実用 CFD コードの FPGA 回路化における技術課題の実験的検討

Experimental study on technical issues of porting real-use CFD code to FPGA circuit

○藤田 直行, 宇宙航空研究開発機構, 東京都調布市深大寺東町 7-44-1, fujita@chofu.jaxa.jp
FUJITA, Naoyuki, Japan Aerospace Exploration Agency, 7-44-1 Jindaiji-higashi Chofu, TOKYO

With a progress of computer technology, HPC (High Performance Computing) system becomes fast and large. Without using HPC system, some research activities may be progress. But very large numerical simulation, e.g. analyzing combustion and turbulence mechanism, should be activated using HPC system. Parallel processing technology has improved HPC system so long time, but the technology is going to reach the ceiling. So this research proposes one of the new technologies which progress the performance. In this paper real-used CFD FORTRAN program is ported to HDL. In the porting process, some technical issues on FPGA based computer will be extracted.

1. はじめに

計算機技術の進歩に伴い、HPC(High Performance Computing)システムも高速化、巨大化している。HPC システムを使用せずとも研究活動が行える分野も増えてきているのは事実であるが、今後も、HPC でなければできない数値シミュレーション（例えば、燃焼や乱流のメカニズム解明）を研究開発機関が推進することは不可欠と考えている。HPC システムは、並列化による速度向上を行ってきているが、そろそろ並列化技術やその利用技術が頭打ちになりつつあり、新しい方式の提案が必要となっている。

本研究の目的は、この新しい方式の一つとしての FPGA(Field Programmable Gate Array)計算機の実現可能性を検討することである。本稿では、スーパーコンピュータ上で実利用されている CFD の FORTRAN プログラムを実際に HDL(Hardware Description Language)へ移植する中で、FPGA 回路化の技術課題を実験的に抽出することを目的とする。

研究の背景(２章)、本稿で想定する FPGA 計算機システム(３章)、技術課題の抽出手順(４章)、解析の対象とした CFD プログラム(５章)を述べた後、６章で抽出された以下８点の技術課題について述べる。

(1) 大量のデータ入出力
(2) 回路規模と実装
(3) メモリスケジューラの実装
(4) ホスト計算機とのインターフェース
(5) 浮動小数点演算器のチューニング
(6) CFD 研究者による FORTRAN プログラムの FPGA 回路化
(7) 実行計算機を意識したソースプログラムのチューニング
(8) マクロの無い演算処理の実装

2. 背景

近年の HPC システムは、計算機技術の進歩やアプリケーションプログラムからの要求により巨大化している。一例として図１に宇宙航空研究開発機構(以下、JAXA と呼ぶ)における 1960 年から 2009 年までの大型計算機システムの変遷を示す。縦軸は演算性能値[FLOPS]、横軸は年代である。2009 年 4 月から稼動を開始した”JSS”は、合計 140TFLOPS の演算能力を持ち、その中核計算エンジン”M システム”は、3,008 個の演算ノードに合計 12,032 コアを持つ並列計算機である[1]。図 2 に”JSS”の構成を示す。”JSS”では、大規模な三次元非定常計算も行われている。表 1 にその計算規模を示すいくつかの指標を示す。図３には、表 1 のシミュレーション 1(液体燃料微粒化)のシミュレーション結果の一例を示す。これは世界最微粒の解像度で計算を行い、液糸および液滴生成過程を正しく捉えることを目的に行われた数値シミュレーションで、これにより乱流噴霧で起こっている物理過程を把握し、実機器への応用を目的にした噴霧 LES へのモデル化の知見を得ることが期待されている[2]。

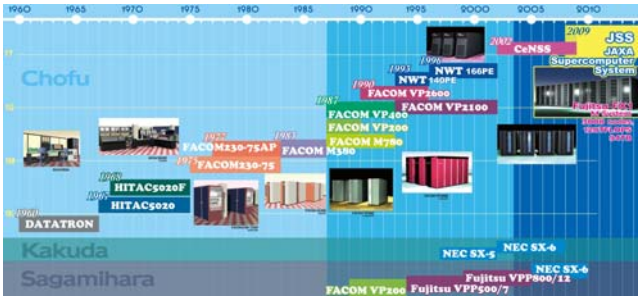


Fig. 1 “JSS” Configuration

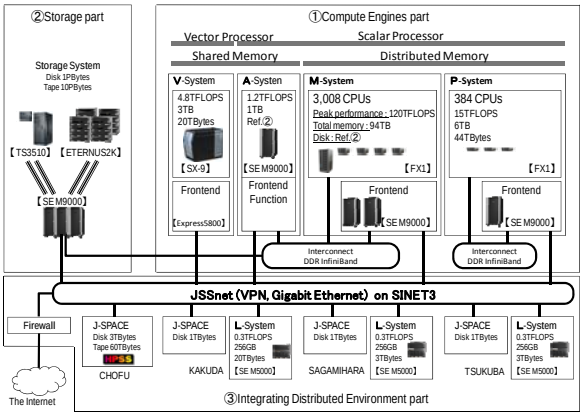


Fig. 2 “JSS” Configuration

Tbl. 1 Large Scale Simulations on “JSS”

Item	Simulation 1	Simulation 2
Field	Liquid Fuel Breakup	Space Plasma
# of Grid point	60 Billions	42 Billions
# of Core	5,760 Cores	5776 Cores
Total Output File Size	153 TBytes	430 TBytes

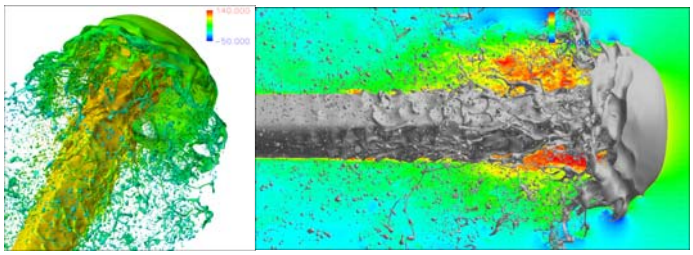


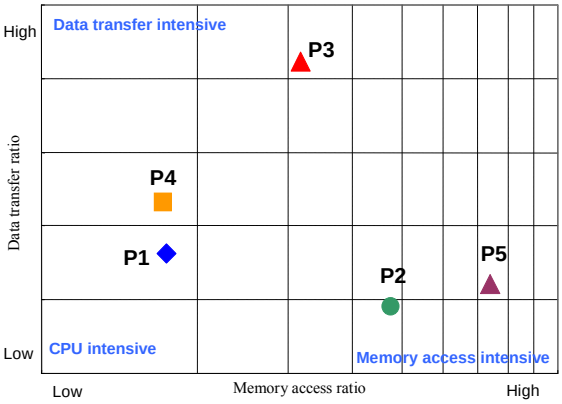
Fig. 3 Liquid Fuel Primary Breakup

ここで、”JSS”の次の世代の計算機(以下、次世代計算機と呼ぶ)でのシミュレーション規模を考えてみる。図３の計算例は従来の計算に比べ数十倍の格子点を使用していることから、次世代計算機でのシミュレーション規模も 10～100 倍になると仮定する。CPU の動作周波数が頭打ちになっていること、CPU のデータ入出力性能が CPU から取り出せるピン数で制限されていること等から、今後 CPU 単体の性能向上はあまり期待できないため、計算機のシステムとしての性能を向上させるには、より多数の CPU を並べたシステムを構築することになると想定される。表１にあるように、”JSS”でのシミュレーションの場合、5,760 個のコアを使い合計 153TBytes のファイルを出力していることから、次世代計算機でのシミュレーションは 5 万～50 万コアで 1.5P～15PBytes のファイル出力を行う計算になると予想できる。この様な大規模計算機システムは、一部の国家プロジェクトを別とすれば、現状の計算機技術や利用技術では、運用面、利用面で成立性が疑問視される。ここで、表１の二例のシミュレーションの実行効率、つまりハードウェア上の最高演算能力に対する実際に処理した演算数の割合は４%程度である。単体コアでの実行効率は一般的に 10%前後であることを考えると、5,000 コアを超える高並列計算の実行効率として４%は悪くない数値であるが改善が望まれる状況でもある。そこで、今までの計算機システム構築技術とは異なる新しい方式を提案し、運用・利用可能な程度にコンパクトな(つまり高い実行効率を持つ)システム方式の実現が必要となっている。

CPU 以外の計算機システムとしては、GPGPU（General Purpose computation on Graphic Processor Unit）“Cell/B.E.”、“ClearSpeed”等のアクセラレータボードを用いたもの等が研究開発されているが、本研究では FPGA を用いた計算機システムを研究対象にする。以下、簡単に理由を述べる。図４に JAXA の代表的アプリケーションとその特性を示す[3]。燃焼、航空、乱流、プラズマからなる５つのアプリケーションの計算負荷、メモリアクセス負荷、データ通信負荷の相対的関係を示している。図で上部に位置するアプリケーションは相対的にデータ通信の割合が大きく、右領域に位置するアプリケーション程メモリアクセスの割合が大きいことを示している。左下に位置するアプリケーションは計算負荷が大きいことを示す。そのため、各アプリケーションの特性としては P1 と P4 は演算負荷が大きいアプリケーション、P2 と P5 はメモリアクセス負荷が大きいアプリケーション、P3 はネットワーク負荷が大きいアプリケーションであることが分かる。

これより、JAXA でのアプリケーション群は、特定の特性を持つ物の集合ではなく、通信性能、メモリアクセス性能、演算性能という様々な特性を持つアプリケーションが混在している環境であることが分かる。このようなアプリケーション環境においては、アーキテクチャやシステム構成が固定されたハードウェアでは、全てのアプリケーションを高効率で実行させることはできないと考え、アプリケーションに合った通信性能、メモリアクセス性能、演算性能を提供できるハードウェア環境の実現可能性の検討が必要であると考えた。これより、この条件に適したプラットフォームとしてFPGAを用いた計算機システムを研究対象として設定した。

なお、図４の中の並列化手法にあるMPIとは、Message Passing Libraryを使ったプロセス間並列の手法を、XPFとは、富士通(株)製のデータ並列言語であるXPFortranを使った並列化を、VisIMPACTとは、富士通の提唱するVirtual Single Processor by Integrated Multi-core Parallel Architecture [4] を使った並列化手法を指している。



Name	Application	Numerical Method	Parallelization Method
P1	Combustion	FDM+Chemistry	MPI+VisIMPACT
P2	Aeronautics	FVM(Structured)	MPI+VisIMPACT
P3	Turbulence	FDM+FFT	XPF+VisIMPACT
P4	Plasma	PIC	MPI+VisIMPACT
P5	Aeronautics	FVM(Unstructured)	MPI+VisIMPACT

Fig. 4 JAXA benchmark program and its characteristics

FPGA は、システムの高速化、消費電力低減や、商品開発サイクルの短縮等の目的で、画像処理、天体の多体問題、携帯電話地上局や、家電製品等に多用されている。しかし、CFD での利用は進んでいない。この大きな理由のひとつが、CFD は高精度(64bit 程度の浮動小数点)演算を必要とすることである。また、FPGA を用いたシステムの開発環境、特に、HDL による回路設計は、研究開発コードの主流である FORTRAN の環境に比較して、その利用に、高い専門性を必要としている点も理由のひとつとである。

FPGA を用いた計算機やリコンフィギャラブル計算機の研究は行われているが、実用レベルや HPC システム等の大規模システムに対する研究は少ない。現在までの適用分野は、整数演算を主として行う画像処理や信号処理及び、単精度やそれ以下の浮動小数点計算を行う分野での研究開発や実用化が行われている。FPGA と関連して ASIC(Application Specific Integrated Circuit)を用いた大規模システムとしては、"GRAPE"が知られている。FPGA を用いた計算機については、CRAY Research が XD1 というシステムを発表しているが、これは CPU のアクセラレータとして FPGA を使用することを念頭に置いたシステムであると考えられる。

３．想定する計算機システム

話が前後するが、本章では、次章以降で行った技術課題の抽出の結果を踏まえ、今後検討を進める計算機システムの基本的構成について述べておく。これは、本稿の基本的な目標の方向性を予め述べておくことで、次章以降の判読性を高めることが目的である。

図５に、本稿で想定する計算機システムの基本構成を示す。メ

モリと FPGA が交互に接続された基本構造を持つ。浮動小数点演算等を行う演算部はスループット 1clock のパイプライン構造を持ち、この演算部とメモリ間のデータ入出力はメモリインターフェース部が担当する。メモリから演算部には演算部の周波数で 1clock 毎にデータが入力され、パイプライン通過後の演算結果が 1clock 毎に次段のメモリに出力される構造である。パイプラインのレイテンシは演算の複雑さに依存する。メモリと FPGA は、それぞれ単体のチップで構成されるものではなく、一般的には複数のチップの集合体である。

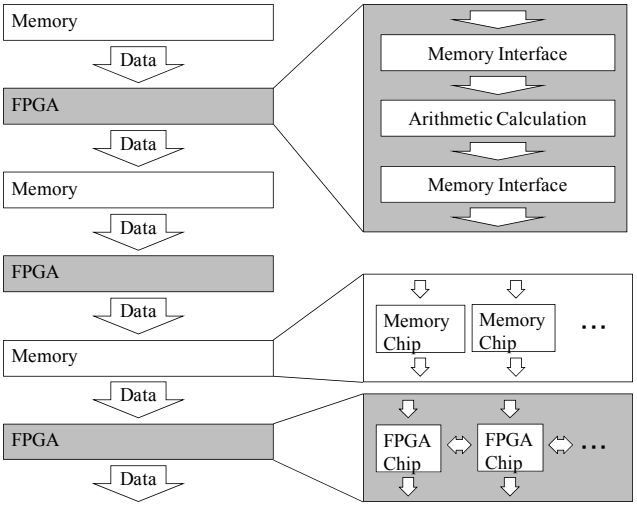


Fig. 5 FPGA computer base structure

4. 技術課題の抽出手順

1 章で述べた通り、本稿では、スーパーコンピュータ上で実利用されている CFD の FORTRAN プログラムを実際に HDL へ移植する中で、FPGA 回路化の技術課題を実験的に抽出することを目的としている。今回行った実験的抽出手順を図６に示す。６章では、この手順を実施する中で抽出した技術的課題を列挙している。

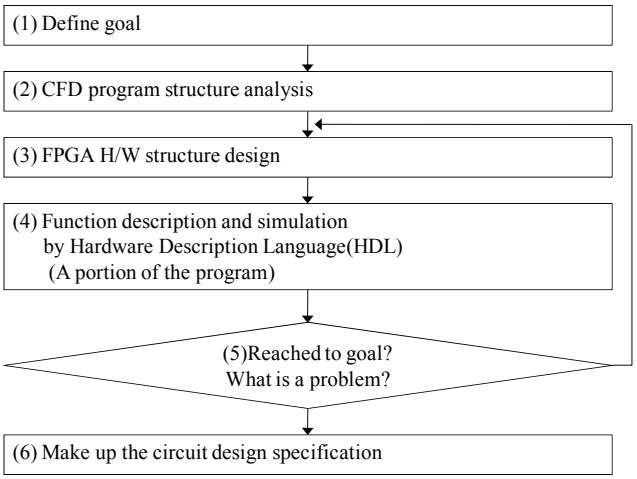


Fig. 6 Procedure of experimental study

第一段階として、FPGA 回路化による高速化の目標を設定する。今回の場合、Itanium2 1.6GHz の 100 倍の実行速度を目標に設定したが、プログラム全体での FPGA 回路の実行速度の算出途中であるため、本稿では実行速度の具体的評価については述べていない。

第二段階は、対象とする CFD プログラムの構造解析を行う。プロファイラ等を用い FORTRAN で記述された CFD プログラムを分析し、サブルーチン毎ないしはサブルーチン内を分割したブロック毎に演算コスト分布やデータの流れを作成する。これらの資料から並列化の可能性に着目しブロック間の構造を解析し、並列処理可能な部分の特定を行い、並列処理への分解と処理の分流と合流を明らかにすると共に FPGA 回路化による並列効果の高い部分を演算密度やテータの再利用性に着目し特定する。この段階では、主として FORTRAN を回路化することの技術的課題が抽出される。なお、今回対象とした CFD プログラムについては５章で詳しく述べる。

第三段階は、FPGA ハードウェアの構造設計を行う。最初に、第二段階で作成したブロック毎に回路化を実施する。続いて、ブロック毎の回路を全て一つに繋げ回路化した場合の必要リソース量を試算する。一つに繋げた回路は大きなパイプラインとなっているため、このパイプライン内の演算タイミング調整を必要に応じて行うと共に、パイプラインのレイテンシを試算する。この段階で、プログラム全体の演算器数や回路規模を見積ることが可能になる。

第四段階は、第三段階で作成した回路を HDL で記述した後、論理シミュレーション環境を構築する。Itanium2 で使用したものと同じ入力データを、作成した回路の入力データとし論理シミュレーションを行い、Itanium2 と回路の出力データを比較し回路の検証を行う。なお、第四段階は今回対象としたプログラムの中のひとつのサブルーチンについてのみ実施し、有効数字 15 桁の範囲内で一致することを確認している。他の部分については、このサブルーチンの解析結果で得られた情報を標準値と仮定し、FPGA リソースの必要数算定に用いた。

第五段階は、作成した回路が設定した高速化の目標に達しているかの判定を行う。判定は、作成したパイプライン回路のスループットとレイテンシから演算性能を算出することにより行う。達していない場合は、第三段階に戻り FPGA ハードウェアの構造設計を見直す。この段階で、高速化を阻害している原因等の回路化作業を行う中で気付いた技術的課題の全体的な抽出を行う。

本稿の目的は上述の第五段階で達成されるが、今後の活動としては、目標を達成した回路を実際の FPGA 計算機に実装し、その性能や機能を評価･確認するため回路設計仕様書を作成（第六段階）することになる。

5. 対象 CFD プログラム

FPGA 回路化を行う CFD プログラムは図４に示す５つのプログラムの中から選ぶこととした。５つのプログラムの構造分析等をした結果、以下の理由により、P2 のプログラムを選定した。

- 陽解法と陰解法の構造を持つサブルーチンの両方が、演算コストの高いサブルーチンとして含まれている。これは、CFD プログラムの代表的な特徴を顕著に表すプログラムであることを示し、評価対象として適切であると判断した。
- 合理的な仮定を設定することで、FPGA 回路化の対象コード行数が 2,000 行程度の少量となる。これは、FORTRAN から HDL への変換を手作業で行うことを考えると、重要なことである。
- アルゴリズムが確定しており、今後のプログラムの変更が少ない。従って演算コスト分布やデータの流れ等の構造解析の結果が変化する可能性が将来に亘って低く、今回の検討結果が長期に亘って有効になると期待できる。

図７に P2 での計算結果の一例を、図８に流体計算に関するフローチャートを示す。これは、”UPACS”[5]と呼ばれるプログラムであり、圧縮性完全気体の三次元流れを計算できる。Cell Center 有限体積法で離散化を行い、MUSCL 法により対流項の高精度化を

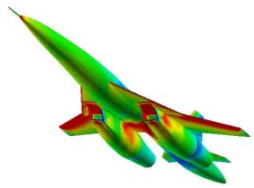


Fig. 7 Example of P2 result

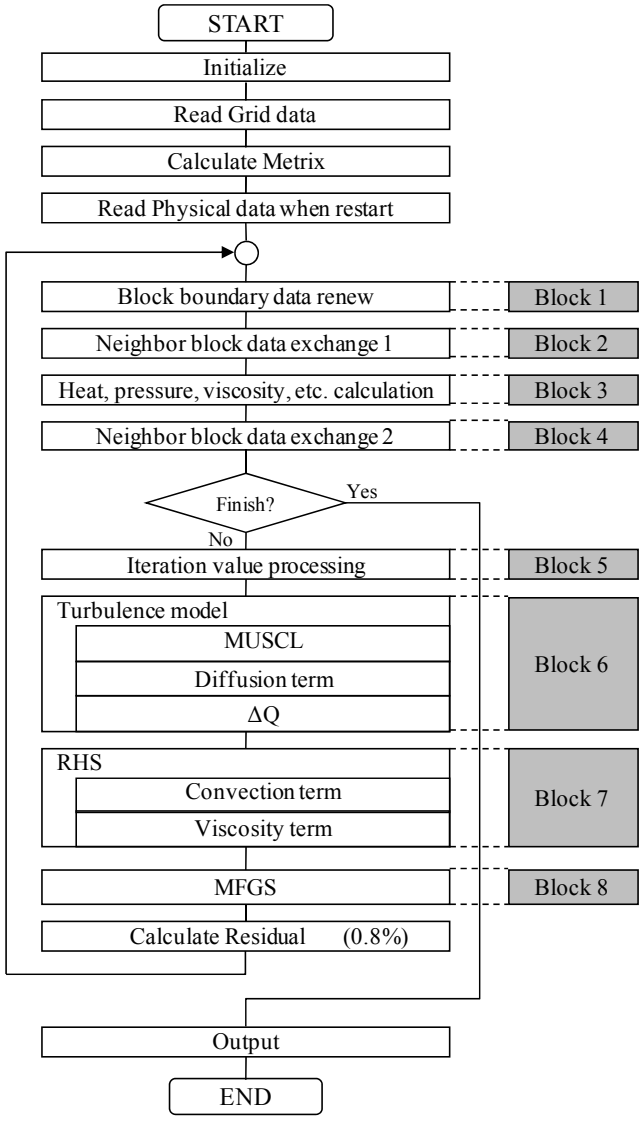


Fig. 8 P2 program fluid dynamics flow chart

し、時間積分は、Euler 陽解法、Runge-Kutta 陽解法、MFGS 陰解法の三種類から、乱流モデルは、Baldwin-Lomax、Spalart Allmaras の二種類から、それぞれパラメータを指定することにより選択可能となっている(表２)。図９には、UPACS で使用されている変数の定義位置を示す。なお、本稿で FPGA 回路化の対象にしたのは、図８にある Block1 から Block8 の部分である。

６．技術課題

本章では、４章で述べた手順により得られた技術課題を列挙していくが(6.2 節)、その前に、対象プログラム P2 の全体構造について述べ、本稿の解析対象範囲を明確にする(6.1.1 小節)と共に、FPGA ハードウェア構造設計結果の概要(6.1.2 小節)を述べる。

Tbl. 2 “UPACS” Solver

Discretization	Cell Center Finite Volume
Convection term	MUSCL
Time integration	Euler explicit Runge-Kutta explicit MFGS implicit (Select one of three)
Turbulence model	Baldwin-Lomax Spalart Allmaras (Select one of two)

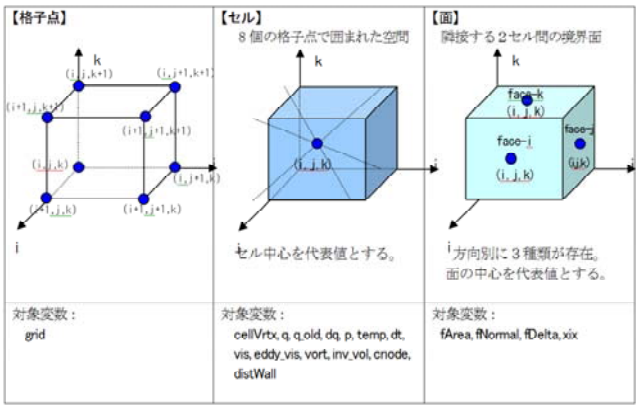


Fig. 9 P2 variable

６．１ FPGA 回路化結果

６．１．１ 分析対象の縮退

P2 の流体計算に関係するフローチャートは図８に示したが、P2 にはこれを内包するより大きな構造が存在する。CFD では高品質の格子を高速に生成するために格子をマルチブロック化することがある。P2 は、このマルチブロック化された格子における計算をマルチブロックを意識することなくソルバー部分を記述できるように考えられたものである。各ブロックの周囲には face、edge、corner という三種類の袖領域と呼ぶ部分があり、この袖領域の値はマルチブロック階層が持つ機能によって更新される(図 10)。この機能により、プログラム作成者はマルチブロックを意識することなく、単一格子でのソルバーを記述すれば良いようになっている。図８はこの単一格子でのソルバーのフローチャートを示している。

P2 のマルチブロック処理も含めた全体のコード量は FORTRAN で約 20,000 行であるが、図８及び表２で示したソルバーの中核部分のコード量はこの数分の１である。更に、時間積分の方法や乱流モデルを決定すると、今回解析の対象とした図８の Block1 から Block8 までのコード量は約 2,000 行になる。プログラムは一般的に、ソフトウェアとしての汎用性やメンテナンス性を確保するために、多くの機能をひとつのコードに書き込みパラメータ等を用いてその機能の中のひとつを指定するという方法が採られる。P2 における時間積分ルーチンや乱流モデルの可選択性もこれにあたる。本稿の解析では、このようなコードについては、選択が完了したことを前提にして解析対象をプログラム全体の 10 分の１に縮退している。ソフトウェアとして P2 を実行する場合でも、この選択はソフトウェアが実行される初期段階で固定的に決定されるため、この作業により P2 の計算結果そのものには影響を及ぼさない。

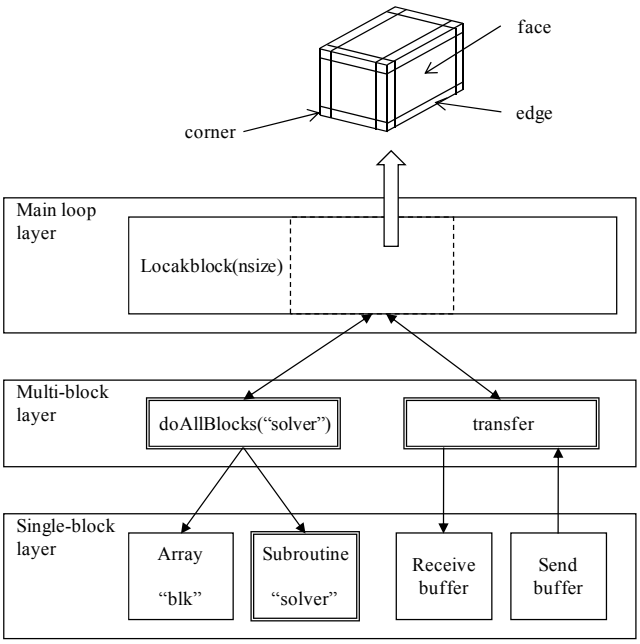


Fig. 10 Multi-block processing

６．１．２ 回路規模

４章の繰り返しになるが、FPGA ハードウェアの構造設計は、以下の手順で行う。

STEP1:回路化

CFD プログラムの構造設計段階で作成したブロック毎に回路化を実施する。

STEP2:パイプライン化

ブロック毎の回路を全て一つに繋げ回路化した場合の必要リソース量を試算する。

STEP3:タイミング調整

一つに繋げた回路は大きなパイプラインとなっているため、このパイプライン内の演算タイミング調整を必要に応じて行うと共に、パイプラインのレイテンシを試算する。

STEP4:回路規模推定

上記 STEP1～STEP3 を全てのブロックについて実施し、プログラム全体の演算器数や回路規模を見積る。

以下に、この手順の実例を示す。但し、STEP1 の回路化と STEP3 のタイミング調整については、紙面の関係上、図８の Block6 内 MUSCL 部の一部分のみを示す。これは、図 11 に示す FORTRAN プログラムに相当する部分である。

```
qc(1:4) = blk%q(i, j, k, 1:4)
rho_inv = 1.0/qc(1)
qc(2:4) = qc(2:4)*rho_inv
qc(5)   = blk%q(i, j, k, iMu)*rho_inv
```

Fig. 11 Example FORTRAN program

図 12 に、回路化とタイミング調整の結果を示す。除算器が１個、乗算器が４個の回路になり、次 Block への出力タイミング調整のために、変数/信号 qc(1)の出力部分に 64bit 幅の 65 段のタイミング調整回路を設定したことが分かる。

図 13 に、図８の Block1 から Block8 までを一つのパイプラインにした結果を示す。Block1 から Block8 までが順番に並んでいることが確認できる。図 12 に示した回路はこの一部を詳細に示したものである。Block2 と Block4 がパイプラインの並びからはみ出しているが、後にも述べるが、これは２つの Block には演算器が含まれていないため、メモリ空間の参照のみの回路となり、パイプラインではなくメモリ制御部として回路化されたためである。

表３に、Block1 から Block8 までの演算器数と、それから推定した FPGA 回路化に必要な FPGA チップ数を示す。ここで、想定 FPGA チップは Xilinx 社製 Virtex4(XC4VLX80:ロジックセル数 80,640)とし、チップへのパイプライン回路実装効率を 80%、複数のチップに跨った実装をすることによる回路分割損として 20%の使用ロジックセル数増加、メモリインターフェース部等パイプライン以外の回路実装にパイプラインと同量のロジックセルが必要であると仮定した。すなわち、演算器数から算出した FPGA チップ数に 3=1÷0.8×1.2×2 を乗じた個数を実用上の FPGA チップ数とした。なお、加減乗除算器及び平方根器については、各演算器１個当たりの必要ロジックセル数を表４のように仮定した。

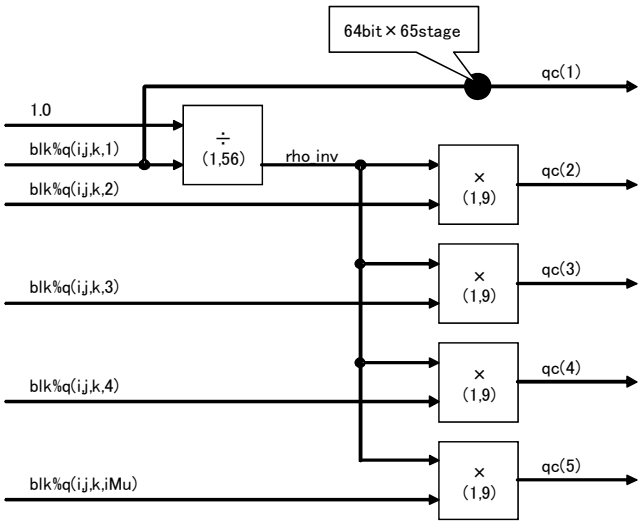


Fig. 12 Circuit and timing adjustment

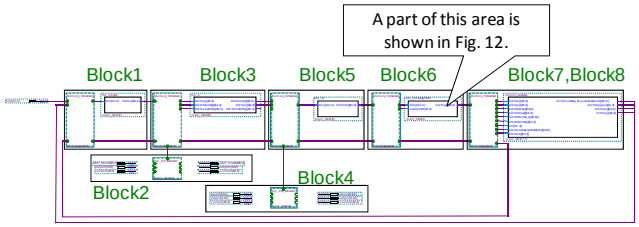


Fig. 13 Full program level pipeline circuit

Tbl. 3 Circuit size

Block #	+/−	×	÷	√	Ideal # of FPGAs	Actual # of FPGAs
1	22	41	0	0	1.8	5.5
3	37	69	36	4	6.4	19.4
5	32	60	5	1	3.2	9.6
6	238	229	61	0	17.7	53.1
7	384	380	98	3	29.1	87.5
8	255	330	47	1	20.4	61.3
Total	968	1109	247	9	78.9	236.7

Tbl. 4 Circuit size

Operation	Logic Cell Usage
+/−	1577.25
×	2785.5
÷	6831
√	6831

６．２　技術課題

本節では、6.1 節で述べた作業をする中で実験的に抽出された CFD プログラムの FPGA 回路化に関する技術的課題を列挙する。

(1)大量のデータ入出力

図８の Block8(以下、単に Block8 等と呼ぶ)では、

入力ﾃﾞｰﾀﾚｰﾄ=入力ﾃﾞｰﾀ数×1ﾃﾞｰﾀのﾊﾞｲﾄ幅×FPGA 動作周波数　・・・(式1)

から、

14.4G[ﾊﾞｲﾄ/秒]=18[変数]×8[ﾊﾞｲﾄ]×100M[Hz]

のデータ入力帯域が必要となることが分かった。3.2G[バイト/秒]のメモリモジュールを用いると仮定すると、５個のメモリモジュールを並列に FPGA へ接続し、帯域を確保する必要がある。これは、今回解析した結果の一部であるが、多くの場所で複数個のメモリモジュールを並列動作させて帯域を確保する必要があることが確認できた。

(2)回路規模と実装

表３に示したように、Block1 から Block8 までを FPGA 回路化した場合、237 個の Virtex4(XC4VLX80)が必要になることが分かった。内蔵 DSP の活用や、ロジックセル数の多いチップの採用等を考えることにより、FPGA チップ数を減少させることが必要と考えられる。また、図 13 のパイプラインは数万 clock のレイテンシを持つことが分かってきているが、演算データの到着待ちや条件分岐が存在するためパイプライン全体が常に動作しているわけではない。この動作特性と FPGA のリコンフィギャラブル性を利用し、ダイナミックに FPGA チップに回路を再構成することで、パイプラインを構成する FPGA チップ量を低減する方法の検討も課題の一つであると考えられる。

(3)メモリスケジューラの実装

図５の基本構成を持つ FPGA 計算機の場合、図 13 に示したパイプラインへ 1clock 毎に必要なデータを入力し続けるメモリスケジューラの実装が必要となる。この課題については一部研究が進んでいる[6]。

(4)ホスト計算機とのインターフェース

ホスト計算機が持つ外部機器との高帯域な通信技術を考えると、現時点では PCI-Express 32 レーンの 8.0G[バイト/秒]が挙げられる。一方で(1)で述べた通り Block8 で必要とするデータ入力帯域は 14.4G[バイト/秒]程度である。PCI-Express の実行性能がピーク値の半分の 4.0G[バイト/秒]程度だとすると、32 レーンの PCI-Express を 4 レーンセットにしてホスト計算機とのインターフェースを構築しなくてはならないが、これは物理的な接続構成や通信制御を考えると現実的ではない。これより、プログラムの一部を FPGA 回路化するのではなく、全体を FPGA 回路化し、ホスト計算機との通信は最初と最後のデータ受け渡し程度にすべきであると考えられる。

(5)浮動小数点演算器のチューニング

今回の検討では倍精度浮動小数点演算器の実行コアは、Xilinx Floating Point Operator を用いて生成した。倍精度浮動小数点演算器コアは回路全体に占める割合が高いため、小型化が望まれる部分である。独自の回路を作成することによる小型化が期待される。

(6)CFD 研究者による FORTRAN プログラムの FPGA 回路化

回路設計のノウハウを持たない CFD 研究者にとって、FORTRAN で書かれた CFD プログラムを FPGA 回路化するのは困難である。FOTRAN を HDL するコンパイラないしはトランスレータの開発が望まれる。または、FPGA を意識した高級言語仕様を策定し、CFD 研究者が FORTRAN でプログラミングを行うようにその言語でプログラミングをできるようにする方法も考えられる。いづれにしても、プログラム開発のための環境整備、コンパイラや統合デバッグ環境等の開発が待たれる。

(7)実行計算機を意識したソースプログラムのチューニング

プログラム開発環境に関する課題という意味で(6)と関連するが、既存の特定アーキテクチャの計算機での演算性能の向上を目標にしてチューニングされたプログラムをそのまま FPGA 回路化した場合、無駄な処理が多い回路になる傾向がある。FORTRAN でプログラミングする場合でも、FPGA 回路化を、つまり、データフローを意識したコーディングをすることが重要である。

(8)マクロの無い演算処理の実装

exp()や６乗根など、マクロが用意されていない演算処理の実装方法の検討が必要である。IP コアの CPU を活用する、DSP を利用する、ルックアップテーブル方式を採用する等の方法が考えられる。

６．まとめ

スーパーコンピュータで実利用されている圧縮性完全気体の三次元流れを計算する CFD プログラムの演算処理部分を、FPGA のパイプライン回路に手作業で置き換える中で、回路化に関する８個の技術的課題の抽出を行った。

FPGA 計算機の実現には多くの課題が存在することが分かったが、引き続き、プログラムの条件分岐部分の回路化とその性能検証を行い、プログラム全体としての FPGA 回路化の成立性を検証し、FPGA 計算機の実現に向けて研究を進めていく予定である。

参考文献

(1) 藤田直行、″ JAXA 次期スーパーコンピュータシステム″JSS″の設計思想と構成概要、″ 第 22 回数値流体力学シンポジウム論文集、I5-3、2008

(2) 新城淳史、松山新吾、溝渕泰寛、小川哲、梅村章、”液体燃料微粒化初期過程の数値解析、” 第 41 回流体力学講演会/航空宇宙数値シミュレーション技術シンポジウム 2009 講演集、pp.337-340、2009

(3) 高木亮治、藤田直行、松尾裕一、″JAXA Supercomputer System(JSS)の初期性能評価、″ 第 41 回流体力学講演会/航空宇宙数値シミュレーション技術シンポジウム2009講演集、pp.333-336、2009

(4) Fujitsu Limited、”ホワイトペーパー:富士通 SPARC64TMVII プロセッサ、” http://img.jp.fujitsu.com/downloads/jp/jhpc/sparc64vii-wpj.pdf, 2008

(5) R. Takaki, K.Yamamoto, T.Yamane, S.Enomoto, and J.Mukai, The Development of the UPACS CFD Environment, High Performance Computing, Proceedings of ISHPC, Springer, pp.307-319, 2003

(6) Hirokazu MORISHITA, Yasunori OSANA, Naoyuki FUJITA, Hideharu AMANO, Exploiting Memory Hierarchy for a Computational Fluid Dynamics Accelerator on FPGAs, ICFPT2008, pp.193-200