特 集 ファジィ制御ハードウェア

デジタル・ファジィ推論チップの現状と展望†

戸 貝 正 喜*

1. はじめに

ファジィチップ開発の究極の目的は、ファジィ・コンピュータである。解り切ったことを、こと細かに記述してやらないと作動してくれない従来型のコンピュータに比べ、人間のもつ「あいまいさ」をファジィ変数として捉え「あいまいに処理する」ソフトコンピューティング指向のコンピュータがファジィ・コンピュータであると言えよう。

RISC にせよ CISC にせよ、従来型のアーキテクチャに基づくコンピュータに頼っていたのでは、爆発的に増大するソフトウェアのステップ数を抑制する事は出来ない。こうした、プログラム量の爆発に対処するためには、コンピュータ側に何らかの知的能力をもたせ、人間の使う道具(ツール)としての完成度を高める必要がある。ファジィ・コンピュータはそれを可能にする。

ファジィ理論の応用がこれ程までに一般化してきた理由の一つに、ファジィ変数を、従来の人工知能アプローチの様に、単に記号として取り扱うのではなく、メンバーシップ関数という数値データによって定義でき、ファジィ推論も数値処理により実行出来る点が挙げられよう。したがって、従来の数値処理を目的としたコンピュータでも、ファジィ理論を利用したプログラムは実行可能で

ある。しかし、その処理には無駄が多く、時間が掛かり過ぎる。可能であることと最適であることとは、全く別問題であることを認識しなくてはならない。

2. ファジィ・コンピュータ設計上考慮すべき事項

従来型のコンピュータの問題点の一つは、プログラム言語自体が、2値論理に基づいて構成されている点である。プログラムのそれぞれのステートメントが、真(即ち1)であるか、偽(即ち0)であるかしか評価しない。ファジィ演算やファジィ推論など、ファジィ論理体系に基づいた体系では、0と1の中間の値をも取り扱わなければならないので、従来型の2値論理のパラダイムに基づいた言語では効率良く記述出来ない。その結果、冗長なプログラムとなってしまい、実行に時間がかかる。

もう一つの問題は、コンピュータのアーキテクチャそのものに起因するものである。ファジィ変数を定義するメンバーシップ関数は、関数の集合や数値の集合体(ベクトル)と見なすことが出来る。さらに、こうしたファジィ変数を用いて行われるファジィ推論は、本質的には並列演算である。従って、逐次実行型のノイマン型アーキテクチャを持つコンピュータを用いてファジィ演算を行っていたのでは、非効率であり、かつ冗長なプログラムとなってしまふ。

プログラミング言語に起因する不自由さは、述語論理に基づいて Prolog という非手続き言語が作られたように、ファジィ論理に基づく高度な並

† Perspectives and Trends of Digital Fuzzy Inference Processors

* Masaki TOGAI

日本インフラロジック株式会社
Nihon InfraLogic Co. Ltd.

列性をもつファジィ・プログラミング言語を開発することによって解決しようとしている。

ノイマン型アーキテクチャに起因する不自由さは、ファジィ言語変数、ファジィ・ルール、ファジィ推論等を直接取り扱える命令体系や並列処理機能を持つアーキテクチャの採用により、確実に改善出来る。さらに、メモリーにロジックを付加したり、ファジィ連想メモリーなどを利用することにより、演算実行速度の向上が図れる。いずれにせよ、言語における論理概念とコンピュータ・アーキテクチャにおける論理概念との相違、つまり“セマンティック・ギャップ”をいかに小さく出来るかが問題となる。このため、ファジィ・コンピュータの実現のためには、

- ・最新 VLSI 技術の利用
 - ・ファジィ論理演算指向型アーキテクチャの採用
 - ・コンパイル技術の積極的活用
- などを図る必要がある[1][2]。

従来型のデジタル・コンピュータは、中央演算素子(CPU)、メモリー、周辺機器制御回路などの各モジュールを、共通のデータ・バスで接続したシステムである。こうしたシステム構成においては、データ・バスを通じて行われるモジュールから他のモジュールへのデータ転送速度が、システム全体の処理速度を決定する。従って、全システムを1つのチップ上に構築し、モノリシック(ワンチップ)化することにより、飛躍的に効率を向上させることができる。この典型が、マイクロ・コンピュータである。

通常のデジタル・コンピュータシステムは、プログラムを蓄積し、逐次実行して行くシリアル処理である。ところが、例えば、ファジィ推論は並列性を持つばかりでなく、前件部処理と後件部処理とが比較的独立性を有していることから、並列処理やパイプライン処理を行うことにより、高速化が図れる構造になっている。

最近の RISC 型コンピュータは、命令種類が少ない上に、1つの命令が少ないマシンサイクルで実行出来るようなアーキテクチャを備えている。ファジィ・コンピュータにおいても、同様に、フ

ァジィ演算が、簡単な命令と数少ないマシンサイクルで実行出来るようなアーキテクチャの工夫が必要となる。

従来方法によるファジィ演算と、ファジィ命令を持たせた場合の演算例とをリスト1にそれぞれ示す[3]。

(i)には、従来方法による MIN 演算例を示す。同じく MIN 演算をファジィ命令で記述した例を(ii)に示す。一方、(iii)には、従来方法によるメンバーシップ関数のグレードを計算する例を、(iv)には、ファジィ命令を持たせた場合の記述例を示す。

リスト1 従来型命令によるファジィ演算と ファジィ演算命令

(i) MIN 演算(従来型プロセッサ)

```
CMP AX,BX      ; AX と BX を比較する。
JL  NEXT       ; AX の方が小さければ
                NEXT ヘジャンプする。
MOV AX,BX      ; BX の値を AX にコピーする。
NEXT:          ; 次の命令を実行する。
```

(ii) MIN 演算(ファジィ・プロセッサ)

```
MIN R1,R2,R3    ; R1 に[R2]と[R3]との
                小さいほうの値を書き込む。
```

(iii) グレード計算(従来型プロセッサ)

```
MOV AX, MEMBER ; メンバーシップ関数データの
                ベースアドレスをロードする。

MOV DS, AX      ;
MOV BX, VALUE   ; 変数値をロードする。
MOV AX, [BX]    ; メンバーシップ値獲得。
```

(iv) グレード計算(ファジィ・プロセッサ)

```
MBF R1, MEMBER ; R1 のグレード値をメンバー
                シップ関数 MEMBER から求め、R1 に格納する。
```

この様に、従来のアーキテクチャでは、少なくとも4命令以上の機械語コードを必要としたファジィ演算が、適切な命令を用意することにより、1命令で等価な記述を行うことが可能となる。このように、ファジィ演算命令を追加し、例えば、ファジィ変数そのものを命令の引数として、直接

定義出来るようにすれば、ファジィ・プログラミング言語との親和性も向上し、“セマンティック・ギャップ”も小さくできる。ファジィ命令は、マイクロコードにより容易に追加出来るが、命令のマシンサイクルを削減するためには、ファジィ命令実行用の専用ロジック演算回路をコンピュータ内部に持つ必要がある。ファジィ論理演算回路の1つに、ファジィ ALU (FALU) がある。FALU をデータパスの一部として組み込む方法や、コプロセッサとして組み込む方法などがある。

一方、ファジィ・コンピュータを効率よく実行させるためには、高級言語であるファジィ・プログラミング言語を、コンピュータの命令セットに変換(コンパイル)してやらなければならない。この際、最適化コンパイル技術を駆使して、この変換効率が極力 1 : 1 に近づくように工夫してやる必要がある。

例えば、MAX や MIN 演算を C 言語のマクロを用いてリスト 2 (i) のように記述した場合と、(ii) に示すように論理演算を用いた場合とでは、知識ベースのコンパイル効率が異なり、実行速度に大きな影響を及ぼす。

実行効率検証のために、リスト 2 をコンパイルして行ったベンチマークテストの結果を表 1 に示す。論理演算による定義の方が、マクロによる定義より、およそ 1.5 倍程コンパイル効率がよいことが分かる。

リスト 2 ファジィ演算のマクロと論理演算による定義例

(i) マクロによる定義例

```
#define MAX(a,b) (((a) > (b)) ? (a) : (b))
#define MIN(a,b) (((a) < (b)) ? (a) : (b))
```

(ii) 論理演算による定義例

```
#define MAX(a,b) ((a) | (b))
#define MIN(a,b) ((a) & (b))
```

3. デジタル推論プロセッサの設計例

3.1 概要

大手 ASIC メーカーである VLSI テクノロジー社は、ファジィベンチャー企業であるトガイ・インフラロジック社との提携により、ASIC に容易に組み込むことのできるファジィコアをリリースした。このコアは、あらかじめ ASIC チップに内蔵され、ファジィ推論に必要な基本機能を果たす心臓部(コア)のデザインのことで、このコアの周辺にユーザーの必要とする機能を付加することにより、多種多様なファジィ ASIC を実現するものである。

システムの精度は、8 ビットから 32 ビットまで任意の精度をパラメータの変更のみで実現することができるばかりでなく、ビルトイン・セルフ・テスト (BIST) 機能も付加されており、ASIC の開発並びに使用時にトレースなどが容易にできるよう工夫されている。

他の周辺回路やホスト CPU とのインターフェースも容易にできるようにと、インターフェース制御信号数も極力少なくしてある。

このようにコンパクトな設計になっているため、多種多様な形態の ASIC に応用できる。例えば、特定のファジィ機能実現のためのカスタムファジィ IC に用いる方法のほか、Z 80, ARM 6, i 386 などといった通常のプロセッサと組み合わせることにより、標準的なプロセッサとピンコンパチブル

表 1 リスト 2 のコードのコンパイル効率の比較

	i386(16MHz)	i386(20MHz)	i486(18MHz)	i486(33MHz)
マクロ(μS)	224	151	160	47
論理演算(μS)	151	102	120	27
相対速度	1.48	1.48	1.33	1.74

な汎用ファジィ・プロセッサとすることもできる。さらには、FCA のみを搭載しファジィ推論加速用のコプロセッサとして、マイクロコントローラなどのホスト・プロセッサとインターフェースさせることができるなど、FCA は様々なアプリケーションを実現できる。

3.2 アーキテクチャ

FCA はハーバード型の RISC アーキテクチャになっている。図 1 にそのブロック図を示す。

このコアでは、ファジィ AND、ファジィ OR、ファジィ NOT のようなファジィ演算のみならず、MAX-DOT 方式に基づいたファジィ推論や、重心法、高さ法、アルファ法といった非ファジィ化法も直接サポートする機能をもっているため演算効率がよい。

前件部、後件部においてそれぞれ取り扱える変数の数も、システムの基本精度により異なり、例えば、10 ビットの FCA では、それぞれ 1 から 1,024 変数までの取り扱いが可能である。また、同様に取り扱いえるルール数も、FCA の基本精度によるが、ルールメモリーの容量にも制限される。

前件部及び後件部で用いるメンバーシップ関数

の形状は、特に制限はなく任意に規定できる。特に、折れ線表示、二次曲線表示、いわゆる S, π 並びに Z 型関数表示、マップ表示のメンバーシップ関数処理は、FCA が直接サポートしているため処理速度が大幅に短縮されている。

推論方法は MAX-DOT 法を基本に、高速演算可能なように改良を加えた方式である。10 ビットの FCA の場合、前件部 5 変数、後件部 2 変数の形式をもったルール 49 個をすべて用いて推論を行った時の推論速度は 70 μ S 以下と高速であり、FC 110 DFP(8 ビット)に比べると 3 倍以上の速度が得られている。

FCA を用いたファジィ推論加速チップもすでにリリースされている[4]。

4. 今後の展望

ファジィ応用が進むにつれ、高性能ながら低コストなプロセッサが必要となってきた。また、こうしたハードウェアの出現により、さらにファジィ応用が加速される。これまでのファジィ応用は主として、

(1) 汎用プロセッサ(ソフトファジィ)

図 1 FCA の内部ブロック図

(2) アナログ・ファジィ推論プロセッサ

(3) デジタル・ファジィ推論加速プロセッサ

により、実現されてきた。汎用プロセッサを用いる大きな利点は、応用ばかりでなく汎用のソフトウェアを実行できる柔軟性にある。さらには、オペレーティング・システム、最適化されたコンパイラ、デバッガなどの豊富な開発ツールが用意されている。しかし、ファジィ推論などを含むファジィ演算には不適當であり、ファジィ演算を高速に実行しようとする、結果として高価格なプロセッサを使用せざるを得なくなる。

一方、アナログ手法は、未だに研究段階にあり、その効用も議論の余地があるところである。

デジタル型加速プロセッサは、高性能ながらもホスト・プロセッサを必ず必要とする不都合がある。結果として、システムがコスト高になってしまう欠点がある。また、このアプローチでは、

ファジィ・プロセッサ自体、限られた機能しか提供できない。

これらの理由により、今後の傾向としては、汎用のプロセッサにファジィ演算機能を付加することにより、汎用性を損なうことなくファジィ演算の性能を向上させたタイプのプロセッサが主流となるであろう。そうした兆しは既に欧州にあり、Ungering 等[5]は、汎用の 8-Bit マイクロコントローラにファジィ機能を付加させた「F 166」を開発した。また、Avodadro 等[6]は、通常の RISC 命令の他に、ファジィ応用のために積和演算と MIN 演算命令を付加した RISC プロセッサ「FLORA」を開発した。図 2 に、ファジィ演算ユニット (FCU : Fuzzy Calculation Unit) を付加した F 166 の ALU のブロック図を示す。また、ファジィ推論を各種のデバイスで行った場合の推論速度の比較を表 2 に示してある。

図 2 FCU を付加した ALU の構造

表 2 デジタル・ファジィ・プロセッサの性能比較例

(Compute time is in μ s)	8051 micro FPL	i486 (33MHz)	FC110	FCA	FLORA (50MHz)	F166
2 入力, 1 出力, 7 ルール	2860	83	42	15	1	>2200
7 入力, 3 出力, 25 ルール	25490	700	198	66	8	>2200

5. まとめ

最近の研究成果[7]によれば、ファジィ・プロダクション・システムを効率よく処理するためには、ルール結論部処理及び非ファジィ化処理の最適化が重要であることが指摘されている。また、ファジィ推論演算において、約70%の時間はデータの転送に費やされていることからしても、ファジィ・プロセッサの設計においても、こうした汎用プロセッサの設計上考慮すべき事項を十分意識する必要があることを示唆している。汎用コンピュータの技術革新が、ファジィ・プロセッサの性能向上にも大きく寄与することになる。

参 考 文 献

- [1]ヘネーシ&パターソン：コンピュータ・アーキテクチャ、日経 BP 社(1992)
 - [2]坂村：コンピュータ・アーキテクチャ、共立出版(1984)
 - [3]畑田、戸貝：“命令セット・アーキテクチャから見たファジィプロセッサの研究”、第9回ファジィシステムシンポジウム講演論文集、pp.749-752(1993)
 - [4]VY86C570 - 12-Bit FCA Fuzzy Coprocessor, VLSI Technology(1993)
 - [5]Ungering et al：“Architecture of a Fuzzy-Processor Based on an 8-bit Microprocessor,” to be presented at Third IEEE Int'l Conf. on Fuzzy Systems, June(1994)
 - [6]Avogadro et al：“An Optimized RISC Instruction Set for Fuzzy Applications,” to be presented at Third IEEE Int'l Conf. on Fuzzy Systems, June(1994)
 - [7]畑田、戸貝：“ファジィ推論エンジンの最適化手法とその効果”、第10回ファジィシステムシンポジウム講演予定(1994)
- [問い合わせ先]
〒135 東京都江東区東陽6-5-9
マックストロービル6F
日本インフラロジック株式会社
戸貝正喜
☎：03-3648-2151
☎：03-3648-2090

著 者 紹 介

戸貝 正喜 (とがい まさき)

米国 Togai InfraLogic, Inc. 会長、
日本インフラロジック株式会社 代表取締役

1971年 防衛大学電気工学教室 卒業。1982年 米国デューク大学大学院博士課程修了。Ph.D.。AT&T ベル研究所主任研究員、ロックウェルインターナショナル中央研究所プロジェクトマネージャ等を経て、ファジィプロセッサ、ファジィ開発環境及び各種ファジィ応用製品の開発・販売のため、1987年にトガイ・インフラロジック社(米国カリフォルニア州)、日本インフラロジック株式会社設立し現在に至る。