計算機アーキテクチャ 97−14
（1992. 12. 4）

FMPPによるレイトレーシングの高速化手法について

山田 哲也　　安浦 寛人

九州大学 大学院総合理工学研究科 情報システム学専攻

〒816 福岡県春日市春日公園 6-1

E-mail: {tetsuya, yasuura}@is.kyushu-u.ac.jp

メモリはデータを記憶する回路であるが，メモリのビットまたはその集合を1単位と捉え，それに付加回路を付ければ，記憶以外の機能を持つ回路ができる．このような回路は機能メモリといわれている．特に付加回路を算術論理演算回路とし，上記の集合を1プロセッサとみなすと，メモリ内でSIMD型超並列プロセッサを構成できる．これを我々はFMPP(Functional Memory type Parallel Processor)と呼んでいる．本稿ではメモリセルに代表的な機能メモリであるCAMを用いたFMPP上で，コンピュータグラフィックスの代表的な3次元映像生成アルゴリズムであるレイ・トレーシング・アルゴリズムを実現する手法について述べる．レイトレーシングは，光学モデルを基本にしたアルゴリズムであるため優れた画像を提供するが，処理時間が非常に長い．レイ・トレーシングの中心演算である交差判定演算の並列性を抽出し，FMPPによる処理の高速化について考察を行なった．通常のDRAM上で12Mバイトのデータをトランジスタ数100万のFMPPを用い交差判定演算を実行すると，逐次的に演算を行なった場合と比較して，約100倍の高速処理が可能である．

High speed Ray Tracing Algorithms on FMPP

Tetsuya Yamada　Hiroto Yasuura

Department of Information Systems
Interdisciplinary Graduate School of Engineering Sciences
Kyushu University
Kasuga-shi, Fukuoka 816 Japan

E-mail: {tetsuya,yasuura}@is.kyushu-u.ac.jp

FMPP (Functional Memory type Parallel Processor) is an SIMD type parallel processor architecture which consists of a small arithmatic logical unit added to each word or set of words. In this article, each memory cell of FMPP is based on CAM (Content Addresable Memory). Ray tracing is widely used for rendering high quality graphic images. Finding the nearest intersected object for each lay is the most time consuming processing in raytracing. We propose parallel algorithms for calculation of the intersections. When we apply FMPP (100M transistors) to the intersection calculation (100M Byte Data), the computation time becomes 100 times faster than the computation on a sequential computer.

1 はじめに

現在，コンピュータ・グラフィックス (CG) は産業界のみならず，芸術，教育の分野でも広く用いられている．言葉や数値では表しにくい現象も可視化することで，理解を助けることも多い．今後，益々CG の果たす役割は大きくなるものと思われる．

CG の分野でリアルな映像を提供する手法にレイ・トレーシング・アルゴリズムがある．これは，光学モデルを用いたアルゴリズムであるため，影，反射，屈折を表現するのに優れている．反面，計算時間が長くなるため，大きな計算能力を必要とし，これまでに数多くの専用マシンが提案試作されている．レイ・トレーシング・アルゴリズムの処理に最も時間を要するのは交差判定演算である．しかし，その処理は比較的単純な演算 (3 × 3 逆行列と 3 × 1 ベクトルの積) を反復するものであり，個々の演算は独立に計算できるので並列に演算を行なうことができる．

機能メモリとは，データの記憶機能しかもたないメモリに目的に応じて必要な演算機能を付加したメモリのことをいい，様々なものが提案されている．CAM(Content Addresable Memory) は機能メモリの代表的なものなものといえる．通常，連想メモリとも呼ばれ，データ内容の一部を参照することによりアクセスを行なうものである．

機能メモリにおいて，メモリの各ワードあるいはワードのグループをプロセッサとみなすと，メモリ全体をプロセッサの集合とみなすことができる．これを我々は機能メモリ型並列プロセッサアーキテクチャ(Functional Memorytype Parallel Processor:FMPP) と呼んでいる [1]．付加回路としてワードごとに算術論理演算器を与え，これを 1 プロセッサとみなせば，メモリは算術論理演算用並列プロセッサとなる．メモリ内で演算を行なうため，簡単な算術論理演算しか扱えないものの，メモリアクセスに伴うボトルネックのない超並列プロセッサということができる．FMPP はホストマシンに接続し，SIMD 型並列マシンを形成し，各プロセッサは同一処理を行なう．プロセッサ間の通信はバスを介して行なう．本稿では CAM を基本メモリセルとした FMPP を取り扱う．

FMPP は同種の演算を再帰的に極めて多数回計算するものに絶大な効果を発揮する．本稿では FMPP を用い，レイ・トレーシング・アルゴリズムの交差判定を高速化する手法を考える．FMPP をワークステーション等のメモリ部に付加し，交差判定演算を FMPP に，他をホストマシンに担当させることにより，全体として複雑なレイ・トレーシングの計算が実現できる．交差判定はレイ・トレーシングの計算時間の大きな部分を占めるので，全体の処理時間をかなり短縮することができると思われる．

本稿では 2 章において FMPP の概要と命令セットを述べる．3 章ではレイ・トレーシングの概略，4 章ではそのうちの交差判定についてその並列化に触れる．5 章で評価及び考察を述べる．6 章でまとめと問題点を挙げる．

2 機能メモリ型並列プロセッサアーキテクチャFMPP

FMPP はメモリ-演算器構成により 4 つに分類される．

- BPWP(Bit Parallel Word Parallel)
  ビットごとに付加回路を備える．ハードウェアコストが大きい．
- BSWP(Bit Serial Word Parallel)
  全ワードの共通ビットに 1 付加回路を備える．ビット数分の付加回路を用意すればよいが，並列性は低い．
- BPWS(Bit Parallel Word Serial)
  1 ワードに 1 付加回路を備える．並列性が BPWP に比べ 1/(ビット数) に低下する．
- BPBP(Bit Parallel Block Parallel)
  1 ブロックに 1 付加回路を備える．1 ワードで納まらないデータの格納ができる．並列度は BPWP 方式に比べ 1/(1 ブロックのワード数) の低下で済む．ブロック内部のデータのみを用いて演算が可能である [2].

今回は BPBP 方式を用いている．図 1 に本稿で用いる FMPP のブロック図を示す．メモリを数ワードからなる小ブロックに分割し，そのブロックをそれぞれ 1 個のプロセッサとして動作させる [3]．1 ブロックに 1 つの演算回路を設ける．アドレス部に

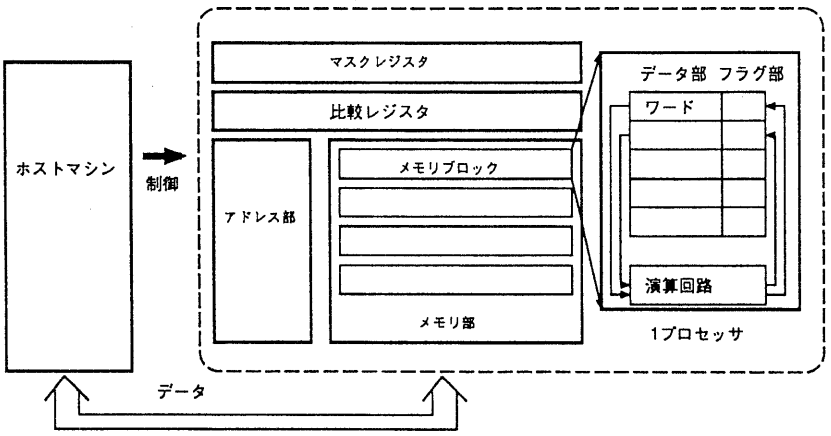

図 1: FMPP のブロック図

はROM，データ部にはRAMを用い，それぞれアドレスバス，データバスでホスト計算機のアドレスバス，データバスに接続されている．ROM部はアドレスデコーダも兼ねており，データのアドレスを記憶している．演算器は基本的には論理回路，加算器，シフト器，キャリ伝搬器からなる．ワード間の加算，論理演算は1クロックで行う．FMPPの単位セルにはCAMセルを用いているので，一致検索，並列書き込み機能も装備されている．1ワードはデータ部分を32ビット，状態フラグを8ビットの計40ビットで構成される．マスクレジスタはメモリ領域及びアドレス領域においてビット単位で制御を行う．ビットシフト操作により n ビットの乗算も $O(n)$ ステップで実現できる．

本稿で利用するFMPPの命令セットを次に示す．

① MASKSET(data) マスクレジスタセット
書込みまたは一致検索時のマスクデータをセットする．

② REF *function* (data) 一致検索
アドレス部またはデータ部のマスクつき一致検索を行なう．完全に一致したものに論理値1，一致しないものには論理値0を各ワードに備えられた簡単な論理演算器に送り，フラグの内容と *function* で指定した論理演算を行ないフラグに結果を格納する．*function* にはthrough，and，orがある．マスクされたビットは値にかかわらず一致したとみなされる．(don't care)

③ FUNC(adr1,adr2,adr3) 2ワードの論理演算
ブロック中任意の2ワードのビットサイズ論理演算を行ない，演算結果はブロック中どのワードにでも書き込むことができる．データの指定はアドレスで行なう．FUNCにはAND，OR，EXORを備える．

④ ADD(adr1,adr2,adr3) 2ワードの算術加算
ブロック中の任意の2ワードの加算を行ない，演算結果はブロック中どのワードにでも書き込むことができる．データの指定はアドレスで行なう．

⑤ BITCOPY(adr1,adr2,i,j) ブロック内での任意のビットから任意のビットへの転送を行なう．$adr1_i \rightarrow adr2_j$

⑥ LSHIFT(adr) ブロック中の1ワードについて全ビット左にシフトさせる．

⑦ 並列書込み WRITES(data)
選択フラグが立っているものに対しての並列書込みを行なう．マスクしたビットは書き換えられない．

⑧ 全プロセッサ選択 SETS
すべてのプロセッサを選択状態とする．

⑨ 全プロセッサ非選択 RESETS
すべてのプロセッサを非選択状態とする．

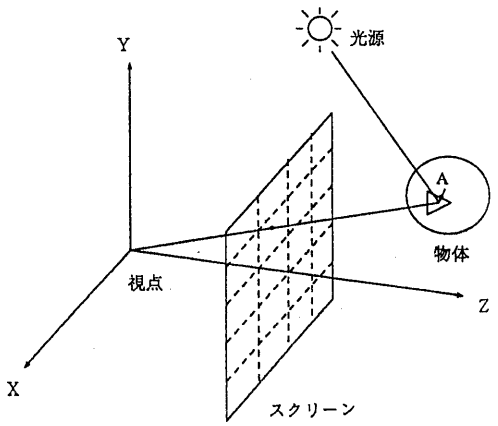

図 2: レイ・トレーシングの原理

⑩ アドレスによる選択状態指定
SETSADR(adr)
アドレスによりプロセッサの選択状態を定める.

3 レイ・トレーシングの概要

レイ・トレーシング・アルゴリズムは物体空間の中で光の動きをシミュレーションして映像を生成する手法である. 具体的に言うと, 人が物を見る過程と逆方向, つまり視点から光源への視線を出し, 視線と物体の交点から物体を表示する方法である. レイ・トレーシングの原理を図 2に示す. 隠面消去と輝度計算のステージから構成される.

- 隠面消去
  視点から各ピクセルを通る視線 (一次視線) を出し, その視線と交差する物体があるかどうかを調べ (交差判定), 各ピクセルから可視となる物体面状の交点 (最も距離の近い交点) を求める.

- 輝度計算
  各ピクセルの色を決めるため, 物体との交点 A から光源方向の二次視線を出し, 追跡を行なう (交差判定). もし, 二次視線が新たな物体と衝突すれば交点 A は影となる. 交点 A がないとき, つまり一次視線がどの物体とも衝突しなかったときは背景色を与える. 物体に反射, 屈折特性があるときは, 反射, 屈折方向に二次視線を出す. 以後衝突する物体がある場合, 物体の光学属性に合わせてその交点から再帰的に追跡を行なう (交差判定). 全ての色の重ね合わせでピクセルの色を決める.

交差判定は隠面消去と輝度計算の双方に現れ, レイ・トレーシングの処理のうち簡単な画像で約 75%, 複雑な画像では約 95%を要すると指摘されている [4].

4 交差判定とその並列処理

4.1 交差判定

ここでの交差判定は各ピクセルから可視となる点を求めるものとし, 対象となる物体の表面は予め三角形で細分化されているものとする. これをここでは三角ポリゴンと呼ぶ. 視線は原点にくるよう座標変換を行なっているので, 視線の方向ベクトルは, 原点とピクセルを結ぶ方向になる.

定義 1 (交差判定)

- 入力 : 三角ポリゴンの座標 (P_{1x}, P_{1y}, P_{1z}), (P_{2x}, P_{2y}, P_{2z}), (P_{3x}, P_{3y}, P_{3z})
- 視線の方向ベクトル $V = (a, b, c)$
- 出力 : $(T_1, T_2, T_3), Q$(各視線と交わる三角ポリゴンのうち, 視点から最も距離の近いものとの交点の座標)
- 計算式

$$\begin{pmatrix} T_1 \\ T_2 \\ T_3 \end{pmatrix} = \begin{pmatrix} P_{1x} & P_{2x} & P_{3x} \\ P_{1y} & P_{2y} & P_{3y} \\ P_{1z} & P_{2z} & P_{3z} \end{pmatrix}^{-1} \begin{pmatrix} a \\ b \\ c \end{pmatrix} \tag{1}$$

交点が三角ポリゴン内にあるための条件は, 上記の T_1, T_2, T_3が

$$T_1 \geq 0 \text{ かつ, } T_2 \geq 0 \text{ かつ, } T_3 \geq 0$$

を満たすことである. 式 (1) の逆行列は既に計算されているものとし,

$$\begin{pmatrix} P'_{1x} & P'_{2x} & P'_{3x} \\ P'_{1y} & P'_{2y} & P'_{3y} \\ P'_{1z} & P'_{2z} & P'_{3z} \end{pmatrix}$$

で表す.
次に上記の条件を満たす三角ポリゴンのうち視点に最も距離の近いものを選び，視線とその三角ポリゴンの交点を求める．交点の座標を Q とすれば

$$Q^T = \frac{V^T}{(T_1 + T_2 + T_3)}$$

で表される.

4.2 交差判定の並列処理

交差判定の計算量が膨大なものになるのは，各ピクセルを通過する視線が空間中のすべての三角ポリゴンと交差するかどうかを全て計算するためである．そのため，交差判定の計算を高速化するための研究が盛んに行なわれている．専用高速計算機を利用して単位時間当たりの交差判定の数を増やす方法や必要な交差判定の数を減らすアルゴリズムがある．前者には，視線が独立であることを利用したMIMD型並列計算機，後者にはオクトリ法，空間等分割法がある [5].

産業界で用いられるグラフィックスモデルには数百万個のオブジェクトを含む．視線の数はピクセル数であるから，1024 × 1024 の解像度の場合，$1024^2 = 1M$存在する．従って，三角ポリゴンと視線の数は双方とも数100万個であり，そのすべての組み合わせの行列演算を行なう必要がある．我々は交差判定演算が，各々のベクトルに対し，すべての逆行列との積をとり，それらは独立に計算できることに着目し，視線と三角ポリゴンをそれぞれFMPPに割り当てることにした．視線の数と三角ポリゴンの数はそれぞれ n, m とする.

- 行列要素をFMPPの各プロセッサに割り付ける (オブジェクト並列)
  1回の処理ではベクトルは共通，n 回異なるベクトルをホストマシンより放送して計算を行う.
- ベクトルをFMPPの各プロセッサに割り付ける (視線並列)
  1回の処理では行列は共通，m 回異なる行列を放送して計算を行う.

条件として，逆行列が予め計算されているとしているが，一旦値を計算しておけば，一次視線のみならず高次視線まで共通なので，前処理で計算済みと仮定している．FMPPのアルゴリズムを以下に示す．主にデータ転送，乗算，加算，フラグ操作，正負判定，最大値検索から成る．基本演算の命令数は表1の様になる.

演算	命令数	オーダ
データ転送	5	$O(1)$
乗算	$4n-2$	$O(n)$
正負判定	4	$O(1)$
最大値検索	$3n+1$	$O(n)$

表 1: 基本演算の命令数

定義 2 (FMPPでの交差判定)

- 入力：逆行列成分 $(P'_{1x}, P'_{1y}, P'_{1z})$, $(P'_{2x}, P'_{2y}, P'_{2z})$, $(P'_{3x}, P'_{3y}, P'_{3z})$
- 視線の方向ベクトル $V = (a, b, c)$
- 出力：
  - *(視線並列)* 視線の方向ベクトル $V = (a, b, c)$, 逆行列成分のインデックス
  - *(オブジェクト並列)* 視線の方向ベクトル $V = (a, b, c)$, 逆行列成分の座標 $(P'_{1x}, P'_{1y}, P'_{1z})$, $(P'_{2x}, P'_{2y}, P'_{2z})$, $(P'_{3x}, P'_{3y}, P'_{3z})$
- 計算式

$$\begin{pmatrix} T_1 \\ T_2 \\ T_3 \end{pmatrix} = \begin{pmatrix} P'_{1x} & P'_{2x} & P'_{3x} \\ P'_{1y} & P'_{2y} & P'_{3y} \\ P'_{1z} & P'_{2z} & P'_{3z} \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

FMPPで演算を行なうのは具体的には以下のものである.

- 各三角ポリゴンと視線の方向ベクトル V に対して T_1, T_2, T_3 の計算
- $T_1 \geq 0$ かつ, $T_2 \geq 0$ かつ, $T_3 \geq 0$
  (交点が三角ポリゴン内にあるか否かの判定)
- 上記の条件を満たすもののみ $T_1 + T_2 + T_3$ の計算
- 上記のうち最大のものを検索する

以上の条件を満たすものにはフラグをたて，読みだし，交点計算はホスト側で行なう.
視線の方向ベクトルを1プロセッサとした視線並

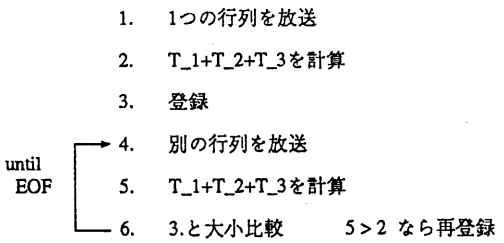

図 3: 視線並列アルゴリズム

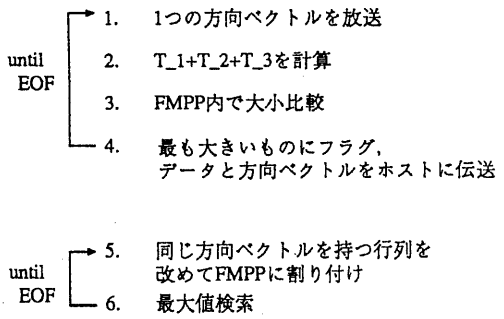

図 4: オブジェクト並列アルゴリズム

列と三角ポリゴンを 1 プロセッサとしたオブジェクト並列の処理の流れを図 3と図 4に示す．ここで図 3の登録とは $T_1 + T_2 + T_3$の計算値と行列データを FMPP 内の各プロセッサに保存することである．視線並列では全処理が終了した時点で各プロセッサにはその視線と交わる最も距離の近い三角ポリゴンのアドレス計算に利用できるインデックスが保存されているのに対し，オブジェクト並列では，1 処理ごとに各方向ベクトルと交わる最も距離の近い三角ポリゴンの座標が読み出される．1 プロセッサのワード数は視線並列の場合 23，ベクトル並列の場合 36 となる．

データが FMPP の容量より大きい場合には，視線並列は容量に関わらず各視線に対し，それぞれ三角ポリゴンが求まっているのに対し，オブジェクト並列は同じ方向ベクトルを持つ三角ポリゴンを FMPP に再割り付けして調べるか，ホスト側で最大値検索を行なう．

5 評価及び考察

レイ・トレーシングの交差判定演算について，逐次型計算機と FMPP の実行時間，トランジスタ

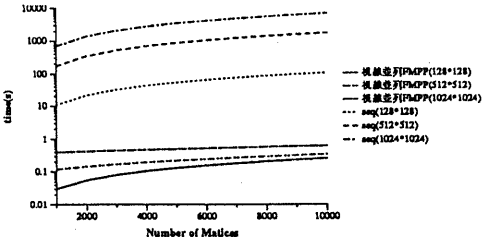

図 5: 理想的な視線並列 FMPP と逐次型の処理時間

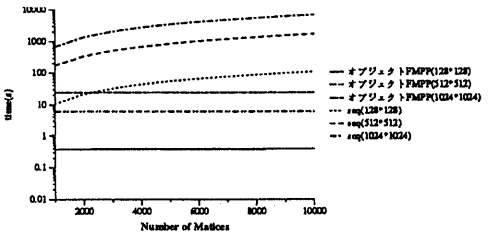

図 6: 理想的なオブジェクト並列 FMPP と逐次型の処理時間

数を調べた．尚，実行時間に関しては，$T_1 \geq 0$ かつ $T_2 \geq 0$ かつ $T_3 \geq 0$ の判定までに要するものとしている．これは，FMPP がその後一致検索機能を利用して一定時間で求まるのに対し，逐次型計算機の場合は与えられる問題に応じて処理時間が大幅に異なることを考慮したためである．

逐次型計算機は乗算器と加算器を備え，レジスタ上で演算を行うと仮定する，JUMP 時に *condition code* を参照するとすると，1 回の処理に 33 ステップを要する．

5.1 理想的な FMPP

行列数は 10000 とする．

ベクトル数	128×128	512×512	1024×1024
FMPP のプロセッサ数	128×128	512×512	1024×1024
視線並列 (対逐次型)	420 倍	5000 倍	11000 倍
オブジェクト並列 (対逐次型)	277 倍	285 倍	285 倍

表 2: 理想的な FMPP の逐次型に対する性能向上率

ベクトル数	128×128	512×512	1024×1024
DRAM バイト数	192M	3M	12M
トランジスタ数 (10^6)	1.57	25.2	101
FMPP のバイト数	160K	2.5M	10M
FMPP のトランジスタ数	375M	5.86G	23.5G
FMPP のトランジスタ数 (対 DRAM)	250 倍	250 倍	250 倍

表 3: DRAM データと FMPP のトランジスタ数

図 5と図 6に理想的な FMPP について視線並列とオブジェクト並列の場合の逐次型に対する処理時間，表 2に性能向上率を示す．クロック周波数は 50MHz, ベクトル数はピクセル数に合わせ，128 × 128，512 × 512，1024 × 1024 の 3 通りについて調べた．行列数は 10000 としている．視線並列ではすべてのベクトルが，オブジェクト並列ではすべての行列がメモリ上にあるものとしている．視線並列の場合，ステップ数は $36n + 95$，オブジェクト並列ではステップ数は $36n + 5$ である．表 3 にトランジスタ数を比較している．逐次型の場合データは DRAM 上にあるとしている．FMPP では，基本メモリセルに SRAM を用いた CAM から構成される．FMPP のトランジスタ数は膨大なものであるが，ベクトル数が増大しても逐次型に対し一定倍に留まっている．

表3に FMPP と DRAM のトランジスタ数を比較している．FMPP では，基本メモリセルを SRAM を用いた CAM から構成される．この場合の FMPP のトランジスタ数は膨大なものであるため，現実的でないといえる．

5.2 メモリ容量に制限のある FMPP

次にデータ量が FMPP の容量より大きい場合，つまり数度にわたり FMPP 上のデータを更新する必要がある場合について述べる．限られた容量の FMPP を用いて処理を行なう場合に逐次型に対しどの程度の性能向上が得られるかについて述べる．

まず，オブジェクト並列では，三角ポリゴンは与えられる問題に大きく依存するので，データ数より多いプロセッサを用意している場合，無駄が生じる．視線並列では，ピクセルの数は一定 (通常 1024 × 1024) であるため，問題に依存しないこと，更にアドレス計算が容易であることが挙げられる．以上よりピクセルを 1 プロセッサに割り当てた方が高効率，高速であることがいえる．よって以後この場合について容量制限の FMPP を考察する．

1 プロセッサに複数のピクセルを含める方法も考えられる．これは演算器を減らし，全体のトランジスタ数を押えることとリプレイスの時間を省略できるようにみえる．しかし，実際にトランジスタ数を消費するのは CAM メモリセルで，1 ビット当たり 18 個であり，演算回路部分は 1000 個に過ぎず，2 ワード分にも満たない．以上から 1 プロセッサに複数ピクセルを載せても小容量化に貢献するとは考えられないので，1 プロセッサに 1 ピクセルを載せるものとする．

図 7にピクセル数 1024 × 1024 のとき，FMPP のプロセッサ数が 64 × 64，256 × 256 の場合の逐次型に対する処理速度を調べた．データ 1 ワードをリプレイスするのに 6 ステップ要するものとする．行列数が増すにつれてリプレイス処理の割合は小さくなる．行列数が 10000，プロセッサ数が 256 × 256 のとき逐次型に対し約 1500 倍，64 × 64 のとき約 130 倍の結果を得た．トランジスタ数は FMPP 部分については，プロセッサ数が 256 × 256 のとき 1.47G トランジスタ数，64 × 64 のとき 93.8M トランジスタ数の結果を得た．後者の場合，全データを DRAM に格納するのに必要なトランジスタ数 101M とほぼ等しいトランジスタ数を用いた FMPP で逐次型に対し約 130 倍の性能を発揮することが判明した．

6 おわりに

FMPP を用い，レイ・トレーシング・アルゴリズムの交差判定演算の高速化の検討を行なった．

- 理想的な FMPP を用いたときの逐次型計算機に対する処理時間とトランジスタ数
- メモリ容量に制限を与えた場合の逐次型計算機に対する FMPP の処理時間とトランジスタ数

前者の場合，FMPP の処理速度は逐次型の数万倍，後者の場合 100 倍の結果を得た．後者ではメモリの現実的制約より FMPP の処理速度は格段に落ちてはいるものの十分高性能である．レイ・トレー

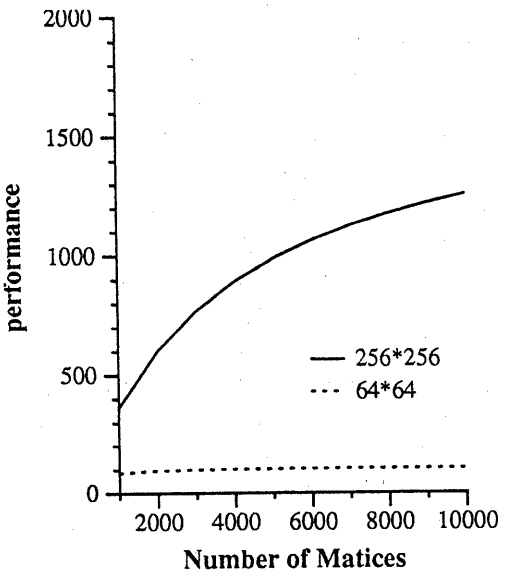

図 7: データのリプレイスを考慮した場合

シング処理のうち，交差判定演算を 90%，残りを 10%とすると，アムダールの法則より交差判定の高速化度が 100，10000 のとき，全体の速度向上率はそれぞれ 9.17，9.99 倍であり大差がない．つまり，交差判定に関していえば高速化度が 100 倍で十分で，他の処理を更に高速化することを考えるほうが望ましいといえる．

FMPP が今後大容量化すると，データの読みだし，リプレイスがボトルネックとなると思われる．メインメモリから FMPP へのデータ伝送の高速化と FMPP による浮動小数点演算を今後の課題とする．

謝辞

日頃ご討論頂く九州大学 大学院総合理工学研究科 村上和彰講師，ならびに，安浦研究室の諸氏に感謝致します．

本研究の一部は文部省科学研究費補助金一般研究 B「機能メモリ型並列プロセサアーキテクチャとそれを用いた超並列アルゴリズムの研究」(02452160) および重点領域研究 (2)「超並列アルゴリズム設計のためのデータ構造と計算モデルに関する研究」(04235207) による．

参考文献

[1] 安浦寛人. “機能メモリによる超並列処理”. 情報処理, Vol. 32, No. 12, pp. 1260–1267, December 1991.

[2] 小林和淑, 安浦寛人, 田丸啓吉. “ビット並列ブロック並列方式による機能メモリ型並列プロセッサアーキテクチャーの提案”. 情報処理学会第 43 回全国大会, pp. 6-57-6-58, October 1991.

[3] 小林和淑, 田丸啓吉, 安浦寛人. “新しい機能メモリの提案とその応用について”. 平成 3 年電気学会電子・情報・システム部門全国大会, pp. 209–212, 1991.

[4] Turner Whitted. “An Improved Illumination Model for Shaded Display ”. *cacm*, Vol. 23, No. 6, pp. 343–349, June 1980.

[5] 鷲島敬之, 西澤貞爾浅原重夫. “並列図形処理”. コロナ社, 1991.