**什么是多处理器片上系统（MPSOC）**

随着商业和大规模计算密集型科研应用对计算平台性能上越来越严格的要求，使得多处理器片上系统(MPSOC)的研究受到越来越广泛的关注。由于多处理器片上系统在单芯上集成了多个异构处理单元，使其可以完成更加复杂的功能，在图像处理、多媒体和嵌入式系统等应用领域前景广阔。多处理器片上系统通过对不同应用场景的定制化设计来满足市场对计算平台性能的苛刻要求，这种灵活的，定制化的设计促使多处理器片上系统成为新一代SOC的主流设计趋势。[1-2]Tilak Agerwala, Siddhartha Chatterijee. Computer architecture: Challenges and opportunities for the next decade[J]. IEEE Micro. 2005. 25(3): 58-69.

多处理器片上系统(MPSoC)是一种由多个处理器单元和专用功能模块构成的一个复杂的单芯片集成计算系统，通常是为嵌入式应用程序而设计，用于满足应用程序对于系统在计算性能，实时性，功耗与成本等方面的需求[1] W. Wolf, “The future of multiprocessor systems-on-chips,” inProc. 41st Assoc. Comput. Machinery/IEEE Design Automat. Conf. (DAC), 2004, pp. 681–685.

MPSOC不是仅仅是简单的实现片上多处理器(chip multiprocessor)的片上集成，片上多处理器强调将更多的处理单元放在单片上用于提高单位面积晶体管密度，在设计时并不考虑应用的需求，而MPSOC实现的是定制化的集成芯片，通过定制体系结构来满足不同应用在成本和功耗等方面的需求。

MPSOC体系结构常常用于多任务应用程序的计算，有效的利用MPSOC体系结构的异构性可以有效的满足多任务应用程序在性能，功耗和成本等方面的要求，典型的MPSOC体系结构如下图所示，由3部分组成：

（1）处理器单元，包括处理器（Processor）、数字信号处理芯片（DSP）、专用集成电路（ASIC）用于执行计算任务；

（2）存储（Memory）模块用于保存数据；

（3）通信网络（Network）用于组件之间的通信。



图1 MPSOC体系结构

**MPSOC上的映射与调度问题**

通信、多媒体和计算机网络等领域，将对计算平台固有的成本压力转换成对多处理器片上系统的性能要求，如何高效的发挥多处理器片上系统的计算性能成为目前研究的热点。

MPSOC上的组件通过片上互连总线连接成为一个整体，为适应不同种类应用程序对不同组件的应用需求MPSoC通常被设计成异构的，AMD公司在2006年提出基于“融合”技术的多核处理器产品路线图，实现了同构片上多处理器系统向异构片上多处理器系统的转化。在MPSOC上我们使用不同种类的处理器来加速应用程序的不同部分，从而优化多任务应用程序的性能。MPSOC结构的复杂性使得对应用性能的优化提出了更高的要求。

近年来，对于MPSoC的研究上围绕系统开发，系统安全，编程模型与任务调度等方面的研究已经拥有了一些重要成果。为了优化应用程序的性能，例如，程序执行时间，需要把一个复杂的任务分解，将分解后的子任务映射到异构处理器上，寻找一组合适的映射关系用于满足实时性，成本与功耗等约束条件成为MPSoC系统亟需解决的关键问题之一。然而多处理器上任务的调度与映射问题属于NP难问题[2]。J. Kim, S. Lee, H.l Shin, Y. Lee and H. Bae, “Effective task mapping and scheduling techniques for heterogeneous multi-core systems based on zone refinement,” Computer Sciences and Convergence Information Technology (ICCIT), 6th International Conference. 2011, pp. 363 – 366.

由于异构嵌入式系统自身结构的复杂性与多任务应用程序规模的不确定性，使得传统方法在处理多处理器上任务的调度与映射问题时失去了有效性，特别是针对大规模的多任务应用程序的调度与映射问题。改进算法的效率并构建任务映射与调度机制成为一个亟待解决的问题。

当前研究提出许多基于启发式的方法来优化任务的映射与调度问题，例如禁忌搜索算法（Tabu Search）[3-4]

T. Wiangtong, P.Y.K. Cheung and W.Luk, “Comparing three heuristic searchmethods for functional partitioning in hardware–software code-sign.” Design Automation for Embedded Systems, 6(4):425–449, 2002.

P. Eles, Z. Peng, K. Kuchcinski and A. Doboli, “System level hardware/software partitioning based on simulated annealing and tabu search.” Design Automation for Embedded Systems, 2:5–32, 1997

遗传算法（Genetic Algorithms）[5]

M. Grajcar, “Genetic list scheduling algorithm for scheduling and allocation on a loosely coupled heterogeneous multiprocessor system." In DAC’99: the 36th ACM/IEEE conference on Design Automation, pages 280–285, 1999

以及Kernighan-Lin [6]方法

F. Vahid and T.D. Le, “Extending the Kernighan/Lin Heuristic for Hardware and Software Functional Partitioning.” Design Automation for Embedded Systems,2(2):237–261, March 1997.

以上方法被证明能够在合理的时间内获得对优化问题的一个局部最优的结果。但是这些方法在处理问题时只关注了映射与调度问题的一个方面，即单一的处理调度或映射问题，并且当求解空间的约束条件比较苛刻时，算法在问题求解上的性能就会严重下降，优化的结果往往不尽人意。蚁群算法[7][8][9]

G. Wang, W. Gong, B. DeRenzi and R. Kastner, “Ant colony optimizations for resource and timing constrained operation scheduling.” IEEE Transactions on Computer-Aided Design of Integrated Circuitsand Systems, 26(6):1010–1029, June 2007.

P.C. Chang, I.W. Wu, J.J. Shann and C.P. Chung, “ETAHM: An energy aware task allocation algorithm for heterogeneous multiprocessor.” In Proceedings of DAC ’08, pages 776 –779, june 2008.

F. Ferrandi, P.L. Lanzi, C. Pilato, D. Sciuto and A. Tumeo, “Ant colony heuristic for mapping and scheduling tasks and communications on heterogeneous embedded systems.” IEEE Trans. on Computer-AidedDesign of Integrated Circuits and Systems, 29(6):911–924, June 2010.

是另外一个在处理映射与调度问题中比较常用的优化方法，使用蚁群算法优化问题时对问题的调度与映射的优化是同时进行的，当优化问题的规模比较小时，蚁群算法显示了其有效性和可行性。但是当面对大规模的优化求解问题时，由于优化空间的增加以及计算复杂度的上升，蚁群算法对与该问题的求解上失去了有效性。因此，目前还没有一个通用的，能够高效的处理复杂应用程序在多处理片上系统的调度与映射问题的优化算法。

本文基于抽象模型给出任务的形式化定义，然后进行调度策略与算法的设计

目标体系结构的抽象

本文我们抽象出一个通用的异构多处理器片上系统（MPSOC）的体系结构用于进行一般化说明，其形式化定义如下：

A=PUC(1)

其中，P表示一组异构的处理单元，用于处理应用程序的不同部分，C表示一组通信组件，用于进行处理单元之间进行数据的传输。一个抽象异构多处理器片上系统的一个简单示例，如下图所示，它由四个异构的处理单元，分布式内存和单一的系统总线作为通信组件组成的。



关于这个抽象的目标体系结构，我们还应注意一下事项：

1. 对于目标平台上的相关资源（例如，本地内存），这些资源可以分为两种类型：1)可再生资源RR，即任务运行所分配的资源在任务完成时资源得到释放，释放后的资源可以继续分配使用；2)不可再生资源的NR，即任务运行所分配的资源在任务完成时得不到释放，已经分配的资源在任务完成后不能继续分配使用。
2. 为了简化任务的优化调度，我们假设每个硬件在同一时间最多只能运行一个任务。
3. 当任务A和任务B之间产生数据传输时，如果任务A和任务B在同一个执行硬件上运行，那么数据传输的耗时为0。如果任务A和任务B不在同一个执行硬件上运行，那么数据传输的耗时将通过传输数据量的大小与系统总线的带宽进行估算。

多任务应用程序

什么是多任务应用程序

一个多任务应用程序由N个子任务组成，任务之间具有时序关系和数据的传输依赖，每个子任务在运行时需要占用单一的执行单元。

多任务应用程序的抽象

在处理异构嵌入式系统上问题的映射和调度时需要将多任务应用程序抽象成能够在目标平台上进行调度的模型。本文我们将多任务应用程序抽象成一个有向无环图（DAG），其形式化定义如下：有向无环图G，*G* = (*V, T)，*其中， V代表一组计算操作(计算任务)，T表示任务之间的数据传输操作(数据通信)，即任务之间的依赖关系。

在多任务应用程序的执行顺序上存在以下约束：

如果有向无环图G存在一条边*tt*(*v’, v*)∈*T*，这表明任务v和任务v’之间有时序上的依赖关系，任务v只有在任务v’执行完并且所有指向v的数据传输完毕后才能执行。在有向无环图G中的每一条边*tt*(*v’, v*)∈*T*都标有一个权重，这表示由父任务v’到子任务v之间的数据传输的大小。

一个简单的多任务应用程序的有向无环图如下所示：



带有这种数据流的应用程序的主要代表有，科学计算和多媒体应用程序需要高并行计算的领域，在上面的例子中，这些应用程序的行为是可以静态预测的并且用于减少程序执行时间的优化可以是离线执行的，由于在多任务应用程序中，具有数据以来关系的子任务之间进行数据传输的十分频繁，而且根据不同的任务类型，数据传输的数量往往差别很大，因此，在目标平台上对数据传输任务的映射与调度也成为必须考虑的问题。

我们用G=（T, E）来表示在嵌入式系统上进行多任务应用程序的调度，在目标平台上，资源Q被分为可更新R和不可更新N两部分。

任务j：被定义为在目标平台上的某一主键上执行的活动。因此，多任务应用程序的任务可以被抽象成集合J来表示。

执行点i：当任务j在目标平台的一个组件ak上执行时，对资源数量和执行时间的要求。事实上，每一个任务j根据可以有不同的执行点，这是根据任务的类型和平台的组件类型决定的，一个任务j可以在多个执行组件上执行，但并不一定能在所有的组件上。例如: 程序中的执行任务不能分配到通讯组件上执行，此外，一些特殊计算组件只能执行某些特定的计算任务。集合I包含了任务集合J中所有可用的执行点。表格I展示了一个执行点的示例，它显示了任务在每一个执行点上所需要的时间和资源数量，同样，对于每一个任务，在组件P3上我们产生了两个执行点.这是由于对组件P3上资源的消耗不同而决定的。