维普资讯 http://www.cqvip.com

计算机科学 1998Vol. 25№. 6

SYSTOLIC 计算*)

Systolic Computing

杨超峰 胡铭曾 张 毅

(哈尔滨工业大学计算机科学与工程系 哈尔滨 150001)

摘 要 The theory of systolic computing is twenty years old now. Many design methodologies, techniques and systolic arrays are proposed. In this paper, we discuss the nature of systolic computing and present a framework for all of study on systolic computing. As a result, all of papers in the literature can be analyzed systematically, then places where study is incomplete are exposed, especially two which are not received any attentions at all.

关键词 Systolic Computing, Non-Von Architectures, Processor Arrays

1. SYSTOLIC 概念

在 70 年代末,Kung 提出了一个新的非冯式体系结构,称为 SYSTOLIC 结构[1,2]。这种计算结构对于规整的算法,能获得极高的吞吐率。其特点是:大量的处理单元互连成阵列(因而人们通常称 SYSTOLIC 结构为 SYSTOLIC 阵列),只有周边处理单元能访问存储器,数据从周边处理单元输入,然后沿着一定的路线在阵列内传输,形成一个个数据流。处理单元位于这些数据流的交叉点上,它们利用传输到的数据完成相应的运算。有些数据流中的数据不变,使得数据被该流通过的各处理单元重用,有些数据流中的数据改变,它们流出阵列的数据就是计算结果,而在阵列中的数据是部分结果。

由上可得 SYSTOLIC 计算的本质:1)一个数据能被其所在流通过的所有处理单元使用,不象传统的 Von Neuman 结构,每次使用该数据都要访问存储器,这样一来,大大消除了传统结构的存储器瓶颈问题(Kung 称之为 Von Neuman 瓶颈);2)阵列中的处理单元被数据流连成一条条超级流水线,数据不变的流水线高效地输送数据,数据改变的流水线完成所需的运算,因而这种结构具有高度的并行性以及流水性,能获得极高的计算性能。将 SYSTOLIC 结构作一个类比,它就像人体的血液循环系统,心脏(存储器)促使血液(数据)在体内(阵列)各部分循环。这就是 Kung 得到 SYSTOLIC(心收缩)这一术语的原因。

SYSTOLIC 概念一提出,就吸引了许多研究者的兴趣,除了上述提到的该概念所蕴涵的理论价值外,还有下面一些重要原因:

· 大量的计算密集型算法,如数字信号处理,数值计算,模式识别等等,算法结构很规整,并行度很高,易于用 SYSTOLIC 阵列实现。

· SYSTOLIC 阵列是作为面向算法的专用计算部件提出来的,阵列是算法的直接映射,结构简单规整,易于 VLSI 实现。随着半导体技术(VLSI/WSI)的进步,一个 SYSTOLIC 阵列可以在芯片级实现,为实现 SYSTOLIC 概念蕴涵的高度流水性提供了支持。

· 设计专用芯片的成本很高,但是,FPGA 的出现为解决该问题提出了一个途径。我们可以设计一个多 FPGA 系统[3],作为运算加速器附加于通用计算机,编译器将算法中计算密集的部分提取出来,用 SYSTOLIC 阵列实现,而 SYSTOLIC 阵列由后端的多 FPGA 系统实现。FPGA 的可重构特性避免了设计专用芯片的高额费用[4]。这种思想就如同为通用机提供了一个硬件计算资源库,类似于软件设计中的软件库。

*)本文的研究得到国家 863 计划的支持。杨超峰 博士生,主要研究方向是高性能计算,高性能体系结构等。胡铭曾 博导。张毅 学士。

维普资讯 http://www.cqvip.com

2. SYSTOLIC 阵列设计时涉及的问题

可以用 SYSTOLIC 阵列实现的算法为一致依赖算法(URE)以及可以转化为一致依赖算法的线性依赖算法(ARE)。SYSTOLIC 阵列设计的基本过程可以简要地描述如下:线性依赖算法转化为一致依赖算法,一致依赖算法对应的计算图与阵列在拓扑结构上是"同构"的,因而可以通过对一致依赖算法的计算图进行变换,映射到阵列上,最后由电子工程师将得到的阵列用 VLSI 实现。

在本节将总结设计 SYSTOLIC 阵列时要涉及到的问题。我们的意图是为所有关于 SYSTOLIC 阵列的研究建立一个框架,从而可以对文献进行系统地分析,以及揭示出目前研究还不完善的地方。

2.1 通用阵列还是专用阵列好?

有些研究者作了设计通用机的尝试,如 Warp[5],Matrix-1,SLAPP 等,这些设计都拥有高性能的处理单元,处理单元内设有大容量局部存储器,以保证通用性。在处理单元间有高带宽的互连通道,以支持细粒度的流水。但是,高性能的处理单元意味着阵列是高性能单机的互连,而不能在芯片级实现,其后果是设计与制造费用很高,处理单元数目受限,如 Warp 仅有 10 个。从本质上看,这些机器是在传统多机系统中引入 SYSTOLIC 概念的数据重用而得。其运行机制与采用宏流水的运行机制的传统多机系统类似。

至于专用阵列,利用 VLSI 技术,能在芯片级实现,处理单元间传输数据以字(甚至位)为基本单位。这使得计算粒度极小,通常为一个元操作,如乘、加,从而可以获得计算能力极高的芯片级阵列,如我们设计的 ME 芯片[6],它完成动态图象编码的运动估计,有 256 个处理单元,吞吐率为 10GOPS。不过,专用性可能导致需求量小,而芯片的设计成本很高,使得性能价格比低。

归纳起来,通用阵列和专用阵列的根本区别在于数据重用度。Kung 提出 SYSTOLIC 概念时仅指出了该概念的本质在于数据重用,但并没有考虑要达到何等程度,才能发挥该概念的计算潜力。通用阵列的数据重用度低限制了阵列的计算能力,实际上,它能否比传统的多机系统更有效是值得怀疑的,也许它可由传统的多机系统来替代。如何来评价专用阵列还是通用阵列好是一个非常有指导性的工作,但是目前还没有这方面的尝试。该问题将在第 3 节作进一步的论述。

2.2 算法转化技术

已有技术将线性依赖算法转化为一致递归算法[7],其基本原理是选取一组基向量,使得所有依赖向量可以表示为它们的非负整线性组合。其实,设计 SYSTOLIC 阵列时通常要进行的数据广播消除[8]是转化技术的一个特例。可以将算法转化为多阶段算法,其每一阶段都是一致递归算法[8]。

但是,基向量组的选取不是唯一的,该如何选取最优的一组呢?目前已有系统化的方法将一致依赖算法映射成阵列,使得阵列的设计自动化,但是还没有一个系统化方法能包含这些转化技术直接将线性依赖算法映射成阵列,更不用说系统化地选择最优向量组了。

2.3 映射技术

映射过程就是要确定两个函数:分配函数 Π 和调度函数 S,分配函数 Π 将算法对应的计算图中所有节点分配到相应的运算单元,调度函数 S 确定算法对应的计算图中各节点的执行时间。

这两个函数的选取要满足一些条件:首先要保证映射的正确性,即先序条件和计算无冲突条件,具体实现时可能还要满足一些额外条件;另外,还要进行最优化设计,通常的最优化测度为计算时间 T、阵列大小 A、AT 及 AT^2 等。通常将这两个函数限定为线性函数,其理由是:首先,有两个数学基础支持线性变换,运算上有线性代数,最优化设计上有线性规划;另外,线性变换也是很有效的,它很早就在多机系统中用作调度函数,也已证明线性调度比最优的自由调度仅差一个常数。后来的程序重构理论也一般采用线性变换[10]。另外,线性变换可将卷积、矩阵乘等算法的所有人工 SYSTOLIC 阵列统一起来。线性变换的一个简单扩展是模数变换[11],它能设计出含有环路的阵列,如 Torus,但失去了数学基础。

选取这两个函数需要大量的计算来判定它们是否满足条件,仅靠直觉是不够的,而人脑对计算并不善长,针对一个特定的算法,依靠设计者的一些独特发现,可能能设计出好的阵列。不过,最好有系统化的设计方法来使设计自动化。目前,比较典型的系统化方法有三个。

1)数据依赖法(DM)[12,13]:它将阵列的最优化设计归结为一个求解整数规划问题,即最优设计目标表示成一个 Π 和 S 的线性函数,对 Π 和 S 的约束条件表示为一组线性等式或不等式。整数规划是一个 NP 完全问题,为了提高求解速度,必须配合一些简化策略,如先确定 S 以减小变量个数,目标函数为

维普资讯 http://www.cqvip.com

凸函数时可以将整数规划归结为一个等价的线性规划问题等。

2)数据依赖图法[14]。它将抽象的计算图表示成具体的图形,利用人的直觉选择一个合适的调度向量S和投射向量P。其实,它与数据依赖法在本质上是一样的,因为S和P分别等价于数据依赖法的Π和S。不过,利用图形使设计过程直观。但是,人的直觉仅限于有限的几个简单的向量,不一定能得到最优解。特别是,三维以上的计算图无法表示成直观的图形。

3)参数法(PM)[15,16]:它类似于数据依赖法,不过,引入了三类参数(周期t、数据的传输速度V和数据分配)来替代Π和S,搜索最优解的复杂性为多项式时间。

目前SYSTOLIC阵列设计的一个潮流是低维阵列的设计,如已提出了一系列求解矩阵乘的线性阵列[17]。数据依赖法和参数法均能用来设计低维阵列,数据依赖图法对此无能为力。

2.4 大问题的划分技术

划分可以采用两种实现途径:1)重新设计求解任务的算法,使之成为一些小任务的集合。这种方法要依赖于特定任务,缺乏通用性,很难实现自动化。2)不重新设计算法,对原算法的计算图进行处理,大致有两类处理策略:首先是局部并行全局串行(LPGS)[12]:将计算图划分成块,每一块均可由阵列一次完成,然后按照正确的顺序逐块地扫描整个计算图。其次是局部串行全局并行(LSGP):映射原计算图得到一个虚拟阵列,再在各方向上选择一个压缩因子,将大的虚拟阵列压缩为小的实阵列,此时,实阵列的一个处理单元对应虚拟阵列的若干处理单元[18]。LPGS和LSGP已很完善,但是它们有局限性,需要研究新技术。这将在第3节讨论。

2.5 系统集成

SYSTOLIC阵列是整个系统的一部分,将它集成到整个系统并不是一件容易的事。因为SYSTOLIC阵列(特别是二维阵列)需要极高的I/O带宽。因而,要有有效的存储器子系统相配套,才能充分发挥SYSTOLIC阵列的计算能力。

SYSTOLIC阵列的研究工作集中于结构设计,关于系统集成的很少。文[19]是一个较完善的工作,其中提出了一个高性能的计算系统:整个系统由主存、通用处理器及协处理器三部分组成。协处理器是系统性能的关键,采用SYSTOLIC阵列作为其计算引擎。由于阵列和主存之间需要数据缓冲器来实现速度匹配,该文认为协处理器应该包含阵列和缓冲器两部分,即用VLSI实现阵列时,应该将阵列和缓冲器集成在同一芯片内,至于两者所占芯片面积的比重要遵从原则:与主存以固定的数据带宽互连,以利于系统集成。该文得出下面一些重要结论:

1)阵列所占的比重随着芯片面积的增大而减小,缓冲器占了芯片的大部分面积。例如,若一个PE相当于100字所占的面积,线性阵列在一个5M的芯片中实现仅占总面积的4%,二维阵列所占比率略大。

2)用同一大小的芯片实现,线性阵列的性能与二维阵列相当。例如,对于100字大的PE,线性阵列仅比二维阵列下降6%,对于5000字大的PE,下降也仅为26%。

上述结论是以矩阵乘为例获得的。以前的文献均认为对于矩阵乘,线性阵列需$O(N^2)$计算时间,而方阵为$O(N)$。这与上面的结论大相径庭。其原因是它们仅考虑了阵列,而没有考虑到由于缓冲器占用大量芯片面积,使得能实现的方阵很小。这意味着以往评价阵列的测度并不合理,这一方面还需进一步的研究。

2.6 时钟同步

时钟频率是决定系统性能的一个重要指标,但是时钟扭斜制约了其无限制地增大,随着VLSI工艺的进步,该问题将更加突出。SYSTOLIC阵列的研究者一般都忽略该问题,认为这是电子工程师的事情,事实上它与结构研究关系密切。首先,它可以在具体的电路设计前进行,另外,不同结构的阵列要采用不同的时钟同步策略。关于该问题比较重要的文献是[20],它从理论上证明了时钟扭斜会制约系统性能这一观点,给出了两个重要结论:线性阵列中时钟扭斜有上限;二维阵列中时钟扭斜无上限。关于线性阵列的结论是当前研究低维阵列热门的原因之一。如何设计出有效的时钟分发网络来克服二维阵列中的时钟扭斜仍是一个问题。

3. 未受重视的问题

上节中,在分析已有工作的基础上,提出了一些尚待进一步研究的问题。在本节,将讨论两个目前还完全未受重视的问题。

3.1 阵列模型

阵列的设计可看作计算图的拓扑结构到阵列的拓扑结构的变换过程。因为阵列的实现是受VLSI技术限制的,它的结构应该首先提出来,然后改造计

维普资讯 http://www.cqvip.com

算图的结构，使其能映射到阵列上。因而，确定一个合理的阵列结构是非常重要的，将影响阵列设计的整个过程。这个阵列结构就是所谓的阵列模型。

在文献中，有些模型是隐式体现在设计出的阵列中的。如：Kung 在早期的设计[1,2]中有些含有广播线，这种结构称为半 SYSTOLIC(Semi-Systolic)。

有些是显式讨论的，这些模型比较重要，因为它们是形式化给出的。Moldovan 建立了一个模型[12]、用三元组(G,F,T)表示，G 是阵列的拓扑结构，处理单元之间按近邻原则互连；F 是处理单元要完成的操作集；T 指定了处理单元进行特定运算及传输数据的时序。但是，该模型的定义不是非常严格，它仅明确限制了阵列的结构，而没对阵列的执行机制作任何规定，另外，对 F 和 T 也没有任何约束。参数法[15,16]采用的模型在阵列的执行机制和结构上作了进一步的约束，它限制拓扑结构是严格一致的及数据在阵列中只能直线传输。

上面是专用阵列的模型。通用机必然要采用不同的模型，但是目前也只有类似于传统多机系统的描述。

目前，阵列模型非常混乱，不同的研究者均依个人喜好采用自己的模型、多数是隐式的，其结果是关于阵列的研究很混乱。如：在任务调度时，文[21]考虑了通信延迟，计算与通信重叠、Cache 和存储器等因素，这些研究成果不能为专用阵列所用。

如何采用一些评价尺度、用模拟的方法对不同模型进行定量比较，进而提出一些模型，有充分的灵活性、高的计算能力及可实现性。这对 SYSTOLIC 计算的研究有重要的指导意义。

3.2 大问题的划分技术

LSGP 和 LPGS 处理后的计算图与原图维数相同，对于高维(四维以上)的问题，DM 和 GPM 原则上是可以进行超低维阵列设计的，但是超低维设计的阵列效率会很低。目前在文献中还没有一个具体的低维设计实例，这也说明了这一问题。一个解决方法是只用高维算法的某几维来设计阵列，利用该法有一些设计实例，但是，没有任何系统化方法能支持这一点，此时问题的根本点在于：该选取哪几维？我们认为可将数据重用度作为选择基准，程序重构技术[10]作为实现技术。 (下转第 69 页)

(上接第 8 页)

予了大力支持。政策上，DoD 的仿真与建模办公室(DMSO)专门成立了 AMG(体系结构管理组)负责 HLA 框架内有关标准的制定，以使 HLA 一开始就沿着健康的道路发展，DoD 明确规定今后军事领域内的仿真活动必须在 HLA 的框架下开展，已有的仿真系统要转换成 HLA 兼容方式；财政上，DoD 投入资金支持 HLA 方面的研究工作，并资助 HLA 核心 RTI 原型的开发。

新的技术往往首先应用于军事领域，HLA 也不例外，但 HLA 并不局限于军事领域，由于 HLA 从更高层次上理解仿真的要求和特点，它在结构和功能上是合理和完善的，能适应各类型仿真的发展，并支持它们之间的互操作和可重用。因此 HLA 在商业领域也有很好的应用前景，如娱乐业的网络游戏的开发等都可以应用 HLA 框架，事实上目前 DoD 正积极寻求与商业领域的合作[2]。可以说 HLA 将对未来仿真的应用与发展产生深远的影响。

参 考 文 献

[1] DMSO, High Level Architecture Baseline Defination, August 15 1996, http://www.dmso.mil

[2] J. S. Aronson, Approaches to Runtime Communications for Distributed Simulations, Proc. of 1997 Spring SIW

[3] J. O. Calvin, et al, An Introduction to the High Level Architecture (HLA) Run-Time Infrastructure (RTI), Proc. of the 15th DIS Workshop, 1996

[4] D. Fullford, Distributed Interactive Simulation: It's Past, Present, and Future. Proc. of 1996 Winter Simulation Conference, 1996

[5] R. Weatherly et al, Aggregate Level Simulation Protocol, Proc. of the 1991 Summer Computer Simulation Conference, Maryland, July 1991

[6] Steve Vinoski, CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments, IEEE Communications Magazine, (14) 2 1997

[7] 邱小刚，黄柯棣，HLA 中对象模型的分析，全国仿真会议，张家界，1997

[8] 童军，李伯虎，分布交互仿真技术的新发展，全国仿真会议，张家界，1997

维普资讯 http://www.cqvip.com

语流中录制合成音节单元的策略，对系统的合成质量有较明显的改进。

3.4 抽取有效语音单元

以音节为合成单位进行波形连接的文语转换方法，音节单元录制的质量对系统的总体效果有重大的影响，本系统采用了以下策略以保证语音库音节单元的质量，即重复录音，概率择优[6]。对于每个音节单元，录制8遍，在建立合成音节库时，选取其中一个作为音节单元，选取的策略如下：

a)舍弃时长小于所有该类语音单元平均时长80%的语音单元，这样处理的目的是，一方面，这些被删除的语音单元的音质往往不太好，另一方面，在合成时如果使用这些被删除的语音单元，在调整时长时，时长的增加有可能超过限定的时长增长因子(目标语音单元时长/原始语音单元时长)的阈值，造成语音信号的失真或不自然。在本系统中，时长增长因子设为1.22。

b)计算所有语音单元平均采样能量，舍弃平均采样能量大于该值的语音单元，这样处理可以减小在合成时声音的剧烈起伏振荡。

c)对于进行以上两个步骤后剩下的语音单元，通过试听和合成测试，最后，在每一类型语音单元选定一个作为系统使用的语音单元。

(下转第89页)

(上接第4页)

主要参考文献

[1] H. T. Kung et al., Systolic arrays (for VLSI), Sparse Matrix Proc. Society for Industrial and Applied Mathematics, 1978

[2] H. T. Kung, Why systolic arrays, IEEE Computer, vol. 15 Jan. 1982

[3] S. Hauck, Multi-FPGA Systems, PH. D Thesis, Dept. CS&E, University of Washington, 1995

[4] B. K. Fawcett et al., Reconfigurable Processing with Field Programmable Gate Arrays, in Inter. Conf Application Specific Array Processors, 1996

[5] M. Annaratone, et al., The Warp Computer: Architecture, Implementation and Performance, IEEE Trans. Computer, 36(12)1987

[6] 杨超峰，颜玲，傅宇卓，胡铭曾，支持MPEG-2标准的ME芯片设计，体系结构年会'97，1997.10

[7] W. Shang et al., On uniformation of affine dependence algorithms, IEEE Trans. Computer, 45 (7) 1996

[8] Y. Wong et al, Transformation of broadcast into propagation in systolic arrays, J. of Parallel and Distributed Computing, 14(2)1992

[9] J.-C. Tsay et al., Design of efficient regular arrays for matrix multiplication by two step regularization, IEEE Trans Parallel Distrib System, 6(2)1995

[10] W. Li et al., A singular loop transformation framework based on non-singular matrices, Int J. Parallel Programming, 22(2)1994

[11] H. J. Lee et al., Automatic generation of modular mapping, ASAP'96, 1996

[12] D. I. Moldovan et al., Partitioning and mapping algorithms in fixed size systolic arrays, IEEE Trans. Comput, vol. C-35, 1986

[13] W. Shang et al., On time mapping of uniform dependence algorithms into lower dimensional processor arrays, IEEE Trans. Parallel and Distributed Systems, 3(3)1992

[14] S. Y. Kung, VLSI Array Processors, Englewood Cliffs, NJ: Prentice Hall, 1988

[15] G.-J Li et al., The design of optimal systolic arrays, IEEE Trans. Comput. vol. C-34, 1985

[16] K. Ganapathy et al., Optimal Synthesis of Algorithm-specific Lower-Dimensional Processor Arrays, IEEE Trans. on Parallel and Distributed Systems, 7 (4)1996

[17] V. K. Prasanna Kumar et al., Designing linear systolic array, J. Parallel and Distrib. Computing, vol. 7, 1989

[18] A Darte, Regular partitioning for synthesizing fixed-size systolic array, J. VLSI Integration, vol. 12, 1991

[19] K. Ganapathy, et al., Design a scalable processor array for recurrent computations, IEEE Trans. on Parallel and Distributed Systems, 8(8)1997

[20] A. L. Fisher et al., Synchronizing large VLSI processor arrays, IEEE Trans. Computers, C-34(8) 1985

[21] J. Teich, et al., Scheduling of partitioned regular algorithms on processor arrays with constrained resources, ASAP'96, 1996