

メッシュ接続FPGA アレーにおける高性能ステンシル計算

小林 諒平[†] 佐野 伸太郎^{†,☆}
高前田 (山崎) 伸也^{†,††} 吉 瀬 謙 二[†]

FPGA は高い性能を達成するカスタムのハードウェアアクセラレータを容易に構築する事を可能にする注目すべきデバイスである。本稿では、科学技術計算において重要な計算カーネルの 1 つであるステンシル計算のための、多数の小規模 FPGA を用いたスケーラブルな計算手法を提案する。本稿では 2D メッシュ型に接続された複数の FPGA で構成されるステンシル計算システムのアーキテクチャとその初期実装について述べる。隣接する FPGA 間の通信オーバーヘッドを削減するために各 FPGA における計算順序を調整することで、高い通信と計算のオーバーラップ率を実現する。まず、単一 FPGA の性能を評価したところ、0.16GHz で動作する場合には 2.37W の消費電力で 2.24GFlop/s の性能を達成することを確認した。また 1 個の FPGA の結果を元に、256 個の FPGA で構成するシステムの性能および電力あたりの性能を見積もったところ、全体で 573GFlop/s の性能を 0.94GFlop/sW の電力あたりの性能で実現できることがわかった。

High Performance Stencil Computation on Mesh Connected FPGA Arrays

RYOHEI KOBAYASHI,[†] SHINTARO SANO,[†]
SHINYA TAKAMAEDA-YAMAZAKI^{†,††} and KENJI KISE[†]

FPGA (Field Programmable Gate Array) is a remarkable device to easily develop custom hardware accelerators with higher performance. In this paper, we propose scalable stencil computation mechanism by employing multiple small FPGAs. Stencil computation is one of the most important kernels in scientific computations. This paper describes the architecture of our multi-FPGA-based stencil computation system with 2D-mesh topology and the details of primary implementation. In order to eliminate the handshaking overhead among the neighbor FPGAs, computation order is customized for each FPGA to increase the overlap rate of computations and communications. The evaluation result shows that a single FPGA node archives 2.24GFlop/s in 0.16GHz operations with 2.37W power consumption. We estimated the system performance of 256 FPGAs. The 256 FPGAs system achieves 573GFlop/s with 0.94GFlop/sW efficiency.

1. はじめに

FPGA は、ハードウェアでありながら回路構成を容易に変更できるという柔軟性を備えている。このため、計算する内容に合わせた専用のハードウェアを構成できる。近年、FPGA を用いて科学技術計算用のアクセラレータを開発する研究が盛んに行われている¹⁾²⁾³⁾。科学技術計算の重要なアプリケーションの一つであるステンシル計算⁴⁾を、FPGA を用いて高速に解く専用計算機の構築が試みられている²⁾。

我々はメニーコアプロセッサ向けの高速なシミュレーション環境として、2次元メッシュ状のFPGAアレー、ScalableCore システムを開発している⁵⁾。ScalableCore システムでは、小容量のFPGAをメッシュ状に複数接続する構成を採用する。本稿では、このScalableCore システムと同様のFPGAアレーに最適なステンシル計算手法を提案する。ステンシル計算のデータセットをブロック分割し、各FPGAに割り当てることにより、並列に処理を行なう。本手法では、高いパイプライン充填率を維持したまま、許容できるFPGA間通信のレイテンシを増やすために、各FPGAにおける計算順序を最適化する。

本稿の構成を以下に示す。2章で関連研究について述べる。3章でステンシル計算について説明する。4章で計算手法を提案する。5章で提案アーキテクチャを評価する。6章で大規模なアレーシステムの検討を行う。最後に、7章でまとめる。

[†] 東京工業大学 大学院情報理工学研究科
Graduate School of Information Science and Engineering,
Tokyo Institute of Technology

^{††} 日本学術振興会 特別研究員 (DC1)
JSPS Research Fellow

[☆] 現在、株式会社 東芝 セミコンダクター&ストレージ社
Presently with Toshiba Semiconductor & Storage Products Company

2. 関連研究

ステンシル計算をマルチコアプロセッサや GPU に最適化した事例は数多く報告されている。

Augustin ら⁶⁾ は、動作周波数 2.26GHz の Intel Xeon E5220 クラッドコアプロセッサを使用してステンシル計算を行なっている。このとき、1 コアの実行で 2.8GFlop/s を達成している。これはピーク性能 9GFlop/s の 31%にあたる。また、2つのプロセッサが持つ 8 コアにより得られた性能は、15.9GFlop/s であった。これは、ピーク性能 72.3GFlop/s の 21.8%にあたる。

Phillips ら⁷⁾ は NVIDIA TESLA C1060 GPU を使用したステンシル計算を行なっている。このとき、単一の GPU によって得られた性能は 51.2GFlop/s であった。これは倍精度演算のピーク性能の 65.6%にあたる。この計算性能は、GPU クラスタにすることでさらに低下する。例えば $256 \times 256 \times 512$ の格子サイズの場合、16GPU を使用しても計算性能はピーク性能の 42.2%に相当する。

FPGA によるステンシル計算用ハードウェアについての研究も幾つか報告されている²⁾⁸⁾。佐野ら²⁾ は、プログラム可能なシストリックアレイ構造のステンシル計算用ハードウェアを提案し、複数の FPGA (ALTERA Staratix ファミリ) を用いて試作実装している。佐野らはセルオートマトン向けに提案されているパイプラインスケジューリング法⁹⁾ を適用したアーキテクチャを実装することによって、一定のメモリ帯域のまま計算性能を向上させている。本研究とは、実装するアーキテクチャや FPGA の種類において異なっている。佐藤ら⁸⁾ は FPGA アレーを用いてポアソン方程式を演算する回路を実装している。

また、FPGA を複数接続したアレーシステムを開発したという事例も報告されている。Mencer ら¹⁾ は 512 個の FPGA を 1 次元接続することによって科学計算用のアクセラレータ CUBE を構成している。吉見ら³⁾ は CUBE を使用し、整数演算主体のアプリケーションである文字列編集距離アルゴリズムを検討している。

3. ステンシル計算

3.1 アルゴリズム

科学技術計算の分野では、ある時刻のデータセットを更新して次の時刻のデータセットを得る計算カーネルがよく用いられる。その中でも固定パターンによってデータセットの要素を更新する反復計算はステンシル計算と呼ばれる。

ステンシル計算は偏微分方程式の近似解を求める手法の一つとして用いられており、偏微分方程式が支配方程式である熱力学、流体力学、電磁気学など様々な分野においての多くの計算問題に適用できる。

図 1 は、 $x \times y$ の 2 次元格子のデータセットを使用するステンシル計算である。図 1 では、4 近傍要素の

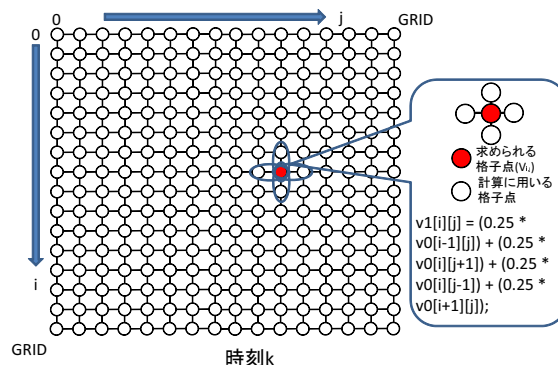


図 1 2 次元格子のデータセットを使用するステンシル計算の様子。

```

1: for (k = 0; k < Iteration; k++) {
2:   for (i = 1; i < GRID-1; i++) {
3:     for (j = 1; j < GRID-1; j++) {
4:       v1[i][j] = (0.25 * v0[i-1][j]) + (0.25 * v0[i+1][j]) + (0.25 * v0[i][j-1]) + (0.25 * v0[i][j+1]);
5:     }
6:   }
7:   for (i = 1; i < GRID-1; i++)
8:     for (j = 1; j < GRID-1; j++) v0[i][j] = v1[i][j];
9: }

```

図 2 ステンシル計算の擬似コード。

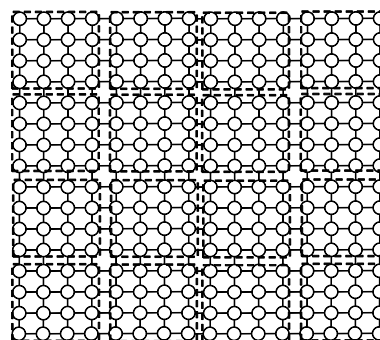


図 3 ステンシル計算の並列化のためのブロック分割。

データを参照して、次の時刻の要素を決定する。

図 2 はステンシル計算の反復解法の擬似コードである。k は時刻を表し、i, j は任意の格子点の座標 (i, j) を表す。また、座標 (i, j) の格子点を $V_n[i][j]$ と表す。ここで n はバッファ番号 0, 1 である。図 2 の 4 行目より、 $V_n[i-1][j]$, $V_n[i][j-1]$, $V_n[i][j+1]$, $V_n[i+1][j]$ のそれぞれの格子点に重み定数 0.25 を掛けて、足し合わせることで更新後の格子点 $V_n[i][j]$ が得られる。そして各格子点において時刻 k から k+1 への更新は図 2 の 7, 8 行目のループで実行される。

3.2 ブロック分割による並列計算

図 3 に示すように、ステンシル計算は複数のブロックに分割し、並列計算を進めることができる。ここで、ステンシル計算が隣接格子のデータのみを利用するため、高い並列効果を見込むことができる。ただし、ブロック境界の格子点の計算は、他のブロックの境界の格子点のデータが必要となる。並列に動作させ、それ

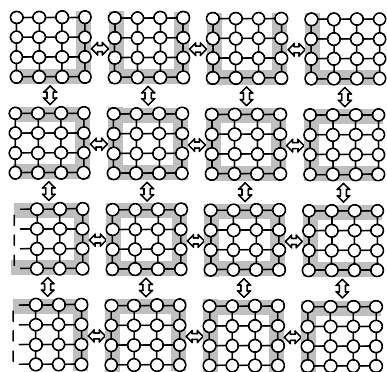


図4 FPGA アレーにマッピングされるデータセット。

それぞれのブロックを1つのFPGAに割り当てる場合、他のブロックの値を共有する必要がある。

4. FPGA アレーに適した効率的なステンシル計算手法の提案

4.1 FPGA へのデータ割り当ての方針

図4にFPGAアレーにマッピングされるデータセットを示す。ただし、実施にはより多くの要素がFPGAにマッピングされる。丸は格子点を表す。4×4の格子点の集まりは1つのFPGAを表す。矢印は通信を表す。灰色で示した部分は、他のFPGAへ通信が必要な領域を表す。

図4のように、想定するFPGAアレーシステムはタイル状に接続される構成である。またデータセットをFPGAアレーの縦・横の数に応じて分割する。分割したデータを各FPGAで計算する。計算した境界領域のデータは隣接ノードに転送される。このときの、格子点の計算順序を4.2で述べる。FPGA内部のアーキテクチャについて、4.3節で述べる。

4.2 格子点の計算順序

図5には単純化したステンシル計算のモデルを示す。上下2つのFPGAノード、FPGA(A)とFPGA(B)が同じ計算順序で計算を行うモデルを図5(a)に示す。図中の丸は1つの格子点を示し、丸中のアルファベットはFPGAのID、番号は計算順序を示す。矢印は計算の方向を示す。例えば図5(a)のFPGA(a)の計算はA0, A1, A2, A3の1行目からA0, A1, A2, A3, ..., A12, A13, A14, A15という順序で行われる。点線四角は1つのFPGAに割り当てられるデータセットを示す。実際の計算には上下左右に余分に1列のデータが必要だが、この図では省略している。すべての格子点を1度計算するための処理をイテレーションと定義する。

各FPGAはそれぞれのFPGAに割り当てられた、16点(0番から15番)の格子点の値を毎イテレーション更新する。簡単のために計算は1サイクルで終わるとする。また、同一FPGA内のデータも1サイクルで取得可能であるとする。すべてのデータの計算の完

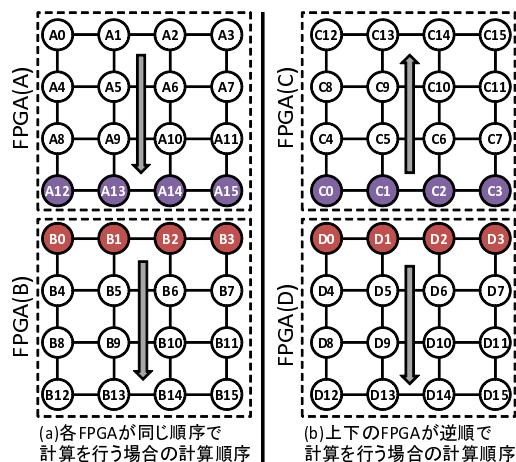


図5 FPGAが格子点を計算する順序、(b)が提案手法

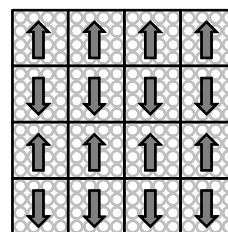


図6 FPGAアレーが格子点を計算する提案手法の順序

了後、次の時刻のイテレーションに進む。このとき、1イテレーションは16サイクルで終了する。最初のイテレーションのサイクルは0から始まり、2番目のイテレーションの開始サイクルは16から始まる。計算終了直後にデータを上書きした場合、次の計算に必要なデータが更新されている可能性がある。それらはFIFOなどによって適切に処理されていると仮定する。

図5(a)のB1格子点の計算に必要な格子点を示し、ストールを発生することなく計算を開始するために許容できる通信レイテンシについて述べる。B1を計算するためには、上下左右のA13, B5, B0, B2の格子点が必要である。このとき、通信が必要となる格子点はA13である。他の格子点は同じFPGA内にあるため通信は必要ない。格子点A13は13サイクル目に計算される。次のイテレーションのB1番の計算をストールさせないためには、14, 15, 16の3サイクルのうちに通信を終わらせなければならない。このように必要なデータの計算が終了するサイクルと計算開始のサイクルの間の時間が通信レイテンシとなる。

上下2つのFPGAノード、FPGA(C)とFPGA(D)が上下逆順で計算を行うモデルを図5(b)に示す。FPGA(C)は4行目から計算を行う。つまりC0, C1, C2, C3の4行目からC0, C1, C2, C3, ..., C12, C13, C14, C15という順序で計算される。このときD1番の計算をストールさせないためには、2~16サイクルの15サイクルの余裕がある。このように計算順序を変

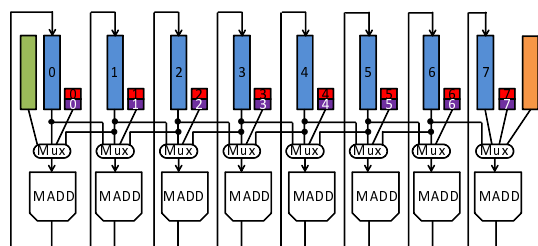


図 7 8 個の MADD を持つアーキテクチャ。

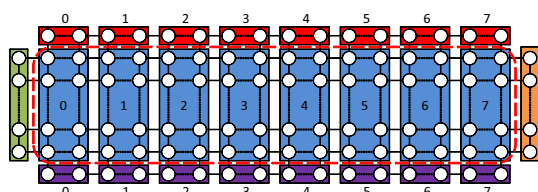


図 8 格子点と BlockRAM の関係。

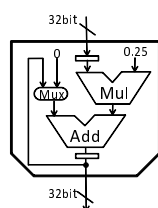


図 9 積和演算器 (MADD)。

更することで、許容できる通信レイテンシが増加する。

図 6 に計算順序を最適にした各 FPGA の計算方向 (提案手法) を示す。図 6 の四角は FPGA を表し、矢印が計算方向を表す。図 6 の 1 行目と 3 行目の FPGA が、図 5(b) の FPGA(C) と同じ計算方向である。図 6 の 2 行目と 4 行目の FPGA が、図 5(b) の FPGA(D) と同じ計算方向である。

このとき、矢印が離れていく方向での通信は図 5 で説明したとおり、約 1 イテレーションの通信レイテンシが許容できる。矢印が向き合う方向での通信も同様に約 1 イテレーションの通信レイテンシが許容できる。これは、図 5(b) の FPGA(C) と FPGA(D) を上下逆転して配置したときに、隣り合う辺の計算サイクルが等しい。

FPGA 間の左右の通信について考える。図 5 の FPGA(C) を左右に二つ並べたとき、C3 と C0 が隣り合う。このため、許容できる通信レイテンシは右側の FPGA で 12 サイクルである。これは、1 イテレーションにかかるサイクルから 1 辺の格子数を引いたサイクルに等しい。

このように、提案手法では、上下逆順に計算することで、ほぼ 1 イテレーションの処理サイクル数まで通信レイテンシを許容できる。

4.3 FPGA 内部のアーキテクチャ

図 7 に 8 個の積和演算器を持つアーキテクチャを示す。図の四角は BlockRAM^{*}である。MADD は積和演算器である。

図 7 の BlockRAM と格子点の関係を図 8 に示す。図 7 の BlockRAM の番号と図 8 の BlockRAM の番号はそれぞれ対応する。図 8 のように、各 FPGA にマッピングされたデータセットは縦方向に分割されて、破線で囲まれた番号 0~7 の各 BlockRAM に格納される。つまり 1 個の FPGA に 64×128 のデータセットがマッピングされる場合、図 8 の破線で囲まれた番号 0~7 の各 BlockRAM には 8×128 のデータセットが格納される。また、境界領域のデータ^{☆☆}はマッピングされたデータセットとは別の BlockRAM に格納する。ただし、1 個の FPGA での実装では、境界領域のデータは更新せず、常に同じ値を利用する。これは FPGA が隣接しないことから、境界領域のデータが通信されないためである。このため、境界領域の BlockRAM は入力が存在しない。

FPGA に実装する積和演算器 (MADD) のアーキテクチャを図 9 に示す。図 9 の四角はレジスタである。乗算器、加算器ともに IEEE 754 に準拠した単精度浮動小数点数演算器である。乗算器と加算器のパイプラインはそれぞれ 7 段とする。これに、2 個のレジスタを接続するので、データパスのパイプラインは 16 段となる。つまり、8 段の乗算器と 8 段の加算器を接続しているデータパス構成となっている。

MADD のパイプライン動作を図 10 に示す。図中の丸は格子点の値を表す。図中の数字入りの四角は格子点の値に重み定数 0.25 を掛け合わせたものである。乗算器、加算器のパイプライン段数は 8 段である。図 10(a) に格子点の番号を示す。

まず、0 から 7 サイクルにおいて、求める格子点の上段に位置する格子点 1~8 を BlockRAM から読み出し、乗算器の入力とする。次に、8 から 15 サイクルにおいて、乗算器から結果が出力されると同時に、求める格子点の左側に位置する格子点 10~17 が乗算器に送られる。そして、16 から 23 サイクルにおいて、求める格子点の右側に位置する格子点 12~19 が乗算器に送られる。このとき同時に、格子点 1~8 を 0.25 倍した値と、格子点 10~17 を 0.25 倍した値を足し合わせる処理が行われる。同様の処理を行うと、上下左右の格子点のデータに重み定数を掛けて足し合わせる計算結果が、41~48 サイクルにて出力される。

計算結果を BlockRAM に書き込む際に、同じ時刻中に使用する格子点のデータを更新してはならない。そのため、BlockRAM にデータを書き込む前に FIFO などの一時バッファを実装する防止策が用いられる。しかし、提案アーキテクチャでは FIFO などの一時バッ

^{*} BlockRAM とは各 FPGA が持つ低レイテンシの SRAM である。

^{☆☆} 境界領域のデータとは、上下左右に隣接する FPGA から通信されるデータである。

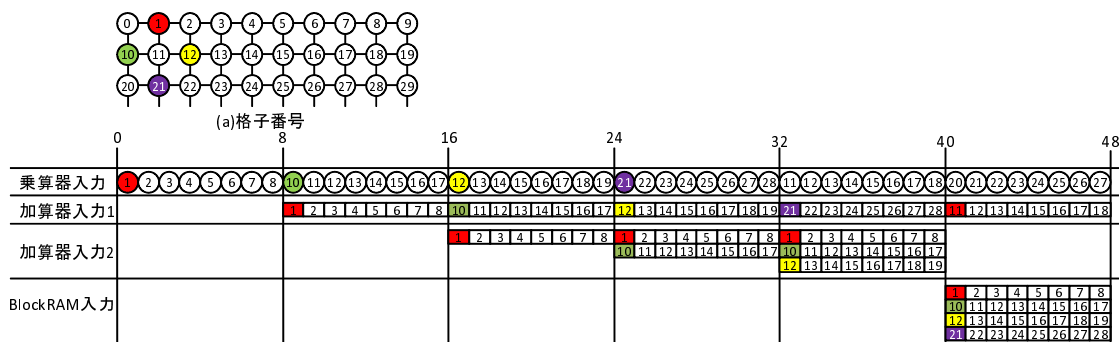


図 10 積和演算器 (MADD) のパイプライン動作。

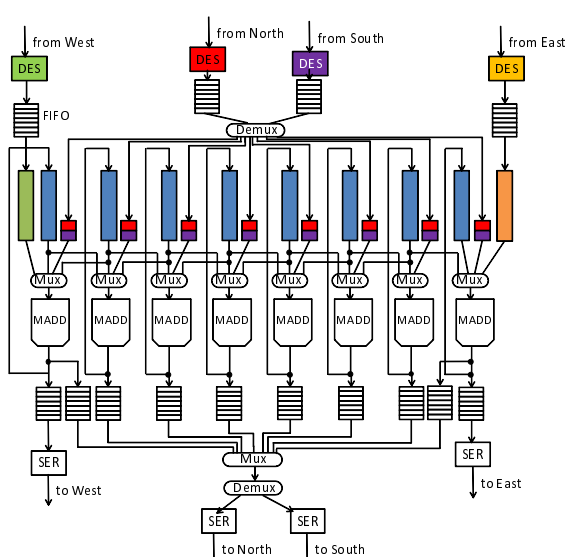


図 11 通信を含む FPGA ノードのアーキテクチャ。

ファを実装する必要がない。これは、MADD のパイプラインが FIFO の役割を果たすためである。図 10 の例では、40 から 48 サイクルにおいて、格子点 11~18 の値が更新される。この格子点 11~18 の値は、32 から 40 サイクルで乗算器に入力され、それ以降使用しない。このため、1 個の FPGA では、FIFO の使用なしに値の更新順が守られる。しかしこのスケジューリングは、計算格子の横幅が乗算器、加算器のパイプライン段数と等しい場合に限られる。つまり本稿の場合、乗算器、加算器のパイプライン段数が 8 段であるので、1 つの MADD が処理する計算格子の横幅は 8 となる。

この実装によりパイプラインの充填率をほぼ 100% にして動作させることができる。また、この実装は値の更新のためのバッファが必要ない。このため、高い演算性能と、少ない回路規模を達成できる。

4.4 実装の全体図

図 11 に通信を含むアーキテクチャを示す。図 7 と

の差分について説明する。図の DES は隣接 FPGA からデータを受け取るデシライザである。SER は隣接 FPGA へデータを送信するシリアルライザである。

デシライザの出力には FIFO を用意する。この FIFO によってデータの更新順が保たれる。通信は境界領域のみ発生するため、書き込みは境界領域を保存する BlockRAM だけである。

シリアルライザの入力にも FIFO を用意する。この FIFO は MADD の値を入力とする。ただし、境界領域の値だけを格納する。

5. 評価

5.1 環境

図 12 に、使用する FPGA アレーのハードウェア構成を示す。*FPGA をメッシュ状に接続することによって、ステンシル計算の問題サイズに合わせてアレーシステムを自由にスケールリングすることが可能である。

FPGA アレーの各ノードは FPGA(Xilinx Spartan-6 XC6SLX16) を搭載し、各 FPGA が持つ BlockRAM の容量は 72KB である。FPGA に実装する MADD は、Xilinx のコアジェネレータが生成する IP コアを利用する。MADD を 1 つ実装するために 1 個の Spartan-6 FPGA が有する DSP ブロック 32 個の内、4 個を消費する。したがって、1 個の FPGA に実装できる MADD の数は 8 つとした。

これらの回路は Verilog HDL で記述した。回路情報の生成に Xilinx ISE 13.3 を用いた。

実装した回路の検証と動作速度の比較のために、C 言語で記述したステンシル計算プログラムを使用する。検証用には、FPGA の浮動小数点演算と精度が同じ Softfloat ライブラリを用いてプログラムを記述した。ただし、動作速度の比較には、高速動作のため Softfloat ライブラリを用いていない版も記述し、これを利用した。

次節では MADD を 8 つ実装した 1 個の FPGA についての性能評価を示す。計算問題サイズ^{☆☆}は 64×128

[☆] 図 12 の SRAM は使用しない

^{☆☆} 計算可能な格子点の総数は、72KB(BlockRAM の容量) ÷ 4B(格子点のデータ：単精度浮動小数点) より 18K である。た

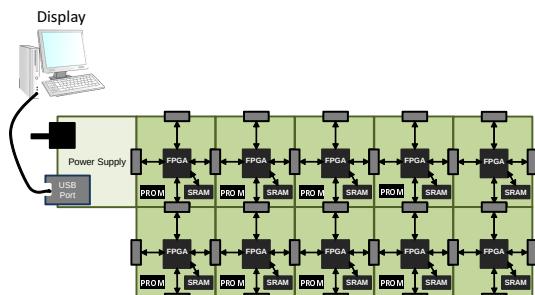


図 12 メッシュ接続 FPGA アレーの構成図.

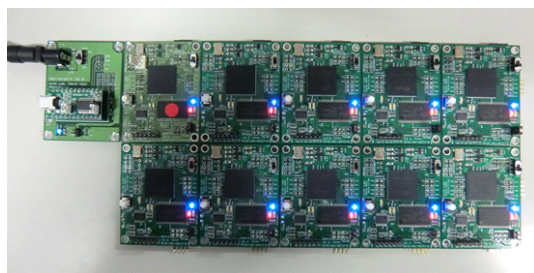


図 13 使用する FPGA アレー.

表 1 ハードウェア資源使用量

Device Utilization Summary			
Slice Logic Utilization	Used / Available	Utilization	
LUTs	4,560 / 9,112	50%	
Slices	1,527 / 2,278	67%	
BlockRAM	24 / 32	75%	
DSP48A1	32 / 32	100%	

の 2 次元データセット, イテレーション回数を 50 万とした. 計算結果は USB で接続された PC に出力し, C で記述したステンシル計算のプログラム結果と比較した. その結果, すべての格子点の値が一致した.

5.2 ハードウェア資源使用量

FPGA に実装される 1 つの MADD の LUT の使用率は 9% であり, 8 つの MADD が実装される FPGA の LUT の使用率は 50% である. ☆1 個の FPGA が消費するハードウェア資源を表 1 に示す. MADD を 8 個実装するのに全ての DSP ブロックを使用するので DSP ブロックの使用率は 100% となる.

5.3 単一 FPGA ノードの性能

FPGA アレーの性能を解析するための設計パラメータを表 2 に定義する. 動作周波数を F GHz, 各 FPGA に実装される MADD の数を N_{MADD} , FPGA の数を

表 2 設計パラメータ

動作周波数	F GHz
FPGA の数	N_{FPGA}
MADD の数	N_{MADD}
ハードウェアピーク性能	P_{peak} GFlop/s
演算回数	OP
格子点の総数	$GRID$
イテレーション回数	$ITER$

N_{FPGA} とする. 各 MADD はサイクルごとに乗算と加算を同時に実行出来る. これにより単一の MADD は $2F$ GFlop/s のハードウェアピーク性能を有するので, 単一の FPGA は $2FN_{MADD}$ GFlop/s のハードウェアピーク性能を有する. 以上より, N_{FPGA} 個の FPGA アレーのハードウェアピーク性能 P_{peak} GFlop/s は次式で表される.

$$P_{peak} = 2 \times F \times N_{FPGA} \times N_{MADD} \quad (1)$$

動作周波数が 0.16GHz の時, N_{MADD} が 8, N_{FPGA} が 1 であるのでハードウェアピーク性能 P_{peak} は 2.56GFlop/s となる. しかしながら, 図 2 より, 1 要素の計算に 7 浮動小数点演算であるため, 演算器の稼働率は $(4 + 3)/8 = 87.5\%$ となる. したがって, 動作周波数が 0.16GHz の場合に得られるステンシル計算のピーク性能は $2.56 \times 0.875 = 2.24$ GFlop/s となる.

動作周波数を変化させた, 1 個の FPGA のステンシル計算のピーク性能と実効性能を図 14 に示す. 実効性能は, 浮動小数点演算の総数/実行時間によって求める. 浮動小数点演算の総数は, 表 2 より格子点を求めるための演算回数☆☆を OP , 格子点の数を $GRID$, イテレーション回数を $ITER$ と定義すると,

$$OP \times GRID \times ITER = 7 \times 64 \times 128 \times 500000$$

と求めることができる. 実行時間はストップウォッチによって計測した.

ステンシル計算のピーク性能と実効性能がほぼ同じであり, この提案した計算手法のオーバーヘッドが少ないことがわかる. また, 比較のため同様の計算を C により記述し, 最適化オプション-O3 によってコンパイルした. 動作周波数 3.4GHz の Intel Core i7-2600 プロセッサのシングルスレッドで実行したところ, 実効性能は 8.64GFlop/s であった. この結果は文献 6) の 2.8GFlop/s と比較して, 十分な性能が得られている. 0.16GHz で動作する 1 個の FPGA の実効性能は 2.24GFlop/s, 3.4GHz で動作する Intel Core i7-2600 の実効性能は 8.64GFlop/s であった. つまり, 1 個の FPGA で Intel Core i7 の 26% の性能を達成した.

5.4 単一 FPGA ノードの消費電力量

単一 FPGA ノードの消費電力を測定するため, 動

だし, 横幅はスケジューリングの条件と MADD の数から 64 となる.

☆ 9% には, MADD 以外の USB 出力のための通信機構が含まれる. また, 複数の MADD を実装するため最適化が効く. このため, 9% × 8 よりも小さくなる.

☆☆ 演算回数とは, 1 要素のデータが時刻 k から $k+1$ への更新に要する演算の総数. ここでは乗算が 4, 加算が 3 となるので演算回数は 7 となる.

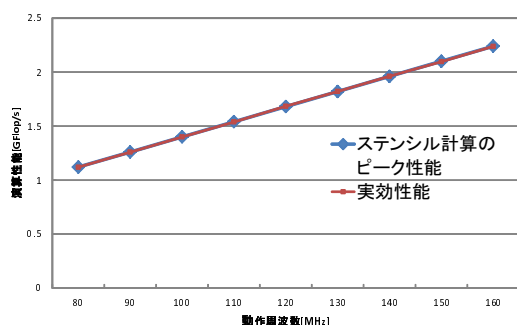


図 14 1 個の FPGA のステンシル計算のピーク性能と実効性能.

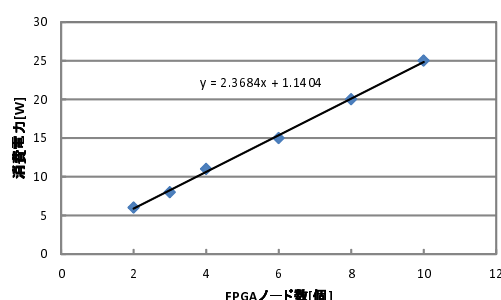


図 15 FPGA ノード数を変化させたときの消費電力 (0.16GHz).

作周波数 0.16GHz の FPGA ノードを複数接続し、WattChecker により計測した。10 個の FPGA システムでは消費電力は 25W であった。接続する FPGA ノードの数を変化させた消費電力を図 15 に示す。FPGA ノード 1 つについての消費電力を求めるために、プロットされた点に対し線形近似をとったところ約 2.37W であった。図 15 中の近似式の 1.1404 という値は、各 FPGA ノードに電力を供給する電源ボードの消費電力と考えられる。

6. 大規模 FPGA アレーシステムの検討

現在、通信を含む FPGA アレーシステムは構築中であるため、本稿では、単一 FPGA ノードの評価結果を用いて 256 個*の FPGA からなるシステムの性能を見積もる。

6.1 通信性能

通信性能について、まずレイテンシについて議論し、次にバンド幅について述べる。

1 つの FPGA が計算するグリッドサイズは 64×128 とする。これを 8 分割し、1 つの MADD が 8×128 のデータを計算する。1 回の計算に 4 つのデータを使用する。このため、1 イテレーションの計算には $8 \times 128 \times 4$ の 4096 サイクルが必要である。先に述べたように、通信レイテンシはほぼ 1 イテレーションの余裕がある。

* ScalableCore システムは 180 ノードで安定して動作しており、この結果から 256 個という値を算出した。

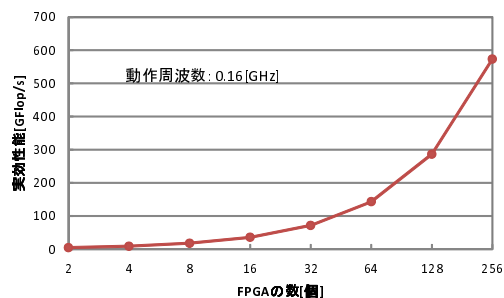


図 16 FPGA の数を増加させた場合の実効性能向上率の見積もり.

4096 サイクルは十分に大きいので、通信レイテンシは問題にならない。

次に、要求される通信バンド幅について述べる。1 イテレーションの左右の通信は、 128×32 bit である。上下の通信は、 64×32 bit である。このため、左右の通信に要求される通信バンド幅は上下よりも大きい。左右の通信に要求されるバンド幅は、 128×32 bit / 4096 cycle で 1 bit/cycle となる。このため、(通信バンド幅 [Mbps] **)/(動作周波数 [MHz(Mcycle/s)]) ≥ 1 が成り立てば、通信を待つ必要がない。ただし、通信に必要なスタートビットや終了ビットなどの影響を除いて考えている。

6.2 256 ノードでの実効性能の見積もり

この節では、動作周波数 F を 0.16GHz, N_{MADD} を 8, N_{FPGA} を 256 とする場合の実効性能の見積もりを示す。

図 16 に、FPGA の数を増加させた場合の実効性能向上率の見積もりを示す。式 (1) より、ハードウェアピーク性能 P_{peak} が 655 GFlop/s である。しかしながら演算器の稼働率は 87.5% であるので、FPGA アレーが通信オーバーヘッド考慮せずに達成し得る実効性能の上限は $655 \times 0.875 = 573$ GFlop/s となる。

また電力あたりの実効性能の見積もりを示す。図 15 の近似式より、256 個の FPGA ノードを接続した FPGA アレーの消費電力は 607W と見積もれる。そのため、電力あたりの実効性能の見積もりは 0.944 GFlop/sW となる。この値は、2011 年 11 月の Green500 ランキング¹⁰⁾ において 10 から 11 位 (0.928 から 0.958 GFlop/sW) に相当する値である。

この FPGA アレーは ScalableCore ボード⁵⁾ を使用しているので、ステンシル計算に対しての実装を最適化する余地は残されていると言える。そのため最適な実装を施せば、より上位を狙っていけるものと期待できる。

6.3 消費電力についての考察

この節では、消費電力が高い理由と消費電力を抑える方法について議論する。

現在使用している FPGA アレーは、FPGA ノード 1 つに対して 5V の電圧を印加している。しかし、Xil-

** シリアル通信であるので送信・受信は共に 1bit ずつである。

inx のデータシートから Spartan-6 は 2.5V, 1.2V の電圧によって動作することができ, 実際に安定化電源器 (ALINCO DM-320MV) から電圧を印加したところ, 正常に動作することが確認できた。

この FPGA アレーの各ノードは 5V の印加電圧をリニアレギュレータを用いて 2.5V, 1.2V の電圧に降下させている。ノードに 5V の電圧を印加しているのは, 文献 5) の ScalableCore システムを安定に動作させるためである。ScalableCore システムの FPGA ノードに印加される電圧は電源ボードから離れているもののほとんど徐々に降下しているため, FPGA の駆動電圧より下回ってしまう可能性がある。そのため 5V の電圧を印加することで, 電源ボードから離れている FPGA ノードを安定に動作させている。

この電圧が降下する問題は, ステンシル計算に使用する FPGA アレーにおいても生じる可能性がある。しかし, この問題はステンシル計算専用のボードを設計することで回避することができる可能性がある。その結果, 低電圧で FPGA アレーを動作させることが可能となる。

7. おわりに

本稿ではメッシュ接続の FPGA アレーを用いて, 科学技術計算において重要な計算カーネルの一つである ステンシル計算に対する計算手法を提案した。1 ノードの実効性能は動作周波数が 0.16GHz であるとき, 2.24GFlop/s という結果を示した。

256 ノードを接続したメッシュ接続 FPGA アレーの実効性能の見積もりとしては, 動作周波数が 0.16GHz であるとき, ハードウェアピーク性能は 655GFlop/s であるため, 達成し得る実効性能の上限は 573GFlop/s である。また, 電力あたりの実効性能の見積もりは 0.944GFlop/s/W である。

今後の課題としては通信モジュールを組み込んだ FPGA アレーを実装し, その実効性能を測定すると共に, より低電力に最適化した設計を施すことが挙げられる。

謝辞 本研究の一部は, 科学技術振興機構・戦略的創造研究推進事業 (CREST) 「アーキテクチャと形式的検証の協調による超ディペンダブル VLSI」の支援による。

参考文献

- 1) Mencer, O., Tsoi, K. H., Cramer, S., Todman, T., Luk, W., Wong, M. Y. and Leong, P.: Cube: A 512-FPGA cluster, *Programmable Logic, 2009. SPL. 5th Southern Conference on*, pp. 51–57 (2009).
- 2) Sano, K., Hatsuda, Y. and Yamamoto, S.: Scalable Streaming-Array of Simple Soft-Processors for Stencil Computations with Constant Memory-Bandwidth, *Field-Programmable Custom Computing Machines (FCCM), 2011*

IEEE 19th Annual International Symposium on, pp. 234–241 (2011).

- 3) 吉見真聡, 西川由理, 天野英晴, 三木光範, 廣安知之, オスカーメンサー: ストリームアプリケーション向け大規模 FPGA アレイ CUBE の性能評価, 情報処理学会論文誌. コンピューティングシステム, Vol. 3, No. 3, pp. 209–220 (2010-09-17).
- 4) Datta, K., Murphy, M., Volkov, V., Williams, S., Carter, J., Oliker, L., Patterson, D., Shalf, J. and Yelick, K.: Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures, *Proceedings of the 2008 ACM/IEEE conference on Supercomputing, SC '08*, Piscataway, NJ, USA, IEEE Press, pp. 4:1–4:12 (2008).
- 5) Takamaeda-Yamazaki, S., Sano, S., Sakaguchi, Y., Fujieda, N. and Kise, K.: *International Symposium on Applied Reconfigurable Computing (ARC 2012)* (2012).
- 6) Augustin, W., Heuveline, V. and Weiss, J.-P.: Optimized Stencil Computation Using In-Place Calculation on Modern Multicore Systems, *Proceedings of the 15th International Euro-Par Conference on Parallel Processing, Euro-Par '09*, Berlin, Heidelberg, Springer-Verlag, pp. 772–784 (2009).
- 7) Phillips, E. and Fatica, M.: Implementing the Himeno benchmark with CUDA on GPU clusters, *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, pp. 1–10 (2010).
- 8) 佐藤一輝, 黎江, 高橋健一, 田向権, 小林祐一, 関根優年: FPGA アレイに実装するポアソン方程式と CIP 法演算回路の性能評価, 電子情報通信学会技術研究報告. CAS, 回路とシステム, Vol. 109, No. 396, pp. 19–24 (2010-01-21).
- 9) Kobori, T., Maruyama, T. and Hoshino, T.: A Cellular Automata System with FPGA, *Field-Programmable Custom Computing Machines, 2001. FCCM '01. The 9th Annual IEEE Symposium on*, pp. 120–129 (2001).
- 10) The Green500 List: <http://www.green500.org>.