計算機アーキテクチャ 100－8
（1993．6．11）

教育用マイクロプロセッサ KITE とその開発支援環境

田中 康一郎† 小羽田 哲宏‡ 久我 守弘† 末吉 敏則†‡

† 九州工業大学 マイクロ化総合技術センター

‡ 九州工業大学 情報工学部

E-mail : tanaka@cms.kyutech.ac.jp
alum@mickey.ai.kyutech.ac.jp
kuga@cms.kyutech.ac.jp
sueyoshi@ai.kyutech.ac.jp

書換え可能な FPGA を利用して，学習者が自らの手で設計から動作検証まで行える教育用マイクロプロセッサ KITE とその開発支援環境について述べる．KITE は基本的な計算機構成要素から成る簡素な 16 ビット・マイクロプロセッサであり，設計手法としては回路図入力による方法やハードウェア記述言語による方法，あるいはその組合せを利用できる．また，開発支援環境として評価用ボードとクロスソフトウェアを整備しているので，設計完了後直ちに LSI 実装でき，自分が設計したマイクロプロセッサの動作を手に取って確認できる．KITE はデバッグや改良のために何回でも設計のやり直しができるので，論理回路設計の経験の浅い初心者の教育にも適している．

The KITE microprocessor and its Development Support Environment

Koichiro Tanaka† Tetsuhiro Kohada‡ Morihiro Kuga† Toshinori Sueyoshi†‡

† Center for Microelectronic Systems
Kyushu Institute of Technology
Iizuka, 820 Japan

‡ Department of Artificial Intelligence
Kyushu Institute of Technology
Iizuka, 820 Japan

This paper presents an educational microprocessor called KITE and its development support environment using reconfigurable FPGA where students can perform their own design, implement, and test its operations. KITE is a 16-bit microprocessor comprising the basic computer components and students utilizes schematic editor and/or hardware description languages as design methodologies. Furthermore, the development support environment is completely provided with an evaluation board and cross software for students to directly implement in LSI the completed design and confirm the operations of the designed microprocessor. With KITE, students can debug and improve their designs for a number of times, and facilitate even those with little logic circuit design experience.

1 はじめに

半導体集積技術の向上に伴う開発期間の長期化，多品種少量生産のASIC化の進展により，相対的にLSI開発工程における設計の比重が増し，CADシステムを使用したLSI設計教育の必要性が高まってきている．また，LSI技術の進歩が次第にシステムレベルでの設計力に依存する傾向があり，最近では日本の大学におけるLSI設計教育の必要性が大学のみならず産業界からも言われるようになってきた．一方，大学や高等専門学校などの教育機関でもさまざまな方法によるLSI設計教育が検討されているが，実際にLSIの設計教育を行うには予算的制約や時間的制約などの問題を解決する必要があり，LSI実装を前提とした設計教育を行っている教育機関はまだ少ない[1],[2],[3].

そこで，我々は書換え可能なFPGA(Field Programmable Gate Array)を利用して予算的/時間的制約を緩和し，計算機工学教育や集積回路工学(論理回路設計)教育に適した教育用マイクロプロセッサKITEの開発を行っている[4],[5]．本研究で採用したFPGAは米Xilinx社のLCA(Logic Cell Array)[6]であり，デバイス内部のコンフィグレーション用SRAMに構成データをロードすることによって任意の論理回路を実現でき，学生は自分が設計したマイクロプロセッサの動作を手に取って確認できる．また，SRAM方式FPGAの採用によりデバッグや改良のために何回でも設計のやり直しができるので，大学院教育のみならず学部教育のように論理回路設計の経験の浅い初心者の教育にも適している．

本稿では，まずKITEマイクロプロセッサの概要について述べる．次いで，先に報告した回路図入力による設計の場合[5]と比較しながら，ハードウェア記述言語による設計手法，開発フロー，実装結果，実装時間について説明する．さらに，KITEマイクロプロセッサの開発支援環境について述べ，最後に本学で実施している設計教育事例を紹介する．

2 KITEマイクロプロセッサの概要

2.1 教育用マイクロプロセッサの要件

教育用マイクロプロセッサを開発するに当たり，設計仕様は次のような要件を満たす必要性がある．

- マイクロプロセッサのアーキテクチャは集積度を考慮してできるだけ簡素なものとし，かつ論理回路の基本的構成要素をすべて含むこと．
- 命令セットは，教育用として必要最小限の命令を用意すること．
- マイクロプロセッサ内部の動作や状態，すなわちデータの流れや各種レジスタの値などを外部から把握できるよう，可観測性が高いこと．
- 特別な測定器を用いなくても，計算機の動作を逐一観測できるように，命令単位ならびにクロック単位でプログラムの実行を制御できること．
- システムソフトウェアの教育において実験演習による概念の定着を図れるように，メモリ空間，データ長，命令セットに配慮し，実験対象のモデル計算機としても利用できること．

KITEマイクロプロセッサは上記の要求を考慮して方式設計ならびに機能設計を行った．

2.2 基本アーキテクチャ

KITEマイクロプロセッサは，現行の実装デバイスの集積度を考慮して，16ビットのマイクロプロセッサとしている．

アドレス空間にはメモリ空間とI/O空間があり，メモリ空間には4Kワード(アドレス幅12ビット)，I/O空間には256ワード(アドレス幅8ビット)を用意している．また，両空間へのアクセスは，命令/データ共にすべてワード単位(16ビット単位)で行う．

レジスタセットには，16ビットレジスタが2つ，12ビットレジスタが4つの計6つがある．具体的には，16ビットレジスタとしては命令レジスタとアキュムレータがあり，12ビットレジスタとしてはプログラムカウンタ，スタックポインタ，インデックスレジスタ，アドレスレジスタがある．その他に，演算結果の状態を保持する4ビットのフラグレジスタがある．

シーケンス制御部は，布線論理方式(ワイヤードロジック)としている．これには次の2つの理由がある．第一に，論理回路の設計教育が大きな目的の1つである．第二に，マイクロプログラム方式ではマイクロプログラム格納用のメモリを実装デバイスの内か外に設ける必要がある．本研究で使用した実装デバイスの場合，内部にマイクロプログラム用メモリを実装するとゲート利用率が著しく低下することになるし，一方外部にマイクロプログラム用メモリを設けるとインタフェースにI/Oパッドを多数占有してしまい現在の実装デバイスでは可観測性を損なうことになる．

2.3 命令セットとアドレス形式

命令セットは，表1に示すようなソフトウェアの作成に不自由にならない程度の基本的な命令を用意している．具体的には，データ転送命令が5種類，分岐命令が7種類，スタック命令が2種類，システム制御命令が2種類，演算命令が15種類であり，ニーモニック数で31種類，全命令では84種類である．アドレス形式は，直接，インデックス修飾，即値，レジスタ直接，含意の5種類がある．なお，命令形式は5種類であり，すべて16ビット固定長である．

表 1: 命令セット

データ転送命令		演算命令	
LD	Load	ADD	Add
ST	Store	SUB	Subtract
MV	Move	INC	Increment
IN	In	DEC	Decrement
OUT	Out	OR	Inclusive Or
		EOR	Exclusive Or
分岐命令		AND	And
		NOT	Not
JP	Jump always	LSL	Logical Shift Left
JPC	Jump on Carry	ASR	Arithmetic Shift Left
JPV	Jump on Overflow	LSR	Logical Shift Right
JPZ	Jump on Zero	ASR	Arithmetic Shift Right
JPS	Jump on Sign	ROL	Rotate Left
CALL	Call Subroutine	ROR	Rotate Right
RET	Return from Subroutine	SWP	Byte Swap
スタック命令		システム制御命令	
PUSH	Push	NOP	No Operation
POP	Pop	HALT	Halt

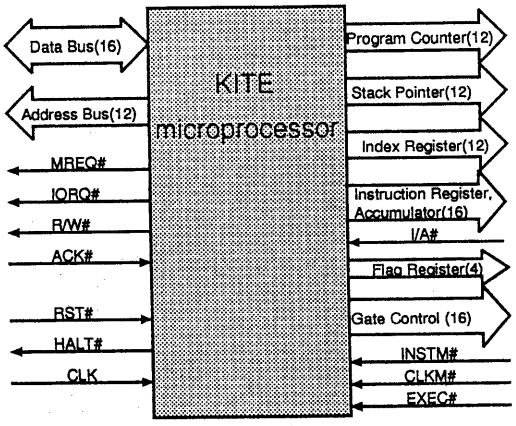


図 1: 入出力端子

2.4 ピン割当て

入出力端子は，図 1に示すようにマイクロプロセッサの基本動作に必要となる端子 (図 1 左側) とマイクロプロセッサ内部の動作や状態の観測用端子 (図 1 右側) に大別できる．基本動作に必要となる端子としては，データバス (16 ビット)，アドレスバス (12 ビット)，制御信号 (7 ビット) がある．観測用端子としては，レジスタセット (プログラムカウンタ，スタックポインタ，インデックスレジスタ，命令レジスタとアキュムレータ，フラグレジスタ) と各種レジスタの開閉ゲート観測，マイクロプロセッサの動作モード (3 ビット) がある．このように多くの観測用端子を設けることで，学習者はマイクロプロセッサ内の状態や動作を容易に把握することができる．

3 開発の流れ

KITE マイクロプロセッサは現在，図 2に示すように回路図入力による方法と 3 種類のハードウェア記述言語による方法で設計することができる．回路図入力による設計は既に報告しているので [5]，本稿では説明を割愛する．また，ハードウェア記述言語による設計については，紙面の都合で VHDL(VHSIC Hardware Description Language)[7] と ABEL-HDL[8] の 2 種類について説明する．

3.1 ハードウェア記述言語による設計

図 2に示す開発の流れに従い，VHDL と ABEL-HDL による開発工程をそれぞれ以下に述べる．

[1] VHDL による設計
VHDL では，まず方式設計/機能設計完了後，マイクロプロセッサを階層化して，モジュール単位の設計を行う．VHDL では図 3の (3) で示すように順序回路をステートマシンで記述することにより回路を設計できる．このため，(1) のように論理シンボルを用いて設計する場合と比べると，順序回路の設計が格段に容易になる．また，VHDL ではステートマシン記述以外にも算術演算子等を使用することができ，従来コンピュータ上でプログラムを作成していたのと同じ感覚で論理設計を行うことができる．

次に，記述した回路を EXEMPLAR[9] を用いて，ネットリスト (XNF:Xilinx Netlist Format) に変換する．この処理を施すことで機能レベルの回路データが論理回路レベルのネットリストとなる．また，Vantage[10] を用いることで機能レベルのシミュレーションを行うことができるため，各階層ごとに機能レベルの動作検証を行いながら設計することができる．なお，VHDL による設計は AoutoLogic[11] と QuickSim II[12] の組合せによっても行うことができる．

ネットリスト生成後，自動配置配線プログラムを用いることで構成データと呼ばれる実装に要するビットストリームデータを生成する．この LCA 専用 CAD システムの自動配置配線機能を利用することで，設計者は LSI 実装の際にレイアウト設計を行う必要がなくなり，論理設計に専念できる．構成データの生成後，そのデータを評価用ボード (KITE マイクロプロセッサボード) 上の FPGA にロードすることで実装を行い，動作検証を行う．

[2] ABEL-HDL による設計
ABEL-HDL の場合も，方式設計/機能設計完了後に，階層化設計を行う．ABEL-FPGA でも図 3の (2) で示すように順序回路をステートマシンを記述することで論理回路を設計できる．また，ABEL-HDL は VHDL と比較すると機能レベルが低いが，VHDL と同様に算術演算子も利用できるので設計時間の短縮を図ることができる．

設計後，ABEL-FPGA[13] を用いて Open-ABEL と呼ばれるネットリストに変換する．この処理では主に論理合成を行っている．

次にこのデータを XNF に変換する．現在，この処理を行うには FPGA フィッターと EXEPLAR があり，ここでは EXEMPLAR を用いた．それ以降は VHDL と同様の過程をたどる．

本研究における実装の結果から，ハードウェア記述言語を使用した場合はいずれも回路図入力によって設計する場合と比較して設計時間が大幅に短縮することを確認できた．

3.2 実装結果

VHDL と ABEL-HDL といった 2 種類のハードウェア記述言語を用いて，図 4のような内部構成の異なる 2 種類のマイクロプロセッサを設計し，実装した．実装結果を比較できるように，図 5に回路図入力による設計の場合と共に示す．

本研究で使用した実装デバイスは，米国 Xilinx 社の LCA の 4000 シリーズであり，5,000 ゲート相当の論理回路を実装することができる．実装では，論理セル (CLB:Configurable Logic Block) 利用率 90%を目標に実装を行った．

3 通りの設計手法による実装結果を比較すると，VHDL の場合には現行実装デバイスのゲート規模で KITE マイクロプロセッサの仕様を完全に満たすものを実装することができず，ALU の機能の一部に制約を課すことにより回路図入力による設計のものと同等のゲート規模となる．一方，ABEL-HDL の場合にはほぼ

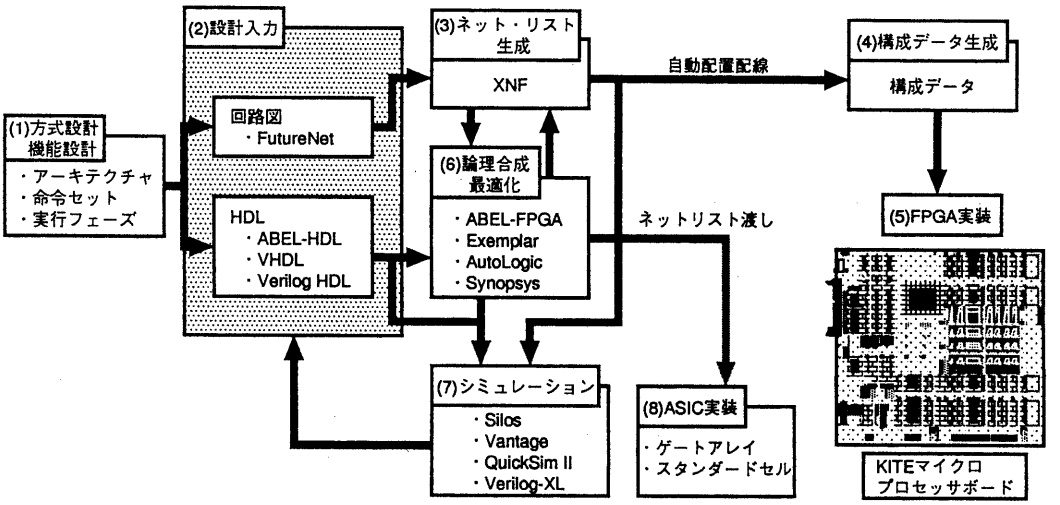


図 2: 開発フロー

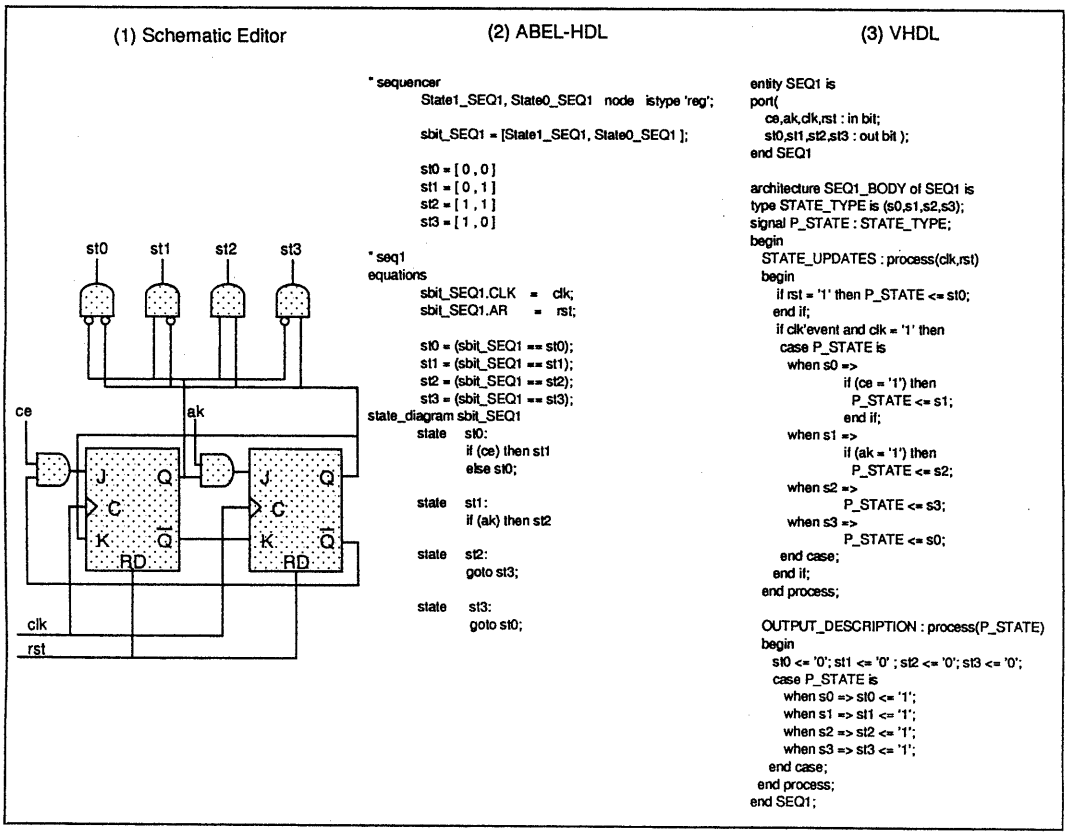


図 3: ハードウェア記述言語の記述例

同等のゲート規模で実装できている．これは，ABEL-HDL が VHDL ほど論理合成/最適化ツールに依存していないためであると考えられる．なお，VHDL で設計した完全な仕様に準拠したマイクロプロセッサは，10,000 ゲート相当の論理回路を実装可能な XC4010 では実装できており，今後の論理合成/最適化ツールの性能向上により回路図エディタで設計したものと同等になることが期待される．

Partition, Place and Route Summary Ver 1.30	(1) Schematic Editor		(2) ABEL-HDL		*(3) VHDL	
	Single Bus	Three Buses	Single Bus	Three Buses	Single Bus	Three Buses
Input XNF Design Statistics						
Number of Logic Symbols:	790	818	845	843	795	814
Number of Flip Flops:	134	101	135	102	131	97
Number of 3-State Buffers:	116	100	116	100	112	112
Number of IO Pads:	111	111	111	111	111	111
Number of Hard Macros:	0	0	0	0	0	0
Number of Nets:	1123	1120	1161	1110	1095	1090
Number of Pins:	4083	3959	4410	4099	4064	4006
Equivalent "Gate Array" Gates:	3690	3470	4118	3744	3787	3664
- From Logic:	3690	3470	4118	3744	3787	3664
- From Random Access Memories:	0	0	0	0	0	0
- From Read Only Memories:	0	0	0	0	0	0
Partitioned Design Utilization Using Part 4005PG156-6						
Occupied CLBs:	95%	93%	96%	94%	93%	96%
Packed CLBs:	75%	74%	84%	78%	72%	77%
Package Pins:	99%	99%	99%	99%	99%	99%
FG Function Generators:	75%	74%	84%	78%	72%	77%
H Function Generator:	43%	37%	31%	27%	23%	41%
Flip Flops:	21%	16%	21%	16%	21%	15%
Memory Write Control:	0%	0%	0%	0%	0%	0%
3-State Buffer:	23%	19%	23%	19%	22%	22%
3-State Buffer Output Lines:	57%	57%	57%	57%	57%	57%
Address Decoder	0%	0%	0%	0%	0%	0%
Address Decoder Output Lines:	0%	0%	0%	0%	0%	0%
Routing Summary						
Number of unrouted connections:	0	0	0	0	0	0
Number of pips used:	6181	5415	6036	5606	4947	4889
Number of local lines used:	2706	2305	2551	2441	2132	2025
Number of double lines used:	909	811	894	824	629	670
Number of long lines used:	383	363	371	359	325	315
Number of global lines:	18	20	20	17	22	16
Number of Decoder lines used:	0	0	0	0	0	0
Cpu Times						
Netlist Preparation:	00:00:04	00:00:04	00:00:04	00:00:04	00:00:04	00:00:04
Partition/Placement:	00:00:41	00:00:35	00:00:35	00:00:31	00:00:36	00:00:36
Placement Improvement:	00:00:35	00:00:45	00:00:26	00:00:33	00:00:22	00:00:29
Routing:	00:07:33	00:02:33	00:08:35	00:03:07	00:02:18	00:01:36
Total:	00:09:12	00:04:12	00:09:35	00:04:29	00:03:32	00:02:56

SPARC Station 10 Model 41 (Memory 128Mbyte)　*ただし、演算命令を限定したサブセット

図 5: 実装結果

3.3 実装時間

ハードウェア記述言語を用いて FPGA に実装する際に最も時間を必要とする処理は，論理合成/最適化と自動配置配線である．

VHDL で設計した回路と，ABEL-HDL で設計した回路とも論理合成/最適化に要する時間は 4,000 ゲート相当の回路規模で数分程度である．また，通常は階層ごとにコンパイルを実行しているため，訂正時には誤りのある階層のファイルだけ再コンパイルすればよい．そのため，論理合成/最適化処理は短時間で実行できる．

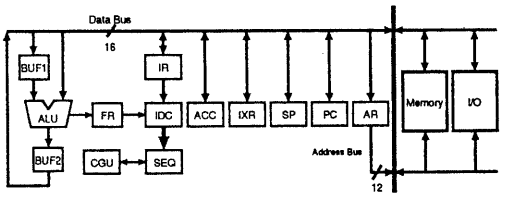
自動配置配線処理はパーソナルコンピュータとワークステーションで実行できる．図 2に各コンピュータごとの実装時間を示す．サンマイクロシステムズ社のワークステーション (SPARC Station 10) を用いて自動配置配線処理を行うと図 5の実装時間からすべて 10 分以内実装できていることが分り，ハードウェア記述言語による設計の場合もワークステーションを使用することで円滑に設計演習を行えることを確認できた．

表 2: 実装時間

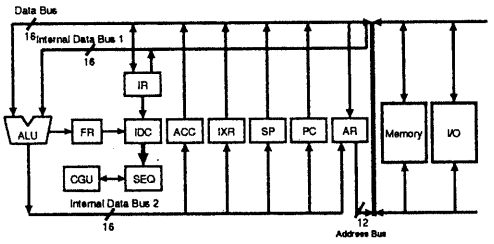
Software	Computer	Single Bus	Three Buses
Partition, Place and Route Ver.1.10	NEC PC-9801RA21 (I386DX 20Mhz) Memory 10Mbytes	20:32:56	04:23:21
	NEC PC-9801RA21 (I486SX 16Mhz) Memory 12Mbytes	06:06:30	02:00:10
Partition, Place and Route Ver.1.30	SUN SPARC Station 2 (SPARC 40MHz) Memory 64Mbytes	00:23:21	00:06:25
	SUN SPARC Station 10 (SUPER SPARC 40MHz) Memory 128Mbytes	00:09:12	00:04:12

4 開発支援環境

KITE マイクロプロセッサを実際に動作させるためには，FPGA を実装して稼働させる環境を必要とする．そこで，我々は KITE マイクロプロセッサ設計の



a) 1バス構成



b) 3バス構成

Acc = Accumulator
ALU = Arithmetic and Logic Unit
AR = Address Register
CGU = Clock Generating Unit
FR = Flag Register
IDC = Instruction Decoder
IR = Instruction Register
IXR = Index Register
PC = Program Counter
SEQ = Sequencer
SP = Stack Pointer

図 4: 内部構成

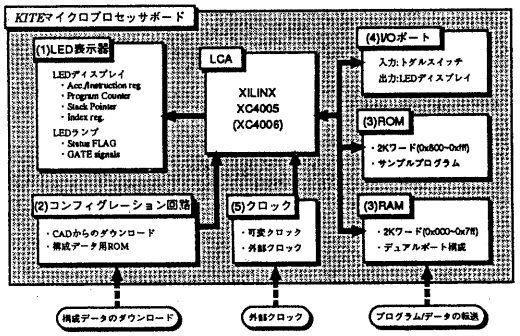


図 6: KITE マイクロプロセッサボードの構成

ための評価用ボード，つまりKITEマイクロプロセッサボードを開発した．併せて，マイクロプロセッサの動作を詳細に検証するためにクロスソフトウェア環境の構築も行った．

4.1 KITE マイクロプロセッサボード

KITE マイクロプロセッサボードには，図6に示す機能を実装している．また，KITE マイクロプロセッサボードの写真を図7に示す．以下に，図6中の番号を用いてKITE マイクロプロセッサボードの機能を説明する．

1. KITE マイクロプロセッサの観測用端子を表示するために，13 個の 7 セグメント LED ディスプレイと 20 個の LED ランプを用意している．これらは，マイクロプロセッサから出力されるレジスタの状態や制御信号を表示する．
2. 専用の CAD システムによって構成データをダウンロードケーブルを介して FPGA に転送するロード機構を備えており，転送終了後直ちに FPGA はマイクロプロセッサとして動作を開始できる．また，動作検証済みの構成データを専用の EPROM に書き込んでおくことで，コンピュータ入門教育用にスタンドアロンで動作できるワンボードマイコンとなる．
3. KITE マイクロプロセッサのメモリ空間として ROM と RAM をそれぞれ 2K ワードずつ実装している．ROM 領域には簡単なテストプログラムを用意しており，マイクロプロセッサの実装後，直ちに動作検証をすることができる．一方，RAM 領域にはパーソナルコンピュータからプログラムやデータを転送できるので，パーソナルコンピュータ上のアセンブラやシミュレータを利用して，プロセッサ動作の検証を詳細に行うことができる．
4. KITE マイクロプロセッサの I/O 空間として 16 ビットの入力ポート (トグルスイッチ) と 16 ビットの出力ポート (LED ディスプレイ) を実装している．
5. KITE マイクロプロセッサの動作確認を容易に行えるように，クロック周波数を 1MHz から 0.1Hz までの範囲 (16 段階) で変更できる．また，KITE マイクロプロセッサの機能である命令動作モードやクロック動作モードなどの動作モードの変更も可能である．

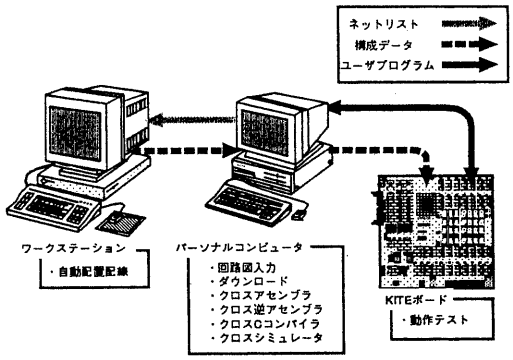


図 8: 開発支援環境

KITE マイクロプロセッサボードが上記の機能を有することで，学習者はマイクロプロセッサの内部状態や動作を視覚的に把握することが可能となり，計算機の動作原理の理解や実装時のデバッグに役立つ．また，実装デバイスとして FPGA を採用したことにより，必ずしもマイクロプロセッサ全体を一度に実装させる必要はなく，構成要素ごとに実装して動作確認をした後に全体の実装を行うことも可能である．

4.2 クロスソフトウェア

クロスソフトウェア環境としては，図8に示すように，クロスアセンブラ，クロス逆アセンブラ，クロス C コンパイラ，クロスシミュレータを用意しており，これらはパーソナルコンピュータ上で動作する．

クロスシミュレータは，以下の機能を持つ．

- KITE マイクロプロセッサのシミュレーションを行うことができる．シミュレーション実行時には，(1) レジスタの値を変更できる，(2) ブレークポイ

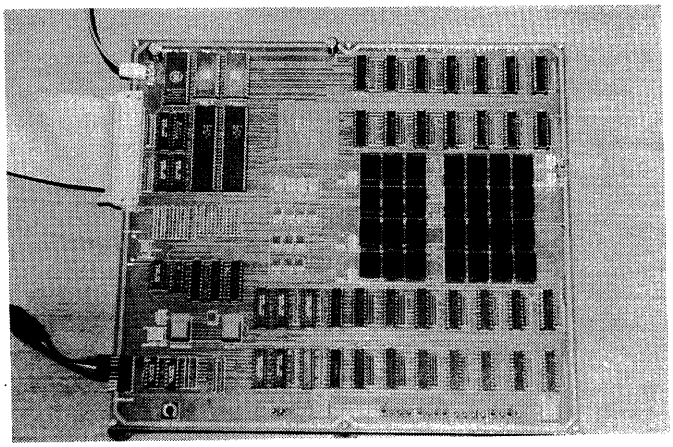


図 7: KITE マイクロプロセッサボード

ントが設定できる，(3)1 命令ごとのトレースができるなどの特長を持つ．

- コンピュータのディスク内に保管されているオブジェクトコードファイルとのデータ交換が可能である．
- ローダの機能を有しており，KITE マイクロプロセッサボード上の RAM 領域とデータ交換が可能である．

このクロスソフトウェア環境は，LSI 設計演習における回路のデバッグに役立つだけでなく，自らが設計したマイクロプロセッサを用いてアセンブリ言語や C 言語によるプログラミング演習，あるいは言語処理系の作成演習を行うことができる．

5 教育実験事例

本学では現在，LSI 技術の教育研究を支援するために設置されたマイクロ化総合技術センターの CAD システムを利用して，3 年次の学生を対象に KITE マイクロプロセッサの開発を学生実験の一テーマとして実施している．学生は 2 人一組で設計演習を遂行し，本年度は 1 回あたり 2 コマの計 6 回 (3 週間) の期間を通じて KITE マイクロプロセッサの開発を行う．ただし，6 回の実験時間ではマイクロプロセッサ全体をすべて設計するのは難しいため，今回は 1 バス構成のマイクロプロセッサにおける算術論理演算回路，命令デコーダ，シーケンサ等の基本構成要素の設計を主に行わせ，レジスタ等の構成要素についてはライブラリとして提供している．なお，本年度はカリキュラムの都合で暫定的な試行期間であり，来年度からはもっと長い実験期間を通じて KITE マイクロプロセッサ全体を設計し，さらにその言語処理系 (コンパイラ) まで開発する学生実験を実施する計画である．

本年度実施している KITE マイクロプロセッサの設計演習では，回路図入力による設計手法を採用している．ただし，表 2から分かるように，パーソナルコンピュータで自動配置配線を行っていては実験時間内に納まるほど短時間に実装を行うことが難しい．このため，負荷は軽いがインタラクティブな操作が主体となる回路図入力やデバッグについてはパーソナルコンピュータを利用する一方，負荷の重い自動配置配線にはワークステーションを利用している．このようにワークステーションとパーソナルコンピュータで役割分担を行っているが，図 9に示すように学内 LAN を利用することによって多人数教育の場合でも円滑に遂行できる学生実験環境を構築でき，コスト削減にも役立っている．また，学生実験利用に際しての学科間競合に対しては，パーソナルコンピュータを 12 台と 6 台の 2 つの教室に分け，2 学科が同時に利用申請した場合でも学生実験を実施できる体制を整えて解決を図っている．KITE マイクロプロセッサの開発を行っている実際の学生実験風景を図 10に示す．

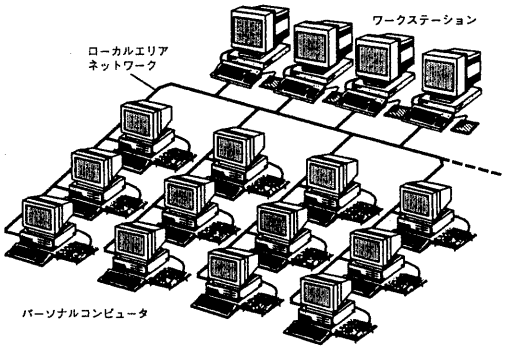


図 9: 学生実験環境の構成

6 おわりに

本稿では，書換え可能な FPGA を利用して，学習者が自らの手で設計，実装，動作検証まで行える教育用マイクロプロセッサ KITE とその開発支援環境について述べ，マイクロプロセッサの設計が学生実験の一テーマとして実施できることを示し，その設計教育事例を紹介した．設計手法としては回路図入力による設計とハードウェア記述言語による設計，あるいはその

図 10: 学生実験風景

組合せを利用することができ，教育目的や実験演習時間に合わせて選択することができる．

KITE マイクロプロセッサボードはディジタル回路の基本的構成要素をすべて含む簡素なアーキテクチャをその場で LSI として実装でき，ASIC 時代の設計教育として短時間で効果的な実験演習を実施できる．また，実装デバイスとして書換え可能な FPGA を採用したことで計算機構成要素の段階的開発ができる上，デバッグや改良のために何回でも設計のやり直しができるので，論理回路設計に経験が浅い初心者にも適している．さらに，特別な測定装置を用いなくとも，マイクロプロセッサ内部のデータの流れや各種レジスタの値などを外部から把握できるように可観測性を高めており，コンピュータの動作原理を実体験できる教材としてコンピュータ入門教育にも利用できる．

なお，ここで報告した KITE マイクロプロセッサは，OS 等のシステムソフトウェアを作成する際に必要不可欠な「割り込み機能」や「デュアルモード機能」を実装していない．そこで，これらの機能を実装して一貫した計算機工学教育が行えるように，さらに集積度の高い FPGA を用いた KITE マイクロプロセッサボードを現在開発中である．このボードは VHDL のように高レベルの機能記述が可能なハードウェア記述言語を利用した場合でも現行の KITE マイクロプロセッサの仕様ならばフルスペックで実装できる．

最後に，前述の設計サンプルを含む KITE マイクロプロセッサボードやクロスソフトウェアから成る開発支援環境を他の大学や高等専門学校などの教育機関にも利用できるように準備を進めており，教材として活用して頂ければ幸いである．

7 謝辞

本研究を遂行するにあたり，ご支援頂いた本学有田五次郎教授に感謝致します．また，日頃ご討論頂く有田・末吉研究室の諸氏に感謝の意を表す．

参考文献

[1] 情報処理学会：大学等における情報処理教育のための調査報告書 (1991).

[2] 末吉，船越，有田：FPGA 専用 CAD による論理回路設計教育の学生実験事例，電気関係学会九州支部連合大会講演論文集，No.1025(1991).

[3] 神原，安浦：計算機教育用マイクロプロセッサの開発とその応用 -集積回路技術を利用した情報工学実験-，情報処理学会誌，Vol.33, No.2, pp.118-127 (1992).

[4] 末吉，田中，船越，松尾，有田：書換え可能な LSI を用いた教育用マイクロプロセッサの開発，第 43 回情報処理学会全国大会講演論文集，2Q-11 (1991).

[5] 末吉，田中，柴村：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，情報処理学会研究報告，92-ARC-96-15, (1992).

[6] Xilinx, Inc.: *Programmable Gate Array Databook* (1992).

[7] IEEE, Inc.: *IEEE Standard VHDL Language Reference Manual* (1991).

[8] Data I/O,Corp.: *ABEL User Manual* (1990).

[9] Exemplar Logic, Inc.: *CORE User Manual* (1992).

[10] Vantage Analysis System, Inc.: Vantage Spreadsheet User Guide (1992).

[11] Mentor Graphics, Corp.: *AutoLogic User Inteface Reference Manual* (1992).

[12] Mentor Graphics, Corp.: *QuickSim II User's Manual* (1992).

[13] Data I/O, Corp.: *ABEL-FPGA User Manual* (1991).