RC-011

差分法専用計算機における FPGA 間時分割通信機構の遅延評価

Evaluating Inter-FPGA communication of Custom Computing Machines for Finite Difference Methods

王 陸洲 WANG Luzhou　佐野 健太郎 Kentaro SANO　初田 義明 Yoshiaki HATSUDA　飯塚 尊則 Takanori IIZUKA　山本 悟 Satoru YAMAMOTO

1　緒言

近年、熱力学、流体力学、電磁気学など多くの分野において、莫大な時間やコストを必要とし、また危険を伴う実験に代わり、安価で実験では不可能な情報を得ることのできる数値シミュレーションが広く用いられている。しかしながら、大規模化・複雑化する数値シミュレーションにさらなる計算性能が求められる一方で、計算に用いられるスーパーコンピュータや PC クラスタなどの汎用マイクロプロセッサに基づく計算機システムでは、その汎用性のためにシステムが大規模になるにつれ効率良く性能向上を得ることが困難となりつつある。その理由として、メモリ帯域不足や命令レベル並列性不足によるプロセッサ単体の稼働率の低下や、プロセッサ間通信のオーバーヘッドによる並列処理効率の低下が挙げられる。

このような背景の下、計算問題に特化した専用構造を持つ専用計算機により、高い計算性能を効率良く実現する試みが多数行われている [1–7]。現在、専用計算機を実現する主な手段として、ASIC（Application Specific Integrated Circuit）や FPGA（Field-Programmable Gate Array）による実装が挙げられる。このうち、ASIC はハードウェア資源を有効に利用して高速、小面積かつ低消費電力の高性能な専用回路を実現可能ではあるものの、半導体プロセスに掛かる初期コストが高く、用途の限られる専用計算機では量産になり難いため、高価な計算機になりがちである。これに対し、FPGA は回路を再構成することにより様々な計算機を実現する汎用のプログラマブルデバイスであるため、ASIC に特有の初期コストがほぼ無いに等しい。また、近年の FPGA では高性能化が進み、大規模化、高集積化に加え、乗算器である DSP（Digital Signal Processing）ブロックのような専用ユニットを大量に搭載するようになり、数値シミュレーションに必要な浮動小数点演算性能において汎用マイクロプロセッサを超えつつあると報告されており [8, 9]、性能の面においても ASIC との差が縮まっている。

このような FPGA 技術の進展を受け、2004 年に James らは 3 次元電磁場解析を行う専用計算機 [1]、Ronald らは分子動力学計算のための専用計算機 [3]、そして、2008 年に Alexander らは金融シミュレーションを行う専用計算機 [4] を提案しており、それぞれの計算問題に対して有効な専用構造を明らかにすると共に、その有用性を評価している。しかしながら、これらの計算機ではメモリ帯域の不足を根本的には解消できず、システムの大規模化に対する計算性能の向上ついての議論がまだ不十分である。

これに対して本研究室では、流体力学、熱力学、電磁気学などの多くの分野において支配方程式となる偏微分方程式を解くための差分法に着目し、特定の計算問題ではなく、解法自体が持つ普遍的な性質を利用した差分法専用計算機の実現を目指して研究を行っている。将来的には回路を再構成することにより性質の異なる差分法に適応できる柔軟な専用計算機を考えているが、現時点ではまず差分法の一例として、様々な数値シミュレーションにおいて利用されている、線形偏微分方程式に対し直交格子を用いた差分法に特化した専用計算機を提案している [10, 11]。

本計算機は、多数の計算要素（Processing Element, PE）からなるアレイ構造を持ち、並列計算により高性能を実現する。その結果、原理的に PE 数に比例した計算性能とメモリ帯域を持つ。多数の FPGA に分割実装することにより、大規模 PE アレイを実現できる。分割実装では、FPGA の入出力帯域不足により性能向上が妨げられる恐れがあるが、これまでの研究では、2 次元メッシュネットワークにより接続された FPGA アレイに PE アレイを割り当てることにより、FPGA 間のデータ通信が計算サイクルの数パーセント程度で済み、時分割通信を行うことにより通信帯域を低く抑えることが可能であることを示している [12]。しかしながら、時分割通信は通信帯域の低下と引き替えに計算性能の制約と成りうる通信遅延を増加させてしまう。そこで、本論文では FPGA 間通信のために時分割通信機構の設計を行い、通信帯域と通信遅延を評価し、これらが現実の計算問題に対し計算性能の制約とならないことを明らかにする。

2　差分法のための専用計算機

2.1　差分法

差分法は、流体力学、熱力学、電磁気学などの多くの分野における支配方程式である線形偏微分方程式を解く主な手法の一つであり、格子生成に莫大なコストの掛かる非構造格子を用いた有限体積法や有限要素法と異なり、単純な直交格子を用いるためにアルゴリズムが単純になることが特徴である。例えば、近年、数値流体力学の分野では差分法の一種である境界埋め込み法（Immersed Boundary Method, IB 法）[13–15] が非構造格子の生成が困難な複雑形状周りの解析を容易に行えるように改良され、注目されている。

IB 法のような線形偏微分方程式に対し直交格子を用いた差分法では、場の変数を離散化し、偏微分方程式を近似した差分式、もしくは差分式を更に変形した式を計算することにより近似解を得る。単純な例として、ここでは次の 2 次元のラプラス方程式について考える。

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0 \tag{1}$$

まず、間隔が Δx と Δy の 2 次元直交計算格子において、2 次元のスカラ関数 $f(x, y)$ に対して離散化を行う。離散化された f を x 方向の添字 i と y 方向の添字 j を用い、$f_{i,j} \equiv f(i\Delta x, j\Delta y)$ と定義すると、微分に 2 次精度中心差分を適用して $f_{i,j}$ について解くことにより次の差分式が得られる。

$$f_{i,j} = c_1 f_{i-1,j} + c_2 f_{i+1,j} + c_3 f_{i,j-1} + c_4 f_{i,j+1} \tag{2}$$

ただし、$c_1, c_2, \ldots, c_4$ は Δx と Δy を含む定数である。

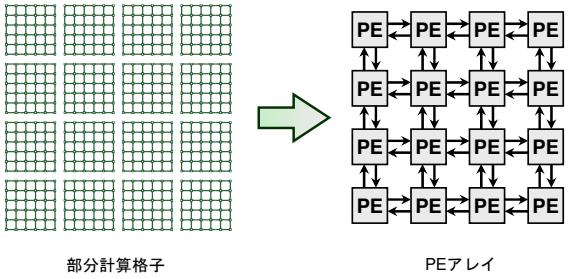

図 1　アレイ型専用計算機の構造と計算格子の関係

次に、$f_{i,j}$ に対して任意の初期値を与え、式(2)を右辺の計算結果をもって左辺を更新する漸化式として繰り返し計算を行うと、ある条件の下では $f_{i,j}$ の値が原方程式(1)の解に収束する。これは連立方程式の解を数値的に得る反復法の一つである Jacobi 法である。

式(2)は 2 次元問題に対する漸化式であるが、3 次元の場合においても、Δz 方向の差分を表す項が増えるのみで、累算に帰着できる。また、ベクトル変数や多変数問題の場合においても、変数の数に応じて累算式が増えるが、式自体は累算の形になる。他の高次差分スキームも同様に、同じ格子点において f の偏微分を定義された別の変数と見なすことにより、多変数問題となり、最終的に累算に帰着できる。

一般的に、このような数項からなる累算は、複数の格子点について同時に計算できる並列性、複数の格子点について同種の演算を行う規則性、そして、計算には対象格子点の近傍の数点の値のみを使う局所性、という 3 つの性質を持つ。本研究では、偏微分方程式を高速に解くため、これらの性質を利用する差分法計算機を提案している [10, 11]。

2.2　差分法専用計算機のアーキテクチャ

高い計算性能を実現するためには、複数の計算要素(Processing Element, PE)による、並列性を活かした並列計算が有効であると考えられるが、各 PE への計算の割り当てと PE 間のデータ通信が問題となる。

まず、差分法の持つ 3 つの性質を活かす点において、図 1 に示すような独立した演算回路を持つ多数の PE のアレイにより大規模な並列計算を行うシストリックアーキテクチャが適しており、PE アレイの均一構造が並列性と規則性に、PE 間の局所通信が局所性にそれぞれ有効であると考えられる。しかしながら、通常のシストリックアーキテクチャでは PE を通して外部からデータを流すことに主眼を置いており、再利用するデータも一旦外部メモリに送り出してから再度取り込むことを行う。このため、限られたメモリ帯域を有効に利用できず、外部との入出力と PE 間のデータ通信がボトルネックとなる。これに対し、外部との入出力を回避するため、メモリ素子の近傍に演算回路を分散配置する計算メモリアーキテクチャ [16–19] を導入することにより、全てのデータを PE アレイに格納し、計算中における外部メモリとの通信を無くし、PE 間の通信を削減可能である。このため、シストリックアーキテクチャと計算メモリアーキテクチャを融合したシストリック計算メモリアーキテクチが差分法計算に適していると考えられる。

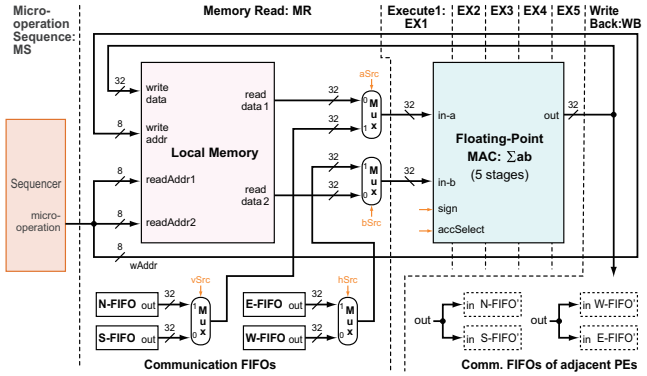

図 2　PE のデータパスのパイプライン構成

シストリック計算メモリアーキテクチャに基づき設計したアレイ型計算機は、図 1 に示すように、ローカルメモリと演算回路を持つ多数の PE を 2 次元メッシュネットワークにより接続した 2 次元のアレイ構造を持つ。2 次元計算の場合、計算格子を部分計算格子に分割して個々の PE に割り当て、アレイ全体において並列計算を行う。3 次元計算の場合には、x,y,z 空間に広がる計算格子の x 方向と y 方向のみに対して分割することにより、2 次元の FPGA アレイによる並列計算が可能である。一方、差分法の持つ局所性のため、通信は隣接する PE 間の局所通信に限られると同時に、各 PE は自分の持つローカルメモリに対して計算を行う。以上より、アレイ型計算機は PE 数に比例した演算性能とメモリ帯域を持つ。

本研究では、差分法計算に特化した PE を実現するにあたり、式(2)の累算の各項を計算する浮動小数点積和演算器(MAC)を持つデータパスを設計する。このデータパスはマイクロプログラムにより制御され、複数サイクルかけて累算を計算する。専用のデータパスとして累算全体を計算する長い演算パイプラインも考えられるが、このようなプログラマブルな構成の方が境界条件など例外的な累算に対する柔軟性に優れ、効率良く演算器を稼働できる。データパスは 5 段を含めた 8 段のパイプラインであり、複数サイクルにかけて累算を計算する。MAC はサイクル毎に 2 つの入力を受け、その積を計算をし、0 または前の計算結果と加算する。入力にはローカルメモリまたは隣接 PE から送られるデータを格納する FIFO (First-In, First-Out) から選択可能である。計算結果はローカルメモリへ書き込むと同時に、東(E)西(W)南(S)北(N)にある隣接 PE に送信可能である。送信したデータは数サイクル経ってから隣接 PE に届き、隣接 PE では受け取ったデータを FIFO に溜め、必要に応じて FIFO から取り出す。このため、PE 間の通信にはある程度の遅延が許容される。また、FIFO はデータを入力した順に出力するため、。このため、PE 間においてデータを送信する順番と使用する順番が一致するようにマイクロプログラムを作成する必要がある。

2.3　差分法専用計算機のマイクロ命令セット

PE を制御するためのマイクロ命令セットを表 1 に示す。ここで、out は MAC の出力レジスタ、M[a] はアドレスが a のローカルメモリ参照であり、LoopC、JumpC と PC はそれぞれループカウンタ、アドレスカウンタとプログラムカウンタである。

算術命令には、乗算の後に累加算を行う accp 命令、乗算のみ

表 1　マイクロ命令セット

Opecode	Dst1,	Dst2	Src1,	Src2,	動作
accp	- ,	- ,	a1,	a2,	PC++, out + M[a1] × M[a2] → out;
accp	- ,	- ,	a1,	W ,	PC++, out + M[a1] × W-FIFO → out;
accp	a3,	- ,	a1,	a2,	PC++, out + M[a1] × M[a2] → out, M[a3] ;
accp	- ,	SW,	a1,	a2,	PC++, out + M[a1] × M[a2] → out, S-FIFO, W-FIFO;
lset			N,	A,	PC++, N → LoopC, A → JumpC;
bnz					if (LoopC == 0) {PC++;} else {LoopC--, JumpC → PC;}
accpbnz	- ,	SW,	a1,	a2,	accp と bnz を同時に実行

```
 1:    mulp  -     , - , c1, W
 2:    nop
 3:    nop
 4:    accp  -     , - , c2, f[1,0]
 5:    nop
 6:    nop
 7:    accp  -     , - , c3, S
 8:    nop
 9:    nop
10:    accp  f[0,0], SW, c4, f[0,1]
```

図 3　累算式を計算するマイクロプログラム

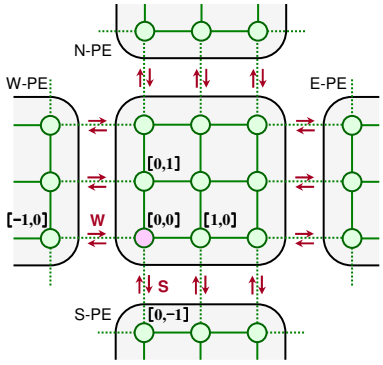

図 4　部分計算格子における格子点の添字

を行う mulp 命令、およびこれらの乗算結果に対して符号反転を行う accm と mulm 命令がある。これらの命令のオペランドには、accp を例に示したような組み合わせが可能である。

一方、制御命令にはループ制御命令である lset と bnz があり、ある範囲の命令列に対して定数回の繰り返し実行を可能とする。lset はループの先頭に配置し、ループ回数 N を即値として指定する。これに対して bnz はループ末尾に配置し、分岐処理を行う。さらに、lset と bnz は階層化されており、多重ループに対応している。また、bnz は任意の算術命令と同時に実行することが可能である。

例として、$f_{[0,0]} = c_1 f_{[-1,0]} + c_2 f_{[1,0]} + c_3 f_{[0,-1]} + c_3 f_{[0,1]}$ を計算するマイクロプログラムを図 3 に示す。ただし、nop 命令はタイミングを調節するためにある何もしない疑似命令である。

図 4 は図 3 のプログラムにおいて想定している、部分計算格子における格子点の添字と PE の関係を表した図である。ここで、$[i', j']$ は PE が担当する部分計算格子における格子点の相対的な添字を表す。この例では、4 項の累算を 4 命令に分けて計算し、西(W)と南(S)の PE のデータを 1 つずつ使用し、計算結果を西(W)と南(S)に送り出している。一般的に、i 項からなる累算は i 命令に分けて計算することになる。

また、現在の実装では、MAC の計算結果を 3 段前のパイプラインステージにフォワーディングすることにより累算を実現しているため、累算は 3 サイクル毎の入力に対して連続に行なわれる。このため、積和演算器の稼働率を高めるためには、実際のプログラムでは 3 つの累算式の各項を交互に入力するようにスケジューリングを行い、nop を無くす必要がある。

このように、各 PE では担当する部分計算格子の各格子点に対して逐次計算を行い、PE アレイ全体で並列計算を行う。

2.4　差分法専用計算機の性能

各 PE が持つ積和演算器はサイクル当たり 1 つの乗算と 1 つの加算を計算可能である。そのため、PE の動作周波数を F [Hz] とすると、PE のピーク性能は $2F$ [Flops] になる。今、$n \times n$ 個の PE アレイがあるとすると、n^2 の PE により並列処理を行うため、全体のピーク性能 P [Flops] は次式により与えられる。

$$P = 2Fn^2 \tag{3}$$

これに対し、演算器の稼働率を E とすると、実効性能 P_e [Flops] は次式により与えられる。

$$P_e = EP = 2EFn^2 \tag{4}$$

各 PE において行われる計算は PE 数に依存しないため、計算のみに依存する E は PE 数に依存しない。一方、動作周波数 F は PE の内部構造のみに依存するため PE 数に依存しない。以上より、アレイ型計算機の計算性能は PE 数に比例する。

2.5　FPGA アレイによる差分法専用計算機

単一の FPGA に実装できる PE 数は限られるため、より高性能な計算を実現するためには、複数の FPGA を用いた大規模 PE アレイの実装が不可欠である。複数の FPGA を用いる場合、FPGA 間の通信帯域がボトルネックとなりやすいため、FPGA 間の通信を小さく抑えた設計が必要である。

これに対し、PE の内部にはローカルメモリと演算器間の通信に加え、多数の制御信号があり、広い帯域を必要とするため、PE 内のデータパスを分断した非均質な分割をすべきではない。一方、計算メモリアーキテクチャを導入することにより、PE 間の通信を抑えているため、PE を最小単位とした分割が FPGA 間の通信を小さくできると考えられる。

次に、PE 間には局所通信のみが行われるため、FPGA への PE の割り当ては、PE をノード、隣接 PE をエッジで結んだグラフについて考える場合において、切断するエッジを少なくするように各 FPGA に割り当てるノードを括る問題に等しい。PE は 2 次元メッシュネットワークにより接続されているため、PE

図 5　複数 FPGA への PE アレイの均等分割

アレイを正方形領域の部分 PE アレイに分割することが望ましいと考えられる。

さらに、回路設計と動作検証の容易性の観点から、全ての FPGA に対し共通した設計を行うことが望ましい。今、PE アレイ自体は均質であるため、各 FPGA に同じ大きさの部分 PE アレイを割り当てることにより設計の共通化が可能である。

以上により、図 5 に示すように、2 次元メッシュネットワークにより接続される FPGA のアレイに PE アレイを分割実装することにより FPAG 間の通信帯域を低減できると期待できる。また、FPGA 間に十分な入出力帯域が利用可能であれば、FPGA アレイに実装される PE の総数は使用する FPGA 数に比例するため、FPGA 数に比例した計算性能が期待できる。

一方、図 3 に示したマイクロプログラムのように、各 PE では累算を複数のサイクルに分けて計算するが、途中の計算結果は隣接 PE に送る必要はないため、通信は最後の 1 サイクルに限られる。また、図 4 から分かるように、差分法の持つ局所性のため、隣接 PE へ送るデータは、さらに部分計算格子のある方向の境界に位置する数格子点に限られる。この 2 つの理由により PE 間の通信では連続して行われず、計算サイクルに対し数パーセント程度になっている。これより、FPGA 間において時分割通信を行うことにより、FPGA 間帯域を低減できると考えられる。

3　時分割通信機構

3.1　FPGA 間における実際の通信

PE 間における実際の通信の模式図を図 6 に示す。図の横軸はクロックサイクルであり、棒は各サイクルにおいて送信されるデータを表す。棒の面積は送信データ量、高さは通信に必要な帯域を表す。図に示すように、実際の計算問題では、断続的なデータ送信が行われる。このため、図 7 に示すように、データ送信のないサイクルを用いて、通信を複数のサイクルをかけて小分けに行う時分割通信が要求帯域を低減するのに有効であると考えられる。

また、本論文では隣接する 2 つの FPGA 間の接続をリンクと呼ぶが、FPAG 内部では十分に広い帯域が利用できるため、時分割通信機構が必要とされるのはリンクを跨る PE 間に限られる。さらに、差分法が持つ規則性のため、同一リンクにおいて各 PE からの送信は一斉に行われる。このため、リンクにおける通信も図 6 と同様である。

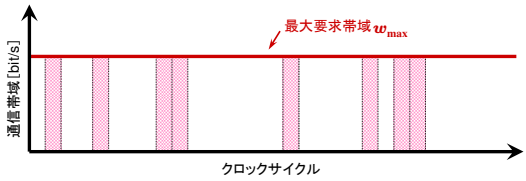

図 6　実際の計算問題における PE 間の通信と要求帯域

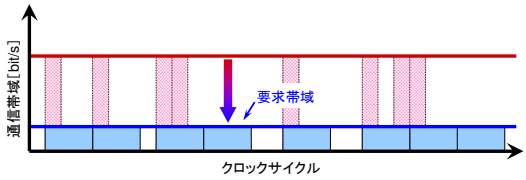

図 7　時分割通信による要求帯域の削減

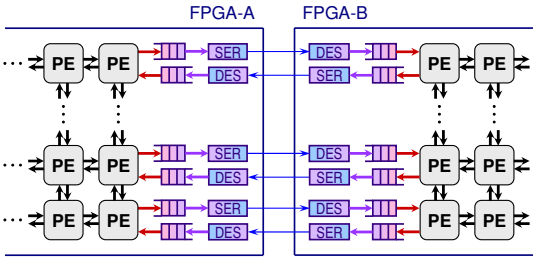

図 8　FPGA を跨ぐ PE 間に導入される時分割通信機構

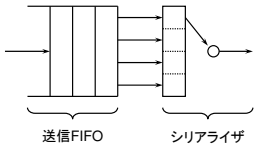

図 9　通信分散機構の送信部の模式図

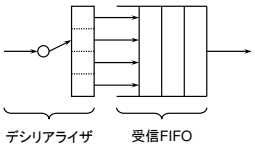

図 10　通信分散機構の受信部の模式図

3.2　時分割通信機構の構成

本研究では、図 8 に示すように時分割通信機構として、送信側の PE と FPGA の I/O ユニットの間に FIFO とシリアライザ、受信側の FPGA の I/O ユニットと PE の間にデシリライザを設ける。

送信部の模式図を図 9 に示す。模式図では、便宜的にシリアライザの時分割数 J を 4 とする。受信部は各サイクル毎に PE からデータを受け取り、受信 FIFO に書き込むことが可能である。これに対し、FIFO 読み出しは FIFO にデータが溜まっており、かつシリアライザにデータが無いときに行われる。FIFO から読み出されたデータはシリアライザのレジスタに格納され、J 分割され、J サイクルかけて FPGA の I/O に送り出される。また、シリアライザされたデータの先頭を示す制御信号を別経路により送信する。

受信部の模式図を図 10 に示す。受信部では、シリアライザから送信される制御信号によりデータの先頭を検出し、J サイクルかけて受信したデータをレジスタに組み立てる。組み立てたデータは PE に渡し、受信 FIFO に格納する。

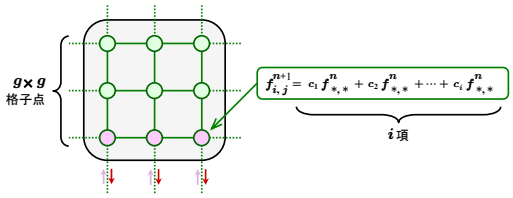

図 11　PE 中の部分計算格子および漸化式

3.3　時分割通信における要求帯域 w_{TDM}

一般に、数値計算は繰り返し計算するカーネル部と、その前後にある初期化と終了処理から成る。初期化と終了処理は 1 回しか実行されないのに対し、一般的な科学計算においてカーネル部は繰り返し実行され、計算時間の大部分を占める。このため、カーネル部のみについて考える。また、差分計算に関しては PE 間の通信において入力と出力は対称であるため、本論文では片方向の通信帯域について考え、送信の視点から定式化を行う。

ここで、カーネル部における 1 回の繰り返しを反復と呼ぶ。各反復では全く同じ計算を行うため、それぞれの反復において、あるリンクに対して送信するデータ量 S と一反復の計算時間 T を用いて、要求されるリンク当たり時分割通信要求帯域 w_{TDM} を以下のように定義する。

$$w_{\mathrm{TDM}} \equiv \frac{S}{T} \tag{5}$$

このように定義される w_{TDM} は時分割通信が理想的に行えた場合に得られる要求帯域の下限値である。このため、リンク当たりに実装する帯域を w とすると、$w_{\mathrm{TDM}} \leqq w$ となる。一方、w の上限値は FPGA の総入出力帯域 W_{FPGA} をリンク数 l により割ったリンク当たりの利用可能帯域により与えられる。このため、w の範囲は次式の通りとなる。

$$w_{\mathrm{TDM}} \leqq w \leqq \frac{W_{\mathrm{FPGA}}}{l} \tag{6}$$

ここで、W_{FPGA} は実装により実測値を得る他にないが、w_{TDM} は理論的に導くことが可能である。

問題を簡単にするため、各 FPGA は $n \times n$ の PE からなる部分 PE アレイを持ち、各 PE は図 11 のように $g \times g$ の格子点からなる部分格子を計算すると仮定する。また、各格子点に対し、i 項からなる累算を行うとする。

まず、PE がサイクル当たり送信する単位データサイズを s とすると、毎サイクル連続してデータ送信が行われる場合、動作周波数 F を掛けた sF が PE 間に必要な片方向の帯域である。FPGA 間の各リンクでは、部分 PE アレイの 1 辺に並ぶ n 個の PE がデータ送信を行うため、時分割通信機構を導入しない場合に必要とされる最大の帯域 w_{max} は次式により与えられる。

$$w_{\mathrm{max}} = nsF \tag{7}$$

次に、反復当たりの命令数を I_{iter} 命令とし、そのうち I_{com} 命令が該当するリンクに対して送信を行うとすると、式(5)中の S は nsI_{com} と、T は $T = \frac{I_{\mathrm{iter}}}{F}$ と表せる。これらを式(5)に代入し、式(7)を適用すると、次式が得られる。

$$w_{\mathrm{TDM}} = \frac{nsI_{\mathrm{com}}F}{I_{\mathrm{iter}}} = \frac{I_{\mathrm{com}}}{I_{\mathrm{iter}}} w_{\mathrm{max}} \tag{8}$$

ここで、PE アレイ全体では並列に計算が行われるため、1 反復の間、各 PE は担当する $g \times g$ 格子点に対し、それぞれ i 項からなる累算を i 命令により計算する。したがって、$I_{\mathrm{iter}} = ig^2$ である。また、差分法が持つ局所性のため、ある隣接 PE に計算結果を送る必要があるのは部分格子点の 1 辺に位置する g 格子点のそれぞれの最終結果のみであり、$I_{\mathrm{com}} = g$ 命令である。

以上より、w_{TDM} は次のように求まる。

$$w_{\mathrm{TDM}} = \frac{w_{\mathrm{max}}}{ig} \tag{9}$$

このため、各 PE に割り当てる格子点が多いほど、また累算の項数が多いほど、より大きな要求帯域の削減が期待できる。

一方、シリアライザの入力帯域は PE の出力帯域 w_{max} であり、出力帯域はリンクの通信帯域 w であるため、次式の関係が成り立つ。

$$J = \frac{w_{\mathrm{max}}}{w} \tag{10}$$

式(6)の $w_{\mathrm{TDM}} \leqq w$ の関係を式(10)に代入し、式(9)の関係を利用すると、次式のように帯域による J の最大値 J_w が求まる。

$$J \leqq J_w = ig \tag{11}$$

実際に時分割通信機構を実装する場合、一般的にシリアライザの分割数は整数に限られる。このため、FPGA 間通信の仕様設計では、J_w を決めた後、適切な J を利用可能な整数から選択し、w を調整する必要がある。

3.4　時分割通信における通信遅延

時分割通信機構はシリアル変換を用いるため、帯域の低下と引き換えに遅延が増加する。送信したデータは隣接 PE の計算に間に合わなければならないため、通信遅延も通信帯域に並ぶもう一つの重要なパラメータである。通信を伴う命令が実行されてから、送信するデータが相手 PE に届くまでにかかるサイクル数を通信遅延 D と定義すると、プログラムにおいてデータを送信してから参照するのに D サイクル後でなくてはならない。この制限を遅延制約と呼ぶことにする。プログラムにおいて、通信命令を実行してから、送信するデータが最初に参照される命令間隔が許容される最大の遅延であるため、これを D_{max} とすると、遅延制約は次式により表せる。

$$D \leqq D_{\mathrm{max}} \tag{12}$$

ここで、D を次式のように 3 つの項に分け考える。

$$D = D_{\mathrm{PE}} + D_{\mathrm{FPGA}} + D_{\mathrm{TDM}} \tag{13}$$

ここで、D_{PE} は FPGA 内部の PE 間通信にも発生するパイプラインや FIFO の書き込みのために遅延である。これに対して、D_{FPGA} はリンクを跨った通信で発生する FPGA の I/O 間の通信のために生じる遅延である。D_{TDM} は時分割通信機構のハードウェアによる遅延である。D_{PE} は PE の設計に依存し、図 2 に示す設計では $D_{\mathrm{PE}} = 7$ である。D_{FPGA} は FPGA の I/O や物理的な配線などに依るパラメータであり、D_{TDM} はハードウェアとソフトウェアの両方に依存するパラメータであり、実測値を用いる必要がある。

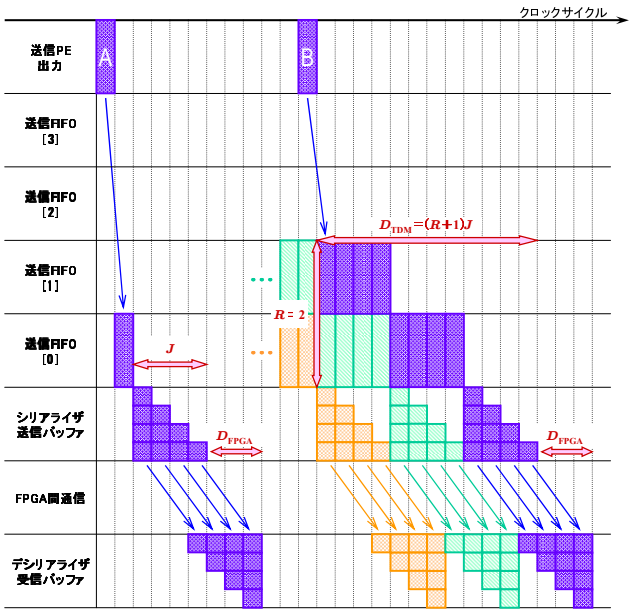

図 12　時分割通信の時空間図

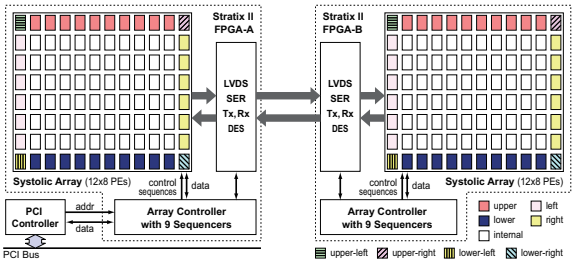

図 13　2 つの FPGA を用いた実装

表 2　実装の諸元

全 PE 数	192（$= 8 \times 12 \times 2$）
リンク当たりの片方向境界 PE 数 n	8
シーケンサ数	9
PE の動作周波数 F	106 MHz
FPGA の片方向 I/O 帯域 W_{FPGA}	79.7 Gbit/s
通信遅延 D	$(R+1)J+13$ cycle
D_{PE}	7 cycle
D_{FPGA}	6 cycle

図 12 は時分割通信の時空間図である。横軸はクロックサイクル、縦軸はデータを保持するユニットである。なお、D_{PE} を省略し、$D_{\mathrm{FPGA}} = 3$ とした。

送信 A は先行する送信と十分に離れた場合である。3.2 節で述べたように、送信側では FIFO 書き込みに 1 サイクル、シリアライザに J サイクルかかる。一方、受信側ではデシリアライザにおいても J サイクルかかるが、パイプライン処理のために隠蔽され、全体で $D_{\mathrm{TDM}} = J + 1$ サイクルで済む。

対し、送信 B は先行する送信と十分に離れていない場合である。送信側の FIFO に溜まっている先行命令の通信データの処理待ちが発生するため、遅延が送信 A の場合よりも長くなる。送信時に対象データも含め、送信 FIFO に溜まるデータの数を R と定義すると、最大値として $D_{\mathrm{TDM}} = (R+1)J$ サイクルになる。ただし、$R = 1$ は送信 A のような通信に該当する。

以上より、D は次のように表せる。

$$D = 8 + D_{\mathrm{FPGA}} + (R+1)J \tag{14}$$

これを、式 (12) に代入すると、遅延による J の最大値 J_D は次式の通りになる。

$$J \leqq J_D = \frac{D_{\max} - 8 - D_{\mathrm{FPGA}}}{R+1} \tag{15}$$

ここで、$D_{\max}$ はプログラムに依存し、D_{FPGA} は FPGA の仕様や FPGA 間の物理的な配線に依存する。一方、R はプログラムと J に複雑に依存する。このため、J_D を陽に与えることは難しい。これに対して、仮の J を決めることにより、プログラム毎に R と $D_{\max}$ を求め、後に仮決めした J が $J \leqq J_D$ を満たすかを確かめることは可能である。

4　実装と評価

4.1　PE アレイの実装

2 次元の FPGA アレイに実装される大規模 PE アレイの性能向上を評価するため、2 つのベンチマーク計算問題を解析し、帯域制約と遅延制約に基づき時分割通信機構を設計した。また、2 次元 FPGA アレイの一部として、時分割通信により結ばれる 2 つの FPGA に分割される PE アレイを実装した。実装には ALTERA 社の StartixII FPGA (EP2S180-5) を 2 つ搭載した評価基板 DN7000k10PCI を用いた。StartixII はハイエンド向けのシリーズであり、EP2S180 は同シリーズの中でも最大規模の FPGA である。

本実装の概略を図 13 に示す。2 つの FPGA には合計 $8 \times 12 \times 2 = 192$ 個の PE からなる PE アレイを実装できた。これらの PE は 106 MHz で動作する。FPGA の I/O 帯域は、LVDS (Low Voltage Differential Signaling) により、片方向 $W_{\mathrm{FPGA}} = 79.7$ Gbit/s 利用可能である。これは、結果的に本実装において時分割通信機構を導入する必要のない十分な帯域であるが、FPGA 当たり 4 つのリンクを必要とする 2 次元 FPGA アレイの実装、さらに I/O 帯域がより狭い安価な FPGA を用いた場合を想定し、利用する I/O 帯域を限定して実装した。性能が少し劣っても安価であれば、より多くの FPGA を利用して巨大な FPGA アレイを構築することにより、高い性能が得ることが可能である考えられる。以上より、実装の諸元を表 2 に示す。

4.2　ベンチマーク問題

ベンチマーク問題として、Red-Black Successive Over-Relaxation 法 (RB-SOR) [20] による 2 次元熱伝導問題の計算、および Finite-Difference Time-Domain 法 (FDTD) [21, 22] による 2 次元電磁波伝播問題の計算を用いた。

2 次元熱伝導問題は図 14 に示すように、均質な長方形領域の左辺に高温熱源として $\phi = 1$、他の辺には低温熱源として $\phi = 0$ を与えたときの温度分布を求める問題とした。定常状態になるため、支配方程式である熱伝導方程式は式 (1) のラプラス方程式になる。RB-SOR 法は Jacobi と同様、反復法の一つではあるが、Jacobi 法よりも収束性が良く、かつ並列性が失われないように改良した手法である。RB-SOR 法の累算は式 (2) の右辺に $\phi_{i,j}$

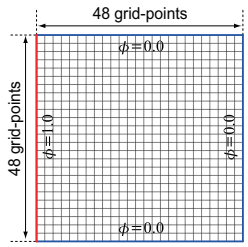

図 14　2 次元熱伝導問題の計算条件

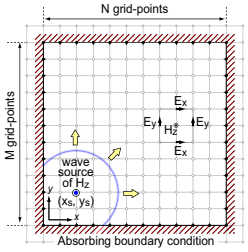

図 15　2 次元電磁波伝播問題の計算条件

を含む項を加えた 5 項の累算からなる。また、格子点数を 192 × 192 点とし、PE 当たり 8 × 24 点を割り当てた。反復数は 15 万回とした。

2 次元電磁波伝播問題は図 15 に示すように、空間の左下方に周期的な磁場を印加したとき、発生する電磁波の伝播を計算する問題とした。また、空間は無限に広がるとし、周囲の境界には Mur の吸収境界条件を課した。FDTD 法は、ベクトル微分に適した直交格子である Yee 格子 [21, 22] に基づき、電磁場の支配方程式であるマクスウェル方程式を時空間領域で直接解く時間進行法の一つである。2 次元の電磁波は、電場と磁場のいずれを 2 次元にするかにより 2 種類の表現があるが、ここでは電場を 2 次元、磁場を 1 次元とした表現を用いた。このため、累算式は、電場を計算する 3 項からなる累算が 2 つ、磁場を計算する 5 項からなる累算が 1 つ、計 3 つの式になる。また、格子点数は 144 × 72 点とし、PE 当たり 6 × 9 点を割り当てた。反復数を 4000 回とした。

4.3　時分割通信機構の実装および性能評価

これまでに示した実装とベンチマークの諸元から、式 (11) と (15) を用いて求めた、帯域によるシリアライザの時分割数 J の最大値 J_w および遅延による J の最大値 J_D を表 3 に示す。J_w の算出には、式 (11) を用い、累算の項数 i と境界格子点数 g において異なる値が該当する場合、条件の厳しい、小さい値を用いた。一方、J_D の算出には、ソフトウェアシミュレーションにより J を幾つか仮決めして、式 (15) を用いて R の最大値を求めた。分割数は実装に良く用いられる 2 の指数に設定した。このうち、FDTD における $J = 32$ は $J \leqq J_D$ を満たしていない。なお、遅延の許容値 $D_{\max}$ はプログラムを解析して得た実測値である。

表 3 に示すように、FDTD の J_w は 18 と、RB-SOR の 40 より小さい値となっている。このため、FDTD の方が帯域に関する制約が厳しいと言える。これは、FDTD 法では累算の項数と PE 当たりの格子点数が少ないためである。科学計算において一般的に用いられる 2 次精度以上の差分法において、累算は少なくとも 3 つの項を持つことから、項数に関しては FDTD 以上に小さくはならない。一方、PE 当たりの格子点数は、PE のローカルメモリ容量と場の変数の数に依存する。このため、変数の数が多い計算問題においては、通信帯域が計算性能の制約になる恐れがある。

また、J_D においても FDTD の方が小さい値となっている。これは、RB-SOR の送信 FIFO 内のデータ数 R の最大値は 1 となり、データが溜まらないのに対し、FDTD の送信 FIFO 内の最大データ数は 3 と 6 となり、データが溜まるためである。これに関して、RB-SOR 法は $D_{\max}$ が大きくなるように命令スケジューリングを行っているのに対し、FDTD はまだスケジューリングが不十分であり、改善の余地があると考えられる。命令スケジューリングは遅延制約を大きく左右するため、スケジューリング手法を確立することが計算性能を引き出すために必要不可欠である。

表 3　帯域制約および遅延制約に関わる諸元

ベンチマーク問題	2 次元熱伝導問題		2 次元電磁波伝播問題	
数値手法	RB-SOR		FDTD	
累算の項数 i	5		3, 3, 5	
境界格子点数 g	8, 24		6, 9	
帯域による J の最大値 J_w	40		18	
遅延の許容値 $D_{\max}$	116 cycle		82 cycle	
D_{FPGA}	6 cycle		6 cycle	
分割数 J	$16=2^4$	$32=2^5$	$16=2^4$	$32=2^5$
FIFO 内の最大データ数	1	1	3	6
遅延による J の最大値 J_D	102	102	22.7	11.3
$J \leqq J_D$	可	可	可	不可

実装において、両方のプログラムが共通に実行できるように、J=16 として時分割通信機構を実装した。また、送信 FIFO は 32 個のデータまで溜められるように設定した。実装の結果、リンク当たりに必要な帯域は 1.75 Gbit/s であった。これは使用している FPGA の I/O 帯域 (79.7 Gbit/s) の約 50 分の 1 であり、FPGA 当たりに 4 リンク必要とする 2 次元 FPGA アレイを実装する場合においても 10 倍ほどの余裕がある結果が得られた。これは、より安価な FPGA を用いることができる可能性、そして制約のより厳しい計算問題にも対応できる可能性があることを意味する。

また、時分割通信機構の合成に使用したのは LUT(Look Up Table) と容量の最も小さい 512bit のメモリブロック (M512) のみであった。各 FPGA において、LUT と M512 の使用率がそれぞれ 68.8% と 77.4% であるのに対し、時分割通信機構が占めているのはそれぞれの内の 2.9% と 1.7% であった。これは時分割通信機構の実装にはハードウェア資源は十分であると言える。

5　結言

本論文では、原理的に PE 数に比例する計算性能を持つ差分法専用計算機の大規模実装を目指して、計算機を複数の FPGA に分割実装する際、時分割により FPGA 間に要求される帯域を低減させる通信機構を提案し、通信帯域と通信遅延による制約を明らかにした。また、差分法専用計算機を 2 つの FPGA に分割して試作し、実際の計算問題を解析した結果、帯域制約と遅延制約が共に余裕があり、十分に大きい 16 という分割数をもって実装することができた。そのため、FPGA 間の実装に必要な帯域は、実装に用いた FPGA の I/O 帯域に比べて十分に低く、より制約の厳しい計算問題にも対応できると考えられる。余裕を利用し、試作に用いた FPGA よりも安価な FPGA を用いて安価な高性能計算機を実現することも期待できる。さらに、時分割通信機構は PCI インタフェース等を含めた専用計算機全体に必要なハードウェア資源の数パーセント程度のみを消費しており、回路規模の点において問題にはならないと考えられる。

一方、通信遅延はプログラムに対する命令スケジューリングにより大きく左右されることが明かとなった。このため、スケジュリーングの最適化手法を確立すること、および、2 次元 FPGA アレイを用いた実装が今後の課題である。さらに、3 次元計算問題を 2 次元 PE アレイを用いて計算することも可能ではあるものの、より高い拡張性を得るためには 3 次元 FPGA アレイを用いた 3 次元の PE アレイに対する評価も検討している。また、FPGA は日々性能向上しているため、異なる世代の FPGA に対して評価を行い、差分法専用計算機の性能向上の傾向を予測することも検討している。

参考文献

[1] James P. Durbano and Fernando E. Ortiz, "Fpga-based acceleration of the 3d finite-difference time-domain method", Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM2004), pp. 156–163, April 2004.

[2] Gerald R. Morris, Viktor K. Prasanna, and Richard D. Anderson, "A hybrid approach for mapping conjugate gradient onto an FPGA-augmented reconfigurable supercomputer", Proceedings of the 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, pp. 30–12, April 2006.

[3] Ronald Scrofano, Maya B. Gokhale, Frans Trouw, and Viktor 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, pp. 23–34, April 2006.

[4] Alexander Kaganov, Paul Chow, and Asif Lakhany, "FPGA acceleration of monte-carlo based credit derivative pricing", Proceedings of the International Conference on Field Programmable Logic and Applications, pp. 329–334, September 2008.

[5] Tsutomu Hoshino, Toshio Kawai, Tomonori Shirakawa, Junichi Higashino, Akira Yamaoka, Hachidai Ito, Takashi Sato, and Kazuo Sawada, "Pacs: A parallel microprocessor array for scientific calculations", ACM Transactions on Computer Systems, vol. 1, no. 3, pp. 195–221, 1983.

[6] 岩崎洋一, "専用並列計算機による「場の物理」の研究", 平成４年度～８年度研究費補助金研究成果報告書, 1997.

[7] Y. Iwasaki, "Computers for lattice field theories", Nucl.Phys.Proc.Suppl., vol. 34, pp. 78–92, January 1994.

[8] Keith D. Underwood and K. Scott Hemmert, "Closing the gap: Cpu and fpga trends in sustainable floating-point blas performance", Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, pp. 219–228, 2004.

[9] K. Underwood, "Fpga vs. cpus: Trends in peak floating-point performance", Proceedings of the International Symposium on Field-Programmable Gate Arrays, pp. 171–180, February 2004.

[10] Kentaro Sano, Takanori Iizuka, and Satoru Yamamoto, "Systolic architecture for computational fluid dynamics on fpgas", Proceedings of the 15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM2007), pp. 107–116, April 2007.

[11] Kentaro Sano, WANG Luzhou, Yoshiaki Hatsuda, and Satoru Yamamoto, "Scalable fpga-array for high-performance and power-efficient computation based on difference schemes", Proceedings of the International Workshop on High-Performance Reconfigurable Computing Technology and Applications (HPRCTA'08), pp. digital library (DOI: 10.1109/HPRCTA.2008.4745679, 9 pages), November 2008.

[12] "複数 FPGA によるアレイ型差分法専用計算機のための FPGA 間通信帯域評価", 第 7 回情報科学技術フォーラム (FIT) 論文集, pp. 25–28, September 2008.

[13] C.S.Peskin, "The fluid dynamics of heart valves: Experimental, theoretical and computational methods", Annual Review of Fluid Mechanics, vol. 14, pp. 235–259, 1981.

[14] H. Nishida and K. Sasao, "Incompressible flow simulations using virtual boundary method with new direct forcing terms estimation", Proceedings of the 4th Internationial Conference of Computational Fluid Dynamics, pp. 185–186, 2006.

[15] Satoru Yamamoto and Koichiro Mizutani, "A very simple immersed boundary method applied to three-dimensional incompressile navier-stokes solvers using staggered grid", Proceedings of 5th Joint ASME-JSME Fluids Engineering Conference, vol. FEDSM2007-37153, 2007.

[16] J. E. Vuillemin, P. Bertin, D. Roncin, M. Shand, H. H. Touati, and P. Boucard, "Programmable active memories: reconfigurable systems come of age", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 4, no. 1, pp. 56–69, Mar 1996.

[17] David Patterson, Krste Asanovic, Aaron Brown, Richard Fromm, Jason Golbus, Benjamin Gribstad, Kimberly Keeton, Christoforos Kozyrakis, David Martin, Stylianos Perissakis, Randi Thomas, Noah Treuhaft, and Katherine Yelick, "Intelligent ram(iram): the industrial setting, applications, and architectures", Proceedings of the International Conference on Computer Design, pp. 2–9, October 1997.

[18] David Patterson, Thomas Anderson, Neal Cardwell, Richard Fromm, Kimberly Keeton, Christoforos Kozyrakis, Randi Thomas, and Katherine Yelick, "A case for intelligent ram: Iram", IEEE Micro, vol. 17, no. 2, pp. 34–44, March/April 1997.

[19] Duncan G. Elliott, Michael Stumm, W.Martin Snelgrove, Christian Cojocaru, and Robert Mckenzie, "Computational ram: Implementing processors in memory", Design & Test of Computers, vol. 16, no. 1, pp. 32–41, January-March 1999.

[20] Louis A. Hageman and David M. Young, Applied Iterative Methods, Academic Press, 1981.

[21] J. Kim and P. Moin, "Application of a fractional-step method to incompressible navier-stokes", Journal of Computational Physics, vol. 59, pp. 308–323, June 1985.

[22] John C. Strikwerda and Young S. Lee, "The accuracy of the fractional step method", SIAM Journal on Numerical Analysis, vol. 37, no. 1, pp. 37–47, November 1999.