Vol.2010-ARC-191 No.3
2010/10/18

SIMD アレイプロセッサにおける マルチスレッド実装方式の検討

野本 祥平　小林 悠記　京 昭倫　岡崎 信一郎†

さらなる普及が予想される画像認識を用いたアプリケーションを処理するため，著者らは，低コスト・高性能・高いプログラマビリティを両立できる高並列 SIMD アレイの画像認識プロセッサを開発してきた．そうした中で，SIMD アレイによる並列処理に適したアルゴリズムであっても，多くの場合，並列化が困難な逐次処理部が付随するため，SIMD アレイの実行効率が低下することが課題となっていた．本稿では，上記課題を解決するため，SIMD アレイプロセッサ IMAPCAR2 をベースに，マルチスレッド実行の導入を検討・評価する．並列処理部と逐次処理部が利用する演算資源がほぼ排他であることに着目し，SIMD アレイによる並列処理と，SIMD アレイの制御プロセッサによる逐次処理とを，マルチスレッド実行する構成を提案する．そして，必要な回路規模の見積もりをした結果，制御プロセッサで 8.9%，SIMD アレイで 0.8%，64PE 構成の SIMD アレイプロセッサ全体で 1.6%の追加コストに抑えられることが分かった．以上より，非常に低コストに, IMAPCAR2 に対して, マルチスレッド実行の機構を付加できることを示す．

Study of the multi-threading implementation for the SIMD array processor

Shohei NOMOTO, Yuki KOBAYASHI, Shorin KYO, and Shinichiro OKAZAKI†

For the application using the image recognition, that is expected to be more widely used, the authors have developed an image recognition processor based on the highly parallel SIMD array that achieves low cost, high performance, and high programmability. But the performance efficiency of the SIMD array is decreased, due to inevitable sequential portion existing in the most algorithms, despite of their possibility to parallel operation of the SIMD array. To solve this issue, we evaluate the cost of integrating multi-thread execution into the SIMD array processor based on the IMAPCAR2, We propose the multi-thread architecture in which the parallel operation and sequential operation can be independently and simultaneously executed by the SIMD array and the control processor, because each of the operations use the different computational resource. We also estimated the amount of circuits, and the increased circuit size is 8.9% required additionally for the control processor, 0.8% for the SIMD array, and 1.6% for the SIMD array processor consists of 64PE. Based on the estimation result, it is demonstrated that the cost for achieving the multi-thread SIMD array is sufficiently low.

1. はじめに

コンピュータ技術の発展に伴い，画像認識を用いたアプリケーションが身近になってきた．利用者の顔を認識するデジタルサイネージ[1]や，実世界に置かれるマーカを認識し，カメラ画像に 3DCG を重畳する拡張現実感[2]などが登場している．こうしたサービスが，携帯電話などのモバイル環境にも進出し始めている．

これらのシステムでは，リアルタイムな画像処理・画像認識が必要となるため，高い演算性能が要求される．同時に，モバイル環境の厳しい熱設計をクリアする低消費電力性と，多様な画像認識アルゴリズムに対応し，開発期間短縮や保守性を向上させるための高いプログラマビリティも要求される．これらの要求を満足するため，我々は，画像認識プロセッサ IMAP(Integrated Memory Array Processor)[3]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 SIMD アレイ型アーキテクチャを採用することにより，制御に要する回路規模を抑制し，その分，演算に用いる回路規模(PE 数)を増やせるため，プロセッサ全体で優れたコスト性能比を実現できる．また，近年の画像認識アルゴリズムの多様化・複雑化に伴い，SIMD 型アーキテクチャが苦手とする処理も増えてきている[4]．それに対応するため，4 つの PE から 1 つの独立に動作するプロセッサ(PU)を再構成する MIMD モードを備える動的切り替え可能な SIMD/MIMD 型プロセッサ IMAPCAR2[5]を開発してきた．

一方で，画像認識をモバイル環境で利用するには，複雑化する環境の中で対象物を見つける必要があるため，アルゴリズムの高度化・複雑化，プロセッサの高性能化が不可避である．そうした中で，著者らは，モバイル環境の限られたコストの中で，より高い性能を実現するため, IMAPCAR2 アーキテクチャの改良を進めている. そして, SIMD アレイの高効率化を進める中で, SIMD アレイによる並列処理に適した処理であっても，並列処理に付随する逐次処理により，SIMD アレイの稼働率が下がるという課題が明らかとなってきた．本稿では，その課題を解決するため，SIMD アレイによる並列処理と，SIMD アレイの制御プロセッサによる逐次処理とのマルチスレッド実行について検討・評価を行う．以下，2 章では，SIMD アレイプロセッサにおける，マルチスレッド実行の必要性を述べる．3 章では，低コストに，マルチスレッド実行を可能とする実装方式について説明する．4 章では，検討中のマルチスレッド実行の実装に必要な，追加回路コストについて評価する．最後に，本稿のまとめと今後の展開を 5 章で述べる．

†ルネサスエレクトロニクス株式会社 技術開発本部　先行研究統括部
〒２１１－８６６６　神奈川県川崎市中原区下沼部１７５３

Vol.2010-ARC-191 No.3
2010/10/18

2. マルチスレッド実行の必要性

本章では，SIMD アレイプロセッサにおけるマルチスレッド実行の必要性を述べる．まず検討する上で，ベースアーキテクチャとした IMAPCAR2 の特徴を述べる．次に，SIMD アレイプロセッサのプログラム構造の特徴を解説し，その高速化には，並列処理と逐次処理の多重実行が有効であることを説明する．次に，多重実行の実現方式として，SIMD アレイによる並列処理と，制御プロセッサによる逐次処理のマルチスレッド実行が有効であることを述べる．

2.1 IMAPCAR2 の特徴

図 1 に示すように，SIMD 動作時の IMAPCAR2 は，1 つの制御プロセッサ CP(Control Processor)と，メモリ(RAM)と密結合した PE(Processing Element)をリング状に多数結合した SIMD アレイとで構成される．制御プロセッサは命令･データキャッシュを有し，IMAPCAR2 全体の制御や，SIMD アレイへの命令供給を，各 PE は，スクラッチパッドとして利用可能なメモリ(RAM)を持ち，制御プロセッサより供給される命令を実行する．IMAPCAR2 の制御プロセッサは，3 段のパイプラインからなり，外部メモリ上の命令のキャッシュ，およびキャッシュからの命令フェッチを行う命令プリフェッチ部(PF)，キャッシュからの命令を一時保存し発行する命令フェッチ部(IF)，命令バッファからの命令を実行する演算部(EXC)から構成される．

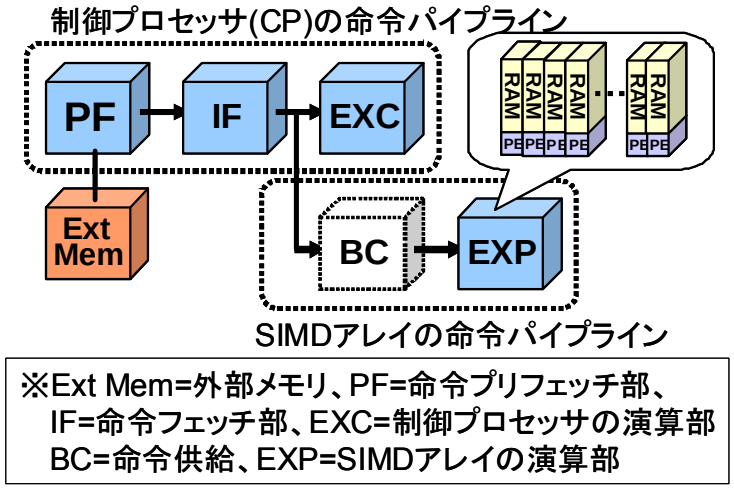


図 1. IMAPCAR2 の命令パイプライン

IMAPCAR2 では，制御プロセッサと SIMD アレイの命令パイプラインを密結合しており，制御プロセッサの PF および IF が，制御プロセッサへの命令と SIMD アレイへの命令の双方を供給している．SIMD アレイの命令は，1 サイクルの命令ブロードキャスト(BC)の後，各 PE の演算部(EXP)で実行される．また処理性能を向上させるため，可変命令長の VLIW(Very Long Instruction Word)実行方式を採用しており，1 サイクルに最大 6 命令を同時発行可能な命令供給能力を持つ．そして，最大 6 命令を制御プロセッサ向けに，最大 5 命令を SIMD アレイ向けに発行できる．その際に，それぞれの命令は混在して発行することが可能であり，SIMD アレイ上での処理と，それに付随する制御プロセッサ上の処理を同時に指定することが出来る．

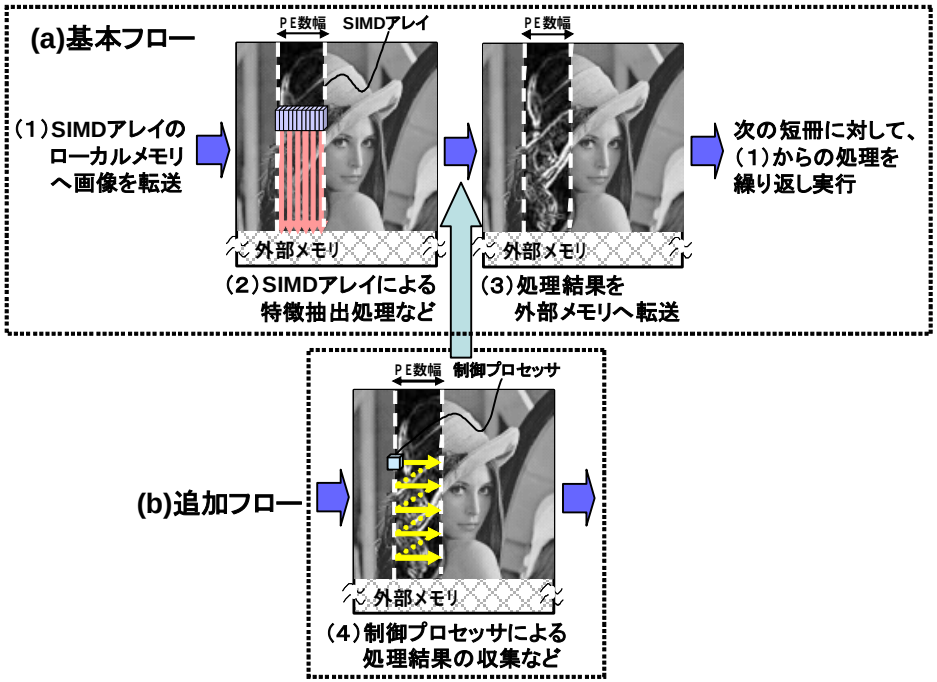


図 2. SIMD アレイプロセッサにおける画像認識の処理フロー

2.2 SIMD アレイプロセッサにおけるプログラム構造

図 2 に，SIMD アレイプロセッサにおける画像認識の処理フローを示す．処理対象となる画像のサイズは，一般に，SIMD アレイの幅より大きいため，基本フローでは，

(1)外部メモリ上の画像の一部(短冊)を SIMD アレイのローカルメモリに転送し，(2)特徴抽出などの処理を行い，(3)その結果を外部メモリへ書き戻す．基本フローでは，図 2 における(1)～(3)を，各短冊に対して行っていく．また，外部メモリとローカメモリの間のデータ転送は，次の短冊のデータをバックグラウンド転送するなどして，隠蔽することが出来る．一方で，基本フローの(2)と(3)の間に，(4)のような追加フローを行うアルゴリズムも多数存在する．追加フローでは，(4)SIMD アレイ処理結果の集計などを行う．同様の処理としては，SIMD アレイ処理結果を基にした，判別ルールの適用，確率量の計算などが存在する．こうした処理は，SIMD アレイによる並列処理を適用できないため，高速化することが出来ず，処理全体のボトルネックとなってしまう．並列処理により高性能を実現するプロセッサにおいては，プログラム中に存在する逐次処理が，その性能を律則する要因となってしまう．これは，アムダールの法則[6]として一般的に知られた事実であり，SIMD アレイプロセッサの高速化においても，逐次処理への対応は必要不可欠である

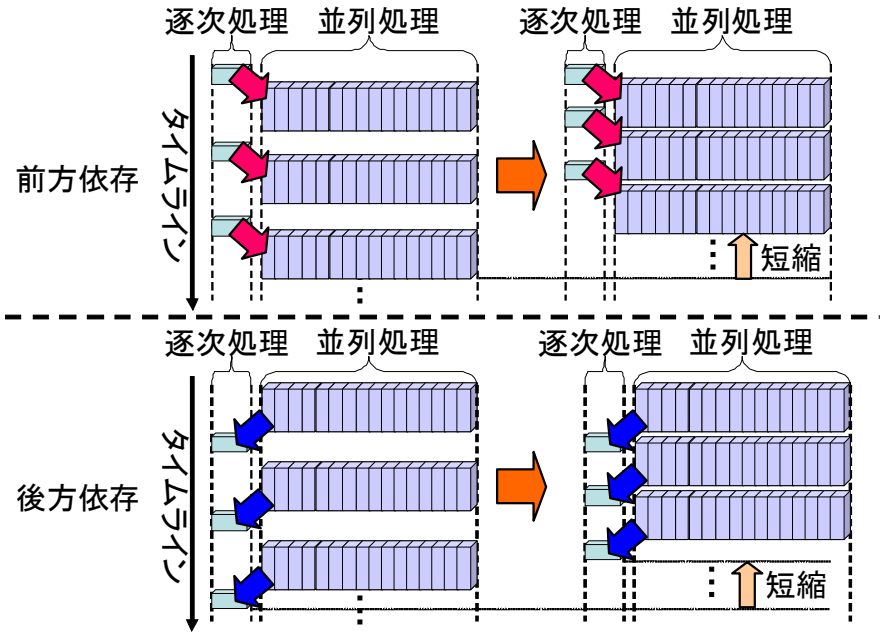


図 3．処理フローにおける定型パターン

図 3 は，追加フローを含む画像認識アルゴリズムを解析し明らかとなった，プログラム中に存在する並列処理と逐次処理の構造的特徴である．図 3 に示すように，前方依存および後方依存のいずれの場合でも，SIMD アレイプロセッサにおける処理フローは，逐次処理と並列処理が交互に繰り返し実行される構造となる．その際に，前方依存における逐次処理では，並列処理を行うための前処理として，データ転送の制御やパラメータ計算などが主に行われる．後方依存における逐次処理では，演算結果の収集や検定処理などが主に行われる．一方で，逐次処理と並列処理との間の依存関係は，一方向のみである場合が非常に多いことも分かった．前方依存であれば，右向きの矢印で示す逐次処理から並列処理への依存関係，後方依存であれば，左向きの矢印で示す並列処理から逐次処理への依存関係である．一方向のみの依存関係であれば，図 3 の右側に示すように，依存関係の無い方向の逐次処理と並列処理をオーバーラップさせ，多重実行することが可能である．これにより，オーバーラップして実行した逐次処理のサイクル分だけ，処理時間を短縮し，処理性能を向上させることが出来る．

一方で，多重実行する並列処理と逐次処理は，それぞれの実行サイクルや制御シーケンスが異なるため，これまで単一のシーケンスのみを扱ってきた SIMD アレイプロセッサでは，それらを多重実行することができない．このため，２つのシーケンスを独立に制御可能な，マルチスレッド・アーキテクチャが必要になる．

2.3 多重実行方式の検討

並列処理と逐次処理の多重実行を可能とするマルチスレッド・アーキテクチャとして，以下の３つの異なる実装方式を検討した．１番目は，SIMD アレイプロセッサの外側のホストプロセッサに，逐次処理をオフロードする方式，２番目は，SIMD アレイプロセッサの制御プロセッサを完全に２重化し，一方の制御プロセッサにおいて逐次処理を実行する方式，３番目は，SIMD アレイプロセッサの制御プロセッサにおいて，２つの異なる実行スレッドを多重実行する拡張を行い，一方のスレッドにおいて逐次処理を実行する方式(SMT; Simultaneous Multi Threading)[7]である．これら３つの実装方式について，その特性を表 1 にまとめる．

まず，回路規模の観点では，ホスト方式が最も優れる．オフロードエンジンとしての利用を想定する SIMD アレイプロセッサは，システム全体の制御を行うホストプロセッサの存在を前提とできるため，その追加コストはほぼゼロである(◎)．一方，２重化方式では，実現コストの高いキャッシュも含む制御プロセッサの全てを２重化するため，追加コストは大きい(△)．SMT 方式は，スレッド情報(制御レジスタ，汎用レジスタ)のみを２重化すればよいため，追加する回路規模は小さい(○)．

スレッド間の同期・データ共有に要するオーバヘッドについては，SMT 方式が最も優れる．ホスト方式では，異なるプロセッサ間で，同期処理やデータ転送を行う必要があるため，遅延時間やデータ転送時間が大きい(△)．２重化方式は，それらの処理

を１つのプロセッサ内で完結できるものの，別々のキャッシュを持つため，コヒーレンシを維持するためのオーバヘッドが必要となる(○)．一方，SMT 方式は，単一のキャッシュを用いるため，コヒーレンシを維持するオーバヘッドも不要である(◎)．

表 1．マルチスレッド実行の実装方式の比較

	ホストプロセッサにオフロード	制御プロセッサの2重化	SMT方式
回路規模	◎	△	○
同期･通信オーバヘッド	△	○	◎
ピーク性能	△	◎	○
画像認識への適応性	△	○	◎

ピーク性能については，２重化方式が最も優れる．ホスト方式では，ホストにおいて他の処理を実行する場合に，大きな性能低下が予想される(△)．一方，２重化方式と SMT 方式は, SIMD アレイプロセッサのみで, 性能向上できる点で優れる．しかし, SMT 方式はキャッシュと演算器を共有するため, ピーク性能で２重化方式に劣る(○)．

最後に, 画像認識への適応性については, SMT 方式が最も優れる．ホスト方式では，画像認識を利用するアプリケーションを実行する余力が，ホストから無くなってしまう(△)．一方，２重化方式と SMT 方式では，そういった問題は無いが，画像認識の特性から，SMT 方式の方が優れる(◎)．2.2 で述べたように，並列処理と多重実行したい逐次処理は，並列処理にかかる前・後処理であり，逐次処理が必要とする処理性能は必ずしも高くない．また，並列処理が SIMD アレイを，逐次処理が制御プロセッサをほとんど利用するため，演算器の競合はほぼ発生しないため，制御プロセッサを２重化する恩恵があまり無い．以上の４つの検討結果をもとに，画像認識における並列処理と逐次処理を多重実行する方式として，SMT 方式を用いることにした．

3. マルチスレッド実行の詳細実装

本章では，SIMD アレイプロセッサ IMAPCAR2 におけるマルチスレッド実行の実装方式について詳細に説明する．はじめに，ベースアーキテクチャである IMAPCAR2 のハードウェアに対する，変更点について述べる．次に，可変命令長の VLIW 実行方式を採用する IMAPCAR2 に適した，スレッドの選択ポリシーを説明する．

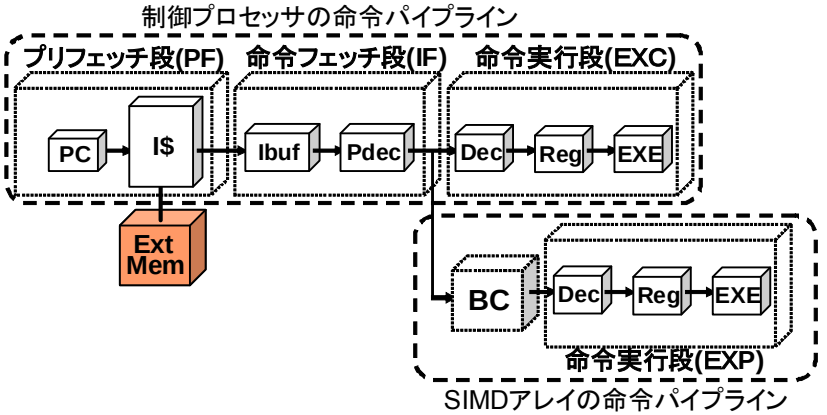
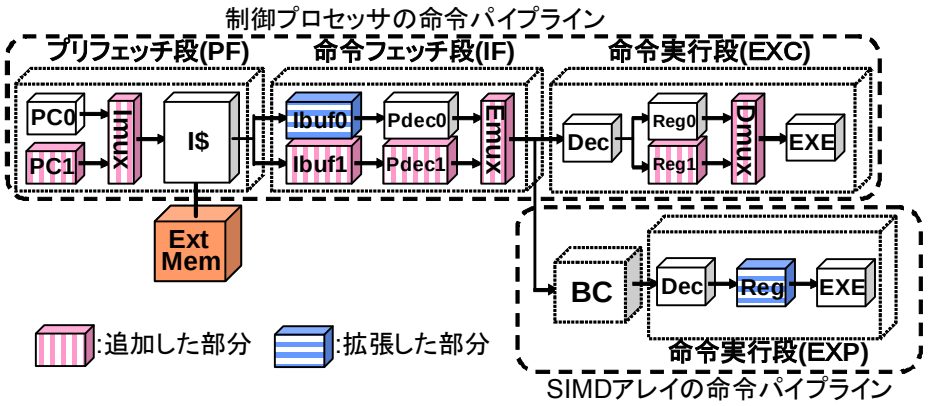


図 4．IMAPCAR2 のハードウェア構成

3.1 マルチスレッド実行に向けたハードウェア構成

ここでは，SIMD アレイプロセッサにおいて，マルチスレッド実行を行うハードウェア実装について述べる．はじめに，ベースアーキテクチャである IMAPCAR2 のハードウェア構成を説明し，その後，マルチスレッド実行のために変更した部分について説明する．図 4 に，３つのステージ(段)から構成される IMAPCAR2 のハードウェア構成を示す．１ステージ目は，プリフェッチ段(PF)であり，プログラムのシーケンスを制御するプログラムカウンタ(PC)と，外部メモリ上の命令列をキャッシュする命令キャッシュ(I$)から構成される．プログラムカウンタの値は命令キャッシュに供給され，その値に応じた命令列が命令キャッシュより読み出される．２ステージ目は，命令フェッチ段(IF)であり，命令キャッシュからフェッチされた命令列を，一旦バッファするための命令バッファ(Ibuf)と，命令バッファに格納された命令列をプレコードするプレデコーダ(Pdec)から構成される．命令バッファは，４命令ワード(=4x4Bytes)の命令列を格納するレジスタから構成され, 可変命令長の VLIW 実行方式を用いるため，最短の VLIW 命令(=4Bytes) であれば４命令，最長の VLIW 命令(=16Bytes)であれば１命令を格納できる．また，IMAPCAR2 では，制御プロセッサと SIMD アレイ向けの命令を VLIW 化しており，それらの分割処理，各命令スロットへの命令割り当ては，プレデコーダにおいて行っている．３または４ステージ目は，制御プロセッサまたは

SIMD アレイの命令実行段(EXC/EXP)であり，命令をデコードする命令デコーダ(Dec)と，演算器へのデータ供給と演算結果を格納するレジスタファイル(Reg)と，実際の演算を行う演算器(EXE)から構成される．制御プロセッサと SIMD アレイは物理的に離れており，SIMD アレイへの命令供給には，１ステージを使った命令供給(BC)が必要であるため，SIMD アレイの命令実行は，制御プロセッサの命令実行に比べて，１サイクル遅れて行われる．



※Ext Mem=外部メモリ、PC0/1=プログラムカウンタ、Imux=アドレスセレクタ、I$=命令キャッシュ、Ibuf0/1=命令バッファ、Pdec0/1=プレデコーダ、Emux=実行命令セレクタ、Dec=命令デコーダ、Reg0/Reg1/Reg=レジスタファイル、Dmux=データセレクタ、EXE=演算部、BC=命令供給

図 5．マルチスレッド実行に対応したハードウェア構成

図 5 に，マルチスレッド実行に対応した IMAPCAR2 のハードウェア構成を示す．マルチスレッド実行に対応するため，新たに追加したハードウェア(赤縦網掛け)と，一部拡張したハードウェア(青横網掛け)とがあり，以下では，それらの変更点について説明する．プリフェッチ段(PF)では，まず，逐次処理と並列処理を行う２つのスレッドの命令シーケンスを制御するため，プログラムカウンタを２重化している(PC0/PC1)．次に，どちらのスレッドの命令列をフェッチするかを選択するため，各スレッドに対応するプログラムカウンタ値を選択するアドレスセレクタ(Imux)を追加している．アドレスセレクタの選択ポリシー，および命令キャッシュを共有とする理由は，3.2 で説明する．

表 2．ハードウェア構成の変更点のまとめ

変更・拡張点	目的
プログラムカウンタ(PC1)を追加	追加スレッドの命令シーケンスを格納するため
アドレスセレクタ(Imux)を追加	追加スレッドによる命令キャッシュへのアクセスを可能とするため
命令バッファ(Ibuf0)を2倍化 命令バッファ(Ibuf1)を追加	追加スレッドの命令を一時格納するため
プレデコーダ(Pdec1)を追加	追加スレッドの命令をプレデコードするため
実行命令セレクタ(Emux)を追加	それぞれのスレッドから、演算器の競合が発生しない命令を選択するため
制御プロセッサにレジスタ(Reg1)を追加	追加スレッドの実行コンテキストを保存するため
制御プロセッサにデータセレクタ(Dmux)を追加	それぞれのスレッドのデータを、共用する演算器で利用するため
SIMDアレイのレジスタ(Reg)の一部のみを2重化	追加スレッドがSIMDアレイのデータを収集するため

命令フェッチ段(IF)では，まず，各スレッドの命令を別々にバッファする必要があるため，命令バッファを２重化し(Ibuf0/Ibuf1)，それぞれが格納可能な命令ワード数を，Ibuf0 で８命令ワード(=32Bytes)，Ibuf1 で４命令ワード(=16Bytes)としている．その理由は 3.2 で説明する．次に，２つのスレッドの命令列を多重実行するため，命令バッファに格納された命令を処理するプレデコーダ(Pdec0/Pdec1)を２重化している．そして，プレデコード結果を基に，命令スロットの衝突を検出し，各命令スロットへ渡す命令を選択する実行命令セレクタ(Emux)を追加している．IMAPCAR2 では，可変長の VLIW 実行方式を採用しているため，プレデコーダによる各命令スロットへの命令割り当てを判定した後でないと，２つのスレッドが利用する命令スロットの衝突を検出できないため，上記の構成をとっている．

制御プロセッサの命令実行段(EXC)では，演算器へのデータ供給と演算結果を格納するレジスタファイル(Reg)を２重化すると共に，演算器(EXE)へ供給するデータを選択するためのデータセレクタ(Dmux)を追加している．SIMD アレイによる並列処理を行うスレッドにおいても，制御プロセッサでの命令実行(シーケンス制御，アドレス計算，パラメータ分配など)が必要であるため，制御プロセッサのレジスタファイル(Reg)

を２重化する．一方，SIMD アレイの命令実行段(EXP)では，SIMD アレイのデータを集計する際に必要となる，最低限のレジスタファイル(Reg)のみを２重化している．逐次処理では SIMD アレイを用いた演算を行わないが，SIMD アレイのデータを集計する処理は，逐次処理でも扱えるようにした方が効率的であったため，それに必要なレジスタファイル(Reg)のみを２重化している．マルチスレッド実行に対応するための変更点の一覧と，それらの理由を表 2 にまとめる．

3.2 スレッドの選択ポリシー

ここでは，図 5 のアドレスセレクタ(Imux)における，スレッドの選択ポリシーについて詳細に説明する．まず IMAPCAR2 において，画像認識のいくつかのアルゴリズムを実装し，それらの IPC(Instruction Per Cycle)の傾向を分析することにより，IMAPCAR2 に適した選択ポリシーを決定する．表 3 は，IMAPCAR2 において，画像認識における３つの基本的なフィルタと，２つの特徴抽出フィルタを実行し，それぞれの IPC をまとめたものである．

表 3．IMAPCAR2 における画像認識処理の IPC の一例

	逐次処理	並列処理	合計IPC
Sobel3x3	1.18	1.32	2.50
RGB2YC	1.18	2.19	3.37
Guassian3x3	1.2	1.80	3.01
Bott_dot	2.13	1.92	4.05
White_line	2.17	2.66	4.83

表 3 には，各アルゴリズムに含まれる逐次処理と並列処理の IPC と，それぞれの IPC を合計した合計 IPC を記載する．IMAPCAR2 では，制御プロセッサと SIMD アレイ向け命令を VLIW 化しており，双方を合わせて，最大６つの命令を実行することが出来る．このため，期待される最大の IPC は６である．しかしながら，表 3 が示すように，逐次処理と並列処理のそれぞれ単体では，その IPC は高々2.66(White_line の並列処理)でしかない．つまり，毎サイクルにおいて最大６命令を実行可能な，命令供給能力と演算能力を，完全に生かしきれていないことになる．そこで，マルチスレッド実行を実装するにあたっては，活用できていない命令供給能力を有効利用し，少ない追加ハードコストで実現できる実装方式を検討した．

図 6 に，検討した命令バッファの構成を示す．命令キャッシュの命令供給能力を生かすため，一方のスレッドの命令バッファ(Ibuf0)が保持する命令ワード数を，最大同時実行命令数(6)を実行する際に消費される命令ワード数(4)の２倍の８とした．そして，当該命令バッファへ供給される命令ワード数(nfi)，当該バッファが保持する命令ワード数(nsi)，当該バッファから消費される命令ワード数(nci) に基づき，アドレスセレクタ(Imux)を制御するスレッド選択ポリシーを導入した．具体的には，式（１）を満たす場合のみ，アドレスセレクタ(Imux)は PC1 を選択し，それ以外の場合は PC0 を選択する．つまり，次のサイクルにおいて，Ibuf0 が保持する命令ワードの数が，８以上となる場合のみ，PC1 を選択し，Ibuf1 への命令供給を行う．この様子を図 7 に示す．

式（１）　$(nfi + nsi - nci) \geq 2 \times 4$

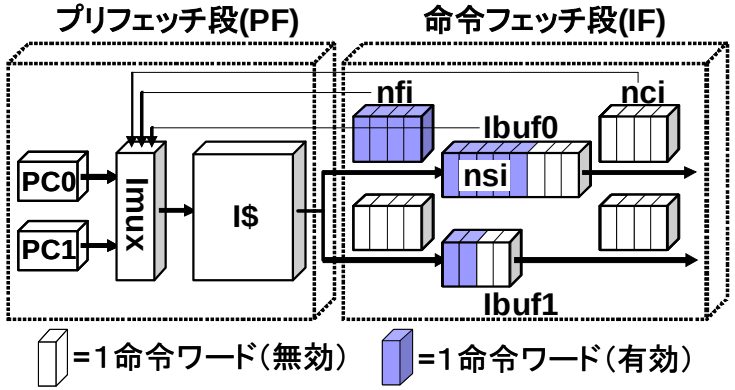


図 6．マルチスレッド対応した命令バッファの構成

サイクル N では，Ibuf1 が命令キャッシュを利用する条件(式１)を満たすため，アドレスセレクタ(Imux)は PC1 を選択する．サイクル N+1 では，命令キャッシュから Ibuf1 へ，４命令ワードの命令供給が行われる．一方，Ibuf0 は，最大消費命令ワード数（４）の２倍の８命令ワードを格納しているため，１サイクル間だけ Ibuf0 への命令供給は止まるものの，その命令実行を継続することが出来る(サイク N+1，N+2)．図 6 に示した命令バッファの構成を用いることにより，Ibuf0 を利用するスレッドにおいて，消費命令ワード数(nci)<供給される命令ワード数(nfi)である場合に，消費されなかった命令ワードを，命令バッファ(Ibuf0)に蓄積することが可能となる．そして，先に述べたスレッド選択ポリシーを用いることにより，Ibuf0 を利用するスレッドに影響を与えることなく，Ibuf1 への命令供給を行うことができる．つまり，IMAPCAR2 では，有効

利用できていなかった命令キャッシュの命令供給能力を，時間軸方向に繰り延べし，別のスレッドへの命令供給に利用することが可能となる．

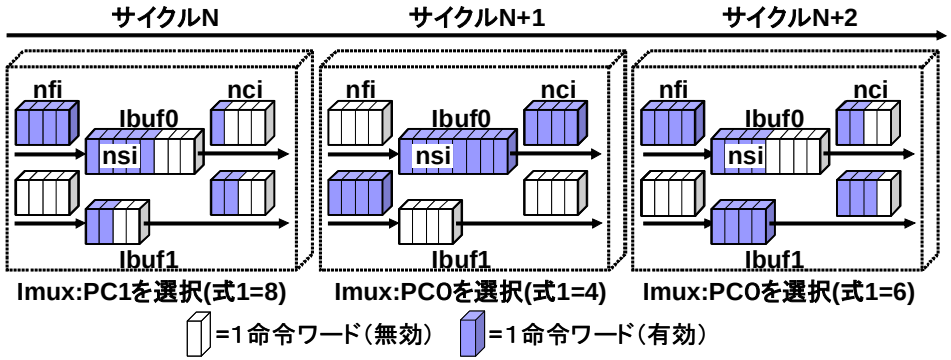


図 7. 命令バッファの動作例

4. 論理合成による回路規模の評価

本章では，３章で説明したマルチスレッド実行を行う SIMD アレイプロセッサの RTL 設計および論理合成を行い，その回路規模を見積もる．そして，低コストにマルチスレッド実行を実装できることを示す．まず，IMAPCAR2 の RTL をベースに，図 5 に示したマルチスレッド実行に必要なモジュールの追加および拡張を行う．合成ライブラリはルネサスエレクトロニクスの UX8(40nm プロセス)を用い，想定する動作周波数を 300MHz として，Synopsys 社の Design Compiler を用いた論理合成を行った．

表 4. 制御プロセッサ(CP)と SIMD アレイ(PE8:8PE 構成)の合成結果

	CP	PE8	32PE	64PE	128PE
増加率	8.9%	0.8%	2.3%	1.6%	1.2%

表 4 には，ベースとした IMAPCAR2 に対する，マルチスレッド対応した SIMD アレイプロセッサにおける，制御プロセッサ(CP)と SIMD アレイ(PE8:8PE 構成)の回路規模の増加率(%)を示す．また，制御プロセッサと SIMD アレイの回路規模を基に計算した，32/64/128PE 構成の SIMD アレイプロセッサ，それぞれの増加率も併記する．表 4 が示すように，SIMD アレイプロセッサにおけるマルチスレッド実行は，制御プロセッサ(CP)で 8.9%，SIMD アレイ(PE8)で 0.8%の回路を追加するだけで，実装できる．また，SIMD アレイプロセッサでは，１つの制御プロセッサと多数の PE により SIMD アレイを構成するため，プロセッサ全体の回路規模の増加率は SIMD アレイの増加率（0.8%）に漸近する．

表 5. 論理合成結果の内訳

	CP			PE8
	PF	IF	EXC	EXP
増加率	0.4%	145.7%	28.9%	1.8%
占有率(対CP、PE8)	0.2%	2.8%	6.5%	0.7%

表 5 には，制御プロセッサ(CP)と SIMD アレイ(PE8)における回路規模の増加率の内訳を示す．具体的には，制御プロセッサのプリフェッチ段(PF)，命令フェッチ段(IF)，命令実行段(EXC)，および SIMD アレイの命令実行段(EXP)における，IMAPCAR2 のそれぞれに対する回路規模の増加率(%)と，それぞれで増加した回路規模が，IMAPCAR2 における制御プロセッサ全体と SIMD アレイ全体の各々に占める占有率(%)も示す．表 5 からは，命令フェッチ段(IF)の増加率が突出して大きいことが分かる．これは，マルチスレッド実行に際して，IMAPCAR2 の命令フェッチ段(IF)をほぼ２重化したことと，２つのスレッドの命令を混在して実行するための命令セレクタ(Emux)を新規追加したことによる．しかし，制御プロセッサ全体に対する増加量が占める割合(占有率)という観点では，制御に関わる段(PF/IF)の増加量の占有率は，命令実行に関わる段(EXC)の占有率に比べると小さい．これは，制御において扱うデータは抽象度が高く，そのデータ量も少ないこと，制御プロセッサの命令実行段(EXC)では，回路規模が大きいレジスタファイルを二重化していることに起因する．一方で，同じ命令実行段であっても，SIMD アレイの命令実行段(EXP)は，回路規模の増加率も占有率も非常に小さい．これは，3.1 で述べたように，SIMD アレイに対しては，マルチスレッド実行に必要最低限のレジスタファイルのみを２重化したことによる．

5. まとめと今後の展開

本論文では，SIMD アレイプロセッサにおけるマルチスレッド実行に関する実装方式について述べた．SIMD アレイプロセッサにおける処理は，SIMD アレイによる並列

処理と，それに付随する逐次処理から構成され，それらをマルチスレッド実行することにより，性能向上が実現できることを述べた．そして，IMAPCAR2 における実装方式を検討し，VLIW 実行方式だけでは完全に生かし切れていなかった命令供給能力を，別のスレッドの命令供給に活用するための命令バッファの構成およびスレッド選択ポリシーを提案した．また，SIMD アレイにおいては，SIMD アレイのデータを集計する際に必要となる最低限のレジスタファイルのみを２重化した．こうした実装により，制御プロセッサで 8.9%，SIMD アレイで 0.8%程度の回路を追加するだけで，マルチスレッド実行を実現できることが分かった．なお，SIMD アレイプロセッサ全体のコストは，32PE で 2.3%，64PE で 1.6%，128PE で 1.2%であり，非常に低コストにマルチスレッド実行を実装できることを示した．

今後は，画像認識の実アプリケーションを用いて，SIMD アレイプロセッサにおけるマルチスレッド実行の詳細な性能検証を行う．定性的には性能改善が期待できるが，マルチスレッド実行の振る舞いは動的に決定される要因が大きいため，メモリバスの転送競合，キャッシュ衝突，演算リソースの競合といった，性能を低下させる要因を静的に見積もることが困難である．このため，それらを考慮したアーキテクチャのシミュレータを開発し，より詳細な性能評価を進めることで，提案するマルチスレッド実行の有効性を検証する予定である．

参考文献

[1] Digital Signage, : http://www.nec.co.jp/solution/video/d_signage.html
[2] ARToolKit, : http://www.hitl.washington.edu/artoolkit/
[3] Shorin Kyo, et al.: An integrated memory array processor architecture for embedded image recognition systems, Computers, IEEE Transactions(2007)
[4] Ryusuke Miyamoto, et.al.: Pedestrian Recognition Suitable for Night Vision Systems, IJCSNS(2007)
[5] Shorin Kyo, et al.: A low-cost mixed-mode parallel processor architecture for embedded systems, ICS(2007)
[6] Amdahl Gene: Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities, AFIPS Conference(1967)
[7] Tullsen, et al.: "Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor", ISCA(1996)