計算機アーキテクチャ 123－7
（1997. 3. 7）

マルチスレッド型プロセッサ向きのキャッシュメモリの構成と評価

相原孝一　伊藤英治　丹 康雄　日比野 靖

北陸先端科学技術大学院大学 情報科学研究科
〒923-12 石川県 能美郡 辰口町 旭台1-1

マルチスレッド型パイプラインプロセッサの下で、その効果を有効に発揮させるべくキャッシュ機構のパイプライン化を提案する。そして、マルチスレッド型プロセッサのキャッシュの問題であるヒット率に関し、スレッド数に見合うキャッシュ容量を与えればヒット率は十分に維持できることを示す。さらに、極めて高いスループットのキャッシュからレイテンシの大きな主メモリへのアクセス要求に対する、キャッシュ-主メモリインタフェースの構成について調べ、キャッシュからのメモリ要求頻度に見合う主メモリスループットを与えることで、システム全体としてスループットの低下が低く抑えられることを示す。

Organization and Evaluation of Cache Memory for Multithreaded Processor

Kouichi AIHARA　Eiji ITOH　Yasuo TAN　Yasushi HIBINO

School of Information Science,
Japan Advanced Institute of Science and Technology,
1-1,Asahidai,Nomi-gun,Ishikawa,923-12 Japan

Email Address: aihara@jaist.ac.jp e-itou@jaist.ac.jp
ytan@jaist.ac.jp hibino@jaist.ac.jp

Abstruct

A Pipelined cache mechanism for a multithreaded pipeline processor is proposed. The cache having capacity balanced with the number of threads shows high hit rate. And, a cache main-memory interface between a very high throughput cache and a large latency main-memory system is investigated. Providing main-memory throughput in proportion to the frequency of memory requests from the cache memory, it is shown that the system holds the required throughput.

1 はじめに

MOS デバイスは、その物理寸法を縮小することによって、動作速度が高速になる性質を持つ。近年の LSI 製造技術の進歩によって、MOS デバイスの微細化が進み、素子の高速な動作速度が実現されている。しかし、物理寸法を縮小しても配線遅延は減少しない。この結果、動作クロックを上げることによって、プロセッサの高性能化を実現することが困難な状態になっている [3]。この点を解消する手段として、プロセッサのパイプライン・ステージを細分化することによって配線遅延を減少させ、動作クロックの高速化を図る方法が考えられる [7] [9]。しかし、パイプライン・ステージを細分化すると、単一ストリームから命令を発行する限り、命令間の依存関係によって、パイプラインの持つ性能を引き出すことができないという問題が生じる。

この問題を解決する方法の 1 つとして、マルチスレッド型プロセッサがある [5] [6] [8]。マルチスレッド型プロセッサは、単一のプロセッサで複数の命令ストリーム (スレッド) を実行するプロセッサであり、原理的にハザードの問題は解決できる。

プロセッサの高度なパイプライン化が進むにつれ、プロセッサのクロックサイクルおよびスループットはキャッシュ機構に依存することになる。また複数の命令ストリームを扱わなければならないので、キャッシュの容量もそれに見合った容量を用意しなければならない。そうなるとキャッシュの大容量化にともなって配線遅延によるキャッシュメモリのアクセス速度も低下することになる。

そのため本稿ではキャッシュでのスループットを稼ぐべくキャッシュ機構のパイプライン化と、メモリアクセス速度の向上を図るためにキャッシュのメモリアレイの分割を提案する。

次にマルチスレッド型プロセッサ上でのキャッシュのヒット率、およびスループット問題について検討し、マルチスレッド型プロセッサに適したメモリの構成を行い、シミュレーションにより評価を行う。

2 キャッシュのパイプライン化

マルチスレッド型プロセッサでは、キャッシュメモリのアクセス時間ではなくスループットが重要である [1]。プロセッサの高いスループットを実現するためのキャッシュメモリのパイプライン化を提案する。

キャッシュアクセスは次のような手順で行われる。

1. アドレスデコード
2. メモリセル・アレイの読みだし

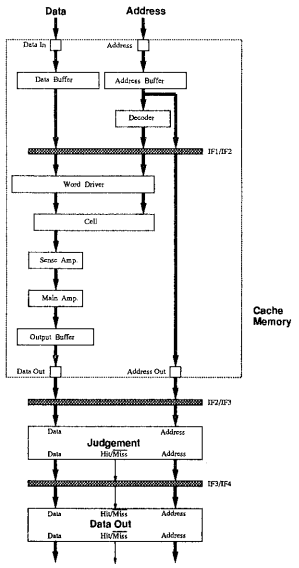

図 1: 4 ステージパイプライン化キャッシュ

3. 判定
4. データ排出

これらの各操作を分割することによって、図 1 に示すような 4 ステージから構成されるパイプライン化キャッシュを実現することができる。

この中のデコーダからアンプに至る部分がクリティカルパスとなっている [4]。ここで注目すべき点はデコーダ部とメモリセル部である。デコーダ部においてはデコーダ部で行う処理を数段の処理に分割しパイプライン化を適用することができる。これによりデコーダ部でのスループットの向上をねらう。

2.1 メモリセルアレイの分割

メモリの大容量化が進むにつれメモリセルアレイの配線遅延が問題となってくる。ここで、集積回路の性質であるスケーリング則に基づき配線遅延について検討する。スケーリング係数を K とする。配線遅延時間は、容量と線抵抗の CR 積となるためスケーリングにかかわらず 1 である。メモリのアクセスはデコードを行った後メモリセルアレイのアクセスを行う。デコード時間はゲートの遅延時間に支配されるので 1/K になる。一方、メモリセルアレイのアクセス時間は配線遅延時間に支配されるので、スケーリング比 K を大きくしてもアクセス時間の短縮は期待できない。そこでメモリセルアレイを分割し、各ブロックにデコーダを配置し階層的にデコードを行なう。メモリセルアレイを N 分割すると線抵抗が

表 1: パイプラインステージ間の論理段数

	Global Row Decoder	Sub-Global Row Decoder	Local Row Decoder	判定	データ排出
段数	7	6	6	7	4

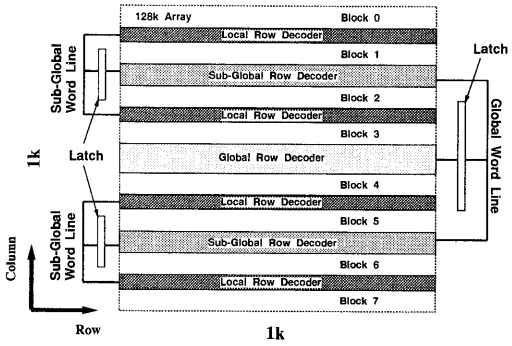

図 2: メモリセルアレイの分割によるアドレスデコードのパイプライン化

$1/\sqrt{N}$、容量が $1/\sqrt{N}$ となるので、配線遅延時間は 1/N に短縮することになる。またデコードの部分においてはデコードとメモリセルのブロック選択が並列に行えるため、両者により大容量化したキャッシュへの高速なメモリアクセスが期待できる。

2.2 階層的デコード法

プロセッサのスーパーパイプライン化によりクロックサイクルがキャッシュの操作に依存することになる。キャッシュ中の操作でも特に遅いものにデコーダがある。そこで、デコーダを数段の処理に分割しパイプライン化することによって階層的にデコードを行うことを検討する [2]。具体的に検討するために、1Mb(1k×1k) のメモリセルアレイを 8 つのブロックに分割し語線 (行選択線) を 3 つのレベルに分けて行い、各レベルの論理段数を見積りパイプラインとの関係を検討した。図 2 にメモリセルアレイの分割によるアドレスデコードのパイプライン化を示し、表 1 にパイプラインステージ間の論理段数を示す。

3 メモリシステム構成

3.1 キャッシュのマッピング方式

3.1.1 スレッド専用キャッシュとスレッド共用キャッシュ

マルチスレッド型プロセッサからはキャッシュへのアクセスが各クロックサイクル毎に行われる。このため、各スレッド間でキャッシュブロックの競合が起こるのではないかという問題が懸念される。

スレッド毎にキャッシュを用意する場合各スレッドのマッピングは自身に用意されたキャッシュ内でのみ許されるが、キャッシュを共用する場合そのような制約がないので、独自のキャッシュではブロック数が足りない場合でもキャッシュを共用することでその部分がカバーできる可能性がある。

3.1.2 ダイレクトマップとセットアソシアティブ

しかし、各スレッド間でのブロックの競合も起こり得る。この場合キャッシュのマッピング方式をセットアソシアティブ化することによって回避することが考えられる。

マルチスレッドを扱うキャッシュのマッピング方式においてセットアソシアティブを採用するとした場合、連想度をどの程度にするのが適当かを調べる必要がある。

3.2 ミスペナルティ

ミスペナルティは、メインメモリへのアクセス時間、また、後述するメインメモリアクセス要求待ち列での待ち時間、さらに、キャッシュの更新時間の和である。以下、これら各々について考察する。

3.2.1 メモリ要求待ち列

マルチスレッド型プロセッサでのキャッシュへのアクセス権はクロックサイクル毎に異なるスレッドに切替えられるため、1 つのスレッドがキャッシュミスを起こし、そのスレッドの実行を停止しても、次のクロックサイクルで次のキャッシュへのアクセス権をもつスレッドがキャッシュにアクセスする。このときこのスレッドが引続きキャッシュミスを起こす可能性もある。データキャッシュも同様に考えることができる。

メインメモリへのアクセスは 3.2.2 に述べるようにキャッシュミスの時のみ起こるのではない。

しかも、プロセッサとメインメモリの速度差は非常に大きい。メインメモリのスループットよりもその時間あたりにキャッシュから発生するメインメモリへの参照数が大きければメインメモリへのアクセスに待ち列が発生することになる。待ち列の長さは下記のメモリサイクル利用率に依存する。この値は 1 より十分小さくなければならない。

メモリサイクル利用率=
メインメモリアクセスサイクル時間
× メインメモリアクセス確率 (1)

また、待ち列の長さの短縮には、メインメモリをマルチユニット化し、ユニット毎に要求待ち列を設けることが有効である。この方策のねらいは、メモリバンク毎のメモリサイクル利用率を低下させることである。

3.2.2 メインメモリ要求の削減

キャッシュへの書き込み方式で、ライトスルー方式の場合はライトが発生するたびにメインメモリへのアクセスが行われる。これに対してライトバック方式の場合、ライトはキャッシュのブロックに対してだけ行われる。データが変更されたキャッシュ中のブロックは、置き換えの対象になった時にメインメモリへライトバックされる。

メインメモリへのトラフィックを減少させることは待ち列の長さを抑えることができるため、プロセッサのスループット向上にそのままつながる。このため、マルチスレッド型プロセッサでのキャッシュの書き込み方式にはライトバック方式が適切であると考えられる。

3.2.3 ブロック更新ペナルティの最小化

メインメモリからデータがあがってきた時のキャッシュへの書き込みに対してどのような方式をとるかによってもプロセッサのスループットに大きな影響を及ぼす。この解決策をいくつか検討してみる。

A. キャッシュの完全なマルチポート化

この方式では各々専用のポートが用意されているため、あるスレッドがキャッシュへアクセスを行っている時でも別のスレッドのブロックを更新することができる。しかし、ハード的に実現するには2つのポートのために専用のデコーダとビット線を用意しなければならなくコストが高くつく。

B. キャッシュのマルチバンク化

キャッシュへのアクセス権は、プロセッサと主メモリとの間でクロスバースイッチにより選択される。構造的には簡単だが、バンク競合が起こり得る。

C. 各スレッドに対応したバッファの設置

更新されるブロックデータがメインメモリからあがってきたら、それに該当するスレッドのバッファに書き込む。ミスを起こしたスレッドはキャッシュにアクセスせず、自分のバッファに命令/データを読みに行く。その時キャッシュステージにはスレッドのアクセスが起きないので、キャッシュへの書き込みを行うことができる。この手法はコスト的に見ても、またブロックの更新によるペナルティを解消する点でももっとも好ましい。

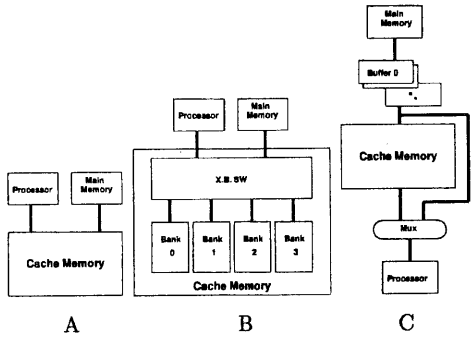

図 3: キャッシュの各更新方法

4 シミュレーション

3章で述べた方策の有効性を示すために、キャッシュのヒット率、およびスループットについてシミュレーション実験を行なった。

実験で用いたトレースデータは、アドレス生成器により作成したものを使用している。扱うスレッド数は16とする。

キャッシュの総容量は64kワード(1スレッドあたり4kワード)とした。なお、キャッシュの容量はマッピングの方式にかかわらず一定とする。

実験で示されているスループットは、実行されたクロックサイクルに対して、どのくらいの比率でデータが処理されたかを表すものである。よって、このスループットがシステム本来の性能を計る尺度になる。

4.1 連想度とキャッシュヒット率

まず、スレッド専用キャッシュとスレッド共用キャッシュとのヒット率の比較を行なった(表2)。

次に、キャッシュの連想度をあげることによって、キャッシュミス率がどのように変化するかシミュレーションを行なった(図4)。

4.2 メモリアクセスペナルティとスループット

書き込み方式の違いによってスループットがどのように影響してくるか、メモリアクセスペナルティ

表 2: 専用キャッシュと共用キャッシュ

	専用	共用			
		1-way	2-way	4-way	8-way
命令	0.997	0.996	0.998	0.998	0.998
データ	0.895	0.939	0.971	0.988	0.990

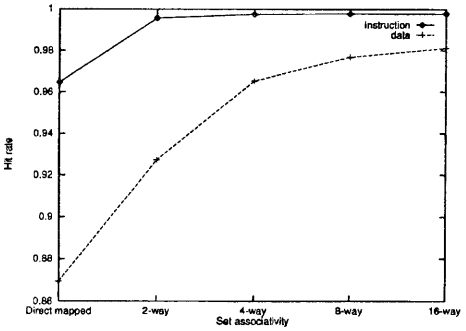

図 4: 連想度に対するキャッシュヒット率

を変化させてシミュレーションを行なった (図 5)。

また、メモリユニットを増やした場合のスループットの変化を、メモリアクセスペナルティをパラメータとしてシミュレーションを行なった (図 6)。

5 考察

5.1 キャッシュ機構のパイプライン化

キャッシュ機構のパイプライン化はレイテンシを増加させるが、クロックサイクルの短縮が実現され、キャッシュのスループットを稼ぐことができる。

またキャッシュメモリアレイを分割して物理寸法を小さくし配線遅延を削減することによって、パイプラインピッチをつめると同時に、キャッシュの大容量化も可能になると考えられる。大容量化はヒット率の向上も期待できる。ヒット率の向上はキャッシュミスペナルティの低下にそのままつながり、システム全体のスループットの向上につながる。

シミュレーションでも示されたようにメモリの書き込み方式をライトバック方式とすることでメモリトラフィックを低く抑えることができるが、問題点はキャッシュ機構が複雑化することでありレイテンシが増加する。しかし、キャッシュ機構のパイプライン化をさらにすすめることにより、ライトバック方式を採用してもキャッシュのスループットの低下にはつながらない。

このため、プロセッサ全体のスループットを向上させる上でキャッシュ機構のパイプライン化は大きな役割を果たしているといえる。

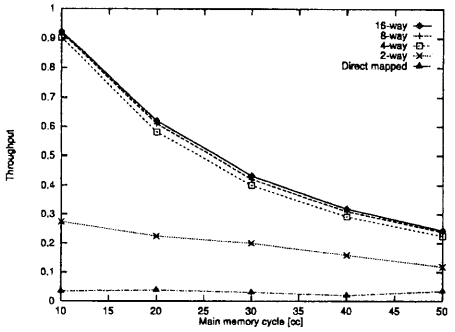

図 5: メモリアクセスペナルティに対するスループット (ライトスルー方式)

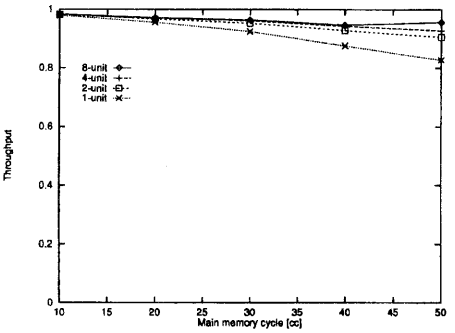

図 6: メモリユニット数とスループット (ライトバック方式)

5.2 キャッシュのヒット率

本報告で仮定しているプロセッサは 16 のスレッドを処理することとしているので、キャッシュ容量も 1 つのスレッドが要求する容量の 16 倍のものを用意した。

結果からも分かるようにスレッド間どうしでのブロックの競合は影響していないどころかキャッシュを各スレッドが共用する場合、専用キャッシュと比べヒット率が向上している。このことから、キャッシュをスレッド間で共用することによって、独立でキャッシュをもつ場合の総和に対して小さな容量で済む可能性を示している。

図 4 を見ても明らかなように、キャッシュの連想度を増やすことにより、キャッシュのヒット率を増加させる効果がある。ダイレクトマップでヒット率が低いのは、スレッド間での競合が起こっているためである。連想度を持たせることにより、競合する

ブロックを同じラインにマッピングすることができる。しかし、この結果からも分かるように連想度が増加するにつれヒット率の上昇は見られなくなる。

5.3 キャッシュのスループット

5.3.1 メモリサイクル時間

メモリアクセスペナルティが増大するにつれてスループットが急激に落ち込んでくる。これは、3.1式に示した待ち列の発生条件がメモリアクセスペナルティが大きくなると成立するためである。

5.3.2 書き込み方式

キャッシュの書き込み方式についてライトバック方式を採用することによりメモリアクセスペナルティが大きくなった場合のスループットの低下がそれほど見られなくなった。これはキャッシュにある程度プログラムがマッピングされるとメモリアクセスはストア命令の生起が支配的となる。ライトスルーの場合、ストア命令が起きるたびにメモリへアクセスしているのでメモリ要求待ち列が発生しやすい状況になる。ストア命令の頻度が5%から10%程度で発生することを考えるとキャッシュの書き込み方式がスループットに与える影響はかなり大きなものになるということがいえる。

5.3.3 メモリのマルチユニット化

メモリをマルチユニット構成とし、ユニット毎に待ち列をつくることにより、待ち列長を短縮でき、スループットの低下を防ぐのに役立つことが確かめられた。ユニット数を8とすれば、メモリサイクル時間がクロックの50倍でもスループットの低下はわずかである。

6 むすび

本稿ではマルチスレッド型プロセッサ向きのキャッシュ機構のパイプライン化を提案した。キャッシュ機構のパイプライン化はレイテンシを増加させるが、クロックサイクルの短縮によりスループットを向上させることができる。キャッシュメモリアレイを分割して物理寸法を小さくし配線遅延を減少させることによって、キャッシュのパイプラインピッチをさらにつめることができる。また、キャッシュのパイプライン化は複雑な書き込み方式の実現も可能にする。

本稿でのキャッシュメモリの構成、実験を通じて、キャッシュのヒット率のみで正しい性能の評価はできないということができる。キャッシュのヒット率がダイレクトマップ方式でかなり高い結果を得ていたのにもかかわらず、スループットで高い性能を得ることができなかったことからも明らかである。

マルチスレッド型プロセッサの高性能化のためにはメモリ要求待ち列の長さを抑えることが重要であり、それにはメインメモリの性能が重要であるということがいえる。しかし、メインメモリの速度向上の比はプロセッサの速度向上の比には追い付けていない。そのため、メインメモリのマルチユニット化などによりメモリのバンド幅を広げることが全体のシステムの性能を向上させる鍵となる。

参考文献

[1] 伊藤英治, 相原孝一, 丹 康雄, 日比野 靖, "関数型プログラムの実行に適したマルチスレッド型プロセッサ・アーキテクチャの提案", 情報処理学会研究会報告, 96-ARC-121, Vol.96, No.121, pp.81-88, 1997

[2] Toshihiko Hirose, et al., "A 20ns 4Mb CMOS SRAM with Hierarchical Word Decording Architecture", *ISSCC DIGEST OF TECHNICAL PAPERS*, pp.132-133, 1990.

[3] 井口秀之, "次世代集積技術による RISC アーキテクチャプロセッサの設計と評価", 北陸先端科学技術大学院大学修士論文, 1995

[4] Koichiro Ishibashi, et al., "A 300MHz 4-Mb Wave-Pipeline CMOS SRAM using a Multi-Phase PLL", *ISSC DIGEST OF TECHNICAL PAPERS*, pp.308-309, 1995.

[5] Daniel C. McCrackin, "Eliminating Interlocks in deeply Pipelined Processor by DelayEnforced Multistreaming", *IEEE Trans. on Computer*, Vol.40, No.10, pp.1125-1132, Oct 1991.

[6] Won Woo Park, Donald S. Fusell and Roy M. Jenevein, "Performance Advantages of Multithreaded Processors", sl Proc. of the Third Int'l Conf. on Parallel Processing, Vol.1, pp.97-101, 1991.

[7] David A. Patterson and John L. Hennecy, "COMPUTER ORGANIZATION & DESIGN THE HARDWARE/SOFTWARE INTERFACE", Morgan Kaufmann Publishers Inc, 1994.

[8] R. Guru Prasadh and Chuan-lin Wu, "A Benchmark Evaluation of a Multi-threaded RISC Processor Architecture", sl Proc. of the 20th Int'l Conf. on Parallel Processing, Vol.1 pp.84-91, 1991.

[9] MIPS Technology Inc. "R4400 MICROPROCESSOR PRODUCT INFORMATION", 1996.