計算機アーキテクチャ　101－17
（１９９３．　８．　２０）

並列オブジェクト指向トータルアーキテクチャA-ＮＥＴ
- PEの実装設計 -

岩本善行　吉永努　馬場敬信

宇都宮大学工学部

並列オブジェクト指向トータルアーキテクチャA-NETにおける要素プロセッサ（PE）のアーキテクチャおよび、そのハードウェア設計から、ファームウェア設計、実装設計について述べる。我々の研究室では並列オブジェクト指向言語A-NETLを設計し、この言語の専用機としてA-NET計算機を設計している。A-NET計算機はVLSI化された1,000台規模のノードプロセッサからなる高並列計算機の実現を前提とし、また、応用分野によって複数のネットワークトポロジを選択できる。本稿では、この様なマクロレベルのアーキテクチャにもとづいた、PEのハードウェア設計、ファームウェア設計、プリント配線基板の実装設計について述べる。

A Parallel Object-Oriented Total Architecture A-NET
- Implementation of PE -

Yoshiyuki Iwamoto　Tsutomu Yoshinaga　Takanobu Baba

Department of Information Science
Utsunomiya University

This paper describes the design and implementation of a processing element (PE) for a parallel object-oriented total architecture A-NET. The A-NET multicomputer is to contain about 1,000 nodes. The PE has the A-NETL language-oriented features, such as a high-level machine instruction set, a tagged architecture, and special message passing constructs. A hardware and firmware combined approach has been taken to support the features efficiently. The results of hardware implementation,such as the hardware debugger, printed circuit board design, and the use of building blocks, are also included in the paper.

1.はじめに

計算機の対象とする応用分野は従来の科学技術計算、数値処理から、言語理解、知識工学などの人工知能分野[1]や、各種のシミュレーション、画像処理などに広がりを見せている[2]。これらの分野では処理する情報量が多く、処理の内容自体が高度なものとなっており、計算機の高速化が不可欠である。

また、知識工学やシミュレーションのような応用分野における問題領域では、多くの場合、並列に処理可能な複数の独立した手続きを含む。このような処理を、並列性の考慮のなされていないプログラミング言語で記述すると、問題や処理の表現が本来のものとかけ離れたものとなってしまう。

オブジェクト指向では問題領域に現われるオブジェクトの持つ機能や知識を素直に表現することにより、理解の容易なプログラム記述を目指している[3]。また、並列オブジェクト指向計算モデルはメッセージパッシングを基本としているため、並列性の自然な記述に適している。

このようにオブジェクト指向は並列処理に適しており、他にもConcurrent-Smalltalk等に対してJ-MachineがMITで開発中である。我々の研究室では並列オブジェクト指向トータルアーキテクチャA-NET[4]に関する研究を行っている。A-NETでは並列オブジェクト指向言語A-NETL[5]を設計し、これを用いて応用プログラムの記述、およびワークステーション上でネットワークワイドなシミュレーションを行ってきた。

A-NET計算機の各ノードプロセッサ（以降ノード）[6]はユーザオブジェクトの実行単位となるメソッドを実行する要素プロセッサ（以降PE）と、メッセージ転送を行うルータの対からなる。各PEは局所メモリを持ち、PE-ルータ共有メモリを介して、他のノードのPEとメッセージ交換を行うため、A-NET計算機は粗結合型のマルチコンピュータに位置づけられる。また、A-NETではVLSI化された1000台規模のノードからなる高並列計算機[7]の実現を前提として設計を行っているため、PEの各要素はできるだけ簡潔なハードウェア構成に抑えている。このようなPEのアーキテクチャ上の特長として次の３点が挙げられる。

(1)高機能機械命令セット
(2)タグ付きアーキテクチャ
(3)特殊レジスタセット

本稿では、以上の方針のもとに設計したPEのアーキテクチャとそのハードウェア構成、ファームウェアの設計と評価について述べた後、その実装設計について述べる。

2.PEのアーキテクチャ

2.1 高機能命令セット

プロトタイプの実現を容易にするため、A-NETL指向の高機能命令セットを定義し、これをマイクロプログラムによって実現する。命令は、メッセージ送信や構造体データ操作命令など高機能命令セットを16種79命令定義する。このことにより、実行オブジェクトはコンパクトになり、オブジェクトロード時の負荷を軽減できる。また、命令形式は、機械命令によってオペランドの数や種類が異なるため、バイト境界を持つ1～16バイトの可変長命令とする。オペランド数は最大15個とし、機械命令コード中にその数を含む。メッセージ送信命令では引数をオペランドとして最大13個まで持つことを許している。

メッセージ受理に伴い頻繁に発生するコンテクストスイッチを高速化するため、オペランドで指定するデータはレジスタを使用せず、局所メモリ上に配置する。

プロトタイプの作成を前提としているため、マイクロプログラミング方式を採用し、命令セットの変更に柔軟な対応を可能としている。マイクロ命令は76ビット水平型とし、1マシンサイクル166nsec（30MHz、５フェイズ）の命令先取り制御を行なう。

2.2 タグ付きアーキテクチャ

A-NET計算機では動的にデータ型付けを行ない、GCやOS支援のためにタグ付きアーキテクチャを採用している。タグ部はデータフラグと下位32ビットのデータの型を示すデータ型コードからなる。データフラグとしては、並列に実行されるオブジェクト間の同期をとる未来フラグとGC支援や、構造体のループチェックを行なうフラグからなる。データ型としては、整数や浮動小数、オブジェクト識別子等の1語からなる、構造を持たないもの6種類と、配列やメソッド等の複数語からなる構造を持つもの7種類がある。

2.3 特殊レジスタセット

オブジェクトの読み込みにより、局所メモリへのオブジェクトの再配置が行なわれるが、この再

配置のコスト軽減のためオペランドのアドレス指定をベースアドレス方式、分岐先アドレス指定を相対アドレス方式とする。ベースアドレスとしては、一時変数ベースアドレス（TBA）、リテラルベースアドレス（LBA）、状態変数ベースアドレス（SBA）、メッセージ引数ベースアドレス（PBA）が用意されている。以前は、一時変数とメッセージ引数のオペランドは、TBAによって局所メモリ上のアドレスを指定していた。そのため、メッセージを受理し、共有メモリに書き込まれたメッセージ引数は、局所メモリへのコピーを必要としていた。このメッセージ引数を共有メモリから直接読み込めるようにし、データのコピーによるオーバーヘッドを軽減する。

コンテクストスイッチを高速化するためにこれらのレジスタ群を２セットもち、システムモード、ユーザモードの各状態に対応させる。

3.PEのハードウェア

3.1 設計方針

ハードウェアを設計するにあたり、次のような方針を立てた。

(1)メッセージ送受信コストの軽減

メソッドの実行に対するメッセージ送受信コストを軽減し、システム全体の性能向上を図る必要がある。

(2) VLSI指向の簡潔なハードウェア構成

PEは分散メモリ型のマルチプロセッサ・システムの処理要素であるため、メモリまで含めて１チップ化することが望まれる。

(3)タグ処理のオーバーヘッドの軽減

タグ付きアーキテクチャではデータ処理に先立って、データタグの内容を判別し、その結果によってそのデータに対応した処理を施す。このデータ処理に先立つタグ部の判定がオーバーヘッドとなることが予想されるため、これを軽減する必要がある。

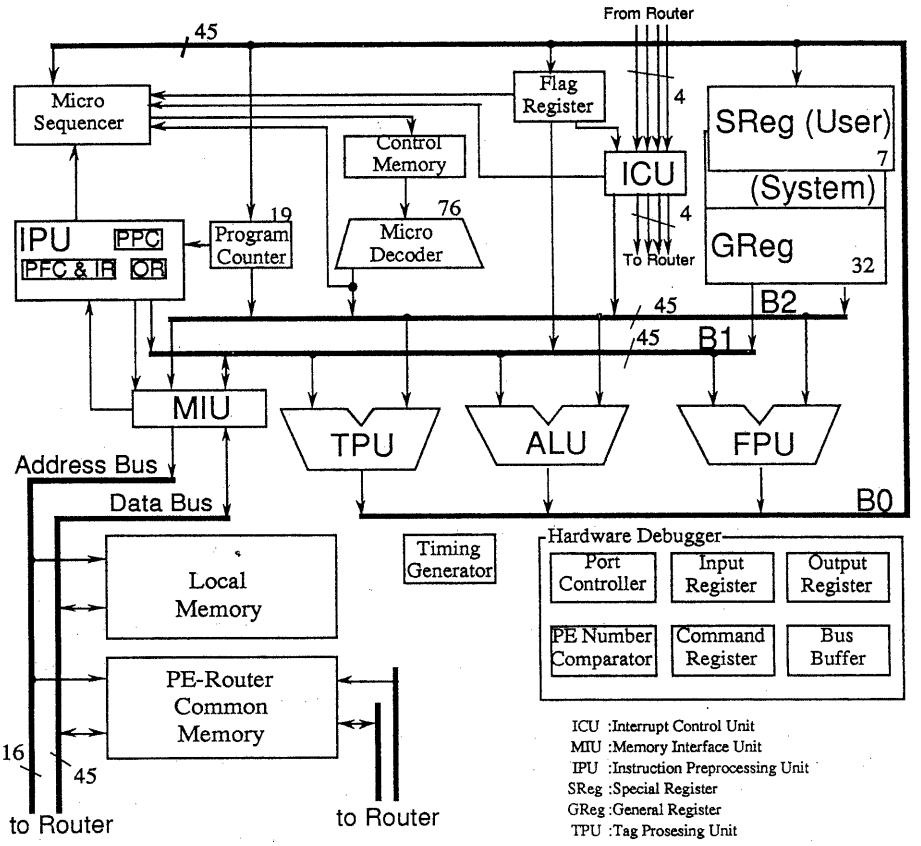

図１.PEのハードウェア構成

3.2 PEのハードウェア構成

以上の方針に基づいて設計したハードウェアの構成を図1に示す。命令プリフェッチを行なう命令前処理ユニット（IPU）、データのタグを処理するタグ処理ユニット（TPU）、システム用とユーザ用の2セットの特殊レジスタ群（SReg）、割り込み制御ユニット（ICU）、メモリインターフェイスユニット（MIU）などが特徴的なユニットとして挙げられる。その他にファームウェアでデータ処理を行なうための汎用レジスタ（GReg）、数値・論理演算ユニット（ALU）、浮動小数点演算ユニット（FPU）、機械命令およびデータが配置されるローカルメモリ（LM）、メッセージ送信時にPEから他ノードのPEに送るメッセージを格納するPE-RT間共有メモリ（CM）、ファームウェアの実行を制御する順序制御回路とマイクロプログラムの載るコントロールメモリ、マイクロデコーダからなる。

3.3 命令前処理ユニット

データはローカルメモリ上にワードアドレスによって配置され、そのアクセスは1語単位で行なわれる。これに対して機械命令はバイトアドレスによって指定される。そこで19ビットのプログラムカウンタ（PC）を用意し、その上位16ビットをワードアドレス、下位3ビットを1語中のフィールドを指定する5進カウンタとする。

命令前処理ユニットでは、ローカルメモリから機械命令を1語単位で読み込み、PCの上位16ビットのワードアドレスと下位3ビットのフィールド指定により1語中に埋め込まれた命令コードとオペランドの取り出しを行なう。また、タグ付きオペランドのタグ部の削除、符号拡張、ベースアドレス指定を行なう。

3.4 タグ処理ユニット

データはデータタグとデータ本体から構成される。タグ処理とデータ本体の処理を同時に行なうことによってタグ処理によるオーバーヘッドを軽減する。2つのデータのデータ型の比較や、どちらか一方のデータ型のチェックを行ないその結果をフラグレジスタに反映させるタグチェック回路と、タグ部をALUで処理するためにどちらか一方のタグをB0バスの下位8ビットに移動させるタグ移動回路から構成される。

3.5 レジスタファイル

PEにメッセージが到着すると割り込みによってOSのメッセージ受理メソッドが起動される。この時にユーザとシステム間で起こるコンテクストスイッチによる特殊レジスタの内容の退避および復帰によるオーバーヘッドを緩和するために、システム用とユーザ用の2セットの特殊レジスタ群（SReg）を用意する。これによりメッセージ受理割り込みによるユーザ実行イメージのコンテクストへの退避を不要とする。

3.6 割り込み処理ユニット

メッセージ受理に伴うRTからの外部割り込み、並列に実行されているオブジェクト間の同期をとるフューチャトラップや、データ処理時のエラーなどのトラップを支援する。外部割り込み要求は、機械命令境界において、割り込みが存在するかを反映するフラグをチェックすることにより認識される。これに対し、トラップは実行中に何らかのエラーが生じた場合、その機械命令の終了を待たず直ちに起動されなければならない。そこで機械命令実行中にエラーが発生したことを認識するために、そのエラー信号をもとにしてマイクロシーケンサに割り込みをかけ、トラップ処理を行なわせる方法をとる。このようにPEではマクロおよびマイクロレベルによって割り込み制御を行なう。

3.7 メモリインターフェイスユニット

ルータからPEへのメッセージやオブジェクトの転送は、RTがPEに割り込みをかけることによってメモリ領域を要求し、その確保された領域に対してルータがローカルメモリに直接データを書き込むことによって行なわれる。そのため、ローカルメモリに書き込むときはPE-ルータ間でバス調停が必要となる。そこでメモリインターフェイスユニットではバス調停およびメモリアクセス制御を行なう。また、命令前処理ユニットからの命令プリフェッチ要求に対しての機械命令の読み出しを行なう。

3.8 ハードウェアデバッガ

プロトタイプマシンのハードウェアデバッグや、PEへのマイクロプログラム転送、さらに各ノードへのオブジェクトプログラムの転送やメッセージ通信を行なうために以下のような方針をもとに設計した。

(1)ハードウェアデバッグ時に必要になると思われる、全PEの停止、およびリセットや、通信遅延時間を揃えることが必要となることから、ホスト-ノード間の通信距離を全て

等しくする[8]。

(2)ホスト-ノード間の通信の多くはオブジェクトプログラムの転送で占められ、ルータ-ルータ間の通信に対し、ホスト-ノード間の転送速度が実行性能に与える影響は少ないため高速転送に対応する必要はない。

(3)ホストは多数のノードと結合するため実装の複雑さを抑える工夫が必要である。数1,000台規模のノードからなる高並列計算機の実装を想定すると、通信線数をできる限り少なく抑えることが要求される。

(4)ハードウェアデバッグではバスやレジスタなどのPE内ファシリティに対する読み書き操作を行なうが、ルータが正常に動作している保証がないため、ルータを介さずホストから直接デバッグ操作を行なえるようにする。

以上の設計方針にもとづき、ハードウェアデバッガのホスト-PE間結合はバス方式を採用する。このことによってホスト-ノード間の通信は、上記のようなPEのハードウェアデバッグを行なう際にホストからのコマンド転送等を行なうホスト-PE間結合と、ホスト計算機から各ノードに対する静的なオブジェクトの割り付けやホスト-ノード間のメッセージ送受信を実現するホスト-ルータ間結合の2本のバスが設定される。ホスト-ノード間結合におけるこれら2つのデータ線は各ノード間でディジーチェーン化して実現され、また、これらデータ線のためのハンドシェーク信号は各ノードの信号の調停を行なう回路を設けて実現される。

4.ファームウェア

4.1 フィールド構成

図2にマイクロ命令のフィールド構成を示す。各バスを制御するバスフィールド、ALU,FPUでの演算の種類を制御する演算フィールド、メモリの読み書きを制御するメモリフィールド、TPUでのタグ処理を制御するタグフィールド、命令前処理ユニットを制御するIPUフィールド、マイクロプログラムの順序制御を行なうシーケンサを制御する制御フィールド、マイクロ分岐アドレスや、16ビットのリテラルを提供するリテラルフィールドから構成される。

4.2 マイクロプログラム作成支援環境

マイクロプログラム[9]の記述を容易にするために、マイクロ言語を定義し、アセンブラ、リンカを作成した。さらに、マイクロプログラムのデバッグのためにマイクロプログラムシミュレータを作成した。これは、ハードウェアが試作段階であるためにLSIなどの変更が起こりうるため、これらの変更に柔軟に対応するために各機能ユニットごとに独立させて実現した。また、機能的には、多くのマイクロステップを実行させる可能性があるため速度を重視し、デバッグ用のための複雑な機能は持たず、最小限の機能しか持たせていない。このシミュレータによって各機械命令を実際に実行して、デバッグ、性能の検証を行なった。

4.3 マイクロプログラムの特徴的処理

以下に、作成したマイクロプログラムについて特徴的な処理を述べる。

(1)可変長命令

メッセージ送信命令に使用される可変長命令は、その命令のオペランドの長さを命令コード中に含むが、この命令コードはシーケンサによって256方向分岐の一時分岐を行ない、そこからさらに実際の実行部へジャンプする。この命令コードは一時分岐のためのオフセットに使用されただけで、バスに乗ることはなく、よってレジスタ等に保存ができない。実行時にはオペランド長を参照する必要があるが、マイクロプログラムを小さくするために、オペランド長によって実行部を分けていない。このため、この一時分岐先でオペランドの長さを与え、実行部へ渡さなければならないが、このオペランド長を定数として与えるためのリテラルフィールドは、ここから各命令実行部への分岐命令によって使用しているため、オペランド長をバスフィールドと演算フィールドを用いた

フィールド名	バス				演算		メモリ	タグ				IPU	制御部		リテラル
フィールド	IB0	OB0	B1	B2	ALU	FPU	LM	TM	BS	DF	DT	IPU	TS	SEQ	LIT
ビット長	2	6	6	6	9	3	2	2	1	4	5	4	4	6	16
使用効率	-	58.1	61.0	36.9	45.4	7.9	16.5	-	-	7.8	10.4	0.7	15.8	55.2	63.9

(全76bits)

図2.マイクロ命令のフィールド構成と静的使用効率

ニーモニック	Min	Max	ニーモニック	Min	Typ	Max
MV	14	16	SHIFT	22		26
ADD	25	32	STRUCT	17		70～
MULT	25	48	BRANCH	13		51
DIV	262	699	NOP		5	

図3　マシンサイクル数

レジスタ同士の演算によって算出している。

(2)演算命令

ALU、FPUは並列に動作可能であるため、add、subは同時に演算を行ない、TPUのタグ処理の結果に基づいてどちらかを選択してバスに流すようにする。

乗算は、浮動小数点数に対してはFPUにその機能があり、2サイクルで算出可能であるのに対し、整数に対してはコストを削減するためにハードウェア乗算器を使用せず、ALUにおいてループを用いて演算している。この結果、浮動小数点演算の方が整数演算よりも早く終了するという逆転現象が起きている。

除算は、ALU、FPU共にその機能がないため、ALUを用いて引き戻し法[10]によって実現している。

4.4 評価

このようにして作成されたマイクロプログラムを評価する基準として、そのサイズ、各種ハードウェア資源の使用効率、各機械命令を実行するために必要なマシンサイクル数、および評価用プログラムを作成した。

全マイクロプログラムサイズは、2,699ワードで、そのうち実際に命令が置かれているサイズは2,491ワード（93.3%）である。これは、さらにハードウェアデバッグ用のプログラムを付加する予定であるが、試作機では制御メモリに8kワードを用意していることから、制御メモリを減らして、コストの削減が可能となる。

次に、各種資源の静的使用効率をマイクロフィールドのサイズの使用効率から求めたものを前掲の図2に示す。これらの値はマイクロ命令を最適化する場合に有効であり、各フィールド中で最大値をとるリテラルフィールドが、先の可変長命令の例からも、ネックになると思われる。

マシンサイクル数を算出したものの一部を図3に示す。マシンサイクル数はPEの性能を規定する上で重要なものであるが、高機能命令を実現したことや、最適化がまだ十分でないために、1命令におけるマシンサイクル数は多い。

最後に、評価用プログラムであるが、例として4色で地図を塗り分ける問題を取り上げた。この評価用プログラムを機械命令で記述し、実行した際のPEの性能は0.2MIPSであるという結果を得た。この主な要因として、命令の定義レベルが高いこと、また、この評価用プログラムでは、特にマシンサイクル数の多い構造体操作命令を多用していることなどが挙げられる。

5.PEの実装設計

5.1 主要ブロックの実装

PEのハードウェアはマイクロプログラム制御であること、タグ部を除くデータの処理部が32ビットを扱えることから、シーケンサ、ALU、FPU、レジスタファイルをAMD社のAm29300シリーズのビルディングブロックを用いて設計した。また、その他使用した部品は表1のとおりである。Pin Grid Array(PGA)は、上記のALU、FPUを含む25個であるが、そのうち15個が、Complex Plogrammable Logic Device（CPLD）である。また、DIPは106個で、そのうちPLDが9個を占める。これまでPLDは35個を使用していたが、これらの多くを機能ブロック単位でCPLDに集積化したためであり、これによって基板サイズを小さくしている。

表1　使用コンポーネント

PGA	25
CPLD	15
DIP	106
PLD	9
SIP	7
LED	30
7Segment LED	9
コネクタ	6
コンデンサ	240

5.2 デバッグ支援

先のマイクロプログラムシミュレータを作成したときの経験をもとに、デバッグ時に必要となるであろう、Flag Registerの各ビット、Timing Generatorのフェイズ、動作状態（Run、Halt、Wait）、演算を実行しているブロック（ALU、FPU、TPU）などの状態を表わす信号線にLEDを取付けた。また、Micro Sequencerから出力されているマイクロアドレス、機械命令の実行アドレス（Program Counter）を表示するのに7Segment LEDを付けた。

5.3 基板外とのインターフェイス

ルータとのインターフェイスは、アドレスバス（DMA用16Bit、共有メモリ用13Bit）、データバス（パリティ5Bitを含む45Bit×2）、割り込み（リクエスト、アクノリッジ各4本）、およびそれらの制御通信を行なうため、160ピンと140ピンのコネクタを各1つ使用する。また、ホスト（VME）との通信のためには、40ピンのフラットケーブル用コネクタを使用している。ルータとのコネクタのピンのなかで実際に使用しているピン数は136ピンで、ホスト用コネクタにおいて実際に使用しているピンは15ピンである。

5.4 基板概要

プリント配線基板の諸値を表2に示す。プリント基板の大きさは50cm×40cmで、ピン間を3本とする。さらに、基板の縮小と信頼性を上げるために自動配線は使用せず、各信号線を1本ずつマニュアルで配線する。基板はデバッグ時のパターンカットを容易にするため、4層基板とし、内層をGND、VCC面に使用、部品面、半田面に配線する。これを両面レジストで覆い、部品面にシルク印刷を施す。

表2　基板諸元値

Pin間	3本
Track幅	6mil
Viaサイズ	20mil
Via Hole サイズ	14mil
Track間	6mil
Track/Via間	7mil
Via間	7mil

1mil = 1/1000inch

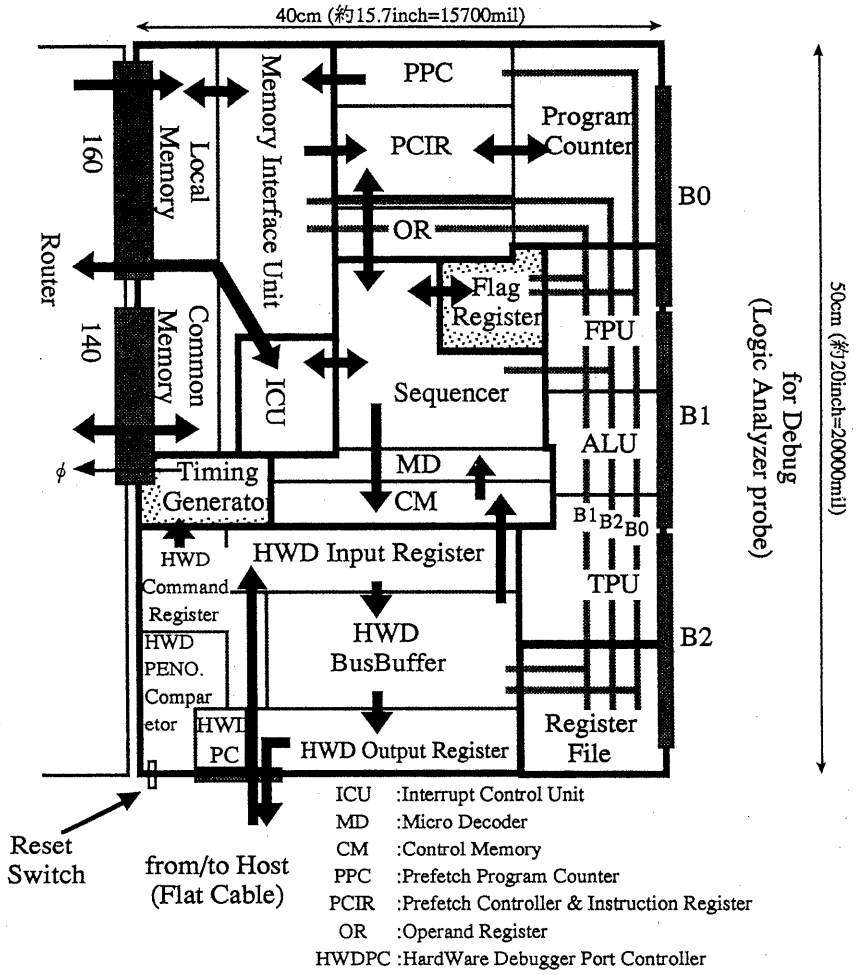

図4　基板レイアウト

5.5 各ブロックの配置と配線

実際のプリント基板における各ブロックの配置を図4に示す。矢印は主なデータの流れを表わし、B1、B2、B0は45ビットのバスを表わす。基本的には回路図のルートの配置をそのまま基板上に配置し、バスの接続状況や各ブロック間の配線ができるだけ短くなるように工夫した。

PE、ルータの両方に送られる基本クロック（30MHz）は、遅延を最小に抑えるために、両者の中央に配置する。基板の右端にはハードウェアデバッグ時に、バスをモニタしたり、必要なデータを流すため使用するロジックアナライザのためのプローブを用意した。

5.6 開発環境

PEの回路図設計、基板の配置配線はNEC PC-9801DA上で行なっている。CADシステムとしては、回路図設計とその動作検証に米OrCAD System社の回路作成ツールOrCADを使用している。

論理回路設計には論理機能の記述可能なPLDおよびMACHを多用するため、AMD社のPALASMを用いた。さらに、基板の配置配線には豪Protel Technology社のProtel-Autotraxを使用する。ただし、OrCADの出力したネットリストは直接Autotraxの入力としては使用できないため、コンポーネント名をライブラリ名に変換するために簡単な変換ツールを作成した。

印刷は、Autotraxに付属のTraxplotを使用し、PostScriptファイルをワークステーションからプリンタに出力している。

6.おわりに

現在、プリント基板の配線を行なっており、約50%程度まで終了している。

PEのプリント配線基板製作が完了した後、ホスト上に構築された対話型のハードウェアデバッガにより、ホスト-ノード間結合を介してPEのハードウェアデバッグを行なうと同時に、PEのデータ処理やコンテクスト処理におけるデータ収拾を行なう。さらに、2台のノードからなるA-NET計算機を製作し、ノード間メッセージ通信や、ノード間オブジェクト転送に関する検証を進める。さらに、1,000台規模のノードプロセッサからなる高並列計算機の実現に向けて研究開発を進める。

謝辞

本研究は、一部文部省科研費（試験研究（B）課題番号04555077、重点領域研究（超並列）課題番号05219203、奨励研究課題番号05780226）の補助を受けて行なっている。特に、試験研究についてご協力いただいている東大米澤明憲教授、日本電気小池誠彦研究部長、東芝前田明部長、日立坂東忠秋部長、当大学青木恭太助教授、熊谷毅助教授に感謝する。また、実装に当たりご協力いただいている日本AMD吉沢俊介部長、Larry Hollatz AMD副社長、東芝斉藤光男研究主幹、岩佐繁明研究主務、大昌電子半田芳男工場長、橋村隆夫課長に感謝する。

【参考文献】

[1] A.Yonezawa: AI Parallelism and Programming, Proc. IFIP 86, pp.111-113 (1986).

[2] 小池 他: 階層レベル並列推論シミュレーションマシンMAN-YO, 信学論, Vol. J71-D, No.8, pp.1391-1398(1988)

[3] 米澤明憲:オブジェクト指向型プログラミングについて, コンピュータソフトウェア, Vol.1 No.1, pp.29-41(1984)

[4] T.Baba et al.: A Parallel Object-Oriented Total Architecture; A-NET, Proc. Super Computing '91, pp.278-285(1990)

[5] T.Yoshinaga and T.Baba: A Parallel Object-Oriented Language A-NETL and Its Programming Environment, Proc. COMPSAC '91, pp.189-196(1991)

[6] 吉永 他: A-NET計算機のノードプロセッサとその実行方式, JSPP '91, pp189-196(1991).

[7] 馬場敬信:超並列マシンへの道,情報処理学会誌,Vol.32,No.4,pp348-364(1991)

[8] 大倉義典: 並列オブジェクト指向計算機A-NETホスト-ノード間における通信機能の実現, 宇都宮大学工学部情報工学科卒業論文, 91-05(1992)

[9] 岩本善行: A-NET計算機のPEにおけるマイクロプログラムの作成とその評価 宇都宮大学工学部情報工学科卒業論文 91-04(1992)

[10] 馬場敬信: マイクロプログラミング, 昭晃堂(1985)

[11] 寺岡孝司:並列オブジェクト指向計算機における要素プロセッサのアーキテクチャ, 宇都宮大学大学院情報工学専攻修士論文, 91-03(1992)

[12] T.Baba:Microprogrammable Parallel Computer -MUNAP and Its Applications-,MIT Press(1987)