第19卷 第1期 光 学 学 报 Vol. 19, No. 1
1999年1月 ACTA OPTICA SINICA January, 1999

自适应光学波前计算的并行性研究*

陈 严** 孔铁生 梁甸农
(国防科技大学电子工程学院, 长沙 410073)

摘 要 在自适应光学系统中, 波前计算是指由波前斜率计算、波前复原运算以及波前控制算法等组成的一类处理任务。本文研究了从波前计算算法表达层到结构层的优化映射的实现方法, 根据现有的并行计算理论提出了适用于波前计算的两种基于单指令流多数据流(SIMD)结构的并行算法、倍增算法和分组算法。在此基础上, 以乘-累加(MAC)时间和输入/输出(I/O)时间作为衡量算法性能的两个指标, 在通用数字信号处理(DSP)芯片构筑的单指令流多数据流阵列机上, 对上述两种算法的计算效能进行了详细分析和比较。

关键词 并行算法, 自适应光学, 波前计算, 单指令流多数据流结构。

1 引 言

Hardy 在自适应光学系统进展述评[1]中指出, 波前计算需要解决的主要问题是数据的处理速率、串行计算速度慢, 可能成为自适应光学系统的真正障碍, 并行处理是克服这一障碍的唯一途径。波前计算一般是指由波前斜率计算、波前复原运算以及波前控制算法等组成的一类处理任务。对于哈特曼(Hartmann-Shack)波前探测器, 这类任务的处理对象是二维图像数据, 它是对数据简单、重复的递归计算, 具有规则性和可划分性, 非常适合采用单指令流多数据流结构提高运算速度。本文根据现有的并行计算理论重点研究了从波前计算算法表达层到结构层的优化映射的实现方法, 提出了适用于波前计算的两种基于单指令流多数据流结构的并行算法, 即倍增算法和分组算法; 在此基础上, 以乘-累加时间和输入/输出时间作为衡量算法性能的两个指标, 对上述两种算法在由通用数字信号处理芯片构筑的单指令流多数据流阵列机上的计算效能进行了详细分析和比较。

2 波前计算的算法改造

作者曾对直接斜率波前复原算法的并发性进行过研究[2, 3], 最终的波前误差向量 $\boldsymbol{E}$ 是每个子孔径在 x、y 方向的波前斜率经过 I 次递归运算而得到。直接斜率波前复原算法经并发性提取被改造为如下的一阶递归形式:

* 国家科委863高科技项目。

** 现通信地址: 北京市复兴路14号19分队(重型办), 北京 100843。

收稿日期: 1997-11-18; 收到修改稿日期: 1998-02-20

$$\boldsymbol{E}^k = \boldsymbol{E}^{k-1} + \left(\boldsymbol{m}_{xk} - \frac{1}{N}M_x\boldsymbol{l}\right)s_{xk} + \left(\boldsymbol{m}_{yk} - \frac{1}{N}M_y\boldsymbol{l}\right)s_{yk} =$$

$$\boldsymbol{E}^{k-1} + \boldsymbol{a}_k s_{xk} + \boldsymbol{b}_k s_{yk}, \qquad k = 1, 2, \cdots, I \tag{1}$$

式中, $\boldsymbol{E}^k$ 表示 N 维波前误差向量 $\boldsymbol{E}$ 的第 k 次迭代值, N 为波前校正器的数目, I 是参加迭代的子孔径数目。M_x 和 M_y 均为 $N \times I$ 矩阵, $\boldsymbol{m}_{sk}$、$\boldsymbol{m}_{yk}$ 分别为矩阵 M_x 和 M_y 的第 k 列矢量; s_{xk} 和 s_{yk} 分别为第 k 个子孔径 x、y 方向的波前斜率; N 维向量 $\boldsymbol{a}_i$、$\boldsymbol{b}_i$ 分别满足:

$$\boldsymbol{a}_i = \boldsymbol{m}_{xk} - \frac{1}{N}M_x\boldsymbol{l} \tag{2a}$$

$$\boldsymbol{b}_i = \boldsymbol{m}_{yk} - \frac{1}{N}M_y\boldsymbol{l} \tag{2b}$$

波前误差向量的初始值 $\boldsymbol{E}^0 = [0, 0, \cdots, 0]^T$。$\boldsymbol{l}$ 为 I 维列矢量, 满足:

$$\boldsymbol{l} = [1, 1, \cdots, 1]^t$$

在自适应光学系统中, 除波前复原运算之外, 波前计算任务还包括波前斜率计算和波前控制算法。波前斜率计算属于累加问题, 因此可以很容易地转换为如(1)式的一阶递归问题; 同样, 也可以将(1)式推广到如波前控制算法所描述的求解多阶递归计算问题。本文为简化分析和保持叙述的连贯性, 仍以直接斜率波前复原算法作为研究对象。

为了方便分析, 只考虑直接斜率波前复原算法(1)式的标量表达形式:

$$\left.\begin{aligned} e^1 &= c_1 \\ e^i &= \mu_i e^{i-1} + c_i, \qquad i = 2, 3, \cdots, N \end{aligned}\right\} \tag{3}$$

其中, (1)式中的递归次数 I 用 N 表示, $\mu_i = 1$, $i = 2, 3, \cdots, I$; e^i 表示波前误差向量 $\boldsymbol{E}$ 中的任意一个元素 e 的第 i 次递归值, c_i 是向量 $\boldsymbol{C}$ 中与 e 位置相同的元素 c 的第 i 次递归值, 其中 $\boldsymbol{C}$ 满足:

$$\boldsymbol{C} = \boldsymbol{a}_i s_{xi} + \boldsymbol{b}_i s_{yi}, \qquad i = 1, 2, \cdots, N$$

$\boldsymbol{a}_i$、$\boldsymbol{b}_i$、s_{xi}、s_{yi} 的定义同上。

3　倍增算法和分组算法

本节将根据现有的并行计算理论[4], 针对以直接斜率波前复原运算为代表的波前计算问题, 提出了两种基于单指令流多数据流结构的并行算法: 倍增算法和分组算法。在讨论倍增算法和分组算法之前, 首先引入以下定义:

定义1　给定正整数 i、j 和 N, 且满足: $i \leqslant j$, $j \leqslant N$, 定义:

$$Q(j, i) = \sum_{l=1}^{j}\left(\prod_{k=l+1}^{j}\mu_k\right)c_l \tag{4}$$

$$P(j, i) = \prod_{k=i}^{j}\mu_k \tag{5}$$

且规定, 当 $s \geqslant 1$ 时, $\prod_{k=j+s}^{j}\mu_k = 1$。

根据定义 1, 函数 $Q(j, i)$ 具有下述三个性质[7]:

1) $Q(i, i) = c_i$, $\qquad i = 2, 3, \cdots, N$;

2) 当 $i < j$ 时, 对任一非负整数 s, 当 $i + s < j$ 时, $Q(j, i) = \left(\prod_{k=j+s+1}^{j}\mu_k\right)Q(i + s, i) + Q(j, i + s + 1)$ 成立;

3) $Q(j,\ 1) = e^{j}$, $\quad j = 1, 2, \cdots, N$。

3.1 倍增算法

令 $j = 2i$ 和 $j = 2i + 1$, 根据 Q 的定义和上述三个性质, 波前误差的计算公式(3)可写成如下形式:

$$\left.\begin{aligned} e^{2i} &= \Big(\prod_{k=i+1}^{2i}\mu_k\Big) Q(i,\ 1) + Q(2i,\ i+1) \\ e^{2i+1} &= \Big(\prod_{k=i+1}^{2i+1}\mu_k\Big) Q(i,\ 1) + Q(2i+1,\ i+1) \end{aligned}\right\} \tag{6}$$

从上式看出, 如果在第 k 步要计算出 e^{2i}, 则在第 $k-1$ 步必须并行计算出 $Q(i,\ 1)$ 和 $Q(2i,\ i+1)$。由 $Q(i,\ 1)$ 和 $Q(2i,\ i+1)$ 的定义知, 它们的结构相同, 且独立, 因而可以同时计算, 如果使用 N 台处理机, 则经过 $\log_2 N$ 步可计算出全部结果 $e^1 \sim e^N$, 在对(6)式组织并行计算之前, 首先引入下列记号:

S: 倍增法的计算步;

$\boldsymbol{P}^S$、$\boldsymbol{Q}^S$: 第 S 步计算(6)式时, 相应以 $P(j,\ i)$ 和 $Q(j,\ i)$ 为元素的 N 维向量, 定义:

$$\boldsymbol{P}^S = [P(S,\ 1), P(S,\ 2), \cdots, P(S,\ i), \cdots, P(S,\ N)]^T, \quad S = 1, 2, \cdots, \log_2 N \tag{7}$$

$$\boldsymbol{Q}^S = [Q(S,\ 1), Q(S,\ 2), \cdots, Q(S,\ i), \cdots, Q(S,\ N)]^T, \quad S = 1, 2, \cdots, \log_2 N \tag{8}$$

$\boldsymbol{P}^S$: 将 $\boldsymbol{P}^S$ 后移 2^{S-1} 个分量, 其前面空出分量补以 1, 定义:

$$\begin{aligned} \boldsymbol{P}^S = [&\underline{P}(S,\ 1), \underline{P}(S,\ 2), \cdots, \underline{P}(S,\ i), \cdots, \underline{P}(S,\ N)]^T = \\ &[1,\ 1, \cdots, 1, P(S,\ 2^{S-1}+1), \cdots, P(S,\ N-2^{S-1})]^T, \quad S = 1, 2, \cdots, \log_2 N \end{aligned} \tag{9}$$

$\boldsymbol{Q}^S$: 将 $\boldsymbol{Q}^S$ 后移 2^{S-1} 个分量, 其前面空出分量补以 0, 定义:

$$\begin{aligned} \boldsymbol{Q}^S = [&\underline{Q}(S,\ 1), \underline{Q}(S,\ 2), \cdots, \underline{Q}(S,\ i), \cdots, \underline{Q}(S,\ N)]^T = \\ &[0,\ 0, \cdots, 0, Q(S,\ 2^{S-1}+1), \cdots, Q(S,\ N-2^{S-1})]^T, \quad S = 1, 2, \cdots, \log_2 N \end{aligned} \tag{10}$$

根据上述记号, 并引入代数运算算子⊙, (6)式的向量化计算公式如下:

$$\left.\begin{aligned} \boldsymbol{P}^{S+1} &= \boldsymbol{P}^S \odot \boldsymbol{Q}^S \\ \boldsymbol{Q}^{S+1} &= \boldsymbol{P}^S \odot \boldsymbol{Q}^S + \boldsymbol{Q}^S, \quad S = 0, 1, 2, \cdots, \log_2 N - 1 \end{aligned}\right\} \tag{11}$$

其中, 算子⊙定义为:

$$\begin{bmatrix} x_1 \\ x_i \\ x_N \end{bmatrix} \odot \begin{bmatrix} y_1 \\ y_i \\ y_N \end{bmatrix} = \begin{bmatrix} x_1 y_1 \\ x_i y_i \\ x_N y_N \end{bmatrix} \tag{12}$$

根据(11)式的数学描述, 可以得到一种基于单指令流多数据流模型求解波前误差的并行算法, 这个算法的具体描述形式详见附录中的算法 1, 称之为直接斜率波前复原倍增算法。

3.2 分组算法

分组是一种将原问题划分为可并行求解的子问题的计算方法, 其计算步骤是, 1) 将给定的问题划分为 k 个独立的几乎等尺寸的子问题; 2) 用 k 台处理机并行求解诸子问题。在波前复原算法中采用倍增技术, 需要 N 台处理机并行工作才能保证经过 $\log_2 N$ 步可计算出全部结果 $e^1 \sim e^N$。但是, 实际情况是处理机数目往往小于 N, 这时可将(3)式分成若干组, 然后分别对每组组织并行计算。

将(3)式中 N 个递归元素分成 k 组, 每组 J 个元素, 即 $N = kJ$, 且记为 $\{e^{iJ+j}\}$, $i = 0, 1,$

…, $k-1$; $j=1, 2, \cdots, J$。

$$\left.\begin{aligned} e^{iJ+1} &= \mu_{iJ+1}E^{ij} + c_{iJ+1} = Q(iJ+1, 1) \\ e^{iJ+2} &= \mu_{iJ+2}a_{iJ+1}E^{iJ} + \mu_{iJ+2}c_{iJ+1} + c_{iJ+2} = Q(iJ+2, 1) \\ &\cdots\cdots \\ e^{(i+1)J} &= \mu_{(i+1)J}\cdots\mu_{iJ+1}E^{iJ} + \cdots + \mu_{(i+1)J}c_{(i+1)J-1} + c_{(i+1)J} = Q[(i+1)J, 1] \end{aligned}\right\} \quad (13)$$

从(13) 式可知, 各组之间对应分量的计算是一致的, 仍可采用(4)、(5) 式函数 P、Q 的定义组织并行计算, 其中 $P(j, i)$ 和 $Q(j, i)$ 以及 e^{iJ+j} 的递推关系式满足:

$$\left.\begin{aligned} P(j, 1) &= \mu_j P(j-1, 1) \\ P(iJ+j, iJ+1) &= \mu_{iJ+j}P(iJ+j-1, iJ+1) \end{aligned}\right\} \quad (14)$$

$$\left.\begin{aligned} Q(j, 1) &= \mu_j Q(j-1.\ 1) + c_j \\ Q(iJ+1, iJ+1) &= \mu_{iJ+1}Q(iJ+j-1, iJ+1) + c_{iJ+1} \end{aligned}\right\} \quad (15)$$

$$\begin{aligned} e^{iJ+1} = \mu_{iJ+j}\cdots\mu_{iJ+1}e^{iJ} + \mu_{iJ+j}\cdots\mu_{iJ+2}c_{iJ}\cdots + \mu_{iJ+1}c_{iJ+j-1} + c_{iJ+j} = \\ P(iJ+j, iJ+1)\ e^{iJ} + Q(iJ+j, iJ+1), \\ i=0, 1, \cdots, K-1;\ j=0, 1, \cdots, J \end{aligned} \quad (16)$$

同样, 在利用分组方法对(14) 式~ (16) 式组织并行计算之前, 首先引入下列记号:

S: 分组方法的计算步;

$\boldsymbol{P}^S$、$\boldsymbol{u}^S$: 第 S 步计算(14) 式时, 分别得到以 $P(j, i)$ 和 μ_i 为元素的两个 K 维向量, 定义:

$$\boldsymbol{P}^S = \{P(S+1, 1), \cdots, P(S+J, J), \cdots, P[S+(K-1)J, (K-1)J)]^T, \quad S=1, 2, \cdots, J \quad (17)$$

$$\boldsymbol{u}^S = [\mu_{1+S}, \cdots, \mu_{kJ+1+S}, \cdots, \mu_{(k-1)J+1+S}]^T, \quad S=0, 1, \cdots, J-1 \quad (18)$$

$\boldsymbol{Q}^S$、$\boldsymbol{C}^S$: 第 S 步计算(15) 式时, 分别得到以 $Q(j, i)$ 和 c_i 为元素的两个 k 维向量, 定义:

$$\boldsymbol{Q}^S = \{Q(S+1, 1), \cdots, Q(S+J, J), \cdots, Q[S+(K-1)J, (K-1)J]\}^T \quad (19)$$

$$\boldsymbol{C}^S = [c_{1+S}, \cdots, c_{kJ+1+S}, \cdots, c_{(k-1)J+1+S}]^T, \quad S=0, 1, \cdots, J-1 \quad (20)$$

$\boldsymbol{P}^S$、$\boldsymbol{Q}^S$、$\boldsymbol{E}^S$: 第 S 步计算(16) 式时, 得到一个以 $P(j, 1)$、$Q(j, 1)$ 和 e^{SJ+j} 为元素的三个 J 维向量, 定义:

$$\boldsymbol{P}^S = \{P(SJ+1, SJ+1), \cdots, P(SJ+j, SJ+1), \cdots, P[(S+1)J, SJ+1]\}^T, \quad S=0, \cdots, K-1 \quad (21)$$

$$\boldsymbol{Q}^S = \{Q(Sj+1, SJ+1), \cdots, Q(SJ+j, SJ+1), \cdots, Q[(S+1)J, SJ+1)]\}^T, \quad S=0, \cdots, K-1 \quad (22)$$

$$\boldsymbol{E}^S = [e^{SJ+1}, \cdots, e^{SJ+j}, \cdots, e^{SJ+J}]^T, \quad S=0, 1, \cdots, K-1 \quad (23)$$

显然, 要对 $\{e^{iJ+j}\}(i=0, 1, \cdots, k-1;\ j=1, 2, \cdots, J)$ 组织并行计算, 必须分两个阶段进行。第一阶段, 对(14) 和(15) 式组织并行计算。根据上面的分组和记号, 在 k 个处理机同时工作下, (14) 和(15) 式按 i 方向并行, j 方向串行, 经过 $(J-1)$ 个计算步可求出所有的 $P(j, i)$ 和 $Q(j, i)$; 在每一个计算步, k 个处理机所求得的 $P(j, i)$ 和 $Q(j, i)$ 还须完成数据置换, 为下一阶段的并行计算准备数据。此阶段的向量化并行计算公式如下:

$$\left.\begin{aligned} \boldsymbol{P}^{S+1} &= \boldsymbol{P}^S \odot \boldsymbol{u}^S \\ \boldsymbol{Q}^{S+1} &= \boldsymbol{Q}^S \odot \boldsymbol{C}^S, \quad S=0, 1, 2, \cdots, J-1 \end{aligned}\right\} \quad (24)$$

式中算子⊙ 的定义与(12) 式相同。

第二阶段, 在计算出所有的 $P(j, i)$ 和 $Q(j, i)$ 并完成数据置换之后, 在 J 个处理机同时

工作下，(16) 式再按 i 方向串行，j 方向并行，经过 $k-1$ 个计算步可求出所有的$\{e^{iJ+j}\}$，$i=0, 1, \cdots, k-1$; $j=1, 2, \cdots, J$。这一阶段的向量化并行计算公式如下:

$$\boldsymbol{E}^{S+1}=\boldsymbol{P}^{S}e^{SJ}+\boldsymbol{Q}^{S}, \qquad S=0, 1, 2, \cdots, K-1 \tag{25}$$

根据并行计算公式(24)和(25)，可得到另一种基于单指令流多数据流模型求解波前误差的并行算法，此算法的具体形式见附录中的算法 2，称之为直接斜率波前复原分组算法。

4 两种算法在单指令流多数据流阵列机上的计算效能

不同算法在同一计算机结构上的计算效率往往是有差异的，特别是上节提出的两种波前复原并行算法，倍增算法和分组算法在单指令流多数据流结构上可能差别更大。因此，本节针对上述两种算法在单指令流多数据流结构上的计算效能进行讨论。

4.1 预备知识

对于一个给定的问题，如果其新的并行算法已设计出来，通常人们使用算法的运行时间、加速比、效率和算法成本等准则[5]来评估新的算法，这些准则的定义如下:

1) 并行算法的运行时间是指算法在并行计算机上解一个问题所需时间，即表示算法开始执行到算法执行完以后的这一段时间。

2) 并行算法加速比 S_P 定义为:

$$S_P=\frac{\text{算法在单处理机上的实际执行时间}}{\text{使用 } P \text{ 台处理机时算法的实际执行时间}} \tag{26}$$

3) 并行算法的效率 E_P 定义为:

$$E_P=S_P/P \tag{27}$$

4) 并行算法的成本(Cost)定义为:

并行算法的运行时间乘以算法所使用的处理机数目。

4.2 单指令流多数据流模型上的计算效能

近年来，随着超大规模集成(VLSI)技术的发展，诞生了一批专用于数字信号处理的高速处理器件(数字信号处理芯片)。由于这类芯片的特定应用背景，它们所具有的结构与通用微处理器有明显的不同，片内高速算术处理单元、流水线操作、多个独立的存储器设置以及并行功能单元均使数字信号处理芯片的性能得到了改善，乘–累加时间是评价这类可编程数字信号处理芯片性能最基本的统计数据。

李晓梅等对线性递归问题的并行算法在单指令流多数据流阵列机上的计算效能进行了比较详细的分析和讨论[5]。但是，这些讨论是基于通用微处理器构筑的单指令流多数据流模型，并将乘法和加法时间作为衡量整个计算性能的两个要素。显然，在数字信号处理芯片构筑的单指令流多数据流模型中，实际的情况是：乘–累加取代了乘法时间和加法时间成为衡量计算性能的一个关键指标。其次，在分组算法中，一阶递归问题的并行计算是分两个阶段进行的，在不同的阶段，每个处理机处理的对象不同，即第一阶段得到的数据必须经过数据置换，为第二阶段的并行计算准备好数据。数据置换通过输入/输出操作实现，因此数据的输入/输出时间必须计入并行算法的计算时间内，遗憾的是，上述文献中忽略了这可观的输入/输出时间。

这一节中，以乘–累加时间和输入/输出时间作为衡量并行算法的计算效能的两个基本指标，它们分别用 τ_{MAC} 和 $\tau_{I/O}$ 表示。在以下讨论中，只考虑变量为标量的情况，按(1)式的串行计算需 $N-1$ 步，每步需要完成两次乘–累加，没有输入/输出操作。因此，(1)式串行计算

的时间 T_1 为:

$$T_1 = 2\tau_{MAC}(N-1) \tag{28}$$

4.2.1　倍增算法

设处理机的数目为 P, 当 $P = N$ 时, 按算法 1 组织并行计算, 需要经过 $\log_2 N$ 步计算得到所有的结果; 当 $P < N$ 时, 则首先将 N 分成 $\left\lceil \frac{N}{P} \right\rceil$ 段, 每段按算法 1 组织并行计算, 需要 $\log_2 P$ 计算步, 得到所有结果则需要 $[\log_2 P]\left\lceil \frac{N}{P} \right\rceil$ 计算步。

在一个计算步中, 倍增算法需要完成四次乘–累加, 没有输入/输出操作。T_D、S_D 和 E_D 分别用来表示倍增算法的运行时间、加速比和效率, 它们满足:

$$\begin{cases} T_D = 4\tau_{MAC}[\log_2 P]\left\lceil \frac{N}{P} \right\rceil \\ S_D = [2\tau_{MAC}(N-1)/4\tau_{MAC}][\log_2 P]\left\lceil \frac{N}{P} \right\rceil \\ E_D = [2\tau_{MAC}(N-1)/P]4\tau_{MAC}[\log_2 P]\left\lceil \frac{N}{P} \right\rceil \end{cases} \tag{29}$$

4.2.2　分组算法

设处理机的数目为 P, $N = KJ$, 假设 $P < K$、$P < J$。根据算法 2, 要得到最终的结果必须分两步: 第一步, 首先应对(24) 式组织并行计算。由于 $P < K$, 又需对 K 进行分段, 假定分成 $\left\lceil \frac{K}{P} \right\rceil$ 段。那么在 P 个处理机同时工作下, (24) 和(25) 式按 i 方向并行, j 方向串行, 经过 $\left\lceil \frac{K}{P} \right\rceil (J-1)$ 个计算步可求出所有的 $P(j, i)$ 和 $Q(j, i)$, 每个计算步需要完成四次乘–累加和两次输入/输出操作。所用的时间为:

$$T_P^1 = (4\tau_{MAC} + 2\tau_{I/O})\left\lceil \frac{K}{P} \right\rceil (J-1) \tag{30}$$

第二步, 在计算出所有的 $P(j, i)$ 和 $Q(j, i)$ 之后, 应对(25) 式组织并行计算。由于 $P < J$, 又需对 J 进行分段, 假定分成 $\left\lceil \frac{J}{P} \right\rceil$ 段。那么在 P 个处理机同时工作下, 按 i 方向串行, j 方向并行, 经过 $\left\lceil \frac{J}{P} \right\rceil (K-1)$ 个计算步可求出所有的 $\{e^{iJ+j}\}$, $i = 0, 1, \cdots, k-1$; $j = 1, 2, \cdots, J$。每个计算步需要完成两次乘–累加和两次输入/输出操作。所用的时间为:

$$T_P^2 = (2\tau_{MAC} + 2\tau_{I/O})\left\lceil \frac{J}{P} \right\rceil (K-1) \tag{31}$$

T_P、S_P 和 E_P 分别表示分组算法的运行时间、加速比和效率, 它们满足:

$$\begin{cases} T_P = T_P^1 + T_P^2 \\ S_P = 2\tau_{MAC}(N-1)/(T_P^1 + T_P^2) \\ E_P = 2\tau_{MAC}(N-1)/P(T_P^1 + T_P^2) \end{cases} \tag{32}$$

4.2.3　两种并行算法的计算效能

TMS320C50 是美国 TI 公司生产的第五代 16 位定点数字信号处理器, 它的单周期的指令执行时间为 50 ns; 片内随机存取存储器(RAM) 可达 12 k×16 bits; 最大寻址空间为 192 k×16 bits; 单指令周期可完成乘–累加操作; 在无等待状态下, 两个指令周期可完成输入/输出操作。根据每个乘–累加和输入/输出操作在 TI 公司的 TMS320 系列数字信号处理芯片上

(以 TMS320C50 为例) 的实际执行时间 τ_{MAC} 和 $\tau_{I/O}$ 可以得到两种并行算法在单指令流多数据流结构上的计算效能, 如图 1 和图 2 所示。

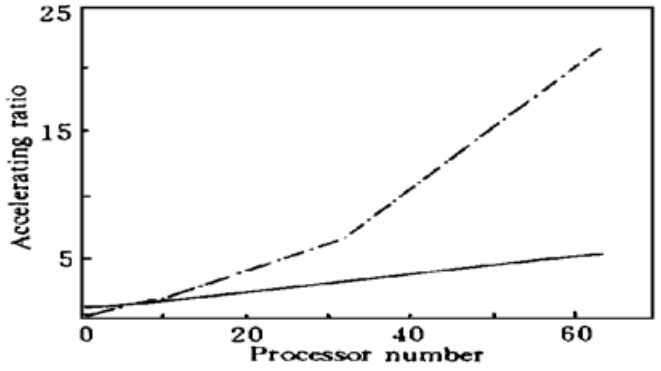

Fig. 1 Relation of accelerating ratio and processor number

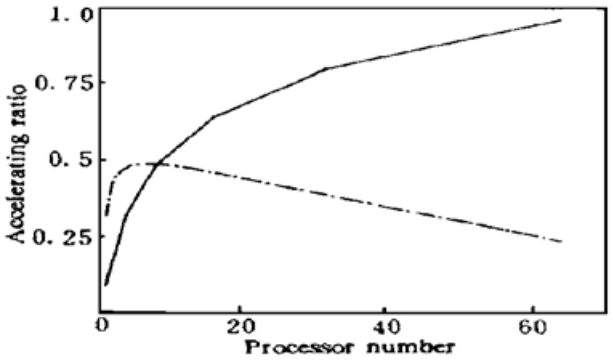

Fig. 2 Relation of nomalizing cost and processor number

图 1 是两种算法的加速比曲线, 横坐标代表处理机数目, 纵坐标代表加速比; 图 2 是两种算法的成本曲线, 横坐标代表处理机数目, 纵坐标代表归一化成本; 图中的实线表示倍增算法, 点划线表示分组算法, 并假设 $N = PJ$ 且 $P = K$ 为 2 的方幂。

从图中看出, 在单指令流多数据流阵列处理结构中, 两种并行算法的加速比与递归计算次数、分段数以及处理单元数目是紧密联系的。当递归次数与处理机数相等时, 倍增法和分组方法的加速比均达到最大。这是因为处理机数目越多, 倍增法和分组方法中所需的计算步越少, 并行算法的执行时间就越短。

此外, 还发现这样一个事实, 当处理机数目 P 超过某个值后, 分组算法的性能将超越倍增算法。为解释这个现象, 假设 $\tau = \tau_{MAC} = \tau_{I/O}$, 根据(29) 和(32) 式可得

$$T_D = 4\tau(\log_2 P)N/P \tag{33}$$

$$T_P = T_P^1 + T_P^2 = 6\tau(N/P - 1) + 4\tau(N/P^2)(P - 1) = 10\tau N/P - 6\tau - 4\tau N/P^2 \tag{34}$$

显然, 如果分组算法的计算时间 T_P 小于倍增算法的计算时间 T_D, 即 $T_P - T_D \leqslant 0$, 这就等效于分组算法的加速比将超过倍增算法。此时, 处理机数目 P 与递归次数 N 的关系满足下式:

$$(10P - 4P\log_2 P - 4)N \leqslant 6 \tag{35}$$

以上分析说明在将一个实际的递归问题映射到单指令流多数据流结构之前, 可根据递归项数和处理机的实际数目是否满足(35) 式以确定是采用倍增算法还是分组算法。

结 论 本文以波前复原运算为代表, 研究了从波前计算算法表达层到结构层的优化映射的实现方法, 提出了两种基于单指令流多数据流结构的并行算法: 直接斜率波前倍增算法和分组算法。在此基础上, 以乘-累加时间和输入/输出时间作为衡量算法性能的两个指标, 在通用数字信号处理芯片构筑的单指令流多数据流阵列机上, 对上述两种算法的计算效能进行了详细分析和比较。本文的研究结果对高分辨率自适应光学系统中的实时信号处理系统结构的设计和实现具有指导意义。

对中科院成都光电技术研究所八室全体同志在室内模拟试验中给予的支持和帮助, 表示深深的谢意。

参 考 文 献

[1] J. W. Hardy. Adaptive optics ——a progress review. *Proc. SPIE*, 1991, **1542**: 2~17

[2] 陈　严, 孔铁生, 梁甸农. 高速自适应光学波前处理器——流水式多SIMD结构. 电子学报, 1998, **26**(3): 100~102

[3] Chen Yan, Tie-sheng Kong, Dian-hong Liang. Pipelining multiple SIMD architecture for Hartmann-Shack sensor. *Proc. SPIE*, 1997, **2661**: 34~42

[4] 陈国良. 并行算法的设计与分析. 北京: 高等教育出版社, 1994. 26~46

[5] 李晓梅, 蒋增荣. 并行算法, 长沙: 湖南科学技术出版社, 1992. 123~198

附录: 用类-Algo 语言描述的两种直接斜率波前复原并行算法

算法1　SIMD模型上的直接斜率波前复原倍增算法

输入: 初值向量 $\boldsymbol{P}^0$、$\boldsymbol{Q}^0$

输出: $\boldsymbol{Q}^S$

```
begin
  (1) for all k par-do
      (1.1) P(0, k) ← μ_k
      (1.2) Q(0, k) ← c_k
      end for
  (2) for S = 1 to (log2 N)
        for all k par-do
          (2.1) P(S, k) ← P(S − 1, k) P(S − 1, k)
          (2.2) Q(S, k) ← P(S − 1, k) Q(S − 1, k) + Q(S − 1, k)
        end for
      end for
end
```

算法2　SIMD模型上的直接斜率波前复原分组算法

输入: 初值向量 $\boldsymbol{P}^0$、$\boldsymbol{Q}^0$

输出: $\boldsymbol{E}^S$

```
begin
  (1) for all k par-do
      (1.1) P[(k − 1)J + 1, (k − 1)J + 1] ← μ_{(k−1)J+1}
      (1.2) Q[(k − 1)J + 1, (k − 1)J + 1] ← c_{(k−1)J+1}
      end for
  (2) for j = 2 to J
      for all k par-do
      (2.1) P[(k − 1)J + j, (k − 1)J + 1] ← μ_{(k−1)J+j} P[(k − 1)J + j − 1, (k − 1)J + 1]
      (2.2) Q[(k − 1)J + j, (k − 1)J + 1] ← μ_{(k−1)J+j} Q[(k − 1)J + j − 1, (k − 1)J + 1] + c_{(k−1)J+j}
      (2.3) out P[(k − 1)J + j, (k − 1)J + 1]
      (2.4) out Q[(k − 1)J + j, (k − 1)J + 1]
      end for
      end for
  (3) for k = 1 to k − 1
      for all j par-do
        (3.1) in P(kJ + j, kJ + 1)
        (3.2) in Q(kJ + j, kJ + 1)
        (3.3) e^{kJ+j} ← P(kJ + j, kJ + 1) e^{kJ} + Q(kJ + j, kJ + 1)
      end for
      end for
end
```

The Parallel Algorithms of Wavefront Computing in Adaptive Optics

Chen Yan Kong Tiesheng Liang Diannong

(*National University of Defense Technology, Department of Electronic Engineering, Changsha* 410073)

(Received 18 November 1997; revised 20 February 1998)

Abstarct In adaptive optics system, wavefront computation refers to a class of processing consisting of wavefront slope computation, wavefront reconstruction and wavefront control algorithm. We studied the realization of the optimization mapping of the wavefront computing algorithm from the representive layer to the architecture layer. Two SIMD-based parallel algorithms, the doubling algorithm and the partitioning algorithm are proposed according to the existing parallel computing theory. On the basis above, the MAC cycle and I/O cycle are used to scale the performance of the two algorithms. Their computing efficiencies on the SIMD array processor are analyzed and compared in detail.

Key words parallel algorithm, adaptive optics, wavefront computation, simple instruction and multiple data (SIMD) architecture.