普通高等教育“十五”国家级规划教材

数字系统自动设计实用教程

刘明业　主编

高等教育出版社

内容提要

本书较全面而深入地介绍了 EDA 技术的理论和实践，主要讨论应用计算机设计数字系统（计算机/ASIC）的有关理论和算法，力求做到循序渐进，深入浅出。全书共分 11 章，主要内容包括：数字系统设计基础，VHDL 语言编译方法与模拟技术，自动逻辑综合理论与时序逻辑综合的基本问题，数据流综合中的操作调度与资源分配问题的有关理论与算法，工艺映射技术，逻辑图自动生成技术以及 VHDL 语言设计操作等。

本书可以作为普通高等学校计算机软、硬件专业以及信息处理、自动控制、电子工程和通信技术等专业的高年级本科生和研究生的教材，也可供从事 EDA 工具研发和 ASIC 设计的教师和技术人员参考使用。

书后附有光盘，其中的教学软件内容涵盖高级综合全过程，可供师生在教学实践中使用，以更好地掌握教学内容。

图书在版编目（CIP）数据

数字系统自动设计实用教程/刘明业主编. —北京：高等教育出版社，2004.7
ISBN 7-04-014611-8

Ⅰ.数… Ⅱ.刘… Ⅲ.数字系统-系统设计：计算机辅助设计-高等学校-教材 Ⅳ.TP271

中国版本图书馆 CIP 数据核字（2004）第 062156 号

策划编辑 刘建元　　**责任编辑** 董建波　　**市场策划** 陈　振
封面设计 李卫青　　**责任印制** 杨　明

出版发行	高等教育出版社	**购书热线**	010-64054588
社　　址	北京市西城区德外大街 4 号	**免费咨询**	800-810-0598
邮政编码	100011	**网　　址**	http://www.hep.edu.cn
总　　机	010-82028899		http://www.hep.com.cn
经　　销	新华书店北京发行所		
印　　刷	人民教育出版社印刷厂		
开　　本	787×1092　1/16	**版　　次**	2004 年 7 月第 1 版
印　　张	23.75	**印　　次**	2004 年 7 月第 1 次印刷
字　　数	500 000	**定　　价**	40.00 元（含光盘）

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

前　言

本书为普通高等教育“十五”国家级规划教材，其内容为应用计算机自动设计数字系统（计算机/ASIC）的有关理论和算法。涉及电子设计自动化（EDA）领域中的硬件描述语言（VHDL/Verilog）、模拟技术、自动综合理论、操作调度与资源分配、工艺映射以及逻辑图自动生成等问题。本书以作者研发的专用集成电路高层次自动设计系统 Talent 为依托，参考国内外同类工作发展状况，结合国内当前教学的实际需要，系统而全面地阐述了数字系统自动设计有关基础性问题，主要限于高层次设计（前台设计）的问题。

第一章为数字系统自动设计基础，阐明设计描述的层次划分，举例详述高级综合所要解决的基本问题、综合的硬件目标的体系结构模型以及自动设计的全过程。最后给出一个设计自动化原型系统（Talent），暗示全书涵盖的基本内容。

第二、三、四章讨论 VHDL 语言及其编译的基本方法与模拟技术，其中 VHDL 语言参照 IEEE Standards VHDL Language Reference Manual（IEEE Std 1076，2000 年版）的基本框架，但远远没有涵盖其全部内容及有关的细节规定。当今 VHDL 语言规模之广，内容之深，本教材因篇幅所限难以详述，只能起到入门的作用。有志深究者，请读原本参考文献[6]，或参考其编译本参考文献[2]。

VHDL 编译方法一章阐明了 VHDL 编译中的特定问题，给出一个简例的最终形式，而对现代 VHDL 编译的中间数据结构的 CDFG/DFG 以及 Petri 网等很少述及。现在有大量研究成果发表，可供读者借鉴。

模拟技术一章主要讲述模拟的基本概念和 VHDL 的模拟特性，因此本教材也着重阐明其要点，没有也不可能详述一个真实的模拟系统，只借助一个原型的构思，道出一个完整的模拟器的全貌。读者如果真能领会，以此为起点，阅读文献，也不难研发出自己所期待的模拟系统。

如果读者已熟悉高级程序设计语言（如 C/C++ 语言），具有形式语言及其编译方法的基本知识，在学习过程中可注意分析、比较 VHDL 与 C/C++ 具有某些类似的语句与语法现象，这将有助于更深入地理解和掌握 VHDL 语言及其编译和模拟技术。

第五、六章讨论自动逻辑综合和时序逻辑综合的基本问题，目的是将读者带入高级综合中的控制流综合的领域。

第七、八章讨论数据流综合中的操作调度与资源分配，大都是经典的理论与方法。近年来即使在国内高级综合的问题也获得认可，高级综合的研究成果如雨后春笋般涌现。对未来的 ASIC 高级综合怀有憧憬者，本书所述及的基本内容，可以为读者今后的深入研究奠定

基础。

第九章工艺映射技术,主要讨论基于规则的多层次、多目标工艺映射及面向 FPGA/CPLD 的工艺实现。这些年来现场可编程器件的发展突飞猛进,本书也只作些基础性论述。

第十章逻辑图自动生成技术,讨论以 VHDL 语言结构描述或连线表自动生成的划分、布局及布线等的原理与算法。这是数字系统自动设计结果的一种重要存档形式。其理论与方法与 PCB 设计类同,并且划分方法对于调度、分配前 CDFG/DFG 行为描述的划分也是可借鉴的。

第十一章以 Talent 应用为例讨论自动设计中的实际操作方法,通过举例说明测试向量的引入,模拟运行,综合与工艺映射及其验证,乃至逻辑图自动生成中的分页、出图等问题。

书中附有光盘,内容为本课程的教学实践软件,其功能涵盖高级综合、工艺映射和逻辑图自动生成以及模拟验证。读者可在微机上运行某些题目,借以更深入地掌握 VHDL 高级综合技术。须知,本软件仅供教学实验用,其功能也很大程度上取决于读者对它的熟悉和掌握。

每章后都附有复习思考题供读者学习,参考文献前的附录给出了 VHDL 语言词法元素、VHDL 语言词汇表及本书专用 EDA 词汇表,供读者在深入研究 VHDL 语言及 EDA 技术时参考。"

本书的内容可分成相对独立的几大块。例如,第二、三、四章讲语言、编译及模拟,主要针对从事 VHDL 语言设计的读者;第五、六章针对数字逻辑电路的自动设计;第七、八、九章着重讨论高级综合等。授课老师可根据需要与教学计划学时数灵活选择。

书中讲述的内容是针对 EDA 技术发展的主流,而不完全对应于 Talent 的每一部分。此外近些年 EDA 技术及半导体制造技术以惊人的速度在飞速发展,并出现了许多新的发展领域,如深亚微米技术的集成电路设计、可测试性设计技术、低功耗电路的设计技术等,限于篇幅和编者本人的能力,这些宝贵的篇章都一律割爱了。

本书选材的主要目标是面对国内的现实,能在更多的普通高等学校的计算机专业中早日开设出这门课程,为未来有志从事 EDA 工作的学生打下基础。

本教材的读者对象是普通高等学校计算机软、硬件专业,以及信息处理、自动控制、电子工程和通信技术等专业的高年级本科生和研究生,也可供从事 EDA 工具研发的教师和技术人员参考。

刘明业主持并完成本书稿的编写,参加编写的还有陈东瑛、石峰、吴清平、谢巍、袁媛、王作建、刘沁楠、赵阳生及叶梅龙。陈辉煌教授在百忙中接受高等教育出版社的函聘,审阅了全书内容。本书的编写工作承蒙厦门大学各级领导、师长和朋友的关怀、指导和帮助。谨此一并致以诚挚的谢意。

限于我们工作的努力和认识程度,书中难免存在缺点、疏忽,甚至谬误,恳切希望广大读者批评指正。

刘明业

2003 年除夕于厦门大学

目　录

第一章　数字系统自动设计基础

1.1　计算机自动设计和层次式描述

现代计算机是相当复杂的电子系统，只有按其自然的结构和层次进行设计，才能有条不紊地完成设计任务。这就是通常所说的“由顶向下”(Top - down)和“由底向上”(Bottom - up)的设计方法(如图 1 - 1 所示)。从本质上来说，数字系统自动设计的过程就是在某些约束条件下一系列的信息变换和扩展，由上一级的设计(例如行为功能级)细化到下一级(如逻辑级)的过程，各级的行为功能应该是等价的。理想的自动化设计系统，最好是在设计人员输入系统描述和各种约束条件后，计算机能自动生成逻辑图或微程序，然后再自动扩展成工程设计，并生成测试数据。这种系统能够将较高设计层次的“粗框图”扩展到下一个层次较为详细的“细框图”。目前最为实用的还是设计验证，例如对某一设计层次的行为或逻辑功能的验证，或者与其下一层次的元件构成同一系统行为的等价性进行对比验证等。

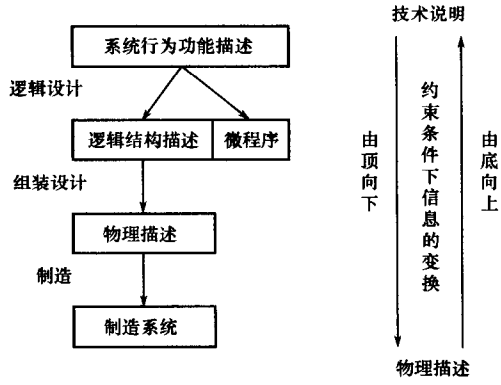


图 1 - 1　由顶向下及由底向上设计过程

数字系统的描述实质上就是在相应层次上对其行为功能特性的定义，并且这常常分为多个层次进行，如系统的行为(功能)级、体系结构级/寄存器传输级、逻辑级和电路级等。设计人员在不同的层次上进行设计，只考虑在特定层次上感兴趣的问题，因而可以简化设计的复杂程度。当然，这就要求每个层次上的描述都是无二义性的。现在利用 Y 形图(如图1 - 2 所示)来研讨设计层次与描述范畴的问题。该图中沿每个轴将数字系统的描述划分为 4 个层次，每个层次上又都包含 3 个描述范畴：行为领域描述、结构领域描述和物理领域描述。沿 Y 轴离中心愈远的层次愈抽象。行为领域描述是定义系统的功能，相应于不同的层次，可以用系统行为的流图、算法、寄存器传输级方程、布尔方程及微分方程等不同的方法进行描述。结构领域描述是描述构成数字系统的元件及它们之间的连接关系，这可按元件的规模分为 4 个不同的层次：处理器及存储器、硬件模块级，ALU、MUX 和寄存器级，门和触发器级以及

晶体管(电路)级等。物理领域描述是描述数字系统的物理实现特征,可以分别描述底板、多芯片模块、芯片、单元以及掩模板图形、多边形等。而各轴之间有箭头的弧表示设计工具的作用。只有将上述不同领域、不同层次的信息结合起来,才能全面地描述出一个完整的数字系统。不同抽象层次上的设计对象如图表 1 - 1 所示。

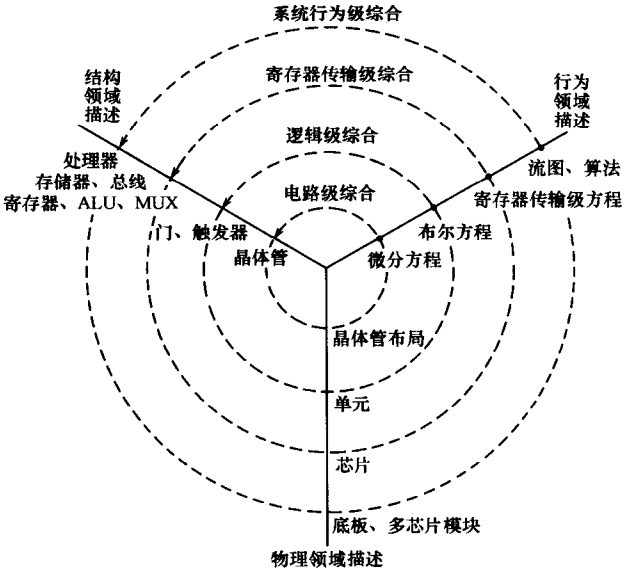


图 1 - 2　描述范畴与层次

表 1 - 1　不同抽象层次上的设计对象

层次名	行为领域描述	结构领域描述	物理领域描述
系统级	性能规定 流图 算法	处理器 控制器 存储器 总线	机柜 多芯片模块 底板 芯片
微体系结构级/ 寄存器传输级	寄存器传输级方程	ALU 多路转换器 乘法器 寄存器 存储器	芯片 布图 模块布图
逻辑级	布尔方程 时序状态	门 触发器	模块 单元
电路级	微分方程 函数	晶体管 连线	晶体管布局 连线 结点

硬件描述语言是描述数字系统的有效工具。它已成功地应用于模拟验证、设计综合及布局、布线等系统。运用硬件描述语言可以直观、准确地描述数字系统，为人—机之间提供一种良好的界面。现在已提出的描述语言有几十种，甚至上百种。它们从不同的角度描述数字系统，并且应用在数字系统自动设计的不同阶段。

首先是应用在模拟中。模拟技术是对设计的合理性和正确性的验证工具，它具有重要的意义。设计过程中各个层次都需要进行模拟。在进行系统性能评价时，曾使用 GPSS、SIMSCRIPT 和 SIMULA 等通用模拟语言来研究系统的行为，并进行系统性能的评价。在各种模拟语言中，非常重要的模拟语言是寄存器传输级语言，如 DDL、AHPL、CDL、Verilog 和 ISPS 等，它们以指令操作原理为基础，通过描述寄存器及其相互间的数据传输、数据操作的相互关系来描述数字系统，方便而且直观，与硬件的动作直接相对应。寄存器传输级描述是从逻辑级进一步抽象而得到的，其主要基本单元有：

(1) 存储器和寄存器等基本记忆单元，由触发器阵列组成，它们可能是通用寄存器堆，也可能是专用的程序计数器、地址计数器和堆栈指针等。在寄存器级不考虑它们的具体实现技术(如器件的选择等)。

(2) 完成特定操作功能的功能部件，是由组合逻辑或时序逻辑电路网络构成的，并且经常是将若干功能结合成集中的部件，如算术逻辑部件 ALU 等。

(3) 用于构成各部件之间连接的连接部件，由引线、总线及相应的控制电路构成，如通用总线(多输入、多输出)、多路转换器(多输入、单输出)及多路分配器(单输入、多输出)等。

行为级和寄存器传输级的描述可用于功能模拟和高级综合的输入。通过模拟可以验证系统的行为功能和数据流，而综合的结果可以生成硬件逻辑的描述，其有效性取决于设计和描述语言模型的精度。

在行为级和寄存器传输级(RTL)的描述中，可以将数字系统分为数据和控制两个部分(如图 1-3 所示)。数据部分是表示数据的寄存、存储和处理部件及其相互间的连接，它描述出系统中数据在各个部件间的流动结构，也称为数据流部分。控制部分为控制数据处理的时序状态机。控制部分可用该时序机的结构表示，也可用数据部分的操作次序表示，因此可称为控制流部分。由此，在自动逻辑综合中常将整个数字系统分为数据处理和控制信号生成两部分分别进行处理，这是现代 ASIC 设计的常用方式之一。行为级和寄存器传输级描述语言也可以用于微程序模拟和微码的自动生成。现在微程序设计已不仅仅是实现控制逻辑的手段，它对计算机系统结构也产生了重大的影响，且成为软件固化的方法。固件设计在计算机研制工作量中所占的比重日益增大，固件设计自动化涉及到微程序描述语言、微程序模拟、控制存储器字位分配、各种文件自动生成等问题。

图 1-3 数据与控制部分示意图

逻辑设计完成后，设计人员可以用门级逻辑模拟程序验证系统逻辑设计的正确性。现

代的逻辑模拟器的模型可用若干种不同的时间延迟值和信号状态值对电路进行模拟，完成更精确而严格的检查，可以发现逻辑电路中存在的冒险、竞争等工作不可靠的因素。

再下一层次是电路模拟或称为电路分析。这是在已知电路结构及有关参数和输入波形的情况下，计算出电路中各有关的电流和电压，画出波形，统计出频率特性等，从而优化电路中某些参数。与上述各层次不同，电路模拟主要是连续量计算。

计算机设计自动化成熟最早并且应用最广的是工程设计领域。它把实现系统某种行为功能的具体硬件组装到插件上和机柜里，这就涉及硬件划分、布局和布线等许多技术问题，许多工作是对应于不同的组装层次进行的，如将电路分封到不同的器件里，再将器件分封到不同的插件板上等。工程设计是计算机设计中工作量最大、最繁杂、最容易出现错误的重复性工作，它是 CAD 技术最早的应用领域。因此，目前工程设计几乎全部实现了自动化。

自动测试是计算机设计最为关键的环节之一，主要是借助计算机寻找逻辑电路中存在的故障。为实现自动测试，一个重要的问题是编出测试码。用计算机产生测试码称为测试码自动生成。目前，测试码生成的算法主要有单通路敏化法、D 算法、九值算法、布尔差分法、主通路敏化法、星算法等。已有的算法在最坏的情况下，时间复杂度是指数级的，即测试生成问题是 NP 完全问题。目前，完全型测试只能适用于中等规模的电路，还不能适应 LSI 和 VLSI 发展的需要。解决这一问题的重要途径是可测试性设计，即电路设计的出发点之一，就是要充分地考虑到它的测试问题。

1.2　高级综合的基本问题

高级综合（High - Level Sythesis，HLS）通常是指从目标系统的行为、算法描述到寄存器传输级结构描述的自动设计过程。它从对设计对象的行为描述开始，在给定的目标速度、实现造价乃至可测试性等约束条件下，生成满足相应条件并能实现目标系统行为的结构描述。行为描述是表征系统输入与输出之间的对应关系或映射，而结构描述表示组成系统的基本元件及其相互联接。通常对同一行为描述可能存在多种实现结构，即行为描述和实现结果是一对多的映射关系。综合的基本任务是要在同时满足各种约束条件的情况下，寻求一种性能优良的实现结构。通常这是由一系列设计工具集成的高级综合系统来完成的一个复杂的设计过程。

下面以移位加乘法器为例阐明高级综合的全过程。其 VHDL 语言（详见第二章）的行为级描述如下：

```
entity MULT is
  port( A_PORT,B_PORT : in Bit_vector( 3 downto 0);
                M_OUT:out Bit_vector( 7 downto 0);
                CLK:in CLOCK;
                START:in Bit;
```

```
                DONE:out Bit
  );
end MULT;
architecture SHIFT_MULT of MULT is
begin
   process
      variable A,B,M : Bit_vector;
      variable COUNT : Integer;
   begin
      wait until(START = 1);
      A:= A_PORT;
      COUNT:=0;
      B:= B_PORT;
      DONE <='0';
      M:= B"0000";
      while( COUNT <4) loop
         if(A(0) ='1') then
            M:= M + B;
         end if;
         A:= SHR( A,M[0]);
         M:= SHR( M,'0');
         COUNT:= COUNT +1;
      end loop;
      M_OUT <= M & A;
      DONE <='1';
   end process;
end SHIFT_MULT;
```

其中将4位乘法器定义为硬件实体MULT,它具有乘数输入端口A_PORT和被乘数输入端口B_PORT,启动信号START和同步时钟CLK,输出乘积端口M_OUT及完成乘法的回答信号DONE,如图1-4(a)所示。移位加相乘算法用进程语句process说明。

综合过程的第一步将输入的VHDL语言行为描述经语法分析器完成词法检查、语法分析,并生成中间格式(BIF)的描述。可暂定其为数据控制流通图(Control/DataFlow Graphs,CDFG),如图1-4(b)所示。须知描述中进程语句内是顺序执行的。CDFG中的CFG表征行为描述的顺序、条件分支及循环结构,而DFG表征赋值语句描述的操作,如图1-4(c)所示。CFG的每个结点有一数据流基本块,表示该结点执行的操作,如图1-4(c)所示。本例各数据块中赋值语句的操作数互不相关,它们可并发执行。因此该图中DFG清晰地表示出了VHDL顺序的进程描述中存在的并发性。

众所周知,上述乘法器的抽象描述可用带有数据通道的有限状态机(Finite State Machine with a Datapath,FSMD)的目标结构模型实现:将CDFG调度(scheduling)到4个状态

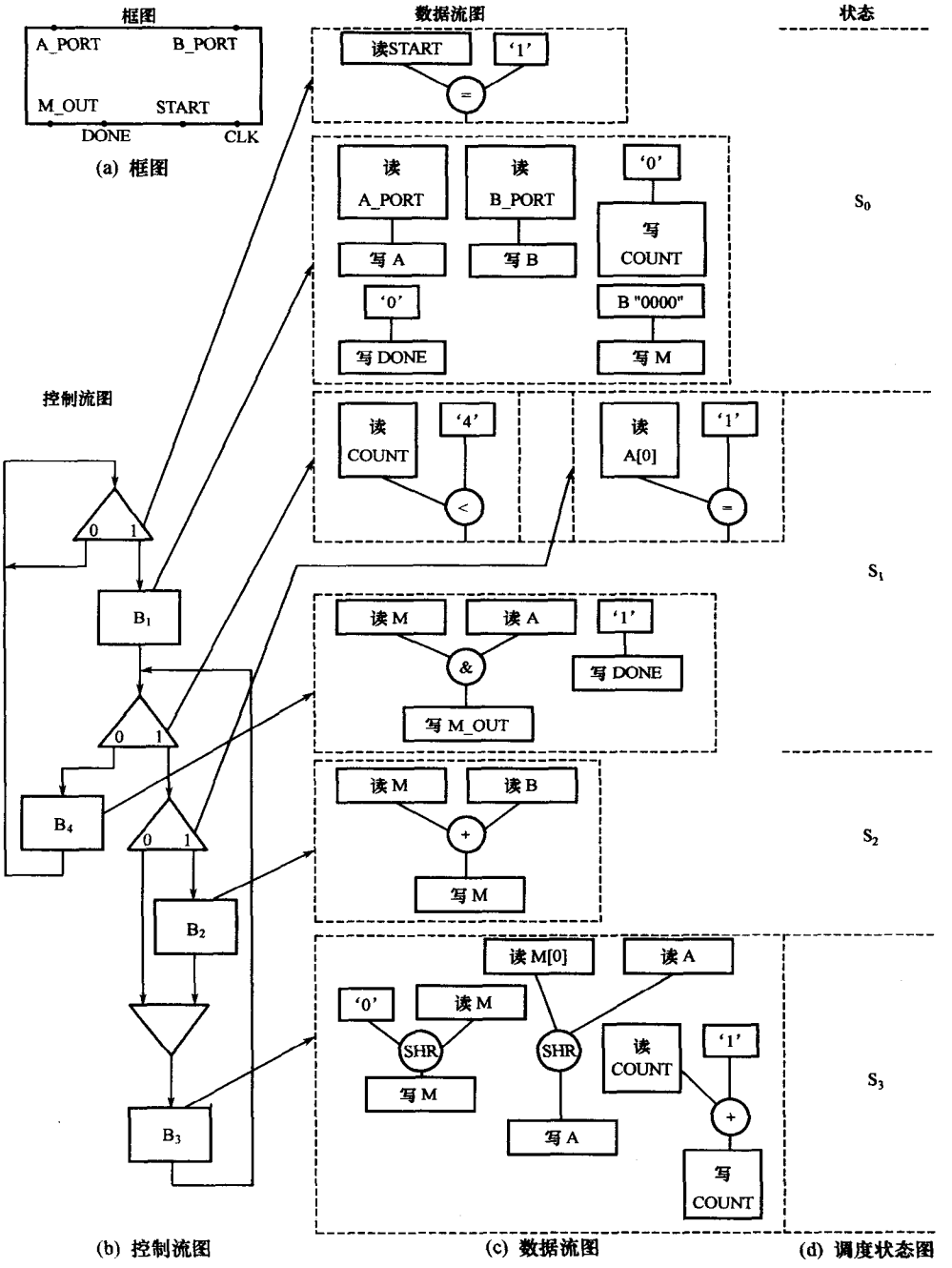


图 1－4　移位加乘法器的 CDFG

(节拍、控制步)S_0、S_1、S_2 和 S_3。在状态 S_0,当 START = 1 时,输入两个相乘的数 A 和 B,内部变量清"0"。B1 中各语句无数据相关,可调度到同一状态 S_0。状态 S_1 开始执行移位加乘法的循环,S_2 将被乘数 B 累加到部分积中(若 A(0) = 1)。S_3 将部分积右移一位,计数器加 1。在状态 S_1,若计数器内容为 4,则乘法结束,发出 DONE <= 1,端口 M_OUT 送出乘积。将 CDFG 调度后的状态标识到各结点上,于是在抽象的行为和设计状态之间建立起联系(如图 1-4(d)所示)。这实际上是完成了微操作的时间安排。

第二步 7 是选择寄存器传输级(Register Transfer Level,RTL)的寄存器单元及功能单元,并实现它们之间的连接,称为分配(Allocation),如图 1-5(a)所示。其中表示出 I/O 端口、4 个寄存器 A、B、Count 和 Mult、加法器 Adder、比较器 Compar、两个移位器 $Shift_1$ 和 $Shift_2$。再根据 CDFG 中的变量和调度后的操作将它们连接起来,保证数据在数据通道中正确传输。为消除输入通道上的多义性,增设多路器 $Mux_1 \sim Mux_4$(图 1-5(b)所示)。

最后根据调度和分配的结果归纳出控制器的设计,如图 1-5(c) 中的表格,每列对应一个状态及相关的条件;每行(除最后一行) 对应于数据通道的功能单元;行与列交叉处为控制信号;最后一行为相应状态的后继状态,整个表格实际上是实现乘法操作的有限状态机 FSM 的状态转换表。

更进一步的工作是将上述所获得的寄存器传输级单元及其互相连接的描述映射到逻辑级的描述。这要针对特定的单元库进行,又称为工艺映射或硬件约束(Binding Links)。它将上述不断具体化、精细化和完善化的过程增加的信息与行为描述联系起来。同时也完成逻辑级的优化,乃至版图设计。

综上所述,高级综合完成了从抽象行为描述到相应的功能单元、寄存器单元及其互相连接乃至实现过程的控制单元的自动生成。其主要任务可归纳为:

(1) 设计的描述、编译和中间格式的自动生成;

(2) 划分(Partition),将规模较大的系统行为描述或设计结构在给定的条件下分割成若干较小的子描述或子结构;

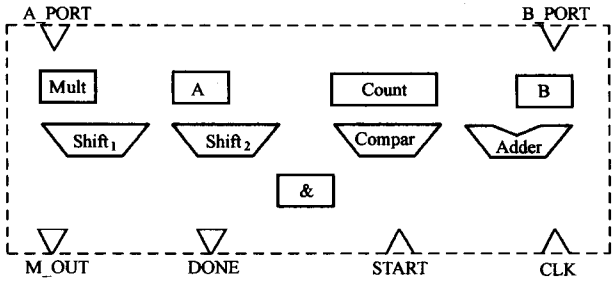
(3) 调度,将各变量赋值和操作完成的时间划分成若干间隔,或称为控制步骤;

(4) 分配(Allocation),将上述变量赋值和各种基本操作划分到寄存器(堆)、功能部件及其互联通道分别执行;

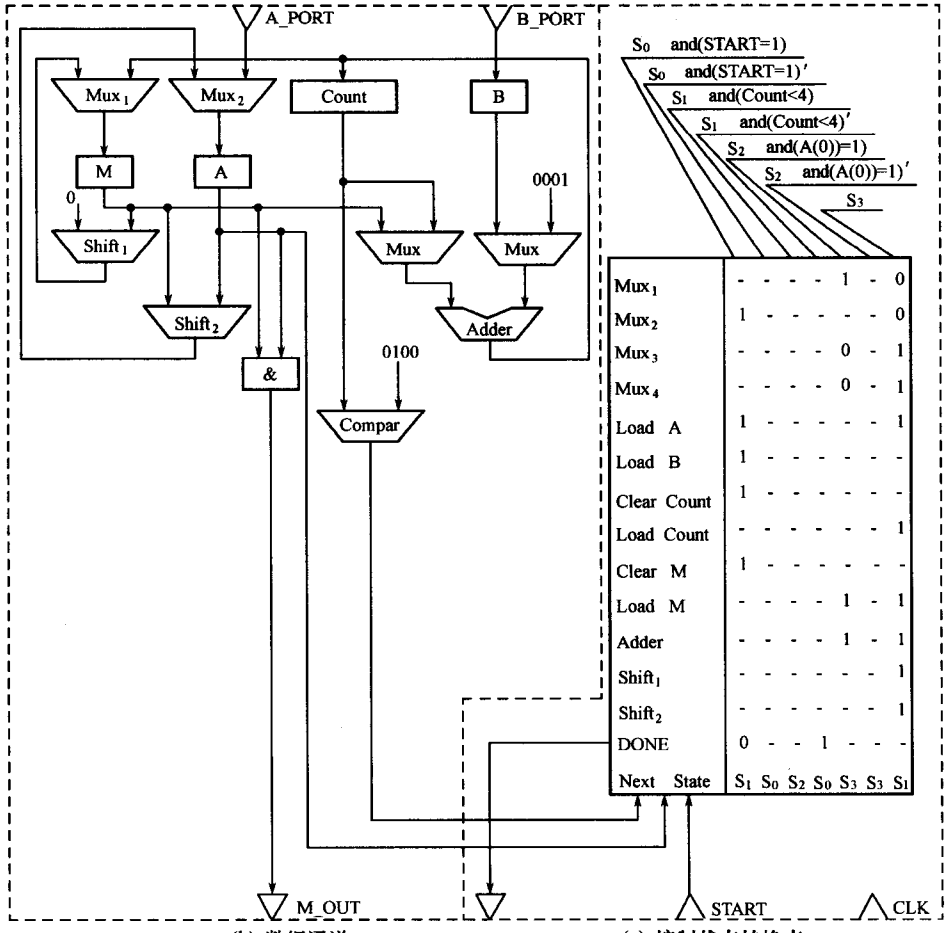
(5) 工艺映射(或称装配),它将高级综合获得的寄存器传输级或逻辑结构级的网表,根据给定的单元库,映射为库中相应元件及其互相连接;

(6) 数据库和集成环境的建立。

成功的高级综合需要有一个较好的框架支持各层次上的设计,包括数据库、图形及集成环境。要求它能将各层次上的不同设计工具协调一致,并能在由顶向下的各层次上正向传递信息和反馈信息。同时也能为设计的确认、生产、测试及管理提供必要的信息。特别是需要提供交互式设计方式。



(a) 初始单元设置



(b) 数据通道　　(c) 控制状态转换表

图 1－5　移位乘法器的硬件组成

1.3 综合目标的体系结构模型

1.3.1 设计风格与目标的体系结构

设计过程中,首先要选择设计风格,然后确定目标的体系结构。这里使用"设计风格"一词是指设计的重要技术方案,如优先中断、指令缓冲器、数据高速缓冲器、面向总线的数据通道、串行 I/O 接口、直接内存访问等。目标的体系结构定义为具体元件及其参数,以及它们之间的连接。例如,一个处理器的体系结构包括寄存器堆(Register File,RF)中寄存器的数目、数据通道中总线的数目、流水线的级数、状态位的数目、能够发生转移分支的数目等。

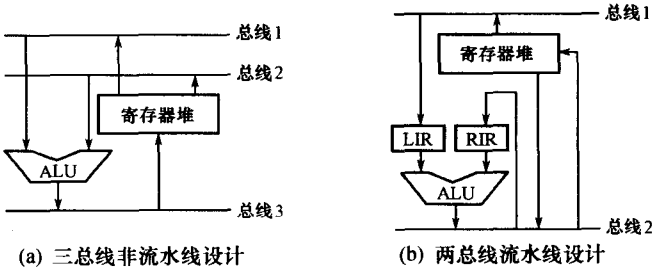
每种综合算法都假定一个确定的目标体系结构。目标体系结构越接近实际,综合结果的质量越高,但算法就可能越复杂。显然,设计质量和综合算法复杂性之间应该进行合理的折衷。例如图 1-6(a)中的三总线结构,在一个时钟周期内,可从寄存器中同时取出两个操作数,在 ALU 中完成运算后,结果存入寄存器堆。因此,如果变量 a、b、c、d、X 和 Y 存放在寄存器堆的任意寄存器中,而且每个周期是 100 ns,则完成下述两个二进制运算

X <= a + b(100 ns)

Y <= c - d(100 ns)

总共需要两个时钟周期(200 ns)。

如图 1-6(a)所示数据通道模型很简单,但是总线在组件芯片上需占用较多的硅片面积;特别是寄存器堆和 ALU 不能同时工作:当通过 ALU 传送信息时,就不能通过寄存器堆传送信息;其逆亦然,因此三总线不是最好的方案。



(a) 三总线非流水线设计　　(b) 两总线流水线设计

图 1-6 两种数据通道的比较

通过在 ALU 的输入端设置两个寄存器(LIR 和 RIR),并将时钟周期平分成两个半周期,可极大地提高 ALU 的使用效率。图 1-6(b)示出这种体系结构的数据通道模型,上述运算可改为

LIR <= a;　　RIR <= b;　　(50 ns)