Vol. 41 No. SIG 5(HPS 1) 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 Aug. 2000

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*

中村 宏[†] 近藤正章[††,☆]
大河原英喜[†,☆☆] 朴 泰祐[††]

近年のプロセッサは，クロック周波数の向上，命令レベル並列性の活用などにより高性能化が図られているが，一方で主記憶の性能はプロセッサほど改善されてはいない．特に，ハイパフォーマンスコンピューティングにおいては，このプロセッサと主記憶との性能格差の問題が深刻である．そこで，この問題の解決を目指した新しいプロセッサアーキテクチャSCIMAを提案する．提案するアーキテクチャは，プロセッサチップ上に主記憶の一部を実装するものである．本論文では，そのアーキテクチャの基本構成，およびシミュレータを用いた性能評価を示す．評価結果より，SCIMAは従来のキャッシュ型のアーキテクチャに比べ非常に高い性能を達成することが分かった．

SCIMA: A New Architecture for High Performance Computing

Hiroshi Nakamura,[†] Masaaki Kondo,[††,☆] Hideki Okawara[†,☆☆] and Taisuke Boku[††]

Processor performance has been improved due to clock acceleration and ILP extraction techniques. Performance of main memory, however, has not been improved so much. The performance gap between processor and memory will be growing in the future. This is very serious problem in high performance computing because effective performance is limited by memory in most cases. In order to overcome this problem, we propose a new VLSI architecture called SCIMA, which integrates processor and a part of main memory into a single chip. This paper presents its architecture and performance evaluation. The evaluation results reveal the superiority of SCIMA compared with conventional cache-based architecture.

1. はじめに

近年のプロセッサはトランジスタ集積度の進歩を背景に，クロック周波数の向上，スーパスカラやVLIWに代表される命令レベル並列性の活用による性能向上が著しい．一方，主記憶は，容量面においては飛躍的に進歩しているものの，速度面においてはプロセッサほどの性能向上を示していない．したがって，プロセッサとメモリの性能格差の問題が深刻化している．

この問題に対処するために，従来からキャッシュメモリが用いられてきた．しかし，データセットの大きな大規模科学技術計算などのハイパフォーマンスコンピューティング（HPC）分野のアプリケーションでは，キャッシュが有効に機能しないことが多い[1]．キャッシュ容量に比べデータセットが非常に大きく，またデータの時間的局所性がほとんどないため，キャッシュミスによる主記憶へのアクセスが頻発し，性能が大きく低下する．

この性能低下は，スループット不足とレイテンシ増大に起因するが，スループットが不足している場合はレイテンシ隠蔽手法は有効ではない．キャッシュミス時のペナルティ低減方法として，キャッシュプリフェッチ[2]が広く使われているが，メモリスループットが不足している場合は，性能はそのメモリスループットで抑えられてしまい，プリフェッチが有効に機能しないことが報告されている[3]．したがって，メモリシステム全体としてのスループットを確保し，そのうえで適切なレイテンシ隠蔽手法を用いるべきである．

今後の半導体実装技術を考えると，オンチップ記憶のスループットとオフチップ記憶のスループットの性能格差はさらに広がると考えられるため，集積度の

† 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
†† 筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba
☆ 現在，東京大学先端科学技術研究センター
Presently with Research Center for Advanced Science and Technology, The University of Tokyo
☆☆ 現在，株式会社富士通研究所
Presently with FUJITSU LABORATORIES LTD.

向上を頼りにプロセッサチップ上の記憶を増やし，オフチップへのアクセスを低減することが必須である．キャッシュブロッキング（タイリング）[4]は，ソフトウェア的な手法でデータの再利用性を向上させることで，データへのアクセスをキャッシュという上位階層の記憶に閉じるさせる手法であり，プロセッサチップ上に実装可能なキャッシュ容量がこれからも増大するという技術動向を考えると今後も有望な手法である．

しかし，従来のキャッシュでは，データのアロケーションとリプレースメントがハードウェアで制御されるため，どのデータをキャッシュに載せるか指定できない（配列 A は載せたいが B は載せたくないなど），ラインコンフリクトによる同一配列のブロック内データの干渉（self-interference）や異なる配列間のデータの干渉（cross-interference）を防げない，などの問題がある．self-interference の問題に対しては，適切なブロックサイズを選ぶアルゴリズム[5]や，配列サイズを変更しコンフリクトを防ぐ手法[6]などが提案されているが，いずれもキャッシュ構成に応じてプログラムを変更する必要があり，また cross-interference の問題は依然残る．

これらの問題は，データのアロケーションやリプレースメントがハードウェアによって暗黙的に制御されるキャッシュをソフトウェアから制御しようとするために生じる．定型的なアクセスが多い大規模科学技術計算では，データのアロケーションをハードウェアではなくむしろプログラムによりソフトウェア的に制御することが可能で，その方が性能面で有利と考えられる．そこで，チップ上に実装するメモリとして，従来のキャッシュに加えて，アドレス指定可能な主記憶の一部を SRAM として載せる（以降では，そのメモリをオンチップメモリと呼ぶ）新しいアーキテクチャ，SCIMA（Software Controlled Integrated Memory Architecture）を提案する．

これまで，SCIMA に関して初期的性能評価を行ってきたが[8]，本論文ではクロックレベルシミュレータを用いた詳細な性能評価を行い，SCIMA の有効性を示すことを目的とする．

関連研究

プロセッサチップにオンチップメモリとして DRAM を混載する研究は多い．IRAM[9]は，プロセッサに DRAM を搭載し，内部メモリの高バンド幅を活用し，さらにロジック部にはベクトルプロセッサを配置して高性能化を狙うものである．また，PPRAM[10]も同様にプロセッサ内部に DRAM を搭載し，チップ内のメモリの高バンド幅，低レイテンシを活かし高性能化を図るものである．しかし，これらの研究では必要なデータのほとんどがオンチップメモリに収まることを想定しており，チップ外からのデータ供給に関しては検討していない．SCIMA では，HPC の大きいデータセットは DRAM を用いてもプロセッサチップ上に載せることは難しいと考え，容量を犠牲にしてもアクセス速度を重視し，SRAM を載せることを考えている．

ソフトウェア制御可能なオンチップメモリを利用し高速化を狙う研究として，Compiler-Controlled Memory[7]がある．これは，コンパイル時に発生する spill code で spill されるデータを載せるための，コンパイラによって制御可能な小容量メモリであるが，適用対象が spill code に限定されている．データセットが小さい場合に，小容量の scratch pad RAM を用いて特定のアプリケーションに最適なリプレースメント制御を行うプロセッサ[12),13)]もある．しかし，いずれもデータセットが限られた特定のアプリケーションを対象とするものである．

SCIMA は，データセットの大きい HPC に適したメモリアーキテクチャとして SRAM オンチップメモリを採用し，アプリケーションに合わせて柔軟にユーザがオフチップとの間でリプレースメント制御を行える点で，他の研究とは異なる．

2. SCIMA

2.1 概　　要

提案するアーキクテチャ，SCIMA の構成を図 1 に示す．SCIMA ではプロセッサ上にキャッシュだけでなく，アドレス指定可能なオンチップメモリも搭載する．従来のキャッシュではデータのアロケーションやリプレースメントは，ハードウェアにより暗黙的に制御されるのに対し，オンチップメモリではそれらの制御をソフトウェアで明示的に行える．これがキャッシュとオンチップメモリの本質的な違いである．ここで，オ

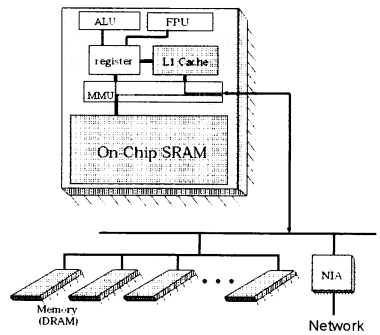

図 1 SCIMA の構成図
Fig. 1 Structure of SCIMA.

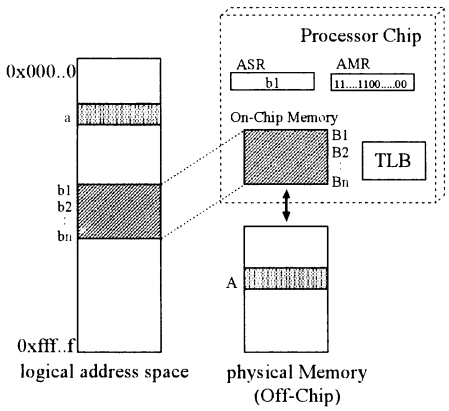

図 2 アドレス空間の概要
Fig. 2 Organization of address space.

ンチップメモリに加えキャッシュも搭載しているのは，必ずしもすべてのデータアクセスをソフトウェアが解析できない場合を考慮しているためであり，その場合は，キャッシュを用いる．

2.2 アドレス空間

図 1 に示したように，SCIMA ではオンチップメモリとオフチップメモリという性質の異なるメモリを扱う．そのためプログラム上で両者を明示的に区別する必要がある．そこで SCIMA では論理アドレス空間上にオンチップメモリ領域をマッピングする．しかし，オンチップメモリ領域は連続する大きなブロック領域であるため，従来の TLB で管理するとページ数が増大し，高速なオンチップメモリ領域へのアクセスで TLB ミスの頻発による性能低下を招く．そこで，オンチップメモリ領域は 1 つの大きいページのように扱うこととし，**図 2** に示すように，通常の TLB とは別に以下の 2 つの特殊レジスタで管理することにする．

On-Chip address mask register（AMR） オンチップメモリの容量を示すためのマスクレジスタ：AMR の下位 m ビットが 0 であれば，オンチップメモリ容量が 2^m B であることを表す．

On-Chip address start register（ASR） オンチップメモリ領域をマップする領域の先頭論理アドレスを保持するレジスタ：先頭アドレスは，オンチップメモリ領域サイズのアラインメントに一致するものとする．したがって，AMR の下位 m ビットが 0 であれば，ASR の下位 m ビットも 0 となる．

メモリアクセスがあった場合，アクセス先アドレスと AMR の論理積をとり，その値が ASR と等しければ，該当アドレスはオンチップメモリ領域であり，下位 m ビットは，オンチップメモリ内オフセットとなる．そうでなければ，該当アドレスはオフチップメモリ領域となる．

オンチップメモリとキャッシュの間の包含関係

オフチップメモリ領域は，通常のページ単位の管理を行い，ページ単位で cacheable/uncacheable の属性を設けるものとする．これは従来のプロセッサでも広く実装されている機構であり，本アーキテクチャでもその機構を用いる．しかし，オンチップメモリ領域はつねに uncacheable とする．したがって，キャッシュとオンチップメモリとの間には包含関係はない．

メモリ階層間のデータ転送の種類

SCIMA ではアドレスによって，オンチップメモリ領域とオフチップメモリ領域が分離されており，さらに後者に対しては，cachable 属性のページに対してはキャッシュを通したオフチップメモリアクセス，uncachable 属性のページに対してはキャッシュを通さないダイレクトなオフチップメモリアクセスが行われる．したがって，SCIMA では以下の戦略でデータアクセスを行うことが可能である．

- アクセスが定型的なデータに対しては，オンチップメモリを利用すべく
  register ↔ On-Chip Memory ↔ Off-Chip Memory
- アクセスが定型的ではないが再利用性があるデータに対しては，キャッシュを利用すべく
  register ↔ cache ↔ Off-Chip Memory
- アクセスが定型的でなく再利用性にも乏しいデータに対しては，キャッシュ上の干渉を防ぐべく，uncacheable 属性を用いて
  register ↔ Off-Chip Memory

2.3 拡張命令

新しくオンチップメモリを導入したことにともない，以下の命令の追加・拡張を行う．具体的にはオフチップ・オンチップメモリ間のデータ転送命令とオンチップメモリ・レジスタ間のデータ転送命令である．

2.3.1 page-load, page-store

オフチップメモリとオンチップメモリ間でデータ転送を行う命令として，page-load，page-store 命令を追加する．この転送は page☆という大きな粒度で行うことを基本とし，DMA 転送により実現される．

また，転送サイズは可変とし，ブロックストライド転送の機能を付け加える．この機能により，オフチッ

☆ ここでの "page" は，仮想記憶における論理/物理アドレス変換の管理単位としての "ページ" とは異なり，オンチップメモリの管理単位としてのブロックを指す．以降 "page" の表記はこの意味で用いる．

プメモリ上の不連続領域のデータをパッキングしてオンチップメモリに持ってくることが可能となる．これらのデータを再利用する場合，オンチップメモリ上では連続するため処理効率が向上する．HPCアプリケーションでは，多次元配列要素に対するアクセスなどが多いために有用な機能である．

2.3.2 load-multiple/store-multiple

オンチップメモリとレジスタの間にビット幅の広いデータパスを設けることは将来的に難しくないと考えられる．そこで，従来のload/store命令に加え，1命令で複数の連続ワードを複数のレジスタへ/からオンチップメモリから/へ一度に転送する命令として，load-multiple/store-multiple命令を導入する．この命令は1命令で多くのレジスタを扱うため，論理レジスタ数を従来の32本から拡張する必要がある．なお，本論文では以降load/store命令とload/store-multiple命令は区別せず，load/store命令として統一的に扱う．

2.4 オンチップメモリとキャッシュの統合

プロセッサチップ上に実装可能な記憶容量は，半導体集積技術やチップ上に実装するプロセッサコア部のトランジスタ数などに依存する．しかし，チップ上の記憶容量がこれらの要因で決定されたときに，オンチップメモリとキャッシュに割り振るべき容量はアプリケーションの性質に応じて変わる．そこで，ここでは，キャッシュとオンチップメモリでハードウェアを共用し，総容量は一定だがオンチップメモリとキャッシュの容量の割合にある程度の自由度を持たせる構成について述べる．

2.4.1 ハードウェア構成

2.2節で述べたASR，AMRのほかに，下記の特殊レジスタをハードウェア的に用意し，キャッシュとオンチップメモリを統合して利用する．

Way Lock Register（WLR） キャッシュの連想度（Way数）と同ビット数のWay Lock Registerを用意する．各Wayに対応するビットがセットされている場合，そのWayがオンチップメモリとしてロックされていることを表す．

On-chip Memory Valid（OMV） オンチップメモリとしての利用の有無を示す1ビット．

ハードウェア構成を**図3**に示す．図3では，Way数が4の32 KBキャッシュを，16 KBキャッシュと16 KBオンチップメモリとして用いる場合を例示している．

オンチップメモリ領域の割当ては，用意されたシステムコールを介して行い，Wayロック時には該当Wayの内容はフラッシュされ，ロックされるWayに対応する WLRのビットとOMVをセットし，AMRとASRに必要な値をセットする．AMRのWAYビット部によってロックされるWay数が，ASRのWAYビット部によってどのWayからロックされるかが決まる．ここで，WAYビット部のビット数は $\log_2(Way数)$ となる．図3の例では，AMRのWAYビット部が10なのでWay2つがオンチップメモリとして用いられ，ASRのWAYビットが00となのでWay0から2つのWayがロックされる．したがって，そのときのWLRは0011となる．オンチップメモリとして用いるサイズによって，**表1**の組合せ☆がある．

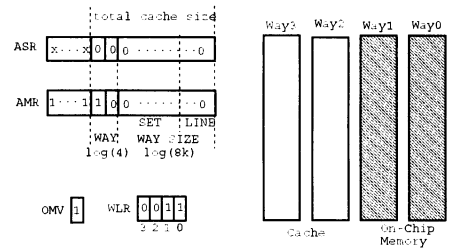

図3 キャッシュとオンチップメモリの統合
Fig. 3 Organization of the unified cache and on-chip memory.

表1 オンチップメモリ領域の割当て
Table 1 Configuration of the cache and on-chip memory.

オンチップメモリ容量	ASR WAY部	AMR WAY部	オンチップメモリとなるWay	WLR	OMV
32 KB	00	00	way0, 1, 2, 3	1111	1
16 KB	00	10	way0, 1	0011	1
	10	10	way2, 3	1100	1
8 KB	00	11	way0	0001	1
	01	11	way1	0010	1
	10	11	way2	0100	1
	11	11	way3	1000	1
0 KB	—	—	なし	0000	0

2.4.2 アクセス動作

メモリアクセス時には，該当アドレスをAMRでマスクし，ASRと比較することでオンチップメモリ領域か否かが分かる．オンチップメモリへのアクセスの場合，該当アドレスのWAYビット部がアクセスすべきWAYを表している．たとえば図3の場合，アドレスのWAYビット部が00ならWay0，01ならWay1をアクセスすればよい．重要なのは，各Wayにおけるどのセットをアクセスするかは，オンチップメモリ領域/オフチップメモリ領域によらず，アドレスの下位ビット（図3のSETビット）から一意に定まる点

☆ OMVが1のときは，ASR WAY部とAMR WAY部よりWLRは決定し，OMVが0のときはWLRの全ビットは0となるので，厳密にはWLRは冗長な情報である．

である．したがって，アクセス動作は，

(1) オンチップメモリ領域か否かの判定（ASR, AMR, OMV を用いる）と，アドレスの SET ビットに対応する全 Way のセットへのアクセスを同時に行う．

(2) オンチップメモリと判定されれば，WAY ビットに決定される Way の対応セットを用いて処理を行う．そうでなければ，(1) で参照したタグのチェックを WLR が 0 となっている Way に対して行い，通常のキャッシュアクセス動作を行う．

となり，通常のキャッシュアクセス動作に比べてクリティカルパスはほとんど延びない．

このように，キャッシュの一部を特別の用途にロックする機構は SH-4[14] で採用されているが，SH-4 では DSP モード下の prefetch 命令だけが扱える "キャッシュ" としてロックするのであって，リプレースメント/アロケーションの制御は依然ハードウェアで自動的に行われる．それに対し，SCIMA ではリプレースメント/アロケーションの制御をソフトウェアに解放したメモリ空間として提供するものであり，その点で本質的に異なる．

2.5 メモリアクセス順序保証

2.3 節で述べたように，オンチップメモリは，load/store 命令によるレジスタとのデータ転送，および page-load/store 命令によるオフチップメモリとの間のデータ転送の対象となる．したがって，同じアドレスに対するこれらの複数の命令がある場合，それらはプログラム上での命令順と同じ順で実行される必要がある．問題となる場合は以下のように分類できる．

[A]（オンチップメモリ領域への）load/store → （オンチップメモリ領域への）load/store

[B] page-load/store → page-load/store

[C] page-load/store → （オンチップメモリ領域への）load/store

[D]（オンチップメモリ領域への）load/store → page-load/store

ここで，[A] については従来のプロセッサが複数のキャッシュアクセスの順序を保証するのと同様の機構で，また [B] についてはオフチップへの page-load/store 命令を必ず命令順に実行することで回避できる．一方，[C]，[D] については，オンチップメモリを page（page-load/store 命令の転送単位である page）単位に分割し，**図 4** に示すハードウェアを用いて以下の管理を行う．

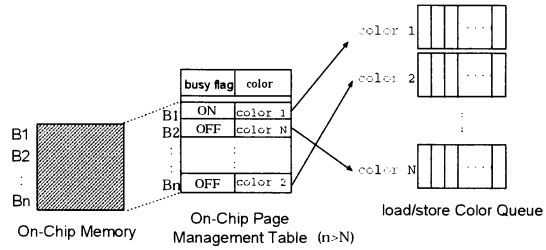

図 4 順序保証に必要なハードウェア
Fig. 4 Hardware for correct access order.

[C] の場合の順序保証

On-Chip Page Management Table（以下OCPMT）と呼ぶ管理ハードウェアを設ける．page-load/store 命令が発行されると該当 page の busy flag を ON にし，page-load/store が終了した時点で OFF にする．オンチップメモリへの load/store 命令が発行されるとアクセス該当 page に対する OCPMT の busy flag をチェックし，busy flag が ON であった場合は OFF になるまで待つ．

[D] の場合の順序保証

オンチップ領域への load/store 命令が発行されると該当 page のカラーに対応するキューに入り，終了するとキューから削除される．page-load/store も同様に対応するキューに入り，それらがキューの先頭に来たとき，すなわち同一 page に対する先行する load/store が終了した時点で発行される．

OCPMT にカラー番号を示すタグを設け，アクティブな page に対するダイナミックなカラー割当てをハードウェアで管理し，各カラーごとにキュー（図 4 の *load/store Color Queue*）を割り当てる．割り当てるカラーがなくなった場合は，フリーなキューができてカラーが回収されるまで命令発行をストールさせる．ページごとに静的にカラーを設けるのではなく動的にカラーを割り当てるのは，キューの数を減らしハードウェア量を削減するためである．

この順序保証を行う場合の load/store，page-load/store 命令の処理は，文献 15) により詳しく述べてある．

2.6 cacheable/uncacheable 属性およびコヒーレンスについて

page-load/store 命令において，対象となるオフチップメモリが uncacheable であれば該当オフチップメモリとキャッシュとの間のコヒーレンスは考える必要がない．しかし cacheable の場合は，たとえば実行前に対応するキャッシュの内容をフラッシュするなど，何らかのコヒーレンス制御が必要となる．性能向上のためには page-load/store 命令をできるだけ高速に処理

する必要があるため，当該オフチップメモリ領域をつねに uncacheable にすることが望ましい．

3. 性能評価

本論文では，カーネルとして行列積，実アプリケーションとして，筑波大学の計算物理学研究センターで行われている QCD（量子色力学）計算[16] の 2 つを取り上げて評価を行う．行列演算は HPC 分野では多く用いられる処理であり，その中でも行列積はキャッシュを用いた最適化が多く提案されている．そこで行列積を取り上げて SCIMA の有効性を示すことは重要である．また，HPC の実アプリケーションでの有効性を示すことも，SCIMA の目的から重要である．本章では，性能評価環境，および，各ベンチマークの最適化に関して述べる．

3.1 性能評価環境

SCIMA では 2 章で述べたアーキテクチャ拡張を行うが，具体的には MIPS IV アーキテクチャを拡張したものとして評価を行う．

本来はアーキテクチャ拡張に対応した最適化コンパイラを開発するのが理想的である．しかし，オンチップメモリへ配置するべき配列の宣言，page-load/store で転送すべきデータとそのタイミング，および cacheable/uncacheable の指定は，データアクセスが定型的な HPC アプリケーションではユーザによる最適化が比較的容易かつ的確だと思われるので，ユーザがソースレベルで特別な予約関数を用いて行うこととした．

評価は，既存のコンパイラが生成するバイナリを入力とするクロックレベルシミュレータを開発して行った．上記の予約関数に関しては，既存のコンパイラが生成するアセンブリコードに対して，MIPS IV ISA では使われない命令（具体的には co-processor 命令）を用いて必要な情報を挿入するプリプロセッサを作成し，シミュレータ側でその追加情報を解釈実行することで対処する．

開発したシミュレータは，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が，ループ構造の多い HPC アプリケーションの評価においては妥当と思われる．データキャッシュについては，今のところ lock-up free な一次キャッシュのみをサポートしている．入力となるバイナリを既存のコンパイラで作成するため，演算器やロードストアユニットを増やした場合の命令スケジューリングの質が良くないことが予想される．そこで，リザベーション・ステーションを用いた out-of-order 実行機構も入れてある．なお，本論文はキャッシュとオンチップメモリの比較を主眼としているので，同じ質のバイナリコードを用いることで十分公平な比較ができると考えている．

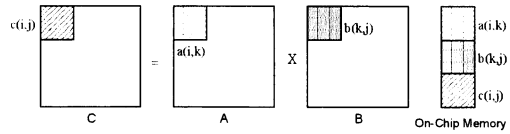

図 5 SCIMA 用コードにおけるブロッキング
Fig. 5 Blocking on SCIMA code.

シミュレータの構成を含め，この性能評価環境の詳細は文献 17) に述べてある．

3.2 行列積

C = A × B の行列積においてすべての配列は N × N の正方行列とし，配列の大きさは評価において変化させた．行列積には種々の最適化が提案されているが，本評価においては以下に述べる最適化手法を行った．

3.2.1 キャッシュ/オンチップメモリ–ブロッキング

評価では，キャッシュ用のコードと SCIMA 用のコードの 2 種類のプログラムを作成し，各々，キャッシュ向け最適化とオンチップメモリ向け最適化を行った．キャッシュ用コードでは一般的なキャッシュブロッキングを用いる．すなわち，キャッシュを A，B，C の行列用に 3 等分して考え，それぞれの行列の再利用性の高いブロッキングされた小行列がキャッシュに載るようにする．また，SCIMA 用コードにおいては，**図 5** に示すように page-load/store を用いてデータのリプレースメントをユーザ制御で行う．

3.2.2 レジスタブロッキング

三重ループ（ijk）に対して外部ループ（ij）をそれぞれ r 回アンローリングしてレジスタブロッキングを行う．最内ループで必要となるレジスタ数を考え，$r^2 + 2r <$ レジスタ数となるようにアンローリングを行う．

3.3 QCD 計算

QCD（量子色力学）は，素粒子の強い相互作用を記述する場の理論である．今回とりあげた QCD 計算では，ハドロンの基本構成粒子であるクォークと，それを結び付けるグルオン場を 4 次元格子空間上でシミュレーションする．各格子点にはクォークの自由度を表す 3 × 4 の複素行列 G が，隣接する各格子点を結ぶリンクにはグルオンの自由度を表す 3 × 3 の複素ユニタリ行列 U が用意される．この計算では，既知の行列 $D(U)$ と H を用い，$H = D(U)G$ という線形方程式を解いてグルオン伝搬関数 G を求める時間が

表 2 イテレーションループの構造
Table 2 Structure of the iteration loop.

処理ルーチン	計算に使う配列 → 結果を store する配列
1 continue	
MULT 間処理 1	
call RBMULT	B_e (0.5 MB), U (1.5 MB) →G_o (0.25 MB)
call localmult	G_o (0.25 MB), M_o (1.5 MB)→G_o (0.25 MB)
call RBMULT	G_o (0.5 MB), U (1.5 MB) →V_e (0.25 MB)
call localmult	V_e (0.25 MB), M_e (1.5 MB) →V_e (0.25 MB)
MULT 間処理 2	
call RBMULT	R_e (0.5 MB), U (1.5 MB) →G_o (0.25 MB)
call localmult	G_o (0.25 MB), M_o (1.5 MB)→G_o (0.25 MB)
call RBMULT	G_o (0.5 MB), U (1.5 MB) →T_e (0.25 MB)
call localmult	T_e (0.25 MB), M_e (1.5 MB) →T_e (0.25 MB)
MULT 間処理 3	
goto 1	

ほとんどを占める．具体的には BiCGStab 法を用いた反復法で解いており，反復法の過程で繰り返されるイテレーションループが，計算時間の大半を占める．そこで，このイテレーションループに注目して評価を行った．

評価においては，イテレーションループの構造，アクセスする配列のサイズ，再利用性を考慮する必要があるが，データサイズとして，1 プロセッサあたり $6^3 * 12$ の 4 次元格子空間を想定した☆．

3.3.1 データアクセスの特徴

イテレーションループの構造を**表 2** に示す．この表から分かるように，計算に用いられる各配列は，再利用性とアクセスサイズに関して以下の性質を持つ．

G, R, B, V, T ルーチン間での再利用性が高い．また RBMULT ルーチン内においても最大 8 回アクセスされ，ルーチン内再利用性も高い．

U イテレーションあたり 4 回呼ばれる RBMULT ルーチンのみでアクセスされ，ルーチン内では 1 回しかアクセスされない．値は更新されない．サイズは 1.5 MB である．

M localmult ルーチンのみでアクセスされ，ルーチン内では 1 回しかアクセスされない．値は更新されない．イテレーションあたり odd 部（M_o）を用いた localmult ルーチンが 2 回，even 部（M_e）を用いた localmult ルーチンが 2 回，計 4 回 localmult ルーチンが呼び出される．M_o と M_e のサイズは各々1.5 MB である．

3.3.2 オンチップメモリを用いた最適化手法

表 2 より分かるように，再利用性の高い G，R，B，V，T に関しては LRU 置き換えアルゴリズムが最適であるのでこれらの配列はキャッシュに載せる．一方，U，M は再利用性に乏しいが，反復法で繰り返し実行されるイテレーションループをまたがった再利用性はあるので，可能であれば，U，M に関しても極力再利用を行うことが望ましい．重要なのは，U，M の再利用性のタイムスパン（一度使われてから次に使われるまでの時間間隔）が G，R，B，V，T に比べて大変長いことである．したがって，U，M もキャッシュに載せることにすると，再利用性のタイムスパンが短くすぐに使われる G，R，B，V，T を追い出す可能性がある．そこで，U，M はキャッシュではなくオンチップメモリに載せることにする．具体的には，最内ループで用いる U，M のサイズが 64 KB であることに着目し，オンチップメモリ上に 64 KB のテンポラリバッファを確保する．そして，残りのオンチップメモリには，M より再利用性が高い U を容量的に許されるだけ固定して載せ，載らない U，M については，必要となるたびにこのバッファに対してそのつど page-load する．なお，4.2 節の評価で，オンチップメモリがまったくない場合には全データをキャッシュ上で扱う．

4. 評価結果

本章では 3.2 節と 3.3 節で述べた行列積計算と QCD 計算の評価結果を述べる．

4.1 行列積

4.1.1 評価条件

SCIMA と従来のキャッシュ型アーキテクチャとを比較するため，3.2.1 項で述べたように，キャッシュ用コードと SCIMA 用コードを作成しそれぞれ下記の仮定をした．

- キャッシュ用コードに対するキャッシュは容量 32 KB，連想度 2，ラインサイズ 32 B：オンチップメモリはなし．
- SCIMA 用コードに対するキャッシュは 128 B☆☆，連想度 2，ラインサイズ 32 B：オンチップメモリは 32 KB，page サイズ 4 KB，カラー数は 8.

また，以下の共通の仮定をおいた．

- レジスタ数：整数，浮動小数点ともに 32 個．レジスタ拡張をしないので 2.3.2 項で述べた SCIMA 用の multiple 命令は用いない．
- 同時発行命令数：整数演算 2，浮動小数点演算（積和型）1，ロードストア 1，すなわちキャッシュ・

☆ 現在，ピーク性能 614.8 GFLOPS の並列計算機 CP-PACS を用いて $24^3 \times 48$ の 4 次元格子空間をシミュレーションしている．物理学的な見地から $48^3 \times 96$ の空間のシミュレーションが必要とされており，そのためにはピーク 64 TFLOPS 程度の計算能力が必要と推測されている．この性能を 16 GFLOPS/PU × 4096 PU で達成することを想定し，1 プロセッサあたりのデータは $6^3 \times 12$ とした．

☆☆ 整数データのみを載せる．全メモリアクセス中整数データへのアクセスはわずか 1%程度であった．

オンチップメモリのスループットは 8 B/cycle.

- 浮動小数点積和演算命令レイテンシ：4 cycle.
- load/store 命令レイテンシ：キャッシュ・オンチップメモリともに 1 cycle.
- オフチップメモリスループット：8 B/cycle.

浮動小数点積和演算器が 1 なので理論的ピーク性能は 2 FLOPC（FLoating-Operations Per Cycle）である．浮動小数点演算器数は 4.1.4 項で変化させる．また，キャッシュ/オンチップメモリスループットとオフチップメモリスループットは同じであるが，この比率は 4.1.5 項で変化させる．キャッシュ/オンチップメモリサイズがともに 32 KB なので，キャッシュ用コード，SCIMA 用コード，ともにブロックサイズ 36×36 でブロッキングした．

このような条件のもとで，オフチップメモリレイテンシを 40，10，0 cycle の 3 通りに変化させた．40 cycle は実際のオフチップメモリレイテンシを，10 cycle は L2 キャッシュのレイテンシを仮定した．0 cycle というレイテンシはありえないが評価用に仮想的に採用した．

4.1.2 キャッシュ用コードの検証

まずキャッシュ用コードがどの程度最適化されているかを検証するため，SGI Origin200（180 MHz，ピーク 360 MFLOPS）で実行したときの性能を**図 6** に示す．4.1.1 項で述べた条件とこの計算機は，キャッシュの構成，レジスタ数，演算器数などが同じである．比較のため，最適な行列積ルーチンを生成する ATLAS（Automatically Tuned Linear Algebra Software）[18] をこの計算機上で用いた場合の性能も示す．ATLAS と同程度の性能を達成していることから，キャッシュ評価用コードはかなり最適化されているといえる．若干 ATLAS の方が性能が良い理由は，この計算機が搭載している 2 次キャッシュに関する最適化をキャッシュ用コードでは考慮していないのに対し ATLAS では考慮しているためと思われる．

4.1.3 キャッシュと SCIMA の性能比較

図 7 に行列サイズを変化させたときの SCIMA 用コード，およびキャッシュ用コードの性能を示す．性能を FLOPC（FLoating-Operations Per Cycle）で表す．

図 7 から，オフチップメモリレイテンシを増加した場合，SCIMA では高い性能を維持する一方で，キャッシュでは大きく性能が低下することが分かる．

これは，SCIMA ではオフチップメモリからのデータを 4 KB 単位で転送するため，レイテンシの影響をあまり受けず，高い実効スループットを維持できるのに対し，キャッシュでは 32 B という小さいサイズのライン転送になるため，レイテンシの影響が大きく実効スループットが大きく低下するためと考えられる．たとえば，レイテンシが 0 cycle の場合は，キャッシュと SCIMA はほぼ同じ性能であるのに対し，レイテンシが 0 cycle から 40 cycle になると，SCIMA では 4〜5%程度しか低下しないのに対し，キャッシュでは 45%近く低下する．

オフチップメモリトラフィック量の違いも，SCIMA とキャッシュの性能格差の大きな要因である．**表 3** にキャッシュ，SCIMA のそれぞれについてのオフチップメモリトラフィック，また参考までにキャッシュにおけるキャッシュミス率を示す．表 3 から分かるように，キャッシュでは SCIMA に比べてオフチップメモ

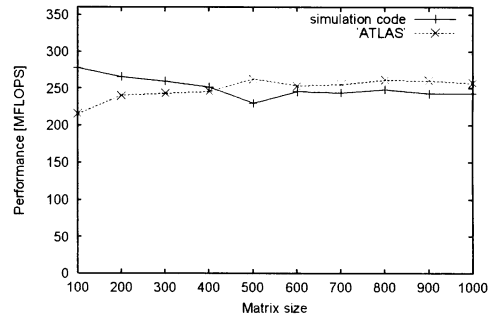

図 6 SGI Origin 上でのキャッシュ評価用コードの性能
Fig. 6 Performance of the cache code on SGI Origin.

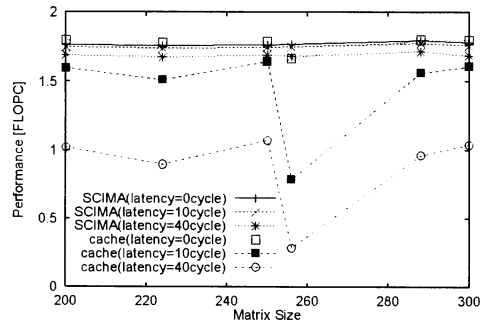

図 7 SCIMA とキャッシュの性能
Fig. 7 Performance comparison of SCIMA and cache.

表 3 行列積計算におけるオフチップメモリトラフィック
Table 3 Off-chip memory traffic on the matrix multiply.

	キャッシュ	SCIMA
Matrix size	traffic（cache miss 率）	traffic
200	6.31 MB（0.0434）	4.57 MB
224	11.3 MB（0.0551）	6.57 MB
250	10.9 MB（0.0370）	8.14 MB
256	80.0 MB（0.2606）	9.64 MB
288	21.3 MB（0.0494）	12.2 MB
300	20.5 MB（0.0416）	14.7 MB

リへのアクセストラフィックが 1.5〜2 倍にもなる．これは，キャッシュ上でラインコンフリクトが起こるためと考えられる．特に行列サイズが 256 のときにはトラフィックの差は 8 倍程度もあり，図 7 でも性能が極端に悪い．これは文献 6) で self-interference と呼ぶ問題であり，行列サイズが 2 のべき乗のときには同一配列のブロック自体がキャッシュ上でコンフリクトするためである．SCIMA ではデータをオンチップメモリへ転送する際のアロケーションをプログラムで制御できるためこれらの問題は起こらず，オフチップメモリトラフィックを低く抑えることが可能となっている．

本章の仮定ではオフチップメモリのスループットを楽観的に大きくしているため，このトラフィック量の差がそれほど性能には影響していない．オフチップのスループットがオンチップに比べて相対的に小さくなった場合については 4.1.5 項で議論する．

キャッシュにおいては，レイテンシ 10 cycle 程度の L2 キャッシュを載せることが多いが，SCIMA（latency = 40 cycle）とキャッシュ（latency = 10 cycle）を比較しても，キャッシュは SCIMA の 9 割ほどの性能しか出ていない．これは，L1 キャッシュと L2 キャッシュから構成されるアーキテクチャよりも，SCMIA の方が性能的にも実装面積的にも有利であることを示している．

4.1.4 浮動小数点演算器を増やした場合

将来的には，集積度の向上によりさらに多くの浮動小数点演算器を LSI 上に実装することが比較的容易になると考えられる．そこで浮動小数点演算器数のみを増加させてみた．オフチップメモリのレイテンシは 40 cycle，および 10 cycle の場合について，行列サイズ 200 のときの性能を**図 8** に示す．

図 8 では，浮動小数点演算器数 2 で SCIMA とキャッシュともに性能が頭打ちになっている．これは，今回のプログラムが (ijk) = (441) でアンローリングしたレジスタブロッキングを採用しているためである．このとき，最内ループでは load 回数 8 に対して 16 回の multply-add 演算が行われるため，load/store 数：演算器数の比を 1 : 2 以上に増やしても効果がない．

SCIMA とキャッシュを比較すると，浮動小数点演算器を 2 に増加させた場合の性能向上率は SCIMA の方が良い．たとえばレイテンシ 10 cycle のとき，SCIMA は 1.48 倍だがキャッシュは 1.31 倍であり，SCIMA の方が 1.13 倍ほど向上率が良い．また，レイテンシ 40 cycle の場合は，この向上率の比は 1.25 倍と広がる．これは，プロセッサ内部の処理が高速になるにつれオフチップメモリアクセスによる性能低下の影響が深刻になるためである．オフチップメモリアクセスがボトルネックであるキャッシュの場合，集積度向上の恩恵が性能向上につながりにくいことを表している．

4.1.5 同時 load/store 数を増やした場合

今後の半導体技術の動向を考えると，ピンボトルネックの影響を受けるオフチップメモリのバンド幅は，キャッシュやオンチップメモリのバンド幅に比べ，相対的に小さくなると思われる．そこで，オンチップメモリとキャッシュのバンド幅を向上させ，1 cycle に発行可能な load/store 数を増加させた場合の性能を評価した．このとき，load/store 数：浮動小数点演算器数の比が 1 : 2 になるように浮動小数点演算器数も増加した．**図 9** に評価結果を示す．

図 9 より，SCIMA の場合には load/store 数が 1 から 2 になったときに，レイテンシが 10 cycle の場合でほぼ 2 倍，レイテンシが 40 cycle の場合でも 1.9 倍の性能向上が得られるが，キャッシュの場合にはレイテンシが 10 cycle の場合で 1.6 倍の性能向上にとどまり，40 cycle の場合ではほとんど性能向上しない．これは，表 3 で示したとおり，キャッシュではオフチッ

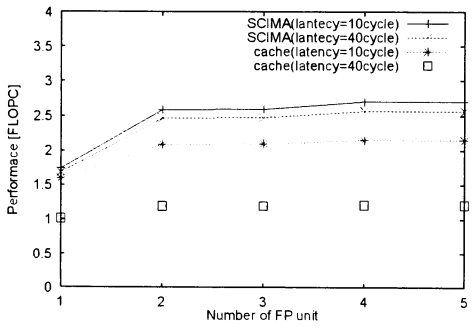

図 8 浮動小数点演算器数を変化させた場合
Fig. 8 Performance in increasing floating-point functional units.

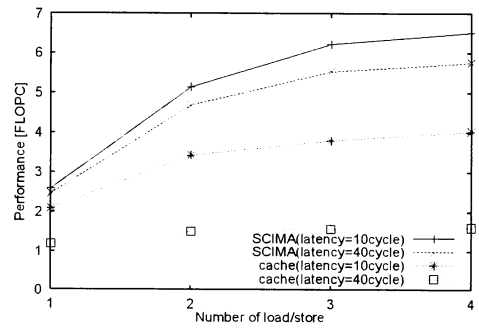

図 9 load/store 数を変化させた場合（load/store：演算器数 = 1 : 2）
Fig. 9 Performance in increasing load/store units.

プトラフィックがSCIMAの1.5ないし2倍もあるため，オフチップメモリスループットを相対的に小さくするとその影響が顕著になってくるためである．これはすなわち，将来的にはオフチップトラフィックを低減することの重要性が増すことを示している．したがって，オンチップメモリ上のデータのアロケーションを制御できるために，ラインコンフリクトなどによる意図しないオフチップトラフィックを抑えることのできるSCIMAの有効性は今後さらに増すと思われる．

なお，SCIMAにおいてもload/store数を3以上にすると性能向上率が鈍っているが，これはレジスタ32個の制約のもとではデータ依存関係により命令レベルの並列度が抑えられるためである．レジスタ数を増やしアンローリング回数を増やせば，性能はさらに向上するものと思われる．

4.2 QCD計算

4.2.1 評価条件

4.1節の行列積計算ではオンチップメモリのみとキャッシュのみの場合の比較に注力したが，本節ではさらに，2.4節で述べたオンチップメモリとキャッシュの統合機構を用いることによるSCIMAの有効性について評価する．評価においては下記の仮定をおいた．

- レジスタ数：整数，浮動小数点ともに32個．レジスタ拡張をしないので2.3.2項で述べたSCIMA用のmultiple命令は用いない．
- 同時発行命令数：整数演算8，浮動小数点演算（積和型）4，ロードストア4，すなわちキャッシュ・オンチップメモリのスループットは最大で32 B/cycle．
- オンチップ記憶の総容量を2 MBとし，512 KB/way，4 wayアソシアティブキャッシュに対して，オンチップメモリとキャッシュの割合を変更する．
- キャッシュラインサイズ：32 B．
- オンチップメモリ：pageサイズ4 KB，カラー数8．
- 浮動小数点積和演算命令レイテンシ：4 cycle．
- load/store命令レイテンシ：キャッシュ・オンチップメモリともに1 cycle．
- オフチップメモリスループット：8 B/cycle．
- オフチップメモリレイテンシ：40，10，0 cycle．

オンチップ記憶を2 MBとしているが，HP社のPA-8500マイクロプロセッサでは1.5 MBの一次キャッシュを内蔵しており，すでに実現可能な範囲にきていると思われる．また，浮動小数点演算，オンチップ記憶のスループット，オフチップのスループットも現行技術で可能なものとして仮定した．

表4 キャッシュとオンチップメモリの構成
Table 4 Configuration of the cache and on-chip memory on QCD evaluation.

	キャッシュ容量（連想度）	オンチップメモリ容量
(a)	2 MB（4way）	0 MB
(b)	1.5 MB（3way）	0.5 MB
(c)	1 MB（2way）	1 MB

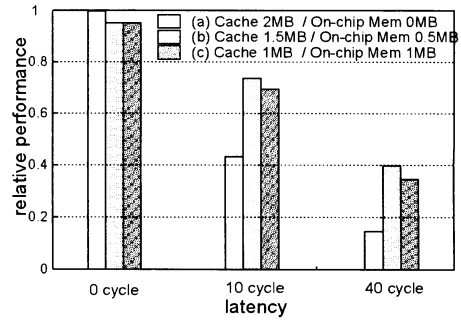

図10 オフチップメモリレイテンシを変化させたときの相対性能
Fig. 10 Relative performance for various off-chip memory latency.

オンチップ記憶の総容量を2 MB（512 KB/way，4 way）としているので，2.4節の機構を用いて，具体的には**表4**に示す(a)〜(c)の3通りにオンチップメモリとキャッシュの割合を変更して評価を行う．キャッシュ容量0MBも可能であるが，3.3.2項で述べたように再利用性の高いG，R，B，V，Tをキャッシュに載せるプログラムを想定しており，明らかに性能が悪くなるので評価対象から外した．

4.2.2 評価結果

図10に，レイテンシを変化させた場合の表4に示す3つの構成における性能を示す．縦軸には，(a)の構成におけるレイテンシ0 cycleのときの性能を基準とした相対性能をとった．

この図から以下のことが分かる．まず，レイテンシが0 cycleのとき，3つの構成の中では(a)が5%ほど高い．しかし，レイテンシが増えたときの性能低下は(a)が最も大きく，(b)のオンチップメモリ0.5 MBの構成が最も性能が良い．たとえば，レイテンシが40 cycleのとき，(b)は(a)に対して2.7倍の性能を示す．この原因はオフチップメモリトラフィック量から説明できる．オンチップメモリとキャッシュの割合を変化させた場合のメモリトラフィックを**表5**に示す．

表5の(a)と(b)の比較から分かるように，オンチップメモリを0 MBから0.5 MBに増やすことで，オフチップメモリトラフィックを2割近く削減できる．これは，再利用性のタイムスパンの長いU，Mが

表 5 QCD 計算におけるオフチップメモリトラフィック
Table 5 Off-chip memory traffic on QCD computation.

	キャッシュ	オンチップメモリ	合計
構成 (a)	18.0 MB	0 MB	18.0 MB
構成 (b)	4.2 MB	10.4 MB	14.6 MB
構成 (c)	5.3 MB	8.6 MB	13.9 MB

キャッシュを経由しないようにしたために，タイムスパンが短くすぐに再利用される G，R，B，V，T がキャッシュ上で U，M との干渉を受けず，意図に反してキャッシュから追い出されなくなるため，および，オンチップメモリ上に U を固定して載せることができるため，タイムスパンの長い U の再利用性も活用できるようになるからである．これは，オンチップメモリにおけるデータのアロケーション/リプレースメントをソフトウェア制御可能にしたことの利点である．レイテンシが大きくなるとオフチップメモリの実効的なスループットが低下するため，オフチップメモリトラフィックの総量が多い (a) は，レイテンシの増大にともなって性能が大きく低下する．

構成 (b)，(c) では，再利用性に乏しい U，M のデータがすべてオンチップメモリ経由となるため (a) よりもオンチップメモリのトラフィックは増える．レイテンシが 0 cycle のときに若干 (a) より性能が下がるのはこのためと考えられる．キャッシュは lock-up free なのに対し，オンチップメモリへの page-load は 2.5 節で述べた厳しい順序保証制御をするため，page-load の間は後続の命令はストールする．レイテンシが小さくスループットが比較的十分なときは，このストール時間による性能低下の影響が出る．しかしレイテンシが大きいときは，この影響よりもトラフィックが少ないことが効くため (a) よりも性能が良くなる．

(b) と (c) を比較すると，(c) の方がオンチップメモリ容量が増えるためにオンチップメモリ上により多くの U を載せることができるため，オンチップメモリに関するトラフィックは減少するが，逆に，キャッシュ容量が減少し連想度が減少することでミス率が増加してキャッシュに関するトラフィックが増える．合計のトラフィックでは (c) の方が小さいものの，転送サイズが小さい方が実効的なスループットは大きく低下するため，32 B ライン転送が多い (c) の方が性能が悪くなると考えられる．

将来的にはオフチップレイテンシはさらに大きくなると思われる．レイテンシが大きくなると実効的なオフチップメモリのスループットは低下するため，できるだけオフチップメモリトラフィックを減らし，アクセスするデータ単位を大きくすることが重要である．表 5 は，SCIMA が提供するオンチップメモリがこの両方に対する解決を与えていることを表しており，SCIMA の有効性は今後ますます強まると考えられる．

5. まとめと今後の課題

本論文では，プロセッサチップ上に高速なソフトウェアによる制御が可能なメモリを載せた新しいプロセッサアーキテクチャSCIMA を提案し，そのアーキテクチャ，および，行列積計算と QCD 計算における性能評価を示した．SCIMA は，従来のアーキテクチャとの上位互換性を保つことができるが，キャッシュとオンチップメモリの割合をある程度自由に変更することができるため，たとえばキャッシュのみの構成とすることで従来とまったく同じメモリ階層を提供することも可能なアーキテクチャである．

行列積計算における評価では，オンチップメモリ上のデータアロケーション/リプレースメントをソフトウェアで制御可能なため，キャッシュ上での不要なコンフリクトを防ぐことができ，オフチップメモリトラフィックを 2/3 から 1/2 程度に抑えることができた．また，オンチップメモリ・オフチップメモリ間でのデータ転送を大きな粒度で行えるため，キャッシュに比べオフチップメモリレイテンシの影響を受けないことも分かった．そのため，たとえばオフチップメモリレイテンシ 10 cycle のときのキャッシュ構成より，オフチップレイテンシ 40 cycle の SCIMA の方が性能が良かった．このことは，大容量の L2 キャッシュを用いた従来のアーキテクチャよりも SCIMA の方が性能的にも実装面積的にも有利であることを示している．さらに，今後の半導体技術の進歩により，浮動小数点演算器数の増加あるいはチップ上のメモリのバンド幅の向上がもたらされる場合，SCIMA はキャッシュに比べ，その半導体技術の進展に見合う性能向上が得られるアーキテクチャであることも分かった．

実アプリケーションである QCD 計算においても，データのアロケーション/リプレースメントがソフトウェア制御可能であるという特徴を用いることで再利用性が異なる配列どうしのキャッシュ上での干渉を防ぐことができた．そのため，オフチップメモリトラフィックを 2 割近く抑えることができオフチップレイテンシ 40 cycle のときには従来のキャッシュ構成のものより 2.7 倍の性能向上を達成することができた．

これらの結果から，提案する SCIMA は，ハイパフォーマンスコンピューティングに適したアーキテクチャであると結論できる．今後の課題としては，他の実アプリケーションにおける評価，追加・拡張命令の

ためのコンパイラの作成，具体的な設計によるクロック周波数に与える影響の考察，および，並列計算機への適用検討があげられる．

謝辞 本研究を行うにあたり，ご助言，ご討論いただいた筑波大学計算物理学研究センターの関係者の皆様，ならびに東京大学南谷崇教授に感謝いたします．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，文部省科学研究費補助金（基盤研究 (C) No.10680335）によるものである．

参 考 文 献

1) Callahan, D. and Porterfield, A.: Data Cache Performance of Supercomputer Applications, *Proc. Supercomputing '91*, pp.564–572 (1990).
2) Chen, T. and Baer, J.L.: A Performance Study of Software and Hardware Data Prefetching Scheme, *Proc. 21th Int'l Symp. on Computer Architecture*, pp.223–232 (1994).
3) Burger, D., Goodman, J. and Kagi, A.: Memory Bandwidth Limitation of Future Microprocessor, *Proc. 23rd Int'l Symp. on Computer Architecture*, pp.78–89 (1996).
4) Lam, M., Rothberg, E. and Wolf, M.: The cache performance and optimizations of Blocked Algorithms, *Proc. ASPLOS-IV*, pp.63–74 (1991).
5) Coleman, S. and McKinley, K.S.: Tile size selection using cache organization and data layout, *Proc. PLDI*, pp.279–289 (1995).
6) Panda, P., Nakamura, H., Dutt, N. and Nicolau, A.: Augmenting Loop Tiling with Data Alignment for Improved Cache Performance, *IEEE Trans. Comput.*, Vol.48, No.2, pp.142–149 (1999).
7) Cooper, K. and Harvey, T.: Compiler-controlled memory, *Proc. ASPLOS-VIII*, pp.2–11 (1998).
8) 大河原英喜，中村 宏，吉江友照，金谷和至：ハイパフォーマンスコンピューティング適したメモリ階層の検討，情報処理学会研究報告，ARC-133, pp.55–60 (May 1999).
9) Patterson, D., Anderson, T., Cardwell N., Fromm R., Keeton K., Kozyrakis C., Thomas, R. and Yelick, K.: A Case for Intelligent RAM: IRAM, *IEEE Micro* (Apr. 1997).
10) 村上和彰，岩下茂信，宮嶋浩志，白川 暁，吉井卓：メモリーマルチプロセッサー体型 ASSP（Application-Specific Standard Product）アーキテクチャ：*PPRAM*，信学技報，ICD96-13, CPSY96-13, FTS96-13 (Apr. 1996).
11) Elliott, D.G., Snelgrove, W.M. and Stumm, M.: Computational RAM: A Memory-SIMD Hybrid and its Application to DSP, *Proc. CICC*, pp.30.6.1–30.6.4 (May 1992).
12) Sony's emotionally charged chip, *MICROPROCESSOR REPORT*, Vol.13, No.5 (1999).
13) Strongarm speed to triple, *MICROPROCESSOR REPORT*, Vol.13, No.6 (1999).
14) SH-4 ハードウェアマニュアル. http://www.hitachi.co.jp/Sicd/Japanese/Products/micom/shmicom.htm (1998).
15) 近藤正章，坂井修一，朴 泰祐，中村 宏：HPC向けプロセッサのメモリアーキテクチャの基本構成，情報処理学会研究報告，ARC-134, pp.1–6 (Aug. 1999).
16) 専用並列計算機による場の物理の研究，筑波大学計算物理学研究センター平成 6 年 8 月研究進捗状況報告書 (1994).
17) 大河原英喜，近藤正章，中村 宏，朴 泰祐：ハイパフォーマンスコンピューティング適したメモリアーキテクチャの予備評価，情報処理学会研究報告，ARC-136 (Jan. 2000).
18) Whaley, R.C. and Dongarra, J.J.: Automatically Tuned Linear Algebra Software (ATLAS). http://www.netlib.org/atlas/index.html

(平成 12 年 2 月 8 日受付)
(平成 12 年 6 月 2 日採録)

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．

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．

大河原英喜

昭和50年生．平成10年東京大学工学部計数工学科卒業．平成12年同大学大学院工学系研究科計数工学専攻修士課程修了．平成12年（株）富士通研究所入社．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．