URC97002

ASICs Approach for the Implementation of a Symmetric Triangular Fuzzy Coprocessor and its Application to Adaptive Filtering

Saleh abdel-hafeez, Scott Starks, and Bryan Usevitch
Department of Electrical and Computer Engineering
University of Texas at El Paso

ABSTRACT

This paper discusses the implementation of a fuzzy logic system using an ASICs design approach. The approach is based upon combining the inherent advantages of symmetric triangular membership functions and fuzzy singleton sets to obtain a novel structure for fuzzy logic system application development. The resulting structure utilizes a fuzzy static RAM to store the rule-base and the end-points of the triangular membership functions. This provides advantages over other approaches in which all sampled values of membership functions for all universes must be stored. The fuzzy coprocessor structure implements the fuzzification and defuzzification processes through a two-stage parallel pipeline architecture which is capable of executing complex fuzzy computations in less than 0.55μs with an accuracy of more than 95%, thus making it suitable for a wide range of applications. Using the approach presented in this paper, a fuzzy logic rule-base can be directly downloaded via a host processor to an on-chip rule-base memory with a size of 64 words. The fuzzy coprocessor's design supports up to 49 rules for seven fuzzy membership functions associated with each of the chip's two input variables. This feature allows designers to create fuzzy logic systems without the need for additional on-board memory. Finally, the paper reports on simulation studies that were conducted for several adaptive filter applications using the least mean squared adaptive algorithm for adjusting the knowledge rule-base.

I. Introduction

Fuzzy logic systems (FLS) have been successfully applied to a wide variety of practical problems. Notable applications have centered on areas such as control, expert systems, digital signal and image processing, and robotics [1-3]. The desire to use fuzzy logic in real-time has led to the development special-purpose fuzzy hardware systems [4]-[6]. Many of these systems require the use of high-cost VLSI fuzzy logic circuits and memory chips. Often the speed of these systems is slow due to the time it takes to retrieve and save truth values. Computational accuracy can be a drawback as well. It is established in [7] that the design of an FLS can be made easier by simplifying the internal parameters of the system. Despite these simplifications, the resulting design is still capable of supporting a wide class of applications.

The aim of this paper is to present the ASICs hardware development of a fuzzy coprocessor based upon the concept of the reduced symmetric fuzzy singleton set reference [8] which helps alleviate some of the drawbacks associated with man y current fuzzy hardware systems. This fuzzy coprocessor has the following features:

(1) two singleton inputs,
(2) one crisp output,
(3) seven symmetric triangular membership functions associated with each input,
(4) fuzzy static RAM for rule-base storage, and
(5) on-chip fuzzification and defuzzification processes.

The hardware implementation can be described using VHDL code where the schematic and the detailed characteristics of the circuit are generated using an optimization compiler by Mentor Graphics.

The fuzzy coprocessor's hardware implementation requires a 64-byte Static RAM, an 8-bit sign adder/subtracter, one 8-bit sign multiplier, and an 8-bit comparator as illustrated in Figures 3-4. The design, which contains approximately 10,000 gates, has been implemented using FPGAs Alters technology and runs at a 10 MHz clock speed. Simulation results indicate that the design can run at a 25 Mhz clock speed using 1.2μ CMOSN technology. The paper also discusses the application of the proposed architecture to problems of interference noise cancellation.

II. *FLS Coprocessor Design Procedures*

A discussion of the operation of the FLS coprocessor was initially presented in [8]. In the present work, we will expand upon these discussions and present modifications which lead to a more cost-effective implementation. The FLS coprocessor chip provides two inputs (xl, x_2) and one output. We denote the maximum number of membership functions by the symbol K and for the present study, the maximum value for K is 7. The membership functions are constructed such that they are symmetric triangular to the center of the domain and the domain of themselves as shown in Figure 1. The end-point pair (a_{ij}^-, a_{ij}^+) completely specifies the j^{th} membership function associated with the input xi. Thus the set of all end-point pairs

$\{ (a_{ij}^-, a_{ij}^+), \quad i = 1, 2 \text{ and } j = 1,2, \ldots K,$

completely describes the K membership functions associated with each of the 2 inputs. The structure of the membership functions restricts the absolute value of slope to be equal for all membership functions in the same universe of discourse, U_i. Also, we note from the figure that each input will be matched to exactly two membership functions in U_i. Restrictions that we place on our design enable us to store all end-points associated with our membership functions and up to 49 rules in our knowledge base in a 64-byte static RAM.

In our design, the implication and inference operations are evaluated by using product operators. This approach has been shown to yield very good results in a number of engineering applications [9]. A centroid defuzzification scheme is used to determine the output of the FLS coprocessor chip.

The design procedure for the FLS coprocessor is outlined through the following four steps:

We begin by defining K fuzzy sets associated with each universe of discourse, U_i (i= 1,2) by specifying the end-point pairs [a;, a_{ij}^+] as described above. The corresponding rules of our knowledge rule-base are denoted as $M_{F_{ij}}^L$ (L= 1,2, . . . ,m=49). We use symmetric triangular membership functions of the form

$$M_{F_{ij}^L}(x_i) = \begin{cases} 1 - \dfrac{|xi - a_{ij}|}{C_{ij}} & \text{for } |x_i| \le C_{ij} \\ 0 & \text{Otherwise} \end{cases} \qquad \text{.....(1)}$$

where C_{ij} is a normalizing constant to control the slope

i = 1, 2

j = 1, 2, …, K=7

L= 1,2, ..., m=49.

2) Next, we construct a set of IF-THEN fuzzy rules in the following form:

$$L_1 = \text{IF } x_1 \text{ is } F_{1j}^1 \text{ and } x_2 \text{ is } F_{2j}^1; \text{ Then } m_1 \text{ is } Q'$$

.....

$$L_m = \text{IF } x_1 \text{ is } F_{1j}^m \text{ and } x_2 \text{ is } F_{2j}^m; \text{ Then } m_m \text{ is } Q\text{“} \qquad \text{----------(2)}$$

where y_i is the consequent associated with rule L_i. Reference [8] provides more detail for this step.

3) Construct the fiber F: U → R based on the M rules of step 2 as follows:

$$F(x) = \sum_{L=1}^{m} Q^L \prod_{j=1}^{k} \prod_{h=1}^{k} (M_{F_{ij}^L}(x_1) M_{F_{2h}^L}(x_2)) \qquad \text{---------(3)}$$

In this step, we form products for each of the pairs of strengths associated with each fuzzy set in each universe of discourse. We note that due to the structure imposed through our fuzzy membership functions, that the majority of these products will be zero and the denominator of products will be unity. Also, we note that the terms Q^L (L=1, 2, ... 49) are free parameters and the filter is nonlinear.

4) At this step, we use the following LMS algorithm to update the filter parameters Q^L as specified in step 2. At each time points= 1,2, . . . we perform the following adaptation:

$$Q^L(s) = Q^L(s-1) + \alpha[O_d(s) - F(x(s))]\left(\prod_{j=1}^{k} \prod_{h=1}^{k} (M_{F_{ij}^L}(x_1) M_{F_{2h}^L}(x_2))\right) \qquad \text{----------(4)}$$

where $a \leq 1$ is our learning factor and $O_d(s)$ denotes the desired output. Minimization of the LMS cost function

$$K = E\left\{(O_d(s) - F(x(s)))^2\right\}$$

ensures that the input sequence $\mathbf{x[s]}$ optimally matches the desired output sequence $O_d(s)$ at each time point $s = 1,2, \ldots$. Finally, a graphical representation shown by Fig. 1 summarizes all the previous steps.

Therefore, the filter $F(\mathbf{x}(s))$ given in Eq. 3 can match any input-output pair $[x(s);O_d(s)]$ to arbitrary accuracy by properly choosing the parameters Q^L. However, this is the only degree of freedom we have available during the adaptation procedure because the end-points of the symmetric triangular membership functions, (a_{ij}^-, a_{ij}^+), are chosen according to a maximum input limit before the adaptation takes place.

III. ASICs Design of the FLS Coprocessor

The FLS coprocessor layout given in Fig. 3 is based on the mathematical description presented in the previous section and the concepts of Fig. 1. We obtain 8-bit real-time operation by processing all computations in parallel with two levels of pipelines separated by a high-speed 8-bit storage buffer. The 8-bit high-speed signed parallel comparator, given in Fig. 4, compares the released input value with all the end-points of the symmetric triangular membership functions. There are seven of these end-points. The comparator releases the upper and lower addresses of the matched membership values as well as the end-point which is greater than or equal to the input value.

The addresses released from the comparator are stored in a 3-bit D-type flip-flop register, where they are concatenated via a 3-bit multiplexer to generate a 6-bit address bus. This maps to a designated rule-base location stored in the 64-byte static RAM. The matched membership function degree α_i of the given input is calculated by subtracting the input from the release end-point and multiplying the result by the appropriate positive slope. Since we have two matched membership functions for any given input value and the membership functions are normalized to one, the other degree is simply evaluated by assessing the inverse of the first evaluated degree. Four 8-bit D-type flip-flops are used as a temporary storage during this computation.

The computation of the second stage of the pipe-line is achieved by cross multiplying the matched degree values from the given two inputs with the appropriately retrieved rule-base values. These results are then aggregated to produce the de fuzzified crisp output according to Eq. 3. In order to speed up the computations during the two stages of the pipelines, the 8-bit signed adder/subtracter has been designed with two stages 4-bit carry look-ahead structure while the 8-bit signed integer multiplier is designed with Wallace trees structure, which are described in [10].

The components are designed using the VHDL language and optimized by Autologic 11 (Mentor Graphics EDA design tool). Table 1 illustrates the area and the delay of the coprocessor components optimized under smallest area for the Alters technology implementation and smallest area and fastest time for implementation using the 1.2μ CMOSN technology. The 1.2μ technology provides less delay in terms of its critical path analysis and thus allows the circuit to run at 25 Mhz while still providing an output every 0.55 μs. Using the 1.2u CMOSN technology, we can fabricate the circuit on a single chip with a dimension of 3*3 mm

IV. Application to Adaptive Noise Cancellation

Although the hardware of the FLS coprocessor chip design is simple, the structure itself can incorporates a wide class of applications based upon LMS adaptive filter approaches. We will describe how the structure can support applications related to interference canceling using the LMS approach.

As the name implies, adaptive noise cancellation is based upon subtracting noise from a received signal. Here the operation is controlled in an adaptive manner for the purpose of improving the signal-to-noise ratio. Fig. 2 shows the general model for an adaptive noise canceler which employs dual inputs and a closed loop adaptive feedback system. The two inputs to the system are derived from a pair of sensors: a primary sensor and a reference (auxiliary) sensor. The primary input supplies an information-bearing signal and a sinusoidal interference which are uncorrelated with one another. The reference input supplies

a correlated version of the sinusoidal interference. The input data is assumed to be real valued such that the primary input can be modeled as:

$B(n) = d(n) + A_o \cos(w_o n + \Phi n)$

where d(n) is an information-bearing signal which is characterized by an Autoregresive process $d[n] = v(n) - 0.8458 d[n-1]$ such that v(n) is a white-noise process with zero mean and variance $\sigma^2 = O$. Here, A_o is the amplitude of the sinusoidal interference, w_o is the normalized angular frequency, and Φ_0 is the phase. The reference input is given as $U(n) = A\cos(w_o n + \Phi)$ where the amplitude A and the phase Φ are different from those in the primary input but the angular frequency w_o is the same. Consequently, applying the adaptive process presented in Eq. 4 (section 2), the results are depicted in Graph 1 for different values for the learning factor.

V. Conclusions

By incorporating symmetric triangular membership functions, the coprocessor FLS chip offers a number of significant advantages. It does not require the use of division components. This is attributable to the symmetrical unity in the denominator of Eq. 3 (see [4] for proof). This in turn accelerates computations and minimizes the area needed to implement the chip. In addition, the symmetric triangular membership structure provides a simple and effective means of storing membership functions via their end-points. This enables us to compute strengths through trivial algebraic computations and allows for easy and fast memory access.

The coprocessor can store a knowledge rule-base of 49 rules and can produce a final output for the case of two input variables every 0.55 μs through two pipeline stages using a 20 Mhz internal clock. All computations are performed through an 8-bit data bus segmented to a 4-bit fixed point arithmetic decimal point. Though the chip is limited to a single class of membership functions and performs implication, inference and defuzzification in only one manner, it is versatile enough to support a wide range of applications. Its knowledge rule-base can be adaptively altered to achieve optimized results by employing a learning algorithm.

Acknowledgment

We wish to extend appreciation to NASA for its support of this work through cooperative agreement NCCW-0089.

References

[1] P. J. King and E.H. Madani, "The Application of fuzzy control systems to industrial process," Automatic, vol. 13, no. 3, pp. 235-242, 1977.

[2] S. -G. Kong and B. Kosco, "Adaptive fuzzy system for backing up a truck-and-trailer," IEEE Trans. Neural Networks, vol. 3, no. 2, pp. 211-223, March 1992.

[3] L. X. Wang and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least squares learning," IEEE Trans. on Neural Networks, vol. 3, no. 5, pp. 807-814, 1992.

[4] J. W. Fattaruso, S.S. Mahant-shetti, and J. B. Barton, "A fuzzy logic inference processor," IEEE J. Solid-state Circuits, vol. 29, No. 4, pp. 397-401, April 1994.

[5] H. Watanabe, W. D. Dettloff, and K. E. Yount, "A VLSI fuzzy logic controller with reconfigurable, cascadable architecture," IEEE J. Solid-state Circuits, vol. 25, no. 2, pp. 367-382, April 1990.

[6] T. Yamakawa, "A fuzzy inference engine in nonlinear analog mode and its application to a fuzzy logic control, " IEEE Trans. Neural Net., vol. 4, no. 3, pp. 496-521, May 1993.

[7] V. Catania, A. Piuliafito, M. Reuse, and L. Vita, "A VLSI fuzzy inference processor based on discrete analog approach," IEEE Trans. on Fuzzy Systems, vol. 2, no. 2, pp. 93-106, May 1994.

[8] S. M. Abdel-hafeez and S. Starks," A VLSI modified architecture for reduced symmetric fuzzy singleton set and its applications," ***in application and science of artificial neural nets II, S. K. Rogers and D. W. Puck: Editors,*** Proc. SPIE, vol. 2760, April 1996.

[9] J. M. Mendel," Fuzzy logic systems for engineering; A Tutorial," Proc. of the IEEE, vol. 83, no. 3, March 1995.

[10] J. P. Hayes (1994), Computer Architecture and Organization, 2nd cd., McGraw-Hill, New York.

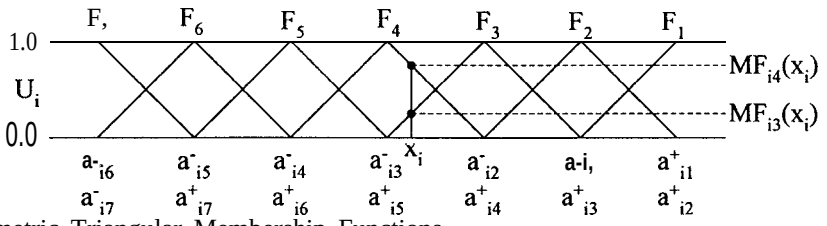
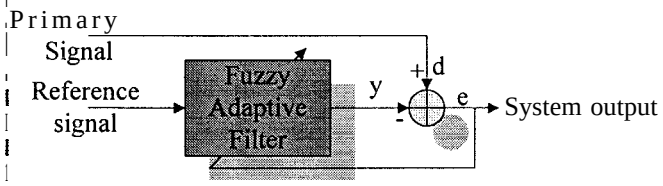
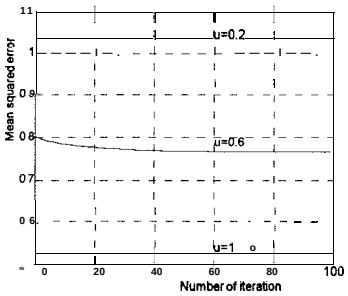


Fig. 1. Symmetric Triangular Membership Functions



1 Fig. 2. A Fuzzy adaptive filtering applications for inference noise cancellation.



Graph. 1. Transient behavior of mean squared error in fuzzy adaptive fisrt-order inference noice canceller for different learning factors:

Table 1. Gate counts and maximum delay for fuzzy coprocessor in 1.2u CMOSN and Altera technology

Coprocessor Components	CMOS Transistor Counts	ALTERA Gate Counts	CMOS Maximum Delay (ns)	Altera Maximum Delay (ns)
8-bit Signed Adder/Subtracter	310	45	15.433	28.9
8-bit Signed Special Comparator	2,104	269	19.998	40.9
8-bit Signed Multiplier	2,126	367	29.75	74.7
6X64-byte Signed RAM	28, 230	3329	10.1	10.4

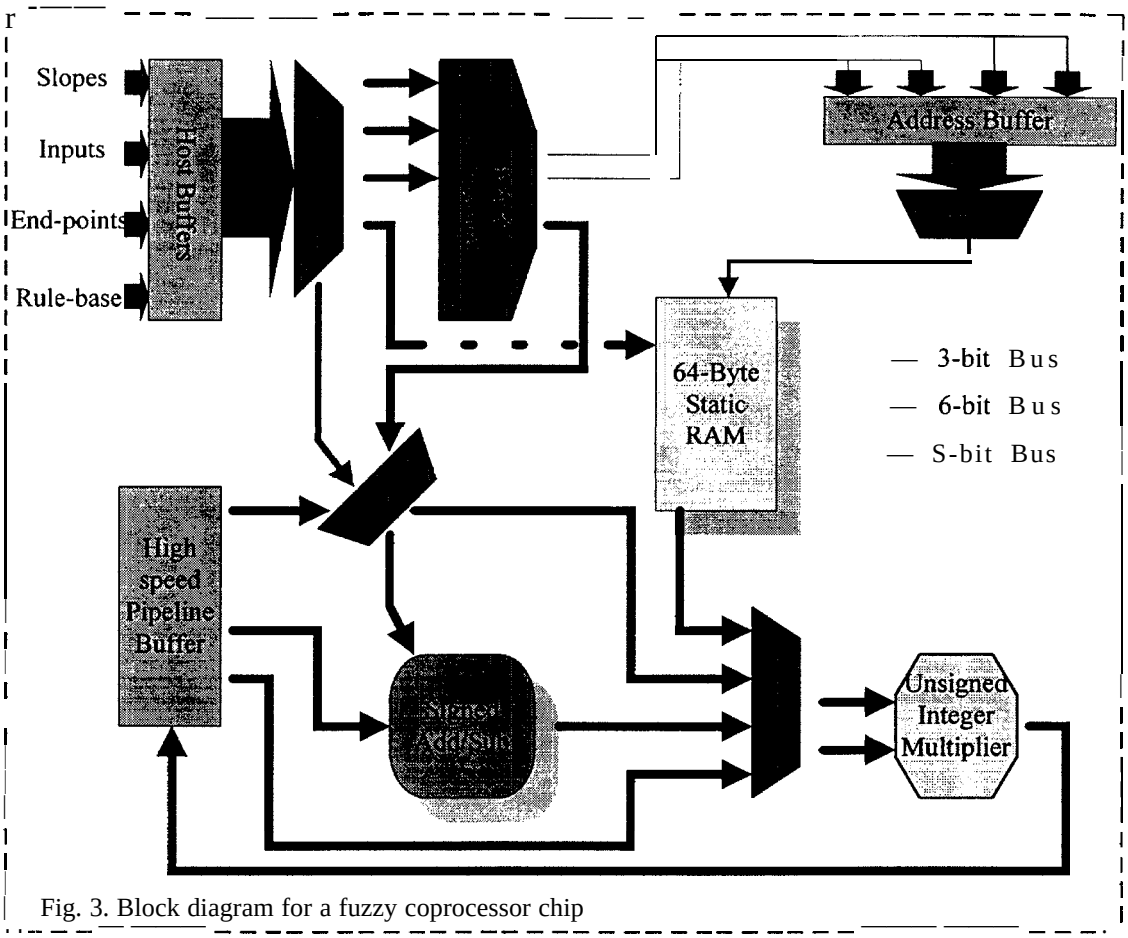


Fig. 3. Block diagram for a fuzzy coprocessor chip

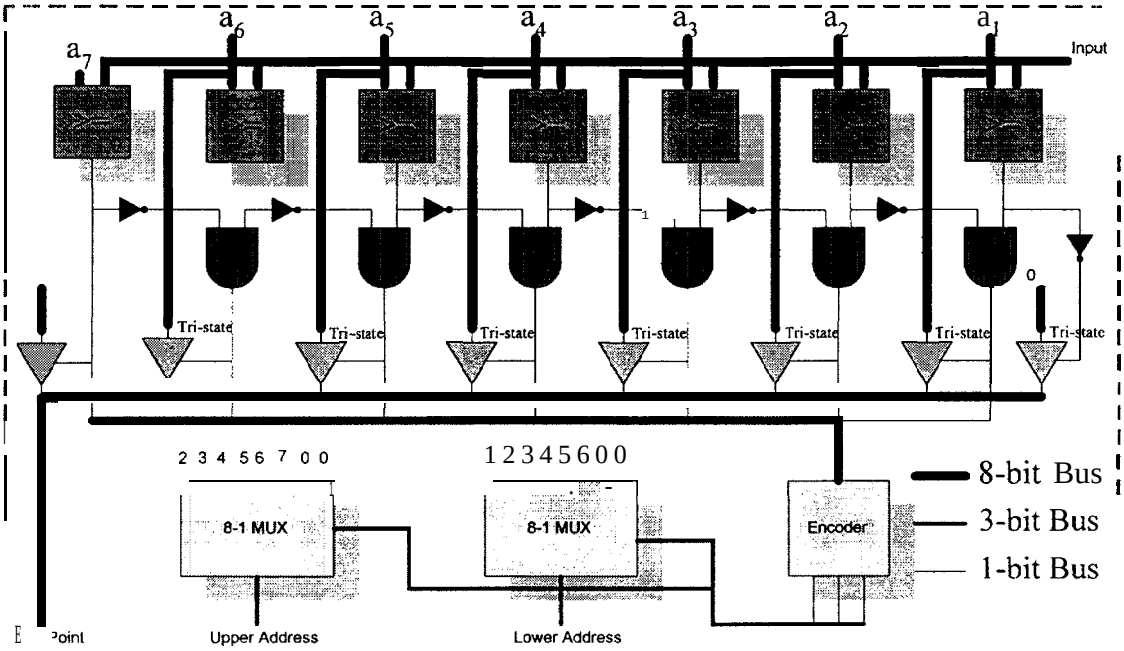


Fig. 4. Block diagram for high speed 8-bit signed comparator