粗粒度可重构阵列中的蚁群优化映射

周理，刘冬培，刘衡竹，陈书明
（国防科学技术大学计算机学院）

摘要: 粗粒度可重构阵列（CGRA）结构兼具高性能和灵活性的特点，近年来应用广泛。CGRA 的性能发挥依赖于有效的映射算法开发程序的并行性。本文提出了一种 CGRA 上的应用程序映射方法，它采用最大-最小蚁群系统（MMAS）方法将有向无环数据流图（DAG）映射到 CGRA 上。本文进一步研究了映射算法的优化，在减少映射时间的同时保证解的质量。与其他启发式方法相比，本文提出的算法能够在更少的映射时间内得到更好的结果。

关键词：粗粒度可重构；应用映射；蚁群优化
中图分类号：TP332.1

Ant Colony Optimization for Application Mapping in Coarse-grained Reconfigurable Array

Zhou Li, Liu Dongpei, Liu Hengzhu, Chen Shuming
(Computer School, National University of Defense Technology)

Abstract: Coarse-grained reconfigurable array (CGRA) architecture has become popular because of its performance and flexibility. The efficiencyof CGRA relies on an efficient application mapping algorithm to exploit parallelisms. In this paper, we proposed the min-max ant colony system (MMAS) algorithm to map data acyclic graph (DAG) onto CGRA. Optimization of MMAS is studied to reduce mapping time while maintaining the quality of solutions. Comparisons with other heuristic algorithms show that our approach obtains better results in less mapping time.

Key words: application mapping; ant colony optimization; CGRA

0 引言

粗粒度可重构阵列（Coarse-grained Reconfigurable Array, CGRA）是一种高效灵活的硬件体系结构，适合于计算密集型的应用。在软件无线电、多媒体处理等数字信号处理领域，CGRA 获得了大量应用[1]。这些领域的应用包含大量重复的计算，因此 CGRA 可以通过大量的处理单元（Processing Element, PE）实现程序的加速。另外，CGRA 的处理单元都是可重构的。 PE 和及其 PE 互连网络都可以通过加载上下文缓存来完成不同的功能。与现场可编程门阵列（Field Programmable Gate Array, FPGA）不同，CGRA 中的重构是字级的（word-level）而不是比特级的（bit-level）。CGRA 中的功能单元因此能够针对特定领域的应用程序进行优化，以 CGRA 构建的系统不但可以获得硬件般的性能，同时还具备软件般的灵活特性。

如何将应用程序映射到 CGRA 上影响到系统诸多性能，例如程序执行延时、功耗等等。通常，应用程序是以有向无环数据流图形式（Directed Acyclic Graph, DAG）表示的，图中的节点代表了操作，节点之间的边代表了数据的依赖关系。映射就是在资源约束条件下确定每一个节点将在哪个 PE、什么时候开始执行。这个问题已经证明是 NP 问题。目前有两种程序映射方式：时域映射和空域映射[2]。在空域映射中，PE 的配置字是保持不变的，这样就要求应用程序足够简单，在硬件资源的限制之内。应用程序通常需要划分得足够小。而在

基金项目：高等学校博士学科点专项科研基金（No.20094307110009）
作者简介：周理（1986），男，博士生，主要研究方向：微处理器设计，数字信号处理
通信联系人：陈书明（1961-），男，教授，主要研究方向：微处理器设计. E-mail: smchen@nudt.edu.cn

时域映射中，PE 每个周期都不断变换配置字，时间轴上足够的空间能够容纳大规模的应用。本文提出了一种基于最大最小蚁群系统（Min-Max Ant System, MMAS）的时域映射方法，将 DAG 映射到 CGRA 上。该方法用于优化程序在 CGRA 上的执行时间。与其他算法相比，该方法能够减少映射时间，获得较好的结果。

1 基于最大最小蚁群系统的 CGRA 映射

给定一个应用程序核心，可以用图 $G=<V,E>$ 来表示，每个 $v_i \in V$ 代表了应用程序中的一个操作。每个 $e_i=<p,q>\in E$，表示了操作 q 会用到操作 q 的结果。而目标 CGRA 系统，也可以采用一个图 $C=<P,L>$ 来表示，$p_i \in P$ 代表了 CGRA 中的一个 PE。每个 $l_i=<u,v>\in L$，表示了 PE u 可以通过公共寄存器文件或互连网络使用 PE v 的结果。映射就是要找到两个函数：$M_v:V\to P$ 和 $M_t:V\to N^+$。这两个函数决定了操作放到哪 PE 上执行和操作将在何时执行。一个合理的映射应当不致使 CGRA 中的 PE 资源冲突。同时对于 $e_i=<p,q>\in E$ 都有合适的路径将数据从源 PE 传递到目的 PE。

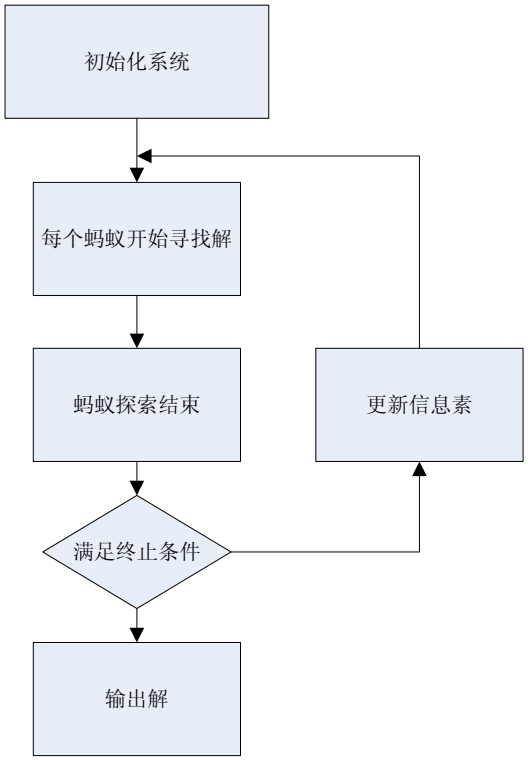

图 1 蚁群算法的流程图

Fig.1 The procedure of ant colony optimization

蚁群算法在调度问题上已经被证明优于遗传算法（Genetic Algorithm，GA）和模拟退火算法（Simulated Annealing，SA）[3]。在算法当中，蚂蚁被放置到一个起点并开始逐步构建解。在每个阶段，蚂蚁有个‘可见’的下一步骤集合，每只蚂蚁将从自己的这个集合中按‘能见度’挑选一个作为自己移动的下一步。一个步骤的‘能见度’与这个步骤所能带来的好处相关。如果做出某个步骤所产生的好处越多，这个步骤被选中的概率也就越大。蚂蚁在做出选择的同时释放信息素，信息素是一种全局的启发因素，能够避免蚂蚁陷入局部最优。MMAS 是蚁群算法的一种改进[4]，在 MMAS 中，只有找到了当前最优解的蚂蚁才会释放信息素，

MMAS 还限制了每条路径上的信息素上下限。这样改进能够加速算法收敛。避免算法过早停滞。蚁群算法的主要步骤如图 1 所示。

在 CGRA 映射问题中，所有候选的步骤构成一个集合 $S=\{s=(v,p)\,|\,v\in V,p\in P\}$，$s_i=(v_i,p_i)$ 表示将 DAG 中的操作 v_i 映射到 CGRA 的 p_i 上，最初的候选集合可以表示为 $S=\{s=(v,p)\,|\,\forall v'\in V,<v',v>\notin E\}$。一旦蚂蚁做出决定选择了候选集合中的某个元素 s_i，候选集合将相应地更新，s_i 将加入已选集合 D 中（D 初始化为空集）。候选集合的更新按照 $S=(S/\{s_i\})\cup\{u=(v,p)\,|\,\forall v'\in V,<v',v>\in E\rightarrow\exists p'\in p,<v',p'>\in D\}$ 进行，即当 DAG 中的某个节点的全部父亲节点已经被映射，那么该节点就可以被映射。在第 d 个阶段，选中某个 s_i 的概率可以如下计算：

$$p_{d,s_i}=\frac{\tau_{d,s_i}^{\alpha}\bullet\eta_{d,s_i}^{\beta}}{\sum_{s_j\in S}\tau_{d,s_j}^{\alpha}\bullet\eta_{d,s_j}^{\beta}}$$

其中 τ_{d,s_i} 是上次探索的蚂蚁留下来的信息素，η_{d,s_i} 是本阶段选择 s_i 的所得到的好处。α 和 β 是全局启发因子和局部启发因子的重要程度。所有蚂蚁完成解的探索后，信息素将按照 $\tau_{d,s_i}(t+1)=\rho\tau_{d,s_i}(t)+\Delta\tau_{d,s_i}$ 更新。当中 ρ 是蒸发因子，表示信息素蒸发的速度。$\Delta\tau_{d,s_i}$ 不为 0 当且仅当 s_i 是当前最优解的某个步骤。

针对步骤 $s_i=(v_i,p_i)$，本地启发因子 η_{d,s_i} 的计算与何时 v_i 能够在 PE p_i 上最早执行的时间有关。 这个时间越早，η_{d,s_i} 的值就越大。可以通过下列公式计算。

$$\eta_{d,s_i}=\frac{1}{startTime(s_i)}$$

$$startTime(s_i)=\max_{<v',v_i>\in E}\{M_t(v')+t(v',p')+route(M_v(v'),M_t(v'),p_i)\}$$

$startTime(s_i)$ 是由 v_i 的父亲节点的执行时间以及数据从生产者路由到消费者的时间决定的。$route(M_v(v'),M_t(v'),p_i)$ 函数采用迷宫路由策略（Maze Route）从 PE 资源占用表中寻找一条路径并返回路径的延时。迷宫路由是一种宽度优先搜索的方式，它从数据源节点所在 PE 的时间槽开始搜索，直到找到一个目标节点所在 PE 的空闲时间槽。当存在两条以上的路径需要路由时，我们优先路由时间要求紧迫的路径。这样能够获得路由延时的均衡。图 2 给出了 η_{d,s_i} 计算的一个例子。

图 2a 中给出了一个简单的 CGRA 和 DAG，图 2b 中，A 和 B 已经分别被映射到 PE1 和 PE3 上，并且占用时刻 1 和时刻 2。对于 $s_i=(C,PE3)$，需要计算 η_{d,s_i}，即评估将 C 映射到 PE3 上能带来的好处。由于 C 存在两个数据依赖，由于 A 开始时间早于 B，所以首先对 $<A,C>$ 进行迷宫路由，如图 2c 所示，数据最早能到第 4 时刻到达。对 $<B,C>$ 的迷宫路由如图 2d 所示，最早能在第 3 时刻到达。因此这个例子中 $\eta_{d,s_i}=\frac{1}{4}$。类似的，我们可计算得到 $\eta_{d,(C,PE1)}=\frac{1}{4}$ 和 $\eta_{d,(C,PE2)}=\frac{1}{3}$。

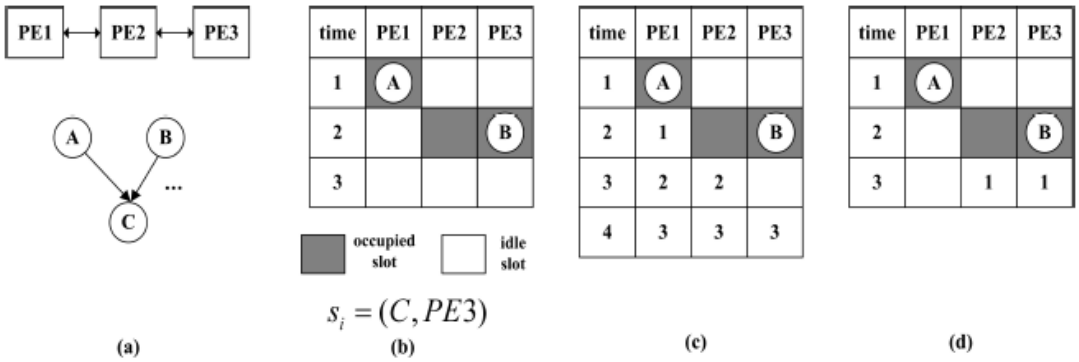

图 2 η_{d,s_i} 的计算方法示例 （a） CGRA 和 DAG 实例（b）资源占用表（c）路由 A 到 C（d）路由 B 到 C

Fig. 2 An example of η_{d,s_i} calculation (a) instance of CGRA and DAG (b) resource reservation table (c) route from A to B (d) route from B to C

2 算法优化

蚁群算法的一个缺点就是收敛时间长，即使对于改进的 MMAS 也是仍然存在。在 CGRA 映射中，需要用到迷宫路由，这样增加了蚂蚁的探索时间，使得算法的运行时间更长。一种可行的优化方案是舍弃一些很大概率导致结果差的步骤，从而减少对解空间的搜索。在文献[5]中，已经说明了 DAG 中节点的亲缘关系会极大地影响映射结果。在本文设计的算法中，我们考虑节点的两种类型亲缘关系：父子关系和共有子孙关系。一种非常直观的结论是，如果我们将具有亲缘关系的两个节点映射到足够近的 PE 上，那么容易得到好的映射结果。

记节点 A 和 B 之间是否有父子关系为 $AF_0(A,B)$，如果 $AF_0(A,B)=1$ 则表示他们之间有父子关系，反之 $AF_0(A,B)=0$。我们采用变量 $AF_i(A,B), i>0$ 来表示节点 A 和 B 在第 i 层子代中共有的子孙节点数目。这样我们可以定义亲缘关系的度量值如下：

$$affinity(A,B)=\sum_{i=0}^{g} 2^{g-i} \times AF_i(A,B)$$

g 参数表示了算法中需要考虑的 DAG 层数深度，父子关系是亲缘关系中影响最大的，而层数越深，共有子孙节点对亲缘关系的影响越小。这一点与现实中的亲缘关系是相同的。同时，我们还定义两个 PE p 和 q 之间的距离 $dis(p,q)$，它表示这 CGRA 体系结构中两个 PE 之间的跳步数。有了亲缘关系的度量和 PE 距离度量，我们就可以定义两个步骤 s_i 和 s_j 之间代价 $co(s_i,s_j)=affinity(v_i,v_j)^{dis(p_i,p_j)}$。在面对多个候选的步骤时，我们计算他们每一个的惩罚 $penalty(s_i)=\sum_{s\in D} co(s_i,s)$，即 s_i 与所有已经选择了的 s 的代价总和。

为了减少映射算法的运行时间，一些惩罚值高的候选步骤将被舍弃。算法中给定一个惩罚值上限 MP，所有超过这个值的步骤不纳入蚂蚁的搜索范围。因为这些惩罚值高的选择往往会造成差的解。通过实验确定，合适的 MP 对于解的质量影响不大，但是能够极大地减少映射时间（大于 50%）。

3 实验结果

我们在一种高效的 CGRA 结构上评估了提出的映射算法。这种结构由多个可重构的计算簇

组成，并由 mesh 网络相连。每个可重构的单元包含 4 个普通 PE，记为 CPE0~3，一个共享的 PE 记为 SPE。 普通 PE 中包含简单的逻辑和数字运算，并且是常用的运算。共享的 PE 中包含复杂的运算操作，或是不常用的运算操作。CPE 和 SPE 都有配置缓存，通过和 DMA 交换外部数据和上下文数据。如图 3 所示，这种结构的 CGRA 消耗更少的面积并且能够更高效地完成运算。

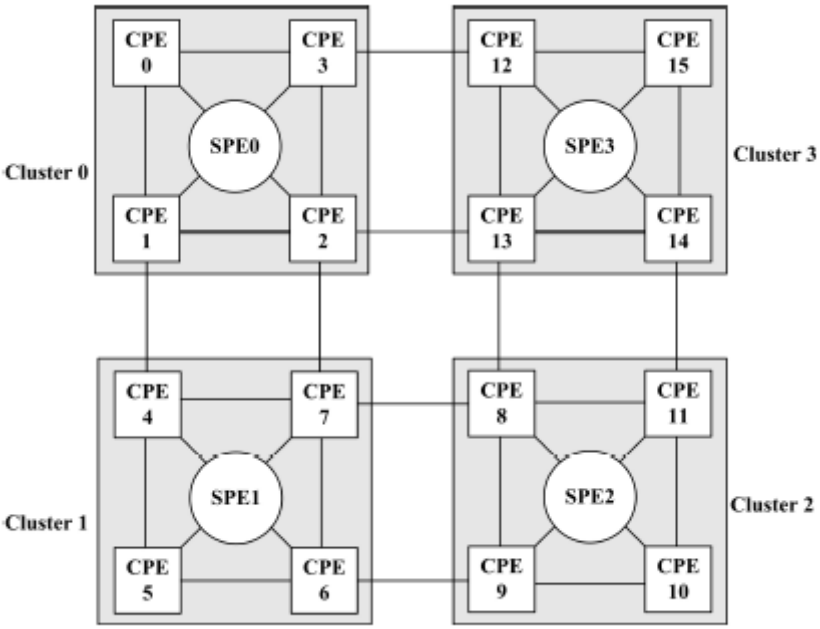

图 3　目标 CGRA 结构图

Fig. 3　target CGRA architecture

我们从 5 个现实生活中的应用中抽取了 DAG 用于实验，包括 3 个无线通信应用和 2 个多媒体应用：WCDMA 信道解码、MIMO-OFDM 解码、LTE 信道解码、H.264 解码、MPEG4 解码。DAG 的总数有 415 个，大小从 49 个操作到 713 个操作不等。为了评估性能，我们将 MMAS 映射同 SA 和 GA [6]中做了比较。 同时还采用 ILP 运算得到的值作为参考。表 1 为不同映射算法所获得解的质量。

表 1　映射质量对比，用 CGRA 上的执行周期衡量

Tab. 1　Comparison of solution’s quality measured by CGRA cycles

应用核心	*ILP*	*SA*	*GA*	*MMAS 未优化*	MMAS 加优化
WCDMA 信道解码	1431	1695	1608	1509	1516
MIMO-OFDM 解码	3655	4272	4136	3880	3902
LTE 信道解码	4278	5063	4751	4560	4589
H.264 解码	5692	6624	6394	6021	6074
MPEG4 解码	4936	5793	5527	5207	5237
平均损失（相对 ILP）		17.28%	12.26%	5.93%	6.63%

注：ILP 有可能在 3 天之仍内仍然无法获得结果，表中给出的是 ILP 运行能够得到的最好结果

从表中可以看到，MMAS 算法获得了较好的结果，相对 ILP 获得的最优结果，MMAS 损失仅为 5.93%，分别比 SA 和 GA 要高出 11%和 6%。由于加优化过后的 MMAS 在探索空间上减小了，限制了一部分搜索的随机因素，得到的解性能略有下降（不到 1%）。

表 2 给出了算法的运行时间比较。可以看到，没有优化的 MMAS 运行时间与 SA 或 GA 相当，但是经过优化后的 MMAS 运行时将显著减少，约 70%的映射时间被节省下来，而优化后的 MMAS 对于解的质量基本没有影响。

表 2　运行时间对比(单位：秒)

Tab.2　Comparison of running time (sec)

应用核心	*SA*	*GA*	*MMAS 未优化*	MMAS 加优化
WCDMA 信道解码	80716	91531	95768	20559
MIMO-OFDM 解码	234571	249522	255421	67360
LTE 信道解码	315298	357457	328516	79481
H.264 解码	373849	375630	378563	109015
MPEG4 解码	316895	325744	326792	91346

注：ILP 的运行时间不确定，有的甚至不能再 3 天之内获得结果。故没有列出

4 结论

本文给出了一种基于 MMAS 的 CGRA 映射算法，它将 DAG 映射到目标 CGRA 上。对于给的 CGRA 结构，该算法能够在合理的时间内找到优化的映射结果。实验表明，我们的算法能够获得优于 SA 和 GA 的结果，并且映射时间更少。针对 MMAS 映射做的优化非常有效，在减少算法运行时间的同时对结果几乎没有影响。

[参考文献] (References)

[1] K. Choi. Coarse-Grained Reconfigurable Array: Architecture and Application Mapping[J]. IPSJ Transactions on System LSI Design Methodology, 2011, 4, 31-46.

[2] Dorigo, M., Maniezzo, V., Colorni, A. Ant system: optimization by a colony ofcooperating agents[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B:Cybernetics, 1996, 26, 29-41.

[3] Lee, G., Choi, K., Dutt, N.D. Mapping Multi-Domain Applications Onto Coarse-Grained Reconfigurable Architectures[J]. IEEE Transactions on Computer-Aided Design,2011, 30, 637-650.

[4] Sttzlea, T., Hoosb, H.H. MAX-MIN ant system[J]. Future Generation Computer Systems,2000, 16, 889-914.

[5] Park, H., Fan, K., Kudlur, M., Mahlke, S. Modulo graph embedding: mapping applications onto coarse-grained reconfigurable architectures[C]. Proceedings of the international conference on Compilers, architecture and synthesis for embedded systems, 2006, pp. 136-146

[6] Braun, T.D., Siegel, H.J., Beck, N., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., Yao, B., Hensgen, D. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems[J]. Journal of Parallel and Distributed computing, 2001, 61, 810-837.