Vol. 39 No. 6 情報処理学会論文誌 June 1998

超並列計算機 JUMP-1 における 分散共有メモリ管理プロセッサ MBP-light

安生 健一朗[†2] 井上 浩明[†1] 佐藤 充[†3]
工藤 知宏[†4] 天野 英晴[†1] 平木 敬[†5]

超並列マシン JUMP-1 は，1000 を越えるノード数の大規模システムで効率良くキャッシュコヒーレントな分散共有メモリを構築することを目的としたシステムである．本論文では JUMP-1 用分散共有メモリ管理プロセッサ MBP(Memory Based Processor)-light の設計について述べる．MBP-light は高速処理を要求されるタグ参照，応答パケットの生成および収集等を完全にハードウェア化する一方，コアプロセッサに Buffer-Register アーキテクチャを用いることにより，簡単な構成で高い性能を実現する．ゲート数および実際の JUMP-1 におけるキャッシュプロトコル処理の実行時間を評価し，Sequent 社の NUMA-Q で用いられている SCLIC と OBIC チップに比べて，より少ないゲート数で処理時間において同等な性能を示すことがわかった．また，MBP-light のコアプロセッサとして教育用 32 bit RISC プロセッサとして普及している DLX を用いた場合の構成と比較して，Buffer-Register アーキテクチャによる MBP Core を用いた場合の構成が分散共有メモリ管理の基本処理を行なう上で性能，ゲート数共に有利であることを示す．

MBP-light: DSM management processor on Massivelly Parallel Processor JUMP-1

Ken-ichiro Anjo,[†1] Hiroaki Inoue,[†1] Mitsuru Satoh,[†3]
Tomohiro Kudoh,[†4] Hideharu Amano[†1] and Kei Hiraki [†5]

JUMP-1 has been developed for building an efficient cache coherent distributed shared memory on a massively parallel processor with more than 1000 processing elements. MBP(Memory Based Processor)-light is designed and implemented as a distributed shared memory controller for JUMP-1. In the MBP-light, operations which require high speed execution, such as tag reference, acknowledge packet generation, and collection are completely implemented with a hardware. Other operations are executed by software of the core processor which adopts the buffer-register architecture for efficient manipulation of packets. Estimation results show that MBP-light supports a comparable performance with smaller gates than those of SCLIC chip and OBIC chip used in Sequent NUMA-Q. Furthermore, we show that the buffer-register architecture used in MBP-light is suitable for distributed shared memory management protocols, by comparing with a common 32-bit RISC processor.

1. はじめに

キャッシュコヒーレントな分散共有メモリを持つマルチプロセッサシステム CC-NUMA(Cache Coherent Non-Uniform Memory Access model) は，システムサイズに応じスケーラブルな性能が得られる可能性を持つ一方，現在の小規模バス結合型マルチプロセッサからのプログラム移植が容易であることから，将来の大規模マルチプロセッサシステムの有望な形式として各地で研究が行なわれ，商用機も出現している．この代表例として Stanford 大学の DASH[1)]/FLASH[2)]，

†1 慶應義塾大学理工学部
Faculty Science and Technology, Keio University
†2 NEC
NEC Corporation
†3 (株) 富士通研究所
Fujitsu Laboratories, Ltd.
†4 新情報処理開発機構
Real World Computing Partnership
†5 東京大学理学部情報科学科
Department of Information Science, Faculty of Science, The University of Tokyo

SGI 社の Origin2000[3]，MIT の Alewife[4]，Sequent 社の NUMA-Q[5] などのシステムがある．これらの CC-NUMA では，分散共有メモリの管理を行なうコントローラがシステムの性能とコストを左右する鍵となる．DASH ではこれを完全にハードウェア化したため，プロトコルの柔軟性，制御ハードウェアの複雑化の点で問題があった．FLASH や NUMA-Q など最近の CC-NUMA ではプロトコル制御用のコアプロセッサを内蔵した専用コントローラチップを用い，プロトコル制御方式の柔軟性を確保している．

しかしながら，これらのシステムはプロセッサ数が数十から数百程度の規模では効率良く分散共有メモリが実現できるものの，プロセッサ数が数千を越える規模に拡張する場合，ディレクトリの管理方式，メッセージの転送方式を含む多くの問題点を持つ．

JUMP-1[6][7] は，数千プロセッサを越す超並列マシン上に効率の良い分散共有メモリ，同期，メッセージ転送を実現するためのテストベットである．JUMP-1 は，従来に比べてはるかに多数のプロセッサ上でキャッシュコヒーレントな分散共有メモリを実現するために，従来と異なる様々な手法を用いている．JUMP-1 では分散共有メモリの管理用プロセッサとして，MBP(Memory Based Processor)[8][9] が提案された．本論文では，MBP において高速処理が要求される操作を完全にハードウェア化する一方，よりコンパクトな実装が可能となる Buffer-Register アーキテクチャを提案し，それに基づくコアプロセッサ実現方式を示す．さらに本方式を用いて MBP を実現した，MBP-light チップの構成と実装に関して述べ，典型的な操作の実行時間とゲート数を評価する．

2. 超並列マシン JUMP-1 の分散共有メモリ

JUMP-1 は，文部省重点領域研究「超並列原理に基づく情報処理基本体系」の一環として，1994 年より 8 大学の共同で開発を進めている．JUMP-1 のクラスタは図 **1** に示すように，二次元トーラスの Fat-tree 状の階層構造を持つ結合網 RDT(Recursive Diagonal Torus)[10] により接続されている．RDT ネットワークは RDT ルータチップ [11][12] により実現される．さらに各クラスタは，強力で経済性に優れた並列 I/O システムである高速シリアルリンク STAFF-Link により，ディスクおよび画像データ用 Frame Buffer(FB) と接続される [13]．

図 **2** に示すように，JUMP-1 クラスタは，4 つの RISC プロセッサ (SuperSPARC+) が，高機能の L2 キャッシュと共有バス (Cluster Bus:CBus) を介して

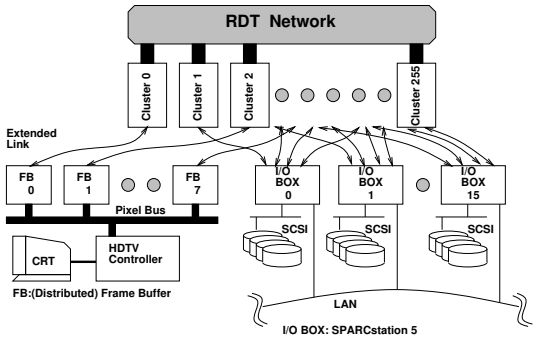


図 1 JUMP-1 の構成
Fig. 1 Structure of JUMP-1

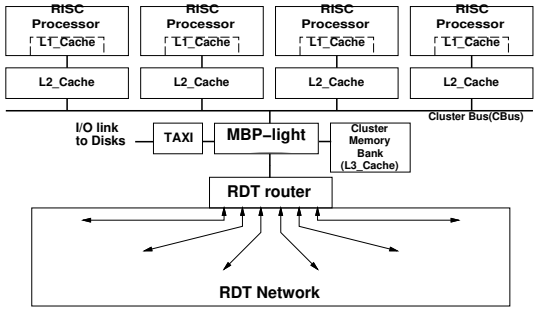


図 2 JUMP-1 クラスタの構成
Fig. 2 Structure of JUMP-1 cluster

本研究による MBP-light に接続されている．MBP-light は共有バス，クラスタメモリ，ルータチップに接続され，キャッシュコヒーレントな分散共有メモリを構成するためのプロトコル制御およびデータ転送を実現する．

DASH/FLASH， Alewife， NUMA-Q 等従来の CC-NUMA は，キャッシュの管理はライン単位で行ない， 基本的に他のノードのメモリのコピーは，それぞれのノードに接続されたキャッシュ中に置かれる．キャッシュ制御プロトコルは単純な無効化型である．さらに， 結合網は単純な格子またはリングで，ディレクトリ管理は，1 対 1 のメッセージ転送で行なわれる．

しかし，このような従来の手法では，システムサイズの増加に伴いディレクトリの容量が大きくなり，高速なメモリ内に格納することが困難となる．また，1 対 1 転送に基づく無効化型プロトコルでは， 大量のデータを多数のノードで共有するようになると， 無効化とデータのコピーのやり直しが頻発し，高い効率を得ることが難しい．

これら諸問題点を解決する目的で，JUMP-1 では以下の方式を用いる．

1) すべてのノードは，単一のアドレス空間を共有する．この空間は，各クラスタがページ (4K バイト) 単位で GPMT(Global Page Map Table) を持ち管理する．GPMT は，ホームメモリに対してはそのページの共有関係を示す縮約階層ビットマップ，ホームでないメモリに対してはホームメモリのクラスタ番号を持つ他，ページの属性，共有されているキャッシュライン数等の情報を持つ．
IVY[14]，Tempest[15] などの VSM(Virtual Shared Memory) 同様，プロセッサの仮想記憶管理用ハードウェア MMU(Memory Management Unit) が管理するページテーブルをアドレス変換に利用する．[☆]それぞれのクラスタ上のクラスタメモリは，その一部を他のクラスタのメモリのキャッシュ領域 (L3 キャッシュと呼ぶ) として利用することができる．プロセッサの発生した論理アドレスは，MMU により物理アドレスに変換され，そのアドレスがクラスタメモリに存在すればアクセスされ，存在しなければ GPMT の参照によりページの割り付けが行なわれる[☆☆]，
2) 一般の VSM とは異なり，メモリ上のキャッシュライン単位にタグを持ち， ラインが有効であるかどうかとプライベートであるかどうかを高速に判別する．ページが割り付けられていても，アクセスしようとしたラインが有効でない場合， ホームクラスタからライン単位で転送する．
3) ノード間で頻繁にデータをやりとりするアプリケーション用に更新型プロトコルを導入し， ページ単位で使い分ける．
4) ディレクトリの管理に， 結合網 RDT の階層性を利用した縮約階層ビットマップディレクトリ方式 (Reduced Hierarchical Bitmap Directory scheme: RHBD)[7)16)] を用いる．この方法ではキャッシュの無効化や更新のためのパケットが，縮約されたビットマップに従って階層的にマルチキャストされ，それらの処理の終了を確認するための応答パケットを結合網内でコンバインしながら収集する．
5) 各プロセッサは，L2 キャッシュとして専用かつ高機能なスヌープキャッシュを持つ， L2 キャッシュは，様々なキャッシュプロトコルを実現することができる上，書き込み用のバッファを持つことにより，弱いコンシステンシィモデルの実現も管理する．さらに，効率の良いメッセージ転送を行なう機能を持つ[17]．

JUMP-1 の分散共有メモリ管理手法に関する詳細は[9]を参照されたい．

3. MBP-light の構成

3.1 設 計 方 針

プロトコル制御を効率良く行なうため，FLASH の MAGIC や NUMA-Q の SCLIC，OBIC チップでは，テーブルジャンプ機能やタスク切り換え機能を持つコアプロセッサを内蔵している．FLASH の MAGIC では，プロセッサ，ネットワーク，I/O からのアクセスは全て同一のシーケンスで処理される．パケットはデータ部とヘッダ部に分解され，ヘッダ部はコアプロセッサ中のレジスタに転送される．このコアプロセッサでヘッダ部を操作し，同時にデータ部は専用のバッファに蓄えられる．コアプロセッサでの処理の後，ヘッダ部とデータ部を結合しパケットを転送する．一方，NUMA-Q の分散共有メモリは，共有バスの管理，ノード内アクセスのタグ判定を行なう OBIC とクラスタ間で行なう必要のあるプロトコル処理を行なう SCLIC によって管理される．OBIC はハードウェアのみで構成されており，SCLIC はコアプロセッサを内部に持つ構成となっている．SCLIC では，MAGIC 同様パケットをヘッダ部とデータ部に分解し，ヘッダ部のみコアプロセッサで処理する．

一方，以下に JUMP-1 のプロトコルコントローラに対する要件を挙げる．

1) クラスタ内メモリアクセスや，縮約階層ビットマップディレクトリの性能を十分発揮するために，応答パケットの生成や収集をできるだけ高速に行なう必要がある．また，この処理の頻度が高い場合を考慮し，この処理が他のパケット処理をなるべく妨げないような構成にする必要がある．

2) JUMP-1 では多様なキャッシュプロトコルをサポートしており，そのためのコマンド等のパケットへの付加情報が多く，制御も複雑である．そのため，それらの情報の管理やパケット制御などは柔軟に管理する必要があり，ハードウェアで実現するよりもコアプロセッサ上のソフトウェアで実現する方が望ましい．

3) 分散共有メモリ上で同期メッセージを転送する必要等から，コアプロセッサ上ではパケットのデータ部のタグも操作する必要が生じる．このため，ヘッダ部とデータ部を分解してヘッダ部のみをコアプロセッサで扱う方法を利用することは出来ず，パケット全体

[☆] MBP ではホームメモリとクラスタの対応関係を柔軟にし，OS レベルの保護を行なうため，もう一段アドレス変換を行なうハードウェアを持つが[9]，本論文で述べる MBP-light ではこの機能をハードウェアでは持たず，実現する場合は MBP Core のソフトウェアを用いる．ここでは説明を簡単にするため，アドレス変換は一回として解説している．

[☆☆] ダミーページの利用により， プロセッサに接続された L2 キャッシュ上に直接ラインをキャッシュすることも可能である．

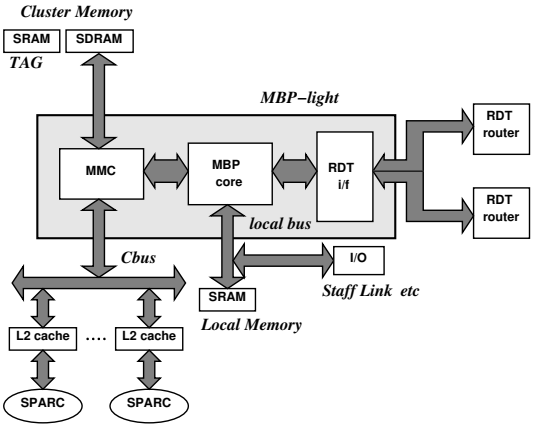


図 3 MBP-light の全体構成
Fig. 3 Structure of MBP-light

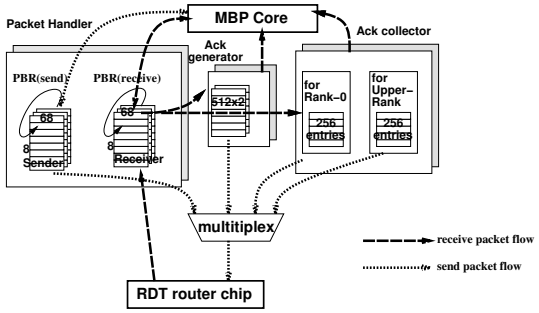


図 4 RDT インタフェースの構成
Fig. 4 structure of RDT interface

をコアプロセッサで扱う手法を用いる必要がある．しかしその際，パケットをバッファ・レジスタ間で転送するオーバヘッドは最小限にとどめる必要がある．

この要件から，MBP-light では以下の設計方針を採用した．

1) 要件 1 から，応答パケットの生成，収集やクラスタ内メモリアクセスはコアプロセッサで処理するよりも専用ハードウェアで処理する方がより高速であり，負荷も分散される．そのため，それぞれ専用ハードウェアを設ける．

2) 要件 2 から，コアプロセッサを実装し，パケットに対してこのコアプロセッサ上でプロトコル制御に関する処理を行なう．

3) 要件 3 から，コアプロセッサ MBP Core は，バッファを特殊なレジスタとして扱う Buffer-Register アーキテクチャを用い，パケットバッファを通常のレジスタとして扱える様にし，パケットの生成を高速に行なうことを可能とする．

図 **3** に上記の方針に基づく MBP-light の構成を示す．MBP-light は，RDT ルータの制御を行なう RDT インタフェース，クラスタメモリと共有バスの制御を行なう MMC(Main Memory Contoroller)，プロトコル制御を行なう MBP Core の三つの部分から構成される．以下，これらの構成要素について述べる．

3.2 RDT インタフェース

RDT インタフェースは，図 **4** に示すように，パケットの送受信を行う Packet Handler，応答パケットの収集を行う Ack Collector，応答パケットの自動生成を行う Ack Generator から構成される．この 3 つのモジュールはすべて独立のコントローラにより制御され，高速動作を必要とするパケット制御を MBP Core のソフトウェアを介することなしに実行する．

Packet Handler:

RDT ルータチップはピン数の制限から転送ビット幅は 17bit としている． JUMP-1 では， 転送容量を確保するため 1 つの MBP-light チップに対してルータチップを 2 個接続し，転送ビット幅を 34bit としている．

RDT の代表的なマルチキャスト用パケットは可変長で，最初の 4flit のヘッダはマルチキャスト用ビットマップ，パケット長，転送情報に加え，応答パケット収集時に用いるキーが格納される．5flit 目にはアクセスアドレスが格納され，6flit 目にはパケットの種類とコマンド情報が格納される．さらにデータを伴うパケットであれば，7–14flit 目にデータフリットとして使用可能で，共有メモリ上で同期をとる場合に使用されるタグ 2 ビットも一緒に転送可能である．また，さらなる付加情報のための 1 フリットの予備フリットを持つ．Packet Handler の送信部では，MBP Core で生成されたパケットに 1 フリットのエラー検出用のパリティフリットを付加し，最小 7flit ~ 最大 16flit のパケットとして RDT ルータに送信する．これに対し応答パケットは固定長で，3flit 中に全ての情報が格納される．

Packet Handler の受信部は，RDT ルータチップから到着したパケットを 68bit × 8flit のバッファに格納する．これらのバッファは 3 組あり，サイクリックバッファを構成している．RDT ルータから到着したパケットは，順にバッファ内に格納される．後述するように，これらのバッファは MBP Core 内でレジスタとして直接アクセスされる．RDT ルータチップは 2 個一組で MBP-light チップに接続されるため，転送情報を含む最初の 4flit は同一のフリットが重複して到着するが，Handler 受信部は受信時に自動的にこれ

らの重複をとり除く．これと同時にパケットの型を識別し，必要に応じて Ack Collector と Ack Generator を起動する．またパリティは受信中にチェックし，エラーがあれば MBP Core に報告する．

Packet Handler の送信部は，MBP Core，Ack Collector，Ack Generator により生成されたパケットを RDT ルータチップへ送信する．MBP Core からのパケットの転送には，受信バッファ同様，68bit × 8flit 3 組から成るサイクリックバッファを用いる．最初の 4flit に関しては 2 個の RDT ルータチップに送出する必要があるが，これは自動的にコピーが行われる．受信バッファ同様，このバッファは MBP Core から直接扱うことができる．

Ack Collector:

無効化，更新等のマルチキャストパケットを発生したノードは，その終了の確認のための応答パケットを収集しなければならない場合が生じる．縮約階層ビットマップ方式では応答パケットの収集を効率的に行わないと MBP-light チップと RDT ルータチップ間のリンクが混雑することがシミュレーションの結果わかっている [18]．RDT ルータチップは最下層の階層の応答パケットを 1 エントリ分自動的に収集する機能を持つが，ルータチップ内のエントリが溢れた分と上位階層の収集は MBP-light の役目となる．

Ack Collector は，最下層用と上位階層用のそれぞれに対して，256 エントリのテーブルを 2 組ずつ持ち，それぞれの階層における応答パケットの収集を，エントリ登録を含めて完全にハードウェア制御で行なう．収集終了時には，さらに上位の階層で収集するための応答パケットを生成し，送出する．そのノードがマルチキャスト発生元であった場合は MBP Core にパケットが渡される．単一ノードからのマルチキャストが重なって行われた場合，エントリが不足して登録が不可能になる場合がある．この時のみ MBP Core のソフトウェアによる処理が必要になる．

Ack Generator:

マルチキャストパケットを受け取ったノードは，速やかに応答パケットを発生する必要がある．Ack Generator は 512 エントリのテーブルを 2 組備え，マルチキャストパケット中の送信ノード番号から自動的に応答パケットを組み立てて送出する．

JUMP-1 の分散共有メモリは管理の単位がページであるので，マルチキャストパケットが到着しても，受け取ったノードがそのパケットに対応するキャッシュラインを持っているとは限らない．Ack Generator 中のテーブルには，キャッシュラインのコピーがそのノードに存在するかどうかを示すタグが格納されており，Ack Generator はこのテーブルを引き，到着したパケットは処理する必要がないと判断した場合は，即座に Packet Handler のバッファ中から削除する．この判断はパケットのヘッダ部のみで高速に行われるため，パケットの後続フリットの受信中にキャンセルが行われ，MBP Core はまったく関与する必要はなくなる．この機構により，不要なパケットにより MBP Core を起動することが避けられる．

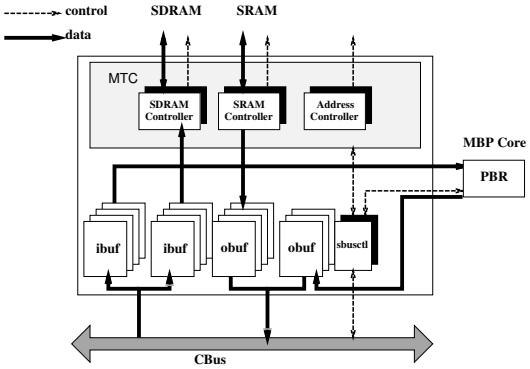


図 5 MMC の内部構成
Fig. 5 structure of MMC

3.3 MMC(Main Memory Controller)

MMC は直接クラスタメモリを扱うためのユニットである．MBP-light では，クラスタメモリに対するクラスタバス，RDT ルータの両者からのアクセスを受け付けなくてはならないため，メモリの集中管理が必要になる．また，クラスタ内ローカルメモリアクセスはできるだけ高速に行なわないと全体の性能低下の原因となるため，ノード内のメモリアクセスをハードウェアにより高速に行なう必要がある．そのために，メモリアクセスをハードウェア的に管理する MMC ユニットが必要となる．以下では MMC の構成と機能について説明する．

内部構成:

MMC は図 5 に示すように，クラスタバスコントローラ (sbusctl)，入出力バッファ(ibuf,obuf)，メモリタイミングコントローラ (mtc) から構成される．

クラスタバスコントローラ (sbusctl)

クラスタバスコントローラは，JUMP-1 のクラスタバスとパケットをやりとりするユニットである．クラスタバスコントローラでは，バスからのパケットを受け取ると，それを入力バッファに格納する．また出力バッファにパケットが溜ると，バスへリクエストを発行し，パケットを出力する．

入出力バッファ(**ibuf, obuf**)

入力バッファではクラスタバスから受け取ったメモリアクセスパケットを，出力バッファではクラスタバスへ出力すべきパケットを保持する．JUMP-1 のクラスタバスでは，デッドロックを回避するため，パケットはリクエストとリプライに分離されている．MMC では内部にリクエスト用に 4 つ，リプライ用に 4 つの計 8 つの入力バッファをもち，後述のメモリタイミングコントローラでメモリアクセスを行なっている最中や，MBP Core によってソフトウェア処理が行なわれている間にも複数のパケットを受け取ることができる．

また，出力バッファは MMC がハードウェアで生成するリプライパケットを 4 つまで保持することができ，その他に MBP Core がクラスタバスへ出力するパケットをリクエストパケット，リプライパケットそれぞれ 1 つまで保持することができる．

メモリタイミングコントローラ (**mtc**)

メモリタイミングコントローラでは，JUMP-1 のクラスタメモリに対してメモリアクセスリクエストを発生する．メモリタイミングコントローラでは，パケットのアドレス部を検査し，ハードウェア的にアクセスすべきパケットであることを確認した後，タグ用 SRAM およびデータ用 DRAM をアクセスする．

また，メモリタイミングコントローラは，MBP Core とのインタフェース制御も行なっている．MBP Core が ibuf のパケットを読みだす際にはメモリタイミングコントローラが ibuf 中の該当パケットを 3.4 章で述べる PBR(Packet Buffer Register) に直接書き込み，MBP Core でのパケットの読み出しが可能となる．また，MBP Core が obuf へ書き込む際には，一度 MBP Core は PBR 中にパケットを生成しておき，メモリタイミングコントローラがそのパケットを obuf へコピーする．

さらに，メモリタイミングコントローラでは，データ書き込み時には ECC(Error Correcting Code) を付加し，データ読みだし時にはエラー検査を行なう．

タグアクセス:

MBP-light の重要な役割のひとつに，メモリアクセス時のタグアクセスがある．これは CC-NUMA を構成する上で必要不可欠な機能であり，すべてのメモリアクセスの度に実行されるので高速に行なう必要がある．

MMC ではメモリアクセスが発生すると，タグ用 SRAM とデータ用 DRAM を同時にアクセスし始める．DRAM がアクセス可能になる前に SRAM のデータを検査する．もし MBP Core によるソフトウェア処理が必要な場合は，DRAM アクセスを中止して処理を MBP Core に依頼する．SRAM アクセスは DRAM のセットアップに完全に隠されるので，タグアクセスによる余分なオーバーヘッドは存在しない．

3.4 MBP Core

Buffer-Register Architecture:

MBP-light では，緊急性の高い処理はすべて RDT インタフェースと MMC がハードウェアにより行なうため，MBP Core が行なう処理は，実際のキャッシュラインの転送やプロトコル制御等を含む処理である．これは，テーブル参照とパケットの生成，分解が主である．FLASH における MAGIC，NUMA-Q の SCLIC チップでは，これらの処理を高速化するため，パケットバッファのヘッダ部を分離し，プロセッサのレジスタ上に転送して直接扱えるようにしている．ところが，JUMP-1 ではパケットのヘッダ部が大きく，種類によってサイズが可変である．また，データ部に含まれるタグもプロトコル制御に関連する．したがって，データも含めたパケットバッファ全体をレジスタとして扱えることが望ましい．ところが，パケットバッファは転送効率を改善するため，68bit 幅で容量自体も大きく，これらをプロセッサの GPR(General Purpose Register) として扱うためには，膨大なハードウェア量が必要で，かつ効率も悪い．

そこで，MBP Core は汎用の 16bit 幅の GPR(16 本) とパケットバッファとしての役割をする 68bit の PBR(Packet Buffer Register)112 本を分けて扱う命令セットを持つ．PBR は，GPR をポインタとしてアクセスされ，PBR-GPR，PBR-PBR 間の演算やデータ転送が可能である一方，その内容は MMC や RDT インタフェースを介して直接パケットとして転送される．演算がバッファとレジスタ間で行われることからこの構成を Buffer-Register アーキテクチャと呼ぶ．Buffer-Register アーキテクチャでは，PBR は単なる内部メモリではなく，パイプライン中では GPR 同様にストールを伴うことなく扱うことができる．

MBP Core の構成:

図 **6** に MBP Core の構成を示す．MBP Core は命令長 21bit，データ長 16bit で 4 段パイプラインを持つ．MBP Core は命令とローカルデータ格納用の 21bit 幅 × 64K のローカルメモリを外部に持つ．MMC が制御するクラスタメモリやタグメモリは，PBR との間で主としてキャッシュライン単位のブロック転送でアクセスされる．さらに，MBP Core は，256 × 16bit

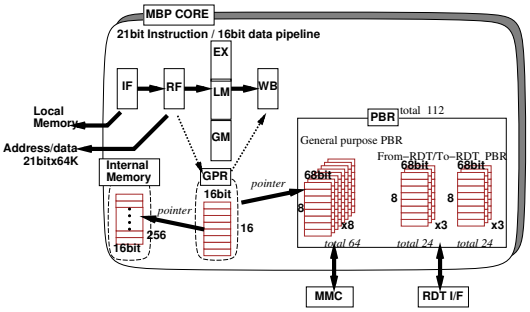


図 6 MBP Core の構成
Fig. 6 Structure of MBP Core

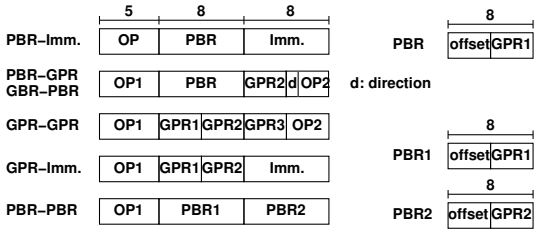


図 7 代表的な命令フォーマット
Fig. 7 Typical instruction format

の内部メモリを持つ．この内部メモリは PBR 同様，GPR をポインタとしてアクセスされ，応答パケット用のビットマップの格納やジャンプテーブルとして用いられる．

PBR(Packet Buffer Register) の構成:

PBR は RDT 送信・受信用がそれぞれ 24 本，汎用が 64 本あり，合計 112 本である．PBR をアクセスする際には，GPR をポインタとし，4 bit のオフセットをバイト単位で指定して PBR にアクセスする．PBR は 1 本が 68 bit で構成され，オフセット 0 ~ 7 をアクセスすると通常の 64 bit のデータ領域をアクセスでき，オフセット 8 をアクセスするとデータに対する 4 bit のタグ領域をアクセスできる．この場合は下位 4 bit のデータ以外は意味を持たない．また，オフセット 9 ~ 15 は使用しない．

MMC と RDT インタフェースの構造上の相違から，それぞれインタフェースの方法は異なっている．RDT 送受信用のそれぞれの PBR 24 本は，RDT の 1 パケット分 (PBR 8 本分) の大きさを単位としてそれぞれ 3 パケット分に分割して，FIFO を構成する．MBP Core はこれらを組単位に指定して，RDT インタフェースに制御を移して直接パケット送受信を行なう．この間指定した PBR 領域には MBP Core からアクセスすることはできず，データの送受信が終わった段階で再び制御を切り替える．この切り替えは 1 クロックで終了する．これに対して MMC を介して行なうクラスタメモリのアクセスおよび共有バス転送用の専用バッファとの間のパケットの授受は，1 命令でブロック転送の形で行われる．この命令は，3 種類の PBR の任意の場所に対して可能であるが，場合によっては 10 クロック以上 (最大 15 クロック) 要する．この場合，後続命令をストールさせることは性能を大きく劣化させるため，後続命令のうち転送中のものと別種の PBR をアクセスする命令は，転送終了を待たずに実行することができる．すなわち Out-of-order completion を許す．したがって，同種の PBR をアクセスしない命令を埋め込むことができれば，転送時間は隠蔽される．同種の PBR をアクセスする必要がある場合は，メモリ転送同期命令を実行し，転送が終了していることを保証する必要がある．

プロトコルにより，利用するパケットの形式はほぼ決まっているため，それぞれのバッファのヘッダは PBR 上にあらかじめ作っておき，ノード番号やアドレスに依存する部分のフィールドに対してのみ GPR-PBR 命令でデータを書き込む方法が有効である．汎用 PBR は 64 本用意されているため，代表的なパケットのヘッダはあらかじめ作成して格納しておくことが可能である．

命令セット:

図 7 に MBP Core の命令形式と代表的な命令を示す．GPR は 3 ポートメモリであり，GPR 同士の演算が 3 アドレッシング方式で行われるのに対し，PBR を含む演算は 2 ポートで 2 アドレッシング方式である．PBR は，バイト単位のアドレッシングがなされており，GPR をポインタ (4bit のディスプレースメント付き) としてアクセスされる．たとえば，GPR-PBR 命令は，GPR2 の内容と，GPR1 によって指定された PBR 中の 8bit の間で演算が行われる．PBR に対する演算は特定の 8bit，16bit に対して行われるが，これはパケットのヘッダのそれぞれのフィールドの多くが 8bit 以内であり，フィールド単位で独立の意味を持っているためである．GPR-GPR，GPR-PBR 間の演算は豊富であるが，PBR-PBR 間の演算は 68 bit データ転送等ごく限られたものである．

パイプライン構成:

図 6 中に示すように，MBP Core は 4 段のパイプライン構成を取る．GPR は 16bit の 3 ポートレジスタであり，演算命令で扱うデータは基本的に 16bit 以内である．また，ディスプレースメントの範囲は 4bit に過ぎない．MBP-light は，CPU(SuperSPARC+) の

クロックに同期して動作するため，50MHz のクロックで動作する必要がある．今回実装に用いた ASIC のゲート遅延を考えると，1 クロック内に「GPR の読み出し+ディスプレースメント演算」あるいは「PBR の読み出し+16bit 演算」を行なうことが可能である．このため，今回の実装では，1 ステージ内でこれらの処理を行なうことにより，ステージ数を減らし，ステージ間のフォワーディングを単純化する方法を採用した．具体的には，4 つのステージで，以下の操作を行なう．

(1) **IF(命令フェッチ):** 外部ローカルメモリから命令を読み出す． この処理は 1 クロックで終了する．

(2) **RF(レジスタフェッチ):**

メモリアクセス: GPR を読み出し，実効アドレス計算を行なう．

分岐: 判断，アドレス計算，分岐を行なう．

その他の命令: 命令コード中で指定された GPR 双方の値を読み出し，ディスプレースメント演算を行なう．

(3-1) **EX(実行):** PBR を読み出し 16bit 演算を行なう．

(3-2) **LM(ローカルメモリアクセス):** ローカルメモリ，I/O のアクセスを行ない，遅い I/O に対するアクセスではストールする．

(3-3) **GM(MMC との交信):** MMC を介してクラスタメモリアクセスおよびバスパケットの転送を行なう．このステージが MMC と転送中，後続の関連する PBR をアクセスしない命令は LM，EX を通って先に進むことができる．このことにより Out-of-order completion 機能を実現する．

(4) **WB(書き戻し):** 結果の格納を PBR，GPR に対して行なう．

命令とローカルデータは共にローカルメモリに格納されるため，ローカルメモリアクセス命令は structural hazard によりパイプラインを 1 クロックストールさせる．また分岐命令は 1 クロックの遅延スロットを必要とする．

イベントの処理:

MBP Core は MMC，RDT i/f，外部 I/O デバイスからの要求によるテーブルジャンプを伴う割り込みをサポートしている．割り込みは IF ステージで受け付け，パイプライン中の命令がすべて終了するのを待って割り込みルーチンにジャンプする簡単かつ正確なものである．割り込み要因は MMC，RDT i/f 内部でキューイングされ，全ての割り込み要因が満足されるまで連続して割り込みがかかる．多重割り込み可能であり，テーブルジャンプの優先順位を制御することも可能である．このテーブルジャンプは，割り込みに伴って自動的に行なう以外にも，テーブルジャンプ命令によって起動し，ポーリングに利用することもできる．

表 1 MBP-light のゲート数
Table 1 Number of gates in MBP-light

ブロック名	ゲート数	メモリ量 (bits)
MBP Core	35,734	6,144
RDT i/f	19,832	38,704
MMC	43,203	0
総計換算	98,769	44,848

表 2 SCLIC, OBIC のゲート数とメモリ量
Table 2 Number of gates in OBIC and SCLIC

ブロック名	ゲート数	メモリ量 (bits)
SCLIC	140,000	152,000
OBIC	170,000	8,700
総計換算	310,000	160,700

4. 性 能 評 価

4.1 ゲート数と動作速度

MBP-light は，東芝 CMOS エンベッデッドアレイ TC203E340 T3S35(197,000 ゲート)(3.3V/5V 混在)を用いており，352 ピンの TBGA パッケージで実装する．設計には VHDL を用い，論理合成には Mentor Graphics 社の Autologic を用いた．仮遅延シミュレーションは Mentor Graphics 社の QuicksimII を利用した．Autologic による合成結果，MBP-light 自体は 60MHz で動作が可能であり，JUMP-1 は Super-SPARC+のクロックに合わせて 50MHz の単一システムクロックで動作する予定である．現在，仮遅延シミュレーションが終了している．MBP-light のハードウェア量を評価した結果を表 **1** に示す．表中ではメモリ量はゲート数換算は困難なためビット数で示している．MBP-light では総ゲート数 197,000 ゲートのうちランダムロジックは約半分を占めており，残りは内部メモリに使用される．

また，表 **2** に NUMA-Q における SCLIC と OBIC に使用されているゲート数を示す [5]．MBP-light と OBIC, SCLIC を比べると，MBP-light では内蔵 RAM の量，ランダムロジック共に大幅に小さな実装が行なわれていることがわかる．MBP Core に Buffer-Register アーキテクチャを導入し，ハードウェア量を減らした効果がここに表れている．

4.2 メモリアクセス処理時間

4 章で示したキャッシュプロトコルの実行時間のうち MBP-light における遅延を Mentor Graphics 社の VHDL シミュレータで測定した．MBP-light は 60MHz で動作が可能であるが，JUMP-1 は 50MHz で

表 3 Read Miss 時の処理時間
Table 3 Read miss Latency

操作	C_{50}	T_{50}	T_{stall}
MMC 受信 (R)	10	200	0
割り込み, デコード (R)	26	520	120
RDT パケット生成送信 (R)	56	1,120	100
Requester total	92	1,840	220
Network Latency	4n	80n	–
RDT i/f 受信 (H)	8	160	0
割り込み, デコード (H)	17	340	0
タグ 調査 (H)	23	460	0
データ読みだし (H)	20	400	120
RDT パケット生成 (H)	43	860	60
Home total	110	2,220	180
Network Latency	4n	80n	–
RDT i/f 受信 (R)	16	320	0
割り込み, デコード (R)	16	320	0
データ更新 (R)	28	560	220
タグ更新 (R)	15	300	160
CBus パケット送信 (R)	20	400	280
Requester total	95	1,900	660
Total	298+8n	5,960+160n	1,060

C_{50} : 50MHz 動作時クロック数
T_{50} : 50MHz 動作時処理時間 (ns)
T_{stall} : MBP Core のストール時間 (ns)
(H): Home における処理
(R): Requester における処理
n : ネットワークホップ数

の動作が予定されているため，今回の評価では 50MHz 動作時の処理時間を測定した．あるクラスタにおいてアクセスしたラインのコピーがクラスタ内に存在せず，そのラインのホームノードに有効なラインが存在した場合の Read 要求処理時間を表 **3** に示す．

この処理は，1. 要求を発生したノード (Requester:R) で CBus からパケットを受信し Home ノード (H) に Read 要求パケットを送信する，2. Home ノードにおいてパケットを受信し，Read reply パケットを返送する，3. Requester ノードでパケットを受信し，CBus に Read reply パケットを送信する，の 3 ステップに分けることができる．これらのステップとその詳細な内訳についての評価結果を表 3 に示す．図に示す T_{stall} はクラスタメモリ，タグメモリ，パケット送受信，ローカルメモリアクセスなどにより起こるストール時間である．上記の処理のうち 3. の部分では，Read reply を受け取った Requester が行なう処理の流れには互いにデータの依存関係が存在し，表 3 に示すようにクラスタメモリやタグメモリのアクセスのためのストール時間が全体の処理時間の 1/3 を占めている．また，ホームノードのデータを読むのにトータルで 5,960 + 160×(ネットワークのホップ数)$nsec$ の時間が必要であることがわかる．

NUMA-Q で用いられている SCLIC チップでも同様の操作の処理時間が評価されており，SCLIC チップは 4 ノード先の Read request 操作に 4-10μsec[5] 要している．このことから， MBP-light の実行速度は 60MHz で動作した場合 (5.0+ 0.13× ホップ数)μsec は，SCLIC とは同等の性能を示していると言える．従って MBP-light は OBIC,SCLIC と比較して，ゲート数で有利であり，同等の性能を示している．

4.3 Buffer Register Architecture

次に，MBP Core の構成を一般的な構成のマイクロプロセッサと比較して評価する．比較対象として，教育用の 32bit RISC プロセッサとして広く普及している DLX[19] を選んだ． DLX を用いて MBP-light を構成した場合の評価モデルを図 **8** に示す．これは，MBP Core の PBR を RDT 内部メモリと MMC 内部メモリに分割したモデルであり，RDT や MMC への送信はこの内部メモリ上からハードウェアで行なわれるものと仮定した．また，純粋に Core 部の比較のみを行なうため，MMC および RDTi/f は同一のハードウェアを用いると仮定した．また，DLX を用いた場合は RDT 用，MMC 用それぞれの内部メモリ間で DMA を行なうことができると仮定した．以降，今回実装した MBP-light を MBP Core 版，DLX を用いて MBP-light を構成した場合を DLX 版とする．

評価の際，MBP Core 版における VHDL シミュレーションによる評価結果からクラスタメモリへのアクセス時間，バスへのパケット送信時間，受信時間をそれぞれ 15，13，11 クロックかかると仮定し，DLX 版においてはハードウェア操作を行なう MMC-DLX インタフェースが存在するものと仮定した．

この DLX 版と MBP Core 版において MBP-light における基本処理プログラムを動作させ，そのクロック数を比較した．この結果を表 **4** に示す．

MBP Core 版と DLX 版を比較すると，MBP Core 版の利点は PBR を持っており，パイプラインをストールすることなくレジスタとしてパケットデータをアクセスできる点，DLX の利点は 32 bit データを扱うことが可能である点と 16 bit の直値を指定できる点である．そのため，DLX は主にパケットのビットフィールドを扱う際に有利であるが，一方で DLX はレジスタ・メモリ間演算はできないため，内部メモリ上のパケットデータを操作する際は必ず内部レジスタとメモリ間でロード/ストアする必要が生じる．また，MBP Core 版では PBR 間の 68 bit 転送命令を活用することができるために，キャッシュラインデータを 4 命令 (クロック) で転送することが可能であり，データ付き

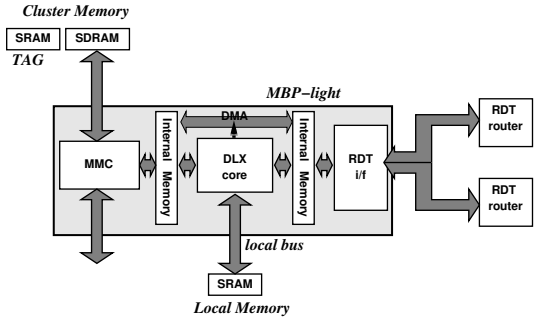


図 8 DLX を用いた MBP-light のアーキテクチャモデル
Fig. 8 Architecture model of MBP-light using DLX

表 4 MBP Core 版と DLX 版での基本処理の性能比較
Table 4 Comparison between MBP Core and DLX in typical process

処理	Tm	Td	Td/Tm
CBus パケット生成 (データなし)	28	29	1.04
CBus パケット生成 (データ付き)	28	36	1.29
CBus デコード	27	31	1.15
RDT パケット生成 (データなし)	49	54	1.10
RDT パケット生成 (データ付き)	60	69	1.15
RDT パケットデコード	15	20	1.33
グローバルアドレス変換	23	23	1.00
ローカルアドレス変換	40	50	1.25
タグの比較	28	35	1.25
タグ更新	42	50	1.19
データ更新 (Global Memory)	17	22	1.29

Tm : MBP Core 版クロック数
Td : DLX 版クロック数

パケットの生成時には DLX 版に比べ 3 クロック有利になる．さらに，MBP Core 版では PBR に対してはバイト単位で任意にオフセットの指定が可能であるが，DLX ではメモリアクセスする際に整列化されていないデータに対するアクセスを許していない．このため，ワード境界をまたがったデータに対してアクセスする際には MBP Core 版よりも 3 ~ 6 クロックのオーバヘッドが生じる．

DLX 版，MBP Core 版ともに有利不利な点があるが，実際のプロトコル処理の一部である RDT パケットや CBus パケット生成の処理時間においては，表 4 に示すように RDT パケット，CBus パケットのどちらを扱う際のいずれの場合も MBP Core 版が最大 33%有利であることがわかる．また，グローバルアドレス変換やローカルアドレス変換はローカルメモリ上のテーブルを引き，アドレスを変換する操作を行なっている．ローカルアドレス変換の場合には RDT パケットのアドレス部分からページ番号を切り出す際に整列化されていないデータに対するアクセスが生じるために MBP Core 版が有利になっている．また，タグやデータの更新の際には，MBP Core 版においてクラスタメモリやタグメモリへのアクセスが Out-of-completion であることで，MBP Core 版は 5 ~ 8 クロック程有利になっている．まとめると，パケットをバイト単位でパイプライン中から自由にアクセスできる PBR を利用していることによる利点や PBR 間の 68 bit 転送のサポート，タグメモリやクラスタメモリアクセスが Out-of-completion であることが，DLX の 16 bit の直値指定可能であるなどの利点を上回っていることがわかった．

表 5 MBP Core 版と DLX 版のクリティカルパス遅延とゲート数比較
Table 5 Comparison between MBP Core and DLX in number of gates and delay in a critical path

プロセッサ	クリティカルパス遅延	ゲート数
MBP Core 版	12.7nsec	22,973
DLX 版	13.3nsec	27,405

表 6 MBP Core と DLX のゲート数詳細比較
Table 6 Comparison between MBP Core and DLX in number of gates in detail

ユニット	MBP Core(ゲート)	DLX(ゲート)
ALU	960	1,800
命令フェッチ	845	1,709
デコード	3,103	3,714
実行	6,232	2,535
メモリアクセス	459	1,701
ライトバック	6,461	9
汎用レジスタ	3,736	15,692
その他	1,177	888
トータル	22,973	27,405

さらに，MBP Core 版と同一のライブラリを用いて DLX 版を論理合成した場合のクリティカルパスの遅延時間とゲート数を表 **5** に示し，詳細のゲート数を表 **6** に示した．ゲート数は純粋に CPU 部について比較した．すなわち MBP Core 版については，PBR 自体と MMC および RDT i/f とのインタフェース回路は除いている．一方 DLX 版については，PBR に対応する内部メモリ，DMA コントローラ回路は除いて評価した．

クリティカルパスは 32bit 演算を装備する分，DLX の方が若干長くなっている．一方，ゲート数については，MBP Core は PBR に対するマルチプレクサや特殊命令用のハードウェアにより，一般的な 16bit RISC に比べてハードウェア量は大きいが，それでも bit 長の長い DLX 版に比べ 17%程度有利である．また，ここで設計した DLX 版は比較用であるため，外部割り

込み，例外処理等を実装しておらず，実際に DLX をコアプロセッサとして用いる場合はさらに差は広がると考えられる．

以上，MBP Core は Buffer-Register アーキテクチャをとることにより，汎用の 32bit RISC をコアプロセッサに持つ場合に比べ，ハードウェア要求量は削減することができ、処理に要するクロック数，クロック周期共に小さくすることができ，特にパケット生成時に高い性能を得ることができることが確認された．

5. おわりに

超並列マシン JUMP-1 用の分散共有メモリ管理プロセッサ MBP-light を設計した． MBP-light は高速性を要求される応答パケットの生成や収集は完全にハードウェア化する一方，コアプロセッサに Buffer-Register アーキテクチャを用いることにより，簡単で高い効率を実現する．ゲート数と実行時間を評価した結果， NUMA-Q で用いられている OBIC, SCLIC チップに比べて，少ないゲート数で， 同等の性能を実現していることが明らかになった．また Buffer-Regsiter アーキテクチャを用いた MBP-light のコアプロセッサは，32bit RISC の DLX に比べ同程度のハードウェア量で，より高い周波数のクロックを用いることができ，特にパケット生成に関しては 33%程度高速であることがわかった．今後は，Stanford FLASH の MAGIC チップに関して詳細が公表され次第，比較検討を行なう必要がある．また，現在 JUMP-1 の全体性能を MBP-light の他，L2 キャッシュの動作，ページを制御するソフトウェア，結合網等をシステム全体を適切にモデル化してシミュレーション中であり，JUMP-1 全体の性能の中で，MBP-light の性能が及ぼす影響について調査する必要がある．さらに実装終了後は，実機による評価を行なう．

謝辞 MBP-light の設計・開発にあたり多大なご支援をいただいた京都大学の富田眞治 教授に感謝いたします．ならびに豊橋技術科学大学の中島浩教授，九州工業大学の久我守弘助教授，神戸大学の中條拓伯助手，京都大学 五島正裕 助手をはじめとする JUMP-1 開発グループのメンバに感謝いたします．また，MBP の構想の確立に御尽力いただいた東京大学の松本尚 助手に感謝いたします．MBP-light の設計には，Mentor Graphics 社の University Program を利用しました．ここに深く感謝します．

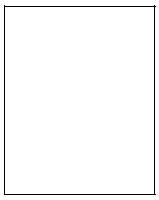
なお，本研究の一部は文部省科学研究費・重点領域研究 (1)(課題番号 04235130「超並列ハードウェア・アーキテクチャの研究」) および試験研究 (A)(1)(課題番号 06508001「超並列計算機プロトタイプの開発と試作」によります．

参 考 文 献

1) Lenoski, D., Laudon, J., Gharachorloo, K., Gupta, A. and Hennessy, J.: The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor, *17th ISCA* (1990).
2) Kuskin, J. and et, a.: The Stanford FLASH Multiprocessor, *Proc. The 21st ISCA*, pp. 302 – 313 (1994).
3) Laudon, J. and Lenoski, D.: The SGI Origin 2000: A CC-NUMA Highly Scaleble Server, *Proc. of 24th ISCA* (1997).
4) Chaiken, D. and Agarwal, A.: Software-Extended Coherent Shared Memory: Performance and Cost, *Proc. The 21st ISCA*, pp. 314 – 324 (1994).
5) Lovett, T. and Clapp, R.: STiNG: A CC-NUMA Computer System for the Commercial Marketplace, *Proc. The 23rd ISCA*, pp. 308 – 317 (1996).
6) Tanaka, H., Muraoka, Y., Amamiya, M., Saito, N. and Tomita, S.(eds.): *The Massively Parallel Processing System JUMP-1*, オーム社 (1996). ISBN4-274-90083-5.
7) Matsumoto, T., Kudoh, T., Nishimura, K., Hiraki, K., Amano, H. and Tanaka, H.: Distributed Shared Memory Architecture for JUMP-1: A General-Purpose MPP Prototype, *Proc. IEEE 1996 International Symposium on Parallel Architectures, Algorithms and Networks*, pp. 131 – 137 (1996).
8) 松本尚: 局所処理と非局所処理を分離並列処理するアーキテクチャ, 第 43 回情処全大 (6), pp. 115 – 116 (1991).
9) 松本尚, 平木敬: Memory-Based Processor による分散共有メモリ, 並列処理シンポジウム JSPP'93 論文集, pp. 245 – 252 (1993).
10) Yang, Y., Amano, H., Shibamura, H. and Sueyoshi, T.: Recursive Diagonal Torus: An interconnection network for massively parallel computers, *The 5th IEEE symposium on Parallel and Distributed Processing*, pp. 591 – 594 (1993).
11) Nishi, H., Nishimura, K., Anjo, K., Amano, H. and Kudoh, T.: The JUMP-1 Router Chip: Versatile Router for Supporting a Distributed Shared Memory, *Proc. of 15th IPCCC*, pp.158–164 (1996).
12) Nishi, H., Anjo, K., Kudoh, T. and Amano, H.: The RDT Router Chip: A versatile router for supporting a distributed shared memory,

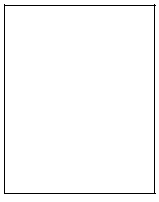
IEICE transaction on Information and Systems, Vol. E80-D, No. 9, pp. 854–862 (1997).
13) Nakajo, H., Ohtani, S., Matsumoto, T., M., K., Hiraki, K. and Kaneda, Y.: An I/O Network Architecture of the Distributed Shared-Memory Massively Parallel Computer JUMP-1, *Proc. 1997 International Symposium on Supercomputer* (1997).
14) Li, K.: A Shared Virtual Memory System for Parallel Computing, *Int. Conf. on Parallel Processing, St. Charls, IL*, pp. 94–101 (1988).
15) Hill, M.D., Larus, J.R. and Wood, D.A.: Tempest: A Substrate for Portable Parallel Programs, *COMPCON '95* (1995).
16) 西村克信, 工藤知宏, 西宏章, 楊愚魯, 天野英晴: 相互結合網 RDT 上での階層マルチキャストによるメモリコヒーレンシ維持手法, 情報処理学会論文誌, Vol. 37, No. 7, pp. 1367 – 1377 (1996).
17) 五島正裕, 松本重光, 森眞一郎, 中島浩, 富田眞治: Virtual Queue: 超並列計算機向きメッセージ通信機構, JSPP95 論文集, pp. 225 – 223 (1995).
18) 佐藤充, 三吉貴史, 松本尚, 平木敬, 田中英彦: シミュレーションを用いた疑似フルマップの定量的評価, 情報処理学会研究会報告, pp. 201 – 224 (1994). ARC107-26.
19) Hennessy, J. L. and Patterson, D. A.(eds.): *Computer Architectuer A Quantitative Approach*, Morgan Kaufmann (1996). ISBN 1-55860-329-8.

(平成 9 年 11 月 7 日受付)
(平成 10 年 4 月 7 日採録)



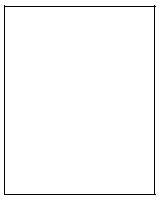
安生健一朗

平成 8 年慶應義塾大学理工学部電気工学科卒業．平成 10 年同大学大学院理工学研究科計算機科学専攻修士課程修了．現在 NEC 所属．WS クラスタ，超並列計算機の研究に従事．



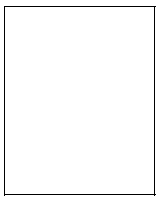
井上 浩明

平成 9 年慶應義塾大学理工学部物理学科卒業．現在同大学大学院理工学研究科計算機科学専攻修士課程在学中．超並列計算機の開発に従事．



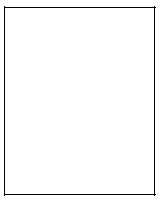
佐藤 充

平成 4 年東京大学工学部電気工学科卒業．平成 6 年同大学大学院工学系研究科情報工学専攻修士課程修了．平成 9 年同大学大学院工学系研究科情報工学専攻博士課程修了．現在 (株) 富士通研究所所属．



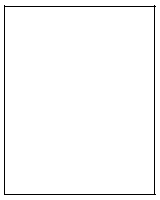
工藤 知宏 (正会員)

平成 3 年慶應義塾大学大学院理工学研究科博士課程単位取得退学．工学博士．東京工科大学情報工学科講師，助教授を経て，現在新情報処理開発機構所属．軽量通信・メモリアーキテクチャの開発に従事．



天野 英晴 (正会員)

昭和 61 年同大大学院工学研究科電気工学専攻博士課程修了．現在, 慶應義塾大学理工学部情報工学科助教授．計算機アーキテクチャの研究に従事．



平木 敬 (正会員)

昭和 51 年東京大学理学部物理学科卒業．昭和 57 年同大学院理学研究科博士課程修了．同年電子技術総合研究所入所．理学博士．計算機アーキテクチャ全般，特にデータフローマシン，分散共有メモリマシンの研究に従事．元岡賞，市村賞，情報処理学会論文賞各受章．平成 3 年から東京大学理学部助教授を経て，平成 7 年から同大学院理学敬研究科教授，現在に至る．昭和 63 年から平成 2 年まで IBM ワトソン研究センター招聘研究員．