N_009

ハード/ソフト協調学習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

Development of an Instruction Set Definition Tool and a Processor Debugger for the Hardware / Software Co-learning System

難波 翔一朗† 中村浩一郎† Hoang Anh Tuan † 山崎 勝弘† 小柳 滋†

Shoichiro Namba Koichiro Nakamura Hoang Anh Tuan Katsuhiro Yamazaki Shigeru Oyanagi

1. はじめに

近年，LSI の高集積化が進む中，要求されるシステム規模は爆発的に増加し，システム LSI の開発ではハードウェアとソフトウェアを仕様検討段階から併せて設計するハード/ソフト協調設計の手法がとられている．また，システムの性能や制約に従って LSI に組み込むプロセッサには命令の追加や構成をカスタマイズできるものが増えており，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は半導体設計には必要不可欠なものである．このような背景から，大学の計算機工学の教育においてもハードとソフト両方の知識を学習する体制が求められており，本研究室では特にプロセッサに着目し，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・コラーニングシステムを開発している[1]-[6]．本論文では，コラーニングシステムにおける命令セット定義ツールとプロセッサデバッガについて示し，その実装と評価について述べる．

2. ハード/ソフト・コラーニングシステムの概要

2.1 従来のハード/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（以下 HSCS と略す）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，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．図 1 の上部に HSCS の学習フローを示す．学習内容は，基本命令セット MONI のプロセッサアーキテクチャ（シングル，マルチ，パイプライン，スーパスカラ）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，MONI 命令セットに対応するプロセッサの設計，検証，実装(ハードウェア学習)からなる．我々は HSCS 上で MONI の他に SOAR[5]，TOHD[6]プロセッサを実装し，複数のプロセッサでの検証を行ってきた[4]．

2.2 命令セット定義可能な学習システムへの拡張

HSCS は基本命令セット MONI をベースにした学習システムであるが，組み込みシステム向けのプロセッサやマイコンの設計では，システムに最適な命令セットを検討し，プロセッサのアーキテクチャをカスタマイズする必要がある．大学教育においても学習者が独自の命令セットを考案し，プロセッサの作りこみができることが望ましい．そこで我々は，従来の HSCS に新たに命令セット定義可能な機能を追加し，学生独自のプロセッサ設計に特化した学習環境を開発した．本学習システムでは，従来のシステムに加えて，ホスト端末上で利用する命令セット定義ツールと，

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

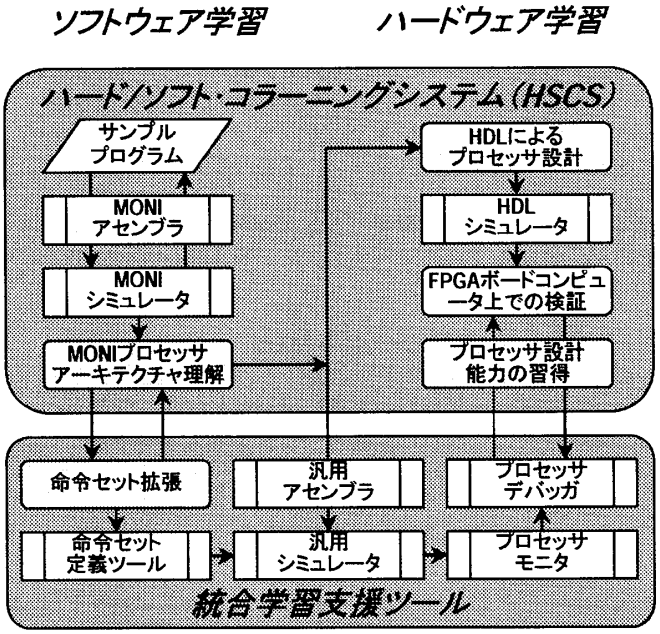

図 1：命令セット定義可能な協調学習システムの学習フロー

FPGA 上でプロセッサの動作検証を行うプロセッサデバッガを盛り込み，それらを密接に連携させてプロセッサのソフトウェアとハードウェアの関係を学ぶ．

図 1 に命令セット定義可能に拡張した HSCS の学習フローを示す．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 シミュレータで評価する．ここで基本的なプロセッサのハードウェアアーキテクチャや命令実行時の動作について理解し，アセンブリプログラミングに必要な最低限の命令やアドレッシングモードについて学ぶ．

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．命令セットの定義には命令セット定義ツールを，評価には汎用アセンブラ・シミュレータを用いることで，実際の課題プログラムをシミュレーションしながら，拡張した命令セットが有効であるかを検証する．ここでは実行効率を意識した命令セット設計を行い，ソフトウェア面での高効率化技術を養う．

最後に，作成した命令セット用のプロセッサ設計と実装を行う．ここでは特に，論理的には実現可能な命令でも，実際のハードウェアにした場合に，最大動作周波数の低下や回路規模の拡大を引き起こすことを理解し，要求を満たすハードウェア設計技術を習得する．要求を満たさない場合は再度命令セット定義からやり直し，命令セットとプロセッサの設計を繰り返し行うことで，ハードウェアとソフ

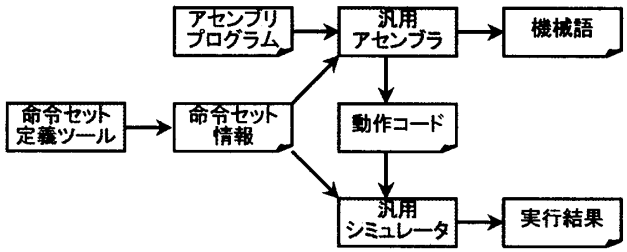

図 2：命令セット定義の流れ

トウェアの境界を理解する．学習者が設計したプロセッサコアは我々が開発したプロセッサ評価用 FPGA ボードコンピュータシステム[3]に搭載し，プロセッサデバッガ・モニタを用いて実機での検証を行う．これらのツールを用いることで FPGA 内部のレジスタやメモリの値をホスト PC 上から読み書きでき，学習者は自作プロセッサの検証に集中し，ハード/ソフトのトレードオフという本来の学習目的を強く理解しながら学習ができる．

3. 命令セット定義ツールの設計

学習者に命令セット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で命令セットの定義を行う命令セット定義ツール，及び命令セットを評価する汎用アセンブラ・シミュレータを開発した．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，もしく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．命令セット情報にはプロセッサのレジスタ構成や命令長，命令の動作内容などの定義が含まれる．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と汎用シミュレータで用いる動作コードを出力する．動作コードはアセンブリコードの命令群を汎用シミュレータ上で実行可能に変換した動作内容の情報である．汎用シミュレータでは動作コードと命令セット情報のレジスタ構成などのアーキテクチャの情報を読み込み，シミュレーションの実行結果を出力する．

3.1 命令セット定義ツール

一般的に命令セットを設計するためには，アドレッシングモード，命令形式，レジスタ構成，命令動作について定義する必要がある．本ツールでは命令長，レジスタ数，命令セットのフィールド，フォーマット，命令と順を追って定義でき，専門の記述言語の習得なしに命令セット設計に専念できる環境を提供する．本ツールで設定可能な項目を表 1 に示す．現在は，定義する命令セットを RISC に限定しているため，命令長は 8，16，32，64 の固定長から選択する．定義できるレジスタは汎用レジスタとプログラムカウンタ（PC），スタックポインタ（SP）である．その他の専用レジスタを用いる場合は汎用レジスタを余分に定義して代替とする．アドレッシングモードはレジスタ間接と即値を定義可能としたが，命令の動作記述を工夫することで様々なモードに対応できる．

表 1：命令セット定義ツールで設定できる項目

項目	対応
命令長	8，16，32，64bit 固定長
命令セット・命令形式	自由に設定可能
プロセッサの構成様式	RISC を対象
レジスタ	汎用レジスタ，PC，SP
アドレッシングモード	レジスタ間接，即値，他
メモリ・アーキテクチャ	自由に設定可能
擬似命令（マクロ命令）	対応

3.2 汎用アセンブラ

汎用アセンブラの最大の特徴は，命令セット情報を与えることで，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できる点にある．命令セット情報に含まれる命令形式と各命令に対応する機械語の情報をアセンブル時に読み出すことで，様々な命令セット用のアセンブリソースをアセンブルできる．汎用アセンブラを利用することで，HSCS 上で学習者が独自に命令セットを設計し，実機のプロセッサで実践的な評価ができる．汎用化のためにアセンブリソースの構文は以下に示すものとした．

- ラベル，オペコードはアルファベットのみ
- 定数表現は 10 進数表示
- レジスタ値は‘$’にアドレスを付けて参照
- メモリ値は‘*’にアドレスを付けて参照
- コメントは”//”～改行まで

図 3 に汎用アセンブラのプログラム例として MONI 命令セット用の最大値検索のアセンブリソースを示す．

4. プロセッサデバッガの設計

HSCS でのプロセッサの開発では，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 ボードコンピュータ上に実装したプロセッサを学習者が自由に操作することで，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，変更できるプロセッサデバッガを開発した．

```
// 最大値検索のプログラム
    CLEAR  $0          // $0 = 0
    LD     $0  *$0     // $0 = MEM[$0] (number of data)
    ADDI   $4   $0  1  // $4 = $0 + 1  (store address)
    LD     $1  *$0     // $1 = MEM[$0] (max value)
LP: SUBI   $0   $0  1  // $0 = $0 - 1  (load address)
    BEQZ   $0   BR     // if($0 == 0) goto BR
    LD     $2   $0     // $2 = MEM[$0]
    SLT    $3   $1  $2 // $3 = ($1 < $2) ? 1 : 0
    BNEZ   $3   LP     // if($3 != 0) goto LP
    COPY   $1   $2     // $1 = $2
    JUMP   LP          // goto LP
BR: ST    *$4   $1     // MEM[$4] = $1
    HALT
```

図 3：汎用アセンブラのプログラム例

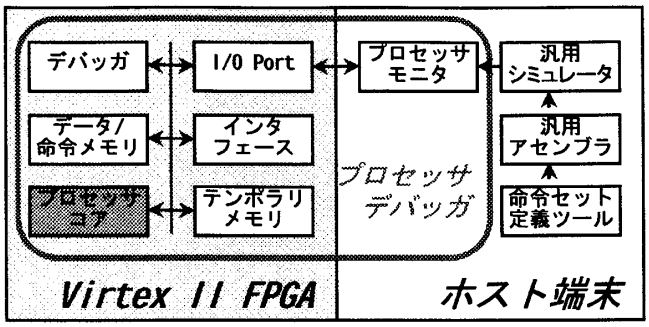

図4：FPGAボードのシステム構成

4.1 プロセッサデバッガの構成

プロセッサデバッガを含むFPGAボード上のシステム構成を図4に示す．プロセッサデバッガはシステム全体を制御するデバッガ，通信フレームを制御するインタフェース，ホストPCとのデータ通信用パラレルポートドライバのI/O Port，及びホストPC上でコマンドの発行やデータの転送を行うソフトウェアのプロセッサモニタなどで構成される．また，システム内にはデータ/命令メモリとテンポラリメモリを用意する．プロセッサコアがアクセスするデータ/命令メモリにはハーバードアーキテクチャを採用し，学習者が様々なマイクロアーキテクチャでプロセッサコアを設計できることを考慮した．テンポラリメモリはホストPCとの送受信データを一時的にためることによって，内部での制御を単純化するとともに，データの信頼性を確保する．

デバッガはプロセッサコアの実行，ブレイクポイントの設定，ホストPC・FPGA内部のメモリ，レジスタ間のデータ転送やFPGA内部のメモリ間のデータ転送などシステム全体の制御を行う．インタフェースは，ホストPCから送られてくるコマンドフレームとデータフレームを識別する．データフレームならばテンポラリメモリへのデータ転送を行い，コマンドフレームならばデバッガがコマンドを読み出すために，インタフェース内のレジスタに格納する．I/O Portはパラレルケーブルの送受信ドライバとして動作する．

プロセッサデバッガでは，ホストPC上のプロセッサモニタとFPGA上のインタフェースモジュールをパラレルケーブルで接続し，コマンドフレームとデータフレームを送受信することで，メモリの読み書き，プロセッサコアの実行など，様々な制御を行う．図5にデータ・コマンドフレームの構成を示す．各フレームはそれぞれを識別するASCIIコードのヘッダから始まる．データフレームにはテキスト開始の意味を持つ0x02（STX：Start of Text）を，コマンドフレームにはヘッダ開始の意味を持つ0x01（SOH：Start of Header）を割り当てる．コマンドフレームはヘッダ，デバッグ操作用のコマンドコード，転送データ数，転送開始アドレス，及びエラー検出用のチェックサムから構成されている．データフレームはヘッダ，128ビットのデータ，及びチェックサムで構成される．

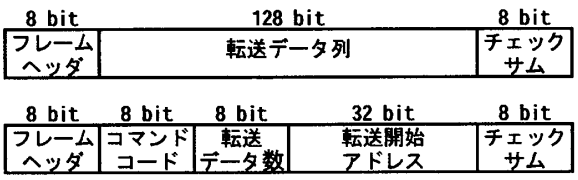

図5：データ・コマンドフレームの構成

表2：プロセッサデバッガのコマンド一覧

コマンド	意味
run [モード]	プロセッサの実行
halt	プロセッサの停止
set [アドレス]	ブレイクポイントの設定
del [アドレス]	ブレイクポイントの削除
read [ターゲット]	メモリ，レジスタの読み出し
send [ターゲット]	メモリ，レジスタの書き込み
load [ファイル]	送信データをファイルから読み出し
save [ファイル]	受信データをファイルに書き込み
init	プロセッサモニタのリセット
exit	プロセッサモニタの終了

4.2 プロセッサモニタの設計

プロセッサモニタはFPGAボードに対してコマンドの発行や，データ転送を行う．プロセッサモニタが発行できるコマンド一覧を表2に示す．プロセッサの実行には，runコマンドを用い，ブレイク実行，1命令実行，通常実行のモードでの実行が可能である．ブレイクポイントの設定と削除には，命令アドレスを引数として持たせる．データ/命令メモリへの書き込みは，書き込むデータを予めホストPC上にバイナリファイルで用意しておき，loadコマンドによってプロセッサモニタ上に読み込む．その後，sendコマンドによって読み出したファイルのデータをデータメモリ，命令メモリ，レジスタに書き込む．書き込み対象とするデバイスはsendコマンドの引数として指定する．

5. ハード/ソフト・コラーニングシステムへの実装

本システムの開発では，デバッガ，インタフェースなどの内部ロジックをVerilog-HDLで，FPGAボードとホストPCとのデータ送受信で使用するパラレルケーブルドライバをHandel-C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++で行った．FPGAボードとホストPC間のデータの送受信は，Celoxicaから提供されるライブラリをC++，Handel-Cで使用することによって，比較的容易に確立することができた．

5.1 命令セット定義ツールの実装

命令セット定義ツールは，UIをトップとして命令セットの情報を管理するクラス群と，解析を行うクラス群の二つに分けられる．図6に命令セット定義ツールのクラス図を

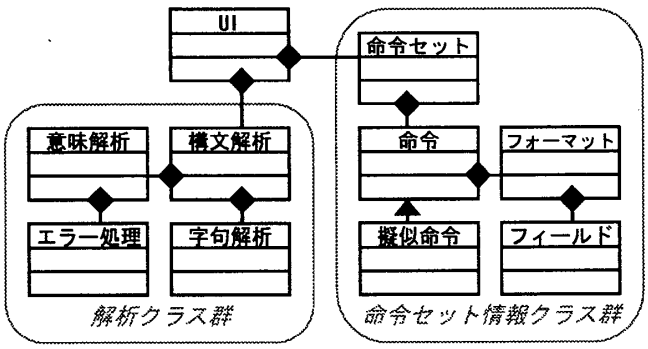

図6：命令セット定義ツールのクラス図

示す．命令セット情報のクラス群では，命令セット名，命令長，フィールド，フォーマット，命令，擬似命令の定義や保持を行う．また，汎用アセンブラ・シミュレータの動作時には命令セットの情報を提供するデータベースとして機能する．命令セット情報の解析を行うクラス群では，定義済みの命令セット情報の記述について字句解析，構文解析，意味解析を行い，再びツール上で編集を行えるようデータの整合性をチェックする役割を担う．

5.2 プロセッサデバッガの実装

本システムでは 100 万システムゲート相当の Xilinx Virtex II が搭載されている Celoxica 社の RC200 ボードを用いる．本ボードでは，ホスト PC との通信に用いるパラレルポートのデバイスが CPLD を介して接続されているため，FPGA 上のデバッガが CPLD を制御することで通信を確立した．

プロセッサデバッガを含む本システムの実装規模と動作周波数を表 3 に示す．プロセッサコアには MONI シングルサイクルプロセッサを用いた．100 万システムゲートのうち全体の使用率は，Flip Flop で 18%，LUT で 34%であった．FPGA の使用率には十分な余裕があるため，様々なプロセッサコアを搭載することが可能であることが分かる．また，最大動作周波数は 34.4MHz であり，本システムで使用する 24.6MHz のクロック周波数以上であった．またプロセッサコアの最大動作周波数が 51.2MHz と他のモジュールと比べて低いことから，プロセッサコアの最大動作周波数が向上すれば，システム全体の最大動作周波数も向上するものと思われる．

5.3 評価と考察

命令セット定義ツールを用いて MONI，SOAR 命令セットを定義し，汎用アセンブラで最大値・最大公約数・挿入ソートのプログラムをアセンブルできることを確認した．また，本研究室内の学生に使用してもらい，操作性や汎用性についてアンケートを実施し，評価を行った．寄せられた意見には，学習用として十分である，作成したい命令セットが実現できるなどがあり，命令セットを定義するツールとしてある程度の汎用性や機能性を達成できた．その反面，1 命令ごとの定義が細かく複数の命令を定義するのは面倒である，定義すべき項目が多いなどの意見もあり，学生実験で使用するためには，汎用性を確保しつつも定義可能な項目を限定し，単純な操作のみで利用できるようにする必要がある．

プロセッサデバッガを用いた FPGA ボードコンピュータ上では MONI シングルサイクルプロセッサの実装を行った．表 4 に従来の HSCS との機能比較を示す．プロセッサデバッガを用いることにより，従来の HSCS では不可能であったレジスタへの書き込みや読み出し，ブレイクポイントの設定や解除が可能となり，プログラム実行途中でのメモリとレジスタのデータ確認，書き換えが行えるようになった．また，HSCS では 7 セグメント LED 上での確認やデータの転送に複数のツールを用いた操作が必要であったが，全ての操作をプロセッサモニタ上で行えることから，学習の効率，設計に関する利便性が大いに向上した．汎用シミュレータについては，基本命令のシミュレーションが可能であり，どの程度の汎用性を持たせるかについて，現在，検討を進めている．

表 3：実装規模と動作周波数

	Flip Flop 数	使用率 (%)	LUT 数	使用率 (%)	動作周波数 (MHz)
CPU (MONI)	148	1	1017	9	51.2
I/O Port	1342	13	810	7	182.9
Debugger	296	2	1168	11	74.5
Interface	155	1	194	1	157.6
All	1872	18	3535	34	34.4

表 4：機能比較

	従来の HSCS	プロセッサモニタ
メモリアクセス	専用ツールを使用	データの送受信可
レジスタアクセス	不可	データの送受信可
ブレイク実行	不可	設定・削除可
プロセッサ実行	ボードのボタン操作	実行可能

6. おわりに

本論文では，ハード/ソフト・コラーニングシステムのための命令セット定義ツール，汎用アセンブラ，及びプロセッサデバッガの設計と実装について述べた．命令セット定義ツールによって学生が独自の命令セットを定義でき，汎用アセンブラによって定義した命令セットでのプログラムがアセンブル可能になった．併せて，プロセッサデバッガを用い FPGA ボード上で自作プロセッサの動作検証を行うことで，プロセッサアーキテクチャ，命令セットアーキテクチャ，及び両者の関係をより深く学習することができる．今後，汎用シミュレータを完成させ，命令セット定義から FPGA ボードコンピュータ上での検証までを一貫して行える環境の構築を目指す．

参考文献

[1] 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.

[2] 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.

[3] 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.

[4] 中村，他：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] 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.

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