国家自然科学基金资助课题

并行计算机程序设计导论

全惠云　高汉平　康立山　陈毓屏　著

武汉大学出版社

CAD 五周通

赵长利　张强华　主编

陕西电子杂志社出版发行

陕西杨陵科技印刷厂印刷

787×1092　1/16 开本

1995 年 11 月第 1 版　　1996 年 3 月第 2 次印刷

印数：1—8000 册　23.75 印张　570 千字

国内统一刊号：CN61—1224/TN

定价：23.00 元

武汉大学学术丛书
编委会

全惠云，1949 年出生，1976 年毕业于武汉大学数学系计算数学专业，现任湖南师范大学理学院副教授。多年从事计算方法和计算机科学教学和研究，主要研究方向是计算机并行算法。独立或与人合作在国内外有关学术杂志上发表过论文十多篇。与康立山教授合作著的《数值解高维偏微分方程的分裂法》一书获 1990 年华东地区优秀科技著作二等奖。参加过多项国家自然科学基金课题研究。主持过国防科工委、湖南省教委有关课题的研究。当前正主持武汉大学软件工程国家重点实验室一个开放课题的研究工作。

高汉平，湖北黄冈市人，1953 年出生。1979 年毕业于华师黄冈分院数学系，现任黄冈师专计算机系副教授，计算机软件教研室主任。

1985 年至 1989 年其间曾在国家科委武汉计算中心、武汉大学计算机科学系软件助教班进修，1994 年至 1995 年作为访问学者在武汉大学软件工程国家重点实验室工作、进修。

长期从事计算机专业的软件课程的教学工作。先后参加了《优化学生知识结构探索推行 CBE 方法》和《并行计算》的课题研究；发表科研论文十多篇；合作编著出版了《计算机语言》和《程序设计与软件工程》，现从事并行计算和并行程序设计方面的研究。

康立山，1956 年毕业于武汉大学数学系，现任武汉大学计算机科学学院教授、博士生导师。近二十多年来一直从事分布式并行计算研究。其研究成果“异步并行算法与区域裂解法”等三次获国家教委科技进步一等奖和国家自然科学奖。

1980—1981 年赴法国巴黎大学（南）进修；1985—1993 年间三次赴美国讲学；1990 年去日本讲学。1993—1994 年受聘为澳大利亚国立大学访问研究员。此外还担任国际杂志：“Parallel Algorithms and Applications”与“Neural, Parallel and Scientific Computations”编委。1995 年 10 月在武汉大学成功地主持召开了“并行算法国际会议”。

陈毓屏，副教授。1967 年毕业于武汉大学。1980 年开始参加分布式并行计算系统的研制工作。一直承担国家自然科学基金与 863 智能计算机系统基金等项目的研究。其成果“异步并行算法”等两度获国家教委科技进步一等奖，1993 年获国家自然科学奖。研究成果“演化计算及其并行处理”1996 年获国家教委科技进步（甲类）一等奖。1993—1994 年受聘为澳大利亚国立大学访问研究员；1995 年 6—8 月为美国德克萨斯大学访问学者。

前　言

二十多年来，随着并行计算机和向量计算机的出现和发展，并行计算的研究也迅速崛起，使得一批具有重大挑战意义的科学与工程计算问题的解决成为可能。在并行计算机上进行并行计算，传统的串行程序设计已力不从心，它极大地制约着并行系统效率的发挥。因此，研究和发展并行程序设计就显得尤为重要。

人们都知道著名计算机科学家沃思关于程序设计曾经描述过一个公式：

程序＝数据结构＋算法

显然，这里指的是串行程序，那么并行程序呢？并行程序设计与串行程序设计在编程的语言和环境上存在着哪些重大差异呢？这正是本书所要探讨的主要内容。

不言而喻，并行程序设计需要并行编程语言，因而对并行语言的研究一直是一个极有吸引力的热门课题，形形色色的并行语言层出不穷，例如 CM Fortran、C* 和 *Lisp 等。CM Fortran 及其他一些扩充的 Fortran 语言统称为 HPF(High Performance Fortran)，是目前最受青睐的一类并行语言。HPF 能够提供比标准 Fortran 和 Fortran 90 更多的信息。在此之前，很多研究机构已进行了广泛研究，其中以美国 Rice 大学的 Fortran D 语言和奥地利 Vienna 大学的 Vienna Fortran 语言影响最大。1991 年底，Kennedy 和 Geoffery Fox 建议成立一个非官方的组织，来对这种语言进行定义和标准化。于是，一个由工业界和学术界联合组成的机构——高性能 Fortran 研究会 HPFF(High Performance Fortran Forum)宣告成立。经过两年多的努力，终于在 1993 年推出了一种能够满足上述要求的新的 Fortran 语言标准——高性能 Fortran(HPF)。HPF 的目标是以 Fortran 90 为基础的并行描述语言，其开发的目的是：

(一)支持数据并行程序设计；

(二)最大限度发挥多指令流多数据流(MIMD)及单指令流多数据流(SIMD)计算机的处理能力；

(三)定义可以适应各种体系结构计算机的标准语言。

HPF 对 Fortran 90 的主要扩充有：(1)数据分布特性。HPF 提供了一组数据分布指令用以规划数据在系统中各存储体中的合理分配，在负载均衡和提高数据访问局部性之间达到合理折衷。ALIGN 指令说明数组间的相对存储位置关系，DISTRIBULE 指令则说明一个数组在抽象处理机的存储器中应如何进行分配。(2)并行语句。为了显式表达式并行计算，HPF 提供了一个新的语句(FORALL)和伪指令(INDEPENDENT)，FORALL 实现了对一个数组区域的赋值，其含义类似于 Fortran 90 中的数组赋值语句，但描述能力更强。INDEPENDENT 指令则指出一个特定代码段中的语句之间不存在任何顺序上的依赖关系，该指令只为编译优化(并行执行)提供信息。其它扩充领域还有：内部函数和 HPF 库，局部过程，并行 I/O 语句，顺序与存储联系等。

在HPF的定义中还包括了一个HPF子集，以尽早实现HPF编译器，作为到HPF全集的过渡。目前世界上的主要计算机厂商和科研机构正积极开展HPF编译系统的研制，在有关国际学术杂志和会议论文集中也有一些介绍，相信HPF会有越来越广阔的前程。

有人得出结论，并行算法设计会变得越来越容易，但是，除非借助良好的编程工具，否则并行算法的实现将是一件非常困难的事。因此，研制与开发并行程序环境是人们迎接并行处理技术挑战所面临的重大课题。目前，受到推崇的有可移植的异构编程环境PVM和Linda以及CM信息传递库CMMD(Connection Machine Message)。需要指出的是，一般来说，不同的并行机系统提供的并行环境不尽相同。为了统一互不兼容的用户界面，1992年由欧美40个主要研究组织联合成立了MPI(Message Passing Interface)委员会，负责制定信息传递界面的新标准，支持最佳的可移植平台。并于1994年发布了MPI的定义与标准化版本MPI1，其目标是要开发一个广泛用于编写消息传递程序的标准，要求用户界面实用、可移植、高效、灵活，能广泛用于各类并行机，特别是分布式计算机。MPI吸取了近年来涌现出的一批可移植的信息传递环境的优点和经验，同时从句法与语法两方面确定核心库函数，使之能适用更多的并行机。同时，相应的标准MPI2平台正在研制中，极有可能形成一个国际通用信息标准平台，因而MPI目前备受青睐。

本书试图从以上两个方面去阐述程序设计中由“串行思维”到“并行思维”的转换。

本书是在作者数年来从事并行算法研究和并行程序设计的基础上，结合国内外并行计算研究的最新动态撰写的，其中大量素材取之于作者1993年在澳大利亚国立大学讲学期间，在CM-5并行计算机系统上进行的大量数值实验所获得的一些经验和重要结果。

感谢潘正君博士和陈炬桦博士，他们分别撰写了PVM和Linda这两章中的部分章节，从而丰富了本书的内容。感谢武汉大学软件工程国家重点实验室的领导和同行们给予的有力支持。感谢澳大利亚国立大学I. Macleod博士对我们在CM-5计算机上进行数值实验时给予的帮助。感谢武汉大学出版社金丽莉同志在本书出版过程中给予的多方帮助。

由于高性能计算机科学的发展极为迅速，本书只能起一个抛砖引玉的作用。由于我们的学识水平有限，书中不当之处，恳请同行批评指正。

作 者

1996年8月于武昌珞珈山

目　录

第一部分　并行计算机和并行算法

第一章　并行计算机概述 …… 3
§1.1　并行处理系统 …… 3
§1.1.1　从串行机到并行机 …… 3
§1.1.2　计算机与算法的分类 …… 4
§1.1.3　并行处理机的几种形式 …… 5
§1.1.4　程序语言 …… 7
§1.1.5　性能测量 …… 7
§1.2　多道处理机系统 …… 8
§1.2.1　互连 …… 8
§1.2.2　共享存储器系统 …… 9
§1.2.3　局部存储器系统 …… 11
§1.2.4　对局部存储系统的共享 …… 13
§1.2.5　处理机与进程 …… 14
§1.3　度量程序操作 …… 14
§1.3.1　粒度 …… 14
§1.3.2　加速与效率 …… 14
§1.3.3　Amdahl 法则 …… 15
§1.3.4　负载平衡与吞吐量 …… 15

第二章　并行算法概述 …… 16
§2.1　并行算法发展的几个阶段 …… 16
§2.2　同步并行算法 …… 19
§2.3　异步并行算法 …… 23

第二部分　并行编程语言

第三章　CM Fortran 概述 …… 27
§3.1　CM Fortran 的模式 …… 28
§3.2　CM Fortran 的结构与特点 …… 29
§3.3　输入输出初步 …… 33
§3.3.1　输入语句(READ 语句) …… 33

§3.3.2 输出语句(PRINT 语句和 WRITE 语句) …… 35
§3.4 格式语句…… 36

第四章 CM Fortran 控制结构…… 38
§4.1 条件结构…… 38
§4.2 CASE 结构…… 45
§4.2.1 CASE 结构的一般形式…… 45
§4.2.2 CASE 结构的控制执行…… 46
§4.2.3 CASE 结构的标识符…… 47
§4.3 循环结构…… 48
§4.3.1 DO 循环结构 …… 48
§4.3.2 DO TIMES 循环结构 …… 50
§4.3.3 DO WHILE 循环结构 …… 51
§4.3.4 EXIT 语句与 CYCLE 语句 …… 52
§4.3.5 循环结构的嵌套…… 54
§4.3.6 隐含 DO 循环…… 57

第五章 CM Fortran 数组与数据处理…… 59
§5.1 数组的定义和有关说明…… 59
§5.1.1 数组的定义和数组说明符…… 59
§5.1.2 数组说明语句…… 60
§5.1.3 数组的下标与存储次序…… 62
§5.1.4 数组段(部分数组)…… 64
§5.2 数组的赋值、运算和输入/输出…… 67
§5.3 不同形式的数组说明…… 72
§5.4 数组的屏蔽…… 76
§5.5 数组元素的分配语句 FORALL …… 81
§5.6 动态分配…… 83

第六章 CM Fortran 数组变换…… 88
§6.1 数据移动函数…… 88
§6.1.1 循环移动函数 CSHIFT …… 88
§6.1.2 截止移位 EOSHIFT 函数…… 90
§6.1.3 矩阵的转置函数…… 92
§6.2 数组的归约函数…… 93
§6.2.1 求数组中最大元素的值函数…… 94
§6.2.2 数组的乘积…… 95
§6.2.3 求和函数…… 96
§6.2.4 计数函数…… 97

§6.2.5 ALL 和 ANY 函数 …… 97
§6.3 数组的构造函数 …… 99
§6.3.1 对角线构造数组函数 …… 99
§6.3.2 数组归并构造函数 …… 100
§6.3.3 数组的压缩与扩散函数 …… 100
§6.3.4 复制函数和扩展函数 …… 102
§6.3.5 重新整形函数 …… 104
§6.4 向量点积和矩阵的乘法 …… 106
§6.4.1 向量点积 DOTPRODUCT …… 106
§6.4.2 矩阵的乘法 MATMUL …… 107
§6.5 数组应用实例 …… 108

第三部分　并行程序通信

第七章 CMMD 概述 …… 119
§7.1 程序模型 …… 119
§7.2 通信协议 …… 119
§7.3 CMMD 的输入输出 …… 121
§7.4 CM-5 的体系结构 …… 121
§7.5 一个简单的 CMMD 程序 …… 122

第八章 CMMD 同步通信函数 …… 124
§8.1 缓冲区和数组 …… 124
§8.2 发送消息函数 …… 124
§8.3 接收信息函数 …… 126
§8.4 同时发送和接收函数 …… 127
§8.5 两个节点之间的信息交换函数 …… 128
§8.6 节点信息函数(辅助函数) …… 129
§8.7 信息检测函数 …… 130
§8.8 信息存取器函数 …… 130

第九章 CMMD 异步通信函数 …… 132
§9.1 异步发送函数 …… 132
§9.2 异步接收函数 …… 133
§9.3 非块化发送函数 …… 134
§9.4 异步检测函数 …… 135
§9.5 MCB 存取器函数 …… 135
§9.6 释放信息控制块子程序 …… 136
§9.7 等待异步信息子程序 …… 136
§9.8 节点广播函数 …… 137

第十章 CMMD 应用实例 …… 138
§10.1 例题及其算法 …… 138
§10.2 程序及其说明 …… 140
附录 10.1 CM Fortran 源程序 …… 141
附录 10.2 CM Fortran 源程序数值结果的图形显示 …… 146
附录 10.3 安装在国立澳大利亚大学的 CM-5 系统 …… 148

第四部分 并行计算机编程环境与分布式程序设计

第十一章 PVM …… 151
§11.1 PVM 概述 …… 151
§11.2 启动与配置 PVM …… 152
§11.3 编写 PVM 应用程序 …… 155
§11.3.1 C 语言编程示例 …… 155
§11.3.2 Fortran 语言编程示例 …… 159
§11.3.3 编写应用程序应该注意的几个问题 …… 162
§11.3.4 编译和运行 PVM 应用程序 …… 165
§11.3.5 程序调试 …… 165
§11.4 PVM 库函数使用指南 …… 167
§11.4.1 进程控制类函数 …… 167
§11.4.2 信息类函数 …… 170
§11.4.3 动态配置类函数 …… 174
§11.4.4 信号函数 …… 175
§11.4.5 错误信息处理函数 …… 177
§11.4.6 信息传递类函数 …… 177
§11.4.7 动态进程组类函数 …… 189
§11.5 PVM 应用实例 …… 193

第十二章 Linda …… 199
§12.1 C-Linda …… 199
§12.1.1 Tuple 空间的数据结构 …… 199
§12.1.2 C-linda 对 Tuple 空间的存取操作 …… 200
§12.1.3 Tuple 配备规则 …… 201
§12.1.4 C-Linda 的程序结构、编译、运行 …… 203
§12.2 C-Linda 应用实例 …… 203
附录 12.1 串行程序 …… 207
附录 12.2 同步并行程序 …… 209
附录 12.3 异步并行程序 …… 213

参考文献 …… 218

第一部分　并行计算机和并行算法

第一章　并行计算机概述
第二章　并行算法概述

概　要

本部分介绍作为并行程序设计基础的并行计算机和并行算法概况。通过这部分的介绍，读者可了解到并行计算机和并行算法发展动向，以及编程的 SIMD 数据并行模式和 MIMD 信息传递模式等，认识并行设计的必要性。

第一章 并行计算机概述

所谓并行处理是指计算机同时处理多条指令、多个数据或多个任务。要实现对一个问题的并行处理光有可并行处理的硬件结构是不够的，还必须有支持并行处理的软件和处理问题的并行算法。现代并行处理技术包含并行的系统结构、支持并行处理的软件和实现并行处理的算法等方面的内容。本章仅就并行处理的系统结构作一概要的叙述。

§1.1 并行处理系统

并行处理系统是由 n 台处理机或 n 台等效的处理机(器)组成，受统一的操作系统控制，解算同一问题，系统硬件的价格不能大于单处理机(器)硬件价格的 n 倍，系统的运算速度要能接近单处理机(器)速度的 n 倍。并行处理系统是提高处理速度和解决大规模问题的重要因素，展现出多处理机组成更大的计算机系统以及由超大规模集成电路(VLSI)组成计算机的前景，近二十年对并行处理技术的研究一直是计算机科学领域内的重大课题之一。并行处理系统是一门综合性的学科，它涉及并行的系统结构、并行算法、并行程序语言和并行操作系统等。下面仅对并行计算机发展、分类、语言等作一粗略描述，以便我们对并行处理系统有一个初步的了解。

§1.1.1 从串行机到并行机

众所周知，第一代计算机全部结构是按串行方式工作的，它遵循 Burks 等人于 1946 年设计的存储程序计算机的基本概念，通常称为 Von Neumann 结构。串行计算机的主要特点是计算机每一操作必须按顺序执行，或一次执行一个操作(如存储器的取数或存数，运算或逻辑操作，输入或输出操作)。这些早期计算机能存储并完成串位计算，应用于位串行。

1953 年 IBM 公司完成了具有并行运算的第一台商业计算机——IBM701 机，从而拉开了并行计算机的序幕。在并行计算机的发展过程中，经历了两个水平层次，这两个层次都要试图增强计算特殊类别的操作运算，早期努力的中心点是关于发展高速运行的巨型计算机，以解决大量科技上的问题，例如精确地预报天气，海洋和天体物理的模拟，遥测地球资源数据处理，地震的探测等，均需要使用超高速的巨型并行处理计算机。70 年代中期，多数通用巨型计算机是 Cray-1 型的向量处理机，其速度可达 130Mflops(每秒钟为百万兆浮点操作运算)。80 年代后期，Cray Y-MP 有 8 台处理机和 Cray-2 有 4 台处理机的运行速度可达每秒 25 亿次，这些巨型计算机有限地展示并行性，但它仅仅在少量大功率的计算机中并行运行。

并行计算机发展的第二个要素是围绕生产超微型计算机的设想，实现多机联接的超级计算机，以此来降低费用。这就是使用向量处理机或者包括许多并行处理机(请参阅多道处

理机系统)或两者都使用。这种计算机中的并行处理不仅出现于各种超级计算机中,而且日益遍及各种工作站中的并行处理(如Sun公司生产的Sun工作站)。

并行计算机的主要功能结构和特点可概括为:

1)流水线结构　其特点相当于工业生产中的装配线技术,改进运算和控制部件的功能。

2)功能结构　为执行不同的功能提供几个独立部件,例如逻辑部件,加法或乘法部件,并允许这些部件同时处理各种数据。

3)阵列结构　在统一控制下,提供一个很多相同的处理部件组成的阵列,这些部件同时执行相同的操作,但是与其相对应的专用存储器中的数据是不同的。

4)多道处理结构　提供n台处理机,但各台处理机只执行其各自的指令,通常经过一个共同的存储器相互通信。

在众多的并行计算机中,某些特定的计算机可能兼有上述并行特点的部分或全部。例如:一个处理机阵列可以把流水线运算部件作为它的处理部件,同时在多部件计算机中,某一功能部件可以是一个处理机阵列。

§1.1.2　计算机与算法的分类

计算机按其指令流与数据流的执行方式可分为四类:

(1)SISD计算机　即单指令流单数据流计算机。这类机器就是常用的(冯·诺依曼型)计算机。为这类机器编写的程序都是根据一种"串行算法"编制的。

(2)SIMD计算机　即单指令流多数据流计算机。如阵列式计算机:ILLIAC Ⅳ,流水线计算机:国产757机与银河(YH-Ⅰ,STAR-100,CYBER 205,Cray-1,ICL-DAP,IBM 360/91等都属于SIMD。它们都要求使用一类"同步并行算法"。这类算法基于进行向量与矩阵运算时各分量的高度同步并行性,故这类计算机也称为向量机。自从1972年并行计算机系统ILLIAC Ⅳ与1974年STAR-100投入运行以来,同步并行算法已有较多的研究。特别是Cray-1与CYBER205等类型的计算机已批量生产,因此与之相适应的同步并行算法的发展也较快。我国在研制银河机的同时也开展了同步并行算法的研究,并取得了可喜的进展。

(3)MIMD计算机　即多指令流多数据流计算机。多处理机与多计算机就属于这种类型。如我国的银河-Ⅱ(YH-Ⅱ)多处理机系统、武汉大学的WuPP-80并行处理系统、Carnegie-Mellon大学的Cmmp与C*m系统、Loughborough大学的LuT_2与LuT_4等多计算机系统、California工学院的Hypercube、Rochester大学的Butterfly™等多处理机系统,还有IBM公司的ICAP系列、Wisconsin-Madison大学的CRYSTAL,以及多向量机(MSIMD或称Pipelined MIMD)HEP、Cray X-MP系列、Cray-2等等。这类机器的并行运算过程通常都具有高度的"自治性",即异步地并行运算。它要求一类异步并行算法的支持。MIMD计算机在科学计算方面的运用尚处在试验阶段,没有公认的计算机模型。因此异步并行算法的研究也处在试验阶段,但这类计算机有较高的性能价格比、灵活的系统可扩充性、良好的实时性、可靠性与容错性等优点。因此,作为新一代计算机的雏型,各式各样的MIMD试验系统越来越多,与之相适应的异步并行算法的研究成果也越来越多,越来越引人注目。

(4)MISD计算机　即多指令流单数据流计算机,这种机器很少用于科学计算。

综合上述，目前用于科学计算的计算机与算法可粗略地分类于下：

科学计算机
- 串行计算机：SISD（串行算法）
- 并行计算机
  - SIMD（同步并行算法）
  - MIMD（异步并行算法）

我们强调上面的分类是粗略的，这样对研究算法设计有好处，易于抓主要矛盾，尤其对初学者。但实用上，由于应用问题千差万别，计算机结构五花八门，并行算法尚未完善，故现在的并行计算机上实际运行的是"串、并"混合算法。正因为如此，并行计算机的性能并未最有效地发挥。

由于SIMD计算机只在数据流上有并行性，而指令流仍然是串行的，故有人认为这类计算机算法（只有数据流并行）仍属串行算法，或称为向量化算法，用以区别具有二维并行性（数据流并行，指令流并行）的MIMD算法。从算法发展历史来看，人们都称SIMD算法为并行算法，只是今天MIMD计算机大量涌现之后才出现了更细的划分。因此我们认为，作为粗略分类以区别一维并行性（数据流）与多维并行性，还是分别称为同步并行算法与异步并行算法为好。至于MIMD计算机和MIMD算法也有同步与异步之分。从其所含处理机之台数来说又有浅度并行（10～100台）、深度并行（100～10000台）与极度并行（10000台以上）之分；从进程的粒度来分有高级并行（数学模型级）、中级并行与低级并行（算术表达式级）之分；从存贮的方式上又有共享存贮与分布式存贮之分；从机器的连接与通信方式等方面又可分成许多种。所以，机器分类越细，算法分类也越细，分类过细反而不易抓住主要矛盾。

§1.1.3　并行处理机的几种形式

尽管并行处理的概念可以追溯到现代计算机的发明之前，但并行处理作为一门技术被深入地研究与普及应用却是近十多年的事情，其蓬勃发展更是近几年才出现的，而并行处理机要像传统冯·诺依曼结构那样被广泛而有效地应用则需要进一步研究和推广。

并行处理机有三个基本结构类型：流水线结构SIMD计算机、阵列结构SIMD计算机和多道处理结构MIMD计算机。

1. 流水线结构SIMD计算机

流水线结构是基于时间重叠原理，把每条指令分成若干个操作（如取指令、译码、取操作数、执行和写结果），每个操作分别由不同的部件执行，以此达到同时处理多条指令。对每个部件而言，每条指令中的相同操作像流水线一样被连续处理，从而实现时间重叠的并行计算。当流水线不断流时，实际的执行速度相当于流水执行部件一级通过的速度，即机器周期。

许多早期开发的并行处理计算机，就是流水线向量处理机。在50年代末，由英国曼彻斯特大学设计和制造的ATLAS计算机就是一种早期的使用流水线计算机。一条指令的执行被分为四段（指令提取、操作数位置计算、操作数提取和算术处理），该阶段可以制成流水线，于是在理想状况下，指令1的第4段，指令2的第3段，指令3的第2段，指令4的第1段能够被连续运行，要使用好这种设备，必须有一种"远见"，以便确定哪些指令可以联结而不致于影响程序的逻辑意图。

向量流水线是向量操作数和向量指令流水线的意图之简单延伸。向量运算之所以很适合流水线作业的原因是它们在其分量上形成一种可预见的正规的一列标量操作。作为一个简单的例子，设$a+b\Rightarrow c$，这个加法操作可以划分为三个阶段，每个阶段需要一个时钟循环

来执行。于是：

* 在第一个时钟循环时：　a_1 与 b_1 在加法第一阶段；
* 在第二个时钟循环时：　a_1 与 b_1 在加法第二阶段；
　　a_2 与 b_2 在加法第一阶段；
* 在第三个时钟循环时：　a_1 与 b_1 在加法第三阶段；
　　a_2 与 b_2 在加法第二阶段；
　　a_3 与 b_3 在加法第一阶段。

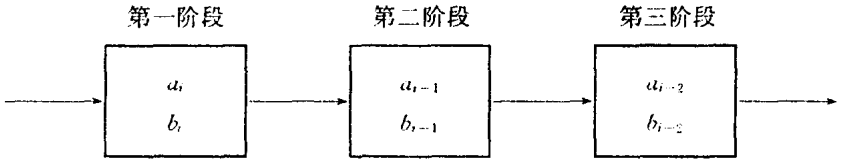

图 1-1　第 i 个时钟循环的流水线形态

这以后，所有的这三个阶段都连续地工作，直到进入 a 与 b 的最后一个分量。第 i 个时钟循环的示意图如图 1-1 所示，结果是：$a_{i-2}+b_{i-2}\Rightarrow c_{i-2}$ 加法完成。

2. 阵列结构处理机

一个阵列结构处理机(或数据并行处理机)是属于 SIMD 计算机。阵列结构是基于空间里的重复原理，将许多具有相同功能和处理能力的处理器按一定的拓扑结构互相连接而构成的。在这种结构中，各个处理部件在单指令流的控制下并行处理各自的数据流，因此特别适用于向量和矩阵处理。在阵列机中，每个处理器都要同等地担负多种运算功能，其复杂程度视需要而定，可以是简单的一位微处理器，也可以是功能强大的 32 位微处理器。阵列机的专用性很强，其性能与算法结构有着密切的联系，对于某些特定用途(如信号处理、图像处理和电路模拟等)而言，可以达到超高速运算。例如，由 16384 个一位处理单元按 128×128 阵列结构组成的大规模并行处理机(MPP)，对 8 位整数加法速度可达每秒 65 亿次；由 65536 个一位处理单元组成的 CM-2 的运算速度可达每秒 25 亿次，对特殊问题可达每秒 70 亿次。1995 年我国通过鉴定的曙光 1000 大规模并行计算机系统(MPP)，突破了大规模并行处理的关键技术，采用了多项 90 年代的国际最新技术。特别是它的蛀洞路由器芯片和并行优化编译器，在世界上堪称是最先进的技术。曙光 1000 由 36 个结点处理机组成，其中 32 个计算结点机，2 个服务结点机，2 个输入输出结点机，采用 6×6 的网格结构。曙光 1000 的峰值速度可达每秒 25 亿单精度浮点运算，求解线性方程组的速度可达每秒 11.2 亿次浮点运算。还有 ILLIAC Ⅳ，DAP 及 IBM GF-11 和 TF-11 等都属此类计算机。

思维公司推出的换代产品 CM-5，集 SIMD 和 MIMD 优点于一身，是目前数据并行计算机最成功的产品。

CM-5 体系中最重要的创新就是它的体系结构超越了 SIMD 和 MIMD 两者。既有 SIMD 编程的简易性，又有 MIMD 执行的灵活性。它提供了使机器的所有部分都恰到好处地保持协调同步的硬件和软件的结合。CM-5 的构造块是标准的 RISC 处理器另加上了一些并行处理部件，它们是一个特殊的存储控制器、一个通信接口，并且为浮点运算增加了 4 个 64 位的向量单元(总的峰值能大于 100Mflops)，两个称为控制网络和数据网络的通信网络