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

2005－ARC－163 (3)
2005／5／31

DIMMnet-2ネットワークインタフェースにおけるプリフェッチ機構の実装と評価

宮代具隆[†1] 宮部保雄[†1] 伊澤徹[†1]
北村聡[†1] 箱崎博孝[†2] 田邊昇[†3]
中條拓伯[†4] 天野英晴[†1]

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、プリフェッチ機構を備えたReadモジュールを設計・実装した。このプリフェッチ機構は、ベクトル命令によって不連続なデータへ効率的にアクセスを行うことができる。また、対角要素を対象とする行列計算にベクトル命令を実際に適用し、現時点で約18%の処理時間短縮が可能であることを示した。

Implementation and Evaluation of the Prefetching Mechanisms on DIMMnet-2

TOMOTAKA MIYASHIRO,[†1] YASUO MIYABE,[†1] TETSU IZAWA,[†1]
AKIRA KITAMURA,[†1] HIROTAKA HAKOZAKI,[†2] NOBORU TANABE,[†3]
HIRONORI NAKAJO[†4] and HIDEHARU AMANO [†1]

DIMMnet-2 is a network interface card which is connected to a memory slot of a commodity PC. Design and implementation of the interface module for processing reading requests from the host PC are introduced. The module equipps a prefetching mechanism which can correct data fragments by a vector access command so as to improve vector processing in the host PC. With the vector operations, the performance improvement of the diagonal matrix computation is shown.

1. はじめに

近年、CPUの性能向上によって1サイクルあたりの実行時間は短くなっているが、メモリのアクセスレイテンシはCPUの1サイクルと比較すると長く、CPUが命令を実行する上でメモリへのアクセスタイムがボトルネックとなることが多い。

このような問題を解決するために、現存するCPUとメモリによって構成されるシステムでは、ほとんどが階層型のキャッシュメモリを採用している。キャッシュアーキテクチャはシーケンシャルアクセスに対して効果を発揮するよう設計されており、関係データベース処理や対角行列の演算などで発生する不連続アクセスに対しては、キャッシュラインあたりの有効データが減り、メモリのバンド幅やキャッシュ領域が浪費されてしまう。またそれ故、キャッシュのミスヒットも増える。

そこで、不連続なメモリアクセスが行われる前に、ベクトル命令によってアクセス対象となる不連続領域をまとめてバッファに溜めておき、それに対してブロックアクセスを行う。このようなベクトル命令によるデータのプリフェッチ[1]により、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、上記のようなプリフェッチ機構を備えたReadモジュールをハードワイヤードで実装した。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章でベクトル命令の動作を示し、4章でプリフェッチ機構の詳細を述べる。5章でその評価を示し、6章にて関連研究を紹介し、7章でまとめる。

†1 慶應義塾大学
Keio University
†2 横浜国立大学
Yokohama National University
†3 (株) 東芝, 研究開発センター
Corporate Research and Development Center,Toshiba
†4 東京農工大学
Tokyo University of Agriculture and Technology

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を構

築することを目的とした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1に示すような試作ボード[4]が完成している。ボードの中央部にはFPGA(Xilinx Virtex-II Pro XC2VP70-7FF1517C)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、プリフェッチ機構、プリフェッチされたデータを溜め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規格に対応している。ボードの右上部には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2枚搭載されている。

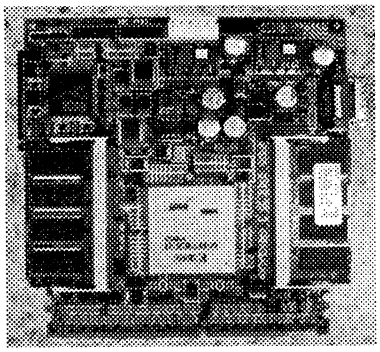

図1 DIMMnet-2 試作基板の概観

2.2 DIMMnet-2上の資源へのアクセス

FPGA上に実装するCoreLogicの内部構造を図2に示す。

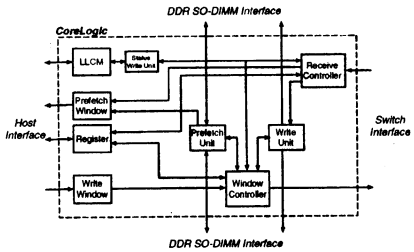

図2 DIMMnet-2 CoreLogic の構造

DIMMnet-2では、ホストPCは図2の左部に示される、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, Registerなど)を経由して、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。

バッファには、SO-DIMMに書き込むデータを格納するWrite Windowと、SO-DIMMから読み出したデータが格納されるPrefetch Windowがあり、512Byteを1枚のWindowとしている。

Write Windowは各プロセスに1KByte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は各プロセスに2KByte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。

ユーザプロセスがCoreLogic内に設けられたUser Registerに必要な情報を書き込むことで、各種ベクトルアクセス命令がWindow Controllerに発行される。

命令を解釈したWindow Controllerは、必要に応じてSO-DIMMへのアクセスを行うUnitを起動する。Unit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[5]があり、これらが実際にベクトル命令を実行する。本研究の設計・実装対象であるプリフェッチ機構は、このPrefetch Unitのことを指す。

これらのUnitにアクセスされる各SO-DIMMの領域は、8Byteごとに交互にアドレスが割り振られている。これは、2枚のSO-DIMMに同時にアクセスできるようにしバンド幅を向上させるためである。

3. ベクトル命令

DIMMnet-2のコントローラで処理可能なベクトル命令の一部を以下に示す。

3.1 連続アクセス命令 (VL, VS)

```
VL(SRCOff, DSTOff, Size)
VS(SRCOff, DSTOff, Size)
```

SRCOffで指定されたアドレスから、Sizeで指定された大きさのデータをDSTOffで指定されたアドレスへ格納する命令である。

SO-DIMMからのREAD操作に相当するVLでは、図3に示されるように、SO-DIMMからPrefetch Windowへとデータが読み出される。一方、SO-DIMMに対するWRITEに相当するVSでは、図4に示されるように、Write WindowからSO-DIMMへとデータが

書き込まれる。

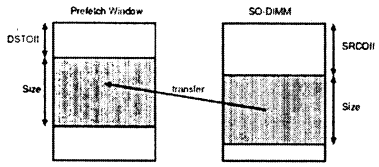

図 3 連続ロード (VL)

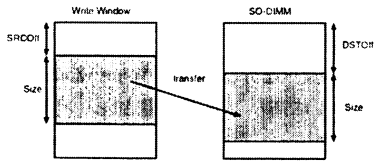

図 4 連続ストア (VS)

3.2 ストライドアクセス命令 (VLS, VSS)

```
VLS(SRCOff, DSTOff, Stride, Iteration, DTYPE)
VSS(SRCOff, DSTOff, Stride, Iteration, DTYPE)
```

SRCOff で指定されたアドレスから、Stride で指定された間隔で配置された DTYPE サイズのデータを Iteration 回読み出し、DSTOff で指定されたアドレスから連続した領域に格納する命令である。

VLS では、図 5 に示されるように、SO-DIMM から Prefetch Window へとデータが読み出される。一方 VSS では、図 6 に示されるように、SO-DIMM から Prefetch Window へとデータが読み出される。

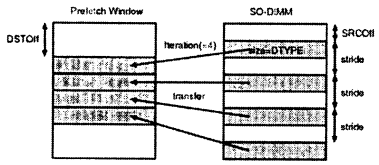

図 5 ストライドロード (VLS)

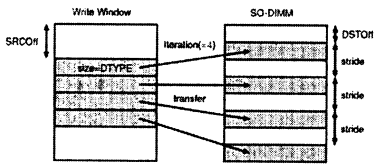

図 6 ストライドストア (VSS)

4. 実　　装

4.1 プリフェッチ機構の概要

プリフェッチ機構 (Prefetch Unit) は大きく分けて 4 つのモジュールから成り、それらは図 7 に示すように接続されている。

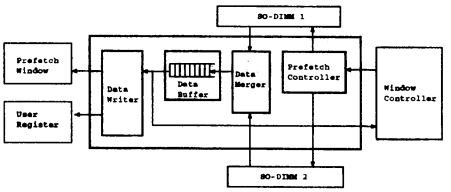

図 7 Prefetch Unit 内外のモジュール間の接続

個々のモジュールの概要を以下に示す。

- Prefetch Controller

  Prefetch Unit を統括するモジュールである。コントローラからプリフェッチ要求を受け取り、発行されたベクトル命令の種類を判断して各モジュールの初期設定を行い、SO-DIMM インタフェースを介して Read アクセスを行う。

  2 節で述べたように、SO-DIMM のアドレスは 8Byte ずつ 2 枚交互に割り当てられている。よって、ストライドアクセスなどで複数回 Read アクセスを発行する場合、2 枚の SO-DIMM からデータが前後して到着する可能性がある。そのため、データの順序を制御するためのタグを SO-DIMM インタフェースから出力されるディスティネーションアドレスの一部に埋め込んだ上で Read アクセスを行っている。

- Data Merger

  2 枚の SO-DIMM から読み出されるデータを 4Byte 単位でアラインし、まとめた上で後述の Data Buffer に格納するモジュールである。Data Merger はデータ受信部と、データ整列部の二つからなる。

  データ受信部は 2 枚の SO-DIMM 毎にリングバッファを持っており、受信したデータをタグと分割して格納する。データ受信部は独自にカウンタを持っており、このカウンタの値が受信したデータのタグと一致するものを取り出すことで、2 枚の SO-DIMM から読み出されたデータの順序を保証している。

  データ整列部では、データ受信部のリングバッファからデータを取り出し、バイトアラインメントを行う。ここでベクトル命令の SRCOff によるデータのミスアラインが補正される。

- Data Buffer

  Data Merger によってアラインされたデータを一時的に格納するモジュールである。32bit 幅、深さ 128 の FIFO 4 つからなり、最大で 2KByte までデータを格納することができる。リモートホス

トをディスティネーションとするベクトル命令が発行された時には、Prefetch Window にはデータを書き込まず、Window Controller へデータを転送する。

- Data Writer

  Data Buffer からデータを読みだし、ベクトル命令の DSTOff に合わせてアラインメントした上で Prefetch Window へと書き込むモジュールである。書き込みを行うと同時に、Prefetch Window Flag の管理も行う。Prefetch Window Flag は、書き込み対象となる Prefetch Window の 512Byte の領域を 128Byte で 1 ラインとし、1 ラインに付き 1bit、計 4bit のフラグで書き込みの終了をホスト側に知らせる役割を持っている。

4.2 VLS 発行時の動作

VLS(ストライドロード) が発行された場合の大まかな流れを以下に示す。各モジュールは必要な回数だけ動作を終えると待ち状態へ戻るようになっている。

(1) Prefetch Controller がリクエストをラッチ、各モジュールを設定

(2) Prefetch Controller が SRCOff に Stride を加算しながら、SO-DIMM に対して Iteration 回リードアクセスを発行

(3) Data Merger が SO-DIMM から読み出されたデータを連結し、SRCOff を元にアラインして Data Buffer に格納

(4) Data Buffer 内に 128bit 以上のデータがバッファリングされ次第、Data Writer がデータを要求

(5) Data Buffer から読み出したデータを Data Writer が DSTOff に合わせてデータをアラインして Prefetch Window に転送

5. 評　　価

5.1 評価環境

実装を行った Prefetch Unit の処理できるベクトルアクセス命令のうち、VLS を発行した場合の動作を中心に、RTL シミュレーションと実機計測の両面から評価を行った。

シミュレーションによる評価では、ncverilog 05.10-s018 をシミュレータに用い、回路内部での理論的な遅延時間を求めた。実機評価では表 1 に示される環境にて、実際にベクトル命令をユーザプロセスから発行した時の実行時間を計測した。

また、今回の評価では IA32 アーキテクチャに基づくプロセッサを用いたため、MTRR(Memory Type Range Register) を適切に設定することにより、各 Window へのアクセス高速化を行っている。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キャッシュ上の古いデータを無効化した。

表 1　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	gcc 3.3.5
compile option	-march=pentium4 -msse2

5.2 VLS の基本性能

Window Controller から Prefetch Unit へ VLS が発行された時点から、Prefetch Window に実際にデータを転送し終わる (Prefetch Window Flag が全て立つ) までの時間を RTL シミュレーションによって求めた。データ長は 4,8,16,32,64,128Byte の 6 通りとし、ソースアドレスは 0x00 とした。

また、ストライド間隔は、最大データ長の 128Byte より十分長い 256Byte 以上で、かつ、後述のアクセスパターンの偏りを考慮し、2 枚の SO-DIMM に均等にアクセスが分散される 264Byte とした。Iteration 数はサイズ (DTYPE)×Iteration がちょうど転送上限の 512Byte になる値とし、処理に要した実行時間 (ns) からバンド幅 (MByte/s) を求めた。また、VLS と同様の処理を VL の連続発行によって実現し、VLS の処理時間と比較した。結果を表 2 に示す。

表 2　VLS と VL の性能比較

	バンド幅 (MB/s)		
データサイズ (Byte)	VLS	VL	性能比
4Byte × 128	283.9	21.0	13.51
8Byte × 64	514.0	42.2	12.17
16Byte × 32	567.8	84.0	6.76
32Byte × 16	762.9	161.7	4.72
64Byte × 8	871.9	277.4	3.14
128Byte × 4	939.0	498.2	1.88

4Byte×128 の VLS の時に、同等の処理を行う VL に比べ 13.51 倍のバンド幅を達成した。

5.3 アクセスパターンによる VLS の性能変化

データサイズが 4Byte の場合について、ストライド間隔を 16Byte から 60Byte まで 4Byte ずつ増やしながら実行し、処理時間を測定した結果を図 8 に示す。横

軸がストライド間隔、縦軸が処理時間を表している。

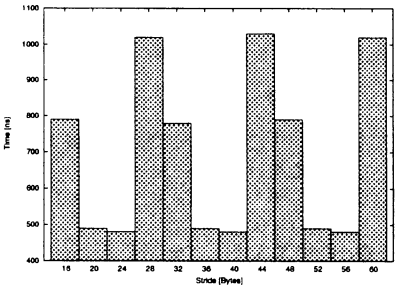

図 8 VLS 実行時のストライドと処理時間の関係

結果から、ストライド間隔の 16Byte を 1 周期として処理時間が増減を繰り返していることがわかる。これは、2 枚の SO-DIMM へのアドレス割り振りが 8Byte ずつになっており、SO-DIMM へのアクセスパターンに偏りが生じるためである。ソースアドレスを 0x00 とした時の、ストライド間隔 (16,20,24,28Byte) におけるアクセスパターンを図 9 に示す。

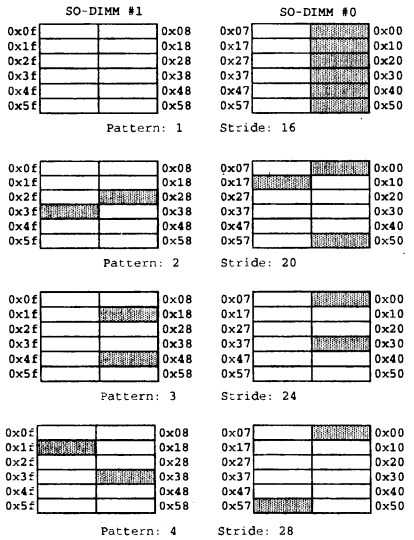

図 9 VLS におけるアクセスパターン

パターン 1 では片方の SO-DIMM にリクエストが集中し、必要サイクルが増加している。対応策としては、ベクトル命令を利用する側のプログラムが配列のサイズを 1 増やすなどして回避する方法が挙げられる。パターン 2 と 3 では、左右の SO-DIMM に均等にリクエストが割り振られている。パターン 4 では開始時のアクセスが均等に割り振られず、SO-DIMM インタフェースのタイミング仕様に合致しないため、パターン 1 以上に必要サイクルが増えてしまう。これは Prefetch Unit 側で適切なバッファリングを行うことで、処理時間の改善が期待できる。

このようなストライド間隔による処理時間の大幅な違いは、データ長が 8Byte 以下の時にのみ発生する。16Byte 以上の場合、一度のアクセスで必ず左右の DIMM に対する Read リクエストが発生するため、アクセスに大幅な偏りが生じることはないためである。

5.4 VLS の動作時間内訳

VLS(4Byte×128, アクセスパターン 3) の処理時間の内訳を表 3 に示す。処理中は SO-DIMM への Read 要求発行と SO-DIMM インタフェースの busy 状態による待機が繰り返されるため、それらは合計値として示した。このデータは Read アクセスが片方の SO-DIMM に集中するパターン 1 の場合である。

表 3 VLS(512Byte) 処理時間の内訳

動作	必要 clock 数
要求受け付け	0
各モジュール初期設定	+2
SO-DIMM への連続 Read 要求発行 (合計)	+128
SO-DIMM インタフェースの処理完了待ち (合計)	+16
Prefetch Window へデータ転送	+16
計	162 clk

5.5 VLS の実機評価

実機評価に際し、以下に示すような、SIZE×SIZE の大きさの倍精度浮動小数点のデータを持つ対角行列 A の逆行列を求めるプログラムを作成した。

```
for ( i = 0;  i < SIZE; i++ ) {
  r[i][i] = 1 / A[i][i];
}
```

行列 A の対角要素は SIZE+1 個毎に出現するため、これをストライドとして VLS を適用することが出来る。DTYPE は行列の要素が double 型であるため 8Byte とした。一度の VLS では転送上限サイズの 512Byte に合わせた 64 個の対角要素しか取得できないため、プリフェッチしたデータを計算している間に、次の VLS を発行する方法を取った。問題サイズを 128 から 2048 まで 2 倍ずつ増加させ、VLS 使用時の実行時間と、適用前プログラムの実行時間を比較した結果を表 4 に示す。

表 4 VLS 使用時と通常時の実行時間

SIZE	通常時 (us)	VLS 使用時 (us)	実行時間比
128	307.01	316.08	+2.93%
256	1442.65	1181.55	-18.09%
512	5740.51	5061.79	-11.83%
1024	11442.18	10650.81	-6.91%
2048	23074.20	22062.73	-4.38%

問題サイズが 256 の時に、もっとも実行時間が短縮し約 18%実行時間を減らすことができた。また、問題サイズが 128 の時は逆に実行時間が増加する結果と

なった。これは VLS 起動のオーバヘッドが実行時間に比べて大きい為だと考えられる。問題サイズ 512 以降は、高速化したものの問題サイズ 256 に比べて高速化の影響は少なかった。今回は、ベクトル命令に合わせたインラインアセンブラの埋め込みがまだ完全でなかったため、予想外の箇所でボトルネックが生じている可能性があり、この点を突き止めていく必要がある。

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[6] 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、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ない。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[7][8][9]。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アクセラレータとしては、TKDM[10] 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る。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基板は現在のところ DIMMnet-2 だけである。

7. 結論と今後の課題

本研究では、プリフェッチ機構を備えた Read モジュールである Prefetch Unit を DIMMnet-2 上に実装し、ベクトル命令 VLS によって不連続データに対する効率的なアクセスが可能であることを示した。

今後の課題としては、その他に実装が予定されている各種ベクトル命令を実装を完了させ、実機レベルで検証した上で、それを使った NAS CG ベンチマーク等のアプリケーションを動作させることが挙げられる。また、タイミング制約である 100MHz での動作を安定して達成するための論理回路の最適化なども今後の課題の一つである。

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 (株) 日立 IT の今城氏、岩田氏、上嶋氏、慶應義塾大学の西助手、渡邊氏、大塚氏に感謝いたします。

参考文献

1) 田邊 昇, 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, 情報処理学会アーキテクチャ研究会, Vol. 2003-ARC-154, pp. 139–144 (2003).
2) 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).
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) 宮部保雄: DIMMnet-2 ネットワークインタフェースの通信制御部の実装と評価, 慶應義塾大学理工学部 卒業論文 (2004).
6) 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).
7) 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).
8) 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).
9) 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).
10) 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).