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

2006－ARC－170 (15)
2006／11／30

DIMMnet-2を用いた間接メモリアクセスの高速化

宮代具隆[†] 宮部保雄[†] 北村聡[†]
田邊昇[††] 中條拓伯[†††] 天野英晴[†]

本研究では、メモリスロット装着型ネットワークインタフェースであるDIMMnet-2上に、リストアクセス機構を設計・実装した。このリストアクセス機構は、オンボードメモリ上の不連続なデータに対してアドレスリストを用いた効率的なアクセスを行うことができる。本論文では、この詳細について述べるとともに、NAS Parallel Benchmarks (CG) のカーネル部にこのリストアクセス命令を適用することで、クラスCにおいて1.4倍の速度向上が可能であることを示した。

The Acceleration of Indirect Memory Accesses Using DIMMnet-2

TOMOTAKA MIYASHIRO,[†] YASUO MIYABE,[†] AKIRA KITAMURA,[†]
NOBORU TANABE,[††] HIRONORI NAKAJO[†††] and HIDEHARU AMANO [†]

DIMMnet-2 is a network interface card which is connected to a memory slot of a commodity PC. A hard-wired list access module is designed and implemented on the card for processing vector access operation. The mechanism corrects fragmented data in the on-board memory using an address list. This paper introduces the detail and the performance evaluation of the list access mechanism. With the list access operations, the performance of a part of the NAS Parallel Benckmarks kernel (CG, class C) reaches 1.4 times compared to the original kernel.

1. はじめに

現在、PC等をはじめとする主な計算機で利用されているキャッシュメモリは、連続アクセスに対して効果を発揮するように設計されているため、キャッシュライン単位でメモリへのアクセスを行う。そのため、関係データベース処理や疎行列の演算などで発生する不連続アクセスにおいては、キャッシュラインあたりの有効データが減り、メモリバンド幅を有効に使うことができない。また、このような場合キャッシュ内に不要なデータが増加するため、キャッシュのミスヒットも増える。

そこで我々は、図1に示すようなベクトルアクセス命令をサポートするメモリモジュールを提案してきた[1]。まず、不連続アクセスが行われる前に、予めベクトルアクセス命令を発行しておき、アクセス対象となる不連続領域をまとめてバッファにプリフェッチする。

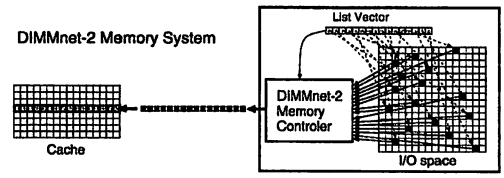

図1 DIMMnet-2を用いた場合のキャッシュの挙動

その後、実際にデータが必要になった時にバッファに対してブロックアクセスを行うことで、メモリのアクセスレイテンシを隠蔽すると共に、キャッシュラインあたりの有効なデータが増加し、メモリバンド幅を浪費しない効率的なアクセスが可能となる[2]。

本研究では、メモリスロット装着型ネットワークインタフェースであるDIMMnet-2[3]上に、アドレスリストを用いた不連続アクセスを行うリストアクセス機構をハードワイヤードで実装した。また、NAS CGベンチマークのカーネル部に、このリストアクセス命令を適用することで、処理時間の短縮が可能であることを示す。

以降、2章でDIMMnet-2ネットワークインタフェースの概要を示し、3章でリストアクセス命令の詳細を、4章でリストアクセス機構の実装の詳細を述べる。ま

† 慶應義塾大学
Keio University
†† (株) 東芝, 研究開発センター
Corporate Research and Development Center,Toshiba
††† 東京農工大学
Tokyo University of Agriculture and Technology

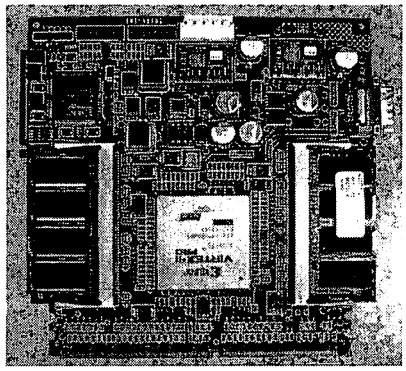

図 2 DIMMnet-2 試作基板の概観

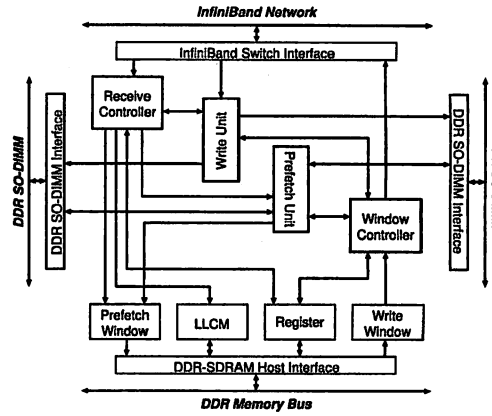

図 3 DIMMnet-2 CoreLogic の構造

た、5 章でその評価を示し、6 章にて関連研究を紹介し、7 章で結論と今後の予定について述べる。

2. DIMMnet-2

2.1 DIMMnet-2 試作基板

DIMMnet-2 は、大規模な共有メモリシステムの構築を目的とした PC クラスタ用ネットワークインタフェースで、DDR-SDRAM スロットに装着される。現在、図 2 に示す試作ボード[4] が完成している。ボードの中央部には FPGA(Xilinx Virtex-II Pro XC2VP70-7FF1517C)[5] を搭載しており、ここにネットワークの制御を行うコントローラや、ベクトルアクセス機構、プリフェッチされたデータを格納するバッファなどの論理回路 (CoreLogic) を実装する。CoreLogic は 100MHz で動作させ、PC-1600 の規格に対応している。ボード右上部には InfiniBand コネクタが搭載されており、これを利用してリモートホストとの通信を行う。また、ボードの左右に各 256MByte のノート PC 用 DDR SO-DIMM が計 2 枚搭載されている。

2.2 DIMMnet-2 へのアクセス

FPGA 上に実装された CoreLogic の内部構造を図 3 に示す。DIMMnet-2 では、ホスト PC は図 3 の下部に示される、ホストインタフェースと直結したバッファやレジスタ群 (Prefetch Window, Write Window, LLCM, Register) を介して、ネットワークインタフェース上の SO-DIMM へ間接的にアクセスを行う。これは、近年の DDR-SDRAM の高速な転送速度を考慮すると、ホスト PC から FPGA を経由してボード上の SDRAM に直接アクセスするのは、遅延や電気的な問題により困難だと考えられるためである。

ユーザが利用可能なバッファには、以下に示すものがある。

- Write Window は SO-DIMM に書き込むデータを格納するためのバッファで各プロセスに 2KByte(512Byte の領域を 4 つ) ずつ割り当てられる。ホストからは書き込み専用で、コントローラからは読み出し専用のデュアルポートメモリとなっている。
- Prefetch Window は SO-DIMM から読み出したデータを格納するバッファで、各プロセスに 2KByte(512Byte の領域を 4 つ) ずつ割り当てられる。ホストからは読み出し専用で、コントローラからは書き込み専用のデュアルポートメモリとなっている。
- LLCM(Low Latency Common Memory) はパケット受信時のステータス保持などに使用される汎用バッファで、ホスト・コントローラの双方から読み書き可能となっている。
- Register はホスト・コントローラの双方から読み書き可能なレジスタで、各種ベクトルアクセス命令の発行や、コントローラのステータスの取得等に使用される。

これらのバッファ・レジスタはユーザプロセスのアドレス空間にマップされており、ユーザプロセスからはメインメモリへのアクセスと同様の操作でアクセスできる。また各領域は、Pentium Pro 以降の IA32 プロセッサで利用することのできる MTRR(Memory Type Range Register) を適切に設定することで、各バッファへのアクセス高速化を図っている。

表 1 に示すように、Write Window はキャッシュを汚さずに高バンド幅で書き込み可能な Write-Combining 属性とした。また、Prefetch Window は読み出し時にバーストアクセスとなる Write-Back 属性とし、データを読み出す前に CLFLUSH 命令を使用することで、

表 1 MTRR とアクセス権の設定

Module	MTRR	Host	Controller
Prefetch Window	WriteBack	Read Only	Write Only
Write Window	WriteCombining	Write Only	Read Only
LLCM	Uncachable	Read/Write	Read/Write
Register	Uncachable	Read/Write	Read/Write

キャッシュ上の古いデータを無効化してから利用する。

LLCM と Regsiter はステータスの取得やコマンドの発行などに使用されるため、キャッシュされないよう Uncachable 属性とした。

2.3 コントローラ内部の動作

ホスト PC が Register にベクトルアクセス命令を書き込むと、Window Controller が命令を解釈し、必要に応じて SO-DIMM へのアクセスを行うモジュールを起動する。モジュールには、SO-DIMM から Prefetch Window にデータを読み出す Prefetch Unit と、Write Window に書かれたデータを SO-DIMM に書き込む Write Unit があり、これらが実際にベクトルアクセス命令を実行する。

また、2 枚の SO-DIMM の領域は、8Byte ごとにアドレスが交互に割り振られたインタリーブドメモリとなっており、デュアルチャンネルでのアクセスが可能である。

SO-DIMM からの Read もしくは Write が終了すると、各モジュールは Register 内の対応するフラグを立てる。ホスト PC はこのフラグをポーリングすることで、ベクトルアクセス命令の終了を知ることができる。

3. リストアクセス命令

DIMMnet-2 のベクトルアクセス機構で処理可能なリストアクセス命令 (VLI, VSI) の概要を以下に示す。これらの命令は予め SO-DIMM 上に格納されたアドレスリストを参照して間接アクセスを行う。

3.1 リストロード命令 VLI

`VLI(SRCOff, DSTOff, LIST, Iteration, DTYPE)`

VLI は、図 4 に示すように、まず LIST で指定されたアドレスからアドレスリストを読み出し、次にそのリストに従い DTYPE サイズのデータを Iteration 個読み出して、Prefetch Window の DSTOff で指定されたアドレスに格納する。このとき、DTYPE には 4, 8, 16, 32, 64, 128Byte を、Iteration には、1 から 255 までの値を指定することが可能である。また、SRCOff で指定した値はリストに加算され、ゼロ以外の値を指定すればオフセットからの相対アクセスとなる。

3.2 リストストア命令 VSI

`VSI(SRCOff, DSTOff, LIST, Iteration, DTYPE)`

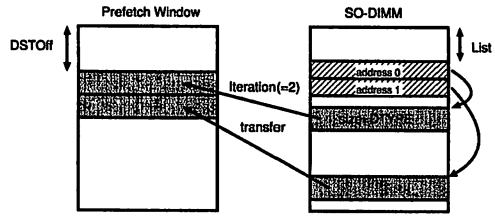

図 4 リストロード (VLI)

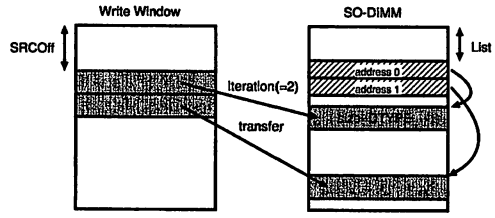

図 5 リストストア (VSI)

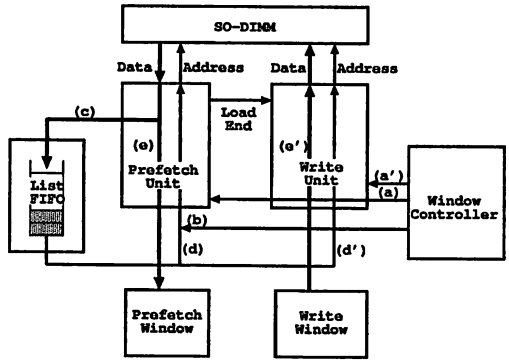

図 6 リストアクセス機構

VSI は、図 5 に示すように、VLI と同様の手順でリストを読み出した後、さらに Write Window の SRCOff で指定されたアドレスからデータを読み出し、リストに従って SO-DIMM へとデータを書き込む。DTYPE や Iteration については VLI と同様である。また DSTOff も同様に、相対アクセス時のオフセットとして用いられる。

4. 実　　装

前述の機能を実現するため、リストを格納するための FIFO を追加し、リストアクセス機構を Prefetch Unit と Write Unit に追加した。設計は Verilog-HDL を用いて行い、対象の FPGA に実装した。その詳細を図 6 に示す。

4.1 VLI 発行時の動作

VLI(リストロード) が Register に書き込まれると、Window Controller が命令の解釈を行い、Prefetch Unit に要求を伝える (図 6-(a))。要求を受けた Prefetch Unit は、SO-DIMM から該当のリスト領域を読み出し (図 6-(b))、List FIFO へとアドレスデータを格納する (図 6-(c))。読み出しが完了次第、List FIFO からデータを読み出し (図 6-(d))、再び SO-DIMM へアクセスを行い、今度は読み出されたデータを Prefetch Window へと書き込み、命令は完了となる (図 6-(e))。

4.2 VSI 発行時の動作

VSI(リストストア) は Prefetch Unit と Write Unit が協調して動作を行う。Register に書き込まれた命令を解釈した Window Controller は、Prefetch Unit と Write Unit の双方に VSI の要求を伝える (図 6-(a, a'))。

要求を受けた Prefetch Unit は VLI の時と同様に SO-DIMM からデータを読み出し (図 6-(b))、List FIFO へとアドレスデータを格納する (図 6-(c))。読み出しが完了次第、今度は Write Unit へと読み出し完了を通知する (図 6-(Load End))。

Write Unit はこの間待機しており、読み出し完了通知を受けると List FIFO からアドレスデータを読み出し (図 6-(d'))、Write Window から SO-DIMM へとデータを書き込み、命令は完了となる (図 6-(e'))。

5. 評　　価

5.1 評価環境

今回実装を行ったリストアクセス機構を用いて、基本性能となるメモリバンド幅の測定と、実際のアプリケーションに VLI を適用した際の加速率の測定の 2 点を行った。評価に用いた環境を表 2 に示す。

表 2　評価環境

CPU	Pentium4 2.6GHz
L1 Cache(Data)	8KB
L2 Cache	512KB
Memory	PC-1600 512MB × 1
	DIMMnet-2 × 1
Chipset	VIA VT8751A
OS	RedHat 8.0 (Kernel 2.4.27)
compiler	g++ 3.3.6
compile option	-O0 -march=pentium4 -msse2

5.2 間接アクセス時のメモリバンド幅

4096 × 4096 の倍精度のデータが格納されている行列 a からランダムに生成したリスト idx に従ってデータを読み出し、変数 b へと格納する以下に示すようなプログラムを作成した。

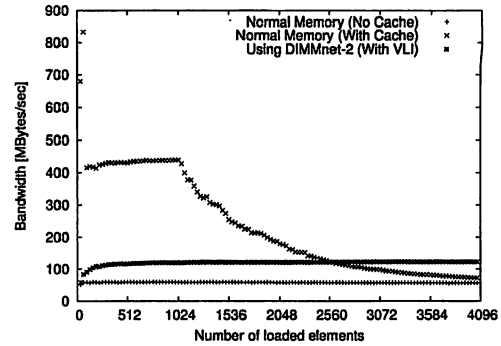

図 7　VLI のメモリバンド幅

```
for (i = 0; i <= max; i++) {
  b = a[idx[i]];
}
```

このプログラムを以下に示す 3 つのパターンにおいて実行し、その実行時間と転送サイズから、読み出す要素数 (max) 毎にメモリバンド幅を求めた。行列 a とリスト idx は通常のメモリ領域と DIMMnet-2 の双方に格納しておき、ローカル変数 b へ対象の a を全て格納し終わるまでの時間を実行時間とした。

- 通常のメモリを用いた場合 (キャッシュ有り)
- 通常のメモリを用いた場合 (キャッシュ無し)
- DIMMnet-2 の VLI を用いた場合

キャッシュ無しの場合は、試行のたびに CLFLUSH 命令を発行し該当領域のキャッシュを無効化している。また、DIMMnet-2 の Prefetch Window は 1 枚 512 Byte であるため、一度の VLI で読み出せる倍精度型のデータは 64 要素となる。それ以上のデータを読み出す場合は Window を 2 枚使用し、VLI の命令発行と読み出しを交互に行った。

読み出した要素数に対するバンド幅をプロットした結果を図 7 に示す。尚、結果ははじめに 100 回試行した後、その後 10000 回実行して得られた値の平均値を用いている。

キャッシュ無しの場合の純粋なメモリバンド幅は、読み出す要素数が極端に少ない場合を除き、一様に約 50MBytes/sec となった。これに対して DIMMnet-2 を用いた場合、VLI 発行やベクトルアクセス命令完了フラグのポーリングによるオーバーヘッドの影響が少なくなる 512 要素以降、キャッシュ無しに比べて 2.4 倍となる約 120MBytes/sec で安定している。

また、行列 a の要素を繰り返し使うことを想定し

たキャッシュ有りの場合のメモリバンド幅は、1024要素の読み出しまではアクセス対象となる要素がCPUのキャッシュに乗り切るため、64要素付近でピーク値となる約800MBytes/secを記録した後、その後は400MBytes/sec程度を示している。しかし、1024要素以降はアクセス対象となる要素がキャッシュから溢れ、要素数が増すにつれてバンド幅が低下している。2560要素以上のデータを読み出す場合では、通常のメモリよりもDIMMnet-2のバンド幅が上回った。

5.3 NAS CG ベンチマーク

次に、実際のアプリケーションでVLIを使用した際の予備評価として、シリアル版NAS Parallel Benchmarks[6]のCG法のカーネル部のみにベクトルアクセス命令を適用し、通常のメモリで実行した際とDIMMnet-2を用いた場合の双方について実行時間を測定し、クラス別に比較を行った。

NAS Parallel BenchmarksはNASAによって開発された並列コンピュータ向けのベンチマーク集で、問題の規模によりS, W, A, B, Cとクラスが分かれており、後者になるほど問題サイズは大きくなる。

また、今回評価に用いたCG法は、共役勾配法によって正値対象疎行列の固有値を求めるアルゴリズムを実行するが、その実行時間の大半は以下のコード☆に示される処理に費されている。

```
for (j = 1; j <= lastrow-firstrow+1; j++) {
  sum = 0.0;
  for (k = rowstr[j]; k < rowstr[j+1]; k++) {
    sum = sum + a[k] * p[colidx[k]];
  }
  w[j] = sum;
}
```

コード中には配列colidxによる配列pへの間接参照が存在し、問題サイズが大きくなると、間接参照部がキャッシュに全て乗り切らずメモリへのアクセスが頻発し、これがボトルネックとなって性能が低下する。

そこで、予めpとcolidxをSO-DIMMに格納しておき、この部分にDIMMnet-2のベクトルアクセス命令VLIを適用した。メモリバンド幅測定時と同様にPrefetch Windowは2枚使用し、交互に命令の発行とデータの読み出しを行った。

カーネル部の平均実行時間をクラス別に示したものが図8である。クラスB以下においては、演算対象となるの行列pの要素とcolidxの大部分がキャッシュに乗ってしまうため、DIMMnet-2を用いた場合よりも、通常のメモリで実行した場合の方が実行時間が速い。しかし、クラスCにおいては、アクセス対象の要素がキャッシュから溢れオリジナルに比べて28%実行時間が削減され、1.4倍の速度向上が見られた。問題サイズが大きくなるほどVLIによる間接アクセスが有効に働くため、より大きなサイズの問題では更に速度の向上が期待できる。

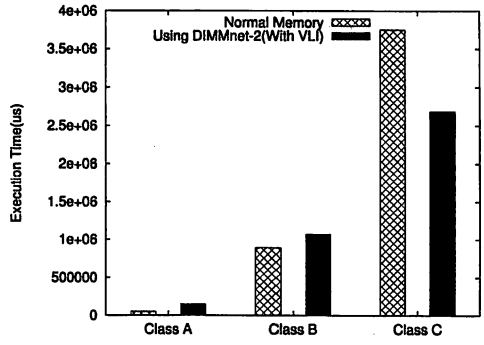

図8 NAS CG カーネル部の実行時間

今回の予備評価では、純粋にNAS CGのカーネル部のみの評価を行ったが、本研究に先立って行われたソフトウェアエミュレーションの結果[2]では、CG法のクラスBについては1.81倍、クラスCについては3.39倍の性能向上が期待できるとの報告がなされている。今回得られた結果はそれに比べると遅く、まだハードウェア・ソフトウェアの両面において改善の余地があるものと考えられる。

性能低下の要因に関して、エミュレーション結果では、VLIを発行してから命令完了のフラグが立つまでの時間が1000nsを越えると性能が大きく落ちるとの結果が出ている。現在の実装において命令完了までの時間を測定してみると、NAS CGの場合リストの内容にも左右されるが、平均で3500ns程度となっており、この遅延が性能低下の主な原因と予測される。

今後はこの点について改良を行った上で、キャッシュの制御に関する命令の調整など、ソフトウェアでの最適化も図っていく必要がある。

6. 関連研究

メモリ上の不連続なデータに連続的なアクセスをする研究としては、Impulse[7]が挙げられる。Impulseでは、メモリコントローラが使用していない物理アドレスに不連続な領域へのエイリアスを割り当てることで、

☆ RWCPによって移植されたC言語版のNAS CGを用いた

データの連続化を行っている。しかし、メモリコントローラに独自のものを使用するため既存のチップセットを使用することができず、またOSにも変更を加える必要がある。

また、キャッシュに対して特殊なコントローラを装備し、コンパイラが生成するコマンドによってストライドベクトルの収集等を自動的に行うハードウェアも提案されている[8][9][10]。しかし、これらの提案はいずれも特殊なマルチプロセッサのキャッシュシステムを対象としており、一般のPCに用いることはできない。

これらに対してDIMMnet-2は一般的なDDR-SDRAMスロットに装着されるため、既存のCPUやチップセットを利用することができる点に特徴がある。

また、実機での実現例があるメモリスロット装着型のアクセラレータとしては、TKDM[11]が挙げられるが、これはSDR-SDRAMスロットに対応したものであり、またホストインタフェース部がUncachable属性のメモリ領域にマップされているため、ホストとのデータ転送速度の面で問題がある。DIVA[12]はDDR-SDRAMスロットに対して装着可能で、かつネットワークインタフェースを持つProcessing-In-Memoryシステムであるが、これは不連続アクセス機構は有していない。

7. 結論と今後の課題

本研究では、リストアクセス機構をDIMMnet-2上に設計・実装した。その結果、通常のメモリでは50MBytes/sec程度のメモリバンド幅となる間接アクセスにおいて、ベクトルアクセス命令VLIを用いることで、120MBytes/secでのデータ読み出しが可能であることを示した。また、NAS CGベンチマークのカーネル部を用いてアプリケーションレベルの予備評価を行い、クラスCにおいて1.4倍の速度向上が可能であることを示した。

今後はソフトウェアエミュレーションでの結果と照らし合わせた上でハードウェア・ソフトウェアの最適化を進め、今回予備評価に用いたNAS CGベンチマークにおいて、カーネル部だけでなく、全体にDIMMnet-2のベクトルアクセス命令を適用した場合の評価を行う予定である。

謝辞 本研究は総務省戦略的情報通信研究開発推進制度(SCOPE)の一環として行われたものである。DIMMnet-2の開発に関する議論、開発にご参加頂いている日立情報通信エンジニアリング株式会社の今城氏、岩田氏、上嶋氏、横浜国立大学の箱崎氏、慶應義塾大学の渡邊氏、大塚氏に感謝いたします。

参考文献

1) 田邊 昇, 中武 正繁, 箱崎 博孝, 土肥 康孝, 中條 拓伯, 天野 英晴: プリフェッチ機能付きメモリモジュールによる不連続アクセスの連続化, 情報処理学会アーキテクチャ研究会, Vol. 2004-ARC-157, pp. 139–144 (2004).
2) 田邊 昇, 安藤 宏, 箱崎 博孝, 土肥 康孝, 中條 拓伯, 天野 英晴: プリフェッチ機能を有するメモリモジュールによるPC上での間接参照の高速化, 情報処理学会論文誌コンピューティングシステム, Vol. 46, No. SIG 12(ACS 11), pp. 1–12 (2005).
3) 田邊 昇, 濱田 芳博, 三橋 彰浩, 中條 拓伯, 天野 英晴: メモリスロット装着型ネットワークインタフェース DIMMnet-2 の構想, 情報処理学会アーキテクチャ研究会, Vol. 2003-ARC-152, pp. 61–66 (2003).
4) 北村 聡, 伊豆 直之, 田邊 昇, 濱田 芳博, 中條 拓伯, 渡邊 幸之介, 大塚 智宏, 天野 英晴: DIMMnet-2 ネットワークインタフェースボードの試作, 情報処理学会アーキテクチャ研究会 (SWoPP2004), Vol. 2004-ARC-159, pp. 151–156 (2004).
5) Xilinx: . `http://www.xilinx.co.jp/`.
6) NASA: NAS Parallel Benchmarks. `http://www.nas.nasa.gov/Software/NPB/`.
7) John Carter, Wilson Hsieh, Leigh Stoller, Mark Swanson, Lixin Zhang, Erik Brunvand, Al Davis, Chen-Chi Kuo, Ravindra Kuramkote, Michael Parker, Lambert Schaelicke and Terry Tateyama: Impulse: Builing a Smarter Memory Controller, *Fifth International Symposium on High Performance Computer Architecture(HPCA-5)*, pp. 70–79 (1999).
8) 吉田 明正, 前田 誠司, 尾形 航, 笠原 博徳: "Fortran マルチグレイン並列処理におけるデータローカライゼーション手法", 情報処理学会論文誌, Vol. 36, No. 7 (1995).
9) 中済 光昭, 岡本秀輔, 曽和将容: 分散共有メモリ型並列コンピュータにおけるプログラム制御キャッシュメモリ, 情処研報 96-ARC-119 (1996).
10) 坂本 勝人, 藤原 崇, 川口貴裕, 岩井 啓輔, 森村知宏, 天野英晴: マルチグレイン並列処理を利用したソフトウェア制御キャッシュの開発, 信学報 CPSY SWoPP 98 (1998).
11) Plessl, C. and Platzner, M.: TKDM - a reconfigurable co-processor in a PC's memory slot, *Proceedings of 2003 IEEE International Conference on Field-Programmable Technology 2003(FPT2003)*, pp. 252–259 (2003).
12) Sumit D. Mediratta, Craig Steele and Jeffrey Draper: An Area-efficient and Protected Network Interface for Processing-In-Memory Systems, *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 2951–2954 (2005).