A-03

線形アレイ型 VLIW プロセッサの面積評価

An Area Evaluation for Linear Array VLIW Processor

上利　宗久 †1　　中田　尚 †1　　中島　康彦 †1
Munehisa AGARI　Takashi NAKADA　Yasuhiko NAKASHIMA

1. はじめに

近年の画像処理や科学技術計算では，単一コア性能が頭打ちとなり，アクセラレータとしてメニコアプロセッサ[1),2)] や小規模のベクトル演算[3)] が利用されている．これらのプロセッサでは，一般的な並列プログラミング手法が利用できる柔軟性がある代わりに，演算器に十分なデータを流し込むために広帯域のデータパスが必要であり，高消費電力であるという問題がある．また，GPGPU に代表されるアクセラレータの性能を最大限に引き出すためには，人手によるアーキテクチャ毎のチューニングが不可欠[4)] である．一方，リコンフィギュアラブルデータパス[5)] は，膨大な演算器を配置することで，専用ハードウェアと同様に，比較的低い動作周波数でも高性能プロセッサに匹敵する性能を達成し得る利点がある．しかし，機械語命令を実行する汎用プロセッサとの接続性に難点があり，広く普及しているとはいえない．

以上の背景から，我々は機械語命令レベルの互換性を維持しつつ，膨大な演算器を効率よく動作させるアーキテクチャとして，線形アレイ型 VLIW プロセッサを提案している[6)]．本稿では，まず 2 章において提案プロセッサの特徴と概要について述べる．3 章ではシミュレータによる予備評価結果をもとに，提案プロセッサの面積削減手法を提案する．4 章では FPGA への実装に向けた設計状況をもとに，面積規模を見積もり，実装が十分に可能であることを示す．

2. 線形アレイ型 VLIW プロセッサ

提案プロセッサは，図 1 のように，既存の VLIW プロセッサの演算器群を直列に接続した構造である．従来型 VLIW では演算器バイパスとして実現される演算器間ネットワークを演算器と演算器の間にレジスタを挟み込む複数段の演算器アレイに展開し，命令デコーダを流用してループ構造の全命令を各演算器に写像することで，段間ネットワークを動的に構築する仕組みを備える．本プロセッサの特徴は，上記構造に対し，ループの回転数に相当する入力データを初段から，順次流し込むことにより，機械語命令レベルの互換性を保ったまま，毎サイクル 1 つの処理結果を得られる点である．

提案プロセッサの課題として，キャッシュならびにレジスタファイルの分散配置が必要なことがある．任意の VLIW 命令列をアレイに写像するためには，任意の段にロード命令を配置する必要があるが，数多くのロード命令が単一の L1 キャッシュ

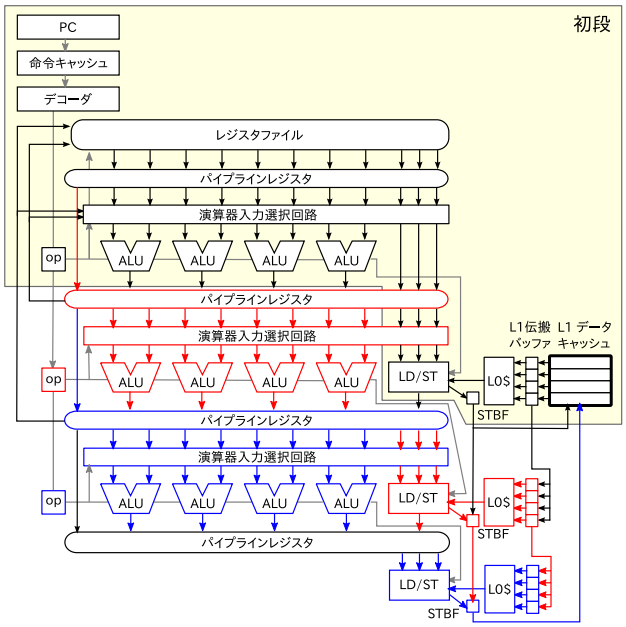

図 1　線形アレイ型 VLIW プロセッサの概要

を直接参照することは，キャッシュメモリに必要なポート数の点から非現実的である．各段の演算器が 1 組のレジスタファイルを参照することも，同様に非現実的であり，キャッシュやレジスタファイルを各段に分散配置する仕組みが必要となる．

しかし，レジスタやキャッシュがプロセッサ内で分散していても，同一イタレーションに関与する VLIW 命令に対して，いずれの段においても同一内容を見せることができれば，通常 VLIW 動作と論理的に同じ演算結果が得られる．すなわち，初段の演算器は，アレイ実行の第 1 サイクルでは，第 1 イタレーションに対応したデータを使用でき，第 2 サイクルでは第 2 イタレーションに対応したデータを使用できればよい．このためには，アレイ構造に配置した VLIW 命令間を演算状態が伝搬するのと同じ速度で，レジスタ，キャッシュおよびストアバッファの内容を併走させればよい．提案プロセッサは，イタレーション間に依存関係が無いことを前提とすることで，演算器ネットワークを簡素化し，性能見通しの良さと，優れた演算効率を目指すアーキテクチャである．

2.1 アレイ動作の詳細

提案プロセッサは，大きく分けて，**通常実行**，**アレイ設定**，**アレイ実行**の 3 つの動作モードを備える．**通常実行**では，初段のみが動作し，既存の VLIW プロセッサと同様に動作する．初段はレジスタファイルと L1 キャッシュを備え，既存プロセッサと異なる点は，演算器アレイとの接続のみである．

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

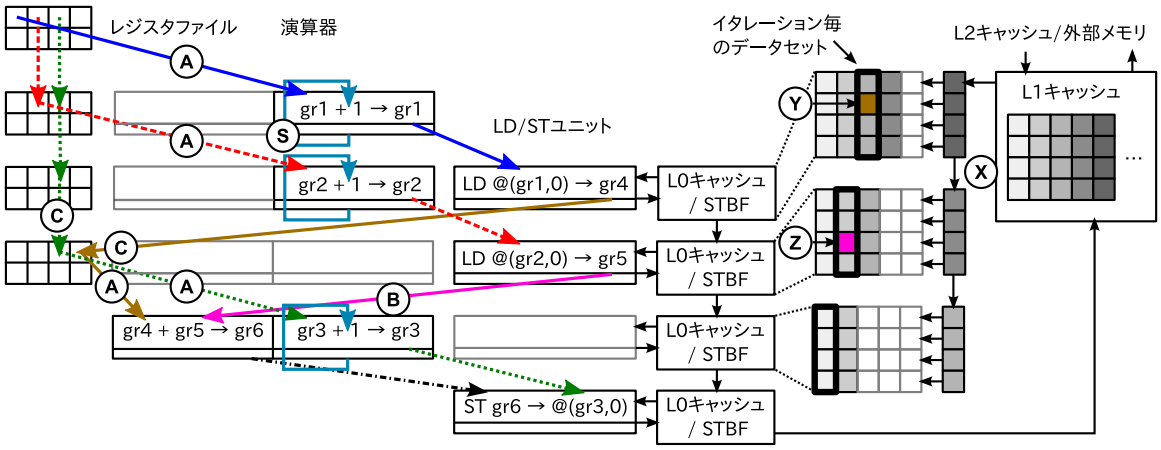

図 2　アレイ動作中のデータフロー

通常実行において，デコーダがアレイ起動用のヒント命令を検知すると，アレイ設定を開始する．アレイ設定では，図 2 のようにヒント命令の次命令以降をループカーネルとして，各段の演算器へと 1 命令ずつ割り当てる．そして，無条件前方分岐命令をループの終端と見なし，その命令までの命令列を用いて，演算器ネットワークを構築する．ヒント命令はプリフェッチ命令を拡張した命令であり，入力データのフェッチ方法と，出力アドレス範囲の指定を行う．この指定を元に，アレイ実行中は初段の 4Way L1 キャッシュを実質的には，Way0 を入出力バッファ，Way1-3 を入力バッファとして利用する．なお，アレイ動作開始前に Way0 の Dirty ラインを追い出し出力バッファを確保し，Way0-3 へのプリフェッチにより入力データを確保する．この仕組みにより，アレイ実行中のキャッシュミスが無いことを保証し，アレイの制御を簡略化している．

プリフェッチと演算器ネットワーク構築の双方が完了次第，アレイ実行を開始する．各段は，小規模の L0 キャッシュとストアバッファ(STBF) を備える．L1 キャッシュは，初段の L0 キャッシュと，最終段のストアバッファとのみ接続されており，データの流れは一方向のループである．L1 キャッシュは毎サイクル，ヒント命令を元にデータセットの差分を Way0-3 から初段の L0 キャッシュへと送り出す．アレイ各段は，演算を行いながら，この差分とストアバッファを後段へと順次伝搬する．最終段のストアバッファの内容は，初段の L1 キャッシュへと書き戻され，1 イタレーションの実行が完了する．

アレイ内でループ外への分岐の成立を検出すると，アレイ実行の終了手続きを開始する．まず，分岐先を再開アドレスとして記録し，アレイ実行の停止指示とともに次段以降に伝搬する．ただし，この時点では分岐の成立したイタレーションより手前のイタレーションはまだ実行中であるため，即座にアレイ実行を止めることはできない．停止指示の伝搬が最終段に到達するのと同時に，最終イタレーションの実行が完了するので，最終段から初段に対し，アレイ実行の停止と再開アドレスを通知する．そして初段は，再開アドレスから通常実行へと復帰し，アレイ実行が終了する．

2.2　値 の 伝 搬

提案プロセッサでは，FR-V[7] 準拠の機械語命令を用いる．ただし，LD/ST ユニットは 1 つ，汎用レジスタは 32 本，メディアレジスタは 32 本などと仕様を一部変更している．

アレイ実行中，各演算器は前段から計算状態を受け取り，VLIW 1 命令分の演算結果を後段へと送る．ただし，演算器は図 2 に示すように，同一段のレジスタ (A) ないしは，前段の演算器出力 (B) からのみ入力を得ることができる．そのため，直接利用できないレジスタ値は，段間でレジスタの値を後段へと伝搬する必要がある (C)．キャッシュの伝搬は，帯域削減のためにイタレーション間の差分データのみを転送する (X)．各段は，小規模の L0 キャッシュに L1 キャッシュの内容を部分的に持ち，段毎に異なる世代のデータセットからデータを読み出すことができる (Y,Z)．ストアバッファは，毎サイクル 1 出力できればよいので 1 エントリとした．

2.3　レジスタ値更新

アレイ動作中は，イタレーション間の依存関係には対応しないものの，ループカウンタや LD/ST アドレスの更新は必要である．即値を用いたレジスタ値更新命令（Reg = Reg ± Imm）を可能とするために，図 2 中 (S) のように，自身へのバイパスを設ける．このバイパスによりサイクル毎の Imm を増減し，ループカウンタやアドレス更新に対応する．

2.4　シミュレータによる予備評価

外部バスとの転送速度が 8bytes/cycle，L1 キャッシュが 4way 32KB，L0 キャッシュが 4 block 128B，L0 間転送は 16bytes/cycle，といった現実的な仕様を仮定した RTL シミュレータにより，32 段のアレイについて，320 × 240 画素で，各画素は RGB1 バイトを含む 4 バイトである画像に対し，拡大縮小・鮮鋭化・ノイズ除去といった画像処理プログラムを用いて評価した結果，初段のみの通常実行の平均 IPC は 1.1〜2.0，アレイ実行時の最大 IPC は 23〜76，平均 IPC は 8.1〜33.9 であり，イタレーション間に依存のないプログラムに対し，高い性能を達成できることが分かった[6]．

表 1　アレイ一段の演算器構成と，入力レジスタ数

	LD/ST	ALU × 3	BRC	MEDIA × 4	計
GR 入力	3	6	2	0	11
MR 入力	1	0	0	8	9

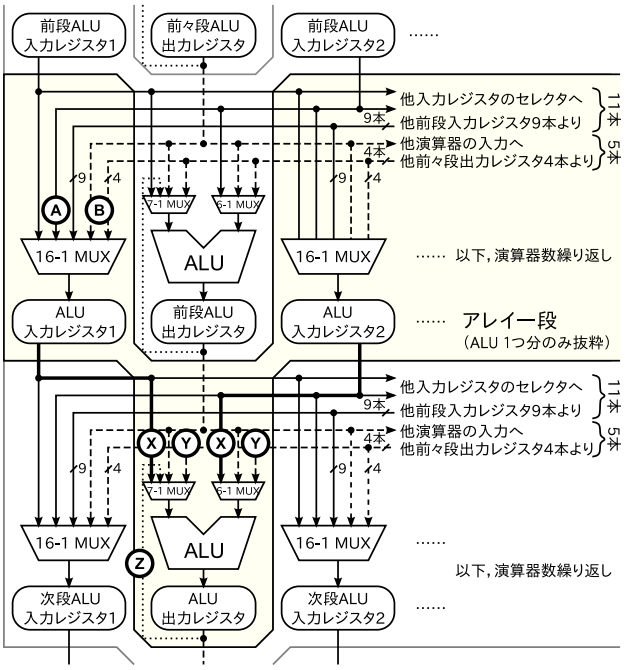

図 3　演算器の入出力レジスタと段間ネットワーク

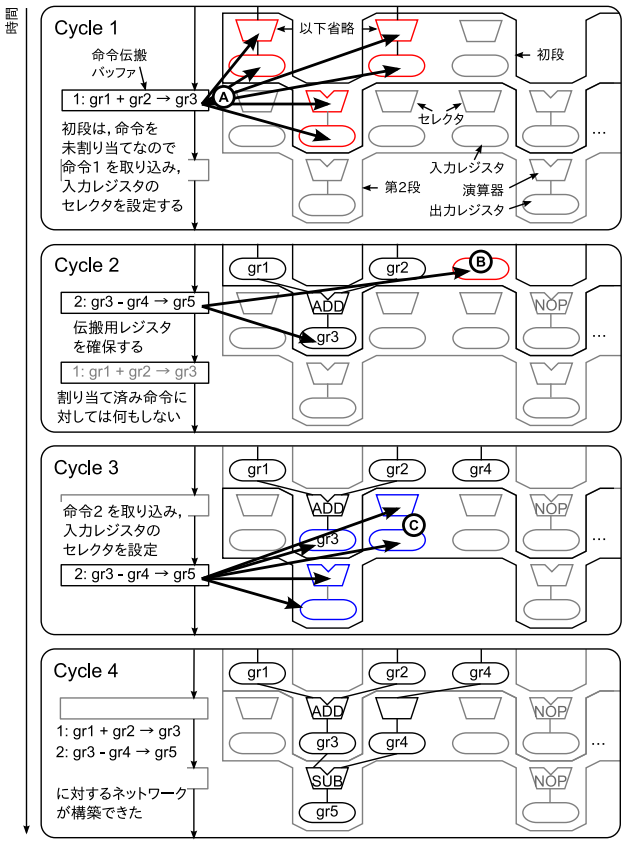

図 4　段間ネットワークの構築例

一方で，レジスタ値伝搬に必要なハードウェアが大きく，実際に伝搬に必要なレジスタのみを伝搬する仕組みを実装しなければならない．本稿では FPGA への実装に向け，回路規模を抑えつつ，レジスタ値を伝搬する設計について検討する．

3. 物理レジスタ数の削減

各段の演算器の構成を表 1 に示す．ロードストアユニット (LD/ST) はアドレス計算用に GR 入力レジスタを 2 つと，ストア値のために GR,MR 入力レジスタをそれぞれ 1 つずつ備える．3 個の ALU および 1 つの分岐ユニット (BRC) はそれぞれ 2 つの GR 入力レジスタを持ち，4 個のメディア演算器 (MEDIA) は合計で 8 つの MR 入力レジスタを持つ．すなわち，各段には GR 入力レジスタ 11 本，MR 入力レジスタ 9 本が必須であり，レジスタファイルを用いて毎サイクル動作を行うためには，例えば，汎用レジスタファイルには 11R5W のポートが必要であり，回路規模は大きくなるだろう．

しかし，画像処理プログラムによる評価の結果，各段に必要なレジスタは伝搬用途を含めても，汎用レジスタ (GR) が最大 7 本，メディアレジスタ (MR) が最大 9 本であることが分かった．すなわち，最低限必要となる演算器の入力レジスタで，伝搬に必要なレジスタ数を満足できるため，空き入力レジスタを伝搬に利用する手法を提案する．

3.1 入力レジスタによる段間ネットワークの構成

ALU を例に段間ネットワークの構成を図 3 に示す．レジスタファイルの代わりに入力レジスタを用い，入力は前段からの伝搬用である前段入力レジスタ値 (A) と，演算結果の伝搬用である前々段演算器出力レジスタ値 (B) から選択する．汎用レジスタであれば前段入力レジスタは 11 個，前々段演算器出力レジスタは 5 個であり，新しいレジスタ値の選択には，16-1 セレクタが必要である．これは，入力レジスタ間に演算器と同程度の遅延が許容されるため，配線などの制約は厳しくないと考えた構成である．各演算器は，割り当てられた命令を元に，ポートの入力レジスタ (X)，前段演算器出力レジスタ 5 つ (Y)，および自身のバイパス (Z) から入力を選び演算を行う．

3.2 段間ネットワークの構築

初段で内部形式にデコードされた命令を，初段から最終段へ伝搬しながら,「命令取り込み」と「伝搬ネットワーク構築」の 2 段階で，段間ネットワークの構築を行う．まず，命令を受け取った段は，その段に命令が未割り当てであれば，命令を取り込み，演算器には命令を，また，入力レジスタにはソースオペランドを割り当てる．新しく割り当てられたレジスタには，前段以前からの伝搬が必要である．そこで，命令が割り当て済みの段では，伝搬命令のソースオペランドを後段で必要となるレジスタとして，空き入力レジスタを伝搬用レジスタに割り当て，前段入力レジスタあるいは前々段出力レジスタから値を受け取れるように，セレクタを設定する．

命令列 $gr1 + gr2 \rightarrow gr3; gr3 - gr4 \rightarrow gr5;$ に対する，命

表 2 基準プロセッサと提案プロセッサの cell area 見積もり

	逐次コア	追加コア	32 コア換算
提案プロセッサ	873,992	315,537	10,655,639
基準プロセッサ	829,217	829,217	26,534,931

令割り当てと段間ネットワーク構築の例を図 4 に示す．まず，Cycle 1 では，初段に $gr1 + gr2 \rightarrow gr3$（命令 1）が伝搬される．初段は命令をまだ取り込んでいないため，この命令の取り込みとレジスタの割り当てを行う．入力レジスタに $gr1$ および $gr2$ を割り当て，セレクタを設定する (A)．初段では，入力レジスタはレジスタファイルから読み出されるが，ここでは省略する．次に，Cycle 2 では，命令 1 は第 2 段に進み，初段には新たに $gr3 - gr4 \rightarrow gr5$（命令 2）が伝搬される．第二段は，命令 1 が Cycle 1 で初段に割り当て済みであり，何もしない．一方初段は，命令を取り込み済みであり，命令 2 のための伝搬用レジスタを確保する必要がある．ソースオペランドのうち，$gr3$ は初段の演算結果からバイパスするため，$gr4$ のみを初段の空き入力レジスタへと割り当てる (B)．Cycle 3 では，第 2 段に命令 2 を取り込む．第 2 段 ALU の第 2 オペランド用入力レジスタに $gr4$ を割り当て，セレクタを設定し，伝搬経路を確保する (C)．図ではセレクタの入力は省略しているが，図 3 のように，全ての前段入力レジスタと全ての前々段演算器出力から選択できる．ここでは，Cycle 2 で初段の入力レジスタに割り当てておいた $gr4$ を伝搬する．以上により Cycle 4 に示す段間ネットワークが得られる．

なお，割り当て途中で全レジスタが埋まり，レジスタの伝搬ができない場合や，アレイ段数を超える長さの命令列が見つかった場合には，その命令列はアレイ実行できないため，設定を破棄し，ヒント命令の次命令から通常実行を再開する．

4. 線形アレイ型 VLIW プロセッサの面積評価

表 2 に Synopsys 社の Design Compier 2007.03 による，0.24μ テクノロジに向けた回路規模の見積もり結果を示す．回路規模は構成要素毎に合計し，提案プロセッサの初段に相当する部分を逐次コア，アレイ 1 段に相当する部分を追加コアとして示した，提案プロセッサの初段は，アレイ実行のための L0 キャッシュを備えるため，基準プロセッサの逐次コアよりも面積が大きくなる．アレイ部は，L1 キャッシュやデコーダといったフロントエンドや，レジスタファイルを持たない代わりに，L0 キャッシュと段間ネットワークを備える．評価の結果，基準プロセッサコアの 38%の面積で，1 コア相当の演算器が追加でき，多段のアレイ構成が実現可能であることが分かった．

提案プロセッサの面積効率について考察を行うために，基準プロセッサの 1 コアを逐次コアとし，同一コアを追加コアとして並べたメニコア構成について検討する．画像処理プログラムにおいて妥当であった 32 段アレイと同規模で実現できるのは，最大 13 コアである．

シミュレーションの評価結果より，アレイ実行の逐次実行に対する性能比は 6.8 倍〜17.8 倍であり，提案プロセッサは，一部プログラムにおいて，同等規模で実現しうるメニコアの理論性能である 13 倍を上回った．しかし，プリフェッチのオーバヘッドの占める割合の高いプログラムにおいて，アレイ実行は，逐次実行の 6.8〜8.5 倍の性能にとどまっている．各プログラムはイタレーション毎の依存関係が無いことを前提としており，並列化はそれほど難しくなく，同等性能のメニコア構成に面積効率で劣る可能性がある．ただし，アレイ実行は逐次実行と変わらない外部メモリ帯域でこの性能を実現しているが，メニコアが線形に性能を伸ばすためには，同等の帯域で各コアに十分にデータを供給できる必要がある．バス競合，および，キャッシュラインやメモリバンクの衝突といった要素を含めて，メニコア構成の性能を評価し，提案プロセッサの面積効率を評価することは，今後の課題である．

5. まとめと今後の課題

本稿では線形アレイ型 VLIW プロセッサについて，論理規模の削減手法を提案し，実装が十分に可能であることを示した．評価の結果，基準プロセッサコアの 38%の論理規模で同量の演算器を追加でき，提案プロセッサが面積効率に優れる可能性を示した．今後はさらに実装を進め，より正確な面積評価と面積効率の評価を行う．

また，提案プロセッサは低電力化を考慮したアーキテクチャである．アレイ実行中は，各段が毎サイクル同一命令を実行するため，命令フェッチ，デコード，分岐予測などのフロントエンド部分を完全に停止できる．さらに，アレイ設定が終了しても空いているレジスタや演算器は，少なくとも次のアレイ設定まで用いられることはないため，千サイクル単位のスパンで電力制御が可能である．電力消費をモデル化し，シミュレーションによる消費電力の見積もりを予定している．

加えて，アプリケーションを科学技術計算に広げ，FFT や LINPACK などを提案プロセッサ上に実装し，性能や追加すべき機能について検討を行う予定である．

参 考 文 献

1) Vangal, S. et al.: An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS, ISSCC, pp.98–99 (2007).
2) Bell, S. et al.: Tile64 Processor: A 64-Core SoC with Mesh Interconnect, ISSCC, pp.88–89 (2008).
3) Seiler, L. et al.: Larrabee: a many-core x86 architecture for visual computing, ACM Transactions on Graphics, vol.27, no.3, pp1–15 (2008).
4) 佐藤功人, et al.：ストリーム処理記述言語の GPU 向け自動最適化の検討，先進的計算基盤システムシンポジウム SACSIS2009 論文集，pp.361–368，May. (2009).
5) Becker, J. and Hübner, M.: Run-time reconfigurability and other future trends, SBCCI'06, pp.9–11 (2006).
6) 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 論文集, pp.293–300, May. (2009).
7) Shiota et al.: A 51.2GOPS, 1.0GB/s-DMA Single-Chip Multi-Processor Integrating Quadruple 8-Way VLIW Processor, ISSCC, pp.18–19 (2005).