DOI: 10.35598/mcfpga.2020.011

Fuzzy Logic Custom Instruction Set for NIOS II Processor

Sergey A. Ivanets
ORCID 0000-0002-9587-0783
Department of Biomedical Radio-Electronic Devices and Systems
Chernihiv National University of Technology
Chernihiv, Ukraine
Sergey.Ivanets@gmail.com

Artem P. Fesenko
ORCID 0000-0001-8730-3327
Department of Biomedical Radio-Electronic Devices and Systems
Chernihiv National University of Technology
Chernihiv, Ukraine
gudrunas.ch@gmail.com

Oleksandr M. Fesiuk
Department of Biomedical Radio-Electronic Devices and Systems
Chernihiv National University of Technology
Chernihiv, Ukraine
fesuks1@gmail.com

***Abstract*—The article describes a way to implement operations on fuzzy sets using additional processor instructions. As a target, a NIOS II soft processor is used. Due to the hardware implementation of instructions, the speed of their execution increases significantly. The integration of fuzzy instructions into the NIOS II processor instruction set simplifies the process of developing programs that use fuzzy inference algorithms.**

Keywords*—*fuzzy logic, FPGA, custom instruction, NIOS II.

I. Introduction

The idea of developing processors with fuzzy logic is based on fuzzy mathematics. The mathematical theory of fuzzy sets was introduced by L. A. Zade [1]. Since its introduction, it was under intensive research and, as a result, have opened wide opportunities for system analytics. Based on fuzzy sets theory systems, correspondingly, have widened the application of fuzzy logic. Unlike the traditional formal logic, that operates with accurate and clear definitions such as true and false, yes and no, zero and one, fuzzy logic deals with values in certain analog or discrete range called linguistic variables, for example, “far”, “close”, “warm”, “cool”. Fuzzy logic control algorithms are called fuzzy inference systems.

Fuzzy inference systems have been successfully applied in fields such as automatic control, data classification, decision analysis, expert systems, and computer vision. Because of its multidisciplinary nature, fuzzy inference systems are associated with a number of names, such as fuzzy-rule-based systems, fuzzy expert systems, fuzzy modeling, fuzzy associative memory, fuzzy logic controllers, and simply (and ambiguously) fuzzy systems [2].

Embedded systems increasingly use FPGAs due to their superior cost and flexibility compared to custom integrated circuits [3]. FPGA systems often incorporate two types of processors, soft and hard. Soft processors are implemented using the fabric itself. Hard cores are fabricated separately and could offer higher performance compared to soft processors, but they are inflexible and wasted when not needed. Accordingly, there is a need to develop soft cores that provide high performance. The ways of improving the performance of soft processors were researched in [1]. According to research, among different techniques of improving the performance of soft processors, the most effective one is hardware processor system extension according to specific application domain. This approach is capable of bringing up to 100x performance improvement. There are two ways of implementing processor system extension: custom peripherals and custom processor instructions. Custom instructions approach if preferred way, if performance increase is considered in [4].

II. Fuzzy logic

The typical structure of fuzzy inference system or fuzzy controller includes the following components (Fig. 1):

- fuzzification unit;
- inference mechanism unit;
- rule base unit;
- defuzzification unit.

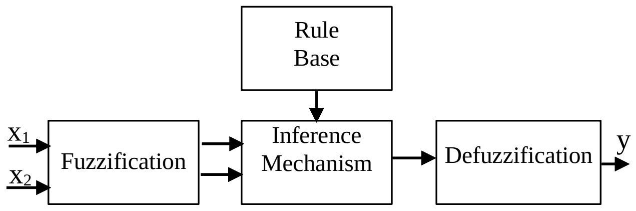

Fig. 1. The structure of typical fuzzy controller.

That is, according to the structure, fuzzy logic controller involves four main stages: fuzzification, rule base, inference mechanism and defuzzification. Fuzzification unit is responsible for converting real world crisp signals into fuzzy values for further processing. The inference mechanism determines the matching degree of the current fuzzy input with respect to each rule and decides which rules are to be fired according to the input field. Next, the fired rules are combined to form the control actions [5].

During the fuzzification step, the current system input values are compared against stored input membership functions to determine the degree to which each label of each system input is true. This is accomplished by finding the y-value for the current input value on a membership function for each label of each system input. A membership function shows the relationship between the precision (digital) input value and the fuzzy variable. Membership functions can be of various shapes, for example, triangular or trapezoidal (Fig. 2). The end result of the fuzzification step is a table of fuzzy inputs representing current system conditions.

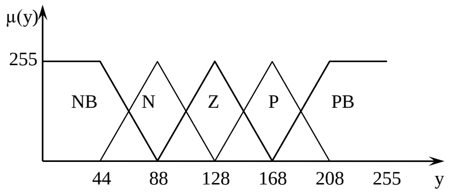

Fig. 2. Membership functions.

The use of piecewise membership functions is explained by the simplicity of their hardware implementation. To determine the trapezoid membership function (fig. 3), it is necessary to use four points a, b, c, d.

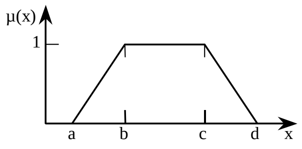

Fig. 3. Trapezoid membership functions.

Then the trapezoid membership functions will be represented by the following system:

$$\mu(x)=\begin{cases}0, x \le a\\ \dfrac{x-a}{b-a}, a \le x \le b\\ 1, b \le x \le c\\ \dfrac{d-x}{d-c}, c \le x \le d\\ 0, d \le x\end{cases} \tag{1}$$

For the duration of fuzzy inference each rule is evaluated sequentially, but the rules as a group are treated as if they were all evaluated simultaneously. Two mathematical operations take place during rule evaluation. The fuzzy AND operator corresponds to the mathematical minimum operation and the fuzzy OR operation corresponds to the mathematical maximum operation. The fuzzy ABD is used to connect antecedents within a rule. The fuzzy OR is implied between successive rules. Before evaluating any rules, all fuzzy outputs are set to zero (meaning not true at all). As each rule is evaluated, the smallest (minimum) antecedent is taken to be the overall truth of the rule. This rule truth value is applied to each consequent of the rule (by storing this value to the corresponding fuzzy output) unless the fuzzy output is already larger (maximum). If two rules affect the same fuzzy output, the rule that is most true governs the value in the fuzzy output because the rules are connected by an implied fuzzy OR.

Rule base in fuzzy controller consists of IF-THEN rules. These IF-THEN rule statements are used to formulate the conditional statements that comprise fuzzy logic.

A single fuzzy IF-THEN rule assumes the form:

if x is A then y is B

where A and B are linguistic values defined by fuzzy sets on the ranges (universes of discourse) X and Y, respectively. The IF-part of the rule “x is A” is called the antecedent or premise, while the THEN-part of the rule “y is B” is called the consequent or conclusion.

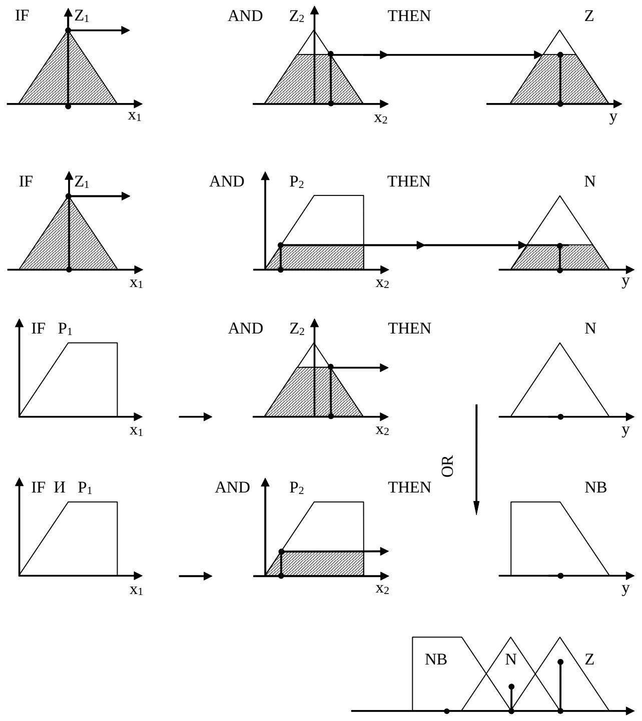

Fig. 4. Fuzzy inference.

Interpreting IF-THEN rules is a three-part process (fig. 4):

1. Fuzzify inputs: Resolve all fuzzy statements in the antecedent to a degree of membership between 0 and 1. If there is only one part to the antecedent, then this is the degree of support for the rule.

2. Apply fuzzy operator to multiple part antecedents: If there are multiple parts to the antecedent, apply fuzzy logic operators and resolve the antecedent to a single number between 0 and 1. This is the degree of support for the rule.

3. Apply implication method: Use the degree of support for the entire rule to shape the output fuzzy set. The consequent of a fuzzy rule assigns an entire fuzzy set to the output. This fuzzy set is represented by a membership function that is chosen to indicate the qualities of the consequent. If the antecedent is only partially true, (i.e., is assigned a value less than 1), then the output fuzzy set is truncated according to the implication method.

III. System-On-Chip with NIOS II processor

FPGA-based System-On-Chip (SOC) use processor cores. With Altera/Intel FPGA we can use NIOS II soft processor [6]. To work with this processor using the Quartus Prime software we need to create a microprocessor system. Figure 3 describes the functional diagram of simple microprocessor system with NIOS II soft processor. In addition to the NIOS II processor, the smallest system has a timer (T), RAM and ROM for data and instruction, JTAG-UART for debagging and download software and PIO for connecting external devices. All these modules are connected by Avalon system bus (fig. 5).

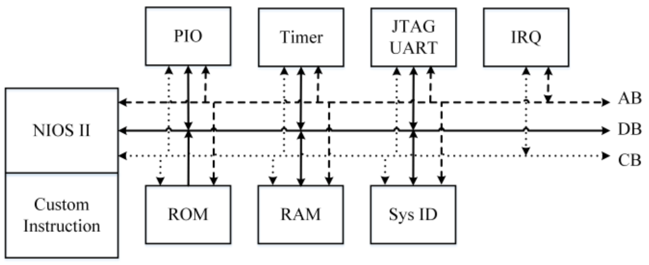

Fig. 5. Functional Diagram of NIOS II System.

All computational operations are performed by the NIOS II processor and NIOS II controls external devices using the bus Avalon, which includes: the address bus (AB), data bus (DB) and control bus (CB).

IV. Custom instruction set

User instructions ("custom instruction") is the instructions for the processor, which creates the user, which allows to significantly accelerate the speed of operation of the processor. For NIOS II processor have the ability to create up to two hundred fifty-six of such instructions. To implement fuzzy logic support in NIOS II, it is necessary to implement such instructions in the processor:

- FUZZ – fuzzification for two inputs;
- RULE – calculation base of fuzzy rules;
- DEFUZZ – defuzzification.

When the fuzzification step begins, the current value of the system input is in an accumulator of the NIOS II, one index register points to the first membership function definition in the knowledge base, and a second index register points to the first fuzzy input in RAM. As each fuzzy input is calculated by executing a FUZZ instruction, the result is stored to the fuzzy input and both pointers are updated automatically to point to the locations associated with the next fuzzy input. The FUZZ instruction takes care of everything except counting the number of labels per system input and loading the current value of any subsequent system inputs.

RULE instruction. Rule evaluation is the central element of a fuzzy logic inference program. This step processes a list of rules from the knowledge base using current fuzzy input values from RAM to produce a list of fuzzy outputs in RAM.

The complete rules are stored in the knowledge base as a list of pointers or addresses of fuzzy inputs and fuzzy outputs. For the rule evaluation logic to work, there must be some means of knowing which pointers refer to fuzzy inputs and which refer to fuzzy outputs. There also must be a way to know when the last rule in the system has been reached.

- One method of organization is to have a fixed number of rules with a specific number of antecedents and consequents.
- A second method, employed in Freescale M68HC11 kernels, is to mark the end of the rule list with a reserved value, and use a bit in the pointers to distinguish antecedents from consequents [7].
- A third method of organization, used in the Freescale HSC12, is to mark the end of the rule list with a reserved value, and separate antecedents and consequents with another reserved value [8]. This permits any number of rules, and allows each rule to have any number of antecedents and consequents, subject to the limits imposed by availability of system memory.

These fuzzy outputs can be thought of as raw suggestions for what the system output should be in response to the current input conditions. Before the results can be applied, the fuzzy outputs must be further processed, or defuzzified, to produce a single output value that represents the combined effect of all of the fuzzy outputs.

The defuzzification instruction (DEFUZZ) calculates the value that best describes the fuzzy value of the output linguistic variable. For defuzzification we use the center of gravity method, sometimes called the center of gravity method for singletons. The calculation of the sums required by the method of the center of gravity turns out to be several orders of magnitude faster than the numerical integration required in the method of the center of the region:

$$Y = \frac{\sum_{i=1}^{p} Y_i \cdot a_i}{\sum_{i=1}^{p} a_i}, \qquad (2)$$

where Yi is the value of the center of the maximum for the i-th term;

ai - weight of the i-th term.

V. Conclusions

Custom instructions are one of the benefits of software processors, as they are added directly to the processor core and instruction set. Using custom instructions in NIOS II processor significantly speeds up the operation of fuzzy control algorithms and simplifies the task of writing programs for such algorithms. This instructions increase processor size in FPGA chip, but are one of the most effective ways to speed up program execution.

References

[1] L.A. Zadeh, “Fuzzy sets” *Information and Control*, Vol. 8, Issue 3, pp. 338-353, June 1965, doi: 10.1016/S0019-9958(65)90241-X.

[2] J. Jantzen Foundations of Fuzzy Control: A Practical Approach. John Wiley & Sons, 2013. 352 p.

[3] Proektuvannya komp'yuternykh system na osnovi mikroskhem prohramovanoyi lohiky: monohrafiya / avt: V. V. Kazymyr, V. V. Lytvynov, S. A. Ivanetsʹ. – Chernihiv: Chernihivsʹkyy natsionalʹnyy tekhnolohichnyy universytet, 2013. – 305 s.

[4] Embedded Design Handbook. Intel Corp., 2020. – 497 p.

[5] C. C. Lee, "Fuzzy logic in control systems: fuzzy logic controller. I," in *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 2, pp. 404-418, March-April 1990, doi: 10.1109/21.52551.

[6] Nios II Custom Instruction User Guide. Intel Corp., 2020. – 66 p.

[7] HC11. MC68HC11F1. Technical Data. Freescale Semiconductors, 2004. 158 p.

[8] S12CPUV2 Reference Manual. HCS12 Microcontrollers. Freescale Semiconductors, 2006. 452 p.