3K-09

Linuxの動くFPGAシステムを3倍速くする試み

小川 愛理† 吉瀬 謙二‡

†東京工業大学 情報工学科 ‡東京工業大学 大学院情報理工学研究科

1 はじめに

FPGAの性能向上に伴い、これまでASICが採用されてきた分野においてもFPGAが用いられるようになっている。今後、OSの動くコンピュータシステムにおいてもFPGAが用いられるようになるであろうと考え、我々はLinuxの動くFPGAシステムを提案している。

提案したFPGAシステムでは、計算機で最も一般的なISAであるx86をサポートしている。提案したFPGAシステムの開発では、x86をサポートし、かつOSが動く既存のFPGAシステムを改良することで、開発期間の短縮を目指している。しかし、既存のシステムは近年の計算機と比較して非常に低速であり、実用的なシステムとは言えない。そのため我々は、回路の簡略化とキャッシュの利用によりシステムの高速化を図った。本稿ではそのアプローチについて述べる。

2 ao486 SoC

FPGAを用いた計算機システムを開発するにあたって、我々は現在の計算機との互換性を保つため、コアのアーキテクチャにx86アーキテクチャを採用した。

x86アーキテクチャの計算機の開発では、ao486[1]と呼ばれるVerilog HDLで記述されたオープンソースのプロジェクトを基盤として使用する。図1にao486 SoCの概略図を示す。ao486はIntel 80486SXの全ての機能を実装したソフトコアプロセッサであり、ao486 SoCにはこのプロセッサに加えてVGAコントローラー、PS2コントローラーなどの様々なデバイスコントローラーが含まれる。ao486 SoCはTerasic社のAltera DE2-115 FPGAボード上で動作し、開発者のドキュメントによるとLinuxカーネル3.13とWindows95を起動することができる。

現在はこのシステムに一部改良を加え、Altera DE2-115 FPGAボードを用いてLinux (Tiny Core 5.3)とFreeDOS 1.1が動作するのを確認している。我々は、このシステムに第3章で述べる高速化手法を加えることで速度向上を達成し、実用的な計算機システムの実現を目指す。

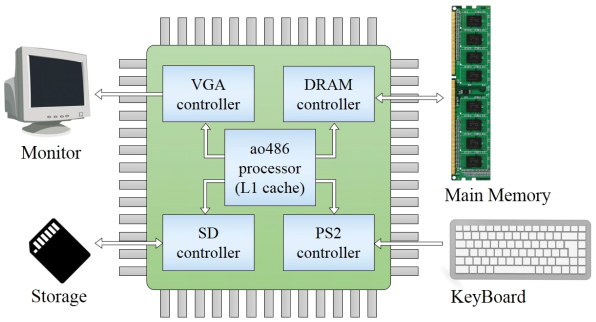

図 1: ao486 SoC の概略図

3 高速化手法

本章では、ao486 SoCの具体的な高速化の手法について述べる。高速化には以下の2つの手法を用いた。

- 回路の簡略化による動作周波数の改善
- L2キャッシュの実装によるメモリアクセス時間の改善

3.1 回路の簡略化

ao486 SoCは、実装の簡単化のためにAltera社のソフトコアプロセッサであるNios Ⅱ[2]を使用している。Nios Ⅱは計算機が起動する際のBIOSの転送等を行っており、多くのデバイスコントローラーと接続している。

我々は、このNios Ⅱの代わりにBIOSの転送等を行うモジュールをVerilog HDLで記述し、さらに不必要なモジュールの削除などを行うことで回路を簡略化した。この回路構成の見直しにより、クリティカルパスの伝播遅延時間が軽減され、結果的に30MHzで動作していたao486 SoCを75MHzで動作させることに成功した。

3.2 L2キャッシュの実装

ao486 SoCには16KBのL1命令キャッシュ及びデータキャッシュが実装されている (どちらも4wayセットアソシアティブ方式) が、実行させるプログラムによってはL1キャッシュを有効化する命令がサポートされて

A challenge to the 3x speed up of FPGA systems running Linux OS

Eri OGAWA†and Kenji KISE‡
†Department of Computer Science
Tokyo Institute of Technology
‡Graduate School of Information Science and Engineering
Tokyo Institute of Technology

図 2: Altera DE2-115 FPGA ボード上で動作する DOOM のスクリーンショット

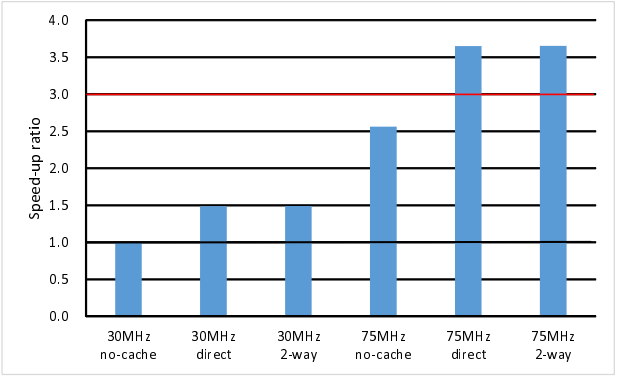

図 3: 30MHz 動作・キャッシュなしのバージョンを 1 として、各高速化手法を加えた際の速度向上率

いないことから、プログラムの実行時間におけるメモリアクセス時間の占める割合が大きくなってしまうことがある。そこで、我々は ao486 SoC に L2 キャッシュを実装することで、メモリアクセス時間の改善を図った。L2 キャッシュは命令キャッシュとデータキャッシュの両方を兼ねる。

実装したキャッシュは、ダイレクトマップ方式と 2-way セットアソシアティブ方式の 2 種類であり、それぞれキャッシュサイズは 64KB、ラインサイズは 1 ワードである。どちらのキャッシュもライトバック方式を採用し、書き込みと読み出し両方の高速化を図る。

2-way セットアソシアティブ方式のデータの置き換えアルゴリズムには LRU を採用した。LRU の実現には、どちらのセットが最近使用されたかを示す 1bit のカウンタを各エントリ毎に用意する。キャッシュデータに置き換えの必要が生じた時には、基本的に LRU の規則に従い置き換えるセットを選択する。しかし、キャッシュデータがメインメモリのデータから変更されている場合には、キャッシュミスの際のデータの追い出しに伴いメインメモリへライトバックする必要があるため、キャッシュのミスペナルティが大きくなってしまう。そのため、この場合にはメインメモリとの整合性が取れているキャッシュラインを優先して置き換える。

4 評価と考察

第 3 章の高速化手法を適用した計算機システムにおいて、その速度向上率を計測した。計測には、FreeDOS 1.1 上で動作する idsoftware 社のコンピュータゲーム DOOM[3] を用い、実行時間の比較を行った。図 2 に計測の様子を示す。計測方法は、Altera DE2-115 FPGA ボードを用い、各高速化手法を実装した場合とベースライン (30MHz 動作・L2 キャッシュなし) の両方の計算機システム上で DOOM を起動し、起動からデモンストレーションが終わるまでの実行サイクル数をホストコンピュータ上で計測した。

図 3 に、ベースラインの実行時間を 1 として、各高速化手法を施した際の速度向上比率を示す。なお、この計測では L1 キャッシュが有効化されていないため、計測結果は L1 キャッシュの影響を受けていない。動作周波数を 30MHz から 75MHz にすると 2.5 倍の速度向上が得られた。また、L2 キャッシュの追加により 1.4 倍の速度向上が得られた。結果として 75MHz 動作の 2-way セットアソシアティブ方式の L2 キャッシュが付いたバージョンは、ベースラインと比べて 3.7 倍の速度向上を達成した。これにより、当初に比べ DOOM は格段に俊敏な動作が可能になった。

L2 キャッシュの追加により、メモリアクセス時間が改善され実行時間の削減に繋がったが、ダイレクトマップ方式と 2-way セットアソシアティブ方式の間で速度向上に大きな差は見られなかった。これは、今回測定に用いたアプリケーションの使用する命令数が少なく、ダイレクトマップ方式でもキャッシュミスが起きにくかったためと考えられる。

5 まとめ

我々は、OS の動く計算機システムに回路の簡略化と L2 キャッシュを加えることで最大 3.7 倍の速度向上を達成した。これにより、ベースラインと比較して大幅に性能が向上した。今回の実装ではダイレクトマップ方式と 2-way セットアソシアティブ方式に大きな性能差が得られなかったが、キャッシュの最適化により更なる性能向上が期待できる。

今後の課題としては、この計算機システムを用いて FPGA で構成された先進的な計算機システムを提案、開発していくことがあげられる。

参考文献

[1] ao486, https://github.com/alfikpl/ao486

[2] Nios Ⅱ プロセッサリファレンスハンドブック 日本アルテラ, http://www.altera.co.jp/literature/hb/nios2/n2cpu_nii5v1_02_j.pdf

[3] idsoftware, http://www.idsoftware.com/en-gb/