計算機アーキテクチャ教育とその支援環境

井上 訓行†

集積回路技術の発展に対応したアーキテクチャ設計実習とCADツールを用いたLSI設計教育が重要になってきている．大学等の情報工学教育においてもこれらのための実習環境を構築する必要がある．本稿では計算機アーキテクチャの設計実習を目的として，アーキテクチャの決定からLSI化まで研究室内で実習できる環境を構築する際の問題点を検討し，一解決法としてプロセッサチップにFPGA (Field Programmable Gate Array) を利用する方法を提案し，マイクロプログラム制御方式のプロセッサを対象とした教育システムを構築している．構築されたシステムではRAM構造のLSI (FPGAとRAM) のみからなるボード上にアーキテクチャを実現する方法が採用され，その特徴は設計結果を実装時間を必要とせずただちにLSI化し，検証し，必要なら何回でも設計し直せることである．また，ハードウェアデバッグはパーソナルコンピュータの全面的な支援の下に行われ，記号形式で記述されたマイクロ命令によってレジスタトランスファーレベルのデバッグが可能になっている．マイクロプログラミングとデバッグのために機械独立型のマイクロアセンブラが採用されている．約3年の間に数種類の異なるアーキテクチャのプロセッサが本システムを利用して開発され，教育上有効なシステムであることが確認されている．

An Education Environment for Teaching Computer Architecture

Noriyuki Inoue†

Experimental design courses for computer architecture and LSI design have become important in the engineering senior education of university. These courses are realizable by means of ASICs and CAD. In this paper, a newly developed environment is presented, where students are able to learn how to design, implement, and verify microprocessors by themselves. A printed circuits board and a personal computer (PC) are utilized as a unified system. The board consists of a field programmable gate array based on static RAM technology, as a processor, and RAMs for both control and main memories. The designed processor is implemented on the board without wiring and verified by using debugging tools on the PC. The PC supplies microinstructions and clock pulses, and then displays on its screen the results of individual register transfer operations. For this purpose, a machine-independent microassembler and disassembler are implemented on the PC.

1. はじめに

集積回路技術の進展により大規模で高機能のハードウェアを容易に利用できるようになっている．一方，その内部構造は複雑になり理解が困難になってきている．ソフトウェア教育においてはワークステーション等の高機能のハードウェアが有効に活用されており，ハードウェア教育においてもこれらに対応したアーキテクチャ設計とLSI設計の実習教育が必要になってきている．

しかし，現状では市販のマイクロプロセッサがアーキテクチャとインタフェースの教育に利用され[1]，アーキテクチャの設計実習教育はほとんど行われていない．また，論理回路の実験ではゲートレベルのICを用いて簡単な回路の設計製作が行われている場合が多く，この方法ではマイクロプロセッサ程度の規模の回路を実装することは困難であり，LSI設計実習もできない．集積度の高いASICとCADシステムを利用することによって大学等の実習教育においてもマイクロプロセッサの開発が可能になってきているので，これらを利用してアーキテクチャ実習教育環境を構築する必要がある．

アーキテクチャ教育にはアーキテクチャの設計から実装までが必要であるという立場から，本稿ではまずアーキテクチャ実習環境について考察し，つぎに実際に教育システムを構築している．このシステムはマイクロプログラム制御方式のプロセッサを開発対象として，すべてRAM構造のLSIからなるボードの中にアーキテクチャを実現する方法を採用している．CADシステムを用いてパーソナルコンピュータ

† 京都産業大学工学部
Faculty of Engineering, Kyoto Sangyo University

(PC) またはワークステーション (WS) 上に設計された設計データとマイクロプログラムをこのボードにダウンロードしてプロセッサを実現する．PC 上のマイクロアセンブラを中心としたデバッグ支援システムが構築されており，記号形式で記述されたマイクロプログラムによってレジスタトランスファー (RT) レベルのデバッグが行われる．

2. アーキテクチャ実習教育環境

アーキテクチャの設計から実装までの統合した教育の実習環境をアーキテクチャ設計，論理設計，実装設計の3段階に分けて考察する．

2.1 アーキテクチャ設計

アーキテクチャ教育の実習方法には

① 既存のマイクロプロセッサを利用する方法

② 独自のプロセッサを開発する方法

がある．①の方法では既存のプロセッサは確実に動作する利点があり，多くの大学で市販のマイクロプロセッサがワンボードマイコンの形で使われている．しかしこれらは内部が複雑でわかりにくい，動作を外部から観測できない，公開されていない部分があるなどの欠点がある．

このため外部からの観測性に優れた簡単なアーキテクチャの教育用マイクロプロセッサチップが，ASIC や FPGA (Field Programmable Gate Array) を利用して開発されている[4),5)]．しかし，いずれの場合も学習者はアーキテクチャの設計と実現には関与できないので，アーキテクチャ教育としては不十分であると考えられる．

②の方法では学習者が自分で設計したアーキテクチャをハードウェアで実現する．教育的には自由に設計したアーキテクチャを実現する方法が理想的であるが，実習環境のための費用と実習時間の点から何らかの制約が必要になる．制約としては次の3つが考えられる．

（a）指導者がアーキテクチャを提示し，その全部または一部を実装する．

（b）アーキテクチャの一部（命令セットなど）を固定し，その他の部分を自由に設計して実現する．

（c）実装方法と実装規模を制限し，その範囲内で学習者が自由にアーキテクチャを決めて実現する．

（a），（b）の方法は一部で実施が検討されているが[4),5)]，アーキテクチャの設計という点からこの2つの方法は制約がきつすぎると考えられる．

2.2 論理設計

アーキテクチャを実現するためのブロック図やタイミング図を作成し論理回路に変換する過程である．この段階では CAD システムの利用が最大の焦点となる．CAD システムを全く利用しない方法から全面的に採用する方法までいろいろな利用形態が考えられる．産業界でのハードウェア製造工程では CAD システムの利用が必須になっている現状から，大学においても可能な限り CAD システムを導入するべきであると考えられる．また，実装段階でプログラマブル素子を利用したり，LSI を試作する場合には CAD システムは不可欠である．CAD システムの選択には①記述レベル，②論理最適化能力，③処理時間，④学習に要する時間等を考慮しなければならない．

2.3 実装設計

集積度の高い素子が利用可能になり，実装工程より設計過程が相対的に重要になってきている．同時に LSI 設計教育の重要性が高まっている．この観点から次の3つの方法について検討する．

① ゲートレベルの IC によるプリント基板への実装

② プログラマブル素子の利用

③ LSI チップの試作

①の方法は電子回路も含めた製作実験という観点では優れているが，アーキテクチャ教育ができる規模のプロセッサを実装するには時間と労力がかかりすぎる．②の方法は①と③の中間的な方法で，チップの中をプログラムすることによって論理回路部分のみを LSI 化する．部分的ではあるが LSI 設計教育が可能である．プログラマブル素子は設計に自由度があり，適切なチップを選択すれば実験室内で容易に繰り返し設計を実現できる．しかし，規模の点で制限を受けるため2ビットプロセッサの製作実験等が行われている[2)]．③の方法は LSI 設計教育の観点から理想的であるが製作に要する時間と費用の点から，各学習者のチップを作ることは困難である．

以上の検討の結果，大規模なプログラマブル素子である FPGA の利用が最も適切であると考えられる．

3. 実習システムのアーキテクチャ

前章の考察からアーキテクチャ実習教育の目標を次のように定める．

[目標] 学習者が自由に設計したアーキテクチャをハードウェアで実現する

LSI 教育の必要性と時間的，予算的制約から次の方法をとる．

【方法】 CAD システムを利用し，FPGA 上にアーキテクチャを実現する．

以下実習システムの設計と実現における3つの重要な点について検討する．

3.1 プロセッサチップの選択

教育用に利用する場合，学習者による設計には設計ミスが多く，また，設計の途中で部分的に実装して動作を確認する必要も生じるので，利用する FPGA は

① 設計を実験室内でただちに書き込めること

② 何回でも書き換えられること

が必要である．

現在マイクロプロセッサを1チップ化できる程度の FPGA は数種類入手可能である[3]．これらの中で①，②を満たすので，XILINX 社の LCA (Logic Cell Array) が最も適切であると考えられる．LCA は組合せ回路とフリップフロップからなる論理ブロックがアレイ状に配置され，その間に配線領域がとられている[10]．論理ブロックと配線領域は RAM 構造になっているので，構成 (Configuration) データを外部からダウンロードするだけでただちに設計をプログラムでき，書き換え回数に制限はない．

3.2 制御方式の検討

計算機の制御方式には大きく分けて結線（ワイヤードロジック）制御とマイクロプログラム制御がある．アーキテクチャ教育の観点からマイクロプログラム制御方式には次の利点がある．

① マイクロプログラミング自体がアーキテクチャ教育において重要な位置をしめる．

② 命令セットの設計に自由度が高く，単純なハードウェアでもいろいろな命令セットを実現できる[6),7)]．特にスタック機構や割込み制御も複雑な制御回路を組み込まずに実現できる．

③ 結線制御に比べ設計が容易である．特に設計変更に柔軟に対応できる[6),7)]．

④ RT レベルのデバッグが容易である．

⑤ マイクロアセンブラなどの汎用の開発支援ツールを用意できる[8)]．

⑥ プロセッサの設計が時間的な理由などでできないときでも，マイクロプログラミングの実習ができる．

これらの理由から，命令セットと制御回路の複雑さの関係を評価をできないなどの欠点はあるが，マイクロプログラム制御方式が適切である．

一般に制御メモリには ROM が用いられるが，教育用システムではデバッグ時にマイクロプログラムを頻繁に書き換える必要があるので RAM の方が適当である．この場合，RAM にマイクロプログラムを書き込む支援環境が必要である．

3.3 デバッグ環境

教育用システムではハードウェアデバッグが特に重要であり，デバッグ作業によって内部動作の理解が深まる．ハードウェアのデバッグにはデータ，条件等の設定と結果の観測が必要であるが，LSI 化した場合必要な観測点に直接触れることができないことが多いので，内部動作の確認が難しくなる．教育用マイクロプロセッサでは，観測バスを設ける方法や[4)]，必要な観測点をすべて外部端子に出す方法[5)] がとられている．しかしこれらの方法では

① 余分の配線領域や外部端子が多量に必要になる

② 学習者が設計する場合設計上の負担になる

などの問題がある．本稿ではマイクロプログラム制御方式を利用して，チップ内に余分なハードウェアを追加せずに，マイクロ命令で指定可能なすべての RT レベルの動作確認ができる方法を採用する．

実際の設定，観測には

① スイッチ，表示灯を用いる方法

② PC を利用する方法

がある．①の方法は直接操作，観測ができる利点がある．しかし，マイクロプロセッサのように設定，表示点数が多いと著しく操作しにくくなる．②の方法はマイクロアセンブラなどの PC 上のツールを利用して，操作を容易にすることが可能である．しかし，観測が間接的になり動作確認の実感を伴わない場合がある．

①②の両方を備え状況に応じて使い分けのできるシステムが必要である．

4. システム構成

本章ではマイクロプログラム制御方式の計算機を設計対象として開発したシステムの構成とデバッグ環境について述べる．

4.1 ハードウェア構成

本システムは学習者が設計した計算機を実現するボード（以下 MD (Microcomputer Development) ボードと呼ぶ），デバッグボード，MD ボードを支援する PC から構成されている．

MD ボードはプロセッサ用チップ (LCA) と外付

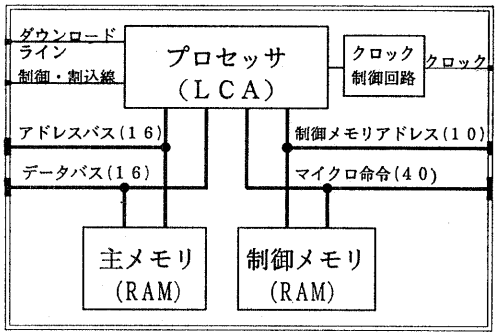


図 1 MD ボード
Fig. 1 Microcomputer development board.

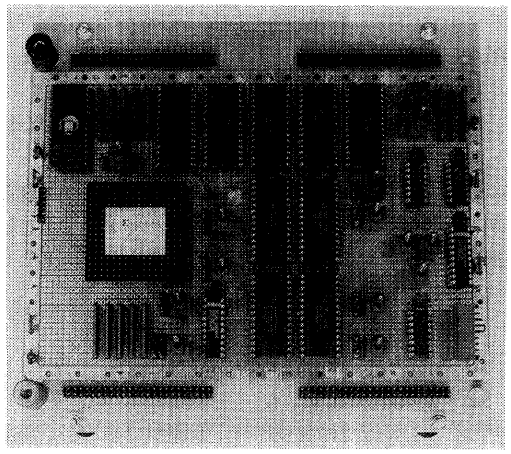


図 2 MD ボードの写真
Fig. 2 Photo of Microcomputer Development Board.

けの制御メモリ，主メモリから構成され，**図1**に示すように接続されている．構成要素がすべて RAM 構造になっている点が本ボードの大きな特徴である．MD ボードには入出力部はなく，デバッグボードと PC の全面的な支援を受ける．このためにボード上の接続線はすべてコネクタを通して外部に出されている(**図2**)．

MD ボードと PC の間は**図3**に示すように3種類の線で接続されている．

① ダウンロード線

LCA はチップ内に外部からの構成データを受け取る機能があり，この線を用いて LCA の構成 (Configuration) を行う．

② 計算機の入出力線

MD ボードのデータバスが直並列変換器を通して PC の直列入出力ポートに接続され，PC が MD

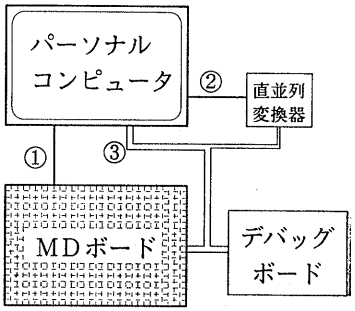


図 3 システム構成
Fig. 3 System configuration.

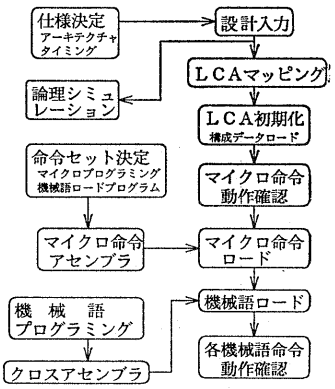


図 4 プロセッサの開発手順
Fig. 4 Flow of microprocessor development.

ボード上に開発された計算機の入出力装置となる．

③ デバッグ線

MD ボードから出ているすべての線が PC の並列入出力ポートに接続され，デバッグ時に必要なデータの入出力に使用される．マイクロ命令やクロックもここから供給できる．

図3のシステムを用いて MD ボード上に計算機を開発する手順の概略を**図4**に示す．

4.2 ソフトウェア構成とマイクロアセンブラ

ソフトウェアは CAD システムと開発デバッグ支援システムの2つである．CAD システムは設計入力，論理シミュレーションおよび LCA へのマッピングの3つの部分から成っているが，いずれも PC および WS 上の市販の CAD システムを利用している．

開発デバッグ支援システムは，①マイクロアセンブラ，②クロスアセンブラ，③デバッグツールから構成されており，マイクロアセンブラがその中心になっている．本節ではマイクロアセンブラについて述べ，次節でデバッグ支援について述べる．

本システムで使用するマイクロアセンブラは，設計によってターゲット計算機が異なるため機械独立型でなければならない．この型のマイクロアセンブラにはいくつかの種類があるが，デバッグのためにマイクロ命令と直接の対応が必要なことと，逆アセンブラを同時に開発するため，リスト形式[8]を用いる．リスト形式ではマイクロ命令の各フィールドに対しニモニックとビットパターンの対応表をマイクロ命令定義ファイルで与え，マイクロプログラムはニモニックの羅列で記述する．ドキュメント性を高めるため対応するマイクロ命令ビットを持たない冗長なニモニックが導入されている．図5に図7のプロセッサのマイクロ命令定義ファイルを示す．この定義ファイルはC言語のヘッダファイルになっており，各行は

{"mnem", b1, b2, value}

の形をとる．mnem はマイクロ命令のビット位置 b1～b2 の値が value であることを示す．冗長なニモニックでは b1, b2, value を CODE_LENGTH, 0, 0とし，マイクロ命令アドレス部は b1, b2 のみが有効となる．また，アセンブル時に対応するニモニックの指定されていないビット位置には0が入れられる．

次に例として図4の中の機械語プログラムを PC から主メモリへロードする部分のマイクロプログラムを示す．下線部が冗長なニモニックであり，これによって RT 形式に近い記述になっている．

```
: MAIN_LOOP
    IF(SW) _GOTO FETCH
                    ; SW : Mode Switch
: WAIT_IN_CHAR
    IF( I ) _GOTO READ_CHAR
                    ; I : Input Flag
    GOTO WAIT_IN_CHAR
: READ_CHAR
    dR1<=sIO
: WRITE_TO_MEMORY
    dM<=sR1, aPC    ; Addressed by PC
    PC++, GOTO MAIN_LOOP
```

```
/* 冗長なニモニック */
 {"=", CODE_LENGTH, 0,0},
 {"LAT", CODE_LENGTH, 0,0},
 {",", CODE_LENGTH, 0,0},
 {"IF", CODE_LENGTH, 0,0},
 {"(", CODE_LENGTH, 0,0},
 {")", CODE_LENGTH, 0,0},
 {"GOTO", CODE_LENGTH, 0,0},
/* Sourse Register */
 {"sPCH", 39,37, 0x1},
 {"sPCL", 39,37, 0x2},
 {"sMARH", 39,37, 0x1},
 {"sMARL", 39,37, 0x2},
 {"sM", 39,37,0x3},
 {"sIO", 39,37,0x3},
 {"sR0", 39,37,0x4},
 {"sR1", 39,37,0x5},
 {"sR2", 39,37,0x6},
 {"sR3", 39,37,0x7},
/* Destination Register */
 {"NULL", 36,33, 0x0},
 {"dPCH", 36,33, 0x1},
 {"dPCL", 36,33, 0x2},
 {"dMARH", 36,33, 0x3},
 {"dMARL", 36,33, 0x4},
 {"dM", 36,33, 0x6},
 {"dIO", 36,33, 0x6},
 {"dLAT", 36,33, 0x7},
 {"dR0", 36,33, 0xC},
 {"dR1", 36,33, 0xD},
 {"dR2", 36,33, 0xE},
 {"dR3", 36,33, 0xF},
/* ALU & Shifter Function */
 {"SHR", 32,29, 0x4},
 {"RCR", 32,29, 0x5},
 {"SHL", 32,29, 0x6},
 {"RCL", 32,29, 0x7},
 {"", 32,29, 0x8},
 {"@", 32,29, 0x9},
 {"&", 32,29, 0xA},
 {"|", 32,29, 0xB},
 {"^", 32,27, 0x30},
 {"+", 32,27, 0x38},
 {"--", 32,27, 0x3C},
 {"++", 32,27, 0x33},
 {"-", 32,27, 0x37},
/* Other Function */
 {"C2_LD", 26,25, 0x1},
 {"FLAG", 26,25, 0x2},
 {"sM", 24,24, 1},
 {"dM", 24,24, 1},
 {"PC+", 23,23, 1},
 {"sPCH", 22,22, 1},
 {"sPCL", 22,22, 1},
 {"aPC", 22,22, 1},
 {"aMAR", 22,22, 0},
/* Condition Select */
 {"C1", 21,18, 0x1},
 {"C2", 21,18, 0x2},
 {"Z", 21,18, 0x3},
 {"S", 21,18, 0x4},
 {"I", 21,18, 0x5},
 {"O", 21,18, 0x6},
 {"GOTO", 21,18, 0x7},
 {"INT9", 21,18, 0x9},
 {"BUS0", 21,18, 0xA},
 {"BUS1", 21,18, 0xB},
 {"INT12", 21,18, 0xC},
 {"INT13", 21,18, 0xD},
 {"SW", 21,18,0xE},
 {"INST", 21,18, 0xF};

/* Bit 17-10 Not Used */
/* Label Field */
   Micro labelField = {
     NULL, 9,0, 0};
```

図5　マイクロ命令定義ファイル
Fig. 5　Microinstruction Definition File.

4.3 デバッグ支援

MD ボードと PC の間のデバッグ線（マイクロ命令，アドレスバス，データバス）を使ってプロセッサチップを制御すると，任意の RT 操作を行うことができる．例えばレジスタAにデータをセットするには，データバスに必要なデータを出してマイクロ命令

A←databus

を実行する．この結果を見るには，マイクロ命令

null←A

を出して，データバスを観測する．ここで null は行き先レジスタなしを示す．

このようにして1ステップずつマイクロ命令を対話形式で実行し結果を観測すれば，初期の RT レベルのデバッグに非常に有効である．ここで重要なことは，レジスタの出力はゲートのみを通してバスに出るので，観測にはクロックを必要としない．したがって，どの時点で内部状態を観測しても次の動作には影響を与えず，観測バスを設けたのと同じ機能を持つことになる．

マイクロプログラム制御方式のプロセッサでは，動作中のデータバスとマイクロ命令を同時に観測すると，バス上のデータの出所と行き先がわかる．バス上のデータを PC に取り込み，行き先レジスタの表示を更新すると，常にその時点のすべてのレジスタの内容を表示しておくことができる．また，マイクロ命令はビット数が多い上にいろいろなフィールドに分かれているため，16 進や2進形式で入力せず，前述のマイクロアセンブラを用いて RT 形式の記述から2進形式のマイクロ命令を生成している．観測表示のときは逆アセンブラがマイクロ命令定義ファイルを参照してマイクロ命令をニモニック表示する．

4.3.1 デバッグモード

① シングルステップモード

マイクロ命令を1つずつ対話型で2進形式に変換して送出し，結果を表示する．このモードは最も初期のRT レベルのデバッグに使用され，クロックも PC から供給される．**図6**(a)にデバッグ中の画面のハードコピーを示す．上部のレジスタ表示部はレジスタの内容のみを更新し，下部のマイクロ命令は1行ごとに入力しただちに実行される．

② 連続モード

マイクロアセンブラによってファイルに生成されているマイクロ命令を，クロックとともに連続してプロセッサに供給する．このモードは機械語の作成，デバッグ時に使われる．

③ 観測モード

制御メモリ内のマイクロ命令を実行させて動作を観測する．表示のために逆アセンブラが走るので，観測できる速さはソースプログラムの複雑さに依存する．観測中の画面は図6(b)に示すように1ステップが16 進表示とその逆アセンブル結果の2行に表示される．

④ 高速モード

動作は観測モードと同じであるが，高速化のため観測結果を表示せず，ファイルに取り込む．

⑤ 制御メモリ書き込みモード

制御メモリアドレス，マイクロ命令と制御メモリライト信号を出し，マイクロ命令を PC のファイルから制御メモリに書き込む．

```
    <<送信モード1>>  [シングルステップ]

PCH=00  PCL=01 MARH=2D MARL=02  LAT=1D     M=B2
 R0=3A   R1=B2   R2=1D   R3=53  FLG=00  BUS=B2

DSW
データバスへの出力値 [16進数] => [現在 00] => 3A
R0 <= DSW
R1 <= R0
R2 = SHR R0
LAT = R2
R3 = LAT + R1, FLAG
DSW
データバスへの出力値 [16進数] => [現在 3A] => 00
PC1 <= DSW
NULL <= PCL
PC++, NULL <= PCL
R1 <= M, aMAR
M <= R1, aPC
```

(a) シングルステップモード
(a) Single step mode.

```
    <<受信モード1>>  [逆アセンブル]

PCH=01  PCL=03 MARH=20 MARL=00  LAT=25     M=08
 R0=24   R1=00   R2=00   R3=00  FLG=08  BUS=08

<001>  <60 01 7C 00 00>
FETCH   NULL=M, aPC, INST
<01C>  <8F 80 80 00 00>
ADDR0   LAT<=R0, PC++
<01D>  <67 81 40 00 00>
        MARH<=M, aPC
<01E>  <00 00 80 00 00>
        PC++
<01F>  <69 81 5C 01 1A>
        MARL<=M, aPC, GOTO _ADR0
<11A>  <79 C5 9C 00 01>
_ADR0   R0=LAT+M, aMAR, FLAG, PC++, GOTO FETCH
<001>  <60 01 7C 00 00>
FETCH   NULL=M, aPC, INST
```

(b) 観測モード
(b) Monitor mode.

図6 デバッグ中の画面のハードコピー
Fig. 6 Hardcopy of screen in debugging.

4.3.2 プロセッサ設計上の問題点

上述の方法でデバッグするためには，MD ボード上にプロセッサを設計する際に次の制限がある．

① チップの内部を観測するために，外部からデータを読むとき（メモリリード，I/O リードなど）以外は，内部バス(のうちの1本)のデータを外部端子を通してチップ外のデータバスに出しておく必要がある．通常の設計においてこれは問題にならない．

② 外部からデータをセットするために，外部データをソースとして，データをセットされるレジスタを行き先レジスタに指定できるマイクロ命令を組み込んでおく必要がある．

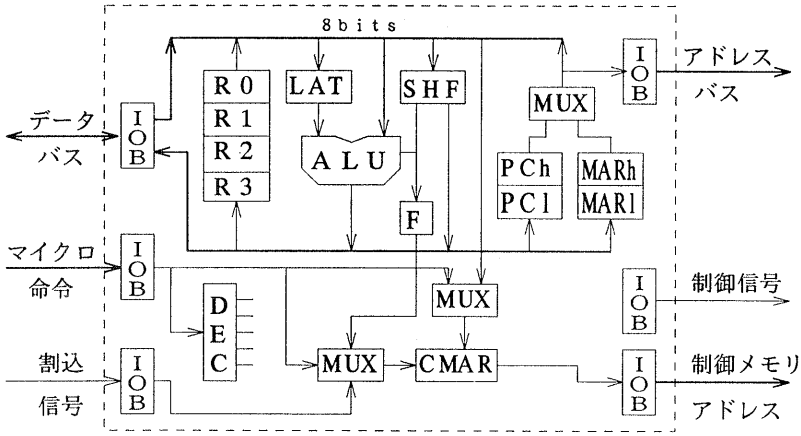
③ バスを通さずに内容を変更できるレジスタ(例えばカウンタ）の変更後の内容はバスに出るまで観測できない．観測のためにマイクロ命令を挿入しなければならないので，この種のレジスタは使用しない方が望ましい．

5. 実験と評価

MD ボードを用いて約3年の間に数種類の異なるアーキテクチャのプロセッサが開発されている．

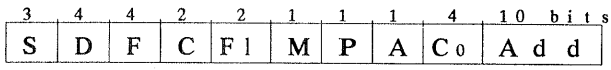
5.1 開発されたプロセッサの例

図7に示す8ビットプロセッサは2バス構造の基本的な要素のみを持つ単純なプロセッサである．このプロセッサのアドレスバスを8ビットにし，汎用レジスタを1つにするとほぼ最小構成の8ビットプロセッサとなり，学生実習で設計するときの基本になると考えられる．LCA 内の配置配線図を**図8**に示す．このプロセッサを用いて KUE-CHIP[4] のエミュレーションが行われた．



IOB	入力出力バッファ	MUX	マルチプレクサ
R0～R3	レジスタ	PCh(l)	プログラムカウンタ上位（下位）
LAT	ラッチ	MARh(l)	メモリアドレスレジスタ上位（下位）
ALU	算術論理演算器	DEC	デコーダ
SHF	シフタ	CMAR	制御メモリアドレスレジスタ
F	フラッグ(キャリー1，2，ゼロ，符号，あふれ)		

(a) ブロック図
(a) Block diagram.



S	Source Register
D	Destination Register
F	ALU&Shifter Function
C	Carry Control
Fl	Flag Control
M	Memory I/O Selection
P	PC Count
A	Address Selection
Co	Condition Selection
Add	Control Memory Address
8 bits	are not used

(b) マイクロ命令
(b) Microinstruction.

図 7 8ビットマイクロプロセッサ
Fig. 7 8-bit microprocessor.

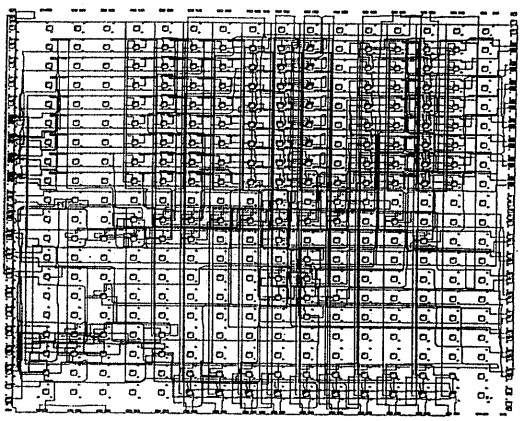


図 8 8ビットマイクロプロセッサの内部配線図
Fig. 8 Layout of 8-bit microprocessor.

図 9 に示す16ビットのプロセッサは1バス構造の演算回路を高速化したプロセッサであり，π の値5,000桁を約30分で計算できる(クロック 4MHz)．このプロセッサは情報処理技術試験の計算機 COMET[9]の機械語レベルのエミュレーションが可能である．

これらのプロセッサは基本的な機能のほかに，文献4)で KUE-CHIP の拡張機能として要求されているマイクロプログラム制御，メモリ空間4,000語，割込み機構とスタックの導入，ローダの ROM 化（ここではマイクロプログラム化）はすべて実現されている．MD ボード上ではこれらの機能は容易に実現でき，学部上級の半年から1年の実習教育用としては十分なシステムであると考えられる．

5.2 実習課題と実施方法

実習課題は

① コマンドが数個のモニタプログラムの作成

② 簡単な応用プログラムの作成

例えば電卓のシミュレーション，e, π の多倍長計算，プリンタの制御など

③ ①②で使用する2進16進変換命令，2進10進変換命令，乗除算命令，多倍長加減算命令などのマイクロプログラム化

ができるプロセッサを開発する．

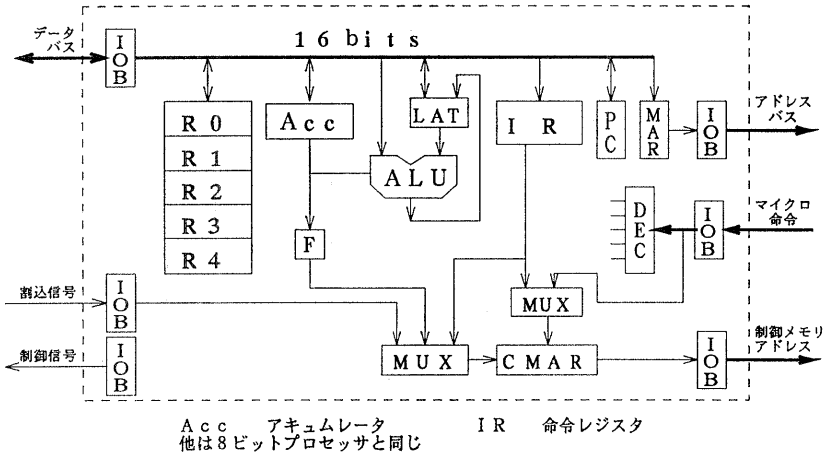


図 9 16ビットマイクロプロセッサ
Fig. 9 16-bit microprocessor.

制約は使用している LCA (XC 3090) の規模による

① ゲート数 9,000 (論理ブロック 320 個)

② 内部バス 40 本

と，MD ボードによる

③ アドレス，データバス各 16 ビット

④ マイクロ命令 40 ビット

であり，③④は外部端子の位置も固定されている．

この制約の下で各自特徴のあるアーキテクチャを設計する．ハードウェア，マイクロプログラム，ソフトウェアを考慮して検討しなければならない点には

① 語長：8ビットまたは16ビット

② バス構造：8ビットでは2または3バス，16 ビットでは1または2バス

③ レジスタの数と機能

④ 演算回路の機能

⑤ マイクロプログラム制御部の機能

などが考えられる．

指導者①～④の制約の下で①～⑤のどのような組合せが実現可能かおよその目安を与えておき，設計の各時点で問題点（特に制約内での実装可能性）をチェックしなければならない．

5.3 実習結果と問題点

学習者は「独自の計算機」を完成させたときの満足感は大きく，将来の自信にもつながっている．また，プログラミングの段階になって機械語の重要性に対する認識が深まり，既存の計算機の命令セットを考え直すよい機会になっている．

しかし，前節で示した課題を達成するには，約1年の実習期間が必要であり，設備と全体のカリキュラムの関係から一部の興味のある学生のみを対象に実施されている．より多くの学生を対象にするため，MD ボード上にマイクロプログラム教育用プロセッサをあらかじめ登載しておき，マイクロプログラミングによる命令セットアーキテクチャの実習を検討している．

システムの問題点としては①より規模の大きい LCA (XC 4010 など) の採用，②ハードウェア記述言語 (HDL) による設計入力の導入などがある．①についてはシステムを改良中である．②について，本学科では回路図入力による CAD 実習はすべての学生を対象に行われているので，その延長として本システムでも回路図入力を採用している．しかし，アーキテクチャ教育における HDL による機能記述の有効性を検討するために，HDL が使用できる CAD システムの導入が必要である．

6. おわりに

アーキテクチャ教育の実習環境について検討し，学習者が自由に設計したアーキテクチャをただちに実現し，検証し，必要なら何回でも設計し直し，ソフトウェアの開発まで可能な教育用システムを構築した．本システムの特徴は次の点に要約される．

① RAM 構造の LSI (LCA と RAM) のみからなる MD ボードの中に学習者が設計したアーキテクチャを実現する．

② PC から設計データ，マイクロプログラム，機械語プログラムを順にダウンロードする方法が採用され

ている．

③ PCの支援によりハードウェアデバッグが行われる．①②によって実装（配線など）に要する時間と労力を設計にまわすことにより，アーキテクチャの本質的な部分の教育が容易になる．また，設計したハードウェアとソフトウェアが，実装時間を必要とせずに，シミュレーションではなく，実機で検証できる点で効果的である．③によって記号形式のマイクロプログラムで RT レベルのすべての操作を PC から制御，観測できる．この方法はハードウェアデバッグに非常に有効である．約3年の間に数種類の異なるアーキテクチャのプロセッサが本システムを用いて開発され，システムの有効性が確認されている．

しかし，開発対象のプロセッサがマイクロプログラム制御方式に限られているので，今後結線制御方式に対応する方法を検討しなければならない．また，学習者が開発したアーキテクチャをどのように評価するかが課題である．

謝辞 貴重なご意見をいただいた査読者の方々と京都産業大学の奥川峻史教授また，システムの実現（ボード，マイクロアセンブラ，設計例等の作成）に協力していただいた京都産業大学の岡田光博氏（現同大学計算機センタ）と古田勇次氏（現 NEC）に感謝します．

参 考 文 献

1) 渋井二三男，竹本宣弘：マイクロコンピュータインターフェース学習環境とその支援システム，電子情報通信学会誌，Vol. 75, No. 5, pp. 488-495 (1992).
2) 庄野克房：大学における LSI 設計教育，電子情報通信学会誌，Vol. 75, No. 5, pp. 530-533 (1992).
3) Clark, T. R.: Fitting Programmable Logic, *IEEE Spectrum*, Vol. 29, No. 3, pp. 36-39 (1992).
4) 神原弘之，安浦寛人：計算機教育用マイクロコンピュータの開発とその応用，情報処理，Vol. 33, No. 2, pp. 118-127 (1992).
5) 末吉敏則，田中康一郎，柴村英智：再構成可能な論理 LSI を用いた教育用マイクロプロセッサ：KITE，情報処理学会計算機 アーキテクチャ研究会，96-15 (1992).
6) 荻原 宏：マイクロプログラミング，産業図書 (1977).
7) 馬場敬信：マイクロプログラミング，昭晃堂 (1985).
8) 馬場敬信：マイクロプログラム記述言語とその処理系，情報処理，Vol. 28, No. 12, pp. 1573-1584 (1987).
9) 日本情報処理開発協会：情報処理技術者試験案内書 (1992).
10) XILINX: Programmble Gate Array Data Book (1992).

（平成4年 9 月17日受付）
（平成5年11月11日採録）

井上 訓行（正会員）

昭和18年生．昭和41年京都大学工学部数理工学科卒業．日本電気(株)を経て昭和44年より京都産業大学に勤務．現在同大学工学部助教授．論理設計，計算機アーキテクチャ等に興味を持つ．「コンピュータの論理設計」（共訳，共立出版）など．電子情報通信学会会員．