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

2004－EVA－8 (7)
2004／3／18

リコンフィギャラブルレイトレーシングマシンの演算制度に関する評価

中村 昇悟　　柴田 裕一郎　　小栗 清

{nezumi, shibata, oguri}@pca.cis.nagasaki-u.ac.jp

長崎大学工学部情報システム工学科
〒852-8521 長崎市文教町 1-14

概 要

FPGA(Field Programmable Gate Array) などのリコンフィギャラブルデバイスにレイトレーシングマシンを実装するとき、固定小数演算のみを用いることは回路規模において大きなメリットとなる。また、固定小数演算に用いるビット幅を小さくすることも回路規模を小さくすることにつながり、その分、多くの演算器を並列に並べることにより性能向上が見込める。そこで本研究では、異なるビット幅の固定小数点演算を用いて生成された画像と浮動小数演算を用いて生成された画像を、複数のサンプルについて画像評価指数を使って比較をする。そして画像品質の劣化具合、および固定小数演算の必要なビット幅を検討する。実験の結果、固定小数演算に必要なビット幅は 30 ビットであることが明らかになった。また、固定小数演算を用いたレイトレーシング処理において、38 ビットほどのビット幅があれば固定小数演算における最良の画質を得られることを明らかにした。

Evaluation of Arithmetic Precision Required for a Reconfigurable Raytracing Machine

Shougo NAKAMURA　　Yuichiro SHIBATA　　Kiyoshi OGURI

Department of Computer and Information Sciences,
Faculty of Engineering, Nagasaki University
1-14 Bunkyo-machi, Nagasaki 852-8521 Japan

Abstract

For efficient implementation of a raytracing machine on reconfigurable devices such as Field Programmable Gate Arrays (FPGAs), it is advantageous to use only fixed point arithmetic. In addition, reduction of the bit width of fixed-point arithmetic enables high performance parallel processing making the best use of multiple small arithmetic units in a reconfigurable device. In this paper, we evaluate the arithmetic bit width required for a raytracing algorithm, comparing the quality of graphics generated by floating point arithmetic and fixed point arithmetic with various bit widths. Evaluation results show that 30-bit fixed point arithmetic is sufficient for the algorithm, and 38-bit arithmetic achieves the best quality in fixed point arithmetic.

1 はじめに

FPGA (Field Programmable Gate Array) や PCA (Plastic Cell Architecture) [1]、ダイナミックリコンフィギャラブルプロセッサ [2] など、再構成可能なデバイスを用いた新しいリコンフィギャラブルシステムが注目されている。これらのシステムは、汎用計算機のもつ柔軟性を維持しつつ専用計算機並の高い性能を達成し得るが、浮動小数用演算器を持たないことから、浮動小数演算を多用するアプリケーションを苦手とする欠点がある。このため、信号処理や画像処理のアプリケーションをリコンフィ

ギャラブルシステムに実装する際には、アプリケーションの要求する演算精度を保ちながらデータの整数化や固定小数化を行うのが一般的である。

本報告では、コンピュータグラフィックスのレイトレーシング処理をリコンフィギャラブルシステムで実現することについて検討する。レイトレーシング処理の演算の精度は画像の品質に影響を与えるが、固定小数点演算のビット幅を小さくすることは回路規模の縮小につながり、その分、多くの演算器を並列に並べることにより性能向上が見込める。そこで、浮動小数演算や異なるビット幅の固定小数演算を用いてレイトレーシングの処理を行い、得られた画像を画像評価指数を使って比較をする。そして画像品質の劣化具合、および固定小数演算に必要なビット幅について検討する。

2 関連研究

東北大学および宮城工業専門学校では、FPGA を用いた並列 CG 処理マシンである画像生成インテリジェントメモリ 3DCGiRAM を提案している [3]。3DCGiRAM では、1 セルのメモリ上に各部分空間内の物体情報を置き、セル内の計算エンジンで局所的に処理する。この計算処理はレイトレーシング法に基づくが、対象を三角パッチだけに絞ることでハードウェア化に適した Arenberg によるアルゴリズム [6] を採用し、交差判定の処理を単純化している。

各計算エンジンでは 32 ビットの固定小数点演算が行われる。座標系を [0.0 ～ 1.0] で正規化をし、Arenberg のアルゴリズムを改良することで、計算中に現れる全ての変数値を [−4.0 ～ 4.0] に収めている。

3DCGiRAM では、ソフトウェアシミュレータにより 64 ビット浮動小数演算で生成した画像を固定小数演算で生成した画像と比較することで、必要な演算幅を決定している。しかし、画像の比較は生成された画像の劣化具合を視認するにとどまっており、客観的な評価指数を使った比較は行なっていない。また、32 ビットと 16 ビットの 2 種類のみの固定小数演算を比較対象として選んでいる。

リコンフィギャラブルシステムには、アプリケーションに要求される最低限の精度の演算器を用いることで、動作周波数や並列度を向上できるメリットがある。そこで本報告では、複数のサンプルについて、異なったビット幅の固定小数点演算を用いて生成された画像と、64 ビットの浮動小数点演算を用いて生成された画像を、画質の評価指数を使って比較する。そしてレイトレーシング処理に必要な、最低限の演算ビット幅について検討する。

3 実験の方法

3.1 画質評価の指数

今回の実験では、異なる演算精度で計算された画像の比較に、ピーク信号対雑音比 PSNR (Peak-Signal to Noise Ratio) を用いた。PSNR は原画と処理画像の平均二乗誤差の逆数で表され、両画像の相違が大きければこの値は小さくなる。一般的に、PSNR≒40dB までは原画との見分けが難しく、20dB 付近になると見るに耐えない画質になると言われている [4]。

画像の大きさを $M \times N$ とし、原画像の画素値を $(R(m,n), G(m,n), B(m,n))$、比較対象となる画像の画素値を $(R'(m,n), G'(m,n), B'(m,n))$ とする。なお、各画素値は 0 ～ 1 の値をとるものとする。このとき PSNR は、

$$PSNR[dB] = -10\log_{10}\frac{MSE}{3}$$

で表される。ここで MSE は、

$$MSE = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}([R(m,n) - R'(m,n)]^2 + [G(m,n) - G'(m,n)]^2 + [B(m,n) - B'(m,n)]^2)$$

で与えられる。

3.2 実験用プログラム

実験のために実装したレイトレーシングのプログラムは、平面・球体を描写でき、反射・屈折やテクスチャマッピングの処理を行なえる。また、ハイライトや物体によってできる影の処理も実装している。座標系は [0 ～ 1] の正規化を行なっており、固定小数演算の場合には、符合 1 ビット、整数部 10 ビットで、残りのビットを小数部とした。プログラムの実装には C++を用い、固定小数演算を行なう為に固定小数のクラスライブラリである fPoint[5] を使用した。

3.3 評価用画像

前述のプログラムにより 64 ビットの浮動小数点演算で生成した画像と異なったビット幅の固定小数点演算で生成した画像を PSNR で比較し、画質の劣化具合を調べる。以下の 6 種類の画像で比較を行なった。なお、全ての画像において、視点位置は $(0, 0, -600)$、光源の RGB 値は $(1, 1, 1)$、環境光は $(0.2, 0.2, 0.2)$ とした。

画像 1（球体のみの画像） 光源の向きは $(-1, 1, -1)$、背景色の RGB 値は $(0.5, 0.5, 0.5)$、球体数は 4 個である。各物体の物体情報を表 1 に示す。

表 1: 画像 1 の物体情報

	x 座標	y 座標	z 座標	半径	色 (R,G,B)
球体 1	-120	-50	-25	100	(0,0,1)
球体 2	120	-50	-25	100	(1,0,0)
球体 3	0	20	250	120	(1,0,1)
球体 4	0	250	350	70	(0,1,0)

画像 2（床面のある画像） 光源の向きは $(0, 1, -1)$、背景色は $(0.25, 0.25, 0.25)$、球体数は 7 個である。各物体の物体情報を表 2 に示す。

表 2: 画像 2 の物体情報

	x 座標	y 座標	z 座標	半径	色 (R,G,B)
球体 1	80	100	212.9	70	(1,0,0)
球体 2	-80	100	212.9	70	(0,1,0)
球体 3	-80	-100	-52.9	70	(0,0,1)
球体 4	80	-100	-52.9	70	(1,1,0)
球体 5	200	0	80	70	(1,0,1)
球体 6	-200	0	80	70	(0,1,1)
球体 7	0	0	80	70	(1,0.4,0)
床面		-200			(0.25,0.5,0.75)

画像 3（完全鏡面反射する物体がある画像） 光源の向きは $(0, 1, -1)$、背景色は $(0, 0, 0)$、球体数は 7 個である。各物体の物体情報を表 3 に示す。

画像 4（バンプマッピングをした画像） 光源の向きは $(0, 1, -1)$、背景色は $(0.25, 0.25, 0.25)$、球体数は 4 個である。各物体の物体情報を表 4 に、球体にしたバンプマッピングの情報を表 5 示す。

表 4: 画像 4 の物体情報

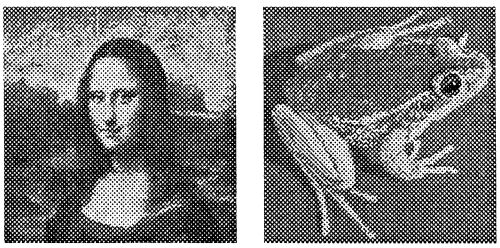
	x 座標	y 座標	z 座標	半径	色 (R,G,B)
球体 1	-180	-150	250	160	(0,0,1)
球体 2	180	-150	250	160	(1,0,0)
球体 3	230	200	500	200	(0,1,0)
球体 4	-230	200	500	200	(1,1,0)
床面		-350			(0.25,0.5,0.75)

画像 5（市松模様をマッピングした画像） 光源の向きは $(0, 1, -1)$、背景色は $(0.5, 0.5, 0.5)$、球体数は 2 個である。各物体の物体情報を表 6 に示す。

画像 6（画像を球体にマッピングした画像） 光源の向きは $(0, 1, -1)$、背景色は $(0.5, 0.5, 0.5)$、球体数は 2 個である。各物体の物体情報を表 7 に示す。また、球体 1 にマッピングした画像を図 1(a)、球体 2 にマッピングした画像を図 1(b) に示す。

表 5: 画像 4 のバンプマッピングの設定

	波の中心座標 (v,y)	波の波長	波の振幅
球体 1	$(\pi, 0.6\pi)$	0.105	0.25
球体 2	$(\pi, 0.6\pi)$	0.105	0.25
球体 3	$(\pi, 0.6\pi)$	0.21	0.25
球体 4	$(\pi, 0.6\pi)$	0.21	0.25



(a) 球体 1 にマッピング (b) 球体 2 にマッピング

図 1: 球体にマッピングした画像

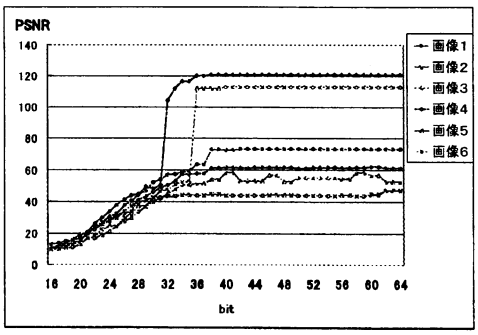
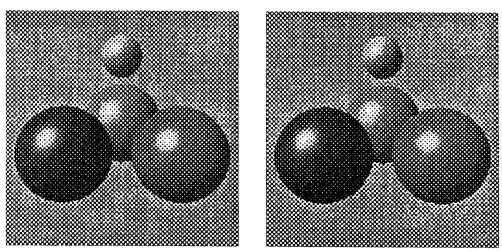


図 2: 各ビット幅における PSNR



(a) 原画像 (b) PSNR≒40dB

図 3: 画像 1

表 3: 画像 3 の物体情報

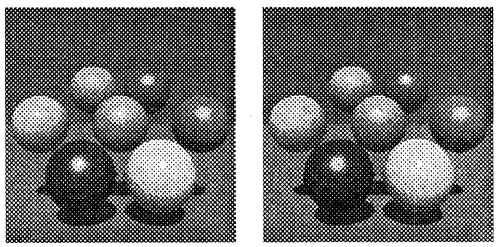
	x 座標	y 座標	z 座標	半径	色 (R,G,B)	反射の有無
球体 1	0	220	80	65	(1,0,0)	無
球体 2	75	-175	50	75	(0,1,0)	無
球体 3	-75	-175	50	75	(0,0,1)	無
球体 4	200	-200	50	50	(1,1,0)	無
球体 5	-200	-200	50	50	(1,0,1)	無
球体 6	-150	50	100	130	(0.125,0.125,0.125)	有
球体 7	150	50	100	130	(0.125,0.125,0.125)	有
床面		-250			(0.25,0.5,0.75)	無

表 6: 画像 5 の物体情報

	x 座標	y 座標	z 座標	半径	模様の縦・横幅	色 1(R,G,B)	色 2(R,G,B)
球体 1	-100	-60	0	140	$36 * rad$	(0.5,1,0.25)	(1,1,1)
球体 2	180	-20	280	180	$36 * rad$	(0,1,1)	(1,1,0)
床面		-200			800	(0.5,0,0)	(0,0,0)

4 結果と考察

各画像における PSNR の値を示したグラフを図 2 に示す。画像 1 では PSNR は固定小数 34bit から減少し始め、26bit のときに 40dB となった。画像 2 では PSNR は 35bit から減少し始め、固定小数 30bit で 40dB になった。画像 3 の場合は、PSNR の減少は 36bit から始まり、30bit で 40dB になった。画像 4 では、34bit から PSNR が減少し始め、28bit で 40dB になった。画像 5 では 32bit から PSNR が減少し始め、28bit で 40dB になった。画像 6 では 37bit から PSNR が減少し、27bit で 40dB になった。各画像における原画像と PSNR が 40dB 付近の画像を図 3 から図 8 に示す。

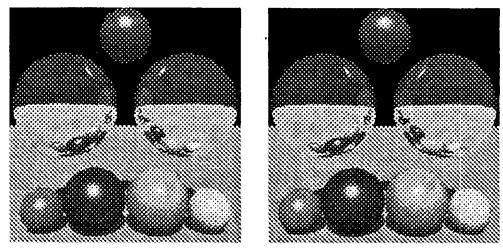


(a) 原画像　(b) PSNR≒40dB

図 4: 画像 2

各画像について PSNR が 40dB、20dB になったビット幅をまとめたものを表 8 に示す。この表より、全ての画像において PSNR=40dB を満たすには 30 ビット必要であり、PSNR=20dB を満たすには 23 ビット必要であることが分かる。

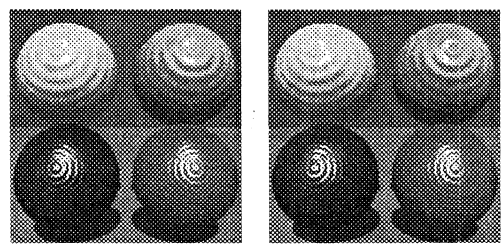
図 2 に示した結果より、市松模様のマッピングの場合、浮動小数点 64 ビットで生成した画像と固定小数点 64 ビットで生成した画像を比較すると PSNR



(a) 原画像　(b) PSNR≒40dB

図 5: 画像 3

が 47dB と低い。その理由を調べる為、画像 5 と同じ物体配置でいろいろな場合の画像を比較した。その結果を表 9 に示す。



(a) 原画像　(b) PSNR≒40dB

図 6: 画像 4

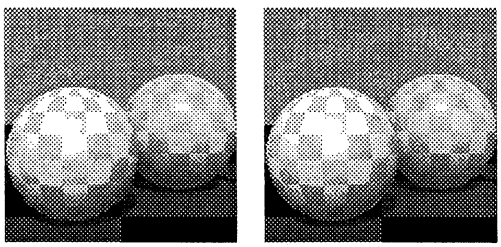
この表中の d は市松模様の大きさを決める変数のことである。床に市松模様マッピングをすると、PSNR が低くなっている。また、市松模様の大きさを小さくするほど、PSNR が更に下がっていることが分かる。これは、固定小数演算では床にマッピ

表 7: 画像 6 の物体情報

	x 座標	y 座標	z 座標	半径	マッピングのサイズ	色 (R,G,B)
球体 1	-100	-60	-25	140	$48 * rad$	
球体 2	170	60	280	180	$36 * rad$	
床面		-200				(0,0.25,0.5)

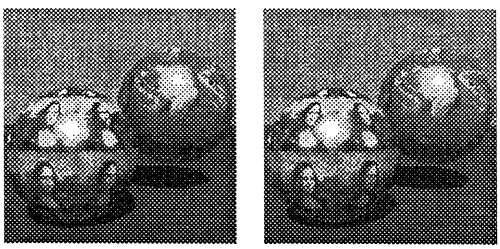
表 8: 各画像の PSNR=40dB,20dB になったビット幅

PSNR	画像 1	画像 2	画像 3	画像 4	画像 5	画像 6
40dB	26bit	30bit	30bit	28bit	28bit	27bit
20dB	20bit	23bit	23bit	21bit	21bit	21bit



(a) 原画像　(b) PSNR≒40dB

図 7: 画像 5



(a) 原画像　(b) PSNR≒40dB

図 8: 画像 6

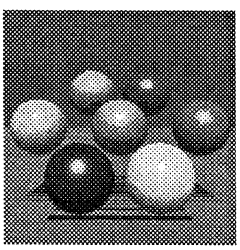


図 9: PSNR≒20dB における画像 2

表 9: 画像 5 のさまざまな場合の画質比較

条件	MSE	PSNR
マッピングなし	0	122.36
球体のみマッピング	0	121.56
全てマッピング (床 $d = 800$)	$5.5 * 10^{-5}$	47.34
全てマッピング (床 $d = 400$)	$2.6 * 10^{-4}$	40.61
全てマッピング (床 $d = 200$)	$5.3 * 10^{-4}$	37.67
全てマッピング (床 $d = 100$)	$6.6 * 10^{-4}$	36.61

ングする市松模様の境界が浮動小数演算とは異なってしまい、床面の奥の方でマッピングされる市松模様が大きく異なってしまうからだと推測する。しかし、肉眼で確認したところ浮動小数点 64 ビットで生成した画像と固定小数点 64 ビットの画像で生成した画像にそれほどの違いはなく、むしろ固定小数点 64 ビットの画像の方が市松模様の境界がはっきりしてるようにも見えた。つまり、平面への市松模様のマッピングは、実際の画質の劣化よりも PSNR の値が低くなってしまうということである。

画像 2 の PSNR20dB の画像を図 9 に示す。図 4(b) と見比べると分かるが物体の影が一番劣化していることが分かる。これは他の全ての画像で同じような状態になった。そこで画像 4 と同じ物体配置で影のある画像と影のない画像を比較して、その劣化具合を調べた。その結果を表 10 に示す。この結果、影のある画像の方が劣化が大きく、34 ビットの場合は PSNR において 2 倍ほどの差ができていることが分かる。よって、影の劣化が画像の劣化に影響していることが分かった。影のない画像を固定小数演算で生成する場合は、必要なビット幅を減らすことができる。また影を計算するアルゴリズムを改良することにより影の部分の劣化を押えることができれば、影のある画像でも固定小数演算のビット幅をもっと減らすことができると推測する。

図 2 より、すべての画像について、PSNR は 34~38 ビットの間で下がり始める傾向があり、それ以上においてほぼ画質が変わらないことが分かった。各画像において、16 ~ 64 ビットのビット幅で

表 10: 影の有無の画質比較

ビット幅	影無しの MSE	影有りの MSE	影無しの PSNR	影有りの PSNR
34	0	$1.6 * 10^{-5}$	111.45	52.54
28	$9.9 * 10^{-5}$	$5.8 * 10^{-4}$	44.82	37.14
22	$8.3 * 10^{-3}$	$4.8 * 10^{-2}$	25.57	17.97
16	0.31	0.34	9.89	9.50

表 11: PSNR の最大値とビット幅 32bit における PSNR の値

	画像 1	画像 2	画像 3	画像 4	画像 5	画像 6
PSNR 最大値	120.76	112.83	58.19	62.13	47.33	73.13
ビット幅 38bit の PSNR	120.74	112.71	54.16	61.27	44.56	72.94

計測した PSNR の最大値と、ビット幅 38 ビットにおける PSNR の値のを表 11 に示す。

この表より、16 ～ 64 ビットにおける PSNR の最大値と 38 ビットにおける PSNR の値があまり変わらないことが分かる。固定小数演算を用いて画像を生成する場合、ビット幅が大きければ画質の良い画像を生成できるわけではなく、38 ビット以上のビット幅は不要であることが分かった。つまり、固定小数演算を用いた画像生成において、38 ビットほどのビット幅があれば固定小数演算における最良の画質を得られることを示している。

6 つの画像の固定小数演算による劣化具合を測定した結果より、固定小数のビット幅は 30 ビットは必要ということが分かった。これは、あくまでも本研究で劣化の限界値として定めた PSNR=40dB を満たすビット幅である。しかし、生成した画像を見れば分かるが、原画像と PSNR≒40dB の画像を肉眼で見ても、ほとんど違いが分からないので十分原画像の画質を持っていると言える。

本研究に用いた画像生成のプログラムは座標系において [0 ～ 1] で正規化を行なったが、それ以外の部分で正規化できていない部分がある為、整数部 10 ビットとっている。アルゴリズムを改良して全ての変数を小さい範囲に正規化できれば、必要なビット幅を小さくできると推測する。

5 まとめ

本報告では、レイトレーシング法を用いた 3DCG の画像生成を固定小数演算で行なう場合に必要十分なビット幅について検討した。本研究で実装した画像生成のプログラムは、描写できる物体が平面と球体のみであるが、反射・屈折やテクスチャマッピングなどの処理を実装し、さまざまな画像を生成できるようにした。実験の結果、PSNR 40dB を満たす固定小数演算に必要なビット幅は 30 ビットであることが明らかになった。また、固定小数演算を用いて画像を生成する場合、38 ビットほどのビット幅があれば固定小数演算における最良の画質を得られることが分かった。今後は、座標系以外の変数の正規化や影を計算するアルゴリズムなどの改良を行ない、より少ないビット幅でより良い画像を生成できるようにする予定である。また、平面や球体以外の複雑な物体の描写についても検討をする予定である。演算精度について十分に検討した後は、リコンフィギャラブルデバイスへの実装に向けて、演算器の構成や全体のアーキテクチャについて検討していきたい。

参考文献

[1] H. Ito, R. Konishi, H. Nakada, K. Oguri, M. Inamori, and Akira Nagoya "Dynamically Reconfigurable Logic LSI – PCA-1: The First Realization of the Plastic Cell Architecture" IEICE Trans. on Information and Systems, pp. 859-867 (2003).

[2] 天野英晴 "ダイナミックリコンフィギャラブルプロセッサの研究開発動向" 情報処理学会研究報告, vol.2003, no.105, pp. 139–144 (2003).

[3] 鈴木健一, 帰山芳行, 杉山潤, 斎田泰昌, 大庭信之, 小林広明, 中村維男 "画像生成インテリジェントメモリ 3DCGiRAM の演算器設計" 並列処理シンポジウム JSPP 論文集, pp. 295–302 (2001).

[4] 波田哲朗, 國武惠明, 八幡勝也, 宮野章 "JPEG/JPEG-2000 方式による内視鏡画像の評価" 医療情報学 22 (Suppl.), pp. 702–703 (2002).

[5] "The Scalar, Vector, Matrix and Tensor Library" http://www.netwood.net/ edwin/svmtl/

[6] J.Arenberg "A Ray-Trianbles Intersection Algorithm" The Ray Tracing News, vol.3, no.1, pp. 2–4 (1989).