FPGA を用いたフォトンマッピング法高速化手法の提案と性能評価

久原 拓也

1　はじめに

現実世界における光の動向を追跡して画像をレンダリングするコンピュータグラフィックス (Computer Graphics:CG) の技術に，る．フォトンマッピング法は，レイトレーシング法の拡張手法として，1996 年に H.W.Jensen によって提案された手法である [1]．フォトンマッピング法は，計算量が多く処理時間が長い一方で高い並列性を持っており，Graphics Processing Unit(GPU) や，Cell Broadband Engin(Cell/B.E.) といった並列処理を得意とするアーキテクチャによる高速化が研究されている [2] [3] [4]．一方で，フォトンマッピング法では，非常に多くのフォトンと呼ばれる仮想光子に対し繰り返し演算が行われるため，頻繁にメモリアクセスが発生する．CPU や GPU といったメモリアクセスが低速なアーキテクチャでは，メモリアクセスが高速化のボトルネックとなると考えるこれに対し，メモリへのアクセスを含めた処理設計を固有に定義できる専用ハードウェアでは，大きな高速化を期待できる．しかし，専用ハードウェアは，複雑なプログラムの設計を行うのは難しく，また，フォトンマッピング法のように様々な問題サイズを取るプログラムのアルゴリズムを検討するには不向きである．この問題を解決するデバイスとして，FPGA(Field Programmable Gate Array) がある．FPGA は，プログラムによって，柔軟に専用ハードウェアの論理回路を再構築できるデバイスであり，ASIC などに比べ，動作周波数は劣るものの，設計が容易であるためアルゴリズムの検討などに用いられることが多い．また，近年では FPGA 自体の性能が向上し，FPGA を用いたアプリケーション高速化に注目が集まっている．

そこで，本研究では，FPGA を用いたフォトンマッピング法の高速化を検討する．まず，フォトンマッピング法の実行プロファイルを取ることでパラメータと計算時間の関係，およびフォトンマッピング法の計算時間の内訳を分析する．次に，分析結果を元に，フォトンマッピング法の FPGA 実装について述べ，最後に実装の評価と考察を述べる．

2　フォトンマッピング法

2.1　フォトンマッピング法の計算手順

フォトンマッピング法の計算手順について述べる．

まず，第 1 段階として，あらかじめ設定した数だけ，光源からフォトンをランダム方向に放射する．フォトンとは，光のエネルギーと，空間中の位置データを持つ仮想の光子である．放射されたフォトンは，定められた回数の中で，反射や屈折，拡散を繰り返す．この時，フォトンが干渉した空間中の物体表面の座標，フォトンのエネルギーをフォトンマップというデータに記録する．

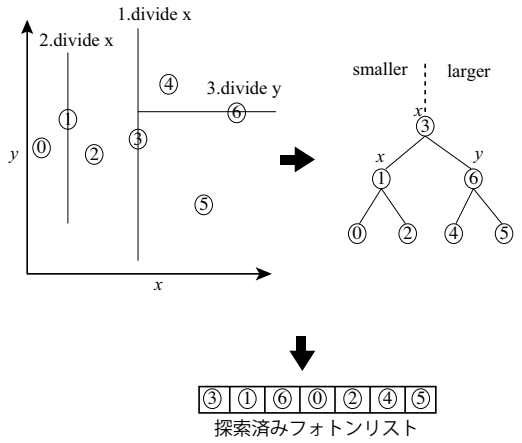

Fig.1　$x-y$ 平面空間における KD 木への格納

次に，第 2 段階として，視点からレイ (視線) を空間中に放射する．レイは，フォトンと同様に反射や屈折を繰り返したのちに空間中の一点に到達する．この処理をレイトレースといい，レイが到達した点をクエリと呼ぶ．画像の生成は，このクエリ近傍のフォトンから色を算出することで，レイが通過した画素の色を決定して行う．クエリ近傍のフォトン探索は，第 1 段階で作成したフォトンマップを元にフォトンを参照し，フォトンがあらかじめ設定した上限探索半径 R 以内に存在するか判定することで行う．

2.2　KD 木

フォトンマッピング法では非常に多くのフォトンを扱うため，全てのフォトンに対し探索を行うことは現実的ではない．そこで，フォトンマップのデータ構造は，クエリ付近のフォトンを効率良く探索できる構造であることが望ましい．このようなデータ構造として，KD 木がある [5]．KD 木は，多次元の 2 分探索木であり，1 つの次元に対応する軸の分割に木のノードが対応している．フォトンマップでは，3 次元 KD 木を扱うことになる．

Fig. 1 に，$x-y$ 平面上に分散したフォトンを KD 木構造に格納する様子を示す (実際は 3 次元だが，簡易化のため 2 次元平面で示す)．

KD 木は以下のような手順で構築する．

1. フォトンの集号について，x, y, z 軸上の各軸について最大値と最小値の差を求め，この差が最も大きい軸を分割軸とする．
2. フォトンの集合について，分割軸成分の中央値を持つフォトンを発見する．このフォトンを軸分割ノード

とする．その際，ノード情報としてフォトンのデータに分割軸を記録する．

3. ノードが分割した軸について，軸分割ノードの値よりも，値が小さいフォトン集団を左の子集団，大きいフォトン集団を右の子集団とする．子集団に含まれるフォトン数が 1 つになるまで，それぞれの集団について，(1) から処理を繰り返す．

このような処理を繰り返すことで，Fig. 1 に示すように，KD 木のリーフノードは細かく分割された空間を管理することになる．KD 木の探索では，クエリが与えられると，まず，木をリーフノードまで辿ることで，クエリが存在する空間を特定する．次に，ルートノードまで戻りながら，広い範囲を探索するという手順をとる．クエリ q を KD 木に入力し，探索フォトン数 N 個のフォトン p を探索する時，KD 木をリーフノードまで辿る操作手順を以下に示す．

1. フォトン p を KD 木から取り出す．探索開始時には，ルートノードを取り出す．
2. p と，クエリ q の絶対距離 D_1 を計算する．
3. D_2 が探索半径 R 内にあれば，p を探索済みフォトンリスト (探索されたフォトンを格納するリスト) に追加する．探索済みフォトンリストに N 個のフォトンが入っている時は，探索済みフォトンリストの中で最も D_1 の大きいフォトンと p を入れ替える．
4. p と，q について，p に記録された分割軸 (x,y,z) 上の距離 D_2 を計算する．
5. 距離が正の場合は p の右の子ノードを，負の場合は左の子ノードを次に探索するフォトンとする．
6. (1)〜(5) をリーフノードにたどり着くまで繰り返す．

次に，ルートノードまで戻りながら，広い範囲を探索する操作手順をいかに示す．

1. 探索フォトンを親ノードのフォトン P とする．
2. q と P について，親フォトンに記録された分割軸上の距離 D_3 を計算する．
3. D_3 が探索半径 R にあり，探索していない子ノードがあれば，探索していない子ノード以下のツリーをリーフノードまで探索する．
4. (1)〜(3) をルートノードにたどり着くまで繰り返す．

この 2 つの探索手順を繰り返すことで探索を行う．探索半径 R は探索開始時に初期値を設定するが，探索済みフォトンリストに N 個のフォトンが集まって以降は，探索済みフォトンリスト中の最も大きい D を R に設定する．このように探索範囲を縮めながら探索を行うことで，無駄な探索を省くことができる．

3 処理時間の調査

3.1 フォトンマッピング法における処理時間の内訳

フォトンマッピング法の高速化を検討するにあたり，処理時間の内訳を計測した．計測環境を Table 1 に示す．

計測項目として，処理全体を，フォトンマッピング法の主な処理内容であると考えられる，レイの放射，フォトンの放射処理，KD 木の構築処理，KD 木の探索処理，それ以外の処理，の 5 つに分類し，処理時間を比較した．プログラムの実行にあたっては，放射フォトン数と探索フォトン数をパラメータとした．結果を，Fig. 2 に示す．

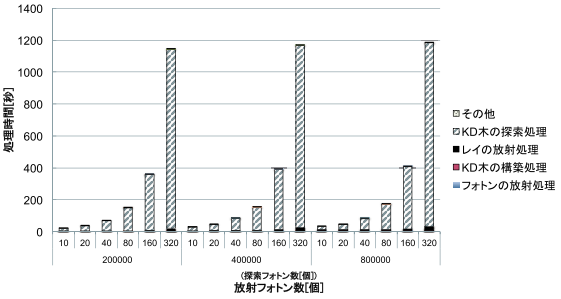

Fig.2　フォトンマッピング法における処理ごとの処理時間内訳

Fig. 2 から，フォトンマッピング法の処理全体で，フォトン探索処理 (斜線部) の占める割合が大きいことがわかる．また，その傾向は放射フォトン数，探索フォトン数が増加するごとに強くなることがわかる．高品質な CG の生成には，1M 個以上の放射フォトン数，500 個以上の探索フォトン数が用いられるため，実用的なフォトンマッピング法の高速化を考えるとき，フォトン探索処理の処理時間が大きな問題となる．そこで，本研究報告では，フォトンマッピング法の中でも，特にフォトン探索処理について高速化の検討を行うこととする．

3.2 フォトン探索処理におけるパラメータの関係

フォトン探索処理について，パラメータによって，どのように処理時間が変化するかを調査した．計測環境として Table 1 の環境を用い，コンパイラオプションに-O3 を用いた．クエリ数と放射フォトン数，探索フォトン数の 3 つをパラメータとした時の，処理時間を折れ線グラフで，1 クエリあたりの平均処理フォトン数を棒グラフで示す．まず，放射フォトン数を 0.1M 個に固定し，横軸にクエリ数を取って，クエリ数と探索フォトン数を変化させた際のグラフを Fig. 3 に示す．

Fig. 3 の折れ線グラフを見ると，クエリ数は処理時間と比例な関係にあることがわかる．また，クエリ数によって多少の誤差はあるものの，1 クエリあたりの処理フォトン数に変化が少ないことがわかる．次に，クエリ数を 0.1M 個に固定し，横軸に放射フォトン数を取って，放射フォトン数と探索フォトン数を変化させたグラフを Fig.

Table1　計測環境

CPU	Core i7 2600K (3.4GHz)
Memory	8GB
OS	Ubuntu 11.04 x86_64
CPU code Compiler	gcc 4.4.5
Language	C++

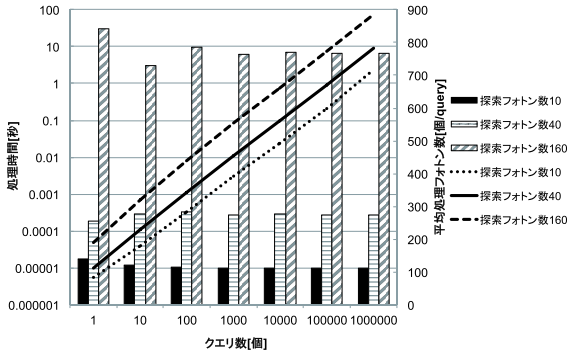

Fig.3　KD 木におけるクエリ数と処理時間

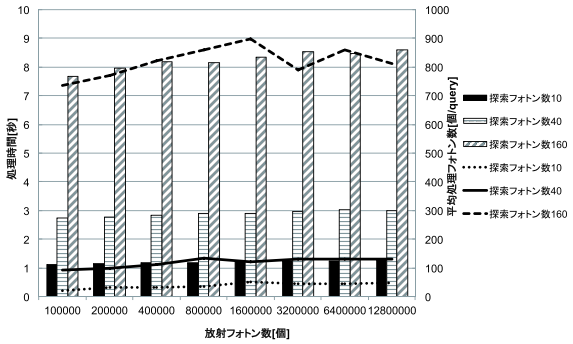

Fig.4　KD 木における放射フォトン数と処理時間

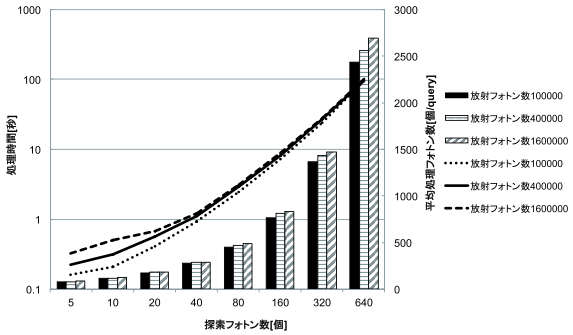

Fig.5　KD 木における探索フォトン数と処理時間

4 に示す．

Fig. 4 の折れ線グラフと棒グラフを見ると，放射フォトン数の増加に対し，処理フォトン数，処理時間があまり変化していないことがわかる．一方で，探索フォトン数の変化によって，処理フォトン数，処理時間が大きく伸びていることがわかる．この関係を確認するため，クエリ数を 0.1M 個に固定し，横軸に探索フォトン数を取って，放射フォトン数と探索フォトン数を変化させたグラフを Fig. 5 に示す．

Fig. 5 を見ると，探索フォトン数の増加により，処理時間と処理フォトン数が飛躍的に増加していることがわかる．この原因として，次のようなことが考えられる．2 章 2 節で述べたように，フォトン探索では，探索フォトン数分のフォトンが見つかると，探索半径を探索済みフォトンに合わせて，縮めていくことができる．この時，探索フォトン数が大きくなると，探索済みフォトンリストが埋まるまでの処理フォトン数が増加する．また，多くのフォトンをリストに入れることで，入れ替え処理の発生するフォトンの候補が増え，処理フォトン数が増加しやすくなる．一方で，放射フォトン数が増加しても，探索フォトン数が少ないと，KD 木の大部分が枝刈りされ，処理フォトン数はあまり変わらない．もう一つの理由として，探索フォトン数が増加することで，探索済みフォトンリストから最大距離のフォトンを探索する処理時間が長くなる，ということが考えられる．この処理は，探索フォトン数 N に対し，時間計算量が $O(N)$ で増加する．一方で，KD 木では放射フォトン数 p に対して，距離計算などの処理の時間計算量は $O(\log p)$ で増加する．以上から，探索フォトン数は，他のパラメータと比較して処理時間への影響が大きいといえる．

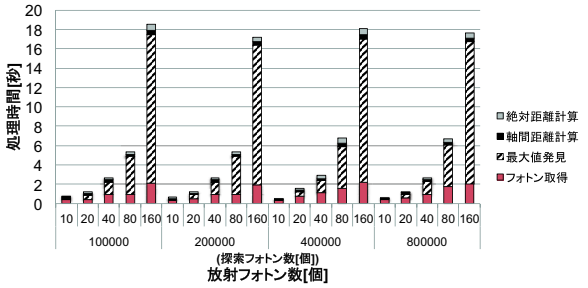

Fig.6　フォトン探索処理における処理ごとの処理時間内訳

3.3　フォトン探索処理における処理時間の内訳

フォトン探索処理法の高速化を検討するにあたり，処理時間の内訳を計測した．計測項目として，フォトン探索処理を，軸上の距離計算，絶対距離計算，フォトン取得，最大値発見，の 5 つの処理に分類した．Fig. 6 にこの結果を示す．

Fig. 6 から，処理時間の大部分が最大値発見に割かれていることがわかる．この傾向は，探索フォトン数が増加することで顕著になる．以上から，フォトンマッピング法の中で，フォトン探索処理が大きい処理時間を占めること，探索フォトン数によって，大きく処理時間が変化すること，フォトン探索処理の計算時間では，最大値発見処理の占める割合が大きいことを確認した．

4　FPGA によるフォトン探索の実装

4.1　実装の方針

フォトン探索処理の高速な実装を行うにあたり，時間計算量の大きい最大値発見処理の高速化と，単位時間当たりに処理可能なフォトンの数で表されるスループットの向上を実装の方針とした．最大値発見処理を高速化することで，1 フォトンあたりの処理時間の削減が期待できる．一方で，スループットを向上することで，単位時間あたりの処理フォトン数を増大できる．

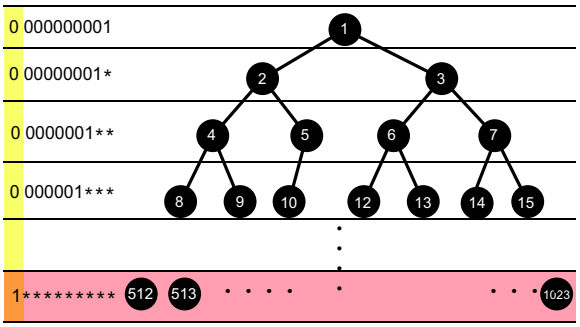

Fig.7　FPGA 上での KD 木

4.2　FPGA 上での KD 木の扱い

本研究で提案する手法では，FPGA 上では，フォトン探索のみを行い，フォトンの放射，KD 木の構築，画素計算などの処理は FPGA に接続するホスト PC で行うこととする．ホスト PC 上で KD 木の構築が完了すると，ルートノードから順にフォトンデータを FPGA に送信し，FPGA 側で受信した順番で KD 木を再構築する．KD 木の再構築後，ホスト PC から FPGA にクエリが送信され，FPGA 内でクエリを元に近傍フォトンを探索し，近傍フォトンをホスト PC に送信する．この時，近傍フォトンのデータを送信するのではなく，KD 木上のフォトンのインデックスをホスト PC に送信する．インデックスからはホスト PC 側の KD 木上からフォトンを，特定できるため，転送するデータ量を削減できる．今回実装したプログラムでは，フォトンが放射される空間として 1000 × 1000 × 1000 の空間を想定し，座標情報を 30bit(10bit×3) で保持している．これに，分割軸情報として 2bit を加えたものが 1 フォトンのデータ量となる．フォトンデータは，FPGA の組み込みメモリ (デュアルポート BlockRAM) に格納する．デュアルポート BlockRAM は，一度に 2 つのアドレスに対しデータの読み書きが可能なメモリである．格納方法として，アドレス i のフォトンの左の子フォトンを $i \times 2$ のアドレスに，右の子フォトンを $i \times 2 + 1$ のアドレスに格納する方法を取った．また，KD 木へアクセスする bit 幅を Fig. 7 に示すように KD 木の高さで固定した．このようにすることで，アクセスするインデックス値の最上位 bit を確認することでアクセスするフォトンがリーフノードであるかどうかを判断できる．

4.3　フォトン探索モジュールの設計

実装するハードウェアとして，Fig. 8 の構成を取った．主な構成要素である，TreeModule，QueryModule，SearchModule について説明する．

TreeModule は，KD 木が格納された組み込みメモリを持つモジュールである．初期動作として，フォトンデータを受信し，組み込みメモリ上に KD 木を構築する．外部からフォトンの読み出し要求と，読み出しアドレスを受信すると，次のクロックサイクルで，フォトンデータ SeachModule に対して送信する．

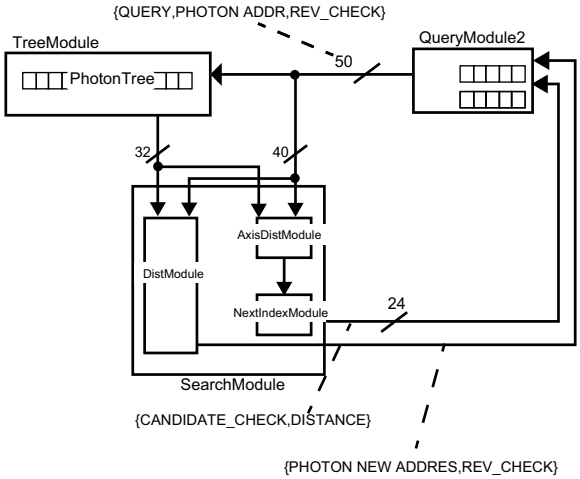

Fig.8　フォトン探索の FPGA 実装

QueryModule は，1 つのクエリ処理を管理するコントローラモジュールである．内部に，クエリ情報と，クエリに対応した探索済みフォトンリスト，探索済み距離リスト，探索半径を持つ．探索済みフォトンリストはフォトンのアドレスを，探索済み距離リストはフォトンとクエリとの距離を保存する組み込みメモリであり，それぞれ探索フォトン数分の深さを持つ．QueryModule は，決められた周期で TreeModule に対しフォトンの読み出し要求を発行すると同時に，クエリデータ，探索範囲，KD 木の探索方向を SearchModule に送信し，探索結果としてフォトンアドレスと，探索範囲にフォトンがあったかどうか，フォトンとクエリとの距離，次の探索で送信するフォトンアドレス，KD 木の探索方向を得る．フォトンが探索内にあった場合は，フォトンとクエリとの距離を探索済み距離リストに追加する．探索済み距離リストに探索フォトン数分の情報が格納されている場合，探索済み距離リストから最大値を発見し，次の探索範囲に設定し，次の読み出し要求を行う．

SearchModule は，QueryModule から送られたクエリと，TreeModule から送られたフォトンを用いてフォトンの探索を行うモジュールである．フォトンの探索は，AxisDistModule，NextIndexModule，DistModule の 3 つの演算モジュールを通じて行う．AxisDistModule は，フォトンとクエリの軸上の距離を計算するモジュールである．NextIndexModule は，AxisDistModule の結果と，受信したフォトンアドレス，探索方向から次にアクセスするフォトンアドレスと，次の探索時の探索方向を計算するモジュールである．DistModule は，フォトンとクエリの絶対距離を計算するモジュールである．この距離と，受信した探索範囲を元に，フォトンが探索範囲内にあるかどうかの判定も行う．DistModule は，AxisModule と NextIndexModule と独立に動作するため並行して計算を行う．絶対距離計算は，その他の計算に比べて演算時間が長いため，演算結果は，NextIndexModule，DistModule とで独立して行う．

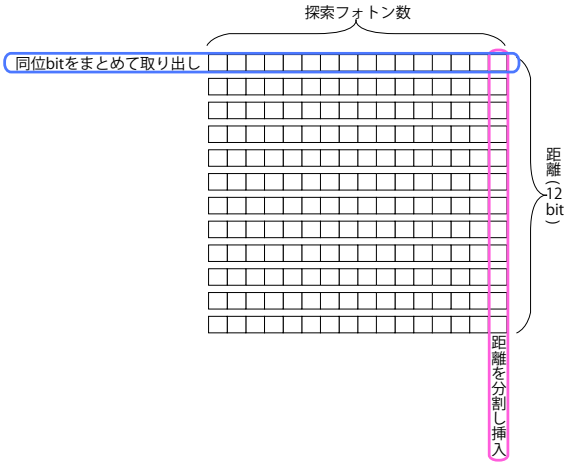

Fig.9　距離のメモリへの挿入

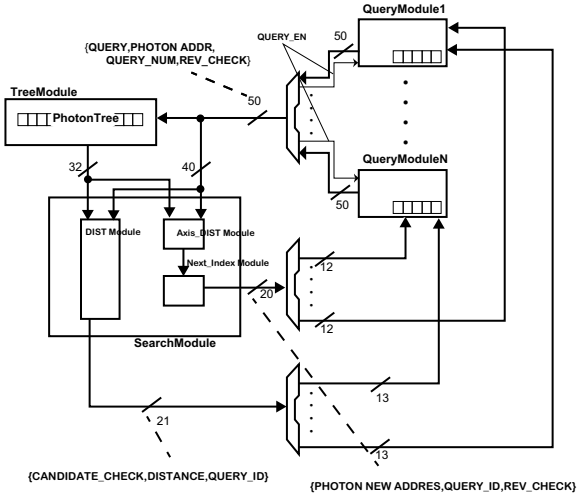

Fig.10　QueryModule の多重接続

4.4　最大値発見の高速化

QueryModule 内で行われる距離の最大値発見は，探索フォトン数分の距離の比較を行わなければならない．探索フォトン数は，高画質な画像の生成には 100 個以上必要とされるため，大きなボトルネックとなる可能性がある．探索フォトン数が 100 個と大きい値になるのに対し，格納される距離の bit 幅は $1000 \times 1000 \times 1000$ の空間では 12bit，$10000 \times 10000 \times 10000$ の空間では 15bit と値の伸びが緩やかである．そこで，距離を比較するのではなく，各距離の同位 bit をフィルターに通すことで，最大値を発見する手法を用いた．まず，得られた距離の各 bit を Fig. 9 のようにメモリの格段に挿入する．このようにデータを格納することでメモリから，同位 bit を一度に取り出すことができる．次に，12bit の最大値を格納する bit 列と 12bit の全て 1 で初期化したフィルタを用意し，次の手順でフィルタを更新していく．

1. メモリから n 段目の bit 列を取り出す．最初の操作では，最上位 bit 列を取り出す．
2. 取り出した bit 列とフィルタの論理積をとる．
3. 論理積が全て 0 で無ければ，フィルタを論理積で置き換える．全て 0 であれば，フィルタを更新しない．
4. 取り出した論理積のビット論理和が 0 で無ければ，1 を，0 であれば 0 を最大値 n 桁目の値とする
5. n を 1 増やす．
6. 1 から 5 をメモリの最下段の bit 列を取り出すまで繰り返す．

2. の操作によって，各 bit 列から 1 である bit を抽出できるため，メモリの最下段まで操作を行った時のフィルタの内 bit が 1 となっている箇所が最大距離の格納されている位置であることがわかる．また，4. の操作によって，フィルタを更新しながら最大値を導出できる．これにより，探索フォトン数によらず，最大値を発見することが可能になる．

4.5　モジュールの多重接続によるメモリアクセスの最大化

QueryModule は，SearchModule からの探索結果や，QueryModule 内での最大値発見処理の結果を元に次の探索を開始するため，各処理が行われている間 TreeModule へのメモリアクセスが行われない時間が発生する．そこで，QueryModule を複数 TreeModule に接続し，各 QueryModule から連続にフォトンを読み出すことで，メモリに対するアクセスを最大化した．接続する QueryModule 数をフォトンの読み出し要求間のクロックサイクル数と同じにすることでメモリアクセスの空白を全て埋めることが可能になる．このハードウェア構成を Fig. 10 に示す．TreeModule にアクセスする QueryModule は，マルチプレクサを介して 1 クロックサイクルごとに順番に割り当てる．SearchModule では，QueryModule から，探索に用いるデータと共に，QueryModule の ID を受信し探索結果と共に送り返すことで，どの QueryModule の探索結果であるかを判断する．この構成により 1 クロックサイクルごとに 1 つのフォトンを読み出すことができる．

また，KD 木にはデュアルポート BlockRAM を用いているため，一度に 2 つのフォトン取り出しが可能である．そこで，複数の QueryModule，SearchModule をまとめ，TreeModule に対して 2 つ接続した．これにより，1 クロックサイクルごとに 2 つのフォトン読み出しが可能となり大きなフォトン数になった場合にも高いスループットを維持することが可能となる．

Table2　資源使用量

SLICES		LUTS		BlockRAM	
Used	Usable	Used	Usable	Used	Usaable
7718(1.29[%])	595200	10177(3.41[%])	297600	45	1064

5　評価

KD 木の探索を行うハードウェアを Verilog-HDL で実装し，ISE13.1 を用いて Virtex6(XC6VSX475T) を対象とした合成を行った．

Virtex6(XC6VSX475T) は 36bit 幅 ×1024 エントリの 36Kb の BlockRAM が 1,064 個 (38,304Kb 相当) 組み込まれており，現在流通している FPGA の中では比較的多数のフォトンを格納できる FPGA である．

TreeModule に対するクエリの投入から次のクエリの投入までに必要となるクロックサイクル数は 21 であることから，TreeModule の 1 ポートあたり 21 個の QueryModule を接続することでメモリアクセスを最大化し，1 クロックあたり 2 個のフォトン読み出しが可能となる．よって，1 つの TreeModule と，42 個の QueryModule, 2 つの SearchModule を載せた回路を合成した．

ハードウェア全体の合成結果を Table 2 に示す．

この時，最大動作周波数は 215MHz であった．1 クロックあたりに取り出せるフォトンの個数が 2 個であることから, スループット (クロックサイクル×動作周波数) は，520M[OPS] となる．

この結果を，C 言語で実装した KD 木探索のスループットと比較した．放射フォトン数 1023 および探索フォトン数 10 をパラメータとして与え，1M クエリの処理に要する時間を計測した．計測環境は Table 1 を用い，コンパイラオプション-O3 で計測したところ，処理時間 1.63 秒，処理フォトン数 78568098 個，スループット約 48M[OPS] という結果が得られた．この結果から，FPGA を用いた実行は，C 言語プログラムによる実行に比べ，約 11 倍のスループットが得られることが確認された．また，Table 2 に示すように，実装したハードウェアが必要とするロジック資源は SLICES，LUTS 共に比較的少量である．このことから，本実装にはさらにスループットを向上させる余地があると考えられる．例えば，TreeModule を含めた全ての回路を複数積載するといったアプローチや大きなフォトン数に効率的に対応するために，一部のフォトンをスライス上に配置するといった木構造の工夫などが挙げられる．

6　まとめと今後の展望

本研究報告では，高品質な CG を実現するフォトンマッピング法を FPGA を用いて高速に計算する手法について述べた．FPGA 実装に先立ち，フォトンマッピング法に関するパラメータと計算時間の関係を実行プロファイルをもとに解析し，フォトン探索処理が処理時間の大部分を占めることを明らかにした．また，探索済みのフォトンリストに関する処理と探索フォトン数が，計算時間への影響が大きいことを確認した．

フォトンの探索処理の FPGA 実装においては，フォトン探索処理の内計算時間の長い最大値発見処理について，bit 演算を用いることによる高速化を検討した．また，複数のクエリを並列処理することよるスループットの向上を試みた．これらの実装から，組込みメモリへのアクセス時間の空白を埋め，1 クロックサイクルあたり 2 つのフォトンの読み出しが可能となり，C 言語実装と比較して約 11 倍のスループットが得られることを確認した．

参考文献

1) Henrik Wann Jensen. Global illumination using photon maps. In *Proceedings of the eurographics workshop on Rendering techniques '96*, pp. 21–30, London, UK, 1996. Springer-Verlag.
2) T. Purcell, C. Donner, M. Cammarano, H.J. Jensen, and P. Hanrahan. Photon mapping on programmable graphics hardware. *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, Vol. 2003, pp. 41–50, 2003.
3) 大西信寛, 鎌田俊昭, 西川由理, 設楽明宏, 吉見真聡, 藤代一成, 天野英晴. Cell broadband engine を用いた photon mapping の実装と評価 (2010 年並列/分散/協調処理に関する『金沢』サマー・ワークショップ swopp2010). 電子情報通信学会技術研究報告. CPSY, コンピュータシステム, Vol. 110, No. 167, pp. 19–24, 2010-07-28.
4) B. D. Larsen. Real-time global illumination by simulating photon mapping.
5) Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2009.