2005－SLDM－118（22）
2005／1／26

実時間画像エフェクト処理のための ASIP アーキテクチャの提案

吉村　龍洋†　　坂主　圭史†　　武内　良典†　　今井　正治†

† 大阪大学 大学院情報科学研究科 情報システム工学専攻
〒 565-0871 大阪府吹田市山田丘 1-5
E-mail: †{r-yosimr,sakanusi,takeuchi,imai}@ist.osaka-u.ac.jp

あらまし　入力された画像に対して視覚的効果を加える画像エフェクト処理を高速に行うための ASIP アーキテクチャを提案する．提案するアーキテクチャでは，画像エフェクト処理に特化した演算器，メモリアクセス回路用いることで，低 H/W コストで高速な画像エフェクト処理を実現する．提案するアーキテクチャに基づいた ASIP を設計し，命令セットシミュレータで実行速度を評価し論理合成を行った結果，低 H/W コストでも実用的な処理速度が得られた．

キーワード　ディジタル画像処理，画像エフェクト処理，ASIP，メモリアクセス回路

ASIP Architecture for Real-Time Graphical Effect Processing

Tatsuhiro YOSHIMURA†, Keishi SAKANUSHI†, Yoshinori TAKEUCHI†, and Masaharu IMAI†

† Graduate School of Information Science and Technology, Osaka University
1-5 Yamada-Oka, Suita-shi, Osaka, 565-0871 Japan
E-mail: †{r-yosimr,sakanusi,takeuchi,imai}@ist.osaka-u.ac.jp

Abstract　Graphical effect processing realizes a variety of visual representation. In this paper, we propose an ASIP architecture for real-time graphical effect processing. The proposed architecutre consists of ASIP（Application Specific Instruction set Processor）and memory access unit specified for graphical effect processing. We implemented an ASIP based on the proposed architecture, and evaluated execution time of graphical effects processing on instruction set simulator（ISS）. Results of logic synthesis and evaluation on ISS show that the proposed architecture can process real-time graphical effects with additional small H/W cost.

Key words　digital image processing, graphical effect, ASIP, memory access circuit

1. はじめに

ディジタル画像処理は，医療用や放送用などの大型機器から，携帯電話やディジタルカメラなどの小型機器に至るまで幅広く用いられている．特に実時間性が求められるようなアプリケーションに対しては，ディジタル画像処理を専用 H/W 化した LSI を機器に組み込むことで，高速なディジタル画像処理を実現している．文献 [1] では，演算の一部または全体を H/W で実装した IP (Intellectual Property) を複数用意し，CPU のみの処理では実時間制約を満たせない場合に，必要な IP をシステムに追加することで高速な画像処理を実現している．

ディジタル画像処理には，圧縮伸長処理，動き検出などの認識処理，画像に対して視覚的効果を与えるエフェクト処理等が存在する．一般に圧縮伸長処理や解析処理では，使用するアルゴリズムが規格や仕様として決められており，いろいろなアルゴリズムを組み合わせて使用されることは少ない．したがって，目的に応じたアルゴリズムが多く提案されており，アルゴリズムを専用 H/W 化するための方法も多く提案されている．しかしながら，画像に視覚的効果を与える画像エフェクト処理は，様々な映像表現を実現する目的で使用されるため，一つの機器の中においても多種類の画像エフェクト処理が実行される．したがって，文献 [1] のように個別の処理に特化した H/W を組み合わせるという手法では，様々な画像エフェクト処理を実現しようとすると，実現する画像エフェクト処理の数に比例して回路規模が増大してしまうため，画像エフェクト処理の実装にあたっては，多種類の画像エフェクト処理で H/W 資源を共有できるような柔軟なアーキテクチャが必要である．

一般に，画像エフェクト処理では，メモリ上に置かれた入力画像を読み込み，視覚的効果を施した出力画像をメモリに書き込む．したがって，画像エフェクト処理を実時間で処理するためには，メモリアクセスにかかるコストを減らしたアーキテクチャが必要である．

メモリアクセスコストを削減する方法として，メモリキャッシュを用いる方法がある．しかし，画像エフェクト処理は，入力画像全体に対して処理を行う必要がある一方で，出力画像の1ピクセルの計算に必要なデータは，入力画像中の1ピクセル，あるいはその周辺ピクセルに限られる．したがって，画像エフェクト処理における入力データの再利用性は低く，キャッシュに読み込まれたデータはすぐに不要になってしまうことが多いため，メモリキャッシュを搭載しただけではメモリアクセスのコストを大幅に削減することはできない．

文献[2]では，演算回路を持つ複数の小容量メモリに処理対象の画像を分割して格納し，それらを並列に動作させることで，メモリアクセスと演算処理を並列に行うアーキテクチャを提案している．しかしこの手法では，小容量メモリ毎に演算回路を持つため回路規模が増大する．また，互いに接続されていないメモリ間でデータの転送を行うような画像処理には向かない．

そこで本稿では，アプリケーションに対して柔軟性を持つASIP（Application Specific Instruction set Processor）を用いることでH/W資源を効率的に共有し，かつ画像エフェクト処理に特化したメモリアクセス回路を用いることでメモリアクセスのコストを低減した，画像エフェクト処理のためのアーキテクチャを提案する．提案アーキテクチャは，画像エフェクト処理に特化した命令を組み込んだプロセッサと，専用のメモリアクセス回路を用いることで，画像エフェクト処理に必要な演算処理を高速にプロセッサで実行し，メモリアクセスを効率化することで，画像エフェクト処理を実時間で処理する．

以下，本稿の構成について述べる．第2節では，画像エフェクト処理の特長をまとめ，画像エフェクト処理に特化したメモリアクセス回路について述べる．第3節では，プロセッサで高速に行う必要のある演算処理を解析し，それらを専用H/Wとして実現する方法について述べる．第4節では，提案するアーキテクチャをHDLで実装した結果と，ISS（Instruction Set Simulator）を用いて画像エフェクト処理の実行速度を測定した結果について述べる．

2. 画像エフェクト処理用メモリアクセス回路

2.1 画像エフェクト処理の特長の解析

画像エフェクト処理では，まず入力画像中の参照するピクセルの座標を計算し，その座標のピクセル値や座標値を用いて出力画像のピクセル値を計算する．そこで，入力画像中で参照されるピクセルの座標の計算と出力画像のピクセル値の計算に着目して，画像エフェクト処理を分類する．ここで，入力画像を$I_i, i = 1, \ldots, n$，出力画像をI_0，画像I_iの座標(x_j, y_j)のピクセル値を$p_i[x_j, y_j]$と表す．また，ピクセル値，x座標値，y座標値を求める関数をそれぞれf_p, f_x, f_yとする．f_p, f_x, f_yは画像エフェクト処理に応じて決定される線形あるいは非線型の関数である．このとき，画像エフェクト処理は，出力画像の全ての座標(x, y)に対してピクセル値$p_0[x, y]$を求める処理として定義される．なお，各分類で例示する画像エフェクト処理については，付録にエフェクト処理実行後の出力画像を掲載する．

単ピクセル演算系　単ピクセル演算系の画像エフェクトには，ソラリゼーションエフェクト，風エフェクト等がある．単ピクセル演算系の画像エフェクトでは，出力画像I_0のピクセル値$p_0[x_0, y_0]$を，入力画像I_1上の同じ座標(x_0, y_0)のピクセル値$p_1[x_0, y_0]$と座標値(x_0, y_0)を用いて計算する．

$$p_0[x_0, y_0] = f_p(x_0, y_0, p_1[x_0, y_0])$$

座標変換系　座標変換系の画像エフェクトには，極座標変換エフェクト，ジグザグエフェクト等がある．座標変換系の画像エフェクトでは，入力画像I_1中の参照するピクセルの座標(x_1, y_1)を計算し，座標(x_1, y_1)のピクセルデータ$p_1[x_1, y_1]$を出力画像I_0のピクセル値$p_0[x_0, y_0]$とする．

$$\begin{aligned} x_1 &= f_x(x_0, y_0) \\ y_1 &= f_y(x_0, y_0) \\ p_0[x_0, y_0] &= p_1[x_1, y_1] \end{aligned}$$

バンプマッピング系　バンプマッピング系の画像エフェクトでは，入力画像I_2のピクセルデータ$p_2[x_0, y_0]$を用いて，入力画像I_1の参照するピクセルの座標(x_1, y_1)を計算し，座標(x_1, y_1)のピクセルデータ$p_1[x_1, y_1]$を出力画像I_0のピクセルデータ$p_0[x_0, y_0]$とする．

$$\begin{aligned} x_1 &= f_x(x_0, y_0, p_2[x_0, y_0]) \\ y_1 &= f_y(x_0, y_0, p_2[x_0, y_0]) \\ p_0[x_0, y_0] &= p_1[x_1, y_1] \end{aligned}$$

テクスチャマッピング系　テクスチャマッピング系の画像エフェクトでは，出力画像I_0のピクセルデータ$p_0[x_0, y_0]$を，座標値(x_0, y_0)，入力画像I_1の同じ座標(x_0, y_0)のピクセルデータ$p_1[x_0, y_0]$，テクスチャ画像I_2の同じ座標(x_0, y_0)のピクセルデータ$p_2[x_0, y_0]$を用いて計算する．

$$p_0[x, y] = f_p(x_0, y_0, p_1[x_0, y_0], p_2[x_0, y_0])$$

畳み込み演算系　畳み込み演算系の画像エフェクトには，エンボスエフェクト等がある．畳み込み演算系の画像エフェクトでは，出力画像I_0のピクセルデータ$p_0[x_0, y_0]$を，入力画像I_1の座標(x_0, y_0)のピクセルデータ$p_1[x_0, y_0]$とその周囲の座標のピクセルデータを用いて計算する．

$$\begin{aligned} x_1 &= f_{x1}(x_0, y_0) \\ y_1 &= f_{y1}(x_0, y_0) \\ x_2 &= f_{x2}(x_0, y_0) \\ y_2 &= f_{y2}(x_0, y_0) \\ &\vdots \\ p_0[x_0, y_0] &= f_p(p_1[x_1, y_1], p_1[x_2, y_2], \ldots) \end{aligned}$$

再帰演算系　再帰演算系の画像エフェクトとして，各種のIIRフィルタが存在する．再帰演算系の画像エフェクトでは，出力画像I_0のピクセルデータ$p_0[x_0, y_0]$を，入力画像と出力画像の座標(x_0, y_0)の周囲の複数の座標(x_0, y_0の値から計算)のピクセルデータを用いて計算する．

$$
\begin{aligned}
x_{01} &= f_{x01}(x_0, y_0) \\
y_{01} &= f_{y01}(x_0, y_0) \\
x_{11} &= f_{x11}(x_0, y_0) \\
y_{11} &= f_{y11}(x_0, y_0) \\
x_{02} &= f_{x02}(x_0, y_0) \\
y_{02} &= f_{y02}(x_0, y_0) \\
x_{12} &= f_{x12}(x_0, y_0) \\
y_{12} &= f_{y12}(x_0, y_0) \\
&\vdots \\
p_0[x_0, y_0] &= f_p(p_0[x_{01}, y_{01}], p_0[x_{02}, y_{02}], \ldots, \\
&\qquad p_1[x_{11}, y_{11}], p_1[x_{12}, y_{12}], \ldots,)
\end{aligned}
$$

画像生成系　画像生成系の画像エフェクトとして，文字や模様を生成する各種の画像エフェクトが存在する．画像生成系の画像エフェクトでは，入力画像を必要とせず，固有のピクセルデータ生成関数を用いて画像の生成を行う．

$$
p_0[x_0, y_0] = f_p(x_0, y_0)
$$

座標変換系の画像エフェクト処理では，入力のピクセルデータをそのまま出力のピクセルデータとするため，メモリから読み出したピクセルデータをプロセッサまで転送する必要はない．したがって，図1のように，変換前と変換後の座標情報をもとに，入力画像のメモリ領域から出力画像のメモリ領域へピクセルデータを移動させる（アドレスマッピング）ことで，プロセッサからのメモリアクセスのコストを削減する．

また，単ピクセル演算系の画像エフェクト処理では，出力画像と同一の座標のピクセルデータのみを入力として使用するため，図2のように，入力画像の先頭から逐次的にピクセルデータをプロセッサに転送し，計算後のピクセルデータを同じ順番で出力画像に書き込むストリーム処理によって，プロセッサからのメモリアクセスのコストを削減する．

その他の系列の画像エフェクト処理についても，プロセッサからのメモリアクセスのコストを削減する方法を簡単に述べる．バンプマッピング系の画像エフェクト処理では，入力画像 I_2 のピクセルデータを先頭から逐次的にプロセッサに転送し，プロセッサで計算した座標をもとに入力画像 I_1 から出力画像 I_0 へのアドレスマッピングを行うことでメモリアクセスのコストを削減できる．テクスチャマッピング系の画像エフェクト処理では2つの入力画像に対して入力ストリームを持つことで，畳み込み演算系，再帰演算系の画像エフェクト処理では処理に必要な各座標に対して入力ストリームを持つことで，それぞれ単ピクセル演算系の画像エフェクト処理と同様にストリーム処理によってプロセッサからのメモリアクセスのコストを削減できる．画像生成系の画像エフェクト処理では，出力画像への書き込みを先頭から逐次的に行うことでメモリアクセスのコストを削減できる．

そこで本稿では，プロセッサとメモリの間にメモリアクセス回路を置き，メモリアクセス回路で，入出力画像へのアクセス

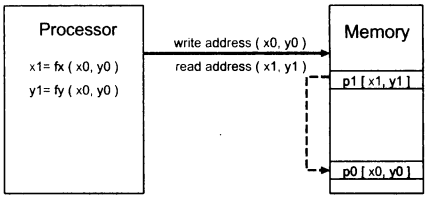

図1　座標変換系の画像エフェクト処理でのデータの流れ

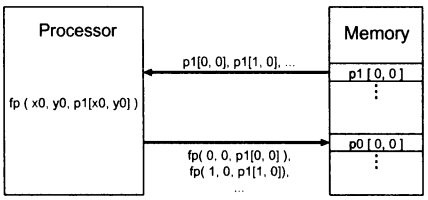

図2　単ピクセル画素値演算系の画像エフェクト処理でのデータの流れ

に関する処理を実行することで，プロセッサのメモリアクセスにかかるコストを軽減する方法を提案する．

2.2　メモリアクセス回路

本節では，メモリアクセスパターンの解析に基づいた画像エフェクト処理のためのメモリアクセス回路を提案する．提案するメモリアクセス回路は，以下のような機能を持つ．

- x, y 座標による画像へのアクセス

入力画像，出力画像へのアクセスを，ピクセルの x, y 座標の指定によって行う．

- ピクセル座標生成

画像をストリーム処理するときに，画像の先頭から順番に座標を自動生成する．

- 入力画像の先読み

入力画像を先頭から逐次的に読み込み，内部のバッファに格納する．プロセッサから読み込み要求があったときに内部のバッファからデータを取り出してプロセッサへ渡す．

- アドレスマッピング

指定された入力画像の座標と出力画像の座標をもとに，入力画像の読み込みから出力画像への書き込みまでの処理をプロセッサを介さずに行う．

- 補間処理

入力画像のピクセルの座標として指定された（x, y）の値が小数部分を持つ場合，指定された座標の近傍のピクセルデータを使って補間処理を行う．

- 書き込み要求のバッファリング

書き込みデータと，出力画像の x, y 座標の値を内部バッファに書き込み，内部バッファからピクセルデータと x, y 座標を取り出してメモリアクセスを行う．プロセッサでは内部バッファへの書き込み終了後次の処理へ移る．

以上の機能を持ったメモリアクセス回路の構成を図3に示し，各機能ブロックについて説明する．

（1）Core I/F は，プロセッサから入力された要求を識別し，その結果に応じて各機能ユニットへ要求を転送する．（2）Store

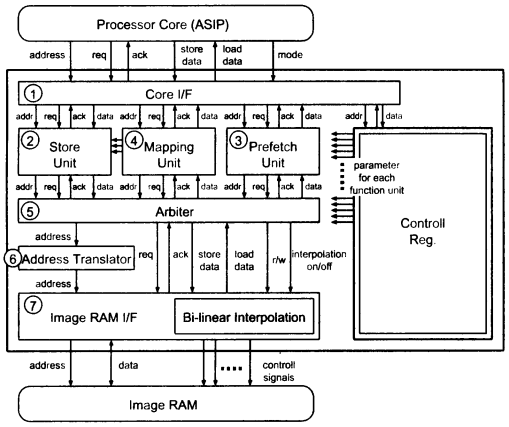

図 3　画像エフェクト処理に特化したメモリアクセス回路

Unit は，画像メモリへの書き込み要求をバッファリングする．(3) Prefetch Unit は，画像の先読みを行う．(4) Mapping Unit は，指定された座標からピクセルデータを読み込み，プロセッサを介さずに Store Unit へ転送することで，座標変換を実現する．(5) Arbiter は，Store Unit, Mapping Unit, Prefetch Unit からのメモリアクセス要求を調停する．(6) Address Translator は，Store Unit, Mapping Unit, Prefetch Unit から出力された x, y 座標を，実際の物理アドレスへ変換する．(7) Image RAM I/F は，入力画像，および出力画像を格納するためのメモリへのアクセスを行う．Store Unit，Prefetch Unit，Mapping Unit は，それぞれ内部にピクセル座標生成回路を持ち，ストリーム処理時には画像の先頭から順番に座標を生成する．なお，図 3 の構成は入力画像と出力画像が全て同じメモリに格納されているときの構成であるが，Arbiter，Address Translator，Image RAM I/F をメモリ毎に持つことで，出力画像と入力画像で別のメモリを使用するアーキテクチャや，複数の入力画像を異なるメモリから読み込むようなアーキテクチャに対しても，提案するメモリアクセス回路は適用できる．

図 4 に，単ピクセル演算系の画像エフェクト処理を実行するときの，プロセッサ，メモリアクセス回路，メモリの動作を示す．単ピクセル演算系の画像エフェクト処理の場合，メモリアクセス回路では入力画像のピクセルデータの先読み処理（Read）と，プロセッサから返された，演算処理後のピクセルデータの書き込み処理（Write）を行う．先読みと書き込みの座標はメモリアクセス回路内部で生成する．このとき，プロセッサは先読みされたピクセルデータをメモリアクセス回路から取得する処理と，演算処理を施したピクセルデータをメモリアクセス回路へ転送する処理にかかる時間以外は，ピクセルの演算処理を実行することができる．

同様に，バンプマッピング系の画像エフェクト処理を実行するときの，プロセッサ，メモリアクセス回路，メモリの動作を図 5 に示す．メモリアクセス回路では，2 種類の入力画像のうち，座標変換に使用する画像のピクセルデータの先読み処理（Read1）と，プロセッサから転送された，座標変換後の座標を

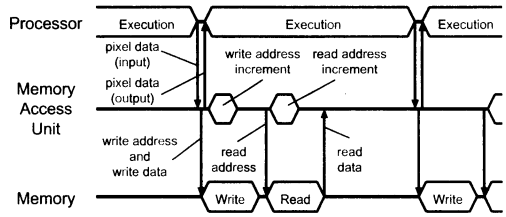

図 4　単ピクセル演算実行時の処理の流れ

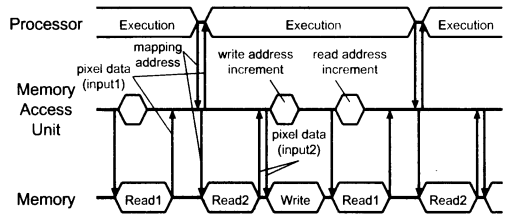

図 5　バンプマッピング実行時の処理の流れ

使った座標変換処理（Read2, Write）を行う．座標変換で使用する入力画像の座標と出力画像の座標は，メモリアクセス回路内のピクセル座標生成回路で生成する．このとき，プロセッサは，先読みしたピクセルデータをメモリアクセス回路から取得する処理と，座標変換を行った後の座標をメモリアクセス回路へ転送する処理にかかる時間以外は，座標変換処理を実行することができる．

3. 演算処理の専用 H/W 化

本節では，さまざまな画像エフェクト処理で使用される演算処理の専用 H/W 化について説明する．

3.1 演算処理の解析

画像エフェクト処理を行ったときの実行サイクル数，および関数単位での実行サイクル数を ISS（Instruction Set Simulator）を用いて測定し，H/W による高速化が必要な演算処理を抽出する．

本稿では，Gimp [4]，Adobe 社の Photoshop [5] に組み込まれているフィルタ処理を対象とし，DLX [3] 互換プロセッサ上で実行したときの各演算処理の実行サイクル数を測定した．ただし，使用したプロセッサは浮動小数点命令の代わりに固定小数点命令を持つ．また，実行サイクル数は画像メモリへのアクセスにかかる時間を除いている．

表 1 に，解析した画像エフェクト処理の一部について，実行サイクル数の多い演算処理を示す．解析結果から，極座標変換などのような座標変換系エフェクト処理では距離関数，偏角計算，三角関数の実行サイクル数が多く，単ピクセル演算系のようにピクセルに対する演算を行う画像エフェクトでは，ピクセルデータに対する算術演算，論理演算の実行サイクル数が多いことがわかる．また，風エフェクト等では，自然なばらつきを表現するために乱数生成処理が多く用いられる．

以上の結果から，本稿では距離計算，偏角計算，三角関数，

表 1　画像エフェクト処理を構成する演算処理の内訳

エフェクト名	演算処理（関数単位）	比率 [%]
ジグザグ（座標変換系）	距離計算	90.240
	三角関数	5.208
	その他	4.552
極座標変換（座標変換系）	距離計算	49.242
	偏角計算	49.195
	その他	1.563
エンボス（畳み込み系）	ピクセル加減算	46.212
	ピクセル，固定小数乗算	20.776
	固定小数→ピクセル変換	19.030
	その他	13.982
風（単ピクセル演算系）	ピクセル加減算	29.824
	ピクセル，固定小数乗算	25.490
	ピクセル輝度計算	25.107
	乱数生成	3.821
	その他	15.758

ピクセル論理演算，ピクセル算術演算，乱数生成を H/W 化する．

3.2　距離計算，偏角計算，arctan

距離計算，偏角計算，arctan 関数は全て CORDIC アルゴリズム [6] によって計算できるため，これらの計算は CORDIC アルゴリズムを H/W 化することで実現する．

3.3　sin, cos, arcsin, arccos

三角関数の sin, cos, arcsin, arccos 関数は，arctan 関数や距離計算のように CORDIC アルゴリズムを使っても計算可能である．しかし，CORDIC アルゴリズムよりも高速な計算法として，LUT（Look Up Table）に一定間隔（$\frac{\pi}{128}$ 刻みなど）でサンプリングした値を格納し，LUT から取り出した値を線形補間する方法がある．sin, cos 関数は周期関数であり，また，arcsin, arccos 関数は定義域が狭いことから，sin, cos, arcsin, arccos 関数は十分小さな LUT で実現可能であるため，本稿では LUT と線形補間回路を用いてこれらの関数を実現する．他の三角関数として tan 関数があるが，tan 関数は sin, cos 関数の結果を用いて計算を行う．

3.4　ピクセルデータに対する論理演算，算術演算

RGB フォーマットで表されたピクセルデータに対する論理演算や算術演算は，RGB それぞれの画素に対して並列に行えばよいため，RGB 画素それぞれについて ALU を 用いることで高速に処理できる．

また，演算結果をピクセルデータと同じフォーマットで求める場合，加算や乗算を行う際にオーバーフローが生じる可能性があるが，そこで，加算や乗算を含んだ演算処理の途中結果を RGB 専用のアキュムレータに格納し，演算処理が終わった後にアキュムレータから RGB フォーマットで取り出すことで，一時的に RGB の画素の値が RGB フォーマットで表せる値を超えるような処理も実現できる．

3.5　乱　　数

乱数生成法として，LFSR（Linear Feedback Shift Register）法や，線形合同法 [7]，加算乱数生成法 [7] 等がある．一般的にもっとも周期の短い LFSR 法でも，画像エフェクト処理に使用する分には十分であるため，LFSR 法を使って乱数生成処理を実現する．

表 2　論理合成結果

	最大遅延時間 [ns]	面積 [gate]
プロセッサコア	5.30	約 95,000
メモリアクセス回路	6.71	約 50,000

テクノロジ: 0.11 μm CMOS

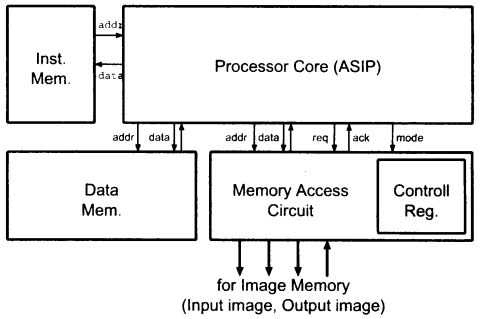

図 6　画像エフェクト処理プロセッサの構成

4.　評価実験

本節では，第 2 節，第 3 節で提案したアーキテクチャの評価を行う．はじめに，専用 H/W を組み込んだ ASIP とメモリアクセス回路を実装したときの論理合成結果を述べ，次に，画像エフェクト処理を実行したときの実行速度を ISS を用いて評価した結果を述べる．

4.1　論理合成結果

第 2 節で述べたメモリアクセス回路，および第 3 節で述べた専用 H/W を用いて各種演算処理を行う命令セットを持ったプロセッサコア（ASIP）を HDL で実装した．表 2 に，プロセッサコアとメモリアクセス回路を Synopsys 社の Design Compiler を用いて論理合成した結果を示す．

4.2　画像エフェクト処理の実行速度の評価

提案するアーキテクチャを持つ ASIP における，画像エフェクト処理の実行速度を，ISS を用いて評価した結果を述べる．

図 6 に，実装したメモリアクセス回路とプロセッサコアを用いた画像エフェクト処理システムの構成を示す．図 6 で，Inst. Mem. は命令メモリである．命令メモリには，画像エフェクト処理を C 言語で実装し，専用のコンパイラを用いてコンパイルしたプログラムが格納される．Data Mem. は画像エフェクトプログラムの中で使用するデータのうち，画像データ以外の一時変数を格納するためのメモリである．Processor Core は，DLX [3] 互換プロセッサに，第 3 節で述べた専用 H/W を用いて各種演算処理を行う命令を加えたプロセッサである．プロセッサの設計には ASIP 設計環境 ASIP Meister（PEAS-III）[8] を用いた．Memory Access Circuit は第 2 節で述べたメモリアクセス回路である．メモリアクセス回路の外部に画像データを格

表 3 ISS による実行速度の測定結果

画像エフェクト名	実行速度 [fps]		高速化率
	提案手法	DLX プロセッサのみ	
ジグザグ	55.99	2.60	21.53
極座標変換	73.18	1.42	51.54
ソラリゼーション	80.29	4.96	16.19
風	60.17	3.50	17.19
置き換え	95.10	19.55	4.86
エンボス	80.29	2.96	27.12

納するためのメモリが存在する.

画像エフェクト処理を図 6 の提案アーキテクチャで実行したときの実行速度と，画像エフェクト処理に特化した専用命令を持たない DLX プロセッサを使って実行したときの実行速度を表 3 に示す．測定の環境では，画像を格納するメモリへのアクセス時間は R/W ともに 20 [ns] とした．また，画像サイズは 256×256 [pixel×pixel] とし，システムクロックの周波数は 100 MHz とした.

表 3 の結果から，提案するアーキテクチャを持つ ASIP では，RISC プロセッサを用いた場合に比べて数倍から数十倍の速度で画像エフェクト処理を実行できることがわかる．また，提案するアーキテクチャの実行速度から，実時間処理として十分なフレームレートを達成できることがわかる.

5. 終わりに

本稿では，様々な画像エフェクト処理を実時間で実現するための手段として ASIP に着目し，画像エフェクト処理に特化した ASIP アーキテクチャを提案した．提案するアーキテクチャは，画像エフェクト処理において演算量の多い処理を高速化するための H/W を持つ ASIP と，画像エフェクト処理に特化したメモリアクセスユニットから成る．提案するアーキテクチャを HDL で実装して論理合成し，ISS によって性能を評価した結果，専用命令を持たない RISC プロセッサに比べて数倍から数十倍の性能が得られ，実時間処理として十分なフレームレートを達成できることがわかった.

謝辞

本研究を進めるにあたり貴重な御意見を頂いた，株式会社アクセル 佐々木譲氏，柴田高幸氏，森屋和喜氏，松浦一教氏，蟹江幸司氏に感謝致します．また，実装にあたってコンパイラと ASIP 設計の環境を提供して頂いた，田中浩明氏，小林悠記氏を始めとする大阪大学今井研究室の皆様に感謝致します.

文　献

[1] 石津 任章, 武田 幹雄, 馬場 祥宏, 佐野 誠, 藤原 義也, 原 正純, 森江 隆, 岩田 穆, "画像処理 IP と FPGA 実装," 信学技報 H14.10.24, 2002.

[2] Jeffrey C. Gealow and Charles G.Sodini, "A Pixel-Parallel Image Processor Using Logic Pitch-Matched to Dynamic Memory," IEEE Journal of Solid-State Circuits, Vol. 34, No. 6, June 1999, pp.831-pp.839.

[3] John L. Hennessy and David A. Patterson, "Computer Architecture: A Quantitive Approach," Morgan Kaufmann Publishers Inc., 1990.

[4] "GIMP," http://www.gimp.org/.

[5] "Photoshop," http://www.adobe.com/.

[6] J.E. Volder, "The CORDIC trigonometric computing technique," IRE Trans. Electron. Comput., vol. 8, pp.330-334, Sept. 1959.

[7] D.E. Knuth, "The Art of Computer Programming=3," 渋谷 政昭, サイエンス社, 1981 年.

[8] Makiko Itoh, Shigeki Higaki, Jun Sato, Akichika Shiomi, Yoshinori Takeuchi, Akira Kitajima, and Masaharu Imai, "PEAS-III: an ASIP design environment," Proceedings of International Conference on Computer Design, 17-20 September 2000, pp.430-436.

付　録

本稿で参照した画像エフェクト処理の出力画像を図 A.1-5 に示す.

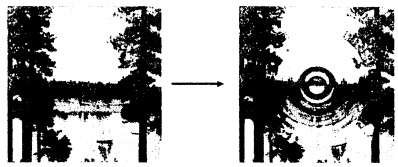

図 A·1 ジグザグエフェクト

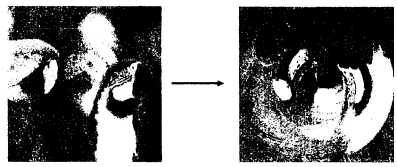

図 A·2 極座標変換エフェクト

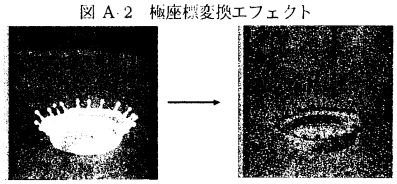

図 A·3 ソラリゼーションエフェクト

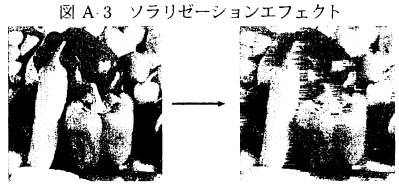

図 A·4 風エフェクト

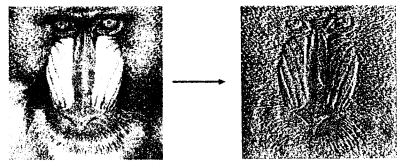

図 A·5 エンボスエフェクト