仮想化機構による 演算器アレイ型アクセラレータの効率化

岩上拓矢[†] 吉村和浩[†]
中田尚[†] 中島康彦[†]

我々は，一般的な機械語命令を演算器アレイに写像して高い効率で実行する線形アレイ型アクセラレータ LAPP（Linear Array Pipeline Processor）を提案している．LAPP は多数の演算器をアレイ状に配置し，プログラムの最内ループから演算器アレイのデータパスを構成し，必要最小限のユニットだけで実行することによって，高性能と低消費電力を両立している．しかしこの従来型の LAPP ではスループットを高めるために 1 演算器に 1 命令を固定して割り当てるという物理制約があり，演算器数を超える命令列を高速実行できなかった．本稿ではこのような物理演算器数を超えた命令列を高速実行するために時分割実行による仮想化機構を提案する．本機構は演算器アレイにおいて各演算器で複数命令を時分割実行することによって仮想的に大きな演算器アレイを構築し，物理演算器数を超える長い命令列の高速実行を可能にする．評価の結果，従来型の LAPP と比べ電力あたり性能は約 0.89 倍に低下するものの，面積あたり性能を約 1.15 倍に向上させることができた．

Improvement of FU Array by Virtualization Mechanism

Takuya Iwakami,[†] Kazuhiro Yoshimura,[†] Takashi Nakada[†] and Yasuhiko Nakashima[†]

We have previously proposed Linear Array Pipeline Processor (LAPP), which can map an inner loop of conventional VLIW codes onto Function Unit (FU) array and use minimum required FUs to exploit performance per watt. However, under a fixed mapping, one FU will be specified to a certain instruction and the longest map-able data flow path in the loop kernel is thus limited by the physical depth of the FU array. To address this problem, we propose a virtualization mechanism to extend the mapping ability of the FU array in LAPP. Specifically, this mechanism virtually extended the depth of the FU array by time-sharing the FUs to execute multiple VLIW instructions inside the loop kernel. Our evaluation results indicate that the virtualization will introduce an 11% performance per watt cost from the original LAPP, due the impact to working frequency. However, with virtualization, we can successfully use an 18-stage LAPP to substitute a baseline 36-stage LAPP. The performance per area is accordingly increased to 115% of the baseline LAPP.

1. はじめに

近年，画像処理や科学技術計算に向けた汎用アクセラレータとして，Larrabee[1] などのメニコアや，PPA[2] などの CGRA（Coarse-Grained Reconfigurable Array）が有望視されている．メニコアは汎用プロセッサのコア数をさらに増やしたもので，汎用プロセッサを超える性能を実現している．汎用プロセッサを元にしているため一般的な命令セットが使え，バイナリ互換性が確保されている．しかし電力あたり性能はメニコア化のオーバーヘッドのため単一コアの電力あたり性能と比較して低下する．CGRA は演算器の 2 次元アレイを用い専用ハードウェアに近い構造をしているため単一コアを超える電力あたり性能を実現している．しかし演算器アレイを用いた特殊な構造をしているため，バイナリ互換性はなく，性能を引き出すには新規のコンパイラやプログラムの開発が必要である．

先行研究[3] において，高い電力あたり性能とバイナリ互換性を両立するアクセラレータとして，我々は線形アレイ型アクセラレータ **LAPP**（Linear Array Pipeline Processor）を提案した．LAPP は従来の汎用プロセッサを線形に拡張した構造をしており，ループ中の命令列を演算器アレイに写像し高いスループットで演算を行う．一般的な機械語命令を扱えることからバイナリ互換性があり，専用コンパイラが必要ない

† 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

ことや過去のソフトウェア資産を有効に活用できることが特長である．演算器アレイ上で複数のイタレーションをパイプライン動作させることによって大量の命令を並行実行でき，高速実行中は必要最小限のユニットだけで動作させ，細かな電源管理が可能であることから，LAPP は高い電力あたり性能を実現している．

本稿では，先行研究の従来型 LAPP では高速実行の対象外となっていた「物理演算器数を超える命令列」を写像できるように演算器アレイを拡張する仮想化機構を提案する．従来型 LAPP は演算器アレイ上の演算器と命令が 1 対 1 に対応するため，命令数が演算器数を超える命令列は高速実行できなかった．そこで本機構では各演算器に複数の命令（N 命令）を割り当て，高速実行する際は N 重の時分割実行することで，仮想的に N 倍の大きさの演算器アレイとして動作させる．本機構によって，従来高速実行できなかった命令列に対する高速化を実現する．以後，先行研究で提案された LAPP を**従来型 LAPP**，本稿で提案する仮想化機構を備えた LAPP を**仮想化 LAPP** と呼称する．

2 章では従来型 LAPP の概要を述べる．3 章では従来型 LAPP の問題点を述べた後，時分割実行による仮想化機構について詳述する．4 章では評価モデルとパラメータについて述べ，5 章では性能・回路規模・電力の見積りに基づき，仮想化 LAPP の評価を行う．面積あたり性能および電力あたり性能の観点から仮想化 LAPP の評価について論じる．最後に，6 章で本稿のまとめを行う．

2. LAPP

本章では従来型 LAPP と仮想化 LAPP に共通する LAPP の実行モデル，構造，各動作モード，低消費電力機能について述べる．

2.1 実行モデル

LAPP は，イタレーション間に依存関係のない最内ループ（**ループカーネル**）を対象として高速化し，ループカーネル以外は従来の VLIW プロセッサと同一の速度で実行する．LAPP は高速実行の起動命令（データプリフェッチ命令と兼用）から後方分岐命令までを ループカーネルとして認識する．LAPP は 3 つの動作モードを有している．まず高速実行中を表す**アレイ実行**，アレイ実行に入る前に演算器アレイを設定する**アレイ設定**，それ以外の**通常実行**である．通常実行からアレイ設定にはプリフェッチ命令を契機として遷移する．アレイ設定では入力配列と対応する L1 データキャッシュの各ウェイにデータをプリフェッチし，同時にループカーネルを演算器アレイに写像し演算器ネットワークを構築する．演算器アレイの設定ではループカーネルの命令を演算器アレイの演算器に写像する．そしてデータプリフェッチ，命令写像およびネットワークの構築が完了し次第，アレイ設定からアレイ実行に遷移する．アレイ実行では L1 データキャッシュからロードしたデータを演算器アレイに流し込み，実行する．アレイ実行の終了はループ脱出用分岐命令の成立であり，終了処理の後，通常実行へ復帰する．定常状態では演算結果が毎サイクルストアされ，そのときの IPC はループ内の命令数と等しくなり，高いスループットを実現する．

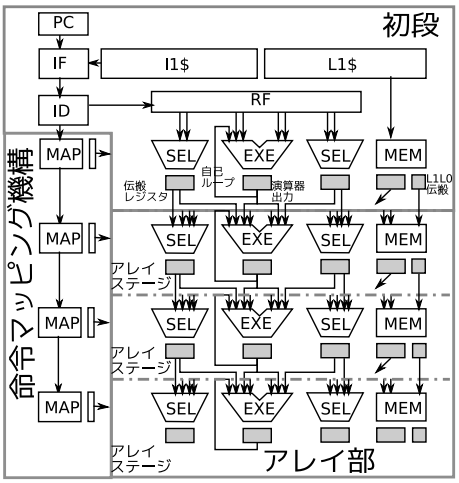

図 1 LAPP の構造

2.2 LAPP の構造

LAPP の全体図を図 1 に示す．LAPP は大きく分けて**初段**，**アレイ部**，**命令マッピング機構**から構成される．

初段は従来プロセッサと同様にプログラムカウンタ（PC），命令フェッチ（IF），命令デコーダ（ID），レジスタファイル（RF），演算器（EXE），メモリアクセスユニット（MEM），L1 データキャッシュ（L1$），命令キャッシュ（I1$）から構成される．

アレイ部は複数段の**アレイステージ**を線形に接続した構造をしている．アレイステージは演算器，メモリアクセスユニット，L1L0 伝搬，セレクタ（SEL）および**伝搬レジスタ**から成る．レジスタファイルはセレクタと伝搬レジスタに分散して配置され，伝搬レジスタは演算器の入力レジスタと兼用である．LAPP は VLIW を基本とするため，EXE には複数の演算器を含む．メモリアクセスはロードやストアを処理するための小規模データキャッシュ（L0$）を含み，後段の L0$にデータを供給するため L1L0 伝搬レジスタを備える．図 1 で灰色の箱で示された伝搬レジスタ，演算器出力，L1L0 伝搬がアレイ実行時のパイプラインレジスタに当たる．これらパイプラインレジスタの内容

は次のアレイステージにしか渡されない．しかしループカウンタの更新（例．i++）のような**自己更新命令**は自演算器の結果を自演算器で使用する．そのために各演算器には**自己ループ**と呼ばれる，自演算器の結果を自身の入力にバイパスさせるループが備えられている．

命令マッピング機構はアレイ設定時に，各段の演算器や伝搬レジスタに命令やレジスタを割り当てるために利用され，各アレイステージに対応している**マッパ**（MAP）が縦に接続された構造をしている．マッパは対応するアレイステージの演算器や伝搬レジスタにループカーネルの命令やレジスタを割り当てる．これら命令やレジスタ番号などの**割り当て情報**はここに格納される．命令割り当ての詳細については文献 4) を参照されたい．

2.3 低消費電力機能

LAPP の各ユニットは実行時には電源オン，Dynamic Voltage Scaling（DVS）による低消費電力モード，電源オフ，クロックオフの状態がある．電源のオンオフは Power Gating（PG）によって行い，クロックのオンオフは Clock Gating（CG）によって行う．初段の演算器や L1$は，通常実行からアレイ実行を通して使用されるため常に電源オンの状態にある．I1$に関してはアレイ実行中はキャッシュアクセスがないため，アクセスできないがその内容を保持する低消費電力モードにできる．命令マッピング機構はアレイ設定時のみ電源を供給し，他の動作モードでは電源オフとなる．さらに命令割り当てにおいて使用されないアレイステージは電源を供給しない．また使用されるアレイステージであっても，割り当てられなかった演算器や伝搬レジスタは電源がオフのままとする．割り当てられた演算器や伝搬レジスタは CG を適用しクロックのみオンとする．なお本稿では，DVS，PG，CG に必要な回路は全体の回路規模に対して十分小さいため考慮しない．

3. 時分割実行による仮想化機構

3.1 仮想化のアイデア

2 章で述べた従来型 LAPP では設計時に決定されるアレイの段数が制約となり，それを超える命令列は演算器アレイに写像できずアレイ実行できない．これは各アレイステージには 1 個の VLIW 命令しか割り当てることができないためである．

仮想化機構は，演算器を時分割利用することで仮想的に大きな演算器アレイを構築し，これまで物理演算器数により写像できなかったループカーネルのアレイ実行を可能にする．具体的には 1 段に複数（N 個）の VLIW 命令を割り当て，アレイ実行時はこの N 命令を切り替えて時分割で実行する．またこの仮想化機構は従来の演算器アレイにデータパスを追加し制御を変更するだけで，アレイステージを増やすよりもハードウェア量が少なくて済む．以後，1 段あたりの命令割り当て数をアレイ実行の多重化度と呼び，N で表す．演算器アレイの仮想化とは多重化度を高めることで，従来型 LAPP と比べ N 倍の大きさの仮想的な演算器アレイを提供する．段数を L と置くと，仮想化によって演算器アレイの段数を超える L × N 命令までの高速実行を可能にする．

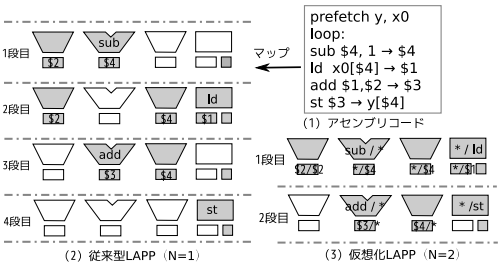

図 2 複数命令割り当ての例

3.2 複数命令割り当て

仮想化 LAPP における複数命令割り当てについて従来型 LAPP との比較を通して説明する．N=2 としたときの複数命令割り当ての例を**図 2** に示す．入力配列 x0 に定数を加算し，その結果を配列 y に書き込むというアセンブリコードを図 2（1）に示す．ここでアレイ実行に必要な分岐命令（bra）やアドレス計算等は省略した．従来型 LAPP であれば図 2（2）のように各命令とレジスタが各段に割り当てられる． 仮想化の複数命令割り当てでは，図 2（3）のように従来型の 1 段目と 2 段目が仮想化の 1 段目，従来型の 3 段目と 4 段目が仮想化の 2 段目へと割り当てられる．ここで*は命令やレジスタが割り当てられないことを表している．

3.3 時分割によるアレイ実行

時分割によるアレイ実行の例を**図 3** に示す．ここで，図 2（1）における 3 段目と 4 段目だけを抜き出している． 従来型 LAPP では図 3（1）のように Clock Cycle (CC) 1 で，前段の結果である $2，$4 および $1 を入力とし演算して $3 と $4 を出力している．次の CC 2 では 1 段目に次のイタレーションのデータが入力される．同時に 2 段目に $3 と $4 が入力され，メモリアクセスユニットにストアされる．仮想化 LAPP では図 3（2）のように CC 1 で前段の結果である $2，$4 および $1 が入力され，$3 と $4 を出力するところまでは同じだが，CC 2 のストアが物理的には同じアレイステージにあるため，自段に入力するために従来のパス（A）とは別の自段へのパス（B）を用い，あ

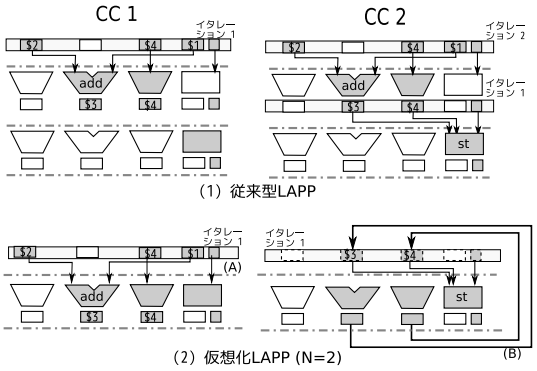

図 3　時分割によるアレイ実行

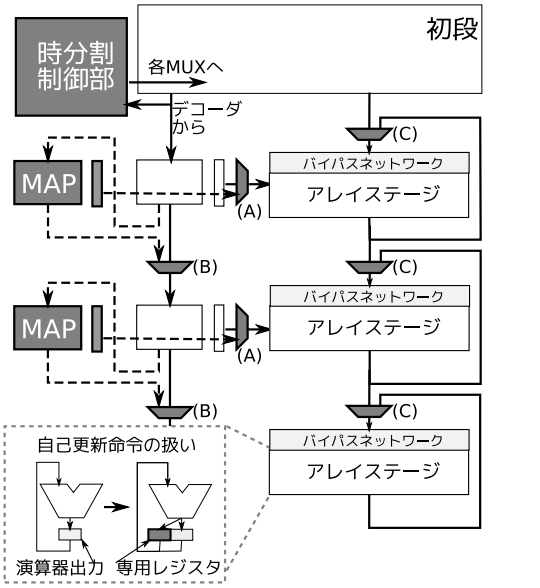

図 4　仮想化機能を備えた LAPP（$N_{max} = 2$）

たかも前段から $3 と $4 が入力されたように扱う．従来型 LAPP では次の CC 2 で 1 段目にイタレーション 2 が入力されているが，仮想化でイタレーション 2 が入力されるのは CC 3 となる．

3.4　仮想化機構の全体構成

仮想化機構を備えた LAPP の全体図を**図 4** に示す．図 4 は，仮想化の最大多重度（以後 N_{max} とおき，実行時の多重化度 N と区別する）が 2 の構成である．仮想化のための時分割制御部，マッパを追加する．これ以外にもバイパスネットワークと演算器に変更を加える．以後，各ユニットについて詳細に述べる．

3.5　時分割制御部

制御部は，実行時の 1 段に割り当てる VLIW 命令数 N の決定とアレイ設定やアレイ実行における各段のデータパスの制御や命令の切り替えを行う．N はループカーネルによって動的に変更され，式 ; $N = \lceil M/L \rceil$ から決定される．$M \leqq L$ の場合，N=1 としてそのままアレイ部に写像できるため従来型と同等のアレイ実行が可能である．一方 $M > L \times N_{max}$ の場合，仮想化した段数を超えアレイ実行できないため通常実行での動作となる．N によって命令割り当ての結果が変わるため，N はアレイ設定の前に決定しなければならない．従来の LAPP ではループカーネルはプリフェッチ命令から後方分岐命令までという仕様であり，アレイ設定に入る前に VLIW 命令数を知ることはできない．そこでプリフェッチ命令を拡張して，アレイ設定に入るループカーネルの VLIW 命令数を表すフィールドを付加する．

3.6　命令マッピング機構

アレイステージと割り当て情報がこれまでの 1 対 1 から 1 対多の対応となり，命令マッピング機構はデータパスが大きく変更される．アレイステージからみて N_{max} 個のマッパが水平方向に置かれる．時分割実行において N 個の割り当て情報を切り替えるために，図 4 のように N_{max} ポートの MUX（A）がマッパ・アレイステージ間に配置される．N が変わると各段が使用する割り当て情報の数が異なるためマッパ間の接続が変わり，N=1 のときは実線，N=2 のときは点線のデータパスが使用される．このデータパスをアレイ設定の前に変更するために N_{max} ポートの MUX（B）が挿入されている．

3.6.1　アレイステージ

図 2 の（2）に示したように，時分割実行には次のアレイステージに渡す演算結果を自分のアレイステージに渡す必要がある．このため，図 4 のように前段のパイプラインレジスタか自段のパイプラインレジスタかを選択する 2 ポートの MUX（C）を設置する．時分割実行により N サイクルに 1 回の実行になり，入力スループットは 1/N となるため，パイプラインレジスタは増加しない．

3.6.2　自己ループ

自己更新命令は演算結果を一時的に演算器出力に保持し，次のサイクルではその演算器出力の結果を自己ループを介して演算する．従来のアレイ実行中は毎サイクル同じ命令が実行されるため，演算器出力を自分自身の入力に接続することによって正しく実行できた．ところが時分割実行をすると，サイクル毎に実行する命令が異なるため演算器出力に保持していたレジスタ値を N–1 サイクルの間保持する必要がある．そこで演算器に自己更新命令用のバッファを設け次にその自己更新命令を実行するまでそのレジスタ値を保持する．演算器に割り当てられる N_{max} 命令すべてが自己更新命令というケースが考えられるので，演算器出力のほ

表 1　RTL シミュレータの諸元

初段	
命令デコード幅	最大 8/cycle
汎用レジスタ (GR) 数	32
メディアレジスタ (MR) 数	32
外部バスとの転送速度	8bytes/cycle
命令キャッシュ	4way 16KB (64bytes/line)
L1 キャッシュ	4way 16KB (64bytes/line)
L1L0 転送	16bytes/cycle
ストアバッファ	4entries
アレイ部	
命令写像設定	2cycle/段
GR 用伝搬レジスタ数	11
MR 用伝搬レジスタ数	9
L0 キャッシュ	4blocks × 16words
L0 伝搬	16bytes/cycle
ストアバッファ	4entries
最終段から L1 への転送	4bytes/cycle

かに N_{max}–1 個のバッファを用意する．

4. 評 価 方 法

本章では，性能・回路規模・消費電力の観点から仮想化 LAPP を評価するための，シミュレータのパラメータや評価モデルについて述べる．

4.1 命令セットと評価ベンチマーク

評価には画像処理プログラムを用い，命令セットとして 8-way VLIW を採用した．VLIW の内訳はアドレス計算ユニット（EAG）を含む 1 ロードストアユニット，1 ブランチユニット（BRC），3 ALU，4 メディア演算器（MEDIA）である．ロードおよびストアは，アドレス計算とメモリアクセスのそれぞれ 1 サイクルとし，全体のレイテンシは 2 とする．

評価に用いる画像処理ベンチマークは，フレーム補間（F–1, 2, 3），画像拡大（Z），鮮鋭化（S），メディアンフィルタ（M），輪郭抽出（E），輪郭ノイズ除去（N）の 8 カーネルから成る．各カーネルはメディア命令を使い手動最適化しており，ループカーネル直前にプリフェッチ命令を挿入している．入力データは RGB 各 1 バイトを含む 4 バイトの 320×240 画素とし，各カーネルには最大 3×3 画素が入力される．これらカーネルをすべてアレイ実行するには 33 段必要であるため，評価には 36 段構成（初段＋アレイステージ 35 段）を基準として用いる．

4.2 性能モデル

性能指標としては実行サイクル数と総命令数から計算した平均 IPC を用い，それぞれ LAPP の RTL シミュレータから算出する． サイクル数と IPC にはアレイ実行中の時間以外にループ外の処理，プリフェッチ，アレイ設定のサイクルも含める．この RTL シミュレータの諸元を**表 1** に示す．36 段構成の通常実行，アレイ設定，アレイ実行のサイクルをそれぞれ C_{normal}，C_{setup}，C_{array}，実行した総命令数を $Inst$ として，提案手法による IPC_N は次の式から算出する．

$$IPC_N = Inst/(C_{normal} + C_{setup} + C_{array} \times N)$$

表 2　回路規模および電力見積もり

ユニット名	ユニット個数 初段	ユニット個数 アレイ 1 段	回路規模 [ゲート]	電力 [μW]
PC	1	0	1,050	1,577
IF	1	0	48,155	53,500
ID	1	0	25,943	22,200
RF	1	0	91,830	31,400
I1$	1	0	198,400	90,310
L1$	1	0	382,700	143,100
EAG	1	1	2,235	2,209
ALU	3	3	10,765	6,373
MEDIA	4	4	7,677	4,983
BRC	1	1	1,557	1,287
伝搬レジスタ	20	20	1,900	3,920
MEM, L0$	1	1	34,652	4,952
MAP（1 段）	-	-	24,700	26,200
		初段合計	768,600	
		アレイステージ (1 段)	138,100	
		命令マッピング機構 (36 段)	889,200	

4.3 回路規模モデル

回路規模の指標としては NAND ゲート換算したゲート数を用いる．評価において初段だけの従来プロセッサ，従来型 LAPP として 36 段構成，仮想化 LAPP として 36 段を基準に，18，9，6 段を用いる．各構成の N_{max} はそれぞれ 2，4，6 となり，$L \times N_{max}$ から計算されるアレイ実行可能な最大命令数は等しい．どの構成でも最大の命令数とマッパの総数は等しいため，命令マッピング機構は常に同一のものを使用する．

回路規模パラメータは HDL 記述の RTL モデルを元に見積もった．回路規模パラメータを**表 2** に示す．第 2，3 列は初段およびアレイステージにおけるユニットの個数を表し，第 4 列に各ユニット 1 個あたりの回路規模を，第 5 列に 4.4 節で述べる電力見積もりの結果を示している．最後の 3 行には初段，アレイステージ，命令マッピング機構の回路規模合計を示す．回路規模は，HDL 記述を 180nm，1.8V テクノロジ，100MHz 制約において Design Compiler 2008.09 によって論理合成し，得られた cell area をゲート数に換算したものである．I1$と L1$はキャッシュのコントローラも含んでおり，キャッシュ本体は CACTI 3.2[5] から見積もっている．

4.4 電力モデル

電力指標としては実行時間全体の平均消費電力を用いる．表 2 の第 5 列に電力パラメータを示す．作成した RTL モデル上でベンチマークプログラムをキャッシュミスやストールすることなく実行したときの論理回路部分の平均消費電力を PrimeTime PX 2007.12-SP03 を用いて測定したものである．I1$と L1$のキャッシュ本体に関して CACTI 3.2 によって見積もった．この電力パラメータと，キャッシュミスやストールのシミュ

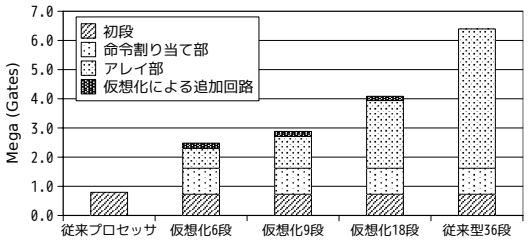

図 5 各構成の回路規模内訳

レーションを含む RTL レベルシミュレータから得られた動作サイクル数から求めた各モジュールの動作時間を積算することで従来型 LAPP の消費電力量を求め，全実行時間で除算し平均消費電力を求めた．仮想化 LAPP の平均消費電力についても仮想化機構を組み込んだ RTL モデルと RTL レベルシミュレータを用いて同様に算出した．

文献 6) では，DVS により理論上，オン時の 33% 前後になると述べており，本稿では DVS が適用されたユニット（I1$および L1$）の消費電力はオン時の 33% になると仮定する．CG によりクロックの供給を止めたユニット（I1$および L1$以外）は 10%，PG により電源の供給がないユニットは 0% と仮定する．

5. 評　　価

本章では，回路規模・性能・消費電力の各指標を単独に見て仮想化 LAPP の評価と分析を行い，また総合評価として，面積あたり性能および電力あたり性能の観点から評価を行う．

5.1 回路規模比較

本節では従来型 LAPP と仮想化 LAPP との回路規模の比較を行う．縦軸は回路規模（M gates）で，各構成について初段，命令マッピング機構，アレイ部，仮想化による追加回路を積み上げて，**図 5** に示す．

まず仮想化による追加回路のための回路規模増は全体の約 4% と小さい．これは追加回路を構成する MUX やレジスタよりも各段の演算器や伝搬レジスタが大きいためである．従来プロセッサは，初段だけから構成されるのに対し，LAPP はアレイ実行のための回路が追加され，初段から回路規模が増加している．回路規模は従来型 36 段が最も大きいが，仮想化 18 段は従来型 36 段から大きく減少している．アレイ部の回路規模は段数に比例しているため，回路規模の削減は段数の減少による．しかし仮想化 9 段や 6 段では，段数に比例しない初段や命令マッピング機構の占める割合が大きく，全体の回路規模としては従来型 36 段の $1/N_{max}$ の回路規模よりも大きくなる．

5.2 性 能 比 較

本節では，仮想化 LAPP の目的である性能向上を確認するために，従来型 LAPP と仮想化 LAPP との IPC の比較を行う．従来型および仮想化 LAPP の各構成でベンチマークごとの IPC を**表 3** に示す．

表 3 ベンチマークごとの平均 IPC

ベンチマーク	従来プロセッサ	仮想化 6 段	仮想化 9 段	仮想化 18 段	従来型 36 段
FI–1 (20)	1.4	6.1	6.1	**9.1**	12.1
FI–2 (33)	1.3	7.2	10.2	**17.7**	28.1
FI–3 (10)	1.3	5.6	5.6	9.0	9.0
Z (29)	1.9	10.2	14.2	**23.6**	35.1
S (27)	1.7	7.9	11.0	**18.3**	27.3
M (23)	2.0	8.9	8.9	**14.9**	22.3
E (14)	1.2	4.4	7.6	11.9	11.9
N (26)	2.0	5.3	6.9	**9.8**	12.4
算術平均	1.6	6.9	8.8	14.3	19.8

表 4 ベンチマークの平均消費電力（mW）

	従来プロセッサ	仮想化 6 段	仮想化 9 段	仮想化 18 段	従来型 36 段
平均電力	401	413	463	589	690

まず従来プロセッサと仮想化 LAPP を比較する．従来プロセッサで実行した際の平均 IPC は 1.6 である．仮想化 LAPP はすべてのベンチマークでアレイ実行が可能となるため従来プロセッサだけの通常実行よりも約 4.3 から 8.9 倍の性能向上を達成している．

次に従来型 36 段と仮想化 LAPP を比較すると，仮想化 LAPP は従来通りアレイ実行できる FI–3 や E を除くと従来型 36 段以下の性能となっている．しかし仮想化 LAPP の平均 IPC は従来型 36 段の $1/N_{max}$ とはなっていない．実行する命令数はどの構成でも変化しないので，この性能の増減は単純にサイクル数の違いによる．つまり仮想化によってアレイ実行時のサイクル数が N_{max} 倍となっても，アレイ実行以外のサイクルがあるため全体のサイクル数が N_{max} 倍とならないためである．このような理由で，仮想化 LAPP においては従来型 36 段からの性能低下が抑えられているといえる．

5.3 消費電力比較

5.3.1 ベンチマークごとの消費電力

各構成での全ベンチマークの平均消費電力を**表 4** に示す．消費電力は従来プロセッサが最も小さく，段数の増加に伴い消費電力も増加する．従来プロセッサには LAPP にある低消費電力機能はないが，そもそも演算器が 1 段分しかないため消費電力は少ない．従来型 36 段では各演算器に 1 命令を割り当てるためアレイ実行の際にはループカーネルの命令数と同数の演算器が動き，多くの電力を消費する．一方，仮想化 LAPP は従来プロセッサと従来型 36 段の中間の電力となっている．これは仮想化 LAPP では N_{max} 重の時分割実行により，同時に動く演算器が減るためである．そのため最も N_{max} の大きな仮想化 6 段が消費電力は最

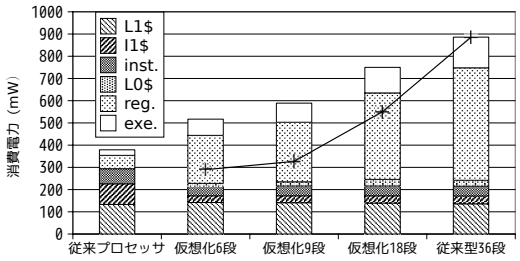

図 6 ベンチマーク S の消費電力内訳

も小さく，従来プロセッサと同程度となっている．

5.3.2 消費電力の内訳

仮想化によるオーバーヘッドを分析するためにベンチマーク S の消費電力の内訳について述べる．各構成における消費電力内訳を**図 6** に示す．縦軸が消費電力（mW）で，L1$は L1 データキャッシュ，I1$は命令キャッシュ，inst. は命令フェッチ，命令デコーダおよび命令マッピング機構，L0$はメモリアクセスユニット，reg. は伝搬レジスタとレジスタファイル，exe. は演算器を表す．

L1$は実行中全体を通して動きつづけるため，全構成で消費電力は同程度となる．I1$や inst. の消費電力は従来プロセッサが最も大きい．従来プロセッサの I1$や inst. はキャッシュミス中を除き常に通常電力で動き続け，LAPP ではアレイ設定やアレイ実行時は低電力モードが適用される．L1$と I1$において従来型 LAPP と仮想化 LAPP の間で動作に変更はないため，消費電力にも変化がない．

L0$，reg. および exe. はアレイステージのユニットであり，従来プロセッサから仮想化 6，9，18 段を経て最大の従来型 36 段へと消費電力が増加している．これは段数が増加することにより同時に動作するユニット数が増加するためである．

アレイステージの消費電力が段数に比例すると仮定し，キャッシュの消費電力はそのままに，従来型 36 段の消費電力の $1/N_{max}$ となる消費電力を図 6 に折れ線で示す．もし段数とアレイステージの消費電力が比例するならば，仮想化 LAPP の消費電力は折れ線上に乗るはずである．しかし積み上げた消費電力は理想的な消費電力を大きく超えており，それぞれの消費電力も従来型 36 段の $1/N_{max}$ とはなっていない．仮想化によって段数は確実に $1/N_{max}$ となっているため，これは以下のような仮想化によるオーバーヘッドが発生しているものと考えられる．

（1）今，2 段のアレイステージを N=2 の仮想化で 1 段にまとめる例の消費電力を考える．ここではレジスタなどを無視して演算器だけに注目する．まず 1 段目，2 段目ともに ALU 命令の場合，2 命令割り当てられた 1 段では ALU を共用するため，アレイ実行時の 1 段あたりの消費電力は従来型・仮想化ともに等しくなる．ところが仮想化ではアレイ実行の時間が 2 倍に延びるため，従来の 1 段あたりの平均消費電力よりも仮想化の平均消費電力が大きくなる．

（2）次に 1 段目が ALU 命令，2 段目が メディア命令のように別のユニットを使用する場合，2 命令割り当てられた 1 段では ALU と メディア演算器 は 2 サイクルに 1 回しか演算しないが，アレイ実行中の全期間において電力を消費する．これは何も演算しないサイクルにおいて PG が適用できないため，アレイ実行時の 1 段あたりの消費電力は，従来よりも増加する．従来型 LAPP は 1 命令 1 演算器で理想的な構成であり，nop のような何も演算をしない命令が演算器に割り当てられず無駄な電力消費はない．

（1）および（2）の理由から，各段の消費電力が従来型 36 段の $1/N_{max}$ とはなっていない．このように仮想化によるオーバーヘッドは，アレイ実行の時間が増加することと複数命令割り当てが関係しており，仮想化の構造に起因している．

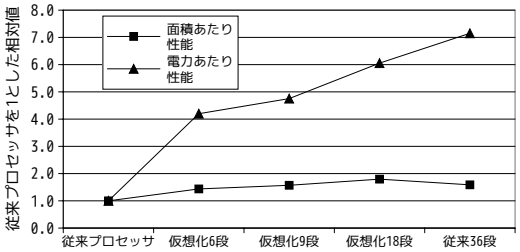

図 7 面積あたり性能と電力あたり性能

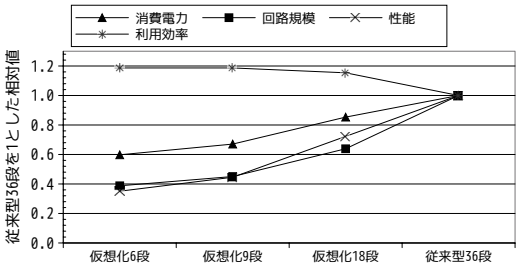

図 8 従来型 36 段比の減少率

5.4 総合評価

5.3 節までの測定結果から面積あたり性能と電力あたり性能を算出し，従来プロセッサを 1 とした相対値を**図 7** に示す．さらに従来型 36 段を基準とした性能・回路規模・電力の減少率，アレイステージの利用率を**図 8** に示す．

5.4.1 面積あたり性能

図 7 より面積あたり性能は仮想化 LAPP ではすべ

て従来プロセッサよりも高く，仮想化 18 段で最大 1.8 倍の性能向上となっている．それに加えて従来型 36 段よりも面積あたり性能が約 15% 高くなっている．

図 8 によると，仮想化 LAPP はアレイ実行を時分割実行するため従来型の LAPP よりも性能は低下するとともに，要求されるアレイステージも減少するため回路規模は小さくなっている．仮想化 18 段では低下した性能の相対値が，削減した回路規模のそれを上回っており，面積あたり性能が従来型 36 段比で向上したのは回路規模削減の効果が大きい．これは従来型ではまれにしか使用されず性能向上への貢献が低かったアレイステージが削減され，仮想化により演算器や伝搬レジスタなどを含むアレイステージの利用率が上がったためと考えられる．

5.4.2 電力あたり性能

図 7 によると，仮想化 LAPP の電力あたり性能はすべて従来プロセッサよりも高く，段数が多くなるほど従来型 36 段の性能に近づいている．仮想化 18 段は従来プロセッサの約 1.5 倍の電力を消費し，約 8.9 倍の性能を実現しており，結果として電力あたり性能は最大 6.1 倍と大きく向上している．

図 8 では，消費電力は回路規模の削減ほどは減少していない．これは 5.3 節で述べたようにキャッシュの消費電力が仮想化で減少しないことと，アレイ実行に関しては仮想化によるオーバーヘッドが存在するためである．このため，電力あたり性能の低下は回路規模の削減とのトレードオフと見なすことができる．

以上の結果から，仮想化 LAPP は従来プロセッサを超える面積あたり性能および電力あたり性能を実現している．さらに仮想化 18 段ではこれに加えて，電力あたり性能は一部低下するが従来型 LAPP を超える面積あたり性能を実現している．

5.4.3 アレイステージの利用率

従来 LAPP 比での面積あたり性能の向上についてアレイステージの利用率の観点から分析する．なおアレイステージの利用率は，段数に対するループカーネルの VLIW 命令数の割合から求めている．図 8 の利用率によると，仮想化の多重度が増加するにしたがって利用率が上がっている．従来型 36 段から仮想化 18 段において，利用率は段数が減少し相対的に使用しないアレイステージが削減されたため大きく改善する．しかし仮想化 9 段，6 段では利用率の向上が高止まりするため，多重化度の増加によってオーバーヘッドが増加し，性能低下が顕在化する．このような利用効率と性能のバランスから，仮想化 18 段が最大の面積あたり性能となる．

6. おわりに

本稿では，線形アレイ型アクセラレータ LAPP において，演算器アレイを拡張する仮想化機構を提案した．この仮想化機構は 1 演算器に N 命令を割り当て，高速実行中は各命令を N 重の時分割実行することによって，従来高速実行できなかった「物理演算器数を超える命令列」の高速実行を可能にする．評価の結果，仮想化機構を備えた LAPP は先行研究で提案された従来型 LAPP よりも面積あたり性能は 1.15 倍に向上するが，仮想化のオーバーヘッドによって電力あたり性能は従来型 LAPP の 0.86 倍となった．しかしながら従来プロセッサと比べ面積あたり性能は約 1.6 倍，電力あたり性能は約 6.1 倍に向上することがわかった．

謝　辞

なお，本研究の一部は研究成果最適展開支援事業（課題番号 221Z0021）および科学研究費補助金（若手研究（B）課題番号 22700053）による．本研究は東京大学大規模集積システム設計教育研究センターを通し，シノプシス株式会社および日本ケイデンス株式会社の協力で行われたものである．

参　考　文　献

1) Seiler, L., Carmean, D., Sprangle, E., et al.: Larrabee: A Many-Core x86 Architecture for Visual Computing, *IEEE Micro*, Vol. 29, No. 1, pp. 10–21 (2009).
2) Hyunchul, P., Yongjun, P. and Scott, M.: Polymorphic pipeline array: a flexible multicore accelerator with virtualized execution for mobile multimedia applications, *MICRO 42*, pp. 370–380 (2009).
3) 岩上拓矢，吉村和浩，上利宗久，中田 尚，中島康彦：プログラマビリティを備える低電力アクセラレータの評価, SACSIS2010 論文集 (poster), pp. 103–104 (2010).
4) Kazuhiro, Y., Takuya, I., Takashi N., Jun Y., Hajime S., Yasuhiko, N.:An Instruction Mapping Scheme for FU Array Accelerator, IEICE Transactions on Information and Systems, Vol. E94-D, No. 2, pp. 286–297 (2011).
5) Wilton, S.J.E. and Jouppi, N.P.: CACTI : an enhanced cache access and cycle time model, *IEEE Journal of Solid-State Circuits*, Vol. 31, No. 5, pp. 677–688, (1996).
6) Krisztin, F., Nam, S. K., Steve, M.,David, B., et al.: Drowsy caches: simple techniques for reducing leakage power, *SIGARCH Computer Architecture News*, Vol. 30, No. 2, pp. 148–157 (2002).