計算機アーキテクチャ　83－32
（１９９０．７．１８）

ファジィコンピュータのハードウェアアーキテクチャの一提案

徳永秀和　勝亦敦　山本創造　井上由文　安信誠二

国際ファジィ工学研究所 (LIFE)

人間のあいまいさを取り扱うために、ファジィ情報処理が注目されている。このファジィ情報処理には、ファジィ集合の処理と記号処理とを統一的に扱えると共に、ファジィ集合の表現と演算を容易かつ高速に行なえることが、要求される。我々は、ファジィ情報処理を効率良く処理し、ヒューマンインタフェースの優れた、ファジィコンピュータシステムの開発を行なっている。本システムの基本となる、ファジィ情報処理プログラムの開発環境として、オブジェクト指向プログラミングシステム MoNo(Meta Object Names Object) を、提案した。本稿では、MoNo を高速処理するための、ハードウェアアーキテクチャを提案する。

A Proposal on Fuzzy Computer Hardware Architecture

Hidekazu Tokunaga　Atushi Katumata　Sozo Yamamoto　Yoshifumi Inoue　Seiji Yasunobu

Laboratory for International Fuzzy Engineering Research (LIFE)
Siber Hegner Blg. 4F, 89-1, Naka-ku, Yokohama, 231 Japan

Fuzzy information procesing is gaining much attention as a means of handling uncertainty in human knowledge. To make fuzzy information processing, it is needed to handle fuzzy information and symbolic information, also representation and processing of fuzzy sets easily and speedily. We are now developing a fuzzy computer system, which can process fuzzy information efficiently and is good in human-interface. An object-oriented programming system MoNo(Meta Object Names Object) was proposed as a developing environment of this system. This paper proposes a hardware architecture for high speed execution of MoNo.

1. はじめに

人間の持つ主観量によるあいまいさ(ファジィ性)を取り扱う理論として、あいまいさをファジィ集合で表現し、処理するファジィ理論[1, 3]が、注目されている。このファジィ理論は、様々な分野に応用されており[7]、その有効性が示されている。また、今後も広範囲な分野で応用されることが期待される。

このような、ファジィ理論は、従来の数値や記号に基づく情報処理に、ファジィ性を含む知識(ファジィ情報/ファジィ知識)の処理を 付加したファジィ情報処理により行なわれる。

このファジィ情報処理では、高速な数値処理以外に、ファジィ理論と記号処理との統一処理と、ファジィ集合の表現と演算の容易かつ高速な処理などの機能が必要である。

現在、ハードウェアにより、制御用のファジィ推論の高速な実行を目的としたシステム[2]や、ソフトウェアにより、ファジィ集合処理の汎用化を目的としたシステムなどの、ファジィ情報処理を支援するシステム[6]が提案され、上記機能の一部が実現されている。

我々は、上記の機能を全て満足した、ファジィ情報処理を支援するファジィコンピュータシステムを開発している。

本稿では、まずファジィコンピュータのプログラム開発環境として提案しているオブジェクト指向プログラミングシステム MoNo[1]の概要について述べ、MoNo の高速処理を実現するハードウェアアーキテクチャについて述べる。

2. オブジェクト指向プログラミングシステム MoNo

ファジィコンピュータ上でのプログラム開発は、ファジィ情報処理の機能を持つオブジェクト指向プログラミングシステム MoNo[8]により行なわれる。

MoNo はファジィ情報処理を実現するために次の機能を持つ。

1) オブジェクトによるデータの表現
   a) ファジィ集合オブジェクト
   b) 知識オブジェクト
2) メタオブジェクト機構
   a) 多様なファジィ集合演算への対応
   b) 多様なオブジェクト指向パラダイム[2]の実現
3) メッセージ伝達 (message passing) 処理
4) 複製・追加によるオブジェクトの生成

3. ハードウェアアーキテクチャ

3.1 ハードウェアへの要求

ファジィコンピュータ上のプログラムは MoNo により記述する。このため、次に示す MoNo の処理をハードウェア化することにより、ファジィ情報処理の高速処理が期待できる。

1) ファジィ集合演算処理
2) ファジィ関係やファジィ積分等の数値演算処理
3) メッセージ伝達処理
4) オブジェクトの生成やガーベジ・コレクションに伴うメモリ管理処理

3.2 要求実現のための基本方針

上記の 1,2) のハードウェアへの要求に対しては、ファジィ集合演算で最も特徴的な処理を高速処理するファジィ集合処理演算器と、通常の数値演算器を複数個用いて、これらを並列に動作することにより実現する。

複数個の演算器を並列に動作させるには、VLIW[3][4]方式が有効である。従来の VLIW 方式では、コンパイラによる静的なプログラム解析によ

[1] MoNo = Meta Object Names Object

[2] Object-OrientedParadigm = OOP

[3] Very Long Instruction Word

り、バス、メモリ構成を含む演算器構成に最適なコードを生成する[5]。しかし、MoNo では、以下に述べるような、メッージ伝達により計算を行うため、コンパイラによる静的な解析が困難である。

メッセージ伝達による計算モデルは、オブジェクトがメッセージを受け取ると、そのメッセージに対応した、メッセージ伝達、数値、ファジィ集合等のデータに対する演算命令などの、アクションを実行することになる。

このように、各演算器に対する命令は、実行時に動的に生成される。このような環境下で、各演算器を効率良く並列に動作させ、ファジィ情報処理を高速処理するために、ハードウェアアーキテクチャの基本方針を、以下の方式とした。

1) オブジェクト管理：MoNo を解釈し(オブジェクトの生成処理、メッセージ伝達処理など)、各演算器に対する逐次的な演算命令流を、動的に発生する。

2) 機能分散命令流方式: 動的に発生された命令流を、各演算器毎の機能分散命令流に分散し、演算器間で同期をとりながら並列処理する。

3) 高機能演算命令： MoNo の解釈時に演算器構成を考える必要をなくし、演算命令の発生を容易にするため、高機能演算命令を用いる。

4) 同期処理: 演算器間の同期処理と、バスのスケジューリングは、全てハードウェアでサポートすることにより、機能分散命令流を高速に処理する。

5) メモリアーキテクチャ： 数値データを保持するメモリは、複数のバンクに分割し, 複数の演算器が異なるバンクに対して同時アクセス可能とすることにより、高速データ転送を実現する。

3.3 全体構成

上記の方式を実現するファジィコンピュータのハードウェアは、次のような構成要素よりなる(図1)。

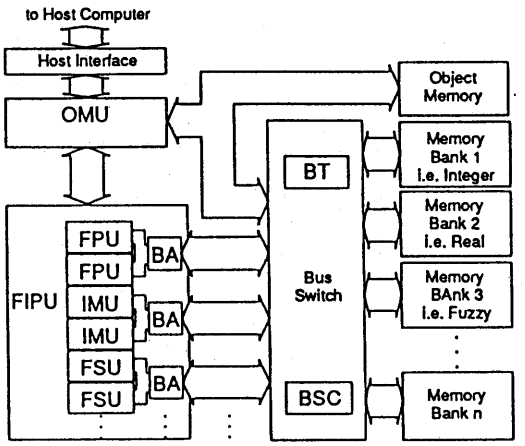
1) ホストインタフェース：VME バス制御

2) オブジェクト管理部 (OMU)：オブジェクト指向言語 MoNo の メッセージ伝達とオブジェクト生成を処理し、高機能演算命令を発生

3) 機能分散命令流方式演算部 (FIPU)：複数の機能演算器により 高機能演算命令を分散並列処理

4) メモリ：数値データを保持する 16 のバンクにに分割されたメモリと、 オブジェクト管理用のメモリにより構成

5) バススイッチ：バンクメモリの切替え



OMU : Object Management Unit
FIPU : Function-Partitioned Instraction-Stream-Method Processing Unit
FPU : Floating Point Unit
IMU : Integer Multiplier Unit
FSU : Fuzzy Sets Unit
BA : Bas Arbitor
BT : Bank Table
BSC : Bas Switch Controller

図 1: ハードウェアアーキテクチャ

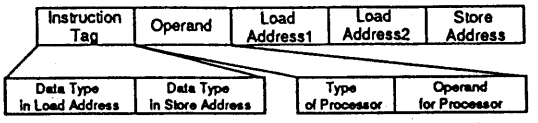
3.4 高機能演算命令

高機能演算命令は、演算器間の同期が容易になるように、以下の特徴を持たす (図 2)。

1) メモリ - メモリ演算モデル : 2 つまたは 1 つのデータをメモリより読み、任意の処理を行ない、処理結果のメモリへの書き込み操作を指示する。

2) 読み込みデータに対する制限: 読み込むデータは、必ず処理する演算器と同じタイプのデータ型である。すなわち、整数演算を行なう命令の読み込みデータは必ず整数型である。

3) 型に対するタグを付加 : 読み込みデータと書き込みデータにタグを付ける。演算器の選択のために、命令の型に対するタグを付ける。



- Instruction Tag — Data Type in Load Address ··· Load Addressのデータの型; Data Type in Store Address ··· Store Address1,2のデータの型
- Operand — Type of Processor ··· 演算器種別; Operand for Processor ··· 個々の演算器への命令
- Load Address1 — 引数 1 のアドレス; Data Type in Load Addressの示すデータ型と同じ型のデータのアドレス
- Load Address2 — 引数 2 のアドレス; Data Type in Load Addressの示すデータ型と同じ型のデータのアドレス
- Store Address — 演算結果を格納するアドレス

図 2: 高機能演算命令

3.5 メモリアーキテクチャ

メモリアーキテクチャは、以下のような構成である (図 1)。

1) オブジェクト管理用のメモリ

2) 複数のバンクに分割された数値データ用メモリ

3) 各バンクメモリからバススイッチへのメモリバンクバス

4) 演算器の各型からバススイッチへの演算器型バス

5) それぞれのメモリバンクバスを任意の演算器型バスと接続するためのバススイッチ

複数のメモリバンクは格納するデータの型により分割される。このようにすると、それぞれの演算器が自分と同じ型のデータ型のメモリバンクをアクセスする場合には、同一型の演算器間でのアビトレーションのみでアクセスできる。したがって、この場合複数の異なる型の演算器は、同時にメモリアクセスが可能となる。

メモリバンクのデータの型は、オブジェクト管理部により指示される。指示されたデータ型はバススイッチ制御器内のテーブルに保持され、演算器型バスとメモリバンクバスとの接続情報として使用される。

バススイッチ制御器は、演算器より出された要求に対応して、メモリアクセスの競合を解消し、演算器のメモリアクセスを可能とする。

4. 機能分散命令流方式演算部

4.1 構成と命令流

数値演算を高速処理する機能分散命令流方式演算部は、次に述べる構成要素よりなる (図 3)。

1) 逐次命令キュー (SIQ) : オブジェクト管理部より発生した高機能演算命令を、発生順に保持

2) 演算器命令キュー (PIQ) : 他の演算器との同期処理のための フラグビットを付加された、各演算器用の高機能演算命令を保持

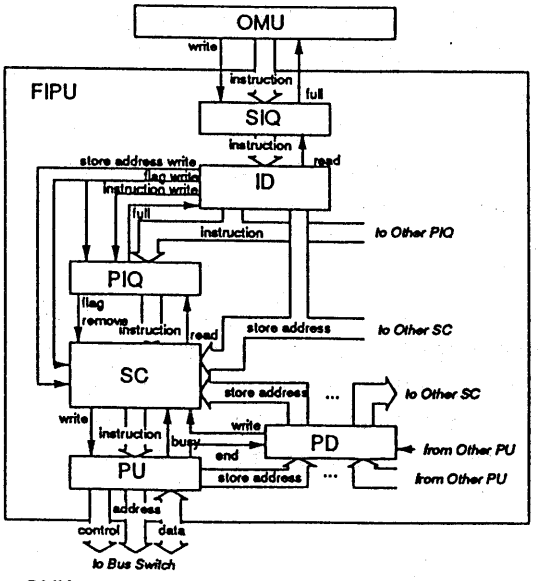
3) 演算器 (PU) : 高機能演算命令を並列処理するため、整数演算器、浮動小数点演算器、ファジィ演算器で構成

4) 命令分配器 (ID) : 逐次命令キューから各演算器命令キューに命令を分配し、同期のために必要な処理を実行

5) 同期制御器 (SC) : 演算器命令キューの高機能演算命令を、他の演算器との同期を考えて、処理可能な命令を演算器に転送

6) 演算器判別器 (PD) : 同期のための演算器からの信号を どの同期制御器に送るかを判別

次に、命令の流れを述べる。高機能演算命令は、オブジェクト処理部より発生された順に逐次



OMU : Object Manager
FIPU : Function-Partitioned Instruction Streams Method Processing Unit
SIQ : Sequencial Instruction Queue
ID : Instruction Distributer
PIQ : Processor Instruction Queue
SC : Synchronization Controller
PU : Processor
PD : Processor Discriminator

図 3: 機能分散命令流方式演算部

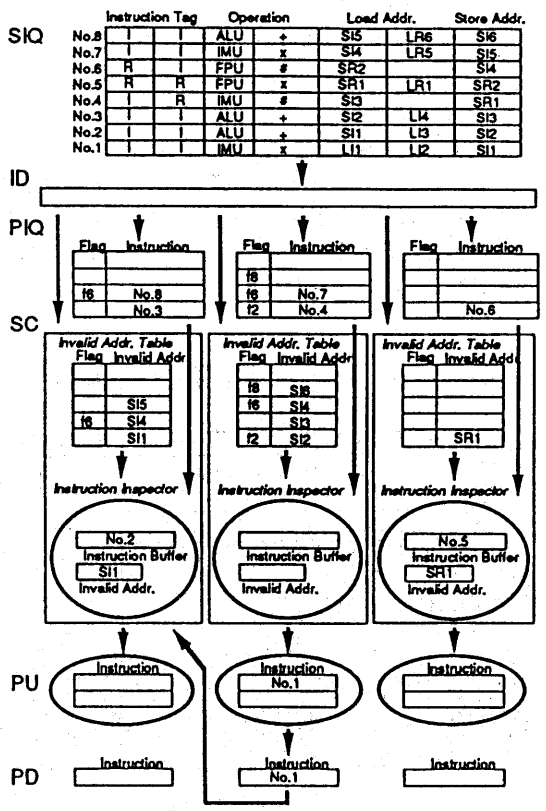


図 4: 命令流と同期

命令キューに保持される。命令分配器により逐次命令キューの先頭より命令を実行すべき演算器の演算器命令キューに送られる。同期制御器では、演算命令キューの先頭の命令を、実行可能かどうかを判断し、実行可能ならば演算器に送る。

4.2 演算器間の同期方法

演算器間の同期は、未実行の先行命令の書き込みデータのアドレスと、これから実行する命令の読み込みデータのアドレスのチェックにより行なう。逐次命令キューから演算器命令キューに転送する時に、適当な同期制御器の Invalid Address Table に、書き込みアドレスを追加する。そして、演算器命令キューから演算器に命令を転送する時に、Invalid Address Table のアドレスと転送しようとする命令の読み込みアドレスをチェックする。また、Invalid Address Table のアドレスのチェック範囲を示すために、演算器命令キューと Invalid Address Table の Flag ビットを用いる。

図 4 は、逐次命令キューに No.1 から No.8 までの命令があり、これを全て演算器キューに送った後に、No.1 の命令が実行された時の様子を示している。フラグの fn の n は、No.n の命令が演算器キューに送られた時に、立てられたことを示している。

4.3 演算器の処理概要

演算器は、実行可能な高機能命令を受けとり実行する。演算器の内部構成は、データロード部、実行部、データストア部に分けられ、メモリアクセスと実行を同時に処理する (図 5)。

このような、2 段のパイプライン処理を行なう

ために、演算器内には Instruction Buffer を持たせ、ここに連続する2命令を格納可能とした。そして、データロード部では、ロード前にそのアドレスと前命令のストアアドレスとの一致を調べることにより、演算器内の同期をとる。

また、ファジィ集合演算器ではベクタに対する処理の高速化が必要となる。そこで、ファジィ集合演算器は任意長のベクタに対する演算を1命令で実行する機能を持たせている。

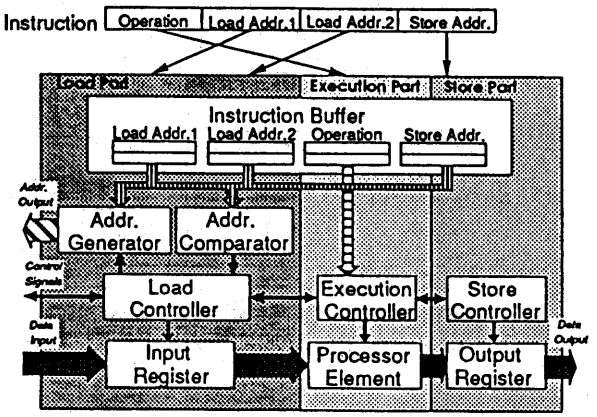


図 5: 演算器の構成

5. おわりに

我々は、ファジィ情報処理を支援するためにファジィコンピュータのプロトタイプ開発を進めている。ここでは、ファジィコンピュータのハードウェアアーキテクチャを中心に

1) ファジィ情報処理のアプリケーション開発からの要求を満足する、メタオブジェクトの概念を導入したオブジェクト指向 プログラミングシステム MoNo の基本機能

2) MoNo を解釈 (オブジェクトの生成処理、メッセージ伝達処理など) する オブジェクト処理部と、数値演算を高速処理する機能分散命令流方式演算部により構成される、ハードウェアアーキテクチャの基本設計

3) 機能分散命令流方式演算部の同期機構など内部構成の基本設計

について述べた。

今後は、ここに提案したハードウェアを製作し、ファジィ情報処理の支援に有効であることを、実証していく。

謝辞

本稿をまとめるにあたり、有益な助言を頂いた LIFE の研究員の皆様に感謝致します。また、本研究の機会を頂いた LIFE の寺野所長ならびに、新日本製鐵 (株)、山武ハネウエル (株)、(株) 日立製作所をはじめとする組合員各社に深く感謝します。

参考文献

[1] L. A. Zadeh. Fuzzy sets. *Information and Control*, pp. 338–353, 1965.

[2] 渡辺浩之. ファジィ推論法のＶＬＳＩ化. 電子情報通信学会論文誌, Vol. J72-A No.2, pp. 179–187, 1989.

[3] 寺野寿郎, 浅居喜代治, 菅野道夫 (編). ファジィシステム入門. オーム社, 1987.

[4] 安部正人, 永田仁史, 牧野正三, 城戸健一. *vliw* 計算機 *kidoch* のメモリ管理機構. コンピュータシステム研究会報告書. 電子情報通信学会, 1989.

[5] Jone R. Ellis, editor. *Bulldoc:A Compiler for VLIW Architectures*. The MIT Press, 1986.

[6] 馬野元秀. Lisp によるファジィ集合処理システム. 第3回ファジィシステムシンポジウム, pp. 167–172, 1987.

[7] 安信誠二. ファジィ推論を利用した列車自動運転. 情報処理, pp. 970–975, 8 1989.

[8] 井上 由文, 安信誠二. ファジィ情報処理の機能を持つオブジェクト指向プログラミングシステム mono の概念について. 第4回人工知能学会全国大会予稿集. 人工知能学会, 1990.