2007 年 11 月 28 日

サイエンティフィック・システム研究会
科学技術計算分科会 第 2 回会合資料

リコンフィギャラブルシステムの ハイパフォーマンスコンピューティングへの応用

慶應義塾大学 天野英晴
E-mail: hunga@am.ics.keio.ac.jp

アブストラクト

リコンフィギャラブルシステムは、FPGA などプログラマブルな素子を用いてアプリケーションをハードウェアとして実装することで、高速かつ柔軟性に優れたシステムを実現する。従来、FPGA 内部の演算素子は PC などに用いられる CPU の浮動小数点演算装置に比べ性能が 2 桁低く、高性能計算への応用は困難であった。しかし最近の高性能 FPGA の登場により、浮動小数点演算を用いた本格的な科学技術計算の高速化の可能性が現実味を帯びている。最近の実装例について報告し、その将来性と問題点を議論する。

キーワード

FPGA、リコンフィギャラブルシステム、高性能演算

1 はじめに

リコンフィギャラブルシステムは、FPGA を代表とする書き換え可能なデバイス上に対象とするアルゴリズムを直接ハードウェア化して実行することにより、ハードウェアの高性能とソフトウェアの柔軟性を共に実現することを狙ったシステムで、近年の FPGA の急速な発展と共にその応用分野を広げている [1]。従来、FPGA は、CPU 内で利用されている専用演算器や ASIC に対して性能面で 2 桁劣り、量産時のコストも、ASIC や組み込み CPU に比べて高価であった。このため、リコンフィギャラブルシステムは、画像処理、音声処理、パターンマッチング、暗号解読等通常の PC や組み込み用 CPU では性能が不足し、ASIC で専用ハードウェアを作成するには製作チップ数が少なく開発コストを回収できない分野に限定的に用いられてきた。
しかし、最近、以下の変化によりその応用分野が急速に広がっている。

- Xilinx 社 Virtex シリーズ、Altera 社 Stratix シリーズに代表されるハイエンドの FPGA は、その構成要素である LUT(Look Up Table) 数の増大により、100 万ゲートを越えるランダムロジックの実装が可能になった。一方、大規模なメモリ、DLL、乗算器、DSP、CPU、ハイスピードリンクなどハードコアマクロをチップ内に組み込むようになった。この乗算器、DSP は整数演算のみが可能であるが、周辺論理回路と組み合わせれば、浮動小数点演算回路を多数搭載することが可能である。また、内部メモリも，大規模演算を行うために必要なデータを格納するのに十分な程大きくなった。このため、ハイエンドの FPGA を多数用いて汎用のスーパコンピューティングを行うことが性能面で可能となった。
- Xilinx 社 SPARTAN シリーズ、ALtera 社 Cyclone シリーズに代表されるローエンドの FPGA は、コストダウンの徹底と最新プロセスの利用により数万ゲート搭載可能なチップが数百円のオーダで利用可能となっている。これらにより、処理の中心に FPGA を用いた組み込みシステムが、家電、車載、携帯機器などでコスト面で可能となった。

有名な Hennessy と Patterson のテキスト [2] には以下の指摘がある。「テクノロジの改善が一定の率であっても、新しい能力を可能にすることができる一定のしきい値に達すると、その影響が飛躍的に大きくなることがある。」FPGA はここ数年でこのしきい値を越えつつあり、その応用分野は今後、急速に広がり、マイクロプロセッサや ASIC に並んで、IT 機器の主役の一つとなることが予想されている。
本稿では、このうち、ハイエンド FPGA を用いたリコンフィギャラブルシステムのハイパフォーマンスコンピューティングの応用に関して最近の動きを紹介し、その問題点を延べ、一つの解決法として我々の ReCSiP プロジェクトについて紹介する。

2　リコンフィギャラブルシステムによる高性能計算

2.1　FPGA の登場とハイパフォーマンスコンピューティングへの応用

FPGA(Field Programmable Gate Array) は、その柔軟性によりいくつかのタイプに分類できるが、このうち論理機能と配線情報を記憶するのに SRAM を用いる方式は、通常の CMOS プロセスを利用できることから 1990 年代以降に大発展を遂げ、その速度はいささかも衰えを見せていない。Xilinx 社は、同社の初期の代表的な製品である XC4000 シリーズを発売した 1991 年を SRAM 型の構成が確定した年と位置付けており、2005 年までに論理回路数で 200 倍、速度は 40 倍、低消費電力化で 50 倍、低コスト化で 500 倍を達成したとしている [3]。
SRAM 型 FPGA の発展と共に、これらの FPGA を多数接続して、大規模演算を高速化するシステムが登場した。初期のシステムで有名なのは 1991 年、92 年に米国計算機科学センター (旧 SRA) で開発された Splash, Splash2[4] である。このシステムは、Xilinx 社の XC4010 を 16 個直線状に接続し、さらにクロスバスイッチで相互接続して構成したクラスタ 3 つをインタフェースボードに接続した構成を持っている。この構成は、データをパイプライン的に流しながら計算していくアルゴリズムに特化しており、このため、主としてシストリックアルゴリズムによるパターンマッチングや画像処理に用いられた。Splash は DNA 塩基配列間のマッチングで当時の代表的スーパーコンピュータ CRAX-2 の 330 倍の性能を発揮して注目された。日本でも神戸大学の RM I-IV[5] を代表とする実験機が各地で開発され、NTT では、通信制御用に ATTRACTOR などの大規模システムが開発された。1999 年には三菱電機から商用大型機 RASH[6] が発売された。RASH は、Altera 社 FLEX10K100A を 8 チップ 1 ボードに搭載し、それぞれを格子状に直結配線し、さらにバスによって SRAM と接続した構造を持つ。FPGA 複数に大規模な対象回路を搭載して、SRAM を共有することが可能であり、DES の秘密鍵探索処理や合成開口レーダの画像生成処理に利用された。
90 年代に開発された FPGA を複数用いた大規模システムは、商用化された例はあるものの、いずれも少数生産の実験機レベルであり、用途もパターンマッチング、画像処理、秘密鍵探索処理など、通常のスーパーコンピュータが得意としないアプリケーションに限定されていた。

2.2　FPGA の発展と最近の動向

2000 年代に入り、FPGA はプロトタイピングおよびハイパフォーマンス処理用のハイエンドの製品 (Xilinx 社の Virtex II, IV, V, Altera 社の Stratix-II,III) と組み込み用のローエンドの製品 (Xilinx 社の SPARTAN 2/3, Altera 社の Cyclone-II/III) に分化が進んだ。さらに、Xilinx 社は、Virtex IV で論理回路を重視した LX, 演算機能を重視した EX, ハイパフォーマンスリンクと CPU を内蔵してネットワークコントローラを意識した SX の 3 シリーズを設け、さらに細かく分けた応用分野に特化した製品を市場に投入した。このハイエンド製品を利用することでハイパフォーマンス処理への可能性はさらに広がった。
CRAY 社は、AMD Opteron 2 Node および Xilinx 社 Virtex II Pro を強力なスイッチで接続した構成のボードを多数、高速ネットワーク Rapid Array により結合したシステム CRAY-XD1 を開発した。このシステムはクラスタ型のコンピュータの各ノードに FPGA が付いていて特定の演算を加速できる構成であり、FPGA に処理を分散して交通シミュレーション [9] や画像処理を行う事例が報告されている。また、SGI 社の RASC[10] は、NUMA 型マルチプロセッサの各ノードに Virtex を接続したシステムで、Virtex から直接分散共有メモリがアクセスでき

る。これらのシステムは、汎用のスーパコンピュータの各ノードに FPGA を持ち、処理の一部を分散して高速化を行うもので、数値演算自体は、それぞれのノードで行われ、FPGA はパターンマッチングなど特殊な処理の一部をオフロードする目的で使われている。このようなホスト PC と FPGA との接続をより密にするため、Xilinx 社は Intel FSB(Front-Side Bus) に直結する Virtex-5 のボードを開発している [3]。
これに対して、最近のハイエンド製品を利用して、浮動小数点演算自体を FPGA により実行させるシステムが開発され始めている。PROGRAPE-3[7] は、天体を対象とする粒子系シミュレーションを目的として開発され、1 ボードに Virtex II を 4 個+接続用 1 個を搭載し、これらを多数接続することが可能である。PROGRAPE-3 は、浮動小数点演算パイプラインを用いて粒子間の相互作用の演算を専用に実行する。東北大の RAPLAS[8] もレイトレーシング用に浮動小数点演算を FPGA により高速化しており、後の章で紹介する我々の ReCSiP プロジェクトでも、細胞シミュレーション用の浮動小数演算を FPGA により高速化している。Berkeley 大学の BEE2[11] は、Virtex II FPGA を 5 個搭載したボードであり、ハイパフォーマンス用リコンフィギャラブルシステム用の汎用テストベッドとして様々な機関で利用されている。BEE2 は、将来の大規模並列計算機システムのアーキテクチャを広く研究開発する RAMP プロジェクトで採用されている。このプロジェクトで開発された RAMP-Blue は、BEE2 ボードを 8 枚接続した大型システムで、FPGA 中のソフトプロセッサでマルチプロセッサを構成すると共に、他の論理回路部分を用いてハイパフォーマンス処理を行うことができる。
他にも、FPGA 上で効率の良い浮動小数点演算回路を作る方法、これらを用いた数値演算への応用事例は、最近の FCCM, FPL などの国際学会で多数発表されている。これら最近のリコンフィギャラブルシステムのハイパフォーマンス処理への応用は、12 月 11 日から北九州国際会議場で開催される FPT2007 において George Washington Univ. の Tarek El-Ghazawi 教授によるチュートリアルでより詳しく紹介される予定である [12]。ご興味のある方はぜひ参加されたい。

2.3 FPGA によるハイパフォーマンスコンピューティングの問題点

2.3.1 浮動小数点演算器の問題

かつて FPGA 上で浮動小数点演算を行った場合、CPU 上の演算器や ASIC 上の演算器に比べて 1 桁-2 桁遅いと言われた。この状況は、FPGA 上に整数乗算器、積和演算器の専用回路が搭載されてから改善されたが、それでも FPGA 上の浮動小数点演算器は、これらの専用回路とランダムロジックで実装されるため、動作速度の点で専用の浮動小数点演算回路に比べて不利なのは否めない。CPU の浮動小数点演算器が 2-4GHz の周波数で動作するのに対して、FPGA 上で浮動小数点演算器を作るとどうしても 200MHz 程度の動作周波数となる。もちろんソフトウェアにより動作する CPU の浮動小数点演算器に比べ、FPGA 上では、メモリや他の浮動小数点演算器間でデータを直接やりとりすることができる利点はあるものの、やはり単体同士を比較すると数倍は遅いと考えられる。さらに、リコンフィギャラブルシステムは、ホスト PC のアクセラレータとして用いるため、同程度の動作速度では要求を満足することができない。FPGA 数個を用いたシステム上で、通常の PC の最低数倍の性能を実現する必要がある。ハイエンドの FPGA は一チップ数万円するため、上記の性能が実現できなければ、通常の PC や、PC を用いた安価なクラスタに対して性能価格比の点で優位に立つことができなくなり、リコンフィギャラブルシステムによる性能の加速は、現実的ではなくなる。
そこで、以下の方法が採用されている。

- IEEE 標準のフォーマットに準拠せず、その対象アプリケーションに適合した精度で仮数部と指数部を実現する。また、丸め処理を単純化する。RAPRAS、PROGRAPE,ReCSiP ではいずれもそれぞれ独自のフォーマットを用いている[1]。
- 演算パイプラインを細かく設定し、複数のパイプラインが並列に動作するように、演算処理を工夫する。

このような工夫がうまくいけば、FPGA 1 個から 4 個で PC の数倍から数十倍の性能を実現でき、アクセラレータとしての役割を果たすことができる。

[1] ReCSiP における確率モデルシミュレーションでは IEEE 標準をフルサポートしたがこのため性能はある程度犠牲にしている

ちなみに、現在の整数演算器に代わって浮動小数点演算器をハードコアとして搭載した FPGA が発売される予定は今の所は存在しない。これは、開発費を賄うだけの販売量がハイパフォーマンスコンピューティングの市場では望めないこと、高速で動作可能な浮動小数点演算器を搭載したとしても、周辺回路の動作周波数はやはり 200MHz 前後となることが予想され、大きな性能向上が見込めないこと。などが理由である。浮動小数点演算器を多数搭載したリコンフィギャラブルデバイスの商用化が望めるとすれば、FPGA の延長線上ではなく、ClearSpeed[13] などの SIMD アクセラレータに演算器間接続に柔軟性を与えること、あるいは現在でも粗粒度構成要素を用いている動的リコンフィギャラブルプロセッサ [15] に浮動小数点演算器を搭載することから始まるのではないかと予想される。

2.3.2 プログラミングの問題

現在のハイパフォーマンスコンピューティング用のリコンフィギャラブルシステムは、HDL(Hardware Description Language) で内部の回路を作り込んでいるのが普通である。これは、上記に述べたように、アクセラレータとして意味のある性能を出すためには、複数の浮動小数点演算器パイプライン間に効率良くデータを流してやらなければならないためである。しかし、この場合はユーザは既に出来上がったシステムのみを用いることになる。PROGRAPE における粒子シミュレーション、ReCSiP における細胞シミュレーション、RAPLAS におけるレイトレーシングなどがこれに当たり、ユーザによってはこれで十分な場合もある。しかし、このようなユーザであっても、いつプログラムの変更、拡張が必要になるかもしれない。そもそもユーザによってまったく手を入れることのできないシステムは、リコンフィギャラブルシステムの利点の一つである柔軟性を殺してしまうことになる。

ハードウェアに詳しくないユーザが、簡単にプログラミングする方法がない、という点がリコンフィギャラブルシステムをハイパフォーマンス分野で用いる場合の大きな足かせになっている。これに対してはいくつかの方法が試されている。まず、分野を限定すれば、エンドユーザにプログラムの自由を与える方法をサポートすることができる。我々の ReCSiP プロジェクトでは、後に紹介するように SDML という反応式記述言語からハードウェアを吐き出すことのできるシステムを構築している。PROGRAPE はアセンブラに似た形式で演算を記述することで、粒子シミュレーションを越えた科学技術計算用の回路を生成できるシステムを提案している [14]。

より汎用的な方法としては、マルチメディア処理などでは既に一般的になっている C レベル記述を用いる方法が考えられ、既に Bach-C[19] などでは浮動小数点演算器を含む回路の C レベル記述から自動生成することができる。現在の所、これらの言語処理系は、機能合成能力が低く、FPGA 上で性能を出すための並列化と一定の FPGA 規模に収めるための逐次化は、完全にプログラマがコントロールする必要がある。このため、問題の記述は HDL よりも容易であっても、一般のプログラマにとっては依然として負担が大きい。Cyber[20] など、ASIC 設計用に高度の機能合成能力を持つツールが、FPGA のハイパフォーマンスコンピューティング用にチューンアップしてくれれば、プログラマの負担は減らせる可能性があり、このことは現在でも技術的には不可能ではない。しかし、ハイパフォーマンスコンピューティングにおけるユーザプログラムの市場規模の問題で、現在のツール作成者がこの点に取り組んでくれるかどうかは疑問である。

もう一つの方法は、並列計算機用の自動並列化コンパイラ、MPI などの通信ライブラリ、OpenMP などのプラグマの利用である。しかし、これらのプログラムパラダイムは、仮想化されたメモリと、仮想化された計算資源を仮定している点で、リコンフィギャラブルシステムに適用する場合は FPGA のサイズとメモリに合う回路を吐き出すことが難しいだろう。

プログラミングは、リコンフィギャラブルシステムをハイパフォーマンス処理に用いる場合、おそらく最大の問題である。しばらくは、対象分野毎にエンドユーザにも利用可能なツールを作り、後々は、ハードウェア記述 C、システム記述 C の普及と科学技術計算分野への進展を期待しつつ、関連分野で研究を行っていくのが現実的であろう。

2.3.3　スケーラビリティの問題

リコンフィギャラブルシステムによるアクセラレータは、対象とする問題に応じて最適な回路構成を実現するのが理想である。しかし、対象の問題に回路が依存するということは、逆に対象問題が大きければ、搭載する FPGA の規模を越えてしまうことが容易に起きうる。多数の FPGA を密結合するシステム上では、一つの FPGA の規模を越えた問題をどのように分割するかも困難な問題である。
これに対する解決法は、プログラムの問題と関連し、一筋縄では行かない。BEE2 などでは、FPGA のサイズは、多くの問題を扱う単一タスクが十分格納できる程大きいと考え、分割されたタスクを FPGA 単位に割り当てる。この考え方は最近の FPGA の規模と、その規模を一杯にする回路の配置配線に要する時間などを考えると一定の説得力がある。しかし、この考え方でもタスク数がシステムの FPGA 数を越えると問題が生じ、本格的なハイパフォーマンスへの応用の一般的な解決法とはいえない。
現在の所、もっとも有効は方法は、対象問題を限定してサイズに依存せず問題を解くことのできる回路を搭載することで、ReCSiP、PROGRAPE、RAPLAS などでは基本的にはこの方法を用いている。すなわち、システムに搭載可能なサイズの回路を繰り返し用いて、サイズの大きなアプリケーションを扱う方法で、回路自体を大きなサイズを扱えるように設計するわけである。しかし、これは対象問題を限定し、設計者の努力の結果実現されており（このような手法を見つけること自体が研究の対象である)、一般的な問題に対してこのような回路を実装できるわけではない。
Bach-C などの C レベル記述では、設計が FPGA のサイズに入るかどうかは、設計者が、配置配線を繰り返し行って調整する必要がある。将来、機能合成ツールによるハードウェア見積りと、最適化のステップが整備されれば、アプリケーションのサイズが決れば、それをリコンフィギャラブルシステムに適合させる設計が容易になるだろう。その場合でも、メモリも計算資源も仮想化されたプロセッサのプログラマにとって、スケーラビリティの問題を考慮しつつ設計することは、かなりの負担になるだろう。
一方、FPGA などのリコンフィギャラブルデバイスに対してシステムサイズを意識しないですむように仮想化させる機構を取り付ける研究は、90 年代から行われていた。構成情報メモリを複数チップ内に持たせて実行中に利用していないメモリに外部から構成情報をロードする仮想ハードウェア技術 [16] は、1992 年に既に提案され、Xilinx 社からもこの機能を持ったマルチコンテキスト FPGA が提案されている [3]。しかし、これらの機構は粗粒度の動的リコンフィギャラブルプロセッサでは一般的になったが、FPGA には導入されていない。これは、LUT などの細粒度構成要素を用いている FPGA の場合、構成情報メモリを複数持たせる場合のメモリに要する面積が、それによって実現される論理回路に対して大きく、マルチコンテキスト化するメリットが小さいためである。これに代わって、FPGA をいくつかの領域に分割して、アプリケーションを稼働させながら、部分的に再構成を行う部分再構成が、特に最近の FPGA において一般的になっている。しかし、この方法は、外部から構成データを設定するのに時間がかかることから、回路構成を仮想化することは、現実的とはいえない。やはり、FPGA のサイズの問題はツールレベルで解決する以外にないように思う。

3　細胞シミュレータ ReCSiP

3.1　プロジェクトの目的

ReCSiP プロジェクトは、2003 年から慶應義塾大学、長崎大学、ERATO(科学技術振興機構北野共生システムプロジェクト) の共同で行われているプロジェクトで、数 10 万円規模のリコンフィギャラブルボードを、通常の PCI バスに装着することにより、クラスタコンピュータ並の速度で細胞シミュレーションを実現するシステムを開発することが目的である。現在、ReCSiP ボードは三世代目の ReCSiP-3 が開発されており、API、常微分方程式を解く解析的なアルゴリズムの他にモンテカルロ法を用いるアルゴリズム、フロントエンドのコンパイラ、スケジューラなどの開発が終わっている。
リコンフィギャラブルシステムの立場から考えると、ReCSiP プロジェクトは現在、ハイパフォーマンスコンピューティングにおいてリコンフィギャラブルシステムの陥っている問題に以下のように解決法を与えることを

狙っていた。

- 反応式を記述する言語である SBML の記述から、直接細胞シミュレーション用のハードウェアを生成することにより、エンドユーザが簡単に自分の問題を記述することができるようにする。
- 生成されたハードウェアは、対象とする問題にできるかぎり合わせて高速化する一方、サイズの点でスケーラブルなものであるようにする。また、ユーザの要求する計算精度を維持する。
- ホスト PC の 10 倍以上の性能を達成する。

3.2 プロジェクトの目的

本稿の読者は、細胞シミュレーションそれ自体には興味がないと思われるだめ、プロジェクトの詳細に関しては参考文献をご覧いただくことにして、その概略は以下の通りである。

ハードウェアボード ハードウェアボードは、Virtex II を用いた ReCSiP、Virtex II Pro を用いた ReCSiP-2, Virtex IV を用いた ReCSiP-3 が稼働している。図 1 は ReCSiP-2 のボード写真で、図 2 はそのブロックダイアグラムである。ReCSiP ボードは、メモリへの同時アクセス性能を強化するため、SSRAM 複数個を大規模 FPGA(Vertex XC2V8000) に接続している。ホスト PC とは QuickLogic 社のアンチヒューズ型 FPGA を用いて 64MHz の PCI バスへの接続を可能としている。シリアルリンクを用いてボード間の直接接続も可能である。ReCSiP ボードは、目的に適合した商用ボードが存在しなかったことから開発したが、それ自体大きな特徴を持つものではなく、プロジェクトで開発した細胞シミュレータは、十分な大きさの FPGA を搭載しており、PCI バスとの接続が可能であれば、他のボードにも移植することができる。

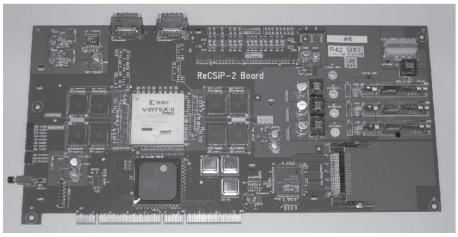

図 1: ReCSiP-2 の写真

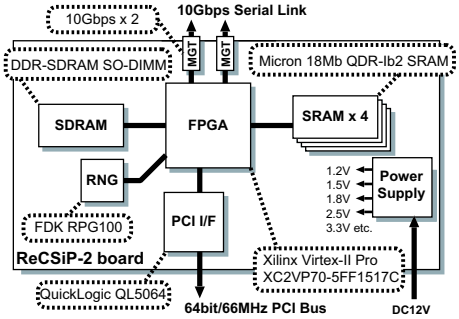

図 2: ReCSiP-2 のブロックダイアグラム

常微分方程式の直接解法によるシミュレータ 生化学シミュレーションは常微分方程式を解くことに帰着されるが、これを数値積分により解く方法を試みた。図 3 で示す単精度の浮動小数点加算器、乗算器から成る積分モジュールを転送スイッチで接続したシステムを対象問題毎に生成する。

ソルバーシステムは、複数の積分モジュールを並列に稼働させることができるが、これだけではパイプラインをフルに埋めることができず、ホスト PC に対して目覚ましい性能向上を達成することができない。しかし、幸いにして細胞シミュレーションは、パラメータサーベイが目的であるため、同一の反応式を複数のパラメータで解く場合がほとんどである。この点を利用し、複数のパラメータに対応するタスクをパイプラインに流して、これをフルに埋める方法を導入した。このことにより、ソルバーとその動作を作り込めば、ホスト PC の 80 倍程度の性能向上を達成することができる [17][18]。

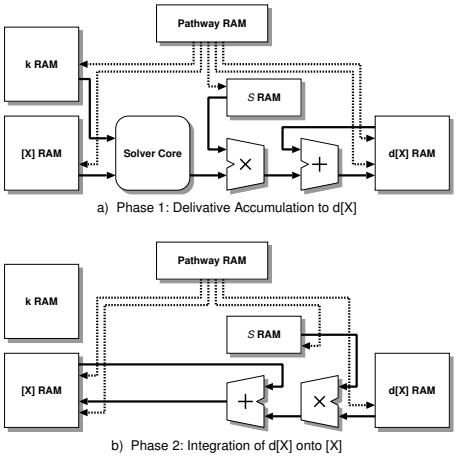

図 3: 積分モジュールの一例

確率モデルに基づくシミュレータ　常微分方程式を直接解く方法に比べて、モンテカルロ法に基づく確率モデル生化学シミュレーションは計算量は大きいが、精度、安定性の点で利点がある。この方法の基本的な手法である FRM(First Reaction Method) を実装した。こちらの方法はデータユニットと浮動小数点演算ユニットを接続したアーキテクチャを生成する。確率モデルに基づく方法は高い並列性を持つため、こちらもホスト PC の 80 倍の性能向上を実現した [21][22]。確率モデルに基づく方法については、効率の良い NRM(Next Reaction Method) への拡張 [23] を行い、ここでは IEEE Standard を浮動小数点演算を実現しつつ PC の 8 倍の性能を達成している。さらに、様々な構成の比較とスケーラビリティの強化を行っている。

シミュレータの自動生成　バイオインフォマティクスの研究者が ReCSiP を用いるためには、ハードウェア記述言語はもとより、プログラミング言語を用いることも避けることが望ましい。そこで、反応式を記述する標準言語である SBML(System Biology Markup Language) より直接ハードウェアおよびスケジューラを生成するシステムを開発した [24]。このシステムは、常微分方程式の直接解法を行うシステムのハードウェア記述、対応するホストのプログラム、内部メモリデータを自動生成する。システムの全体構成を図 4 に示す。残念ながら人手でチューンアップする場合よりも自動生成された回路は性能が落ちるが、それでも複数パラメータのタスクを同時実行することでパイプラインを埋め、ホスト PC の 10 倍程度の性能向上を実現している。

4　おわりに

ハイパフォーマンスコンピューティングへのリコンフィギャラブルシステムの適用について、最近の事例を紹介し、問題点を述べ、我々の ReCSiP プロジェクトの概略を紹介した。現在の所、FPGA1 個または数個にパイプライン化した浮動小数点演算器を数 10 個搭載し、パイプラインがフルに埋まるタスクを流せば、最新 PC の 10 倍以上の性能向上を得ることが可能な情勢になっている。生化学シミュレーション、粒子シミュレーション、レイトレーシングなどの分野で強力なアクセラレータを開発することに成功しており、今後、より分野が広がっていることが期待される。

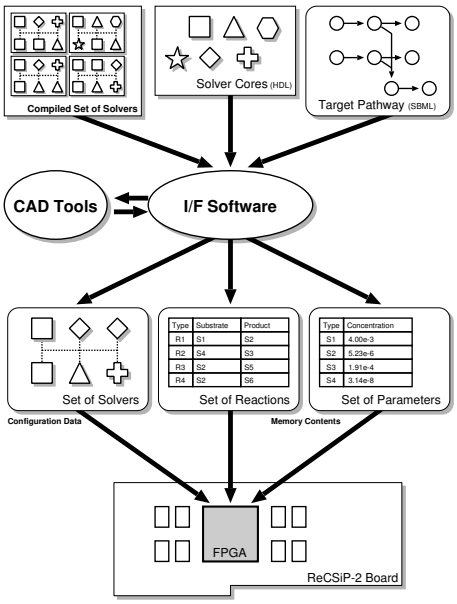

図 4: ReCSiP システムの全体構成

しかし、このようなシステムを効率良く実現するためにはハードウェア記述言語によるチューニングが必要で、汎用的なプログラムの枠組みから高い性能の科学技術計算用の回路を吐き出すためには、まだ問題が多い。リコンフィギャラブルシステムをハイパフォーマンスコンピューティングに本格的に応用するためには、まずは様々な分野で ReCSiP, PROGRAPE, RAPLACE のようなプロジェクトを行い、経験を積んで裾野を広げて行く方法が現実的と考えられる。

参考文献

[1] 末吉敏則、天野英晴　編著 “リコンフィギャラブルシステム,” オーム社, 2005.

[2] J.L.Hennessy, D.A.Patterson: “Computer Architecture: A Quantitative Approach the 4th Edition,” Morgan Kaufmann, 2006.

[3] S.Trimberger: “Redefining the FPGA,” Keynote speech on FPL2007, Aug. (2007)

[4] J.Arnord, D.Buell, E.Davis: “SPLASH 2,” in Proc. the ACM Symposium on Parallel Algorithms and Architectures, pp.316-322, (1992).

[5] 富田昌宏、菅原直昭、澄川文徳、平野浩太郎, “汎用エンジン RM-II とその構成” 並列処理シンポジウム JSPP 論文集 pp.151-158, (1993).

[6] 浅見広愛、飯田全広、中島克人、森伯郎、“FPGA ベース並列マシン RASH での DES 暗号解析処理の改良” 情報処理学会論文誌 Vol.41, No.SIG(HPS 1), pp.50-57, (2000)..

[7] 中里直人、濱田剛、“FPGA による流体シミュレーションの高速化” 信学技法 RECONF2005-80,(2005)..

[8] Y.Kaeriyama, et.al, “Hardware for a ray tracing technique using plane-sphere instersections,” Proc. of FPL2006, pp.315-320, (2006).

[9] Justin L. Tripp, Henning S. Mortveit, Anders A. Hansson, Maya Gokhale “Metropolitan Road Traffic Simulation on FPGAs,” Proc. of FCCM2005, (2005).

[10] “SGI RASC Technology,” http://www.sgi.com/products/rasc/

[11] J.Wawrzynek, “Adventures with a Reconfigurable Research Platform” Keynote, in FPL2007, (Aug. 2007).

[12] Tarek El-Ghazawi , “High Performance Computing on Reconfigurable Systems,” Tutorial of FPT2007, to appear (Dec. 2007).

[13] “ClearSpeed Whitepaper: CSX Processor Architecture,” http://www.clearspeed.com/

[14] T.Hamada and N.Naksato, “Pgr: A software package for reconfigurable super computing,” Proc. of FPL 2005 (2005).

[15] H.Amano, “A survey on Dynamically Reconfigurable Processors,” IEICE Trans. on Comm, pp.3179-3187, Dec. (2006).

[16] X-P. Ling and H.Amano, “WASMII: A Data Driven Computer on a Virtual Hardware ” Proc. of FCCM pp.33-42, Apr. (1993).

[17] 長名保範、他 “FPGA を用いた汎用生化学シミュレータ ReCSiP”, 電子情報通信学会論文誌, Vol. J89-D No. 6 pp. 1163-1172. (2006).

[18] Y.Osana and et.al., “Performance Evaluation of an FPGA-based Biochemical Simulato r ReCSiP ” Proc. of FPL pp.845-850, Aug. (2006).

[19] Bach-C 言語マニュアル、シャープ株式会社, 2004.

[20] 若林一敏、LSI 開発はこう変わる C 言語設計の最前線（1）、
http://techon.nikkeibp.co.jp/NEWS/dac2003/030317_1.html

[21] 吉見真聡　他 “FPGA を用いた確率モデル生化学シミュレータ” 情報処理学会論文誌 Vol.48, No.SIG 3 (ACS17) pp.45-58, (2007).

[22] M.Yoshimi and et.al., “An FPGA Implementation of High Throughput Stochastic Simulat or for Large-Scale Biochemical Systems ” Proc. of FPL pp.227-232, Aug. (2006).

[23] M.Yoshimi and et.al., “FPGA Implementation of a Data-Driven Stochastic Biochemical Simulator with the Next Reaction Method ” Proc. of FPL Aug. (2007).

[24] Yow Iwaoka and et.al., “An Acceleration of a Biochemical Simulator on Programmable Hardware,” The Seventh International Conference on Systems Biology(ICSB2006), Sep. (2006).