方法的具体实施途径，也是 EDA 工具软件本身的组成结构。

1.5.1　设计输入

图 1-2 是基于 EDA 软件的 FPGA 开发流程框图。以下将分别介绍各设计模块的功能特点。对于目前流行的用于 FPGA 开发的 EDA 软件，图 1-2 的设计流程具有一般性。

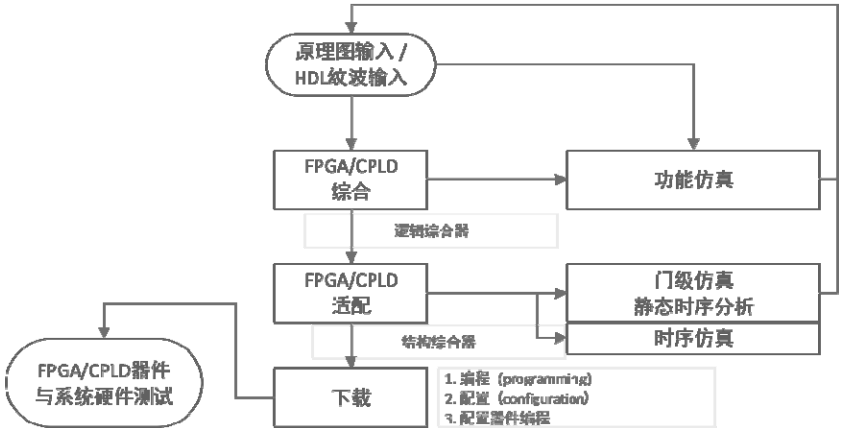


图 1-2　基于 EDA 的 FPGA 开发流程

将电路系统以一定的表达方式输入计算机，是在 EDA 软件平台上对 FPGA/CPLD 开发的最初步骤。通常，使用 EDA 工具的设计输入可分为两种类型。

1．图形输入

图形输入通常包括状态图输入和电路原理图输入。

状态图输入方法就是根据电路的控制条件和不同的转换方式，使用绘图的方法在 EDA 软件的状态图编辑器上绘出状态图，然后由 EDA 编译器和综合器将此状态变化流程图编译综合成电路网表。

电路原理图输入方法是一种类似于传统电子设计方法的原理图编辑输入方式，即在 EDA 软件的图形编辑界面上绘制能完成特定功能的电路原理图。原理图由逻辑器件（符号）和连接线构成，原理图中的逻辑器件可以是 EDA 软件库中预制的功能模块，如与门、非门、或门、触发器以及各种含 74 系列器件功能的宏功能块，甚至还有一些类似于 IP 的宏功能块。

2．硬件描述语言代码文本输入

这种方式与传统的计算机软件语言编辑输入基本一致，就是将使用了某种硬件描述语言的电路设计代码，如 Verilog 或 VHDL 的源程序，进行编辑输入。

1.5.2　综合

综合（synthesis）的字面含义应该是把抽象的实体结合成单个或统一的实体。因此，综合就是把某些东西结合到一起，把设计抽象层次中的一种表述转化成另一种表述的过程。在电子设计领域，综合的概念可以表述如下：将用行为和功能层次表达的电子系统转换为低层次的、便于具体实现的模块组合装配的过程。事实上，自上而下的设计过程中的每一步都可称为一个综合环节。现代电子设计过程通常从高层次的行为描述开始，以底层的结构甚至更低层次描述结束，每个综合步骤都是上一层次的转换。

（1）从自然语言转换到 VHDL 语言算法表述，即自然语言综合。

（2）从算法表述转换到寄存器传输级（register transport level，RTL）表述，即从行为域到结构域的综合，也称行为综合。

（3）从 RTL 级表述转换到逻辑门（包括触发器）表述，即逻辑综合。

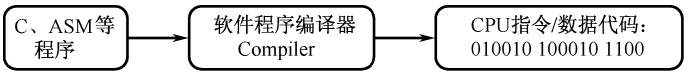
（4）从逻辑门表述转换到版图级表述（如 ASIC 设计），或转换到 FPGA 的配置网表文件，可称为版图综合或结构综合。有了版图信息就可以把芯片生产出来了。有了对应的配置文件，就可以使对应的 FPGA 变成具有专门功能的电路器件了。

显然，综合器就是能够将一种设计表述形式自动向另一种设计表述形式转换的计算机程序，或协助进行手工转换的程序。它可以将高层次的表述转化为低层次的表述，可以将行为域转化为结构域，可以将高一级抽象的电路描述（如算法级）转化为低一级的电路描述（如门级），并且可以用某种特定的“技术”（如 CMOS）实现。

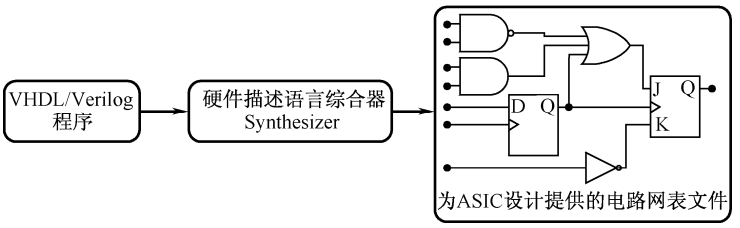
从表面上看，VHDL 等硬件描述语言综合器和软件程序编译器都是一种“翻译器”，都能将高层次的设计表达转化为低层次的设计表达，但它们却具有许多本质的区别。

如图 1-3 所示，编译器将软件程序翻译成基于某种特定 CPU 的机器代码，这种代码仅限于这种 CPU，不能移植。它并不代表硬件结构，更不能改变 CPU 的结构，只能被动地为其特定的硬件电路所利用。如果脱离了已有的硬件环境（CPU），机器代码将失去意义。此外，编译器作为一种软件的运行，除了某种单一目标器件，即 CPU 的硬件结构，不需要任何与硬件相关的器件库和工艺库参与编译。因而，编译器的工作单纯得

多，编译过程基本属于一种一一对应式的、机械转换式的“翻译”行为。



（a）软件语言设计目标流程



（b）硬件语言设计目标流程

图 1-3　编译器和综合的功能比较

综合器则不同，同样是类似的软件代码（如 VHDL 程序代码），综合器转化的目标是底层的电路结构网表文件，这种满足原设计程序功能描述的电路结构不依赖于任何特定硬件环境，因此可以独立地存在，并能轻易地被移植到任何通用硬件环境中，如 ASIC、FPGA 等。换言之，电路网表代表了特定的且可独立存在和具有实际功能的硬件结构，因此具备了随时改变硬件结构的依据，综合的结果具有相对独立性。

另一方面，综合器在将使用硬件描述语言表达的电路功能转化成具体的电路结构网表过程中，具有明显的能动性（如状态机的优化），它不是机械的一一对应式的“翻译”，而是根据设计库、工艺库以及预先设置的各类约束条件，“自主”地选择最优的方式完成电路设计。即对于相同的 VHDL 表述，综合器可以用不同的电路结构实现相同的功能。

如图 1-4 所示，与编译器相比，综合器具有更复杂的工作环境。综合器在接收 VHDL 程序并准备对其综合前，必须获得与最终实现设计电路硬件特征相关的工艺库的信息，以及获得优化综合的诸多约束条件。一般地，约束条件有多种，如设计规则、时间约束（包括速度约束）、面积约束等。通常，时间约束的优先级高于面积约束。设计优化要求当综合器把 VHDL 源码翻译成通用原理图时，将识别状态机、加法器、乘法器、多路选择器和寄存器等。这些运算功能根据 VHDL 源码中的符号（如加、减、乘、除），都可用多种方法实现。例如，加法可实现的方案有多种，有的面积小，速度慢；有的速度快，面积大。VHDL 行为描述强调的是电路的行为和功能，而不是电路如何实现。而选择电路的实现方案正是综合器的任务，即综合器选择一种能充分满足各项约束条件且成本最低的实现方案。

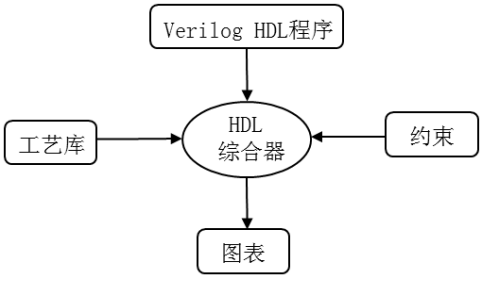


图 1-4　HDL 综合器运行流程

注意，VHDL（也包括 Verilog、SystemVerilog）方面的 IEEE 标准主要指的是文档的表述、行为建模及其仿真，至于在电子线路的设计方面，VHDL 并没有得到全面的标准化支持。也就是说，HDL 综合器并不能支持 IEEE 标准的 VHDL 的全集（全部语句程序），而只能支持其子集，即部分语句，并且不同的 HDL 综合器所支持的 VHDL 子集也不完全相

同。这样一来，对于相同的 VHDL 源代码，不同的 HDL 综合器可能综合出在结构和功能上并不完全相同的电路系统。对此，设计者应给予充分注意：对于不同的综合结果，不应对综合器的特性贸然做出评价，而应在设计过程中尽可能全面了解所使用的综合工具的特性。当然，随着 EDA 技术的不断进步，可综合的 VHDL 正逐渐走向标准化。

1.5.3　适配（布局布线）

适配器（fitter）也称结构综合器，它的功能是将由综合器产生的网表文件配置于指定的目标器件中，使之产生最终的下载文件，如 JEDEC、JAM、POF、SOF 等格式的文件。适配所选定的目标器件必须属于原综合器指定的目标器件系列。通常，EDA 软件中的综合器可由专业的第三方 EDA 公司提供，而适配器则需由 FPGA/CPLD 供应商提供。因为适配器的适配对象直接与器件的结构细节相对应。

适配器就是将综合后的网表文件针对某一具体的目标器件进行逻辑映射操作，其中包括底层器件配置、逻辑分割、优化、布局布线等操作。适配完成后可以利用适配所产生的仿真文件做精确的时序仿真，同时产生可用于对目标器件进行编程的文件。

1.5.4　仿真与时序分析

在编程下载前必须利用 EDA 工具对适配生成的结果进行模拟测试，就是所谓的仿真（simulation）。仿真就是让计算机根据一定的算法和一定的仿真库对 EDA 设计进行模拟，验证设计的正确性，以便排除错误，它是 EDA 设计过程中的重要步骤。

图 1-2 所示的功能仿真与门级仿真通常由 FPGA 公司的 EDA 开发工具直接提供，也可以选用第三方的专业仿真工具（也可将第三方仿真器软件集成在 EDA 开发工具中，就如 Quartus 那样），它可以完成两种不同级别的仿真测试。

（1）功能仿真。即直接对 HDL、原理图描述或其他描述形式的逻辑功能进行测试模拟，以了解其实现的功能是否满足原设计要求的过程。仿真过程不涉及任何具体器件的硬件特性。不经历适配阶段，在设计项目编辑、编译（或综合）后即可进入门级仿真器进行模拟测试。直接进行功能仿真的好处是设计耗时短，对硬件库、综合器等没有任何要求。

（2）时序仿真。即接近真实器件时序性能运行特性的仿真。仿真文件已包含了器件硬件特性参数，因而仿真精度高，但时序仿真的仿真文件必须来自针对具体器件的适配器。综合后所得的 EDIF 等网表文件通常作为 FPGA 适配器的输入文件，产生的仿真网表文件中包含了精确的硬件延迟信息。时序仿真往往耗时较长。

（3）门级仿真。一般指综合适配后在门级网表上进行的仿真。仿真文件已包含了器件硬件特性参数，因而仿真更符合实际器件运行行为。在进行门级仿真时可以选择是否使用时延信息（时延文件），若使用即为时序仿真或称为时序门级仿真。

（4）静态时序分析。时序仿真和功能仿真都是基于有系统激励输入的仿真验证，属于动态时序分析范畴。若纯粹分析电路各个部分的延迟，那么就需要进行静态时序分析（static timing analysis，STA）。STA 是对设计进行时序估计、设计优化的重要手段，在现代 EDA 设计中占据重要地位。现在越来越多的设计验证采用静态时序分析加不带延迟文件的门级仿真来代替时序仿真，以提高验证覆盖率，缩短验证时间。

1.5.5　RTL 描述

RTL（register transport level，寄存器传输级）的概念会经常出现，这里对它做一些说明。

RTL 的概念最初产生于对某类电路的描述。RTL 描述是以规定设计中的各种寄存器形式为特征，然后在寄存器之间插入组合逻辑。这类寄存器或者显式地通过元件具体装配，或者通过推论进行隐含的描述。传统概念下的 RTL 电路的构建特色是由一系列组合电路模块和寄存器模块相间级联而成，即组合电路与时序电路各自独立且级联，而信号的通过具有逐级传输的特征，故称此类电路为寄存器传输级电路。

此后，这个概念进一步泛化，便引申为一切用各种独立的组合电路模块和独立的寄存器模块，但不涉及低层具体逻辑门结构或触发器电路细节（所谓技术级，英文全称为 technology level）来构建描述数字电路的形式都称为 RTL 描述，而且即使不包含时序模块，或只有寄存器模块的同类描述形式也都泛称为 RTL 描述。所以现在所谓的 RTL 仿真，即功能仿真，就是指不涉及电路细节（如门级细节）的 RTL 模块级构建的系统的仿真；而涉及电路细节和时序性能的时序仿真则称为门级仿真。

1.6　可编程逻辑器件

可编程逻辑器件（programmable logic devices，PLD）是 20 世纪 70 年代发展起来的一种新的集成器件。PLD 是大规模集成电路技术发展的产物，是一种半定制的集成电路，结合 EDA 技术可以快速、方便地构建数字系统。

数字电子技术基础知识表明，数字电路系统都是由基本门来构成的，如与门、或门、非门、传输门等。由基本门可构成两类数字电路：一类是组合电路；另一类是时序电路，它含有存储元件。事实上，不是所有的基本门都是必需的，如用与非门单一基本门就可以构成其他的基本门。任何的组合逻辑函数都可以转化为“与–或”表达式，即任何的组合电路可以用“与门–或门”二级电路实现。同样，任何时序电路都可由组合电路加上存储元件（即锁存器、触发器）来构成。由此，人们提出了一种可编程电路结构，即可重构的电路结构。

1.6.1　PLD 的分类

可编程逻辑器件的种类有很多，几乎每个大的可编程逻辑器件供应商都能提供具有自身结构特点的 PLD 器件。由于历史的原因，可编程逻辑器件的命名各异，在详细介绍可编程逻辑器件之前，有必要介绍几种 PLD 的分类方法。

1．按集成度分类

按集成度划分，一般可分为两大类器件。

（1）低集成度芯片。早期出现的 PROM（programmable read only memory，可编程只读存储器）、PAL（programmable array logic，可编程阵列逻辑）、可重复编程的 GAL（generic array logic，能用阵列逻辑）都属于这类。一般而言，可重构使用的逻辑门数在 500 门以下，称为简单 PLD（简称 SPLD）。

（2）高集成度芯片。如现在大量使用的 CPLD、FPGA 器件，称为复杂 PLD（简称 CPLD）。

2．按结构分类

从结构上可分为两大类器件。

（1）乘积项结构器件。其基本结构为“与–或”阵列的器件，大部分 SPLD 和 CPLD 都属于这个范畴。

（2）查找表结构器件。由简单的查找表组成可编程门，再构成阵列形式。大多数 FPGA 属于此类器件。

3．按编程工艺分类

从编程工艺上划分，可分为如下几种类型。

（1）熔丝（fuse）型。早期的 PROM 器件就是采用熔丝结构的，编程过程就是根据设计的熔丝图文件来烧断对应的熔丝，以达到编程的目的。

（2）反熔丝（anti-fuse）型。是对熔丝技术的改进，在编程过程中通过击穿漏层使两点之间获得导通，这与熔丝烧断获得开路正好相反。

（3）EPROM 型。称为紫外线擦除电可编程逻辑器件，是用较高的编程电压进行编程的，当需要再次编程时，用紫外线进行擦除。Atmel 公司曾经有过此类 PLD，目前已淘汰使用。

（4）EEPROM 型。即电可擦写编程器件，现有部分 CPLD 及 GAL 器件采用此类结构。它是对 EPROM 工艺的改进，不需要紫外线擦除，而是直接用电擦除。

（5）SRAM 型。即 SRAM 查找表结构的器件。大部分的 FPGA 器件都是采用此种编程工艺，如 AMD-Xilinx 和 Intel-Altera 的 FPGA 器件采用的就是 SRAM 编程方式。这种方式在编程速度、编程要求上要优于前 4 种器件，不过 SRAM 型器件的编程信息存放在 RAM 中，在断电后就丢失了，再次上电需要再次编程（配置），因而需要专用器件来完成这类自动配置操作。

（6）Flash 型。Actel 公司为了解决上述反熔丝器件的不足之处，推出了采用 Flash 工艺的 FPGA，可以实现多次可编程，同时做到掉电后不需要重新配置。现在 AMD-Xilinx 和 Intel-Altera 的多个系列 CPLD 也采用 Flash 型。

在习惯上，还有另外一种分类方法，即掉电后是否需要重新配置器件。CPLD 不需要重新配置，而 FPGA（大多数）需要重新配置。

1.6.2　PROM 可编程原理

介绍 PLD 器件的可编程原理之前，在此首先介绍结构上具有典型性的 PROM 结构。但在此之前需熟悉一些常用逻辑电路符号及描述 PLD 内部结构的专用电路符号。

目前流行于国内高校数字电路教材中的“国标”逻辑符号原本是全盘照搬 ANSI/IEEE Std 91a-1984 版的 IEC 国际标准符号，且至今并没有升级。然而由于此类符号表达形式过于复杂，即用矩形图型逻辑符号来标志逻辑功能，故被公认为不适合表述 PLD 中复杂的逻辑结构。因此数年后，IEEE 又推出了 ANSI/IEEE Std 91a-1991 标准，于是国际上几乎所有技术资料和相关教材很快就废弃了原标准（1984 版本）的应用，继而普遍采用了 1991 版本的国际标准逻辑符号。该版本符号的优势和特点是用图形的不同形状来标志逻辑模块的

功能，即使图形画面很小，也能十分容易地辨认出模块的逻辑功能。

本书全部采用 ANSI/IEEE Std 91a-1991 标准符号，故在图 1-5 中做了比较。图 1-5 即 ANSI/IEEE Std 91a-1991 版与 ANSI/IEEE Std 91a-1984 版（与中国国标相同）的 IEC 国际标准逻辑门符号对照表。

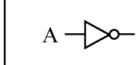
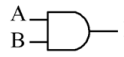
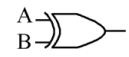
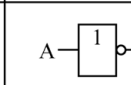
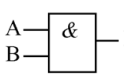
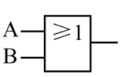
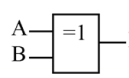
	非门	与门	或门	异或门
IEEE 1991 版标准逻辑符号				
IEEE 1984 版标准逻辑符号				
逻辑表达式	$\overline{A}$=NOT A	F=A • B	F=A+B	F=A⊕B

图 1-5　两种不同版本的国际标准逻辑门符号对照表

在流行的 EDA 软件中，逻辑符号采用的是 ANSI/IEEE Std 91a-1991 标准。由于 PLD 的复杂结构，用 1991 版本符号的好处是能十分容易地衍生出一套用于描述 PLD 复杂逻辑结构的简化符号。如图 1-6 所示，接入 PLD 内部的“与−或”阵列输入缓冲器电路，一般采用互补结构，它等效于图 1-7 的逻辑结构，即当信号输入 PLD 后，分别以其同相和反相信号接入。图 1-8 是 PLD 中“与”阵列的简化图形，表示可以选择 A、B、C 和 D 四个信号中的任一组或全部输入与门。在这里用以形象地表示“与”阵列，这是在原理上的等效。当采用某种硬件实现方法时（如 NMOS 电路），图中的与门可能根本不存在，但 NMOS 构成的连接阵列中却含有了“与”的逻辑。同样的道理，“或”阵列也用类似的方式表示。图 1-9 是 PLD 中“或”阵列的简化图形表示。图 1-10 是在阵列中连接关系的表示。十字交叉线表示两条线未连接；交叉线的交点上打黑点，表示固定连接，即在 PLD 出厂时已连接；交叉线的交点上打叉，表示该点可编程，即在 PLD 出厂后通过编程，其连接可随时改变。

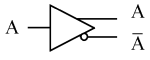


图 1-6　PLD 的互补缓冲器

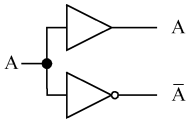


图 1-7　PLD 的互补输入

图 1-8　PLD 中“与”阵列的表示

图 1-9　PLD 中“或”阵列的表示

图 1-10　阵列线连接表示

PROM 作为可编程只读存储器，其 ROM 除作为只读存储器外，还可作为 PLD 使用。一个 ROM 器件主要由地址译码部分、ROM 单元阵列和输出缓冲部分构成。对 PROM 也可以从可编程逻辑器件的角度来分析其基本结构。

为了更清晰直观地表示 PROM 中固定的“与”阵列和可编程的“或”阵列，PROM 可以表示为 PLD 阵列图，以 4×2 PROM 为例，如图 1-11 所示。

式（1-1）是已知半加器的逻辑表达式，可用 4×2 PROM 编程实现。

$$S = A_0 \oplus A_1,\quad C = A_0 A_1 \qquad (1\text{-}1)$$

图 1-12 的连接结构表达的是半加器逻辑阵列。

$$F_0 = A_0\overline{A}_1 + \overline{A}_0 A_1，\quad F_1 = A_1 A_0 \tag{1-2}$$

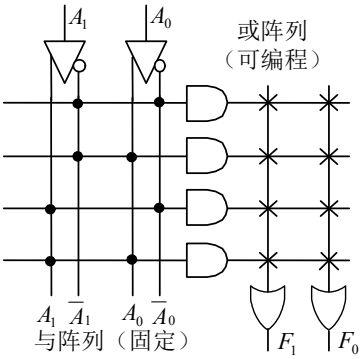


图 1-11　PROM 表达的 PLD 阵列图

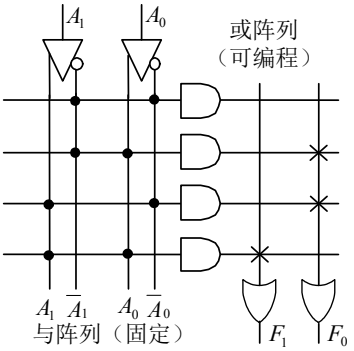


图 1-12　用 PROM 完成半加器逻辑阵列图

式（1-2）是图 1-12 结构的布尔表达式，即所谓的“乘积项”方式。式中的 A_1 和 A_0 分别是加数和被加数，F_0 为和，F_1 为进位。反之，根据半加器的逻辑关系，也可以得到图 1-12 的阵列点连接关系，从而可以形成阵列点文件，这个文件对于一般的 PLD 器件称为熔丝图（fuse map）文件。对于 PROM，则为存储单元的编程数据文件。

显然，PROM 只能用于组合电路的可编程上。输入变量的增加会引起存储容量的增加，这种增加是按 2 的幂次增加的，多输入变量的组合电路函数是不适合用单个 PROM 来编程表达的。

1.6.3　GAL

1985 年，美国 Lattice 公司在 PAL 的基础上推出了 GAL 器件，即通用阵列逻辑器件。GAL 首次在 PLD 上采用了 EEPROM 工艺，使 GAL 具有电可擦除重复编程的特点，彻底解决了熔丝型可编程器件的一次可编程问题。GAL 在“与-或”阵列结构上沿用了 PROM 的“与”阵列可编程、“或”阵列固定的结构（见图 1-13），但对 PROM 的 I/O 结构进行了较大的改进，即在 GAL 的输出部分增加了输出逻辑宏单元（output logic macro cell，OLMC），此结构使得 PLD 器件在组合逻辑和时序逻辑中的可编程或可重构性能都成为可能。

图 1-13 所示为 GAL16V8 型号的器件，它包含了 8 个逻辑宏单元（OLMC），每一个 OLMC 可实现时序电路可编程，而其左侧的电路结构是“与”阵列可编程的组合逻辑可编程结构。专业习惯是将 OLMC 及左侧的可编程“与”阵列合成一个逻辑宏单元，即标志 PLD 器件逻辑资源的最小单元，由此可以认为 GAL16V8 器件的逻辑资源是 8 个逻辑宏单元，而目前最大的 FPGA 的逻辑资源达数十万个逻辑宏单元。也有将逻辑门的数量作为衡量逻辑器件资源的最小单元，如某 CPLD 的资源约 2000 门等，但此类划分方法误差较大。

GAL 的 OLMC 单元设有多种组态，可配置成专用组合输出、专用输入、组合输出双向口、寄存器输出、寄存器输出双向口等，为逻辑电路设计提供了极大的灵活性。由于具有结构重构和输出端的功能均可移到另一输出引脚上的功能，在一定程度上简化了电路板的布局布线，使系统的可靠性进一步提高。

在图 1-13 中，GAL 的输出逻辑宏单元（OLMC）中含有 4 个多路选择器，通过不同的选择方式可以产生多种输出结构，分别属于 3 种模式，一旦确定了某种模式，所有的 OLMC

都将工作在同一种模式下。图 1-14 所示为其中一种输出模式对应的结构。

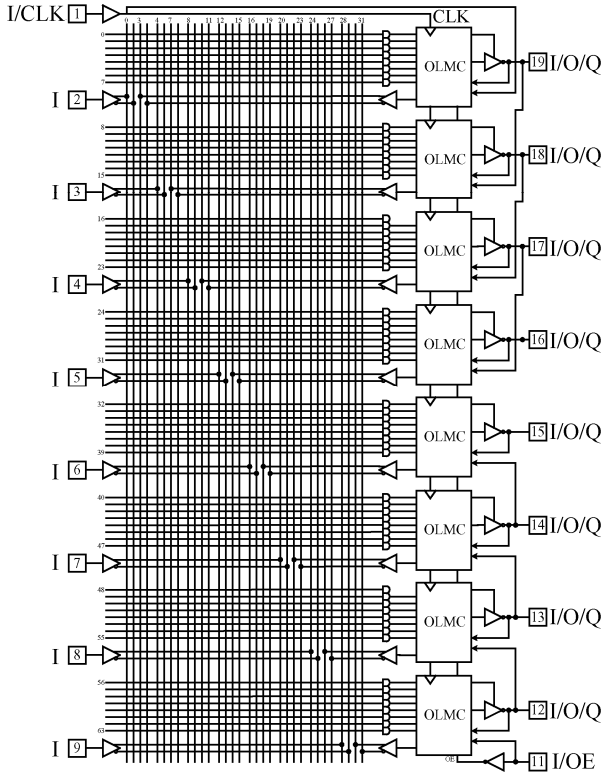


图 1-13　GAL16V8 的结构

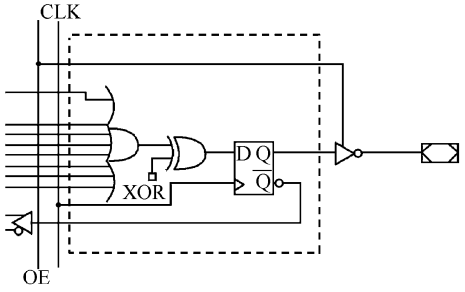


图 1-14　寄存器输出结构

1.7　CPLD 的结构与可编程原理

CPLD 即复杂可编程逻辑器件（complex programmable logic device）。早期 CPLD 是从 GAL 的结构扩展而来，但针对 GAL 的缺点进行了改进。在流行的 CPLD 中，Altera 的 MAX7000S 系列器件具有一定典型性，在这里以此为例介绍 CPLD 的结构和工作原理，其中的许多结构（如 I/O 结构）与 FPGA 也类似，望读者关注，并注意比较。

相比于 FPGA，CPLD 的逻辑资源要小得多。MAX7000S 系列器件包含 32～256 个逻辑宏单元（logic cell，LC），其单个逻辑宏单元结构如图 1-15 所示。每 16 个逻辑宏单元组成一个逻辑阵列块（logic array block，LAB）。与 GAL 类似，每个逻辑宏单元含有一个可编程的“与”阵列和固定的“或”阵列，以及一个可配置寄存器，每个逻辑宏单元共享扩展乘积项和高速并联扩展乘积项，它们可向每个逻辑宏单元提供多达 32 个乘积项，以构成复杂的逻辑函数。MAX7000S 结构包含 5 个主要部分，即逻辑阵列块、逻辑宏单元、扩展乘积项（共享和并联）、可编程连线阵列、I/O 控制块。以下简要介绍相关模块。

1．逻辑阵列块

一个 LAB 由 16 个逻辑宏单元的阵列组成。MAX7000S 结构主要是由多个 LAB 组成的阵列以及它们之间的连线构成。多个 LAB 通过可编程连线阵列（programmable interconnect