A Fuzzy Logic Controller with Reconfigurable, Cascadable Architecture

Wayne D. Dettloff
Kathy E. Yount

Microelectronics Center of North Carolina
P.O. Box 12889
Research Triangle Park, NC 27709

Hiroyuki Watanabe

Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599-3175

Abstract

A general purpose fuzzy logic inference engine for real-time control applications has been designed and fabricated in a 1.1, 3.3v, DLM CMOS technology. Up to 102 rules are processed in parallel with a single 688K transistor device. Features include a dynamically reconfigurable and cascadable architecture, TTL compatible host interface, laser programmed redundancy, special mode for testability, RAM rule storage and on-chip fuzzification and defuzzification.

Introduction

Traditional process control depends upon the establishment of a mathematical model of the system. Expert systems were created to mimic the behavior of a skilled human operator for those processes too complex to be modeled in real-time. Fuzzy logic inference engines use an expert system paradigm for automatic process control [1], and have been successful in controlling chemical processes, trains [2], cement kilns [3], and small aircraft [4]. This paper describes a general purpose fuzzy controller which takes advantage of the high speed and integration capability afforded by VLSI.

Previous work [5] prototyped a similar processing element but was only intended to demonstrate plausibility. Features including RAM rule storage, a simple host interface, a dynamically reconfigurable architecture, TTL compatible input and output ports ensure maximum flexibility at the system level. Real-time operation is obtained by processing all rules in parallel with three levels of pipelining. A test mode is incorporated to guarantee testability and a unique form of redundancy allows for a usable device even if only a few data paths remain defect-free.

Fuzzy Logic

Regular expert systems are based on the concept of "crisp" sets, of which any given item is either a member or not a member. Mathematically, a set $S$ with characteristic function $f_S : \{0,1\}$ and $e$ in the universal set may be defined:

$$e \in S \text{ if } f_S(e) = 1,$$

$$e \notin S \text{ if } f_S(e) = 0.$$  

Boolean logic will not return a "1" until an input becomes a member of a specific set of values, and will return a "0" otherwise. Fuzzy logic replaces "true" and "false" with continuous set membership values ranging from 0 to 1, which mirrors our natural language concepts. Action is required when temperature is "approximately 92°C" (Fig. 1), so 87°C returns a degree of membership of 0.75, i.e. $\mu_T(87°C) = 0.75$.

![Figure 1: Approximately 92°C](image)

Zadeh [6] proposed the extension of ordinary set theory to fuzzy sets:

Intersection $C = A \cap B$

$$\mu_C(e) = \min(\mu_A(e), \mu_B(e)), e \in U.$$  

Union $C = A \cup B$

$$\mu_C(e) = \max(\mu_A(e), \mu_B(e)), e \in U.$$  

Complement $-A$

$$\mu_{-A}(e) = 1 - \mu_A(e), e \in U.$$  

Intersection (Fig. 2), union, and complement are fundamental operations and can be used to build virtually any logical function.

![Figure 2: Fuzzy Set Intersection](image)
One of the most important inference rules, *modus ponens*,

Premise : A is true
Implication : If A then B
Conclusion : B is true

complements fuzzy logic because implication is also readily translated from the human decision making process. Professionals express even the most complex practices in the form of rules, usually with fuzzy sets as operators.

If (1) the temperature is approximately 92°C and (2) the viscosity is very thick
Then (3) increase amount of water slightly.

Typically, 10-100 such rules comprise an entire controller, and are initially obtained by interviewing experts. Since fuzzy predicates are inherently range based, fewer rules are required than regular Boolean-based expert systems which demand precise matches.

Formulating conclusions based on the compositional rules of inference for approximate reasoning was suggested by Zadeh[7]. For example, three rules with two inputs (A' and B') and one output (C') can be implemented using the fuzzy set operations (Fig. 3). Combination of multiple rules is achieved by simple ORing[5].

Rule 1: If (A' is A1) and (B' is B1) then (C1 is C1')
Rule 2: If (A' is A2) and (B' is B2) then (C2 is C2')
Rule 3: If (A' is A3) and (B' is B3) then (C3 is C3')

The fuzzified inputs A' and B' are broadcasted to all the rules simultaneously to be compared to stored premises (IF-parts). Conceptually, the better the match of all the inputs to a stored rule the more influence it will have in the final weighting.

Weights \( \alpha_i^A \) and \( \alpha_i^B \) are calculated by

\[
\alpha_i^A = \frac{\max (\min (A', A_i))}{\times},
\]

\[
\alpha_i^B = \frac{\max (\min (B', B_i))}{\times},
\]

In the example, \( \alpha_1^A = 0.0 \) indicates a complete mismatch of the input proposition to the stored premise which translates to a complete non-contribution of Rule 1 to the output. Rule 2 has \( \alpha_2^A = 0.35 \) and \( \alpha_2^B = 0.70 \), so the former determines how well the rule matches \( (\alpha_2) \) since

\( \omega_i = \min \left[ \alpha_i^A, \alpha_i^B \right] \).

Similarly, \( \omega_3 = 0.65 \) since \( \alpha_3^A = 0.85 \) and \( \alpha_3^B = 0.65 \).

The conclusion of each rule is C_i (shaded in Fig. 3)

\[ C_i = \min (\alpha_i, C_i) \]

which represents the amount of action (THEN-part) each rule contributes. Combining all three rules to obtain the fuzzy result C' is calculated by

\[ C' = \max (C_1', (\max C_2', C_3')). \]

Since rule 3 had the best overall match of the input conditions, the associated action weighs heavily in the output. Finally, the adjusting mechanism will require a "crisp" number which is accomplished by calculating the center of area (C*) of the resultant fuzzy function.

\[ C = \frac{\int C \, dC}{\int \frac{dC}{dC}} \]

Figure 3: Inference

475
VI SI Implementation

Fig. 4 shows the fuzzy chip in a typical real-time, closed-loop controller configuration with 4 inputs and 2 outputs.

![Fuzzy Logic Controller](image)

**Figure 4: Fuzzy Logic Controller**

Host interaction is accomplished via a TTL compatible, infinite wait state, interrupt driven, memory mapped interface. Rules are downloaded into the fuzzy memory at boot-time and can be updated dynamically with minimal disruption. Two words are reserved for the status registers which control the mode (load, process, or test) and the configuration.

Since THEN processing can not commence until all premises have been compared to the inputs and defuzzification can not begin until the entire output function is defined, a three stage pipeline naturally emerges. Fuzzification, the process of converting crisp values to fuzzy sets, can occur simultaneously with the IF processing. A qualified clocking scheme guarantees the separation of calculations yet allows all of the logical blocks to operate simultaneously.

Fig. 5 depicts the logic required to implement the intersection or min function of two 4-bit membership expressions.

![Min Unit](image)

**Figure 5: Min Unit**

Each 98 transistor block operates in 25ns and can be isolated and 100% stuck-at fault tested[8].

The number of system inputs and outputs consequently determines pin count, rule logic complexity and the number of fuzzifiers and defuzzifiers. Choosing fixed numbers, however, potentially limits flexibility so two architectural features were implemented to maximize applicability. First the chip can be software configured for two input/output combinations, whereupon the latter provides for twice as many total rules.

IF A and B and C and D THEN Do E and Do F,

or

IF A and B THEN Do E.

Secondly, devices can be cascaded (Fig. 6) for a multilayered rule set under software control. Feedback may also be implemented but research into the potential of such a generalized formulation has yet to be reported.

The fuzzifiers are independent counter-addressed RAMs. Inputs are sampled every 64 clock cycles and are used as the starting addresses of the stored expressions. Uncertainty or inaccuracy of the sampled values from the sensors is related to the "width" of the stored fuzzification function. "Crisp" imitation is accomplished by storing a unit pulse in the appropriate fuzzifier. Accessing and broadcasting the 4-bit membership functions every clock cycle is a critical speed path.

Fig. 7 shows a block diagram of the IF logic in every rule. Fuzzified inputs are compared to the rule memory by the first column of min units. Max units then determine the peak of the premise comparison. A small binary tree of min units computes the limiting proposition for the entire rule (oE). Finally, the output goes to the THEN-part which calculates that rule’s contribution (CE) to the final result. Not shown is test

![IF Logic](image)

**Figure 7: IF Logic**
logic designed to allow controllability and observability for each min and max unit. Reconfigure logic steers the data streams to be combined or delayed appropriately. The output proceeds to a tree of max units which combines the results of all the rules and feeds the defuzzifier.

Defuzzification is the computation of the center of area (COA) under the final membership function. Denoting the final fuzzy subset as \( A \) (instead of \( C' \)),

\[
C^* = \frac{\sum_{n=0}^{63} \mu_A(n)}{\sum_{n=0}^{63} \mu_A(n)}.
\]

The denominator is simply the summation of the data stream from the final tree stage. Since the numerator can be computed

\[
\sum_{n=0}^{63} \mu_A(n) = \mu_A(63) + \mu_A(62) + \mu_A(61) + \cdots + \mu_A(0),
\]

by repeatedly adding the denominator an extra 16-bit adder was designed. Finally, a 16x10 divider circuit, implemented as a repeated adder/subtractor, computes the value that appears on the pins. Another critical path in the device is the 16-bit adder, consuming 27ns of the device cycle (Fig. 8). Hence, the three stage pipeline is complete. An input is sampled every major cycle but the result does not appear at the pins until two and a half additional cycles have passed.

Since rules are processed in parallel, global design considerations become paramount in the determination of the number of rules. Clock generation, data bus loading and control signal routing become speed issues above one hundred rules. Secondly, multiple chips can be utilized if an application requires more capability than a single device can provide. Finally, the floorplan of 102 rules fits a standard 84 pad frame with the stipulation of accessing 1224 bits of static RAM in parallel every cycle.

Redundancy is essential for yield enhancement purposes. A processing defect in the data path is remedied with a single laser shot to the kill fuse (Fig. 9) effectively disconnecting the entire rule. Other fuses allow memory congruency to be maintained since any address can be reprogrammed to replace killed ones. Systems requiring only 90 rules can use any device with up to 12 defective memory cells, min/max units, sense amps, etc. Fig. 10 shows Rule 93 eliminated and Rule 92 address lines 4 and 5 inverted.

To facilitate the testing of 688K transistors, a test mode was implemented. RAM's can be fully accessed at device speed. Inputs to the rules can be driven as primary inputs, and outputs of the THEN trees go directly to the pins, isolating these blocks of random logic representing 117K transistors. Furthermore, the rules themselves are transposed to allow immediate access to their internal state and with automatic vector generation software \[8\], 99.8% stuck-at fault coverage was achieved.

Summary

Table 1 summarizes the process\[9\], device specifications and primary architectural features. The device was designed to operate at 36MHz, which translates to 580K fuzzy logical inferences per second with up to 102 rules per inference. Current performance (Fig. 11) indicates a yet unsolved speed limiting design problem. The chip photomicrograph and corresponding layout map is shown in Fig. 12. Future work centers around completing a board and development system for applications prototyping. Simulation software with graphical input capability is currently in use. Potential applications include real-time decision making and control in robotics, aerospace, and a host of complex industrial processes.
Acknowledgements

The defuzzifiers were designed by Jeff Hultquist, Jin-Fang Wang and Jim Symon of the University of North Carolina at Chapel Hill. Don Blevins designed the clocks and Fred Heaton made chip simulation feasible by coding the functional model and employing a high-level design methodology. The authors would also like to thank Mark Kellam for the timely and expert processing of the very first fabrication run, which yielded 100% functional devices.

TABLE 1

<table>
<thead>
<tr>
<th>Die Size</th>
<th>7750µm x 9050µm</th>
</tr>
</thead>
<tbody>
<tr>
<td># Transistors</td>
<td>688,131 (476,160 in RAM)</td>
</tr>
<tr>
<td># Pins</td>
<td>84 (16 Power/GND)</td>
</tr>
<tr>
<td>Package Type</td>
<td>PGA (Standard Pad Frame)</td>
</tr>
<tr>
<td>Power Supply</td>
<td>3.0 - 3.3v</td>
</tr>
<tr>
<td>Power</td>
<td>1.2 W @ 0°C</td>
</tr>
<tr>
<td>Interface</td>
<td>TTL compatible</td>
</tr>
<tr>
<td>Modes</td>
<td>4 in/2 out/51 rules, 2 in/1 out/102 rules, I/Os individually programmable</td>
</tr>
<tr>
<td>Redundancy</td>
<td>Laser programmable</td>
</tr>
<tr>
<td>Test</td>
<td>Over 99.8% stuck-at fault coverage</td>
</tr>
<tr>
<td>Process</td>
<td>1µm N-well CMOS</td>
</tr>
<tr>
<td>Gate Lengths/ gate oxide</td>
<td>1.0µm/22.5 nm</td>
</tr>
<tr>
<td>Poly/Metal1/Metal2</td>
<td>2.8/2.8/4.0 µm pitch</td>
</tr>
</tbody>
</table>

References


