計算機アーキテクチャ　80－3
（１９９０．１．２５）

二階層並列キャッシュ コンシステンシプロトコル

浅野　滋博
東芝　総合研究所

二階層並列キャッシュの一貫性を保つためのコンシステンシプロトコルについて述べる．書き込み時にコピーを無効化することでコンシステンシを保つ．一階層目，二階層目のキャッシュはキャッシュブロック毎に４つの状態を持つことにより不要なバストランザクションが出ないように管理し，二階層ともコピーバック方式のプロトコルが実現できた．また．二階層目をＮ－ｗａｙとし，一階層目にコピーが存在するときは二階層目から追い出さないようにすることでMulti-Level-Inclusionを保ち，プロトコルを簡略化できた．

A DESIGN OF MULTILEVEL PARALLEL CACHE CONSISTENCY PROTOCOL

Shigehiro Asano

TOSHIBA R&D Center

1 Komukai-Toshiba-cyou,Saiwai-ku,Kawasaki-city,Kanagawa 210,Japan

A cache consistency protocol for multilevel parallel cache system is presented. It is based on broadcast-invalidate scheme. Both first and second-level cache has 4 states to maintain consistency and suppress bus-traffic.

To maintain property of Multi-Level-Inclusion, second-level cache is N-way set associative. When some entry must be replaced, an entry which has no copy in above first-level cache is chosen. Multi-Level-Inclusion simplify protocol and reduce bus-traffic.

1　はじめに

共有メモリ型のマルチプロセッサシステムでは，メモリへのアクセス競合をどの程度緩和できるかが台数効果を得るためのキーとなる．共有メモリ型のマルチプロセッサを構成するための方法としてはプロセッサと共有メモリをネットワークを用いて結合する方法と，バスを用いて結合する方法の二つが考えられるが，バスを用いた結合は比較的安価に構成が可能で，しかも拡張性があるという利点がある．バス結合を用いた場合にバスでの競合を緩和させる手段としてスヌープキャッシュが知られている．[Arch86]

一方，ＶＬＳＩ技術の進展にともなってチップ内部の動作速度は年々高速化する傾向にあり，それに比べて比較的低速なメモリとのインタフェースは重要な問題となりつつある．キャッシュメモリの二階層化はプロセッサの速度と，メモリの速度を緩衝する手段として知られている．[Shor88]

以上の様な点を考慮して，筆者は二階層並列キャッシュメモリのコンシステンシプロトコルを設計した．本報告では，コンシステンシを保つための基本的な方式，ならびにプロトコルの詳細に付いて報告する．

2．二階層キャッシュの構成

二階層キャッシュの構成としてはプロセッサ毎に二階層目のキャッシュを独立に持つ図１の構成と，二階層目のキャッシュを複数のプロセッサで共有しツリー状の構造を持つ図２の構成が考えられる．

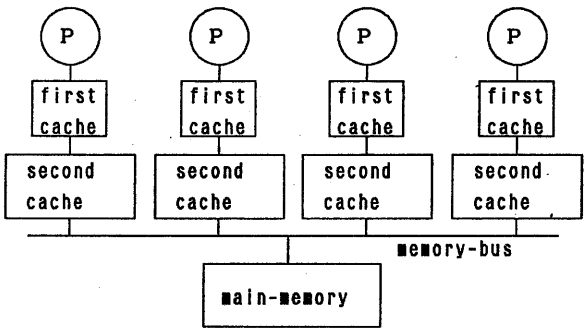

図１　独立構成の二階層キャッシュ

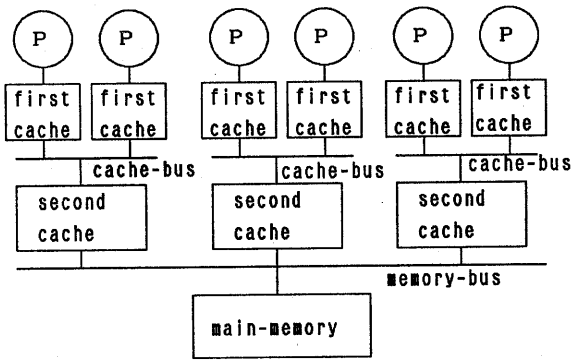

図２　ツリー構造の二階層キャッシュ

本報告では，プロセッサに近いほうのキャッシュをfirst-cache，共有メモリに近いほうのキャッシュをsecond-cache，プロセッサ側のバスをcache-bus，メモリ側のバスをmemory-busと呼ぶことにする．

ところで，並列キャッシュシステムにおいては，バスにトランザクションを発生させる原因として次の二つの要素が考えられる．

１）キャッシュにコピーを持たないアドレスにアクセスしようとしたために起こるメモリからキャッシュへのフェッチ動作（シングルプロセッサのキャッシュミスと同じ意味である）．

２）他のキャッシュにコピーを持つアドレス（共有ブロック）に書き込もうとしたために起こるコンシステンシを保つためのトランザクション（プロセッサ間で通信をするためのトランザクション）．

図２の構成の場合は，通信をするプロセッサがsecond-cacheを共有する組であればmemory-busにコンシステンシのためのトランザクションを発生させることなくトランザクションがcache-busだけに発生する構成が可能である．

本報告では図２の構成のシステムを対象としたが，図１の構成でも同様に機能する．

3．コンシステンシを保つための方法

スヌープキャッシュでコンシステンシを保つための方法として大きく分けて二つの方法が存在する．

１）書き込みを起こしたブロックの他のキャッシュの

コピーを無効化するインバリデート方式.

2）書き込みを起こした内容を放送により，コピーを持つ他のキャッシュを更新するブロードキャストライト方式.

本報告で扱うプロトコルは，インプリメントの容易さからインバリデート方式を採用した.

無効化方式では，プロセッサから書き込もうとしたキャッシュブロックが，他のキャッシュと共有されているときには他のキャッシュを無効化してから書き込みを行う.

無効化をするためのトランザクションをなるべく少なくするためにキャッシュブロックの状態としてどの様な状態を持ち，どの様なトランザクションを発生させるかで，今日まで様々なプロトコルが提案されてきた．本報告ではそれらの中で比較的簡単な Berkeley のプロトコル[Borr84]を二階層に拡張し，二階層ともコピーバックを行うキャッシュプロトコルを提案する．プロトコルの特徴は下記の通りである.

・コピーバック方式
・書き込み時に他のキャッシュを無効化する
・オーナーシップに基づく
・キャッシュ間転送を行う

4．Multi-Level-Inclusionとsecond-cacheの役割

Multi-Level-Inclusionは[Baer87]で提案された概念で，階層化されたキャッシュのコンシステンシを保つために重要な概念である．この性質は

[上の階層（first-cache）にあるエントリが必ず下の階層（second-cache）にある．]

という性質である.

本プロトコルでは二階層共にコピーバックであるために上の階層のコピーが下の階層にあるという性質は保証できない．その代わりに，上の階層に存在する有効なキャッシュブロックのアドレスが必ず下の階層に存在し，それをエントリと呼ぶことにした.

この性質を用いれば，memory-bus上にあるトランザクションのうち必要な（すなわち，上の階層にトランザクションを伝える必要がある）トランザクションだけをcache-busに伝えることでcache-bus間の干渉を低く抑えられる.

一方，second-cache自身もキャッシュブロック毎に状態を持ち他のsecod-cacheに接続されるプロセッサにトランザクションを伝達する必要がなければmemory-busにトランザクションを発生させない.

このように，second-cacheは必要なトランザクションだけを通過させ，複数のバス間での干渉を低く抑えるように機能する.

5．プロトコルの詳細

この章ではプロトコルの詳細に付いて述べる.

5．1 first-cacheの状態とcache-bus上のコマンド

first-cacheはキャッシュブロック毎に4通りの状態を持っている.

表5．1 first-cacheの各キャッシュブロックの状態

INV (INValid)	無効
UNO (UNOwned)	他のfirst-cacheと共有 オーナシップなし
EXC (EXClusive own)	他のfirst-cacheと共有無し. オーナシップあり
NON (Not exclusive owN)	他のfirst-cacheと共有 オーナシップあり

cache-bus上のコマンドは以下の通りである.

表5．2 cache-bus上のコマンド

RSH (Read SHared)	キャッシュにデータをフェッチする
RFO (Read For Ownership)	キャッシュにデータをフェッチし他のコピーは無効化する
WFI (Write For Invalidate)	他のコピーを無効化する
WWI (Write Without Invalidate)	second-cacheにデータをコピーバックする.

この他にも，second-cacheからcache-busに出されるコマンドがあるが，後で説明することにする.

ＣＰＵからの要求に応えてfirst-cacheは表１の動作を行う．

表１ＣＰＵからのコマンドに対するfirst-cacheの応答

	Read	Write
ＩＮＶ	RSH/UNO	RFO/EXC
ＥＸＣ	no	no
ＮＯＮ	no	WFI/EXC
ＵＮＯ	no	WFI/EXC

注）ｍｉｓｓした場合それがＥＸＣまたはＮＯＮならばＷＷＩ（flush）をやってからＩＮＶと同様に扱う．noはバスにコマンドを出さないことを示している．

cache-busのコマンドに応えてfirst-cacheは表２の動作を行う．

表２中にあらわれるＦＡＩとＦＷＩはsecond-cacheからcache-busに出るコマンドで次のような働きをする．

表５．３　second-cacheからcache-busへのコマンド

ＦＡＩ (Flush And Invalidation)	first-cacheからsecond-cacheにデータを書き込みfirst-cacheを無効化
ＦＷＩ (Flush Without Incalidation)	first-cacheからsecond-cacheに書き込んでfirst-cacheは無効化しない

５．２　second-cacheの状態とmemory-busのコマンド

second-cacheは以下に示す４状態を持っている．

表５．４　second-cacheの４状態

ＩＮＶ	無効
ＵＮＯ	他のsecond-cacheとブロックを共有している可能性がある．オーナシップなし．
ＥＸＣ	上のfirst-cacheが最新の値で，オーナシップをもつ．他にコピーがない．
ＮＯＮ	他にコピーがある．オーナシップを持つ

また，second-cacheからmemory-busに出されるコマンドはcache-busの場合と同様に

ＲＳＨ
ＲＦＯ
ＷＦＩ
ＷＷＩ

がある．

second-cacheはcache-busのコマンドに応答して表３の動作をする．

表２cache-busのコマンドに対するfirst-cacheの応答

	RSH	RFO	WFI	WWI	FAI	FWI
INV miss	no	no	no	no	no	no
EXC	datatoC /NON	datatoC /INV	never	never	datatoC /INV	datatoC /UNO
NON	datatoC	datatoC /INV	-/INV	never	datatoC /INV	datatoC /UNO
UNO	no	- /INV	-/INV	no	- /INV	no

datatoCはcache-Busにデータを応答すること

表３cache-Busのコマンドに対するsecond-cacheの応答

	RSH	RFO	WFI	WWI	FAI	FWI
INV	RSHtoM /UNO datatoC	RFOtoM /EXC datatoC	never occur	never occur	never occur	never occur
EXC	no	no	no	-/NON	never occur	never occur
NON	datatoC	WFItoM /EXC datatoC	WFItoM /EXC	never occur	never occur	never occur
UNO	datatoC	WFItoM /EXC datatoC	WFItoM /EXC	never occur	never occur	never occur

表３中では，
missでreplaceしようとしたブロックがＮＯＮの時はＷＷＩｔｏＭをやってからＩＮＶと同じに扱う．ただし，IFCにmissした場合は応答しない．
・ＲＦＯｔｏＭはmemory-BusへのＲＦＯコマンド
・ＷＦＩｔｏＭはmemory-BusへのＷＦＩコマンド
・ｄａｔａｔｏＣはcache-Busへのｄａｔａの応答
・ＷＷＩｔｏＭはmemory-BusへのＷＷＩコマンド

memory-bus上のコマンドにsecond-cacheは表４のように応答する．

表４　memory-busに対するsecond-cacheの応答

	ＲＳＨ	ＲＦＯ	ＷＦＩ	ＷＷＩ
ＩＮＶ miss	no	no	no	no
ＥＸＣ	FWItoC datatoM /NON	FAItoC datatoM /INV	never occur	never occur
ＮＯＮ	datatoM	if_ubaz WFItoC datatoM /INV	if_ubaz WFItoC /INV	never occur
ＵＮＯ	no	if_ubaz WFItoC /INV	if_ubaz WFItoC /INV	no

表４中では，
・ＦＷＩｔｏＣはcache-BusにＦＷＩを出すことである
・ＦＡＩｔｏＣはcache-BusにＦＡＩを出すことである
・ＷＦＩｔｏＣはcache-BusにＷＦＩを出すことである
・ｄａｔａｔｏＭはmemory-Busにｄａｔａを返すことである
・ubazはそのエントリのＵ－Ｂｉｔがすべて０であることを示している．Ｕ－Ｂｉｔについては次の節で説明する．

表５にsecond-cacheがmemory-busに出したコマンドに対するメモリの応答を示す．

表５．　メモリのmemory-busに対する応答

	ＲＳＨ	ＲＦＯ	ＷＦＩ	ＷＷＩ
	if_cache datatoM	if_cache datatoM	no	write data to memory

５．３　second-cacheとreplacementアルゴリズム

Multi-Level-Inclusionの性質を満たすために，本プロトコルでは特別なreplacementアルゴリズムを採用している．Multi-Level-Inclusionの性質を保つためにはsecond-cacheからあるエントリが追い出されるときに，first-cacheにそのエントリがないことを保証しなければならない．

この性質を満たすための一つの方法は，second-cacheからエントリを追い出す際に，first-cacheのエントリを無効化する方法である．しかし，そのためにはcache-busにトランザクションを発生させることが必要となる．

もう一つの方法は，first-cacheに有効なエントリが存在する場合にはsecond-cacheのエントリを追い出さないようなreplacementのアルゴリズムにすることである．本プロトコルではこの方法を採用することにした．

５．３．１　replacementアルゴリズムのためのsecond-cacheの構成

first-cacheに存在するエントリが必ずsecond-cacheに存在するためには少なくともsecond-cacheの容量がそれに接続されるfirst-cacheの容量を上回っている必

要がある．一般にはfirst-cacheがＮｗａｙでsecond-cacheがＭ台のプロセッサで共有されるとき，Ｎ＊Ｍｗａｙであればよい．

以後は簡単のためfirst-cacheがダイレクトマップ，second-cacheはＮｗａｙであるとして説明をすすめる．

５．３．２　Ｕ－Ｂｉｔ

second-cacheの各エントリのタグ領域にＵ－Ｂｉｔと呼ぶ情報を付け加える．Ｕ－Ｂｉｔはそのエントリがfirst-cacheに使用されているかどうかを示している情報で，これを利用してreplaceされるエントリが決定される．

Ｕ－Ｂｉｔはエントリ毎にsecond-cacheに接続されるプロセッサの数（ｗａｙ数と一致する）だけ存在し，これが１であることは対応するfirst-cacheがこのエントリを使用中であることを示している．Ｕ－Ｂｉｔを立てるときは同じセット中の他のエントリの対応するfcacheのＵ－Ｂｉｔは下ろす．Ｕ－Ｂｉｔを下ろすときは，その外にＷＷＩをそのエントリに行ったときである．

Ｕ－Ｂｉｔはcache-busのコマンドに対して変化させる．但し，説明のため以下に述べる記述法を使う．

Ｕ［エントリ番号，キャッシュ番号］＝１または０

エントリ番号はセット中のエントリの番号で，Ｗａｙの数だけある．キャッシュ番号はfirst-cacheの番号である．

（１）ＲＳＨ

Ｕ［選択されたエントリ，コマンドを発行したfirst-cache］＝１

Ｕ［同じセットの中の選択されなかったエントリ，コマンドを発行したfirst-cache］＝０

（２）ＲＦＯ

Ｕ［選択されたエントリ，コマンドを発行したfirst-cache］＝１

Ｕ［同じセットの中の選択されなかったエントリ，コマンドを発行したfirst-cache］＝０

Ｕ［選択されたエントリ，コマンドを発行したfirst-cache以外のfirst-cache］＝０

（３）ＷＦＩ

Ｕ［選択されたエントリ，コマンドを発行したfirst-cache以外のfirst-cache］＝０

（４）ＷＷＩ

Ｕ［選択されたエントリ，コマンドを発行したfirst-cache］＝０

（５）ＦＡＩ

Ｕ－Ｂｉｔは変化させない．

（６）ＦＷＩ

Ｕ－Ｂｉｔは変化させない．

［その他］

新しいエントリをsecond-cacheに確保するときはそのエントリのＵ－Ｂｉｔをすべて０にしてから上記の操作を行う．

［性質］

以上のコマンドにより，Ｕ－Ｂｉｔがfirst-cacheの状態を完全に反映していない状態が過渡的に起こる．first-cacheがＵＮＯのときはreplace時にバスにコマンドを出さないのでＵ－Ｂｉｔが立ったままfirst-cacheがなくなることになる．この状態は次にRSHまたはRFOで読まれるときに正しいＵ－Ｂｉｔの状態になるはずである．

図３はsecond-cacheを３つのプロセッサで共有する場合を説明した図である．

図ではプロセッサＢとプロセッサＣが同じエントリを共有している．

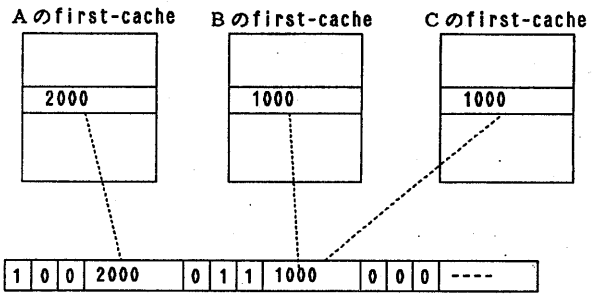

図３　Ｕ－Ｂｉｔの例

5．3．3　U－Ｂｉｔによるreplacementエントリの決定

セット中でreplaceされるエントリを次の順序で決定する．

1）ＩＮＶになっているエントリ
2）Ｕ－Ｂｉｔが全て0のエントリ（つまりどのfcacheにも現在使用されていないエントリ）の中からrandomな一つ（ＬＲＵを使用しても良い）
3）fetcheしようとしているfirst-cacheに対応するＵ－Ｂｉｔが1のエントリ

以上の方法により制約が満たされるようにreplaceが行われる．

以上のような方法によりreplaceされるエントリが決定され，Multi-Level-Inclusionが常に保たれる．

5．4　プロトコルでとりえる状態の組合せ．

本プロトコルではキャッシュブロックの状態は表6に示すように管理される．

6．デッドロックの回避

second-cacheはcache-bus，memory-busの両方からの要求に応えているが，バスのプロトコルによってはデッドロックを生じることがある．

例えば，図4のようにプロセッサP1がcache-bus C1にRSHの要求を出し，second-cache S1にミスしてmemory-busにRSH要求を出そうとしてcache-bus　C1をとりっぱなしにしている．一方，memory-busのWFIコマンドはcache-bus　C1を必要としてmemory-busをとりっぱなしにするとここでデッドロックが生じる．

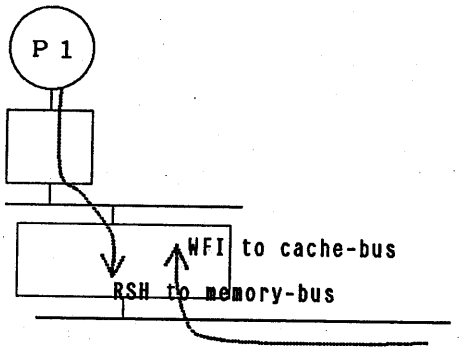

図4　デッドロックの例

この状態を回避するためにはsecond-cacheがデッドロックを検出したらcache-busに信号を出してcache-busのマスタとなっているfirst-cacheにバスの使用を止めさせればよい．

表6　first-cache,second-cache間でとりえる状態の組合せ

secondが	ＥＸＣ	ＮＯＮ	ＵＮＯ	ＩＮＶ	
他のsecond cacheは	INV	UNO INV	NON UNO INV	EXC NON UNO INV	NON,EXCのブロックはsecond-cache間で1個のみ．
上のfirst-cacheは	EXC NON UNO INV (*2)	UNO INV (*1)	UNO INV	INV	

(*1)NONの上のfirst-cacheにNONは来ない．その理由は，first-cacheがEXCであるときに（その下のsecond-cacheはEXC）他のミニクラスタからRSHが来るとfirst-cacheとsecond-cacheの内容を一致させた上でfirst-cacheの状態をUNOに変えているからである．
(*2)EXCの上のfirst-cacheには必ずownershipを持ったキャッシュブロックがある．

7．まとめ

本報告では並列二階層キャッシュのコンシステンシプロトコルの一つを提案した．並列二階層キャッシュのコンシステンシプロトコルとして現在までに知られているものとしては[山岡８８]や[Wils87]にあげられる方法がある．[Wils87]はWrite-Onceプロトコル[Good83]を二階層に適用したものと考えられる．本報告で示した方法はキャッシュ間転送を行うことにより更にバスのトランザクションを効率化して，下位のキャッシュ及びメモリはデータのスワップ領域のような働きをする．

また，本プロトコルではreplacementアルゴリズムによりバスに無効化のトランザクションを発生させずにMulti-Level-Inclusionの性質を保てる．

現在，二階層キャッシュの有功性を確認するためにシミュレータによる評価をすすめている．シミュレータでは各階層でのメモリアクセスの速度，バスの速度等を変化させることができる．

シミュレータで確認するのは次のような項目である．

（１）二階層キャッシュでは共有ブロックに書き込みが起こってmemory-busまで書き込みの情報を伝えなくてはならないときには，一階層の場合に比べて書き込みのレーテンシが大きい．どの様なパラメータのもとでなら二階層キャッシュが有利になるのか？

（２）図１の構造と，図２の構造で図２の構造の効果が出るのはプロセッサ間のデータの共有に特徴がある時だけである．共有の偏りがどうなれば効果があるのか？

今後は以上のような項目についての評価を進めていく予定である．

[参考文献]

[Arch86] James Archibald and Jean-loup Baer
'Cache Coherence Protocols:Evaluation Using a Multiprocessor Simulation Model' ACM Trans. on Compt.Vol.4,No.4,November 1986

[Baer87] Jean-Loup Baer and Wen-Hann Wang
'Architectural Choices for Multilevel Cache Hierarchies' in Proc. of International Conference on Parallel Processing 1987

[Borr84] Gaetano Borriello, et al.
'Design and Implementation of An Integrated Snooping Data Cache' UCB/CSD TR# 84/199 September 1984

[Good83] James R. Goodman
'Using Cache Memory to Reduce Processor-Memory Traffic' in 10th Annual International Symposium on Computer Architecture, june, 1983

[Shor88] Robert T.Short, Henry M.Levy
'A Simulation Study of Two-Level Caches' in Proc.of The 15th Annual International Symposium on Computer Architecture 1988

[Wils87] Andrew W. Wilson Jr.
'Hierarchical Cache/Bus Architecuter for Shared Memory Multiprocessors' in 14th Annual International Symposium on Computer Architecture, june, 1987

[山岡８８] 山岡　彰　他，
'Ｍ－６８Ｘにおけるメモリシステム' コンピュータアーキテクチャシンポジウム論文集，昭和６３年５月，情報処理学会