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

2008－ARC－176 (8)
2008－EMB－ 7 (8)
2008／1／15

# 動的切り替え可能な SIMD/MIMD 型プロセッサにおける MIMD コアの低コスト実現法

野本 祥平　京 昭倫　古賀 拓也　岡崎 信一郎
NEC システムIPコア研究所 〒211-8666 神奈川県川崎市中原区下沼部1753

近年，自動車における，画像認識を用いた安全システムの普及が始まっている．これらのシステムでは，リアルタイムな画像認識を熱設計が厳しい車載環境で行う必要があるため，画像認識プロセッサには，高い演算性能のみならず低消費電力性が求められる．また，複雑化する画像認識アルゴリズムに対応するため，様々な粒度の並列性を活用でき，なおかつ高いコスト性能比を実現することも必要となる．こうした中で，著者らは，高並列 SIMD プロセッサによる低コスト・高性能・低消費電力と，MIMD プロセッサによる柔軟な並列性の活用の双方の性質を併せ持つプロセッサの実現を目標に研究開発を進めてきた．本稿では，上記性質を持つプロセッサの実現に向け，SIMD プロセッサの Processing Element(PE)の回路資源に着目し，複数の PE を組み合わせて1つの MIMD コアを実現する実装方式を提案・評価する．具体的には，1)PE のメモリ・レジスタファイルを流用したキャッシュ機構，2)SIMD/MIMD コアにおけるデータパスの共有，3)PE 演算器群を流用した浮動小数点パス，を実現した．これらの実装方針をとることにより，動的切り替え可能な SIMD/MIMD 型プロセッサを実現するための追加コストを，MIMD コア全体の回路規模の約 10%までに抑えることが出来ることを示した．これにより，低コスト・高性能・低消費電力・柔軟性を併せ持つ画像認識プロセッサの実現可能性を示した．

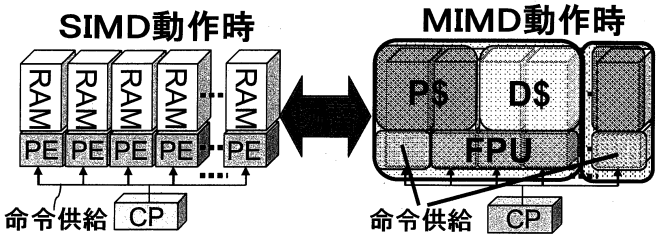
# Low-Cost Implementation of the MIMD Core For a Runtime Mode Switching SIMD/MIMD processor

Shohei NOMOTO　Shorin KYO　Takuya KOGA　Shinichiro OKAZAKI
NEC　System IP Core Research Laboratories, 1753 Shimonumabe Nakahara Kawasaki, Kanagawa 211-8666, Japan

Safety systems that use image recognition are starting to be used commercially in cars. Because such systems must perform real-time image recognition in a car environment that imposes severe heat design requirements, the image recognition processor has to realize both high operation performance and low power consumption. Moreover, in order to support increasingly complex image recognition algorithms, the system must support various granularity of parallelism as well as achieve high cost performance. Against this backdrop, the authors have been working on the development of a system that achieves low cost, high performance, and low power consumption through the use of a highly parallel SIMD processor, as well as flexible parallelism through the use of a MIMD processor. This paper proposes and evaluates a method for realizing a MIMD core that combines multiple processing elements (PE), with particular focus on the circuit resources reuse of the PEs of the SIMD processor Concretely, a three-pronged approach was employed: 1) organizing of a cache mechanism using the memories and register files of PEs, 2) sharing the data paths between the SIMD and MIMD cores, and 3) realizing a floating point path by using the execution units of the PEs. Through this approach, the additional footprint cost required for the realization of a dynamically switched SIMD/MIMD processor was successfully reduced to approximately 10% of the total circuit area of the MIMD core. This demonstrates the feasibility of highly competitive image recognition processors that combine low cost, high performance, low power consumption, and flexibility.

## 1． はじめに

高齢化が進む中で，より安全・安心な車システムへのニーズが高まっている．そうした中で，画像を用いた自動車の安全走行システムへの関心が高まっている[1][2]．これらのシステムでは，リアルタイムな画像処理/認識が必要となるため，高い演算性能が必要とされる．また車載環境では，厳しい熱設計をクリアすることが必要となる．さらには多様化する画像認識アルゴリズムに対応するため，高いプログラマビリティも要求される．このため低い電力消費で，高い演算性能を実現するプログラマブルな画像認識プロセッサが強く求められている．このようなプロセッサとして，例えば VCHIP(Vision/Video Chip)[3]では，画像認識で一般的に使われる処理(エッジフィルタ等)を専用ハードウェアとして実装している．専用ハードウェアを用いることにより，多様なアルゴリズムへの柔軟性が失われるが，想定済みの定型的処理に対しては高い演算性能を低コストで実現できる．また Visconti(Vision based Sensing, CONTrol, and Intelligence)[4]は3つのコアを持ち，それぞれのコアには，2つのヘテロジニアスな SIMD 演算器(8way×8bit 幅)を持つ．SIMD 演算器により画素レベルの並列性を活用し，3つのコアによりスレッドレベルの並列性を活用できることから，VCHIP と比べるとコスト性能比はやや悪化するが，様々な画像認識アルゴリズムに柔軟に対応できるようになる．




※PE=Processing Element, CP=Controll Processor
P$=Instruction Cache, D$=Data Cache,
FPU=Floating Point Unit

図1. 動的切り替え可能な SIMD/MIMD 型
プロセッサの概要図

こうした中で，我々はこれまで，画像認識プロセッサ IMAP(Integrated Memory Array Processor)[5]の研究開発を行ってきた. IMAP では，メモリと密結合した PE(Processing Element)を多数配置する高並列な SIMD 型アーキテクチャを採用している. 個々の PE レベルでは，Visconti 等での SIMD 演算器と比べるとコスト性能比は劣る. しかし，独立したメモリからのデータ供給の自由度が高まり，高並列での利用に，より適した構成といえる. また全 PE に同一の動作をさせることにより，制御に要する回路規模を抑制できる. その分，演算に用いる回路規模(PE数)を増やせるため，プロセッサ全体では優れたコスト性能比を実現することが可能となる. また，こうした構成をとることで，高い演算性能を実現するのに，動作周波数を上げるのではなく，並列性を上げるというアプローチを容易にとることが出来る. これらにより，高い演算性能と低消費電力を両立させることが可能となる.

一方で SIMD 型アーキテクチャは，活用できる並列性が限定されるという課題もある. 近年，画像認識アルゴリズムの多様化・複雑化に伴い，サイズが異なる画像領域を処理する場合等が増えている[6]. このような場合には，各画像領域を処理するPEの処理時間にバラツキが生じ，高並列 PE アレイの稼働率が下がり，性能が律速されてしまう. このため我々は，上記の課題を解決すべく，動的切り替え可能な SIMD/MIMD 型プロセッサ（XCコア）[7]の開発を行ってきた. 図1に示すように，XCコアでは，複数の PE の資源を流用することにより，1つの独立に動作するプロセッサ(浮動小数演算器付き)を実現している. これにより，画像領域毎に処理時間が異なる画像認識アルゴリズムに対しても柔軟に対応することができる. またPEの資源を流用することにより，動的切り替え可能な SIMD/MIMD 型プロセッサを実現するためのコストを低く抑えることが出来る.

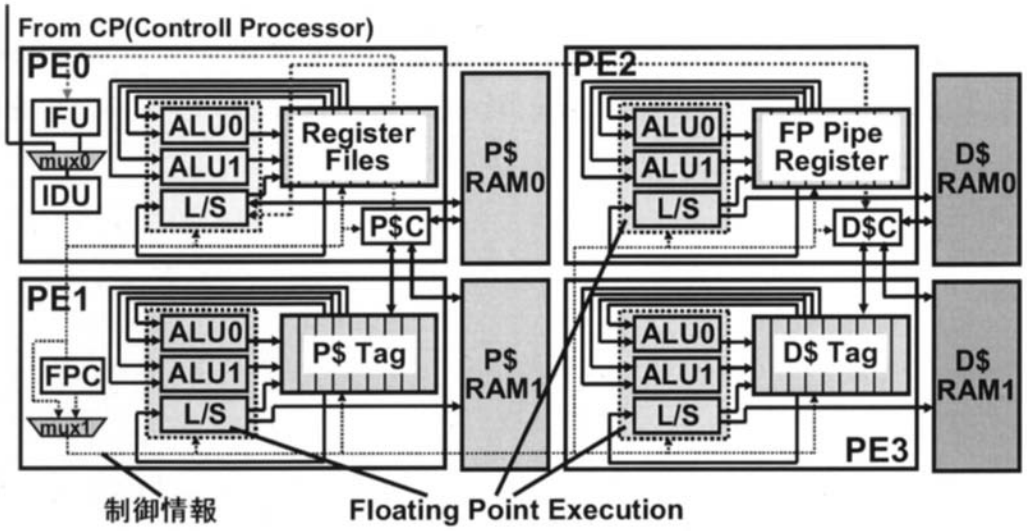
本報告では，XC コアを低コストに実現する上で，最も重要となる動的切り替え可能な MIMD コアの構成方法について詳細に報告する. 以下，2章では，MIMD コアのアーキテクチャについて詳細に述べる. 3章では，MIMD コアを RTL 実装し，SIMD/MIMD を切り替えるのに要する追加回路規模について評価する. 最後に4章では，まとめと今後の課題について述べる.

## 2. MIMD コアのアーキテクチャ

本章では，動的切り替え可能なSIMD/MIMD型プロセッサを構成する MIMD コアのアーキテクチャについて詳細に述べる. 図2に，提案する MIMD コアのブロック図を示す. 図2に示すように，本 MIMD コアは，SIMD 動作時の 4 つの PE のメモリブロック，レジスタファイル群，データパス，演算器群を流用することにより実現されている. これにより，SIMD 型プロセッサに MIMD コアの機能を新規に追加した場合に比べて，追加となる回路規模を大幅に抑えることを可能としている. 以下では，低コストに複数の PE から切り替え可能な MIMD コアの実現に向けた，1)命令・データキャッシュ，2) 制御・データパス，3)浮動小数点パスの構成について述べる.

### 2. 1. 命令・データキャッシュの実装

ここでは，MIMD コアにおける命令・データキャッシュの実装について述べる. MIMD コアにおいて，最も大きな回路規模を占めるのがキャッシュ機構である. このため低コストな MIMD コアを実現するに当たり，その回路規模を抑えることが重要となる. 以下では，MIMDコアにおいて，低コストに命令・データキャッシュを構成する実現法について述べる. まず命令・データキャッシュは，以下の3つの部分から構成される. 1）命令およびデータを格納するキャッシュメモリ部，2）キャッシュの制御情報を格納するキャッシュタグ部，3）キャッシュ全体を制御するキャッシュ制御部である. これら3つのうち特に，キャッシュメモリ部とキャッシュタグ部が回路規模の大部分を占める. 一方キャッシュ制御部は制御ロジックのみであるため，必要とする回路規模は，それほど大きくはならない. このためキャッシュメモリ部とキャッシュタグ部の2つを，SIMD/MIMD 型プロセッサの双方で利用できるリソースから構成し，キャッシュ制御部は MIMD 用に新たに追加する構成




**※IFU=Instruction Fetch Unit, IDU=Instruction Decode Unit**
**FPC=Floating Point Controller, P$C=Program Cache Controller**
**D$C=Data Cache Controller**

図2. 4PEの再構成による MIMD コアの構成

とした.

まず, キャッシュメモリ部について説明する. キャッシュメモリ部を実現するに当たり, SIMD 動作時に, PE が利用するローカルメモリに着目した. IMAP アーキテクチャでは SIMD 動作時に, 各 PE が処理するデータは, ローカルメモリに格納されていることを想定している. また, 高い性能を実現すべく, データ処理とデータ転送を同時に行なえるようにするためのバッファ領域も必要となる. このため, 通常は数Kバイト以上の大きな容量の SRAM が, PE のローカルメモリとして使われている. 一方 MIMD コアにおけるキャッシュメモリ部は,多数の命令およびデータを格納し, 短レイテンシでの読み書きが要求されるため, 大容量の SRAM を必要とする. しかし, 大容量の SRAM は大きな回路面積を占めることになる.

そこで今回の実装では, 複数 PE の持つローカルメモリを, MIMD 動作時にはキャッシュメモリ部として利用することにより, 追加する回路規模を最小限に抑えることとした. 一般に SRAM を制御するための信号線は, 読み書き制御/アドレス/データ線と少数である. このため, PE の持つローカルメモリ(SRAM)を共有するために追加すべき資源は, PE とキャッシュ制御部からの信号線を切り替えるためのセレクタのみで済む. したがって, SRAM を共有することによって発生する回路オーバヘッドは僅かと見積もられる. 図2に, MIMD コアが, PE の持つローカルメモリ(SRAM)を利用する構成を示す. 今回の実装では, 図2に示すように, PE0/1, PE2/3 のローカルメモリをそれぞれ, 命令キャッシュのメモリ部(P$ RAM0/1), データキャッシュのメモリ部(D$ RAM0/1)として利用する構成とした.

次に, キャッシュタグ部について説明する. キャッシュタグ部は主に, 以下の2つ, 1)アクセスアドレスの一部(タグ)を格納する Flip-Flop(F/F)部, 2) 要求されたタグを出力するセレクタ部, で構成できる. F/F 部は多数のタグを格納すると共に, F/F 自体が大きな素子であることから, 実現コストは高くなってしまう. また, セレクタ部は F/F 部に格納された多数のタグから, 要求された1つを選択する必要があるため, 多数のセレクタが必要となる. そのため, セレクタ部と F/F 部とを新規に追加することは, 回路規模の大幅増加が予想される. 以上のことから, 回路規模を抑えるために, PE の持つリソースによってキャッシュタグ部を構成する実装とした.

そこで我々は, PE の持つレジスタファイル群に着目した. レジスタファイル群は, PE の演算結果の一時格納場所として機能し, 指定された場所へのデータ書き込み/指定された場所からのデータ読み出しを実現する. これらの機能は, キャッシュタグ部に要求される機能と同様である. このため, 今回の実装では, PE のレジスタファイル群を, キャッシュタグ部として利用する実装とした. またレジスタファイル群を制御するための信号線は, 読み出しアドレス/ 書き込みアドレス/書き込みデータ線と少数である. このため, レジスタファイル群を共有するために追加する資源は, PE とキャッシュ制御部からの信号線を切り替えるためのセレクタのみで済み, 共有することによって発生する回路オーバヘッドは僅かである. 今回の実装では, 図2に示すように, PE1/2 のレジスタファイルをそれぞれ, 命令およびデータキャッシュのキャッシュタグ部として用いる構成とした.

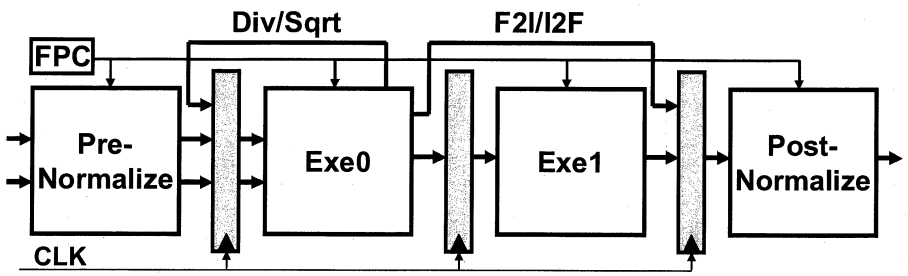



図3. 浮動小数点パイプラインの概要

### 2. 2. 制御・データパスの実装

ここでは，まず MIMD コアにおける制御パスについて述べる．制御パスは，主にMIMDコアの命令列の実行順序を制御する部分（命令制御部），命令キャッシュから得られた命令列をデコードし演算器の制御情報を生成する部分（命令デコード部）から構成される．これら2つのうち，命令制御部が必要とする回路規模は少ないと想定され，PE において代替できる要素が見当たらなかったため，新規に追加することとした．今回の実装では，図2に示すように，命令制御部(IFU)を PE0 に追加する構成とした．

一方で，命令デコード部は命令制御部に比べて回路規模が大きく，共有化することによるメリットは大きいと判断した．そこで SIMD 実行時の PE と，MIMD 実行時の MIMD コアが実行する命令セットの大部分を共通化することとした．これにより，PE が持っている命令デコーダ部(IDU)を，MIMD コアもほぼ共有することが出来る．また演算器群への制御情報がほぼ共通となることから，命令のデコード結果を保存するバッファも共通化することができ，回路規模の増加を抑えることが可能となる．今回の実装では，図2に示すように，PE0 に命令制御部(IFU)と SIMD/MIMD で共有する命令デコーダ部(IDU)へ供給する命令列を選択するセレクタ(mux0)を，新規に追加する実装とした．このように，共有化によるオーバヘッドは命令列のセレクタ追加によるものに留まり，回路規模への影響は僅かである．

次に，データパスについて述べる．前述したように，SIMD/MIMD 実行の双方で用いる命令群の大部分を共通化することにより，データパス自体も大部分が共通化されている．また共有化によるオーバヘッドは，先ほど述べた命令列のセレクタで済む．一方で，共通化されていないデータパスも存在する．MIMD 実行時にのみに使用する，分岐制御/スタック制御/キャッシュ L/S 制御部などは共通化していない．これらは，いずれも制御部であることから，代替リソースが PE 側に見当たらず，追加する回路規模も僅かであると判断したため，MIMD 用に新規に追加することにした．今回の実装における MIMD コアのデータパスは，図2に示すように，PE0 の演算器群(ALU0，ALU1，L/S)と汎用レジスタ群(Register Files)をデータパスとして用いる構成とした．

### 2. 3. 浮動小数点パスの実装

2. 2で述べたように，MIMD コアが使用するデータパスは，PE0 のデータパスのみとなっている．このため，PE1～PE3 の演算器群が有効活用されずにいる．これらの演算器群については，設計の初期段階において，以下の二通りの活用法が考えられた．1)MIMD コアの並列実行性能の向上，2)浮動小数点パスの追加，である．いずれの活用法においても，MIMD コアの性能向上は可能であったが，今回の実装では，SIMD/MIMD の切り替えの低コスト化に主眼を置き，浮動小数点パスを選択することとした．1)を実現する場合には，より広い命令フェッチ幅や，PE のデータパスの大幅な拡張が必要となり，追加する回路規模の大幅な増加をもたらす可能性があった．一方，2)を実現する場合には，浮動小数点パスを PE のデータパスに追加するのみで，PE のデータパスへの変更は最小で済む．以上のことから，PE1～PE3 の演算器群を用いて，浮動小数点パスを実現する構成とした．

浮動小数点パスの実現により，これまで，固定小数点で処理を書き換えざるを得なかったケースでもソースコードの書き換えが不要となり、プログラマの労力を軽減することが期待される．また，画像認識アルゴリズムの多様化に伴い，より広いレンジの値を扱う必要がある場合に，柔軟に対応することができるようになる．さらには，画像認識以外のアプリケーションへの適用可能性も広がる．以下では，浮動小数点パスの実装について述べる．

浮動小数点パスでは，IEEE754[8]で規定されたビットパターンに対して，表1に示す演算を実行する．四則演算に加えて，平方根の計算，前述したビットパターンと整数値の変換命令を実行する．また，表1には，各命令のレイテンシとスループットも示す．

浮動小数点演算ではレイテンシが生じるため，効率的な演算を実現するためには，各命令の実行を可能な限りパイプライン化することが必要となる．このため今回の実装では，図3に示すように，浮動小数点パスを4段のパイプラインとして実現した．このパイプラインは，小数点位置を合わせるステージ(Pre-Normalize)，各命令の演算を実行するステージ(Exe0，Exe1)，演算結果の小数点位置を調整するステージ(Post-Normalize)から構成される．ここでは浮動

迂回するパスを設け，効率的な演算の実現に努めた．ただし除算・平方については，パイプライン処理するためのリソースが確保できなかったため，Exe0 ステージの演算器を繰り返し利用して，演算結果を求める実装とした．また，浮動小数点パスと PE0 データパスが用いるレジスタファイルを共通化する構成とした．これにより，PE0 データパスの変更を最小限に，浮動小数点パスを実現することが出来る．

表1．浮動小数点命令のレイテンシとスループット

| 命令種 | Latency | Throughput |
|---|---|---|
| Adder | 4 | 1 |
| Sub | 4 | 1 |
| Mul | 4 | 1 |
| Div | 26 | 26 |
| Sqrt | 26 | 26 |
| Int▶Float | 2 | 1 |
| Float▶Int | 2 | 1 |

一方，パイプライン動作を実現するに当たっては，同時動作する多数の演算器群が必要であった．このため今回の実装では，図2に示すように，PE1～PE3 の演算器群と PE3 のレジスタファイルを利用して，浮動小数点パスを実現した．また，図2に示すように，浮動小数点パスの制御を実現する制御回路(FPC)を PE1 に設けた．MIMD 動作時には，FPC からの制御信号を PE1～PE3 に供給することにより，PE1～PE3 の演算器群とレジスタファイルを共有する構成とした．また回路規模の観点からは，特に図3の Exe0 と Exe1 で使用する加減算・乗算器と，Post(Pre)-Normalize で使用するシフタ群が大きな回路規模を占めるため，これらの演算器群を共有化した．これにより，追加する回路規模を大きく抑えることが出来た．

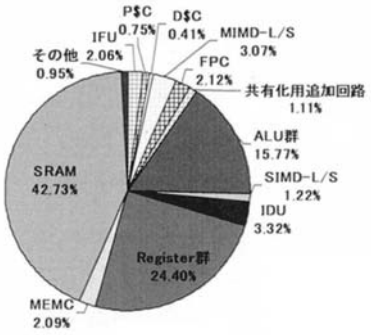
## 3．論理合成による回路規模の評価

本章では，動的切り替え可能な SIMD/MIMD 型プロセッサの MIMD コアを RTL 記述し論理合成することにより，その回路規模を見積もる．これにより，2章で述べた回路規模を削減する実装方式の有効性を評価する．図2に示すように，SIMD 動作時には4つの PE(Processing Element)として，MIMD 動作時には1つの PU(Processing Unit) として動作する構成を持つ動的切り替え可能な MIMD コアを評価対象とした． 本 MIMD コアの性能諸元を表2に示す．図2と表2に示す MIMD コアを Verilog-HDL で記述し，NEC の 90nm プロセス・ライブラリを用いて，150MHz 動作をターゲットに論理合成を行った．論理合成ツールには，Synopsys[9]社の Design Compiler を用いた．

表2．切り替え可能な MIMD コアの諸元

| PE/PU毎 | SIMD動作時 | MIMD動作時 |
|---|---|---|
| 動作形態 | 4PE | 1PU |
| 命令供給 | CPより供給 | 命令キャッシュ (8KB, 2way-8entry) |
| データメモリ | スクラッチパッド (4KB) | データキャッシュ (8KB, 2way-8entry) |
| データパス | 5way-VLIW | 3way-VLIW |
| 浮動小数 | × | ○ |

図4に，合成結果を基に計算した各モジュールが占める回路規模の比率を示す．また，切り替え可能な MIMD コア全体に対して，MIMD 専用回路と共有化用追加回路が，MIMD コア全体に占める割合を，表3にまとめる．まず，図4の円グラフから判るように，本 MIMD コアの大部分は，ALU 群，レジスタファイル群と SRAM から構成されている．このため，2章で述べた，1)PE の SRAM とレジスタファイル群を用いた命令・データキャッシュ，2)SIMD/MIMD 実行時のデータパスの共有，3)ALU 群を再利用した浮動小数点パス，が有効に作用することが出来る．これにより，図4の網掛け部分が示すように，MIMD 実行時のみ利用されるモジュール群の回路規模の比率を大幅に抑えることが出来た．実際に，MIMD 専用回路（IFU，P$C，D$C，MIMD-L/S，FPC）が占める回路規模の比率は，表3が示すように，切り替え可能な MIMD コア全体の 8.5%に抑えられている．また，SRAM やデータパスなど回路規模が大きく，共有化によるメリットが大きいモジュールのみを共有化することにより，資源共有のために追加された共有化用追加回路(セレクタなど)の回路規模も，約 1.3%に抑えることが出来た． 以上から，動的に切り替え可能な MIMD コアを実現するための回路オーバヘッド全体（＝実現コスト）は，MIMD コア全体の約 10%の回路規模を占めるに過ぎないことが判った．以上のことから，提案する実装方式を用いることによって，動的に切り替え可能な SIMD/MIMD プロセッサを低コストに実現できることを確認した．




※IFU=Instruction Fetch Unit, P$C=Program Cache Controller, D$C=Data Cache Controller, MIMD-L/S=Load/Store Unit for MIMD, FPC=Floating Point Controller, PE-L/S=Load/Store Unit for SIMD, IDU=Instruction Decode Unit

図4．MIMD コアにおける回路規模占有率
表3．MIMD コア全体に占める
切り替えオーバヘッドの割合

表3. MIMD コア全体に占める
切り替えオーバヘッドの割合

| (1)MIMD 専用回路(%) | (2)共有化用 追加回路(%) | (1)+(2) 実現コスト(%) |
|---|---|---|
| 8.50 | 1.28 | 9.78 |

※回路規模に換算して比較

一方で, MIMD 実行時に, プログラム流を制御する命令フェッチ部(IFU)や, キャッシュ L/S 部(PU L/S), 浮動小数制御部(FPC)が, 他の MIMD 専用モジュールに比べて, 実現コスト内において大きな割合を占めていることが分かった. 以下では, その原因について説明する. まず IFU では, 目標の動作周波数を満足すべく, IFUを2段のパイプラインに分割したため, 新たなバッファと複雑な制御回路が必要となり, 回路規模が増加した. 次にキャッシュ L/S 部(MIMD-L/S)は, MIMD 実行時に, 画像処理向きの高効率なデータ転送をサポートする命令を追加したため, 制御ステータスが複雑化し, 回路規模が増加した. 最後に FPC では, 浮動小数演算の例外処理に伴う判断/処理が多く, 回路規模の増加が増加した.

## 4. おわりに

動的切り替え可能な SIMD/MIMD 型プロセッサにおける MIMD コアの構成法について述べた. 主に, 以下の3つの施策を行うことにより, 動的切り替え可能な MIMD コアを低コストに実現することが出来た. 1)SIMD 動作時の複数 PE が持つレジスタファイルとローカルメモリから, 命令・データキャッシュを構成する, 2)SIMD/MIMD における命令を共通化し, 共通のデータパスを利用する. 3)複数 PE のデータパスからなる演算器群を共有化して, 一つの浮動小数点パスを実現する. これらの実装方式をとることにより, 動的な切り替えを可能とする MIMD コアの実現コストを, MIMDコア全体の回路規模の約 10% 程度に抑えられることが確認できた. したがって本実装方式をとることにより, 柔軟で高性能な画像認識プロセッサを SIMD コア並みの低コストで実現できるようになる.

今後は, 実アプリケーションを用いて, 動的切り替え可能な SIMD/MIMD 型プロセッサの詳細な性能検証を行う予定である. 本プロセッサに適した画像認識アルゴリズムを設計・実装し, 今回実装した MIMD コアのキャッシュ性能や演算性能, SIMD/MIMD コア間でのデータ転送能力, そして SIMD 実行と MIMD 実行の使い分け方について, 詳細な検証を行い, より高い性能の実現可能性を検討する予定である.

## 5. 謝辞

本報告を作成するに当たり, 有益なコメントを頂いた NEC システムIPコア研究所の Lieske Hanno 氏, 新 淳氏に, この場を借りて, 感謝申し上げます.

## 参考文献

[1]津川定之, より高度な自動車へ-画像処理のITS 応用-, 映像情報メディア学会誌, Vol.58, No.7, pp.889-892(2004)
[2]二宮芳樹, 今改めて車に載せるカメラを考える, Vision Engineering Workshop 2007, pp.158-159(2007)
[3]高野, 門司, 近藤, 大塚, 日立評論, 2005-04, Vol.87, No.4, pp.329-332(2005).
[4]Jun Tanabe, Yasuhiro Taniguchi, and etc, Visconti:Multi-VLIW Image Recognition Processor based on Configurable Processor, IEEE 2003 Custom Integrated Circuits Conference, pp.185-188(2003).
[5]Shorin Kyo, Shinichiro Okazaki and Tamio Arai, An integrated memory array processor architecture for embedded image recognition systems, Proceedings of the 32nd International Symposium on Computer Architecture, pp.134-145(2005).
[6]Ryusuke Miyamoto, Hiroki Sugano, and Yukihiro Nakamura, Pedestrian Recognition Suitable for Night Vision Systems, International Journal of Computer Science and Network Security, Vol.7, No.1, January 2007
[7]Shorin Kyo, Takuya Koga, Hanno Lieske, Shouhei Nomoto, Shin'ichiro Okazaki, A low-cost mixed-mode parallel processor architecture for embedded systems, International Conference on Supercomputing 2007, pp.253-262(2007).
[8]IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std, 754-1985(1985)
[9]Synopsys, http://www.synopsys.co.jp