情報処理学会論文誌　コンピューティングシステム　Vol.5　No.4　13–23 (Aug. 2012)

時分割実行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[1]　吉村 和浩[2,†1]　中田 尚[2,†2]　中島 康彦[2,a)]

受付日 2012年1月17日, 採録日 2012年4月24日

概要：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画像処理向け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させている．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の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 1.17 倍に向上させることができた．

キーワード：アクセラレータ，仮想化，時分割実行，演算器アレイ

Improvement of FU Array Accelerator by Time Division Execution

Takuya Iwakami[1]　Kazuhiro Yoshimura[2,†1]　Takashi Nakada[2,†2]
Yasuhiko Nakashima[2,a)]

Received: January 17, 2012, Accepted: April 24, 2012

Abstract: 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 117% of the baseline LAPP.

Keywords: accelerator, time division execution, virtualization, FU array

[1] 株式会社フィックスターズ
Fixstars Corporation, Shinagawa, Tokyo 141–0032, Japan
[2] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology, Ikoma, Nara 630-0192, Japan
[†1] 現在，株式会社富士通研究所
Presently with FUJITSU LABORATORIES LTD.
[†2] 現在，東京大学
Presently with The University of Tokyo
[a)] nakashim@is.naist.jp

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，MIC [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．汎用プロセッサをもと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が，電力あたり性能はメニコア化のオーバ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専用のコンパイラやプログラミング言語の開発が必要である．

先行研究 [3], [4] において，既存バイナリをほぼそのまま利用しつつ高い電力あたり性能を実現する画像処理向け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 [5]．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 章では関連研究を紹介する．3 章では従来型 LAPP の概要を述べる．4 章では従来型 LAPP の問題点を述べ，それをベースに仮想化機構について詳述する．5 章では評価モデルとパラメータについて述べ，6 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，7 章で本稿のまとめを行う．

2. 関連研究

限られた計算資源を用いて，仮想的により大きな計算を行う手法として，時分割実行は非常に有効な手段である．特に CGRA を含む汎用アクセラレータにおいては，適用範囲をより大きくするために重要であり，多くのアクセラレータにおいて利用されている．

動的再構成が可能な汎用アクセラレータとしては FPGA があげられる．たとえば Xilinx 社は Virtex-4 以降の FPGA で動的再構成をサポートしており [6]，同時に利用されない回路を個別に用意することにより，実行の流れに応じて，対応する回路を再構成しつつ実行を行うことができる．これにより，より小さな FPGA で所望の回路を実現することが可能である．しかし，再構成にはミリ秒オーダの時間が必要であるため，頻繁に再構成を行うことは現実的ではなく，機能・モジュール単位での再構成が想定されている．そのため，本稿で対象とするような最内ループを時分割実行する目的とは相容れない技術である．

FPGA と比較して演算器の粒度が粗い CGRA においては，より時間粒度の細かい 1 サイクル単位での動的再構成技術が利用されている．

たとえば，ADRES [7], [8] では 4 命令発行の一般的な VLIW 構成のバックエンドとして，4×3 程度の CGRA を接続している．搭載演算器数が比較的少ないために，必然的に時分割実行機能を搭載している．専用コンパイラが，制限付きの C 言語により記述されたプログラムから，4×3 命令発行に対応する CGRA 命令を生成するが，1 つのループに対して複数の CGRA 命令を生成することにより複数サイクルで 1 つのイタレーションの実行が行われる．

PPA [2] では，2×2 の CGRA を 8 コア接続し，ループカーネルに応じてアレイサイズの構成を変更できる．そして，異なるループカーネルをパイプライン実行することで，対応可能なループサイズの拡大と演算器の動作率の向上の両立を実現している．

これらの CGRA では，複数サイクルを使って 1 イタレーションを実行するため，どの命令をどのサイクルで実行するのかということを，専用コンパイラなどを用いて構成情報として生成する必要がある．しかしループカーネルが大きくなればそのスケジューリングは非常に困難である．本稿では従来型 LAPP の既存機械語コードを利用可能であるという特徴を保持したまま，物理演算器数を超える命令を含むループカーネルの実行を時分割実行を用いた仮想化機構で実現する．

3. 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．

3.1 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トリガとして遷移する．通常プロセッサにおけるプリフェッチ命令は，先頭アドレスと長さで示される単一の配列に対するプリフェッチを行うが，LAPP ではこのプリフェッチ命令に副作用のない演算命令（gr0 を書き込み先とする addi 命令など）を同一 VLIW 命令にパックすることにより，追加した演算命令のソースレジスタや即値のフィールドに 2 番目以降の入力配列や出力配列のプリフェッチ情報を付加する．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．

3.2　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．

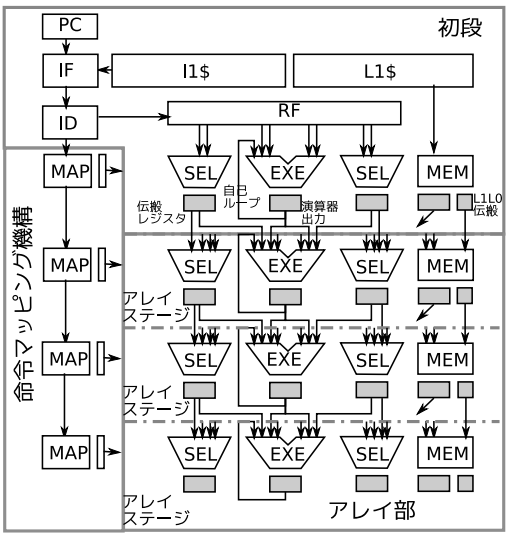

図 1　LAPP の構造
Fig. 1　Execution model of LAPP.

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**からなる．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あたる．これらパイプラインレジスタ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**割当て情報**はここに格納される．命令割当ての詳細については文献 [9] を参照されたい．

3.3　低消費電力機能

LAPP の各ユニットは実行時には電源オン，DVS（Dynamic Voltage Scaling）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 PG（パワーゲーティング）によって行い，クロックのオンオフは CG（クロックゲーティング）によって行う．初段の演算器や L1$は，通常実行からアレイ実行を通して使用されるためつねに電源オンの状態にある．I1$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フとして待機する．

4. 仮想化機構

本章では，従来型 LAPP に存在する物理演算器数による制約について述べた後，この制約を解消するための仮想化機構のアイデアと具体的な構成について述べる．

4.1 従来型 LAPP における制約

3 章で述べた従来型 LAPP では設計時に決定される物理演算器数が制約となり，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．

対象プログラムの変更によってこの問題を解決するためには，物理アレイ段数に収まるようにループを分割することが考えられるが，そのためにはループ途中の中間結果をいったん L1 キャッシュに書き出す必要がある．一般に最終結果よりも中間結果の方がサイズが大きいため，中間結果の書き出しと後続ループでの再読み込みは無視できないオーバヘッドとなる．また，中間結果が L1 キャッシュに格納されることにより，キャッシュ容量を圧迫し入出力データの再利用率を低下させることも考えられる．

さらに，現在の実装ではアレイ実行における 1 イタレーションあたりの出力を 1 ワードに限定することにより L1 キャッシュの書き込みポートを削減しており，これを拡張するためには書き込みポートを増やすか，複数サイクルをかけて書き込みを行う必要がある．

以上の理由により，物理演算器数を超えるループを単純な分割によりアレイ実行することは困難である．仮に拡張を行ったとしてもハードウェアが複雑化するとともに，実行サイクル数についても中間結果の一時保存のために分割数を大幅に超えるオーバヘッドが必要となることが予想される．したがって，本稿では物理演算器数を超えるループについてはアレイ実行を行わないこととする．

4.2 仮想化のアイデア

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パスを追加し制御を変更するだけなので，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 とおくと，仮想化によって演算器アレイの段数を超える $L \times N$ 命令までの高速実行を可能にする．

4.3 複数命令割当て

仮想化 LAPP における複数命令割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）やアドレス計算などは省略した．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ていないことを表している．

4.4 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．

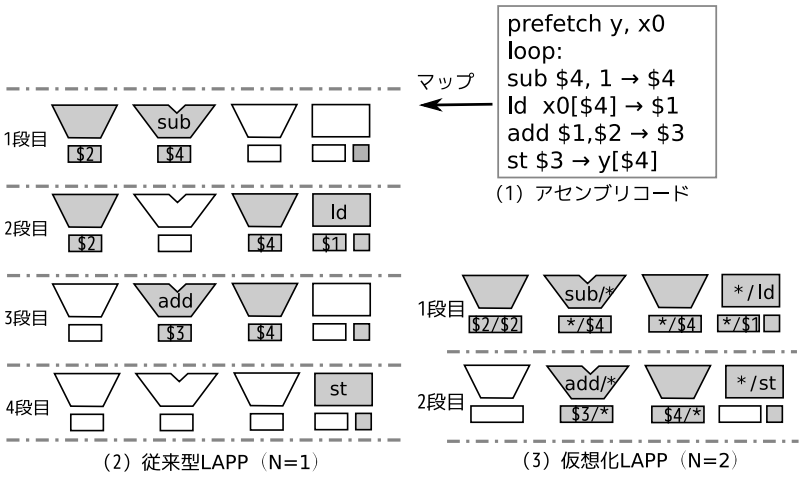

図 2　複数命令割当ての例

Fig. 2　Example of mapping multiple instructions.

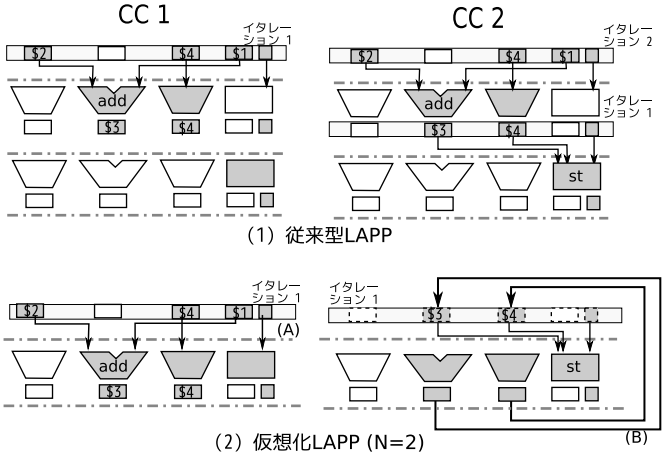

図 3　時分割によるアレイ実行
Fig. 3　Time division array execution.

従来型 LAPP では図 3 (1) のように CC（Clock Cycle） 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．

4.5　時分割実行機構の全体構成

時分割実行による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とマッパ（MAP）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．

4.6　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，ループカーネルの VLIW 命令数を M，物理段数を L とすると，式；$N = \lceil M/L \rceil$ から決定される．M≤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

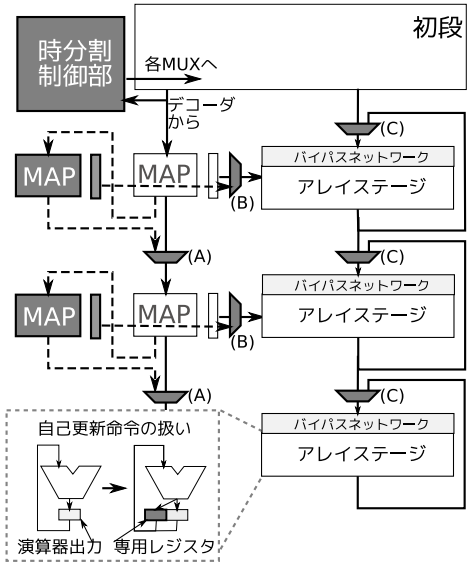

図 4　時分割実行機能を備えた LAPP（$N_{max} = 2$）
Fig. 4　LAPP with time division execution ($N_{max} = 2$).

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．具体的には，従来型 LAPP のプリフェッチ命令は通常プロセッサにおけるプリフェッチ命令と副作用のない演算命令を同一 VLIW 命令にパックすることにより実現されていることを利用し，パックする命令を追加することにより互換性を維持したまま拡張することが可能である．

4.7　命令マッピング機構

従来型 LAPP では 1 つのマッパがアレイ 1 段の設定情報を生成しており，アレイ段とマッパの対応は 1 対 1 であっ

た．ここに時分割実行機構を組み込むと，この対応関係が1対多の対応となり，命令マッピング機構のデータパスが大きく変更される．時分割実行においては，1段のアレイステージに対して N_{max} 個のマッパを置く．このようにすることにより，個々のマッパは多重化度 N に応じて L × N 段の従来型 LAPP と同じ動作となる．アレイ設定時には L × N 個のマッパを直列に接続する必要がある．このために N_{max} ポートの MUX（A）を挿入する．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．また，アレイ実行時は各段で N 個の割当て情報を切り替えながら動作する．具体的には図 4 のように N_{max} ポートの MUX（B）がマッパとアレイステージの間に配置される．N=1 のときは実線のパスがつねに選択され，N=2 のときは実線と点線のデータパスが交互に選択される．

4.7.1　アレイステージ

図 2 (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身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．

4.7.2　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，サイクルごと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，演算器出力のほかに N_{max}–1 個のバッファを用意する．

5.　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と評価モデルについて述べる．

5.1　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 1 ロードストアユニット（アドレス計算ユニット（EAG）を含む），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（FI–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 カーネルからなる．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．

表 1　RTL シミュレータの諸元
Table 1　Parameters of RTL simulator.

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	8 bytes/cycle
命令キャッシュ	4 way 16 KB（64 bytes/line）
L1 キャッシュ	4 way 16 KB（64 bytes/line）
L1L0 転送	16 bytes/cycle
ストアバッファ	4 entries
アレイ部	
命令写像設定	2 cycle/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	4 blocks × 16 words
L0 伝搬	16 bytes/cycle
ストアバッファ	4 entries
最終段から L1 への転送	4 bytes/cycle

5.2　性能モデル

性能指標としては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．この RTL シミュレータの諸元を**表 1** に示す．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．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．

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

5.3　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 × 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つね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モデルをもとに見積もった．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5.4節で述べる電力見積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180 nm，1.8 Vテクノロジ，100 MHz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 [10]から見積もっている．また，PGを適用する回路の面積オーバヘッドは文献[11]から15%とした．

5.4　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．これは各ユニットが動作したときの電力（μW）であり，RTLモデルのシミュレーションの結果からPrimeTime PX 2007.12-SP03によって見積もり，I1$とL1$のキャッシュ本体に関しては回路規模と同様にCACTI 3.2によって見積もった．また各ユニットの動作履歴はRTLシミュレータで計測した．以上により，RTLシミュレータから得られた動作履歴と電力パラメータを使って，平均消費電力を求める．本稿で用いる電力モデルではDVSが適用されたユニットの消費電力は，文献[12]からオン時の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は静的消費電力のみを消費するが，今回用いた180 nmテクノロジにおける静的消費電力は0.3%未満[13]であるので0%と仮定し，PGにより電源の供給がないユニットは0%と仮定する．

表 2　回路規模および電力見積り
Table 2　Area and power estimations.

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	1×1.15	2,235	2,209
ALU	3	3×1.15	10,765	6,373
MEDIA	4	4×1.15	7,677	4,983
BRC	1	1×1.15	1,557	1,287
伝搬レジスタ	0	20×1.15	1,900	3,920
MEM, L0$	0	1×1.15	34,652	4,952
MAP（1段）	-	-	24,700	26,200
		初段合計	814,873	
		アレイステージ（1段）	160,364	
		命令マッピング機構（36段）	1,022,580	

6.　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．

6.1　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（10^6 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，約65%となっている．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，それぞれ45%，38%となっている．

一方，従来型の6，9，18段は命令割当て部とアレイ部がそれぞれ段数に比例した回路規模となっている．初段部分は仮想化と同様にすべての構成で同一であり，仮想化のための追加回路は存在しない．

6.2　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

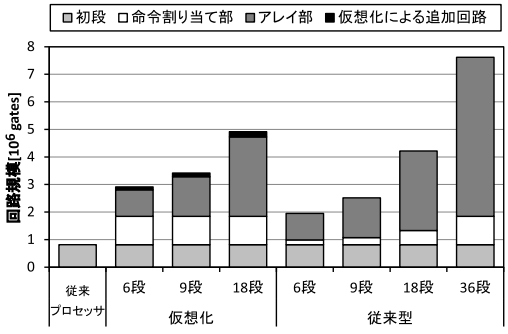

図 5　各構成の回路規模内訳
Fig. 5　Breakdown of area.

表 3　ベンチマークごとの平均 IPC
Table 3　Performance evaluations.

ベンチマーク	命令数	従来プロセッサ	仮想化			従来型			
			6 段	9 段	18 段	6 段	9 段	18 段	36 段
FI–1	20	1.4	4.6	6.1	9.1	1.4	1.4	1.4	12.1
FI–2	33	1.3	7.2	10.2	17.7	1.3	1.3	1.3	28.1
FI–3	10	1.3	5.6	5.6	9.0	1.3	1.3	9.0	9.0
Z	29	1.9	10.2	14.2	23.6	1.9	1.9	1.9	35.1
S	27	1.7	7.9	11.0	18.3	1.7	1.7	1.7	27.3
M	23	2.0	6.4	8.9	14.9	2.0	2.0	2.0	22.3
E	14	1.2	3.1	7.6	11.9	1.2	1.2	11.9	11.9
N	26	2.0	5.3	6.9	9.8	2.0	2.0	2.0	12.4
幾何平均		1.6	5.9	8.4	13.5	1.6	1.6	2.7	17.7

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．

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 3.7 から 8.4 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 は従来どお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わらないので，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．

その一方で従来型 6 段と 9 段では，アレイ実行ができなかったためにすべてのベンチマークにおいて従来プロセッサと同一の性能となっている．従来型 18 段では比較的ループサイズの小さい FI–3 と E でアレイ実行が行えたが，それ以外のベンチマークではアレイ実行が行えなかった．その結果，従来型 18 段でも平均 1.7 倍の高速化率にとどまっている．

以上の結果から従来型 LAPP の段数をそのまま減らすことは，許容できない性能低下を引き起こすことが明らかとなった．したがって，以降では従来型 36 段と仮想化 LAPP についての評価のみを行う．

6.3　消費電力比較

6.3.1　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，段数の増加にともない消費電力も増える．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 段が消費電力は最も小さく，従来プロセッサの約 1.03 倍となっている．

表 4　ベンチマークの平均消費電力（mW）
Table 4　Power evaluations.

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

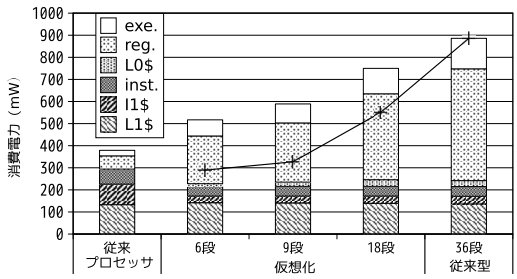

図 6　ベンチマーク S の消費電力内訳
Fig. 6　Power breakdown of benchmark S.

6.3.2　消費電力の内訳

仮想化によるオーバ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詳しく述べる．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$は 1 次データキャッシュ，I1$は 1 次命令キャッシュ，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つね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．これは段数が増えることにより同時に動作するユニット数が増えるためである．

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ヘッドが発生しているものと考えられる．

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．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，これらの命令が割り当てられたアレイ段では 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．

この理由から，1 段あたりの消費電力が従来型 36 段と比較して増加するため，段数が $1/N_{max}$ となっても消費電力は $1/N_{max}$ よりも大きくなる．このように仮想化によるオーバヘッドは，アレイ実行の時間が増加することと複数命令割当てが関係しており，仮想化の構造に起因している．

6.4　総合評価

6.3 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・電力の減少率を**図 8**，アレイステージの利用率を**表 5** に示す．

6.4.1　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 1.6 倍の性能向上となっている．それに加えて従来型 36 段との比較においても面積あたり性能が約 17%高くなっている．

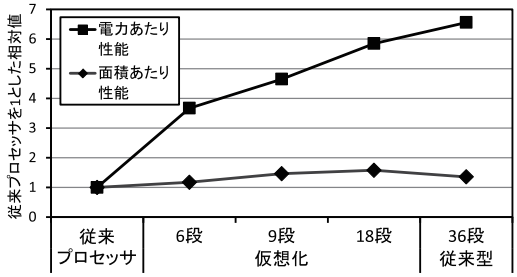

図 7　面積あたり性能と電力あたり性能
Fig. 7　Performance per area and power.

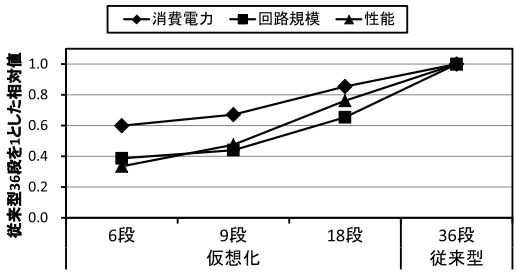

図 8　従来型 36 段比の減少率
Fig. 8　Reduction ratio from existing model.

表 5　アレイステージ使用率
Table 5　Utilization rate of array stages.

	仮想化			従来型
	6 段	9 段	18 段	36 段
平均使用段数	5.63	7.75	13.13	22.75
平均使用率	94%	86%	72%	63%

図 8 によると，仮想化 LAPP は必要に応じて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ことがあるものの，実装されるアレイステージが減少するため回路規模は小さくなっている．仮想化 18 段では性能低下の割合が，回路規模の削減割合を上回っており，面積あたり性能が従来型 36 段比で向上している．つまり，従来型では使用率が低く性能向上への貢献が少なかった後半の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と考えられる．

6.4.2　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 倍の電力で約 8.4 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 5.6 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 6.3 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．

また，電力あたり性能の低下は，ベンチマーク実行中の電力の増加と性能の低下を意味しており，あるベンチマークを完了するまでの電力量の増加に直結する．すなわち，電力あたり性能の低下率の逆数は，電力量の増加率と等しくなる．

6.4.3　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，この利用率は各構成の物理段数に対するループカーネルが割り当てられた段数の割合であり，4.6 節で定義した変数 N，M，L を用いると (M/N)/L で求められる．表 5 の利用率によると，仮想化 LAPP では仮想化の多重度が増えるに従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，物理段数が減少し利用頻度の低いアレイステージが削減されたため利用率が大きく改善されている．しかし仮想化 9 段，6 段では利用率の向上がそれぞれ 86%，94%と高止まりするため，多重化度の増加によってオーバヘッドが増加し，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．

7.　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した．

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，最大多重化度を 2 としたときに，物理段数が 1/2 となり面積が約 65%に削減されるとともに，面積あたり性能は 1.17 倍に向上した．その一方，仮想化のオーバヘッドによって電力あたり性能は従来型 LAPP の 0.89 倍となった．

謝辞　なお，本研究の一部は先端的低炭素化技術開発（次世代低電力デバイス安定化計算機構成方式）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，ローム株式会社，および日本ケイデンス社の協力で行われたものである．

参考文献

[1] Skaugen, K.: Petascale to Exascale: Extending Intel's HPC commitment, ISC 2010 keynote (2010), available from ⟨http://download.intel.com/pressroom/archive/reference/ISC_2010_Skaugen_keynote.pdf⟩.

[2] Park, H., Park, Y. and Mahlke, S.: Polymorphic pipeline array: A flexible multicore accelerator with virtualized execution for mobile multimedia applications, *Micro-42: Proc. 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pp.370–380 (2009).

[3] 中田　尚，上利宗久，中島康彦：画像処理向け線形アレイ VLIW プロセッサ，先進的計算基盤システムシンポジウム SACSIS2009，pp.293–300 (2009).

[4] 中田　尚, 吉村和浩, 下岡俊介, 大上　俊, Naveen, D.V.R., 中島康彦：画像処理向け線形アレイアクセラレータの性能評価，情報処理学会論文誌 コンピューティングシステム，Vol.5, No.3 (ACS38), pp.74–85 (2012).

[5] 岩上拓矢，吉村和浩，中田　尚，中島康彦：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，先進的計算基盤システムシンポジウム SACSIS2011，pp.136–143 (2011).

[6] Dye, D.: Partial Reconfiguration of Xilinx FPGAs Using ISE Design Suite, Technical Report WP374 (v1.1), Xilinx White Paper (2011).

[7] Bouwens, F., Berekovic, M., Sutter, B.D. and Gaydadjiev, G.: Architecture Enhancements for the ADRES Coarse-Grained Reconfigurable Array, *HiPEAC'08*, pp.66–81 (2008).

[8] Mei, B., Vernalde, S., Verkest, D. and Lauwereins, R.: Design Methodology for a Tightly Coupled VLIW/Reconfigurable Matrix Architecture: A Case Study, *DATE*, pp.1224–1229 (2004).

[9] Yoshimura, K., Iwakami, T., Nakada, T., Jun, Y., Shimada, H. and Nakashima, Y.: An Instruction Mapping Scheme for FU Array Accelerator, *IEICE Trans. Information and Systems*, Vol.94-D, No.2, pp.286–297 (2011).

[10] Shivakumar, P., Jouppi, N.P. and Shivakumar, P.: CACTI 3.0: An Integrated Cache Timing, Power, and Area Model, Technical report, Technical Report 2001/2, Compaq Computer Corporation (2001).

[11] Zhao, L. et al.: Geyser-2: the second prototype CPU with fine-grained run-time power gating, *Proc. 16th Asia and South Pacific Design Automation Conference*, pp.87–88 (2011).

[12] Flautner, K., Kim, N.S., Martin, S., Blaauw, D. and Mudge, T.: Drowsy caches: Simple techniques for reducing leakage power, *SIGARCH Computer architecture News*, Vol.30, pp.148–157 (2002).

[13] Tsai, Y.-F., Duarte, D., Vijaykrishnan, N. and Irwin, M.: Impact of process scaling on the efficacy of leakage reduction schemes, *ICICDT '04, International Conference on Integrated Circuit Design and Technology*, pp.3–11 (2004).

岩上 拓矢

2011 年奈良先端科学技術大学院大学情報科学研究科博士課程前期修了．同年株式会社フィックスターズ入社．在学中は低電力アクセラレータに関する研究に従事．

吉村 和浩　（正会員）

2007 年龍谷大学理工学部電子情報学科卒業．2009 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．2012 年同研究科博士後期課程修了．博士（工学）．同年株式会社富士通研究所入社．現在に至る．計算機アーキテクチャとプロセッサ開発に興味を持つ．電子情報通信学会正員．

中田 尚　（正会員）

2004 年豊橋技術科学大学大学院工学研究科情報工学専攻修士課程修了．2007 年同大学院工学研究科電子・情報工学専攻博士後期課程修了．博士（工学）．同年奈良先端科学技術大学院大学情報科学研究科助教．2012 年東京大学大学院情報理工学系研究科特任助教．現在に至る．計算機アーキテクチャとシミュレーションに関する研究に従事．ACM 会員．

中島 康彦　（正会員）

1986 年京都大学工学部情報工学科卒業．1988 年同大学大学院修士課程修了．同年富士通（株）入社．VLIW 型プロセッサ，命令エミュレーション，高速 CMOS 回路設計などに関する研究開発に従事．工学博士．1999 年京都大学総合情報メディアセンター助手．同年同大学大学院経済学研究科助教授，2006 年奈良先端科学技術大学院大学情報科学研究科教授（コンピューティング・アーキテクチャ講座担当），現在に至る．計算機アーキテクチャに興味を持つ．電子情報通信学会，IEEE-CS，ACM 各会員．