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

2007－ARC－173 (17)
2007／6／1

[招待講演] 高並列アレイ型プロセッサIMAPCARのアーキテクチャとその技術展望

京 昭倫†

† NEC システムIPコア研究所
〒211-8666 川崎市中原区下沼部1753
E-mail: †s-kyo@cq.jp.nec.com

あらまし 画像の圧縮伸長や認識などのメディア情報処理は、演算性能に対する要求が急昇する一方で、機器の携帯性維持のためには低消費電力であること、また開発期間短縮や保守性向上のためには高い柔軟性が必要であり、こうした高演算性能、低消費電力、そして高い柔軟性といった相反する3つの要求をバランスよく満足できる現実的なアーキテクチャ解の一つがメニーコア構成のプロセッサである。本稿ではその一つの実現例であり、画像認識処理アプリケーション向けに製品化された、128個の4Way VLIWコアをワンチップに集積した高並列SIMDプロセッサIMAPCARの実現に至るまでの研究概要と共に、他分野への展開をも含めたその今後の技術展開について述べる。
キーワード SIMDプロセッサ，画像認識，メニーコア、メモリアクセスパタン，並列アルゴリズム

[Invited paper] Architecture of the Highly Parallel Array Processor IMAPCAR and its Technology Perspective

Shorin KYO†

† System IP Core Research Laboratories, NEC Corporation
1753 Shimonumabe, Nakahara-ku, Kawasaki, 211-8666 Japan
E-mail: †s-kyo@cq.jp.nec.com

Abstract Media information processing such as image compression and recognition are demanding increasingly high computational performance nowadays, while still require to be low power for maintaining mobility and high reliability, and as well to provide high flexibility for reducing the development and maintainance effort of its applications. To balance these trading-off items (performance, power, and flexibility), many-core multiprocessor is becoming one promising processor architecture choice. In this paper, the author briefly described the total design strategy of IMAPCAR, a highly parallel SIMD processor integrating 128 of 4-Way VLIW cores for image recognition applications, and as well its technology perspective toward widening its application fields.
Key words SIMD processor, Image recognition, Many-core, Memory access pattern, Parallel algorithm

1. まえがき

画像の圧縮伸長や認識などのメディア情報処理は、演算性能に対する要求が急昇する一方で、機器の携帯性維持のためには低消費電力であること、また開発期間短縮や保守性向上のためには高い柔軟性 (=高いプログラム可能性と容易性) が必要であり、こうした高演算性能、低消費電力、そして高い柔軟性といった相反する3つの要求をバランスよく満足できる現実的なアーキテクチャ解の一つとして、メニーコア・マルチプロセッサがある。一方、メニーコア・マルチプロセッサへの処理のマッピングには、逐次アルゴリズムに基づいて設計された既存アプリケーションの並列化という新たな作業を必要とし、そのハードルは決して低くない。本稿では、そうしたハードルへのチャレンジの一例であるIMAPCARのアプローチについて述べる。

128個のSIMD型4ウェイVLIWコアを集積した商用版プロセッサIMAPCAR [12] はそのプロトタイプ版であるIMAP-CE [7] [9] と同様に、「高並列SIMD型1次元プロセッサアレイ (LPA:Linear Processor Array)」に分類される構成を持つ。このLPAというアーキテクチャ構成を出発点にIMAP-CEそしてIMAPCARの研究では、「柔軟でありかつ実効性能の高い組込み向けLPA型画像認識プロセッサ」の実現を目標に、従来ではLPAによる並列化手法が明確でなかった各種画像認識処理

に対し、まず並列化手法の整理を行い、次にそれらの並列化手法の記述に適した高級言語を設計し、最後に当該高級言語コードの効率的実行に適した LPA 型プロセッサを開発した [9]。

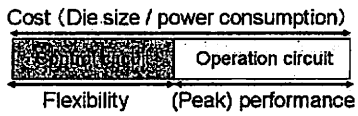
以降 2 章では、まず LPA 型高並列 SIMD プロセッサという構成を選択した理由、3 章では画像認識処理を応用ターゲットとした場合の LPA 向け並列化手法の整理分類、4 章ではプログラミング言語設計、そして 5 章では並列化手法のハードウェアサポートを考慮して決定された LPA 型高並列 SIMD プロセッサのの特徴について述べる。最後に 5 章で今後の技術展望について述べの後、6 章で全体をまとめる。

2. アーキテクチャの選択

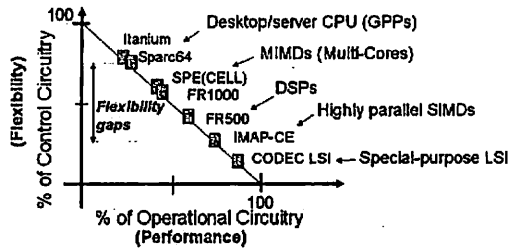
一般に、プロセッサを構成する回路は大きく制御関連回路と演算関連回路に分類できる。制御関連回路とは命令発行管理回路、外部インタフェース回路、そしてプログラムキャッシュおよびその関連制御回路等を指し、一方演算関連回路とはレジスタファイルを含むデータパス、データメモリまたはデータキャッシュおよびその関連制御回路等を指す。通常、演算関連回路がプロセッサの（ピーク）演算性能に寄与するのに対し、制御関連回路はプロセッサの柔軟性の向上に寄与するといえる。一方で図 1(a) が示すように、制御関連回路と演算関連回路の規模の合計が全回路コスト (ダイサイズ、消費電力) に相当する。そのため自ずと、コスト・演算性能・柔軟性の間にトレードオフ関係が発生する。図 1(b) は、幾つかの代表的なプロセッサ構成のカテゴリ毎の制御関連回路と演算関連回路の割合 (COR: Control versus Operational circuit Ratio) を、実際に当該カテゴリに属する商用プロセッサ例のダイ写真から類推してプロットしたグラフであり、当該カテゴリのプロセッサが有する柔軟性の一般的イメージとよく一致している。

図 1 からわかることは、どのようなカテゴリに属するプロセッサであろうと、コスト一定の制約の下では柔軟性と演算性能はトレードオフの関係にならざるを得ない。そうした中、高並列 SIMD プロセッサというカテゴリは、限定されたコストの中で演算性能を最大化でき、かつプログラム可能性をある程度工夫の余地がある形で残せる選択肢といえる。同様に LPA というシンプルな結合形態も、その低コスト性と画像処理との相性の良さ [5] からくる選択である。しかし一方でその代償として発生する柔軟性低下 (図 1(b))、すなわちプログラム開発におけるユーザ負荷の増大を、いかに工夫して緩和していくかが大きな課題となる。

それに向けた対策として IMAP-CE や IMAPCAR の研究では、まず応用ターゲットを (特に車載向け) 組込み用画像認識に定め、当該分野のアプリケーションを LPA にマッピングするための並列化方式を整理分類した。次に、それらの並列化方式の利用を前提にプログラミング言語そしてプロセッサを詳細設計し、提案並列化方式に沿って開発されたアプリケーションの性能を予見可能なものにすることで、プログラム並列化に際してのユーザ負荷軽減を目指した。以下 3〜5 章ではこれらのアプローチについてまとめるが、詳細は [9] を参照されたい。



(a) Relationship between performance, cost, and flexibility



(b) The Control versus Operational circuit Ratio (COR) observation of several representative processor LSI implementations

図 1 Relationship between performance, cost, and flexibility, and the Control versus Operational circuit Ratio (COR)

3. LPA 向け並列化方式

並列プロセッサのピーク性能は、単純に演算ユニット (PE) 当たりの性能× PE 数で計算される。しかし、より重要な「実効性能」は、アプリケーションを動作させた時の PE 稼働率で決まる。PE 稼働率はアプリケーションが持つ演算処理の各 PE へのマッピング方法、すなわち並列アルゴリズムの設計方法によって大きく変わる。一方、LPA 向け並列アルゴリズムの設計手法 (並列化方式) はこれまで整理されていなかった。

画像認識処理のアルゴリズムは一般に、低レベル処理(注1)、中レベル処理(注2)、高レベル処理に分類できる。このうち処理量が多いのは低レベル処理や中レベル処理の一部であり、それらが LPA による高速化の対象となるが、単に画素並列で処理を行えばよい種類の低レベル処理は別として、それ以外の、より複雑な画素参照パタンが必要となる低〜中レベル処理に対しては、個別の処理タスクに対し幾つか並列アルゴリズムが提案されている [1] [2] [4] 程度に留まっていた。

それに対し我々はまず、低〜中レベル処理が 7 種類の典型的な画素操作グループ [6](注3)(図 2) に分けられることに着目した。次に並列化方式設計の見通しをよくするため、「データ参照時のローカリティ」および「データ定義時の順序制約の有無」の違いをもとに、さらにそれらを下記 4 種類の画素参照定義パタンのカテゴリに分類整理した。

(注1)：例えば画素レベルでの雑音除去やひずみ補正などの前処理、エッジや色、動きなどの検出などの局所特徴抽出を行う処理
(注2)：例えばニューラルネットワークなどの局所特徴抽出の結果に対する判別処理やヒストグラム等の統計量計算、ステレオ計算、テンプレートマッチング等
(注3)：PO (Point Operation), LNO (Local Neighborhood Operation), GlO (Global Operation), SO (Statistical Operation), GeO (Geometrical Operation), RNO (Recursive Neighborhood Operation), and OO (Object Operation)

- LU(Local/Unconstrained)：局所近傍的画素参照、順序制約無し画素定義。PO と LNO が本カテゴリに属する。
- GU(Global/Unconstrained)：大域的画素参照、順序制約無し画素定義。GIO、SO そして GeO が本カテゴリに属する。
- LS(Local/Statically-constrained)：局所近傍的画素参照、静的順序制約有り画素定義。RNO が本カテゴリに属する。
- LD(Local/Dynamically-constrained)：局所近傍的画素参照、動的順序制約有り画素定義。OO が本カテゴリに属する。

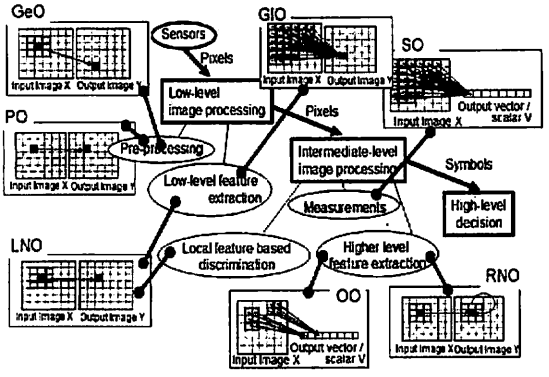


図 2　Distribution of the seven typical operation groups

LU～LD の各種画素アクセスパタンに対する LPA 向け並列化方式は、図 3 に示す LPA のワーキングモデルを想定した。すなわち、PE 全体のローカルメモリ領域の集合体としてのスペースを 2 次元メモリ面 (2-D memory plane) と呼び、その上を処理に応じ移動可能な PUL(Pixel-Updating-Line:画素更新線) (注4)の形状で表現した。

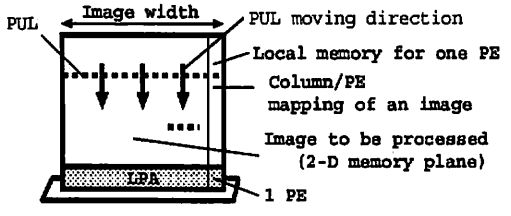


図 3　The 2D memory plane and PULl

LU は、データ定義における順序制約が存在しないことから、各 PE は他 PE と独立に処理を進められるため容易に LPA 上で並列化できるが、さらに隣接 PE が自分のローカルメモリから参照した画素を、そのまま隣接 PE 間結合経由で利用すれば、メモリアクセス回数を最小化できる。それには、全 PE が同一の Y 位置にある画素を処理対象とする必要があり、その結果として水平型 (row-wise) の PUL が得られる。例えば 2D 画像フィルタなら水平型 PUL で 2 次元メモリ面内の所定領域を一掃することで並列化される (図 3)。

(注4)：LPA の各 PE が同時に処理する画素位置 (=2 次元メモリ面上のアドレス位置) をつなげあわせて得られるライン

GU については、大域的画素参照の並列化方法がポイントとなる。リング状の一次元結合しか持たない LPA の場合、大域的データ移動またはデータ統合に伴う処理は、レーテンシーが PE 数分の繰り返しに比例するしてしまうことは避けられないが、複数のデータ移動あるいはデータ統合処理を同時に行なえるようにすれば、処理のスループットを改善できる。この場合、各 PE が相異なる Y 位置をアクセスしながらデータを PE 間で受け渡すため傾斜しかつ両端で折り返した形状で移動するシストリック型 (Row-systolic)PUL が得られる。一例として図 4 に、SO グループに属する画素濃淡ヒストグラム計算の LPA 向け並列化の様子を示す。なお、2 次元メモリ面内の Y 位置に、各 PE が同じ種類のデータをそれぞれ格納しているものとする。各 PE は同時に相異なる Y 位置のデータを参照しては、左から右 (あるいはその反対) へ、但し最右端からは最左端 (あるいはその反対) へと、次々とバケツリレー式に隣接 PE から参照データを受け取り、同時に自分の参照データも隣接 PE に渡すように動作させる。これにより、2 次元メモリ面内に散在する T 種類の情報が N 個の PE により、N × ((T-1) ÷ N+1) 回の PUL 移動 (但し T=N=6) で、全 PE に収集される。

LS の並列化には、静的に存在する画素更新の順序制約を遵守の上、2 次元メモリ面内のデータアクセスを並列に行なう必要がある。それには、予め順序制約を x 方向と y 方向に分解の上、X 方向の順序制約は各 PE の「動作・動作しない」で充足させ、一方 Y 方向の順序制約は各 PE によるアクセスアドレス値の増減タイミングの制御で充足させる。その結果、順序制約の内容にもよるが、多くの場合では傾斜型 (slant-systolic) の PUL が得られる。図 5 に、左上半分に位置する画素が更新済みになれば中心画素が更新可能となる順序制約での例を示す。

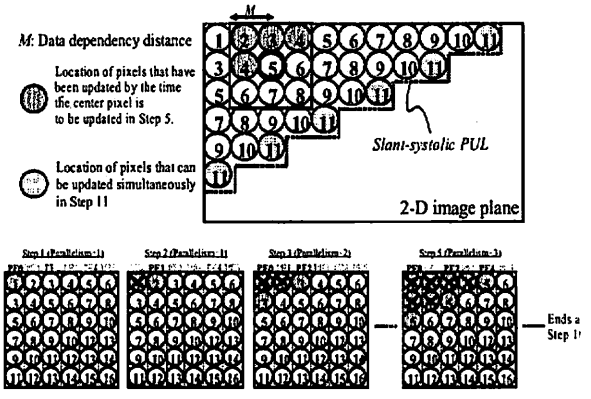


図 5　Example of *slant-systolic PUL*

最後に LD の並列化には、処理が進むにつれて動的に変化する画素更新の順序制約を遵守の上、2 次元メモリ面内でのデータアクセスを並列に行なう必要がある。これにはアクセスアドレス情報を一時格納するソフトウェアスタックを 2 次元メモリ面内の所定領域に各 PE 固有で用意し、各 PE が当該スタックをまず「1) 開始アクセスアドレス情報 (seed) のプッシュ」に

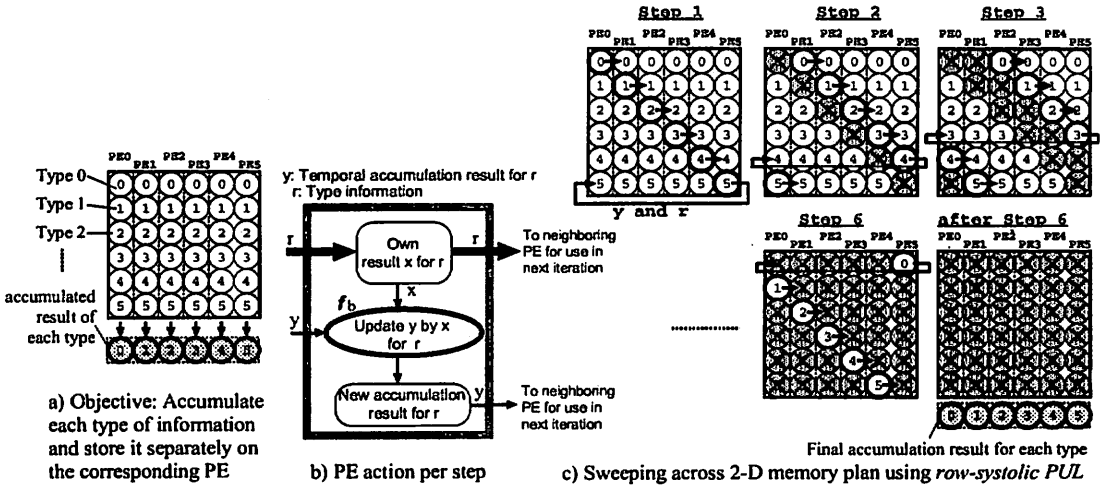


図 4 Example of *row-systolic PUL*

より初期化した後、「2) アクセスアドレスの伝播(注5)」の動作を全スタックが空になるまで繰り返す。これにより、任意の画素更新の順序制約を守りながら PUL を移動させることが可能となる (Autonomous PUL、動作例は図 6 参照)。

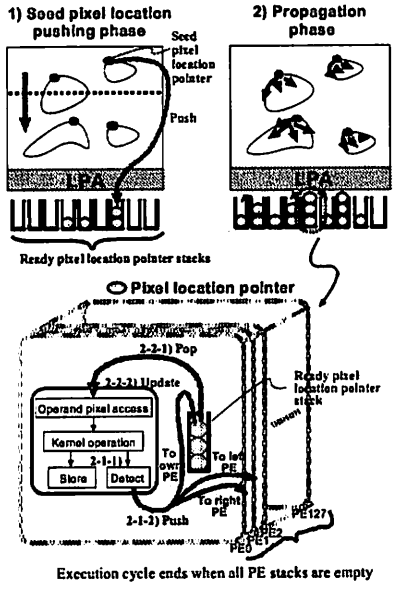


図 6 Mechanism of *autonomous PUL*

図 7 に 7 つの画素処理グループと LU~LD の各メモリアクセスパタンに対応する LPA 向け並列化方式 (4 種類の PUL) の関係を示す。なおアプリケーションレベルでは、各並列化方式は単独よりも、組合せて利用されることが多いと予想される。例えば 2D-FFT は水平型 PUL(列単位の 1D-FFT)、シストリック型 PUL(列と行を転置) そして最後にもう一度水平型 PUL(列単位の 1D-FFT) の組合せで並列化できる。以降、便宜上このように PUL に着目した並列化方式を「ライン方式 (line methods)」と呼ぶ。ライン方式利用時に期待される各画素操作カテゴリの演算オーダーは [9] を参照されたい。

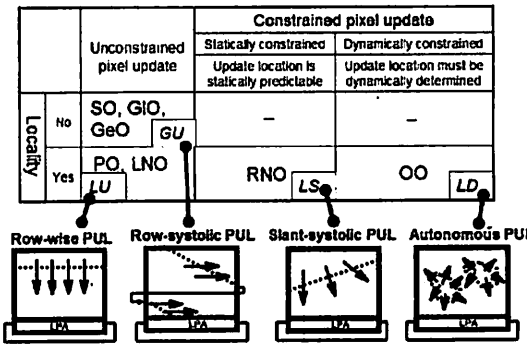


図 7 Correspondence between pixel operation groups, memory access pattern categories, and parallelizing techniques

4. プログラミング言語の設計

ライン方式に対するソフト側およびハード側からのサポートには、所定 (LD の場合は任意) の形状とタイミングで動作する PUL を、簡潔に記述できるプログラミング言語、そして当該記述を効率よく実行できる LPA 型プロセッサの設計が重要となる。そうした背景の下に設計された拡張 C 言語 1DC(One Dimensional C) [9]。の C からの主な拡張を以下にまとめる。

- 画像の列数と同数のスカラー要素の集合であり、従来の C の算術論理演算記述で全要素が一斉に演算対象となる変数を宣言する変数宣言子 sep

(注5):次の 2-1 と 2-2 からなる伝播動作:2-1:隣接領域からの実行開始可能画素位置の新規検出とそれらのスタックへのプッシュ、2-2:画素位置情報のポップと当該位置の画素の更新。

- sep 条件式を評価し、結果が 0(命令を実行しない) と非 0 (命令を実行する) の PE に PE 群を二分する分岐構文 `mif...melse, mwhile, mfor, mdo`
- E_{sep}、A_{sep}、E_{sca} をそれぞれ *sep* 式、*sep* データ配列、そしてスカラー式とすると、:>E_{sep} と:<E_{sep} で左と右隣 PE 上 E_{sep} 要素の参照、$A_{sep}[E_{sep}]$ でインデックス・アドレッシング、E_{sep}:[E_{sca} :] で E_{sep} の第 E_{sca} 番目要素の参照、そして:&& E_{sep} と :|| E_{sep} で E_{sep} の全要素の論理積と和をそれぞれ指定する *sep* 演算子

上記拡張仕様によって実現された 6 種類の拡張演算記述 (図 8) とライン方式からの要求との対応は表 1 に示す通りである。

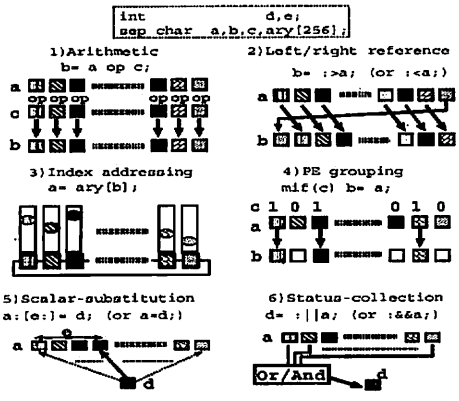


図 8 Six primitive 1DC syntax forms

表 1 Correspondence between PUL types and the 1DC syntax

PUL type	Row-wise	Row-systolic	Slant-systolic	Auto-nomous
1) Arithmetic	X	X	X	X
2) Left/right ref.	X	X	X	X
3) Index addr.	-	X	X	X
4) PE grouping	-	-	X	X
5) Scalar-subs.	-	-	X	X
6) Status-collect.	-	-	-	X

5. LPA 型高並列 SIMD プロセッサの構成

IMAPCAR チップ全体は大きく 1) 外部インタフェース部 (EXTIF)、2) プログラム・データキャッシュを有し命令発行や全体制御、そしてスカラー型命令の実行等を担当する制御プロセッサ (CP)、そして 3)128 個の 4Way VLIW コア (PE アレイ)、という 3 つのブロックに分けられる。CP はクロック毎、最大 4 命令を同時に発行でき、そのうち最大 1 命令は自分自身向け、そして最大 4 命令までを PE 向けに発行・ブロードキャストできる。各 PE は 2KB のローカルメモリ (IMEM)、28 個の 8bit 汎用レジスタ等を持ち、また VLIW 実行機構によりクロック毎最大 4 命令を同時処理可能である。CP と PE のパイプラインステージ構成を図 9 に示す。

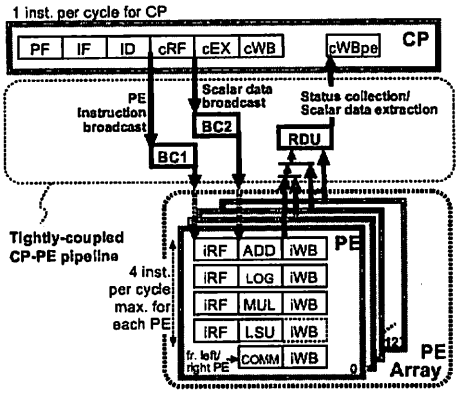


図 9 CP and PE array pipeline stages

128 個の PE は 1 次元状に互いに接続され、かつ最右端 PE は最左端 PE と接続されたリンク状結合をとる。CP からの命令ブロードキャストを効率よく実現するために、PE アレイは PE8 と呼ぶ 8 個の PE からなるブロックにまとめられ、命令やデータのブロードキャストは、CP からまず計 16 個存在する PE8 ブロックに対して行い、次に PE8 内で各 PE へ分配するという階層構造とした。図 10 にチップ全体のブロック構成とダイ写真、表 2 にチップの諸元を示す。また、1DC での 6 種類の拡張記述を効率よく実行するのに対応したハードウェア構成上の特徴を以下にまとめる。これらの工夫の下、1DC 記述に対するコンパイラ生成コードを用いた画像処理タスクのベンチマークから IMAPCAR が、P4 の数十分の 1 の消費電力でその数倍の処理性能を実現できることが報告されている [13] [8]。

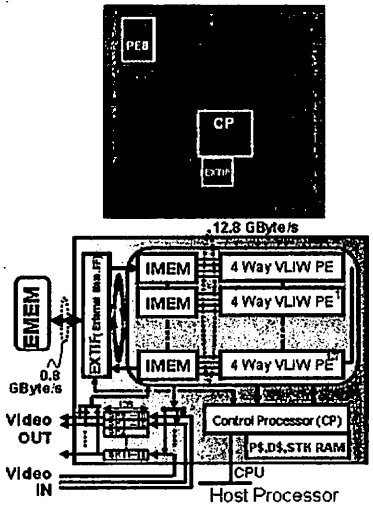


図 10 IMAPCAR block diagram/die photo

(1) sep 型データの演算 (arithmetic) : VLIW 実行形式により、クロック毎最大 4 つの PE 命令を同時実行可能とした。

(2) PE 間データ転送 (left/right reference) : 隣接 PE のレジスタ間の直接結合を設け、PE 間のスカラーデータ参照を

表 2 IMAPCAR chip specification [12]

Number of PE	128
Processor clock	100MHz
Peak Performance (8bit operation)	100GOPS
Process technology	0.13um CMOS
Power supply	1.2V (Core), 3.3V(I/O)
Power dissipation	< 2W
Operation temperature	-40 to +85 deg. C
Package	500pin, ABGA

シングルサイクルで可能にした。

（3） インデックスアドレッシング動作 (index addressing)：各 PE に独立した RAM を持たせ、各 PE が相異なるメモリアドレスへアクセスすることを可能にした。

（4） PE グルーピング動作 (PE grouping)：幾つかの単純な論理演算を組み合わせた布線論理命令を PE の RISC 命令セットに追加することで本動作の効率化を実現した。

（5） スカラー代入・抽出・データ集計動作 (scalar-substition, status-collection)：CP と PE 間の密結合パイプライン構造でスループット 1 を実現した。

6. 今後の技術展望

IMAP-CE そして IMAPCAR の研究では、適用分野を画像認識処理に限定することで、対応すべきメモリアクセスパタンをある程度絞り込むことができ、それにより並列化方式を整備できた。しかしながら今後、適用分野を拡大していこうとした場合に、下記に示す幾つかの技術課題のクリアが必要になると予想される。

（1） 負荷不均衡の問題：画像データが処理対象の場合は発生頻度が低いが、2 次元メモリ面内での処理対象データ分布が不均一の場合、特定 PE に演算負荷が集中する可能性が大きい。

（2） 制御並列性への対応：現状 PE アレイ全体が CP の発行する単一の制御シーケンスに従うため、データ並列性よりも制御並列性が顕著なアプリケーションに対応できない。

一方、[3] や [11] で指摘されているように、SIMD 型並列計算に本質的な PE 毎による「データの自律性」以外には、表 3 に示す幾つかの自律性の追加が可能である。そこで、そうしたアプローチにより高並列 SIMD プロセッサの適用可能なアプリケーション範囲を広げていくことが考えられる。

表 3 PE autonomies

a) 動作の自律性	PE 毎による命令実行・不実行の選択
b) addressing の自律性	PE 毎による異なるアドレスへアクセス
c) 通信/結合の自律性	PE 毎による異距離 PE とのデータ交換
d) 命令の自律性	PE 毎による相異なる命令の実行

但し、こうした PE 自律性の追加はハードコスト増を伴うため、サポートすべき並列化方式からの要請の強さとのバランスを考慮の上で行なう必要がある。IMAP-CE や IMAPCAR では、a)〜d) のうちコスト要求の低い a) の「動作の自律性」や b) の「アドレッシングの自律性」のハードサポートを実現したことで、各種形状やタイミングで移動する PUL を可能にした。今後は、負荷不均衡の問題の解決には通信/結合の自律性の追加、制御並列性への対応には命令自律性の追加が有効と考えられるが、その際に、いかに高並列 SIMD プロセッサが持つ優れたコスト性能比を維持しながらそれらを実現していくかが課題となろう [10]。

7. 終わりに

コスト制約の下でアプリ側が要求する高い演算性能を提供しつつも、そのトレードオフとして発生する柔軟性低下をいかに (見かけ上) 低減できるかは今後、あらゆるメニーコア型マルチプロセッサにとり大きな研究課題となろう。本稿では、そうした課題に対する取り組みの一例として、LPA 型高並列 SIMD プロセッサである IMAPCAR の場合について、主にアーキテクチャ選択指針および並列化方式の整備手法について述べた。また今後の技術展望として、低コストで PE 自律性を拡大していくという技術的方向性に言及した。今後は、こうした方向性での研究開発が活発になっていくことに期待したい。

文 献

[1] A.L.Fisher and P.T.Highnam, "Computing the Hough Transform on a Scan Line Array Processor", IEEE Trans. on PAMI, Vol.11, No.3, pp.262.265, 1989.

[2] P.E Danielsson, "Parallelsim in Low Level Vision Algorithms and Architectures", Proc. of the 6th Scandinavian Conference on Image Analysis (SCIA), Finland, pp.25.31, 1989.

[3] T.J.Fountain, "Introducing local autonomy to processor arrays", In: H.Freeman (ed.) Machine vision: algorithms, architectures and systems, Academic press, pp.31-56, 1988.

[4] D.Helman, J.JaJa, "Efficient Image Processing Algorithms on the Scan Line Array Processor", IEEE Trans. on PAMI, Vol.17, pp.47.56, Jan. 1995.

[5] P.P.Jonker, "Why linear arrys are better image processors", Proc. of Int. Conf. on Pattern Recognition, Vol.3, pp.334-338,1994.

[6] E.R.Komen, "Low-level Image Processing Architectures", Ph.d Thesis, Delft University of Technology, Netherlands, 1990.

[7] S. Kyo et al., "A 51.2-GOPS Scalable Video Recognition Processor for Intelligent Cruise Control Based on a Linear Array of 128 Four-Way VLIW Processing Elements", IEEE Journal of Solid-State Circuits, Vol.38, No.11, Nov. 2003.

[8] S. Kyo, S.Okazaki, "Mapping In-Vehicle Image Recognition Tasks on the IMAPCAR Highly Parallel Image Processor", in Proc. of ITS Congress, Oct. 2006.

[9] S. Kyo et al., "An Integrated Memory Array Processor Architecture for Embedded Image Recognition Systems", IEEE Trans. on Computers, Vol.56, No.5, pp.622-634, May 2007.

[10] S. Kyo et al., "A Low Cost Mixed-mode Parallel Processor Architecture for Embedded Systems", to appear in Proc. of Internation Conference on Supercomputing, June 2007.

[11] P.J.Narayanan, "Processor Autonomy on SIMD Architectures", in Proc. of the ACM International Conference on Supercomputing, pp.127-136, 1993.

[12] S.Okazaki, S. Kyo, "IMAPCAR: A Highly Parallel Integrated Memory Array Processor for In-vehicle Image Recognition Applications", in Proc. of ITS Congress, Oct. 2006.

[13] K.Sakurai et al., "Implementation of Overtaking Vehicle Detection Using the IMAPCAR Highly Parallel Image Processor", in Proc. of ITS Congress, Oct. 2006.