LC-001

マイクロプロセッサの設計と検証に基づいたハード/ソフト・コラーニングシステムの拡張

難波 翔一朗† 中村 浩一郎† 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

近年、システムLSIの設計需要が高まる中、システム全体の制御を担う汎用プロセッサには、高速で低コスト、低消費電力であることが求められている。これらの機能を実現するためには、プロセッサの命令セットと内部のマイクロアーキテクチャの設計技術を深く理解することが必要であり、我々の研究室ではそれらを体系的に学習できるハード/ソフト・コラーニングシステムを開発している[1]-[5]。我々は実際にコラーニングシステムを利用して単一サイクルのプロセッサを設計、動作検証を行った。本論文ではプロセッサの設計を通じて浮かび上がったシステムの問題点を考察し、それらを考慮したコラーニングシステムの拡張案を示す。

2. ハード/ソフト・コラーニングシステム

ハード/ソフト・コラーニングシステムとは、学習者がプロセッサアーキテクチャをソフトウェアとハードウェアの両側面から設計することによって、両者の理解を協調的に進める学習システムである[1][2]。図1にハード/ソフト・コラーニングシステムの学習体系を示す。学習内容には、プロセッサアーキテクチャを意識したアセンブリプログラミング(ソフトウェア学習)と、定義された命令セットに対応するプロセッサ設計(ハードウェア学習)のフローがある。本システムでは、現在までに基本命令セットMONIを中心とする学習環境を提供し、ハードとソフトの協調学習体系を実現している。

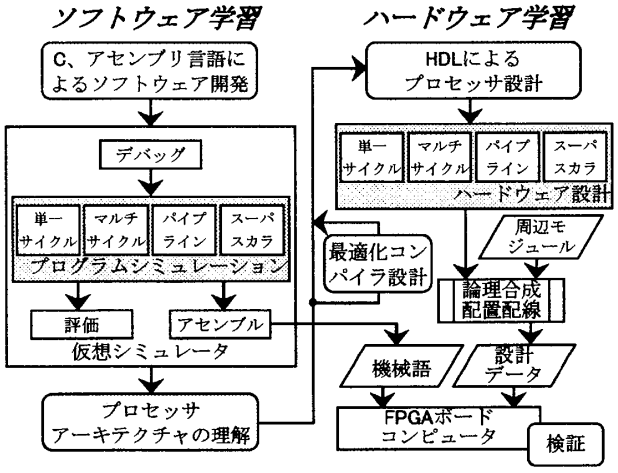

図 1:ハード/ソフト・コラーニングシステムの学習フロー

Improvement of a Hardware/Software Co-learning System based on a Microprocessor Design and Verification, Shoichiro Namba, Koichiro Nakamura, Katsuhiro Yamazaki, and Shigeru Oyanagi.
† Graduate School of Science and Engineering, Ritsumeikan University.

3. 単一サイクルマイクロプロセッサの設計

3.1 設計思想

本研究では、16bit単一サイクルのRISCプロセッサを設計した。図2にその命令形式を示す。Register、Immediate、Transfer、Jumpの4つの命令形式と25個の命令を独自に定義し、プロセッサのマイクロアーキテクチャは、高速化のためできるだけ単純な構成とした。プロセッサの特徴として①3オペランド方式、②演算は汎用レジスタ間のみに限定、③状態レジスタを用いた分岐、④三つのデータ転送アドレッシング、などがある。

Register	op (5)	fn (2)	rs (3)	rt (3)	rd (3)
Immediate	op (5)	imm (5)		rt (3)	rd (3)
Transfer	op (5)	imm (8)			rd (3)
Jump	op (5)	imm (11)			

図 2:命令形式

3.2 プロセッサの構成と設計

図3に設計したMPUのデータパスを示す。MPUは、命令デコーダ(ID: Instruction Decoder)、制御装置(CU: Control Unit)、レジスタファイル(RF: Register File)、算術論理演算ユニット(ALU: Arithmetic and Logical Unit)、状態レジスタ(SR: Status Register)、比較器(CMP: Comparator)、プログラムカウンタ(PC: Program Counter)、マルチプレクサ(MUX: Multiplexer)、及び3ステートバッファのモジュールから構成される。

MPUの開発環境としてFoundation ISE 6.3i、論理合成ツールにXST、シミュレータにModel Sim XE、設計言語としてVerilog HDLを使用し設計を行った。表1にはISEのTiming Analyzerで計算したMPUの回路規模と性能評価を示す。実装命令数が少ないためマ

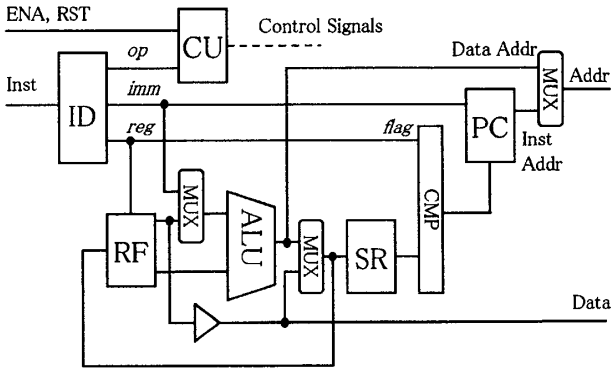

図 3:MPU のデータパス

表1:MPUの回路規模と性能評価

モジュール	HDL記述量(行)	スライス数	LUT数	ゲート数	最大遅延(ns)	最大動作周波数(MHz)
ID	94	18	35	267	13.8	72.6
CU	72	19	38	255	13.5	73.9
RF	18	32	64	4096	11.0	91.0
ALU	30	166	289	2172	24.3	41.2
SR	17	4	5	46	10.7	93.8
CMP	19	2	3	18	11.5	87.0
PC	30	14	19	248	8.3	120.6
MUX	11	2	16	18	11.9	84.4
MPU	120	461	799	6822	43.5	23.0

イクロアーキテクチャが単純にでき、MPUのゲート数は6822システムゲートと小さく抑えられた。1命令を実行するために必要な遅延は、ディスプレースメント付きレジスタ間接のロード命令が最大であり、命令をフェッチしアドレスを計算してメモリからRFにデータを転送するまでの合計時間をMPU全体の最大遅延とする。この結果、最大動作周波数は23MHzであった。MONIのシングルサイクルプロセッサの最大動作周波数、37MHzと比較するとやや下回る結果となったが、この要因として、分岐判定に状態レジスタの値と即値のフラグを比較するための機構が含まれているためである。

4. FPGA上での動作検証

設計したMPUは、2通りの方法（ハード/ソフト・コラーニングシステムとMPU検証システム）でFPGA上に実装し動作検証を行った。実装環境として、Celoxica社のRC100ボード、FPGAチップにXilinx社のSpartan II FPGA(20万システムゲート)を用いた。

4.1 ハード/ソフト・コラーニングシステムを用いた実装

ハード/ソフト・コラーニングシステムでは、基本プロセッサMONIと、MONIをFPGAボード上で動作させるための周辺モジュール群が用意されている。今回作成した単一サイクルMPUをMONIに代えて周辺モジュールと共にFPGAボード上に搭載し、実機検証を行った。作成したMPUとMONI搭載用の周辺モジュールとの結合には、データ/アドレスバスの調整や動作開始のタイミングを合わせる必要があったが、MPUの上位モジュールにおいてインタフェースを調節することで、FPGA上に実装したシステムで動作を確認することができた。MPUの検証にはNまでの和、バブルソート、最大値検索をテストプログラムとして用いた。

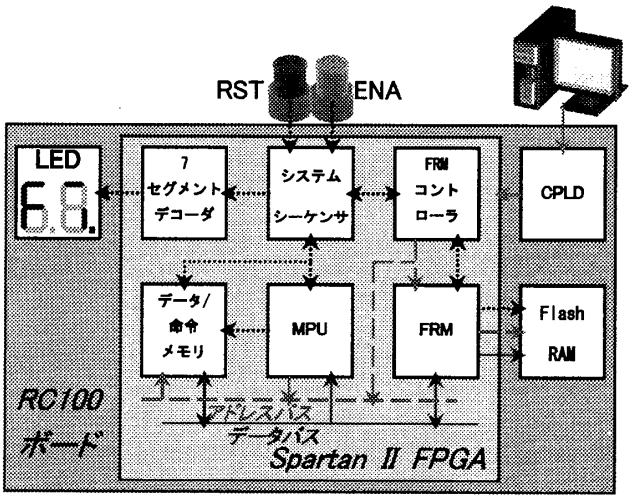

図4:MPU検証システム構成

4.2 MPU検証システムを用いた実装

独自に作成したMPU検証システムの構成を図4に示す。FPGAチップの内部には、設計したMPU、システムの起動から終了までの動作手順を制御するシステムシーケンサ、FlashRAMとメモリ間のデータ転送を行うFRM(Flash RAM Module)とそのコントローラ、Xilinx社から提供されているIPから生成したデータ/命令メモリなどを搭載している。FPGA外部にはMPUで実行する命令/データを格納するFlashRAMと、FPGAのコンフィギュレーションデータを一時格納しておくCPLDが搭載されている。

4.3 MPU検証システムの評価

表2と表3に検証に用いたMPU検証システムの回路規模と性能評価を、表4にテストプログラムの実行結果を示す。なお、IPから生成したメモリはブラックボックス化されているため、検証データはシステム全体の項目に含まれるものとして評価を行った。

表2のゲート数に注目すると、システムの周辺モジュールの回路規模は非常に低いが、メモリを含むシステム全体では15万ゲートとある程度大きい結果となった。これは、システム内に実装した命令/データメモリがFPGA内のゲートの多くを消費しているためであると考えられる。また、Spartan II FPGAの使用率は全スライス数の27%であった。

表2:回路規模

モジュール	HDL記述量(行)	スライス数	LUT数	ゲート数	FPGA使用率
システムシーケンサ	122	19	36	308	1%
FRMコントローラ	249	52	69	1088	2%
7セグメントデコーダ	32	4	7	42	0%
クロックカウンタ	12	1	16	448	0%
MPU	120	461	799	6822	19%
システム全体	231	657	1052	155911	27%

表3:性能評価

モジュール	最大動作周波数(MHz)	最大遅延(ns)
システムシーケンサ	145.4	6.9
FRMコントローラ	77.8	12.8
7セグメントデコーダ	103.0	9.7
クロックカウンタ	183.0	5.5
MPU	23.0	43.5

表4:実行結果

プログラム	静的命令数	動的命令数	レジスタ使用数	実行内容
バブルソート	18	599	7	10データの並べ替え
選択ソート	22	415	7	
挿入ソート	17	149	8	
シェルソート	26	226	8	
最大値検索	13	57	5	10データの最大値
Nまでの和	7	34	2	10までの和
最大公約数	15	34	3	288と54の最大公約数
乗算(自然数)	9	211	3	12×15
乗算(整数)	13	212	3	
累乗算	23	162	5	3の9乗
ランレングス符号化	140	1958	8	5バイトのデータの圧縮
二値化	34	16936	7	50×34のpgm画像

表3に示す性能評価では、周辺モジュールにMPUの23MHzを下回る最高動作周波数はないことから、MPU検証システムがシステムクロック周波数を低下させることは回避できている。今後は、MPUの最適化やパイプライン化を行い、動作速度を上げることで、システム全体の最大動作周波数を向上させることが課題である。

表4の静的命令数では、バブルソート、Nまでの和のプログラムに着目すると、状態レジスタを用いることで分岐命令を単純化したため非常に小さく抑えられたことがわかる。しかし、ランレングス符号化では140行と大きな値となった。これは、命令セットに関数呼び出しを実現する命令が存在しないため、同じ処理をプログラム内に複数記述したためである。動的命令数では、二値化のプログラムが非常に大きくなった。画素値のロード、閾値との比較、ストアの三つの処理を各画素について繰り返した結果である。また、二値化のプログラムでは、FPGA内部にメモリを実装したためメモリ容量が2キロバイトしかなく、非常に小さな画像での検証となった。今後はFPGA外部に搭載されているメモリモジュールを利用する、もしくはより容量の大きいFPGAを搭載するボードを対象に実装することが課題となる。

5. ハード/ソフト・コラーニングシステムの拡張

5.1 現システムの問題点と拡張の検討

FPGAでの動作検証を通じて、以下のような改善点が浮かび上がった。①FPGAボード上でのデバッグが困難。LEDでしか内部レジスタの確認ができない。②スイッチの数が限られ、外部からの操作性が悪い。③論理合成に時間がかかる。デバッグを行うにはシステム全体の論理合成が必要であり、ボード上での動作確認に時間がかかる。④MONIとは異なる独自のMPUを搭載するには専用のアセンブラやシミュレータなどの開発環境が必要である。このような点を踏まえ、現在のコラーニングシステムを発展させ、学習のためのプロセッサ設計が円滑に進められるよう、また、ハードウェアとソフトウェアの関係をより深く学習できるようシステムの拡張を検討した。具体的には基本命令セットMONIを拡張できる機構を追加し、学習者にはどのような命令を追加すればよいかを考慮してもらう。これにより、命令セットとプロセッサアーキテクチャの相互の関係をより深く学習できる。また、ターゲットデバイスにホスト端末との通信機能が充実しているCeloxica社のRC200ボードを、FPGAチップにVirtex II FPGAを用いることで、ホスト端末からFPGA上のコンピュータシステムを操作できる機能を盛り込み学習者にプロセッサの設計に専念できる環境を整える。

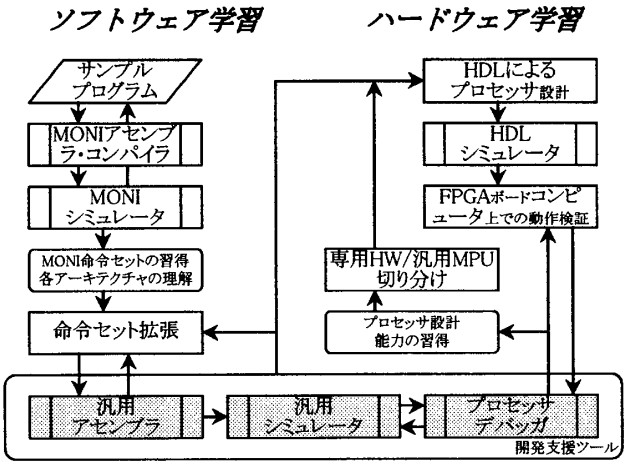

図5:ハード/ソフト・コラーニングシステムの拡張

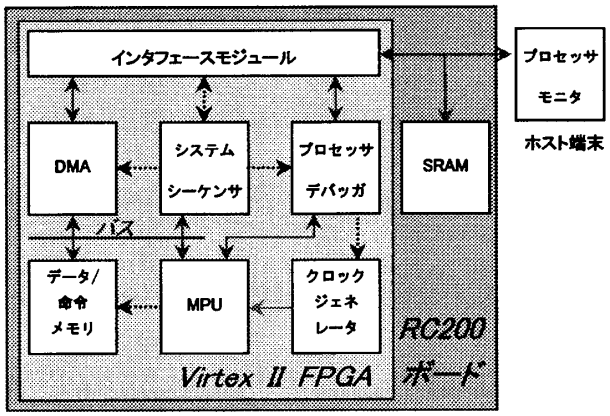

図6:プロセッサデバッガを含むコンピュータシステムの構成

5.2 拡張システムの機能

図5に拡張後のハード/ソフト・コラーニングシステムの学習フローを示す。拡張したハード/ソフト・コラーニングシステムでは、FPGA上でプロセッサのデバッグを行うプロセッサデバッガとホスト端末上で利用する汎用アセンブラ・シミュレータを盛り込み、それらを密接に連携させてプロセッサ設計を行う。

(1) プロセッサデバッガ

図6にプロセッサデバッガを組み込んだコンピュータシステムの

構成を示す。プロセッサデバッガは、ホスト端末上で動作しているプロセッサモニタと通信し、プロセッサ内部のレジスタ値の観測や変更、また、プログラムのステップ実行などを実現するハードウェアモジュールである。プロセッサデバッガは、クロックジェネレータを介してプロセッサに供給されるクロックを制御することで、1クロック/1フェーズ/1命令/1プログラム毎など様々なステップ実行モードでプロセッサの検証を可能にする。プログラムの実行前に、あらかじめホスト端末からデバッグのためのブレイクポイントが指定されていれば、プロセッサのPCがブレイクポイントにたどり着いた時点でプロセッサのクロックを停止させ、レジスタ値の操作を可能する。ブレイクポイント以前の実行命令に対しては、プロセッサデバッガがプロセッサの動作中に内部のすべてのレジスタと実行命令を外部のSRAMに履歴を保存することで、実行終了後にホスト端末側の汎用シミュレータで再現し、デバッグを行う。ホスト端末とプロセッサとの通信では、FPGA内に組み込んだインタフェースモジュールがデバッグプログラムのデコーダとしてシステムシーケンサ、DMA、プロセッサデバッガに適切に指示を与える。

このように、プロセッサデバッガとホスト端末上の命令シミュレータとを協調して動作させることで、シミュレーション結果と実機での動作結果とを比較しながらデバッグを進められる。実機とシミュレーションの実行結果を比較することは、プログラムによるアルゴリズムのバグと、ハードウェアのアーキテクチャのバグとの分離が図られる利点がある。また、プロセッサのデバッグ操作にホスト端末上の開発支援ツールを用いることで、今までの時間がかかっていたFPGA上でのデバッグをスムーズに進められ、学習者の負担を減らすことが可能である。以上のような機能を実現するためデバッグプログラムを実現するコマンドには、レジスタ値の観測/変更、メモリ内のデータ/命令の変更、ブレイクポイントの設定、次のステップまでの実行などのコマンドを提供する。

(2) 汎用アセンブラ・汎用シミュレータ

ニーモニックと機械語の対応表、及び各命令を専用の動作記述言語で表現した定義ファイルを与えることで、学習者が設計した命令セットのアセンブラ/シミュレータとして機能する。汎用アセンブラ・シミュレータの最大の特徴は、命令定義ファイルを与えることで、異なる命令セットアーキテクチャで組まれたアセンブリプログラムを汎用的にアセンブル、もしくはシミュレーションできる点にある。この機能を実現するために、専用の動作記述言語を用い、1つの命令を基本的な加減算やビットシフト、ロード/ストア命令などの組み合わせで表現する。すべての命令の振る舞いは定義ファイルにまとめられ、シミュレータではこれを参照しながらインタプリタでアセンブリソースを解釈しシミュレーションを行う。これにより、ハード/ソフト・コラーニングシステム上で、ユーザが独自にMPUを設計し検証することが可能となり、より発展的な協調学習を行うことができる。

5.3 学習フロー

(1) ソフトウェア学習において、学習者はMONIシミュレータを用いてサンプルプログラムのコンパイル、シミュレーションを行い、MONIプロセッサの構造と動作を理解する。サンプルプログラムを改変することで実行性能にどのような影響があるかを確認する。

(2) (1)で学んだMONIプロセッサを実際にHDLにより設計する。この段階ではハードウェア特有の回路規模や遅延時間などの制約を考慮した設計空間探索能力を養う。

(3) 汎用アセンブラ・シミュレータを用いて拡張の有用性を確認しながらMONI命令セットの拡張を行う。この段階ではソフトウェア側における実行効率の良い命令セットアーキテクチャの設計空間探索能力を養う。

(4) 設計した命令セットを実行するプロセッサを設計する。(3)で拡張したMONI命令セットをハードウェア面から検討し、実現する。

(5) 作成したMPUの動作検証をFPGA上で行う。MPUと共にFPGA内に実装したデバッグモジュールとホスト端末側のプロセッサデバッガの通信によって回路のデバッグを行いながら進める。

6. おわりに

本研究では、単一サイクルプロセッサを設計し、ハード/ソフト・コラーニングシステム上と、独自のコンピュータシステム上で動作確認を行った。また、プロセッサデバッガや汎用アセンブラ・シミュレータの設計の検討を行い、拡張ハード/ソフト・コラーニングシステムの構想を検討した。今後、プロセッサデバッガの実現、及び汎用アセンブラと汎用シミュレータの設計開発を行うことにより、命令レベルのシミュレーションからハードウェア上でのデバッグまでを統合した学習システムの体系を実現する。

参考文献

[1] 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，5T-5，2004.

[2] 大八木睦 他：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，情報処理学会，第 66 回全国大会論文集，5T-6，2004.

[3] 中村浩一郎 池田修久 小柳滋 山崎勝弘：プロセッサアーキテクチャ教育用 FPGA ボードコンピュータシステムの開発，FIT2004，LC-008，2004.

[4] 中村浩一郎 池田修久 小柳滋 山崎勝弘：ハード/ソフト・コラーニングシステムにおける各種マイクロプロセッサの設計と実装，電子情報通信学会，VLSI 設計技術／コンピュータシステム研究会，2005.1.

[5] 中村浩一郎 池田修久 小柳滋 山崎勝弘：ハード/ソフト協調学習システムにおけるアーキテクチャ学習に基づいたプロセッサ設計，情報処理学会，第 67 回全国大会，1ZB-7 2005.