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

2007－ＳＬＤＭ－128（２）
2007／1／17

レイトレーシング専用コンピューティングシステムRAPLASのFPGAによる実装

財津 大地† 帰山 芳行† 鈴木 健一† 江川 隆輔† 大庭 信之†† 中村 維男†

† 東北大学大学院情報科学研究科 〒980-8579 仙台市青葉区荒巻字青葉 6-6-01
†† 日本アイ・ビー・エム(株) 東京基礎研究所 〒242-8502 大和市下鶴間 1623-14
E-mail: †{zaitsu,yoshi,suzuki,egawa,nakamura}@archi.is.tohoku.ac.jp, ††ooba@jp.ibm.com

あらまし レイトレーシング法による画像生成を高速化するために、専用システムRAPLASが提案されている。本研究では、RAPLASのFPGAによる実装を目的とし、FPGAの特徴を生かした設計に基づいて実装を行う。本稿では、高性能化が進むFPGAに搭載されている算術演算回路を活用し、専用の浮動小数点演算器を設計することで、RAPLASのための高速な交差判定ユニットを実現する。また、この設計に基づき、FPGA上での交差判定ユニットの動作検証も行う。

キーワード コンピュータグラフィックス、レイトレーシングハードウェア、交差判定手法

FGPA Implementation of the Computing System RAPLAS for Ray-Tracing

Daichi ZAITSU†, Yoshiyuki KAERIYAMA†, Kenichi SUZUKI†, Ryusuke EGAWA†,
Nobuyuki OHBA††, and Tadao NAKAMURA†

† Graduate School of Information Sciences, Tohoku University. Aramaki Aza Aoba 6-6-01, Aoba-ku, Sendai-shi, 980-8579 Japan
†† IBM Research Laboratory, IBM Japan, Ltd. Shimo-tsuruma 1623-14, Yamato-shi, 242-8502 Japan
E-mail: †{zaitsu,yoshi,suzuki,egawa,nakamura}@archi.is.tohoku.ac.jp, ††ooba@jp.ibm.com

Abstract Ray-tracing is one of the rendering methods based on the global illumination model, which is extensively used in the computer graphics. Although it generates photo-realistic images, it requires a large number of computations in the ray-object intersction test. To accelerate the intersection test, a new method based on a plane-sphere intersection algorithm and a hardware system RAPLAS have been proposed. This paper presents hardware implementation of the intersection core, which is the main unit of RAPLAS, using a FPGA. High performance intersection core is implemented by designing a dedicated floating-point calculator using DPS cores.

Key words Computer Graphics, Ray-tracing hardware, Intersection method

1. はじめに

現在、コンピュータグラフィックスは、映画やテレビコマーシャルなどの画像・映像製作の分野に限らず、建築や工業製品の意匠検討などの様々な分野において幅広く利用されている。これらの分野では高品質な画像を高速に生成する技術が強く望まれている。

高品質な写実的画像を生成する技術として、大域照明モデルを用いた画像生成手法がある。大域照明モデルを用いた画像生成手法では、Kajiyaによって提唱されたレンダリング方程式[1]に基づき、光源から直接届く一次光だけではなく、他の物体から反射・屈折・拡散した高次光を含めた光線と物体の物理的作用を忠実に再現することにより、反射・屈折・拡散により生じる光の集光によるコースティックス、ソフトシャドウなどの写実的な表現も再現することができる。

大域照明モデルを用いた画像生成の実現手法の1つに分散レイトレーシング法[2]がある。1984年、Cookらにより提案されたこの画像生成手法は、Whittedにより提案されたレイトレーシング法[3]の原理を応用することにより、Whittedによるレイトレーシング法では表現不可能であったアンチエリアシングや

ソフトシャドウなどの表現を可能とする手法である。また、本手法は、被写界深度やモーションブラーなどのカメラ効果も表現でき、より高度な写実的画像の生成を可能にする。分散レイトレーシング法では、初期光線の放射や、平面光源によるソフトシャドウの効果の計算、ランベルト面における拡散反射効果のサンプリングなどにおいて、放射状に多数の光線を拡散させ、それらの光線群と物体プリミティブとの交差判定処理を繰り返し行う。そのため、光線と物体との交差判定処理が膨大な量になってしまい、高速な画像生成の妨げとなっている。

このような状況において、我々は分散レイトレーシング法の高速化を目的として、平面-包含球交差判定手法を用いた手法[4][5]を提案している。本手法は、平面と包含球を用いた交差判定手法を分散レイトレーシング法に適用することにより、拡散光線と物体との交差判定処理を効率化し、処理量の大幅な削減を実現する。また、我々は本手法に基づく専用システム RAPLAS(Ray tracing Architecture based on PLane And Sphere intersection) を提案している。

本論文では、RAPLAS の実現に向けて、2 つの主要なサブユニットである平面-包含球交差判定サブユニット、光線-三角形交差判定サブユニットの設計、そして、これらサブユニットの FPGA 実装を行う。近年高性能化が進む FPGA に搭載されている高速な算術演算専用回路に着目し、専用の浮動小数点演算器を設計することにより、高速な交差判定処理ユニットを実現する。

2. レイトレーシング専用コンピューティングシステム RAPLAS

2.1 平面-包含球交差判定手法を用いた高速化

平面-包含球交差判定手法[4][5]では、分散レイトレーシング法において繰り返し行われる、同一の点を始点として放射状に光線を拡散させる光線群の処理に注目する。平面と包含球による交差判定を用いた分散レイトレーシングの処理手順を図 1 に示す。各ステップにおける処理内容の詳細を以下に述べる。

ステップ 1： 物体プリミティブを含む包含球を生成する

ステップ 2： 始点から空間に配置した仮想平面上の格子点を通過する光線群の生成を行う

ステップ 3： 光線群の始点と、格子点を結ぶ直線を通過する 2 種類の平面 (行平面・列平面) を生成する

ステップ 4： ステップ 3 の処理を繰り返し行うことにより、直行する平面群を生成する

ステップ 5： 各平面と包含球との交差判定処理を行う

ステップ 6： 包含球が直行する 2 平面の両方と交差する場合、2 平面の交線がなす光線と、包含球内の物体プリミティブとの交差判定処理を行う

なお、本システムでは、物体プリミティブは全て三角形パッチであると定義する。これは、多くの物体プリミティブはその

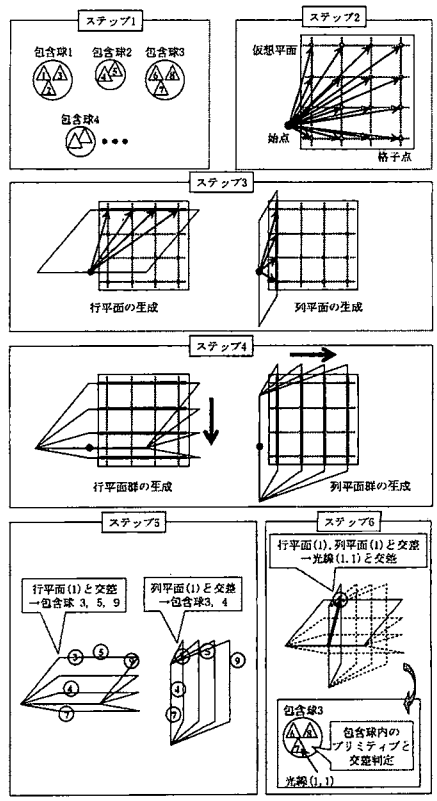

図 1 平面-包含球交差判定を用いた分散レイトレーシングの処理手順

最小基本単位である三角形に分解して表現することが可能であり、一般的には、物体プリミティブは三角形パッチに分割した後に画像生成処理が行われるためである[6]。

上述した平面-包含球交差判定手法を用いた高速化手法では、以下の 2 点により、処理量の大幅な削減を行うことが可能である。

- 交差判定処理回数の大幅な削減が可能

光線群が平面上の M × N 個の格子点を通過すると仮定すると、従来の方法では包含球 1 つと光線との交差判定処理回数の計算量オーダーは $O(M \times N)$ である。しかし、平面を用いる場合、$O(M+N)$ で済むため、処理回数を大幅に削減することが出来る。

- 少ない交差判定処理量

光線と包含球の交差判定を行うには、減算、外積計算、除算をそれぞれ 1 回づつ、内積計算を 2 回行う必要がある。しかし、平面と包含球の交差判定の場合、内積計算と加算をそれぞれ 1 回行うだけで交差判定ができるため、交差判定処理における浮動小数点演算量を削減することができる。

以上の 2 つの理由から、分散レイトレーシング法において平面と包含球の交差判定手法を導入することにより、包含球 1 個あたりの光線との交差判定処理回数を削減できるだけでなく、各交差判定処理における浮動小数点演算の処理量をも削減することが可能となる。

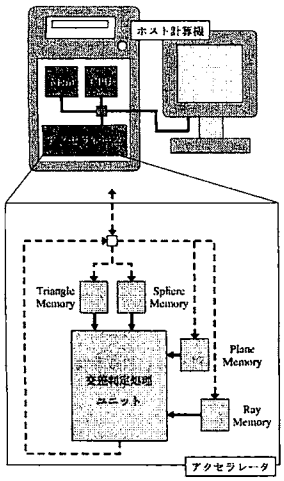

図2 RAPLAS のシステム構成

2.2 レイトレーシング専用計算システム RAPLAS

2.2.1 RAPLAS の概要

RAPLAS は、平面と包含球による交差判定手法を用いた分散レイトレーシング法のための専用システムである。本システムは、平面と包含球による交差判定手法に内在する処理の並列性を抽出し、専用ハードウェアで効率的に処理することにより、分散レイトレーシング法の高速処理を実現する。

2.2.2 システムの構成

図2に RAPLAS の構成を示す。RAPLAS はホスト計算機と、それに接続され交差判定処理を専門に行うアクセラレータから構成される。ホスト計算機には汎用の計算機が用いられ、平面群や光線の生成、シェーディング計算、ディスプレイ出力などの処理を担当する。そして、処理の大半を占める交差判定処理をアクセラレータが担当する。ホスト計算機が、アクセラレータ内のメモリにデータを書き込み、そのデータを元に、アクセラレータが平面-包含球交差判定、光線-三角形交差判定を行う。交差判定結果はホスト計算機へ返送され、ホスト計算機ではシェーディング計算や新たな平面群の生成が行われる。

アクセラレータは交差判定処理ユニット、および、各種データが格納される4つのメモリから構成される。Triangle Memory には三角形の識別番号 (ID)、幾何情報、Sphere Memory には包含球の ID、幾何情報、内包する三角形の ID、Plane Memory には平面の ID、幾何情報、Ray Memory には光線の ID、幾何情報がそれぞれ格納される。交差判定処理ユニットでは4つのメモリから入力されるデータを元に、平面-包含球交差判定処理、光線-三角形交差判定処理が行われる。

3. 交差判定処理ユニットの設計

3.1 FPGA による実装

近年の技術進歩によって、FPGA のロジック容量は年々大きくなっている。また、ロジックだけではなくメモリ、CPU コアなどの様々なハードウェアリソースが FPGA チップに搭載され、高速・高性能化している。Xilinx 社の FPGA である Virtex-4、Virtex-5 シリーズには、XtremeDSP スライス [7] と呼ばれる高速な算術演算専用回路が再構成可能なロジックの中に組み込まれており、Virtex-4 シリーズのコアには、この XtremeDSP スライスが最大で 512 個搭載されている。

一方、交差判定処理ユニットで行われる交差判定処理の特徴として、多くの浮動小数点演算処理が必要だということが挙げられる。1 枚の平面と 1 つの包含球との交差判定を行うには、6 つの浮動小数点演算が必要であるため (3.3.3 参照)、16 × 16 本の光線を放射状に拡散させる場合、1 つの包含球との交差判定処理に合計で 192 の浮動小数点演算が必要となる。また、光線-三角形交差判定処理については、1 本の光線と 1 枚の三角形との交差判定処理に 35 の浮動小数点演算が必要となる (3.3.4 参照)。

このように、RAPLAS システムの交差判定処理には多数の浮動小数点演算が要求されるが、平面-包含球交差判定処理、光線-三角形交差判定処理の各演算フローには大きな時間軸方向・空間軸方向の並列性が存在する。この様な性質を有するため、交差判定処理ユニットは FPGA に搭載されている多数の DSP コアを有効に活用することができる。DSP コアを用いて浮動小数点演算器を構成することにより、FPGA により提供されるハードウェア資源を存分に活用し、交差判定処理の高速化を実現できると考えられるため、本研究では交差判定処理ユニットの実装対象デバイスとして Xilinx 社 Virtex-4 FPGA を用いる。

3.2 浮動小数点演算器の設計

本論文において実装を行う交差判定処理ユニットは、浮動小数点演算器を組み合わせて構成される。そこで、交差判定処理ユニットの設計を行なうにあたり、高性能化が進む FPGA に搭載されている算術演算専用回路を利用した、専用の浮動小数点加減算/乗算器の設計について述べる。

3.2.1 設計方針

現在、汎用プロセッサにおいて、浮動小数点演算の規格として IEEE754 が定める単精度/倍精度形式 [8] が広く用いられている [9]。IEEE754 形式では図3に示すように、データは符号部 1 ビット、指数部 8 ビット、仮数部 23 ビットの計 32 ビットを用いて表現される。また、丸め処理については、切り上げ、切り下げ、切捨て、そして、最も近い偶数への丸めの 4 つのモードが用意され、演算の精度が保たれるよう工夫されている。

一方、本論文にて設計を行なう浮動小数点演算器では、独自の 24 ビット浮動小数点形式を採用する。この形式では図3に示すように、データは符号部 1 ビット、指数部 7 ビット、仮数部 16 ビットからなる。上記形式の採用理由は、実装対象ある Xilinx 社 Virtex-4 FPGA ボードに搭載されている XtremeDSP を用いるためである。XtremeDSP は最大で 18 ビットの算術演算を行うことが可能である。そこで、浮動小数点演算回路内では、16 ビットの仮数に桁上げのための 1 ビットと、暗黙の 1 を付加した 18 ビットを XtremeDSP の入力データとして演算を行う。また、演算速度を向上させるため、本浮動小数点演算器では特別な丸め回路の設計は行なっていない。そのため、浮動小数点の仮数部の演算中の丸め処理に関しては、全て切り捨てを適用

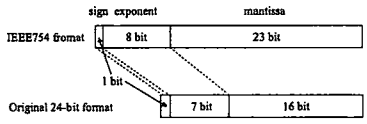

図 3　浮動小数点形式の比較

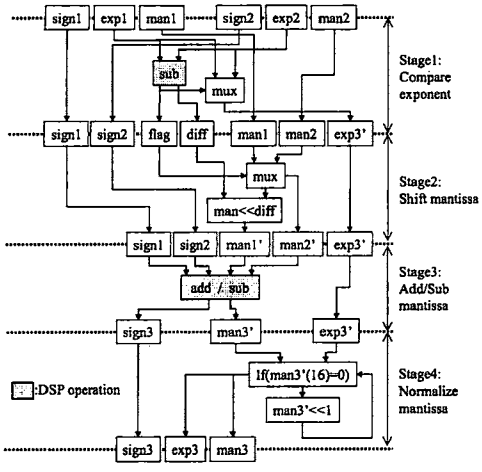

図 4　浮動小数点加減算器の構成

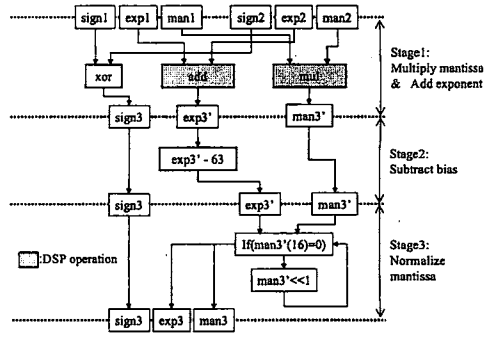

図 5　浮動小数点加減算器の構成

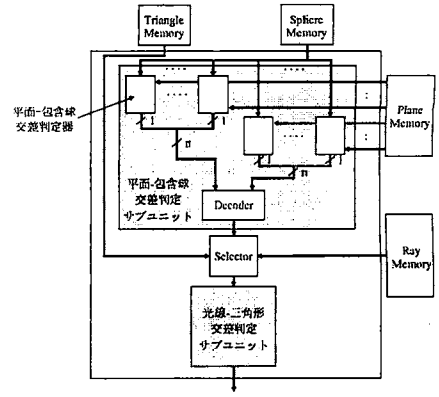

図 6　交差判定処理ユニットの構成

する。

画像生成処理の特徴として、高いスループットが要求されることが挙げられる。これより、本論文で実装を行うアクセラレータを構成する浮動小数点演算器にも高スループットが望まれる。そこで、設計する浮動小数点演算器は、スループットの向上のために、パイプライン方式をとる。

3.2.2　浮動小数点加減算器

浮動小数点加減算器の構成を図 4 に示す。図中の sign1、exp1、man1 はそれぞれ、入力データ 1 の符号部、指数部、仮数部を、同様に sign2、exp2、man2 は入力データ 2 の符号部、指数部、仮数部を表す。浮動小数点加減算回路は 4 段のパイプラインステージからなる。図中で DSP operation と示した処理は、XtremeDSP を用いて処理を行うことを示しており、浮動小数点加減算器では指数部比較のための減算、そして、仮数部の加減算処理に用いている。

以上のような、算術演算専用回路の活用と、パイプライン化により、高速で高スループットな演算器を実現する。

3.2.3　浮動小数点乗算器

浮動小数点乗算器の構成を図 5 に示す。図中の記号の意味については浮動小数点加減算器と同様である。浮動小数点乗算回路は 3 段のパイプラインステージからなる。ステージ 1 では、指数部の加算と、ステップ 2 の仮数部の掛け合わせが行なわれる。続いてステージ 2 では、ステップ 1 の残りの処理であるグタの減算が行なわれる。最後に、ステージ 3 では仮数の正規化が行なわれる。浮動小数点加減算器と同様に、以上のような算術演算専用回路の活用と、パイプライン化を行っている。

3.3　交差判定処理ユニットの設計

3.3.1　交差判定処理ユニットの構成

図 6 に交差判定処理ユニットの構成を示す。交差判定処理ユニットは平面-包含球交差判定サブユニット、そして、光線-三角形交差判定サブユニットの 2 個のサブユニットとセレクタから構成される。平面-包含球交差判定サブユニットでは平面と包含球との交差判定処理、光線-三角形交差判定サブユニットでは、光線と三角形との交差判定処理を行う。平面-包含球交差判定サブユニットと光線-三角形交差判定サブユニットの間に位置するセレクタは、平面-包含球交差判定サブユニットから送られてくる情報を元に、平面と包含球との交差の有無を判断し、交差している場合は Triangle Memory や Ray Memory よりデータをフェッチし、光線-三角形交差判定サブユニットへ送る。光線と三角形との交差判定処理の結果は、光線-三角形交差判定サブユニットからホスト計算機へ返送され、輝度計算・反射光線の生成に用いられている。

3.3.2　処理の流れ

メモリより送られてきた包含球のデータは平面-包含球交差判定器へ分配され、各平面のデータについては、それぞれ担当する平面-包含球交差判定器へ送られる。その後、各平面-包含球交差判定器では交差判定処理が行われ、判定結果を示す 1 ビットのフラグが生成される。全ての平面-包含球交差判定器から出

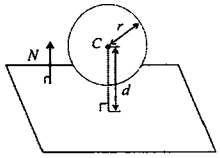

図7　平面と包含球

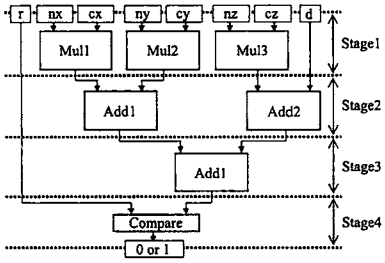

図8　平面-包含球交差判定器の構成

力されるフラグはデコーダに送られ、そこで、交差判定器に入力された包含球と交差する平面のIDが抽出される。セレクタでは、デコーダから送られてくる包含球と平面のIDより、次の光線-三角形交差判定サブユニットに送られる光線と三角形のIDが計算され、それぞれのデータがメモリからフェッチされる。フェッチされたデータはFIFOに格納され、順次光線-三角形交差判定サブユニットに送られて交差判定処理が行われる。

3.3.3　平面-包含球交差判定ユニットの設計

平面-包含球交差判定器では平面と包含球の交差判定処理が行われる。図7に示すように、中心座標 $C(c_x, c_y, c_z)$、半径 r の包含球と、法線ベクトル $N(n_x, n_y, n_z)$ で表される平面を考える。平面上の任意の点を $P(p_x, p_y, p_z)$ とすると、平面の方程式は以下のように表される。このとき平面の方程式は以下のように表される。

$$n_x x + n_y y + n_z z + p = 0 \tag{1}$$

これより、平面と包含球の中心との距離 d は

$$d = |n_x c_x + n_y c_y + n_z c_z + p| \tag{2}$$

と表され、$d \leq r$ が成り立つ場合、平面上の光群線と包含球内のプリミティブが交差する可能性が生じるため、続いて光線-三角形交差判定処理を行う。

設計した平面-包含球交差判定器の構成を図8に示す。画像生成処理においては、レイテンシの短縮よりもスループットを増大することの方が重要であるため、平面-包含球交差判定器はパイプライン方式をとっている。また、図中のMul、Addと表された演算では3.2にて設計した浮動小数点演算器を使用する。平面-包含球交差判定器は浮動小数点演算器のパイプラインステージを考慮すると12のパイプラインステージから構成され、合計12個のDSPコアを用いる。

3.3.4　光線-三角形交差判定サブユニットの設計

平面-包含球交差判定処理の結果、平面と包含球が交差する場合、平面を構成する光線群と包含球内のプリミティブが交差する可能性が生じるため、計算結果は平面-包含球交差判定器から光線-三角形交差判定サブユニットへと送られ、光線と三角形との交差判定処理が行われる。設計する光線-三角形交差判定サブユニットでは、光線と三角形の交差判定手法としてMöllerによる交差判定手法[10]、[11]を用いる。

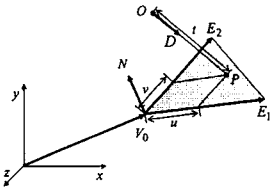

図9　Möllerによる交差判定手法

図9に示すような座標系 x, y, z における光線と三角形を考える。ここで、V_0：三角形の1頂点の位置ベクトル、E_1, E_2：V_0 から他の頂点へのベクトル、N：三角形の法線ベクトル、D：光線の単位方向ベクトル、O：光線の始点の位置ベクトル、P：三角形の存在する平面と光線の交点の位置ベクトル、t：光線の原点 O と三角形の存在する平面と光線の交点 P との距離、u, v：交点 P を E_1, E_2 を用いて表した場合のそれぞれのベクトルの寄与度である。Möllerによる交差判定手法では光線と三角形の交点は媒介変数 u, v で表現され、それぞれ以下の式により求められる。

$$\begin{pmatrix} t \\ u \\ v \end{pmatrix} = \frac{1}{N \cdot D} \begin{pmatrix} N \cdot (V_0 - O) \\ -(D \times (V_0 - O)) \cdot E_2 \\ (D \times (V_0 - O)) \cdot E_1 \end{pmatrix} \tag{3}$$

これより、式(4)の条件のときに光線と三角形が交差することが確定する。

$$0 \leq u,\ 0 \leq v,\ u + v \leq 1 \tag{4}$$

しかし、式(3)における除算を行うには加減算処理や乗算処理と比較して多くの処理時間とハードウェアコストがかかってしまう。そのため、計算式より除算を除外するために、式(3)を以下のように変形する。

$$\begin{pmatrix} t' \\ u' \\ v' \end{pmatrix} = N \cdot D \begin{pmatrix} t \\ u \\ v \end{pmatrix} = \begin{pmatrix} N \cdot (V_0 - O) \\ -(D \times (V_0 - O)) \cdot E_2 \\ (D \times (V_0 - O)) \cdot E_1 \end{pmatrix} \tag{5}$$

式(5)を用いる場合、交差条件は

$$0 \leq u',\ 0 \leq v',\ u' + v' \leq N \cdot D \tag{6}$$

となる。

設計した光線-三角形交差判定サブユニットの構成を図10に示す。平面-包含球交差判定器と同様、光線-三角形交差判定サブユニットはパイプライン方式をとる。また、図中のMul、Add、Subと表された演算では3.2にて設計した浮動小数点演算器を使用し、浮動小数点演算器のパイプラインステージを考慮すると、29のパイプラインステージから構成される。また、光線-三角形交差判定処理サブユニットは合計70個のDSPを用いる。

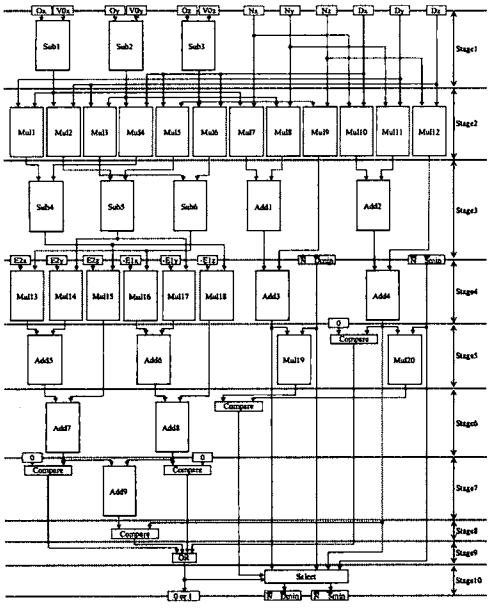

図 10　光線-三角形交差判定サブユニットの構成

表 1　実装対象

Board	Avnet MB Development Board	
FPGA	Device	Xilinx XC4VSX35
	Logic Cells	34560
	DSP	192
	Block RAM	3456Kbit

4. 論理合成・実装結果

今回実装に用いたのは、表 1 に示す Xilinx 社の FPGA XC4VSX35-10FF668、64MB DRAM などが搭載される Avnet 社の評価ボードである。評価回路は、RTL レベルの VHDL で記述し、Xilinx 社の ISE8.2 を用いて論理合成及び配置配線を行った。

論理合成に基づく、2 つの交差判定処理サブユニットのハードウェアのコスト及び性能を表 2 に示す。設計した 2 つの交差判定処理サブユニットは全スライスの約 88%を占有する規模である一方、DSP 使用率も約 86%であり、FPGA に搭載されている DSP を十分に活用できている。

論理合成・配置配線を行った各交差判定処理サブユニットは、JTAG 経由でホスト計算機から評価ボードへ転送して実装した。また、評価ボードをシリアルケーブルでホスト計算機と接続し、動作結果の確認をシリアルポート経由で行った。実装した各交差判定処理サブユニットは、100MHz での高速動作を確認している。

5. ま と め

本論文では、平面と包含球を用いた交差判定手法に基づく分散レイトレーシング法と、本手法を専用ハードウェアにより実現するレイトレーシング専用計算機システム RAPLAS について述べた。また、RAPLAS の 2 つの主要構成ユニットである平面-包含球交差判定処理サブユニットと光線-三角形交差判定処理サブユニットの設計、そして、FPGA 実装による動作確認を行った。設計を行なった各交差判定処理サブユニットは、近年高性能化が進む FPGA に搭載される高速な算術演算専用回路に着目し、専用の浮動小数点演算器を設計することにより、交差判定処理を高速化するだけでなく、FPGA に搭載されている多くの DSP コアを有効的に活用する設計となっている。

今後の課題として、交差判定処理ユニット全体の設計・実装、及び、システム全体での性能評価が挙げられる。

表 2　設計回路のハードウェアコスト及び性能

	Slice(使用率)	DSP(使用率)	動作周波数 (MHz)
平面-包含球交差判定サブユニット (4×4 光線)	8248(53.7%)	96(50.0%)	139.8
光線-三角形交差判定サブユニット	5282(34.4%)	70(36.5%)	167.4

文　献

[1] J. T. Kajiya: "The rendering equation", SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM Press, pp. 143–150 (1986).

[2] R. L. Cook, T. Porter and L. Carpenter: "Distributed ray tracing", SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM Press, pp. 137–145 (1984).

[3] T. Whitted: "An improved illumination model for shaded display", Commun. ACM, **23**, 6, pp. 343–349 (1980).

[4] Y. Kaeriyama, D. Zaitsu, K. Komatsu, K. Suzuki and T. N. Nobuyuki Ohba: "Hardware for a ray tracing technique using plane-sphere intersections", Symposium Short Paperes Proceedings of Eurographics Symposium on Parallel Graphics and Visualization(EGPGV' 06), pp. 9–12 (2006).

[5] Y. Kaeriyama, D. Zaitsu, K. Komatsu, K. Suzuki and T. N. Nobuyuki Ohba: "Ray tracing hardware system using plane-sphere intersections", FPL '06: Proceedings of International Conference on Field Programmable Logic and Applications, pp. 315–320 (2006).

[6] J. Schmittler, S. Woop, D. Wagner, W. J. Paul and P. Slusallek: "Realtime ray tracing of dynamic scenes on an fpga chip", HWWS '04: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, New York, NY, USA, ACM Press, pp. 95–106 (2004).

[7] Xilinx: "XtremeDSP". `http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex4/capab%ilities/xtremedsp.htm`.

[8] IEEE: "American National Standard - IEEE Standard for Binary Floating Point Arithmetic", American National Standards Institute, Inc. (1985).

[9] Intel: "IA-32 IntelR Architecture Software Developer's Manual". `http://developer.intel.com/design/intarch/pentium4/docs_pentium4_proc.htm#M%anuals`.

[10] R. Barzel Ed.: "Graphics Tools - the jgt editors' choice", A K Peters, Ltd (2005).

[11] T. Möller and B. Trumbore: "Fast, minimum storage ray-triangle intersection", journal of graphics tools, **2**, 1, pp. 21–28 (1997).