クラスタ コンピューティング向け高速ネットワークカードの
バス インタフェース部

System bus interface part of fast network card for cluster computing

朝川 達

平成12年2月15日

内容梗概

現在の分散システムは，システム内の計算機を複数利用して，細粒度の並列アプリケーションを効率よく実行することが困難である．分散システムにおいて，細粒度の並列処理を実行することを妨げているのはノード間の低速，高遅延な通信である．

我々の研究室では，これら分散システムに存在する問題を解決し，並列アプリケーションが実行可能かつ，快適なマルチユーザ環境を提供する計算機システムである「コンピュータ コロニー」の実現を目指している．現在，「コンピュータ コロニー」のプロトタイプ ハードウェアを開発している．これは，既存のワークステーションと専用ネットワーク カードからなるノードを高速ネットワークで結合した共有メモリベースのワークステーション クラスタである．

この専用ネットワーク カードは要件として，1) 高速，低遅延な通信が行えること，及び 2) プロトタイプとして，様々な通信方式を実験，評価できるものであることの2点を満たさなければならない．1) 高速，低遅延な通信を実現するためには，通信資源保護のためのシステムコールやプロトコル処理のオーバ ヘッドを低減しなければならない．この課題に対し我々は，ネットワーク インタフェースをシステム バスに接続し，通信を通常のメモリアクセスと同様に扱うとともに，仮想記憶管理機構による資源保護を利用することで対応する．これにより，ネットワーク インタフェースがプロセッサのキャッシュのコヒーレンスを維持しつつ直接メモリ アクセス行えることが可能となり，ローカルメモリの一部をリモートメモリに対するキャッシュとして利用することが可能となる．さらに，システム規模に見合った軽量な通信プロトコルを用い，フロー制御をネットワーク インタフェース上で行うことによりオーバヘッドの低減を図る．そして，2) の要件に対しては，ネットワークカード上にプロトコルプロセッサ及びFPGA(Field Programable Gate Array) を搭載し，「コンピュータ コロニー」にとって最適な通信方式を探るためにプログラマブルな構成をとることで対応する．

本研究では，ネットワーク カードの一部であるバス インタフェース部の一部のFPGA への実装を行った．このバス インタフェース部はシステム バスとのインタフェースを行う部分であり，システムバス中に流れるトランザクションをスヌープし，リモートメモリ アクセスの必要性を判定するとともに適切な通信を開始する．また，ローカルメモリにキャッシングされたデータのコヒーレンス制御を行う．このバス インタフェース部はバス インタフェース コントローラ部，バス送受信バッファ部，及び，ローカルメモリをキャッシュとして利用する際に必要な情報をキャッシュ ブロックサイズ単位で保

持する主記憶タグから構成される．

バス インタフェース コントローラ部は常にシステムバスをスヌープしており，システム バス上にメモリ アクセスが発生すると，バスに流れる物理アドレスによって主記憶タグを参照し主記憶の状態を調べる．そのブロックが無効であることが判明すると，無効を意味するコヒーレンス制御信号をバスに流し，主記憶が無効なデータを送るのを阻止するとともに，適切な時間を経て有効なデータをシステム バス上に流す．有効なデータの供給は，バス送受信バッファ部が行う．メモリ アクセス中に無効の信号がバスに流れると，バス送受信バッファ部はリモートメモリアクセスが必要であると判断し，リモートメモリへのアクセスを実行する．トランザクションのメッセージをバッファに取り込む．有効なデータが送られて来ると，バッファから取り出しシステムバスへと流す．このようにして，無効なデータに対するアクセスがあっても，有効なデータを送ることができる．

また，トランザクション中に流れるメッセージは，デッドロック回避のための優先順位制御を行うために，要求メッセージと返答メッセージに区別してバッファに収める．この管理を，ディレクトリとキューを用いて管理し，操作を容易に行えるようにした．

以上の設計によりネットワーク インタフェースによるユーザ レベルの通信，及びハードウェアによるリモートメモリの内容のキャッシュ コヒーレンス制御が実現する．

Abstract

Existing distributed systems cannot execute parallel applications efficiently by using computers in system. The slow and high latency communication prevents users from execution of a fine grain parallel processing.

Our laboratory aims to realize "Computer Colony" ,which is a computer system that enables users to process parallel applications and provides comfortable multiuser computing environment. Now, we develop prototype hardware for "Computer Colony". It is a workstation cluster connected high speed network and based on distributed shared memory that consists of several workstations equipped with high speed network card on their system bus, and it realizes high speed and low latency communication between nodes.

The network card is requested that 1)it can execute high speed communication and that 2)it can experiment and evaluate communication method. 1)For the realization of high speed communication for fine grain parallel processing, overhead of communication resource protection and protocol processing must be reduced. Therefore we connect a network interface to system bus. By this connection the communication between nodes can be handled like memory access ,using resource protection by control system for virtual memory. And, because network interface can memory access directly, a part of local memory can be used as cache for remote memory. Furthermore flow control is executed on network interface. These realize low overhead. For 2) programmable structure , we arrange protocol processor and FPGA(Field Programmable Gate Array) which are programmable devices on network card, to find out the best method of communication for "Computer Colony"

I have designed a part of the system bus interface inside the FPGA. This part controls the interface between a system bus and the network card. This part snoops transaction on system bus and judges needs of communication and begins communication. This part consists of the bus buffer part for sending and receiving, the system bus interface contoroller part and main memory tags which maintains memory state information for cache block.

The system bus interface part always snoops a system bus. When a processor accesses the main memory, the system bus interface controller part accesses the main memory tags to see the state of the memory block pointed by the address on the system bus. If the memory block is invalid, the system bus interface controller part prevent the main memory from supplying invalid data and the system bus buffer part for sending and receiving communicate with the machine which has the valid data block. And the work is executed by the bus buffer part for sending and receiving. The bus buffer part for sending and receiving detects needs for remote memory access. When remote memory access is needed for this part buffers address flits and data flits which carry information in transaction. And when this part received a valid data this part send to system bus.

Flits carried in transaction is discriminated reply flits and request flits, avoiding dead lock. This discrimination is controlled by directory and queue. This makes operation easy.

By the above-mentioned designs communication at user level and cache coherence control for remote memory is realized.

1 はじめに

最近，パーソナル コンピュータやワーク ステーションなどの計算機が，マイクロプロセッサの性能向上に伴い，高速化，小型化，低価格化している．研究室やオフィスなどではこれらの計算機をネットワークで接続し，計算機資源を共有する分散システムが一般的に普及している．現在普及している分散システムは，プリンタ，ディスクといった周辺機器を共有することができ，個人に対し端末が割り当てられ，通常利用に関しては快適な作業環境が実現されている．

しかし，現状では，分散システム内の計算機資源を十分に生かし切っているとは言い難い．例えば，プロセッサやメモリなどの計算資源の利用率は，一般的にかなり低いものとなっている．分散システム内の計算機全体に負荷がかかっているような状態はあまり存在しない．大規模なジョブを投入し，それを計算資源に余裕のある計算機を利用して並列に実行させるというようなことは困難である．これは，現在普及している分散システムのノードとして採用されている計算機は，もともと単体での利用を想定されているものにネットワークの機能を付加したものであるため，通信コストが非常に高いことに依っている．通信コストを高くしている要因として，オペレーティングシステムによる資源保護のためのシステムコールや，汎用性はあるが煩雑な通信プロトコルなどによるオーバヘッドなどが挙げられる．

そこで我々は，従来の分散システムの問題点を解決し，細粒度の並列処理を行うことにより，ネットワーク全体の計算資源を有効利用し，ユーザにとって快適な効率の良い，並列アプリケーションを実行するシステムとして「コンピュータ コロニー」を提案する．

本稿では，「コンピュータ コロニー」に必要な高速通信を実現するためのハードウェアの構成法と，現在開発しているワークステーション クラスタを基本とした分散システム ベースのプロトタイプ ハードウェアの構成，及びワークステーション クラスタをネットワーク接続する高速ネットワークカードの構成を示し，さらにネットワークカード内の実際に設計を行ったバス インタフェース部について述べる．

以下，2 章では，我々が実現を目指している並列応用を指向した分散システム「コンピュータ コロニー」の概要を示し，そのプロトタイプ ハードウェアの概要を述べる．3 章では，バス インタフェースについてその構成をのべ，4 章で，バス インタフェースの機能，動作について述べる．最後に，5 章においてまとめを行う．

2 コンピュータ コロニーの概要

本章では，並列応用を指向した分散システム「コンピュータ コロニー」について述べる．コロニーとは生物学用語で「群体」を表し，個々が有機的に結合した集合構造をとる近接した状態で住んでいる同種の生物の集合を意味する．すなわち，「コンピュータ コロニー」は群体のように複数の計算機が協調しながら動作し，(ユー

ザに対して) 全体として 1 つの計算機であるかのように振る舞う計算機システムである．

「コンピュータ コロニー」はハードウェア，ソフトウェア (オペレーティング システム) を含めた総合的なシステムである．現在の分散システムをハードウェア/ソフトウェアの双方から再検討し，問題点を明確にする．その上でそれを解決する分散システムの再構成を行う．

2.1　コンピュータコロニーの目標

「コンピュータ コロニー」は，分散システムを基本としたシステムである．

その目標は分散システムを拡張し，以下に挙げる事柄を達成することである．

1. ユーザにとって快適なコンピューティング環境．
2. 導入，維持，拡張の容易さ．
3. システム内の計算資源を有効利用し高性能な計算機環境．

すなわち，従来の分散システムで実現されている，ユーザにとっての快適性，導入，維持，拡張の容易性に加えて，その問題点である，遊んでいる計算資源を有効利用し高性能な計算機環境を構築することである．そして，システム内の計算資源を有効利用する方法として，並列アプリケーションの実行に対し動的な負荷分散を行い，システム内の計算資源を有効に利用する．

2.2　コンピュータ コロニーの要件

分散システムには快適性や導入，維持，拡張の容易さといった長所がある．しかし，「コンピュータ コロニー」を実現するためには，現在の分散システムにおける以下に述べる通信コストの問題を解決しなければならない．

現在の分散システムでは，並列処理を行うことはほとんど不可能に近い．それは，通信コストが非常に高いことによる．

最近では，高バンド幅のネットワークを用いてスループットの高い通信を行うことができるようになりつつある．しかし，並列処理を行う際に重要になってくるのは，ネットワークのスループットの高さではなく通信ごとのレイテンシの短さである．レイテンシを長くしている要因の大部分を占めているのが，通信に関わるソフトウェアのオーバヘッドである．

このようなオーバヘッドには以下のようなものがある．

- 通信に関わる資源保護のためのシステムコール．
- TCP/IP などのプロトコル処理．

クライアント/サーバのような粒度の粗い並列処理であればさして問題にならないが，細粒度の並列処理を必要をする並列アプリケーションを動作させ，高い性能を実現するには，このようなオーバヘッドを大幅に削減する必要がある．

2.3　高速通信のための要件

高速通信を実現するための要件には以下のようなものがある．

- システムコールを介さない通信．
  通信資源保護のためのシステムコールにはミリ秒単位の膨大な時間がかかる．これを削減しなければならない．
- プロトコル処理によるオーバヘッドの削減．
  柔軟で汎用性はあるが，複雑なプロトコルによるオーバヘッドも大きい．これもまた削減しなければならない．
- 通信回数の低減
  他ノードへアクセスするよりはノード内のみで処理したほうが当然速い．従って通信回数を低減することも必要である．

2.3.1　実現に向けた方針

要件を満たすために以下の方針をとる．

- ネットワーク インタフェースをシステムバスに接続．
  メモリアクセス，キャッシュの操作をネットワークインタフェースが直接行うことができる．

- 共有メモリベースの通信プリミティブ．ネットワーク上に仮想的なアドレス空間をつくり，それを用いて通信を行う．

2.3.2 実現

前節の方針により以下のように高速通信を実現できる．

1. リモートメモリの内容のキャッシングネットワーク インタフェースをシステム バスに接続することで，ネットワーク インタフェースが，直接，キャッシュに対する完全な操作を行うことができる．従って，リモートメモリ アクセスの際に，リモートメモリの内容を，自ノードの主記憶にキャッシングし，そのコヒーレンス制御をハードウェアが行うことができ，通信コストを削減できる．

   メインメモリ上の物理ページ以下のように区別し，リモートメモリのキャッシングをサポートする．

   - ホーム ページ
     物理メモリが実装されている物理アドレスに位置する．メモリ上にデータの実体が存在する．ホーム ページが存在するノードをホーム ノードという．
   - ダミー ページ
     物理メモリが実装されていない物理アドレスに位置する．ダミー ページに対するアクセスが発生すると，ネットワーク カードが検出し，対応するホーム ページにアクセスすべく，ノード間通信を行う．
   - コピー ページ
     物理メモリが実装されている物理アドレスに位置する．ホーム ページの内容をキャッシングしている．従ってホーム ページとの間でコヒーレンス制御を行う必要がある．また，無効化されている場合は，ダミー ページにアクセスした場合と同様に対応するホーム ページにアクセスする．

2. ユーザレベル通信
   ユーザレベル通信とは，OS のシステム コールを介さずに行う通信である．OS が介入する通信では，通信時にシステム コールによって通信資源の保護，通信イベントの通知などを行っており，そのための処理やコンテクスト スイッチが通信時のオーバヘッドとなっていた．

   これに対し，我々が提案するネットワークインタフェースでは仮想記憶機構の資源保護のメカニズムとバスのスヌープ機構を用いることで以下のように通信オーバヘッドを軽減する．

   - 通信資源の保護
     システム バスを利用した通信では，MMU(Memory Management Unit，メモリ管理ユニット) によるメモリ保護機構を利用することで，通信機構をユーザに対して安全に解放することができる．すなわち，通信時にユーザが行う仮想アドレスでのメモリアクセスを，MMU が物理アドレスでのメモリアクセスに変換する．この物理アドレスへの変換により，通信資源が通常のメモリアクセスと同様に保護される．
   - 自発的な通信
     システム コールによる，通信開始の通知が行われないので，ネットワークインタフェースがシステム バスをスヌープして，自発的に通信を開始する必要がある．本通信機構では，先に述べた 3 種の物理ページ (ホーム ページ，コピー ページ，ダミー ページ) 及び主記憶タグ (共有，無効) を用いて通信の必要性を判断する (図 1)．
     - 物理アドレスがコピー ページを指している場合 ：
       主記憶タグを調べ，無効なブロックにに対するアクセスであった場合には，リモートメモリ アクセスを行う．
     - 物理アドレスがダミー ページをさしている場合 ：

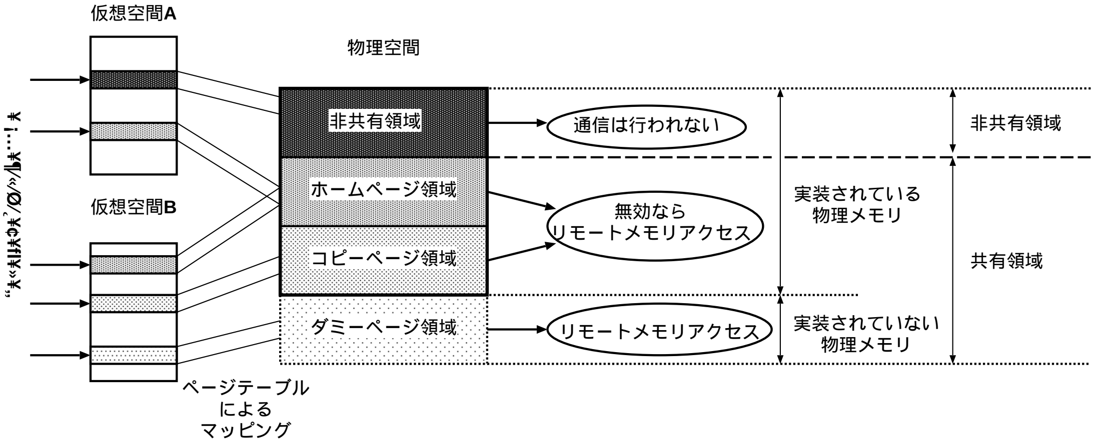

図 1: 物理ページの管理

アクセスしているブロックは物理メモリ上にないので，リモートメモリ アクセスを行う．

従って，メモリ アクセスという形のユーザ レベルで，通信を行うことができ，システム コールを介さない通信を行うことができる．

2.4 プロトタイプ ハードウェア

「コンピュータ コロニー」に必要な高速通信機構を実験，評価するための環境としてワークステーションを用いたプロトタイプ ハードウェアを構成する．

2.4.1 方針

プロトタイプ ハードウェアは以下の方針で，設計する．

1. 低コストでの評価
   従来の分散システムのノードであったPC/WSに新たに開発するネットワーク カードを接続し，高速ネットワークでクラスタを構成する．
2. 様々な通信方式の実験ができる
   プロトタイプとして様々な通信方式が実験できるようにプログラマブルな要素を含める．

2.4.2 構成

プロトタイプ ハードウェアは，低コストでコンピュータ コロニーのハードウェア環境の評価を行うため，従来の分散システムのノードであったPC/WSにネットワーク インタフェースとして新たに開発するネットワーク カードを接続し，高速ネットワークでワークステーション クラスタを構成するという方針をとる．プロトタイプ ハードウェアの全体図を図2に示す．

現在開発しているプロトタイプ ハードウェアはノード マシンとしてSPARC station 20を用いる．SPARC stationのプロセッサであるSPARCはMBusという独自のシステムバスを持っている．SPARC station 20はマザーボード上に2つのMBusスロットがあり，1つには既にプロセッサモジュールが差されている．もう1つのスロットにネットワークカードを差すことで，ネットワーク カードをMBusに接続でき，システム バスを利用した通信を行うことができる．またSPARC station 20はSMP(Symmetric MultiProcessor:対称型マルチプロセッサ)に拡張できる仕組みになっており，MBusはそのためのキャッシュコヒーレンス信号をサポートしている．ネットワーク カードははこの信号を利用して，キャッシュに対する操作を行うことができる．

従って，このSPARC station 20のMbus拡張用スロットにネットワークインタフェースで

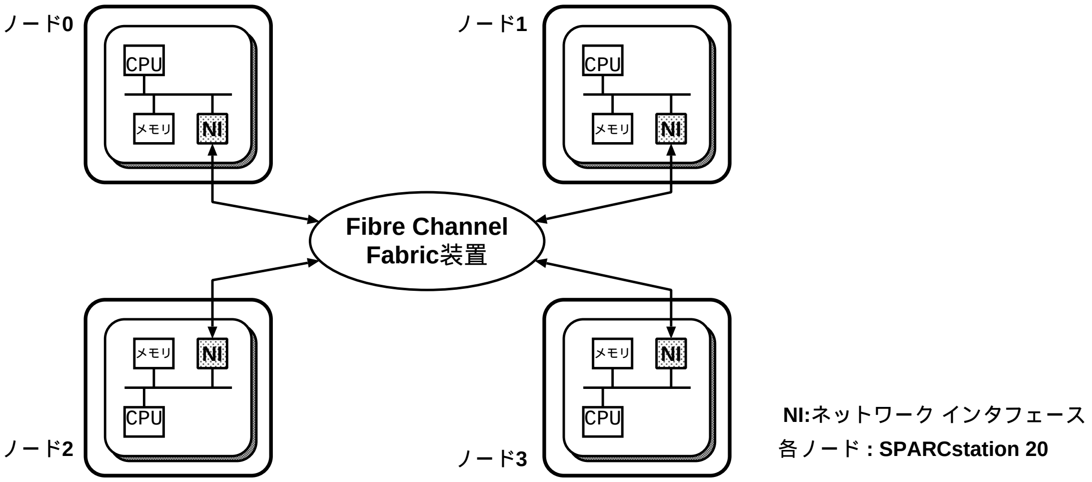

図 2: プロトタイプ ハードウェアの構成

あるネットワーク カードを差したものをノードマシンをして採用する．これらのノード4台を，Fibre Channelでつないでプロトタイプ ハードウェアを構成する．

2.5 ネットワーク カード

プロトタイプ ハードウェアのネットワーク インタフェースとして現在開発中のネットワーク カードについて述べる．このネットワーク カードは，SPARC station 20のシステム バスである，MBusに接続され，これまでに述べてきた高速通信を実現しようとするものである．

2.5.1 構成

図3にネットワーク カードの構成を示す．各部分の役割は以下の通りである．

1. **MBus** インタフェース部．
   - 主記憶タグ部 (**B_map**)
     自ノードシステムの物理メモリのアドレス空間全体をカバーする主記憶タグである．各ブロックについて，共有/無効等の情報を保持する．主に，ノード間で共有するブロックのコヒーレンス制御に利用する．また，ネットワーク カードが独自にリモートメモリ アクセスの必要性を関知する際にも利用する．
   - **MBus** インタフェース コントローラ
     MBus トランザクションを常にスヌープし，適宜B_mapを参照し，コヒーレンス制御の信号をMBusに流す．MBus，送受信バッファに対して適切な処理を行う．
2. 送受信バッファ．
   - **MBus** 用送受信バッファ部 (**M_buf**)
     MBus用送受信バッファ，及びそのコントローラ．MBusを流れるトランザクションの情報を蓄積する．データ中にはトランザクションのタイプ，サイズ，物理アドレスなどの情報を含む．
   - **Fibre Channel** 用送受信バッファ (**F_buf**)
     Fibre Channelネットワーク用の送受信バッファ，及び，そのコントローラ．
3. プロトコルプロセッサ部．
   - プロトコル プロセッサ
     最も頻度の高い処理はリモートメモリ アクセス時の物理アドレスとネットワーク アドレスの変換である．その他，ハードウェアで実装が困難な

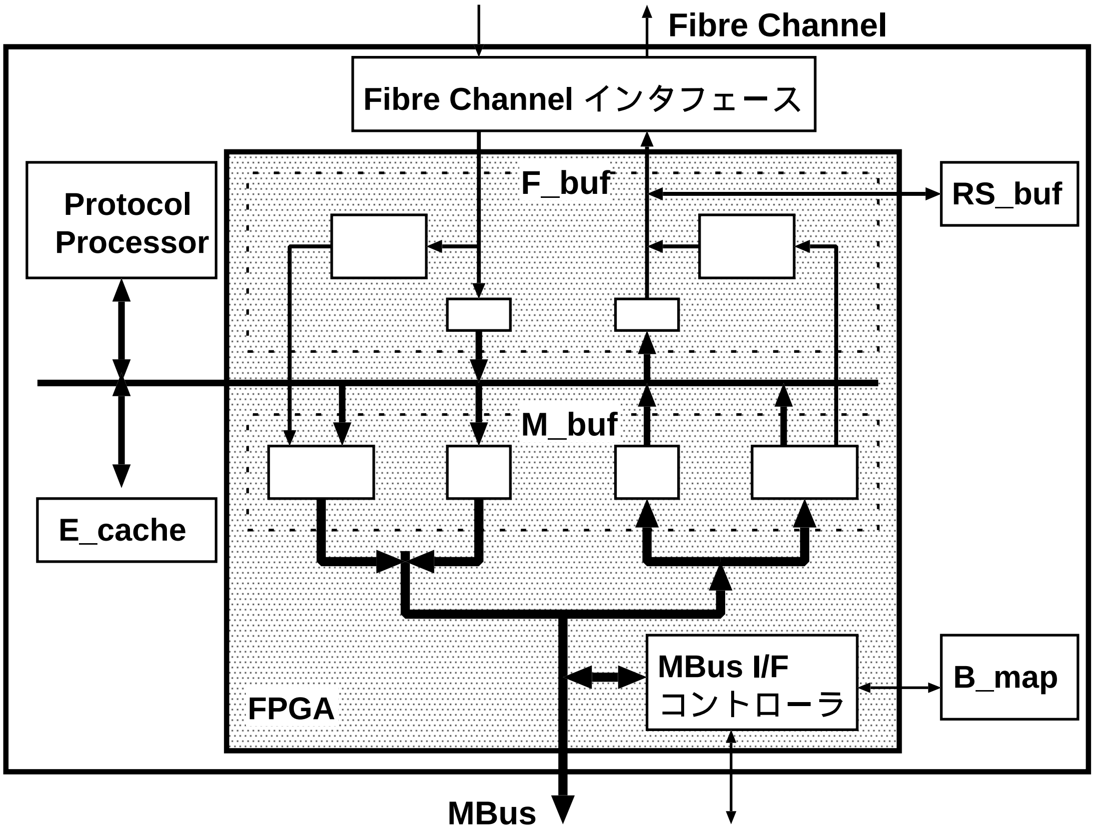

図 3: ネットワーク カードの構成

動作をサポートする．また，プロトコル プロセッサ上で動作するソフトウェアによるプログラムと，変更可能なFPGA(Field Programmable Gate Array)部と合わせて様々な通信方式を試すことができる．

- **外部キャッシュ(E_Cache)**
  ボード上のプロトコル プロセッサ用の2次キャッシュ．自ノードの主記憶の内容をキャッシングすることにより，自ノードプロセッサとデータを共有できる形にし，プロトコル プロセッサが巨大なメモリ空間を利用できるようにする．

4. **再送用バッファ(RS_buf)**．
   ハードウェアによるフロー制御を行うため，送信したデータを保存しておくバッファ．一部の例外を除いて，バッファの管理を含めたすべての処理をハードウェア上で実装する．

5. **Fibre Channel インタフェース**．
   Fibre Channel のネットワーク用インタフェース．

F_buf，M_buf，MBus インタフェース コントローラはFPGA内に実装する．

2.6 ネットワークカードの機能

前節で述べたネットワーク カードの各要素の役割について述べる．ネットワーク カードは次のような機能を持つ．

2.6.1 資源保護とネットワーク アドレス

仮想空間IDと仮想アドレスを連結したものをネットワーク アドレスとすることで，通信

時の資源保護をMMU，プロトコルプロセッサ，E_cacheを利用して以下のように自然に行うことができる(図4)．

1. ユーザの仮想アドレスでのアクセスをMMUがページテーブルを用いて物理アドレスに変換．
2. ネットワークインタフェースが物理アドレスを受取り，あらかじめE_cacheにキャッシングしておいた逆引きページテーブルを用いてプロトコルプロセッサが仮想アドレスに変換．仮想空間IDと連結しててネットワークアドレスを作成．
3. ネットワークアドレスを用いて，リモートアクセス．
4. 受信ノードのプロトコルプロセッサが，あらかじめE_cacheにキャッシングしておいたページテーブルを用いて物理アドレスに変換し受信側の物理メモリにアクセスする．

このように，OSによって作成されたこれらのテーブルを，参照に伴う保護違反チェック機能を利用することで資源を保護する．

2.6.2 ハードウェアによるフロー制御

Fibre Channel送受信用のバッファF_bufに加えてフロー制御のための再送バッファ(RS_buf)を用意し，受信完了の合図を出し合うことで，ハードウエアによるフロー制御を実現する．

2.6.3 MBusの信号を用いたキャッシュコヒーレンス制御

ネットワークカードはリモートメモリアクセスを行うだけではなく，ブロック単位(32Byte)でリモートメモリの内容のローカルメモリへのをキャッシングを支援する．以下では，自ノードの主記憶とリモートメモリとのコヒーレンス制御のために必要な機能と，その機能の実現のためにMBusインタフェースに要求されることを述べる．

MBusでは，それぞれの有効なデータに対してそれを配布する権限を持つキャッシュは唯一つであり，キャッシュのデータが一旦更新されれば他のプロセッサが持つ更新前のデータのコピーは全て無効化されるという規則がある．

よって，ネットワークカードでノード間でのコヒーレンスをとるためには，以下の4つの要件を満足しなければならない．

1. 自ノードのキャッシュ無効化
   自ノードの主記憶上の3次キャッシュに，リモートメモリのブロックをキャッシュしていた場合，キャッシュ元のブロックが更新されたときに自ノードのキャッシュを無効化する必要がある．

2. 自ノードの書き込みアドレスの検知
   自ノードプロセッサの2次キャッシュまたは主記憶上の3次キャッシュで書き込みがあった場合，そのコピーを無効化させる必要があるが，これの実現のためには，ネットワークカードが書き込まれたアドレスを検知し，無効化を行うノードに通知する必要がある．

3. ローカルメモリの無効なデータの読み込みの中断
   まだ自ノード主記憶ににキャッシングされていないブロック，あるいは，他ノードでの書き込み発生に伴って無効化されたブロックに対して自ノードのプロセッサが読み出そうとしている場合，その読み込みを中断させることができなければならない．そして有効なデータを持つノードからデータを転送させる必要がある．

4. キャッシュからの書き戻し要求への対応自ノードプロセッサの2次キャッシュのキャッシュラインの変更に伴い，自ノード主記憶へ書き戻しが行われた際に，他ノードに存在する当該ブロックのコピーに対して無効化を要求すると共に，当該ブロックの主記憶タグを有効なものにする．

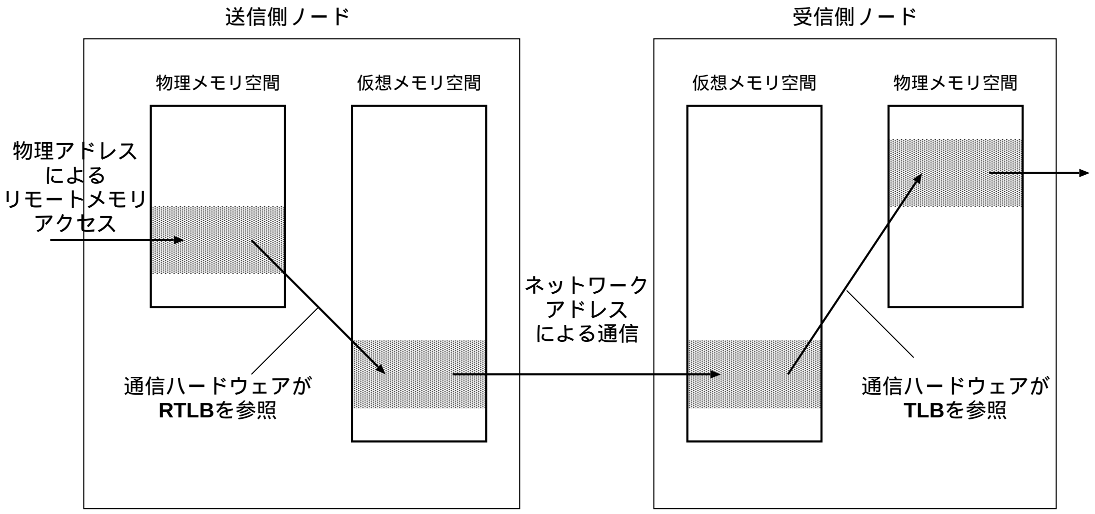

図 4: アドレス変換

3　ネットワーク　カード上のMBus インタフェース部の設計とその構成

本章では，ネットワーク カードのMBus インタフェース部に要求される機能と実現方法について述べる．MBus インタフェース部は，MBus に流れたメモリアクセスなどのトランザクションを，必要に応じてネットワーク内に取り込むMBus 送受信バッファとコントローラ，及び主記憶タグを正確に機能させ，それを参照し，必要に応じてキャッシュに対して共有，無効などの信号をバスに流すメモリ コントローラで構成される．

これらの部分は，ネットワーク カード内のFPGA 部に配置される．

3.1　要求仕様

ネットワーク カードの MBus インタフェース部は，MBus インタフェース仕様に適合したモジュールとして振る舞うことが要求されている．MBus インタフェース仕様は，2 種類の適合レベルが用意されており，ネットワーク カードは，キャッシュ コヒーレンシ，共有メモリのマルチプロセッサ システムを構築するための信号とトランザクションが定められているレベル2 の Mbus モジュールとして設計しなければならない．

MBus トランザクション中には，64 ビットにマルチプレクスされたアドレスとデータで構成されたメッセージという単位の流れで構成される．これらは MBus 中の幅 64 の信号線 MAD 上を流れる．またメッセージ中のアドレス及びデータをそれぞれアドレス フリット，データ フリットという．1 つのフリットはメッセージにおける 1 サイクルで流れる最小の構成要素である．

さらに，前章までに述べた高速通信の要件を満たすため，以下の要求を満足しなければならない．

- リモートメモリ アクセスの際の **MBus** トランザクションの受信
  MBus をスヌープすることにより，スレーブ モジュールとしてトランザクションを開始し，リモート アクセスの必要性を判定する．MBus の仕様よりこの判定はアドレス フリットが流れてから (このサイクルをアドレス フェーズといい A と記す)7 サイクル以内に行う必要がある．この判定は，数サイクルを要するのでアドレス フリットをいったんバッファに蓄積しておく必要がある
- 他ノードからの **MBus** トランザクション

の送信
他ノードから自ノード主記憶へのリモートアクセスのトランザクションをMBusへ送る．この時，ネットワーク カードは，MBusマスタ モジュールとして振る舞わねばならず，バスの使用権を獲得する必要がある．また，MBusの仕様より，MBusマスタモジュールはデータを最高速度で送受信できなければならない．そのためディレイなしのバースト転送に対応しなければならない．

- キャッシュのコヒーレンス制御
主記憶にリモートメモリの内容をキャッシングする．それらのキャッシュアクセスの際に主記憶タグを参照し，適切なコヒーレンス制御を行う．

3.2 構成

ネットワーク カードのMBusインタフェース部は，MBusトランザクションのメッセージを蓄積するMBus送受信バッファ部とMBusをスヌープし主記憶タグを参照してMBusに対してコヒーレント信号を流すMBusインタフェース コントローラ部の大きく2つで構成される．

- **MBus** 送受信バッファ部
MBus送信バッファ部とMBus受信バッファ部で構成されさらに各部は3つの部分からなる．
    1. バッファ部 (Mx_buf)
    トランザクションを収めるバッファ．アドレス フリットとデータ フリットを別に保持する．4つまでメッセージを貯めておくことができる．
    2. ディレクトリ部 (Mx_dir)
    Mx_bufの管理部．バッファの使用状況，メッセージの性質(返答か要求か)，トランザクションが収められた順番，次に使うべきバッファに関する情報を保持する．
    3. 制御部 (Mx_ctl)
    MBusに流れたトランザクションを受け取りバッファに収めたり，Mx_bufに入ってきたトランザクションを，MBusに流したりする．

MBus送受信バッファ部は，送信側と，受信側の2つを区別する際には，送信側にはS(Send)，受信側にはR(Receive)の文字を”x” と置き換え，MS_buf，MR_buf，MS_dir，MR_dir，MS_ctl，MR_ctlのように名付ける．

図5にMBus送信バッファ部の全体図を，図6にMBus受信バッファ部の全体図をそれぞれ示す．

- **MBus** インタフェース コントローラ部
MBusインタフェース コントローラ部は2つの部分からなる．
    1. MBusスヌープ部
    MBusトランザクションをスヌープし，MBus コヒーレンス制御信号MIH/MSH，及びMBusトランザクション制御信号MRTYを制御する．また，必要に応じてキャッシュの無効化のトランザクションを発生させる．スヌープの際は主記憶タグを操作するためにB_mapコントローラへ適切な操作を要求する．
    2. 主記憶タグ
    自ノード主記憶の状態を32バイト毎に表現したタグで，FPGA 外のSDRAM中に収める．有効，無効，共有の状態を表す．
    3. B_mapコントローラ
    SDRAM上に書き込まれるB_mapへの操作を行うためSDRAMに対する書き込み，読み出し，SDRAMの初期化を制御する．この部分は，共同研究者である杉野秀和氏により既に実装されている[6]．

MBusインタフェース コントローラ部の全体図を図7に示す．

以下それぞれの部分について詳細に述べていく．

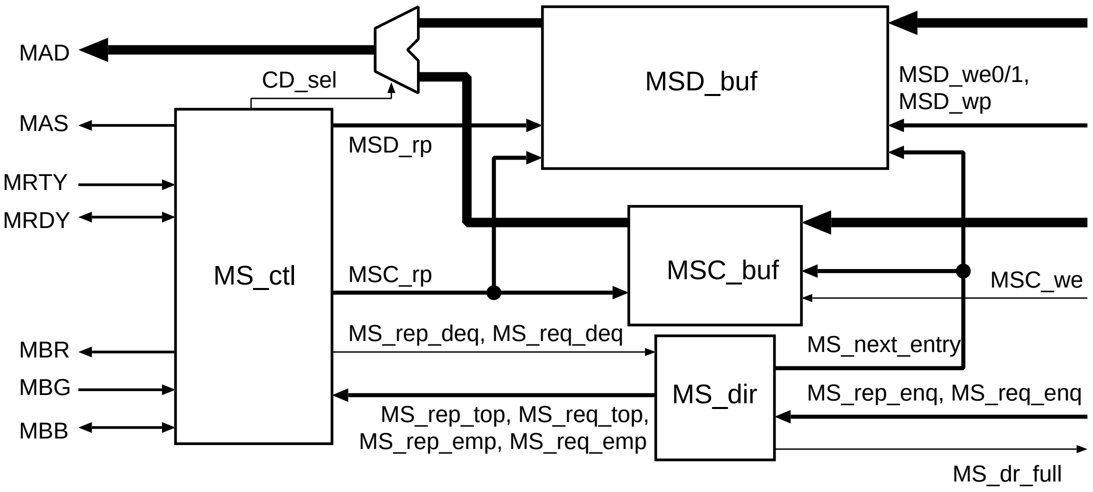

図 5: MBus 送信バッファ部

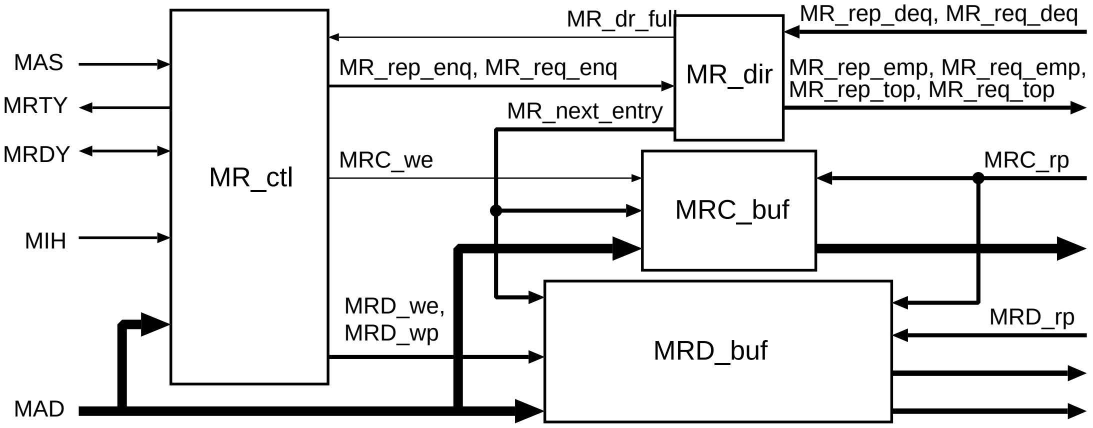

図 6: MBus 受信バッファ部

3.2.1 バッファ部 (Mx_buf)

Mx_buf は，FPGA 内部におかれる Mbus 用の送受信バッファである．

Mx_buf は，送信用の MS_buf と受信用の MR_buf からなる．MS_buf と MR_buf はそれぞれ，アドレス フリット用の MSC_buf と MRC_buf，データ フリット用の MSD_buf と MRD_buf からなる．MSC_buf と MRC_buf を合わせて MC_buf，MSD_buf と MRD_buf を合わせて MD_buf と呼ぶ．

MC_buf はレジスタファイル，MD_buf はデュアルポート RAM で構成されている．従って，どちらも書き込みはクロックの立ち上がりで同期しており，また，読み出しは非同期である．書込はクロックの立ち上がり時にライトイネーブルがアクティブであればそのとき流れている入力データを保持する．読み出しは，アドレスをいれた数 nsec 後に有効なデータを得ることができる．クロック周波数が 50MHz であるとき 1 サイクルは 20nsec であるので書き込み，読み出しともに 1 サイクルで行うことができる．

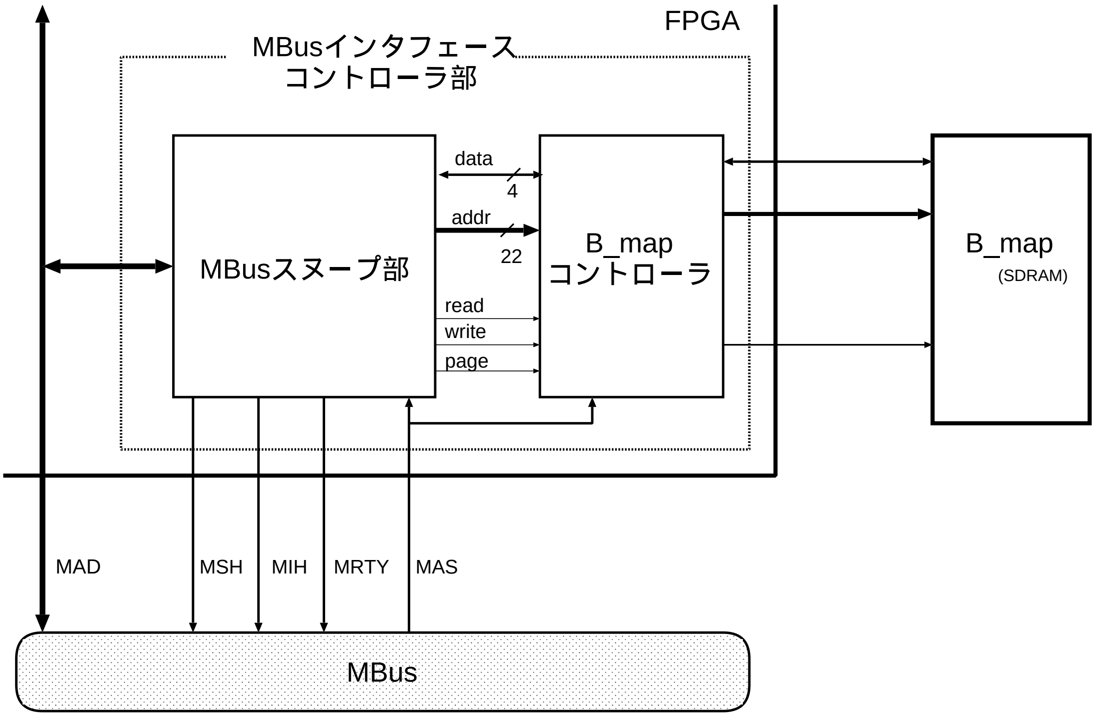

図 7: MBus インタフェース コントローラ部

3.2.2 送信バッファの部の動作

出力はMBusのMADに接続される．従ってMSC_buf，MSD_bufのどちらを接続するのか選択する必要がある．これにはバッファ選択用の内部信号CD_selを用いる．

3.2.3 受信バッファ部の動作

MRC_buf，MRD_buf，共にMBusのMADに接続される．どちらのバッファに収めるかはライト イネーブル信号によって決められる．

- 物理的構成．

MC_buf 通信ボードを介して行われるトランザクションのほとんどは，コヒーレンス制御を行う読み込み(CR:Coherent Read)，又は，コヒーレンス制御を行った後に無効化する(CRI:CoherentRead and Invalidate)である．このトランザクションは，データ サイズが32バイト(64ビット×4)で固定されている．また，本プロトタイプ ハードウェアでは，64バイト，128バイトのデータ転送をサポートしない．従って，アドレス フリット用バッファは，データ フリット用バッファに対してその容量が1/4でよく，また，より複雑なアクセスを必要とするので，多ポートのレジスタ ファイルとして構成する．

MD_buf データ フリット用バッファには，Xilinx FPGAのSelect RAMをデュアル ポートRAMの構成で使用する．Select RAMにおけるデュアル ポートRAMの最小構成は，16ワードの1ビットメモリなのでこれを64個並べて64ビット幅16ワードの構成とする．

図8，図9にMRD_buf，MSD_bufの構成を示す．

また，32ビット幅でのインタリーブ構成とし，データの上位32ビット(MAD[63:32])をbank0に下位32ビ

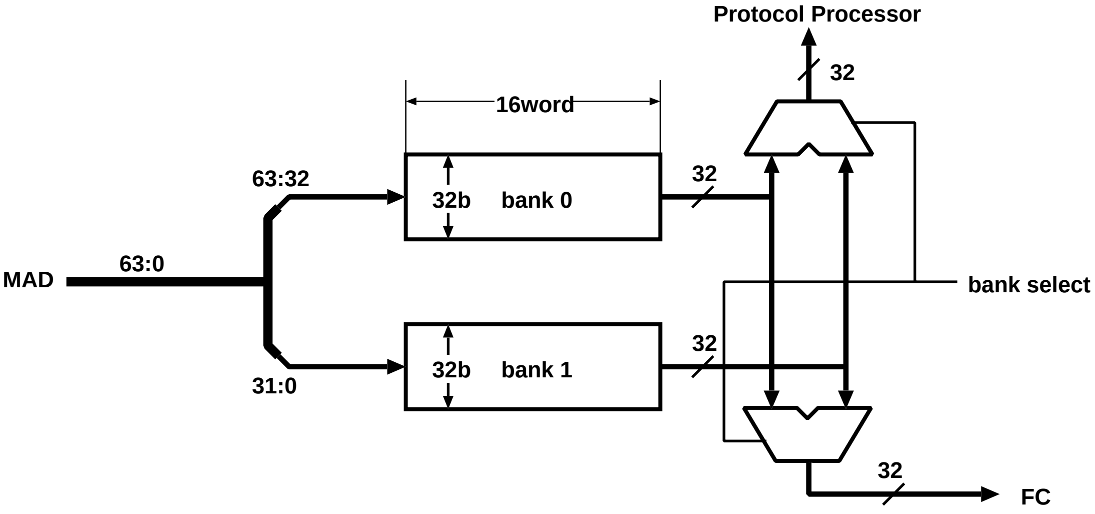

図 8: MRD_buf

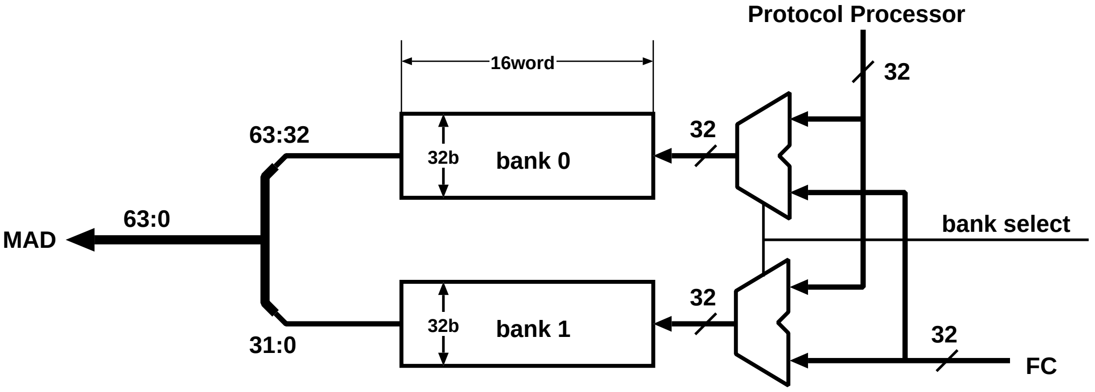

図 9: MSD_buf

ット(MAD[31:0])をbank1に格納する．これは，プロトコル プロセッサであるSPARCliteのバス幅が32ビットであるためであり，FPGAのピン数の節約もしている，また，FC側，SPARClite側が交互に使用することで，無駄なサイクルを低減できる．

- アクセス制御．

エントリ 1つのメッセージに対して割り当てられるバッファの単位をエントリと呼ぶ．1つのエントリは64ビットのレジスタ1つ(MC_buf)と64ビット4ワードのRAMを占有する．つまり一つのトランザクションあたりアドレス フリットが1つとデータ フリットが4つ分のバッファを割り当てることになる．

エントリの指定 MC_bufにおいて4つのレジスタファイルから1つのレジスタを選択するのに2ビット，MD_bufにおいて16ワードのRAMから1ワード選択するのに4ビット必要である．そこで，MC_bufの選択に使う2ビットをMD_bufの選択に使う4ビットのうちの上位2ビットにすることで，MC_bufとMD_bufのエントリをう

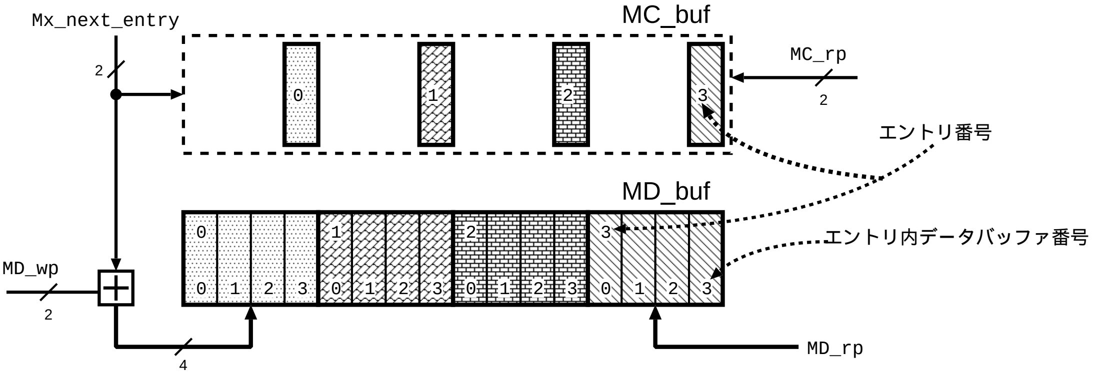

図 10: エントリの管理

まく対応させることができる．従って，エントリを指定する番号として，MC_buf のアドレスを用いることができる．ライトやリードのトランザクションでバーストされたデータ フリットをバッファに収める/取り出す際には，下位 2 ビットをカウンタで指定することで，データ フリットを順番に収める/取り出すことができる．

図 10にエントリの管理の様子を示す．

表??に MC_buf の，表??に MD_buf の信号表をそれぞれ示す．MBus 側ではバンクは無関係である．従って MRD_din は 64 ビットであるし，MRD_w_en は 1 本しかない．また，MSD_dout は 64 ビットである．

3.2.4 ディレクトリ部 (Mx_dir)

Mx_dir は Mx_buf を管理する部分である．Mx_dir は Mx_buf にメッセージを入れる側と出す側のコントローラにとって十分な機能を用意し，インタフェースを使いやすいものにする．

要求される仕様は以下の通りである．

- 空きエントリの管理．
  - 空きエントリが有る場合は次に使用すべきエントリを示す．
  - 空きエントリが無い場合はメッセージの受付を止める．
- デッドロック回避のためのメッセージの優先順位制御．
  - 返答メッセージと要求メッセージの 2 レベルでの制御．
  - 各レベル内では FIFO(First In First Out) 性を保証する．

メッセージの優先順位制御を行うため，トランザクションが発生した順序とそのメッセージが Fibre Channel に送出される順序が異なる．よって，単純なカウンタで，エントリを管理することはできない．

このような要求を満たすために，空きエントリを管理するディレクトリと返答/要求のメッセージが納められたエントリを保持する返答/要求キューで構成する．Mx_dir の全体図を図 11に示す．

Mx_dir の機能について説明する．

- 空きエントリの管理
  エントリがすべて使用中である場合は MxC_full をアクティブにし，メッセージを入れる側に通知する．空きがある場合には Mx_next_entry で次に使用するべきエントリをキューとバッファに指示する．返答メッセージ/要求メッセージを入れる信号である Mx_rep_enq / Mx_req_enq の OR をとった Mx_dir_s がアサートされると，そのとき Mx_next_entory で指していたエントリを使用中にする．同時に MC_full，Mx_next_entry を更新する．

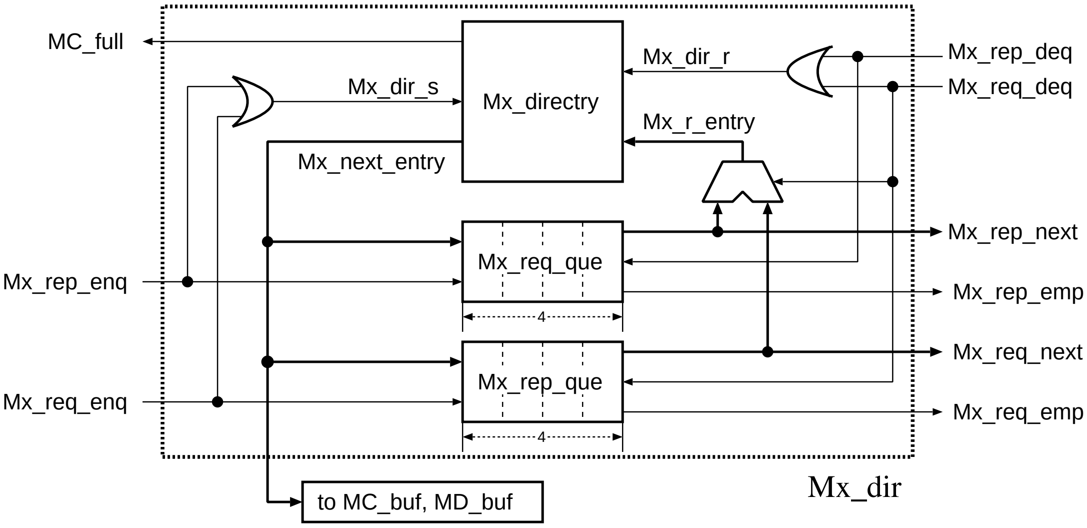

図 11: Mx_dir の構成

返答メッセージ/要求メッセージを取り出す信号である Mx_rep_deq / Mx_req_deq がアサートされるとセレクタによってそれぞれのキューの先頭が選択され Mx_r_entry に入れられる．同時に Mx_dir_r がアサートされ Mx_r_entry で指されたエントリが空き状態になる．

- 優先順位制御
  それぞれのキューはエントリ番号を入れる幅である幅 2 ビット深さ 4 のシフト レジスタである．これで返答/要求キュー内での FIFO 性は保証される．キュー内には最後尾の次の場所を指すポインタがあり，次にエントリ番号を入れる場所を示している．

  返答/要求キューの先頭のエントリ番号は Mx_rep_next / Mx_req_next から取り出す側へ伝えられる．Mx_rep_enq / Mx_req_enq がアサートされると Mx_next_entry で指されたエントリ番号を返答/要求キューの最後尾に次に収める．Mx_rep_deq / Mx_req_deq がアサートされるとすべてのデータは先頭側へ 1 つシフトする．

図 12に Mx_dir の動作例を示す．図中では Mx_req_enq をアサートし，次に Mx_rep_deq をアサートしたときに，どのように動作するかを示したものである．点線の円で書かれた部分がアサートによって更新された部分である．

enq と deq が同時にアサートされた場合も問題なく動作する．図 12で説明すると，仮に最も上の図の状態で，Mx_req_enq と Mx_rep_enq が同時にアサートされたとしても最も下の図の状態に移る．

図 11の各信号とその役割をまとめて，付録の表??に示す．

3.2.5 制御部 (Mx_ctl)

Mx_ctl はトランザクションの送受信を制御する．送信側コントローラ MS_ctl と受信側コントローラ MR_ctl がある．表??に MR_ctl に，表??に MS_ctl にそれぞれ接続された信号を示す．

本節では構成のみ示し，その機能及び動作は次章で示す．

- **MS_ctl**
  MS_ctl は状態制御部，バス獲得制御部，信号出力部バースト転送制御部からなる．

  バスのアービトレーションのための信号には，バス獲得要求信号 MBR，アービターからの許可信号 MBG，バスの使用中を示す信号 MBB の 3 種があり，これらはすべ

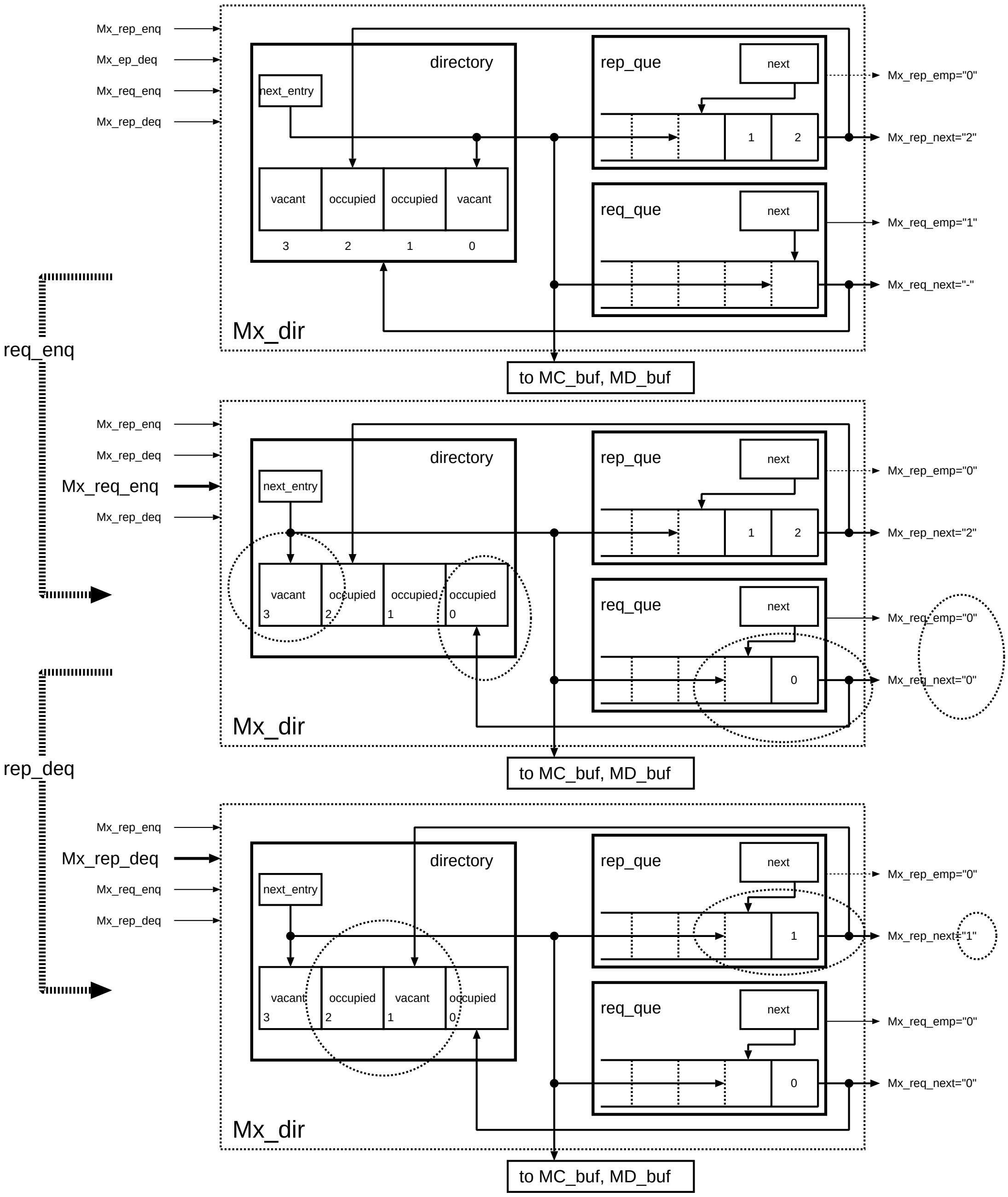

図 12: Mx_dir の動作例

てアクティブローである．

- **MR_ctl**
  MR_ctl はデコード部，状態制御部，信号出力部，バースト転送制御部からなる．

3.2.6 MBus スヌープ部

MBus スヌープ部はスヌープしたメモリアクセスに応じて，B_map の読み出しを B_map コントローラへ指示し，読み出されたデータに基づいて MIH/MSH をアサートする．

表??に B_map コントローラとのインタフェースを示す．

3.2.7 B_map コントローラ

B_map コントローラの内部構造は詳しくは述べないが，PAGE_READ，PAGE_BANK についてのみ説明する．

- **PAGE_READY**
  主記憶のページアウトに対応して，B_map もページ単位タグの書き込みができる．この信号はページライトが可能であるかを示すものである．
- **PAGE_BANK**
  ページライト中のバンクを示す．同じバンクにリード/ライト要求が来た場合すぐに実行できない．

4 MBus インタフェース部の動作

本章では，ネットワークカードの一部である MBus インタフェース部の基本的な動作について詳細に述べる．MBus に発生するキャッシュコヒーレンス制御を行うトランザクションには，以下のようなものがある．

Coherent Read(CR) キャッシュコヒーレンシを維持する読み込み．無効なデータへのアクセスがあると，有効なデータを持つキャッシュはアクセスを中断させ，有効なデータを送る．

Coherent Write and Invalidate(CWI)
コピーの無効化．当該ブロックの有効なコピーを持つキャッシュはデータを無効化する．

Coherent Read and Invalidate(CRI)
キャッシュコヒーレンシを維持する読み込みと無効化．上記 CR と CI を同時に行う．

Coherent Write and Invalidate(CWI)
データの書き込みとコピーの無効化．データを書き込んだ後，当該ブロックの有効なコピーを持つキャッシュはデータを無効化する．

以下，CR，CI，CRI，CWI と略記する．

4.1 送信バッファ制御部の動作

送信側コントローラは MBus に対して，トランザクションを流す必要がある．このとき，ネットワークカードは MBus マスタモジュールとして働くため，バスの獲得をバスアービターに対して要請しなければいけない．従って，そのための専用の制御機構が必要である．

バス獲得制御部の状態遷移図を図 13に示す．

送信の必要が発生したときにネットワークカードは MBR をアサートしバスをアービターに対し要求する．アービトレーションの結果，許可が与えられれば MBG がアサートされ，次にバスを使用する権利が与えられる．もし MBB がインアクティブであればすぐに MBB をアサートしバスを使用することができるが，アクティブである場合はインアクティブになるまで待つ．

図 14にバスがグラントされていない状態からのバス獲得及び書き込みの様子を示す．

それまで空だったエントリに要求メッセージが入り，MS_req_emp がインアクティブになる．この時 MS_req_top からは要求キューの先頭のエントリが出される．そこで MBus トランザクションを発生させるためバスを獲得する．この例では最も短いサイクル数で獲得している．MBB をアサートすると同時に MAS をアサートしトランザクションを開始する．

MAS はマスタがアサートしアサートされたサイクルに MAD にアドレスフリットが流れてい

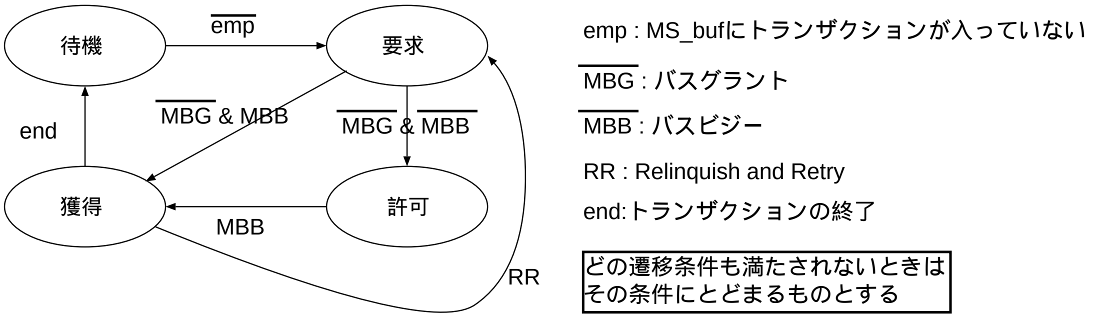

図 13: バス獲得のための状態機械

ることを示す．MAD が MSC_buf と MSD_buf のどちらをソースとするかはMAD_sel で選びハイのときは MSC_buf が選択されローのときは MSD_buf が選択される．ライト トランザクションにおいてスレーブの MRDY アサートはマスターに対する次のデータの要求を意味し最後の MRDY アサートはトランザクションの終了を意味する．最後の MRDY がアサートされたのを確認して MS_req_deq をアサートし，キューから取り出す．

バスを獲得する条件はアービターからの許可 (MBG) が降り，かつ，バスがどのマスターにも占有されていないことである．

4.2 受信バッファ制御部の動作

MR_ctl は MBus 受信バッファコントローラである．

ネットワーク カードは，OS を介さず，自発的にリモートメモリ アクセスを行えなければならない．MR_ctl はリモートメモリ アクセスの必要性を判断し，開始する．

リモートメモリアクセスを行わなければならない場合は以下の場合である．またサイクル A というのはアドレス フリットが流れたサイクルを意味する．

1 については，サイクル A+2 で MIH/MSH がアサートされる場合があるが，これはプロセッサのキャッシュによるものなので無視する．2 の場合は，無効化をを当該ブロックを共有しているノードへ通知しなければならない．3 は物理アドレスで判断でき，ネットワークカードのみがアクノリッジを返す．

1 の場合について MBus 上での信号の様子を図 15に示す．MBus の仕様で MIH をアサートしたモジュールは MIH アサートから 4 サイクル以降にデータを送り出さなければならない．図ではリモートメモリアクセスに時間がかかることを考慮して途中を省略している．

3 の場合の MBus 上での信号の様子を図 16 に示す．

4.3 MBus スヌープ部

MBus スヌープ部の動作は以下の通りである．

1. バスにトランザクションが流れるのを検出する．
2. リードの命令と物理アドレスを B_map コントローラに送り B_map を参照させる．
3. B_map からタグを受け取りそれにしたがって MIH/MSH をアサートする．

1 の時点で，以下の場合はトランザクションのリトライを要求するため，MRTY をアサートする．

- MBus アドレスのうちのページバンク番号に対応するビットと PAGE_BANK とが同一でかつ PAGE_READY がインアクティブである場合
- リード、ライト要求するときに ACT_READY がインアクティ

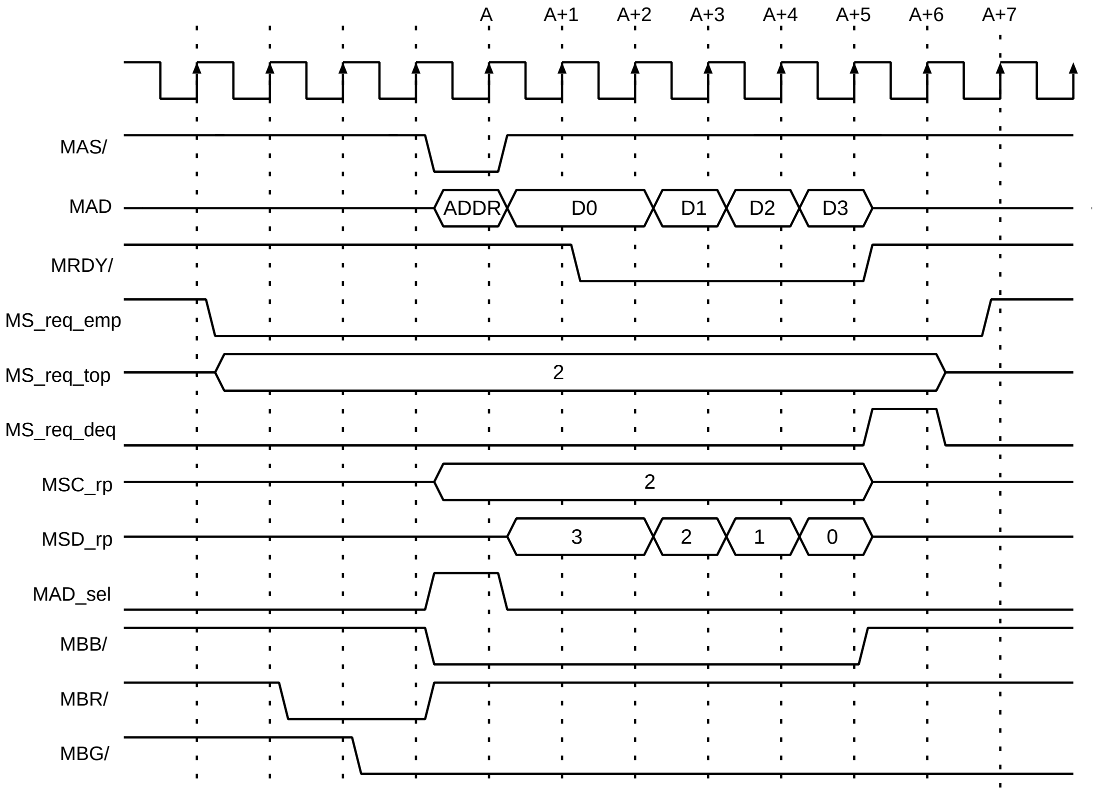

図 14: バスの獲得

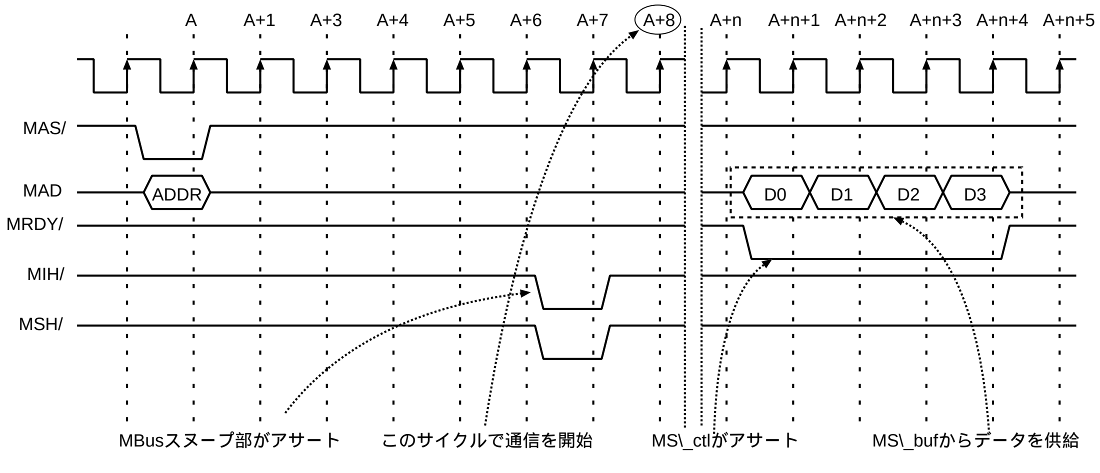

図 15: 無効な主記憶に対するアクセスからのリモートメモリアクセスの起動

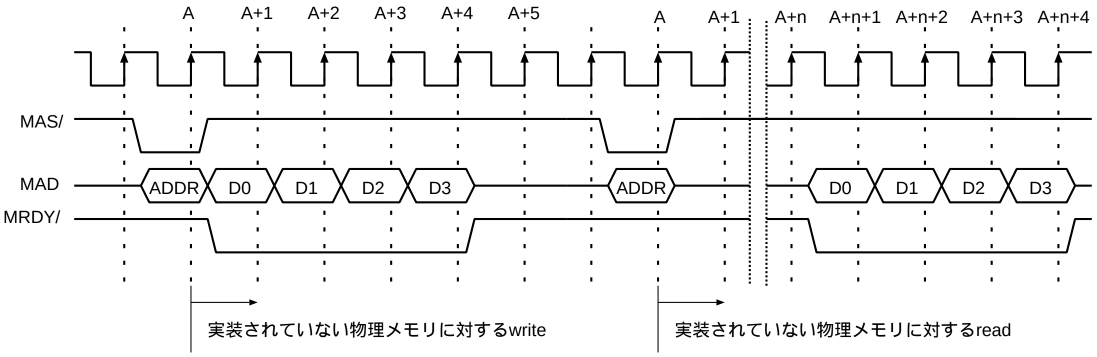

図 16: 実装されていない主記憶へのアクセスからのリモートアクセスの起動

ブである場合

5　おわりに

本稿では，並列応用を指向した分散システム「コンピュータ コロニー」を提案し，そのプロトタイプ ハードウェアのネットワーク インタフェースであるネットワーク カードについて述べ，さらにネットワーク カードのバス インタフェース部の設計，実装について述べた．

「コンピュータ コロニー」は従来の分散システムの問題を解決し，従来の分散システムでは困難であった細粒度の並列処理により，ネットワーク結合された計算資源の有効利用を目指すものである．

本稿では，プロトタイプ ハードウェアのネットワーク カードのシステム バスインタフェースの設計と実装について述べてきた．また，この部分の実現により，リモートメモリのローカルメモリへのキャッシングが可能になりそのコヒーレンス制御を，ハードウェア上で実行できること，及び通信の必要性を判定し，自発的な通信が可能になることについて述べた．

今後は，ネットワーク カードの詳細な設計，プロトコル プロセッサ部のプログラミング，専用OS 「コロニア」の実装などを進めることが必要である．

謝辞

本研究の機会を与えてくださった富田眞治教授に深甚の謝意を表します．

また，本研究に関して適切なご指導を賜った森眞一郎助教授，五島正裕助手に深く感謝致します．

最後に，本研究の共同研究者である 伊達新哉氏，Damien Le Moal 氏，増田峰義氏，杉野秀和氏，鶴崎剛大氏をはじめとして，日頃様々な角度から助力して下さった京都大学大学院情報学研究科通信情報システム専攻富田研究室の諸兄に心より感謝致します。

参考文献

[1] Ben Catanzaro : SPARC マルチプロセッサアーキテクチャ(1995).

[2] A. S. Tanenbaum :OS の基礎と応用 (1995)．

[3] SPARC Technology Business: SuperSPARC & MultiCache-Controller User's Manual,(1994).

[4] 伊達新哉 :コンピュータ ·コロニーを実現する高速通信機構, 信学技報 CPSY99-52, pp.41-48．(1999)．

[5] 山添博史 :分散オペレーティングシステム Colonia の共有メモリ管理とそのエミュレ

ータ, 特別研究報告書, 京都大学工学部情報工学科 (1997).

[6] 杉野秀和 :並列アプリケーションを指向した分散システムの通信ハードウェアにおけるメモリコントローラ, 特別研究報告書, 京都大学工学部情報学科 (1999)．