LC-002

プロセッサ設計支援ツールの設計・実装とハード/ソフト協調学習システムの評価

Processor Design Aided Tool Implementation and Evaluation of the Hardware / Software Co-learning System

難波 翔一朗† 志水 建太† 山崎 勝弘† 小柳 滋†

Shoichiro Namba Kenta Shimizu Katsuhiro Yamazaki Shigeru Oyanagi

1. はじめに

半導体の高集積化が進む中，システム LSI へ求められる機能は多様化し，組み込みシステムの開発には，今までの設計資産やプラットフォームを利用した開発手法がとられている．このような開発手法では，システムの性能や制約に従って，プロセッサの命令の追加や構成をカスタマイズすることが多く，ハードとソフト両方の知識に加え，プロセッサにおける命令セットとマイクロアーキテクチャの知識が必要不可欠である．また，大学教育においてもプロセッサを用いた計算機システムの学習システムは，KITE[1], PICO[2], City-1[3]等のように広く研究されている．このような背景から，本研究室では学習者が命令セットとマイクロアーキテクチャを設計することで，その境界を学習するハード/ソフト協調学習システム（Hardware Software Co-learning System:HSCS）を開発している[4]-[9]．本システムの特徴は，学生が独自の命令セットを定義でき，設計したプロセッサを FPGA ボード上で実機検証を行うことにより，ハードとソフトのトレードオフを考察できる点にある．本論文では，プロセッサ学習システムへ追加したプロセッサ設計支援ツール（命令セット定義ツール，汎用アセンブラ，プロセッサデバッガ・モニタ）について述べ，本研究室 4 回生によるプロセッサ設計の実習結果から，本ツールの検証と学習効果の評価を行う．

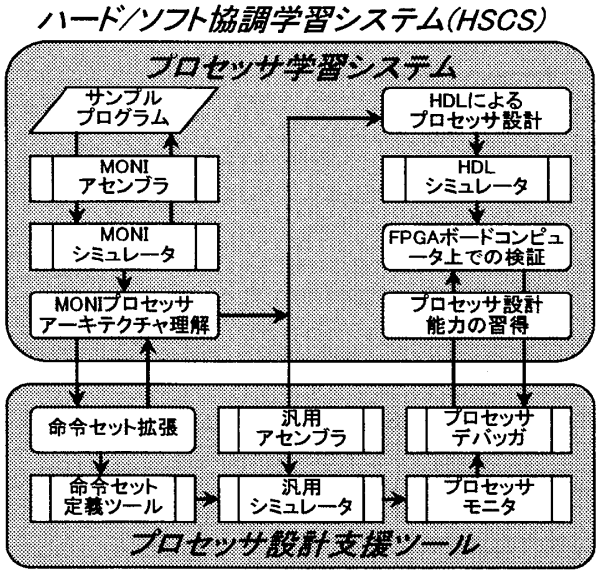

図 1 ハード/ソフト協調学習ステムの学習フロー

†立命館大学大学院 理工学研究科，Graduate School of Science and Engineering, Ritsumeikan University.

2. ハード/ソフト協調学習システム

ハード/ソフト協調学習システムとは，学習者がプロセッサを，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，コンピュータシステムを体系的に学習する教育システムである．本システムは，本研究室で MIPS のサブセットとして定義した，基本命令セット MONI[6]をベースとしたアセンブリプログラミングとプロセッサ設計を行うプロセッサ学習システム，及び学生独自の命令セットとプロセッサを設計するプロセッサ設計支援ツールによって構成される．

図 1 に HSCS の構成と学習フローを示す．まず，学習者はサンプルプログラムの課題について，MONI 命令セットを用いてアセンブリプログラミングを行い，MONI シミュレータで実行して，プロセッサアーキテクチャを理解する．また，MONI プロセッサを HDL で設計し，FPGA ボードコンピュータ上で動作検証して，プロセッサの基本的な設計能力を取得する．次に，学習者は MONI 命令セットを拡張し，サンプルプログラムをより効率よく実行できる命令セットを作成する．この際，命令セット定義ツールと汎用アセンブラを用いて，実際の課題プログラムをアセンブルしながら，拡張した命令セットが有効であるかを検証する．さらに，作成した命令セット用のプロセッサ設計と実装を行う．実機上での検証にはプロセッサモニタ・デバッガを利用する．これらのフローを通して，学習者は要求を満たす命令セットとプロセッサの設計技術を習得し，プロセッサ設計におけるハードウェアとソフトウェアの性能のトレードオフについて学ぶ．

本研究室では，2004 年度から 4 回生の卒業研究の導入実習として，HSCS での MONI アセンブリプログラミングと，プロセッサモニタ・デバッガを用いたプロセッサ設計を行い，複数のプロセッサにおいて実機上で検証を行っている．

3. プロセッサ設計支援ツールの設計

3.1 命令セット定義ツールと汎用アセンブラ

学習者に独自のプロセッサ設計を課題として与えるには，命令セットの定義とその検証を容易に繰り返し行える環境が必要である．そこで我々は，必要な情報を入力すること

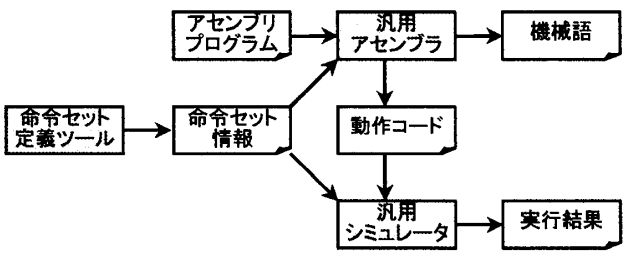

図 2 命令セット定義の流れ

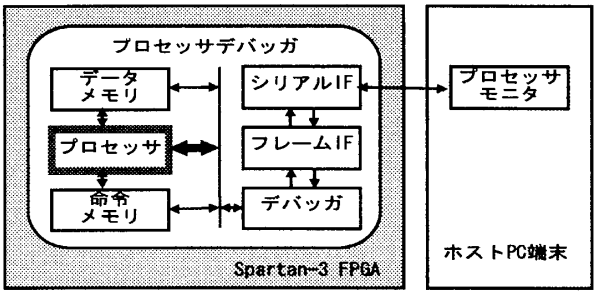

図 3　プロセッサモニタとプロセッサデバッガの構成

表 1　プロセッサデバッガのデバッグコマンド

コマンド	ターゲット	意味
send	dm / im / rf / pc	メモリ・レジスタの書込み
read	dm / im / rf / pc	メモリ・レジスタを読出し
save	dm / im / rf / pc	メモリ等をファイルに保存
list	dm / im / rf / pc	メモリ・レジスタの表示
load	dm / im / rf / pc	ファイルからロード
set	bp レジスタ	ブレイクポイントの設定
del	bp レジスタ	ブレイクポイントの削除
init	dm / im / rf / pc	メモリ・レジスタの初期化
run	プロセッサ	ブレイクポイントまで実行
run inst	プロセッサ	1 命令実行
run clk	プロセッサ	1 クロック実行
halt	プロセッサ	プロセッサの停止

dm:データメモリ,im:命令メモリ,
rf:レジスタファイル,pc:プログラムカウンタ

で命令セットの定義を行う命令セット定義ツール，及び異なる命令セットで記述されたプログラムをアセンブル可能な汎用アセンブラを開発した．図 2 に命令セット定義の流れを示す．命令セット定義ツールは既存の命令セット情報，または学習者からの定義を入力とし，命令セット情報を出力する．汎用アセンブラは学習者が定義した命令セット情報と，その命令セット専用のアセンブリプログラムを入力として，実際のプロセッサで動作させる機械語を出力する．

3.2　プロセッサモニタとプロセッサデバッガ

従来の HSCS でのプロセッサの開発では，当初，HDL シミュレータと FPGA 上の 7 セグメント LED を駆使して検証を行ってきたが，ソフトウェアを実行するプロセッサの検証にはチップ内部のレジスタやメモリの変化まで監視しなければならず，非常に労力を要する作業であった．そこで我々は，FPGA ボード上に実装したプロセッサを，コマンド操作で自由に操作でき，プログラムの実行結果やある時点でのメモリやレジスタの内容を，ホスト PC 上で確認可能なプロセッサモニタとプロセッサデバッガを開発した．図 3 にプロセッサモニタ/デバッガの構成を示す．プロセッサを実機検証する本ツールは，ホスト PC 上でコマンドの発行やデータの転送を行うプロセッサモニタ，システム全

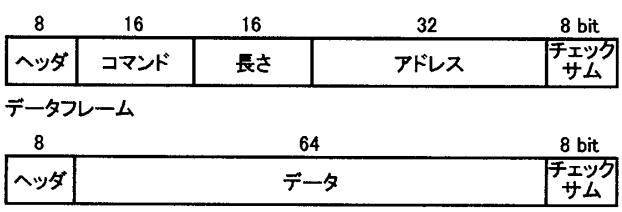

図 4　フレームの構成

表 2　プロセッサデバッガの機能

項目	特徴
利用目的	・周辺モジュールとして利用可 ・ハードウェアデバッガとして利用可
通信機能	・RS232C を用いたシリア通信 ・データ転送は 10 バイト(=1 フレーム)の単位で行う
メモリアーキテクチャ	・ハーバードアーキテクチャを採用 ・入出力ポートの共有/個別を選択可 ・アクセスのタイミングはクロックの立ち上がり/立ち下がりで選択可 ・クロック速度を調節することで同期/非同期アクセスを選択可
システムバス	・デバッグ用バスとプロセッサ用バスを分離 ・双方向バスとのインタフェースモジュールによって簡単に接続可

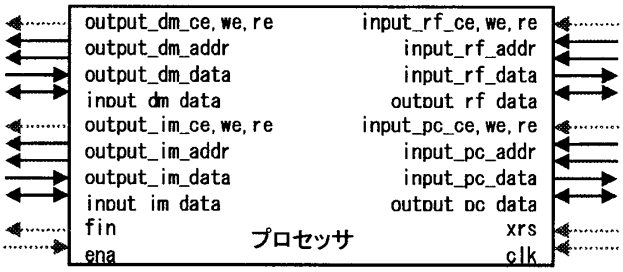

図 5　デバッガに搭載するプロセッサのインタフェース

体の制御を行うデバッガ，及び通信フレームを制御するフレーム IF/シリアル IF から構成される．

(1) プロセッサモニタ

プロセッサモニタは，ホストPC上から実機上のプロセッサの制御とデバッグを行うために，ユーザが入力したデバッグコマンドを解釈し，プロセッサデバッガへ処理指示を与えるモジュールである．表 1に本ツールで実装したコマンドを示す．デバッグコマンドは，メモリ/レジスタの読み書き，ファイル入出力，ブレイクポイントの設定，各種実行用のコマンドを用意した．本ツールでは，実機上のデバッガと通信するために，図 4 に示すフレームを利用する．フレームは，実機上で解釈・実行しやすいように，コマンドフレームとデータフレームの 2 種類を設けた．制御用のコマンドフレームをプロセッサデバッガに送信し，必要であればデータフレームによって転送データの送受信を行う．

(2) プロセッサデバッガ

プロセッサデバッガは，FPGAボード上にユーザが作成したプロセッサを実装するための周辺モジュールと，ハードウェアデバッガとしての機能をRTLで実現したIPである．設計したプロセッサをプロセッサデバッガに搭載し，論理合成することで，FPGA上に実装可能なデータを容易に生成できる．プロセッサデバッガの機能を表 2に示す．本モジュールは，コマンドフレームによってプロセッサの実行/停止や，メモリ/レジスタのデータの更新を行う．搭載するプロセッサの内部アーキテクチャは，設計者によって様々であるが，図 5 のインタフェースに統一することで，一つのプロセッサデバッガで様々なプロセッサが検証可能になった．また，メモリとの入出力は表 2のメモリアーキテクチャ項目に示す通り，複数の項目を組み合わせて選択でき，自由度の高いプロセッサ設計が可能である．データ/命令メモリは，学習用として理解しやすいハーバードアーキテク

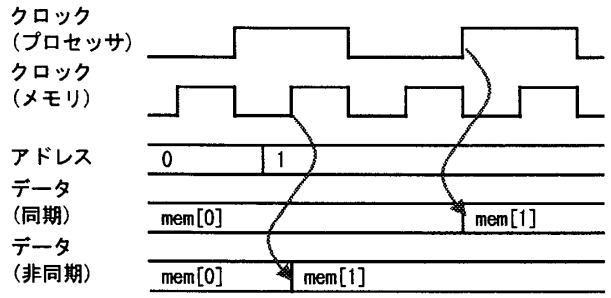

図 6　メモリデータ読み出しのクロック同期 / 非同期

表 3　SOAR を含むプロセッサデバッガの規模と性能

プロセッサ アーキテクチャ	メモリ アクセス	スライス 総数 1920	FPGA 使用率 (%)	最高動作 周波数 (MHz)
SOAR(16 ビット)	同期	504	26	39.2
SOAR(32 ビット)	同期	895	44	37.7
プロセッサデバッガ (SOAR 含む)	—	1,745	91	49.8

チャを対象とし，入出力ポートの共有/個別，アクセスのタイミングの立ち上がり/立ち下がり，データ読み出しのクロック同期/非同期をユーザによって選択できる．クロック同期 / 非同期の違いを図 6 に示す．クロック非同期のメモリアクセスでは，プロセッサよりも数十倍高速な周波数のクロックを用いることで，仮想的にプロセッサクロックに同期しないアクセスを実現している．

4. プロセッサ設計支援ツールの実装

4.1 プロセッサデバッガの実装

本システムの開発では，搭載するプロセッサ，及びプロセッサデバッガを Verilog-HDL で，ホスト PC 上で動作するプロセッサモニタ，命令セット定義ツール，及び汎用アセンブラを C++で作成した．開発環境は Micorsoft 社の Visual Studio 2005，及び Xilinx 社の ISE9.1i を用いる．シリアルポートの規格は RS232C を，ソフトウェア側のポートには Windows の API を利用する．対象ボードは 20 万システムゲートの FPGA を搭載する Spartan-3 Starter Kit ボードを用い，クロックの動作周波数は 50MHz，シリアル通信のボーレートは 119,200bps をターゲットとする．

本ツールの実装はまず，2004 年度の設計で完成している 16 ビット単一サイクルSOARプロセッサ[7]を用いて検証し，プロセッサデバッガ自体が正しく動くか検証を行う．その後，本研究室の 4 回生が作成したプロセッサを実装して，デバッガとしての動作検証を行い，本ツールとHSCSの評価を行う．表 3にSOARを含めたプロセッサデバッガの設計規模と性能を示す．プロセッサデバッガ(SOAR含む)全体のFPGAの使用率は 91%であるが，残り 9%程度の範囲であればSOAR以上のプロセッサの搭載も可能である．また，表 1のデバッグコマンドは，SOARプロセッサに対しては，メモリやレジスタのデータの読み書き，プロセッサの実行と停止，ブレイクポイントの設定のコマンドを実装し，その動作を確認した．これにより，デバッガとして必要な機能を満たすコマンドを提供できたと言える．

表 4　HSCS での学習時間

プロセッサ アーキテクチャ	メモリ アクセス	MONI アセンブリ プログラミング 問題数 / 時間	HDL 設計時間
MONI 単一サイクル	非同期	8 / 25	24
	同期		27
MONI 3 段マルチサイクル	非同期		27
	同期		27
MONI 4 段マルチサイクル	非同期		9
	同期		11
ARM ライク	同期	1 / 35	360

表 5　プロセッサの実装結果

プロセッサ アーキテクチャ		メモリ アクセス	スライス 総数 1920	FPGA 使用率 (%)	最高動作 周波数 (MHz)
MONI	単一サイクル	非同期	720	37	31.1
		同期	726	37	31.1
	3段マルチサイクル	非同期	899	46	43.7
		同期	919	47	43.5
	4段マルチサイクル	非同期	824	42	44.4
		同期	882	45	45.0
ARM ライク		同期	1122	58	49.8

4.2 学生によるプロセッサを用いた動作検証

MIPS アーキテクチャの授業を受けた，本研究室の二人の 4 回生が作成した複数のプロセッサをプロセッサデバッガに搭載することで，本ツールの有効性の検証と HSCS 全体の学習効果を示す．2006 年度，一人の学生が MONI を 6 種類のアーキテクチャで実装を行った．もう一人の学生はARMプロセッサを参考に作成した独自のプロセッサ（ARM ライクプロセッサと呼ぶ）を設計した．HSCSを利用してこれらのプロセッサのために要した学習・設計時間を表 4に示し，それぞれのプロセッサの実装規模を表 5 に示す．

HSCS ではプロセッサを設計する前に，MONI のアセンブリプログラミングによる学習を行っている．MONI 単一/3 段マルチ/4 段マルチを作成した学生は，HDL の設計前に 8 プログラム（整数の階乗・三角形の種類判定・二次方程式の根の判別・素数判定・符号付き整数の商と剰余・一次方程式直線・8×8 行列引き算/掛け算/足し算・BCD コード加算/減算）について，25 時間程度の時間をかけて学習した結果，メモリアクセスの違いを含めると 6 種類のアーキテクチャのプロセッサを設計し，実機上で動作検証を行うことができた．一方，ARM ライクプロセッサを設計した学生は，素数判定のプログラムを 35 時間かけて行った後，HDL の設計で 360 時間程度かかる結果となった．ARM ライクプロセッサにおいては，HDL シミュレータ上で N までの和，素数判定のプログラムが正しく動いていることを確認した．

実装規模と動作周波数に着目すると，MONI の単一サイクルより 3 段，4 段マルチサイクルの方が速い結果を得た．また，これらのプロセッサの設計時間に着目すると，それぞれ 30 時間以下となった．ARM ライクプロセッサでは，MONI とは異なり，一から命令セットの選定とアーキテクチャの設計を行わなければならず，非常に時間のかかる結果となった．

表 6　SOAR の設計時間

工程	設計時間(時間)
命令セット設計	10
HDL によるプロセッサ設計	40
FPGA 実装環境の構築	40
FPGA 上での動作検証	60
合計	150

4.3 プロセッサデバッガを利用した実機検証

MONI 単一/3 段マルチ/4 段マルチプロセッサの実機検証では，プロセッサデバッガを利用し実機上に実装を行った．利用したデバッグコマンドは im / dm へのデータの送受信とプロセッサの run / halt 命令である．プロセッサの実機での検証環境として，LED やボードのスイッチを利用せず，ホスト PC 上のプロセッサモニタからのコマンド操作で，データの送受信が確立できた．表 1 のデバッグコマンドは全て使用可能であるので，これらを用いてプロセッサのデバッグを行うことが今後の課題である．

5. ハード/ソフト協調学習システムの評価

5.1 プロセッサデバッガの評価

表 6はSOARプロセッサの設計時間である．SOARプロセッサ設計者はMIPSアーキテクチャの授業を受け、Velirogをマスターした上で，プロセッサデバッガが無い従来のHSCSを用いた．FPGA上で動作検証を行った結果，約 150 時間かかった．今回の実験では，プロセッサデバッガを用いてMONIをFPGAボード上に実装し，30 時間未満で設計できた．これは，本ツールによって，実機への実装環境が整えられていたことや，データ転送とプロセッサ実行のコマンドが利用できたことが大きく，プロセッサ設計支援ツールの目的であるプロセッサの実装環境を整えた効果が現れたと評価できる．

デバッグの機能の検証では，SOAR プロセッサを実装し，ボードのスイッチやLEDを使用することなく実機検証できることを確認でき，デバッガとしての機能が有効であることが示せた．また，MONI 以外の命令セットとして ARM ライクなプロセッサの設計例もあり，本システム上で複数の命令セットのプロセッサ設計ができることを示した．

5.2 ハード/ソフト協調学習システムの評価

2006 年度の学習では，HSCS でのソフトウェア学習を行い，その後ハードウェア学習においてプロセッサ設計を行った．この結果，MONI プロセッサの設計に挑戦した学生では，アセンブリプログラミングとプロセッサ設計において同じ命令セットを使用したことから，比較的短時間での FPGA ボード上での検証まで行うことができた．また，アセンブリプログラミング時に用いた MONI シミュレータの上で，表示されていたアーキテクチャの図によって，回路イメージが持てたことも有効であると思われる．一方，ARM ライクプロセッサの設計では，アセンブリプログラミングの評価を行える環境が不十分であったため，命令セットレベルでの学習が十分に行えず，HDL によるプロセッサ設計では難航する結果となったと考えられる．しかし，MONI とは異なる命令セットを考案し試作できたことは，命令セット設計とプロセッサ実装の学習フローを取り込んだ本システムの有用性を示す結果である．今後，命令セット定義ツールと汎用アセンブラ・シミュレータを用いて命令セットの評価を行い，FPGA ボード上に実装することや、学生の人数を増やし HSCS の検証を行うことが課題となる。

HSCS のソフトウェア学習とハードウェア学習の関係では，MONI のアセンブリプログラミングにおいて 8 題のプログラミングを行った学生が，多くのプロセッサアーキテクチャで FPGA ボード上に実装を行う結果となった．8 題の課題は比較的容易な問題が多い．このことから，より多くのアセンブリプログラミング学習が，HDL によるプロセッサ設計で有効であると考察できる．

6. おわりに

本論文では，ハード/ソフト協調学習システムにおけるプロセッサ設計支援ツールの実装について，主にプロセッサデバッガを中心に述べた．データ転送，プロセッサの実行，及びブレイクポイントのデバッグコマンドを用いて，SOAR プロセッサが正常に動作することを確認した．また，プロセッサデバッガを用いた本研究室の 4 回生によるプロセッサ設計を示し，6 種類の MONI プロセッサの実装と ARM ライクプロセッサの設計について述べた．MONI プロセッサの実装では，30 時間未満の設計時間で HDL による設計を行え，プロセッサ設計支援ツールによる設計時間の短縮を図ることができた．

今後の課題として，プロセッサデバッガを用いたプロセッサの実機検証，ARM ライクプロセッサの FPGA ボード上への実装，及び独自に定義した命令セットの評価などが挙げられる．将来的には，本システムを学部の学生実験で利用することが目標である．

参考文献

[1] 末吉，他：KITE マイクロプロセッサによる計算機工学教育支援システム，IEICE，Vol.J84-D-1，No.6，pp.917-926，2001.

[2] 西村，他：教育用パイプライン処理マイクロプロセッサ PICO^2 の開発，IPSJ，Vol.2000，No.2，pp.141-148，2000.

[3] 高橋，他：FPGA を用いたスーパースカラ設計教育に関する一考察，IPSJ，Vol.2003，No.49，pp.43-50，2003.

[4] 池田，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.

[5] 大八木，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.

[6] 中村，他：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.

[7] 難波，他：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，FIT2005，LC-002，2005.

[8] Hoang Anh Tuan，et al.:A FPGA Based Hardware / Software Co-learning System，IEICE Technical Report，RECONF 2005-48，pp.43-48，2005.

[9] 難波，他：ハード/ソフト協調学習システムのための命令セット定義ツールとプロセッサデバッガの開発，FIT2006，N-009，2006.