低レイテンシ1対1結合 マルチポート・インターリーブ・キャッシュの評価

嶋田 創[†] 安藤秀樹[†] 島田俊夫[†]

プロセッサの高性能化のために，データ・キャッシュには高いバンド幅と低いレイテンシが要求される．現在の多くのプロセッサでは，高いバンド幅を得るためにキャッシュはインターリーブによりマルチポート化されている．しかしインターリーブ・キャッシュでは，ロード/ストア・ユニットとの間に相互結合網が必要となるため，低レイテンシ化が困難である．この問題に対して，キャッシュの各バンクとロード/ストア・ユニットを1対1で結合することにより複雑な相互結合網を取り除き，アクセスを小さなバンクに限定する構成のキャッシュが有効と考えられる．この構成のキャッシュを，我々はPPCインターリーブ・キャッシュ（point-to-point connected interleaved cache）と呼ぶ．本論文では，PPCキャッシュの有効性を，回路レベルおよびアーキテクチャ・レベルでの評価により検証する．PPCキャッシュでは命令発行前にアクセスするバンクを求める必要があり，これが求められなかったときにペナルティを被る．まず最初に，このペナルティを含めたPPCキャッシュによるアクセス時間を評価し，大きな改善が見られることを確認した．次に，クロック・サイクル時間と実行サイクル数のトレードオフを考慮し，32Kバイトで最善にパイプライン化されたキャッシュを持つ16命令発行のプロセッサの性能を評価した．PPCキャッシュを持つプロセッサは，従来のキャッシュを持つ場合に比べ，クロック・サイクル時間の下限の仮定に応じて，最大10〜11%，平均6〜7%性能を改善できることを確認した．

Evaluation of a Low-latency Point-to-Point Connected Multiported Interleaved Cache

Hajime Shimada,[†] Hideki Ando[†] and Toshio Shimada[†]

Both high bandwidth and low latency are required to a data cache for high-performance processors. In most current processors, the cache is multiported by interleaving to achieve high bandwidth. However, it is difficult for the interleaved cache to achieve low latency because an interconnection network is necessary between the cache and the load/store units. To solve this problem, a cache organization that removes the complicated interconnection network by connecting each bank with a load/store unit by point-to-point and restricts a cache access to a small bank will be effective. We call the cache with this organization a PPC (point-to-point connected) interleaved cache. This paper validates the effectiveness of the PPC cache with both circuit-level and architectural-level evaluation. The PPC cache requires the bank number to be accessed before instruction issue. Penalties are imposed when the bank number is not obtained. We first evaluate the access time of the PPC cache, taking the penalties into account, and found a significant improvement over the conventional cache. Also, we evaluate the 16-issue processor performance with the 32 K-byte best pipelined cache, considering trade-offs between the clock cycle time and the execution cycle count. We confirmed that the processor with the PPC cache improves the performance by a maximum of 10-11% or an average of 6-7% over that with the conventional cache depending on the assumption of the lower bound of the clock cycle time.

1. はじめに

近年のプロセッサは，高速なクロックで動作し，複数の命令を同時に実行することで高い性能を得ている．このためにメモリ・システムには，低いレイテンシと高いバンド幅が求められる．クロックはより速く，命令発行バンド幅はより広くなる将来のプロセッサでは，メモリ・システムに対するこのような要求はさらに強くなる．一般にメモリ・システムは階層化されているが，その中でも1次キャッシュの性能改善は計算機の性能に大きな影響を与えるため，非常に重要である．

† 名古屋大学大学院工学研究科
Graduate School of Engineering, Nagoya University

本論文では，データ・キャッシュの低レイテンシ，および，高バンド幅を実現する機構について検討する．

キャッシュのレイテンシを削減するには，ヒット率を向上させ，アクセス時間を短くすることが必要である．ヒット率は，連想度を上げ容量を増加させることで改善できるが，連想度の増加は複雑さを急速に増加させるので，大きな改善のためには容量を増加させることが必要である．しかし一方で，容量を増加させるとアクセス時間が増加するので，トレードオフが存在する．特に，近年のディープ・サブミクロン半導体技術においては，ゲート遅延に比べて配線遅延が大きくなっているため，アクセス時間の増大は深刻化している．このため，容量を大きくすることでレイテンシを削減することが困難になってきている．

一方，キャッシュのバンド幅拡大については，同時に複数のアクセス要求に応える必要があるため，通常，マルチポート化により対応している．マルチポート化には，一般には次の 4 つの方法がある．1 つは，メモリ・セルをマルチポートにすることである．この方法は，各ポートから独立にキャッシュをアクセスすることができるという点で理想的であるが，実現に高いコストを要する．また，バンド幅に対しスケーラビリティがない．2 つめの方法は，空間的多重化である．この方法は，キャッシュのコピーを必要なポート数だけ用意することにより，マルチポート化するものである（たとえば，Compaq Alpha 21164[1]）．この方法は，読み出しに関しては，単一ポートのキャッシュのアクセス時間を維持したまま，バンド幅拡大を実現できるという点で優れている．しかし書き込みについては，複数のコピーの内容を同一にするために，データのストアはすべてのコピーに行わなければならないため，バンド幅は拡大しないという欠点がある．また，複数のコピーが必要なので高コストである．3 つめの方法は，時間的多重化である．単一ポートのキャッシュを 1 サイクルに複数回アクセスすることにより，マルチポートを実現する（たとえば Compaq Alpha 21264[2]）．この方法は低コストであるが，多くのポートを必要とする将来のプロセッサでは使えない．最後の方法は，キャッシュを複数のバンクに分割してインターリーブする方法である．この方法では，前述の 3 つの方法に比べ，バンクにおけるアクセス競合によるバンド幅低下や，ロード /ストア・ユニットとバンクを結合する相互結合網によるアクセス時間増大という問題があるが，コストとスケーラビリティの点で優れている．このため，多くのプロセッサで採用されている（たとえば，Intel Pentium[3]，MIPS R10000[4]，AMD Athlon[5]）．

本研究の目的は，マルチポート・インターリーブ・キャッシュのレイテンシの改善にある．この目的のために，我々は文献 6) で提案されているロード /ストア・ユニットとインターリーブ・キャッシュの各バンクを 1 対 1 で結合する構成に着目した[7),8)]．本構成のキャッシュを搭載するプロセッサでは，命令を命令ウィンドウへ書き込む前に，何らかの方法によりアクセスするバンクを求める．命令ウィンドウからは，決定されたバンクに結合されたロード /ストア・ユニットに発行され，目的のバンクにアクセスする．本方式は，インターリーブ・キャッシュから複雑な相互結合網を削除することにより，レイテンシを低減することができる．また，アクセス時間はバンクを構成する小さな SRAM のアクセス時間で決まるため，これによってもレイテンシを低減することができる．以上 2 つの効果により，スケーラビリティの高い大容量，マルチポートのキャッシュを構成することができると考えられる．本方式の有効性について，文献 6) では，2 バンクの場合に限定し，簡単な評価が行われている．しかし，バンク数が 2 より大きな場合の評価や，キャッシュのアクセス時間やスーパスカラ・プロセッサの動的スケジューリングの効果を考慮した詳細な評価は行われておらず，有効性の定量的確認が不足している．本論文では，これらの点を考慮した詳細な評価を行う．以下，ポート数分のロード /ストア・ユニットを用意し，インターリーブ・キャッシュの各バンクと 1 対 1 で結合する構成のキャッシュのことを，**PPC キャッシュ**（point-to-point connected cache）と呼ぶ．

2 章ではキャッシュのアクセス時間について概説する．3 章では提案の構成と機構について説明する．4 章では提案機構を評価する．5 章で関連研究について述べ，6 章で本研究をまとめる．

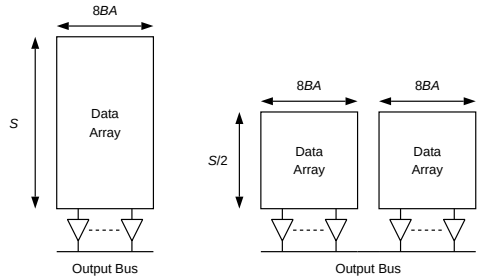
2. キャッシュのアクセス時間

本章では，キャッシュのアクセス時間について概説する．最初に単一ポートの場合について，次にインターリーブによるマルチポートの場合について説明する．

2.1 単一ポート・キャッシュ

キャッシュはデータ・アレイとタグ・アレイの 2 つの部分よりなる．アクセス時間は，それぞれの信号パスの長い方で決まる．データ・アレイおよびタグ・アレイそれぞれの信号パスの遅延時間 $T_{access}(data)$，$T_{access}(tag)$ は，以下の式で表される．

$$\begin{aligned} T_{access}(data) = {} & T_{addr}(data) + T_{decode}(data) \\ & + T_{wordline}(data) \\ & + T_{bitline}(data) \end{aligned}$$



(a) 元の構成 (b) ビット方向に分割した構成

図 1 アクセス時間を削減するためのキャッシュの構成例
Fig. 1 Cache organization example to reduce the access time.

$$+ T_{sense}(data) + T_{select}(data) + T_{output}(data)$$

$$\begin{aligned} T_{access}(tag) = {} & T_{addr}(tag) + T_{decode}(tag) \\ & + T_{wordline}(tag) + T_{bitline}(tag) \\ & + T_{sense}(tag) + T_{cmp}(tag) \\ & + T_{select}(tag) + T_{output}(data) \end{aligned}$$

ここで，$T_{addr}(array)$，$T_{decode}(array)$，$T_{wordline}(array)$，$T_{bitline}(array)$，$T_{sense}(array)$ はそれぞれ，$array \in data, tag$ 側のアドレス駆動遅延，アドレス・デコード遅延，ワード線遅延，ビット線遅延，センス・アンプ遅延である．$T_{select}(data)$ はアレイから読み出されたデータから要求されているデータを選択する出力マルチプレクサを通過する遅延，$T_{select}(tag)$ はその出力マルチプレクサを制御する遅延，$T_{cmp}(tag)$ はタグ比較の遅延，$T_{output}(data)$ は出力バスの遅延である．

キャッシュの各アレイは，論理的には 1 つのアレイであるが，アクセス時間を短縮するために，物理的には通常複数に分割されている[9]．**図 1** に，1 つのデータ・アレイをビット方向に 2 つに分割した例を示す．図において，A は連想度，B はブロック・サイズ（単位：バイト），S はセット数である．同図 (a) に示した単一のアレイの場合，$\log_2 S$ ビットのアドレスをデコードするデコーダにより，S 本のワード線の中の 1 本を駆動する．ワード線により選択されたセルは，ビット線を駆動し，センス・アンプにより信号は増幅される．タグ比較結果から得られる選択信号（図では省略）によってデータが選択され，バスに出力される．

同図 (b) は，ビット方向に 2 つに分割した場合である．各アレイを**サブアレイ**と呼ぶ．$\log_2 S - 1$ ビットのアドレスが両方のサブアレイに与えられ，データが読み出される．アドレスの第 $\log_2 S$ 番目のビットでサブアレイを選択する．選択されたサブアレイからの出力は，トライステート・バッファを介して出力バスに出力される．アドレス・デコーダは，サブアレイに多く分割するほどその複雑さが低減されるので，$T_{decode}(array)$ が短くなる．また，ビット線が短くなるので $T_{bitline}(data)$ が短くなる．しかし一方で，アドレスは複数のアレイに分配されなければならないので，$T_{addr}(array)$ が長くなる．また，出力バスに結合されるトライステート・バッファ数の増加と出力バス長の増加により $T_{output}(array)$ が長くなる．さらに，タグ・アレイとデータ・アレイの距離が増加するので，$T_{select}(tag)$ が長くなる．これらは，アレイや出力マルチプレクサの配置を工夫[10]することにより改善されるが，依然としてアクセス時間に大きな影響を与え，アレイ分割とトレードオフの関係にある．

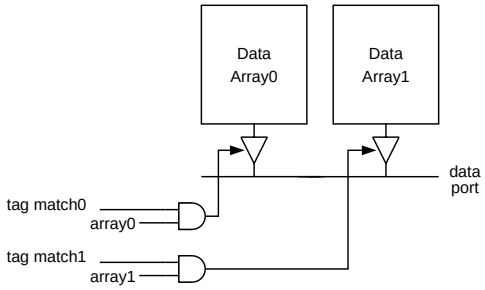
分割にはこのほかワード方向の分割がある．この分割では，ワード線が短くなるので $T_{wordline}(array)$ が短くなる．一方でアドレス・デコーダの数が増えるので，$T_{addr}(array)$ が長くなる．

以上述べたように，分割にはトレードオフが存在するが，一般には，容量が大きいほど最適なサブアレイ数は多くなる．

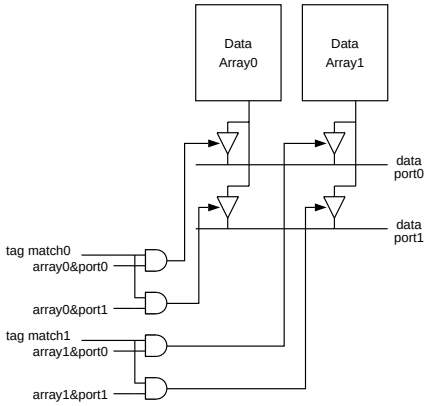
2.2 インターリーブ・キャッシュ

マルチポート・インターリーブ・キャッシュの遅延時間は，単一ポート・キャッシュの場合に比べて増加する．これは，インターリーブ・キャッシュにはバンクとポートとの相互結合網（通常クロスバー）が必要なためである．しかし前節で述べたように，高速化のために一般にキャッシュは，物理的にすでに複数のサブアレイに分割され，それらはバスで結合されているので，これを拡張してクロスバーを形成すれば，遅延が大きく増加することはない．

図 2 に，データ・アレイから読み出されたデータを出力バスに出力する論理回路を示す．この図は，データ・アレイが 2 つに分割された場合の例である．同図 (a) は文献 11) に示されている単一ポートの場合の回路であり，(b) は 2 ポートのインターリーブに対応するように拡張した回路である（実際の回路は，大きな負荷を駆動するドライバが挿入されているなど高速化のための工夫がある．ここでは図を簡単化するために論理のみ示す）．単一ポートの場合，タグ比較結果（tag match0/1）とアドレスから生成されるアレイを指定する信号（array0/1）の論理積をとった信号を，トライステート・バッファの制御ピンに与え，アレイの出力を選択し，データ・バスを介してデータ・ポートに出力する．これに対して 2 ポートの場合，まず，



(a) 単一ポート



(b) 2 ポート

図 2 出力回路
Fig. 2 Output circuit.

2 つのポートに出力する 2 つのバスを用意し，各アレイの出力を 2 つのバスにトライステート・バッファで結合する．選択は，タグ比較結果とアレイを指定する信号に加え，どのポートに出力するかを指定する信号（port0/1）の論理積とする．

クロスバーによる遅延について詳しく考えてみる．クロスバーによる遅延は，網をデータが通過する時間と網のスイッチを制御（図 2 では，トライステート・バッファの選択）する時間の 2 つに分けられる．まず，データ通過時間について考える．キャッシュの容量が小さい場合，アクセス時間を最小とするサブアレイ数は少ない．インターリーブ・キャッシュに必要なバンク数が，最適なサブアレイ数より多い場合，必要以上に分割されることとなる．これによりアクセス時間が増加する．一方，容量が大きい場合，アクセス時間最小の構成では，多くのアレイに分割される．必要なバンク数より最適なサブアレイ数が多くなれば，このようなペナルティを課せられることがないため，インターリーブによる遅延増加は小さい．明らかなことであるが，ポート数が多いほどこのペナルティを被らないキャッシュ容量の下限が上がる．

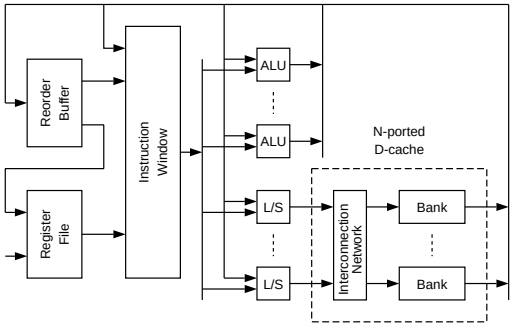


図 3 基本スーパスカラ・プロセッサの構成
Fig. 3 Baseline superscalar processor organization.

これに対して，網のスイッチは単一ポート・キャッシュには必要ないので，つねに遅延増加を引き起こす．スイッチに要する時間の多くは，どのアレイの出力をどの出力バスに出力するかの選択肢が増えることによるものである．図 2 に示した回路では，タグ比較結果を出力するゲートのファンアウトが増加することにより遅延が増加する．

3. 1 対 1 結合インターリーブ・キャッシュ

本章ではまず最初に，本論文で仮定するスーパスカラ・プロセッサの基本構成を述べる．その後，PPC キャッシュを組み込んだ構成を示す．次に，命令発行前にバンク番号を求める機構について述べる．さらに，本機構を強化する方式について説明する．本論文では，ロード/ストア命令が使用できるアドレシング・モードとして，最も一般的なベース相対のみの場合を考えるが，このほかによくサポートされているアドレシング・モードとして，インデクス修飾の場合について議論する．最後に，PPC キャッシュをプロセッサに組み込むには，アドレス計算とキャッシュ・アクセスを別々にスケジューリングする**分離ロード/ストア**と呼ばれる方式が必要であるが，その長所と短所について議論する．

3.1 基本構成

図 3 に，本論文で仮定するスーパスカラ・プロセッサの基本構成を示す．**図 4** には，ロード/ストア命令のパイプライン構成を示す．フロントエンドは，本研究には無関係なので省略している．

命令はデコード後，レジスタ・ファイルとリオーダ・バッファを参照しオペランドを得，命令ウィンドウに書き込まれる．命令ウィンドウより命令は各機能ユニットに発行され実行される．ロード/ストア命令については，さらにデータ・キャッシュをアクセスする．データ・キャッシュの長いアクセス時間がクロック・サイ

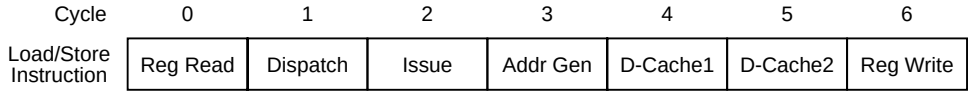


図 4 基本プロセッサにおけるロード /ストア命令のパイプライン
Fig. 4 Pipeline for load/store instructions in the baseline processor.

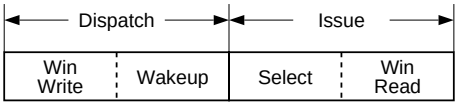


図 5 Dispatch と Issue ステージでの処理内容
Fig. 5 Breakdown of Dispatch and Issue stages.

クル時間に与える悪影響を避けるため，最近の高速なプロセッサでは，アクセス動作はパイプライン化されている場合が多い．この例では 2 段にパイプライン化されている．

3.3 節での説明を容易にするために，命令ウィンドウへの書き込みから発行までの処理をより細かく分解して説明する．**図 5** に，Dispatch と Issue の 2 つのパイプライン・ステージの処理内容とタイミングを示す．Dispatch ステージの前半で，命令は命令ウィンドウに書き込まれる（Win Write）．後半では，次のサイクルに実行が終了する命令の結果タグが命令ウィンドウにブロードキャストされ，タグ比較の結果，参照オペランドが利用可能かどうかを表すフラグが更新される（Wakeup）．次に Issue ステージの前半で，参照オペランドがそろっている命令の中から，発行する命令が選択され（Select），後半で，選択された命令が命令ウィンドウより読み出され，機能ユニットに送り出される（Win Read）．

3.2 PPC キャッシュを組み込んだ構成

図 6 に PPC キャッシュを組み込んだスーパスカラ・プロセッサの構成を示す．基本構成との大きな違いは次の 2 点である．第 1 に，バンク数と同数のロード /ストア・ユニットを用意し，それらとバンクを 1 対 1 で結合する．第 2 に，命令ウィンドウに命令を書き込む前に，アクセスするバンクの番号を計算あるいは予測により求めるユニット（**BNU**: bank number calculation/prediction unit）を設ける．得られたバンク番号を命令とともに命令ウィンドウに書き込む．発行時に参照し目的のロード /ストア・ユニットに命令を発行する．ロード /ストア・ユニットで実効アドレスを計算し，結合されたバンクにアクセスする．

PPC キャッシュは，インターリーブ・キャッシュから複雑な相互結合網を削除することにより，レイテンシを低減することができる．また，アクセス時間はバンクを構成する小さな SRAM のアクセス時間で決まるため，これによってもレイテンシを低減することができる．一方，欠点としては，通常のインターリーブ・キャッシュと比べ，データパスが大きくなる場合があることである．バンク数と同数のロード /ストア・ユニットが必要なため，バンク数を多くしバンク競合を削減しようとした場合，従来構成よりも多くのロード /ストア・ユニットが必要になる．このようなことは通常のインターリーブ・キャッシュでは必要ない．ロード /ストア・ユニットの増加によりコスト増加，バイパス遅延の増加などが生じる．

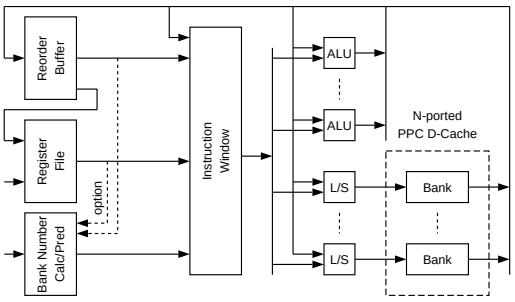


図 6 PPC を搭載するスーパスカラ・プロセッサの構成
Fig. 6 Superscalar processor organization with the PPC cache.

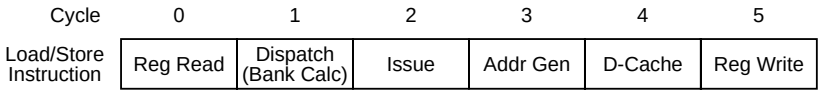
3.3 バンク番号の計算と予測

BNU はバンク番号を得るユニットである．バンク番号を得るには，計算による方法や予測による方法が考えられる．本節ではそれらの方法について説明する．

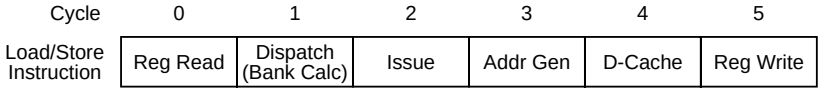
3.3.1 バンク番号の計算

バンク番号を計算により求める場合，レジスタ・ファイルあるいはリオーダ・バッファよりベース・レジスタを得た後，アドレスの下位よりバンク番号が得られる桁までオフセットを加算して求める．この方法はレジスタ参照時にベース・レジスタがすでに利用可能な場合にのみ可能である．計算はアドレスの下位のみなので，計算時間は短い．たとえば，ワードによる 4 ウェイ・インターリーブの場合，4 ビットの計算ですむ．

計算によってバンク番号を得る場合のロード /ストア命令のパイプラインを，**図 7** に示す．PPC キャッシュのアクセス時間は従来のキャッシュより短いので，1 サイクルでアクセスできるとする．図 7(a) に，BNU



(a) ベース・レジスタが利用可能な場合



(b) ベース・レジスタが利用可能でない場合

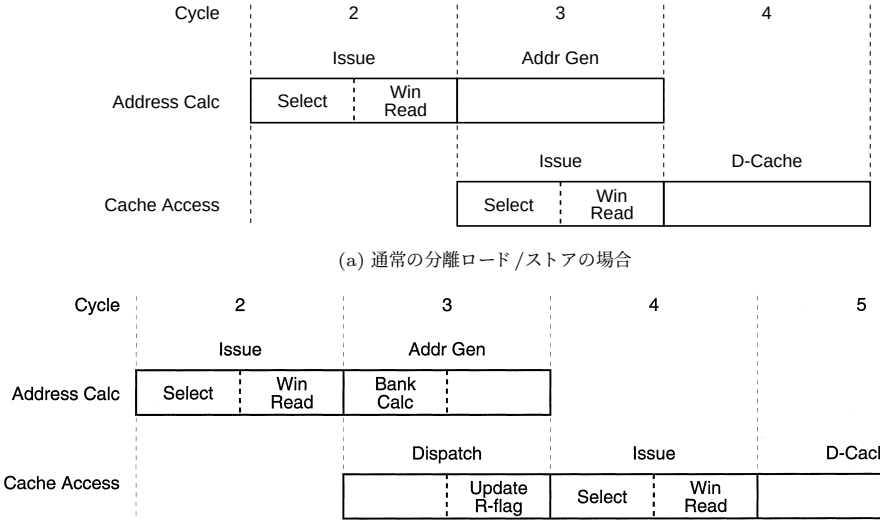
図 7 BNU でバンク番号を計算して得る場合のパイプライン
Fig. 7 Pipeline when bank numbers are obtained by calculation in BNU.

においてバンク番号が得られる場合のパイプライン構成を示す．Dispatch ステージの前半でバンク番号を計算し，後半で命令ウィンドウに書き込む．書き込まれたバンク番号は，命令がどの資源を要求するかを示すフラグの 1 つであり，Issue ステージの Select 時に参照される．バンク番号の書き込みタイミングは，命令ウィンドウへの命令の書き込みタイミングより半サイクル後ろとなるが，参照は Issue ステージなので問題ない．この仕様に合わせて，命令が要求するその他の資源についても，要求資源フラグをセットするタイミングを Dispatch の後半に変更する．

一方，ベース・レジスタが利用可能でないために，BNU でバンク番号が得られない場合，利用可能になるのを待つのではなく，通常どおり命令ウィンドウに書き込む．その後，分離ロード/ストアと同様の動作で，アドレス計算の後，目的のロード/ストア・ユニットに発行する．ここで，分離ロード/ストアとは，1 つのロード/ストア命令を，アドレス計算とキャッシュ・アクセスの 2 つの操作に分解し，それらを別々にスケジュールし実行する方式をいう．図 7 (b) にパイプライン示す．サイクル 2 に，ベース・レジスタが利用可能になるとする．命令は，実効アドレスを計算するため ALU に発行される．このとき，通常の命令発行と異なり，当該命令が格納されている命令ウィンドウのエントリを削除しない．サイクル 3 で実効アドレスを計算する．このサイクルの早期にバンク番号を得ることができるので，命令ウィンドウ内の当該命令のエントリに，得られたバンク番号を書き込む．サイクル 4 では，命令ウィンドウの当該命令のエントリに実効アドレスが書き込まれ，再スケジューリングされて目的のロード/ストア・ユニットに再発行される．サイクル 5 では，ロード/ストア・ユニットのアドレス計算回路をバイパスし，データ・キャッシュをアクセスする．サイクル 6 で得られたデータを書き込む．

以上の動作は，分離ロード/ストアとほぼ同一であるが，本方式では 2 度目の発行サイクルを隠蔽できない点が異なる．**図 8** に命令の発行タイミングの詳細を示す．図の上方にあるサイクル番号は，図 7 (b) のそれと対応している．図 8 (a) は通常の分離ロード/ストアの場合である．アドレス計算とオーバラップしてキャッシュ・アクセス操作の発行動作を行うことができるため，オーバヘッドは生じない．同図 (b) に PPC キャッシュにおける場合を示す．アドレス計算の前半にバンク番号が得られ（Bank Calc），そのサイクルの後半に命令ウィンドウの当該エントリの要求資源フラグを更新する（Update R-flag）．次のサイクルに選択，命令ウィンドウの読み出しが行われ，再発行を完了する．以上のように，PPC キャッシュの場合，ロード/ストア・ユニットはキャッシュのバンクと 1 対 1 で結合されているため，資源要求の調停のためにバンク番号が必要である．このため，バンク番号が命令ウィンドウに書き込まれなければ，発行する命令の選択を開始することができない．このようなことは通常の分離ロード/ストアは生じない．このため，再発行に余分に 1 サイクルを要する．つまり，再発行のサイクルが，BNU でバンク番号を得られなかったことによるペナルティということができる．

計算によりバンク番号を得る方法では，数ビットの加算器が必要なだけなので，3.3.2 項で述べるストライド予測による方式より低コストであるという利点がある．一方，欠点としては，4 章で評価を行うが，ベース・レジスタが計算時点で利用可能である確率があまり高くないという点である．このため，PPC キャッシュの恩恵を得る機会が減少する．もう 1 つの欠点は，実装によってはクロック速度に悪影響を与える可能性があることである．本説明では，命令の Dispatch と Issue を 2 サイクルで行うプロセッサを仮定したため，バンク番号の計算を Dispatch ステージの前半に組み込むことができ，クロック速度に影響を与える可能性を少なくすることができた．しかし，Dispatch と



(a) 通常の分離ロード /ストアの場合

(b) PPC キャッシュにおいて BNU でベース・レジスタが利用可能でない場合

図 8 通常の分離ロード /ストアと PPC キャッシュにおける命令発行のタイミング
Fig. 8 Timing of instruction issue in usual split load/stores and PPC cache.

Issue を 1 サイクルで行うようなプロセッサでは，クロック速度に影響を与えてしまう．レジスタ読み出しのステージで行うように変更しても，レジスタ読み出しはクリティカル・パスの 1 つであるから，やはりクロック速度に影響を与える可能性がある．この問題は，計算するビット数を減らすことができれば低減することができる．もし桁上げを予測することができれば，バンク番号指定に使用する桁のみの加算ですませることができる．我々は文献 8) において，論理積 1 ゲートを用いた予測方法を提案した．この方法では，たとえばライン・インターリーブでは，ベース・レジスタとオフセット値におけるライン内オフセット部（LO: line offset）の最上位ビットの論理積をとったものとする．LO の最上位桁のレジスタ値とオフセット値が (1,1) のときに桁上がりがあり，(0,0) のときには桁上がりがないことを利用するものである．(1,0) あるいは (0,1) のときは LO の最上位より下位のビットに依存するが，これらのビットがランダムでも 50%の確率で予測は成功する．したがって，この予測手法は，最終的には 75%程度の確率で成功すると考えられる．また，オフセット値は 0 である確率は高く（文献 8) での測定では 24%），このときは桁上がりはないので，予測成功の確率はさらに高いと期待できる．

3.3.2 バンク予測と予測失敗からの回復動作

3.3.1 項で述べた方法では，バンク番号計算時にベース・レジスタが利用可能でない場合，1 サイクルのペナルティを被る．これに対し，バンク番号の予測を行い，それに従って投機的にロード /ストア・ユニットに発行すれば，レイテンシ削減に効果がある．

バンク予測として，大きく分けて 2 つの方法を示す．1 つは，最新のベース・レジスタが利用できない場合，その以前の値を予測値とし，計算により求める方法を提案する．以下これを**古いレジスタ予測方式**と呼ぶ．ベース・レジスタの更新によってバンク番号が影響を受けないなら，予測は正しいものとなる．たとえば，ワードによる 4 ウェイ・インターリーブの場合，4 ワードの整数倍のストライドで変化するならば，アクセスするバンクは変化しないので 100%正しく予測できる．また，ベースが 1 バイトのストライドで変化する場合，ベース・アドレスの変化がワード境界を超えるときのみ予測を誤るので，75%の予測精度を得ることができる．この方法は，予測に要するコストが非常に小さいという利点がある．一方，バンク番号を得るためには計算が必要なため，実装によっては前述のようにクロック速度に影響を与える可能性がある．

2 つめの方法は，一般的なアドレス予測方式を修正して用いる方法である[6),12)]．バンク予測は，アドレス予測[13)~17)]のサブセットと考えられる．予測器の修正には，バンク予測に特化するいくつかの方法が考えられる．まず，バンク番号はアドレスの一部なので，値履歴表にはバンク番号を指定する部分を含むアドレスの下位のみ保持すればよい．これによりコストを大幅に削減できる．また，後述するが予測誤りによるペナルティは小さい．加えて，少ないバンクを予測する

ことは比較的容易である．以上を考慮すると，タグ・チェックや信頼性推定機構を省略する選択肢がある．これらは予測率（全ロード/ストア命令に対し正しくバンクを予測できた割合）の向上とコスト削減の効果がある．古いレジスタ予測方式と異なり，レジスタ読み出しと並行して予測を行うことができるので，クロック速度への影響はほとんどない．

予測により命令が発行される場合，その検証はアドレス計算時に行う．発行後も予測誤りに備えて，命令ウィンドウのエントリを削除せず，予測が正しいことが分かった時点で削除する．予測が誤りと分かった場合，3.3.1 項で説明した方法と同様の手順で，命令を再発行する．再発行のための 1 サイクルが，予測誤りのペナルティである．また，命令ウィンドウのエントリの削除が遅れるので，命令ウィンドウがより頻繁に一杯になるという問題がある．これも実装によっては，ペナルティとして課せられる．

3.4 性能向上のための強化

予測により命令を発行する場合，使用されないすべてのロード/ストア・ユニットに対しブロードキャストすれば，たとえ予測を誤ってもペナルティを被る確率を下げることができる[6]．ブロードキャストを行う命令の選択方法としては，たとえば，予測により発行する命令のうち，プログラム順で最も古いものとする方法や，予測の信頼性が低いものとする方法が考えられる．

命令ウィンドウとすべての機能ユニットの間にはすでに相互結合網が形成されているので，ブロードキャストによる通信の複雑さの増加はほとんどない．予測ミスを知らせる信号は，ブロードキャストされた各ユニットからの予測ミス信号の論理積をとる必要があるが，予測の正誤はサイクルの早期に判明するので問題ない．

3.5 インデクス修飾アドレシングの場合

本論文では，ロード/ストア命令が使用できるアドレシング・モードとして，ベース相対のみの場合を仮定しているが，このほかによくサポートされているアドレシング・モードとして，インデクス修飾がある．たとえば，SPARC では，ベース相対とインデクス修飾の両方をサポートしている．本節では，インデクス修飾アドレシングの場合について議論する．

アドレシング方式が異なるだけなので，これまで説明した PPC キャッシュの方式において変更しなければならない点は，BNU のみである．まず，計算によりバンクを求める機構においては，当然であるが，2 つのレジスタを加えてバンクを求める必要がある．バンク番号を求める計算時間は変わらない．2 つのレジスタが利用可能な確率は，1 つのレジスタが利用可能な確率より低いので，計算可能な確率は下がり，BNU としてのこの方式の有効性は低下する．古いレジスタの予測でも同様に，2 つのレジスタの参照が必要なため，予測精度は低下する．ストライド予測については，アドレスをどのようにして生成するかは，アドレスの履歴とは無関係なので，予測精度は変わらないと思われる．

3.6 分離ロード/ストア

PPC キャッシュでは，前述のように，分離ロード/ストア方式を使用しなければならない．本節では，分離ロード/ストアの長所/短所について議論する．

短所としては，まず命令発行バンド幅をより多く消費するということがあげられる．また，アドレス計算に ALU を用いるので，ALU がより多く消費される．これらにより IPC（instruction per clock cycle）が低下する可能性がある．回路上の短所としては，通常の場合，アドレス計算とキャッシュ・アクセスの遅延の合計が設計目標を満たせばよいが，分離すると，それぞれの遅延が設計目標を満たさなければならないということがあげられる．アドレス計算のサイクルにキャッシュ・アクセスの一部の回路を移動させることが可能な通常の場合に対し，そのようなことは分離の場合できない．

長所としては，メモリ依存を考慮した正確なスケジューリングが可能な点である．メモリ依存を知るためには，データ・アドレスが必要である．分離ロード/ストアでは，データ・アドレスが命令ウィンドウの中で参照できるので，アドレスを比較することにより依存が判明し，それに基づいた正確なスケジューリングが可能となる．これに対し，通常のロード/ストアでは，スケジューリング時にアドレスが判明していないので，先行するストアがすべて発行されなければロードは発行できない．これは偽の依存を発生させることになり，IPC が低下する．

分離ロード/ストアで，メモリ依存を解析してスケジューリングする場合，命令ウィンドウにアドレスを格納するサイクルが必要となる．このサイクルを必要なときのみ挿入する最適化は容易である．つまり，先行するすべてのストアが発行されているなら，ロードはアドレス計算に続いてキャッシュをアクセスするという方法がとれる．このタイミングでの動作を図 8 (a) で説明した．

本章の説明では，分離ロード/ストアを 1 つの命令ウィンドウで実装したが，実装の容易さの点から，

表 1 仮定した 0.18 μm プロセス技術におけるトランジスタと配線の特性
Table 1 Features of transistors and wires in the assumed 0.18 μm process technology.

Transistor				Mid-Level Metal			Top-Level Metal		
Gate Cap. (fF/μm)	Junction Cap. (fF/μm)	Transistor Res. (K-μm)	Supply Voltage (V)	Width (nm)	Res. (m/m)	Cap. (fF/m)	Width (nm)	Res. (m/m)	Cap. (fF/m)
1.56	1.51	3.6	1.8	320	107	0.253	530	36	0.270

キャッシュ・アクセス操作を別のウィンドウで保持する方が有利である．一般に，この命令ウィンドウのことを LSQ（load/store queue）と呼ぶ．メモリ依存を考慮したスケジューリングのために，アドレス比較を行う必要があるが，これは別のバッファにする方が実装が容易である．また，ロードが LSQ 内にとどまっている先行ストアに依存していた場合，LSQ 内でデータを受け取るよう回路を構成する必要がある．このフォワーディング機構を実装するためにも，別バッファにする方が有利である．ただし，これは実装上の問題であり，論理的には 1 つのウィンドウで構成する方法とエントリ数の違いを除いて等価である．また，3.1 節の基本構成の説明では，一般性を欠かないために，分離しない通常のロード/ストア方式を仮定したが，分離ロード/ストアの方が正確なスケジューリングにより高い IPC が得られるので，以下の評価においては，比較対象の基本プロセッサも分離ロード/ストアを仮定している．

4. 評価結果

本章では，最初にインターリーブ・キャッシュのアクセス時間の評価を行い，PPC キャッシュによるアクセス時間の削減について評価する．次に，BNU の性能によって，PPC キャッシュの実効的なアクセス時間が変化するので，それについて評価する．さらに，BNU に用いるストライド・バンク予測器について評価する．最後に，PPC キャッシュをプロセッサに搭載した場合の性能を評価する．

4.1 キャッシュ・アクセス時間の評価

アクセス時間の評価方法について述べた後，評価結果を示す．

4.1.1 評価方法

CACTI[10),11)]を修正し，キャッシュのアクセス時間を求めた．CACTI は，与えられたキャッシュ容量，ブロック・サイズ，連想度の下で，アクセス時間を最小にするアレイの分割数と，そのときのアクセス時間を求めるプログラムである．我々は，CACTI に対し次のような修正を加えた．

第 1 に，インターリーブ・キャッシュのアクセス時間を測定できるよう，相互結合網に関わる回路を加えた．第 2 に，最近のプロセス技術に適応できるように，トランジスタと配線のパラメータについて次のことを行った．トランジスタの特性に関し CACTI で仮定されているパラメータを，最小加工寸法と電源電圧に対してスケーリングできるようようにした（たとえば文献 18) 参照）．配線に関するパラメータは，文献 19) に示されているものを用いた．**表 1** に，測定に用いた 0.18 μm プロセス・パラメータを示す．我々が仮定したトランジスタ特性は，半導体回路の論文誌に掲載されている文献，たとえば，文献 20) のものとほぼ一致しており，妥当なものと考える．第 3 に，配線遅延の影響を小さくするために，サブアレイ配置の最適化，負荷の大きなトランジスタのサイジング，長い配線へのリピータの挿入，配線層の使い分け（アレイ内部に中層メタル，アレイ間に高層メタル）を行った．修正の詳細については，本論文の範囲を超えるので省略する．

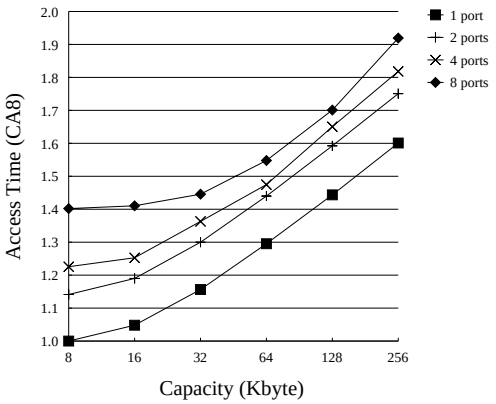


図 9 キャッシュのアクセス時間
Fig. 9 Cache access time.

4.1.2 アクセス時間

図 9 に，32 バイト・ブロック，2 ウェイ連想で，容量を 8 K〜256 K バイトで変化させたときのアクセス時間の測定結果を示す．縦軸は，8 K バイトのキャッシュのアクセス時間 959 ps で規格化した遅延である．以下では，8 K バイトのキャッシュのアクセス時間の r 倍の遅延のことを，r CA8 と表すこととする．8 KB

バイトのキャッシュを基準としているのは，この 10 年間のハイエンドの多くのプロセッサにおいて，搭載するキャッシュ容量は 8 K〜64 K バイトの範囲で変動している．また，キャッシュ・アクセスはプロセッサのクリティカル・パスの 1 つであるため，これがクロック・サイクル時間を決定することがよくある（このため，プロセッサの世代が新しくなるときに容量が小さくなることもある）．キャッシュ容量の変動の範囲での最小値である 8 K バイトでの遅延を基準とすれば，クロック・サイクル時間との関係が直感的に分かる．

図 9 から分かるように，単一ポートでは，32 K バイト以上では，容量を倍にするごとにおよそ 0.14 CA8 で増加していく．16 K バイト以下では，その量は小さい．

ポートを増やすと，2.2 節で述べたように，アレイの必要以上の分割と相互結合網の遅延でアクセス時間が増加する．容量が大きい場合（32 K バイト以上）は，遅延増加は相互結合網によるもののみである．1 ポートから 2 ポートにすると，網のスイッチ論理が必要となり，遅延が大きく増加する．この量はおよそ 0.14 CA8 である．ポートをそれ以上にしても，選択論理のファンアウトが増加するだけなので増加量は小さく，2 から 4 ポートにするとおよそ 0.05 CA8，4 から 8 ポートにするとおよそ 0.07 CA8 増加する．一方，容量が小さい場合，アレイの分割が大きく作用する．この影響は大きく，たとえば，容量 128 K バイトでは単一ポートから 8 ポートにすると，0.25 CA8 の遅延増加であるが，容量 8 K バイトでは 0.40 CA8 も増加する．

図 10 に，PPC キャッシュによるアクセス時間の削減の割合を示す．一般に，ポート数が多いほど，より多くのバンクに分割されるため，PPC キャッシュによるアクセス時間削減率は大きい．たとえば，64 K バイトのキャッシュでは，8 ポートで 35%，2 ポートでも 20%ほどアクセス時間を削減できる．

また，前述のように単一ポート・キャッシュの遅延時間は，容量を増加させると，緩やかな増加から急な増加傾向を示すので，キャッシュ容量を大きくすれば PPC キャッシュによるアクセス時間削減量も大きくなる．しかし，単一ポート・キャッシュの遅延時間の容量に対する増加量は一定なので（増加率が一定というわけではない），PPC キャッシュの効果は容量増加に対し低減の傾向を示す．図 10 より，2 ポートと 4 ポートの場合でこの現象が見られる．8 ポートの場合は測定の範囲でこの現象は見られないが，さらに大きな容量に対して測定すれば観測されると思われる．

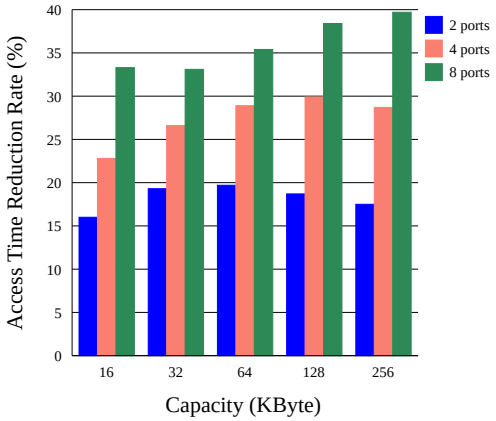


図 10 PPC キャッシュによるアクセス時間の削減
Fig. 10 Access time reduction with the PPC cache.

4.2 PPC キャッシュの実効アクセス時間

PPC キャッシュを使用することにより，前節で示したアクセス時間削減の利益を享受するためには，BNU でアクセスするバンクが正しく得られることが必要である．3.3 節で述べたように，BNU で正しいバンク番号が得られなければ，1 サイクルのペナルティを被る．これを考慮に入れたアクセス時間が，PPC キャッシュの**実効的なアクセス時間**（effective access time）である．実効アクセス時間は，キャッシュ・アクセス時間の期待値であり，PPC キャッシュの性能を知るための指標となる．本節では，実効アクセス時間を評価する．なお，実効アクセス時間に対し，4.1 節で求めたキャッシュそのもののアクセス時間のことを，**物理アクセス時間**（physical access time）と呼ぶこととする．

4.2.1 評 価 方 法

BNU で正しいバンク番号が得られる確率を **CBR**（correctly bank number obtained rate），正しいバンク番号が得られなかった場合のペナルティを **PPC ペナルティ**と呼ぶこととする．PPC キャッシュの実効アクセス時間は，次の式で求められる．

$$Effectiveaccesstime = PPCphysicalaccesstime + PCCpenalty \times (1 - CBR)$$

ここで，PPC ペナルティは命令の再発行に必要な 1 クロック・サイクルに相当する時間であるが，これを以下の計算では PPC の物理アクセス時間とする．キャッシュ・アクセスがプロセッサのクリティカル・パスの 1 つであることはよく知られている．ここでは，PPC キャッシュを導入したプロセッサでも PPC キャッシュのアクセス時間がクロック・サイクル時間を決定すると仮定する．この仮定は，キャッシュ以外のクリティ

表2 ベンチマーク・プログラム
Table 2 Benchmark programs.

program	input	instructions	% loads	% stores
compress95	bigtest.in	95M	20.8%	13.7%
gcc	genoutput.i	84M	26.2%	14.3%
go	2stone9.in	75M	21.3%	6.7%
ijpeg	specmun.ppm	450M	19.1%	7.8%
li	train.lsp	183M	25.7%	16.4%
m88ksim	ctl.in	420M	20.2%	9.8%
perl	scrabbl.in	80M	27.5%	18.8%
vortex	vortex.in	80M	30.0%	25.0%

カル・パスがクロック・サイクル時間を決定することがあるので，つねに正しいわけではない．特に，1つのバンクが小さいほど，他のパスがクロック・サイクル時間を決定する可能性が大きい．したがって，この仮定の下でのPPCキャッシュの実効アクセス時間は下限を示すこととなる．このように，以下の測定結果を見るには注意を要するが，BNUを考慮したPPCキャッシュの実効アクセス時間を知るうえではよい指標と考える．

BNUのCBRは，プロセッサの動的な振る舞いに関係するため，プロセッサの実行シミュレーションが必要である．シミュレーションには，SimpleScalar Tool Set[21]中のout-of-order実行シミュレータを基に，我々の機構を組み込んだシミュレータを用いた．命令セットはMIPS R10000である．ベンチマーク・プログラムにはSPECint95を用い，そのバイナリはGCC ver.2.7.2.3，オプション-O6 -funroll-loopsでコンパイルして得た．**表2**に各ベンチマーク・プログラムに対する入力セット，実行命令数，ロードとストアの占める割合を示す．入力は，測定時間が過大にならないように，関数の出現頻度をほぼ維持しつつ調整している．また，プログラムの特定の部分を繰り返すプログラムについては，その部分を抜き出し繰返し回数を調整している．

データ・キャッシュのポート数が，2，4，8である3つのプロセッサ・モデルP2，P4，P8を仮定した．データ・キャッシュは，ポート数と等しいバンクに分割されているとする．P2モデルは，4命令発行で，命令ウィンドウとして32エントリのRUU（register update unit）と16エントリのLSQ（load/store queue）を持つ．機能ユニットには，2つのALU，2つのロード/ストア・ユニット，1つの整数乗除算ユニット，2つの浮動小数点ALU，1つの浮動小数点乗除算ユニットを持つ．これらの機能ユニットのレイテンシはR10000とほぼ同じである．P4, P8モデルはP2モデルに対し，上記資源（命令ウィンドウと機能ユニット）をそれぞれ2倍，4倍持つ．

命令ウィンドウにRUUを用いているので，命令ウィンドウのエントリはコミットされるまで削除されない．したがって，3.3.2項で述べた発行後のエントリの削除の遅れにより命令ウィンドウ占有時間が長くなることによるペナルティは課せられない．ただしコミットも遅れるならば，そのペナルティは課せられたことになる．また，従来のキャッシュを用いるプロセッサにおいてもロード/ストア命令は，LSQを用い分離方式で，メモリ依存を考慮した正確なスケジューリングが行われる．

L1データ・キャッシュは，ポート数以外は全モデルで同一であり，容量32Kバイト，32バイト・ブロック，2ウェイ連想である．ヒット・レイテンシは，従来のキャッシュの場合2サイクル，PPCキャッシュの場合1サイクルとした．L1命令キャッシュは完全，L2キャッシュは，単一ポート，容量2Mバイト，統合，64バイト・ブロック，4ウェイ連想で，6サイクルのヒット・レイテンシ，36サイクルのミス・レイテンシとした．分岐予測器には，13ビットのインデクス，6ビットの履歴のgshare[22]と，2Kエントリ，4ウェイ連想のBTB，および，16エントリのリターン・アドレス・スタックを用いた．分岐予測ミス・ペナルティは5サイクルとした．なお，4.4節ではキャッシュのヒット・レイテンシと分岐予測ミス・ペナルティを種々変えて測定を行うが，これらがCBRに与える影響は，測定の結果，非常に小さい．このため，上述のパラメータでの測定結果のみ示す．

BNUと命令発行に組み込む機構として，以下の5つの機構を仮定した．

C ：ベース・アドレスが利用可能なとき，バンクを計算により求める．
O ：古いレジスタ予測によってバンクを予測する．
S ：ストライド予測によってバンクを予測する．

S+B ：ストライド予測によってバンクを予測し，発行の際，プログラム順で最も早い命令を使用されないロード/ストア・ユニットにブロードキャストする．

S+B+C ：計算可能な場合は計算により，そうでないときは，ストライド予測によってバンク番号を得る．予測により発行する場合は，前述のポリシーでブロードキャストする．

なお，ストライド・バンク予測器は 4K エントリの表より構成される．この表は，ロード/ストア命令の命令アドレスでインデクスされ，各エントリは対応する命令が最後にアクセスしたアドレスの下位よりバンク番号までのビット（最終値）と，その前にアクセスしたアドレスとの差分の下位よりバンク番号までのビット（ストライド）を保持する．最終値とストライド（初期値はいずれも 0）を加えて予測値とする．タグはない．また，ストライドが検出されたかどうかを表す状態も存在しない．この方式は，文献 16) で示された 3 状態の予測器より単純であるが，バンク予測としては本方式の方が有効である．このことについては，4.3 節で述べる．

4.2.2 実効アクセス時間

まず最初に，BNU に C 機構のみを実装した場合の評価結果を示す．C 機構のみでは，ほとんど場合で実効アクセス時間を削減することはできなかった．**表 3** に，従来のキャッシュに対する PPC キャッシュの実効アクセス時間の増加率を示す．正の数は増加したことを意味し，負の数は減少したことを意味する．同表から分かるように，ほとんどの場合で実効アクセス時間は増加している．特に，ポート数が少ないプロセッサほど物理アクセス時間の削減量が小さいため，実効アクセス時間の増加率が大きく，P2，P4 では，すべてのベンチマークで増加した．

このように実効アクセス時間を削減できなかった理由は，BNU でバンク番号を計算する際に，ベース・レジスタが利用可能な割合が小さいことである．**表 4** に，BNU でベース・レジスタが利用可能であった割合を示す．同表より分かるように，平均で 40〜50%程度しか利用可能でない．このため，頻繁にペナルティが課せられ実効アクセス時間が増加した．以上より，C 機構より高い確率で正しくバンクを指定できる機構が必要であることが分かる．

次に，C 機構以外の場合の実効アクセス時間の削減率を**図 11** に示す．正の数は減少したことを意味し，負の数は増加したことを意味する．また，**図 12** に CBR を示す．評価結果より次のことが分かる．

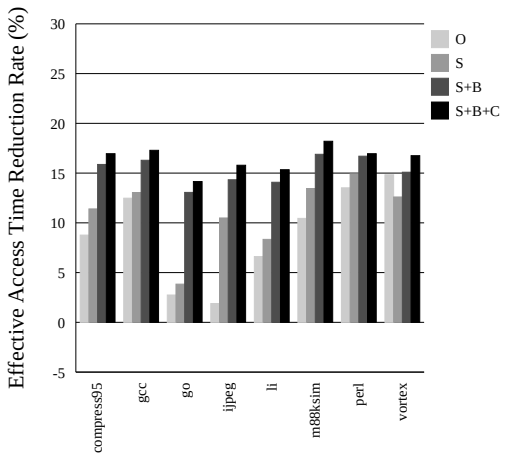
- 予測を用いることで CBR は大きく向上する．たとえば P4 において，O 機構では 59〜78%，S 機構では 71〜86%まで向上する．当然ながら，ポート数が少ないほど CBR は高い．S 機構の CBR はベンチマーク平均で，P2 で 90%なのに対し，P8 では 72%である．
- 予測にブロードキャストを導入すれば，ポートが多い場合において大きく CBR を向上できる．S から S+B 機構にすることにより，平均で P2 では 5%ポイントしか改善しないが，P8 では 7%ポイント改善する．特に，ストライド予測で CBR の低い gcc，go，li において効果が大きい．
- さらに，ベース・レジスタ利用可能時に計算を行うことを加えれば，P8 において CBR は 2〜6%ポイント改善する．
- ポート数が多いほど高度な BNU/発行機構を使う利点が現れてくる．その結果，ポート数が増加しても CBR が大きく低下することはない．ベンチマーク平均で P2 から P8 の CBR の減少量は，O 機構では 21%ポイントもあるが，S+B+C 機構では 13%ポイントと小さくなる．
- CBR はポート数が多い場合の方が低いが，物理

表 3 C 機構を持つ PPC キャッシュによる実効アクセス時間の増加率 (%)
Table 3 Increase of the effective access time in PPC cache with C mechanism (%).

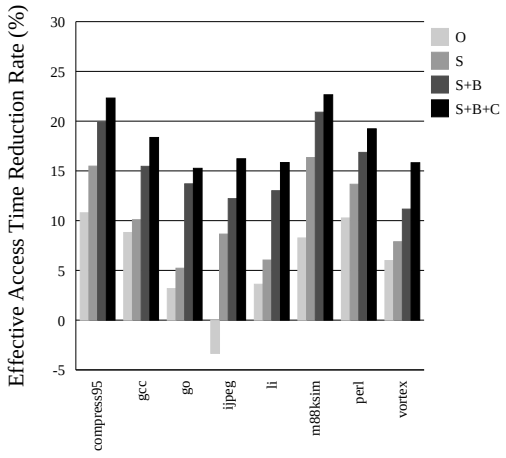
program	P2	P4	P8
compress95	9.1	0.5	-8.8
gcc	20.6	15.4	7.8
go	31.0	21.3	11.8
ijpeg	16.8	11.7	10.0
li	25.2	20.2	12.5
m88ksim	20.0	13.4	4.4
perl	18.4	15.7	8.6
vortex	16.2	14.2	7.8

表 4 BNU でベース・レジスタが利用可能な割合
Table 4 Availability rate of base registers at BNU (%).

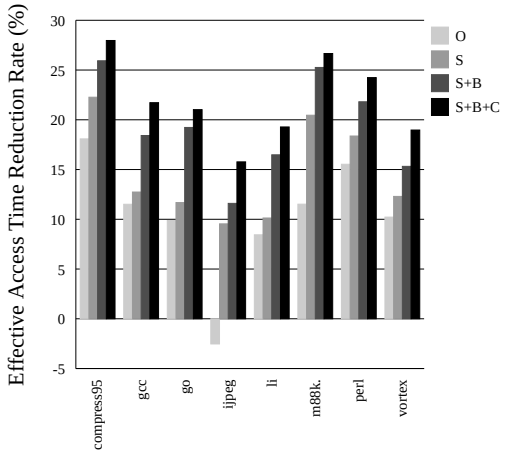
program	P2	P4	P8
compress95	64.8	62.9	63.5
gcc	50.5	42.6	38.7
go	37.6	34.6	32.7
ijpeg	55.2	47.7	35.4
li	44.7	36.1	31.6
m88ksim	51.3	45.3	43.7
perl	53.2	42.2	37.4
vortex	55.9	44.3	38.7



(a) P2

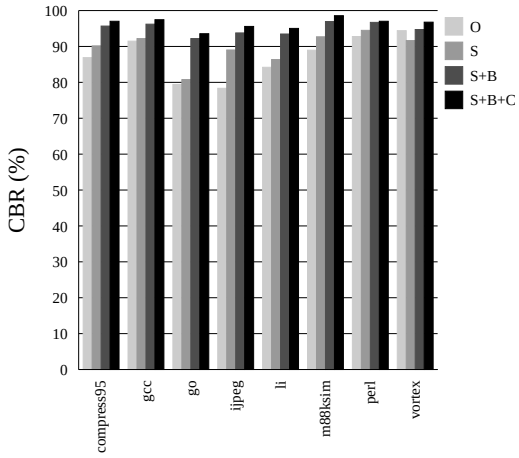


(b) P4

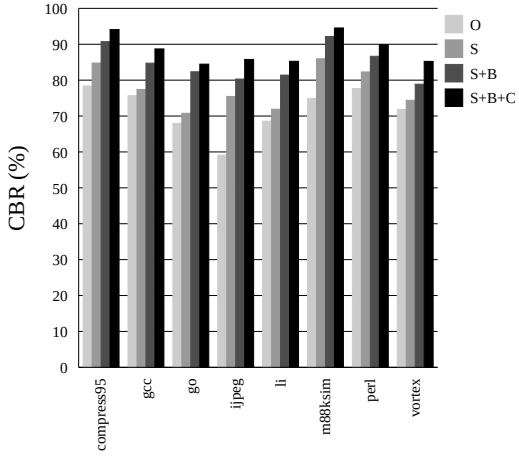


(c) P8

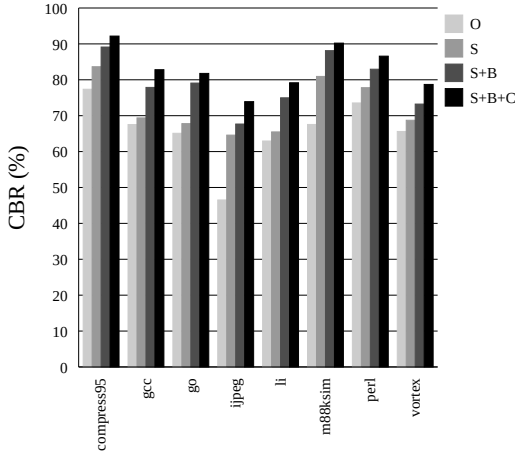
図 11 実効アクセス時間の削減率
Fig. 11 Effective access time reduction rate.



(a) P2



(b) P4



(c) P8

図 12 CBR
Fig. 12 CBR.

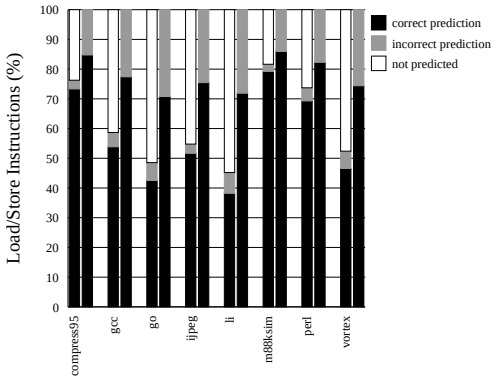


図 13 予測率
Fig. 13 Prediction rate.

アクセス時間の削減率が大きいので，実効アクセス時間の削減率はポート数が多いほど大きい．削減率は S+B+C 機構の場合で，P2 で平均 16%，P8 では 22%もある．

4.3 ストライド・バンク予測器の評価

前節で述べたように，我々はストライド・バンク予測器として，通常のストライド・アドレス予測器とは異なり，ストライド検出を表す状態とタグの両方を持たない構成とした．PPC キャッシュでは，バンク予測を誤ってもバンク予測を行わなくても，ともに 1 サイクルのペナルティを被るので，全ロード/ストア命令に対する予測精度（予測率と呼ぶ）が高い方が有利である．

図 13 に，P4 モデルにおける予測率の測定結果を示す（他のモデルでも同様の傾向）．各ベンチマークに 2 本の棒グラフがある．左がタグと状態を持つ通常のストライド予測器の場合であり，右がこれらを削除した簡易予測器の場合である．同図より分かるように，予測率はどのベンチマークでも簡易予測器の方がかなり大きい．予測率は平均で，通常の予測器で 57%であるのに対して，簡易予測器では 78%ある．

表 5 に，キャッシュを 4 ポートのワード・インターリーブとした場合の 1 エントリに必要なビット数を示す．32 ビット・アーキテクチャで，4 K エントリの表を仮定している．ADR は通常のストライド・アドレス予測器，BAN K は最終値とストライド・フィールドにバンクに関係する下位 4 ビットを保持する予測器，OUR は我々の提案するストライド・バンク予測器に対応する．同表から分かるように，我々の予測器のコストは，通常の予測器の 10%ほどしか要しない．4 K エントリの表のコストは，わずか 4 K バイトである．

表 5 ストライド・アドレス予測器とストライド・バンク予測器のコスト
Table 5 Cost of stride address predictor and stride bank predictors (bits).

scheme	tag	state	last value	stride	total
ADR	18	2	32	32	84
BANK	18	2	4	4	28
OUR	0	0	4	4	8

4.4 プロセッサ性能評価

本章では，従来のキャッシュを持つプロセッサに対し，PPC キャッシュを持つプロセッサがどれほど性能を改善するかを評価する．本評価では，クロック・サイクル時間と実行サイクル数の両方を考慮し，性能を比較する．クロック・サイクル時間について以下の仮定を置く．

- クロック・サイクル時間の下限として，**控えめな仮定**と**積極的な仮定**の 2 つの場合を想定する．各仮定におけるクロック・サイクル時間は，文献 19) より，それぞれ 1.08 CA8，0.54 CA8 とする．これらの数字については後で説明する．
- クロック・サイクル時間を決定するクリティカル・パスの遅延は，キャッシュ・アクセス以外については，与えられたクロック・サイクル時間の下限になっているとする．したがって，クロック・サイクル時間は，次の式で与えられる：

$$clockcycletime = max(givenlowerbound, cacheaccesstime/n)$$

ここで，n はキャッシュ・アクセスのパイプライン段数である．
- 積極的な仮定の場合は，控えめな仮定の場合に比べパイプライン段数が倍であるとする．この効果を分岐予測ミスペナルティに反映する．本測定では，控えめな仮定で 5 サイクル，積極的な仮定で 10 サイクルとする．

クロック・サイクル時間の下限の 2 つの仮定について説明する．文献 19) によると，クロック・サイクル時間の下限は，トレンド，SIA（Semiconductor Industry Association）の予測[23]，ISSCC（International Solid-State Circuits Conference）掲載の回路[24]などから，今後，8〜16 FO4 の間で推移すると予測している．ここで FO4 とは，半導体技術に独立な遅延の単位であり，360*Ldrawn (ps) という時間の定数である．ここで，Ldrawn (*m) は，与えられた半導体技術における最小ゲート長である．n FO4 とは 1 FO4 の n 倍の時間を意味する．ゲート遅延は Ldrawn にほぼ比例して短縮されるので，FO4 を単位とする遅延は，半導体技術に対し独立な数となる．4.1.2 項で書いたように 1 CA8 は 959 ps なので，クロック・サイ

表 6 クロック・サイクル時間下限の控えめな仮定におけるキャッシュ・アクセスのパイプライン段数とクロックサイクル時間（CA8）の関係
Table 6 Relationship between the number of pipeline stages and clock cycle time (CA8) under the conservative assumption of the lower bound of the clock cycle time.

モデル	パイプライン段数		
	従来		PPC
	1	2	1
P2	1.30	1.08	1.08
P4	1.36	1.08	1.08
P8	1.45	1.08	1.08

クル時間の下限は CA8 で表すと 0.54〜1.08 CA8 となる．本章では，下限の最小値と最大値の 2 つの場合を想定して評価を行う．以下では，最初に控えめな仮定の場合について，次に積極的な仮定の場合について評価する．

4.4.1 クロック・サイクル時間下限の控えめな仮定の場合

従来のキャッシュの場合，P2，P4，P8 の各プロセッサ・モデルにおけるキャッシュ・アクセス時間は，図 9 よりそれぞれ，1.30 CA8，1.36 CA8，1.45 CA8 である．これに対し，PPC キャッシュの場合，それぞれ，1.05 CA8，1.00 CA8，0.97 CA8 である．クロック・サイクル時間の控えめな仮定における下限は 1.08 CA8 であるから，従来のキャッシュの場合，どのプロセッサ・モデルにおいても，2 段までのパイプライン化ならばクロック・サイクル時間を低減できる．一方，PPC キャッシュの場合，キャッシュ・アクセス時間はどのプロセッサ・モデルにおいても下限より短いので，キャッシュ・アクセスは最も長いクリティカル・パスではない．よって，パイプライン化の必要はなく，どのモデルにおいても，クロック・サイクル時間は下限値である．**表 6** にキャッシュ・アクセスのパイプライン段数とクロック・サイクル時間の関係をまとめる．表において，下線を引いた数字は下限によって抑えられたクロック・サイクル時間である．以下では，キャッシュ・アクセスのパイプライン段数（従来，PPC）が，(1,1) および (2,1) の場合について評価を行う．なお，バンクを求める機構として，O，S，S+B，S+B+C の 4 つの機構に加え，バンクを完全に予測できる場合（Perfect）について測定した．

パイプライン段数 (1,1) の場合

図 14 に性能測定結果を示す．どのプロセッサ・モデルにおいても，クロック・サイクル時間の削減が大きく作用し，PPC キャッシュを用いた場合性能が大きく向上する．Perfect の場合では従来と PPC キャッシュの場合で IPC は変わらないので，性能差はクロック速度のみから生じる．表 6 に示したように，ポート数の多いキャッシュほどクロック・サイクル時間の削減は大きいので，より大きな性能向上率を達成している．クロック・サイクル時間の改善が少ない P2 においても，19.1〜20.2%（S+B+C 機構）の大きな性能向上を示す．一方で，ポート数が多いほど CBR は低いので，Perfect から実際の BNU にした場合の性能低下は大きい．それでもなお P8 においては，28.9〜32.8%（S+B+C 機構）の大きな性能向上を示している．

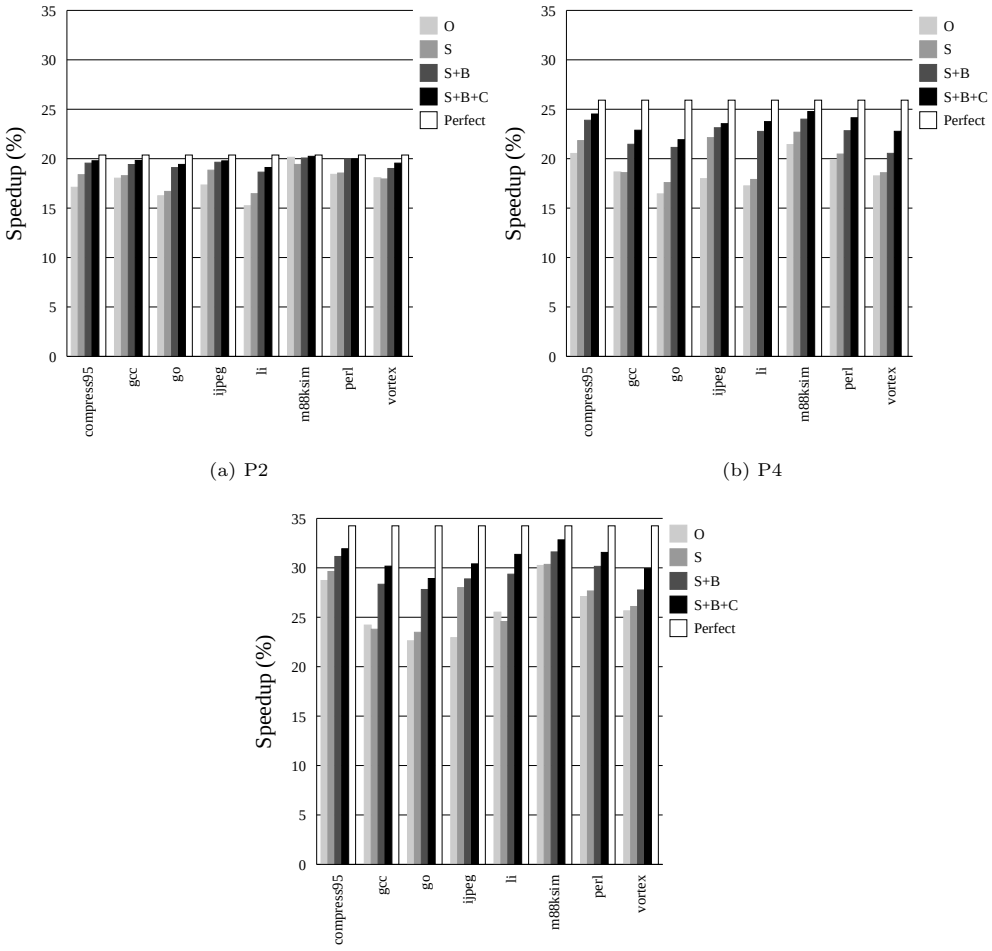
パイプライン段数 (2,1) の場合

従来キャッシュのパイプライン化によりクロック・サイクル時間に違いがなくなり，性能差は IPC の違いより生じる．**図 15** に測定結果を示す．

どのプロセッサ・モデルでも，(1,1) の場合に比べ性能向上率は小さくなっている．一般に，ポート数の少ないプロセッサでは，狭いキャッシュ・バンド幅によりメモリ・アクセスのスループットが低く抑えられ，ロード /ストア命令の長いレイテンシが他の命令の実行によって隠蔽される．このため，図 15 から分かるように，P2 では完全な BNU の場合で得られる性能向上率は最高で 7.0%，平均で 5.0%と大きくない．実際の BNU においては CBR が高いため，これに近い性能を達成しているが，2.1〜6.5%（S+B+C 機構）の改善にとどまっている．

逆に，多くのポートを持つプロセッサでは，長いレイテンシが露出され，性能に大きな影響を与える．完全な BNU において，P8 では最高 12.9%，平均 9.9%性能が向上する．しかし一方で，ポート数が多ければ CBR が減少するので，完全な BNU からの損失量は大きい．それでも P8 では最高 10.7%，平均 7.1%（S+B+C 機構）の性能向上を達成している．

同一のプロセッサ・モデルにおいて，各 BNU の機構に対する性能は，当然ながら，CBR が高い順となっている．また，正の性能向上を得るには，CBR にはかなり高い値が要求されることが分かる．ijpeg においては，O 機構ではどのプロセッサ・モデルでも性能向上を得ることができていない．O 機構の CBR は図 12 より，47〜78%であるが，この程度では不十分であることが分かる．S 機構を用いれば，すべてのベンチマークで性能向上を得ることができるが，P8 においても最高 7.4%，平均 3.7%にとどまる．これに発行時ブロードキャストを加えれば，性能は最高 9.5%，平均 5.9%にまで向上する．さらに C 機構を加えれば性能は前述の値，最高 10.7%，平均 7.1%にまで向上



(a) P2 (b) P4

(c) P8

図 14 キャッシュ・アクセスを従来の場合 PPC の場合ともに 1 サイクルで行った場合の性能向上
Fig. 14 Speedup with a single-cycle cache access in both cases of the conventional and the PPC cache.

する．3.3 節で述べたように，BNU に C 機構を組み込む場合は，クロック速度に悪影響を及ぼす可能性がある．この場合は，S+B 機構を組み込むのが最善といえる．

4.4.2 クロック・サイクル時間下限の積極的な仮定の場合

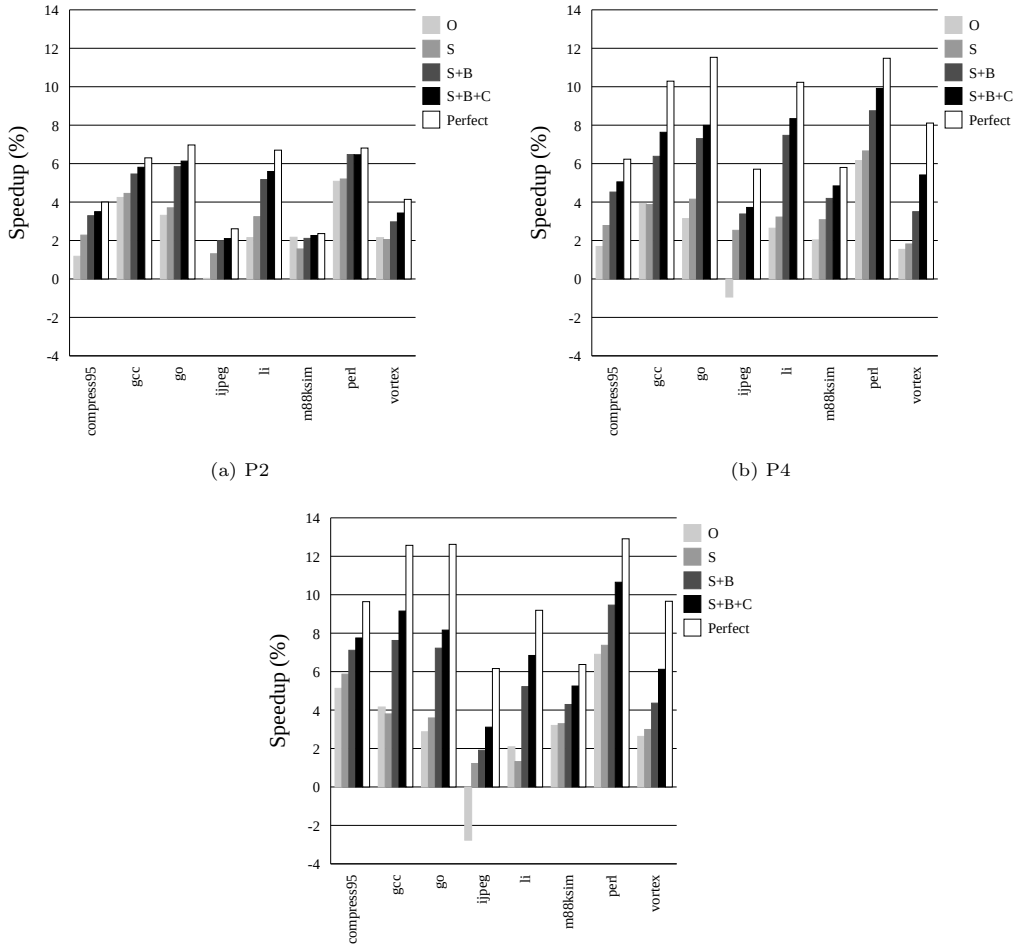
クロック・サイクル時間の下限は 0.54 CA8 と低いため，キャッシュのパイプラインを控えめな仮定の場合より深くし，クロック・サイクル時間を低減することができる．**表 7** にキャッシュ・アクセスのパイプライン段数とクロック・サイクル時間の関係をまとめる．下線を引いた数字は下限によって抑えられたクロック・サイクル時間である．

ここでは誌面の節約のため，まず，従来と PPC の両キャッシュの場合について，各パイプライン段数に

表 7 クロック・サイクル時間下限の積極的な仮定におけるキャッシュ・アクセスのパイプライン段数とクロックサイクル時間 (CA8) の関係
Table 7 Average performance normalized by the performance of the processor with non-pipelined conventional cache.

モデル	パイプライン段数				
	従来			PPC	
	1	2	3	1	2
P2	1.30	0.65	0.54	1.05	0.54
P4	1.36	0.68	0.54	1.00	0.54
P8	1.45	0.72	0.54	0.97	0.54

おけるベンチマークの平均性能を示し，その後に，平均性能が最も高いパイプライン段数のときの比較をベンチマークごとに示す．**表 8** に平均性能の測定結果を示す．最も性能の低い場合であるパイプライン化して



(a) P2

(b) P4

(c) P8

図 15 クロック・サイクル時間下限の控えめな仮定における性能向上率（キャッシュ・アクセスのパイプライン段数 (従来，PPC)=(2,1)）

Fig. 15 Speedup under the conservative assumption of the lower bound of the clock cycle time (pipeline stage of the cache access (conventional, PPC) = (2,1)).

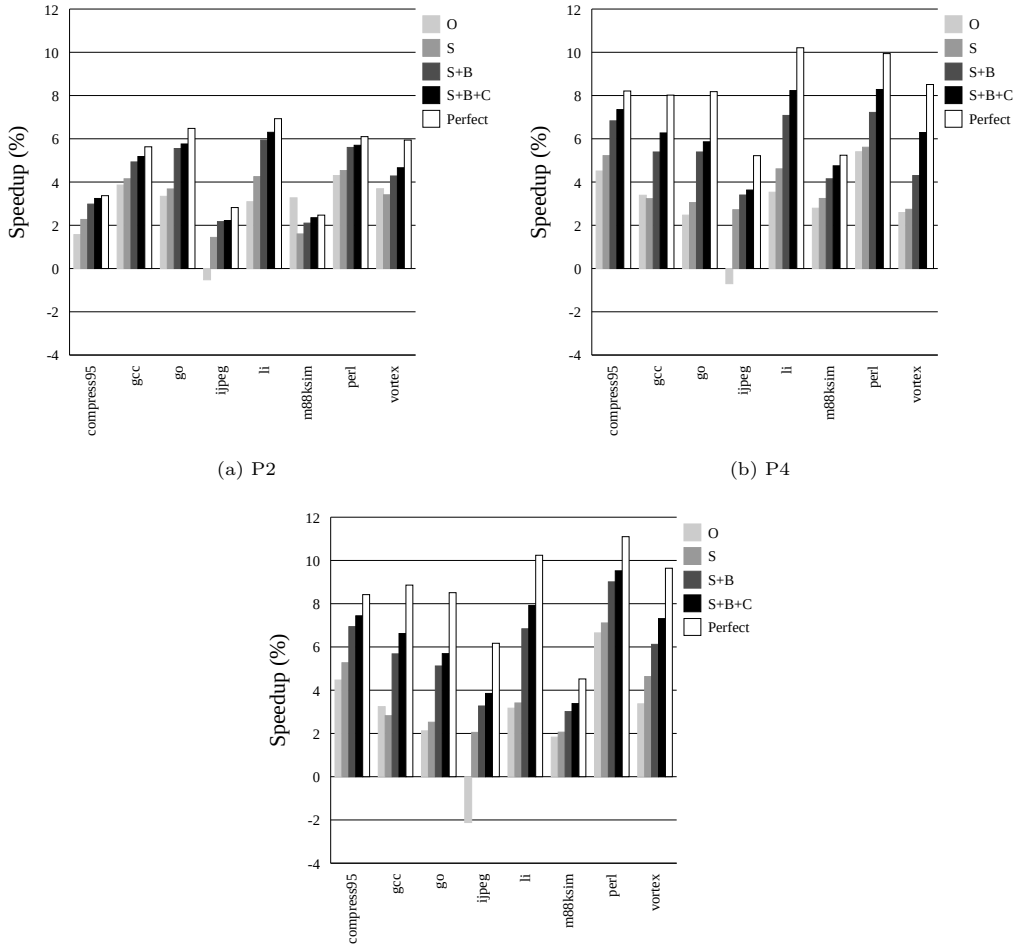
表 8 パイプライン化していない従来キャッシュを持つプロセッサの性能で規格化した平均性能

Table 8 Relationship between the number of pipeline stages and clock cycle time (CA8) under the aggressive assumption of the lower bound of the clock cycle time.

モデル	パイプライン段数				
	従来			PPC	
	1	2	3	1	2
P2	1.00	1.91	2.20	1.23	2.29
P4	1.00	1.87	2.18	1.34	2.32
P8	1.00	1.87	2.30	1.46	2.45

いない従来キャッシュを持つプロセッサの性能で正規化している．同表より，最も性能が高くなるパイプライン段数（従来，PPC）は，プロセッサ・モデルによらず (3,2) であることが分かる．

図 16 に，この場合において，PPC キャッシュを搭載したプロセッサの性能向上率をベンチマークごとに示す．クロック・サイクル時間は，従来の場合も PPC の場合も下限値となっているので，性能差は IPC の違いのみから生じる．一部の例外を除いて，どのモデルのどのベンチマークにおいても，控えめな仮定における評価のパイプライン (2,1) と同様の傾向を示している（P2，O 機構の m88ksim の性能が高いのは分岐アドレス予測精度が偶然高まったことによる）．これは IPC の違いによってのみ性能差が生じているからである．ただし，パイプラインは (3,2) なので，(2,1) の場



(a) P2

(b) P4

(c) P8

図 16 クロック・サイクル時間下限の積極的な仮定における性能向上率（キャッシュ・アクセスのパイプライン段数 (従来，PPC)=(3,2)）

Fig. 16 Speedup under the aggressive assumption of the lower bound of the clock cycle time (pipeline stage of the cache access (conventional, PPC)=(3,2)).

合よりレイテンシ削減が性能に与える影響が小さい．また，分岐予測ミス・ペナルティは 5 から 10 になっているので，これによっても影響は小さくなる．それでもなお，S+B+C 機構で P2 で 2.2〜6.3%，P4 で 3.6〜8.3%，P8 で 3.4〜9.5%の性能改善を達成している．

5. 関 連 研 究

Wilson らは，インターリーブ・キャッシュにおけるバンク競合を緩和し，バンド幅を拡大する**ライン・バッファ**と呼ぶ小さなバッファをロード /ストア・ユニットに付加することを提案した[25),26)]．ライン・バッファは，最近アクセスした少数のデータあるいはそれを含むブロックを保持する．キャッシュに対し読み出し要求を出す際に，同時にライン・バッファをアクセスする．バンク競合によりキャッシュが応答できない場合でも，ライン・バッファにデータがあれば，要求を満足することができる．Wilson らはさらに，ロード要求を待ち合わせるバッファに連想検索機構を設け，読み出されたデータを待ち合わせている他のロードにもデータを供給する **load all** 機構を提案した[25),26)]．ライン・バッファと load all を併用することにより，単一ポートのプロセッサにおいて，彼らが用いたベンチマーク・プログラムで，ロードの 37〜63%（ロード / ストアの 24〜52%）がキャッシュをアクセスせずに実行を完了することができるという評価結果を得ている．しかし依然として多くのキャッシュ・アクセスが残されており，アクセス時間短縮が必要である．

同様にバッファを用いてバンド幅を拡大するキャッ

シュ機構として，**LBIC**（locality-based interleaved cache）がある[4]．この機構では，各バンクにバッファを用意し，同一のキャッシュ・ブロックに行く要求を1つに結合することにより，要求量を削減する．この機構は，実効的にバンド幅を拡大することができるが，レイテンシを削除することはできない．

アドレスを予測することにより，データをプリフェッチする機構が多く研究されている（たとえば，文献14), 27)）．アドレス予測は値予測のサブセットであり，多くの研究がされている（たとえば，文献13)~17)）．アドレス予測以外にアドレスを得る方法として，Austinらはベース・レジスタを保持するキャッシュを用意することを提案している[28]．アドレス予測により得られるプロセッサの性能向上が，文献29) に詳細に測定されている．16命令発行で4ポート・データ・キャッシュを持つプロセッサに対する彼らの測定によると，予測誤りに対する方式として単純な無効化回復方式をとった場合，5%以下の性能向上しか得られない．再実行回復方式を採れば5〜10%の性能向上を得ることができるが，ハードウェアは複雑化する．この性能向上率はPPCキャッシュと同程度であるが，4.3節で述べたようにストライド・アドレス予測器は我々のストライド・バンク予測器より約10倍のコストを要する．

ロード命令のレイテンシを削減する機構として，Austinらは，実効アドレスの計算をXOR演算で代行するfast address calculation機構[30]を提案している．この技術はPPCキャッシュのアクセスにも利用でき，我々の研究とは直交する．

Choらは，静的な1つのメモリ参照命令は，実行時にはある1つのメモリ領域（グローバルかスタックかヒープ）を参照するという局所性に着目した[31]．領域ごとにメモリ・アクセスのパイプラインとキャッシュを持つことにより，複雑さの増加を抑制しつつキャッシュのバンド幅を増加させる方式を提案している．

Neefsらは，PPCキャッシュと同様の機構において，種々の値予測器をバンク予測に利用した場合の予測精度やプロセッサ性能を評価している[12]．しかし，バンク予測器を組み込んだ正確な評価や，キャッシュのアクセス時間から求められるクロック・サイクル時間とIPCの両方を考慮した総合的な評価は行っていない．

6. ま と め

本論文では，マルチポート・キャッシュのレイテンシを削減するPPCキャッシュと呼ぶ機構を評価した．PPCキャッシュは，インターリーブ・キャッシュの各バンクを1対1でロード/ストア・ユニットと結合し，従来必要であった複雑な相互結合網を取り除き，アクセスを小さなバンクに限定する．これによりレイテンシを大幅に削減することができる．この構成では，命令の発行前にどのバンクにアクセスするかを求める必要がある．この機能を果たすBNUと呼ぶユニットに，予測を含む種々の方式を示した．

PPCキャッシュの有効性を確認するために，回路レベルおよびSPECint95を用いたアーキテクチャ・レベルでのシミュレーションによる評価を行った．その結果，最も高い性能を示すBNUと命令発行方式の場合で，8ポート・インターリーブ32Kバイトのキャッシュを持つ16命令発行のプロセッサにおいて，実効アクセス時間を平均22%削減できることが分かった．またこの場合，平均83%の割合で命令を正しいロード/ストア・ユニットに発行できることが分かった．さらに，従来のキャッシュとPPCキャッシュを搭載したプロセッサの性能を，クロック・サイクル時間と実行サイクル数のトレードオフを考慮し，それぞれ最善のキャッシュ・パイプラインにおいて比較した．その結果，32Kバイト，8ポートのデータ・キャッシュを持つ16命令発行のプロセッサにおいて，クロック・サイクル時間の下限を控えめに見積もった場合，最大11%，平均7%の性能向上を，クロック・サイクル時間の下限を積極的に見積もった場合，最大10%，平均6%の性能向上を達成できることを確認した．

謝辞 本研究の一部は，文部省科学研究費補助金基盤研究（C）課題番号1068034，同じく11680351，財団法人大川情報通信基金の支援を受けている．

参 考 文 献

1) Digital Equipment Corporation: *Alpha 21164 Microprocessor Hardware Reference Manual* (1995).
2) L. Gewnnap: Digital 21264 Sets New Standard, *Microprocessor Report*, Vol.10, No.14, pp.11–16 (1996).
3) Intel Corporation: *Pentium (R) Processor Family Developer's Manual Vol.1* (1995).
4) Rivers, J.A., Tyson, G.S., Davidson, E.S. and Austin, T.M.: On High-Bandwidth Data Cache Design for Multi-Issue Processors, *Proc. 30th Int. Symp. on Microarchitecture*, pp.46–56 (1997).
5) Advanced Micro Devices Inc.: *AMD Athlon Processor Technical Brief* (1999).
6) Yoaz, A., Erez, M., Ronen, R. and Jourdan, S.: Speculation Techniques for Improving Load Related Instruction Scheduling, *Proc. 26th Int. Symp. on Computer Architecture*, pp.42–53

(1999).
7) 嶋田 創, 安藤秀樹, 島田俊夫：クロスバスイッチをなくしたマルチバンクキャッシュ, 情報処理学会研究報告, 99-ARC-135, pp.75–80 (1999).
8) 嶋田 創, 安藤秀樹, 島田俊夫：クロスバスイッチをなくしたマルチバンクキャッシュ, 2000 年並列処理シンポジウム JSPP2000, pp.107–114 (2000).
9) Wada, T., Rajan, S. and Przybylski, S.A.: An Analytical Access Time Model for On-Chip Cache Memories, *IEEE Journal of Solid-State Circuits*, Vol.27, No.8, pp.1147–1156 (1992).
10) Reinman, G. and Jouppi, N.: Extensions to CACTI. Unpublished document.
11) Wilton, S.J.E. and Jouppi, N.P.: An Enhanced Access and Cycle Time Model for On-Chip Caches, WRL Research Report 93/5 (1994).
12) Neefs, H., Vandierendonck, H. and Bosschere, K.D.: A Technique for High Bandwidth and Deterministic Low Latency Load/Store Access to Multiple Cache Banks, *Proc. 6th Int. Symp. on High-Performance Computer Architecture*, pp.313–324 (2000).
13) Lipasti, M.H., Wilkerson, C.B. and Shen, J.P.: Value Locality and Load Value Prediction, *Proc. 7th Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, pp.136–147 (1996).
14) Chen, T.F. and Baer, J.L.: Effective Hardware-based Data Prefetching for High Performance Processors, *IEEE Trans. Comput.*, Vol.44, pp.609–623 (1995).
15) Gabbay, F. and Mendelson, A.: The Effect of Instruction Fetch Bandwidth on Value Prediction, *Proc. 25th Int. Symp. on Computer Architecture*, pp.133–143 (1997).
16) Wang, K. and Franklin, M.: Highly Accurate Data Value Prediction using Hybrid Predictors, *Proc. 30th Int. Symp. on Microarchitecture*, pp.281–290 (1997).
17) Sazeides, Y. and Smith, J.E.: The Predictability of Data Values, *Proc. 30th Int. Symp. on Microarchitecture*, pp.248–258 (1997).
18) Weste, N.H.E. and Eshraghian, K. : *Principles of CMOS VLSI Design, A System Perspective*, Second Edition, Addison Wesley (1993).
19) Agarwal, V., Hrishikesh, M.S., Kechler, S.W. and Burger, D.: Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures, *Proc. 27th Int. Symp. on Computer Architecture*, pp.248–259 (2000).
20) Amrutur, B. and Horowitz, M.A.: Speed and Power Scaling of SRAM, *IEEE Journal of Solid-State Circuits*, Vol.35, No.2, pp.175–185 (2000).
21) Burger, D. and Austin, T.M.: The SimpleScalar Tool Set, Version 2.0, Technical Report CS-TR-97-1342, University of Wisconsin-Madison Computer Sciences Department (1997).
22) McFarling, S.: Combining Branch Predictors, Technical Report TN-36, Digital Equipment Corporation Western Resarch Laboratory (1993).
23) Semiconductor Industry Association: *The National Technology Roadmap for Semiconductors* (1999).
24) Naffziger, S.: Subnanosecond 0.5*m 64b Adder Design, *International Solid-State Circuits Conference*, pp.362–363, In Digest of Technical Papers (1996).
25) Wilson, K.M., Olukotun, K. and Rosenblum, M.: Increasing Cache Port Efficiency for Dynamic Superscalar Microprocessors, *Proc. 23rd Int. Symp. on Computer Architecture*, pp.147–157 (1996).
26) Wilson, K.M. and Olukotun, K.: High Bandwidth On-Chip Cache Design, *IEEE Trans. Comput.*, Vol.50, No.4, pp.292–307 (2001).
27) Farkas, K.I., Chow, P., Jouppi, N.P. and Z. Vranesic: Memory-System Design Consideration for Dynamically-Scheduled Processors, *Proc. 24th Int. Symp. on Computer Architecture*, pp.133–143 (1997).
28) Austin, T.M. and Sohi, G.S.: Zero-Cycle Loads: Microarchitecture Support for Reducing Load Latency, *Proc. 28th Int. Symp. on Microarchitecture*, pp.82–92 (1995).
29) Reinman, G. and Calder, B.: Predictive Techniques for Aggressive Load Speculation, *Proc. 31st Int. Symp. on Microarchitecture*, pp.127–137 (1998).
30) Austin, T.M., Pnevmatikatos, D.N. and Sohi, G.S.: Streaming Data Cache Access with Fast Address Calculation, *Proc. 22nd Int. Symp. on Computer Architecture*, pp.360–380 (1995).
31) Cho, S., Yew, P-C. and Lee, G.: Decoupling Local Variable Accesses in a Wide-Issue Superscalar Processor, *Proc. 26th Int. Symp. on Computer Architecture*, pp.100–110 (1999).

(平成 13 年 5 月 1 日受付)
(平成 13 年 8 月 16 日採録)

嶋田 創（学生会員）

1976年生．1998年名古屋大学工学部情報工学科卒業．2000年名古屋大学大学院工学研究科情報工学専攻博士課程前期課程修了．現在，名古屋大学大学院工学研究科電子情報学専攻博士課程後期課程在学中．計算機アーキテクチャの研究に従事．

安藤 秀樹（正会員）

1959年生．1981年大阪大学工学部電子工学科卒業．1983年大阪大学大学院修士課程修了．京都大学工学博士．1983年三菱電機（株）LSI研究所．ISDN用ディジタル信号処理LSI，第5世代コンピュータ・プロジェクトの推論マシン用プロセッサの設計に従事．1991年Stanford大学客員研究員．1997年名古屋大学大学院工学研究科電子情報学専攻講師．1998年同大学助教授．1998年東京大学大学院理学系研究科助教授併任．1998年情報処理学会論文賞受賞．計算機アーキテクチャ，コンパイラの研究に従事．

島田 俊夫（正会員）

1968年東京大学工学部計数工学科卒業．1970年東京大学大学院修士課程修了．同年電子技術総合研究所入所．1993年より名古屋大学大学院工学研究科電子情報学専攻教授．人工知能向き言語，LISPマシン，データフロー計算機の研究に従事．最近はマイクロプロセッサのアーキテクチャやチップ内並列処理の研究を行っている．1988年度市村賞，1994年度情報処理学会論文賞，1995年度注目発明賞受賞．工学博士．