2008－ARC－180 (5)
2008／10／20

N倍速を目指すVLIWプロセッサの構想

上利宗久[†1] 中田尚[†1] 中島康彦[†1]

クロック速度の改善により性能を向上させることはもはや困難であり，高い演算能力を得るためには並列度を高める必要がある．現在，高解像度の画像処理などには，一般にSIMD演算やメニーコアが用いられている．しかし，これらは専用のプログラミング手法が必要であり，既存プログラム資産が利用できず，性能予測が困難であるという問題がある．また，現行のVLIWプロセッサの拡張ではレジスタファイルのポート数制約から，並列度の向上は困難である．そこで，我々はレジスタファイル間においてに全ての内容をコピー可能な構成を仮定して，N個のVLIWプロセッサを連結したLinear Array Pipeline Processorを提案し，並列度を高めつつ，かつ既存プログラム資産を利用できるプロセッサの実現を目指す．

Concept to make a VLIW processor N times faster

Munehisa AGARI,[†1] Takashi NAKADA[†1]
and Yasuhiko NAKASHIMA[†1]

Processor manufactures can no longer rely on increasing uniprocessor clock speed to provide performance improvement, it is necessary to improve parallelism to achieve performance higher. In general, there are SIMD instructions or a many-core processor for high resolution image processing. However special programming methods are required so it is difficult to use existing software assets or to predict performance. And it is difficult to extend a parallelism of VLIW processor, because there is physical limit for the number of register ports. In this paper, we propose Linear Array Pipeline Processor that includes n VLIW processors connection, assuming a special register file which can be copied to another in a cycle. We aim to improve parallelism in a way that keeps the compatibility of existing software.

1. はじめに

近年，高解像度の画像処理の需要が高まっている．例えば，デジタルシネマでは4096×2160の解像度が規定[1]されている．このような解像度の画像に対して補正するいったフィルタ処理の計算量は膨大であり，既存のプロセッサによるリアルタイム処理は不可能である．

一方，画像処理や数値計算には，一般に高い並列性があることが知られている．これらの処理ではプロセッサの並列度を上げることにより処理能力を高めることが可能であり，現在では，SIMD演算，メニーコア，VLIWプロセッサなどが用いられている．

しかし，SIMD演算やメニーコアでは専用のプログラミング手法が必要であり[2]既存のプログラム資産が利用できない上，QoSが保証されず性能予測が困難であるといった問題がある．

また，VLIWプロセッサは性能予測が容易であり，低消費電力であるものの，レジスタファイルのポート数制約から，現状では1コアあたり，高々8並列度が実現されている[3]に過ぎず，並列度が不十分である．

本論文では全ての内容を1サイクルでコピーできると仮定したレジスタファイルおよびキャッシュを用いて，N個のVLIWプロセッサを連結し，並列動作させる**Linear Array Pipeline Processor**の構想を示す．これにより，既存プログラム資産を利用しつつ，かつ，並列度を非常に高めたプロセッサの実現を目指す．

2. 構成の概要

現在，VLIWプロセッサの並列度の上限を決定しているのは，レジスタファイルのポート数制約である．この制約を満たしつつ，同時に多数の演算器にデータを供給できるようにするために，我々はレジスタファイル間において，1サイクルで全ての内容をコピー可能な構成を仮定する．具体的には同一ビット位置間に

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

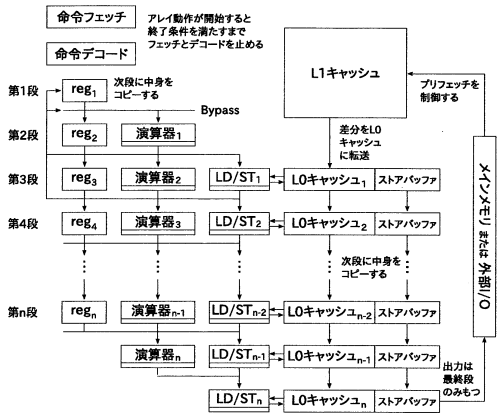

図 1 Linear Array Pipeline Processor の概要

おいてのみコピー可能な仕組み，例えば三次元実装により実現できるとする．その絵でで，既存の VLIW 命令を実行できる N 個の VLIW プロセッサを直列に並べ，それぞれのレジスタファイル，演算器，キャッシュを連結する Linear Array Pipeline Processor を提案する．**図 1** に構成の概要を示す．

提案プロセッサは，通常動作および，アレイ動作という 2 つの動作状態を有する．既存プログラム資産を利用できるように，通常動作時には，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．そのために，初段のレジスタは，初段の演算器および LD/ST ユニットからのフィードバックを備える．

一方，アレイ動作時には N 段全体に VLIW 命令がマップされ，終了条件を満たすまで同じ命令列を繰り返し実行する．以上の構成により，単体の演算器に比べ，最大 N 倍の並列処理を目指す．

2.1 命令マッピング

アレイ動作開始命令を検出すると，命令列を N 段へとマップする．命令マッピングは個々の命令に対し，デコードを行い，各段間のネットワークを形成する処理である．各段へのマップが終了すれば，各段の実行する命令はアレイ動作終了まで固定されるため，アレイ動作中は命令のフェッチとデコードを停止でき，電力消費を抑えることができる．

アレイ動作には分岐という概念が存在しない．そこで分岐命令の代わりに，sel，min，max といった命令によるデータ選択を行う．本プロセッサがストールするのはキャッシュへのデータ供給が間に合わない場合のみであり，適切なキャッシュの制御によって性能を保証できる．

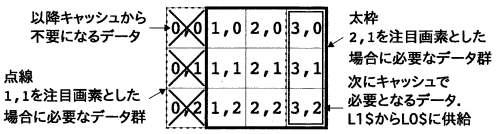

図 2 フィルタ処理時のキャッシュ制御

2.2 キャッシュの制御

アレイ動作開始命令は，キャッシュ制御のヒント命令である．プロセッサはマップ処理が終了するとキャッシュへのデータ供給を開始し，アレイ動作終了条件を満たすまで動作する．

各段は固有のレジスタファイルおよび，L0 キャッシュと呼ぶレジスタファイル程度のキャッシュを有し，メモリからプロセッサへの入力は L1 キャッシュを介して再前段の L0 キャッシュのみが，メモリへの出力は最終段のストアバッファのみが有する．各段は自身のレジスタと前段の演算結果のバイパス，前々段の L0 キャッシュからのロードを元に演算を行う．各段のロードユニットが値を読み出すためには，自身の L0 キャッシュにデータが到着していなければならない．

一般に画像処理においては，対象となる画素群を少しずつずらしながら 1 行ずつ演算を行うような処理が多く，次の処理に必要となるデータは容易に予測可能である．そのため，適切にプリフェッチを行うことにより，読み出しや書き込みに必要な帯域は狭くてよい．本研究では，最適なキャッシュ制御のためのメモリアクセスパターンをあらかじめ指定し，アレイ動作開始命令にヒントとして埋め込むこととする．

プロセッサは，ヒントを元に L1 キャッシュへのプリフェッチを行い，L1 キャッシュから L0 キャッシュへアレイ動作に必要なデータをコピーする．さらに，L0 キャッシュ間では値を後段へコピーすることによりパイプライン動作を行う．

メモリアクセスの例として，**図 2** に示すような注目画素の周辺を含む 9 画素から画素値を決定するフィルタ処理を考える．注目画素を $(1,1)$ から $(2,1)$ に進めた場合，次のイタレーションに必要なデータの差分は 3 画素分であり，これだけを L1 キャッシュから L0 キャッシュに供給できればよく，L0 キャッシュを次段に伝搬する際にも，最低限差分が伝搬できればよい．また，最終的に書き出されるのは 1 画素のデータである．このように比較的単純なメモリアクセスパターンの処理を対象に，既存のプログラム資産の高速化を目指す．

```
for(j=0; j<WD-1; j++) {
  int r = AD(X[a], X[b])
        + AD(X[c], X[d])
        + AD(X[e], X[f])
        + AD(X[g], X[h]);
  a++; b++; c++; d++;
  e++; f++; g++; h++;
  E[i][j] = (r < C) ? 0 : 255;
}
```

図 3 エッジ検出の C コード

命令1	-	-	EAG a	EAG b	LD a>1	LD b>2
命令2	-	-	EAG c	EAG d	LD c>3	LD c>4
命令3	AD12x	-	EAG e	EAG f	LD e>5	LD g>6
命令4	AD34y	-	EAG g	EAG h	LD g>7	LD h>8
命令5	AD56z	ADDxyp	-	-	-	-
命令6	AD78w	ADDzpq	-	-	-	-
命令7	-	ADDwqr	-	-	-	-
命令8	-	-	CMPrC>cc	-	-	-
命令9	-	-	SELcc>s	EAG j	ST s>j	-

図 4 エッジ検出のスケジューリング

3. アレイ動作の例

エッジ検出を例として，アレイ動作について説明する．エッジ検出のアルゴリズムには，**図 3** に示す「注目画素の周囲 3×3 の対角成分の絶対差の合計がしきい値以上の時，その点をエッジとする」処理を仮定する．具体的には，注目画素の座標を (i, j)，入力画素列を X，出力画素列を E とする．アレイ動作に合わせてスケジューリングした結果を**図 4** に示す．AD(Absolute Difference) は，絶対差を求める命令であり，AD12x はレジスタ 1 とレジスタ 2 の内容の絶対差をレジスタ x に書き込む．EAG は，アドレス計算を行う命令である．

アレイ動作では**図 5** のように，図 4 の 9 命令を，命令 N が N 段に対応するようにマップして処理を行う．各段間には従来の VLIW プロセッサにおけるバイパスと同様のネットワークが配置され，自身のレジスタや前段の演算結果，前段の LD/ST ユニットからデータを受け取る．例えば，第 5 段の ADDxyp 命令は，レジスタ x とレジスタ y の内容を加算し，レジスタ p に書き込む．レジスタ x は自身のレジスタファイルから読み出すが，レジスタ y は前段から AD34y の結果をバイパスして利用し，結果 p は後段の ADDzpq へバイパスしている．

キャッシュの流れに着目すると，L0 キャッシュには初段において L1 キャッシュから新しいデータが供給され，徐々に後段へ伝搬する．例えば，第 5 段の L0 キャッシュには，第 4 段の 1 世代前のキャッシュ内容が保存されている．途中段でのストアはストアバッファに蓄積され，最終段でメモリへ書き込まれる．例えば，

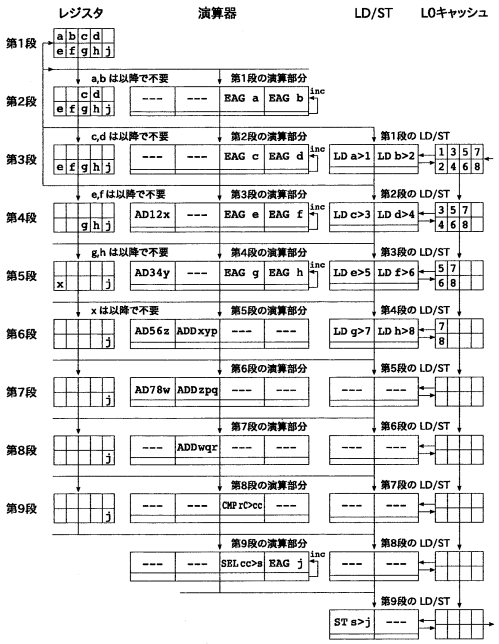

図 5 9 段のアレイ動作例（エッジ検出）

初段において注目画素 $(9, 1)$ の命令 1 が実行される時は，初段からみると 9 世代前の L0 キャッシュを元に最終段において注目画素 $(1, 1)$ の命令 9 が実行され，その処理結果がメモリへと書き込まれる．

このように，9 つのイタレーションが並列実行されるため，処理対象が十分に大きければ，単一のプロセッサにおける逐次実行に比べ，約 1/9 にサイクル数を削減できる．

以上のように，アレイ動作では，既存の VLIW 命令列を並列実行でき，動的なスケジューリングを用いないため，SIMD 演算やメニーコアとは異なり，処理性能を保証することができる．

4. 実現に際しての課題

本提案が仮定したレジスタファイルの実現のために，読み書きに必要なポート数を減らす工夫と，本プロセッサが構造上持つ並列度の制約について考える．

4.1 全コピー可能なレジスタについて

レジスタファイルや L0 キャッシュの全ての内容を 1 サイクルで後段にコピー可能であると仮定した．このようなレジスタファイルやキャッシュの完全な実現は，三次元実装を仮定すれば実現可能であると考えている．しかし，より実現性を高めるために，キャッシュ，レジスタの双方について読み書きを削減する工夫につい

て考える．

まず，キャッシュについては，ヒント情報を元に差分のみをコピーする構成を検討している．

一方，レジスタファイルについては，レジスタに書き込む必要のない一時的な値が多数存在すると考えられる．そのため，マップ時に書き込む必要のあるレジスタを特定し，必要最低限のレジスタのみ読み書きする仕組みにすれば，実現が可能ではないかと予想している．

これらの手法について，図 5 のエッジ処理の例を用いて説明を行う．例えば，第 3 段の AD12x の結果は，第 5 段の ADDxyp で必要となるため，一時的に第 5 段のレジスタ x に書き込まれる．一方，第 4 段の AD34y の結果は本来，第 6 段のレジスタ y に書き込まれるべきであるが，第 5 段以降では参照されないため，第 5 段へとバイパスできればよく，実際に第 6 段のレジスタ y に書き込む必要はない．図 5 では，このような必要最低限の読み書きのみを示している．

また，アドレス値はアドレス計算まで伝搬し，計算結果をレジスタに書き込む必要がある．しかし，EAG 命令に用いる演算器に自己ループを構成し，マップ時に EAG 命令を，アドレスをサイクルごとにインクリメントする命令へ変形することを考える．この構成により，アドレス計算結果の書き込みのみならず，途中段におけるレジスタの伝搬の抑制が期待できる．例えば，図 5 で，レジスタ j は第 9 段の EAG j 以外に使用されず，伝搬を削減できる．同様に，アドレス計算に用いるレジスタの伝搬は全て削減でき，最終的に読み書きが必要なのは，第 5 段のレジスタ x のみとなる．

このようにレジスタファイルの読み書きを抑えることで，物理的なレジスタポート数を増やすことなく，アレイ動作の実現を目指す．

4.2 段数による並列度の制約

提案したプロセッサが並列動作可能な命令数は，段数によって決定される．実装された段数長を超える命令列のアレイ動作はできない．そこで，実際のアプリケーションとして複数の画像処理フィルタを実装し，妥当な段数について検討する必要がある．

また，本プロセッサを直列に接続し，命令列を分割することで見かけ上の段数を増やすことや，複数の処理の並列処理が可能である．一般に，画像処理フィルタなどでは，入出力は毎サイクル 1 画素分のみ供給できればよく，直列接続が可能であると予測している．

5. ま と め

本論文では，高い並列度を実現するために，レジスタファイル間において全ての内容をコピー可能な構成を仮定して，既存の VLIW 命令列をアレイ動作により，最大 N 倍速に並列動作ができるプロセッサの構想を示した．ただし，性能を引き出すためには最適なキャッシュの制御が必要である．

さらに，仮定したレジスタファイルの実現性を高めるために，必要最低限のレジスタのみを読み書きする手法の検討を行った．

今後は，提案したプロセッサのシミュレータを作成し，性能評価を行う．性能評価には画像処理フィルタを用い，複数のフィルタについて必要となるレジスタへの書き込みポート数や命令数，メモリアクセスパターンなどを調査し，妥当な段数やキャッシュ制御方法について検討する．

6. 謝　　辞

本研究の一部は科学研究費補助金（基盤研究 (B) 課題番号 19300012）による．

参 考 文 献

1) DCI Specification, Version 1.2. Errata June 10 2008, http://www.dcimovies.com/DCIDigitalCinemaSystemSpecv1_2.pdf
2) 岡崎 信一郎，京 昭倫，櫻井 和之，古賀 拓也: 画像認識処理指向 LSI の動向とメモリ集積型高並列プロセッサ IMAP，信学技報 PRMU2004-32, pp.29-36 (2004)
3) 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)