FPGAによるCライブラリの部分的な高速化の試み*

3G－8

三浦 誠†　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

ソフトウェアを高速化させる手段として、プログラムをそれと同等の機能を持つハードウェアで 置き換えるという方法が考えられる。しかし、ハードウェアの開発には多くの設計時間がかかるため、出来上がったときには時代遅れのものになっている可能性がある。特にLSI化したものは回路の変更が出来ないため、設計に変更が生じた場合にはまた初めからやり直さなければならず、その分コストも増大する。

近年、内部の回路構成を自由に組み替えることのできるデバイスとして、FPGA(Field Programmable Gate Arrays) が注目されている。これは、ハードウェア組み替え時のランニングコストが低いので、ハードウェアの試作などを簡単に行なうことができる。また、これを利用したReconfigureable Computerの研究もなされている [1] [2]。

本研究では、FPGAを使用したハードウェアプラットホーム“無碍ボード”[3] を用いて、ソフトウェアの一部分をハードウェア化し、それによる高速化の度合いを検証した。

2 無碍ボード

2.1 無碍ボードの仕様

無碍ボードはSunワークステーションの汎用バスであるSBusに接続することのできるFPGAボードである。計算用FPGAにはXilinx社の5000ゲート相当のFPGAを3個使用している。各FPGAは独立して動作するが、全体を1つの回路として動作させることも可能である (図1参照)。FPGAのゲート数は、ピンコンパチブルな上位機種のものと取り換えることで増やすことができる。また、ローカルメモリとして、SRAMがある。これは、Sunワークステーションのアドレス空間からも、無碍ボード上のFPGA上の回路からも使用することができる。現在はこのSRAMを使ってデータのやり取りをしている。

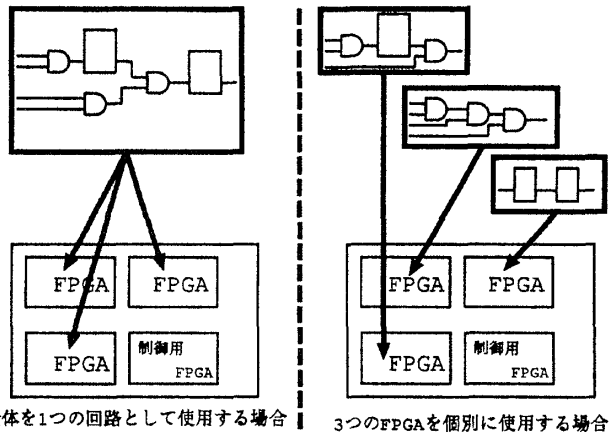

図 1: FPGAの利用方法

2.2 言語の選定

本システムのソフトウェア部分にはC言語を選定した。これは、本システムの動作環境がUNIX系のオペレーティングシステムであるためと、構造化言語のため部分的なハードウェア化がしやすいと考えたためである。ハードウェア化する部分については、NTTで開発されたSFL[4] を利用した。これは、設計やシミュレーションが容易であり、論理合成にかかる時間が短いため、ハードウェア部分の設計、検証が比較的短時間で繰り返し行えるという利点がある。また、ハードウェア部分の更なる高速化を望むのであれば、RTLレベルでの開発も考えられるが、今回は行わなかった。

2.3 ソフトウェア/ハードウェアの切り分け

ソフトウェアを高速化するには、プログラム全体をハードウェア化すれば良いが、ボード上のFPGAはプログラム全体をハードウェア化できるほどゲート数が多くない。複数のFPGAにまたがったハードウェア化も考えられるが、FPGA間を直結している信号線が少ないため、遅延が大きくなってしまう可能性がある。また、プログラム中で処理の一番重い部分を自動的に検出し、その部分をハードウェアに変換することは、プログラムのセマンティクスにまで踏み入らなければならず、かなり難しい。そこで本研究では、Cプログラムの関数の一部分をハードウェア化することによって、ソフトウェアを高速化する方法を考えた。関数単位でハードウェア化

*A partial speed up of C-library using “MUGE-board”
†Makoto Miura
‡Yoshikatsu Tada
§The University of Electro-Communications

したものをライブラリ化することにより、ユーザはハード部分を意識せずに使用することができるという利点を持つ。

3 プログラムの作成

プログラム作成時の流れは、図2のようになる。

- まず、ハードウェア化する関数をSFLで作成する。これを、NTTのPARTHENONシステムを使って論理合成し、XilinxのXACTツールを使ってコンフィギュレーションデータに変換する。
- ハードウェアを呼び出す部分はCコンパイラでオブジェクトコード化し、ライブラリ化しておく。
- プログラムは、通常のCコンパイラを使い、リンク時に前述のライブラリをリンクする。

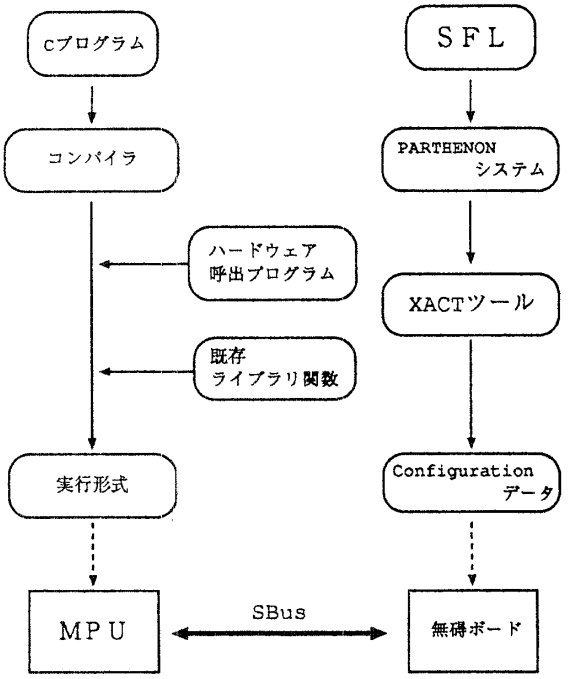

図 2: 無碍ボードを利用したソフトウェア作成の流れ

4 実験

実験として、文字列操作の関数をハードウェア化した。論理合成、コンフィギュレーションデータへの変換はSun SparcStation 20(メモリ 80MB)で行った。動作検証はSun SparcStation LX(メモリ 32MB)で行っている。

4.1 strchrの検証

strchrについてソフトウェア単体と無碍ボードを使用した場合の比較を行なった。実験は、ランダムな256byteの文字列から1文字を検索した。実行結果を表1に示す。なお、括弧ないはデータ転送時間である。

表 1: ctrchrの実行結果の比較

検索回数	ソフトウェアのみ	無碍ボード使用時
500回	約2秒	約150秒 (約70秒)
1000回	約5秒	約420秒 (約150秒)

ハードウェア化したものの実行時間はデータ転送時間を差し引いてもかなり遅い。これは、文字列が小さいので、CPUがキャッシュを使って動作するためである。キャッシュには入り切らない程長い文字列を使用すると結果が変わってくると思われるが、現在のところ、FPGA内のカウンタを大きくすると、動作速度が遅くなってしまうので、あまり長い文字列には対応できない。そのため、動作速度が遅くならないような回路を作成中である。

5 おわりに

現在の無碍ボードにはDMAの機構が無いため、ワークステーション側のメモリ操作はFPGAが直接行なわなければならない。しかし、FPGAの動作が遅いため、FPGA側からSBusの信号を操作するといったことができず、ワークステーション側のメモリを操作することができない。よって、データ転送のオーバーヘッドがネックとなってしまう。これは今後の課題である。

参考文献

[1] Peter M. Athan, and Harvey F. Silverman, "Processor Reconfiguration Through Instruction-Set Metamorphosis," IEEE computer, pp. 11-20, Mar 1993.

[2] M.Wazlowskim L. Agawal, T. Lee, A. Smith, E. Lam, P. Athanas, H. Silverman, and S. Ghosh, "PRISM-II Compiler and Architecture," Proceedings of the IEEE workshop on FCCM, pp. 9-16, 1993.

[3] 齊藤正伸, 多田好克, "ソフトウェアの部分的なハードウェア化による高速化技法の研究", 情報処理学会第36回冬のプログラミング・シンポジウム報告集, pp. 127-134, Jan., 1995.

[4] 中村 行宏, 小栗 清, 野村 亮, "RTL動作記述言語SFL", 電子情報通信学会論文誌, Vol. J72-A, N0.10, pp. 1579-1593, Oct., 1989.