社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2004－ＳＬＤＭ－113 (20)
2004／1／23

FPAccA アーキテクチャへのラジオシティ法の実装と評価

城本 正尋† 河野 陽一† 越智 裕之† 津田 孝夫†

† 広島市立大学 情報科学部 情報工学科
〒 731-3194 広島市 安佐南区 大塚東 3-4-1
Tel:082-830-1550
E-mail: †{shiro,ochi}@arch.ce.hiroshima-cu.ac.jp

あらまし 本稿では，C 言語等の高級言語で書かれたプログラムを高速かつ容易に実行できる粗粒度再構成アーキテクチャとして，基本セル中に単精度浮動小数点演算器を搭載した FPAccA model 2.2 アーキテクチャを提案する．また実用的なアプリケーションの例として，ラジオシティ法によるレンダリングプログラムを取り上げる．0.35μm プロセスでの実現を仮定して評価したところ，0.25μm プロセスで実現されている Cerelon (500MHz) に比べ約 10 倍高速に処理できるという結果が得られた．また，レイアウト面積と消費電力に関しては，0.18μm プロセスを採用することで現実的な値になると考えられる．

キーワード FPGA，粗粒度再構成アーキテクチャ，浮動小数点演算器

Implementation and Evaluation of Radiosity Method on FPAccA Architecture

Masahiro SHIROMOTO†, Yoichi KAWANO†, Hiroyuki OCHI†, and Takao TSUDA†

† Department of Computer Engineering
Faculty of Information Sciences, Hiroshima City University
3-4-1, Ozukahigashi, Asaminami-Ku, Hiroshima, 731-3194, Japan
Tel:+81-82-830-1550
E-mail: †{shiro,ochi}@arch.ce.hiroshima-cu.ac.jp

Abstract In this report, we propose FPAccA model 2.2 architecture, a coarse-grained reconfigurable architecture that has a single-presision floating-point unit in each basic cell, that achieves high-performance execution of programs described in high-level languages such as C with low mapping effort. As an example of application, we implemented a rendering program using radiosity method. Assuming 0.35μm process, FPAccA achieved 10 times higher speed compared with Cerelon (500MHz) with 0.25μm process. It is expected that area and power will be feasible, if 0.18μm process is used.

Key words FPGA, coarse-grained reconfigurable architecture, floating-point unit

1. はじめに

近年の集積回路技術の進歩に伴い，きわめて高機能かつ高性能な LSI が実現できるようになった．しかし，ASIC の開発には多大なコスト，期間を要するようになりつつある．このような中で，小量多品種，短 TAT などの要求に応えるべく，FPGA (Field Programmable Gate Array) に代表される再構成デバイスが開発され，実用化に至っている．

FPGA はアプリケーションに応じて構成情報を変更することで，柔軟に専用のハードウェアを実現することができる．半導体の微細加工技術の進歩によって 1 チップの FPGA に搭載されるゲート数も飛躍的に増加し，より複雑なアプリケーションの実装も可能になってきた．その一方で，FPGA の大規模化による構成情報量や構成情報の生成に要する時間の増大等が問題となっている．そこで，FPGA では複数の基本セルで実現されるような複雑な機能を一つのセルにまとめることで，構成情報の減少や高性能化を狙った粗粒度再構成アーキテクチャが提案されている [1] [2].

FPAccA (Field Programmable Accumulator Array) アーキテクチャ [1] は，FPGA 基本セル中の LUT (Look-Up

Table) を高機能な ALU に置き換えた粗粒度再構成アーキテクチャである．FPGA よりも粗粒度とすることで，構成情報量の削減，構成情報の自動生成に要する時間の短縮，算術演算中心のアプリケーションでの速度，実装密度での優位性を狙っている．この FPAccA アーキテクチャに基づき，そのプロトタイプとして ALU に 8bit 整数演算器を実装した FPAccA model1.0 チップ[1] と，単精度浮動小数点演算器を実装した FPAccA model2.0 チップ[3] が試作されている．

本稿では，実用的なアプリケーションを実装するのに必要となる改良を FPAccA model 2.0 チップにほどこした FPAccA model 2.2 アーキテクチャを提案し，0.35μm プロセスで評価を行なう．実用的なアプリケーションの例としてラジオシティ法によるレンダリングプログラムを取り上げ，これを FPAccA に実装し、Celeron (@500MHz) と比較して 10 倍程度の処理速度の向上が得られることを示す．

以下，2 章ではラジオシティ法の説明をし，3 章で FPAccA model 2.2 アーキテクチャについて述べる．4 章では FPAccA model 2.2 アーキテクチャにラジオシティ法を実装し，5 章でその評価と考察を，6 章でまとめを述べる．

2. ラジオシティ法

ラジオシティ法[4] は熱力学法則に基づいた計算を行なうことによって，光の相互反射を実現させる高品位レンダリング手法である．つまり，物体表面を構成するポリゴン（多角形）を細かいパッチに分割し，各々について光の出入りを計算して輝度を求めようというものである．ラジオシティ法では，物体間の光の相互反射を取り扱うため，間接光による影響を考慮した非常に現実感の高い画像を生成できる．しかし，ラジオシティの計算にはフォームファクタ（form factor）と呼ばれるパッチやエレメント間で光が到達する割合を算出する膨大な計算が必要である．このため，並列計算機を用いて高速化する研究も行なわれている[5]．

2.1 基本手順

ラジオシティ法による画像生成の流れは以下の通りである．

1. モデリング

   ポリゴンで物体を定義する．

2. ポリゴンのメッシュ化

   各ポリゴンを適当な大きさに分割する．分割された各々をパッチと呼び，ラジオシティの計算単位とする．

3. ラジオシティ計算

   光のエネルギーが平衡状態に達しているとして，各パッチの光の入射と反射に注目すると次式が成立する．

$$B_iA_i = E_iA_i + \rho_i \sum_{j=1}^{n} B_jF_{ji}A_j \qquad (1)$$

ここで，B_i，B_j はパッチ i，j のラジオシティエネルギー，

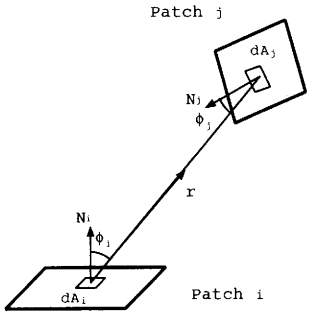

図 1 フォームファクタ

E_i はパッチ i の自己拡散エネルギー，A_i，A_j はパッチ i，j の面積，ρ_i はパッチの反射率，F_{ji} はパッチ j からパッチ i へのフォームファクタ，n はパッチ数である．

フォームファクタとは，あるパッチから他のパッチへ光のエネルギーが到達する率で，二つのパッチが図 1 に示されるような位置関係にあるとき，フォームファクタ F_{ij} は，次のように表される．

$$F_{ij} = \frac{1}{A_i}\int_{A_i}\int_{A_j} \frac{cos\phi_i cos\phi_j}{\pi r^2} H(p_i,p_j)dA_jdA_i$$
$$H(p_i,p_j) = \begin{cases} 1 & visible \\ 0 & invisible \end{cases} \qquad (2)$$

$H(p_i,p_j)$ はパッチ間の遮蔽物を考慮した，パッチ i からパッチ j が可視かどうかを表す関数である．

フォームファクタの対称性 $A_iF_{ij} = A_jF_{ji}$ により，式 (1) は次のように表すことができる．

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_jF_{ij} \qquad (3)$$

そして，式 (3) より得られる式 (4) の連立方程式を解き，全パッチのラジオシティ B_i を求める．

$$\begin{bmatrix} 1-\rho_1F_{11} & -\rho_1F_{12} & \cdots & -\rho_1F_{1n} \\ -\rho_2F_{21} & 1-\rho_2F_{22} & \cdots & -\rho_2F_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_nF_{n1} & -\rho_nF_{n2} & \cdots & 1-\rho_nF_{nn} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix} \qquad (4)$$

4. レンダリング

   求めた各パッチのラジオシティエネルギー B を基にレンダリングを行なう．

2.2 半球法

ラジオシティ法では式 (2) で求められるフォームファクタの計算が処理時間の大部分を占めるので，その高速化が重要となる．式 (2) を積分計算で求めるのは非常に困難であるため，近似値を求める方法として半球法[6] がある．

パッチ i を中心にした半球を考え，この表面を適当な解像度のメッシュに分割する．そして，パッチ i から各メッシュへの

フォームファクタ（デルタフォームファクタ ΔF）を計算し，ルックアップテーブルに入れておく．次に，レイトレーシング法を用いて，すべてのパッチを半球表面に投影し各メッシュについて距離が最短のものを残す．ルックアップテーブルを用いて，パッチ j が投影されたメッシュの ΔF の和を計算すると，F_{ij} が得られる．

2.3 漸進法

式(4)はパッチの数が増えれば巨大な行列となる．そこで，漸進法[7]では各パッチの光の放射に注目し，あるパッチから他のパッチへの光の放射を繰り返すことで，全パッチのラジオシティを平衡状態に徐々に近付けていく．1回の放射は次の代入式を用いて行なわれる．

$$B_j \leftarrow B_j + B_i(\rho_j F_{ji}) \qquad (j = 1, 2, \ldots, n) \qquad (5)$$

この式を，ラジオシティの大きいパッチ i から順に適用することで徐々に解に近付けていき，収束するまで繰り返す．

3. FPAccA model 2.2

3.1 アーキテクチャ

FPAccA model 2.2 アーキテクチャは32bit整数ならびに単精度浮動小数点数の演算を行なう基本セルが2次元アレイ状に並んだ再構成アーキテクチャである．基本セルとして“整数加減算セル”，“浮動小数点加減算セル”，“整数乗算・浮動小数点乗算セル”，“浮動小数点除算セル”，“型変換セル”，“メモリセル”の6種類のセルを有する．これらの基本セルは，セル間の接続を行なう“バス部分”と，演算器ならびにその入出力の選択回路からなる“アキュムレータ部分”から構成されている．なお，全てのセルはパイプライン動作が可能である．

3.1.1 バス部分のアーキテクチャ

FPAccA model 2.2 のバスアーキテクチャは，単精度浮動小数点数を一度に転送可能なデータバス(33bit幅)とゼロフラグ，プラスフラグ，マイナスフラグから構成されるフラグバス(3bit幅)からなり，それぞれ東西南北の隣接する各セルと往復各2本のバスで結ばれている．バス部分のアーキテクチャは6種類全てのセルで同様である．

3.1.2 アキュムレータ部分のアーキテクチャ

a） アキュムレータへの入力

型変換セルとメモリセル以外のセルは入力としてAとBの2つを持ち，“データバス”，“定数”，“自己の演算結果”を入力として与えることができる．“定数”は構成情報によって設定可能な32bitのレジスタである．アキュムレータの演算はA入力とB入力が揃った時点で開始される．A入力とB入力が同時に揃わなかった場合は，先に到達した値が演算器内に保持され，A入力とB入力が揃った時点で演算が開始される．

b） 条件フラグ

算術演算を行なうセルは，計算結果に基づいた条件フラグ（プラス，ゼロ，マイナス）を生成することができる．生成されたフラグを他のセルに入力することで，演算以外の処理を実行させる事が可能である．各セルは，得られたフラグを構成情報によって以下のような処理に対応付けることができる．

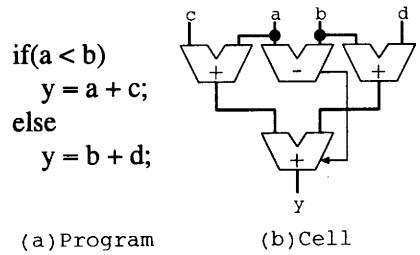

図2 else節を伴う条件分岐処理

- A入力通過（B入力通過）

  構成情報で設定した特定のフラグが入力された場合に，A（もしくはB）入力をそのまま出力に通過させることができる．

- 演算結果廃棄

  構成情報で設定した特定のフラグが入力された場合に，演算結果を出力せず廃棄する．

- 同期動作

  構成情報で設定した特定のフラグが入力するまで演算の開始を遅らせる．

c） メモリアキュムレータ

メモリアキュムレータは32bitレジスタ×4語から構成されている．そのレジスタを構成情報でランダムアクセスメモリかFIFOキューとして利用できる．

- ランダムアクセスメモリ

  それぞれのアドレス(8bit)は構成情報で設定する．外部から入力された値や処理途中の結果の保持などに利用する．

- FIFOキュー

  入力を構成情報で設定された遅延をかけて出力する．セルへの入力のタイミングを保つためのバッファとして利用する．

3.2 簡単な制御構文のFPAccAによる実装例

3.2.1 else節を伴うif文

else節を伴うif文の例として，図2(a)のC言語で書かれたプログラムに対応する論理的なセル間の接続を図2(b)に示す．

図2(b)の中央上段のセルは，図2(a)のプログラムの $a < b$ を実現している．a と b の値を入力し，$a - b$ が行なわれることで，演算の結果に依存したフラグが生成され，中央下段のセルに送られる．左右のセルは条件成立，不成立の場合の演算を予め計算しておき，演算結果を中央下段のセルに入力する．中央下段のセルは，入力されたフラグがマイナスの場合はA入力通過，ゼロかプラスの場合はB入力通過を実行するように構成情報で設定しておく．これにより，出力 y に演算結果を得る．

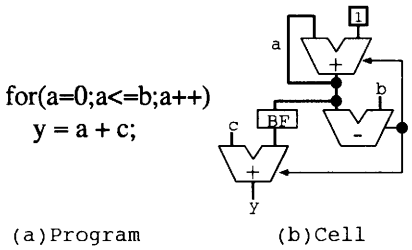

図3 繰り返し処理

3.2.2 繰り返し処理

繰り返し処理の例として，図3(a)のC言語で書かれたプログラムに対応する論理的なセル間の接続を図3(b)に示す．

図3(b)の上段のセルは，図3(a)のプログラムの$a++$を実現しており，aの値に1を足し込む処理を繰り返し行なっている．aに加える1はセル内に備わっている定数用のレジスタを構成情報で設定することで実現する．その下のセルは繰り返し処理の終了条件である$a <= b$を$a - b$を演算することで判別しフラグを出力している．下段左のセルでは，入力されたフラグがマイナスかゼロの場合$a + c$が実行され，それ以外のフラグの場合は演算結果を廃棄する．aはフラグとのタイミングを合わせるため，バッファを通過して入力される．バッファはFIFOの機能を持ったメモリセルで，下段右のセルの演算にかかる時間分の遅延が設定されている．

4. ラジオシティ法の実装

このラジオシティ計算の実装例では，FPAccA単体で全ての処理を行なうのではなく，基本的にはFPAccAを汎用プロセッサの外付けハードウェアエンジンのように利用することを想定している．汎用プロセッサ上の処理が実装部分まで到達した時点で，フォームファクタ計算に必要なデータを汎用プロセッサからFPAccAへ転送し，FPAccAでの計算結果を汎用プロセッサに返すという処理をラジオシティ計算が終了するまで繰り返す．

ラジオシティ法を実現するC言語のプログラム内で，フォームファクタ計算を行う部分の一部をFPAccA model 2.2アーキテクチャに実装した．この部分は，多くの浮動小数点算術演算と条件分岐によって構成され，FPAccAアーキテクチャに実装した場合に高速化ができると考えられる．

実装を行なうプログラムを図4に示す．これをFPAccA model 2.2アーキテクチャのセルに実装したものを図5に示し，実装に使用したセルの個数を表1に示す．

```
for(i=0;i<set;i++){
  for(p_ct=pmin;p_ct<=pmax;p_ct++){

    if(p_ct<p_rsl) phi=((float)p_ct+0.5)*Hs_pls;
    else           phi=(127.5-(float)p_ct)*Hs_pls;
    pp=phi*phi; ppp=pp*phi;
    sin_p=phi-ppp*(c1-pp*(c2-pp*(c3-c4*pp)));
    cos_p=1-pp*(c5-pp*(c6-pp*(c7-c8*pp)));
    if(p_ct<p_rsl){sin_phi=sin_p;cos_phi=cos_p;}
    else          {sin_phi=cos_p;cos_phi=sin_p;}

    tmp1.x=sin_phi*sintheta[i];
    tmp1.y=cos_phi;
    tmp1.z=sin_phi*costheta[i];
    tmp2.x=Rcos_b*tmp1.x-Rsin_b*tmp1.y;
    tmp2.y=Rsin_b*tmp1.x+Rcos_b*tmp1.y;
    tmp2.z=tmp1.z;
    rv.x=tmp2.x;
    rv.y=Rcos_a*tmp2.y-Rsin_a*tmp2.z;
    rv.z=Rsin_a*tmp2.y+Rcos_a*tmp2.z;

    a00=(A0[0].x*rv.x)+(A0[0].y*rv.y)+(A0[0].z*rv.z);
    a01=(A1[0].x*rv.x)+(A1[0].y*rv.y)+(A1[0].z*rv.z);
    a02=(A2[0].x*rv.x)+(A2[0].y*rv.y)+(A2[0].z*rv.z);
    a10=(A0[1].x*rv.x)+(A0[1].y*rv.y)+(A0[1].z*rv.z);
    a11=(A1[1].x*rv.x)+(A1[1].y*rv.y)+(A1[1].z*rv.z);
    a12=(A2[1].x*rv.x)+(A2[1].y*rv.y)+(A2[1].z*rv.z);
    as0=a00+a01+a02; as1=a10+a11+a12;

    Det=1.0e+5; as=1.0; flag=0;
    if(DetA0check){
      if(0.0<=a00&&0.0<=a01&&0.0<=a02&&0.0<as0){
        Det=DetA[0]; as=as0; flag=1;}}
    else{
      if(0.0>=a00&&0.0>=a01&&0.0>=a02&&0.0>as0){
        Det=DetA[0]; as=as0; flag=1;}}
    if(Vn4&&(flag==0))
      if(DetA1check){
        if(0.0<=a10&&0.0<=a11&&0.0<=a12&&0.0<as1){
          Det=DetA[1]; as=as1;}}
      else{
        if(0.0>=a10&&0.0>=a11&&0.0>=a12&&0.0>as1){
          Det=DetA[1]; as=as1;}}
     Zvalue[z++]=DetA/as;}}
```

図4 実装したプログラム

5. 評　　価

ラジオシティ法のFPAccA model 2.2アーキテクチャへの実装に関しての処理速度，面積，消費電力についての評価を述べる．なお評価に用いたテクノロジはローム社の0.35μm メタル3層プロセスである．

5.1 各セルの諸元

FPAccA model 2.2アーキテクチャの各セルを0.35μm プロセスで実現した場合のレイテンシとパイプラインピッチを表2に示す（単位はクロック数）．これらのセルは100MHzで動作可能なので，1クロック当たり10$n$secである．

表3にFPAccA model 2.2アーキテクチャの6種類のセルのレイアウト面積と，各セルにランダムパタンを入力した場合の回路シミュレータPowerMillで見積もった消費電力を示す．なお，クロック周波数は100MHz，電源電圧は3.3Vである．

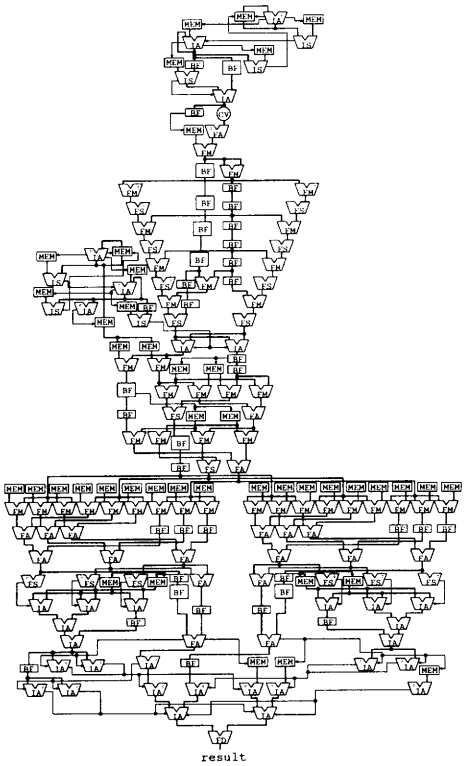

図5 ブロック図

表1 プログラム（図4）の実装に必要なセルの個数

	個数
整数加減算セル	38
浮動小数点加減算セル	35
整数・浮動小数点乗算セル	39
浮動小数点除算セル	1
型変換セル	1
メモリセル	81

表2 各セルのレイテンシとパイプラインピッチ

	Latency	Pipeline Pitch
整数加減算	2	1
浮動小数点加減算	4	1
整数乗算	3	1
浮動小数点乗算	4	1
浮動小数点除算	16	2
型変換 (Float to Int)	3	1
型変換 (Int to Float)	2	1
メモリ (Read and Write)	2	1

5.2 処理速度

FPAccA での計算時間は，図5をもとに論理シミュレーションを行ない，一回あたりの FPAccA での計算時間（単位はクロック数）を求め，それを C 言語プログラム内でラジオシティ計算終了までカウントすることで求めた．

汎用プロセッサの計算時間は，実際にラジオシティ計算を行なうプログラムを実行し，FPAccA model 2.2 へ実装した部分の処理時間を gettimeofday 関数を用いて計測した．また，計測には Celeron (@500MHz) を利用し，コンパイルには gcc -pipe -O2 -c -o(2.95.3) を用いた．

表3 各セルの面積と消費電力

	Area $[mm^2]$	Power $[mW]$
整数加減算セル	1.274	100
浮動小数点加減算セル	1.350	149
整数・浮動小数点乗算セル	2.103	219
浮動小数点除算セル	2.909	532
型変換セル	0.863	74
メモリセル	1.081	117

図6 scene data 1

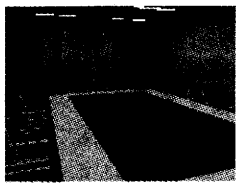

図7 scene data 2

表4 計算時間

		Times [sec]	
		scene data1	scene data2
総計算時間	Celeron (500MHz)	546.1	222.6
実装部分	Celeron (500MHz)	199.2	91.5
	FPAccA model 2.2 (100MHz)	25.0	9.3

表4に結果を示す．scene data1 はポリゴン数3,056，パッチ数5,477（図6），scene data2 はポリゴン数2,300，パッチ数4,306（図7）のシーンデータである．

表4より，今回実装した部分は FPAccA を採用することにより約8倍 (scene data1) ～ 10倍 (scene data2) 高速化ができることが示された．なお，実装を行なった部分の計算時間はラジオシティ計算全体の 40% 程度であることが表4より読みとれる．

5.3 面積と消費電力

表1と表3から，プログラムの実装に必要なレイアウト面積は約 $256mm^2$ となり，消費電力は約 27W となる．

5.4 考　　察

処理速度に関しては，Celeron が 0.25μm プロセスであるのに対し FPAccA が 0.35μm プロセスであったにもかかわらず，後者の方が1桁近く高速であることが示された．

面積と消費電力に関して，今回の合成に使った 0.35μm メタル3層プロセスでは実際のチップに実装するのは難しいが，例えば 0.18μm プロセスを採用すれば現実的な値になると考えられる．

6. ま と め

本稿では，単精度浮動小数点演算を含む実用的なアプリケーションを実装可能な FPAccA model 2.2 アーキテクチャを提

案した．条件分岐処理や繰り返し処理のような高級言語で頻繁に使われる記述を少数のセルで実装でき，高級言語との親和性が高いアーキテクチャであると言える．また本稿では，実用的なアプリケーションの例としてラジオシティ法によるレンダリングプログラムを取り上げ，その一部をFPAccA model 2.2アーキテクチャに実装し，汎用プロセッサと比較して高速であることを示した．今回，実装を行なった部分はアプリケーション全体の40%程度だったが，粗粒度再構成アーキテクチャであるFPAccAの構成情報量がコンパクトであることを生かし，今後動的に再構成可能なアーキテクチャに進化させていくことで，アプリケーションの中でより多くの部分を実装できるようになると考えられる．

今後の課題として，さらに様々なアプリケーションの実装と評価を行ない，FPAccAアーキテクチャを改良していくことが望まれる．また，0.18μmプロセス等で各セルの面積や速度，消費電力の評価を行なうとともに，配置配線ツールやコンパイラの開発も行なわなければならない．

謝　辞

本研究で利用しているラジオシティ法のプログラムを提供して頂きました岡山理科大学工学部情報工学科の上嶋明講師に深く感謝致します．本研究にあたり，東京大学大規模集積システム設計教育センター(VDEC)を通じてローム（株）から提供された0.35μmプロセスの設計規則などを利用させて頂きました．また，本研究はVDECを通じてCadence社，Synopsys社のツールを利用して行なわれたものです．ここに記して謝意を表します．

文　献

[1] 越智 裕之: FPAccA: フィールドプログラマブルアキュムレータアレイ－FPAccA model 1.0チップの設計と評価，情報処理学会論文誌, Vol.40, No.4, pp.1717-1725, 1999.

[2] 佐藤 友美: アイビーフレックスのリコンフィギュラブル・プロセッサ・デバイス－DAP/DNA ダイナミック・リコンフィギュラブル・プロセッサの概要－，第1回リコンフィギャラブルシステム研究会論文集, pp.165-172, 2003.

[3] 河野 陽一，越智 裕之，津田 孝夫: FPAccA model 2.0チップの設計－再構成可能な演算器アレイ－，信学技報, Vol.99, No.503, pp.45-52, VDL99-103 2000.

[4] Goral C. M., Torrance K. E., Greenberg D. P., and Battaile B.: Modeling the Interaction of Light Between Diffuse Surfaces, Computer Graphics, Vol.18, No.3, pp.213-222, 1984.

[5] 上嶋 明，山崎 勝弘: 分散メモリ型並列計算機AP1000+上でのラジオシティ法の並列化，電子情報通信学会論文誌 (D-II), Vol. J80-D-II, No.7, pp. 1852-1859, 1997.

[6] Spencer S. N.: The Hemishere Radiosity Method: A Tale of Two Algorithms, Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics, pp.127-135, 1990.

[7] Cohen M. F., Chen S. E., Wallace J. R., and Greenberg D. P.: A Progressive Refinement Approach to Fast Radiosity Image Generation, Computer Graphics, Vol.22, No.4, pp.75-84, 1988.