計算機アーキテクチャ 144－7
（2001．7．25）

RHINET の概要と Martini の設計/実装

山本淳二[†] 渡邊幸之介[††] 土屋潤一郎[††]
今城英樹[†††] 寺川博昭[†††] 西宏章[†]
田邊昇[†,☆] 工藤知宏[†] 天野英晴[††]

ユーザレベル・ゼロコピー通信を用いることで低レイテンシ・高バンド幅な通信を提供するネットワークRHiNETとそのネットワークプロセッサMartiniについて述べる．MartiniはRHiNETのネットワークインタフェースカードの核となるネットワークプロセッサである．Martiniでは基本のRDMA機能を全てハードワイヤードで処理することで，RHiNETの持つ低レイテンシ・高バンド幅を最大限利用する．その一方で内部にプロセッサを持つことで複雑な処理についても柔軟に対処できる構成となっている．

An Introduction of RHiNET and its Network Processor Martini

JUNJI YAMAMOTO,[†] KOUNOSUKE WATANABE,[††] JUN'ICHIRO TSUCHIYA,[††]
HIDEKI IMASHIRO,[†††] HIROAKI TERAKAWA,[†††] HIROAKI NISHI,[†]
NOBORU TANABE,[†,☆] TOMOHIRO KUDOH[†] and HIDEHARU AMANO[††]

In this paper, RHiNET and its network processor Martini are described. RHiNET provides low latency, high bandwidth, user level zero-processor-copy communication.

Martini is a network processor used as core of RHiNET's network interface card. While Martini processes fundamental RDMA and PIO-based transfer functions by hard-wired logic, it is provided with an on-core processor to support flexible complex functions.

1. はじめに

近年のパーソナルコンピュータ（PC）の性能向上はめざましい．またPCをノードとしたクラスタシステムを用いた並列処理も広く行われるようになっている．このようなクラスタではノード間をMyrinet[1]のようなシステムエリアネットワーク（SAN）で結合することが多い．

SANはパケットを途中で破棄せず，送信順に到着するネットワークであり，低遅延・高バンド幅の通信を提供する．その一方で最大リンク長やネットワークトポロジーに対する制限が厳しいため，クラスタなど一カ所に設置されたシステムに用いられてきた．

一方，フロア内やビル内などある程度広範囲に分散している計算機間はイーサネットに代表されるローカルエリアネットワーク（LAN）で接続されている．一般にLANは敷設可能な距離が100mから1km程度と長く，また取りうるネットワークのトポロジーの制限も緩い．しかし，パケットの破棄・重複を認めているため，TCPなど上位層によるパケットの順序制御・再送制御が必要がある．

現在は高性能な並列分散処理はSANで接続されたクラスタで行われているが，SANに匹敵する性能を持ち，LANと同様に広範囲に分散した計算機を接続できるネットワークがあれば，通常業務に使用してる計算機の余剰性能を利用したり，複数のクラスタ群を接続することでさらなる高性能な計算機資源を作り出すことが可能である．

そこで我々はこのようなLANとSANの双方の利点を持つ，新しいネットワーククラスであるローカルエリアシステムネットワーク（LASN）が提案している[2]．

LASNはLANと同様にネットワークのトポロジーを比較的自由に決められ，また接続距離も100mから1kmまで延ばせると同時にSANと同じくパケットの破棄は行わず，送信順に届くという長所を持つネットワーククラスである．

LASNの一つの実装としてRHiNET[3),4)]がある．本

† 新情報処理開発機構
Real World Computing Partnership
†† 慶應義塾大学
Keio University
††† 日立インフォメーションテクノロジー
Hitachi IT co.,ltd.
☆ 現在，東芝
Presently with Toshiba co.,ltd.

稿ではこのRHiNETの概要と，RHiNETで用いるネットワークインタフェースの核となるネットワークプロセッサMartini[5]の設計と実装について述べる．

2. RHiNET

広く使われているTCP/IPを使用する通信はシステムコールを利用する．TCP/IPでは信頼性の低い物理層を使用しても高信頼な通信を実現するため，多くの努力を払っている．

RHiNETネットワークはネットワークインタフェースとスイッチ，それらを接続する光インタコネクションから構成される．RHiNETではソフトウェアから見た物理層が高信頼となるようハードウェアを設計する．そのため，TCP/IPでの信頼性確保のために行われる処理は不要である．

また，RHiNETではユーザレベル・ゼロコピー通信による低遅延・高バンド幅の通信をサポートする．

ユーザレベル通信はユーザプロセスが直接ネットワークインタフェース（NI）に指示を与えることである．また，ゼロコピー通信はユーザプロセス上にあるデータをNIが直接アクセスしてネットワークへ送信したり，ネットワークから受信したデータを直接ユーザプロセス上の受信領域へ書き込むことである．

このユーザレベル通信とゼロコピー通信を併用することで，通常の通信で必要なシステムコールやユーザ空間とカーネル空間の間で行われるコピーなどを省けるため，低レイテンシな通信が可能である．

ユーザレベル・ゼロコピー通信では，ユーザプロセスが送信データや受信バッファのアドレスを直接NIに伝え，NIはデータやバッファに対してDMAでアクセスする．ユーザプロセスが使用するアドレスは全て仮想アドレスであるが，一般にDMA時に必要なアドレスは実アドレスである．そのため，NIは仮想アドレスから実アドレスへ変換する機構を備える必要がある．

また，プロセス間の不要な干渉を防ぐために通信の保護機構が必要である．特にユーザレベル通信では通信に関わる情報がユーザプロセスから直接NIに渡されるため，NI自身がその情報の正当性を判断する必要がある．

RHiNETの構成要素であるスイッチは4種類の実装・設計が行われている．現在，設計実装中のRHiNET用ネットワークプロセッサではこのうち3種類のスイッチ，RHiNET-2/SW, RHiNET-3/SW, OIP SWと接続が可能である．

RHiNET-2/SW[2]は8Gbpsのリンクを持つgo-and-stopフロー制御を行うスイッチであり，RHiNET-3/SW[6]は10Gbpsのリンクを持つcreditベースのフロー制御を行うスイッチであり，OIP SWは2Gbpsのリンクを持つgo-and-stopフロー制御を行うスイッチである．

次章からはRHiNETのネットワークインタフェースの核となるMartiniについて述べる．

3. Martiniの機能

MartiniはRHiNETのネットワークインタフェースの核をなすLSIである．

MartiniはPCIとDIMMの2種類のホストインタフェースを持つ．PCIをホストインタフェースとするネットワークインタフェースカード（NIC）をRHiNET/NIと呼び，DIMMをホストインタフェースとするNICをDIMMnet-1と呼ぶ．

また，MartiniはRHiNET-2/SW, RHiNET-3/SW, OIP SWの3種類のスイッチに接続できるポートを持つ．

Martiniでは並列処理をサポートするためリモートDMA（RDMA）を基本とした機能（プリミティブ）と，PIOによる送信機構を提供する．

3.1 保護機構

3.1.1 Window

Martiniはユーザプロセスからの要求を受け付けるwindowと呼ぶ領域を持つ．ユーザプロセスは，windowに通信に必要な情報を書き，window上の特定のアドレス（キックアドレス）に書き込む（キックする）ことによりMartiniに通信を要求する．

Martiniは複数のプロセスが通信を行うことを考慮し，複数のwindowを提供している．個々のwindowはホストプロセッサのページ単位に配置されている。ユーザプロセスは通信に先だってシステムソフトウェアを呼び出し，呼び出されたシステムソフトウェアがwindowをユーザプロセスのアドレス空間にマップする．

プロセスは自らのアドレス空間にマップされていないwindowにはアクセスできないので、ホストプロセッサの仮想記憶機構によりアクセス権の管理がおこなわれることになる。

3.1.2 PGID

RHiNETによる通信を行うプロセスにはProcess Group ID（PGID）とProcess ID（PID）の2つのIDが割り振られる．PGIDは相互に通信するプロセス群につけられるシステムグローバルなIDで，PIDは同一PGIDを持つプロセス群内でそのプロセスを特定するユニークなIDである．

システムソフトウェアがwindowをユーザ空間にマップする際に，Martini内部のテーブル（PGIDTBL）にwindowの番号（WINID）と，ユーザプロセスのProcess Group ID（PGID），Process ID（PID）の対応を登録する．

PGIDはユーザから操作できないIDであるため，Martiniではこれを保護機構のキーとして用いる．

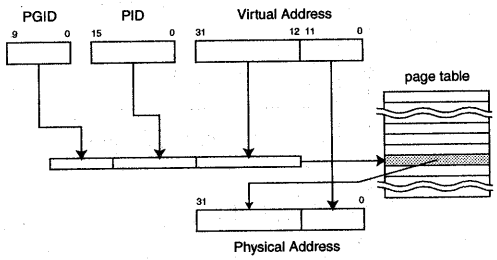

図 1　アドレス変換機構（送信側）
Fig. 1　Address conversion (sender side)

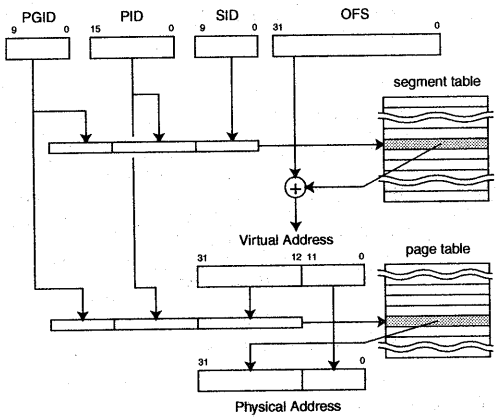

図 2　アドレス変換機構（受信側）
Fig. 2　Address conversion (receiver side)

具体的には次の述べるアドレス変換機構のキーの一部として PGID を用い，PGID が異なるプロセスのアドレスは指定できないようにすることで，関連のないプロセスが不当に干渉することを防ぐ．

3.2　アドレス変換機構

RDMA の実行にはユーザプロセスから与えられた仮想アドレスを実アドレスに変換する機構が必要なため，Martini はアドレス変換をサポートする TLB（PATLB）を備えている（図 1）．

さらにセグメント ID（SID）と呼ぶ ID を提供する．受信側のアドレスはこの SID とオフセット（ROFS）の組み合わせで指定できる．SID とそのセグメントの先頭アドレスの対応はユーザプロセスが自由に決められる．Martini は SID から仮想アドレス（RVA）への変換をサポートする TLB（RVATLB）を備えている（図 2）．

3.3　RDMA ベースプリミティブ

Martini で最も基本となるプリミティブは PUSH と PULL で，それぞれリモートメモリ書込み，リモートメモリ読出しに対応する（図 3）．この 2 つのプリミティブはハードウェアだけで処理できるよう設計されている．

PUSH/PULL の実行に必要な情報を表 1 に示す．ユーザプロセスはこれらの情報を window に書き込み，最後にキックアドレスする．Martini はキックされたことを検出し，プリミティブの実行を開始する．

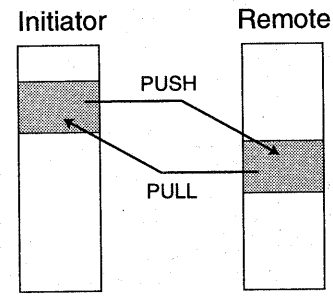

図 3　PUSH / PULL 動作概要
Fig. 3　Operation of PUSH / PULL

表 1　PUSH/PULL で必要な情報
Table 1　Information for PUSH/PULL

変数名	内 容
RPID	相手プロセス番号
IVA	自プロセス側アドレス
RSID	相手プロセスのセグメント番号
ROFS	セグメントに対するオフセット
SIZE	転送データサイズ
STATUS	結果フラグの格納アドレス
RRID	相手ノードまでのルーティング情報
IRID	自ノードまでのルーティング情報

表 1 中，STATUS は PUSH/PULL が終了したことを Martini が通知する時に使用する変数の仮想アドレスである．PUSH/PULL は Martini が処理するため，ユーザプロセスはバッファをいつ使用してよいかが判断できない．そのため，Martini は STATUS で示された変数へ結果を書き込むことでユーザプロセスにプリミティブの終了を通知する．

RRID と IRID はユーザプロセスから与えるが，Martini に誤ったルーティング情報を与えた場合でも，パケットが届いた先のノードに目的とするプロセスがなければ保護機構によってエラーとなり問題は生じない．

3.4　PIO による送信機構

大量なデータ転送では DMA を利用した PUSH / PULL が有効であるが，少量のデータ転送ではアドレス変換や DMA のセットアップタイムが無視できないため，PIO によるデータの送信機構も提供している．PIO 送信機構にはある程度の大きさのデータを送ることができる Block On-The-Fly（BOTF）と，1 ワードアクセスで送信が可能な Atomic On-The-Fly（AOTF）の 2 種類ある．

3.4.1　BOTF

BOTF はユーザプロセスがパケットを作成し，送信する機構である．ユーザプロセスは window 上にヘッダを含めたパケットの全てを構築する．最後に window

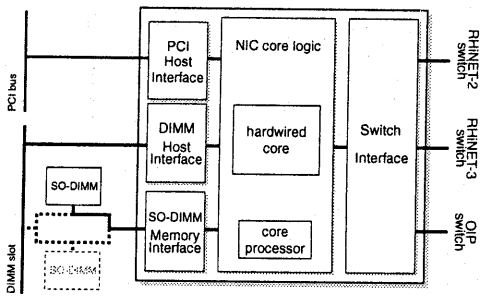

図 4 Martini ブロック図
Fig. 4 Block diagram of Martini

上の指定されたアドレス（キックアドレス）にライトアクセスを行うことで Martini に送信開始を指示する．

Martini はプリミティブと BOTF を区別するため，それぞれのキックアドレスは異なっている．

パケットヘッダには PGID のフィールドがあるが，PGID は保護機構の鍵なので，Martini は送信時に WINID から得られる PGID に置き換える．

3.4.2 AOTF

AOTF はホストからのライトアクセスによってパケットを送信する機構である．ユーザプロセスは window とは別に用意された AOTF 用の領域に書き込みアクセスを行う．Martini はこのアクセスを検出し，あらかじめ Martini 内に用意されているパケットヘッダに書き込まれたデータを追加することでパケットを作成し，送信する．

パケットヘッダはあらかじめシステムソフトウェアにより Martini 内に設定される．ユーザプロセスからは設定できず，不正なアクセスの心配がないため，PGID を含む全ての情報がそのまま送信される．

3.5 コアプロセッサ

PUSH/PULL のプリミティブおよび BOTF, AOTF はハードウェアコントロール部が処理する．処理中に発生する各種 TLB でのミスヒットなどの例外事項や PUSH/PULL 以外のプリミティブの処理を行うため，プロセッサを内蔵している．

4. Martini の構造

Martini は大きく 5 つのブロックからなる（図 4）．

PCI ホストインタフェース部と DIMM ホストインタフェース部はそれぞれ PCI バス，メモリバスに接続され，ホストからのアクセスを Martini 内部に伝える．メモリインタフェース部は Martini に接続される SO-DIMM を制御する．コア部は Martini の中核となるブロックで，ハードワイヤード処理部とコアプロセッサからなる．スイッチインタフェース部は RHiNET-2/SW, RHiNET-3/SW および OIP スイッチのプロトコルをサポートする．

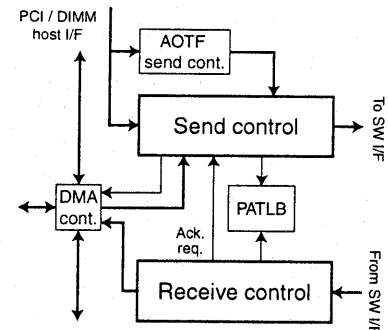

図 5 ハードワイヤ処理部ブロック図
Fig. 5 Block diagram of hardwired core

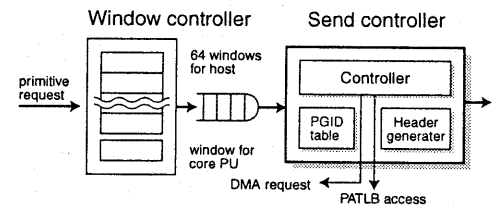

図 6 送信部ブロック図
Fig. 6 Block diagram of send controller

4.1 ハードワイヤー処理部

図 5 にハードワイヤー処理部のブロック図を示す．

ハードワイヤー部は送信部と受信部に分かれており，送信処理と受信処理を並列して行うことができる．

4.1.1 送 信 部

図 6 に送信部の構造を示す．

PUSH プリミティブの処理を例に送信部を説明する．

(1) ユーザプロセスは自分に割り当てられた window にプリミティブ起動に必要な情報を書き込む．

(2) 送信部は window の特定アドレス（キックアドレス）への書き込みを契機に，その window の WINID と書き込まれた情報をキューにコピーする．

(3) PGIDTBL を使用してキューから取り出した WINID から対応する PGID と PID を得る．

(4) 次に PATLB を使用して IVA を実アドレス（PVA）に変換する．

(5) キューから得た情報と PGID, PID を元にパケットヘッダを作成し，送信バッファに送り出す．

(6) 最後に DMA コントローラに対してホスト上のデータのアドレス（PVA）とデータ長を指定してデータを取り出す．DMA で読み出したデータは先のヘッダに連結され送信バッファを通じてスイッチインタフェースに送り出される．

データが複数のページにまたがっている場合は，ページ単位にパケットを分割し，(4) から処理を繰り返す．

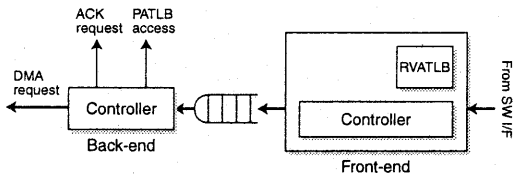

図 7　受信部ブロック図
Fig. 7　Block diagram of receive controller

PGIDTBL のエントリが無効の場合や PATLB にミスヒットした場合は，そこで処理を中断し，コアプロセッサに割り込む．コアプロセッサ上のソフトウェアは割込を受け取ると エラー処理や TLB のリフィル処理を行い，処理終了後，送信部の動作を再開する．

4.1.2 受　信　部

図 7 に受信部の構造を示す．

PUSH プリミティブから生成されたパケット（PUSH パケット）の処理を例に受信部の動作を説明する．

(1) スイッチインタフェースを通して受信したパケットはフロントエンド部でヘッダが解釈される．RVATLB を使用して ヘッダに含まれる PGID, RPID と SID からセグメントの先頭アドレス（RVAbase）を得る．これに ROFS を加え，受信側プロセスの仮想アドレス RVA を求める．

(2) PUSH パケットのようにヘッダに続き DMA すべきデータを持つパケットはフロントエンドに引き続きバックエンドの処理が開始される．

(3) バックエンドは PATLB を用いて RVA から実アドレス（RPA）を求める．次に DMA コントローラを起動し，受信データの DMA を開始する．

(4) さらにバックエンドはヘッダにある送信側アドレスの下位 3 ビットと，受信側アドレスの下位 3 ビットを元に受信データをシフトしつつ DMA コントローラにデータを渡す．
送信側と同様に複数ページにわたる DMA が必要な場合は（3）から処理を繰り返す．

(5) 全ての DMA が終了した時点で，送信側に ACK パケットを送るよう要求する．
要求を受けた送信部はプリミティブを発行したノードに ACK パケットを返送する．

ACK パケットを受け取ったノードは，ACK パケットのヘッダの情報に従い，終了フラグを書き込む．この終了フラグを受け取る変数のアドレスは window の STATUS フィールドに書き込まれ，ユーザプロセスはその変数をポーリングすることで PUSH プリミティブの終了を知る．

受信部も送信部と同様に RVATLB にミスヒットしたり，ハードワイヤード処理できないパケットを受信した場合には処理を中断し，コアプロセッサに割り込む．

4.1.3 代行処理機構

Martini は PUSH と PULL の 2 つのプリミティブに関してはハードワイヤード処理部だけで処理が行えるよう設計されている．しかし，それ以外の lock, barrier, send, receive などの複雑なプリミティブや各種 TLB でのミスヒットなどの例外事項が発生した場合には，コアプロセッサに割り込み，処理を任せる．

この代行処理機構により，基本性能を落とさずに柔軟な処理を可能としている．

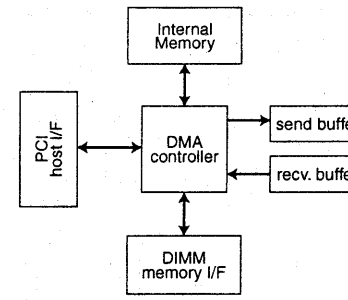

図 8　DMA 制御部
Fig. 8　DMA controller

4.1.4 DMA 制御部

DMA 制御部は Martini 内部で使用される DMA の制御を行う（図 8）．DMA の対象は，ハードワイヤード処理部（送信バッファ，受信バッファ），PCI ホスト I/F 部，メモリ I/F 部，内部メモリ（SRAM）である．このうちの任意の 2 つの間での DMA が可能である．さらに，重ならない 2 つの DMA は同時に実行できるので，例えば，PCI ホスト I/F 部（ホストメモリ）とハードワイヤード処理部内の送信バッファ間で DMA をしつつ，SRAM と メモリ I/F 部（SO-DIMM）間での DMA を実行することが可能である．

4.1.5 AOTF 送信部

ホストが AOTF 用領域に書き込むと，AOTF 送信部が処理を開始する．AOTF 送信部は書き込まれたアドレスのページ番号からパケットヘッダの種（ヘッダシード）を求める（図 9）．AOTF 送信部はヘッダシードに書き込みアドレスのオフセット部，アクセスサイズを追加し，パケットヘッダを完成させる．そして，パケットヘッダと書き込まれたデータから構成されるパケットを，送信バッファを通してスイッチインタフェース部に引き渡す．

4.2 コアプロセッサ部

コアプロセッサ部は，R3000 をベースとした 32bit プロセッサと，256Kbyte のメモリ，割込コントローラから構成される．主にハードワイヤー処理部で処理しきれない複雑な処理を受け持つ．

4.3 スイッチインタフェース部

Martini は RHiNET-2/SW, RHiNET-3/SW, OIP SW の 3 種のスイッチと接続できるインタフェースを持つ．それぞれのスイッチは異なるパケットフォーマットを用いる．

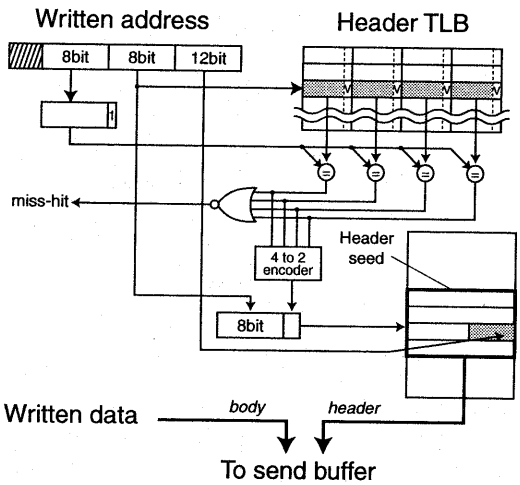

図 9　AOTF 送信部構造

Fig. 9　Structure of AOTF send controller

表 2　Martini の諸元

Table 2　Specifications of Martini

項 目	
デザインルール	0.14μm
ダイサイズ	272.91mm^2
メモリ総量	538Kbyte
I/O 伝送周波数	
RHiNET-2/SW	800MHz
RHiNET-3/SW	800MHz
OIP SW.	250MHz
内部動作周波数	
コア部	66MHz
DIMM ホスト I/F	133MHz
スイッチ I/F	125MHz
パッケージ	784BGA

スイッチインタフェースはその 3 種のパケットフォーマットと，コントロール部で使用するパケットフォーマットの相互変換を受け持つ．また，各スイッチの仕様に合わせたフローコントロールを行う．

5. Martini の諸元

表 2 に Martini の諸元を示す．**表 3** に各部のゲート数一覧を示す

6. おわりに

RHiNET の概念と，Martini の設計と内部構成について述べた．

RHiNET はユーザレベル・ゼロコピー通信により低レイテンシ・高バンド幅な通信を提供する．RHiNET のネットワークインタフェースの核となる Martini は，最も基本となるプリミティブ PUSH, PULL を全てハードワイヤードで実装しているため，高速な処理が期待できる．その一方で，内部にプロセッサを持ち，複雑な機能や例外処理などに対しても柔軟な対応が可能な構成となっている．

表 3　Martini の各部のゲート数

Table 3　Estimation of gate size of Martini

ブロック名	ゲート数
ハードワイヤード処理部	935K
スイッチ I/F 部	299K
DMA 制御部	36K
SRAM I/F 部	14K
コアプロセッサ	175K
その他	66K
PCI ホスト I/F	272K
DIMM I/F	366K
合計	2,163K

参 考 文 献

1) Myricom, Inc.: *http://www.myri.com/*.

2) 西宏章, 多昌廣治, 西村信治, 山本淳二, 工藤知宏, 天野英晴: LASN 用 8 Gbps/port 8*times*8 One-chip スイッチ: RHiNET-2/SW, *JSPP2000*, pp. 173–180 (2000).

3) Kudoh, T., Nishimura, S., Yamamoto, J., Nishi, H., Tatebe, O. and Amano, H.: RHiNET: A network for high performance parallel processing using locally distributed computers, *IWIA 99* (1999).

4) Kudoh, T., Tanabe, N., Yamamoto, J. and Nishi, H.: RHiNET: A network for high performance parallel computing using locally distributed computers, *2000 RWC Symposium*, Real World Computing Partnership, pp. 5–10 (2000).

5) 山本淳二, 田辺昇, 西宏章, 土屋潤一郎, 渡辺幸之介, 今城英樹, 上島利明, 金野英俊, 寺川博昭, 慶光院利映, 工藤知宏, 天野英晴: 高速性と柔軟性を併せ持つネットワークインタフェース用ネットワークインタフェースチップ: Martini, 2000-ARC-140, pp. 19–24 (2000).

6) 西宏章, 上野龍一郎, 多昌廣治, 稲沢悟, 西村信治, 工藤知宏, 天野英晴: LASN 用 10Gbps/port 8x8 ネットワークスイッチ:RHiNET-3/SW, デザインガイア 2000, pp. 13–18 (2000).