計算機アーキテクチャ 146－１４
（２００２．２．１）

SOPC ボードを使ったコンピュータシステムの設計実装およびネットワーク実験への応用

前田 洋一, 楯岡 孝道, 鈴木 貢, 阿部 公輝
電気通信大学情報工学科

プロセッサ、コンパイラ、コンピュータネットワークはいずれも情報関連学科において重要な技術であるが、実験演習ではそれぞれが講義と連動しつつ個別に扱われることが多い。電気通信大学情報工学科では、学生が作成したネットワークプロトコルスタック (TinyIP) を、学生が作成したコンパイラ (TinyC) でコンパイルし、学生が作成したプロセッサ (MinIPS) 上で実行する実験 (CNP 実験) が開発され、学部 3 年生に対して実施されている。学生はチームを組み、メンバー間で協調をとりながら実験を進める。実験では、分割された課題間のインタフェースの理解、個別課題の設計実装、統合されたものが動作しないときの問題解決、仕様の調整などが求められる。本論文では、プロセッサ設計実装部分を中心に、CNP 実験の概要、実施内容、結果、および評価を述べる。

Design and Implementation of a Computer System Using SOPC Board and Its Application to Computer Networking Laboratory

Youichi Maeda, Takamichi Tateoka, Mitsugu Suzuki, Kôki Abe
Department of Computer Science, The University of Electro-Communications

Processor organization, compiler design, and computer networking are important materials covered by comuputer science curricula. They are often treated independently in laboratories associated with corresponding lecture courses. A laboratory for juniors majoring in computer science at the University of Electro-Communications is now under way, where a networking protocol stack(called TinyIP) implemented by students is converted to object codes by a compiler(called TinyC) implemented by students, which in turn are executed on a processor(called MinIPS) implemented also by students. Groups of students each organized as a team cooperatively perform the experiments. The goal of the laboratory(called CNP) is for each student to understand the interfaces between subject components, to design and implement an assigned component, to debug cooperatively the integrated system, as well as to adjust the specifications of components. In this paper, focusing on the processor organization and design, we describe the outline, results, and evaluation of the CNP laboratory.

1 はじめに

プロセッサ、コンパイラ、コンピュータネットワークはいずれも情報関連学科において重要な技術であり、実験演習ではそれぞれ個別に講義と連動して扱われている。プロセッサの設計実装を内容とする学生実験の例は多く (たとえば [1])、コンパイラの演習実験も一般的である。また、ネットワークについても実験で扱われている例がある [2]。

一方、近年の様々な機器にはプロセッサが組み込まれ、システムとして使用されることが多い (たとえば [3])。また、トランジスタの高集積化に牽引され、システム LSI が発展してきている [4]。特定の応用に対してシステム全体としての性能/コストを上げるには、ソフトウェアとハードウェアの関係や応用対象の特性を十分把握していることが重要である。また、広い分野にまたがる問題は、複数の人間により分担し統合する方法によらなければ短期間での解決は難しい。

電気通信大学情報工学科では、学生が作成したネットワークプロトコルスタック (TinyIP) を、学生が作成したコンパイラ (TinyC) でコンパイルし、学生が作成したプロセッサ (MinIPS) 上で実行する実験 (CNP 実験) が開発され、学部 3 年生に対して実施されている [5]。学生はチームを組み、メンバー間で協調をとりながら実験を進める。実験では、分割された課題間のインタフェースの理解、個別課題の設計実装、統合されたものが動作しないときの問題解決、仕様の調整などが求められる。

プロセッサ実験 (P 実験) においては、32 ビットパイプライン化 RISC プロセッサのコア部分を設計実装する。プロセッサコアをはじめ、Ethernet コントローラ、プログラムロード/モニタのための RS232C インタフェースなど、コンピュータシステムの主要な部分はすべて 40 万ゲート FPGA ワンチップ上に実装される。学生は、FPGA がメモリや入出力トランシーバとともに搭載された市販のボード (SOPC ボード) と FPGA メーカから提供される論理合成ツールなどを用い、Verilog-HDL により設計を行なう。ネットワーク実験 (N 実験) は、Linux マシンと SOPC ボード上に実装された MinIPS コンピュータ

システムとのUDP通信を行なうためのプログラムを作成する。コンパイラ実験(C実験)では、MinIPSのオブジェクトコードを生成するクロスコンパイラを作成する。MinIPSシミュレータ[5]を用いて検証されたバイナリコードを実機へロードし実行する。

本論文では、プロセッサ設計実装部分を中心に、2章でP実験の内容を、3章でCNP実験の概要を、4章では実施内容と方法、結果、および評価を述べる。

2 ハードウェアの設計実装

2.1 設計要件

プロセッサの設計要件としては、学部3年生の実験教材として簡潔かつ現代的であること、次にCNP実験で構築する統合システムにおける使用に耐えうるものであることである。さらに講義との連携から教科書[6]で扱われているプロセッサに準拠したものであることが望ましい。

コンピュータシステムとしての要件としては、プロトコルスタックによる通信を行なうための通信ポートを有すること、通信プログラムを動作させるのに十分なメモリ量を有すること、プログラムロードとコンソール機能を有することである。

2.2 システム構成

設計したシステムの構成は、図1の通りである。各部の詳細は以下の各節で述べる。

学生が設計したシステムを実際に動作させるための実装デバイスとしては書換え可能な大容量FPGAを使用する。

Altera社のSOPCボード(System on a Programmable Chip Development Board)[7]は、同社のAPEXシリーズのFPGAが実装された評価用ボードである。このSOPCボードのFPGAには、論理素子を用いて40万ゲート相当の回路を、ESB(EmbeddedSystem Blocks)を用いて最大20kByteのメモリを、実装することができる。また、FPGA以外にも、SSRAM(Synchronous SRAM)や、RS232C、Ethernet等のトランシーバも備えているため、付加回路なしで要件を満たす実験装置を構成できる。

以上の理由からこのシステムの実装にはSOPCボードを使用する。

2.2.1 MinIPSプロセッサ

MinIPSは、MIPS[6]に準拠したアーキテクチャを持っている32bitのRISCプロセッサである[8]。その特徴を次に列挙する。

- MIPS R2000のサブセット
- コプロセッサ命令と乗除算命令などは含まない
- 命令フェッチ(IF)、命令デコード(ID)、実行(EX)、メモリアクセス(MEM)、書き込み(WB)からなる5段パイプライン制御
- 例外は、ハードウエア割り込みのみ(専用の割り込み処理用命令を持つ)

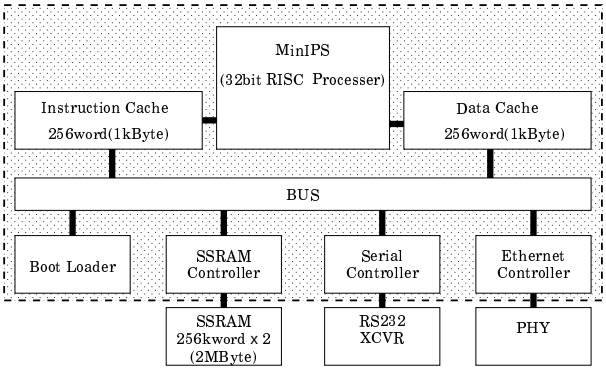

図1: システムの構成
(点線枠内はFPGA上に実装される部分)

図2にプロセッサ全体の構成を、表1に命令セットを示す。これらにより要件は満足される。

2.2.2 キャッシュメモリ

MinIPSはハーバードアーキテクチャであるが、メモリ空間を命令とデータで物理的に分けていないため、命令フェッチとデータアクセスは同時にできない。それを解消するために命令とデータの両方にキャッシュメモリを持たせる。キャッシュの記憶セルにはESBを使用する。

2.2.3 ブートローダ

MinIPSの起動時に動かすプログラムを格納するROM領域を、ESBを用いて実装する。このROMにはプログラムをロードするための最小限のプログラムをあらかじめ書き込んでおく。これにより外部からのプログラムロードが可能になる。

2.2.4 SSRAMコントローラ

通信プログラムを動作させるにはESBによるメモリだけでは容量が不足するため、主記憶はFPGAの外部に持つことにする。そのためにSOPCボードに実装されているメモリであるSSRAMを使う。

2.2.5 RS232Cコントローラ

外部からのプログラムロードおよびコンソールにはRS232Cを使う。RS232Cコントローラの仕様はシミュレータSPIM[6]に合わせてある。これによりシミュレーションから実機への円滑な移行が可能になる。通信フォーマットは9600bps、8ビット、パリティなし、1ストップビットとする。

2.2.6 Ethernetコントローラ

通信実験には、リンク層アドレスが存在し、一般的によく用いられるEthernetを採用する。MinIPSシステムのEthernetコントローラ[9]は、受信では、プリアンブルの除去、CRCの計算と除去を行なう。また、自分宛でないパケットの破棄や壊れたパケットの破棄も行なう。送信では、CRCの計算

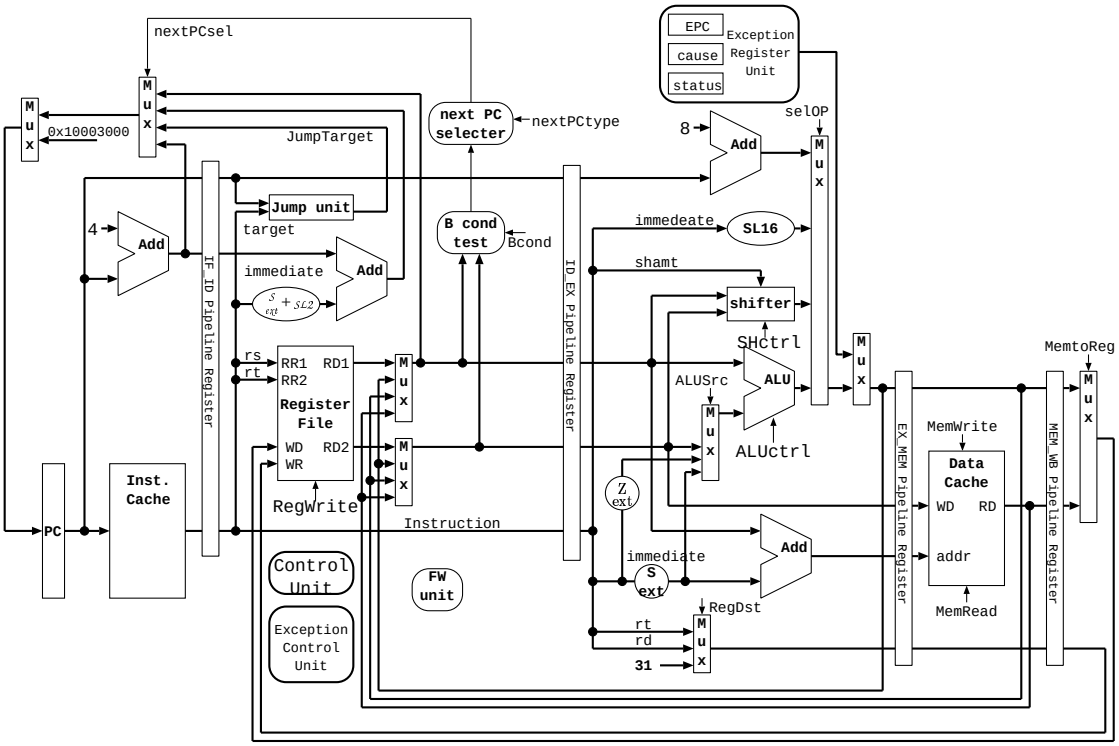

図 2: MinIPS の構成

表 1: MinIPS 命令セット

区分	命令	意味
算術演算	add	$1 = $2 + $3
	sub	$1 = $2 − $3
	addi	$1 = $2 + 100
	addu	$1 = $2 + $3
	subu	$1 = $2 − $3
	addiu	$1 = $2 + 100
論理演算	and	$1 = $2&$3
	or	$1 = $2 | $3
	andi	$1 = $2&100
	ori	$1 = $2 | 100
	sll	$1 = $2 << 10
	srl	$1 = $2 >> 10
	sra	$1 = $2 >> 10
	sllv	$1 = $2 << $3[4 : 0]
	srlv	$1 = $2 >> $3[4 : 0]
	srav	$1 = $2 >> $3[4 : 0]
データ転送	lb	$1=Memory[$2+100]
	lw	$1=Memory[$2+100]
	sb	Memory[$2+100]=$1
	sw	Memory[$2 + 100] = $1
	lui	$1 = 100 × 2^{16}
	mfe†	$1=$epc
	mte†	$status=$1
条件分岐	beq	if ($1 == $2)go to PC+4+400
	bne	if ($1! = $2)go to PC+4+400
	slt	if ($2 < $3) $1 = 1;else $1 = 0
	slti	if ($2 < 100) $1 = 1;else $1 = 0
	sltu	if ($2 < $3) $1 = 1;else $1 = 0
	sltiu	if ($2 < 100) $1 = 1;else $1 = 0
無条件ジャンプ	j	go to 40000
	jr	go to $31
	jal	$31 =PC+4; go to 40000
	rfe	$status[0]=0

† 割り込み処理用命令

と付加、プリアンブルの付加を行なう。MinIPS との接続はメモリ空間に配置されたレジスタを用いて行なう (図 3)。

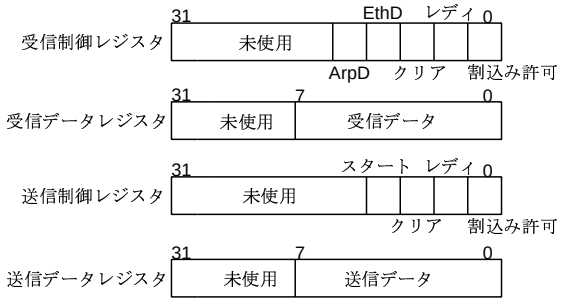

図 3: Ethernet コントロールレジスタ

2.3 実装結果

開発ツール QuartusII Ver.1.1 を用いて、SOPC ボード上の FPGA をプログラムした際の、デバイス使用率を表 2.3 に、コンパイルにかかった時間を表 2.3 に示す。ツールの実行環境は Pentium4 1.7GHz、RDRAM 512MB である。

実装されたプロセッサは 16.5MHz で動作している。

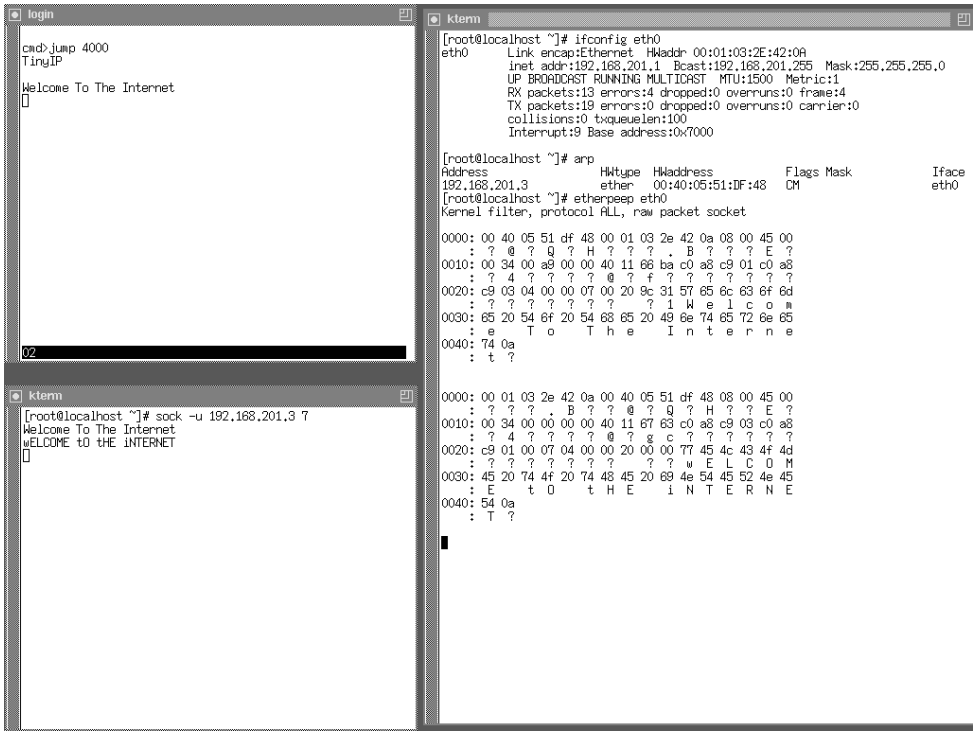

図 4: 実験時の画面

(左上:MinIPS システムのコンソール、左下:UnixPC 上での sock コマンドの実行結果、右側:Etherpeep による観測結果)

表 2: デバイス使用率

	使用量	/	容量	使用率
logic elements	9573	/	16640	57 %
pins	176	/	488	36 %
ESB bits	103936	/	212992	48 %

表 3: コンパイル時間

処理フェーズ	処理時間 (時:分:秒)
Database Builder	00:00:34
Logic Synthesizer	00:11:53
Fitter	00:12:45
Assembler	00:00:11
Delay Annotator	00:00:27
Total	00:25:52

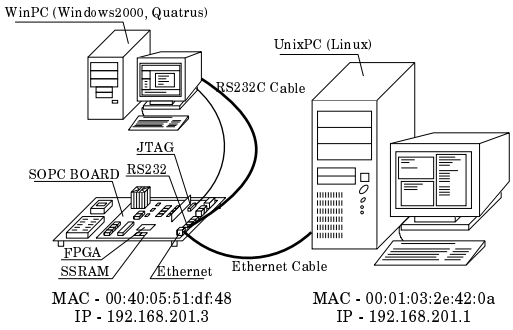

図 5: SOPC ボードとの接続

3 CNP 結合実験

3.1 実験内容

実験装置は図 5 の様に Linux のインストールされた PC(UnixPC)、Quaruts がインストールされた PC(WinPC)、および SOPC ボードからなる。UnixPC と SOPC ボードは、Ethernet クロスケーブルで接続されている。WinPC と SOPC ボードは、RS232C クロスケーブルと FPGA プログラミング用ケーブル (JTAG) で接続されている。

P 実験の学生は MinIPS システムを Verilog-HDL で記述し、Quartus でコンパイルし、得られた回路データを FPGA へ送り、プログラムする。これらの操作は WinPC 上で行なう。MinIPS を起動すると、ブートローダによりプログラムのロードが可能になる。そこで、コンソールを実現するモニタプログラムを書き込む。

N 実験の学生は UDP/IP を TinyIP として実現し、UDP 受信データを RS232C インターフェースを介して表示した後、大文字小文字変換を行い、返信するアプリケーションを作成する。このアプリ

ケーションは、C 実験の学生が TinyC[10] から発展させたコンパイラでコンパイルされる。コンパイルは UnixPC 上で行なう。作成したアプリケーションを MinIPS へロードし、実行する。

アプリケーション起動後、UnixPC 側から sock コマンド [11] を用いて、適当な文字列を含む UDP パケットを送信する。UDP 送受信の様子を、Ethernet フレームを表示するプログラム Etherpeep[5] を用いて観測する。

3.2 結果

実験時の画面を図 4[1]に示す。左上のウインドウは MinIPS システムのコンソールを表示したもの、左下のウインドウは UnixPC 上での sock コマンドの実行結果を表示したものである。また、右側のウインドウは Etherpeep による観測結果を表示したものである。観測結果から、UnixPC から送信されたデータが、MinIPS 上のアプリケーションにより大文字小文字変換されて送り返されている様子が分かる。

4 学生実験の実施

4.1 実験の構成と内容

実験全体は図 6 のように、C 実験、N 実験、P 実験の三つの実験に分かれる。これらの実験はチームを組み、役割分担を決め、それぞれ同時進行で行なう。最後にそれぞれの成果を結合して動作の確認を行なう。

実験は全部で 12 午後[2]とし、終わりの 2 午後を結合実験の時間にあてる。各実験は、はじめの 10 午後の間に完了する必要がある。

ここでは、P 実験についてのみ実施内容の詳細を記述する。

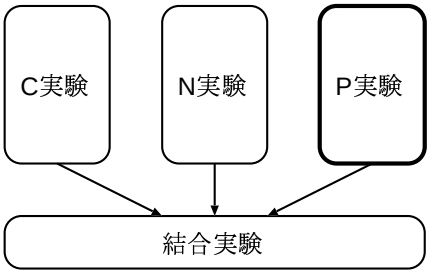

図 6: CNP 実験の進行

実験は次のような 5 段階のステップに分けて行なう。

1. 設計ツールの利用方法の学習
2. 簡単な部品の設計
3. 演算器の設計
4. パイプラインプロセッサの設計
5. MinIPS システム全体のコンパイルと動作確認

他班との兼ね合いもあるため、最低限の設計仕様は固定して学生に渡す。はじめに設計を行なうためのツールの使用方法を学ばせ、次に、マルチプレクサや加算器などの簡単な部品を設計させる。徐々に複雑な部品の設計を行なわせて必要な部品が揃ったところでプロセッサ全体の設計を行なわせる。プロセッサ全体に関しては、全てを記述させるのではなく、パイプラインプロセッサの様子が分かるような形で一部の記述を抜き取り、穴埋め形式で設計を行なわせる。RS232C コントローラや Ethernet コントローラ等のプロセッサ以外の記述は全て与えて、実際にプロセッサが動いているのが確認できるようにする。ここまでで P 実験が完了する。その後、C 実験の受講者が作成したコンパイラで N 実験の受講者の作成したアプリケーションをコンパイルし動作確認を行なう。

以上の実験を各実験 10 名、全体で 30 名を対象に実施した。

4.2 評価および考察

学生の進行状況によっては、早めにヒントを与えることによって進行速度の調整を行なった。それでも戸惑う学生は少なくなかった。しかし、プロセッサが動くところまでできた学生が 8 名、CNP の結合も 3 組が成功した。実験を完遂できなかった学生にも確かな手応えがあったように感じられた。実際、通信実験が成功するたびに周りからも喚声が上がり、その熱気や興奮がこちら側にも伝わって来た。

また、実験終了後に簡単なアンケートを実施した。公正を図るため無記名で行なった。なお、P 実験を行なった 10 名全員から回答を得た。P 実験に対する主な回答結果を図 7 に示す。

まず、学部 3 年生にとって本実験のような大規模な実験を行なうのは初めてであるため、難しいという印象を与えるのではないかと予測を立てていたが、ここは予測通り簡単であると答えた学生はいなかった。しかし、9 割の学生が難しいと回答しているにもかかわらず、同数の学生が実験の理解に至っている点は評価できる。また、P 実験は、大規模な回路の合成を行なうという性質上コンパイルに非常に時間がかかる。そのためほとんどの学生が 10 時間以上、長い学生は 15 時間以上の自習を行なって実験を進めていた。正規の実験時間が 36 時間であることを考えると非常に長い時間であると言える。それにもかかわらず、全員の学生から実験を楽しく行なうことができたという回答を得たことから、本実験が学生の意欲を引き出す魅力を持っていたと言える。

同時に次のような問題点が指摘された。設計規模が大きいことに加え、コンパイル時間が長く、ツー

[1] なおこのスクリーンダンプはアプリケーションのロード後に、RS232C を UnixPC につないで得られたものである
[2] 1 午後は 2 時限

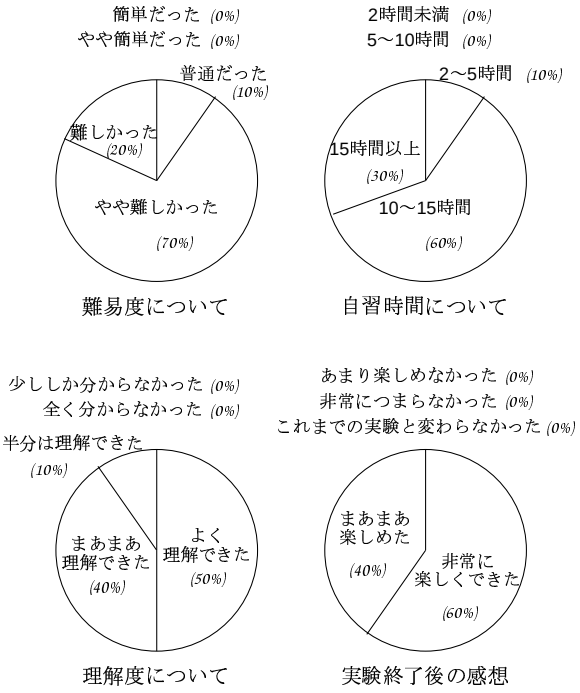

図 7: アンケート結果

ルのエラー解析が甘く、デバクも容易でない。そのためP実験の完成が遅れ、結果として結合実験にかけられる時間が短くなった。作成されたプロセッサのテスト方法が確立されていなかった。他の実験内容が理解できなかった。

これらの問題点に対して次の解決策が考えられる。

- 実行性能が向上したより新しい版の開発ツールを導入する。
- Quartusとは別の、エラー解析の厳しいVerilog処理系をエラーチェックに用いる。
- テストプログラム群を与える。
- 実験の早い段階から各実験の内容および関係について議論させる。

5 おわりに

本論文では、プロセッサ、コンパイラ、コンピュータネットワークという情報関連学科において重要な技術を横断的に扱うCNP実験の概要およびその成果をプロセッサ実験(P実験)を中心に述べた。長時間の自習を必要とする困難な実験にもかかわらず、学生から高い評価を得た。これは本実験の魅力と有効性を示すものである。

学部3年生の実験では、基礎的な課題を時間をかけてじっくり行なうことは必要である。加えて、CNP実験のような現代的かつ包括的な課題を扱うこともまた重要である。先を展望し視野を広げることは、卒業研究や大学院での研究を行なう際、課題を選び、問題点を探るヒントを与える。魅力的な課題は学生の能力を引き出す。また、協調作業の経験は、卒業後社会において仕事をする場合も役立つと思われる。

現状ではCNP間の連携はまだ十分とは言えず、連携をより密にすることは今後の課題である。またCNPに加えオペレーティングシステム関連の課題の導入やアプリケーションの充実も考えられるが、現状ではカリキュラム編成との関係上、実施できていない。

6 謝辞

実験全体にわたり様々な角度からコメントを頂いた河野健二助手、学生実験を行なう際に、実験設備の設置や設定等を行なって下さった奈良岡雅人技官、そして我々と共に未成熟な実験に挑戦してくれた学生諸君に感謝致します。

参考文献

[1] R.B.Brown, R.J.Lomax, G.Carichner, and A.J.Drake: A Microprocessor Design Project in an Introductory VLSI Course, IEEE Trans. Educ., Vol.43, No.3, pp.353-361,2000.

[2] D.Kassabian and A.Albicki: A Protocol Test System for the Study of Sliding Window Protocols on Networked UNIX Computers, IEEE Trans. Educ., Vol.38, No.4,pp.328-334,1995.

[3] 戸田賢二, 石棉陽一, 関山守, 高橋栄一: 組込応用向き小型プロセッサカードの予備的評価, 電子情報通信学会, 情処研報, 2001-ARC-143, pp.43-48, 2001.

[4] 本城和彦編: 一億超トランジスタ時代のシステムLSI, 電子情報通信学会誌, Vol.84, No.8, pp.551-598, 2001.

[5] K.Abe, T.Tateoka, M.Suzuki, K.Kono, and Y.Maeda: An Integrated Laboratory for Processor Organization, Compiler Design, and Computer Networking, to be submitted to IEEE Trans. Educ.

[6] David A.Patterson,John L.Hennessy: Computer Organization & Design: The Hardware/Software Interface, Second Edition, Morgan Kaufmann Publishers, Inc, 1998. (邦訳: 成田 光彰: コンピュータの構成と設計 第2版 (上・下巻), 日経BP社, 1999.)

[7] Altera Co.: System-on-a-Programmable-Chip Development Board User Guide, September 2001.

[8] 葛毅、 大菅大吉、 鶴田三敏、 阿部公輝: 32ビットRISCプロセッサMinIPSの設計と実装, 電気通信大学紀要, Vol.10, No.2, pp.71-78, 1997.

[9] 森田和夫、 阿部公輝: UDP/IPプロトコルのFPGAへの実装と性能評価, 情報処理学会第62回全国大会論文集, pp.5-157–5-158, 2001.

[10] 渡辺坦: コンパイラの仕組み, 朝倉書店, 1998.

[11] W.R.Stevens: TCP/IP Illustrated, Vol.1, Addison-Wesley, 1994.