FPGAを用いたマイクロプロセッサ開発実験

西村 克信 † 渡辺 晴美 ‡ 工藤 知宏 ‡ 天野 英晴 †

† 慶應義塾大学 理工学部
‡ 東京工科大学 工学部

概要

ハードウェア記述言語 (SFL) と書き換え可能なゲートアレイ (FPGA) を用いることにより、マイクロプロセッサの設計から動作確認までを一貫して行なう教育用のマイクロプロセッサ実験システムの構成と、これを実際に学生を対象に行なった実験の実施例について述べる。

本システムは、一度にある程度多くの学生を対象とした実験を行なうことを可能にするもので、比較的安価に実効性の高い実験を実現することができる。

1 はじめに

大学における計算機教育において、コンパイラ/アセンブラ等のソフトウェア、計算機アーキテクチャ、LSI 設計の一貫教育は、学生に広く実際的な知識と経験を与えると共に、創造力を刺激し、計算機に対する興味を高める点で非常に有効である [1]。

このような、一貫教育の鍵となるのは、設計対象となる教育用プロセッサの実現である。従来、教育用プロセッサは、完全なペーパマシンでシミュレータ上でのみ動作するものが多かったが、近年、設計用ツールの普及、書き換え可能なゲートアレイ (FPGA) の集積度の向上などにより、大学における実験教育環境が整備されると共に、実際に教育用プロセッサを製作する実験についての報告が活発になりつつある [2][3][4][5]。

今までに報告された試みの中で、九州大学における Kue-Chip[2], QP-DLX[3]、九州工業大学の Kite[4] は、命令セットが決まっている教育用プロセッサである。学生は教育の初期段階では、既に実装されたチップのモニタ機能を用いてその動作を解析し、命令実行時の CPU の動作についての理解を深める。さらに、コントローラの設計、パイプライン処理等の高速化、LSI チップへの実装を通じてアーキテクチャの様々なレベルの設計技術と LSI 設計実装技術を習得する。チップの開発は実際のゲートアレイを作成することが理想であるが、FPGA を用いても、動作の確認、プログラムの実行が可能である。同時にアセンブラ/コンパイラの作成を行ない、ソフトウェアレベルの設計技術を習得する。

一方、東京大学理学部情報科学科では、完全に命令セット、データパスを完全に学生に対してオープンにして、CPU を製作する実験を試みている。学生は自分で考えた命令セット、データパスを持つオリジナルの CPU を設計し、FPGA 上に実現すると共に、この命令セットに対するアセンブラ/コンパイラの作成をプログラミング実験で行なう。最終的には、レイトレーシングのプログラムを実行し、その動作速度を測定しながら、設計におけるトレードオフに関して体験する [5]。

前者の方法は、初期段階での教育を効果的に行なうことができると共に、テキスト等の作成が容易で、ある程度大人数を対象とすることが可能だが、教育のかなり先の段階にならないと、学生の創意をする部分が出てこない傾向にある。一方、後者の方法はほぼ計算機教育の理想に近いが、全体として多大な時間を要し、多人数を対象とすることが難しいことから多くの私立大学では導入が困難である。

我々は、両者の中間的な方法、すなわち、データパスの概略のみを指定して、命令セットはオープンにした教育用プロセッサによる実験を行なっている。本報告は、今年度これまでに行なった実施例をもとに、ハードウェア記述言語 SFL[6][7][8] による教育用プロセッサ記述の利点および欠点、また今後の課題等について報告する。

2　開発実験の枠組

マイクロプロセッサ開発実験は、実験のための費用、実験時間、学生の学習レベルにおいて様々な導入形態を取ることができるが、共通して以下のステップをとる。

- ハードウェア記述言語による記述を最初に教え、データパスについて記述されたファイルを学生にまず配布し、学生はこれをベースとして設計を行なう。もちろん意欲のある学生がデータパスを変更することは自由である。現在、ハードウェア記述言語には NTT が開発した論理合成システム PARTHENON の記述言語である SFL を用いている。論理設計の実験と組み合わせる場合、ゲートレベルでの設計を行なわせることも可能だが、ハードウェア記述言語を用いた方が、ハードウェア記述言語を全く知らない学生を対象とした場合でも、はるかに少ない実験時間でマイクロプロセッサを作成することができる。

- 命令セットの構成は学生に任せるが、データパスが基本的に決まっているため、ある程度の制限を受ける。基本的には Load/Store 型の RISC アーキテクチャになる。実験時間、学生の学習レベルによっては、命令のコーディングの枠組を与え、代表的な命令の記述を例として与えることもできる。

- ハードウェア記述レベルでのシミュレーションをきちんと行ない、この時点で教員または T.A. の前で動作の確認を行なう。

- 論理合成した回路を FPGA 上で実現し、簡単なプログラムをハンドアセンブルし、実行する。

このマイクロプロセッサ開発実験の流れを図 1 に示す。

実験で作成するマイクロプロセッサの規模は純粋に実験の予算によって決まり、学生のレベルや実験時間の影響を受けない。これに対し学生の基礎知識が初期の段階であったり、実験時間が少ない場合は、命令セットのコーディング例を与えたり、いくつかの命令の記述を例として与えることによって実験時間を短縮することができる。

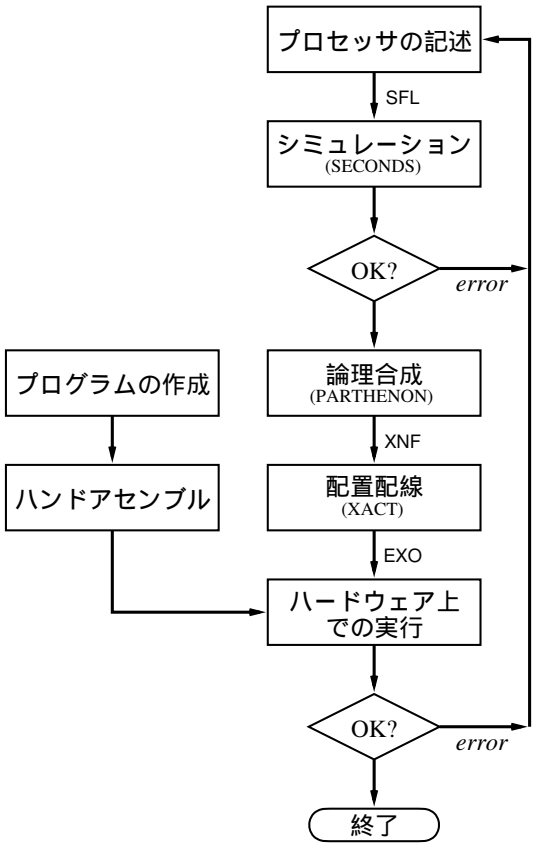

図 1: 実験の流れ

昨年度、慶應義塾大学において FPGA に Xilinx 社の XC3000 シリーズを用いて 8bit のプロセッサを作成する実験を行なった。また今年度からは同 XC4000 シリーズ [9] を用いて 16bit のプロセッサの作成実験を慶應義塾大学と東京工科大学で行なっている。なお、アセンブラ/コンパイラについては別の実験が用意されている。

3　マイクロプロセッサ PICO

実験用マイクロプロセッサ PICO は bit scalable — つまり、同様の考え方で 4bit から 16bit まで、様々なサイズを取ることのできるマイクロプロセッサのフレームワークで、FPGA のサイズに応じて命令セットを構築することができる。

実験教育用であるため、命令セットは様々なバラエティを許すが、基本方針は以下の通りである。

- RISC である。すなわち単一命令長、Load/Store アーキテクチャである。
- レジスタ数は可能な限り多く取る。
- アドレッシングモードはレジスタ間接を基本とする。学生の興味により、ディスプレースメント付き、ポストインクリメント/プリデクリメント、インデックス修飾などを付けくわえることができる。
- データパスは予め決められたものを用いる。

8bit 未満の構成 (PICO-4/PICO-5) は、FPGA の規模が不足する時にやむを得ず用いるものであり、基本的に機械語を直接メモリに入れて動かしてみるオモチャ程度である。8bit 構成 (PICO-8) も簡単ではあるが、アセンブラを作って多少のプログラムを動かすことができる。コンパイラ、アプリケーションを結び付けた一貫教育には 16bit 以上の構成 (PICO-16) が必要である。

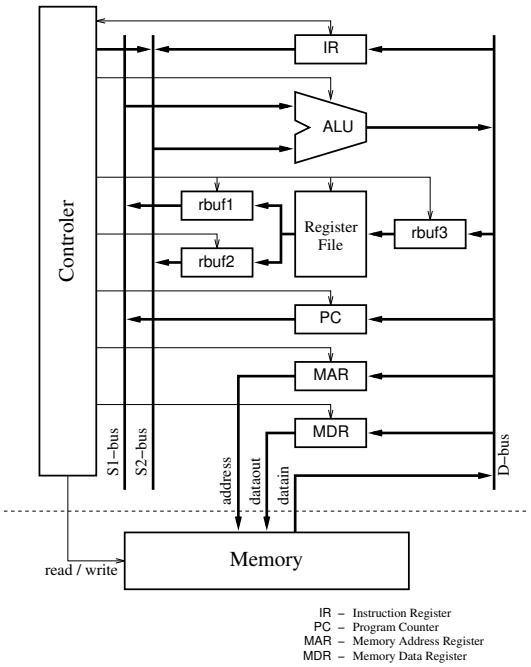

図 2: データパス

3.1 PICO-16 のデータパス

図 2 に PICO-16 のデータパスを示す。データパス上における基本的な操作は、レジスタファイルからのオペランド読みだし、ALU での演算実行、そしてレジスタファイルへの結果の書き込みである。

レジスタファイルは 8words×16bits の RAM と、出力バッファ用のレジスタ (reg1、reg2) と入力バッファ用レジスタ (reg3) から構成される。これ以外に、プログラム・カウンタ (PC)、命令レジスタ (IR)、および外部メモリをアクセスするためのメモリ・アドレス・レジスタ (MAR)、メモリ・データ・レジスタ (MDR) と呼ばれるレジスタが存在する。これらは、内部バスを介してつながっており、それぞれ S1-Bus、S2-Bus、D-Bus と呼ばれる。ソース・バス (S1-Bus、S2-Bus) からデスティネーション・バス (D-Bus) への唯一の経路は ALU 経由である。

3.2 PICO-16 の命令セット例

学生の学習段階が浅い場合、または実験時間が不足する場合、ある程度の命令の枠組を与える必要がある。この場合、以下のような指針を与える。

- 命令は、イミーディエイトをとる命令群とそれ以外から構成される。
- 命令長の制約によりオペランドは 2 つまで。
- イミーディエイトを取る命令は、命令長の半分をイミーディエイト領域に用いる。

このような枠組を与えた場合、命令セットは DLX アーキテクチャに近くなり、Hennessy & Patterson[10] を授業のテキストに用いている場合は、これと結び付けて理解が容易になる。

以下、教育実験用に設計した PICO-16 の命令セットを示す。場合によっては命令形式の概略を示したり、命令の記述例をいくつか示すことにより、学生にとっかかりを与えることができる。

3.2.1 命令形式

すべての命令は 16 ビット長で、5 ビットの命令操作コードを有している。命令形式は I(*Immediate*) 型命令, R(*Register–register*) 型命令, J(*Jump*) 型命令の 3 種類である。これを図 3 に示す。

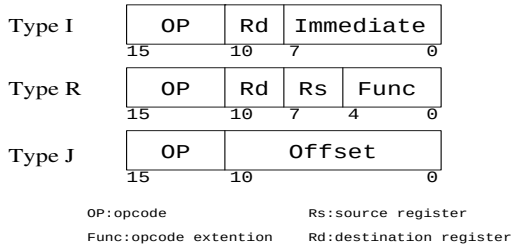

図 3: 命令形式

I 型命令はロード /ストア、即値演算などに使用する。R 型命令は算術論理演算などに使用し、命令操作コードの他にファンクションコードを有している。J 型命令は分岐 (ジャンプ) 命令および条件分岐 (ブランチ) 命令に使用する。命令長の制限によりレジスタ数は R0 から R7 までの 8 つとなる。2 オペランド命令を用いていることを除き、ほぼ DLX に準じた命令セットとなっている。

PICO-16 の命令セットにはロード /ストア、算術論理演算、比較、制御の 4 つのタイプがある。

3.2.2 ロード /ストア命令

LDL(Load Low Immediate) 命令はレジスタの下位半分に値を代入し、上位半分に 0 を設定する。また、LDH(Load High Immediate) 命令はレジスタの上位半分に値を代入する。これにより 16 ビット定数の設定は 2 命令で行える。

命令	形式	意味
LDL	Rd, #imm	Rd ← 0^8 ## $imm_{7..0}$;
LDH	Rd, #imm	Rd ← $imm_{7..0}$ ## $Rd_{7..0}$;
LD	Rd, (Rs)	Rd ← M[Rs];
ST	(Rd), Rs	M[Rd] ← Rs;
MOV	Rd, Rs	Rd ← Rs;

表 1: ロード /ストア命令

3.2.3 算術論理演算命令

すべての算術論理演算はレジスタ–レジスタ間およびレジスタ–即値間のみ可能である (ロード /ストア・アーキテクチャ)。命令には加算、減算、論理積、論理和、排他的論理和、シフトがある。即値形式では 16 ビットに 0 拡張された即値が利用できる。

命令	形式	意味
ADDI	Rd, #imm	Rd ← Rd + (0^8 ## $imm_{7..0}$);
SUBI	Rd, #imm	Rd ← Rd - (0^8 ## $imm_{7..0}$);
ANDI	Rd, #imm	Rd ← Rd & (0^8 ## $imm_{7..0}$);
ORI	Rd, #imm	Rd ← Rd | (0^8 ## $imm_{7..0}$);
XORI	Rd, #imm	Rd ← Rd ^ (0^8 ## $imm_{7..0}$);
SLLI	Rd, #imm	Rd ← $Rd_{15-imm..0}$ ## 0^{imm};
SRLI	Rd, #imm	Rd ← 0^{imm} ## $Rd_{15..imm}$;
SRAI	Rd, #imm	Rd ← $(Rd_{15})^{imm}$ ## $Rd_{15..imm}$;
ROLI	Rd, #imm	Rd ← $Rd_{15-imm..0}$ ## $Rd_{15..16-imm}$;
RORI	Rd, #imm	Rd ← $Rd_{imm-1..0}$ ## $Rd_{15..imm}$;
ADD	Rd, Rs	Rd ← Rd + Rs;
SUB	Rd, Rs	Rd ← Rd - Rs;
AND	Rd, Rs	Rd ← Rd & Rs;
OR	Rd, Rs	Rd ← Rd | Rs;
XOR	Rd, Rs	Rd ← Rd ^ Rs;
SLL	Rd, Rs	Rd ← $Rd_{15-Rs..0}$ ## 0^{Rs};
SRL	Rd, Rs	Rd ← 0^{Rs} ## $Rd_{15..Rs}$;
SRA	Rd, Rs	Rd ← $(Rd_{15})^{Rs}$ ## $Rd_{15..Rs}$;
ROL	Rd, Rs	Rd ← $Rd_{15-Rs..0}$ ## $Rd_{15..16-Rs}$;
ROR	Rd, Rs	Rd ← $Rd_{Rs-1..0}$ ## $Rd_{15..Rs}$;

表 2: 算術論理演算

3.2.4 比較命令

比較命令は、2 つのレジスタの比較 (==, !=, >, <, >=, <=) を行う。比較結果が偽 (false) のときデスティネーション・レジスタ (Rd) には 0 が設定され、真 (true) のとき 0 以外の値が設定される。

命令	形式	意味
SEQ	Rd, Rs	if (Rd == Rs) Rd ← 1^{16}; else Rd ← 0^{16};
SNE	Rd, Rs	if (Rd != Rs) Rd ← 1^{16}; else Rd ← 0^{16};
SGT	Rd, Rs	if (Rd > Rs) Rd ← 1^{16}; else Rd ← 0^{16};
SLT	Rd, Rs	if (Rd < Rs) Rd ← 1^{16}; else Rd ← 0^{16};
SGE	Rd, Rs	if (Rd >= Rs) Rd ← 1^{16}; else Rd ← 0^{16};
SLE	Rd, Rs	if (Rd <= Rs) Rd ← 1^{16}; else Rd ← 0^{16};

表 3: 比較命令

3.2.5 制御命令

プログラムの制御はジャンプ命令とブランチ命令でなされる。ジャンプ命令はデスティネーション・アドレスの指定法とリンクの有無の違いによって 4 つに分けられている。

- J、JAL 命令は符号付き 11 ビット・ディスプレースメントがプログラムカウンタに加算される。
- JR、JALR 命令はジャンプ先のアドレスが格納されているレジスタを指定する。
- JAL、JALR 命令はリターンアドレスを R7 に格納する (手続き呼び出しで使う)。
- BEQZ、BNEZ 命令は R7 と 0 の比較結果により、符号付き 11 ビット・ディスプレースメントがプログラムカウンタに加算される。

命令	形式	意味
J	disp	PC ← PC + $((\mathtt{disp}_{10})^5$ ## $\mathtt{disp}_{10..0})$;
JR	Rd	PC ← PC + Rd;
JAL	disp	R7 ← PC + 2;
		PC ← PC + $((\mathtt{disp}_{10})^5$ ## $\mathtt{disp}_{10..0})$;
JALR	Rd	R7 ← PC + 2;
		PC ← PC + Rd;
BEQZ	disp	if (R7 == 0)
		PC ← PC + $((\mathtt{disp}_{10})^5$ ## $\mathtt{disp}_{10..0})$;
BNEZ	disp	if (R7 != 0)
		PC ← PC + $((\mathtt{disp}_{10})^5$ ## $\mathtt{disp}_{10..0})$;

表 4: 制御命令

4 実験用ボード

マイクロプロセッサ開発実験を限られた時間内に行なう場合に最も問題になるのは、FPGA のコンフィグレーションデータを作成後、実際にデータを FPGA に転送して動作を確認する作業である。学生自身が、FPGA および周辺回路をユニバーサル基板上に半田付けおよびラッピングを用いて実現するのが理想であるが、このような方法では、マイクロプロセッサの実装のみならず、外部メモリ、入出力装置などの周辺回路や、動作確認のためのモニタ機能の実装に多大な時間を要する。また、不用意な電源配線により高価な FPGA を破壊する可能性もある。

そこで、これらの実装を簡単化するための実験用ボードを作成した。図 4 に示すように、このボードはマイクロプロセッサを実装する FPGA である XC4010、プログラム格納用のスタティック RAM(64kbyte)、入力用スイッチ、内部データパス/プログラムカウンタのモニタ用 LED が備えられている。柔軟性を増すため、FPGA のピンと周辺回路の間にはラッピングポストが設けられている。これは、回路規模によってはピン配置を指定すると FPGA 上に実装できない可能性があるためで、この場合学生はそれぞれが設計したマイクロプロセッサのピン配置、モニタしたい出力に合わせてラッピングを行なう。

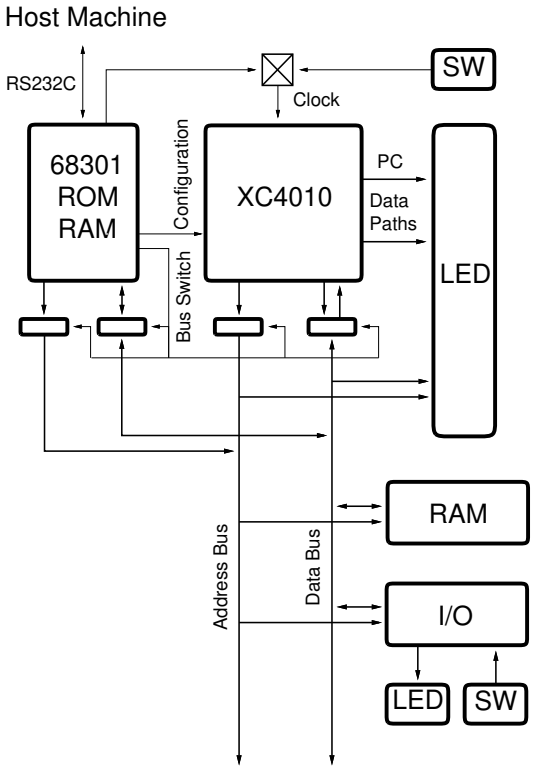

図 4: プロセッサボード

FPGA のコンフィグレーションデータと、PICO 自体のプログラムは、実験開始時に転送する必要がある。これらのデータとプログラムをホストマシンから直接転送するため、ボードには RS232C インタフェースと、これを制御する 16bit CPU 68301 が搭載されている。この CPU は簡単なモニタプログラムの制御下で動作し、FPGA の動作後、RAM の内容を確認することなどができる。

また、FPGA へのクロックはスイッチを用いて手動で与えることもできるし、68301 のタイマ出力を用いて任意の周波数の信号を与えることもできる。

5　開発実験の実施例

今年度、慶應義塾大学および東京工科大学において、マイクロプロセッサの開発実験を行なっている。以下にその概要と、現在までに判明している問題点をあげる。

5.1　開発実験

実験は、慶應義塾大学では理工学部電気工学科 3 年生を対象に 6 時間 ×3 週の計 18 時間、東京工科大学では工学部情報工学科 3 年生を対象に 4 時間 ×6 週の計 24 時間かけて行なっている。

論理合成には NTT 社の PARTHENON 2.2.0 を、配置配線には Xilinx 社の XACT 5.0.0 を、また PARTHENON の出力形式である NLD 形式ファイルから、XACT の入力形式である XNF ファイルの変換に NTT 伝送システム研究所の nld_xnf 2.1b を使用している。さらに、FPGA に XC4010 を使用した前述の実験用ボードを用いて、開発したマイクロプロセッサの動作確認を行なっている。

開発実験のステップは以下の通りである。

STEP1　ハードウェア記述言語 SFL の記述法を習得する。サイコロカウンタなどの簡単な順序回路を記述し、動作シミュレータ SECONDS で動作を確認する。さらに論理合成を行ない合成された回路をプリントして検討する。このステップでは班を組まず、各員がそれぞれの記述を行ない、動作を確認する。

STEP2　対象とするマイクロプロセッサ PICO の概略について説明を受け、班毎に命令セットを決め、分担して記述を行なう。時間的制約から、基本命令セットのコーディング例を与え、実装すべき命令も代表的な数種に絞った。ただし、意欲のある班は、FPGA のサイズが許す限りさまざまな命令を実装することができる。SECONDS 上で全ての命令の動作を確認した段階で、このステップを終了する。

STEP3　論理合成、ネットリスト変換、Xilinx 社の CAD による配置配線を行ない、コンフィグレーションデータを得る。前述の実験用ボードに転送して、動作確認を行なう。可能ならば、配置配線ツールによるコンフィグレーションデータへの変換は実験 T.A. が行なっておくのが良い。

5.2　問題点

実験の経験により以下の問題点が明らかになった。

- 論理合成、配置配線が学生にとってはほとんどブラックボックスであり、またかなりの時間を要するため、退屈してしまう学生が多かった。ちなみに、最小セットの PICO-16 では配置配線に約 20 分 (SS10/SunOS5.3) で終了したが、フルセットの PICO-16 だと配置配線に 3 時間以上かかった。
- 一度に多数の命令を実装し、配置配線の段階で入りきらなくなってしまう班がでた。このため、せっかく記述した命令を削除することになり、学生のやる気を大いにそいでしまった。
- SFL の記述は、バス構造が明示的でない。このため、学生は命令を付け加えていく際に、直接レジスタに代入する記述をどんどん増やしてしまう。このような記述は、論理合成で多くのマルチプレクサを発生させることになり、比較的単純なものでもチップに入り切らなくなってしまう。
- 必要ゲート数を減らすために FPGA のライブラリを用いようと試みたが、配置配線ツールの都合で実現できなかった。ただし RAM のみは可能であったため、PARTHENON 用のライブラリファイル (NLD 形式、PCD 形式) を記述し使用した。また、RAM に対する書き込みタイミングを満足するのに複数のステートが必要であった。しかし、PICO の限られたステート内でこれを実現するのは難しかった。そこで、ライブラリ中で同期回路用のクロック (`m_clock`) を用いて 1 ステートで完了するように記述した。

6　おわりに

報告した実験システムの特長のひとつは、安価に導入できる点にある。実験用ボード自体は、FPGA の価格を含め 20 万円前後で実現できる。さらに大学では、NTT 社より PARTHENON および変換ツールの

無償貸与を受けることができ、ランニングコストはほとんどかからない。また、ある程度多くの人数を対象とすることができることから、多くの私立大学での導入が可能であると思われる。

今後、さらに改良を重ね、プロセッサの記述に幅を持たせると共に、アセンブラ、コンパイラ、チップ設計などの他の教育と連係したものに発展させていきたい。

謝辞

PARTHENON を御提供いただいた NTT 社および LCA ツールへの変換ツールを配布してくださった同社伝送システム研究所の筒井章博氏に感謝いたします。

また、学生実験の T.A. を熱心に努めていただいた慶應義塾大学理工学部の西 宏章氏、山本 純氏、黒澤 飛斗矢氏ならびに東京工科大学の後藤 繁生氏に感謝いたします。最後に、熱心に実験を行なってくれた慶應義塾大学理工学部電気工学科ならびに東京工科大学工学部情報工学科の学生諸君に感謝します。

参考文献

[1] 柴山 潔、新實 治男, “大学における計算機アーキテクチャの教育方法に関する考察”, 情報処理学会計算機アーキテクチャ研報 100-4, Jun. 1993.

[2] 越智、澤田、岡田、上嶋、神原、濱口、安浦, ” 計算機工学・集積回路工学教育用マイクロプロセッサ KUE-CHIP2”, 信学報 CPSY92-47, Oct. 1992.

[3] 岩井沢 瑞穂 他, “計算機工学一貫教育用マイクロプロセッサ QP-DLX の開発”, 情報処理学会計算機アーキテクチャ研報 100-5, Jun. 1993.

[4] 田中、小羽田、久我、末吉, “教育用マイクロプロセッサ KITE とその開発支援環境”, 情報処理学会計算機アーキテクチャ研報 100-8, Jun. 1993.

[5] 松本尚, “プロセッサ作成学生実験-チップ、ボードからコンパイラ、アプリケーションまで”, マイクロエレクトロニクス研究開発機構 第 12 回ワークショッププロシーディング pp.1-12, Dec. 1993.

[6] 中村行宏、小野定康, “ULSI の効果的な設計法”, オーム社, 1994.

[7] NTT データ通信株式会社, “PARTHENON Reference Manual”, NTT データ通信株式会社, 1989-1990.

[8] NTT データ通信株式会社, “PARTHENON User’s Manual”, NTT データ通信株式会社, 1989-1990.

[9] Xlinx Inc., “The XC4000 Data Book”, Xlinix Inc., 1991.

[10] J.L.Hennessy, D.A.Patterson, “Computer Architecutre: A Quantitative Approach”, Morgan Kaufmann Pub. Inc., 1990.