2007－ARC－174 (10)
2007／8／1

パイプライン接続型 MIAD アーキテクチャによる映像処理技術

細木 浩二† 江濱 真和† 中田 啓明†
柴山 哲也‡ 望月 誠二‡ 岩田 憲一†

PIPE(Programmable Image Processing Element)は，3つのCPUがパイプラインで接続されたマルチプロセッサ構成のメディアプロセッサである．各CPUの命令は，複数のソース/デスティネーションレジスタを複数の2次元配列データとして指定可能なオペランドを有することを特徴とするSIAD(Single Instruction Arrayed Data)型命令である．SIAD型CPUは，3×3コンボリューションフィルタにおいて，乗算器稼働率100%を実現する．複数のSIAD型CPUをパイプライン接続した，“パイプライン接続型MIAD(Multi Instruction Arrayed Data)”アーキテクチャを用いたPIPEは，H.264映像コーデック応用にて，4way-VLIW型プロセッサと比較し，コードサイズを75%削減し，高い命令圧縮効果を得ると共に，低電力化を実現する．

Image Processing Technique Using Pipeline Connection MIAD Architecture

Koji Hosogi† Masakazu Ehama† Hiroaki Nakata†
Tetsuya Shibayama‡ Seiji Mochizuki‡ Kenichi Iwata‡

PIPE, Programmable Image Processing Element, is a media processor which consists of three CPUs with a “Pipeline Connection Architecture”. Each instruction within these CPUs can specify arrayed data as operands and handle multiple source data as multiple vectors. This “Single Instruction Arrayed Data (SIAD)” CPU achieves 100% operation rates on 3x3 convolution filter. In addition, the PIPE using a “Multi Instruction Arrayed Data(MIAD)” architecture, which consists of three SIAD-CPUs, achieves 75% code size reduction compared with a 4Way-VLIW on H.264 function.

1 はじめに

近年の半導体微細化に伴い，高解像度映像処理などの大規模な機能を1つのLSI上に実装可能となっている．映像処理は多岐に亘り，例えば，MPEG-2[1]やH.264[2]などの映像コーデック処理，画像拡大や縮小，エッジ抽出などのフィルタリング処理，また3次元計測の1つであるステレオマッチング[3]などの映像特徴抽出処理などが挙げられる．

映像処理の中で大部分の占める処理は，小容量2次元ブロックのフィルタリング演算である．また，映像コーデックの特徴は，動き補償処理，整数変換処理，画像フィルタなどの2次元ブロックに対する処理が，パイプライン的に多段接続されるデータフローである．このような多段的な2次元ブロックに対する処理をハードワイヤ論理で実現する場合，処理間で依存関係のない粒度の処理に分割し，処理を並列処理することで必要動作周波数を抑え，低消費電力化を実現している．

† 日立製作所 組込みシステム基盤研究所
Embedded System Platform Research Laboratory, Hitachi, Ltd.
‡ ルネサステクノロジ システムソリューション統括本部
System Solution Business Group, Renesas Technology Corp.

一方，DSPを用いた映像処理のソフトウェアによるアプローチも現れている[4]．ソフトウェア・アプローチは，プログラマビリティによる高い汎用性を持つ一方，並列化が困難なため，高い動作周波数が必要であり，消費電力が大きい．

我々は，ソフトウェアによるプログラマビリティと，ハードワイヤ論理の低消費電力という2つの利点を実現するパイプライン接続型MIAD(Multi Instruction Arrayed Data)アーキテクチャを開発した．本アーキテクチャは，2つの概念を有する．1つ目の概念は，SIAD(Single Instruction Arrayed Data)と定義するCPU構成で，複数のデータで構成する2次元データに対する演算を，1つの命令にパッキングすることを特徴とする．2つ目の概念は，複数のSIAD型CPUの階層的に多段接続したマルチプロセッサを実現するパイプライン接続アーキテクチャである．これら2つの概念を有するPIPE(Programmable Image Processing Element)を開発し，リアルタイム映像処理アプリケーションに適用した．

以降2章では，パイプライン接続MIADアーキテクチャに関する説明，3章は，本アーキテクチャをH.264映像コーデック・アプリケーションに適用した評価結果，及び本アーキテクチャとSIMD/VLIW型アーキテクチャにて比較を示し，4章で全体を纏める．

2 アーキテクチャ

2.1 Single Instruction Arrayed Data

2.1.1 SIAD 命令フォーマット

単一の命令で複数のデータ演算を行うSIMD (Single Instruction Multi Data)命令[5]は，演算並列度の向上により，高い演算スループットを得ることができる．一般的に，SIMD命令は横方向若しくは縦方向の単一方向のデータに対し，並列演算を行う構成である．しかしながら，コンボリューションフィルタなどの映像処理応用では、データ配列方向とは異なる2次元矩形演算が必要である．これをSIMDに投入する場合，フィルタ演算の前後にて，以下に示す前後処理が必要となる．

- 座標変換のためのデータ転置処理
- フィルタ演算の中間値の演算精度を維持するためのビット拡張処理
- 最終演算結果のデータ精度を得る丸め込み処理

この前後処理による性能低下の要因を，2 つ挙げる．

- 1 命令で単一ペアのソースデータのみ供給し，1 つの演算結果を得る
- RISC は，データの位置移動演算(シフト，転置，ビット拡張，丸め込みなど)に 1 命令を必要とする

我々は，これら性能低下要因を解決するCPU方式として，SIAD(Single Instruction Arrayed Data)アーキテクチャを提案する．まず，図 1 にSIAD命令フォーマットを示す．

SIAD命令フォーマットは，一般的なRISC [6]の命令フォーマットに加え、幅オペランド(Width)と高さオペランド(Count)を有し，複数のソースデータやデスティネーションを配列データとして指定可能である．本方式では，オペランドで指定するソース／デスティネーション番号をポインタの起点とし，複数サイクルかけてポインタ制御することで，1 つの命令で複数のソースデータを供給し，複数のデスティネーションを得ることが出来る．このように，1 命令で複数サイクルの時間軸情報を制御可能とすることにより，水平方向，垂直方向を問わず，2 次元のデータ配列をソースとデスティネーションに指定する．

データメモリに対するアクセスを制御する LD/ST 命令は，Width と Count に加え，折り返し Pitch を指定するオペランドを有し，データメモリ上に配置された 2 次元配列領域の一部に対してアクセス可能である．

SIMD による 2 次元配列演算は，ソースやデスティネーションのみが異なる、同一命令オペコードの命令列を逐次実行する．一方 SIAD は，ソースとデスティネーションの更新を，高さオペランドで指定する方式であるため，命令列方向で，命令の圧縮を行うことが出来る．

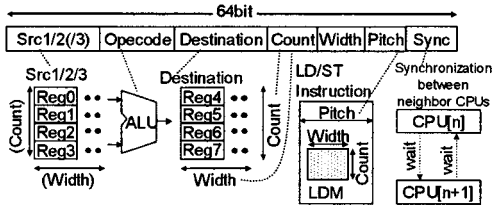

図 1 SIAD 命令フォーマット

なお，SIAD 命令フォーマットには，マルチプロセッサ間の同期を制御する同期化オペランド(Sync)を有する．本機能に関しては後述する．

2.1.2 SIAD 型 ALU

図 2 にSIAD型ALUの基本構成を示す．各データパスは，SIMD型ALUのように，演算器を並列に配置し，かつ，演算器がパイプライン的に接続された構造である．本説明では，4 つのデータパスをパイプライン接続した構成について説明する．

初段データパスはマッピング部で，レジスタファイルから出力されるソースデータに対し，シフト処理やバイト拡張処理を行う．レジスタファイルの参照ポインタ制御と，マッピング部の制御はカウンタ部(Counter)で実行することで，データの位置変更を伴う 2 次元配列データを、2 段目以降の演算器に投入する．

2 段目から 4 段目のデータパスは，SIMD 構成のように、演算器を並列配置した構造である．2 段目は乗算器，3 段目は乗算結果を複数サイクル間加算するシグマ加算器，4 段目は，シグマ加算器の演算結果の丸め込みを行うバレルシフタである．

これら複数の演算器をパイプライン接続し，レジスタファイルの参照ポインタとマッピング部の制御をハードウェアで実行することにより，1 つの命令で，2 次元配列データを供給し，パイプラインで 4 種の異なる演算を実行する．特にシフタや丸め込みなどのデータの位置移動に関する命令を 1 命令中に隠蔽することで，命令圧縮率を向上する．

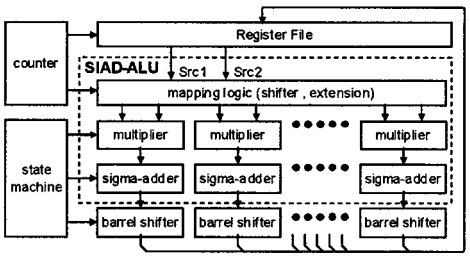

図 2 SIAD 型 ALU 基本構成

2.1.3 SIAD 動作原理

2 次元コンボリューションフィルタは，多くの映像処理に用いられる汎用性の高いデジタルフィルタである．

図 3 は，2 次元コンボリューションフィルタ命令における、SIAD型ALUへのソースデータ供給シーケンスを示すもので，SIADの動作原理について説明する．代表的なコンボリューションフィルタは，3×3 のフィルタマスクを用いるが，本稿では 2×2 のフィルタマスクを用いて説明する．

演算結果f(Destination:Reg8-Reg11)は，2 次元配列a(Src1:Reg0-Reg4)に対し，フィルタマスクb(Src2:Reg5)で畳み込み演算を行った結果である．各データ要素は，小文字[r]で表現する．

例として，Destination の r18 の算術式を式(1)に示す．

式(1)が示す通り，演算結果 r18 の各データ要素は，4 つの乗算結果を加算した値である．一般的には，本演算

結果に対し，ビット精度調整を行うため，ビットシフトによる丸め込みを行う．

$$r18 = r01 \times r05 + r20 \times r15 + r11 \times r25 + r21 \times r35 \quad \cdots\cdots(1)$$

Src1 と Src2 の供給は，マッピング部により逐次制御され，乗算器に投入する．Src1 は，フィルタマスクの列数に達するまで，サイクル毎にデータ要素分左シフトしてデータ供給し，列数に達した時点で，次の行データを供給する．Src2 はフィルタマスクを複製して各演算器に供給し，サイクル毎にフィルタマスク値をシフトしてデータ供給する．

図中のハッチングで示した Src1 の{r01, r20, r11, r21}と Src2 の{r05, r15, r25, r35}を，サイクル毎に乗算器に投入し，逐次，演算結果をシグマ加算器で加算することにより，SIMD の様に、4サイクル後，デスティネーション r18 を得る．本演算を 4 並列で演算することにより，1 行分の演算結果{r08, r18, r28, r38}を得ることが出来る．

次に，Src1 の読出し行をインクリメントし，同様にソースデータを供給することで，4 行分の演算結果 f を得る．この時，総演算サイクルは 16 サイクルで，乗算器の稼働率 100%を実現する．

2 次元コンボリューションフィルタは，前処理として非常に複雑なシフト処理が必要な 2 次元配列演算である．SIAD 型 ALU は，マッピング部により、この複数サイクルに亘る複数ソースデータの供給を制御することにより，前処理による処理サイクル数を削減可能である．

$$f(x,y) = \sum_{i,j} a(x+i,y+j) \times b(i,j) \ ; \ \text{here, } (i, j=0 \sim 1)$$

$$a = \begin{bmatrix} Reg0 \\ Reg1 \\ Reg2 \\ Reg3 \\ Reg4 \end{bmatrix} = \begin{bmatrix} r00, & r10, & r20, & R30, & r40 \\ r01, & r11, & r21, & R31, & r41 \\ r02, & r12, & r22, & r32, & r42 \\ r03, & r13, & r23, & R33, & r43 \\ r04, & r14, & r24, & R34, & r44 \end{bmatrix} = \text{Src1}$$

$$b = \begin{bmatrix} Reg5[0] & Reg5[1] \\ Reg5[2] & Reg5[3] \end{bmatrix} = \begin{bmatrix} r05, & r15 \\ r25, & r35 \end{bmatrix} = \text{Src2}$$

$$f = \begin{bmatrix} Reg8 \\ Reg9 \\ Reg10 \\ Reg11 \end{bmatrix} = \begin{bmatrix} r08, & r18, & r28, & r38 \\ r09, & r19, & r29, & r39 \\ r0a, & r1a, & r2a, & r3a \\ r0b, & r1b, & r2b, & r3b \end{bmatrix} = \text{Destination}$$

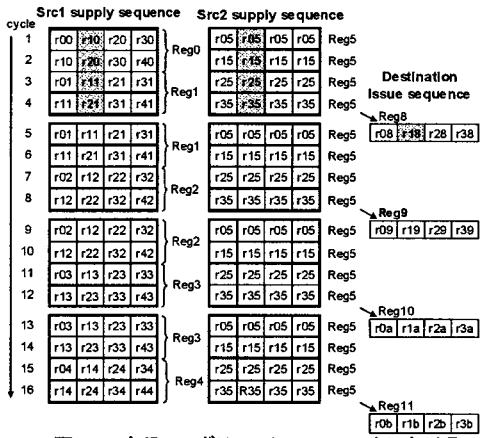

図 3 2 次元コンボリューションフィルタにおけるソースデータ供給シーケンス

同様な方式で、マッピング部によりソースデータの供給を制御することで、行列の内積演算などの演算を、同一の SIAD 型 ALU 構成で実現する。

式(2)は，SIAD 型 ALU における N タップフィルタの演算スループット示す．例えば，3×3 タップのコンボリューションフィルタは 9 タップのフィルタ演算である。8way-SIAD 型 ALU において，，3×3 タップのコンボリューションフィルタは、8/9 の演算スループットを得る．

$$\text{throughput[elements/cycle]} = \frac{\text{[number of SIAD ways]}}{\text{[number of filter taps]}} \quad \cdots\cdots(2)$$

2.2 パイプライン接続アーキテクチャ

2.2.1 Multi Instruction Arrayed Data

パイプラインは、汎用プロセッサやハードワイヤ論理の演算スループットを向上する基本技術である．特に，フィードバックなどのデータ依存性のないデータフローにおいて，大きなスループットを得ることが出来る．

図 4 にパイプライン接続アーキテクチャで構成した PIPE(Programmable Image Processing Element)の構成図を示す．PIPEは，主にデータの供給を行う LD-CPU，演算を行うMedia-CPU，データ格納を行う ST-CPUが密に結合されたマルチCPU構成である．これらCPUのALU出力は，自身のレジスタファイルへのライトバックポートへの接続と共に，次段CPUのレジスタファイルに、パイプラインで直列接続(パイプライン接続アーキテクチャ)した構造である．

LD-CPU と ST-CPU はローカルデータメモリに接続され，データアクセスを制御する．Media-CPU は主に 2 次元フィルタ演算などのメディア処理を行う CPU である．

各 CPU は，それぞれ独立したプログラムカウンタと命令デコーダを有し，独立して動作する．各 CPU は SIAD 型 CPU で，これを複数接続した MIAD(Multi Instruction Arrayed Data)構成である．

本パイプライン接続アーキテクチャは，マルチスカラ[7]と類似した構成であるが，2 点の概念が異なる．

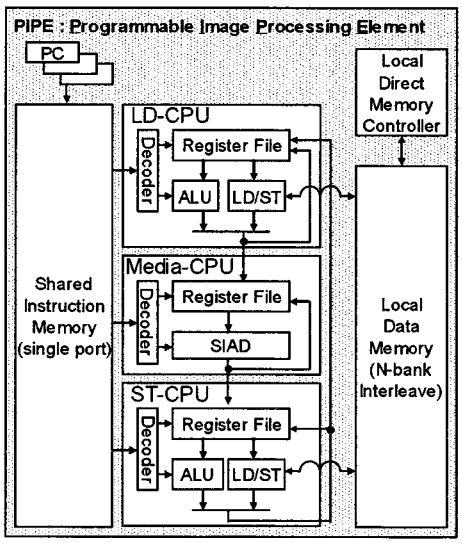

図 4 MIAD 型 PIPE の構成

1つ目は，Media-CPUとデータメモリ間の接続がなく，データアクセス向けの専用CPU(LD-CPUとST-CPU)を有することである．本構成は，マルチスカラと比較し，データメモリの参照ポート数を削減する．

2つ目は，マルチスカラのタスクレベルCPU間同期とは異なり，命令レベルのCPU間同期化を行うことである．各命令は隣接するCPUとの同期を制御するための同期化オペランドフィールドを有し，命令レベルで隣接CPUとの同期化を実現する．

同期は入力同期と出力同期の2種類からなる．入力同期は，前段CPUの演算結果を後段CPUがソースデータとして使用する場合，前段CPUによる演算結果格納が終了するまで，後段CPUは停止する同期である．一方，出力同期は，後段CPUのレジスタファイルに格納領域が確保可能となるまで、前段CPUが演算結果の格納を停止する同期である．

本パイプライン接続により，データメモリからのデータリードとメディア処理演算，及びデータメモリへのデータ格納を，パイプラインで並列処理する．本構成は，スーパスカラ[8]やVLIW型CPU[9]とは異なり，各CPUが独立したプログラムカウンタで独立に動作するため，各CPUは隣接CPUの稼動状況に応じて自動的に停止，再開を行うデータ駆動型[10]の振る舞いをする．

また，パイプライン接続では，スーパスカラやVLIWで使用する共有型レジスタファイルとは異なり，レジスタファイルをFIFOの様に使用した直列接続構造のため，レジスタファイルのアクセスポート数を削減できる．

LD-CPU，Media-CPU，ST-CPUは共有型命令メモリを共有利用し，時分割で命令供給を受ける．共有型命令メモリにおける命令メモリのアクセス競合は，散発的に命令フェッチを行うSIAD型CPUの利用により，頻度を少なくすることが出来る．

2.2.2 マルチPIPE接続

図5は、複数のPIPEを接続したマルチPIPE構成を示した図である。シフトレジスタで構成するシフト型バスを介して，複数のPIPEを多段接続することが出来る．またPIPEと外部メモリ間のデータ通信を行うグローバルDMAコントローラ(GDMAC)を接続し、大規模システムを構築できる。

本構成によれば，3つのSIAD型CPUによるMIAD型PIPEを、パイプラインで複数接続した構成である．SIADは，ループ処理などの命令レベルのパイプラインを実現し，MIADはデータ読み込みとメディア処理およびデータ書込みレベルや最粒度のコードレベルのパイプラインを実現する．マルチPIPE接続では，プロセスレベルのパイプラインや並列処理を実現する．この様に，マルチPIPE構成は，複数階層におけるパイプライン処理を実現可能な構成である．

シフト型バスは，PIPE毎にフリップフロップで区切られ，隣接するPIPEには1サイクルのレイテンシでデータ通信を行う．n個離れたPIPEに対しては，nサイクルのレイテンシでデータ転送を行う．シフトレジスタにより分割されたバス構成は，同時に、複数のPIPE間のデータ通信を可能とする．m個のPIPEで構成したマルチPIPE構成におけるバスバンド幅は，最大で，m×バス幅である．

また，PIPE間にフリップフロップを配置するバス構成のため，隣り合うPIPEに対し，タイミングクリティカルになりにくい短い配線長で配線できる．従って，PIPE数を拡張する場合，グローバル配線の多いクロスバスイッチ型バスと比較し，容易にPIPEの追加接続が可能で，スケーラビリティを確保する．

更に，右から左に接続する右回りシフト型バスと，左から右に接続する左回りシフト型バスの2系統のバスを有する．右回りシフト型バスでは，主にPIPE間通信，およびGDMAC経由による外部メモリへのデータ書出しを行う．左回りシフト型バスは，主に，GDMACを経由した外部メモリからのデータ読出しを行う．

この2系統バス構成により，PIPE間通信と外部メモリからの読出し転送は競合せず，外部メモリ読出しによる性能低下を低減できる．映像処理では外部メモリに格納された大容量の画像に対しプリフェッチを行うことが多い．大容量データのプリフェッチは，アクセスレイテンシの低減よりも，高いデータ転送スループットが必要である[11]．シフト型バスは，PIPE毎に転送遅延が増加する課題が存在するが，高いスループットの確保が可能であり，映像処理ではアクセスレイテンシの増加による性能劣化は少ない．

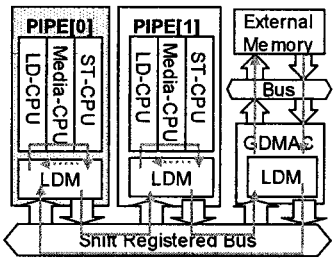

図5 マルチPIPE構成

3 評価

3.1 評価アプリケーション

SIADは，1命令で複数のソースデータを供給し，シフト演算などの前処理と，フィルタ演算などの映像処理，シグマ加算やバレルシフタなどの後処理を行い，複数のデスティネーションを得ることが可能な，高抽象度の命令構造である．この高い命令抽象度は，コードサイズ縮小に伴う命令メモリ容量削減と命令フェッチ数の削減による低電力化の効果を得る．

また，MIAD構成における共有型命令メモリの使用により，命令メモリ数若しくは命令メモリのアクセスポート数を削減できる．

表1 評価に使用したH.264映像コーデック処理内容

コーデック分類	項番	処理内容
デコード	(i)	デブロッキングフィルタ
	(ii)	動き補償
	(iii)	逆量子化/逆整数変換，イントラ予測
エンコード	(iv)	デブロッキングフィルタ
	(v)	少数画素精度動き検索，動き補償
	(vi)	整数変換，量子化，逆量子化，逆整数変換，イントラ予測

本節では，SIADおよび共有型命令メモリを利用したMIADの評価を行う．評価プログラムは，本アーキテクチャに最適な2次元配列演算を利用するH.264映像コーデックである．
表1に，評価に使用したH.264映像コーデックのプログラム内容を示す．これらの処理は，映像処理において特に複雑な機能を含む処理であるため，SIADとMIADの実アプリケーションへの適応性について評価できる．

3.2 評価モデル

評価には表2に示す評価モデルのPIPEを使用する．評価対象(ii)(iii)(v)(vi)では，2つのMedia-CPUを使用し，LD-CPUとST-CPUと合わせて計4個のCPUを使用する．命令長は全て64ビットである．データメモリは，バンクインタリーブ構成で，他PIPEからのデータ通信の影響は，比較的少ないモデルである．

表2 評価モデル

項 目	(i), (iv)	(ii), (v)	(iii), (vi)
Media-CPU 数[個]	1	2	2
総 CPU 数[個]	3	4	4
SIAD-Way[way]	8	10	8
乗算器精度[bit×bit]	8×8	16×8	16×8
命令長[bit]	64	64	64
データメモリ幅[bit]	64	64	64
データメモリバンク数[個]	4	3	3

3.3 静的評価：命令圧縮

本節ではコードサイズについて評価する．コードサイズは，必要命令メモリ容量を示す指標で，容量削減はLSI回路規模の削減につながる．図6に，PIPEと4way-VLIWとのコードサイズ比較結果を示す．
4way-VLIW[12]は，主にデータメモリをアクセスする2つのALUと，主にメディア処理を行う2つのSIMD型ALUで構成するVLIWである．VLIWの命令長は4wayにおいて136ビットで，NOP命令を圧縮可能[13]，かつプレディケーションによる分岐命令削減を可能とする構成である．
評価プログラムは，H.264映像デコード処理における(iii)逆量子化/逆整数変換/イントラ予測である[14]．
結果，4way-VLIWのコードサイズ11kバイトに対し，PIPEは2.7kバイトであった．これは，約75%の命令圧縮を実現したことを示す．

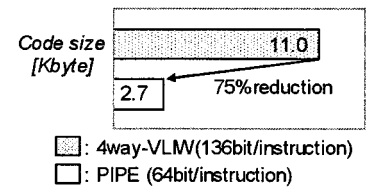

図6 PIPEと4way-VLIWとのコードサイズ比較 [処理-(iii)：逆量子化，逆整数変換，イントラ予測]

プロセッサにおける演算スループットの向上は，ALUの利用率を向上することである．RISCにおけるALU利用率は，キャッシュミスを考慮しない場合，命令数と等しく，命令当りのALU利用率は1である．言い換えれば，1命令で，ALUに1つ，若しくは1ペアのソースデータを投入することを意味する．よって，ALU利用率を別の定義で表現すると，ALUに対しソースデータを供給した回数である．
SIADでは，ALU利用率と命令数が等しくないため，上記の定義により，ALU利用率を定義する．2.1.3節で示した2×2行列のコンボリューションフィルタでは，1命令で16サイクル間ソースデータを供給するため，キャッシュミスを考慮しない場合，ALU利用率は1である．一方，1命令で16サイクル分のソースデータを供給する観点より，SIADの1命令は，RISCの16命令に相当する命令の情報量を有する．これは，命令長を無視した場合、命令を16倍圧縮したことを示す．
式(3)に命令圧縮の度合いを示す命令圧縮度(ICR)を定義する．ICRは，1命令当りのソースデータ供給数である．本値が1よりも大きい場合，命令は圧縮されていることを意味する．RISCのICRは1である．一方，SIADにおける2×2行列のコンボリューションフィルタ命令のICRは16である．

$$ICR = \frac{[\text{number of source data processing}]}{[\text{number of instructions}]} \quad \ldots\ldots(3)$$

図7にH.264映像コーデックにおける各処理のICRを示す．評価プログラム(i)～(vi)では，全プログラムにおいてICR値は2.9以上である．特にH.264映像デコードにおける動き補償処理(ii)では，ICR値6.76であり，高い命令圧縮を実現している．動き補償処理は，その大部分が2次元ブロック演算であるため，特にSIADによる命令圧縮の効果が大きい．

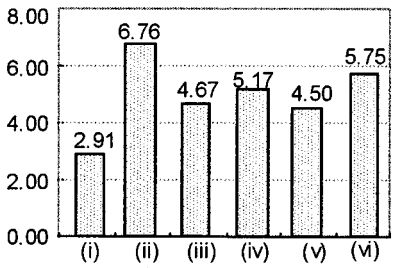

図7 H.264コーデックにおける命令圧縮度ICR

3.4 動的評価：命令フェッチ数の削減

前節の評価では，命令圧縮によるコードサイズの削減に着目した．本節では，評価プログラムの実動作による性能と電力の評価を行う．
表3は，評価プログラムの実処理において，サイクル当りの命令メモリフェッチ回数を示したものである．本値が1より小さい場合，命令メモリの読出しを停止することが可能で，命令メモリフェッチで消費する電力を削減できる．表が示す通り，各評価プログラムにおいて，サイクル当りの命令メモリフェッチ数は、0.28～0.48の範囲にある．本値は，LD-CPUと1つ若しくは2つのMedia-CPU，及びST-CPUで構成するMIADにおける値であり，1CPU当りでは，0.1程度の値となる．
従って，MIAD構成においても，サイクル当りの命令数を1以下に削減可能であることを示し，共有型の命令メモ

リを使用できる事を示す.

LSI 内で使用するメモリは，アレイ状に配置された記憶素子と，アドレスデコーダ部およびアンプ部で構成する．プロセスの微細化と共に，メモリのアドレスデコーダ部とアンプ部の面積比重が増加している．よって，同一メモリ容量の場合，メモリ数を削減することで，メモリの面積を削減出来る．

これらの評価結果をまとめると，MIAD 構成は，命令メモリのアクセス数削減による消費電力の削減と，メモリ数の削減による面積削減の 2 つの効果を得る．

表 3 サイクル当りの命令メモリフェッチ数

処理内容	命令メモリフェッチ回数 [命令]	処理サイクル [サイクル]	サイクルあたりの命令メモリフェッチ数 [命令/サイクル]	
			全 CPU	1CPU 当り
(i)	296	976	0.30	0.10
(ii)	301	1, 003	0.30	0.08
(iii)	383	939	0.41	0.10
(iv)	304	982	0.31	0.10
(v)	348	1, 246	0.28	0.07
(vi)	586	1, 221	0.48	0.12

4 おわりに

4.1 まとめ

本稿では，SIAD アーキテクチャとパイプライン接続アーキテクチャ，及び，これらを纏めたパイプライン接続型 MIAD アーキテクチャについて示した．また，映像処理の 1 つである H.264 映像コーデック応用における、本アーキテクチャの評価結果を示した．

評価の結果，データ移動に関する命令の削減と，繰返し処理に関する命令の削減により，コードサイズの削減と，命令メモリ数の削減を実現できた．また，命令メモリアクセス回数削減による、低消費電力化が可能である見通しを得た．

4.2 今後の課題

デジタル民生機器では，特に低消費電力化とゲート規模の削減が重要課題である．命令供給に関するゲート規模をハードワイヤ論理のシーケンサ程度まで削減し，データパスの最適化を行う事により，ソフト方式においても，低消費電力化とゲート規模削減を実現できると考える．また，命令圧縮による命令供給の高効率化は，メニーコアやタイルプロセッサ，動的再構成プロセッサの電力削減に関して効果があると考える．

現在，本技術を活用し，SD 解像度及び HD 解像度向けの H.264 コーデック LSI を開発中である．今後は，映像コーデックに加え，様々な映像処理応用に適応し，更なる効率向上を目指す．

参考文献

[1] ISO/IEC: Information technology-Generic coding of moving pictures and associated audio information, (1996).

[2] ISO/IEC: ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification.", (2003).

[3] 三浦清志，張山昌論，亀山充隆：再起的演算に基づくステレオマッチング VLSI プロセッサのアーキテクチャ，情報処理学会研究報告「システム LSI 設計技術」(2003-SLDM-111)，Vol.2003 No.105，pp. 117-122 (2003).

[4] 五嶋健治，青木哲史，幡生敦史，松原岳次，山田慎一，宮崎孝：低消費電力 DSP を用いた H.264 ビデオデコーダの開発，情報処理学会研究報告「システム LSI 設計技術」(2004-SLDM-116)，Vol.2004 No.102，pp. 55-59 (2004).

[5] 京昭倫，岡崎信一郎，黒田一朗：画像フィルタ処理の高速化に向けたメディア拡張プロセッサ用 SIMD コンパイラ，情報処理学会研究報告「計算機アーキテクチャ」(2003-ARC-154)，No.154-15，pp. 85-90，(2003).

[6] J.L. Hennessy，D.A. Patterson：Computer Architecture: A Quantitative Approach，Morgan Kaufmann Publishers，Inc.，(1990).

[7] T.N. Vijaykumar and G.S. Sohi: Task Selection for a Multi-scalar Processor，31st International Symposium on Micro-architecture (MICRO-31)，(1998).

[8] M. Johnson: Superscalar Microprocessor Design，PTR Prentice Hall，(1991).

[9] R.P. Colwell，R.P. Nix，J.S. O'Donnell，D.B. Papworth and P.K. Rodman: A VLIW architecture for a trace scheduling compiler，IEEE Transactions. Computers.，Vol.C-37，No.8，pp.967-979，(1988).

[10] 坂井修一，平木敬，山口喜教，児玉祐悦，弓場敏嗣：データ駆動計算機のアーキテクチャ最適化に関する考察，情報処理学会論文誌，Vol.30，No.12，pp. 1562-1572，(1989).

[11] 細木浩二，東嶋重樹，田代卓，川口敦生，西岡清和：メディアプロセッサ MAPCA のデータ転送方式，情報処理学会研究報告「計算機アーキテクチャ」(2002-ARC-146)，Vol.2002，No.9，pp. 91-95 (2002).

[12] 小島啓二，西岡清和，川口敦生，細木浩二，鷹田憲久：ソフトウェアによる実時間ディジタルメディア処理向けシステムオンチップ，電子情報通信学会論文誌(EIC-D-I)，Vol.J87-D-I，No.12，pp.1051-1058，(2004).

[13] 田中和彦，鍬田真，野尻徹，西岡清和：VLIW プロセッサにおける命令圧縮方式，第 63 回情報処理学会全国大会講演論文集(1)，pp. 37-38，(2001).

[14] 近久真章，東嶋重樹，藤平連，川口敦生：BroadGear を用いた H.264 ビデオデコーダ，電子情報通信学会ソサイエティ大会講演論文集，Vol.2005 年_エレクトロニクス，No2(20050907)，pp.88，(2005).