ハイパフォーマンス
コンピューティング 83－6
（2000．10．6）

NASPB CG, FT における SCIMA の性能評価

岩本 貢[†] 渡辺亮介[†] 近藤正章[†]
中村 宏[†] 朴 泰祐[††]

大規模な科学技術計算において従来のメモリアーキテクチャは有効ではなく，オフチップメモリへのアクセスが頻発するような場合に計算速度は大きく低下する．そこでチップ上のデータを有効利用することが重要になる．ところが，従来用いられるキャッシュアーキテクチャはいくつかの問題点がある．そこで，キャッシュの一部をアドレス指定可能なメモリとして用いることで，ユーザによるチップ上でのデータの制御が可能なオンチップメモリ記憶を実装するプロセッサアーキテクチャ SCIMA が提案されている．本稿では、Nas Parallel Benchmarks の CG, FT を例にとり SCIMA アーキテクチャが従来のキャッシュアーキテクチャより有効であることをシミュレーションを用いて示す．

Performance evaluation of SCIMA for NASPB Kernel CG, FT

MITSUGU IWAMOTO,[†] RYOSUKE WATANABE,[†]
MASAAKI KONDO,[†] HIROSHI NAKAMURA[†] and TAISUKE BOKU[††]

The performance gap between processor and memory will be growing in the near future. This is a serious problem especially in high performance computing. In order to overcome this problem, we have presented the new VLSI architecture called SCIMA. SCIMA has more effective data management ability than the conventional cache-based architecture, because data replacement and the data allocation of on-chip memory can be controlled by software. In this paper, we show the effectiveness of SCIMA compared to the cache-based architecture with the clock level simulator for the Nas parallel Benchmarks CG, FT as examples.

1. はじめに

近年のプロセッサはトランジスタ集積度の向上により，クロック周波数の向上，命令並列性の有効利用による性能向上が著しい．ところが，CPU の速度向上に比べメモリの速度向上が遅く，結果としてメモリの性能速度に計算速度が大きく影響されている[1)2)]．そのためにデータを高速なチップ上のメモリ（一般的にはキャッシュ）にいかに効率よくもってくるかが重要になる．

ところがキャッシュはデータのリプレースメントやアロケーションがハードウェアによって制御されているため配列内干渉，配列間干渉により無駄なオフチップトラフィックが発生し，性能が下がることがある．さらに，ライン単位でオフチップへのアクセスを行うので規則的 (例えば連続) な load/store にもライン単位毎にオフチップメモリレイテンシがかかるという問題も生じる．

そこで HPC 分野のアプリケーションを対象とし，ユーザがソフトウェアでデータ制御可能なオンチップメモリを実装し，チップ上のデータの制御を効率的に行うことを目的としたアーキテクチャ，SCIMA (Software Controlled Integrated Memory Architecture) が提案されている[4)5)]．オンチップメモリを用いると次のような利点があると考えられる．

- 必要なデータをオンチップメモリに載せる際にその場所が指定できるので，必要なデータをオフチップへ追い出すことなくチップ上で再利用することができる．またキャッシュ上で配列間干渉を起すデータをオンチップメモリに載せることで配列間干渉をなくすことができる．
- バースト転送によりライン転送に比べオフチップメモリへのアクセスレイテンシの総和を減らすことができる．

SCIMA の構成を図 1 に示す．また，従来のキャッシュアーキテクチャおよび SCIMA のメモリ階層の比較を図 2 に示す．従来ハードウェア制御であったキャッシュとオフチップメモリとの転送に加え，レジスタ-オフチップメモリ間，オンチップ-オフチップメモリ間の転送がソフトウェアで制御可能になっている．

† 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

本稿では Nas Parallel Benchmarks (NASPB)[6] のうち，CG, FT を例にとり SCIMA がキャッシュを用いたアーキテクチャより有効であることを示す．

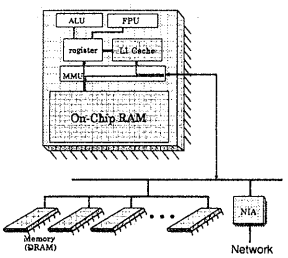

図 1 SCIMA の構成

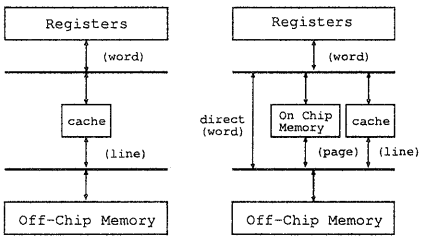

図 2 キャッシュアーキテクチャと SCIMA のメモリ階層の比較

本稿の構成は以下の通りである．2 章で SCIMA のアーキテクチャの概要を示す．3 章で評価環境について述べ，4 章でシミュレートするベンチマークについて説明し，SCIMA を用いた場合の最適化の戦略について述べる．4 章で示した戦略に基づいたシミュレーション結果を 5 章で示し考察を加える．最後に 6 章でまとめと今後の課題を述べる．

2. SCIMA

2.1 概　　要

本節では文献[4][5] をもとに，SCIMA アーキテクチャの概要を示す．SCIMA では，プロセッサ上にキャッシュに加えアドレス指定可能なオンチップメモリを搭載する．オンチップメモリ上のデータのアロケーションやデータのリプレースメントはユーザがソフトウェアで制御する．全てのデータの制御をユーザが行うのは不可能なので，性能向上につながると考えられるいくつかのデータをオンチップメモリに載せるように指定し，その他のデータはキャッシュで扱い，ハードウェアで制御を行う．

SCIMA では，ハードウェア的にはオンチップメモリはキャッシュと統合された SRAM になっており，総容量一定のもと，キャッシュとオンチップメモリの容量の比を可変に出来る．具体的には，SCIMA はキャッシュのいくつかの way をオンチップメモリとして用いる構成をとっている．例えば今回のシミュレーションでは，オンチップメモリを用いる際，32kB, 4way キャッシュのうち，2way, 16kB をキャッシュとして用い，残りの 16kB をオンチップメモリとして用いることが出来る．このように SCIMA はキャッシュとオンチップメモリを統合した構成をとることで，キャッシュとオンチップメモリの容量比を可変に出来るようになっている．チップ上の記憶容量が決まったとき，最適な容量比はアプリケーションによって異なるため，このような構成は有効である．

2.2 拡張命令

ISA(命令セット) 上に新たに page-load, page-store と呼ばれる主記憶とオンチップメモリ間のデータ命令を追加する．これらの命令によるデータ転送は page と呼ばれる大きな (今回のシミュレーションでは 4kB) 粒度で行い，DMA 転送により実現される．ここでいう page とは，ページテーブル上のページとは異なり，オンチップメモリの管理単位を指す．

page-load, page-store 命令は，さらにブロックストライド転送をサポートする．これにより，オフチップメモリの不連続領域をパッキングしてオンチップメモリの連続領域に転送することが可能になり，多次元配列に関する演算が多い HPC 分野ではこのような機能は有効に活用されると考えられる．

3. 評価方法

3.1 評価条件

シミュレーションの時間を考慮し，評価対象として NASPB の class W を選んだ．一般の HPC 応用では，キャッシュの容量を越える大きなデータセットを用いると考えられるので，チップ上のメモリ (キャッシュ，オンチップメモリ) の容量が，class W のデータセットより小さい 32 kB を仮定する．そこで，キャッシュのみを用いる際は，キャッシュサイズ 32 kB，連想度を 4 とした．

オンチップメモリを用いる際には，キャッシュのみの場合と総容量を同じにし，オンチップメモリ 16 kB，キャッシュ16 kB とした．キャッシュはオンチップメモリと統合されているためキャッシュの連想度は 2 となる．なお，それ以外の演算器数などのパラメータは次のようにした．この仮定は CG, FT のシミュレーションに共通である．

- レジスタ数 : INT=32, FP=32.
- 演算ユニット数 : INT=2, FP(add,sub,mult)=1.
- load/store スループット : キャッシュ 1 word / cycle，オンチップメモリ 1 word/cycle. (常にトータルで 1 word / cycle となるようにする)
- オフチップスループット 4B/cycle.

- オンチップページサイズ：4kB.
- multiply-add 演算レイテンシ：4.
- load/store レイテンシ：キャッシュ，オンチップメモリ共に 2cycle.
- オフチップメモリアクセスレイテンシ：40cycle.
- reservation station エントリ数：INT,FP,LS 用各 32.

この条件下で，ラインサイズを 32, 64, 128, 256 B と変化させ，データをとる．

3.2 評価方法

SCIMA では 2.2 節で述べたアーキテクチャ拡張を行なうが，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なう．

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なうこととした．

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．

シミュレータは命令キャッシュは常にヒット，分岐予測は常に成功，という仮定のもとでシミュレーションを行うが，これはループ構造の多い HPC アプリケーションでは妥当な仮定であると考えている．

4. ベンチマークの概要，最適化の戦略

NASPB (Nas Parallel Benchmarks) とは科学計算における基本的サブルーチンライブラリレベルのプログラムを用いたベンチマークの代表的なものである．今回はそのうち CG, FT を取り上げる．本節ではプログラムの概要を説明し，SCIMA を用いる際の最適化の戦略について検討する．

4.1 Kernel CG

プログラムの概要

NASPB CG は大規模疎行列の固有値をもとめるプログラムである．固有値を求める反復計算の過程で逆行列を計算する必要が生じる．逆行列は陰解法により求められるが，その際に (疎行列) × (ベクトル) の計算 $q = Ap$ を行い，この計算が計算時間の大半を占める．最内ループは，`sum = sum + a(k) * p(colidx(k))` と書かれ，a を連続に読みながら，p を間接参照によりランダムにアクセスする．今回扱う class W のデータセットは，7000×7000 の行列 A の中に非負成分として a を 4MB(非零率 1%) もち，p は 7000 要素をもつ．sum, a, p はすべて倍精度である．`coldix(k)` は A 中の非負成分 `a(k)` の列番号を保持しており，整数配列である．特にメモリアクセスボトルネックを引き起こしやすい倍精度の p, a のデータアクセスの特徴をまとめると次のようになる．なお，a，p 共にデータサイズが大きいのでループを計算するあいだにキャッシュより大きいサイズのデータアクセスを行う．

a: アクセスは連続となるが再利用性は無い．

p: `colidx` を通して間接参照され，アクセスはランダムであるが，データの再利用性はある．

最適化の戦略

(**Cache-opt**) まず，キャッシュのみのアーキテクチャにおける最適化を行う．今回の仮定では，配列 p のデータサイズはキャッシュサイズより大きく，容量性のミスが起きると考えられるため，キャッシュブロッキング[3] を行う．キャッシュは 32kB, 4way キャッシュをであるが，ブロッキングのサイズはいくつかの実験の結果，配列 p を 7 分割するように設定する．分割された p はキャッシュの 1way (8kB) 分のサイズである．これにより p の再利用性を十分活用することで，オフチップへのトラフィックが減少し，その結果としてサイクル数が減少すると考えられる．しかし，ブロック内での配列間干渉によるラインコンフリクトが起きることが予想され，計算に必要な配列が追い出される可能性は完全には避けられない．以下ではこの戦略を Cache-opt と呼ぶ．

一方，a は連続に一度しか読まれない配列なので再利用性は見込めず，ブロッキングしても効果はない．またライン転送の初期レイテンシが多くかかる問題は依然として解決されない．そこで配列間干渉，初期レイテンシの問題を解決するためにオンチップメモリを用いた戦略を 2 つ検討する．

(**OCMp**) まず，ランダムにアクセスされる配列 p をオンチップメモリに載せることで p のと他の配列の配列間干渉を避けることができると考えられる．オンチップメモリに出来るだけ多くの p を載せるため，ブロッキングは p を 4 分割することとし，オンチップメモリに 4 分割された p を載せる．この方法を以下，OCMp と呼ぶ．

(**OCMa**) OCMp ではランダムに読まれる配列 p と他配列との配列間干渉は避けられるが，連続にアクセスされる a のライン毎に初期レイテンシがかかるという問題は解決されない．そこで，配列 a をオンチップメモリに載せるという戦略も考えられる．これにより大粒度でのデータ転送を活用することが可能になり，性能が向上すると考えられる．しかし，ランダムにアクセスされる配列 p と連続にアクセスされる他の配列がキャッシュ上で干渉する可能性がある．このような

コンフリクトを避けるため，ブロッキングはブロックサイズがキャッシュ 1way 分になる 7 分割を採用する．

4.2 Kernel FT

プログラムの概要

NASPB FT は 3 次元 FFT を行うプログラムである．class W でのデータサイズは $128\times128\times32$ で，各要素は複素数の倍精度である．もとのコードで，キャッシュ上での配列内干渉を避けるため，配列を scratch 配列と呼ばれる配列に一回コピーして FFT の計算を行っている (つまり，キャッシュでの最適化は行われており，もとのコードが既に Cache-opt になっている)．ラインサイズが 64B より大きい場合，y 方向への FFT 計算をする際の scratch 配列へのコピーに無駄なライン転送が生じ，性能低下を起す可能性がある．

さらに，コピーの転送粒度がラインサイズであることから，転送に初期レイテンシがかかる回数が増え，性能に影響をおよぼすと考えられる．

最適化の戦略

ライン単位で scratch 配列のコピーを行うため，3 次元から 2 次元分のデータを scratch 配列にコピーする際のライン単位の転送には無駄が生じるが，SCIMA はストライド転送が可能であるため，そのような不要な転送が起こらない．

また，scratch 配列へのコピーの転送粒度がラインであることから，この scratch 配列をオンチップメモリへの転送命令，page-load に置き換えれば転送粒度が大きくなることでの初期レイテンシのかかる回数の減少によるサイクル数の減少が見込める．

5. 評価結果，考察

ここでは 4 節で取り上げた最適化の変更を行わないプログラムを ORI と略記する．FT に関しては元のプログラムがキャッシュでの最適化を行っており，ORI は Cache-opt になる．なお，評価では CG, FT のサブルーチンの 1 イテレーションを計測する．

5.1 オフチップトラフィック

Kernel CG

まず，シミュレーション結果についてオフチップトラフィックから考察する．CG のオフチップトラフィックを図 3 に示す．またデータを表 1 に示した．ORI のラインサイズ 64B 以上のものに関してはトラフィックが大きいので省略した．

まず，ORI と Cache-opt を比較すると，ブロッキングの効果が現れ，ラインが 32B のとき 52%トラフィックが減少する．さらに Cache-opt と OCMp を比較すると，p による配列間干渉によるトラフィックが減少し，全てのラインサイズで 5〜6%のトラフィック減少が見られる．CG においては配列 p 以外の配列は連続に読まれ，再利用性のある p を適当なブロックサイズでキャッシュに載せることでラインコンフリクトをほとんど押さえられることがわかる．

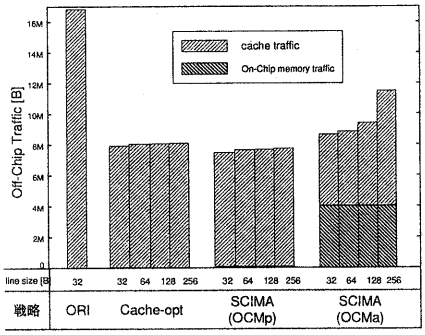

図 3 CG のオフチップトラフィック

一方，OCMa はブロッキングの効果から ORI よりはトラフィックが減少するが，16kB, 2way のキャッシュ上で配列 p と coldix など他配列が干渉を起こすために，Cache-opt や OCMp よりトラフィックが増える結果となっている．

Kernel FT

FT のオフチップトラフィックを図 4，またそのデータを表 2 に示す．まず，Cache-opt をみるとラインサイズが増加するに従い，トラフィックが大きく増加している．ラインサイズが 32, 64B のときは FFT を行う 3 次元配列のうち 2 次元を scratch 配列にコピーする際の無駄なライン転送は起きないが，FFT の計算時に scratch 配列と他の配列がラインコンフリクトを起し，トラフィックが増える．さらに，ラインサイズが 128, 256B のときはラインコンフリクトに加え，3 次元 FFT を行う際の scratch 配列へのコピーで無駄なライン転送が生じ，さらにトラフィックが増える．そのため，ラインサイズが 64B, 256B のときのトラフィックは明らかに大きくなる．

ところが SCIMA では，FFT で必要なデータが常にオンチップメモリ上にあることで，FFT の計算時に scratch 配列と他の配列がコンフリクトを起さないことと，主記憶からのストライド転送の効果により，ライン転送で起きる無駄なデータの転送もなくなる．その結果，ラインサイズによらないオフチップメモリトラフィックが達成できる．この効果により SCIMA は ORI よりラインサイズ 32B で 33%のオフチップトラフィック削減に成功している．

表 2 FT のオフチップトラフィック，単位/B (かっこ内はオンチップメモリトラフィック)

line	Cache-opt	SCIMA
32B	74959008	50334752(50331648)
64B	100802176	50334912(50331648)
128B	156150400	50335232(50331648)
256B	264067584	50335744(50331648)

表 1　CG のオフチップトラフィック，単位/B (かっこ内はオンチップメモリトラフィック)

line	ORI	Cache-opt	OCMp	OCMa
32B	16821280	7923360	7494816(65536)	8663344(4067216)
64B	24425088	8041344	7644160(65536)	8866896(4067216)
128B	38447872	8067200	7681280(65536)	9432848(4067216)
256B	65713408	8078080	7726848(65536)	11509648(4067216)

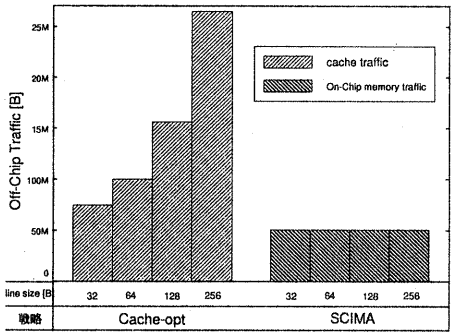

図 4　FT のオフチップトラフィック

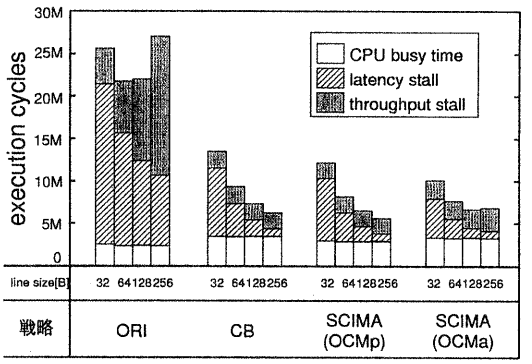

図 5　CG のサイクル数

5.2　実行サイクル数

実行サイクル数をシミュレーションにより求める．さらに評価の内訳を明らかにするために，実行サイクル数を以下のような 3 つの仮定のもとで測定した．

- C_{norm}: 所用サイクル数全体．3 節で示した評価条件で評価．
- C_{thinf}: オフチップメモリスループットが無限大であると仮定した場合．
- C_{ideal}: オフチップメモリスループットが無限大かつ，オフチップメモリレイテンシ 0 cycle と仮定した場合．

これらのデータに基づき，スループット不足による CPU の停止時間を示す throught stall を $C_{\mathrm{norm}} - C_{\mathrm{thinf}}$，初期レイテンシによる待ち時間に相当する latency stall を $C_{\mathrm{thinf}} - C_{\mathrm{ideal}}$ として求める．また，計算時間に相当する CPU busy time を C_{ideal} とする．

Kernel CG

結果を図 5 に示す．ラインサイズ 32B のときに注目すると，ORI からみて，Cache-opt は 1.9 倍，Cache-opt からみて OCMp は 1.1 倍，OCMa では 1.3 倍の性能向上が確認できる．まず，ORI はキャッシュの容量性ミスなどによるオフチップトラフィックが大きいため，サイクル数が他の戦略に比べ非常に多い．ラインサイズが増加するとオフチップとの転送回数が減ることで latency stall は減少する．ところが，ラインサイズの増加は無駄なライン転送を含むためオフチップトラフィックの増加をもたらし，throughput stall は増加する．それらのトレードオフの結果としてラインサイズが 64B のときが最も性能が良くなる．

ORI の容量性ミスをキャッシュブロッキングによって改善した Cache-opt はブロッキングによるオフチップトラフィックの減少が影響し，throughput stall の減少による性能向上が見られる．また，latency stall も転送回数の減少により大きく減少する．

さらに配列間干渉が原因でキャッシュから追い出されてしまう可能性のある配列 p をオンチップメモリに載せることで (OCMp)，配列間干渉によるトラフィック減少により throughputstall が減少する．トラフィックの項でこの効果が 5%程度であることは示した通りであり，throughput stall の減少も同様に 5%程度の減少である．また，throughput stall の減少とともに 4 分割された配列 p のオンチップメモリへの転送をページ単位で行うことによる latency stall の減少の効果も見られる．結果として全体でサイクル数は Cache-opt に比して 9〜11%減少する．

もっともサイズの大きな配列 a をオンチップに載せる OCMa の結果をみると latency stall が大幅に減少していることがわかる．しかし，配列 p をキャッシュに載せることでの配列間干渉が起き，thoughput stall の増加が確認できる．特にラインサイズが大きい場合にその傾向が顕著に現われ，性能はラインサイズ増加に比して単調に減少せずラインサイズが 128B のときに最高性能になる．

結果として，ラインサイズが 32B, 64B では OCMa が，ラインサイズ 128B, 256B では OCMp が最も性能が良い．

Kernel FT

FT のサイクル数を図 6 に示す．まず Cache-opt をラインサイズ別に比較する．ラインサイズが 32B, 64B のときは `scratch` 配列へのコピーに際し，ライン転送による不要な転送は起きないので throuput stall は変わらないが，転送粒度が大きくなるため latency stall が減少する．ところがラインサイズが 128B, 256B

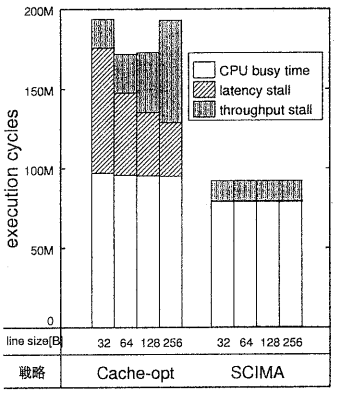

図 6 FT のサイクル数

のときはライン転送による無駄なデータ転送が起き，throughput stall が大きくなり性能が低下する．

一方 SCIMA は FFT 計算時の `scratch` 配列と他配列のコンフリクトや，ストライド転送の効果で無駄なライン転送も起きず，一定の throughput stall を達成している．また，大粒度転送の効果として latency stall も少ない．なお，SCIMA の CPU busy time が Cache-opt に比べ小さいのは，page-load, page-store を用いることでの実行命令数の減少による．結果としてラインサイズが 32B のとき SCIMA は Cache-opt に比べ 2.1 倍の性能向上を達成している．

6. おわりに

まず CG ではブロッキングの効果が顕著に現われることが分かった．そのうえで，

(1) キャッシュで避けられない配列間干渉を減らすことによる性能向上．

(2) 連続に読まれる配列の大粒度転送による latency stall の減少．

が SCIMA で実現できることを示し，シミュレーションを通してその性能向上の様子を示した．今回のデータセットでは (2) による性能向上のほうがより顕著であったが，(1) の効果は特にデータの局所性に依存し，データによって (1) による性能向上の効果が顕著な場合も十分考えうる．CG は (1),(2) による性能向上が見込めるが，容易に想像がつくように (1),(2) 戦略を組合わせて用いるとさらなる性能向上が期待できると考えられる．そのためには

- キャッシュ/オンチップメモリの容量比をどれくらいにするか
- 配列 `a`, `p` を一度にどれくらいオンチップメモリに載せるか

などを検討する必要がある．この戦略の評価については今後の課題としたい．

FT においては CG よりも顕著な性能向上が認められた．これは上記 (2) の要因に加え，

(3) 主記憶からのストライド転送による効果

が性能向上をもたらすと考えられる．FT に関してはこれらの効果が `scratch` 配列へのコピーをオンチップメモリ領域へのコピーに置き換えることで同時にえられる．

今回取り上げた NASPB CG, FT 以外にも，オンチップメモリを用いることにより性能向上を見込める例は報告されているが[4)5)]，SCIMA を用いる際の最適化の指針を模索することは今後も重要な研究課題である．そのためには他のベンチマークやアプリケーションによる性能評価をさらに行うことが必要であると考えている．

謝辞 本研究を行なうにあたり，御助言，御討論頂いた筑波大学計算物理学研究センターの関係者各位に感謝致します．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) によるものである．

参考文献

1) A. Salsbury, F. Pont, and A. Nowatzyk, "Missing the memory wall: The Case for Processor/ Memory Integration," Proc. of ISCA '96, pp. 90-101, May, 1996.

2) D. Burger, J. R. Goodman, and A. Kägi, "Memory Bandwidth Limitations of Future Microprocessors," Proc. of ISCA '96, pp. 78-89, 1996.

3) M. S. Lam, E. E. Rothberg, and M. E. Wolf, "The cache preformance and optimizations of Blocked Argorithm," Proc. of ASPLOS-IV, April, 1991.

4) 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，Vol 41, No. SIG 5(HPS 1), pp.15-27, 2000 年 8 月．

5) 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に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-136，2000 年 1 月．

6) D. Bailey, T. Harris, W. Saphir, R. Wijngaart, A. Woo, and M. Yarrow, "The NAS Parallel Benchmarks 2.0," NASA Ames Research Center Report, NAS-05-020, 1995. Available from `http://www.nas.nasa.gov/Software/NPB/`