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[†1,☆] 井上 浩明[†1] 佐藤 充[†2]
工藤 知宏[†3] 天野 英晴[†1] 平木 敬[†4]

超並列マシンJUMP-1は，1000を超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分かった．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ううえで性能，ゲート数ともに有利であることを示す．

MBP-light: DSM Management Processor on Massively Parallel Processor JUMP-1

Ken-ichiro Anjo,[†1,☆] Hiroaki Inoue,[†1] Mitsuru Satoh,[†2]
Tomohiro Kudoh,[†3] Hideharu Amano[†1] and Kei Hiraki[†4]

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]，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

†1 慶應義塾大学理工学部
Faculty Science and Technology, Keio University
☆ 現在，日本電気株式会社
Presently with NEC Corporation
†2 株式会社富士通研究所
Fujitsu Laboratories, Ltd.
†3 新情報処理開発機構
Real World Computing Partnership
†4 東京大学理学部情報科学科
Department of Information Science, Faculty of Science, The University of Tokyo

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超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，数千プロセッサを超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．

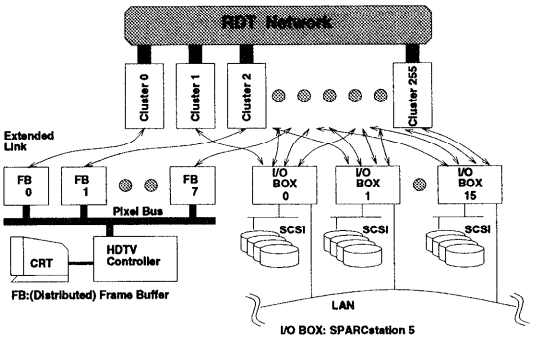


図1 JUMP-1の構成
Fig. 1 Structure of JUMP-1.

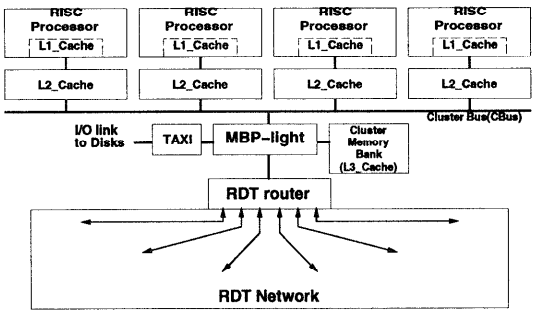


図2 JUMP-1クラスタの構成
Fig. 2 Structure of JUMP-1 cluster.

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ともな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

持つほか，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うえ，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文献9)を参照されたい．

☆ 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レベルの保護を行うため，もう1段アドレス変換を行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1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．

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からのアクセスはすべ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うOBICとクラスタ間で行う必要のあるプロトコル処理を行う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あげる．

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できず，パケット全体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．

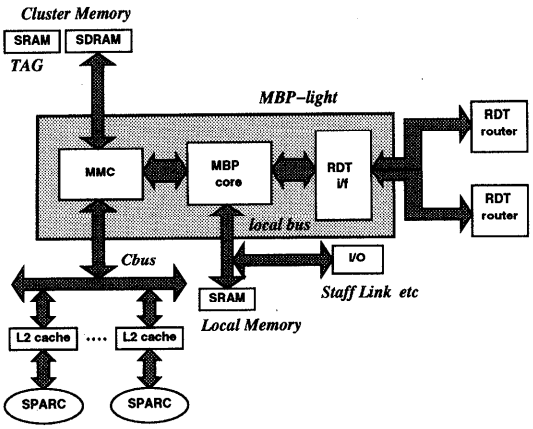


図3 MBP-light の全体構成
Fig. 3 Structure of MBP-light.

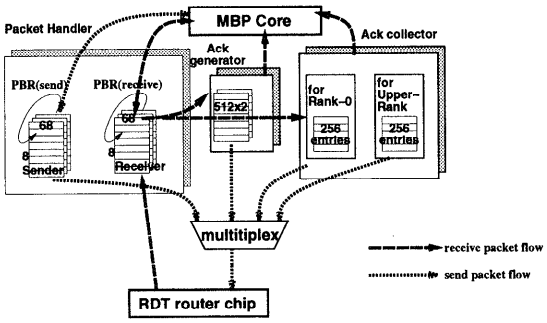


図4 RDT インタフェースの構成
Fig. 4 structure of RDT interface.

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ようにし，パケットの生成を高速に行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う MMC (Main Memory Contoroller)，プロトコル制御を行う MBP Core の3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 17 bit 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 34 bit 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 4 flit 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．5 flit 目にはアクセスアドレスが格納され，6 flit 目にはパケットの種類とコマンド情報が格納される．さらにデータをともなうパケットであれば，7〜14 flit 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 7 flit〜最大 16 flit 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，3 flit 中にすべ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 68 bit × 8 flit 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1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 4 flit 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取り除く．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，68 bit × 8 flit 3 組から成るサイクリックバッファを用いる．最初の 4 flit 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分かっている[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．このときのみ 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．

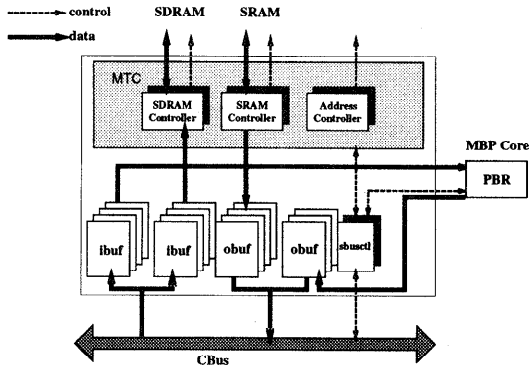


図 5 MMC の内部構成
Fig. 5 Structure of MMC.

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たまると，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持ち，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っている最中や，MBP Coreによってソフトウェア処理が行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っている．MBP Coreがibufのパケットを読み出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節で述べる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，MBP CoreはPBR中にパケットを一度生成しておき，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，データ読み出し時にはエラー検査を行う．

タグアクセス：

MBP-lightの重要な役割の1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を構成するうえで必要不可欠な機能であり，すべてのメモリアクセスのたびに実行されるので高速に行う必要がある．

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うため，MBP Coreが行う処理は，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，68 bit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16 bit幅のGPR（16本）とパケットバッファとしての役割をする68 bit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同様にストールをともな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21 bit，データ長16 bit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21 bit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 × 16 bit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

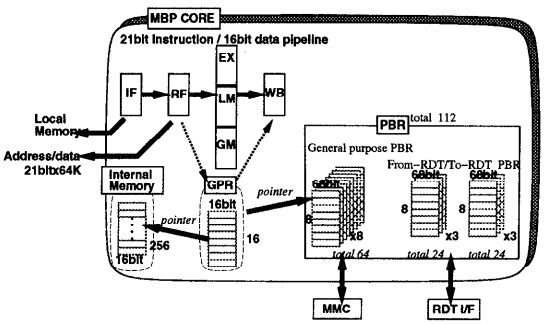


図6 MBP Core の構成
Fig. 6 Structure of MBP Core.

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

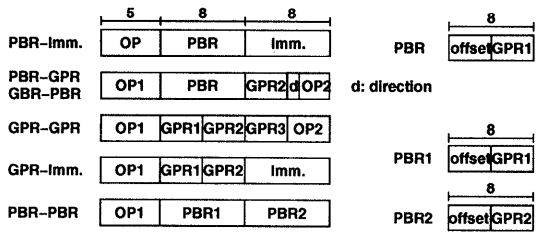


図7 代表的な命令フォーマット
Fig. 7 Typical instruction format.

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 どうしの演算が 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（4 bit 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 8 bit の間で演算が行われる．PBR に対する演算は特定の 8 bit，16 bit 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 8 bit 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とる．GPR は 16 bit 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 16 bit 以内である．また，ディスプレースメントの範囲は 4 bit にすぎない．MBP-light は，CPU（SuperSPARC+）のクロックに同期して動作するため，50 MHz 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+16 bit 演算」を行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う．

分岐：判断，アドレス計算，分岐を行う．

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う．

(3-1) **EX**（実行）：PBRを読み出し16 bit演算を行う．

(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に対して行う．

命令とローカルデータはとも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ともなう割込みをサポートしている．割込みは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．割込み要因はMMC，RDT i/f内部でキューイングされ，すべての割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，割込みにともなって自動的に行う以外にも，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．

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）（3.3 V/5 V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60 MHzで動作が可能であり，JUMP-1はSuperSPARC+のクロックに合わせて50 MHz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．

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

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，ランダムロジックともに大幅に小さな実装が行われていることが分かる．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60 MHzで動作が可能であるが，JUMP-1は50 MHzでの動作が予定されているため，今回の評価では50 MHz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，

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
データ読み出し (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}：50 MHz 動作時クロック数，T_{50}：50 MHz 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

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分かる．

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 4〜10 μsec [5] 要している．このことから，MBP-light の実行速度は 60 MHz で動作した場合 $(5.0 + 0.13 \times$ ホップ数$)$ μsec は，SCLIC とは同等の性能を示しているといえる．したがって，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 32 bit 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われるものと仮定した．また，純粋に Core 部の比較のみを行うため，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．

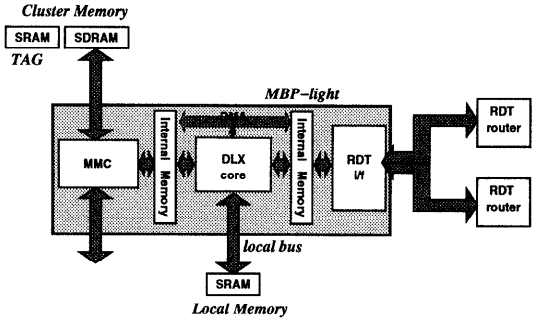


図8 DLX を用いた MBP-light のアーキテクチャモデル
Fig. 8 Architecture model of MBP-light using DLX.

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

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

き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%有利であることが分かる．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クロックほど有利になっている．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分かった．

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．

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	12.7 nsec	22,973
DLX版	13.3 nsec	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

クリティカルパスは32 bit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16 bit 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32 bit 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，クロック周期とも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，32 bit 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分かった．今後は，Stanford FLASHのMAGICチップに関して詳細が公表され次第，比較検討を行う必要がある．また，現在JUMP-1の全体性能をMBP-lightのほか，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の構想の確立にご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

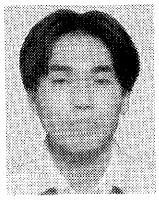
1) Lenoski, D., Laudon, J., Gharachorloo, K., Gupta, A. and Hennessy, J.: The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor, *17th ISCA* (1990).
2) Kuskin, J., et al.: The Stanford FLASH Multiprocessor, *Proc. 21st ISCA*, pp.302–313 (1994).
3) Laudon, J. and Lenoski, D.: The SGI Origin 2000: A CC-NUMA Highly Scaleble Server, *Proc. 24th ISCA* (1997).
4) Chaiken, D. and Agarwal, A.: Software-Extended Coherent Shared Memory: Performance and Cost, *Proc. 21st ISCA*, pp.314–324 (1994).
5) Lovett, T. and Clapp, R.: STiNG: A CC-NUMA Computer System for the Commercial Marketplace, *Proc. 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).
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回情報処理学会全国大会論文集（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 Sucyoshi, 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. 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，ARC107-26, pp.201–224 (1994).
19) Hennessy, J.L. and Patterson, D.A. (Eds.): *Computer Architectuer A Quantitative Approach*, Morgan Kaufmann (1996).

(平成 9 年 11 月 7 日受付)
(平成 10 年 4 月 3 日採録)

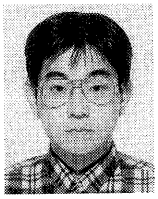
安生健一朗

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．同年NEC入社．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．計算機アーキテクチャの研究に従事．

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年慶應義塾大学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系研究科教授，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．