可重构系统中软硬任务划分方法研究

张 丹 赵荣彩 单 征 韩 林 瞿 进
(国家数字交换系统工程技术研究中心 郑州 450002)

摘 要 软硬件任务划分是可重构系统开发过程中的重要设计步骤,其划分结果直接影响到可重构系统的性能。目前的软硬件任务划分技术大多只考虑了对应用程序或算法的划分结果,忽略了FPGA在配置和通信时的开销,从而导致实际应用效果不理想。介绍了一种基于性能评估的软硬件任务划分方法,即通过对FPGA计算开销、配置开销、通信开销的预评估测试,结合改进的模拟退火算法得出可重构系统中的软硬任务划分结果。实验结果表明,该划分方法具有较好的划分效果和算法收敛速度。

关键词 可重构,FPGA,性能评估,并行,模拟退火
中图法分类号 TP301 **文献标识码** A

Research on Hardware/Software Task Partitioning for Reconfigurable System

ZHANG Dan ZHAO Rong-cai SHAN Zheng HAN Lin QU Jin
(China National Digital Switching System Engineering and Technology Research Center, Zhengzhou 450002, China)

Abstract As the crucial design steps in reconfigurable system development process, the results of the hardware/software task partitioning directly affect the performance of reconfigurable system. Most of the current hardware/software task partitioning only consider the partitioning results of the application or the algorithm, ignoring the overheads of the FPGA configuration and communication, resulting in the actual result is not satisfactory. This paper presented a hardware/software task partitioning methods based on performance evaluation, which generates the hardware/software task partitioning results of reconfigurable system by evaluating and testing the overhead of computing, configuration and communication of FPGA, combined with improved simulated annealing algorithm. The experimental results show that the partitioning method has good accuracy and faster convergence speed.

Keywords Reconfigurable, FPGA, Performance estimation, Parallel, Annealing simulation

1 引言

近年来,基于FPGA的可重构计算技术逐渐成为高性能计算技术研究中的热点。FPGA能够同时兼具ASIC的执行效率和通用处理器的灵活性,具有高性能、低功耗等优势,能够满足人们对高效能计算的需求,在许多应用领域都有着广阔的应用前景[1,2]。但是并非所有应用程序在诸如FPGA等可重构器件上执行都能够获得理想的加速效果,特别是当FPGA作为独立的协处理器部件、采用松耦合连接方式时,通用处理器与FPGA之间的通信延时急剧增加。即使应用程序在FPGA上能够获得较好的执行速度,但是由于受硬件配置时间和通信延迟的影响,应用程序在可重构计算系统中执行的整体性能反而可能降低。

因此,可重构系统中的任务划分需要充分考虑可重构系统中可能会影响系统执行性能的多方面因素,对系统执行性能进行综合评估。本文介绍了一种基于执行性能预评估测试的软硬任务划分方法,并验证了该划分方法的有效性。

2 相关研究

软硬件任务划分是可重构系统设计开发的关键步骤,其任务是在满足某些约束条件下,将系统功能最优化地映射到相应的软硬件结构上。根据映射的目标系统结构的不同,软硬件任务划分问题可以分为双路划分(bi-partitioning)和多路划分(multi-way partitioning)。双路划分是指可重构系统中只含有一个通用处理器和可重构器件的情况,其应用范围最为广泛,同时也是可重构系统中任务划分的基础[3]。

文献[4]中提出了一种针对动态重构计算系统的任务划分和调度方法。这种方法将大规模应用分解成若干个具有相互约束关系的子任务,采用基于遗传算法和爬山算法的融合算法实现了可重构系统中的软硬件任务划分,文中最后对其任务划分方法进行了仿真分析。文献[5]中提出了一种可重构系统中的软硬件计算任务划分算法,该算法能够对CPU和

到稿日期:2011-04-16 返修日期:2011-07-20 本文受国家高技术研究发展计划(863)项目(2009AA012201),国家工信部核高基重大专项(2009zx01036-001-001)资助。

张 丹(1981—),男,博士生,主要研究领域为高性能计算,E-mail:andrew-zd@163.com;赵荣彩(1957—),男,博士,教授,博士生导师,主要研究领域为先进编译技术、网络安全;单 征(1977—),男,博士,副教授,主要研究领域为高性能计算、网络安全;韩 林(1978—),男,博士,讲师,主要研究领域为并行计算;瞿 进(1981—),男,博士生,主要研究领域为高性能计算。

FPGA 等数字电路进行执行性能评估，并通过一定的优化给出任务划分结果，但该算法不适合复杂应用程序。文献[6]中采用了一种层次化的模型对 FPGA 的硬件结构进行描述，然后依据基于链式调度的软硬件代码划分方法对应用程序进行划分，并在划分过程中采用了一种基于 IP 核调用的代码转换机制，减少了应用程序代码向硬件描述语言转换时的复杂性。这种任务划分方法能够将应用程序的执行时间平均减少 20%以上。这种方法极度依赖 IP 核的完备程度，具有一定的可扩展性，但灵活性较差。文献[7]中提出了一种基于单处理器结构的嵌入式系统软硬件划分模型，该模型通过构造目标函数和系统条件约束，提出了基于克隆选择算法的软硬件划分方法，但是文中忽略了所划分出的任务之间的通信延迟和调度开销，不符合实际应用中的需求。

3 基于执行性能评估的软硬件任务划分方法

不同类型的应用程序本身在算法结构和计算特征等方面都是极度异构的，因此若想得到一种较优的软硬件任务划分结果，则必须有一种在不破坏应用程序原有算法结构的基础上能够充分反映应用程序计算特征（如并行性、通信强度等）的表示方法。为此，本文采用了任务流图来刻画应用程序的算法结构和计算特征，并以此作为软硬件任务划分的基础。

3.1 应用程序执行特征提取

定义 1 任务流图 TFG(Task Flow Graph)是一个五元组，即 $TFG=(V_i, E_{ij}, V_i^P, V_i^M, V_{ij}^C)$，其中，$V_i$ 表示图中的节点；E_{ij} 表示连接节点之间的有向边，$E_{ij} \subset V_i \times V_j$，表示任务之间的依赖关系；$V_i^P$ 表示任务 i 的计算量，在初始状态下的任务流图中表示为该任务的串行计算量；V_i^M 表示任务 i 的存储需求（即该任务处理过程中待处理数据量）；V_{ij}^C 为边的权值，表示任务 V_i 和 V_j 之间的通信开销。

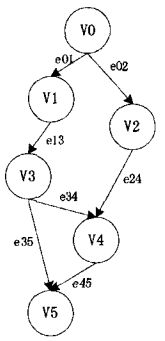

图 1 任务流图示例

任务流图是一种应用程序功能和特点的直观表示方法，也是进行软硬件任务划分的重要依据。初始任务流图的生成过程也是对应用程序功能特点和计算特征的分析和抽取过程。在初始任务流图生成过程中，文中借鉴了 Polychronopoulos[8] 提出的 HTG(Hierarchical Task Graph)的方法，并根据任务划分的需求对 HTG 进行了扩展。HTG 生成的基本思想是对应用程序中的强连通区域进行检测，并形成具有关联关系的层级结构。在 HTG 中包含有开始节点、结束节点、基本块节点、call 节点（对应一个子程序）、循环节点（对应一段嵌套循环程序）、复合节点（对应一个 HTG 子图）。

初始任务流图生成过程如下：

(1)采用 Polychronopoulos 算法生成应用程序的层次任务图 HTG；

(2)通过程序插桩计算出 HTG 中各个节点的 V_i^P 和 V_i^M；

(3)计算节点内的全局输入、输出参数，得出节点的 V_{ij}^C；

(4)在 HTG 中添加各个节点的属性参数，生成初始 TFG。

在应用程序算法结构比较简单，或者控制流、数据流关系明确的情况下，可以直接生成 TFG，不需要进行 HTG 表示。

3.2 基于可重构资源限制的任务粒度调整

任务粒度调整过程主要是针对可并行任务而言。任务粒度确定的过程需要考虑任务流图中的任务节点对计算资源、存储资源以及通信资源的需求和具体执行器件之间的相互制约关系，通过对每个任务执行时是否值得进行数据和计算的重复进行衡量，可以减少通信和可重构器件的初始化配置延迟等开销。首先考察初始 TFG 中所有循环结构的可并行性（包括 TFG 子图中包含的循环结构节点），分别将其划分至并行节点集合和串行节点集合。

然后再考察可并行任务集合中的节点与其相邻节点之间的通信开销。如果其相邻任务节点也为可并行循环节点，且两个任务之间的通信时间为两个任务中执行时间较长者的两倍以上，即 $V_{ij}^C \geqslant 2\times \mathrm{Max}(V_i^P, V_j^P)$，则对两个任务的逻辑资源需求和存储资源需求进行衡量，如果可重构器件硬件逻辑资源和存储资源均能够满足这两个任务的需求总和，则将两个任务进行合并。若 $V_{ij}^C < 2\times \mathrm{Max}(V_i^P, V_j^P)$，或者相邻两个任务的逻辑资源需求总和大于可重构器件资源限制，则不进行任务合并（也可以根据可重构器件的硬件属性设定阈值，本文根据多次实验结果，采用设定阈值为 2）。在这里需要特别说明的是，如果相邻两个任务在合并后的存储资源需求大于可重构器件存储资源的限制，则优先考虑采用对时序上较为靠后的任务进行循环分块，若经过循环优化后仍不能够满足资源限制，则不能够进行任务合并。完成任务合并之后按照任务节点合并的具体情况，重新计算并修正 TFG 中相应任务节点的各个属性参数。

3.3 软硬任务初筛

在可重构系统中，由于可重构器件（FPGA）与 CPU 在硬件结构上的差异，决定了可重构器件适合于执行应用程序中控制操作相对简单、具有规则的数据访问操作，以及计算量较大的可并行任务。而 CPU 则适合于执行那些控制操作较为复杂、计算量小，或者其它一些无法在可重构器件上执行的任务。应用程序中的可并行部分采用 FPGA 进行加速一般能够实现高性能、低功耗、低成本，但是直接将 TFG 中的所有可并行任务节点移植到可重构硬件平台上运行则不一定能够获得比 CPU 更好的执行性能。因此需要对影响 FPGA 性能的各种关键因素进行分析，通过预评估 FPGA 执行性能，综合衡量所能够获得的性能收益来判定一个任务是否适合划分至 FPGA 上执行。对任务在 FPGA 上执行性能的评估不仅考虑 FPGA 计算执行时间，还需要考虑诸如硬件配置、数据通信等开销。本文采用了前期研究成果[9]中的 FPGA 并行化开销测试算法对任务在 FPGA 上的执行性能进行评估。首先依次对 TFG 中并行任务集合的任务节点进行并行化开销的评估（参见文献[9]），如果加速效果明显，则将该任务标记为硬件任务，若不具有加速效果或者加速效果不明显则将该任务标记为软件任务。相比 CPU 而言，FPGA 主频较低，并且在执行时还有硬件初始化配置等额外开销，所以将串行任务集合中

的任务节点均标记为软件任务，划分至 CPU 上执行。

通过采用 FPGA 并行化开销测试算法对 TFG 中并行任务集合中的节点进行评估的软硬任务初筛之后，TFG 中拟划分至硬件执行的任务节点数目会进一步减少，这样有助于提高在任务划分过程中寻找合适软硬任务划分方案的算法的收敛速度。

3.4 基于改进模拟退火算法的软硬任务划分

软硬任务划分方案的确定过程是组合优化问题，属于 NP 困难问题。本文采用了改进的模拟退火算法来求解全局近似最优的软硬任务划分。模拟退火算法是一种基于迭代求解策略的随机搜索算法，具有与算法选取的初始值无关、渐进收敛性、能够以一定概率接受恶化解、防止过早收敛等优点[10,11]。

本文在任务划分过程中对模拟退火算法进行了改进，通过在模拟退火算法中添加两个记忆器 G_{best} 和 T_{best} 来分别记录当前降温过程中的最优划分结果的任务流图和任务最短执行时间。在算法初始阶段，令 G_{best} 等同于初始化任务流图，T_{best} 等于初始化任务流图的执行时间，在迭代过程中每接受一个新划分方案的任务流图，将当前任务流图的执行时间 T_{new} 与 T_{best} 进行比较，若 $T_{new}<T_{best}$，则用当前任务划分方案的任务流图 G_{new} 和当前任务流图执行时间 T_{new} 来替换 G_{best} 和 T_{best}。当算法结束时，最终的 T_{end} 与记忆器中的 T_{best} 进行比较，从中选取 T 值较小的任务划分方案作为最终的任务流图 G_{end}。具体步骤如下：

(1)设定初始化温度 C_0，每个温度下的退火次数 n，降温系数 k，将任务集合中的所有串行任务固定采用 CPU 进行执行，任务集合中的可并行任务可以随机指定执行器件(CPU 或 FPGA)，生成初始化任务划分方案的任务流图 G_0；

(2)随机从可并行任务集合中选取一个任务，改变其执行器件，生成新的任务划分方案 G_{new}，并重新计算任务划分方案的任务流图执行时间 T_{new}，$\Delta T=T_{new}-T_0$；

(3)若 $\Delta T<0$，则接受 G_{new}，并用 T_{new} 和 G_{new} 替换 T_{best} 和 G_{best}；若 $\Delta T>0$，则 G_{new} 按照概率 $P=\exp\frac{\Delta T}{C}<\text{rand}(0,1)$ 被接受，C 为当前温度，当新任务划分方案被接受时，$G_0=G_{new}$，$T_0=T_{new}$；

(4)判断迭代次数是否达到设定的退火次数 n，如果未达到则转步骤(2)；

(5)依据降温方法计算温度 $C_{k+1}=PC_k$，若温度 C_{k+1} 低于设定的最低温度则转到步骤(6)，否则重置迭代变量 n，并转到步骤(2)；

(6)比较 T_{end} 和 T_{best}，若 $T_{end}<T_{best}$，则 $G_{best}=G_{end}$，否则 T_{best} 和 G_{best} 保持不变。G_{best} 即为最终任务划分方案的任务流图。

4 实验与分析

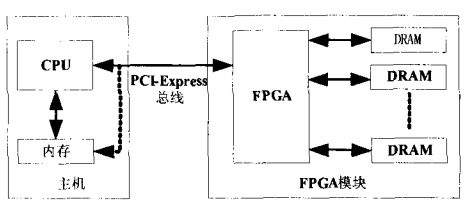

图 2 实验环境

本文采用的可重构硬件实验平台如图 2 所示，FPGA 作为协处理器，和主机通过 PCI-Express 总线进行联接，构成一个主从模式的可重构系统。其中，CPU 负责完成控制任务和部分代码的执行，并且负责对硬件任务的配置和数据传输；可重构硬件主要完成一部分可并行任务的加速，从而减少应用程序的整体执行时间。

为验证任务划分方法的有效性，本文选取了 Livermorec[12] 中 7 个具有代表性的程序片段和 Linpack 程序中的高斯消元部分作为测试用例，具体测试程序功能和介绍如表 3 所列。实验环境包括主机和 FPGA 开发板两部分，具体参数如表 1 和表 2 所列。

表 1 主机端设备参数

设备	型号
处理器	Pentium(R) 4 2.8GHz
内存	512MB
硬盘	80G

表 2 FPGA 设备参数

类型	器件	描述
FPGA	Stratix II GX(EP2SGX90F)	90960LE、4520448bit RAM、192 个 18*18 乘法器、8 个 PLL
存储器	QDRII SRAM	容量：18Mybtes；带宽：36bit
	DDR2	容量：256Mybtes；带宽：72bit
时钟	振荡器	100MHz、155.52MHz、56.2MHz
	晶振	25MHz
接口	PCI-Express	X8
	Ethernet RJ-45	10/100/1000 base

表 3 测试用例简介

测试程序	简介
1	Hydro fragment
2	Inner product
3	Banded linear equations
4	Tri-diagonal elimination, below diagonal
5	Difference predictors
6	Equation of state fragment
7	ADI integration
8	Gaussian elimination

表 4 给出了分别采用可重构硬件和软件执行方式的估算时间与实际执行时间的实验结果。从表 4 的结果中可以看出，估算的硬件执行时间比实际执行时间整体偏高，误差率不超过 10%。软件执行时间与估算结果相比整体偏低，误差率不超过 9.4%。从划分的结果上看，只有第 4 个测试程序中的测试结果过于相近(结果只相差 0.01 毫秒)，此时虽然划分结果为硬件任务，但是考虑到硬件任务和软件任务之间的开发难度和开发周期，最终结果还是将其划分为软件任务。

表 4 软硬件任务划分结果

测试程序	硬件执行时间		软件执行时间		划分结果	
	估算	实际	估算	实际	估算	实际
1	2.15	2.39	2.01	2.13	S	S
2	2.46	2.69	2.26	2.36	S	S
3	1.47	1.68	1.36	1.46	S	S
4	2.29	2.45	2.30	2.39	H	S
5	1.65	1.76	1.53	1.69	A	S
6	3.80	3.97	4.76	4.92	H	H
7	7.10	7.36	10.09	10.34	H	H
8	1.21	1.29	1.09	1.18	S	S

S 表示在 CPU 上执行，H 表示在可重构器件上执行

表 5 是 Gaussian Elimination 测试程序在不同的数据规模下的评估结果和实际测试结果。从表 5 中可以看出，划分结果与实际任务划分结果相同，能够有效提升应用程序在可

(下转第 289 页)

簇进行矩阵转换及求解本征向量的过程中都要用到对应的 D_i, L_i，对于聚集度不大的矩阵，拆分过程很快并且很多节点对应一个 D_i, L_i；相反，聚集度大的矩阵，拆分成独立节点的时间较长，分成的簇很多，对应同一个 D_i, L_i 的节点也较少，各个线程进行到同一步时，读取相同的 D_i, L_i 数据会比读取不同的节省很多数据读取时间。在计算精度方面，计算出来的结果与串行 sstemr 的结果基本相同，误差不超过万分之一。

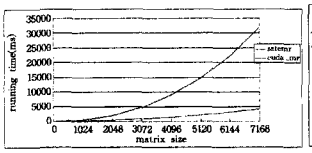

图 4 Wilkinson 矩阵运行时间对比

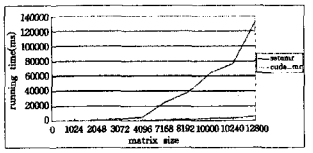

图 5 对角线元素是 4 的矩阵运行时间对比

表 2 对角线元素为 4 的矩阵串行和并行运行时间(ms)

比较值	对应不同规模的对角元素是 4 的矩阵运行时间对比					
	1024	2048	4096	8192	10240	12800
sstemr	1.40e+2	7.63e+2	5.01e+3	3.78e+4	7.69e+4	1.33e+5
cuda_mr	4.06e+2	1.39e+2	3.21e+2	2.13e+3	3.80e+3	6.38e+3
speedup	3.44	5.5	9.6	17.7	20.23	20.76

结束语 本文给出了一种基于 CUDA 的并行 MRRR 算法，用来求解对称三对角矩阵本征值问题，取得了较好的加速效果。实验证明，MRRR 算法适合在 GPU 上并行。但此方法在数据存储及并行策略上还有待改进，希望日后能够在保证精度的基础上改进方法以取得更大加速。

参 考 文 献

[1] Cuppen J J M. A Divide and Conquer Method for the Symmetric Eigenproblem[J]. Number, 1981, 36: 177-195

[2] Dhillion I S. A new $O(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem[D]. EECS Department University of California, Berkeley, 1997

[3] Dhillon I S. The Design and Implementation of the MRRR Algorithm[J]. ACM Transactions on Mathematical Software, 2006, 32(4)

[4] Dhillon I S, Parlett B N. Multiple Representations to Compute Orthogonal Eigenvectors of Symmetric Tridiagonal Matrices [J]. Linear Algebra and its Applications, 2006, 378(1): 1-28

[5] NVIDIA Corporation. NVIDIA CUDA Programming Guide 2.0 Final[M]. NVIDIA Corporation, 2008

[6] Gu M, Eisentat S C. A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem[J]. SIAM J. Mat. Anal. Appl, 1995, 16(1): 172-191

[7] Dhillon I S, Parlett B N. Orthogonal Eigenvectors and Relative Gaps[J]. SIAM J. Matrix Anal. Appl, 2003: 858-899

(上接第 278 页)

重构平台上的执行性能。Gaussian Elimination 测试程序中具有 3 层循环迭代，计算量相对较大，当数据规模增大时，硬件执行性能优势逐渐明显，当数据规模达到一定的量时，可并行任务划分在可重构器件上执行比在 CPU 上执行能够获得更高的性能。

表 5 Gaussian Elimination 测试结果

测试程序	硬件执行时间		软件执行时间		划分结果	
	估算	实际	估算	实际	估算	实际
256 * 256	0.153	0.169	0.131	0.145	S	S
512 * 512	0.458	0.499	0.433	0.472	S	S
1024 * 1024	1.680	1.815	1.714	1.849	H	H
2048 * 2048	6.622	7.091	7.048	7.498	H	H

S 表示在 CPU 上执行，H 表示在可重构器件上执行

从上述 8 种测试程序的实验结果中可以看出，本文提出的基于程序性能评估的软硬件任务划分方法能够有效指导应用程序的任务划分，实现提升应用程序执行性能的目标。

结束语 可重构系统的高性能和灵活性为应用程序提供了良好的计算平台，软硬件任务划分是有效利用可重构资源的关键技术之一。本文提出的采用任务流图的方式对应用程序结构进行描述，能够屏蔽可重构计算系统中不同计算部件的不同设计语言和设计方法给任务划分带来的困难。通过对 FPGA 计算开销、配置开销、通信开销的预评估测试，能够减少划分方案的搜索范围，提高任务划分方案的准确性。最后通过改进的模拟退火算法能够形成任务与执行期间之间的有效映射。实验结果表明，该划分方法具有较好的划分效果，以及较好的算法收敛速度。

参 考 文 献

[1] Hartenstein R. A decade of reconfigurable computing: a visionary retrospective[A]// Proceedings of the conference on Design Automation & Test in Europe, 2001[C]. New York, USA: ACM Press, 2001: 642-649

[2] Compton K, Hauck S. Reconfigurable computing: a survey of systems and software[J]. ACM Computing Surveys, 2002, 34 (2): 171-210

[3] 熊志辉，李思昆，陈吉华. 遗传算法与蚂蚁算法动态融合的软硬件划分[J]. 软件学报，2005，16(4)：503-512

[4] 马平. 可重构系统中的任务划分和任务调度的研究[D]. 天津：河北工业大学，2006

[5] Li Y, Callahan T, Darnell E, et al. Hardware-Software co-design of embedded reconfigurable architecture[A]// Proceedings of Design Automation Conference, 2000[C]. Los Angeles, California: ACM, 2000: 507-512

[6] 沈英哲. 可重构计算系统中软硬件代码划分技术研究[D]. 合肥：中国科技大学，2007

[7] 袁爱平，傅明. 嵌入式系统软硬件划分方法探索[J]. 计算机应用，2008，28(9)：2427-2429

[8] Girkar M, Polychronopoulos C D. The hierarchical task graph as a universal intermediate representation[J]. International Journal of Parallel Programming, 1994, 22(5): 519-551

[9] Zhang Dan, Zhao Rong-cai, Han Lin, et al. A Parallelization Cost Model for FPGA[J]. Advanced Materials Research, 2011, 181-182, part 2: 623-628

[10] Ingber L. Very fast simulated annealing [J]. Math Compute Modeling, 1989, 12(8): 967-973

[11] Noori H, Mehdipou F, Murakam K. A Reconfigurable Functional Unit for an Adaptive Dynamic Extensible Processor[A]// Proceedings of 16th IEEE International Conference on Field Programmable Logic and Applications 2006[C]. Madrid, SPAIN: IEEE Press, 2006: 781-784

[12] Livermore Benchmarks[EB/OL]. http://www.netlib.org/benchmark/livermorec, 1992-10-20