

第七届世界计算机体系结构 大会论文集(摘译)

北京工业大学计算机科学系
软件教研室

一九八一年十月

872083

译 者 的 话

此摘译中共有八篇论文，全部译自80年5月在法国召开的第七届计算机体系结构大会论文集(Conference Proceedings The 7th Annual Symposium on Computer Architecture)。把这几篇论文翻译出来的目的，就是看一看国外在这方面都在做些什么，了解计算机体系结构在计算机软件、硬件研究当中所起的作用及其研究动态。

计算机系统结构即计算机体系结构对我们来说是很新的东西，我们知道得很少，学习得很不够，因此译文当中难免会有错误，敬请读者批评指正。

北京工业大学计算机科学系
软件教研室

一九八一年十月

目 录

一、细胞计算机——一种实用的流水线影象处理器	1
二、软件子系统结构化的实现技术	15
三、高级语言计算机结构的回顾	33
四、关于容错 VLST 处理机系统检测和再组合原理	55
五、对使用位片模拟引教微程序控制结构的研究	72
六、便于翻译程序设计语言的一种结构	96
七、一种大规模并行处理机的结构	107
八、一种高级多语言的多处理机	120

细胞计算机——一种 实用的流水线影象处理器

Robert M. Loughhead and David L. McCubbrey

(密执安环境研究所)

摘 要

本文提出一种用于影象处理的新型体系结构。它由一批相同的、可编程序的串行处理层的流水线构成,称为细胞计算机。对细胞计算机与并行阵列系统进行了比较。细胞计算机通常具有低复杂性、高带宽以及更大的灵活性等优点。介绍了一个第一代系统和一些关于处理的例子。最后叙述了当前的发展方向。

I 引言

数字化影象处理正在许多不同的领域,诸如医学、通讯和自动化等方面得到日益广泛的应用,传统上使用通用(冯诺伊曼)数字计算机进行影象处理,然而这类计算机的主要问题在于:它的体系结构对于处理影象数据是不适宜的。本文叙述一种独特的体系结构,它由一个可编程序的处理层的流水线构成,而各层对于实时影象处理是最优的。与另一些专用的影象处理体系作了比较,比较表明了这种流水线体系在各种典型应用中的优点。最后,叙述了一个实际的系统并提出了当前的发展方向。

II 背景

数字计算机在影象处理方面的首批应用之一于1955年在Lincoln实验室出现。这一工作由Dinneen⁽⁸⁾和Selfridge⁽⁹⁾提出。工作的基础是自动的分块操作。计算机存储器的一部分用作分块空间,在其中进行自动运算。各分块空间与数字化影象共用一个公共的结构。一数字化影象中的每个象元(pixel)可以看作在一给定状态下的一个单元或“细胞”。如果在一数字化影象上定义一个邻域关系和一个象元变换函数,并在空间中平行地使用变换函数,即可将各个象元

状态间的一种结构转化为一种新的结构，它对应于经过修饰或变换的影象，如图 1 所示。分块处理的关键是寻找一系列通常彼此不同的基本邻域变换，它们可将一给定的初始结构变换为所需的结构。这样一系列变换形成一个影象处理程序或算法。

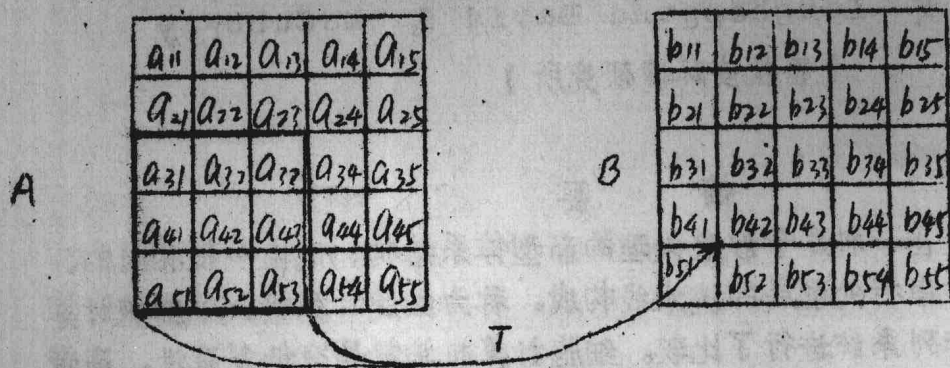


图 1 邻域变换

正如 Sternberg⁽¹⁾于 1978 年所论证的，使用 3×3 窗口的邻域算子（如图 1 中所强调指出的那样）对于处理有噪声干扰的影象是非常有效的。Codd⁽¹⁾在 1968 年曾经证明，一系列 3×3 窗口的算子能够等价于一个使用任意大小窗口的算子。图 2 给出了这种窗口扩展关系的一个例子*。图 2a 中指出了—个目标，它有已知的大小、形状和方向。在图 2b 中有这样五个目标被嵌入到“噪声”中。虽然用肉眼很难检出这些目标，但是由 36 个最近邻（ 3×3 窗口）变换所组成的一个序列却可以检出和增强它们，如图 2c 所示。这一算子序列等价于一个使用 12×12 窗口的单个算子。这是一个关于最近邻算子在影象处理中的应用的例子。有关这一方面的问题可参看 Preston 等⁽⁸⁾在 1979 年所作的综述。

不幸的是，通用（单指令单数据）处理机执行分块数字化影象变换的速度极为缓慢。这一问题源于以下各种系统因素：存贮结构（用 1 维向量处理 2 维数据结构）、计算能力（一个中央处理单元必须执行对所有程序/数据的存取与计算）以及有限的地址空间（一个典型的影象对于主存通常是过于庞大了）。因此，提出了各种专用体系结构以解决上述问题。

* 因印刷困难，图 2 从略。

III 并行影像处理机的历史沿革。

二维并行阵列(图3)是一种实现分块变换的专用体系结构。阵列的每个单元是一个特殊的处理元,它包括一个存贮器记录器,用以记录当前象元值。各象元的二维结构形成一幅数字化影像。每个处理元同阵列中的各个最近单元直接联结以形成最近邻关系。

串行影像输入与转存

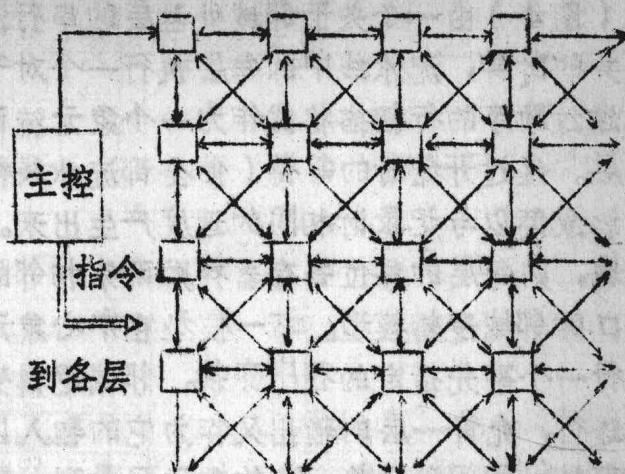


图3 并行阵列处理机

并行阵列的各处理元还包含一个最近邻变换逻辑组件,用以实现象元变换。变换指令由主控并行地传送到每个组件上,全体变换指令所形成的序列存贮在主控中,它们构成算法。每个变换在全部象元上同时执行。每个逻辑组件的输出是一新的象元状态,用它取代象元状态存贮器中的原有数值。一个完整的影像变换算法通过一系列逻辑组件程序步和象元状态邻域变换而实现。

这种结构由 Unger⁽¹³⁾于1959年首次提出。其后基于上述概念而提出或建立的系统可参看 McCormick⁽⁷⁾、Duff⁽⁴⁾、Fung⁽⁵⁾等。然而,阵列系统也存在着一些固有的问题,包括硬件过于复杂,对实时处理的带宽限制以及输入/输出困难等。这些问题将在第V节中予以研究。

IV 细胞计算机体系结构

并行阵列影像处理机的各种缺点导致密执安环境研究所 (ERIM) 发展了一个交互作用的并行结构, 即细胞计算机 (Cytocomputer) 它是本文的主要论题。“细胞计算机”这一名称由“Cyto”(希腊文“细胞”)和“computer”(计算机)构成, 前者描述它的分块 (“细胞”) 式体系结构, 后者则表示这一系统 (在“计算机”一词的正式意义下) 是一台计算机。Sternberg⁽¹⁰⁾ 在 1976 年首先提出, 一个细胞计算机 (图 4) 由一个关于邻域处理层的串行流水线构成, 各层具有一个公共的时钟, 流水线中的每层执行一个对于整个影像的简单变换。影像通过顺序的行扫描格式作为一个象元流而输入流水线, 并以匀速通过各层。经过开始时的等待 (使全部流水线得以填满) 之后, 被处理过的影像便以与记录时相同的速度产生出来。若假定窗口寄存器取 9 点邻域, 则每层的移位寄存器存贮两条相邻的扫描线以构成一个 3×3 窗口的邻域逻辑模型。这一模型在中心象元上根据该点和各邻点数值执行一个事先指定的程序变换。邻域逻辑变换在数据传送的时钟周期内进行, 允许一层的输出又作为它的输入以相同速度出现。每当经过一离散时间间隔后将一新的象元记录到一层内。与此同时, 全部延迟部件的内容移动一个单元。此外, 不参加邻域运算的各种操作, 例如换算、置位等, 可在一个独立的逐点逻辑部件 (它是邻域逻辑线路的简化) 中实现。

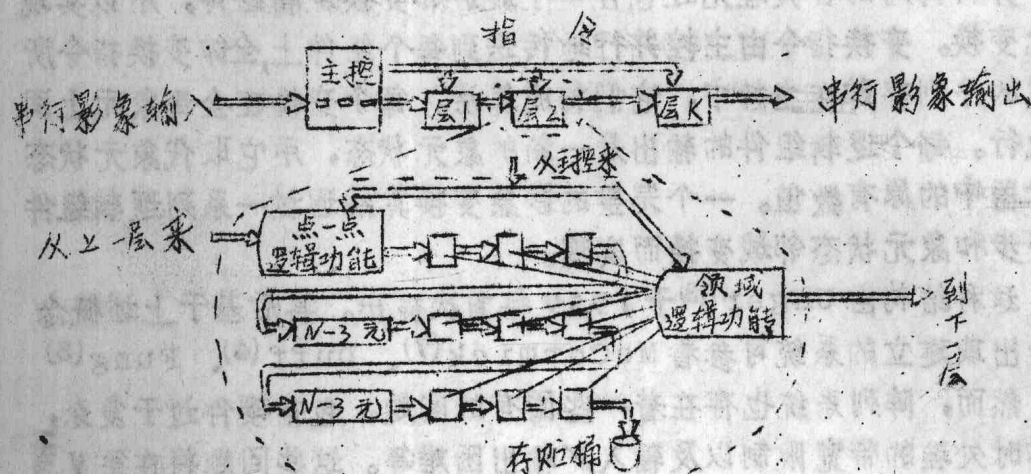


图 4 流水线影像处理机

现在来观察变换过程。假定 3×3 窗口移动而通过一个影象阵，如图5所示。处理层的存贮部分展示了当象元 A_{00} 被读入后门阵列（寄存器）的内容。根据各邻域寄存器的内容，可以计算中心寄存器所对应象元（ A_{00} ）的变换值。经过变换的象元成为一个新的元素并串行输入到下一层。由此可见，在一层上的等待时间为 $N+2$ 个时间单位， N 是线长。整个过程可以设想为由一串 3×3 窗口陆续通过影象，其中每一个处理上一层的输出（还需要一些用于处理影象边界的技术，我们的第一代硬件用一批称为“框架”象元的边界点去包围影象，这些边界点在变换中数值不变）。

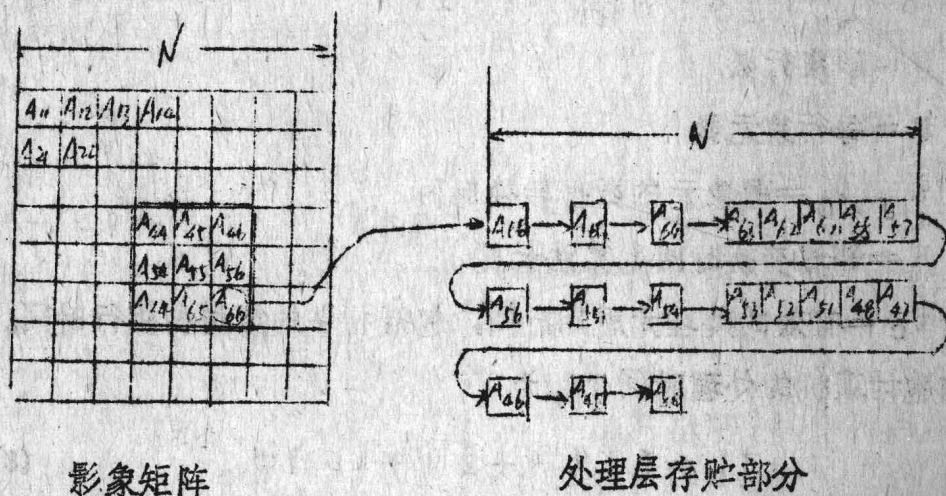


图5. 用移位寄存器存贮实现移动窗口

一般说来，串行处理层的原理或方案是不唯一的（Kruse(8), 1973）。细胞计算机的主要特点在于，它由大量实质相同的，可编软件程序的模型构成一个“链”，以设计高速（实时）影象处理方案。这将大大降低影象处理的代价。

V 细胞计算机与并行阵列系统的比较

本节将比较细胞计算机与并行阵列的几个方面，包括处理时间、

处理器复杂性、总处理能力和容错性能。首先引入一些公式以描述这两种体系结构的各种性能特征。

定义总处理时间 T 为

$$T = T_{I/O} + T_E \quad (1)$$

其中 $T_{I/O}$ = 影象输入和输出的总时间

T_E = 全部算法的执行时间。

时间 T 表示从第一个象元进入处理系统直到最后一个经处理的象元离开系统所需的时间。

对于阵列处理器，总处理时间 T_A 为 *

$$T_A = 2(MN)t_{I/O} + Kt_c \quad (2)$$

其中 M = 影象行数

N = 每行象元数

$t_{I/O}$ = 每象元的数据转换时间

K = 邻域变换时间 (算法长度)

t_c = 有效的处理元周期时间，包括指令的传送和执行时间。

细胞计算机的处理时间 T_c 为

$$T_c = [K(N+2) + MN]t_c \quad (3)$$

其 I/O 时间与处理时间完全一致。 $K(N+2)$ 项表示流水线等待时间， MN 项为影象串行处理所需的时间。由于对细胞计算机有

$t_{I/O} = t_c$ ，所以在等式中只使用 t_c 。

在进行详细的比较之前，必须确定某些条件。首先，假定并行阵列的主控程序和细胞计算机的各层都已由算法的一系列指令所规定。对于这两种系统，这都是一种必要的由务操作。其次，为简单起见

* 等式(2)和(3)假定影象数据变换器是串行的。这在多数应用系统中是真实的。其中的输入输出部分实质上也是串行的，例如光导摄像机和盘存贮器等。

假定 $M=N$ (正方形影象)。第三, 假定当前的并行阵列已经够大, 足以容纳整个影象; 类似地, 假定细胞计算机的层数不小于算法长度 K 。最后, 假定当给定了相同的实现技术方案后, 两种体系结构的 t_c 相等, $t_{I/O} = t_c$, 而且当前可以忽视阵列中的指令传送时间。

对于各种不同的算法长度 K 比较时间 T_A 与 T_C 的值。由图 6 可见, 对于算法长度相对 M (影象行数) 而言较小的情形, 细胞计算机的处理时间最短。当算法步数接近于影象行数时两种系统的处理时间大致相同。对于那些操作步数大于影象行数的算法, 阵列系统显得较为有利, 这是由于以下的事实: 若用细胞计算机处理影象, 则在第一个被处理元退出系统之前全部影象都已进入了流水线, 这时某些层将会空闲而造成浪费。但是, 若是一个细胞计算机系统用于连续地处理一系列影象幅, 上述情形便不会发生了。

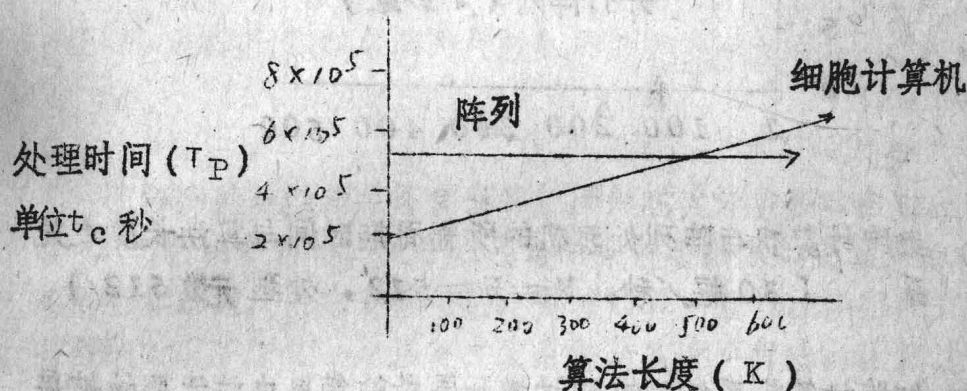


图 6 细胞计算机与阵列处理机的总处理时间与算法长度之关系
($M=N=512$)

在以上的比较中假定阵列包括 262144 个处理元, 这显然是不现实的。现在考虑具有相同复杂性的各种系统。设阵列的处理元个数给定为 P , 于是影象必须分为 $\frac{MN}{P}$ 个子部分分别处理。由于所谓 N^2 问题, 一阵列的大小最大只能为 96×96 (Cordella 等⁽²⁾, 1978), 因此在处理较大影象时阵列必须“步进”地处理整个影象。由于边界传输的影响, 这些步骤必须发生一定数量的与算法有关的重叠, 这减

低了阵列处理的速度。因此，在处理较大影象时，阵列的性能只能被认为是最好情形下的结果。阵列的周期数将是 $2MN + K \left(\frac{MN}{P} \right)$ ，因为算法必须在每个子影象上都执行一次。为了保持图 6 中关于处理时间相等的条件，假定 $P = M = N = 512$ 再来进行比较，对于一个每秒处理 30 幅影象数据的系统，算法长度对必需的周期时间的影响如图 7 所示。

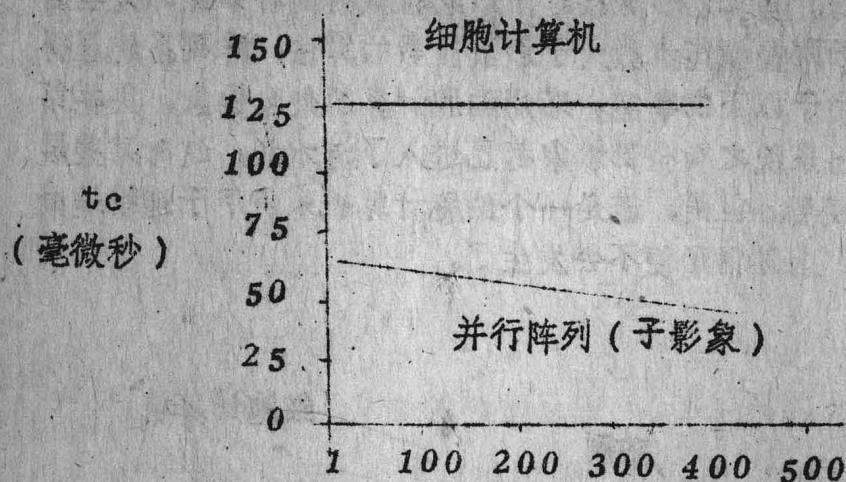


图 7 细胞计算机与阵列处理机的所需周期时间与算法长度之关系 (30 幅/秒, $M = N = 512$, 处理元数 512)

由图可见，细胞计算机系统能够通过增加层数的简单办法无限地扩展算法长度，而并行阵列则必须加快循环周期。当然，在系统总处理能力不变的前提下细胞计算机流水线的总等待时间将会增加，给定因子 $\frac{Y+K}{K}$ 在处理中随着硬件层数 X 的增加而增大。可以采用一些技术

(如并行流水线) 通过减少 N 而增加总处理能力，但这已超出了本文的范围。同样，也可以在并行运算中引入与/或使用各种阵列处理器，但这需要附加的硬件，从而意味着增加复杂性。

现在假定没有子影象处理而考虑处理器的复杂性。在细胞计算机中，每层的复杂性随存贮要求的增加而线性地增长，与影象大小的增长速

度一致。层数与层间联系数保持为常数 K 。相比之下,阵列处理器的处理元个数与互相联系数以速度 N^2 增加(见图8)。虽然可以改进集成电路工艺而增加简单装置中的线路数,但是处理元数的最大可能界限问题总会提上日程。

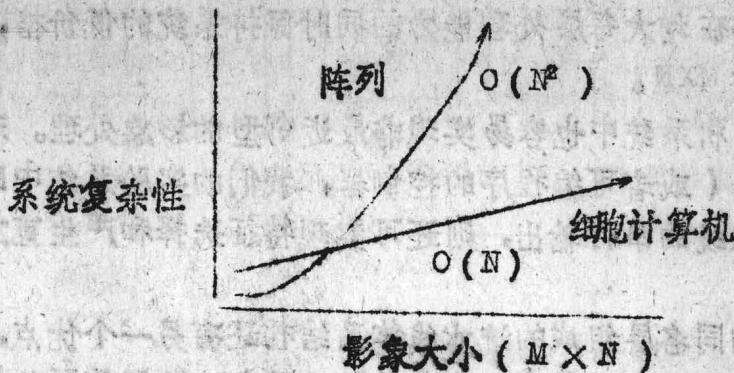


图8 细胞计算机与阵列处理机的系统复杂性与影象大小之关系
(算法长度 K 为常数)

在阵列处理器中,即使复杂性不超过允许界限,处理元的最大个数 MN 仍将由于价格、大小以及连结条件等因素而受到苛刻的限制。局部存贮的大小经常是有限的,邻域计算一般以位串行方式进行。但在细胞计算机中使一个层更为复杂却是实际可行的,因为这通常无需更多条件。复杂性的这种提高可以导致算法的缩短。我们将在这一点上加以发挥,因为它可能并不是显而易见的。

如前所述,一个最近邻型的变换在计算每一象元的结果状态时要利用围绕这一象元的窗口中各象元的初始状态。如果每个象元只需用一个bit记录,一个 3×3 窗口便有总共 $512 = 2^9$ 种可能的内容,而变换函数总共可能有 2^{512} 个。用一个512-bit的随机存取存储器(RAM)足以描述全部变换函数中的任一个。然而,若考虑8-bit的影象,变换函数的总个数便将大得惊人:

$$(2^8)^{2^{(9 \times 8)}} \text{ 种函数!}$$

显然，在RAM中描述全部变换肯定是不现实的，但是不妨在复杂性与速度二者之间取一个折中方案。一个简单的处理器可以只包含一批基本操作和一个至少1-bit的象元暂存器，并且应是全功能的，例如线路设计中的与非(NAND)门。但是，如果各层不能直接计算更复杂的函数，其后果将是扩充算法长度，即要求一算法具有更多个变换。不过，在细胞计算机系统中容易选取一些特别复杂的变换作为基本操作。这样便能够在增大每层处理能力的同时保持系统的低价格，因为K一般要远低于MXN。

在细胞计算机系统中也容易实现非最近邻型的影象处理。若用一个或多个计数器(或者可编程序的控制器，我们的实验系统中即使用后者)监督流水线的串行输出，则还可实现特征选择和产生直方图等各项算法。

最后，由相同各层组成的流水线体系结构还有另一个优点，即其固有的容错能力。如果系统中包含的层数多于算法实际需要的层数，便可在运行中容易地避开有错的各层而不致降低系统性能。这一点容易使用电子学方法完成，即用一内部或外部线路将输入数据直接传递到出错层的输出。但在并行阵列中要实现这一点则是很困难的，因为这样或者需要在每个处理模型中增加冗余设备，或者需要建立名为“插入”总线的电子线路去替换处理模型，而这种线路是一种复杂而且昂贵的系统。

VI 当前的实用系统

本文提出了一个用于影象处理计算机的独特体系结构。一个基于细胞计算机概念的处理器的已在ERIM建立并运行了一年以上。本文提出的各个例子都由这一处理器提供。本节将更详细地描述这一系统。

ERIM的影象处理实验室包括一个通用的操作处理器(图9)。它包括两种不同的层，即2维变换与3维变换分层结构，两种结构分别为88层和25层。它们由小规模集成电路(SSI)和中规模集成电路(MSI)构成。当算法需要更多步骤时可通过改变各层的程序和使经过部分处理的影象数据再入来实现。这一处理机证明了采用细胞计算机概念可以有效地增加实验室的数据处理能力。实验系统的带宽为1.6千象元/秒，程序运行时间低于1毫秒/层。

细胞计算机

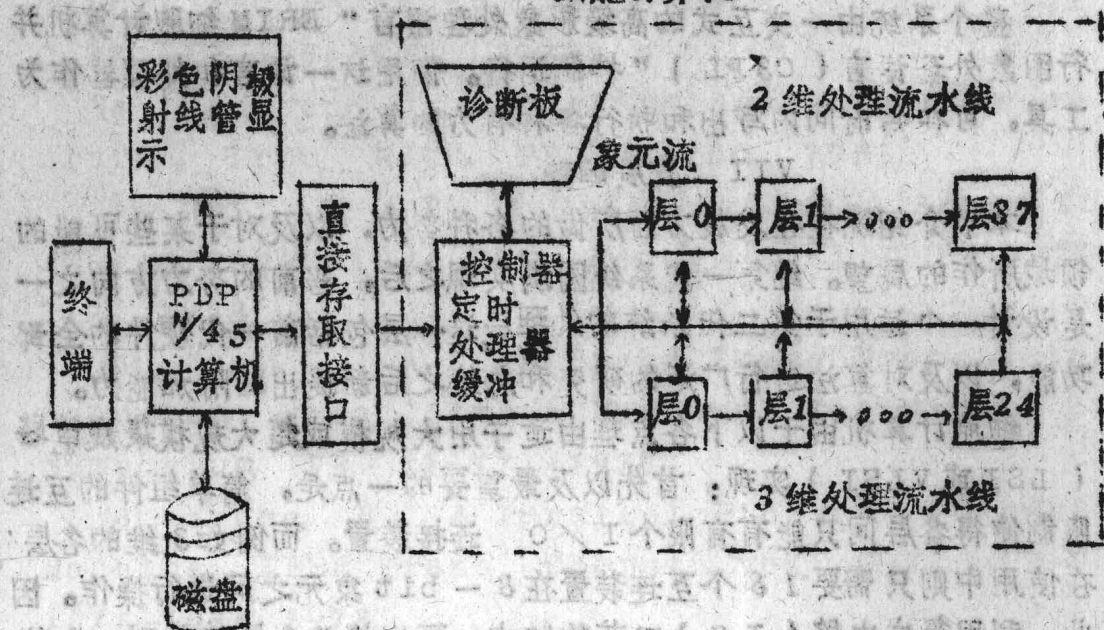


图9 ERIM 影象处理实验室

为了说明实验系统以及先进的处理算法的处理能力，考虑图10^{*}所指出的应用^{*}。图10a是一个**煤原肢一芽组织切片**的显微影象，由密执安医科大学解剖部的Thomas G. Connolly 博士提供，它是一个组织再生研究中的一部分。对影象处理的目标是自动分离、判明和标出不同的细胞核，如图10b所示。使用一个150步的算法，在我们的系统中处理这一 1024×1024 的影象大约需要0.7秒。

在各处理层上执行的指令集乃是经过对于影象处理算法的精心研究和软件模拟而选定的。被执行的变换分为两类，分别由两种不同层结构执行，和Sternberg⁽¹²⁾于1979年所述。其中一点，例如2维变换，在轮廓状的影象或者具有有限个独立状态的影象上实现。另外一类（称为3维）变换对三维的二值影象数据（可以解释为一任意参考平面上的高）的腐蚀与膨胀进行平滑。这两种层集中在两独立的流水线中，见图9。

基本指令（在单个变换中执行）的选择显然有赖于处理器的结构，无论并行阵列或者细胞计算机都是如此。本系统所用指令集的详细内容将在另文中叙述。

* 由于印刷困难，图10从略。

整个系统由一交互式的高级影象处理语言“ERIM细胞计算机并行图象处理语言(C3PL)”控制运行。使用这一语言和处理器作为工具,可在短时间内写出和执行各种有力的算法。

VII 发展趋势

本节介绍现在在发展方面所做的各种努力,以及对于某些可能的领域所作的展望。继第一代系统圆满实现之后,当前的努力方向之一是设计一个适用于第二代系统的处理层。这一层包括前一代硬件的全部功能,以及对算法进行广泛的研究和总结之后新提出的附加能力。

细胞计算机由于以下各点理由适于用大规模或超大规模集成电路(LSI或VLSI)实现:首先以及最重要的一点是,简单组件的互连限制使得各层间只能有有限个I/O连接装置。而例如3维的各层在使用中则只需要18个互连装置在8-bit象元之间进行操作。因此,利用集成电路(IC)工艺的特点,可以将几个层组装到一个简单组件中而无需增添新的装置。此外,每一系统的大量相同部分对于电气制造商们是很有好处的。最后,这一设计的灵活的可编程性质意味着本系统具有多种应用领域和广阔的市场。

VIII 结 论

本文介绍了一种用于影象处理的交互式体系结构。在当前的世界形势下,从性能价格比,处理能力和系统灵活性等几个方面来看,细胞计算机显然比并行阵列更为可取。当然,并行阵列系统仍将继续研究和生产;但是它们是否能实现上述各种优点中的任意一个则是大可怀疑的。我们希望,这篇文章能够引起对于影象处理系统的体系结构的进一步分析、讨论和发展。

鸣 谢

作者们深切感谢密执安环境研究所(ERIM)的全体工作人员在本文的准备过程中所做的贡献,以及ERIM对本文工作的支持。

参 考 文 献

(1) Codd, E. F., Cellular Automata, New York, Academic Press, 1968.

(2) Cordella, L., M. J. B. Duff, S. Levialdi, An Analysis of Computational Cost in Image Proce-

ssing : A Case Study, IEEE Trans. Comput. vol. C-27, PP. 904-910, 1978.

(8) Dinneen, G., Programming Pattern Recognition, Proc. WJCC, PP. 94-100, 1955.

(4) Duff, M. J. B., A Cellular Logic Array for Image Processing, Pat. Recog., vol. 5, PP. 229-247, 1973.

(5) Fung, L., A High Speed Image Processing Computer, Proc. 17th ACM Annual Technical Symposium, PP. 11-17, 1978.

(6) Kruse, B., A Parallel Picture Processing Machine, IEEE Trans. Comput., vol. C-22, PP-1075-1086, 1973.

(7) Mc Cormick, B. H., The Illinois Pattern Recognition Computer — ILLIAC III, IEEE Trans. Electron. Commun., vol. EC-12, PP. 791-813, 1963.

(8) Preston, K., Jr., M. J. B. Duff, S. Levialdi, P. E. Norgren, J. Toriwaki, Basics of Cellular Logic with Some Applications in Medical Image Processing, Proc. IEEE, vol. 67, No. 5, PP. 826-856, 1979.

(9) Selfridge, O., Pattern Recognition in Modern Computers, Proc. WJCC, PP. 94-97, 1955.

(10) Sternberg, S., Automatic Image Processor, U. S. Patent 4,167,728.

(11) Sternberg, S., Cytocomputer Real-Time Pattern Recognition, Proc. 8th Automatic Imagery Pattern Recognition Symposium, P. 205, 1978.

(12) Sternberg, S., An Architecture for Real-Time Biomedical Image Processing, Proc. 3rd Inter. Computer Software Software and Applica-

Conference in press. A Case Study

(19) Unger, S. H., Pattern Recognition and

Detection, Proc. IEEE, vol. 47, P. 1737, 1959.

IEEE Proc. WJCC, P. 94-100, 1955

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

陈祖荫译

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959

Image Processing, Proc. IEEE, vol. 47, P. 1737, 1959