ハード/ソフト・コラーニングシステムにおける アーキテクチャ選択可能なプロセッサシミュレータの設計

大八木睦† 池田修久† 山崎勝弘† 小柳滋†

†立命館大学大学院 理工学研究科

1. はじめに

現在の半導体業界はシステム LSI 設計が主流である。システム LSI はプロセッサをシステムに組み込んだことが大きな特徴で、これはつまりソフトウェアを組み込むことができるということである。この点から、システム LSI におけるプロセッサとソフトウェアの関係の理解は重要であり、大学教育においてハード、ソフト両方を理解した技術者の育成が急務である。そこで、大学においてプロセッサをテーマにハード・ソフトを協調学習するハード/ソフト・コラーニングシステムの構築を行う。本稿ではシステムの構成要素であるアーキテクチャ選択可能なプロセッサシミュレータについて述べる。

2. ハード/ソフト･コラーニングシステム

2.1 システム概要

ハード/ソフト・コラーニングシステムは、プロセッサアーキテクチャを意識した上でのプログラミング学習を行うハードとソフトの協調学習システムである。ハードウェア面ではシミュレータを用いて複数のプロセッサアーキテクチャを理解し、その知識を用いて HDL によるプロセッサ設計を行う。また、設計したプロセッサを FPGA ボードコンピュータに搭載し、その周辺回路と共に実機検証を行う[3]。ソフトウェアはシミュレータを用いた C 言語やアセンブリ言語によるプログラミング、プロセッサアーキテクチャ毎の最適化コンパイラ設計、及びプログラムの評価などを通して学習する。

2.2 システム構成要素

次に、コラーニングシステムを構成する主な要素について説明する。

♦ MONI プロセッサ

独自に定義した 16 ビット教育用プロセッサ。単一サイクル、マルチサイクル、パイプライン、スーパースカラの 4 種類のアーキテクチャがある。学習者は各プロセッサのコアとなる部分を HDL により設計し、ハードウェア設計の学習を行う。

♦ アーキテクチャ選択可能なプロセッサシミュレータ

MONI プロセッサを対象とした命令セットシミュレータである。複数のアーキテクチャを対象として、種々の実行モード、データパスやレジスタの表示、ハザード通知、プログラムの評価などの機能がある。

♦ 最適化コンパイラ

4 種類のプロセッサアーキテクチャにおける最適化コンパイラを設計する。

Design of a Variable Processor Simulator in a Hardware/Software Co-Learning System

Mutsumi Oyagi†, Nobuhisa Ikeda†, Katsuhiro Yamazaki† ,and Sigeru Oyanagi†

†Department of Computer Science, Faculty of Science and Engineering, Ritsumeikan University

3. プロセッサシミュレータの設計

3.1 要求仕様

設計するシミュレータは 16 ビット MONI 命令セットを対象とした命令セットシミュレータである。単一サイクル、マルチサイクル、パイプライン、スーパースカラの 4 種類のアーキテクチャを任意に選択することができ、各アーキテクチャにおいて複数の実行モードを用意する。それぞれの実行モードにおいてメモリ、プログラムカウンタ、レジスタの変化の様子を逐一表示し、使用しているユニットが目で見て確認できるような GUI を持つ。また、プログラム実行後には、メモリアクセス数、分岐回数、ストール回数などを表示し、プログラムの評価に用いる。学習者はプロセッサアーキテクチャの理解や、アーキテクチャに最適なプログラミングの学習を行う。

3.2 シミュレータの画面構成

設計したシミュレータの画面構成を図 1 に示す。

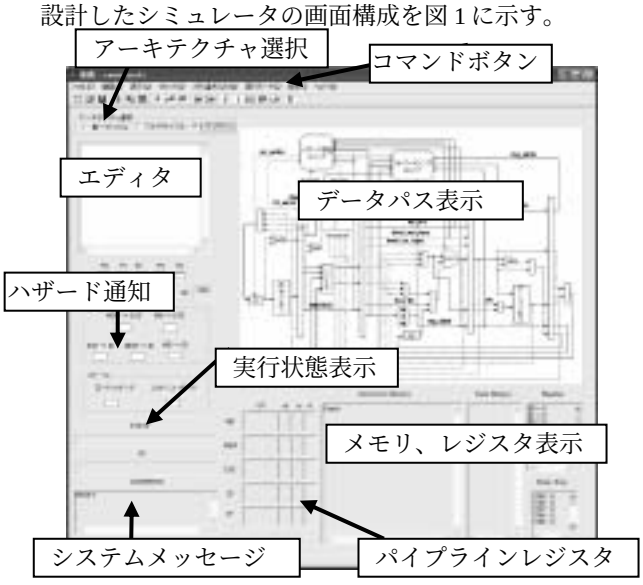

図 1：シミュレータの画面構成

作成したアセンブリプログラムをエディタに開き、メモリに書き込む。アセンブルエラーがなければアーキテクチャ選択後、実行を開始する。実行終了後も、データやアーキテクチャを変更しながらデバッグを繰り返し行える。そして、最後にシミュレーションデータを表示し、プログラムの評価を行う。

3.3 シミュレータの機能

シミュレータには 1 命令実行、1 クロック実行、プログラム実行、ブレーク実行、設定行数実行、及びストール発生まで実行の 6 種類が選択できる。

- 単一サイクル： 1 命令実行と 1 クロック実行は同じ動作を行う。現在実行されている命令を表示し、各命令で使用されるユニットを強調表示する。
- マルチサイクル： 1 クロック実行の際、フェーズランプを用いて、現在命令のどのフェーズを実行しているかを示す。同時にデータパスの強調表示を行う。
- パイプライン： フォワーディングやハザードが起きている状況をランプとデータパスで視覚化する。また、ストール発生まで実行するモードもパイプラインでは有効である。
- 共通： プログラムの最後まで実行するプログラム実行、デバッグ用にブレーク実行と設定行数実行がある。

ここで、パイプライン選択時のシミュレータの実行画面を図 2 に示す。ランプ点灯やバブル挿入により、フォワーディングやストールが発生している様子が分かる。

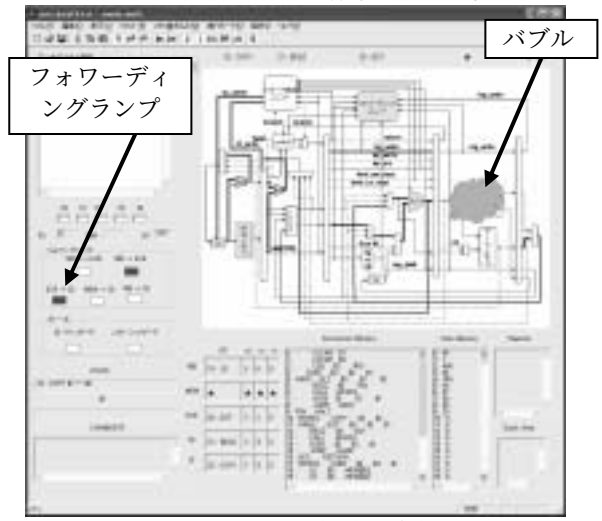

図 2：パイプライン選択時の実行画面

4. C++によるプロセッサシミュレータの試作

4.1 開発環境

開発ツールとして VisualC++6.0 を用いた。SDI（Single Document Interface）形式を採用し、単一の窓構成である。主なクラスには以下のものがある。

- Document：ファイル編集などユーザデータの保持
- View：ウィンドウの表現方法を定義
- MainFrame：SDI 形式のフレームを制御
- Dialog：独自に定義したダイアログを制御

4.2 モジュール構成

モジュールはリセット、メモリ書き込み、実行、状態表示の４つに大別できる。モジュール構成を図 3 に示す。

- Imwrite()：命令メモリ書き込み

命令メモリに書き込む関数である。エディタから命令を読み出し、２次元配列 inst_mem[][]に格納する。同時にエラーチェックを行う。

- Inst()：１命令実行

命令実行と同時に各レジスタの内容を表示する。

- Clk()：１クロック実行

マルチサイクルが選択されている場合は、命令フェッチ→命令デコード→実行→メモリアクセス→ライトバックのステートに従い、実行する。

- Pipe()：パイプライン実行

命令を実行しながらパイプラインステージに見立てた配列 pipe[][]に格納。pipe[][]の依存関係を調べ、フォワーディングやハザードを検出する。

- OnDraw()：画像表示

Inst()、Clk()、Pipe()から値を受け取り、それぞれの状態に合った画像を表示する。

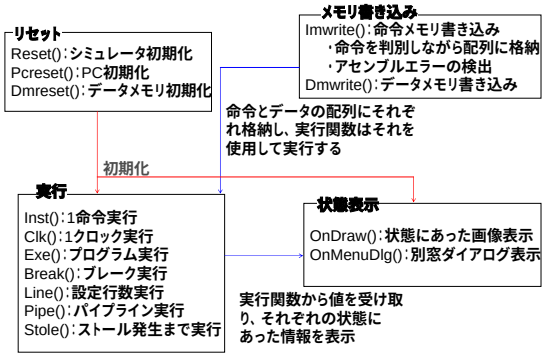

図 3：モジュール構成

5. シミュレータのテストと考察

テストプログラムとしてバブルソートとシェルソートをパイプラインモードでデータ数を 50 として実行した。プログラム実行後に得られたデータは、バブルソートは総命令数：16828、総クロック数：25121、CPI：1.5 である。シェルソートは総命令数：3975、総クロック数：4728、CPI：1.2 である。表 1 のように実行したプログラムにおいてストール発生回数とその割合が表示されるため、アーキテクチャの学習だけでなくパフォーマンスデバッガとしても利用できる。例えば、バブルソードでは RETURN 命令によるストールの占める割合が５割近くあるため、サブルーチンの数を減らし、１つのモジュールを大きくするといったプログラムの変更が望ましい。

表 1：ストール発生回数

	合計	LD	Branch	CALL	RETURN	JUMP
バブル	8289	1225	694	1274	3822	1274
(%)		(14.8)	(8.4)	(15.4)	(46.1)	(15.4)
シェル	749	283	151	3	9	303
(%)		(37.8)	(20.2)	(0.4)	(1.2)	(40.5)

6. おわりに

ハード/ソフト・コラーニングシステムとその構成要素であるアーキテクチャ選択可能なプロセッサシミュレータについて述べた。今後、スーパースカラシミュレータを実現し、また最適化コンパイラとの融合を図ることで、システムの完成を目指したい。

参考文献

[1]John L. Hennessy, David A. Patterson 著, 成田光彰訳: コンピュータの構成と設計 (上) (下), 日経 BP 社, 1999.

[2]林晴比古著:新 VisualC++6.0 入門シニア編, ソフトバンクパブリッシング, 2001.

[3]池田 修久 他, ハード/ソフト・コラーニングシステムにおける FPGA ボードコンピュータの設計, 情報処理学会 第 66 回全国大会論文集, 2004