SoC 设计和测试技术：理论与实践

刘文松　朱　恩　赵春光
徐　勇　欧乐庆　茅文深　·编著

东南大学出版社
SOUTHEAST UNIVERSITY PRESS

SoC 设计和测试技术：理论与实践

刘文松　朱　恩　赵春光
徐　勇　欧乐庆　茅文深　编著

东南大学出版社
SOUTHEAST UNIVERSITY PRESS
·南京·

内 容 提 要

本书共分7章内容：首先，整体介绍VLSI设计技术的发展现状和重点问题。其次，系统讲述硬件描述语言、可编程逻辑器件、逻辑综合、自动布局布线等理论知识。第三，融合工程实践，对SoC设计和测试流程中的理念和方法展开论述。

图书在版编目(CIP)数据

SoC设计和测试技术：理论与实践/刘文松等编著.
—南京：东南大学出版社，2016.9
ISBN 978-7-5641-6780-6

Ⅰ.①S… Ⅱ.①刘… Ⅲ.①集成电路-芯片-设计
②集成电路-芯片-测试 Ⅳ.①TN4

中国版本图书馆CIP数据核字(2016)第237091号

SoC设计和测试技术：理论与实践

出版发行	东南大学出版社
出 版 人	江建中
社　　址	南京市四牌楼2号
邮　　编	210096
经　　销	全国各地新华书店
印　　刷	江苏凤凰数码印务有限公司
开　　本	700 mm×1000 mm　1/16
印　　张	12.5
字　　数	245千字
版　　次	2016年9月第1版
印　　次	2016年9月第1次印刷
书　　号	ISBN 978-7-5641-6780-6
定　　价	39.00元

(本社图书若有印装质量问题，请直接与营销部联系，电话：025—83791830)

前　言

近年来，我国对集成电路产业的政策支持力度空前。2014 年 6 月，国务院印发《国家集成电路产业发展推进纲要》，制定三阶段目标。第一阶段：到 2015 年，实现集成电路产业销售收入超过 3 500 亿，32/28 nm 实现规模量产，65/45 nm 关键设备和 12 英寸硅片等关键材料进入应用。第二阶段：到 2020 年，行业销售收入年均增速超过 20%。第三阶段：到 2030 年，集成电路产业链主要环节达到国际先进水平，部分企业进入国际第一梯队。在出台政策的同时，正式成立国家集成电路产业投资基金，基金总规模达到 1 387.2 亿元，已参与投资近 600 亿的相关产业项目。

在国家政策和资金的双重支持下，我国集成电路技术实力显著增强，设计能力与国际先进水平差距大幅缩小，制造工艺取得长足进步，出现了诸如海思、展讯、炬力等一大批优秀的国内集成电路设计公司。

SoC 是在由集成电路向集成系统转变的大方向下产生的。随着半导体制造工艺的进一步提升，芯片集成度越来越高，可将接口电路、信息处理、模数转换、内存存储、逻辑控制等系统功能集成到一个芯片中，打破了原有的整机、电路与元件、器件之间的界限。SoC 中的接口电路、信息处理、模数转换、内存存储等模块可以 IP 核复用的形式实现。通常包含一个或多个处理核，以软件形式实现电路系统整体的逻辑控制。软硬件协同设计是 SoC 和传统 ASIC 设计最大的区别之一。

SoC 可以有效降低电子信息系统/设备的开发成本，缩短开发周期，提高产品的竞争力。同时，产品适合于大规模批量生产，成本低和可靠性高。正是在市场需求和技术发展的双重驱动下，SoC 逐渐成为集成电路设计公司采用的主流开发方式。随着 SoC 集成规模的日趋庞大，SoC

设计流程、SoC 架构、电子级系统设计、IP 核设计与选择、RTL 代码编写、先进的验证方法、低功耗设计、可测性设计技术及后端设计等均对最终产品的性能和可靠性产生巨大影响。因此，相关的理论基础和理念方法也日益为工业界所重视。

笔者在东南大学信息学院射频与光电集成电路研究所朱恩教授指导下，就相关领域的研究内容展开论述，涉及整个设计流程中的硬件描述语言、可编程逻辑器件、逻辑综合、自动布局布线等理论知识，并结合工程实践讲述了 SoC 设计和测试的主要理念和方法等内容。本书同时汇聚了射频与光电集成电路研究所众多校友的辛勤付出。Synopsys 公司孟凡生校友为逻辑综合工具章节的撰写提供大量资料，高通公司黄宁校友完成 SoC 测试章节的撰写，华为公司张亮校友为自动布局布线章节的撰写提供大量资料，在此一并感谢！最后，希望能以此为机会，与更多的集成电路设计人员就相关的理论方法和工程实践共同展开交流探讨，共同夯实基础并提升电路设计水平。

编　者

2016 年 7 月

目　录

1　SoC 设计概述 …… (1)

1.1　发展概貌 …… (1)

1.2　主要设计方法——自顶向下方法 …… (2)

1.3　设计流程中的重点问题 …… (3)

1.4　工具的支持 …… (5)

2　硬件描述语言 Verilog …… (6)

2.1　Verilog 语言的一般结构 …… (6)

2.1.1　模块 …… (6)

2.1.2　数据流描述方式 …… (6)

2.1.3　行为描述方式 …… (7)

2.1.4　结构描述方式 …… (8)

2.1.5　混合描述方式 …… (8)

2.2　Verilog 语言要素 …… (9)

2.2.1　标识符、注释和语言书写的格式 …… (9)

2.2.2　系统任务和函数 …… (10)

2.2.3　编译指令 …… (10)

2.2.4　值集合 …… (11)

2.2.5　数据类型 …… (13)

2.2.6　位选择和部分选择 …… (15)

2.2.7　参数 …… (15)

2.3　表达式与操作符 …… (16)

2.4　结构描述方式 …… (19)

2.4.1　常用的内置基本门 …… (19)

2.4.2　门时延问题 …… (20)

2.4.3　门实例数组 …… (20)

2.4.4　模块和端口 …… (21)

2.4.5　模块实例语句 …… (21)

2.4.6 模块使用举例 …… (22)
2.5 数据流描述方式 …… (23)
2.5.1 连续赋值语句 …… (23)
2.5.2 举例 …… (24)
2.5.3 连线说明赋值 …… (24)
2.5.4 时延 …… (24)
2.5.5 连线时延 …… (25)
2.5.6 举例 …… (26)
2.6 行为描述方式 …… (27)
2.6.1 过程结构 …… (27)
2.6.2 时序控制 …… (28)
2.6.3 语句块 …… (30)
2.6.4 过程性赋值 …… (32)
2.6.5 if 语句 …… (34)
2.6.6 case 语句 …… (34)
2.6.7 循环语句 …… (35)
2.7 设计共享 …… (36)
2.7.1 任务 …… (36)
2.7.2 函数 …… (37)
2.7.3 系统任务和系统函数 …… (38)
2.8 HDL 仿真软件简介 …… (41)
3 可编程逻辑器件 …… (51)
3.1 引言 …… (51)
3.2 GA 概述 …… (51)
3.3 PLD 概述 …… (52)
3.3.1 PLD 的基本结构 …… (52)
3.3.2 PLD 的分类 …… (53)
3.3.3 PROM 阵列结构 …… (53)
3.3.4 PLA 阵列结构 …… (54)
3.3.5 PAL(GAL)阵列结构 …… (54)
3.3.6 FPGA(Field Programmable Gate Array) …… (55)
3.3.7 PLD 的开发 …… (60)

3.4　FPGA 的开发实例 …… (61)
3.4.1　Quartus II 的启动 …… (62)
3.4.2　建立新设计项目 …… (63)
3.4.3　建立新的 Verilog HDL 文件 …… (65)
3.4.4　建立新的原理图文件 …… (66)
3.4.5　设置时间约束条件 …… (67)
3.4.6　引脚绑定 …… (69)
3.4.7　编译 …… (70)
3.4.8　仿真 …… (72)
3.4.9　器件编程 …… (74)
4　逻辑综合 …… (76)
4.1　引言 …… (76)
4.2　组合逻辑综合介绍 …… (76)
4.3　二元决定图(Binary-Decision Diagrams) …… (79)
4.3.1　ROBDD 的原理 …… (79)
4.3.2　ROBDD 的应用 …… (81)
4.4　Verilog HDL 与逻辑综合 …… (82)
4.5　逻辑综合的流程 …… (86)
4.6　门级网表的验证 …… (90)
4.6.1　功能验证 …… (90)
4.6.2　时序验证 …… (91)
4.7　逻辑综合对电路设计的影响 …… (91)
4.7.1　Verilog 编程风格 …… (92)
4.7.2　设计分割 …… (94)
4.7.3　设计约束条件的设定 …… (95)
4.8　时序电路综合举例 …… (96)
4.9　Synopsys 逻辑综合工具简介 …… (102)
4.9.1　实例电路——m 序列产生器 …… (103)
4.9.2　利用 Synopsys 的 Design Compiler 进行综合的基本过程 …… (104)
4.10　总结 …… (109)
5　自动布局布线 …… (110)
5.1　自动布局布线的一般方法和流程 …… (110)

5.1.1 数据准备和输入 …… (110)
5.1.2 布局规划、预布线、布局 …… (111)
5.1.3 时钟树综合 …… (112)
5.1.4 布线 …… (114)
5.1.5 设计规则检查和一致性检查 …… (115)
5.1.6 输出结果 …… (115)
5.1.7 其他考虑 …… (115)
5.2 自动布局布线软件介绍 …… (115)
5.2.1 Apollo 一般情况介绍 …… (116)
5.2.2 Apollo 库的文件结构 …… (116)
5.2.3 逻辑单元库——TSMC 0.25 μm CMOS 库 …… (117)
5.3 自动布局布线的处理实例 …… (117)
5.3.1 电路实例 …… (117)
5.3.2 数据准备和导入 …… (127)
5.3.3 数据导入步骤 …… (127)
5.3.4 布图 …… (129)
5.3.5 预布线 …… (133)
5.3.6 单元布局 …… (135)
5.3.7 布线 …… (137)
5.3.8 数据输出 …… (139)
5.3.9 自动布局布线的优化 …… (140)
6 SoC设计 …… (143)
6.1 SoC的基本概念 …… (143)
6.1.1 SoC的特征和条件 …… (143)
6.1.2 SoC的设计方法学问题 …… (144)
6.2 基于平台的SoC设计方法 …… (149)
6.2.1 一般方法 …… (149)
6.2.2 设计分工 …… (151)
6.3 ARM平台SoC设计方法 …… (153)
6.3.1 简介 …… (153)
6.3.2 标准的SoC平台 …… (155)
6.3.3 支持工具和验证方法 …… (158)

6.3.4　操作系统端口 …… (163)
6.3.5　ARM 的扩展 IP …… (163)
6.3.6　第三方伙伴计划 …… (164)
6.4　研究方向 …… (164)
7　SoC 测试方法 …… (166)
7.1　引言 …… (166)
7.2　测试步骤 …… (166)
7.3　常用的可测试性设计方法 …… (168)
7.3.1　扫描路径法 …… (168)
7.3.2　内建自测试法 …… (170)
7.3.3　边界扫描法 …… (172)
7.4　缺陷和故障 …… (177)
7.4.1　缺陷分类 …… (177)
7.4.2　故障模型及其分类 …… (177)
7.5　测试向量生成 …… (183)
7.6　SoC 测试面临的挑战 …… (184)
参考文献 …… (186)

1 SoC设计概述

1.1 发展概貌

集成电路的开发始于1958年，当时有得州仪器公司和仙童半导体公司两家企业在同时开展。时至今日，半导体已经成为不断前行的电子产业中最为核心的部分。随着科技的不断发展，半导体技术一直遵循着摩尔规律向前推进。1965年，摩尔指出：晶体管和电阻的数量每年会翻一番，原因是工程师们正不断缩小晶体管的体积，这被称为摩尔定律。它意味着半导体的性能与容量将以指数级增长，并且这种增长趋势还将延续下去。1975年，摩尔再次指出：每隔24个月晶体管的数量将翻一番。如今，历史证明了摩尔定律的正确性：在21世纪初期，主流集成电路设计从0.18 μm逐步发展到0.13 μm，进而实现了90 nm、45 nm和22 nm。2013年9月，英特尔公司首次展示了代号为“Broadwell”的下一代处理器芯片，采用14 nm工艺制造。除了在工艺上改进，2011年5月，英特尔公司成功开发出首个3D晶体管，称为tri-Gate。业界许多专家估计，著名的摩尔定律还将使用至少10年，目前的工艺仍有巨大的改进潜力。

市场需求和微电子自身的发展带动了以微细加工（集成电路特征尺寸不断缩小）为主要特征的多种工艺集成技术和面向应用的系统级芯片的发展。随着半导体产业进入纳米加工时代，单一集成电路芯片就可以实现一个复杂的电子系统，诸如手机芯片、数字电视芯片、机顶盒芯片等。片上系统（System on Chip，SoC）设计技术始于20世纪90年代中期。随着半导体工艺技术的发展，IC设计者能够将越来越复杂的功能集成到单个芯片上，SoC正是在集成电路（Integrated Circuit，IC）向集成系统（Integrated System，IS）转变的大方向下产生的。1994年摩托罗拉公司发布的FlexCore系统（用来制作基于68000和PowerPC的定制微处理器）和1995年LSI Logic公司为索尼公司设计的SoC，可能是基于IP（Intellectual Property）核完成SoC设计的最早报道。SoC可以利用成熟IP来降低整个设计的复杂度和风险性，显著地提高了ASIC的设计能力并加快了产品的上市时间，因此发展得非常迅速，引起了工业界和学术界的广泛关注。

SoC的高集成度也给设计者和测试者带来了许多新的挑战。纳米工艺下的SoC测试需要处理测试数据量大、测试功耗过高和测试时间过长这三个主要问题，

而它们最终都将导致 SoC 测试成本急剧增加。在这个 SoC 技术快速发展的时期，如果不给复杂 SoC 的测试和可测试性设计以足够的重视，测试就有可能成为 SoC 设计项目中的“瓶颈”，甚至会制约 SoC 技术的向前发展。国内在 SoC 领域的研究起步较晚，且大多集中在 SoC 设计方面，给予 SoC 测试与可测试性设计的关注及投入相对较少，因此 SoC 的测试与可测试性设计技术是我国 IC 产业链中的薄弱环节之一。另外，从国防安全的角度出发，对 SoC 进行自主研发、生产及测试是十分必要的。

1.2 主要设计方法——自顶向下方法

采用自顶向下(Top-down)方法的设计次序是：行为设计、结构设计、逻辑设计、电路设计和版图设计，如图 1.1 所示。

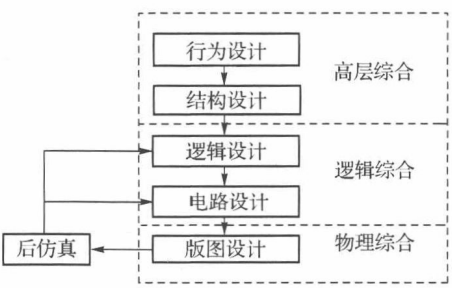

图 1.1 Top-down 方法的设计次序

在行为设计阶段：要确定芯片的功能、性能、面积、工艺和成本。

在结构设计阶段：要根据芯片的特点，将其分解为接口清晰、相互关系明确、尽可能简单的子系统，再利用子系统设计出一个较好的总体结构，这个结构可能包括有算术运算单元、控制单元、数据通道、各种算法状态机等。

在逻辑设计阶段：要考虑各种功能模块的具体实现问题。由于同一功能块可能有多种实现方法，所以在这一阶段，要尽可能采用规则结构来实现功能块，同时要充分利用已经过考验的逻辑单元或模块。这一阶段要进行逻辑仿真，以确定逻辑设计的正确性。

在电路设计阶段：逻辑图将进一步转换成电路图，在这个阶段，可能需要进行电路仿真，以确定电路特性、功耗和时延等。

在版图设计阶段：要根据电路图绘制用于工艺制造的电路版图。完成版图后，需要先对实际版图进行参数提取，再进行电路后仿真。如果仿真性能达不到要求，需要返回到电路设计或进一步到逻辑设计，进行适当修改，最终达到设计目标。

1.3 设计流程中的重点问题

在图1.1中，典型的设计流程被划分成三个综合阶段：高层综合、逻辑综合和物理综合。

在英文文献中，表示综合的词是Synthesis，Synthesis的原义是合成、综合，意指用合成方法合成出新的事物，在集成电路设计领域，综合是指将一种形式的设计转换成另一种设计形式的过程。

1）高层综合

高层综合也称行为级综合(Behavioral Synthesis)，高层综合就是将系统的行为、各个组成部分的功能及其输入和输出，用硬件描述语言HDL(如VHDL和Verilog)加以描述，然后进行行为级综合，同时通过高层次硬件仿真进行验证。

高层综合的任务是将一个设计的行为级描述转换成寄存器传输级的结构描述。它首先翻译和分析用HDL语言描述的设计，并在给定的一组性能、面积或功耗的约束条件下，确定需要哪些硬件资源，如执行单元、存储器、控制器、总线等，以及确定在这一结构中各种操作的次序。还可通过行为级和寄存器传输级硬件仿真进行验证。由于同一个设计可能有多种硬件结构的实现方式，因此，高层综合的目的就是要在满足目标和约束的条件下，找到一个代价最小的硬件结构，并使设计的功能最佳化。

2）逻辑综合

逻辑综合将逻辑级行为描述转换成使用门级单元的结构描述(门级结构描述称为网表描述)，同时还要进行门级逻辑仿真和测试综合。

逻辑级行为描述可以是状态转换图、有限状态机，也可以是布尔方程、真值表或硬件描述语言。逻辑综合过程还包括一系列优化步骤，如资源共享、连接优化和时钟分配等。优化的目标是面积、速度、功耗等指标中的一种或几种的折中。

逻辑综合一般分两个阶段：①与工艺无关的阶段：这时采用布尔操作或代数操作技术来优化逻辑；②工艺映射阶段：根据电路性质(如组合型或时序型)及采用的结构(多层逻辑、PLD或FPGA)做出具体的映象，将与工艺无关的描述转换成门级网表、PLD或FPGA中的一种执行文件。

在逻辑综合的过程中，需要进行细致的时序和时延分析以及逻辑仿真。逻辑仿真是保证设计正确的关键步骤，过去通常采用软件模拟的方法，近年来则强调硬件手段，如通过PLD或FPGA进行。测试综合是提供测试图形的自动生成，为可测性设计提供高故障覆盖率的测试图形。测试综合可以消除设计中的冗余逻辑，诊断不可测的逻辑结构，还可以自动插入可测性结构。

3）物理综合

物理综合也称版图综合(layout synthesis)，物理综合将网表描述转换成版图。这时对每个单元要确定其几何形状、大小及位置，还要确定单元间的连接关系。它的任务是将门级网表自动转换成版图。布图的详细步骤如图 1.2 所示。

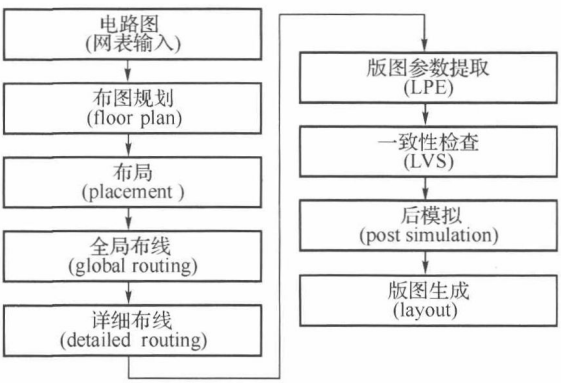

图 1.2　布图的详细步骤

布图规划是对设计进行物理划分，同时对设计的布局进行规划和分析。在这一步骤里，面向物理的划分，其层次结构可以与逻辑设计时的划分有所不同。布图规划可以估算出较为精确的互连延迟信息、预算芯片的面积，分析布线的稀疏度。

布局是指将模块在满足一定的目标函数的前提下布置在芯片上的适当位置。一般布局时总是要求芯片面积最小、连线总长最短、电性能最优并且容易布线。布局又分为初始布局和迭代改善两个子步骤。进行初始布局的目的是为了提高布局的质量以及减少下一阶段即迭代改善时的处理次数，而迭代改善是一个优化的过程，它是决定布局质量的关键。

布线是根据电路连接关系的描述(即连接表)，在满足工艺规则和电学性能的要求下，在指定区域(面积、形状、层次等)内完成所需的全部互连，同时尽可能地对连线长度和通孔数目进行优化。一般有两种布线方法：一种是面向线网的布线方法，它是直接对整个电路进行布线，布线时通常采取顺序方式；另一种称为分级布线，它将布线问题分为全局布线和详细布线，这是一种面向布线区域的布线方法。这种方法通过适当的划分，将整个布线区域分为若干个布线通道区，然后进行适当的布线分配，即将一个线网的所有端点的走线路径分配到相应的通道区中，接着是进行详细布线，对分配到当前通道区中的所有线网段集合，按照一定的规则，确定它们在通道中的具体位置。

在完成布局布线后，要对版图进行设计规则检查、电学规则检查以及版图与电路图的一致性检查。在版图寄生参数提取的基础上，再次进行电路分析(即后模

拟)。只有在所有的检查都通过并证明正确无误后,才可将布图结果转换为掩膜文件。然后通过掩膜版发生器或电子束制版系统,将掩膜文件转换生成掩膜版。

上述设计流程又可分成前端设计和后端设计两个阶段,这是在深亚微米电路出现之前,人们就已习惯的分法。

前端设计主要进行系统和功能的设计以及结构和电路的设计。

后端设计主要完成版图设计。

前端和后端之间的沟通主要通过网表和单元库,前端设计完成后将网表交给版图设计人员。一般而言,只要布线能够通过,时序要求基本上都能够得到满足。

1.4 工具的支持

20世纪70年代初,为了解决电路和版图设计自动化问题,出现了计算机辅助设计(CAD)技术;80年代,为了解决结构和逻辑设计的自动化问题,出现了计算机辅助工程(CAE)技术;90年代初,为了解决行为设计的自动化问题,出现了电子设计自动化(EDA)技术。在CAD/CAE技术中,主要解决给定电路图到版图和测试生成的自动转化问题。在EDA技术中,主要解决给定概念到电路图的自动转化问题。新出现的技术一般涵盖和改进了已有的技术,例如EDA技术涵盖了CAD/CAE技术等。

CAD技术主要用来帮助设计人员进行电路性能的分析和完成电路版图的编辑,如何在计算机上完成复杂的运算并解决图形的编辑及设计规则的检查是主要需要解决的问题。CAD技术主要是以晶体管级的性能仿真以及以多层版图的手工编辑为主,并形成了众多的单点工具。CAD技术的出现将部分设计工作从手工劳动当中解放出来,同时建立了模拟的概念和基本理论。

CAE技术主要用来进行集成电路设计、开发与生产过程的标准化,同时实现各个层面上集成电路设计活动的自动化。这种技术需要解决的主要难题是如何建立集成电路的标准设计库以及如何解决不同层面设计的接口等问题。这个时期,出现了标准设计库,出现了硬件描述语言Verilog和VHDL,出现了全线设计工具。CAE技术孕育了众多的新思想,例如可测性设计,出现了集成电路设计工具的全线产品,形成了集成电路设计、开发和生产的流程。

EDA技术主要用来实现从概念到电路的设计自动化,实现电子系统设计与芯片设计的融合,需要解决的主要难题是电路综合理论的创立和完善、电路验证理论的出现和发展。这个时期的主要特征是广泛使用硬件描述语言和逻辑综合进行电路设计,电路验证理论和方法逐渐成熟,行为综合理论开始形成,产生了众多的新思想和新方法,例如时序验证等,实现了集成电路设计工作的重点转移,形成了新一代集成电路设计方法学。

2 硬件描述语言 Verilog

Verilog HDL 是一种硬件描述语言，可用于从算法级、门级、寄存器级到开关级的多层次的数字电路系统建模，也可用于时序建模。用 Verilog 编写的模型可用 Verilog 仿真器进行验证。

Verilog 从 C 语言中继承了多种操作符和结构，并提供了扩展的建模能力。

Verilog 语言的国际标准为 IEEE Std 1364—1995(1995 年)。

2.1 Verilog 语言的一般结构

2.1.1 模块

模块是 Verilog 的基本描述单位，用于描述某个设计的功能或结构，同时也可描述与其他模块通信的外部端口。一个模块可以调用另一个模块。

下面是一个半加器电路的设计实例：

```
module HalfAdder(A,B,Sum,Carry);
input A,B;
output Sum,Carry;
assign #2 Sum=A^B;
assign #5 Carry=A & B;
endmodule
```

该模块的名字是 HalfAdder，模块的输入端口为 A 和 B，输出端口为 Sum 和 Carry。

在模块中，可用下述方式描述一个设计：

(1) 数据流描述方式；

(2) 行为描述方式；

(3) 结构描述方式；

(4) 混合描述方式。

下面通过实例介绍这些方式。

2.1.2 数据流描述方式

用数据流方式描述一个设计的最基本机制就是使用连续赋值语句，例如：

```
module HalfAdder(A,B,Sum,Carry);
input A,B;
output Sum,Carry;
assign #2 Sum=A^B;
assign #5 Carry=A & B;
endmodule
```

模块 HalfAdder 有 2 个输入端口和 2 个输出端口，以 **assign** 为前缀的语句是连续赋值语句；连续赋值语句是并发执行的，各语句的执行顺序与其在描述中出现的顺序无关。

2.1.3 行为描述方式

电路的行为使用下述过程语句描述：

(1) **initial** 语句：此语句只执行一次。

(2) **always** 语句：此语句总是循环执行。

在这两种语句中被赋值的对象只能是寄存器类型的。

语句 **initial** 和 **always** 在 0 时刻开始并发执行。

例如：

```
module FA(A,B,Cin,Sum,Cout);
    input A,B,Cin;
    output Sum,Cout;
    reg Sum,Cout;
    reg T1,T2,T3;
        always@(A or B or Cin)
        begin
                Sum=(A^B)^Cin;
                T1=A & Cin;
                T2=B & Cin;
                T3=A & B;
                Cout=(T1|T2)|T3;
        end
endmodule
```

其中，**reg** 是寄存器数据类型的一种。**always** 语句中字符@后面的是事件控制语句。相关联的顺序过程是 **begin** 和 **end** 包含的语句。只要 A、B 或 Cin 中至少有一个值发生变化，顺序过程就执行。