社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－ARC－177 (30)
2008－HPC－114 (30)
2008／3／6

ExpEtherにおけるRDMA通信機構の実装

内山幸憲[†] 今田啓介[†] 辻聡[†]
大塚智宏[†] 鈴木順[††] 樋口淳一[††]
飛鷹洋一[††] 天野英晴[†]

NECの開発したExpEtherは、Ethernetと PCI Expressを統合するネットワークインタフェースである。本稿では、ExpEtherを対象とした、RDMA通信機構を備えるネットワークインタフェースコントローラの設計と実装を行った。設計においては、RDMA通信の標準化に向けて仕様が定められているiWARPを参考とした。RTLシミュレーションを用いたコントローラ単体での性能評価において、データ送信の最大スループットは881MBytes/secに、データ受信の最大スループットは795MBytes/secとなった。また、通信遅延の測定では、STag情報の参照や物理アドレスの取得がRDMA通信における通信遅延の増加の主な要因となることが判明した。

Implementation of the RDMA Communication Mechanism on ExpEther

Yukinori Uchiyama,[†] Keisuke Imada,[†] Akira Tsuji,[†] Tomohiro Otsuka,[†]
Jun Suzuki,[††] Junichi Higuchi,[††] Youichi Hidaka[††] and Hideharu Amano[†]

ExpEther by NEC is a network interface for a bridge between PCI Express and Ethernet for network connected virtual computer environment. This paper focuses on design and implementation of the RDMA communication mechanism on ExpEther network interface card. The design is based on iWARP specification which is intended to be a standard of RDMA communication mechanism. Evaluation using RTL simulation shows the network interface controller we designed achieves 881MBytes/sec of throughput on sending, and 795MBytes/sec of throughput on receiving. It also revealed that the reference of STag infomation and retrieval of physical addresses occupy a large part of total communication latency.

1. はじめに

近年、PCやサーバ、ストレージ、ルータなどを用いたシステムが様々な用途で活用されるようになってきた。結果、システムを構成するプラットフォームの形態は多様化の一途を辿っており、それらの接続に用いられるネットワークの規格も様々なものが混在するようになった。それに伴い、システムの管理運用コストが増加するという問題が浮上している。そして、これを解決するために仮想化技術が注目されるようになってきた。

日本電気株式会社(NEC)[1]が開発しているExpEther[2]は、LANの標準規格であるEthernetとPCの内部バスとして広く普及しているPCI Expressを統合する機能により、ローカルのPCI Expressのトランザクションを Ethernetを介してリモートに転送する機能を持つ。これにより、コンピュータのPCI Expressバス接続はEthernetネットワークへと拡張される。そして、コンピュータは通常の装置内部バスに接続されているI/Oにアクセスするのと同様の手段で、実際はネットワークに接続されているI/Oを参照することが可能となる。このようにI/Oの仮想化を行うことにより、複数のシステムまたはコンピュータによるI/Oデバイスへのアクセス、または共有を容易に実現することができる。更に、ソフトウェアレイヤの仮想化技術が行う処理を、ハードウェアレイヤの仮想化技術に置き換えることで、システムの性能を向上させることができる。しかし、現在のExpEtherの持つI/O仮想化の機能だけでは、単一のコンピュータと複数のI/Oから成るシステム構成に限られてしまう(図1左)。

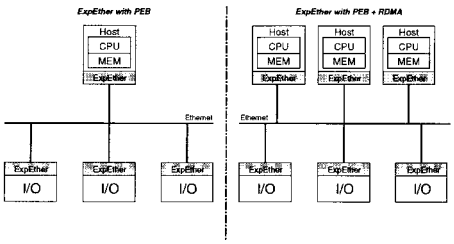

図1 ExpEtherによるシステム構成

そこで、本稿ではRemote Direct Memory Access(RDMA)通信機構を備えるネットワークインタフェースコントローラを設計し、ExpEtherへの実装を行った。このコントローラのRDMA通信機構の設計には、RDMAの標準化に向けてまとめられたiWARP[3]の仕様を採り入れている。RDMA通信により、コンピュータのメモリ間を高速に接続することが可能となり、複数のCPU、メモリ、I/Oから成るシス

† 慶應義塾大学
Keio University
†† 日本電気株式会社
NEC Corporation

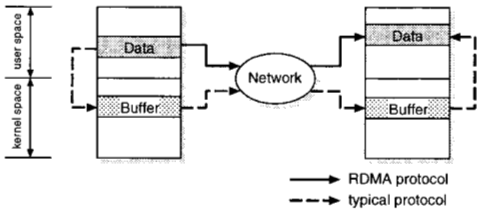

図 2　一般的な通信と RDMA 通信のデータの流れ

テム構成を実現することができる (図 1 右)。また、仮想化技術と協調動作させることにより、実際にはネットワークで接続されている複数のコンピュータが、ネットワークの存在を意識することなく互いのメモリを参照することができるようになる。ExpEther ではこれらの技術の実現により、多様化するシステムを統合し、一元管理することを目標としている。

以下、2 節において RDMA 通信について述べ、3 節にて RDMA 通信機構の標準化に向けてまとめられた iWARP の概要に触れる。また、4 節で実装対象である ExpEther の概要を述べる。5 節では、iWARP による RDMA 通信機構を採り入れたネットワークインタフェースコントローラの設計について説明し、6 節にてその実装に触れる。7 節では、実装を行ったネットワークインタフェースコントローラの RTL レベルシミュレーションにおける評価を示す。最後に 8 節でまとめる。

2. RDMA 通信

一般に、ネットワークを介して通信する場合、ユーザプロセスと Network Interface Card(NIC) 間で通信データの受渡しをする際に、通信データをカーネル空間に用意されたバッファに一時的にコピーしなければならない。このメモリ間のデータコピーは PIO で行われるため、転送速度は低く、通信オーバヘッドを大きくする要因の一つとなる。この様子を表したのが図 2 の破線で示されるデータの流れである。

一方、RDMA 通信はこれと異なり、通信データをホストメモリ上のバッファに一時的にコピーすることなく、直接ユーザプロセスと NIC 間でのデータの受渡しを実現する。これは、あらかじめ通信データが置かれたメモリ領域とデータを受信するメモリ領域を各々NIC に登録しておき、通信が実行されると NIC がメモリとネットワークの間で DMA 転送を行うことで実現される。この様子を表したのが図 2 の実線で示されるデータの流れである。

RDMA 通信ではこのようにして一時的なバッファコピーを排除し、通信レイテンシを改善することができる。また、データコピーによるホストプロセッササイクルやメモリバンド幅の消費も生じないため、ホストプロセッサに与える負荷の影響も極めて低くなる。

3. iWARP

iWARP は RDMA 通信機構の標準化に向けてまとめられた、RDMA プロトコルを含む RDMA 通信の一連のアーキテクチャである。その仕様は RDMA コンソーシアム[3] によって策定されている。

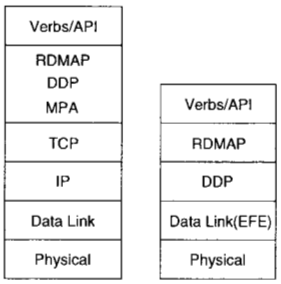

図 3　iWARP プロトコルスタック (左) と実装を行うプロトコルスタック (右)

iWARP で定義されるのは、主に RDMA プロトコルとしての Direct Data-placement Protocol(DDP)[4] と RDMA Protocol(RDMAP)[5]、加えて RDMA 通信のためのインタフェースの振舞いを記述する Verbs[6] である (図 3)。

iWARP におけるユーザアプリケーションからの通信の起動は、Work Request(WR) と呼ばれる処理要求を Queue Pair(QP) に投入することで行われる。QP には Send Queue(SQ) と Receive Queue(RQ) の 2 種類の要求受付けのリソースが含まれ、主に送信要求が SQ に、受信要求が RQ に発行され処理される。また、QP に受理された WR は、Work Queue Element(WQE) として内部表現される。

Send/Receive モデルによる通信では、データの送信側となるデータソースでは送信データのメモリ領域を示す Steering Tag(STag) という識別子を含めた WR を SQ に投入することでデータ通信が開始される。これに対し、データの受信側となるデータシンクでは受信データを格納するメモリ領域を示す STag を含めた WR を RQ に投入していなければならない。一方、RDMA によるデータ通信の場合は、データソースもしくはデータシンクとなる側のみが送信データの存在するメモリ領域と受信データを格納するメモリ領域のそれぞれを示す STag を含んだ WR を SQ に投入することで通信が開始される。この時、データシンクでは RDMA によるデータ受信を意識することは無い。また、SQ、RQ の WR が完了すると、その情報が Completion Queue Element(CQE) として生成され、必要に応じて Completion Queue(CQ) に投入される。ユーザアプリケーションは CQE の情報により WR の完了状況を知ることができる。

4. ExpEther

本節では、ExpEther の持つ機能について述べる。まず、PCI Express と Ethernet を統合する PCI Express-to-Ethernet Bridge(PEB) の概要に触れる。続いて、Ether Forwarding Engine(EFE)[7] の概要に触れる。

4.1 PEB

PEB は PCI Express のトランザクションレイヤプロトコルと Ethernet の Media Access Control(MAC) 層のブリッジ

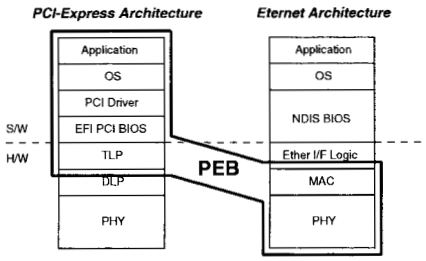

図 4　PEB 概要

を行う。この概要を図 4 に示す。これは PCI Express のトランザクションレイヤパケットを Ethernet フレームでカプセリングすることにより行われる。ExpEther は PEB のこの機能により、PCI Express によるバス接続を Ethernet ネットワークへと拡張する。

4.2　EFE

EFE は MAC 層において end-to-end の輻輳制御と再送制御を行い、パケットロスのないネットワークを構築する。

また、EFE はデータセンタ内やクラスタ内のサーバ間通信などといった伝搬遅延時間の非常に短い環境において、データ転送遅延を短くすることのできる輻輳制御方式を持つ。この輻輳制御方式は遅延ベースの方式であり、プロービングと予測 Round Trip Time(RTT) を用いることでスループットの立上りを早くし、スイッチでのキューイング遅延を削減している。

5. 設　　計

本節では、ExpEther に実装を行う RDMA 通信機構を備えたネットワークインタフェースコントローラの設計について述べる。このネットワークインタフェースコントローラは iWARP の仕様を採り入れた設計となっている。実装を行うプロトコルスタックと iWARP で定められたプロトコルスタックの比較を図 3 に示す。

本章で述べる設計は ExpEther 開発ボードを対象としており、全てのロジックは ExpEther 開発ボード上の Altera 社[8] の FPGA である StratixII GX に実装する。

5.1　ExpEther ネットワークインタフェースコントローラの概要

本ネットワークインタフェースコントローラは iWARP で定義される通信オペレーションのうち、主に次のものをサポートする。

- RDMA Write
- RDMA Read

ExpEther ネットワークインタフェースコントローラは処理に汎用プロセッサを用いず、すべての処理をハードワイヤードで実現する。図 5 は、ExpEther ネットワークインタフェースコントローラのブロック図である。コントローラは大別して 3 つのブロックにより構成される。

- **PCI Express Interface**
  PCI Express Interface として、Inventure[9] 製の IP コアを用いる。
- **EFE**
  ネットワーク側とのインタフェースとして、ExpEther の機能の 1 つである EFE を用いる。
- **RNIC Core**
  RDMA 通信をはじめとする様々な処理を取りまとめる、ネットワークインタフェースコントローラの制御部である。

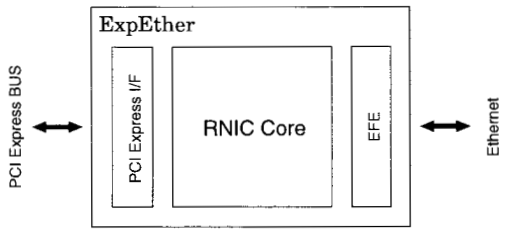

図 5　コントローラのブロック図

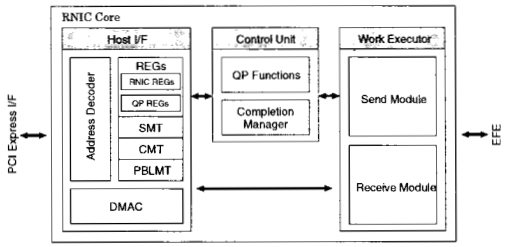

図 6　RNIC Core のブロック図

5.2　RNIC Core の設計

図 6 に RNIC Core のブロック図を示す。RNIC Core はホストとデータのやり取りを行う Host Interface 部と、RDMA 通信などの様々な処理の制御を行う Control Unit 部、データの送受信を実行する Work Executor 部により構成される。

5.2.1　Host Interface

Host Interface はホストが RNIC Core の設定や要求発行をするためのレジスタ群、加えて通信に利用する情報を格納するための RAM によって構成される。また、PCI Express Interface とデータをやり取りするための Address Decoder と DMA Controller(DMAC) を含む。図 6 で示されるモジュールの内、次のものが Host Interface に含まれる。

- **Registers(REGs)**
  RNIC Core や QP Function の設定を行うための設定レジスタ群や、WR を受け付けるための要求発行レジスタが含まれる。
  - **RNIC REGs**
    ホストから RNIC Core の起動や動作設定を行うためのレジスタ群。
  - **QP REGs**
    QP Function の設定を行うためのレジスタ群に加えて、WR を受け付けるための SQ や RQ の窓口となる要求発行レジスタで構成される。

- **STag Management Table(SMT)**
  STag に関連するパラメータなどの情報を格納するための RAM。
- **Completion Management Table(CMT)**
  CQ に関連するパラメータなどの情報を格納するための RAM。
- **Physical Buffer List Management Table(PBLMT)**
  PBL と呼ばれる物理アドレスのリストを格納するための RAM。
- **Address Decoder**
  PCI Express Interface を通じて行われるホスト主導のデータ転送の処理を行う。
- **DMAC**
  RNIC Core 主導のデータ転送の処理を行う。

5.2.2 Control Unit

Control Unit は QP Function と Completion Manager により構成され、Host Interface と Work Executor の間に入り様々な制御を行う。

- **QP Function**
  QP Function は iWARP で定義される QP と同等の役割を担うモジュールである。QP Function はホストからの WR を受理し、内部表現である WQE の形式に変換して、要求処理部である Work Executor へと処理の要求を出す。また、WQE 処理の順序制御や RDMA Read オペレーションの制御、Completion Queue Element(CQE) の生成など、iWARP による通信の要となるモジュールである。QP Function はユーザが通信を行う際の要求発行の対象となるため、複数の QP Function を用いることで同時に複数の通信を行うことができる。QP Function の処理対象は SQ、RQ の他に、RDMA Read 処理を制御する Outbound RDMA-Read Request Queue(ORRQ)、Inbound RDMA-Read Request Queue(IRRQ) である。QP Function の動作設定や SQ、RQ への WR の受理は Host Interface 部の QP REGs で行われる。
- **Completion Manager**
  Completion Manager は QP Function で生成された Completion Queue Element(CQE) の CQ への配置を行う。CQ への配置は、QP Function に設定されるパラメータにより CMT テーブルを参照し、物理アドレスを得ることで行われる。この物理アドレスを用いて Host Interface 部の DMAC により DMA によるデータ転送が行われる。

5.2.3 Work Executor

Work Executor はデータの送信に関連する処理を行う Send Module と、データの受信に関連する処理を行う Receive Module により構成される。

- **Send Module**
  Send Module は Control Unit 部の QP Function から WQE の処理要求を受け取り、ネットワークへのデータ送信や、送受信バッファの登録処理などを実行する。データ送信の際、WQE と Host Interface 部が持つ情報により直接ホストメモリから送信データを取得することができる。ホストメモリからの送信データの読み込みには、Host Interface 部の DMAC を使用し、DMA によるデータ転送が行われる。
- **Receive Module**
  ネットワークから受信した DDP セグメントをそのヘッダ情報に従い処理する。ペイロードは直接ホストのメモリに配置されるか、Host Interface 部のバッファに格納される。ホストメモリへのデータ書き込みには、Host Interface 部の DMAC を使用し、DMA によるデータ転送が行われる。

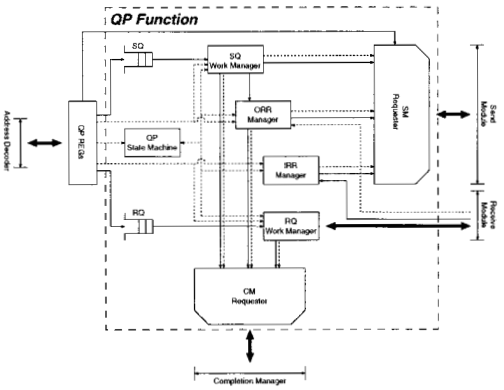

図 7 QP Function の構造

6. 実　　装

本節では、5 節で示した ExpEther ネットワークインタフェースコントローラのうち、RNIC Core の Control Unit 部の QP Function、及び Work Executor 部の Send Module と Receive Module の実装について触れる。

6.1 QP Function

図 7 に QP Function の構成を示す。QP REGs は Host Interface 部のモジュールであるが、QP Function の動作において重要な情報を保持するため併記した。

SQ、RQ、IRRQ、ORRQ は、それぞれのキューの WQE の処理を制御するコントローラを備えている。また、QP Function の状態を管理するステートマシンを持つ。

6.2 Send Module

Send Module は各 QP Function から WQE の処理要求を受け付けることで、WQE に従ったオペレーションを実行する。また、WQE のオペレーションが完了するとその完了状態に対応した完了ステータスを WQE の処理要求を出した QP Function に返す。

図 8 に Send Module の構成を示す。Send Module は QP Function の他、SMT や PBLMT、DMAC といった外部モジュールと接続される。

6.2.1 Work Acceptor

Work Acceptor は複数の QP Function からの WQE 処理要求を順番で受け入れるアービタである。この Work Acceptor に受け入れられた QP Function からの WQE が Send

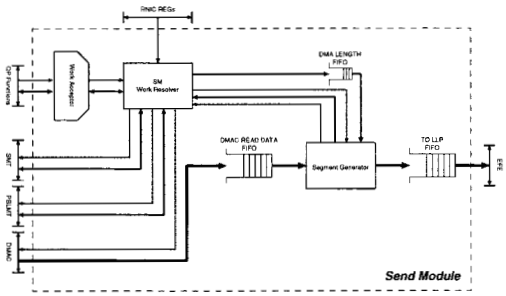

図 8 Send Module のブロック図

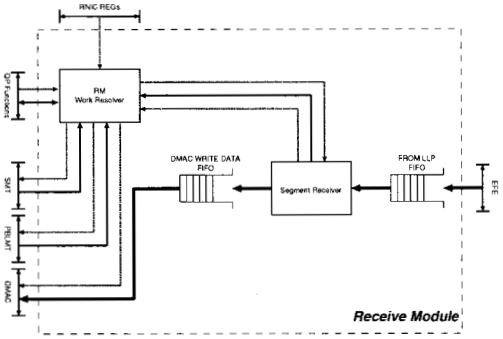

図 9 Receive Module のブロック図

Module で処理されることになる。Work Acceptor が受け入れた WQE は SM Work Resolver へと渡され実行される。オペレーションが完了すると SM Work Resolver から完了ステータスが通知され、WQE 処理要求を出した QP Function に WQE 処理の完了と完了ステータスの通知を行う。

6.2.2 SM Work Resolver

SM Work Resolver は Work Acceptor が受け入れた WQE に従いオペレーションを実行する。主な処理は、オペレーションの正当性の検証や、DMAC へのホストメモリからの送信データ読み込みの要求である。DMAC への読み込み要求に用いる送信バッファの物理アドレスは、WQE が持つ STag を用いて SMT や PBLMT を参照することで得ることができる。また、DMAC に要求した DMA 転送長を Segment Generator に通知するために、DMA LENGTH FIFO に格納する。

6.2.3 Segment Generator

Segment Generator は DDP セグメントの生成を行い、EFE とのインタフェースになる TO LLP FIFO へ DDP セグメントの書き込みを行う。DDP セグメントの生成は、DDP ヘッダの作成と、DMAC がホストメモリから読み込んだ送信データを DMAC READ DATA FIFO から取得し、これを DDP セグメントのペイロードとすることで行われる。また、DMAC READ DATA FIFO からの読み込みは DMA LENGTH FIFO から取得する DMA 長に即して行われる。基本的にセグメンテーションは EFE の最大ペイロード長である 1472Byte に則して行われるが、DMA 長の境界においても行われる。これは、DMAC READ DATA FIFO のデータ幅が 64bit であり、かつ DMA 長が 64bit アラインとは限らず、DMA データの最後のデータの有効ビットの保存に必要なためである。

6.3 Receive Module

Receive Module は DDP セグメントを EFE から受け取り、DDP セグメントに応じた処理を実行する。

図 9 に Receive Module の構成を示す。Receive Module は QP Function の他、SMT や PBLMT、DMAC といった外部モジュールと接続される。

6.3.1 RM Work Resolver

RM Work Resolver における主な処理は、Segment Receiver が受信した DDP セグメントのヘッダ情報の解析によるセグメントの正当性の検証である。加えて、DDP セグメントが RDMA Write によるものであれば、DMAC にホストメモリへの書き込み要求を行う。

6.3.2 Segment Receiver

Segment Receiver における主な処理は、EFE からの DDP セグメントの受信に加え、DDP ヘッダの RM Work Resolver への通知、ペイロードの DMAC WRITE DATA FIFO への転送である。DDP セグメントの受信は、EFE とのインタフェースである FROM LLP FIFO を読み込むことで行う。

7. 評　　価

本節では、実装を行った RNIC Core の基本通信性能を示す。評価項目として、送信と受信のスループット、及びオペレーションの通信遅延を測定した。

7.1 評価環境

評価測定は Verilog-HDL を用いた RTL シミュレーションで行い、PCI Express Interface の動作周波数である 125MHz と同等の動作周波数で動作するものとした。また、DMAC がホストから読み込むデータは、Send Module が読み込み要求を出した直後に DMAC READ DATA FIFO から取得可能であるとした。

7.2 スループット

スループットの測定は、RDMA Write オペレーションの転送データ長を変化させて行った。送信時の値の算出は SQ に RDMA Write オペレーションの WQE が投入された直後から、全ての転送データが TO LLP FIFO へ書き込まれるまでの時間から導いた。受信時の値の算出は FROM LLP FIFO から DDP セグメントの先頭が読み込まれてから、全ての転送データが DMAC WRITE DATA FIFO へ書き込まれるまでの時間から導いた。

送信と受信のスループットの測定結果を図 10 に示す。結果より、送信時は転送データ長が 4KBytes を超えるあたりから最大スループットに漸近することが分かる。最大スループットはおよそ 881MBytes/sec となった。一方、受信時は転送データ長が 1KBytes を超えるあたりから最大スループットに漸近することが分かる。転送データ長 2KBytes 付近のスループットが一旦落ち込んでいるが、これは転送データ長が EFE の最大データユニット長である 1472Bytes を超えると DDP メッセージが分割され、DDP セグメントの検証

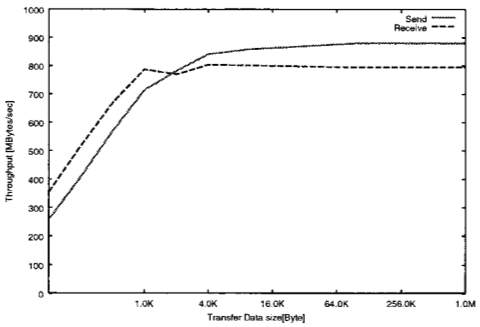

図 10 スループット

表 1 通信遅延

オペレーション	通信遅延 [ns]
RDMA Write	352
RDMA Read	168

が複数回行われるようになることによる遅延のためである。最大スループットはおよそ 795MBytes/sec となった。受信のスループットが送信に比べて低くなっているのは、送信時の STag 検証が DDP メッセージにつき 1 度で済むのに対し、受信時は DDP セグメント毎に STag の検証と物理アドレスの取得が必要となり、SMT や PBLMT へのアクセスがより多く発生することによる要因が大きい。

7.3 通信遅延

通信遅延は、RDMA Write と RDMA Read のそれぞれのオペレーションについて測定を行った。RDMA Read オペレーションでは、1 つの RDMA Read Request メッセージが送信される。RDMA Write オペレーションの通信遅延時間は、QP に WQE が投入されてから、初めのペイロードデータが TO LLP FIFO に出力されるまでの時間とした。RDMA Read オペレーションの通信遅延時間は、オペレーションが送信する RDMA Read Request メッセージがペイロードデータを持たないため、全てのヘッダが TO LLP FIFO に出力し終わるまでの時間とした。

表 1 に通信遅延の測定結果を示す。また、RDMA Write オペレーションの通信遅延の処理内訳を表 2 に示す。RDMA Read の通信遅延の処理内訳は、RDMA Write オペレーションのものから、STag 読み込み、PBL 読み込みによる遅延を除いたものとほぼ等しい。

結果より、通信遅延のうち大きな割合を占めるのが、STag 情報と物理アドレスの取得による SMT と PBLMT へのアクセスであることが分かる。一方、RDMA Read オペレーションの通信遅延が RDMA Write オペレーションの半分以下の値となっているのは、STag 情報と物理アドレス取得が不要であり、SMT と PBLMT へのアクセスが発生しないためである。

8. まとめ

本研究では、RDMA 通信機構を備えるネットワークインタフェースコントローラを設計し、ExpEther への実装を行った。また、その基本通信性能の評価を行った。

表 2 RDMA Write オペレーションにおける通信遅延の処理内訳

モジュール	処理内容	クロック
SQ Work Manager	WQE 取得	3
	フェンス判定	1
	リクエスト	1
SM Requester	リクエスト受理、リクエスト	1
Work Acceptor	リクエスト受理、リクエスト	1
SM Work Resolver	リクエスト受理	1
	オペレーション検証	1
	STag 読み込み	10
	STag 検証	1
	PBL 読み込み	10
	DMA 要求	1
Segment Generator	DMA 長取得	3
	セグメント長算出	1
	ヘッダ書き込み	4
	DMA データ読み込み	4
	ペイロード書き込み	1

RTL シミュレーションを用いたコントローラ単体での性能評価において、125MHz の動作周波数で 64bit のデータ幅を持つ ExpEther 開発ボードにおける理論スループットである 953.6MBytes/sec に対して、データ送信では 92.4%、データ受信では 83.4%の性能を達成することが分かった。通信遅延の測定では、STag 情報の参照や物理アドレスの取得が RDMA 通信における通信遅延の増加の主な要因となることが判明した。

今後は、ExpEther 開発ボードにおける実機動作検証を進めていく予定である。

謝辞

本研究の一部は、総務省の委託研究「次世代バックボーンに関する研究開発」プロジェクトの成果である。

参考文献

1) 日本電気株式会社: . http://www.nec.co.jp.
2) J Suzuki, Y Hidaka, J Higuchi, T Yoshikawa and A Iwata: ExpressEther - Ethernet-Based Virtualization Technology for Reconfigurable Hardware Platform, *High Performance Interconnects*, pp. 45–51 (2006).
3) RDMA Consortium: . http://www.rdmaconsortium.org.
4) Hemal Shah, James Pinkerton, Renato Recio and Paul Culley: Direct Data Placement over Reliable Transports (Version 1.0) (2002). http://www.rdmaconsortium.org.
5) Renato Recio, Paul Culley, D. Garcia and Jeff Hilland: An RDMA Protocol Specification (Version 1.0) (2002). http://www.rdmaconsortium.org.
6) Jeff Hilland, Jim Pinkerton, Renato Recio and Paul Culley: RDMA Protocol Verbs Specification (Version 1.0) (2003). RDMA Consortium.
7) Hideyuki Shimonishi, Junichi Higuchi, Takashi Yoshikawa and Atsushi Iwata: A Congestion Control Algorithm for Very Short Distance Communications, *IEICE technical report NS2006-244*, pp. 453–458 (2006).
8) Altera Corporation: . http://www.altera.com.
9) Inventure inc.: . http://www.inventure.co.jp.