教育用パイプライン処理マイクロプロセッサ PiPICO

西村克信　　額田多政　　天野英晴

慶應義塾大学大学院 理工学研究科 計算機科学専攻

〒223-8522 横浜市港北区日吉 3-14-1

nisimura@am.ics.keio.ac.jp

あらまし　大学における計算機教育において、実際に教育用マイクロプロセッサを設計し、FPGA 上に実装する実験・演習は、学生の計算機アーキテクチャに対する理解を深める点で重要であり、各地で試みられている。われわれは、学生に命令セットを開放し、自由に命令セットを設計させる一方で制限された資源と時間内に実験を終えるための枠組みとして PICO を提案し、実験用ボードを実装し、実際に学生実験に用いてきた。しかし、現在の PICO と実験用ボードは、最近の計算機アーキテクチャの基本であるパイプライン化に対応しない等の問題点を持っていた。そこで、本稿では、PICO をパイプライン化しパイプライン構成の教育を可能にした PiPICO と、新しい実験用ボードを設計した。予備実験の結果、充分、実験時間中に終了する配置配線時間で、PiPICO は新たな実験用ボード上の FPGA に実装できることが明らかになった。

キーワード　教育用マイクロプロセッサ, パイプライン処理, FPGA, アーキテクチャ教育

PiPICO: A microprocessor framework for education of pipelined architectures

Katsunobu Nishimura　　Kazumasa Nukata　　Hideharu Amano

Dept. of Computer Science, Keio University

Abstract　A design and implementation of a microprocessor in student experimental lab. is important for computer architecture education. A lot of educational microprocessors and experimental kits have been proposed and developed. We have also proposed and tried an educational microprocessor framework PICO. This framework allows students to design their own instruction set in the limited hardware and design time which can be prepared in student lab. Here, we extend our framework for education of recent pipelined architecture, and design a new experimental kit. Preliminary design result shows that the extended framework called PiPICO can be easily implemented on an FPGA with 36,000 gates with a reasonable time for place-and-route.

key words　Educational microprocessor, Pipelined architecture, FPGA, Architecture education

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マイクロプロセッサの実現である。従来の教育用マイクロプロセッサは、ペーパマシンとして実現され、シミュレータ上でのみ動作するものが多かった。しかし近年、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、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。これらの例として九州大学におけるKUE-CHIP2[2]、QP-DLX[3]、九州工業大学のKITE[4]、KITE2[5]、DLX-FPGA[6]、広島市立大学の安佐[7]、City-1[8, 9]、東大理学部情報工学科の実験[10]、明治大学のスタックマシン[11]などが挙げられる。

これに対し我々は、データパスのみを指定し命令セット等をオープンにした教育用マイクロプロセッサによる学生実験を行っている[12, 13]。これは、学生が与えられたデータパス上でプロセッサをハードウエア記述言語PARTHENON/SFL[14]を用いて自由に記述し、シミュレータ上で容易に動作を確認することができる。また、設計したプロセッサの論理合成/配置配線を行ない、書き換え可能なゲートアレイ（FPGA）上に転送し、実際のハードウエア上でその動作を確認することができる。

このような教育用マイクロプロセッサを作成する試みは、一応計算機アーキテクチャの基本を理解した学生に対して、実験の枠組の中で行なっている場合が多い。しかしPARTHENON/SFLは、教科書の上でしか存在しないモデルコンピュータをシミュレータ上で簡単に実現できる。これは、アーキテクチャの初等教育における講義や演習に利用することにより、学生の学習意欲を増大し、学習効果を高めることのできる可能性を持っている。このことを利用し、計算機アーキテクチャの授業の最初にSFLとパルテノンを導入し、モデル計算機を設計しながら計算機アーキテクチャを理解していく演習つき授業に関して、検討を行ない[15]、試行した[16]。この結果、様々な問題点が明らかになったものの、アーキテクチャ教育の初等段階に関しての教育システムをひととおり構築することができた[17]。

しかしこれまでの我々の試みは、計算機アーキテクチャの基礎を学ぶためのもので、設計するマイクロプロセッサは、古典的な3バス方式のRISCプロセッサであった。そのため最近の計算機アーキテクチャでは、基本的な動作となりつつあるパイプライン化がなされていない。アーキテクチャ教育の次の段階では、マイクロプロセッサのパイプライン処理に関して修得する必要があると思われる。そこで本報告では、アーキテクチャ教育の次の段階として、パイプライン化について検討する。今までの方針を引き継ぎ、初期段階で導入した教育用16bitマイクロプロセッサPICOをパイプライン化する方法について述べる。また、これまで明らかになった様々な問題点について考察する。

2 教育用マイクロプロセッサ PICO

2.1 データパス

教育用マイクロプロセッサPICO[12, 13]は、4ビットから16ビットまでの様々なサイズをと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ある。学生はその意欲と時間に応じて、命令セットを自由に付加して自分達自身のプロセッサを構築していくことができる。ただし、実際に学生が実験を行う場合、実験時間、実装を行なうFPGAに制限があるのが普通である。そこで、PICOではデータパスの概略はあらかじめ決めておく。図1に、16ビット版のPICOであるPICO-16のデータパスを示す。

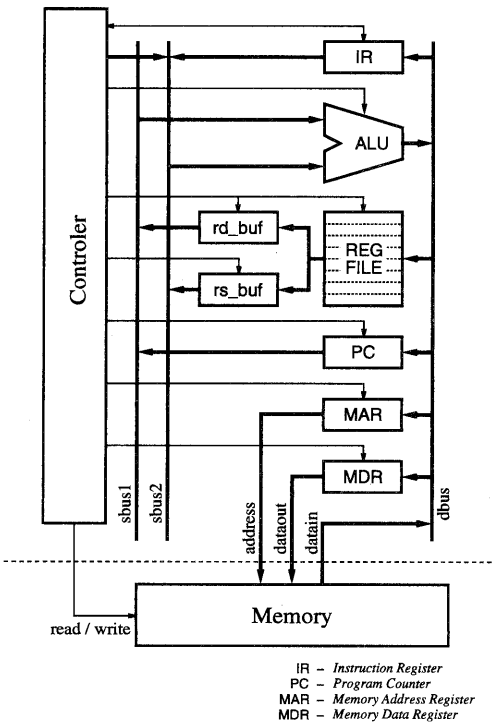

図1: PICOのデータパス

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出し、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、FPGA内部に実装されるRAMと出力バッファ用のレジスタ（rd_buf、rs_buf）から構成される。また、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（sbus1、sbus2、dbus）を介して接続されている。

2.2 命令セット例

学生の学習段階が浅い場合や実験時間が不足する場合は、さらに命令の枠組もある程度与える必要がある。この場合、以下のような指針を与える。

- 命令長の制約によりオペランドは2つまで
- 命令は即値をとる命令群とそれ以外から構成される
- 即値をとる命令は命令長の半分を即値領域に用いる

この指針に沿って設計したPICO-16の命令セットの構成例を以下に示す。この構成でXilinx社XC4010（10000ゲート相当）1チップに実装可能である。場合によっては命令形式の概略を示したり、命令のSFLによる記述例を示すことにより、学生にとっかかりを与えることができる。

2.2.1 命令形式

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。すべての命令は16ビット長で、4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2に示す。

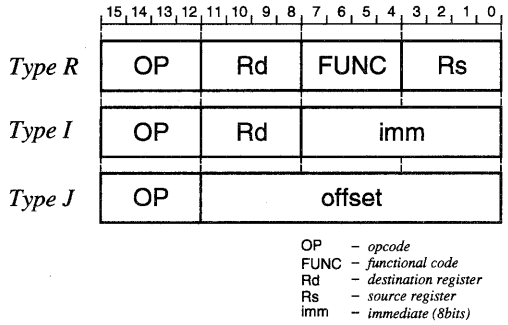

図2: 命令形式

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4ビットのファンクションコードを有している。J型命令は、分岐命令および条件分岐命令に使用する。命令長の制限によりレジスタ数はR0からR15までの16とした。またR0は、値が常に0となる。

2オペランド命令を用いていることを除き、ほぼMIPS系のアーキテクチャ[18, 19]に準じた命令セットとなっている。したがって、これを教材として用いている場合は、理解が容易になる。

2.2.2 ロード/ストア命令

PICOの基本形は、完全なロード/ストア・アーキテクチャであり、メモリを直接操作できるのは、これらの命令のみである。簡単なレジスタ間接アドレッシングのみ用意しているが、学生の希望によっては短いdisplacementを付けることもできる。レジスタ／メモリ演算機能を付けるのは、全面的に命令セットの考え方を変えるので、今まで試みた学生は居ない。

表1: ロード/ストア命令

命令	形式	意味
LD	Rd, (Rs)	Rd ← M[Rs];
ST	(Rd), Rs	M[Rd] ← Rs;
MOV	Rd, Rs	Rd ← Rs;

2.2.3 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。命令には加減算、論理演算、シフト演算などがある。即値形式では16ビットにゼロ拡張された即値が利用できる。

表2: 算術論理演算

命令	形式	意味
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 ← $Rd_{Rs-1..0}$ ## $Rd_{15..Rs}$;
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 ← $Rd_{imm-1..0}$ ## $Rd_{15..imm}$;
LHI	Rd, #imm	Rd ← $imm_{7..0}$ ## 0^{8};

2.2.4 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。学生によっては flag を設ける設計を行う場合もあり、この場合は、比較命令をフラグのセットに用いる。

表 3: 比較命令

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

2.2.5 制御命令

プログラムの制御は分岐命令と条件分岐命令でなされる。分岐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 12 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 R15 に格納する（手続き呼び出しで使う）。
- BEQZ、BNEZ 命令は R15 と 0 の比較結果により、符号付き 12 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。

表 4: 制御命令

命令	形式	意味
J	offset	PC $\leftarrow$ PC + ($(\text{offset}_{11})^4$ ## $\text{offset}_{11..0}$);
JR	Rs	PC $\leftarrow$ PC + Rs;
JAL	offset	R15 $\leftarrow$ PC;
		PC $\leftarrow$ PC + ($(\text{offset}_{11})^4$ ## $\text{offset}_{11..0}$);
JALR	Rs	R15 $\leftarrow$ PC;
		PC $\leftarrow$ PC + Rs;
BEQZ	offset	if (R15 == 0)
		PC $\leftarrow$ PC + ($(\text{offset}_{11})^4$ ## $\text{offset}_{11..0}$);
BNEZ	offset	if (R15 != 0)
		PC $\leftarrow$ PC + ($(\text{offset}_{11})^4$ ## $\text{offset}_{11..0}$);

上記はあくまで標準的な方法であり、flag を用いた分岐を行う学生もあり、分岐条件の設定もいろいろな方法で行われている。学生による差がもっとも生ずる部分である。

2.3 PICO-16 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 3）。このボードは教育用マイクロプロセッサを転送する FPGA である XILINX/XC4010、プログラム格納用の RAM、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。

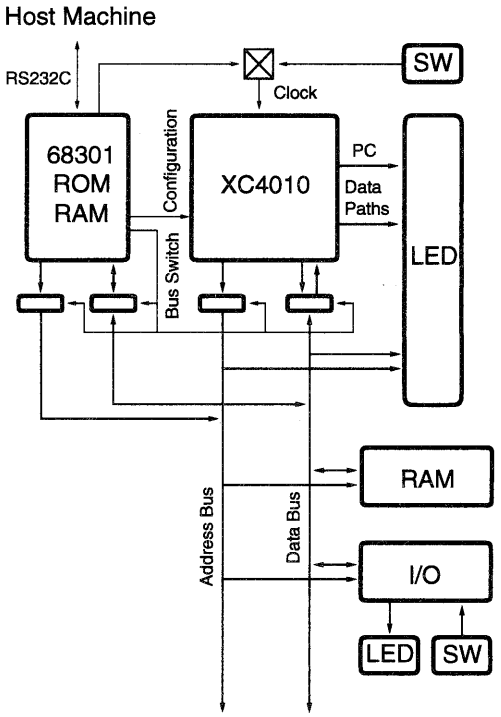

図 3: PICO-16 実験用ボードの概略

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 16 ビット CPU（TMP68301）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、FPGA へのクロック供給、RAM の内容の確認などを行なうことができる。

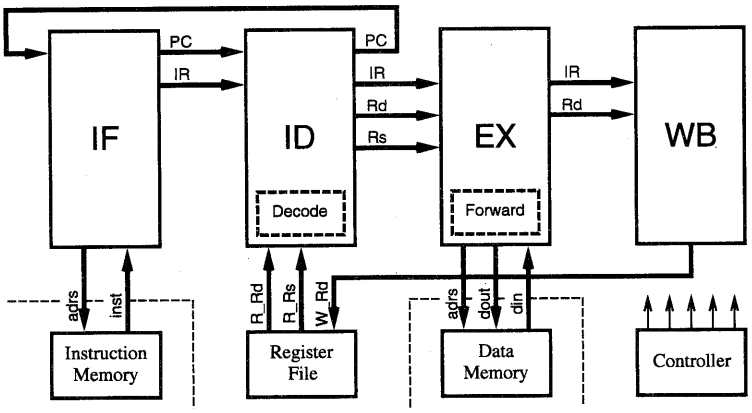

図 4: PiPICO のブロック図

2.4 現状における問題点

1994 年度より、PICO-16 および前述の実験用ボードを用いて、慶応義塾大学ならびに東京工科大学の学部 3 年生を対象に学生実験を行なってきた。その結果、いくつかの問題点が明らかになってきた。

パイプライン化の必要性: 今までの PICO-16 は古典的な 3 バス方式の RISC プロセッサであり、現在の計算機アーキテクチャを理解する上で基本となるパイプラインによる動作を考慮していない。最近は計算機アーキテクチャの教育は、古典的な 3 バス方式のデータパスを深くは教えずに、早い段階でパイプライン方式を導入する方法が確立しつつある [19]。将来は学部 3 年の実験においてもパイプライン方式のマイクロプロセッサを設計することが望ましい。

実験用ボードの FPGA 容量の不足: 実験用ボードは、設計当時としてはかなり大きめの FPGA を用いた (XC4010:10000 ゲート相当) が、現状の PICO-16 でも、すべての命令を実装すると使用率が 90%を越えてしまい、配置配線に多大な時間を要してしまう。また学生の中には、マルチプレクサやレジスタを意識せずに記述することにより、回路規模が大きくなり過ぎ FPGA に入りきらなくなってしまうことがあった。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削いでしまった。また、このボードでは、パイプライン構成の PICO を搭載することは不可能である。

実験用ボードの観測機能と I/O の問題: シミュレータ上ではマイクロプロセッサの内部が詳細に観測できるのに対し、実験用ボードではその機能が存在せず、せいぜいバスの状態を LED で観測する程度であった。そのため、マイクロプロセッサの内部状態を観測し、その上で動くプログラムのデバッグを行なうようなことが難しかった。また実験用ボード上には、入出力用として 8 ビット分のスイッチと LED しか用意されていない。このため、実装した PICO が物理的に I/O を制御を行う能力が不足し、結果として動作させるプログラムはソーティングや素数の計算など、メモリ上のデータに対する操作が主となった。この場合、学生はモニタを介してメモリの内容を確認するだけなので、シミュレーションとの区別がつきにくく、実際に自分の設計したマイクロプロセッサが動作したという実感を得ることが難しい。

以上の問題点を改善するため、我々は、PICO のパイプライン化および新しい実験用ボードの作成を行っている。本稿ではこの点について報告を行う。

3 PICO のパイプライン化

3.1 データパス

PICO-16 の設計方針および命令セットはそのままで、パイプライン化した教育用マイクロプロセッサ PiPICO を図 4に示す。

PiPICO のパイプラインの動作は、文献 [19] に書かれているパイプライン化手法とほぼ同じになるが、PiPICO ではメモリアクセスに displacement を用いないため、MEM ステージと EX ステージをひとつにまとめることができる。その結果、以下の 4 ステージで動作する。

IF — *Instruction Fetch* ステージ
命令を *Instruction Memory* からフェッチし、IR (*Instruction Register*) に格納する。

ID — *Instruction Decode* ステージ

命令のデコードを行なう。また、レジスタの値を読み出すと共に、PCをインクリメントする。分岐命令に関してはここで、飛び先を計算する。

EX — *EXecution* ステージ
ALUを用いる演算を行なう。また、ロード/ストア命令に関しては、このステージで *Data Memory* に対する読み書きを行なう。

WB — *Write Back* ステージ
計算結果をレジスタに格納する。

また、構造ハザードを回避するため、命令を格納するための *Instruction Memory* とデータを格納するための *Data Memory* はそれぞれのステージに別々に用意する。

3.2 観測機能

教育用マイクロプロセッサを設計する上で一番重要になるのは、プロセッサの内部状態の観測機能であると思われる。外部から内部状態が参照できない場合、実際にどのような処理が行なわれているのかわからず、結局のところシミュレータに頼ってしまうことになる。これではシミュレータ上で実現されるペーパーマシンと変わらず、実際にハードウエア上で動作させる意味が無くなってしてまう。

そこで今回設計したPiPICOでは、内部の観測機能を強化した。具体的には特別な信号線を設け、外部からプロセッサ内にある任意のパイプラインレジスタの読み書きをできるようにした。ただし、プロセッサの動作中にこれらレジスタの読み書きはできないため、一時的にプロセッサの動作を止める機能が必要になる。また、ハードウエア言語によるこれらの記述は、かなり技術を要する。したがって、学生にはあらかじめこれらを記述した枠組みを与えることにより、実験をスムーズに行なう工夫が必要となる。

3.3 モデルプロセッサとハードウエア量の評価

文献 [7] 同様、PICOでもステップアップ方式でパイプラインプロセッサを構築していく。以下の段階を想定して、それぞれのハードウェア量の評価を行った。

step1 ADD, LHI, LD, ST, J命令等の簡単な命令のみを実装したパイプライン化されたPICOを学生に提示する。このパイプラインは、フォワーディング、ストール機能を持たない基本的なものである。学生はこのベース（PiPICO/P1）をシミュレーションすることによりパイプラインの動作の概略を知る。

step2.1 上記のPiPICO/P1に命令を付け加えたPiPICO/P2を設計する。命令は、演習用の課題（例えばソーティング、加算の繰り返しによるかけ算等）が達成できるものを揃えればよく、学生が自分で好きなように考える。ここで、最も問題となるのは分岐命令で、これについてはヒントを与える必要がある。初期段階ではALUにより飛び先を計算させてもよいし、飛び先計算専用加算器を持たせても良い。また、分岐は遅延分岐を原則とし、ストールはさせない。

step2.2 上記のPiPICO/P2をシミュレーションし、課題のプログラムを実行し、フォワーディングの問題、遅延スロットの問題等を認識する。プログラムが正しく動作するようにNOP命令を挿入する。

step2.3 PiPICO/P2を論理合成し、ゲート数、最大動作周波数を評価し、パイプライン化しないPICOと比較する。さらに、FPGA上に実装し、実際に動作を確認する。

step3 フォワーディング機能を付け加えたPiPICO/P3を設計し、PiPICO/P2と比較する。

基本となる命令を実装したPiPICO/P1、拡張を行なったPiPICO/P2、フォワーディングを行なったPiPICO/P3の、ゲート換算数、CLB数等を表5に示す。この数字からXLINX社のFPGA、XC4036EXを用いれば演習が可能であることがわかる。また、表5にこのFPGAを使った際の使用率を併せて示す。

	PICO/P1	PICO/P2	PICO/P3
ゲート	7183	9084	9487
CLB	735 56.7%	1142 88.1%	1197 92.4%
IOB	150 52.0%	150 52.0%	150 52.0%
DFF	380 12.0%	380 12.0%	400 12.6%

表 5: PICO/P のゲート換算数

4 PiPICO 実験用ボード

今回新たに設計した、実験用ボードの概略を図5に示す。

ボードの基本構成は、前回のものと変わっておらず、FPGA(PiPICO) のコンフィグレーション、メモリへの命令の書き込みのために、制御・インタフェース用 CPU（*Control Processor*）が搭載されている。命令格納用メモリ（*Instruction Memory*）およびデータ格納用メモリ（*Data Memory*）は、PiPICOおよび *Control Processor* のどちらからでも自由に読み書きできる。

前回のボードで問題になった観測機能を実現するために、表示用のLEDと共に、*Control Processor* からもPiPICOの内部状態をモニタすることができるように工夫されてい

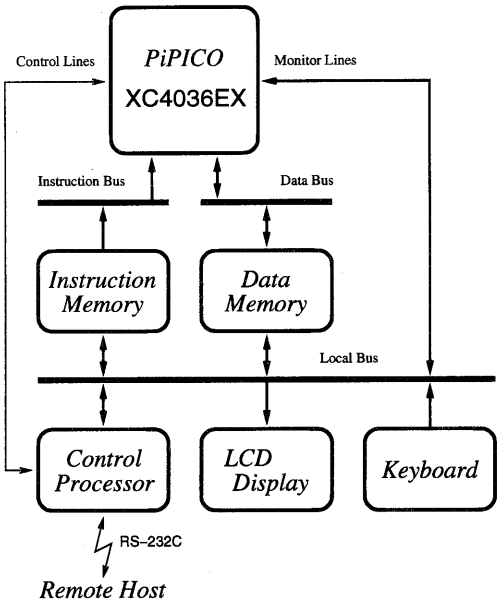

図 5: PiPICO 実験用ボードの概略

る。すなわち、PiPICO のモニタ信号線は、*Control Processor* のアドレス空間上にマップされ、特定番地を読むことにより、PiPICO の内部状態を自由に観測することができる。

入出力装置としては、簡単なキーボードと液晶表示器を備えている。これにより、PICO 上のプログラムにより目で見える形の制御を行うことができる。これらは PiPICO のアドレス空間上にマップされ、メモリの特定番地の読み書きを行うことにより、アクセスできる。

5 まとめ

学生が自由に命令セットを設計していく実験用のマイクロプロセッサフレームワーク PICO の枠組みを、パイプライン構造に拡張した PiPICO を提案し、そのための実験用ボードを設計した。現在、実験用ボードはボード設計が終了しており、PiPICO 実験は 1999 年度の東京工科大学情報工学科 3 年生および慶應義塾大学情報工学科 3 年生で試験的に導入される予定である。

参考文献

[1] 柴山, 新實. 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. 情処研報, Vol. 93, No. 49, pp. 27 – 34, June 1993. 93-ARC-100-4.

[2] 越智 ほか. 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. 信学技報, Vol. 92, No. 290, pp. 3 – 10, October 1992. CPSY92-46.

[3] 岩井沢 ほか. 計算機工学一貫教育用マイクロプロセッサ QP-DLX の開発. 情処研報, Vol. 93, No. 49, pp. 35 – 42, June 1993. 93-ARC-100-5.

[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. 情処研報, Vol. 93, No. 49, pp. 59 – 66, June 1993. 93-ARC-100-8.

[5] 末吉 ほか. FPGA を利用した教育用マイクロプロセッサ KITE2 – システムソフトウェア教育への応用 – 情処研報, Vol. 94, No. 50, pp. 25 – 32, June 1994. 94-ARC-106-4.

[6] 末吉, 井上, 奥村, 久我. 教育用 32 ビット RISC マイクロプロセッサ DLX-FPGA と教材ボードの開発. 第 3 回 FPGA/PLD Design Conference and Exhibit 論文集, pp. 579 – 588, July 1995.

[7] H.Ochi. ASAver.1: An FPGA-Based Education Board for Computer Architecture System Design. *Proc. of ASP-DAC 97*, pp. 157 – 165, January 1997.

[8] 高橋 ほか. マイクロコンピュータ設計教育環境 City-1. 情処研報, Vol. 97, No. 17, pp. 41 – 48, February 1997. 97-DA-83-6.

[9] 高橋, 吉田. 完全なインターロックを行なうパイプライン CISC/RISC の設計教育. 情処研報, Vol. 97, No. 103, pp. 97 – 104, October 1997. 97-DA-85-15.

[10] 松本. 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, December 1993.

[11] 鈴木, 井口, 山田. FPGA を用いた実習用仮想マイクロプロセッサの試作. 第 3 回 FPGA/PLD Design Conference and Exhibit 論文集, pp. 609 – 617, July 1995.

[12] 西村, 渡辺, 工藤, 天野. FPGA を用いたマイクロプロセッサ開発実験. 第 5 回パルテノン研究会 資料集, pp. 65 – 74, November 1994.

[13] 西村, 工藤, 天野. 教育用 16 ビットマイクロプロセッサ PICO-16. 第 3 回 FPGA/PLD Design Conference and Exhibit 論文集, pp. 589 – 596, July 1995.

[14] 中村, 小野. ULSI の効果的な設計法. オーム社, 1994.

[15] 天野, 西村. 計算機入門教育におけるパルテノンの利用. 第 7 回パルテノン研究会 資料集, pp. 73 – 82, November 1995.

[16] 天野. パルテノンの計算機初等教育への利用 – 実施報告 –. 第 10 回パルテノン研究会 資料集, pp. 51 – 56, April 1997.

[17] 金森, 西村, 天野. 教育用マイクロプロセッサ・パイプライン化 PICO. 第 11 回パルテノン研究会 資料集, pp. 59 – 66, December 1997.

[18] D.A.Patterson, J.L.Hennessy, 成田光彰 訳. コンピュータの構成と設計 [上]. 日経 BP 社, 1996.

[19] D.A.Patterson, J.L.Hennessy, 成田光彰 訳. コンピュータの構成と設計 [下]. 日経 BP 社, 1996.