Vol.2010-ARC-187 No.24
Vol.2010-EMB-15 No.24
2010/1/29

マルチコア向けオンチップメモリ貸与法における実行コード生成法の改善

福本尚人[†1]　今里賢一[†1]
井上弘士[†1]　村上和彰[†1]

本稿では，マルチコア・プロセッサ向けのオンチップメモリ貸与法を改良し，評価を行った．オンチップメモリ貸与法では，プロセッサコアを「演算用」だけでなく「メモリ性能向上用」に活用することで，性能向上を目指す．メモリ性能向上用のコアは自身が持つオンチップメモリを演算用のコアへ貸与する．本方式では，適切なコア分配ならびに貸与メモリのデータ割当てが極めて重要である．本稿ではこれらの方式を改良する．具体的には，一回の事前実行で得た情報をもとに，主記憶アクセス回数最小となるメモリ性能向上用コアの割当てデータの決定，ならびに，性能モデリングに基づく適切なコア分配を行う．評価した結果，最大で 63% の性能向上を達成した．

Improving Execution Code Generation for On-chip Memory Lending on Multicores

Naoto Fukumoto,[†1] Kenichi Imazato,[†1] Koji Inoue[†1] and Kazuaki Murakami[†1]

This paper proposes the concept of performance balancing, and reports its performance impact on a multicore processor. Integrating multiple processor cores into a single chip, can achieve higher peak performance by means of exploiting thread level parallelism. However, the off-chip memory bandwidth which does not scale with the number of cores tends to limit the potential of multicore processors. To solve this issue, the technique proposed in this paper attempts to make a good balance between computation and memorization. Unlike conventional parallel executions, this approach exploits some cores to improve the memory performance. These cores devote the on-chip memory hardware resources to the remaining cores executing the parallelized threads. In our evaluation, it is observed that our approach can achieve up to 63% of performance improvement compared to a conventional parallel execution model in the specified program.

1. はじめに

複数のプロセッサコアを 1 チップに搭載したマルチコア・プロセッサが主流となっている．マルチコア・プロセッサでは複数コアで並列処理を行うことで，高性能化を達成できる．半導体微細化技術の進歩とともに搭載されるコアの数は増加する傾向にある．そのため，マルチコア・プロセッサにおいて並列処理性能を高めることは極めて重要である．

マルチコア・プロセッサでは，常に期待する性能が得られるわけではない．シングルコア・プロセッサと比較して，プロセッサ－メモリ間の性能差の拡大（いわゆる，メモリウォール問題）が深刻化するためである．低速なオフチップメモリへのアクセスは膨大な時間を要する．さらに，マルチコア・プロセッサではオフチップメモリバンド幅の不足により，アクセス時間が増加する可能性が高い．なぜならば，複数コア搭載によりメモリアクセス頻度が増加する一方で，オフチップメモリバンド幅はコア数に比例しないためである．つまり，コア数の増加により演算性能は向上するが，相対的なメモリ性能の低下により，プロセッサ全体の性能が抑制される．

この問題を解決するために，我々はマルチコア向け演算/メモリ性能バランシング技術を提案した[4]．従来のマルチコア実行では，スレッドレベル並列性を最大限活用するために，全てのプロセッサ・コアで並列化プログラムを実行する．これに対し，提案方式では，一部のコアをメモリ性能向上用に活用する．メモリ性能向上用のコアは，自身が持つオンチップメモリ資源をプログラム実行用のコアに貸与する．プログラムの特性に合わせてメモリ性能向上用のコア数を調節することで高性能化を達成する．本方式では， 貸与メモリに如何に有効なデータを割当てるか，ならびに，適切な実行コア配分を如何に求めるかが極めて重要である．文献 4) では，配列データに特化したデータ割当てを行った．また，最適なコア配分は数回の事前実行による性能予測により見積もった．これらの方式では，特定のプログラム以外ではデータ割当ての効果がなく，コア配分予測に必要な事前実行の回数が多いといった問題があった．

そこで，本稿では，一回の事前実行により得られた情報で，貸与メモリのデータ割当てならびに実行コア配分を決定する方式を提案する．また，提案方式を Cell BroadBand En-

†1 九州大学 大学院 システム情報科学府/研究院
Graduate school / Faculty of Information Science and Electrical Engineering, Kyushu University

Vol.2010-ARC-187 No.24
Vol.2010-EMB-15 No.24
2010/1/29

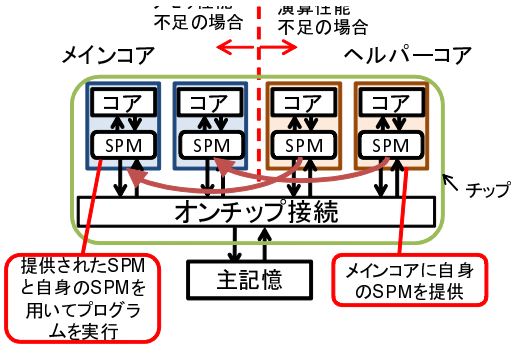

図 1 前提とするマルチコア・プロセッサモデルと提案手法の概念図

gine(Cell/B.E.) に実装し，その有効性を評価する．本稿で提案するデータ割当て方式は事前実行により取得したプロファイル情報を使用して主記憶アクセス回数を最小にする．また，最適な実行コア配分は，プロファイル情報を用いて性能モデリングを行うことで算出する．

本稿の構成は以下の通りである．第 2 節では，メモリ貸与法の概要とこれまでの実装方法の問題点をいう．次に第 3 節で，これまでの問題点を改良する方式を提案する．その後，第 4 節でベンチマーク・プログラムによる定量的な評価を行い，第 5 節でまとめる．

2. メモリ貸与法とその問題点

2.1 対象マルチコア・プロセッサ

図 1 に前提とするマルチコア・プロセッサモデルを示す．各コアはそれぞれソフトウェア制御のオンチップメモリである SPM (Scratch-Pad Memory) を搭載する．各 SPM はオンチップネットワークにより接続され，SPM のデータの入替えは DMA (Direct Memory Access) 転送によって行われる．なお，コアでの命令実行と DMA 転送はオーバラップすることができる．また，アドレスを指定することにより，SPM －主記憶間ならびに SPM －SPM 間の DMA 転送が可能である．

2.2 メモリ貸与法の概要

2.2.1 基 本 概 念

一般的に，マルチコア・プロセッサでは全てのコアでプログラムを並列処理することで高性能化を狙う．しかしながら，必ずしも高性能を達成できるわけではない．メモリボトルネックが顕著に表れる場合には，実行コア数に見合った性能が得られない．このような場合，一部のコアをメモリ性能向上用に活用することで，より高い性能を実現できる．

演算/メモリ性能バランスを考慮したメモリ貸与法 (以下，メモリ貸与法と略す) では，各コアを以下のように使い分ける．

- メインコア: 並列化プログラムを実行.
- ヘルパーコア: 自身の SPM を他のメインコアに提供 (並列化プログラムの実行は行わない).

つまり，図 1 に示すように，ヘルパーコアは自身の SPM をメインコアに貸与し，プログラム実行を行わない．そして，提供された SPM は，メインコアにおける自身の SPM －主記憶間の階層メモリとして利用される．つまり，メモリ貸与法を適用した場合，従来手法において主記憶アクセスが発生する場面において，オンチップメモリ間転送により高速にデータを供給することが可能である．つまりヘルパーコア数が多いほど，メモリ性能の向上を達成できる．しかしながら同時に，メインコアが減少するため，演算性能が低下する．そこで本手法では，図 1 のようにプログラムの特性に応じて，メインコアとヘルパーコアの比を適切に決定する．

2.2.2 DMA 転送先の判別法

前節で説明したとおり，メインコアは，ヘルパーコアの SPM を自身の階層メモリとして利用する．これを実現するために，メインコアは DMA 転送ごとに，求めるデータが保有されている箇所を特定する．ヘルパーコアは，一定のデータサイズ単位 (以降，データブロックと呼ぶ) でデータを自身の SPM へ読込む．本手法では，アクセス先を高速に判別するために，ヘルパーコアの保持するデータブロックのアドレス（以降，タグと呼ぶ）を各メインコアの SPM に保持する．メインコアは DMA 転送時にタグを参照し，データが保有されている場所を特定する．アクセス先がヘルパーコアの SPM の場合は，アクセス先アドレスを変換し DMA 転送を行う．そうでなければ，通常通り DMA 転送を行う．このアクセス先の判定とアドレス変換は DMA 転送ごとに行うため，性能へ与える影響は大きい．ヘルパーコアの SPM のデータ割当てを細粒度で行った場合，メインコアが検索するタグ数が多くなるため，当該作業に要する時間は増える．そのため本稿では，データブロックのサイズを SPM のサイズとする．

2.2.3 貸与メモリの割当てデータの決定

メモリ貸与法では，メインコアはヘルパーコアの SPM を自身の SPM －主記憶間の階層メモリとして活用する．ヘルパーコアがメインコアの要求するデータを保有している場合，

Vol.2010-ARC-187 No.24
Vol.2010-EMB-15 No.24
2010/1/29

メインコアは高速なオンチップメモリ間転送によりデータを取得できる．そのため，ヘルパーコアの SPM により有用なデータを読込むことで，メインコアのメモリ性能を改善することができる．

貸与メモリの割当てデータは，メインコアの DMA 転送によるストール時間が極力小さくなるように決定する．ただし，割当てデータはプログラムの実行前に決めなければならない[⋆1]．割当ての決定は，以下の手順で行われる．まず，事前実行やソースコード解析などにより，メインコアのプログラムの分析を行う．次に分析結果を用いて，DMA 転送対象データを貸与メモリへ読込むことによるストール時間の削減量の大きいデータを求める．最後に，求めたデータを貸与メモリの割当てデータとする．

2.2.4 最適なコア配分の予測

メモリ貸与法では，適切なメインコア数，ヘルパーコア数の配分でプログラム実行ができなければ，性能向上は達成できない．最適なコア配分は実行するプログラムによって変化する．しかしながら，コアの配分はプログラム実行前に決める必要がある．したがって，何らかの方法によりプログラム実行前にコア配分ごとの性能を見積もり，最も性能の高いコア配分を求める必要がある．

2.3 メモリ貸与法の現状と問題点

本節では，これまで我々が提案したメモリ貸与法[4] の現状と問題点について整理する．まず最初に貸与メモリの割当てデータの決定法について整理し，次に最適なコア配分の予測について議論する．

2.3.1 貸与メモリの割当てデータの決定法

これまでの貸与メモリへの割当てデータは，非常に簡易な方法で決定されていた．具体的な指定方法としては，「ループ内で使用される共有データの中で最も小さいアドレスからヘルパーコアの SPM 容量分」としていた．このような割当ては，連続したアドレスにアクセスのあるプログラムに対して有効な割当てデータを選択できる．しかしながら，想定していないプログラムに対しては効果的な割当てデータを選択することはできない．例えば，最も小さいアドレスを持つ共有データより非常に大きいアドレスのデータに頻繁にアクセスがある場合，主記憶アクセスの削減効果が低い．

2.4 最適なコア配分の予測

最適なコア配分は，複数回の事前実行を元に性能予測を行うことで求められる．具体的な方法としては，異なるメインコア数とヘルパーコア数の配分で事前実行を数回行い，それぞれの実行時間を取得する．そして，取得した実行時間を用いて，メインコア数を変数とする二次関数でメモリ貸与法の実行時間を近似する．その後，最も実行時間の短いメインコアとヘルパーコアの配分を最適なコア配分とする．この方法の問題点は，事前実行回数が多いことである．最低でも 3 回対象プログラムの事前実行が必要であるため，プログラムの解析時間が長い．

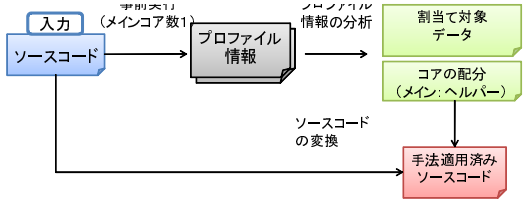

図 2 メモリ貸与法適用までのソースコードの変換手順

3. メモリ貸与法の改善

3.1 メモリ貸与法の適用手順

本稿で提案するメモリ貸与法はソースコードを図 2 の手順で変換することで適用される．まず，メインコア数 1 で事前実行を行い，プログラムの並列実行部の割合や DMA 転送のトレースなどのプロファイル情報を取得する．次にプロファイル情報を用いて，ヘルパーコアの SPM へ割当てるデータと，メインコア数とヘルパーコア数の配分を決定する．最後に，これらの予測結果を用いてソースコードを変換し，メモリ貸与法が適用されたソースコードを得る．次節にてヘルパーコアの SPM のデータ割当て決定法を説明する．その後第 3.3 節にて，最も性能が高くなるメインコア数とヘルパーコア数の配分の予測方法を説明する．

3.2 貸与メモリの割当てデータの決定法

本方式の貸与メモリの割当てデータ決定法では，プログラム実行中にデータを入替えない前提で，メインコアの DMA 転送時間最小を目的として SPM のデータ割当てを決定する．メインコアの DMA 転送時間は共有資源でのアクセス競合などによって変化する．したがって，プログラム実行前に DMA 転送時間の変化を求めることは難しい．そこで本稿では，メインコアの主記憶に対する DMA 転送時間は等しいものとして，データ割当てを決定する．

割当てデータは以下の手順で決定される．まず，あらかじめ，メインコア数 1 で事前実行

⋆1 貸与メモリにソフトウェアキャッシングを適用した場合を除く

Vol.2010-ARC-187 No.24
Vol.2010-EMB-15 No.24
2010/1/29

を行い，各データへの DMA 転送回数を取得しておく．次に，ヘルパーコアのデータブロック単位で DMA 転送回数を集計する．最後に，DMA 転送回数の多いデータブロックから順に貸与メモリへの割当てデータとする．これにより，メインコアの主記憶アクセス回数を最小にできる．

本節で説明した方式は，コンパイル時に自動で SPM のデータ割当てを行う Static allocation[1)] をメモリ貸与法向けに修正したものである．おもな修正点はデータ割当ての粒度である．文献 1) ではデータ割当ては変数単位で行う．これに対してメモリ貸与法では，データ割当てを一定サイズごとに行い，また割当てサイズを大きくしている．これは DMA 転送先判別のオーバーヘッドの削減するためである．

3.3 最適なコア配分の予測法

最適なコア配分は，事前実行によるプロファイル情報に基づき，性能予測を行うことで求める．まず，コア配分決定方法に使用する性能モデル式の導出を行う．あるメインコアの実行時間 $T(m,n)$ は，演算実行に要する実行時間 $T_{exe}(n)$，DMA 転送に要する時間 $T_{mem}(m,n)$ を用いて，

$$T(m,n) = T_{exe}(n) + T_{mem}(m,n) \tag{1}$$

と表すことができる．m,n はそれぞれヘルパーコア数とメインコア数である．ここでは演算と DMA 転送は同時に実行できないものとする．$T_{exe}(n)$ と $T_{mem}(m,n)$ は以下のように表すことができる．

$$T_{exe}(n) = T_{exe}(1) \times \left(\frac{F}{n} + 1 - F\right) \tag{2}$$

$$T_{mem}(m,n) = AC(n) \times (HR_{SPMR}(m) \times AT_{SPM} + (1 - HR_{SPMR}(m)) \times AT_{main}) \tag{3}$$

各項の定義は以下のとおりである．

- F: プログラム中の逐次実行部分の実行時間が，全体のプログラム実行時間に占める割合 (n=1 のとき)
- AC: DMA 転送回数
- AT_{SPM}:ヘルパーコアの SPM アクセスに要する時間
- AT_{main}: 主記憶アクセスに要する時間
- HR_{SPMR}: メインコアが要求するデータがヘルパーコアの SPM に存在する確率

ここで，共有資源においてアクセス競合による待ち時間が発生しないと仮定すると，上記の項のうちメインコア数およびヘルパーコア数によって変化する項は，$AC(n), HR_{SPMR}(m)$ である．$AC(n)$ を以下の式によって表す．

$$AC(n) = AC(1) \times \left(\frac{F}{n} + 1 - F\right) \tag{4}$$

一方，$HR_{SPMR}(m)$ は，m に対するヒット率の変化は，プログラム中の各区間におけるヘルパーコアのデータ割当てと DMA 転送のトレースにより求めることができる．

メインコア数 1 における事前実行により得たプロファイル情報をもとに，上記のモデル式を用いて $T(m,n)$ を求める．その後，最も $T(m,n)$ の小さい m,n の組合せを算出する．式中の $F, AC(1), HR_{SPMR}(m)$ は事前実行により抽出したプロファイル情報をもとに算出する．F は，プログラム中の逐次実行部の実行時間を時間測定用の関数により取得し，プログラム全体の実行時間との比により求める．また $AC(1)$ はプロセッサに搭載されているハードウェアカウンタの値を利用して求めることが可能である．$T_{exe}(1)$ は，式 (1) に $m = 0, n = 1$ を代入することで算出でき，残りの項はデータシートなどを参考に値を決定する．

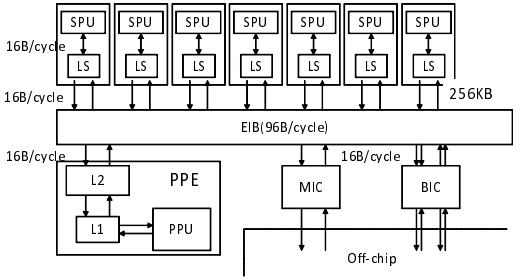

図 3 Cell/B.E. のブロック図

4. 評　　価

4.1 評 価 環 境

提案手法の評価には，Cell Broadband Engine (Cell/B.E.) を用いた．Cell/B.E. は図 3 のように 8 個の SIMD 型コア (SPE) と 1 個の汎用コア (PPE) で構成される．各 SPE は 256KB の SPM を持つ．本評価で使用した実機では動作する SPE は 7 個である．この 7 個の SPE をメインコアまたはヘルパーコアとして使用する．SPM のデータ入替えは DMA 転送によって行われる．DMA 転送によるデータ書込み時間は周りのコアの SPM へ書込む

Vol.2010-ARC-187 No.24
Vol.2010-EMB-15 No.24
2010/1/29

表 1 各モジュールのアクセス時間

アクセス対象	アクセス時間 [CC]
周りのコアの LS (AT_{SPM})	106+転送サイズ/8
主記憶 (AT_{main})	300+転送サイズ/4

表 2 ベンチマーク・プログラムの入力

プログラム名	入力
HIMENO	$17 \times 17 \times 33$ $33 \times 33 \times 65$
SUSAN	Large input
LU	512×512
FFT	229376 point
MATRIX_MUL	256×256 512×512

場合と主記憶へ書込む場合で変わらない．そこで，主記憶からの読込み時間が最小となるようにヘルパーコアのデータ割当てを決定する．性能モデル式に使用する各モジュールのアクセス時間は，表 1 を使用する．これらの値は，文献[2] ならびに文献[5] を参考に決定した．実行時間はメインコアの実行時間のうち最も長いものとする．

評価には，5 つのマルチスレッドのプログラムを用いた．Cell/B.E. はスーパーコンピュータのアクセラレータから，ゲームコンソールまで多くの分野で使用されていため，複数の分野からベンチマークプログラムを選択した．これらのプログラムのソースコードを Cell/B.E. 向けに修正した．HIMENO[7] では，3 重ループの 2 番目のループで並列化を行った．MiBench[3] の SUSAN では，(1) 初期化，(2) エッジ強調，(3) エッジ補正，(4) 原画像重ね，の 4 ステップのうち，(2) (4) を並列化した．SPLASH-2[6] の LU はもとのプログラムが並列化されているため，並列化アルゴリズムには手を加えていない．これら 3 つのプログラムでは，ダブルバッファリングなどの最適化技術を適用し，可能な限りメモリ性能が向上するようにチューニングを行った．FFT と MATRIX_MUL は IBM Cell SDK 3.1 のサンプルコードからソースコードを取得した．FFT はコア数を 2 のべき乗以外で実行できるように修正した．これらのベンチマークプログラムの入力を表 2 に示す．

Cell/B.E. における提案手法の効果を議論するため，以下のような評価モデルを定義する．

- **CONV**：従来の単純並列実行モデル．全てのコアを利用して並列化プログラムを実行する（つまり，メインコア数 7，ヘルパーコア数 0 としてプログラム実行）．
- **PB-PREDICT:** 現実的なコア分配を前提としたモデル．第 3.3 節で提案した方式に基づきメイン/ヘルパーコア配分を決定する．事前実行には評価対象と同一の入力データを使用する．ヘルパーコアの SPM のデータ割当ては第 3.2 節で提案した方式を用いる．
- **PB-IDEAL**：理想的なコア配分を前提としたモデル．同一入力を用い，全てのメインコア数とヘルパーコア数の組合せについて事前実行を行う．これにより最も高い性能を実現するコア分配を実施する．ヘルパーコアの SPM のデータ割当て方法は PB-PREDICT と同じである．

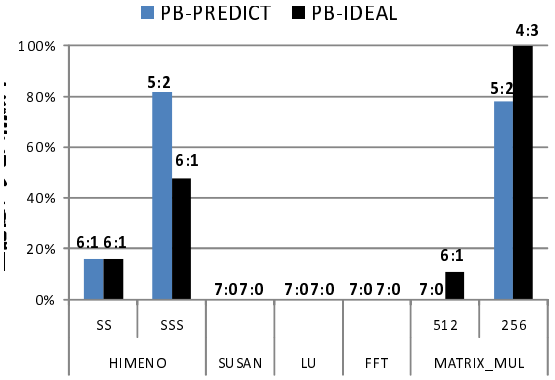

図 4 主記憶アクセスの削減率

4.2 主記憶アクセス削減効果

図 4 に各ベンチマークプログラムにおけるメモリ貸与法による主記憶アクセスの削減率を示す．バーは左から PB-PREDICT，PB-IDEAL である．バーの上にある数字はメインコアとヘルパーコアの比を示す（メインコア数：ヘルパーコア数）．グラフより，HIMENO ならびに MATRIX_MUL では，主記憶アクセスを削減できていることが分かる．同一プログラムであっても入力サイズの小さいプログラムのほうが主記憶アクセスの削減率が高い．これは，入力サイズが小さいとプログラム中に使用するデータ容量が小さくなり，効率よく貸与メモリへデータを割当てることができたためである．その他のプログラムでは，主記憶アクセス削減効果が得られていない．これは，従来の全コア実行が最も性能が高いと予測したためである．この場合ヘルパーコアがないため，メインコアの主記憶アクセス削減効果は得られない．

4.3 性　　能

図 5 に各評価対象モデルに対する性能向上を示す．縦軸は CONV モデルを 1 としたときの相対性能である．横軸はベンチマークプログラム名と入力を表す．各バーは左から，CONV，PB-PRDICT，PB-IDEAL を表す．また，バー上にある数字は，メインコア数とヘルパー

Vol.2010-ARC-187 No.24
Vol.2010-EMB-15 No.24
2010/1/29

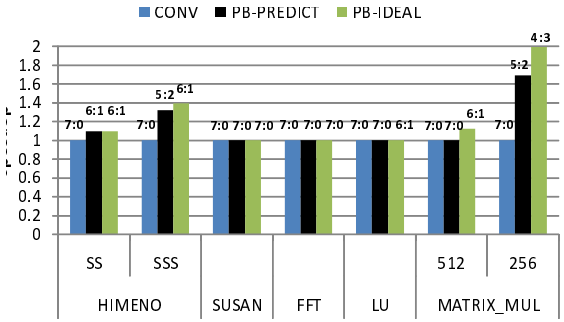

図 5 提案方式による性能向上

コア数を示す．

まず，PB-PREDICT の性能向上について議論する．メモリ貸与法を適用することで，HIMENO, MATRIX_MUL (256) において，高性能化を達成している．これは，オフチップメモリアクセス削減による性能向上が，メインコア減少による性能低下を上回ったためである．その他のプログラムではメモリ貸与法の適用により性能向上を達成できていない．これは，従来の全コア実行が，最も性能が高いメインコアとヘルパーコアの配分と予測したためである．文献 4) における結果と比較すると，HIMENO(SS) では本稿の手法の方が 4%性能が低下している．HIMENO は連続した配列に対してアクセスが繰り返されるプログラムであり，従来の割当て手法により主記憶アクセスを効果的に削減できる．つまり，従来手法に対して有利なプログラムと比較しても性能低下は小さい．

次にコア配分の予測精度について議論する．HIMENO(SS), SUSAN, FFT, LU では PB-PREDICT と PB-IDEAL の結果が等しい．つまり，これらのプログラムでは最適なコア配分を予測できている．また，その他のプログラムにおいても二番目に最適なコア配分を算出できており，PB-PREDICT と PB-IDEAL の性能差は抑えられている．従来手法[4] と比較しても，同等の精度[⋆1]が得られている．つまり，事前実行回数を減らしてもコア配分の予測精度は落ちていない．

⋆1 従来：4 個中 2 個正解　今回：8 個中 5 個正解

5. おわりに

本稿では，以前提案したマルチコア向けオンチップメモリ貸与法において，貸与メモリのデータ割当て法ならびに最適なコア配分の予測方法を改良した．具体的には，貸与メモリのデータ割当てを主記憶アクセス回数最小となるように決定し，またコア配分の予測に必要な事前実行回数を削減した．本手法を実装し，評価した結果，最大で 63%の性能向上を達成した．今後は，ヘルパーコアの割当てデータを改良することで，さらなる高性能化を目指す．

謝辞　日頃から御討論頂いております九州大学安浦・村上・松永・井上研究室ならびにシステム LSI 研究センターの諸氏に感謝いたします．なお，本研究は一部，半導体理工学研究センター（STARC) ならびに科学研究費補助金（課題番号：21680005）との共同研究による．

参 考 文 献

1) O. Avissar, R. Barua, and D. Stewart, An optimal memory allocation scheme for scratch-pad-based embedded systems. ACM Transactions on Embedded Computing Systems, pp.6-26, 2002.
2) T. Chen, R. Raghavan, J. N. Dale, and E. Iwata. Cell Broadband Engine Architecture and its First Implementation Performance View. IBM Journal of Research and Development, 51(5): pp.559-572, 2007.
3) M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge and R. B. Brown. MiBench: A Free, Commercially Representative Embedded Benchmark Suite. The IEEE 4th Annual Workshop on Workload Characterization, pp.3-14, 2001.
4) 林徹生，福本尚人，今里賢一，井上弘士，村上和彰. 演算/メモリ性能バランスを考慮した Cell/B.E. 向けオンチップメモリ活用法とその性能評価，情報処理学会 第 170 回 ARC 研究会, pp.105-110，2008 年
5) M. Kistler, M. Perrone, F. Petrini. Cell Multiprocessor Communication Network: Built for Speed. Micro, IEEE, 26(3), pp. 10-23, 2006.
6) S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. The Intl. Symposium on Computer Architecture, pp.24-36, 1995.
7) Himeno Benchmark: http://accc.riken.jp/HPC/HimenoBMT/index_e.html