2008－SLDM－133（5）
2008/1/16

リコンフィギャラブルマシンSRC-6におけるDMA転送の最適化手法

志田さや香† 柴田裕一郎† 小栗 清†

† 長崎大学工学部情報システム工学科,
852-8521 長崎県長崎市文教町 1-14
E-mail: †{shida,shibata,oguri}@pca.cis.nagasaki-u.ac.jp

あらまし リコンフィギャラブルマシンでは，CPU と FPGA 間のデータ転送がボトルネックになることが多く，転送時間の短縮が求められている．このため SRC-6 の DMA 転送は複数のオンボードメモリにインタリーブしながらストリーミング処理することが可能となっている．しかし，FPGA の資源制約が大きなアプリケーションでは，インタリーブの前処理として CPU 上でのデータ並べ換えを行う必要がある．本稿では，そのオーバーヘッドを評価し，トレードオフポイントを明らかにした．その結果，1 データ列あたり 150 KB 以下の演算を扱う場合，CPU 上で並べ換えをした後インタリーブしながらストリーミング処理を行うことで速度向上が実現できることを示した．

キーワード FPGA，DMA 転送，インタリーブ，レイテンシ隠蔽

An optimization method of DMA transfer for the SRC-6 reconfigurable machine

Sayaka SHIDA†, Yuichiro SHIBATA†, and Kiyoshi OGURI†

† Department of Computer and Information Sciences, Nagasaki University,
Bunkyo-machi 1-14, Nagasaki-shi, Nagasaki Prefecture, 852-8521 Japan
E-mail: †{shida,shibata,oguri}@pca.cis.nagasaki-u.ac.jp

Abstract DMA transfer between a CPU and an FPGA often becomes a bottleneck of current reconfigurable machines. To mitigate this problem, the DMA transfer of SRC-6 supports streaming processing with a on-board memory interleave. However, as a preprocessing of the interleave, the CPU must reorder the data for applications with severe FPGA resource constraints. This paper empirically evaluates this overhead to reveal the trade-off point. The results show that the speedup is achieved by interleaved streaming DMA when FPGAs treat 150 KB or lower of data per stream.

Key words FPGA, DMA transfer, interleave, latency hiding

1. はじめに

近年，FPGA(Field Programmable Gate Array) のようなリコンフィギャラブルデバイスは高速なマイクロプロセッサと組み合わせることにより，高性能な汎用コンピューティングシステムの一部を構成している．現在，数多くのハイエンドな商用マシンが登場しており [1]～[4]，科学技術シミュレーションや暗号などの分野で用いられている [5]～[7].

FPGA の高い柔軟性はアプリケーションの効率的な実装を実現することが可能であるが，実装法によっては，最大性能の 1/1000 程度の性能しか達成できないことがあることも指摘されている [8]．したがって，リコンフィギャラブルマシンの真価を引き出すには，幅広い分野における実用的アプリケーションの実装経験を蓄積し，アプリケーション実装法の指針を確立する必要がある．

またリコンフィギャラブルマシンでは，マイクロプロセッサと FPGA ボード間の DMA データ転送がボトルネックとなる場合が多い [9], [10]．そこで，これらのマシンでは DMA 転送を高速化あるいは隠蔽化するための工夫が行われている．例えば，SRC 社の SRC-6 [2] では DMA 転送の機能の一つとして複数バンクのオンボードメモリにインタリーブしながらストリーミング処理を行うことが可能である．しかし，インタリーブを行うためには，CPU 側でデータ列の並び換えを行う必要がある．本稿では，CPU 側で転送データの並び換えを行うことでオンボードメモリの並列性を生かす最適化手法を提案し，その効果を評価する．

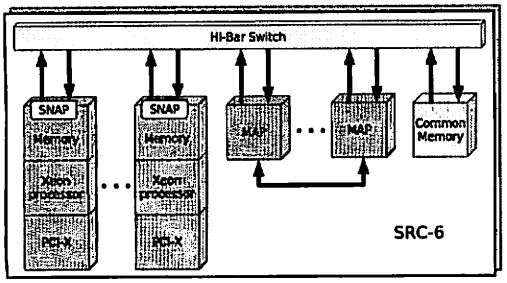

図 1 SRC-6 の構成
Fig. 1 Structure of the SRC-6

George Washington 大学の El-Araby らはリコンフィギャラブルマシンにおけるデータ転送時間と演算時間の関係から速度向上率を定式化している [10]. ストリーミング処理による DMA 転送と演算を並列化することができ，かつ，DMA 転送の入出力が同時に行えない SRC-6 の場合，DMA 転送における入力と出力のデータ量が等しい時に最も良い性能であることが示されている．

この研究では DMA ストリーミング転送と演算の処理速度の関係から最適化を行っている．これに対して，本研究ではリコンフィギャラブルマシンに高性能マイクロプロセッサが用いられていることを基に，DMA ストリーミング転送の最適化方法について議論する．

本稿の構成は次の通りである．続く第 2 章では SRC-6 の構成と DMA ストリーミングの仕組みを紹介する．次に第 3 章ではこれまでの研究成果と新たに出現した問題点について述べる．その後第 4 章では DMA ストリーミングを用いた設計手法について述べ，続く第 5 章では DMA ストリーミングと並び換え処理速度の関係について評価する．最後に第 6 章で本稿の結論を述べる．

2. リコンフィギャラブルマシン SRC-6

本研究で使用する SRC 社の SRC-6 は，コプロセッサ型のリコンフィギャラブルマシン [1] で，アーキテクチャのベースは Intel 社の Xeon プロセッサである．プロセッサの動作周波数は 2.8 GHz，メインメモリの容量は 3 GB である．図 1 に SRC-6 の構成を示す．FPGA は図 1 の MAP(Multi-Adaptive Processor) 内に配置されており，MAP は Hi-Bar スイッチと呼ばれるクロスバスイッチに接続されている．また Hi-Bar スイッチには，内部接続を行うインタフェースである SNAP を介してメモリとマイクロプロセッサ，PCI-X バスが接続されている．そのため，SRC-6 はマルチプロセッサ構成をとることも可能である．加えて，グローバルな共有メモリも Hi-Bar スイッチに接続されている．

SRC-6 に搭載されている FPGA は Xilinx 社の Virtex II XC2V6000 であり，周波数は 100 MHz 固定で動作している．また，MAP にはオンボードメモリ (OBM) として 4 MB の SRAM が 6 バンク搭載されており，ホスト側から受けとった

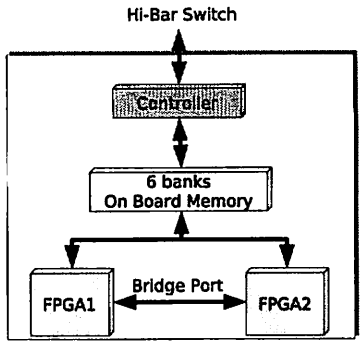

図 2 MAP の構成
Fig. 2 Structure of MAP

値はこの SRAM に保持される．メインメモリとオンボードメモリ間のアクセスは DMA 転送で行われる．

図 2 に MAP の構成を示す．MAP 内には 3 個の FPGA が搭載されており，1 個は Controller として，他の 2 個はユーザが使用するユーザチップとして用いられる．まず，Controller を介してユーザチップで利用するデータが DMA 転送で OBM に格納される．全てのデータがマッピングされると，計算が開始される．2 個の FPGA をどのように利用するかは，ユーザが決定できる．

OBM へは，通常，主となる FPGA，つまり FPGA1 のみがアクセス権を持っている．このアクセス権はバンク毎に切り替えることが可能である．しかし，アクセス権を切り替えると，再度変更するまで FPGA1 からアクセスすることは不可能である．また，2 つの FPGA 間には Bridge Port と呼ばれる双方向のポートが 3 ポート用意されており，このポートを通じてアクセス権の変更やデータのやり取りが行われる．全ての計算が終了すると，Controller を介して，OBM から DMA でメインメモリにデータが渡される．これが処理の一連の流れとなっている．

OBM の各バンクとユーザチップは双方向ポートで接続されている．つまり，各バンクに同時にアクセスすることが可能である．しかしながら，ポート数は各々1 ポートであるため，読み込みと書き込みを同時に行うことはできない．このため，OBM へのデータの割り当ては読み込み/書き込みの競合のために生じる遅延を考慮する必要がある．

2.1 DMA ストリーミング

DMA ストリーミングとは，OBM に全てのデータを転送した後に演算を開始する通常の DMA とは異なり，データ転送と演算を同時に行う方法である．SRC-6 には，CPU から MAP に DMA 転送されるデータ列からデータを 1 個ずつ取り出す関数 stream_dma_cpu と，1 個のデータ列から 2 個ずつデータを取り出す関数 stream_dma_cpu_dual が用意されている．どちらの場合も，OBM のバンクをバッファとして用いることで実現している．

図 3 に DMA ストリーミングの仕組みを示す．ここで示す

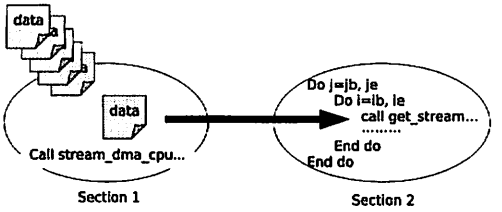

図 3 DMA ストリーミングの仕組み
Fig. 3 Streaming DMA

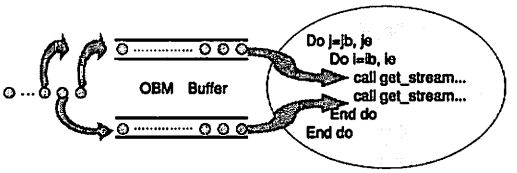

図 4 dual DMA ストリーミングの仕組み
Fig. 4 Dual streaming DMA

Section とは，互いに独立した制御の下で並列に動作できる FPGA 内部の処理単位のことである．Section 1 では OBM のインタフェースを使ってデータを連続転送する．Section 2 では到着したデータを get_stream 関数を用いて呼び出し，演算を行う．各 Section は並列に処理されるため，演算処理を行っている間にデータが連続転送されている．このことにより，DMA 転送待ち時間が短縮され，システム全体の処理時間の短縮につながる．また，Section 2 のループの処理速度に合わせて DMA ストリーミングの速度も変化するため，連続転送されるデータが失われることはない．

図 4 に dual DMA ストリーミングの仕組みを示す．図 3 では，1 バンクをバッファとして用いることで，データ転送を行なっていたが，図 4 では OBM の 2 バンクをバッファとして用いている．1 個のデータ列から 2 バンクへインタリーブを行うことで，演算処理の際には各バンクから 1 個ずつ，計 2 個のデータを同時に取り出すことができる．

3. これまでの研究成果と問題

これまで我々は，海洋大循環モデルの 1 つである Parallel Ocean Program (POP) 内の，順圧方程式で用いる 9 つの参照点から係数行列を導き出す barotropic operator 関数を実装することで高速化手法の検討を行ってきた [11]．図 5 に barotropic operator 関数のアルゴリズム，図 6 に barotropic operator 関数の性能評価を示す．実装では 2 個の FPGA を用いて並列処理を行い (Arch.1–4)，OBM へのアクセス回数を削減する目的で FPGA 内部メモリ (Arch.2, Arch.4) を利用した．このような FPGA を用いた実装を行う際に問題となったのが DMA 転送である．DMA 転送中は FPGA での演算を並列に行うことができず，これがボトルネックとなるために，入力データのうちデータ列 *A0* の転送方式を Single DMA ストリーミングに変更し，

```
Initial values AX(:,:,bid) = 0.0_8
do j=jb, je
  do i=ib, ie
    AX(i,j,bid) = A0 (i  ,j  ,bid) * X(i  ,j  ,bid) + &
                  AN (i  ,j  ,bid) * X(i  ,j+1,bid) + &
                  AN (i  ,j-1,bid) * X(i  ,j-1,bid) + &
                  AE (i  ,j  ,bid) * X(i+1,j  ,bid) + &
                  AE (i-1,j  ,bid) * X(i-1,j  ,bid) + &
                  ANE(i  ,j  ,bid) * X(i+1,j+1,bid) + &
                  ANE(i  ,j-1,bid) * X(i+1,j-1,bid) + &
                  ANE(i-1,j  ,bid) * X(i-1,j+1,bid) + &
                  ANE(i-1,j-1,bid) * X(i-1,j-1,bid)
  end do
end do
```

図 5 barotropic operator 関数のアルゴリズム
Fig. 5 Algorithm of the barotropic operator function

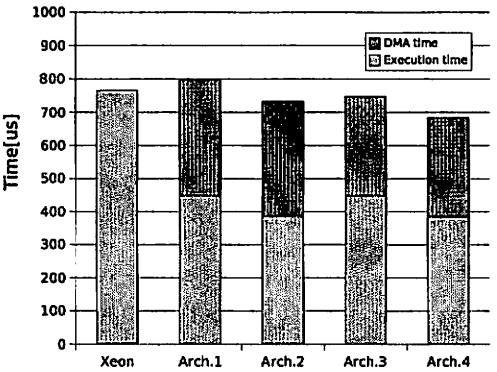

図 6 barotropic operator 関数の性能評価
Fig. 6 Performance results of the barotropic operator function

その他のデータ列は通常の DMA 転送を用いた (Arch.3–4)．その結果，DMA 転送待ち時間が短縮され演算時間にストリーミング処理が含まれたことによって，関数全体の処理時間が削減された．このことからさらに，データを一度に 2 度用いることが可能な Dual DMA ストリーミングを用いることを考えた．1 個のデータ列から一度に 2 個のデータを取るようにすることで並列度は高くなるが，その並列度を満たすためには回路規模が倍増してしまう．現時点で FPGA の使用率は 60%程度であることから，現在のシステムでは使用率を倍増させることが困難である．そのため，回路面積が増加するような改良ではなく，Dual DMA ストリーミングを用いて異なる 2 個のデータ列から同時に値を取り出すことができるようにデータ列 *A0* 及び *AE* の並べ換えを試みることとした．

全体の処理時間を比較するにあたって，2 種類の DMA ストリーミングを用いた場合のどちらが高速であるか検討する必要がある．例えば，今回用いた barotropic operator 関数において，データ列は入出力合わせて 6 個用いている．これらは DMA ストリーミングを利用しない場合，6 回に分けて DMA 転送される．Dual DMA ストリーミングを利用する場合には CPU 上での並べ換え処理を行い，データ列 4 個の DMA 転送と 1 回の Dual DMA ストリーミングを行う．一方 Single DMA ストリーミングを用いると，CPU 上での並べ換え処理がない代わ

りにデータ列5個を通常のDMA転送したのち，1回のSingle DMAストリーミングを行う必要がある．

しかし，DMAストリーミングを用いた実装結果はデータ列のサイズが問題となる可能性がある．また，並び換え処理についても同様と言える．そこで本稿では，データ列のサイズによる並べ換え処理速度について検討を行った．

4. 設　　計

SRCコンパイラには，高水準言語であるFORTRANやCのプログラムから，自動的にハードウェアを設計する機能が搭載されており，プログラムのループ部分がパイプライン化される．さらに，データ転送やOBMへのデータ配置に関してもFORTRANやCのコードで記述することができる．このとき，プログラムの記述方法によっては，OBMのバンクアクセスなどが変化するためループ1回あたりのクロックサイクル数(clocks per iteration)が異なり，性能に影響を及ぼす．つまり，自動生成による設計を利用する際であってもハードウェアの規模やメモリバンクの並列度，パイプラインピッチやclocks per iterationを考慮したプログラムを設計する必要がある．今回はその中でも，データ転送，特にDMAストリーミングに焦点をあてる．

図6ではSingle DMAストリーミングによる演算時間の遅延は見られなかった．また，ストリーミング処理が演算時間に含まれていることから，本稿ではDMAストリーミングによる演算時間の遅延はなく，ストリーミング転送の時間は完全に演算時間に隠蔽されると仮定して，通常のDMA転送と並べ換え処理について評価を行う．データ量の等しいデータ列2個について以下の2つの場合について実装を行った．

- 通常のDMA転送及びSingle DMAストリーミングを利用する
- CPU上での並び換え処理を行った後，Dual DMAストリーミングを利用する

DMAストリーミングは，CPUからFPGAへデータ転送を行うポートが1本しかないため，同時に2つのDMAストリーミングを行うことができない．そこでSingle DMAストリーミングを用いた設計では，一方のデータ列はSingle DMAストリーミングを用いて転送し，もう一方のデータ列は通常のDMA転送を用いてデータ転送を行うこととした．実際の処理の流れとしては，まず通常のDMA転送によってデータを転送し，その後演算処理と並列にSingle DMAストリーミングを用いることとなる．

一方，Dual DMAストリーミングでは，OBMを2バンク使用することでインタリーブする形となる．これを踏まえて，アプリケーション上では2個のデータ列として扱われているものを，Dual DMAストリーミングする前にCPU側で1個のデータ列に結合させておく．この際に2個のデータ列の値を交互に取り出しながら1個のデータ列に並べ換える．その結果，各バンクにはデータ列の偶数番目だけが1バンクに，もう一方のバンクにはデータ列の奇数番目だけが格納される．つまり，CPUで並び換えられたデータ列は，バッファとして用いられているOBMに転送された際に元のデータ列の並びで各バンクに格納することができる．そのため，演算処理を行う時には2個のデータ列からそれぞれ1個ずつ値を取り出すことと等しくなる．Dual DMAストリーミングを用いた処理の流れは，まずCPU上で2個のデータ列を1個のデータ列に並べ換え，演算処理と並列にDual DMAストリーミングを用いた．

5. 評　　価

本章では，2種類のDMAストリーミング転送についての評価とPOP内の2つの関数を用いた性能評価を行う．

5.1 DMAストリーミングと並べ換え

第3章で述べた通り，DMAストリーミングの違いによる演算処理の低下はほとんど見られなかった．したがって，あるデータ列のDMA転送速度とそのデータ列2個を並べ換える処理の速度比較を行なった．並び換え処理の評価環境はSRC-6のホストプロセッサである2.8GHz Xeonプロセッサを用い，コンパイラにはifort 8.1を使用している．

DMA転送と並び換え処理の実行時間を図7及び図8に示す．図8は図7の一部を拡大したグラフである．1バンクあたり4MBのデータを格納することができるため，1Byteから4MBのデータについて評価を行った．DMA転送に関しては，転送時間がデータ量に比例している．一方，並び換え処理は約500KBまではデータ量の二乗に従い，その後はデータ量に比例して処理時間がかかっている．データ列サイズが約500KBを越えたあたりから処理時間がデータサイズに比例しているのは，L2キャッシュサイズが1MBであり並べ換えに用いる総データ量がキャッシュサイズを越えてしまうため，ハードウェアプリフェッチの効果が得られなくなっていると考えられる．一方，通常のDMA転送はデータ転送量のみに依存してることが分かる．

図8より，およそ1データ列あたり150KB以下の並び換えを行う場合にDual DMAストリーミングが有効であることが示された．POPでは1データ列の大きさをユーザが指定することによって，計算粒度の変更が可能であるため，データ列を150KB以下にすることでDual DMAストリーミングを効果を十分に引き出すことができると言える．

5.2 barotropic operator

5.1節の結果を基にbarotropic operator関数に対して，Dual DMAストリーミングを用いて実装を行った．図9に実行結果を示す．Execution timeにDMAストリーミング時間も含まれており，DMA timeは通常のDMA転送時間のみを表している．また，Reorderingは CPU上での並べ換え処理時間である．Arch. 5及びArch. 6では入力データ列の一部をDual DMAストリーミングを用いて転送し，さらにArch. 6においては出力データをSingle DMAストリーミングを用いて転送した．その結果，Single DMAストリーミング同様，DMA転送時間が短縮され，DMAストリーミングを含む演算時間はほとんど変化がなかった．さらに，Single DMAストリーミングと比較すると，Dual DMAストリーミングの方が高速であるという結果に至った．また，入力だけでなく出力データにもDMAスト

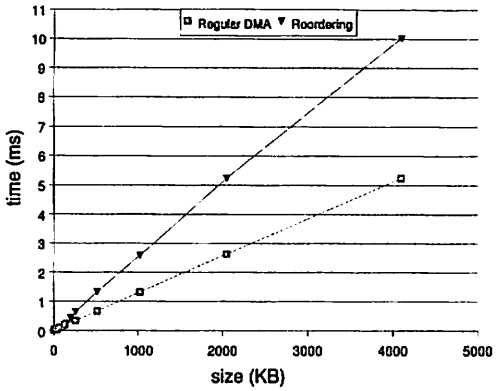

図 7　DMA 転送と並び換え処理 (1)
Fig. 7　DMA transfer and reordering (1).

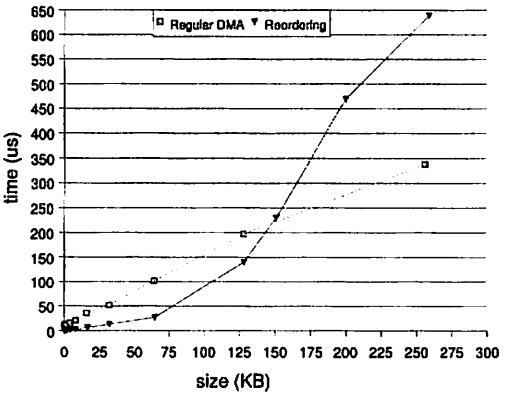

図 8　DMA 転送と並び換え処理 (2)
Fig. 8　DMA transfer and reordering (2).

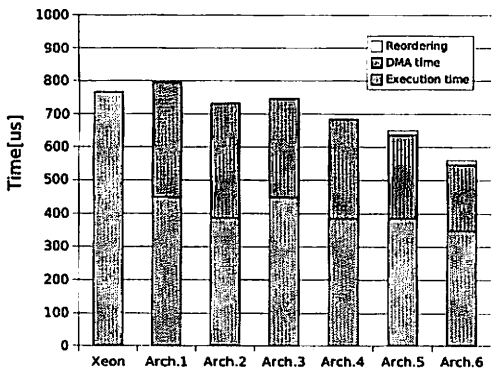

図 9　Dual DMA ストリーミングを用いた性能評価
Fig. 9　Performance results using Dual DMA streaming

リーミングを用いることで，さらに DMA 転送及び演算の双方で処理時間が削減された．

```
GRADX = 0.0_8
GRADY = 0.0_8

do j=1,ny_block-1
  do i=1,nx_block-1
    GRADX(i,j) = DXUR(i,j,bid)*p5*(F(i+1,j+1) - F(i,j) -
                                   F(i  ,j+1) + F(i+1,j))
    GRADY(i,j) = DYUR(i,j,bid)*p5*(F(i+1,j+1) - F(i,j) +
                                   F(i  ,j+1) - F(i+1,j))
  end do
end do

where (k > KMU(:,:,bid))
  GRADX = 0.0_8
  GRADY = 0.0_8
endwhere
```

図 10　grad のアルゴリズム
Fig. 10　Alrotithm of the grad

5.3 grad

さらに POP 内の別関数 grad に対して DMA ストリーミングを用いた評価を行った．この関数では，時間に基づくエネルギーの各方向における勾配を求めている．図 10 に grad 関数のアルゴリズムを示す．本稿では $DXUR$ 及び $DXYR$ のデータ列に DMA ストリーミングを用いて処理を行い，その他のデータ列は通常の DMA 転送を利用してデータ転送を行った．grad 関数のアルゴリズムは 3 つのループ処理で構成されている．SRC コンパイラはこれらをそれぞれパイプラインとして処理をする．DMA ストリーミングは 2 つ目のループ処理と並列にデータ転送を行うことになる．また，FPGA は 2 個使用することが可能であるが，grad 関数の処理はそれほど大きなものではないため，1 個の FPGA のみを用いて処理を行っている．

図 11 に grad の実行結果を示す．SRC-6 のホストプロセッサである Xeon を用いてソフトウェア実行した結果との比較を行った．grad 関数では一部，32 ビット整数型の配列を用いる必要がある一方で，DMA 転送はデータサイズを 64 ビットに制約しているので，MAP を用いて実装した設計において，32 ビット整数型から 64 ビット整数型への拡張を CPU 上で行ってから DMA 転送するようにした (図 11 の Extention)．その結果，整数型の拡張はマイクロ秒未満であり，正確に計測することができなかった．上記に加え，Dual DMA ストリーミングの場合は並べ換えが処理時間に含まれる．図 11 からソフトウェア実行との処理時間を比較すると，通常の DMA 転送だけを用いたものは 1.31 倍，Single DMA ストリーミングを用いると 1.37 倍，Dual DMA ストリーミングを用いた場合は 1.46 倍の速度向上を示している．MAP 上で実装した 3 つの設計手法の実行時間だけを比較すると，DMA ストリーミングによる転送量が増すほど，わずかではあるが遅くなっている．しかしながら，それ以上に 1 データ列あたりの DMA 転送にかかる時間が長かったため，全体としては速度向上を実現している．

図 11 では，入力データのみを DMA ストリーミングを用い

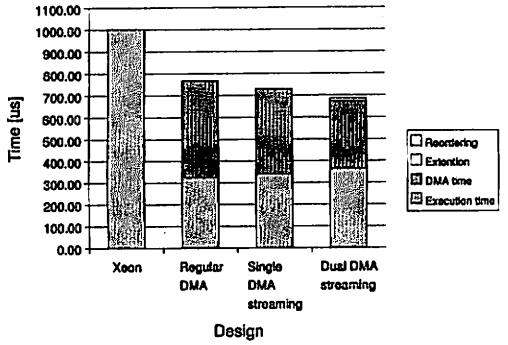

図 11 grad 関数の性能評価

Fig. 11 Performance results of the grad function

て処理している．これに加えて出力データにも DMA ストリーミングを利用することで，演算処理速度の向上が見込まれる．また，今回は FPGA1 個のみを用いて演算を行っていて回路規模も 50%前後であることから，FPGA 内及び 2 個の FPGA を用いた並列化や FPGA 内部メモリを用いて OBM へのアクセス回数を削減することも可能である．

6. おわりに

本稿では，SRC-6 のボトルネックである DMA 転送の最適化手法について述べた．2 種類の DMA ストリーミング転送について前処理を考慮して行った評価に基づいて，さらに実際のアプリケーションの一関数を用いて比較評価を行った．その結果，並べ換えを伴う Dual DMA ストリーミングの場合，データ列あたり 150 KB 以下のサイズであれば，Single DMA ストリーミングよりも効果的であることを示した．今回の実装によってボトルネックであった DMA 転送待ち時間を軽減することができた．これまでのプログラム変更や転送手法についての研究に加え，さらに転送されるデータの配置に関しても検討を行う必要がある．また，FPGA を複数個使用する場合のアプリケーションの分割方法について評価を行うことも今後の課題として挙げられる．

謝辞

本研究に対して有益なアドバイスを頂いた South Carolina 大学 Duncan Buell 博士，Subrahmanyam Bulusu 博士，George Washington 大学 Proshanta Saha 博士に感謝致します．本研究の一部は文部科学省大学教育の国際化推進プログラム（海外先進研究実践支援）及び日本学術振興会科学研究費補助金（若手研究 (B)）による．

文　献

[1] 末吉，天野（編）："リコンフィギャラブルシステム"，オーム社 (2005).
[2] "SRC Computers, Inc.", http://www.srccomp.com/ (2007).
[3] "Cray Inc.", http://www.cray.com/ (2007).
[4] "Silicon Graphics, Inc.", http://www.sgi.com/ (2007).
[5] K. Gaj, T. El-Ghazawi, A. Michalski, M. Huang, C. Shu, E. El-Araby, M. Taher and P. Sha: "Reconfigurable Computing Machines: An empirical analysis", SuperComputing 2003 (2003).
[6] R. Scrofano, M. Gokhale, F. Trouw and V. K. Prasanna: "A hardware/software approach to molecular dynamics on reconfigurable computers", Proceedings of the 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'06), Vol. 0, pp. 23–34 (2006).
[7] M. C. Smith, J. S. Vetter and S. R. Alam: "Scientific computing beyond CPUs: FPGA implementations of common scientific kernels", Proceedings of the 8th International Conference on Military and Aerospace Programmable Logic Devices (MAPLD'05) (2005).
[8] O. Mencer: "Computing with FPGAs", IEEE Symposium on Low-Power and High-Speed Chips (COOL Chips IX), pp. 327–343 (2006).
[9] M. B. Gokhale, C. D. Rickett, J. L. Tripp and C. H. Hsu: "Promises and pitfalls of reconfigurable supercomputing", Proceedings of the 2006 International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'06), pp. 11–20 (2006).
[10] E. El-Araby, M. Taher, K. Gaj, T. El-Ghazawi, D. Caliga and N. Alexandridis: "System-level parallelism and throughput optimization in designing reconfigurable computing applications", Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04), p. 136b (2004).
[11] S. Shida, Y. Shibata, K. Oguri and D. A. Buell: "Implementation of a barotropic operator for ocean model simulation using a reconfigurable machine", Proceedings of the International Conference on Field Programmable Logic and Applications (FPL'07), pp. 589–592 (2007).