超並列マシン JUMP-1 のための分散共有メモリ管理プロセッサ

佐藤 充[†1] 天野英晴[†2] 安生健一朗[†2]
周東福強[†2] 西宏章[†2] 工藤知宏[†3]
山本淳二[†4] 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 は高速性を要求されるタグ参照, 応答パケット生成, 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. ゲート数と実行時間を評価を行ない, NUMA-Q で用いられている SCLIC チップに比べて, ゲート数, 処理時間共に有利であること示した.

A ditributed shared memory controller for a massively parallel computer JUMP-1

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 Architecture) 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, 現在の小規模共有メモリ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.

Stanford 大学の DASH [4]/FLASH [3], SGI 社の Origin2000, MIT の Alewife [1], Sequent 社の NUMA-Q [6] などのシステムがこの代表例である. 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である. 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 [11] [14] 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) [15] [16] が提案された. 本論文では,MBP チップにおいて高速処理を要求す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 Architecture 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, 実質的には 1994 年より 8 大学の共同で開発を進めている. JUMP-1 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) [18] により接続されたクラスタから構成される. さらに, 各クラスタは, 強力で経済性に優れた 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) に対して接続される [12].

†1 東京大学工学系研究科 (現在 富士通株式会社)
†2 慶応義塾大学理工学部
†3 東京工科大学情報工学科 (現在 新情報処理開発機構)
†4 慶應義塾大学理工学部 (現在 新情報処理開発機構)
†5 東京大学大学院理学系研究科

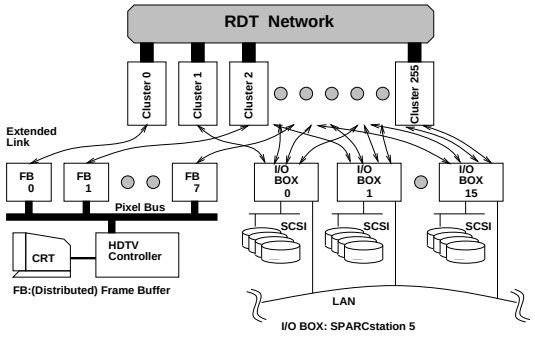

図 1　JUMP-1 の構成

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 (バスチップ) を介して本論文のテーマである MBP-light に接続されている. MBP-light は, 共有バス, メインメモリ, ルータチップに接続され, 全クラスタで共有する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の制御およびデータ転送を実現する.

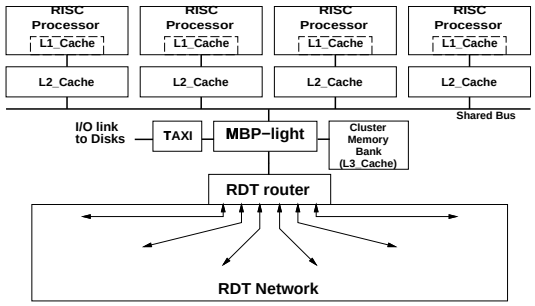

図 2　JUMP-1 クラスタの構成

DASH/FLASH [3], Alewife [1], NUMA-Q [6] 等従来の CC-NUMA は, キャッシュの管理はライン単位で, 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, 高速なメモリに格納することが困難であり, 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(Grobal 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 [5], Tempest [2] 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存在しない場合, 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 Bit-map Directory scheme:RHBD) [17] [13] を用いる. この方法ではキャッシュ管理用のメッセージが, 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, 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) 共有メモリ上で効率の良いメッセージ転送機構を設ける [9].

JUMP-1 の分散共有メモリ管理手法に関する詳細は [16] 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 チップでは, 強力な割り込み処理機能を持つ 32bit のコアプロセッサを内蔵している. パケットはデータ部とヘッダ部に分解さ

☆ MBP ではホームメモリとクラスタの対応関係を柔軟にし, O.S. 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 [16], 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.

れ, ヘッダ部はコアプロセッサに渡され, プロセッサパイプライン中でレジスタとして直接アクセスされる. データ部は専用のバッファに蓄えられ, 転送時にコアプロセッサで生成されたヘッダ部と組み合わされる.

JUMP-1 のプロトコルコントローラで この方法を用いると, 以下の問題点が生じる.

1) 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は, 応答パケットの生成や収集を高速に行なう必要がある. これはコアプロセッサでいかに高速な割り込み処理機能を設けても対処することが困難である.

2) 縮約階層ビットマップディレクトリでは, マルチキャスト用のビットマップをヘッダに入れる必要がある. また, プロトコルやページ制御を柔軟にするためヘッダにつける付加情報が多く, しかもパケットの種類によりヘッダサイズが異なる. さらに, 共有バス, 結合網共にパケットヘッダの bit の利用効率を高めるため, フィールディングが複雑である. また, 共有メモリ上で同期メッセージを転送する必要等から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, ヘッダとデータ部を自動分解してヘッダ部のみをコアのパイプラインで扱う方法の実現が困難である.

この問題点を解決するため, MBP-light では以下の設計方針を採用した.

1) 応答パケットの生成, 収集, タグのアクセス等, 高速動作を要求される処理は, 全て専用ハードウェアにより行なう.

2) コアプロセッサ MBP-Core は, 割り込み機能を持たない簡単な構造であるが, 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ることにより, メモリのアクセス, パケットの生成を高速に行なう.

図 3に上記の方針に基づく MBP-light の構成を示す. MBP-light は, 結合網の制御を行なう 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, 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, これらの構成要素を解説する.

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 Collecter, 応答パケットの自動生成を行う ACK Generater から構成される. この 3 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 MBP-light 1 チップに対してルータチップを 2 個用い, 転送ビット幅を 34bit としている.

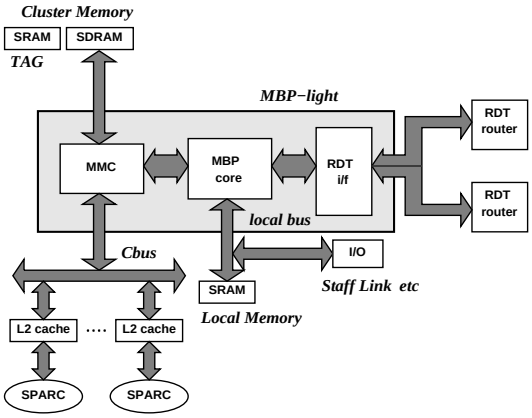

図 3 MBP の全体構成

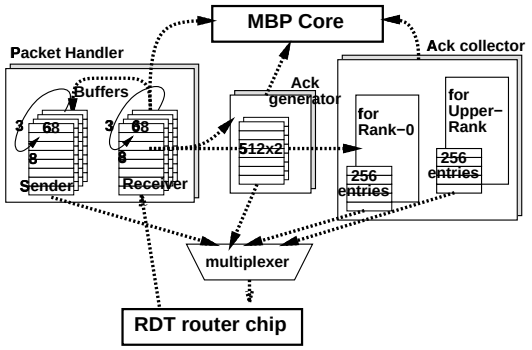

図 4 RDT インタフェースの構成

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 3flit のヘッダはマルチキャスト用ビットマップ, パケット長, 転送情報を含む. これに加え, 4flit 目には ACK 収集時に用いるキーと転送元アドレス, 5flit 目にはパケットの種類とコマンド情報が格納される. 6flit 目にはアクセスアドレスが格納され, 7-14flit がタグ 2bit を付けたデータ, 最後の 1flit はエラー検出用の縦パリティである. データ中のタグは共有メモリ上で同期を取る場合等に用いられる. これに対し, 応答パケットは 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3 組に格納する. これらのバッファは 3 組でサイクリックバッファを構成しており, 順にパケットが格納される. 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 3flit は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 Collecter と ACK Generator を起動する. また縦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 hadler の送信部は, MBP Core, Ack Col-

lecter, 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 3flit 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 Collecter:

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, 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か混雑することがシミュレーションの結果わかっている [10]. 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 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, 応答パケット中の発生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, キャッシュラインが存在するかどうかを示すタグのコピーが格納されており, ACK Generator はこのテーブルを引いて到着したパケットが不必要と判断した場合は, 即座に Packet handler のバッファ中から削除する. 判断はパケットのヘッダ部のみで高速に行われるため, ボディ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 の性能が阻害されることが避けられる.

3.3 MMC

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してクラスタバス, 相互結合網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.

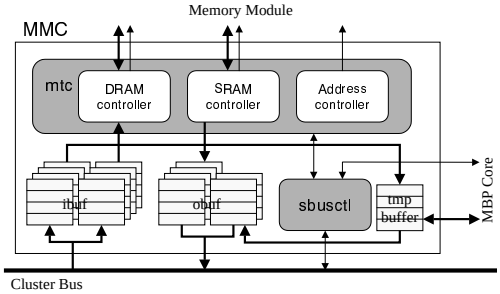

図 5 MMC の内部構成

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)

入出力バッファでは, クラスタバスから受け取ったメモリアクセスパケットおよび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.

出力バッファには, MMC がハードウェア的に生成するリプライパケットの他に, MBP Core がクラスタバスへ出力するパケットを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，ブロックタグ用 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では，データ書き込み時には ECC コードを付加し，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，ブロックタグ用 SRAM と 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タグ用 SRAM のデータを検査し，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，処理を MBP Core に依頼する．タグ用 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の変更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．

この問題を解決するため，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．演算がレジスタとバッファ間で行われることからこの構成を Buffer-Register Architecture と呼ぶ．

図 6 に MBP Core の構成を示す．MBP Core 自体は命令長 21bit データ長 16bit の 4 段パイプラインの単純な構成である．MBP Core は主に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 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．

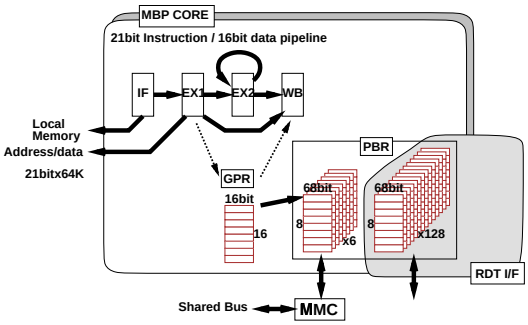

図 6　MBP Core の構成

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 は 2 ポートで 2 アドレッシングである．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，GPR1 の内容と，GPR2 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も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 間の演算はシフト付きデータ転送等ごく限られたものである．

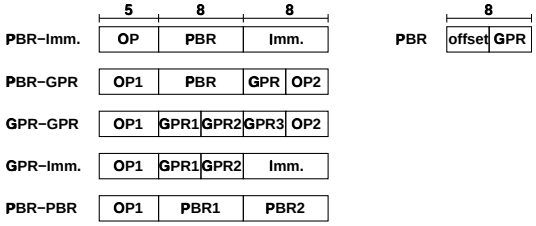

図 7　代表的な命令フォーマット

PBR は，RDT 送信受信用がそれぞれ 24 本，汎用

が 64 本である．MMC と RDT インタフェースの構造上の相違から，インタフェースの方法は異なっている．RDT 送受信用の PBR 64 本はそれぞれ 3 組に分離されており，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．これに対してクラスタメモリのアクセス，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，命令内でブロック転送の形で行われ，8 クロックを要するが，PBR の任意の場所に関して可能である．

プロトコルにより，利用するパケットの形式は大体決まっているため，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．汎用目的 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．

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．このため，60MHz の 1 クロックの間に，「GPR の 2 ポートの同時読み出し+16bit 演算」あるいは，「GPR の読み出し+ディスプレースメント演算+SPR の読み出し」が可能である．すなわち，MBP Core のパイプラインは比較的長いピッチの中で通常のパイプライン中で行う処理を 2 回分行なうことで，ステージ数を少なくするアプローチを取っている．具体的には，4 つのステージでは以下の操作を行う．

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

(2) **EX1(実行 1):**

GPR-GPR: 命令コード中で指定された GPR 双方の値を読み出し，演算を行う．

PBR-GPR/PBR-PBR: 命令コード中で指定された GPR の値を読み出し，ディスプレースメントを計算し，PBR を読み出す．

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．

(3) **EX2(実行 2):** 一部の命令のみが実行するステージで，繰り返し実行を許す．

PBR-PBR: PBR を再度読み出す．

メモリアクセス: ローカルメモリのアクセス．

その他，共有バスパケット転送，クラスタメモリのアクセス，内部メモリのアクセスを行う．

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

パイプライン制御を簡単にするため，ローカルメモリアクセス命令，PBR-PBR 命令は 1 クロックパイプラインをストールさせる．分岐命令は 1 クロックの遅延スロットを必要とする．EX2 ステージで共有バスパケット転送，クラスタメモリのアクセスが行われる場合，パイプラインは 7 クロックストールする．

イベントの処理:

MBP Core は割り込み機能そのものはもたないが，I/O, RDT インタフェース,MMC からの要求に速やかに対応する必要性はある．このため，MBP Core は MMC, RDT インタフェース，外部機器からの要求を受け付けるイベントレジスタを持ち，レジスタの内容に応じてテーブルジャンプを行う．MBP-Core はひとつの作業を終了したら，直ちにイベントレジスタの内容によりテーブルジャンプする動作を繰り返す．

4. 評　価

4.1 動作速度

MBP-light は，現在 VHDL レベルの設計がほぼ終了し，動作検証および論理合成後の仮遅延シミュレーションの段階である．基本的な操作に要する遅延時間を論理シミュレーションにより測定した結果を表 1に示す．MBP-light 各部は 60MHz で動作するように設計されている☆.

表 1 代表的操作の動作時間

操作	Inst.	CLK	T_1	$T_{1/3}$
RDT 受信	1	20(4)	334	-
RDT 送信	1	18(1)	301	-
Bus 受信	1	12	200	-
Bus 送信	1	10	167	-
Cl.Mem.	1	8	134	-
ACK 収集	0	5	84	6
ACK 返送	0	11	184	14
R.R.1(L)	15	58	969	10
R.R.2(R)	11	56	935	22
R.R.3(L)	10	38	635	18
R.R.Total	36	152	2538+5H	50+5H
W.I.	21	60	1002	28

RDT 送／受信: RDT パケットの送受信
BUS 送／受信: Bus パケットの送受信
Cl.Mem.: クラスタメモリに対する 1 Line データ転送
R.R.: Clean 領域のリモートホストに対する 1 line Read
W.I.: Shared 領域に対する書き込みと無効化送信
H: RDT ルータにおけるパケットの転送 hop 数
Inst: 命令数
CLK: クロック数
T_1: 60MHz 動作時の実行時間 ($nsec$)
$T_{1/3}$: JUMP-1/3 の実行時間 (μsec)

表の上半分は基本的な操作の実行時間である．RDT パケットの受信，送信は，MBP Core 内の PBR を直

☆ 配置配線の結果タイミングについて変更する可能性がある

接用いる．ここに示す時間は, パケット (最大長:16flit) の先頭を受信してから MBP Core 内でアクセス可能になる時間またはパケットの最後のフリットを送出し終わる時間である．( ) 内の数値はバッファの切替えおよび MBP Core のテーブルジャンプに要する時間で, それ以外が RDT インタフェースでの送受信時間となる．これに対し, 共有バスパケットの送受信時間は, バス制御用のバッファへの転送時間を示し, 実際にプロセッサの L2 キャッシュとのやりとりにはさらにバス上での転送時間を加える必要がある．クラスタメモリの 1 ブロックアクセス時間はキャッシュラインに相当する 32byte データを PBR との間で転送して, MBP Core がアクセス可能になるまでの時間である．

表の下半分に分散共有メモリ管理における代表的な処理の実行時間を示す．応答パケットの収集と, 応答パケットの返送は RDT インタフェースのハードウェアが行なうため, きわめて高速である．これに対して, MBP Core のプログラムに要する制御はある程度の時間を要する．ここでは, 2.3 節に示した他のクラスタのメモリのキャッシュブロックの読み出し (ブロックがクリーンの場合) に要する時間 (Read Requet) と書き込みと無効化要求送信に要する時間 (Write Invalidate) である．Read Request は, 要求を発生したノード (Local:L) で 1. 要求パケットを送出する処理, 2. 要求を受けたリモートノードの応答 (Remote:R), 3. Local ノードでの受信の 3 ステップに分けることができ, 全体で, $2.54\mu sec + 5\times$(ネットワークのホップ数) となる．また書き込みと無効化要求の送出には約 $1\mu sec$ を要する．

表中に, JUMP-1 のソフトウェア開発用テストベッド JUMP-1/3 [8] で同様の処理を行なった場合の実行時間を示す．JUMP-1/3 では, MBP-light の代わりにプロトコル制御用に 32bit の DSP (TMS320C40) を用いている．I/O, ルータは同じ構成である．両者を比較すると, ハードウェアが行なう処理に関しては約 70 倍, MBP Core で行なう処理に関しては約 20 倍の性能を実現していることがわかる．

NUMA-Q で用いられている SCLIC チップ, FLASH で用いられている MAGIC でも同様の操作の処理時間が評価されている． SCLIC チップは 4 ノード先の Read Request 操作に 4-10μsec 要し [6], MAGIC は同様の操作を 1.1μsec で実現する [3]．このことから, MBP-light の実行速度は MAGIC には及ばないが, SCLIC よりは高速であることがわかる．

4.2 トレースデータによるシミュレーション

上記データを基に, SPLASH 並列プログラム集 [7] の中から, MP3D, LU, 他に FFT のアドレストレースを用いて行なった. このシミュレーションは, 以下の点で JUMP-1 の構成とは異なっている．(1)R3000 のインストラクションセットのトレースであり, Super Sparc+と若干の差がある．(2) クラスタ構造ではなく, 単一プロセッサのシステムとしている．(3) 結合網での衝突は考慮されていない. シミュレーション環境は文献 [8] に詳細を示す.

via home の単純な invalidate プロトコルおよび update プロトコルを想定して 8 ノードのシステムに関して MBP-light でのオーバヘッドを測定した結果を図 8に示す． この図では,Remote と記された部分は MBP-light が他のノードから依頼された処理を行なっている時間であり, この間 CPU は並行に動作することができることから純粋のオーバヘッドではない. この結果からは, アプリケーションによってオーバヘッドの割合が非常に異なっていることがわかる．このうち特に MP3D と FFT の Update プロトコルでは Write Miss のオーバヘッドが大きいが, これは Write Miss が起きるとプロセッサをストールさせる最も簡単な方法を用いたことによる．これは緩いコンシステンシィモデルの利用により改善することができる．また,FFT の Invalidatate プロトコルは Page Fault のストール (PF stall) が大きいが、これは問題の実行時間に対するコールドミスの割合が大きいためと考えられる.

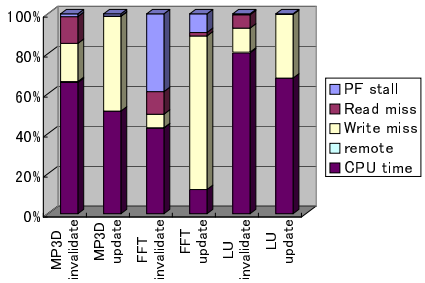

図 8 実行時間の内訳

今回の評価から,MBP-light の利用により現実的な稼働率が得られることが明らかになった. しかし, 今回のシミュレーションは, 環境が実際の JUMP-1 と異なる点が多いため, さらなる評価が必要である.

4.3 ゲート数

MBP-light のハードウェア量を評価した結果を表 2 に示す.

表 2 ゲート数

ブロック名	換算ゲート数
MBP Core	13,355
RDT i/f	15,089
MMC	16,008
Embeded RAM	147,130
総計	191,582

表によると, ゲート数の多くは内蔵 RAM が占め, ランダムロジックはきわめて少量のハードウェアで構

成されていることがわかる. SCLIC チップは, ランダムロジックが 140,000 ゲート, 内蔵 RAM が 152,000 ゲートである [6]. これと比べると, MBP-light の内蔵 RAM の量はほぼ等しいが, ランダムロジック部, 特にコアプロセッサが大幅に小さいことがわかる. これは MBP Core に Buffer-Register Architecture を導入し, ハードウェアを簡単にしているためである. MAGIC はハードウェア量の詳細評価が発表されていないが, SCLIC チップよりもさらに複雑かつ大規模な構成になることが予想される.

5. おわりに

超並列マシン JUMP-1 用の分散メモリ管理プロセッサ MBP-light を設計した. MBP 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 SCLIC チップに比べて, ゲート数, 処理時間共に有利であることが明らかになった. シミュレーションの結果からも, 実用的な稼働率が得られることが期待される. さらなる評価を行なう一方, 実装を終了し, 実機による評価を行なう予定である.

謝　　辞

文部省重点領域研究で共同に開発を当たった京都大学の富田眞治博士, 中島浩博士 (現豊橋技術科学大学), 五島正裕博士, 神戸大学の中條拓伯氏に感謝いたします. 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氏に感謝いたします. MBP 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.

参 考 文 献

1) D. Chaiken and A. Agarwal. Software-Extended Coherent Shared Memory: Performance and Cost. In *Proc. The 21st ISCA*, pp. 314 – 324, 1994.
2) Mark D.Hill, James R.Larus, and David A.Wood. Tempest: A Substrate for Portable Parallel Programs. In *COMPCON '95*, 1995.
3) J. Kuskin and al. et. The Stanford FLASH Multiprocessor. In *Proc. The 21st ISCA*, pp. 302 – 313, 1994.
4) D. Lenoski and et al. The Stanford DASH Multiprocessor. *IEEE Computer*, Vol. 25, No. 3, pp. 63 – 79, 1992.
5) K. Li. A Shared Virtual Memory System for Parallel Computing. In *Int. Conf. on Parallel Processing, St. Charls, IL*, pp. 94–101, August 1988.
6) T. Lovett and R. Clapp. STiNG: A CC-NUMA Computer System for the Commercial Marketplace. In *Proc. The 23rd ISCA*, pp. 308 – 317, 1996.
7) E. Rothberg, J.P. Smith, and A. Gupta. Working sets, Cache Sizes, and Node Granularity for Large Scale multiprocessors. In *Proc. of the 20th ISCA*, 1993.
8) 安生健一朗, 他. 分散共有メモリを持つ ws クラスタ: JUMP-1/3. JSPP-97 論文集, May 1997.
9) 五島正裕, 他. 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, May 1995.
10) 佐藤, 三吉, 松本, 平木, 田中. 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. 情処アーキテクチャ研報, pp. 201 – 224, July 1994. ARC107-26.
11) H. Tanaka, Y. Muraoka, M. Amamiya, N. Saito, and S. Tomita, editors. *The Massively Parallel Processing System JUMP-1.* オーム社, 1996. ISBN4-274-90083-5.
12) H. Nakajo, S. Ohtani, T. Matsumoto, Kohata M., K. Hiraki, and Y. Kaneda. An I/O Network Architecture of the Distributed Shared-Memory Massively Parallel Computer JUMP-1. In *to appear in Proc. 1997 International Symposium on Supercomputer*, 1997.
13) 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, July 1996.
14) K. Hiraki, H. Amano, M. Kuga, T. Sueyoshi, T. Kudoh, H. Nakashima, H. Nakajo, H. Matsuda, T. Matsumoto, and S. Mori. Overview of the jump-1, an mpp prototype for general-purpose parallel computations. In *Proc. IEEE International Symposium on Parallel Architectures, Algorithms and Networks*, pp. 427 – 434, 1994.
15) 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, October 1991.
16) 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, May 1993.
17) T. Matsumoto, T. Kudoh, K. Nishimura, K. Hiraki, H. Amano, and H. Tanaka. Distributed Shared Memory Architecture for JUMP-1: A General-Purpose MPP Prototype. In *Proc. IEEE 1996 International Symposium on Parallel Architectures, Algorithms and Networks*, pp. 131 – 137, 1996.
18) Y.L. Yang, H. Amano, H. Shibamura, and T. Sueyoshi. Recursive Diagonal Torus: An interconnection network for massively parallel computers. *The 5th IEEE symposium on Parallel and Distributed Processing*, pp. 591 – 594, December 1993.