計算機アーキテクチャ 135－11
（１９９９．１１．２７）

オンチップマルチプロセッサのキャッシュメモリの構成

寺澤 卓也†

†東京工科大学 メディア学部
terasawa@media.teu.ac.jp

半導体技術の発達により，１チップに集積できるトランジスタ量は非常に多くなっている．このような状況を踏まえ，１チップ上に小規模なマルチプロセッサシステムを構築する研究が盛んに行なわれており，チップも試作されている．このようなオンチップマルチプロセッサでは，チップ内外の転送速度のギャップが大きいため，チップ外との通信を最小化する必要がある．本研究では，キャッシュアクセスに関するプロセッサのストールを抑え，チップ内のキャッシュを効率的に使用することを目指し，セットアソシアティブキャッシュの各 way に異なるプロセッサが順に並列にアクセスする Cyclic キャッシュを提案する．

Cyclic Cache: An Organization of Cache Memory System for On-chip Multiprocessors

Takuya Terasawa†

† School of Media Science, Tokyo University of Technology
terasawa@media.teu.ac.jp

Advances on VLSI technologies provide us a huge amount of transistors on a chip. To make the best use of these resources, many architectures that realize a multiprocessor on a chip are proposed and evaluated. Since these on-chip multiprocessors have a significant latency gap between on-chip and off-chip communications, accesses to off-chip shared memories must be minimized. In this paper, a new cache control scheme, Cyclic Cache is proposed to prevent processors from stalling on cache accesses, and to achieve a efficient use of on-chip cache memories. In Cyclic Cache, each processor accesses on independent way of a set associative cache. Processors change the way to access cyclically and synchronously.

1 はじめに

半導体技術の発達により近年のマイクロプロセッサの集積度は非常に高くなっている．このような高い集積度を生かし，1チップ内に高度な機能を内蔵するアーキテクチャが多数研究されている [1]～[10]．複数の汎用マイクロプロセッサを1チップに集積するオンチップマルチプロセッサ (もしくはマルチプロセッサチップ，チップマルチプロセッサ) はそのようなアーキテクチャの一つである．

筆者らは4～8程度のシンプルな汎用プロセッサをキャッシュメモリとバスを介して結合したオンチップマルチプロセッサのためのキャッシュプロトコルとして新Keioプロトコルを提案し検討してきた [1]．

オンチップマルチプロセッサではチップ内外の動作速度のギャップが大きいため，チップ外に配置した主記憶との間の通信を極力避ける必要がある．新Keioプロトコルでは，オーナシップに基づいてできるだけチップ内のキャッシュ間転送でミスに対処するようにプロトコルの工夫を行なっている．また，書き込み無効化型と書き込み更新型の混在を許すことにより，データの特性に合わせた制御を行なう．これによりプロトコルの工夫で対処できることについてはぎりぎりまでの改良がなされた．

更に，リプレイスの対象となったラインの再利用の可能性を考慮し，できるだけチップ内にとどめておく方法としてZキャッシュと呼ぶ機構を提案している [5]．これらにより性能は若干向上するが，さらなる性能向上を実現するには結合網やキャッシュメモリの構成方法を含めた検討を行なう必要がある．

本稿ではセットアソシアティブキャッシュの各wayには同時にアクセス可能なことを利用したキャッシュ制御手法を提案する．

2 オンチップマルチプロセッサ

オンチップマルチプロセッサは図1に示すように複数のプロセッサコアとキャッシュメモリ，相互結合網を1チップ上に集積するもので，一般に以下の特徴がある．

- プロセッサ間の結合網を高速・高バンド幅にすることができる
- チップ内外の速度ギャップが大きい
- 機構の統合，共通化の余地が生じる

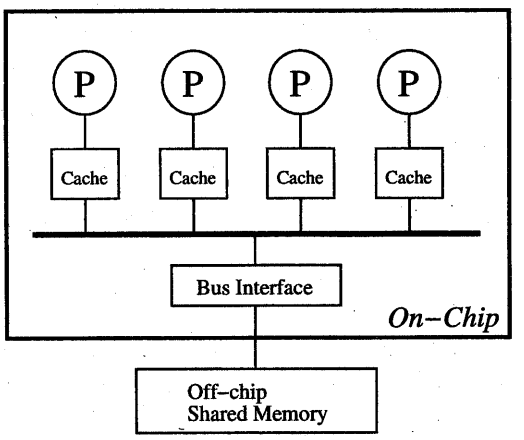

図 1: オンチップマルチプロセッサの構成例

このような特徴に基づき様々なアーキテクチャが提案されてきた．これらは大きく分けてプロセッサやキャッシュメモリの構成・制御に重点をおくものとマルチスレッド指向のものとに分けられる．

マルチスレッド指向のアーキテクチャ[8][9] ではスーパースカラプロセッサの利用できる命令レベル並列性 (ILP) の限界を破るため，複数のスレッドにまたがるILPを利用しようとするものである．

また，オンチップマルチプロセッサでは大量のトランジスタを使用することから，同程度のトランジスタ量を用いて，現行のスーパースカラプロセッサを発展させた場合と比較してどちらが高性能を達成できるかという検討も行なわれている [3]．

木透らが行なった共有キャッシュとスヌープキャッシュの比較 [7] では，マルチポートメモリを用いた共有キャッシュの性能について検討されている．共有キャッシュをマルチポートメモリで実現する場合，消費するチップ面積が増大するため，同じチップ面積ではキャッシュメモリを縮小せざるを得ず，また，1ポートではポートのコンフリクトが発生するため，必ずしも共有キャッシュの性能は高くないとしている．

2.1 キャッシュコヒーレンシ

共有キャッシュのアプローチをとらず，各プロセッサがプライベートキャッシュを持つ構成では，キャッシュコヒーレンシの問題に対処しなければならない．

オンチップマルチプロセッサではチップ内外のスピードのギャップが大きいため，キャッシュコヒーレンシについてはできるだけチップ内で解決する必要がある．図2に示す新Keioプロトコル[1]では，書き込み無効化型と書き込み更新型の混在を許し，キャッシュミスに際してはチップ内でキャッシュ間転送を行なうことによってチップ外の主記憶へのアクセスを極力減らしている．

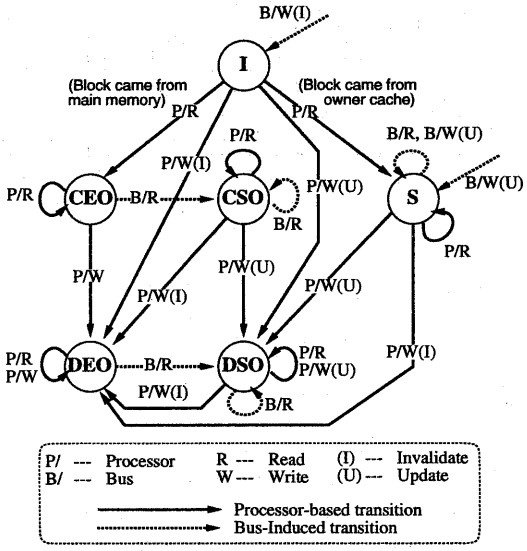

図 2: 新Keioプロトコルの状態遷移

様のタイミングでそのデータを利用できるというのが理想的である．しかし，全プロセッサがこれを同時に行なうことを考えると，その実現には理想的な多ポートメモリと膨大な配線が必要となり，現実的ではない．一方，シングルポートのキャッシュメモリではプロセッサ側のアクセスとバス側のアクセスが衝突し，いずれかが待たされることで性能が低下する．そこで，コストおよび実現可能性と性能のトレードオフを考慮してこれらの間に位置する手法を考案する必要がある．

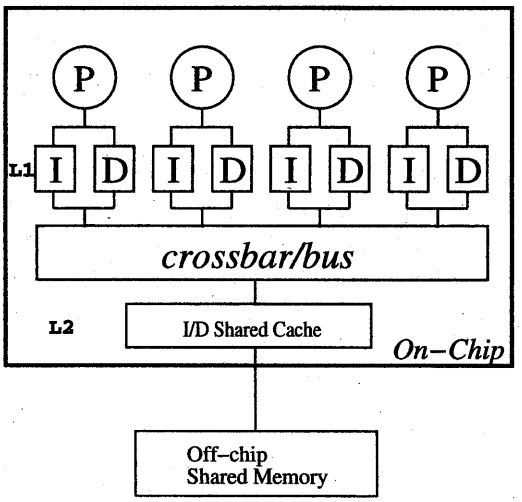

図 3: 提案するオンチップマルチプロセッサの構成

3 Cyclic キャッシュ

プロトコルの改善だけでは性能の向上は限定されるため，さらなる性能向上をはかるにはオンチップに集積されることのメリットを積極的に利用する必要がある．具体的には，オンチップマルチプロセッサでは少ないコストで他のプロセッサのキャッシュにアクセス可能なことを利用する．このアプローチとしてはキャッシュの制御機構を集中化する方式[2]や，他のプロセッサのキャッシュを自らのキャッシュの疑似的なwayとして扱うVLNWキャッシュ[10]などがある．

他のプロセッサのキャッシュにアクセスする場合，あるプロセッサがアクセスを行なう際に他のプロセッサのキャッシュにも同時にアクセスし，いずれかにヒットすれば自らのキャッシュにヒットしたのと同

3.1 Cyclic キャッシュの機構

本研究では，セットアソシアティブキャッシュの各wayは独立にアクセス可能なことに注目し，タグを2ポート化することで効率的に複数のプロセッサのキャッシュにアクセスする手法を提案する．図3にこの構成を示す．なお，同期のための機構はキャッシュとは別に存在すると仮定する．また，各コンポーネントについては以下のような構成を想定する．

- プロセッサ：トランジスタ数と高速動作を考え，MIPS R3000などの比較的シンプルなプロセッサコアで4程度
- 1次キャッシュ：プロセッサ毎に命令・データ各32KB程度，各4wayセットアソシアティブ，ラインサイズ32byte程度

- 2次キャッシュ：プロセッサ間共有・命令データ共通で1MB程度，4wayセットアソシアティブ，ラインサイズは1次キャッシュと同じ

- 結合網：タグアクセス用クロスバおよびデータ転送用の2本の高バンド幅のバス

近年のマイクロプロセッサではパイプラインを理想的に動作させるために1次キャッシュは重要な役割を果たしている．このような1次キャッシュに対して従来のキャッシュコヒーレンシプロトコルに基づくキャッシュ間転送を行なうと，バス側からのキャッシュアクセスによりプロセッサ側のアクセスが阻害され，性能の低下を招く．そこで，次のような拡張を行なう．

- 1次データキャッシュのタグをマルチポート化し，プロセッサ側に影響を与えずにバス側からもアクセスできるようにする．しかし，マルチポート化のコストとアクセス速度を考慮し，キャッシュのデータメモリ部分のマルチポート化は行なわない．また，1次データキャッシュはライトスルーとする．

- プロセッサから見て1次キャッシュの外側にライトスルーバッファを設ける．これは機能を限定したキャッシュメモリとみなすことができ，1次キャッシュからライトスルーで書き込まれるラインと書き込み更新で他のプロセッサから送られてくるラインを保持する．このバッファの先の2次キャッシュにはライトスルーしない．このバッファのタグは1次データキャッシュと共用する．他のプロセッサからdirtyラインの要求があった場合はこのバッファから転送することで，1次データキャッシュの動作を妨げない．

ここで，cleanとは2次キャッシュのラインと同一内容であること，dirtyとは2次キャッシュのラインと内容が異なることを表す．

単純に2次キャッシュを個別に持つアプローチも考えられるが，それではチップ中に同一ラインのコピーが増え，効率が悪い．しかし，1次キャッシュにコピーがあることはアクセス速度の点から意義がある．

次に，他のプロセッサのキャッシュへ高速にアクセスするため，次のようなパイプライン機構を導入する．このため，図4に示すように，各プロセッサのキャッシュタグを他のキャッシュからも読み出し可能にするため，クロスバスイッチを用いて結合する．

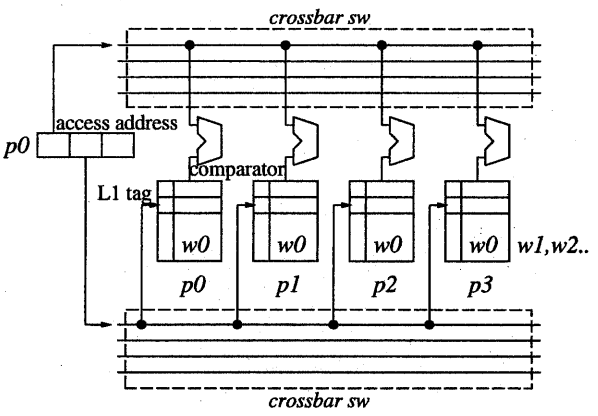

図4: 他のプロセッサのキャッシュタグへのアクセス

3.2 readの場合

今，4つのプロセッサをp0～p3，各プロセッサの1次キャッシュの4つのwayをw0～w3と呼ぶことにする．p0のreadアクセスを例に，この機構を説明する．なお，リプレイスについては後述する．

1. p0はデータをreadする際には通常の4wayセットアソシアティブキャッシュとして自分の1次キャッシュの4つのwayすべてに同時にアクセスすると同時に，他の3つのプロセッサの1次キャッシュおよび2次キャッシュのw0のタグにもクロスバを通してアクセスする．

2. 自分の1次キャッシュにヒットした場合は通常のキャッシュヒットとしてアクセスを行なう．自分の1次キャッシュにミスした場合はライトスルーバッファをチェックするとともに，1.に書いたように他の3つのプロセッサの1次キャッシュと2次キャッシュのw0のタグにアクセスし，ヒットしていないかチェックする(実際にはこれらはp0のキャッシュのヒット判定と同時)．

3. ヒット判定の結果は次のパタンが考えられる

    - いずれかもしくは複数の1次キャッシュのw0のdirtyラインにヒットする

- 2次キャッシュのw0にヒットする
- 以上の両方が同時に生じる

これらの場合は2本あるライン転送用バスのどちらか都合の良い方を利用してラインをp0の1次キャッシュに転送する．なお，1次キャッシュ(dirty)と2次キャッシュの両方にヒットした場合は1次キャッシュから転送する．この際，1次キャッシュからのライン転送はライトスルーバッファから行なう．また，いずれかもしくは複数の1次キャッシュのw0のcleanラインにヒットするケースも考えられるが共有2次キャッシュは十分に大きく，各プロセッサの1次キャッシュのスーパーセットになると仮定してcleanな1次キャッシュからの転送は行なわない．

いずれのw0にもヒットしなかった場合は次のクロックでw1に，その次のクロックでw2にというように順にアクセスし，最初にヒットしたラインから上記の方法でラインを転送する．これにより，他のプロセッサの1次キャッシュ中(dirty)もしくは共有2次キャッシュにラインが存在する時は最悪でも最後のw3へのアクセスでいずれかのキャッシュにヒットする．

この機構を用いると，プロセッサによってスタートのwayを変えておくことによって全プロセッサ同時にでもヒット判定が行なえる．バススヌープの方式に比べ，バスマスタになるアービトレーションの時間が不要であり，高速にヒット判定とライン転送が行なえる．ライン転送は2本あるバスの空いている方を使用し，共に使用中の場合はキューに入り，先に解放された方を使用する．

3.3 writeの場合

readと同様にp0の1次キャッシュのすべてのwayと他のプロセッサの1次キャッシュのw0のタグにアクセスする．

1. p0の1次キャッシュにヒットした場合はそれに書き込むと同時にライトスルーバッファにも書き込む．この後，プロセッサは先の処理に進む．本機構では書き込み更新，書き込み無効化の両方に対応する．ここでは，書き込み更新の場合について説明する．他のプロセッサの1次キャッシュにもヒットした場合は以下のように更新をを行なう．
   - w0のいずれかにヒットした場合は2本のバスのうち利用可能な方を使用して変更されたデータを転送する．転送を1度で済ませるため，p0以外のすべてのプロセッサはバス上のデータをアップデートバッファに取り込む．このため，p0はw0のタグにアクセスする際に書き込み更新であることをあらかじめ他のキャッシュに伝える．この作業は最初にヒットしたwayで行なう．
   - その後，p0はw1, w2,... と他のプロセッサの1次キャッシュのwayのタグに順にアクセスを行なう．ヒットした場合はアップデートバッファに取り込んでおいたデータをライトスルーバッファに取り込むようにそのプロセッサのキャッシュコントローラに指示する．コントローラは1次キャッシュの該当するラインのタグのupdateフラグを立て，次回そのプロセッサがそのラインにアクセスした時にライトスルーバッファから更新部分のデータが供給されるようにする．この制御のため，1次キャッシュのタグに若干のフラグを追加する．
2. p0の1次キャッシュにミスした場合はreadミスの場合とwriteヒットの場合を同時に行なう．すなわち，readミスの場合と同様の方法でラインを探し，見つかった場合，p0に転送するのと同時にp0が書き込んだデータを他のプロセッサのキャッシュに送る．ただし，readの場合と異なり，他のプロセッサの1次キャッシュからの転送を優先する．これは他のプロセッサがそのラインにwriteを行なっている可能性があり，それが反映された状態のラインを得るためである．なお，p0は自分の1次キャッシュにミスしてもライトバッファにデータを書き込んで先に進むことができる．ラインが得られた後にライトバッファのデータを反映させる．

3.4 リプレイス

リプレイスによってdirtyラインが1次キャッシュから2次キャッシュに追い出される場合は，writeヒッ

トなどと同様に順に他のプロセッサの1次キャッシュのwayにアクセスする．そして，ヒットした場合は，当該ラインのdirtyビットをクリアする．追い出されるラインは最初のアクセスの時にバスを用いて2次キャッシュのバッファに転送される．やがて，2次キャッシュのwayにヒットした場合はバッファからラインが書き込まれる．もし，ヒットしなかった場合は2次キャッシュの適正な場所にラインを書き込む．

4 まとめ

本稿ではCyclicキャッシュ機構を提案した．この機構ではセットアソシアティブキャッシュの各wayは並列にアクセスできることを利用して，複数のプロセッサが同時に他のプロセッサのキャッシュのwayにアクセスすることができる．アクセスするwayをプロセッサ毎に異なったものとし，これらを順に変更していくことにより，アクセスをオーバラップさせることができ，効率の良いヒット判定とライン転送が実現できる．

本稿ではCyclicキャッシュの提案と概要を説明するに留まった．今後，以下の点について更に検討し，現実的な性能評価を行なっていく予定である．

- キャッシュプロトコルの詳細の検証
  本稿ではread/writeの場合のCyclicキャッシュの動作について概要を述べた．今後は様々な場合を想定した細部の検証を進めるとともにチューニングを行なう必要がある．

- 配線可能性の検証
  キャッシュタグ間のクロスバスイッチ結合やライン転送用の高バンド幅のバスの配線が現実的に可能かどうか検証する．

- チップサイズの見積り
  実際にCyclicキャッシュによるマルチプロセッサをオンチップに実装する場合，どの程度のチップ面積となるかをCAD設計等によって見積もる．

5 謝辞

本研究の一部は文部省科学研究費補助金 (奨励研究 (A) 課題番号 10780206) による．

参考文献

[1] 寺澤, 井上, 黒澤, 天野, “オンチップマルチプロセッサのキャッシュメモリの検討”, 信学技報, CPSY95-17, pp.47-54, Apr. 1995.

[2] 高橋, 高野, 鈴木, 田胡, “オンチップマルチプロセッサのアーキテクチャ検討”, 信学技報, CPSY95-3, pp.17-24, Apr. 1995.

[3] K. Olukotun, B. A. Nayfeh, L. Hammond, K. Wilson, K. Chang, “Th Case for a Single-Chip Multiprocessor,” Proceedings of ASPLOS VII, pp.2-11, 1996.

[4] L. Hammond, B. A. Nayfeh, L. Olukotun, “A Single-Chip Multiprocessor,” COMPUTER, Vol.30, No.9, pp.79-85, 1997.

[5] 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, Apr. 1996.

[6] 宮嶋, 岩下, 村上, “高性能システム・オン・チップ構成法に関する性能評価” 信学技報, HPC62-6, pp.33-38, Aug. 1996.

[7] T. Kisuki, M. Wakabayashi, J. Yamamoto, K. Inoue, H. Amano, “Shared vs. Snoop: Evaluation of Cache Structure for Single-chip Multiprocessors,” Proceedings of EUROPAR97, 1997.

[8] 鳥居, 近藤, 本村, 西, 小長谷, “オンチップ制御並列プロセッサ MUSCAT の提案”, 情報処理学会論文誌, Vol.39, No.6, pp.1622-1631, Jun. 1998.

[9] 小林, 岩田, 安藤, 島田, “非数値計算プログラムのスレッド間命令レベル並列を利用するプロセッサ・アーキテクチャSKY”, JSPP'98 論文集, pp87-94, Jun. 1998.

[10] 井上, 若林, 木村, 天野, “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”, 信学技報, CPSY98-58, pp.75-81, Aug. 1998.