可変構造型並列計算機のキャッシュ・アーキテクチャ†

——キャッシュ構成およびキャッシュ内コヒーレンス処理——

岩田英次†† 森眞一郎†† 村上和彰††
福田晃†† 富田眞治††

本論文は，『可変構造型並列計算機』の各プロセッシング・エレメント (PE) に設けたキャッシュの構成，および，仮想アドレス・キャッシュで問題となる同義語問題の解決法（キャッシュ内コヒーレンス処理）について述べている．本システムは，128 台の RISC プロセッサを 128×128 のクロスバー網で相互結合した MIMD 型のマルチマイクロプロセッサ・システムである．各 PE に分散配置したローカルメモリに対して他 PE からも相互結合網経由で直接アクセス可能とするローカル/リモート・アーキテクチャ（分散共有メモリ構成）を採用する．リモートアクセスの遅延を隠蔽し，かつ，ローカルメモリからでも待ちサイクルなしで命令/データを供給するため，高速大容量のダイレクト・マッピング方式の仮想アドレス・キャッシュを各 PE に設けた．しかし，仮想アドレス・キャッシュは，キャッシュ間コヒーレンス（マルチキャッシュ・コンシステンシ）に加えて，キャッシュ内コヒーレンス（同義語発生防止）も保証する必要がある．これらのコヒーレンスを保証するため，タグアレイを仮想および実アドレス対応に 2 個設けるデュアル・ディレクトリ構成とした．さらに，キャッシュ内コヒーレンス処理で生じる“衝突”の影響を軽減するため，ダイレクト・マッピング方式の仮想アドレス・タグアレイに対して，実アドレス・タグアレイは 4 ウェイのセット・アソシアティブ方式とした．

1. はじめに

我々は，実行すべき並列アルゴリズムの構造に対して，柔軟に計算機構成を適応させる『可変構造型並列計算機』の開発を進めている[1),2)]．本システムは，128 台のプロセッシング・エレメント (PE) を 128×128 のクロスバー網で相互結合した MIMD 型のマルチマイクロプロセッサ・システムであり，相互結合網およびメモリを次のように可変構造化した“ダイナミック・アーキテクチャ”を採用している．

①可変構造型ネットワーク・アーキテクチャ：プロセッサ-プロセッサ結合およびプロセッサ-メモリ結合として任意の結合形態を提供する．

②可変構造型メモリ・アーキテクチャ：プロセスおよびプロセッサに対して任意のメモリ・イメージを提供する．

可変構造型ネットワークは，相互結合網として非閉塞網であるクロスバー網を採用したことで実現している[3)]．また，可変構造型メモリは次のように実現した．まず，プロセスに対して任意の論理メモリ・イメージ（プロセスから見た記憶空間構成）を与えるために，仮想記憶機構を備えた．一方，プロセッサに対して任意の物理メモリ・イメージ（プロセッサから見た記憶空間構成）を与えるために，“ローカル/リモート・アーキテクチャ（分散共有メモリ構成）”を採用した[4)]．これは，各 PE に分散配置したローカルメモリ（容量 8 MB）を自 PE に単に専有させるのではなく，他 PE からもクロスバー網を経由してアドレスにより直接アクセス可能とする方式である．ある PE のプロセッサから見たとき，クロスバー網を経由してアドレスにより直接アクセス可能な他 PE のローカルメモリを“リモートメモリ”と呼ぶ．

各 PE には，大容量 (128 KB) かつ高速（アクセスおよびサイクル時間 60 ns）なダイレクト・マッピング方式の仮想アドレス・キャッシュを設けた．これは，プロセッサとして SPARC RISC プロセッサ（動作周波数 16.67 MHz）を用いており，その性能を最大限に引き出すためには待ちサイクルなしで命令およびデータを供給する必要があるからである．また，リモートメモリ・アクセスはローカルメモリ・アクセスに比べて 10 倍ほど時間を要することから[3)]，このアクセス遅延を隠蔽するためにもキャッシュは不可欠である．

キャッシュを設けたことで，コヒーレンス (coherence)/コンシステンシ (consistency) 問題が生じる[10)]．本システムでは，以下の 3 種類のコヒーレンスを保証する必要がある[5),6),8)]．

† Cache Architecture of the Kyushu University Reconfigurable Parallel Processor —Cache Design and Intra-cache Coherence Protocol— by EIJI IWATA, SHIN-ICHIRO MORI, KAZUAKI MURAKAMI, AKIRA FUKUDA and SHINJI TOMITA (Department of Information Systems, Interdisciplinary Graduate School of Engineering Sciences, Kyushu University).

†† 九州大学大学院総合理工学研究科情報システム学専攻

①キャッシュ内コヒーレンス：仮想アドレス・キャッシュでは多義語および同義語（synonym）が生じ得るので，これらの発生を抑止する．

②キャッシュ—メモリ間コヒーレンス：キャッシュ内容とメモリ（ローカルおよびリモート）内容とを一致させる．

③キャッシュ間コヒーレンス：あるメモリ・ブロックのコピーが任意のPEのキャッシュに存在し得ることから，複数キャッシュ間での一貫性（multicache consistency）をとる．

本論文では，本キャッシュの構成，および，上記①のキャッシュ内コヒーレンス処理について述べる．なお，上記②③については，別論文として報告するつもりである．まず第2章で，本キャッシュの設計方針，特にダイレクト・マッピング方式の仮想アドレス・キャッシュを採用した根拠，多義語および同義語問題の解決方法，についてまとめる．第3章では，デュアル・ディレクトリを採用した本キャッシュの構成および動作を述べる．第4章で，キャッシュ内コヒーレンス処理，および，その際に生じる“衝突”について説明する．そして第5章で，衝突の影響を軽減するための方法を示した後，その効果を確率的解析およびシミュレーション解析で検証する．

2. 設計方針

可変構造型並列計算機をそのキャッシュ・システムに着目した場合の構成を図1に，また，キャッシュの諸元を表1にそれぞれ示す．

キャッシュの設計にあたっては，次の項目について検討を行った[9]．

①アクセス・アドレス：仮想アドレス・キャッシュ vs 実アドレス・キャッシュ

②連想度：ダイレクト・マッピング vs セット・アソシアティブ

③ブロック・サイズ

④キャッシュ内コヒーレンス処理

⑤キャッシュ—メモリ間コヒーレンス処理（メモリ更新アルゴリズム）

⑥キャッシュ間コヒーレンス処理

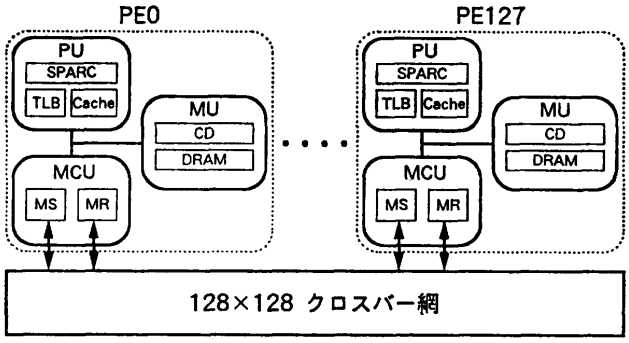


図1 可変構造型並列計算機のキャッシュ・システム
Fig. 1 Cache system of the Kyushu University Reconfigurable Parallel Processor.

表1 キャッシュの諸元
Table 1 Cache specifications.

		初期仕様	現仕様
キャッシュ	アクセス・アドレス	仮想アドレス／実アドレス	仮想アドレス
	Unified／Split	Unified（単一キャッシュ）	
	連想度	2	1（ダイレクト・マッピング）
	メモリ更新アルゴリズム	ストア・スルー	
	リプレースメント・アルゴリズム	LRU（Least Recently Used）	
	アドレス・ブロック・サイズ	32B	32B／64B
	トランスファ・ブロック・サイズ	8B	32B／64B
	コヒーレンス・ブロック・サイズ	32B	32B／64B
	セット数	1024／128	2048
	キャッシュ・サイズ	64KB／8KB	64KB／128KB
RTB	連想度	2	4
	リプレースメント・アルゴリズム	LRU	
	セット数	1024	512

上記①～④に関して，以下の設計方針を採った．

2.1 アクセス・アドレスおよび連想度

2.1.1 選 択 肢

仮想記憶機構を備える場合，キャッシュをアクセスするのに仮想アドレスを用いるか実アドレスを用いるかで，次の2つの選択肢が存在する[10]．

- 仮想アドレス・キャッシュ
- 実アドレス・キャッシュ

実アドレス・キャッシュでは，キャッシュ・アクセスに際して仮想アドレスを実アドレスへ変換しなければならない．このアドレス変換（TLB（アドレス変換バッファ）検索）とキャッシュ・アクセス（アドレスタグ読出し）との前後関係に関して，次の3つの選択肢が存在する．

- プリアクセス：アドレス変換が完了する前に，タグ比較に用いるアドレスタグの読出しを済ませる．
- パラレルアクセス：アドレス変換と並行して，タグ比較に用いられる可能性のあるアドレスタグ候補をあらかじめ読み出しておく．そして，アドレス変換が完了して実アドレスが判明した時点で，これら候補の中から真にタグ比較に用いるアドレスタグを選択する．
- ポストアクセス：アドレス変換によって実アドレスが判明した後，アドレスタグの読出しを開始する．

さらに，キャッシュの連想度（ウェイ数）を1とするか2以上とするかで，次の2つの選択肢が存在する[10),12)]．

- ダイレクト・マッピング
- セット・アソシアティブ

以上から，キャッシュのアクセス・アドレスおよび連想度に関して，次の8つの選択肢が存在することになる．

①V（仮想アドレス）+D（ダイレクト・マッピング）
②V（仮想アドレス）+S（セット・アソシアティブ）
③R/Pre（実アドレス/プリアクセス）+D（ダイレクト・マッピング）
④R/Pre（実アドレス/プリアクセス）+S（セット・アソシアティブ）
⑤R/Para（実アドレス/パラレルアクセス）+D（ダイレクト・マッピング）
⑥R/Para（実アドレス/パラレルアクセス）+S（セット・アソシアティブ）
⑦R/Post（実アドレス/ポストアクセス）+D（ダイレクト・マッピング）
⑧R/Post（実アドレス/ポストアクセス）+S（セット・アソシアティブ）

2.1.2 考 察

これら8つの選択肢は，以下の項目に対して影響を与える（**表2**参照）．

（1） キャッシュ・アクセス時間

タグ・アレイとデータ・アレイへは同時にアクセスするものとする．また，タグ読出し時間（T_{tag}）+タグ比較時間（T_{comp}）の方がデータ読出し時間よりも長いと仮定する．

表2に示すように，同じ連想度なら仮想アドレス（V），実アドレス/プリアクセス（R/Pre），実アドレス/パラレルアクセス（R/Para），実アドレス/ポストアクセス（R/Post）の順にアクセス時間が長くなる．また，同じアクセス・アドレスでは，セット・アソシアティブ（S）の方がダイレクト・マッピング（D）より

表2 アクセス・アドレスおよび連想度に着目したキャッシュ構成方式の比較
Table 2 Various cache configuration regarding the access-address and associativity.

方式	アクセス・アドレス	アドレス変換との前後関係	連想度（ウェイ数）	キャッシュ・アクセス時間	キャッシュ・サイズ	ハードウェア・コスト
V+D	仮想アドレス（V）		1（D）	$T_{tag}+T_{comp}$	L×S	H_{way}
V+S			2以上（S）	$T_{tag}+T_{comp}+T_{sel}$	L×S×W	$H_{way}\times W+H_{sel}(W)$
R/Pre+D	実アドレス（R）	プリアクセス（Pre）	1（D）	$\max\{T_{tag},T_{TLB}\}+T_{comp}$	L×S（≦P）	$H_{TLB}+H_{way}$
R/Pre+S			2以上（S）	$\max\{T_{tag},T_{TLB}\}+T_{comp}+T_{sel}$	L×S×W（≦P×W）	$H_{TLB}+H_{way}\times W+H_{sel}(W)$
R/Para+D		パラレルアクセス（Para）	1（D）	$\max\{T_{tag},T_{TLB}\}+T_{mpx}+T_{comp}$	L×S（≒P×SS）	$H_{TLB}+H_{way}+H_{mpx}(SS)$
R/Para+S			2以上（S）	$\max\{T_{tag},T_{TLB}\}+T_{mpx}+T_{comp}+T_{sel}$	L×S×W（≒P×SS×W）	$H_{TLB}+(H_{way}+H_{mpx}(SS))\times W+H_{sel}(W)$
R/Post+D		ポストアクセス（Post）	1（D）	$T_{TLB}+T_{tag}+T_{comp}$	L×S	$H_{TLB}+H_{way}$
R/Post+S			2以上（S）	$T_{TLB}+T_{tag}+T_{comp}+T_{sel}$	L×S×W	$H_{TLB}+H_{way}\times W+H_{sel}(W)$

T_{tag}：タグ読み出し時間　S：セット数
T_{comp}：タグ比較時間　L：ライン・サイズ（バイト）
T_{mpx}：タグ選択時間　W：ウェイ数
T_{TLB}：TLB検索時間　P：ページ・サイズ（バイト）
T_{sel}：ウェイ選択時間　SS：スーパーセット当たりのセット数

H_{way}：キャッシュのウェイ当たりのハードウェア・コスト
$H_{sel}(W)$：ウェイ選択のためのハードウェア・コスト（Wの関数）
H_{TLB}：TLBのハードウェア・コスト
$H_{mpx}(SS)$：セット選択のためのハードウェア・コスト（SSの関数）

もウェイ選択時間 (T_{sel}) だけアクセス時間が長い.

(2) キャッシュ・サイズ

キャッシュ・サイズは次式で表される.

ウェイ当りの容量×ウェイ数

=ライン・サイズ×セット数×ウェイ数

ライン・サイズ(L)を固定した場合，キャッシュ・サイズを大きくするにはセット数(S)を増やす方法とウェイ数(W)を増やす方法の2通りがある．いずれを採るかは，ハードウェア・コストとヒット率とのトレードオフにより決まる.

仮想アドレス(V)および実アドレス/ポストアクセス (R/Post) には，ウェイ当りの容量に関する制限がない．よって，セット数を増やすことでキャッシュ・サイズを容易に大きくできる．一方，実アドレス/プリアクセス (R/Pre) は，ウェイ当りの容量をページ・サイズ (本システムでは 4KB) 以上に大きくできない．これは，実アドレスが判明する前にセットを特定しなければならず，キャッシュ・アクセスにページ内オフセット (仮想アドレスと実アドレスとで共通する部分) しか使えないからである．したがって，キャッシュ・サイズをページ・サイズ以上に大きくするにはウェイ数を増やすしかない．実アドレス/パラレルアクセス (R/Para) では，この制約を解消している.

(3) ハードウェア・コスト

実アドレス・キャッシュ(R)では，TLB の装備が前提となる．しかし，仮想アドレス・キャッシュ(V)でもミスヒット時にはアドレス変換を行うので，その高速化のためには TLB が必要となる.

キャッシュ自体のハードウェア・コストに関しては，まずセット・アソシアティブ(S)ではウェイ選択のためのハードウェア (H_{sel}) が必要である．そのコストはウェイ数に比例して大きくなる．また，タグ比較器もウェイ数分必要である．よって，同一キャッシュ・サイズのダイレクト・マッピング(D)に比べて，セット・アソシアティブ(S)の方がコストが高い.

また，実アドレス/パラレルアクセス (R/Para) では，同一ページ内オフセットでアクセスされる複数セット (この集合をスーパセットと呼ぶ[14]) から同時にアドレスタグ候補を読み出せるように，タグ・アレイをマルチバンク化する必要がある．バンク数はスーパセット当りのセット数に等しい．さらに，実アドレス判明後にスーパセットから1個のセットを選択するためのハードウェア (H_{mpx}) が必要となる．そのコストはスーパセット当りのセット数に比例して大きくなる．なお，タグ・アレイとデータ・アレイとを同時にアクセスする場合，データ・アレイに対しても同様の措置が必要である.

(4) ヒット率

同一キャッシュ・サイズであれば，連想度が増すにつれてヒット率が向上する.

また，実アドレス・キャッシュ(R)では，TLB ヒットがキャッシュ・ヒットの前提条件となるため，キャッシュ・ヒット率が TLB ヒット率に依存する.

(5) 多義語および同義語問題

仮想アドレス・キャッシュ(V)では，多義語および同義語問題を解決する必要がある (2.3 節参照).

2.1.3 選択肢の決定

2.1.1 項で定義した8つの選択肢の中で，まず R/Pre+D ではキャッシュ容量が 4KB しかなく不十分あり，また，R/Post では SPARC プロセッサ (マシンサイクル 60 ns) に対して待ちサイクルなしで命令およびオペランドを供給することが不可能である．残りの5つの選択肢 V+D, V+S, R/Pre+S, R/Para+D, R/Para+S は，アクセス時間およびキャッシュ・サイズともに要件を満たしている.

ヒット率に関しては，同一キャッシュ・サイズであれば連想度が増すにつれてヒット率が向上する．しかし，大容量のキャッシュ (本システムでは 64KB または 128KB) では連想度のヒット率への影響は小さいことが報告されている[12),13)]．さらに，セット・アソシアティブ(S)化によるハードウェア・コストの増加はかなり大きい．また，パラレルアクセス (Para) 化も同程度のコスト増加を招く.

以上から，コスト/パフォーマンスを重視して，V+D (仮想アドレス+ダイレクト・マッピング) を選択した．ただし，2.3 節で述べるように，多義語および同義語問題への対応が必要となった.

2.2 ブロック・サイズ

キャッシュ管理の単位となるブロックには，以下の3種類がある[14)].

①アドレス・ブロック (AB; ライン): アドレスタグに対応する単位.

②トランスファ・ブロック(TB): キャッシュ—メモリ間またはキャッシュ間における転送単位.

③コヒーレンス・ブロック(CB): キャッシュ—メモリ間およびキャッシュ間の一貫性を保証する単位.

これらのブロック間の大小関係は任意に定められ

る．本キャッシュの初期仕様においては，

TB(8B)＜AB(32B)＝CB(32B)

としていた（表1参照）．これにより，以下の効果をねらった[6]．

- TB＜AB とすることで，リモート・メモリからのクロスバー網を経由するライン・フェッチにおいて，TB の未着を許す（回線中途切断への対処）．その結果1個の AB 内部で TB の不在が起こり得て，次のライン・フェッチ時には当該 TB のみを転送することで転送時間を短縮する．
- AB＝CB とすることで，コヒーレンス処理の際のタグ・アレイへのアクセス回数を1回のみとする．

しかし，TB＜AB とした場合，TB 対応に存在ビットが必要となり，さらに，キャッシュ・ミス時の処理が煩雑となる．よって，現仕様では，一般的なキャッシュと同様，

TB＝AB＝CB

としている．

このとき，ブロック・サイズ（ライン・サイズ）の大小によるヒット率の変動はプログラム参照パターンに依存し一概に特定できないが，16～64 B 程度が望ましいとの報告がある[11]．そこで，3.1 節で述べるように，ライン・サイズとしては，32 B および 64 B を選択可能としている．

2.3 キャッシュ内コヒーレンス処理

仮想アドレス・キャッシュを用いる場合，以下の2つの問題を解決する必要がある．

（1） 多義語問題

多重仮想空間環境下では，同じ仮想アドレスでも仮想空間が異なっていれば一般に異なる実アドレスと対応する（多義語）．よって，仮想アドレス・キャッシュでは，多義語の仮想アドレスにより誤ってヒットすることが起こる．これを防ぐには，次の2つの方法がある．

①仮想空間スイッチ時にキャッシュをフラッシュする．

②アドレスタグに仮想空間識別子を持たせ，これで仮想空間を識別する．

方法①は，小容量キャッシュ，あるいは，空間スイッチの間隔が十分長い場合に有効である．しかし，本システムのような大容量キャッシュにおいては，空間スイッチが頻繁に起きる場合のヒット率低下は無視できない．

そこで，①と②の両方法を採用している．すなわち，キャッシュのフラッシュ・コマンドを用意するとともに，8ビットの仮想空間識別子（空間 ID）をアドレスタグに含め256空間まで識別可能としている．空間 ID の管理は OS が行う．また，空間スイッチ時にキャッシュをフラッシュするか否かも OS 次第である．

（2） 同義語問題

あるデータを共有する場合，異なる仮想アドレスが同じ実アドレスに対応することがある．これら仮想アドレスを同義語と呼ぶ．このとき仮想アドレス・キャッシュでは，当該共有データのコピーがキャッシュ内に複数存在し得るので，これらコピー間の一貫性を保証する必要がある．

この同義語問題は，キャッシュ間コヒーレンスと類似している．すなわち，後者が複数キャッシュに存在する同一メモリ・ブロックのコピー間の一貫性を保証するのに対し，前者は1キャッシュ内に存在する同一メモリ・ブロックの複数コピー間の一貫性を保証する．

同義語問題の解決策としては，次の2つの方法がある[10]．

①同義語が生じないよう，データの共有形態に制限を設ける．すなわち，1仮想空間内でのデータ共有を禁止する．また，異なる仮想空間間でデータを共有する場合，その仮想アドレスは同一にする（異なる場合，空間スイッチ時にキャッシュをフラッシュ）．

②実アドレスを仮想アドレスに変換する逆変換バッファ（RTB）を設ける．これにより，実アドレスによっても仮想アドレス・キャッシュへのアクセスを可能とし，対応するメモリ・ブロックのコピー間の一貫性を保証する．

また，動的なキャッシュ間コヒーレンス処理においても，実アドレスから仮想アドレスへの逆変換が必要なため RTB は不可欠である[6]．したがって，同義語問題の解決策としても，RTB を用いた方法②を採ることができる[14]．よって，方法②を採用する．これに基づくキャッシュ内コヒーレンス処理の詳細は4章で述べる．

なお，方法①は純粋に OS の構成および制御に依存しており，方法①を採るか否かは OS 次第である．

3. キャッシュの構成と動作

3.1 デュアル・ディレクトリ

キャッシュの構成を図2に示す．2.1 節で述べたように，ダイレクト・マッピング方式の仮想アドレス・

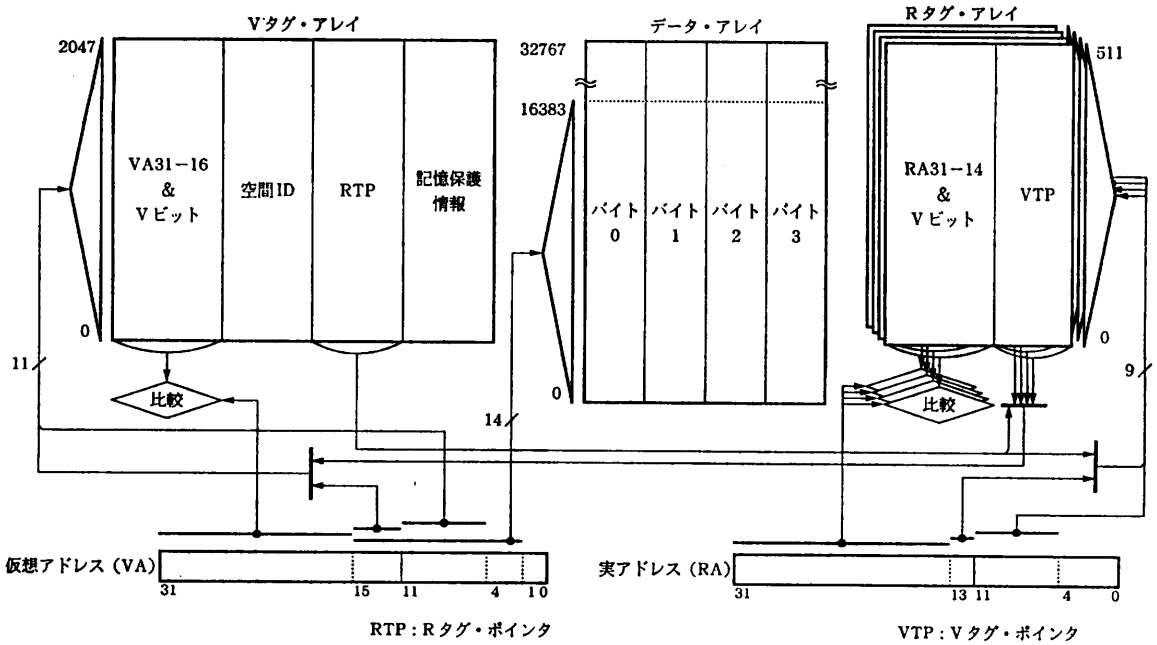


図 2 キャッシュの構成（ライン・サイズが32B の場合）
Fig. 2 Cache organization (Line size=32 B).

キャッシュである．また，2.3 節で述べたように，コヒーレンス処理の際に実アドレスでもアクセスできるように RTB を設けている．すなわち，次の 2 個のディレクトリを有するデュアル・ディレクトリ・キャッシュ[14]構成となっている．

①Vタグ・アレイ：仮想アドレスタグ等を保持するディレクトリ．

②Rタグ・アレイ：実アドレスタグ等を保持するディレクトリ (RTB).

また，2.2 節で述べたように，ライン・サイズとして 32 B および 64 B のいずれも選択可能であるが，説明を簡単にするため以下 32 B の場合について述べる．

（1） Vタグ・アレイ

セット数 2048 のダイレクト・マッピング方式のディレクトリである．通常は 32 ビット長の仮想アドレス (VA 31-0) 中の 11 ビットのセット・アドレス (VA 15-5) でインデックスする．このうち 4 ビット (VA 15-12) は仮想ページ番号である．

Vタグ・アレイの各エントリ（Vタグ・エントリ）は，以下の 4 つのフィールドからなる．

①仮想アドレスタグ（Vタグ）：アクセス・アドレスとの比較に用いる上位仮想アドレス (VA 31-16)，および，ライン有効(V)ビット．

②空間 ID：2.3 節で述べた 8 ビットの仮想空間識別子．

③Rタグ・ポインタ (RTP)：①の仮想アドレスタグと対応関係にある実アドレスタグを登録しているRタグ・エントリへのポインタ．Rタグ・アレイのセットを指定する実ページ番号の下位 2 ビット (RA 13-12)，および，当該セット内のウェイ番号から成る．

④記憶保護：ページテーブル・エントリの記憶保護フィールドのコピーで，読出し権および実行権のレベル（スーパバイザ/ユーザ）検査に用いる．

（2） Rタグ・アレイ

セット数 512，ウェイ数 4 のセット・アソシアティブ方式のディレクトリである．通常は 32 ビット長の実アドレス (RA 31-0) 中の 9 ビットのセット・アドレス (RA 13-5) でインデックスする．このうち 2 ビット (RA 13-12) は実ページ番号である．

Rタグ・アレイの各エントリ（Rタグ・エントリ）は，以下の 2 つのフィールドからなる．

①実アドレスタグ（Rタグ）：アクセス・アドレスとの比較に用いる上位実アドレス (RA 31-14)，および，エントリ有効(V)ビット．

②Vタグ・ポインタ (VTP)：①の実アドレスタグと対応関係にある仮想アドレスタグを登録しているVタグ・エントリへのポインタ．Vタグ・アレイのセットを指定する仮想ページ番号の下位 4 ビット

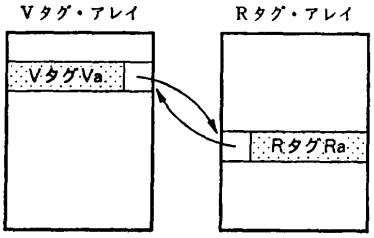


図3 Vタグ—Rタグ間双方向リンク
Fig. 3 Double-link between V-tag and R-tag.

(VA 15-12).

なお，Rタグ・アレイのエントリ数はVタグ・アレイ同様 2048 である．しかし，連想度はVタグ・アレイの1（ダイレクト・マッピング）に対して，Rタグ・アレイは4ウェイと増えている．その理由は5章で述べる．

（3） Vタグ—Rタグ間双方向リンク

図3に示すように，対応関係にある有効なVタグとRタグとの間には双方向リンクを張る[14]．リンクは，セット・アドレスのページ内オフセット部分(A 11-5：仮想アドレスと実アドレスとの共通部分）とタグ・ポインタ（VTP および RTP）とを次のように連結したものである．

①Rタグへのリンク：RA 13-12|A 11-5 およびウェイ番号

②Vタグへのリンク：VA 15-12|A 11-5

キャッシュ内コヒーレンス処理およびキャッシュ間コヒーレンス処理においては，これらのリンクを用いてRタグ・アレイないしVタグ・アレイをアクセスする．

3.2 データ・アレイ

幅 4 B, 深さ 32 K, 容量 128 KB の SRAM である．ライン・サイズが 32 B の場合は，半分の 64 KB のみ使用する．選択されたライン・サイズにより，アクセス・アドレスは次のように定まる．

①32 B のとき：VA 15-2（14 ビット）

②64 B のとき：VA 16-2（15 ビット）

本キャッシュは単一 (unified) キャッシュであり，命令/オペランド，および，スーパバイザ/ユーザの区別を行うことなく，いずれのメモリ・ブロックのコピーも本データ・アレイに存在し得る．また，非共有キャッシュであり，データ・アレイへのアクセス要求は SPARC プロセッサのみ行える．

3.3 通常動作

本キャッシュへのアクセス要求元には，次の2つがある．

①SPARC プロセッサ：32 ビットの仮想アドレスでVタグ・アレイおよびデータ・アレイにアクセスし，命令フェッチおよびオペランド・ロード/ストアを要求する．

②MCU（メッセージ通信ユニット）：キャッシュ間コヒーレンス処理の一環として，32 ビットの実アドレスでRタグ・アレイにアクセスし，コヒーレンス・ブロックの無効化を要求する．

上記②の場合の動作については，参考文献6)，9) を参照されたい．ここでは，①の場合について述べる．SPARC プロセッサからのアクセス要求に対する応答として，以下の3種類がある．

①読出しヒット：命令フェッチおよびオペランド・ロード時にヒットした場合，直ちに当該データを SPARC に返す．

②読出しミス：命令フェッチおよびオペランド・ロード時にミスヒットした場合，ライン・フェッチ要求をバスに送出し，ライン・リプレース処理に入る．ライン・フェッチに必要な実アドレスは TLB から得る．また，ライン・リプレース処理の一環として，キャッシュ内コヒーレンス処理（4章参照）を行う．当該ライン・フェッチが完了するまで他のアクセス要求は受け付けない（ブロッキング・キャッシュ）．

③書込み：オペランド・ストアの場合はヒット/ミスヒットに関わらず，メモリ書込み要求をバスに送出する（ストア・スルー）．実アドレスは TLB から得る．また，書込みミスの場合，読出しミス同様キャッシュ内コヒーレンス処理（4章参照）を行う．

3.4 無効化機能

プログラムによりキャッシュ全体あるいは一部を無効化するための手段として，以下の3種類のコマンドを用意する．

①フラッシュ・コマンド：Vタグ・アレイおよびRタグ・アレイ内のすべてのタグを無効化する．

②Vタグ・パージ・コマンド：Vタグ・アレイに仮想アドレス（および空間 ID）を与えてヒットした場合，当該Vタグ，および，それとリンクしているRタグを無効化する．

③Rタグ・パージ・コマンド：Rタグ・アレイに実アドレスを与えてヒットした場合，当該Rタグ，および，それとリンクしているVタグを無効化する．

4. キャッシュ内コヒーレンス処理

4.1 処理手順

2.3 節で述べた同義語問題をRタグ・アレイを用いて解決する手順について述べる．このキャッシュ内コヒーレンス処理は，ライン・リプレース処理およびオペランド・ストア処理の一環として行う．

(1) ライン・リプレース時

ライン・リプレース時におけるキャッシュ内コヒーレンス処理の手順は，以下のとおりである（**図4**参照）．なお，ライン・リプレースに伴いリプレース・アウトする（キャッシュから追い出す）ラインの仮想および実アドレスをそれぞれ Va および Ra，また，リプレース・インする（キャッシュ内にフェッチする）ラインの仮想および実アドレスをそれぞれ Vb および Rb と記す．

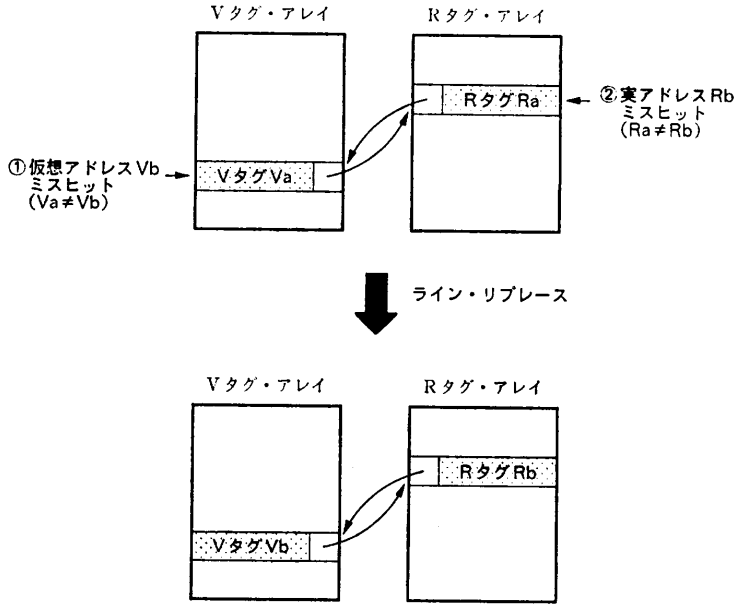
①TLB を用いて，ミスヒットを起こした仮想アドレス Vb を実アドレス Rb に変換する．以後，ライン・フェッチ処理と並行してコヒーレンス処理を進める．

②実アドレス Rb を用いて，Rタグ・アレイをアクセスする．

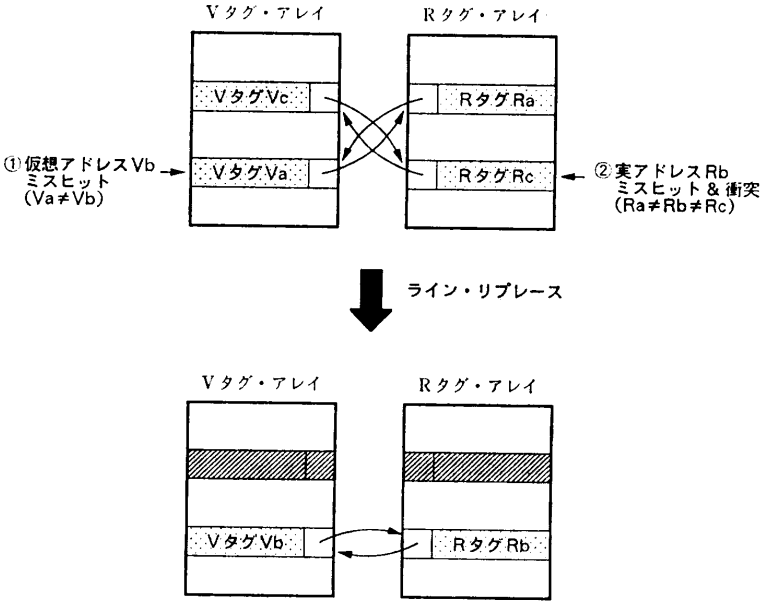
a) ヒットした場合：ヒットしたRタグ・エントリをそのままリプレース・インするラインのためのエントリとして用いる．このとき，当該実アドレスと対応関係にある仮想アドレスは，仮想アドレス Vb の同義語である可能性がある．

b) ミスヒットした場合：リプレース・インするラインのために，Rタグ・エントリを1個空けなければならない．Rタグ・アレイは4ウェイ・セット・アソシアティブなので，4エントリの中から1個を選択する．これには，各Rタグ・エントリの VTP（Vタグ・ポインタ：VA 15-12）と仮想アドレス Vb の VA 15-12 とを比較する．

- 一致するものがあれば，当該Rタグ・エントリを選択する．実アドレス Ra と Rb が同一セットに属する場合に相当する（図4(a)参照）．



(a) 実アドレス Ra と Rb が同一セットに属する場合
(a) In the case of real addresses Ra and Rb belonging to the same set.



(b) 実アドレス Ra と Rb が異なるセットに属する場合
(b) In the case of real addresses Ra and Rb belonging to the different set.

図4 キャッシュ内コヒーレンス処理
Fig. 4 Intra-cache coherence.

●そうでない場合，無効なRタグがあればそのエントリを，なければ任意の有効なRタグ・エントリを選択する．実アドレス Ra と Rb が異なるセットに属する場合に相当する（図4(b)参照）．無効なRタグがなく有効なRタグを選択せざるを得ないとき，これをRタグ間の“衝突(collision)”[15]と呼ぶ（4.2節参照）．

③上記②で選択したRタグ・エントリの VTP を用いてVタグ・アレイをアクセスする．そして，当該Rタグからリンクしているタグを無効化する．

④ライン・フェッチが終了すると，リプレース・インしたラインのVタグとRタグとの間に双方向リンクを張る．

（2） オペランド・ストア時

キャッシュに不在のラインに対してオペランド・ストアを行った場合（書込みミス），以下のキャッシュ内コヒーレンス処理を行う．

①TLB を用いて，ミスヒットを起こした仮想アドレスを実アドレスに変換する．以後，オペランド・ストア処理と並行してコヒーレンス処理を進める．

②この実アドレスを用いて，Rタグ・アレイをアクセスする．ミスヒットの場合，処理完了．

③ヒットした場合，そのRタグ・エントリの VTP を用いてVタグ・アレイをアクセスする．そして，当該Rタグからリンクしているタグを無効化する．

4.2 衝　　突

ライン・リプレース時のキャッシュ内コヒーレンス処理のステップ②において，リプレース・インするラインのRタグの登録エントリをめぐって衝突が起きる可能性があった．この様子を図4(b)を用いて説明する．

①まず，Vタグ Va とRタグ Ra の組，および，Vタグ Vc とRタグ Rc の組が有効である．ここで，仮想アドレス Vb とVタグ Va との不一致 (Va≠Vb) によりミスヒットが起こり，ライン・リプレース処理に入る．

②ライン・リプレースに伴い，リプレース・アウトするラインのVタグ Va とRタグ Ra を無効化する．そして，リプレース・インするラインのVタグ Vb とRタグ Rb を登録する．Vタグ Vb は，Vタグ Va と入れ替わりに同一エントリに登録される．一方，Rタグ Rb は，Rタグ Ra と異なるセットに属しているので Ra と同一エントリには登録できない．Rタグ Rb を登録すべきエントリは他に存在し，そこには既に有効なRタグ Rc が登録されている．つまり，Rタグ Rb と Rc が衝突する．

③Rタグ Rb を登録するためには，Vタグ Vc—Rタグ Rc 間の双方向リンクを解く必要がある．その結果，本来ライン・リプレースとは無関係であったVタグ Vc までも無効化される．

④ライン・リプレース処理が完了したとき，Vタグ Vb とRタグ Rb の間に双方向リンクが張られ，これらの組が有効となる．

このように，衝突が原因で，ライン・リプレースとは本来無関係であった（Vタグ Vc の）ラインが無効化され，有効なラインが1個減少する．衝突が多発すると，有効ライン数の減少，ひいてはキャッシュ・ヒット率の低下を招くおそれがある．よって，衝突頻度を皆無あるいは低く抑えるために，なんらかの処置を施す必要がある．

5. 衝突問題の解決

5.1 原因および対処

衝突の原因は，スーパセット・サイズ（スーパセット当りのライン数）とRタグ・アレイのセット・サイズ（セット当りのライン数＝ウェイ数）とが異なる点にある．スーパセットとはある1つのページ内オフセットでアクセス可能なラインの集合であり，そのサイズはキャッシュ・サイズとページ・サイズの比に等しい[14]．本キャッシュの場合，キャッシュ・サイズ 64 KB（ライン・サイズ 32 B のとき）およびページ・サイズ 4 KB であるから，スーパセット・サイズは 16 となる．つまり，同一ページ内オフセットでインデックスされるVタグ・エントリとRタグ・エントリがそれぞれ 16 個ずつ存在する．これに対して，Rタグ・アレイのセット・サイズは4である．同一スーパセットに属する 16 個のRタグ・エントリが4個の異なるセットに分かれて属することになる．よって，リプレース・イン/アウトする2つのラインの実アドレスがそれぞれ異なるセットに属する可能性，すなわち，衝突の可能性がある．

逆に言えば，リプレース・イン/アウトする2つのラインの実アドレス同士が同一セットに属すれば，衝突は起きない．これには，次の2つの方法がある．

①Rタグ・アレイのセット・サイズをスーパセット・サイズと等しくする．つまり，16 ウェイのセット・アソシアティブとする．

②2つの仮想アドレスが同一セットに属するときは，

それらの実アドレスも同一セットとなるように，仮想アドレス-実アドレス間の対応関係をとる．たとえば，仮想ページ番号の下位4ビット (VA 15-12) と実ページ番号の下位4ビット (RA 15-12) とが一致するように，アドレスを写像する．

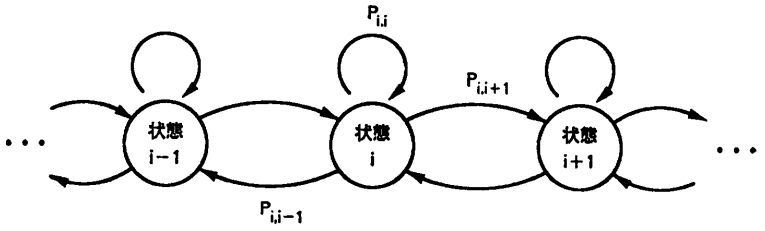


図 5 状態遷移図
Fig. 5 State transition diagram.

まず，上記②の方法は，OS で行うアドレス写像に対して制約を課すことになりあまり好ましくない．さらに，仮想ページと実ページ・フレームとの対応関係が固定されるので，フラグメンテーションの問題が生じる．一方，方法①には，このような問題は起きない．しかし，2.1.2 項で述べたように，タグ・アレイのハードウェア・コストはウェイ数に比例して大きくなる．Rタグ・アレイの連想度を16ウェイにするには，膨大なハードウェア・コストが必要である．よって，現実的には，衝突によるキャッシュの性能低下，および，連想度向上に必要なハードウェア・コストとの間のトレードオフをとって，Rタグ・アレイの連想度を決定することになる．

5.2 Rタグ・アレイの連想度

確率的解析およびトレース駆動シミュレーションの2手法を用いて，Rタグ・アレイの連想度を決定した[7]．

5.2.1 確率的解析

マルコフ連鎖を用いた確率的解析により，Rタグ・アレイの連想度とミスヒット率との関係を調査した[15]．まず，以下の定義を行う．

L：総ライン数 (=2048)
S：スーパセット・サイズ (=16)
w：Rタグ・アレイの連想度 (ウェイ数) $(1 \leqq w \leqq S)$
i：有効なライン数 $(1 \leqq i \leqq L)$
状態 i：キャッシュの有効ラインが i 個ある状態
m_i：総ライン数 i の (Rタグ・アレイのない) キャッシュにおけるミスヒット率
π_i：定常状態でキャッシュが状態 i にある確率
$p_{i,j}$：状態 i から状態 j への定常推移確率

図5にキャッシュの状態遷移図を示す．いまキャッシュが状態 i にあるとすると，キャッシュミスによりキャッシュの状態は次のように遷移する．

①状態 i →状態 $i+1$：無効なVタグとRタグとを有効なVタグ Vb とRタグ Vb とでリプレースした．よって，有効なライン数が1個増える．

②状態 i →状態 $i-1$：有効なVタグ Va とRタグ Rc とを有効なVタグ Vb とRタグ Rb とでリプレースした (図4(b)参照)．このとき，Rタグ Rb と Rc が衝突し，Vタグ Vc が無効化される．よって，有効なライン数が1個減る．

③状態 i →状態 i：上記①および②以外．たとえば，有効なVタグ Va とRタグ Ra とを有効なVタグ Vb とRタグ Rb でリプレースした (図4(a)参照)．この場合，有効なライン数は不変である．

さて，定常推移確率 $p_{i,i+1}$，および $p_{i,i-1}$ は，次のようになる．

$$p_{i,i+1}=\pi_i \cdot m_i \cdot \left(\frac{L-i}{L}\right) \cdot \left\{1-\left(\frac{i}{L}\right)^w\right\}. \quad (1)$$

$$p_{i,i-1}=\pi_i \cdot m_i \cdot \left(\frac{i}{L}\right)^{w+1} \cdot \left(1-\frac{w}{S}\right). \quad (2)$$

ここで，定常状態においては，次のように $p_{i,i+1}=p_{i+1,i}$ が成り立つ．

$$\pi_i \cdot m_i \cdot \left(\frac{L-i}{L}\right) \cdot \left\{1-\left(\frac{i}{L}\right)^w\right\} = \pi_{i+1} \cdot m_{i+1} \cdot \left(\frac{i+1}{L}\right)^{w+1} \cdot \left(1-\frac{w}{S}\right). \quad (3)$$

$(i=1, 2, \cdots, E-1)$.

また，次の正規化条件が成り立つ．

$$\sum_{i=1}^{L} \pi_i = 1. \quad (4)$$

$L-1$ 個の式(3)と式(4)より L 個の π_i を求める．これより，総ライン数 L のキャッシュのミスヒット率 m，および，使用効率 u を下式で求める．

$$m=\sum_{i=1}^{L} m_i \cdot \pi_i. \quad (5)$$

$$u=\sum_{i=1}^{L} \frac{m_i \cdot \pi_i}{L}. \quad (6)$$

Rタグ・アレイのウェイ数 (w) を1，2，4，8として求めた結果を表3に示す．なお，m_i の値として，後述するトレース駆動シミュレーションにおけるRタ

表3 確率的解析結果
Table 3 Results by probabilistic analysis.

	$w=1$	$w=2$	$w=4$	$w=8$
ミスヒット率 m (%)	9.480	9.416	9.370	9.348
ライン・リプレース率 r (%)	0.876	0.812	0.766	0.745
キャッシュ使用効率 u (%)	49.15	63.16	78.09	86.42

w：ウェイ数

グ・アレイなしのキャッシュのミスヒット率を用いた．また，ミスヒット率 m_r の代わりに，ライン・リプレース率（ライン・リプレース回数/キャッシュ・アクセス回数）r_i を用いて同様の計算を行った結果も表3に示した．

5.2.2 シミュレーション

トレース駆動シミュレーションにより，Rタグ・アレイの連想度がキャッシュ性能に与える影響を測定した．トレースデータは，Dhrystone ベンチマーク・プログラムを Sun-4 (SPARC プロセッサ) 上で実行させて採取した．シミュレーションは，以下の条件で行った．

①ライン・サイズは 32B，キャッシュ・サイズは 64 KB とする．

②キャッシュ間コヒーレンス処理によるライン無効化は生じない．

③キャッシュ内コヒーレンス処理によるライン無効化は生じない（同義語は生じない）．

④仮想アドレスと実アドレス間のアドレス写像は，次の規則に従う．すなわち，仮想ページ番号の下位4ビット (VA 15-12) を上位の適当な4ビットと入れ替えて実ページ番号を得る．したがって，衝突は起こり得る．

シミュレーション結果を表4に示す．これは，まず25万回キャッシュ・アクセスを行った後に行った50万回のキャッシュ・アクセスに関するものである (warm start)．ストア・スルーであるので，キャッシュへの書込みはすべてミスヒットとして扱っている．ただし，衝突はライン・リプレース（読出しミス）時に生じるので，ライン・リプレース回数および率を別に取り出した．

表4には，Rタグ・アレイのウェイ数を 1, 2, 4, 8としたときのシミュレーション結果を示している．また，比較のために，Rタグ・アレイなしのキャッシュのシミュレーション結果も示した．なお，これはウェイ数が 16 の場合（衝突が起きない場合）に相当する．

5.2.3 連想度の決定

表3および表4より，Rタグ・アレイの連想度が，ミスヒット率，ライン・リプレース率，キャッシュ使用効率などの性能に影響を与えていることがわかる．

まず，Rタグ・アレイの導入に伴い，衝突による性能低下が現れている．特に，Rタグ・アレイなしの場合に比べて，連想度1（ダイレクト・マッピング）のRタグ・アレイを用いたときの性能低下は著しい（表4参照）．しかしながら，連想度を2，4，8と増していくと，性能低下の度合は小さくなっていく．連想度が4以上になると，ミスヒット率はRタグ・アレイなしのキャッシュとほとんど変わらない．同時に，連想度を4から8に上げてもミスヒット率減少にあまり寄与していないことがわかる．ただし，キャッシュ使用効率については向上がまだ見られる（ミスヒット率とキャッシュ使用効率との間に明確な相関関係が見られないのは，参照の局所性によるものと考えられる）．

以上から，Rタグ・アレイの連想度は，ハードウェア・コストとのトレードオフも図って4ウェイとした．

表4 シミュレーション結果
Table 4 Results by simulation.

	Rタグ・アレイなし	Rタグ・アレイあり			
		w=1	w=2	w=4	w=8
キャッシュ・アクセス回数	500,000				
命令フェッチ回数	381,068				
ロード回数	75,913				
ストア回数	43,019				
ミスヒット回数	46,586	55,952	47,228	46,781	46,694
ライン・リプレース回数	3,567	12,933	4,209	3,762	3,675
衝突回数		12,240	3,250	2,291	1,244
ミスヒット率 (%)	9.317	11.190	9.446	9.356	9.339
ライン・リプレース率 (%)	0.713	2.587	0.842	0.752	0.735
キャッシュ使用効率 (%)	100	13.23	26.32	51.95	87.70

w：ウェイ数

6. おわりに

以上，可変構造型並列計算機のキャッシュの設計方針，構成および動作，キャッシュ内コヒーレンス処理，

および，衝突問題について述べた．

本システムでは，高速性および大容量性を重視してダイレクト・マッピング方式の仮想アドレス・キャッシュを採用した．本キャッシュでは，以下の3種類のコヒーレンスを保証する必要があった．

①キャッシュ内コヒーレンス
②キャッシュ－メモリ間コヒーレンス
③キャッシュ間コヒーレンス

上記①③のコヒーレンスを保証するため，タグアレイを仮想および実アドレス対応に2個設けるデュアル・ディレクトリ・キャッシュ構成とした．本論文では，①のキャッシュ内コヒーレンス処理を詳述した．なお，②③のコヒーレンス処理については，別の機会に報告する予定である．

さらに，キャッシュ内コヒーレンス処理の際に生じる衝突問題について述べた．衝突によるキャッシュ性能低下を緩和するため，ダイレクトマッピング方式の仮想アドレス・タグアレイに対して，実アドレス・タグアレイはセット・アソシアティブ方式とすることにした．その連想度は，確率的解析およびトレース駆動シミュレーションにより評価を行った結果，4ウェイと決定した．

謝辞 本キャッシュ・システムの初期設計に携わった濱口一正氏（現 キヤノン(株)），ならびに，現在我々とともに設計・開発を行っている甲斐康司，上野智生，徳永直哉の各氏，および，日頃ご討論いただく富田研究室の皆様に感謝いたします．

PE 開発においては富士通(株)本体事業部・スーパーコンピュータ開発部の内田啓一郎氏，高村守幸氏，および，京セラ(株)システム機器開発課の重村慎二氏，システム実装設計に関してはアジアエレクトロニクス(株)の横田孝氏ならびに電子機器事業部の各位，ニシム電子工業(株)の和田勲夫氏に，ご助言ご協力をいただいている．謹んで感謝いたします．

また，衝突問題に関しては，米 Wisconsin 大学 Madison 校の Prof. M. D. Hill に貴重な御意見を頂いた．記して感謝します．

本研究は一部文部省科学研究費補助金による．

参考文献

1) Murakami, K. et al.: The Kyushu University Reconfigurable Parallel Processor—Design Philosophy and Architecture—, *Proc. IFIP 11th World Computer Congress*, pp. 995-1000 (1989).
2) Murakami, K. et al.: The Kyushu University Reconfigurable Parallel Processor—Design of Memory and Intercommunication Architectures—, *Proc. 1989 Int'l. Conf. on Supercomputing*, pp. 351-360 (1989).
3) 森ほか：可変構造型並列計算機の PE 間メッセージ通信機構，情報処理学会論文誌，Vol. 30, No. 12, pp. 1592-1602 (1989).
4) 甲斐ほか：可変構造型並列計算機のローカル/リモート・メモリ・アーキテクチャ，情報処理学会研究報告，ARC-80-11 (1990).
5) 岩田ほか：可変構造型並列計算機におけるキャッシュ・コヒーレンス処理，第 39 回情報処理学会全国大会論文集，5X-2 (1989).
6) 岩田ほか：可変構造型並列計算機のキャッシュ・システム，情報処理学会研究報告，ARC-79-3 (1989).
7) 岩田ほか：可変構造型並列計算機のキャッシュの単体性能評価，第 40 回情報処理学会全国大会論文集，4L-6 (1990).
8) 岩田ほか：統合型並列化コンパイラ・システム—コンパイラ支援キャッシュ・コヒーレンス制御—，情報処理学会研究報告，ARC-83-21 (1990).
9) Mori, S. et al.: The Kyushu University Reconfigurable Parallel Processor—Cache Architecture and Cache Coherence Schemes—, *Proc. Int'l. Symp. on Shared Memory Multiprocessing*, pp. 218-229 (1991).
10) Smith, A. J.: Cache Memories, *ACM Computing Surveys*, Vol. 14, No. 3, pp. 473-530 (1982).
11) Smith, A. J.: Line (Block) Size Choice for CPU Cache Memories, *IEEE Trans. Comput.*, Vol. C-36, No. 9, pp. 1063-1075 (1987).
12) Hill, M. D.: A Case for Direct-Mapped Caches, *IEEE Computer*, Vol. 21, No. 12, pp. 25-40 (1988).
13) Hill, M. D.: *Aspects of Cache Memory and Instruction Buffer Performance*, Ph. D. dissertation, Tech. Report 87/381, Computer Science Dept., Univ. of California, Berkeley (1987).
14) Goodman, J. R.: Coherency for Multiprocessor Virtual Address Caches, *Proc. 2nd Int'l. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS II)*, pp. 72-81 (1987).
15) Hill, M. D.: Private Communication (1989).

（平成2年8月8日受付）
（平成3年4月9日採録）

岩田 英次（正会員）

1966年生．1989年九州大学工学部電子工学科卒業．1991年九州大学大学院総合理工学研究科情報システム学専攻修士課程修了．同年ソニー(株)入社．現在同社総合研究所に所属．在学中可変構造型並列計算機の研究に従事．ディジタル信号処理，並列処理，計算機アーキテクチャの研究に興味を持つ．

森 眞一郎（正会員）

1963年生．1987年熊本大学工学部電子工学科卒業．1989年九州大学大学院総合理工学研究科情報システム学専攻修士課程修了．現在同大学院博士課程に在学中．並列処理，計算機アーキテクチャの研究に従事．

村上 和彰（正会員）

1960年生．1982年京都大学工学部情報工学科卒業．1984年同大学院工学研究科情報工学専攻修士課程修了．同年富士通(株)本体事業部に入社．主として汎用計算機Mシリーズのアーキテクチャ開発に従事．1987年九州大学工学部助手，1988年同大学院総合理工学研究科情報システム学専攻助手，現在に至る．計算機アーキテクチャ，コンパイラ，並列処理等の研究に従事．著書「計算機システム工学（共著，昭晃堂）」．電子情報通信学会，日本応用数理学会，ACM，IEEE，IEEE-CS 各会員．

福田 晃（正会員）

1954年生．1977年九州大学工学部情報工学科卒業．1979年同大学院修士課程修了．同年NTT研究所入所．1983年九州大学大学院総合理工学研究科情報システム学専攻助手．1989年同大学助教授，現在に至る．工学博士．計算機システムの性能評価，並列処理，オペレーティング・システムなどに興味をもつ．訳書：オペレーティングシステムの概念（共訳，培風館）．電子情報通信学会，日本 OR 学会，ACM，IEEE 各会員．

富田 眞治（正会員）

1945年生．1968年京都大学工学部電子工学科卒業．1973年同大学院博士課程修了．工学博士．同年京都大学工学部情報工学教室助手．1978年同助教授．1986年九州大学大学院総合理工学研究科教授．1991年京都大学工学部情報工学科教授，現在に至る．計算機アーキテクチャ，並列処理システムなどに興味を持つ．著書「並列計算機構成論」「計算機システム工学」「並列処理マシン」など．電子情報通信学会，IEEE，ACM 各会員．