オンチップマルチプロセッサ用半共有型疑似連想キャッシュ

井上敬介[†] 若林正樹[†]
木村克行[†] 天野英晴[†]

プロセス技術の発達により増えた集積度を，現在のプロセッサは持て余している．そこで，半導体資源を有効に利用するためにマルチプロセッサを 1 チップ上に実装することが提案されている．

本論文では，オンチップマルチプロセッサにおけるチップ内のキャッシュメモリの利用効率を上げる手法として，他のプロセッサのキャッシュを擬似的に自分の way に見せる pSAS (Pseudo Set Associative and Shared) キャッシュを提案する．pSAS キャッシュはキャッシュラインのコピーの数が減るため，半ば共有キャッシュの状態になり，実際にキャッシュされるデータの量を増やすことができる．オンチップマルチプロセッサ向けスヌープキャッシュプロトコルである新 Keio プロトコルキャッシュに対し，pSAS キャッシュは全体で 10%，最大で 16%性能が向上し，その効果が確認できた．

pSAS: Pseudo Set Associative and Shared Cache for on-chip multiprocessor

Keisuke Inoue,[†] Masaki Wakabayashi,[†] Katsuyuki Kimura[†] and Hideharu Amano[†]

A high performance microprocessor which issues multiple instructions has been implemented. However, the performance improvement of such microprocessors will become difficult because of its complicated structure. To utilize silicon resources efficiently, a on-chip multiprocessor has been widely researched.

In this paper, in order to increase effective utilization of on-chip cache memory of such a on-chip multiprocessor, pSAS(Pseudo Set Associative and Shared) cache is proposed. In this cache, cache module with snoop mechanism attached to the other processor in the same chip can be used as an extra way of its own cache. It combines advantages of both snoop and shared cache. Simulation results show that the performance is improved 10% in average, 16% at maximum.

1. はじめに

現在の高性能プロセッサは，スーパスカラや VLIW といった命令レベルの並列度を利用し複数命令を同時実行する技術で，処理能力を増やしてきた．一方，半導体技術の向上によりプロセスルールは着実に進み，1 チップ上に搭載可能なトランジスタ数は増加の一途を辿っている．このことによりさらにプロセッサの集積度を上げることが可能であるが，現状以上にプログラム中の命令レベルの並列度を引き出すことは難しく，演算器を増やしても性能は上がらない．そのため，半導体技術の向上により増加したチップ面積を十分活用することが難しい状況にある．

† 慶應義塾大学理工学部
Faculty of Science and Engineering, Keio University

チップ面積を有効利用する方法としては，主記憶混載，チップ上でマルチプロセッサを構成するオンチップマルチプロセッサ，およびその両方を用いたもの[1)~5)]が提案され，研究，試作が進んでいる．オンチップマルチプロセッサには従来のボード上での実装にはない新たな自由度があり，プロセッシングエレメント部に関してもマルチスレッド指向，共有レジスタ等の提案がされている．

我々は，従来のプロセッサコアを利用したオンチップマルチプロセッサでの，プロセッサの接続法やメモリアーキテクチャを検討している．

1 チップ上でマルチプロセッサを構成する手法の中には共有キャッシュ方式とスヌープキャッシュ方式がある．前者は

- 共有キャッシュが同一チップ内にある場合，強力なバスやクロスバにより内蔵したプロセッサと密

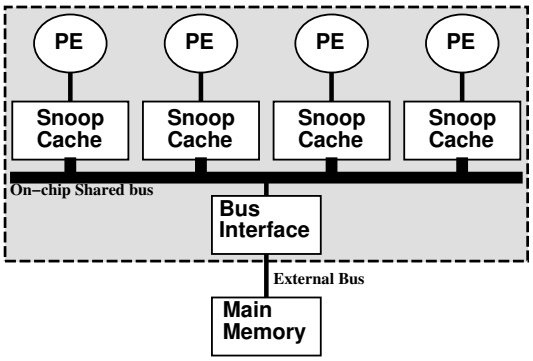

図 1 スヌープキャッシュの構成
Fig. 1 Snoop Cache Structure

に結合することができる．

- マルチポートメモリを用いると，調停作業を容易かつ高速に行なうことができる．
- メモリコンシステンシ維持が容易に行える．

といった利点があり，後者は共有バスのデータラインを強化し，アドレス送出やアービトレーションの時間を除けば，1 クロックでキャッシュライン 1 ライン分のデータを送ることができる強力な機能を持たせることが可能である．この 2 つを比較した結果 [6]，1 ポート共有キャッシュはアクセスのコンフリクトによる遅延が，マルチポート共有キャッシュはマルチポート化によるキャッシュメモリ量の減少やメモリアクセスの遅延が性能に影響し，高速で幅の広いバスを用いてキャッシュラインを 1 度に転送するスヌープキャッシュの方が良い性能を示すことが分かっている．

しかし，アプリケーションのワーキングセットの大きさによっては，スヌープキャッシュ方式が同一キャッシュラインのコピーを持つこでチップ全体でキャッシュできるデータ量が減り，1 ポート共有キャッシュの方が性能が良い場合もある．

本稿では以上を踏まえて，スヌープキャッシュの利点を残したままチップ内のキャッシュメモリを有効に使用する手法として，pSAS キャッシュを提案する．

以下 2章でまずオンチップマルチプロセッサにおけるスヌープキャッシュについて述べ，3章で pSAS キャッシュの構成について述べる．そして，残りの章で性能評価と考察を行う．

2. スヌープキャッシュ

本章では，スヌープキャッシュの構成と，シングルチップマルチプロセッサ上での実装に関して述べる．

2.1 スヌープキャッシュの構成

図 1 にここで検討するシングルチップマルチプロセッサ上のスヌープキャッシュの構成を示す．それぞれのプロセッシングエレメント (PE) は L2 キャッシュとしてスヌープキャッシュを持つ．同一チップ上での実装を考え，L2 キャッシュ間を接続するバスは，ラインサイズ分のバス幅を持つと仮定している．このバスを利用することにより，キャッシュ間のラインの転送自体は 1 クロックで終了する．

一方，主記憶はチップ外部に置かれているため，主記憶へのアクセスは通信バンド幅は I/O ピンネック，主記憶のアクセス速度などで制限を受ける．チップ内外の転送容量の差が大きくなると，主記憶へのアクセスは大きなペナルティになると考えられる．このことから主記憶へ極力アクセスしないスヌープキャッシュプロトコルの検討が行われた [4]．

また，共有変数を用いた同期もチップ内で高速に行えるよう，同期機構がキャッシュに組み込まれている [4]．

2.2 これまでの検討

我々は，オンチップマルチプロセッサ上でのスヌープキャッシュと共有キャッシュの性能を評価して比較した [6] が，スヌープキャッシュの共有キャッシュに対する利点は以下の通りである．

(1) ヒットした場合，他のプロセッサからのアクセスと衝突しない．
(2) 各プロセッサが専用にキャッシュを持つことができるため，密結合が可能で，アクセスクロック数を削減することができる．

しかし一方で，複数のプロセッサがそれぞれにデータをキャッシュするため，同一チップ内に複数のデータのコピーが生じる結果となり，チップ内のキャッシュメモリ全体として利用効率が悪くなる．

寺澤は，スヌープキャッシュのキャッシュ間転送能力を強化すると共に，同一チップ内の他のプロセッサのスヌープキャッシュの空いているブロックを自分のキャッシュの Victim Cache として利用する Z-Cache[7] を提案したが，利用された側のキャッシュに不要なブロックが挿入されるための性能低下により，ほとんど性能が改善されなかった．

3. pSAS キャッシュ

キャッシュの性能改善方法に，まずダイレクト・マップ方式でアクセスして，ヒットしない場合に，多少時間がかかっても複数セットにアクセスする疑似セット・アソシアティブ方式 [8] がある．これは，ダイレクト・マップ方式の高速なキャッシュアクセスと，セット・アソシアティブ方式のキャッシュヒット率の良さを兼

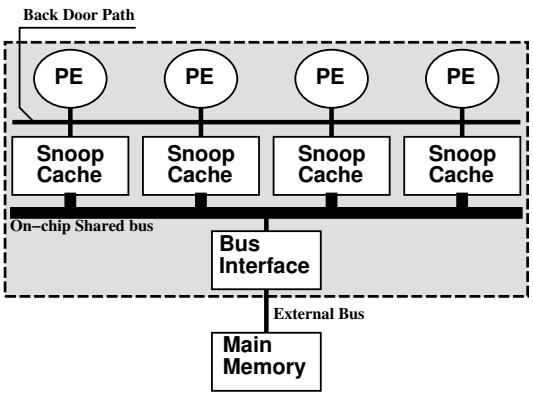

図 2 pSAS キャッシュの構成
Fig. 2 pSAS Cache Structure

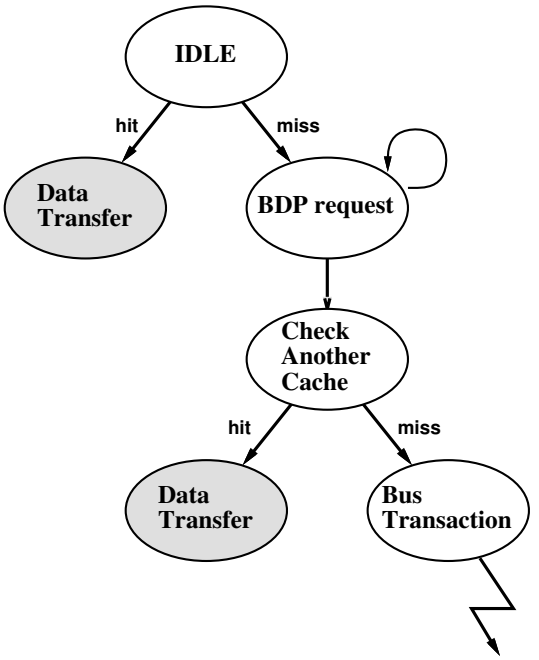

図 3 pSAS キャッシュコントローラの状態遷移図
Fig. 3 pSAS Cache State Machine

ね備えている．一方，オンチップマルチプロセッサ上では専用のデータパスを設けることにより，他のプロセッサのキャッシュに多少時間をかけてアクセスすることが可能である．そこで，他のプロセッサのスヌープキャッシュを自分のプロセッサの疑似セットとして扱い，まず，通常のキャッシュのように自分のプロセッサにアクセスし，キャッシュミスした場合に，他のプロセッサのキャッシュに直接アクセスすることで，スヌープキャッシュの問題点である全体のメモリ利用効率の悪さを改善できるのではないかと考えた．この方法は，他のキャッシュを自分のキャッシュの疑似セットとして扱い，またそれは共有キャッシュとして見ることもできることから pSAS (Pseudo Set Associative and Shared) キャッシュと呼ぶ．

3.1 構成と動作

図 **2** に pSAS キャッシュの構成を示す．各プロセッサはキャッシュを持っており，他のキャッシュを自分のキャッシュの疑似セットとして扱うためのデータパスとして Back Door Path (BDP) が追加されている．BDP のビット幅はプロセッサコアの 1 ワード分程度である．pSAS キャッシュはこの BDP を通して自分のキャッシュより数クロックの遅延で他のキャッシュにアクセスすることができる．pSAS キャッシュは極力自分のキャッシュへのアクセスは他から阻害されない方針で設計されており，例えばタグメモリは自プロセッサと BDP 側で同時に読み出すことができる．

次に，pSAS キャッシュの動作について説明する．図 **3** にプロセッサがメモリアクセスを行った場合の pSAS キャッシュのステートマシンの概略を示す．まず，自分のキャッシュでキャッシュミスとなった場合，BDP アービタに request を出す．BDP を取得後，BDP 経由でアドレスを全キャッシュタグへ流して他のキャッシュでのヒット判定を行う．キャッシュヒットした場合は，プロセッサは BDP を通してそのキャッシュへ直接アクセスする．この時キャッシュラインの移動・コピーは行われない．

他のキャッシュでのキャッシュミス判定は 2 通り考えることができる．

(1) チップ内に同一キャッシュラインの存在を許さない方法．他の全キャッシュのタグが busy 状態でなくなるまで待ち，確実にチップ内にそのキャッシュラインが存在しないことを確認した後，図 2 にある Bus Interface を通してチップ外にある主記憶へアクセスする．

(2) チップ内に同一キャッシュラインの存在を許す方法．BDP を確保した時点であるキャッシュでアクセスが衝突し，残りのキャッシュでキャッシュヒットしなかった場合，衝突が発生したキャッシュを待たずにキャッシュミスと見なしてスヌープキャッシュのバストランザクションを発生させる．この方式は pSAS キャッシュにバススヌープ機能が必要である．また，write 時はキャッシュコヒーレンスを保つために，全キャッシュメモリにアクセスできるまで待つ必要がある．

本稿では pSAS キャッシュがあるキャッシュへの混雑を自動的に避ける weak competitive な動作を期待して (2) の方法を中心に評価を行なっている．

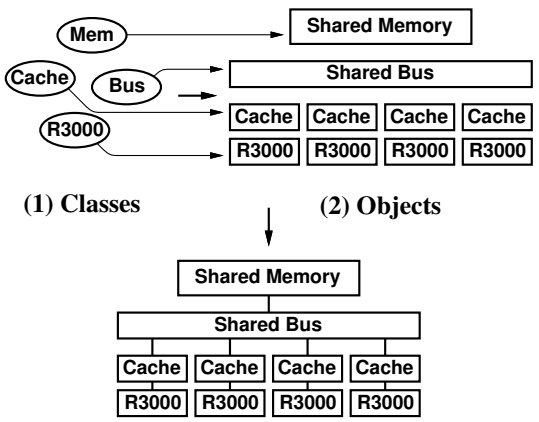

図 4 ISIS の概観
Fig. 4 ISIS is Instruction level Simulator

BDP の本数は 1 本としている．BDP の本数が増えれば，BDP の取得待ち時間が減り，パフォーマンスが上がることが考えられるが，他のキャッシュから read のみ行える VLNW キャッシュ[9] で予備評価を行い，あまり効果がないことが分かっている．また，同一キャッシュラインの存在を許す場合，BDP が複数あると write の競合の処理とメモリコンシステンシ維持が煩雑であるという問題がある．

4. 評価環境と条件

pSAS キャッシュの評価には命令レベルシミュレータによるシミュレーションを用いた．評価環境，条件について述べる．

4.1 並列計算機シミュレータ ISIS

並列計算機シミュレータ ISIS[10] は，様々な並列計算機シミュレータに共通する機能ブロックを抽出し，より汎用性を高めてライブラリ化したシミュレーション環境である．ISIS を使用してシミュレータを実装する場合，ライブラリで用意された機能ブロックをそのまま使用することができるため，シミュレータ実装者は対象の並列計算機特有の機能のみを実装するだけで良い．

図 4 に ISIS の構成を示す．ISIS では，計算機内部の個々の機能ブロックをユニットとしてカプセル化し，各ユニットを独立実装する方式を採用している．特定の並列計算機のシミュレータを実装する場合には，必要なユニットを結合させることで所望の並列計算機を構成すればよい．既に R3000 プロセッサのクロックレベルシミュレータやメモリ，キャッシュなどのユニットがライブラリで用意されているため，特殊なハードウェアを用いていない並列計算機であれば実装済みのユニットを組み合わせるだけでシミュレータを実装することができる．また，ライブラリに用意されていないユニットをシミュレータ実装者が追加することができるため，どのようなアーキテクチャを持つ並列計算機であっても対応が可能である．

ユニット間の結合と通信のインタフェースを規定するために，情報を仲介する存在としてパケット，結合を仲介する存在としてポートという概念を導入している．パケットは，相互接続されたユニット間でやり取りされる全ての情報を表現する．例えばルータ間で送受信されるフリット，プロセッサがバスに出力するメモリアクセス要求などがパケットである．ポートは，ユニット同士を結合している通信路への入出力端子を表現する．接続したいユニット同士のポートを接続することで，ユニット間の通信路が構築される．ユニット間の通信を明確に規定することで，個々のユニット同士の依存関係を最小限に留めている．

4.2 アプリケーション

評価に用いるアプリケーションプログラムとして，SPLASH2[11] ベンチマーク集から以下のアプリケーションを選択した．アプリケーションのパラメータはオンチップキャッシュに載りきらないデータサイズとなるように選んでいる．

- FFT
  このプログラムでは，6 ステップの FFT アルゴリズムを用い，1 次元複素数配列の高速フーリエ変換を行うものである．等分点の数を 65536 に設定して実行した．
- LU
  このプログラムでは，行列を右下 3 角行列と右上 3 角行列に分解する．この計算では，ローカリティを引き出するために個々の行列をいくつかのブロックに分けて計算する．256×256 の行列を用い，ブロックのサイズは 16 に指定した．
- RADIX
  整数の並列基数ソートを行う，要素数は 524288 個に設定した．
- OCEAN
  ガウス–ザイデル法により海洋中の反主流と境界流の影響をシミュレートする．各プロセッサにはサブグリッドが割り当てられ，その境界で通信が生じる．66×66 のグリッドで実行した．

4.3 構　　成

ここで想定するオンチップマルチプロセッサでは小 ~ 中規模の RISC プロセッサを用いることにより，1

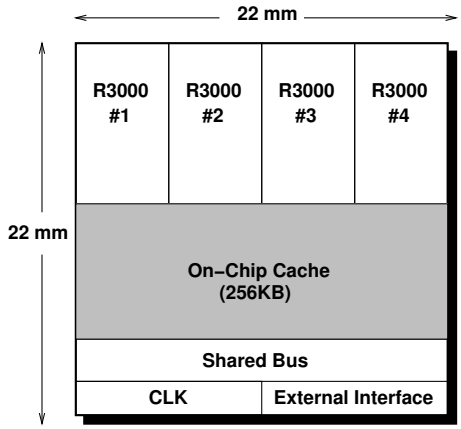

図 5 フロアプラン
Fig. 5 Floorplan

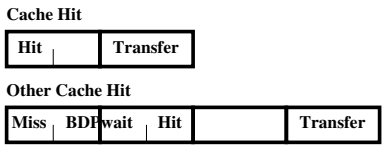

図 6 アクセス時間
Fig. 6 Access Ratency

プロセッシングエレメントあたりのチップ面積を小さくし，搭載プロセッサ数とキャッシュメモリの増加を狙っている．今回は図 5 に示すような，$0.25\mu m$ のプロセス技術で $500mm^2$ 程度のダイサイズを持つチップを想定した．

プロセッシングエレメントは R3000 クラスの RISC プロセッサを 4 台搭載している．キャッシュサイズは全体で 256KB 程度とした．4 プロセッサあるので 1 プロセッサあたりのキャッシュ容量は 64KB となる．

次にアクセス時間について述べる．アクセス時間は使用するデバイス，テクノロジ，レイアウトに依存するため，ここで述べるクロック数はアクセスレイテンシの比率を示すものとする．チップ外の主記憶へのアクセスは高周波数動作時の DRAM のアクセスタイム等を考慮し，イニシャルデータの到着までのアクセスペナルティを 50 クロックとする．それ以降は 1 クロックに 1 データが転送される．また，キャッシュのアクセス時間は自分のキャッシュにヒットした場合は 2 クロック，他のキャッシュにヒットした場合は最低 4 クロックのアクセス時間を必要とする．また，今回は CPU の L1 データキャッシュは disable している．図 **6** に概略を示す．

4.4 条 件

以下にシミュレーション条件をまとめる．プロセッサ数，キャッシュ容量は，先に述べたチップ面積を考慮して定めてある．この構成で各アプリケーションの初期化を除いた実行時間を測定した．

- プロセッサ数 4
- チップ内のキャッシュ容量 256KB
- 各プロセッサのキャッシュ容量 64KB
- セット数 1, 2, 4(way)
- キャッシュラインサイズ 16, 32(byte)
- 共有バス幅 (キャッシュラインサイズと同じ)
- 自キャッシュヒット時のアクセス時間 2clk
- 他キャッシュヒット時のアクセス時間 4clk
- 主記憶のイニシャルアクセス時間 50clk

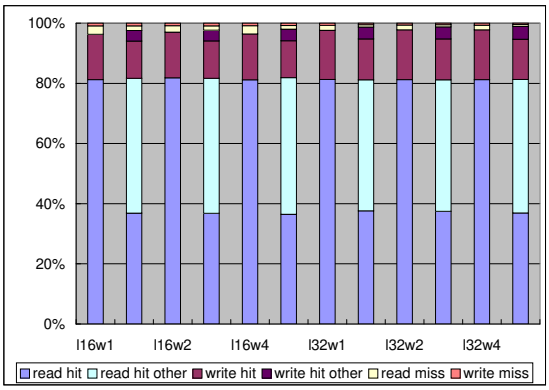

図 7 FFT のキャッシュヒット率
Fig. 7 cache hit ratio of FFT

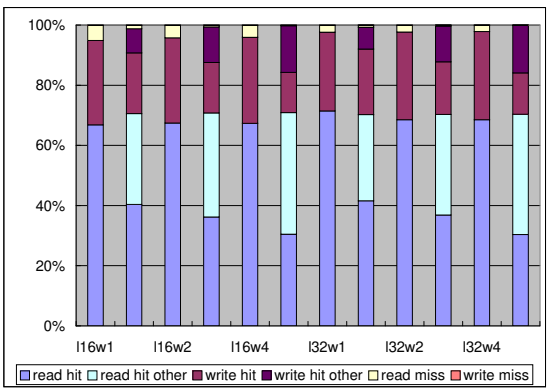

図 8 LU のキャッシュヒット率
Fig. 8 cache hit ratio of LU

5. 評 価

5.1 キャッシュヒット率

図 **7**，図 **8**，図 **9**，図 **10** に各アプリケーションでのキャッシュヒット率の結果を示す．それぞれの図で隣り合う 2 本のグラフが 1 セットとなっており，左が新

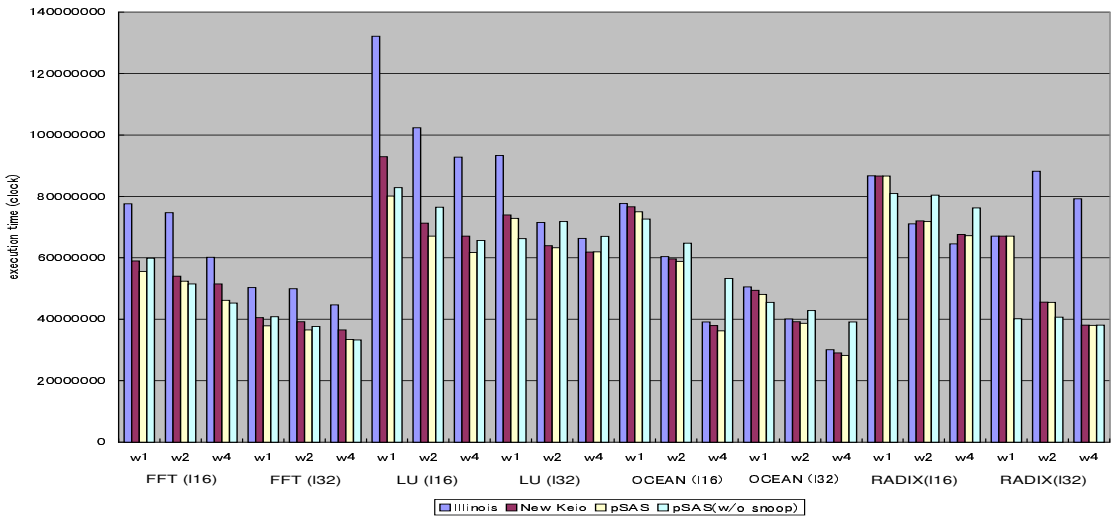

図 11 実行時間
Fig. 11 execution time

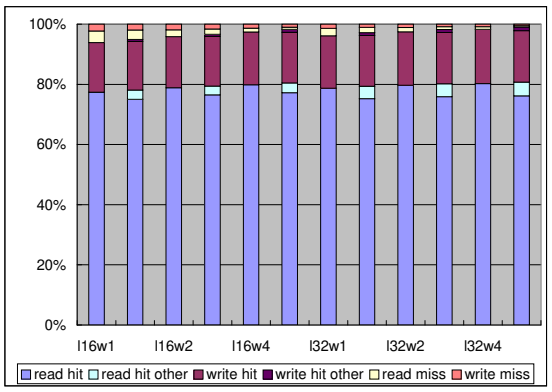

図 9 OCEAN のキャッシュヒット率
Fig. 9 cache hit ratio of OCEAN

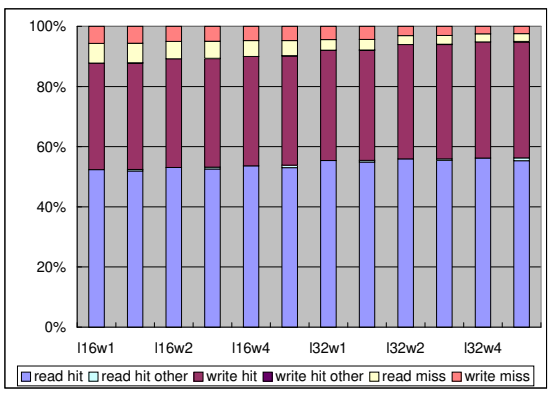

図 10 RADIX のキャッシュヒット率
Fig. 10 cache hit ratio of RADIX

Keio プロトコル，右が pSAS キャッシュである．この 2 本 1 組が同一条件での結果で，条件はラインサイズが 16，32 バイト，セット数が 1，2，4 ウェイの組合わせで 6 種類ある．

FFT（図 7）は 0.7 ~ 1.6%ヒット率が向上した．pSAS キャッシュではキャッシュヒットは半分以上が他のキャッシュへのヒットとなっており，アプリケーションの持つ共有データの相互参照度の高さが分かる．さらに詳細に検討するため，表 **1** と表 **2** にキャッシュラインサイズ 16byte，ダイレクトマップの場合の各プロセッサのキャッシュアクセスとメモリアクセスを示す．新 Keio プロトコルに比べて pSAS キャッシュは read cache miss が 49%減少しているが，それ以外はあまり変化がない．read from memory と write back の回数もほぼ変化がないことから，FFT ではキャッシュライン転送の減少によって性能が上がったことが分かる．

LU（図 8）ではさらに効果があり，2.0 ~ 4.0%ヒット率が向上した．表 **4** を見ると各プロセッサの read hit other と write hit other が多く，どのプロセッサも互いにデータを読み書きしていることが分かる．このようなアクセスパターンでは通常のスヌープキャッシュでは read と invalidate の繰り返しとなり，チップ上のキャッシュメモリの利用効率が落ちる．pSAS キャッシュではそれが改善され，表 **3** と比べると，メモリアクセスが 34.2%減った．

一方，OCEAN（図 9）と RADIX（図 10）は，他のキャッシュにほとんどヒットしていない．これは，OCEAN，RADIX 共にデータを分割して各プロセッ

表 1 FFT のメモリアクセスの詳細 (新 Keio l16w1)
Table 1 memory access of FFT (New Keio l16w1)

	PE1	PE2	PE3	PE4	合計
read hit	1289678	9970714	10329835	10330935	31921162
read miss	407197	408569	146821	146810	1109397
write hit	1774283	1380687	1384462	1384474	5923906
write miss	184668	52928	49160	49180	335936
read mem	207737	119553	195666	195686	718642
write mem	188778	21514	78949	78970	368211

表 2 FFT のメモリアクセスの詳細 (pSAS l16w1)
Table 2 memory access of FFT (pSAS l16w1)

	PE1	PE2	PE3	PE4	合計
read hit	1143916	9470002	1559513	1549236	13722667
read hit(o)	407808	798154	7728663	7736915	16671540
read miss	145114	127662	146518	146515	565809
write hit	1265522	579921	1384489	1384458	4614390
write hit(o)	512955	804552	30	13	1317550
write miss	180475	49143	49142	49142	327920
read mem	325590	1057	195663	195661	717971
write mem	152614	57052	78941	78934	367541

表 3 LU のメモリアクセスの詳細 (新 Keio l16w1)
Table 3 memory access of LU (New Keio l16w1)

	PE1	PE2	PE3	PE4
read hit	4823377	7369626	7523238	6208586
read miss	784520	224067	203880	737817
write hit	2771035	2783288	2787356	2540336
write miss	37104	135	63	4671
read mem	157704	215608	156426	83083
write mem	96689	109026	101233	70469

表 4 LU のメモリアクセスの詳細 (pSAS l16w1)
Table 4 memory access of LU (pSAS l16w1)

	PE1	PE2	PE3	PE4
read hit	2033806	5030077	3470292	5035825
read hit(o)	3677261	2075140	3857083	2051557
read miss	97814	61253	84501	178024
write hit	1452601	1712876	2350388	2267082
write hit(o)	1320614	832096	436993	516326
write miss	34977	30	0	5
read mem	132737	34313	84463	177480
write mem	63715	31157	40482	87721

サが分担しており，プロセッサ間でのデータの共有度が極めて低いためである．OCEAN ではヒット率が 1%向上，RADIX はほぼ変わらずで，このようなアプリケーションでは pSAS キャッシュは効果がないことがわかる．

5.2 実 行 速 度

図 11 に新 Keio プロトコルキャッシュ，pSAS キャッシュ，参考比較用の Illinois プロトコルキャッシュ，スヌープ機能なし pSAS キャッシュでの各アプリケーションの実行時間を示す．pSAS キャッシュは新 Keio プロトコルに対し，全体で 10%性能が向上している．特に LU では最大 16%性能が向上した．

チップ上に同一キャッシュラインの存在を許さない，スヌープ機能のない pSAS キャッシュ (pSAS(w/o snoop)) はその振る舞いが共有キャッシュに近くなり，スヌープキャッシュとの比較ではデータセットがキャッシュに載ったかどうかで結果がばらついた．一概に良いとは判断できず，共有キャッシュとの比較検討も必要であると言える．

5.3 トレードオフ

スヌープ機能付 pSAS キャッシュは従来のスヌープキャッシュプロトコルに以下のハードウェアを追加している．

- back door path と arbiter
- 各タグメモリに back door path 用比較器
- タグメモリの 3 ポート化

これらはいずれも膨大なハードウェア量ではなく，今回想定するチップ面積を考えれば十分実装可能である．前者 2 つの項目は自分のキャッシュをアクセスする場合のクリティカルパス上に存在しないため動作速度への影響が少ないと考えられるが，3 つ目のタグメモリの 3 ポート化はタグの読み出し遅延の増加が懸念される．

6. ま と め

本報告では他のスヌープキャッシュを自分のプロセッサの擬似セットとして扱うことにより、よりキャッシュメモリを半共有する pSAS キャッシュを提案し、評価

を行った。

ほぼ全ての条件で新 Keio プロトコルよりさらに性能が向上した．特にダイレクトマップ方式のキャッシュならば，疑似セットの効果が十分に現われ，最大 16%性能が向上した．また，pSAS キャッシュは LU のようなデータアクセスパターンが不均一なアプリケーションで特に効果が見られた．

今後の課題として参照頻度の高いキャッシュラインを自分のキャッシュへコピーすることを検討したいと考えている．

参 考 文 献

1) Kunle Olukotun, B. A. N., Hammond, L., Wilson, K. and Chang, K.: The Case for a Single-Chip Multiprocessor, *Architectural Support for Programming Languages and Operating Systems VII*, pp. 2–11 (1996).
2) Nayfeh, B. A., Hammond, L. and Olukotun, K.: Evaluation of Design Alternatives for a Multiprocessor Microprocessor, *International Conference on Parallel Precessing*, pp. 67–77 (1996).
3) Fillo, M., Keckler, S. W., Dally, W. J., Carter, N. P., Chang, A., Gurevish, Y. and Lee, W. S.: The M-Machine Multicomputer (1995).
4) Terasawa, T., Ogura, S., Inoue, K. and Amano, H.: A Cache Coherence Protocol for Multiprocessor Chip, *Proc. of IEEE International Conference on Wafer Scale Integration*, pp. 238–247 (1995).
5) 高橋真史, 高野祐之, 鈴木清吾, 田胡治之: オンチップマルチプロセッサのアーキテクチャの検討, 信学技報 CPSY, pp. 17–24 (1995).
6) Kisuki, T., Wakabayashi, M., Yamamoto, J., Inoue, K. and Amano, H.: Shared vs. Snoop:Evaluation of Cache Structure for Siglechip Multiprocessors, *EUROPAR97* (1997).
7) 寺澤 卓也: Z キャッシュ：オンチップマルチプロセッサ用キャッシュ, 情報処理学会論文誌, Vol. 37, No. 4, pp. 666–669 (1996).
8) Hennessy, J. L. and Patterson, D. A.: *COMPUTER ARCHITECTURE A QUANTITATIVE APPROACH SECOND EDITION*, Morgan Kaufmann Publishers (1996).
9) 井上 敬介, 若林 正樹, 木村 克行, 天野 英晴: シングルチップマルチプロセッサ用半共有型スヌープキャッシュ, 電子情報通信学会技術研究報告, CPSY98-58 (1998).
10) 若林 正樹, 米田 卓司, 天野 英晴: 並列計算機シミュレーションライブラリの提案, 電子情報通信学会技術研究報告, VLD97- (1997).
11) Woo, S. C., Torrie, M. O. E., Singh, J. P. and Gupta, A.: The SPLASH-2 Programs: Characterization and Methodological Considerations, *Proc. of 22nd Ann. International Symposium on Computer Architecutre*, pp. 24–36 (1995).

(平成？年？月？日受付)
(平成？年？月？日採録)

井上 敬介（学生会員）

平成 6 年慶應義塾大学理工学部電気工学科卒業．平成 8 年同大大学院理工学研究科計算機科学専攻修士課程修了．現在同大学院後期博士課程．計算機アーキテクチャの研究と並列計算機の実装に従事．

若林 正樹

平成 8 年慶應義塾大学理工学部電気工学科卒業．平成 10 年同大大学院理工学研究科計算機科学専攻修士課程修了．現在同大学院後期博士課程．計算機アーキテクチャの研究に従事．

木村 克行

平成 9 年慶應義塾大学理工学部電気工学科卒業．現在同大学院修士課程．並列計算機のキャッシュアーキテクチャの研究とデバッグに従事．

天野 英晴（正会員）

昭和 56 年慶應義塾大学工学部電気工学科卒業．昭和 61 年同大大学院工学研究科電気工学専攻博士課程修了．現在慶應義塾大学理工学部情報工学科助教授．工学博士．計算機アーキテクチャの研究に従事．