社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2006－ARC－167（4）
2006－HPC－105（4）
2006／2／27

配線遅延を考慮したキャッシュメモリ高速化手法

蛭田　智則†　増田　渓介††*　山名 早人†††
†早稲田大学大学院理工学研究科　††早稲田大学理工学部　†††早稲田大学理工学術院

概要　マイクロプロセッサとメモリとの速度ギャップの増大により、高速なキャッシュメモリの必要性が増している。しかし、近年のプロセスの微細化に伴い、マイクロプロセッサに搭載されるキャッシュメモリ量は増加し、キャッシュメモリにおける配線遅延が増加している。そのため、今後キャッシュメモリアクセスがシステムのボトルネックになる可能性がある。そこで、本稿では、配線遅延を考慮したキャッシュメモリ高速化手法を提案する。本手法を SimpleScalar 3.0d に実装し、SPEC95 CINT、SPEC2000 CINT を用いて評価した結果、平均 1.17 倍の IPC 向上が得られた。

Optimization Technique for Cache Memory Considering Wire Delay

Tomonori HIRUTA†　Keisuke MASUDA††　Hayato YAMANA†††
†Graduate School of Science and Engineering, Waseda University
††Science and Engineering, Waseda University
†††Science and Engineering, Waseda University

Abstract The increase of the gap between processor speed and memory speed makes cache memory more important. However, wire delay in large cache grows that results from the process miniaturization. Therefore, cache memory access will become bottle neck. This paper proposes an optimization technique for cache memory considering wire delay. We implement this technique with SimpleScalar 3.0d and evaluate with SPEC95 CINT and SPEC2000 CINT. In the result, IPC grows at the average of 1.17 times.

1. はじめに

近年、マイクロプロセッサにおいて著しい速度向上が成し遂げられてきた。その一方で、メモリ速度は、マイクロプロセッサほど向上しておらず、主記憶へのアクセスによる遅延がシステム全体のボトルネックとなっている[1]。そのため、従来のプロセッサでは、高速・小容量のキャッシュメモリをマイクロプロセッサとメインメモリの間に設け、メモリの一部をコピーしておくことによってこのボトルネックを緩和している。しかし、プロセスの微細化に伴うキャッシュメモリ量の増大により、キャッシュメモリにおける配線遅延が増大し、キャッシュメモリのアクセスレイテンシも増大すると考えられる。したがって、今後は、キャッシュメモリアクセスによる遅延が、システムのボトルネックになる可能性がある。

そこで、本稿では、配線遅延を考慮したキャッシュメモリ高速化手法について提案する。提案手法はデバ

*現在　三洋電機株式会社

イスレベルではなく、プロセッサアーキテクチャレベルで配線遅延の緩和を試みている。

提案手法では､バンク化されたキャッシュにおいて、頻繁にアクセスされるラインを、キャッシュコントローラからの距離が近いバンクに移動し、配線遅延によるアクセスの遅延を削減する。具体的には、LRU 置換法を用い、キャッシュヒットしたラインのデータを、キャッシュコントローラからの距離が近いバンクに移動する。しかし、従来のキャッシュモデルでは、キャッシュ内でのデータの移動は不可能であるため、本手法ではキャッシュモデルとして D-NUCA[2]を用いる。D-NUCA は､提案手法と同様にプロセッサアーキテクチャレベルで配線遅延の緩和を試みた手法であり、スイッチを用いて、バンク間のデータを移動する。D-NUCA では、データの移動は、隣接バンクであるのに対し、本手法ではデータの配置が LRU 順になるようにデータを移動する。

提案手法の有効性を示すため、サイクル駆動のプロセッサシミュレータである SimpleScalar 3.0d を用いて本手法を実装し、SPEC95 CINT、SPEC2000 CINT を用いて評価を行った。

以下 2 節で配線遅延について述べ、3 節でキャッシュメモリについて述べる。4 節で提案手法について述べ、5 節で提案手法の評価について述べる。6 節で本稿をまとめる。

2. 配線遅延とプロセス

配線遅延とは、配線抵抗や配線間容量によって生じる電気信号の遅延である。配線遅延*T*は、式(1)で表され、配線抵抗Rと配線容量Cの積RCに比例する。

$$T = 0.69R * C \quad \cdots (1)$$

単位長の配線抵抗Rは、式(2)で表され、低効率に比例し、配線の幅・高さに反比例する。

$$R = \rho_{wire} \frac{1}{W_{wire} * H_{wire}} \quad \cdots (2)$$

ここで、ρ_{wire}は配線の抵抗率を表す。また、W_{wire}・H_{wire}は配線の幅・高さを表す。配線の幅・高さはプロセスの微細化(ここでのプロセスとは､シリコンウェハから半導体チップを製造する過程を意味する)と同じ割合で減少していくと仮定できる。したがって、単位長の配線抵抗Rはプロセスの微細化の2乗に反比例して増加する。配線容量Cは、プロセスの微細化に直接影響を受けないが、配線間距離が縮小するため、影響を受ける配線数が増加するため、配線間容量は増加していくと考えられている[3]。

配線遅延の増加は、データ転送レイテンシの増加、クロック同期の困難化等、LSI に対して大きな影響を与える。したがって、プロセスの微細化が進んだ現在では、配線遅延を十分に考慮する必要があると考えられる。

3. キャッシュメモリ

3.1 プロセスとメモリ量

プロセスの微細化に伴い、マイクロプロセッサに搭載されるキャッシュメモリ量は増大し続けている（表1)。表1は、キャッシュ設計ツールであるCacti[5]を用いて、将来の2 次キャッシュへのアクセスレイテンシの増加を予想したものである。プロセッサの動作周波数はプロセスが次の段階へと微細化される毎に2 倍になると仮定し、キャッシュ容量は、面積がほぼ同じになるように設定している。なお、近年になってプロセッサの周波数の伸びは緩やかになっているが、Intel社は65nmプロセスにおいて9GHzで動作する演算器の開発に成功している[6]。

近年のマイクロプロセッサのキャッシュメモリ量は2MB程度であるが､次世代のプロセッサでは4MBを超えると推測される。しかし、表1に示すように、キャッシュメモリ量が増大するにつれ、配線遅延によりキャッシュメモリへのアクセスレイテンシも増大する。

表 1　プロセスとアクセスレイテンシ

年	プロセス	周波数	L2キャッシュメモリ量	アクセスレイテンシ
2001	130nm	1.5GHz	1MB	4
2004	90nm	3.0GHz	2MB	9
2007	65nm	6.0GHz	4MB	20
2010	45nm	12.0GHz	8MB	64
2013	32nm	24.0GHz	16MB	104

特に2010年以降に着目すると、キャッシュメモリへのアクセスアクセスレイテンシは45nmプロセスでは50サイクル、32nmプロセスでは、100サイクルを超える。

2節で述べたように、単位長の配線抵抗は、プロセスの2乗に反比例して増加するため、仮に、プロセッサにおいてキャッシュが占める面積は同じであっても、配線抵抗が増加し、キャッシュメモリへのアクセスレイテンシが増大する。表１では、デバイス技術の進歩を考慮しておらず、今後、動作周波数がどこまで伸びるかは不透明であるため、アクセスレイテンシが表1の通りに増加するとは限らないが、十分に考慮する必要がある。

今後は、プロセスの微細化により、キャッシュメモリアクセスにおける遅延もシステムの速度を低下させる要因になると考えられる。したがって、キャッシュメモリアクセスがシステムのボトルネックとならないよう、キャッシュメモリアクセスによる遅延を緩和していく必要がある。そこで、本稿では、デバイスレベルではなく、プロセッサアーキテクチャレベルから、配線遅延によるキャッシュメモリアクセスレイテンシ増加の緩和を試みる。

3.2 関連研究

従来のキャッシュモデルでは、複数のパイプラインから同時にキャッシュアクセスがあった場合、ポート数による制限による待ちが発生し、パフォーマンスの低下の要因となっている。また、タグ検索回路、データの出力回路は、最も時間がかかるラインアクセスにタイミングを合わせている。したがって、キャッシュコントローラからの距離が短く、配線遅延の影響が小さいラインへのアクセスも、キャッシュコントローラからの距離が遠く、配線遅延の影響が大きいラインへのアクセスにタイミングを合わせる必要がある。そのため、プロセスの微細化が進むつれ、キャッシュメモリアクセスによる遅延も増大する。

この問題を解決する手法として、キャッシュのバンク化が挙げられる(図 1b)。この手法は、IBM 社の Power5 プロセッサに用いられている。バンク化されたキャッシュでは、複数のパイプラインから同時にキャッシュアクセスがあった場合でも、同じバンクに対するアクセスでなければ、競合が発生することはない。また、回路が各バンクで閉じたものとなり、ラインへのアクセスの際、他のラインへのアクセスにタイミングを合わせる必要がない。そのため、キャッシュメモリアクセスによる遅延を緩和することができる。しかし、バンク数が多くなると、キャッシュコントローラから各バンクまでの配線数が多くなるため、キャッシュメモリ面積が増加するという問題がある。

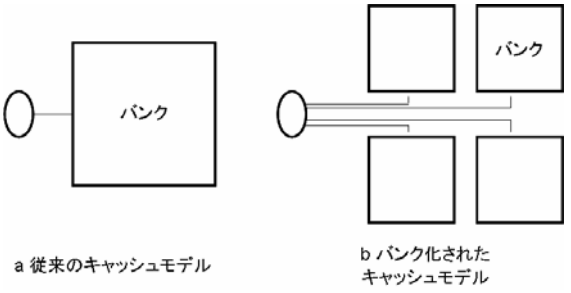

図 1　キャッシュモデル

別のアプローチとして非同期式キャッシュがある[7]。非同期式キャッシュでは、メモリ領域へのアクセスは、クロックと同期せず、非同期に行われる。従来の同期式のキャッシュでは 8 サイクルであったアクセスレイテンシが、非同期式キャッシュでは 5 サイクルに緩和される。また、クロック分配回路やラッチが必要なくなるため、消費電力削減の面でも有効である。

上の 2 つの手法は、従来のアーキテクチャを基に改良を行ったものであるが、新たなキャッシュアーキテクチャを導入してアクセスレイテンシを緩和する手法として D-NUCA(Dynamic Non Uniform Cache Architecture)が挙げられる[2]。D-NUCA はバンク間のデータ移動が可能なキャッシュモデルであり、2002 年に kim らによって提案された。D-NUCA は、バンク間でのデータの移動を可能にすることで、キャッシュへのアクセスレイテンシを削減する。D-NUCA の構成を図 2 に示す。D-NUCA は、複数の小バンクで構成され、各バンクはスイッチに接続されている。スイッチは、2D メッシュ上に配置され、各スイッチは point-to-point で接続される。そのため、配線数が増

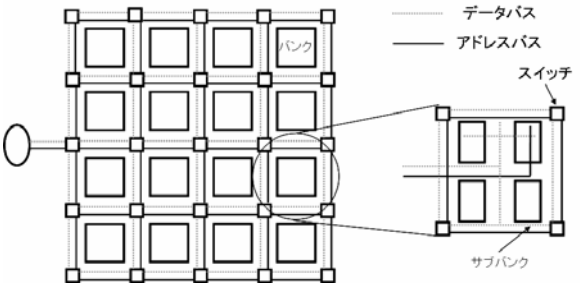

図 2　D-NUCA

加し、キャッシュメモリ面積が増大するのを防ぐことができる。また、バンク、スイッチにはバッファが設けられている。さらに、各バンクは、複数のサブバンクによって構成され、各サブバンクに対しては、同時アクセスが可能である。

従来のキャッシュモデルを改良する手法は、プロセスの微細化によるレイテンシの増加を防ぐことはできない。それに対し、D-NUCA では、バンクの大きさを調整することで、微細化によるレイテンシの増加を緩和することが可能である。しかし、データの移動が隣接バンク間に限られるため、空間的局所性を活用できない。

4. 提案手法

本節では、提案手法について説明する。3 節で述べたように、キャッシュのバンク化は配線遅延の緩和に有効である。提案手法は、バンク化されたキャッシュをさらに有効に活用する。バンク化されたキャッシュにおいて、頻繁にアクセスされるラインが、キャッシュコントローラからの距離が近いバンクに存在すれば、配線遅延によるアクセスの遅延をさらに削減できると考えられる。同様の手法として D-NUCA が挙げられるが、D-NUCA では、データの移動は、隣接バンクであるのに対し、本手法ではデータの配置が LRU 順になるようにデータを移動する。LRU 順に移動することで、同じラインに連続してアクセスする場合のアクセスレイテンシを最小にする。

提案手法ではバンク間でのデータ移動が必要であるため、キャッシュモデルとして D-NUCA を用いる。提案手法では、図 3 に示すように、各 set の各 way を、キャッシュコントローラから way 順になるよう、各バンクに割り当てる。そして、LRU 置換法を用い、キャッシュヒットしたラインを、キャッシュコントローラからの距離が近いバンクに移動する。way1 に存在したラインは way2 に移動する。以後、この動作をラインヒットした way まで連続して行う。D-NUCA に比べ、移動のコストが増加するが、提案手法では、空間的局所性があるプログラムにおいて、アクセスレイテンシの小さいバンクに連続してアクセスすることが可能になるため、アクセスレイテンシを削減できる。

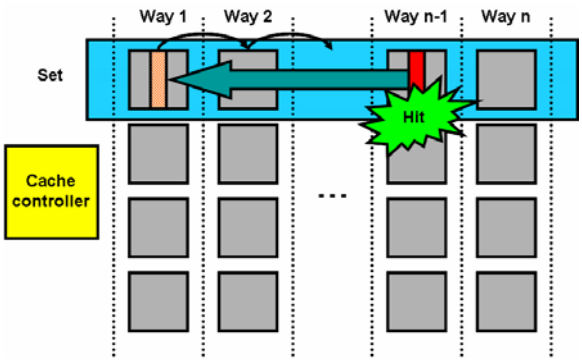

図 3　提案手法の動作例

5. 評価

5.1 実験環境

提案手法による性能向上を検証するため、SimpleScalar ver3.0d[8] の sim-outorder を用いて提案手法を実装し、IPC の向上率を求めた。sim-outorder はアウトオブオーダー実行可能なサイクル駆動のプロセッサシミュレータであり、提案手法実装による性能への影響を計測することが可能である。本稿では、Intel 社の Pentium M プロセッサアーキテクチャを基に、sim-outorder のパラメータを設定した（表 2)。ベンチマークプログラムとして、SPEC95 CINT から 6 プログラム、SPEC2000 CINT から 6 プログラム採用した。また、今回の実験は、65nm プロセス(周波数 6GHz、キャッシュ容量を 4MB)のプロセッサとしている。

5.1.1 アクセスレイテンシ

実験では、表 1 より 2 階層キャッシュの平均アクセスレイテンシを 20 とした。提案手法では、キャッシュコントローラから最も近いバンクのアクセスレイテ

表 2　SimpleScalar パラメータ諸源

L1 命令キャッシュ	方式	ダイレクトマップ
	容量	16KB
	ラインサイズ	32B
	レイテンシ	1サイクル
L1データキャッシュ	方式	4way
	容量	16KB
	ラインサイズ	32B
	レイテンシ	1サイクル
L2統合キャッシュ	方式	8way
	容量	4MB
	ラインサイズ	64B

ンシは、2 階層キャッシュにおいて最速の場合のレイテンシと同じ値とし、10 サイクルに設定した。キャッシュコントローラから最も遠いバンクのアクセスレイテンシは、2 階層キャッシュにおいて最遅の場合のレイテンシと同じ値とし、30 サイクルに設定した。さらに、各バンクのアクセスレイテンシが 1 以下になるよう Cacti で計算し、各バンクのサイズは 64KB とした。したがって、提案手法におけるバンク数は 64 となる。ウェイ数は 8way であるので、各バンクは 8*8 の 2D メッシュ型の配置となる。また、最短のバンクと最遠のバンクのレイテンシの差は 20 であり、通過するスイッチ数の差は 10 であるので、各バンク間のレイテンシは 2 となる。したがって、アクセスするまでに通過するスイッチ数を a(1<=a<=11)、データ移動にかかるオーバーヘッドを b とすると、レイテンシ T は式(3)で表される。

$$T = 10 + 2*(a-1) + b \quad \cdots (3)$$

オーバーヘッド b は、移動した way 数分だけ新たにバンクに書き込みが発生することから、各バンクのアクセスレイテンシ 1 と way 数の積となる。したがって、先頭の way1 にアクセスする場合、データ移動がないのでオーバーヘッドは 0、way8 にアクセスする場合、7 回のデータ移動が発生するのでオーバーヘッドは 7 となる。

5.2 評価結果

評価結果を表 3 に示す。表 3 の理論値とは、提案手法において、最もアクセスレイテンシの小さなバンクに必要なデータが存在すると仮定した場合の値である。

実験から、提案手法を用いることで平均 1.17 倍の IPC 向上率が得られることがわかった。なお、移動のオーバーヘッドを考慮にしない場合の IPC 向上率は 1.22 倍である。特に、m88ksim、vpr では、高い IPC 向上率が得られた。これらのプログラムは、L2 キャッシュミス率が低く、また空間的局所性が高いため、提案手法が有効に働いたと考えられる。

また、gap、mcf では IPC の低下が見られた。空間的局所性を活用できず、L2 キャッシュのミス率が高いため、メモリアクセス時のバンク間のデータの移動がボトルネックになったと考えられる。

いくつのかのプログラムでは、IPC の低下が見られたが、IPC の平均低下率は 5%未満であり、プログラム全体の平均向上率が 17%であることを考慮すると、提案手法はキャッシュメモリのアクセスレイテンシの緩和に有効であると考えられる。

6. おわりに

本稿では、プロセッサアーキテクチャレベルで配線遅延を緩和する手法を提案した。提案手法を SimpleScalar3.0d を用いて実装し、SPEC95 CINT、SPEC2000 CINT を用いて評価した結果、平均の 1.17 倍の IPC 向上率が得られた。しかし、キャッシュミス率の高いプログラムにおいて、IPC の低下が見られた。メモリアクセス時のデータ移動に関して検討する必要がある。

提案手法は、頻繁にデータの移動が発生するため、消費電力が高くなると考えられる。そのため、今後は消費電力の観点から提案手法の改良を行っていく必要

表 3　評価結果

	compress	go	ijpeg	li	m88ksim	perl	gap	gcc	mcf	parser	perlbmk	vpr
2階層キャッシュ	1.39	0.44	1.76	1.15	0.69	0.51	0.58	0.56	0.20	0.93	0.50	0.43
提案手法	1.46	0.54	1.78	1.36	0.97	0.60	0.56	0.59	0.20	1.02	0.63	0.70
理論値	1.50	0.69	1.94	1.39	1.06	0.76	0.75	0.86	0.22	1.15	0.79	0.72
L2キャッシュミス	0.017	0.010	0.005	0.020	0.010	0.004	0.038	0.002	0.050	0.006	0.000	0.000
向上率	1.05	1.23	1.01	1.18	1.40	1.18	0.97	1.04	0.99	1.10	1.24	1.62

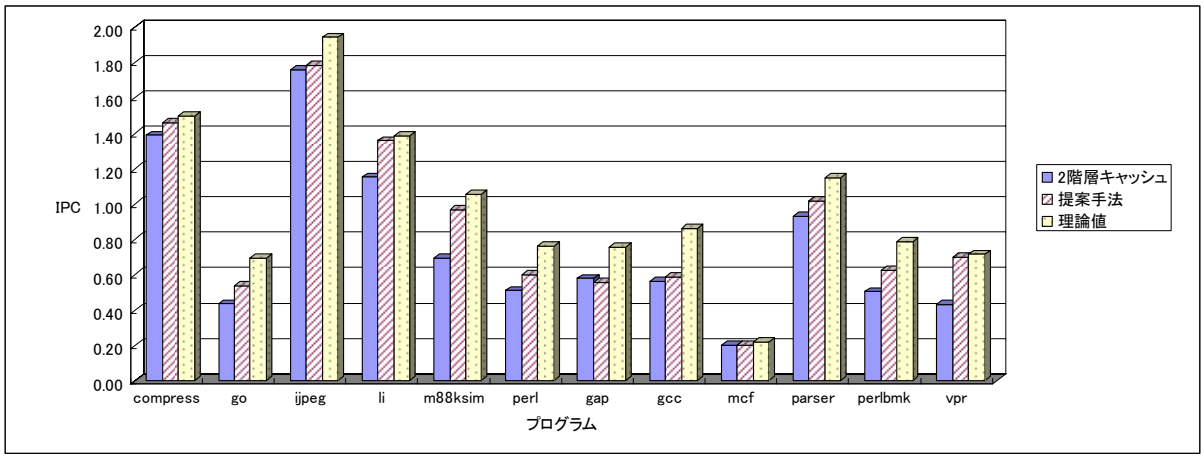

図 4　IPC 向上率

があると考えられる。提案手法では、ラインがキャッシュコントローラから LRU 順に並んでいるため、way 適応型のキャッシュや、キャッシュサイズを動的に変更する手法が有効に利用できる可能性がある。したがって、これらの手法との連携を今後の課題とする。

謝辞　本研究の一部は、21世紀COEプログラム「プロダクティブICTアカデミア」及び、科研費基盤B「ヘルパースレッドを用いたマルチスレッディングプロセッサのための高速化技術研究」によるものである.

参考文献

[1] Hennessy, J. L., and Patterson, D. A.: Computer Architecture: A Quantitative Approach, 3rd Edtion, Morgan Kaufmann Publishers(2002)

[2] Changkyu kim, Doung Burger and Stephan W.Keckler: An Adaptive,Nonuniform Cache Architectures for Wire-Delay Dominated On-Chip Caches, Proc. of the 10th Int. Con. on Architectural Support for Programming Languages and Operating Systems(2002)

[3] 山本一郎, レイアウト設計1, SoC 設計技術LSI 設計編(B コース), 半導体理工学研究センター(2003)

[4] Linda. W and R. Mike: ITRS Overview,ITRS(International Technology Roadmap for Semicondoctors) Public Home Page, http://public.itrs.net,2004

[5] CACTI, Western Research Laboratory - Compaq, http://research.compaq.com/wrl/people/jouppi/CACTI.html

[6] S. Wijeratne, N. Siddaiah, S. Mathew, M. Anders, R. Krishnamurthy, J. Anderson, S. Hwang, M. Ernest and M. Nardin: A 9GHz 65nm Intel Pentium®4 Processor Integer Execution Core, ISSCC(2006)

[7] Naffziger, S. Stackhouse, B. and Grutkowski, T.: The Implementation of a 2-core Multi-Threaded Itanium-Family Processor, ISSCC(2005)

[8] D.Burger and T.M.Austin:The SimpleScalar Tool Set, Version 2.0,University of Wisconsin-Madison Computer Sciences Department Technical Report no.1342(1997)