計算機アーキテクチャ 119－37
（1996. 8. 29）

スーパスカラ・プロセッサ用データ・キャッシュの実現方式の検討

佐々木 敬泰† 土江 竜雄†
弘中 哲夫† 児島 彰†

スーパスカラ・プロセッサでは，スーパスカラ度に見合う十分なデータ供給バンド幅を確保するため，ロード／ストア・ユニットの多重化を行う．しかし，同一サイクルに多重度分のロード／ストア命令を実行できなければ多重化したロード／ストア・ユニットに見合った性能向上は望めない．そこで，複数のロード／ストア命令に対応できるマルチポートのデータ・キャッシュの実現方式について検討する．

Study of Implimentation Method of Data Cache for Superscalar Processors

TAKAHIRO SASAKI†, TATSUO TUCHIE†, TETSUO HIRONAKA†
and AKIRA KOJIMA†

To achieve enough data bandwidth balanced with superscalar degree, the load/store units must be multiplied. However if we don't have enough data bandwidth to process the load/store instructions provided by the multiple load/store unit every clock cycle, the multiplied load/store unit will be useless. This paper discuss the technique to implement multiport data cache that supplies enough bandwidth for multipled load/store requests.

1. はじめに

現在，単一パイプライン・プロセッサによる設計開発は限界の兆しが見えはじめ，商用ベースでも並列処理を行うスーパスカラ・プロセッサや VLIW が注目を集めている．しかし，実行ユニットの並列性が向上するのに伴い，データのロード／ストアによるボトルネックがプロセッサの性能を妨げる大きな要因となっている．多重化した実行ユニットに見合うだけの十分なデータ供給を行うためには，ロード／ストア・ユニットを多重化する必要がある．多重化したロード／ストア・ユニットに対してマルチポート・メモリを用いて計算機を構成する方法も考えられるが，一般にマルチポート・メモリは高価であり，多重化したロード／ストア・ユニットに対して十分な容量を供給することは困難である．しかし，高価なマルチポート・メモリを用いる代わりに，シングルポート・メモリとマルチポート・キャッシュを用いれば，安価にシステムを構築することが可能である．

本研究では，多重化した実行ユニットに見合うだけのデータ供給のできるマルチポート・データキャッシュの実現方法として，シングルポート・メモリを用いたマルチポート化について検討する．

† 広島市立大学 情報科学部 情報工学科
Department of Computer Engineering, Faculty of Information Sciences, Hiroshima City University.

2. シングルポート・メモリを用いたマルチポート化

シングルポートメモリを用いてマルチポート・キャッシュを実現する方法として以下の4通りの方法 (D, IS, ID/D, ID/I) が提案されている [Noudomi93].

D(Duplicated) 型 (図 1)

データアレイおよびタグアレイの完全なコピーをポート毎に設ける．この方式では各ポートがデータアレイの完全なコピーを持つため，各ポートは任意のアドレスに対して自由にアクセスできる．しかし，この方法ではコヒーレンス制御の必要があり，実装が非常に困難である．図中の NET1 は，ロード／ストア命令のポート分別のためのビットフィールドを各ロード／ストア・ユニットに入力するためのものであり，LSU(n) は n 番目のロード／ストア・ユニットを，P(n) は n 番目のポートを，T はタグアレイのハードウェア量を，D はデータアレイのハードウェア量を表す．以下，特に断らない限りこの記号を用いる．

I(Interleaved) 型

データアレイをインターリーブする．これは更に，インターリーブされたデータアレイの各バンクをポート間で共有するか否かで，次の2つの選択肢に分かれる．

IS(Interleaved, Shared) 型 (図 2)

インターリーブされたデータアレイの各バンクをすべ

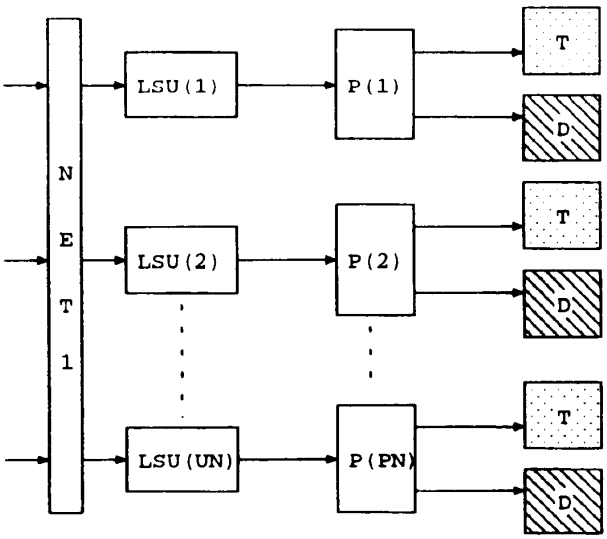


図1 D 型

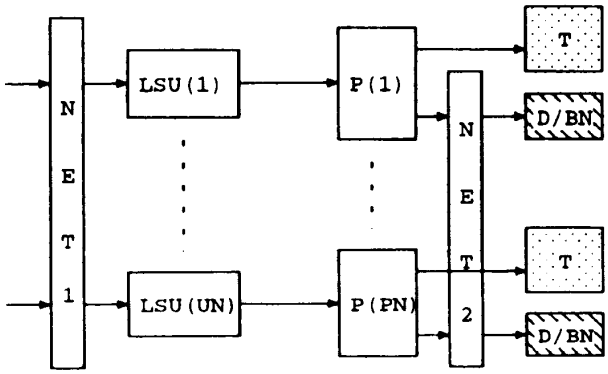


図2 IS 型

てのポートが共有する．タグアレイについては，ポート毎に完全なコピーを設ける．バンク数はポート数とは無関係に決まる．この方式ではラインサイズより小さい単位でインターリーブできるため，同時に連続したアドレスに対してアクセスできる．しかし，バンクコンフリクトの調停をロード／ストア・ユニットが受け付けた後行うので，タイミング制御が非常に難しく実装は困難である．図中の BN はバンク数を，NET2 は各ポートが任意アレイへアクセスするための相互接続のネットを表す．

ID(Interleaved, Dedicated) 型

インターリーブされたデータアレイの各バンクはいずれかのポートに占有される．よって，バンク数はポート数に等しい．各ポートはプロセッサのすべてのロード／ストア・ユニットに共有される必要がある．これは更にタグアレイをインターリーブするか否かで次の 2 つの選択肢に分かれる．

ID/D(ID/Duplicated) 型 (図 4)

タグアレイの完全なコピーをポート毎に設ける．インターリーブ幅はラインサイズとは無関係に決められるので，ワード単位でインターリーブすることで連続したデータアドレスに対して同時にアクセスすることができる．図中の UN はロード／ストア・ユニットの数を表す．インターリーブ幅とは，キャッシュの 1 ラインを何ワード毎に分けるかを表すもので，図 3 の例では 1 ラインを 1 ワード毎に 4 つに分けているので，インターリーブ幅は 1 （ワード）となる．

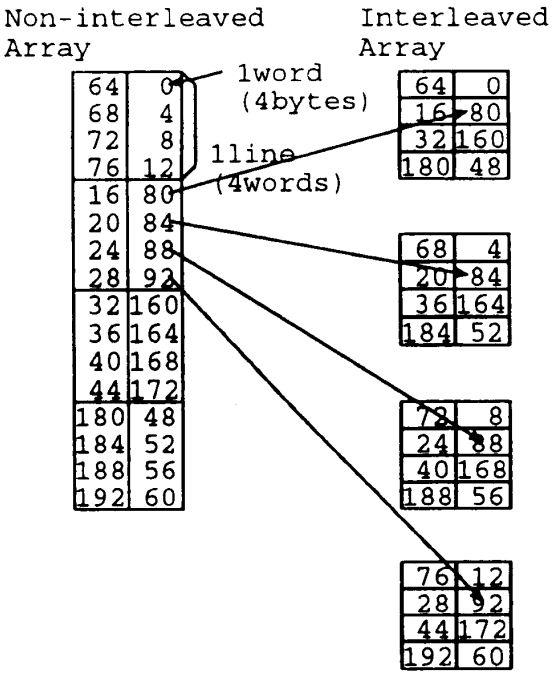


図3 インターリーブの例

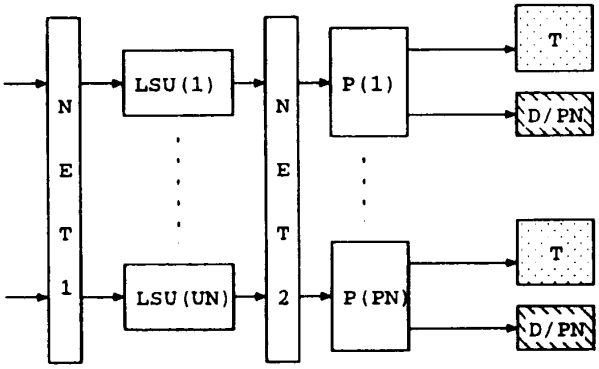


図4 ID/D 型

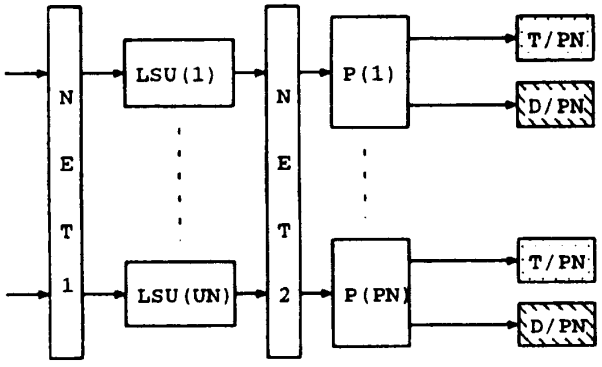


図5 ID/I 型

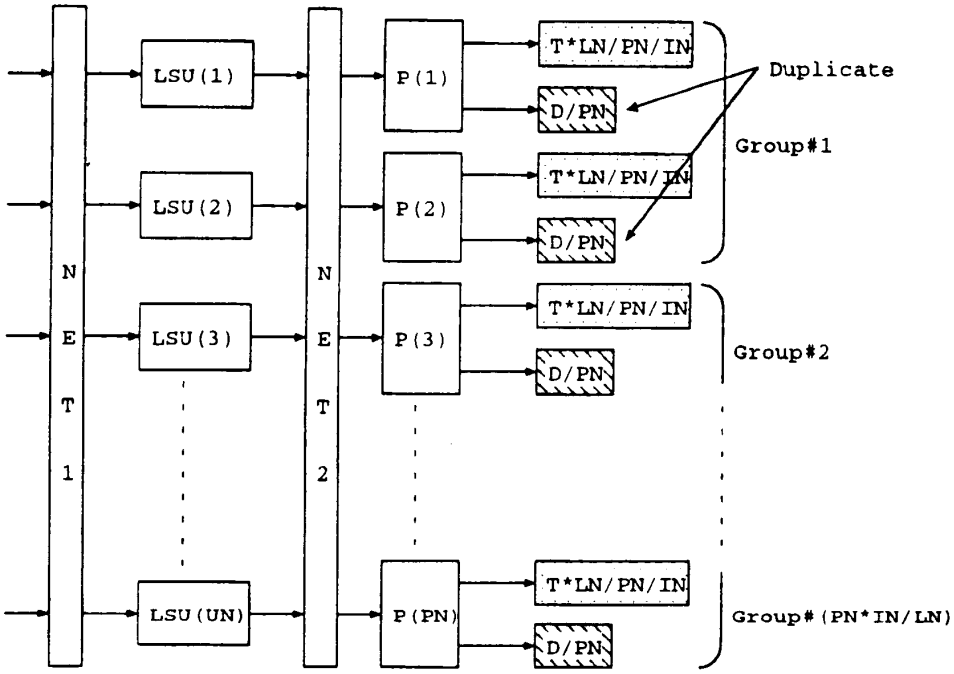


図6 ID/D/I 型

ID/I(ID/Interleaved) 型 (図 5)

タグアレイをデータアレイを同じ幅でインターリーブする．インターリーブされたタグアレイの各バンクは，対応するデータアレイ・バンクを占有しているポートに占有される．よって，インターリーブ幅はラインサイズに等しくなるので，連続したアドレスのデータに対して同時にアクセスすることはできない．

これら (D, IS, ID/D, ID/I 型) の方式はシングルポート・メモリでマルチポート・キャッシュを実現しているが，バンクコンフリクトの発生とミスヒット時の処理でトレードオフが発生する (表 1, 2 参照)．ID/I 型のようにタグアレイをインターリーブしてミスヒット時のリプレースによるロックを減少させると，連続アドレスのアクセスではインターリーブ幅が大きいためバンク・コンフリクトが発生しする．一方 ID/D 型のようにタグアレイをワード単位でインターリーブして同時に連続アドレスのアクセスを可能にさせるとミスヒット時にすべてのタグアレイの更新が必要になる．そこで，同時に連続アドレスのアクセスを可能にし，かつ効率的なデータ供給のできる方式，ID/D/I 型を新たに提案し，その後，各方式による比較を行う．

ID/D/I(ID/D/Interleaved) 型 (図 6)

先述の ID/D 型では連続アドレスに対するアクセスは高速であるが，ミスヒットを起こした時，すべてのタグアレイを更新する必要があった．これはプロセッサの並列度が上がり，ロード／ストア・ユニット数が多くなった時には重要な問題となる．ミスヒット処理の間は LOCKUP-FREE CACHE [Kroft81], [Sohi90] などで対処できるが，リプレース時にはすべてのロード／ストア・ユニットがロックしてしまう．そこで，同時に連続したアドレスのデータに対してアクセスでき，かつリプレース時にもすべてのポートがロックしないような構成，ID/D/I 型を提案する．これは，図 7 の様に ID/D 型を複数セット用意したもので，ID/D 型と同様に，同時に連続したアドレスのデータに対してアクセスでき，ミスヒットを起こした時も，リプレース時に他のグループに属するポートには何ら影響を与えることがない．図中のグループ (Group#n) は，1 ラインを構成するポートの集まりで，グループ 1 つが ID/D 型 1 つに相当する．また，同じグループ内のタグアレイは同じ内容も持つ．この方式は，構成上ポート数が多くなりがちなので，スーパスカラ度の高いプロセッサや VLIW など向けである．

この方式では，以下の特徴がある．

- インターリーブ幅がラインサイズと無関係に決められる (連続アドレスの同時読み出しが可能)
- ミスヒット時でも関係のないバンクに影響を与えない
- バンクコンフリクト調停後の制御はシングルポート・データキャッシュ並の容易さ
- 構成上ポート数が多くなる
- 上記に関連して，ロード／ストア・ユニットとポートを結ぶネットが大規模になる

3. バンクコンフリクト調停／コヒーレンス制御のためのユニット

文献 [Noudomi93] では D 型の具体的な構成，および IS, ID/D, ID/I 型のバンクコンフリクトの調停の手

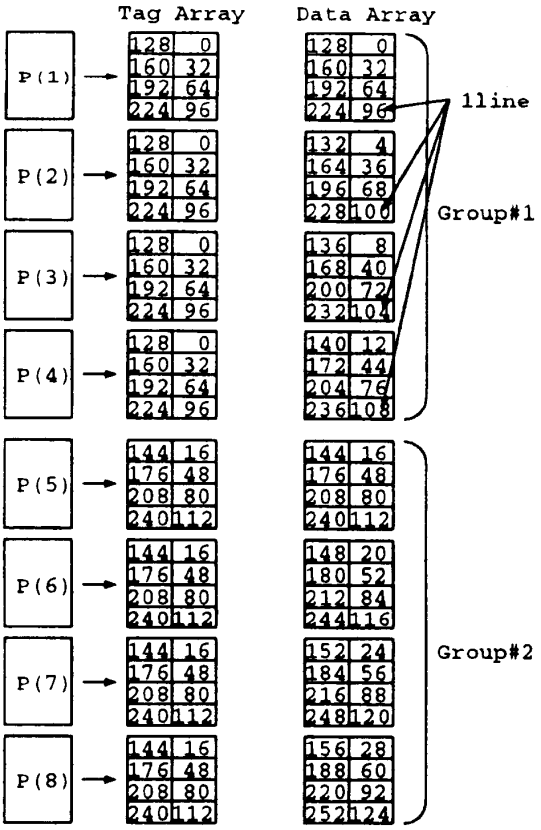


図7 ID/D/I 型のインターリーブ

段については一切触れていない．そこで，D, IS, ID/D, ID/I 型について，具体的な構成方式を検討すると共に，新規に提案した ID/D/I 型の構成方式についても検討する．

3.1 D 型のコヒーレンス制御

D 型では，タグアレイおよびデータアレ共に，各ポートが完全なコピーを持つため，バンクコンフリクトは発生しないが，全タグアレイが常に同じデータを持つ必要があるためコヒーレンス制御のためのユニットがなくてはならない．そこで，コヒーレンス制御を容易にするために，1 クロック内で実行できるストア命令を 1 つにするという制限をかける方式を提案し，これを Dp 型 (図 8) と呼ぶことにする．ここで，ストア命令と同クロック内でロード命令の実行を許すか否かで 2 通りに分かれる．

Dp1 型

1 クロックで，1 つのストア命令か，または複数のロード命令ができる．すなわち，ストア命令とロード命令が同時に実行されることはない．この方式では各ロード／ストア・ユニットは完全に独立して動作できるので，バンクコンフリクト・チェックのための比較器の必要がない．ストア命令の時は全データアレイに対して書き込みを行う．

Dp2 型

1 クロックで，1 つ以下のストア命令と，後続のロード命令を複数個実行できる．この方式では，ストア命令

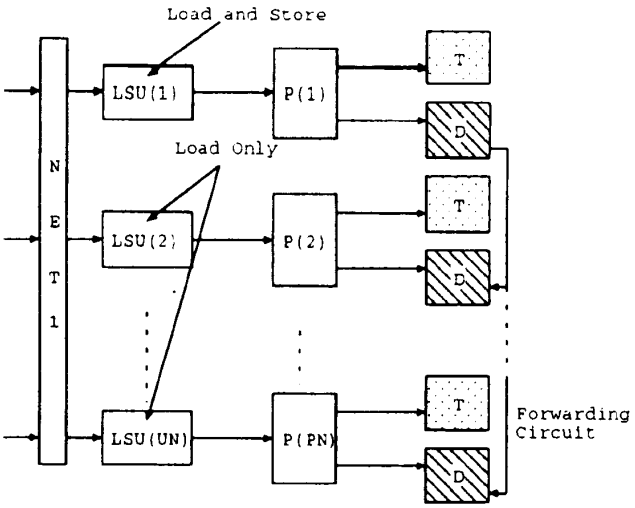


図8 Dp 型

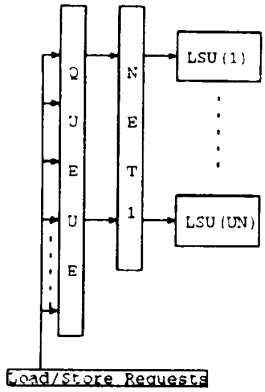
と後続のロード命令のアドレス比較をし，同一アドレスであればロード命令に対してフォワーディングをする必要がある．このためにアドレス空間に応じた分だけのビット幅の比較器が (ロード／ストア・ユニット数 − 1) 個と，フォワーディングのための回路が必要になる．ただし，バイトアクセスのストア命令でフォワーディングを許すと，ストアを許されているポート P(1) からのデータと，データアレイから取り込んだデータとバイト単位でのマージが必要になる．このことにより，ハードウェアが非常に大きくなるため，フォワーディングが有効なのはストア命令がワード単位のアクセスの時だけという制限をかけることにする．

3.2 IS 型のバンクコンフリクト調停

IS 型は，各ポートでロード／ストア命令を受け付けた後にバンクコンフリクトの調停を行うため，データアレイへのアクセスのタイミング制御が難しい [Noudomi93]. よって実装するにあたって現実的でないので，本論文ではこれ以上 IS 型の考察は行わない．

3.3 ID/D, ID/I 型のバンクコンフリクト調停

バンクコンフリクトの調停は，図 9 の様なユニットで行う．ID/D, ID/I 型では，あるアドレスの内容をもつ可能性のあるデータアレイは高々 1 つである．よって複数のロード／ストア・ユニットから同時にアクセス要求が来た場合は命令の発行順序に従い，ポートの割当を行う必要がある．図中の Xn にはロード／ストア・ユニットからアクセス要求のあったデータのメモリアドレスの一部を入れる．ここで Xn に渡すデータは，図 10 の Cache Bank フィールドで，ID/D, ID/I によってビット位置は異なる．Pn は n 番目のポートが保持するメモリアドレスの一部で，ビット位置は図 10 の Cache Bank フィールドの位置と同じである．もし，n 番目のポートへのアクセス要求を出しているロード／ストア・ユニットが 1 つであれば，そのロード／ストア・ユニットがポートへのアクセス権を得るが，2 つ以上のロード／ストア・ユニットから要求があった場合は Priority



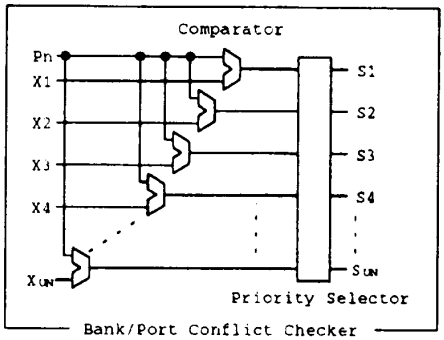


図9 バンクコンフリクトの調停とキュー

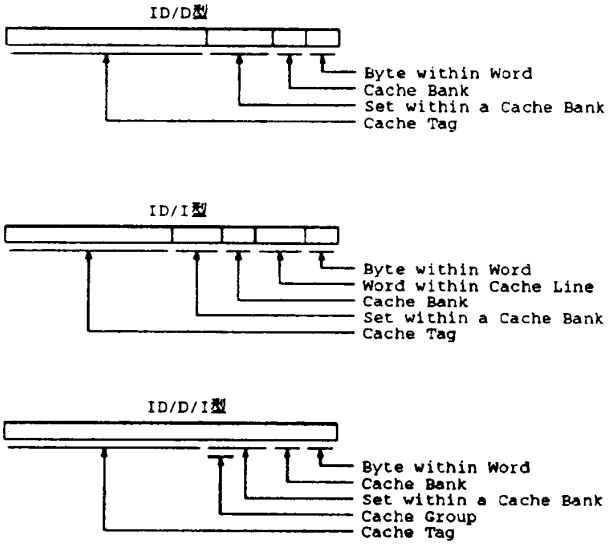


図10 フィールド構成

Selector により，最も先行する命令がアクセス権を得る．この時ポートへのアクセス権を得た命令は実行され，キューから消去されるが，アクセス権を得ることができなかった命令は次クロックで再び調停を行う．このような仕組みを用いることでロード／ストア命令の実行順序の保証を行うことができる．

3.4 ID/D/I 型のバンクコンフリクト調停

ID/D/I 型のバンクコンフリクトの調停も ID/D 型や ID/I 型と同様に行うことができる．Xn に渡すデータは ID/D 型と同じである．

4. 各構成方式の比較および評価

以上の 5 通りの方式について検討する．各構成方式の特徴をまとめたものを表 1, 2, 3 に示す．

4.1 バンクコンフリクトの影響

D 型では各ポートがデータアレイの完全なコピーを持つため，バンクコンフリクトは発生しない．以下，IS/ID/D,ID/I,ID/D/I 型の4 方式についてバンクコンフリクトを抑える手段について考察する．IS,ID/D,ID/I の 3 方式については以下のような手法が考案されている [Noudomi93].

IS 型 ポート数とバンク数の間に相応関係がないので，ポート数を増やすことなくバンク数を増やすことができる．また，インターリーブ幅に関する制約もないので，インターリーブ幅を小さくすることもできる．

ID/D 型 ポート数とバンク数が等しいので，バンク数を増やすとそれに従ってハードウェアが増え，結果全体のハードウェア量の増加を招くため，むやみにバンク数を増やすことはできない．インターリーブ幅に関する制約はないので，インターリーブ幅を小さくすることが可能である．

ID/I 型 ポート数とバンク数が等しいので，むやみにバンク数を増やすことができない．また，インターリーブ幅はラインサイズに等しいので，インターリーブ幅を小さくすることもできない．

今回新たに提案した ID/D/I 型については，以下の手法でバンクコンフリクトを抑えることができる．

ID/D/I 型 ポート数とバンク数が等しいので，バンク数を増やすとそれに従ってハードウェアが増え，結果全体のハードウェア量の増加を招くため，むやみにバンク数を増やすことはできない．ただし，ID/D 型とは違い，構成上バンク数は多くなりがちなので，スーパスカラ度の大きいプロセッサや VLIW 向けである．インターリーブ幅に関する制約はないので，インターリーブ幅を小さくすることが可能である．

4.2 ミスヒット時の影響

すべてのポートにタグアレイの完全なコピーを設ける D, IS, ID/D 型では，ミスヒットが起きた場合，すべてのタグアレイを更新する必要がある．さらに D 型ではすべてのデータアレイの更新も必要である．一方 ID/I 型では，タグアレイ／データアレイともにインターリーブしているため，ミスヒットを起こしても影響を受けるバンクは 1 つだけである [Noudomi93] .

ID/D/I 型では，グループ内では同じタグアレイを持っているため，ミスヒットを起こした時にグループ内のすべてのタグアレイを更新する必要があるが，他のグループに属するバンクは何ら影響を受けない．これは，LOCKUP-FREE CACHE を導入した場合でもリプ

ポート数	PN	ウェイ数	WN	ライン長	LN	ロード／ストアユニット数	UN
バンク数	BN	セット数	SN	インターリーブ幅	IN	1 ライン分のタグサイズ	TN

表 1　マルチポートキャッシュの構成方式の比較 1

タイプ	バンクコンフリクト	コヒーレンス制御	タグアレイのインターリーブ	データアレイのインターリーブ	同一ラインの同時アクセス
D	なし	必要	しない	しない	可
IS	あり	不要	しない	する	可
ID/D	あり	不要	しない	する	可
ID/I	あり	不要	する	する	不可
ID/D/I	あり	不要	する	する	可

表 2　マルチポートキャッシュの構成方式の比較 2

タイプ	ミスヒット時にリプレースするタグアレイ	バンク数を増やした時の効果	インターリーブ幅
D	すべて	バンク数はユニット数と同じ	インターリーブしない
IS	すべて	バンクコンフリクトを減らせる	ラインサイズと無関係
ID/D	すべて	バンク数はポート数と同じ	ラインサイズと無関係
ID/I	該当アレイ	バンク数はポート数と同じ	ラインサイズに等しい
ID/D/I	該当アレイを含む $\frac{LN}{IN}$ 個のアレイ	バンク数はポート数と同じ	ラインサイズと無関係

表 3　マルチポートキャッシュの構成方式の比較 3

タイプ	ポート／バンクコンフリクト調停のためのユニット	タグアレイのサイズ	データアレイのサイズ	パラメータの制約
D	32ビット比較器 $\times (UN-1)$ ※	$PN \times TN \times SN \times WN$	$PN \times LN \times SN \times WN$	$PN = BN = UN$
IS	$log(BN)$ビット比較器 $\times UN \times PN$ UNビットアービタ $\times BN$	$PN \times TN \times SN \times WN$	$LN \times SN \times WN$	$BN = UN, IN \leq LN$
ID/D	$log(PN)$ビット比較器 $\times UN \times PN$ UNビットアービタ $\times BN$	$PN \times TN \times SN \times WN$	$LN \times SN \times WN$	$PN = BN, IN \leq LN$
ID/I	$log(PN)$ビット比較器 $\times UN \times PN$ UNビットアービタ $\times BN$	$TN \times SN \times WN$	$LN \times SN \times WN$	$PN = BN, IN = LN$
ID/D/I	$log(PN)$ビット比較器 $\times UN \times PN$ UNビットアービタ $\times BN$	$\frac{LN}{IN} \times TN \times SN \times WN$	$LN \times SN \times WN$	$PN = BN, IN \leq LN$

※ Dp1 型の場合は必要なし

レース中でも他のグループに属するバンクが使用できるという効果がある．

5. おわりに

以上，今後重要視されてくるであろうスーパスカラ度の高いプロセッサや VLIW でのデータキャッシュの構成について述べた．ロード／ストア・ユニットの多重化にともない，バンクコンフリクトによる性能低下，リプレース時によるユニットのロックなど様々な問題が生じる．本論文ではこれらの問題の解消方法と，それにより発生するトレードオフについて述べ，両問題を解消する新しい手段，ID/D/I 型の提案を行った．

今後は，様々なアーキテクチャのシミュレータ [Tuchie96] を使用して，各方式の実質的な性能差を評価していく予定である．また本論文中で行っていない，LOCKUP-FREE CACHE との連携による性能評価も今後の課題である．

参考文献

[Noudomi93] 納富 昭，久我守弘，村上和彰，富田眞治:“DSN 型スーパースカラ・プロセッサ・プロトタイプのロード／ストア・パイプライン,” 情処研報 ARC-86-4, 1991.

[Kroft81] D. Kroft：“Lockup-free Instruction Fetch / Prefetch Cache Organization,” Proc. 8th Ann. Int'l Symp. Comput. Architect., 1981.

[Sohi90] Sohi, G. and Franklin, M.：“High-Bandwidth Data Memory System for Superscalar Processors,”Computer Science Department University of Wisconsin-Madison, Computer Sciences Technical Report #968, Sep. 1990.

[Tuchie96] 土江 竜雄ほか:“教育研究用スーパスカラ・プロセッサ・シミュレータ Mikage の概要,” 情処研報, ARC(SWoPP96), 1996.