計算機アーキテクチャ 87－4
（1991. 3. 11）

並列オブジェクト指向トータルアーキテクチャ A-NETの要素プロセッサ

鈴木 充　　吉永 努　　馬場 敬信

宇都宮大学 工学部 情報工学科

A-NET 計算機の要素プロセッサは，並列オブジェクト指向言語 A-NETLの効率的な実行を目的として設計され，(1) メッセージ送受信などの高機能命令の実装，(2) 動的データ型付けやメッセージ駆動方式のためのハードウェア支援，(3) コンテクストチェンジ時のオーバヘッド軽減のためのレジスタ構造，などの特徴を持つ．具体的には，可変長命令の切り出しとベースアドレス方式のオペランドを処理するための命令前処理ユニット，タグ付きデータ構造と未来型メッセージによる実行モデルを支援するためのタグ処理ユニット，メッセージ受理時にルータとのメモリアクセスを制御するメモリインタフェースユニット，ルータからの外部割り込みと同期のための内部割り込みなどを扱う割り込み制御ユニット，メッセージ割り込み時にユーザの実行イメージを退避せずに済ますための２セットの特殊レジスタセットなどを設けている．

The Processing Element for the Parallel Object-Oriented Total Architecture A-NET

Mitsuru Suzuki, Tsutomu Yoshinaga, and Takanobu Baba

Department of Information Science, Faculty of Engineering,

Utsunomiya University

A processing element(PE) has been designed for the A-NET multicomputer system. It executes programs, described in a parallel object-oriented language A-NETL. The major characteristics are as follows: (1) it has a high-level machine instruction set such as messege sending and receiving capabilities,(2) it supports a dynamic data typing and message-driven mechanism by the hardware, (3) it realizes high-speed context switching to decrease the overhead with message exchanges. The special hardware units include an *instruction preprocessing unit* for a variable length instruction access and base addressing mechanism, a *tag processing unit* for a tagged data structure and future-type message passing, a *memory interface unit* for a common memory access between the PE and a router, an *interrupt control unit* for external interruptions, caused by the arrival of the messages, and internal interruptions for syncronization, and *two sets of special registers* for a high-speed context change.

1．はじめに

我々は、並列オブジェクト指向を核概念とし、応用分野、プログラミング言語、計算機の3者を統合的な立場から考慮したトータルアーキテクチャA-NETの研究を行っている[1,2]。これまで、プログラミング言語として、並列オブジェクト指向言語A-NETL[3]の開発を行い、その処理系がほぼ完成している。

A-NET計算機の設計思想として、次の2点が挙げられる。

(1) A-NETL専用の言語指向計算機

オブジェクト指向アーキテクチャによって、A-NETLの効率的実行を狙う。

(2) 中粒度、高並列

多数プロセッサによる並列処理によって、高性能化を図る。また、マルチコンピュータシステム[4,5]においては、要素プロセッサの1チップ化が望まれる。

本稿では、以上の設計思想を反映した、A-NET計算機の要素プロセッサ（PE）のアーキテクチャ[6,7]について説明する。

2．PEの命令セットアーキテクチャ

2.1 設計方針

マクロレベルアーキテクチャの設計方針として、A-NET計算機がA-NETL専用、かつPEがマルチコンピュータシステムの処理要素であることから、以下の設計方針が挙げられる。

(1) 高機能な機械命令セット

PEは高並列を目指したマルチコンピュータシステムの処理要素であるので、ローカルメモリサイズを大きく取ることは望ましくない。また、処理過程において、オブジェクトの動的なノード間転送が行われるため、そのコストを抑えることは全体の処理性能の向上には不可欠である。そこで、A-NETL指向の高機能な命令セットを定義し、オブジェクトをコンパクトな機械語コードに翻訳する。具体的には、A-NETLのプリミティブメソッドの多くを、それぞれ1つの機械命令によって実現する。この結果、命令によってオペランドの数が異なるため、機械命令はバイト単位の可変長命令とする。

(2) タグ付きアーキテクチャ

A-NETLでは、変数の型は動的に決定される。そこで、データ型コード、並列に実行されるオブジェクト間の同期をとるためのフラグ、およびガーベジコレクション（GC）を効率よく行うためのフラグなどをタグとしてまとめ、各データに付加する。

(3) ローカルメモリ中心の処理

メッセージ受理に伴って頻繁に発生するコンテクスト・チェンジを高速化することが必要となる[8,9]。また、リストなどの構造体データをすべてレジスタ上に持たせることは困難である。そこで、機械命令レベルではレジスタを使用せず、オペランドで指定するデータはローカルメモリ上に置くものとする。

(4) 相対アドレッシング

オブジェクトの動的生成にともなって、ローカルメモリへのオブジェクトのロード、再配置が行われる。この時のコストを軽減するために、オペランドのアドレス指定をベースアドレス方式とし、分岐先アドレス指定を相対アドレス方式とする。

2.2 機械命令セット

上記の設計方針に沿って定義された機械命令は、17種類81命令からなる。

特徴的な命令としては、メッセージ送受信命令、構造体操作命令、コンテクスト処理命令などが定義されている。各命令は、1バイト長の命令コードと複数のオペランドから構成される。多くの命令は、2つ、または3つのオペランドを持つが、メッセージ送信命令では引数をオペランドで指定可能とし、最大15個のオペランドを許している。

オペランドは、8、あるいは16ビット長であり、それぞれ、タグ付き、タグ無しのものがある。図1にタグ付きオペランド形式を示す。タグ付きオペランドの場合、上位2ビットによって下位の6／14ビットのデータが、即値か、あるい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）の3つのいずれかのベースアドレスからのオフセット値であるのかを指定する。この下位6／14ビットのデータは、即値を示す場合は符号

付き整数として扱われ、ベースアドレスからのオフセット値を示す場合は符号なし整数として扱われる。

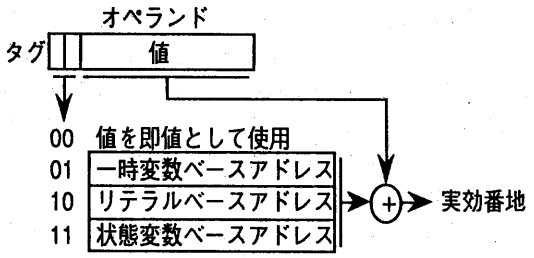


図1　タグ付きオペランド形式

2.3 データ構造

図2に、(a) 1ワードからなる構造を持たないデータと、(b) 複数ワードからなる構造を持つデータの例を示す。ここでは、構造を持つデータの例としてリストを示す。

構造を持たないデータとしては整数、浮動小数点数、真偽値、オブジェクト識別子、セレクタ、NULL（未定義値）の6種類、構造を持つデータとしては、文字列、配列、リスト、辞書、メソッド、コンテクスト、メッセージの7種類を用意している。

タグの内訳は、並列に実行されるオブジェクト間の同期をとるためのフューチャフラグ（f）、コピー型GCを支援するためのニューエリアフラグ（n）、GCおよびメッセージ送信において構造体のループチェックを行うためのマークフラグ（m）、下位32ビットのデータ値の型を示すデータ型コード（dt）、およびGC用のコピーフラグ（c）となっている。

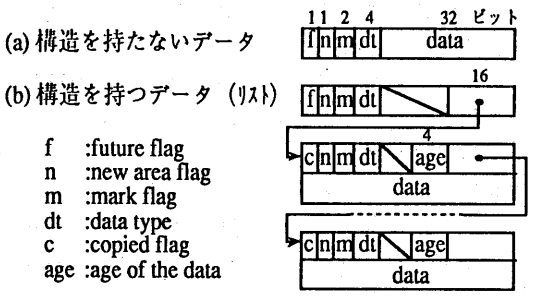


図2　データ構造

2.4 割り込み制御

PEの実行制御において、ルータ（RT）がPEのローカルメモリへ、メッセージやオブジェクトを転送したことを通知するために外部割り込みを定義し、フューチャトラップや機械命令実行時のエラー処理を行うために内部割り込みを定義する。

外部割り込みとしては現在表1に示す4種類が定義されており、割り込みのレベル数はノンマスカブルとマスカブルの2レベルを定義する。

表1　外部割り込み

ノンマスカブル外部割り込み
・OSの初期化 ・ユーザオブジェクトの割付 ・GCの起動
マスカブル外部割り込み
・メッセージ受理

3. PEのハードウェア構成

3.1 設計方針

(1) VLSI指向のシンプルなハードウェア構成

PEはマルチコンピュータシステムの要素プロセッサであるため、1チップ化することが望まれる。

(2) タグ付きアーキテクチャのオーバヘッド軽減

タグ付きアーキテクチャでは、データ処理を行う時にタグの内容を判定し、その結果によって、データに対する処理内容を変更する。したがって、データ処理に先立つタグの判定がオーバヘッドとなるため、これを軽減する必要がある。

(3) メッセージ送受信コストの軽減

マルチコンピュータシステム全体での性能の向上を図るためには、プロセスの実行に対するメッセージ送受信コストを軽減する必要がある。

(4) メッセージ駆動と同期処理の支援

PEはローカルメモリへのメッセージの到着をRTから知らされなければならない。また、並列に実行されるオブジェクト間の同期はフューチャトラップによって行われる。そこで、これらを効率的に実現する必要がある。

3.2 全体構成とその特徴

(1) 全体構成

PEのハードウェア構成を図3に示す。PEの構成ユニットには、特徴的なユニットとして、命令前処理ユニット（IPU）、タグ処理ユニット（TPU）、2セットの特殊レジスタ（SReg）、割り込み制御ユニット（ICU）、メモリインタフェースユニット（MIU）がある。その他のユニットとしては、ファームウェアでデータ処理を行うための汎用レジスタ（GReg）、ALU、FPU、および、ローカルメモリ、PE-RT間共有メモリ、そして、それらの構成ユニットをマイクロプログラムによって制御するマイクロシーケンサ、制御メモリ、マイクロデコーダなどがある。

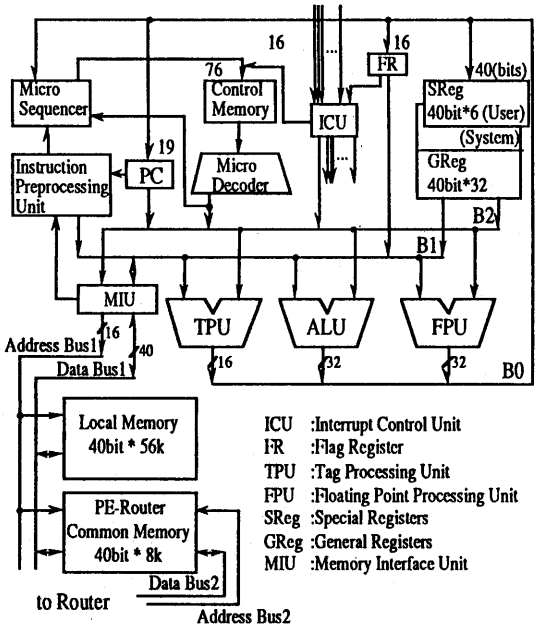


図3　PEのハードウェア構成

(2) 特徴的なハードウェア機構

PEのハードウェアを構成するにあたって、3.1の設計方針に基づいて次の機構を設ける。

(a) 命令前処理ユニット（IPU）

データはメモリ上にワードアドレスによって配置され、そのアクセスはワード単位で行われる。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）の上位16ビットをワードアドレス、下位3ビットをワード中のフィールドを指定する5進数値として、19ビットのバイトアドレスを示すものとする。

命令前処理ユニットは、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って、1語中に埋め込まれている命令コードとオペランドの取り出しを行う。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などを行う。

命令前処理ユニットの構成は、図4に示すように、2語の命令レジスタ（IR）、オペランド処理回路（OPL）、および命令プリフェッチコントローラ（PFC）からなる。2語のIRは、ローカルメモリから1語単位で読み込まれた機械命令を格納しており、PCのバイトアドレスによって示される8ビット長の命令コードや、8あるいは16ビット長のオペランドの取り出しを行う。オペランド処理回路（OPL）は、IRより取り出されたデータが命令コードの場合、それを命令コードレジスタ（OCR）に格納し、マイクロシーケンサに送る。タグ付きオペランドの場合は、上位2ビットのデータをタグレジスタ（TR）に格納し、マイクロシーケンサへの入力、および特殊レジスタへのベースアドレス指定として使用する。そして、タグ部の削除、および32ビット長への符号拡張を行いオペランドレジスタ（OCR）に格納する。命令プリフェッチコントローラ（PFC）は、ローカルメモリからIRへの命令の読み出しを自動的に行い、命令取り出し時にIRが空であることを防ぐ。

(b) タグ処理ユニット（TPU）

先に説明したように、データはタグと値データから構成されている。そこで、タグ処理と値データ処理を同時に行うことによって、タグ処理のオーバヘッドを軽減する。例えば、整数か浮動小数点数か分からない、2つのデータを加算する場合、TPUでタグ中のフラグチェックと型識別子の比較を行い、それと平行してALUとFPUで加算を行う。このとき、FPUの演算速度はALUよりも遅く、ALUでの演算とレジスタへの結果の格納は同じサイクル中で行えるのに対して、FPUでの演算結果のレジスタへの格納は、演算を行ったサイク

ルの次のサイクルで行われる。このため、まず、ALUの演算結果をレジスタに格納する。そして、2つのデータの型が一致し、それが整数型の場合はそこで処理を終了し、浮動小数点数の場合は次のサイクルでFPUの加算結果を格納する。一致しなかった場合は、整数型データをFPUで浮動小数点数型データに変換し、再び加算を行う。

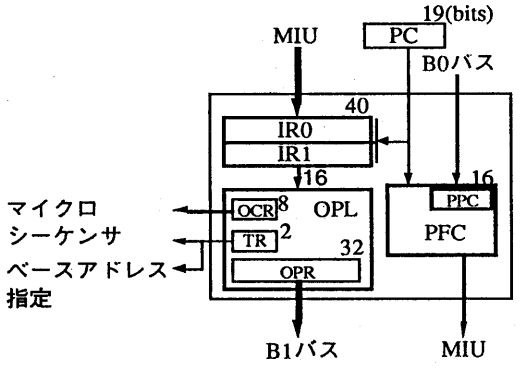
タグ処理ユニットは、図5に示すように、タグチェック&比較回路とデータ移動回路から構成される。タグチェック&比較回路は、図5(a)に示すように、2つのデータ（B1バス、B2バス）のデータ型の比較、あるいは一方（B1あるいはB2）のデータ型のチェックを行い、その結果をフラグレジスタのDTフラグに反映させる。同様にデータフラグのチェックあるいは比較を行い、その結果をフラグレジスタのDFフラグに反映させる。また、一方のデータフラグのセット、あるいはリセットを行い、B0バスの上位4ビットに出力する。

図5(b)にデータ移動回路を示す。この回路では、タグデータをALUで処理する必要があるため、B1あるいはB2のタグデータをB0の下位に移動させる。また、処理されたデータをタグの位置に戻すため、B1あるいはB2の下位8ビットのデータをB0のタグの位置に出力する。また、B1あるいはB2のタグをB0のタグの位置に出力する。

(c) 2セットの特殊レジスタ

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、ユーザとシステム間で起こるコンテクスト・チェンジによる特殊レジスタの内容の退避・復帰のオーバ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した。

特殊レジスタは、図6に示すように、メモリ上のメッセージやコンテクストを指し示すためのレジスタ（message/context）、および3つのベースアドレス（TBA、LBA、SBA）を保持するレジスタなどから構成される。この2セットのレジスタ群はバンク化されており、システム用とユーザ用



IR　：命令レジスタ
OPL：オペランド処理回路
OCR：オペコードレジスタ
TR　：タグレジスタ
OPR：オペランドレジスタ
PFC：プリフェッチコントローラ
PPC：プリフェッチ
　　プログラムカウンタ
MIU：メモリインタフェースユニット
PC　：プログラムカウンタ

図4　命令前処理ユニット（IPU）

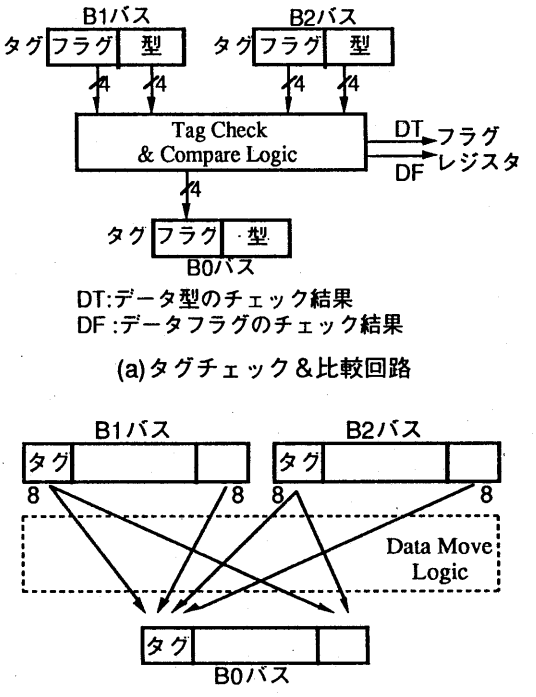


図5　タグ処理ユニット（TPU）

レジスタの切り替えは、PEの実行モードがシステムモードであるのかユーザモードであるのかによって自動的に行われる。

(d) 割り込み制御ユニット（ICU）

メッセージ受理に伴うRTからの外部割り込み、および並列に実行されるオブジェクト間の同期機構であるフューチャトラップをサポートする。

割り込み制御ユニットは、図7に示すように、(a) 割り込みコントローラと(b) マイクロ割り込みコントローラの2つの回路によって構成される。割り込みコントローラは、RTからの外部割り込みをサポートするものであり、RTからの割り込み要求に対して、割り込みマスクによってマスクされない有効な割り込みが存在すると、それを割り込み要求フラグとしてフラグレジスタに反映させる。マイクロシーケンサでは、機械命令境界でそれを受け付けることによって割り込み処理を行う。このとき、受理された割り込み番号は割り込み番号レジスタ（INR）に格納される。マイクロ割り込みコントローラは、内部割り込みをサポートするものであり、フューチャフラグのチェック結果や各構成ユニットからのエラー信号をマイクロ割り込み要求信号入力として、マイクロシーケンサに割り込みをかける。それと同時に、受理したマイクロ割り込み要求に対応するマイクロ割り込みベクタも送る。このように、内部割り込みは、マイクロシーケンサへの割り込みとして実現する。

(e)メモリインタフェースユニット（MIU）

RTからPEへのメッセージやオブジェクトの転送は、RT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われる。このため、RT-PE間でバスアービトレーションが必要となる。そこでメモリインタフェースユニットでは、バスアービトレーション、およびメモリアクセス制御を行う。

メモリインタフェースユニットは、図8に示すように、バスアービタと、メモリアクセスコントローラから構成される。バスアービタは、PE-RT間でのローカルメモリアクセスにおいて、バスの調停を行う。通常、バスマスタはPEであり、RTがローカルメモリをアクセスする場合は、RTがバスリクエストを行い、それに対してPEがバス

システム用レジスタ

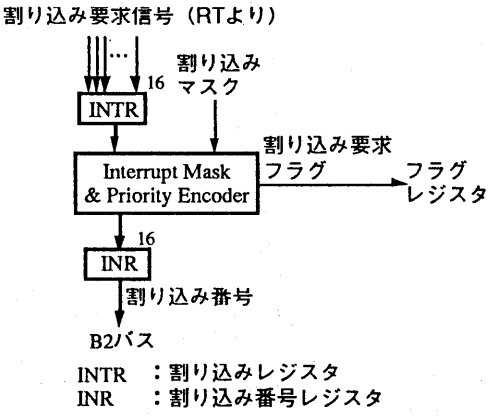
8	32 (bits)	
ﾀｸﾞ	message/context	
0	S,M,Pr	PC
0		返答先
6	size	TBA
6	size	LBA
6	size	SBA

ユーザ用レジスタ

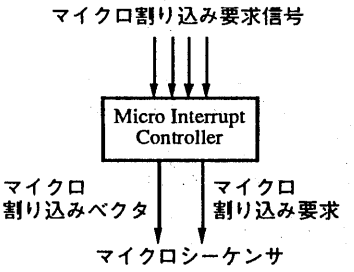
8	32 (bits)	
ﾀｸﾞ	message/context	
0	C,S,M,Pr	PC
8	履歴リスト	
6	size	TBA
6	size	LBA
6	size	SBA

C　：コンテクストの状態
S　：簡易コンテクスト
M　：実行モード（システム、ユーザ）
Pr　：プライオリティ
TBA：一時変数ベースアドレス
LBA：リテラルベースアドレス
SBA：状態変数ベースアドレス

図6　特殊レジスタ（SReg）



(a)割り込みコントローラ



(b)マイクロ割り込みコントローラ

図7　割り込み制御ユニット（ICU）

を開放できる状態であればバスグラントによってRTにバスを開放することを知らせる。RTは一連のメモリアクセス作業を終了するとバスリクエストを解除し、バスの使用権を放棄する。メモリアクセスコントローラは、PEのメモリアクセスを制御する。PEでは、メモリをアクセスする場合、B2バスにアドレス（、B1バスにデータ）を乗せ、このユニットにメモリ読み出し（書き込み）サイクルの開始を示すマイクロ命令を与えることにより、メモリの読み出し（書き込み）を行う。また、命令前処理ユニットからの命令フェッチ要求に対して、機械命令の読み出しを行う。

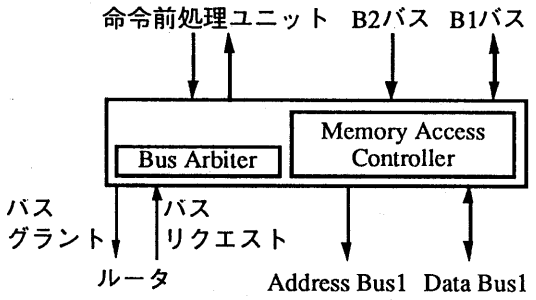


図8　メモリインタフェースユニット（MIU）

4．PEの制御方式

PEは、マイクロプログラム制御方式により制御する。マイクロプログラム制御方式は、シンプルなハードウェア構成を持つ計算機で、高機能な機械命令を実現するのに有効である。また、実験研究用の計算機として、ファームウェア、ハードウェアレベルの変更を容易にすることに有効である。

4.1 マイクロ命令定義

マイクロ命令は1語76ビット水平型で、15のフィールドからなる。

以下に各フィールドについて簡単に説明する。

1. バスフィールド（IB0 OB0 B1 B2）

フィールド長：20ビット

B0へデータを入出力するユニットの制御と、B1とB2へのデータを出力するユニットを制御する。

2. 演算フィールド（ALU FPU）

フィールド長：12ビット

B1、B2上のデータに対し、ALUあるいはFPUでどのような演算を行うのかを制御する。

3. タグ処理フィールド（TM BS DF DT）

フィールド長：12ビット

タグ処理ユニットの制御を行い、B1、B2上のタグのフラグチェックや、データ型識別子のチェック、比較などを行う。

4. メモリフィールド（LM）

フィールド長：2ビット

メモリインタフェースを制御し、メモリの読み出し、書き込みサイクルの開始を行う。

5. 命令前処理フィールド（IPU）

フィールド長：4ビット

IPUを制御し、IPU内の命令レジスタから、命令コードやオペランドを取り出し、シーケンサあるいはB1バスへの出力を行う。

6. 順序制御フィールド（TS SEQ）

フィールド長：10ビット

TSフィールドはマイクロシーケンサの条件入力の選択を制御する。

SEQフィールドはマイクロプログラムの順序制御を行う。

7. リテラルフィールド（LIT）

フィールド長：16ビット

分岐アドレス指定や、データ処理に使用する定数としての16ビットのリテラルを与える。

4.2 タイミング制御

1マシンサイクルは125nsで、25nsの5つのフェーズからなる。

マイクロ命令の先読みを行う二段のパイプライン制御によって、マイクロ命令読み出しのオーバヘッドを削減している。各処理ユニットは基本的に図9に示すタイミングに従って動作する。

この制御の流れを、斜線で塗られた部分に従って説明する。（i+1）サイクルの実行は、iサイクルのマイクロシーケンサの実行によって得られた、マイクロアドレスによる制御記憶からのマイクロ命令の読み出しに始まる。そして、読み出したマイクロ命令をマイクロデコーダによってデコードした制御信号によって（i+1）サイクルでの各ユニットの制御を行う。その制御の内容は、マイクロシーケンサによる次のマイクロアドレスの決定、各種レジスタからのデータの読み出し、

TPU、ALU、FPUでのデータ処理、レジスタへのデータ処理結果の書き込み、およびIPUによる命令コード処理とオペランド処理などである。

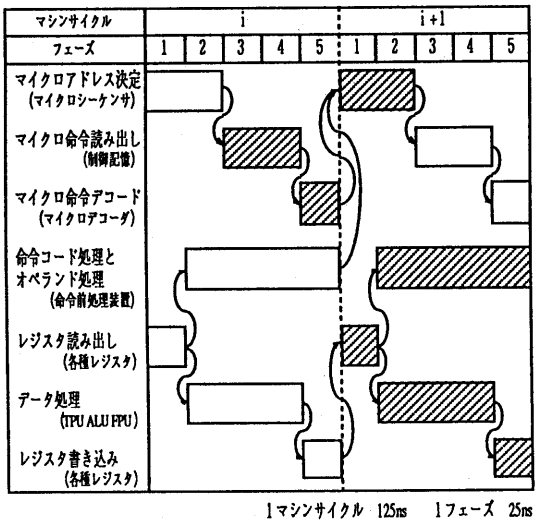


図9　タイミング

5. まとめ

PEでは、並列オブジェクト指向計算機の要素プロセッサとして、高機能な命令セットを定義し、それに対して命令前処理ユニットを装備した。また、データとタグの同時処理、コンテクスト・チェンジの高速化、メッセージ駆動と同期機構のサポート、PE-RT間のメモリアクセス制御のために、それぞれ、タグ処理ユニット、2セットの特殊レジスタ、割り込み制御ユニット、およびメモリインタフェースユニットを装備した。

マイクロプログラム制御は、シンプルなハードウェアにおいて高機能な機械命令を実現するのに有効である。また、マクロレベルアーキテクチャやハードウェアレベルアーキテクチャでの仕様の変更による、相互のレベルへの影響を吸収することが出来るため、計算機の試作において、非常に有効であり、PEでマイクロプログラム制御を採用したことは妥当であった。

現在、PEハードウェアの詳細設計がほぼ完了している。今後、PEのプロトタイプの試作を行い、PEアーキテクチャの妥当性を検証する。

謝辞

我々と共に研究を進めている濱田 正泰君、菅原 幸子さん、寺岡 孝司君、茂木 久君に感謝致します。

参考文献

[1]T.Baba, et al. : "A Parallel Object-Oriented Total Architecture: A-NET," Proc. Supercomputing '90, pp.278-285 (1990).

[2]T.Baba, et al. : "A Network-Topology Independent Task Allocation Strategy for Parallel Computers", Proc. Supercomputing '90, pp.878-887 (1990).

[3]岩本　他:"並列オブジェクト指向言語A-NETLの言語処理系", 情報処理学会第38回大会, 4P-1 (1989).

[4] 富田眞治: "並列計算機構成論", 昭晃堂, pp.195-238（昭 61 - 11）.

[5]W.J.Dally, et al. : "Architecture of a Message-Driven Processor", Proc. of the 14th ACM/IEEE symposium on Computer Architecture, pp.189-196 (June 1987).

[6]鈴木　他:"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PEのハードウェア構成", 情報処理学会第40回大会, pp.1165-1166 (1990-3).

[7]寺岡　他:"並列オブジェクト指向トータルアーキテクチャA-NET　—PEのアーキテクチャ—", 情報処理学会第41回大会, 6P-7 (1990).

[8]吉永　他:"A-NET並列オブジェクト指向計算機のオペレーティングシステム", 情報処理学会コンピュータシステムシンポジウム(Mar 1991,発表予定)

[9]W.Horwat, et al. : "COSMOS: An Operating System for a Fine-Grain Concurrent Computer", MIT CVA memo (1990).