HIGH-SPEED FUZZY INFERENCE PROCESSOR USING ACTIVE RULES IDENTIFICATION

Shih-Hsu Huang, Shi-Zhi Liu, Yi-Rung Chen, and Jian-Yuan Lai
Department of Electronic Engineering,
Chung Yuan Christian University,
Chung Li, Taiwan, R.O.C.

ABSTRACT

Once the input values are given, the active rules in a fuzzy inference execution have been determined. Based on the observation, our approach is to identify the active rules before fuzzy inference execution. To achieve this goal, our architecture provides the following two mechanisms: (1) a mechanism to ignore the non-active rules before fuzzy inference execution; and (2) a mechanism to arrange the active rules for fuzzy inference execution. The proposed architecture has been implemented using a 0.35μm cell library. Implementation data show that, if the number of active rules is only 4, the inference speed can achieve 23.755 MFLIPS. To the best of our knowledge, our approach is the fastest hardware implementation.

1. INTRODUCTION

Fuzzy logic [1] has been successfully employed in many complex applications. With the progress made in real-time applications, the inference speed required may reach up to the range of Mega fuzzy logic inferences per second (MFLIPS). Several hardware architectures [2-6] have been proposed to support general-purpose fuzzy inference execution at higher speed. However, these architectures have the following property: *in each fuzzy inference execution, all the rules in the knowledge base are required.* As a result, the inference speed is limited by the total number of rules in the knowledge base.

In fact, many fuzzy applications have the following property: *in an individual fuzzy inference execution, the active rules are only a small part of the total rules.* For example, as the fuzzy systems described in [7-10], the maximum number of active rules among all the combinations of input values is only four. Since the non-active rules make null contribution to a fuzzy inference execution, the non-active rules can be ignored. Based on this observation, some hardware architectures [11,12] have been proposed to ignore the non-active rules in the earlier stage of the pipeline. However, they [11,12] still need to compute the weight of each rule during the fuzzy inference execution; otherwise, they cannot determine the set of active rules.

In this paper, we propose a new architecture for the fuzzy applications, whose active rules are few in each fuzzy inference execution. Note that, once the input values are given, the active rules in a fuzzy inference execution has been determined. Based on this property, our approach only extracts the active rules according to the input values. Compared with previous hardware architectures [11,12], the main advantage of our approach is that: previous hardware architectures ignore the non-active rules *during* the fuzzy inference execution, whereas our approach ignores the non-active rules *before* the fuzzy inference execution.

Following the same specification in [12], each membership function is assumed to be trapezoid-shaped. The new features in this paper are below:

(1) The rules in the knowledge are sorted in the sequence of their antecedent membership functions. Therefore, we can use binary search strategy to extract the active rules with respect to the input values.
(2) The set of active rules are dependent on the input values. Therefore, to handle the dynamic condition, we design a scheduling unit to arrange 4 active rules to enter the fuzzy inference execution per pipeline stage cycle.

2. MOTIVATION

If the weight of a rule is zero, it has null contribution to the fuzzy inference. Therefore, when the input values are given, we say that a rule is *active* if and only if the weight of this rule is positive. Note that, in each fuzzy inference execution, only a part of the total rules are active.

Let's use the following fuzzy system as an example. The fuzzy system has two input variables: X and Y. The antecedent membership functions of input variable X are A_0, A_1 and A_2, respectively. The antecedent membership functions of input variable Y are B_0, B_1 and B_2, respectively. Figure 1 shows the antecedent membership functions. Therefore, the fuzzy system has 9 rules as follows.

IF (X is A_0) and (Y is B_0) THEN …
IF (X is A_0) and (Y is B_1) THEN …
IF (X is A_0) and (Y is B_2) THEN …
IF (X is A_1) and (Y is B_0) THEN …
IF (X is A_1) and (Y is B_1) THEN …
IF (X is A_1) and (Y is B_2) THEN …
IF (X is A_2) and (Y is B_0) THEN …
IF (X is A_2) and (Y is B_1) THEN …
IF (X is A_2) and (Y is B_2) THEN …

If the fuzzified inputs of variables X and Y are X_1 and Y_1, respectively, then the active rules as follows:

IF (X is A_0) and (Y is B_0) THEN …
IF (X is A_0) and (Y is B_1) THEN …
IF (X is A_1) and (Y is B_0) THEN …
IF (X is A_1) and (Y is B_1) THEN …

If the fuzzified inputs of variables X and Y are X_2 and Y_2 respectively, then the active rules as follows:

IF (X is A_0) and (Y is B_1) THEN …
IF (X is A_0) and (Y is B_2) THEN …
IF (X is A_1) and (Y is B_1) THEN …
IF (X is A_1) and (Y is B_2) THEN …

If the fuzzified inputs of variables X and Y are X_2 and Y_1, respectively, then, the active rules as follows:

IF (X is A_1) and (Y is B_0) THEN …
IF (X is A_1) and (Y is B_1) THEN …
IF (X is A_2) and (Y is B_0) THEN …
IF (X is A_2) and (Y is B_1) THEN …

If the fuzzified inputs of variables X and Y are X_2 and Y_2, respectively, then the active rules as follows:

IF (X is A_1) and (Y is B_1) THEN …
IF (X is A_1) and (Y is B_2) THEN …
IF (X is A_2) and (Y is B_1) THEN …
IF (X is A_2) and (Y is B_2) THEN …

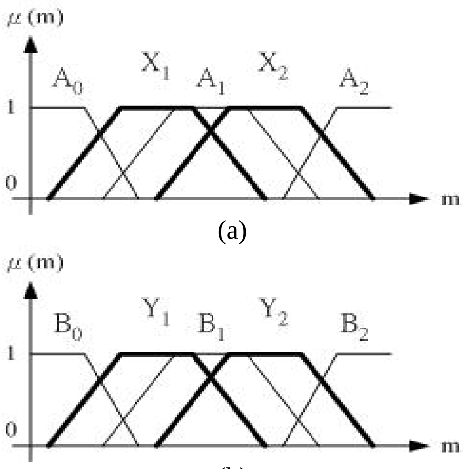


Figure 1: Intersection between fuzzy input variables and membership functions.

In this example, we observe that: for each input variable, the number of membership functions that are overlapped with a fuzzified input is only 2. Therefore, in each fuzzy inference execution, the number of active rule is only 4 (i.e., 2*2). Especially, with the increase of the total rules, the percentage of active rules is very low. Based on the above discussion, we know that the active rules are only a part of the total rules. If we can ignore the non-active rules before fuzzy inference execution, the performance can be significantly improved.

3. THE PROPOSED ARCHITECTURE

The proposed fuzzy inference processor has two inputs and one output. For each input, the maximum number of sets of membership functions is 32. For each output, the maximum number of sets of membership functions is 16. The maximum number of fuzzy rules is 1024. To speed up the process of fuzzy inference, we apply parallel and pipeline structure to design the fuzzy inference processor.

Our architecture is designed based on [12]. We divide the fuzzy inference processor into 8 pipeline stages: Fuzzifier (fuzzy rule database), Detection, Scheduling Unit, Fuzzy Decoder, Access Rule, Fuzzy Decision, Maximum Unit, Accumulator and Divisor. Each pipeline stage needs 8 clock cycles to complete its process. Note that the last six pipeline stages of our fuzzy inference processor are the same as [12]. Due to the limitation of pages, in this paper, we only introduce the first two pipeline stages of our fuzzy inference process.

3.1 FUZZY RULE DATABASE

In order to match up our fuzzy inference processor structure, the membership function and the fuzzy rule database are designed as fixed forms. In the design of membership function, we have some constrains. As shown in Figure 2, if i is smaller than j, then A_{ia} must be equal to or smaller than A_{ja}, and A_{id} must be equal to or smaller than A_{jd} (A_{ib} is not necessarily equal to or smaller than A_{jb}, and A_{ic} is not necessarily equal to or smaller than A_{jc}).

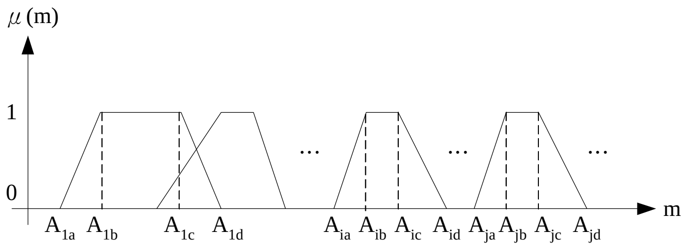


Figure 2: Membership functions

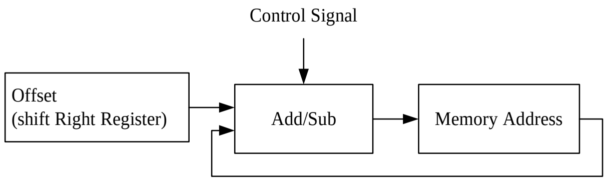
Furthermore, the membership functions in the fuzzy rule database are sorted according to the antecedent part of fuzzy rules. Table 1 gives a fuzzy rule database example, in which the membership functions of variables X and Y are {A_0, A_1, A_2} and {B_0, B_1, B_2}, respectively.

	Antecedent	
	A_i	B_i
R_0	A_0	B_0
R_1	A_0	B_1
R_2	A_0	B_2
R_3	A_1	B_0
R_4	A_1	B_1
R_5	A_1	B_2
R_6	A_2	B_0
R_7	A_2	B_1
R_8	A_2	B_2

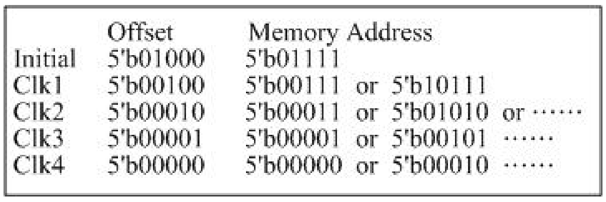
Table 1: The antecedent part of fuzzy rule database.

3.2 DETECTION

The detection unit is to find intersection between membership functions and input variables. It is time-consuming if we sequentially check the intersection between membership functions and input variables. Since our design of fuzzy inference processor has most 32 sets of membership function at the antecedent part., we use binary search strategy as shown in Figure 3. The detection unit is to find: (1) the address of starting membership function (*StartA*) and the address of end membership function (*EndA*) that intersects with input variable *X*; (2) the address of starting membership function (*StartB*) and the address of end membership function (*EndB*) which are intersected simultaneously with input variable *Y*. In Figure 3 (a), we use two registers to store the offset between the present memory address and the next memory address. The *Control Signal* is used to control the Add/Sub to perform addition or subtraction. Figure 3 (b) illustrates the change of two registers in Figure 3 (a), in which the notation Clk1 denotes the first clock cycle, Clk2 denotes the second clock cycle, and so on.



(a)



	Offset	Memory Address
Initial	5'b01000	5'b01111
Clk1	5'b00100	5'b00111 or 5'b10111
Clk2	5'b00010	5'b00011 or 5'b01010 or ……
Clk3	5'b00001	5'b00001 or 5'b00101 ……
Clk4	5'b00000	5'b00000 or 5'b00010 ……

(b)

Figure 3: Architecture of binary sSearch.

Figure 4 gives the circuit used in the detection unit to perform the binary search strategy for finding the address of starting membership function that intersects with input variable *X* (StartA). In Figure 4, when the *Rnew* signal is 0, it will obey the rule data and rule address given to update the corresponding membership function; when the *Rnew* signal is 1, it will access the membership function corresponds to the address of starting membership function which obtained from binary search.

Let's consider an input variable *X* that has five membership functions: A0, A1, A2, A3 and A4. Suppose that the elements of A0, A1, A2, A3 and A4 are (0, 0, 5, 10), (5, 10, 15, 20), (10, 15, 20, 25), (25, 30, 35, 40), and (35, 40, 45, 50), respectively. When the element of input variable X is (13, 14, 15, 16), we can use the following binary search strategy to find the address of starting membership function (*StartA*) that intersects with input variable *X*:

1. At the beginning, the initial values of memory address and offset are 2 and 2, respectively. Because the membership function A2 intersects with input variable *X*, the address of starting membership function (*StartA*) that intersects with input variable *X* have to be smaller than or equal to 2. The *Control Signal* sends out *Sub* signal at the same time. Consequently, memory address is 0 and offset is 1 at next global clock cycle.

2. Due to the intersection between membership function A0 and input variable *X* is a null set and the place of A0 is in the front of input variable *X*, the address of starting membership function (*StartA*) that intersects with input variable *X* have to be bigger than 0. The *Control Signal* sends out *Add* signal simultaneously. Consequently, memory address and offset are updated to 1 and 0, respectively, at next global clock cycle.

3. The address of starting membership function (*StartA*) that intersects with input variable *X* is updated to 1, since A1 that intersects with input variable X.

4. We can obtain *StartB* using similar steps. The *StartB* is updated to 2.

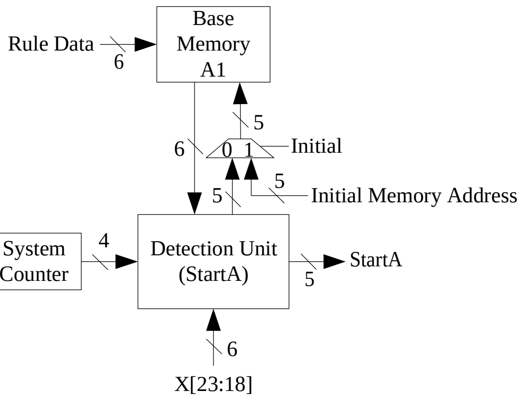


Figure 4: Detection unit.

4. IMPLEMENTATION RESULTS

The proposed fuzzy inference processor has been implemented by using a 0.35μm cell library. Through verification and timing analysis, the clock rate of the global clock is up to 190MHz. Because a pipeline stage takes 8 global clock cycles, the maximum performance of the proposed architecture is 23.75 MFLIPS. Table 2 depicts the maximum performance of fuzzy inference processor under different number of active rules.

Table 3 tabulates the comparisons of our approach with other hardware architectures, including [2], [4], [5], [6], [11], and [12]. Note that, due to the parallel processing, the number of inputs (outputs) has almost no influence on the circuit performance. Therefore, even though the input numbers of these architectures are not the same, we still can compare them. Furthermore, although these architectures are implemented in different process technologies, we find that the improvement of our approach is very significant. Therefore, our approach is the fastest hardware implementation.

We use the proposed fuzzy processor to implement two control systems, including backing-up control system and cart-pole balancing. Figure 5 gives the control surface of backing-up control system. Figure 6 gives the control surface of cart-pole balancing.

Number of active rules	Performance (MFLIPS)
0~4	23.75 MFLIPS
5~8	11.85 MFLIPS
9~12	7.92 MFLIPS

Table 2: Processor Performance.

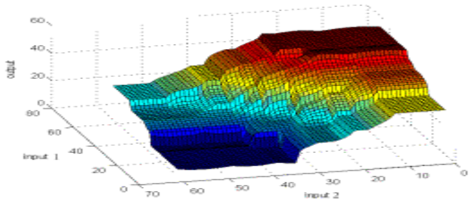


Figure 5: Control surface of backing-up control system.

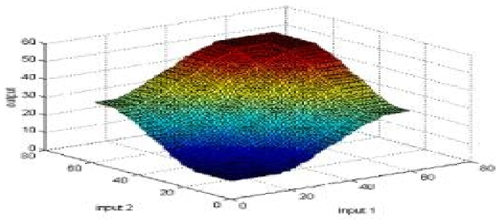


Figure 6: Control surface of cart-pole balancing.

5. CONCLUSION

In this paper, we present a high-speed VLSI fuzzy inference processor with rule analysis. The maximum frequency reaches to 190MHz, and the maximum performance reaches p to 23.75 MFLIPS (Mega Fuzzy Logic Inferences Per Second). Compared with the existing hardware implementations, our approach is the fastest hardware implementation.

6. REFERENCE

[1] L.A. Zadeh, “Fuzzy sets,” Inf. Control, vol.8, pp. 338-351, 1965.

[2] H. Watanabe, W.D. Dettloff and K.E. Yount, “A VLSI Fuzzy Logic Controller with Reconfigurable, Cascade Architecture”, IEEE Journal of Solid-State Circuits, vol. 25, pp. 376-381, 1990.

[3] A. Gabriellu and E. Gandolfi, “A Fast Digital Fuzzy Processor”, IEEE Micro, pp. 68-79, vol. 19, no. 1, 1999.

[4] J.M. Jou and P.Y. Chen, “An Adaptive Fuzzy Logic Controller: Its VLSI Architecture and Applications”, in IEEE Trans. on VLSI systems, vol. 8, pp 52-60, 2000.

[5] S.H. Huang and J.Y. Lai, “A High Speed VLSI Fuzzy Logic Controller with Pipeline Architecture”, Proc. of IEEE International Conference on Fuzzy Systems, vol. 3, pp.1054-1057, 2001.

[6] S.H. Huang and J.Y. Lai, “A High-Speed VLSI Fuzzy Inference Processor for Trapezoid-Shaped Membership Functions”, Journal of Information Science and Engineering. vol. 21, no. 3, pp. 607-626, 2005.

[7] D.V. Cleave and K.S. Rattan, “Tuning of Fuzzy Logic Controller using Neural Network”, Proc. of National Aerospace and Electronics Conference, pp. 305—312, 2000.

[8] V. Kunsriraksakul, B. Homnan and W. Benjapolakul, “Comparative Evaluation of Fixed and Adaptive Soft Handoff Parameters Using Fuzzy Inference Systems in CDMA Mobile Communication Systems”, Proc. of IEEE International Conference on Vehicular Technology, vol. 2, pp. 1017—1021, 2001.

[9] H. Zhuang and X. Wu, “Membership Function Modification of Fuzzy Logic Controllers with Histogram Equalization”, IEEE. Trans. on Systems, Man, Cybernetics, vol. 31, pp. 125—132, 2001.

[10] B.D. Liu, C.Y. Chen and J. Y. Tsao, “Design of Adaptive Fuzzy Logic Controller Based on Linguistic-Hedge Concepts and Genetic Algorithms”, IEEE Trans. on System, Man and Cybernetics, vol. 31, pp. 32—53, 2001.

[11] G. Asica, V. Catania and M. Russo, “VLSI Hardware Architecture for Complex Fuzzy Systems”, IEEE Trans. on Fuzzy Systems, vol. 7, no. 5, pp. 553-570, 1999.

[12] S.H. Huang and J.Y. Lai, “A High Speed Fuzzy Inference Processor with Dynamic Analysis and Scheduling Capabilities”, IEICE Transactions on Information and Systems, vol. 88-D, no. 10, pp. 2410-2416, 2005.

		Ours	[2]	[4]	[5]	[6]	[11]	[12]
Specification	Inputs	**2**	4/2	2	2	2	3	2
	Outputs	**1**	2/1	1	1	1	1	1
	Rules	**1024**	102	50	64	64	18	1024
Implementation	Technology	**0.35μm**	1.10μm	0.8μm	0.35μm	0.35μm	0.5μm	0.35μm
	Performance (MFLIPS)	**23.75**	0.58	0.49	2.50	7.00	3.30	4.00

Table 3: Comparisons among different architectures.