HPC 向けオンチップメモリプロセッサアーキテクチャ SCIMA の SMP 化の検討と性能評価

高橋睦史[†1] 近藤正章[†2,†3] 朴泰祐[†4]
高橋大介[†4] 中村宏[†2] 佐藤三久[†4]

SCIMA はプロセッサとオフチップメモリの性能格差に起因する問題の解決を目的とする HPC 向けアーキテクチャであり，オンチップメモリを利用した明示的なデータ転送のプログラミングが可能となっている．一方，近年の HPC 向けアーキテクチャでは SMP 構成が不可欠となっているが，それはオフチップメモリへのアクセストラフィックの増大をもたらし，深刻な性能低下を招く恐れを生じている．そこで本論文では，SCIMA の SMP 化への対応を検討し，その構成について性能評価を行った．その結果，SCIMA による大きな粒度でのデータ転送とバストラフィックの削減によって，より大きなスケーラビリティが得られることが分かり，SMP 構成においても SCIMA が有効であることが分かった．

SMP Configuration and Performance Evaluation of SCIMA ——On-chip Memory Processor Architecture for HPC

Chikafumi Takahashi,[†1] Masaaki Kondo,[†2,†3] Taisuke Boku,[†4]
Daisuke Takahashi,[†4] Hiroshi Nakamura[†2]
and Mitsuhisa Sato[†4]

SCIMA is a processor architecture equipped with addressable on-chip memory to solve the memory-wall problem caused by processor/memory performance gap. In this paper, we propose to modify current SCIMA for SMP-enabled configuration and present preliminary performance evaluation on SMP-SCIMA system. As a result, it is shown that SCIMA's feature to reduce the off-chip traffic and optimize the data transfer granularity improves the performance on several benchmarks even with SMP configuration compared with traditional cache-only architecture.

1. はじめに

SCIMA（Software Controlled Integrated Memory Architecture）[1),2)] はプロセッサチップ上に従来のキャッシュに加えて，データ転送のプログラミングが可能なメモリ（software controllable on-chip memory，以降 SCM と呼ぶ）を導入することにより，HPC アプリケーションにおいて最も深刻であるプロセッサ/オフチップメモリ間のデータ転送量を最小化し，かつデータ転送粒度を最適化することを目的としたプロセッサアーキテクチャである．SCIMA に関しては，これまでに単一プロセッサ構成を基本とした性能評価が行われ[3),4)]，キャッシュのみを持つプロセッサに対する優位性が示されてきた．

一方，近年の HPC 向けアーキテクチャでは，相互結合網上のノード規模を抑えつつプロセッサ数を増やすため，SMP 構成をとることが不可欠となっている．SMP 化された計算ノードでは，ネットワークインタフェースの共有化やシステムのコンパクト化が図られるが，その反面，オフチップメモリに対するアクセストラフィックの増大により，深刻な性能低下を招く恐れがある．SCIMA では，このようなアクセストラフィックを最小化することがその本質であるため，これを SMP 化することにより，通常のキャッシュ型アーキテクチャ（以降，CACHE only）に比べ，ノード内

†1 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
†2 東京大学先端科学技術研究センター
Research Center for Advanced Science and Technology, The University of Tokyo
†3 科学技術振興事業団
Japan Science and Technology Corporation
†4 筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

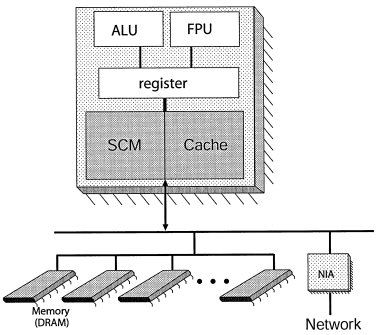

図 1 SCIMA の構成図
Fig. 1 Structure of SCIMA.

プロセッサ数に対する高いスケーラビリティを得られることが予想される．

そこで，SCIMA の基本アーキテクチャを SMP に対応させ，そのような構成における予備的な性能評価を行うことが本論文の目的である．

2. SCIMA

図 1 に，SCIMA の構成を示す．SCIMA では，プロセッサチップ上にキャッシュに加えて SCM を搭載する．キャッシュはハードウェア制御により暗黙的にデータ転送が操作されるのに対し，SCM はソフトウェアにより明示的に転送を指示することが可能である．また，単純な連続アドレスに対するアクセスだけでなく，ストライド転送等を指定できるようになっている．

SCIMA では，論理アドレス空間上に SCM をマッピングする．SCM に割り当てられたアドレスはオフチップメモリ上では無効となり（すなわち，SCM とオフチップメモリには包含関係はない），これを管理するために SCM の開始アドレスを保持する ASR（SCM Address Start Register）と SCM 容量を保持する AMR（SCM Address Mask Register）の 2 種類のレジスタが導入されている．SCM 以外のアドレスへのアクセスは通常のプロセッサ同様にキャッシュを通じて行われるか，もしくはキャッシュを通さずレジスタ/オフチップメモリの直接データ転送が行われる．

また SCIMA では，SCM/オフチップメモリ間のデータ転送を指定するために，page-load/page-store 命令（以下，p-load/p-store）が拡張命令として加えられる．この命令は，page（通常の仮想記憶におけるページとは異なる）と呼ばれる比較的粒度が大きな SCM の管理単位に対するデータ転送を指示するものである．また，この命令はブロックストライド転送機能を備え，たとえば多次元配列に対するアクセスにおいて，キャッシュで生じるような無駄なデータ転送を極力抑えることが可能となっている．さらに，必要に応じてキャッシュラインよりもはるかに大きなデータブロックを転送することにより，データ転送のオーバヘッドを相対的に小さくすることも可能になっている．

SCIMA ではキャッシュと SCM をハードウェア的に統合し，複数の way で構成されているオンチップメモリに対して，way 単位でキャッシュと SCM を切り替えることが可能である[1)]．これを利用し，アクセスが定型的なデータについては SCM を，定型的ではないが再利用性があるデータについてはキャッシュを利用し，アクセスが不定形かつ再利用性に乏しいデータについては，レジスタ/オフチップメモリの直接データ転送を行うことができる．

3. SMP 化の検討

3.1 アドレス空間

従来の SCIMA では，SCM とオフチップメモリを，同一のアドレス空間における排他的な領域として定義してきた．SMP 化においては，当然オフチップメモリの領域についてはアドレス空間がプロセッサ間で共有化される．しかし，SCM を共有とすることは，データの局所性を最大限に生かすという SCIMA の基本概念に反することになる．したがって，SCM はプロセッサごとに設けるべきである．このため，以下の 2 つの方針が考えられる．

(1) 各プロセッサにおける SCM にマップされた空間のアドレスは同一とし，その領域に対するアクセスはそのプロセッサ固有の SCM に対するアクセスとなる．
(2) アドレス空間上にプロセッサ台数分の SCM 領域を確保し，各プロセッサにはその一部がそれぞれ割り当てられる．他のプロセッサの SCM 領域にマップされた領域にもアドレスは割り当てられるが，その領域へのアクセスは不可能である（segmentation violation となる）．

(1) の方法はシンプルではあるが，SPMD プログラム上で混乱を生じやすい．我々はユーザが意識的に SCM へのアクセスを記述することを想定し，概念的に分かりやすい (2) の方法を選択することにした．各プロセッサには従来と同様に ASR および AMR が存在するが，その内容はプロセッサごとに異なる（オンチップメモリサイズの異なるプロセッサが混在することも，理論的には可能である）．4 プロセッサによる SMP 構成の場合の構成とアドレス空間の様子を**図 2** に示す．

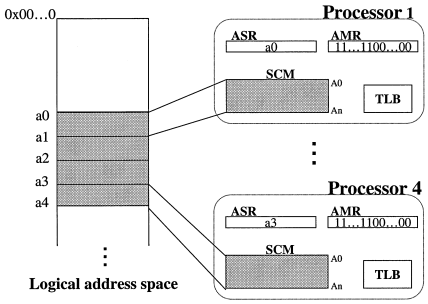

図 2 SMP 構成によるアドレス空間
Fig. 2 Address space of SMP-SCIMA.

3.2 SCM 間データ転送

前節で述べたアドレス空間構成では，あるプロセッサから他のプロセッサの持つ SCM に直接アクセスすることができない．SCM は高速にアクセスすることのできるプロセッサ上のメモリであり，仮にプロセッサ間においてオフチップメモリを介さない直接のデータ転送を行うことが可能であるならば，さらなる性能が得られる可能性がある．しかし，これを実現するハードウェアを実装することは，メモリアクセス機構を非常に複雑にする．また，SCIMA 自体がデータセットの非常に大きな HPC アプリケーションをターゲットとしているので，SCM 間で小さな量のデータを交換できることがどの程度有効であるかという点で疑問が残る．したがって現時点では，あるプロセッサは自プロセッサ内の SCM にのみアクセスできることとし，他プロセッサ内の SCM に対する直接のデータアクセスは，各種アプリケーションの要求などを考慮し今後検討していく予定である．

4. SCM プログラミング

本章では，SCIMA を適応した SMP マシン向けプログラムの例として，行列積演算と NAS Parallel Benchmarks[5]（以下 NPB）の Kernel CG を取り上げ，SCIMA を利用したプログラミングと，それによって得られる利点を示す．

4.1 行列積演算

まず，ここでは $C = A \times B$ で表される，N 次の倍精度浮動小数の行列積演算について取り上げる．行列積演算を単純に記述すると 3 重ループ構造となるが，キャッシュヒット率を考えた場合に時間的局所性を生かすことができない．そこで最適化手法の 1 つとしてキャッシュブロッキング[6]（タイリング）がしばしば用いられる．

並列化としては，行列 C についての 1 次元もしくは 2 次元分割が考えられる．比較的少数のプロセッサでの並列化では，1 次元分割で十分である．よって，これ以降は 1 次元分割で並列化された後の，各プロセッサにおける最適化について考える（**図 3** 参照）．行列 C を行方向に分割することで，1 つの CPU あたりのデータセットは，分割された C と同様に分割された A，加えて B 全体となる．この場合，分割後の各ブロックに関しては前述のキャッシュブロッキングが適用できる．これは，行列をいくつかの小行列に分割し，各フェーズのワーキングセットをキャッシュに収めるようにする方法である．

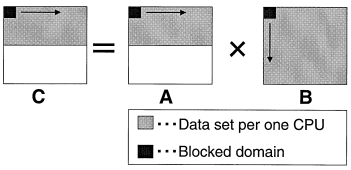

図 3 行列積の並列化（2CPU 時）
Fig. 3 Parallelization of matrix multiply (case of 2 CPUs).

しかし，行列積演算にタイリングを用いた場合，アドレスが不連続になる方向のアクセスにおいてラインコンフリクトが生じる可能性がある．通常，ラインコンフリクトが発生した場合，n-way set assiciative なキャッシュ制御を用いたり，データのアクセスパターンにソフトウェア的な変更を行うことでコンフリクトを回避する．行列積においては行列の転置を行うことによりコンフリクトが回避できる．

SCIMA 向けアルゴリズムでは，CACHE 向け最適化で行った，キャッシュブロッキングされた A, B, C すべての行列の領域について，p-load/ p-store を用いて SCM とのデータ転送を記述してやればよい．この場合，SCM ではデータの配置や置き換えがソフトウェアによる制御で行われるため，コンフリクトミスは発生しない．さらに，SCIMA ではストライドアクセスに対応した明示的なデータ転送ができるので，意図しないデータ転送は発生せず，転送量を必要最小限に抑えられる．また，これらの特性を活かしつつ，SCM/オフチップメモリのデータ転送粒度をハードウェアが許す限り大きくできるため，キャッシュのみのシステムに比べバスまたはメモリアクセストランザクションのオーバヘッドを最小化できる．

4.2 NPB Kernel CG

NPB Kernel CG は，正値対称な大規模疎行列の固有値を CG（Conjugate Gradient）法によって求めるベンチマークである．この固有値は 2 重ループで収束させながら計算されるが，その計算に必要な実行サイクル数のうち大部分は疎行列とベクトルの積を

求めることに費やされる．この行列のベクトル積は $q = \sum_i (A[i] \times p[colidx[i]])$ という形で，ベクトル q の 1 要素を求めるのに，行列 A への連続アクセスと，ベクトル p の要素への間接アクセスを必要とする．Class W のベンチマークにおいては，A のサイズは 7000 × 7000 で非零率 1%の疎行列，p は $colidx$ によってランダムに間接参照される 7000 要素のベクトルである（精度は倍精度浮動小数）．

並列化に関しては，行列 A に着目して，1 次元，もしくは 2 次元分割が行える．本評価では比較的少数のプロセッサの SMP 構成を考えるため，1 次元行分割のみを考える．以降では，この条件下での各プロセッサにおける最適化を考察する．

図 4 に示すような並列化を行った後，各プロセッサにおいて CACHE only で前述の行列・ベクトル積演算を行うと，順次アクセスではあるが再利用性のない A と，$colidx$ を用いた間接参照によるランダムアクセスのため局所性が活かせない p のキャッシュミスが多発する（このオリジナル版を以下 CACHE-org と呼ぶ）．このとき，ベクトル p をブロック分割し，その分割したブロックにアクセスするよう A と $colidx$ をあらかじめ p のブロックにあわせて並べ替え，p の参照に局所性を発生させることによって，p についてはキャッシュブロッキングによる性能改善が可能となり，キャッシュミスを軽減できる[4]（この手法を以下 CACHE-opt と呼ぶ）．しかし，A に関してはデータの再利用性がないのでキャッシュブロッキングによる性能改善は見込めない．

ここで NPB Kernel CG に SCIMA の適用を考えると，先ほどキャッシュブロッキングを行った p を SCM に転送することができる．加えて行列 A や，$colidx$ も SCM に転送することができる（この手法を以下 SCIMA-opt と呼ぶ）．CACHE only ではブロッキングを行った p と連続アクセスとなる A を比較的小さな粒度であるラインサイズ単位でデータ転送を行っているのに対し，SCIMA では page という大きな粒度の単位で転送できるので，SCIMA の適応によりメモリアクセスのレイテンシによるオーバヘッドの影響がキャッシュと比較して相対的に小さくなる．また，明示的に SCM/オフチップメモリ間のデータ転送を指定できるので，キャッシュにおけるラインごとのデータ転送のように不要なデータを転送することなく効率的にバスを使用することが可能である．加えて，ラインコンフリクトが起きる心配もない．

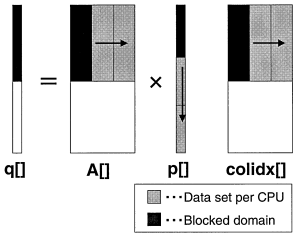

図 4 NPB Kernel CG の並列化（2CPU，3 分割時）
Fig. 4 Parallelization of NPB Kernel CG (case of 2 CPUs, divided into 3 blocks).

5. 性能評価環境

SMP 化された SCIMA の性能を評価するため，計算機シミュレーションによる評価を行う．本論文では，SMP 化された SCIMA に対する並列プログラミング環境として，スレッドプログラミングの代表的なライブラリである Pthreads によるプログラミングを考える．将来的には OpenMP や自動並列化コンパイラ等によるコード生成も検討する予定であるが，先述のアドレス空間において SCM に対するデータ転送を記述する都合上，スレッドによる明示的な並列プログラミングが現時点では有効である．

評価用のターゲットプログラムには，前章で示した行列積演算と NPB Kernel CG を用いる．行列積演算は HPC 分野に置いて多用される処理であり，また，NPB は非常に多くの並列計算機で評価がなされていることから，この両者は性能評価の指標として有効であると考えられる．

本論文で用いたシミュレータは，MIPS IV の ISA（命令セット）を拡張した ISA を用いる既存の SCIMA シミュレータ[1]を元に，バス結合型 SMP マシンのシミュレーションが行えるような変更を加えたものである．キャッシュは L1 キャッシュをシミュレートし，キャッシュコヒーレンスには MSI プロトコル[7]を採用する．並列プログラムに関しては，基本的な Pthreads ライブラリに対応する．また，命令キャッシュはつねにヒット，分岐予測はつねに成功という条件でシミュレーションを行う．また，リザベーションステーションを用いた out-of-order 実行もサポートしている．

p-load/p-store のような SCIMA 専用の命令は関数の形で擬似的にプログラムに挿入し，評価には既存の MIPS 用コンパイラを用いる．拡張命令と Pthreads に関する命令は，プリプロセッサを用いてアセンブラコードレベルで ISA 中で使用していない擬似命令に変換，バイナリにコンパイルする．シミュレータはこのバイナリコードを読み込み，擬似命令をフックする

ことでシミュレーションを行う．

シミュレータに与える共通のパラメータに関しては，特に指定がない場合は以下の値を使用する．

- レジスタ数：
  - 整数：32
  - 浮動小数点：32
- 演算ユニット数：
  - 整数演算：2
  - 浮動小数点演算（積和）：1
  - 浮動小数点演算（除算/平方根）：1
  - load/store：1
- reservation station エントリ数：
  - 整数：32
  - 浮動小数点：32
  - load/store：32
- load/store 命令レイテンシ：2 cycle
- バスバンド幅：4 B/cycle
- SCM ページサイズ：4 KB

キャッシュおよび SCM の容量に関しては，CACHE only と SCIMA の比較をするため，**表 1** の 2 つの構成のパラメータを与え，チップ上のメモリを 8 KB × 4 way とし，SCIMA ではそれぞれの way をキャッシュもしくは SCM に切り替えて演算を行うことを想定する．キャッシュの連想度は，表 1 で示すようにキャッシュとして利用する way 数と同一とする．また，オフチップメモリのアクセスレイテンシおよびキャッシュラインサイズは重要なパラメータのため，シミュレーション条件として別途与える．なお，ある CPU がオフチップメモリアクセスをリクエストすると，そのリクエストが終了するまでバスは占有され，他の CPU はオフチップメモリアクセスを実行できないものとしてシミュレーションを行う．

また，行列積プログラムに与えるブロッキングサイズは，キャッシュもしくは SCM 容量に 1 ブロックが収まるように以下の値を与える．

- CACHE only：36 × 36
- SCIMA：32 × 32

NPB CG の CACHE-opt および SCIMA-opt についてのブロッキングサイズは，いくつかの実験の結果，ともに 1000 要素（7 分割）とした．これはオンチップメモリの 1 way（8 KB）分のサイズである．

ここで用いるパラメータでは，SCM やキャッシュのサイズが現実よりかなり小さくなっている．これはシミュレーションの効率を上げ，かつデータの動きを観察しやすいようにするためである．しかし，ここにあげるプログラミングや最適化手法は基本的にこれらのサイズに依存しないため，示される結果も本質的には現実のそれに一致すると考えられる．ただし，行列積演算での評価では行列のサイズが小さいことから，行列の転置のオーバヘッドがその利点を上回ると考えられるので，転置は行わない．

表 1 シミュレーションパラメータ
Table 1 Simulation parameters.

	Cache size	OCM size
CACHE only	32 KB (4 way)	0 KB
SCIMA	8 KB (1 way)	24 KB

OCM: On-Chip Memory

6. 性能評価結果

6.1 行列積演算

図 5 に，行列サイズ 300 の演算時におけるキャッシュラインサイズが 32 B および 128 B，オフチップメモリのアクセスレイテンシが 40 cycle 時の，実行サイクル数を示す．実行サイクル数の内訳は，オフチップメモリのスループット不足に起因するストールサイクル数を Throughput stall（T_t），オフチップメモリのアクセスレイテンシによるストールサイクル数を Latency stall（T_l），実際に CPU が処理を行っているサイクル数を CPU Busy Time（T_b）として表している．本論文ではこの 3 種類のサイクル数を，プロセッサの総実行サイクル数 T，オプチップメモリのスループットが無限大と仮定した場合の実行サイクル数 T_∞，オフチップメモリのスループットが無限大，かつオフチップメモリのアクセスレイテンシが 0 cycle と仮定した場合の実行サイクル数 T_p を用いて，以下のように定義する．

$$T_b = T_p$$
$$T_l = T_\infty - T_p$$

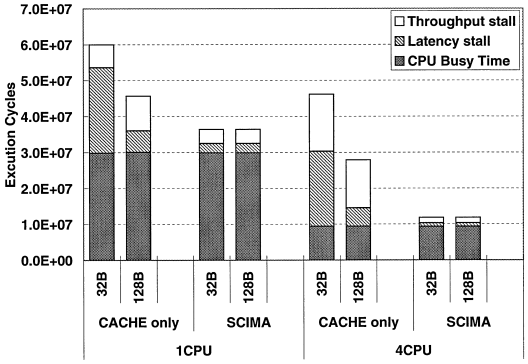

図 5 行列積におけるキャッシュライン別実行サイクル数内訳
Fig. 5 Detail of execution cycles according to cache line size on matrix multiply.

表2 行列積における総バストラフィック量（4CPU時）
Table 2 Amount of bus traffic on matrix multiply (case of 4 CPUs).

Matrix size	CACHE only	SCIMA
50	0.13 Mbyte	0.16 Mbyte
100	0.97 Mbyte	0.80 Mbyte
150	3.98 Mbyte	2.70 Mbyte
200	8.24 Mbyte	5.44 Mbyte
250	14.6 Mbyte	9.00 Mbyte
300	28.2 Mbyte	17.2 Mbyte

$$T_t = T - T_\infty$$

まず，1CPU・4CPU時を通して，CACHE onlyはSCIMAと比較して実行サイクル数が長くなっている．内訳を見ると，Throughput stallとLatency stallが実行サイクル数を増大させる要因となっていることが分かる．SCIMAでは必要最小限のデータのみ転送できるため，Throughput stallがキャッシュに比べ短縮されている．ここで，Throughput stallはほぼ総データ転送量に比例すると考えられるので，CACHE onlyでは本来必要のない，余分なデータ転送が多いことがうかがえる．

表2に，キャッシュラインサイズが32Bのときにおける行列サイズ別の総バストラフィック量を示す．これを見ると，行列サイズが一定以上の大きさの場合，CACHE onlyではSCIMAに比べて約1.5倍の総バストラフィック量があることが分かる．このとき，キャッシュラインサイズは32B，ブロッキングサイズは36要素すなわち288Bとなり，キャッシュラインサイズの倍数にあたるのでライン転送時に同一ライン内の不要なデータ転送は発生していない．したがって，トラフィックの増加は，ラインコンフリクトが発生するためであると考えられる．一方，キャッシュラインサイズが128Bの場合では，図5よりThroughput stallが32Bのときよりも多くなる．これは，ブロッキングのサイズがラインサイズの倍数になっておらず，ライン転送時に同一ライン内の不要なデータ転送が発生するためである．ラインコンフリクトに加えてこの無駄なデータ転送の影響により，総トラフィック量が増えると考えられる．

Latency stallについては，SCIMAはpageという比較的大きい粒度でデータ転送を行っているが，CACHE onlyではpageより粒度の小さなキャッシュラインサイズでデータを転送しているので，SCIMAと比較して転送回数が大きくなり，オフチップメモリのアクセスレイテンシの影響を受けやすくなっていると考えられる．加えて，キャッシュラインサイズ単位でのデータ転送に起因する意図しないデータ転送により総データ転送量も大きくなっているので，さらにアクセスレイテンシの影響を受けやすくなっていると考えられる．

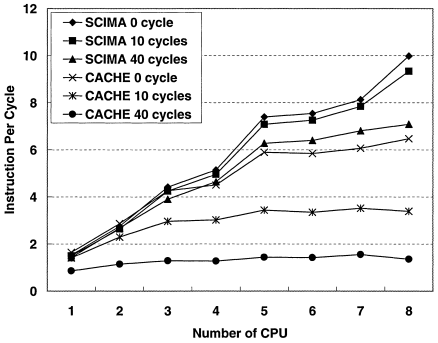

図6 行列積におけるスケーラビリティ
Fig. 6 Scalability on matrix multiply.

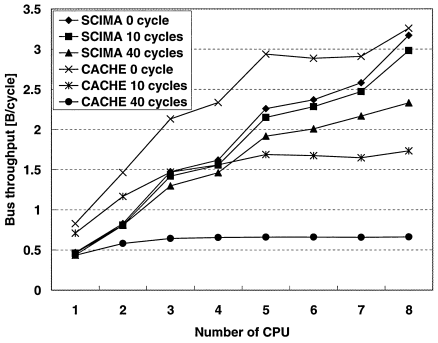

図7 行列積におけるバススループット
Fig. 7 Bus throughput on matrix multiply.

次に図5における4CPU時の総実行サイクル数の短縮を見ると，CACHE onlyでは4CPU時は1CPU時と比較して2種類のstallに占められるサイクル数が大きくなり，並列化による性能向上を妨げている．その点に着目すると，SCIMAはその2種類のstallによる影響を受けないので，CACHE onlyよりも大きなスケーラビリティが得られていると考えられる．さらに，図5に現れているが，キャッシュラインサイズの大小はThroughput stallとLatency stallのトレードオフとなって現れ，最適化が困難となることがあるが，SCIMAではキャッシュラインサイズに左右されず適切なデータ転送粒度を指定できるので，最適化が容易である．

次にスケーラビリティを評価する．**図6**は，行列サイズ300におけるオフチップメモリのアクセスレイテンシが0cycle，10cycle，40cycleと変化したときのSMP上のプロセッサ数の増加に対する1cycleあたりの実行命令数（IPC）を示している．

まずCACHE onlyに注目すると，オフチップメモリのアクセスレイテンシが大きくなるにつれてIPCが

小さくなっていることが分かる．さらにCPU数が大きくなるほど，その影響は大きくなる．この原因としては，アクセスレイテンシの増大によりデータ転送1回あたりのオーバヘッドが増大するが，SCIMAでは大きな粒度でのデータ転送により転送回数自体が少ないので，影響を受けにくいことが理由と考えられる．逆にCACHE onlyでは大きく影響を受け，CPU数が増大した場合にはオーバヘッドでバスが混雑し，演算がストールしてしまう．

また，SCIMAでは並列度の増加に従って大きな性能向上が見られるが，CACHE onlyでは並列化による性能向上がSCIMAほど得られていない．このときの1 cycleあたりのバススループットを**図7**に示す．SCIMAではCPU数の増加に従ってバススループットが増えているにもかかわらず，CACHE onlyのレイテンシ40 cycle時では約0.66 B/cycleに達するとそれ以上バスのスループットは増加しないことが分かる．これは，CACHE onlyではオフチップメモリのアクセスレイテンシの影響でバス帯域が飽和しているものと考えられる．キャッシュラインサイズは32 Bでスループットが4 B/cycle，オフチップメモリのアクセスレイテンシが40 cycleであるから，1ラインの転送に必要なサイクル数は$(32/4)+40=48$ cycleであり，1 cycleあたりのデータ転送量が$32/48 \approx 0.667$ B/cycleと理論的に求まることからこの推測が裏付けられる．SCIMAにおいては，アクセスレイテンシはブロックストライド転送中のブロック（32要素）ごとにかかるので，アクセスレイテンシが40 cycleの場合のバススループットは$256/((256/4)+40) \approx 2.46$ B/cycleとなる．同様に各条件における理論最大バススループットを**表3**に示す．これより，バススループットがネックとなることでCACHE onlyではスケーラビリティが得にくく，逆にSCIMAでは大きなスケーラビリティが得られることが分かる．

なお，オフチップメモリのアクセスレイテンシが0 cycleの場合，2 CPU以上においてCACHE onlyよりもSCIMAのほうが良好な性能を示しているが，これは表2に示されているように，同じ演算でも発生するデータ転送量はSCIMAのほうが少なく，データ転送が効率的に行えているからと考えられる．これは，図6において，メモリのアクセスレイテンシが0 cycleのときのSCIMAとCACHE onlyを比較したときに，性能は両者ほぼ同等，もしくはSCIMAのほうが良い性能を出しているにもかかわらず，図7においてCACHE onlyのほうがバススループット，すなわちバスの占有量が大きいことからも裏付けられる．SCIMAは必要最小限のデータのみの転送で演算を行っていることから考えると，この点からも，CACHE onlyでは本来必要のない余分なデータ転送が多いことがうかがえる．

また，今回の評価では行列の転置を行わなかったが，転置を行った場合CACHEの性能を改善できる可能性がある．しかし，転置によるオーバヘッドが発生するうえに，転置を行ったとしてもコンフリクトを完全に解消することは難しいので，SCIMAの優位性は揺るがないと考えられる．これに関して，我々の評価条件の下で予備実験を行ったが転置による効果が得られないことを確認した．

今回の評価では，キャッシュラインサイズに対し，オフチップメモリのレイテンシが大きくなっている．通常のプロセッサでは，2次あるいは3次のキャッシュがあり，ラインサイズと，より下層のメモリへのアクセスレイテンシのバランスが我々の評価よりも良くとられている．この観点では，キャッシュを1次のみとした条件が厳しく見えるかもしれない．しかし，レイテンシが0 cycleの場合でも，なおSCIMAの性能が上回っていることにより，依然SCIMAの優位性は失われていない．いずれにしても，より深い階層のキャッシュを持つ場合については今後さらに評価を行う必要がある．

表3 理論最大バススループット
Table 3 Theoretical max bus throughput.

	Off-chip memory Access Latency	Theoretical Max Bus Throughput
SCIMA	0 [cycle]	4.000 [B/cycle]
	10 [cycle]	3.460 [B/cycle]
	40 [cycle]	2.462 [B/cycle]
CACHE only	0 [cycle]	4.000 [B/cycle]
	10 [cycle]	1.778 [B/cycle]
	40 [cycle]	0.667 [B/cycle]

6.2 NPB Kernel CG

1 CPUおよび4 CPU実行時でオフチップアクセスレイテンシが40 cycleのとき，3種類のプログラムについてキャッシュラインサイズを変化させてシミュレーションを行った結果を**図8**に全実行サイクル数を行列積の評価と同様に分類して示す．

4 CPU時に着目すると，図8から分かるように，どちらのラインサイズにおいてもSCIMAはCACHE onlyで最適化を行ったプログラムよりも良好な性能を示している．ここでラインサイズが32 Bのときに注目すると，CACHE-orgとCACHE-optを比較した場合，50%以上の性能の向上が見られる．これはキャッシュブ

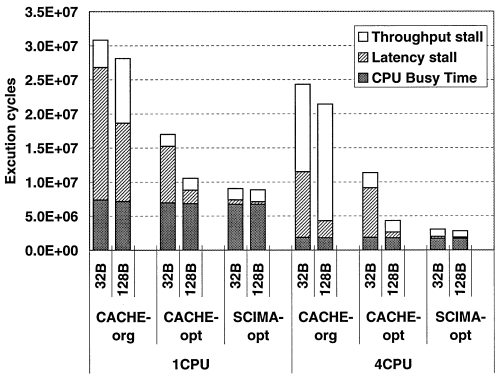

図 8 CG におけるキャッシュラインサイズ別実行サイクル数
Fig. 8 Detail of execution cycles according to cache line size on NPB Kernel CG.

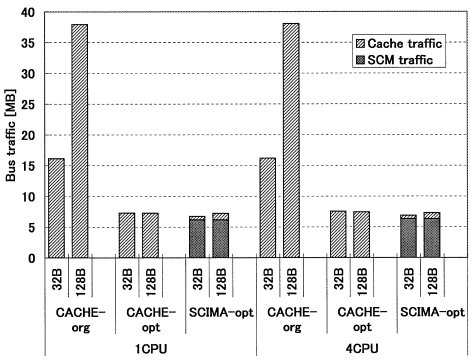

図 9 CG における総バストラフィック量
Fig. 9 Amount of bus traffic on NPB Kernel CG.

ロッキングの効果によりトラフィック量が減少したためと考えられる．次に，CACHE-opt と SCIMA-opt を比較すると，実行サイクル数が約 73%減少している．これはメモリブロッキングされた p を SCM に転送したことで配列間干渉によるトラフィックを減少させるとともに，連続アクセスとなるが再利用性のない行列 A とベクトル $colidx$ に対してキャッシュラインよりも大きな粒度でデータのロードを行い，Latency stall を軽減できたためだと考えられる．しかし，キャッシュラインが大きくなるにつれて後者の理由による優位性は少なくなる．

また，CACHE-org では，ラインサイズによって Throughput stall が変化するが，CACHE-opt ではほとんど変化しない．これは，CACHE-org ではラインサイズが大きくなるとラインコンフリクトが頻発するが，CACHE-opt では，p に対してブロッキングを行ったことにより p と他の配列とのコンフリクトが抑えられたためと考えられる．**図 9** に示された，総データトラフィック量を見ると，CACHE-opt は必要なデータしか転送しない SCIMA-opt とほとんど転送量が変わらないことからもブロッキング後の CACHE-opt では意図しないラインコンフリクトが抑えられていることが分かる．

Latency stall については，ラインサイズが大きくなると CACHE-org と CACHE-opt ともに Latency stall が小さくなっている．これは，データ転送の粒度が大きいと転送回数が減少し，結果として Latency stall が小さくなったものと考えられる．この点において，SCIMA はデータ転送粒度を大きくできるので有利となる．

なお，図 9 では，SCIMA-opt はラインサイズの増加にともなってトラフィック量が若干増加している．これは，SCM に載せなかった，ループを制御するための配列と解を格納するベクトル q 等について，8 KB・連想度 1 のキャッシュにロードされる際にキャッシュコンフリクトを発生したものと考えられる．

最後に，両アプリケーションを通じていえることは，CPU Busy Time において，CACHE only と SCIMA の差は見受けられなかったことである．これは，SCIMA において必要となる，p-load/p-store 命令の追加が，実行サイクル数に影響を与えるほどのオーバヘッドを引き起こさないことを示している．

また，今回はバスバンド幅 4 B/cycle の条件のもと評価を行ったが，今後の半導体技術動向を考えると，プロセッサと主記憶間の性能格差は拡大し，プロセッサの処理能力に対してバスバンド幅が相対的に小さくなると考えられる．SCIMA はバスバンド幅を有効に活用できることから，将来的に予想される，より厳しいバスバンド幅の条件のもとでは，キャッシュに比べ SCIMA の優位性はさらに増すと考えられる．

7. 関連研究

近年，プロセッサチップ上にキャッシュ以外のメモリを搭載することで性能向上を目指す研究が数多く行われている．Ranganathan ら[8]は従来の連想キャッシュの一部をオンチップメモリとして用いることで性能改善を図る手法を提案している．また，プロセッサチップ上に一時的に計算結果などのデータを保管するために scratch pad RAM を搭載するチップ[9),10)]もある．しかし，これらは現段階で SMP を想定した評価は行われていない．オンチップメモリを搭載した並列アーキテクチャとして OSCAR[11)]が提案されているが，これはプログラムの並列度の向上のために設計されたシステムで，オンチップメモリを活用することによりバスの有効利用を目指す SCIMA とは設計思想そ

のものが異なっている．キャッシュを用いたSMPシステムの評価については非常に多くの研究がなされている．Rothmanら[12]は，SMPマシンにおけるキャッシュミスを原因により分類し，その情報を利用してdead sharingなどを回避することでバストラフィックを削減し性能向上を図っている．

8. おわりに

本論文では，HPC向けオンチップメモリプロセッサアーキテクチャSCIMAについて，SMP構成の検討を行い，その検討した構成についてシミュレーションにより，行列積演算，およびNPB Kernel CGを用いて性能評価を行った．これらの性能評価結果より，SCIMAは大きな粒度でのデータ転送を行うことから，オフチップメモリのアクセスレイテンシによる影響を受けにくいことが分かった．また，ユーザの意図しないデータ転送を引き起こさないことによるバストラフィック量の削減が，SMP構成においても有効であることを示した．加えて，キャッシュのみを用いたSMP構成の計算機では，シングルプロセッサ構成の計算機と比較してアクセスレイテンシの増大による性能低下が大きいが，SCIMAを適用した場合，この影響を軽減できることも分かった．また，今後ますます厳しくなると考えられる，メモリに対する相対バンド幅の減少に対し，バンド幅を有効活用できるSCIMAは従来のキャッシュアーキテクチャに比べ有望と考えられる．

図8のCACHE-optの結果からも分かるように，CACHE onlyではラインサイズの増減により大きく性能が変化する．また，ラインサイズによるアプリケーションの最適化を考えた場合，移植性の低下や，多階層キャッシュへの最適化は困難であるという問題も発生する．この点，SCIMAではラインサイズという概念は存在せず，大きな粒度によるバスの使用効率の良いデータ転送が行えるという点で優位性がある．その結果，SCIMAはCACHE onlyと比較して大きなスケーラビリティが得られていると考えられる．

キャッシュを用いたアーキテクチャでも，キャッシュプリフェッチ[13]を用いることにより，これまで示してきたような事例において一定の性能改善は見込めると予想される．しかし，バス帯域が圧迫されているSMPマシンの現状ではプリフェッチによるレイテンシ隠蔽の効果は見込めないと考えられる．また，メモリインタリーブによりメモリバンド幅を向上すれば，キャッシュプリフェッチによりさらに効果的にレイテンシを隠蔽できるが，その場合でも意図しないデータ転送が発生する可能性が残るので，明示的にオフチップメモリ/SCM間のデータ転送が行えるSCIMAの有効性は変わらないと推測される．

以上の結果から，SMP構成の計算機においてSCIMAを適用することは非常に有効であると考えられる．

また，SMP化されたSCIMAでは，不必要なデータのload/storeを避けることができるため，false sharing[14]の問題の解決が見込まれる．したがって，今後はこの問題についても評価を行い，また，より正確な評価を行うために，他のベンチマークや実アプリケーションによるプログラムでの評価も行う必要があると考えている．加えて，マルチレベルキャッシュを想定した構成においても，SCIMAの優位性を確認していく必要があると考えている．

謝辞 本研究に関して，ご助言，ご討論いただきました，筑波大学計算物理学研究センターの関係者および，東京大学南谷・中村研究室の各位に感謝いたします．なお，本研究の一部は日本学術振興会未来開拓学術研究推進事業「計算科学」（Project No.JSPS-RFTF 97P01102），および科学研究費補助金（基盤（B）No.14380136）によるものである．

参考文献

1) 中村 宏，近藤正章，大河原英喜，朴 泰祐：ハイパフォーマンスコンピューティング向けアーキテクチャ*SCIMA*，情報処理学会論文誌：ハイパフォーマンスコンピューティングシステム，Vol.41, No.SIG 5 (HPS 1), pp.15–27 (2000).
2) Kondo, M., Okawara, H., Nakamura, H. and Boku, T.: SCIMA: Software Controlled Integrated Memory Architecture for High Performance Computing, *Proc. ICCD-2000*, pp.105–111 (2000).
3) 近藤正章，中村 宏，朴 泰祐：SCIMAにおける性能最適化手法の検討，情報処理学会論文誌：ハイパフォーマンスコンピューティングシステム，Vol.42, No.SIG 12 (HPS 4), pp.37–48 (2001).
4) 岩本 貢，渡辺亮介，近藤正章，中村 宏，朴 泰祐：NASPB CG，FTにおけるSCIMAの性能評価，情報処理学会研究報告，2000-HPC-83, pp.31–36 (2000).
5) Bailey, D., Harris, T., Saphir, W., van der Wijngaart, R., Woo, A. and Yarrow, M.: The NAS Parallel Benchmarks 2.0, *NASA Ames Research Center Report*, NAS-05-020 (1995).
6) Lam, M.S., Rothberg, E.E. and Wolf, M.E.: The cache performance and optimizations of Blocked Algorithms, *Proc. ASPLOS-IV*, pp.63–74 (1991).
7) Culler, D.E. and with Anoop Gupta, J.P.S.:

Parallel Computer Architecture, Morgan Kaufmann Publishers Inc., pp.293–299 (1999).
8) Ranganathan, P., Adve, S. and Jouppi, N.P.: Reconfigurable Caches and their Application to Media Processing, *Proc. ISCA-27*, pp.214–224 (2000).
9) Diefendorff, K.: Sony's Emotionally Charged Chip, *Microprocessor Report*, Vol.13, No.5 (1999).
10) Turley, J.: StrongArm Speed to Triple, *Microprocessor Report*, Vol.32, No.6 (1999).
11) Kasahara, H., Okamoto, M., Yoshida, A., Ogata, W., Kimura, K., Matsui, G., Matsuzaki, H. and Honda, H.: OSCAR Multi-grain Architecture and Its Evaluation, *Proc. IWIA-'97*, pp.106–115 (1997).
12) Rothman, J.B. and Smith, A.J.: Analysis of Shared Memory Misses and Reference Patterns, *Proc. ICCD-2000*, pp.187–198 (2000).
13) Chen, T.-F. and Baer, J.-L.: A Performance Study of Software and Hardware Data Prefetching Schemes, *Proc. ISCA-21*, pp.223–232 (1994).
14) Jeremiassen, T.E. and Eggers, S.J.: Reducing false sharing on shared memory multiprocessors through compile time data transformations, *SIGPLAN Notices*, Vol.30, Issue 8, pp.179–188 (1990).

(平成14年9月24日受付)
(平成15年1月 1 日採録)

髙橋 睦史（学生会員）

昭和55年生．平成14年筑波大学第三学群情報学類卒業．現在，同大学大学院システム情報工学研究科在学中．ハイパフォーマンスコンピューティング向けプロセッサに関する研究に従事．

近藤 正章（学生会員）

昭和50年生．平成10年筑波大学第三学群情報学類卒業．平成12年同大学大学院工学研究科博士前期課程修了．現在，東京大学大学院工学系研究科博士後期課程在学中．計算機アーキテクチャ，ハイパフォーマンスコンピューティングの研究に従事．

朴 泰祐（正会員）

昭和59年慶應義塾大学工学部電気工学科卒業．平成2年同大学大学院理工学研究科電気工学専攻後期博士課程修了．工学博士．昭和63年慶應義塾大学理工学部物理学科助手．平成4年筑波大学電子・情報工学系講師，平成7年同助教授，現在に至る．超並列処理ネットワーク，超並列計算機アーキテクチャ，ハイパフォーマンスコンピューティング，並列処理システム性能評価の研究に従事．電子情報通信学会，日本応用数理学会，IEEE各会員．

高橋 大介（正会員）

昭和45年生．平成3年呉工業高等専門学校電気工学科卒業．平成5年豊橋技術科学大学工学部情報工学課程卒業．平成7年同大学大学院工学研究科情報工学専攻修士課程修了．平成9年東京大学大学院理学系研究科情報科学専攻博士課程中退．同年同大学大型計算機センター助手．平成11年同大学情報基盤センター助手．平成12年埼玉大学大学院理工学研究科助手．平成13年筑波大学電子・情報工学系講師．博士（理学）．並列数値計算アルゴリズムに関する研究に従事．平成10年度情報処理学会山下記念研究賞，平成10年度情報処理学会論文賞各受賞．日本応用数理学会，ACM，IEEE，SIAM各会員．

中村 宏（正会員）

昭和60年東京大学工学部電子工学科卒業．平成2年同大学大学院工学系研究科電気工学専攻博士課程修了．工学博士．同年筑波大学電子・情報工学系助手．同講師，同助教授を経て，平成8年より東京大学先端科学技術研究センター助教授．計算機アーキテクチャ，ハイパフォーマンスコンピューティング，計算機の上位レベル設計支援，非同期式計算システムの研究に従事．本会平成5年度論文賞，平成6年度山下記念研究賞各受賞．電子情報通信学会，IEEE，ACM各会員．

佐藤　三久（正会員）

昭和 34 年生．昭和 57 年東京大学理学部情報科学科卒業．昭和 61 年同大学大学院理学系研究科博士課程中退．同年新技術事業団後藤磁束量子情報プロジェクトに参加．平成 3 年，通産省電子技術総合研究所入所．平成 8 年より，新情報処理開発機構つくば研究センタに出向．同機構並列分散システムパフォーマンスつくば研究室室長．平成 13 年より，筑波大学電子・情報工学系教授．理学博士．並列処理アーキテクチャ，言語およびコンパイラ，計算機性能評価技術等の研究に従事．日本応用数理学会会員．