Vol. 37 No. 5　　情　報　処　理　　May 1996

解　説

1. 第五世代プロジェクトにおける計算機アーキテクチャの研究をふりかえって†

小　池　汎　平††

1. はじめに

本稿では，1982年より10年以上にわたって続けられた第五世代コンピュータプロジェクトの中で行われた計算機アーキテクチャに関する研究について概観し，その問題点，今後の展望について論ずる．

2. 背　景

第五世代コンピュータプロジェクトは今から約15年前の1982年からスタートした．計画が発表されると同時に，かなりセンセイショナルな取り上げられ方をされ，世界的な注目を浴びることとなった．「人間と同じように思考する機械を日本が世界に先駆けて開発」などと(昨今のインターネットブームもそうだが)話題ばかりが一人歩きしすぎてしまった感があったが，プロジェクトの本来の目標は，将来の情報処理が知識情報処理へ向かうという予測に基づき，より高機能な人工知能プログラムを高速に実行する計算機システムを開発するための基礎技術を開拓することにあったと見るのが妥当であろう．

第五世代プロジェクトは，人工知能応用プログラムの研究，プログラミング言語の研究，計算機アーキテクチャの研究など多岐にわたる研究を含んでいたが，プロトタイプハードウェアの開発はこの中でも目玉となる研究の1つであった．このプロジェクトにおける計算機アーキテクチャ研究の位置付けは

- Lispマシンなどに代表される高級言語計算機
- 並列計算機
- データベースマシン

など，プロジェクト発足当時に，計算機アーキテクチャ研究分野で話題を集めていたいくつかのテーマの研究を統合的に推し進め，将来の新しい計算機アーキテクチャを模索することであった．

当時は，人工知能の研究に，記号処理Lispが多く用いられていた．Lispプログラムは単純な実装方法では多くのオーバヘッドを持つことから，計算機アーキテクチャの改良によってこれらのオーバヘッドを解消して高速に実行させようとする，いわゆるLisp専用計算機(Lispマシン)を開発し商用化しようという試みが各所で盛んであった．

プロジェクト発足前には，出発点としてすでに実績のあるLispを採用するかどうかという議論もあったようであるが，第五世代プロジェクトでは研究対象のプログラミング言語として，当時発表されて間もない論理型プログラミング言語を採用した．これにともない，第五世代プロジェクトにおけるアーキテクチャ研究の具体的な目標は，論理型プログラミング言語を高速に実行する(並列)計算機の開発ということになった．

3. 逐次型推論マシンアーキテクチャの研究

3.1 PSI-Iの開発

第五世代プロジェクトにおける計算機アーキテクチャの研究は，逐次型推論マシンPSI(Personal Sequential Inference machine)の1号機PSI-I[1]の開発によってスタートした．PSIは，論理型プログラミング言語で書かれたプログラムを逐次処理で高速実行する計算機であったが，当時商用化されていたLispマシンと同様に，プロジェクトの研究者に1人1台ずつ研究ツールとして大量に導入される，高性能パーソナルコンピュータ(今でいうワークステーション)としても位置付けられていた．

† A Retrospective Overview on Computer Architecture Researches in the Fifth Generation Computer Project by Hanpei KOIKE (Electrotechnical Laboratory).

†† 電子技術総合研究所

PSIで用いられる核言語として，逐次型論理型言語KL0が設計された．KL0は，Prologと似た逐次型論理型言語であり，オペレーティングシステムなどの記述が可能となるように，システム制御に必要な低レベルなシステム述語などが用意されていた．このことから，KL0は「PSIの機械語」などとも呼ばれた．

KL0は，Prologと大変に似た言語であり，PSI-IでのKL0の実行のメカニズムとしては，D. H. D. WarrenらがEdinburgh大学で開発し，当時世界最高速を誇っていたDEC-10 Prolog[2]が参考にされた．KL0のプログラムは，PSI-Iの命令列にコンパイルされて実行される．PSI-Iの命令セットは，ユニフィケーションなどの処理を行う高水準命令セットになっており，PSI-Iはこの命令セットをマイクロプログラムで書かれたインタプリタによって解釈実行する[3]．

KL0に用意されている様々なデータ型を識別するために，ポインタにはタグがつけられており，これを効率的に操作する目的でPSI-Iではタグアーキテクチャが採用された．また，KL0の1つの手続きは一般に複数の定義節からなっているが，手続きが呼ばれたときの引数のデータ型により実行可能な定義節の選択が必要となる．このために，PSI-Iではデータ型による柔軟な多重分岐が行えるようなハードウェアが採用された．これらのアーキテクチャ上の工夫により，PSI Iは30 KLIPSという速度を達成した．

3.2 PSI-IIの開発

PSI-Iの開発に引き続き，PSI-II[4]の開発が行われた．PSI-IIの開発の目標はシステムの小型化と高速化である．システムの小型化を達成するための主な手段として回路のLSI化が進められた．一方，速度の向上のために，PSI-IIでは，KL0の実行メカニズムとして，PSI-Iのものとは異なる方式が採用された．

PSI-IIにおいて新しいKL0実行メカニズムが採用されるきっかけとなったのは，PSI-Iの実行メカニズムのもととなったDec-10 Prologの開発者の1人でもあるD. H. D. Warrenが新しく発表した衝撃的な論文であった[5]．この論文で彼は，WAM (Warren's Abstract Machine)と呼ばれる新しいProlog実行用の命令セットを持った仮想計算機を提案し，構造体表現法を構造体共有方式から構造体コピー方式へ変更するなどの改良により，クロック10 MHzのパイプライン計算機を使用した場合に450 KLIPSの速度が達成可能であることを主張した．このことから，PSI-IIの新しい実行メカニズムはWAMの影響を強く受けたものとなり，その結果として，200 KLIPSを超す性能を達成することができた．

PSI-IIは，PSI-Iとのソフトウェア互換性を保ちながら，大幅な速度向上を達成し，その一方でデスクサイドキャビネット程度に小型化されて，研究用ツールとして大量に利用された．また，PSI-IIは，後に述べる並列推論マシン実験機であるMulti-PSIの要素プロセッサとしても利用された．

4. 並列推論マシンアーキテクチャの研究

4.1 初期の並列アーキテクチャの研究

第五世代プロジェクト発足当時は，並列計算機アーキテクチャの模索期であり，リダクションマシンやデータフローマシンなどの先進的なモデルが提案されて注目を集めるなどしていた時期であった．逐次型推論マシンの開発と並行して進められた初期の並列アーキテクチャの研究は，これらのデータフローマシンとリダクションマシンを参考として，基本的モデルを検討することから始まった．この結果，データフローモデルに基づく並列推論マシンPIM-Dとリダクションモデルに基づくPIM-Rが提案され，PIM-Dの実験機のハードウェアの開発などが行われた．また，この時期には関係データベースマシンDeltaの開発も行われた．

同時期に，後のPIMアーキテクチャに影響を及ぼすこととなったと思われる興味深い研究が行われた．委託研究として富士通研究所で行われた「株分けモデル」の研究である．この研究では，数台の汎用マイクロプロセッサからなる実験機を用いて，要素プロセッサ上に高速な逐次型Prolog処理系が実装され，これを拡張することによってOR並列Prolog処理系の作成が行われた．プロセッサ間での負荷分散戦略として，空きプロセッサからの要求があったときにのみ，探索木のなるべく根本に近いところにある未探索のゴールをわたす，すなわち，負荷分散は必要最小限，なるべく大きな処理単位で行う，という戦略

が採用され，このような方式で処理を行わせたときに高い性能が得られることを実証した．「株分けモデル」の研究は，要素プロセッサとして逐次型推論マシンの延長にあるものを用い，負荷分散は，なるべく大きな処理単位で必要最小限だけ行うという，後のMulti-PSI, PIMで採用される方式の原型となるものであった．

4.2 Multi-PSI

研究が中期に入ると，核言語KL 1の設計の進行とともに，並列推論マシンPIMのアキテクチャもしだいに明確化されていった．しかし，PIMの開発に本格的に着手するためには，様々な予備実験を重ねる必要があった．そこで，本格的なPIMの開発に先だって，並列論理型言語の実装方式の検討，様々な実験データの収集，そして，実機を用いた並列プログラミングの経験の早期の蓄積などを目的として，並列処理の実験機Multi-PSIの開発が行われた．

Multi-PSIは，その名前が示すように，開発済みの逐次型推論マシンのハードウェアを流用して構成された実験機である．Multi-PSIは，PSI-IIのCPUにメッセージ交換のハードウェアを追加したものを要素プロセッサとし，このような要素プロセッサ64台を8×8の2次元メッシュ網で接続した構成をとっている．

Multi-PSIは，けっして最高性能を狙って並列処理に特化させたハードウェアというわけではなかったが，安定した技術の採用により早期のシステム稼働を実現することができた．この結果，Multi-PSIを用いて，PIMの開発の準備として並列論理型言語の実装に関する様々な実験が行われるとともに，実際に並列に動くハードウェア上で様々な並列応用プログラムの開発も行うことによって多くの研究者が並列プログラミングについて実際的な経験を幅広く得ることもできた．この点でMulti-PSIの開発は，数々の有意義な成果を収めることによって，その後の第五世代コンピュータプロジェクトの進行に様々な面で大きく貢献することとなった．

4.3 5つのPIMアーキテクチャ

Multi-PSIでの経験をふまえて，プロジェクトの中期以降は，最終目標であるPIMの開発に全勢力が傾けられることとなった[6]．PIMにおいて高い総合性能を達成するために重要な技術目標として

- 単体プロセッサ性能の向上
- 高性能な要素プロセッサの結合方式の実現

の2点が掲げられた．PIMの開発において，これらの目標を達成するためにいくつかの異なるアプローチが提案され，それらの有効性の確認のために，それぞれの方法に重点をおいた5種類のモジュール，すなわち

- 推論処理モジュールPIM/p
- 並列実行モジュールPIM/m
- 接続実行制御モジュールPIM/c
- 交信管理モジュールPIM/i
- 知識ベース操作管理モジュールPIM/k

が開発されることとなった．これらのモジュールを総合することにより，プロセッサ台数1000台規模の並列推論マシンPIMが構成されることになる．各PIMモジュールの全体構成を**表-1**に，各PIMモジュールの要素プロセッサの構成を**表-2**に示す．表では比較のために先に述べたMulti-PSIのデータも示してある．

単体プロセッサ性能の向上の鍵を握るのは，要素プロセッサのCPUの命令対系とその実現方式である．PIMの開発時期は，世の中のプロセッサ設計技術がCISC方式からRISC方式へと大きく変化する時期と重なっており，PIMのCPUの設計もこのような動向の影響を強く受けることとなった．その結果，PIMの要素プロセッサのCPUは，あるものはRISC方式を，あるものはマイクロプログラム方式を，あるものは長命令(LIW)方式を採用する，というように大変バラエティに富んだものとなった．

重要なポイントの1つは，単純な命令セットを採用することにより高速なハードウェアが実現できるとされるRISC方式が，高機能な命令による実現が前提とされてきた論理型言語の処理にどこまで有効であるかであった．この点で注目すべき工夫がなされたのはPIM/pのCPUである[7]．PIM/pのCPUでは基本方式としてRISC方式が採用されたが，プロセッサ内に命令パイプラインと密に結合した内部命令メモリが設けられた．この命令メモリの中に複雑な処理を実現するためのマクロ命令を格納することにより，手続き呼び出しのオーバヘッドなしに複雑な処理を実現する命令コードの呼び出しを可能とし，単純な命令セ

表-1 PIM および Multi-PSI の全体構成

	PE	クラスタの構成			疎結合ネットワークの構成	
	総数	PE 数	メモリ	バス容量（最大）	トポロジ	バンド幅
PIM/p	512	8	256 MB	130 MB/s	6 次元超立方体	33 MB/s×2
PIM/m	256	1	80 MB	40 MB/s	2 次元メッシュ	8 MB/s
PIM/c	256	8	160 MB	200 MB/s	クロスバ（32×32）	40 MB/s
PIM/i	16	8	320 MB	30 MB/s	SCSI 接続	—
PIM/k	16	4	1 GB	80 MB/s（各階層）	2 階層キャッシュ	—
Multi-PSI	64	1	80 MB	25 MB/s	2 次元メッシュ	10 MB/s

表-2 PIM および Multi-PSI の要素プロセッサの構成

	PE アーキテクチャ			
	命令セット	プライベートメモリ	キャッシュ方式（状態数）	キャッシュサイズ（命令/データ）
PIM/p	RISC＋マクロ呼出	内部命令メモリ（8 k×6 B） ローカルメモリ（2 MB）	スヌープ 書込無効型（4）	64 KB
PIM/m	マイクロプログラム	書換可能制御記憶（32 k×64 bit）	書戻型（3）	5 KB/20 KB
PIM/c	マイクロプログラム	レジスタファイル（256 W） 書換可能制御記憶（32 k×104 bit）	スヌープ 書込無効型（5）	80 KB
PIM/i	長命令方式（LIW）	ローカルメモリ 160 KB（命令）80 KB（データ）	スヌープ 書込更新型（6）	160 KB/160 KB
PIM/k	RISC	ローカルメモリ 128 KB（命令）128 KB（データ）	2 階層スヌープ 書込無効型（4）	上層：128 KB/256 KB 下層：1 MB/4 MB
Multi-PSI	マイクロプログラム	書換可能制御記憶（16 k×53 bit）	書戻型（3）	20 KB

ットを持った高速 RISC プロセッサ上で，並列論理型言語を効率良く実行するために必要となる高機能命令を実現することを可能とした．

要素プロセッサ同士の結合方式には，それぞれの PIM で特徴のある方式が採用されている．PIM/p, PIM/c, PIM/i は 8 台の要素プロセッサをバス結合によって密結合したクラスタ構成をとっており，これらのクラスタ同士が，PIM/p は 2 系統の 6 次元超立方体網(ハイパキューブ網)で，PIM/c は 32×32 のクロスバ網で，PIM/i は SCSI バスで結合されている．これらのクラスタ内の各要素プロセッサは共有バスを介してメインメモリを共有しており，各要素プロセッサがスヌープキャッシュを持つことにより，メモリアクセス時間の短縮とバストラフィックの低減が図られている．PIM/m は Multi-PSI と同様にクラスタ構成はとらず，要素プロセッサ同士が 16×16 の 2 次元メッシュ網で結合されている．また，PIM/k は階層バス構成により多数のプロセッサがメモリを共有する方式を採用した．4 台の要素プロセッサが上層のバスで結合され，これらのバス同士が下層のバスで結合されている．

4.4 KL1 実装のためのソフトウェア技術

PIM ハードウェアの上での KL 1 プログラムの実行は，KL 1 コンパイラと KL 1 の実行を支援するファームウェアからなる KL 1 処理系によって行われる[8]．KL 1 処理系はアーキテクチャの異なる 5 種類の PIM のために作成しなければならなかったため，処理系作成の作業を効率良く進めるために，VPIM(Virtual PIM)と呼ばれる仮想マシンを想定した処理系を作成し，この処理系を個々の PIM に移植するという方法がとられた．

KL 1 で書かれたプログラムは，まず VPIM の機械語である KL 1-B へコンパイルされ，次に，それぞれの PIM の機械語へと変換される．

VPIMの機械語へコンパイルされる段階では様々な最適化が施される．VPIMの機械語は，この段階できめの細かい最適化が可能となるように設定された[9]．

クラスタ構造をとるPIMではクラスタごとに独立したメモリアドレス空間を持つ．これはKL 1の実装を行うときに難しい問題を生む．KL 1ではデータは多数のリストやベクタ同士がポインタ参照によって結合されている．このようなポインタ参照がクラスタ間にまたがった場合に，これをどのように実現するかが問題となった．この問題を解決するためにPIMでは，外部ポインタ参照を管理する輸出表/輸入表と呼ばれる表を用いる方法が用いられた[10]．

また，KL 1では，Lispなど他の記号処理言語と同様に，不要となったメモリ領域を再利用のために回収するガベジコレクションという処理が必要である．これをメモリ空間の共有されないクラスタ間にまたがってどのように実現するかも重要な問題であった．PIMでは，これを重み付き参照カウント(Weightcd Reference Count)方式と呼ばれる方式で解決した[10]．一方，クラスタ内のガベジコレクションには従来からあるコピー方式が用いられたが，KL 1のプログラムはメモリ消費が激しく，ガベジコレクションの頻度が高まりオーバヘッドが増加するという問題点があった．これを解決するために，多重参照ビット(Multiple Reference Bit)と呼ばれる手法が考案され，メモリ領域の即時回収によるメモリ消費の低減が図られた[11]．

5. 性能評価とその問題点

計算機ハードウェアも言語処理系も，その有効性の確認は，プログラムの実行速度を測定することによって行われるのが一般的である．

論理型言語処理系や推論マシンの実行速度は，LIPS(Logical Inference Per Second)またはRPS(Reduction Per Second)という単位で計られることが多い．これは1秒間に何回の推論，すなわちゴールの書換えが行われたかを表す．推論を行った回数などと書くといかにも大げさだが，実際にはこの数値はリストの連結を行うappendというプログラムを実行させて1秒間にいくつのリスト要素の連結を行うことができたかを測定して求めることが多い．このようにして求めた速度であることを明示するためにappend LIPSまたはRPSという用語も用いられる．append LIPSは約20年前に開発されたDEC-10 Prolog[2]で用いられた指標であるが，そのまま使われ続けて今に至っている．PIMの要素プロセッサのappend RPS値とこれをもとに計算した最大構成台数におけるピーク性能値を**表-3**に示す．

表-3 PIMおよびMulti-PSIの基本性能

	マシンサイクル	単体性能 append RPS	ピーク性能
PIM/p	80 ns	305 KRPS	156 MRPS
PIM/m	65 ns	600 KRPS	154 MRPS
PIM/c	66 ns	55 KRPS	14 MRPS
PIM/i	240 ns	65 KRPS	1 MRPS
PIM/k	100 ns	76 KRPS	1.2 MRPS
Multi-PSI	200 ns	125 KRPS	8 MRPS

しかし，我々はこの数字から何を読み取れるだろうか．append LIPSは複雑な並列推論マシンの性能のごく一部の側面しか表していない．この数値を高める努力が他のプログラムの実行の高速化にも貢献する可能性がある一方で，この数値をなりふり構わず高めたがために別のプログラムでの性能を落としてしまう危険性も否定できない．またユーザは，はたして「自分の解かせたい問題」が「PIMを用いることによって既存の計算機よりも高速に解かせることができるのか」を知ることができない．それにもかかわらず，この数値が一人歩きをし続け，PIMの設計者たちがこの数値ばかりを追いかける競争に駆り立てられ続けてきたように思えてならない．

一般の計算機の性能指標としては，たとえばSpec値などのように，ユーザが高い頻度で利用すると考えられる応用プログラムを幅広く集めたベンチマークを用いて速度を測定することが一般化している．PIMや論理型言語の処理系のベンチマークとしてもappend以外に

- パズルの求解などのトイプログラム
- 評価目的に特化した人工プログラム
- システム固有の応用プログラム

などを用いることが行われており，様々な興味深い分析も加えられている[12]．しかし，残念ながらPIMには，だれもがベンチマークとして納得す

ることのできる応用プログラムが揃っているわけではないというのが現状である．

第五世代コンピュータプロジェクトの特殊性は，新しいシステムと新しいアプリケーションを同時に生み出そうというプロジェクトであったことである．けっして，既存のシステムの枠組みの中で新しいアプリケーションを考えるという試みでも，既存のアプリケーションの性能を極限まで上げるために新しいシステムを開発するという試みでもなかった．ここに，新しいシステムで初めて可能となる新しいアプリケーションが開拓されるという大変魅力的な可能性が秘められていた一方で，新しいシステムができるまではアプリケーションをどう作ってよいか分からないし，アプリケーションがなければシステムはどう作ってよいか分からない，という膠着状況に陥る危険性も常に存在した．PIM の性能評価の問題はまさにこの現れにすぎない．

このような状況下で新しいシステムの性能評価を行うためには，将来利用が予測される応用プログラムの性能をその時点で最大限近似する方法の確立が必要となる．たとえば

1. 応用プログラムの構成要素となり得る基本処理/アルゴリズム(たとえばソート，表検索，ヒストグラムの作成など)を収集する
2. それらに，その時点で存在する応用プログラムの特性を反映した重み付けを行う
3. 新たな応用プログラムの登場とともに実情に則した改定を繰り返す

という方法が考えられるのではないだろうか．また

4. 既存のシステムと共通の指針を用いる

という視点も必要であろう．基本処理/アルゴリズムの少なくとも一部について，KL 1 による実現と C 言語なり FORTRAN なりによる実現との比較を可能にすることは，PIM の特性のより深い理解につながるとともに，新たなユーザ層の開拓にもつながると思われる．

6. おわりに

以上，第五世代コンピュータプロジェクトで行われたアーキテクチャ研究について概観してみた．プロジェクト全体を追ってみて気がつくのは，常にその時その時の研究の流行を巧みに取り入れていっていることである．WAM の採用しかり，RISC プロセッサの採用しかり，スヌープキャッシュの採用しかりである．悪く言えば，このような研究に終始していたと言えなくもない．このような理由から，独創的なアーキテクチャ構築を目指したプロジェクトを目指しながら，結局，既存技術からの飛躍的進展がどれほど果たせたのか，という疑問も残るかもしれない．

しかし，10 年間という長いようでとても短い研究期間で，単なるハードウェアの実験機ではなく，ユーザが実際に利用できる程度に実用的なシステムを，その要素技術の独創性はともかく，基本ソフトウェアも含めて構築したいということこそ，本プロジェクトで最も評価すべきことなのだと思う．

アーキテクチャの研究はシステムの研究である．まさにここに先進的アーキテクチャを目指す研究の難しさがある．目新しい要素技術を追求するあまり，ハードウェア，オペレーティングシステム，プログラミング言語，アプリケーションの整合性がとれないままに研究が終わってしまうようでは，結局アーキテクチャの革新につながる真の技術を生むことにはつながらない．独創的なアーキテクチャの構築といえども，一般ユーザの利用に耐えられるだけの実用性を持ったシステムを構築することを避けて通れるわけではない．

ICOT のしてきた研究を見ていると，ユーザに利用してもらうということにいかに重点をおいてきたかがよく分かる．研究中期に開発された MULTI-PSI という一見地味な実験機が，実は長い期間にわたって大きな役割を果たしたことは注目すべきことである．また，ユーザが新しいシステムをすぐに有効に利用できるようになるとは限らないという問題点を解決する試みとして，システム研究者とアプリケーション研究者でペアを組ませ，両者の協力のもとに応用プログラムの開発を行うという研究体制がとられたが，これもおおいに評価すべき工夫であり，類似のプロジェクトがぜひ参考にすべき方法だと思う．

新しいシステムのもとで，実用的な新しいアプリケーションが作られ，その特性と問題点が理解され，さらなる性能向上のために，新たなアーキテクチャが考案されていく，というのが本来の筋道であり，そのような気の遠くなるような繰り返

しによって，これまで計算機技術は発展してきたし，これからも発展していくのだと思う．やればできることをやっても研究にはならない，などといって，積み重ねるべきことを積み重ねないでいると，結局，CPU も OS も人の作ったものを借りてこなければならなくなるのである．

参 考 文 献

1) Taki, K. et al.: Hardware Design and Implementation of the Personal Sequentioal Inference Machine (PSI), Proc. Intl. Conf. on Fifth Generation Computer Systems (1984).
2) Warren, D. H. D.: Implementing Prolog-compiling Predicate Logic Programs, Research Reports 39 and 40, Dpt. of Artificial Intelligence, Univ. of Edinburgh (1977).
3) Yokota, M. et al.: A Microprogrammed Interpreter for the Personal Sequential Inferce Machine, Proc. Intl. Conf. on Fifth Generation Computer Systems (1984).
4) Nakashima, H. and Nakajima, K.: Hardware Architecture of the Sequential Inference Machine: PSI-II, Proc. of 1987 Symposium on Logic Programming (1987).
5) Warren, D. H. D.: An Abstract Prolog Instruction Set, Tech. Note TN-309, SRI (1983).
6) Taki, K.: Parallel Inference Machine PIM, Proc. Intl. Conf. on Fifth Generation Computer Systems (1992).
7) Shinogi, T., Kumon, K., Hattori, A., Goto, A., Kimura, Y. and Chikayama, T.: Macro-Call Instruction for the Efficient KL 1 Implementation on PIM, Proc. Intl. Conf. on Fifth Generation Computer Systems (1988).
8) Hirata, K. et al.: Parallel and Distributed Implementation of Concurrent Logic Programming Language KL 1, Proc. Intl. Conf. on Fifth Generation Computer Systems (1992).
9) Kimura, Y. and Chikayama, T.: An Abstract KL 1 Machine and its Instruction Set, Proc. 4th IEEE Symp. on Logic Programming (1987).
10) Ichiyoshi, N., Rokusawa, K., Nakajima, K. and Inamura, Y.: A New External Reference Management and Distributed Unification for KL 1, Proc. Intl. Conf. on Fifth Generation Computer Systems (1988).
11) Chikayama, T. and Kimura, Y.: Multiple Reference Management in Flat GHC, Proc. the 4th Intl. Conf. on Logic Programming (1987).
12) Kumon, K. and Hatazawa, H.: Evaluation of Parallel Inference Machines, Proc. Intrl. Conf. of Fifth Generetion Computer Systems (1994).

(平成 8 年 2 月 13 日受付)

小池 汎平（正会員）

1961 年生．1984 年東京大学工学部電子工学科卒業．1986 年同大学院工学系研究科情報工学専門課程修士課程修了．1989 年同博士課程単位取得退学．同年同大学工学部電気工学科助手，1991 年同講師，1995 年同助教授．1996 年通産省工業技術院電子技術総合研究所入所．この間 1994 年より 1996 年までマサチューセッツ工科大学コンピュータサイエンスラボラトリ客員研究員．工学博士．並列処理計算機の研究に従事．平成 4 年度元岡賞受賞．