Vol.2011-SLDM-150 No.10
2011/5/19

高位合成ツールを利用した ハードウエアアルゴリズムの最適化

福井　啓[†1]　藤田昌宏[†2]

近年大規模・高速化が求められる、HPC(High Performance Computing) の分野において、順調な大規模化が進む FPGA(Field Programmable Gate Array) を用いて実現しようという研究が報告されている。本研究は FPGA を用いて、特定分野における数値計算の高速化の実現を目指したものである。一般に計算アルゴリズムを FPGA 上で実装するには時間がかかる、また大きな労力が必要である。本研究では主要な計算部分はデータフローグラフを書くことによってハードウエアを記述できる高位合成ツールを用いており、開発期間の短さを利用して様々な実装の比較をした。本発表では、津波のシミュレータの実装を通して、様々な実装やアーキテクチャを速度の観点から比較をする。

Optimization of Hardware Algorithms Utilizing a High Level Synthesis Tool

Akira Fukui[†1] and Masahiro Fujita[†2]

Recently, there are some reports which suggest the utilization of FPGAs for HPC(High Performance Computing). The aim of this research is to accelerate specific calculatons. Generally, Development of FPGAs requires a long development term and a lot of labor. In this research, the main calculation part is designed utilizing a high level synthesis tool. In this paper, some different implementations of TUNAMI(tsunami simulator) are compared in terms of speed.

†1 東京大学大学院 工学系研究科 電気系工学専攻
Dept. of Electrical Engineering and Information Systems, The University of Tokyo
†2 東京大学大規模集積システム設計教育研究センター
VLSI Design and Education Center, The University of Tokyo

1. はじめに

HPC(High Performance Computing) の性能向上は科学計算など、様々な分野で高まっている。これらの数値計算の高速化をハードウエアを用いて実現する研究が様々なアプローチで行われている。具体的には GPGPU(General Purpose computing on Graphics Processing Unit) を用いる方法、汎用プロセッサーを用いた分散処理をする方法、そして FPGA を用いた方法などが挙げられる。

近年、FPGA(Field Programmable Gate Array) の回路構成を変更可能な特性を利用して HPC に利用する動きがある。FPGA は現在のところ規模、動作周波数共に年々向上している。これは FPGA 自体が単純な構造の繰り返しであり、近年の半導体微細化の恩恵が直接性能向上につながるためである。FPGA は用途に応じた専用回路を組むことが出来るため、その性質を充分活用すれば高いパフォーマンスを発揮することが期待できる。

一方でハードウエアの設計には、ソフトウエアに比べて開発に時間がかかるという問題がある。またストリーム型のアーキテクチャに実装するためのアルゴリズムの変換、パイプライン化や最適化を意識して設計することが求められる。これらの問題を解決する技術として、高位合成ツールが注目されている。高位合成ツールとは、C/C++などソフトウエアプログラムにおいて広く用いられている言語やそれらを確証した言語を用いて、より抽象度の高い記述から RTL 設計の記述を生成するツールである。高位合成ツールを利用すると、通常の RTL から設計を始める場合に比べて比較的少ない記述量でハードウエアの設計が可能になり、開発期間の短縮が見込まれる。またパイプライン化や演算器の共有といった最適化が自動化されていることも開発効率向上への期待につながる。

本研究では FPGA 実装による津波のシミュレーターの高速化にを目指している。津波の予測は地震発生時の安全確保のために非常に重要であり、FPGA 上に津波のシミュレータを実装し、処理手順やアーキテクチャによる違いを速度の観点から比較した。また現在 GPU などのハードウエアを用いた津波のリアルタイムシミュレーションが注目されており[1)]、これらの研究との比較も行った。また、最終的な目標である TUNAMI シミュレーターの高速化について、今後の課題や方向性を議論した。

2. 津波のシミュレーション

2.1 Real-Time Simulation と Pre Simulation

津波のシミュレーションには以下に紹介する 2 つの方法がある。ひとつは Real-Time Simulation、もうひとつは Pre Simulation である。本研究では Pre Simulation に使用する事前シミュレーションの高速化、また FPGA を用いた Real-Time Simulation の実現を目指す。

Real-Time Simulation

Real-Time Simulation はその名のとおり地震発生後にシミュレーションを開始し、津波が岸に到達する前にシミュレーションを終えるモデルである。計算機の進歩とともに実現の可能性がでてきた。これまではスーパーコンピュータを用いないと演算が間に合わなかったが、最近は複数の GPU を並列に用いることでリアルタイムシミュレーションを実現した例も報告されている[1)]。

Pre Simulation

Pre Simulation は事前に多数の震源を想定したシミュレーションをしておき、地震発生後にその結果を重ね合わせることで津波の規模や到達時刻を予測するモデルである。重ね合わせの計算は比較的短時間で行うことができるため、津波到達前に予測が可能であることが可能である。しかし、実際とは違う震源でシミュレーションをしているため、どうしてもリアルタイムシミュレーションに比べて精度が落ちてしまうという弱点がある。

2.2 TUNAMI(津波シミュレーター)

本研究では東北大学の津波シミュレータ TUNAMI-N1(Tohoku University's Numerical Analysis Model for Investigation of Near-field tsunamis, No.1) を用いた。なお、このシミュレータは FORTRAN でかかれており、計算は単精度浮動小数点数で行われる。また TUNAMI-N2,N3 もあり、これらは水深の浅いところでは非線形長波、深いところでは線形長波を用いることにより、より精度の高い予測を可能にしている。本研究では TUNAMI-N1 の FPGA 実装を行うが、今後 N2,N3 の FPGA 実装による高速化も検討している。

現在、標準的な数値解析に用いられている数値モデルは、沖合いでは線形長波理論、沿岸では非線形長波理論を用いるものが採用されており、これまで津波予報や、津波被害推定など防災事業に積極的に利用されている。この理由として、線形問題における差分スキームの安定性が理論的に検討され、打ち切り誤差などの数値誤差の発生原因やその伝搬特性も明らかになっていること、津波到達時刻や沿岸での最高水位が精度よく計算できることなどが挙げられる[5)]。

3. GPU を用いた関連研究

GPU を用いて津波の計算を高速化したという報告は多数ある。文献[1)] では、単独の GPU(GTX295 x4, 480 core, 1200MHz) で CPU(Core i7 CPU920@2.67GHz) のシングルコアに比べて 62 倍の高速化が可能であることが報告されている[1)]。

加えて文献[1)] では、複数の GPU を用いて津波の Real-Time Simulation に成功したという報告がなされている[1)]。8 つの GPU を並列に走らせる、マルチ GPU での実装にも成功したとしている。複数の GPU を用いた場合、GPU や連動する CPU との通信時間が大きな遅延の要因となり、全体のパフォーマンスを低下させてしまうことが多い。青木らはまずシミュレーション対象の海域をわけ、それぞれの海域を別々の GPU で処理した。この際、境界付近をオーバーラップさせることで GPU 間の通信を減らしスケーラブルなマルチ GPU システムの実装に成功したと報告されている。

4. 高位合成ツールの利用

組み込みシステムや各種デジタルシステムの高機能化に伴い、システム LSI 設計における大規模化が進むとともに、開発規模も拡大し、開発工程を含めた複雑さが増している。こうした問題を解決する方法として、高位合成技術が注目されている。

本研究では Maxeler Technologies の高位合成ツールである MaxCompiler を用いた。これは、Java で記述されたデータフローグラフを入力すると、その設計を VHDL の記述に変換するツールである。VHDL からの FPGA の構成情報を生成するために、Xilinx 社のツールを利用している。また MaxCompiler では CPU 上で動作するホストプログラムを用いて、FPGA へのデータの入出力などの制御を行う。このホストプログラムは C 言語で記述する。

5. 津波シミュレータの FPGA への実装

5.1 津波シミュレータの計算のフロー

津波のシミュレーターの概略は図 1 のようになっている。全体は Input, Initial Condition, Main Loop の 3 つの部分にわかれている。以下に各ステップでの操作を説明する。

まず Input ではシミュレーションに必要なデータの入力を行う。今回扱うシミュレータは、二次元の座標で表される四角形の領域内での波の伝搬をモデル化したものである。ここではシミュレータに座標上の各点における静水時の水深の情報と、震源の座標・強さを入力する。

入力が完了したら Initial Condition において、これらの情報を元に座標の各点における初期波の高さ、線流量の初期情報を算出する。

初期波と線流量の初期情報の生成が終わったら、Main Loop に入る。ここでは現在の各点における波高と線流量情報を元に、次のタイムステップにおける各点の波高・線流量情報を算出する。この操作をあらかじめ指定した回数 (T 回) 繰り返す。TUNAMI-N1 のメインループは大きく 3 つの部分に分けられる。1 つめは Mass Conservation である。ここでは座標全域を対象に、次のタイムステップにおける波の高さを計算する。2 つめは Open Boundary Condition である。ここではシュミレーション対象領域の端のみを対象に、境界条件の計算を処理する。3 つめは Momentum Conservation である。ここでは座標全域を対象に、次のタイムステップにおける縦方向、横方向の線流量を計算する。今回の実装では、主にこの 3 つの操作を各タイムステップ毎に行った。これ以外にも各地域の最大波高を記録したりといった関数も存在するが、物理的にシミュレーションの本質に関わるのはこの部分である。なお、今回は TUNAMI-N1 で用いられる関数のみを紹介するが、TUNAMI-N2,N3 では浅いところでは非線形長波、深いところでは線形長波を用いるなど、より高度なモデル化を行っている。

最終的な津波の出力は各タイムステップ毎の各座標における波の高さである。実行と同時にファイルに出力する。

またこのシミュレータは汎用プロセッサー上での動作を前提として作られており、波の高さが一定値以下になると、波の高さを 0 として扱うなどの処理が含まれている。またその後の処理では、波の高さが 0 の場合は Mass Conservation や Momentum Conservaton などの処理を実行しないことになっており、計算コストの削減を図っている。

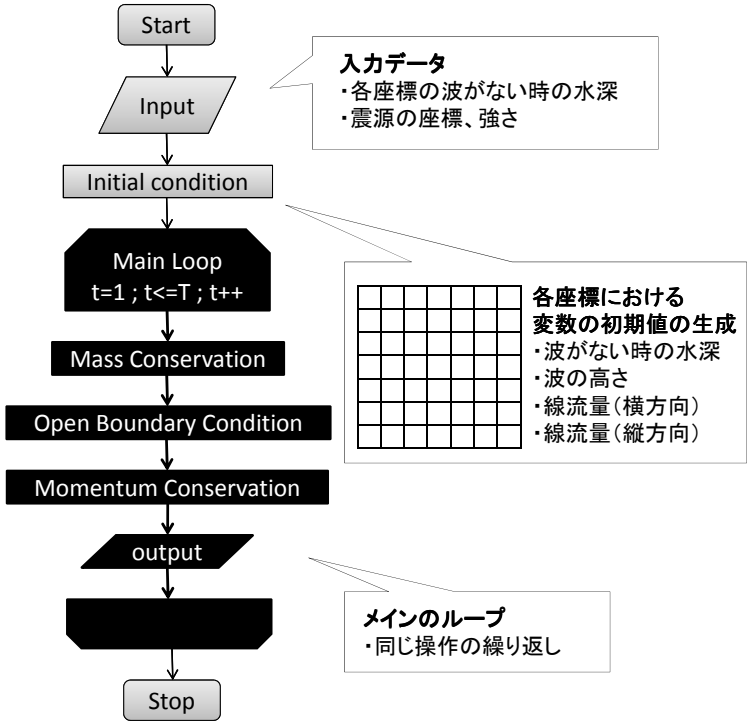

図 1 津波シミュレータの計算のフロー

5.2 ハードウエア化

FPGA は同じ処理を繰り返すような処理において高速化が期待できる。繰り返し構造が多いと、パイプライン化や処理の並列化がしやすいためである。今回の実装でもハードウエア化するのは、同じ処理を繰り返す変数の更新部分になる。

シミュレータの計算手順を考慮し、以下のようにソフトウエア、ハードウエア処理を行う。ここで、ソフトウエア処理は CPU 上で、ハードウエア処理は FPGA 上での処理を意味する。Input(座標上の各点における静水時の水深の情報と、震源の座標・強さの情報の入力) は初めに行うだけなのでソフトウエアで行う。また Initial(入力データからの初期波の高さ、

操作	所要時間[s]
入力	0.05[s]
初期波の生成	2.33[s]
メインのループ	429.9[s]

図 2 シミュレータのプロファイリング

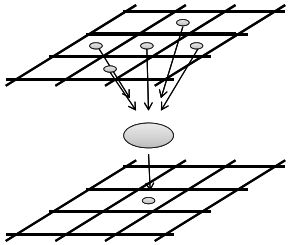

図 3 次のタイムステップの計算のイメージ

線流量の算出) も初めに一度行うだけなので、ソフトウエアで行う。そして Main Loop(次の各ステップ毎の波高、線流量の更新の操作) をハードウエアで行う。

実際にシミュレータを実行すると、各ステップの実行時間のうちわけは図 2 のようになっている。ハードウエア化する変数の更新部分が実行時間のボトルネックになっており、この部分を高速化するので大幅な速度向上が見込める。

5.3 ループの統合

ループに含まれる計算は典型的な convolution 計算であり、図 3 のように、1 つ前の時刻における、隣接する座標の波の状態を引数に次の時刻の波の状態を計算するものとなっている。

TUNAMI-N1 では、図 1 のメインのループ内の、Mass Conservation, Open Boundary Condition, Momentum Conservation はそれぞれが独立したループになっている。つまり、メインのループの内部にループが 3 つ含まれていることになる。また一つ目のループで算出された値を元に次のループで別の変数を算出するという順序関係があるため、前のループが

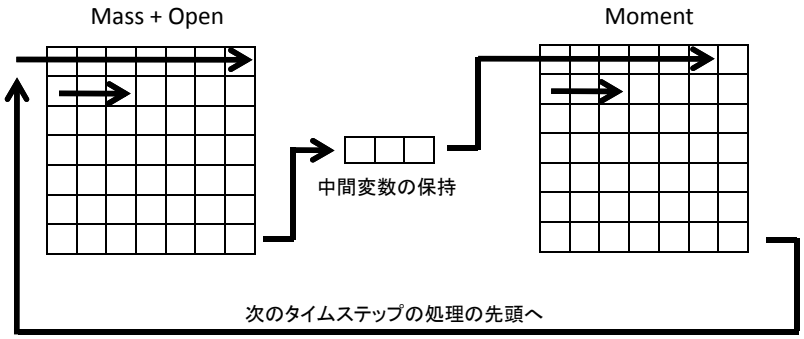

図 4 本実装における処理の手順

終わるまで次のループの計算ができない。このため一つのループの計算をしている間一度中間データを保持する必要がある。

FPGA において高速にアクセスできるメモリは BRAM だが、全てのループの中間変数を保持するとここがボトルネックになってしまう可能性がある。FPGA の開発においては、どれかひとつのリソースだけを使い過ぎるような実装は避ける必要がある。これはどのリソースにもそれぞれ制限があるためである。それぞれのリソースの使用状況を考慮し、全体のトレードオフを意識してチップ全体のパフォーマンスを向上させる方法を考えていく必要がある。

そこで本実装では、Mass Conservation と Open Boundary Condition の計算の間に依存関係がない点に注目し、2 つのループの統合を行った。一度のループで Mass Conservation, Open Boundary Condition 両方の計算をし、座標に応じて計算結果を選択する方式をとった。この結果、実装に要する FF、BRAM を削減することに成功した。なお、本実装における処理の順序を図 4 に示す。

6. 速度の比較

本実装と CPU との速度の比較をした。速度の指標にはシミュレーションを 1 タイムステップ進めるために必要な時間を用いた。まず CPU に関しては、元の fortran でかかれた TUNAMI-N1 を Intel Core i5 プロセッサー上で実行し、200 回のループの平均時間を用い

TUNAMI-N1 (i5, 1 core)	81.7[msec]
FPGA (Virtex 6 SX315T)	3.47[msec]

図 5 CPU と FPGA の速度の比較

LUTs	8.64% (16995/196800)
FFs	5.03% (19811/393600)
BRAMs	6.53% (46/704)
DSPs	1.19% (16/1344)

図 6 FPGA のリソースの使用状況

た。また FPGA に関しては、実行に必要なサイクル数 (1040*668) と本実装の動作周波数 (200MHz) から算出した。なおこの際、FPGA 外部との通信がボトルネックになっていないことは確認してある。結果は図 5 のようになった。パイプライン化とループの統合を行っただけの簡単な実装で CPU 比で 23 倍の高速化に成功した。

7. 今後の課題

7.1 FPGA に適した実装

ここでは今後のさらなる高速化の余地について議論する。今回の実装において使用した Virtex 6 SX315T の LUT, FF, BRAM, DSP のそれぞれの使用状況は図 6 のようになった。割合的に最も多く使用している LUT でも全体の 10 %以下しか使用していない。このことから、今後残りのリソースを有効利用することにより、更なる高速化ができると考えられる。

具体的には現在１つしかないパイプラインの本数を増やすアプローチが考えられる。この場合現在の実装に比べて、一時的に保存する必要のあるデータが増えたり、全体の制御が複雑になったりする可能性が考えられるが、リソースの使用上今日を考えると全体のパフォーマンスは大きく向上することが期待できる。

また青木らは、GPU 上での津波のシミュレーションにおいて、縦方向の線流量と横方向の線流量を別々に計算するなどの方法での実装について報告している。今後は FPGA により適した処理手順を考えていきたい。

7.2 計算の精度と最適化の余地

数値計算の精度と使用するビット幅には密接な関係がある。ビット幅が大きいほど切り捨てによる情報の欠落は小さくなる。

一方 FPGA ではビット幅が大きくなるほど多くのリソースを必要とする。使用するリソースが多くなると全体で実装できるパイプラインの本数に制限がかかる。このため、ビット幅を増加させることは高速化という観点ではマイナスにはたらく。

今後は TUNAMI のコードを分析し、精度についての考察をしていく。シミュレーションに単精度の float の精度が必要なのか、また部分的に bit 幅を削減する余地のある部分があるか、などの点に注目してさらなる高速化実現の余地を探っていきたい。

7.3 TUNAMI-N2,N3 の実装

TUNAMI-N2,N3 ではより高度なモデル化を行うために、浅いところでは非線形長波理論、深いところでは線形長波理論を用いるため、シミュレーションに際して別の式を適用する。

ソフトウエアでは条件分岐をして片方しか実行しないため、実効速度に大きな影響はないと考えられる。一方で FPGA ではダイナミックな処理ができないため、両方の場合を計算しておいてあとで選択する形式をとらざるを得ない。この場合、必要な演算器や FF などのリソースが増えてしまうため、高速化にあたって不利になる要因と考えられる。

7.4 高速化とデータ転送速度

FPGA 上での開発にあたっては、外部とのデータの入出力量が可能なデータ転送速度を越えてしまうことが原因で速度が頭打ちしてしまうことがしばしばある。

津波のシミュレータの出力は各タイムステップの各座標における波の高さである。本実装では 1 サイクルにつき 1 つの float 値が出力される。シミュレーションの出力としてこれらの値を FPGA から CPU やメモリに転送する必要がある。現在の速度は 4[byte] × 200[MHz]=800[Mbyte/s] となっており、今後 FPGA 上の残りのリソースを活用した高速化に成功した場合、メモリ転送速度の上限に抵触する可能性がある。

Vol.2011-SLDM-150 No.10
2011/5/19

8. 結　　論

本論文ではFPGAを用いた津波シミュレータの高速化における、処理手順や実装について速度の観点から議論した。

FPGAに津波シミュレータTUNAMI-N1を実装し、チップ全体の5 %から9 %程度のリソース使用率にもかかわらず、CPU比で23倍の高速化に成功した。青木らによる津波のシミュレーションに関する報告によると、GPUを用いるとCPU比で62倍の高速化が可能であると報告されている。今後は残っているリソースを利用し、より高い並列性を実現するためのアルゴリズムの変換、不要な演算の削除や統合をはじめとする最適化処理などを通してさらなる高速化を目指していきたい。

参 考 文 献

1) M.C. Acuna, T. Aoki, "Real-time Tsunami Simulation Accelerated by Parallel GPUs," *Processings of Computational Engineering Conference*, Vol.14, pp.307-310, 2010.
2) K. Shunichi, K. Kosuke, S, Yusuke, "Real-time tsunami simulation using GPU," **土木学会論文集**, Vol.66, No.1, pp.191-195, 2010.
3) 今村文彦, 後藤智明, "差分法による津波数値計算の打ち切り誤差" **土木学会論文集**, No.375/2-6, pp.241-250, 1986.
4) 佐山順二, 今村文彦, 後藤智明, 首藤伸夫, "外海域における津波の高精度計算法に関する検討" **土木学会論文集**, pp.177-181, 1987.
5) 佐山順二, 今村文彦, 後藤智明, 首藤伸夫, "津波数値解析における分散波理論モデルの適用性と新しい数値計算法の提案" **土木学会論文集**, Vol.63, pp.55-66, 2007.