1C－8　高級言語からFPGAへの部分コンパイルによる高速化技法の研究

齊藤正伸†　多田好克†
電気通信大学大学院情報システム学研究科††

1　はじめに

現在の計算機のプログラムは、命令をメモリからプロセッサに読み込んで実行するソフトウェアの形をとっている。これに対し、そのプログラムをハードウェアで実現したものはソフトウェアでの実行よりもはるかに速い。これは専用ハードウェアによって汎用性を捨てることによる高速化や、速度が重要視される部分に重点的にハードウェア資源を割り当てること、またその部分の多重化、メモリから命令を持ってくることがなくなることでバスがそのぶん解放されることなどが主な理由である。

本研究ではハードウェアの量に制約があることを前提に、プログラムの一部をハードウェア化する方法を考える。ハードウェア、ソフトウェアのプログラムは同一の記述を用いて設計し、ハードウェア部分はFPGA(Field Programmable Gate Array)上に実現する。

2　背景

ある機能を高速化する際の方法の一つとして、その機能をハードウェア化するというものがある。しかし、ハードウェア化にはコストがかかり、数量が見込める場合や、コストがかかっても良い場合にしか作られない。また、専用ハードウェアの設計に時間がかかると、開発の開始時と完成時の技術格差により、出来上がった時はそれほど魅力のないものになってしまう恐れもある。これまではハードウェア化には多くの資金と時間が必要であり、かなりのリスクを伴うものであった。

現在ハードウェアの設計にハードウェア記述言語が使われるようになってきた。これは、以前の設計方法に比べ、設計時間を大幅に短縮することを可能にする。また、FPGAは内部の機能を自由に作ることが可能である。速度的にはASICよりは遅いが、それでもソフトウェアによる実行に比べれば十分速いものを作ることが可能である。FPGAには内部を再プログラム可能なものが多く製品化されている。これによって回路の設計→検証のサイクルが大幅に短縮できる。さらに、これを用いたハードウェアでは、構造によってはハードウェアに物理的な変更を一切加えずに機能を変更できる。このハードウェア記述言語とFPGAを用いたハードウェアの利用により、ハードウェアを従来ソフトウェアで行なっていた設計、デバッグの流れのように開発することが可能となる。

アプリケーションのハードウェア化による高速化の最初のステップはアプリケーションのどの部分をハードウェア化するかを決めることであった。これは、ソフトウェアに比べてハードウェアは一旦開発が開始されると設計の変更が容易ではなかったからである。特にハードウェアとソフトウェアの境界の変更は容易なものではない。しかし、FPGAを使った汎用のハードウェアを作成しておき、ハードウェア記述言語によって内部の設計をする方法ではかなりの部分までが改善される。

3　ハードウェアとソフトウェアの協調設計

プログラム作成時にその速度が問題となる場合がある。高速化が必要な場合、プログラム全体をハードウェア化できるのであれば問題がないが、普通はそうではない。前述したハードウェアにもそのハードウェアのサイズによってインプリメントする機能の大きさに制限がある。そのため、プログラムの一部をハードウェア化することが考えられる。ソフトウェア部分は従来の設計方法で設計し、ハードウェア部分はハードウェア記述言語で設計することも可能であるが、本研究ではC言語に若干の拡張を施しただけの言語一つだけで設計し、このプログラムからソフトウェアとハードウェアを合成する。この方法の利点は、ハードウェア記述言語よりもさらに高位の言語であるためハードウェアの設計がより容易になることと、ハードウェアとソフトウェアの境界を簡単に設定できること、ハードウェア資源の配分が簡単に行なえることである。ハードウェアとソフトウェアの境界の設定はCのpragma

†Saitoh Masanobu, Tada Yoshikaysu
††University ofElectro-Communications Graduate school of Information systems

で行なう。

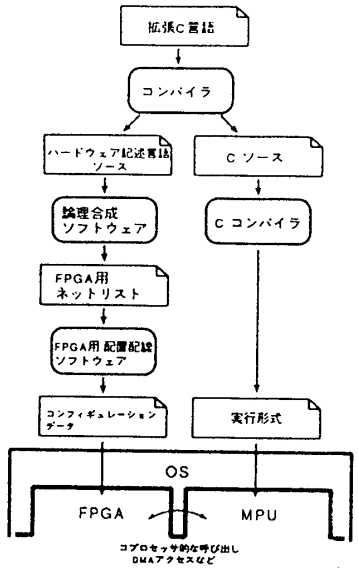

図 1 システムの開発フロー

プログラムを記述したファイルをコンパイラに通すと、ハードウェア化する部分がハードウェア記述言語で書かれたファイルと、ハードウェア化部分の記述がハードウェアとの通信に置き換わった C 言語で書かれたファイルの 2 種類のファイルが作成される。図 1 の中で、ハードウェア記述言語から FPGA 用配置配線ソフトウェアへのネットリストを生成する部分が FPGA 内の回路の動作速度に大きく影響する。特に、FPGA に載せる回路の遅延の予測、計算、その値の利用が問題となる。

本研究では使用するハードウェアとしては、複数の FPGA を相互に結合し、さらに FPGA からアクセスできるローカルなメモリを搭載したボードを想定している。このボードを計算機のバスに接続し、コプロセッサ的な使い方をする。

4 ハードウェア合成の戦略

ハードウェア化の基本戦略としては大きく分けて以下の二つの方法が考えられる。

- プロセッサ (CPU コア) のテンプレートを用い、実現する機能に適した構造に調整する方法
- できるだけデータの流れそのままに、実現する機能を直接マッピングする方法

本研究では後者の方法をとる。この方法はテンプレートを用いる方法に比べて高速であるが、実現する回路の規模が大きくなりやすい。プロセッサのテンプレートを用いる方法では ASIC 開発の場合はよく適しているが、FPGA をターゲットとした場合は速度の点で問題がある。汎用のものを作る土台である FPGA 上に専用回路を構築するのであって、汎用のものを作る土台である FPGA 上に汎用の計算をするためのマイクロプロセッサを作ることは 2 重に汎用性を持たせたことになり、ゲート規模の縮小には効果があるが高速化の効果は薄いと思われる。また、この場合はホストコンピュータのバスに別のマイクロプロセッサを接続することと同じことでもある。ただし FPGA 上にマイクロプロセッサを構築することは、プロセッサのアーキテクチャの学習、研究、開発を目的とするのであれば非常に有効なことであり、この理由からこの方法もインプリメントする予定がある。

レジスタ

FPGA のゲート規模にあわせ、適した個数のレジスタを必要なビット長で生成する。

再帰のループへの変更

再帰構造は極力 while/for ループなどの構造に変えて合成する。これは少ないローカルメモリをスタックとして使いたくないためである。

ホストとのインタフェース

実現する処理の特性によってパイプ的なインタフェースを採用したり、ローカルメモリを介して一気に読み込み／書きだしを行なうなど、適した方法を選択する。

5 おわりに

現在本方針に従ったコンパイラを作成中である。これと並行して FPGA を載せた Sbus ボードの開発も行なっている。このボードは Xilinx の LCA の XC4005 を 3 個載せたものになる予定である。基本システムが完成後は C 言語からハードウェアへの変換部分を中心に研究を進める予定である。

6 参考文献

[1] David E. Van Den Bout et al. “AnyBoard: An FPGA-Based Reconfigurable System” IEEE Design & Test of Computers. September 1992. page 21-30