様　式　Ｃ－１９、Ｆ－１９、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成 ２７ 年　 ５ 月 １７ 日現在

機関番号：１３９０１
研究種目：基盤研究(C)
研究期間：2012～2014
課題番号：２４５０００５８

研究課題名（和文）階層型組み込みメニーコア向けタスク配置手法の研究

研究課題名（英文）Task Mapping for Hierarchical Many-core Processors in Embedded Systems

研究代表者
枝廣　正人（EDAHIRO, MASATO）

名古屋大学・情報科学研究科・教授

研究者番号：５０５７８８５４

交付決定額（研究期間全体）：（直接経費）　　4,000,000円

研究成果の概要（和文）：階層型メニーコアアーキテクチャに対し、階層構造を考慮したタスク配置手法を提案した。提案手法はLSI配置向けに提案された階層クラスタリング法をもとにしているが、タスクマッピング問題に適用するため、タスク間通信量や、アーキテクチャ内の配線の使用頻度の偏り等を考慮している。従来手法と比較した結果、通信コスト、アプリケーション完了時間を削減し、通信コストの最小化によってアプリケーション自体の実行時間が削減できるということを示した。

研究成果の概要（英文）：In this research, we proposed a task mapping method that considers features of task graphs and performance characteristics of hierarchical many-core architectures.
We proposed the "HCME" toward task mapping method for "CMesh" which is a type of hierarchical many-core architectures. We also proposed a merge technique required when there are more number of tasks than the number of cores.
We compared our proposed mapping method with existing algorithms. As a result of evaluation experiments, our proposed method reduced communication cost and application completion time.

研究分野：組込みシステム

キーワード：組込みシステム　マルチコア　メニーコア　タスク配置

１．研究開始当初の背景

半導体微細化の進展により、一つのLSI上に複数のプロセッサ・コアが搭載されたマルチコアや、グラフィックスのような特定用途向けには数十、数百のプロセッサ・コアが搭載されたメニーコアが広く使われている。メニーコアは、単体プロセッサの動作周波数向上による方法と比較して低消費電力で性能向上できるため、将来の主流になると言われている。しかしながら、実際にその性能を享受するためには、ソフトウェアに含まれる複数のタスク（スレッド、プロセスなどともよばれる）をプロセッサ・コアに割り付けていく必要がある。

現在発表されている汎用用途向けメニーコアはタイル型とよばれるアーキテクチャであるが、さらにプロセッサ・コアが増えた時には階層化し、階層型メニーコアになると考えられている。そのため、タスク配置は階層構造を考慮し、プロセッサ・コアに割り付けていく必要がある。

２．研究の目的

我々はLSI回路設計において、回路をLSI上に小面積、高性能に配置するため、回路のグラフ構造をLSI上に配置するためのアルゴリズムについて研究し、回路を階層化することにより効率の良い配置を行う「階層クラスタリング法」を提案した[1]。

タスクは、タスク間に依存関係を持つ場合が多く、タスクグラフとよばれるグラフ構造を持つ。本研究では、階層型メニーコアアーキテクチャの性能特性も考慮しつつ、階層クラスタリング法を階層型メニーコアおよびタスクグラフ向けに発展させ、階層型メニーコア上で効率よく並列実行できるようなタスク配置の実現を目的とした。

LSI配置問題とタスク配置問題は、接続する要素を近くに配置するという意味で基本は同じである。しかしながらタスクには実行順序があり、大きく異なっている。

これらをふまえ、現状のアルゴリズムを発展させ、階層型組み込みメニーコア上のタスク配置向けの新しいアルゴリズムを考案することが本研究の目的である。

３．研究の方法

階層クラスタリング手法は、枝集合と節点集合を持つグラフ構造を、２つの節点集合に分割し、集合間の枝数を最小化する手法の一つである。節点集合を何らかの方法で２分割し、枝数が減少するように節点を交換する方法が知られているが、節点が多数ある場合には局所最適解に陥りやすく、接続の強い節点をグループ化（クラスタ化）する方法の場合には微調整が難しいという難点がある。それを改良するために階層的にクラスタ化していく方法が階層クラスタリング法である。この方法を用いて階層型メニーコア向けタスク配置を行うため、以下の研究を進めた。

(1) アーキテクチャの持つ階層に合わせたタスククラスタリング

最終的にタスクを階層型メニーコアに配置していくため、タスクのクラスタ化はメニーコアアーキテクチャの持つ階層と整合していることが望まれる。しかしながら、一般に並列モデルから生成したタスクグラフ構造はアーキテクチャの持つ階層構造とは整合しない。そのため、ターゲットとなる階層化メニーコアで効率よい実行ができるようなクラスタ化手法について研究を進めた。クラスタは階層的に行い、はじめはプロセッサ数よりも多く、最終的には２クラスタにする。そのことにより、アーキテクチャ上で大局的な視点で配置することも、局所的に微調整することも可能とする。

(2) アーキテクチャ性能特性に則したタスク配置

タスク間（クラスタ間）通信は、実際のアーキテクチャ上での通信となるため、通信性能特性を考慮しながらタスク（クラスタ）を配置していく必要がある。具体的に言えば、通信が遅い場合にはタスク（クラスタ）粒度を大きくし、通信オーバーヘッドの割合を小さくする必要があり、通信が速い場合には粒度は小さくてもよい。また、アーキテクチャ階層に応じて通信性能は異なるため、それらの性能特性を考慮し、階層的なクラスタを階層型メニーコアアーキテクチャに配置し、効率よい実行を達成できるようなアルゴリズムについて研究を進めた。

４．研究成果

(1) 提案アルゴリズム

提案手法は三つの部分からなる。

a．クラスタリング :タスクグラフをクラスタ化し、階層構造を構成
b．分割 :クラスタを均一に分割
c．配置 :分割結果の位置関係を決定

既存手法では「分割」と「配置」によって配置問題を解いているが、提案手法では前述の階層クラスタリング法の特徴を反映し、クラスタリングステージを追加した。

a．クラスタリング

階層クラスタリング法では、構成要素を接続するエッジ数に関してクラスタリングが行われていた。階層クラスタリング法をタスク配置手法に拡張するため、エッジ数ではなく、タスク間の通信量に関してクラスタリングを行うものとした。通信量の多いエッジを持つ２つのタスクを次々に結合していく。結合が進むたびにタスクの数は１つずつ減り、最終的には２つのクラスタとなる。クラスタリングを行う目的としては、分割において局所最適解に陥りにくい。通信量の大きいエッジは早期にクラスタリングされるため、大きな通信が同一コアクラスタ内にマッピングされやすい等の利点がある。

b．分割
分割のステージではタスククラスタ内に分割線を挿入し、分割線を横切る通信を考慮してタスククラスタを分割する。この分割では、通信コストの削減を狙う他、プロセッサアーキテクチャを考慮し、全体的なコアクラスタ間の通信を減らし、通信の分散を狙う目的もある。

c．配置
配置のステージでは分割されたクラスタの位置関係を決定する。配置のステージで実現するのは通信コストの最小化と通信の分散の 2 点である。配置を決定する際にも分割時と同様のモデルを導入する。

d．全体のアルゴリズム
クラスタリング、分割、配置を繰り返し、最終的なマッピングを決定する。アーキテクチャの形に応じて分割、配置のパラメタを調整する事によって、様々なアーキテクチャに対応できる。以下が提案手法のアルゴリズムである。

① （クラスタリング） 1 つのタスクにつき一回結合できるものとして、通信の重みが大きいタスクを、結合できる箇所が存在する限り、順に結合。
② （クラスタリング） 1 つのクラスタを 1 つのタスクとみなす。①、②を繰り返し、最終的に 2 つのクラスタにする。
③ （分割）内包タスク数が同数になるようクラスタを分割。
④ （配置）分割後のクラスタを配置。
⑤ 配置されたクラスタをデクラスタリングし、タスクグラフとする。
⑥ 分割されたそれぞれのクラスタにおいて 1 クラスタずつ①～⑤を繰り返す。十分に分割され、配置が完了したら終了。

アルゴリズムは図１のように進行する。図の簡単化のため、タスクの数を６としている。

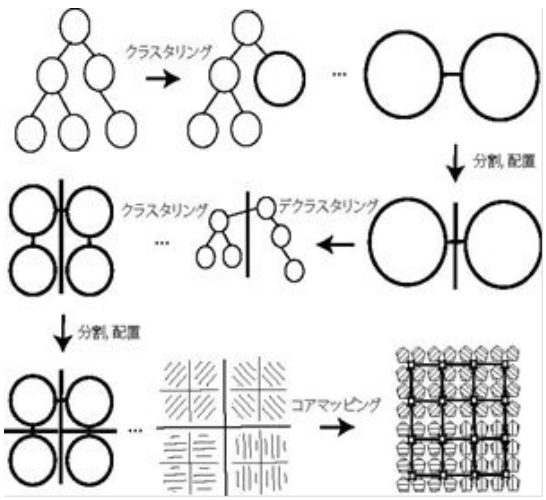

図１．アルゴリズム全体のイメージ

(2) 実験
①実験手法
既存手法としては、貪欲法、グループ化法の大きく二つに分けられるが、これらの方法および元の階層クラスタリング法[1]と提案手法の比較を行う。貪欲法からは NN Embed 法[2]、Topo-LB 法[3]を選択し、グループ化法からは Cluster-Based ILP 法[4]を選んだ。ここで、これら 3 つの既存手法は階層を持つアーキテクチャを想定していないため、アーキテクチャ情報においてクラスタ内のコア間距離を０と設定することにより、階層に対応したアルゴリズムに拡張した。
タスクグラフはグラフ生成ツールを用いたものとセンサレスモータ制御から生成されたタスクモデルを用いた。グラフ作成ツールは TGFF[5]を用い、16 タスクから 1024 タスクのタスクグラフをランダムに各 10 種類作成した。プロセッサアーキテクチャとしては集中メッシュトポロジを用いた。各マッピング手法を用いてマッピングを決定し、通信コスト、負荷の分散、実行時間、アプリケーション完了時間によって評価を行った。
モータ制御についてはセンサレスモータ制御アルゴリズムより生成したタスクグラフを用いた。実験には以下の PC を使用した。
OS: Windows7 Professional 64bit
CPU: Intel Xeon W5590 3.33GHz(2 ソケット)
メモリ: 24GB

②実験結果（通信コスト）
通信コストでの比較は図２のようになった。このグラフは提案手法の通信コストを 1 とした場合の各手法の通信コストのグラフである。グラフの x 軸の値は「コア数（マージ前のタスク数）-マージ法の種類」である。マージ法の種類は１が通信コストのみを考慮したもの、２が負荷分散のみを考慮したもの３が通信コストと負荷分散の両方を考慮したものである。なお、今回は実行時間に関して 8 時間の制限時間を課しており、実行時間が 8 時間以上となったものに関しては結果無しとしている。グラフでは C-Based ILP 法の 256 コアへのマッピングが結果無しとなっており、見やすさのためにグラフの値を最大値としている。
提案手法と NN Embed 法を比較した場合、NN Embed 法では、マッピング決定時に多くのランダム要素が入り、アーキテクチャ全体の情報を考慮していないのに対し、提案手法にはランダム性が無く、局所的なアーキテクチャ情報だけでなく、全体的なアーキテクチャの情報を考慮しているため、大きな差が生まれている。
Topo-LB 法は最初に配置されるコアの位置をもとに処理が進むが、その初期配置が Topo-LB 法の評価関数の仕様上、毎回のトポロジの中心付近になる。最適マッピング解を考える場合、最初に選ばれたタスクが配置されるべき場所が中心付近でなく、端に近い部

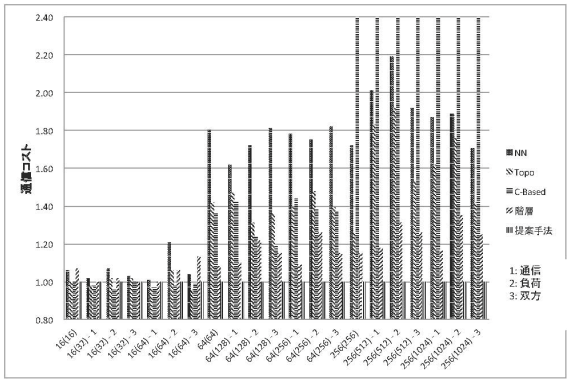

図２．通信コスト

分であった場合、その時点で最適解に到達する事が不可能となるという欠点がある。一方、提案手法では一つのタスクをあるコアにマッピングし、そのコアを中心として処理が進むという性質は無く、全体のマッピング結果が決定されて初めてコアへのマッピングが行われる。つまり、各タスクはそれぞれふさわしいコアにマッピングされるのである。しかし、16 コアへのマッピングでは、提案手法を Topo-LB 法が上回っている点が見られる。これは、提案手法内で使用している KL 法[6]では、分割決定の評価関数が通信コストよりも同数での分割ということに重きを置いており、このことによるコストの悪化が前述した Topo-LB 法の問題点によるコストの悪化を上回ってしまったためであると考えられる。

Cluster-Based ILP 法に関しては、線形計画法の特性上、一度に広い範囲のマッピング問題を解く事が不可能である。つまりマッピング問題の対象が 12 タスクを超えると実用的な時間でマッピング問題を解けなくなる。そのため、今回の実験では線形計画法で解く範囲を 8 タスク→ 8 コアのマッピングとした。そのため、マッピング解の大部分がグラフカット手法である KL 法によって決定されており、その点がコストの悪化に繋がったと考えられる。また Cluster-Based ILP 法内で用いられている KL 法には、グループを崩しながらタスクの交換を行うという機能は無いため、局所最適解に陥りやすいこともコストの悪化の一因になっていると考えられる。KL 法の介入が少ない 16 コアへのマッピングでは優れた通信コストを見せてはいるものの、線形計画法を用いてマッピング問題を解く手法は大規模なアーキテクチャへのマッピングには不向きであると考えられる。

元の階層クラスタリングを改良したものでは、内包タスク数に差を許して分割を行い、分割後にクラスタ間でタスクを移動するというクラスタリング部の差、そして配置法として NN Embed 法を用いているという配置部の差がこの通信コストの差に繋がっている。

③実験結果（実行時間）

各マッピング手法とマージ法の組み合わせによる実行時間のグラフを図３に示す。C-Based ILP による 256 コアへのマッピング部分は、実行に 8 時間以上かかったため値無しとしてグラフの値を最大値としている。NN Embed 法と Topo-LB 法のオーダはタスクグラフのエッジの数を$|E|$とした場合、$O(|E|)$であり、対象アーキテクチャのコア数を$|v|$とした場合、KL 法($O(|v|^3)$)がコア数オーダで呼び出される提案手法と階層クラスタリング法は $O(|v|^4)$となっている。この計算量については、アルゴリズム実装上の工夫で下げられると考えており、それは今後の課題である。

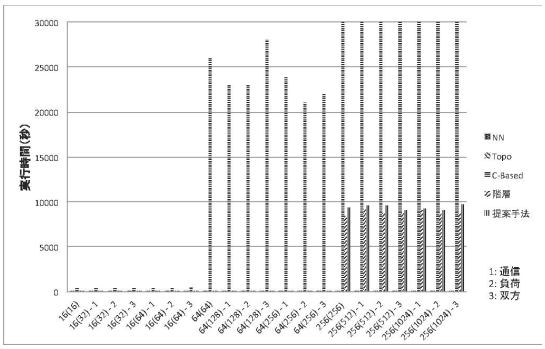

図３．実行時間

ただし、組込みアプリケーションにおいては、組込みプロセッサ上で同じアプリケーションが数年から数十年も作動し続けるという事が起こる。そういった場合、優れたマッピング結果によって実行時間が低減できるのであれば、数時間程度の計算時間は許容範囲であると考えられる。元の階層クラスタリングを改良したものでは、タスク数の差を許してクラスタを分割するという性質上、通信コストは悪くなるが、実行時間が小さいという性質を示した。

④実験結果（アプリケーション完了時間）

アプリケーション完了時間での比較は図４のようになった。なお、クラスタ間通信に掛ける定数は 10 とした。このグラフは提案手法の完了時間を 1 とした場合の各手法の完了時間のグラフである。グラフでは C-Based ILP 法の 256 コアへのマッピングが結果無しとなっており、見やすさのためにグラフの値を最大値としている。

通信コストの比較と同様の傾向が見られる事が分かり、通信コストの最小化によってアプリケーション自体の実行時間が削減できるということを示している。ただし、タスクグラフのクリティカルパスが通信量の小さいエッジを多く含む場合、アプリケーション完了時間が削減しにくいという欠点があり、クリティカルパスとなりうるタスク群を事前に調査し、補正を掛けてマッピングする等の解決策が必要であると考える。

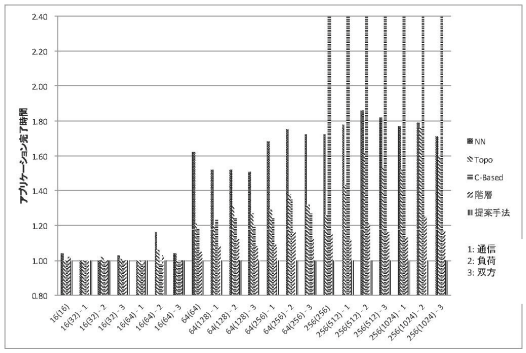

図４．アプリケーション完了時間

⑤ 実験結果（モータ制御）

センサレスモータ制御[7]から生成した 110 タスクのタスクグラフを用いて、通信コスト、アプリケーション完了時間、実行時間を指標として評価を行った。110 タスクのタスクグラフを、通信コストのみを考慮したマージ方法によって 64 タスクにマージし、64 コア集中メッシュトポロジにマッピングした。図５にその結果を示す。

通信コスト（提案手法を１に正規化）				
NN	Topo	C-Based	階層	提案手法
1.31	1.26	1.10	1.05	1

アプリケーション完了時間（提案手法を１に正規化）				
NN	Topo	C-Based	階層	提案手法
1.21	1.16	1	1	1

実行時間（秒）				
NN	Topo	C-Based	階層	提案手法
0.16	0.31	260000	39.02	42.24

図５．モータ制御モデルにおける比較

通信コストとアプリケーション完了時間の比較では、提案手法の通信コストを 1とした場合の各手法の割合を示しており、実行時間は各手法の実行時間(秒)である。ランダム生成したタスクグラフにおける結果と同等の傾向を示しているが、提案手法と他手法との差が全体的に縮まっている。これは、センサレスモータ制御のタスクグラフの通信量の重みが小さく、差異が出にくい事が原因だと考えられる。およそ 40 秒の処理時間で、通信コストにおいて平均 15%、アプリケーション完了時間において平均 8%の削減値は、組込みタスク配置としては優れていると言える。

<引用文献>

[1] Edahiro, Masato and Yoshimura, Takeshi, New Placement and Global Routing Algorithms for Standard Cell Layouts, ACM/IEEE Design Automation Conf. (DAC), 642 - 645, 1990.

[2] Lo, Virginia M., et al., OREGAMI: Tools for mapping parallel computations to parallel architectures, International Journal of Parallel Programming, Vol. 20, No.3, 1991, pp.237-270.

[3] Agarwal, Tarun, et al., Topology-aware task mapping for reducing communication contention on large parallel machines, Parallel and Distributed Processing Symposium, 2006.

[4] Tosun, Suleyman, Cluster-based application mapping method for Network-on-Chip, Advances in Engineering Software, Vol.42, No.10, 2011, pp.868-874.

[5] http://ziyang.eecs.umich.edu/~dickrp/tgff/

[6] Kernighan, Brian W. and Shen Lin, An efficient heuristic procedure for partitioning graphs, Bell system technical journal, Vol.49, No.2, 1970, pp.291-307.

[7] Akimatsu, Ryunosuke and Doki, Shinji, Improvetorque response using the inverter overmodulation range in position sensorless control system of PMSM, Industrial Electronics Society, IECON 2013 -39th Annual Conference of the IEEE Digital Object Identifier, 2013.

５．主な発表論文等

（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計１件）

油谷 創，枝廣 正人，階層構造を持つメニーコアアーキテクチャへのタスクマッピング，情報処理学会論文誌，査読有，Vol.56，No.8，2015，掲載予定．

〔学会発表〕（計１件）

油谷 創，枝廣 正人，階層構造を持つメニーコアアーキテクチャへのタスクマッピング，情報処理学会第３４回組込みシステム研究会，2014 年 9 月 17 日，札幌.

６．研究組織

(1)研究代表者

枝廣　正人（EDAHIRO，Masato）

名古屋大学大学院情報科学研究科

研究者番号：50578854

(2)研究分担者　なし

(3)連携研究者　なし

(4)研究協力者

油谷　創（ABRADANI，Sou）