計算機アーキテクチャ　113－1
（１９９５．　８．　２３）

次々世代汎用マイクロプロセッサ・アーキテクチャ *PPRAM*の概要

岩下茂信　　宮嶋浩志　　村上和彰

九州大学 大学院総合理工学研究科 情報システム学専攻

〒816 福岡県春日市春日公園 6–1

E-mail: ppram@is.kyushu-u.ac.jp
http://www.is.kyushu-u.ac.jp/ ppram/

21 世紀初頭の製品化を目指して，新しい汎用マイクロプロセッサ・アーキテクチャ*PPRAM*（*Parallel Processing Random Access Memory/Practical Parallel Random Access Machine*）を提案している．*PPRAM* とは，一言で言えば「大容量メモリおよび複数のプロセッサを１チップに集積し，分散メモリ型マルチプロセッサ構成により本質的に高いチップ内メモリ・バンド巾を活用すると同時に，グローバル・レジスタ・ファイルを各プロセッサが共有することでチップ内プロセッサ間での超低レイテンシ通信／同期を可能にしたオンチップ・マルチプロセッサ・アーキテクチャ」である．本稿では，個々のインプリメンテーション（＝アーキテクチャ）に依存しない，*PPRAM* のアーキテクチャ上の枠組（*architectural framework*）について述べている．

PPRAM: A 21st Century's Microprocessor Architecture — Architectural Framework Brief —

Shigenobu IWASHITA　　Hiroshi MIYAJIMA
Kazuaki MURAKAMI

Department of Information Systems
Interdisciplinary Graduate School of Engineering Sciences
Kyushu University
6–1 Kasuga-koen, Kasuga-shi, Fukuoka 816 Japan

E-mail: ppram@is.kyushu-u.ac.jp
http://www.is.kyushu-u.ac.jp/ ppram/

This paper describes the *architectural framework* of *PPRAM*(*Parallel Processing Random Access Memory/Practical Parallel Random Access Machine*), a 21st century's microprocessor architecture. *PPRAM*is an on-chip multiprocessor integrated with a large size of DRAM. Memory is distributed for each processor as local memory (i.e., distributed-memory multiprocessor) to exploit its inherent high memory bandwidth. Memory itself consists of 2-dimensional array of memory cells, and a whole row of the memory array (e.g., 1024 bits) can be read and written at once. Each processor provides more than one row buffer which cantains the contents of a row of the memory array. Some of these row buffers works as a sort of cache memory, so that each buffer corresponds to a cache line. A pair of a processor and its local memory (PE) is interconnected with other PEs inside and outside the chip through some interconnection network. Inside a chip, all the processors share a register file, called GRF (Global Register File). Each processor is a sort of RISC processor, and it has its own (local) register file. The GRF provides a low-latency communication and synchronization among processors on the same chip.

1 はじめに – *PPRAM* とは –

我々は，21世紀初頭の製品化を目指して，新しい汎用マイクロプロセッサ・アーキテクチャ*PPRAM*（*Parallel Processing Random Access Memory*, *Practical Parallel Random Access Machine*）を提案している [1, 2].

PPRAM とは，一言で言えば「大容量メモリおよび複数のプロセッサを1チップに集積し，分散メモリ型マルチプロセッサ構成により本質的に高いチップ内メモリ・バンド巾を活用すると同時に，グローバル・レジスタ・ファイルを各プロセッサが共有することでチップ内プロセッサ間での超低レイテンシ通信／同期を可能にしたオンチップ・マルチプロセッサ・アーキテクチャ」である.

本稿では，個々のインプリメンテーション（＝アーキテクチャ）に依存しない，*PPRAM* のアーキテクチャ上の枠組（*architectural framework*）[1]について述べる．まず，2章で *PPRAM* の特長を列挙した後，3章でメモリについて，4章でグローバル・レジスタについて，そして，5章で相互結合網を用いたPE間通信について，それぞれ述べる.

2 特長一覧

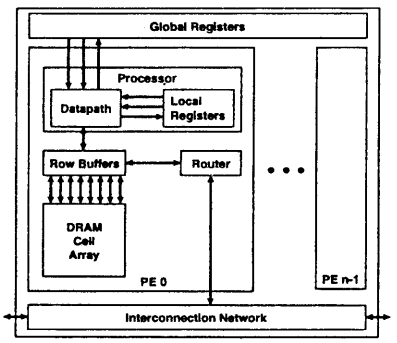

図 1: *PPRAM* の論理構成

PPRAM は，以下の特長を有する.

新しい計算機システム構成法のための汎用機能部品

- 1971年のマイクロプロセッサ誕生以来続けられてきた「マイクロプロセッサ・チップとメモリ・チップ」という『分業（＝分チップ）』体制に基づくシステム構成法に取って替わる，
  - 数1000万 Tr/マイクロプロセッサ・チップ
  - 数 Gb/メモリ・チップ

  時代の新しいシステム構成法
- すなわち,「メモリ+プロセッサ」チップ（=*PPRAM*）を基本構成要素として，所望の記憶容量および処理性能を満たすまで当該チップを繰り返し配置することで計算機システムを構成

大容量 DRAM とプロセッサをモノリシック化

- 「メモリが主で，プロセッサは従」を基本方針に，まずその時代で最大容量の DRAM を搭載し，そのメモリ容量およびメモリ・バンド巾に見合うだけのプロセッサ能力を1チップに集積化（*monolithic processor-memory chip*）
- メモリとプロセッサを1チップ化したことにより，非常に大きなオンチップ・メモリ・バンド巾を活用可
- 同時に，要求されるオフチップ・メモリ・バンド巾を低減可

分散メモリ型オンチップ・マルチプロセッサ

- 「プロセッサは質より量」を基本方針に，出来るだけシンプル，かつ，機能的には完備な RISC プロセッサをチップ内メモリ容量およびオンチップ・メモリ・バンド巾に見合っただけ，かつ，出来るだけ多数搭載
- 大きなオンチップ・メモリ・バンド巾を活用するために，すべてのメモリをローカル・メモリとしてプロセッサ毎に分散配置する**分散メモリ型マルチプロセッサ構成**を採用（図1および図2参照）
- 共有メモリ型プログラミングをサポートするために，**単一グローバル・アドレス空間**および**仮想共有メモリ支援機構**を装備
- PE（Processor Element：プロセッサとローカル・メモリの対）間は，相互結合網とグローバル・レジスタ・ファイルの2手段により結合
- 同一のプロセッサから成る**均質型マルチプロセッサ構成**に加えて，機能的に異なるプロセッサの混載を許した**非均質型マルチプロセッサ構成**も可能
- マルチプロセッサを1チップ化したことにより，グローバル・レジスタ・ファイルを介した超低レイテンシ通信／同期が可能

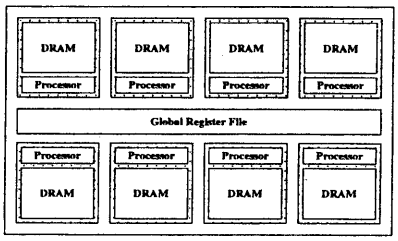

図 2: チップ・レイアウト例 (8PE の場合)

[1] *PPRAM* という用語は，ある特定のプロセッサあるいはプロセッサ・アーキテクチャを指す個体名称ではなく，たとえばRISC, VLIW，スーパースカラといった用語に相当する集合名称である．たとえば，RISC においては，(i) 単一固定長命令，(ii) キャッシュの存在を前提にしたロード／ストア・アーキテクチャ，(iii) 命令パイプラインに適した命令セット，等の条件が RISC のアーキテクチャ上の枠組を与えている.

超高オンチップ・メモリ・バンド巾を活用する複数の行バッファ

- 2次元メモリ・セル・アレイの1行分[2]を保持する行バッファ（row buffer）をローカル・メモリとプロセッサとの間に複数個装備
- 1メモリ・アクセス・サイクルでローカル・メモリと行バッファとの間で1行分のデータ転送が可能
- これら行バッファは，キャッシュ・メモリ，および，相互結合網を介したDMA転送用バッファとして使用

グローバル・レジスタ・ファイルを介したチップ内超低レイテンシ通信／同期

- すべてのプロセッサが，論理的には1個のグローバル・レジスタ・ファイルを共有（図3に示したレジスタ空間の例を参照）
- 物理的には，グローバル・レジスタ・ファイルの完全なコピーを各プロセッサ毎に複製配置し，それらを共有バスで結合
- 各プロセッサは，通常のレジスタ・ファイル（＝ローカル・レジスタ・ファイル）同様，グローバル・レジスタ・ファイル内の任意のグローバル・レジスタ[3]にアクセス可
- OSの許可の下，ユーザ・プログラムがグローバル・レジスタ・ファイルに直接アクセスすることが可
- 各グローバル・レジスタに付加したFull/Emptyビットを用いて，グローバル・レジスタを以下の用途に使用可能
  - 共有/同期/ロック変数（排他制御を保証）
  - チャネル変数（生産者－消費者間同期を保証）
- グローバル・レジスタへの書込みをトリガにした他プロセッサへの割込みが可能（アクティブ・メッセージの実現）

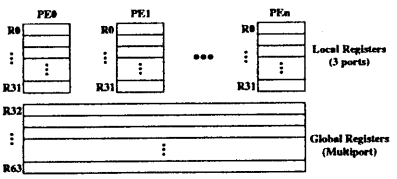

図 3: レジスタ空間例

相互結合網を用いたチップ内＆チップ間DMA転送

- チップ内相互結合網のトポロジーはインプリメンテーション依存で，1～3次元アレイ，あるいは，ハイパーキューブのいずれか（図4の2次元アレイの例を参照）
- チップ間相互結合網のトポロジーもインプリメンテーション依存で，1～2次元アレイのいずれか（図5の2次元アレイの例を参照）
- フロー制御はワームホール・ルーティングあるいはバーチャル・カット・スルー
- アドレス変換機構で保護されたPUT/GETインタフェースをユーザ・プログラムに直接提供

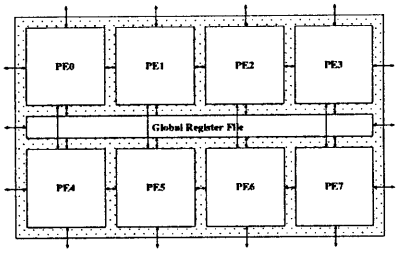

図 4: チップ内PE接続例（2次元アレイの場合）

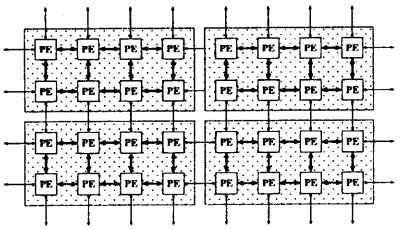

図 5: チップ間接続例（2次元アレイの場合）

RISCを基本に命令セットを拡張

- 基本命令セットは，32/64ビットRISC命令セット[4]
  - 固定単一命令長
  - ロード／ストア・アーキテクチャ
- *PPRAM*拡張命令セット
  - グローバル・レジスタに対するアクセス命令
    - 少なくとも，ローカル・レジスタとグローバル・レジスタとの間のMOVE命令，さらにインプリメンテーション次第では，
    - グローバル・レジスタに対するLOAD/STORE命令
    - グローバル・レジスタを用いた算術論理演算命令および分岐命令
  - ローカル・メモリからキャッシュ（行バッファ群）へのREAD命令，および，キャッシュからローカル・メモリへのWRITE命令
  - ローカル・メモリとリモート・メモリとの間のPUT/GET命令
  - 自プロセッサへのクロック信号供給を停止するSLEEP命令

[2] インプリメンテーションによるが，512～1024ビット程度．
[3] インプリメンテーションによるが，最大で3個（ソース・レジスタ2個とデスティネーション・レジスタ1個）．
[4] インプリメンテーションによるが，既存のRISC命令セットないしCISCコア命令セットを基本命令セットにすることも可．

消費電力をプログラム制御可能とした低消費電力化マイクロプロセッサ

- SLEEP 命令の実行により自プロセッサへのクロック信号供給を停止可
- スリープ（クロック信号供給停止）中のプロセッサへの割込みにより，当該プロセッサへのクロック信号供給を再開
- システム・プログラムは，ユーザの指示により受動的に，あるいは，処理の負荷や並列度に応じて能動的に，スリープすべきプロセッサ数を決定

現在のスーパースカラ処理から最終的なメッセージ交換型並列処理までシームレスな移行

- 各PEをスーパースカラの機能ユニットと見なすことで，*PPRAM* を均質型／非均質型スーパースカラ・プロセッサとして利用可
- ランタイム・システムにより仮想共有メモリを実現することで，共有メモリ型マルチプロセッサ上で単一アドレス空間を前提に開発されたアプリケーション・プログラムを *PPRAM* 上に移植可
- PUT/GET インタフェースを用いた低オーバヘッドなメッセージ交換型並列プログラムの開発，あるいは，コンパイラによる生成が可能

3 メモリ

PPRAM は分散メモリ構成を採るが，その物理アドレス空間としてはチップ内外を問わず単一のグローバル・アドレス空間を構成する．

*PPRAM*のある1プロセッサに着目すると，*PPRAM* を構成要素とするシステムは全体として以下の階層メモリ構成を採る．

1. レジスタ：ローカル・レジスタおよびグローバル・レジスタ
2. キャッシュ（行バッファ群）
3. ローカル・メモリ
4. リモート・メモリ（他プロセッサのローカル・メモリ）：オンチップおよびオフチップ
5. 補助記憶（ハードディスク，等）

上記の各階層間に設けられるデータ転送パスは，以下の通り．

- ローカル・レジスタ↔グローバル・レジスタ
- レジスタ←(キャッシュ)→ローカル・メモリ
- ローカル・メモリ↔リモート・メモリ/補助記憶

ここで，分散共有メモリ構成（CC-NUMA ないし COMA）は採用していないので，

- レジスタ←(キャッシュ)→リモート・メモリ
- レジスタ←(キャッシュ)→ローカル・メモリ↔リモート・メモリ

というデータ転送パスがハードウェアでは直接用意されていない点に注意されたい．

上記の3種類のデータ転送パスに関して，

- ローカル・レジスタ↔グローバル・レジスタは4章で，
- レジスタ←(キャッシュ)→ローカル・メモリは本章で，
- ローカル・メモリ↔リモート・メモリ/補助記憶は5章で，

それぞれ述べる．

3.1 ローカル・メモリおよびキャッシュの構成

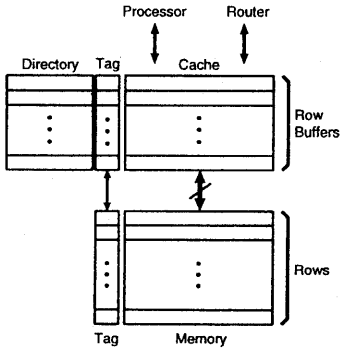

図 6: ローカル・メモリおよびキャッシュの構成例

図6にローカル・メモリおよびキャッシュの構成例を示す．

ローカル・メモリは，メモリ・セルを要素とする2次元アレイ構造となる．その1行分を保持する行バッファを複数個設けて，これをキャッシュとする．すなわち，1個の行バッファが1キャッシュ・ラインに相当し，キャッシュとローカル・メモリとの間の転送単位は行となる．

キャッシュの構成に関して，

- 命令キャッシュとデータ・キャッシュを分離するか統合するか？
- 論理アドレス・キャッシュとするか物理アドレス・キャッシュとするか？
- 連想度に関して，ダイレクト・マップ，セット・アソシアティブ，フルアソシアティブのどれにするか？

等はインプリメンテーションに依存する．ライト・ポリシーに関しては，行バッファという性格上，ライト・スルーよりもライト・バックの方が適している．

なお，図6では，キャッシュに対してプロセッサとルータの双方がアクセスする形態，すなわち，キャッシュが DMA 転送用バッファも兼ねた構成となっているが，両者を分離した構成も可能である．

3.2 仮想共有メモリ支援機構

PPRAM は分散メモリ型マルチプロセッサであるが，ランタイム・システムによる仮想共有メモリ [3] の実現を支援する機構を備える．これは，Tempest[6] の fine-grain access control と等価の機能を提供する．

すなわち，ローカル・メモリの行単位に以下の3状態を示すタグを設ける．

Invalid 当該行は無効．

Read-Only 当該行は有効で，読出しのみ可．

Read-Write 当該行は有効で，読出し/書込み可．

各行バッファにも同様のタグを設け，行を行バッファにキャッシングする際に対応するタグの内容をコピーする．

メモリ・アクセス命令を実行する度に対応するタグの内容をチェックし，以下に示す組合せに対しては割込みを起こす．

	Invalid	Read-Only	Read-Write
LOAD	割込み		
READ	割込み		
STORE	割込み	割込み	
WRITE			

3.3 メモリ・アクセス命令

メモリ・アクセス命令として，以下の命令を定義する．

LOAD キャッシュあるいはローカル・メモリからレジスタへデータをロードする．当該データを含む行が行バッファに存在しない（キャッシュ・ミス）場合，当該行を行バッファにキャッシングする．

READ ローカル・メモリの指定された行を行バッファにキャッシングする．

STORE レジスタからキャッシュへデータをストアする．当該データをストアすべき行が行バッファに存在しない（キャッシュ・ミス）場合，当該行を行バッファにキャッシング（ライト・アロケート）した後ストアする．

WRITE 行バッファからローカル・メモリに指定された行を書き戻す．

さらに，制御命令として，タグ設定命令を備える．

4 グローバル・レジスタ

4.1 グローバル・レジスタ・ファイルの構成

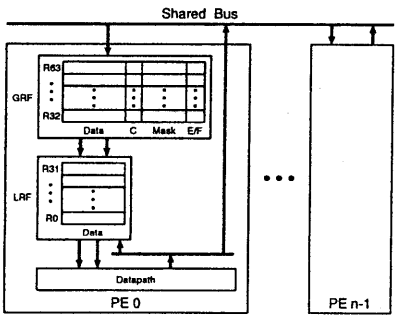

図 7: レジスタ・ファイルの構成例

図7に，ローカルおよびグローバル・レジスタ・ファイルの構成例を示す．

ローカル・レジスタ・ファイルは，通常のレジスタ・ファイルと同様の構成を採る．一方，グローバル・レジスタ・ファイルに関しては，その完全なコピーを各プロセッサ毎に複製配置し，それらを共有バスで結合した構成を採る．あるプロセッサがあるグローバル・レジスタに書込みを行なう際には，共有バスを介して当該書込み値を全プロセッサに放送して，各プロセッサの当該グローバル・レジスタに書き込む．これにより，グローバル・レジスタ・ファイルのコピー間のコンシステンシを維持し，すべてのプロセッサが論理的には1個のグローバル・レジスタ・ファイルを共有しているように見せかける．

図7に示すように，各グローバル・レジスタはデータ格納用のフィールド以外に，次の3種類のフィールドを備える．

C (Capability) ユーザ・モードにおける当該グローバル・レジスタへのアクセスが許可されているか否かを示す．

Mask チップ内PE数に等しいビット長のマスク．対応するプロセッサからの当該グローバル・レジスタへの書込みが許可されているか否かを示す．このマスクの設定次第で，同一番号のグローバル・レジスタを複数のグループ[5]に分けて使用することが可能となる．

F/E Full/Empty ビット．

なお，ローカルおよびグローバル・レジスタのレジスタ数ならびにレジスタ長はインプリメンテーションに依存する．

4.2 グローバル・レジスタに対するアクセス操作

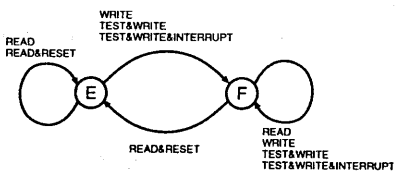

図 8: F/E 状態遷移

グローバル・レジスタに対するアクセス操作 (operation) として，以下の操作を定義する（表1および図8参照）．

READ

- F/E=E の場合：何もしない．
- F/E=F の場合：グローバル・レジスタから値を読み出す（F/E は F のまま）．

[5] たとえば，アプリケーション毎にグループを形成する．

表 1: グローバル・レジスタに対するアクセス操作

	Empty	Full
READ	失敗	読出し
READ&RESET	失敗	読出し&F/E←E
WRITE	書込み&F/E←F	
TEST&WRITE	書込み&F/E←F	失敗
TEST&WRITE&INTERRUPT	書込み&F/E←F&割込み	失敗

READ&RESET
- F/E=E の場合：何もしない．
- F/E=F の場合：グローバル・レジスタから値を読み出し，F/E を E にする．

WRITE F/E の値に関わらず，グローバル・レジスタに値を書き込み，F/E を F にする．

TEST&WRITE
- F/E=E の場合：グローバル・レジスタに値を書込み，F/E を F にする．
- F/E=F の場合：何もしない．

TEST&WRITE&INTERRUPT
- F/E=E の場合：グローバル・レジスタに値を書込み，F/E を F にする．同時に当該書込みの行なわれたグローバル・レジスタを有するプロセッサに対して割込みをかける．当該書込み値が割込み先アドレスとなる．
- F/E=F の場合；何もしない．

グローバル・レジスタに対するアクセス命令は，上記の操作を基にインプリメンテーション毎に別途定義する．少なくとも，ローカル・レジスタとグローバル・レジスタとの間の MOVE 命令が備わっていれば良い．さらに，インプリメンテーション次第では，

- ローカル・メモリとグローバル・レジスタとの間の LOAD/STORE 命令
- グローバル・レジスタを用いた算術論理演算命令および分岐命令

を備えても良い．

4.3 グローバル・レジスタの用途

グローバル・レジスタの用途としては，以下のものがある．

- 共有/同期/ロック変数
- チャネル変数

4.3.1 共有/同期/ロック変数

グローバル・レジスタを共有変数，同期変数，ロック変数，等の（広義の）共有変数として使用する．この時，F/E ビットをロック・ビットとして用いることで，共有変数単位での排他制御を保証する．以下に，共有変数に対するアクセス・プロトコルを示す（図 9参照）．

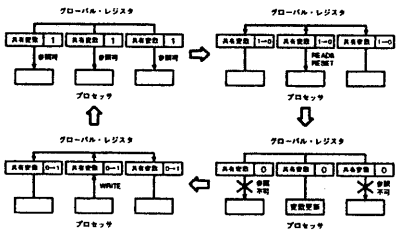

図 9: 共有/同期/ロック変数としての用途

共有変数参照 READ 操作を実行．

- F/E=E の場合：読出し失敗，エスケープ・シーケンス[6]へ．
- F/E=F の場合：読出し成功，次ステップへ．

共有変数更新

1. READ&RESET 操作を実行．
   - F/E=E の場合：読出し&ロック失敗，エスケープ・シーケンスへ．
   - F/E=F の場合：共有変数の値を読み出すと同時に，F/E を E にして当該共有変数をロックする．以降，アンロックされるまで，当該共有変数に対する READ 操作および READ& RESET 操作はすべて失敗する．
2. 読み出した値を用いての更新処理．
3. WRITE 操作により，更新した値を共有変数に書き戻し，F/E を F にして当該共有変数をアンロックする．

4.3.2 チャネル変数

グローバル・レジスタを通信用のチャネル変数として使用する．この時，F/E ビットにより，生産者－消費者間同期を保証する．以下に，通信プロトコルを示す（図 10参照）．

送信側 TEST&WRITE 操作を実行．

- F/E=E の場合：送信成功，次ステップへ．
- F/E=F の場合：送信失敗，エスケープ・シーケンスへ．

[6]エスケープ・シーケンスにてその後の対処法を決定する．

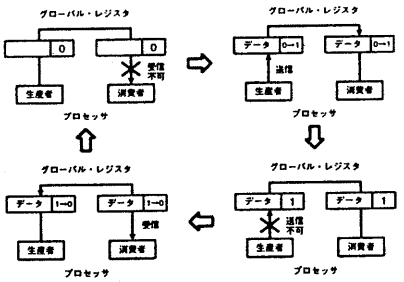

図 10: チャネル変数としての用途

受信側 READ&RESET 操作を実行.

- F/E=E の場合：受信失敗，エスケープ・シーケンスへ.
- F/E=F の場合：受信成功，次ステップへ.

さらに，チャネル変数を介してアクティブ・メッセージ [5] を送受信することも可能である．以下に，その通信プロトコルを示す.

送信側

1. TEST&WRITE&INTERRUPT 操作を実行.
   - F/E=E の場合：割込み成功.
   - F/E=F の場合：割込み失敗，エスケープ・シーケンスへ.
2. TEST&WRITE 操作を実行.
   - F/E=E の場合：送信成功，次ステップへ.
   - F/E=F の場合：送信失敗，エスケープ・シーケンスへ.

受信側

1. 割込み発生，ハンドラ起動.
2. READ&RESET 操作を実行.
   - F/E=E の場合：受信失敗，エスケープ・シーケンスへ.
   - F/E=F の場合：受信成功，次ステップへ.

5 PE 間通信

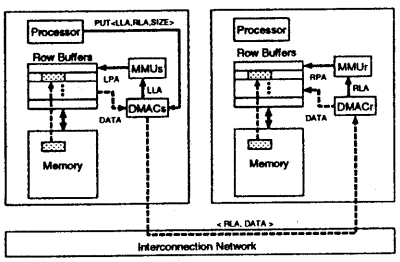

図 11: PUT 命令の実行

PPRAM は，PE 間通信手段として次の 2 種類を備える.

- グローバル・レジスタ上に設けたチャネル変数を介した同一チップ内 PE 間通信
- チップ内およびチップ間の相互結合網を介したローカル・メモリとリモート・メモリ（オンチップおよびオフチップ）との間の DMA 転送

前者については，4.3.2項で述べた．本章では，後者について述べる.

DMA 転送手段として，アドレス変換機構で保護された PUT/GET インタフェース [4] をユーザ・プログラムに直接提供する．PUT/GET インタフェースとして，少なくとも以下の 2 命令を定義する．なお，以下の記述では，キャッシュは，物理キャッシュ，かつ，DMA 転送用バッファも兼ねた構成（図 6参照）を仮定している.

PUT 図 11参照.

1. PUT 命令を実行するプロセッサ（PUT 元プロセッサ）が，ルータ内の DMA コントローラ（$DMAC_s$）に対して，ローカル論理アドレス（LLA），リモート論理アドレス（RLA），および，転送サイズを渡す.
2. DMA コントローラ（$DMAC_s$）は，メモリ管理ユニット（MMU_s）を用いてローカル論理アドレス（LLA）をローカル物理アドレス（LPA）に変換する.
3. 続いて，DMA コントローラ（$DMAC_s$）は，ローカル物理アドレス（LPA）を用いて所望のデータをキャッシュから読み出す．キャッシュがミスヒットした場合は，所望のデータを含む行をローカル・メモリからキャッシュにコピーして，当該データを読み出す.
4. ルータにより，PUT 先 PE の DMA コントローラ（$DMAC_r$）に対して，リモート論理アドレス（RLA）および当該データを送信する.
5. PUT 先 PE の DMA コントローラ（$DMAC_r$）は，メモリ管理ユニット（MMU_r）を用いてリモート論理アドレス（RLA）をリモート物理アドレス（RPA）に変換する.
6. 続いて，PUT 先 PE の DMA コントローラ（$DMAC_r$）は，リモート物理アドレス（RPA）を用いて受信データをキャッシュに書き込む．キャッシュがミスヒットした場合は，当該データを書き込むべき行をローカル・メモリからキャッシュにコピーして，当該データを書き込む.
7. PUT 元 PE の DMA コントローラ（$DMAC_s$）は，ローカル論理アドレス（LLA），リモート論理アドレス（RLA），および，転送サイズを更新する.
8. 指定された転送サイズに達するまで，2〜7 を繰り返す.
9. 転送が完了したら，その旨を示すフラグを PUT 元および PUT 先プロセッサにおいて設定する.

さらに，オプションにより，PUT 元および／あるいは PUT 先プロセッサに対して割込みをかける．

GET

1. GET 命令を実行するプロセッサ（GET 元プロセッサ）が，ルータ内の DMA コントローラ（$DMAC_r$）に対して，ローカル論理アドレス（LLA），リモート論理アドレス（RLA），および，転送サイズを渡す．
2. ルータにより，GET 先 PE の DMA コントローラ（$DMAC_s$）に対して，ローカル論理アドレス（LLA），リモート論理アドレス（RLA），および，転送サイズを送信する．
3. GET 先 PE の DMA コントローラ（$DMAC_s$）が PUT 命令における PUT 元 PE の DMA コントローラ（$DMAC_s$），また，GET 元 PE の DMA コントローラ（$DMAC_r$）が PUT 命令における PUT 先 PE の DMA コントローラ（$DMAC_r$）となって，PUT 命令の 2〜8 を実行する．
4. 転送が完了したら，その旨を示すフラグを GET 元および GET 先プロセッサにおいて設定する．さらに，オプションにより，GET 元および／あるいは GET 先プロセッサに対して割込みをかける．

6 おわりに

以上，*PPRAM* のアーキテクチャ上の枠組について述べた．

PPRAM プロジェクトでは今後，以下の作業を計画している．

- 本稿で述べた *PPRAM* のアーキテクチャ上の枠組に基づき，具体的なインプリメンテーション（＝アーキテクチャ）を個別に策定する．これには，プロトタイプ・アーキテクチャおよび評価ベースラインとして我々が現在開発を行なっている $PPRAM_{mf}$[7] に加えて，産業界との協力による既存の商用汎用マイクロプロセッサ・アーキテクチャ（SPARC, MIPS, PA-RISC, PowerPC, Alpha, x86）の *PPRAM* 化，ないし，新規 *PPRAM* 型汎用マイクロプロセッサ・アーキテクチャの産官学界との協同開発も含んでいる．
- 上記で策定したアーキテクチャに基づき，個別のマイクロプロセッサの設計開発を行なう．これには，我々の手による $PPRAM_{mf}$ 仕様のマイクロプロセッサ 1 号機の設計開発に加えて，*PPRAM* 型汎用マイクロプロセッサの産官学界との協同開発も含む．
- 産官学に広く協力を求めて，コンパイラ，ランタイム・システム，OS，等の基本ソフトウェアを整備する．
- 産官学に広く協力を求めて，*PPRAM* を構成要素とした計算機システム (PDA, パソコン, WS, 並列マシン, 等) の構築を推進する．
- 以上述べた産官学に横断する協同作業を円滑に進めるため，何らかの協議機関を設置する．

また，我々自身の今後の研究計画としては，上記の $PPRAM_{mf}$ の策定および $PPRAM_{mf}$ 仕様のマイクロプロセッサ 1 号機の設計開発に加えて，以下を予定している．

- *PPRAM* 性能モデルの開発
- $PPRAM_{mf}$ をベースラインとした性能評価用シミュレータの開発
- *PPRAM* 向きコンパイラ最適化技法の開発および蓄積
- ベンチマーク・プログラムの作成，および，$PPRAM_{mf}$ シミュレータによる性能評価
- *PPRAM* を構成要素とした計算機システム構成法の構築とその CAD 化

[7] *PPRAM* として備えるべき最低限の機能 (mf: minimal functionality) のみを備えたアーキテクチャ．

謝辞

日頃から御討論頂く，九州大学 大学院総合理工学研究科 安浦寛人 教授，岩井原瑞穂 助手，安浦研究室の諸氏，米 Univ. of Wisconsin-Madison の Prof. J. Goodman, Prof. G. Sohi, Prof. M. Hill, Prof. D. Wood, および，Computer Architecture Group の諸氏に感謝致します．

参考文献

[1] 村上和彰, 吉井 卓, 岩下茂信, "21 世紀に向けた新しい汎用機能部品 *PPRAM* の提案," 情処研報, ARC-108-8, 1994 年 10 月.

[2] 村上和彰, 岩下茂信, 吉井 卓, "21 世紀に向けた新しい汎用機能部品 *PPRAM*— 並列計算モデルの検討 —," 情処研報, ARC-110-20, 1995 年 1 月.

[3] 吉井 卓, 岩下茂信, 村上和彰, "仮想共有メモリの性能評価," 情処研報, HPC-55-15, 1995 年 3 月.

[4] 林憲一, 土肥実久, 堀江健志, 小柳洋一, 白木長武, 今村信貴, 清水俊幸, 石畑宏明, 進藤達也, "PUT/GET インターフェースのハードウェアサポートによる並列プログラムの効率的実行," 並列処理シンポジウム *JSPP'94* 論文集, pp.233-240, 1994 年 6 月.

[5] Eicken, T., Culler, D.E., Goldstein, S.C., and Schauser, K.E., "Active Messages: A Mechanism for Integrated Communication and Computation," *Proc. of 19th Ann. Int. Symp. on Computer Architecture*, pp.256-266, May 1992.

[6] Hill, M.D., Larus, J.R., and Wood, D.A., "Tempest: A Substrate for Portable Parallel Programs," 並列処理シンポジウム *JSPP'95* 論文集, pp.123-128, May 1995.