ICOT Technical Report: TR-564

TR-564

PIM/m要素プロセッサの
アーキテクチャ

中島　浩，武田保孝，中島克人(三菱)

May, 1990

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

PIM/m 要素プロセッサのアーキテクチャ

中島　浩　　武田　保孝　　中島　克人

（三菱電機）

梗概

第5世代コンピュータ・プロジェクトの一環として，並列推論マシンPIM/mを開発している。PIM/mの基本的な設計方針は，2次元メッシュ状のネットワーク，マイクロプログラム制御によるKL1-Bの実行など，Multi-PSI/v2を継承したものである。しかし，要素プロセッサ数がMulti-PSI/v2の4倍である256に拡大されたのに伴い，実装密度はVLSI技術を用いることによって4倍に向上している。また，要素プロセッサのアーキテクチャに大幅な変更を加えることにより，単体性能も3～5倍に向上している。

1　はじめに

第5世代コンピュータプロジェクトの一環として，大規模並列推論マシンPIM/mを開発している[1]。PIM/mは，我々が既に開発した中規模の並列推論マシンMulti-PSI/v2[2]の後継機として位置付けられている。従って，2次元格子状のネットワークや，マイクロプログラム制御により高水準の機械命令を実行するCISCタイプの要素プロセッサなど，その基本的な設計方針はMulti-PSI/v2を継承したものとなっている。

しかし，要素プロセッサのアーキテクチャや実現方式は，プロセッサ数の拡大や性能向上のために大幅に変更されている。例えば，要素プロセッサの核部に高集積のVLSIを用いることにより実装密度が飛躍的に向上し，Multi-PSI/v2の4倍である256個の要素プロセッサをほぼ同一の筐体に実装することが可能となった。

性能面では，要素プロセッサのアーキテクチャの変更が大きな効果を発揮している。その重要なものの1つとして，5ステージのパイプラインの導入がある。PIM/mでは，通常の計算機で用いられているデコード，アドレス計算，データ・フェッチに加えて，論理型言語の実行には必須であるデータのタイプ・チェックもパイプライン化されているため，実行ステージの負荷を大きく軽減することができた。また，キャッシュ・メモリを命令用とデータ用に分離する，所謂ハーバード・アーキテクチャの採用によって，メモリ・アクセスに伴うオーバヘッドが著しく改善されている。これらのアーキテクチャ改良と，デバイス技術の進歩による遅延時間短縮の相乗効果により，ベンチマーク・プログラムの性能は3～5倍に向上している。

以下，本稿ではPIM/mの要素プロセッサのアーキテクチャについて，従来からの改良点や論理型言語の処理に特有の機能を中心に述べる。

2　PIM/mの概要

PIM/mは最大16×16の2次元格子状に256個の要素プロセッサを結合した疎結合のマルチ・プロセッサである。システムの構成単位は4×8＝32の要素プロセッサであり，これが1つの筐体に実装される。この筐体を

Architecture of PIM/m Processor Element
Hiroshi NAKASHIMA, Yasutaka TAKEDA and Katsuto NAKAJIMA
Mitsubishi Electric Corporation

8個並べることにより $16 \times 16 = 256$ の最大構成が実現できる他，$8 \times 8 = 64$ や，$8 \times 16 = 128$ などの構成にも可能である。各筐体には4つのSCSIポートがあり，ディスクなどの入出力装置や，フロント・エンド・プロセッサ(FEP)を接続することができる。

PIM/mで実行されるプログラムは，並列論理型言語KL1[3]を用いて記述される。KL1はPrologと同様に論理型言語であるが，AND関係にあるゴールの並列実行や，単方向ユニフィケーションによる同期やストリーム通信など，並列処理を陽に記述する機能を持っている。KL1で記述されたプログラムは，抽象機械命令であるKL1-B[4]にコンパイルされ，マイクロプログラム・エミュレータにより実行される。

一方，フロントエンド・プロセッサ(FEP)では，Prologにオブジェクト指向の機能を負荷した言語であるESP[5]，及びそのベースである論理型言語KL0が実行される。FEPのCPUには，PIM/mの要素プロセッサがほとんどそのまま用いられるため，要素プロセッサにはKL1/KL0の双方を実行するための柔軟な構成が必要となる。

3 要素プロセッサのアーキテクチャ

要素プロセッサは，図1に示すように3つのVLSIチップPU, CU, NUを中心に構成されている。PU (Processing Unit)は，5ステージのパイプラインを持ったマイクロプロセッサであり，32KwのWCSに格納されたマイクロプログラムによりKL1/KL0を実行する。CU (Cache Unit)は1Kwの命令キャッシュ，4Kwのデータ・キャッシュ（データ・アレイはチップ外）の他，アドレス変換バッファや主記憶の制御回路からなる。NU (Network Unit)は，隣接する4プロセッサを結ぶネットワーク・チャネルのスイッチングを行うとともに，PUが行うメッセージの送受信を制御する。なお，要素プロセッサのマシン・サイクルは60nsである。

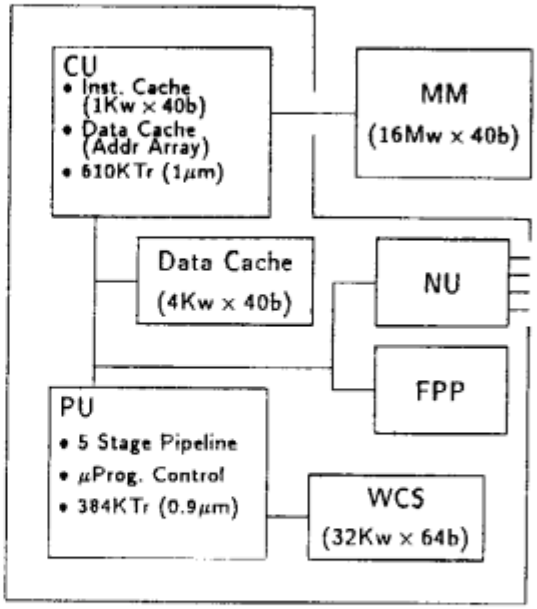

図 1: 要素プロセッサ

3.1 PU (Processing Unit)

PUは図2に示すように，**D, A, R, S, E**の5ステージから構成されている。PUはWAMをベースとしたKL1/KL0のためのマクロ命令を実行する。WAMのレジスタの内，引数レジスタ(**An/Xn**)は32ワードのRAMであるRF (Register File)を用いて実現される。また，その他の制御用レジスタは32ワードのRAMであるWR (Work Register)に格納される。但し，プログラム・カウンタと2つの構造体ポインタは，ハードウェア・カウンタを用いて実現されている。

D (Decode)ステージには命令のデコードのためのRAMテーブルがあり，マイクロプログラムの実行開始アドレスの他，アドレス計算，オペランドの生成方法など，下流のステージを制御するためのナノ・コードが格納されている。デコーダをRAM用いて構成したことは，KL1/KL0の双方のサポートや，マイクロプログラム開発の容易化に多大な効果をもたらしている。

A (Address Calculation)ステージは，ナノ・コードにしたがって以下の資源をソースとするアドレス計算を行う。

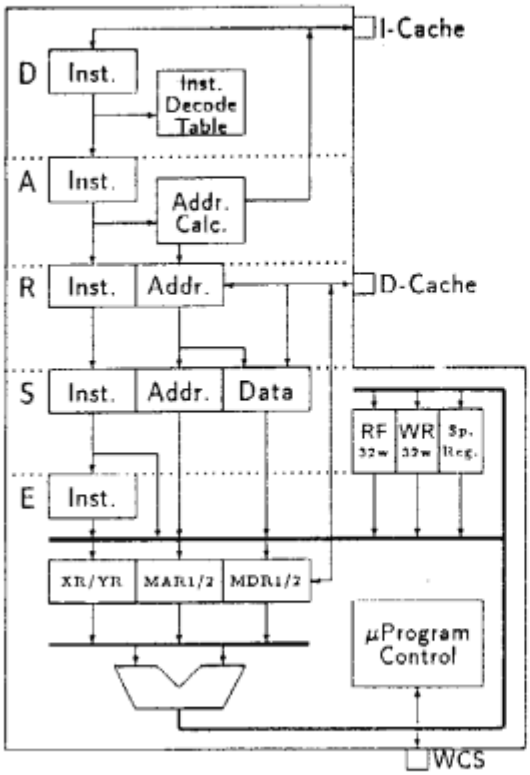

図 2: PU の構成

- 命令のオペランド・フィールド
- プログラム・カウンタ
- RF
- アドレス・レジスタ（2個）

2つのアドレス・レジスタには，KL1実行時にはゴールの情報を格納するフレームのベース・アドレスと，alternativeのアドレスが格納される。またKL0実行時には，環境フレームのベース・アドレス(**E**)と，continuation (**CP**)が格納される。なお，**A**ステージでは，分岐を含む命令フェッチの制御も行う。

R (Read Data)ステージでは，**A**ステージが行ったアドレス計算の結果に基き，主記憶からオペランドを読出す。

S (Set up)ステージは，ナノ・コードにしたがって，以下の資源の中から3つのオペランドを選択して，**E** (Execution)ステージに引渡す。

- 命令のオペランド・フィールド
- 主記憶上のオペランド及びそのアドレス
- RF
- WR
- 構造体ポインタ（2個）

このオペランド・セットアップ操作は，通常は**R**ステージに相当する部分で行われるが，PUでは特に**S**ステージを設けてセットアップ操作を行っている。これは，データ・タイプ判定とデレファレンスをパイプライン化するためである。

まず**S**ステージには，データ・タイプ判定のための機能として以下のものがある。

(1) 主記憶上のオペランドのタグと即値との比較結果に基く，**E**ステージで実行されるマイクロプログラムの実行開始番地の修飾。

(2) 主記憶上のオペランドのタグによる多方向分岐のセットアップ。

(3) **E**ステージに引渡すオペランドのタグと即値との比較結果に基く2方向分岐条件のセットアップ。

これらの機能の内(1)と(2)を実現するためには，**R**ステージと**E**ステージの間に特別なステージが必要となる。

Sステージのもう一つの重要な機能として，デレファレンス機能がある。デレファレンスには，RFをルートとするものと主記憶上のデータをルートとするものとがある。**R**ステージでは，前者の場合にはRFの指定されたエントリが*reference*である時にのみ，また後者の場合には無条件でオペランド・フェッチを行う。いずれの場合も，**S**ステージはフェッチしたデータが*reference*か否かを判定し，*reference*でないデータが得られるまで繰返しオペランド・フェッチを行う。

また，KL1を実行する際には，デレファレンス時にMRB (Multiple Reference Bit) [6]

を用いた参照パスの単一性の判定が行われる。KL1 のデータには図3に示すように MRB が付加されており，MRB がオフであるポインタが指しているデータの MRB がオフであれば，他にはそのデータを指しているポインタは存在しない。従って，MRB がオフであるようなポインタの連鎖については他に参照パスが存在しないため，ポインタの終端にあるデータに関するユニフィケーションが完了した時点で，リアルタイムにゴミとして回収できる。

このリアルタイム・ガベージ・コレクションをサポートするために；

- **SRP**(Signle Reference Path)
  ポインタ連鎖中の全ての MRB がオフ
- **COL**(Collectable):
  最初の2つのポインタの MRB がオフ

の情報が E ステージに渡される。また，この情報に基きマイクロプログラムの実行開始番地を修飾することもできる。従って，図3に示すようにポインタ連鎖やデレファレンス結果の回収の要否を容易に判断することができる。

E ステージは，フェーズ1とフェーズ2の並行に動作する2つのブロックから構成され，マイクロプログラムによって制御される。フェーズ1には命令レジスタ，RF, WR, 及びいくつかの特殊レジスタがあり，これらの中から2つのものが選択されてフェーズ2に送られる。なお，マクロ命令を実行するマイクロプログラム・ルーチンの最後のマイクロ命令では，フェーズ1は S ステージによって制御され，オペランドのセットアップのために使用される。

フェーズ2には，2つのメモリ・アドレス・レジスタ MAR1/MAR2，2つのメモリ・データ・レジスタ MDR1/MDR2，及び2つの作業用レジスタ XR/YR がある。フェーズ2ではこれらの中から2つのレジスタを選択し，その演算結果をフェーズ1及びフェーズ2の

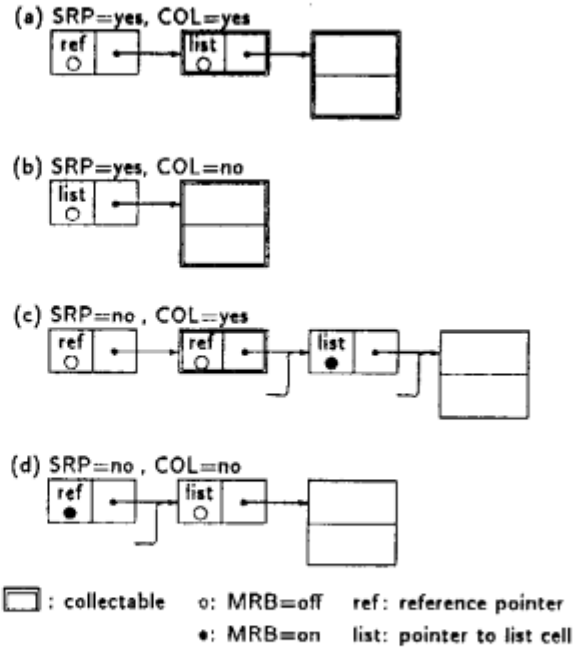

図3: MRB によるリアルタイム・ガベージ・コレクション

レジスタに書込む。なお，タグの付け替えや MRB のオン／オフなど，タグ操作のための機能がどちらのフェーズにも用意されている。また，マイクロプログラムの制御機能として，フェーズ1/2で選択したレジスタのタグによる2方向分岐，MDR1/2 のタグによる多方向分岐など，タグの判定のための様々な機能も用意されている。

3.2 CU (Cache Unit)

CU は図4に示すように，命令キャッシュ，データ・キャッシュ，及び主記憶インタフェースから構成されている。なお，キャッシュ・メモリやアドレス変換バッファの容量については，Multi-PSI/v2 の要素プロセッサである PSI-II[7] でコンパイラを実行した際のデータに基く評価を行い，チップ面積などの実装上の問題を勘案して決定した[8]。

命令キャッシュには，4w/block，ダイレクト・マッピング方式の 1Kw のキャッシュ・メモリと，2セット，64 エントリのセット・ア

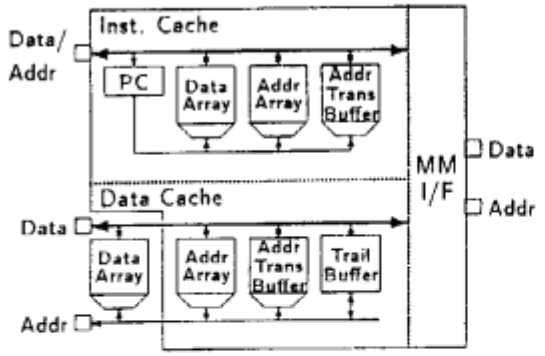

図 4: CU の構成

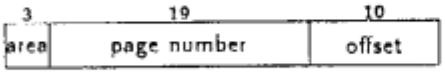

図 5: 論理アドレス

ソシアティブ方式のアドレス変換バッファがある。命令キャッシュのヒット率の予測値は94.4%であり，キャッシュ容量を2倍にしてもヒット率は全くといってよいほど向上しないため，妥当な容量であると考えられる。アドレス変換バッファについては，実質的に100%のヒット率が得られている。

チップのピン数削減のために，PU と命令キャッシュのインタフェースはアドレス／データ共用信号となっているため，命令キャッシュにはプログラム・カウンタのコピーがある。このプログラム・カウンタの操作に関しては，命令フェッチの際には自動的にインクリメントされるのでオーバヘッドはないが，PU が分岐処理を行う際には正しいプログラム・カウンタの値をセットする必要がある。しかし，このための性能低下は，分岐の頻度が約20%であること，その内の1/3以上は他の処理と並行処理されること，などを勘案すると1.5%程度と比較的軽微であると見積もられる。

データ・キャッシュには，4w/block，ダイレクト・マッピング方式の4Kw のキャッシュ・メモリと，2セット，64エントリのセット・アソシアティブ方式のアドレス変換バッファがある。但し，データ・アレイはチップ・サイズの関係からチップ外部に置かれている。データ・キャッシュのヒット率は，ほとんど理想値に近い99.2% と予測されている。

データ用のアドレス変換バッファに関しては，そのエントリ・アドレスの生成方式が問題となる。PIM/m の論理アドレスは図5に示すように，*area* と言うフィールドを持っており，複数のスタックが異なる *area* に割当てられる。スタックの深さは数ページ程度であることが多いため，単純に論理ページ番号の下位ビットをアドレス変換バッファのエントリ・アドレスとすると，*area* の間でエントリの奪い合いが発生することが予想される。そこでエントリ・アドレスを，*area* とプロセス番号の一部を結合したものと，論理ページ番号の下位ビットとの排他的論理和とした。この結果，*area* やプロセス間でのエントリの奪い合いが大幅に低下し，実質的に100% のヒット率を得ることができた。

この他，データ・キャッシュには論理型言語に特有のバックトラック処理をサポートするための，Trail Buffer という32エントリのスタックがある。Trail Buffer は Trail Stack の最上部のキャッシュであり，変数の代入のためのメモリ書込操作の際に，そのアドレスが自動的にプッシュされる。また，バックトラックの際にはアドレスのポップ・アップと変数のリセットのための書込が CU 内部で行われる。

主記憶（物理記憶）の各ワードには，1ビット誤り訂正/2ビット誤り検出可能な7ビット ECC が付加されている。主記憶インタフェースでは，ECC による誤り訂正，検出，ECC の付加の他，主記憶の動作タイミングの生成も行う。

4 性能評価

表1は，PIM/m 要素プロセッサ，Multi-PSI/v2, PSI-II，及びいくつかの論理型言語専用プロセッサの，*append* の性能を示したものである。なお，表には KL1 などの並列論理型言語 (PLPL) と，ESP, Prolog などの逐次型の論理型言語 (SLPL) の双方の性能を示している。

表から明らかなように，PIM/m の性能は Multi-PSI/v2，PSI-II に比べて，KL1 では

表 1: *append* の性能

	KLIPS PLPL/SLPL
PIM/m	833/ 1282
Multi-PSI/v2(PSI-II)	179/ 430
Pegasus[9]	—/≈350
PLM[10]	—/≈400
CHI-II[11]	—/ 490
KCM[12]	—/ 833
IPP[13]	—/ 1035
IP1704[14]	—/ 1100
PIM/p[15]	≈600/ —

PLPL: Parallel Logic Programming Language

SLPL: Sequential Logic Programming Language

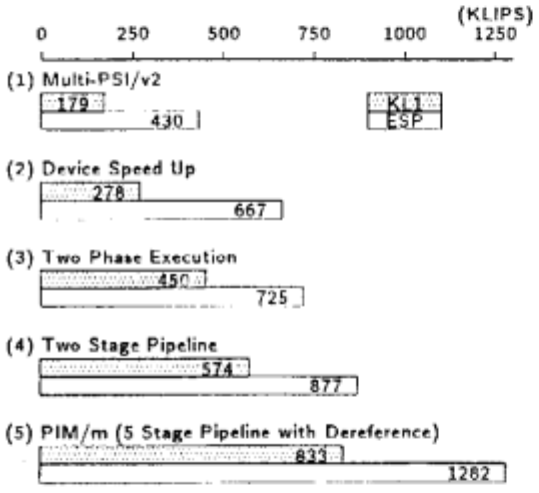

図 6: パイプライン化による性能向上

5 倍，ESP では 3 倍に向上している。これは VLSI 化によるサイクル・タイムの短縮だけではなく，論理型言語特徴を考慮したパイプラインの導入やキャッシュ・メモリの構成など，アーキテクチャの大幅な改良に負う所が大きい。

特に，パイプライン化されたデータ・タイプ判定／デレファレンスが有効であると考えられるので，以下の 5 つのアーキテクチャについて *append* の性能を評価した。

(1) Multi-PSI/v2: Multi-PSI/v2 及び PSI-II の性能。データ・タイプ判定／デレファレンスは以下の操作により行われる。

(a) レジスタ・ファイルの読出

(b) 読出したデータのタグと即値の比較

(c) 比較結果に基くマイクロプログラムの分岐先アドレス生成

(d) マイクロ命令のフェッチ

この操作は 1 マシン・サイクル (Multi-PSI/v2 では 200ns，PSI-II では 155ns) で行われるが，この操作のためのパスがクリティカル・パスとなっている。

(2) Device Speed Up: Multi-PSI/v2 及び PSI-II の論理素子を，PIM/m と同等の速度のものとした場合の性能。マシン・サイクルは 100ns 程度と想定される。

(3) Two Phase Execution: (1) のクリティカル・パスを (a) と (b)，及び (c) と (d) の 2 フェーズに分割した場合の性能。この構成は PU の E ステージと同様であり，60ns のマシン・サイクルで動作することができる。

(4) Two Stage Pipeline: (a) と (b) の操作のためにマクロ命令レベルのパイプライン・ステージを設けた場合の性能。この構成は PU の S 及び E ステージを組み合わせたものと殆ど同様であるが，デレファレンスはパイプライン化されていない。

(5) PIM/m: デレファレンスがパイプライン化されている PIM/m の性能。また，デレファレンス結果に対するデータ・タイプ判定の場合，E ステージでの条件分岐が不要となるので，(4) よりも強力である。

評価結果は図 6 に示すとおりであり，パイプライン化されたデータ・タイプ判定／デレファレンスが，マシン・サイクルとサイクル数の両面で性能向上に大きく寄与していることが明らかである。

5 おわりに

高集積の VLSI チップを用いることにより，従来の 4 倍の実装密度を達成することができ，

大規模な並列処理が実現できるようになった。また、論理型言語の特徴を考慮したパイプラインの導入やキャッシュ・メモリの構成などにより、従来に比べて3 ～ 5倍に性能を向上させることができた。

参考文献

[1] S. Uchida, K. Taki, K. Nakajima, A. Goto and T. Chikayama, Research and Development of the Parallel Inference System in the Intermediate Stage of the FGCS Project. *Proc. Intl. Conf. on Fifth Generation Computer Systems 1988* (1988).

[2] Y. Takeda, H. Nakashima, K. Masuda, T. Chikayama and K. Taki, A Load Balancing Mechanism for Large Scale Multiprocessor Systems and Its Implementation. *Proc. Intl. Conf. on Fifth Generation Computer Systems 1988* (1988).

[3] K. Ueda, Guarded Horn Clauses: A Parallel Logic Programming Languages with the Concept of a Guard. *TR 208, ICOT*(1986).

[4] Y. Kimura and T. Chikayama, An Abstract KL1 Machine and its Instruction Set. *Proc. 4th IEEE Symp. on Logic Programming* (1987).

[5] T. Chikayama, Unique Features of ESP. *Proc. Intl. Conf. on Fifth Generation Computer Systems 1984* (1984).

[6] T. Chikayama and Y. Kimura, Multiple Reference Management in Flat GHC. *Proc. 4th Intl. Conf. on Logic Programming*(1987).

[7] H. Nakashima and K.Nakajima, Hardware Architecture of the Sequential Inference Machine : PSI-II. *Proc. 4th IEEE Symp. on Logic Programming* (1987).

[8] 中島浩：PSI-II のメモリ・アーキテクチャ評価，情報処理学会計算機アーキテクチャ研究会 (1990).

[9] K. Seo and T. Yokota, Design and Fabrication of Pegasus Prolog Processor. *Proc. Intl. Conf. on Very Large Scale Integration* (1989).

[10] T. P. Dobry, A. M. Despain and Y. N. Patt, Performance Studies of a Prolog Machine Architecture. *Proc. 12th Intl. Symp. on Computer Architecture* (1985).

[11] S. Habata, R. Nakazaki, A. Konagaya, A. Atarashi and M. Uemura, Co-Operative High Performance Sequential Inference Machine: CHI. *Proc. 1987 Intl. Conf. on Computer Design* (1987).

[12] H. Benker, J. M. Beacco, M. Dorochevsky, Th. Jeffré, A. Pöhlmann, N.Noyé and B. Poterie, KCM : A Knowledge Crunching Machine. *Proc. 16th Intl. Symp. on Computer Architecture* (1989).

[13] S. Abe, T. Bandoh, S. Yamaguchi, K. Kurosawa, and K. Kiriyama, High Performance Integrated Prolog Processor IPP. *Proc. 14th Intl. Symp. on Computer Architecture* (1987).

[14] 相川 健，高代 慶昭，木下 常雄：AI ワークステーション用VLSI，東芝レビュー，Vol. 44, No. 10 (1989).

[15] T. Shinogi, K. Kumon, A. Hattori, A. Goto, Y. Kimura and T. Chikayama Macro-Call Instruction for the Efficient KL1 Implementation on PIM. *Proc. Intl. Conf. on Fifth Generation Computer Systems 1988* (1988).