Available online at www.sciencedirect.com

ScienceDirect

Fuzzy Sets and Systems 160 (2009) 554–568

FUZZY sets and systems

www.elsevier.com/locate/fss

Computationally efficient active rule detection method: Algorithm and architecture

Mahdi Hamzeh[a], Hamid Reza Mahdiani[a, b, *], Ahmad Saghafi[a], Sied Mehdi Fakhraie[a], Caro Lucas[c, d]

[a]*Silicon Intelligence and VLSI Signal Processing Laboratory, School of Electrical and Computer Engineering, University of Tehran, North Kargar Ave., Tehran 14395-515, Iran*
[b]*Computer and Electronics Department, Sh. Abbaspour University of Technology, Iran*
[c]*Center of Excellence for Control and Intelligent Processing, University of Tehran, Iran*
[d]*School of Cognitive Science, IPM, Iran*

Received 15 September 2007; received in revised form 22 April 2008; accepted 1 May 2008
Available online 21 May 2008

Abstract

In this paper, a new active rule detection algorithm is proposed which is efficiently implemented in dedicated fuzzy processors. Here, its advantages are analytically attested. A novel realization architecture is proposed that has higher performance and uses lower hardware resources in comparison to the other reported architectures. The structure of the proposed active rule detection unit is scalable in terms of the number of inputs, the number of membership functions and their bit widths. The proposed architecture is flexible in term of membership function shape as well.
© 2008 Elsevier B.V. All rights reserved.

Keywords: Active rule detection; Algorithm; Architecture; Fuzzy processor; Scalable

1. Introduction

Fuzzy logic [42,43] is used in an increasing number of applications. These applications include process control [25], decision making support systems [27] and signal processing [8]. The time constraints of various fuzzy systems may differ considerably according to the demands of different applications. In washing-machine controllers and auto focus imaging devices, for instance, the required inference speeds are quite low, while in real-time applications, they are extremely high. Among them, we can mention of the applications reported in [15,26,5,18,24,32]. There are some successful efforts to propose simpler algorithms to reduce the amount of computations in fuzzy processes [6] which result in higher speeds [7] and smaller hardware [28]. However, there is still need for higher performances. Thus, the designers have to use either high speed special purpose fuzzy hardware [15,2] or very high performance general purpose processors [34,4,11] and digital signal processors [16,23] to provide the necessary computational power.

* Corresponding author at: Silicon Intelligence and VLSI Signal Processing Laboratory, School of Electrical and Computer Engineering, University of Tehran, Iran. Tel.: +98 912 1907099; fax: +98 21 88006064.
E-mail addresses: mhamzeh@ece.ut.ac.ir (M. Hamzeh), mahdiani@gmail.com, mahdiany@ut.ac.ir (H.R. Mahdiani), saghafi@ieee.org (A. Saghafi), fakhraie@ut.ac.ir (S.M. Fakhraie), lucas@ipm.ir (C. Lucas).

0165-0114/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.fss.2008.05.009

Although using general purpose processor makes the system flexible, they cannot support real-time and computational intensive applications. Hence, it is unavoidable to use dedicated fuzzy processing engines to implement complicated systems such as fuzzy image filtering, real-time fuzzy traffic handling of networks, real-time fuzzy task scheduling, etc. [15,26,5,18,24,32]. To achieve the best performance of these dedicated engines, their critical path blocks should be efficiently improved and tuned [1]. The active rule selection operation which determines active rules among non-active ones is on the critical path of any fuzzy processing engine. Even a small improvement on this block results in good overall benefits as shown in the next sections.

Almost all reported conventional fuzzy processors use a similar processing structure. In these architectures, processing starts with the fuzzification operation [12,13,21,19,3,14,31]. At this stage, the membership degrees in a subset or all of the membership functions of the selected input are calculated. Considering different active rule detection units constitutes the major variation in the reported fuzzy processors. This block is responsible for selecting active rules among the available rules. Although the active rule selector unit makes the functionality of a hardware faster and more efficient, there are many processors and controllers that operate without this important block [20,29,10,9]. In spite of noticeable effect of the rule selection stage on the performance improvement, execution of rule selector unit in most of the existing processors starts late and after fuzzification [12,13,21,19,3,14,36]. In these architectures, non-zero fuzzified data is used in rule exploration for activation. This sequence of operations increases the processing latency especially for the processors in which the fuzzification operation runs sequentially. This delay occurs due to the high degree of the membership computations for all inputs in all corresponding membership functions. The processing latency can be significantly reduced by computing the required membership functions only.

In some fuzzy controllers, detecting active rules is accomplished in parallel with fuzzification. Weiwei et al. [40] proposed a fuzzy processing architecture to control idle speed of a car engine. It uses parallel components for active rule selection and fuzzification. The architecture of their active rule selector is static and has been designed particularly for a specific application which cannot be adapted to run other applications. This selection is performed by comparing the selected input with the most significant bits of a predefined membership function. These parameters are static and hardwired.

Another architectural restriction of the reported work in [12,13,3,22] is the assumption of having not more than two overlapped membership functions. This restriction implies losing generality in hardware which restricts the system design especially when an automatic tool is responsible for rule generation.

The other important disadvantage of the all above described active rule selection architectures is that they are not scalable in terms of the number of system inputs, membership functions and their bit width. While the scalability is an important feature that makes the unit reusable for other applications. The lack of flexibility in the shapes of the membership functions constitute another important limitation of the reported architectures as well.

In some architectures, there is a register file that is used to save some intermediate results which makes it possible to determine whether a membership function is active or not. In the architecture discussed in [12,13], the controller asserts enable port of the register file for saving an active membership function. When a non-zero fuzzified data is detected in the fuzzification unit, the number of active membership functions and their degrees of membership are saved. Then, a rule association matrix retrieves consequent active rules for use in inference and defuzzification units.

Ascia et al. [3] use two fuzzification units to operate in parallel. This makes the degree of membership computation two times faster. The rule analyzer unit searches for the rules with non-zero degree of membership in their antecedents. In that architecture, the rule processing unit retrieves rules from rule memory unit and processes the active rules only, where they are detected by the rule analyzer block.

Ikeda et al. [21] proposed a processor architecture which saves activation status of all membership functions in a single long Zero Flag Register instead of a register file. After sequentially calculating the degrees of membership of all inputs in all corresponding membership functions, the activation status of each rule is then determined using Zero Flag Register and each bit of the Active Rule Register is used to determine whether its corresponding rule is active or not.

2. A novel active rule detection method

In conventional approaches, all of the corresponding membership functions for every input should be calculated first which is one of the most time and energy consuming steps. The rule processing or rule activation detection is performed then according to the achieved degree of membership values. However, there are many non-active membership functions for each input set, for which there is no need to perform any calculations. This fact have been observed by other

Download English Version:

https://daneshyari.com/en/article/391054

Download Persian Version:

https://daneshyari.com/article/391054

Daneshyari.com