# レイトレーシング法を高速処理する専用並列レンダリング・マシン『熱視線』の要素プロセッサ・アーキテクチャ

## ——VLIW アーキテクチャおよび性能評価——

權 五 鳳[†] 村 田 誠 治[††*]
村 上 和 彰[††] 富 田 眞 治[†††]

筆者らは，レイトレーシング (ray tracing) 法を高速処理する専用並列レンダリング・マシン『熱視線』を開発している．『熱視線』は，マルチプロセッサ処理，マクロパイプライン処理，および，命令レベル並列処理の3レベルの並列処理を活用し高速化を図っている．すなわち，システム全体をマルチプロセッサ構成として画面分割型の並列処理を行い，次に個々の要素プロセッサを3ステージのマクロパイプライン構成として1本の光線に対する機能分割型のオーバラップ処理を行い，そして各ステージのプロセッサに VLIW (超長形式機械命令) アーキテクチャを採用している．本論文では『熱視線』の要素プロセッサ・アーキテクチャ，特にマクロパイプラインの各ステージ (物体探索ステージおよび交点計算ステージ) における命令レベル並列処理および VLIW アーキテクチャについて述べている．性能評価の結果，VLIW 処理により最低 1.91 倍～最高 2.31 倍の性能向上が可能であることを示している．

# Processing Element Architecture of a Parallel Rendering Machine for High Speed Ray-Tracing

## ——VLIW Architecture and Preliminary Performance Evaluation——

OUBONG GWUN,[†] SEIJI MURATA,[††*] KAZUAKI MURAKAMI[††] and SHINJI TOMITA[†††]

This paper introduces a parallel rendering machine for high-speed ray tracing, which machine exploits three levels of parallel processing: i) multiprocessing at the system level, ii) macropipelining at the processing-element level, and iii) instructional-level parallel processing at the stage level. This paper focuses on instruction-level parallel processing at the stage level. First, a ray tracing algorithm, called *3MPRT* (*3-stage macropipelined ray tracing*), and *3MPPE* (*3-stage macropipelined processing element*) architecture, which implements *3MPRT* directly, are presented to show how the ray-tracing task is implemented through three stages: *OS* (object search), *IC* (intersection calculation), and *SC* (shading calculation). This paper then describes the VLIW architectures for *OS* and *IC* stages. The VLIW architectures are evaluated through simulation, varing the benchmark scene characteristics and the object-space subdivision. From the simulation results, the paper concludes that the VLIW implementation is 1.91～2.31 faster than a non-VLIW implementation of *3MPRT*.

† 九州大学工学部情報工学科
Department of Information Sciences Faculty of Engineering, Kyushu University
†† 九州大学大学院総合理工学研究科情報システム学専攻
Department of Information Systems, Interdisciplinary Graduate School of Engineering Sciences, Kyushu University
††† 京都大学工学部情報工学科
Department of Information Science, Faculty of Engineering, Kyoto University
* 現在 富士通(株)
Presently with Fujitsu Co., Ltd.

## 1. はじめに

レイトレーシング (*ray tracing*: 光線追跡，視線探索) 法[26]は，反射，屈折，透過，影などの表現に優れ，現実感の高い3次元映像が生成可能なアルゴリズムである．しかしながら，計算量が非常に多いという短所があり，高速性を必要する分野においてはあまり実用化されていない．Whitted[26] によれば，レイトレーシング法の処理時間の75～95%は光線と物体 (オブジェクト) との交差判定に費やされている．したが

って，レイトレーシング法の高速化にあたっては，①交差判定回数の削減，および，②交差判定処理自体の高速化・高スループット化が重要であり，以下の種々のアプローチが考案されている[5]．

① 交差判定の回数を削減する：
- (a) 交差判定すべき光線の数を削減する：
  - i) Zバッファ併用法[25]
  - ii) 画素選択型光線追跡法[1]
  - iii) 寄与率法[21]
- (b) 1本の光線に関する交差判定回数を削減する：
  - i) 外接体 (bounding volume) 法[25]
  - ii) ライト・バッファ (light buffer) 法[19]
  - iii) オクトリー (octree) 分割法[16]
  - iv) ボクセル (voxel) 分割法 (空間等分割法)[14),22]

② 交差判定処理自体を高速化・高スループット化する：
- (a) レイトレーシング法のアルゴリズム自体に内在する空間的並列性を活用して，負荷分散型のマルチプロセッサ処理を行う：
  - i) オブジェクト空間分割法[13),24]
  - ii) 画面 (イメージ空間) 分割法：
    - A. データベース複写型[6)~8]
    - B. データベース共有型[15),17]
- (b) レイトレーシング法のアルゴリズム自体に内在する時間的並列性を活用して，機能分散型のマクロパイプライン処理を行う[15]
- (c) 交差判定計算に命令レベル並列処理を適用する[11),2]

われわれは，上記の高速化技法のうち以下のものを採用した，レイトレーシング専用並列レンダリング・マシン『熱視線』を開発している[3),4),10),18]．

- ①-a-iii 寄与率法：反射，屈折などの2次光線に対して，対応するピクセルの輝度値への寄与率があるしきい値以下になった場合，光線追跡を打ち切る．
- ①-b-iv ボクセル分割法：オブジェクト空間をボクセルに等分割し，ボクセルごとにその内部に存在する物体の情報を付加する．交差判定回数が減少する反面ボクセルのトラバース処理が増加するが，これは3DDDA (3 Dimensional Digital Differential Analyzer)[22] により高速化する．
- ②-a-ii-B マルチプロセッサ処理：画面をピクセル単位で分割し，個々のピクセルに関する処理を各要素プロセッサに割り当てる．各要素プロセッサは，他の要素プロセッサとはまったく独立に処理を進めることができる．ただし，データベースは，全要素プロセッサで共有する．
- ②-b マクロパイプライン処理：要素プロセッサを3ステージにマクロパイプライン化し，個々の要素プロセッサ自身の処理能力（スループット）も高める．
- ②-c 命令レベル並列処理：マクロパイプラインの各ステージを担当するプロセッサに，VLIW (Very Long Instruction Word) アーキテクチャを採用する．

本論文では，『熱視線』の要素プロセッサ・アーキテクチャ，特にマクロパイプラインの各ステージにおける命令レベル並列処理および VLIW アーキテクチャについて述べる．まず第2章で，われわれが開発した3MPRT アルゴリズムと呼ぶレイトレーシング法，および，『熱視線』の要素プロセッサで3MPRT アルゴリズムを直接実装した 3MPPE と呼ぶプロセッシング・エレメントについて概説する．第3章および第4章では，3MPPE 内の物体探索プロセッサおよび交点計算プロセッサにおける命令レベル並列処理および VLIW アーキテクチャについて詳述する．第5章で，その性能をシミュレーションにより評価する．

## 2. 要素プロセッサ・アーキテクチャ

### 2.1 3MPRT アルゴリズム

レイトレーシング法の個々の光線処理に関するタスクは，まず交差判定と輝度計算の2個のサブタスクに分割できる．『熱視線』では，1本の光線に関する交差判定回数を削減するために，ボクセル分割法[14),22]を採用している．これにより，交差判定サブタスクは，さらに物体探索と交点計算の2個のサブタスクに分割可能となる．すなわち，1本の光線の処理は，①物体探索，②交点計算，③輝度計算の計3個のオーバラップ可能なサブタスクに分割可能である．そこで，**図1**に示すように，各サブタスクを1ステージとし，3ステージのマクロパイプラインにより1本の光線を処理する 3MPRT (3-stage MacroPipelined Ray-Tracing) アルゴリズムを開発した[4),18]．本アルゴリズムでは，光線は基本的に①物体探索→②交点計算→③輝度計算へと順にステージを進めながら個別に処理される．ステージ間には FIFO バッファが設けられ

ており，各ステージの処理は独立したプログラムにより非同期に行われる．

3 MPRT アルゴリズムにおける各光線の処理過程は，おおむね以下のようになる．

① 物体探索ステージ：光線が物体探索ステージに入力されると，光線の進行方向に沿ってオブジェクト空間をトラバースする．
- 注目しているボクセル内にプリミティブが存在する場合，当該ボクセルの識別子を光線に添付して交点計算ステージへ進ませる．
- そうでない場合は，次のボクセルへ進む．

② 交点計算ステージ：ボクセル識別子を用いてボクセル内の全プリミティブの形状データを読み出し，光線とプリミティブとの交点計算を行う．
- 1次光線（Initial Ray）ないし分岐光線（Secondary Ray）があるプリミティブと交差する場合，最も視点に近いプリミティブの識別子を光線に添付して輝度計算ステージに進ませる．
- 影光線（Light Ray）があるプリミティブと交差する場合，その影光線は消滅する．
- いずれの光線もプリミティブと交差しなかった場合は，戻り光線（Return Ray）として物体探索ステージへ戻される．

③ 輝度計算ステージ：プリミティブ識別子を用いてプリミティブの輝度データを読み出し，輝度計算を行う．
- 影光線を生成して，物体探索ステージへ進ませる．
- さらに，プリミティブの種類によっては，反射および屈折光線といった分岐光線を生成して，物体探索ステージへ進ませる．

そして，元の光線は消滅する．

### 2.2 『熱視線』の全体構成

図2に『熱視線』の全体構成を示す．フロントエンドにホスト・プロセッサを，また，バックエンドにフレーム・バッファを備える．3MPRT アルゴリズムを

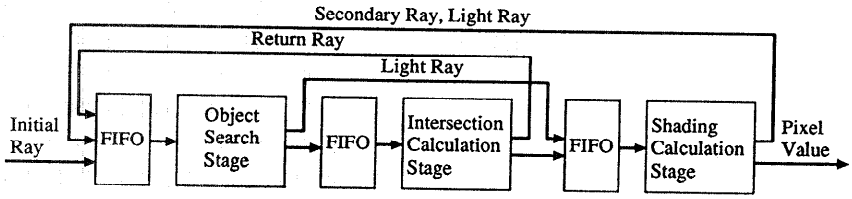
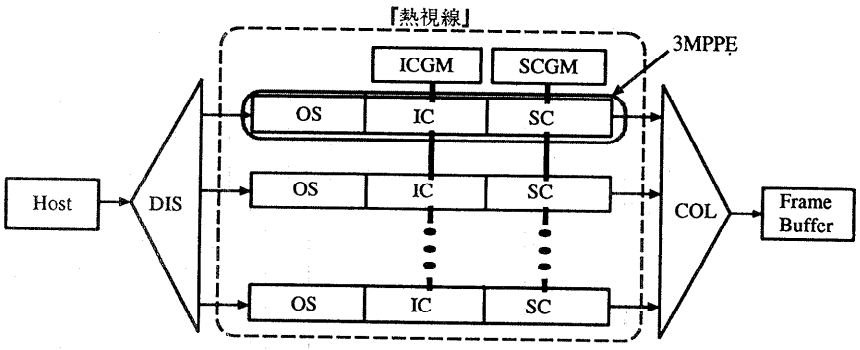



図 1 マクロパイプラインの論理構造
Fig. 1 Logical structure of macropipeline.




3MPPE: 3-stage Macro Pipelined Processing Element
COL: Pixel Value Collector
DIS: Load Distributer
IC: Intersection Calculation Processor
ICGM: Intersection Calculation Global Memory
OS: Object Search Processor
SC: Shading Calculation Processor
SCGM: Shading Calculation Global Memory

図 2 『熱視線』の全体構成
Fig. 2 System configuration.

直接実装したプロセッシング・エレメント（3MPPE: 3-stage MacroPipelined Processing Element）を要素プロセッサとするメモリ共有型マルチプロセッサである．構成要素は，以下の通りである．

- 負荷分配器（DIS）：1次光線生成に必要なデータ（負荷）を3MPPEに分配する．また，ホストによる動的負荷分散・均衡化を可能とするために，各3MPPEの負荷を監視し，それをホストに伝播する機能を有する．
- ピクセル輝度値コレクタ（COL）：各3MPPEの光線処理結果（ピクセル輝度値）を集めて，随時フレーム・バッファに書き込む．
- グローバル・メモリ：光線処理過程で用いる4種類のデータベース（ボクセル・データ，プリミティブ・リスト，プリミティブ形状データ，および，プリミティブ輝度データ）のうち，交点計算ステージおよび輝度計算ステージに占有される3種類のデータベースを，それぞれステージ固有のグローバル・メモリに格納する[10]．
  - —交点計算用グローバル・メモリ（ICGM）：プリミティブ・リストおよびプリミティブ形状データを格納する．全3MPPEの交点計算プロセッサ（IC）に共有される．
  - —輝度計算用グローバル・メモリ（SCGM）：プリミティブ輝度データを格納する．全3MPPEの輝度計算プロセッサ（SC）に共有される．
- プロセッシング・エレメント（3MPPE）：次節参照．

### 2.3 プロセッシング・エレメント

**図3**に3MPPEの構成を示す．3MPRTアルゴリズムを直接実装するため，3ステージのマクロパイプライン構成を採っている．個々のステージは，専用のプログラムおよびプログラム・カウンタを有する独立したプロセッサである．

各プロセッサは，以下の構成方式を採る．

- 物体探索プロセッサ（OS）：整数演算ユニット2個および浮動小数点演算ユニット2個を有するVLIWプロセッサ．
- 交点計算プロセッサ（IC）：整数演算ユニット1個および浮動小数点演算ユニット3個を有するVLIWプロセッサ．
- 輝度計算プロセッサ（SC）：汎用機能ユニット1個を有する逐次プロセッサ．

上記の各プロセッサは，FIFOバッファを内蔵する通信ユニット（CMU）を介してリング上に接続される．デッドロック・フリー，高スループット，低レイテンシといった要件を満たすために，文献9）で提案されている“ワームホール・ルーティング＋構造化バッファ・プール”方式を採用している．

全3MPPEは，グローバル・メモリを共有しているが，このとき，グローバル・メモリへのアクセス時間が問題となる．レイトレーシング法の光線処理においても，データ参照の局所性があることは知られている[17]．よって，キャッシュ・メモリを導入することにより，実効メモリ・アクセス時間の低減が期待できる．しかも，光線処理過程ではすべてのデータベース

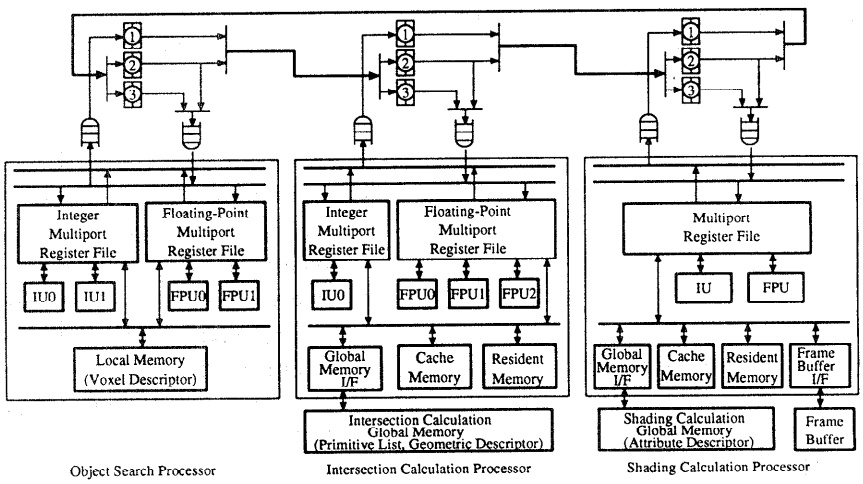



図3 マクロパイプライン物理構造
Fig. 3 Physical structure of macropipeline.

が read-only なので，キャッシュの一貫性は保証する必要もない．

くわえて，光線処理においては，他のデータに比べて参照頻度の高いデータが存在する．これは，映像として表示されるのは視点から見える物体のプリミティブであることから，容易に想像できる．このような参照頻度の高いデータは，キャッシングされると同時にリプレース・アウトされないようにする必要がある．つまり，キャッシュ・メモリ内に常駐させたほうがよい．

以上の特徴から，交点計算プロセッサ (IC) および輝度計算プロセッサ (SC) それぞれに対しては，次のような，2階層3種類のメモリからなる構成を採った．

- ローカル・メモリ：第1層メモリ．
  —レジデント・メモリ：参照頻度の高いデータの完全なコピーを各 3 MPPE のレジデント・メモリに格納する．これらのデータのコピーは，グローバル・メモリに格納する必要はない．
  —キャッシュ・メモリ：グローバル・メモリから必要なデータのコピーを適宜キャッシングする．
- グローバル・メモリ：第2層メモリ．
  レジデント・メモリに格納すべきデータ以外をすべて格納する．

## 3. 物体探索プロセッサにおける命令レベル並列処理

### 3.1 3 DDDA アルゴリズム

空間等分割法を採用しているので，交点計算を行う前に光線がたどるボクセルを決定する必要がある．これには Synder の 3 DDDA (3 Dimensional Digital Differential Analyzer) アルゴリズムを用いる[22]．以下にアルゴリズムを示す．

```
while(tin < tbo){
 if(tx < ty){
  if(tx < tz){
   tin=tx; tx=tx+adx; ix=ix+inx;}
  else{
   tin=tz; tz=tz+adz; iz=iz+inz;}
  }
 else{
  if(ty<tz){
   tin=ty; ty=ty+ady; iy=iy+iny;}
  else{
   tin=tz; tz=tz+adz; iz=iz+inz;}
  }
 {メモリ・アクセス}
}
```

上記の **while** ループはプリミティブを含むボクセルが見つかるまで，あるいは，光線が物体空間の終わりに到達するまで繰り返される．たとえば，物体空間を一辺あたり 50 個のボクセルに分割すると，最大 87 回 ($\approx\sqrt{3}\times 50$) 上記ループを繰り返す．逐次処理では，上記ループの計算に 23 ステップ要するため，ボクセル探索のステップ数は最大約 2000 になる．したがって，物体探索ステージの高速化は上記ループのステップ数を削減することにかかっている．

上記のループは，以下の各演算よりなる．

① 浮動小数点比較 (if(tx<ty) 等)
② 浮動小数点 move (tin=tx 等)
③ 浮動小数点加算 (tx=tx+adx 等)
④ 整数加算 (ix=ix+inx 等)
⑤ メモリ・アドレス計算のための整数加算および乗算
⑥ メモリ・アクセス

このうち②，③，④は互いにデータの依存関係がなく，VLIW 化して複数の機能ユニットを用いて同時に実行できる．また，メモリアクセス関連命令 (⑤，⑥) とオーバラップ処理可能である．上記ループを整数演算ユニット2本と浮動小数点演算ユニット2本でソフトウェア・パイプライン化して処理する．これにより逐次処理では 23 ステップ要するのが，VLIW では 14 ステップに，ソフトウェア・パイプライン化によって 11 ステップに減少し，2倍の性能向上が可能である．

### 3.2 物体探索プロセッサの構成

物体探索プロセッサは，以下の要素から構成される VLIW プロセッサである (図3参照)．

① 演算ユニット：整数演算ユニット (IU) を2本，浮動小数点演算ユニット (FPU) 2本を備える．
② マルチポート・レジスタファイル：整数データ用 (IRF) および浮動小数点データ用 (FPRF) に2組の独立したレジスタ・ファイルを設ける．それぞれ 32 ビット長×128 語の書き込みポート 2，読出しポート5のマルチポート・レジスタファイルである．
③ メモリユニット：2 MB の SRAM 構成でボクセルデータの完全なコピーを格納する．

### 3.3 処理過程

物体探索ステージの処理過程を**図4**に示す．処理の主な内訳は次の通りである[10]．

- 光線データの FIFO バッファからのロード：25.
- 初期化：60（初期光線のみ）．
- ボクセル探索：11×I（I：ボクセル探索の繰り返し数）．
- 光線データの FIFO バッファへのストア：25.

2本の光線データに対するFIFOから/へのロード/ストア処理をオーバラップすることにより，ロード/ストア処理に要する時間を隠蔽する．

## 4. 交点計算プロセッサにおける命令レベル並列処理

### 4.1 交点計算アルゴリズム

レイトレーシング法における形状モデルとして，三角ポリゴン，2次曲面，一般陰関数，パラメトリック・パッチなどがあるが，三角ポリゴンおよび2次曲面がよく用いられる．以下三角ポリゴンと2次曲面の処理について述べる．

① 三角ポリゴン：

交点計算は三角ポリゴンの3点を含む平面式の計算，この平面と光線との交点計算，交点が三角ポリゴンの内部に存在するかどうかの判定からなる．平面式は一般に

$$\alpha x_i+\beta y_i+\gamma z_i-1=0$$

で表現される．平面式の係数 $\alpha$，$\beta$，$\gamma$ を求める問題は，三つの頂点 $(x_1, y_1, z_1)$，$(x_2, y_2, z_2)$，$(x_3, y_3, z_3)$ より表される 3×3 逆行列の式，

$$\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix}=\begin{bmatrix}x_1 & y_1 & z_1\\ x_2 & y_2 & z_2\\ x_3 & y_3 & z_3\end{bmatrix}^{-1}\begin{bmatrix}1\\ 1\\ 1\end{bmatrix}$$

を解く問題に帰着される．この行列の計算は対称性のある浮動小数点演算が中心であり，$\alpha, \beta, \gamma$ の間にはデータ依存性がなく，それぞれ並列に求めることが可能である．また，光線との交点はそれぞれ，x軸，y軸，z軸に対応して独立に求めることが可能である．

② 2次曲面：

処理の中心は，2次曲面を表現する式より交点を求める計算である．2次曲面を表す式および光線を表す式は，次のようになる．

2次曲面：$Ax_i^2+By_i^2+Cz_i^2+Dx_iy_i+Ey_iz_i+Fz_ix_i+Gx_i+Hy_i+Iz_i+J=0$

光線：$\begin{bmatrix}x_i\\ y_i\\ z_i\end{bmatrix}=t\begin{bmatrix}U_x\\ U_y\\ U_z\end{bmatrix}+\begin{bmatrix}V_x\\ V_y\\ V_z\end{bmatrix}$

上記2式より，2次曲面と光線との交点を求める式はtに関する2次方程式，

$$at^2+bt+c=0$$
$$a=AU_x^2+BU_y^2+CU_z^2+\cdots$$
$$b=2AU_xV_x+2BU_yV_y+2CU_zV_z+\cdots$$
$$c=AV_x^2+BV_y^2+CV_z^2+\cdots$$

のtの解を求める計算に帰着できる．処理の中心は，2次方程式の係数 a，b，c を2次曲面の係数と光線ベクトルから求める計算である．これは，同一形態の浮動小数点演算が中心であり，2次方程式の各係数は独立に求めることが可能である．

三角ポリゴンおよび2次曲面いずれのプリミティブも，係数計算に関しては，加算あるいは乗算を中心とした同一形態の浮動小数点演算が多数個（ほぼ3の倍数個）並列処理可能である．また，交点計算および法線ベクトル計算に関しては，x，y，z の各軸に関する浮動小数点演算を同時に行うことが可能である．よって，交点計算ステージにおいては，浮動小数点演算ユニットを3本備える．これにより，ほぼ3倍の性能向上が可能となる．

### 4.2 交点計算プロセッサの構成

交点計算プロセッサは，以下の要素から構成されるVLIW プロセッサである（図3参照）．

① 演算ユニット：整数演算ユニット（IU）を1本，浮動小数点演算ユニット（FPU）を3本備える．

② マルチポート・レジスタファイル：整数データ用（IRF）および浮動小数点データ用（FPRF）に2組の独立したレジスタ・ファイルを設ける．

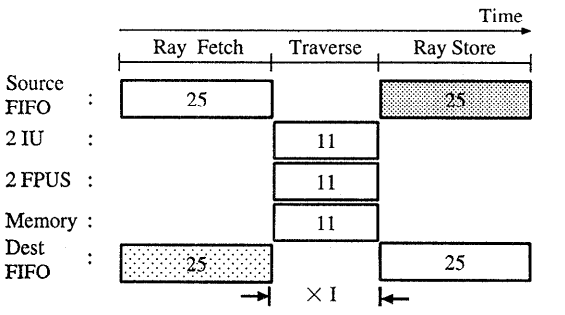



**図4** 物体探索プロセッサの処理
Fig. 4 Object search sequence.

FPRF は32ビット長×128語の書込みポート3，読出しポート7のマルチポート・レジスタファイルである．

③ メモリユニット：プリミティブ形状データの中で参照頻度の高いデータを格納するレジデント・メモリ（1 MB, SRAM 構成），および，それ以外のデータを適宜格納するキャッシュ・メモリ（1 MB, SRAM 構成）を設ける．

### 4.3 処理過程

交点計算ステージの処理過程を**図5**に示す．処理の主な内訳は次の通りである．

- 光線データの FIFO バッファからのロード：25.
- プリミティブ形状データのメモリからのロード：
  ① 三角ポリゴン：10×T.
  ② 2次曲面：11×(P−T).
  (P：ボクセル内のプリミティブの数，T：ボクセル内の三角ポリゴンの数)
- 交点計算
  ① 三角ポリゴン：40×T.
  ② 2次曲面：50×(P−T)（ただし，2次曲面と光線の交点を求める2次方程式が異なる2つの実解をもつとき）.
- 光線データの FIFO バッファへのストア：25.

物体探索プロセッサの処理と同様，2本の光線の FIFO から/へのロード/ストアをオーバラップ処理する．さらに，ボクセル内のm番目のプリミティブの処理と (m+1) 番目のプリミティブ ($1 \leq m \leq P-1$) の形状データのフェッチに関しても，オーバラップ処理する．

## 5. 性能評価

3 MPPE で採用した VLIW アーキテクチャについて，ソフトウェア・シミュレータを用いて性能評価を行った．

### 5.1 評価方法

本シミュレータは 3 MPRT アルゴリズムを忠実に実行し，3 MPPE の動作を機能レベルで近似的にモデル化している．評価モデルとしては VLIW アーキテクチャの効果を測定するため，次の三つを用いた．

- *MACRO・VLIW* (S)：マクロパイプライン・アーキテクチャおよび VLIW アーキテクチャともに採用していない．一般的な逐次プロセッサ・モデル．レジスタは，128個．
- *MACRO・VLIW* (M)：マクロパイプライン・アーキテクチャを採用した3 MPPE．ただし，物体探索プロセッサおよび交点計算プロセッサは VLIW アーキテクチャではなく，一般的な逐次プロセッサである．つまり，三つのプロセッサは同一構造を採る．各々，128 個のレジスタを備える．
- MACRO・VLIW (V)：マクロパイプライン・アーキテクチャおよび VLIW アーキテクチャを採用した 3 MPPE．OS プロセッサと IC プロセッサが VLIW アーキテクチャである．各々，128 個のレジスタを備える．

ここでは，VLIW アーキテクチャの効果について着目しているので，それ以外の要因に関しては上記の三つのモデルとも以下の理想的な条件を仮定した．すなわち，データ・キャッシュはヒット率100% とした．また，評価モデルMとVの通信モデルは，送信 FIFO バッファおよび受信 FIFO バッファを無限長にしてオーバーフローが生じないようにし，送信FIFOバッファの出口から受信FIFOのバッファの入口までのネットワーク・レイテンシを零とした．

コードには，物体探索ステージでは Synder[22]，交点計算ステージでは Kuchkuda[23] を参照してハンドコンパイルした．Kuchkuda のコードは逐次処理に最適化されたものである．ベンチマーク・シーンとしては，Haines[20] が提案している6シーンのうち**表1**に示した五つを用いた．ピクセル数は512×512である．

各ベンチマーク・シーンおよび各評価モデルに対して，オブジェクト空間の分割数，すなわち，次元あたりの分割数を 20, 50, 100 と変化させて，以下の項目

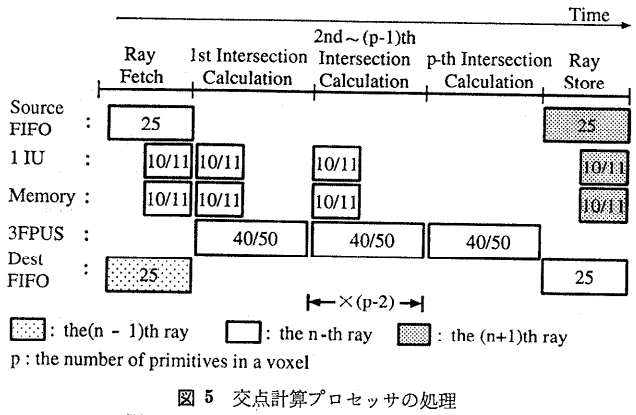



**図5** 交点計算プロセッサの処理
Fig. 5 Intersection calculation sequence.

を測定した.

- モデルVの映像生成時間(**表2**参照):評価モデルVに対してホストにおける前処理時間(空間分割処理,等)を除いた正味のレイトレーシング実行時間.さらに,各プロセッサにおける以下の所要時間の内訳を調査した.
  - —処理時間:映像生成時間全体から,以下の通信時間および待ち時間を除いた時間.
  - —通信時間:送信FIFOバッファへの書込み,および,受信FIFOバッファからの読出しに要した時間.
  - —待ち時間:受信FIFOバッファが空のために生じた待ち時間.
- 速度向上率(表3参照):評価モデルS, M, Vの正味のレイトレーシング実行時間と速度向上率.
- 命令レベル並列度(表4参照):評価モデルVのOSプロセッサおよび,ICプロセッサにおける平均命令レベル並列度,および,各演算ユニットの使用率.

**表1** ベンチマーク・シーン
Table 1 Benchmark scenes.

| ベンチマーク・シーン | | balls | mount | rings | tetra | tree |
|---|---|---|---|---|---|---|
| プリミティブ数 | | 7382 | 8196 | 8401 | 4096 | 8191 |
| | ポリゴン | 1 | 8192 | 1 | 4096 | 1 |
| | 球 | 7381 | 4 | 4200 | 0 | 4095 |
| | 円錐 | 0 | 0 | 0 | 0 | 4095 |
| | 円柱 | 0 | 0 | 4200 | 0 | 0 |
| 光源数 | | 3 | 1 | 3 | 1 | 7 |
| 反射面 | | 有 | 有 | 有 | 無 | 無 |
| 透過面 | | 無 | 有 | 無 | 無 | 無 |

### 5.2 評価結果および考察

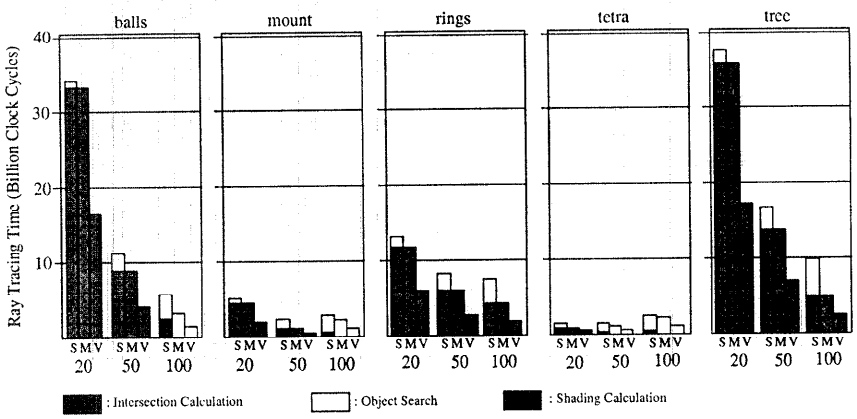
表2-4と**図6**に測定結果を示した.

まず,表2に,評価モデルVに対して,また図6に,評価モデルS, MおよびVに対して,次元あたりの分割数を20, 50, 100と変化させた場合の映像生成時間を示す.

図6のモデルSの棒グラフの模様は,物体探索,交点計算,および,輝度計算の各処理時間を示している.また,モデルMおよびモデルVのほうは,マクロパイプライン処理およびマクロパイプライン+VLIW処理においても最も処理時間の大きかったステージを示している.これから,以下のことがわかる.

① 映像生成時間は,空間分割数とベンチマーク・シーンに大きく依存する.映像生成時間が最小になる空間分割数は,ベンチマーク・シーンにより異なる.

- mountおよびtetraでは,分割数を20→50と増加させると性能が向上するが,さらに50→100と増加させると逆に低下する.これから,mountおよびtetraの場合,分割数が20と100の間に最適な分割数が存在するといえる.
- balls, rings,および,treeでは,分割数を20→50→100と増加させるに従って性能が向上している.この場合,最適な空間分割数は100以上




S: $\overline{MACRO} \cdot VLIW$ $M: MACRO \cdot \overline{VLIW}$ $V: MACRO \cdot VLIW$
20,50,100: the number of grids per dimension

**図6** 映像生成時間[単位:1,000,000,000クロック・サイクル]
Fig. 6 Ray tracing time.

表 2 映像生成時間［モデル V，単位：1,000,000 クロック・サイクル］
Table 2 Ray tracing time.

| ベンチマーク・シーン | | balls | | | mount | | | rings | | | tetra | | | tree | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 次元当り分割数 | | 20 | 50 | 100 | 20 | 50 | 100 | 20 | 50 | 100 | 20 | 50 | 100 | 20 | 50 | 100 |
| 全処理時間 | | 16,846 | 4,581 | 1,887 | 2,329 | 892 | 1,130 | 6,055 | 3,064 | 2,263 | 832 | 584 | 1,114 | 17,357 | 7,337 | 3,060 |
| [時間比]† | | [100:27:11] | | | [100:38:48] | | | [100:50:37] | | | [74:52:100] | | | [100:41:17] | | |
| 物体探索 | 処理時間 | 442 | 868 | 1,593 | 247 | 494 | 914 | 357 | 681 | 1,139 | 215 | 539 | 1,070 | 571 | 1,146 | 2,016 |
| プロセッサ | [割合]‡ | [3%] | [19%] | [84%] | [11%] | [55%] | [81%] | [6%] | [22%] | [50%] | [26%] | [92%] | [96%] | [3%] | [16%] | [66%] |
| (OS) | 通信時間 | 203 | 225 | 275 | 104 | 114 | 143 | 251 | 278 | 310 | 41 | 44 | 43 | 237 | 237 | 237 |
| | [割合]‡ | [1%] | [5%] | [15%] | [4%] | [13%] | [13%] | [4%] | [9%] | [14%] | [5%] | [8%] | [4%] | [1%] | [3%] | [8%] |
| | 待ち時間 | 16,200 | 3,487 | 18 | 1,976 | 283 | 73 | 5,445 | 2,104 | 813 | 575 | 0.06 | 0.07 | 17,049 | 5,989 | 805 |
| | [割合]‡ | [96%] | [76%] | [1%] | [85%] | [32%] | [6%] | [90%] | [69%] | [36%] | [69%] | [0%] | [0%] | [96%] | [81%] | [26%] |
| 交点計算 | 処理時間 | 16,569 | 4,209 | 1,167 | 2,058 | 637 | 243 | 5,762 | 2,703 | 1,800 | 622 | 222 | 142 | 17,514 | 6,895 | 2,412 |
| プロセッサ | [割合]‡ | [98%] | [91%] | [62%] | [88%] | [71%] | [21%] | [95%] | [88%] | [80%] | [74%] | [38%] | [13%] | [98%] | [94%] | [79%] |
| (IC) | 通信時間 | 186 | 207 | 257 | 97 | 106 | 136 | 236 | 264 | 296 | 33 | 37 | 36 | 211 | 211 | 211 |
| | [割合]‡ | [1%] | [5%] | [11%] | [4%] | [12%] | [12%] | [4%] | [9%] | [13%] | [4%] | [6%] | [3%] | [1%] | [3%] | [7%] |
| | 待ち時間 | 90 | 164 | 461 | 174 | 147 | 751 | 56 | 96 | 166 | 175 | 237 | 936 | 131 | 230 | 436 |
| | [割合]‡ | [1%] | [4%] | [24%] | [8%] | [17%] | [67%] | [1%] | [3%] | [7%] | [22%] | [56%] | [84%] | [1%] | [3%] | [14%] |
| 輝度計算 | 処理時間 | 176 | 180 | 178 | 60 | 64 | 55 | 120 | 122 | 124 | 17 | 31 | 23 | 83 | 81 | 80 |
| プロセッサ | [割合]‡ | [1%] | [4%] | [9%] | [3%] | [7%] | [5%] | [2%] | [4%] | [5%] | [2%] | [5%] | [2%] | [0%] | [1%] | [3%] |
| (SC) | 通信時間 | 44 | 45 | 45 | 24 | 24 | 24 | 36 | 36 | 36 | 10 | 10 | 10 | 33 | 33 | 33 |
| | [割合]‡ | [0%] | [1%] | [2%] | [1%] | [3%] | [2%] | [1%] | [1%] | [2%] | [1%] | [2%] | [1%] | [0%] | [0%] | [1%] |
| | 待ち時間 | 16,625 | 4,355 | 1,663 | 2,244 | 803 | 1,051 | 5,898 | 2,905 | 2,103 | 804 | 543 | 1,081 | 17,740 | 7,222 | 2,945 |
| | [割合]‡ | [99%] | [95%] | [89%] | [96%] | [90%] | [93%] | [97%] | [95%] | [93%] | [97%] | [93%] | [97%] | [100%] | [99%] | [96%] |
| OS:IC:SC 負荷比§ | | 2:76:1 | 4:20:1 | 8:6:1 | 4:25:1 | 6:8:1 | 13:4:1 | 3:37:1 | 6:18:1 | 9:13:1 | 9:24:1 | 14:6:1 | 33:5:1 | 6:152:1 | 12:62:1 | 19:23:1 |

†: 分割数 20，50，100 の中で最も映像生成時間の大きいものを 100 とした場合の時間比

‡: 各プロセッサの所要時間内に占める割合

§: 負荷=処理時間+通信時間

かもしれないし，あるいは，50 と 100 との間に存在するかもしれない．

② 評価モデルVで映像生成時間が最小となるのは，物体探索，交点計算，輝度計算の各プロセッサの負荷（OS：IC：SC 負荷比）が最も均衡している空間分割数においてである．つまり，空間分割数の増加に伴って増加する物体探索プロセッサの負荷と，逆に減少する交点計算プロセッサの負荷とが交差する点である．

③ 上述の通り，ボクセル分割法では，最適な空間分割数を決定することが重要である．理論的な方法としては，Cleary らが最適な空間分割数 $N_m$ を以下のように求めることを提案している[12]．

$$N_m=\sqrt{N\frac{\rho\chi\left(t_i+\frac{t_r}{2}\right)+\alpha t_i}{2\rho t_x}}$$

ここで，$N$：物体数，$\rho$：光線の平均探索距離，$\chi$：物体空間内のプリミティブの平均外周，$t_i$：最初の物体との交点計算に費す時間，$t_r$：2番目以降の物体との交点計算に費す時間，$t_x$：次ボクセルの探索時間，$\alpha$：物体の平均面積，である．

しかしながら，これらの変数の値を求めるのは容易ではない．別の方法として，画素を粗くしたレイトレーシングを前処理として施し，その結果に基づいて空間分割数を決定することが可能である．たとえば，1000×1000 の画素数でシーンを生成したい場合に，前処理で 100×100 の画素数で空間分割数を3通り程度変えてレイトレーシングを行うとする．このとき，前処理に要する時間は，本処理に要する時間の 1/100×3=3% 程度で済む．

**表3**に，モデルSとモデルMに対するモデルVの速度向上率を，そして表4に物体探索プロセッサおよび交点計算プロセッサでの VLIW 処理の効果を示す．これから，以下のことがわかる．

- 評価モデルVの評価モデルSに対する性能向上率は最低 2.08 倍～最高 3.87 倍である．各ベンチマーク・シーンにおいて評価モデルVの性能が最高となるのは，映像生成時間が最小となる空間分割数のときである．
  - —balls，rings，および，tree の場合，分割数 100 のとき両評価モデルともに映像生成時間が最小となり，性能向上率もそれぞれ 3.87 倍，3.41 倍，および，3.48 倍と最高となる．
  - —mount および tetra の場合，分割数 50 のとき両評価モデルともに映像生成時間が最小となり，性能向上率もそれぞれ 3.48 倍および 3.33 倍と最高となる．
- 評価モデルVの評価モデルMに対する性能向上率は最低 1.91 倍～最高 2.31 倍である．
  - —balls および tetra の場合，分割数 100 のとき性能向上率はそれぞれ 2.31 倍および 2.27 倍と最高となるが，映像生成時間は必ずしも最小のときではない．
  - —mount，rings および tree の場合，分割数が 20 のとき性能向上率はそれぞれ 2.20 倍，2.06 倍，および 2.05 倍と最高となるが，映像生成時間は必ずしも最小のときではない．
  - —物体探索プロセッサにおける，評価モデルMに対する評価モデルVの速度向上率は，処理時間に関しては 2.21～2.29 倍，通信時間に関しては 1.85～2.00 倍である．
  - —交点計算プロセッサにおける，評価モデルMに対する評価モデルVの速度向上率は，処理時間に関しては 2.00～2.31 倍，通信時間に関しては 1.86～2.03 倍である．
  - —通信時間に関して，評価モデルMに対する評価モデルVの速度向上率が2倍程度となっているのは，評価モデルMでは FIFO バッファの読み書きを逐次的に行っていたが，評価モデルVでは並列に行っているからである．
- 評価モデルVの評価モデルMに対する性能向上率が示すように，VLIW 処理は空間分割数に依存しない．
- 各プロセッサの平均命令並列度（処理時間に対してのみ計算）は，最大演算並列度を4としたとき，物体探索プロセッサは 2.27～2.29 と一定である．交点計算プロセッサでは若干ばらつきがある．ベンチマーク・シーン balls が 2.01～2.02，mount と tetra が 2.27～2.31，rings と tree が 2.06～2.11 である．交点計算プロセッサにおいて平均命令並列度で若干の差があるのは，各ベンチマーク・シーン内に占めるプリミティブの違いによる．balls，rings および tree が，球，円錐そして円柱といった2次曲面で，また，mount と tetra が三角ポリゴンで占められていることから，2次曲面に比べて三角ポリゴンが高い並列度を示すことがわかる．
- 評価モデルVにおいて各プロセッサの使用率（処

表 3　速度向上率
Table 3　Speedup ratio.

| ベンチマーク・シーン | | balls | | | mount | | | rings | | | tetra | | | tree | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 次元当り分割数 | | 20 | 50 | 100 | 20 | 50 | 100 | 20 | 50 | 100 | 20 | 50 | 100 | 20 | 50 | 100 |
| 映像生成時間† | S | 35,173 | 11,526 | 7,308 | 5,771 | 3,109 | 3,288 | 13,930 | 8,420 | 7,726 | 2,096 | 1,951 | 2,976 | 38,552 | 17,839 | 10,651 |
| | M | 33,586 | 8,982 | 4,375 | 4,918 | 1,800 | 2,453 | 12,484 | 6,214 | 4,471 | 1,673 | 1,321 | 2,540 | 36,768 | 14,915 | 5,851 |
| | V | 16,846 | 4,581 | 1,887 | 2,339 | 892 | 1,130 | 6,055 | 3,064 | 2,263 | 832 | 584 | 1,114 | 17,857 | 7,337 | 3,060 |
| 速度向上率 | S/M | 1.04 | 1.28 | 1.67 | 1.16 | 1.82 | 1.34 | 1.11 | 1.35 | 1.72 | 1.25 | 1.47 | 1.17 | 1.04 | 1.19 | 1.82 |
| | M/V | 1.99 | 1.96 | 2.31 | 2.10 | 2.01 | 2.16 | 2.06 | 2.02 | 1.97 | 2.0 | 2.26 | 2.27 | 2.05 | 2.03 | 1.91 |
| | S/V | 2.08 | 2.51 | 3.87 | 2.46 | 3.48 | 2.90 | 2.30 | 2.74 | 3.41 | 2.51 | 3.33 | 2.67 | 2.15 | 2.43 | 3.48 |

†単位:*1,000,000*クロック・サイクル

表 4　VLIW 処理効果
Table 4　VLIW processing effects.

| ベンチマーク・シーン | | | balls | | | mount | | | rings | | | tetra | | | tree | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 次元当り分割数 | | | 20 | 50 | 100 | 20 | 50 | 100 | 20 | 50 | 100 | 20 | 50 | 100 | 20 | 50 | 100 |
| 物体探索プロセッサ(OS) | 処理時間† | M | 994 | 1,975 | 3,641 | 562 | 1,129 | 2,095 | 807 | 1,552 | 2,605 | 493 | 1,237 | 2,459 | 1,301 | 2,542 | 4,627 |
| | | V | 442 | 868 | 1,593 | 247 | 494 | 914 | 357 | 681 | 1,139 | 215 | 539 | 1,070 | 571 | 1,146 | 2,016 |
| | | M/V | 2.24 | 2.27 | 2.28 | 2.27 | 2.28 | 2.29 | 2.26 | 2.27 | 2.28 | 2.29 | 2.29 | 2.29 | 2.27 | 2.21 | 2.29 |
| | 通信時間† | M | 406 | 449 | 549 | 207 | 225 | 284 | 501 | 554 | 619 | 76 | 83 | 81 | 470 | 471 | 470 |
| | | V | 203 | 225 | 275 | 104 | 114 | 143 | 251 | 278 | 310 | 41 | 44 | 43 | 237 | 237 | 237 |
| | | M/V | 2.00 | 1.99 | 1.99 | 1.99 | 1.97 | 1.98 | 1.99 | 1.99 | 1.99 | 1.85 | 1.88 | 1.88 | 1.98 | 1.98 | 1.98 |
| | 平均命令並列度‡ | V | 2.27 | 2.28 | 2.29 | 2.28 | 2.29 | 2.29 | 2.28 | 2.29 | 2.29 | 2.29 | 2.29 | 2.29 | 2.29 | 2.29 | 2.29 |
| | 各演算器使用率§ | V | 0.56 | 0.57 | 0.57 | 0.57 | 0.57 | 0.57 | 0.57 | 0.57 | 0.57 | 0.57 | 0.57 | 0.57 | 0.57 | 0.57 | 0.57 |
| 交点計算プロセッサ(IC) | 処理時間† | M | 33,148 | 8,429 | 2,346 | 4,651 | 1,440 | 544 | 11,967 | 5,604 | 3,725 | 1,429 | 513 | 329 | 36,217 | 14,264 | 4,995 |
| | | V | 16,569 | 4,209 | 1,167 | 2,058 | 637 | 243 | 5762 | 2703 | 1800 | 622 | 222 | 142 | 17514 | 6895 | 2412 |
| | | M/V | 2.0 | 2.0 | 2.0 | 2.25 | 2.26 | 2.23 | 2.07 | 2.07 | 2.06 | 2.29 | 2.31 | 2.31 | 2.06 | 2.06 | 2.07 |
| | 通信時間† | M | 346 | 388 | 489 | 193 | 212 | 271 | 460 | 514 | 578 | 67 | 74 | 72 | 419 | 421 | 420 |
| | | V | 186 | 207 | 257 | 97 | 106 | 136 | 236 | 264 | 296 | 33 | 37 | 36 | 211 | 211 | 211 |
| | | M/V | 1.86 | 1.87 | 1.90 | 1.98 | 2.00 | 1.99 | 1.94 | 1.94 | 1.95 | 2.03 | 2.00 | 2.00 | 1.98 | 1.99 | 1.99 |
| | 平均命令並列度‡ | V | 2.01 | 2.01 | 2.02 | 2.29 | 2.29 | 2.31 | 2.09 | 2.07 | 2.06 | 2.29 | 2.28 | 2.27 | 2.11 | 2.11 | 2.10 |
| | 各演算器使用率§ | V | 0.50 | 0.50 | 0.50 | 0.57 | 0.57 | 0.57 | 0.51 | 0.52 | 0.51 | 0.57 | 0.57 | 0.56 | 0.52 | 0.52 | 0.52 |

†単位:*1,000,000*クロック・サイクル
‡処理時間に対しての平均命令並列度
§処理時間に対しての各演算器使用率

理時間に対してのみ計算）は物体探索プロセッサが 0.56〜0.57 で全ベンチマーク・シーンで一定である．交点計算プロセッサはベンチマーク・シーンにより若干差があり 0.50〜0.57 である．低い使用率を示すのが，ベンチマーク・シーン balls で，mount と tetra が高い使用率を示す．

- VLIW アーキテクチャによりハードウェア量が4倍になったものの，性能はたかだか2倍程度しか達していない．これは，今回の VLIW シミュレーション用コードが，逐次処理用コードを単に改良したものであることに原因がある．より大きな性能向上を図るには，並列処理に向いたコードの開発，あるいは，物体探索および交点計算ステージで使用するアルゴリズムを更に並列性の高いアルゴリズムに改良するという方法が考えられる．

## 6. おわりに

以上，レイトレーシング法を物体探索，交点計算，輝度計算の三つのサブタスクでオーバラップ処理する3MPRT (3-stage MacroPipelined Ray Tracing) を提案し，それを直接実装する要素プロセッサ3MPPE (3-stage MacroPipelined Processing Element) の命令レベル並列処理，VLIW プロセッサの構成，等について述べた．そして 3MPPE をソフトウェア・シミュレーションにより評価した．

性能評価の結果，VLIW 処理により最低 2.08〜最高 3.87 倍の性能向上が可能であることが判明した．また，VLIW 処理は空間の分割数に依存しないことも判明した．しかしながら，物体探索プロセッサおよび交点計算プロセッサでの平均命令並列度は，VLIW アーキテクチャの採用によりハードウェア量が4倍になったものの，たかだか2倍程度にしか達していない．また，各演算器の使用率も，それぞれ0.5程度である．これは，今回の VLIW シミュレーション用コードが，逐次処理用コードを改良したものであることに原因がある．これへの対処としては，並列処理に向いたコードの開発が考えられる．あるいは，物体探索および交点計算ステージで使用するアルゴリズムを更に並列性の高いアルゴリズムに改良するという方法も考えられる．

3MPPE は，われわれが開発しているメモリ共有型マルチプロセッサ・システム『熱視線』の要素プロセッサである．今後は，これを用いたマルチプロセッサ・システム上での負荷分散，レジデント・メモリ，キャッシュ・メモリの効果などについて評価を行う予定である．

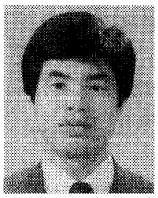
**謝辞** 九州大学大学院総合理工学研究科情報システム学専攻の安浦寛人教授，ならびに，安浦研究室の諸氏に感謝します．

## 参考文献

1) 秋本，間瀬：画素選択型光線追跡法，信学論，Vol. J 69-D, No. 12, pp. 1943-1952 (1986).
2) 河合，山下，大野，吉村，西村，下條，宮原，大村：並列画像生成システム LINKS-2 のアーキテクチャ，情報処理学会論文誌，Vol. 29, No. 8, pp. 729-739 (1988).
3) 權，村上，富田：『熱視線』：視線探索法を高速処理する専用並列レンダリング・マシン—マクロパイプライン・アーキテクチャ—，情報処理学会研究会報告，ARC-85-6 (1990).
4) 權，村田，村上，富田：レイトレーシング法を高速処理する専用並列レンダリング・マシン『熱視線』の要素プロセッサ・アーキテクチャ—マクロパイプライン・アーキテクチャおよび性能評価—，情処論，Vol. 34, No. 2, pp. 257-272 (1993).
5) 鷲島，西澤，浅原：並列図形処理，コロナ社 (1991).
6) 出口，西村，吉村，河田，白川，大村：コンピュータグラフィックスシステム LINKS-1 における画像生成の高速化手法，情報処理学会論文誌，Vol. 25, No. 6, pp. 944-952 (1984).
7) 出口，西田，西村，河田，白川，大村：視線探索法による画像生成のための木構造並列処理システム，信学論，Vol. J 69-D, No. 2, pp. 170-179 (1986).
8) 平井，日高，浅原，鷲島：画像生成用 SIMD 型マルチプロセッサシステム MC-2，情報処理学会研究会報告，MDP-29-5 (1986).
9) 堀江，池坂，石畑：並列計算機 CAP-II のルーティング・コントローラ，情報処理学会研究会報告，ARC-83-38 (1990).
10) 村田，權，村上，富田：『熱視線』：視線探索法を高速処理する専用並列レンダリング・マシン—マクロパイプラインの各ステージにおける命令レベル並列処理—，並列処理シンポジウム JSPP '91 論文集，pp. 109-116 (1991).
11) 吉田，成瀬，高橋，内藤：グラフィックス計算機 SIGHT の基本構成，情処研報，CA-60-4 (1985).
12) Cleary, J. G. and Wyvill, G.: Analysis of an Algorithm for Fast Ray Tracing Using Uniform Space Subdivision, *The Visual Computer*, Vol. 4, pp. 65-83 (1988).
13) Dippe, M. and Swensen, J.: An Adaptive Subdivision Algorithm and Parallel Architecture for Realistic Image Synthesis, *Proc. SIGGRAPH '84*, pp. 149-158 (1984).

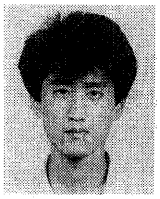
14) Fujimoto, A., Tanaka, T., and Iwata, K.: ARTS: Accelerated Ray Tracing Systems, *IEEE Computer Graphics & Applications*, Vol. 6, No. 4, pp. 16-26 (1986).
15) Gaudet, S., Hobson, R., Chilka, P., and Calvert, T.: Multiprocessor Experiments for High-Speed Ray Tracing, *ACM Trans. Graphics*, Vol. 7, No. 3, pp. 151-179 (1988).
16) Glassner, A. S.: Space Subdivision for Fast Ray Tracing, *IEEE Computer Graphics & Applications*, Vol. 4, No. 10, pp. 15-22 (1984).
17) Green, S. A. and Padden, D. J.: Exploiting Coherence for Ray Tracing, *IEEE Computer Graphics & Applications*, Vol. 9, No. 6, pp. 12-26 (1989).
18) Gwun, O., Murata, S., Murakami, K. and Tomita, S.: A Parallel Rendering Machine for High-Speed Ray Tracing, *Proc. Int'l. Symp. Supercomputing '91*, pp. 173-182 (1991).
19) Haines, E. A. and Greenberg, D. P.: The Light Buffer: A Shadow-Testing Accelerator, *IEEE Computer Graphics & Applications*, Vol. 6, No. 9, pp. 6-16 (1986).
20) Haines, E. A.: A Proposal for Standard Graphics Environments, *IEEE Computer Graphics & Applications*, Vol. 7, No. 11, pp. 3-5 (1987).
21) Hall, R. A. and Greenberg, D. P.: A Testbed for Realistic Image Synthesis, *IEEE Computer Graphics & Applications*, Vol. 3, No. 10, pp. 10-20 (1983).
22) Synder, J. M. and Barr, A. H.: Ray Tracing Complex Models Containing Surface Tessellations, *Proc. SIGGRAPH '87*, pp. 119-128 (1987).
23) Earnshaw, R. A.: *Theoretical Foundations of Computer Graphics and CAD*, Springer-Verlag (1987).
24) Ullner, M. K.: *Parallel Machines for Computer Graphics*, Ph. D. Thesis, California Institute of Technology, Report Number 5112: TR: 83 (1983).
25) Weghorst, H., Hooper, G. and Greenberg, D. P.: Improved Computational Methods for Ray Tracing, *ACM Trans. Graphics*, Vol. 3, No. 1, pp. 52-69 (1984).
26) Whitted, T.: An Improved Illumination Model for Shaded Display, *Comm. ACM*, Vol. 23, No. 6, pp. 343-349 (1980).

(平成4年10月7日受付)
(平成5年5月12日採録)



**權 五鳳** (正会員)

1954年生. 1980年韓国高麗大学工学部電気工学科卒業. 1983年同大学院制御工学専攻修士課程修了. 同年韓国大丘工業専門大学専任講師. 1992年九州大学大学院総合理工学研究科博士課程単位取得退学. 同年九州大学工学部情報工学科助手, 現在に至る. 計算機アーキテクチャ, 並列処理システム, コンピュータ・グラフィックスの研究に従事. ACM, IEEE 各会員.

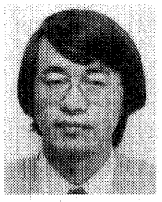


**村田 誠治** (正会員)

1968年生. 1991年九州大学工学部情報工学科卒業. 1993年九州大学大学院総合理工学研究科情報システム学専攻修士課程修了. 同年富士通(株)入社, 現在に至る.

**村上 和彰** (正会員)

1960年生. 1982年京都大学工学部情報工学科卒業. 1984年同大学院修士課程修了. 同年富士通(株)本体事業部に入社, 汎用計算機Mシリーズのアーキテクチャ開発に従事. 1987年九州大学工学部助手, 1992年同大学院総合理工学研究科講師, 現在に至る. 計算機アーキテクチャ, 並列処理, 性能評価などの研究に従事. 著書「計算機システム工学(共著)」,「ヘネシー&パターソン: コンピュータ・アーキテクチャ(共訳)」. 情報処理学会研究賞(平成3年度), 情報処理学会論文賞(平成3年度)受賞. 電子情報通信学会, 日本応用数理学会, ACM, IEEE, IEEE-CS 各会員.



**富田 眞治** (正会員)

1945年生. 1968年京都大学工学部電子工学科卒業. 1973年同大学院博士課程修了. 工学博士. 同年京都大学工学部情報工学教室助手. 1978年同助教授. 1986年九州大学大学院総合理工学研究科教授, 1991年京都大学工学部情報工学科教授, 現在に至る. 計算機アーキテクチャ, 並列処理システムなどに興味を持つ. 著書「並列計算機構成論」「計算機システム工学」「並列処理マシン」など. 電子情報通信学会, IEEE, ACM 各会員.